Capstone 1 Final Report: Sentiment Analysis on Amazon Reviews

Written by Pravati Swain

Table of Contents

1. Introduction

E-commerce has revolutionized today’s shopping experience. Online customer reviews and ratings of every product/brand/service help the consumers to make a smart and informed decision before buying a product or service. Customer reviews are also beneficial for manufacturers to improve their products and services. Retail websites like Amazon.com allow users to submit their opinion in the form of numerical stars (from 1-5) or comments about a product. These star ratings provide excellent labels for training ML models to predict the sentiment of text related to a particular product or line of products. This allows sellers to get an overview of sentiment for a large number of unlabelled reviews and in some cases can and help sellers understand how customers are feeling about their product. In this project, I will analyze and build an ML model to evaluate the positive and negative sentiment of an Amazon.com product review.

2. Client Profile

Sentiment analysis of product reviews has diverse applications. E-commerce companies such as Amazon.com, eBay.com, etc and technology companies (Apple, Microsoft, etc) can use it to predict what people think about their product or market trend. Social media companies can also be used to study the sentiment of social conversations. The machine learning algorithm could be used for any kind of business that has an online database of reviews. They can predict the sentiment of the review to understand customer emotions and hence can improve their products or services.

3. Description of Dataset

For this project, I will use the **baby products**, **grocery_foods** and **call phones** reviews datasets from Amazon.com. All the three datasets contain between 200,000 to 150, 000 reviews. The dataset is made available by Julian McAuley, UCSD and is available by request. The dataset contains information about reviewer ID, asin (product ID), reviewer name, helpful (helpfulness rating of the review), review text, overall (rating of the product), summary (summary of the review), review time. For this project, the mostly used features are review text and rating. Every rating is based on a 5-star scale, where all the ratings ranged from 1 to 5 star. The datasets were downloaded as the JSON file and loaded into Pandas DataFrame in the jupyter notebook.

Amazon Review Data Link: http://jmcauley.ucsd.edu/data/amazon/

4. Data Wrangling

After acquiring the dataset and loading into Pandas DataFrame several data cleaning and data preprocessing steps were required in order to use for data analysis and machine learning.

- I checked for missing and duplicate values and dropped the duplicate values.
  - df.shape and df.columns attribute to find out the number of rows and columns, and column names respectively
  - Renamed some of the columns using df.rename() function.
  - Counted the missing values using isnull() and info() methods
  - Filled the missing values of the reviewer's name with 'No Name' using fillna() method
  - Checked for duplicate reviews using duplicated() method and drop the duplicate values using drop_duplicates() method.
- The short reviews of less than 3 words were removed.
- Since the model won't work on non-English reviews I dropped the non-English reviews using nltk library.
- A new feature 'label' added where 5 rated reviews are labelled as 1 and less than 5 rated reviews are labelled as 0. Similar steps are done in all of the three datasets.

5. Exploratory Data Analysis and Inferential Statistics

In this exploration of the Amazon product reviews dataset, I analyzed the features of review text with respect to ratings. For this project, I used three product categories: baby products, grocery and gourmet foods, and cellphones dataset.

These are the questions I looked in the data:

- What does the distribution of data based on review categories (or star ratings) and distribution of review length look like?
- What does the distribution of the number of ratings in different product categories look like?
- How do the reviews change across high and low star ratings based on review length and percentage of uppercase words per review for different product categories?
- What are the most frequently used words for high and low ratings?

5.1 Distribution of rating and review length by product categories

I have plotted the distribution of ratings for the three different product categories and noticed that most of the reviews are five stars in all three categories (**Fig 1**). I also see that cell phones tend to have more ratings of 1 than the other categories.

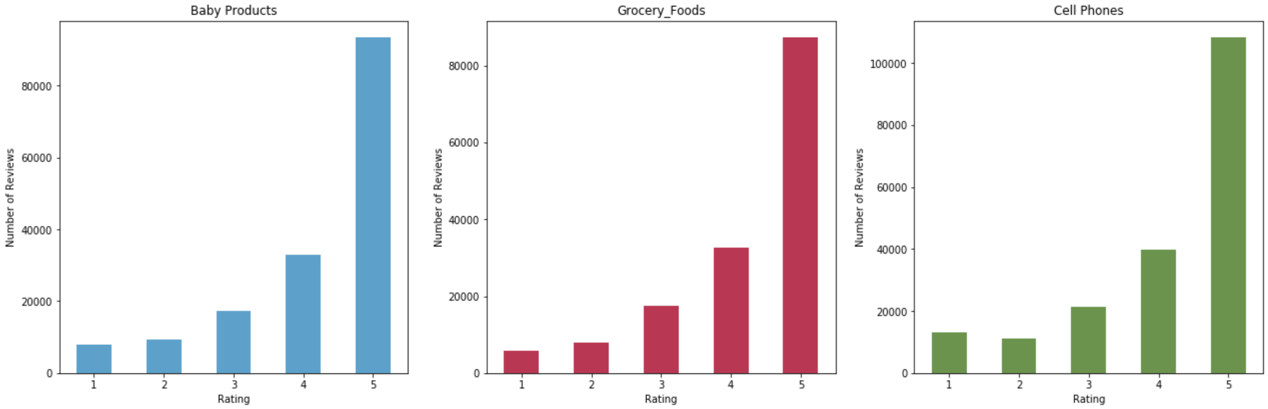


Fig 1 Distribution of rating by product categories

From the distribution of review length by product categories (**Fig 2**), we can see the cell phone reviews are shorter in length compared to the other two product categories. About 74.8% of cell phone reviews are of length less than equal to 100, whereas baby products and grocery food categories have about 56% and 53% of reviews of length less than or equal to 100 respectively.

