

# Commercial Data Analysis Based on Online Reviews

## Summary

For the products sold on the Internet, customers can leave reviews under the corresponding products and rate the products. As manufacturers of commodities, there is an urgent need to know how to obtain useful information from them. The work of our team is divided into four steps.

In the first step, we decided to use the bag of words model to translate known text reviews into a sparse matrix to save them, classify the reviews based on the star rating of each review, and then call our improved logistic regression model for training. In the second step, we improved our model based on the first model, and used words embedding in conjunction with the distributed cc for text analysis, and then still called our improved logistic regression model. In the third step, we analyzed the star rating changes of each kind of commodities through the Time-Series model. Then we calculated the correlation coefficient and the autocorrelation coefficient respectively, and finally used ARIMA for modeling. In the fourth step, we analyzed combinations of the review text and the Time-Series to analyze the temporal correlation between ratings and reviews.

The first model was trained to overfitting, and the second model was trained to maintain the correct rate while avoiding overfitting, which means that the second model is more suitable for analysis of reviews. The third model conforms to the expectation in the test and fitting, and shows the change of commodity market reputation over time. In the fourth step, there is a certain error between the analysis and the reality.

Through the second model, we can make predictions and judgments on product reputation based on real-time updated user reviews. In addition, the parameters of our model represent the influencing factors of the corresponding words or phrases on the product, which can help manufacturers identify necessary Design functions. Our time series analysis enables merchants to perceive the changes of products' reputation over time, and further master the potential relationship between ratings and reviews.

Keywords: Words embedding; Logistic regression; Time-Series; Online shopping

## Contents

1	Introduction.....	2
1.1	Background.....	2
2	Choice and Basic Settings of the Model.....	2
2.1	Choice of the Model.....	2
	Bag-of-Words & Logistic Regression.....	2
	Word Embeddings & Glove & Logistic Regression.....	3
	Time Series .....	3
2.2	Basic settings of the Model.....	3
3	Details of the Model.....	4
3.1	Bag of Words & Logistic Regression .....	4
	Bag of words .....	4
	Generate a logistic regression .....	6
3.2	Word Embeddings & Glove & Logistic Regression.....	7
3.3	Time Series Analysis.....	10
	Dataset Sorting.....	10
	Stationary Series Judgement.....	11
	Further Analysis of Previous and Subsequent reviews .....	15
4	Model Extension and Simulation Analysis .....	16
4.1	Problem1: Data Measures for Reviews.....	16
4.2	Problem2: Time-based Measures and Patterns .....	16
4.3	Problem3: Indicate a Potentially Successful or Failing Product.....	16
4.4	Problem4: Previous Ratings and Subsequent Reviews.....	17
4.5	Problem5: Specific Word and Ratings .....	17
5	Strengths and Weaknesses.....	17
5.1	Strengths .....	17
5.2	Weaknesses .....	17
5.3	Further Work .....	18
6	A Letter .....	18
7	References.....	20
8	Appendix.....	20

# 1 Introduction

## 1.1 Background

When people shop online, it's just inevitable to see hundreds of thousands of reviews from all kinds of people. Some of the reviews do reflect the true quality of the commodity, while some are simply written for commercial purpose.

Starting in 2015, Amazon began weighting stars given by reviewers based on factors including how recent they are and whether they come from “verified” purchasers. These approaches enable both sellers and buyers to set up standards for better use of reviews.

For sellers, since the reviews contain a mass of information like the edges of competitors' commodities and the qualities that attract consumers most, proper ways are definitely needed for further analysis. We are asked to analyze the reviews of three kinds of commodities from Amazon, identify proper data measures, find the relationships between different parameters and estimate the potential tendency of the sales.

## 2 Choice and Basic Settings of the Model

### 2.1 Choice of the Model

#### Bag-of-Words & Logistic Regression

A problem with modeling text is that it is messy, and techniques like machine learning algorithms prefer well defined fixed-length inputs and outputs. Moreover, machine learning algorithms cannot work with raw text directly; the text must be converted into digital numbers. Specifically, vectors of numbers.<sup>[1]</sup>

Under this circumstance, we need to figure out a way to turn the text words into

some sort of data which can be executed by the computer. In language processing, the vectors  $x$  are derived from textual data, in order to reflect various linguistic properties of the text.<sup>[2]</sup> This kind of method is called feature extraction. As far as I know, many NLP algorithm is fit with this problem. So we decided to use the Bag of Words to handle with this problem.

## Word Embeddings & Glove & Logistic Regression

In short, word embedding algorithm is a method to transform the natural words into vector or matrix so that our computer can accept and analyze the information through the data. Word embedding is a kind of technique which means we can create a mapping between words and the unique vectors.

The Global Vectors algorithm is an upgraded method from the word2vec, in the study, we find out that glove can study word vectors efficiently. However, glove needed to be downloaded as a corpus. Glove is a method that combines the global statistics of matrix decomposition techniques such as LSA with the local context based on the word2vec algorithm. Instead of using windows to define the local contexts, Glove uses statistics from the entire text corpus to construct a clear word context or co-occurrence matrix. The result is a new model that, to some extent, leads to a better word embedding result.

## Time Series

A time series is a statistical technique that is used to discover the variation patterns of a group of data in chronological order and to make predictions.

## 2.2 Basic settings of the Model

- The amount of reviews reflects the sales status of the certain kind of commodity, which means we can judge the development of sales directly from the reviews.
- Before generating the logistic regression, we divide the star rating into two labels. If the star rating is lower than 3 stars, we define this review as a useless review and assign this review 0. On the contrary, we assign 1 to the rest of the reviews, which get higher than 3 stars. The generated file is named as label.csv.
- When processing the relationships between time and rate, we have combined the rates

---

by month. Years with less than 20 reviews are ignored.

- We have eliminated the wrong commodities' data during the processing of pacifiers' reviews.

## 3 Details of the Model

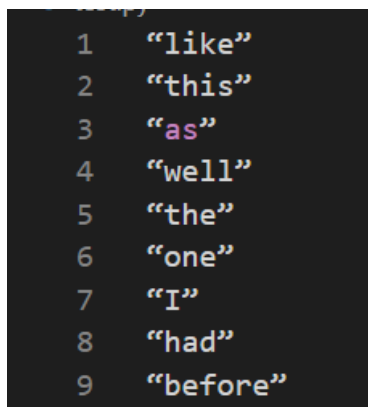
### 3.1 Bag-of-Words & Logistic Regression

#### Bag of words

First of all, once we obtain a review, we can make a list of all of the words in our model vocabulary. Take the hair dryer as an example, there's a review as follows:

“Like this as well as the one I had before.”

