Amazon Product Review: Sentiment Analysis and Star Rating Prediction

Install required libraries

```
!pip install swifter
!pip install pyLDAvis
!pip install unidecode
!pip install TextBlob
!pip install text2emotion
```

```
Collecting swifter
  Downloading https://files.pythonhosted.org/packages/f4/3b/04bf42b94a22725241b47e025
     |████████████████████████████████| 634kB 7.5MB/s
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.6/dist-package
Collecting psutil>=5.6.6
  Downloading https://files.pythonhosted.org/packages/33/e0/82d459af36bda999f82c7ea86
     |████████████████████████████████| 471kB 14.7MB/s
Requirement already satisfied: dask[dataframe]>=2.10.0 in /usr/local/lib/python3.6/di
Requirement already satisfied: tqdm>=4.33.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: ipywidgets>=7.0.0cloudpickle>=0.2.2 in /usr/local/lib/
Requirement already satisfied: parso>0.4.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: bleach>=3.1.1 in /usr/local/lib/python3.6/dist-package
Collecting modin[ray]>=0.8.1.1
  Downloading https://files.pythonhosted.org/packages/ab/a9/ead212fa94de8f14459e22b06
     |████████████████████████████████| 542kB 20.1MB/s
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.6/dis
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.6/dist-package
Collecting partd>=0.3.10; extra == "dataframe"
  Downloading https://files.pythonhosted.org/packages/44/e1/68dbe731c9c067655bff1eca5
Requirement already satisfied: toolz>=0.7.3; extra == "dataframe" in /usr/local/lib/p
Collecting fsspec>=0.6.0; extra == "dataframe"
  Downloading https://files.pythonhosted.org/packages/a5/8b/1df260f860f17cb0869817015
     |████████████████████████████████| 92kB 9.4MB/s
Requirement already satisfied: ipython>=4.0.0; python_version >= "3.3" in /usr/local/
Requirement already satisfied: nbformat>=4.2.0 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: traitlets>=4.3.1 in /usr/local/lib/python3.6/dist-pack
Requirement already satisfied: ipykernel>=4.5.1 in /usr/local/lib/python3.6/dist-pack
Requirement already satisfied: widgetsnbextension~=3.5.0 in /usr/local/lib/python3.6/
Requirement already satisfied: webencodings in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: packaging in /usr/local/lib/python3.6/dist-packages (f
Collecting ray>=1.0.0; extra == "ray"
  Downloading https://files.pythonhosted.org/packages/12/87/44476ad712acc1f7957cbf88d
     |████████████████████████████████| 23.1MB 2.0MB/s
Collecting pyarrow==1.0; extra == "ray"
  Downloading https://files.pythonhosted.org/packages/a1/0a/a89de6d747c4698af128a4639
     |████████████████████████████████| 17.2MB 125kB/s
```

```
Collecting locket
  Downloading https://files.pythonhosted.org/packages/d0/22/3c0f97614e0be8386542facb3
Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: decorator in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/lib/python3
Requirement already satisfied: pygments in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.6/dist-pack
Requirement already satisfied: pexpect; sys_platform != "win32" in /usr/local/lib/pyt
Requirement already satisfied: pickleshare in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /usr/local/lib/python3.6/di
Requirement already satisfied: jupyter-core in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.6/dist-pack
Requirement already satisfied: tornado>=4.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: notebook>=4.4.1 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.6/dist-pack
Collecting aiohttp-cors
  Downloading https://files.pythonhosted.org/packages/13/e7/e436a0c0eb5127d8b491a9b83
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (fr
Collecting redis<3.5.0,>=3.3.2
```

Import required libraries

```
import time
import swifter
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
import calendar

from textblob import TextBlob
import text2emotion as te

import gensim
from gensim import corpora
import pyLDAvis
import pyLDAvis.gensim

from sklearn.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, classifi

import nltk
from nltk import FreqDist
nltk.downloader.download('vader_lexicon')
```

```
from nltk.sentiment import SentimentAnalyzer

import spacy
from spacy.lang.en.stop_words import STOP_WORDS
from spacy.lang.en import English

import re
import gzip
import string
import unidecode
from bs4 import BeautifulSoup
import plotly.graph_objects as go
from wordcloud import WordCloud,STOPWORDS

import sys
import heapq

sns.set()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
/usr/local/lib/python3.6/dist-packages/nltk/twitter/__init__.py:20: UserWarning: The twy
  warnings.warn("The twython library has not been installed. "
```

```
"""
We will ignore FutureWarning and DeprecationWarning
"""
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=DeprecationWarning)

"""
We will ignore warnings
"""
warnings.filterwarnings("ignore")

if not sys.warnoptions:
    warnings.simplefilter("ignore")
program_start_time=time.time()
```

Download dataset

```
!wget https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Mobile_Electronics_v1
```

```
--2020-11-28 23:22:09--  https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_
Resolving s3.amazonaws.com (s3.amazonaws.com)... 52.216.141.134
Connecting to s3.amazonaws.com (s3.amazonaws.com)|52.216.141.134|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 22870508 (22M) [application/x-gzip]
Saving to: 'amazon_reviews_us_Mobile_Electronics_v1_00.tsv.gz'

amazon_reviews_us_M 100%[===================>]  21.81M  23.8MB/s    in 0.9s

2020-11-28 23:22:10 (23.8 MB/s) - 'amazon_reviews_us_Mobile_Electronics_v1_00.tsv.gz' sa
```

Read the file

```
%%time
with gzip.open('amazon_reviews_us_Mobile_Electronics_v1_00.tsv.gz') as f:
    df = pd.read_csv(f, sep='\t', error_bad_lines=False)

df.head()
```

```
b'Skipping line 35246: expected 15 fields, saw 22\n'
b'Skipping line 87073: expected 15 fields, saw 22\n'
CPU times: user 1.11 s, sys: 112 ms, total: 1.23 s
Wall time: 1.24 s
```

Data information

```
df.shape
```

```
(104852, 15)
```

The dataset origannly contains 104852 rows and 15 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 104852 entries, 0 to 104851
Data columns (total 15 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   marketplace        104852 non-null  object
 1   customer_id        104852 non-null  int64
 2   review_id          104852 non-null  object
 3   product_id         104852 non-null  object
 4   product_parent     104852 non-null  int64
 5   product_title      104852 non-null  object
 6   product_category   104852 non-null  object
```

```
 7   star_rating        104850 non-null  float64
 8   helpful_votes      104850 non-null  float64
 9   total_votes        104850 non-null  float64
 10  vine               104850 non-null  object
 11  verified_purchase  104850 non-null  object
 12  review_headline    104848 non-null  object
 13  review_body        104849 non-null  object
 14  review_date        104850 non-null  object
dtypes: float64(3), int64(2), object(10)
memory usage: 12.0+ MB
```

```
df.columns
```

```
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'],
      dtype='object')
```

Columns Description:

marketplace: Marketplace of the product

customer_id: ID of the reviewer

review_id: ID of the review

product_id: ID of the product

product _parent: ID of the product parent

product_title: Product name

product_category: Category of the product

star_rating: Rating of the product from 1 to 5, 1 being the lowest

helpful_votes: Helpful votes of the review

total_votes: Total votes of the review

vine: Indicator for vine review

verified_purchase: Indicates if the purchase is verified or not

review_headline: Customer review title

review_body: Customer review summary

review_date: Customer review date

```
df.isnull().sum()
```

```
marketplace          0
customer_id          0
review_id            0
product_id           0
product_parent       0
product_title        0
product_category     0
star_rating          2
helpful_votes        2
total_votes          2
vine                 2
verified_purchase    2
review_headline      4
review_body          3
review_date          2
dtype: int64
```

We have few reviews that do not contain certain information such as star_rating, review_headline etc.

```
# Making a copy of the dataframe
data = df.copy()
```

```
data.shape
```

```
(104852, 15)
```

Missing data

```
# Dropping rows with missing information
data.dropna(axis = 0, how ='any', inplace = True)
data = data.reset_index(drop=True)
```

```
data.isnull().sum()
```

```
marketplace          0
customer_id          0
review_id            0
product_id           0
product_parent       0
product_title        0
```

```
product_category     0
star_rating          0
helpful_votes        0
total_votes          0
vine                 0
verified_purchase    0
review_headline      0
review_body          0
review_date          0
dtype: int64
```

```
data.shape
```

```
(104847, 15)
```

The dataset now contains 104847 rows and 15 columns

```
# Unique Reviews
len(pd.unique(data['review_id']))
```

```
104847
```

All the reviews are unique

Data Statistics

```
data.describe()
```

	customer_id	product_parent	star_rating	helpful_votes	total_votes
count	1.048470e+05	1.048470e+05	104847.000000	104847.000000	104847.000000
mean	2.793730e+07	5.015301e+08	3.763160	1.244032	1.615440
std	1.508714e+07	2.871676e+08	1.523537	7.070485	7.910005
min	1.007100e+04	5.352400e+04	1.000000	0.000000	0.000000
25%	1.471380e+07	2.593731e+08	3.000000	0.000000	0.000000
50%	2.650319e+07	4.939016e+08	4.000000	0.000000	0.000000
75%	4.223490e+07	7.440083e+08	5.000000	1.000000	1.000000
max	5.309657e+07	9.999508e+08	5.000000	769.000000	791.000000

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

Deriving additional information

```
# Converting review_date to datetime object to extract month, day & year
%%time
data['review_date'] =  pd.to_datetime(data['review_date'], format='%Y-%m-%d')
```

```
CPU times: user 25.2 ms, sys: 2.02 ms, total: 27.3 ms
Wall time: 29.7 ms
```

```
# Extracting month, day and year
%%time
data['day'] = data['review_date'].apply(lambda r:r.day)
data['month'] = data['review_date'].apply(lambda r:r.month)
data['year'] = data['review_date'].apply(lambda r:r.year)
```

