Problem Chosen	2020 MCM/ICM Summary Sheet	Team Control Number
C		**2021473**

Summary

In the online marketplace created by Amazon, customers have the opportunities to rate and review products, which in turn affects other customers' purchasing decisions.

In this paper, we establish a Useful Review Sorting Model and an ARIMA-BP Model to identify the patterns and relationships within and between star ratings, reviews, helpfulness ratings, and sales in the online marketplace of microwave oven, baby pacifier, and hair dryer.

First, we analyze the three product data sets to describe the patterns and relationships between the indicators. we find the most popular brands in the three markets are Conair, Wubbanub, Whirlpool. Besides, the length of text and $help_votes$ are positively correlated. Afterwards, we construct a LSTM model for reviews emotion analysis. It classifies reviews into 5 classifications. Then we discuss the association between star rating and reputation and reviewing motivation and find two linear regression models. The reputation of reviews are evaluated by LSTM model's prediction results.

Next, we build a Useful Review Sorting model to identify the most informative reviews in each market. The Useful Review Sorting model takes into full account of text-based and rating-based information, including helpfulness rating, timeliness, depth, attributes words, intensity of emotional expression and modifiers of the reviews and ratings.

Further, We establish the an ARIMA model and a BP-Nutural Network Model, which are used to capture a product's time-based and text-based and rating-based patterns in each market, respectively.

Finally, we formulate an ARIMA-BP model by combing the ARIMA model and the BP-neural network model to determine whether a product is successful or failing . Based on the results of the Useful Review Sorting model and the ARIMA-BP model, we propose an online sales strategy and identify some potentially important design features that would enhance product desirability.

Keywords: Ratings and Reviews; LSTM model; ARIMA-BP model; ARIMA-BP model

A Wealth of Reviews - Optimized Marketing Strategy and Design Scheme Based on LSTM and ARIMA-BP Model

March 13, 2020

Summary

In the online marketplace created by Amazon, customers have the opportunities to rate and review products, which in turn affects other customers' purchasing decisions.

In this paper, we establish a Useful Review Sorting Model and an ARIMA-BP Model to identify the patterns and relationships within and between star ratings, reviews, helpfulness ratings, and sales in the online marketplace of microwave oven, baby pacifier, and hair dryer.

First, we analyze the three product data sets to describe the patterns and relationships between the indicators. we find the most popular brands in the three markets are Conair, Wubbanub, Whirlpool. Besides, the length of text and $help_votes$ are positively correlated. Afterwards, we construct a LSTM model for reviews emotion analysis. It classifies reviews into 5 classifications. Then we discuss the association between star rating and reputation and reviewing motivation and find two linear regression models. The reputation of reviews are evaluated by LSTM model's prediction results.

Next, we build a Useful Review Sorting model to identify the most informative reviews in each market. The Useful Review Sorting model takes into full account of text-based and rating-based information, including helpfulness rating, timeliness, depth, attributes words, intensity of emotional expression and modifiers of the reviews and ratings.

Further, We establish the an ARIMA model and a BP-Nutural Network Model, which are used to capture a product's time-based and text-based and rating-based patterns in each market, respectively.

Finally, we formulate an ARIMA-BP model by combing the ARIMA model and the BP-neural network model to determine whether a product is successful or failing . Based on the results of the Useful Review Sorting model and the ARIMA-BP model, we propose an online sales strategy and identify some potentially important design features that would enhance product desirability.

Keywords: Ratings and Reviews; LSTM model; ARIMA-BP model; ARIMA-BP model

Contents

To: Marketing Director of Sunshine Company
From: Team # 2021473 ,MCM 2020
Date: March 9, 2020
Subject: Usage of Cleaner, Renewable Energy Sources

Honorable Marketing Director of Sunshine Company,

We propose to analyze the three product data files of ratings and reviews from 2013 to 2015 for you, and summarize the key patterns and relationships embedded in them, give our recommended sales strategy and identified popular design features.

Summary for the files:

First, we find there is a positive correlation between positive reviews and sales.The top 5 products with highest sales in these three markets all have favorable ratio of more than 60%, some even more than 80%. Simultaneously, customers giving star_ratings of 1 and 5 are more likely to write a review. Besides, the more detailed the content is, the more help_votes the reviews obtain.

Furthermore, vine reviews tend to get more help_votes. We also observe that, the reviews of certain products are concentrated, which helps to predict peak period.

Sales Strategy: Based on the above analysis, we explore the following strategies to help our products make a hit:

1. Promote products with key points embedded in reviews getting the highest favorable ratio. For instance, the key words of hair_dryer are velocity and styling effect, and pacifiers' highlights are durability, anti-lost and soft material.

2. Participate in Amazon Vine Program to quickly attract attention and develop brand recognition. Users are very sensitive to microwaves, hair dryers and other appliances.

3. Pay attention to consumers' reviews and make a quick response to negative reviews.

4. Intensify promotion at the peak period, while hair_dryers are most popular in the first quarter.

Design Focus: Aiming to satisfy customer's demand, we recommend the following four key points in designation.

1. In the hair_dryer market, high velocity, low noise, good durability, light weight, qualified after-sales service is the pursuit of the users. Apart from these, poor after-sales service is the main point in negative reviews.

2. In the pacifiers market, the customers emphasize on doll-pacifier integrated design, easiness to find and wash, material,etc. Among them, doll pacifier integrated design best meets the core needs - to appease the baby's mood, exercise their ability to suck.

3. In the microwave oven market, the best-selling product have some common features, like black or white appearance and stainless steel material. Customers also focus on the appropriate occupied space and comprehensive function. Besides, improve stability and avoid program errors are critical to avoid a poor star rating.

4. The most popular brands in three markets are Conair, Wubbanub, Whirlpool. Their design features of them can be referred to.

Wish our advice can inspire you in pursuit of a brilliant succeed in the marketplace. We We are looking forward to your achievement.
Yours sincerely
MCM 2019 Team 2021473

1 Introduction

1.1 Restatement of the problem

As the largest online e-commerce company in the country, Amazon collects mountains of ratings data and reviews from customers which contribute to identify desired product designation. Therefore, the feedback there is crucial to enterprise's promotion and sales activities. A brand-new product needs to take account of customer-supplied ratings and reviews with the potential benefit of gains on brand recognition.

Sunshine Company hopes to devise a feasible strategy to sell three new products–a microwave oven, a baby pacifier, and a hair dryer with great success. Consulted by the company, we design a set of marketing goals for the promotion.

1.2 Outline

Our objective is to use quantitative and qualitative methods to identify and describe the patterns and relationships within and between star ratings, reviews, helpfulness ratings and sales, develop and implement a model to predict the time-based pattern, judge the influence of text-based and ratings-based factors on sales, and identify a reliable strategy will help Sunshine Company succeed in the three new online marketplace product offerings. To realize this objective, we will proceed as follows:

- **Analysis of Three Product Data Set** based on quantitative and qualitative methods to obtain a preliminary analysis of the patterns and relationships within and between star ratings, reviews, helpfulness ratings and sales and quantify the customer-supplied ratings and reviews.
- **Formulate a Useful Review Sorting model** based on ratings and reviews to identify the most informative reviews.
- **Formulate an ARIMA model** to identify the effects of text-based and ratings-based factors on sales.
- **Formulate an ARIMA-BP model** by combing the ARIMA model and the BP-neural network model to determine whether a product is successful or failing.
- **Identify a strategy** based on the results of the Useful Review Sorting model and the ARIMA-BP model and identify potentially important design features that would enhance product desirability.

1.3 Why ARIMA-BP Model

Although the ARIMA model can predict the reputation (sales) of a product, it only takes into account time-based features. In addition to past reputation (sales), the reputation of a product (sales) is related to other factors, so the ARIMA model alone does not fully capture all the factors that may affect reputation (sales). Therefore, we further add the BP-neural network model to the ARIMA model to capture the text-based and the ratings-base factors, and then build the ARIMA-BP model to predict product reputation (sales).

1.4 Model Assumptions

- The reviews will be more convincing and effective with higher helpfulness ratings.
- Review users are generally more inclined to browse the latest ones.
- For the same commodity, the latest review can reflect the latest information on goods and services, which is more instructive for consumers while making decisions.
- Reviews with longer text descriptions usually contain more comprehensive information. On the other hand, reviews will be too long to hit the point. The negative impact on the credibility from the review's length is increasing once out of certain range.
- The latest date when new customers read reviews is 2015/9/1.

1.5 Mathematical Notations

Table 1: Mathematical Notations

Symbol	Definition
r	Review
$H_v(r,p)$	Helpfulness rating
$T_v(r,p)$	Total number of votes
ζ_1	Increase in weight
T_e	Timeliness of review
t_{write}	Writing date
t_{read}	Reading date
L_e	Depth of Review
N_a	The number of emotional words
N_b	The number of product attributes words
N_t	Total length of the review
P_f	Quantified results of product attributes word
w_f^i	Corresponding weight of product attributes word
n_p^i	The appearance of the ith attribute
N_e	Intensity of emotional expression
w_f	The number of emotional words
$Reputation_{i,t}$	The reputation of commodity i at time interval t
$sales_{i,t}$	The sales of product i
$goodreview_{i,t}$	Favorable rate, measured by the number of star ratings
L, M, H, E	Weight of 0.1, 0.2, 0.3, 0.4 respectively

2 Data Preprocessing

Given that we have data from 2012-2015 of the three markets, the original database is extremely large, so we need to screen the most informative data. We first classify the data into three categories: redundant data, default data, and normal data. Next, we remove redundant to make it clearer to select the required data. Then, we delete default data. Finally, we summarize and analyze normal data.

2.1 Redundant Data Processing

There is data redundancy in the column “marketplace”, which includes the same content of “US”, so we delete this column. Besides, the contents of product category in each market are nearly the same, which are “Beauty”,”Major Appliances”,”Baby” respectively. We also remove them for convenience.

2.2 Default Data Processing

After observing product_title, we discover that some products do not belong to hair_dryer, pacifier and microwave oven, like conditioners and brush. Because the product_id in Amazon must contain information that can be used to identify the product, so we select the useful data according to naming rules. In the column ”verified_purchase”, there is data labeled with “N” and other invalid value. However, only the reviews from real buyers matter. Therefore, we delete them as default data.

3 Analysis of Three Product Data Set

3.1 Quantify indicators based on reviews and ratings

In order to facilitate follow-up analysis, we quantified the customer-supplied ratings and reviews, by constructing six indicators, which consist of the formal features and the content features.

1. The formal features of the customer-supplied ratings and reviews

(1) Helpful Rating. We utilize ratio method to derive the relationship between single review’s helpful ratings and the totality, in which we set the quantitative attributes of reviews without helpful ratings as zero to eliminate their effects on weighting. Then, we assign higher weights to reviews with more helpful votes. The formula of our model is as follow.

$$H(r,p)=\begin{cases} 0 & Y(r,p)=0 \\ \dfrac{H_v(r,p)}{T_v(r,p)} & 1\le Y(r,p)\le 10 \\ \dfrac{H_v(r,p)}{T_v(r,p)}\cdot\zeta_1 & 10<Y(r,p)\le 40 \\ \dfrac{H_v(r,p)}{T_v(r,p)}\cdot\zeta_2 & Y(r,p)>40 \end{cases} \tag{1}$$

(2) Timeliness of Review. Customer tend to browse latest reviews, so we quantify review’s timeliness through calculating the difference between writing date and reading date. Considering that extreme values cause deviation, we develop parameter σ to minimize their influence. Formula2 gives the general form of review timeliness.

$$T_e = \frac{360 - (t_{write} - t_{read})}{\sigma} \tag{2}$$

We assume that the largest difference is 360 days, while higher values will be treated as 360 days. t_{write} is writing date. t_{read} is reading date. We define 2015/9/1 is the reading date because the latest writing date is 2015/08/31. Next, we utilize σ to standardize timeliness and smaller the deviation. σ is 10, denoting the quantitative range of timeliness as $[0, 36]$.

