# Efficient Modeling and Prediction of E-commerce Recommendation System: A Case Study on Amazon-Product Review Dataset

Yulong Dong, Daniel Sitompul, Khanh Thai

December 8, 2023 Link to project video:
`https://www.youtube.com/watch?v=JnomNXqTu_8&ab_channel=KhanhThai`

**Abstract**

With the rapid growth of e-commerce, products' reviews become increasingly important in affecting e-commerce businesses' sales and sustainability, as customers' purchasing decisions are influenced by others' recommendations and complaints about the products. Our research conducts an analysis into the Amazon beauty products reviews dataset from 2004-2018 to build a star-rating predictor model, based on different features about the product on Amazon. First, we conduct feature engineering to identify a good set of features associated to products' overall ratings on a binary classification of positive (5 star rating), and unsatisfactory (1 to 4 star rating). Secondly, we test different machine-learning algorithms (i.e. logistic regression, random forest, and stacking models) to build a rating-based satisfaction predictor of products based on text-based features of review and other relevant features on Amazon. To train and test the model with a good reference of performance, we randomly reserve 70% as training data to train the algorithmic performance and then use 30% of data as testing data to confirm the performance of the model. We tested the models using various metrics including ROC curve, AUC score and accuracy metrics. Based on the ROC curve, we found that predicting binary outcomes proves to be more manageable (compared to multiple categories), especially in cases with imbalanced datasets. We used logistic regression and random forest as potential models, and compare models' performance and find the model with the best performance. We also employ the K-fold cross validation technique to study the potential room of model complexity reduction and model selection, whilst respecting the tradeoff between bias and variance. Our findings can draw insights into characteristics of products' reviews, and how products' online presence may affect the businesses' reputation and products' sales. We also built a recommendation system using the classic collaborative filtering algorithm. The numerical example indicates that the recommendation system can conduct reasonable product recommendations.

## 1 Introduction

In the increasingly digitized world, eCommerce is taking precedence over traditional brick-and-mortar business models [5]. With the growing importance of businesses' online image, the role of customer reviews has become increasingly important and has been shown to improve customers' perception of products and attract new shoppers to make guided decisions about their purchasing decisions, acting as e-word of mouth (e-WOM) [4, 2]. Thus, being able to draw insights into potential predictive potential of historical data about products' and consumers' behaviors can garner great importance to e-commerce businesses' strategies in signaling needs for improvement/changes to the products, and/or directive marketing of selective products that have high probabilities to garner interest and satisfy customers' needs, based on customers' data and historical review patterns (i.e., recommendation systems and the positive effect feedback mechanism to garner customers' trust).

In this paper, we'll be exploring the following research question: "Can we predict customers' ratings for products on a binary classification system: positive and negative, based on the reviewer-product engagement on Amazon, such as text-based features of reviews (e.g., sentiment analysis), product-specific features (e.g., products' style) and datetime features (e.g., time of review submission)". This research aims to investigate factors that influence products' ratings, namely, important features yielding good/bad ratings based on historical data. The identification of important features contributing to positive ratings can provide insights to the feature engineering of building efficient recommendation systems and also benefits the reduction of model complexity by pruning unwanted redundancies. These findings would eventually benefit the design of a well performed recommendation system on large-scale real business.

In some literature, the rating prediction is formulated as either binary classification problems or multi-class ordinal classification problems. The choice of the formalism depends on not only the business purpose but also the data availability. In this work, we investigate the rating prediction in two manners: binary classification (5-star rating or not) and a full 5-class ordinal classification. Upon initial data exploration, we found that the dataset is extremely imbalanced. Most overall ratings are five while those less than five become increasingly small. This phenomena aligns with the market behavior as higher default rating is more common for frequent reviewers and a large amount of products are in good quality. After playing with standard logistic regression and its multi-class ordinal counterpart, we conclude that the imbalanced dataset significantly undermines the quality and performance of the modeling and testing. This observation motivates us to focus on a binary classification problem in the remaining research.

Another challenge for the modeling is that the product-specific features are turned out to be extremely sparse. For example, the volume and dimension of many products miss proper specification of units. Such worse calibration of features isolates them from modeling for scientific faithfulness. At the same time, rich reviews provide numerous text-based features to build the classifier. Past literature have been concentrated on exploring different characteristics related to review text: such as review length, valence of review and sentiment analysis, and have suggested an association between review text features and the products' overall ratings [1, 2]. However, given the size of the dataset, we are not able to unlimitedly explore the rich structure of reviews as the extremely high-dimensional text-based feature space could yield an overfitted model. To remedy this problem, we aggregate review texts as a scalar sentimental score feature but vectorize the review summary using an important technique in natual language processing (NLP) called term frequency inverse document frequency vectorization (TF-IDF). Though this treatment is a consequence of the relatively short length of review summary, the resulting feature space is still high-dimensional, namely, with a dimension greater than one thousand. This leaves potential rooms for the reduction of model complexity. We investigate it by stacking a PCA-based dimensionality reduction and a random forest classifier. A study with K-fold cross validation suggests that keeping only 15% of the feature space is sufficient to have an accurate model.

As a summary, to effectively address the core research question, the following sub-questions will be used to guide our analyses:

1. *Can we select features from customers' reviews helpful to predict products' overall rating?*

   In the pursuit of constructing an effective rating prediction model, our research takes on a comprehensive approach, recognizing that the foundation of efficiency lies not only in selecting features strongly correlated with the response variable but also to investigate features' interactions with each other (collinearity) to strategically isolate irrelevant features. The aim is twofold: to highlight the significance of influential features and avoid overfitting to reduce model complexity and statistical variance. We aim to achieve these goals by investigating into the questions below: a/ How do different products' features interact with each other, and influence product rating? Which specific features exhibit the strongest correlation and are most relevant to predicting customers' product ratings? b/ How can the model address issues of sparsity in the dataset, especially with products that have few reviews? c/ How can textual data from customer reviews be effectively transformed to meaningful features for the model? d/ Are there non-linear or linear interactions between features in the data, and the product ratings?

2. *What model may give the best accuracy to the customer rating prediction model?*

   A good model will give a good prediction with the least error in both training and testing set (unseen data), as well as providing the most explainable meaning to real-world context of product ratings. Our goal is to shortlist and test out a few promising models and methodologies by investigation into the following sub-question: a/ How do different types of models (linear and non-linear) perform under diverse quality metrics, for both trained and unseen data?

3. *Is there any potential room for improving the model to accelerate training and inference on large-scale real business applications?*

   By testing with different techniques to control model complexity, are there certain techniques to prevent overfitting the model (e.g., regularization, PCA) perform better in certain model designs than others (e.g., linear regression, random forest, etc.)? And which methodology and model can achieve the optimal sweet-spot in balanced between complexity and variance?

## 2 Description of data

**Overview of data sampling and variable types** Our dataset is composed of Amazon Review Data from 2004-2018, with information about customers' reviews of products. Data was collected from Amazon's platform by UCSD researchers.

We are focusing on a subset of data with reviews for beauty products in the "All_Beauty_5.json" dataset, which has 5.2k reviews. This subset of data has 12 features (e.g., ratings, review text, time, votes, etc.). Each row of the data (i.e., granularity) represents each review submitted by a customer; featuring information into user verification, review time, reviewerID, the product style (i.e., dimensions of image, color, product type, etc.), reviewer name, review text, summary of comment, unixReview time, number of vote the comment gets, and finally whether the review included an image or not. There are several type of data that we are currently exploring in this subset of data, which is summarized in Table 1.

| No | Type | Column | Description | Variable Type |
|---|---|---|---|---|
| 1. | String | reviewerID | ID of each reviewer | Qualitative |
| | | asin | ID of each product bought | Qualitative |
| | | reviewerName | Name of reviewer | Qualitative |
| | | reviewText | Written review of product | Qualitative |
| | | summary | Summary of review on product | Qualitative |
| 2. | Integers | Overall | Rating | Quantitative |
| | | vote | Number of people voting the review | Quantitative |
| 3. | Boolean | Verified | Verified or non verified reviewer | Quantitative |
| 4. | Time/Date | reviewTime | Date of the review time | Quantitative |
| | | unixReviewTime | Unix format of review time | Quantitative |
| 5. | Dictionary | Style | Mixed of Info about the product (Size, color, flavor, etc) | Mixed of Quantitative and Qualitative |

Table 1: Original dataframe and its description.

**Initial Exploratory Data Analysis** Upon retrieving the data from UCSD's open-source platform, we investigated into the distribution of each features, distribution of missing entries, the relationship between features, and whether normalization of different features may be needed (i.e., to efficiently do comparisons across different features that may have different measure of units) or transformations of existing values may be needed for further analyses.

The initial exploratory test shows the distribution of overall rating across all users based on the type of users and time. We find that most of the ratings given by the users are mostly in the 5 stars, with the average rating by verified users (4.8/5) is slightly higher than unverified users (4.58/5) (Fig. 1). We then explore the reviews given by the verified and non verified users. When comparing relationship between verified status and how long (i.e., how many characters), on average, these users' reviews are, it appears that unverified users tend to leave much longer review text (on average, 467.9 characters) compared to verified users (140.947 characters) (Fig. 2). Although there are distinguishable differences in review-writing behaviors across verified vs unverified users, looking into the distribution of product-ratings left by different types of users reveal relatively similar distribution (i.e., left-skewed, highly biased and mode at 5/5 rating), we decided to not further explore this feature as a potential feeder to our model of predicting overall rating.

We explore the temporal dynamics of customer reviews, investigating how factors influencing product ratings have changed over time. This temporal analysis adds a crucial layer to our research, unraveling the intricate dynamics of customer reviews across different periods and shedding light on their impact in constructing an efficient and explainable prediction model. We will focus on model selection and hyperparameter tuning to provide accurate predictions. Through this comprehensive exploration, we aim to bridge the gap between theoretical understanding and practical implementation, culminating in a robust and effective rating predictor for customer reviews. We also understand the overall rating distribution based on year, we visualize it with boxplot as shown in Figure 10. While we find that (figure 10) most ratings are 5 (¿75%), figure 3 also shows that the year variable correlation is not obvious (i.e., not much changes in rating distribution across the years) but we can see differentiation in rating distribution with month data, thus, we decided to proceed further investigating into the months data to get better prediction for our model (Fig. 3).

We then investigated each variables' missing entries and the density of missing entries (Fig. 4. & Fig. 5.). From this initial review, it shows that the variables "style", "vote", "image" have the largest density of missing entries (12.2%, 89.5%, 98.1% respectively). The reviewText and summary also have missing entries, however, the density of missing entries seems relatively low ( 0.09%) (Fig. 6.). Secondly, we further investigated the product-specific features, and saw that these features are extremely sparse, with too many null values and the units are not consistent across features and are difficult to normalize to the same scale (Fig. 6.). Combining with the observation that the style-specific feature of the product is missing (see Fig. 8), we pin down the missing data and isolate them in further modeling. Furthermore, we showcase extensive work in visualizing the data distribution in the later section when presenting results to visualize our awareness and treatment to outliers in the dataset. For text-based entries (e.g., summary text, review text), these variables show promising distribution (i.e., low NA results) and have been the core variable used in many previous research into predicting products' rating; thus, we decided to further transform and explore helpful features of these text-based variables to integrate into our model as inputs.

Additionally, We also looked into the distribution of the aggregated overall star rating (from 1 to 5), and saw that the aggregated data is highly imbalanced with the majority of stars being 5/5 (Fig. 7.), which cautions that using this dataset for training may lead to a bias towards positive ratings (i.e., 8) but may perform less well in accurately predicting low-ratings. With the highly-skewed left distribution and high bias for 5/5, we decided to categorize distribution 1-4 as a single "negative rating" in order to create greater balance between positive (5/5) and negative, making our research question a focus on binary classification to introduce a bit more balanced to the dataset.

After deciding on promising features, we decided to remove product-specific features from the dataset, drop any observations without reviews, normalize vote *and change null value to 0), and set the products' overall rating as well as number of votes to be our prediction targets.