```
CPU times: user 1.46 s, sys: 25.9 ms, total: 1.49 s
Wall time: 1.49 s
```

```
data.head(2)
```

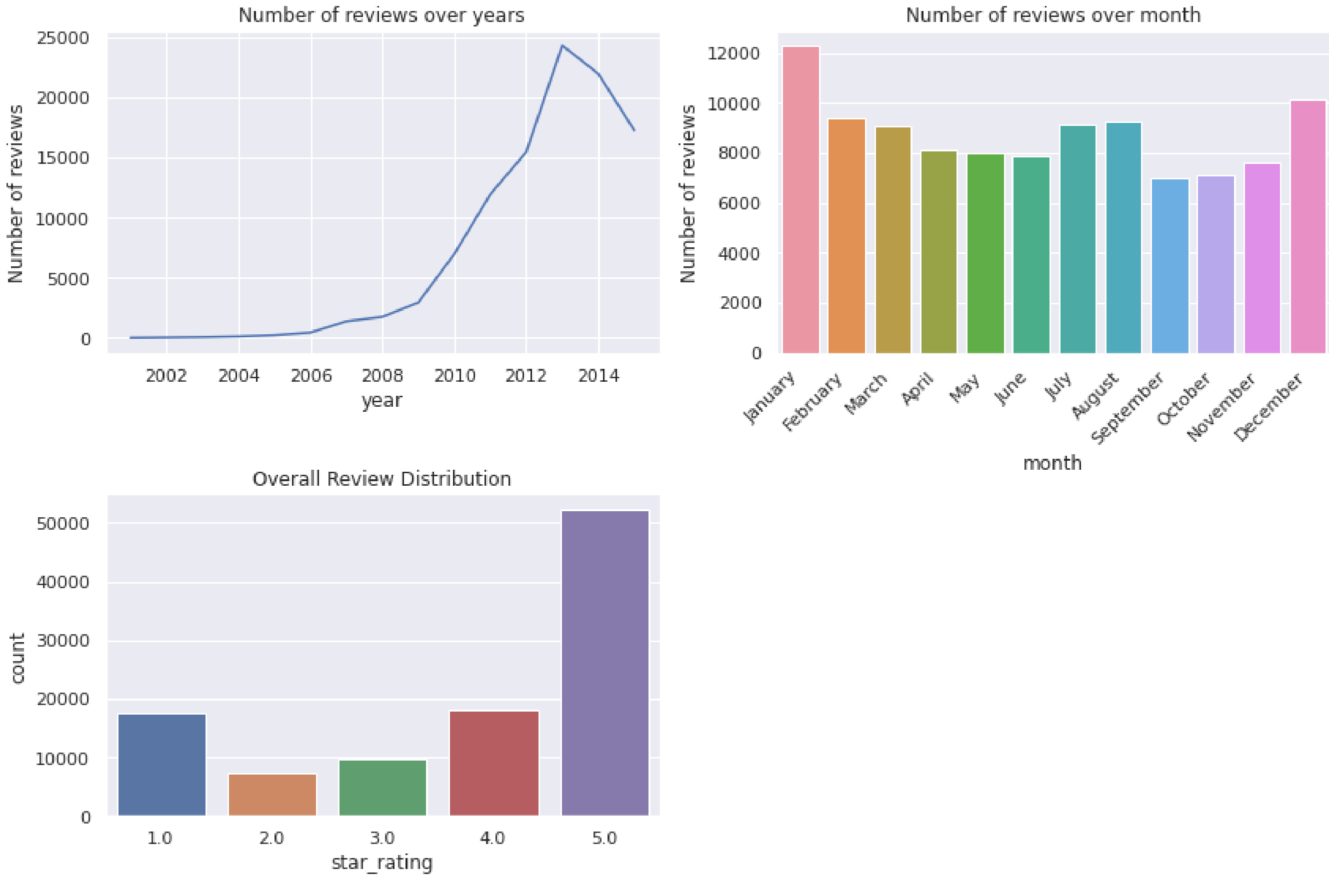
	marketplace	customer_id	review_id	product_id	product_parent	product_t
0	US	20422322	R8MEA6IGAHO0B	B00MC4CED8	217304173	Blacl DR600
1	US	40835037	R31LOQ8JGLPRLK	B00OQMFG1Q	137313254	GENSSI G GPS Two Smart Pl Car Ala

Review Trend over time

```
%%time
f, axes = plt.subplots(2,2, figsize=(12,8))
# Yearly Reviews
yearly = data.groupby(['year'])['review_id'].count().reset_index()
yearly = yearly.rename(columns={'review_id':'Number of reviews'})
year_trend = sns.lineplot(x='year',y='Number of reviews',data=yearly, ax = axes[0,0])
year_trend.set_title('Number of reviews over years')

# Monthly Reviews
monthly = data.groupby(['month'])['review_id'].count().reset_index()
monthly['month'] = monthly['month'].apply(lambda x : calendar.month_name[x])
monthly = monthly.rename(columns={'review_id':'Number of reviews'})
month_trend = sns.barplot(x='month',y='Number of reviews',data=monthly, ax = axes[0,1])
month_trend.set_title('Number of reviews over month')
month_trend.set_xticklabels(month_trend.get_xticklabels(), rotation = 45, horizontalalignment

# Getting overall ratings for products
sns.countplot(x = 'star_rating', data = data, ax = axes[1,0] ).set_title('Overall Review Dist
```

```
f.delaxes(axes[1][1])
f.tight_layout()
```

```
CPU times: user 316 ms, sys: 91 ms, total: 407 ms
Wall time: 310 ms
```



Rating Trend over the years

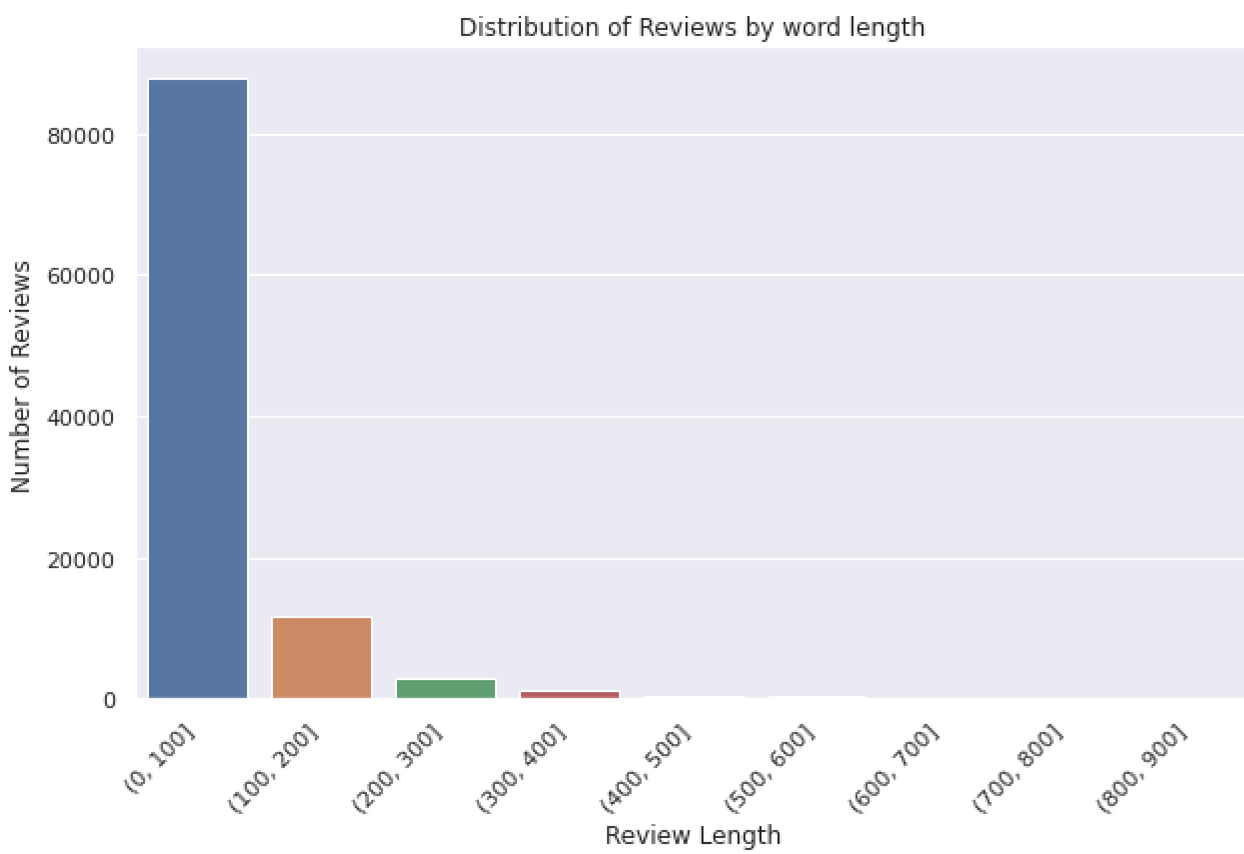
- *There is an increasing trend for number of ratings given by the users to products on Amazon which indicates that a greater number of users started using the Amazon e-commerce site for online shopping and a greater number of users started giving feedback on the products purchased from 2006 to 2014. There is a significant increase in number of ratings given by users from 2012 to 2013.*
- *Notice the peak on 2013. Two major events support this. Amazon began to offer Sunday delivery option for purchases. See [news article]() here. That surely resulted in lots of new members and increased ratings & reviews.*

Distribution of overall ratings

- *Many users have given a rating of 5 to products followed by 4 and 1 whereas very few users have given a low rating of 2 and 3.*

Distribution of reviews by word length

```
%%time
plt.figure(figsize=(10,6))
electronics_reviews = data[['review_id','customer_id','review_body','review_headline','star_r
electronics_reviews['review_length'] = electronics_reviews['review_body'].apply(lambda x: len
reviews_word_length = electronics_reviews.groupby(pd.cut(electronics_reviews.review_length, n
reviews_word_length = reviews_word_length.rename(columns={'review_length':'count'})
reviews_word_length = reviews_word_length.reset_index()
#print(reviews_word_length)
reviewLengthChart = sns.barplot(x='review_length',y='count',data=reviews_word_length)
reviewLengthChart.set_title('Distribution of Reviews by word length')
reviewLengthChart.set_xticklabels(reviewLengthChart.get_xticklabels(), rotation = 45, horizon
plt.xlabel("Review Length")
plt.ylabel("Number of Reviews")
plt.show()
```

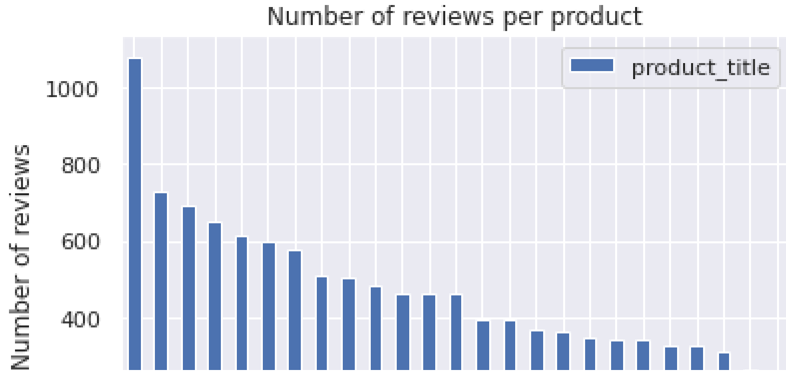



```
CPU times: user 619 ms, sys: 4.61 ms, total: 624 ms
Wall time: 631 ms
```

Reviews per product