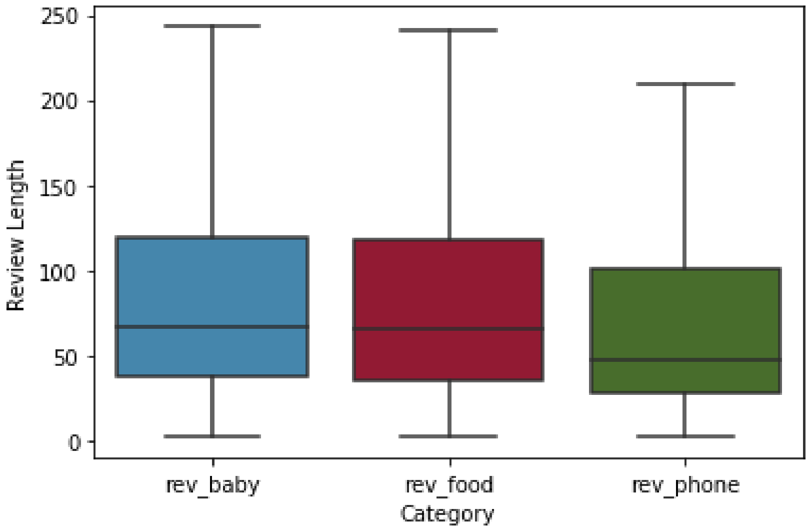


Fig 2 Distribution of review length by product categories

In all of the three product categories, five star rated reviews are found to be the shortest length followed by one star rated reviews (**Fig 3**). Two, three and four star rated reviews are longer and show different trends in different products categories. In the case of baby products and cell phones categories four star rated reviews are longer and in grocery_foods three star rated reviews

are longer. From this exploration we can see that, when people are highly satisfied or dissatisfied with the product, they write very short reviews and, in other cases, they have to explain where and which part of the product they are unsatisfied with hence the reviews are longer. As we can see an example of a short 5 star review below:
Example of 5 star review with less than 5 words:

'Love this product'
'Great little headset.'
'Nice Case. Great Price.'

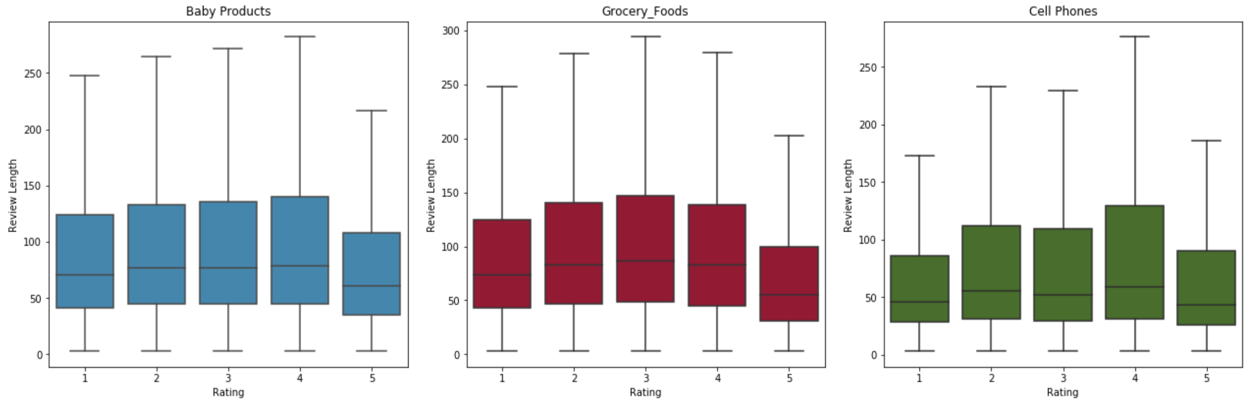


Fig 3 Distribution of Review Length by Rating

5.2 Distribution of percent capitals by ratings

The distribution of % capitals by ratings (**Fig 4**) shows that the reviews corresponding to 1-star and 5-star ratings have more uppercase letters and the three star rated reviews have least upper case letters. It might be due to highly satisfied or dissatisfied customers writing reviews of short sentences which contain more uppercase letters.

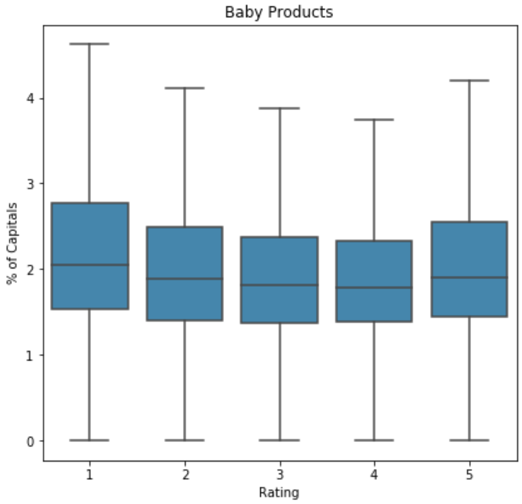


Fig 4 Distribution of % of Capitals by Rating

The % of capitals show bimodal distribution in the three product categories (**Fig 5**), one peak centered at around 20% and the other peak centered at around 70% of capitals.

