Final Project

Emily Nguyen

Introduction

This dataset is from Kaggle.com, and is titled “Sephora Products and Skincare Reviews”. The user who uploaded this dataset collected this data in March 2023 via a Python Scraper on the Sephora.com website. The dataset contains information about Sephora products, such as brand name, prices, number of reviews, ingredients, size, and more. The variables that will be used in this report are brand_name (brand of the product), loves_count (loves is when a customer adds a product to their wishlist), reviews (number of reviews), rating (rating out of 5), size (size of the product in oz), price_usd (price in $), limited_edition (whether the product is limited edition), online_only (whether the product is only available online), out_of_stock (whether the product is out of stock), and sephora_exclusive (whether the product is exclusive to Sephora). The original dataset has 8494 rows and 27 columns. After pre-processing the data to create a new data frame with only the variables used to answer the questions and dropping rows with null values, the new dataset has 6695 rows and 12 columns.

Question #1: When predicting whether a product is out of stock or not, which model (Logistic Regression or Gradient Boosting Tree) is better at determining stock availability?

In order to determine the answer to this question, I had to run both a Logistic Regression and a Gradient Boosting Tree Regression. The first model that I created was the Logistic Regression model. The predictor variables that I used were brand_name, loves_count, reviews, and sephora_exclusive, while the target variable was out_of_stock. I split the dataset into a training set and a testing set with an 80/20 split, respectively. This means that 80% of the data will used to train the model, and 20% will be used to test the model. I then one-hot-encoded the brand_name variable because it was categorical, which turns it from a categorical variable to continuous variables for each brand name. Next, I z-scored all of the variables to ensure that they were evaluated on the same scale since the variables are originally measured on different units. After pre-processing the data, I created and trained the Logistic Regression model on the training dataset. I then used the training model to make predictions on both the training and testing set. In order to assess the model’s performance, I printed out the Accuracy, Precision,

Recall, F1, and ROC AUC scores for both the training and testing sets. Accuracy measures how often the model is correct, Precision measures how many of the predicted positives are correct, Recall measures how often the model is correct for positive cases, F1 combines Precision and Recall, and ROC AUC measures the model's ability to distinguish between positive and negative cases. The model can have four different outcomes: True Positive, True Negative, False Positive, and False Negative. These four outcomes are used to calculate the metrics mentioned above. For example, Accuracy is measured by dividing the TP+TN by all predictions. To help visualize the outcomes, Figure 1 shows a Confusion Matrix of the testing set where there were 1266 true negatives, 2 true positives, 71 false negatives, and 0 false positives.

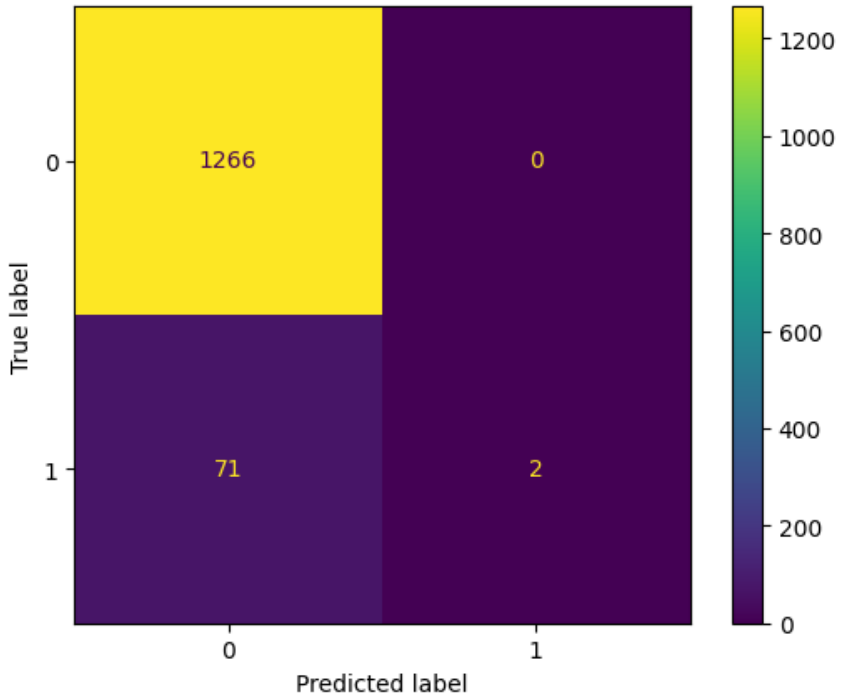

Figure 1: Confusion Matrix on Test Set for Logistic Regression Model

For this question, I will be focusing on the Accuracy and ROC AUC scores. The Accuracy score for both the training and testing set was 0.94. This indicates that in terms of how often the model was correct, the model is consistent for both sets. The ROC AUC score for the training set was 0.85 and 0.76 for the testing set. Since the score was higher for the training set, this could indicate that the model is overfitting as it performs better on data that it has seen than on data that it has not seen.

After running the Logistic Regression Model, I created and trained a Gradient Boosting Tree Model on the same dataset. Figure 2 shows the Confusion Matrix for the GBT Model of the testing set where there were 1260 true negatives, 8 true positives, 65 false negatives, and 6

false positives. The Accuracy score for both the training and testing set was 0.95. This indicates that in terms of how often the model was correct, the model is consistent for both sets. The ROC AUC score for the training set was 0.83 and 0.75 for the testing set. Again, similar to the Logistic Regression Model, the higher training set score could indicate that the model is overfitting and needs to be simplified.

