DMDW Project Report
Sentiment Analysis

Group no: 7

Team Members:
1. Prathmesh Chandwade (181080018)
2. Kapil Bhise (181080017)
3. Tejas Bhatambarekar (181080016)

Guided By:
Dr. S G Bhirud

Table of contents

Abstract

The world we see nowadays is becoming more digitalized. In this digitalized world e-commerce is taking the ascendancy by making products available within the reach of customers where the customer doesn't have to go out of their house. As now a day's people are relying on online products so the importance of a review is increasing. For selecting a product, a customer needs to go through thousands of reviews to understand a product. But in this prospering day of machine learning, going through thousands of reviews would be much easier if a model is used to polarize those reviews and learn from it. We used a supervised learning method on a large scale amazon dataset to polarize it and get satisfactory accuracy.

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 we will be using 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 or service made by a customer who has purchased and used, or had experience with, the product or service. Customer reviews are a form of customer feedback on electronic commerce and online shopping sites. The reviews may themselves be graded for usefulness or accuracy by other users.

Sentiment analysis (also known as opinion mining or emotion AI) 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, and healthcare materials for applications that range from marketing to customer service to clinical medicine.

Purpose

The goal of our project is to analyze polarization of the customer reviews using machine learning models and accordingly, help customers to get an insight of specific products on an e-commerce site. This will help customers to decide on the product and it’s qualities.

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

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import math
import warnings
warnings.filterwarnings('ignore') # Hides warning
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore",category=UserWarning)
sns.set_style("whitegrid") # Plotting style
%matplotlib inline # Plots show up in notebook
np.random.seed(7) # seeding random number generator

csv = "1429_1.csv"
df = pd.read_csv(csv)
df.head(2)
```

Output:

	id	name	asins	brand	categories	keys	manufacturer
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
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

2 rows × 21 columns

reviews.text	reviews.title	reviews.userCity	reviews.userProvince
This product so far has not disappointed. My c...	Kindle	NaN	NaN
great for beginner or experienced person. Boug...	very fast	NaN	NaN

```
data = df.copy()
data.describe()
```

Output:

	reviews.id	reviews.numHelpful	reviews.rating	reviews.userCity	reviews.userProvince
count	1.0	34131.000000	34627.000000	0.0	0.0
mean	111372787.0	0.630248	4.584573	NaN	NaN
std	NaN	13.215775	0.735653	NaN	NaN
min	111372787.0	0.000000	1.000000	NaN	NaN
25%	111372787.0	0.000000	4.000000	NaN	NaN
50%	111372787.0	0.000000	5.000000	NaN	NaN
75%	111372787.0	0.000000	5.000000	NaN	NaN
max	111372787.0	814.000000	5.000000	NaN	NaN

```
data["asins"].unique()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34660 entries, 0 to 34659
Data columns (total 21 columns):
id                      34660 non-null object
name                    27900 non-null object
asins                   34658 non-null object
brand                   34660 non-null object
categories              34660 non-null object
keys                    34660 non-null object
manufacturer            34660 non-null object
reviews.date            34621 non-null object
reviews.dateAdded       24039 non-null object
reviews.dateSeen        34660 non-null object
reviews.didPurchase     1 non-null object
reviews.doRecommend     34066 non-null object
reviews.id              1 non-null float64
reviews.numHelpful      34131 non-null float64
reviews.rating          34627 non-null float64
reviews.sourceURLs      34660 non-null object
reviews.text            34659 non-null object
reviews.title           34655 non-null object
reviews.userCity        0 non-null float64
reviews.userProvince    0 non-null float64
reviews.username        34658 non-null object
dtypes: float64(5), object(16)
memory usage: 5.6+ MB
```

```
data["asins"].unique()
```

Output:

```
array(['B01AHB9CN2', 'B00VINDBJK', 'B005PB2T0S', 'B002Y27P3M',
       'B01AHB9CYG', 'B01AHB9C1E', 'B01J2G4VBG', 'B00ZV9PXP2',
       'B0083Q04TA', 'B018Y229OU', 'B00REQKWGA', 'B00IOYAM4I',
       'B018T075DC', nan, 'B00DU15MU4', 'B018Y225IA', 'B005PB2T2Q',
       'B018Y23MNM', 'B00OQVZDJM', 'B00IOY8XWQ', 'B00LO29KXQ',
       'B00QJDU3KY', 'B018Y22C2Y', 'B01BFIBRIE', 'B01J4ORNHU',
       'B018SZT3BK', 'B00UH4D8G2', 'B018Y22BI4', 'B00TSUGXKE',
       'B00L9EPT8O,B01E6AO69U', 'B018Y23P7K', 'B00X4WHP5E', 'B00QFQRELG',
       'B00LW9XOJM', 'B00QL1ZN3G', 'B0189XYY0Q', 'B01BH83OOM',
       'B00BFJAHF8', 'B00U3FPN4U', 'B002Y27P6Y', 'B006GWO5NE',
       'B006GWO5WK'], dtype=object)
```

```
asins_unique = len(data["asins"].unique())
print("Number of Unique ASINs: " + str(asins_unique))
```

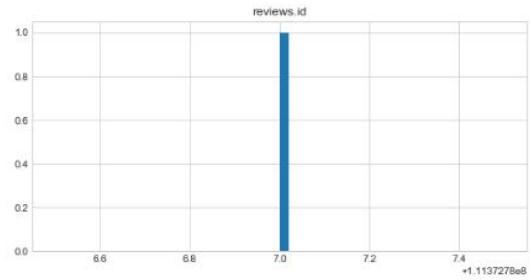
Output:

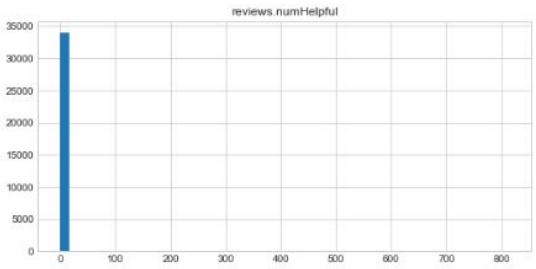
Number of Unique ASINs: 42

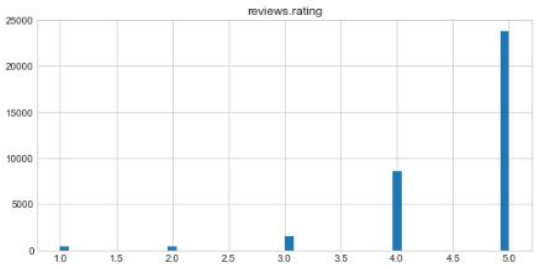
Visualizing the distributions of numerical variables:

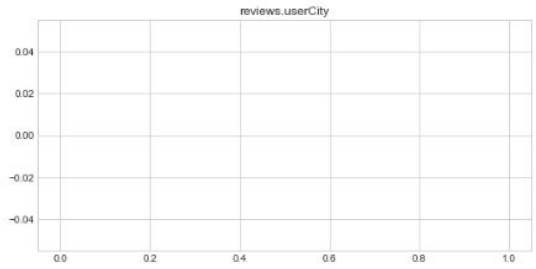
```
# Builds histogram and set the number of bins and fig size (width, height)
data.hist(bins=50, figsize=(20,15))
plt.show()
```

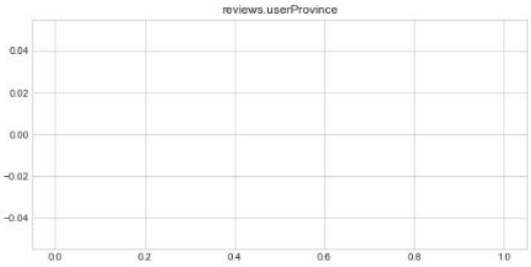
Output:











2. Split into Train/Test