(3) Depth of Review. We analyze the depth with reviews' effective length that is obtained through calculating the appearance of emotional words and product attributes words included. Here the depth is smoothed with logarithm to weaken the effect of denominator, calculated as follows:

$$L_e = \frac{ln(N_a + N_b)}{ln(N_t)} \tag{3}$$

WhereL_e is depth. N_a and $N_b$are the number of emotional words and product attributes words, while N_t means total length of the comment.

2. Features of Review's Content

(1) Product attributes words. Product attributes words and its synonyms involved in the review appear more frequently, the more detailed the description of the relevant feature is, thus the review has higher reference value. According to three tsv files provided, we classify the attributes of three products based on 5 dimensions, including Specific Terminology, Product Performance, Appearance, User Experience and Technical Index.

According to the assumptions, we have:

$$P_f = \sum_{i=1}^{5} w_f^i \cdot n_p^i \tag{4}$$

where $P_f$and w_f^i represent quantitative results of product attributes word and corresponding weight, while $n_p^i$is the appearance of the i^{th}attribute.

(2) The Intensity of Emotional Expression. While commenting, different emotional words tend to show different intensity of expression, thus we measure emotional intensity of reviews with the number of emotional words. The equation developed is as follows.

$$N_e = n_f \tag{5}$$

where N_e is the intensity of emotional expression and w_f shows the number of emotional words.

(3) Number of Modifiers. We assume that the more the number of qualifiers, the more intense the emotional expression of writers is, thus denoting stronger usefulness of text content. Common modifiers are showed in Table2.

While quantifying qualifiers, we first divide them into low(L), medium(M), high(H) and extreme(E) levels. L, M, H, E means weight of 0.1, 0.2, 0.3, 0.4 respectively. Quantitative

Table 2: The Intensity of Modifier and Examples

The Intensity of Modifier	Example
Extremely high(E)	Extremely, most, exceptionally, too, overly, in no way, perfectly
High(H)	Good, very, great, true, special, many, enough, quite, especially, more, particularly, greatly, deeply, rather, often
Medium(M)	Relatively, comparative, more, some, unlikely, not very, increasingly, not always
Low(L)	A little, slightly, just, constantly, never, regularly, rarely, once

formula is as below.

$$E_f = \sum_{i=1}^{4} w_f^i \cdot n_p^i \tag{6}$$

3.2 The patterns within star-ratings, reviews and helpful votes

In this part, We assume that reviews can be categorized into two clusters. Reviews whose $star_ratings$ are equal or greater than four are positive reviews, otherwise, they are negative reviews and sales equals the number of reviews. We first delete over 813 pieces of data before 2011 in the three markets and utilize the data from 2011 to 2015, for the reason that five years is reasonable for a market cycle. If the years before are taken into consideration, there will be products with very large amount of reviews from accumulation of years rather than from its popularity, which leads to deviation. Besides, the reviews from 2011-2015 are the latest ones having more analytical value.

1.The 5 Most-Reviewed Products and corresponding number of good reviews

Product 1:hair dryer

We choose the five most-reviewed products as the benchmark products to obtain most useful information for the following reasons:

(1) The reviews of Top 5 accounts for 18.53%, so we can derive that they occupy 18.53% of market attention.The most popular one among them is pertaining to Conair, whose total market share reaches 31.09%.

(2) These five products all have over 60% positive feedback, as shown in Figure1, which means their review texts best reflect customer's demand for product design and other features.

Product 2:Pacifier

As shown in Figure2 in this market, the 5 Most-Reviewed Products occupy 16.1% of total review volume , which means 16.1% of market attention. We discover the market share of Wubbanub reaching 25.82%, who has the top 3 most-reviewed products. After further calculation, we observe that the favorable percentage of these five products are also more than 60%, with the highest reaching 80.58 %.

Further, we analyze the text pertaining to top 5, the customers mention whether the nipple

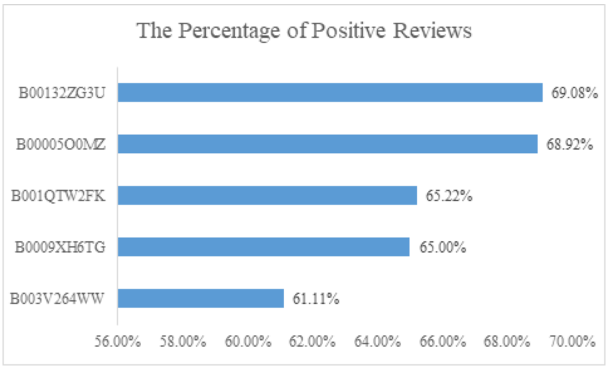

Figure 1: The 5 Most-Reviewed Hair Dryers and their amount of positive reviews

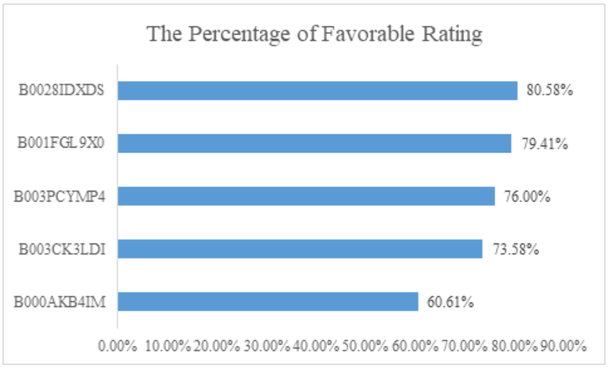

Figure 2: The 5 Most-Reviewed Pacifiers and their quantity of positive reviews

is easy to find(with appropriate size, anti-lost chain) and wash is critical. Apart from these, doll pacifier integrated design matters. All these advantages serve to soothe babies' feeling and exercise their sucking ability.

Product 3: Microwave oven

Top5 account for 29.00% of reviews, attracting nearly one third of total attention. The phenomenon reflects high degree of concentration in the microwave oven segment. After further analysis, we learn that these five products' favorable ratio and sales are positively correlated(Figure3, indicating that higher-ranked products are more in line with users' expectation. Therefore, Top 2 products are more preferred by consumers. However, different from dryers and pacifiers, 30% of all reviewers view the fifth-ranked product unfavorably. It is suggested to avoid the issues mentioned in this product's reviews, so as to establish a good reputation.

Given the information, we discover that Whirlpool is the most preferred brand with total market share of 25.66%, followed by Sharp. Besides, top 3 of the best-selling products belong to Whirlpool. These products have some common features, like black or white appearance and stainless steel material. Furthermore, we find that users also focus on the occupied space and function. Specifically, Top4 have barbecue and automatic menu functions in addition to heating. On the other hand, although Samsung which ranks the fifth also gets a large amount of sales, its negative reviews pile up owing to program errors and poor after-sale protection. And the reason why it has great sales volume is that Samsung has a good reputation on other products such as television, so a good brand image is critical. Top 5 products are showed below.

2. Analysis of help_votes

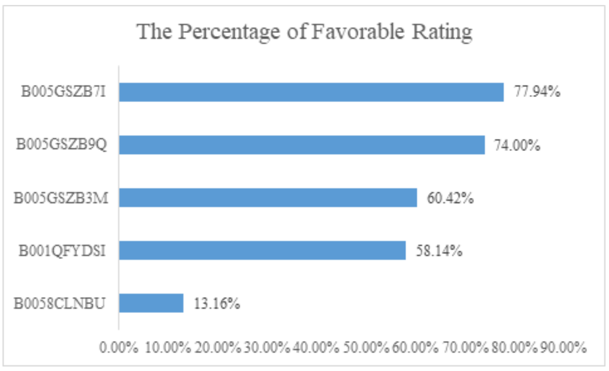

Figure 3: The 5 Most-Reviewed Microwave Ovens

Product 1: hair dryer

After analyzing the data, we observe that there are more than 125 words in 72.92% of the reviews whose help_votes are over 30. All of these reviews cover various aspects of velocity, heat, touch, sound, smell, etc, indicating that the more detailed the reviews, the more help_votes they receive.

Product 2: Pacifier

In this market, there are more than 150 words in 77.27% of the reviews whose help_votes are over 30, evaluating the smell, materials and attracting design mainly.

Product3: Microwave oven

We find the almost 85.49% of the comments with help_votes higher than 30 have more than 150 words. These reviews explain the appearance, functions and capacity, etc.

3.3 The relationship between star-ratings, reviews and helpful votes

1. Analysis of vine reviews_votes

The proportions of reviews from vine voice in the three markets are identified as 3%, 1%, 1.5% respectively. Although all of the proportions are very small, the products are more likely to gain good feedback from vine voice, which may associated with the access threshold of Amazon vine program. Also, vine reviews tend to get more help_votes. It is deserved to mention that the highest hep_votes winner(help_votes=814) in the microwave oven comes from vine voice. The above data show that vine plan will help to raise awareness of the product

2. The changing pattern of monthly averaged quantity of reviews

Taking hair_dryers as example, we count the number of reviews each quarter and consolidate the 5 Most-Reviewed hair dryers into Figure4. We find that the reviews of Top 5 hair_dryers are concentrated in the first quarter, reflecting high demand in this period. The company can intensify promotion during this peak. Microwave oven and pacifiers can also increase marketing efforts at their peak period.

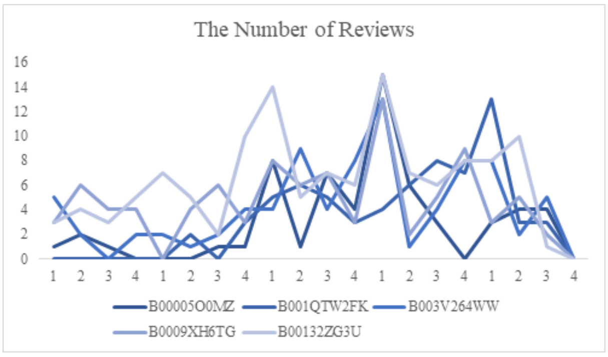

Figure 4: The 5 Most-Reviewed hair dryers' reviews in each quarter

3. Relationship between reviews and star_rating

(1) The LSTM Model

In this section, we will describe a LSTM model for better evaluating the rating levels' association with descriptors in text-based reviews. According to our research, the conclusion is that the positive descriptors related to higher rating levels while the negative related to lower ones.

- **Building Text-Based Data Sets**. First we preprocessed data provided by MCM, we extracted review body and save all reviews into certain format data set automatically by program. Then we read files and construct matrix for LSTM model training. And this is text-based data sets. We also extracted token form data set.
- **LSTM: Text-Based Emotion Analysis**
  Long Short Term Memory (LSTM) is a deep learning method, an improvement on RNN (Recurrent Neural Network) but without gradient vanishing and exploding. In our study, we created data sets and token, and then we finally got well-trained LSTM model.
- **Training Model and Analysis**
  To improve the accuracy of the model, we constructed LSTM based on Keras framework. We used this model to predicts text-based emotion automatically. LSTM model will get probability, which also means weight, of each product, and these parameters is the input data of BP neural network. So here comes the conclusion that LSTM helps in text-based measurement.

(2) Linear Regression Model

This section discusses the linear relationship between how star ratings and other variables. The conclusion is that with the star rating ascending, we will get more reviews with better reputation.

