

Final Report of Traineeship Program 2025

On
“Analysis of Chemical Components”

MEDTOUREASY



27th June 2025



ACKNOWLEDGMENTS

The traineeship opportunity that I had with MedTourEasy was a great change for learning and understanding the intricacies of the subject of Data Visualizations in Data Analytics; and also, for personal as well as professional development. I am very obliged for having a chance to interact with so many professionals who guided me throughout the traineeship project and made it a great learning curve for me.

Firstly, I express my deepest gratitude and special thanks to the Training & Development Team of MedTourEasy who gave me an opportunity to carry out my traineeship at their esteemed organization. Also, I express my thanks to the team for making me understand the details of the Data Analytics profile and training me in the same so that I can carry out the project properly and with maximum client satisfaction and also for sparing his valuable time in spite of his busy schedule.

I would also like to thank the team of MedTourEasy and my colleagues who made the working environment productive and very conducive.



TABLE OF CONTENTS

Acknowledgments.....i

Abstract iii

Sr. No.	Topic	Page No.
1	Introduction	
	1.1 About the Company	6
	1.2 About the Project	7
	1.3 Objectives and Deliverables	8
2	Methodology	
	2.1 Flow of the Project	9
	2.2 Use Case Diagram	12
	2.3 Language and Platform Used	13
3	Implementation	
	3.1 Problem Statement	15
	3.2 Data Collection	15
	3.3 Data Cleaning	15
	3.4 Data Filtering	16
	3.5 Tokenization and Corpus Creation	17
	3.6 Vectorization	18
	3.7 Dimensionality Reduction	19
	3.8 Visualization	20
	3.9 Prototyping - Power BI	22
	3.10 Development Of Dashboard	23
4	Sample Screenshots and Observations	
	Code with outputs of project	29
6	Conclusion	
7	Future Scope	
8	References	

ABSTRACT

In the ever-evolving cosmetic industry, consumers are faced with an overwhelming variety of products, each containing numerous chemical ingredients that are often difficult to understand. This complexity poses a significant challenge for individuals with sensitive skin types or specific skincare needs, as interpreting ingredient lists without scientific knowledge can lead to poor product choices, skin irritation, or adverse reactions. To bridge this gap, this project presents a data science-driven solution: a content-based recommendation system that suggests cosmetic products—specifically moisturizers—based on ingredient similarity.

Using a dataset of 1472 cosmetic products from Sephora, the study focuses on moisturizers suitable for dry skin, a common concern among skincare users. The ingredients were extracted and preprocessed through text normalization and tokenization. These ingredient lists were then embedded into vector space using word embedding techniques, enabling the model to understand the semantic relationships between chemical components. To visualize these relationships and discover clusters of similar products, t-distributed stochastic neighbor embedding (t-SNE) was applied for dimensionality reduction.

The resulting 2D representations of ingredient embeddings were visualized using Bokeh, allowing for interactive exploration of ingredient similarities and clusters. This approach not only empowers consumers to identify safer and more effective products but also introduces a scalable method for analyzing ingredient compositions across various product categories. Furthermore, the methodology implemented in this project sets the foundation for a broader recommendation system that can incorporate additional parameters such as user reviews, skin concerns, and dermatologist ratings.

By combining machine learning, natural language processing, and interactive data visualization, this project demonstrates the potential of technology to personalize skincare decisions, reduce trial-and-error purchases, and promote ingredient transparency in the cosmetic industry. It serves as a stepping stone toward smarter, more informed cosmetic product selection.



1.1 About the Company

MedTourEasy, a global healthcare company, provides you the informational resources needed to evaluate your global options. MedTourEasy provides analytical solutions to our partner healthcare providers globally.

1.2 About the Project

Consumers often struggle to interpret the complex chemical ingredients listed on cosmetic product labels, especially when trying to find products suitable for specific skin types like dry or sensitive skin. This project aims to make that decision-making process easier and safer by using data science to analyze, cluster, and visualize cosmetic ingredients across different products. The core objective of this project is to develop a content-based recommendation system for cosmetic products—particularly moisturizers for dry skin—based on ingredient similarity. Using a dataset of 1,472 cosmetics collected from Sephora, we filtered the products based on skin type suitability, focusing specifically on those designed for dry skin. The dataset includes product attributes such as brand, name, price, label category, and a detailed list of ingredients.

The project involves several steps: extracting and preprocessing ingredient data, converting textual ingredients into meaningful numerical representations using natural language processing (NLP) techniques, and reducing high-dimensional data using t-SNE (t-distributed stochastic Neighbor embedding) to uncover clusters of similar ingredients. These clusters are then visualized using interactive Bokeh plots, enabling clear, user-friendly exploration of ingredient-based product relationships. By analyzing ingredient similarities, the system can recommend alternative products with similar formulations, helping users avoid trial-and-error purchases and potential skin reactions. This project provides a foundation for future work in intelligent product recommendation systems in the cosmetics and personal care industry, contributing to transparency, safety, and personalization in skincare decisions.



Hence, this project aims at collecting and analyzing a large dataset of cosmetic products to build an intuitive and interactive ingredient-based recommendation system. It enables the visualization of ingredient similarities to help users make informed skincare decisions. The project is majorly divided into three subsections, as below:

- **Data Preparation and Filtering**
In this phase, a dataset containing 1,472 cosmetic products was collected from Sephora. The data was cleaned and filtered to focus on moisturizers suitable for dry skin. The ingredient lists were extracted, normalized, and tokenized to prepare them for analysis.
- **Ingredient Embedding and Clustering**
The cleaned ingredient text data was converted into vector representations using word embedding techniques. These vectors were then processed using t-SNE, a dimensionality reduction algorithm, to identify and visualize clusters of similar ingredients and products.
- **Visualization and Interactive Recommendation**
Using the Bokeh visualization library, the clustered ingredient data was plotted to form an interactive dashboard. This dashboard allows users to explore cosmetic products visually, based on ingredient similarity, offering personalized product suggestions based on their skin type.



Each of the above sub-sections has been implemented and visualized using Python on the Google Colab platform with the help of libraries such as Pandas, NumPy, scikit-learn, and Bokeh. The dashboards and visualizations were built using Bokeh to create intuitive and interactive plots. These plots allow users to explore clusters of similar ingredients, making it easier to identify safe and suitable cosmetic products. The recommendation engine, backed by natural language processing and machine learning, helps users and businesses analyze product compositions and draw meaningful conclusions.



