Project Report
Title: Amazon Review Sentiment Analysis

Author: Ruchita Jadhav

Overview:

The world of Ecommerce has millions of users each day. With an increased number of users, there is always a requirement to understand the mindset of people. WIth no direct contact, views and reviews are the only way companies capture user information. Sometimes ad-clicks play an important role as well. However reviews cannot be substituted with any other form of data capture. One particularly effective approach in recent years has been to build personalized recommendation engines into their platform or interface. Determining the specific topics and sentiments associated with given sports and outdoors products is essential in building a recommendation engine. This project is mainly to understand the concepts covered in class and apply them to a specific domain.

Tools and Models:

Dataset: Amazon Review Data, (~12 GB)
Tools: Python, MatplotLib, Scikit-Learn (Countvectorizer, NGram), NLTK
Models: Logistic Regression with feature modeling, Topic Modelling on the review data set

Goals and Problems:

1. Goal 1: Data Capture
2. Goal 2: Pre-process the data
3. Goal 3: Modeling: To understand the relationship between reviews and sentiment
4. Goal 4: Interpretation and Analysis

Dataset

Amazon Review Data, (~12 GB)

- The data set we are using is the Amazon Reviews Dataset. This data set is hosted on https://s3.amazonaws.com/amazon-reviews-pds/readme.html
- There are multiple datasets targeting specific products, but we will be looking into the holistic dataset which covers a wide range of products and details pertaining to them
- To be more specific we will be using the data set named - .https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_multilingual_US_v1_00.tsv.gz

Steps to Reproduce:
Open the link, click open in playground, and run all

Google Colab Link - https://colab.research.google.com/drive/1rWLDGCaJU2zvb8p-fy0J4stD5IkeRlTE

Data Exploration

Data Exploration consists of the following pipelines:

1. Data Ingestion
2. Data Description
3. Data Cleaning
4. Data Augmentation
5. Data Visualization
6. Data Aggregation and Results
7. Data Persistence
8. Sentiment Analysis

Data Ingestion:

- The data is in a csv format hence we use pandas.read_csv method with delimiter = “/t”
- We also skip some bad lines using error_bad_lines=False

Data Description and Data Cleaning:

- We first see the number of columns in the dataframe. There are 16 columns.

```
In [8]: data1.columns

Out[8]: Index(['marketplace', 'customer_id', 'review_id', 'product_id',
               'product_parent', 'product_title', 'product_category', 'star_rating',
               'helpful_votes', 'total_votes', 'vine', 'verified_purchase',
               'review_headline', 'review_body', 'review_date', 'sentiment_rating'],
              dtype='object')
```

- We see that product_category, star_rating, review_date, total_votes, product_parent could be useful candidates to draw some analysis on sentiment
- To check the number of total records in the dataframe we can use the count method. We see that there are 6900886 records. There is one column with fewer count values, indicating null values.

```
In [16]: data1.count()

Out[16]: marketplace          6900886
         customer_id          6900886
         review_id            6900886
         product_id           6900886
         product_parent       6900886
         product_title        6900886
         product_category     6900886
         star_rating          6900885
         helpful_votes        6900885
         total_votes          6900885
         vine                 6900885
         verified_purchase    6900885
         review_headline      6900811
         review_body          6900810
         review_date          6900562
         sentiment_rating     6495960
         dtype: int64
```

- We remove any null/ na values using dropna()
- To check the data statistics for columns with continuous values we use describe() method. We can see that the mean star_rating is 4.3 and minimum = 1 with maximum being 5. 75th Percentile has 5 and 25th percentile as 4 star rating indicating unbalanced dataset.

	customer_id	product_parent	star_rating	helpful_votes	total_votes
count	6.900886e+06	6.900886e+06	6.900885e+06	6.900885e+06	6.900885e+06
mean	2.918797e+07	4.933005e+08	4.306591e+00	2.044545e+00	3.251676e+00
std	1.565196e+07	2.861562e+08	1.146197e+00	3.184494e+01	3.633977e+01
min	1.000100e+04	2.254720e+05	1.000000e+00	0.000000e+00	0.000000e+00
25%	1.501694e+07	2.495109e+08	4.000000e+00	0.000000e+00	0.000000e+00
50%	2.881076e+07	4.965205e+08	5.000000e+00	0.000000e+00	0.000000e+00
75%	4.414300e+07	7.448221e+08	5.000000e+00	1.000000e+00	2.000000e+00
max	5.309659e+07	9.999881e+08	5.000000e+00	2.755000e+04	2.872700e+04

