Chuwei Lyu, Xinyi Shu, Shuran Wang, Yijing Zhou
Data Analytics
2019/12/15

Secret of Sephora:
A Study of Finding Potential Award Winning Products

PART I. INTRODUCTION

Sephora is the largest cosmetics retailer in the U.S., with more than 6,000 products on sale. As the holiday season approaching, we are interested in finding out ideal products among the enormous amounts of total products. We referred to the Allure Best of Beauty Awards, which nominates the most amazing products in each category(e.g. Makeup, Skincare, Bath & Body) annually by the editors of Allure magazine. We wanted to examine the differences between allure and non-allure products' attributes and use the dataset of both allure and non-allure products to find out potential award-winning products. In particular, we used the model we built to predict men's potential allure products, as Allure doesn't provide Man's Beauty list currently. We believe that our results will be helpful for those who need advice on selecting the right men's product.

PART II. WEB SCRAPING

We obtained all the data we needed from Sephora.com by web scraping. Our target features include: product name, product brand name, price, number of likes, number of reviews, average ratings (5-star based), number of 1~5 star ratings, and reviews. Each review has: individual rating, review title, review content, recommend/ not recommend, and number of helpful/ unhelpful votes for this particular review.

One particular challenge we faced was lazy load. The website sephora.com only loads 6 reviews for each product at a time. Instead of using the Selenium package, we came up with an alternative. It turned out that for every product, all the reviews were saved in one place linked at "api.bazaarvoice.com/data/reviews.json", and we reached that by going to 'inspect'--'network'--'reviews.json'. From there we scraped all the reviews we needed for further sentiment analysis and machine learning models.

PART III. DATA CLEANING AND VISUALIZATION

With all the data on hand, we continued to make some visualizations about the dataset's general statistics. Before that, we did some basic data cleaning. To start with, we dropped all NaNs, corrected the price column and created percentage of all star rating columns. Then, we removed the products with less than 100 reviews to reduce the impact of extreme cases (e.g. some products have only 5 reviews and rate 5 stars). In that case, we assume that 100 is a fair enough number of unbiased feedback. In addition, although there may be fake reviews, we assume that tens of these reviews won't have much influence on

the results of hundreds of thousands of reviews in total. Therefore, removing the products with less than 100 reviews also eliminate the impact of fake reviews.

After data preprocessing, we first plotted the top 15 brands with the highest average ratings and their corresponding prices (See Appendix 1). The brand's average prices ranged from 10 dollars to 100 dollars. Therefore, it seemed that a brand's price had no significant impact on their average ratings. We then plotted the price distributions of all the products (See Appendix 2). The mean price was around 35 dollars, and the sample had lots of outliers. Therefore we believed that it was necessary to remove those outliers. What's more, we explored the different memberships' contributions. Sephora has three levels of membership, from the lowest BI to the highest level rouge. People who don't have a membership are guests. From the chart, we found that the average helpfulness of rouge members was the highest(See Appendix 3). We believed that it was because rouge members spend most at Sephora, which meant that they have a better understanding of products and write unbiased reviews. Besides, they also have a lower possibility of cheating on their reviews. Finally, we generated a word cloud that depict keywords on the website (See Appendix 4).

PART IV. TEXT MINING

From web scraping, we obtained all the reviews for every product. To make sense of these texts, we chose to use sentiment analysis techniques. The goal here is to determine whether or not the reviewer likes the product, i.e. whether they hold a positive or negative point of view for this product. The best one that serves our need here is the Vader score for every product.

We took the Vader compound score, which ranges from -1 to +1, with -1 being completely negative and +1 being completely positive. The score directly reflects the average sentiment contained in all the reviews for a single product. We consider this score an important feature in our machine learning model, because it shows how much users love this product, and thus the potential to win an award.

A fun fact we found is that you do not need to read through all the reviews to get a good enough idea about the product. We computed the Vader score for 10%, 20%, 30% … to 100% of the reviews for every product. A correlation analysis shows that at 30% level, the correlation between the Vader score for total reviews reached 0.72.

We also attempted to use these reviews text for model validation by building an LDA model with 2 topics, as advised. The idea is to separate the whole product set into 2 groups: the good ones and moderate ones, and validate our model prediction. However, this plan did not go through because topic separation proves to be infeasible in our case. Topic 0 and topic 1 remains to look fairly similar, despite the effort of changing passes, adjusting stopwords and special words, etc. We consider this is a reasonable outcome, since the whole text is basically talking about the same thing: their view on beauty products.

