Problem Chosen	2021 MCM/ICM Summary Sheet	Team Control Number
C		2015193

Amazon Online Shopping: An NLP-Based User Behavior and Commodity Value Analyse

Summary

While online shopping platforms such as Amazon offer customers online shopping opportunities, they also facilitate related companies to design sales strategies based on customer and product conditions. As a consultant, our team modeled and analyzed the problem with ratings and reviews as research objects.

To model the relationship between star rating, review and usefulness rating is essentially to find out the commonality of the three. We combine several machine learning methods to classify the comment text negatively and positively. It is found that the matchaccuracy of linear support vector machines to microwave and hairdryers was 0.98 and 0.96, respectively, and the best results were compared with the rating measures, excluding mismatched data.

Our model use NLP to obtain a comprehensive score G for emotional orientation calculations. Based on the importance of help degree H, validity E, and professional P to the role of reviews. Then construct the GHEP model to obtain the value which is $C^* = A \cdot E \cdot P \cdot G$. The quantified values of ratings and ratings are normalized and weighted to form the review metric M. Then find three product models with the highest evaluation measures for hairdryer, microwave and pacifier which are 591023894, 771401205 and 246038397. The evaluation metrics are 0.996, 0.987, 0.996 , with the feature of professional, countertop and infant respectively. Through studying the change in product reviews in different years, we find that the evaluation of hairdryer and microwave has only a small range of fluctuations in the later stage and the overall evaluation curve tilted upward, which indicates that the reputation of these two types of goods in the market is gradually rising. Taking the three product types with the highest ratings as examples, the evaluation metrics of these products have been slowly decreasing since their first online sales in 2012, which are potential failure products.

For causal inferences about the relationship between stars and reviews, using bootstrap-based PSM, three models with the most products among the three categories were extracted. Randomly sort thees nine sub-samples. After using multiple Logistic Regression to estimate the propensity score by MLE and calculate the ATT estimate after the k-nearest neighbor method is successfully matched. It is found that high star ratings can significantly cause the highest 55.2%, the lowest 0.2% of the review sentiment scores rise, otherwise decreasing. By selecting the seven specific descriptors with the highest frequency of occurrence from the previously trained thesaurus, drawing their corresponding violin plots in the microwave and hair dryerdictionaries. We find that words with positive emotions such as perfect whose probability distribution is concentrated in high star ratings. On the contrary low quality descriptors such as disappointed are closely related to low star ratings.

Then we submitted a report to the Marketing Director of Sunshine Company, summarized the topics we studied, and proposed a six-point strategy for the company's sales based on the laws discovered by our research.

Finally, we conducted a Sensitivity Analysis and recognize that when changing the value of n, our model is robust.

Keywords: Evaluation Measure; GHEP Model; Machine Learning; Propensity Score Matching; NLP

Contents

1 Introduction

Online shopping platforms such as Amazon have become an essential means of life, whose share price once at a time up to more than $2,050.50. Unlike traditional offline retailing, online shopping offers more easily access to customer pre-purchase needs and post-purchase feedback, even the frequency, amount and time interval of their purchases. Customer evaluation information on goods online helps companies grasp market preferences and develop corresponding marketing strategies, which requires us to use these big data information effectively to study the time dimension of the ratings, reviews and help scale models of microwave ovens, pacifiers and hairdryers.

Here the personal rating refers to the customer via a value of 1 to 5 to express the satisfaction of the product. While the comment refers to the customer's text message and help rating refers to whether the comment on the customer's purchase decision is helpful. We will do the following to determine the sales strategy and functional design:

- Via machine learning for emotional matching, select the best classification algorithm and through which reject the mismatched data
- With the help of natural language processing (NLP) for emotional analysis, we get the comprehensive score G of emotional tendency calculation and combine the help degree, effectiveness and professional construction model to calculate the comprehensive evaluation measurable value
- Observe the trends of large categories and a certain model of products at the sight of a time dimension, consequently identifying market ingresss and product design strategies
- Infer the causality of past star rating from current reviews and the correlation between the keywords of the review and the star rating

2 Assumptions and Symbols

2.1 Model Hypothesis

- Assuming that all e-commerce companies are honest operation, there is no shop owner take a brush and praise behavior to improve product sales
- Suppose Amazon vine members and verified buyers (i.e. verify_purchase) play a greater role in the product's reviews than non-vine members and non-buyers
- Assuming that the difference in evaluation measures is small for different months, i.e. the annual evaluation measure can be obtained through the average of the annual daily evaluation measure
- When taking samples by bootstrap method, the number of bootstrap samples B is set at 500 in order to improve the accuracy of the hypothesis test

- Since there are a limited number of control groups in the sample sorted by product model (no more than 25% of the total), there is a returned sample match in the PSM based on bootstrap

2.2 Symbols and Definitions

Table 1: Notations

Symbols	Description
G	Grade
H	Helpfulness
E	Effectivity
P	Professional
V_h	Helpful Votes
V_t	Total Votes
A	Help Multiplier
C^*	Comment Quantification Value
R	Standardization of Rating
C	Standardization of C^*
M	Overall Measures
α	Weight Factor
ATT	Average Effect Treatment on the Treated
$\mathbb{P}(\boldsymbol{X}_i)$	The Propensity Score of the Individual i

3 Establishing the model

3.1 Establishment of Star Rating and Review Relationships

Before we doing all the modeling work, that whose ratings and comment text (review_body) emotionally inconsistenced customer reviews need to be filtered and culled to reduce the impact of haphazardly rated customers on the overall product rating. Build a qualitative pattern of star ratings, reviews, and usefulness ratings through machine learning algorithms and calculate the accuracy of various algorithms to select the optimal algorithm to identify the relationship between star rating and reviews.

3.1.1 The Process of Reviews Emotion Analysis

Machine Learning Emotion Analysis based on Python, that is to select a portion of positive sentiment comment text and negative sentiment comment text. Then use machine learning methods to train so that getting the emotional classifier and use this emotional classifier to classify all comment texts positively and negatively[?]. This module aims to perform sentiment analysis on all customer reviews to determine whether they are positive or negative comments and then compare them with rating metrics to remove those reviews that are inconsistent with the rating.

Via machine learning to determine the emotional tendency of the test text to be tested[?], that is, positive or negative. Since a result needs to be predicted based on a given input and there should be an example of an input or output pair, it belongs to a supervised two-category machine learning, with class labels neg (negative) and pos (positive).

Firstly clear the missing and unreasonable data caused by misalignment[?], the data should be processed into an acceptable format for the nltk thesis. Defining the label star rating follows distributed 0-1 by whether it is greater than 3 (greater than 3 marks positive, otherwise negative). Comments are then processed through nltk's thesaurus, including word-breaking, removal of English symbols, removal of de-stop words[?] (e.g. I, you, etc.), acquisition of trunk words (excluding problems caused by tenses, idioms, lexical composition, etc.), standardizing comments through regular expressions and then splitting them into good_text and bad_text two lists (store positive and negative star reviews) respectively.

Next calculate the frequency of words in positive text and the frequency of words in negative text through chi-square statistics and obtain feature words with higher information content. The information content value of a word is equal to the positive chi-square statistics plus negative chi-square statistics[?]. Process the data into a data form that can be processed by the machine learning library that comes with nltk, disrupt the data order, split the training set and test set, select 1/3 of the comment text as the training set, and all the comment text as the test set[?]. Finally, use various machine learning algorithms to verify, and compare the prediction accuracy scores through experiments to select the best classification algorithm. The specific process is as follows:

3.1.2 Experimental Results and Data Preprocessing

According to the described process, we use Logistic Regression, SVC, LinearSVC, NuSVC, MultinomialNB, BernoulliNB and other machine learning algorithms to build a qualitative relationship[?] between star rating and review metrics.

Figure **??** is the accuracy of classification matching obtained by various machine learning algorithms. It is found that the accuracy of linear support vector machine for emotion evaluation of microwave and pacifier is higher than other methods. Therefore, we use linear support vector machine(LinearSVC) for emotion matching. According to the experimental results, some mismatched data were found and filtered and comments with titles[?] that did not match the text were eliminated.