Data Augmentation:

- As part of data augmentation we see that the sentiment analysis is a pure classification problem. Hence we bin the star_rating column into 2 categories - positive and negative. Negative = 0 < star_rating < 2.5 and Positive = 2.5 < star_rating < 5.

```
bins = [1,2.5,5]
labels = ['negative', 'positive']
data1['sentiment_rating'] = pd.cut(data1['star_rating'], bins=bins, lab
```

- We see that the newly created column sentiment_rating is unbalanced. In order for us to balance this dataset we perform a data aggregation group by on sentiment_rating and from each group we sample 3000 records and append to a final_dataframe

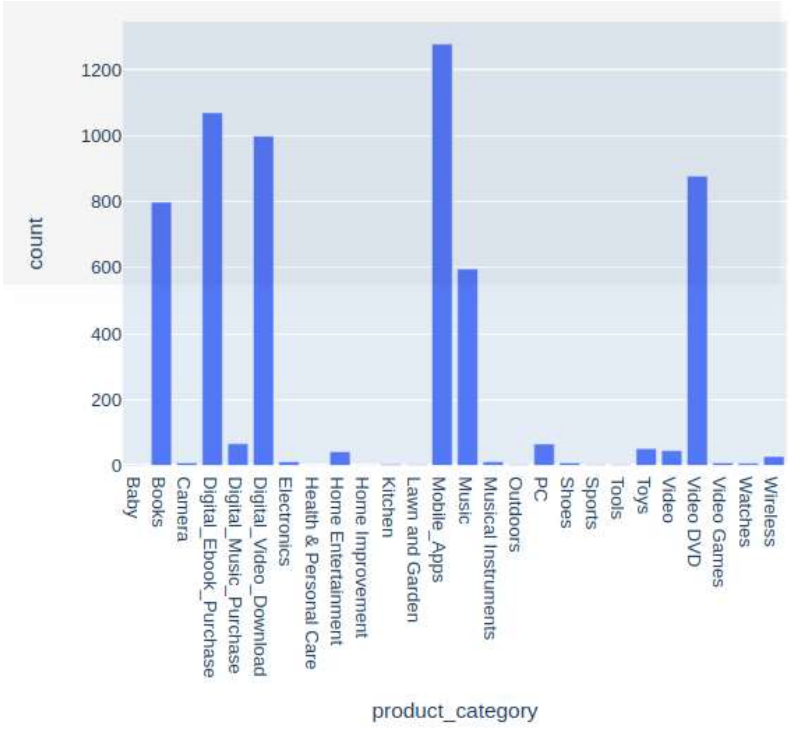
```
g = data1.groupby('sentiment_rating')
negative = g.get_group('negative')
positive = g.get_group('positive')
negative = negative.sample(3000, random_state=20)
positive = positive.sample(3000, random_state=20)
final_data = negative.append(positive)
```

- We also like to analyze the review_date column, hence we augment this column into separate fields of month, year, day of week, hour, using pandas datetime package.

