**Report on Project 2**

**Analysis on Amazon Ratings and Review Data**



**CS/MSA6500 Big Data Analytics**

**Supervised by:** Prof. Robert Dyer

**Submitted by:**

Aditya Jillellamudi

**Content**

**Project Overview**

**Task0**

Details on import

Details on Analysis

**Task1**

Details on import

Details on Analysis

**Task2**

Details on import

Details on Analysis

**Summary**

**Efforts Expended**

**Source Code**

# Project Overview:

We are analyzing Amazon Ratings and Reviews data in this project. We are dealing with reading and processing data available in csv and json files. The different analysis has been done various Hadoop and big data technologies and different analysis tolls and libraries like Spark, Sparks, Pandas, Numpy, Seaborn, matplotlib, sclearn, nlkt, Word cloud, etc.

## Tasks Performed:

1. Filter/Clean Dataset:
   * Remove any duplicate ratings where a user had rated an item more than once with only the newest rating kept.
   * Converted Unix Epoch format date time format to a cognizable format.
   * Add columns necessary for later tasks should be added to the dataset.
2. Exploratory Data Analysis of the Dataset:
   * Compute graphs and design tables for the following:
     1. users vs number of ratings per user: do most users have a few or a lot of ratings?
     2. items vs number of ratings per item: do most items have a few or a lot of ratings?
     3. ratings distribution: how many times does each rating appear?
     4. time vs ratings: do certain months have more ratings?
     5. distribution of average ratings for each user - does it change if you only consider users with 5 or more ratings? 10 or more ratings?
     6. distribution of average ratings for each item - does it change if you only consider items with 5 or more ratings? 10 or more ratings?
3. Open Ended Analysis
   * Exploratory Data Analysis
   * Time Series Analysis
   * Clustering Based Analysis

# Task 0

**Details of data import and Problems faced:**

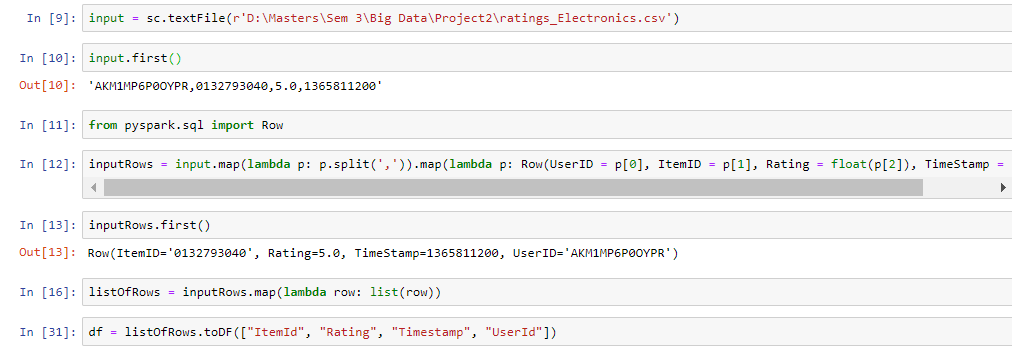
The data was imported from the provided link where multiple datasets of Amazon ratings and reviews: <http://jmcauley.ucsd.edu/data/amazon/>. The electronics ratings csv file has been used for Task 0 and the file has 7824482 rows with file size as 304 Mb.

We started writing pyspark code inside Docker on the pyspark shell. We also wanted to use python 3 so we changed the default python 2 to be set to python 3.

We faced issues on docker, while running the commands, we were often getting Java Runtime errors, which suggested there are not enough resources our system was providing, so we installed pyspark and setup the environment on our local system and ran pyspark code through jupyter notebook from then onwards. We downloaded the required file explicitly on the local machine which took 4 mins to download.

We were provided Spark+Jupyter+Docker image, which again led to some problems of uploading data again and again, which again was not development time. So, we continued the setup on local machine for running our code and various analysis.

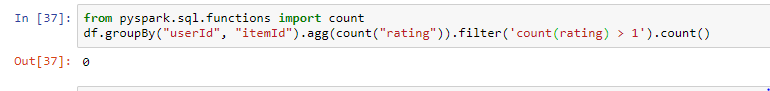
We read the data and converted it to a pyspark dataframe. ‘df’ is the pyspark dataframe RDD using the below commands.



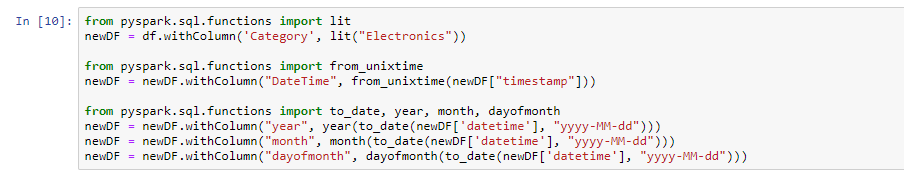
**Details on Filtering in Task 0:**

The first task is Task 0, where we clean and filter the data. It involves following 3 tasks:

1. Removing of duplicate records that states that the filtered data will have only one rating for every product by any user. We ran the following query on ‘df’ using **groupby** which resulted in count 0. We identified there were no duplicates.

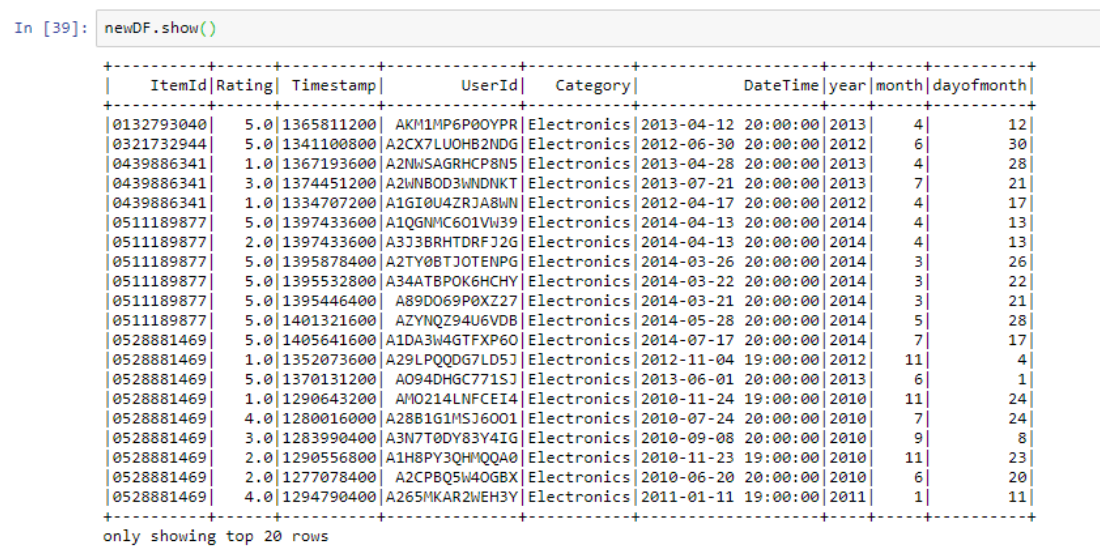


1. Conversion of the epoch time format to human readable time format. We added a new column called ‘Datetime’ to represent the human readable data time using the **from\_unixtime**.
2. Add new columns for later analysis tasks using **lit**: We added 4 more columns: Category, Year, Month, DayOfMonth.



**Output of the Task 0:**

After Task0 completion the ‘newDF’ stores the filtered dataset which looks something like the below :



# Task 1

**Details of data and import:**

We took the final output of Task 0, and there are 4201770 records. For each question from a to f, we have different classification criteria. According to these criteria, using filter function to divide records, and count the total number of each classification in pyspark. Finally, we visualize the outcomes by histogram and line chart.

We faced issue on how to out the data from docker to local machine, because we want to read the csv file on the local machine. Then we moved ahead with performing code changes and analysis on local machine. We further changed the plan of importing the data to csv as well. We decided to use the pyspark dataframe, convert it to pandas dataframe and then use the pandas dataframe to perform exploratory data analysis.

And for the analysis, it is a little simplified when we analyze outcomes firstly. We just focusing on the conclusions that can be drawn directly from the histogram or line chart, especially the questions e and f. Later, from the discuss and learn from online, we add some critical thinking based on the background of data, and it looks more meaningful.

**Detail each problem on Task 1:**

There are a huge number of users, so we planned to categorize the data as the number of users who gave ratings like the below:

We categorized the ratings as follows for below sub tasks:

* Number of ratings equal to 1
* Number of ratings between 2 to 5
* Number of ratings between 6 to 10
* Number of ratings between 11 to 50
* Number of ratings greater than 50.

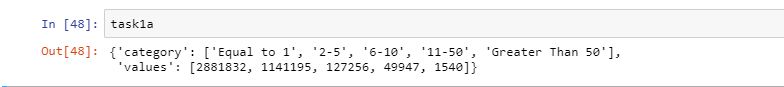
**Output of the Task 1:**

1. **Users vs number of ratings per user: do most users have a few or a lot of ratings?**

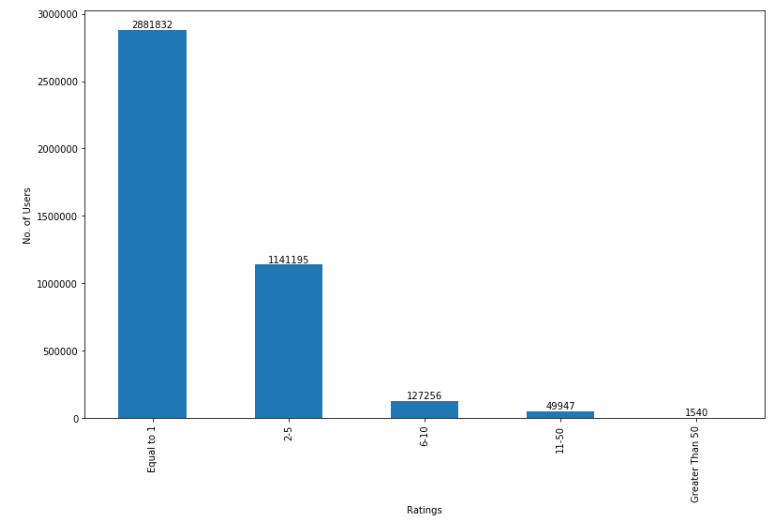
For this part we grouped the data by user id to identify the number of ratings provided by the user. The following shows the code snippet and the top 20 resulting rows with the number of ratings for every user.