3.2 Construction of Comment Measures: GHEP Model

The basic idea of the GHEP model is to judge the customer's evaluation of a certain product through the four most meaningful indicators in customer reviews, that is, G: Grade, H: Helpfulness, E: Effectivity, P: Professional. The scores represent the ratings for emotional words extracted from natural language processing (NLP). The degree of help is determined by the number of useful votes and the total number of votes while the validity refers to the comment customer being a verified purchaser (i.e. verify_purchase)

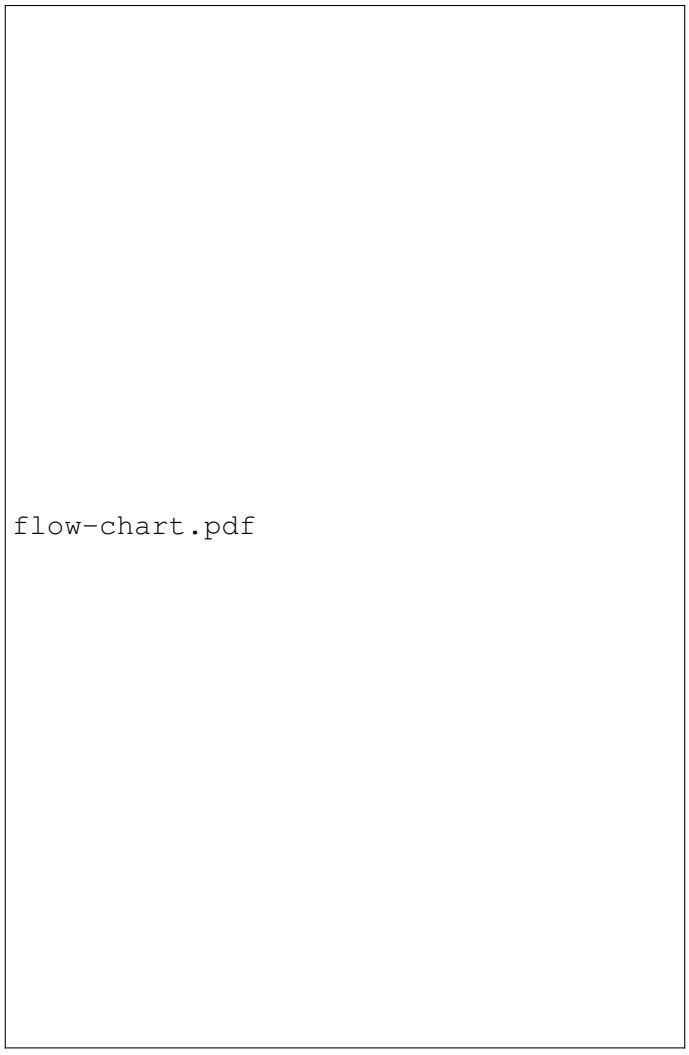


Figure 1: Flowchart of Algorithm

and the professional comment customer is Amazon Professional Commentator (vine). Where H, E, P can reflect the importance of the review.

3.2.1 Determination of Score G based on NLP

Instead of using machine learning algorithms, we do short text emotional tendencies by matching dictionaries different from Part 3.1. Use NLTK's built-in Vader sentiment analyzer to classify comments as positive, negative or neutral using the vocabulary of positive and negative vocabulary and calculate their proportion. Then conduct a comprehensive evaluation of the proportion of positive, negative and neutral terms. The comprehensive score of each comment is the compound value, which is defined as G'. This process is implemented by Python. After normalization, the score G is obtained $(0 < G < 1)$.

$$G = \frac{G' - G_{\min}}{G_{\max} - G_{\min}} \tag{1}$$

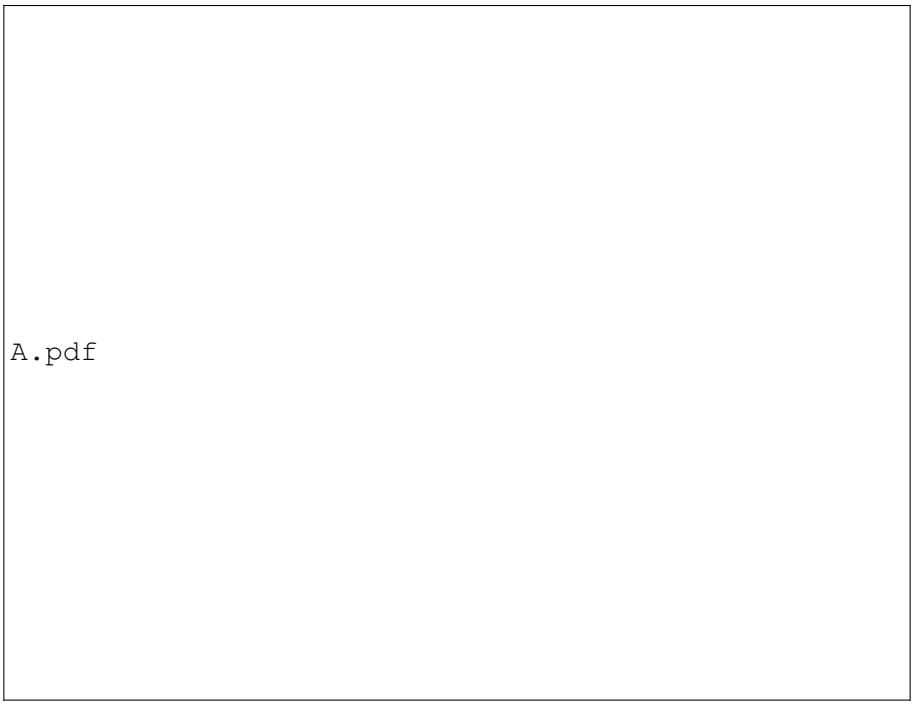


Figure 2: Accuracy Comparison of Machine Learning Algorithm

3.2.2 Setting of other Indicators

According to the assumption, vine members and verify_purchase have a greater effect on the evaluation. Considering that validity and professionalism are $0-1$ variables, weighting as an independent index is not suitable for determining the weight. Moreover, they represent the authenticity and professionalism of reviews and should strengthen the review score G, including strengthening the quantitative value of positive reviews and reducing the quantitative value of negative reviews. Therefore, the characteristics of validity E and professional P should be: when the score G of the review is greater than 0.5, E and P should be able to increase the quantified value of the review; when the score G of the review is less than 0.5, E and P should be able to make the comment quantization value is reduced. Here we set the comment validity E and comment specialty P as

$$E = \begin{cases} (2G)^2 = 4G^2 & \text{Purchased} \\ 1 & \text{Not Purchased} \end{cases} \tag{2}$$

$$P = \begin{cases} (2G)^2 = 4G^2 & \text{Professional} \\ 1 & \text{Non-Professional} \end{cases} \tag{3}$$

The degree of help must not only consider the useful vote rate (the ratio of useful votes to the total votes), but also need to pay attention to our useful votes. For example, 99 useful votes with 100 total votes and a comment useful votes 1 with the total number of votes is 1. Comparing these two, it is clear that the useful vote rate of the former is smaller than the latter, but the useful vote number of the former is much larger than the

latter. The degree of help should be much larger than the latter. Therefore, the useful vote rate cannot be simply used to measure help degree.

In this way, we need to comprehensively consider the useful vote rate and the useful vote number. And if the number of useful votes is too large, the degree of help H will be much greater than the degree of validity E and professional P, then the role of E and P will be diluted when calculating the quantitative value of the review. Therefore, after comprehensive consideration, we define the degree of help as the product of the useful vote rate and the cube root of the useful votes:

$$H = \begin{cases} \frac{V_h}{V_t} \times \sqrt[3]{V_h} = \frac{V_h^{\frac{4}{3}}}{V_t} & V_t \neq 0 \text{ and } 0 < V_h \leq 100 \\ \sqrt[3]{100} = 4.64159 & V_t \neq 0 \text{ and } V_h > 100 \\ 1 & V_t = 0 \end{cases} \tag{4}$$

Where V_h is the number of useful votes, i.e. helpful_votes in the data set, and V_t is the total number of votes, i.e. total_votes in the data set.

In addition, the help degree is only a rating of the usefulness of the review, rather than a quantitative value indicating the review. This is not the same as the score of the star rating. Help degree cannot measure a comment as a separate metric which only affects the weight of score G. When the score G of the review is large, H should be able to increase the quantization value of the review; when the score G of the review is small, H should be able to reduce the quantization value of the review. So the helper multiplier is defined as

$$A = \begin{cases} H & G > 0.6 \\ 1 & 0.4 \leq G \leq 0.6 \\ 1/H & G < 0.4 \end{cases} \tag{5}$$

3.2.3 Comment Quantitative Value C*

We have already quantified the four indicators of G, H, E, P, and the GHEP model shows that the quantitative value of the review is determined by the score G as the base quantity and the helper coefficient A, validity E, and professional P as the enlargement or reduction multiplier. The product of them is the quantified value of the review.

$$C^* = A \times E \times P \times G \tag{6}$$

3.3 Rating-Review-based Evaluation Metrics

The rating has been directly scored by the customer, ranging from 1-5. We use this data as the rating score. However, there is a large difference in the range of measurement values for ratings and reviews, the original data of the two are normalized. For cost indicators, the score of a review or rating is directly proportional to the value of the indicator, consequently we have:

$$R = \frac{R' - R_{\min}}{R_{\max} - R_{\min}} \qquad C = \frac{C' - C_{\min}}{C_{\max} - C_{\min}} \tag{7}$$

So a single evaluation metric is a linear weighting of the two:

$$M = \alpha_1 R + \alpha_2 C \tag{8}$$

Where α_1 and α_2 are weights. According to the concepts of information and entropy from information theory, the information entropy of R and C indicators can be calculated to obtain the respective information entropy. On this basis, the weights of each evaluation index are calculated to obtainthe value of α_1 and α_2. We selected the specific product with the highest evaluation metric among the three types of products as Table X shows: Through comparison, we found that the three categories of new products such as hair dryers, microwave ovens and pacifiers (product_parent) ranked in the top five:
Haidryer: 591023894, 391944105, 47684938, 670161917, 16483457
Microwave Oven: 771401205, 692404913, 464779766, 423421857, 305608994
Pacifier: 246038397, 392768822, 392768822, 820728887, 491213241
Based on their slogan for product_title, we found that highly rated products in each model have these characteristics:
Haidryer:conair, sed, revolution, professional, ionic, micro, turbo, fold, etc.
Microwave Oven:counter, slim fry, grilling, ceramic enamel, countertop, whirlpool, sharp, over-the-range, etc. Mostly three stars.
Pacifier: free, infant, booster, etc. Mostly wubbanub.