1.3 Objectives and Deliverables

The primary objective of this project is to develop a content-based recommendation system that simplifies cosmetic product selection based on ingredient similarity, with a specific focus on moisturizers for dry skin. The project aims to use data science techniques to create an interactive visualization that enables users to understand product formulations and make safer, more informed skincare decisions.

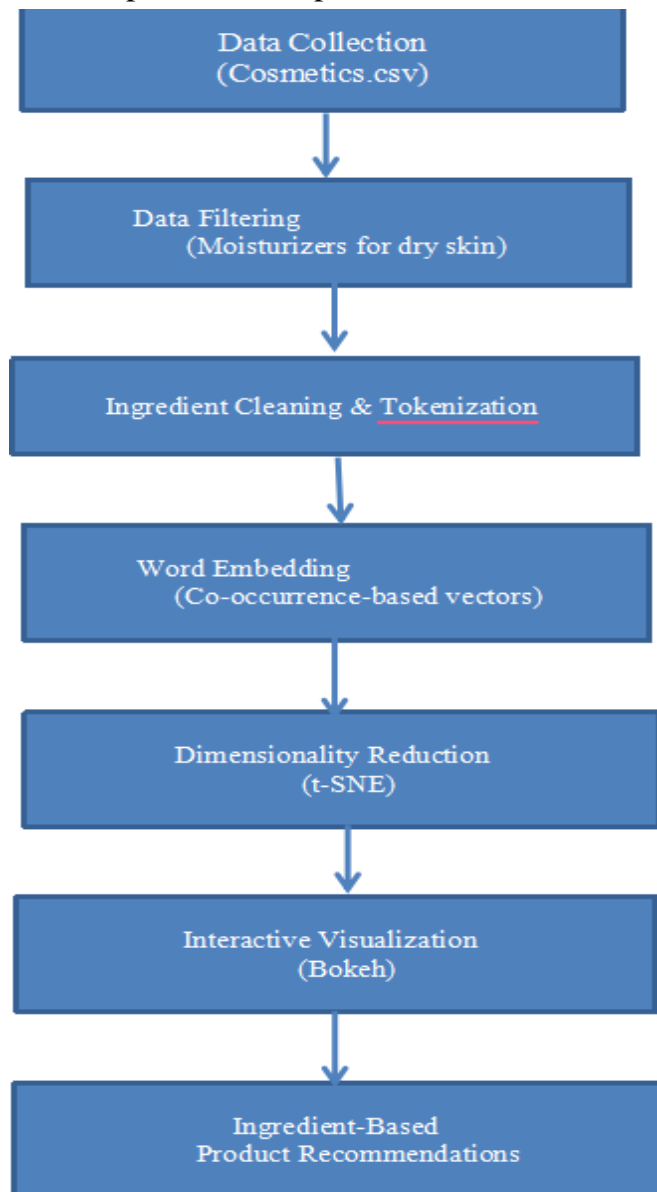
The key deliverables of the project are as follows:

- **Collection and preprocessing** of a real-world dataset containing 1,472 cosmetic products with ingredient lists, brand information, and skin-type suitability.
- **Filtering and categorization** of the dataset to isolate moisturizers recommended for dry skin.
- **Text processing and tokenization** of cosmetic ingredients using Natural Language Processing (NLP) techniques.
- **Vector representation** of ingredients using word embedding to capture semantic relationships between chemical components.
- **Dimensionality reduction** using the t-SNE algorithm to project high-dimensional ingredient vectors into a 2D space for clustering and visualization.
- **Development of interactive visualizations** using Bokeh to explore ingredient similarities and clusters in a user-friendly format.
- **Recommendation of alternative products** based on ingredient similarity to help users avoid harmful formulations and find suitable options.
- **Comprehensive project documentation and final report** summarizing the methodology, tools used, insights derived, and future scope of the system.

I. METHODOLOGY

2.1 Flow of the Project

The project followed the following steps to accomplish the desired objectives and deliverables. Each step has been explained in detail in the following section.





The methodology flowchart outlines the end-to-end process followed in this project, starting from raw data ingestion to generating personalized product recommendations based on ingredient similarity. Each stage in the flowchart plays a critical role in transforming unstructured cosmetic ingredient data into interactive, insightful visualizations and outputs.

1. Data Collection

The process begins with collecting a dataset of **1,472 cosmetic products** from Sephora, which includes product names, brand, price, skin-type suitability, and most importantly, detailed **ingredient lists**.

2. Data Filtering

The dataset is filtered to retain only products categorized as **moisturizers** and specifically marked as **suitable for dry skin**. This ensures that the system is trained and evaluated on a focused segment relevant to the target use case.

3. Ingredient Cleaning & Tokenization

Ingredient lists, initially in raw text form, are **cleaned** (e.g., lowercased, special characters removed) and then **tokenized** by splitting them into individual chemical entities. This prepares the data for further NLP-based transformation.

4. Word Embedding

Each tokenized ingredient is embedded into a **high-dimensional vector space** using **co-occurrence-based word embedding** techniques. This step captures the relationships between ingredients based on how frequently they appear together in different products.

5. Dimensionality Reduction (t-SNE)

Since the embedded ingredient vectors are in high-dimensional space, **t-SNE (t-distributed stochastic neighbor embedding)** is applied to reduce them to **2D**. This makes it possible to visualize the relationships between ingredients and clusters of similar products.