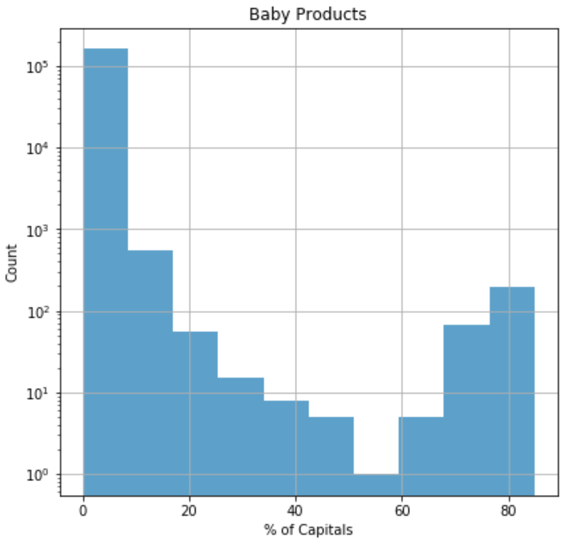


Fig 5 Distribution of % of Capitals

In **Fig 6** we have plotted the average number of reviews >= 60% capital letters grouped by rating. 1 and 5 star reviews are more likely to contain reviews with >= 60% capital letters across product categories then other star ratings.
Example of a reviews with >= 60% capitals:

'AMAZING SOUND, MOVIE THEATER QUALITY' (5-star)
"COULDN'T DOWNLOAD ANYTHING" (1-star)

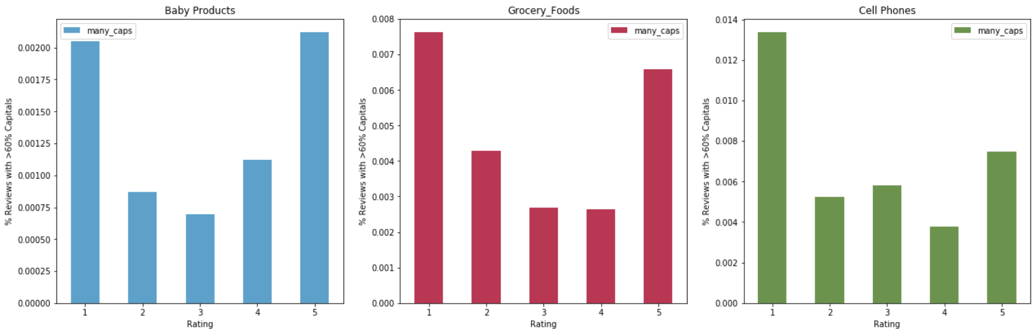
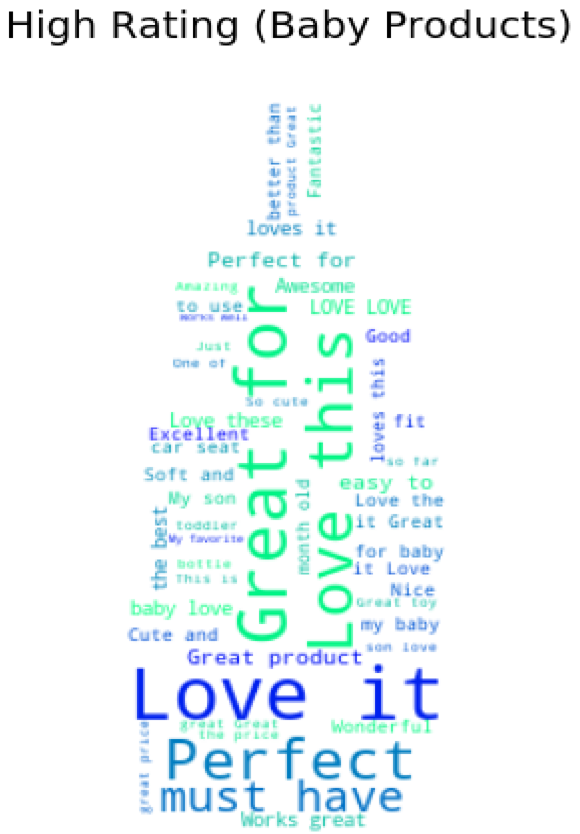
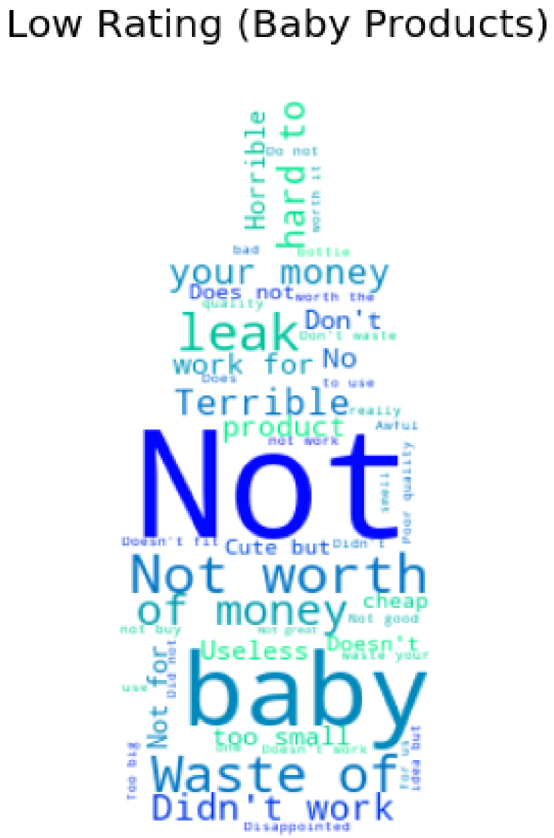