To be able to create graphs, we created categories dictionary to be used by matplot lib code to generate the graph using the below code:



**Graph and analysis for (a)**



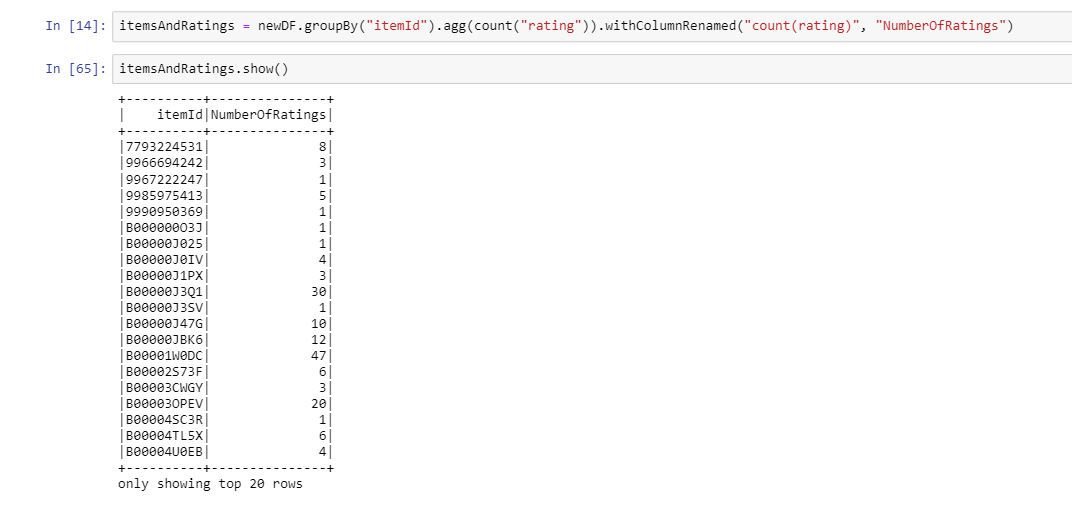
From the histogram, we can analyze two things.

* + There are few numbers of users who shop regularly and review regularly.
  + There are few numbers of users who put efforts to review regularly.

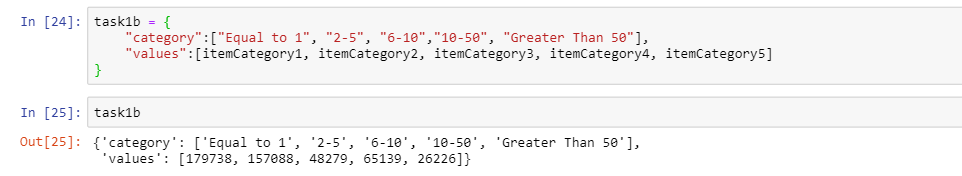
From the histogram we can see that most percentage of the people have reviewed only once or less that 5 times.

1. **Items vs number of ratings per item: do most items have a few or a lot of ratings?**

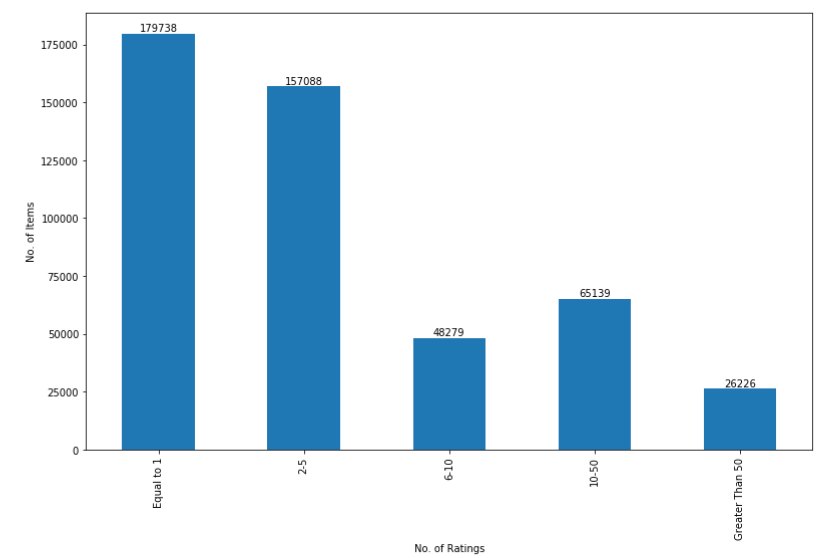
For this part we grouped the data by item id to identify the number of ratings provided to every product. The following shows the code snippet and the top 20 resulting rows with the number of ratings for every product.



To be able to create graphs, we created categories dictionary to be used by matplot lib code to generate the graph using the below code:



**Graph and analysis for (b):**



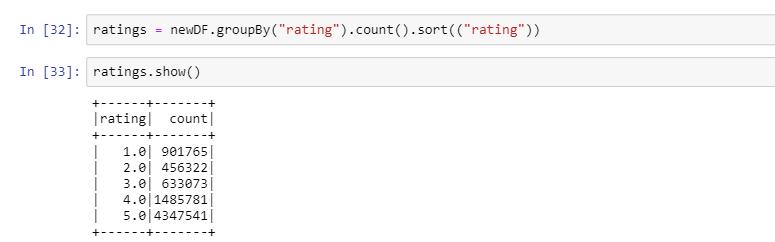
From the histogram, we can observe that

* + There the few items which got number of ratings above 5.

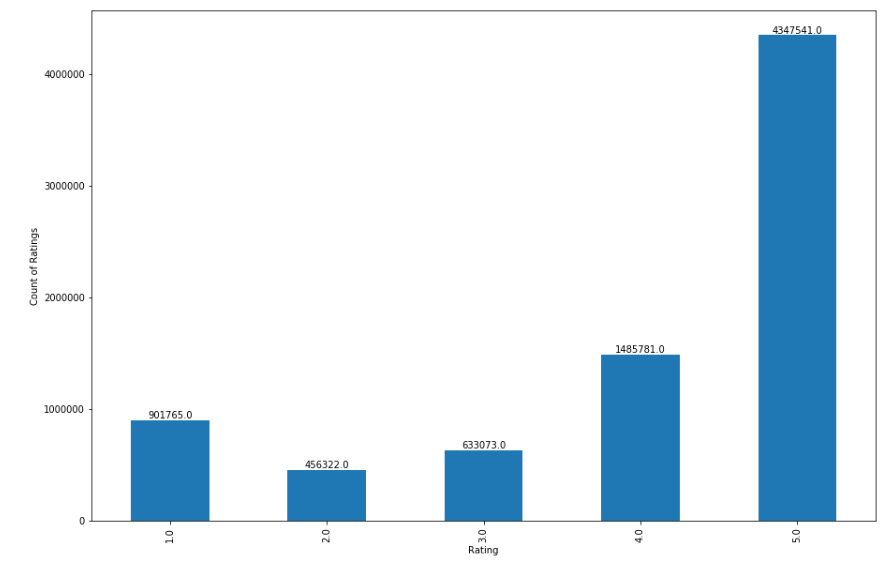
From the histogram we can see that most percentage of the items have reviewed only once or less that 5 times.

1. **Ratings distribution: how many times does each rating appear?**

For this part we grouped the data by ratings to identify the number of ratings provided to every rating. The following shows the code snippet and the number of ratings for every rating.



**Graph and analysis for (c):**

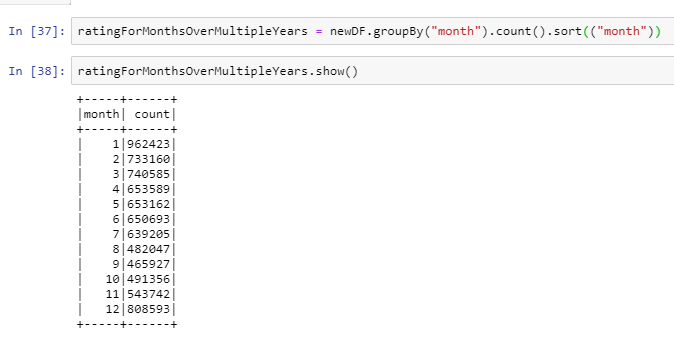


From the plot, we can calculate that 56% of reviews are 5-star rating, and 19% of reviews are 4-star rating, and 8% of reviews are 3-star rating, and 6% of reviews are 2-star rating, and 11% of reviews are 1-star rating.