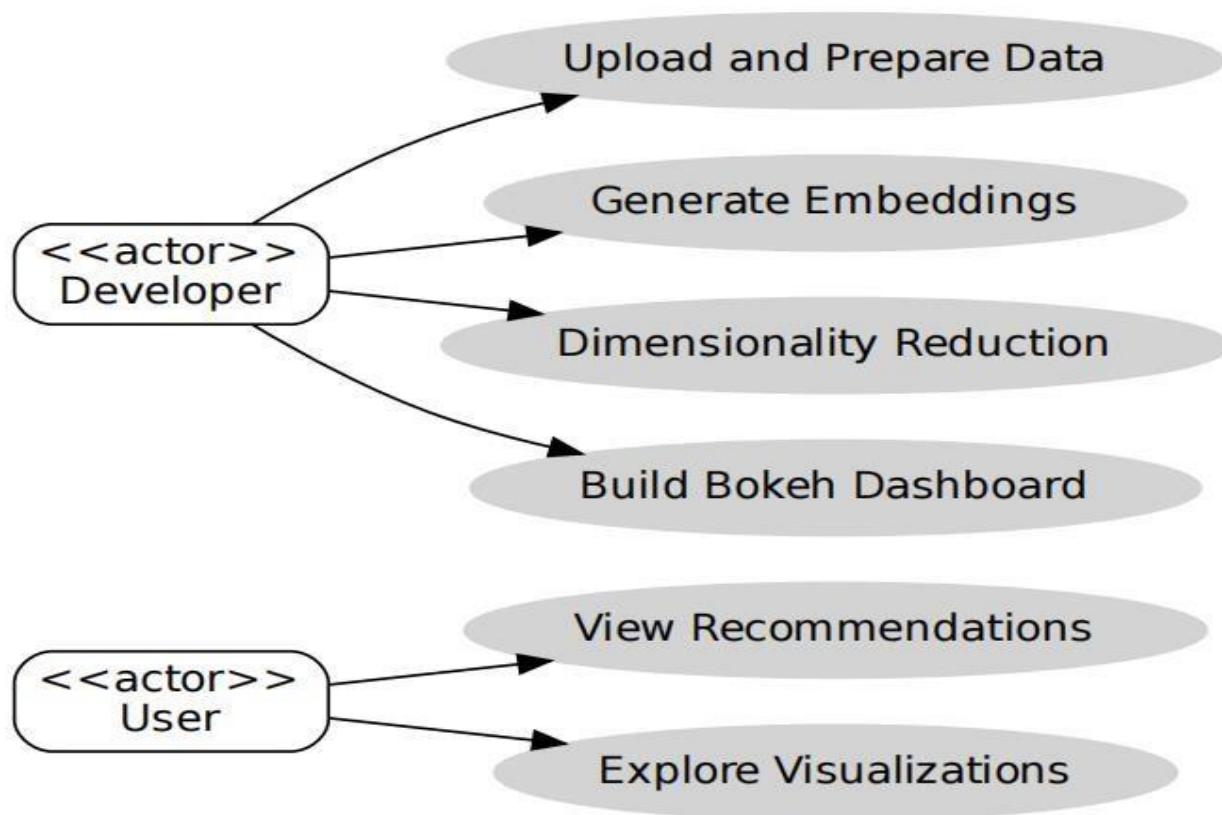
6. Interactive Visualization (Bokeh)

The 2D ingredient clusters are plotted using **Bokeh**, a powerful Python visualization library. This allows the creation of **interactive dashboards** where users can explore products based on ingredient similarity.

7. Product Recommendations

Based on visual proximity and clustering, the system can recommend alternative products with similar ingredient profiles. This helps users make safer, more informed skincare decisions.

2.2 Use Case Diagram



The UML Use Case Diagram represents the primary interactions between users and system functionalities in the **Analysis of Chemical Component**. It identifies two main actors: the **User** and the **Developer**, each performing distinct sets of operations within the system.

The diagram emphasizes the inclusion of clustering and dashboard generation as integral backend tasks, while filtering and exploration represent front-end interactive use cases that enhance user decision-making.



2.3 Language and Platform Used

2.3.1 Language: Python

Python is a high-level, general-purpose programming language known for its readability, simplicity, and extensive support for data science and machine learning applications. Developed by **Guido van Rossum**, Python has become a widely adopted language in the analytics and AI community due to its strong ecosystem of libraries and frameworks.

In this project, Python was used for all core development tasks including data preprocessing, natural language processing, dimensionality reduction, and interactive visualization.

Key features of Python relevant to this project:

Simple and readable syntax with support for functional and object-oriented programming

Vast collection of open-source libraries like **pandas**, **scikit-learn**, **NumPy**, and **Bokeh**

Built-in support for machine learning, visualization, and NLP tasks

Easy integration with notebooks, dashboards, and web applications

2.3.2 Platform: Google Colaboratory (Colab)

Google Colab is a cloud-based, Jupyter Notebook environment provided by Google. It enables users to write and execute Python code in a browser without requiring local setup. It was used in this project as the primary development and experimentation environment.

Major features of Google Colab used in this project:

Cloud-hosted Python notebooks with **zero setup**

Pre-installed support for essential libraries (pandas, matplotlib, scikit-learn, etc.)

Real-time execution of code with inline output (tables, charts, plots)

Integration with **Google Drive** for file storage and collaboration

Access to **free GPUs** (optional) for faster computations

Seamless support for markdown + code explanations for reporting



2.3.2 Package and Libraries used

This project was implemented using Python, supported by a range of powerful libraries and packages that enabled data preprocessing, natural language processing, dimensionality reduction, and visualization. The key libraries used are listed below:

1. pandas

Used for data loading, cleaning, filtering, and manipulation of the `cosmetics.csv` dataset.

Enabled operations like column extraction, tokenization, and DataFrame transformation.

Version Used: `pandas >= 1.3`

2. NumPy

Used for numerical operations and efficient handling of arrays and matrices during preprocessing and vectorization.

Version Used: `numpy >= 1.21`

3. scikit-learn (sklearn)

Provided tools for:

CountVectorizer: to convert text (ingredient list) into numerical vectors using bag-of-words model

t-SNE: for dimensionality reduction and clustering of similar products based on ingredients

Version Used: `scikit-learn >= 1.0`

4. Bokeh

Used to build an **interactive visualization** of the t-SNE output.

Allowed users to explore clusters of products with tooltips showing product name and brand.

Version Used: `bokeh >= 2.4`

5. matplotlib (optional)

Used in initial stages to explore data distribution and verify clusters (if used during EDA).

6. Google Colab

Not a package but the environment used to run all code, with built-in support for the above packages.

These libraries, when combined, allowed the system to analyze cosmetic product ingredients, generate a meaningful 2D projection of similarities, and present the results through an interactive user interface.



II. IMPLEMENTATION

3.1. Problem Statement

In the modern skincare industry, consumers face a major challenge in interpreting cosmetic product labels. Ingredient lists are often long, technical, and filled with unfamiliar chemical names. For individuals with specific skin concerns especially **dry skin** choosing the right product can be a time-consuming and confusing process. There is a lack of transparent, ingredient-level insights and personalized product suggestions based on actual chemical compositions.

3.2. Data Collection

The dataset used in this project was sourced from **Sephora**, a leading cosmetics retailer. The dataset, titled `cosmetics.csv`, contains **1,472 cosmetic products** with the following features:

Column Name	Description
Name	Name of the cosmetic product
Brand	Brand/manufacturer
Label	Type of product (e.g., Moisturizer, Cleanser)
Price	Product price in USD
Ingredients	Full list of ingredients in comma-separated format
Dry, Oily, Combo, Sensitive	Binary indicators of suitability for different skin types

For this project, the dataset was filtered to retain only products categorized as **Moisturizers** and specifically suited for **Dry skin**.

