Sentiment Analysis of Customer Reviews

Kapil Bhise 181080017
Prathmesh Chandwade 181080018
Tejas Bhatambarekar 181080016

Contents

Introduction

- Product reviews are becoming more important with the evolution of traditional brick and mortar retail stores to online shopping. Consumers are posting reviews directly on product pages in real time.
- With the vast amount of consumer reviews, this creates an opportunity to see how the market reacts to a specific product.
- We will be attempting to see if we can predict the sentiment of a product review using machine learning tools, particularly the Support Vector Machine.
- The dataset used is from www.kaggle.com.
- This project was completed using Jupyter Notebook and Python with Pandas, NumPy, Matplotlib, Sci-kit learn.

Background

- A customer review is a review of a product by a customer who has purchased and used the product. The reviews may themselves be graded for usefulness or accuracy by other users.
- Sentiment analysis is the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.
- Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media.

Purpose

- Analysing the sentiments of product reviews
- Analyse which products should be kept and which should be dropped from Amazon’s product catalog
- Try to predict scores for reviews based on certain words and assign them positive or negative sentiment

Dataset

- The dataset is provided at:

  https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products

- This is a list of over 34,000 consumer reviews for Amazon products like the Kindle, Fire TV Stick, and more provided by Datafiniti's Product Database.
- The dataset includes basic product information, rating, review text, and more for each product.

Methodology

1. Importing the raw data
2. Split into train/test
3. Data exploration
4. Correlations
5. Sentiment Analysis

Quick Look at the Raw Data

	id	name	asins	brand	categories	keys	manufacturer	reviews.text	reviews.title	reviews.userCity	reviews.userProvince
0	AVqkIhwDv8e3D1O-lebb	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	B01AHB9CN2	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	841667104676,amazon/53004484,amazon/b01ahb9cn2...	Amazon	This product so far has not disappointed. My c...	Kindle	NaN	NaN
1	AVqkIhwDv8e3D1O-lebb	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	B01AHB9CN2	Amazon	Electronics,iPad & Tablets,All Tablets,Fire Ta...	841667104676,amazon/53004484,amazon/b01ahb9cn2...	Amazon	great for beginner or experienced person. Boug...	very fast	NaN	NaN

2 rows × 21 columns

A look at raw data

- As reviews are in reviews.text column, our sentiment analysis will be based on reviews.text column.
  - We labeled each review based on each sentiment
    - title can contain positive/negative information about review
- We dropped reviews.userCity, reviews.userProvince, reviews.id, and reviews.didPurchase since these values are floats (for exploratory analysis only)
- Among all columns, Reviews.text column had minimum missing data.
- There were about 7000 missing values in name column. So, we cleaned up the name column by referencing asins.

Split into Train/Test

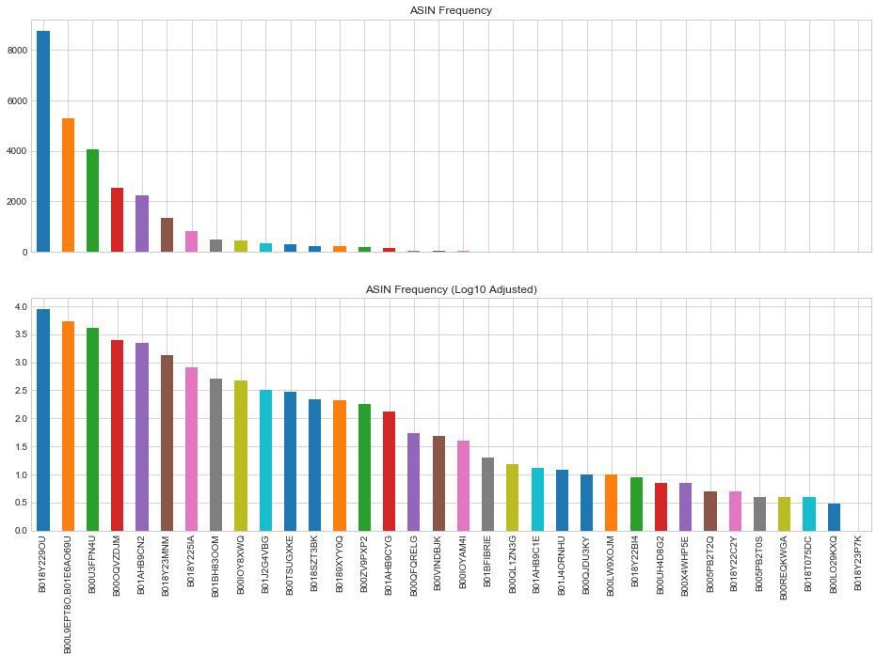
- Our goal is to eventually train a sentiment analysis classifier.
- Since the majority of reviews are positive (5 stars), we will need to do a stratified split on the reviews score to ensure that we don't train the classifier on imbalanced data
- To use sklearn's *Stratified ShuffleSplit* class, we removed all samples that have NaN in review score, then convert all review scores to *integer* datatype