- **Rating Levels and Reputation**

  We used well-trained LSTM model to evaluate reputation by recognizing emotion form reviews. By analyzing these statistic, we got a linear regression formula below. It is obvious that with the increasing of star ratings, a product will gain more good reviews and less bad reviews, otherwise it will gain more bad reviews but less good reviews.

$$y_i = \alpha + \beta x_i + \varepsilon_i \tag{7}$$

- **Rating Levels and Reviewing Motivation**

  We evaluate motivation by quantify reviews' length. A customer will writes more comments if his or her motivation is stronger. By analyzing the variance of rating levels and length of reviews, we stimulate a curve and found that it's a linear regression.

$$y_i = \alpha + \beta x_i + \varepsilon_i \tag{8}$$

- **Quantify indicators based on reviews and ratings**

  In order to facilitate follow-up analysis, we quantified the customer-supplied ratings and reviews, by constructing six indicators, which consist of the formal features and the content features.

4 Identify the most informative reviews

After introducing new products to the market, the company can get the feedback on products through tracking customers' reviews. For the sake of obtaining the most informative review, we propose a type of review utility ranking model, which assists the strategy making.

Specifically, based on the six indicators quantified from customer-supplied ratings and reviews in chapter 2, we use Fuzzy Analytical Hierarchy Process(FAHP) to determine the weight of each indicator, and then combine Technique for Order of Preference by Similarity to Ideal Solution(TOPSIS) method for the usefulness of online reviews calculation and sorting.

4.1 Determine the weight of each indicator

Based on the six indicators quantified from customer-supplied ratings and reviews in chapter 1, we use Fuzzy Analytical Hierarchy Process(FAHP) to determine the weight of each indicator.

1. Importance judgement

Normally expert scoring is used to prioritize the most important indicators in the matrix. Here our team determine the prioritization by comparison and judgment. We first utilize pairwise comparison and score according to prioritization matrix comparison rule in Appendix A. Then we formulate fuzzy equivalence matrix according to the scoring. Finally, we obtain precedence matrix (precision 0.01) after averaging as shown in Table3.

2. Judgment Matrix Construction

We assume $X = x_1, x_2, \ldots, x_n$ as a collection of all attributes. Then we use listing rules in Table3 to denote pairwise comparison approaches and construct matrix $C = (c_{ij})_{n\times m}$ while

Table 3: fuzzy equivalence matrix

H	$H1$	$H2$	$H3$	$H4$	$H5$	$H6$
H1	0.5	0.77	0.48	0.68	0.47	0.38
H2	0.23	0.5	0.28	0.45	0.32	0.23
H3	0.52	0.72	0.5	0.75	0.48	0.4
H4	0.32	0.55	0.25	0.5	0.23	0.2
H5	0.53	0.68	0.52	0.77	0.5	0.48
H6	0.62	0.77	0.6	0.8	0.52	0.5

matrix C is determined as Judgment Matrix. We get:

$$C = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ C_{n1} & C_{n2} & \cdots & C_{nn} \end{bmatrix} \tag{9}$$

3. Calculate weight

(1) Judgment Matrix C does column wise normalization and get Matrix $B = (b_{ij})_{m\times n}$.

$$b_{ij} = \frac{a_{ij}}{\sum_{i=1}^{n} a_{ij}}, (i, j = 1, 2, \ldots, n) \tag{10}$$

(2) Then, we add the elements in Matrix B by row and get Vector $C = (c_1, c_2, \ldots, c_n)^T$

$$c_{ij} = \sum_{j=1}^{n} b_{ij}, (I, j = 1, 2, \ldots, n) \tag{11}$$

(3) We normalize Vector C and get Eigenvector $C = (c_1, c_2, \ldots, c_n)^T$

$$c_{ij} = \frac{c_i}{\sum_{i=1}^{n} c_i}, (i = 1, 2, \ldots, n) \tag{12}$$

Determining weight of index from Equation(10)-(12), we find the normalized weight of the indicators to be: $W = (0.157, 0.095, 0.165, 0.093, 0.185, 0.141)$.

4.2 The Useful Review Sorting model

After quantifying the indicators, we use TOPSIS method to construct The Useful Review Sorting Model.

To eliminate the effect of attributes' different units, we first normalize the raw data, making each variable equally expressive. Then, we develop decision matrix $A = (a_ij)_{m\times n}$:

$$b_{ij} = \frac{a_{ij} - \bar{a}_j}{s_j}, (i = 1, 2, \ldots, m; j = 1, 2, \ldots, n) \tag{13}$$

Where $\bar{a}_j = \frac{1}{m} a_{ij}$, $s_j = \sqrt{\frac{1}{m-1} \sum_{i=1}^{m} (a_{ij} - \bar{a}_j)^2}$.

And then we get the weighted specification matrix $C_w = (c_{ij}^w)_{m\times n}$, where

$$c_{ij}^{w} = w_j \times b_{ij}, (i = 1, 2, \ldots, m; j = 1, 2, \ldots, n) \tag{14}$$

Suppose the j^{th} attribute value in positive ideal solution C^+ is C_j^+; the j^{th} attribute value in negative ideal solution C^- is C_j^-, then:

$$c_j^+ = \max_i c_{ij} \tag{15}$$

$$c_j^- = \min_i c_{ij} \tag{16}$$

The distance between review d_i and positive ideal solution is:

$$s_i^+ = \sqrt{\sum_{j=1}^{n} (c_{ij} - c_j^+)^2}, i = 1, 2, \ldots, m \tag{17}$$

The distance between review d_i and negative ideal solution is:

$$s_i^- = \sqrt{\sum_{j=1}^{n} (c_{ij} - c_j^-)^2}, i = 1, 2, \ldots, m \tag{18}$$

Calculate the queued metric values for each review:

$$\frac{s_i^-}{(s_i^- + s_i^+)}, i = 1, 2, \ldots, m \tag{19}$$

Sort by f^* from big to small, we can get the usefulness of the reviews sorted out.

We quantify the reviews of three products through TOPSIS technique mentioned before. From Equation(16)-(19), we can derive the situations of the three markets.

the positive ideal solution:

$$\begin{cases} C_{oven}^+ = [0.217, 0.178, 0.387, 0.263, 0.157, 0.326] \\ C_{pacifier}^+ = [0.217, 0.178, 0.387, 0.263, 0.157, 0.326] \\ C_{hair}^+ = [0.217, 0.178, 0.387, 0.263, 0.157, 0.326] \end{cases} \tag{20}$$

the negative ideal solution:

$$\begin{cases} C_{oven}^- = [-0.230, -0.182, -0.269, -0.007, -0.367, -0.332] \\ C_{pacifier}^- = [-0.171, -0331, -0.238, -0.193, -0.247, -0.232] \\ C_{hair}^- = [-0.1402, -0.272, -0.263, -0.107, -0.377, -0.442] \end{cases} \tag{21}$$

Using Equation(20)-(21) respectively, the distance to the positive ideal solution s^+ and the distance to the negative ideal solution s^- can be obtained. Value of f_i^* is then calculated by the equation (18). The results using value of f_i^* as standard are shown in Table4 below. Owing to spatial confined, we only list the optimal review results of the three markets.

Table 4: The optimal results of the three markets

Market	Ranking(TOPSIS)	S^+	s^-	f^*
Microwave oven	1	0.824	0.713	0.464
Pacifier	1	0.785	0.715	0.476
Hair dryer	1	0.306	1.151	0.789

4.3 Compare TOPSIS Measure with Helpful votes Measure

The number of helpful votes can also be used to measured whether review is helpful. However, a comparison of the TOPSIS sorting results with the sorting results of the helpful votes shows that, in addition to the helpful votes, other attribute indicators can also have a significant impact on the usefulness of the review. Based on the results of the TOPSIS ranking, product attributes words and emotional expression intensity are higher than helpful voting, reflecting the fact that "helpful voting" is important but not the only factor for the usefulness of reviews. For space, we only select the top 1 helpful review in the product market sorted by helpful votes and compare the attribute indicators of the review sorted by the TOPSIS sorting model, as shown in Table5.

Table 5: Comparison between TOPSIS sorting and helpful votes sorting

	Microwave oven		Pacifier		Hair Dryer	
Ranking(# Helpful votes)	1	6	1	4	1	2
Ranking (TOPSIS)	4	1	3	1	6	1
Helpful votes	1.85	1.42	1.76	0.84	1.64	1.57
Timeliness of review	8	20	16	30	21	18
Depth of review	0.52	0.65	0.51	0.66	0.65	0.71
Product attributes words	3.26	6.32	6.05	6.86	6.02	6.25
The Intensity of Emotional Expression	9.45	13.25	12.26	12.71	7.95	10.5
Number of modifiers	7.15	6.32	4.51	7.4	4.32	5.74

5 ARIMA Model for Time Series Forecasting

In this section, we build an ARIMA model to analyze the time-based pattern of products' reputation. Specifically, we first build a way to measure reputation, which includes two aspects of customer attention and praise. Then we establish the ARIMA model to predict trends in product reputation over time.

5.1 A measure of reputation

The reputation of a product can be divided by two dimensions, including both popularity (whether there are many consumers aware of it), as well as positive rating (whether there are many consumers liking it). Based on the above two aspects, we develop a way to measure reputation:

$$Reputation_{i,t} = sales_{i,t} + goodReview_{i,t} \tag{22}$$

Where $Reputation_{i,t}$ represents the reputation of commodity i at time interval t. Here we define t as a quarter. $sales_{i,t}$ is the sales of $product_i$, indicating whether there are many consumers aware of it. Since the exact volume of sales is not publicly available in Amazon, we utilize the number of reviews as a proxy variable of sales. $goodreview_i$ represents whether there are many consumers liking $product_i$, measured by the number of star ratings over three. The higher the value of $Reputation_i$, the better the reputation of $product_i$. Figure5 shows the change of products' reputation of the three markets over time.

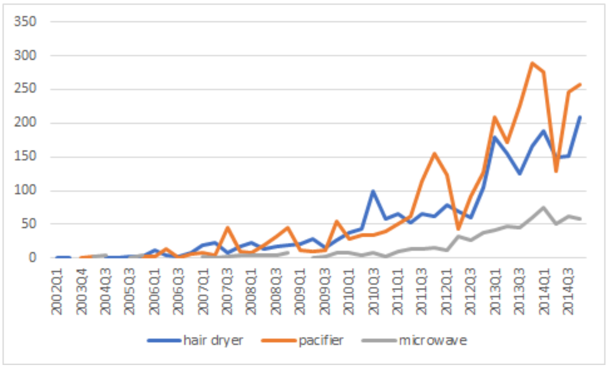

Figure 5: The change of products' reputation of the three markets over time

5.2 The prediction based on the ARIMA model

1. Smoothness Check

By drawing a curve of the trend in picture ?, we find that these three sequences aren't stationary time series. We first apply First Differences Method (FDM) and unit root testing to process the three sequences, getting not-stationary series again. So we repeat the process and obtain stationary series after the second-order difference calculation. Then, we apply these finally stationary data in reputation analysis. Through the above process, we develop our model as ARIMA(p, 2, q).

2. Order Choose

To determine the appropriate p and q for the ARIMA(p, q, d) model, we test the Second Order Difference Sequence through the autocorrelation and partial autocorrelation functions. Both functions are trailing after calculation. The autocorrelation function shrinks significantly after the first order difference within two standard deviations, while the autocorrelation function shrinks significantly after the second order processing within the same range. Furthermore, by observing the p value of Q statistic, we find that the sequence is non-white noise sequence, so we select ARIMA (1,2,2) as the most appropriate model.