This sentence includes 10 words in total. However, ‘as’ is the word which appears twice in this sentence. We will pick out the unique words out. In this sentence, we pick out 9 words, they are:



```
1  "like"
2  "this"
3  "as"
4  "well"
5  "the"
6  "one"
7  "I"
8  "had"
9  "before"
```

Figure 1

The next step is to score the words in each document. This step aims to change the word into different variables so that our machine learning model can handle with this kind of data. Because there are 9 words in total, we can use a document representation of these 9 words, with one position in the vector to score each word. Then we write down the presence of words with a Boolean value, 0 for not exist, 1 for exist. Using the words listed above in our vocabulary at random, we can step through the whole document (“Like this as well as the one I had before.”) and translate it into a binary vector.

The scores of the document would be assigned as follows:

```
1  "like"=1
2  "this"=1
3  "as"=0
4  "well"=0
5  "the"=0
6  "one"=1
7  "I"=1
8  "had"=0
9  "before"=0
```

Figure 2

This sentence is a binary vector and will look as follows:

```
1  [1,1,0,0,0,1,1,0,0]
```

Figure 3

The original order of the words in the sentences has seemingly been discarded. We have a general method to extract features from the documents in our corpus, which can be used for modeling. The newly read comment overlaps the vocabulary of the known words, but it may contain words outside the vocabulary. At this time, we can still code it, in which only the occurrence of the known words is scored, and the unknown words that do not appear are ignored. In the data of the hair dryer set, we eventually get over 13,000 variables in total, in other words, we make a 13,000\*11471 list. 11471 represents the raw number of the data. And the length of the document vector is equal to the number of known words.

Detailed steps

### 1. Pipeline for generating Bow features

- Lower the text
- Tokenize the text and remove the punctuation
- Remove the words that contain numbers
- Remove the stop words
- Assign a tag to every word to define if it corresponds to a noun, a verb etc. using the WordNet lexical database
- Lemmatize the text: transform every word into their root form.
- Remove words with only one letter
- Modify mis-spelled
- Check whether English words

During the generating Bow features process, it is necessary to remove meaningless words, including words contain numbers, stop words, letters. After that, we could transform the words into root form so that we could merge several words which have the same meaning. Before we check whether it's an English word, we should try to modify mis-spelled, which enable us to keep these useful words. The reason we add the process to check whether English words is that other language might rarely show up in these reviews, which might be futile if we generate some features for these words. After that, we use the inverse document frequency to generate the features for each word.

Besides, often a simple bigram approach is better than a 1-gram bag-of-words model for documentation classification. A bag-of-bigrams representation is much more powerful than bag-of-words, and in many cases proves very hard to beat.<sup>[3]</sup> So, we use both single words and 2-word pairs are included in the process of generating features.

### Generate a logistic regression

Before generating the logistic regression, we divide the star rating into two labels. If the star rating is lower than 3 stars, we define this review as a useless review and assign this review 0. On the contrary, we assign 1 to the rest of the reviews, which get higher than 3 stars. The generated file is named as label.csv.

We used inverse of regularization strength as our hyperparameter to tune. Firstly, we generated a grid for the inverse of regularization strength, which returns 31 numbers spaced between -9 and 6 evenly on a log scale. It is a wide enough range, which includes some good enough hyperparameters. We also used 3-fold cross validation to figure out the best hyperparameter. We used Liblinear Solver to train our model.

### Model evaluation

The results of tuning hyperparameter have been shown in the following graphs.

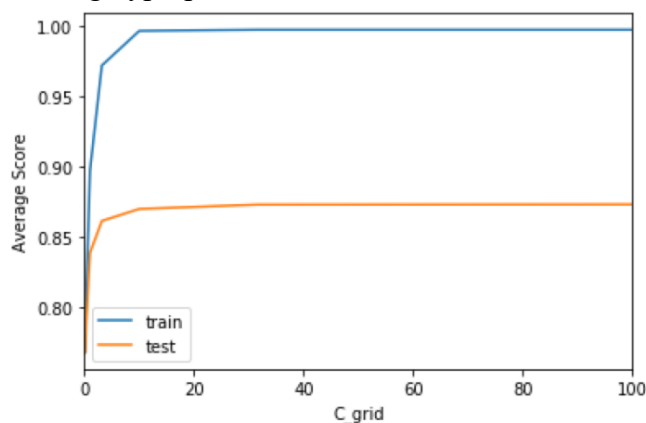


Figure 4: hair dryer

We can zoom the x axis so that the result will be easier to reflect.

The accuracy of the test data set increases until pass a certain threshold, and then it drops; whereas the accuracy of the train data set keeps increasing, which is close to one. The evidence of overfitting could be found in these results. When the inverse of regularization strength increases, the less penalty has been assigned to the large parameters, which weaken the model's ability to prevent overfitting. Hence, we could find that the training accuracy keeps increasing, meanwhile, the test accuracy decreases.

### Disadvantages:

The bag-of-words model ignores the location of the word. Location information is a very important information in text. The location of the word is different from the semantics such as "he likes eating hamburger" and "he loves hamburger" review.

Although the encoding method calculates the number of times a word appears in the text, it cannot distinguish between common words such as "I", "yes", etc.

Because this model is obviously overfitting here, we can come to the conclusion that this model can't fit this star rating and reviews model. **A new model has to be established.**

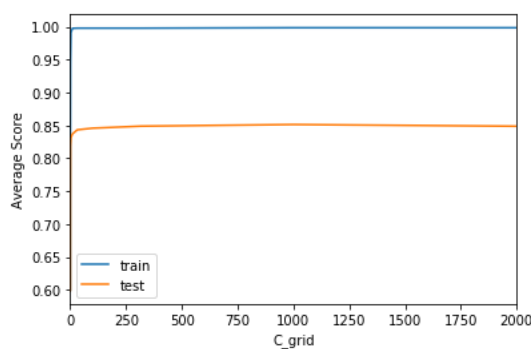


Figure 5: microwave

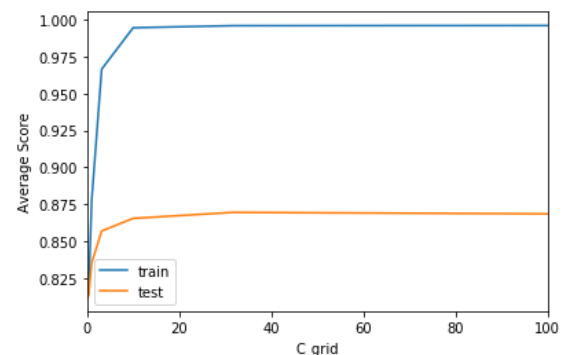


Figure 6: pacifier

## 3.2 Word Embeddings & Glove & Logistic Regression

### Detailed steps:

#### 1. Pipeline for generating Word Embeddings features

- Lower the text
- Tokenize the text and remove the punctuation
- Remove the words that contain numbers



