Skincare Write-Up

Sofia Ward / Bowie Chuang / Christina Pham / Carter Kulm

2025-03-11

Curated Sephora Skincare Generator 3000000

Content Based Filtering Recommendation System

Introduction

With the growing number of skincare products available, choosing the right one for an individual's specific needs can be overwhelming. To help users navigate this vast selection, we developed a content-based skincare recommendation system that suggests products based on their attributes and similarity to other items.

We wanted to build something that would save users money and time from trying product after product, since everyone's skin is different! Advertising and reviews are misleading since company goals don't align with the user's best interest: finding the right skincare! Our model not only selects products based on similarity, but also incorporates ingredient lists, skin types, and targeted skin concerns. This filtering gives users 10 individually catered products to their specific needs. For user accessibility we chose Sephora's Skincare selection.

Web-scraped Data Description

To build our skincare recommendation system, we collected product data from the Sephora website using web scraping techniques. After checking for permissions, we used the robots.txt file to acquire an html of product links, utilizing Selenium's Webdriver and Beautiful Soup. Filtering cosmetics and hair products left us with a categorized link data frame to begin web-scraping!

We encountered several obstacles but through trial and error extracted our information; pop-up ads, scroll-down functionality, custom user-agent strings to avoid bot detection. Scraping itself took copious amounts of time, and we found that smaller segments extracted less N/A's. Altogether we scraped **1,800 links** which filtered out to **500 observations and 12 columns.**

```
# Webscraper_skincare.Rmd for full code

filtered_sephora <- read_excel("data/filtered_sephora.xlsx")

dim(filtered_sephora)
```

```
## [1] 532  12
```

```
glimpse(filtered_sephora)
```

```
## Rows: 532
## Columns: 12
## $ `Brand Name`          <chr> "Dr. Barbara Sturm", "Herbivore", "Dr. Dennis Gr~
## $ `Product Name`        <chr> "Cleanser", "Aquarius BHA + Blue Tansy Clarifyin~
```

```
## $ `Product Category`    <chr> "Cleanser", "Cleanser", "Cleanser", "Cleanser", ~
## $ `Product Price`       <dbl> 108.0, 26.0, 39.0, 52.0, 18.0, 7.0, 40.0, 24.0, ~
## $ `Product Rating`      <dbl> 4.1, 4.6, 4.3, 4.4, 4.6, 4.3, 4.5, 4.3, 4.6, 4.6~
## $ `Product Size`        <chr> "5 oz/ 150 mL", "3.38 oz / 100 mL", "7.5 oz/ 225~
## $ `Product Reviews`     <dbl> 74, 269, 486, 58, 498, 448, 2948, 5380, 45, 498,~
## $ `Product Description` <chr> "A gentle foaming cleanser that removes makeup, ~
## $ `Product Ingredients` <chr> "-Mild Tensides: Provide thorough, yet gentle cl~
## $ `Skin Type`           <chr> "Normal, Dry, Combination, and Oily", "Normal, C~
## $ `Skin Concerns`       <chr> "Dryness, Fine Lines and Wrinkles, and Acne and ~
## $ URL                   <chr> "https://www.sephora.com/ca/en/product/dr-barbar~
```

The data set contains key attributes essential for content-based filtering, including:

- **Product Name**: The name of the skincare product.
- **Brand**: The company or brand that manufactures the product.
- **Category**: The type of product: exfoliates, cleansers, toners, serums, moisturizer, masks).
- **Price**: The cost of the product in USD.
- **Reviews**: The number of user reviews for the product.
- **Size**: The quantity of the product (e.g., 100ml, 1.7oz).
- **Ingredients**: A list of active and inactive ingredients.
- **Description**: A textual summary of the product, often provided by the brand or Sephora.
- **Skin Concern**: The specific skin issues the product addresses (e.g., acne, dryness, hyper-pigmentation).
- **Skin Type**: The recommended skin types for the product (e.g., oily, dry, combination).

Methodology

Data Pre-processing

After collecting the raw data, we performed several pre-processing steps:

1. **Cleaning the Data**: Removed duplicate entries and handled missing values through imputation or deletion.

```
# EDA_skincare.Rmd

Christina0_600 <- read_excel("~/Desktop/Skin_Care_Recommendation/data/final_scraped_1200/Christina0-600
fifi601_1200 <- read_excel("~/Desktop/Skin_Care_Recommendation/data/final_scraped_1200/fifi601_1200_.xl
scraped_sephora <- bind_rows(Christina0_600, fifi601_1200)

filter <- scraped_sephora  %>%
  filter(!(`Skin Type` =="N/A" |`Skin Concerns` == "N/A" | `Product Rating` == "N/A"))

sum(is.na(filter))
```

```
## [1] 0
```