```
from sklearn.model_selection import StratifiedShuffleSplit
print("Before {}".format(len(data)))
dataAfter = data.dropna(subset=["reviews.rating"])
# Removes all NAN in reviews.rating
print("After {}".format(len(dataAfter)))
dataAfter["reviews.rating"] = dataAfter["reviews.rating"].astype(int)

Output:
Before 34660
After 34627


split = StratifiedShuffleSplit(n_splits=5, test_size=0.2)
for train_index, test_index in split.split(dataAfter,
                                           dataAfter["reviews.rating"]):
    strat_train = dataAfter.reindex(train_index)
```

```
    strat_test = dataAfter.reindex(test_index)
```

#Check to see if train/test sets were stratified proportionately in comparison to raw data.

```
len(strat_train)
```

Output:

27701

```
strat_train["reviews.rating"].value_counts()/len(strat_train)
```

Output:

5.0 0.685174
4.0 0.247031
3.0 0.043500
2.0 0.011696
1.0 0.011588
Name: reviews.rating, dtype: float64

```
len(strat_test)
```

Output:

6926

```
strat_test["reviews.rating"].value_counts()/len(strat_test)
```

Output:

5.0 0.689864
4.0 0.244730
3.0 0.042160
1.0 0.011406
2.0 0.011118
Name: reviews.rating, dtype: float64

3. Data Exploration (Training Set)

```
reviews = strat_train.copy()
reviews.head(2)
```

Output:

	id	name	asins	brand	categories	keys	manufacturer
4349	AVphgVaX1cnluZ0-DR74	Fire Tablet, 7 Display, Wi-Fi, 8 GB - Includes...	B018Y229OU	Amazon	Fire Tablets,Tablets,Computers & Tablets,All T...	firetablet7displaywifi8gbincludesspecialoffers...	Amazon
30776	AV1YE_muvKc47QAVgpwE	NaN	B00U3FPN4U	Amazon Fire Tv	Back To College,College Electronics,College Tv...	848719057492,amazonfiretv/51454342,amazonfiret...	Amazon

2 rows × 21 columns

reviews.text	reviews.title	reviews.userCity	reviews.userProvince
we bought this for my 11 year old daughter and...	great for all ages	NaN	NaN
I have the Roku 4, and new Apple TV, this stre...	Great streaming box	NaN	NaN

3.1 names / ASINs

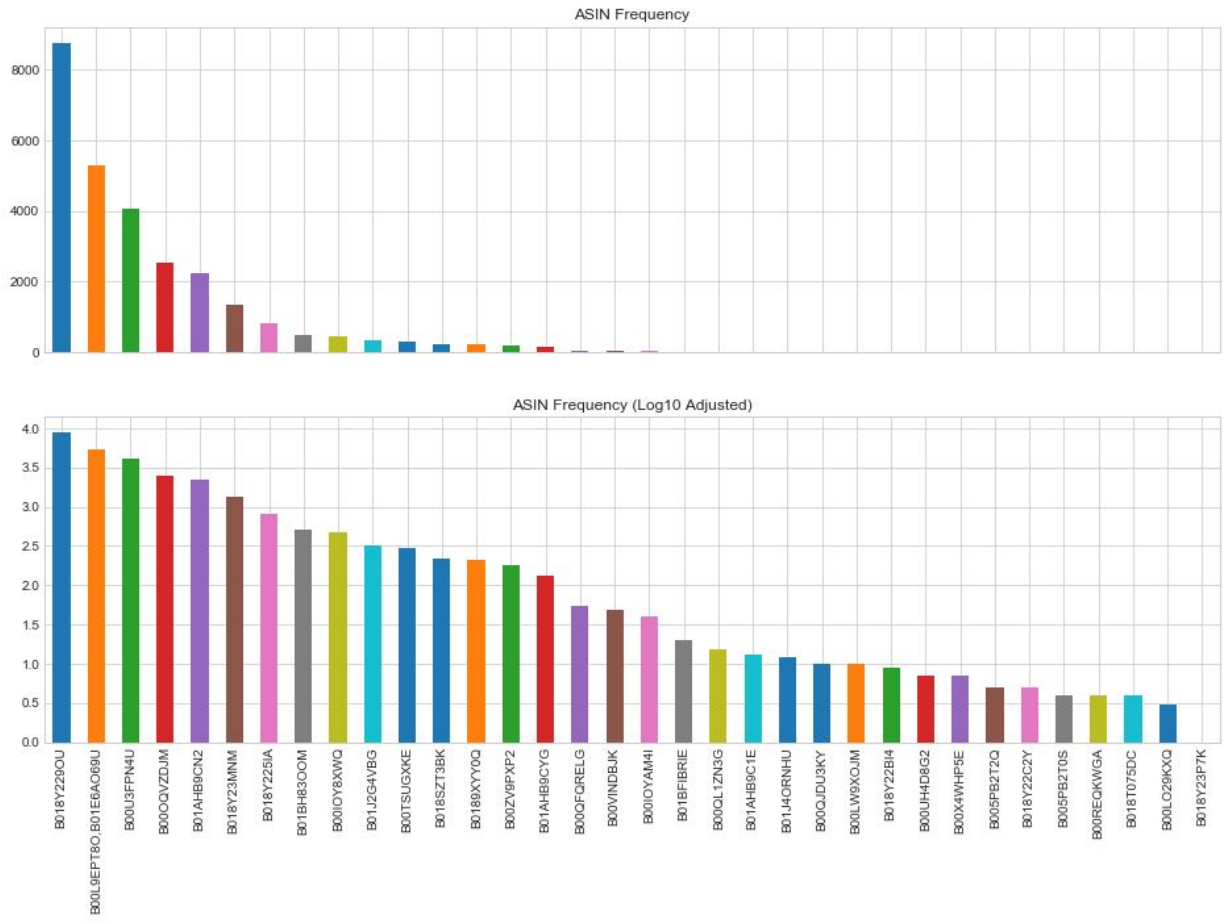
```
len(reviews["name"].unique()), len(reviews["asins"].unique())
```

Output:
(47, 35)