**Data Cleaning and Transformation** We further investigated the potential transformation of the "unixReviewTime" to a more easily readable format, and decided to proceed with day of the week, month, year (Fig. 6.). We'd also tried to investigate making a label for "part of the day" that the review was submitted (e.g., late night, early morning, morning, etc.), but saw that the system does not record the hour of review; thus, decided to not move forward with that transformed feature (Fig. 6.)and to only continue to get day-of-week, month and year. Looking at the distribution of the transformed data (Fig. 3, we found that the number of reviews and ratings vary over the year and within months of the year. Thus, by transforming the unixReviewTime information into day-of-week and month, this allows us to capture insights into temporal data and account for seasonal trends (E.g., weekdays vs non-weekdays, month in the year) in when the review is recorded and its impact on the products' overall rating.

With the text-based variables (e.g., summary, reviewText), we first conducted some further cleaning such as transforming everything to lower-based characters and stemming such as cleaning out non-alphabetic characters and removing any words that are in the list of stopwords (i.e., common words that usually don't contribute to the meaning of the text). We also created a function to define a sentiment score for each cleaned string of text returning a compounded score for each text review (Fig. 10.). We also attempted at vectorizing text data into numerical form via TF-IDF vectorization, abbreviating term frequency inverse document frequency, to capture the popularity of the words but penalizes the commonality of cross documents (Fig. 11.). We'd decided to use TF-IDF in order to appropriately use inverse document to do penalization with weights for words in commonality that has little meaning (e.g., "is" may have very high frequency across reviews, but the word "is" itself is meaningless). From review of the dimensions of the vectorized reviewText and summary, we found that vectorized reviewText has too many features (¿5,000, whilst our sample size was only 5.4k) that may pose complexities and difficulties in usage to as meaningful input for our model. Thus, the full reviewText is aggregated as a scalar sentimental score to reduce the model complexity and avoid overfitting.

## 3 Methodology

After shortlisting and transforming promising variables (i.e., unixReviewTime, summaries text and reviewText), we conducted further feature engineering into promising features within those variables and investigated into its dimensions. First, we engineered a feature called "vec_summary", which detects the appearance of certain words in summary text and saw that it has 1091 features that can capture the qualitative aspects of texts. Next, we utilized datetime_feats (datetime transformation of unixReviewTime), to create a total of 19 booleans (i.e., 7 days, 12 months) features that show the days and months of the reviews in order to provide insight into temporal data and reveal seasonal trends in review. Subsequently, we created a feature capturing sentiment score of summary text by using the VADER lexicon. Similarly, another feature was crafted to assess the sentiment score from "reviewText", also using VADER. The selected features are summarized in Table 2.

| No | Column | Number of feature | Description | Variable Type |
|---|---|---|---|---|
| 1. | vec_summary | 1091 | TF-IDF of summary text (frequency and relative importance of words) | Quantitative |
| 2. | datetime_feats | 7 (days) + 12 (months) | Transformed from the unixReviewTime | Boolean (0/1) |
| 3. | sentiment score summary | 1 | Score based on Vader | Quantitative |
| 4. | sentiment score reviewText | 1 | Score based on Vader | Quantitative |

Table 2: Selected faetures and their descriptions.

Our current matrix, thus, consists of 1112 distinct features (figure 25). To address potential overfitting and enhance model performance, we applied regularization techniques to control the model complexity. We employed both L1 and L2 regularization methods to optimize feature selection. We investigated how the model's RMSE behaves (for different types of models) as regularization strength increases to identify whether regularization is an appropriate method to control overfitting for our model, and what the optimal penalty parameter should be. We also tried other methods to control overfitting and reduce model complexity by integrating PCA with non-linear classifiers and testing to see if accuracy is preserved when feature dimensions are reduced. in order to see if This step was crucial in refining our model's structure and improving its predictive accuracy on both trained and unseen data.

After cleaning the dataset and exploring potential features, we proceeded to design and explore potential rating-predictor models. We split the dataset to randomly reserve 30% of data as testing data and the other 70% as training data in order to gauge the algorithmic performance (Fig. 12). We prepared feature matrices (E.g., X_train and X_test) for training and testing data, as well as target matrices Y_train and Y_test.

After making the training and test dataset, we proceeded to investigate into different potential models that can be used as our rating-predictor and compare the model performance against quality checkpoint (e.g., ROC, confusion matrix, RMSE, etc) to investigate the optimal model to be used for our predictor. As a classification problem, the simplest method is logistic regression and its multi-class counterpart. Logistic regression enjoys the simple structure of the model as a generalized linear model. Furthermore, it conveys clearer statistical meaning of each feature compared with how a complex nonlinear classifier does. However, the simple structure of the model puts the risk of intrinsic modeling bias to the feature engineering based on extensive domain knowledge of the researchers. Usually, nonlinear models like random forest and neural networks could outperform logistic regression by including more hidden machine-learning features. These features are usually referred to as the higher order interactions in the machine learning community. To see the importance of each feature, for each model designed, we visualize the top 30 features that contribute to the classifiers' predicted outcomes, and their weights in the classifier to identify the most influential factors contributing to the model's decision-making process. This provides insights into which variables held the most predictive power, and also motivates us to study the reduction of model complexity by stacking a dimensionality reduction subroutine before applying the classifier. Meanwhile, we also performe a K-Means clustering on the vectorized review summary. By presenting the top 10 most weighted word in each cluster, we also observe great similarity among clusters. This also motivates the potential room of model complexity reduction.

To reduce the dimension of the feature space and accelerate the model training and inference, we deploy a stacking model by passing the raw data to a Principal Component Analysis (PCA) subroutine for dimensionality reduction and further process the compressed data using a random forest classifier. We test the performance of the stacking model with multiple choices of hyperparameter, i.e., hidden feature space dimension outputted by PCA, using a 10-fold cross validation. The accuracy curve indicates that a relatively small hidden feature dimension is sufficient to get an accurate model. To be precise, we could compress the feature dimension to 15% using PCA to preserve the performance but this dimensionality reduction could drastically accelerate the model training and inference.

Finally, we build a recommendation system based on the historical reviewer-product engagement data. The recommendation system is based on a classic algorithm called collaborative filtering. Intuitively, it provides the recommendation by exploring similar products reviewed by other users in the sense of correlation. The underlying formalism is based on performing singular value decomposition (SVD) on a matrix where each entry is the rating of a product given by a reviewer. This matrix can be built as a pivot table derived from the original dataset. By testing our recommendation on the first product and manually search the recommended products, we observe that the recommended products are highly correlated with the given product in not only correlation score but also human knowledge.

## 4 Summary of results

As mentioned in the data description section, based on initial EDA, we've decided to not move forward with the sparse product-specific features (i.e., high density of nulls) for our model, and instead focus more on the rich textual data from the reviews and summaries as well as review-time. We further cleaned the data by dropping rows without review nor summary and normalizing the vote variable, changing null value of vote to "0" (Fig. 5.), and transformed the text-based variables (Fig. 10 & Fig. 11) as well as review-time submission (Fig. 6) as described under "Description of Data". We identified three promising categories of features to act as input for our rating-predictor: datetime, TF-IDF representation of summaries (i.e., vectorized summaries text), sentiment scores of summaries and reviewText as aggregated features.

After making the train-test data split, we created a simple logistic regression for predicting five-star rating (Fig. 13) and plotted a category distribution on training and testing dataset to investigate the distribution of each set. The plot shows that in both training and testing sets, the category “1” (i.e., reviews receiving 5/5 star) is far more frequent than the category labeled as “0” (i.e., review not receiving 5/5 star) which is another caution flag that there exists imbalance in our dataset. However, Fig. 13 indicates a consistent distribution of both categories in training and testing sets, which suggests that the split for training and testing sets was successful in maintaining overall distribution of the variable. The imbalance in classes, as mentioned earlier, may cause the trained rating-predictor model to have a bias towards one category rather than others, which will be further investigated in our summary of results.

As discussed in the early section, though logistic regression enjoys a simple model structure and good statistical interpretability, its performance usually relies on sophisticated feature engineering and manual creation of interactions beyond linearity. This issue could be remedied by employing nonlinear classifiers like random forest. Random forest is an ensemble method based on decision trees which reduces the model variance for better generalization. Moreover, as a nonlinear classifier, random forest is more robust against imbalanced dataset than logistic regression does. We did the numerical test for the justification. To measure the model performance, we compute the RMSE and Log Loss on both training and testing dataset. The former indicates the training error while the later stands for the testing error or generalization error. The choice of RMSE is simple and intuitive as the entry-wise error. However, Log Loss is better suited for the binary classification problem but less obvious in its meaning. Motivated in the early section, we also study the potential room of dimensionality reduction by stacking a PCA subroutine before applying random forest. The hyperparameter in the stacking model is thus the dimension of the hidden feature space outputted by PCA. We employ a 10-fold cross validation to plot the curve of model accuracy. The result indicates that the feature space can be drastically reduced to accelerate further classification but preserve a consistently high accuracy.

**Logistic Regression** Using logistic regression, we find that predicting binary outcomes is proven to be more manageable than classifying multiple categories. The intrinsic reason not only lies in the model complexity but also is due to the fact that the raw 5-calss rating dataset is highly imbalanced. By aggregating all 1- to 4-rating datapoints as a single category, the reformulated binary classification problem becomes less imbalanced and logistic regression could possibly deliver reasonable results. We graphed the ROC curve (Fig. 14) in order to investigate the true positive rate against the false positive rate, with the AUC valued at 0.88 suggesting that the model has a relatively good measure of separability (i.e., 88% chance model can distinguish between the positive class of 5-star reviews vs the other negative classes of reviews). Secondly, our confusion matrix (Fig. 15) that showcases the multi-class classifier (e.g., model always predicts 5 star rating of the time, model predicts 4 star rating 21% correctly of the time, etc.). From the confusion matrix, it was evident that the model performs better in predictions for extreme ratings (5/5 or 1/ 5), however, less well in 2, 3 and 4-star ratings. However, this is another signal that amplifies initial findings of potential bias in the EDA step and investigation into quality of training dataset, that our dataset is imbalance (high density of 5/5 rating). Hence, the model is biased towards predicting higher rating due to the imbalance. In order to circumvent the imbalance and improve the model’s performance on multiple-category prediction, we would take the downsampling trick and complex nonlinear model as our future work.

We visualized the top 30 important features (Fig. 16) and their weights in the logistic regression model, where positive weights indicate positive correlation to likelihood of the predicted outcome being 5/5 star, whereas negative weights indicate negative correlation to likelihood of product getting 5 star. The chart shows a balance of features in both contributing to and detracting from likelihood of product in receiving overall rating of 5/5, without any certain feature overwhelming dominates the predicted outcome. However, this chart also reveals potential reasons that bottlenecked the performance of the logistic regression model. We observe that the vectorized word ”star” exhibits a negative weight in the model. However, the appearance of ”star” should not largely be biased towards negative one as e.g. ”five star” also stands for positive meaning. This indicates that the logistic regression model may not fully capture the association between features and the response variable. As in the example of "star", the alignment with "five" or "one" should deliver completely opposite meanings. This is usually referred to as the interaction between features. As a generalized linear model, logistic regression does not naturally include interactions. Hence, this motivates our research by using nonlinear classifiers like random forest.

**Random Forest** As motivated early, we use random forest classifier to enhance the modeling of rating prediction, and assessed the diagnostic ability of the model in predicting five-star rating via the ROC curve (Fig. 17). The AUC value of 0.99 suggests the model performs fairly well in being able to distinguish between positive vs negative classification (i.e., 99% chance model can distinguish between positive class of 5-star reviews vs the other negative classes of reviews), which is better than simple logistic regression model. The ROC curve towards top left also indicates that the model performs fairly well in both sensitivity and specificity, and has high predictive performance.

Similar to the previous case, we also visualized the top 30 important features and their weights in the random forest (Fig. 18) classifier. Unlike the case of logistic regression, the weight of each feature is not signed due to the formalism of random forest. Hence, the statistical meaning of the weight is not as obvious as that in logistic regression. However, we can still see the reasonability of the weighting as the aggregated sentimental scores become most significant while the word "star" becomes less biased.

