```
#Basic libraries
import pandas as pd
import numpy as np


#NLTK libraries
import nltk
import re
import string
from wordcloud import WordCloud,STOPWORDS
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer

# Machine Learning libraries
import sklearn
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize
from sklearn import svm, datasets
from sklearn import preprocessing


#Metrics libraries
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve, auc

#Visualization libraries
import matplotlib.pyplot as plt
from matplotlib import rcParams
import seaborn as sns
from textblob import TextBlob
from plotly import tools
import plotly.graph_objs as go
```

```
import plotly.graph_objs as go
from plotly.offline import iplot
%matplotlib inline

#Ignore warnings
import warnings
warnings.filterwarnings('ignore')

#Other miscellaneous libraries
from scipy import interp
from itertools import cycle
import cufflinks as cf
from collections import defaultdict
from collections import Counter
from imblearn.over_sampling import SMOTE


raw_reviews = pd.read_csv("/content/Musical_instruments_reviews.csv")
## print shape of dataset with rows and columns and information
print ("The shape of the  data is (row, column):"+ str(raw_reviews.shape))
print (raw_reviews.info())
```

```
The shape of the  data is (row, column):(10261, 9)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10261 entries, 0 to 10260
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   reviewerID      10261 non-null  object 
 1   asin            10261 non-null  object 
 2   reviewerName    10234 non-null  object 
 3   helpful         10261 non-null  object 
 4   reviewText      10254 non-null  object 
 5   overall         10261 non-null  float64
 6   summary         10261 non-null  object 
 7   unixReviewTime  10261 non-null  int64  
 8   reviewTime      10261 non-null  object 
dtypes: float64(1), int64(1), object(7)
memory usage: 721.6+ KB
None
```

```
raw_reviews.head()
```

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime
0	A2IBPI20UZIR0U	1384719342	cassandra tu "Yeah, well, that's just like, u...	[0, 0]	Not much to write about here, but it does exac...	5.0	good	1393545600	02 28, 2014
1	A14VAT5EAX3D9S	1384719342	Jake	[13, 14]	The product does exactly as it should and is q...	5.0	Jake	1363392000	03 16, 2013
2	A195EZSQDW3E21	1384719342	Rick Bennette "Rick Bennette"	[1, 1]	The primary job of this device is to block the...	5.0	It Does The Job Well	1377648000	08 28, 2013
					Nice				

```
#Creating a copy
process_reviews=raw_reviews.copy()

#Checking for null values
process_reviews.isnull().sum()
```

```
reviewerID         0
asin               0
reviewerName      27
helpful            0
reviewText         7
overall            0
summary            0
unixReviewTime     0
reviewTime         0
dtype: int64
```

```
process_reviews['reviewText']=process_reviews['reviewText'].fillna('Missing')


process_reviews['reviews']=process_reviews['reviewText']+process_reviews['summary']
process_reviews=process_reviews.drop(['reviewText', 'summary'], axis=1)
process_reviews.head()
```

	reviewerID	asin	reviewerName	helpful	overall	unixReviewTime	reviewTime	reviews
0	A2IBPI20UZIR0U	1384719342	cassandra tu "Yeah, well, that's just like, u...	[0, 0]	5.0	1393545600	02 28, 2014	Not much to write about here, but it does exac...
1	A14VAT5EAX3D9S	1384719342	Jake	[13, 14]	5.0	1363392000	03 16, 2013	The product does exactly as it should and is q...
2	A195EZSQDW3E21	1384719342	Rick Bennette "Rick Bennette"	[1, 1]	5.0	1377648000	08 28, 2013	The primary job of this device is to block the...
			RustyBill "Sunday					Nice windscreen protects

```
#Figuring out the distribution of categories
process_reviews['overall'].value_counts()
```

```
5.0    6938
4.0    2084
3.0     772
2.0     250
1.0     217
Name: overall, dtype: int64
```

```
def f(row):

    '''This function returns sentiment value based on the overall ratings from the user'''

    if row['overall'] == 3.0:
        val = 'Neutral'
    elif row['overall'] == 1.0 or row['overall'] == 2.0:
        val = 'Negative'
    elif row['overall'] == 4.0 or row['overall'] == 5.0:
        val = 'Positive'
    else:
        val = -1
    return val
```

```
#Applying the function in our new column
process_reviews['sentiment'] = process_reviews.apply(f, axis=1)
process_reviews.head()
```

	reviewerID	asin	reviewerName	helpful	overall	unixReviewTime	reviewTime	reviews	sentiment
0	A2IBPI20UZIR0U	1384719342	cassandra tu "Yeah, well, that's just like, u...	[0, 0]	5.0	1393545600	02 28, 2014	Not much to write about here, but it does exac...	Positive
1	A14VAT5EAX3D9S	1384719342	Jake	[13, 14]	5.0	1363392000	03 16, 2013	The product does exactly as it should and is q...	Positive
2	A195EZSQDW3E21	1384719342	Rick Bennette "Rick	[1, 1]	5.0	1377648000	08 28, 2013	The primary job of this device is to	Positive

```
process_reviews['sentiment'].value_counts()
```

```
Positive    9022
Neutral      772
Negative     467
Name: sentiment, dtype: int64
```

```
# new data frame which has date and year
new = process_reviews["reviewTime"].str.split(",", n = 1, expand = True)

# making separate date column from new data frame
process_reviews["date"]= new[0]

# making separate year column from new data frame
process_reviews["year"]= new[1]

process_reviews=process_reviews.drop(['reviewTime'], axis=1)
process_reviews.head()
```

