Sephora Product Ratings Regression Report

Abigail Ajamian

Course: D214

10/28/2024

Research Question

Sephora is one of the leading companies in e-commerce beauty sales. Antonia Tonnies says Sephora is "the world's number one online beauty retailer." It is projected that e-commerce beauty sales will continue to proliferate, with a revenue prediction of 667.7 billion US Dollars by 2028 (Tonnies, 2024). With the steady upward trend in the e-commerce industry for beauty, it is in Sephora's best interest to continue to be the leading beauty website.

The goal of this analysis is to support Sephora's insight into customer satisfaction with products on the website and allow Sephora to implement data-driven strategies to support customer retention, satisfaction, and marketing decisions.

The research question is, "Can an ordinal logistic regression be performed to determine product characteristics that influence the rating of a product?" The null hypothesis is that an ordinal logistic regression model cannot be made using this data to determine product characteristics that influence the rating of a product. The alternative hypothesis is that an ordinal logistic regression model can be created to determine product characteristics that influence the rating of a product with an accuracy > 70%.

Data Collection

The data collected for this study was from Kaggle.com, an open-source website. This dataset includes 8,494 products sold on Sephora's online website as of March 2023. The dataset contained 27 columns of quantitative and qualitative data about each product, which will be further discussed in the preparation and extraction section.

The data quality is acceptable, which was recognized by Kaggle's usability score for the datasets. A completeness, credibility, and compatibility score are used to calculate this score. The chosen dataset received a score of 10.00, representing 100% in all three categories. One limitation is that Kaggle's usability score does not consider data sparsity, which is 19.35% for this dataset. To overcome this limitation, removal, and imputation were completed in the data preparation step. Data imputation is a method used to "fill in missing values in the dataset" (Vishwakarma, 2023). An advantage of using Kaggle is that it is an open-source website with vast options for finding datasets that can be downloaded locally, eliminating the need to scrape the data yourself.

Data Extraction and Preparation

Python was used to extract and prepare the dataset and complete the analysis. According to Mariana Berga, Python is a general-purpose and multi-paradigm language, meaning that Python is an "object-oriented, functional, and structured programming" language that can be used for various purposes. These characteristics of Python and the vast set of libraries available will be an advantage when completing this analysis (Berga, 2024).

Before starting the data cleaning process, the libraries needed were imported. This included pandas, matplotlib, seaborn, sklearn, numpy, and scipy.stats. Then Pandas' read_csv function was implemented to load the dataset since the data set is a CSV file.

```
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import KMeans


df = pd.read_csv('product_info.csv')
```

The next step in preparing the data is cleaning the dataset of duplicates, missing values, and outliers. The first step was to identify if the dataset consisted of any duplicate observations, which was accomplished by using pandas' duplicated function in conjunction with the pandas' value_counts function. The outcome of using these functions together outputs how many observations that are and are not duplicates. This dataset consisted of zero duplicate observations, resulting in no treatment.

```
# Find Duplicates
df.duplicated().value_counts()
# No duplicates found = No Treatment
```

After the duplicate observations were treated, the missing values of the dataset were detected. This was achieved by combining the isnull and sum functions, which will return a list of every column and how many missing values are in each. As mentioned, many missing values in the dataset must be treated.

```
df.isnull().sum()
```

This dataset included a few columns that were primarily missing values and were not vital to the analysis. Jahnavi C. specified that if columns have a high percentage of missing values, they "will be of no use while predicting the target or output, as they don't contain any information about the data" (C, 2022). So, instead of imputing these values, the whole column was dropped using the drop function. The columns dropped included variation type, variation value, variation desc, ingredients, value price USD, sale price USD, highlights, tertiary category, child max price, and child min price.

```
# Drop columns with too many missing values
df = df.drop(['variation_type','variation_value','variation_desc','ingredients','value_price_usd','sale_price_usd','highlights','tertiary_category','child_max_price','child_min_price'], axis=1)
```

Next, the dependent variable had about 3% missing values; the observations that had missing values were removed. This also removed the missing values that were in the review column. Then, missing values in the column labeled secondary category were imputed with the mode since the variable is categorical. After that, the missing values in size were represented as 0 because the observations with missing size values were due to the product's lack of size. Lastly, the isnull and sum functions were utilized again to verify no more missing values in the dataset.

```
# Remove all rows with missing values in rating column: This should also remove all missing values in the review column
df = df.dropna(subset = ['rating'])
```

```
 # Impute secondary_category with mode
 # Not imputing missing values for size: During encoding missing value will be represented by 0
 df['secondary_category'] = df['secondary_category'].fillna(df['secondary_category'].mode()[0])
```

```
# Not imputing missing values for size: During encoding missing value will be represented by 0
df['size'].fillna(0,inplace=True)
```

```
df.isnull().sum()
```