3.3. Data Cleaning

To ensure accurate ingredient analysis, the dataset underwent several cleaning steps:

- Standardized all ingredient text to **lowercase**
- Removed punctuation and unwanted characters
- **Tokenized** ingredient lists into individual ingredients
- Created a new **corpus column** suitable for vectorization



Data Collection & Cleaning

Step 1: Load the datasetimport pandas as pd

```
df = pd.read_csv("cosmetics.csv")
```

Step 2: Filter moisturizers suitable for dry skin

```
moisturizers = df[(df["Label"] == "Moisturizer") & (df["Dry"] == 1)].reset_index(drop=True)
```

python

Step 3: Clean and tokenize ingredients# Convert to lowercase and split by commas

```
moisturizers["tokens"] = moisturizers["Ingredients"].str.lower().str.split(", ")
```

Step 4: Prepare corpus for NLP processing# Join tokens into a single space-separated string

```
moisturizers["corpus"] = moisturizers["tokens"].apply(lambda x: " ".join(x))
```

Output After Cleaning (Sample)

Product Name	Brand	Token Count	Example Tokens
Ultra Facial Cream	Kiehl's	12	["water", "glycerin", "squalane"]
Hydration Boost Gel	Neutrogena	14	["aqua", "dimethicone", "phenoxyethanol"]

3.4. Data Filtering

After collecting and loading the dataset, the next step was to filter it based on the project's objective: focusing on moisturizer products that are suitable for dry skin.

The original dataset (cosmetics.csv) contained 1,472 product entries from a variety of brands, covering different categories like cleansers, toners, and moisturizers. However, for this project, only products classified as "Moisturizers" and marked suitable for "Dry" skin were relevant.

Sample Code:

```
# Filter for Moisturizers suitable for dry skin
```

```
moisturizers = df[(df["Label"] == "Moisturizer") & (df["Dry"] == 1)].reset_index(drop=True)
```




3.5. Tokenization & Corpus Creation

Each product's ingredient list was:

- Lowercased for standardization
- Tokenized by splitting the string on comma

This transformed each ingredient list into a list of individual components.

The tokenized ingredients were then joined into a single space-separated string for each product. This corpus was required for the next step: text vectorization.

Sample code:

```
# Join tokenized ingredients into single space-separated strings
moisturizers["corpus"] = moisturizers["tokens"].apply(lambda x: " ".join(x))
```

Prepares the ingredient text for NLP vectorization by creating a clean corpus.

3.6. Vectorization (Bag-of-Words)

The CountVectorizer from scikit-learn was used to convert the ingredient text into numerical vectors. Each row in the resulting matrix represents a product, and each column corresponds to a unique ingredient.

Sample Code:

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(moisturizers["corpus"])
```

Transforms the corpus into a numerical format using CountVectorizer, enabling machine learning models to process the text.



3.7. Dimensionality Reduction (t-SNE)

To visualize ingredient similarities, the high-dimensional vector space was reduced to two dimensions using t-SNE (t-Distributed Stochastic Neighbor Embedding). This technique preserves the relative distances between points, which is essential for visual cluster interpretation.

Sample code:

```
from sklearn.manifold import TSNE

tsne = TSNE(n_components=2, random_state=42, perplexity=30)
X_embedded = tsne.fit_transform(X.toarray())
```

Reduces high-dimensional ingredient vectors to 2D coordinates for clustering and visualization.

3.8. Visualization (Interactive Plot using Bokeh)

The 2D clustered data was visualized using Bokeh, an interactive visualization library. A scatter plot was created where each point represents a product. Hover tools were added to display the product's name and brand on mouseover.

Sample Code:

```
from bokeh.plotting import figure, show, output_notebook
from bokeh.models import ColumnDataSource, HoverTool

output_notebook()

source = ColumnDataSource(data=dict(
    x=X_embedded[:, 0],
    y=X_embedded[:, 1],
    brand=moisturizers["Brand"],
    name=moisturizers["Name"]
))
```