3. Prediction of original reputation sequence

Through ARIMA(1,2,2) model, we forecast the value of original sequences. The results are shown in Table6. Owning to the limitation of the scope, only seven quarter's data of pacifier No. 295960359 are listed.

Table 6: Predicted Reputation of Pacifier-295960359

Quarter	Original reputation	Predicted reputation	Relative error(%)	Quarter	Original reputation	Predicted reputation	Relative error(%)
2014Q1	3.12	3.05	-2.24	2015Q1	3.53	3.65	3.4
2014Q2	3.24	3.32	2.47	2015Q2	3.62	3.68	1.66
2014Q3	3.33	3.24	-2.7	2015Q3	3.84	3.82	-0.52
2014Q4	3.42	3.51	2.63				

5.3 Whether a product's reputation is increasing or decreasing

According to predictions with ARIMA (1,2,2) model, we can determine trends of product's reputation in the three markets. The standard used to judge whether a product's reputation is increasing or decreasing after August 31, 2015 is showed below:

- Increasing: if $Reputaion_t > Reputaion_{t-1}$
- Decreasing: if $Reputaion_t < Reputaion_{t-1}$

Table7 gives the forecast results. Here we only extract the trends in reputation of two products in baby pacifiers market to save space.

Table 7: Trends in reputation of two products in baby pacifiers market

Product	Quarter	Reputation	Trend
Pacifier-295960359	2015Q4	3.93	Increasing
Pacifier-295960359	2016Q1	4.12	Increasing
Pacifier- 93476192	2015Q4	3.62	Decreasing
Pacifier- 93476192	2016Q1	3.23	Decreasing

6 BP-Neural network model

Based on the correlation analysis in Chapter 2, we find that product sales are related to consumers' reviews and ratings, so we can use text-based and ratings-based features to predict product sales. Because the text-based and ratings-based features are multidimensional, which makes the ARIMA model in chapter 3 no longer applicable, we build a BP-neural network model to identify the relationship between text-based and ratings-based features and sales.

We select the first 70% of the data set as the training set, and 30% as the validation set, that is, the data from 2013Q1 to 2014Q4 as the training sample, and the data from 2015Q1 to 2015Q3 as the validation sample to test the effectiveness of the model. The parameters of the model are set as follows:

- The selected data is in quarterly units;
- We choose to use a three-layer BP-neural network;
- According to the analysis of the factors influencing sales, the input layer variable is determined as: previous sales volume and 6 indicators identified in Chapter1 , so the number of input nodes is set to 7;
- Since we are forecasting quarterly sales, the output node is set to 1;
- After testing, we determined that the optimal number of implied layer nodes was 4.

The BP-neural network model error values are shown in the Table8.

Table 8: BP-neural network model error

	SSE	MAPE	RMSE
BP-neural network model	32.73	6.94%	5.83

7 ARIMA-BP Model

Although the ARIMA model can predict the reputation (sales) of a product, it only takes into account time-based features. In addition to past reputation (sales), the reputation of a product (sales) is related to other factors, so the ARIMA model alone does not fully capture all the factors that may affect reputation (sales). Therefore, we further add the BP-neural network model to the ARIMA model to capture the text-based and the ratings-base factors, and then build the ARIMA-BP model to predict product reputation (sales).

7.1 Modification of the ARIMA Model

1. Redefine Reputation

Text-based and ratings-source features will be used in the BP-neural network model to predict a product's reputation, so $Reputation$, as defined in the previous chapter, will no longer include $goodreview_{i,t}$. In this section, we only use $Sales$ to represent $Reputation.$

2. The prediction based on the modified ARIMA model

Based on the steps in the last chapter to build the ARIMA model, we constructed the ARIMA (2,1,8) model in our sales forecast after smoothness analysis and parameter determination. Then, we validate the modified ARIMA model in three product markets, as shown in Table9.

Table 9: ARIMA(2,1,8) Model Validation

	Predicted variable	R^2	Statistic value	DF	Sig
Microwave oven	Sales	0.837	87.658	1123	0.015
Pacifier	Sales	0.865	89.389	1415	0.023
Hair dryer	Sales	0.914	96.74	1258	0.008

The results of Table10 show that R^2 in all three product markets exceeds 0.8, indicating that the model fits well. Also, the model is significant at 5%, which also indicates that the ARIMA(2,1,8) model is appropriate. The error value of the modified ARIMA model is shown in the table

Table 10: Modified ARIMA model error

	SSE	MAPE	RMSE
Modified ARIMA model	35.45	7.50%	5.94

7.2 Prediction based on ARIMA-BP Model

After the realization and verification of ARIMA and BP neural network models, we use a weighted parallel method to predict it in combination, and construct the ARIMA-BP model to predict the sales:

$$f(T) = w_1 G_1(T) + w_2 G_2(T) \tag{23}$$

Where $G1(T)$ represents the predicted value of the ARIMA model under the quarterly T, $G2(T)$ represents the prediction value of the BP-neural network model under the quarterly T, and w represents the weight of each single model. We determine the weight of W by equal weight averaging, error variance weighted average and relative error countdown. The evaluation effect of the three methods is shown in Table11.

According to the results of Table11, we can see that the error of the relative error countdown is the smallest, so we choose the error countdown method for the allocation of weights, and then get the following ARIMA-BP neural network combination prediction model:

$$f(T) = 0.43ARIMA(T) + 0.57BP(T) \tag{24}$$

7.3 Comparison of ARIMA, BP neural network and ARIMA-BP model

Based on the weights of the previous section, we build a combination prediction model of ARIMA-BP neural network, and then get a sales forecast chart of two single models and the

Table 11: ARIMA-BP Model evaluation results

	weight		relative error
	Modified ARIMA	BP-neural network	
Equal-power average method	0.5	0.5	7.32%
weighted average method of error variance	0.67	0.33	6.75%
relative error countdown method	0.43	0.57	5.36%

ARIMA-BP model.

The prediction results in the Figure6 show that the predicted value based on the ARIMA-BP model is closer to the actual value. The BP neural network deviates more from the actual value than ARIMA.

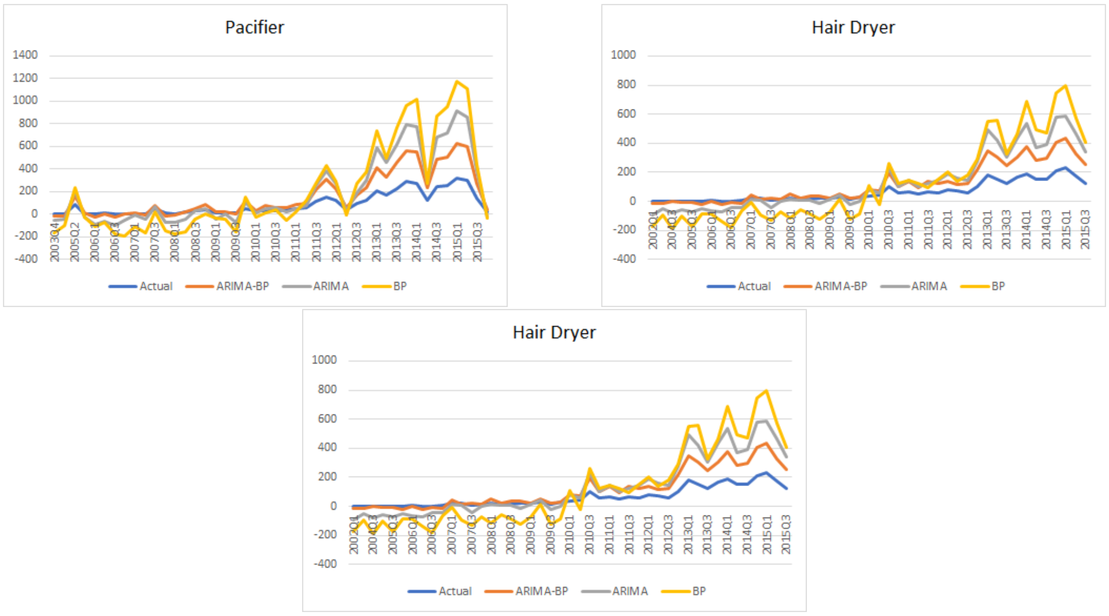

Figure 6: Comparison of ARIMA, BP neural network and ARIMA-BP model

7.4 Whether a product is successful or failing

We judge whether a product is successful or failing after August 31, 2015 by the following criteria:

- Successful: if a product's sales are in the top 30% (in the same product market)
- Failing: if a product's sales are in the back 30% (in the same product market)

Due to space constraints, we report sales trends for only two products in the nipple market, as shown in the Table12.

Table 12: Product sales trends

Product	Quarter	Sales Rank	Trend
Pacifier-295960359	2015Q4	15	Successful
Pacifier-295960359	2016Q1	14	Successful
Pacifier- 93476192	2015Q4	394	Failing
Pacifier- 93476192	2016Q1	361	Failing

8 Strategy for Sunshine Company

Based on the results of the Useful Review Sorting model and the ARIMA-BP model, we propose an online sales strategy and identify some potentially important design features that would enhance product desirability.

Sales Strategy: Based on the above analysis, we explore the following strategies to help our products make a hit:

- ·Promote products with key points embedded in reviews getting the highest favorable ratio. For instance, the key words of hair dryer are velocity and styling effect, and pacifiers' highlights are durability, anti-lost and soft material.
- ·Participate in Amazon Vine Program to quickly attract attention and develop brand recognition. Users are very sensitive to microwaves, hair dryers and other appliances.
- Pay attention to consumers' reviews and make a quick response to negative reviews.
- Intensify promotion at the peak period, while hair dryers are most popular in the first quarter.

Design Focus:Aiming to satisfy customer's demand, we recommend the following four key points in designation.

- ·In the hair dryer market, high velocity, low noise, good durability, light weight, qualified after-sales service is the pursuit of the users. Apart from these, poor after-sales service is the main point in negative reviews.
- ·In the pacifiers market, the customers emphasize on doll-pacifier integrated design, easiness to find and wash, material,etc. Among them, doll pacifier integrated design best meets the core needs - to appease the baby's mood, exercise their ability to suck.
- · In the microwave oven market, the best-selling product have some common features, like black or white appearance and stainless steel material. Customers also focus on the appropriate occupied space and comprehensive function. Besides, improve stability and avoid program errors are critical to avoid a poor star rating.
- The most popular brands in three markets are Conair, Wubbanub, Whirlpool. Their design features of them can be referred to

9 Strengths and Weaknesses

9.1 Strengths

- The Useful Review Sort Model takes full account of the text-based and ratings-based factors of reviews, which can show the most informative reviews to the Marketing Director.

- The ARIMA-BP model captures the time-based patterns, text-based and ratings-based patterns, making it more accurate to predict product sales.

- We assign higher weights to reviews with more helpful votes and set the quantitative attributes of reviews without helpful ratings as zero, which can eliminate their effects on weighting.

9.2 Weaknesses

- Due to lack of data, our validation set contains only data from 2015Q1 to 2015Q3, which can limit the optimization degree of BP-neural network model.

- In the Useful Review Sort Model, the weight of the indicator should be determined according to the consumer's judgment. But the weight selection in this paper is made by three members of our team, which can not represent the preferences of all consumers.

.

References

[1] Zhang Yue, Review Sentiment Analysis and Sales Forecast based Item Selection of Foreign Trade E-commerce [D], Beijing Jiaotong University, 2019

[2] Hu Pengji, Research on Demand Forecast and development Countermeasures of Urban Logistics Based on Arima-BP [D], Tianjin University,2019

[3] LIU Fasheng, Research on sentiment analysis combining attention mechanism and sentence ranking[J/OL], Computer Engineering and Applications:1-11[2020-03-10]

[4] Tsenga F M, Yub H C, Tzengc G H. Combining Neural Network Model With Seasonal Time Series ARIMA Model[J]. Technological Forecasting Social Change, 2002, (69).