	reviewerID	asin	reviewerName	helpful	overall	unixReviewTime	reviews	sentiment	date	year
0	A2IBPI20UZIR0U	1384719342	cassandra tu "Yeah, well, that's just like, u...	[0, 0]	5.0	1393545600	Not much to write about here, but it does exac...	Positive	02 28	2014
1	A14VAT5EAX3D9S	1384719342	Jake	[13, 14]	5.0	1363392000	The product does exactly as it should and is q...	Positive	03 16	2013
2	A195EZSQDW3E21	1384719342	Rick Bennette "Rick	[1, 1]	5.0	1377648000	The primary job of this device is to	Positive	08	2013

```
# Splitting the date
new1 = process_reviews["date"].str.split(" ", n = 1, expand = True)

# adding month to the main dataset
process_reviews["month"]= new1[0]

# adding day to the main dataset
process_reviews["day"]= new1[1]

process_reviews=process_reviews.drop(['date'], axis=1)
process_reviews.head()
```

	reviewerID	asin	reviewerName	helpful	overall	unixReviewTime	reviews	sentiment	year	month	day
0	A2IBPI20UZIR0U	1384719342	cassandra tu "Yeah, well, that's just like, u...	[0, 0]	5.0	1393545600	Not much to write about here, but it does exac...	Positive	2014	02	28
1	A14VAT5EAX3D9S	1384719342	Jake	[13, 14]	5.0	1363392000	The product does exactly as it should and is q...	Positive	2013	03	16

```
# Splitting the dataset based on comma and square bracket
new1 = process_reviews["helpful"].str.split(",", n = 1, expand = True)
new2 = new1[0].str.split("[", n = 1, expand = True)
new3 = new1[1].str.split("]", n = 1, expand = True)

#Resetting the index
new2.reset_index(drop=True, inplace=True)
new3.reset_index(drop=True, inplace=True)

#Dropping empty columns due to splitting
new2=new2.drop([0], axis=1)
new3=new3.drop([1], axis=1)

#Concatenating the splitted columns
helpful=pd.concat([new2, new3], axis=1)


# I found few spaces in new3, so it is better to strip all the values to find the rate
def trim_all_columns(df):
    """
    Trim whitespace from ends of each value across all series in dataframe
    """
    trim_strings = lambda x: x.strip() if isinstance(x, str) else x
    return df.applymap(trim_strings)

#Applying the function
helpful= trim_all_columns(helpful)

#Converting into integer types
helpful[0]=helpful[0].astype(str).astype(int)
helpful[1]=helpful[1].astype(str).astype(int)

#Dividing the two columns, we have 0 in the second columns when dvided gives error, so I'm ignoring those errors
try:
  helpful['result'] = helpful[1]/helpful[0]
except ZeroDivisionError:
  helpful['result']=0

#Filling the NaN values(created due to dividing) with 0
helpful['result'] = helpful['result'].fillna(0)

#Rounding of the results to two decimal places
helpful['result']=helpful['result'].round(2)

#Attaching the results to a new column of the main dataframe
process_reviews['helpful_rate']=helpful['result']

#dropping the helpful column from main dataframe
process_reviews=process_reviews.drop(['helpful'], axis=1)
```

```
process_reviews.head()
```

	reviewerID	asin	reviewerName	overall	unixReviewTime	reviews	sentiment	year	month	day	helpful_rate
0	A2IBPI20UZIR0U	1384719342	cassandra tu "Yeah, well, that's just like, u...	5.0	1393545600	Not much to write about here, but it does exac...	Positive	2014	02	28	0.00
1	A14VAT5EAX3D9S	1384719342	Jake	5.0	1363392000	The product does exactly as it	Positive	2013	03	16	0.93

```
process_reviews['helpful_rate'].value_counts()
```

```
0.00    7215
1.00    2040
0.50     266
0.67     136
0.75     111
        ...
0.56       1
0.15       1
0.13       1
0.43       1
0.69       1
Name: helpful_rate, Length: 65, dtype: int64
```

```
#Removing unnecessary columns
process_reviews=process_reviews.drop(['reviewerName','unixReviewTime'], axis=1)
#Creating a copy
clean_reviews=process_reviews.copy()


def review_cleaning(text):
    '''Make text lowercase, remove text in square brackets,remove links,remove punctuation
    and remove words containing numbers.'''
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text


process_reviews['reviews']=process_reviews['reviews'].apply(lambda x:review_cleaning(x))
process_reviews.head()
```

	reviewerID	asin	overall	reviews	sentiment	year	month	day	helpful_rate
0	A2IBPI20UZIR0U	1384719342	5.0	not much to write about here but it does exact...	Positive	2014	02	28	0.00
1	A14VAT5EAX3D9S	1384719342	5.0	the product does exactly as it should and is q...	Positive	2013	03	16	0.93
2	A195EZSQDW3E21	1384719342	5.0	the primary job of this device is to block the...	Positive	2013	08	28	1.00
3	A2C00NNG1ZQQG2	1384719342	5.0	nice windscreen protects my mxl mic and preven...	Positive	2014	02	14	0.00
4	A94QU4C90B1AX	1384719342	5.0	this pop filter is great it looks and performs...	Positive	2014	02	21	0.00

