Problem Chosen	2020	Team Control Number
C	MCM/ICM Summary Sheet	2018067

Product Data Mining Model based on Emotion and Time Pattern

Summary

Firstly, in order to help the sales of three products provided by sunshine company succeed in Amazon mall, this paper uses lightGBM algorithm which is popular in machine learning to build a model, through natural language emotion analysis technology, the comment text is mapped to the numerical space. At the same time, the data after preprocessing is feature engineered to help the model analyze the data better After the model is established, the F1 score of the model is about 0.9 through model evaluation, which indicates that the model is very successful. At the same time, we get the importance of each column in the data. Finally, we find that the characteristics review_body, rate, review_date and total_votes have a great impact on the rating of a product. Therefore, the company should pay attention to the corresponding data information to successfully operate the product.

Secondly, the A question of second problem, The model established in the first problem has comprehensively considered all relevant data. Once the products of sunshine company start to sell, input the relevant data into the model to get the forecast score, and then analyze the data according to the forecast score and the real score to get the spcific conclusion. For B question, The evaluation score results of comprehensive rating and review are determined by entropy weight method to describe the reputation of products. Through the statistical data of monthly sales volume, favorable rate and poor evaluation rate, the statistical chart is drawn for the analysis based on the time pattern. It is found that the relevant information of microwave fluctuates greatly, while the spacifier is relatively stable. Since the first mock exam of two models combine the relevant text and rating measures, C question uses fuzzy comprehensive evaluation method to draw the relevant index and the score obtained from B question by using the problem model conclusion, and gets the weight of the corresponding index, calculates the comprehensive score by weight, and judges the potential success of a product. D question required to analyze whether a specific rating will lead to a certain type of comment. By calculating the Spearman correlation coefficient and actual data of the two, it is found that a specific rating will lead to a certain type of comment. Finally, by building a dictionary of specific descriptors, the word frequency of related descriptors is counted, and the results are obtained by using the thermodynamic diagram and word cloud visualization analysis, Negative words such as "never" are closely related to one star rating, while positive words such as "love", "like", "great" often appear in four or five star rating.

Finally, this paper summarizes the above data mining and analysis results into a report, which is provided to the marketing director of sunshine company by letter. I sincerely hope that this result can provide the best help for the company.

Keywords: Emotional evaluate; LightGBM; Fuzzy comprehensive evaluation; Word cloud

Contents

1 Introduction

1.1 Problem Formulation

With the continuous explosion of information technology, the amount of data is also growing explosively. How to make full use of the data to extract useful information in the era of big data is a very meaningful topic. As the largest e-commerce company in the United States, Amazon attaches great importance to the user's behavior data. The company provides consumers with the opportunity to score and comment in its online shopping mall. This paper will use these data to build a model to extract information, which will escort the listing and after-sales of three new products launched by sunshine company.

The question of this topic can be divided into two parts. The first part contains only one question, and the second part contains six questions. The specific questions are as follows.

Part I:

The data sets of hair dryer, microwave and pacifier are analyzed, and the mathematical relationships among star rating, comment and helpfulness rating are measured quantitatively to help the three new products of sunshine company succeed.

Part II:

a. Provide a method to extract information based on rating and comment data. When the three products launched are sold on the Internet, sunshine company can use this method to track product sales;

b. This paper discusses the time-based measurement and model of three products' data respectively, and analyzes the change of commodity reputation according to the change of data over time;

c. Determine a text-based measurement method and a combination of star based measurement methods to explore whether the product is a success or failure;

d. To explore whether customers have herd mentality and whether customers' emotions will be mobilized with the evaluation of others;

e. Judge whether the characteristic words of the text are closely related to the rating level;

f. Write a letter to the marketing director of sunshine company to recommend the evaluation results of our team and explain the reasons.

1.2 Model Goals

Sunshine's hair dryer, microwave and pacifier are coming soon. The goal of this model is to use data to extract information reasonably, to help sunshine company track the changes of sales and reputation of products according to product reviews and ratings, and to judge the impact of product ratings and reviews on consumers. In general, the purpose of our team modeling is to contribute to making the correct sales strategy for sunshine company.

2 Assumptions and Notations

2.1 Assumptions

To simplify the given problems and modify it more appropriate for simulating real life conditions, we make the following basic hypotheses.

- It is assumed that the comments are true and there is no praise or intentional smear;
- Suppose there is no irony in the comment;
- Suppose that comments use common words, such as good, recommended, etc;
- Assume the quality of the product remains unchanged;
- It is assumed that customers have the same views on the same kind of products, for example, users have the same views on the products under hair dryer.

2.2 Notations

Symbols	Definition
$rate$	star rating over 4 stars rate
$sales$	sales of product per month
$good$	good review rate
bad	bad review rate
M	evaluation matrix
Rs	Spearman coefficient

3 Related works

3.1 GBDT algorithm

GBDT is a machine learning model integrating multiple decision trees. GBDT's full name is gradient descent tree, and is one of the best algorithms to fit the real distribution in the traditional machine learning algorithm. It is effective, accurate and interpretable, and has achieved superior results in many machine learning tasks. Through multiple iterations, each iteration produces a weak classifier model, and each classifier is trained on the basis of the residual of the previous one. The requirements for weak classifiers are generally simple enough, with low variance and high deviation. Because the training process is to reduce the deviation and to improve the accuracy of the final classifier. However, GBDT has a fatal disadvantage -- it is very slow and needs to traverse the data countless times, resulting in its limited application.

3.2 Fuzzy comprehensive evaluation model

Fuzzy comprehensive evaluation method is widely used in fuzzy mathematics. In the evaluation of a certain transaction, such a kind of problem is often encountered. Because the evaluation transaction is determined by many factors, it is necessary to evaluate each factor. On the basis of each factor making a separate comment, how to consider all factors and make a comprehensive comment is a comprehensive evaluation problem. Fuzzy comprehensive evaluation is a very effective multi factor decision-making method to make a comprehensive evaluation of things affected by many factors. Its characteristic is that the evaluation results are not absolutely positive or negative, but represented by a fuzzy set.

4 Modeling and Analysis

Through certain data analysis and processing, the LightGBM model in machine learning is used to model the data, and the quantitative mathematical model is obtained to help Sunshine Company analyze the market, and finally achieve certain success.

4.1 Data Preprocessing

The title presents three annexes, which respectively provide data on the three categories of hair dryer, dryer and pacifier: 11470, 1615 and 18939. Also, the three data sets provided contain product user ratings and reviews extracted from the Amazon Customer Reviews Dataset thru Amazon Simple Storage Service(Amazon S3). Meanwhile, in each attachment, each row of data contains information of 15 categories of the commodity, and each category represents the specific meaning as shown in the following table1.

Table 1: The specific meaning of category

marketplace (string)	2 letter country code of the marketplace where the review was written.
customer_id (string)	Random identifier that can be used to aggregate reviews written by a single author.
review_id (string)	The unique ID of the review.
product_id (string)	The unique Product ID the review pertains to.
product_parent (string)	Random identifier that can be used to aggregate reviews for the same product.
product_title (string)	Title of the product.
product_category (string)	The major consumer category for the product.
star_rating (int)	The 1-5 star rating of the review.
helpful_votes (int)	Number of helpful votes.
total_votes (int)	Number of total votes the review received.

vine (string)	Customers are invited to become Amazon Vine Voices based on the trust that they have earned in the Amazon community for writing accurate and insightful reviews. Amazon provides Amazon Vine members with free copies of products that have been submitted to the program by vendors. Amazon doesn't influence the opinions of Amazon Vine members, nor do they modify or edit reviews.
verified_purchase (string)	A "Y" indicates Amazon verified that the person writing the review purchased the product at Amazon and didn't receive the product at a deep discount.
review_headline (string)	The title of the review.
review_body (string)	The review text.
review_date (bigint)	The date the review was written.

For a large number of commodity evaluation information data, in order to facilitate the subsequent establishment of mathematical model to solve the problem, the data is preprocessed according to the following steps:

Step 1: After checking all the data, it is found that only four rows of data have missing values, which has little impact on the overall data volume, so the four rows of data are selected for deletion.

Step 2: After careful observation of the data, it can be found that the values of all sample data in the two categories of market place and product category are the same, indicating that these two categories are irrelevant variables in this study, so they are deleted; at the same time, in the data in the column of product_parent, except for the two outliers in the pacifier, all other data, as long as the product_id is the same, The same is true for product_parent, so only one field is reserved.

Step 3: Because product_id and product_title have the same meaning that both stand for a same product, which leads to the data redundancy, delete product title.

Step 4: Add two columns of data, year and month, according to the data of review_date.

Step 5: Text processing, such as word segmentation, stop words removal, punctuation removal, is convenient for later text analysis.

4.2 LightGBM Model

LightGBM is an efficient implementation of GBDT, which accelerates the traditional gbdt training process by more than 20 times, and achieves almost the same accuracy. It mainly includes two core parts, one is GOSS, which is an algorithm to reduce the amount of data and ensure the balance of accuracy, the other is EFB, which can change many mutually exclusive features into low-dimensional dense features, and effectively avoid unnecessary calculation of zero value features.

GOSS algorithm excludes most of the data with small gradients and only uses the

remaining data for information gain estimation. LightGBM research[1]shows that: samples with large gradients play a more important role in calculating information gain. GOSS can obtain very accurate information gain calculation through smaller data.
EFB algorithm reduces the number of features by binding mutually exclusive features together. Mutually exclusive features mean that they rarely appear in non-zero values at the same time, and lightGBM also shows that it is NP problem to find the optimal feature binding, but greedy algorithm can obtain very good approximate probability.

4.2.1 Model Establishmen and analysis

We will follow the following process to build a model and solve the first question.

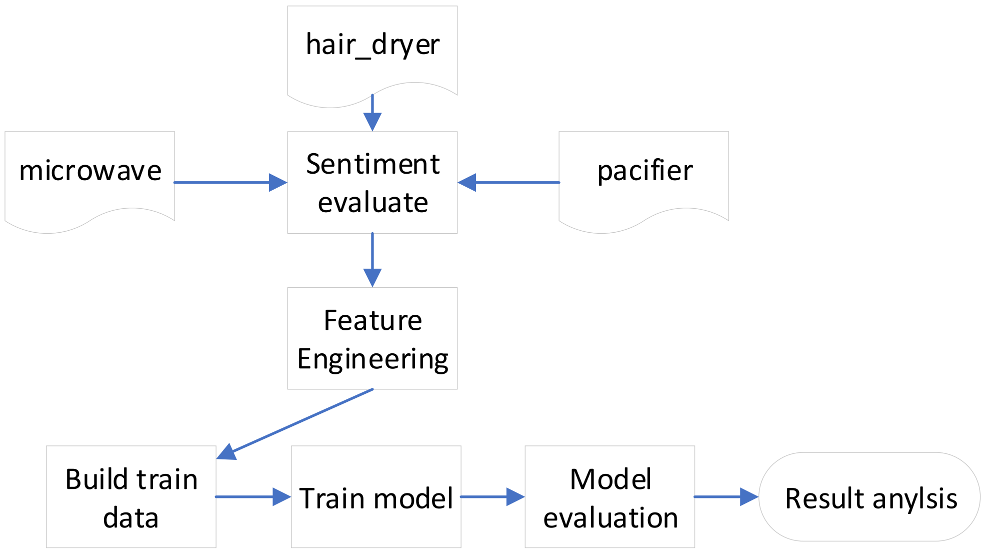


Figure 1: the process of building the model

Among them, hair_dryer, microwave and pacifier are data sets after data preprocessing. In the following part, the above process is described in detail.

4.2.2 Emotional Score

We use TextBlob model which is used in Sentiment Analysis to quantify the reviews_body. The result of TextBlob Sentiment Analysis can return a tuple --(polarity, subjectivity)

Table 2: Emotional analysis results

reviews	polarity	subjectivity
Works great!	1.000	0.750
Love this dryer!	0.625	0.600
Quiet, but does not seem like 1000 watt power⋯	0.000	0.333
Amazing addition to the nursery!	0.750	0.900
…	…	…

The polarity is a floating point number with a range of [-1.0, 1.0]. Positive Numbers mean positive and negative Numbers mean negative. Alfred is a floating point number with a range of [0.0, 1.0], where 0.0 is objective and 1.0 is subjective. We use the polarity value calculated by reviews to replace the original reviews_body data, so as to build our model.

After we use TextBlob to map the review body to the numerical space, We build

Light gradient accelerator (Light) Gradient Boosting Machine (LightGBM) which is widely used in data mining tasks to get the weights to build model. We take star_rating as the output and the other features after preprocessing above as the input to train the model.

4.2.3 Feature Engineering

In order to obtain more data correlation information, we used feature engineering to process the data set, and then put it into the model for training. Feature engineering often plays an important role in the field of data mining. Different feature engineering can obtain different feature sets. A good feature engineering can reveal more relevant information in the data set, thus making the model more accurate.

Our observation data shows that there is a big gap between some comments and ratings. For example, we only give one star for good comments and five stars for bad comments.

Table 3: Comments on inconformity of star rating and review body

Review_id	Star_rating	Review_body
R134FUK2D9TQU6	1	I have used the dryer several times and it works great. I had questions which were answered promptly by other customers which was helpful in making my decision. Definitely recommend.
R1HI3QGXJQ2RUT	5	We owned these from the store and they are exactly the same. Too bad my grandson decided he was done with pacifiers one week later
...	...	...

It is found that there are 12, 5 and 10 pieces of such data in the data set hair player, microwave and pacifier, respectively. This paper considers that the occurrence of such data is abnormal and not normal evaluation information, so this kind of data will be deleted in the post-processing.

At the same time, we make statistics according to all the star rating information of a product_id, and use the following formula to calculate the praise rate of a product _id:

$$rate = \frac{\sum (starRating >= 4)}{count_i} \quad (1)$$

Among them, $count_i$ represent `product_id` for the total number of i.

Finally, we unify all kinds of string category data in the data into continuous numerical category data starting from 0, because the next model needs this form of category data.

4.2.4 Training data construction and Model training

After all the data are processed by feature engineering, a two classifier is trained. Therefore, those with a rating of 4 stars or above are regarded as positive examples, and those with a rating of less than 4 stars are regarded as negative examples. 20% of the data set size is used as the test data to evaluate the model, and the remaining data is used as the training data training model of the model. The final data size is as table 4:

Table 4: Data volume of training set and test set

	Train data	Test data	Total
hair_dryer	9163	2291	11454
microwave	1288	322	1610
pacifier	15130	3783	18913

The hardware environment is 16g memory + Intel core-i78th, and the software platform is windows10. The related parameters of lightGBM model are as follows: num_learners = 64, learning_rate = 0.09, and then input the training data to the training model.

4.2.5 Model Evaluation

Using test data to evaluate the model, the evaluation index is precision, recall, F1 value, and the calculation formula is as follows:

$$precision = \frac{TP}{TP+FP} \tag{2}$$

$$recall = \frac{TP}{TP+FN} \tag{3}$$

$$F1 = \frac{2*P*R}{P+R} \tag{4}$$

Among them, TP indicates that the real evaluation results are positive examples and the predicted results of the model are also the total number of positive examples, FP represents that model predicts the total number of negative cases as positive cases, FN represents the total number of positive cases predicted as negative cases.

The test results are shown in Table 5.

Table 5: Test result

Product	Precision	Recall	F1
hair_dryer	0.812	0.982	0.889
microwave	0.846	0.903	0.874
pacifier	0.892	0.989	0.939

It can be seen from the above table that the F1 value of lightgbm model in three products is high, which proves that the effect of the model is very good.

4.2.6 Result Analysis

The lightGBM model has been successfully established above, and we can use the model to get the following data as table 6:

Table 6: Weight of eigenvalue

	Hair_dryer	Microwave	Pacifier
customer_id	0.000	0.000	0.000
review_id	0.000	0.000	0.000
product_id	0.036	0.000	0.009
helpful_votes	0.041	0.038	0.044
total_votes	0.138	0.122	0.139
vine	0.000	0.000	0.004
verified_purchase	0.028	0.028	0.025
review_body	0.223	0.329	0.285

review_date	0.200	0.213	0.201
year	0.003	0.000	0.000
month	0.053	0.081	0.055
rate	0.274	0.184	0.233

The values in the table are the relative weights of each feature. We can see that the weights of customer_ID and review_ID in the three data sets are all 0, indicating that they are not helpful for the market analysis of sunshine company, only exist as identifiers, and have no impact on the results. Secondly, we can see that the weights of review_body, rate, review_date and total_votes are relatively high, and to prove that they are very important for the rating of a product. Therefore, for sunshine company, we should focus on the above four aspects to achieve success.

4.3 The Solution for Task A

Once the products of sunshine company are sold, the model of the first question will be used to predict the user rating. If the prediction rating is consistent with the actual rating of the user and the rating is good, the importance of the first question will be used to analyze the success of the products, and the successful aspects will be done to the end to ensure that the product reputation does not decline. If the rating is poor, the importance will also be used to analyze the aspects of the products that are not enough, Next, we should focus on improving the product disadvantage and realizing the reversal. If it is inconsistent with the predicted rating and the actual rating of users, we should analyze whether these users are false users or malicious users, so as to deal with these users in a targeted way and ensure the normal sales market。

4.4 Task B——Entropy Weight Model

4.4.1 Model Building

For the three data sets provided, this paper uses the previous results of emotional analysis, combined with rating data, uses information entropy to determine the weight of the two, and finally gets the score of each evaluation, using the score to analyze the time pattern.

Information entropy has three properties: monotonicity, nonnegativity and accumulation：

1. Monotonicity: the higher the probability of occurrence, the lower the amount of information it carries;

2. Non negativity: information entropy can be regarded as a kind of breadth, and non negativity is a reasonable necessity;

3. Accumulation: that is, the measurement of the total uncertainty of multiple random events occurring at the same time can be expressed as the sum of the measurement of the uncertainty of each event, which is also a reflection of the breadth.

Assume that the user's rating and evaluation of goods are the sum of two events, and they are expected to be independent, the probability of their simultaneous occurrence is

$$p(X=A,Y=B)=p(X=A)\dot{p}(X=B) \tag{5}$$

According to the accumulation， we can infer that

$$H(p(X=A,Y=B)) = H(p(X=A)\cdot p(Y=B)) = H(p(X=A)) + H(p(Y=B)) \quad (6)$$

If the sum of the product function value of two variables is equal to the sum of the function value of two variables, it should be a logarithmic function. Considering that the probability is less than or equal to 1, it is less than 0 after taking the logarithm. Considering the second property of information entropy, it is necessary to add a negative sign in the front. Finally, the definition of information entropy given by Claude Shannon, the father of reference information theory, has the formula of information entropy as follows:

$$H(x) = -C\sum_{x\in X} p\,(x)\log p\,(x) \quad (7)$$

Take the data after emotional analysis as evaluation data and star rating as rating data, and finally calculate the product comprehensive score as follows.

$$score = emotionScore \cdot a + starRating \cdot b \quad (8)$$

Emotion score is the score of review_body sentiment analysis, and starrating is the rating of users. A and B calculate the information entropy weight of the two by using the information entropy formula, and calculate the weight values of the three data sets respectively, as shown in table 7:

Table 7: Information entropy weight

	a	b
hair_dryer	0.138	0.861
microwave	0.069	0.930
pacifier	0.169	0.831

Calculate the score of each product to analyze the time-based measurement mode of each data set. In this paper, the overall sales volume, favorable rate, poor rate and other data of each month are calculated by month for visual analysis.单月份整体销量 $sales$ represents overall sales volume in a single month, $good$ indicates favorable rate , bad indicates bad rate. The calculation formula is as follows:

$$sales = \sum_{date\in month_i} data \quad (9)$$

$$good = \left(\sum_{score>4} data\right)/sales \quad (10)$$

$$bad = \left(\sum_{score<1} data\right)/sales \quad (11)$$

$data$ represents a row of data,$date$ is in review_date, and$score$ is the score calculated by formula for each piece of data.

4.4.2 Results

1. Overall sales volume analysis and data selection

According to the pre-processing data, the sales figures of the three products are as follows:

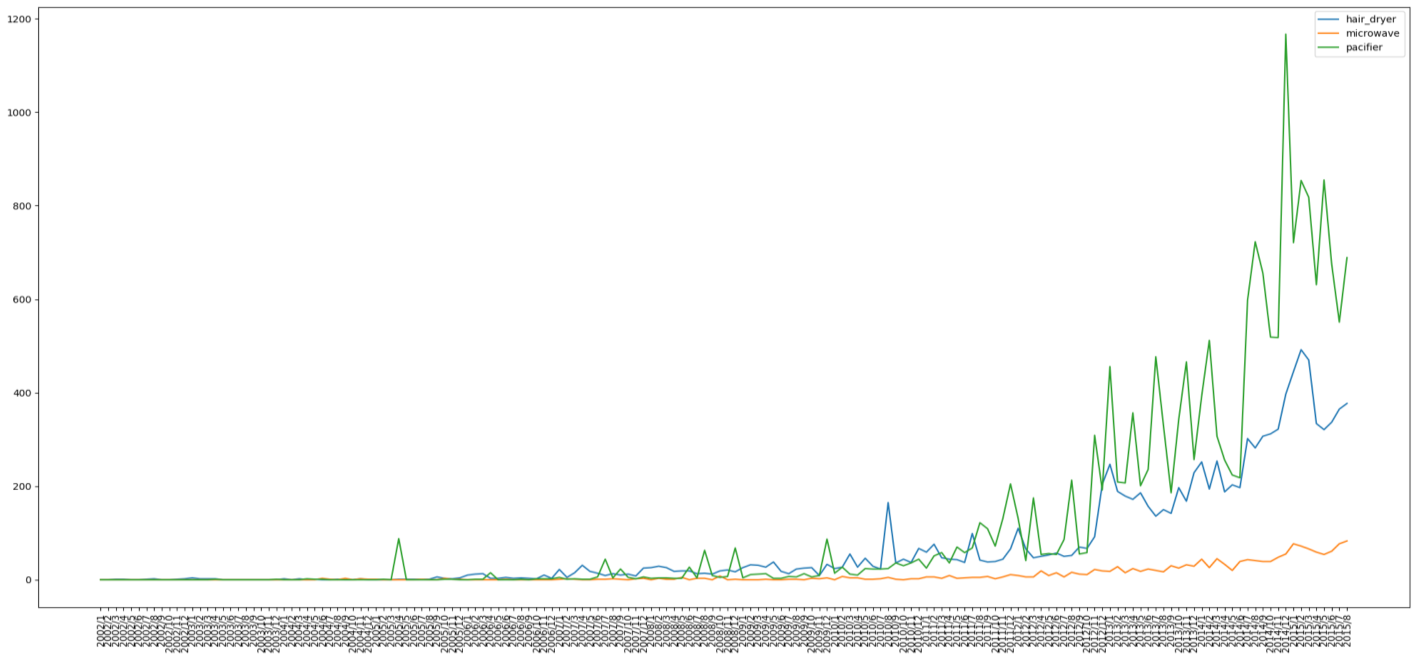


Figure 2: sales

Due to space limitation, image is blurred, and if you want to see the clear version, please refer to the appendix. Through the image, we can know that the sales volume before 2010 is very small. Therefore, the following analysis only takes the data after January 2010 for analysis. Meanwhile, it can be seen that the overall sales volume of the three products is increasing year by year. The sales volume of microwave ovens is relatively low, and the sales volume of diapers is relatively high. The reason may be that the microwave ovens are durable products, while diapers are disposable nondurable products, and the sales volume of hair dryer is in the middle. This is related to the characteristics and uses of the products.