3.4 Various Product Metrics

There are many users commenting on different products in different periods. Although the above has obtained a single evaluation metric, the comments of different customers in different periods are greatly different. In order to obtain the measurement value of a specific type of product, we simply use arithmetic average to get product measure-

Table 2: Product Index Information

	Hairdryer	Microwave	Pacifier
Product Number	591023894	771401205	246038397
G	0.933525	0.99469	0.99438
E	3.485876	3.95771	3.95515
P	3.485876	1	1
H	1.870705	4.64159	4.64159
A	1.870705	4.64159	4.64159
C^*	21.22047	18.27265	18.25494
R	1	1	1
C	1	0.98418	1
α_1	0.317053	0.412355	0.30435
α_2	0.682947	0.587645	0.69569
M	0.996	0.986759	0.996

ment value $M^{'}$:

$$M^{'} = \sum_{i=1}^{n} M_i/n \tag{9}$$

Where n is the number of evaluations and the evaluation metrics of the hair dryer, microwave oven, pacifier are calculated as:

$$M_1 = 0.3099, M_2 = 0.3085, M_3 = 0.3270$$

4 Solving the Model

This part mainly uses the model established above to solve the following problems.

4.1 Time Effects of Product Measures

We use the fitting method to study the changes in product evaluation at different times. Considering that some years have only one day of evaluation data and some years have more than 300 days of data. Based on the assumption that the evaluations converge in the short-term month, there is no necessity to over-refine. Consequently this article uses the annual measurement method to calculate the average evaluation value of the hair dryer, microwave oven, pacifier and some specific products for the j^{th} year.

$$M^{''} = \sum_{i=1}^{n} M_{ijk}/n \tag{10}$$

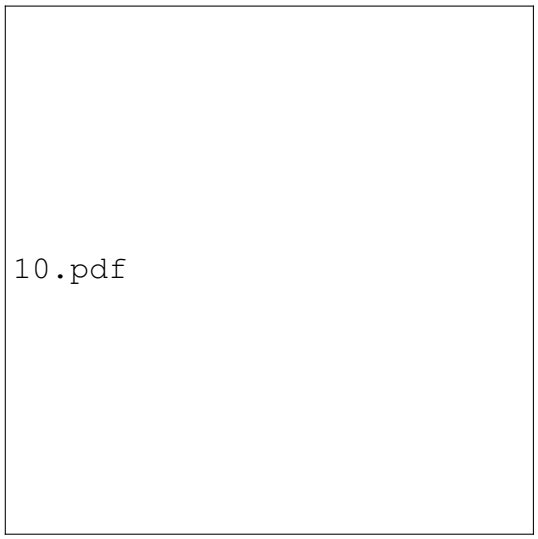
Where i is the number of reviews, j is the year, and k is the product category. We use hairdryer and microwave as examples for analysis. The same method can be used for pacifier.

4.1.1 Hairdryer

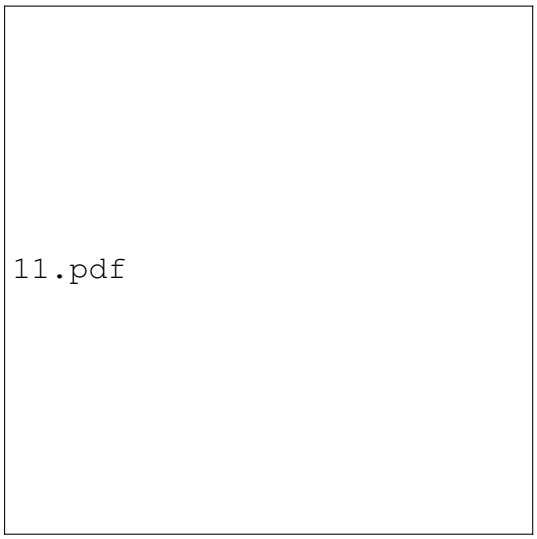
Via the quadratic method to fit, it is obvious that the earlier reputation evaluation is more volatile, which may be related to the small sample size of the statistical evaluation in the early years, which caused the average evaluation value to be easily affected by extreme values and hairdryer just entered the market to derogatory differentiation seriously related. However, it can still be found that the reputation of the hair dryer is generally slowly increasing upwards. If the polynomial method is used for fitting, this trend is even more obvious. So the reputation of hair dryers on the market is rising.

4.1.2 Microwave

Via Sum of Sine fitting, it is found that the reputation evaluation of microwave also fluctuates greatly with the years in the early years. However, reputation evaluation has steadily increased in recent years. Although there is a current downward trend, it will still increase with small fluctuations in the long run. The average rating value over the years are respectively as Figure **??** shows.



(a) Hairdryer



(b) Microwave

Figure 3: Average Rating Value over the years

4.2 The Outlook of Product

Based on the figure of the relationship between the evaluation metric M and the date t established by the rating-comment combination established in Part 3.3, we need to find out the trend of product evaluation development, distinguish and judge potential successful or failed products.

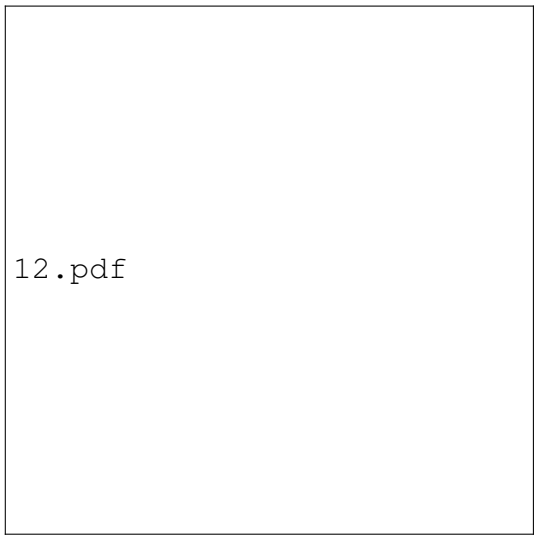
Taking the model 771401205 with the highest microwave oven evaluation value as an example, we found that this product has only been sold since 2012. Due to insufficient data sets, it is not suitable to study changes in reputation evaluation in units of years. Therefore, the time of a single review is used as the time node. The same way hairdryer 591023894 model products, pacifier 246038397 model as an example to do similar research, as the following Figure **??** shows:

The evaluation measurement value of Model 771401205 microwave oven has a downward trend year by year, just as the low score evaluation has increased over the years. We can judge it as a potential failure product. However, Model 591023894 hairdryer only had an extremely high evaluation metric in 2012, and it has also been obviously going downhill in recent years. We can also judge it as a potentially failed product. The pacifier model 246038397 can also be studied in a similar way, which will not be described here.

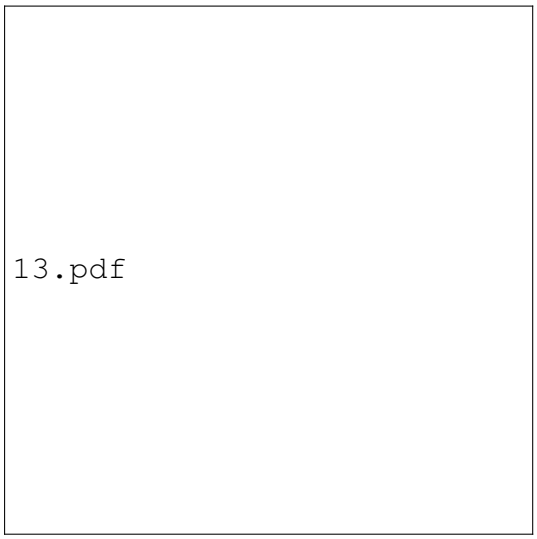
4.3 Causal Inference of Specific Star Ratings and Reviews

4.3.1 A Brief Introduction to Causal Inference

Presently causal inference is becoming prevalent[?], common causal inference methods include: Difference-In-Difference(DID), Synthetic Control Method(SCM), Regression Discontinuity Design(RDD) and Propsity Score Matching(PSM)[?].



(a) Microwave



(b) Hairdryer

Figure 4: The Evaluation Trend of these two Products

However, considering that the users corresponding to the three products provided are not the experimental objects, it is impossible to achieve a controlled trial[?] in the true sense, and it is impossible to construct a counterfactual framework that cares about average treatment effect[?] (i.e. ATE).

4.3.2 Propensity Score Matching based on Bootstrap

Since the sentiment score for a review has been numericalized before, if we would like to study what type of review a particular star is more likely to cause. We need to infer the causal relationship between the star rating and the review. That is, assess what comments current users are more likely to make when they see the star ratings of a product previously provided by another user on the site.