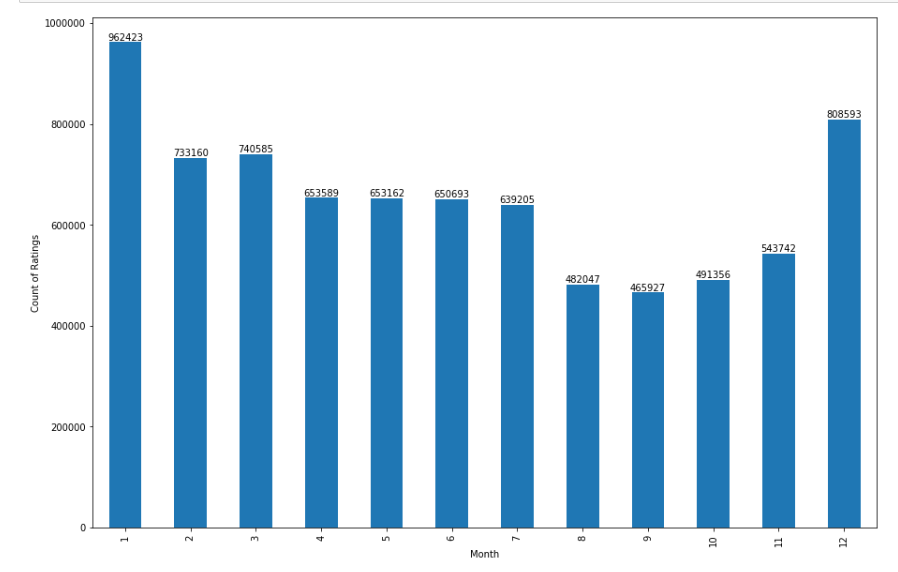
Therefore, we can know that more than half of the reviews give a 5-star rating. Aside from perfect reviews, most reviewers give 4-star or 1-star ratings, with very few giving 2-stars or 3-stars relatively.

1. **Time vs ratings: do certain months have more ratings?**

For this part we grouped the data by month to identify the number of ratings provided in each month. The following shows the code snippet and the number of ratings for every month in multiple years.



**Graph and analysis for (d):**

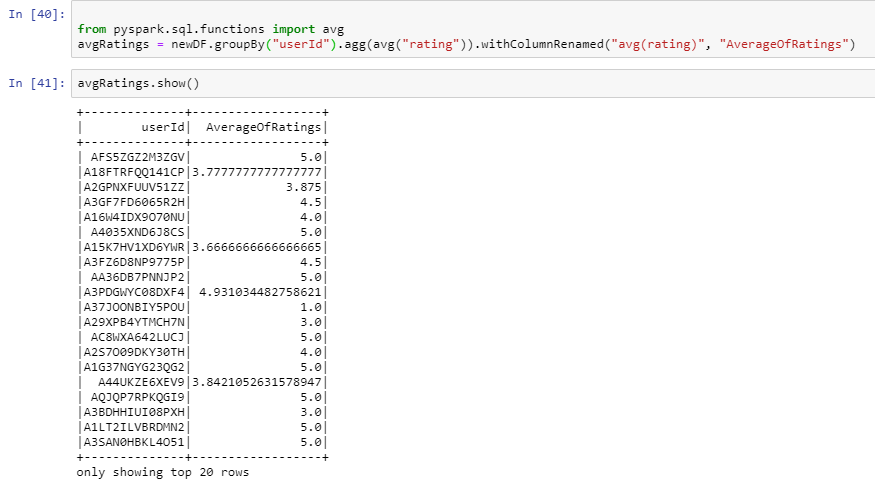


From the histogram, we can see that it can be roughly divided into three period. One is the ratings in December and January, this period has the most ratings. The second period is from February to July, and there has been a bit less ratings. The third period is from August to November, and this period has the fewest ratings.

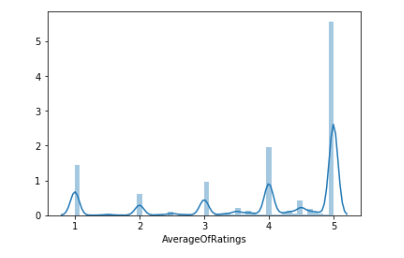
We can assume that December and January are the months of Christmas and new year on which people tend to buy more stuff which results having more ratings. However, from August to November are in the winter, where the activity is less and have low purchases and low ratings.

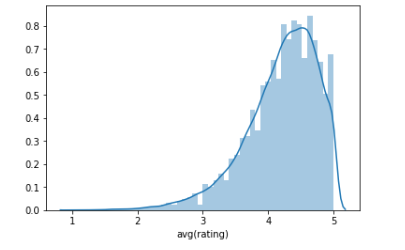
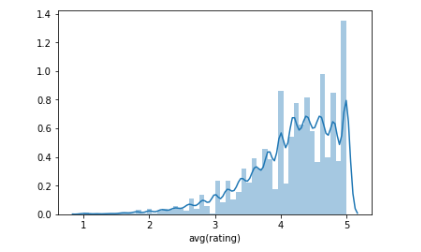
1. **Distribution of average ratings for each user - does it change if you only consider users with 5 or more ratings? 10 or more ratings?**

For this part we grouped the data by user to identify the avg ratings provided in for every user and created the distribution of the 3 cases below.



**Graph and Analysis for (e):**





From the first plot considering distribution of average ratings for each user, we can see the most average rating for each user is around 5-star. Aside from perfect reviews, most reviewers give 4-star or 1-star average ratings, with very few users giving average ratings in 2-stars or 3-stars relatively. And the average rating of 5-star is about three times as frequent as 4-star reviews.

From the second plot consider users with more than 5 ratings, we can see the most average rating for users with 5 more ratings is around between 4-star and 5-star. But we observed that the graph is little skewed towards the high ratings (4&5) with higher mean when compared to the first plot.

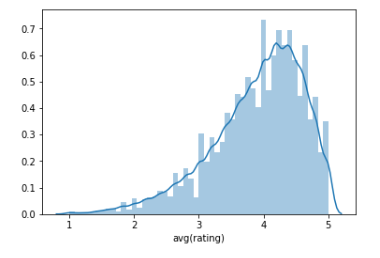
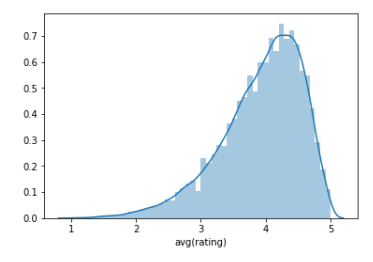
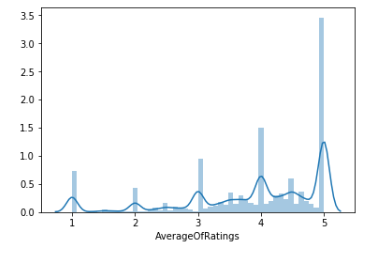
From the third plot consider users with more than 10 ratings, we can see the most average rating for users with 10 more ratings is around between 4-star and 5-star. Again, we observed that the graph is more skewed towards the high ratings (4&5) increasing mean when compared to the second plot.

Comparing with these three distributions, we observed that mean is constantly increasing, and the skewness of the graph also moved towards higher rating. From this we can assume that frequent users tend to buy good products and are satisfied with their products than non-frequent users. This might be because, frequent users are familiar with the website and read the previous reviews and buy good products.

1. **Distribution of average ratings for each item - does it change if you only consider items with 5 or more ratings? 10 or more ratings?**

For this part we grouped the data by item to identify the avg ratings provided in for every product and created the distribution for 3 cases.

**Graph and Analysis for (f):**



Comparing the three Distributions, we again observed the same pattern followed in part (e). Mean is constantly increasing, and the skewness of the graph also moved towards higher rating. From this we can assume that frequent reviewed items tend to have good ratings. This might be because, frequent reviewed items have good amount of information (in reviews) which allows users to have good idea on the products to buy.

# Task 2

**Analysis 1: Exploratory Data Analysis**

The data is imported from <http://jmcauley.ucsd.edu/data/amazon/> link provided in the project assignment. **5-core Digital Music** has been selected to perform Exploratory data analysis as the Ratings-only data is limited and doesn’t have enough data to make inferences. It took around 2 minutes to download the 5-core data.

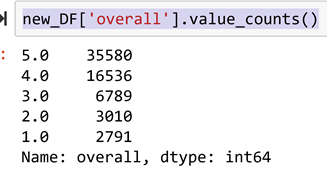
Tried to analyses the distribution on rating similar items and patterns of ratings on hourly and daily basis and different days and months. But all the distributions for all the cocategories were similar and was unable to get the inferences from them. So, had to add new additional data(5-core) to change the plan a little. File has 64,706 Rows with file size as 84.2 Mb.

Faced issues on docker, while running the commands, we were often getting Java Runtime errors, which suggested there are not enough resources our system was providing, so we installed pyspark and setup the environment on our local system and ran pyspark code through jupyter notebook from then onwards. We downloaded the required file explicitly on the local machine which took 4 mins to download.

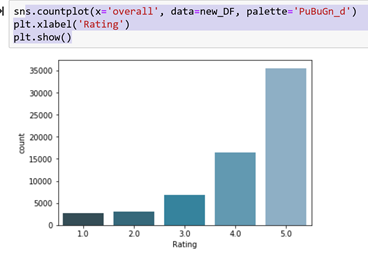
The whole part 1 in the task 2 consists of two analysis “sentiment analysis” and “helpfulness analysis”. Sentiment analysis gives the positive ratings & words and negative words & ratings by comparing for analysis and plotting the word clouds. “helpfulness analysis” gives the which part of the data is helpful and which part we can eliminate and include the analysis with the word size. This analysis took around 5 hours each to figure out the syntax and rectify the errors and refining the analysis.

**Results and analysis:**

This is the distribution of the ratings of the Digital Music data.