```
plot = figure(title="Cosmetic Ingredient Clusters",
tools="pan,wheel_zoom,box_zoom,reset,hover")
plot.circle('x', 'y', size=10, source=source)
plot.add_tools(HoverTool(tooltips=[("Brand", "@brand"), ("Product", "@name")]))
show(plot)
```



3.9. Prototyping – Power BI

A prototype is an early version, model, or release of a product that is constructed to test a design or process. It is generally used by system analysts and users to assess a new design to enhance precision. Prototyping serves to specify a real, working system rather than a theoretical one. Creation of a prototype in some design workflow models is the step between formalizing and testing an idea.

Power BI is Microsoft's business analytics software. It aims to provide interactive visualizations and business intelligence capabilities with an interface that is easy enough to create your own reports and dashboards for end users. It provides cloud- based BI services, known as "Power BI Services," along with the "Power BI Desktop" desktop-based interface. It provides capabilities for data warehouse, including data planning, data discovery and interactive dashboards. It has the following features:

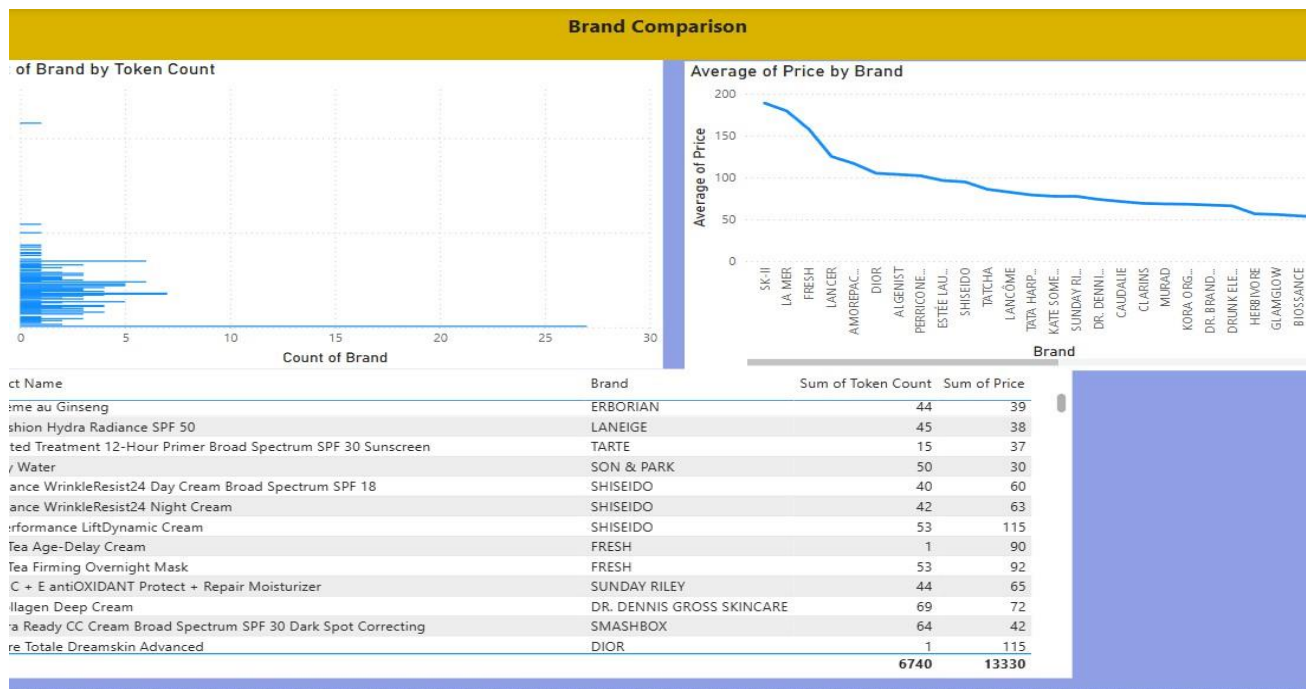
- Easy to connect, model, and visualize data, creating memorable reports personalized with KPIs and brand.
- Can generate fast, AI-powered answers to business questions
- Data is better secured across Power BI reports, dashboards, and data sets with persistent protection that keeps working even when shared outside the organization or exported to other formats such as Excel, PowerPoint, and PDF.
- Input: In the form of Excel, CSV, text, SQL and other formats
- Visualizations: Wide variety of graphs, infographics, KPIs, Filters, Slicers, etc.
- Output: Easily publishable reports and dashboards



3.10. Development of Dashboards

The prototyping phase focused on creating a practical, interactive version of the **Skincare Analyzer Dashboard** to validate the system's logic and explore real-time analytics. The prototype served as a bridge between raw ingredient-level data and an end-user-friendly visualization platform.

- Load and visualize cosmetic product data filtered for dry skin
- Offer dynamic filters for brands, product types, ingredient count, and specific keywords
- Provide dashboards that are easy to navigate for end-users without technical knowledge
- Compare products and brands using visuals like cards, pie charts, scatter plots, and tables





Ingredient Explorer			
Ingredients			
<input type="checkbox"/> Advanced Night Rpr Int Rcv Ampoule Division: El (Estee Lauder)Ingredients: Caprylic/Capric Triglyceride , Squalane , Aleurites Moluccana (Kukui) Seed Oil , Prunus Armeniaca (...)			
<input type="checkbox"/> Algae (Seaweed) Extract, Cyclopentasiloxane, Petrolatum, Glyceryl Distearate, Phenyl Trimethicone, Butylene Glycol, Hydrogenated Vegetable Oil, Cholesterol, Butyrospermum ...			
<input type="checkbox"/> Algae (Seaweed) Extract, Mineral Oil, Petrolatum, Glycerin, Isohexadecane, Microcrystalline Wax, Lanolin Alcohol, Citrus Aurantifolia (Lime) Extract, Sesamum Indicum (Sesame)...			
<input type="checkbox"/> Algae (Seaweed) Extract, Petrolatum, Isocetyl Stearoyl Stearate, Butyrospermum Parkii (Shea Butter), Butylene Glycol, Cyclopentasiloxane, Dimethicone, Glyceryl Stearate, Cetyl...			
<input type="checkbox"/> Almond Oil, Jojoba Oil, Macadamia Oil, Brazil Nut Oil, Evening Primrose Oil, Buriti Oil, Arnica Oil, Rosehip Oil, Calendula Oil, Kiwi Fruit Seed Oil, Tocopherol (Vitamin E), Totorol,...			
<input checked="" type="checkbox"/> Argan Oil Isostearyl Esters**, **Natural.			
<input type="checkbox"/> Avobenzone 3.0%, Homosalate 5.0%, Octisalate 4.5%, Octocrylene 2.8%Water, Butyloctyl Salicylate, Nylon-12, Behenyl Alcohol, Octyldodecyl Neopentanoate, Myristyl Myrista...			
<input type="checkbox"/> Avobenzone 3.0%, Homosalate 8.0%, Octinoxate 7.5%, Octisalate 4.5%, Octocrylene 5.0%Water, Butylene Glycol, Glycerin, Ammonium Acryloyldimethyltaurate/Vp Copolymer,...			
<input type="checkbox"/> Beste™ No.9 Jelly Cleanser: Water, Sodium Lauroyl Methyl Isethionate, Glycerin, Cocamidopropyl Betaine, Cocamidopropyl Hydroxysultaine, Sodium Methyl Oleoyl Taurate, Pr...			
Product Name	Ingredients	Sum of Token Count	Sum of Price
100 percent Pure Argan Oil Light	Argan Oil Isostearyl Esters**, **Natural.	1	48
Total		1	48

3.10.1 Defining Visuals

The visuals in the **Skincare Analyzer Dashboard** were carefully selected to represent relevant metrics, enable dynamic filtering, and simplify product comparisons. Each visual component was mapped to specific fields in the dataset and styled to fit a consistent dark-themed interface, improving readability and user engagement.



1. Cards

- **Purpose:** Display key summary metrics at a glance

2. Pie Chart

- **Purpose:** Visualize the distribution of products by brand.
- **Field:** Brand (Legend)
- **Values:** Product Name (Count)
- Helps in identifying the most represented brands in the filtered view.

3. Donut Chart

- **Purpose:** Illustrate the distribution of product types (e.g., Moisturizer).
- **Field:** Product Type
- **Values:** Product Name (Count)
- Useful for understanding product segmentation within the dataset.

4. Scatter Plot

- **Purpose:** Show the relationship between **ingredient complexity** and **price**.
- **X-axis:** Token Count (number of ingredients)
- **Y-axis:** Price
- **Legend:** Brand
- Enables users to identify outliers, e.g., expensive products with few ingredients.



5. Histogram (Clustered Column Chart)

- **Purpose:** Display the frequency distribution of ingredient counts.
- **Field:** Token Count

- **Values:** Product Name (Count)
- Useful for analyzing how simple or complex the products are in general.

6. Clustered Bar Chart

- **Purpose:** Compare average ingredient complexity across brands.
- **Axis:** Brand
- **Values:** Average of Token Count
- Highlights which brands tend to use simpler or more complex formulations.

7. Table

- **Purpose:** Present detailed product-level data in tabular form.
- **Columns:** Product Name, Brand, Token Count, Price, Ingredients
- Enables users to scan and compare specific products side-by-side.

8. Slicers (Filters)

- **Purpose:** Provide interactivity and data control to the user.

Slicer Field	Function
Brand	Filter visuals by selected brand(s)
Product Name	Narrow down to specific product(s)
Token Count	Filter by complexity range
Ingredients	Text-based search across ingredient lists

Together, these visuals transform static product data into an **interactive analytical tool**, enabling users to explore, compare, and discover skincare products based on meaningful criteria such as ingredient count, price, and brand performance.



3.10.2. Interactive Filtering Implementation

To enhance usability, **Slicers** were used in Power BI. These slicers act as interactive filters that dynamically adjust dashboard visuals. For example:

A **Brand slicer** allows users to select one or more brands and instantly see filtered cards, charts, and tables.

A **Product Name slicer** helps narrow down results to a single product.

A **Token Count slicer** lets users explore products based on the number of ingredients they contain.

An **Ingredients slicer** supports text-based search and discovery.

These filters were styled as dropdowns to save space and configured with search functionality where applicable (especially for ingredients).



III. SAMPLE SCREENSHOTS AND OBSERVATIONS

1. Cosmetics, chemicals... it's complicated