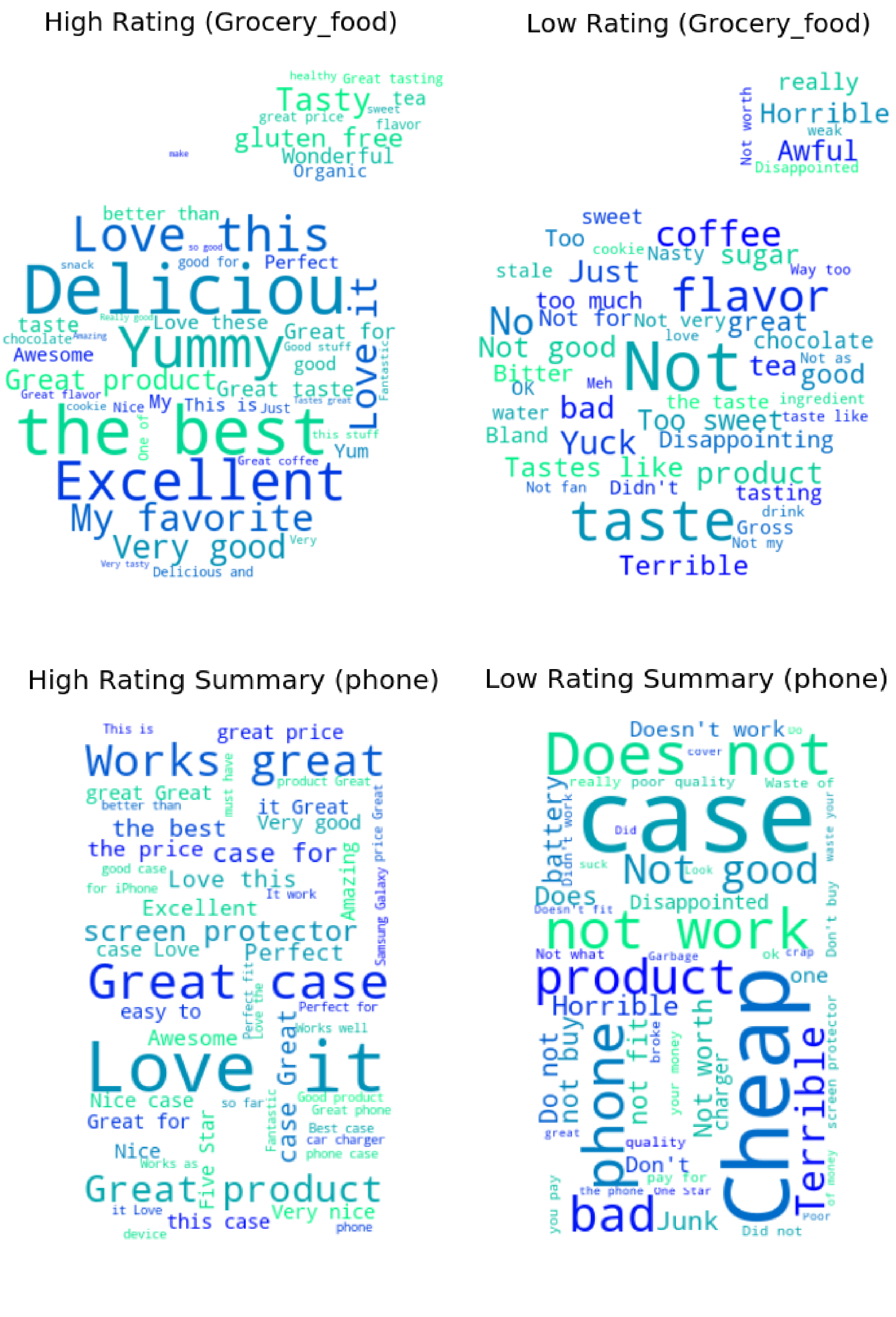
Fig 6 Distribution of % of capitals (> 60%) by rating

5.3 World Clouds for high and low rated reviews

The word clouds from review summaries are plotted by high and low ratings. In this case I have taken 5 stars ratings as high and 1- 4 stars as low ratings. In the word clouds the font size of a word indicates its frequency and importance in the review. The words used more often in the reviews are greater in font size and darker in color. The size of the word decreases as the frequency of the word in the reviews decreases. As we can see from the word cloud of high rated grocery_foods summary, the words like ‘Delicious’, ‘Yummy’, ‘Testy’, ‘Excellent’, ‘the best’, ‘Love this’ looks bigger and darker as they were used more frequently used in the reviews. On the other hand in the word cloud of low rated reviews the words like ‘Not good’, ‘Terrible taste’, ‘Disappointing’, ‘Yuck’ are the most used words.







High Rating (Grocery_food)

Low Rating (Grocery_food)

High Rating Summary (phone)

Low Rating Summary (phone)

Fig 7 Word clouds for high and low rated reviews for all three datasets

5.4 Summary of findings from Exploratory Data Analysis

From exploratory data analysis we found in all of the three product categories:

- Most of the reviews are 5 star
- Five and one star rated reviews are found to be shorter
- Reviews corresponding to 1-star and 5-star ratings have more uppercase letters and three star rated reviews have least upper case letters.
- The % of capitals show bimodal distribution, one peak centered at around 20% and the other peak centered at around 70% of capitals.
- 1 and 5 star reviews are more likely to contain >= 60% capital letters across product categories than other star ratings.

5.5 Inferential statistical data analysis

For the statistical data analysis all the ratings are divided into two groups which are 5-star and not_5 star (1, 2, 3, and 4 star rating). The reviews associated with 5-star rating are compared with lower rated reviews (not_5 star) based on the word counts and percent of uppercase letters present per review. Statistical comparison of these features between 5-star and other lower rated reviews are described below.

1. *Comparison of review length between 5-star and not_5 star rating*

To compare the review length (measured by word count) for 5-star and not5-star rating, an independent two-sample t-test for unequal variance is used. Below are the hypotheses of interest.

Null Hypothesis: The review length for 5-star and other rated reviews are the same.
Alternate Hypothesis: The review length for 5-star and other rated reviews are different.

Baby Products: t = 41.398 , p = 0.000
Grocery_Foods: t = 58.596, p = 0.000
Cell Phones: t = 23.899, p = 0.000

In all the three product categories, since p_value is less than 0.05 we reject the null hypothesis and conclude that the word counts for 5-star rated reviews are significantly shorter than all 1-star through 4-star (not5) rated reviews.

2. *Comparison of % caps between 5-star and not_5 star rating*

An independent two-sample t-test for unequal variance is used to compare the % caps between 5-star and other lower rated reviews.

Null Hypothesis: The % of capitals for 5-star rated reviews are the same as other rated reviews.
Alternate Hypothesis: The % of capitals for 5-star rated reviews are different from the other rated reviews.

