Amazon Reviews of Musical Equipment

A NLP Project

Final Capstone Project

Intro

As a consumer in this day and age, I do most of my shopping online. This trend of doing my shopping online created a certain level of skepticism when I buy products due to a few bad experiences I have had with my purchases.

Due to this fact, product reviews are extremely important to me because they determine if I should purchase a product or not. Such product review data can create a huge impact on the sales of a product from a companies point of view.

Data;

The data was retrieved from http://jmcauley.ucsd.edu/data/amazon/ by accessing the website. The variables from the dataset that would be utilized, are the reviews and the sentiment in the review (positive versus negative).

Use of Specialization;

Techniques intended to be used include use of several plotting techniques to understand the data, use NLP and NLTK (the use of text processing methods such as tokenizations, stop word removal, stemming and vectorizing text via term frequencies (TF) as well as the inverse document frequencies (TF-IDF)). The use of topic modelling would be done with Latent Dirichlet Allocation (LDA) and Sentiment Analysis. For evaluation, Recall and F1 score would be used in addition to Receiver Operating Characteristic score and Confusion Matrix

Product/Business Impact;

This project is valuable because it provides actionable insight on customers reactions to a product. It can assist in better marketing of the product and directed improvement to increase customer satisfaction, which in turn may provide a more favorable view of the product and increase sales. In addition, the model created would assist in predicting whether future reviews are positive or negative towards the product. Project goals include;

- Properly cleaning the data and using visualization to understand the dataset,
- Applying at least 2 models to determine process with highest accuracy.
- Creating a Sentiment analysis and a predictive model.

Being able to analyse that data to determine consumers' sentiments of the products, the flaws and even the most enjoyed feature can allow a company to improve on the product, and market it better based on the top feature mentioned.

The Dataset

The data contains the following columns:

asin – ID of the product

helpful – helpfulness rating of the review, e.g. 2/3

overall – rating of the product

reviewText – text of the review

reviewTime – time of the review (raw)

reviewerID – ID of the reviewer

reviewerName – name of the reviewer

summary – summary of the review

unixReviewTime – time of the review (unix time)

Lets import some packages

```
# Import Packages
import numpy as np
import pandas as pd
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.decomposition import NMF, LatentDirichletAllocation
from matplotlib import pyplot as plt
%matplotlib inline
import scipy
import seaborn as sns
import re

import warnings
warnings.filterwarnings('ignore')
```

Exploring the data

```
df = pd.read_json('Musical_Instruments_5.json', lines=True)
df.dropna()
df.head()
```

	asin	helpful	overall	reviewText	reviewTime	reviewerID	reviewerName	summary	unixReviewTime
0	1384719342	[0, 0]	5	Not much to write about here, but it does exac...	02 28, 2014	A2IBPI20UZIR0U	cassandra tu "Yeah, well, that's just like, u...	good	1393545600
1	1384719342	[13, 14]	5	The product does exactly as it should and is q...	03 16, 2013	A14VAT5EAX3D9S	Jake	Jake	1363392000
2	1384719342	[1, 1]	5	The primary job of this device is to block the...	08 28, 2013	A195EZSQDW3E21	Rick Bennette "Rick Bennette"	It Does The Job Well	1377648000
3	1384719342	[0, 0]	5	Nice windscreen protects my MXL mic and preven...	02 14, 2014	A2C00NNG1ZQQG2	RustyBill "Sunday Rocker"	GOOD WINDSCREEN FOR THE MONEY	1392336000
4	1384719342	[0, 0]	5	This pop filter is great. It looks and perform...	02 21, 2014	A94QU4C90B1AX	SEAN MASLANKA	No more pops when I record my vocals.	1392940800

```python
# Start exploring the dataset. Lets take a look at the columns
print(df.columns)
```

```
Index(['asin', 'helpful', 'overall', 'reviewText', 'reviewTime', 'reviewerID',
       'reviewerName', 'summary', 'unixReviewTime'],
      dtype='object')
```

```python
print("There are {} observations and {} features in this dataset. \n".format(df.shape[0],df.shape[1]))
# Data types
df.dtypes
```

```
There are 10261 observations and 9 features in this dataset.


asin              object
helpful           object
overall            int64
reviewText        object
reviewTime        object
reviewerID        object
reviewerName      object
summary           object
unixReviewTime     int64
dtype: object
```

```
# Checking for Null columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10261 entries, 0 to 10260
Data columns (total 9 columns):
asin              10261 non-null object
helpful           10261 non-null object
overall           10261 non-null int64
reviewText        10261 non-null object
reviewTime        10261 non-null object
reviewerID        10261 non-null object
reviewerName      10234 non-null object
summary           10261 non-null object
unixReviewTime    10261 non-null int64
dtypes: int64(2), object(7)
memory usage: 721.6+ KB
```

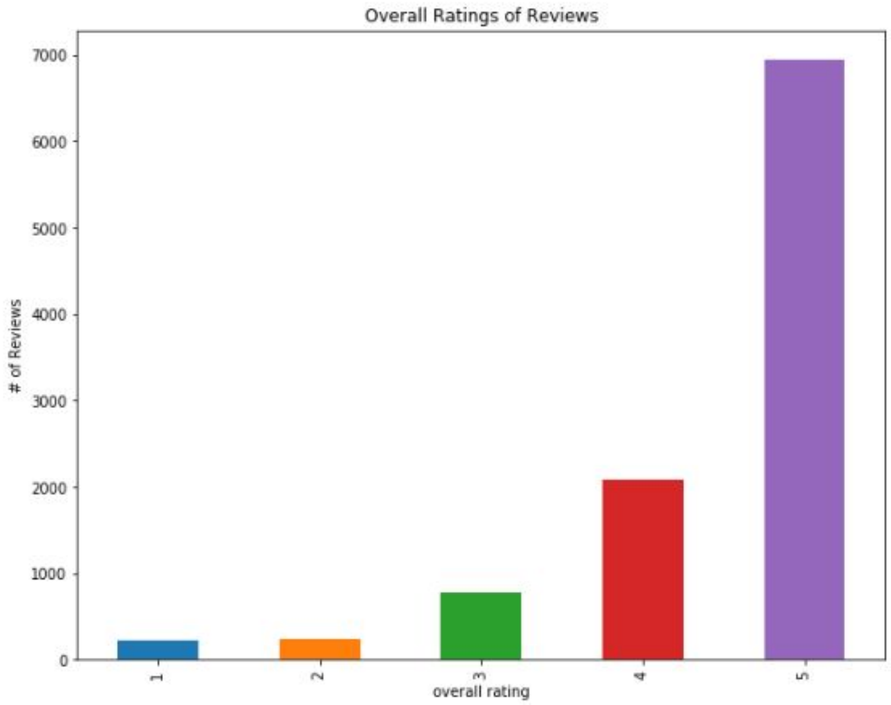
```
# Groupby by rating
reviewText_ = df.groupby("overall")

# Summary statistic of all sentimentS
reviewText_.describe()
```

	unixReviewTime							
	count	mean	std	min	25%	50%	75%	max
overall								
1	217.0	1.363610e+09	3.693997e+07	1.141344e+09	1.347926e+09	1.370995e+09	1.390349e+09	1.405210e+09
2	250.0	1.361242e+09	3.770940e+07	1.190678e+09	1.342116e+09	1.369872e+09	1.389506e+09	1.405210e+09
3	772.0	1.361718e+09	3.633831e+07	1.161389e+09	1.343282e+09	1.369008e+09	1.389053e+09	1.405901e+09
4	2084.0	1.359799e+09	3.914760e+07	1.095466e+09	1.342915e+09	1.369138e+09	1.388707e+09	1.405987e+09
5	6938.0	1.360608e+09	3.757515e+07	1.096416e+09	1.343606e+09	1.367971e+09	1.388945e+09	1.405901e+09