[5] Yu Yong,Si Xiaosheng,Hu Changhua,Zhang Jianxun. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures.[J]. Neural computation,2019.

[6] Chen Yinguang, Researc and Application of Prediction Technology for MEN's Wear Sales data [D], DongHua University, 2019.

Appendices

Appendix A Source Code

Here are programs we used in our models.

Our simulation source written in Python:

```python
import numpy as np
import pandas as pd


# class preProcess
class preProcess(object):
    '''
        This class is for data pre-processing.
        Here follows methods description in Chinese.
    '''
    def __init__(self, hair_dryer_path=None,
                 microwave_path=None, pacifier_path=None):
        '''
        :param hair_dryer_path: path of hair_dryer.xlsx
        :param microwave_path: path of microwave.xlsx
        :param pacifier_path: path of pacifier.xlsx
        '''
        # to DataFrame format
        self.hair_dryer =\
            pd.DataFrame(pd.read_excel(hair_dryer_path))
        self.microwave =\
            pd.DataFrame(pd.read_excel(microwave_path))
        self.pacifier =\
            pd.DataFrame(pd.read_excel(pacifier_path))
        # self.hair_dryer = pd.read_excel(hair_dryer_path)
        # self.microwave = pd.read_excel(microwave_path)
        # self.pacifier = pd.read_excel(pacifier_path)
        return None

    def drop_col(self, str):
        # mydata = mydata.drop(['', ''], axis=1)
        self.hair_dryer = self.hair_dryer.drop([str], axis=1)
        self.microwave = self.microwave.drop([str], axis=1)
        self.pacifier = self.pacifier.drop([str], axis=1)
        # print("self.hair_dryer")
        # print(self.hair_dryer)
        return (self.hair_dryer, self.microwave, self.hair_dryer)

    def del_verified_purchase_N(
            self, hair_dryer_save_path=None,
        microwave_save_path=None, pacifier_save_path=None):
        '''
        This method deletes tuples whose
        verified_purchase is 'N',
        which means the customer didn't
        purchase the product but still made comments.
        :return:None
        '''
        # pattern: df = df.drop(df[df['three'] == 14].index)
        hair_dryer = self.hair_dryer
        microwave = self.microwave
        pacifier = self.pacifier

        # df['thing'] = df['thing'].str.upper()

        # delete if verified_purchase is 'N'.
        # hair_dryer = hair_dryer.drop(
        # hair_dryer[hair_dryer['verified_purchase'] == 'N'].index)
        # microwave = microwave.drop(
        # microwave[microwave['verified_purchase'] == 'N'].index)
        # pacifier = pacifier.drop(
        # pacifier[pacifier['verified_purchase'] == 'N'].index)

        self.hair_dryer = hair_dryer.drop(
            hair_dryer[
                (hair_dryer['verified_purchase'] != 'Y')
                & (hair_dryer['verified_purchase'] != 'y')
                & (hair_dryer['vine'] == 'N')
            ].index
        )
        self.microwave = microwave.drop(
            microwave[
                (microwave['verified_purchase'] != 'Y')
                & (microwave['verified_purchase'] != 'y')
                & (hair_dryer['vine'] == 'N')
            ].index
        )
        self.pacifier = pacifier.drop(
            pacifier[
                (pacifier['verified_purchase'] != 'Y')
                & (pacifier['verified_purchase'] != 'y')
                & (hair_dryer['vine'] == 'N')
            ].index
        )

        # save to excel
        # self.hair_dryer.to_excel(hair_dryer_save_path)
        # self.microwave.to_excel(microwave_save_path)
        # self.pacifier.to_excel(pacifier_save_path)
        return (self.hair_dryer, self.microwave, self.pacifier)

    def delete_error_data(self, sub_str):
        '''
        This method delete the wrong product info mixed in clear data.
        :return:
        '''
        self.hair_dryer = self.hair_dryer.drop(
            self.hair_dryer[
            ~self.hair_dryer['product_title'].str.contains('hair dryer')
            ].index
        )
        self.microwave = self.microwave.drop(
            self.microwave[
            (~self.microwave['product_title'].str.contains('microwave'))
            ].index
        )
        self.pacifier = self.pacifier.drop(
            self.pacifier[
                (
                (~self.pacifier['product_title'].str.contains('pacifier'))
                & (~self.pacifier['product_title'].str.contains('nipple'))
                )
            | (self.pacifier['product_title'].str.contains('nipple sponge brush'))
            ].index
        )

        return None

    def del_helpful_votes_total_votes(self):
        '''
                This method deletes tuples
                whose helpful_votes/total_votes is '' or 0
                :return:None
        '''
        # pattern: df = df.drop(df[df['three'] == 14].index)
        hair_dryer = self.hair_dryer
        microwave = self.microwave
        pacifier = self.pacifier
        # | (hair_dryer['helpful_votes'] != '')

        self.hair_dryer = hair_dryer.drop(
            hair_dryer[
                (hair_dryer['helpful_votes'] == 0)
                | (hair_dryer['helpful_votes'] == '')
                ].index
        )
        self.microwave = microwave.drop(
            microwave[
                (microwave['helpful_votes'] == 0)
                | (hair_dryer['helpful_votes'] == '')
                ].index
        )
        self.pacifier = pacifier.drop(
            pacifier[
                (pacifier['helpful_votes'] == 0)
                | (hair_dryer['helpful_votes'] == '')
                ].index
        )
        return (self.hair_dryer, self.microwave, self.pacifier)

    def transDF(self):

        hair_dryer = self.hair_dryer
        microwave = self.microwave
        pacifier = self.pacifier

        self.hair_dryer = self.hair_dryer.stack()
        self.hair_dryer = self.hair_dryer.unstack(0)
        print("hair_dryer.stack()")
        print(self.hair_dryer)
        return (self.hair_dryer, self.microwave, self.pacifier)

    def mysort(self, str=None):
        self.hair_dryer = \
            self.hair_dryer.sort_values(str, ascending=True)
        self.microwave = \
            self.microwave.sort_values(str, ascending=True)
        self.pacifier = \
            self.pacifier.sort_values(str, ascending=True)
        return (self.hair_dryer, self.microwave, self.pacifier)
# class preProcess


def test(hair_dryer_path=None,
         microwave_path=None, pacifier_path=None):
    pre = preProcess(hair_dryer_path=hair_dryer_path,
        microwave_path=microwave_path,
                     pacifier_path=pacifier_path)
    # print(pre.hair_dryer)
    # pre.drop_col('marketplace')     # delete marketplace
    # pre.drop_col('product_category')
    # print("pre.hair_dryer.test")
    # pre.del_helpful_votes_total_votes()
    # print("pre.hair_dryer.test")
    # print(pre.hair_dryer)
    hair_dryer_save_path = \
        "./test_dataset/final_test/hair_dryer.xlsx"
    microwave_save_path = \
        "./test_dataset/final_test/microwave.xlsx"
    pacifier_save_path = \
        "./test_dataset/final_test/pacifier.xlsx"
    # print("123")
    # pre.del_verified_purchase_N(
    #     hair_dryer_save_path=hair_dryer_save_path,
    #     microwave_save_path=microwave_save_path,
    #     pacifier_save_path=pacifier_save_path
    # )
    # pre.drop_col('verified_purchase')
    # pre.drop_col('vine')
    # print("123")

    # pre.transDF()

    print(pre.hair_dryer)

    # # save to excel
    pre.hair_dryer.to_excel(hair_dryer_save_path)
    pre.microwave.to_excel(microwave_save_path)
    pre.pacifier.to_excel(pacifier_save_path)

    return None


def main(hair_dryer_path=None,
         microwave_path=None, pacifier_path=None):
    pre = preProcess(
        hair_dryer_path=hair_dryer_path,
        microwave_path=microwave_path,
        pacifier_path=pacifier_path
    )
    pre = preProcess(hair_dryer_path=hair_dryer_path,
        microwave_path=microwave_path,
                     pacifier_path=pacifier_path)
    # print(pre.hair_dryer)
    pre.drop_col('marketplace')  # delete marketplace
    pre.drop_col('product_category')
    # print("pre.hair_dryer.test")
    pre.del_helpful_votes_total_votes()
    # print("pre.hair_dryer.test")
    # print(pre.hair_dryer)
    hair_dryer_save_path = \
        "./output_dataset/hair_dryer.xlsx"
    microwave_save_path = \
        "./output_dataset/microwave.xlsx"
    pacifier_save_path = \
        "./output_dataset/pacifier.xlsx"
    # print("123")
    pre.del_verified_purchase_N(
        hair_dryer_save_path=hair_dryer_save_path,
        microwave_save_path=microwave_save_path,
        pacifier_save_path=pacifier_save_path
    )
    pre.drop_col('verified_purchase')
    # pre.drop_col('vine')
    # print("123")
    pre.delete_error_data(sub_str='hair dryer')
    pre.mysort('product_id')
    print(pre.hair_dryer)

    # # save to excel
    pre.hair_dryer.to_excel(
        hair_dryer_save_path, index=False)
    pre.microwave.to_excel(
        microwave_save_path, index=False)
    pre.pacifier.to_excel(
        pacifier_save_path, index=False)
    print("finished")
    return None


if __name__ == "__main__":
    hair_dryer_path = "./dataset/hair_dryer.xlsx"
    microwave_path = "./dataset/microwave.xlsx"
    pacifier_path = "./dataset/pacifier.xlsx"

    hair_dryer_test_path = \
        "./test_dataset/hair_dryer.xlsx"
    microwave_test_path = \
        "./test_dataset/microwave.xlsx"
    pacifier_test_path = \
        "./test_dataset/pacifier.xlsx"

    # test(
    #     hair_dryer_path=hair_dryer_test_path,
    #      microwave_path=microwave_test_path,
    #     pacifier_path=pacifier_test_path
    # )

    main(hair_dryer_path=hair_dryer_path,
         microwave_path=microwave_path,
         pacifier_path=pacifier_path)
```

```
import numpy as np
import pandas as pd
import os
import sys
import codecs
import chardet


# class ProcessData(object)
class ProcessData(object):
    '''
        This class is for data pre-processing.
        Here follows methods description in Chinese.
    '''
    def __init__(self, hair_dryer_path=None,
                 microwave_path=None,pacifier_path=None):
        '''
        :param hair_dryer_path: path of hair_dryer.xlsx
        :param microwave_path: path of microwave.xlsx
        :param pacifier_path: path of pacifier.xlsx
        '''
        # to DataFrame format
        self.hair_dryer=\
            pd.DataFrame(pd.read_excel(hair_dryer_path))
        self.microwave=\
            pd.DataFrame(pd.read_excel(microwave_path))
        self.pacifier=\
            pd.DataFrame(pd.read_excel(pacifier_path))
        # print("len(self.hair_dryer)")
        # print(len(self.hair_dryer))

        # print("self.hair_dryer.iat[1, 2]")
        # print(self.hair_dryer.iat[1, 2])
        return None

    def drop_col(self, str):

        self.hair_dryer=\
            self.hair_dryer.drop([str], axis=1)
        self.microwave =\
            self.microwave.drop([str], axis=1)
        self.pacifier =\
            self.pacifier.drop([str], axis=1)

        return (self.hair_dryer,self.microwave,self.hair_dryer)

    def del_verified_purchase_N(
            self, hair_dryer_save_path=None,
            microwave_save_path=None,
            pacifier_save_path=None):
        '''
        This method deletes tuples whose
        verified_purchase is 'N',
        which means the customer didn't purchase
        the product but still made comments.
        :return:self.parameters