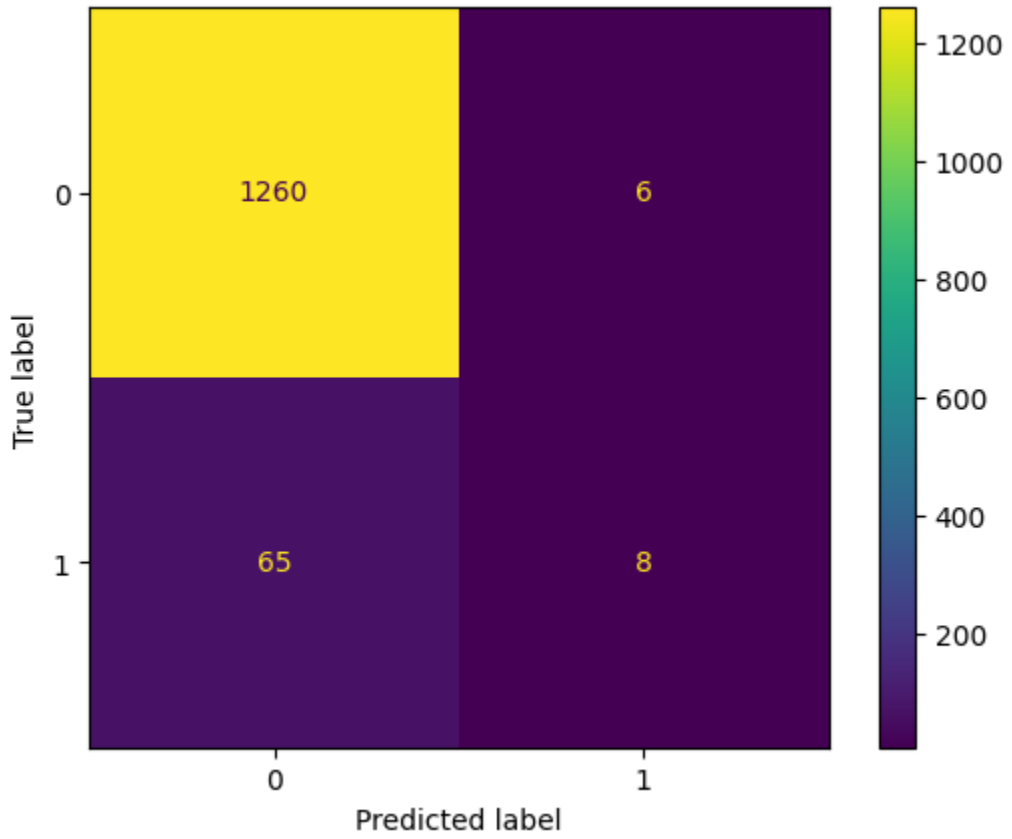

Figure 2: Confusion Matrix on Test Set for Gradient Boosting Tree Model

Now, to answer the question, we will look at the ROC AUC scores for both the Logistic Regression and Gradient Boosting Tree Models. Figures 3 and 4 show the ROC Curve for the Logistic Regression and GBT Models, respectively. The ROC Curve plot helps us visualize how well the model is performing by plotting the False Positive Rate vs. the True Positive Rate. The dashed blue diagonal line represents the ROC value if it was 0.5, meaning that it is no better than flipping a coin to determine the outcome. The solid orange line represents the tradeoff between the Sensitivity and Specificity of the model, where a curve further from the diagonal line and closer to the top left corner indicates a better model. The area under the orange line represents how well the model is at distinguishing between a product being out of stock or not. A higher AUC (area under the curve) value indicates a better-performing model, where a

value of 1 indicates that the model is perfect. As you can see in Figure 3, the AUC value (0.76) for the Logistic Regression model is slightly higher than the AUC value (0.75) for the Gradient Boosting Tree model. This suggests that the Logistics Regression model is better suited at predicting whether a product is out of stock at Sephora. One reason for this could be that Logistic Regression assumes linearity, while Gradient Boosting Tree is more geared towards complex and non-linear relationships.