PART V. MACHINE LEARNING

We then used machine learning to make predictions. Since there were only 240 allure products on sale on the Sephora website, to balance the data, we randomly picked 240 non-allure products with equal weights for each subcategory. Then, we processed the data by dropping attributes including the brand name, product name, category, and subcategory. After shuffling the remaining data, we checked the features' correlations (see Appendix. 5). Note that most numbers were relatively low. The exceptions lied in total review vs. loves and avg_rating vs. fivestar_pct/onestar_pct. However, we still kept them due to limited numbers of features. Since the range of features differed a lot, we then used MinMaxScaler to normalize all features to increase accuracy. Finally, we used train_test_split to divide the training and testing data by divide 70% of data as training data and 30% as testing data.

At the first stage, we had 240 allure products(assigned value as 1) and 240 non-allure products(assigned value as 0). Setting GridSearchCV to be 5, we tried five different models, including Logistic Regression, Decision Tree, Random Forest, Bagging and Neural Network to fit the data and tune hyperparameters. Then we used 2 evaluation metrics, including Accuracy and AUC to compare all the models. But the initial result we got was not satisfying (see Appendix6).

We believed the poor result was due to the fact that our dataset was too small. Therefore, we enlarged our dataset to 1200 non-allures (decided based on the running time limitation) and applied Synthetic Minority Over-Sampling Technique(SMOTE) on our training data to deal with the imbalanced dataset. The method created synthetic (not duplicate) samples of the minority class. Hence making the minority class (1s) equal to the majority class (0s). Before applying SMOTE, we had 852 non-allure products and 156 allure products. After applying SMOTE, we had 852 non-allure products and 852 allure products.

With the balanced dataset, on average, all models performed better (see Appendix 7). Using the Random Forest method, we also derived feature importance. The top 5 important features are the numbers of recommendations, followed by the number of total reviews, the numbers of loves, our sentiment analysis (Vader compound score) on all reviews and prices.

Based on all the results, we chose Random Forest to be the final best model because of its highest value of Accuracy and AUC. However, we noticed that there were not much difference between each model, and the hyperparameters would change each time we ran the gridsearch. We thought it was due to limitation of our data and features.

Finally, we used our model to predict a product list solely for man. Since the attributes in our models are quite general, rather than specific to women products, we assumed the features have the same weight for men's products. Considering the fact that the number of man-exclusive products is too small, and most skincare and makeup products can be used by both male and female, we decided to predict men-exclusive fragrances. Our final results included 20 different men's fragrances that we believe have the potential to be awarded.

We can also get a specific number of recommended products by changing the threshold. For example, in our code file, we set the threshold to get 10 products.

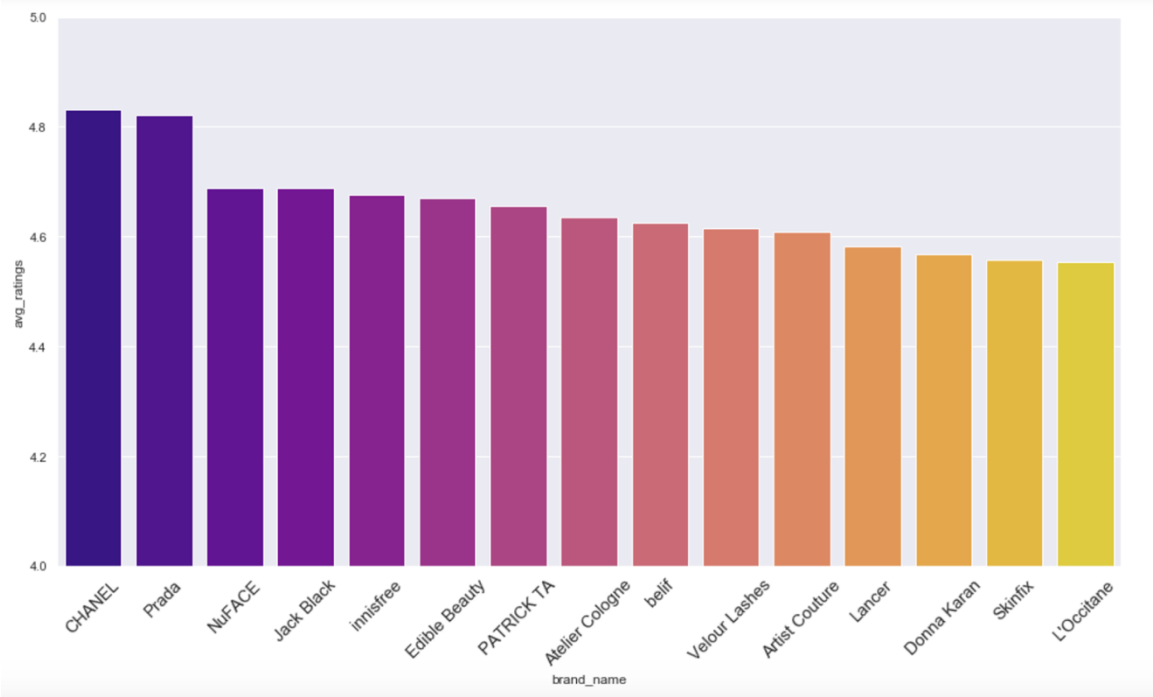
One thing to mention is that every time we reset the model, the parameters of the models will be slightly different. There are two reasons: 1. the property of functions like SMOTE, Shuffle, GridsearchCV etc; 2. the dataset itself is very sensitive to different hyperparameters because of the limited size.