**Stacking Model and Dimensionality Reduction** As motivated before, we study the potential room of dimensionality reduction by stacking a PCA subroutine before applying random forest classifier. The hyperparameter is set to the dimension of the reduced hidden feature space outputted by PCA. We visualize the accuracy curve computed from a 10-fold cross validation versus the choice of hyperparameter. The result is depicted in Fig. 19 which suggests that compressing the feature space dimension to 15% of the raw one is sufficient to have a sufficiently high accuracy. However, such dimensionality reduction drastically accelerates the model training and inference.

**Clustering** We employ clustering method to understand the hardness of multi-class classification by studying the structural information of the dataset. We use a tSNE, t-distributed stochastic neighbor embedding, method to project the raw data from thousand dimension to a two dimensional subspace in a supervised manner according to their overall rating values. The result is depicted in Fig. 20. It reveals the lack in the structure that differentiates the specific rating from 1 to 4. After aggregating them into the same category, the lack in the structural information becomes weakened. Hence, we conclude that aggregating lower ratings into a single category is beneficial for modeling from such intuitive visualization in a two dimensional subspace.

We also employ a K-Means clustering algorithm to study the structure of he vectorized text data. To illustrate, we set the number of clusters to 10 and visualize the cluster label for each data point in Fig. 21. We see that a large portion of data points are assigned to the first cluster. This indicates that there might be some additional structure to be employed so that a lower dimensional feature subspace would preserve the original high dimensional features. Furthermore, by visualizing the top weighted words in each cluster in Fig. 22, we see great similarity among clusters. This observation also suggests the room of dimensionality reduction.

**Recommendation System** We build a recommendation system based on the historical review-product engagement data. We employ a method called collaborative filtering which intuitively recommends similar products by exploring the product-wise correlation computed from the dataset. The basic idea is described as follows. First, the engagement matrix is formulated as $R_{ij}$ where the $(i, j)$-entry is the overall rating of the product $i$ given by the reviewer $j$. This matrix can be computed as a pivot table, see Fig. 23. Then, we perform a SVD to the engagement matrix so that $R = V\Sigma W^\top$ where $V$ is a $n$-by-10 matrix and $n$ is the total number of products. The second number 10 stands for the dimension of the hidden features we would like to explore. Eventually, the correlation between product $p$ and product $q$ calculated as $C_{p_1,p_2} = \mathrm{Corr}(V_{p_1,:}, V_{p_2,:})$. To recommend products similar to a product $i$, we simply pop out products with highest correlation $C_{i,j}$. We visualize the result of such recommendation system in Fig. 24 where we recommend five similar products based on the similarity. We see that the recommended products make a lot of sense in human knowledge and preference.

## 5 Discussion

The research is set to investigate whether an appropriate model can be constructed to predict Amazon products' ratings to positive or negative (binary classification) based on products' features (E.g., shape, image, etc.) and historical textual reviews and date-time features. Our results confirm that text-based features, through sentiment analysis and TF-IDF vectorization, have the strongest correlation and predictive capabilities to products' ratings, followed by date-time features to detect seasonal and temporal variability of reviews. The predictive model performs particularly well when the mentioned features are incorporated to random-forest modeling (non-linear classification that accounts for interaction among features) with integration of PCA with non-linear classifiers to ensure appropriate model dimensions (i.e., avoid overfitting to training data) whilst still preserving the nonlinear interactions between features.

Additionally, from EDA we were able to reveal the unbalanceness of the review dataset towards extreme reviews (with particularly high bias towards 5/5 rating and therefore model has bias towards predicting higher rating). Given that the dataset comprehensively records Amazon review data overtime, we are less concerned with the sampling methodologies of how the dataset was composed and acquired by UCSD researchers, but believe that this unbalanceness may be reflective of a common practice and problem of individuals getting paid to leave fake reviews to boost the products' online image. This practice was noted to be particularly prevalent amongst beauty products [3]. To account for the unbalanceness of the review dataset and taking into consideration the result of the confusion matrix (i.e., model performs better in predicting extreme ratings and less well for those in the middle), as well as clustering which reveals that there is a lack of structure in differentiating rating from 1-4 and by aggregating lower-rating to single negative category (and 5/5 as positive category) the model performs better Fig. 21; thus, we believe that our decision to proceed with binary classification (as opposed to multi-category classification) proves to be more effective in balancing and managing the imbalanced nature of the dataset, allowing the models to be more explainable and enhance their performances. This may also explain why our RMSE is more minimized (model performs better) for binary outcomes rather than multiple category cases. One standard preprocessing to handle an imbalanced dataset is downsampling and recalibrating the classifier which is our future research. On the other hand, the imbalance could be handled by using a more complex nonlinear classifier which exhibits robustness against imbalance. As shown in our numerical result using random forest, the AUC score of random forest has been lifted to 0.99 which drastically improved that of logistic regression namely 0.88.

By testing multiple models of varying complexity in methodologies, we were able to use logistic regression as baseline (due to its simplicity and interpretability) to compare with more advanced modeling methodologies like random forest classifier, which showed superiority in handling high-dimensional data and exhibits greater robustness against the imbalance of the dataset.

This dataset could have potential non-response bias as customer reviews depend on the customers' decision to leave a review and rating for the products. However, not all customers who bought the products may leave a review. Thus, the data may be skewed and representative of only a subset of all Amazon customers (e.g., potential risk that only those who feel strongly about the product leave a review, etc.). Additionally, another potential bias is response bias, there also might be a chance that the reviewer does not talk about the product and instead makes unrelated comments. Additionally, this research is limited to Amazon's reviews, not yet taking into consideration the impact of how reviews of the same product on other sites (e.g., Youtube review, Google review, etc.) can impact customers' behaviors on Amazon's site. Another potential drawback in our visualization of feature weight is that they do not capture potential interactions or correlations between the features which is due to the formalism of logistic regression. Nonetheless, this is circumvented by employing more complex nonlinear classification methods like random forest.

From a bias-variance tradeoff standpoint, a successful machine learning model should simultaneously mitigate the training error and the testing error which requires a control of bias (underfitting) and variance (overfitting) at the same time. A rather simple model can be easy to train and generalize onto unseen dataset. However, its bias could be very large. On the other hand, a very complicated model with numerous parameters could reduce model bias but the sophisticated structure does not generalize well onto unseen dataset due to the high variance. To select a model with proper tradeoff between bias and variance, we practice the cross validation with a stacking model. From the numerical result, we see that a relatively less complex model by proper dimensionality reduction could achieve a high accuracy. However, due to the reduction in the model complexity, it is believed that the variance is also reduced. Moreover, as we employ a random forest classifier which is an ensemble method, the algorithm itself exhibits the reduction in the variance by ensemble technique like majority vote in random forest.

What surprised us was how regularization was not effective at enhancing the model's performance. For example, Fig. 26 demonstrates the impact of regularization on the performance of logistic regression models by investigating RMSE behaviors of both training and test set when L1 and L2 are applied. Contrary to our expectations, L1 regularization does not significantly improve the model's predictive accuracy and appears to increase the testing error, indicating that regularization and model complexity reduction may not be helpful in simple models like logistic regression. However, our clustering analyses suggest that there is room for model reduction.

After testing with integration of PCA with a non-linear classifier (e.g.,random forest), we see that the accuracy of the model is preserved even though the model is compressed to 15 percent (significant reduction in dimension) of the original feature space. This is because there are significant nonlinear interactions between the features; hence, simple selection via L1 penalty ends up excluding important features and their interaction. However, PCA is able to reduce dimensionality with a mixture of original features. By processing with nonlinear classifiers, the model is compressed while at the same time, the interaction is preserved.

## References

[1] Arwa SM AlQahtani. Product sentiment analysis for amazon reviews. *International Journal of Computer Science & Information Technology (IJCSIT) Vol*, 13, 2021.

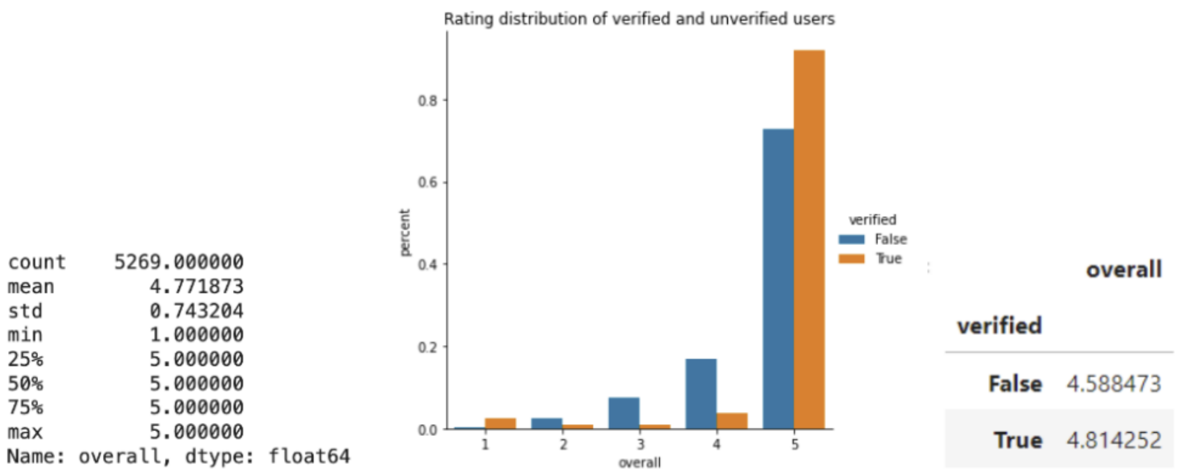
[2] Kapil Kaushik, Rajhans Mishra, Nripendra P Rana, and Yogesh K Dwivedi. Exploring reviews and review sequences on e-commerce platform: A study of helpful reviews on amazon. in. *Journal of retailing and Consumer Services*, 45:21–32, 2018.

[3] Edie Meade. Crossing a Minefield of Fake Beauty Product Reviews — medium.com. `https://medium.com/swlh/crossing-a-minefield-of-fake-beauty-product-reviews-34af00399668`. [Accessed 08-12-2023].

[4] Susan M Mudambi and David Schuff. Research note: What makes a helpful online review? a study of customer reviews on amazon. com. *MIS quarterly*, pages 185–200, 2010.

[5] František Pollák, Peter Markovič, Roman Vavrek, and Michal Konečný. Return to the new normal: Empirical analysis of changes in e-consumer behavior during the covid-19 pandemic. *Behavioral Sciences*, 12(3):85, 2022.