Visualization of the review distribution based on the rating

(1 being the most negative and 5 being the most positive)

```
# Visualization of the review distribution based on the rating
df['overall'].value_counts().sort_values().plot(kind='bar', figsize=(10,8))
plt.title('Overall Ratings of Reviews')
plt.ylabel('# of Reviews')
plt.xlabel('overall rating')
plt.show()
```



```python
# Drop missing values
df.dropna(inplace=True)

# Remove any 'neutral' ratings equal to 3
df = df[df['overall'] != 3]

# Encode 4s and 5s as 1 (rated positively)
# Encode 1s and 2s as 0 (rated negatively)
df['rating'] = np.where(df['overall'] > 3, 1, 0)
df.head()
```

	asin	helpful	overall	reviewText	reviewTime	reviewerID	reviewerNan
0	1384719342	[0, 0]	5	Not much to write about here, but it does exac...	02 28, 2014	A2IBPI20UZIR0U	cassandra tu "Yea well, that's just lik u
1	1384719342	[13, 14]	5	The product does exactly as it should and is q...	03 16, 2013	A14VAT5EAX3D9S	Ja
2	1384719342	[1, 1]	5	The primary job of this device is to block the...	08 28, 2013	A195EZSQDW3E21	Rick Bennette "Ri Bennett
3	1384719342	[0, 0]	5	Nice windscreen protects my MXL mic and preven...	02 14, 2014	A2C00NNG1ZQQG2	RustyBill "Sund Rocke
4	1384719342	[0, 0]	5	This pop filter is great. It looks and perform...	02 21, 2014	A94QU4C90B1AX	SEAN MASLANK

Lets remove any neutral rating.

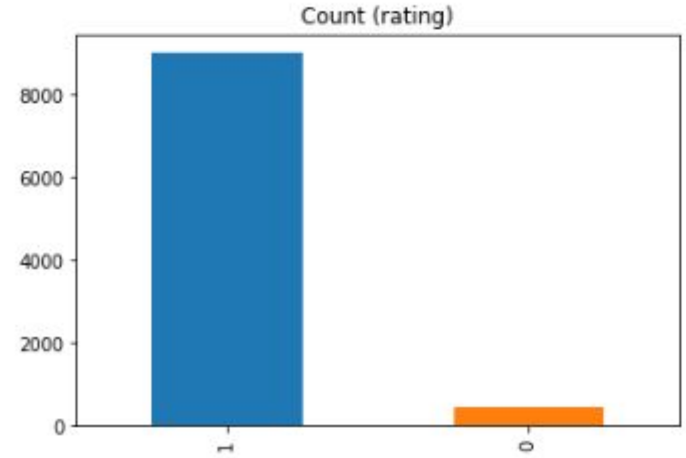
A rating of 3 would be considered a neutral rating.

- Ratings of 4 and 5 would be considered a positive rating
- Ratings of 1 and 2 would be considered a negative rating

Visualization of the positive versus negative reviews

```python
# Visualization of the positive versus negative reviews

rating_count = df.rating.value_counts()
print('Negative Rating:', rating_count[0])
print('Positive Rating:', rating_count[1])
print('Proportion:', round(rating_count[0] / rating_count[1], 2), ': 1')

rating_count.plot(kind='bar', title='Count (rating)');
```

```
Negative Rating: 465
Positive Rating: 8998
Proportion: 0.05 : 1
```



Length of Reviews

Finding correlation between the length of the positive and negative reviews

```python
def length(text):
    '''a function which returns the length of text'''
    return len(text)

# Apply the function to each review

df['length'] = df['reviewText'].apply(length)
df.head()
```

	asin	helpful	overall	reviewText	reviewTime	reviewerID	reviewerName	summary	unixReviewTime	rating	length
0	1384719342	[0, 0]	5	Not much to write about here, but it does exac...	02 28, 2014	A2IBPI20UZIR0U	cassandra tu "Yeah, well, that's just like, u...	good	1393545600	1	268
1	1384719342	[13, 14]	5	The product does exactly as it should and is q...	03 16, 2013	A14VAT5EAX3D9S	Jake	Jake	1363392000	1	544
2	1384719342	[1, 1]	5	The primary job of this device is to block the...	08 28, 2013	A195EZSQDW3E21	Rick Bennette "Rick Bennette"	It Does The Job Well	1377648000	1	436
3	1384719342	[0, 0]	5	Nice windscreen protects my MXL mic and preven...	02 14, 2014	A2C00NNG1ZQQG2	RustyBill "Sunday Rocker"	GOOD WINDSCREEN FOR THE MONEY	1392336000	1	206
4	1384719342	[0, 0]	5	This pop filter is great. It looks and perform...	02 21, 2014	A94QU4C90B1AX	SEAN MASLANKA	No more pops when I record my vocals.	1392940800	1	159

Average length of the Reviews

```python
# Lets create a dataset for the positive and negative reviews

positive_review = df[df['rating'] == 1]
negative_review = df[df['rating'] == 0]

```

```python
# Lets calculate the average length of the reviews

positive_review_mean = np.mean(positive_review['length'])
negative_review_mean = np.mean(negative_review['length'])
print('Average length of the Postive reviews:', positive_review_mean)
print('Average length of the Negative reviews:', negative_review_mean)
```

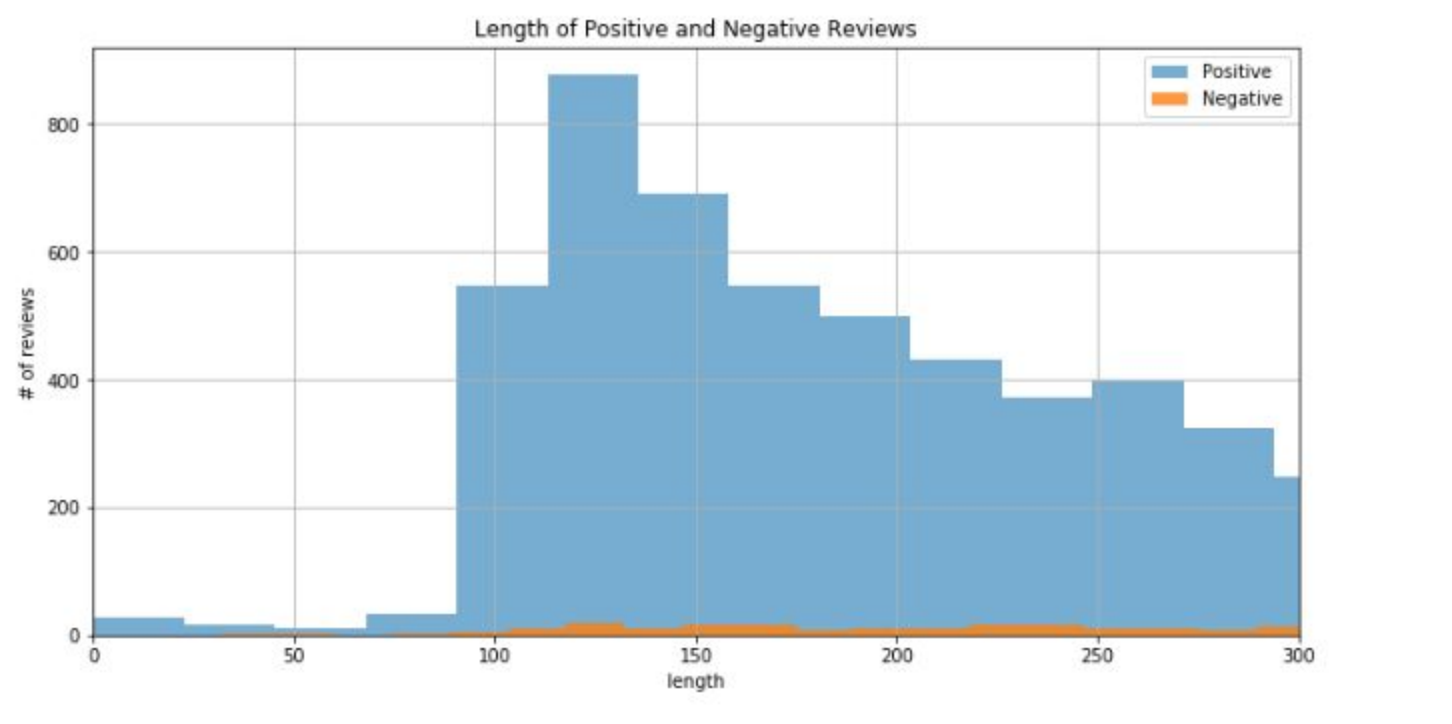
```
Average length of the Postive reviews: 473.7883974216493
Average length of the Negative reviews: 577.7763440860215
```