(1)The analysis of hair_dryer's time pattern

Using the data after preprocessing, we get the monthly positive and negative rate of hair player as shown in Figure 3:

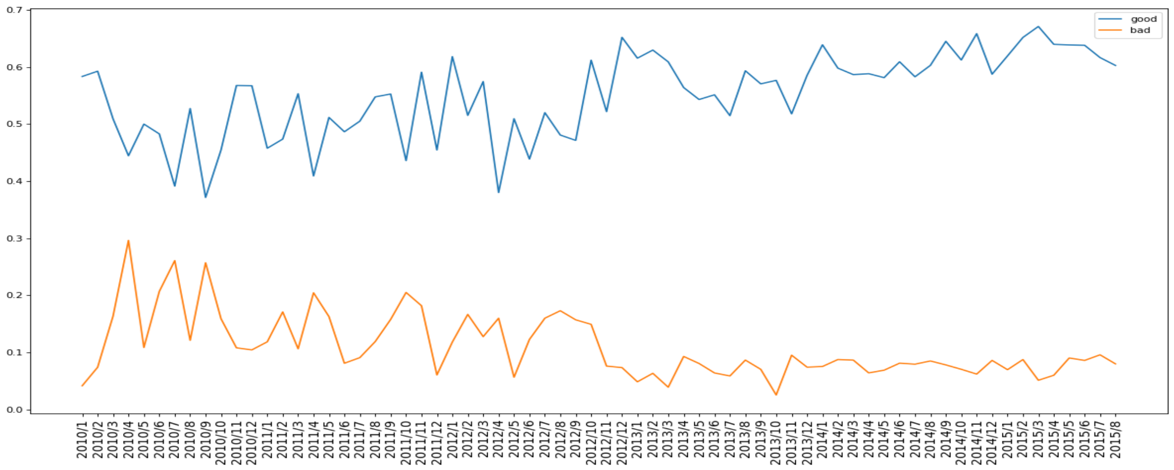


Figure 3: good and bad rates of hair_dryer

From the figure, we can analyze that since January 2010, the overall positive rate of the hair dryer has increased, and the overall negative rate has declined. And the positive rate has always been far higher than the negative rate. Before January 2013, the negative rate of the whole product is high, and the product quality is not objective, but after 2013, the positive rate of the product is basically high, It shows that the quality of hair dryer gradually rises after market baptism, and people are more comfortable to use it, but there will still be a slight fluctuation

(2) Analysis of time pattern of microwave

Using the data after preprocessing, the monthly positive and negative rates of microwave are as follows:

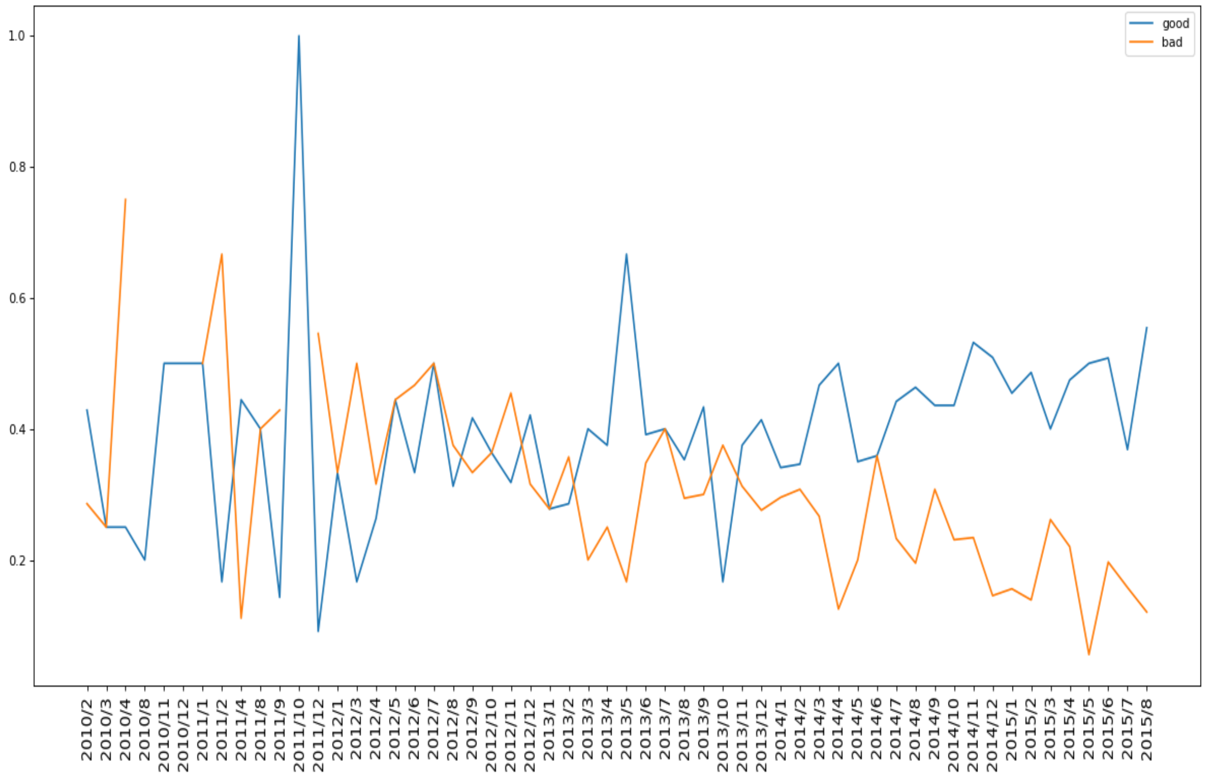


Figure 4: good and bad rates of microwave

From the figure, we can see that in the early stage when the microwave oven was put into the market, the market's poor evaluation rate of the microwave oven was slightly higher than the high evaluation rate, and the good evaluation rate of the microwave oven peaked in October 2011, and then declined; from December 2012 to October 2013, the good evaluation rate and bad evaluation rate of the microwave oven fluctuated in a certain range and were relatively stable. In November 2013 and, the good evaluation rate of the product was within a certain range It fluctuates within the range but is always greater than the rate of poor evaluation. Generally speaking, the poor evaluation rate of microwave oven is declining, and the good evaluation rate is rising.

(3) Analysis of pacifier time pattern

Using the data after preprocessing, the monthly positive and negative rates of pacifier are as figure 5:

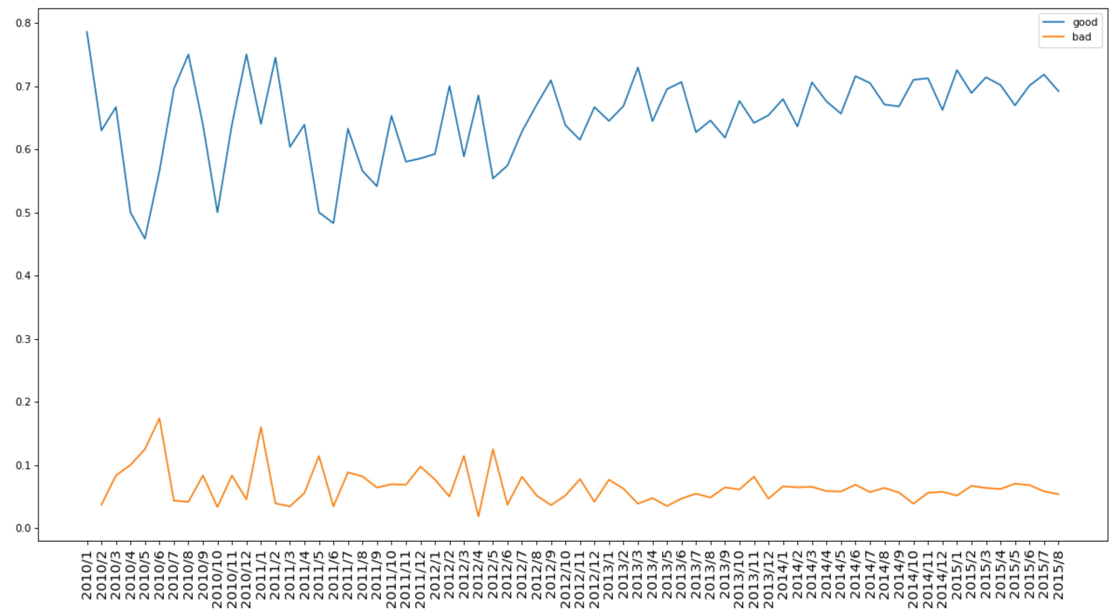


Figure 5: good and bad rates of pacifier

It can be analyzed from the figure that from the time when pacifier put the product into the market to August 2015, the product's favorable rate in the market is far higher than the negative rate, and the fluctuation frequency is getting smaller and smaller, which is in a very stable situation and a very good product.

4.5 The Solution for Task C

It is required to determine the combination of a text-based measurement method and a rating based measurement method to indicate the success or failure of a product. In the first question, the LGB model is established and the impact of the corresponding indicators on the quality of the product is obtained. In the last question, the score of the product is obtained by using the emotional score of rating and comment with the help of entropy weight method, Both of the above solutions refer to comments and ratings, i.e. text and ratings, so the combination of the two solutions best indicates potential successful or failed products.

4.5.1 Establishment and Solution of The Model

Combined with the feature weight of question 1, the features of relevant text and rating are further screened: review_body, review_date, rate, star_rating. At the same time, combined with the score of the previous question, the fuzzy comprehensive evaluation model is established to evaluate the potential success of the product. The basic steps of the fuzzy comprehensive evaluation model are as follows:

1. Select features;
2. Determine the evaluation matrix;
3. Check the evaluation matrix and calculate the weight;
4. Calculate the score and get the result.

The first step is to select features. Next, according to relevant data and data, determine the evaluation matrix M as follows:

$$M=\begin{bmatrix} 1,\ 2\ ,\ \frac{1}{2}\ ,\ \frac{3}{4}\ ,\ \frac{3}{5} \\ \frac{1}{2}\ ,\ 1\ ,\ \frac{1}{2}\ ,\ \frac{1}{2}\ ,\ \frac{1}{3} \\ 2\ ,\ 2\ ,\ 1\ ,\ 2\ ,\ \frac{2}{3} \\ \frac{4}{3}\ ,\ 2\ ,\ \frac{1}{2}\ ,\ 1\ ,\ \frac{1}{2} \\ \frac{5}{3}\ ,\ 3\ ,\ \frac{3}{2}\ ,\ 2\ ,\ 1 \end{bmatrix}$$

Through the consistency test of holding a, it is concluded that CR = 0.018 < 0.1, so the matrix passes the consistency test. Finally, the eigenvector of the matrix is calculated and normalized to obtain the weight of the corresponding features as follows table 8:

Table 8: The weight of the corresponding features

Features	Weights
review_body	0.158
review_date	0.098
rate	0.259
star_rating	0.170
score	0.315

The company can use the weight value to evaluate the success of the product. The larger the value is, the more successful the product is.

4.6 The Solution for Task D

We use the Spearman model to measure the correlation between ratings and reviews. The final result of the Spearman model is a rank correlation coefficient, which is solved according to the order of original data.

1. We sort each piece of data according to the comment time;

2. Extract the scores of star rating and review of each data as two column vectors $R(X_i)$ and $R(Y_i)$;

3. Output $R(X_i)$ and $R(Y_i)$ to the following formula:

$$d=\sum_{i=1}^{N}|R(X_i-R(Y_i)|^2 \tag{12}$$

4. Finally, the correlation RS between the two column vectors is calculated according to the following formula, and the results are shown in table 9.

$$Rs=1-\frac{6*d}{N*(N^2-1)} \tag{13}$$

Table 9:Coefficient for products

	hair_dryer	microwave	pacifier
Coefficient	0.437	0.559	0.372

It can be concluded from the table 9 that the correlation coefficient of the three products is greater than 0.05, so specific star ratings will affect the user to write some kind of comment. And we found in the data that some of the comments were indeed affected by the previous comments. And we found in reviews that some of the ratings are as shown in table 10.

Table 10: Partial comment

Review_id	Reviews
R3NQ2GXMX8FDFI	Here's a novel idea!! Read the reviews BEFORE ...
RK1L6FAK78AN4	- DO NOT BUY - the 5 star reviews are fake
ROTACUL0CWK71	I don't understand the great reviews for this dryer
R3EPB70VVPJALX	Believe the Reviews!!!
…	…

It can be seen from the table 10 that the previous rating comments and other information are mentioned in the comments, indicating that some reviews will be affected by other ratings.

4.7 The Solution for Task E

According to the specific quality descriptors of the text comments, the topic is required to be associated with the rating to find out whether there is deep-seated information. Therefore, this paper digs the specific content of the review body, extracts its related description information, and digs it by means of statistics and drawing. The specific steps are as follows:

1. Establish a specific dictionary of quality descriptors. By consulting materials and relevant data, this paper establishes a user emotion comment degree dictionary, which contains a variety of English description emotions, comment words, degree words and so on. See the appendix for the specific vocabulary;
2. Combine the data of hair dryer, microwave and pacifier, extract the content of review body. According to the classification of user rating, count the words appearing in the dictionary from review body, and get the frequency of each word through the word frequency statistical method. Arrange the results in descending order, and then take out the first 20 words. The following table 11 is obtained:

Table 11: Frequency of common words

words	1	2	3	4	5
old	272	225	353	677	2512
never	298	0	0	0	0
long	241	186	327	563	1392
new	286	0	0	0	0
good	392	368	591	1402	3058
much	228	234	370	606	2079
back	460	212	0	0	0
bad	222	0	0	0	0
high	0	0	232	0	0
however	0	0	265	0	0
need	230	157	252	566	1645

receive	222	0	0	0	0
keep	0	196	261	479	1695
cute	0	0	0	0	1479
perfect	0	0	0	0	1600
star	0	0	0	440	0
hot	0	187	0	0	0
easy	0	0	0	832	3261
dry	386	326	535	1108	3578
n't	1509	1102	1594	2445	5932
really	237	268	417	784	2209
want	255	157	274	437	0
love	0	159	359	1184	8867
great	221	237	388	1405	5797
nice	0	0	0	558	1405
small	0	165	279	549	0
like	642	602	984	1851	4186
best	0	0	0	0	1448
even	401	208	233	0	0
first	355	223	0	0	1363
still	228	0	255	506	0
set	0	163	272	607	1378
well	285	302	531	1019	2932

Use table data to draw thermodynamic diagram (Figure 6) for intuitive analysis.

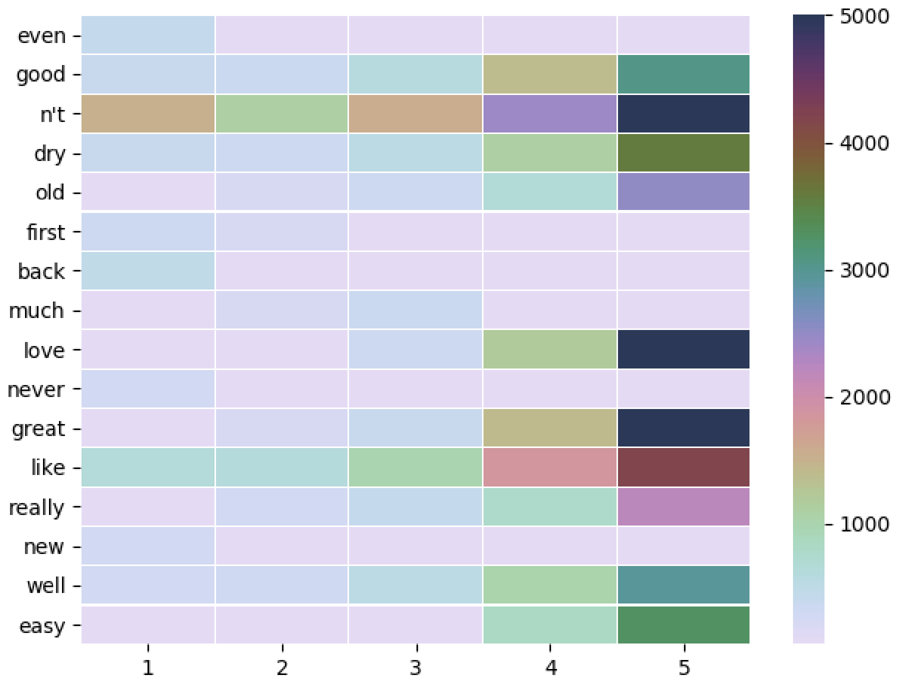


Figure 6: thermodynamic chart

In the figure, the abscissa is the star_rating, and the ordinate is the word. From the figure and table, it can be seen that "n't" is a negative word, which appears 1509 times in review_body of five stars , followed by never, like, etc.; while the word with 5 stars is love (8867 times), followed by great, like, etc, These words hardly appears in one or two stars. This shows that specific description words are closely related to rating. At the same time, we also notice that "n't" appears many times in all stars. Through specific review body data analysis, in one star, many of them appear at the same time with other positive words to express negative emotions, For example, "n't like" and so on. In five-star data, it often forms many positive emotional phrases, such as "like n't allow finger print" and so on.

Based on the above analysis, we summarize the following common star vocabulary:

Table 12: Frequency of common words

1	2	3	4	5
not like	not good	not like	like	love
never	not like	not good	well	great
back	old	much	great	easy
even	not well	old	not	like
bad	dry	however	need	well

5. Conclusion

Through data mining and in-depth analysis, we not only have a lot of perceptual knowledge about the data, but also a lot of rational knowledge. We find that many times, it is very important to investigate a product not only from the perspective of product star_rating, evaluation, review date and other data. And through the statistical analysis of the time patterns of the three products, we get the sales volume of the product, It is found that with the delay of time, pacifier has more and more favorable comments, and the favorable rate is relatively stable. Although the favorable rate of the other two products is rising, there is still a trend of fluctuation, and the development is not stable, especially the phenomenon that the negative rate is greater than the favorable rate once appeared in microave, It shows that the market of microwave can not meet people's needs. Sunshine company should pay attention to its performance, ensure product quality and pay attention to market changes, and take targeted rescue measures for different market environment to ensure the company's benefits.

Using the emotion analysis technology to analyze the emotion in the review_body, and then comparing with the rating, we found that there are five-star-poor rating and one star-high praise and other abnormal situations. At the same time, we found that whether the evaluation affects the rating and whether the rating affects the evaluation. They affect each other, and a good rating will have a certain impact on the next comment, Different comment phrases are often corresponding to different ratings. Therefore, in the actual analysis, if there are different trends between the two, it can prove that there may be some unreasonable factors in the market. At this time, the company needs to investigate and solve them, and finally make the product develop well.

6. Evaluating the Model

6.1 Advantages of the model

1. We use TextBlob to analyze the emotion of each review_body, and score them, so objectively get the rating of the product;

2. In solving problem D, we use the Spearman coefficient analysis, which requires no normal distribution, and it can get the rank coefficient, that is, it can find out the relationship between variables and variables in a certain order;

3. The model objectively uses as much data as possible, and adjusts it according to the real data. Through the visual analysis of the data, it can directly mine out a large amount of information of the data, and analyze the trend, reputation, success and other specific conditions of the three products through the information above.

6.2 Disadvantages of the model

1. Due to the time limits, when building the LightGBM model, the final result is good enough, but may not reach the best;

2. The evaluation matrix of fuzzy comprehensive evaluation method may not be objective, but it has guaranteed the objectivity through data as far as possible;

3. In terms of word frequency statistics, some phrases are not well analyzed, for example, phrases like "n't like" that express emotion through all words are actually segmented, but this is not well done, resulting in incomplete presentation.

6.3 Application of the model

1. The model can be used to evaluate the initial stage of the product and analyze whether it can be sold well;

2. The model can analyze the change of product reputation, and the company can adjust the product sales strategy in time;

3. The model can objectively evaluate the success of the product and feed back relevant information to the company in time

A letter to the marketing director

Dear Sir/Madam,

We are very honored to have this opportunity to show you and your company the results of our mining and analysis of the data of hair_dryer, microwave and pacifier. Here we will give you a comprehensive report about our research in the data.

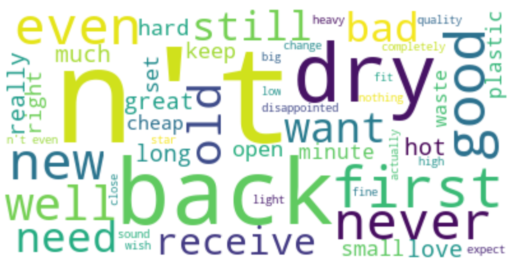
First of all, through the preliminary observations provide three data sets, we find that each product in the marketplace in the US, and product_category respectively within each data set is the same, we think it useless for analysis, so to be deleted. Similarly, found product_parent data with product_id data redundancy, then we delete the product_parent column data. At the same time, do other data preprocessing.

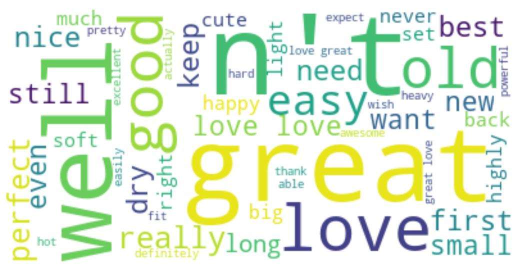
After emotional analysis in natural language processing, we will not quantitative reviews is mapped to the numerical space, using the numerical range of [1, 1] to comment contains specific emotions. The higher the number, the stronger the positive emotion. After the emotional analysis, we carried out feature engineering. A good feature engineering can make the model better grasp the data features, better discover the internal patterns in the data, and improve the accuracy of the model. Through feature engineering, a new feature rating rate was established and some abnormal conditions were removed. For example, a 1-star comment: "I have used the dryer several times and it works great. I had questions which were answered by other customers which was helpful in making my decision. Definitely recommend.". However, this should be a 5-star favorable rating, so the data with opposite star rating and rating were deleted. Fortunately, that's not a lot of data, with only 27 of the three data sets. Then, we divided the data into the training set and the test set according to the ratio of 8:2, and took the ones with a rating greater than or equal to 4 stars as the positive example, and the ones with a rating less than 4 stars as the negative example, and used the machine learning algorithm LightGBM to conduct separate data modeling for each data set. Finally, the F1 value of the model on the three data sets was 0.889, 0.874 and 0.939, which showed that our model was very successful! Not only that, we also learned from the model that review_body, review_date, rate and total_votes are the most important to the model. Your company should mainly focus on the changes of these four types of data, and try to pay little or no attention to other data with low importance such as reviewers. Once your company plans to sell the three products online, you can use the above model to evaluate the sales data of the products, and compare the predicted results of the model with the actual scores of users to get the result of whether the two are consistent. If it is consistent, it will seize the evaluation characteristics of such users and make targeted improvements to the product to achieve better sales. If it is inconsistent, you can focus on whether such users are fake or malicious users and so on.

We also use entropy weight method to obtain product scores from comprehensive analysis of rating and evaluation, which can measure product reputation and other relevant information. In order to evaluate the potential success or failure of a product, we combined the results of the established machine learning model LighGBM and entropy weight analysis, selected important indicators related to text and rating for fuzzy comprehensive evaluation modeling, and finally concluded that the potential success of a product can be indicated by calculating specific scores.

Finally, we use the correlation analysis technique to obtain that certain ratings will trigger certain types of reviews, that is, users' reviews may be influenced by previous ratings. For example, "-do NOT BUY - the 5 star reviews are fake".

At the same time, we set up a specific vocabulary list and drew the following two word cloud maps for the word frequency statistics of all comment data. The font size of the words in the word cloud map is related to how often they appear in the comments. The higher the frequency, the larger the font. We can see from the word cloud map that the one-star word cloud map is mostly negative words, such as “n't” “like”, “never”, “bad”, while the five-star word cloud map is mostly ‘great’, ‘love’, ‘well’, ‘perfect’, etc. Although there is “n't” in the word cloud image of five stars, we find that it mainly represents positive emotions through specific analysis, such as “like n't allow finger print” and so on.





One star word cloud map Five star word cloud map

Our data mining results are presented, I believe you will have a lot of harvest after you see, finally, I hope your company's products can be a great success!

References

[1] KE G, MENG Q, FINLEY T, et al. LightCBM: a highly efficient gradient boosting decision tree [C]// Proceedings of the 2017 Annual Conference on Neural Information Processing Systems. New York: Curran Associates Inc., 2017: 3146 - 3154.

[2] Zhang Jin,Mucs Daniel,Norinder Ulf,Svensson Fredrik. LightGBM: An Effective and Scalable Algorithm for Prediction of Chemical Toxicity-Application to the Tox21 and Mutagenicity Data Sets.[J]. Journal of chemical information and modeling,2019,59(10).

[3] Mudambi S. and D. Schuff "What Makes a Helpful Online Review? A Study of Customer Reviews on Amazon.com " MIS Quarterly 34(1) 2010 pp. 185-200.