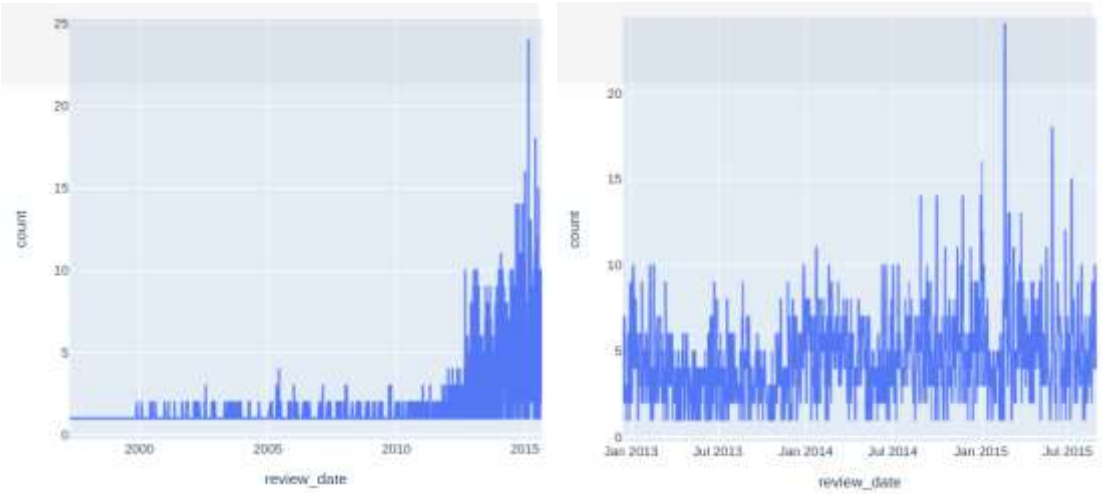
```
final_data['review_date'] = pd.to_datetime(final_data['review_date'])
final_data['month'] = final_data['review_date'].dt.month
final_data['year'] = final_data['review_date'].dt.year
final_data['hour'] = final_data['review_date'].dt.hour
final_data['day'] = final_data['review_date'].dt.dayofweek
```

Data Visualization:

- We check the types of product categories and the distribution of its count in the dataset using a group by on product_category and aggregating the index count. We can see that Mobile_apps, Digital_Ebooks and Video DVD have the highest number of reviews. Mobile_apps have 1260 reviews

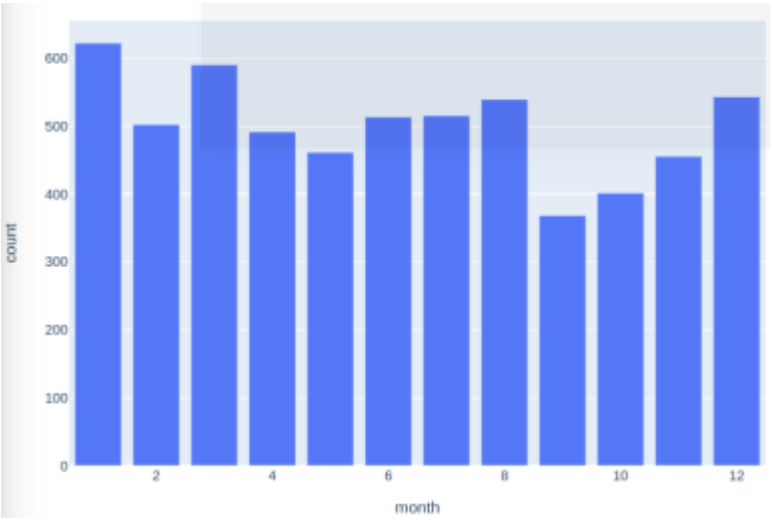


- We are interested in the number of reviews by date. We perform a group by on review_date and aggregate the count on the index field. This helps us give a time series based dataframe which we can plot using plotly.. The reason we use plotty is because plotty features zoom in capability. We can zoom into a particular span of interest to get more information by day or by month. Upon plotting we see that there are a greater number of reviews from 205 to 2016. After zooming into this period we can see that, the number of reviews are the greatest(25) in the month of January 2015