```
%%time
plt.figure(figsize=(10,6))
counts = data["product_title"].value_counts().to_frame()
counts.loc[counts['product_title'] > 250].plot(kind='bar')
plt.xlabel("Products")
plt.ylabel("Number of reviews")
plt.title("Number of reviews per product")
plt.show()
```

```
<Figure size 720x432 with 0 Axes>
```



Subsetting the dataframe to take the required columns

```
# Subsetting the dataframe
reviews = data[['review_headline','review_body', 'star_rating']]
```

```
# Concat review headline and review body columns into one single column
reviews['review'] = reviews['review_headline'].str.cat(reviews['review_body'],sep=" ")
reviews
```

	review_headline	review_body	star_rating	review
0	Very Happy!	As advertised. Everything works perfectly, I'm...	5.0	Very Happy! As advertised. Everything works pe...
1	five star	it's great	5.0	five star it's great
2	great cables	These work great and fit my life proof case fo...	5.0	great cables These work great and fit my life ...
3	Work very well but couldn't get used to not he...	Work very well but couldn't get used to not he...	4.0	Work very well but couldn't get used to not he...
4	Cameras has battery issues	Be careful with these products, I have bought ...	2.0	Cameras has battery issues Be careful with the...
...	...	...	...	...
[illegible]	[illegible]	I've been looking for a	[illegible]	The Cat Barf is

Polarity and Subjectivity

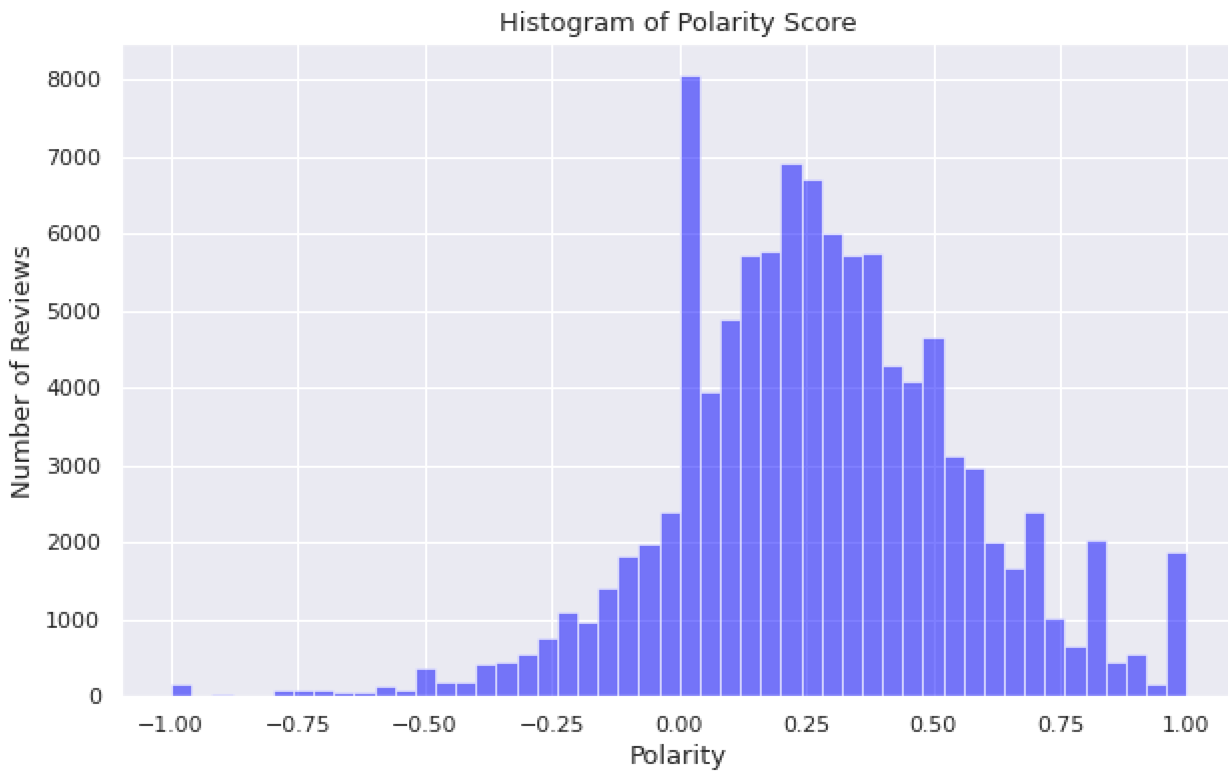
```
## Calculating review polarity using TextBlob
%%time
polarity = lambda x: TextBlob(x).sentiment.polarity
```

```
reviews['review_polarity'] = reviews['review'].swifter.apply(polarity)
```

Pandas Apply: 100% 104847/104847 [01:36<00:00, 1087.60it/s]

```
CPU times: user 1min 25s, sys: 680 ms, total: 1min 26s
Wall time: 1min 26s
```

```
# Plotting Histogram of Polarity Score
num_bins = 50
plt.figure(figsize=(10,6))
n, bins, patches = plt.hist(reviews.review_polarity, num_bins, facecolor='blue', alpha=0.5)
plt.xlabel('Polarity', fontsize=13)
plt.ylabel('Number of Reviews', fontsize=13)
plt.title('Histogram of Polarity Score', fontsize=13)
plt.show()
```

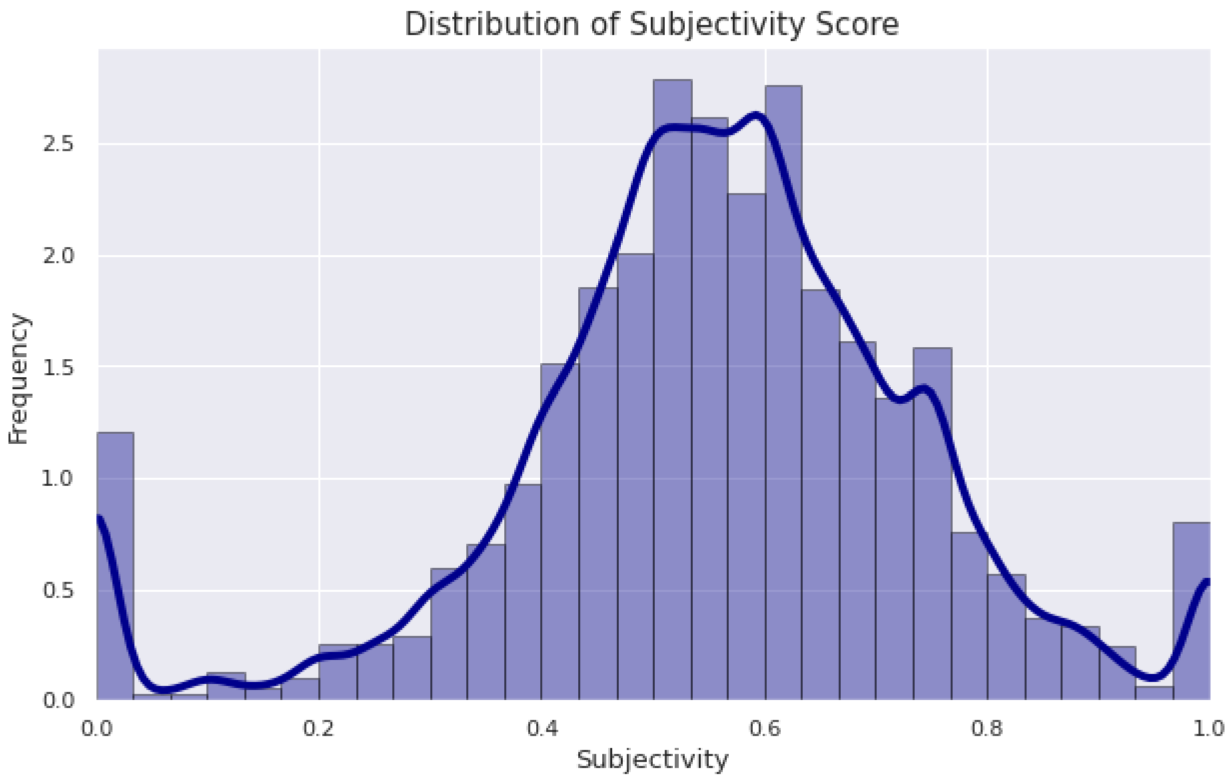


Although maximum reviews have positive emotions, there is peak for neutral reviews and be seen at 0.

```
# Calculating review subjectivity
%%time
sub = lambda x: TextBlob(x).sentiment.subjectivity
reviews['review_subjectivity'] = reviews['review'].swifter.apply(sub)
```

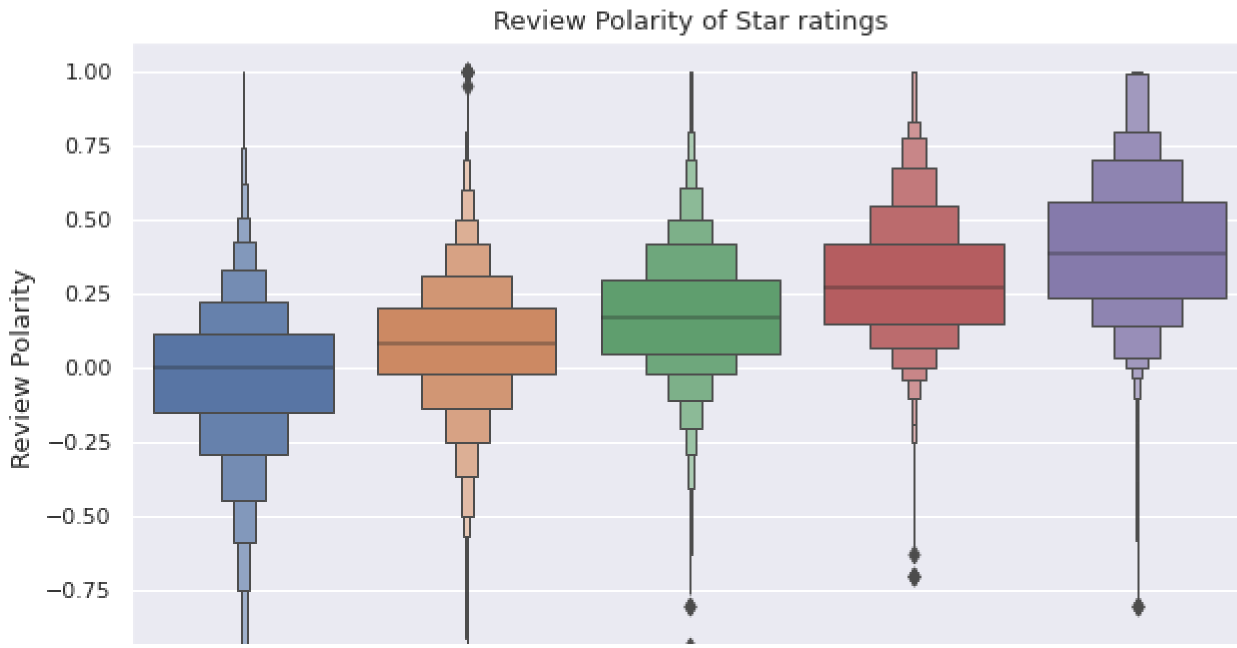
Pandas Apply: 100% 104847/104847 [02:12<00:00, 794.07it/s]

```
# Plotting distribution of subjectivity
plt.figure(figsize=(10,6))
sns.distplot(reviews.review_subjectivity, hist=True, kde=True, bins=int(30), color = 'darkblu
plt.xlim([-0.001,1.001])
plt.xlabel('Subjectivity', fontsize=13)
plt.ylabel('Frequency', fontsize=13)
plt.title('Distribution of Subjectivity Score', fontsize=15)
plt.show()
```



There is quite a normal distribution for subjectivity. However, there are many reviews which are fully subjective and fully objective.