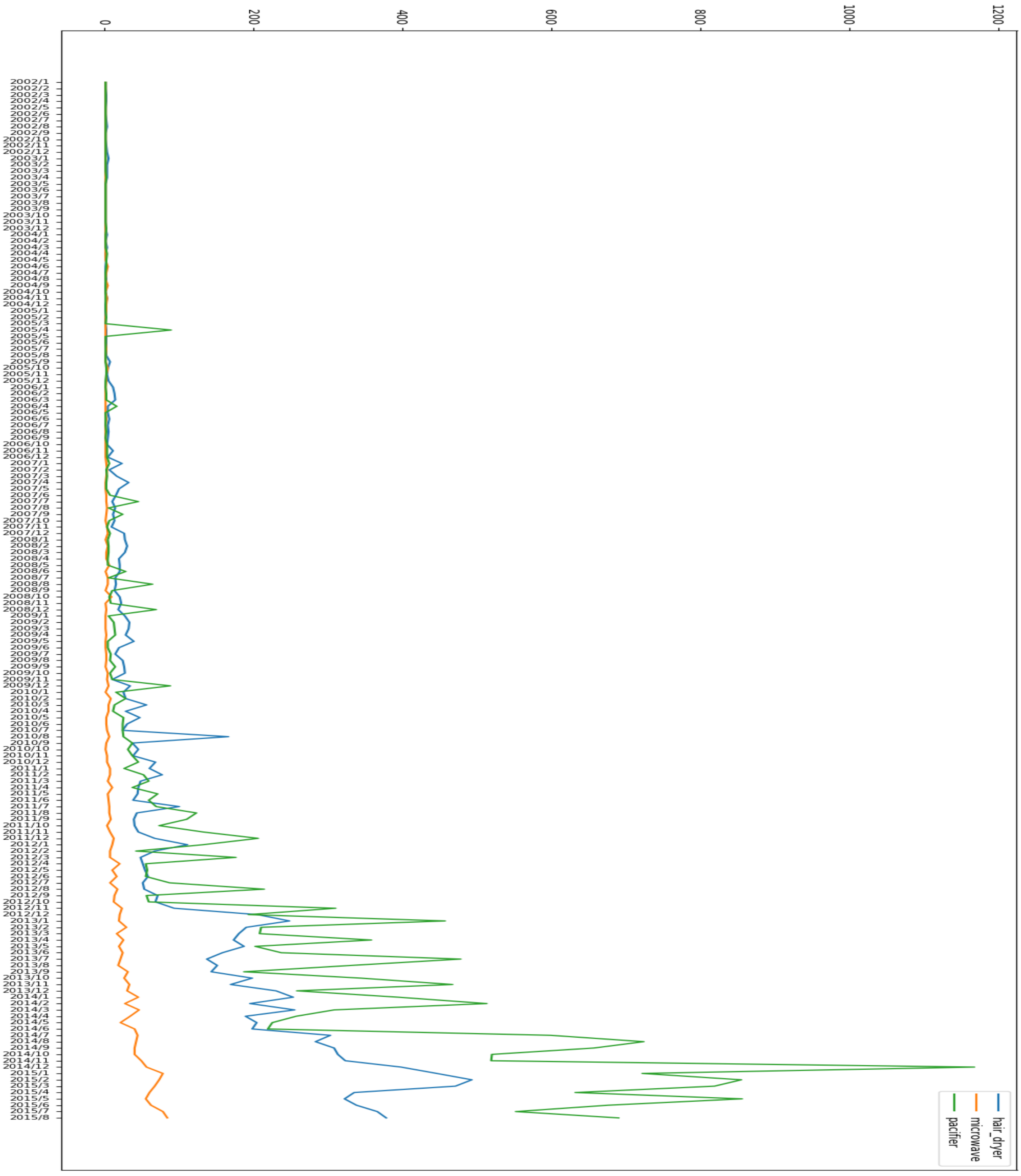
[4] Baek, H., J. Ahn, and Y. Choi. 2012. Helpfulness of online consumer reviews: Readers' objectives and review cues. International Journal of Electronic Commerce 17, no. 2: 99–126.

[5] Pang, B. and Lee, L. “A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts,” in Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, Stroudsburg, PA, 2004. Article 271.

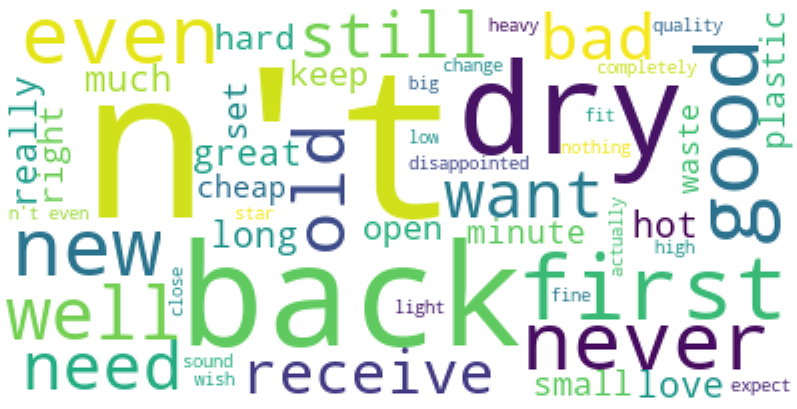
[6] Zhang, M., C. Dellarocas, and N. Awad, “The Impact of Online Movie Reviews on Box Office Performance”, In Workshop on Information Systems and Economics (WISE), College Park, MD, 2004.

Appendices

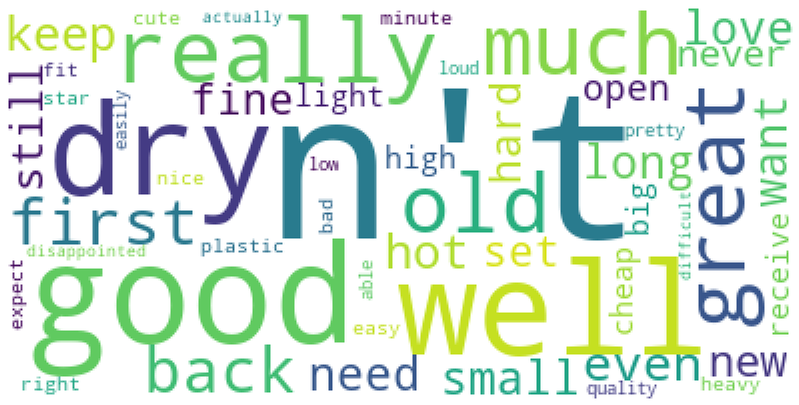
appendix 1: hair_dryer, microwave, pacifier monthly sales statistics



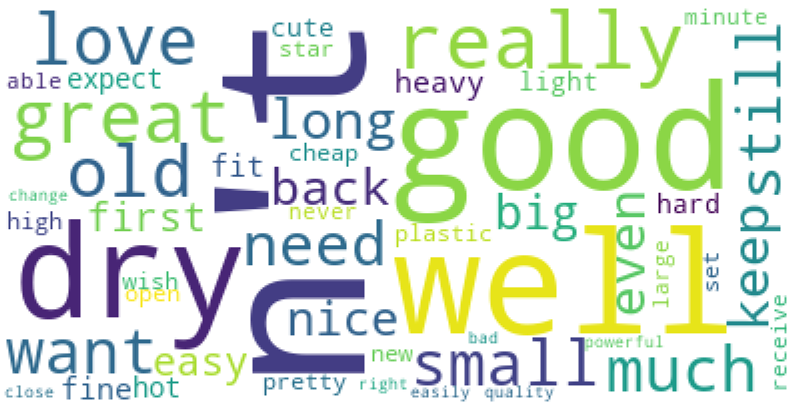
appendix 2: Star rating words cloud chart



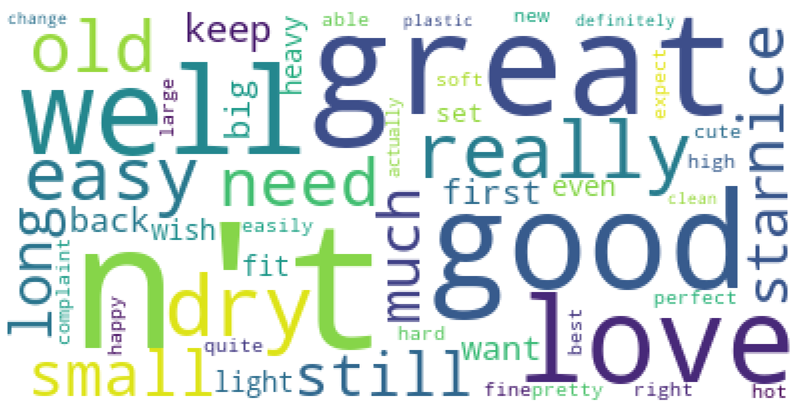
one star review words



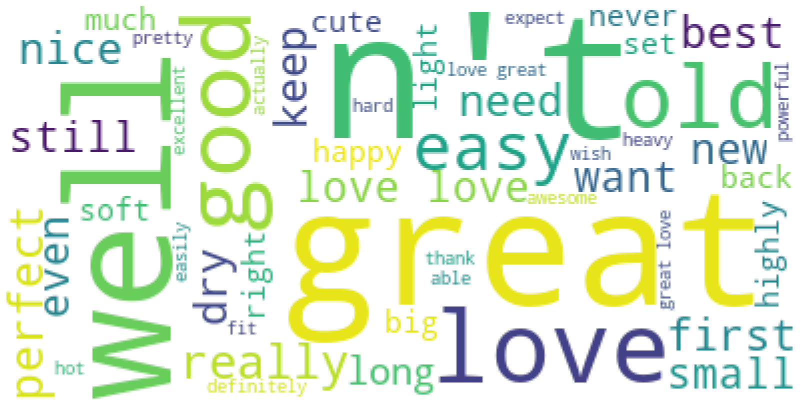
two stars review words



three stars review words



four stars review words



five stars review words

appendix 3: code

code	anylisis	remark	the code of data anylisis

```
import pandas as pd
import matplotlib.pyplot as plt
from tqdm import tqdm
from my_util import pre_process

def null_process():
        hair_dryer=pd.read_csv('../Data/hair_dryer.tsv',sep='\t',encoding='utf-8')
        microwave=pd.read_csv('../Data/microwave.tsv',sep='\t',encoding='utf-8')
        pacifier=pd.read_csv('../Data/pacifier.tsv',sep='\t',encoding='utf-8')
        # print(hair_dryer.head())

        for column in hair_dryer.columns:
                if len(set(hair_dryer[column]))==1:
                        print('hair_dryer:',column)
                if len(set(microwave[column]))==1:
                        print('microwave:',column)
                if len(set(pacifier[column]))==1:
                        print('pacifier:',column)
        print(set(pacifier['product_category']))
        print(set(pacifier['marketplace']))
        print(set(microwave['product_category']))
        print(set(microwave['marketplace']))
        #'product_category', 'marketplace'                  ,
        hair_dryer['review_date']    =    pd.to_datetime    (hair_dryer['review_date'],
format='%m/%d/%Y')
        hair_dryer['year'] = hair_dryer['review_date'].dt.year
        hair_dryer['month'] = hair_dryer['review_date'].dt.month

        pacifier['review_date']    =    pd.to_datetime    (pacifier['review_date'],
format='%m/%d/%Y')
        pacifier['year'] = pacifier['review_date'].dt.year
        pacifier['month'] = pacifier['review_date'].dt.month

        microwave['review_date']    =    pd.to_datetime    (microwave['review_date'],
format='%m/%d/%Y')
        microwave['year'] = microwave['review_date'].dt.year
        microwave['month'] = microwave['review_date'].dt.month
        del hair_dryer['product_category']
        del hair_dryer['marketplace']
        del pacifier['product_category']
        del pacifier['marketplace']
        del microwave['product_category']
        del microwave['marketplace']
        del hair_dryer['product_title']
        del pacifier['product_title']
        del microwave['product_title']
        tmp1 = pacifier[pacifier['product_id'] == 'b0042i2bwg']
        print (tmp1)
        tmp2 = pacifier[pacifier['product_id'] == 'b00db5f114']
        print (tmp2)
        dic1 = {}
        for idx in hair_dryer.index:
                i = hair_dryer.loc[idx, 'product_id']
                j = hair_dryer.loc[idx, 'product_parent']
                if i not in dic1:
                        dic1[i] = [j]
                else:
                        dic1[i].append (j)
        for i in dic1:
                if len (set (dic1[i])) != 1:
                        print ('hair_dryer')
        dic2 = {}
        for idx in microwave.index:
                i = microwave.loc[idx, 'product_id']
                j = microwave.loc[idx, 'product_parent']
                if i not in dic2:
                        dic2[i] = [j]
                else:
                        dic2[i].append (j)
        for i in dic2:
                if len (set (dic2[i])) != 1:
                        print ('microwave')
        dic3 = {}
        for idx in pacifier.index:
                i = pacifier.loc[idx, 'product_id']
                j = pacifier.loc[idx, 'product_parent']
                if i not in dic3:
                        dic3[i] = [j]
                else:
                        dic3[i].append (j)
        for i in dic3:
                if len (set (dic3[i])) != 1:
                        print (i, dic3[i])
                        print ('pacifier')
        #                    pacifier                              ,                    ,
product_id      , product_parent
        del hair_dryer['product_parent']
        del microwave['product_parent']
        del pacifier['product_parent']

        print(hair_dryer['product_id'].count()) #11470
        print(microwave['product_id'].count())#1615
        print(pacifier['product_id'].count())#18939
        hair_dryer=hair_dryer.dropna()
        microwave=microwave.dropna()
        pacifier=pacifier.dropna()
        print(hair_dryer['product_id'].count())#11468
        print(microwave['product_id'].count())#1615
        print(pacifier['product_id'].count())#18937
        #                         ,              ,
        reviewer_body = []
        for i in tqdm (hair_dryer['review_body'].values):
                sent = ''
                for j in pre_process (i):
                    sent = sent + ' ' + j
                        reviewer_body.append (sent)
                hair_dryer['review_body'] = reviewer_body

                reviewer_body = []
                for i in tqdm (microwave['review_body'].values):
                        sent = ''
                        for j in pre_process (i):
                    sent = sent + ' ' + j
                        reviewer_body.append (sent)
                microwave['review_body'] = reviewer_body

                reviewer_body = []
                for i in tqdm (pacifier['review_body'].values):
                        sent = ''
                        try:
                    for j in pre_process (i):
                        sent = sent + ' ' + j
                        except:
                    print (i)
                    sent = i
                        reviewer_body.append (sent)
                pacifier['review_body'] = reviewer_body

                hair_dryer = hair_dryer.dropna ()
                microwave = microwave.dropna ()
                pacifier = pacifier.dropna ()
                print (hair_dryer['product_id'].count ())    # 11468
                print (microwave['product_id'].count ())    # 1615
                print (pacifier['product_id'].count ())    # 18937
                hair_dryer.to_csv('../Data/hair_dryer.csv',encoding='utf-8',index=None)
                microwave.to_csv('../Data/microwave.csv',encoding='utf-8',index=None)
                pacifier.to_csv('../Data/pacifier.csv',encoding='utf-8',index=None)

                #
                # print(hair_dryer[hair_dryer.isnull().values==True])
                # print(microwave[microwave.isnull().values==True])
                # print(pacifier[pacifier.isnull().values==True])

# null_process()
hair_dryer=pd.read_csv('../Data/hair_dryer.csv',encoding='utf-8')
microwave=pd.read_csv('../Data/microwave.csv',encoding='utf-8')
pacifier=pd.read_csv('../Data/pacifier.csv',encoding='utf-8')
print(hair_dryer.columns)
```

```
def fig_star_rating_count():
    tmp1=hair_dryer.groupby(by='star_rating').count()['customer_id']
    plt.subplot(221)
    plt.bar(tmp1.index.values,tmp1.values)
    plt.ylim(0,8000)
    for a, b in zip(tmp1.index.values, tmp1.values):
        plt.text(a, b, '%.0f' % b, ha='center', va='bottom', fontsize=8)
    plt.title('hair_dryer')

    tmp2=microwave.groupby(by='star_rating').count()['customer_id']
    plt.subplot(222)
    plt.ylim(0,800)
    plt.bar(tmp2.index.values,tmp2.values)
    for a, b in zip(tmp2.index.values, tmp2.values):
        plt.text(a, b, '%.0f' % b, ha='center', va='bottom', fontsize=8)
    plt.title('microwave')

    tmp3=pacifier.groupby(by='star_rating').count()['customer_id']
    plt.subplot(212)
    plt.ylim(0,14000)
    plt.bar(tmp3.index.values,tmp3.values)
    for a, b in zip(tmp3.index.values, tmp3.values):
        plt.text(a, b, '%.0f' % b, ha='center', va='bottom', fontsize=8)
    plt.title('pacifier')
    plt.show()
# fig_star_rating_count()

def fig_time():
    # print(hair_dryer.groupby('product_id').count()['customer_id'].describe())
    #
hair_dryer1=hair_dryer[hair_dryer['review_date']>pd.to_datetime('1/1/2013',format='
%m/%d/%Y')]
    y1=hair_dryer.groupby(['year','month']).count()['customer_id']
    y2 = microwave.groupby (['year','month']).count ()['customer_id']
    y3 = pacifier.groupby (['year','month']).count ()['customer_id']
    x=[]
    for i in range(2002,2016):
        for j in range(1,13):
            x.append((i,j))
    x.pop(-1)
    x.pop(-1)
    x.pop(-1)
    x.pop(-1)
    tmp=[]
    for i in x:
        if i in list(y1.index.values):
            tmp.append(y1.loc[i])
        else:
            tmp.append(0)
    y1=tmp
    tmp = []
    for i in x:
        if i in list(y2.index.values):
            tmp.append (y2[i])
        else:
            tmp.append (0)
    y2=tmp
    tmp = []
    for i in x:
        if i in list(y3.index.values):
            tmp.append (y3[i])
        else:
            tmp.append (0)
    y3=tmp

    x=[str(item[0])+'/'+str(item[1]) for item in x]
    plt.figure(figsize=(20,10))
    plt.plot(x,y1)
    plt.plot(x,y2)
    plt.plot(x,y3)
    plt.xticks (size='small', rotation=90, fontsize=8)
    plt.legend(['hair_dryer','microwave','pacifier'],loc = 'best')
    plt.show ()
    print (tmp)
fig_time()
# test1=hair_dryer[hair_dryer['product_id']=='B003V264WW']
# print()

# print(hair_dryer.info())
# print(microwave.info())
# print(pacifier.info())
#
# print(hair_dryer.describe())
# print(microwave.describe())
# print(pacifier.describe())
# #              :                        ,                                  ,
helpful_votes/verified_purchase                ,
# print(hair_dryer[hair_dryer['verified_purchase']=='Y'].count())
```

```

print()
```

code	T1	remark	the code of question 1

```
from textblob import TextBlob
#
# text = "five stars".replace('.','')
# blob = TextBlob (text)
# #
# print ("blob        ")
# print (blob)
# print (blob.sentiment)
from sklearn.model_selection import train_test_split
import pandas as pd
import lightgbm as lgb
from tqdm import tqdm
import numpy as np
cat_cols = ['customer_id', 'review_id', 'product_id', 'vine', 'verified_purchase']
def pre_process(prod,data):
    del data['review_headline']
    dtime = pd.to_datetime (data['review_date'])
    v = (dtime.values - np.datetime64 ('2000-01-01T08:00:00Z')) / np.timedelta64 (1,
'ms')
    data['review_date'] = v

    data[cat_cols].astype('category')
    def map_value(x):
        x_set=set(x.values)
        dic={}
        n=0
        for i in x_set:
            if i not in dic:
                dic[i]=n
                n+=1
        new_x=[]
        for i in x.values:
            new_x.append(dic[i])
        return new_x
    for col in cat_cols:
        data[col] = map_value(data[col])
    def get_sentment(col):
        new_col=[]
        # for i in tqdm(col):
        #     out_put = emotion_eng.getMoodValue(i)
        #     new_col.append(out_put['all_value'])
        for i in tqdm(col):
            out_put = TextBlob (i)
```

```
                new_col.append(out_put.sentiment.polarity)
            return new_col
        data['review_body']=get_sentment(data['review_body'])

        def anylisis(data):
            not_pair = data[((data['star_rating'] == 1) & (data['review_body'] > 0.6)) |
((data['star_rating'] == 5) & (data['review_body'] < -0.6))]
            # print(not_pair)
            return not_pair.index.values

        abnormal_product = {}
        abnormal_product[prod] = (list (anylisis (data)))    # 8
        print(abnormal_product)
        # abnormal_product['microwave'] = (list (anylisis (microwave)))    # 3
        # abnormal_product['pacifier'] = (list (anylisis (pacifier)))    # 18
        data = data[~data.index.isin (abnormal_product[prod])]
        return data
hair_dryer=pd.read_csv('../Data/hair_dryer.csv',encoding='utf-8')
hair_dryer=hair_dryer.dropna()
hair_dryer=pre_process('hair_dryer',hair_dryer)
hair_dryer.to_csv('../Data/new_hair_dryer.csv')

microwave=pd.read_csv('../Data/microwave.csv',encoding='utf-8')
microwave=microwave.dropna()
microwave=pre_process('microwave',microwave)
microwave.to_csv('../Data/new_microwave.csv')

pacifier=pd.read_csv('../Data/pacifier.csv',encoding='utf-8')
pacifier=pacifier.dropna()
pacifier=pre_process('pacifier',pacifier)
pacifier.to_csv('../Data/new_pacifier.csv')

def get_X_y(prod,data):



        cols_x=['customer_id', 'review_id', 'product_id', 'helpful_votes', 'total_votes',
'vine', 'verified_purchase','review_body', 'review_date', 'year', 'month','rate']
        star=[]
        for i in data['star_rating']:
            if i<4:
                star.append(0)
            else:
                star.append(1)
        data['star_rating']=star
        X=data[cols_x]
        X[cat_cols]=X[cat_cols].astype('category')
        # X['customer_id']=X['customer_id'].astype('category')
        y=data['star_rating']

        # min_max_scaler = MinMaxScaler ()
        # X= min_max_scaler.fit_transform (X)
        # da=pd.DataFrame(X,columns=cols_x)
        return X,y
def gen_rate(data):
        tmp = data.groupby ('product_id').count ()['customer_id']
        sums = {}
        for i in tmp.index.values:
                sums[i] = tmp[i]
        rate = {}
        for i in sums:
                cnt = data[(data['product_id'] == i) & (data['star_rating'] < 4)].count ()[0]
                rate[i] = cnt / sums[i]
        rates = []
        for i in data['product_id'].values:
                rates.append (rate[i])
        data['rate'] = rates
        return data
hair_dryer=pd.read_csv('../Data/new_hair_dryer.csv',encoding='utf-8',index_col=0)
microwave=pd.read_csv('../Data/new_microwave.csv',encoding='utf-8',index_col=0)
pacifier=pd.read_csv('../Data/new_pacifier.csv',encoding='utf-8',index_col=0)
hair_dryer=gen_rate(hair_dryer)
microwave=gen_rate(microwave)
pacifier=gen_rate(pacifier)
def model1():
        X,y=get_X_y('hair_dryer',hair_dryer)
        print(len(y))
        X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2,
random_state=0,shuffle=True)
        print("Train data length:", len(X_train))
        print("Test data length:", len(X_test))
        print('              !')

        #           cv and train
        gbm = lgb.sklearn.LGBMClassifier(boosting_type='gbdt', num_leaves=64,
max_depth=-1, learning_rate=0.09, n_estimators=10, max_bin=255,
subsample_for_bin=200000, objective=None, min_split_gain=0.0,
min_child_weight=0.001, min_child_samples=20, subsample=1.0, subsample_freq=1,
colsample_bytree=1.0, reg_alpha=0.0, reg_lambda=0.0, random_state=None,
n_jobs=-1, silent=True)
        gbm.fit(X_train,y_train,sample_weight=None, init_score=None,
                        eval_set=None, eval_names=None, eval_sample_weight=None,
                        eval_class_weight=None,                        eval_init_score=None,
eval_metric=None,
                        early_stopping_rounds=None, verbose=True,
                        feature_name='auto', categorical_feature='auto', callbacks=None)
        print ('Start predicting...')
        #
        y_pred = gbm.predict (X_test)
        from sklearn.metrics import precision_score, recall_score, roc_auc_score
        print('importance:',list(zip(X_train.columns.values,gbm.feature_importances_)))
        precision=precision_score(y_test, y_pred)
        recall=recall_score(y_test, y_pred)
        print(list(zip(y_test.values,y_pred)))
        print ('              ', precision)
        print ('              ', recall)
        print ('auc       ', roc_auc_score (y_test, y_pred))
        print ('F1       ', 2 * (precision * recall) / (precision + recall))
def model2():
        X, y = get_X_y ('microwave',microwave)
        print (len (y))
        X_train,   X_test,   y_train,   y_test   =   train_test_split   (X,   y,   test_size=0.2,
random_state=0, shuffle=True)
        print ("Train data length:", len (X_train))
        print ("Test data length:", len (X_test))
        print ('              !')

        #          cv and train
        gbm   =   lgb.sklearn.LGBMClassifier   (boosting_type='gbdt',   num_leaves=64,
max_depth=-1, learning_rate=0.1,
                                        n_estimators=10,          max_bin=255,
subsample_for_bin=200000, objective=None,
                                        min_split_gain=0.0,
min_child_weight=0.001, min_child_samples=20, subsample=1.0,
                                        subsample_freq=1,
colsample_bytree=1.0, reg_alpha=0.0, reg_lambda=0.0,
                                        random_state=None,          n_jobs=-1,
silent=True)
        gbm.fit (X_train, y_train, sample_weight=None, init_score=None,
                    eval_set=None, eval_names=None, eval_sample_weight=None,
                    eval_class_weight=None, eval_init_score=None, eval_metric=None,
                    early_stopping_rounds=None, verbose=True,
                    feature_name='auto', categorical_feature='auto', callbacks=None)
        print ('Start predicting...')
        #
        y_pred = gbm.predict (X_test)
        from sklearn.metrics import precision_score, recall_score, roc_auc_score
        print ('importance:', list (zip (X_train.columns.values,
gbm.feature_importances_)))
        precision = precision_score (y_test, y_pred)
        recall = recall_score (y_test, y_pred)
        print (list (zip (y_test.values, y_pred)))
        print ('          ', precision)
        print ('          ', recall)
        print ('auc      ', roc_auc_score (y_test, y_pred))
        print ('F1      ', 2 * (precision * recall) / (precision + recall))
def model3():
        X, y = get_X_y ('pacifier',pacifier)
        print (len (y))
        X_train, X_test, y_train, y_test = train_test_split (X, y, test_size=0.2,
random_state=0, shuffle=True)
        print ("Train data length:", len (X_train))
        print ("Test data length:", len (X_test))
        print ('          !')


        gbm = lgb.sklearn.LGBMClassifier (boosting_type='gbdt', num_leaves=64,
max_depth=-1, learning_rate=0.09,
                                          n_estimators=10, max_bin=255,
subsample_for_bin=200000, objective=None,
                                          min_split_gain=0.0,
min_child_weight=0.001, min_child_samples=20, subsample=1.0,
                                          subsample_freq=1,
colsample_bytree=1.0, reg_alpha=0.0, reg_lambda=0.0,
                                          random_state=None, n_jobs=-1,
silent=True)
        gbm.fit (X_train, y_train, sample_weight=None, init_score=None,
                 eval_set=None, eval_names=None, eval_sample_weight=None,
                 eval_class_weight=None, eval_init_score=None, eval_metric=None,
                 early_stopping_rounds=None, verbose=True,
                 feature_name='auto', categorical_feature='auto', callbacks=None)
        print ('Start predicting...')
        #
        y_pred = gbm.predict (X_test)
        from sklearn.metrics import precision_score, recall_score, roc_auc_score
        print ('importance:', list (zip (X_train.columns.values,
gbm.feature_importances_)))
```

```
        precision = precision_score (y_test, y_pred)
        recall = recall_score (y_test, y_pred)
        print (list (zip (y_test.values, y_pred)))
        print ('              ', precision)
        print ('              ', recall)
        print ('auc        ', roc_auc_score (y_test, y_pred))
        print ('F1        ', 2 * (precision * recall) / (precision + recall))
# hair_dryer
model1()
#microwave
model2()
#pacifier
model3()
```