Baby Products: t = 12.134, p = 0.000
Grocery_Foods: t = 13.586, p = 0.000
Cell Phones: t = 12.302, p = 0.000

In all of the three product categories, since p_value is less than 0.05 we reject the null hypothesis and conclude that the % of capitals for 5-star rated reviews are higher than the other rated reviews.

Review length for 5-star rated reviews are shorter than other rated reviews and the % of capitals for 5-star rated reviews are higher than the other rated reviews

6. Machine Learning

The aim of the machine learning model is to take the new reviews and classify the sentiment of the review either as positive (five star) or negative (not five star). Before machine learning we need to pre-process the text reviews. Following preprocessing I divided the datasets into training and testing datasets. Then I did feature extraction or vectorization to encode the text data into integer or floating point values before use as input to machine learning algorithms. I applied and compared two vectorization methods: CountVectorizer and TfidfVectorizer from the scikit-learn library and selected the best performing vectorization method. Then I fitted, tuned the parameters and compared different algorithms using the training datasets for predicting the sentiment of the test dataset. To further improve the performance of a classifier I did thresholding for different business situations.

6.1 Text Pre-Processing

For machine learning we need the text data to be in the numeric form. We use different encoding techniques to encode the text into numeric vectors. Before encoding we first need to clean the text data and this process is called text-preprocessing.
These are the following steps I used to preprocess the text data.

- Removing URL
- Keeping only alphabets (removing numbers and punctuations)
- Lowercase all text
- Tokenization using word_tokenize from NLTK (the review text becomes a list of strings where each word or token is a string)
- Removing stopwords using NLTK english stopwords. Stopwords are the most common words like “the”, “a”, “me”, “is”, “to”, “all” and don’t carry important meaning.
- Lemmatization (to derive the base or dictionary form of each word)

6.2 Train and Test Split

I split the data into training and test sets using Scikit-learn's train_test_split method with a test_size of 30%.

6.3 Vectorizer Selection

Different vectorizers use different scoring methods to encode text data. Here, I used and compared two methods, CountVectorizer and TfidfVectorizer, which are provided by scikit-learn library.

CountVectorizer converts text to word count vectors. In this case an encoded vector (which is a sparse vector) is returned with a length of the entire vocabulary and an integer count for the number of times each word appears in the document. However, the problem with scoring word count is that some words like 'the' have large counts but will not be very meaningful.

TfidfVectorizer uses TF-IDF scoring approach. **Term Frequency** (TF) is a scoring of the frequency of the word in the current document. **Inverse Document Frequency** (IDF) is a scoring of how rare the word is across documents. TF-IDF are word frequency scores that highlight words that are distinct and contain useful information in a given document.

To compare the two vectorizers first I instantiated both CountVectorizer and TfidfVectorizer with default parameters (min_df=1, ngram_range=(1, 2)) and then saved the document-term matrices from the fit and transform steps of vectorization on the X_train and X_test matrices of all the three product categories datasets. Then I instantiated a simple Naive Bayes classifier, MultinomialNB(), and trained it on the X_train document term matrix and y_train and then made class prediction using X_test document term matrix and calculated ROC-AUC scores to compare both the vectorizers.

Then I did hyperparameter tuning to get the best min_df for both the vectorizers. In the case of baby products dataset the min_df was 0.001 for both the vectorizers, however for Grocery_foods and cell_phones dataset I choose the min_df = 50 due to the computational limitation of my PC. Then I did GridSearchCV to select the best parameter (alpha) for the simple Naive Bayes classifier. Then I recalculated the ROC-AUC scores after hyperparameter tuning. **Table 1** , compares the vectorizers with and without hyperparameter tuning for three datasets. Looking at the table we can see TfidfVectorizer worked best and hence I used the document terms matrix from the TfidfVectorizer on other classifiers.

Table 1 Comparison of vectorizers with a Multinomial Naive Bayes Model with and without hyperparameter tuning for baby_products, grocery_foods and cell_phones datasets.

Baby_products Dataset		
Vectorizer	**ROC-AUC**	**Best Parameters**
CountVectorizer	0.742	min_df = 1, alpha=1
TfidfVectorizer	0.832	min_df = 1, alpha =1
CountVec w/ GridSearch	0.828	min_df = 0.001, alpha =5
TfidfVec w/ GridSearch	0.842	min_df =0.001, alpha =1
Grocery_foods Dataset		
Vectorizer	**ROC-AUC**	**Best Parameters**
CountVectorizer	0.758	min_df = 1, alpha=1
TfidfVectorizer	0.839	min_df = 1, alpha=1
CountVec w/ GridSearch	0.819	min_df = 50, alpha=0.1
TfidfVec w/ GridSearch	0.834	min_df = 50, alpha=5
Cell_phones Dataset		
Vectorizer	**ROC-AUC**	**Best Parameters**
CountVectorizer	0.747	min_df = 1, alpha=1
TfidfVectorizer	0.833	min_df = 1, alpha=1
CountVec w/ GridSearch	0.802	min_df = 50, alpha=0.001
TfidfVec w/ GridSearch	0.824	min_df = 50, alpha=10