- 
- Remove the stop words
  - Assign a tag to every word to define if it corresponds to a noun, a verb etc. using the WordNet lexical database
  - Lemmatize the text: transform every word into their root form.
  - Remove words with only one letter
  - Modify mis-spelled
  - Check whether English words

During the generating Word Embeddings features process, it is necessary to remove meaningless words, including words contain numbers, stop words, letters. After that, we could transform the words into root form so that we could merge several words which have the same meaning. Before we check whether it's an English word, we should try to modify mis-spelled, which enable us to keep these useful words. The reason we add the process to check whether English words is that other language might rarely show up in these reviews, which might be futile if we generate some features for these words. Then, we used the given set of pre-trained embedding vectors to generate a 1 by 50 vector for each word and sum each feature in the vector up across all the words in a certain review.

## 2. Generate Logistic Regression Model

Before generating the logistic regression, we divide the star rating into two labels. If the star rating is lower than 3 stars, we define this review as a useless review and assign this review 0. On the contrary, we assign 1 to the rest of the reviews, which get higher than 3 stars. The generated file is named as label.csv.

We used inverse of regularization strength as our hyperparameter to tune. Firstly, we generated a grid for the inverse of regularization strength, which returns 31 numbers spaced between -9 and 6 evenly on a log scale. It is a wide enough range, which includes some good hyperparameters. We also used 3-fold cross validation to figure out the best hyperparameter. We used Liblinear Solver to train our model. It's the same as the former model.

### Model evaluation

The results of tuning hyperparameter have been shown in the following graphs.

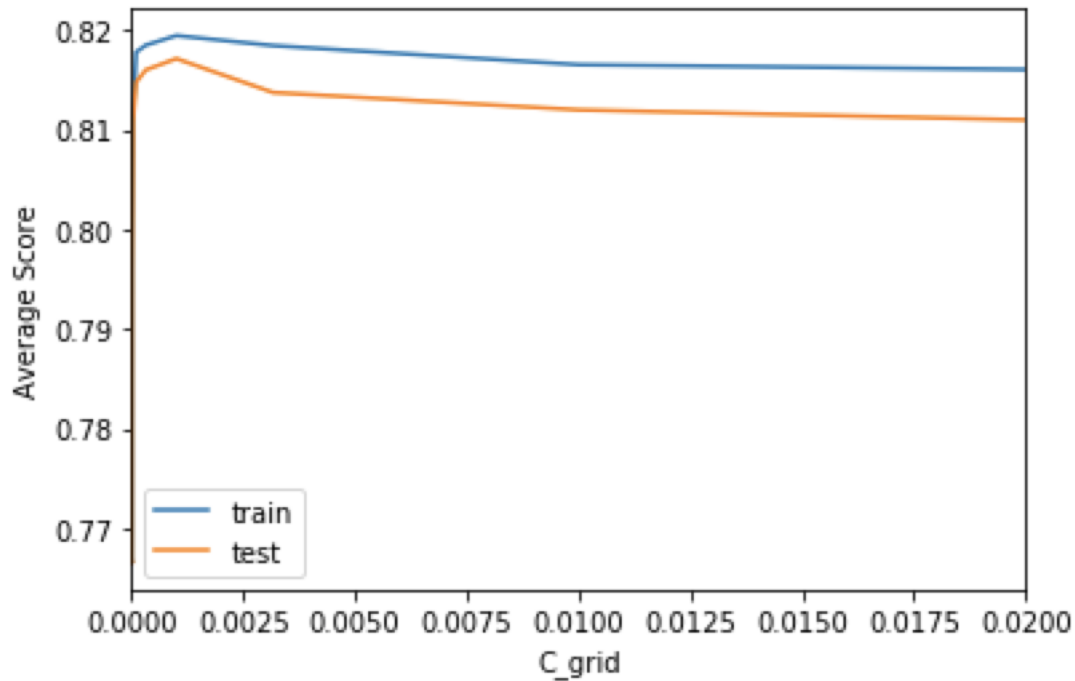


Figure 7: hair dryer

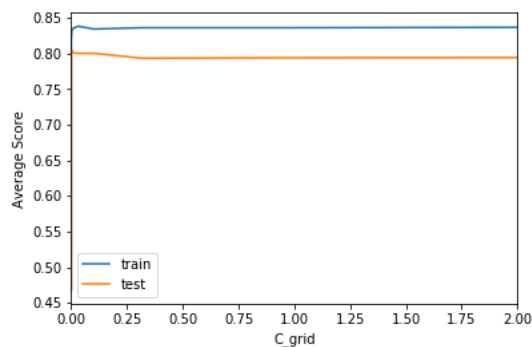


Figure 8: pacifier

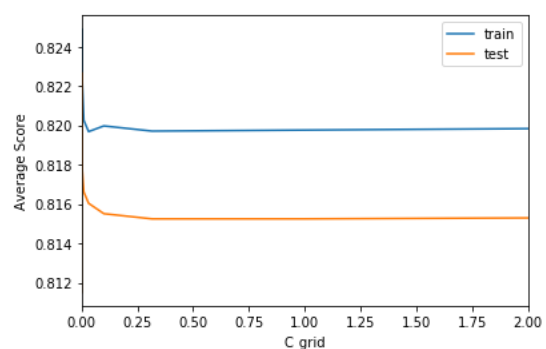


Figure 9: microwave

The accuracy of the train and test data set increases until pass a certain threshold, and then it drops. It's clear that this model successfully prevents the overfitting result as before. The accuracy approaches to 0.817.

### What can the models do?

Our model can be used as a measure of reviews, helping Amazon to rate the future reviews of the three products, such as pacifier hair dryer and microwave oven, and helping Amazon to quickly analyze the impact of this review on product reputation. Where the parameters of our model represent the degree to which each word affects the sentence score.

