Laurence Tran
Applied Data Analysis
May 20, 2023

Final Report: Predicting Beauty Product Popularity

Executive Summary

In today's world of e-commerce, predicting product popularity is essential for businesses to remain competitive and profitable. This is especially important in the beauty industry as it is driven by product trends and brand power. Understanding what makes a product popular would help companies and investors make more informed strategy, marketing and operational decisions.

In this project, I applied several machine learning algorithms to a Sephora dataset to build several predictive models that classify the popularity level of beauty products. Using a dataset of over 9,000 products, I focused on 3 key predictive features: brand, category and ingredient list to validate the two following hypotheses:

1. Using brand and category information, we can predict whether a product would be in the top 20% “loved” on Sephora
2. Using ingredient list information, we can predict whether a product would be in the top 20% “loved” on Sephora

After building several models including Logistic Regression, k-Nearest Neighbors, Decision Tree and Random Forest, I evaluated their performance using accuracy, precision, recall and area under ROC curve. With hypothesis 2, I also detailed a list of most important ingredient phrases in predicting product popularity.

Overall, Random Forest outperformed other models in both hypotheses 1 and 2 with the highest accuracy and precision metrics. Apart from their prediction results, the models’ ingredient phrase output would be of interest to beauty companies, as well as to investors looking to make informed decisions about the beauty industry.

Problem and Significance

Knowing which products are likely to be popular with customers allows companies to make informed decisions regarding production, inventory management, pricing, and marketing strategies. In the beauty industry, where new products are constantly being introduced and customers are always looking for the next "must-have" item, predicting product popularity is especially important. For example, if a product is predicted to be highly popular, a business may choose to increase production quantities to meet demand or invest more in marketing efforts to ensure the product reaches its full potential. Conversely, if a product is predicted to have lower popularity, a business may choose to produce fewer quantities or redirect marketing efforts towards more promising products.

However, there has been very limited publicly available research on how to determine beauty product popularity. As a result, my models aim to classify whether a product would be outperforming 80% of products on Sephora in terms of popularity.

Related Works

1. Machine Learning with Sephora Dataset:

In the project series "Machine Learning with Sephora Dataset"[1], Audrey Tang explored if number of loves on a Sephora product could be predicted using other features from the website. She scraped the Sephora website for a total of 2,836 product observations and 24 features per product, including brand name, product name, product type, price, multiple skin types, multiple skin concerns, and ratings.

As part of data preprocessing, she performed both exploratory data analysis and feature engineering. After handling duplication, incorrect values, missing values and incorrect data types, she performed the following feature engineering steps:

1. Categorical encoding using mean encoding
2. Outliers engineering using a capping technique with inter-quantile range
3. Variable transformation using Box-Cox transformation on price
4. Discretization to transform continuous variables into discrete variables
5. Feature scaling using standard scaler from scikit-learn

After feature engineering, the author performed feature selection by using multiple different techniques to select the right subset of features:

1. Basic Techniques: remove constant features and other features that are highly correlated to each other
2. Mean Squared Error: build a simple decision tree per feature to predict the target, make predictions using this decision tree, and rank the features according to the mean squared error, and select the highest ranked features
3. Lasso Regularization: introduce penalty term to regularize coefficient of variables such that if the coefficient is large enough, the cost function will diminish the coefficient to zero, effectively removing the feature from the model
4. Random Forest: create a random subset of features in each tree

After performing feature selection, the author put all 4 subsets plus the original dataset in a random forest model and still found the original dataset to give the highest R2 across both training and testing datasets.

Following feature selection, she evaluated 6 different models to predict the number of "loves": Linear Regression, Lasso Regression, Nearest Neighbors, Decision Tree, Random Forest, and Gradient Boosting. She used R^2 and MSE to evaluate each model and compare them against each other (Figure 1).