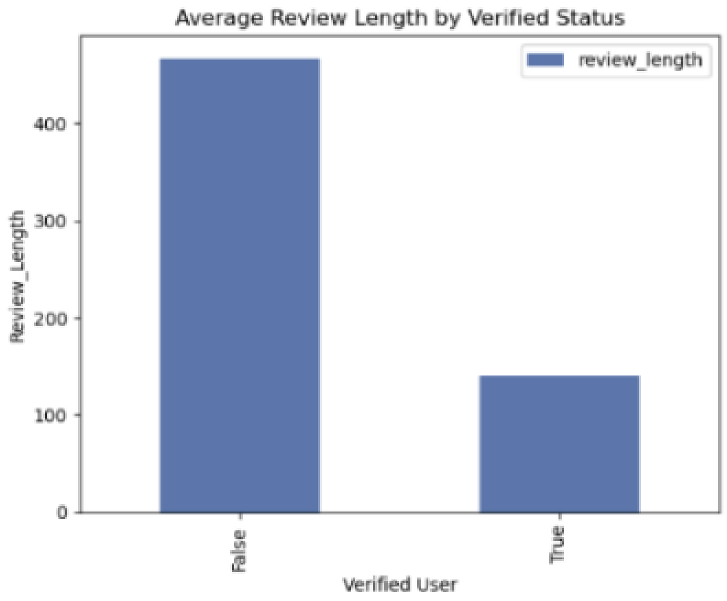
## List of Figures

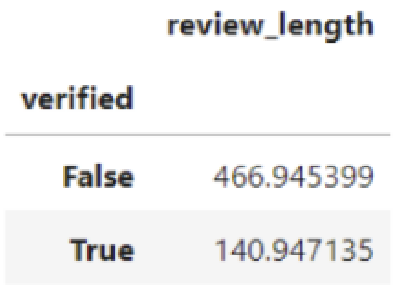


```
count    5269.000000
mean        4.771873
std         0.743204
min         1.000000
25%         5.000000
50%         5.000000
75%         5.000000
max         5.000000
Name: overall, dtype: float64
```

|  | overall |
|---|---|
| **verified** |  |
| **False** | 4.588473 |
| **True** | 4.814252 |

Figure 1: Distribution of rating based on the type users





| | review_length |
|---|---|
| **verified** | |
| **False** | 466.945399 |
| **True** | 140.947135 |

Figure 2: Understanding the length of the reviews based on the user

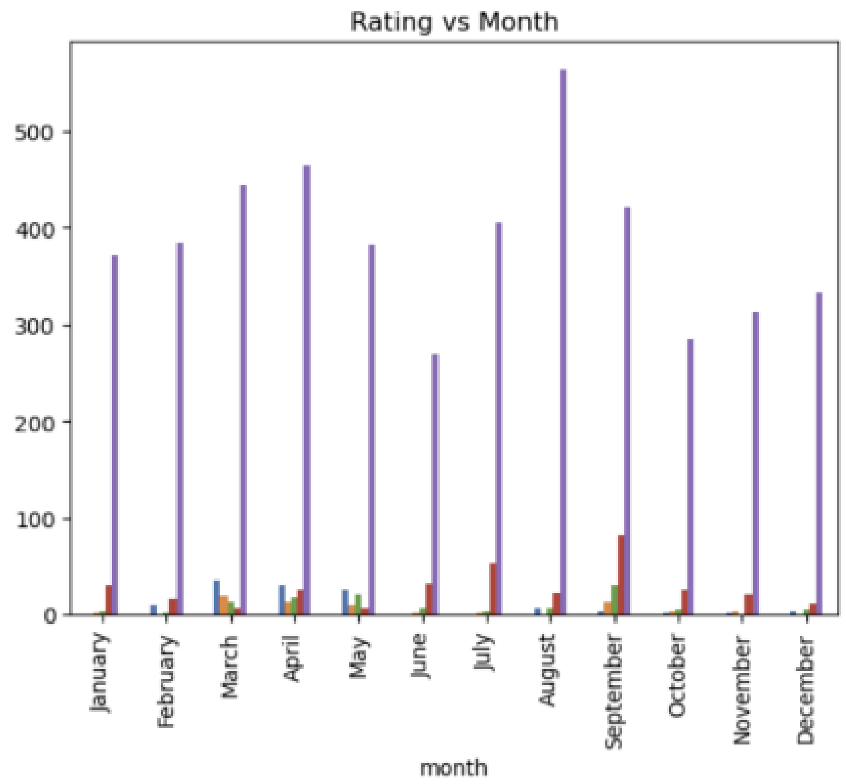
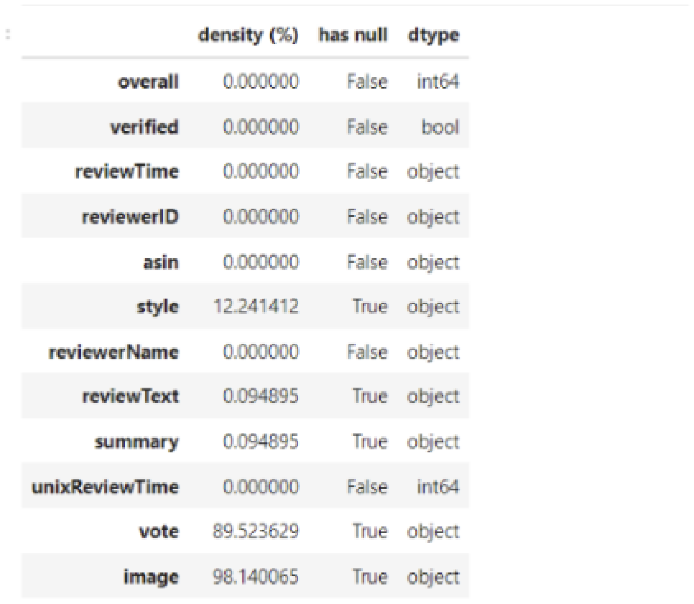
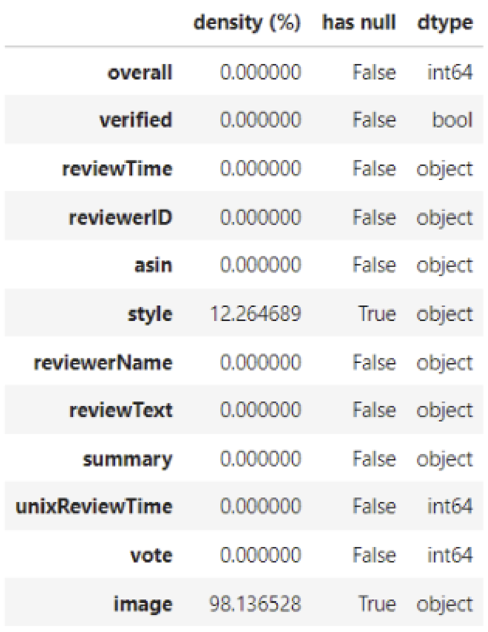


Figure 3: Distribution of Rating based by the Month



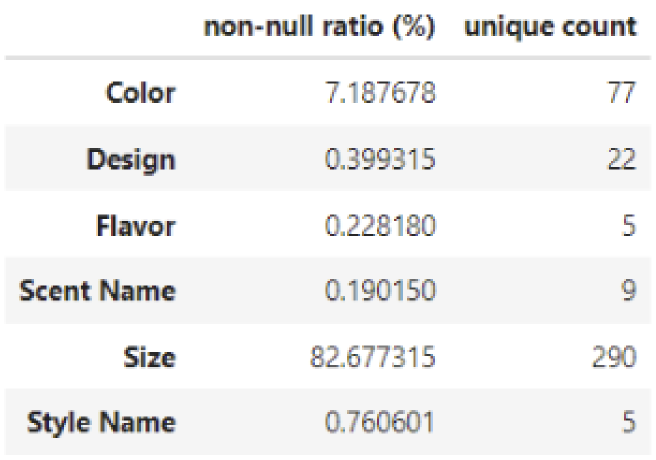
| | density (%) | has null | dtype |
|---|---|---|---|
| **overall** | 0.000000 | False | int64 |
| **verified** | 0.000000 | False | bool |
| **reviewTime** | 0.000000 | False | object |
| **reviewerID** | 0.000000 | False | object |
| **asin** | 0.000000 | False | object |
| **style** | 12.241412 | True | object |
| **reviewerName** | 0.000000 | False | object |
| **reviewText** | 0.094895 | True | object |
| **summary** | 0.094895 | True | object |
| **unixReviewTime** | 0.000000 | False | int64 |
| **vote** | 89.523629 | True | object |
| **image** | 98.140065 | True | object |

Figure 4: Initial missing entries' density for each variable



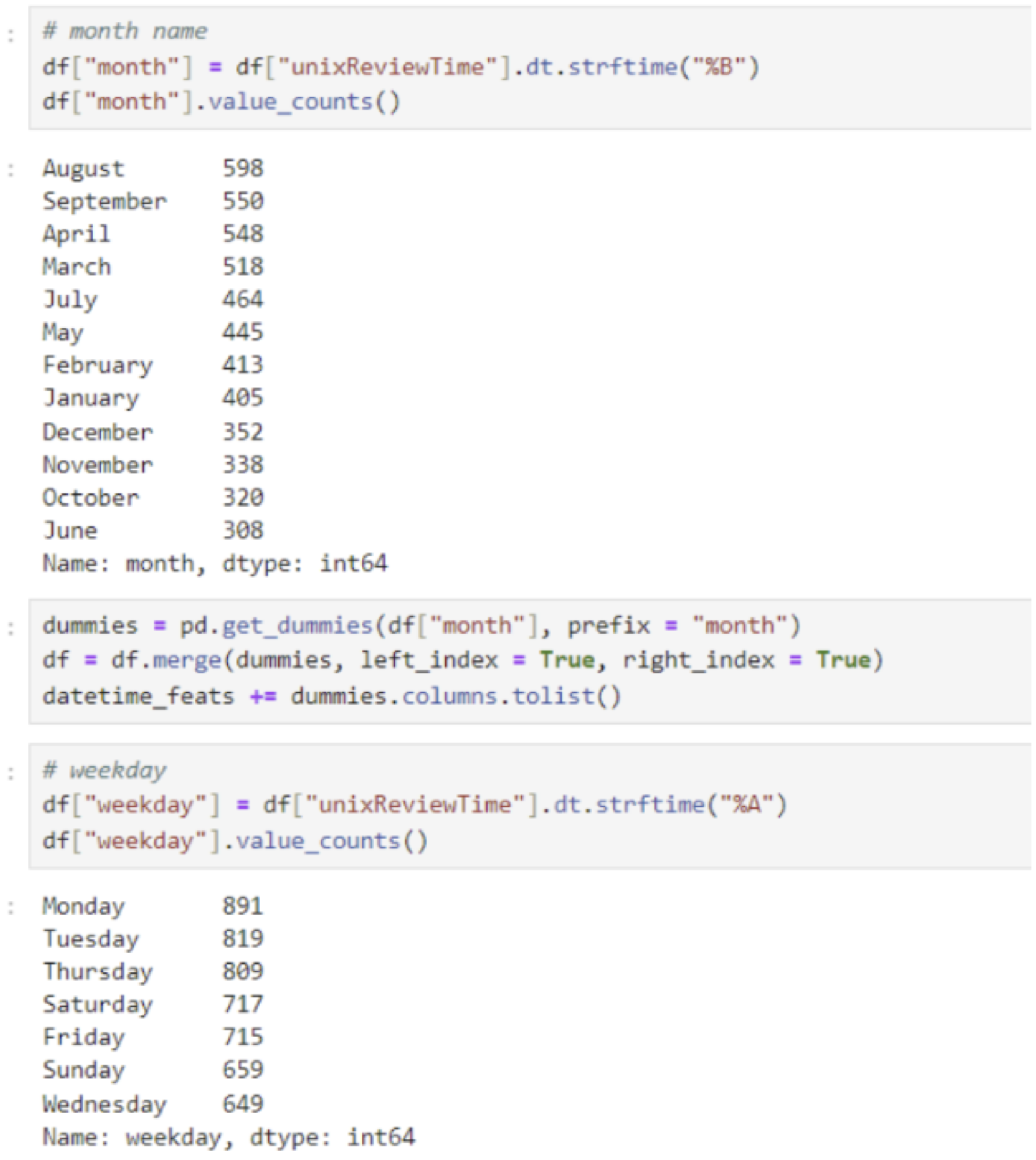
| | density (%) | has null | dtype |
|---|---|---|---|
| overall | 0.000000 | False | int64 |
| verified | 0.000000 | False | bool |
| reviewTime | 0.000000 | False | object |
| reviewerID | 0.000000 | False | object |
| asin | 0.000000 | False | object |
| style | 12.264689 | True | object |
| reviewerName | 0.000000 | False | object |
| reviewText | 0.000000 | False | object |
| summary | 0.000000 | False | object |
| unixReviewTime | 0.000000 | False | int64 |
| vote | 0.000000 | False | int64 |
| image | 98.136528 | True | object |

Figure 5: Revised missing entries' density after rows without review nor summary has been dropped, and "vote" has been normalized and changed null value to 0



|  | non-null ratio (%) | unique count |
|---|---|---|
| **Color** | 7.187678 | 77 |
| **Design** | 0.399315 | 22 |
| **Flavor** | 0.228180 | 5 |
| **Scent Name** | 0.190150 | 9 |
| **Size** | 82.677315 | 290 |
| **Style Name** | 0.760601 | 5 |

Figure 6: EDA into style features, revealing that this variable is too sparse



```
# month name
df["month"] = df["unixReviewTime"].dt.strftime("%B")
df["month"].value_counts()
```

