Data1030 Final Project

Predicting the Ratings of Sephora Products

Siyu Shen

Brown University

Github Link: https://github.com/ssy00603/DATA-1030-project

1 Introduction

At Sephora, the beauty community always shared their online shopping experience. This reviewing system benefit a lot for the potential customers of Sephora. Rating is one of the most important decision tools for those customers. That's why it is important to find out what kind of product will have a good rating which is over 4.0. Therefore, in Sephora website data file, I use 'rating' as target variable. Since rating is range from 0.0 to 5.0, I have a regression problem.

There are 9168 data points and 12 features in 'sephora_website_data.csv' data. To be more specific, 'love' is the number of people loving the product. 'MarketingFlags' is a boolean feature which determines if the product from the website has limited edition, sells exclusive or only online.

This data set has been used by Raghad Alharbi's project[1] in Data Science Immersive Course. The project was about using web scraping methods to collect more than 1,000 useful records from any website of their choice. She analyzed the distributions of features. Also, she finds correlations between each columns.

2 Exploratory Data Analysis

The very first step I took is to visualize features and explore the relationship of different features.

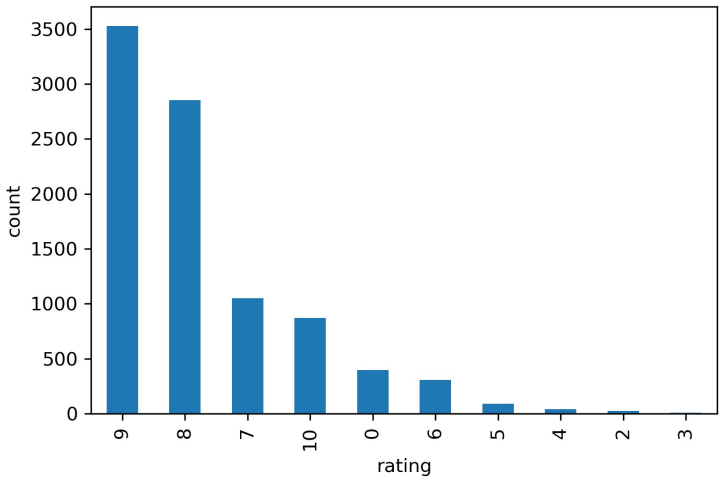

Figure 1: Bar plot of rating

Figure 1 is right skewed and most rating is range from 4.0 to 4.5. Also, seldom customers give a low rate which is range from 0.0 to 2.5.

Also, I want to examine the relationship between different categories of products and their prices. Figure 2 shows that Face Serum and Perfume have higher prices which are higher than or equal to 50 dollars. Face Wash & Cleansers, Hair Styling Products & Roller ball and Travel size have comparable lower prices which are less than 50 dollars.

Furthermore, I want to see that if a products is MarketingFlags will affect the numbers of love. According to Figure 3, the product without MarketingFlags has a smoother distribution which indicates that products with MarketingFlags tend to have less love. However, the number of 'love' have similar right skewed distributions for both.

Then I take a look at the relationship between numbers of love customers give for a product and the number of actual reviews. Figure 4 shows that less 'love' tend to have less reviews since the data points are concentrated in the bottom left corner. There seems to have a weak positive relationship between 'love' and number of reviews.

