Amazon Product Review Analysis

Predicting sales ranking from product reviews

1. Introduction

a) Problem Statement

Amazon reviews provide customers an opportunity to hear what other purchasers think of a product before they spend their cash on a product. Our goal is to develop a model that will predict sales ranking of a product based on the reviews, rating, and price of the product.

b) Background

Amazon.com is the world's largest online marketplace with millions of third-party sellers listing over 12 million products, it is impossible to imagine a company launching a new product and not considering selling the product on Amazon. However, in addition to providing a worldwide marketplace to sellers, Amazon also provides a simple and commonly used review platform that allows customers to voice their opinion of products to the world. In addition to providing purchasers with the opportunity to provide a "start rating" from one to five, the review platform also allows purchasers to write a summary and full-length review of the product. Imagine your company just launched a new home and kitchen product on Amazon and a few customers have reviewed your product; you want to know how well your product will sell based on the reviews you have to know if you should be marketing the product differently.

c) Goal

The aim of this project is to develop a model to predict how well a given product will sell within its category given the previous reviews of the product.

d) Data

The data for this project was obtained from a study completed at UCSD and can be found in the following location: https://deepyeti.ucsd.edu/jianmo/amazon/index.html

The data sets used in this study are the reviews and product metadata for the Home_and_Kitchen products. The data set consists of 6.5 million reviews for over a million products.

Citation:

Justifying recommendations using distantly-labeled reviews and fined-grained aspects. Jianmo Ni, Jiacheng Li, Julian McAuley. Empirical Methods in Natural Language Processing (EMNLP), 2019.

2. Data Cleaning and Wrangling

a) Importing and merging data sets

The data sets are stored as json.gz files and require a custom function to parse the data and read it in as a pandas data frame. We begin by loading the review data which contains 6,898,955 unique product reviews. A data set this large would be cumbersome and slow to work with, so in the interest of time we randomly sampled 1.5 million reviews from the nearly 7 million. Next, we loaded the product metadata and join the two datasets on the “asin” product code.

b) Extracting ranking from main category

The product category column indicates the various categories the products belong to, and the column “rank” contains a list of the product sales ranking within each category for example: "['>#9,714 in Kitchen & Dining (See Top 100 in Kitchen & Dining)', '>#29 in Kitchen & Dining > Bakeware > Baking Tools & Accessories > Baking Cups', '>#3,224 in Kitchen & Dining > Kitchen Utensils & Gadgets']". However, all these products come from the category “Home & Kitchen” so the sales ranking within that category is not useful, so we need to extract the ranking within the next category; in the previous example we want the sales ranking within the category “Kitchen & Dining > Bakeware > Baking Tools & Accessories > Baking Cups”. This was accomplished using several lambda functions in succession to clean up the text and extract the ranking (in the previous example the ranking would be 29).

c) Identifying similar products with different names

Anyone who has browsed products on Amazon knows that the same product can appear under several different names on Amazon and this data set is no different. Additionally, products with the same name may have different product IDs, but we want these products to be uniform for the analysis. To ensure that products appear consistent in the data set, we developed a custom fuzzy logic algorithm using the fuzzy wuzzy package for python. For each product in the data set the algorithm searches for products that have a set ratio greater than 85% (the ratio of similar words in the two titles) **and** a sort ratio greater than 85% (the order in which the words appear in each title); if these two criteria are met, the algorithm treats the products as the same and converts the title and product ID so they match. The algorithm then returns a data frame with the matched product information and a list of the matches; the following is the first element of the list of possible matches:

['Hamilton Beach 31103A Countertop Oven with Convection and Rotisserie',

'Hamilton Beach Countertop Oven with Convection and Rotisserie', 100, 95]

The first two elements of this list are the product names, and the next two elements are the set ratio and sort ratio respectively – it seems these are likely the same product with different names.

d) Handling missing data

There were several thousand products that were missing category, title and brand. Because we cannot know what the true value for any of these, imputation isn't an option, so we needed to drop these reviews. In the end we were left with 1.2 million reviews with complete metadata.

3. Exploratory Data Analysis and Feature Engineering

a) Price and ranking outliers

As with all data sets, there appear to be some outliers in our original data. These outliers are clear when you review the histograms and box plots for price and ranking.