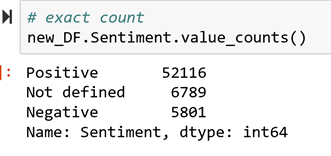
Original dataset contains total 64,706 reviews in which 5-star reviews are 54.98% of all reviews, followed by 4-star reviews (25.55%), followed by 3-star reviews (10.49%), 2-star reviews (4.65%), and finally 1-star reviews (4.31%).



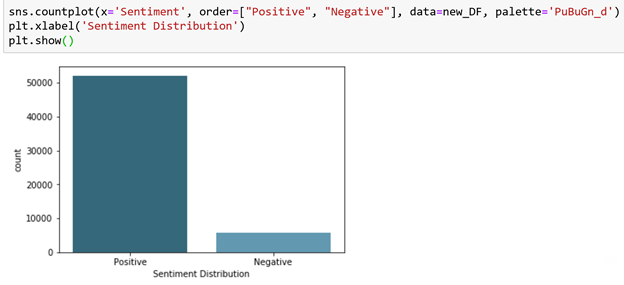
This is also represented in the histogram graph where we can see the clear skewness in the review data. The skewness is towards the high reviews (4,5).

**Sentiment Analysis:**

To make better analysis sentiment column is added to the data by dividing into three parts “positive”, “negative” and "not defined” based on reviews. If a review from a user is higher than 3 it is classified as the “positive” and less than 3 is classified in to “negative”. The reviews which are equal to 3 are neglected ("not defined") as they are neutral and cannot be helping to analyze in this particular case.

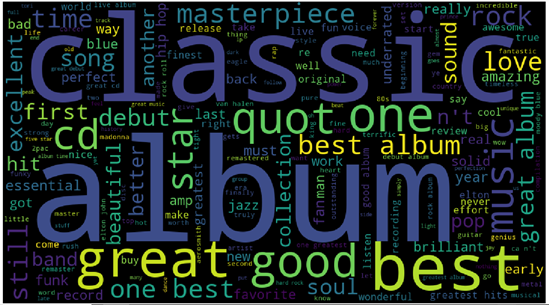


It has been observed that 80.54% (52,116) of data is belonging to positive class (review =4,5), 10.49% (6,789) of data is belonging to negative class (review =1,2). The data is Imbalanced, which concludes most of the customers are satisfied from the ELETRONICS products by amazon.

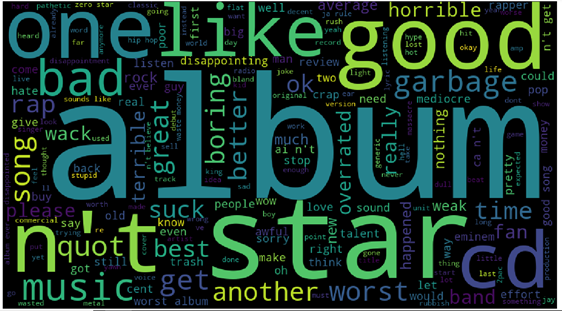


To find the words which are mostly used by the reviewers in the reviews a word cloud plot is used which is an image composed of words used in a particular text or subject, in which the size of each word indicates its frequency or importance.

The below plot is a word cloud for the positive reviews, it is created by using a data-frame pos which only contain the columns of the positive reviews and passing the text review through the wordcloud() function.



In the similar way the word cloud for the negative reviews is created by using a data-frame neg which only contain the columns of the negative reviews and passing the text review through the wordcloud() function.

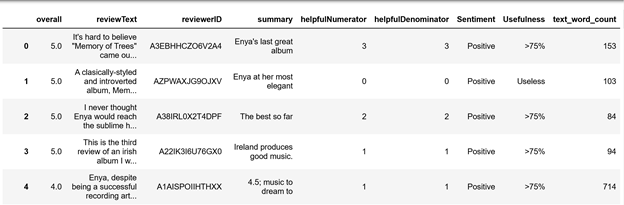


The size of the word is represented by the number of times the words repeated. (Frequency).

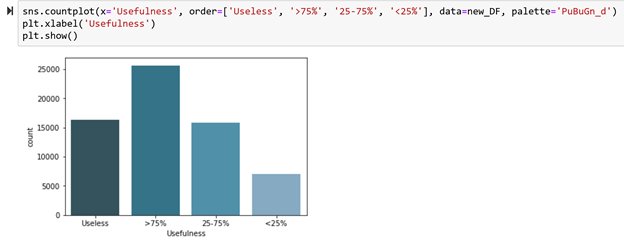
**Helpfulness Analysis:**

The "helpful" column from the original dataset is divided into two new columns separating the array into “helpfullnum”( number of users who found the review helpful) and “helpfulldenom”( number of users who indicated whether they found the review helpful) to refine the data to reduce the size by eliminating the unwanted rows .

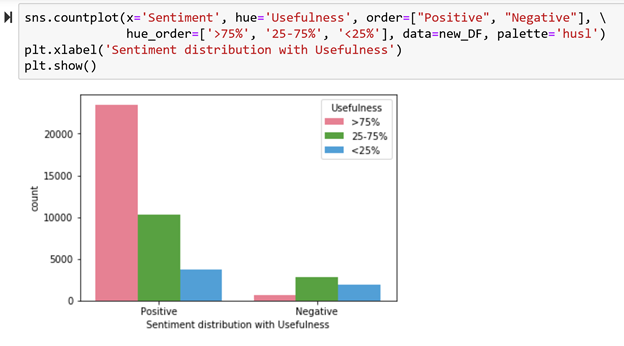
Based on this “helpfullnum” and “helpfulldenom” a new coloumn (“Usefulness”) is added to the dataset to determine whether the text review given by the user is important for the analysis or not. If *["helpfullnum"]/ ["helpfulldenom"] > 0.75* it is classified into useful (>75%) and *["helpfullnum"]/ ["helpfulldenom"] < 0.25* it is classified as “<25%” and 0.25 < *["helpfullnum"]/ ["helpfulldenom"] < 0.75* as “25-75%”, and rest of them as Useless(*["helpfulldenom"]=0).*



The final data frame looks as above



Plotting the histogram for the Usefulness column we observed there are significant amount of Useless reviews, we can eliminate the useless reviews for the analysis which reduces the amount of data and increase the speed of execution and increases the accuracy.



Plotting the graph for “Sentiment distribution with Usefulness” we can see from the above graph positive reviews are found more helpful than the negative reviews.

**Analysis on word count:**

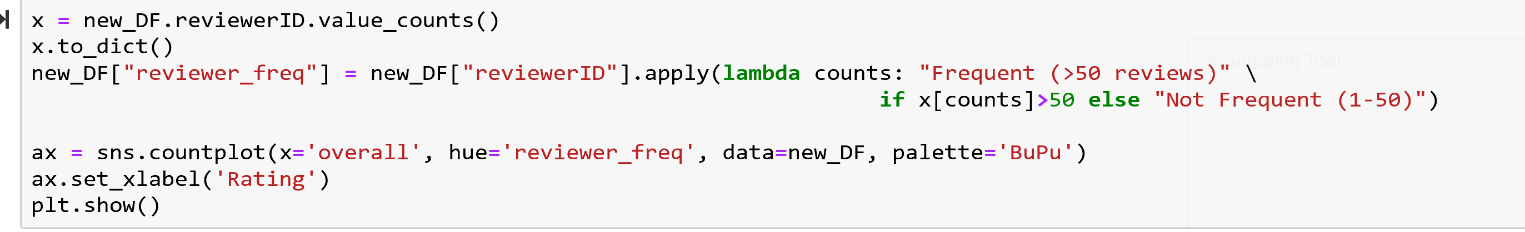


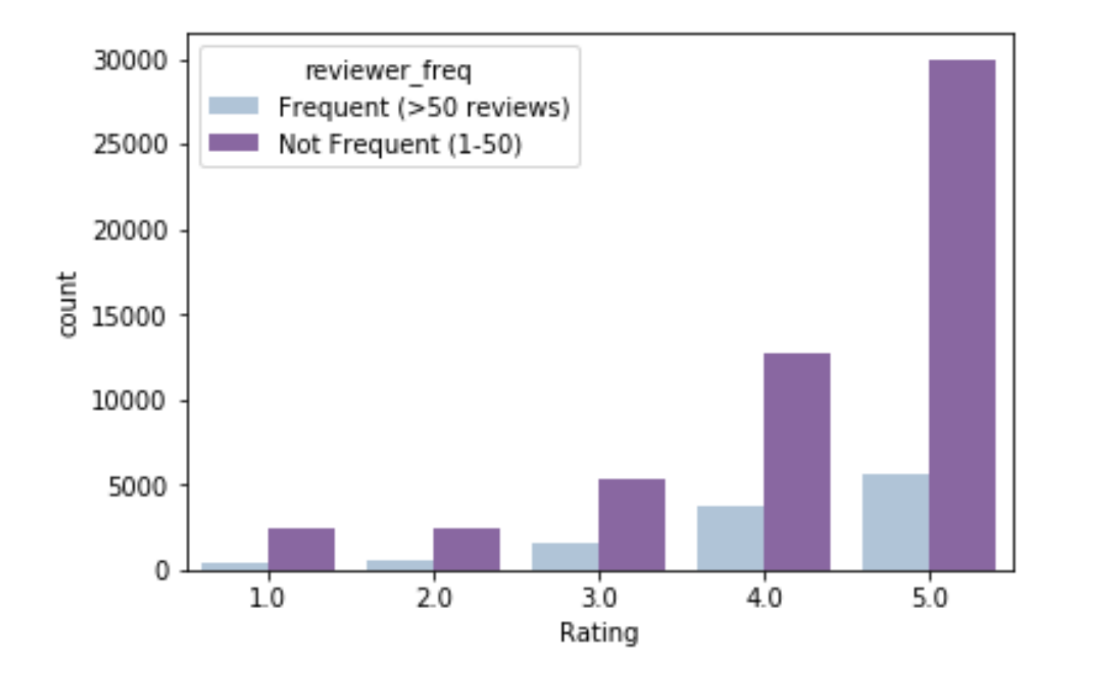
Plotting a box plot for the count of words in the text review(“reviewText”), we can observe that the positive reviews seems to be longer than the negative reviews.