2. **Feature Engineering**: Formatted price, rating, and size for consistency.

```
filter <- filter %>%
  mutate(across(`Product Price`, ~ as.numeric(gsub("\\$", "", .)))) %>%  # take $ out of price
  mutate(
    `Product Price` = as.numeric(`Product Price`),      # Ensure numeric conversions
    `Product Rating` = round(as.numeric(`Product Rating`), 1),
    `Product Reviews` = as.numeric(`Product Reviews`)) %>%
  mutate(`Product Size` = case_when(
```

```
    `Product Size` == "N/A" ~ "No Size",
    TRUE ~ `Product Size`      # change N/A to No size
  ) )
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `Product Reviews = as.numeric(`Product Reviews`)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
head(filter)
```

```
## # A tibble: 6 x 12
##   `Brand Name`              `Product Name`    `Product Category` `Product Price`
##   <chr>                     <chr>             <chr>                        <dbl>
## 1 Dr. Barbara Sturm         Cleanser          Cleanser                       108
## 2 Herbivore                 Aquarius BHA + B~ Cleanser                        26
## 3 Dr. Dennis Gross Skincare Alpha Beta® AHA/~ Cleanser                        39
## 4 Hourglass                 Equilibrium Reb~ Cleanser                        52
## 5 Benefit Cosmetics         Mini The POREfes~ Cleanser                        18
## 6 The INKEY List            Mini Fulvic Acid~ Cleanser                         7
## # i 8 more variables: `Product Rating` <dbl>, `Product Size` <chr>,
## #   `Product Reviews` <dbl>, `Product Description` <chr>,
## #   `Product Ingredients` <chr>, `Skin Type` <chr>, `Skin Concerns` <chr>,
## #   URL <chr>
```

Recommendation System: Content-Based Filtering

Our system uses **content-based filtering**, a recommendation approach that suggests products based on their attributes rather than user interactions. We opted for this approach to respect the privacy of Sephora's customers and for user accessibility.

What is Content-Based Filtering?

Content-based filtering analyzes the characteristics of items to recommend similar products. Instead of relying on user behavior (as in collaborative filtering), it compares the features of a given product to those of other products in the data set.

In our case, we represent each skincare product as a **feature vector**, incorporating product descriptions, ingredients, category, skin concerns, and other relevant attributes. The similarity between products is calculated using **cosine similarity**, which measures the angle between two vectors in a multi-dimensional space. A higher cosine similarity score indicates that two products are more alike.

Mathematically, cosine similarity between two product vectors **A** and **B** is given by:

$$S_C(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Where: - $A \cdot B$ is the dot product of the two vectors, - $\|A\|$ and $\|B\|$ are the magnitudes of the vectors.

Using this approach, when a user selects a product, the system finds other products with the highest cosine similarity, ensuring recommendations are tailored to the product's attributes!

Implementation

We implemented the recommendation system in **Python**, using `scikit-learn` for text vectorization and similarity computations. The key steps include:

1. **Text Vectorization**: We converted textual data (e.g., ingredients, descriptions) into vectors using **TF-IDF (Term Frequency-Inverse Document Frequency)**.

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
import openpyxl
import string

df = pd.read_excel("/Users/fifi/Desktop/Skin_Care_Recommendation/data/filtered_sephora.xlsx")
df
```

```
##                     Brand Name  ...                                                URL
## 0            Dr. Barbara Sturm  ...  https://www.sephora.com/ca/en/product/dr-barba...
## 1                     Herbivore  ...  https://www.sephora.com/ca/en/product/herbivor...
## 2    Dr. Dennis Gross Skincare  ...  https://www.sephora.com/ca/en/product/dr-denni...
## 3                     Hourglass  ...  https://www.sephora.com/ca/en/product/hourglas...
## 4            Benefit Cosmetics  ...  https://www.sephora.com/ca/en/product/benefit-...
## ..                         ...  ...                                                ...
## 527                     La Mer  ...  https://www.sephora.com/ca/en/product/la-mer-t...
## 528               The Ordinary  ...  https://www.sephora.com/ca/en/product/the-ordi...
## 529          Dr. Barbara Sturm  ...  https://www.sephora.com/ca/en/product/dr-barba...
## 530          iNNBEAUTY PROJECT  ...  https://www.sephora.com/ca/en/product/innbeaut...
## 531                      Murad  ...  https://www.sephora.com/ca/en/product/murad-mi...
##
## [532 rows x 12 columns]
```

```
df_recommend2 = df[["Brand Name", "Product Name", "Product Category", "Product Description",
                    "Product Ingredients", "Skin Type", "Skin Concerns"]]
df_recommend2_cols = ["Brand Name", "Product Name", "Product Category", "Product Description",
                    "Product Ingredients", "Skin Type", "Skin Concerns"]
df_recommend2.columns = df_recommend2_cols
# Select columns
# concatenate all the strings and then fit it into TfidfVectorizer

df_recommend2['Product Description'] = df_recommend2['Product Description'].fillna(value = "No Descripti
```