```
reviews.info()
fig = plt.figure(figsize=(16,10))
```

```
ax1 = plt.subplot(211)
ax2 = plt.subplot(212, sharex = ax1)
reviews["asins"].value_counts().plot(kind="bar", ax=ax1, title="ASIN
Frequency")
np.log10(reviews["asins"].value_counts()).plot(kind="bar", ax=ax2,
                                               title="ASIN Frequency
                                               (Log10 Adjusted)")
plt.show()
```

Output:




```
# Entire training dataset average rating
reviews["reviews.rating"].mean()
```

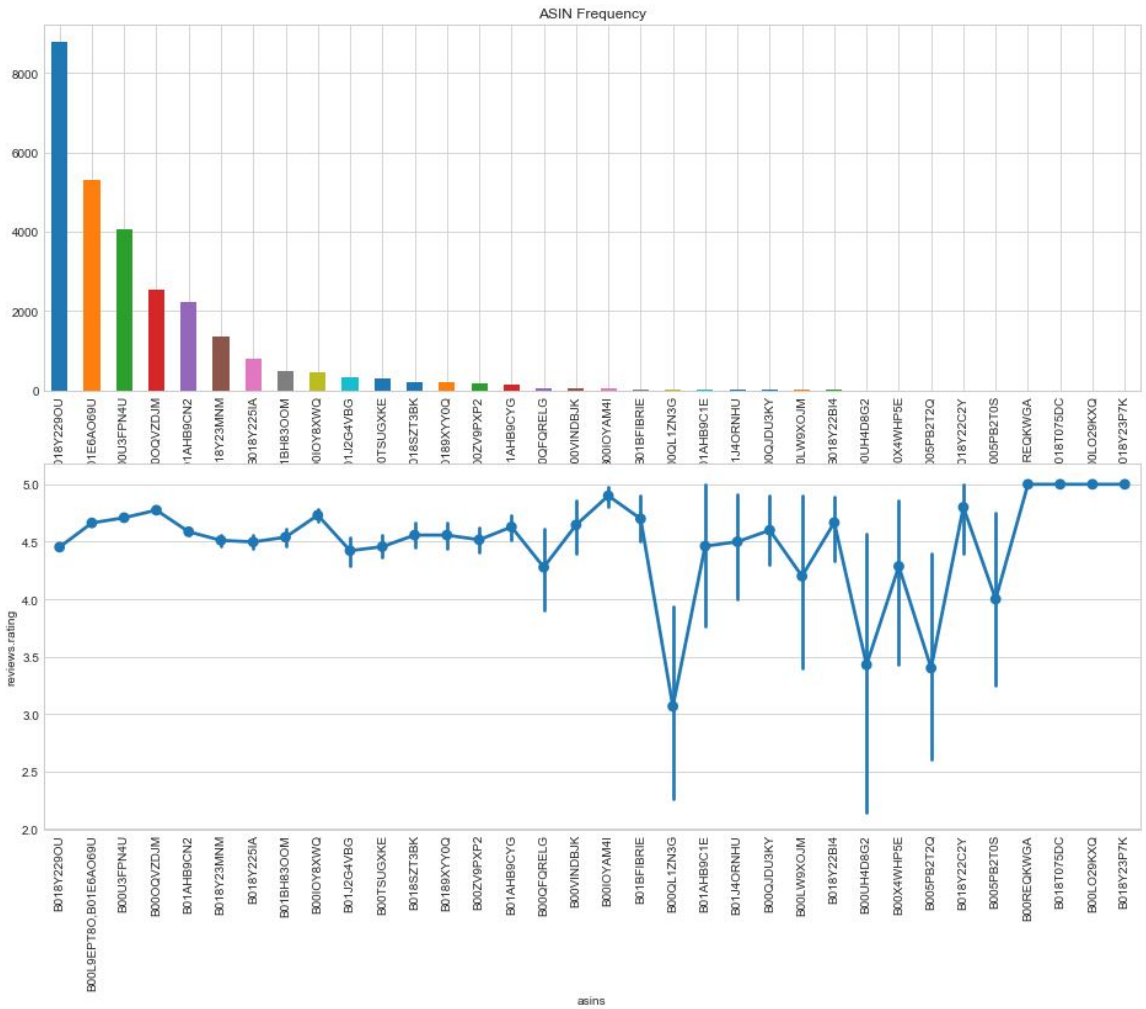
Output:

4.5841072525566435

3.2 reviews.rating / ASINs