        '''
        # pattern: df =\
        # df.drop(df[df['three'] == 14].index)
        hair_dryer = self.hair_dryer
        microwave = self.microwave
        pacifier = self.pacifier

        self.hair_dryer = hair_dryer.drop(
            hair_dryer[
                (hair_dryer['verified_purchase'] != 'Y')
                &(hair_dryer['verified_purchase'] != 'y')
                ].index
        )
        self.microwave = microwave.drop(
            microwave[
                (microwave['verified_purchase'] != 'Y')
                &(microwave['verified_purchase'] != 'y')
                ].index
        )
        self.pacifier = pacifier.drop(
            pacifier[
                (pacifier['verified_purchase'] != 'Y')
                &(pacifier['verified_purchase'] != 'y')
                ].index
        )

        return (self.hair_dryer, self.microwave, self.pacifier)

    def del_helpful_votes_total_votes(self):
        '''
                This method deletes tuples
                whose helpful_votes/total_votes is '' or 0
                :return:None
        '''
        # pattern: df =\
        # df.drop(df[df['three'] == 14].index)
        hair_dryer = self.hair_dryer
        microwave = self.microwave
        pacifier = self.pacifier
        # | (hair_dryer['helpful_votes'] != '')

        self.hair_dryer = hair_dryer.drop(
            hair_dryer[
                (hair_dryer['helpful_votes'] == 0)
                |(hair_dryer['helpful_votes'] == '')
                ].index
        )
        self.microwave = microwave.drop(
            microwave[
                (microwave['helpful_votes'] == 0)
                |(hair_dryer['helpful_votes'] == '')
                ].index
        )
        self.pacifier = pacifier.drop(
            pacifier[
                (pacifier['helpful_votes'] == 0)
                |(hair_dryer['helpful_votes'] == '')
                ].index
        )
        return (self.hair_dryer, self.microwave, self.pacifier)

    def transDF(self):

        hair_dryer = self.hair_dryer
        microwave = self.microwave
        pacifier = self.pacifier

        self.hair_dryer = self.hair_dryer.stack()
        self.hair_dryer = self.hair_dryer.unstack(0)
        print("hair_dryer.stack()")
        print(self.hair_dryer)


        return (self.hair_dryer, self.microwave, self.pacifier)

    def mysort(self, str):
        '''
        :param str:sort by str(name of col)
        :return:
        '''
        self.hair_dryer =\
            self.hair_dryer.sort_values(str, ascending=True)
        self.microwave =\
            self.microwave.sort_values(str, ascending=True)
        self.pacifier =\
            self.pacifier.sort_values(str, ascending=True)
        # error!!!!!!
        return (self.hair_dryer, self.microwave, self.pacifier)

    def tranStr2Num(self, col_name=None):
        '''
        This method transform string to number by col
        :return: self.parameters
        '''
        class_mapping = {
            'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6,
            'H': 7, 'I': 8, 'J': 9, 'K': 10, 'L': 11, 'M': 12,
            'N': 13, 'O': 14, 'P': 15, 'Q':16, 'R': 17, 'S': 18,
            'T': 19, 'U': 20, 'V': 21, 'W':22, 'X': 23, 'Y': 24, 'Z': 25,
            'a': 26, 'b': 27, 'c': 28, 'd': 29, 'e': 30, 'f': 31,
            'g': 32, 'h': 33, 'i': 34, 'j': 35, 'k': 36, 'l': 37, 'm': 38,
            'n': 39, 'o': 40, 'p': 41, 'q': 42, 'e': 43, 's': 44, 't': 45,
            'u': 46, 'v': 47, 'w': 48, 'x': 49, 'y': 50, 'z': 51
        }
        # data[class] = data[class].map(class_mapping)
        # self.hair_dryer[col_name] =\
        # self.hair_dryer[col_name].map(class_mapping)
        self.hair_dryer[col_name]
        return (self.hair_dryer, self.microwave, self.pacifier)

    def cal_var_of_col(self, col_name):
        '''
        :param str: name of col
        :return:
        '''
        hair_dryer = self.hair_dryer[col_name]
        microwave = self.microwave[col_name]
        pacifier = self.pacifier[col_name]

        return (hair_dryer.var(), microwave.var(), pacifier.var())
        # return var

    def cal_avg_var_by_product_id(self):
        '''
        This method calculate average var of different product
        :return:
        '''
        path = "./preprocessedData/product_id/pacifier/pid_"
        n1 = 173    # hair_dryer
        n2 = 868    # microwave
        n3 = 403    # pacifier
        save_path = "./preprocessedData/var/var.xlsx"
        product_id = []
        var_list = []
        for i in range(n3):# hair_dryer:173, microwave: 868
            Si = str(i)
            print(Si)
            df = pd.DataFrame(pd.read_excel(path+Si+'.xlsx'))
            print(df['product_id'])
            product_id.append(df.iat[0, 2])      #
            var_list.append(df['star_rating'].var())#
        print("product_id")
        print(product_id)
        print("var_list")
        print(var_list)
        dict = {'product_id': product_id, 'star_var': var_list}
        new = pd.DataFrame(dict, columns=['product_id', 'star_var'])
        #, columns=['product_id', 'star_var'])
        new.to_excel(save_path, index=False)
        return None

    def select_rows(self):
        '''
        This methods are aiming at searching
        the 5 and 0 star_rating score,
        and use these rows to construct a matrix
        :return: three matrix
        '''
        hair_dryer = self.hair_dryer
        microwave = self.microwave
        pacifier = self.pacifier

        # create list to append data
        # hair_dryer
        length = len(microwave)
        print(length)
        print(microwave.iat[0, 2])
        self.list = list()
        for i in range(length - 1):
            if pacifier.iat[i + 1, 2]!=microwave.iat[i, 2]:
                self.list.append(microwave.iat[i + 1, 2])
        print(self.list)

        save_path = "./preprocessedData/product_id/"
        j = 0
        for x in self.list:
            Sj = str(j)
            # hair_dryer =\
            # self.hair_dryer.loc[self.hair_dryer['product_id'] == x]
            # hair_dryer.to_excel(
            # save_path+'hair_dryer/pid_'+Sj+'.xlsx', index=False
            # )
            microwave =\
                self.microwave.loc[self.microwave['product_id'] == x]
            microwave.to_excel(save_path + 'microwave/pid_' + Sj + '.xlsx',
                               index=False)
            # pacifier = self.pacifier.loc[self.pacifier['product_id'] == x]
            # pacifier.to_excel(
            # save_path + 'pacifier/pid_' + Sj + '.xlsx', index=False
            # )
            j += 1

        # hair_dryer = self.hair_dryer.loc[
        # (self.hair_dryer['star_rating'] == 0)
        # |(self.hair_dryer['star_rating'] == 5)]
        # microwave = self.microwave.loc[
        # (self.microwave['star_rating'] == 0)
        # |(self.microwave['star_rating'] == 5)]
        # pacifier = self.pacifier.loc[
        # (self.pacifier['star_rating'] == 0)
        # |(self.pacifier['star_rating'] == 5)]
        #--------------------------------------------#
        # hair_dryer = self.hair_dryer.loc[
        # (self.hair_dryer['star_rating'] != 0)
        # &(self.hair_dryer['star_rating'] != 5)]
        # microwave = self.microwave.loc[
        # (self.microwave['star_rating'] != 0)
        # &(self.microwave['star_rating'] != 5)]
        # pacifier = self.pacifier.loc[
        # (self.pacifier['star_rating'] != 0)
        # &(self.pacifier['star_rating'] != 5)]

        len1 = [len(hair_dryer), len(microwave), len(pacifier)]
        return (hair_dryer, microwave, pacifier, len1)

    def save_to_excel(self, save_path=None):
        '''
        :param save_path: a vector with path of
        self.hair_dryer, self.microwave, self.pacifier
        :return:self.parameters
        '''
        # save to excel
        self.hair_dryer.to_excel(save_path[0])
        self.microwave.to_excel(save_path[1])
        self.pacifier.to_excel(save_path[2])
        return (self.hair_dryer, self.microwave, self.pacifier)

    def get_class_by_col(self, col_name):
        '''
        This method can recognize how many
        statistic are the same in a col,
        and merge the rows with the same
        statistic in the certain col.
        :param col_name:
        :return:
        '''
        hair_dryer = self.hair_dryer
        microwave = self.microwave
        pacifier = self.pacifier

        self.hair_dryer.groupby(col_name).apply(list)


        # hair_dryer = hair_dryer.set_index(
        #     [col_name,
        #     hair_dryer.groupby(col_name).cumcount() + 1]
        # ).unstack().sort_index(level=1, axis=1)
        # self.hair_dryer.columns =\
        # self.hair_dryer.columns.map('{0[0]}_{0[1]}'.format())
        # self.hair_dryer.reset_index()

        # microwave = microwave.set_index(
        #     [col_name,
        #     microwave.groupby(col_name).cumcount() + 1]
        # ).unstack().sort_index(level=1, axis=1)
        # pacifier = pacifier.set_index(
        #     [col_name,
        #     pacifier.groupby(col_name).cumcount() + 1]
        # ).unstack().sort_index(level=1, axis=1)

        path_111 = "./preprocessedData/transformedData/"
        hair_dryer_save_path =\
            path_111 + "hair_dryer_" + col_name + ".xlsx"
        microwave_save_path =\
            path_111 + "microwave_" + col_name + ".xlsx"
        pacifier_save_path =\
            path_111 + "pacifier_" + col_name + ".xlsx"
        save_path = [
            hair_dryer_save_path,
            microwave_save_path,
            pacifier_save_path
        ]

        self.save_to_excel(save_path=save_path)

        # hair_dryer = hair_dryer[col_name]
        # microwave = microwave[col_name]
        # pacifier = pacifier[col_name]

        return (self.hair_dryer, self.microwave, self.pacifier)

    def get_parameters(self):
        return (self.hair_dryer, self.microwave, self.pacifier)

    def new_col(self, new_col_name=None):
        '''
        create a new col
        :param new_col_name: name of new col
        :return:
        '''
        self.hair_dryer[new_col_name] = None
        self.microwave[new_col_name] = None
        self.pacifier[new_col_name] = None
        return (self.hair_dryer, self.microwave, self.pacifier)

    def count_mean_char_num(self,
                            hair_dryer, microwave, pacifier,
                            len1=0):
        hd = hair_dryer['review_body']
        mc = microwave['review_body']
        pf = pacifier['review_body']

        hair_dryer['num'] = None
        microwave['num'] = None
        pacifier['num'] = None

        hair_dryer['num'] =\
            hair_dryer['review_body'].str.len()
        microwave['num'] =\
            microwave['review_body'].str.len()
        pacifier['num'] =\
            pacifier['review_body'].str.len()
        print("pacifier['review_body'].str.len()")
        print(pacifier['review_body'].str.len())
        print("pacifier['num']")
        print(pacifier['num'])
        str = "./preprocessedData/we.xlsx"
        # pacifier.to_excel(str)
        print("12138")
        print(pacifier)
        print("pacifier.iat[1, 11]")
        print(pacifier.iat[22, 11])
        # pacifier : clear blank data manually.

        # hair_dryer
        N = len(pacifier)
        print("N:", N)
        S_len = 0
        for i in range(N):
            S_len += pacifier.iat[i, 11]
        print("S_len:", S_len)
        Mean_len = S_len / N

        print("length")