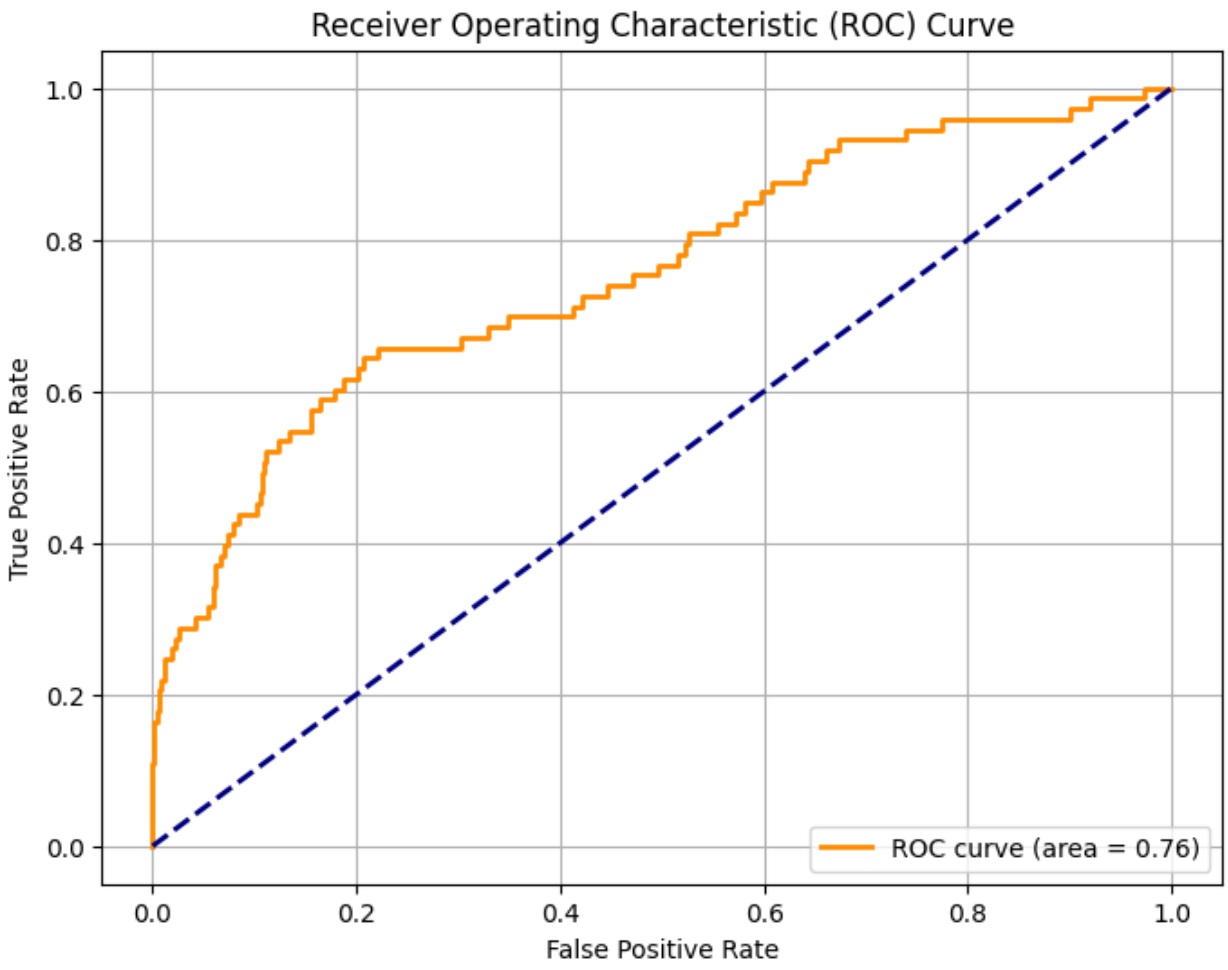

Figure 3: ROC Curve Plot for Logistic Regression Model

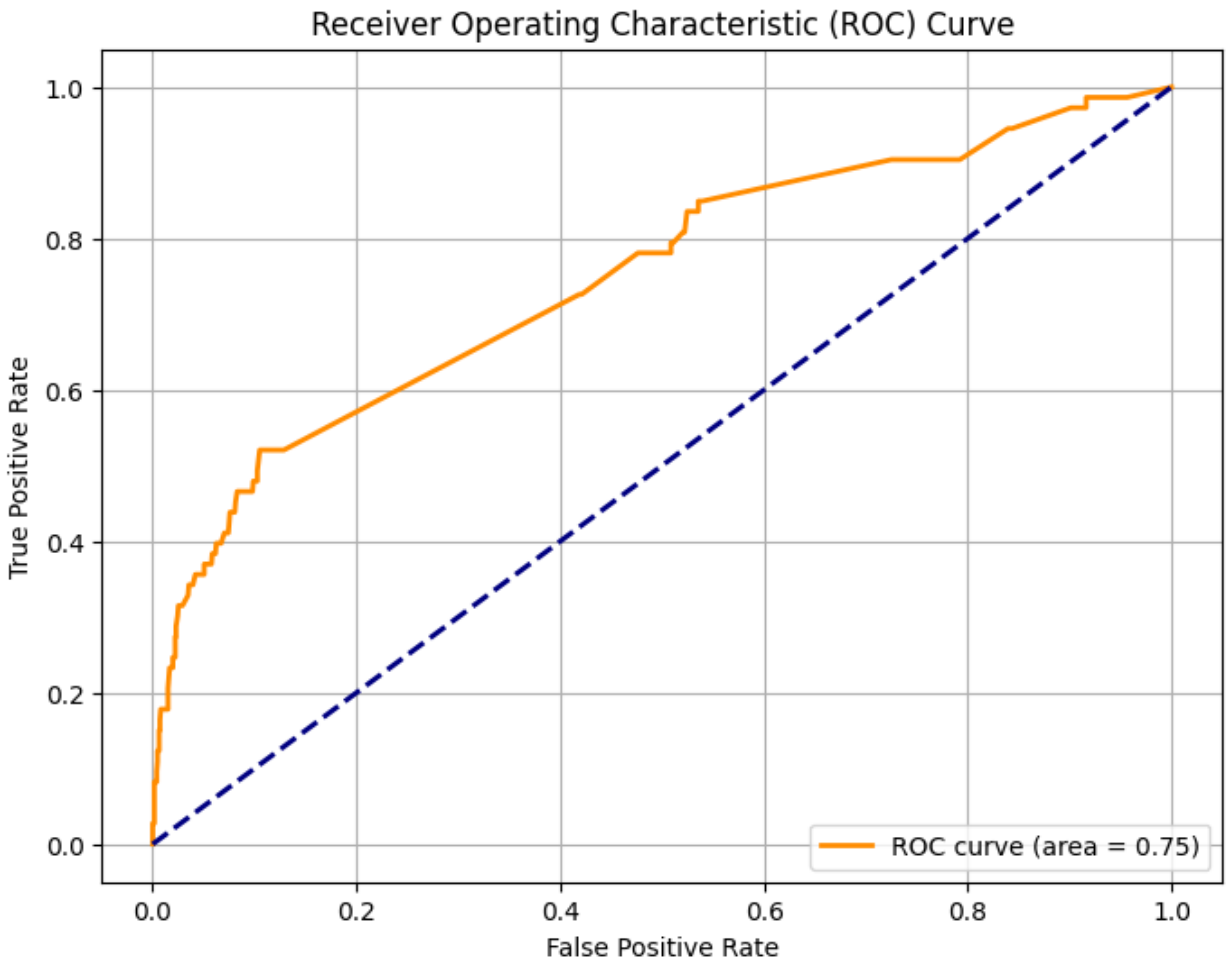

Figure 4: ROC Curve Plot for Gradient Boosting Tree Model

However, while the ROC AUC values for both models indicated that it was a well-performing model, there are some limitations. The first limitation is the imbalanced dataset that was used. Out of the 6695 rows, only 389 were out of stock, which is about 5.8% of the data. This could lead to bias towards the majority class (not out of stock), which is what we see in the model. Additionally, the imbalance could effect the ability of the model to correctly identify the outcomes for the minority class (out of stock). In the future, using resampling techniques or class weighting could help mitigate this issue.

Question #2: Which features have the most significant impact on predicting the price of a product, and can we identify a subset of important features using LASSO regression?

To answer this question, I built two models, a Linear Regression and LASSO Regression model. The predictor variables for both models were rating, brand_name, loves_count, reviews, limited_edition, and online_only. The target variable was price_usd. Similar to the previous question, I did the same data pre-processing before creating the models, including one hot encoding brand_name, splitting the model for validation, and z-scoring the predictor variables. After creating and fitting both the Linear Regression model and LASSO Regression model on the training dataset, I assessed the models by calculating their Mean Squared Error and R-Squared values. Mean Squared Error (MSE) measures the average squared difference between the predicted and actual values. In other words, how far, on average, the model's predictions are from the actual price of the product. R-Squared tells us how well the model's predictions explain the variances in prices. The R-Squared value ranges from 0 to 1, where a value of 1 means that the model is perfect at explaining the price of a product.

The Mean Squared Error value for the training set on the Linear Regression model was 963.81, and 1091.41 for the testing set. A lower value indicates that the model performs better, and since the training set has a lower MSE, it performed better than the testing set. This reveals that the model could be overfitting since it does better on data it has seen than data it has not seen. As for the LASSO Regression model, the Mean Squared Error for the training set was 968.16, and 1100.03 for the testing set. Similar to the Linear Regression model, the LASSO model is overfitting because the training set has a lower MSE value. Figure 5 shows a comparison bar chart of the MSE values for both sets of both models.