code	T2_a	remark	the code of question 2-a

```
from sklearn.decomposition import pca
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import numpy as np
import pandas as pd
import math
from textblob import TextBlob
# blob = TextBlob ("text")
# print(blob.sentiment.polarity)
# out_put = emotion_eng.getMoodValue("great")#out_put['all_value']
# all_low=hair_dryer[(hair_dryer['star_rating']<2) & (hair_dryer['review_body']<-0.6)]
# all_high=hair_dryer[(hair_dryer['star_rating']>3) & (hair_dryer['review_body']>0.2)]
# all_mid=hair_dryer[(hair_dryer['star_rating']>=2) & (hair_dryer['star_rating']<=3) & (0.2>=hair_dryer['review_body']) & (hair_dryer['review_body']>=-0.6)]
# not_pair=hair_dryer[((hair_dryer['star_rating']<2) & (hair_dryer['review_body']>0.8)) | ((microwave['star_rating']==5) & (hair_dryer['review_body']<-0.6))]
#
# a=all_low.count()['star_rating']
# b=all_high.count()['star_rating']
# c=all_mid.count()['star_rating']
# d=hair_dryer.count()['star_rating']
# e=not_pair.count()['star_rating']
# print(a,b,c,e,d-a-b-c)
# print()
# all_low=microwave[(microwave['star_rating']<2) & (microwave['review_body']<-0.6)]
# all_high=microwave[(microwave['star_rating']>3) &
(microwave['review_body']>0.2)]
# all_mid=microwave[(microwave['star_rating']>=2) & (microwave['star_rating']<=3)
& (0.2>=microwave['review_body']) & (microwave['review_body']>=-0.6)]
#                  not_pair=microwave[((microwave['star_rating']<2)               &
(microwave['review_body']>0.8))        |        ((microwave['star_rating']==5)        &
(microwave['review_body']<-0.6))]
#
# a=all_low.count()['star_rating']
# b=all_high.count()['star_rating']
# c=all_mid.count()['star_rating']
# d=microwave.count()['star_rating']
# e=not_pair.count()['star_rating']
# print(a,b,c,e,d-a-b-c)
# print()
# all_low=pacifier[(pacifier['star_rating']<2) & (pacifier['review_body']<-0.6)]
# all_high=pacifier[(pacifier['star_rating']>3) & (pacifier['review_body']>0.2)]
#   all_mid=pacifier[(pacifier['star_rating']>=2)   &   (pacifier['star_rating']<=3)   &
(0.2>=pacifier['review_body']) & (pacifier['review_body']>=-0.6)]
#
#   not_pair=pacifier[((pacifier['star_rating']<2)   &   (pacifier['review_body']>0.8))   |
((pacifier['star_rating']==5) & (pacifier['review_body']<-0.6))]
# a=all_low.count()['star_rating']
# b=all_high.count()['star_rating']
# c=all_mid.count()['star_rating']
# d=pacifier.count()['star_rating']
# e=not_pair.count()['star_rating']
# print(a,b,c,e,d-a-b-c)
hair_dryer=pd.read_csv('../Data/new_hair_dryer.csv',encoding='utf-8',index_col=0)
microwave=pd.read_csv('../Data/new_microwave.csv',encoding='utf-8',index_col=0)
pacifier=pd.read_csv('../Data/new_pacifier.csv',encoding='utf-8',index_col=0)
def anylisis(data):
    all_low=data[(data['star_rating']<2) & (data['review_body']<-0.6)]
    all_high=data[(data['star_rating']>3) & (data['review_body']>0.2)]
    all_mid=data[(data['star_rating']>=2)        &        (data['star_rating']<=3)        &
(0.2>=data['review_body']) & (data['review_body']>=-0.6)]
    # not_pair                  1       5
    not_pair=data[((data['star_rating']==1)       &       (data['review_body']>0.6))       |
((data['star_rating']==5) & (data['review_body']<-0.6))]
    a=all_low.count()['star_rating']
    b=all_high.count()['star_rating']
    c=all_mid.count()['star_rating']
    d=data.count()['star_rating']
    e=not_pair.count()['star_rating']
    # print(a,b,c,e,d-a-b-c)
    # print('        1       5                    :',e)
    return not_pair.index.values
abnormal_product={}
abnormal_product['hair_dryer']=(list(anylisis(hair_dryer)))#8
abnormal_product['microwave']=(list(anylisis(microwave)))#3
abnormal_product['pacifier']=(list(anylisis(pacifier)))#18
print(abnormal_product)
def scaler(X):
    """


    """
    min_max_scaler = MinMaxScaler ()
    x_train= min_max_scaler.fit_transform (X)
    x=pd.DataFrame(x_train,columns=X.columns.values)
    return x
def cal_weight(x):
    '''                                  '''
    #
    x = x.apply (lambda x: ((x - np.min (x)) / (np.max (x) - np.min (x))))

    #    k
    rows = x.index.size   #
    cols = x.columns.size   #
    k = 1.0 / math.log (rows)

    lnf = [[None] * cols for i in range (rows)]

    #              --
    #
    # p=array(p)
    x = np.array (x)
    lnf = [[None] * cols for i in range (rows)]
    lnf = np.array (lnf)
    for i in range (0, rows):
        for j in range (0, cols):
            if x[i][j] == 0:
                lnfij = 0.0
            else:
                p = x[i][j] / x.sum (axis=0)[j]
                lnfij = math.log (p) * p * (-k)
            lnf[i][j] = lnfij
    lnf = pd.DataFrame (lnf)
    E = lnf
    #
    d = 1 - E.sum (axis=0)
    #
    w = [[None] * 1 for i in range (cols)]
    for j in range (0, cols):
        wj = d[j] / sum (d)
        w[j] = wj
        #                                ,

    w = pd.DataFrame (w)
    return w
def get_eval(prod,data):
    data=data[~data['product_id'].isin(abnormal_product[prod])]
    x=data[['star_rating','review_body']]
    # x=scaler(x)
    w = cal_weight (x)   #        cal_weight
    w.index = x.columns
    w.columns = ['weight']

wei={'star_rating':w.loc['star_rating','weight'],'review_body':w.loc['review_body','wei
ght']}
    return wei
#   wei=get_eval('hair_dryer',hair_dryer)   #{'star_rating':   0.8529774897515476,
'review_body': 0.1470225102484523}
# print(wei)

def gen_score(prod,data):
    x=data['star_rating'].values
    y=data['review_body'].values
    wei = get_eval (prod, data)
    print(prod,wei)
    score=np.array(x)*wei['star_rating']+np.array(y)*wei['review_body']
    data['score']=score
    return data
data1=gen_score('hair_dryer',hair_dryer)
data2=gen_score('microwave',microwave)
data3=gen_score('pacifier',pacifier)
print(data1.head())
print(data2.head())
print(data3.head())
```

code	T2_b	remark	the code of question 2-b

```
from sklearn.decomposition import pca
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

```
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import classification_report

from textblob import TextBlob
# blob = TextBlob ("text")
# print(blob.sentiment.polarity)
# out_put = emotion_eng.getMoodValue("great")#out_put['all_value']
# all_low=hair_dryer[(hair_dryer['star_rating']<2) & (hair_dryer['review_body']<-
0.6)]
# all_high=hair_dryer[(hair_dryer['star_rating']>3) &
(hair_dryer['review_body']>0.2)]
# all_mid=hair_dryer[(hair_dryer['star_rating']>=2) & (hair_dryer['star_rating']<=3) &
(0.2>=hair_dryer['review_body']) & (hair_dryer['review_body']>=-0.6)]
# not_pair=hair_dryer[((hair_dryer['star_rating']<2) &
(hair_dryer['review_body']>0.8)) | ((microwave['star_rating']==5) &
(hair_dryer['review_body']<-0.6))]
#
# a=all_low.count()['star_rating']
# b=all_high.count()['star_rating']
# c=all_mid.count()['star_rating']
# d=hair_dryer.count()['star_rating']
# e=not_pair.count()['star_rating']
# print(a,b,c,e,d-a-b-c)
# print()
# all_low=microwave[(microwave['star_rating']<2) & (microwave['review_body']<-
0.6)]
# all_high=microwave[(microwave['star_rating']>3) &
(microwave['review_body']>0.2)]
# all_mid=microwave[(microwave['star_rating']>=2) & (microwave['star_rating']<=3)
& (0.2>=microwave['review_body']) & (microwave['review_body']>=-0.6)]
# not_pair=microwave[((microwave['star_rating']<2) &
(microwave['review_body']>0.8)) | ((microwave['star_rating']==5) &
(microwave['review_body']<-0.6))]
#
# a=all_low.count()['star_rating']
# b=all_high.count()['star_rating']
# c=all_mid.count()['star_rating']
# d=microwave.count()['star_rating']
# e=not_pair.count()['star_rating']
# print(a,b,c,e,d-a-b-c)
# print()
# all_low=pacifier[(pacifier['star_rating']<2) & (pacifier['review_body']<-0.6)]
# all_high=pacifier[(pacifier['star_rating']>3) & (pacifier['review_body']>0.2)]
#   all_mid=pacifier[(pacifier['star_rating']>=2)   &   (pacifier['star_rating']<=3)   &
(0.2>=pacifier['review_body']) & (pacifier['review_body']>=-0.6)]
#
#   not_pair=pacifier[((pacifier['star_rating']<2)   &   (pacifier['review_body']>0.8))   |
((pacifier['star_rating']==5) & (pacifier['review_body']<-0.6))]
# a=all_low.count()['star_rating']
# b=all_high.count()['star_rating']
# c=all_mid.count()['star_rating']
# d=pacifier.count()['star_rating']
# e=not_pair.count()['star_rating']
# print(a,b,c,e,d-a-b-c)
hair_dryer=pd.read_csv('../Data/new_hair_dryer.csv',encoding='utf-8',index_col=0)
microwave=pd.read_csv('../Data/new_microwave.csv',encoding='utf-8',index_col=0)
pacifier=pd.read_csv('../Data/new_pacifier.csv',encoding='utf-8',index_col=0)
def gen_rate(data):
        tmp = data.groupby ('product_id').count ()['customer_id']
        sums = {}
        for i in tmp.index.values:
                sums[i] = tmp[i]
        rate = {}
        for i in sums:
                cnt = data[(data['product_id'] == i) & (data['star_rating'] < 4)].count ()[0]
                rate[i] = cnt / sums[i]
        rates = []
        for i in data['product_id'].values:
                rates.append (rate[i])
        data['rate'] = rates
        return data
hair_dryer=gen_rate(hair_dryer)
microwave=gen_rate(microwave)
pacifier=gen_rate(pacifier)

def anylisis(data):
        all_low=data[(data['star_rating']<2) & (data['review_body']<-0.6)]
        all_high=data[(data['star_rating']>3) & (data['review_body']>0.2)]
        all_mid=data[(data['star_rating']>=2)        &        (data['star_rating']<=3)        &
(0.2>=data['review_body']) & (data['review_body']>=-0.6)]
        # not_pair                    1       5
        not_pair=data[((data['star_rating']==1)       &       (data['review_body']>0.6))       |
((data['star_rating']==5) & (data['review_body']<-0.6))]
    a=all_low.count()['star_rating']
    b=all_high.count()['star_rating']
    c=all_mid.count()['star_rating']
    d=data.count()['star_rating']
    e=not_pair.count()['star_rating']
    # print(a,b,c,e,d-a-b-c)
    # print('           1       5                     :',e)
    return not_pair.index.values
abnormal_product={}
abnormal_product['hair_dryer']=(list(anylisis(hair_dryer)))#8
abnormal_product['microwave']=(list(anylisis(microwave)))#3
abnormal_product['pacifier']=(list(anylisis(pacifier)))#18
print(abnormal_product)
def scaler(X):
    """


    """
    min_max_scaler = MinMaxScaler ()
    x_train= min_max_scaler.fit_transform (X)
    x=pd.DataFrame(x_train,columns=X.columns.values)
    return x
def cal_weight(x):
    '''                                '''
    #
    x = x.apply (lambda x: ((x - np.min (x)) / (np.max (x) - np.min (x))))

    #     k
    rows = x.index.size    #
    cols = x.columns.size    #
    k = 1.0 / math.log (rows)

    lnf = [[None] * cols for i in range (rows)]

    #              --
    #
    # p=array(p)
    x = np.array (x)
    lnf = [[None] * cols for i in range (rows)]
    lnf = np.array (lnf)
    for i in range (0, rows):
        for j in range (0, cols):
            if x[i][j] == 0:
                lnfij = 0.0
            else:
                p = x[i][j] / x.sum (axis=0)[j]
                lnfij = math.log (p) * p * (-k)
            lnf[i][j] = lnfij
    lnf = pd.DataFrame (lnf)
    E = lnf

    #
    d = 1 - E.sum (axis=0)
    #
    w = [[None] * 1 for i in range (cols)]
    for j in range (0, cols):
        wj = d[j] / sum (d)
        w[j] = wj
        #                                     ,

    w = pd.DataFrame (w)
    return w
def get_eval(prod,data):

    data=data[~data['product_id'].isin(abnormal_product[prod])]
    x=data[['star_rating','review_body']]
    # x=scaler(x)
    w = cal_weight (x)    #          cal_weight
    w.index = x.columns
    w.columns = ['weight']

wei={'star_rating':w.loc['star_rating','weight'],'review_body':w.loc['review_body','wei
ght']}
    return wei
#     wei=get_eval('hair_dryer',hair_dryer)     #{'star_rating':     0.8529774897515476,
'review_body': 0.1470225102484523}
# print(wei)
def gen_score(prod,data):
    x=data['star_rating'].values
    y=data['review_body'].values
    wei = get_eval (prod, data)
    score=np.array(x)*wei['star_rating']+np.array(y)*wei['review_body']
    data['score']=score
    return data

def fig(prod, D):
    data=gen_score(prod,D)[['review_date','year','month','score']]
    # print(data.describe())
```

```
	good=data[(data['score']>4)                                          &
(data['year']>2009)].groupby(['year','month']).count()['score']
	bad=data[(data['score']<1)                                           &
(data['year']>2009)].groupby(['year','month']).count()['score']
	all_of=data[(data['year']>2009)].groupby(['year','month']).count()['review_date']
	good=pd.DataFrame(good,index=good.index.values)
	bad=pd.DataFrame(bad,index=bad.index.values)
	all_of=pd.DataFrame(all_of,index=all_of.index.values)
	bad.rename(columns={'score':'score_bad'},inplace=True)
	x=[str(i[0])+'/'+str(i[1]) for i in good.index.values]
	good['time']=x
	x=[str(i[0])+'/'+str(i[1]) for i in bad.index.values]
	bad['time']=x
	x=[str(i[0])+'/'+str(i[1]) for i in all_of.index.values]
	all_of['time']=x
	all=pd.merge(good,bad,how='left')
	all=pd.merge(all,all_of,how='left')
	all.fillna(0)
	fig = plt.figure(num=1, figsize=(15, 8),dpi=80)
	plt.plot(all['time'].values,all['score'].values/all['review_date'].values)
	# plt.show()
	plt.plot(all['time'].values,all['score_bad'].values/all['review_date'].values)
	# plt.plot(all['time'].values,all['review_date'].values)
	plt.legend(['good','bad'],loc = 'best')

	plt.xticks (size='small', rotation=90, fontsize=13)
	plt.show()

# fig('hair_dryer',hair_dryer)
# fig('microwave',microwave)
# fig('pacifier',pacifier)

def classify(prod,data):
	data=gen_score(prod, data)
	cols_x = ['helpful_votes', 'total_votes', 'verified_purchase', 'review_body',
'review_date', 'month', 'rate','score']
	x=data[cols_x]
	scores=data['star_rating'].values
	y=[]
	for score in scores:
		if score>=4:
			y.append(1)
		else:
			y.append(0)
```

```
    X_train,            X_test,            y_train,            y_test            =
train_test_split(x,y,test_size=0.2,random_state=0)
    ss = StandardScaler ()
    X_train = ss.fit_transform (X_train)
    X_test = ss.fit_transform (X_test)
    lr = LogisticRegression()
    lr.fit (X_train, y_train)
    lr_y_predict = lr.predict (X_test)
    print(lr_y_predict)
    print ('Accuracy of LR Classifier:', lr.score (X_test, y_test))

    print()
classify('hair_dryer',hair_dryer)
print()
```

code	T2_c	remark	the code of question 2-c
code	T2_d	remark	the code of question 2-d

```
import   pandas   as   pd

del_cols                                                                          =
[&apos;customer_id&apos;,&apos;review_id&apos;,&apos;product_id&apos;,&apos
;helpful_votes&apos;,&apos;total_votes&apos;,&apos;vine&apos;,&apos;verified_p
urchase&apos;,&apos;review_date&apos;,&apos;year&apos;,&apos;month&apos;]

hair_dryer=pd.read_csv(&apos;../Data/new_hair_dryer.csv&apos;,encoding=&apos;u
tf-8&apos;,index_col=0)
hair_dryer=hair_dryer.drop(del_cols,axis=1)
microwave=pd.read_csv(&apos;../Data/new_microwave.csv&apos;,encoding=&apos;
utf-8&apos;,index_col=0)
microwave=microwave.drop(del_cols,axis=1)
pacifier=pd.read_csv(&apos;../Data/new_pacifier.csv&apos;,encoding=&apos;utf-
8&apos;,index_col=0)
pacifier=pacifier.drop(del_cols,axis=1)
print(&apos;hair_dryer\n&apos;,hair_dryer.corr(&apos;spearman&apos;))
print()
print(&apos;microwave\n&apos;,microwave.corr(&apos;spearman&apos;))
print()
print(&apos;pacifier\n&apos;,pacifier.corr(&apos;spearman&apos;))
print()
```

code	T2_e	remark	the code of question 2-e

```
import collections
import pickle
import numpy as np
import matplotlib.pyplot as plt
import jieba.analyse
import seaborn as sns
from tqdm import tqdm

from my_util import pre_process
import pandas as pd
import wordcloud
#       TF - IDF   jieba.analyse.extract_tags (sentence, topK=20, withWeight=False,
allowPOS=())
#        TextRank    jieba.analyse.textrank (sentence, topK=20, withWeight=False,
allowPOS=('ns', 'n', 'vn', 'v'))
hair_dryer=pd.read_csv('../Data/hair_dryer.csv',encoding='utf-8')
microwave=pd.read_csv('../Data/microwave.csv',encoding='utf-8')
pacifier=pd.read_csv('../Data/pacifier.csv',encoding='utf-8')
hair_dryer = hair_dryer.dropna ()
microwave = microwave.dropna ()
pacifier = pacifier.dropna ()
def try1():
    def gen_star_sent(n):
        tmp1=hair_dryer[hair_dryer['star_rating']==n]['review_body']
        tmp2=microwave[microwave['star_rating']==n]['review_body']
        tmp3=pacifier[pacifier['star_rating']==n]['review_body']
        star_str=''
        for i in tqdm(tmp1.values):
            for j in pre_process(i):
                star_str=star_str+' '+j
        for i in tqdm(tmp2.values):
            for j in pre_process (i):
                star_str = star_str + ' ' + j
        for i in tqdm(tmp3.values):
            for j in pre_process (i):
                star_str = star_str + ' ' + j
        print()
        return star_str
    one_star_sen=gen_star_sent(1)
    two_star_sen=gen_star_sent(2)
    three_star_sen=gen_star_sent(3)
    four_star_sen=gen_star_sent(4)
    five_star_sen=gen_star_sent(5)
    #
one_star_sen=hair_dryer[hair_dryer['star_rating']==1]['review_body']+microwave[mi
crowave['star_rating']==1]['review_body']+pacifier[pacifier['star_rating']==1]['review
_body']
	#	keywords=jieba.analyse.extract_tags(one_star_sen,	topK=20,
withWeight=False, allowPOS=())
	# print(keywords)
	w = wordcloud.WordCloud(max_words=50)
	w.generate(one_star_sen)
	w.to_file('output1.png')

	#	keywords=jieba.analyse.extract_tags(two_star_sen,	topK=20,
withWeight=False, allowPOS=())
	# print(keywords)
	w = wordcloud.WordCloud(max_words=50)
	w.generate(two_star_sen)
	w.to_file('output2.png')

	keywords=jieba.analyse.extract_tags(three_star_sen,	topK=20,
withWeight=False, allowPOS=())
	print(keywords)
	w = wordcloud.WordCloud(max_words=50)
	w.generate(three_star_sen)
	w.to_file('output3.png')

	#	keywords=jieba.analyse.extract_tags(four_star_sen,	topK=20,
withWeight=False, allowPOS=())
	# print(keywords)
	w = wordcloud.WordCloud(max_words=50)
	w.generate(four_star_sen)
	w.to_file('output4.png')

	#	keywords=jieba.analyse.extract_tags(five_star_sen,	topK=20,
withWeight=False, allowPOS=())
	# print(keywords)
	w = wordcloud.WordCloud(max_words=50)
	w.generate(five_star_sen)
	w.to_file('output5.png')

def try2():
	# words_list=set()
	# with open('emotion_dict/words_list.txt','r',encoding='utf-8') as f:
	#	for line in f:
	#		words_list.add(line.replace('\n',''))
	# def gen_star_sent(n):
	#	tmp1 = hair_dryer[hair_dryer['star_rating'] == n]['review_body']
```