Model	**Train set**		**Test set**	
	R^2	MSE	R^2	MSE
Multiple Linear Regression	0.6683	332,550,167	0.6674	290,766,593
Lasso Regression	0.6682	332,550,395	0.6675	290,712,473
Nearest Neighbors	0.5675	433,587,329	0.5589	385,672,382

[1] Audrey T., "Machine Learning with Sephora Dataset Part 2— Exploratory Data Analysis". Medium (2020) https://medium.com/@audreyctang/machine-learning-w-sephora-dataset-part-2-exploratory-data-analysis-c6f3c53b40a4

Decision Tree	0.9999	10.487	0.3467	571,209,521
Random Forest	0.8124	188,099,603	0.7417	225,824,317
Gradient Boosting	0.9284	71,740,171	0.7817	190,895,245

Figure 1. Results from different models used to predict "loves" in "Machine Learning with Sephora Dataset"

Since Random Forest and Gradient Boosting have the highest R^2 and lowest MSE in the test set, she proceeded with tuning the hyperparameters in these two models.

- Random Forest best parameters: {'bootstrap': True, 'max_depth': 70, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 1400}
- Gradient Boosting best parameters: {'loss': 'lad', 'max_depth': 45, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 2000}

This project is important because similar to my project, it also uses some of the data scraped from the Sephora website. Since the author also attempted to predict the number of loves, I can use her results as the benchmark for my project. In addition, I also learned new ways to explore and visualize a dataset with many features so I can form a better initial hypothesis about how to best predict the number of loves for each product. In particular, I will look at the unique values for each categorical variable to understand the range of different values and the size of the feature space. For numerical variables, I will also plot graphs using seaborn and matplotlib similar to how the author explored her data.

2. Cosmetic Product Selection Using Machine Learning

In "Cosmetic Product Selection Using Machine Learning"[2], the authors created a cosmetic product recommendation system that suggests products based on ingredients suitable for each user attribute. To build the dataset, they scraped the Sephora website for 1,072 unique products with 12 features each. They then pre-processed the data by identifying and handling missing values, encoding data, splitting the data and performing feature scaling.

A corpus of ingredients was then developed to act like a dictionary, which was then used to create a cosmetic-ingredient matrix. The authors performed one-hot encoding on each item based on whether its ingredient list contains an ingredient from the dictionary (Figure 2).

[2] Rubasri S., Hemavathi S., K. Jayasakthi, Sangeerani Devi. A, K.Latha, N.Gopinath. "Cosmetic Product Selection Using Machine Learning". 2022 International Conference on Communication, Computing and Internet of Things (IC3IoT) (March 2022)

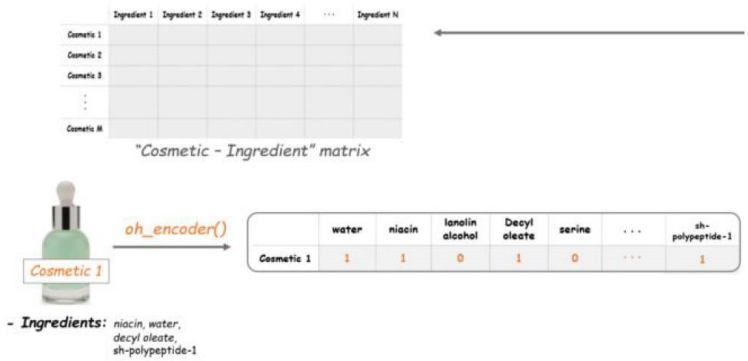

Figure 2. Cosmetic-ingredient matrix using one-hot encoding

Since the feature space grew very large due to the one-hot encoding, they then performed dimension reduction with t-SNE to compress the data for easier computations. Finally, they computed cosine similarity to see how a particular product compared to other products on the market, which served as the foundation for the content-based recommendation system. The system takes 2 inputs: the customer skin type and the product type, and produces a list of products that would be safe for them to use. Using accuracy as the main evaluation metric, the system has an overall accuracy of 93% within the dataset and 85% with external data.