```
# Plotting polarity of star ratings
plt.figure(figsize=(10,6))
sns.boxenplot(x='star_rating', y='review_polarity', data=reviews)
plt.xlabel("Star Rating", fontsize=13)
plt.ylabel("Review Polarity", fontsize=13)
plt.title("Review Polarity of Star ratings", fontsize=13)
plt.show()
```



In the above plot, we can see that polarity increases with star rating. There are very few reviews with 5 rating and negative polarity and 1 rating with positve polarity.

```
#Some Positive reviews that has bad rating
reviews.loc[(reviews.review_polarity == -1) & (reviews.star_rating == 5)].review.head().tolis
```

```
['The mount that comes with the g1wh is horrible. Worked as described. The mount that co
 "THUMPER This sub pounds the hell out of my girls car, and for the price you can't go w
 "Crazy!!! Saw the video and bought this. You know this isn't Bose audio, but it works j
 'terrible its does not record your voice for memos and you have to do it on the compute
```

All the above reviews have the negative words which might have made the polarity negative. These words are: horrible, hell, wrong, disappoint

```
#Some Negative reviews that has good rating
reviews.loc[(reviews.review_polarity == 1) & (reviews.star_rating == 1)].review.head().tolist
```

```
["One Star Wasn't very happy with it it did not have the power that I wanted",
 'One Star I am very happy with the product that I bought',
 "EARPOD doesn'work THE  EARPOD DIDN'T WORK ( had to buy one by Best Buy)<br />DELIVERED
 'One Star The product had a shortage in the cord. I ordered another which worked perfec
 'One Star one works perfectly and one is a defective']
```

All the above reviews have the most positive words which might have made the polarity positive. These words are: Very happy, Best buy, perfectly

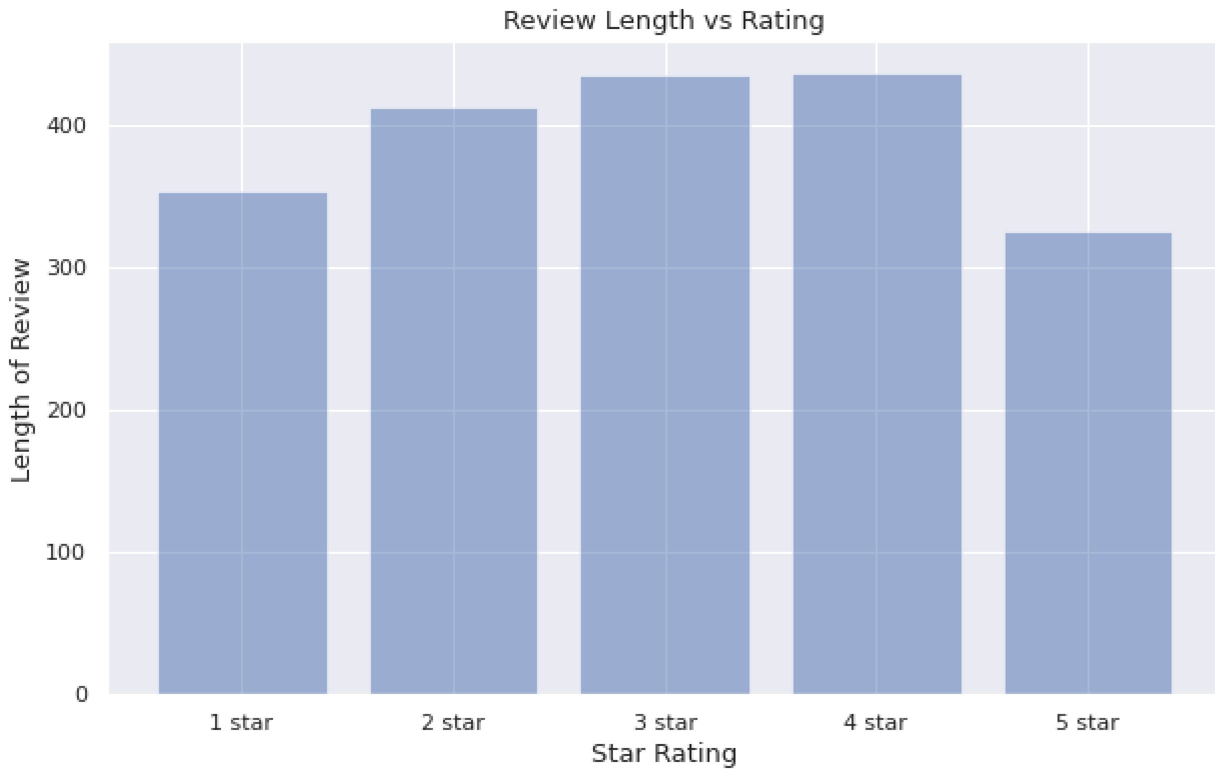
Review length

```
# Calculating length of each review
length_of_review=[]

for i, word in enumerate(reviews.review.tolist()):
    word_length = len(word)
    length_of_review.append(word_length)
reviews['review_length'] =length_of_review
display(reviews)
```

	review_headline	review_body	star_rating	review	review_polarity	review_s
0	Very Happy!	As advertised. Everything works perfectly, I'm...	5.0	Very Happy! As advertised. Everything works pe...	0.666667	
1	five star	it's great	5.0	five star it's great	0.800000	
2	great cables	These work great and fit my life proof case fo...	5.0	great cables These work great and fit my life ...	0.666667	
3	Work very well but couldn't get used to not he...	Work very well but couldn't get used to not he...	4.0	Work very well but couldn't get used to not he...	0.200000	
4	Cameras has battery issues	Be careful with these products, I have bought ...	2.0	Cameras has battery issues Be careful with the...	0.139225	
...	...	...	...	...	...	
	The Cat Barf is	I've been looking for a		The Cat Barf is Gone! I've		

```
# Plotting Star rating vs review length
L1= reviews[reviews['star_rating']==1]['review_length'].mean()
L2= reviews[reviews['star_rating']==2]['review_length'].mean()
L3= reviews[reviews['star_rating']==3]['review_length'].mean()
L4= reviews[reviews['star_rating']==4]['review_length'].mean()
L5= reviews[reviews['star_rating']==5]['review_length'].mean()
plt.figure(figsize=(10,6))
review_length_comparison=[L1,L2,L3,L4,L5]
```

```
objects = ('1 star','2 star','3 star', '4 star' ,'5 star')
y_pos = np.arange(len(objects))
plt.bar(y_pos, review_length_comparison, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('Length of Review', fontsize=13)
plt.xlabel('Star Rating', fontsize=13)
plt.title('Review Length vs Rating', fontsize=13)
plt.show()
```



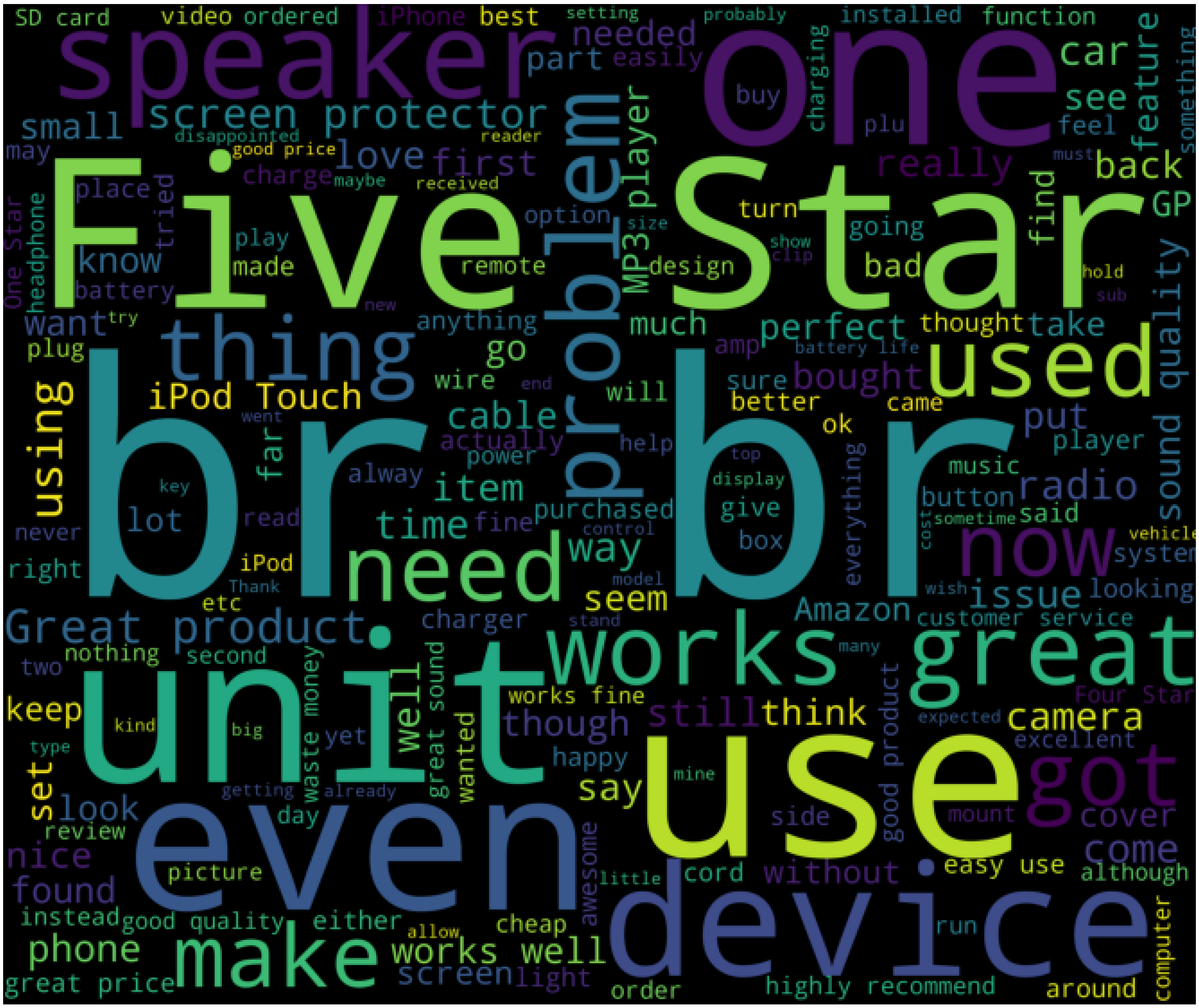
People tend to give very long reviews when they are explaining details about the product. For example, if someone doesn't like a particular feature of some product they explain it in detail. On the other hand, best reviews mostly are given in few words.

Word Cloud