Data Exploration

- We used regular expressions to clean out any unfavorable characters in our dataset, and then preview what the data looks like after the cleaning.
- We explored the following columns to get more information about data:
  - asins - Amazon’s product-type ID
  - name - names of the items
  - reviews.rating - rating by customer associated with the review
  - reviews.doRecommend

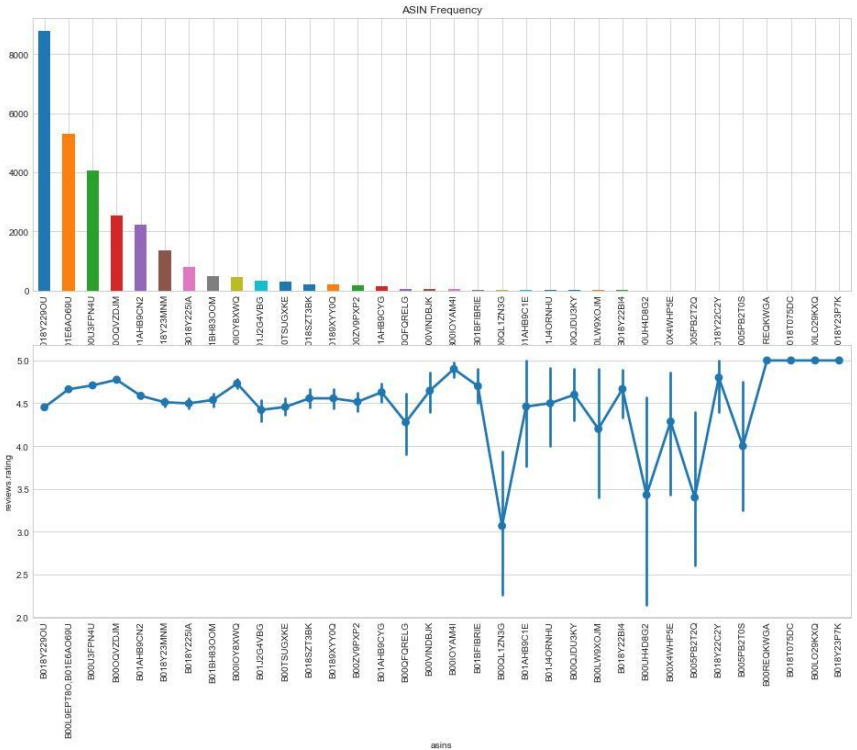
Data Exploration

- Based on the bar graph for ASINs, we see that certain products have significantly more reviews than other products, which may indicate a higher sale in those specific products
- Certain ASINs have better sales, while other ASINs have lower sale, and so we can determine which products should be kept or dropped.



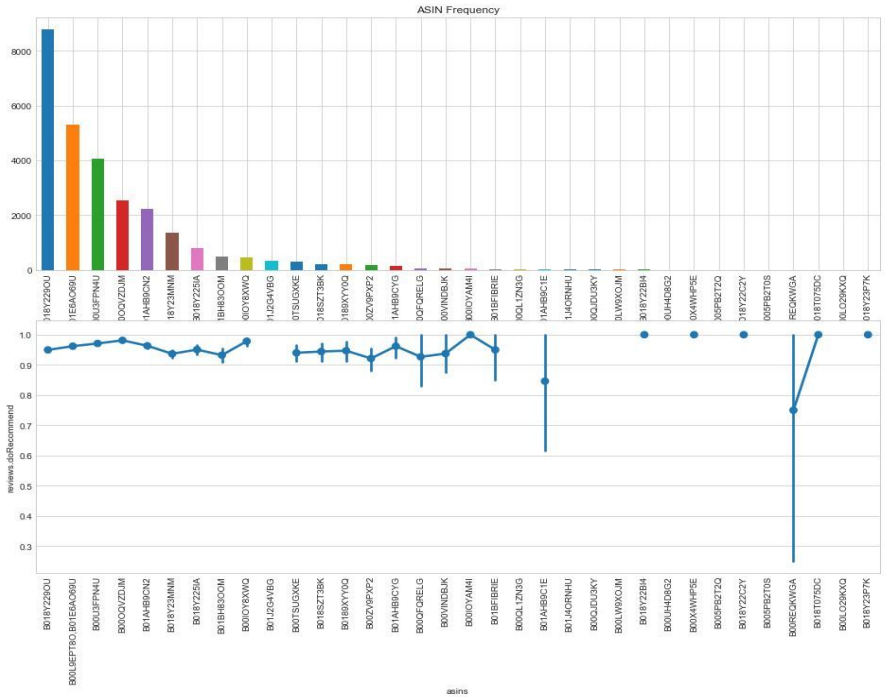
Data Exploration

- The most frequently reviewed products have their average review ratings in the 4.5 – 4.8 range, with little variance.
- For ASINs with lower frequencies, corresponding average review ratings have significantly higher variance. So, average review ratings for ASINs with lower frequencies are not significant for our analysis.