```
asins_count_ix = reviews["asins"].value_counts().index
plt.subplots(2,1,figsize=(16,12))
plt.subplot(2,1,1)
reviews["asins"].value_counts().plot(kind="bar", title="ASIN Frequency")
plt.subplot(2,1,2)
sns.pointplot(x="asins", y="reviews.rating", order=asins_count_ix,
data=reviews)
plt.xticks(rotation=90)
plt.show()
```

Output:

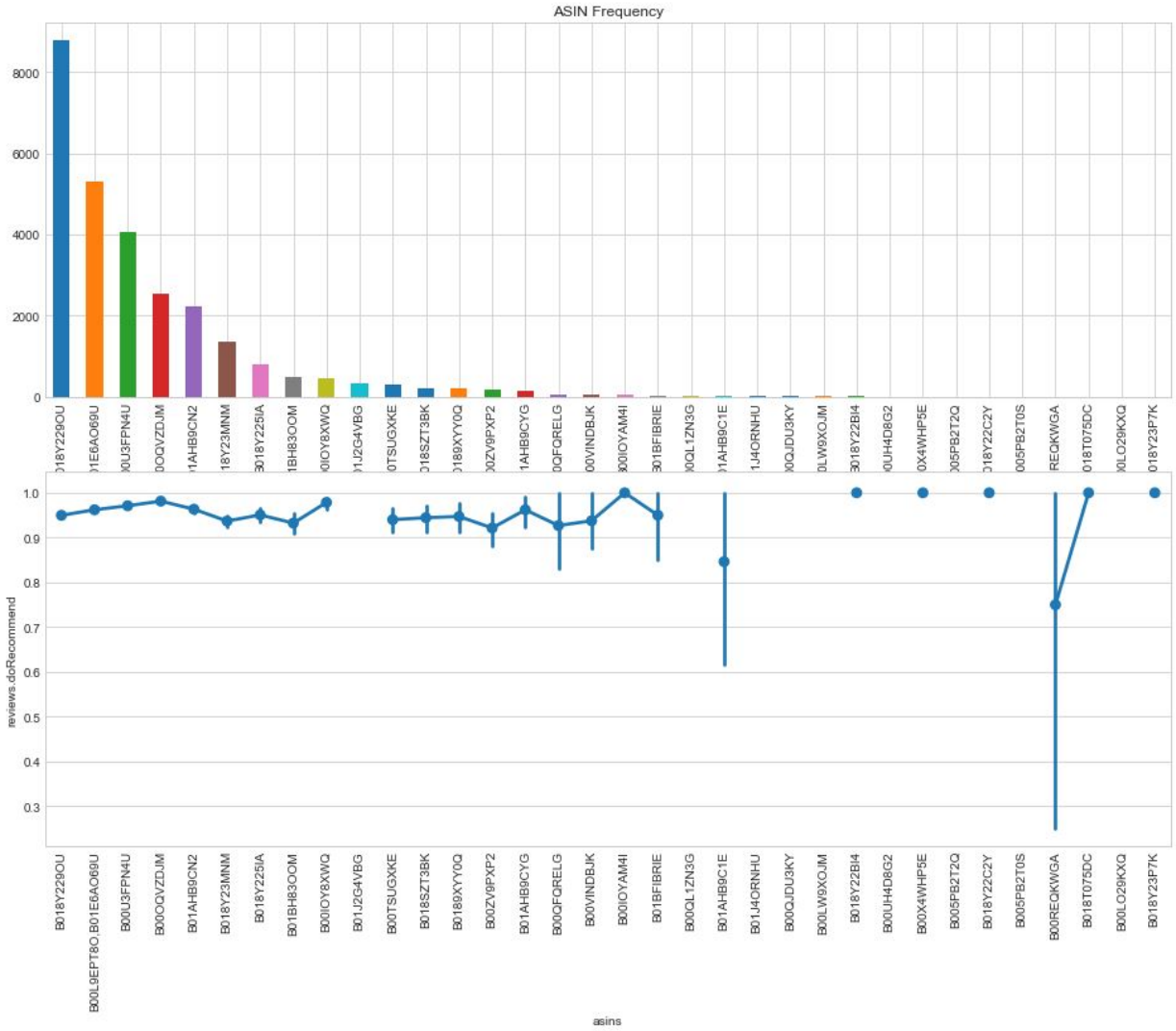


3.3 reviews.doRecommend / ASINs

```
asins_count_ix = reviews["asins"].value_counts().index
plt.subplots(2,1,figsize=(16,12))
plt.subplot(2,1,1)
reviews["asins"].value_counts().plot(kind="bar", title="ASIN Frequency")
plt.subplot(2,1,2)
sns.pointplot(x="asins", y="reviews.rating", order=asins_count_ix,
data=reviews)
plt.xticks(rotation=90)

plt.show()plt.subplots (2,1,figsize=(16,12))
plt.subplot(2,1,1)
reviews["asins"].value_counts().plot(kind="bar", title="ASIN Frequency")
plt.subplot(2,1,2)
sns.pointplot(x="asins", y="reviews.doRecommend", order=asins_count_ix,
             data=reviews)
plt.xticks(rotation=90)
plt.show()
```

Output:



4. Correlations