6.4 Model Comparison

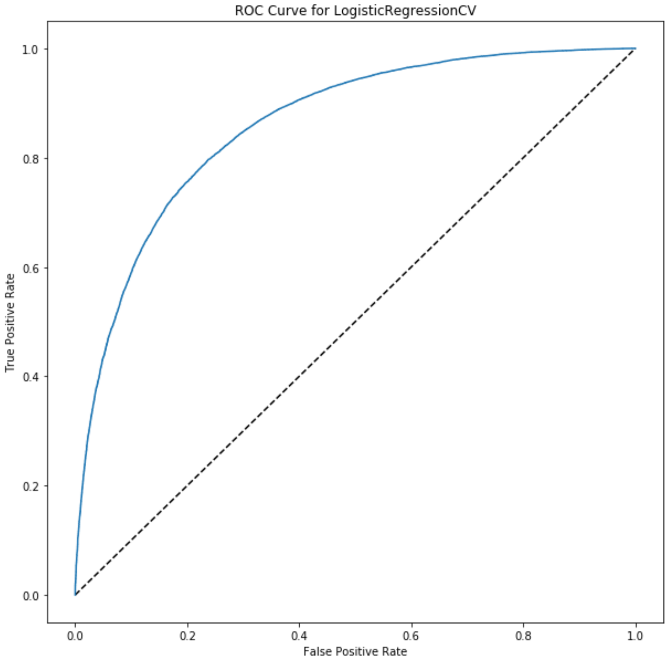
Using the TfidfVectorizer I fitted and tuned three classifiers: Logistic Regression, Multinomial Naive Bayes and Random Forest Trees. For baby_products dataset each classifier used an ngram_range of (1,2) and a min_df of 0.0001, however, for other two datasets(grocery_foods and cell_phones) I used min_df=50. I used the AUC score to compare the models to one another. For three of the models I used GridSearchCV for cross-validation and hyper parameter tuning.

The highest scoring model turned out to be Logistic Regression. The score and parameters of each model are shown in **Table 2** along with the best ROC curve (**Fig 8**) for all the three datasets.

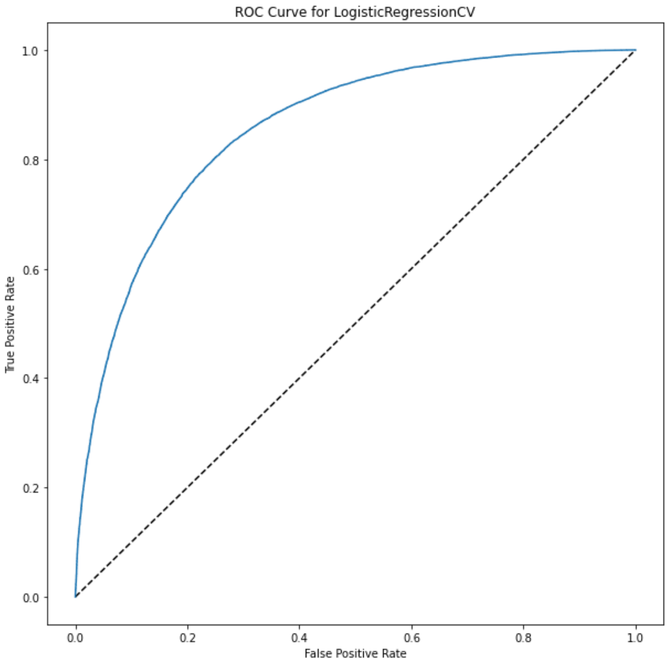
Table 2 Comparison of three machine learning models fitted with TfidfVectorizer for three datasets

Baby_products Dataset		
Classifier	**ROC-AUC**	**Best Parameters**
MultinomialNB	0.842	alpha =1, fit_prior = True
LogisticRegressionCV	0.859	C= 0.85, max_iter=1000
RandomForestClassifier	0.835	max_depth= 100, max_feature= sqr, n_estimators= 500
Grocery_foods Dataset		
Classifier	**ROC-AUC**	**Best Parameters**
MultinomialNB	0.834	alpha=5, fit_prior=True
LogisticRegressionCV	0.856	C= 0.85, l1_ratio= 0
RandomForestClassifier	0.825	max_depth= None, max_features= sqrt, n_estimators=100
Cell_phones Dataset		
Classifier	**ROC-AUC**	**Best Parameters**
MultinomialNB	0.824	alpha =10, fit_prior = True
LogisticRegressionCV	0.846	C= 0.85, l1_ratio= 0
RandomForestClassifier	0.819	max_depth= None, max_features= auto, n_estimator= 100

Baby_products Dataset



Grocery_foods Dataset



Cell_phones Dataset

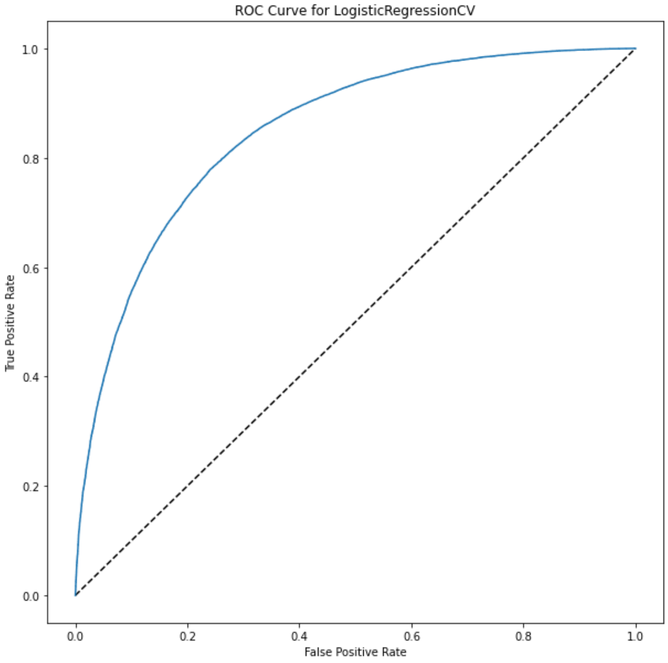


Fig 8 ROC curve for Logistic Regression (best model) for three datasets

6.5 Most Predictive Features

I also looked at 10 most predictive words within the entire corpus along with their probabilities for all the three different datasets.

Baby_products Dataset

Positive words	**P((positive) | word)**
say enough	0.95
recommend everyone	0.93
every penny	0.92
recommend enough	0.91
worth every	0.91
great highly	0.91
highly recommend	0.91
penny	0.91
could happier	0.90
totally recommend	0.90

Bad words	**P((positive) | word)**
give three	0.13
refund	0.12
poorly design	0.12
would return	0.12
execution	0.10
return item	0.09
give four	0.09
one star	0.09
give instead	0.04
reason give	0.03

From the most predictive features of baby products, we can see the words like ‘poorly design’ ‘refund’ and ‘would return’ says that consumers are very unsatisfied with poorly designed products and ask for refund or return the products. On the other hand, words like ‘totally recommend’, ‘worth every’, ‘every penny’ etc says that customers are satisfied with good products with good prices.