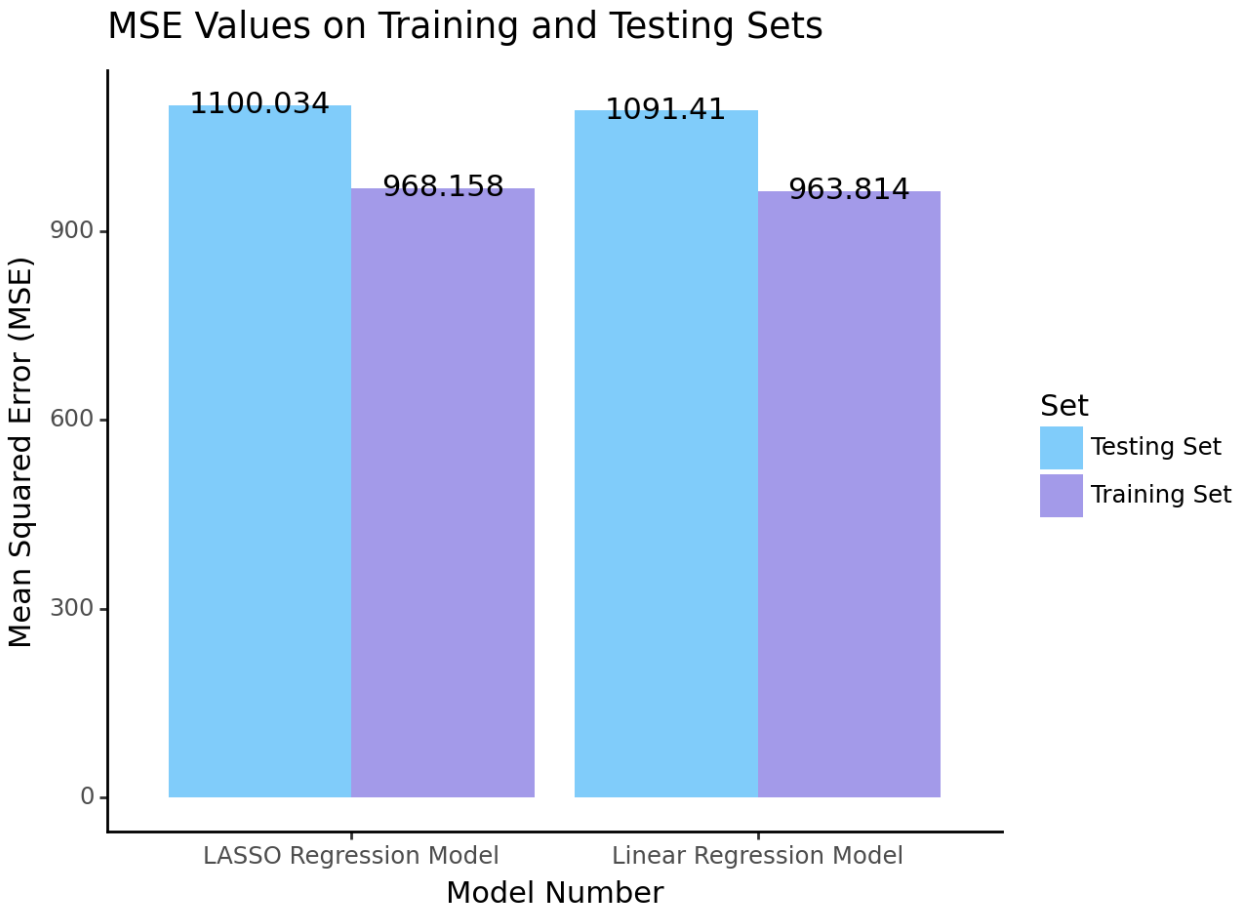

Figure 5: MSE Values on Training and Testing Sets for Both Models

The R-Squared value for the training set on the Linear Regression model was 0.534, and 0.544 for the testing set. Since the testing set performs better than the training set, this indicates that the model is generalizing reasonably well to data it has not seen. As for the LASSO Regression model, the R-Squared value for the training set was 0.532, and 0.540 for the testing set. These values are almost identical to the R-Squared values in the Linear Regression model. Figure 6 shows a comparison of the R-Squared values for the training and testing set for both models.

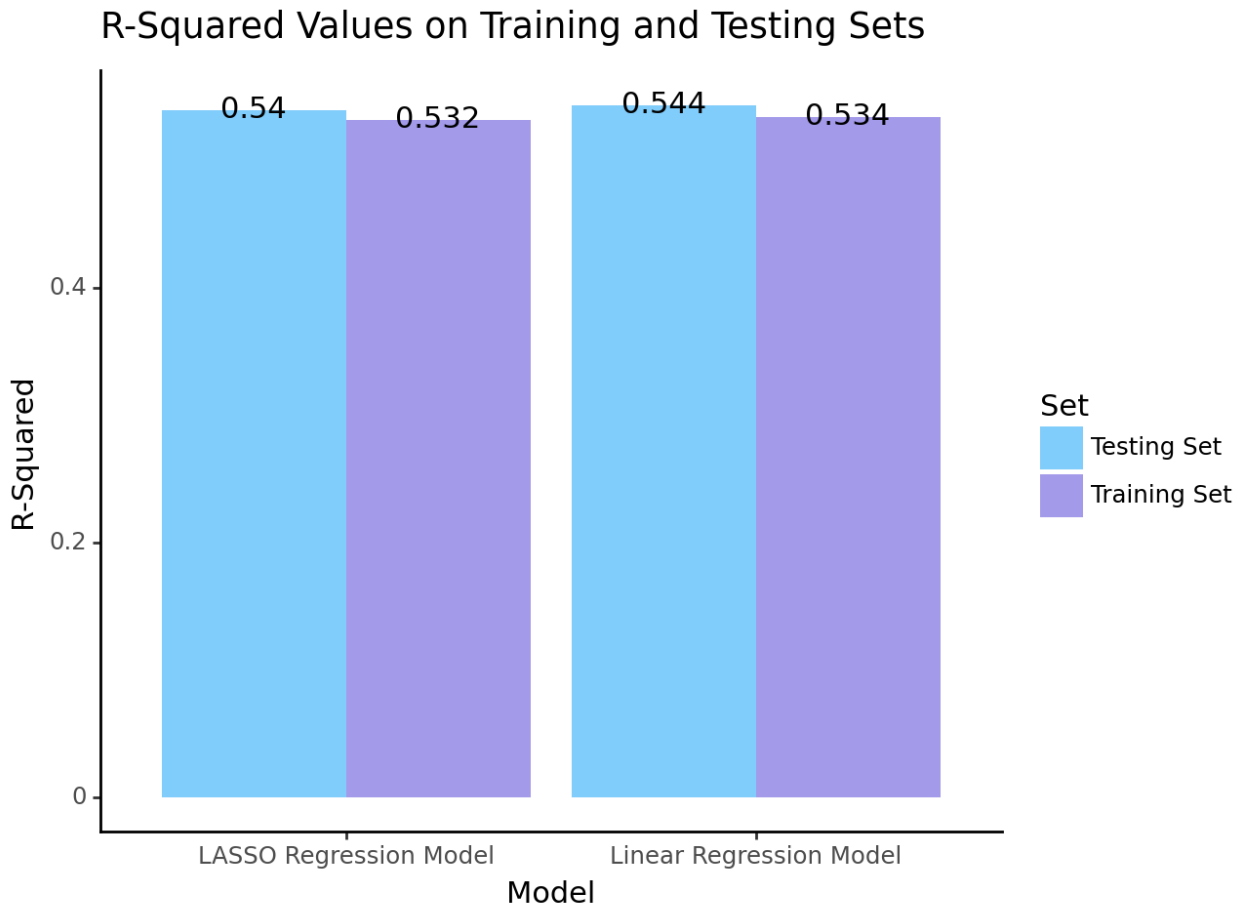

Figure 6: R-Squared Values on Training and Testing Sets for Both Models

Looking at Figure 5 and Figure 6, we can see that the Linear Regression Model performs ever so slightly better than the LASSO Regression Model. The difference in performance between the two models is not large at all, and if we were to only choose one in terms of predicting price, the Linear Regression model would be a better choice, especially since it runs much faster than the LASOO model. However, by only looking at the Linear Regression model, the question of which features have the most impact on predicting the price is not answerd. This is why we will use the LASSO Regression model. LASSO is a regularization technique that is meant to make the model simpler. It penalizes the sum of the absolute value of the coefficients and drags coefficients that don't "pull their weight" or have significant impact exactly to 0.