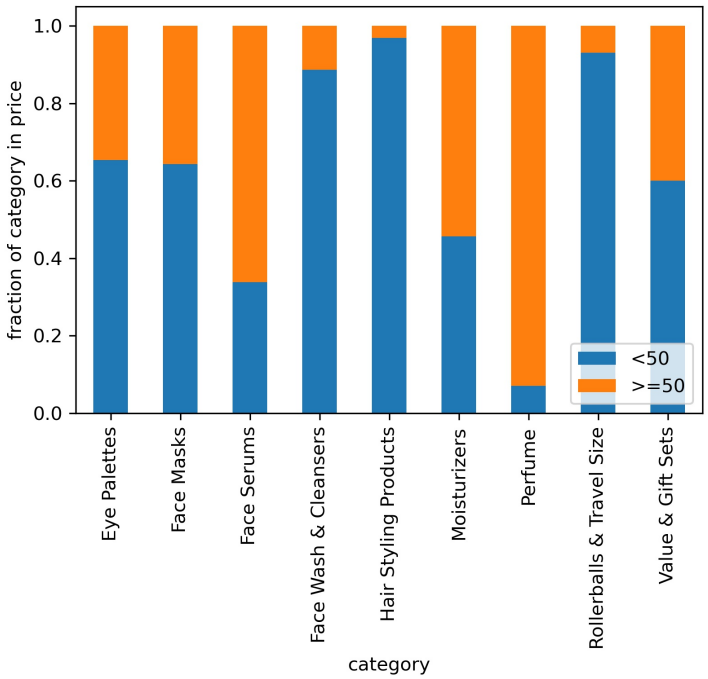

Figure 2: Stacked bar plot of category and fraction in price

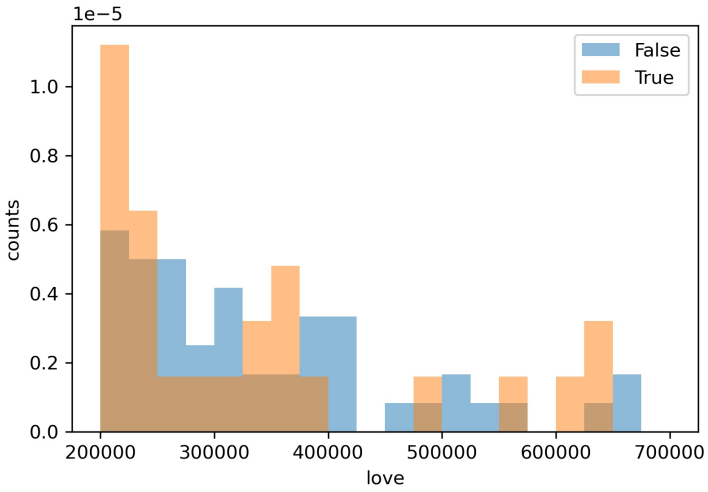

Figure 3: Category-specific histograms of love and MarketingFlags

3 Methods

3.1 Data Splitting

The data set is not IID because some data points may be generated by the same customer with a single user id number. So the data has a group structure. Since 52% of 'size' data has missing value in the original data set, and they are missing not at random. For example, for the gift sets, there is always no size information because it include different products. Therefore, I decided to drop the 'size' features.

First, I split the the data set into other and test with ratio 8:2 using group shuffle split. Then applied Group k-fold where k = 5 for splitting data because we want to random splitting the train and validation set for the data has group structure based on customer ID. Now I have 64% of the data to train models, 16% of

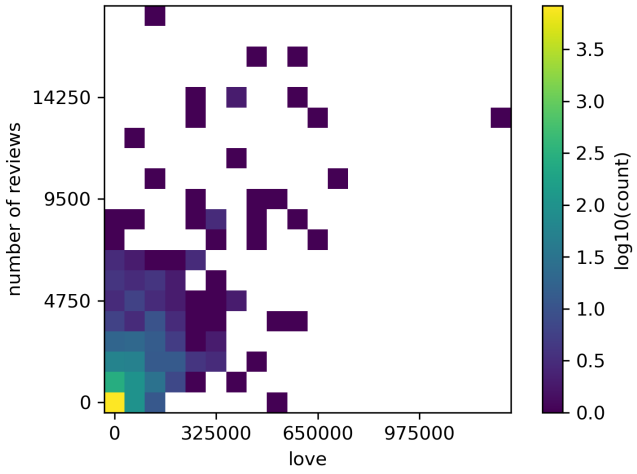

Figure 4: heatmap of love and number of reviews

the data to fit the model onto the validation set, and 20% of the data in the test set.

3.2 Data Preprocessing

There are 10 features we are going to use in the prepossessed data. I used 'rating' as the target variable, and applied One-Hot encoder to unordered categorical data like 'MarketingFlags', 'category', 'exclusive', 'limited_edition', 'limited_time_offer' and 'online_only'. Then standardized all other continuous features like 'love', 'number_of_reviews' and 'price' using StandardScaler encoder. Now I have 153 features and 9168 rows.

3.3 Machine Learning Pipeline

I trained 5 different supervised machine learning models like Lasso, ridge, elastic net, k-nearest neighbor and random forest algorithms. Also, to evaluate the performances of these models, I choose mean squared error(MSE) as my evaluation metric because I am working with a regression problem.