We need to focus on the Average Effect Treatment on the Treated(i.e. ATT) of the actual participants in the project. Given the large data set and the fact that users are independent and non-recurring, that is, the processing effect is only linked to the star rating, independent of the given emotional score, it conforms to the overlap assumption, so causal inference is made using the propensity score matching based on bootstrap (using an existing sample resampling method to infer). Here are the steps:

- Randomize the data
- Multinomial Logistic Regression and Maximum Likelihood Estimation(MLE) of propensity scores. The propensity score of the individual i is the conditional probability of entering the treatment group given $\boldsymbol{X}_i$

$$\mathbb{P}\left(\boldsymbol{X}_i\right) = \mathbb{P}\left(t_i = j \mid \boldsymbol{X}_i\right) = \frac{\exp\left(X_i'\beta_j\right)}{1 + \sum_{i=1}^{k} \exp\left(X_i'\beta_i\right)} \tag{11}$$

Where the likelihood function and log-likelihood function of individual i are respectively

$$L_i\left(\beta_1, \cdots, \beta_k\right) = \prod_{j=1}^{k} \left[\mathbb{P}\left(t_i = j \mid X_i\right)\right]^{\mathbf{1}(t_i=j)} \tag{12}$$

$$\ln L_i\left(\beta_1, \cdots, \beta_k\right) = \sum_{j=1}^{k} \mathbf{1}\left(t_i = j\right) \times \ln \mathbb{P}\left(t_i = j \mid X_i\right) \tag{13}$$

where $\mathbf{1}(\cdot)$ is the indicative function, if the expression in parentheses holds, the value is 1, otherwise 0. Maximize the log-likelihood function to get the propensity score

- Via k-nearest neighbor matching algorithm for one-to-one (default) matching with put back (because the control group accounts for a small proportion)
- After the matching is completed, the estimate of average processing effect ATT of the actual participants and the standardized Bias of the matching variable $\boldsymbol{X}$ are calculated by bootstrap with a set number of 500:

$$\widehat{ATT} = \frac{1}{N_1} \sum_{i \in (t_i=1)} \left(Y_i - \hat{Y}_{0i}\right) \tag{14}$$

$$bias\left(X\right) = \frac{100 \frac{1}{N_1} \sum_{i \in (t_i=1)} \left(Y_i - \hat{Y}_{0i}\right)}{\sqrt{\frac{\mathrm{Var}_{i \in (t_i=1)}(X_i) + \mathrm{Var}_{j \in (t_i=0)}(X_j)}{2}}} \tag{15}$$

Where N_1 is the number of individuals processing group, the smaller the value of the standard deviation, the better the model matching effect can be considered. It is generally believed that as long as the absolute value of the standard deviation is less than 10% will not cause the match failure

4.3.3 Process and via PSM based on bootstrap to data

After analyzing the characteristics of the data, we found that the hair dryer, pacifier, microwave oven three products have a multiple sub-model, since normally in a purchase decision, the reference object for the star is usually based on the same model, but the amount of buyers for some specific models of are too few to analyze.

Therefore, respectively, the hairdryer, pacifier, microwave oven under the three largest purchase of sub-models, a total of nine sample sets, extract its information, and the average number of previous users of the star as a processing effect, that is, to assess the effect of the project. Take the covariate variables such as whether to buy, the total number of comments and the number of words in each comment into account to perform propensity score matching.

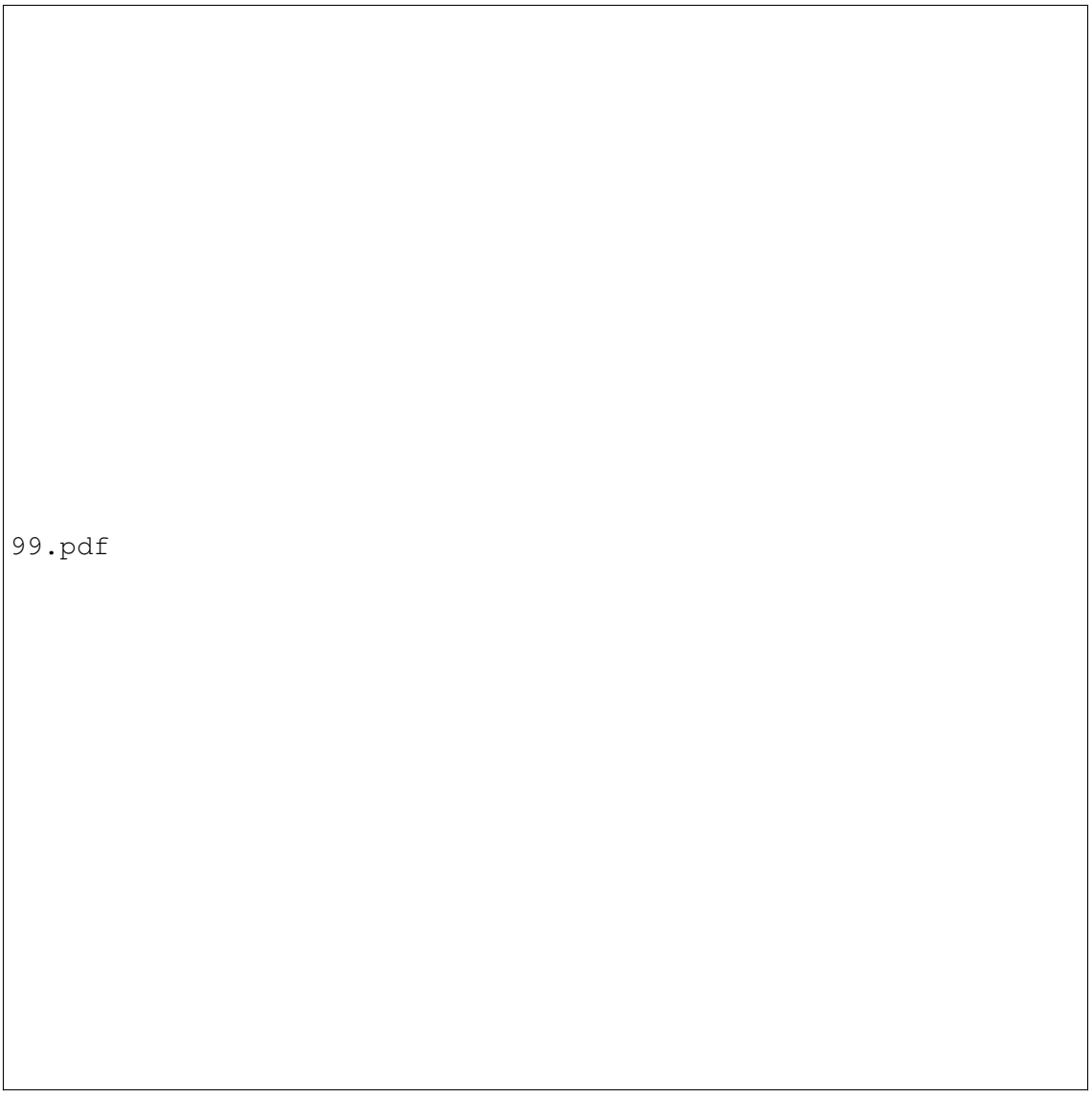


Figure 5: Common Value Range of 9 Sub-Models Propensity Scores

Because the propensity score matching method firstly meets the overlap assumption, i.e. there is overlap between the two subsamples of the processing and control groups, and it is also a prerequisite for matching, which also called the matching assumption. It can be seen intuitively from the nine subplots of Figure **??** that most of the observations are within the common value range (on support), and that each group loses only a small sample when the propensity score matches.

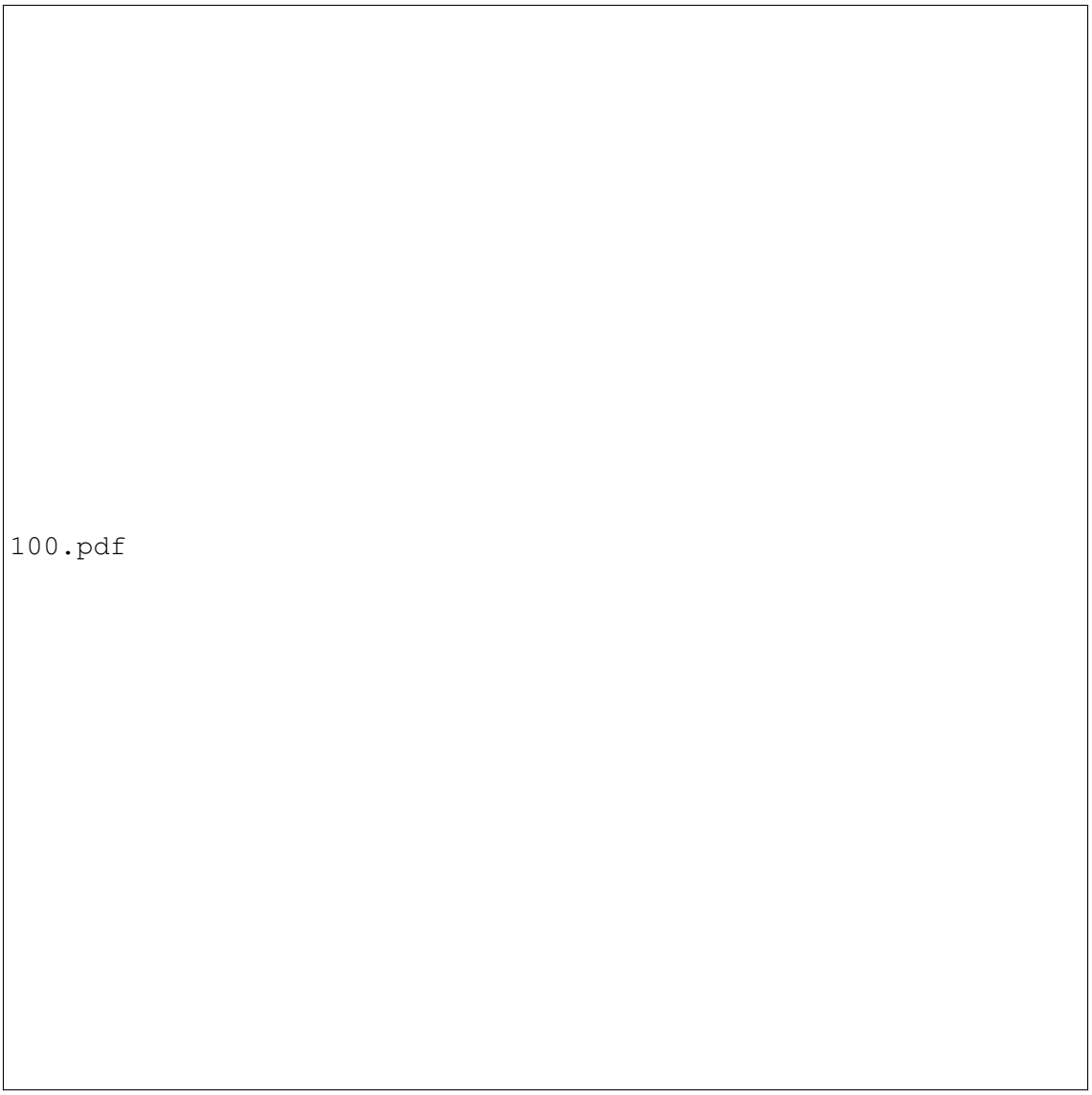


Figure 6: Probability Density of 9 Models Propensity Scores Before and After Matching

Based on the Logistic model, in order to demonstrate the effect of propensity score matching, Figure **??** reportes the probability density of the propensity score values of users who accepted the high-star or low-star processing effects before and after matching. It can be seen that there is a significant difference in the propensity scores between the pre-matching treatment group and the control group, which is reflected on the x axis. The closeness of the probability distribution of the two groups after matching has been well improved, which indicates that the matching results are satisfying.