```
#       tmp2 = microwave[microwave['star_rating'] == n]['review_body']
#       tmp3 = pacifier[pacifier['star_rating'] == n]['review_body']
#
#       star_str = ''
#       for i in tqdm(tmp1.values):
#           for j in pre_process(i):
#               if j in words_list:
#                   star_str = star_str + ' ' + j
#       for i in tqdm(tmp2.values):
#           for j in pre_process (i):
#               if j in words_list:
#                   star_str = star_str + ' ' + j
#       for i in tqdm(tmp3.values):
#           for j in pre_process (i):
#               if j in words_list:
#                   star_str = star_str + ' ' + j
#       print()
#
#       return star_str
#
# one_star_sen = gen_star_sent (1)
# two_star_sen = gen_star_sent (2)
# three_star_sen = gen_star_sent (3)
# four_star_sen = gen_star_sent (4)
# five_star_sen = gen_star_sent (5)
# star_sent = {}
# star_sent['one'] = one_star_sen
# star_sent['two'] = two_star_sen
# star_sent['three'] = three_star_sen
# star_sent['four'] = four_star_sen
# star_sent['five'] = five_star_sen
# pickle.dump (star_sent, open ('star_sent_cloud.pkl', 'wb'))
star_sent=pickle.load(open('star_sent_cloud.pkl','rb'))

w = wordcloud.WordCloud (max_words=50)
w.generate (star_sent['one'])
w.to_file ('output1.png')


w = wordcloud.WordCloud (max_words=50)
w.generate (star_sent['two'])
w.to_file ('output2.png')

w = wordcloud.WordCloud (max_words=50)
```

```
    w.generate (star_sent['three'])
    w.to_file ('output3.png')

    w = wordcloud.WordCloud (max_words=50)
    w.generate (star_sent['four'])
    w.to_file ('output4.png')

    w = wordcloud.WordCloud (max_words=50)
    w.generate (star_sent['five'])
    w.to_file ('output5.png')

def try3(num):
    words_list = set ()
    with open ('emotion_dict/words_list.txt', 'r', encoding='utf-8') as f:
        for line in f:
            words_list.add (line.replace ('\n', ''))

    def gen_star_sent1(n):
        tmp1 = hair_dryer[hair_dryer['star_rating'] == n]['review_body']
        tmp2 = microwave[microwave['star_rating'] == n]['review_body']
        tmp3 = pacifier[pacifier['star_rating'] == n]['review_body']

        star_str = []
        for i in tqdm (tmp1.values):
            for j in pre_process (i):
                if j in words_list:
                    star_str .append(j)
        for i in tqdm (tmp2.values):
            for j in pre_process (i):
                if j in words_list:
                    star_str .append(j)
        for i in tqdm (tmp3.values):
            for j in pre_process (i):
                if j in words_list:
                    star_str .append(j)
        print()

        return star_str

    # one_star_sen = gen_star_sent1 (1)
    # two_star_sen = gen_star_sent1 (2)
    # three_star_sen = gen_star_sent1 (3)
    # four_star_sen = gen_star_sent1 (4)
    five_star_sen = gen_star_sent1 (5)
```

```
# star_sent={}
# star_sent['one']=one_star_sen
# star_sent['two']=two_star_sen
# star_sent['three']=three_star_sen
# star_sent['four']=four_star_sen
# star_sent['five']=five_star_sen
# pickle.dump(star_sent,open('star_sent_count.pkl','wb'))
star_sent=pickle.load(open('star_sent_count.pkl','rb'))
words_set=[]
words_dict={}

word_counts = collections.Counter (star_sent['five'])    #
word_counts_top20 = word_counts.most_common (num)    #              20
# print (word_counts_top20)    #
word_counts_top20 = dict (word_counts_top20)
words_set += list (word_counts_top20.keys ())
words_dict['5'] = word_counts_top20
y5 = list(word_counts_top20.values())
x5 = [5 for _ in range (len (y5))]

word_counts = collections.Counter (star_sent['four'])    #
word_counts_top20 = word_counts.most_common (num)    #              20
# print (word_counts_top20)    #
word_counts_top20 = dict (word_counts_top20)
words_set += list (word_counts_top20.keys ())
words_dict['4'] = word_counts_top20
y4 = list(word_counts_top20.values())
x4 = [4 for _ in range (len (y4))]

word_counts = collections.Counter (star_sent['three'])    #
word_counts_top20 = word_counts.most_common (num)    #              20
# print (word_counts_top20)    #
word_counts_top20 = dict (word_counts_top20)
words_set += list (word_counts_top20.keys ())
words_dict['3'] = word_counts_top20
y3 = list(word_counts_top20.values())
x3 = [3 for _ in range (len (y3))]

word_counts = collections.Counter (star_sent['two'])    #
word_counts_top20 = word_counts.most_common (num)    #              20
# print (word_counts_top20)    #
word_counts_top20 = dict (word_counts_top20)
words_set += list (word_counts_top20.keys ())
words_dict['2'] = word_counts_top20
```

```
    y2 = list(word_counts_top20.values())
    x2 = [2 for _ in range (len (y2))]

    word_counts = collections.Counter (star_sent['one'])    #
    word_counts_top20 = word_counts.most_common (num)    #            20
    word_counts_top20=dict(word_counts_top20)
    words_set+=list(word_counts_top20.keys())
    words_dict['1']=word_counts_top20
    # print (word_counts_top20)    #
    y1 = list(word_counts_top20.values())
    x1=[1 for _ in range(len(y1))]

    words_set=set(words_set)
    sorted(words_set)
    dic={'1':[],'2':[],'3':[],'4':[],'5':[]}
    for i in words_set:
        for j in range(1,6):
            if i in words_dict[str(j)].keys():
                dic[str(j)].append(words_dict[str(j)][i])
            else:
                dic[str(j)].append(0)

    data=pd.DataFrame(dic,index=words_set)
    data.to_csv('../Data/word_count'+str(num)+'.csv',encoding='utf-8')
    cmap = sns.cubehelix_palette (start=1.5, rot=3, gamma=0.8, as_cmap=True)
    sns.heatmap (data,linewidths = 0.05, vmax=5000, vmin=50, cmap=cmap)
    plt.show()
    plt.savefig('words_hot'+str(num)+'.png')