Table 1

	# of Features	Best_C of BoW	Error Rate of Validation Data set	Best_C of WE	Error Rate of Validation Data set
Hair dryer	130554	315227.77	0.18	100000	0.22
Microwave	45921	1000	0.16	1000000	0.20
Pacifier	204576	1	0.13	1	0.18

### 3.3 Time Series Analysis

#### Dataset Sorting

To generate the Time-Series, we classified the data by month and years with too small sample size were excluded. And then the image was made by computer.

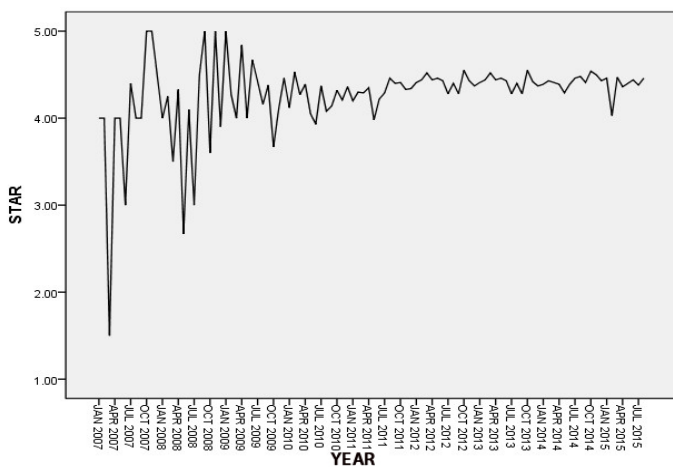


Figure 10: The Time-Series image of pacifier

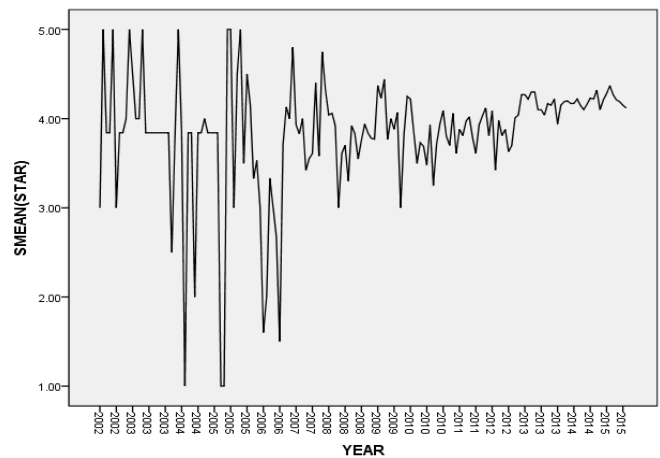


Figure 11: The Time-Series image of hairdryer

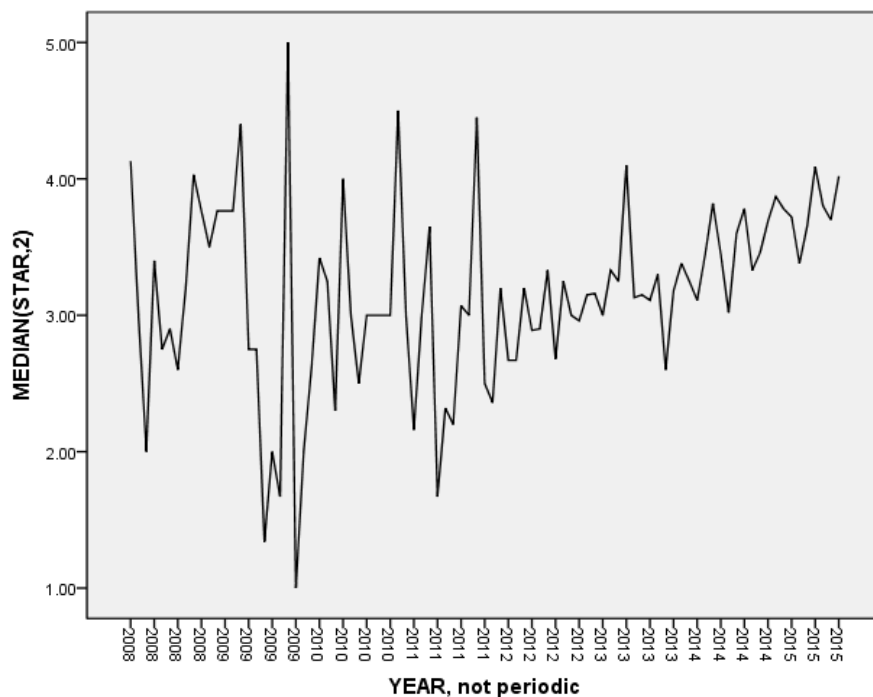


Figure 12: The Time-Series image of microwave

### Stationary Series Judgement

From the images we can find that the star rating of hairdryer and pacifier are stable on the whole. The image of microwave which is in a mess is mainly because of the lack of data. For further exploring the stationarity, the data were treated by difference and seasonal difference respectively to eliminate regular variations and we get the more stable images.

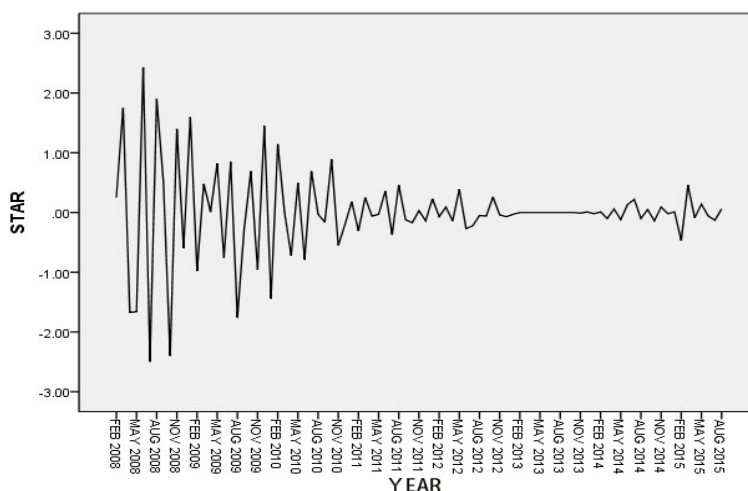


Figure 13: The season difference image of pacifier

The image can indicate the series is stable. Then we moved on calculating the autocorrelation coefficient of the data and holding on Ljung-Box test.