This paper shows that the ingredient list can be a valuable source of information for prediction in beauty products and suggests a way to transform the text inside the ingredient list into categorical variables for Machine Learning purposes. Based on the findings in this paper, I will also make use of the ingredient list to predict product popularity. I will aim to produce a similar cosmetic-ingredient matrix for all products in my dataset.

Description of Solution

1. Dataset

The Sephora Product[3] dataset is from Kaggle and contains information on over 9,000 beauty products. There are 21 features including product details such as brand, category, name, size, rating, number of reviews, price, value price, and number of "loves". The dataset was built using web scraping methods like Selenium and Beautiful Soup.

2. Software

I used libraries such as Pandas, NumPy, Scikit-Learn, and Statsmodels to preprocess the data, build and train models, and make predictions. To visualize the data during preprocessing, I also used Seaborn and Matplotlib. Lastly, since I needed to vectorize the ingredient list to perform predictions based on ingredients, I used NLTK, an NLP library, to obtain a list of stop words to remove from the vectorization of my ingredient list.

[3] Raghad A., "Sephora Website". Kaggle (2020)
https://www.kaggle.com/datasets/raghadalharbi/all-products-available-on-sephora-website

3. Description and analysis of results

My objective was to predict the popularity of a product (number of "loves") based on other features. At first glance, it might make intuitive sense to use ratings or number of reviews to predict popularity. However, since my intention was for the model to predict the popularity of **new** products that haven't been introduced to the market, ratings or reviews wouldn't be available for such products. Therefore, I decided to focus on 3 key features: brand, category, and ingredients.

Since I only want to focus on prediction using 3 features, I will treat this problem as a classification problem because it would be difficult to predict the exact number of "love" based on just 3 variables. In addition, it would be more important for the users of my model (beauty companies, beauty retailers, investors) to know whether a product is likely to be very popular with high accuracy, rather than getting an exact number of "love" with potentially lower accuracy.

3.1 Data preprocessing and Feature Engineering

After performing Data Exploration and getting a sense of the size of the dataset, I proceeded to add a new feature to the dataset called "love_class", which classifies the number of loves for each product into 2 classes:

- High: these are products with number of loves in the top 20 percentile. I decided to pick top 20% instead of an even distribution (50%-50%) because more informed business decisions can be made for these top products in terms of shelf space, inventory, and marketing dollar investments, as compared to a simple popular/unpopular classification.
- Low: these are products with number of loves in the bottom 80 percentile.

In addition, since "brand" is a categorical variable, I needed to encode it before feeding it into a model. As it's a nominal variable, assigning integers to different brands would result in little predictive power because the brands are not related to each other. Performing one-hot encoding on "brands" would enlarge the dataset too much since there are 327 brands. Therefore, I decided to rank the brands by the number of loves they have received from existing products in the training database. Using this ranking, I added a new feature "brand_rank" to the dataset.

Lastly, since "category" is also a categorical variable, I decided to perform one-hot encoding on "category" for each of the models by creating dummy variables and adding them to the list of features. There are 146 brands, which enlarged the feature set but to a manageable extent as compared to performing one-hot encoding on the 327 brands.

3.2 Hypothesis 1: "Love_class" can be predicted with "brand_rank" and "category"

I set out to test my first hypothesis that we can predict whether a product will be in the top 20% popularity based on its category and how its brand ranked in the past. I used 3 types of classification models to test this hypothesis: Logistic Regression, k-NN, and Random Forest. I then evaluated their performance on both the training dataset and testing dataset using 4 metrics: accuracy, precision, recall, and area under ROC curve (ROC AUC).

Metric	**Logistic Regression**		**k-NN (k = 5)**		**Random Forest**	
	Train	Test	Train	Test	Train	Test
Accuracy	0.820081	0.807203	0.804958	0.805748	0.826162	0.817752

Precision	0.607321	0.546875	0.525397	0.534722	0.640468	0.597786
Recall	0.284489	0.252708	0.257989	0.277978	0.298519	0.292419
ROC AUC	0.825871	0.799641	0.738529	0.715965	0.828448	0.793549

Figure 2. Results of 3 different models classifying "love_class" based on "brand_rank" and "category"

Random Forest gave me the highest accuracy, precision and recall out of the three models. At 81.8% accuracy on the test dataset, the model does a good job of predicting whether a product will be in the top 20% popularity by number of loves. Looking at the precision-recall tradeoff across the board, my models generally have higher precision than recall, indicating that they are better at minimizing false positives than minimizing false negatives.