As you can see the average length of a negative review is longer than a positive review. The average length of the negative review comes in at **578 words** per review, while the positive review comes in at **474 words**.

Lets plot the histogram

```
# Lets look at the distribution of the lengths of the positive versus negative reviews

import matplotlib
from matplotlib import pyplot as plt
%matplotlib inline

matplotlib.rcParams['figure.figsize'] = (12.0, 6.0)
bins = 500
plt.hist(positive_review['length'], alpha = 0.6, bins=bins, label='Positive')
plt.hist(negative_review['length'], alpha = 0.8, bins=bins, label='Negative')
plt.xlabel('length')
plt.ylabel('# of reviews')
plt.title('Length of Positive and Negative Reviews')
plt.legend(loc='upper right')
plt.xlim(0,300)
plt.grid()
plt.show()
```

Visualization of the review lengths



Topic modeling

Topic Modeling is a process to automatically identify topics present in a text object and to derive hidden patterns exhibited by a text corpus. Topic Models are very useful for multiple purposes, including:

Document clustering

Organizing large blocks of textual data

Information retrieval from unstructured text

Feature selection

The goal here is to extract a certain number of groups of important words from the reviews. These groups of words are basically the topics which would help in ascertaining what the consumers are actually talking about in the reviews.

Let's do some data preprocessing

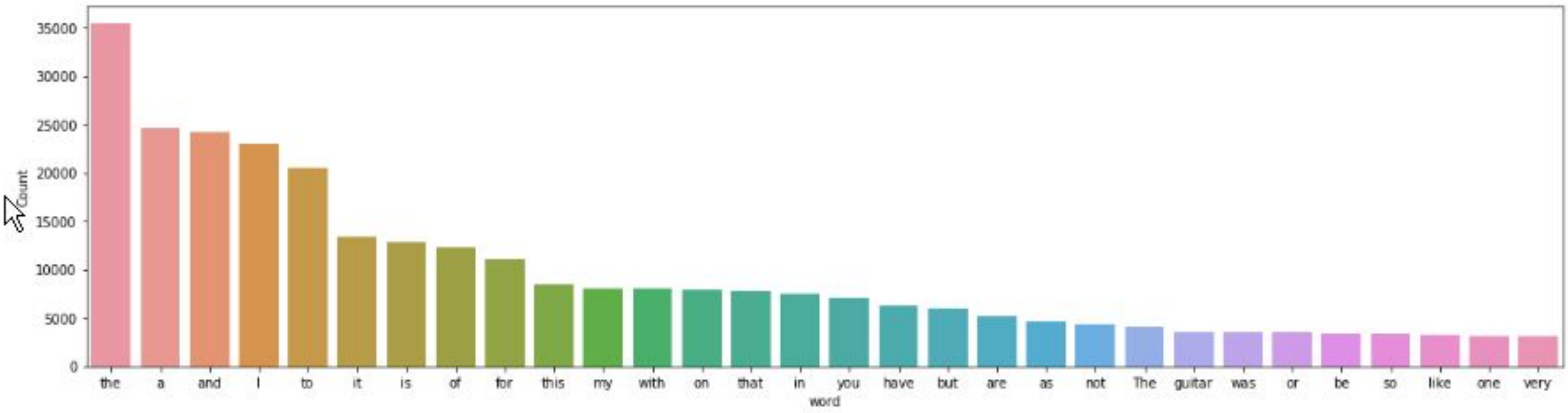
We will remove the punctuations, stopwords and normalize the reviews as much as possible. After every preprocessing step, it is a good practice to check the most frequent words in the data. Therefore, let’s define a function that would plot a bar graph of n most frequent words in the data.

```
import nltk
from nltk import FreqDist

import gensim
from gensim import corpora

import pyLDAvis
import pyLDAvis.gensim
```

```
# Lets create a function to plot most frequent terms
def freq_words(x, terms = 30):
  all_words = ' '.join([text for text in x])
  all_words = all_words.split()

  fdist = FreqDist(all_words)
  words_df = pd.DataFrame({'word':list(fdist.keys()), 'count':list(fdist.values())})

  # selecting top 20 most frequent words
  d = words_df.nlargest(columns="count", n = terms)
  plt.figure(figsize=(20,5))
  ax = sns.barplot(data=d, x= "word", y = "count")
  ax.set(ylabel = 'Count')
  plt.show()
```

The most frequent terms before processing the data

```
freq_words(df['reviewText'])
```

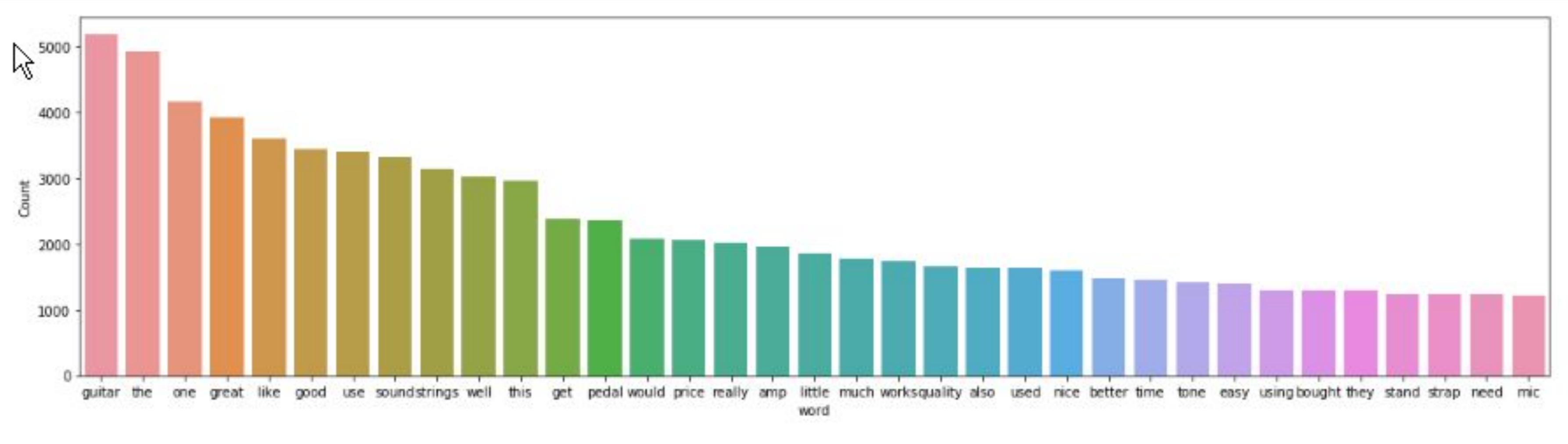


As you can see, the most common words are ‘the’, ‘a’, ‘and’, so on. These words are not so important for our task and they do not tell any story. We’ have to get rid of these kinds of words. Before that let’s remove the punctuations and numbers from our text data.