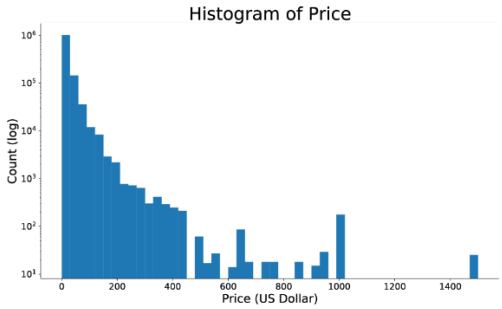


Figure 1a: Histogram of Price.

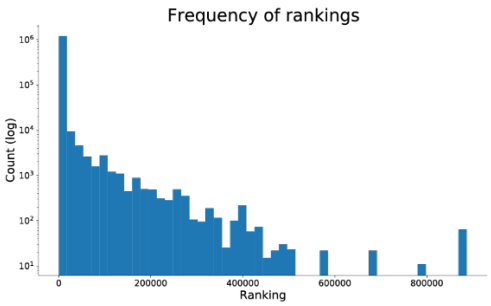


Figure 1b: Histogram of Ranking.

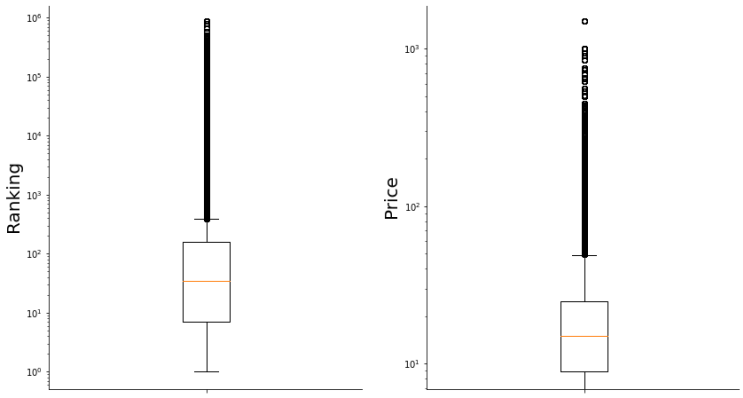


Figure 2: Box plots of ranking and price.

We used 4*IQR (inter-quartile range) as the boundary for outliers.

b) Binning categories and brands, generating the review sentiment, and review length

We planned to use PCA to reduce the dimensionality of the categorical data, however this presents a problem if brands or product categories do not exist in the training data but do exist in the testing data. To deal with this issue, categories that appear less than 500 times in the data are re-categorized as “miscellaneous”. Similarly, brands that appear less than 300 times are labeled as “miscellaneous”.

The next step was to extract the sentiment of the review and the review summary. This was accomplished using the Sentiment Intensity Analyzer class from the nltk vader package. The sentiment of a review and review summary was calculated as the average of the compound polarity score for each sentence. The compound polarity score is normalized to the interval from -1 to 1, with a negative score being a negative sentiment, zero being neutral and a positive score representing positive sentiment. The compound polarity score allowed us a mechanism for incorporating the review text into the analysis. Of course, there are instances for which the reviewer did not write a full review or summary, but simply rated the product; for these reviews we assigned a polarity score of zero (neutral). Finally, we were interested in exploring the possibility that review length could affect sales ranking, so we generated a feature to capture review length by tokenizing the review and dropping punctuation.

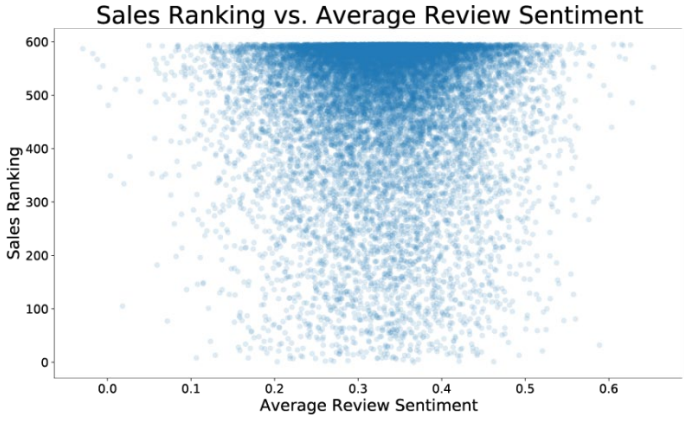


Figure 3: Sales ranking vs. Average Review Sentiment

We can clearly see that there is a non-linear relationship between sales ranking and average review sentiment.

c) Partitioning the data, unsupervised clustering, creating dummy variables, and principle component analysis.