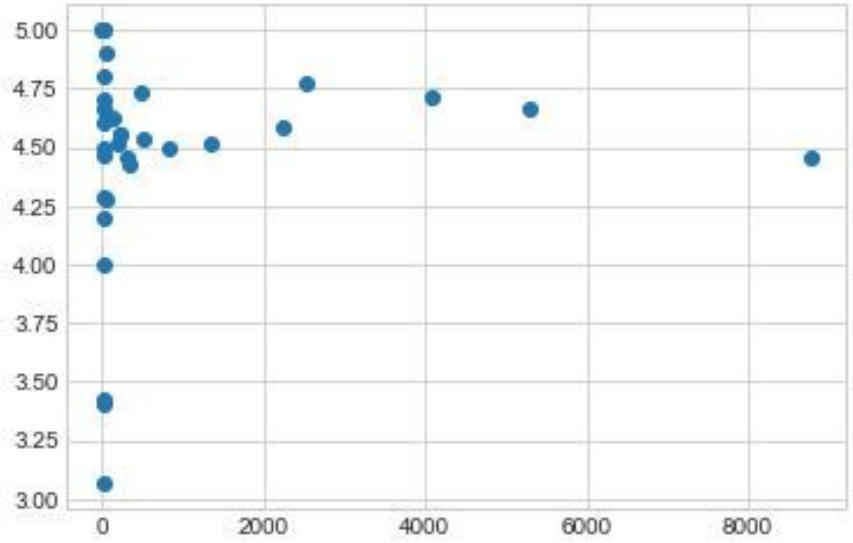
Data Exploration

- The first 19 ASINs show that consumers recommend the product, which is consistent with the “reviews.rating / ASINs” analysis, where the first 19 ASINs have good ratings between 4.0 to 5.0 .
- The remaining ASINs have fluctuating results due to lower sample size, which should not be considered.



Correlations

- From analysis in data exploration between ASINs and reviews.rating, we discovered that there are many ASINs with low occurrence that have high variances, so we concluded that these low occurrence ASINS are not significant in our analysis.
- In correlation analysis between ASINs and reviews.rating, there is almost no correlation which is consistent with our findings.



Sentiment Analysis

Set Target Variable (Sentiments)

```
def sentiments(rating):
    if (rating == 5) or (rating == 4):
        return "Positive"
    elif rating == 3:
        return "Neutral"
    elif (rating == 2) or (rating == 1):
        return "Negative"
# Add sentiments to the data
strat_train["Sentiment"] = strat_train["reviews.rating"].apply(sentiments)
strat_test["Sentiment"] = strat_test["reviews.rating"].apply(sentiments)
strat_train["Sentiment"][:20]
```

Sentiment Analysis

```
# Prepare data
X_train = strat_train["reviews.text"]
X_train_targetSentiment = strat_train["Sentiment"]
X_test = strat_test["reviews.text"]t
X_test_targetSentiment = strat_test["Sentiment"]
print(len(X_train), len(X_test))
```

```
Output:
27701 6926
```

27,701 training samples and 6926 testing samples.

Sentiment Analysis - Feature Extraction

- Here we will turn content into numerical feature vectors using the Bag of Words strategy:
- In order to implement the Bag of Words strategy, we will use SciKit-Learn's CountVectorizer.
- Text preprocessing:
    - Tokenization
    - Stopwords
- Occurrence counting
- Feature Vector
- We will use TfidfTransformer to reduce redundancy.
- Term Frequencies (Tf) divides number of occurrences for each word by total number of words
- Term Frequencies times Inverse Document Frequency (Tfidf) downscales the weights of each word

Sentiment Analysis - Feature Extraction

```
# Replace "nan" with space
X_train = X_train.fillna(' ')
X_test = X_test.fillna(' ')
X_train_targetSentiment = X_train_targetSentiment.fillna(' ')
X_test_targetSentiment = X_test_targetSentiment.fillna(' ')

# Text preprocessing and occurrence counting
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
X_train_counts.shape
```

```
Output:
(27701, 12526)
```

Here we have 27,701 training samples and 12,526 distinct words in our training sample.