By using the cross-validated grid search, a exhaustive search over specified parameter values for an estimator, I tuned parameters of different models with different values. For least absolute shrinkage and selection operator (Lasso), I decided to tuned alpha, the penalty parameter which controls the strength of the penalty term. I tuned alpha with values 1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3. Finally, I conclude that alpha = 0.001 will have the smallest cross-validated MSE = 0.914 and test MSE = 1.067 which indicate it is the best parameter.

For ridge regression, I decided to tuned regularization strength alpha. I tuned alpha with values 1e-3,1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3. Finally, I conclude that alpha = 10.0 will have the smallest cross-validated MSE = 0.911 and test MSE = 1.066 which indicate it is the best parameter.

For elastic net regularized regression, the linear combinations of the L1 and L2 penalties of the lasso and ridge methods, I tuned alpha and l1_ratio. When l1_ratio = 0, the penalty function reduces to the L1 (ridge) term and when l1_ratio = 1 we get the L2 (lasso) term. Also, 'alpha' is used to determine how much weight is given to each of the L1 and L2 penalties. I tuned alpha with values 1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3 and L1 ratio with evenly spaced over [0, 1] to 11 numbers. Finally, I conclude that alpha = 0.001, l1_ratio = 0.3 will have the smallest cross-validated MSE = 0.910 and test MSE = 1.066 which indicate they are the best parameters.

For k-nearest neighbor regressor, I decided to tune number of neighbors(n_neighbors) and weight function(weights). I tuned n_neighbors with values 1, 10, 30, 100 and weights with 'uniform' or 'distance', where 'uniform' means all points in each neighborhood are weighted equally and 'distance' means weight points by the inverse of their distance. Finally, I conclude that n_neighbors = 100 and weights = 'distance' will have the smallest cross-validated MSE = 0.843 and test MSE = 0.921 which indicate they are the best parameters.

Finally, I tried random forest, and I decided to tune the maximum depth of the tree(max_depth) and the portion of numbers of features(max_features). When the depth of trees increases, the model will be

more complex. Also, when using all features would be computationally heavy. Therefore, max_depth and max_features should be controlled to avoid overfitting. I tuned max_depth with evenly spaced over [10, 210] to 11 numbers and max_features with evenly spaced over [0.5, 1] to 11 numbers. Finally, I conclude that max_depth = 30 and max_features = 0.9 will have the smallest cross-validated MSE = 0.264 and test MSE = 0.301 which indicate they are the best parameters and the model is not overfitting.

Model	CV MSE	Test MSE
Lasso	0.914	1.067
Ridge	0.911	1.066
Elastic Net	0.910	1.066
K-Nearest Neighbor	0.843	0.921
Random Forest	0.264	0.301

Table 1: Error rates for different models

3.4 Uncertainties

However, there are uncertainties due to random splitting and non-deterministic random forest methods. Because of bootstrapping, which is resampling the same dataset to create many samples, random forest methods will not guarantee reproducible results. Therefore, even for the same input data, it can exhibit different behaviors on different runs. I tried different random states which will give us a different train seed to measure the uncertainties. We can see that max_depth changed from 30 to 10 and max_features changed from 0.9 to 0.6 for the best model in this case. Moreover, random forest may be overfitting especially when the data set is noisy. Therefore, we need to make sure the model is not overfitting by adjusting parameters. However, during the manually adjusting process, it is likely to miss the best model. We could try more parameters to see if our model improved without overfitting problem.

4 Results

To find the best fit model for predicting ratings, different machine learning models are built and predictions are made based on each model. We tried Lasso, ridge, elastic net, k-nearest neighbor, and random forest models, and conclude that the Random Forest model with max_depth = 30 and max_features = 0.9 is the best model with the lowest error rate. To see how much improvement our model made, I compared it with the baseline model.

The baseline R^2 is 0, and the R^2 for the random forest model is 0.732. The random forest model has a larger proportion of the variance in the target variable 'rating' that is predictable from the other features. Since our model is not overfitting, we can use R^2 to evaluate it. Since R^2 is larger than 0.7, I have a good model mathematically. Besides, the baseline MSE is 1.121, and the MSE for the best random forest model is 0.301. The baseline RMSE is 1.004, and the RMSE for the best random forest model is 0.549. By using the random forest model, the standard deviation of the unexplained variance is smaller. There are 0.453 standard deviations above the baseline RMSE of the random forest model RMSE is. Although the actual difference between the prediction and the true values decreases, the random forest model seems not that accurate. According the the RMSE, the difference between the actual rating and the predicted rating is 0.549 on average. We could see what else we could do to further improve the random forest model in 'Outlook' section.