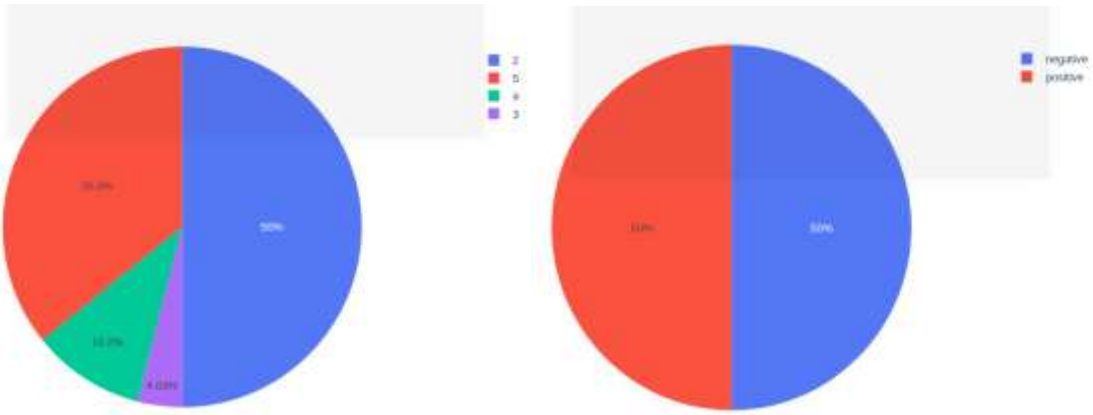


- Next, we are interested in the month which have the highest reviews in the dataset. This will give us an understanding of the months when people generally buy amazon products and write reviews about it. We answer this question by performing a group by on the month field and aggregating the count on index

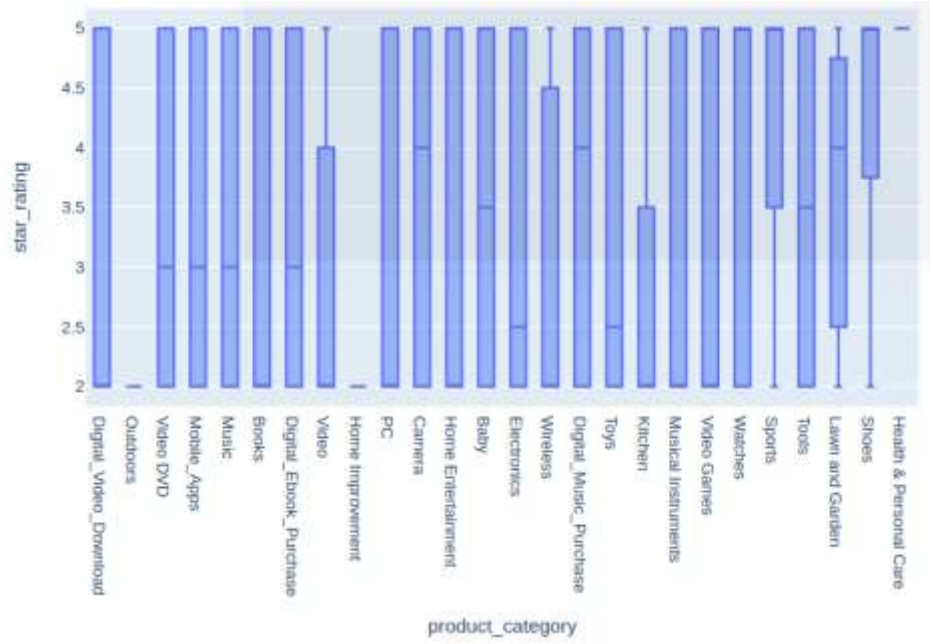
```
reviews_by_month_count = final_data[['month', 'index']].\
                         groupby('month').agg({'index': 'count'}).\
                         rename({'index': 'count'}, axis=1).reset_index()

px.bar(reviews_by_month_count, x='month', y='count')
```



- We want to see how the ratings are distributed in the dataset. Hence we perform a groupby aggregation on start_rating and sentiment_rating field. We can clearly see that there are 50% reviews with rating = 2.0 and 35% reviews with rating 5.0 and 10% with rating 4.0. But running the same analysis on the augmented field sentiment_rating which is a categorical field based on star_rating we see that the positive and negative are both balanced.



- Next, we want to check which category of products have high average star_rating and low average star_rating. We can achieve this with a box plot on the star_rating and product_category column. We see that Video_DVD, Mobile_Apps, Ebook, Music have 5.0 ratings in the third quartile. Whereas, kitchen products have a median rating of 2.0 and third quartile being 3.5.