```
corr_matrix = reviews.corr()
corr_matrix
# Here we can analyze reviews.ratings with asins
```

Output:

	reviews.id	reviews.numHelpful	reviews.rating	reviews.userCity	reviews.userProvince
reviews.id	NaN	NaN	NaN	NaN	NaN
reviews.numHelpful	NaN	1.00000	-0.04372	NaN	NaN
reviews.rating	NaN	-0.04372	1.00000	NaN	NaN
reviews.userCity	NaN	NaN	NaN	NaN	NaN
reviews.userProvince	NaN	NaN	NaN	NaN	NaN

5. Sentiment Analysis

Using the features in place, we will build a classifier that can determine a review's sentiment.

5.1 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]
```

Output:

```
4349     Positive
30776    Positive
28775     Neutral
1136     Positive
17803    Positive
7336     Positive
32638    Positive
13995    Positive
6728     Negative
22009    Positive
11047    Positive
22754    Positive
5578     Positive
11673    Positive
19168    Positive
14903    Positive
30843    Positive
5440     Positive
28940    Positive
31258    Positive
Name: Sentiment, dtype: object
```

```
# 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.

5.2 Extract Features

Here we will turn content into numerical feature vectors using the Bag of Words strategy:

```
# 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.

```
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer(use_idf=False)
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_train_tfidf.shape
```

Output:
(27701, 12526)

5.3 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)
```

```
Pipeline(memory=None,
     steps=[('vect', CountVectorizer(analyzer='word', binary=False,
        decode_error='strict', dtype=<class 'numpy.int64'>,
        encoding='utf-8', input='content', lowercase=True, max_df=1.0,
        max_features=None, min_df=1, ngram_range=(1, 1),
        preprocessor=None, stop_words=None, strip...alse, use_idf=True)),
        ('clf_nominalNB', MultinomialNB(alpha=1.0, class_prior=None,
         fit_prior=True))])
```

5.4 Test Model

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

Output:

0.9344498989315623

Here we see that our Multinominal Naive Bayes Classifier has a 93.45% accuracy level based on the features.

5.5 Testing Other Models

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

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

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

Random Forest Classifier

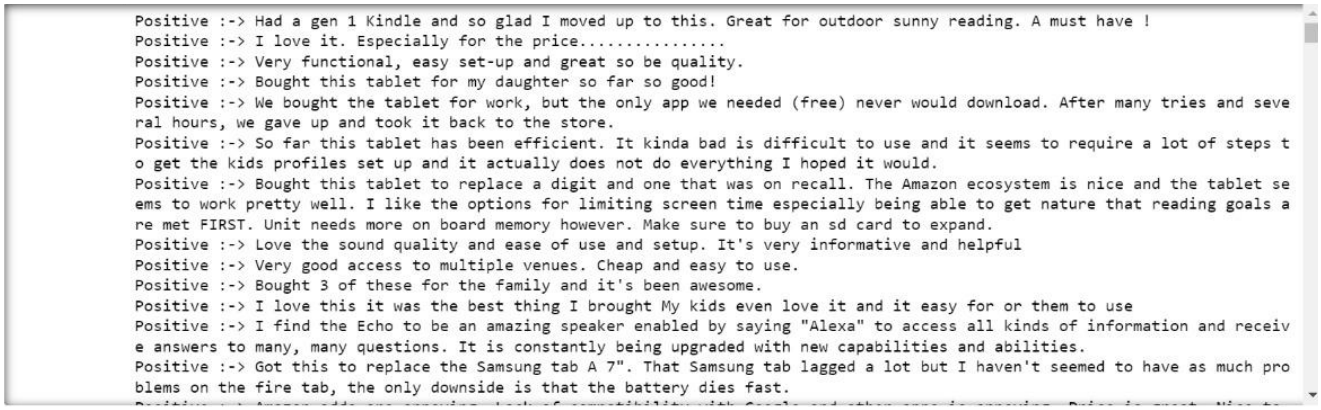
```
from sklearn.ensemble import RandomForestClassifier
clf_randomForest_pipe = Pipeline([("vect", CountVectorizer()),
                                  ("tfidf", TfidfTransformer()),
                                  ("clf_randomForest", RandomForestClassifier())
                                 ])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

Looks like 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%.

Prediction(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-reviews-45cd169447ac