At this point in the project we were ready to start generating some models to engineer features, so it was time to partition the data into training and testing data sets. The data was split using train_test_split with a 25% test size. As a means of exploring the possibility that there may be hidden patterns in the data, we wanted to employ an unsupervised clustering algorithm, in this case we used KMeans. The number of clusters was determined using the "elbow method".

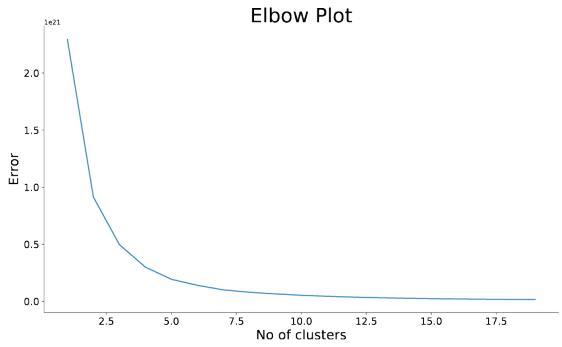


Figure 3: Elbow Plot: Error vs. Number of Clusters.

We decided seven clusters would be sufficient to capture the hidden relationships. The clusters were then used to create several interaction terms with the other continuous variables.

Next, we turned our attention to the categorical features. Adding the dummy variables for the category and brand columns created 4,719 new columns to the data set. While there are plenty of observations to handle this volume of features, we desired to reduce the dimensionality of the dataset, so we performed principle components analysis on the categorical variables. The PCA was able to reduce the number of columns for the categorical features own to 6.

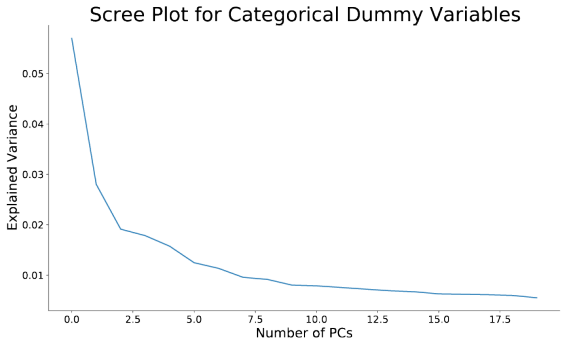


Figure 5: Scree Plot for Principle Components.

An identical pipeline was executed on the test data using the models and transformations trained on the training data.

4. Modeling

a) Linear models

Prior to creating the initial models, the predictor variables were standardized using StandardScaler. The first model we generated was a linear regression model using the linear regressor class from sklearn.linear_model. This initial model performed very poorly; in fact, all linear models performed poorly, with the best performing linear model only having an R-squared of 5.8% on the hold out set. The following figure shows the predictions vs. the true rankings for the best performing linear model.

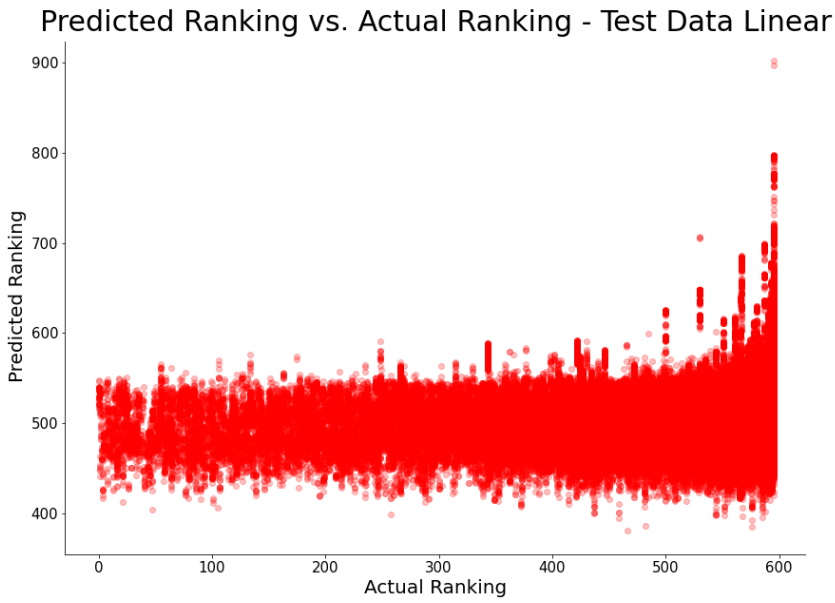


Figure 6: Best fit multiple linear regression predictions vs. actual rankings.