The correlation coefficient measure refers to the degree to which two different events affect each other. The autocorrelation coefficient measures the degree of correlation between two different periods of the same event, and the image is to measure the influence of one's past behavior on one's present. In the analysis of time series [1], for time series

$$\{X_t, x \in T\}$$

take  $T, s \in T$  at will, and define  $T, s$  as the self-covariance function of sequence

$$\{X_t\}: \text{Gamma}(t, s) = E(X_t - \mu_t)(X_s - \mu_s)(t, s)$$

is defined as the autocorrelation coefficient of time series  $\{X_t\}$ , which is abbreviated as

$$\text{ACF: } \text{Rho}(t, s) = \text{gamma}(t, s) / \text{SQRT}(DX_t \times DX_s)$$

Where  $E$  represents the mathematical expectation and  $D$  the variance.

Ljung-Box test is an injection to randomness, or we can say it is a statistical test for the existence of time series lag correlation. The null hypothesis and alternative hypothesis of LBQ respectively are:

①  $H_0$ : The null data is independent with one another, that is to say the correlation coefficient of the population is zero, and some of the correlations that can be observed arise only from the errors of random sampling.

②  $H_1$ : The null data is not independent with one another.

Table 2 Autocorrelation image of pacifier in stars series

Lag	Autocorr -elation	Standard error	Box-Ljung		
			value	df	Sig. <sup>b</sup>
1	.070	.097	.518	1	.472
2	.307	.096	10.735	2	.005
3	.224	.096	16.206	3	.001
4	-.084	.095	16.990	4	.002
5	.138	.095	19.122	5	.002
6	-.178	.094	22.705	6	.001
7	-.024	.094	22.769	7	.002
8	-.152	.093	25.429	8	.001
9	-.005	.093	25.432	9	.003
10	.138	.092	27.670	10	.002
11	.075	.092	28.345	11	.003
12	.172	.091	31.905	12	.001
13	.109	.091	33.338	13	.002
14	.227	.090	39.652	14	.000
15	.101	.090	40.919	15	.000
16	.219	.089	46.902	16	.000

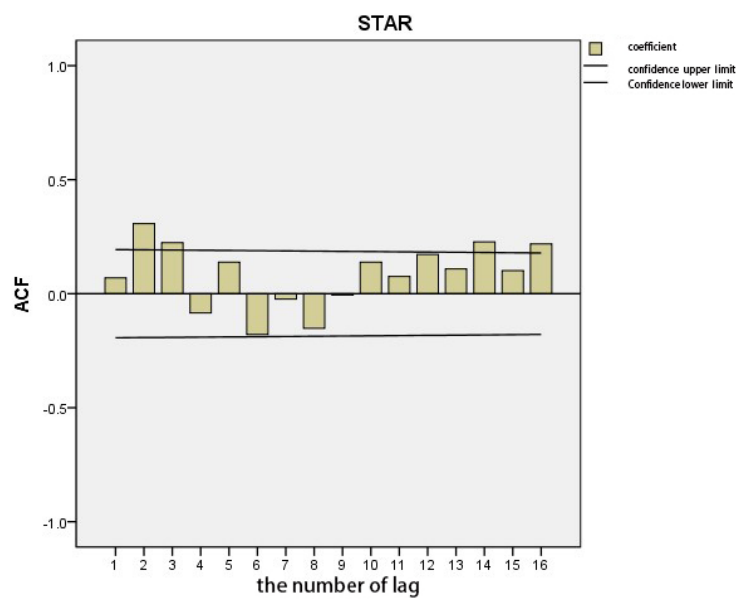


Figure 14

- a. The underlying process of the hypothesis is independence (white noise).
- b. Based on the asymptotic chi-square approximation.

Lag order n: autocorrelation between n data

In this part, judging stationary is mainly through the image of ACF and P value(sig.). When P value is less than 0.5, we can judge that the actual situation is significantly different from the hypothesis.

When  $\text{Sig.} < 0.05$ , it proves the existence of autocorrelation, not all white noise. And then we can calculate the Partial Autocorrelation Coefficient (PACF)

Table 3: **PACF**

log	PACF	Standard error
1	.070	.098
2	.304	.098
3	.208	.098
4	-.215	.098
5	.016	.098
6	-.161	.098
7	-.002	.098
8	-.118	.098
9	.142	.098
10	.220	.098
11	.157	.098
12	-.018	.098
13	-.022	.098
14	.121	.098
15	.014	.098
16	.159	.098

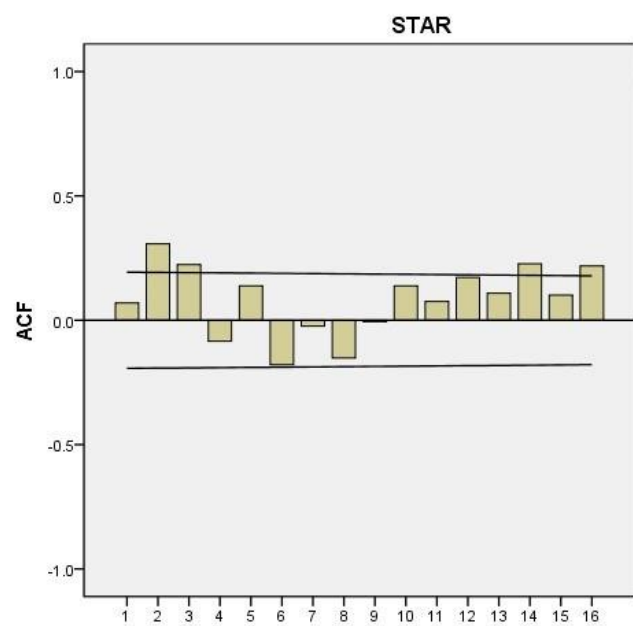


Figure 15

It can be seen from the analysis that the autocorrelation coefficient and partial autocorrelation coefficient are trailing, so the time series of pacifier is a stable.

Therefore, the market reputation of pacifier did not change significantly during the given year.

### Further Analysis of Previous and Subsequent reviews

Specific comments can be divided into three categories: first, comments made in response to comments, second, positive comments, and third, negative comments.

We assumed that in the filtered data, the attitude of the comments was consistent with the score. Negative comments were associated with lower scores, while positive comments were associated with higher scores.