try3(10)
# try2()
```

5.2.1 Model Establishmen and analysis

We will follow the following process to build a model to solve the first question.

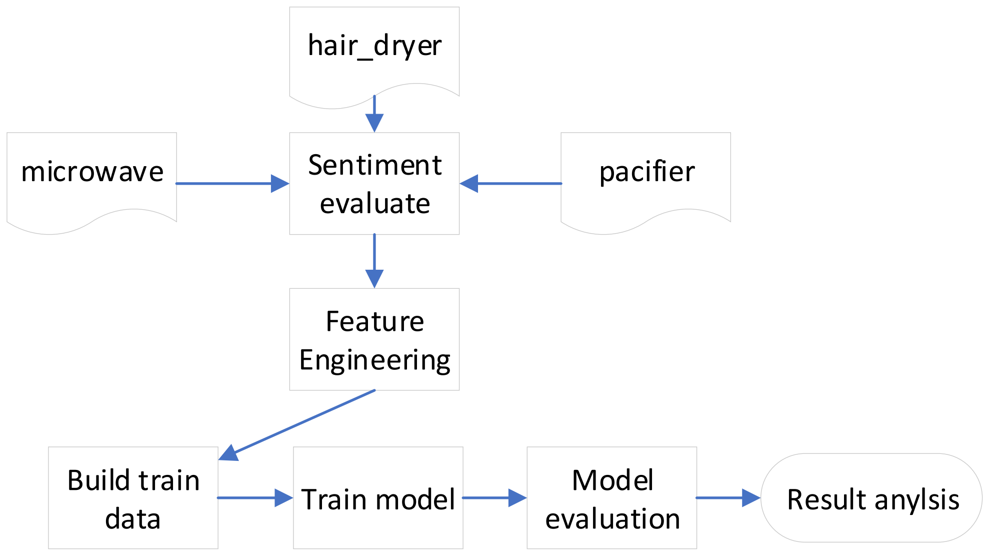


Figure 1: the process of building the model

Among them, hair_dryer, microwave and pacifier are data sets after data preprocessing. In the following part, the above process is described in detail.

5.2.2 Emotional Score

We use TextBlob model which is used in Sentiment Analysis to quantify the reviews_body. The result of TextBlob Sentiment Analysis can return a tuple --(polarity, subjectivity)

Table 2: Emotional analysis results

reviews	polarity	subjectivity
Works great!	1.000	0.750
Love this dryer!	0.625	0.600
Quiet, but does not seem like 1000 watt power···	0.000	0.333
Amazing addition to the nursery!	0.750	0.900
…	…	…

The polarity is a floating point number with a range of [-1.0, 1.0]. Positive Numbers mean positive and negative Numbers mean negative. Alfred is a floating point number with a range of [0.0, 1.0], where 0.0 is objective and 1.0 is subjective. We use the polarity value calculated by reviews to replace the original reviews_body data, so as to build our model.

After we use TextBlob to map the review body to the numerical space, We build Light gradient accelerator (Light) Gradient Boosting Machine (LightGBM) which is widely used in data mining tasks to get the weights to build model. We take star_rating as the output and the other features after preprocessing above as the input to train the model.

5.2.2 Feature Engineering

In order to obtain more data correlation information, we used feature engineering to process the data set, and then put it into the model for training. Feature engineering often plays an important role in the field of data mining. Different feature engineering can obtain different feature sets. A good feature engineering can reveal more relevant information in the data set, thus making the model more accurate.

Our observation data shows that there is a big gap between some comments and ratings. For example, we only give one star for good comments and five stars for bad comments.

Table 3: Comments on inconformity of star rating and review body

Review_id	Star_rating	Review_body
R134FUK2D9TQU6	1	I have used the dryer several times and it works great. I had questions which were answered promptly by other customers which was helpful in making my decision. Definitely recommend.
R1HI3QGXJQ2RUT	5	We owned these from the store and they are exactly the same. Too bad my grandson decided he was done with pacifiers one week later
…	…	…

It is found that there are 12, 5 and 10 pieces of such data in the data set hair player, microwave and pacifier, respectively. This paper considers that the occurrence of such data is abnormal and not normal evaluation information, so this kind of data will be deleted in the post-processing.

At the same time, we make statistics according to all the star rating information of a product_id, and use the following formula to calculate the praise rate of a product _id:

$$rate = \frac{\sum(starRating >= 4)}{count_i}$$

.

Among them,$count_i$ represent `product_id` for the total number of i.

Finally, we unify all kinds of string category data in the data into continuous numerical category data starting from 0, because the next model needs this form of category data.

5.2.2 Training data construction and Model training

After all the data are processed by feature engineering, a two classifier is trained. Therefore, those with a rating of 4 stars or above are regarded as positive examples, and those with a rating of less than 4 stars are regarded as negative examples. 20% of the data set size is used as the test data to evaluate the model, and the remaining data is used as the training data training model of the model. The final data size is as follows:

Table 4: Data volume of training set and test set

	Train data	Test data	Total
hair_dryer	9163	2291	11454
microwave	1288	322	1610
pacifier	15130	3783	18913

The hardware environment is 16g memory + Intel core-i78th, and the software platform is windows10. The related parameters of lightGBM model are as follows: num_learners = 64, learning_rate = 0.09, and then input the training data to the training model.

5.2.3 Model Evaluation

Using test data to evaluate the model, the evaluation index is precision, recall, F1 value, and the calculation formula is as follows:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2*P*R}{P + R}$$

Among them, TP indicates that the real evaluation results are positive examples and the predicted results of the model are also the total number of positive examples, FP represents that model predicts the total number of negative cases as positive cases, FN represents the total number of positive cases predicted as negative cases.

The test results are shown in Table 5.

Table 5: Test result

Product	Precision	Recall	F1
hair_dryer	0.812	0.982	0.889
microwave	0.846	0.903	0.874
pacifier	0.892	0.989	0.939

It can be seen from the above table that the F1 value of lightgbm model in three products is high, which proves that the effect of the model is very good.

5.2.2 Result Analysis

The lightGBM model has been successfully established above, and we can use the model to get the following data:

Table 6: Weight of eigenvalue

	Hair_dryer	Microwave	Pacifier
customer_id	0.000	0.000	0.000
review_id	0.000	0.000	0.000
product_id	0.036	0.000	0.009
helpful_votes	0.041	0.038	0.044
total_votes	0.138	0.122	0.139
vine	0.000	0.000	0.004
verified_purchase	0.028	0.028	0.025
review_body	0.223	0.329	0.285
review_date	0.200	0.213	0.201
year	0.003	0.000	0.000

month	0.053	0.081	0.055
rate	0.274	0.184	0.233

The values in the table are the relative weights of each feature. We can see that the weights of customer_ID and review_ID in the three data sets are all 0, indicating that they are not helpful for the market analysis of sunshine company, only exist as identifiers, and have no impact on the results. Secondly, we can see that the weights of review_body, rate, review_date and total_votes are relatively high, and to prove that they are very important for the rating of a product. Therefore, for sunshine company, we should focus on the above four aspects to achieve success.

5.3 The Solution for Task A

Once the products of sunshine company are sold, the model of the first question will be used to predict the user rating. If the prediction rating is consistent with the actual rating of the user and the rating is good, the importance of the first question will be used to analyze the success of the products, and the successful aspects will be done to the end to ensure that the product reputation does not decline. If the rating is poor, the importance will also be used to analyze the aspects of the products that are not enough, Next, we should focus on improving the product disadvantage and realizing the reversal. If it is inconsistent with the predicted rating and the actual rating of users, we should analyze whether these users are false users or malicious users, so as to deal with these users in a targeted way and ensure the normal sales market。

5.4 Task B——Entropy Weight Model

5.4.1 Model Building

For the three data sets provided, this paper uses the previous results of emotional analysis, combined with rating data, uses information entropy to determine the weight of the two, and finally gets the score of each evaluation, using the score to analyze the time pattern.

Information entropy has three properties: monotonicity, nonnegativity and accumulation：

1. Monotonicity: the higher the probability of occurrence, the lower the amount of information it carries;

2. Non negativity: information entropy can be regarded as a kind of breadth, and non negativity is a reasonable necessity;

3. Accumulation: that is, the measurement of the total uncertainty of multiple random events occurring at the same time can be expressed as the sum of the measurement of the uncertainty of each event, which is also a reflection of the breadth.

Assume that the user's rating and evaluation of goods are the sum of two events, and they are expected to be independent, the probability of their simultaneous occurrence is

$$p(X=A,Y=B)=p(X=A)\dot{p}(X=B)$$

According to the accumulation， we can infer that

$$H(p(X=A,Y=B))=H(p(X=A)\cdot p(Y=B))=H(p(X=A))+H(p(Y=B))$$

If the sum of the product function value of two variables is equal to the sum of the function value of two variables, it should be a logarithmic function. Considering that the probability is less than or equal to 1, it is less than 0 after taking the logarithm. Considering the second property of information entropy, it is necessary to add a negative sign in the front. Finally, the definition of information entropy given by Claude Shannon, the father of reference information theory, has the formula of information entropy as follows.

$$H(x) = -C\sum_{x\in X} p(x)\log p(x)$$

Take the data after emotional analysis as evaluation data and star rating as rating data, and finally calculate the product comprehensive score as follows.

$$score = emotionScore \cdot a + starRating \cdot b$$

Emotion score is the score of review_body sentiment analysis, and starrating is the rating of users. A and B calculate the information entropy weight of the two by using the information entropy formula, and calculate the weight values of the three data sets respectively, as shown in table 7:

Table 7: Information entropy weight

	a	b
hair_dryer	0.138	0.861
microwave	0.069	0.930
pacifier	0.169	0.831

Calculate the score of each product to analyze the time-based measurement mode of each data set. In this paper, the overall sales volume, favorable rate, poor rate and other data of each month are calculated by month for visual analysis.单月份整体销量 $sales$ represents overall sales volume in a single month, $good$ indicates favorable rate , bad indicates bad rate. The calculation formula is as follows:

$$sales = \sum_{date\in month_i} data$$

$$good = \left(\sum_{score>4} data\right)/sales$$

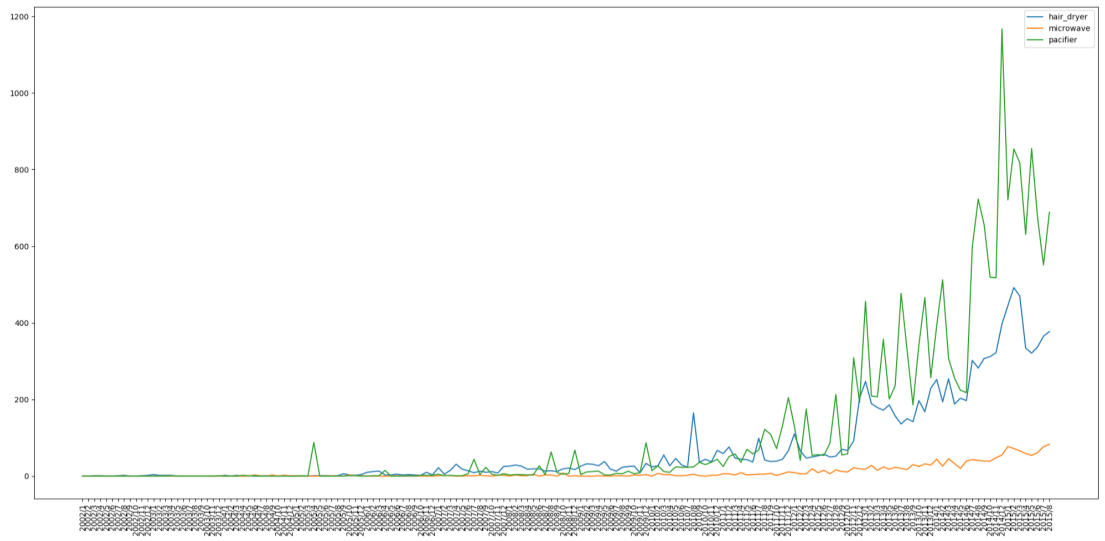
$$bad = \left(\sum_{score<1} data\right)/sales$$

$data$ represents a row of data,$date$ is in review_date, and$score$ is the score calculated by formula for each piece of data.

5.4.2 Results

1. Overall sales volume analysis and data selection

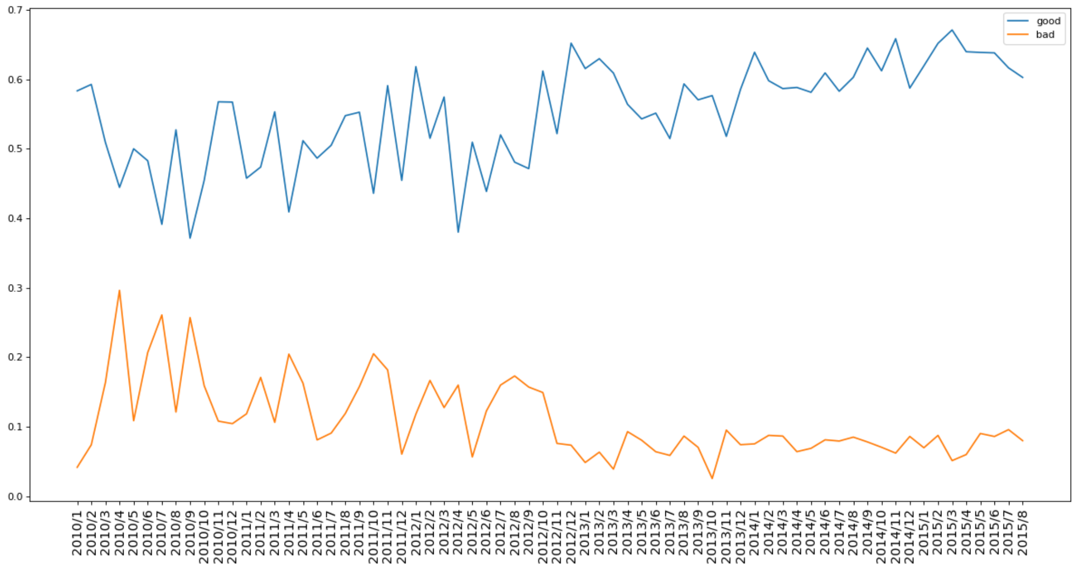
According to the pre-processing data, the sales figures of the three products are as follows:



Due to space limitation, image is blurred, and if you want to see the clear version, please refer to the appendix. Through the image, we can know that the sales volume before 2010 is very small. Therefore, the following analysis only takes the data after January 2010 for analysis. Meanwhile, it can be seen that the overall sales volume of the three products is increasing year by year. The sales volume of microwave ovens is relatively low, and the sales volume of diapers is relatively high. The reason may be that the microwave ovens are durable products, while diapers are disposable nondurable products, and the sales volume of hair dryer is in the middle. This is related to the characteristics and uses of the products.

(1)The analysis of hair's time pattern

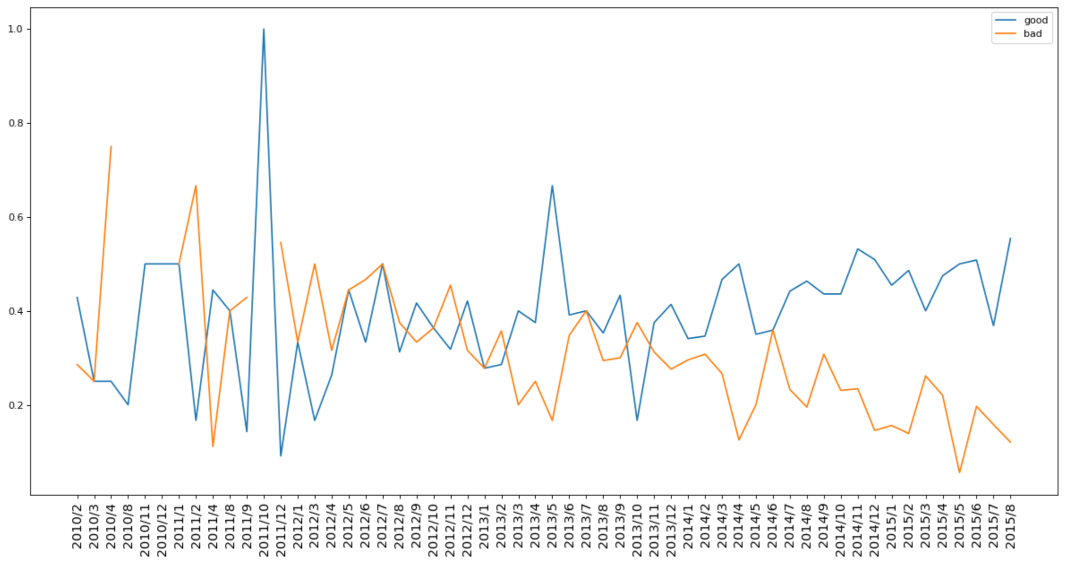
Using the data after preprocessing, we get the monthly positive and negative rate of hair player as shown in Figure 123:



From the figure, we can analyze that since January 2010, the overall positive rate of the hair dryer has increased, and the overall negative rate has declined. And the positive rate has always been far higher than the negative rate. Before January 2013, the negative rate of the whole product is high, and the product quality is not objective, but after 2013, the positive rate of the product is basically high, It shows that the quality of hair dryer gradually rises after market baptism, and people are more comfortable to use it, but there will still be a slight fluctuation

(2) Analysis of time pattern of microwave

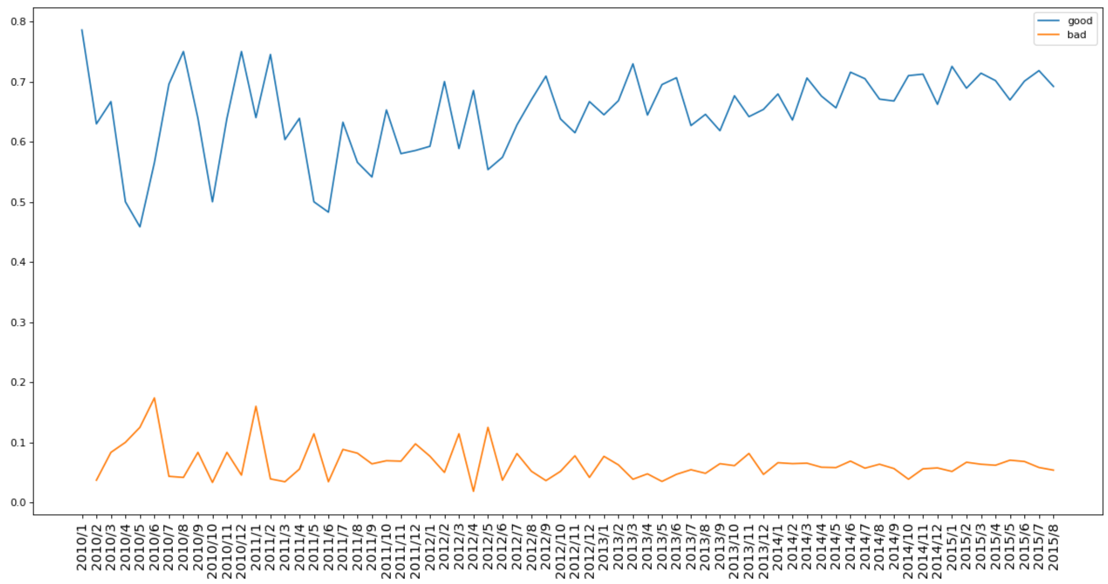
Using the data after preprocessing, the monthly positive and negative rates of microwave are as follows:



From the figure, we can see that in the early stage when the microwave oven was put into the market, the market's poor evaluation rate of the microwave oven was slightly higher than the high evaluation rate, and the good evaluation rate of the microwave oven peaked in October 2011, and then declined; from December 2012 to October 2013, the good evaluation rate and bad evaluation rate of the microwave oven fluctuated in a certain range and were relatively stable. In November 2013 and, the good evaluation rate of the product was within a certain range It fluctuates within the range but is always greater than the rate of poor evaluation. Generally speaking, the poor evaluation rate of microwave oven is declining, and the good evaluation rate is rising.

(3) Analysis of pacifier time pattern

Using the data after preprocessing, the monthly positive and negative rates of pacifier are as follows:



It can be analyzed from the figure that from the time when pacifier put the product into the market to August 2015, the product's favorable rate in the market is far higher than the negative rate, and the fluctuation frequency is getting smaller and smaller, which is in a very stable situation and a very good product.

5.5 The Solution for Task C

It is required to determine the combination of a text-based measurement method and a rating based measurement method to indicate the success or failure of a product. In the first question, the LGB model is established and the impact of the corresponding indicators on the quality of the product is obtained. In the last question, the score of the product is obtained by using the emotional score of rating and comment with the help of entropy weight method, Both of the above solutions refer to comments and ratings, i.e. text and ratings, so the combination of the two solutions best indicates potential successful or failed products.

5.5.1 Establishment and Solution of The Model

Combined with the feature weight of question 1, the features of relevant text and rating are further screened: review_body, review_date, rate, star_rating. At the same time, combined with the score of the previous question, the fuzzy comprehensive evaluation model is established to evaluate the potential success of the product. The basic steps of the fuzzy comprehensive evaluation model are as follows:

1. Select features;
2. Determine the evaluation matrix;
3. Check the evaluation matrix and calculate the weight;
4. Calculate the score and get the result.

The first step is to select features. Next, according to relevant data and data, determine the evaluation matrix M as follows:

$$M = \begin{bmatrix} 1, & 2, & \frac{1}{2}, & \frac{3}{4}, & \frac{3}{5} \\ \frac{1}{2}, & 1, & \frac{1}{2}, & \frac{1}{2}, & \frac{1}{3} \\ 2, & 2, & 1, & 2, & \frac{2}{3} \\ \frac{4}{3}, & 2, & \frac{1}{2}, & 1, & \frac{1}{2} \\ \frac{5}{3}, & 3, & \frac{3}{2}, & 2, & 1 \end{bmatrix}$$

Through the consistency test of holding a, it is concluded that CR = 0.018 < 0.1, so the matrix passes the consistency test. Finally, the eigenvector of the matrix is calculated and normalized to obtain the weight of the corresponding features as follows:

Table 8: The weight of the corresponding features

Features	Weights
review_body	0.158
review_date	0.098
rate	0.259
star_rating	0.170
score	0.315

The company can use the weight value to evaluate the success of the product. The larger the value is, the more successful the product is.

5.6 Task D

We use the Spearman model to measure the correlation between ratings and reviews. The final result of the Spearman model is a rank correlation coefficient, which is solved according to the order of original data.

1. We sort each piece of data according to the comment time;
2. Extract the scores of star rating and review of each data as two column vectors R (XI) and R (Yi);
3. Output R (XI) and R (Yi) to the following formula:

$$d = \sum_{i=1}^{N} |R(X_i) - R(Y_i)|^2$$

4. Finally, the correlation RS between the two column vectors is calculated according to the following formula, and the results are shown in table 9.

$$Rs = 1 - \frac{6 \times d}{N \times (N^2 - 1)}$$

Table 9:Coefficient for products

Product	Coefficient
hair_dryer	0.437
microwave	0.559
pacifier	0.372

It can be concluded from the table 9 that the correlation coefficient of the three products is greater than 0.05, so specific star ratings will affect the user to write some kind of comment. And we found in the data that some of the comments were indeed affected by the previous comments. And we found in reviews that some of the ratings are as shown in table 10.

Table 10: Partial comment

Review_id	Reviews
R3NQ2GXMX8FDFI	Here's a novel idea!! Read the reviews BEFORE ...
RK1L6FAK78AN4	- DO NOT BUY - the 5 star reviews are fake
ROTACUL0CWK71	I don't understand the great reviews for this dryer
R3EPB70VVPJALX	Believe the Reviews!!!
...	...

It can be seen from the table 10 that the previous rating comments and other information are mentioned in the comments, indicating that some reviews will be affected by other ratings.

5.7 The Solution for Task E

According to the specific quality descriptors of the text comments, the topic is required to be associated with the rating to find out whether there is deep-seated information. Therefore, this paper digs the specific content of the review body, extracts its related description information, and digs it by means of statistics and drawing. The specific steps are as follows:

1. Establish a specific dictionary of quality descriptors. By consulting materials and relevant data, this paper establishes a user emotion comment degree dictionary, which contains a variety of English description emotions, comment words, degree words and so on. See the appendix for the specific vocabulary;

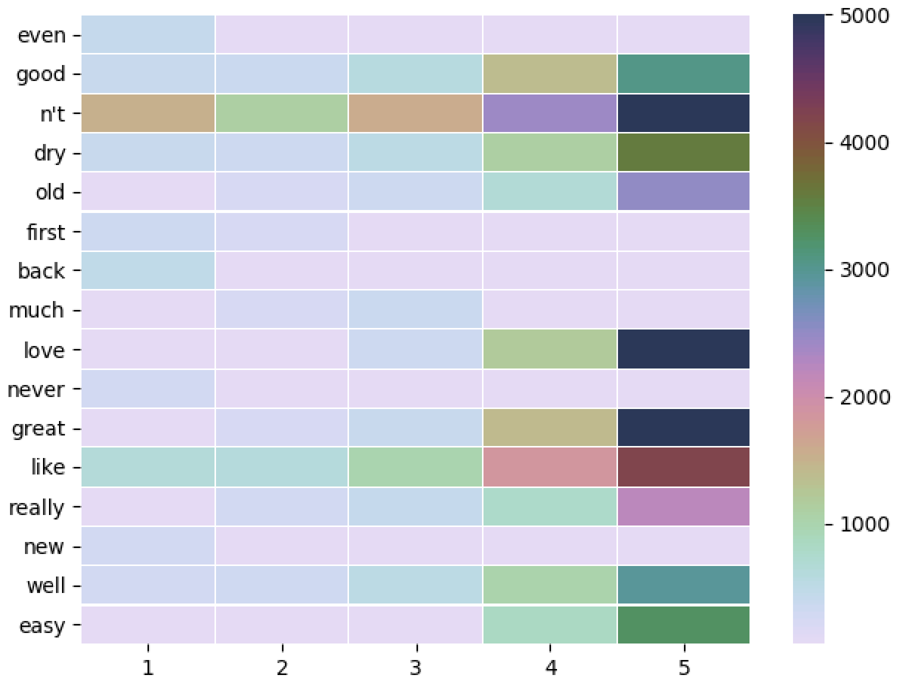
2. Combine the data of hair dryer, microwave and pacifier, extract the content of review body. According to the classification of user rating, count the words appearing in the dictionary from review body, and get the frequency of each word through the word frequency statistical method. Arrange the results in descending order, and then take out the first 20 words. The following table 11 is obtained:

Table 11: Frequency of common words

	1	2	3	4	5
old	272	225	353	677	2512
never	298	0	0	0	0
long	241	186	327	563	1392
new	286	0	0	0	0
good	392	368	591	1402	3058
much	228	234	370	606	2079
back	460	212	0	0	0
bad	222	0	0	0	0
high	0	0	232	0	0
however	0	0	265	0	0

need	230	157	252	566	1645
receive	222	0	0	0	0
keep	0	196	261	479	1695
cute	0	0	0	0	1479
perfect	0	0	0	0	1600
star	0	0	0	440	0
hot	0	187	0	0	0
easy	0	0	0	832	3261
dry	386	326	535	1108	3578
n't	1509	1102	1594	2445	5932
really	237	268	417	784	2209
want	255	157	274	437	0
love	0	159	359	1184	8867
great	221	237	388	1405	5797
nice	0	0	0	558	1405
small	0	165	279	549	0
like	642	602	984	1851	4186
best	0	0	0	0	1448
even	401	208	233	0	0
first	355	223	0	0	1363
still	228	0	255	506	0
set	0	163	272	607	1378
well	285	302	531	1019	2932

Use table data to draw thermodynamic diagram (Figure XXX) for intuitive analysis.



In the figure, the abscissa is the star_rating, and the ordinate is the word. From the figure and table, it can be seen that "n't" is a negative word, which appears 1509 times in review_body of five stars , followed by never, like, etc.; while the word with 5 stars is love (8867 times), followed by great, like, etc, These words hardly appears in one or two stars. This shows that specific description words are closely related to rating. At the same time, we also notice that "n't" appears many times in all stars. Through specific review body data analysis, in one star, many of them appear at the same time with other positive words to express negative emotions, For example, "n't like" and so on. In five-star data, it often forms many positive emotional phrases, such as "like n't allow finger print" and so on.

Based on the above analysis, we summarize the following common star vocabulary:

1	2	3	4	5
not like	not good	not like	like	love
never	not like	not good	well	great
back	old	much	great	easy
even	not well	old	not	like
bad	dry	however	need	well

1.8 Task F

Dear Sir/Madam,

We are very honored to have this opportunity to show you and your company the results of our mining and analysis of the data of hair_dryer, microwave and pacifier. Here we will give you a comprehensive report about our research in the data.

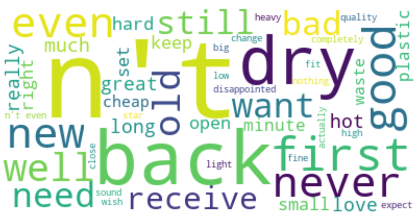
First of all, through the preliminary observations provide three data sets, we find that each product in the marketplace in the US, and product_category respectively within each data set is the same, we think it useless for analysis, so to be deleted. Similarly, found product_parent data with product_id data redundancy, then we delete the product_parent column data. At the same time, do other data preprocessing.

After emotional analysis in natural language processing, we will not quantitative reviews is mapped to the numerical space, using the numerical range of [1, 1] to comment contains specific emotions. The higher the number, the stronger the positive emotion. After the emotional analysis, we carried out feature engineering. A good feature engineering can make the model better grasp the data features, better discover the internal patterns in the data, and improve the accuracy of the model. Through feature engineering, a new feature rating rate was established and some abnormal conditions were removed. For example, a 1-star comment: "I have used the dryer several times and it works great. I had questions which were answered by other customers which was helpful in making my decision. Definitely recommend.". However, this should be a 5-star favorable rating, so the data with opposite star rating and rating were deleted. Fortunately, that's not a lot of data, with only 27 of the three data sets. Then, we divided the data into the training set and the test set according to the ratio of 8:2, and took the ones with a rating greater than or equal to 4 stars as the positive example, and the ones with a rating less than 4 stars as the negative example, and used the machine learning algorithm LightGBM to conduct separate data modeling for each data set. Finally, the F1 value of the model on the three data sets was 0.889, 0.874 and 0.939, which showed that our model was very successful! Not only that, we also learned from the model that review_body, review_date, rate and total_votes are the most important to the model. Your company should mainly focus on the changes of these four types of data, and try to pay little or no attention to other data with low importance such as reviewers. Once your company plans to sell the three products online, you can use the above model to evaluate the sales data of the products, and compare the predicted results of the model with the actual scores of users to get the result of whether the two are consistent. If it is consistent, it will seize the evaluation characteristics of such users and make targeted improvements to the product to achieve better sales. If it is inconsistent, you can focus on whether such users are fake or malicious users and so on.

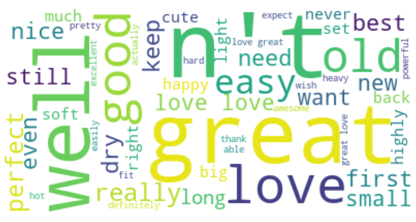
We also use entropy weight method to obtain product scores from comprehensive analysis of rating and evaluation, which can measure product reputation and other relevant information. In order to evaluate the potential success or failure of a product, we combined the results of the established machine learning model LighGBM and entropy weight analysis, selected important indicators related to text and rating for fuzzy comprehensive evaluation modeling, and finally concluded that the potential success of a product can be indicated by calculating specific scores.

Finally, we use the correlation analysis technique to obtain that certain ratings will trigger certain types of reviews, that is, users' reviews may be influenced by previous ratings. For example, "-do NOT BUY - the 5 star reviews are fake".

At the same time, we set up a specific vocabulary list and drew the following two word cloud maps for the word frequency statistics of all comment data. The font size of the words in the word cloud map is related to how often they appear in the comments. The higher the frequency, the larger the font. We can see from the word cloud map that the one-star word cloud map is mostly negative words, such as “n't” “like”, “never”, “bad”, while the five-star word cloud map is mostly ‘great’, ‘love’, ‘well’, ‘perfect’, etc. Although there is “n't” in the word cloud image of five stars, we find that it mainly represents positive emotions through specific analysis, such as “like n't allow finger print” and so on.



One star word cloud map



Five star word cloud map

Our data mining results are presented, I believe you will have a lot of harvest after you see, finally, I hope your company's products can be a great success!

2. Conclusion

Through data mining and in-depth analysis, we not only have a lot of perceptual knowledge about the data, but also a lot of rational knowledge. We find that many times, it is very important to investigate a product not only from the perspective of product star_rating, evaluation, review date and other data. And through the statistical analysis of the time patterns of the three products, we get the sales volume of the product, It is found that with the delay of time, pacifier has more and more favorable comments, and the favorable rate is relatively stable. Although the favorable rate of the other two products is rising, there is still a trend of fluctuation, and the development is not stable, especially the phenomenon that the negative rate is greater than the favorable rate once appeared in microave, It shows that the market of microwave can not meet people's needs. Sunshine company should pay attention to its performance, ensure product quality and pay attention to market changes, and take targeted rescue measures for different market environment to ensure the company's benefits.

Using the emotion analysis technology to analyze the emotion in the review_body, and then comparing with the rating, we found that there are five-star-poor rating and one star-high praise and other abnormal situations. At the same time, we found that whether the evaluation affects the rating and whether the rating affects the evaluation. They affect each other, and a good rating will have a certain impact on the next comment, Different comment phrases are often corresponding to different ratings. Therefore, in the actual analysis, if there are different trends between the two, it can prove that there may be some unreasonable factors in the market. At this time, the company needs to investigate and solve them, and finally make the product develop well.

3. Evaluating the Model

7.1 Advantages of the model

1. We use TextBlob to analyze the emotion of each review_body, and score them, so objectively get the rating of the product;

2. In solving problem D, we use the Spearman coefficient analysis, which requires no normal distribution, and it can get the rank coefficient, that is, it can find out the relationship between variables and variables in a certain order;

3. The model objectively uses as much data as possible, and adjusts it according to the real data. Through the visual analysis of the data, it can directly mine out a large amount of information of the data, and analyze the trend, reputation, success and other specific conditions of the three products through the information above.

7.2 Disadvantages of the model

1. Due to the time limits, when building the LightGBM model, the final result is good enough, but may not reach the best;

2. The evaluation matrix of fuzzy comprehensive evaluation method may not be objective, but it has guaranteed the objectivity through data as far as possible;

3. In terms of word frequency statistics, some phrases are not well analyzed, for example, phrases like "n't like" that express emotion through all words are actually segmented, but this is not well done, resulting in incomplete presentation.

7.3 Application of the model

1. The model can be used to evaluate the initial stage of the product and analyze whether it can be sold well;

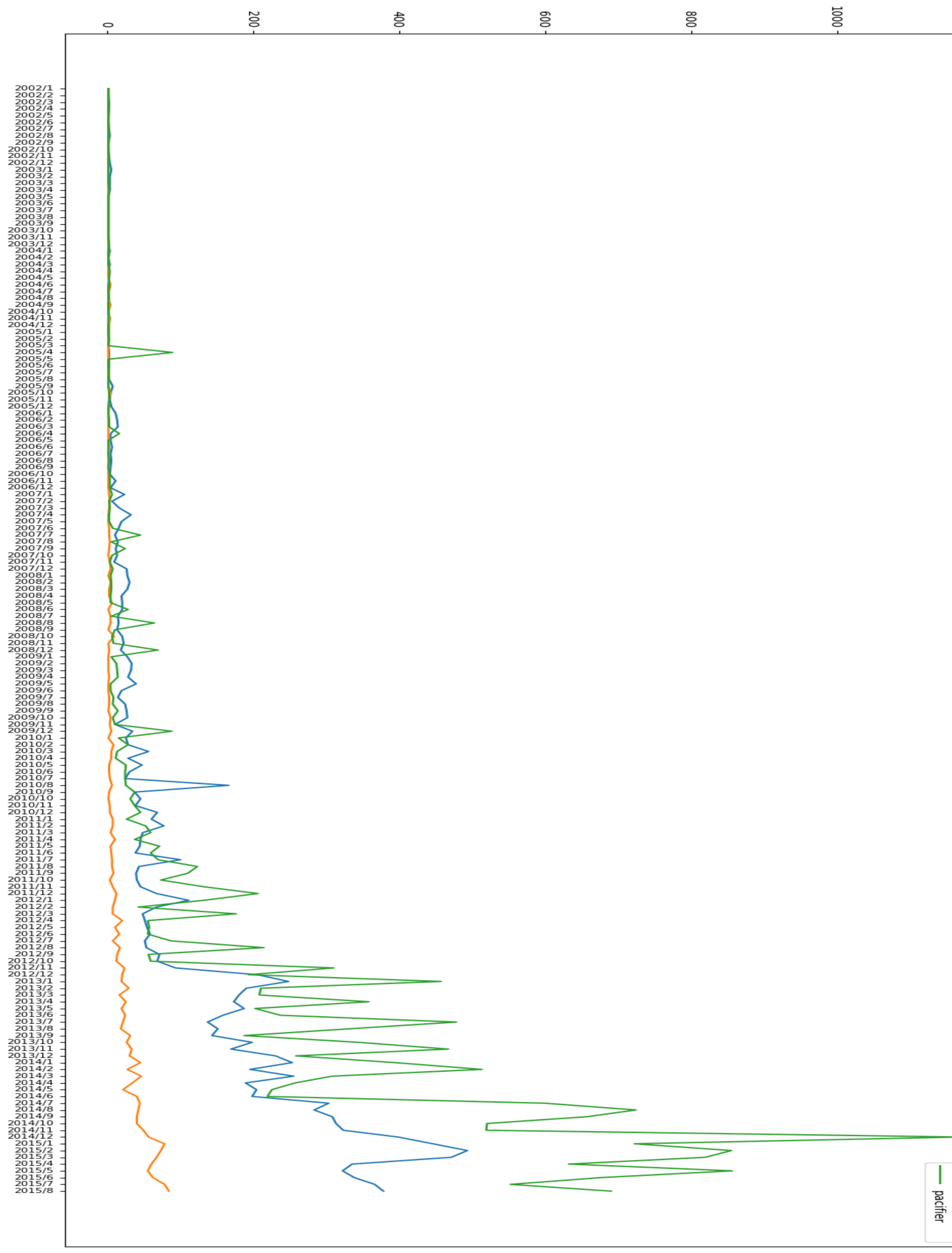
2. The model can analyze the change of product reputation, and the company can adjust the product sales strategy in time;

3. The model can objectively evaluate the success of the product and feed back relevant information to the company in time

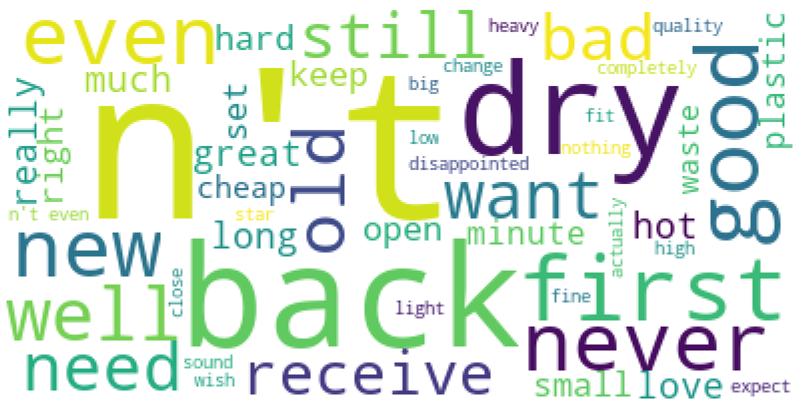
4. Reference

[1] KE G, MENG Q, FINLEY T, et al. LightCBM: a highly efficient gradient boosting decision tree [C]// Proceedings of the 2017 Annual Conference on Neural Information Processing Systems. New York: Curran Associates Inc., 2017: 3146 -3154.
[2] Zhang Jin,Mucs Daniel,Norinder Ulf,Svensson Fredrik. LightGBM: An Effective and Scalable Algorithm for Prediction of Chemical Toxicity-Application to the Tox21 and Mutagenicity Data Sets.[J]. Journal of chemical information and modeling,2019,59(10).
[3] Mudambi S. and D. Schuff "What Makes a Helpful Online Review? A Study of Customer Reviews on Amazon.com " MIS Quarterly 34(1) 2010 pp. 185-200.
[4] Baek, H., J. Ahn, and Y. Choi. 2012. Helpfulness of online consumer reviews: Readers' objectives and review cues. International Journal of Electronic Commerce 17, no. 2: 99–126.
[5] Pang, B. and Lee, L. “A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts,” in Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, Stroudsburg, PA, 2004. Article 271.
[6] Zhang, M., C. Dellarocas, and N. Awad, “The Impact of Online Movie Reviews on Box Office Performance”, In Workshop on Information Systems and Economics (WISE), College Park, MD, 2004.

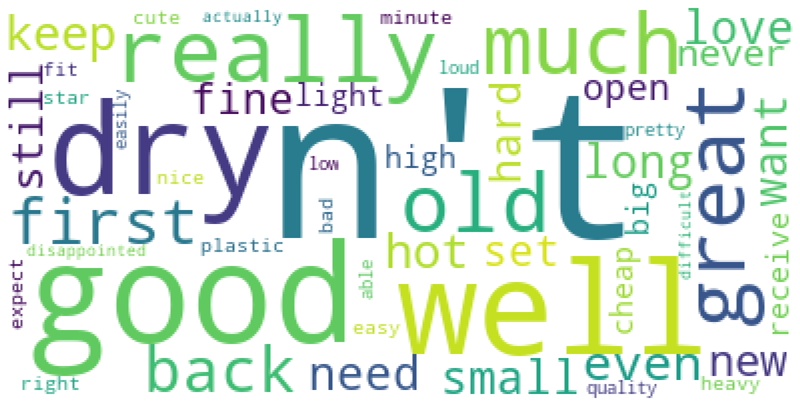
appendix 1: hair_dryer, microwave, pacifier monthly sales statistics


0
200
400
600
800
1000
2002/1
2015/8
pacifier

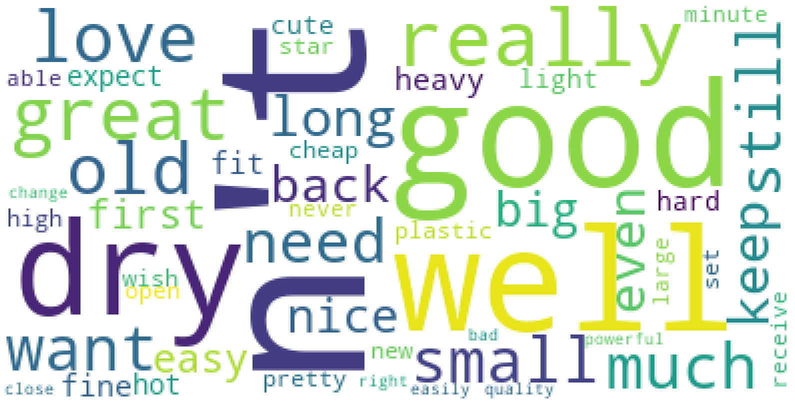
appendix 2: Star rating words cloud chart



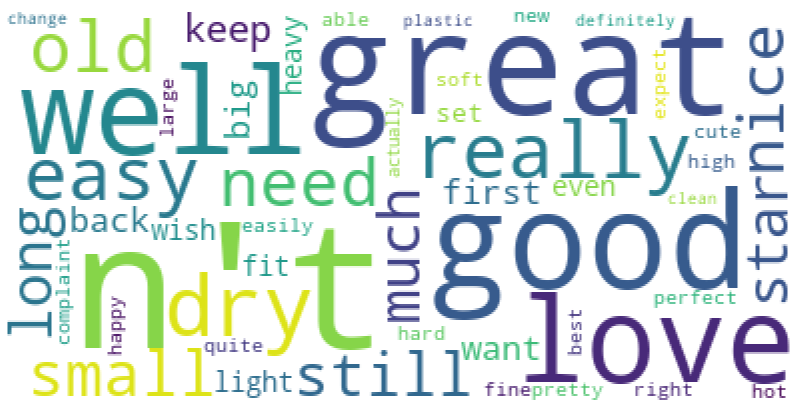
one star review words



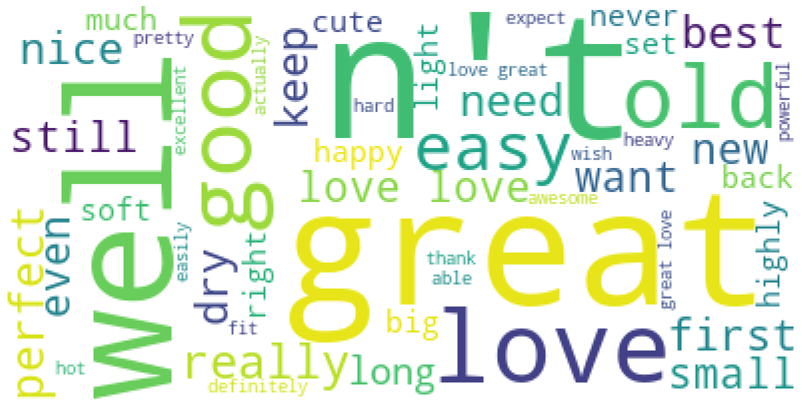
two stars review words



three stars review words



four stars review words



five stars review words

appendix 3: code

code	anylisis	remark	the code of data anylisis