Grocery_foods Dataset

Positive words	**P((positive) | word)**
highly recommend	0.90
great price	0.89
highly	0.87

hook	0.87
absolutely delicious	0.87
glad find	0.87
best ever	0.86
wonderful	0.86
far best	0.85
best price	0.85

Negative words	**P((positive) | word)**
refund	0.23
terrible	0.23
plus side	0.22
taste bad	0.22
think stick	0.20
worst	0.20
one star	0.17
nothing special	0.16
reason give	0.16
give four	0.16

Cell_phones Dataset

Positive words	**P((positive) | word)**
highly recommend	0.89
love love	0.88
highly	0.87
absolutely love	0.85
thank much	0.84
love much	0.84
recommend everyone	0.83
would highly	0.83
love	0.82
love case	0.82

Negative words	**P((positive) | word)**
guess get	0.21
useless	0.20
trash	0.20
send back	0.20
waste	0.19
get pay	0.18
one star	0.18

refund	0.17
reason give	0.16
waste money	0.16

In the case of grocery and gourmet foods, the words like ‘absolutely delicious’, ‘great price’, ‘highly recommend’ are the words used by customers who are very satisfied with the products. On the other hand the words like ‘taste bad’, ‘worst’, ‘refund’, ‘terrible’ says that the food items which taste bad got a bad rating and asked for refund.
Similarly in case of cell phones products, the words like ‘absolutely love’, ‘thank much’, ‘love case’ , ‘recommend everyone’ are used by customers who are satisfied with the products and words like ‘useless’, ‘trash’, ‘waste money’, ‘refund’ used by customers unsatisfied with the products.

6.5 Thresholding for different business situations

The default threshold for interpreting probabilities to class labels is 0.5 and tuning this is called thresholding.

For the sentiment analysis classifier if the predicted probability of a review based on the text of the review is greater than equal to 0.5 then that review is classified as positive or rated five. Since our review datasets are heavily skewed towards positive ratings the default threshold may not be the best threshold for the sentiment classification.

For sentiment classification of amazon product reviews my aim is to minimize the number of false positive rating classifications. Because there are many positive reviews (rated five) and less negative reviews (rated not five) in the datasets, it is less harmful to accidentally classify positive reviews as negative as there are many examples of positive reviews correctly classified. The aim is to classify the True Negative reviews as ‘negative’ or ‘low’ rated reviews as frequently as possible.

Business Case 1: Understanding Overall Customer Satisfaction

One business case for this form of sentiment analysis would be to gauge the overall reaction of customers to a new product in the marketplace. For this, we’re more interested in having an accurate overall sentiment number then we are in diving into individual reviews. For this reason, we optimize for **balanced accuracy**. This way, we are equally valuing our classifications of low and high reviews.

For the best model, LogisticRegression, in case of baby_products dataset the threshold level of 0.58 provides the best results with an balanced accuracy of 0.779. In total there are 10651 mispredictions, 6172 false negatives and 4479 false positives. The best threshold and best scores are shown in **Table 3** below for all the three datasets.

Table 3 The confusion matrix and balanced_accuracy_score at different thresholds

Baby_products Dataset

Threshold	Confusion matrix	balanced_accuray_score
0.5 (Default Threshold)	[[14110 6052] [4269 23768]] ----------------- [['TN' 'FP'] ['FN' 'TP']]	0.774
0.58 (Best Threshold)	[[15683 4479] [6172 21865]]	0.779

Grocery_foods Dataset

Threshold	Confusion matrix	balanced_accuray_score
0.5 (Default Threshold)	[[13452 5776] [4012 22077]]	0.773
0.56 (Best Threshold)	[[14488 4740] [5231 20858]]	0.776

Cell_phones Dataset

Threshold	Confusion matrix	balanced_accuray_score
0.5 (Default Threshold)	[[18222 7318] [5863 26760]]	0.767
0.54 (Best Threshold)	[[19207 6333] [7010 25613]]	0.769

Business Case 2: Customer Satisfaction

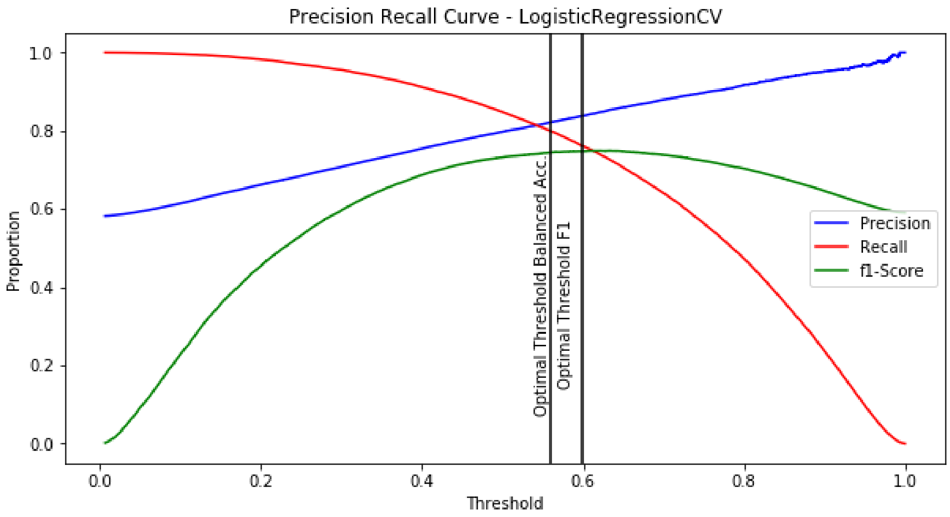
In this business scenario, we are most interested in uncovering negative reviews for the purpose of better understanding how we can improve our product, or perhaps for targeting specific customers that are unsatisfied - perhaps to reach out to them with special offers. As such, we want to choose a metric that is focused on correctly classifying these negative reviews.
To measure this I will use f1-score because it is a better measure when focusing on a specific class - in this case low ratings - to ensure we attain good recall and precision.