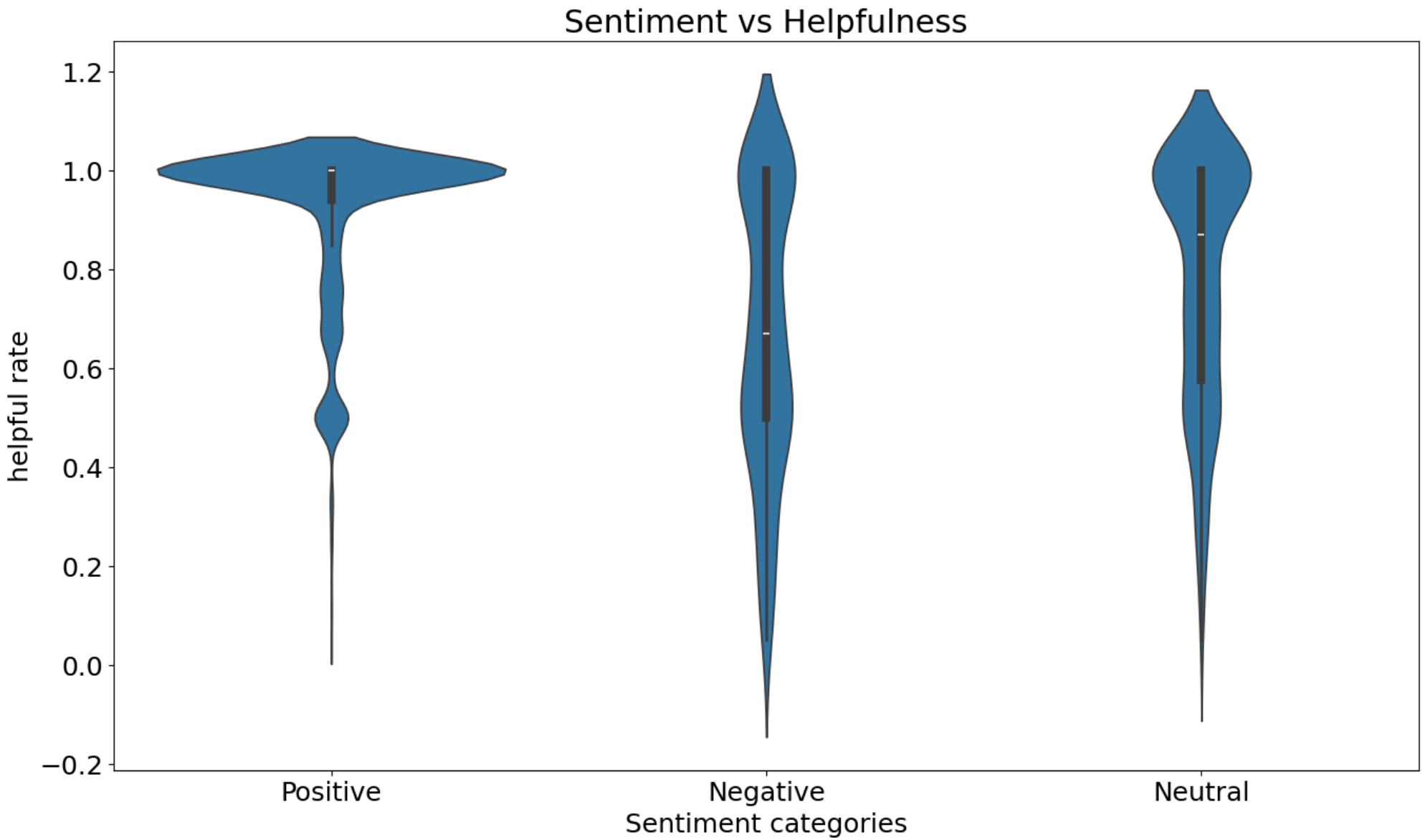
```
stop_words= ['yourselves', 'between', 'whom', 'itself', 'is', "she's", 'up', 'herself', 'here', 'your', 'each',
             'we', 'he', 'my', "you've", 'having', 'in', 'both', 'for', 'themselves', 'are', 'them', 'other',
             'and', 'an', 'during', 'their', 'can', 'yourself', 'she', 'until', 'so', 'these', 'ours', 'above',
             'what', 'while', 'have', 're', 'more', 'only', "needn't", 'when', 'just', 'that', 'were', "don't",
             'very', 'should', 'any', 'y', 'isn', 'who',  'a', 'they', 'to', 'too', "should've", 'has', 'before',
             'into', 'yours', "it's", 'do', 'against', 'on',  'now', 'her', 've', 'd', 'by', 'am', 'from',
             'about', 'further', "that'll", "you'd", 'you', 'as', 'how', 'been', 'the', 'or', 'doing', 'such',
             'his', 'himself', 'ourselves',  'was', 'through', 'out', 'below', 'own', 'myself', 'theirs',
             'me', 'why', 'once',  'him', 'than', 'be', 'most', "you'll", 'same', 'some', 'with', 'few', 'it',
             'at', 'after', 'its', 'which', 'there','our', 'this', 'hers', 'being', 'did', 'of', 'had', 'under',
             'over','again', 'where', 'those', 'then', "you're", 'i', 'because', 'does', 'all']


process_reviews['reviews'] = process_reviews['reviews'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_words)]))
process_reviews.head()
```

	reviewerID	asin	overall	reviews	sentiment	year	month	day	helpful_rate
0	A2IBPI20UZIR0U	1384719342	5.0	not much write but exactly supposed filters po...	Positive	2014	02	28	0.00
1	A14VAT5EAX3D9S	1384719342	5.0	product exactly quite affordablei not realized...	Positive	2013	03	16	0.93
2	A195EZSQDW3E21	1384719342	5.0	primary job device block breath would otherwis...	Positive	2013	08	28	1.00
3	A2C00NNG1ZQQG2	1384719342	5.0	nice windscreen protects mxl mic prevents pops...	Positive	2014	02	14	0.00
4	A94QU4C90B1AX	1384719342	5.0	pop filter great looks performs like studio fi...	Positive	2014	02	21	0.00