Let's remove the Stopwords

```python
# Let's try to remove the stopwords and short words (<2 letters) from the reviews.

from nltk.corpus import stopwords
stop_words = stopwords.words('english')

# function to remove stopwords
def remove_stopwords(rev):
    rev_new = " ".join([i for i in rev if i not in stop_words])
    return rev_new

# remove short words (length < 3)
df['reviewText'] = df['reviewText'].apply(lambda x: ' '.join([w for w in x.split() if len(w)>2]))

# remove stopwords from the text
reviews = [remove_stopwords(r.split()) for r in df['reviewText']]

# make entire text lowercase
reviews = [r.lower() for r in reviews]
```

After removing the Stowords lets see what the most frequent words now look like.



We can see some improvement here. Terms like ‘guitar’, ‘the’, ‘one’, ‘great’ have come up which are quite relevant for the Musical equipment category. However, we still have neutral terms like ‘the’, ‘use’, ‘get’, ‘also’ which are not that relevant.

Let’s tokenize the reviews and then lemmatize the tokenized reviews

```
tokenized_reviews = pd.Series(reviews).apply(lambda x: x.split())
print(tokenized_reviews[1])
```

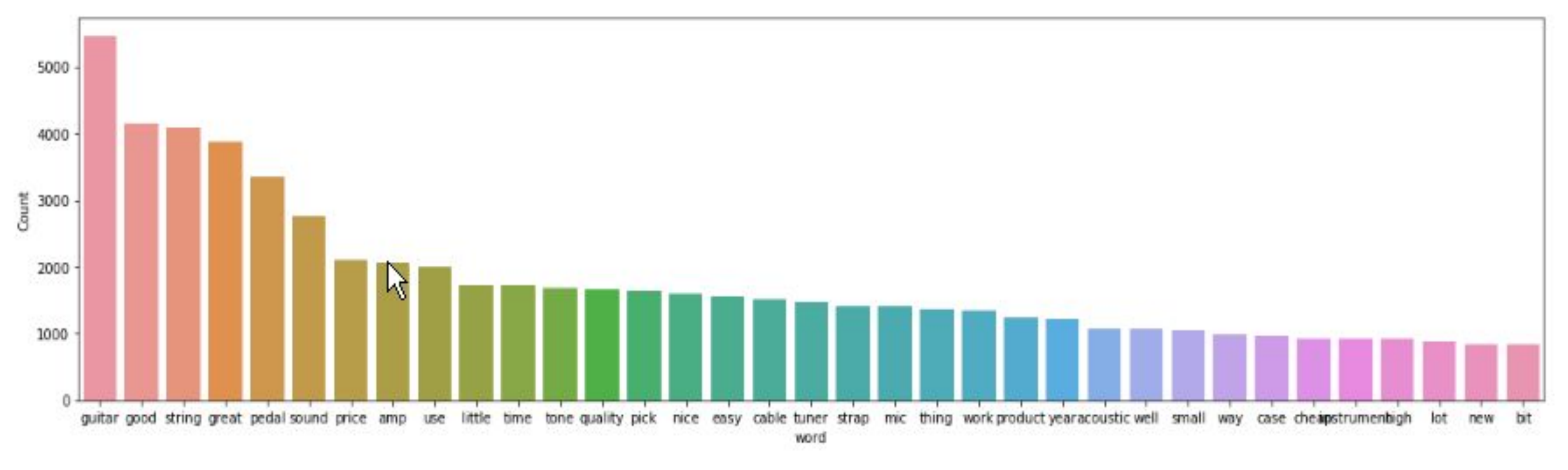
```
['the', 'product', 'exactly', 'quite', 'affordable', 'realized', 'double', 'screened', 'arrived', 'even', 'better', 'expected',
'added', 'bonus', 'one', 'screens', 'carries', 'small', 'hint', 'smell', 'old', 'grape', 'candy', 'used', 'buy', 'reminiscent',
'sake', 'cannot', 'stop', 'putting', 'pop', 'filter', 'next', 'nose', 'smelling', 'recording', 'dif', 'needed', 'pop', 'filte
r', 'work', 'well', 'expensive', 'ones', 'may', 'even', 'come', 'pleasing', 'aroma', 'like', 'mine', 'buy', 'product']
```

```
reviews_2 = lemmatization(tokenized_reviews)
print(reviews_2[1]) # print lemmatized review
```

```
['product', 'affordable', 'double', 'screened', 'bonus', 'screen', 'small', 'hint', 'smell', 'old', 'grape', 'candy', 'reminisc
ent', 'sake', 'pop', 'filter', 'next', 'nose', 'recording', 'dif', 'pop', 'filter', 'work', 'expensive', 'one', 'aroma', 'produ
ct']
```

We have not just lemmatized the words but also filtered only nouns and adjectives.
Let’s de-tokenize the lemmatized reviews and plot the most common words.

After lemmatizing the reviews let take a look at the most frequent word.



It seems that now most frequent terms in our data are relevant. We can now go ahead and start building our topic model.

Using a WordCloud to also visualize the most frequent terms

```python
# to create our wordclouds, I will import the python module "wordcloud"

from wordcloud import WordCloud

# The wordcloud of Reviews in the dataset
plt.figure(figsize=(10,7))
wordcloud = WordCloud(background_color="black", max_words=10000,
                      max_font_size= 100)
wordcloud.generate(" ".join(df.reviews))
plt.title("Keywords in Review", fontsize=20)
plt.imshow(wordcloud.recolor(random_state=17), alpha=0.98, interpolation='bilinear'
plt.axis('off')
```

LDA Model

Let's start by creating the term dictionary of our corpus, where every unique term is assigned an index

```python
dictionary = corpora.Dictionary(reviews_2)

# convert the list of reviews (reviews_2) into a Document Term Matrix using the dictionary prepared above.
doc_term_matrix = [dictionary.doc2bow(rev) for rev in reviews_2]
```

```python
# Creating the object for LDA model using gensim library
LDA = gensim.models.ldamodel.LdaModel

# Build LDA model with 7 topics
lda_model = LDA(corpus=doc_term_matrix, id2word=dictionary, num_topics=7, random_state=100)
```