First, let's analyze the comments made in response to the comments. During the data search, we found that there were not many comments of this kind. Of the reviews on 'pacifier', only 218 were related to the word 'review', representing 2.2% of the total

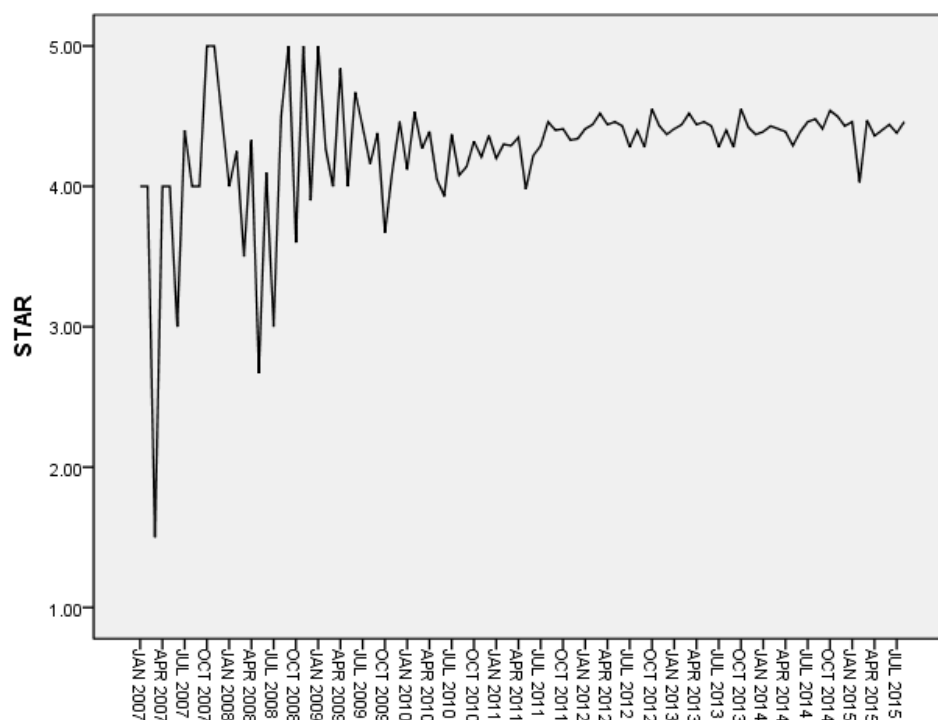


Figure 16

reviews, compared with 68 and 532 in 'hairdryer' and 4.4% and 4.6%, respectively. So, they don't make up much of the population. But that's just to say that a comment with a low score won't cause a lot of comment, but it doesn't rule out that someone might question a high score or agree with a low score. Based on the second question, we obtained the time series of each product's score, as shown in the following figure

Taking 'pacifier' as an example, the sample size in 2008 was too small to be considered, and the score declined in October 2009, but was not affected in November.



We can find that in a small range of time, when the score has a certain degree of decline (that is, when the low score increases), the subsequent score will show a large degree of decline, but will recover in a short time, and relative to the total score, it does not account for a large proportion. Based on this, we can say that low score causes low score to cause a small increase in low score or high score, but it will not cause a large range of score changes. The stationary analysis of the curve shows that the curve is still stable. From this, we come to the conclusion that when a certain proportion of low score appears in the score, it can have a small impact on the subsequent score change, that is, it will produce a small range of poor evaluation, but it has no great impact on the overall evaluation of the product (provided that the product quality and its score can match). In terms of people's online shopping experience, a certain range of low score is more likely to affect whether people buy the goods, and will not significantly improve the rate of bad reviews.

## **4 Model Extension and Simulation Analysis**

### **4.1 Problem1: Data Measures for Reviews**

The Word Embeddings & Glove & Logistic Regression model we trained can help us determine how to rate a sentence review. Our model is a sentence measure based on a large amount of data after analyzing the known review stars and reviews. This can help Sunshine Company to quickly evaluate the three newly listed products through review information. The model parameters are the standard for measuring data.

### **4.2 Problem2: Time-based Measures and Patterns**

This question focuses on the impact of ratings on subsequent reviews over a period of time. It mainly involves time series analysis and text analysis. By combining the second question time series and text correlation analysis, we can know whether the star rating will impact on subsequent review text.

### **4.3 Problem3: Indicate a Potentially Successful or Failing Product**

By summarizing the review type and review information of each piece of data, we trained a model that can score reviews. This model includes the impact of all existing words and phrases on the sentence. The combination of parameters is the best

---

combination given by our model through training.

#### **4.4 Problem4: Previous Ratings and Subsequent Reviews**

This question focuses on the impact of ratings on subsequent comments over a period of time, mainly involving time series analysis and text analysis. By combining the second question time series and text correlation analysis, we can know whether star ratings will impact on subsequent review text.

#### **4.5 Problem5: Specific Word and Ratings**

The parameters of the model we trained represent the impact of a known word or phrase on a comment. We only need to find the parameters corresponding to a specific word or phrase to know its impact on the rating.

### **5 Strengths and Weaknesses**

#### **5.1 Strengths**

The word embedding model together with Glove models can not only learn semantic and grammatical information, but also consider the context of words and the information of global corpus. The dimension of the word vector is very small, which saves the storage and computing resources. It is widely used and can be used in various NLP tasks;

#### **5.2 Weaknesses**

Word and vector are one-to-one relations, which cannot solve the problem of polysemous words. That means the model cannot figure out the true meaning in the reviews if the customers use a word like this (“I really love this hair dryer, it smokes my hair!”), the customer use love in an opposite meaning). In addition, Gloves is also static models, that means it cannot be modified while training. Although it is widely used, it cannot be dynamically optimized for specific tasks.

---

## 5.3 Further Work

- In our model, we ignore the helpful votes and total votes. We may figure out a way to use these two columns.
- During the data labeling, the rating standard we used this time is simply to divide the comments into two levels of 0 and 1. This classification method is too rough and cannot reflect the 1-to-5-star step rating.
- The data given this time is very complicated, and it is not suitable for us to analyze a single product. Our model is only solving the three categories of pacifier hair dryer and microwave oven this time. If a single product can get enough reviews in the future to reach a certain training volume, we can use our model to target a specific Product tracking analysis and comprehensive forecast.

## 6 A Letter

Dear Marketing Director,

It's our honor to help you with analyzing the data from Amazon reviews. We are writing this letter to report our latest findings.