```
# Creating word cloud
def review_word_cloud(reviews):
    words = " ".join(reviews)

    wordcloud = WordCloud(stopwords=STOPWORDS,
                          background_color='black',
                          width=3000,
                          height=2500
                         ).generate(words)
```

```
    plt.figure(1,figsize=(12, 12))
    plt.imshow(wordcloud)
    plt.axis('off')
    plt.savefig("distributions.png")
    plt.show()
```

```
review_word_cloud(reviews["review"])
```



The above wordcloud is formed with the review text before text preprocessing.

Text2Emotion

Emotion Detection from text is useful in many ways. It helps to understand the customers, analyze feedback and reviews.

```
t2e = reviews[['review', 'star_rating']]
```

```
t2e = reviews[['review', 'star_rating']]
t2e = t2e.sample(frac = 0.1)


# Deriving emotion
start = time.time()
t2e["emotion_factor"] = t2e.review.map(te.get_emotion)
print((time.time()-start)/60, 'mins')
```

```
40.04022561709086 mins
```

```
%%time
t2e = pd.concat([t2e, t2e['emotion_factor'].apply(pd.Series)], axis = 1)
```

```
CPU times: user 3.44 s, sys: 121 ms, total: 3.57 s
Wall time: 3.49 s
```

```
t2e
```

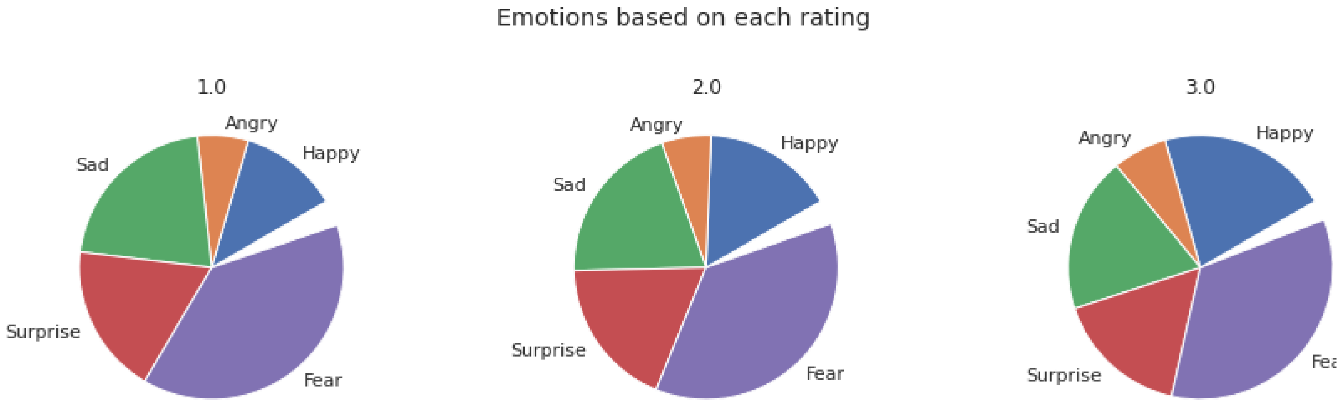
	review	star_rating	emotion_factor	Happy	Angry	Surprise	Sad	Fe
70606	Kick butt Nice amp for low watts. Running kenw...	5.0	{'Happy': 0.5, 'Angry': 0.0, 'Surprise': 0.25,...	0.50	0.00	0.25	0.25	0.
31226	Really poor quality I got this to use at work ...	1.0	{'Happy': 0.13, 'Angry': 0.0, 'Surprise': 0.33...	0.13	0.00	0.33	0.07	0.
41292	Great Price - All Necessary Pieces - Buy it Th...	5.0	{'Happy': 0.25, 'Angry': 0.0, 'Surprise': 0.38...	0.25	0.00	0.38	0.25	0.
95544	Worthwhile Alternative For anyone who finds an...	5.0	{'Happy': 0.29, 'Angry': 0.0, 'Surprise': 0.0,...	0.29	0.00	0.00	0.29	0.
22203	Very bad quality Broke in a month. After disas...	2.0	{'Happy': 0.0, 'Angry': 0.0, 'Surprise': 0.33,...	0.00	0.00	0.33	0.67	0.

```
# Deriving tone of the reviews
def get_tone(dct):
  sorted_d = sorted(dct.items(), key=lambda kv: kv[1], reverse=True)
  if sorted_d[1][1] != 0.0 and (sorted_d[0][1] != sorted_d[1][1]) :
    return  'More ' + sorted_d[0][0]+' Than '+sorted_d[1][0]
  elif sorted_d[0][1] == sorted_d[1][1]:
    return 'Both ' + sorted_d[0][0] + ' and ' + sorted_d[1][0]
  else:
    return sorted_d[0][0]
```

```
# Deriving tone of each review
t2e["Tone"] = t2e.emotion_factor.apply(get_tone)
t2e.head()
```

	review	star_rating	emotion_factor	Happy	Angry	Surprise	Sad	Fear	To
70606	Kick butt Nice amp for low watts. Running kenw...	5.0	{'Happy': 0.5, 'Angry': 0.0, 'Surprise': 0.25,...	0.50	0.0	0.25	0.25	0.00	Mo Hap Th Surpri
31226	Really poor quality I got this to use at work ...	1.0	{'Happy': 0.13, 'Angry': 0.0, 'Surprise': 0.33...	0.13	0.0	0.33	0.07	0.47	Mo Fe Th Surpri

```
# Plotting emotions based on each rating
ll =t2e.groupby("star_rating", as_index=True)[['Happy', 'Angry', 'Sad', 'Surprise', 'Fear']].
ll.reset_index(inplace=True)

fig, axes = plt.subplots(2, 3, figsize=(15, 8))
plt.suptitle('Emotions based on each rating')

for i, (idx, row) in enumerate(ll.set_index('star_rating').iterrows()):
    ax = axes[i // 3, i % 3]
    row = row[row.gt(row.sum() * .01)]
    ax.pie(row, labels=row.index, startangle=30)
    ax.set_title(idx)
fig.delaxes(axes[1][2])
plt.show()
```



Named Entity Recognition

NER helps to get a glance, understand the subject or theme of the reviews

```
# Getting top 20 most helpful votes
helpful = data["helpful_votes"].tolist()
most_helpful = heapq.nlargest(20, helpful)


# Joined the most helpful reviews
df_ner = data.loc[data.index.intersection(most_helpful)]
helpfultext = " ".join(df_ner['review_body'])
helpfultext
```

```
'Works great! Got it in red and black and already in love with it. Hands down one of the
best Bluetooth speakers in this price range. If you're looking for cheap Bluetooth speak
rs SoundBot SB571 is the way to go. This is the worst radio I ever bought toch screen st
p working very fast and the seller is the worst people to deal with don't buy from Waite
a month maybe longer, when it finally arrived it was not a Acten but some other model.<b
/>rear view camera works fine but navigation icon does not respond. Does not flow easily
will return. It was loud at first and then after a couple weeks it kinda just gave up ar
wouldn't turn up even with the volume on my phone all the way up. And then the side of i
```

```
# Deriving entities
from collections import defaultdict
ner = spacy.load("en")
ner_helpful = ner(helpfultext)
ner_dict = defaultdict(list)
for entity in ner_helpful.ents:
    ner_dict[entity.label_].append(entity)
for NER, name in ner_dict.items():
    print(f"{NER}:\n{name}\n")
```

```
CARDINAL:
[one, two, two, about 10-15 feet, One, five]

ORG:
[SoundBot SB571, Sony, Sony, iPod, Considering, Toyota]
```

```
ORDINAL:
[first, first, first, first]

DATE:
[a couple weeks, a couple weeks, 2 days, two weeks]

GPE:
[kinda, kinda]

NORP:
[chinese]

MONEY:
[100 plus dollars]

LAW:
[the Camry Visor Repair]
```

```
# Getting the most reviewed product
product = data[data["product_id"] == 'B00J46XO9U']
```

```
# Joining reviews
product_text = " ".join(product['review_body'])
product_text
```

```
'Very good quality. So far so good. Good product and good seller Great product! They cha
ge my wife\'s phone Works like a charm! I\'ve ordered a ton of these white and black, lo
g and short.  I keep buying them because they are made so well.  I need more to buy for
he office, my car, the house...etc... awesome stuff! It broke less than a month. Don\'t
uy them. I\'ve bought a total of 3 of these cables I\'ve had the first for over a month
nd works great. Lasted longer than the ones I got from Apple. I needed one that worked w
th my lifeproof case and this does This cable works lightening-fast. As soon as I plug i
into my iPhone, it recognizes the charger instantly and begins charging immediately. No
```

```
# Deriving entities
ner = spacy.load("en")
ner_helpful = ner(product_text)
ner_dict = defaultdict(list)
for entity in ner_helpful.ents:
    ner_dict[entity.label_].append(entity)
for NER, name in ner_dict.items():
    print(f"{NER}:\n{name}\n")
```

```
QUANTITY:
[a ton, 3ft, two 3 foot, 3 foot and, 6 foot, 3 foot, 6 foot, 2 charger, 6 ft cables, 3 f

DATE:
[less than a month, over a month, over a year, the day, almost three months, 2 days, dai

CARDINAL:
[3, zero, two, one, 3, 6, 4, 1, one, 2, Only 1, 3, 6, 4, 2, 10, one, two, One, 6, 3, one
```

```
ORDINAL:
[first, secondary, second, second, first, second, second, second, first, second, second,

ORG:
[Apple, iPhone, Apple, OEM, Apple, iPhone/ iPad Air, iPhone, Amazon, iPhone, USB, Apple,

MONEY:
[34;cheaper&#34, 34, 34;skin&#34, #6 iPhone, 34;accessory, 34;not, 34;certified&#34, 34;