Topics that our LDA model has learned

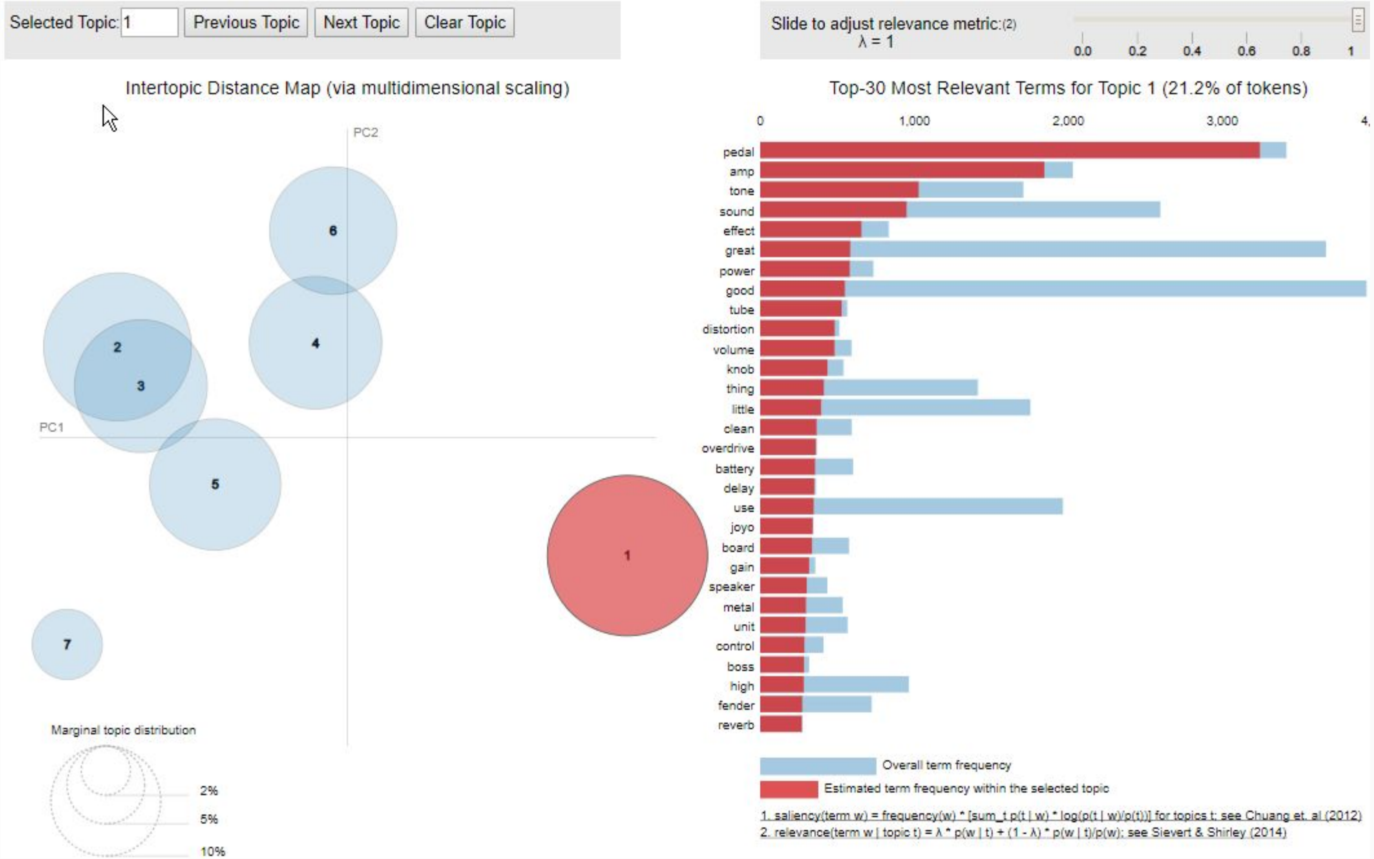
```
# Let's print out the topics that our LDA model has learned.

lda_model.print_topics()
```

```
[(0,
  '0.059*"pedal" + 0.033*"amp" + 0.019*"tone" + 0.017*"sound" + 0.012*"effect" + 0.011*"great" + 0.011*"power" + 0.010*"good" +
0.010*"tube" + 0.009*"distortion"'),
 (1,
  '0.027*"pick" + 0.025*"guitar" + 0.021*"great" + 0.019*"good" + 0.013*"case" + 0.010*"sound" + 0.010*"string" + 0.009*"produc
t" + 0.008*"price" + 0.008*"use"'),
 (2,
  '0.031*"cable" + 0.029*"great" + 0.025*"good" + 0.019*"price" + 0.019*"quality" + 0.011*"sound" + 0.010*"guitar" + 0.009*"nic
e" + 0.008*"product" + 0.008*"use"'),
 (3,
  '0.042*"guitar" + 0.038*"tuner" + 0.015*"easy" + 0.014*"strap" + 0.013*"tune" + 0.011*"capo" + 0.011*"use" + 0.010*"good" +
0.010*"little" + 0.010*"instrument"'),
 (4,
  '0.025*"mic" + 0.014*"microphone" + 0.012*"good" + 0.010*"usb" + 0.009*"use" + 0.009*"great" + 0.008*"display" + 0.008*"recor
ding" + 0.008*"sound" + 0.008*"guitar"'),
 (5,
  '0.024*"time" + 0.018*"guitar" + 0.015*"little" + 0.014*"violin" + 0.012*"pick" + 0.011*"jam" + 0.010*"nice" + 0.009*"thing"
+ 0.008*"strap" + 0.008*"mandolin"'),
 (6,
  '0.070*"string" + 0.032*"guitar" + 0.020*"good" + 0.013*"sound" + 0.012*"great" + 0.011*"time" + 0.009*"light" + 0.008*"tune"
+ 0.008*"instrument" + 0.008*"tone"')]
```

Based on topics 1 and 6 terms like guitar, strap and tuner indicate the music instrument that is being reviewed is a guitar. Topics 3 and 5 seems to refer to the overall quality with terms like good and great.

Visualization

To visualize our topics in a 2-dimensional space we will use the pyLDAvis library. This visualization is interactive in nature and displays topics along with the most relevant words



Sentiment Analysis

Sentiment analysis is part of the Natural Language Processing (NLP) techniques that consists in extracting emotions related to some raw texts. This is usually used on customer reviews in order to automatically understand if some users are positive or negative and why.

The goal of this section is to show how sentiment analysis can be performed using python.

Let's create 2 columns for the Sentiment analysis

- Column for sentiments
- Column for number of characters

```python

# add sentiment anaylsis columns
from nltk.sentiment.vader import SentimentIntensityAnalyzer

sid = SentimentIntensityAnalyzer()
df['sentiments'] = df['reviewText'].apply(lambda x: sid.polarity_scores(x))
df = pd.concat([df.drop(['sentiments'], axis=1), df['sentiments'].apply(pd.Series)], axis=1)

```

```python
# add number of characters column
df['nb_chars'] = df['reviewText'].apply(lambda x: len(x))

# add number of words column
df['nb_words'] = df['reviewText'].apply(lambda x: len(x.split(" ")))
```

Lets create a Doc2vec vector column, train the Doc2vec model, transform each document into a vector data and then add tf-idf column.

```python
from gensim.test.utils import common_texts
from gensim.models.doc2vec import Doc2Vec, TaggedDocument

documents = [TaggedDocument(doc, [i]) for i, doc in enumerate(df['reviews'].apply(lambda x: x.split(" ")))]

# Lets train a Doc2Vec model with our text data
model = Doc2Vec(documents, vector_size=5, window=2, min_count=1, workers=4)

# transform each document into a vector data
doc2vec_df = df['reviews'].apply(lambda x: model.infer_vector(x.split(" "))).apply(pd.Series)
doc2vec_df.columns = ['doc2vec_vector_' + str(x) for x in doc2vec_df.columns]
df = pd.concat([df, doc2vec_df], axis=1)

```

```python
# add tf-idfs columns
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(min_df = 10)
tfidf_result = tfidf.fit_transform(df['reviews']).toarray()
tfidf_df = pd.DataFrame(tfidf_result, columns = tfidf.get_feature_names())
tfidf_df.columns = ['word_' + str(x) for x in tfidf_df.columns]
tfidf_df.index = df.index
df = pd.concat([df, tfidf_df], axis=1)
```

Now lets create a dataset with the highest positive sentiment reviews (with more than 5 words)

```
# highest positive sentiment reviews (with more than 5 words)
df[df['nb_words'] >= 5].sort_values('pos', ascending = False)[['reviews', 'pos']].head(10)
```

	reviews	pos
5945	nice good fine great fantastic love nice good ...	0.956
7509	nice awesome cool ido love	0.755
1481	great great great great great great great grea...	0.746
7605	great strap nice comfort nice pricefit nice fe...	0.717
666	love thing action great sound good easy play r...	0.716
7934	great microphone good rice	0.709
2519	excellent product strong work excellent condit...	0.703
2854	excellent string beautiful solid tone buzzing	0.697
10256	great thank	0.688
1192	easy pack easy instrument light weight sturdy ...	0.683

As expected, the most positive reviews indeed correspond to some good feedbacks.