For baby_products dataset, the optimal threshold for "low" ratings is at 0.63. It outputs an f1-Score of 0.749. For grocery_foods dataset, the optimal threshold for this case is 0.599 with f1_score of 0.747. For cell_phones dataset, the optimal threshold for this case is 0.598 with f1_score of 0.745.

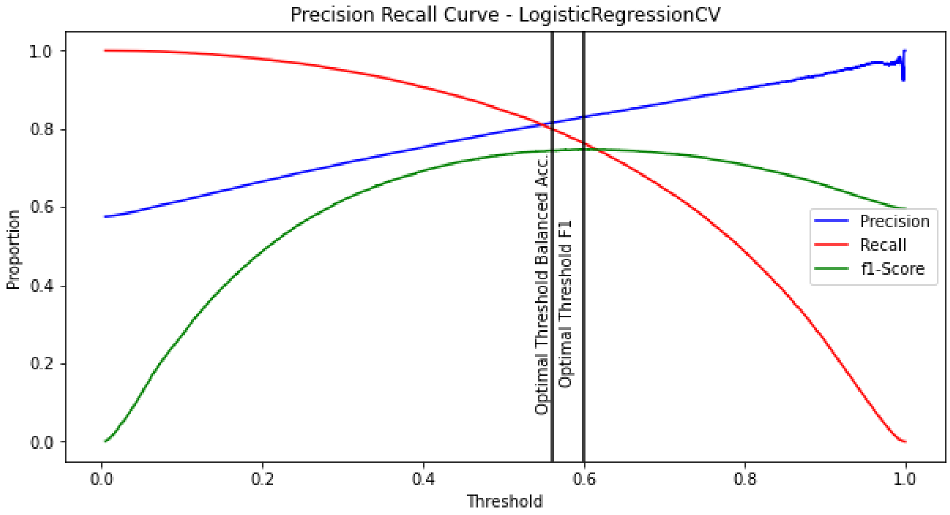
The precision, recall and f1 curves with thresholding for low ratings for all the three datasets are shown in the **Figure 10** below.

Figure 10 Precision, Recall and f1 Curves with Thresholding for "low" ratings

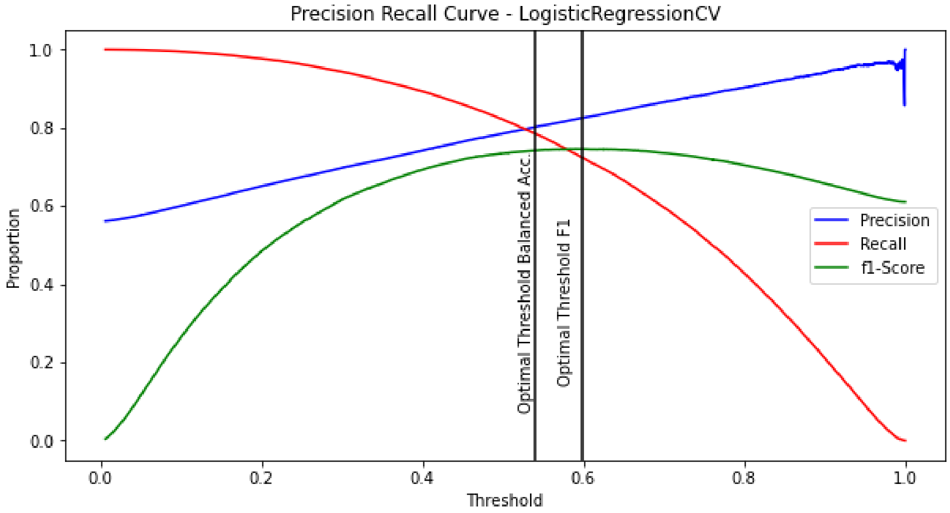
Baby_products Dataset



Grocery_foods Dataset



Cell_phones Dataset



7. Summary

The best model for sentiment classification of Amazon product reviews in all the three different datasets was Logistic Regression classification. The text preprocessing steps which led to the best mode were removing URL, removing numbers and punctuations, tokenization, removing stopwords, and lemmatization. Document terms matrix from the TfidfVectorizer were used for the algorithms.

The Logistic Regression classifier (with C=0.85) was applied to the test data of the three datasets. For baby_products dataset, the model correctly classified 23768 reviews as positive or high rated category and 14110 reviews as negative or low rated category. Another 4269 were misclassified as false negative and 6052 as false positive. Grocery_foods dataset and cell_phones dataset showed similar trends in false positive and false negative values which was better than MultinomialNB and Random Forest Classifier.

To further explore ways to improve the classifier, I did thresholding for different business situations such as product market comparisons and customer satisfaction. The metrics used were the balanced accuracy and F1 score.

For the Grocery_foods and Cell_phones dataset I had increased parameter min_df (minimum document frequency) to 50 because of the computational limitation of my PC but in the future with more processing power we can use the best min_df value and also can explore more powerful models. We can apply this model to classify the sentiment of other text datasets such as twitter, facebook etc.

8. Sources

- **Json Brownlee 2017.** How to Encode Text Data for Machine Learning with scikit-learn link.
- **Json Brownlee 2020**. A Gentle Introduction to Threshold-Moving for Imbalanced Classification link.
- **Ujjawal Verma 2020**. Text Preprocessing for NLP (Natural Language Processing),Beginners to Master link.
- Xing Fang et. al., Journal of Big Data (2015), Sentiment analysis using product review data.