GPE:
[Lightning, Walmart, Otterbox, Apple&#34, America, iPhone, iTunes, Lightning, China, Chi

NORP:
[Lifeproof, Lifeproof, Amazons, Working, Lightning, Lightning, Lightning]

PRODUCT:
[non-Apple, iPad, Amazon Basics, iPad, iPad, non-OEM, Chromo, non-Apple, iPads, USB, 6,

PERSON:
[Apple MFI, Excellent Cables, Nice Cables Great, jack, Charger, Lifeproof, Bummed, Light

LOC:
[iPhones, iPhones, iPhones, iPhones, the Amazon Basics, iPhones, SIX, the iPad Retina, i

FAC:
[5C, MacBook, Lightning, the Apple Store, 5Cs]

PERCENT:
[around 40%, 1%, 25% to 23%, 100%, 5 & 6 plus, 100%, 99%, 100%, 70%, 50%, 100%]

WORK_OF_ART:
[Apple MFI Certified, out=, Geniuses]

TIME:
[a few seconds, last night, 24 hours, midmorning, the afternoon, all night, last night,

LAW:
[the iPhone 6]

EVENT:
[iPhone, iPhone, iPhone]
```

```
# Using displacy to view entities
from spacy import displacy
displacy.render(ner_helpful, style="ent", jupyter=True)
```

Very good quality. So far so good. Good product and good seller Great product! They charge my wife's phone
Works like a charm! I've ordered a ton **QUANTITY** of these white and black, long and short. I keep buying
them because they are made so well. I need more to buy for the office, my car, the house...etc... awesome st
It broke less than a month **DATE** . Don't buy them. I've bought a total of 3 **CARDINAL** of these cables
I've had the first **ORDINAL** for over a month **DATE** and works great. Lasted longer than the ones I got
from Apple **ORG** . I needed one that worked with my lifeproof case and this does This cable works
lightening-fast. As soon as I plug it into my iPhone **ORG** , it recognizes the charger instantly and begins
charging immediately. No need to wiggle the cable around the port to get it to fit properly or for the device to
pick it up. I actually think it works better than the charger my iphone came with! I've had it for over a year
DATE now and the wire is still in excellent shape. No tears to the cable, like you often have with apple
headphones or other chargers, plus the wire is still neat and clean hasn't really accumulated any dirt or
smudges (I bought the white one so that says a lot). The cable also stays straight, i.e. it also doesn't tangle o
twist into a knot that you need to untangle. Whether you're looking for a primary iphone charger or a
secondary **ORDINAL** charging cable for your car or for traveling, this product gets the job done!

Overall, a great item at a great price (no need to overpay at the Apple **ORG** store). It works very well ar
still lasts and looks pretty much the same as the day **DATE** I bought it. Well done with the manufacturing,
definitely no knock-off experience here! Had this for almost three months **DATE** and it started to act up.
There was no visual signs of any broken issues. The cord would charge the phone and then randomly stop
charging. Worked just fine with my kids' iphones. Excellent product in a good price. You can't ask for anythinc
better than that in retail. awesome product, not a thin flimsy cable like the OEM **ORG** part from Apple
ORG . This is very durable and has zero **CARDINAL** connectivity issues. I should have bought this a lor
time ago, and the two **CARDINAL** pack price is great. I have ordered these twice now (because my kids
keep losing them !) and they hold up better than any others and charge the phone quickly Works great! Quali
is ok, but the cables tend to fray around the connection end fairly quickly. I have used several of these and lo
them. I have used a different &# 34;cheaper&#34 **MONEY** ; brand before but you'll soon receive the error

Text Preprocessing

pairs. I was able to stock up on my iPhone/ iPad Air **ORG** 2 cables. Great product fast shipping!! They wo

```
nlp = spacy.load('en_core_web_sm')
spacy_stopwords = spacy.lang.en.stop_words.STOP_WORDS
```

and charge my apple devices greatly. Thank you! hard to find Lightning GPE cables that work for a long

```
def decontracted(phrase):
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    phrase = re.sub(r" v ", " very", phrase)
    phrase = re.sub(r'\bthats\b', 'that is', phrase)
    phrase = re.sub(r'\bive\b', 'i have', phrase)
    phrase = re.sub(r'\bim\b', 'i am', phrase)
    phrase = re.sub(r'\bya\b', 'yeah', phrase)
    phrase = re.sub(r'\bcant\b', 'can not', phrase)
    phrase = re.sub(r'\bdont\b', 'do not', phrase)
    phrase = re.sub(r'\bwont\b', 'will not', phrase)
    phrase = re.sub(r'\bid\b', 'i would', phrase)
    phrase = re.sub(r'wtf', 'what the fuck', phrase)
    phrase = re.sub(r'\bwth\b', 'what the hell', phrase)
    phrase = re.sub(r'\br\b', 'are', phrase)
    phrase = re.sub(r'\bu\b', 'you', phrase)
    phrase = re.sub(r'\bk\b', 'OK', phrase)
    phrase = re.sub(r'\bsux\b', 'sucks', phrase)
    phrase = re.sub(r'\bno+\b', 'no', phrase)
    phrase = re.sub(r'\bcoo+\b', 'cool', phrase)
    phrase = re.sub(r'rt\b', '', phrase)
    phrase = phrase.strip()
    #print("decontracted:",phrase)
    return phrase
```

recommend these for anyone. Thanks. So far so good! Fast shipping & good product. (Had them about 2

```
# exclude words from spacy stopwords list
deselect_stop_words = ['no', 'not']
for w in deselect_stop_words:
    nlp.vocab[w].is_stop = False

# exclude words from spacy stopwords list
select_stop_words = ['#']
for w in select_stop_words:
    nlp.vocab[w].is_stop = True
```

combusted, fallen apart, or stopped working. Just take care of your things and they'll take care of you too.<br

```
def strip_html_tags(text):
    """remove html tags from text"""
    soup = BeautifulSoup(text, "html.parser")
    stripped_text = soup.get_text(separator=" ")
    #print("strip_html_tags:", stripped_text)
```

```
    return stripped_text
```

and are more durable than other cables I have used. It has even resisted some wear from my cat chewing o

```
def remove_accented_chars(text):
    """remove accented characters from text, e.g. café"""
    text = unidecode.unidecode(text)
    #print("remove_accented_chars:", text)
    return text
```

ones. Received two CARDINAL of these. One CARDINAL fell apart right when opened. These are che

```
def remove_extra_characters(text):
    """remove extra characters from text, e.g. aaaaawwwweeeessssoooommmeee"""
    text = re.sub("(.)\\1{2,}", "\\1", text)
    #print("remove_extra_characters:", text)
    return text


def keep_alphabet_numbers(text):
    """keep only words and numbers in the text"""
    text = re.sub('[^A-Za-z0-9]+', ' ', text)
    return text


def remove_urls(text):
    """remove url from the text"""
    # remove hyperlinks
    text = re.sub(r'\w+:\/{2}[\d\w-]+(\.[\d\w-]+)*(?:(?:\/[^\s/]*))*', '', text)
    return text
```

recommend Nice ORG issue free performance I purchased 3 CARDINAL packs of chargers over the

```
def text_preprocessing(text):
    """preprocess text with default option set to true for all steps"""
    text = strip_html_tags(text)
    text = remove_urls(text)
    text = remove_accented_chars(text)
    text = decontracted(text)
    text = remove_extra_characters(text)
    text = keep_alphabet_numbers(text)
    text = text.lower()
    tokens = nlp(text)
    review_text = [word for word in tokens if not word.is_stop]
    review_text = [word.lemma_ for word in review_text]
    return " ".join(review_text)


# Applying text-preprocessing to all the reviews
start = time.time()
reviews['preprocessed_review'] = reviews['review'].swifter.apply(lambda x: text_preprocessing

print((time.time()-start)/60, 'mins')
```

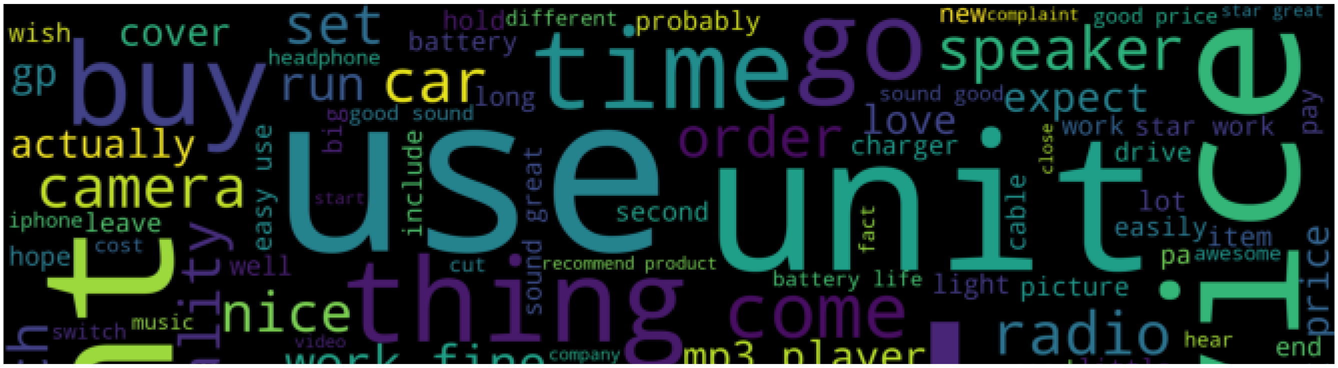
Pandas Apply: 100% 104847/104847 [26:27<00:00, 66.04it/s]

```
# Resetting the index
reviews = reviews.reset_index(drop=True)
```

```
reviews.head()
```

	review_headline	review_body	star_rating	review	review_polarity	review_subjec
0	Very Happy!	As advertised. Everything works perfectly, I'm...	5.0	Very Happy! As advertised. Everything works pe...	0.666667	0.(
1	five star	it's great	5.0	five star it's great	0.800000	0.7
2	great cables	These work great and fit my life proof case fo...	5.0	great cables These work great and fit my life ...	0.666667	0.(
	Work very well but	Work very well but		Work very well but		

```
review_word_cloud(reviews["preprocessed_review"])
```



The above wordcloud is formed using the the preprocessed text

Sentiment Analysis

Sentiment Analysis is the automated process of understanding the sentiment or opinion of a given text. It provides insights by automatically analyzing product reviews and separating them into tags: Positive, Neutral, Negative. In this part, We have used a prebuilt library VaderSentiment which is used in predicting the sentiment of a review based on the lexicon arrangement of the words in a review.