Figure 7 shows a chart of the top 10 features with the best performing coefficients from the LASSO model. They are regarded as the top performing because they have the largest coefficient value, meaning that they have the most significant impact on predicting the price of a product. As shown in Figure 7, all of the features are related to the brand name of the

product indicating that the brand of the product is an important factor in determining the price of the product. On the other hand, Table 1 shows all feature variables that have a coefficient value of 0. This means that these variables are not significant to the model, so they are completely removed from the model. Similar to the top 10 performing variables, these variables are also related to the brand of the product. This suggests that these brands do not play a significant role in determining the price of a product.

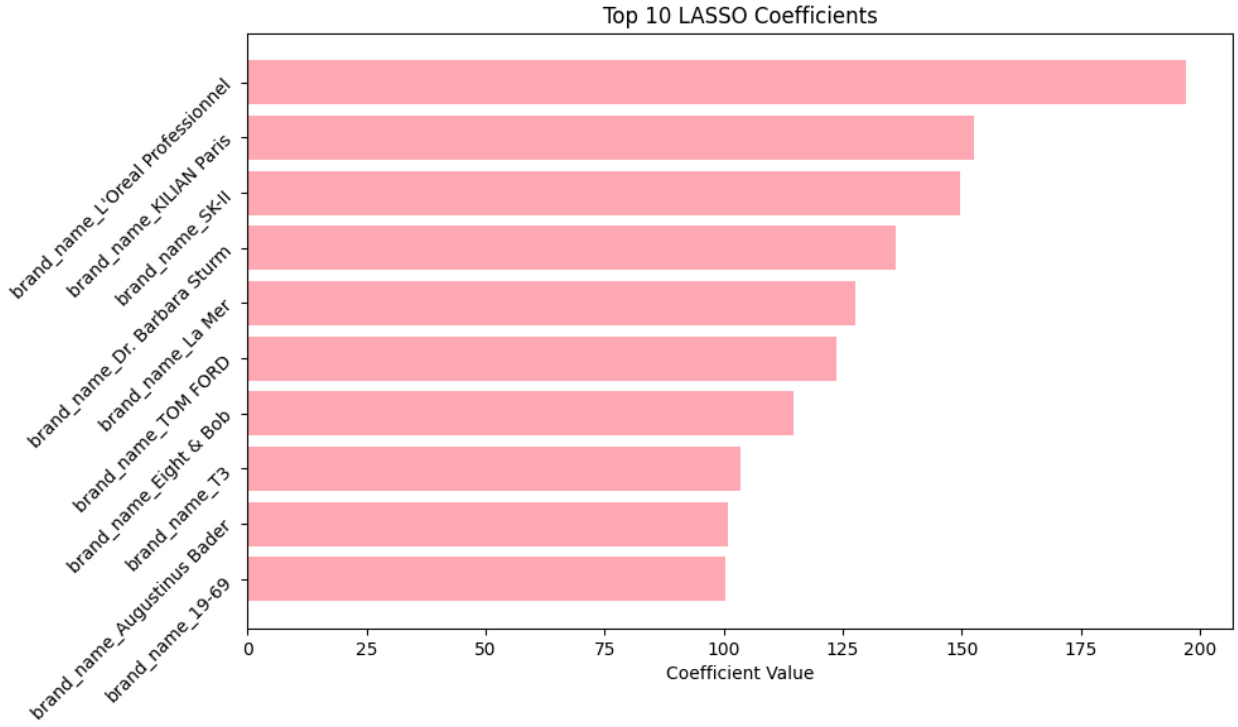

Figure 7: Top 10 Performing Variables from LASSO Model

Feature	**Coefficient**
brand_name_Blinc	0
brand_name_COOLA	0
brand_name_Crown Affair	0
brand_name_DAMDAM	0
brand_name_DOMINIQUE COSMETICS	0
brand_name_Dame	0

brand_name_GLO Science	0
brand_name_Hanni	0
brand_name_Hyper Skin	0
brand_name_Josie Maran	0
brand_name_K18 Biomimetic Hairscience	0
brand_name_KORRES	0
brand_name_Laura Mercier	0
brand_name_Lilly Lashes	0
brand_name_Mount Lai	0
brand_name_OLEHENRIKSEN	0
brand_name_OTHERLAND	0
brand_name_Origins	0
brand_name_Overose	0
brand_name_RIES	0
brand_name_Rahua	0
brand_name_Rossano Ferretti Parma	0
brand_name_SIMIHAZE BEAUTY	0
brand_name_Slip	0
brand_name_St. Tropez	0
brand_name_The Original MakeUp Eraser	0
brand_name_The Outset	0
brand_name_Violet Voss	0
brand_name_Viseart	0
brand_name_Wishful	0

brand_name_alpyn beauty	0
brand_name_fresh	0

Table 1: Table of All Variables with Coefficient Value of 0

While this model was able to identify a set of features that had the most significant impact on predicting the price of a product, there are a couple limitations. The first limitation is that the model's R-Squared value was 0.54 for the LASSO Regression Model. This means that the model only captures 54% of the total variability of predicting the price of the product. While this is not a terrible score, we try to aim closer to a value of 1 when calculating the R-Squared. Additionally, since there were so many brand names, the other feature variables got lost in the mix. In the future, I would run the same models but omit the brand_name variable to see if the other feature variables were significant in predicting the price of the product.

Question #3: When considering rating, size, and price, what clusters emerge, and what kind of products are in those clusters?

As specified in the question, the variables used were rating, size, and price_usd. The data pre-processing methods were different than what was done for the previous two questions. The first thing that I had to do was clean the size column and extract the oz value. The size column's data type was String in the original dataset. Examples of some of the values were "3.3 oz/100 mL" or "2.6 oz". I extracted the number in front of the 'oz' and converted it to a numeric type. If the string did not include an oz value, the entire row was dropped from the dataset. I took all of these extracted values and put them into a new column, extracted_size. After doing the data pre-processing, I was ready to start building the clustering models.

Before choosing which clustering model to use, I plotted a scatter plot of every combination of each variable (Figure 8). I did this to visualize the relationships and see which clustering model would perform the best. I decided to do a K-Means Clustering Model to answer the question. After z-scoring the variables, I had to decide how many clusters I would choose for the K-Means model. To help decide the optimal number of clusters, I created a Within Clusters Sum of Squared Errors plot and a Silhouette Score for different k clusters plot. The Within Clusters SSE plot measures the sum of squared distances between each data point for different values of k (number of clusters). To utilize this plot, shown in Figure 9, to determine the number of clusters to apply, we use the elbow method. The elbow method essentially

means to look for an “elbow” in the plot, and that is the point that we will use. In this case, the “elbow” looks to be around x=2. The Silhouette Scores for Different Ks plot (Figure 10) shows exactly what it says, the different Silhouette Scores for different number of clusters. Silhouette Scores evaluate the performance of a clustering model by measuring cohesion and separation. In other words, whether a data point is close to its own cluster compared to other clusters. Silhouette Scores range from -1 to 1 where a negative value indicates that the data point is assigned to the wrong cluster and a score of 1 means the model is perfect. This means that we want to choose the k value with the highest Silhouette Score. While I could have simply looked at the graph to determine the highest score, I wanted to be precise, so I printed the value of k with the highest Silhouette Score. In this case, it was a k value of 2, which aligned with the Within Clusters SSE plot as well.