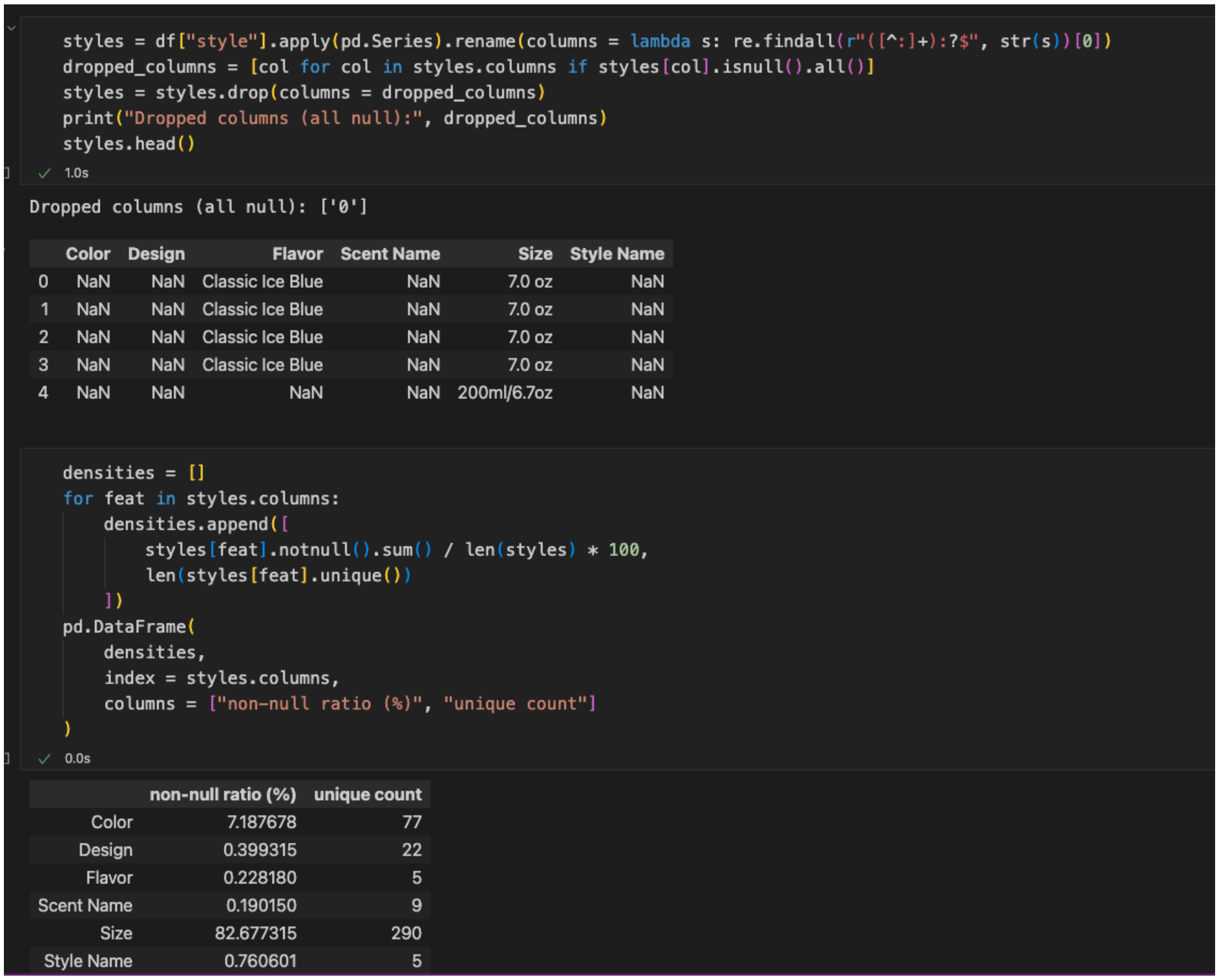
```
August       598
September    550
April        548
March        518
July         464
May          445
February     413
January      405
December     352
November     338
October      320
June         308
Name: month, dtype: int64
```

```
dummies = pd.get_dummies(df["month"], prefix = "month")
df = df.merge(dummies, left_index = True, right_index = True)
datetime_feats += dummies.columns.tolist()
```

```
# weekday
df["weekday"] = df["unixReviewTime"].dt.strftime("%A")
df["weekday"].value_counts()
```

```
Monday       891
Tuesday      819
Thursday     809
Saturday     717
Friday       715
Sunday       659
Wednesday    649
Name: weekday, dtype: int64
```

Figure 7: Date-time features transformation



```python
styles = df["style"].apply(pd.Series).rename(columns = lambda s: re.findall(r"([^:]+):?$", str(s))[0])
dropped_columns = [col for col in styles.columns if styles[col].isnull().all()]
styles = styles.drop(columns = dropped_columns)
print("Dropped columns (all null):", dropped_columns)
styles.head()
```

✓ 1.0s

```
Dropped columns (all null): ['0']
```

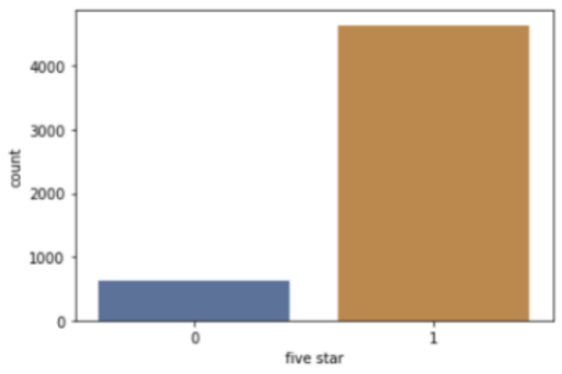
| | Color | Design | Flavor | Scent Name | Size | Style Name |
|---|---|---|---|---|---|---|
| 0 | NaN | NaN | Classic Ice Blue | NaN | 7.0 oz | NaN |
| 1 | NaN | NaN | Classic Ice Blue | NaN | 7.0 oz | NaN |
| 2 | NaN | NaN | Classic Ice Blue | NaN | 7.0 oz | NaN |
| 3 | NaN | NaN | Classic Ice Blue | NaN | 7.0 oz | NaN |
| 4 | NaN | NaN | NaN | NaN | 200ml/6.7oz | NaN |

```python
densities = []
for feat in styles.columns:
    densities.append([
        styles[feat].notnull().sum() / len(styles) * 100,
        len(styles[feat].unique())
    ])
pd.DataFrame(
    densities,
    index = styles.columns,
    columns = ["non-null ratio (%)", "unique count"]
)
```

✓ 0.0s

| | non-null ratio (%) | unique count |
|---|---|---|
| Color | 7.187678 | 77 |
| Design | 0.399315 | 22 |
| Flavor | 0.228180 | 5 |
| Scent Name | 0.190150 | 9 |
| Size | 82.677315 | 290 |
| Style Name | 0.760601 | 5 |

Figure 8: Missing data in style-specific features.



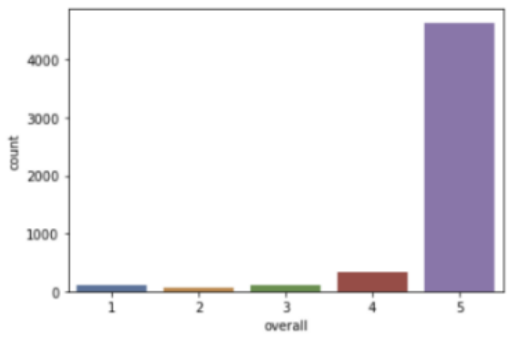
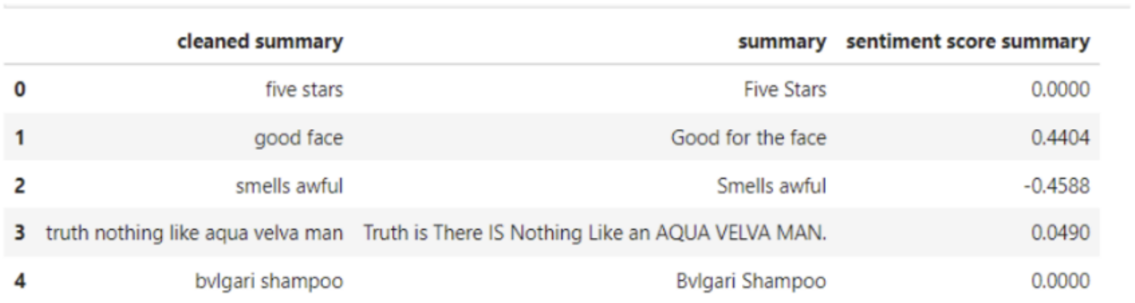
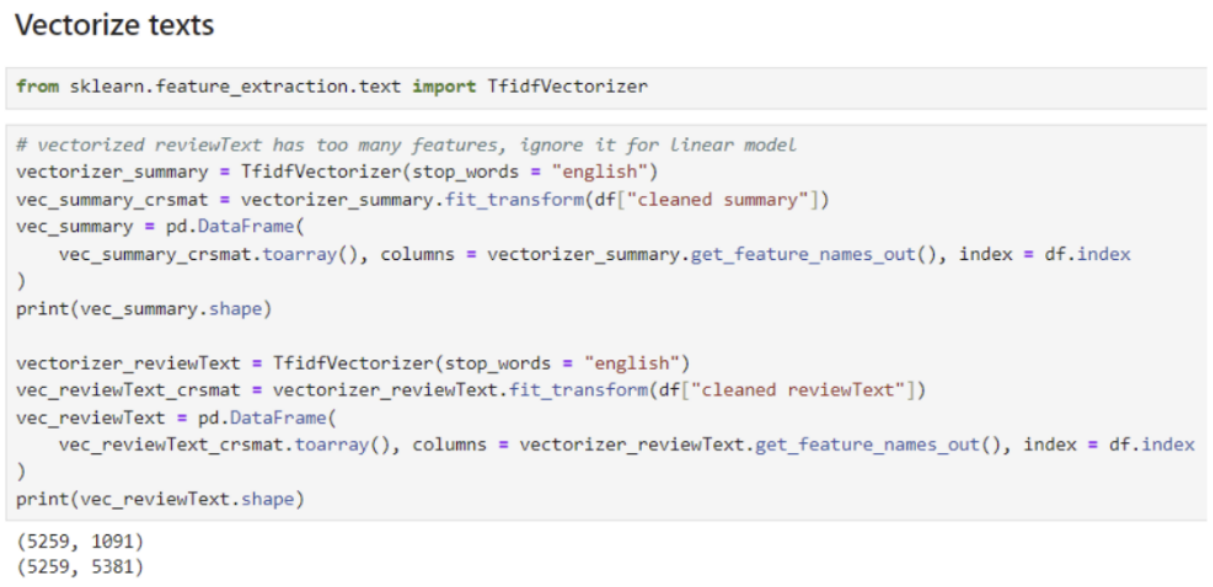


Figure 9: Distribution of five-star rating vs non-five-star rating (left) and distribution of overall rating (right)



| | cleaned summary | summary | sentiment score summary |
|---|---|---|---|
| 0 | five stars | Five Stars | 0.0000 |
| 1 | good face | Good for the face | 0.4404 |
| 2 | smells awful | Smells awful | -0.4588 |
| 3 | truth nothing like aqua velva man | Truth is There IS Nothing Like an AQUA VELVA MAN. | 0.0490 |
| 4 | bvlgari shampoo | Bvlgari Shampoo | 0.0000 |