```
sid = nltk.sentiment.vader.SentimentIntensityAnalyzer()
```

```
# Storing the sentiment scores of postive, neutral & negative sentiments in lists
pos_word_score=[]
neu_word_score=[]
neg_word_score=[]
for i, word in enumerate(reviews.preprocessed_review.tolist()):
    temp= sid.polarity_scores(word)
    pos_word_score.append(temp['pos'])
    neu_word_score.append(temp['neu'])
    neg_word_score.append(temp['neg'])
```

```
reviews['positive_sentiment'] =pos_word_score
reviews['neutral_sentiment'] =neu_word_score
reviews['negative_sentiment'] =neg_word_score
reviews.head()
```

	review_headline	review_body	star_rating	review	review_polarity	review_subjec
0	Very Happy!	As advertised. Everything works perfectly, I'm...	5.0	Very Happy! As advertised. Everything works pe...	0.666667	0.(
1	five star	it's great	5.0	five star it's great	0.800000	0.'
2	great cables	These work great and fit my life proof case fo...	5.0	great cables These work great and fit my life ...	0.666667	0.(
3	Work very well but couldn't get used to not he...	Work very well but couldn't get used to not he	4.0	Work very well but couldn't get used to not he	0.200000	0.:

```
# Deriving average positive, negative and neutral sentiment of the dataset
avg_pos_sentiment = reviews['positive_sentiment'].mean()
avg_neu_sentiment = reviews['neutral_sentiment'].mean()
avg_neg_sentiment = reviews['negative_sentiment'].mean()
print("Average Positive Sentiment of dataset:",avg_pos_sentiment)
print("Average Neutral Sentiment of dataset:",avg_neu_sentiment)
print("Average Negative Sentiment of dataset:",avg_neg_sentiment)
```

```
Average Positive Sentiment of dataset: 0.3263287647715188
Average Neutral Sentiment of dataset: 0.5806806394079025
Average Negative Sentiment of dataset: 0.09294456684502155
```

Topic Modeling using LDA

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 Modeling enables consumers to quickly extract the key topics covered by the reviews without having to go through all of them. It also helps the sellers/retailers get consumer feedback in the form of topics (extracted from the consumer reviews).

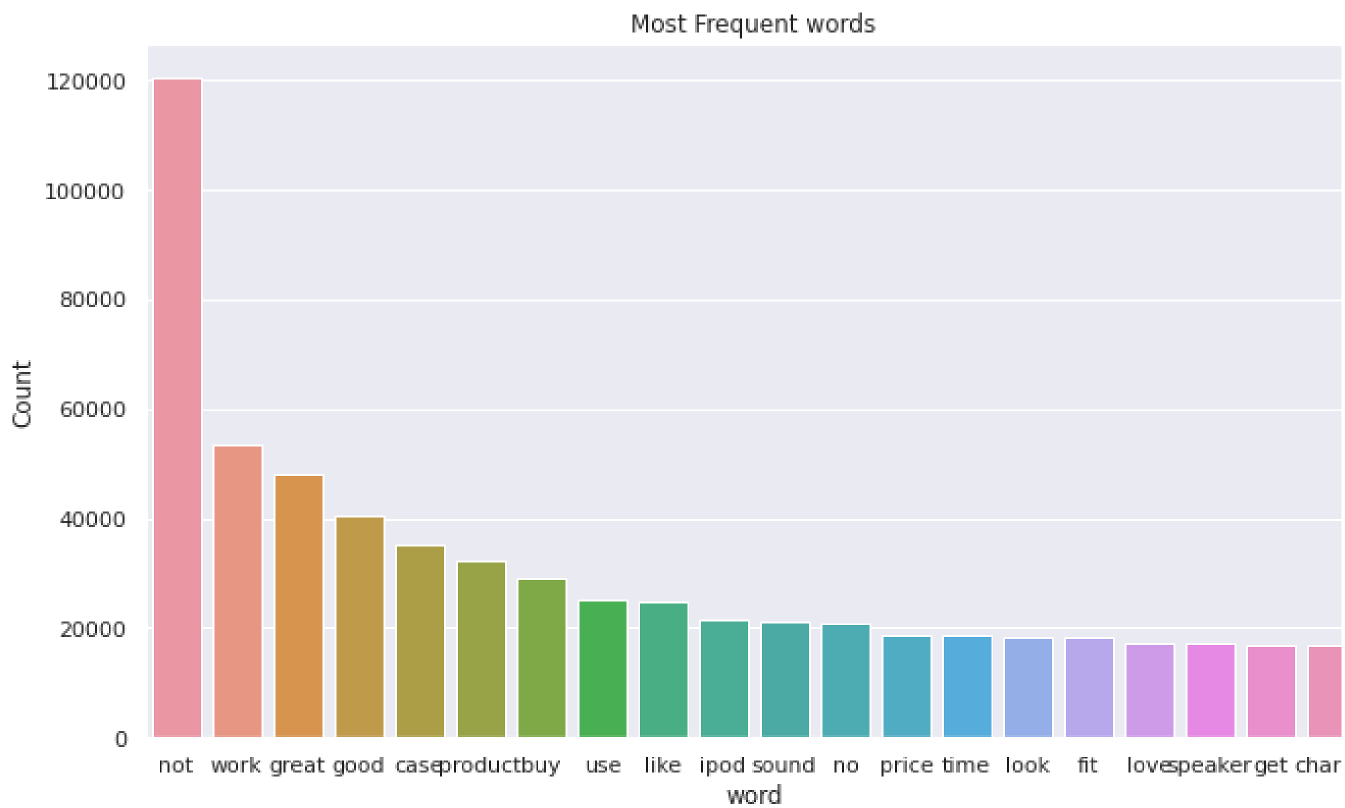
```
# function to plot most frequent terms
def freq_words(x, terms = 30):
  all_words = ' '.join([text for text in x])
  all_words = all_words.split()

  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=(10,6))
  ax = sns.barplot(data=d, x= "word", y = "count")
  ax.set(ylabel = 'Count')
  plt.title("Most Frequent words")
  plt.tight_layout()
  plt.show()
```

```
review_list = [w for w in reviews['preprocessed_review']]
```

manufactured cables that you can purchase from Apple ORG . Only been using for a month DATE

```
freq_words(review_list, 20)
```



Apple ORG cord may be worn and torn but it has never stopped charging. Your results may vary but I'm

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

```
['star', 'great']
```

of fraying and that has not happened to this cord. It looks just like other lightening cables. I have IOS8 and an

```
dictionary = corpora.Dictionary(tokenized_reviews)
```

```
doc_term_matrix = [dictionary.doc2bow(rev) for rev in tokenized_reviews]
```

as well as the Apple ORG branded one CARDINAL Works great Have a iPhone 4s CARDINAL and

```
# Creating the object for LDA model using gensim library
```

```
LDA = gensim.models.ldamodel.LdaModel

# Building LDA model
lda_model = LDA(corpus=doc_term_matrix, id2word=dictionary, num_topics=10, random_state=100,
                chunksize=1000, passes=25)
```

for a couple months DATE , not worn or broken logit Stopped being recognized by my 3 CARDINAL

```
# print topics
lda_model.print_topics()
```

```
[(0,
  '0.048*"scratch" + 0.040*"ok" + 0.031*"dock" + 0.028*"clean" + 0.026*"sub" + 0.022*"ch
 (1,
  '0.041*"charge" + 0.031*"battery" + 0.031*"charger" + 0.029*"protector" + 0.024*"not"
 (2,
  '0.093*"not" + 0.024*"work" + 0.022*"buy" + 0.019*"product" + 0.013*"get" + 0.011*"tim
 (3,
  '0.042*"player" + 0.033*"not" + 0.022*"ipod" + 0.021*"mp3" + 0.020*"use" + 0.019*"play
 (4,
  '0.024*"screen" + 0.020*"video" + 0.015*"mount" + 0.014*"light" + 0.012*"software" + 0
 (5,
  '0.020*"gps" + 0.019*"unit" + 0.017*"book" + 0.015*"transmitter" + 0.015*"protection"
 (6,
  '0.081*"case" + 0.042*"ipod" + 0.034*"not" + 0.033*"fit" + 0.026*"cover" + 0.024*"scre
 (7,
  '0.030*"radio" + 0.018*"unit" + 0.015*"car" + 0.014*"wire" + 0.010*"signal" + 0.010*"1
 (8,
  '0.125*"great" + 0.089*"product" + 0.054*"good" + 0.051*"work" + 0.044*"price" + 0.022
 (9,
  '0.051*"sound" + 0.035*"speaker" + 0.028*"good" + 0.023*"great" + 0.022*"quality" + 0
```

since day one **DATE** . They do exactly what I want them to do. Excellent quality Works great to charge m

The Topic 1 has terms like 'charge', 'charger', 'battery' indicating that the topic is very much related to phone charging. Similarly, Topic 8 seems to be about the overall value of the product as it has terms like 'excellent', 'great', and 'recommend'

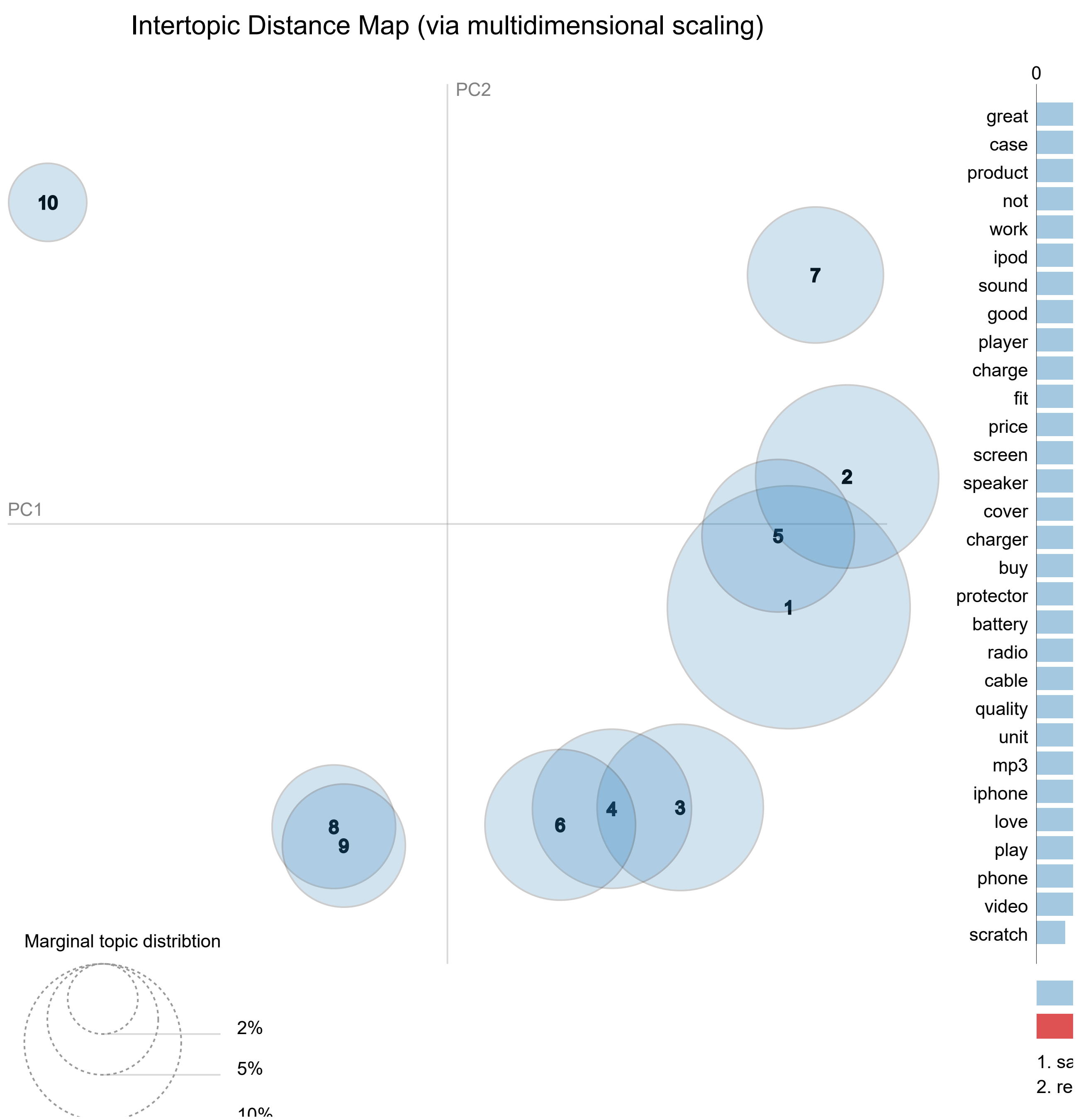
great and has been effectively charging the iPad **PRODUCT** . I like the length also because my daughter

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

yanked from the wall socket. one **CARDINAL** cable works fine, the other cable (regardless of the power