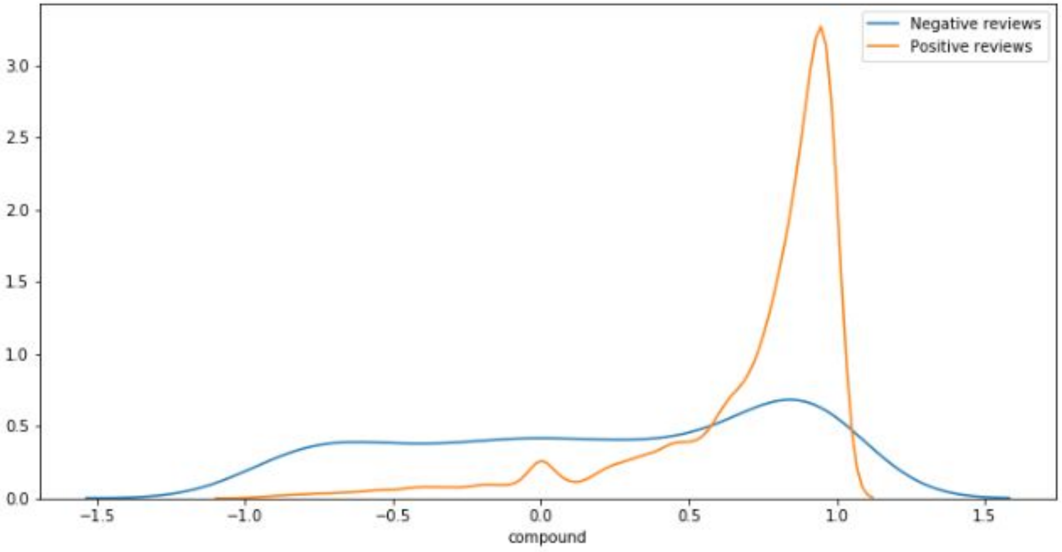
Now lets create a dataset with the highest negative sentiment reviews (with more than 5 words)

```
# lowest negative sentiment reviews (with more than 5 words)
df[df['nb_words'] >= 5].sort_values('neg', ascending = False)[['reviews', 'neg']].head(10)
```

	reviews	neg
2424	cable less month disappointed quality	0.466
1781	problem able use abuse	0.433
8802	tank couple week time trouble shoot sure compo...	0.433
8553	dull inept version tubescreamer flat	0.397
7165	gig frustrating guess pay	0.387
3697	good nothing special couple review bad smell b...	0.365
9101	cool large inconvenient blue yeti	0.364
5441	good ideal big problem tube amp twin reverb fe...	0.362
9697	dull uninspiring pedal don waste time boss dig...	0.362
3664	bad brother guitar year	0.355

Let’s plot the sentiment distribution for positive and negative reviews

```python
import seaborn as sns

for x in [0, 1]:
    subset = df[df['rating'] == x]

    # Draw the density plot
    if x == 1:
        label = 'Positive reviews'
    else:
        label = 'Negative reviews'
    sns.distplot(subset['compound'], hist = False, label = label)
```



Predictive Modeling

The dataset is unbalanced and can therefore create false accuracy in the predictive models. To overcome this we can Oversample the data or Undersample the data.

Despite the advantage of balancing classes, these techniques also have their weaknesses. The simplest implementation of over-sampling is to duplicate random records from the minority class, which can cause overfitting. Under-sampling can cause loss of information because we have so few reviews with negative rating.

In this case, we would balance the data with both methods to determine which is most accurate.

Oversampling

```python
# Determine the max size of the larger rating group
max_size = df['rating'].value_counts().max()
```

```python
max_size
print('Max size of the rating:', max_size)
```

```
Max size of the rating: 8998
```

```python
# Lets do some oversampling to take care of the clas imbalance and create a subset.
lst = [df]
for class_index, group in df.groupby('rating'):
    lst.append(group.sample(max_size-len(group), replace=True))
df_subset = pd.concat(lst)
df_subset.head()
```

	asin	helpful	overall	reviewText	reviewTime	reviewerID	reviewerName	summary	unixReviewTime	rating	...
0	1384719342	[0, 0]	5	Not much write about here but does exactly wha...	02 28, 2014	A2IBPI20UZIR0U	cassandra tu "Yeah, well, that's just like, u...	good	1393545600	1	...
				The							

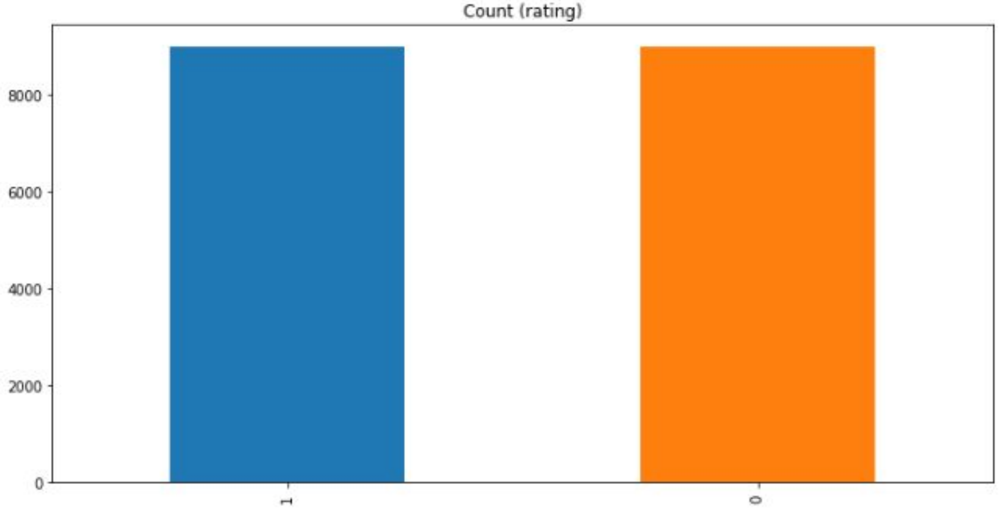
Let's plot the number of positive and negative reviews based on ratings after the Oversampling

```
# Number of positive and neative reviews after the oversampling
df_subset['rating'].value_counts()
```

```
1    8998
0    8998
Name: rating, dtype: int64
```

```
# Lets plot the ratings after the oversampling

rating_count = df_subset.rating.value_counts()

print('Negative Rating:', rating_count[0])
print('Positive Rating:', rating_count[1])
print('Proportion:', round(rating_count[0] / rating_count[1], 2), ': 1')

rating_count.plot(kind='bar', title='Count (rating)');
```

```
Negative Rating: 8998
Positive Rating: 8998
Proportion: 1.0 : 1
```