Currently there are just too many comments online, and companies have been trying to dig out the wealth of data. A mass of people and resources are put into the work, whereas the results still seem to be disappointing. Our work, we admit, is not a ground-breaking achievement in any way, but it does offer proper assistance that will enable your company to fetch the helpful information fast and correctly.

Based on the scoring system of Amazon, we build a combined model including **word-embeddings, glove and logistic regression**, all of which are the cutting-edge tools applied in data science. **Our model can be used as a measure of reviews, helping to rate the future reviews of the three products and to quickly analyze the impact of this review on product reputation.** Additionally, the parameters of our model represent the degree to which each word affects the sentence score. **A time series analysis** is carried out as well since we want to find the time-based measures and patterns that may suggest the potentially successful or failing products. **You can acquire the expected sales of a specific type of product simply by inputting its related data.**

In terms of the three forthcoming products, we have come to more specific conclusions. The potentially important design features of each product are as follow.

	Potentially Important Design Features
Microwave	<ol style="list-style-type: none"> <li>1. Buttons and button pads that are more reliable</li> <li>2. Instructions that are more readable and clear</li> <li>3. Enough power that matches the wattage</li> <li>4. Less plastic and more stainless steel</li> <li>5. Less noise</li> </ol>
Pacifier	<ol style="list-style-type: none"> <li>1. Ones with handle may be better</li> <li>2. Stuffed animals that attract babies' attention</li> <li>3. Glowing</li> <li>4. Better designed soother</li> <li>5. Less smell</li> </ol>
Hair dryer	<ol style="list-style-type: none"> <li>1. Plug that can match different countries' voltage</li> <li>2. Better designed switch</li> <li>3. Flexible warmth</li> <li>4. Not fizzy</li> </ol>

As for your online marketing plans, we have the following advice:

- **Use our model to make predictions** about the reputation of this product through the market feedback comment information immediately after the product is launched. You can coordinate with the production department to control the production quantity in time to prevent overproduction.

- **Collect the review data of known successful or failed products to train the existing model** to ensure that the model training volume is large enough to allow the model to continuously adapt to the market environment and continue to grow into a comprehensive and reliable product prediction model.

- **Refresh the parameters to acquire latest keywords** in the reviews of certain products. Since the market is always changing, significance should be attached to the learning ability of our model. Keep track of the trend and make further decisions.

As listed above, our model should provide great advice for your marketing stratege, and we sincerely hope that you can accept the model and the advice.

Sincerely yours,

MCM2020 Team 2013303

## 7 References

---

- [1] Jason Brownlee, A Gentle Introduction to the Bag-of-Words Model
- [2] Page 65, [Neural Network Methods in Natural Language Processing](#), 2017.
- [3] Page 75, [Neural Network Methods in Natural Language Processing](#), 2017.
- 

## 8 Appendix

```
// =====
// Name           : Bag of words.py
// Author          : Yuxin Yang
// Version         : a.0
// Copyright       : Your copyright notice
// Description     : Bag of words in python.
// =====

import numpy as np
import pandas as pd
import sklearn.linear_model
import sklearn.tree
import sklearn.metrics
from matplotlib import pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
import string
from nltk.corpus import wordnet
import string
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.tokenize import WhitespaceTokenizer
from nltk.stem import WordNetLemmatizer
```

---

```
import enchant
import re
from sklearn.linear_model import LogisticRegressionCV
from autocorrect import spell

import seaborn as sns
from sklearn.model_selection import cross_validate

#sns.countplot(x='is_positive_sentiment', data=y_train_df)

#x_train_df['Word Count'] = [len(Text.split()) for Text in x_train_df['text']]
#x_train_df['Uppercase Char Count'] = [sum(char.isupper() for char in Text) for
Text in x_train_df['text']]
#x_train_df['Special Char Count'] = [sum(char in string.punctuation for char in Text)
for Text in x_train_df['text']]
#
# Clean Data
def get_wordnet_pos(pos_tag):
    if pos_tag.startswith('J'):
        return wordnet.ADJ
    elif pos_tag.startswith('V'):
        return wordnet.VERB
    elif pos_tag.startswith('N'):
        return wordnet.NOUN
    elif pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN

def check_english(text):
    d = enchant.Dict("en_US")
    english_words = []
    for word in text:
        if d.check(word):
            english_words.append(word)
    return english_words

def reduce_lengthening(text):
```

---

```

    pattern = re.compile(r"(\s){2,}")
    return pattern.sub(r"\1\1", text)

def clean_text(text):
    # lower text
    text = text.lower()
    # tokenize text and remove punctuation
    text = [word.strip(string.punctuation) for word in text.split(" ")]
    # remove words that contain numbers
    text = [word for word in text if not any(c.isdigit() for c in word)]
    # remove stop words
    stop = stopwords.words('english')
    text = [x for x in text if x not in stop]
    # remove empty tokens
    text = [t for t in text if len(t) > 0]
    # pos tag text
    pos_tags = pos_tag(text)
    # lemmatize text
    text = [WordNetLemmatizer().lemmatize(t[0], get_wordnet_pos(t[1])) for t in
pos_tags]
    # remove words with only one letter
    text = [t for t in text if len(t) > 1]
    # modify mis-spelled
    text = [spell(reduce_lengthening(t)) for t in text]
    # remove non english and mis-spelled
    text = check_english(text)
    # join all
    text = " ".join(text)
    return(text)

def bag_of_words(x_train_df):
    vectorizer = TfidfVectorizer()
    corpus = x_train_df['text']
    X = vectorizer.fit_transform(corpus)
    vectorizer.get_feature_names()
    feature_train = X.toarray()

    return feature_train

```

---

---

```
def bag_of_words_large(x_train_df):
    bigram_vectorizer = TfidfVectorizer(ngram_range=(1,
2),token_pattern=r'\b\w+\b', min_df=1)
    # analyze = bigram_vectorizer.build_analyzer()
    corpus = x_train_df['text']
    feature_train = bigram_vectorizer.fit_transform(corpus).toarray()

    return feature_train

def generate_train_and_test_data(x_train_df,large):

    x_train_df["text"] = x_train_df["text"].apply(lambda x: clean_text(x))
    if large != 'large':
        x_train= bag_of_words(x_train_df)
    else:
        x_train= bag_of_words_large(x_train_df)
    return x_train

#building the neural net
from keras import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
from keras.utils import to_categorical

x_train_df = pd.read_csv('hair_dryer_review.csv')
y_train_df = pd.read_csv('hair_dryer_label.csv')

# 1.2 LOGISTIC REGRESSION
y_tr = y_train_df.to_numpy()
# RUN TWICE
```