```
# Import libraries
import pandas as pd
import numpy as np
from sklearn.manifold import TSNE
# Load the data
df = pd.read_csv("/content/cosmetics.csv")

# Check the first five rows
display(df.sample(5))
# Inspect the types of products
df.Label.value_counts()
```

	Label	Brand	Name	Price	Rank	Ingredients	Combination	Dry	Normal	Oily	Sensitive
530	Cleanser	KLORANE	Waterproof Eye Make-Up Remover with Soothing C...	16	4.5	Water, Isododecane, Caprylic/Capric Triglyceri...		0	0	0	0
546	Cleanser	KATE SOMERVILLE	Gentle Daily Wash	36	4.2	Water, Sodium C14-16 Olefin Sulfonate, Cocamid...		0	0	0	0
147	Moisturizer	SK-II	GenOptics Spot Essence Serum	225	4.2	Water, Galactomyces Ferment Filtrate*, Butylen...		1	1	1	1
698	Treatment	FOREO	Espada Acne-Clearing Blue Light Pen	149	3.9	Visit the Foreo boutique		1	1	1	1
38	Moisturizer	SK-II	Facial Treatment Essence Mini	99	4.1	Galactomyces Ferment Filtrate (Pitera), Butyle...		1	1	1	1

count

Label

Moisturizer 298

Cleanser 281

Face Mask 266

✓ 6:41 PM



2. Focus on one product category and one skin type

There are six categories of product in our data (moisturizers, cleansers, face masks, eye creams, and sun protection) and there are five different skin types (combination, dry, normal, oily and sensitive). Because individuals have different product needs as well as different skin types, let's set up our workflow so its outputs (a t-SNE model and a visualization of that model) can be customized. For the example in this notebook, let's focus in on moisturizers for those with dry skin by filtering the data accordingly.

[+ Code](#)[+ Text](#)

```
[ ] # Filter for moisturizers
moisturizers = df[df["Label"]=="Moisturizer"]

# Filter for dry skin as well
moisturizers_dry = moisturizers[moisturizers["Dry"]==1]

# Reset index
moisturizers_dry =moisturizers_dry.reset_index(drop=True)
moisturizers
```

	Label	Brand	Name	Price	Rank	Ingredients	Combination	Dry	Normal	Oily	Sensitive
0	Moisturizer	LA MER	Crème de la Mer	175	4.1	Algae (Seaweed) Extract, Mineral Oil, Petrolat...	1	1	1	1	1
1	Moisturizer	SK-II	Facial Treatment Essence	179	4.1	Galactomyces Ferment Filtrate (Pitera), Butyle...	1	1	1	1	1
2	Moisturizer	DRUNK ELEPHANT	Protini™ Polypeptide Cream	68	4.4	Water, Dicaprylyl Carbonate, Glycerin, Ceteary...	1	1	1	1	0
3	Moisturizer	LA MER	The Moisturizing Soft Cream	175	3.8	Algae (Seaweed) Extract, Cyclopentasiloxane, P...	1	1	1	1	1
4	Moisturizer	IT COSMETICS	Your Skin But Better™ CC+™ Cream with SPF 50+	38	4.1	Water, Snail Secretion Filtrate, Phenyl Trimet...	1	1	1	1	1
...
293	Moisturizer	LA MER	The Moisturizing Matte Lotion	270	3.9	Water, Algae (Seaweed) Extract, Propanediol, S...	0	0	1	1	0
294	Moisturizer	HERBIVORE	Jasmine Green Tea Balancing Toner	39	4.2	Jasminum Officinale (Jasmine) Flower Water, Ha...	1	0	0	1	1
295	Moisturizer	CLARINS	Super Restorative Night Age Spot Correcting Re...	136	4.5	Water, Cetearyl Isononanoate, Dimethicone, Gly...	0	0	0	0	0
296	Moisturizer	KATE SOMERVILLE	Goat Milk Moisturizing Cream	65	4.1	Water, Ethylhexyl Palmitate, Myristyl Myristat...	1	1	1	1	1
297	Moisturizer	GO-TO	Face Hero	34	4.8	Almond Oil, Jojoba Oil, Macadamia Oil, Brazil ...	1	1	1	1	1

298 rows × 11 columns



3. Tokenizing the ingredients

To get to our end goal of comparing ingredients in each product, we first need to do some preprocessing tasks and bookkeeping of the actual words in each product's ingredients list. The first step will be tokenizing the list of ingredients in `Ingredients` column. After splitting them into tokens, we'll make a binary bag of words. Then we will create a dictionary with the tokens, `ingredient_idx`, which will have the following format:

{ "ingredient": index value, ... }

```
[ ] # Initialize dictionary, list, and initial index
ingredient_idx = {}
corpus = []
idx = 0