        print(S_len)
        print("Mean len")
        print(Mean_len)



        # vec = [hair_dryer, microwave, pacifier]
        # # hair_dryer
        # N = [
        # len(hair_dryer), len(microwave), len(pacifier)
        # ]
        # S_len = [0, 0, 0]
        # Mean_len = [0, 0, 0]
        # for i in range(3):
        #     for j in range(N[0]):
        #         # S_len[i] += hair_dryer.iat[j, 11]
        #         S_len[i] += vec[i].iat[j, 11]
        #     Mean_len[i] = S_len[i] / N[0]

        print("121234")
        print(hair_dryer['num'])

        # # % time
        # # test['contentLen2'] = test['content'].str.len()
        # print("print")
        # print(hair_dryer['review_body'].str.len())

        return None
# class ProcessData(object)


# preprocessing()
def preprocessing(hair_dryer_path=None,
                  microwave_path=None, pacifier_path=None):
    '''
    :param hair_dryer_path: file_path of hairdryer.xlsx
    :param microwave_path: file_path of microwave.xlsx
    :param pacifier_path: file_path of pacifier.xlsx
    :return:Instance of the Class named ProcessData
    '''
    pre = ProcessData(hair_dryer_path=hair_dryer_path,
                      microwave_path=microwave_path,
                       pacifier_path=pacifier_path)

    pre.drop_col('marketplace')  # delete marketplace
    pre.drop_col('product_category')

    pre.del_helpful_votes_total_votes()

    path_112 = "./output_dataset/"
    hair_dryer_save_path = path_112+"hair_dryer.xlsx"
    microwave_save_path = path_112+"microwave.xlsx"
    pacifier_save_path = path_112+"pacifier.xlsx"

    pre.del_verified_purchase_N(
        hair_dryer_save_path=hair_dryer_save_path,
        microwave_save_path=microwave_save_path,
        pacifier_save_path=pacifier_save_path
    )
    pre.drop_col('verified_purchase')
    pre.drop_col('vine')

    # print(pre.hair_dryer)

    # save to excel
    pre.hair_dryer.to_excel(hair_dryer_save_path)
    pre.microwave.to_excel(microwave_save_path)
    pre.pacifier.to_excel(pacifier_save_path)
    return pre
# preprocessing()


def problemC_2d(hair_dryer_path=None,
                microwave_path=None, pacifier_path=None):
    '''
    :param hair_dryer_path: file_path of hairdryer.xlsx
    :param microwave_path: file_path of microwave.xlsx
    :param pacifier_path: file_path of pacifier.xlsx
    :return:Instance of the Class named ProcessData
    '''
    # ProCess_Data
    # import Preprocessed Data
    pcd = ProcessData(hair_dryer_path=hair_dryer_path,
                      microwave_path=microwave_path,
                      pacifier_path=pacifier_path)

    path_113 = "./preprocessedData/sortedData/"
    hair_dryer_save_path = path_113+"hair_dryer.xlsx"
    microwave_save_path = path_113+"microwave.xlsx"
    pacifier_save_path = path_113+"pacifier.xlsx"

    save_path = [hair_dryer_save_path,
                 microwave_save_path,
                 pacifier_save_path]

    # (hair_dryer, microwave, pacifier) =\
    # pcd.tranStr2Num('product_id')

    # We delete some dirty data manually before mysort().
    (hair_dryer, microwave, pacifier) =\
        pcd.mysort(str='product_id')

    # calculating var of the whole statistic
    # hair_dryer_var, microwave_var, pacifier_var =\
    # pcd.cal_var_of_col('star_rating')
    # print("var of hair_dryer is ", hair_dryer_var)
    # print("var of microwave is ", microwave_var)
    # print("var of pacifier is ", pacifier_var)

    # (hair_dryer, microwave, pacifier, len1) =\
    # pcd.select_rows()
    pcd.cal_avg_var_by_product_id()
    # (hair_dryer_mean_num, microwave_mean_num, pacifier_mean_num) =\
    # pcd.count_mean_char_num(
    #         hair_dryer=hair_dryer, microwave=microwave,
    #         pacifier=pacifier, len1=len1
    # )

    # pcd.get_class_by_col('product_id')

    # pcd.save_to_excel(save_path=save_path)

    return pcd

def problemC_2b(
        hair_dryer_path=None, microwave_path=None,
        pacifier_path=None):
    pcd = ProcessData(hair_dryer_path=hair_dryer_path,
                      microwave_path=microwave_path,
                      pacifier_path=pacifier_path)

    hair_dryer_save_path =\
        "./preprocessedData/sortedData/hair_dryer.xlsx"
    microwave_save_path =\
        "./preprocessedData/sortedData/microwave.xlsx"
    pacifier_save_path =\
        "./preprocessedData/sortedData/pacifier.xlsx"

    save_path = [hair_dryer_save_path,
            microwave_save_path, pacifier_save_path]

    s_dataset = "./dataset/s_dataset/s"
    to_data_train = "./data/train/s"
    to_data_test = "./data/test/s"
    print("start")
    for i in range(5):      #
        Si = str(i+1)
        df =\
            pd.DataFrame(
                pd.read_excel(
                    s_dataset + Si + '.xlsx'))
        # df = df['review_body']
        for j in range(len(df)):    #
            print("(i: ", i, "j: ", j, ")")
            Sj = str(j+1)
            train_path =\
                to_data_train + Si + '/s' + Sj + '.txt'
            test_path =\
                to_data_test + Si + '/s' + Sj + '.txt'
            f1 = open(train_path, 'w',
                      encoding='unicode')
            # f1 = codecs.open(train_path,)
            print(df.iat[j, 10], file=f1)
            f1.close()
            f2 = open(test_path, 'w',
                      encoding='unicode')
            print(df.iat[j, 10], file=f2)
            f2.close()

            # f = codecs.open(file, 'r', in_enc)
            # new_content = f.read()
            # codecs.open(file, 'w', out_enc).write(new_content)
            # Sj = str(j)
            # df_ = df.iat[j, 0]
            # df_.to_csv(to_data_test + Sj + '.xlsx')
            # df_.to_csv(to_data_train + Sj + '.xlsx')


    return None


# __main__
if __name__ == "__main__":
    print("main.start...")

    '''
    pre-processing modules
    '''
    # hair_dryer_path = "./dataset/hair_dryer.xlsx"
    # microwave_path = "./dataset/microwave.xlsx"
    # pacifier_path = "./dataset/pacifier.xlsx"

    # preprocessing(hair_dryer_path=hair_dryer_path,
    # microwave_path=microwave_path, pacifier_path=pacifier_path)

    # We needs to modify/delete col 0 in these excel files manually
    hair_dryer_path = \
        "./preprocessedData/hair_dryer.xlsx"
    microwave_path = \
        "./preprocessedData/microwave.xlsx"
    pacifier_path = \
        "./preprocessedData/pacifier.xlsx"

    # save_path =\
    # [hair_dryer_path, microwave_path, pacifier_path]

    # problemC_2d(hair_dryer_path=hair_dryer_path,
    #              microwave_path=microwave_path,
    #              pacifier_path=pacifier_path)
    problemC_2b(hair_dryer_path=hair_dryer_path,
                microwave_path=microwave_path,
                pacifier_path=pacifier_path)

    print("main.end...")
# __main__
```

```
from keras.models import Sequential
from keras.layers.core import Dense, Dropout
from keras.layers.embeddings import Embedding
from keras.layers.recurrent import LSTM

from keras.preprocessing import sequence
from keras.preprocessing.text import Tokenizer

import re   #
import os

import pandas as pd


# class PreprocessData():
class PreprocessData():
    def __init__(self):
        print("create")
        self.y_train, self.train_text = \
            self.read_files("train")
        self.y_test, self.test_text = \
            self.read_files("test")
        print(type(self.train_text))    # class 'list'
        train_text = pd.DataFrame(
            self.train_text)
        print(train_text)
        return None

    #
    def rm_tags(self, text):  #
        re_tag = re.compile(r'<[^>]+>')  #
        return re_tag.sub('', text)  #

    # read dataset content
    # creates read_files function
    # for reading dataset's folder
    def read_files(self, filetype):
        path = "data/"
        file_list = []

        s1_path = path + filetype + "/s1/"
        for f in os.listdir(s1_path):
            file_list += [s1_path + f]
        s2_path = path + filetype + "/s2/"
        for f in os.listdir(s2_path):
            file_list += [s2_path + f]
        s3_path = path + filetype + "/s3/"
        for f in os.listdir(s3_path):
            file_list += [s3_path + f]
        s4_path = path + filetype + "/s4/"
        for f in os.listdir(s4_path):
            file_list += [s4_path + f]
        s5_path = path + filetype + "/s5/"
        for f in os.listdir(s5_path):
            file_list += [s5_path + f]

        print('read', filetype, 'files:',
              len(file_list))
        print("file_list")
        print(file_list)

        n = [871, 437, 549, 879, 2712]#5448

        all_labels = ([1] * n[0] +
                      [2] * n[1] + [3] * n[2] +
                      [4] * n[3] + [5] * n[4])########
        all_texts = []
        i = 0
        for fi in file_list:
            i += 1
            print("i: ", i)    #s1,s2--1308
            with open(fi, encoding='utf8') as file_input:
                all_texts += [
                    self.rm_tags(
                " ".join(file_input.readline()))]
        return all_labels, all_texts

    def preprocess_data(self):
        # y_train, train_text = self.read_files("train")
        # y_test, test_text = self.read_files("test")
        # token creation
        token = Tokenizer(num_words=3800)########
        token.fit_on_texts(self.train_text)
        # transform text to num list
        # train_text = self.train_text
        # test_text = self.test_text
        x_train_seq = \
            token.texts_to_sequences(self.train_text)
        x_test_seq = \
            token.texts_to_sequences(self.test_text)
        # all sequence are 100 length
        x_train = sequence.pad_sequences(
            x_train_seq, maxlen=380)#######
        x_test = sequence.pad_sequences(
            x_test_seq, maxlen=380)#######
        return x_train, self.y_train, x_test, self.y_test
# class PreprocessData()


# class MyLSTM():
class MyLSTM():
    def __init__(self, x_train, y_train):
        self.x_train = x_train
        self.y_train = y_train
        self.model = Sequential()

    def create_My_LSTM(self):
        self.model.add(Embedding(
            output_dim=32,
            input_dim=3800,
            input_length=380
        ))
        self.model.add(Dropout(0.2))
        self.model.add(LSTM(32))
        self.model.add(Dense(
            units=256, activation='relu'))
        self.model.add(Dropout(0.2))
        self.model.add(Dense(
            units=1, activation='sigmoid'))
        return self.model

    def print_model_summary(self):
        print(self.model.summary())
        return self.model

    def fit_my_lstm(self):
        self.model.compile(
            loss='binary_crossentropy',
            optimizer='adam',
            metrics=['accuracy']
        )
        train_history = self.model.fit(
            self.x_train, self.y_train,
            batch_size=100, epochs=30,
            verbose=2, validation_split=0.2
        )
        return self.model, train_history

    def save_my_lstm_model(
            self, filepath="./model/MyLSTM_for_MCM_C.h5"):
        self.model.save(filepath)
        return self.model

    def load_my_lstm_model(self,
                filepath="./model/MyLSTM_for_MCM_C.h5"):
        self.model.load_weights(filepath)
        return self.model
# class MyLSTM():


def train(x_train, y_train, x_test, y_test):
    print()
    my_lstm = MyLSTM(x_train, y_train)
    my_lstm.create_My_LSTM()
    my_lstm.fit_my_lstm()
    my_lstm.print_model_summary()
    model = my_lstm.save_my_lstm_model()