Table **??** shows the ATT estimates, standard error and the statistic of its corresponding T-test based on bootstrap propensity score matching method. Due to space constraints, we did not list whether to buy, the total number of comments, and the number of words in each comment, showing only the standardized bias of the most important star rating variable before and after the match, but also showed the p-value of the null hypothesis:

Table 3: The ATT Estimated Results based on Bootstrap Propensity Score Matching

	ATT	SE	t	Unmatched Bias	Matched Bias	p-value of null
$Micro_1$	0.034	0.052	0.65	9.60%	-2.5	0.23
$Micro_2$	0.552	0.23	2.4	-12.30%	6.40%	0.853
$Micro_3$	0.048	0.017	2.82	-21.20%	-8.70%	0.641
$Hair_1$	0.002	0.051	0.04	3.80%	-3%	0.553
$Hair_2$	0.17	0.056	3.1	7.80%	2.30%	0.472
$Hair_3$	0.257	0.052	4.94	38%	0.40%	0.944
Pac_1	0.512	0.056	9.2	26.40%	-4.10%	0.385
Pac_2	0.426	0.065	6.52	21.80%	-0.20%	0.969
Pac_3	0.257	0.031	8.09	-9.40%	3.20%	0.147

There is no systematic difference between the treatment group and the control group.

We can analyze the conclusion from two aspects: Firstly as can be seen from Table **??**, the absolute value of the standardization error of the star rating variable has significantly decreased after the matching process of bootstrap, both are less than 10%. And each of the group can not reject the hypothesis that there is no systematic difference between the processing group and the control group at the significance level of 10%, which proves the necessity and effectiveness of matching.

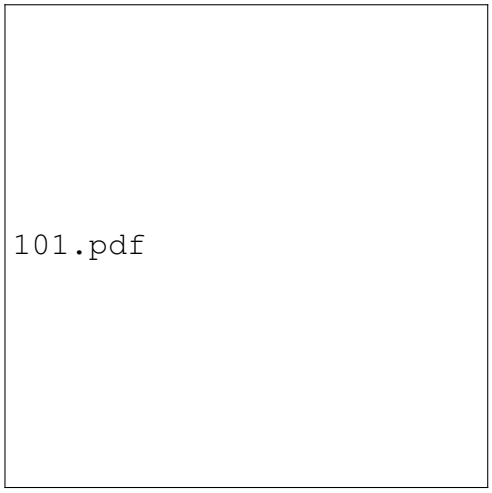
Secondly, in addition to the significant level of less than 5% for the estimate of the corresponding ATT treatment effect in the first group of microwave ovens and hairdryers, the other 7 groups have apparent positive effects, which represents that the high-star rating processing effect can significantly improve the positive emotional score of user reviews. Taking the first group of pacifier products as an example, the processing effect of high star rating can lead to an increase of 55.2% compared to the control group, while the low-star rating project evaluation effect results in the user's emotional score is 55.2% lower than that of the high-star group, and the other groups reached similar and robust conclusions.

Taken together, high star ratings can cause more significant, the highest 55.2%, the lowest 0.2% of positive comments emotional score, low star rating converses, so when the user meets a series of low-star rating reviews, the score decreases significantly. The specific decline depends on the product, its corresponding model and other factors.

4.4 Relationship between Rating and Specific Quality Descriptors

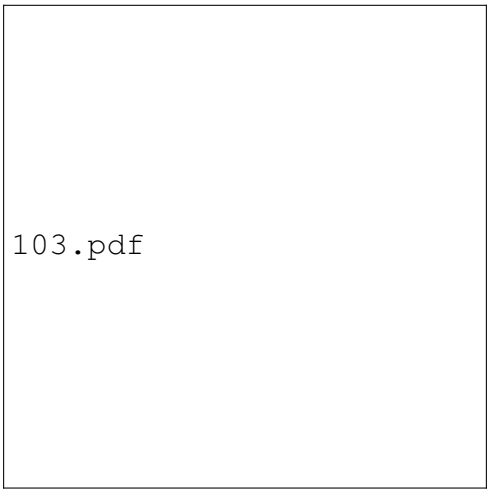
To study the relationship between specific quality descriptors and stars in reviews, the first thing we need to do is to extract a sufficient number of quality description words and filter the seven most frequently occurring words from the previously trained dictionary, such as the high-frequency words in microwave.tsv: great, perfect, love, well, bad, disappointed, stop. In order to observe the correlation between these words and star ratings, we use a violin plot that can show the distribution status and probability density of multiple sets of data. This type of plot combines the features of box and density plots. It

is mainly used to display the distribution of data, the effect of which is better than the box plot on the density level. Since our data capacity is several thousand, the scatter plot is not applicable. In this case, the violin plot can play a better role in solving the visualization problem. We filtered out the seven most frequently occurring words in each file of microwave.tsv, hair_dryer.tsv, and pacifier.tsv, and make the violin plots. The results are as Figure **??** shows: It can be seen from the figure that the positive quality description words



(a) Hairdryer

(b) Microwave



(c) Pacifier

Figure 7: Violin Plots of three Products

such as great, perfect, love, well, best and so on are all concentrated in the high-star part while the median part of negative quality descriptions such as poor, bad, disappointed, stop is mainly concentrated in the low-star part. In summary, the relationship between star ratings and quality descriptors is that the higher the star rating, the more positive the descriptive words, otherwise the closer the negative descriptive words are.

5 Sensitivity Analysis

When defining the help degree H, we set it as the product of the useful vote rate and the useful power of votes, that is

$$H = \begin{cases} \frac{V_h}{V_t} \times \sqrt[n]{V_h} = \frac{V_h^{\frac{n+1}{n}}}{V_t} & V_t \neq 0 \text{ and } 0 < V_h \leq 100 \\ \sqrt[n]{100} = 4.64159 & V_t \neq 0 \text{ and } V_h > 100 \\ 1 & V_t = 0 \end{cases} \tag{16}$$

Previously for the sake of prevent the action of E and P will be diluted by H, we set $n = 3$. This is a purely subjective behavior, if the value of n is artificially set, will it affect the help degree multiplier A and thus act on the value of weight coefficient α_1, α_2? Will it further result in a combination of text-based measures and rating-based measures being influenced only by ratings or comments, i.e. one of R or C has no effect on evaluation measure M? In this regrad, we change the value of n to observe the change of α_1, α_2. Taking the hair dryer as an example, when n ranges 1,2,3,4,5, α_1 gradually rises, while α_2

Figure 8: Parameter Analysis

gradually decreases, and totally has a gentle trend. The values of α_1 and α_2 are above 0.1 throughout the process, and there is no case where too little weight causes the value of R or C to multiply the weight by a value that is almost equal to 0. Consequently both R and C are effective while evaluating measure M, i.e. the model is effective. The results of other analysis with microwave or pacifier are the same as the hairdryer, so the model is robust.

6 Strengths and Weaknesses

6.1 Strengths

- Via five machine learning methods to analyze the emotion of the review and selected the most accurate linear support vector machine. The qualitative relationship between the review and rating was constructed and the dictionary matching of natural language processing was used to obtain a comprehensive score for each comment, i.e. quantitative relationship. Two emotional analysis methods combined with qualitative and quantitative aspects. The results are extremely reliable
- Different from the traditional indicator linear weighting model, the use of proportional multipliers for GHEP models is constructed with G as the core, which can better characterize the effect of important factors on core factors
- PSM matches the different individuals in the sample as similarly as possible through the values of covariates, ensuring the important premise of parallel trends in causal inference
- PSM uses the tendency score as a distance function to measure the distance between individuals. As a one-dimensional variable and the value ranges from 0 to 1, even if the distance is far away, the two individuals may still tend to score equal

6.2 Weaknesses

- The H, E, P of the GHEP model must be associated with the score G to express the importance of certain comments, but their impact on G is not good to judge, so the respective multipliers are not easy to determine when modeling
- According to the empirical process research and evaluation over time to analyze product reputation trends, the problem that lack of quarterly or even monthly consideration occurs. If directly on an annual basis to average, the annual information is covered up, the average annual evaluation value is not much different, the trend line is almost flat, it is difficult to observe the product reputation changes
- The solution of this model only analyzes the product model with the highest evaluation measure of three kinds of products, however does not study the trend of evaluation of hundreds of other models. Consequently we can not make a judgment on those products that fail too much.
- PSM usually requires a large sample capacity, the larger the matching quality, where the sample capacity is only a few hundred, can not guarantee excessive accuracy matching
- The PSM requires that the treatment group and the control group have a large common value range for the propensity scores. Here, due to random sampling, the degree of the common range cannot be guaranteed, and potentially losing representative samples may appear.