We can conclude this by assuming that the positive reviewers are satisfied with the products and have given explanations why they liked the products, results writing longer and good reviews.

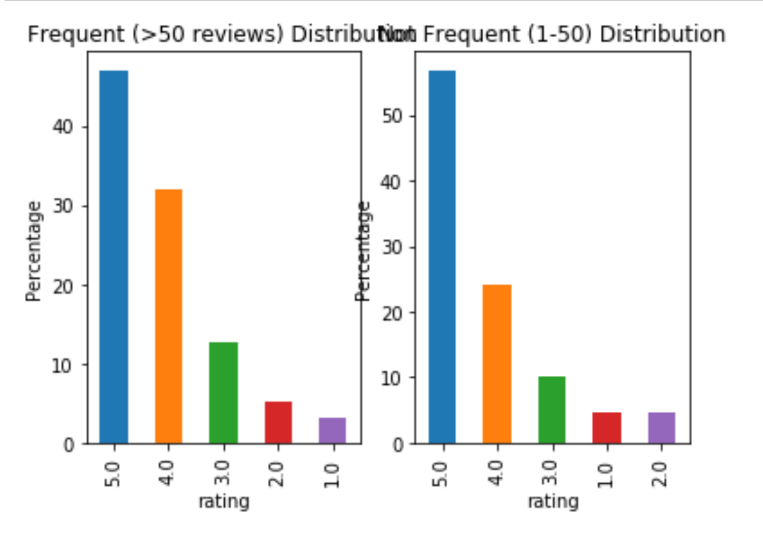
**Analysis on Frequent and non-frequent users:**

Now dividing the data into Frequent users if reviewerID appeared more than 50 times and rest of them as non-frequent users. Plotting the graph for frequency of Frequent and non-frequent users with ratings.





From the above graph we can see that there is significant amount of non-frequent users than the frequent users, and from this graph it is unclear the distribution of ratings for frequent and non-frequent with respective ratings. We will try to divide the plots for frequent users and non-frequent users to analyze.



From the above graph we can see the distribution of the frequent users is more skewed towards the higher rating than the non-frequent users, the Positive reviews (4,3) in the frequent users are more compared to non-frequent users.

From this we can conclude that frequent users tend to buy good products and are satisfied with their products than non-frequent users. This might be because, frequent users are familiar with the website and read the previous reviews and buy good products.

**Analysis 2: Time Series Analysis**

For this task same we used the same file that was used for task 0 and task 1. The electronics ratings csv file has been used for Task 2 as well and the file has 7824482 rows with file size as 304 Mb. To execute the code, it took around 1 min and to complete the whole time series analysis it took 12 hours.

For analysis in task 2 we decided to take aggregate monthly ratings on electronics category from year 1998 to 2014 and tried to predict another 12 months ratings that will occur in electronics category.

Firstly we cleaned the data using different functions in pyspark(code included below) to get monthly data of aggregate rating for each year.

###getting the raw data file

sc = SparkContext.getOrCreate()

input = sc.textFile(r'C:\Users\sameepbabu\Downloads\ratings\_Electronics.csv')

input.first()

>>> 'AKM1MP6P0OYPR,0132793040,5.0,1365811200'

###splitting the string raw data file based on “,”

from pyspark.sql import Row

inputRows = input.map(lambda p: p.split(',')).map(lambda p: Row(UserID = p[0], ItemID = p[1], Rating = float(p[2]), TimeStamp = int(p[3])))

inputRows.first()

>>> Row(ItemID='0132793040', Rating=5.0, TimeStamp=1365811200, UserID='AKM1MP6P0OYPR')

###creating the new column called “Electornics” and converting the TimeStamp into DateTime format

listOfRows = inputRows.map(lambda row: list(row))

df = listOfRows.toDF(["ItemId", "Rating", "Timestamp", "UserId"])

from pyspark.sql.functions import lit

newDF = df.withColumn('Category', lit("Electronics"))

from pyspark.sql.functions import from\_unixtime

newDF = newDF.withColumn("DateTime", from\_unixtime(newDF["timestamp"]))

###Splitting the DateTime format into year, month and day

from pyspark.sql.functions import to\_date, year, month, dayofmonth

newDF = newDF.withColumn("year", year(to\_date(newDF['datetime'], "yyyy-MM-dd")))

newDF = newDF.withColumn("month", month(to\_date(newDF['datetime'], "yyyy-MM-dd")))

newDF = newDF.withColumn("dayofmonth", dayofmonth(to\_date(newDF['datetime'], "yyyy-MM-dd")))

newDF.first()

>>> Row(ItemId='0132793040', Rating=5.0, Timestamp=1365811200, UserId='AKM1MP6P0OYPR', Category='Electronics', DateTime='2013-04-12 20:00:00', year=2013, month=4, dayofmonth=12)

###code to get the number of ratings on monthly basis for all years.

from pyspark.sql.functions import col, concat, lit

newDF= newDF.withColumn("year\_month" , concat(col('year'),lit('\_'),col('month')))

year\_month = newDF.groupBy("year\_month").agg(count("rating")).sort((col("year\_month").asc())).withColumnRenamed("count(rating)", "NumberOfRatings")

year\_month.show()

>>>

+----------+---------------+

|year\_month|NumberOfRatings|

+----------+---------------+

| 1998\_12| 4|

| 1999\_10| 172|

| 1999\_11| 345|

| 1999\_12| 537|

| 1999\_5| 2|

| 1999\_6| 20|

| 1999\_7| 68|

| 1999\_8| 43|

| 1999\_9| 40|

| 2000\_1| 491|

| 2000\_10| 937|

| 2000\_11| 935|

| 2000\_12| 1345|

| 2000\_2| 362|

| 2000\_3| 401|

| 2000\_4| 476|

| 2000\_5| 723|

| 2000\_6| 1212|

| 2000\_7| 896|

| 2000\_8| 840|

+----------+---------------+

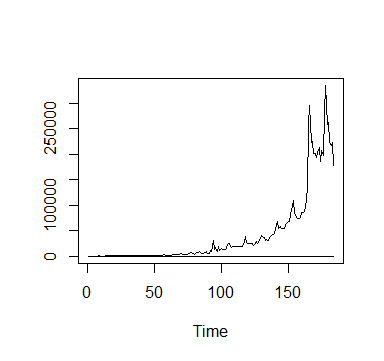
only showing top 20 rows

###to get the output in csv file

year\_month.coalesce(1).write.csv("sample.csv")

The head and tail of the data is given below.

|  |  |  |  |
| --- | --- | --- | --- |
| year | rating | Year | rating |
| 1998\_12 | 4 | 2013\_10 | 207761 |
| 1999\_05 | 2 | 2013\_11 | 198170 |
| 1999\_06 | 20 | 2013\_12 | 282383 |
| 1999\_07 | 68 | 2014\_1 | 334605 |
| 1999\_08 | 43 | 2014\_2 | 257354 |
| 1999\_09 | 40 | 2014\_3 | 262506 |
| 1999\_10 | 172 | 2014\_4 | 225403 |
| 1999\_11 | 345 | 2014\_5 | 216327 |
| 1999\_12 | 537 | 2014\_6 | 223599 |
| 2000\_01 | 491 | 2014\_7 | 177562 |

After getting the data we made a time series plot from year 1998 to 2014 to see the pattern of the data. The time series plot is given below:

From the plot above we can identify that the time series is in increasing trend. This means that the long term mean is not stationary. In addition to that the variance is also not not constant. To confirm this lets use Augmented Dickey-Fuller Test to chech the whether the data is stationary or not.

Augmented Dickey-Fuller Test

data: originalRating

Dickey-Fuller = -2.0202, Lag order = 5, p-value = 0.5677

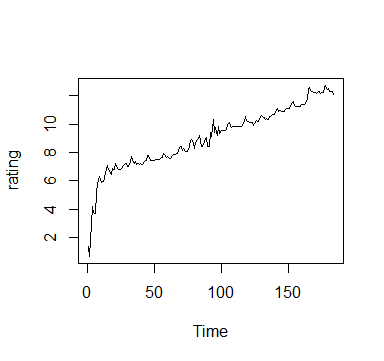
alternative hypothesis: stationary

From the test above we can observe that the p-value is 0.5677 which is above 0.05 level of significance. So we fail to reject the null hypothesis of data and can conclude that data in not stationary.

In order to make the data stationary we need to make some transformation in data. From our original plot we can see that the seasonal pattern of data is increasing in multiplicative rate. This means that the seasonal component at the beginning of the series is smaller than the seasonal component later in the series. To account for this, you’d need to log-transform the data. Head and tail of the log transformed data is given in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| year | rating | Year | rating |
| 1998\_12 | 1.3862944 | 2013\_10 | 12.2441437 |
| 1999\_05 | 0.6931472 | 2013\_11 | 12.1968805 |
| 1999\_06 | 2.9957323 | 2013\_12 | 12.5510196 |
| 1999\_07 | 4.2195077 | 2014\_1 | 12.7207060 |
| 1999\_08 | 3.7612001 | 2014\_2 | 12.4582078 |
| 1999\_09 | 3.6888795 | 2014\_3 | 12.4780292 |
| 1999\_10 | 5.1474945 | 2014\_4 | 12.3256452 |
| 1999\_11 | 5.8435444 | 2014\_5 | 12.2845464 |
| 1999\_12 | 6.2859981 | 2014\_6 | 12.3176095 |
| 2000\_01 | 6.1964441 | 2014\_7 | 12.087075 |

Now after log transformation we made a timeseries plot of the data which is shown below.



