Caroline Frida
May 12, 2025
Automation and Workflows

Skincare Recommender Project Writeup Draft

Introduction:

When shopping online, one issue I've often faced is the difficulty of comparing many different products at once. E-commerce sites typically offer basic filters like price and rating, but they fail to address the unique needs of different users—especially when it comes to skincare. This leaves shoppers with an overwhelming amount of information and little guidance on how to sift through it and decide which product is best for their skin type. For example, it can be tough to choose between five sunscreens with the same ratings and similar prices.

The inability to pull up products side by side and compare them based on a variety of factors makes it even harder to make a decision. This problem is especially prevalent in skincare, where shoppers may have different priorities depending on their skin type (oily, dry, or combination). Each skin type has products that are better suited for it, making skincare shopping a very personal experience.

Currently, most e-commerce websites allow you to filter products based on factors like price or rating. However, what sets my tool apart is the inclusion of specific data from shoppers with similar skin types. These shoppers have provided reviews and ratings that I've used to help calculate which products are best suited for different users with different skin types.

I wanted to create a tool that makes this process easier for shoppers by focusing on filters that are truly relevant to skincare. The key filters I've included are product type, price, rating, and skin type. With these, shoppers can easily find products that meet their individual needs, removing the frustration of opening multiple tabs and comparing products manually.

Ultimately, this tool simplifies product comparison and helps shoppers make more informed decisions, ensuring they select products that are truly right for their skin.

Data/Methods:

Data source: ***https://www.kaggle.com/code/aashidutt3/sentiment-analysis-sephora-reviews/notebook***

This dataset was found on Kaggle. It was collected by Python scrapper in March 2023 and contains information about over 8,000 beauty products from the Sephora online store, including product and brand names, prices, ingredients, ratings and other features (product_df.csv). It also contains over 600,000 user reviews of all products in the skincare category. When users review a product, they also include information about whether they would recommend the product as well as information about their own skin type.

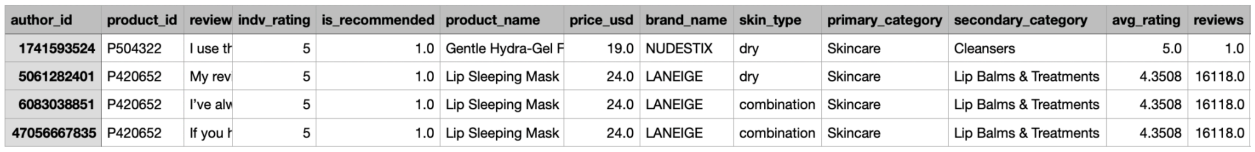
Preprocessing:

This data required a lot of preprocessing as it wasn't already structured in the way I planned to use it for my project. The original product_info.csv contained 24 columns. It had basic product information such as ratings, reviews, sizes, and price. I decided that only half (12) of these variables were useful for my project and dropped the rest. This dataset had over 8,000 rows.

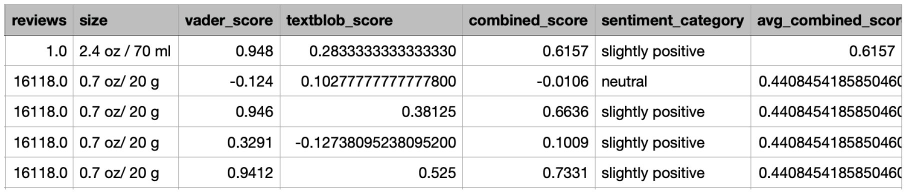
Next, I loaded in the review data. This data was split into 6 different files. For each file only the following columns were kept (['author_id','product_id', 'review_text', 'rating', 'is_recommended','product_name','price_usd','brand_name', 'skin_type']). I also dropped rows where skin_type was missing. Skin_type is the most important variable in my project as it is what connects the user to their most suitable products based on past reviews. If skin_type is missing it makes it difficult to connect users to the best products based on their skin type. In all of the reviews together, 12% of the data was missing skin_type so I found it suitable to drop those rows.