PART VI. CONCLUSION AND LIMITATIONS

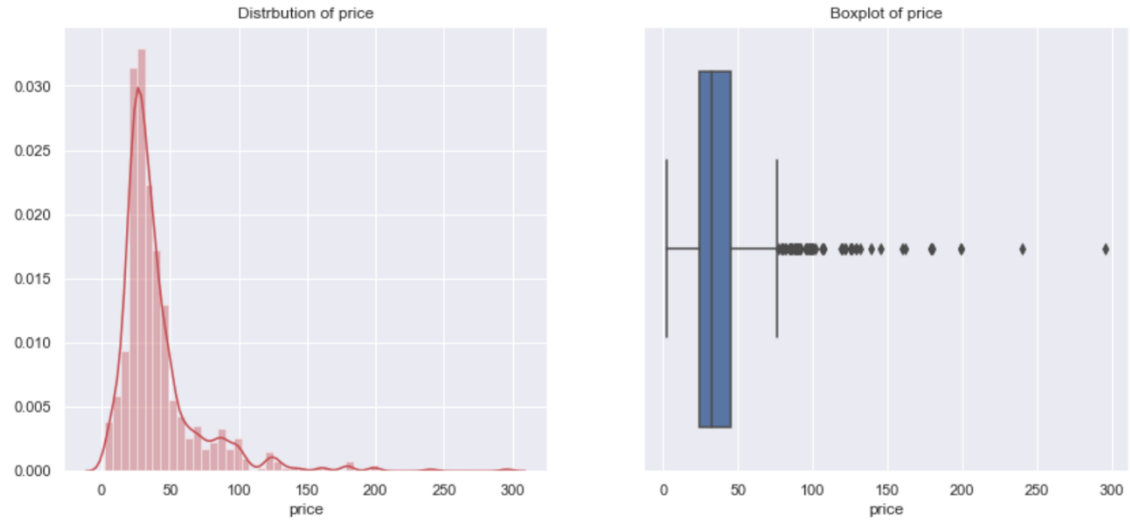
To validate our modelling, we first tried the LDA model. As the results were not satisfying, we decided to compare our results with four major men's fragrance awards: *Trenpotter*, *Gearhungry, GQ* and *Bazzar*. For example, *Trenspotter* has a total of thirty-five products on its award lists, while Sephora has 21 of them on sales. Our list includes 9 of them. Besides, One of the products in our results is awarded in all of the four awards. Therefore, we believe our final result is satisfying.

There're some future works that can be conducted. For example, due to time limit, we only included 1200 products. But we can add much more products and features to make a more precise prediction. What's more, we could try other product categories besides fragrance of men, and validate our data by using other websites.