One disadvantage of this model is it will not be able to predict "love_class" for new products coming from new brands because they would not have a brand_rank available. Therefore, it should only be used on new products launched by one of the existing 324 brands in the dataset.

3.3 Hypothesis 2: "Love_class" can be predicted with ingredient list

To predict "love_class" based on the ingredient list for each product, I first extracted the individual ingredient phrases in each list using regex. This built a "vocabulary" of valid ingredients that I could use to vectorize any ingredient list on. I also used the library NLTK to get a list of stop words that my vectorizer should ignore when evaluating a list of ingredients. I then used CountVectorizer to create a matrix of ingredient phrase counts in the ingredient list, which served as my main features for prediction.

Using the vectorized ingredient lists, I developed 2 models for prediction: Decision Tree and Random Forest. These two models were then compared to each other based on accuracy, precision, recall and ROC AUC (Figure 3).

Metric	**Decision Tree**		**Random Forest**	
	Train	Test	Train	Test
Accuracy	0.959015	0.743003	0.958859	0.781897
Precision	0.921601	0.357285	0.973734	0.434307
Recall	0.867347	0.317376	0.814757	0.210993
ROC AUC	0.983369	0.563724	0.969469	0.690802

Figure 3. Results of 2 different models classifying "love_class" based on "ingredient"

Random Forest gave me the higher accuracy, precision and ROC AUC, while Decision Tree gave me slightly higher recall. Random Forest appears to be the better model with 78% accuracy on testing dataset. In addition, precision and recall both drop significantly from training to testing dataset, so there are signs of overfitting in both of my models.

However, an advantage of evaluating ingredient list is I can extract the most important phrases that my models used to classify whether a product would be in top 20% popularity. There are common phrases across both models such as "iron oxides", "titanium dioxide", "seed oil", and "synthetic fluorphlogopite" (Figure 4). This could serve as useful information for cosmetic companies as they create new formulations for products that hopefully would become popular.

Top 20 most important ingredient phrases	
Decision Tree	**Random Forest**
iron oxides	iron oxides

citric acid	titanium dioxide
titanium dioxide	caprylyl glycol
sodium hyaluronate crosspolymer	seed oil
sodium hydroxide	tin oxide
fruit extract	butylene glycol
seed oil	synthetic fluorphlogopite
butylene glycol	bismuth oxychloride
zinc stearate	zinc stearate
calcium sodium	disteardimonium hectorite
hexylene glycol	sodium dehydroacetate
synthetic wax	propylene carbonate
echinacea purpurea root extract	tocopheryl acetate
disteardimonium hectorite	alcohol denat
synthetic fluorphlogopite	citric acid
bismuth oxychloride	manganese violet
acacia senegal	potassium sorbate
castor oil	leaf extract
alcohol denat	ethylhexyl palmitate
isomethyl ionone	benzyl alcohol

Figure 4. Top 20 most important ingredient phrases for 2 models used to predict "love_class"

Takeaways

1. What did not work, trade-offs, and decision on final solution

There were many trial-and-errors in this project. I initially tried to treat it as a regression problem and performed both linear and polynomial regression to predict the number of "loves" based on "brand_rank", "rating", "number of reviews" and "category". Even after trying multiple combinations of features and polynomial degrees, all my regression models performed poorly with R^2 of less than 0.1 and very high MSE. I also realized later in the process that if I were to use this model to predict popularity of a new product, "rating" and "number of reviews" would not be available for a product that has not launched. Therefore, I pivoted and treated it as a classification problem instead, which would be more informative to my target model users (business owners, retailers, investors). This is the trade-off I made to align with the business case and increase model accuracy.