Finally I concatenated the 6 review files into one large file. I then merged the large review file with the product info. I did this by doing a left join on product_id, keeping all of the review data but adding columns from the product information that was necessary for the next steps. These columns included [['product_id', 'primary_category', 'secondary_category', 'rating', 'reviews','size']]. I then renamed the rating column from the combined reviews (rating_x) to be indv_rating, or the individual rating given by a user. I renamed the rating column from the product dataframe (rating_y) to be avg_rating, or the average rating of the product from the product level data. These average ratings had already been computed and were part of the original product information. This is the final processed data shown below.

Final Processed Data:



author_id	product_id	review	indv_rating	is_recommended	product_name	price_usd	brand_name	skin_type	primary_category	secondary_category	avg_rating	reviews
1741593524	P504322	I use th	5	1.0	Gentle Hydra-Gel F	19.0	NUDESTIX	dry	Skincare	Cleansers	5.0	1.0
5061282401	P420652	My rev	5	1.0	Lip Sleeping Mask	24.0	LANEIGE	dry	Skincare	Lip Balms & Treatments	4.3508	16118.0
6083038851	P420652	I've alw	5	1.0	Lip Sleeping Mask	24.0	LANEIGE	combination	Skincare	Lip Balms & Treatments	4.3508	16118.0
47056667835	P420652	If you h	5	1.0	Lip Sleeping Mask	24.0	LANEIGE	combination	Skincare	Lip Balms & Treatments	4.3508	16118.0



reviews	size	vader_score	textblob_score	combined_score	sentiment_category	avg_combined_scor
1.0	2.4 oz / 70 ml	0.948	0.2833333333333330	0.6157	slightly positive	0.6157
16118.0	0.7 oz/ 20 g	-0.124	0.1027777777777800	-0.0106	neutral	0.440845418585046(
16118.0	0.7 oz/ 20 g	0.946	0.38125	0.6636	slightly positive	0.440845418585046(
16118.0	0.7 oz/ 20 g	0.3291	-0.1273809523809520	0.1009	slightly positive	0.440845418585046(
16118.0	0.7 oz/ 20 g	0.9412	0.525	0.7331	slightly positive	0.440845418585046(

Natural Language Processing:

To enhance the recommendation system, I implemented sentiment analysis to evaluate user reviews and provide more insightful product comparisons. The goal was to understand not just the 1-5 ratings, which can be limited and often lack context, but also the emotions and specific experiences behind the feedback from users with similar skincare needs. The first tool I started with was TextBlob.[1] TextBlob is a Python library used for processing textual data. It provides simple NLP tasks such as "speech tagging, noun phrase extraction, and sentiment analysis". TextBlob is also very easy to use. Once the library is downloaded, you can apply it very simply to text. There are several ways to apply it to your text, I found the most suitable way to apply it to mine was to have it give each review a polarity score. The polarity score is a float with range [-1,1] with -1 being very negative, 0 neutral, and 1 as positive.[2] TextBlob will break the text into individual words. Each word

---

[1] *Textblob*. PyPI. (n.d.). https://pypi.org/project/textblob/

[2] *TextBlob Sentiment: Calculating Polarity and Subjectivity*. TextBlob sentiment: Calculating Polarity and subjectivity. (2015, June 7).