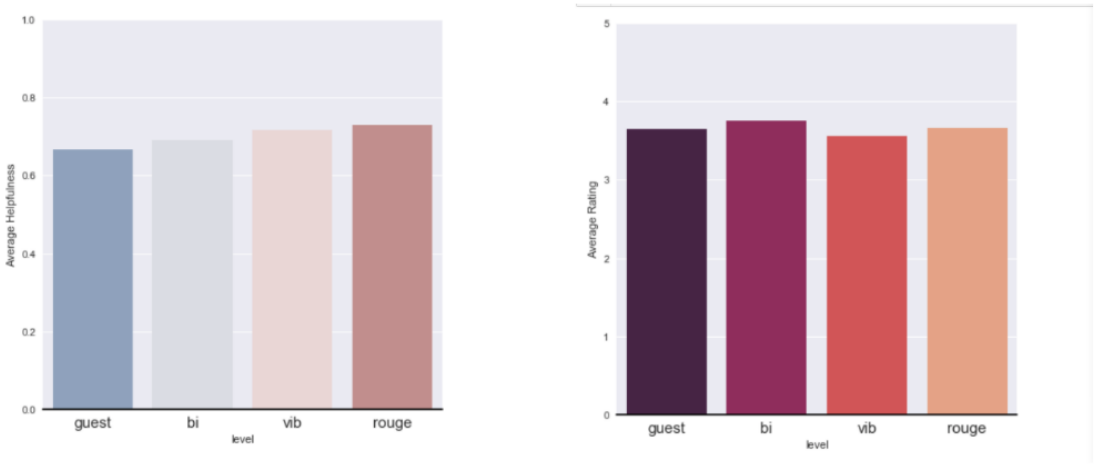
Appendix



Appendix 1

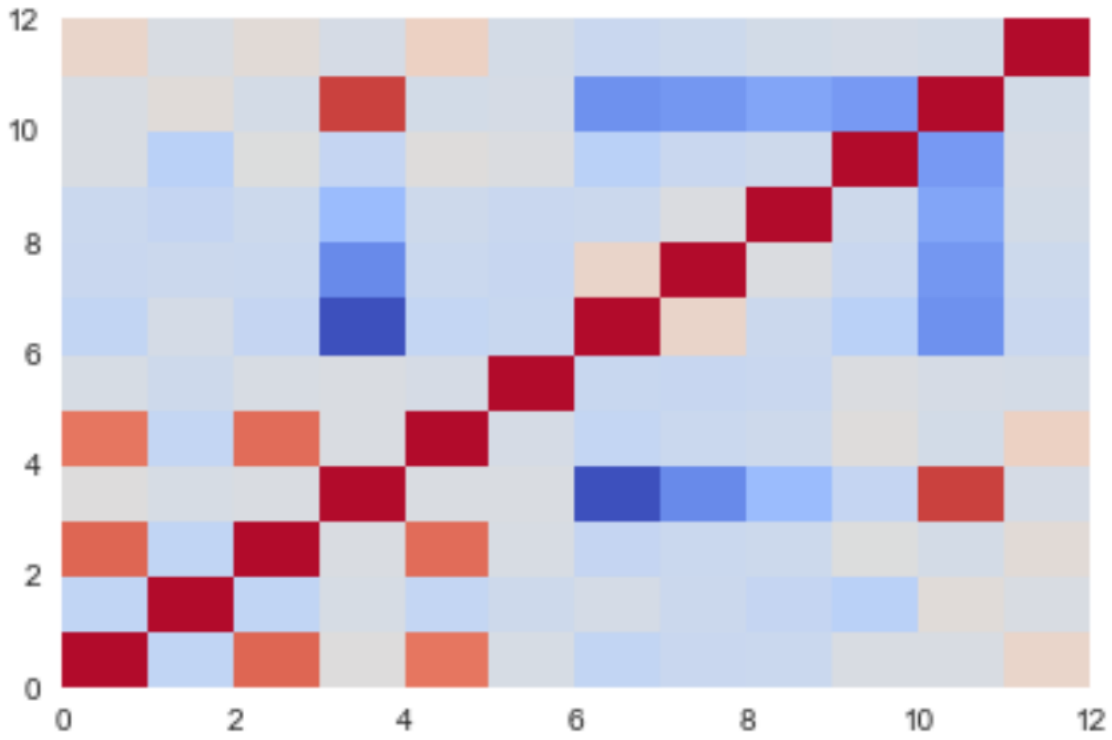


Appendix 2



Appendix 3

Appendix 4



Appendix 5

Model	Accuracy	AUC
Logistic Regression	0.58	0.58
Decision Tree	0.60	0.61
Random Forest	0.65	0.65
Bagging	0.62	0.62
Neural Network	0.61	0.61

Appendix 6

Model	Accuracy	AUC
Logistic Regression	0.71	0.62
Decision Tree	0.70	0.64
Random Forest	0.78	0.66
Bagging	0.74	0.63
Neural Network	0.74	0.65

Appendix 7