---



---

```

x_train = generate_train_and_test_data(x_train_df,'large')

C_grid = np.logspace(-9, 6, 31)
logistic_train_error = []
logistic_test_error = []
for c in C_grid:
    clf_logistic = sklearn.linear_model.LogisticRegression(random_state=0,
solver='liblinear', C=c )
    clf1 = sklearn.model_selection.cross_validate(clf_logistic, x_train, y_tr,
cv=3,return_train_score=True)
    logistic_train_error.append(np.average(clf1['train_score']))
    logistic_test_error.append(np.average(clf1['test_score']))
print(logistic_train_error,logistic_test_error)
best_C = C_grid[np.argmax(np.array(logistic_test_error))]
# best_C = 315227.77
clf_logistic = sklearn.linear_model.LogisticRegression(random_state=0,
solver='liblinear', C=best_C ).fit(x_train, y_tr)

# LOGISTIC PLOT
plt.plot(C_grid,logistic_train_error,label='train')
plt.plot(C_grid,logistic_test_error,label='test')
#plt.xlim([0, 400000])
plt.xlabel("C_grid")
plt.ylabel("Average Score")
plt.legend()
plt.show()

// =====
//  Name           : Word embedding.py
//  Author          : Yuxin Yang
//  Version         : a.0
//  Copyright       : Your copyright notice
//  Description     : Word embedding in python.
//  =====

```

---

```
import numpy as np
import pandas as pd
import sklearn.linear_model
import sklearn.tree
import sklearn.metrics
from matplotlib import pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
import string
from nltk.corpus import wordnet
import string
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.tokenize import WhitespaceTokenizer
from nltk.stem import WordNetLemmatizer
import enchant
import re
from sklearn.linear_model import LogisticRegressionCV
from autocorrect import spell
from keras import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
from keras.utils import to_categorical
```

```
# Clean Data
```

```
def get_wordnet_pos(pos_tag):
    if pos_tag.startswith('J'):
        return wordnet.ADJ
    elif pos_tag.startswith('V'):
        return wordnet.VERB
    elif pos_tag.startswith('N'):
        return wordnet.NOUN
    elif pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
```

---

```
def check_english(text):
    d = enchant.Dict("en_US")
    english_words = []
    for word in text:
        if d.check(word):
            english_words.append(word)
    return english_words

def reduce_lengthening(text):
    pattern = re.compile(r"(\.){2,}")
    return pattern.sub(r"\1", text)

def clean_text(text):
    # lower text
    text = text.lower()
    # tokenize text and remove punctuation
    text = [word.strip(string.punctuation) for word in text.split(" ")]
    # remove words that contain numbers
    text = [word for word in text if not any(c.isdigit() for c in word)]
    # remove stop words
    stop = stopwords.words('english')
    text = [x for x in text if x not in stop]
    # remove empty tokens
    text = [t for t in text if len(t) > 0]
    # pos tag text
    pos_tags = pos_tag(text)
    # lemmatize text
    text = [WordNetLemmatizer().lemmatize(t[0], get_wordnet_pos(t[1])) for t in
pos_tags]
    # remove words with only one letter
    text = [t for t in text if len(t) > 1]
    # modify mis-spelled
    text = [spell(reduce_lengthening(t)) for t in text]
    # remove non english and mis-spelled
    text = check_english(text)
    # join all
    text = " ".join(text)
    return(text)
```

---

## #Part II

### #2.1

```
import sklearn.neighbors
from collections import OrderedDict

word_embeddings = pd.read_csv('glove.6B.50d.txt.zip',
                              header=None, sep=' ', index_col=0,
                              nrows=100000, compression='zip',
                              encoding='utf-8', quoting=3)
# Build a dict that will map from string word to 50-dim vector
word_list = word_embeddings.index.values.tolist()
word2vec = OrderedDict(zip(word_list, word_embeddings.values))

x_train_df = pd.read_csv('hair_dryer_review.csv')
y_tr = pd.read_csv('hair_dryer_label.csv')

x_train_df["text"] = x_train_df["text"].apply(lambda x: clean_text(x))

def create_feature(x_train_df):
    words_vector = []
    for text in x_train_df['text']:
        vector = np.zeros(50)
        for word in text.split( ):
            try:
                # v = word2vec[word]/np.argmax(word2vec[word])
                v = word2vec[word]
                vector += v
            except:
                pass
        words_vector.append(list(vector))
    return(words_vector)

x_train = create_feature(x_train_df)
```

---



---

```

def check_length_train(x_train,y_tr):
    index = 0
    for i in x_train:
        if len(i) != 50:
            x_train.pop(index)

            y_tr.pop(index)
        index+=1
    return x_train,y_tr

def check_length_test(x_test):
    index = 0
    for i in x_test:
        if len(i) != 50:
            x_test.pop(index)

            index+=1
    return x_test

#logistic

C_grid = np.logspace(-9, 6, 31)
logistic_train_error = []
logistic_test_error = []
x_train, y_tr = check_length_train(x_train,y_tr)
x_t, x_v, y_t, y_v = sklearn.model_selection.train_test_split(x_train, y_tr,
test_size=0.2, random_state=1)

for c in C_grid:
    clf_logistic = sklearn.linear_model.LogisticRegression(random_state=0,
solver='liblinear', C=c )
    clf1 = sklearn.model_selection.cross_validate(clf_logistic, x_train, y_tr,
cv=3,return_train_score=True)
    logistic_train_error.append(np.average(clf1['train_score']))
    logistic_test_error.append(np.average(clf1['test_score']))
print(logistic_train_error,logistic_test_error)
best_C = C_grid[np.argmax(np.array(logistic_test_error))]

```

---

```
clf_logistic = sklearn.linear_model.LogisticRegression(random_state=0,  
solver='liblinear', C =best_C ).fit(x_train, y_tr)
```

```
# LOGISTIC PLOT  
plt.plot(C_grid,logistic_train_error,label='train')  
plt.plot(C_grid,logistic_test_error,label='test')  
#plt.ylim([0.75, 0.80])  
plt.xlim([0, 0.02])  
plt.xlabel("C_grid")  
plt.ylabel("Average Score")  
plt.legend()  
plt.show()
```