- PSM can only control the effect of measurable variables, and if there is selection on unobservable(such as age, gender and so on), there will be a "hidden bias" problem

6.3 Model Promotion

The GHEP model can be extended to other models which measure the impact of usefulness indicators on scoring indicators. Meanwhile, the prospect of a product is observed from the time dimension. Customer evaluation metrics can be used to observe figure characteristics based on time patterns and determine future trends based on trends that have just happened.

7 Strategy Recommendation and Conclusions

Theme: Inform Online Market Strategy and Determine Functional Design of Product

Dear Marketing Director,

The study of customer evaluation of three new products in the online market is critical for your company to understand the market in which you are involved, the motivation for participation, and the choice of product design functions, because there are significant differences in the evaluation of sub-products of the three categories, namely hairdryers, microwave and pacifier, and the overall evaluation of these three categories is different. The impact on potential sales cannot be grasped without exploring the effect of text-based measures and rating measures on the final measure of the product. Our team constructed the relationship between reviews and ratings through linear support vector machines and natural language processing, accurately depicting the calculation method of review metrics in the GHEP model, and accurately identifying the categories of products with the highest customer evaluation metrics. The trend curve of time and evaluation measures is characterized and predicted the reputational changes and possible future development of the three major products and their sub-products. To get a fuller picture of the market, we continued to model the impact of past star ratings on current reviews and the relationship between specific quality descriptors and rating levels.
Results:

The most highly rated sub-products in the three categories of hair dryer, microwave oven and pacifier were 591023894, 771401205 and 246038397. Comparing the product_title of the top five sub-products with the highest reviews, it is concluded that the characteristics of these hair dryers are conair, sedu, revolution, professional, ionic, micro, turbo, fold, etc., Microwave ovens are characterized by counter, slim, fry, grilling, ceramic enamel, countertop, whirlpool, sharp, over-the-range, etc., and with Samsung, the characteristics of nipples are free, infant, booster, etc., and mostly in wubbanub. In the study of product review changes in different years, it was found that hair dryers and microwave ovens fluctuated significantly in the early evaluation, only small fluctuations in the later period and the overall evaluation curve tilted upward, indicating that the reputation of these two types of goods in the market is gradually rising. The three most highly rated product types as an example to draw the relationship between the evaluation measure M and date t, the image shows that the evaluation measures of these products since the first sale

in 2012 is a slow decline trend, is a potential failure of the product. At the same time, our study found that past ratings have an impact on current reviews because customers are vulnerable to the influence of those in front of them; text-based reviews with positive words such as perfect are associated with high-star ratings, while low-quality descriptors such as Disappointed are closely associated with low-star ratings.

Strategy:

- The overall evaluation curve for these products in hair dryers and microwave ovens is upward, indicating that their reputation is growing and that these two new products can be selected as the product categories that the company enters the market to sell.
- After selecting the product category, should choose a specific product type, each product type of the current evaluation measure is high, does not mean that its future can be sustained, for example, the most highly rated several products after our research found that may be a potential failure of the product. "So it is important to study its historical trends, that is, the relationship between time and evaluation, and to select products that fit the curve upward, which is likely to be a potentially successful product."
- In product design, to learn from Samsung, wubbanub and so on, such as nipples should meet the characteristics of free, infant, booster, etc. within a reasonable range, for hair dryers and microwave ovens should also be similar to the selection of excellent product features as a functional design target.
- In addition, if you can collect more point evaluation information, can use the relationship between fitting analysis time and evaluation on a quarterly basis, it will help your company to choose different categories of products and different product types in different seasons, to achieve product production interaction.
- Because past ratings will have an impact on the current reviews, so the company should as far as possible to maintain the star rating of their products, otherwise, once the bad reviews, follow-up reviews will also become poor reviews, will always have an impact on the company's product sales, it is difficult to recover.
- Text-based comments such as perfect with positive emotions and high-star contact. Therefore, in order to make the company's product evaluation measures increase, the company can properly guide online shopping customers civilized speech, attitude is positive, but also to ensure the improvement of their own services, to give customers a good mood.

Sicerely yours,
Your friends

References

[1] Sebastián Maldonado,Julio López,Angel Jimenez-Molina,Hernán Lira. Simultaneous feature selection and heterogeneity control for SVM classification: An application to mental workload assessment[J]. Expert Systems With Applications,2020,143.

[2] Basemah Alshemali,Jugal Kalita. Improving the Reliability of Deep Neural Networks in NLP: A Review[J]. Knowledge-Based Systems,2020,191.

[3] The Effect of On-Line Consumer Reviews on Consumer Purchasing Intention: The Moderating Role of Involvement[J] . Do-Hyung Park,Jumin Lee,Ingoo Han. International Journal of Electronic Commerce . 2007 (4)

[4] Alberto Dellacasa Bellingegni,Emanuele Gruppioni,Giorgio Colazzo,Angelo Davalli,Rinaldo Sacchetti,Eugenio Guglielmelli,Loredana Zollo. NLR, MLP, SVM, and LDA: a comparative analysis on EMG data from people with trans-radial amputation[J]. Journal of NeuroEngineering and Rehabilitation,2017,14(1).

[5] A survey of trust and reputation systems for online service provision[J] . Decision Support Systems . 2005 (2)

[6] Designing Novel Review Ranking Systems:Predicting the Usefulness and Impact of Reviews. Ghose A,Ipeirotis P G. Proceedings of the ninth international conference on Electronic commerce . 2007

[7] Bernstein Bezier methods for the Computer Aided Geometric Design of free-form curves and surfaces. Fordon W.J, Riesenfeld R.F. Journal of the ACM . 1974

[8] Approximating surfaces by moving total least squares method. R. Scitovski,S. Ungar, D. Jukic. Journal of Applied Mathematics . 1998

[9] Xiaogang W. Communist Cadres and Market Opportunities: Entry into Self-employment in China, 1978–1996[J]. Social Forces(1):1.

[10] Mccaffrey DF, Griffin BA, Almirall D, et al. A Tutorial on Propensity Score Estimation for Multiple Treatments Using Generalized Boosted Models[J]. Statistics in Medicine, 2013, 32(19):3388-414.

[11] Wyss R, Girman CJ, Locasale RJ, et al. Variable selection for propensity score models when estimating treatment effects on multiple outcomes: a simulation study[J]. Pharmacoepidemiology and Drug Safety, 2013, 22(1):77-85.

[12] Feng P, Zhou X H, Zou Q M, et al. Generalized propensity score for estimating the average treatment effect of multiple treatments[J]. Statistics in Medicine, 2012, 31(7):681-697.

Appendices

Appendix A Code for Part 3.1

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import nltk
from nltk import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

#Import Data
df=pd.read_csv('pacifier1.csv')
#df=df.drop([950,951,952,953],axis=0)
#df=df.iloc[:16000,]
#Eliminate missing and unreasonable data due to misalignment
df1=df['review_date'].isnull()
#df1=df.copy()
print(df1.shape[0])
queshi_list=[]

for i in range(df.shape[0]):
if df1[i] == True:
queshi_list.append(i)
df.drop(index=queshi_list,inplace=True)
df=df[df['star_rating'].isin(['1','2','3','4','5'])]
label=df['star_rating']
label=list(label)
train_data=df.loc[:,['review_body','star_rating']]
#print(train_data['star_rating'][0])

#The stars are distributed 0-1 according to whether they are greater than three
# marking greater than three is positive, otherwise it is negative
for i in range(len(label)):
if int(train_data.loc[i,'star_rating'])>3:
train_data.loc[i, 'star_rating']='1'
else:
train_data.loc[i, 'star_rating'] = '0'

# Process comments through nltk's thesaurus
# including word segmentation, removing English symbols, removing stop words,
# and obtaining stem words
characters = [',', '.','<','br','>','/', ':', ';', '?', '(', ')', '[', ']', '&', '!',
 '*', '@', '#', '$', '%','-','...','^','{','}']
def dropchar(text):
# Participle
texts_tokenized = [[word for word in nltk.word_tokenize(text)]]
token_words = word_tokenize(text)
# Remove stop words
stop_words = stopwords.words('english')
fileter_words = [word for word in token_words if word not in stop_words
```

```
and word not in characters or word ==’not’]
# stemmer
stemmer = PorterStemmer()
fileterStem_words = [stemmer.stem(word) for word in fileter_words]

return ’ ’.join(fileterStem_words)  # Returns a string separated by spaces

train_data[’review_body’] = train_data[’review_body’].apply(dropchar)

#Normalize comments with regular expressions
import re
def preprocessor(text):
text = re.sub(’<[^>]*>’,’’, text)
emoticons = re.findall(’(?::|;|=)(?:-)?(?:\)|\(|D|P)’, text)
text = (re.sub(’[\W]+’, ’ ’, text.lower()) +
’ ’.join(emoticons).replace(’-’, ’’))
return text

def tokenizer(text):
return text.split()

print(train_data[’review_body’][0])
train_data[’review_body’] = train_data[’review_body’].apply(preprocessor)
#train_data[’review_body’] = train_data[’review_body’].apply(tokenizer)
#print(train_data[’review_body’][0])