Figure 10: Transformation of summary variable and creating sentiment score


## Vectorize texts

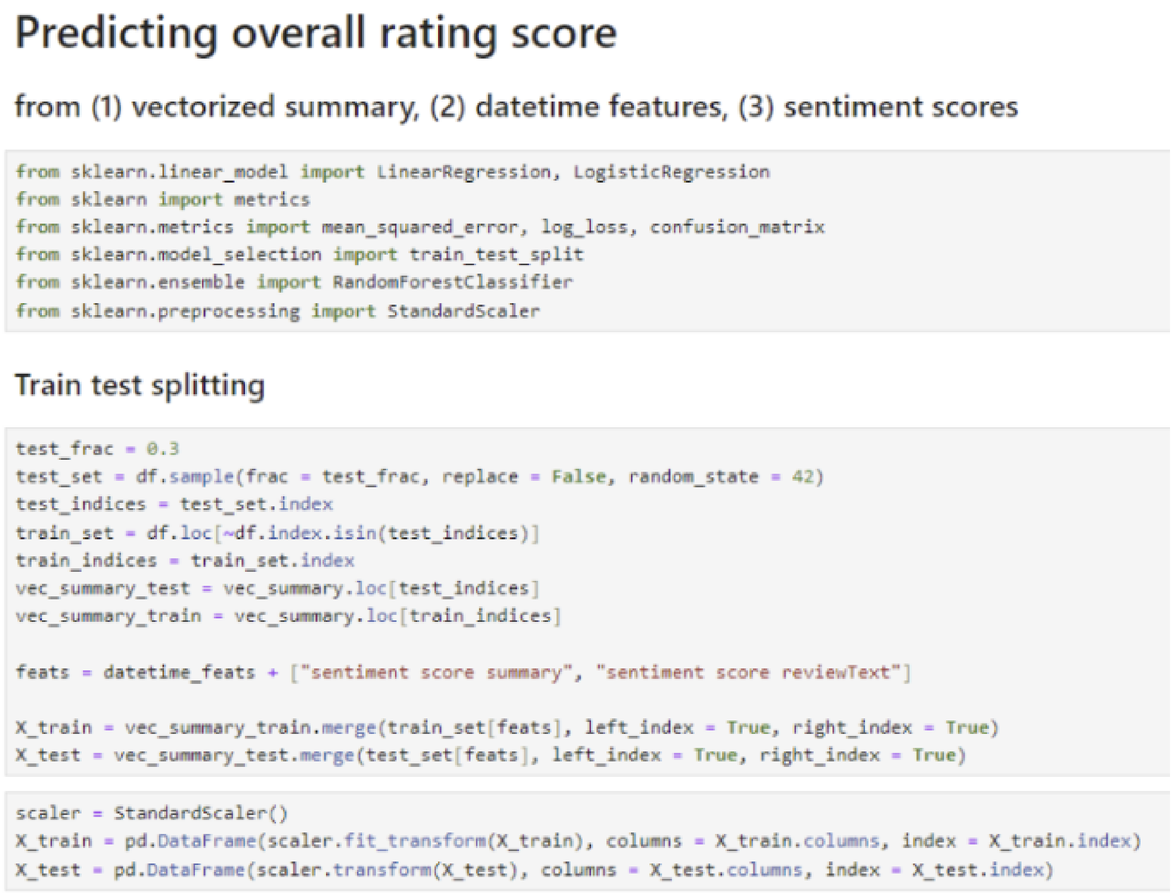
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# vectorized reviewText has too many features, ignore it for linear model
vectorizer_summary = TfidfVectorizer(stop_words = "english")
vec_summary_crsmat = vectorizer_summary.fit_transform(df["cleaned summary"])
vec_summary = pd.DataFrame(
    vec_summary_crsmat.toarray(), columns = vectorizer_summary.get_feature_names_out(), index = df.index
)
print(vec_summary.shape)

vectorizer_reviewText = TfidfVectorizer(stop_words = "english")
vec_reviewText_crsmat = vectorizer_reviewText.fit_transform(df["cleaned reviewText"])
vec_reviewText = pd.DataFrame(
    vec_reviewText_crsmat.toarray(), columns = vectorizer_reviewText.get_feature_names_out(), index = df.index
)
print(vec_reviewText.shape)
```

```
(5259, 1091)
(5259, 5381)
```

Figure 11: Vectorizing texts for reviewText and summary



## Predicting overall rating score

### from (1) vectorized summary, (2) datetime features, (3) sentiment scores

```
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn import metrics
from sklearn.metrics import mean_squared_error, log_loss, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
```

### Train test splitting

```
test_frac = 0.3
test_set = df.sample(frac = test_frac, replace = False, random_state = 42)
test_indices = test_set.index
train_set = df.loc[~df.index.isin(test_indices)]
train_indices = train_set.index
vec_summary_test = vec_summary.loc[test_indices]
vec_summary_train = vec_summary.loc[train_indices]

feats = datetime_feats + ["sentiment score summary", "sentiment score reviewText"]

X_train = vec_summary_train.merge(train_set[feats], left_index = True, right_index = True)
X_test = vec_summary_test.merge(test_set[feats], left_index = True, right_index = True)
```

```
scaler = StandardScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train), columns = X_train.columns, index = X_train.index)
X_test = pd.DataFrame(scaler.transform(X_test), columns = X_test.columns, index = X_test.index)
```

Figure 12: Training rating-predictor methodologies

```
fig, axes = plt.subplots(1, 2, figsize = (12, 7))
plt.subplots_adjust(wspace = 0.35)
sns.countplot(Y_train, ax = axes[0])
axes[0].set_title("training set")
sns.countplot(Y_test, ax = axes[1])
axes[1].set_title("testing set")
plt.suptitle("Category distribution on training and testing dataset")
```