From the plot above we can see that variance looks constant and just by looking at the plot it is difficult of say whether mean is constant or not in the long run. So lets again test for stationary for the transformed data using Augmented Dickey-Fuller Test.

Augmented Dickey-Fuller Test

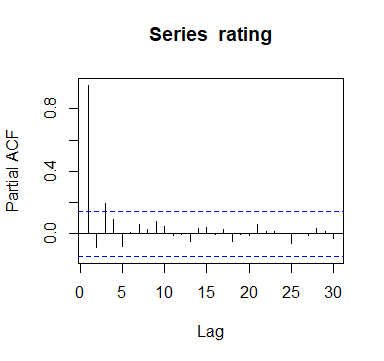
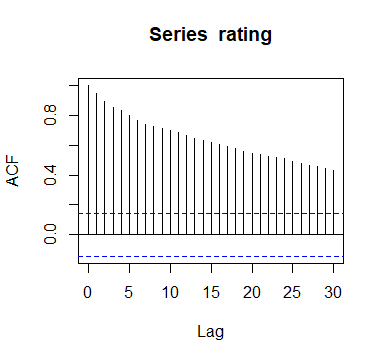
data: rating

Dickey-Fuller = -8.4476, Lag order = 5, p-value = 0.01

alternative hypothesis: stationary

From the test above we can observe that the p-value is 0.o1 which is below 0.05 level of significance. So we reject the null hypothesis of data and can conclude that the timeseries data is stationary.

Once you have a stationary time series data, our next step is to select the appropriate ARIMA model. This means finding the most appropriate values for p and q in the ARIMA(p,d,q) model. To do so, you need to examine the “correlogram” and “partial correlogram” of the stationary time series. A correlogram shows the autocorrelation function. It’s just like a correlation, except that, rather than correlating two completely different variables, it’s correlating a variable at time t and that same variable at time t-k. A partial correlogram is basically the same thing, except that it removes the effect of shorter autocorrelation lags when calculating the correlation at longer lags. The correlogram and partial correlogram of the timeseries data is given below.



From the ACF plot we can see that there is decreasing autocorrelation function as we move with time. As the ACF has a decreasing trend and there is no pattern formed in the ACF we can say that there is no seasonal trend in the time series. Looking at the PACF plot we can see that there is significant PACF until lag 3 and after that there is white noise. So we can assume that the time series can follow AR(3) model. However lets try auto ARIMA function get the better picture of the model that can be used for the time series data. The following table shows the output for the auto.ARIMA function.

Series: rating

ARIMA(3,1,2) with drift

Coefficients:

ar1 ar2 ar3 ma1 ma2 drift

-0.3855 -0.5586 -0.3457 0.3531 0.3973 0.0561

s.e. 0.2200 0.2903 0.0915 0.2248 0.3046 0.0183

sigma^2 estimated as 0.1075: log likelihood=-52.79

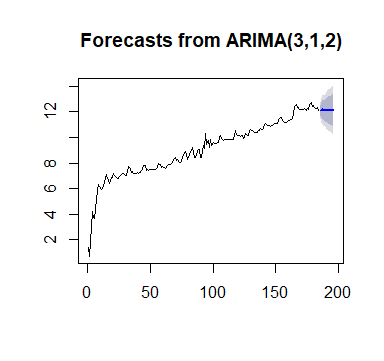
AIC=119.59 AICc=120.23 BIC=142.06

The output tells us that the model that best fits our original data is ARIMA(2,1,5).

What does this mean?

* 2 tells us that we need to take into account the Y value at 2 lags from a given time point t.
* 1 tells us that the time series is not stationary, so we need to take a first-order difference.
* 5 tells us that this model takes into account the error term from 5 preceding/lagged values.

After knowing the model that needs to build from ARIMA model we created the forecast of another 12 months ratings that will occur in electronics category. The predicted time line for the forecast is given below.



From above chart we can see the 95% confidence interval for the range of the monthly ratings in darker shaded part and the lighter shaded part shows the 80% confidence interval in the prediction of monthly rating. Table below shows the log value prediction of monthly ratings for another 12 months.

Point Forecast Lo 80 Hi 80 Lo 95 Hi 95

185 12.11908 11.69629 12.54187 11.47248 12.76568

186 12.11829 11.51723 12.71936 11.19904 13.03755

187 12.16416 11.45306 12.87526 11.07662 13.25170

188 12.13705 11.37571 12.89840 10.97268 13.30143

189 12.11665 11.28033 12.95298 10.83760 13.39570

190 12.12993 11.20096 13.05891 10.70919 13.55067

191 12.14656 11.14954 13.14357 10.62176 13.67136

192 12.13680 11.09053 13.18307 10.53667 13.73693

193 12.12541 11.02307 13.22775 10.43953 13.81129

194 12.13177 10.96633 13.29720 10.34939 13.91415

195 12.13990 10.92057 13.35923 10.27510 14.00470

196 12.13564 10.87079 13.40048 10.20122 14.07005

The R code used for this analysis is presented below.

ym <- read.table("E:/BGSU/4th semester/Big Data/project 2/year\_month.txt", header=TRUE)

t<-data.matrix(ym[1:184,])

ts.plot(t)

originalRating <- t[, 'rating']

adf.test(originalRating, alternative = "stationary")

LogTransformedYM<-log(t)

rating <- LogTransformedYM[, 'rating']

ts.plot(rating)

library("tseries")

adf.test(rating, alternative = "stationary")

acf(rating, lag.max=30)

pacf(rating, lag.max=30)

library("forecast")

auto.arima(rating)

ratingArima<-stats::arima(rating, order=c(3,1,2))

ratingForecasts <- forecast(ratingArima, h=12)

plot(ratingForecasts)

**Analysis 3: Cluster analysis and Logistic Regression**

The data was imported from the provided link where multiple datasets of Amazon ratings and reviews: <http://jmcauley.ucsd.edu/data/amazon/>. The electronics ratings csv file with product review has been used for this analysis and the file had 1689178 rows with file size as 304 Mb. To download the data it took around 3 minutes. To run the code it took around 2 minutes and to complete this analysis group members worked for more than 5 days to understand the process of cluster analysis and uses of different hyperparameters in different functions.

For task 2-analusis 3 our plan was to do cluster analysis based on review summary of Electronics product category. As a part for this analysis we also added logistic regression to analyze and predict whether reviews made were helpful or not to other Amazon customer.

Firstly, we started by importing “Amazon Instant Video” review data. The reason we imported this file was because it had the lowest number of reviews and file size. This will be helpful for us when executing and the running the code as it takes less time. Later we executed the same code in “reviews\_Electronics\_5.json.gz” file. To filter out helpful reviews we took only 4 variables from the overall data of Amazon Electronics product category. The four variable we took are “heplful”, “overall” and “summary”. At first we took “reviewText” but variable “reviewText” had lot of ineffective word which did made any sense in clustering. “summary” variable had the key words which were more effective in creating doing cluster analysis. After importing the file we separated the helpful column into two parts: HelpfulNumerator and HelpfulDenominator using following code.

instantVideo = instantVideo.withColumn("helpfulNumerator", col("helpful")[0])

instantVideo = instantVideo.withColumn("helpfulDenominator", col("helpful")[1])

In data cleaning part of the analysis we converted all the summary reviews into lower case to maintain consistency during clustering. We also removed all the punctuation so that words can be identified properly without any noise in the word. To clean the summery table by removing punctuation marks and making all the words in lower case we used following code:

instantVideo = instantVideo.withColumn("summary", regexp\_replace(lower(col('summary')), '[^\sa-zA-Z0-9]', ''))

To include only helpful reviews, we planned to take reviews that had Helpfulness Denominator greater than 10. After adding this constraint in our data, we were able to reduce our data to 97747 reviews from 168917 reviews. Using this constraint, we will be able to get words that are more effective in review by using following code.

selected = instantVideo.select([c for c in instantVideo.columns if c in ['summary', 'overall', 'helpfulNumerator', 'helpfulDenominator']])

selected = selected.filter('helpfulDenominator > 10')

Finally we added a new column which indicated whether a review is considered helpful or not giving value 1 or 0. If the ratio of helpfulNumerator and helpfulDenominator were less than 0.5 then we considered it as helpful else not helpful. This column will come in use for comparison when we make prediction from logistic regression. We used following code to extract the new column.

from pyspark.sql.functions import when

selected= selected.withColumn("helpful", when((selected['helpfulNumerator']) / (selected['helpfulDenominator']) > 0.50, 1).otherwise(0))

Following table shows the difference in data set before and running the codes mentioned above.