Next, identifying and treating outliers is required. To identify the outliers, seaborn's boxplot function and scipy.stats' z-score function was used. Only the numerical variables, including price_usd and child_count, were searched for outliers. The price variable had many outliers, although only the max outlier of 1,900 was imputed with the median. This is due to background knowledge that Sephora has luxury products with prices that are shown to be outliers but, on average, are expected to be less than $700. The method of imputation to child_count was to impute all outliers with a z-score greater than three with the median.

```
 # Find outliers with boxplot
 sns.boxplot(df['price_usd'])
 print('Stats with Outliers: ',df['price_usd'].describe())
 print(df['price_usd'].median())

 # Dont need z_score: remove only max outlier
```

```
# Remove all above 700
df['price_usd'] = np.where(df['price_usd'] > 700,np.nan,df['price_usd'])
df['price_usd'].fillna(df['price_usd'].median(), inplace = True)
sns.boxplot(df['price_usd'])

#Verify after treatment
print('Stats with Imputed Outliers: ',df['price_usd'].describe())
print(df['price_usd'].median())
```

```
#Find z_score : new column will be deleted later
df['z_child'] = stats.zscore(df['child_count'])

#Visualize before treatment
sns.boxplot(df['child_count'])
print('Stats with Outliers: ',df['child_count'].describe())
print(df['child_count'].median())
```

```
#Impute with median where z_score > 3
df['child_count'] = np.where(df['z_child']>3,np.nan,df['child_count'])
df['child_count'].fillna(df['child_count'].median(), inplace = True)

#Verify after Imputation
sns.boxplot(df['child_count'])
print('Stats with Imputed Outliers: ',df['child_count'].describe())
print(df['child_count'].median())
```

Next in the data-prepping process is data wrangling. This involves encoding categorical variables to be represented by numbers. Most of the categorical variables in the dataset were already categorized by numbers (i.e., 0=No, 1=Yes). However, primary_category, secondary_category, and size needed to be encoded. To encode the primary_category and secondary_category columns, sklearn's preprocessing package was used for the Label Encoder function. These variables were fit and transformed using this function, then stored in a new column with the variable name plus "label" at the end.

```
# Label encode primary_category
encoder = LabelEncoder()
df['primary_category_label'] = encoder.fit_transform(df['primary_category'])
```

```
# Label encode secondary_category

df['secondary_category_label'] = encoder.fit_transform(df['secondary_category'])
```

The size variable had too many different values to use the Label Encoder function. To encode size, the top 4 size values were represented by 1-4; if a product had no size, it was represented by 0, and 5 represented the products with other sizes.

```
# Change labels for size
# no size = 0 , 0.5 oz/ 15 mL = 1, 1 oz/ 30 mL = 2, 1.7 oz/ 50 mL = 3, 3.4 oz/ 100 mL = 4 , other = 5
df['size'] = df['size']
dict_over = {'size':{"0.5 oz/ 15 mL":1,"1 oz/ 30 mL": 2,"1.7 oz/ 50 mL":3,"3.4 oz/ 100 mL":4}}
df.replace(dict_over, inplace = True)
df['size']

for i in df['size']:
    if i not in (0,1,2,3,4):
        df['size'] == df['size'].replace(i,5, inplace= True)
print(df['size'])
```

In addition to encoding the predictor variables, the dependent variable needed to be grouped. This is because the analysis implements an ordinal regression, which means order matters. The method used to complete this is KMeans clustering, with the number of clusters being 5. This was then stored in a new column named rating_clusters.

```
# Use Kmeans clustering to identify groups of ratings
# 5 clusters for ratings 1-5

X = df[['rating']].values.reshape(-1,1) #reshape single-column bc kmeans expects 2D array

kmeans = KMeans(n_clusters=5,random_state=0)
kmeans.fit(X)

labels = kmeans.labels_
centroids = kmeans.cluster_centers_
```

```
(variable) kmeans_df: DataFrame
kmeans_df = pd.DataFrame(centroids)
kmeans_df
```

```
#Change labels
kmeans.labels_[kmeans.labels_ == 0] = 10
kmeans.labels_[kmeans.labels_ == 1] = 7
kmeans.labels_[kmeans.labels_ == 2] = 9
kmeans.labels_[kmeans.labels_ == 3] = 6
kmeans.labels_[kmeans.labels_ == 4] = 8
```

```
#Change labels to represent 1-5 stars
kmeans.labels_[kmeans.labels_ == 10] = 5
kmeans.labels_[kmeans.labels_ == 9] = 4
kmeans.labels_[kmeans.labels_ == 8] = 3
kmeans.labels_[kmeans.labels_ == 7] = 2
kmeans.labels_[kmeans.labels_ == 6] = 1
```

```
df['rating_clusters'] = labels
df[['rating','rating_clusters']]
```