# For loop for tokenization
for i in range(len(moisturizers_dry)):
    ingredients = moisturizers_dry['Ingredients'][i]
    ingredients_lower = ingredients.lower()
    tokens = ingredients_lower.split(',')
    corpus.append(tokens)
    for ingredient in tokens:
        if ingredient not in ingredient_idx:
            ingredient_idx[ingredient] = idx
            idx += 1

# Check the result
print("The index for decyl oleate is", ingredient_idx['decyl oleate'])
```

The index for decyl oleate is 25

4. Initializing a document-term matrix (DTM)

The next step is making a document-term matrix (DTM). Here each cosmetic product will correspond to a document, and each chemical composition will correspond to a term. This means we can think of the matrix as a "cosmetic-ingredient" matrix. The size of the matrix should be as the picture shown below.

The diagram shows a matrix with rows labeled 'Cosmetic 1', 'Cosmetic 2', 'Cosmetic 3', and 'Cosmetic M' on the left, and columns labeled 'Ingredient 1', 'Ingredient 2', 'Ingredient 3', 'Ingredient 4', and 'Ingredient N' at the top. A bracket on the left side of the rows is labeled '# of the items'. A bracket at the bottom of the columns is labeled '# of the ingredients'.

	Ingredient 1	Ingredient 2	Ingredient 3	Ingredient 4	...	Ingredient N
Cosmetic 1						
Cosmetic 2						
Cosmetic 3						
⋮						
Cosmetic M						

To create this matrix, we'll first make an empty matrix filled with zeros. The length of the matrix is the total number of cosmetic products in the data. The width of the matrix is the total number of ingredients. After initializing this empty matrix, we'll fill it in the following tasks.

```
[ ] # Get the number of items and tokens
M = len(moisturizers_dry)
N = len(ingredient_idx)

# Initialize a matrix of zeros
A = np.zeros([M,N])
```

5. Creating a counter function

Before we can fill the matrix, let's create a function to count the tokens (i.e., an ingredients list) for each row. Our end goal is to fill the matrix with 1 or 0: if an ingredient is in a cosmetic, the value is 1. If not, it remains 0. The name of this function, `oh_encoder`, will become clear next.

```
[ ] # Define the oh_encoder function
def oh_encoder(tokens):
    x = np.zeros(N)
    for ingredient in tokens:
        # Get the index for each ingredient
        idx = ingredient_idx[ingredient]
        # Put 1 at the corresponding indices
        x[idx] = 1
    return x
```

6. The Cosmetic-Ingredient matrix!

Now we'll apply the `oh_encoder()` function to the tokens in `corpus` and set the values at each row of this matrix. So the result will tell us what ingredients each item is composed of. For example, if a cosmetic item contains *water*, *niacin*, *decyl oleate* and *sh-polypeptide-1*, the outcome of this item will be as follows.

	Ingredient 1	Ingredient 2	Ingredient 3	Ingredient 4	...	Ingredient N
Cosmetic 1						
Cosmetic 2						
Cosmetic 3						
...						
Cosmetic M						

"Cosmetic - Ingredient" matrix



oh_encoder()

	water	niacin	lanolin alcohol	Decyl oleate	serine	...	sh-polypeptide-1
Cosmetic 1	1	1	0	1	0	...	1

- **Ingredients:** *niacin*, *water*,
decyl oleate,
sh-polypeptide-1



```
[ ] # Make a document-term matrix
i = 0
for tokens in corpus:
    A[i, :] = oh_encoder(tokens)
    i+=1
```

7. Dimension reduction with t-SNE

The dimensions of the existing matrix is (190, 2233), which means there are 2233 features in our data. For visualization, we should downsize this into two dimensions. We'll use t-SNE for reducing the dimension of the data here.

[T-distributed Stochastic Neighbor Embedding \(t-SNE\)](#) is a nonlinear dimensionality reduction technique that is well-suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions. Specifically, this technique can reduce the dimension of data while keeping the similarities between the instances. This enables us to make a plot on the coordinate plane, which can be said as vectorizing. All of these cosmetic items in our data will be vectorized into two-dimensional coordinates, and the distances between the points will indicate the similarities between the items.

```
# Dimension reduction with t-SNE
model = TSNE(n_components=2, learning_rate=200, random_state=42)
tsne_features = model.fit_transform(A)

# Make X, Y columns
moisturizers_dry['X'] = tsne_features[:,0]
moisturizers_dry['Y'] = tsne_features[:,1]
```


8. Let's map the items with Bokeh

We are now ready to start creating our plot. With the t-SNE values, we can plot all our items on the coordinate plane. And the coolest part here is that it will also show us the name, the brand, the price and the rank of each item. Let's make a scatter plot using Bokeh and add a hover tool to show that information. Note that we won't display the plot yet as we will make some more additions to it.

```
from bokeh.io import show, output_notebook, push_notebook
from bokeh.plotting import figure
from bokeh.models import ColumnDataSource, HoverTool
output_notebook()

# Make a source and a scatter plot
source = ColumnDataSource(moisturizers_dry)
plot = figure(x_axis_label = "T-SNE 1",
              y_axis_label = "T-SNE 2",
              width = 500, height = 400)

plot.circle(x = "X",
            y = "Y",
            source = source,
            size = 10, color = '#FF7373', alpha = .8)
```

 BokehDeprecationWarning: 'circle() method with size value' was deprecated in Bokeh 3.4.0 and will be removed, use 'scatter(size=...)' instead. GlyphRenderer(id='p1094', ...)

9. Adding a hover tool

Why don't we add a hover tool? Adding a hover tool allows us to check the information of each item whenever the cursor is directly over a glyph. We'll add tooltips with each product's name, brand, price, and rank (i.e., rating).