Modeling with Multinomial Naive Bayes

In this model we will create a feature using **CountVectorizer**

```python
# generate document term matrix by using scikit-learn's CountVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import RegexpTokenizer

#tokenizer to remove unwanted elements from out data like symbols and numbers
token = RegexpTokenizer(r'[a-zA-Z0-9]+')
vect = CountVectorizer(lowercase=True,stop_words='english',ngram_range = (1,1),tokenizer = token.tokenize)
text_counts= vect.fit_transform(df_subset['reviewText'])
```

```python
from sklearn.model_selection import train_test_split
# Split data into training and test sets

X_train, X_test, y_train, y_test = train_test_split(text_counts,
                                                    df_subset['rating'], test_size=0.3, random_state=1)
```

```python
# fit your model on a train set using fit() and perform prediction on the test set using predict().

from sklearn.naive_bayes import MultinomialNB

#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Generation Using Multinomial Naive Bayes
clf = MultinomialNB().fit(X_train, y_train)
predictions= clf.predict(X_test)
print("MultinomialNB Accuracy:",metrics.accuracy_score(y_test, predictions))
print('ROC_AUC: ', roc_auc_score(y_test, predictions))
```

```
MultinomialNB Accuracy: 0.9525838118170031
ROC_AUC:  0.952626089629581
```

Modeling with Multinomial Naive Bayes

In this model we will create a feature using **TF-IDF**

```
from sklearn.feature_extraction.text import TfidfVectorizer
tf=TfidfVectorizer()
text_tf= tf.fit_transform(df_subset['reviewText'])
```

```
# Split data into training and test sets

X_train, X_test, y_train, y_test = train_test_split(text_tf,
                                                    df_subset['rating'], test_size=0.3, random_state=1)
```

```
# Model Building and Evaluation (TF-IDF)
clf = MultinomialNB().fit(X_train, y_train)
predictions= clf.predict(X_test)
print("MultinomialNB Accuracy:",metrics.accuracy_score(y_test, predictions))
print('ROC_AUC: ', roc_auc_score(y_test, predictions))
```

```
MultinomialNB Accuracy: 0.9511020559362845
ROC_AUC:  0.9513608704960148
```

Modeling with Logistic Regression

In this model we will create a feature using **CountVectorizer**

```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df_subset['reviewText'],
                                                    df_subset['rating'], test_size=0.3, random_state=1)

```

```python
vect = CountVectorizer(min_df=5, ngram_range=(1,2)).fit(X_train)

X_train_vectorized = vect.transform(X_train)

len(vect.get_feature_names())
```

```
38792
```

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score

model = LogisticRegression()
model.fit(X_train_vectorized, y_train)

predictions = model.predict(vect.transform(X_test))
print("Accuracy:",metrics.accuracy_score(y_test, predictions))
print('ROC_AUC: ', roc_auc_score(y_test, predictions))
```

```
Accuracy: 0.9957399518429338
ROC_AUC:  0.9957720588235295
```

As you can see, Logistic regression produces the most accurate model. We are going to stick with the logistic regression model.

Evaluating the logistic regression model with Sensitivity/ Recall and F1 Score.

```python
# Lets get the F1 score and the recall score
from sklearn.metrics import f1_score, recall_score
f1_score = round(f1_score(y_test, predictions), 2)
recall_score = round(recall_score(y_test, predictions), 2)
print("Sensitivity/Recall for Logistic Regression Model 1 : {recall_score}".format(recall_score = recall_score))
print("F1 Score for Logistic Regression Model 1 : {f1_score}".format(f1_score = f1_score))
```

```
Sensitivity/Recall for Logistic Regression Model 1 : 0.99
F1 Score for Logistic Regression Model 1 : 1.0
```

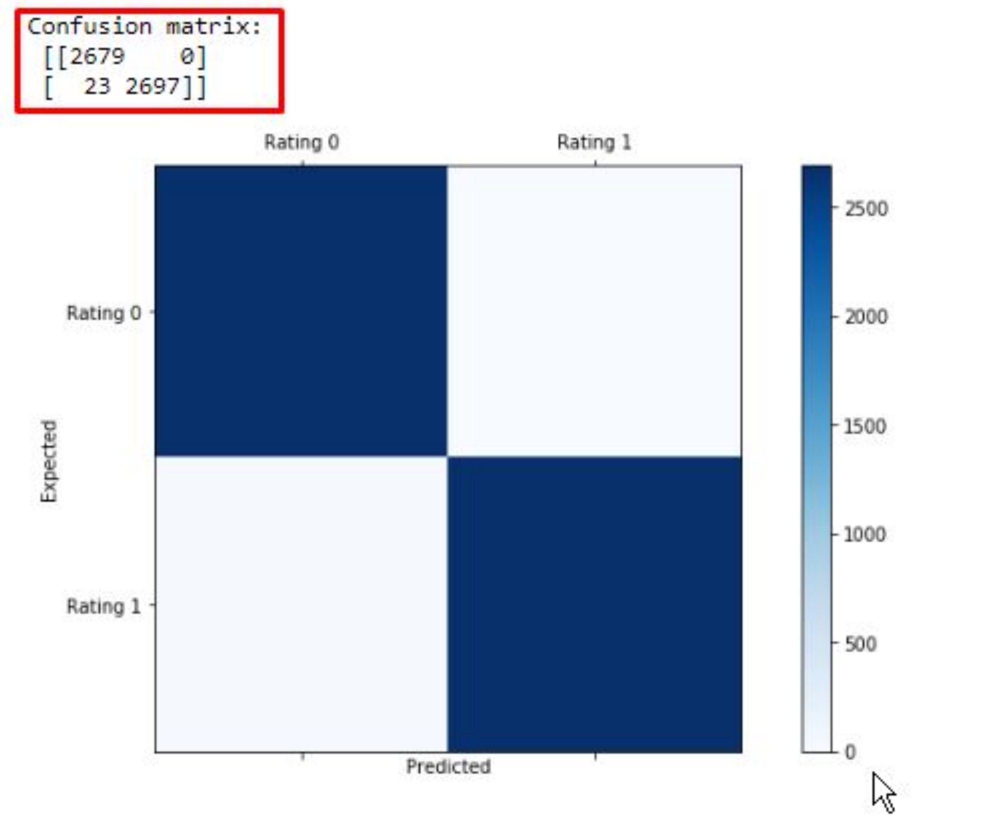
```
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt

conf_mat = confusion_matrix(y_true=y_test, y_pred=predictions)
print('Confusion matrix:\n', conf_mat)

labels = ['Rating 0', 'Rating 1']
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(conf_mat, cmap=plt.cm.Blues)
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Expected')
plt.show()
```

Confusion matrix

An interesting way to evaluate the results is by means of a confusion matrix, which shows the correct and incorrect predictions for each class. In the first row, the first column indicates how many classes 0 were predicted correctly, and the second column, how many classes 0 were predicted as 1. In the second row, we note that all class 1 entries were erroneously predicted as class 0.

Visualization of the Confusion Matrix

The higher the diagonal values of the confusion matrix the better, indicating many correct predictions



Undersampling for comparison