```
import pandas as pd
import matplotlib.pyplot as plt
from tqdm import tqdm
from my_util import pre_process

def null_process():
    hair_dryer=pd.read_csv('../Data/hair_dryer.tsv',sep='\t',encoding='utf-8')
    microwave=pd.read_csv('../Data/microwave.tsv',sep='\t',encoding='utf-8')
    pacifier=pd.read_csv('../Data/pacifier.tsv',sep='\t',encoding='utf-8')
    # print(hair_dryer.head())

    for column in hair_dryer.columns:
        if len(set(hair_dryer[column]))==1:
            print('hair_dryer:',column)
        if len(set(microwave[column]))==1:
            print('microwave:',column)
        if len(set(pacifier[column]))==1:
            print('pacifier:',column)
    print(set(pacifier['product_category']))
    print(set(pacifier['marketplace']))
    print(set(microwave['product_category']))
    print(set(microwave['marketplace']))
    #'product_category', 'marketplace'                          ,
    hair_dryer['review_date']   =   pd.to_datetime   (hair_dryer['review_date'],
format='%m/%d/%Y')
    hair_dryer['year'] = hair_dryer['review_date'].dt.year
    hair_dryer['month'] = hair_dryer['review_date'].dt.month

    pacifier['review_date']   =   pd.to_datetime   (pacifier['review_date'],
format='%m/%d/%Y')
    pacifier['year'] = pacifier['review_date'].dt.year
    pacifier['month'] = pacifier['review_date'].dt.month

    microwave['review_date']   =   pd.to_datetime   (microwave['review_date'],
format='%m/%d/%Y')
    microwave['year'] = microwave['review_date'].dt.year
    microwave['month'] = microwave['review_date'].dt.month
    del hair_dryer['product_category']
    del hair_dryer['marketplace']
    del pacifier['product_category']
    del pacifier['marketplace']
    del microwave['product_category']
    del microwave['marketplace']
    del hair_dryer['product_title']
    del pacifier['product_title']
    del microwave['product_title']
    tmp1 = pacifier[pacifier['product_id'] == 'b0042i2bwg']
    print (tmp1)
    tmp2 = pacifier[pacifier['product_id'] == 'b00db5f114']
    print (tmp2)
    dic1 = {}
    for idx in hair_dryer.index:
        i = hair_dryer.loc[idx, 'product_id']
        j = hair_dryer.loc[idx, 'product_parent']
        if i not in dic1:
            dic1[i] = [j]
        else:
            dic1[i].append (j)
    for i in dic1:
        if len (set (dic1[i])) != 1:
            print ('hair_dryer')
```

```
    dic2 = {}
    for idx in microwave.index:
        i = microwave.loc[idx, 'product_id']
        j = microwave.loc[idx, 'product_parent']
        if i not in dic2:
            dic2[i] = [j]
        else:
            dic2[i].append (j)
    for i in dic2:
        if len (set (dic2[i])) != 1:
            print ('microwave')
    dic3 = {}
    for idx in pacifier.index:
        i = pacifier.loc[idx, 'product_id']
        j = pacifier.loc[idx, 'product_parent']
        if i not in dic3:
            dic3[i] = [j]
        else:
            dic3[i].append (j)
    for i in dic3:
        if len (set (dic3[i])) != 1:
            print (i, dic3[i])
            print ('pacifier')
    #                    pacifier                                ,                    ,
product_id     , product_parent
    del hair_dryer['product_parent']
    del microwave['product_parent']
    del pacifier['product_parent']

    print(hair_dryer['product_id'].count()) #11470
    print(microwave['product_id'].count())#1615
    print(pacifier['product_id'].count())#18939
    hair_dryer=hair_dryer.dropna()
    microwave=microwave.dropna()
    pacifier=pacifier.dropna()
    print(hair_dryer['product_id'].count())#11468
    print(microwave['product_id'].count())#1615
    print(pacifier['product_id'].count())#18937
    #                         ,              ,
    reviewer_body = []
    for i in tqdm (hair_dryer['review_body'].values):
        sent = ''
        for j in pre_process (i):
            sent = sent + ' ' + j
        reviewer_body.append (sent)
    hair_dryer['review_body'] = reviewer_body

    reviewer_body = []
    for i in tqdm (microwave['review_body'].values):
        sent = ''
        for j in pre_process (i):
            sent = sent + ' ' + j
        reviewer_body.append (sent)
    microwave['review_body'] = reviewer_body

    reviewer_body = []
    for i in tqdm (pacifier['review_body'].values):
        sent = ''
        try:
            for j in pre_process (i):
                sent = sent + ' ' + j
        except:
            print (i)
            sent = i
        reviewer_body.append (sent)
    pacifier['review_body'] = reviewer_body

    hair_dryer = hair_dryer.dropna ()
    microwave = microwave.dropna ()
    pacifier = pacifier.dropna ()
    print (hair_dryer['product_id'].count ())    # 11468
    print (microwave['product_id'].count ())    # 1615
    print (pacifier['product_id'].count ())    # 18937
    hair_dryer.to_csv('../Data/hair_dryer.csv',encoding='utf-8',index=None)
    microwave.to_csv('../Data/microwave.csv',encoding='utf-8',index=None)
    pacifier.to_csv('../Data/pacifier.csv',encoding='utf-8',index=None)

    #
    # print(hair_dryer[hair_dryer.isnull().values==True])
    # print(microwave[microwave.isnull().values==True])
    # print(pacifier[pacifier.isnull().values==True])

# null_process()
hair_dryer=pd.read_csv('../Data/hair_dryer.csv',encoding='utf-8')
microwave=pd.read_csv('../Data/microwave.csv',encoding='utf-8')
pacifier=pd.read_csv('../Data/pacifier.csv',encoding='utf-8')
print(hair_dryer.columns)
def fig_star_rating_count():
    tmp1=hair_dryer.groupby(by='star_rating').count()['customer_id']
    plt.subplot(221)
    plt.bar(tmp1.index.values,tmp1.values)
    plt.ylim(0,8000)
    for a, b in zip(tmp1.index.values, tmp1.values):
        plt.text(a, b, '%.0f' % b, ha='center', va='bottom', fontsize=8)
    plt.title('hair_dryer')

    tmp2=microwave.groupby(by='star_rating').count()['customer_id']
    plt.subplot(222)
    plt.ylim(0,800)
    plt.bar(tmp2.index.values,tmp2.values)
    for a, b in zip(tmp2.index.values, tmp2.values):
        plt.text(a, b, '%.0f' % b, ha='center', va='bottom', fontsize=8)
    plt.title('microwave')

    tmp3=pacifier.groupby(by='star_rating').count()['customer_id']
    plt.subplot(212)
    plt.ylim(0,14000)
    plt.bar(tmp3.index.values,tmp3.values)
    for a, b in zip(tmp3.index.values, tmp3.values):
        plt.text(a, b, '%.0f' % b, ha='center', va='bottom', fontsize=8)
    plt.title('pacifier')
    plt.show()
# fig_star_rating_count()

def fig_time():
    # print(hair_dryer.groupby('product_id').count()['customer_id'].describe())
    #
hair_dryer1=hair_dryer[hair_dryer['review_date']>pd.to_datetime('1/1/2013',format='
%m/%d/%Y')]
    y1=hair_dryer.groupby(['year','month']).count()['customer_id']
    y2 = microwave.groupby (['year','month']).count ()['customer_id']
    y3 = pacifier.groupby (['year','month']).count ()['customer_id']
    x=[]
    for i in range(2002,2016):
        for j in range(1,13):
            x.append((i,j))
    x.pop(-1)
    x.pop(-1)
    x.pop(-1)
    x.pop(-1)
    tmp=[]
    for i in x:
        if i in list(y1.index.values):
            tmp.append(y1.loc[i])
        else:
            tmp.append(0)
    y1=tmp
    tmp = []
    for i in x:
        if i in list(y2.index.values):
            tmp.append (y2[i])
        else:
            tmp.append (0)
    y2=tmp
    tmp = []
    for i in x:
        if i in list(y3.index.values):
            tmp.append (y3[i])
        else:
            tmp.append (0)
    y3=tmp

    x=[str(item[0])+'/'+str(item[1]) for item in x]
    plt.figure(figsize=(20,10))
    plt.plot(x,y1)
    plt.plot(x,y2)
    plt.plot(x,y3)
    plt.xticks (size='small', rotation=90, fontsize=8)
    plt.legend(['hair_dryer','microwave','pacifier'],loc = 'best')
    plt.show ()
    print (tmp)
fig_time()
# test1=hair_dryer[hair_dryer['product_id']=='B003V264WW']
# print()

# print(hair_dryer.info())
# print(microwave.info())
# print(pacifier.info())
#
# print(hair_dryer.describe())
# print(microwave.describe())
# print(pacifier.describe())
# #          :                    ,                                ,
helpful_votes/verified_purchase              ,
# print(hair_dryer[hair_dryer['verified_purchase']=='Y'].count())
```

```
print()
```

code	T1	remark	the code of question 1

```
from textblob import TextBlob
#
# text = "five stars".replace('.','')
# blob = TextBlob (text)
# #
# print ("blob        ")
# print (blob)
# print (blob.sentiment)
from sklearn.model_selection import train_test_split
import pandas as pd
import lightgbm as lgb
from tqdm import tqdm
import numpy as np
cat_cols = ['customer_id', 'review_id', 'product_id', 'vine', 'verified_purchase']
def pre_process(prod,data):
    del data['review_headline']
    dtime = pd.to_datetime (data['review_date'])
    v = (dtime.values - np.datetime64 ('2000-01-01T08:00:00Z')) / np.timedelta64 (1,
'ms')
    data['review_date'] = v

    data[cat_cols].astype('category')
    def map_value(x):
        x_set=set(x.values)
        dic={}
        n=0
        for i in x_set:
            if i not in dic:
                dic[i]=n
                n+=1
        new_x=[]
        for i in x.values:
            new_x.append(dic[i])
        return new_x
    for col in cat_cols:
        data[col] = map_value(data[col])
    def get_sentment(col):
        new_col=[]
        # for i in tqdm(col):
        #       out_put = emotion_eng.getMoodValue(i)
        #       new_col.append(out_put['all_value'])
        for i in tqdm(col):
            out_put = TextBlob (i)
```

```
                new_col.append(out_put.sentiment.polarity)
            return new_col
        data['review_body']=get_sentment(data['review_body'])

        def anylisis(data):
            not_pair = data[((data['star_rating'] == 1) & (data['review_body'] > 0.6)) |
((data['star_rating'] == 5) & (data['review_body'] < -0.6))]
            # print(not_pair)
            return not_pair.index.values

        abnormal_product = {}
        abnormal_product[prod] = (list (anylisis (data)))    # 8
        print(abnormal_product)
        # abnormal_product['microwave'] = (list (anylisis (microwave)))    # 3
        # abnormal_product['pacifier'] = (list (anylisis (pacifier)))    # 18
        data = data[~data.index.isin (abnormal_product[prod])]
        return data
hair_dryer=pd.read_csv('../Data/hair_dryer.csv',encoding='utf-8')
hair_dryer=hair_dryer.dropna()
hair_dryer=pre_process('hair_dryer',hair_dryer)
hair_dryer.to_csv('../Data/new_hair_dryer.csv')

microwave=pd.read_csv('../Data/microwave.csv',encoding='utf-8')
microwave=microwave.dropna()
microwave=pre_process('microwave',microwave)
microwave.to_csv('../Data/new_microwave.csv')

pacifier=pd.read_csv('../Data/pacifier.csv',encoding='utf-8')
pacifier=pacifier.dropna()
pacifier=pre_process('pacifier',pacifier)
pacifier.to_csv('../Data/new_pacifier.csv')

def get_X_y(prod,data):



        cols_x=['customer_id', 'review_id', 'product_id', 'helpful_votes', 'total_votes',
'vine', 'verified_purchase','review_body', 'review_date', 'year', 'month','rate']
        star=[]
        for i in data['star_rating']:
            if i<4:
                star.append(0)
            else:
                star.append(1)
        data['star_rating']=star
```

```
    X=data[cols_x]
    X[cat_cols]=X[cat_cols].astype('category')
    # X['customer_id']=X['customer_id'].astype('category')
    y=data['star_rating']

    # min_max_scaler = MinMaxScaler ()
    # X= min_max_scaler.fit_transform (X)
    # da=pd.DataFrame(X,columns=cols_x)
    return X,y
def gen_rate(data):
    tmp = data.groupby ('product_id').count ()['customer_id']
    sums = {}
    for i in tmp.index.values:
        sums[i] = tmp[i]
    rate = {}
    for i in sums:
        cnt = data[(data['product_id'] == i) & (data['star_rating'] < 4)].count ()[0]
        rate[i] = cnt / sums[i]
    rates = []
    for i in data['product_id'].values:
        rates.append (rate[i])
    data['rate'] = rates
    return data
hair_dryer=pd.read_csv('../Data/new_hair_dryer.csv',encoding='utf-8',index_col=0)
microwave=pd.read_csv('../Data/new_microwave.csv',encoding='utf-8',index_col=0)
pacifier=pd.read_csv('../Data/new_pacifier.csv',encoding='utf-8',index_col=0)
hair_dryer=gen_rate(hair_dryer)
microwave=gen_rate(microwave)
pacifier=gen_rate(pacifier)
def model1():
    X,y=get_X_y('hair_dryer',hair_dryer)
    print(len(y))
    X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2,
random_state=0,shuffle=True)
    print("Train data length:", len(X_train))
    print("Test data length:", len(X_test))
    print('          !')

    #        cv and train
    gbm = lgb.sklearn.LGBMClassifier(boosting_type='gbdt', num_leaves=64,
max_depth=-1, learning_rate=0.09, n_estimators=10, max_bin=255,
subsample_for_bin=200000, objective=None, min_split_gain=0.0,
min_child_weight=0.001, min_child_samples=20, subsample=1.0, subsample_freq=1,
colsample_bytree=1.0, reg_alpha=0.0, reg_lambda=0.0, random_state=None,
n_jobs=-1, silent=True)
    gbm.fit(X_train,y_train,sample_weight=None, init_score=None,
                eval_set=None, eval_names=None, eval_sample_weight=None,
                eval_class_weight=None,                eval_init_score=None,
eval_metric=None,
                early_stopping_rounds=None, verbose=True,
                feature_name='auto', categorical_feature='auto', callbacks=None)
    print ('Start predicting...')
    #
    y_pred = gbm.predict (X_test)
    from sklearn.metrics import precision_score, recall_score, roc_auc_score
    print('importance:',list(zip(X_train.columns.values,gbm.feature_importances_)))
    precision=precision_score(y_test, y_pred)
    recall=recall_score(y_test, y_pred)
    print(list(zip(y_test.values,y_pred)))
    print ('          ', precision)
    print ('          ', recall)
    print ('auc       ', roc_auc_score (y_test, y_pred))
    print ('F1        ', 2 * (precision * recall) / (precision + recall))
def model2():
    X, y = get_X_y ('microwave',microwave)
    print (len (y))
    X_train,   X_test,   y_train,   y_test   =   train_test_split   (X,   y,   test_size=0.2,
random_state=0, shuffle=True)
    print ("Train data length:", len (X_train))
    print ("Test data length:", len (X_test))
    print ('          !')

    #        cv and train
    gbm   =   lgb.sklearn.LGBMClassifier   (boosting_type='gbdt',   num_leaves=64,
max_depth=-1, learning_rate=0.1,
                              n_estimators=10,          max_bin=255,
subsample_for_bin=200000, objective=None,
                              min_split_gain=0.0,
min_child_weight=0.001, min_child_samples=20, subsample=1.0,
                              subsample_freq=1,
colsample_bytree=1.0, reg_alpha=0.0, reg_lambda=0.0,
                              random_state=None,          n_jobs=-1,
silent=True)
    gbm.fit (X_train, y_train, sample_weight=None, init_score=None,
            eval_set=None, eval_names=None, eval_sample_weight=None,
            eval_class_weight=None, eval_init_score=None, eval_metric=None,
            early_stopping_rounds=None, verbose=True,
            feature_name='auto', categorical_feature='auto', callbacks=None)
```

```
	print ('Start predicting...')
	#
	y_pred = gbm.predict (X_test)
	from sklearn.metrics import precision_score, recall_score, roc_auc_score
	print ('importance:', list (zip (X_train.columns.values, 
gbm.feature_importances_)))
	precision = precision_score (y_test, y_pred)
	recall = recall_score (y_test, y_pred)
	print (list (zip (y_test.values, y_pred)))
	print ('            ', precision)
	print ('            ', recall)
	print ('auc      ', roc_auc_score (y_test, y_pred))
	print ('F1       ', 2 * (precision * recall) / (precision + recall))
def model3():
	X, y = get_X_y ('pacifier',pacifier)
	print (len (y))
	X_train, X_test, y_train, y_test = train_test_split (X, y, test_size=0.2, 
random_state=0, shuffle=True)
	print ("Train data length:", len (X_train))
	print ("Test data length:", len (X_test))
	print ('            !')



	gbm = lgb.sklearn.LGBMClassifier (boosting_type='gbdt', num_leaves=64, 
max_depth=-1, learning_rate=0.09,
                                    n_estimators=10, max_bin=255, 
subsample_for_bin=200000, objective=None,
                                    min_split_gain=0.0, 
min_child_weight=0.001, min_child_samples=20, subsample=1.0,
                                    subsample_freq=1, 
colsample_bytree=1.0, reg_alpha=0.0, reg_lambda=0.0,
                                    random_state=None, n_jobs=-1, 
silent=True)
	gbm.fit (X_train, y_train, sample_weight=None, init_score=None,
	        eval_set=None, eval_names=None, eval_sample_weight=None,
	        eval_class_weight=None, eval_init_score=None, eval_metric=None,
	        early_stopping_rounds=None, verbose=True,
	        feature_name='auto', categorical_feature='auto', callbacks=None)
	print ('Start predicting...')
	#
	y_pred = gbm.predict (X_test)
	from sklearn.metrics import precision_score, recall_score, roc_auc_score
	print ('importance:', list (zip (X_train.columns.values, 
gbm.feature_importances_)))
```

```
    precision = precision_score (y_test, y_pred)
    recall = recall_score (y_test, y_pred)
    print (list (zip (y_test.values, y_pred)))
    print ('              ', precision)
    print ('              ', recall)
    print ('auc       ', roc_auc_score (y_test, y_pred))
    print ('F1       ', 2 * (precision * recall) / (precision + recall))
# hair_dryer
model1()
#microwave
model2()
#pacifier
model3()
```

code	T2_a	remark	the code of question 2-a

```
from sklearn.decomposition import pca
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import numpy as np
import pandas as pd
import math
from textblob import TextBlob
# blob = TextBlob ("text")
# print(blob.sentiment.polarity)
# out_put = emotion_eng.getMoodValue("great")#out_put['all_value']
# all_low=hair_dryer[(hair_dryer['star_rating']<2) & (hair_dryer['review_body']<-0.6)]
# all_high=hair_dryer[(hair_dryer['star_rating']>3) & (hair_dryer['review_body']>0.2)]
# all_mid=hair_dryer[(hair_dryer['star_rating']>=2) & (hair_dryer['star_rating']<=3) & (0.2>=hair_dryer['review_body']) & (hair_dryer['review_body']>=-0.6)]
# not_pair=hair_dryer[((hair_dryer['star_rating']<2) & (hair_dryer['review_body']>0.8)) | ((microwave['star_rating']==5) & (hair_dryer['review_body']<-0.6))]
#
# a=all_low.count()['star_rating']
# b=all_high.count()['star_rating']
# c=all_mid.count()['star_rating']
# d=hair_dryer.count()['star_rating']
# e=not_pair.count()['star_rating']
# print(a,b,c,e,d-a-b-c)
# print()
# all_low=microwave[(microwave['star_rating']<2) & (microwave['review_body']<-0.6)]
# all_high=microwave[(microwave['star_rating']>3) &
(microwave['review_body']>0.2)]
# all_mid=microwave[(microwave['star_rating']>=2) & (microwave['star_rating']<=3)
& (0.2>=microwave['review_body']) & (microwave['review_body']>=-0.6)]
#                    not_pair=microwave[((microwave['star_rating']<2)              &
(microwave['review_body']>0.8))        |        ((microwave['star_rating']==5)        &
(microwave['review_body']<-0.6))]
#
# a=all_low.count()['star_rating']
# b=all_high.count()['star_rating']
# c=all_mid.count()['star_rating']
# d=microwave.count()['star_rating']
# e=not_pair.count()['star_rating']
# print(a,b,c,e,d-a-b-c)
# print()
# all_low=pacifier[(pacifier['star_rating']<2) & (pacifier['review_body']<-0.6)]
# all_high=pacifier[(pacifier['star_rating']>3) & (pacifier['review_body']>0.2)]
#   all_mid=pacifier[(pacifier['star_rating']>=2)   &   (pacifier['star_rating']<=3)   &
(0.2>=pacifier['review_body']) & (pacifier['review_body']>=-0.6)]
#
#   not_pair=pacifier[((pacifier['star_rating']<2)   &   (pacifier['review_body']>0.8))   |
((pacifier['star_rating']==5) & (pacifier['review_body']<-0.6))]
# a=all_low.count()['star_rating']
# b=all_high.count()['star_rating']
# c=all_mid.count()['star_rating']
# d=pacifier.count()['star_rating']
# e=not_pair.count()['star_rating']
# print(a,b,c,e,d-a-b-c)
hair_dryer=pd.read_csv('../Data/new_hair_dryer.csv',encoding='utf-8',index_col=0)
microwave=pd.read_csv('../Data/new_microwave.csv',encoding='utf-8',index_col=0)
pacifier=pd.read_csv('../Data/new_pacifier.csv',encoding='utf-8',index_col=0)
def anylisis(data):
        all_low=data[(data['star_rating']<2) & (data['review_body']<-0.6)]
        all_high=data[(data['star_rating']>3) & (data['review_body']>0.2)]
        all_mid=data[(data['star_rating']>=2)        &        (data['star_rating']<=3)        &
(0.2>=data['review_body']) & (data['review_body']>=-0.6)]
        # not_pair                    1        5
        not_pair=data[((data['star_rating']==1)       &       (data['review_body']>0.6))       |
((data['star_rating']==5) & (data['review_body']<-0.6))]
        a=all_low.count()['star_rating']
        b=all_high.count()['star_rating']
        c=all_mid.count()['star_rating']
        d=data.count()['star_rating']
        e=not_pair.count()['star_rating']
        # print(a,b,c,e,d-a-b-c)
```