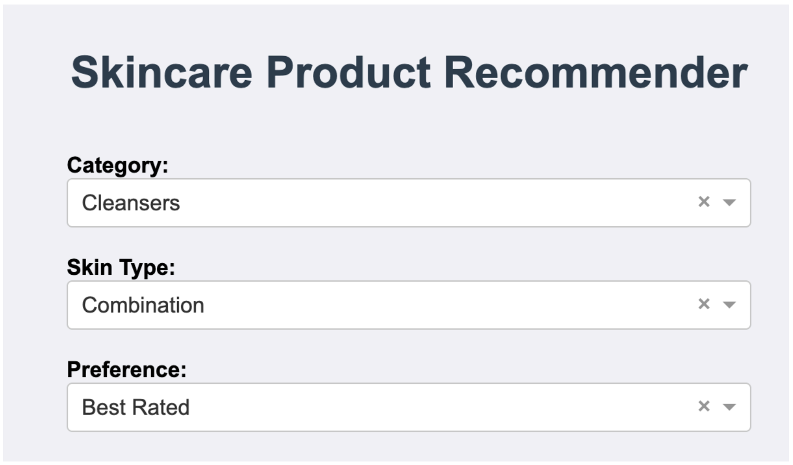
is then matched against a built in sentiment lexicon (from the Pattern library) where words are assigned pre-defined polarity scores. Modifiers (like very) will intensify polarity and negative words (like not) will flip/ reduce the polarity. The individual polarity scores are averaged across the text which produces a single number between -1 and 1.

I also decided to implement VADER. One of the key differences between these two libraries is that VADER is focused on social media.[3] It puts a lot of effort into "identifying sentiments of content that typically appear on social media, such as emojis, repetitive words, and punctuations". Both of these libraries are rule based approaches. This type of approach relies on a predefined set of rules and patterns to identify the sentiment of the text. I decided to choose this over other techniques like machine learning approaches due to the amount of time I had. These models are easy to work with because they do not require any training, they use pre-built lexicons and rules. They also require minimal computational power and are easy to understand. Additionally, VADER is particularly optimized for social media. It is able to handle things like all capital words and slang very well, one of the main reasons I decided to use it on the review text.

When testing the different reviews with these methods I found that each library had its pros and cons and I wanted to choose an approach that balanced accuracy and efficiency. I decided to average these two methods to create a combined score for each review. The final sentiment score ('avg_combined_score_per_product') for each product is an average of all combined scores for each product.

Demonstration:

When using the product recommender dash, users will select a category of skincare from the dropdown. I have selected "Cleansers" here. Next they will select a skin type such as combination (both oily and dry). Finally, they will select a preference, Best Rated or Lowest Price.


Skincare Product Recommender

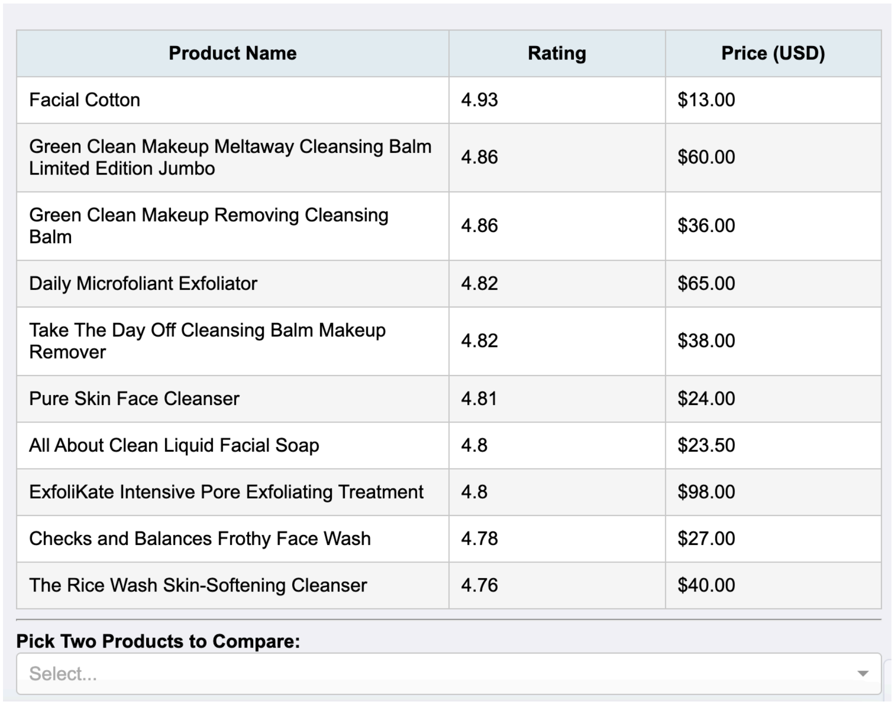
Category:
Cleansers

Skin Type:
Combination

Preference:
Best Rated

Once the user inputs those preferences, the recommender will give the top 10 products based on their selections. An example is shown below.