Data Aggregation and Results:

Based on our analysis so far, we have been able to answer the following questions:

- Which products has lower reviews - Kitchen, Sports and Watches
- Which products have higher reviews - Mobile Apps, Ebooks, Video DVD

- Which products have high and low rating - Mobile Apps, Ebook have high rating median 5.0 and Kitchen has low rating median 2.0.
- Which months have higher number of reviews - January, March and December
- Which year has the highest number of reviews - 2015

Data Persistence:

- We persist the cleaned dataset in a csv format for sentiment analysis
- We use the to_csv method with index set to False and Heder set to True.
- Since we only need the review text and labels we omit the other columns and just save the 2 columns for further analysis.

```
final_data[['sentiment_rating', 'review_body']].\
    to_csv('amazon_cleaned_data.csv', index=False)
```

Sentiment Analysis:

For the sentiment Analysis part we are trying to answer following questions:

- Which products has lower reviews
- Which products have higher reviews
- Factors affecting the reviews
- Products ratings for each category of reviews

This analysis can help us understand which products should be kept, dropped from Amazon's product junk. Also, can we correlate positive and negative sentiments for each product in Amazon's Catalog By using Sentiment analysis

Data Preparation for the Sentiment Analysis steps:

1. Preprocess the data set using Python and NLP techniques
2. Convert text data into numeric format - Example - Ngram, Bi-gram, Encoding, and other forms of tokens in text.
3. Visualizations of the data set.
4. Modelling : To understand the relationship between reviews and sentiment

For this part of the project we will be working with 6000 reviews which we extracted from the randomizing data collected from the steps mentioned above.

Step 1: Preprocessing the review text

For generating the model to predict the ‘positive’ or ‘negative’ we will be taking the reviews from the column 'review_body' and converting these reviews in sentences and applying the below techniques
i. tokenization to convert sentences into word

ii. Removing the repeating words such as Stops words i.e. a, and , the etc. We do this because these words carry no meaning or add no value in generating the sentiment of the text paragraph.

iii. Removing the unwanted punctuations using the regular expression

iv. Convert all the sentences to lower case format so that we are treating the meaning of the words equally.

v. Giving the labels to each word using the sentences polarity corpus

vi. Feature selection for models

Step 2: Bag of word approach for feature selection

We are using the Bag of Words approach for generating the features from the corpus of the review data. It is a common practice in Natural Language Processing techniques for generating the features of texts. It eliminates the grammar dependency and gives values to the recurring words. The idea behind is predicting the sentiment of the entire paragraph based on the word meaning rather that on the placement of the words.

. For this approach we will be using the sentence polarity class from the NLTK library. And we will be giving labels to the words.

```
[ ] import nltk
    nltk.download('sentence_polarity')
    from nltk.corpus import sentence_polarity
```

```
##### Collect the sentences from the sentence_polarity corpus
bag_of_sents = sentence_polarity.sents()

#Obtain the sentences with their labels
sents_with_labels = [(sent, label) for label in sentence_polarity.categories()
                     for sent in sentence_polarity.sents(categories=label)]
```

```
sents_with_labels
```

```
[(['simplistic', ',', 'silly', 'and', 'tedious', '.'], 'neg'),
```

Later on we'll re-shuffle the sentences to randomize the train and test data.

```
#Shuffle the sentences
random.shuffle(sents_with_labels)

#Obtain the most frequent words from these sentences
#Retreiving the most frequent 2000 words

all_sents = [word for (sent, label) in sents_with_labels for word in sent if word not in newstopwords]
all_sents_freq = nltk.FreqDist(all_sents)
most_freq_words = all_sents_freq.most_common(2000)
freq_word_features = [word for (word,freq) in most_freq_words]
```

Separating the data into train and test.

```
[ ]
    #Define the function for words as features
    def document_features(sentence, words):
        unique_words = set(sentence)
        features = {}
        for word in words:
            features['contains({})'.format(word)] = (word in unique_words)
        return features
```

```
[ ] #Obtaining the feature set for the sentences
    feature_set_1 = [(document_features(sent, freq_word_features), label) for (sent, label) in sents_with_labels]

    #Split the feature set data into training and test data
    train_feature_set_1, test_feature_set_1 = feature_set_1[1000:], feature_set_1[:1000]
```

Step 4.