    print("evaluate model")
    scores = model.evaluate(
        x_test, y_test, verbose=1)
    print(scores[1])
    return model


def test(x_train, y_train, x_test, y_test):
    print("test start")
    my_lstm = MyLSTM(x_train, y_train)
    my_lstm.create_My_LSTM()
    model = my_lstm.load_my_lstm_model()
    my_lstm.print_model_summary()
    # start to predict
    predict = model.predict_classes(x_test)
    print(predict[:10])

    # predict
    predict_classed = predict.reshape(-1)
    predict_vector = \
        pd.DataFrame(predict_classed)
    path = "./data/predict_vector.xlsx"
    predict_vector.to_excel(path, index=False)

    print(predict_classed)
    print("test end")
    return predict_classed


SentimentDict = {
    1: "star_rating: 1",
    2: "star_rating: 2", 3: "star_rating: 3",
    4: "star_rating: 4", 5: "star_rating: 5"
}


def display_test_Sentiment(i,
        test_text, y_test, predict_classed):
    print(test_text[i])
    print(
        'label for real: ',
        SentimentDict[y_test[i]],
        'prediction results: ',
        SentimentDict[predict_classed[i]], "    "
    )
    return 0


if __name__ == "__main__":
    print("__main__.start...")
    # pre-processing
    preprocessData = PreprocessData()
    print("preprocessData")
    x_train, y_train, x_test, y_test = \
        preprocessData.preprocess_data()
    '''
     print(x_train)
    df = pd.DataFrame(x_train)
    print("x_train")
    path = "./data/x_train.xlsx"
    df.to_excel(path, index=False)
    print("finished")
    '''
    # print("preprocessData.train_text")
    # print(preprocessData.test_text)
    # df = pd.DataFrame(
    # preprocessData.test_text)
    # print("x_train")
    # path = "./data/test_text.xlsx"
    # df.to_excel(path, index=False)
    # print("finished")

    train(x_train, y_train, x_test, y_test)
    print("predict_classed")
    predict_classed = test(x_train,
                y_train, x_test, y_test)
    print("display_test_Sentiment")
    display_test_Sentiment(0,
        preprocessData.test_text,
        preprocessData.y_test, predict_classed)

    print("__main__.end...")
```

```
# Analytic Hierarchy Process

import numpy as np


# class myAHP
class myAHP:
    # __init__(self)
    def __init__(self):
        # print("__init__(self).start...")
        self.RI_dict = {
             1: 0,       2: 0,        3: 0.58,
            4: 0.90,
            5:  1.12,    6: 1.24,     7: 1.32,
             8: 1.41,     9: 1.45,     10: 1.51,
            11: 1.54,
            12: 1.56,    13: 1.57,    14: 1.59,
            15: 1.59,   16: 1.60,    17: 1.61,
            18: 1.62,
            19: 1.63,   20: 1.64
        }
        self.comparison_matrix = np.array([
            [0.50, 0.77, 0.48, 0.68, 0.47, 0.38],
            [0.23, 0.50, 0.28, 0.45, 0.32, 0.23],
            [0.52, 0.72, 0.50, 0.75, 0.48, 0.40],
            [0.32, 0.55, 0.25, 0.50, 0.23, 0.20],
            [0.53, 0.68, 0.52, 0.77, 0.50, 0.48],
            [0.62, 0.77, 0.60, 0.80, 0.52, 0.50]
        ])
    # __init__(self)

    # get_w(array)
    def get_w(self, array):
        print("get_w(array).start...")

        row = array.shape[0]

        a_axis_0_sum = array.sum(axis=0)
        # print(a_axis_0_sum)

        b = array / a_axis_0_sum  # new matrix b
        # print(b)

        b_axis_0_sum = b.sum(axis=0)
        b_axis_1_sum = b.sum(axis=1)  #
        print("b_axis_1_sum")
        print(b_axis_1_sum)

        w = b_axis_1_sum / row  #

        nw = w * row

        AW = (w * array).sum(axis=1)
        print("AW:")
        print(AW)

        max_max = sum(AW / (row * w))
        print("max_max:")
        print(max_max)

        # calculating
        print("calculating")
        CI = (max_max - row) / (row - 1)
        print("CI:")
        print(CI)
        print("row:")
        print(row)
        print("RI_dictt[row]")
        print(self.RI_dict[row])
        print("/*-------------------------*/")
        CR = CI / self.RI_dict[row]
        print("over!")

        if CR < 0.1:
            print("CR:")
            print(CR)
            # print(round(CR, 3))
            print('success!!')
            return w
        else:
            print(round(CR, 3))
            print('please modify your data')

        print("get_w(array).end...")
    # get_w(array)

    # def main(array)
    def mainAlgorithm(self, array):
        print("main(array).start...")

        if type(array) is np.ndarray:
            result = self.get_w(array)
            return result
        else:
            print(
            'please input an instance of numpy'
            )

        print("main(array).start...")
    # def main(array)

    # comparison_economic_function()
    def comparison_function(self):
        print(
        "comparison_function(self).start..."
        )

        e = np.array([
            [1,   2,   7,   5],
            [1/2, 1,   4,   3],
            [1/7, 1/4, 1,   2],
            [1/5, 1/4, 1/2, 1]
        ])

        a = self.comparison_matrix
        b = self.comparison_matrix
        c = self.comparison_matrix
        d = self.comparison_matrix
        f = self.comparison_matrix

        # print("e")
        e = self.mainAlgorithm(e)
        # print("e")


        # print("a")
        a = self.mainAlgorithm(a)
        # print("a")

        # print("b")
        b = self.mainAlgorithm(b)
        # print("b")
        # print("c")
        c = self.mainAlgorithm(c)
        # print("c")

        # print("d")
        d = self.mainAlgorithm(d)
        # print("d")

        try:
            print("try")

            # res = np.array([a, b, c, d, f])
            res = np.array([a, b, c, d])
            # print("res:")
            # print(res)

            resT = np.transpose(res)
            # print("resT:")
            # print(resT)
            # print("e:")
            # print(e)
            print("ret:")
            ret = (resT * e).sum(axis=1)
            print(ret)
            print("try.end: ret")
            return ret
        except TypeError:
            print('data error')

        print("comparison_function(self).end...")
    # comparison_function()
# class myAHP


# __main__
if __name__ == '__main__':
    print("MyAHP.py__main__.start...")
    ahp = myAHP()
    ahp.comparison_function()
    print("MyAHP.py__main__.end...")
# __main__
```

```
import os
import codecs
import chardet

def list_folders_files(path):
    """
    """
    list_folders = []
    list_files = []
    for file in os.listdir(path):
        file_path = os.path.join(path, file)
        if os.path.isdir(file_path):
            list_folders.append(file)
        else:
            list_files.append(file)
    return (list_folders, list_files)
```

```
def convert(file, in_enc="GBK", out_enc="UTF-8"):
    """
    """
    in_enc = in_enc.upper()
    out_enc = out_enc.upper()
    try:
        print("convert [ " + file.split('\\')[-1] +
              " ].....From " + in_enc + " --> " + out_enc)
        f = codecs.open(file, 'r', in_enc)
        new_content = f.read()
        codecs.open(file, 'w', out_enc).write(new_content)
    # print (f.read())
    except IOError as err:
        print("I/O error: {0}".format(err))

if __name__ == "__main__":
    print("__main__.start...")
    # path = r'G:\Temp'
    # path = "./s1"
    # path = "./data/test/s1"
    # path = "./data/test/s2"
    # path = "./data/test/s3"
    # path = "./data/test/s4"
    # path = "./data/test/s5"

    # path = "./data/train/s1"
    # path = "./data/train/s2"
    # path = "./data/train/s3"
    # path = "./data/train/s4"
    path = "./data/train/s5"
    (list_folders, list_files) = \
        list_folders_files(path)

    print("Path: " + path)
    for fileName in list_files:
        filePath = path + '\\' + fileName
        with open(filePath, "rb") as f:
            data = f.read()
            codeType = \
                chardet.detect(data)['encoding']
            convert(filePath, codeType, 'UTF-8')
    print("__main__.end...")
```

```
import numpy as np
from tqdm import trange #


def sigmoid(x): #
    return 1 / (1 + np.exp(-x))


def sigmoid_derivative(x):  #
    return sigmoid(x) * (1 - sigmoid(x))


class NeuralNetwork:
    def __init__(self, layers): #
        self.activation = sigmoid   #
        self.activation_deriv = sigmoid_derivative  #
        self.weights = []   #
        self.bias = []  #
        for i in range(1, len(layers)): #
            self.weights.append(
                np.random.randn(layers[i-1], layers[i]))
            self.bias.append(np.random.randn(layers[i]))

    def fit(self, x, y, learning_rate=0.2, epochs=3):   #
        x = np.atleast_2d(x)
        n = len(y)  #
        p = max(n, epochs)  #
        y = np.array(y)

        for k in trange(epochs * n):    #
            if (k+1) % p == 0:
                learning_rate *= 0.5    #
            a = [x[k % n]]  #
            #
            for lay in range(len(self.weights)):
                a.append(
                    self.activation(
                        np.dot(
                            a[lay],
                            self.weights[lay])+self.bias[lay]))
            #
            label = np.zeros(a[-1].shape)
            label[y[k % n]] = 1 #
            error = label - a[-1]   #
            deltas = [error * self.activation_deriv(a[-1])] #

            layer_num = len(a) - 2  #
            for j in range(layer_num, 0, -1):
                deltas.append(deltas[-1].dot(
            self.weights[j].T) * self.activation_deriv(a[j]))   #
            deltas.reverse()
            for i in range(len(self.weights)):  #
                layer = np.atleast_2d(a[i])
                delta = np.atleast_2d(deltas[i])
                self.weights[i] += \
                    learning_rate * layer.T.dot(delta)
                self.bias[i] += learning_rate * deltas[i]

    def predict(self, x):   #
        a = np.array(x, dtype=np.float)
        for lay in range(0, len(self.weights)): #
            a = self.activation(
            np.dot(a, self.weights[lay]) + self.bias[lay])
        a = list(100 * a/sum(a))    #
        i = a.index(max(a)) #
        per = []    #
        for num in a:
            per.append(str(round(num, 2))+'%')
        return i, per
```

```
# from NeuralNetwork import NeuralNetwork
```

```
import neuralnetwork as NeuralNetwork
import numpy as np
import pickle
import csv


def train():
    file_name = 'data/train.csv'    #
    y = []
    x = []
    y_t = []
    x_t = []
    with open(file_name, 'r') as f:
        reader = csv.reader(f)
        header_row = next(reader)
        print(header_row)
        for row in reader:
            if np.random.random() < 0.8:
                y.append(int(row[0]))
                x.append(list(map(int, row[1:])))
            else:
                y_t.append(int(row[0]))
                x_t.append(list(map(int, row[1:])))
    len_train = len(y)
    len_test = len(y_t)
    print(
        'training set% d testing  set%d' %
        (len_train, len_test))
    x = np.array(x)
    y = np.array(y)
    nn = NeuralNetwork([784, 784, 10])  #
    nn.fit(x, y)
    file = open('NN.txt', 'wb')
    pickle.dump(nn, file)
    count = 0
    for i in range(len_test):
        p, _ = nn.predict(x_t[i])
        if p == y_t[i]:
            count += 1
    print('accuracy', count/len_test)


def mini_test():
    x = [[0, 0], [0, 1], [1, 0], [1, 1]]
    y = [0, 1, 2, 3]
    nn = NeuralNetwork([2, 4, 16, 4])
    nn.fit(x, y, epochs=10000)
    for i in x:
        print(nn.predict(i))


# mini_test()
train()
```