```
# Find Number of samples which are negative reviews
p_rating = len(df[df['rating'] == 0])
# Get indices of positive reviews
p_rating_indices = df[df.rating == 1].index
# Random sample positive reviews indices
random_indices = np.random.choice(p_rating_indices,p_rating, replace=False)
#Find the indices of negative reviews samples
n_rating_indices = df[df.rating == 0].index
# Concat negative review indices with sample positive review ones
under_sample_indices = np.concatenate([n_rating_indices,random_indices])
#Get Balance Dataframe
under_sample = df.loc[under_sample_indices]
under_sample.head()
```

	asin	helpful	overall	reviewText	reviewTime	reviewerID	reviewerName	summary	unixReviewTime	rating	length
15	B00005ML71	[0, 0]	2	bought this use with keyboard wasn really awar...	08 17, 2013	A2PD27UKAD3Q00	Wilhelmina Zeitgeist "coolartsybabe"	Definitely Not For The Seasoned Piano Player	1376697600	0	623
50	B000068NW5	[2, 2]	2	didn expect this cable thin easily the thickne...	07 6, 2011	A12ABV9NU02O29	C. Longo	Cannot recommend	1309910400	0	387
52	B000068NW5	[0, 0]	1	hums crackles and think having problems with e...	02 9, 2014	A1L7M2JXN4EZCR	David G	I have bought many cables and this one is the ...	1391904000	0	245

Let's plot the number of positive and negative reviews based on ratings after the Undersampling

```python
rating_count_under = under_sample.rating.value_counts()

print('Negative Rating :', rating_count_under[0])
print('Positive Rating :', rating_count_under[1])
print('Proportion:', round(rating_count_under[0] / rating_count_under[1], 2), ': 1')

rating_count_under.plot(kind='bar', title='Count (rating)');
```

```
Negative Rating : 465
Positive Rating : 465
Proportion: 1.0 : 1
```

Modeling with Logistic Regression

In this model we will create a feature using **CountVectorizer**

```
from sklearn.model_selection import train_test_split

X_train_under, X_test_under, y_train_under, y_test_under = train_test_split(under_sample['reviewText'],
                                                                            under_sample['rating'], test_size=0.3, random_state=1)

```

```
vect = CountVectorizer(min_df=5, ngram_range=(1,2)).fit(X_train_under)

X_train_vectorized_under= vect.transform(X_train_under)

len(vect.get_feature_names())
```

```
2571
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
from sklearn import metrics

model_under = LogisticRegression()
model_under.fit(X_train_vectorized_under, y_train_under)

predictions_under = model_under.predict(vect.transform(X_test_under))
print("Accuracy:",metrics.accuracy_score(y_test_under, predictions_under))
print('ROC_AUC: ', roc_auc_score(y_test_under, predictions_under))
```

```
Accuracy: 0.7455197132616488
ROC_AUC:  0.7452290076335878
```

Evaluating the logistic regression model with Sensitivity/ Recall and F1 Score.

```
from sklearn.metrics import f1_score, recall_score
f1_score_under = round(f1_score(y_test_under, predictions_under), 2)
recall_score_under = round(recall_score(y_test_under, predictions_under), 2)
print("Sensitivity/Recall for Logistic Regression Model 1 : {recall_score_under}".format(recall_score_under = recall_score_
print("F1 Score for Logistic Regression Model 1 : {f1_score_under}".format(f1_score_under = f1_score_under))
```

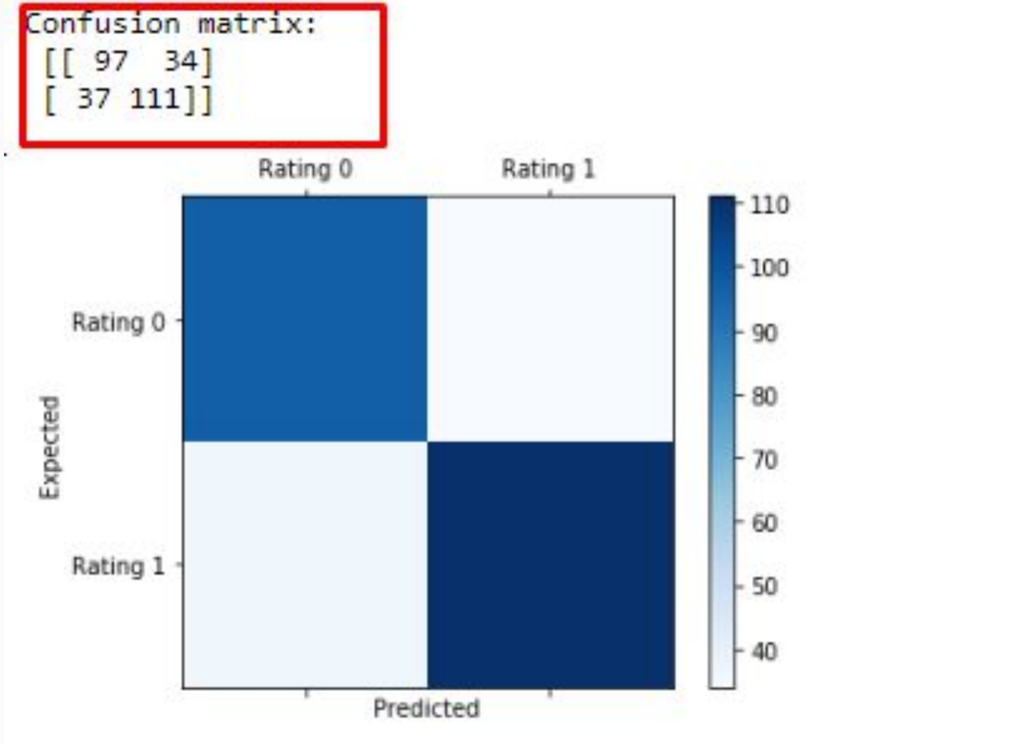
```
Sensitivity/Recall for Logistic Regression Model 1 : 0.75
F1 Score for Logistic Regression Model 1 : 0.76
```

```python
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt

conf_mat = confusion_matrix(y_true=y_test_under, y_pred=predictions_under)
print('Confusion matrix:\n', conf_mat)

labels = ['Rating 0', 'Rating 1']
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(conf_mat, cmap=plt.cm.Blues)
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Expected')
plt.show()
```

Confusion matrix

Visualization of the Confusion Matrix

The higher the diagonal values of the confusion matrix the better, indicating many correct predictions



Conclusion

Online product reviews are a great source of information. From the business/sellers’ point of view. Online reviews can be used to gauge the consumers’ feedback on the products or services they are selling. However, since these online reviews are quite often overwhelming in terms of numbers and information, an intelligent system, capable of finding key insights (topics) from these reviews, will be of great help for both the consumers and the sellers.

We explored the use of data visualization techniques common in data science, such as histograms and pyPlots, to gain a better understanding of the underlying distribution of data in our data set.

As demonstrated above, the dataset had reviews of a musical equipment which included a guitar, microphone and possibly an amp. The sentiment of the the reviews were mostly positive, and the positive reviews were longer in text than negative reviews.

When modeling a dataset with highly unbalanced classes such as this, the classifier always "predicts" the most common class without performing any analysis of the features, which will have a high accuracy rate, which is obviously inaccurate. Oversampling and undersampling was done to determine which would produce a more reliable accuracy rate. As one would predict, oversampling produced the better scores using metrics such as precision, recall, and F1-scores, as well as confusion matrix.

“Information is the oil of the 21st century, and analytics is the combustion engine”

- Peter Sondergaard, Senior Vice President at Gartner

Thanks!

Femi Sulyman

https://github.com/femi18/Projects
Femisu@gmail.com