```
Text(0.5, 0.98, 'Category distribution on training and testing dataset')
```

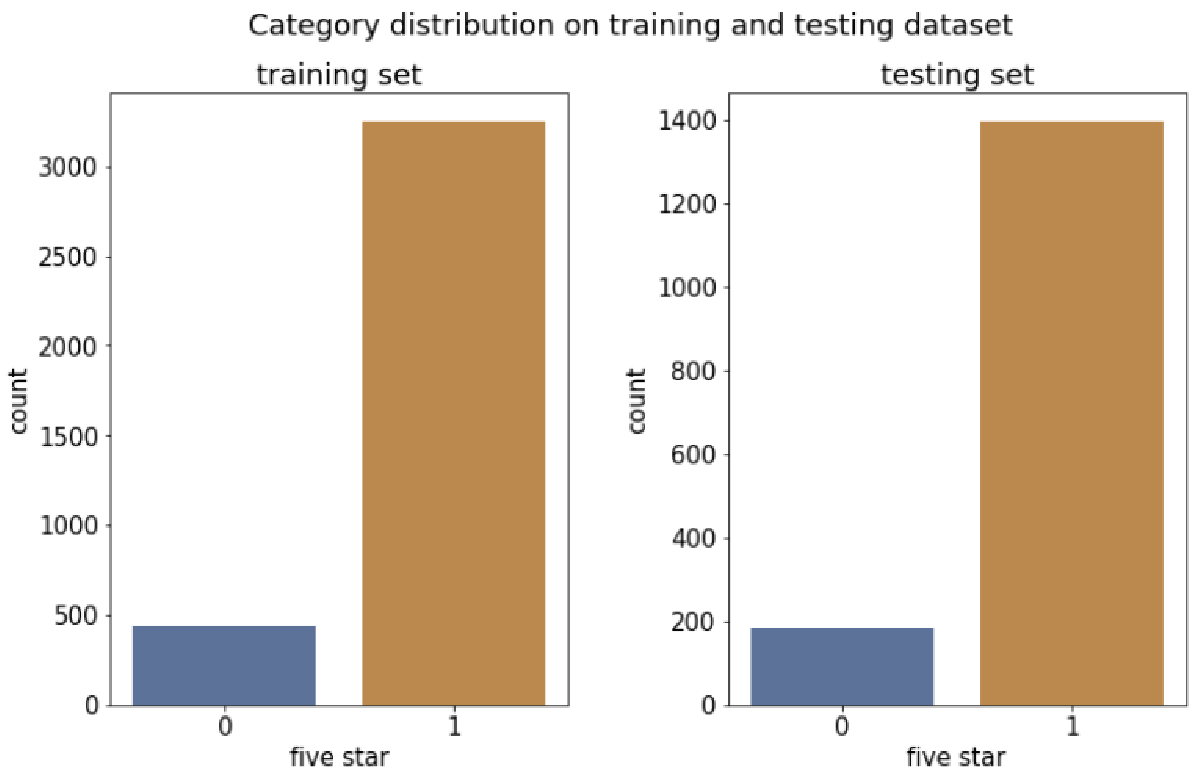


Figure 13: Logistic regression for predicting five-stars

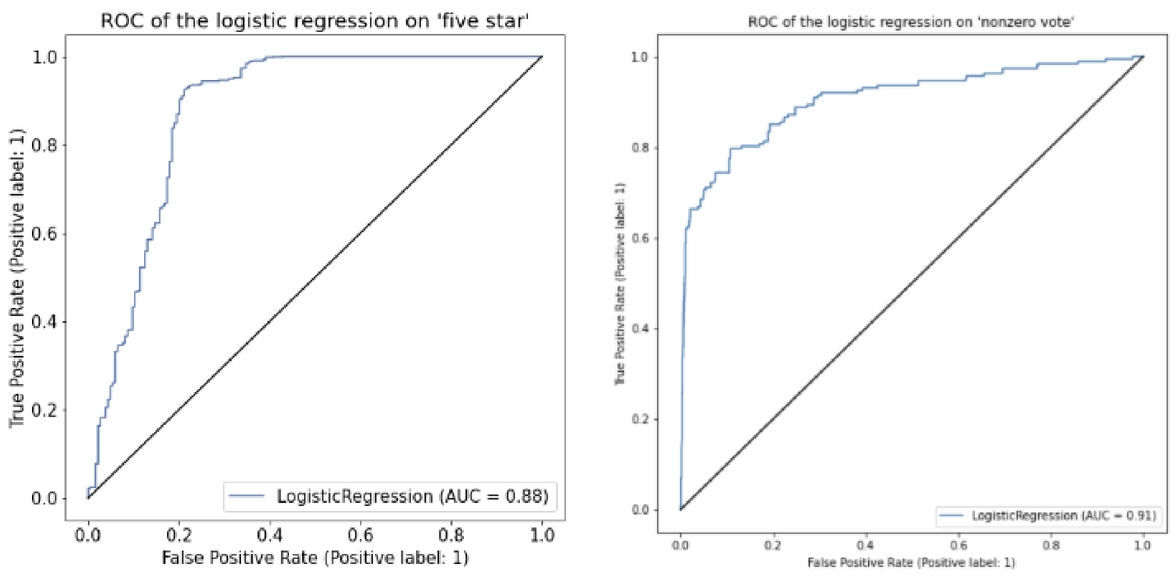


Figure 14: ROC of the logistic regression for predicting 'five star' and 'nonzero vote'

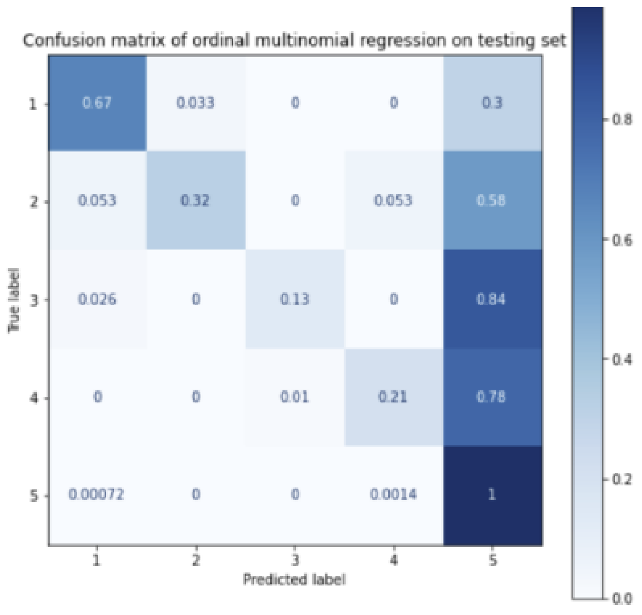


Figure 15: Confusion matrix of predicting specific rating value. Note the dataset is imbalanced.

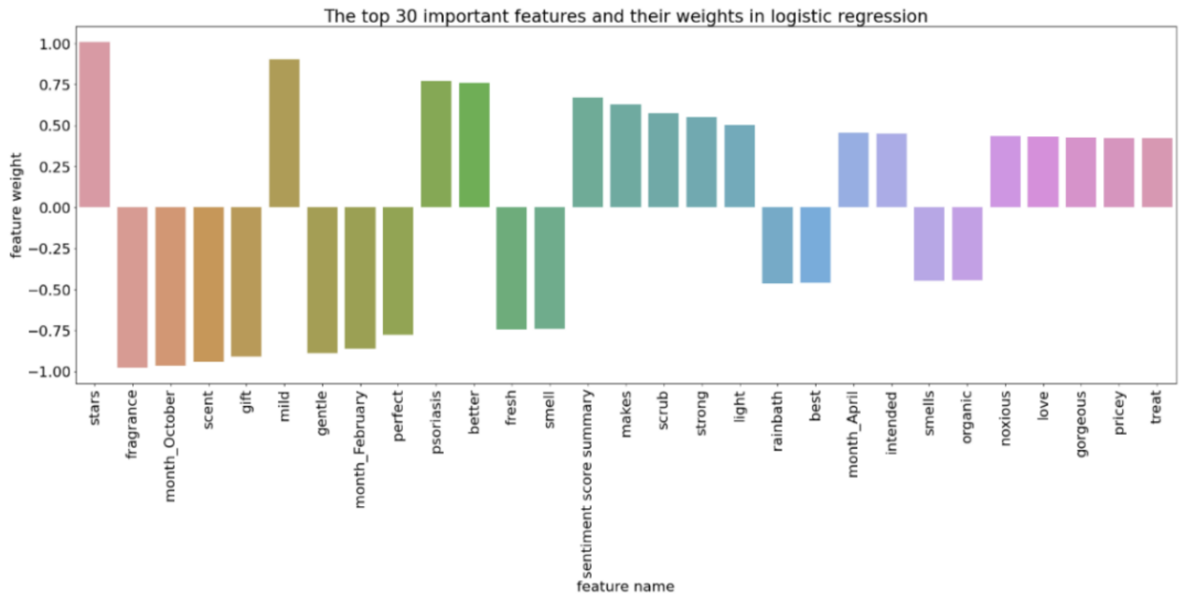


Figure 16: Top 30 important features and their weights in logistic regression.

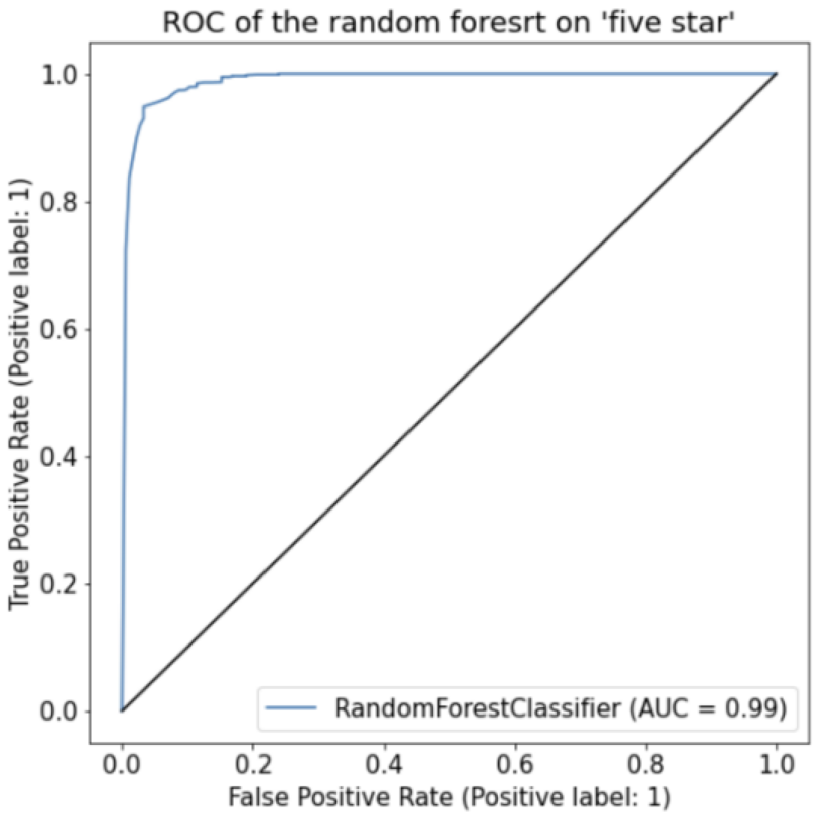


Figure 17: ROC of the Random Forest on ‘five star’

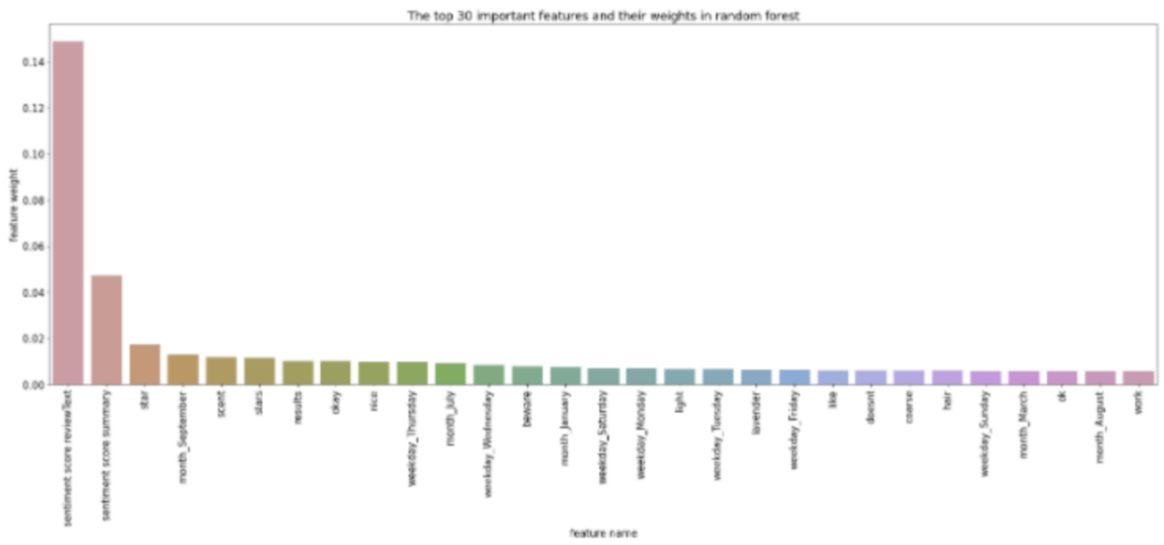


Figure 18: Top 30 important feature name and weight in the random forest model

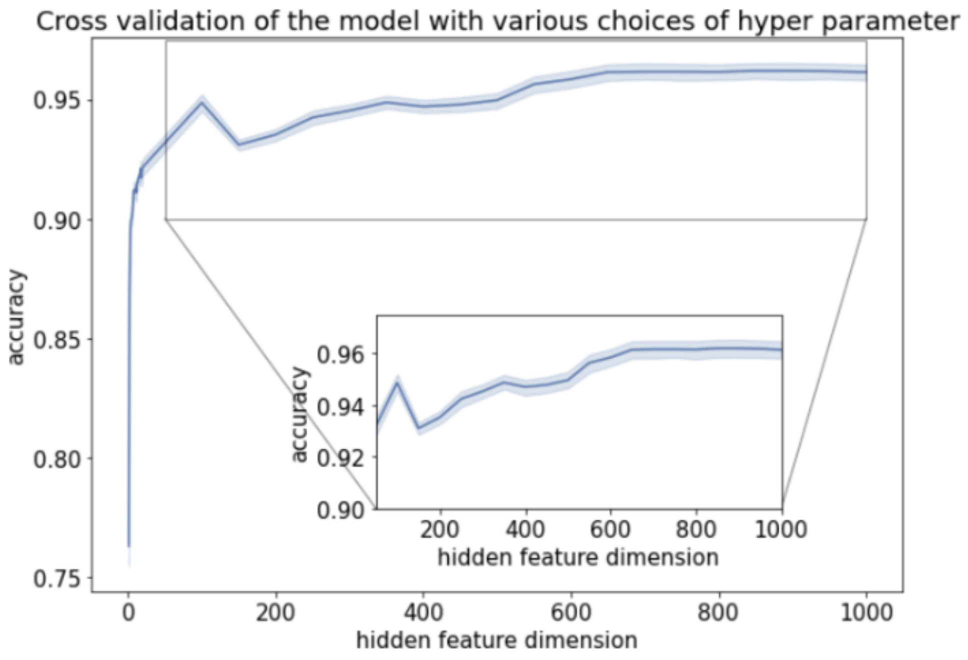


Figure 19: Cross validation of the model with various choices of hyper parameter

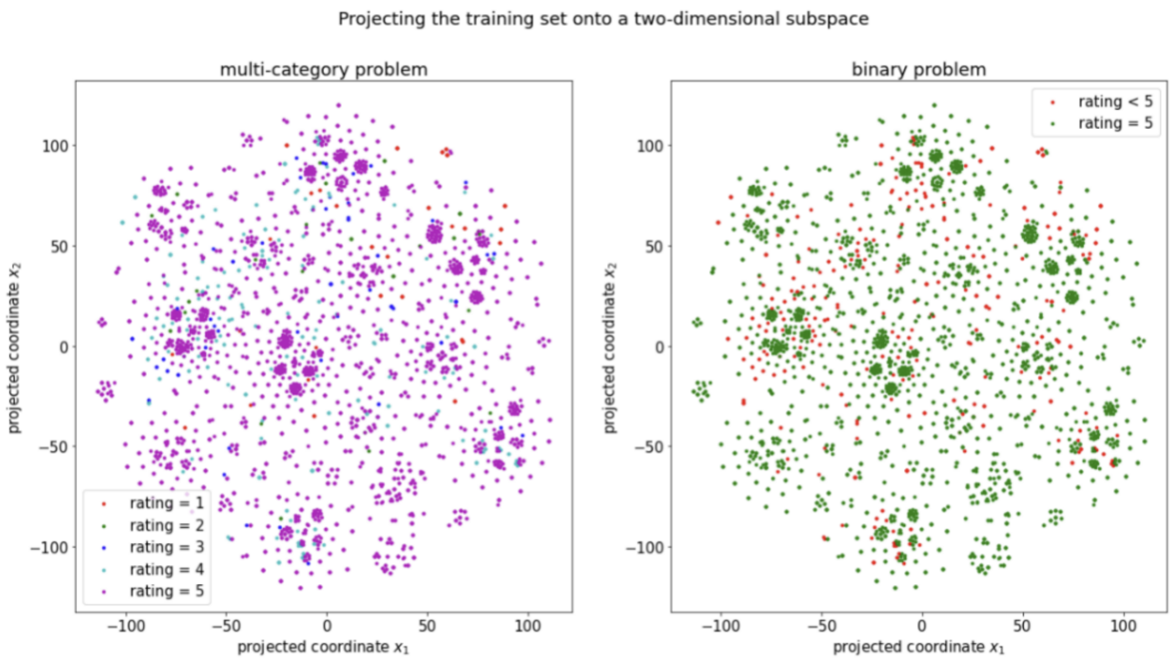


Figure 20: Projecting the training set onto a two-dimensional subspace

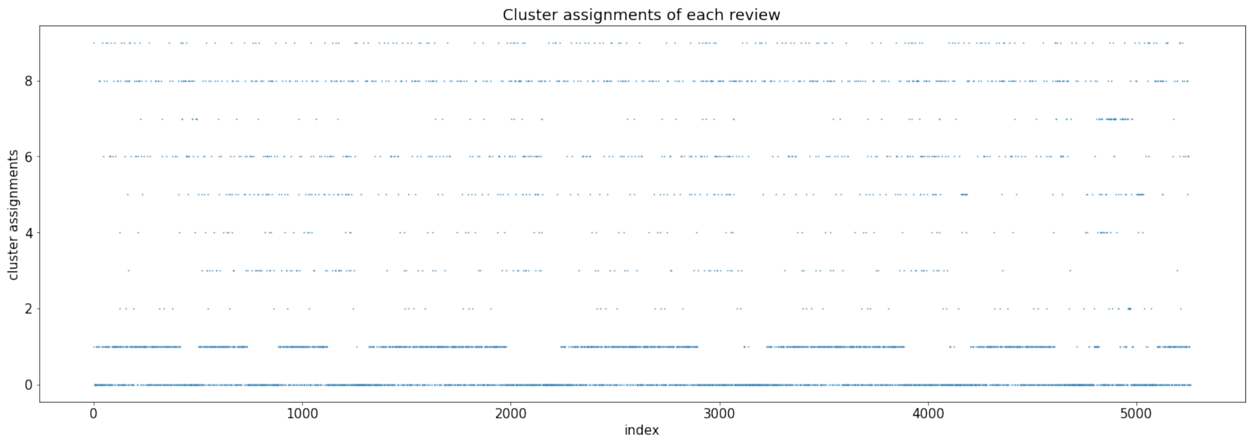
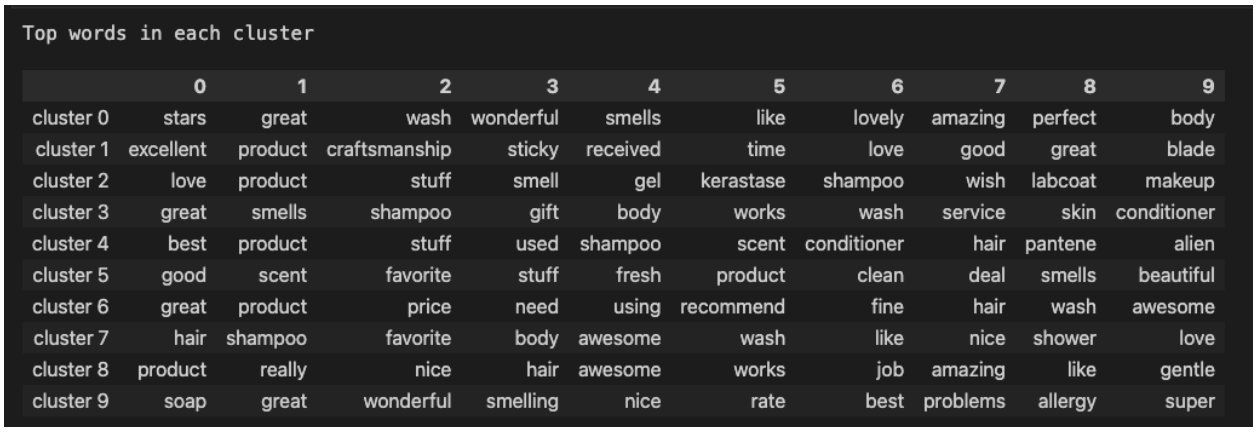


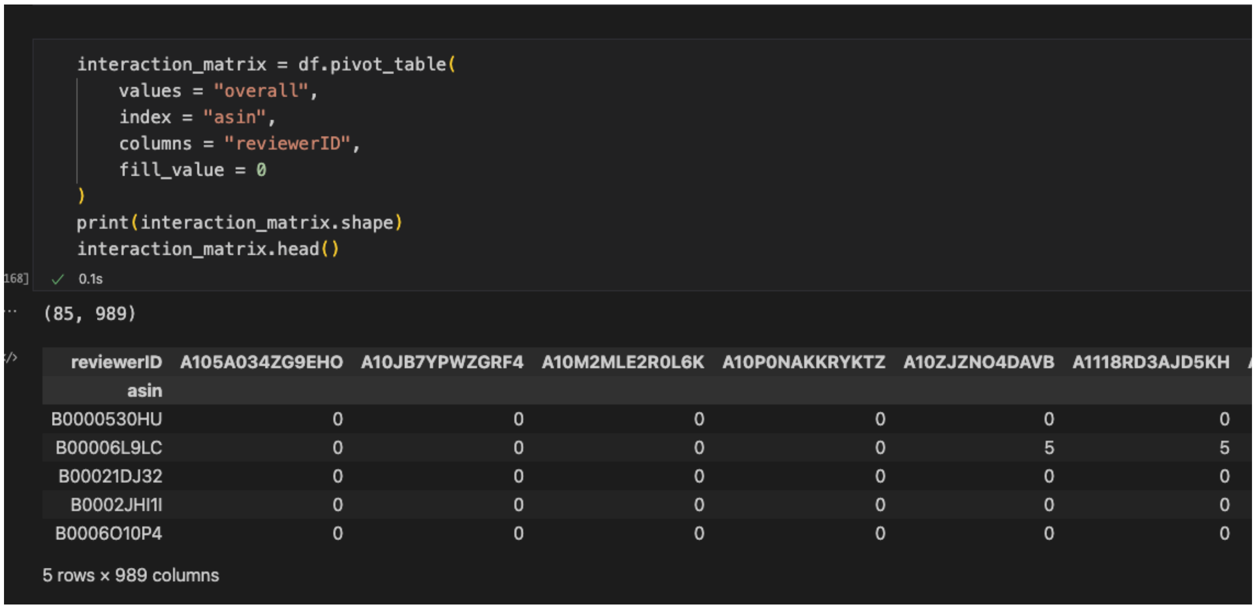
Figure 21: Cluster assignments of each review



Top words in each cluster

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| cluster 0 | stars | great | wash | wonderful | smells | like | lovely | amazing | perfect | body |
| cluster 1 | excellent | product | craftsmanship | sticky | received | time | love | good | great | blade |
| cluster 2 | love | product | stuff | smell | gel | kerastase | shampoo | wish | labcoat | makeup |
| cluster 3 | great | smells | shampoo | gift | body | works | wash | service | skin | conditioner |
| cluster 4 | best | product | stuff | used | shampoo | scent | conditioner | hair | pantene | alien |
| cluster 5 | good | scent | favorite | stuff | fresh | product | clean | deal | smells | beautiful |
| cluster 6 | great | product | price | need | using | recommend | fine | hair | wash | awesome |
| cluster 7 | hair | shampoo | favorite | body | awesome | wash | like | nice | shower | love |
| cluster 8 | product | really | nice | hair | awesome | works | job | amazing | like | gentle |
| cluster 9 | soap | great | wonderful | smelling | nice | rate | best | problems | allergy | super |

Figure 22: Top 10 highest weighted words in each cluster.