Sentiment Analysis - Building a Pipeline from the Extracted Features

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
clf_multiNB_pipe = Pipeline([("vect", CountVectorizer()),
                             ("tfidf", TfidfTransformer()),
                             ("clf_nominalNB", MultinomialNB())])
clf_multiNB_pipe.fit(X_train, X_train_targetSentiment)
```

```
import numpy as np
predictedMultiNB = clf_multiNB_pipe.predict(X_test)
np.mean(predictedMultiNB == X_test_targetSentiment)
```

Output:
0.9344498989315623

Sentiment Analysis

Logistic Regression Classifier

```
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
clf_logReg_pipe = Pipeline([("vect", CountVectorizer()),
                            ("tfidf", TfidfTransformer()),
                            ("clf_logReg", LogisticRegression())])
clf_logReg_pipe.fit(X_train, X_train_targetSentiment)

import numpy as np
predictedLogReg = clf_logReg_pipe.predict(X_test)
np.mean(predictedLogReg == X_test_targetSentiment)
```

Output:
0.937048801617095

Sentiment Analysis

Support Vector Machine Classifier

```
from sklearn.svm import LinearSVC
clf_linearSVC_pipe = Pipeline([("vect", CountVectorizer()),
                               ("tfidf", TfidfTransformer()),
                               ("clf_linearSVC", LinearSVC())])
clf_linearSVC_pipe.fit(X_train, X_train_targetSentiment)

predictedLinearSVC = clf_linearSVC_pipe.predict(X_test)
np.mean(predictedLinearSVC == X_test_targetSentiment)
```

```
Output:
0.9393589373375686
```

Sentiment Analysis

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
clf_decisionTree_pipe = Pipeline([("vect", CountVectorizer()),
                                  ("tfidf", TfidfTransformer()),
                                  ("clf_decisionTree", DecisionTreeClassifier())
                                 ])
clf_decisionTree_pipe.fit(X_train, X_train_targetSentiment)

predictedDecisionTree = clf_decisionTree_pipe.predict(X_test)
np.mean(predictedDecisionTree == X_test_targetSentiment)
```

Output:
0.9018192318798729

Sentiment Analysis

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
clf_randomForest_pipe = Pipeline([("vect", CountVectorizer()),
                                  ("tfidf", TfidfTransformer()),
                                  ("clf_randomForest", RandomForestClassifier())
                                 ])
clf_randomForest_pipe.fit(X_train, X_train_targetSentiment)

predictedRandomForest = clf_randomForest_pipe.predict(X_test)
np.mean(predictedRandomForest == X_test_targetSentiment)
```

Output:
0.9345942824140918

Sentiment Analysis

- We have used Multinominal Naive Bayes, Logistic Regression, Support Vector Machine, Random Forest classifiers to build and test models.
- All the models performed very well (>90%), and we will use the Support Vector Machine Classifier since it has the highest accuracy level at 93.94%.

Sentiment Analysis - Output

```
Positive :-> Had a gen 1 Kindle and so glad I moved up to this. Great for outdoor sunny reading. A must have !
Positive :-> I love it. Especially for the price.................
Positive :-> Very functional, easy set-up and great so be quality.
Positive :-> Bought this tablet for my daughter so far so good!
Positive :-> We bought the tablet for work, but the only app we needed (free) never would download. After many tries and seve
ral hours, we gave up and took it back to the store.
Positive :-> So far this tablet has been efficient. It kinda bad is difficult to use and it seems to require a lot of steps t
o get the kids profiles set up and it actually does not do everything I hoped it would.
Positive :-> Bought this tablet to replace a digit and one that was on recall. The Amazon ecosystem is nice and the tablet se
ems to work pretty well. I like the options for limiting screen time especially being able to get nature that reading goals a
re met FIRST. Unit needs more on board memory however. Make sure to buy an sd card to expand.
Positive :-> Love the sound quality and ease of use and setup. It's very informative and helpful
Positive :-> Very good access to multiple venues. Cheap and easy to use.
Positive :-> Bought 3 of these for the family and it's been awesome.
Positive :-> I love this it was the best thing I brought My kids even love it and it easy for or them to use
Positive :-> I find the Echo to be an amazing speaker enabled by saying "Alexa" to access all kinds of information and receiv
e answers to many, many questions. It is constantly being upgraded with new capabilities and abilities.
Positive :-> Got this to replace the Samsung tab A 7". That Samsung tab lagged a lot but I haven't seemed to have as much pro
blems on the fire tab, the only downside is that the battery dies fast.
```

Conclusion

By implementing this project, we have learnt about sentiment analysis, different models used for the same and most importantly, we came across various steps included in the domain of data mining and data warehousing like data exploration, data preprocessing etc.

Future Scope

Some future works which can be included to improve the model and also to make it more effective in practical cases:

- The model can be incorporate with programs that can interact with customer seeking a score of a particular product.
- As we used a large scale dataset we can apply the model on local market sites to get better accuracy and usability.

References

1. https://www.researchgate.net/publication/344869545_Sentiment_Analysis_on_Amazon_reviews
2. https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products
3. https://towardsdatascience.com/sentiment-analysis-on-amazon-reviews45cd169447ac

Thank You!