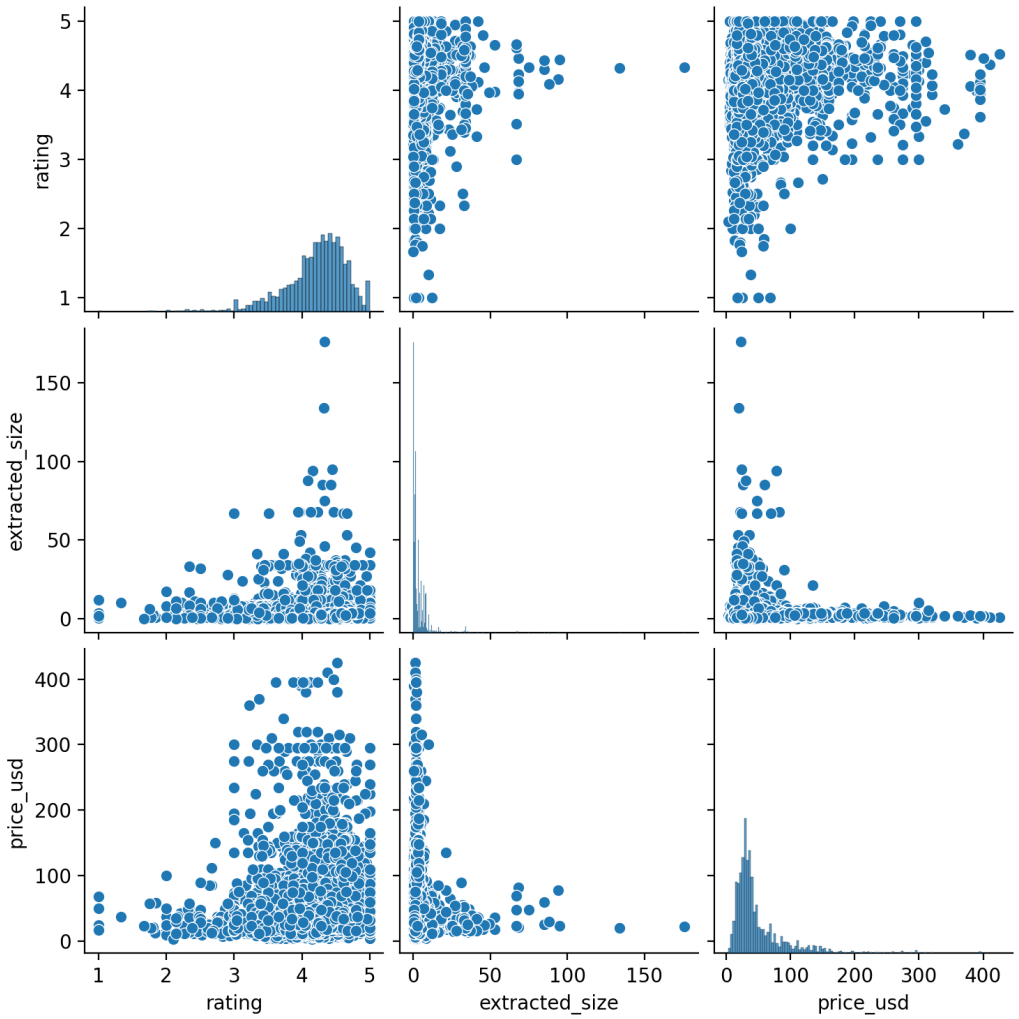

Figure 8: Scatterplots of Pairs of Each Variable Combination

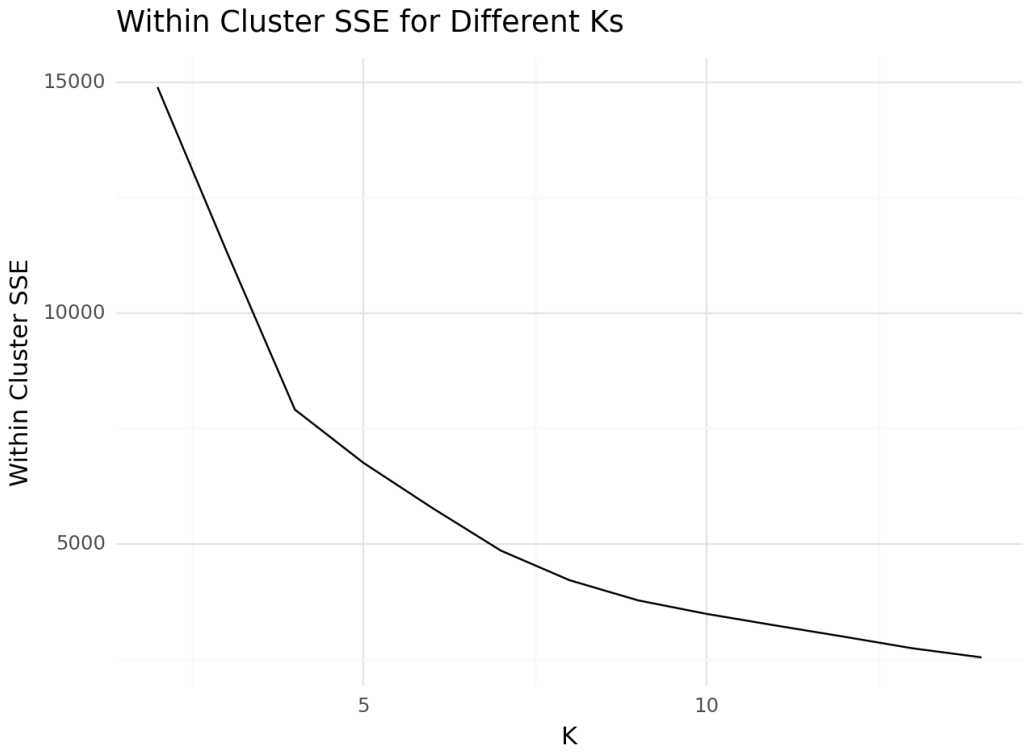

Figure 9: Within Cluster SSE for Different Ks

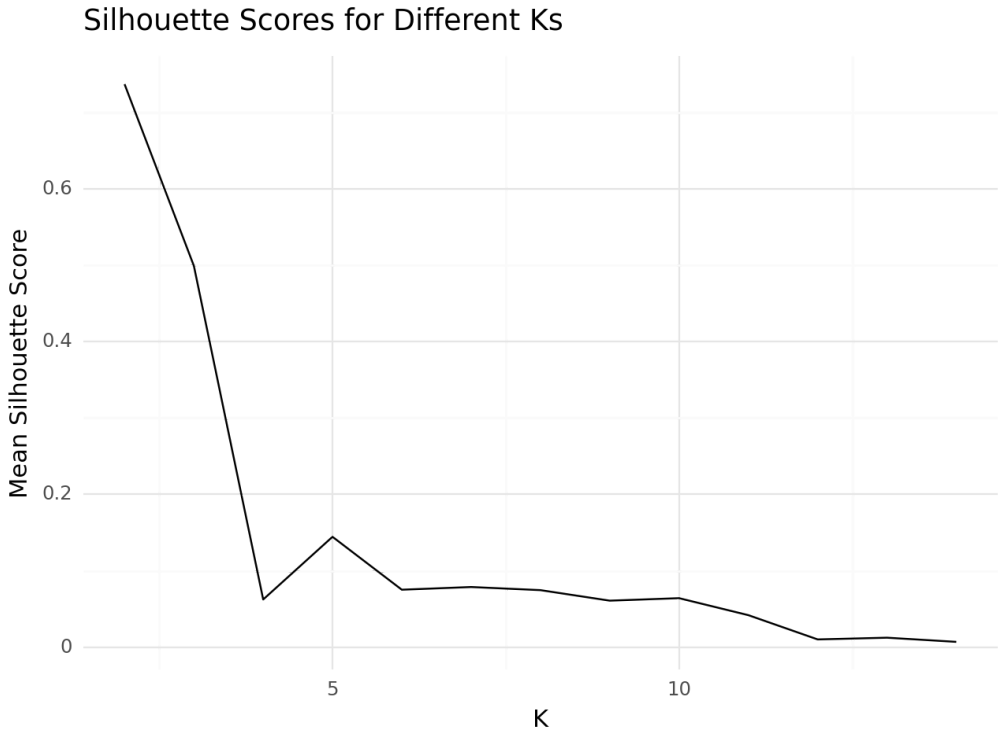

Figure 10: Silhouette Scores for Different Ks

After choosing the number of clusters, I created the K-Means clustering model with 2 clusters. I showed the clusters on a scatterplot using the rating vs. price_usd relationship (Figure 11). I also calculated the Silhouette Score, which turned out to be 0.74. This is suggests that the clusters are well separated and that the data points each cluster are relatively close to each other.