```
## <string>: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/indexi
```

```
df_recommend2
```

```
##                     Brand Name  ...                                      Skin Concerns
## 0            Dr. Barbara Sturm  ...  Dryness, Fine Lines and Wrinkles, and Acne and...
## 1                     Herbivore  ...            Pores, Acne and Blemishes, and Oiliness
## 2    Dr. Dennis Gross Skincare  ...                Pores, Dullness, and Uneven Texture
## 3                     Hourglass  ...                                Dryness and Dullness
## 4            Benefit Cosmetics  ...                                               Pores
## ..                         ...  ...                                                ...
## 527                     La Mer  ...                                     Pores, Oiliness
## 528               The Ordinary  ...      Pores, Uneven Texture, and Acne and Blemishes
## 529          Dr. Barbara Sturm  ...      Fine Lines and Wrinkles, Pores, and Blemishes
## 530          iNNBEAUTY PROJECT  ...   Fine Lines/Wrinkles, Dryness, and Uneven Texture
```

```
## 531                       Murad  ...           Dark Spots, Dullness, and Uneven Texture
##
## [532 rows x 7 columns]
```

```
df_recommend2[df_recommend2['Product Description'].isna()] # final N/A fix
```

```
## Empty DataFrame
## Columns: [Brand Name, Product Name, Product Category, Product Description, Product Ingredients, Skin
## Index: []
```

```
import string
from sklearn.metrics.pairwise import cosine_similarity

# function to remove punctuation from columns
def remove_punctuation(value):
    return value.translate(str.maketrans('', '', string.punctuation))

# df_recommend2 = df_recommend2.astype(str) # make each column a string
df_recommend2['string'] = df_recommend2['Brand Name'].map(remove_punctuation) + " " + \
df_recommend2['Product Name'].map(remove_punctuation) + " " + \
df_recommend2['Product Category'].map(remove_punctuation) + " " + \
df_recommend2['Product Description'].map(remove_punctuation) + " " + \
df_recommend2['Product Ingredients'].map(remove_punctuation) + " " + \
df_recommend2['Skin Type'].map(remove_punctuation) + " " + \
df_recommend2['Skin Concerns'].map(remove_punctuation)
```

```
## <string>: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/indexi
```

```
tfidf = TfidfVectorizer()
tfidf_matrix = tfidf.fit_transform(df_recommend2['string'])
```

1. **Similarity Calculation**: We computed pairwise cosine similarity scores between products.

```
cosine_similarity = cosine_similarity(tfidf_matrix)

similarity_df = pd.DataFrame(cosine_similarity, index = df_recommend2['Product Name'], columns = df_rec

similarity_df.head()
```

```
## Product Name                                        Cleanser  ...  Mini Rapid Dark Spot Correcting S
## Product Name                                                  ...
## Cleanser                                            1.000000  ...                              0.12
## Aquarius BHA + Blue Tansy Clarifying Cleanser       0.180169  ...                              0.06
## Alpha Beta® AHA/BHA Daily Cleansing Gel             0.191638  ...                              0.12
## Equilibrium Rebalancing Cream Cleanser             0.110345  ...                             0.079
## Mini The POREfessional Get Unblocked Makeup-Rem...  0.092545  ...                              0.07
##
## [5 rows x 532 columns]
```

1. **Recommendation Generation**: For a given product, the system returns the top **10** most similar products based on similarity scores.

```python
df_index = pd.Series(df_recommend2.index, index = df_recommend2['Product Name'])
def give_recommendation(product_name):
    product_index = similarity_df.index.get_loc(product_name)
    top_10 = similarity_df.iloc[product_index].sort_values(ascending=False)[1:11]

    print(f"Skin Care Recommendations for customers buying {product_name} :\n")
    print(top_10)

give_recommendation("Vinoclean Gentle Cleansing Almond Milk")
```

```
## Skin Care Recommendations for customers buying Vinoclean Gentle Cleansing Almond Milk :
##
## Product Name
## Vinoclean Makeup Removing Cleansing Oil                                0.451782
## Nourishing Whipped Almond Delicious Shower Body Cleanser               0.303975
## Nourishing Makeup Removing Oil Cleanser with Squalene and Vitamin E    0.303102
## Brighter Days Ahead Cleansing Trio: Cleansing Balm Starter Set         0.299228
## Mini Oat Cleansing Balm                                                0.298498
## Oat Makeup Removing Cleansing Balm                                     0.297734
## Midnight Ritual Retinol Renewal Serum                                  0.293591
## Oat Makeup Removing Cleansing Balm                                     0.289060
## Barrier+ Lipid-Boost Body Cream                                        0.283700
## Goat Milk Moisturizing Cream                                           0.281739
## Name: Vinoclean Gentle Cleansing Almond Milk, dtype: float64
```

EDA Visuals