Original data set:

|  |  |  |  |
| --- | --- | --- | --- |
| **S.no** | **helpful** | **overall** | **summary** |
| **0** | **0:0** | **5.0** | **Gotta have GPS!** |
| **1** | 12:15 | 1.0 | Very Disappointed |
| **2** | 43:45 | 3.0 | 1st impression |
| **3** | 9:10 | 2.0 | Great grafics, POOR GPS |
| **4** | 0:0 | 1.0 | Major issues, only excuses for support |
| **5** | 3:3 | 5.0 | HDMI Nook adapter cable |
| **1689183** | 1:1 | 5.0 | Boom -- Pop -- Pow. These deliver. |
| **1689184** | 0:0 | 5.0 | Thin and light, without compromising on sound ... |
| **1689185** | 0:0 | 5.0 | Same form factor and durability as the S1 with... |
| **1689186** | 0:0 | 5.0 | Superb audio quality in a very comfortable set... |
| **1689187** | 0:0 | 5.0 | Exceptional sound |

Data set after cleaning:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.no** | Overall | Summary | **helpfulNumerator** | **helpfulDenominator** | helpful |
| **0** | 1.0 | very disappointed | 12 | 15 | 1 |
| **1** | 3.0 | 1st impression | 43 | 45 | 1 |
| **2** | 5.0 | real value for the money | 15 | 19 | 1 |
| **3** | 1.0 | what a piece of junk | 8 | 18 | 0 |
| **4** | 5.0 | excellent at any price | 14 | 19 | 1 |
| **97742** | 5.0 | omg i am so happy i could well have a beer at ... | 28 | 32 | 1 |
| **97743** | 5.0 | smart | 10 | 15 | 1 |
| **97744** | 5.0 | to use a technical termwow | 7 | 11 | 1 |
| **97745** | 5.0 | purely cons review | 5 | 12 | 0 |
| **97746** |  | best sounding speaker at this price range | 18 | 23 | 1 |

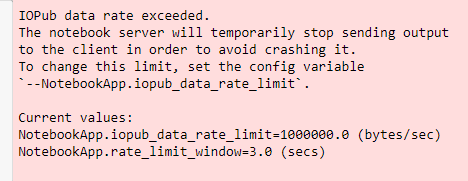
In this task for cluster analysis of the reviews, at first, we will be doing text feature extraction using "bag of words representation" method and also do cluster analysis using K-means clustering. For both the methods we will be using logistic regression

This method is widely used to get numerical features from text content which will give path to do further analysis in our project. To implement this method, we used

**TfidfVectorizer**(min\_df = 0.001, max\_df=0.999, ngram\_range=(1, 4), stop\_words='english')

function from scikit learn package of python. With the help of the function we can tokenize strings and give an integer id for each possible token, for instance by using white-spaces and punctuation as token separators. After that we count the occurrence of tokens in each document. Then we normalize and weight the less occurring but important tokens in most of the reviews.

We came across the problem when our value for min\_df=1 and max\_df=1. The screenshot of the problem is given below:

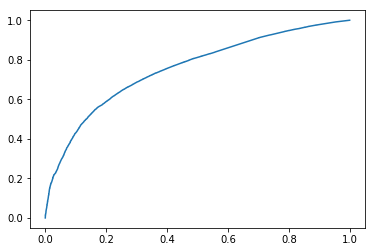


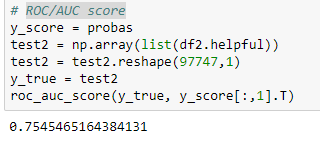
Here the value max\_df=1 means ignore terms that appear in more than 100% of the documents which as a result does not ignore any terms. Another parameter min\_df=1 means ignore terms that appear less than 1% document which as a result does not ignore any terms. Later we decided to exclude the top and bottom 0.001 of text that appears in the overall document. This means we will ignore the terms that appear in more than 0.999 of the documents and less than 0.001 of the document. As a output we got list of following words:

['10', '100', '1080p', '20', '30', '3d', 'absolutely', 'accessory', 'actually', 'adapter', 'addition', 'advertised', 'affordable', 'air', 'alternative', 'amazing', 'amazon', 'amp', 'android', 'angle', 'annoying', 'antenna', 'apple', 'asus', 'audio', 'available', 'average', 'avoid', 'away', 'awesome', 'awful', 'backup', 'bad', 'bag', 'bang', 'bang buck', 'bargain', 'basic', 'bass', 'batteries', 'battery', 'battery life', 'beat', 'beats', 'beautiful', 'believe', 'best', 'best value', 'better', 'better expected', 'beware', 'big', 'bit', 'black', 'bluetooth', 'bluray', 'bose', 'bought', 'box', 'brand', 'bright', 'broke', 'broken', 'buck', 'budget', 'build', 'build quality', 'built', 'buy', 'buyer', 'buyer beware', 'buying', 'cable', 'cables', 'cam', 'camcorder', 'camera', 'camera price', 'cameras', 'canon', 'capacity', 'car', 'card', 'careful', 'case', 'cd', 'charge', 'charger', 'cheap', 'cheaper', 'check', 'choice', 'class', 'clear', 'close', 'color', 'come', 'comes', 'comfortable', 'compact', 'company', 'compared', 'comparison', 'compatible', 'complete', 'computer', 'concept', 'connection', 'construction', 'control', 'convenient', 'cool', 'cord', 'cost', 'couple', 'cover', 'crap', 'customer', 'customer service', 'customer support', 'data', 'day', 'days', 'dead', 'deal', 'decent', 'defective', 'definitely', 'dell', 'description', 'design', 'designed', 'desktop', 'device', 'devices', 'did', 'didnt', 'didnt work', 'died', 'difference', 'different', 'difficult', 'digital', 'digital camera', 'disappointed', 'disappointing', 'disappointment', 'display', 'doa', 'dock', 'does', 'does job', 'does work', 'doesnt', 'doesnt work', 'dont', 'dont buy', 'dont waste', 'dont waste money', 'drive', 'drives', 'dslr', 'dual', 'dvd', 'dvd player', 'easily', 'easy', 'easy install', 'easy setup', 'easy use', 'effective', 'end', 'entry', 'entry level', 'especially', 'exactly', 'excellent', 'excellent camera', 'excellent product', 'excellent sound', 'excellent value', 'exceptional', 'expect', 'expectations', 'expected', 'expensive', 'experience', 'external', 'extra', 'extremely', 'failed', 'fair', 'fan', 'fantastic', 'far', 'far good', 'fast', 'favorite', 'feature', 'features', 'filter', 'finally', 'fine', 'firmware', 'fit', 'fits', 'flash', 'flaw', 'flawed', 'flaws', 'focus', 'frame', 'free', 'fun', 'function', 'functional', 'functionality', 'gadget', 'galaxy', 'gaming', 'garbage', 'garmin', 'gb', 'generation', 'gets', 'getting', 'going', 'good', 'good camera', 'good great', 'good price', 'good product', 'good quality', 'good sound', 'good value', 'got', 'gps', 'great', 'great buy', 'great camera', 'great case', 'great deal', 'great device', 'great features', 'great lens', 'great little', 'great picture', 'great price', 'great product', 'great quality', 'great sound', 'great tv', 'great value', 'hands', 'handy', 'happy', 'hard', 'hard drive', 'hardware', 'hate', 'hd', 'hdmi', 'hdtv', 'head', 'headphone', 'headphones', 'headset', 'heavy', 'heres', 'high', 'high quality', 'highly', 'highly recommended', 'home', 'home theater', 'horrible', 'hp', 'hub', 'huge', 'hype', 'idea', 'ii', 'im', 'imac', 'image', 'image quality', 'images', 'impressed', 'impressions', 'impressive', 'improved', 'improvement', 'included', 'incredible', 'inexpensive', 'install', 'installation', 'instead', 'instructions', 'interface', 'internet', 'ipad', 'iphone', 'ipod', 'isnt', 'issue', 'issues', 'item', 'ive', 'ive owned', 'job', 'junk', 'just', 'keyboard', 'kindle', 'kit', 'know', 'lacking', 'laptop', 'large', 'lcd', 'led', 'lens', 'lenses', 'let', 'level', 'life', 'light', 'lightweight', 'like', 'limitations', 'limited', 'line', 'little', 'little camera', 'live', 'logitech', 'long', 'look', 'looking', 'looks', 'lot', 'lots', 'lousy', 'love', 'low', 'low light', 'low price', 'mac', 'macbook', 'macbook pro', 'machine', 'macro', 'major', 'make', 'make sure', 'makes', 'manual', 'market', 'maybe', 'media', 'mediocre', 'memory', 'micro', 'mini', 'minor', 'misleading', 'missing', 'mixed', 'mode', 'model', 'money', 'monitor', 'monster', 'month', 'months', 'mount', 'mouse', 'movies', 'mp3', 'mp3 player', 'music', 'nas', 'need', 'needed', 'needs', 'netbook', 'network', 'new', 'nice', 'nice camera', 'nice little', 'nikon', 'noise', 'note', 'notebook', 'ok', 'okay', 'old', 'older', 'olympus', 'optical', 'option', 'options', 'original', 'os', 'outstanding', 'overall', 'overpriced', 'owned', 'owners', 'package', 'palm', 'panasonic', 'pay', 'pc', 'pda', 'people', 'perfect', 'perfectly', 'performance', 'performer', 'phone', 'photo', 'photography', 'photos', 'picture', 'picture quality', 'pictures', 'piece', 'piece junk', 'plasma', 'plastic', 'play', 'player', 'pleased', 'plug', 'plus', 'pocket', 'point', 'point shoot', 'poor', 'poor quality', 'poorly', 'port', 'portable', 'potential', 'power', 'powerful', 'pretty', 'pretty good', 'previous', 'price', 'priced', 'pricey', 'prime', 'pro', 'probably', 'problem', 'problems', 'product', 'products', 'professional', 'projector', 'pros', 'protection', 'protector', 'purchase', 'purchased', 'purpose', 'quality', 'quick', 'quiet', 'quite', 'radio', 'range', 'rating', 'read', 'reader', 'ready', 'real', 'really', 'reasonable', 'receiver', 'reception', 'recommend', 'recommended', 'recorder', 'reliable', 'remote', 'replacement', 'results', 'return', 'returned', 'review', 'reviews', 'right', 'rocks', 'room', 'router', 'run', 'samsung', 'sandisk', 'satisfied', 'save', 'save money', 'say', 'says', 'screen', 'sd', 'second', 'security', 'series', 'service', 'set', 'setup', 'sharp', 'shoot', 'short', 'simple', 'simply', 'size', 'sleek', 'slow', 'slr', 'small', 'smart', 'software', 'solid', 'solution', 'sony', 'sound', 'sound quality', 'sounds', 'space', 'speaker', 'speakers', 'speed', 'stand', 'standard', 'star', 'stars', 'stay', 'stay away', 'step', 'stereo', 'storage', 'streaming', 'strong', 'stunning', 'sturdy', 'super', 'superb', 'support', 'supposed', 'sure', 'surprised', 'surprisingly', 'sweet', 'switch', 'tab', 'tablet', 'takes', 'tech', 'technology', 'terrible', 'terrific', 'thats', 'theater', 'thing', 'things', 'think', 'thought', 'time', 'tiny', 'tivo', 'tool', 'toshiba', 'touch', 'toy', 'travel', 'tried', 'tripod', 'true', 'truly', 'try', 'tv', 'unit', 'unreliable', 'update', 'updated', 'upgrade', 'usb', 'usb 30', 'use', 'used', 'useful', 'useless', 'user', 'users', 'using', 'value', 'value money', 'versatile', 'version', 'video', 'vs', 'wait', 'want', 'wanted', 'warning', 'warranty', 'waste', 'waste money', 'watch', 'way', 'weak', 'weeks', 'whats', 'wide', 'wifi', 'windows', 'winner', 'wireless', 'wish', 'wonderful', 'wont', 'work', 'worked', 'working', 'works', 'works advertised', 'works fine', 'works great', 'works like', 'works perfectly', 'world', 'worse', 'worst', 'worth', 'worth money', 'worth price', 'worthless', 'wow', 'wrong', 'year', 'years', 'yes', 'youll', 'youre', 'zoom']