1. **Creating the baseline Model 1 : Naive Bayes with BOW as a Features**

```
#Apply NB Classification on the training feature set
classifier_1 = nltk.NaiveBayesClassifier.train(train_feature_set_1)
```

```
# Display all the measures for a given classifier
def display_accuracy_measures(actual_set, input_classifier):
    reference_sets = collections.defaultdict(set)
    predicted_sets = collections.defaultdict(set)
    for i, (feats, label) in enumerate(actual_set):
        reference_sets[label].add(i)
        predicted = input_classifier.classify(feats)
        predicted_sets[predicted].add(i)
    print('Positive precision:', precision(reference_sets['pos'], predicted_sets['pos']))
    print('Positive recall:', recall(reference_sets['pos'], predicted_sets['pos']))
    print('Positive F-measure:', f_measure(reference_sets['pos'], predicted_sets['pos']))
    print('Negative precision:', precision(reference_sets['neg'], predicted_sets['neg']))
    print ('Negative recall:', recall(reference_sets['neg'], predicted_sets['neg']))
    print ('Negative F-measure:', f_measure(reference_sets['neg'], predicted_sets['neg']))
```

With Naive Bayes Model we go the Accuracy of 74.5%

```
#Determine the Accuracy
print("The baseline accuracy of Model 1 : Word features is  --> ",nltk.classify.accuracy(classifier_1, test_feature_set_1)*100)
```

```
The baseline accuracy of Model 1 : Word features is  -->  74.5
```

```
#Display the other measures
display_accuracy_measures(feature_set_1, classifier_1)
```

```
Positive precision: 0.8012709416522241
Positive recall: 0.7805289814293753
Positive F-measure: 0.790763968072976
Negative precision: 0.7868669226549644
Negative recall: 0.8064153066966798
Negative F-measure: 0.7961111111111112
```

This model has good precision to recall trade-off.

2. Model 2 : Naive Bayes model with Subjectivity Features

Subjectivity helps to determine the polarity of sentences. I.e. Positivity or Negativity.

```
#Feature-2 - Subjectivity Count Features
def readSubjectivity(path):
    flexicon = open(path, 'r')
    sldict = { }
    for line in flexicon:
        fields = line.split()   # default i\s to split on whitespace
        # split each field on the '=' and keep the second part as the value
        strength = fields[0].split("=")[1]
        word = fields[2].split("=")[1]
        posTag = fields[3].split("=")[1]
        stemmed = fields[4].split("=")[1]
        polarity = fields[5].split("=")[1]
        if (stemmed == 'y'):
            isStemmed = True
        else:
            isStemmed = False
        sldict[word] = [strength, posTag, isStemmed, polarity]
    return sldict

SLpath = "subjclueslen1-HLTEMNLP05.tff"
SL = readSubjectivity(SLpath)
```

```
#Define the Subjectivity Count word features
def SL_features(sent, word_features, SL):
    sent_words = set(sent)
    features = {}
    for word in word_features:
        features['contains(%s)' % word] = (word in sent_words)
        # count variables for the 4 classes of subjectivity
        weakPos = 0
        strongPos = 0
        weakNeg = 0
        strongNeg = 0
        for word in sent_words:
            if word in SL:
                strength, posTag, isStemmed, polarity = SL[word]
                if strength == 'weaksubj' and polarity == 'positive':
                    weakPos += 1
                if strength == 'strongsubj' and polarity == 'positive':
                    strongPos += 1
                if strength == 'weaksubj' and polarity == 'negative':
                    weakNeg += 1
                if strength == 'strongsubj' and polarity == 'negative':
                    strongNeg += 1
                features['positivecount'] = weakPos + (2 * strongPos)
                features['negativecount'] = weakNeg + (2 * strongNeg)
    return features
```

Creating the feature set:

```
#Obtaining the feature set for the sentences
feature_set_2 = [(SL_features(sent, freq_word_features,SL), label) for (sent, label) in sents_with_labels]
```

```
feature_set_2[1]
```

```
({'contains(.)': True,
  'positivecount': 5,
  'negativecount': 0,
  'contains(,)': False,
  'contains(film)': False,
  'contains(movie)': False,
  'contains(not)': False,
  'contains(one)': False
```

Running the model and calculating the accuracy:

```
print(feature_set_2[0][0]['positivecount'])
print(feature_set_2[0][0]['negativecount'])
```

```
6
0
```

```
train_set_2, test_set_2 = feature_set_2[1000:], feature_set_2[:1000]
classifier2 = nltk.NaiveBayesClassifier.train(train_set_2)
```

```
print("The accuracy after adding Subjectivity Lexicon features --> ",nltk.classify.accuracy(classifier2, test_set_2)*100)
```

```
The accuracy after adding Subjectivity Lexicon features -->  75.7
```

This Naive Bayes model has better accuracy that the word with BOW appraoch.