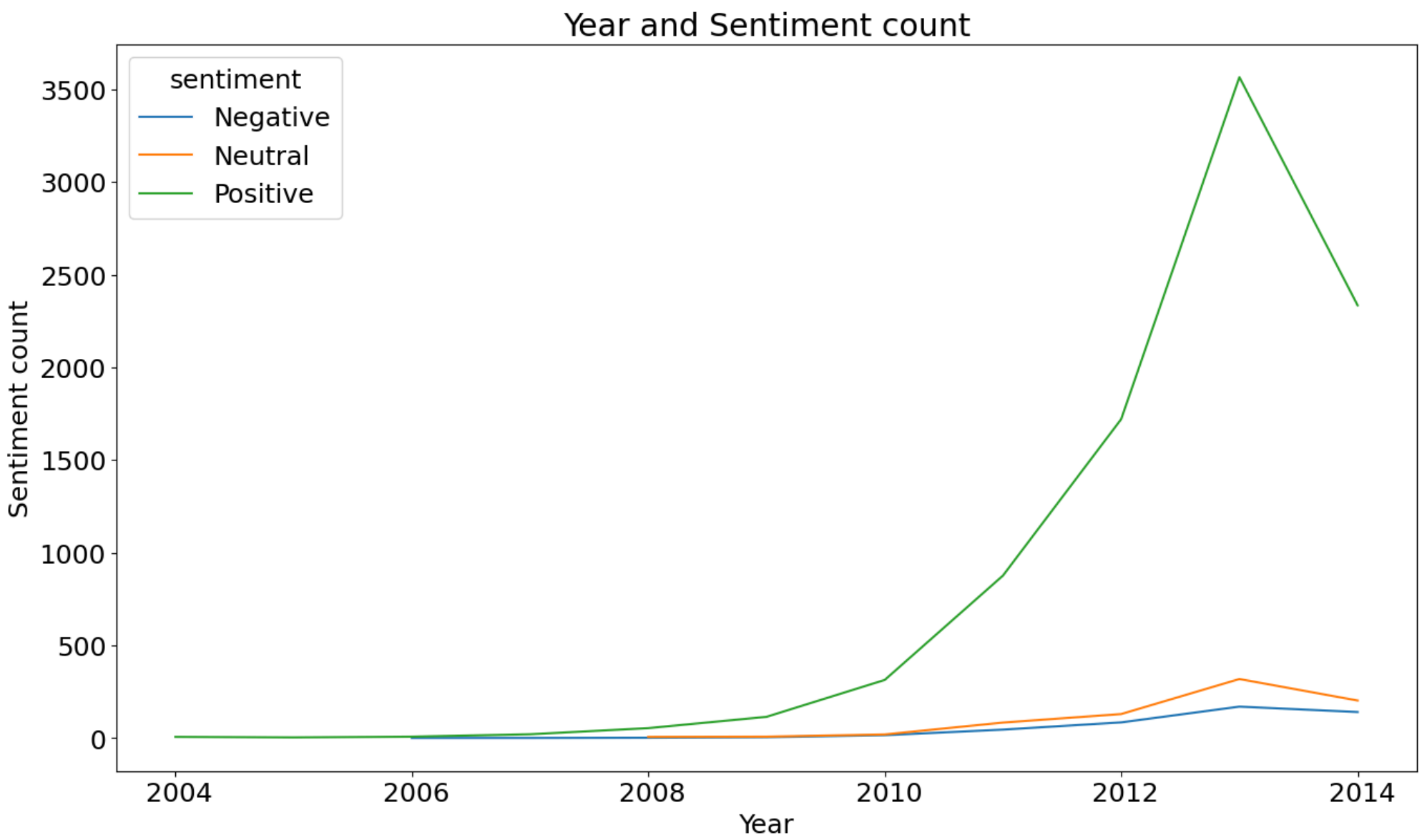
```
pd.DataFrame(process_reviews.groupby('sentiment')['helpful_rate'].mean())
```