#  word stemming
def preprocessing(text):
# Participle
texts_tokenized = [[word for word in nltk.word_tokenize(text)]]
token_words = word_tokenize(text)
# Remove stop words
stop_words = stopwords.words(’english’)
fileter_words=[word for word in token_words if word not in stop_words or word == ’not’]
fileter_words=[word for word in fileter_words if word.isalpha()]
# stemmer
stemmer = PorterStemmer()
fileterStem_words = [stemmer.stem(word) for word in fileter_words]
return ’ ’.join(fileterStem_words) # Returns a string separated by spaces

def preprocessing1(text):
# Participle
texts_tokenized = [[word for word in nltk.word_tokenize(text)]]
token_words = word_tokenize(text)
# Remove stop words
stop_words = stopwords.words(’english’)
fileter_words=[word for word in token_words if word not in stop_words or word == ’not’]
fileter_words=[word for word in fileter_words if word.isalpha()]
# stemmer
stemmer = PorterStemmer()
fileterStem_words = [stemmer.stem(word) for word in fileter_words]
return fileterStem_words #Returns one word separated by comma

train_data[’review_body’]= train_data[’review_body’].apply(preprocessing)
```

```
train1_data=train_data['review_body'].apply(preprocessing1)
train2_data=train_data.copy()
train2_data['review_body']=train2_data['review_body'].apply(preprocessing1)

print(train1_data[0])
# Split into two lists of good_text and bad_text
# save positive star reviews and negative star reviews, respectively
good_text=[]
bad_text=[]
pos_data=train2_data[train2_data['star_rating'].isin(['1'])]
neg_data=train2_data[train2_data['star_rating'].isin(['0'])]

for i in range(len(label)):
if int(train_data['star_rating'][i])==1:
good_text.extend(train1_data[i])
else:
bad_text.extend(train1_data[i])
print(good_text)
def bag_of_words(words):
return dict([(word,True) for word in words])

from nltk.probability import FreqDist,ConditionalFreqDist
from nltk.metrics import BigramAssocMeasures
# Count the frequency of words in positive text and the frequency of words in negative
# text and obtain features (words) with higher information content
# (the first number) (Chi-square statistics)
def jieba_feature(number):
word_fd=FreqDist()   # Count the frequency of all words
con_word_fd=ConditionalFreqDist()
# Count word frequency in positive text and word frequency in negative text
for word in good_text:
word_fd[word]+=1
con_word_fd['pos'][word]+=1
for word in bad_text:
word_fd[word]+=1
con_word_fd['neg'][word]+=1
pos_word_count=con_word_fd['pos'].N()     # Number of positive words
neg_word_count=con_word_fd['neg'].N()    # Number of negative words
# The amount of information in a word is equal to the positive chi-square statistic plus
# the negative chi-square statistic
total_word_count=pos_word_count+neg_word_count
word_scores={}
for word,freq in word_fd.items():
pos_score=BigramAssocMeasures.chi_sq(con_word_fd['pos'][word],(freq,
pos_word_count),total_word_count)
neg_score=BigramAssocMeasures.chi_sq(con_word_fd['neg'][word],(freq,
neg_word_count),total_word_count)
word_scores[word]=pos_score+neg_score
best_vals = sorted(word_scores.items(), key=lambda item: item[1],
reverse=True)[:number]
best_words = set([w for w, s in best_vals])
return dict([(word, True) for word in best_words])

# Process the data into a data form that can be processed
```

```
# by the machine learning library that comes with nltk
def build_features():
#feature = bag_of_words(text())
#feature = bigram(text(),score_fn=BigramAssocMeasures.chi_sq,n=900)
# feature =  bigram_words(text(),score_fn=BigramAssocMeasures.chi_sq,n=900)
feature = jieba_feature(1000)
posFeatures = []
for words in pos_data['review_body']:
a = {}
for word in words:
if word in feature.keys():
a[word] = 'True'
posWords = [a, 'pos']  # Give "pos" to positive text
posFeatures.append(posWords)
negFeatures = []
for words in neg_data['review_body']:
a = {}
for word in words:
if word in feature.keys():
a[word] = 'True'
negWords = [a, 'neg']  # Give "neg" to negative text
negFeatures.append(negWords)
return posFeatures, negFeatures

posFeatures,negFeatures=build_features()
#Shuffle data order, split training and test sets
from random import shuffle
shuffle(posFeatures)    #Randomize the arrangement of text
shuffle(negFeatures)
train=posFeatures[:]+negFeatures[:]
test=posFeatures[:]+negFeatures[:]
data,tag=zip(*test)
#Separate data and tags from test sets for easy testing
#Machine learning algorithm verification (find the best machine learning algorithm)
from nltk.classify.scikitlearn import SklearnClassifier
def score(classifier):
classifier=SklearnClassifier(classifier)
classifier.train(train)
pred=classifier.classify_many(data)
n=0
s=len(pred)
for i in range(0,s):
if(pred[i]==tag[i]):
n=n+1
return n/s
#Through experiments, compare the prediction accuracy scores to get the best
#classification algorithm
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC, NuSVC
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
print('LogisticRegression's accuracy is  %f' % score(BernoulliNB()))
```

Appendix B Code for Part 3.3

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df=pd.read_csv('pacifier1.csv')
#Remove missing rows
df1=df['review_body'].isnull()
print(df1.shape[0])
queshi_list=[]
for i in range(df.shape[0]):
if df1[i] == True:
queshi_list.append(i)

df.drop(index=queshi_list,inplace=True)
df=df[df['verified_purchase'].isin(['y','Y','N','n'])]
label=df['star_rating']
label=list(label)
train_data=df.loc[:,['product_parent','review_body','star_rating']]

characters = [',', '.','<','br','>','/', ':', ';', '?', '(', ')'
, '[', ']', '&', '!', '*', '@', '#', '$', '%','-','...','^','{','}']

#for i in range(len(label)):
#train_data['review_body'][i] = [word for word in train_data['review_body
#'][i] if word not in characters]

print(train_data.head(1))
print(train_data.shape)

import nltk
from nltk import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

def word_num(text):
# Participle
texts_tokenized = [[word for word in nltk.word_tokenize(text)]]
token_words = word_tokenize(text)
fileter_words = [word for word in token_words if word  not in characters ]
return len(fileter_words)  # Returns one word separated by    c   o   m   m   a
train1_data=train_data['review_body'].apply(word_num)

def preprocessing(text):
# Participle
texts_tokenized = [[word for word in nltk.word_tokenize(text)]]
token_words = word_tokenize(text)
# Remove stop words
stop_words = stopwords.words('english')
fileter_words = [word for word in token_words if word not in stop_words
and word not in characters or word =='not']
# stemmer
stemmer = PorterStemmer()
```

```
fileterStem_words = [stemmer.stem(word) for word in fileter_words]

return ' '.join(fileterStem_words)  # Returns a string separated by spaces

train_data['review_body'] = train_data['review_body'].apply(preprocessing)
#print(train_data['review_body'][0])
from nltk.sentiment.vader import SentimentIntensityAnalyzer as SAI

sai=SAI()
result=[]


for word in train_data['review_body']:
pol_score=sai.polarity_scores(word)
pol_score['review_body']=word
result.append(pol_score)

result_data=pd.DataFrame(result)

G=result_data.loc[:,['compound','pos']]
G=np.array(G,dtype=float)

from sklearn import preprocessing
std=preprocessing.MinMaxScaler().fit(G)
G_std=std.transform(G)
G_value=[]
for i in range(len(label)):
G_value.append(G_std[i][0])

E_value=[]
P_value=[]
H_value=[]
A_value=[]
C_value=[]
for i in range(len(label)):
if df['verified_purchase'][i] == 'Y' or df['verified_purchase'][i] == 'y':
E_value.append(pow(G_value[i],2)*4)
else:
E_value.append(1)
if df['vine'][i] == 'y' or df['vine'][i] == 'Y':
P_value.append(pow(G_value[i],2)*4)
else:
P_value.append(1)
if df['helpful_votes'][i] == 0:
H_value.append(1)
elif df['total_votes'][i] != 0 and df['helpful_votes'][i] < 100:
H_value.append(pow(df['helpful_votes'][i],1.2)/df['total_votes'][i])
else: #df['total_votes'][i] != 0 and df['helpful_votes'][i] >=100:
H_value.append(pow(100,1/5))
all_value=np.array(G_value,dtype=float)
print(len(H_value),len(P_value),H_value[510])
for i in range(len(label)):
if G_value[i] < 0.4 and df['total_votes'][i] != 0 and df['helpful_votes'][i] != 0:
A_value.append(1/H_value[i])
```

```
elif G_value[i] > 0.6 and df['total_votes'][i] != 0 and df['helpful_votes'][i] != 0:
A_value.append(H_value[i])
else:
A_value.append(1)
for i in range(len(label)):
C_value.append(A_value[i]*E_value[i]*P_value[i]*G_value[i])
R_value=list(df['star_rating'])
test_value=np.dstack((R_value,C_value))
test_value=test_value[0]
print(test_value)
std=preprocessing.MinMaxScaler().fit(test_value)
test_std=std.transform(test_value)

test_data=pd.DataFrame(test_std,columns=['R_value','C_value'])
R_std=list(test_data['R_value'])
C_std=list(test_data['C_value'])
#test_data.to_csv('test_value.csv',index=None)
#print(test_data)
from sklearn import preprocessing
std=preprocessing.MinMaxScaler(feature_range=(0.002,0.996)).fit(test_data)
test_data=std.transform(test_data)
test_data=pd.DataFrame(test_data)
m=test_data.iloc[: , 0].size#Dataset rows
n=test_data.columns.size#Dataset columns
P=test_data.copy()
SUM=list(test_data.apply(sum))
for i in range(m):
for j in range(n):
P.iloc[i,j]=P.iloc[i,j]/SUM[j]
#The proportion of the i-th sample under the j-th index
import math
k=1/math.log(m)
e=[]#Store entropy
for j in range(n):
sum1=0
for i in range(m):
t=P.iloc[i,j]*math.log(P.iloc[i,j])
sum1+=t
e.append(-k * sum1)
e = [1 - i for i in e]
#Information entropy redundancy
w=[]
for j in range(n):
w.append(e[j]/sum(e))
#Weight of each indicator
print(w)
M_value=[]#Comprehensive score of each indicator
for i in range(m):
sum2=0
for j in range(n):
t=w[j]*test_data.iloc[i,j]
sum2+=t
M_value.append(sum2)
result_list=np.dstack((G_value,E_value,P_value,H_value,A_value,
```

```
C_value,R_std,C_std,M_value))
result_list=result_list[0]
result_list=pd.DataFrame(result_list,columns=['G_value','E_value',
'P_value','H_value','A_value','C_value','R_std','C_std','M_value'])
#result_list.to_csv('pacifier_result.csv',index=None)
```