b) Random Forest and XGBoost Model

It appears there may not be a linear relationship between the feature space and the sales ranking. To investigate a non-linear relationship, we employed the XGBRegressor class from xgboost. Immediately we found a significant improvement over linear model with an R-squared on the hold set of 14.857%. We also tried a RandomForestRegressor from sklearn.ensamble, with similar R-squared values to the XGBRegressor. However, the XGBResgressor is significantly faster than the RandomForestRegressor thus we decided to move forward with the XGBoost regression model. To further improve the performance of the model, the hyperparameters (max_depth, gamma, learning_rate, n_estimators and subsample) were tuned using a Bayesian optimization algorithm with 5-fold cross-validation. This is a very slow process, but the optimal parameters resulted in significant

improvement of the model. The final model has an R-squared of 92.745% on the holdout test data.

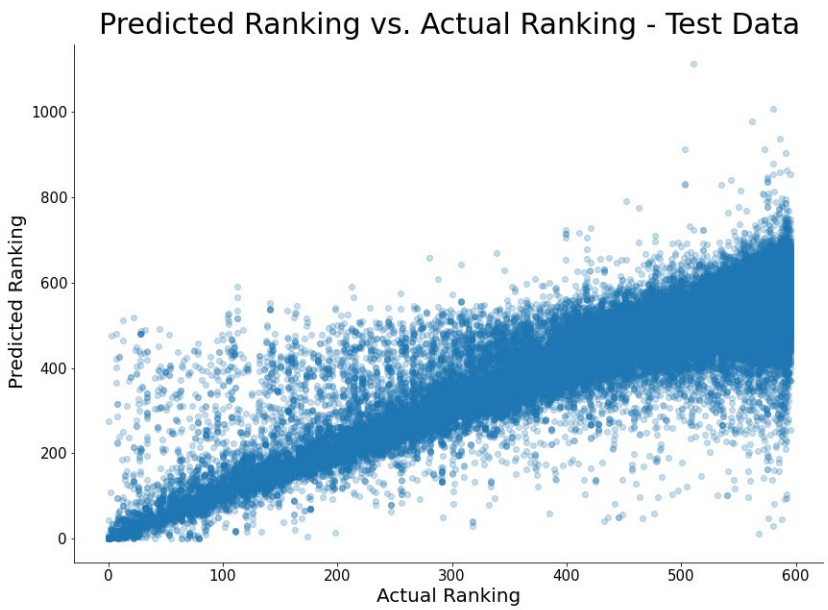
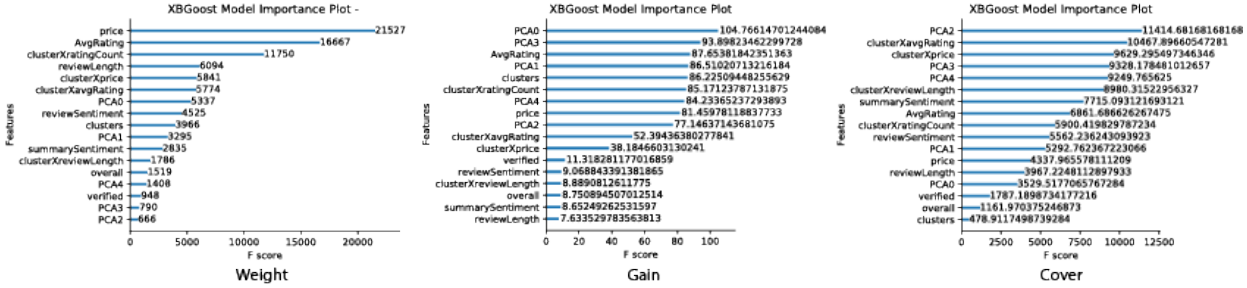


Figure 7: The XGBoost Regression Predictions vs. Actual Ranking.