sentiment	helpful_rate
Negative	0.307559
Neutral	0.275687
Positive	0.260505

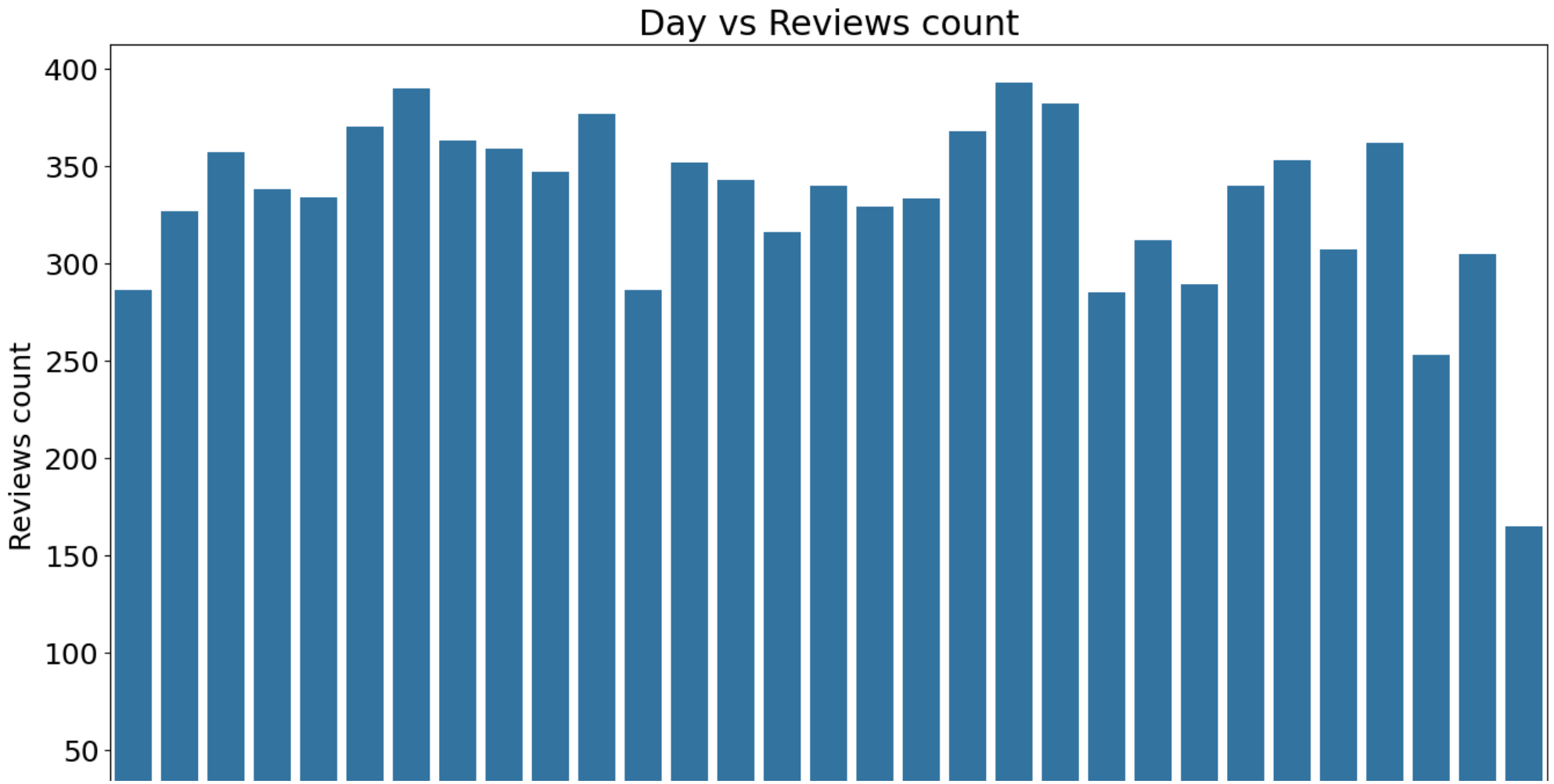
```
#plot layout
plt.rcParams.update({'font.size': 18})
rcParams['figure.figsize'] = 16,9

# Creating dataframe and removing 0 helpfulrate records
senti_help= pd.DataFrame(process_reviews, columns = ['sentiment', 'helpful_rate'])
senti_help = senti_help[senti_help['helpful_rate'] != 0.00]

#Plotting phase
sns.violinplot( x=senti_help["sentiment"], y=senti_help["helpful_rate"])
plt.title('Sentiment vs Helpfulness')
plt.xlabel('Sentiment categories')
plt.ylabel('helpful rate')
plt.show()
```




```
process_reviews.groupby(['year','sentiment'])['sentiment'].count().unstack().plot(legend=True)
plt.title('Year and Sentiment count')
plt.xlabel('Year')
plt.ylabel('Sentiment count')
plt.show()
```


Year and Sentiment count
sentiment
Negative
Neutral
Positive
Sentiment count
0
500
1000
1500
2000
2500
3000
3500
2004
2006
2008
2010
2012
2014
Year

```
#Creating a dataframe
day=pd.DataFrame(process_reviews.groupby('day')['reviews'].count()).reset_index()
day['day']=day['day'].astype('int64')
day.sort_values(by=['day'])

#Plotting the graph
sns.barplot(x="day", y="reviews", data=day)
plt.title('Day vs Reviews count')
plt.xlabel('Day')
plt.ylabel('Reviews count')
plt.show()
```



```
process_reviews['polarity'] = process_reviews['reviews'].map(lambda text: TextBlob(text).sentiment.polarity)
process_reviews['review_len'] = process_reviews['reviews'].astype(str).apply(len)
process_reviews['word_count'] = process_reviews['reviews'].apply(lambda x: len(str(x).split()))
```

```
process_reviews.head()
```