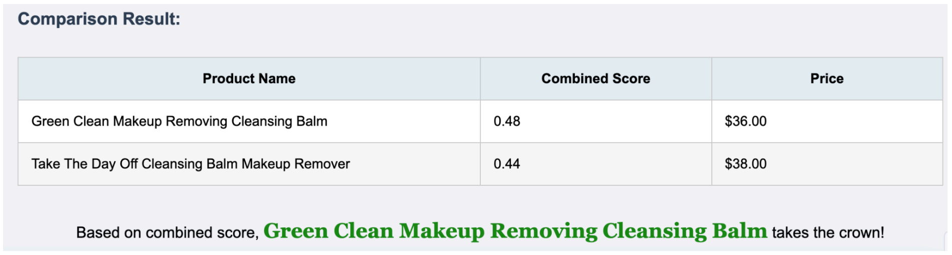
[3] Patil, H. (2023, April 2). *Textblob vs vader for sentiment analysis*. Medium.



Product Name	Rating	Price (USD)
Facial Cotton	4.93	$13.00
Green Clean Makeup Meltaway Cleansing Balm Limited Edition Jumbo	4.86	$60.00
Green Clean Makeup Removing Cleansing Balm	4.86	$36.00
Daily Microfoliant Exfoliator	4.82	$65.00
Take The Day Off Cleansing Balm Makeup Remover	4.82	$38.00
Pure Skin Face Cleanser	4.81	$24.00
All About Clean Liquid Facial Soap	4.8	$23.50
ExfoliKate Intensive Pore Exfoliating Treatment	4.8	$98.00
Checks and Balances Frothy Face Wash	4.78	$27.00
The Rice Wash Skin-Softening Cleanser	4.76	$40.00

Pick Two Products to Compare:

From here the user will select 2 of the top ten products to compare the review sentiment scores. If the user selects 3 products, there will be a warning that asks the users to only select 2 products. An example of the final display is shown below.



Comparison Result:

Product Name	Combined Score	Price
Green Clean Makeup Removing Cleansing Balm	0.48	$36.00
Take The Day Off Cleansing Balm Makeup Remover	0.44	$38.00

Based on combined score, **Green Clean Makeup Removing Cleansing Balm** takes the crown!

If Given More Time:

If I had more time there are two features I would have liked to have. The first is to include a link for the final recommendation so the user could easily navigate to the Sephora website and purchase the product. Second, I would have liked to have a best value preference. I attempted to calculate the price per mL, however there were too many missing sizes. Both of these would have greatly improved the user experience.

Conclusion/ Summary:

Overall, the dashboard provides a personalized skincare recommendation experience by considering a user's skin type, pricing or rating preferences, and review sentiment. It works in two main parts: first, generating a list of top product recommendations based on user inputs and customized ratings from reviewers with the same skin type. The second part allows users to compare their top choices by analyzing sentiment from real user reviews using Natural Language Processing. This combination of user data and sentiment analysis ensures more accurate recommendations.

Bibliography

TextBlob Sentiment: Calculating Polarity and Subjectivity. TextBlob sentiment: Calculating Polarity and subjectivity. (2015, June 7). https://planspace.org/20150607-textblob_sentiment/

Textblob vs vader for sentiment analysis. Patil, H. (2023, April 2). Medium. https://medium.com/@hhpatil001/textblob-vs-vader-for-sentiment-analysis-9d36b0b79ae6#:~:text=A%20critical%20difference%20between%20TextBlob,%2C%20repetitive%20words%2C%20and%20punctuations

Textblob. PyPI. (n.d.). https://pypi.org/project/textblob/