Appendix C Code for Part 4.4

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import nltk
from nltk import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import seaborn as sns

#Import Data
#df=pd.read_csv('hair_dryer.tsv',sep='\t')
df=pd.read_csv('pacifier1.csv')
#Eliminate missing and unreasonable data due to misalignment
df1=df['review_date'].isnull()
print(df1.shape[0])
queshi_list=[]

for i in range(df.shape[0]):
if df1[i] == True:
queshi_list.append(i)
df.drop(index=queshi_list,inplace=True)
df=df[df['star_rating'].isin(['1','2','3','4','5'])]
label=df['star_rating']
label=list(label)
train_data=df.loc[:,['review_body','star_rating']]
#print(train_data['star_rating'][0])

# The stars are distributed 0-1 according to whether they are greater than three
# marking greater than three is positive, otherwise it is negative
for i in range(len(label)):
if int(train_data.loc[i,'star_rating'])>3:
train_data.loc[i, 'star_rating']='1'
else:
train_data.loc[i, 'star_rating'] = '0'

# Process comments through nltk's thesaurus
# including word segmentation, removing English symbols,
#removing stop words, and obtaining stem words
characters = [',', '.','<','br','>','/', ':', ';', '?', '(', ')',
'[', ']', '&', '!', '*', '@', '#', '$', '%','-','...','^','{','}']
def dropchar(text):
# Participle
texts_tokenized = [[word for word in nltk.word_tokenize(text)]]
token_words = word_tokenize(text)
# Remove stop words
```

```
stop_words = stopwords.words('english')
fileter_words = [word for word in token_words if word not in stop_words
and word not in characters or word =='not']
# stemmer
stemmer = PorterStemmer()
fileterStem_words = [stemmer.stem(word) for word in fileter_words]

return ' '.join(fileterStem_words)  # Returns a string separated by spaces

train_data['review_body'] = train_data['review_body'].apply(dropchar)

#Normalize comments with regular expressions
import re
def preprocessor(text):
text = re.sub('<[^>]*>','', text)
emoticons = re.findall('(?::|;|=)(?:-)?(?:\)|\(|D|P)', text)
text = (re.sub('[\W]+', ' ', text.lower()) +
' '.join(emoticons).replace('-', ''))
return text

def tokenizer(text):
return text.split()

print(train_data['review_body'][0])
train_data['review_body'] = train_data['review_body'].apply(preprocessor)
#train_data['review_body'] = train_data['review_body'].apply(tokenizer)
#print(train_data['review_body'][0])

# word stemming
def preprocessing(text):
# Participle
texts_tokenized = [[word for word in nltk.word_tokenize(text)]]
token_words = word_tokenize(text)
# Remove stop words
stop_words = stopwords.words('english')
fileter_words=[word for word in token_words if word not in stop_words or word == 'not']
fileter_words=[word for word in fileter_words if word.isalpha()]
# stemmer
stemmer = PorterStemmer()
fileterStem_words = [stemmer.stem(word) for word in fileter_words]
return ' '.join(fileterStem_words) # Returns a string separated by spaces

def preprocessing1(text):
# Participle
texts_tokenized = [[word for word in nltk.word_tokenize(text)]]
token_words = word_tokenize(text)
# Remove stop words
stop_words = stopwords.words('english')
fileter_words=[word for word in token_words if word not in stop_words or word == 'not']
fileter_words=[word for word in fileter_words if word.isalpha()]
# stemmer
stemmer = PorterStemmer()
fileterStem_words = [stemmer.stem(word) for word in fileter_words]
return fileterStem_words #Returns one word separated by comma
```

```
train_data['review_body']= train_data['review_body'].apply(preprocessing)
train1_data=train_data['review_body'].apply(preprocessing1)
train2_data=train_data.copy()
train2_data['review_body']=train2_data['review_body'].apply(preprocessing1)

print(train1_data[0])
# Split into two lists of good_text and bad_text
# save positive star reviews and negative star reviews, respectively
good_text=[]
bad_text=[]
pos_data=train2_data[train2_data['star_rating'].isin(['1'])]
neg_data=train2_data[train2_data['star_rating'].isin(['0'])]

for i in range(len(label)):
if int(train_data['star_rating'][i])==1:
good_text.extend(train1_data[i])
else:
bad_text.extend(train1_data[i])
print(good_text)
def bag_of_words(words):
return dict([(word,True) for word in words])

from nltk.probability import FreqDist,ConditionalFreqDist
from nltk.metrics import BigramAssocMeasures
# Count the frequency of words in positive text and the frequency of words in negative # te
# (the first number) (Chi-square statistics)
def jieba_feature(number):
word_fd=FreqDist()   # Count the frequency of all words
con_word_fd=ConditionalFreqDist()
# Count word frequency in positive text and word frequency in negative text
for word in good_text:
word_fd[word]+=1
con_word_fd['pos'][word]+=1
for word in bad_text:
word_fd[word]+=1
con_word_fd['neg'][word]+=1
pos_word_count=con_word_fd['pos'].N()    # Number of positive words
neg_word_count=con_word_fd['neg'].N()    # Number of negative words
# The amount of information in a word is equal to the positive chi-square statistic plus
# the negative chi-square statistic
total_word_count=pos_word_count+neg_word_count
word_scores={}
for word,freq in word_fd.items():
pos_score=BigramAssocMeasures.chi_sq(con_word_fd['pos'][word],(freq,
pos_word_count),total_word_count)
neg_score=BigramAssocMeasures.chi_sq(con_word_fd['neg'][word],(freq,
neg_word_count),total_word_count)
word_scores[word]=pos_score+neg_score
best_vals = sorted(word_scores.items(), key=lambda item: item[1],
reverse=True)[:number]
best_words = set([w for w, s in best_vals])
return dict([(word, True) for word in best_words])
```

```
ten_word=jieba_feature(30)
print(ten_word)
word_list=['great','perfect','love','best','poor','disappoint','useless']
great_star=[]
perfect_star=[]
love_star=[]
well_star=[]
bad_star=[]
poor_star=[]
stop_star=[]
words=[]
for i in range(len(label)):
for word in train2_data['review_body'][i]:
if word == word_list[0]:
great_star.append(label[i])
elif word==word_list[1]:
perfect_star.append(label[i])
elif word==word_list[2]:
love_star.append(label[i])
elif word==word_list[3]:
well_star.append(label[i])
elif word==word_list[4]:
bad_star.append(label[i])
elif word==word_list[5]:
poor_star.append(label[i])
elif word ==word_list[6]:
stop_star.append(label[i])
total_num=len(great_star)+len(perfect_star)+len(love_star)+len(well_star)+len(bad_star)
+len(poor_star)+len(stop_star)
for i in range(total_num):
if i<len(great_star):
words.append('great')
elif i<len(great_star)+len(perfect_star):
words.append('perfect')
elif i<len(great_star)+len(perfect_star)+len(love_star):
words.append('love')
elif i<len(great_star)+len(perfect_star)+len(love_star)+len(well_star):
words.append('best')
elif i<total_num-len(poor_star)-len(stop_star):
words.append('poor')
elif i<total_num-len(stop_star):
words.append('disappoint')
else:
words.append('useless')
star_rating=great_star+perfect_star+love_star+well_star+bad_star+poor_star+stop_star
#print(len(great_star),len(perfect_star),len(love_star),len(well_star),len(bad_star),
len(poor_star),len(stop_star))
star_rating=list(map(int, star_rating))
words=list(map(str,words ))
result=np.dstack((words,star_rating))
result=result[0]
result_data=pd.DataFrame(result,columns=['words','star_rating'])
print(result_data.head())
result_data.to_csv('result_data.csv',index=None)
```

```
sns.violinplot( x='words',y='star_rating',data=result_data,
linewidth = 2,
width = 0.8,
palette = 'hls',
order = word_list,
scale = 'area',
gridsize = 50,
inner = 'box',
#bw = 0.8
)
#plt.show()
```