```python
interaction_matrix = df.pivot_table(
    values = "overall",
    index = "asin",
    columns = "reviewerID",
    fill_value = 0
)
print(interaction_matrix.shape)
interaction_matrix.head()
```

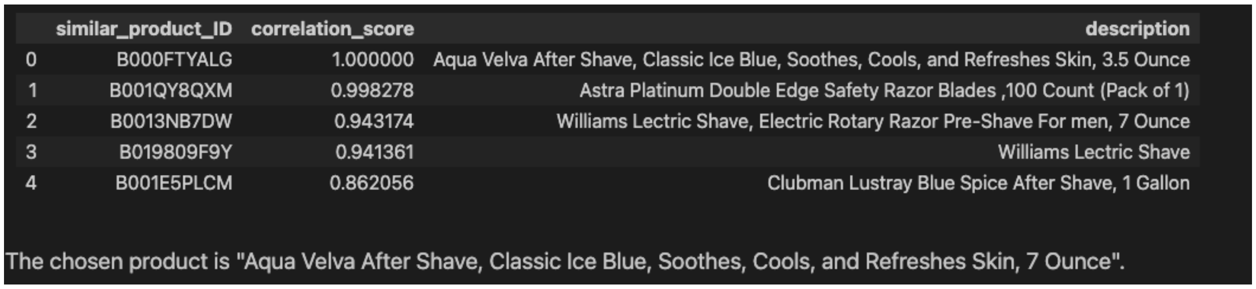
[168] ✓ 0.1s

```
(85, 989)
```

| reviewerID | A105A034ZG9EHO | A10JB7YPWZGRF4 | A10M2MLE2R0L6K | A10P0NAKKRYKTZ | A10ZJZNO4DAVB | A1118RD3AJD5KH |
|---|---|---|---|---|---|---|
| asin | | | | | | |
| B0000530HU | 0 | 0 | 0 | 0 | 0 | 0 |
| B00006L9LC | 0 | 0 | 0 | 0 | 5 | 5 |
| B00021DJ32 | 0 | 0 | 0 | 0 | 0 | 0 |
| B0002JHI1I | 0 | 0 | 0 | 0 | 0 | 0 |
| B0006O10P4 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 989 columns

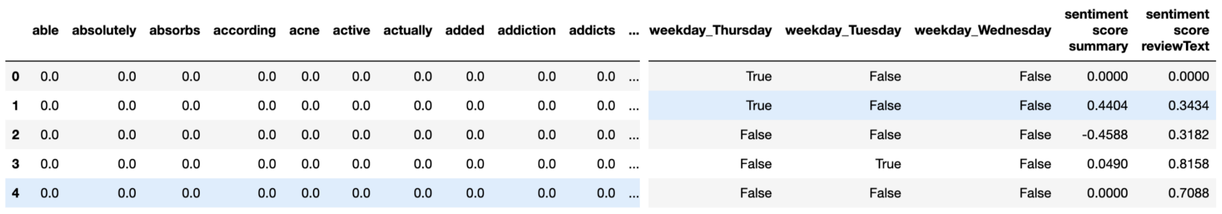
Figure 23: Computation of engagement matrix.



```
   similar_product_ID  correlation_score                                                                      description
0          B000FTYALG           1.000000  Aqua Velva After Shave, Classic Ice Blue, Soothes, Cools, and Refreshes Skin, 3.5 Ounce
1          B001QY8QXM           0.998278         Astra Platinum Double Edge Safety Razor Blades ,100 Count (Pack of 1)
2          B0013NB7DW           0.943174       Williams Lectric Shave, Electric Rotary Razor Pre-Shave For men, 7 Ounce
3          B019809F9Y           0.941361                                                            Williams Lectric Shave
4          B001E5PLCM           0.862056                                   Clubman Lustray Blue Spice After Shave, 1 Gallon

The chosen product is "Aqua Velva After Shave, Classic Ice Blue, Soothes, Cools, and Refreshes Skin, 7 Ounce".
```

Figure 24: An example output of the recommendation system.



| | able | absolutely | absorbs | according | acne | active | actually | added | addiction | addicts | ... | weekday_Thursday | weekday_Tuesday | weekday_Wednesday | sentiment score summary | sentiment score reviewText |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | True | False | False | 0.0000 | 0.0000 |
| 1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | True | False | False | 0.4404 | 0.3434 |
| 2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | False | False | False | -0.4588 | 0.3182 |
| 3 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | False | True | False | 0.0490 | 0.8158 |
| 4 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | False | False | False | 0.0000 | 0.7088 |

Figure 25: Preview of feature engineering.

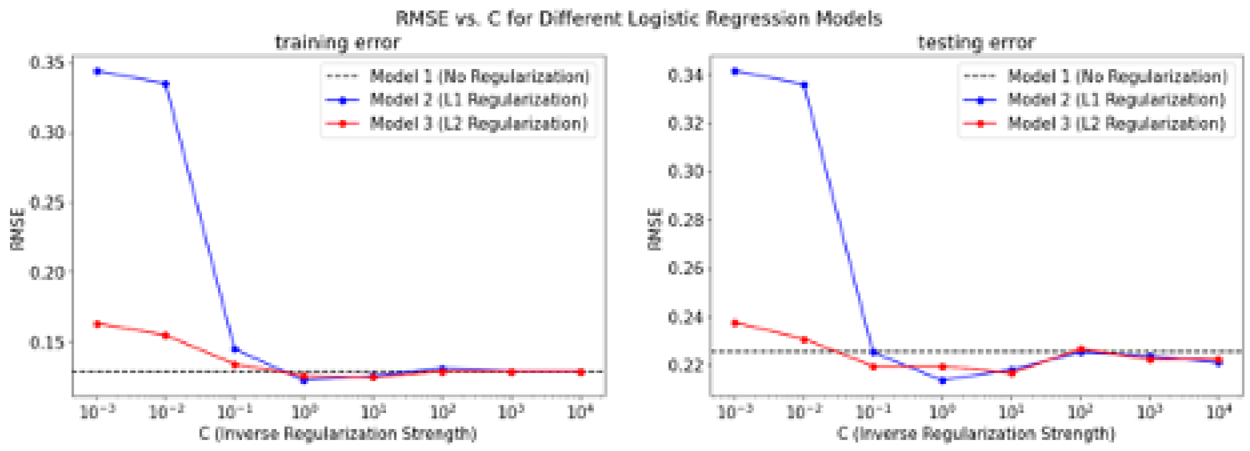


Figure 26: RMSE on training and testing dataset as a function of the inverse regularization strength.