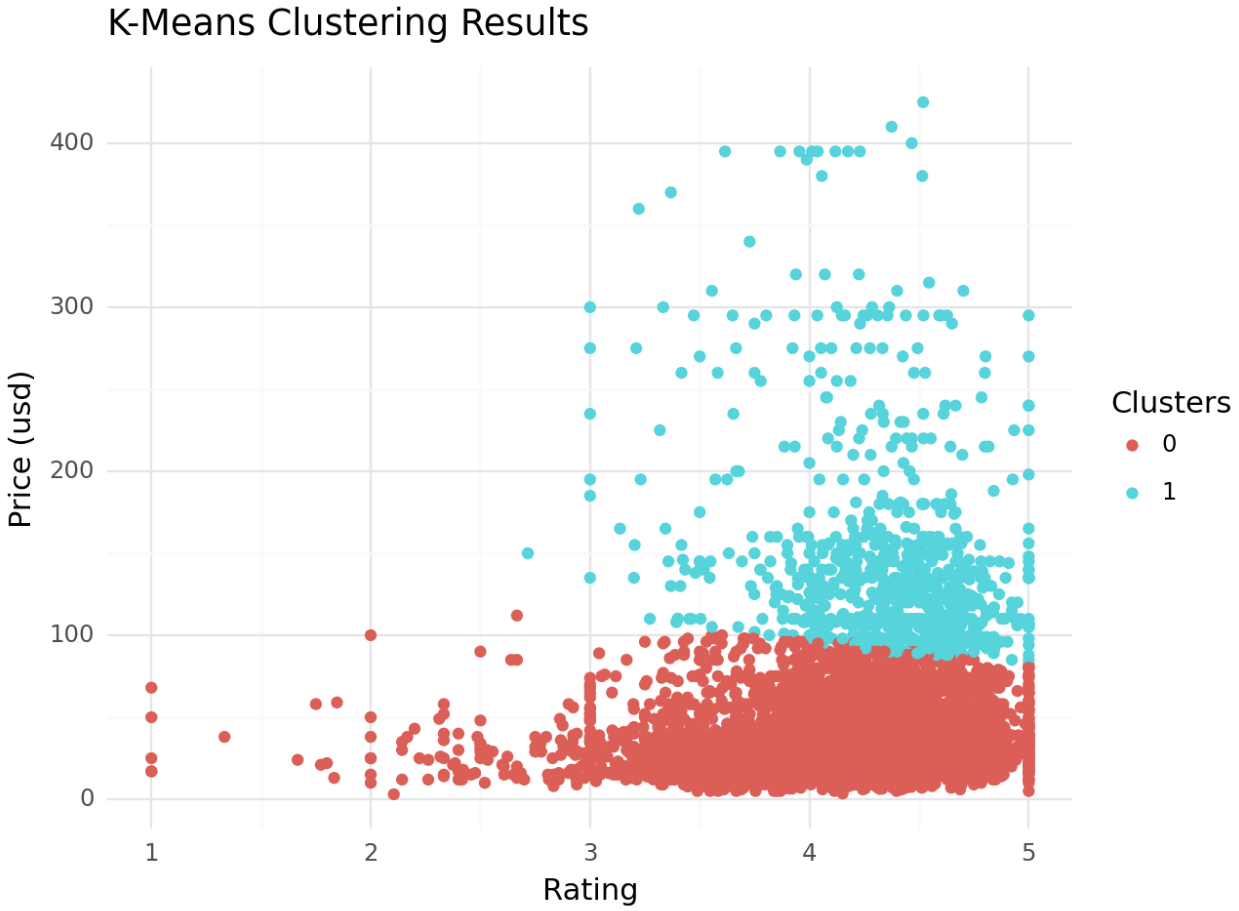

Figure 11: K-Means Clustering Model with 2 Clusters

While the K-Means clustering model had great results, I wanted to see if creating a Gaussian Mixture Model would perform better. In order the determine the optimal number of clusters for the GMM model, I created a BIC for Different Ks plot (Figure 12). BIC or Bayesian Information Criterion measures how well fit the model is, where lower values of BIC are better. Similar to the Silhouette Scores plot, I calculated and printed out the value of k with the lowest BIC score, which turned out to be 12 clusters.