5. Conclusion and future directions

The results of this model suggest that there is a strong relationship between the ranking, review sentiment, review length, summary, price, brand, and category. However, that relationship is highly non-linear. We need to determine the feature importance for the model to draw any conclusions about how our features affect the sales ranking. The feature importance function native to XGBoost is unstable in the sense that each of the three importance types (gain, weight, and cover) resulted in different rankings of the importance of the features.



This instability is not uncommon with models of this size, fortunately the Shap package for Python provides a concrete means of determining the feature importance for confident model

interpretation. The Shap package is based on a calculation of the Shapely value to determine how much each feature contributes to the error reduction in the model. When we look at the summary for the Shap score, it is clear that the cluster feature is very important, as is PCA0. Unfortunately, these features do not lend themselves to interpretation, therefore we need to focus more on the price and average rating features.

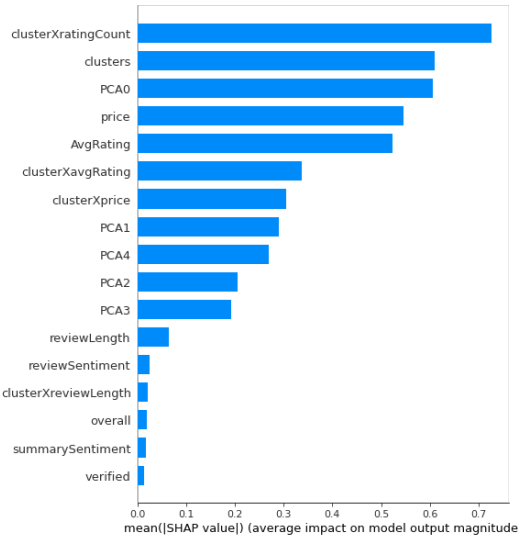


Figure 9: Shap values for the XGBoost model.

We can dig a little deeper into these relationships if we look at the Shap score for each of the predictions in the training set.

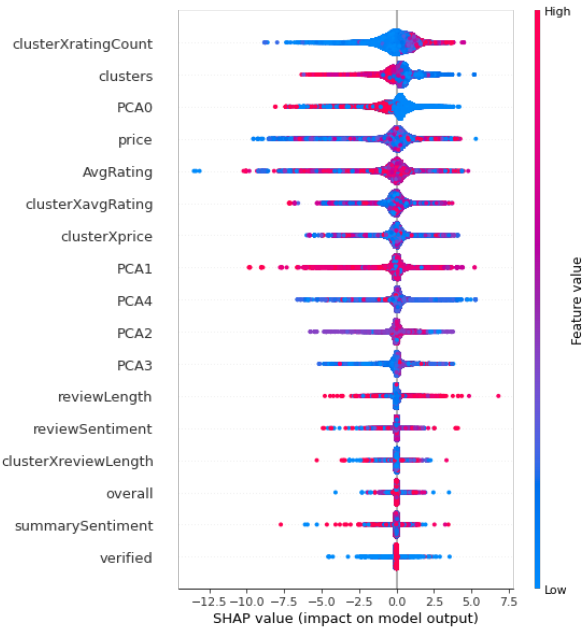


Figure 10: Shap scores and feature values for each prediction in the training set.

The figure above seems to suggest that there may be a negative correlation with price and Shap value, meaning that lower prices are more meaningful in the model. While this does not indicate a causal relationship, it is intuitive to think that lower prices drive sales ranking. With these results, we would have the following recommendation for the developer of a new product:

1. Make sure your product is priced competitively.
2. Given that average star rating is far more important than the average review sentiment, one should not overly concern themselves with the body of the review.
3. The number of reviews seems to be rather important, so it would behoove the sellers to encourage buyers to rate the product.

While the XGBoost regression model predicts the ranking well for the home and kitchen reviews, it would be interesting to expand this analysis to other types of products. In fact, it would be particularly interesting to merge samples from Apparel, Outdoors, Tools, Watches, Wireless, and Home and Kitchen into one large dataset and determine how well the model predicts among the larger subset of data.