I initially also tried to perform one-hot encoding on both "brands" and "category". With 328 brands and 146 brands, the one-hot encoding enlarged the feature space significantly and resulted in long runtime or crashes. I then had to re-evaluate how to best encode these categorical variables, and ultimately decided to create "brand_rank" while keeping one-hot encoding for "category". This is the trade-off I had to make between the size of the feature space and runtime.

In addition, the ingredient list was not clean. While I tried my best to extract the ingredients using different regex expressions, even the best results contained non-ingredient phrases that were added to explain the functionality of the ingredients, such as "help protect against free radical damage while moisturizing" and "instantly brighten the look of dull skin". To completely clean these phrases, I would have to spend much more time on data cleaning. Therefore, I used CountVectorize instead of other techniques so that the feature matrix would rely on the frequency of phrase appearance and these specific phrases would naturally have lower frequency.

After this hurdle, I continued to have issues as my vectorizer would try to parse phrases into words and perform predictions based on single words, which was not as informative as ingredient phrases. For example, predicting based on the word “oxide” would be less helpful than “titanium oxide”, or predicting based on the world “oil” is not as helpful as “grapeseed oil”. Therefore, I had to spend many hours reading documentations to understand how to vectorize based on ingredient phrases.

Ultimately, the decision on which model to use depends on the objective of the user. To maximize accuracy and precision (minimize false positives), the Random Forest model with “brand_rank” and “category” should be used. However, to understand the most important ingredients in predicting popular products, the Random Forest model with ingredient list should be used.

2. Effort and data I couldn’t obtain

I spent approximately 10 hours on literature reviews, reading through the 6-part series “Machine Learning with Sephora Dataset” to extract the most important learnings for my project. Since the dataset I obtained from Kaggle was of high quality, I didn’t have to spend too much time on preprocessing, feature engineering and visualization (~5 hours). However, I did waste a lot of time in the beginning (~8 hours) as I tried to treat the problem as a regression problem and wasn’t able to get a satisfactory model with low enough R^2 values. In later stages of the project, majority of my effort was spent on finding the right approach to encode categorical variables without drastically increasing the feature space (~10 hours), as well as learning to vectorize unstructured data like ingredient lists for prediction purposes. This was my first time handling texts, so I had to spend ~20 hours researching how to extract a valid vocabulary, as well as how to vectorize based on phrases instead of individual words. Towards the end of the project, drafting the final report and presentation took ~7 hours.

In an ideal world, I would love to have a clean ingredient list to work with. Due to time constraints, I couldn’t clean the ingredient list too much and had to settle for a “good enough” result from parsing with regex. Other features that would have been very informative are sources of product browsing traffic (google search, recommendation system, ads, or influencer links) and sales volume. Sources of browsing traffic would be important in predicting products that went viral online, while sales volume would measure actual transaction volume from each product. This information exists but is not publicly available.

3. Project continuation and what I find interesting/challenging about the project

If I had more time, I would start with decoupling ingredient list and product category. It could be possible that certain product categories receive more loves than others and products in those categories could have similar ingredients (e.g., titanium dioxide is common in sunscreens). As a result, the ingredients could be highly correlated with category and I would want to separate them, potentially by exploring the most important ingredient phrases within each category.

Given more time, I would like to also incorporate product reviews into this project. Understanding the sentiments associated with highly popular products could be important to marketing departments as they try to formulate the right image and marketing message for their new products. In addition, it would be interesting to see if certain products became popular because they were considered “dupes” (duplication) of another viral product. This information is usually found in the comment section for each product.

Overall, this project helped me learn a lot about classification models. The most interesting finding I had was uncovering the list of important ingredients that my models used to predict popularity. Since dealing with unstructured data was my biggest learning curve on this project, it was satisfying to see my efforts resulting in information that could be useful to many stakeholders. While structured data is definitely easier to handle, this project showed me that analyzing unstructured data can result in surprising new findings that I previously did not consider.