	reviewerID	asin	overall	reviews	sentiment	year	month	day	helpful_rate	polarity	review_len	word_count
0	A2IBPI20UZIR0U	1384719342	5.0	not much write but exactly supposed filters po...	Positive	2014	02	28	0.00	0.250000	162	25
1	A14VAT5EAX3D9S	1384719342	5.0	product exactly quite affordablei not realized...	Positive	2013	03	16	0.93	0.014286	356	55
2	A195EZSQDW3E21	1384719342	5.0	primary job device block breath would otherwis...	Positive	2013	08	28	1.00	0.167500	315	48
3	A2C00NNG1ZQQG2	1384719342	5.0	nice windscreen protects mxl mic prevents pops...	Positive	2014	02	14	0.00	0.333333	169	22
4	A94QU4C90B1AX	1384719342	5.0	pop filter great looks performs like studio fi...	Positive	2014	02	21	0.00	0.800000	136	21

```
cf.go_offline()
cf.set_config_file(offline=False, world_readable=True)
```

```
process_reviews['polarity'].iplot(
    kind='hist',
    bins=50,
    xTitle='polarity',
    linecolor='black',
    yTitle='count',
    title='Sentiment Polarity Distribution')
```

```
process_reviews['overall'].iplot(
    kind='hist',
    xTitle='rating',
    linecolor='black',
    yTitle='count',
    title='Review Rating Distribution')
```

```
process_reviews['review_len'].iplot(
    kind='hist',
    bins=100,
    xTitle='review length',
    linecolor='black',
    yTitle='count',
    title='Review Text Length Distribution')
```

```
process_reviews['word_count'].iplot(
    kind='hist',
    bins=100,
    xTitle='word count',
    linecolor='black',
    yTitle='count',
    title='Review Text Word Count Distribution')
```

```
#Filtering data
review_pos = process_reviews[process_reviews["sentiment"]=='Positive'].dropna()
review_neu = process_reviews[process_reviews["sentiment"]=='Neutral'].dropna()
review_neg = process_reviews[process_reviews["sentiment"]=='Negative'].dropna()

## custom function for ngram generation ##
def generate_ngrams(text, n_gram=1):
    token = [token for token in text.lower().split(" ") if token != "" if token not in STOPWORDS]
    ngrams = zip(*[token[i:] for i in range(n_gram)])
    return [" ".join(ngram) for ngram in ngrams]

## custom function for horizontal bar chart ##
def horizontal_bar_chart(df, color):
    trace = go.Bar(
        y=df["word"].values[::-1],
        x=df["wordcount"].values[::-1],
        showlegend=False,
        orientation = 'h',
        marker=dict(
            color=color,
        ),
    )
    return trace

## Get the bar chart from positive reviews ##
freq_dict = defaultdict(int)
for sent in review_pos["reviews"]:
    for word in generate_ngrams(sent):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
fd_sorted.columns = ["word", "wordcount"]
trace0 = horizontal_bar_chart(fd_sorted.head(25), 'green')

## Get the bar chart from neutral reviews ##
freq_dict = defaultdict(int)
for sent in review_neu["reviews"]:
    for word in generate_ngrams(sent):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
fd_sorted.columns = ["word", "wordcount"]
trace1 = horizontal_bar_chart(fd_sorted.head(25), 'grey')

## Get the bar chart from negative reviews ##
freq_dict = defaultdict(int)
for sent in review_neg["reviews"]:
    for word in generate_ngrams(sent):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
fd_sorted.columns = ["word", "wordcount"]
trace2 = horizontal_bar_chart(fd_sorted.head(25), 'red')
```

```
# Creating two subplots
fig = tools.make_subplots(rows=3, cols=1, vertical_spacing=0.04,
                          subplot_titles=["Frequent words of positive reviews", "Frequent words of neutral reviews",
                                          "Frequent words of negative reviews"])
fig.append_trace(trace0, 1, 1)
fig.append_trace(trace1, 2, 1)
fig.append_trace(trace2, 3, 1)
fig['layout'].update(height=1200, width=900, paper_bgcolor='rgb(233,233,233)', title="Word Count Plots")
iplot(fig, filename='word-plots')
```

```
## Get the bar chart from positive reviews ##
freq_dict = defaultdict(int)
for sent in review_pos["reviews"]:
    for word in generate_ngrams(sent,2):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
fd_sorted.columns = ["word", "wordcount"]
trace0 = horizontal_bar_chart(fd_sorted.head(25), 'green')

## Get the bar chart from neutral reviews ##
freq_dict = defaultdict(int)
for sent in review_neu["reviews"]:
    for word in generate_ngrams(sent,2):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
fd_sorted.columns = ["word", "wordcount"]
trace1 = horizontal_bar_chart(fd_sorted.head(25), 'grey')

## Get the bar chart from negative reviews ##
freq_dict = defaultdict(int)
for sent in review_neg["reviews"]:
    for word in generate_ngrams(sent,2):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
fd_sorted.columns = ["word", "wordcount"]
trace2 = horizontal_bar_chart(fd_sorted.head(25), 'brown')



# Creating two subplots
fig = tools.make_subplots(rows=3, cols=1, vertical_spacing=0.04,horizontal_spacing=0.25,
                          subplot_titles=["Bigram plots of Positive reviews",
                                          "Bigram plots of Neutral reviews",
                                          "Bigram plots of Negative reviews"
                                          ])
fig.append_trace(trace0, 1, 1)
fig.append_trace(trace1, 2, 1)
fig.append_trace(trace2, 3, 1)


fig['layout'].update(height=1000, width=800, paper_bgcolor='rgb(233,233,233)', title="Bigram Plots")
iplot(fig, filename='word-plots')
```