```
[ ] # Create a HoverTool object
hover = HoverTool(tooltips = [('Item', '@Name'),
                              ('Brand', '@Brand'),
                              ('Price', '$@Price'),
                              ('Rank', '@Rank')])

plot.add_tools(hover)
```



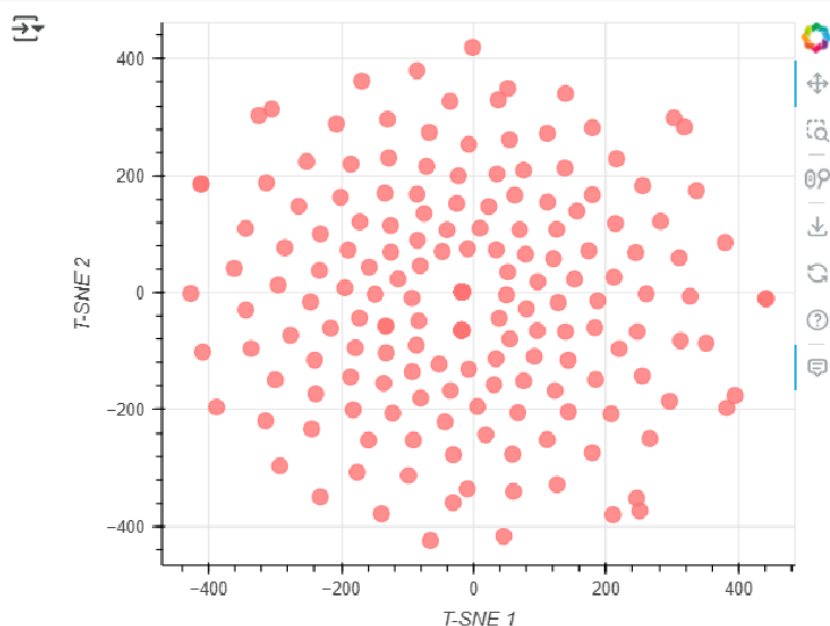
10. Mapping the cosmetic items

Finally, it's show time! Let's see how the map we've made looks like. Each point on the plot corresponds to the cosmetic items. Then what do the axes mean here? The axes of a t-SNE plot aren't easily interpretable in terms of the original data. Like mentioned above, t-SNE is a visualizing technique to plot high-dimensional data in a low-dimensional space. Therefore, it's not desirable to interpret a t-SNE plot quantitatively.

Instead, what we can get from this map is the distance between the points (which items are close and which are far apart). The closer the distance between the two items is, the more similar the composition they have. Therefore this enables us to compare the items without having any chemistry background.

Double-click (or enter) to edit

```
[ ] # Plot the map  
show(plot)
```





11. Comparing two products

Since there are so many cosmetics and so many ingredients, the plot doesn't have many super obvious patterns that simpler t-SNE plots can have ([example](#)). Our plot requires some digging to find insights, but that's okay!

Say we enjoyed a specific product, there's an increased chance we'd enjoy another product that is similar in chemical composition. Say we enjoyed AmorePacific's [Color Control Cushion Compact Broad Spectrum SPF 50+](#). We could find this product on the plot and see if a similar product(s) exist. And it turns out it does! If we look at the points furthest left on the plot, we see LANEIGE's [BB Cushion Hydra Radiance SPF 50](#) essentially overlaps with the AmorePacific product. By looking at the ingredients, we can visually confirm the compositions of the products are similar (*though it is difficult to do, which is why we did this analysis in the first place!*), plus LANEIGE's version is \$22 cheaper and actually has higher ratings.

It's not perfect, but it's useful. In real life, we can actually use our little ingredient-based recommendation engine help us make educated cosmetic purchase choices.

```
# Print the ingredients of two similar cosmetics
cosmetic_1 = moisturizers_dry[moisturizers_dry['Name'] == "Color Control Cushion Compact Broad Spectrum SPF 50+"]
cosmetic_2 = moisturizers_dry[moisturizers_dry['Name'] == "BB Cushion Hydra Radiance SPF 50"]

# Display each item's data and ingredients
display(cosmetic_1)
print(cosmetic_1.Ingredients.values)
display(cosmetic_2)
print(cosmetic_2.Ingredients.values)
```

	Label	Brand	Name	Price	Rank	Ingredients	Combination	Dry	Normal	Oily	Sensitive	X	Y
45	Moisturizer	AMOREPACIFIC	Color Control Cushion Compact Broad Spectrum S...	60	4.0	Phyllostachis Bambusoides Juice, Cyclopentasil...		1	1	1	1	-9.419198	-335.968231
['Phyllostachis Bambusoides Juice, Cyclopentasiloxane, Cyclohexasiloxane, Peg-10 Dimethicone, Phenyl Trimethicone, Butylene Glycol, Butylene Glycol Dicaprylat													
	Label	Brand	Name	Price	Rank	Ingredients	Combination	Dry	Normal	Oily	Sensitive	X	Y
55	Moisturizer	LANEIGE	BB Cushion Hydra Radiance SPF 50	38	4.3	Water, Cyclopentasiloxane, Zinc Oxide (CI 7794...		1	1	1	1	-31.170496	-359.229309
['Water, Cyclopentasiloxane, Zinc Oxide (CI 77947), Ethylhexyl Methoxycinnamate, PEG-10 Dimethicone, Cyclohexasiloxane, Phenyl Trimethicone, Iron Oxides (CI 7													



IV. CONCLUSION AND FUTURE SCOPE

This project successfully demonstrates the application of data science techniques to the cosmetic domain by developing a content-based recommendation system focused on **moisturizers for dry skin**. By leveraging a structured dataset of cosmetic products, we were able to preprocess ingredient data, convert it into numerical embeddings using NLP, and apply **t-SNE** to reduce high-dimensional data into a visually interpretable 2D space.

Using **Bokeh**, we created an interactive visualization that allows users to explore and compare products based on ingredient similarity rather than brand popularity or user reviews. This approach empowers consumers to make more informed choices, especially those with sensitive or specific skincare needs.

The system provides a strong foundation for ingredient transparency, safer skincare recommendations, and opens a new dimension in how cosmetic products are searched, compared, and selected.

Future Scope

While the current system is effective as a prototype, several enhancements can be made to improve its functionality, scalability, and user experience:

1. Multi-Skin Type Expansion

Extend the filtering to support oily, combination, and sensitive skin types alongside dry skin.

2. Sentiment Analysis

Incorporate customer reviews and ratings to factor in **user sentiment** and real-world effectiveness.

3. Advanced Recommendation Logic

Upgrade the system with **hybrid recommendation algorithms** combining both content-based and collaborative filtering.

4. Ingredient Safety Insights

Integrate safety ratings and allergy indicators from trusted dermatological sources to flag harmful or irritating ingredients.

5. Real-Time Web App Deployment

Convert the project into a live **web-based dashboard** using tools like **Flask** or **Streamlit**, allowing users to upload their favorite product and get real-time alternatives.



V. REFERENCES

- Sephora: <https://www.sephora.com>
- pandas: <https://pandas.pydata.org>
- scikit-learn: <https://scikit-learn.org>
- bokeh: <https://bokeh.org>
- Google Colab: <https://colab.research.google.com>