```
    # print('          1        5                    :',e)
    return not_pair.index.values
abnormal_product={}
abnormal_product['hair_dryer']=(list(anylisis(hair_dryer)))#8
abnormal_product['microwave']=(list(anylisis(microwave)))#3
abnormal_product['pacifier']=(list(anylisis(pacifier)))#18
print(abnormal_product)
def scaler(X):
    """

    """
    min_max_scaler = MinMaxScaler ()
    x_train= min_max_scaler.fit_transform (X)
    x=pd.DataFrame(x_train,columns=X.columns.values)
    return x
def cal_weight(x):
    '''                                  '''
    #
    x = x.apply (lambda x: ((x - np.min (x)) / (np.max (x) - np.min (x))))

    #     k
    rows = x.index.size    #
    cols = x.columns.size    #
    k = 1.0 / math.log (rows)

    lnf = [[None] * cols for i in range (rows)]

    #               --
    #
    # p=array(p)
    x = np.array (x)
    lnf = [[None] * cols for i in range (rows)]
    lnf = np.array (lnf)
    for i in range (0, rows):
        for j in range (0, cols):
            if x[i][j] == 0:
                lnfij = 0.0
            else:
                p = x[i][j] / x.sum (axis=0)[j]
                lnfij = math.log (p) * p * (-k)
            lnf[i][j] = lnfij
    lnf = pd.DataFrame (lnf)
    E = lnf
    #
    d = 1 - E.sum (axis=0)
    #
    w = [[None] * 1 for i in range (cols)]
    for j in range (0, cols):
        wj = d[j] / sum (d)
        w[j] = wj
        #                    ,

    w = pd.DataFrame (w)
    return w
def get_eval(prod,data):
    data=data[~data['product_id'].isin(abnormal_product[prod])]
    x=data[['star_rating','review_body']]
    # x=scaler(x)
    w = cal_weight (x)   #      cal_weight
    w.index = x.columns
    w.columns = ['weight']

wei={'star_rating':w.loc['star_rating','weight'],'review_body':w.loc['review_body','wei
ght']}
    return wei
#   wei=get_eval('hair_dryer',hair_dryer)   #{'star_rating':   0.8529774897515476,
'review_body': 0.1470225102484523}
# print(wei)

def gen_score(prod,data):
    x=data['star_rating'].values
    y=data['review_body'].values
    wei = get_eval (prod, data)
    print(prod,wei)
    score=np.array(x)*wei['star_rating']+np.array(y)*wei['review_body']
    data['score']=score
    return data
data1=gen_score('hair_dryer',hair_dryer)
data2=gen_score('microwave',microwave)
data3=gen_score('pacifier',pacifier)
print(data1.head())
print(data2.head())
print(data3.head())
```

code	T2_b	remark	the code of question 2-b

```
from sklearn.decomposition import pca
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

```
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import classification_report

from textblob import TextBlob
# blob = TextBlob ("text")
# print(blob.sentiment.polarity)
# out_put = emotion_eng.getMoodValue("great")#out_put['all_value']
# all_low=hair_dryer[(hair_dryer['star_rating']<2) & (hair_dryer['review_body']<-
0.6)]
# all_high=hair_dryer[(hair_dryer['star_rating']>3) &
(hair_dryer['review_body']>0.2)]
# all_mid=hair_dryer[(hair_dryer['star_rating']>=2) & (hair_dryer['star_rating']<=3) &
(0.2>=hair_dryer['review_body']) & (hair_dryer['review_body']>=-0.6)]
# not_pair=hair_dryer[((hair_dryer['star_rating']<2) &
(hair_dryer['review_body']>0.8)) | ((microwave['star_rating']==5) &
(hair_dryer['review_body']<-0.6))]
#
# a=all_low.count()['star_rating']
# b=all_high.count()['star_rating']
# c=all_mid.count()['star_rating']
# d=hair_dryer.count()['star_rating']
# e=not_pair.count()['star_rating']
# print(a,b,c,e,d-a-b-c)
# print()
# all_low=microwave[(microwave['star_rating']<2) & (microwave['review_body']<-
0.6)]
# all_high=microwave[(microwave['star_rating']>3) &
(microwave['review_body']>0.2)]
# all_mid=microwave[(microwave['star_rating']>=2) & (microwave['star_rating']<=3)
& (0.2>=microwave['review_body']) & (microwave['review_body']>=-0.6)]
# not_pair=microwave[((microwave['star_rating']<2) &
(microwave['review_body']>0.8)) | ((microwave['star_rating']==5) &
(microwave['review_body']<-0.6))]
#
# a=all_low.count()['star_rating']
# b=all_high.count()['star_rating']
# c=all_mid.count()['star_rating']
# d=microwave.count()['star_rating']
# e=not_pair.count()['star_rating']
# print(a,b,c,e,d-a-b-c)
# print()
# all_low=pacifier[(pacifier['star_rating']<2) & (pacifier['review_body']<-0.6)]
# all_high=pacifier[(pacifier['star_rating']>3) & (pacifier['review_body']>0.2)]
#   all_mid=pacifier[(pacifier['star_rating']>=2)   &   (pacifier['star_rating']<=3)   &
(0.2>=pacifier['review_body']) & (pacifier['review_body']>=-0.6)]
#
#   not_pair=pacifier[((pacifier['star_rating']<2)   &   (pacifier['review_body']>0.8))   |
((pacifier['star_rating']==5) & (pacifier['review_body']<-0.6))]
# a=all_low.count()['star_rating']
# b=all_high.count()['star_rating']
# c=all_mid.count()['star_rating']
# d=pacifier.count()['star_rating']
# e=not_pair.count()['star_rating']
# print(a,b,c,e,d-a-b-c)
hair_dryer=pd.read_csv('../Data/new_hair_dryer.csv',encoding='utf-8',index_col=0)
microwave=pd.read_csv('../Data/new_microwave.csv',encoding='utf-8',index_col=0)
pacifier=pd.read_csv('../Data/new_pacifier.csv',encoding='utf-8',index_col=0)
def gen_rate(data):
    tmp = data.groupby ('product_id').count ()['customer_id']
    sums = {}
    for i in tmp.index.values:
        sums[i] = tmp[i]
    rate = {}
    for i in sums:
        cnt = data[(data['product_id'] == i) & (data['star_rating'] < 4)].count ()[0]
        rate[i] = cnt / sums[i]
    rates = []
    for i in data['product_id'].values:
        rates.append (rate[i])
    data['rate'] = rates
    return data
hair_dryer=gen_rate(hair_dryer)
microwave=gen_rate(microwave)
pacifier=gen_rate(pacifier)

def anylisis(data):
    all_low=data[(data['star_rating']<2) & (data['review_body']<-0.6)]
    all_high=data[(data['star_rating']>3) & (data['review_body']>0.2)]
    all_mid=data[(data['star_rating']>=2)      &      (data['star_rating']<=3)      &
(0.2>=data['review_body']) & (data['review_body']>=-0.6)]
    # not_pair              1       5
    not_pair=data[((data['star_rating']==1)     &     (data['review_body']>0.6))     |
((data['star_rating']==5) & (data['review_body']<-0.6))]
    a=all_low.count()['star_rating']
    b=all_high.count()['star_rating']
    c=all_mid.count()['star_rating']
    d=data.count()['star_rating']
    e=not_pair.count()['star_rating']
    # print(a,b,c,e,d-a-b-c)
    # print('        1      5                    :',e)
    return not_pair.index.values
abnormal_product={}
abnormal_product['hair_dryer']=(list(anylisis(hair_dryer)))#8
abnormal_product['microwave']=(list(anylisis(microwave)))#3
abnormal_product['pacifier']=(list(anylisis(pacifier)))#18
print(abnormal_product)
def scaler(X):
    """


    """
    min_max_scaler = MinMaxScaler ()
    x_train= min_max_scaler.fit_transform (X)
    x=pd.DataFrame(x_train,columns=X.columns.values)
    return x
def cal_weight(x):
    '''                               '''
    #
    x = x.apply (lambda x: ((x - np.min (x)) / (np.max (x) - np.min (x))))

    #    k
    rows = x.index.size    #
    cols = x.columns.size    #
    k = 1.0 / math.log (rows)

    lnf = [[None] * cols for i in range (rows)]

    #              --
    #
    # p=array(p)
    x = np.array (x)
    lnf = [[None] * cols for i in range (rows)]
    lnf = np.array (lnf)
    for i in range (0, rows):
        for j in range (0, cols):
            if x[i][j] == 0:
                lnfij = 0.0
            else:
                p = x[i][j] / x.sum (axis=0)[j]
                lnfij = math.log (p) * p * (-k)
            lnf[i][j] = lnfij
    lnf = pd.DataFrame (lnf)
    E = lnf

    #
    d = 1 - E.sum (axis=0)
    #
    w = [[None] * 1 for i in range (cols)]
    for j in range (0, cols):
        wj = d[j] / sum (d)
        w[j] = wj
        #                                          ,

    w = pd.DataFrame (w)
    return w
def get_eval(prod,data):

    data=data[~data['product_id'].isin(abnormal_product[prod])]
    x=data[['star_rating','review_body']]
    # x=scaler(x)
    w = cal_weight (x)    #          cal_weight
    w.index = x.columns
    w.columns = ['weight']

wei={'star_rating':w.loc['star_rating','weight'],'review_body':w.loc['review_body','wei
ght']}
    return wei
#     wei=get_eval('hair_dryer',hair_dryer)     #{'star_rating':     0.8529774897515476,
'review_body': 0.1470225102484523}
# print(wei)
def gen_score(prod,data):
    x=data['star_rating'].values
    y=data['review_body'].values
    wei = get_eval (prod, data)
    score=np.array(x)*wei['star_rating']+np.array(y)*wei['review_body']
    data['score']=score
    return data

def fig(prod, D):
    data=gen_score(prod,D)[['review_date','year','month','score']]
    # print(data.describe())
```

```
    good=data[(data['score']>4)                                              &
(data['year']>2009)].groupby(['year','month']).count()['score']
    bad=data[(data['score']<1)                                               &
(data['year']>2009)].groupby(['year','month']).count()['score']
    all_of=data[(data['year']>2009)].groupby(['year','month']).count()['review_date']
    good=pd.DataFrame(good,index=good.index.values)
    bad=pd.DataFrame(bad,index=bad.index.values)
    all_of=pd.DataFrame(all_of,index=all_of.index.values)
    bad.rename(columns={'score':'score_bad'},inplace=True)
    x=[str(i[0])+'/'+str(i[1]) for i in good.index.values]
    good['time']=x
    x=[str(i[0])+'/'+str(i[1]) for i in bad.index.values]
    bad['time']=x
    x=[str(i[0])+'/'+str(i[1]) for i in all_of.index.values]
    all_of['time']=x
    all=pd.merge(good,bad,how='left')
    all=pd.merge(all,all_of,how='left')
    all.fillna(0)
    fig = plt.figure(num=1, figsize=(15, 8),dpi=80)
    plt.plot(all['time'].values,all['score'].values/all['review_date'].values)
    # plt.show()
    plt.plot(all['time'].values,all['score_bad'].values/all['review_date'].values)
    # plt.plot(all['time'].values,all['review_date'].values)
    plt.legend(['good','bad'],loc = 'best')

    plt.xticks (size='small', rotation=90, fontsize=13)
    plt.show()

# fig('hair_dryer',hair_dryer)
# fig('microwave',microwave)
# fig('pacifier',pacifier)

def classify(prod,data):
    data=gen_score(prod, data)
    cols_x = ['helpful_votes', 'total_votes', 'verified_purchase', 'review_body',
'review_date', 'month', 'rate','score']
    x=data[cols_x]
    scores=data['star_rating'].values
    y=[]
    for score in scores:
        if score>=4:
            y.append(1)
        else:
            y.append(0)
    X_train,    X_test,    y_train,    y_test    =
train_test_split(x,y,test_size=0.2,random_state=0)
    ss = StandardScaler ()
    X_train = ss.fit_transform (X_train)
    X_test = ss.fit_transform (X_test)
    lr = LogisticRegression()
    lr.fit (X_train, y_train)
    lr_y_predict = lr.predict (X_test)
    print(lr_y_predict)
    print ('Accuracy of LR Classifier:', lr.score (X_test, y_test))

    print()
classify('hair_dryer',hair_dryer)
print()
```

code	T2_c	remark	the code of question 2-c

code	T2_d	remark	the code of question 2-d

```
import   pandas   as   pd

del_cols                                                              =
[&apos;customer_id&apos;,&apos;review_id&apos;,&apos;product_id&apos;,&apos
;helpful_votes&apos;,&apos;total_votes&apos;,&apos;vine&apos;,&apos;verified_p
urchase&apos;,&apos;review_date&apos;,&apos;year&apos;,&apos;month&apos;]

hair_dryer=pd.read_csv(&apos;../Data/new_hair_dryer.csv&apos;,encoding=&apos;u
tf-8&apos;,index_col=0)
hair_dryer=hair_dryer.drop(del_cols,axis=1)
microwave=pd.read_csv(&apos;../Data/new_microwave.csv&apos;,encoding=&apos;
utf-8&apos;,index_col=0)
microwave=microwave.drop(del_cols,axis=1)
pacifier=pd.read_csv(&apos;../Data/new_pacifier.csv&apos;,encoding=&apos;utf-
8&apos;,index_col=0)
pacifier=pacifier.drop(del_cols,axis=1)
print(&apos;hair_dryer\n&apos;,hair_dryer.corr(&apos;spearman&apos;))
print()
print(&apos;microwave\n&apos;,microwave.corr(&apos;spearman&apos;))
print()
print(&apos;pacifier\n&apos;,pacifier.corr(&apos;spearman&apos;))
print()
```

code	T2_e	remark	the code of question 2-e

```

import collections
import pickle
import numpy as np
import matplotlib.pyplot as plt
import jieba.analyse
import seaborn as sns
from tqdm import tqdm

from my_util import pre_process
import pandas as pd
import wordcloud
#       TF - IDF   jieba.analyse.extract_tags (sentence, topK=20, withWeight=False,
allowPOS=())
#        TextRank    jieba.analyse.textrank (sentence, topK=20, withWeight=False,
allowPOS=('ns', 'n', 'vn', 'v'))
hair_dryer=pd.read_csv('../Data/hair_dryer.csv',encoding='utf-8')
microwave=pd.read_csv('../Data/microwave.csv',encoding='utf-8')
pacifier=pd.read_csv('../Data/pacifier.csv',encoding='utf-8')
hair_dryer = hair_dryer.dropna ()
microwave = microwave.dropna ()
pacifier = pacifier.dropna ()
def try1():
    def gen_star_sent(n):
        tmp1=hair_dryer[hair_dryer['star_rating']==n]['review_body']
        tmp2=microwave[microwave['star_rating']==n]['review_body']
        tmp3=pacifier[pacifier['star_rating']==n]['review_body']
        star_str=''
        for i in tqdm(tmp1.values):
            for j in pre_process(i):
                star_str=star_str+' '+j
        for i in tqdm(tmp2.values):
            for j in pre_process (i):
                star_str = star_str + ' ' + j
        for i in tqdm(tmp3.values):
            for j in pre_process (i):
                star_str = star_str + ' ' + j
        print()
        return star_str
    one_star_sen=gen_star_sent(1)
    two_star_sen=gen_star_sent(2)
    three_star_sen=gen_star_sent(3)
    four_star_sen=gen_star_sent(4)
    five_star_sen=gen_star_sent(5)
    #
one_star_sen=hair_dryer[hair_dryer['star_rating']==1]['review_body']+microwave[mi
crowave['star_rating']==1]['review_body']+pacifier[pacifier['star_rating']==1]['review
_body']
    #       keywords=jieba.analyse.extract_tags(one_star_sen,       topK=20,
withWeight=False, allowPOS=())
    # print(keywords)
    w = wordcloud.WordCloud(max_words=50)
    w.generate(one_star_sen)
    w.to_file('output1.png')

    #       keywords=jieba.analyse.extract_tags(two_star_sen,       topK=20,
withWeight=False, allowPOS=())
    # print(keywords)
    w = wordcloud.WordCloud(max_words=50)
    w.generate(two_star_sen)
    w.to_file('output2.png')

    keywords=jieba.analyse.extract_tags(three_star_sen,           topK=20,
withWeight=False, allowPOS=())
    print(keywords)
    w = wordcloud.WordCloud(max_words=50)
    w.generate(three_star_sen)
    w.to_file('output3.png')

    #       keywords=jieba.analyse.extract_tags(four_star_sen,      topK=20,
withWeight=False, allowPOS=())
    # print(keywords)
    w = wordcloud.WordCloud(max_words=50)
    w.generate(four_star_sen)
    w.to_file('output4.png')

    #       keywords=jieba.analyse.extract_tags(five_star_sen,      topK=20,
withWeight=False, allowPOS=())
    # print(keywords)
    w = wordcloud.WordCloud(max_words=50)
    w.generate(five_star_sen)
    w.to_file('output5.png')

def try2():
    # words_list=set()
    # with open('emotion_dict/words_list.txt','r',encoding='utf-8') as f:
    #     for line in f:
    #         words_list.add(line.replace('\n',''))
    # def gen_star_sent(n):
    #     tmp1 = hair_dryer[hair_dryer['star_rating'] == n]['review_body']
    #        tmp2 = microwave[microwave['star_rating'] == n]['review_body']
    #        tmp3 = pacifier[pacifier['star_rating'] == n]['review_body']
    #
    #        star_str = ''
    #        for i in tqdm(tmp1.values):
    #              for j in pre_process(i):
    #                    if j in words_list:
    #                          star_str = star_str + ' ' + j
    #        for i in tqdm(tmp2.values):
    #              for j in pre_process (i):
    #                    if j in words_list:
    #                          star_str = star_str + ' ' + j
    #        for i in tqdm(tmp3.values):
    #              for j in pre_process (i):
    #                    if j in words_list:
    #                          star_str = star_str + ' ' + j
    #        print()
    #
    #        return star_str
    #
    # one_star_sen = gen_star_sent (1)
    # two_star_sen = gen_star_sent (2)
    # three_star_sen = gen_star_sent (3)
    # four_star_sen = gen_star_sent (4)
    # five_star_sen = gen_star_sent (5)
    # star_sent = {}
    # star_sent['one'] = one_star_sen
    # star_sent['two'] = two_star_sen
    # star_sent['three'] = three_star_sen
    # star_sent['four'] = four_star_sen
    # star_sent['five'] = five_star_sen
    # pickle.dump (star_sent, open ('star_sent_cloud.pkl', 'wb'))
    star_sent=pickle.load(open('star_sent_cloud.pkl','rb'))

    w = wordcloud.WordCloud (max_words=50)
    w.generate (star_sent['one'])
    w.to_file ('output1.png')


    w = wordcloud.WordCloud (max_words=50)
    w.generate (star_sent['two'])
    w.to_file ('output2.png')

    w = wordcloud.WordCloud (max_words=50)
```

```
    w.generate (star_sent['three'])
    w.to_file ('output3.png')

    w = wordcloud.WordCloud (max_words=50)
    w.generate (star_sent['four'])
    w.to_file ('output4.png')

    w = wordcloud.WordCloud (max_words=50)
    w.generate (star_sent['five'])
    w.to_file ('output5.png')

def try3(num):
    words_list = set ()
    with open ('emotion_dict/words_list.txt', 'r', encoding='utf-8') as f:
        for line in f:
            words_list.add (line.replace ('\n', ''))

    def gen_star_sent1(n):
        tmp1 = hair_dryer[hair_dryer['star_rating'] == n]['review_body']
        tmp2 = microwave[microwave['star_rating'] == n]['review_body']
        tmp3 = pacifier[pacifier['star_rating'] == n]['review_body']

        star_str = []
        for i in tqdm (tmp1.values):
            for j in pre_process (i):
                if j in words_list:
                    star_str .append(j)
        for i in tqdm (tmp2.values):
            for j in pre_process (i):
                if j in words_list:
                    star_str .append(j)
        for i in tqdm (tmp3.values):
            for j in pre_process (i):
                if j in words_list:
                    star_str .append(j)
        print()

        return star_str

    # one_star_sen = gen_star_sent1 (1)
    # two_star_sen = gen_star_sent1 (2)
    # three_star_sen = gen_star_sent1 (3)
    # four_star_sen = gen_star_sent1 (4)
    five_star_sen = gen_star_sent1 (5)
```

```
# star_sent={}
# star_sent['one']=one_star_sen
# star_sent['two']=two_star_sen
# star_sent['three']=three_star_sen
# star_sent['four']=four_star_sen
# star_sent['five']=five_star_sen
# pickle.dump(star_sent,open('star_sent_count.pkl','wb'))
star_sent=pickle.load(open('star_sent_count.pkl','rb'))
words_set=[]
words_dict={}

word_counts = collections.Counter (star_sent['five'])    #
word_counts_top20 = word_counts.most_common (num)    #             20
# print (word_counts_top20)    #
word_counts_top20 = dict (word_counts_top20)
words_set += list (word_counts_top20.keys ())
words_dict['5'] = word_counts_top20
y5 = list(word_counts_top20.values())
x5 = [5 for _ in range (len (y5))]

word_counts = collections.Counter (star_sent['four'])    #
word_counts_top20 = word_counts.most_common (num)    #             20
# print (word_counts_top20)    #
word_counts_top20 = dict (word_counts_top20)
words_set += list (word_counts_top20.keys ())
words_dict['4'] = word_counts_top20
y4 = list(word_counts_top20.values())
x4 = [4 for _ in range (len (y4))]

word_counts = collections.Counter (star_sent['three'])    #
word_counts_top20 = word_counts.most_common (num)    #             20
# print (word_counts_top20)    #
word_counts_top20 = dict (word_counts_top20)
words_set += list (word_counts_top20.keys ())
words_dict['3'] = word_counts_top20
y3 = list(word_counts_top20.values())
x3 = [3 for _ in range (len (y3))]

word_counts = collections.Counter (star_sent['two'])    #
word_counts_top20 = word_counts.most_common (num)    #             20
# print (word_counts_top20)    #
word_counts_top20 = dict (word_counts_top20)
words_set += list (word_counts_top20.keys ())
words_dict['2'] = word_counts_top20
```

```
    y2 = list(word_counts_top20.values())
    x2 = [2 for _ in range (len (y2))]

    word_counts = collections.Counter (star_sent['one'])   #
    word_counts_top20 = word_counts.most_common (num)   #            20
    word_counts_top20=dict(word_counts_top20)
    words_set+=list(word_counts_top20.keys())
    words_dict['1']=word_counts_top20
    # print (word_counts_top20)   #
    y1 = list(word_counts_top20.values())
    x1=[1 for _ in range(len(y1))]

    words_set=set(words_set)
    sorted(words_set)
    dic={'1':[],'2':[],'3':[],'4':[],'5':[]}
    for i in words_set:
        for j in range(1,6):
            if i in words_dict[str(j)].keys():
                dic[str(j)].append(words_dict[str(j)][i])
            else:
                dic[str(j)].append(0)

    data=pd.DataFrame(dic,index=words_set)
    data.to_csv('../Data/word_count'+str(num)+'.csv',encoding='utf-8')
    cmap = sns.cubehelix_palette (start=1.5, rot=3, gamma=0.8, as_cmap=True)
    sns.heatmap (data,linewidths = 0.05, vmax=5000, vmin=50, cmap=cmap)
    plt.show()
    plt.savefig('words_hot'+str(num)+'.png')


try3(10)
# try2()
```