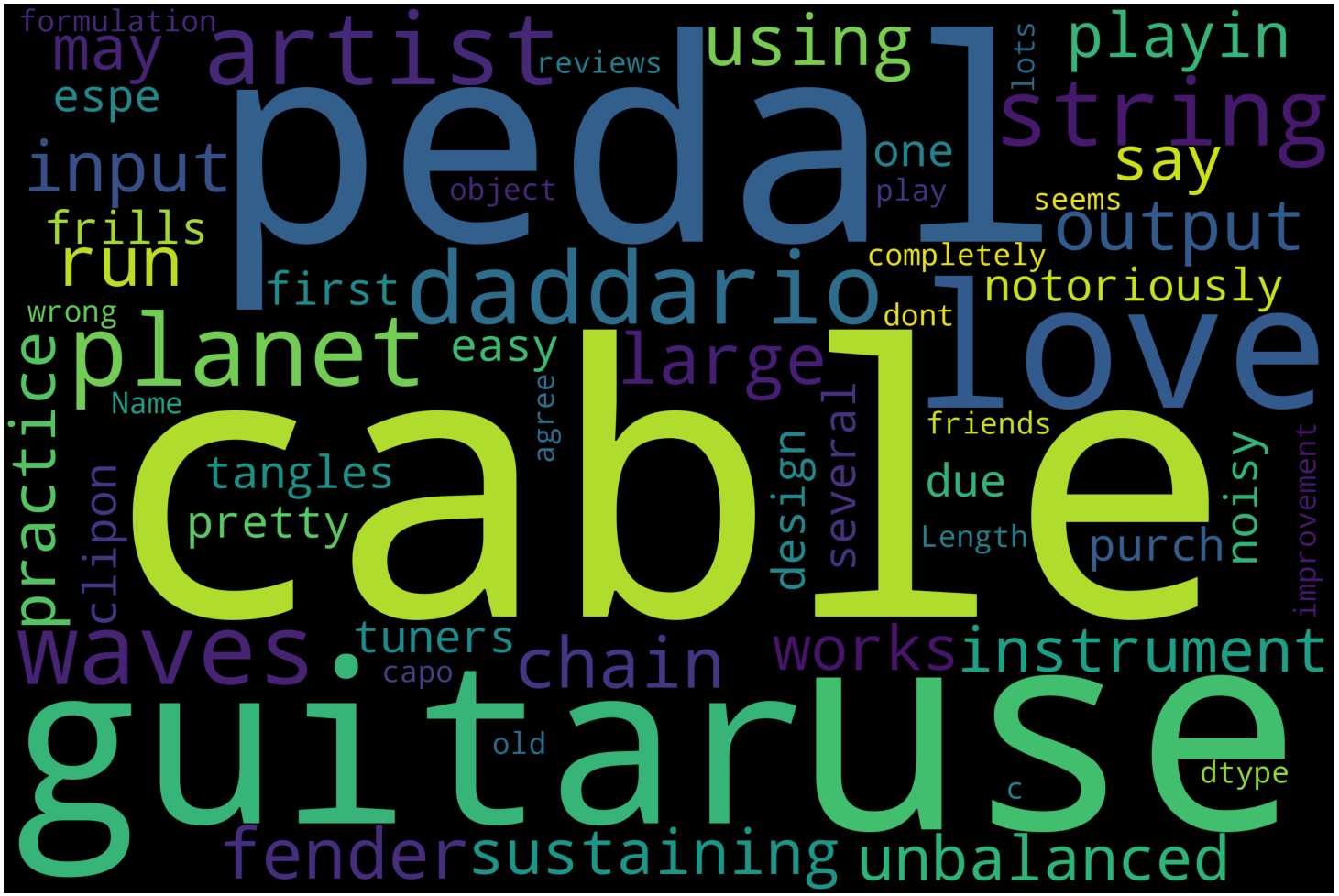
```
## Get the bar chart from positive reviews ##
for sent in review_pos["reviews"]:
    for word in generate_ngrams(sent,3):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
fd_sorted.columns = ["word", "wordcount"]
trace0 = horizontal_bar_chart(fd_sorted.head(25), 'green')

## Get the bar chart from neutral reviews ##
freq_dict = defaultdict(int)
for sent in review_neu["reviews"]:
    for word in generate_ngrams(sent,3):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
fd_sorted.columns = ["word", "wordcount"]
trace1 = horizontal_bar_chart(fd_sorted.head(25), 'grey')

## Get the bar chart from negative reviews ##
freq_dict = defaultdict(int)
for sent in review_neg["reviews"]:
    for word in generate_ngrams(sent,3):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
fd_sorted.columns = ["word", "wordcount"]
trace2 = horizontal_bar_chart(fd_sorted.head(25), 'red')
```

```
# Creating two subplots
fig = tools.make_subplots(rows=3, cols=1, vertical_spacing=0.04, horizontal_spacing=0.05,
                          subplot_titles=["Tri-gram plots of Positive reviews",
                                          "Tri-gram plots of Neutral reviews",
                                          "Tri-gram plots of Negative reviews"])
fig.append_trace(trace0, 1, 1)
fig.append_trace(trace1, 2, 1)
fig.append_trace(trace2, 3, 1)
fig['layout'].update(height=1200, width=1200, paper_bgcolor='rgb(233,233,233)', title="Trigram Count Plots")
iplot(fig, filename='word-plots')
```