Method	R^2	MSE	RMSE
Baseline	0.0	1.121	1.004
Random Forest	0.732	0.301	0.549

Table 2: Predicted scores of different models

I calculated the impurity-based feature importance of random forests and listed the 10 most important features in Figure 5, which are the numbers of reviews, love, and price, etc. An increase in the number of reviews, prices, and loves will increase the customer's rating given for a product. Also, category_High Tech Tools, category_Lip Treatments, and category_Hair Removal, which have values closed to 0, are the 3 least

important features in this case. One interesting observation is that nearly all features that are comparable unimportant are from category feature. This may due to the category contain more than 140 different types of products and people rarely use a certain type of product. The numbers of reviews and loves that have high feature importance make sense because customers always share their experience of a product through these ways besides ratings. If a customer provides a high rate after they use the products, they are more likely to click 'love' for this product.

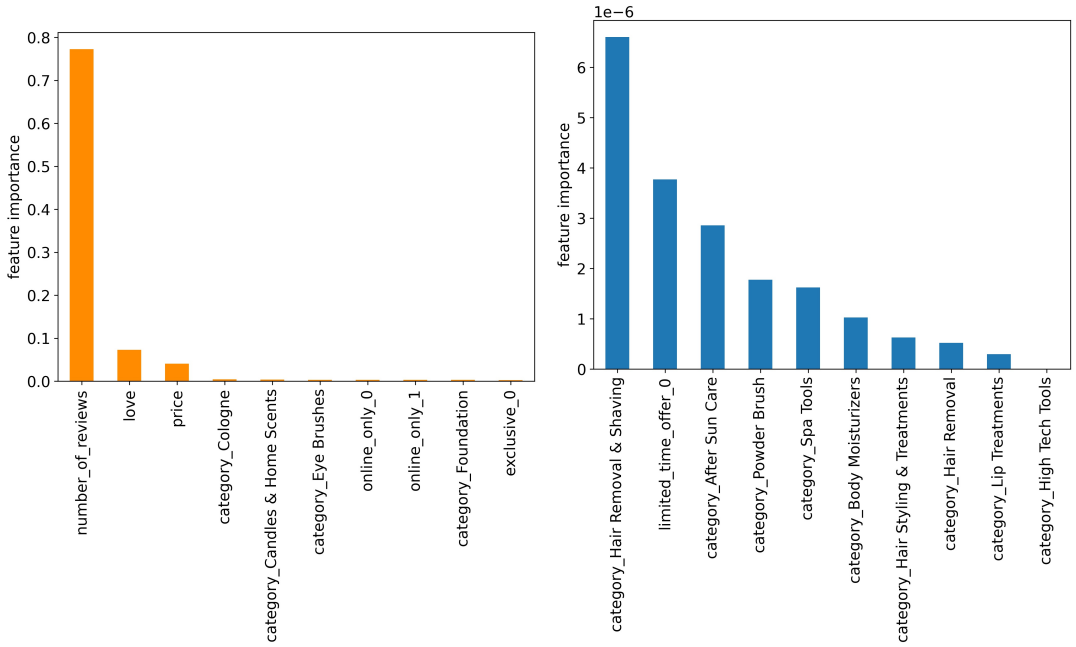

Figure 5: Global feature importance values

5 Outlook

We cannot guarantee our model is perfect. Random forest is less interpretable than a single decision tree. Therefore, I could improve this by constructing and visualizing trees as a sequence of decisions. Another technique I could use is XGBoost which combines the advantages of both random forest and gradient boosting. XGBoost will strengthen the model with weak predictions and make it better[2]. In the future, one way to improve model performance is to gather more data points. For example, if it is possible to obtain the time data customers purchased, we can predict ratings based on past values. Besides, we could collect more numeric features like the costs, sales and profits of the beauty products which will help us to make more accurate predictions.

6 Github repository

For further information go to the url: https://github.com/ssy00603/DATA-1030-project.git

References

[1] Raghad Alharbi. *Sephora Website Data: More than 9,000 Products with their Ratings and Ingredient Lists*, 2020 (accessed October 10, 2020).

[2] Yang Liu. *eXtreme Gradient Boosting (XGBoost): Better than random forest or gradient boosting*, 2020 (accessed November 30, 2020).