The last step of the data preparation was to remove all columns no longer needed for analysis and export the cleaned dataset. The removed columns are product_id, brand_name, loves_count, product_name, and z_child.

```
df = df.drop(['product_id','brand_name','loves_count','product_name','z_child'], axis=1)
```

```
#Export
df.to_csv('cleaned_data.csv')
```

Analysis

Exploratory data analysis, or EDA, is the first step in the analysis process. EDA is "a good practice to understand the data first and try to gather as many insights from it" (Patil, 2022). Completing EDA allowed for a better understanding of the dataset before accomplishing the logistic model. To begin the EDA process, matplotlib's pyplot package was needed to extract univariate statistics.

After extracting univariate statistics, bivariate analysis was completed. This process involves creating visualizations that reveal patterns of the dependent variable based on the predictor variable. The graph type created depended on the data type of the predictor variable. For example, if the variable was continuous data, seaborn was used to create a boxplot visual. The univariate and bivariate analysis aims to gain insight into data patterns and determine if a "statistical association exists between two variables" (Sandilands, 2014).

Next, the ordinal logistic regression model was created for analysis. First, the predictor and dependent variables were split into training and testing sets, with a 70-30 split. Python's mord library was used to create the initial model with the LogisticIT function. This was then fitted to the x_train and y_train datasets. Sklearn's metrics package was used for the classification report function to determine the model's fitness, which showed an accuracy of .39, so 39% accuracy when predicting. This shows that this model is not a good fit.

Feature reduction was implemented to explore further if this model could be a good fit. This was done using the SequentialFeatureSelector function from mlxtend.feature_selection, which selected the top 4 features based on the accuracy score. These features were then used to create a reduced model using the same process. Although the accuracy decreased from 39% to 38%, showing feature reduction did not improve the fit of this model.

A multinomial logistic regression was also completed to determine if the data satisfies the ordinal logistic regression assumption of proportional odds due to Python not having a Brant-Wald test. If the data does not satisfy the assumption, it will show that the multinomial logistic regression is a better fit. This was done using the LogisticRegression function from sklearn.linear_model, with the multiclass attribute equal to multinomial. This model was then fitted to the x_train and y_train datasets and was used to make predictions using x_test. The predictions were then compared to the actual y_test values to determine accuracy. The accuracy was 39%, which is not an improvement from the ordinal logistic regression. Therefore, it is concluded that the proportional odds assumption does not cause a poor fit of the ordinal regression model.

Data Summary and Implications

Based on the analysis outcomes, we failed to reject the null hypothesis. Although the top four predictor variables were identified through feature selection, this model shows that the predictor variables have little statistical influence on the ratings of the products.

There are two approaches for further research. The first is to gather more data on Sephora products that can be joined with this dataset and then complete the ordinal logistic regression again. Sephora has an extensive number of products that could be used for this analysis, and in a smaller dataset, it can be challenging to detect statistical significance and relationships (Analytics, 2024). The second approach is to create a sentiment analysis using the dataset's written reviews. Since the goal of the analysis is to gain insight into customer satisfaction and

make data-driven decisions, a sentiment analysis will allow the opportunity to achieve these goals.

Sources

Analytics, E. I. (2024, January 2). *Decoding data size: Pros and cons of working with small data sets*. Medium. https://medium.com/@analyticsemergingindia/decoding-data-size-pros-and-cons-of-working-with-small-data-sets-bc1ea0792da6

Berga, M. (2024, February 25). *The 6 must-know advantages of python*. Imaginary Cloud: Software Development for Digital Acceleration. https://www.imaginarycloud.com/blog/advantages-of-python

C, J. (2022, March 9). *Data Cleaning - remove columns*. Mage. https://www.mage.ai/blog/data-cleaning-remove-columns

Patil, P. (2022, May 30). *What is exploratory data analysis?*. Medium. https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15

Sandilands, D. (. (2014). Bivariate Analysis. In: Michalos, A.C. (eds) Encyclopedia of Quality of Life and Well-Being Research. Springer, Dordrecht. https://doi.org/10.1007/978-94-007-0753-5_222

Tonnies, A. (2024, August 21). *Top beauty retailers: Growth trends & ecommerce revenue: ECDB.com*. Top Beauty Retailers: Growth Trends & eCommerce Revenue - ECDB. https://ecommercedb.com/insights/top-online-beauty-stores-in-the-global-care-products-market-sephora-leads/4876

Vishwakarma, S. (2023, January 9). *Guide to dealing with sparse datasets?*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2022/12/guide-to-dealing-with-sparse-datasets/