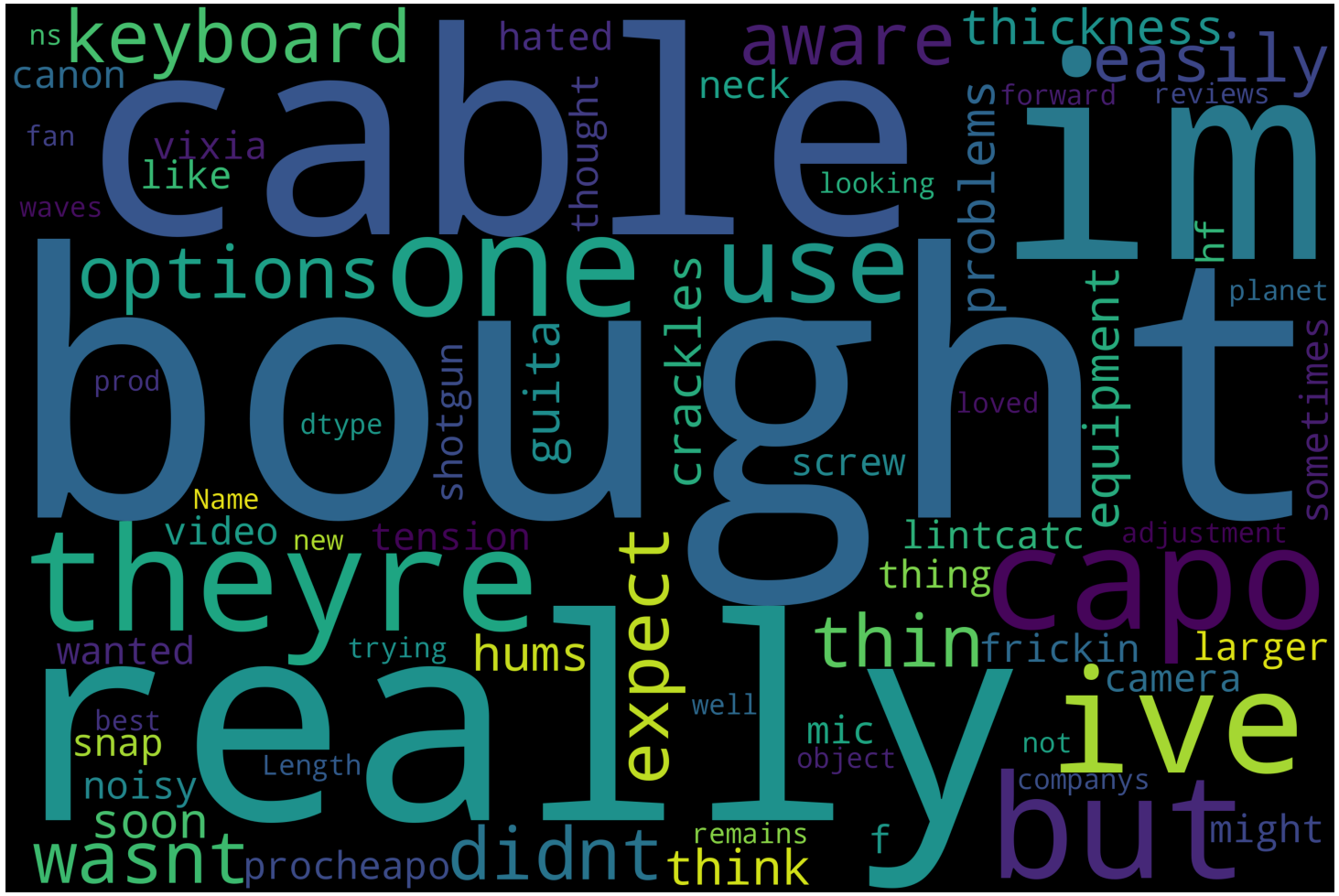
```
text = review_pos["reviews"]
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'black',
    stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```


write
windscreen
thinking
filter
strings
protects
reviews
po
breath
allfive
studio
looks
guitars
stri
dtype
object
developed
past
mic
mxl
pop
realized
block
elixir
made
primary
supposed
wouldnt
call
nice
performs
pu
including
exactly
prevents
taylor
really
tried
Length
trying
fi
much
expected
thank
job
bit
device
quite
great
nanoweb
Name
ne
per
ive
coated
good
stars
affordablei
otherwis
well

```
text = review_neu["reviews"]
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'black',
    stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```


formulation
may
artist
reviews
using
lots
playin
espe
pedal
string
input
one
object
play
say
seems
frills
output
run
completely
first
daddario
notoriously
wrong
dont
planet
easy
large
love
practice
Name
agree
cable
friends
tangles
several
design
due
noisy
clipon
pretty
Length
purch
improvement
waves
tuners
chain
works
instrument
capo
guitaruse
old
dtype
c
fendersustaining
unbalanced

```
text = review_neg["reviews"]
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'black',
    stopwords = stop_words).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```




```
# calling the label encoder function
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'sentiment'.
process_reviews['sentiment']= label_encoder.fit_transform(process_reviews['sentiment'])

process_reviews['sentiment'].unique()
```

```
array([2, 1, 0])
```

```
process_reviews['sentiment'].value_counts()
```

```
2    9022
1     772
0     467
Name: sentiment, dtype: int64
```