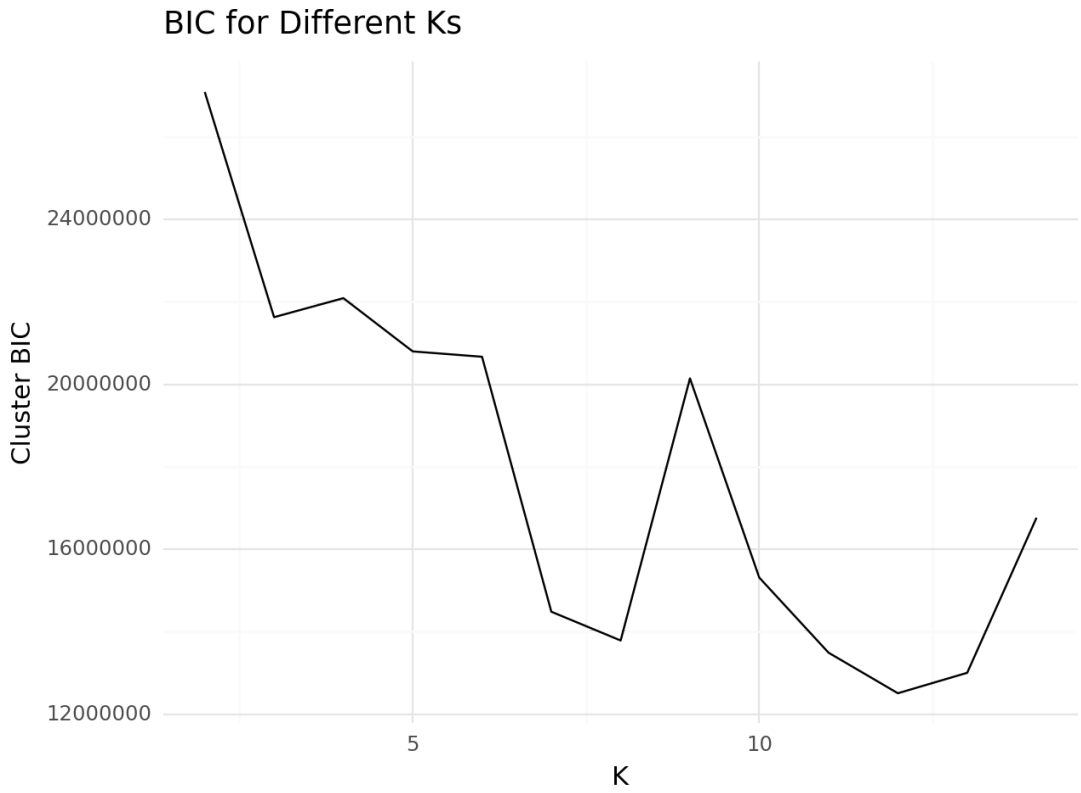

Figure 12: BIC for Different Ks

After determining the optimal number of clusters, I created a GMM model with 12 clusters. Similar to the K-Means model, I showed the clusters on the rating vs. price_usd relationship (Figure 13). I also calculated the Silhouette Score, which turned out to be 0.07. This value is significantly lower than the Silhouette Score for the K-Means clustering model, which indicates that that model has a much better performance than this model.

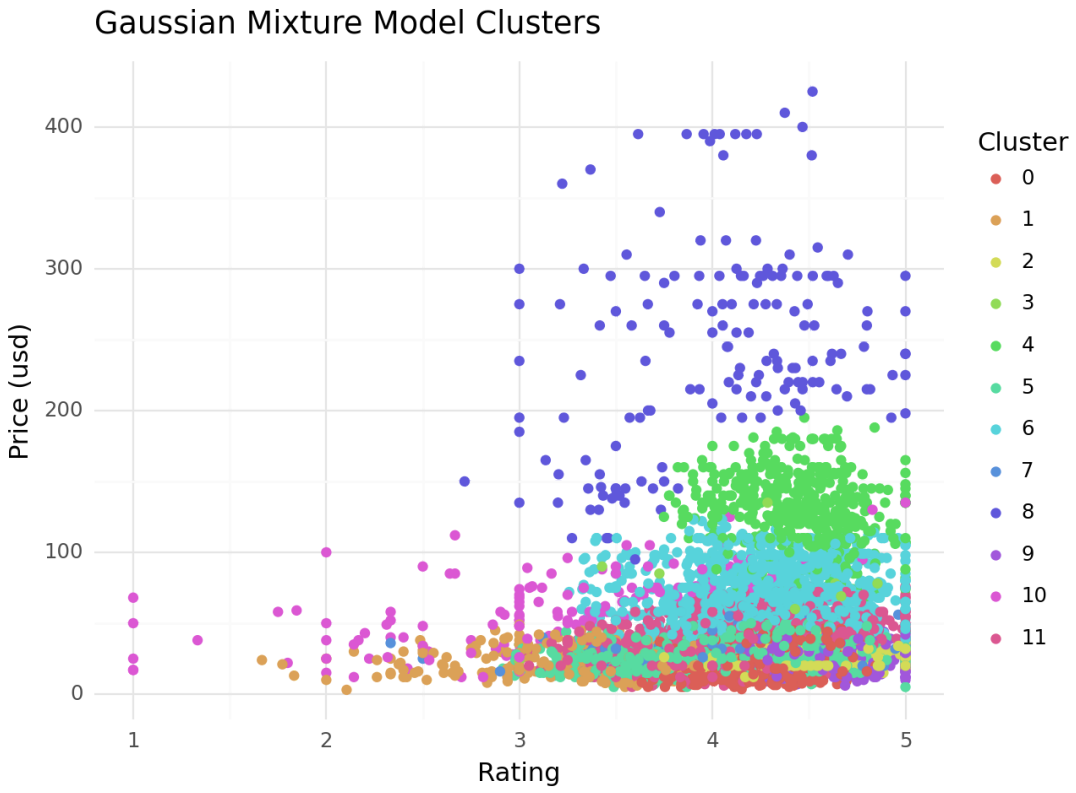

Figure 13: Gaussian Mixture Model with 12 Clusters

Since the K-Means model performed much better than the Gaussian Mixture Model, I created a cluster summary table (Table 2) to answer the question of which type of products are in each cluster. In the first cluster, products tend to have a rating around 4.35, a price of $148.59, and the size of the product is 2.37 oz. In the second cluster, the products tend to have a rating of 4.19, a price of $36.67, and the size of the product is 3.76 oz. While the rating number is relatively close for both clusters, the price difference is well over $100, which is quite large. In the first cluster, products are more expensive and smaller. Using my domain knowledge, I am going to assume that most of the products in this cluster are most likely perfume because perfume tends to be very expensive and comes in relatively small bottles. The second cluster has more inexpensive products compared to the first cluster. The products in the second cluster are also slightly bigger in size. Overall, the biggest difference between the two clusters is the price of the products.

clusters	rating	price_usd	extracted_size
0	4.35	148.59	2.37
1	4.19	36.67	3.76

Table 2: Cluster Summary using K-Means Model

The insights from the type of products in each cluster can help Sephora create product profiles for targeted marketing strategies. For example, if they know that a customer tends to purchase products in the first cluster, the company can personalize that customer's advertisements to reflect products in that cluster. While this model can be useful in those kinds of situations, I think in the future I would look at different variables and see if more distinct clusters emerge since two clusters is very small considering the number of products in each cluster. It would be nice to see a more in-depth analysis of the types of products in each cluster.