3. Model 3 : Naive Bayes with Negation Features

In order to identify the negative words such as wasn't to be considered as 'was' and 'not'. We will be using the Negation feature attribute of the NLP.

```
negationwords = ['no', 'not', 'never', 'none', 'nowhere', 'nothing', 'noone', 'rather', 'hardly', 'scarcely', 'rarely', 'seldom', 'neither', 'nor']
```

```
# Negation Fetures


def NOT_features(document, word_features, negationwords):
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = False
        features['contains(NOT{})'.format(word)] = False
    # Parse the document words in order
    for i in range(0, len(document)):
        word = document[i]
        if ((i + 1) < len(document)) and ((word in negationwords) or (word.endswith("n't"))):
            i += 1
            features['contains(NOT{})'.format(document[i])] = (document[i] in word_features)
        else:
            features['contains({})'.format(word)] = (word in word_features)
    return features
```

Creatinng the traing and test data and running the model with newly formed features.

```
#Obtaining the feature set for the sentences
feature_set_3 = [(NOT_features(sent, freq_word_features, negationwords), label) for (sent, label) in sents_with_labels]
```

```
#Split the feature set data into training and test data
train_feature_set_3, test_feature_set_3 = feature_set_3[2000:], feature_set_3[:2000]

#Apply NB Classification on the training feature set
classifier_3 = nltk.NaiveBayesClassifier.train(train_feature_set_3)
```

```
#Determine the Accuracy
print("The accuracy after adding Negation features --> ",nltk.classify.accuracy(classifier_3, test_feature_set_3)*100)
```

We are getting the accruacy of 77. 10%. Hence this approach is proves to be the best method for running the sentiment analysis.

```
#Display the other measures
display_accuracy_measures(feature_set_3, classifier_3)
```

```
The accuracy after adding Negation features -->  77.10000000000001
Positive precision: 0.9311469121595704
Positive recall: 0.9107109360345151
Positive F-measure: 0.9208155523944997
Negative precision: 0.9126284875183553
Negative recall: 0.9326580378915775
Negative F-measure: 0.9225345579367289
```

Generating the confusion matrix and evaluting the results:

```
## Using the classifier after Negation since it's my best classifier.
nList = []

testList = []

for(features , label) in test_feature_set_3:
    nList.append(label)
    testList.append(classifier_3.classify(features))
```

```
confusionMatrix = ConfusionMatrix(nList,testList)

print(confusionMatrix)
```

```
     |   n   p |
     |   e   o |
     |   g   s |
-----+---------+
neg  |<816>221 |
pos  | 237<726>|
-----+---------+
(row = reference; col = test)
```

Displaying the positive and negative sentences, which is the main essence of the this project.

```
for i in pos_sentence_list_1[:10]:
    print(i, "\n")
```

```
 Sentences classified with 'pos' tag :

i bought 3 of bridgmans books: life drawing, constructive anatomy and heads, features and faces after reading the reviews.

no one mentioned that most of the pages are dedicated to the anatomy of men, however i am interested in the female anatomy, so, these books did not wo

i was expecting to see the movie itself and not a kind of trailer.

when i hear the word fierce, i expect it to be fierce.
```

```
print(" Sentences classified with 'neg' tag : \n \n " )

for i in neg_sentence_list_1[6:10]:
    print(i, "\n")
```

```
 Sentences classified with 'neg' tag :

 i honestly didn't listen to each song to the last second because i'd die of boredom before reaching it.

 diva, i think the only reason i really liked it is the beat, sounds so close to a milli by lil wayne.

 and it should, it was the same producer (bangladesh).

 i wouldn't buy this album.
```

Conclusion:

- The analysis on amazon reviews give us a clear understanding on which products have greater reviews.
- We also analyzed the distribution of the reviews by date it was posted and the categories it belonged to. This would give any firm an idea of targeting the end users with good products
- We also performed sentiment analysis on the reviews posted by users. This will allow us to understand the usage pattern of end users and provide good products.
- It will also allow any firm to redesign their marketing strategy to further please the customers.
- We were able to understand various scripting packages involved in the field of Data Analytics.