The world mentioned above were appearing in most of the document and we will create a vector of those words using tf-idf algorithm to give those words a numerical value based on their reputation on review document.

After that we decided to use logistic regression to make prediction for the review that can be considered as helpful. In order to run logistic regression first we wanted to find best model parameters for the logistic regression. So we used grid\_search.GridSearchCV function from sklearn package to determine the best model parameter. From the best model we created logistic regression. With the help of the logistic regression we predicted the helpfulness of the reviews using the cross validation of the data. In order to evaluate the performance for the model we used receiver operator characteristics curve and ROC/AUC score. The following is the result that we got for cluster analysis using "bag of words representation" method followed by logistic regression.



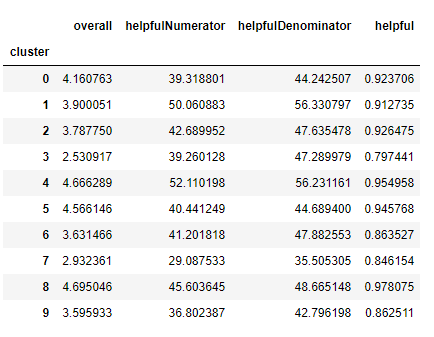


From the ROC curve, we can see that the while capturing true positive rate we are also capturing false positive at higher rate. It means that the model is not capturing the true values properly.

In the next result, we can know the score of ROC/AUC is 0.75, and it is not closer to 1. We can compare this score with the next model that we are going to build and compare which one is better.

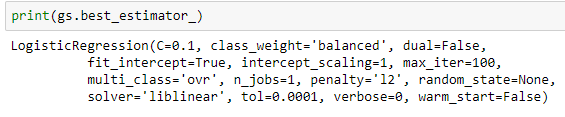
We again repeated the same process again but now we tried clustering the words using K-means neighbor

For cluster analysis using k-means neighbor we figured out top words that was frequently used in the review and added those words into to the training data set. We also decided to make 10 clusters to continue with our analysis. The following table shows the average score for all the clusters.

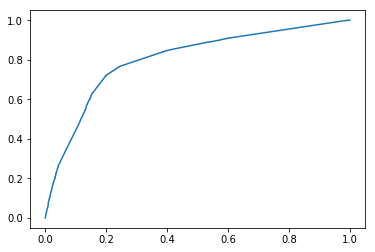


From this result, we can see that the degree of helpful in all the top 10 clusters is around 0.9, and it is very close to 1. And it also proves the importance of the top words.

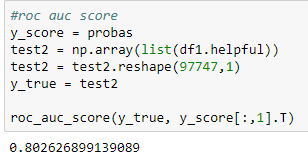
After that we decided to use logistic regression to make prediction for the review that can be considered as helpful. In order to run logistic regression we again repeated the same process as before in finding out the best model parameters for the logistic regression. So we used again grid\_search.GridSearchCV function from sklearn package to determine the best model parameter. As a best model estimator we got following result and implemented these parameters to create logistic regression.



From the best model we created logistic regression. With the help of the logistic regression we predicted the helpfulness of the reviews using the cross validation of the data. In order to evaluate the performance for the model we used receiver operator characteristics curve and ROC/AUC score. The following is the result that we got for cluster analysis using "k-means" method followed by logistic regression.



From the ROC curve, we can see that the curve is clearly above the diagonal line, and the angle between the equal error point is much larger than the diagonal line which deviates from 45 degrees. We can conclude that the classification has a larger true positive rate and smaller false positive rate, the accuracy of the classification is enough large.



Also, the ROC accuracy score is 0.8, means that the true positive rate equal 0.8, and it is larger than 0.5. Therefore, it does prove the effect of classification is better.

Confusion matrix, without normalization

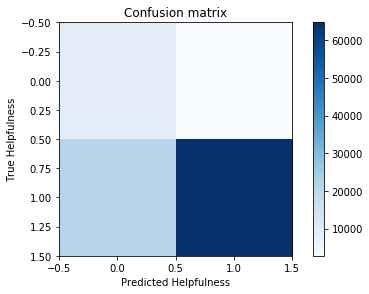
[[ 9201 2795]

[21008 64743]]

Normalized confusion matrix

[[0.77 0.23]

[0.24 0.76]]



In this confusion matrix, the depth of blue represents the degree of helpfulness. There are four different result, true positive, false positive, false negative and true negative from left to right and from top to bottom separately. According to the result, we can know that the true positive is same as true negative, and false positive is same as false negative in the normalized confusion matrix. Moreover, the true predict far outweigh than false predict. Therefore, we can prove that the classification has high correct rate again.

The result of original confusion matrix is different from the normalized confusion matrix, and the reason is that the unit measure of x and y coordinate is different, and the effect of small values can easily be ignored. Once normalized, the results will be more accurate and closer to the real situation.

[('lens', -0.6736176587525573),

('tv', -0.5571637341388087),

('great product', -0.4454875647782483),

('ipod', -0.4371827207848933),

('im', -0.36897623660028467),

('owned', -0.2957230320815657),

('expected', -0.2515360440413292),

('ok', -0.20106027259811332),

('excellent product', -0.19707908840217955),

('service', -0.18351960417462143),

('camera', -0.1782318786316096),

('just', -0.1754267518300947),

('product', -0.15792122336632514),

('good product', -0.10736426522662708),

('buy', -0.10316867316215607),

('hate', -0.07763600788762927),

('excellent value', -0.0558368984968539),

('good price', -0.040150948437429466),

('way', -0.029194219301056284),

('terrible', -0.0025020832797622316),

('bad', 0.06853349277942034),

('far', 0.08348619210842122),

('available', 0.09143006212438266),

('pretty', 0.10505822198056962),

('far good', 0.11416481471930179),

('love', 0.11952074135665025),

('excellent camera', 0.13542551538712586),

('best', 0.1520476269495929),

('horrible', 0.18425784541840826),

('money', 0.19640586195376014),

('options', 0.20111077123920704),

('awesome', 0.24290936811366945),

('speaker', 0.24301869035993012),

('market', 0.26733957175142087),

('customer service', 0.2919316608360291),

('better', 0.3350925718789468),

('thing', 0.33659881821505766),

('pretty good', 0.34027549782261546),

('ive', 0.3458968488954756),

('warranty', 0.3746365178087729),

('wanted', 0.3792801838354424),

('best value', 0.3848141267418181),

('absolutely', 0.38793790498247255),

('customer', 0.4253828099394893),

('bluetooth', 0.44903212429047984),

('poor', 0.46920189094203824),

('nice', 0.4721093596300803),

('good', 0.4889701845691787),

('quality', 0.5192134418718003),

('price', 0.5594546411835593),

('case', 0.5602800070635706),

('sound', 0.5668320750266145),

('score', 0.7190711634731852),

('little', 0.7364397541758378),

('great', 0.7590746577130678),

('good value', 0.9136167225452054),

('better expected', 0.9539949867935091),

('works', 0.98976863088867),

('portable', 1.1177233220639098),

('excellent', 1.33370844342798),

('value', 1.3501490457123313),

('perfect', 1.412516181670701)]

From this result, we can know that the larger the value, the more helpful the word is to the reviews. It is obviously that these words have important affect, such as “perfect”, “value”, “excellent”, “portable”, “works”, and so on. However, there are some words has little effect even negative helpfulness, such as “lens”, “tv”, “im”, “ipod”and so on. It is true that some words will not help the reviews, or even interfere with it.

In conclusion comparing the roc/aoc curve and its score we can say that for amazon electronics category data k-means clustering followed by logistic regression will produce more accurate result for determining helpfulness of the reviews.

# Source Code

**Task 0 and 1:**

****

**Task 2:**

**Exploratory Data Analysis**

****

**Time Series Analysis**

****

**Clustering Analysis**