```
# Visualize the topics
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, doc_term_matrix, dictionary)
vis
```

Selected Topic: 0 Previous Topic Next Topic Clear Topic Slid



Classification Models

```
# Defining features and labels
review_data = reviews.copy()
y = review_data['star_rating'].values
y = y.astype('int')
X = review_data['preprocessed_review']
```

```
# Tfidf vectorizer
```

```
vectorizer = TfidfVectorizer(
        min_df=2,
        max_df=0.95,
        ngram_range = (1,4),
        stop_words = 'english',
        )
# Extract features from reviews.
review_features = vectorizer.fit_transform(X)
```

:) Got this cable a while ago but haven't used it heavily yet. I noticed it doesn't charge sometimes on one of tr

```
# Split the dataset to be 80% training and 20% testing
X_train, X_test, y_train, y_test = train_test_split(review_features,
                                                    y,
                                                    stratify=y,
                                                    random_state = 42,
                                                    test_size = 0.2
                                                    )
print(X_train.shape)
```

```
(83877, 520053)
```

```
# Logistic Regression model
lr = LogisticRegression(random_state=22).fit(X_train, y_train)
print( classification_report(y_test, lr.predict(X_test), digits=4))
```

```
              precision    recall  f1-score   support

           1     0.6684    0.8002    0.7284      3514
           2     0.3813    0.0671    0.1142      1460
           3     0.4792    0.2073    0.2894      1944
           4     0.4304    0.2499    0.3162      3613
           5     0.7125    0.9260    0.8054     10439

    accuracy                         0.6620     20970
   macro avg     0.5344    0.4501    0.4507     20970
weighted avg     0.6118    0.6620    0.6122     20970
```

works as intended. Do not buy other cheap brand cables I have made that mistake in the past. Trust the othe

```
# MultinomialNB model
nb = MultinomialNB()
nb.fit(X_train, y_train)
print( classification_report(y_test, nb.predict(X_test), digits=4))
```

```
              precision    recall  f1-score   support

           1     0.8732    0.2783    0.4221      3514
           2     0.0000    0.0000    0.0000      1460
           3     0.7222    0.0067    0.0133      1944
           4     0.6471    0.0030    0.0061      3613
           5     0.5263    0.9989    0.6894     10439

    accuracy                         0.5451     20970
   macro avg     0.5538    0.2574    0.2262     20970
```

```
weighted avg     0.5867    0.5451    0.4162     20970
```

```
# SGDClassifier
sgd = SGDClassifier(random_state=22)
sgd.fit(X_train, y_train)
print( classification_report(y_test, sgd.predict(X_test), digits=4))
```

```
              precision    recall  f1-score   support

           1     0.6372    0.7866    0.7040      3514
           2     0.2708    0.0089    0.0172      1460
           3     0.5647    0.1055    0.1777      1944
           4     0.4252    0.0551    0.0975      3613
           5     0.6469    0.9762    0.7782     10439

    accuracy                         0.6377     20970
   macro avg     0.5090    0.3864    0.3549     20970
weighted avg     0.5733    0.6377    0.5398     20970
```

phone while charging. Phone is basically unusable when plugged in. The page you are viewing bounces all

```
# RandomForestClassifier
forest = RandomForestClassifier(n_estimators=25, criterion="entropy", random_state=42)
forest.fit(X_train, y_train)
print( classification_report(y_test, forest.predict(X_test), digits=4))
```

```
              precision    recall  f1-score   support

           1     0.6490    0.6713    0.6600      3514
           2     0.2812    0.0123    0.0236      1460
           3     0.4367    0.0514    0.0920      1944
           4     0.3326    0.0435    0.0769      3613
           5     0.6101    0.9685    0.7486     10439

    accuracy                         0.6077     20970
   macro avg     0.4619    0.3494    0.3202     20970
weighted avg     0.5299    0.6077    0.5067     20970
```

```
# LinearSVC
svc = LinearSVC(C = 20, class_weight= 'balanced')
svc.fit(X_train, y_train)
print( classification_report(y_test, svc.predict(X_test), digits=4))
```

```
              precision    recall  f1-score   support

           1     0.6602    0.7271    0.6920      3514
           2     0.2690    0.1555    0.1970      1460
           3     0.3235    0.2320    0.2702      1944
           4     0.3577    0.3006    0.3267      3613
           5     0.7401    0.8384    0.7862     10439

    accuracy                         0.6233     20970
```

```
   macro avg     0.4701    0.4507    0.4544     20970
weighted avg     0.5894    0.6233    0.6024     20970
```

CARDINAL pack and it was a great price and they do the job well. Not like some less expensive cords, thes

Model Tuning

We performed parameter tuning using sklearn's GridSearchCV algorithm. GridSearchCV does an exhaustive search over specified parameter values.

```
from sklearn.model_selection import GridSearchCV
```

bought the 2 CARDINAL pack. One CARDINAL cable works very well in my car and hasn't failed me as

```
parameters = { 'alpha': [0.1, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2] }
nb_clf = GridSearchCV(MultinomialNB(), parameters)
nb_clf.fit(X_train, y_train)
```

```
GridSearchCV(cv=None, error_score=nan,
             estimator=MultinomialNB(alpha=1.0, class_prior=None,
                                     fit_prior=True),
             iid='deprecated', n_jobs=None,
             param_grid={'alpha': [0.1, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75,
                                   2]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

on these cords and have broken others in just days DATE . Price is low in comparison. I've ordered 2

```
# Print out the parameters GridSearchCV decided on.
nb_clf.best_params_
```

```
{'alpha': 0.1}
```

a few weeks DATE we are happy with them. However, these cords would not charge a friend's iphone 5

```
parameters = [{ 'loss': ['hinge', 'log', 'perceptron'],
                'alpha': 10.0**-np.arange(1,7),
                'penalty': ['l1', 'l2', 'elasticnet']}]
sgd_clf = GridSearchCV(SGDClassifier(random_state=22), parameters)
sgd_clf.fit(X_train, y_train)
```

```
GridSearchCV(cv=None, error_score=nan,
             estimator=SGDClassifier(alpha=0.0001, average=False,
                                     class_weight=None, early_stopping=False,
                                     epsilon=0.1, eta0=0.0, fit_intercept=True,
                                     l1_ratio=0.15, learning_rate='optimal',
                                     loss='hinge', max_iter=1000,
                                     n_iter_no_change=5, n_jobs=None,
                                     penalty='l2', power_t=0.5, random_state=22,
                                     shuffle=True, tol=0.001,
                                     validation_fraction=0.1, verbose=0,
                                     warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid=[{'alpha': array([1.e-01, 1.e-02, 1.e-03, 1.e-04, 1.e-05, 1.e-06
```

```
                          'loss': ['hinge', 'log', 'perceptron'],
                          'penalty': ['l1', 'l2', 'elasticnet']}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

wires.....but it just stopped working. We'll see how the other one does, but it's not looking to promising. No

```
# Print out the parameters GridSearchCV decided on.
sgd_clf.best_params_
```

```
{'alpha': 1e-05, 'loss': 'log', 'penalty': 'l1'}
```

went through wash & dry **ORG** . Still works great! Can't beat that, include a 1 year **DATE** warranty!

```
classifiers = [lr, nb, sgd, forest, svc]
names_of_classifiers = ["Logistic Regression","Multinomial Naive Bayes","SGD", "Random Forest
```

cases for iPhone 6/6 plus As described, worked w/out error msg & charges quickly! very good quality and wa

```
# Create, fit and predict the star ratings using the Machine Learning classifiers
acc_score=[]

for index,clf in enumerate(classifiers):

    y_pred = clf.predict(X_test)
    acc= accuracy_score(y_test, y_pred)
    acc_score.append(acc*100)

    if index == len(classifiers) - 1:
      cm=confusion_matrix(y_test,y_pred)
```

```
import os
```

CARDINAL stars. Works great so far. These are Apple **ORG** Certified! DO NOT BUY CORDS FOR YOU

```
# Plotting the accuracy comparison
x = names_of_classifiers
acc = acc_score
fig, ax = plt.subplots()
width = 0.75 # the width of the bars
ind = np.arange(len(acc))  # the x locations for the groups
ax.barh(ind, acc, width, color=(0.2, 0.4, 0.6, 0.6))
ax.set_yticks(ind+width/2)
ax.set_yticklabels(x, minor=False)
for i, v in enumerate(acc):
    temp = float(v)
    ax.text(0.5, i+0.1,str(round(v,2)))
plt.title('Accuracy Comparsion')
plt.xlabel('Accuracy')
plt.ylabel('Classifiers')
#plt.show()
plt.savefig(os.path.join('test.png'), dpi=300, format='png', bbox_inches='tight')
```

Package came on time everything arrived as outlined in listing. Good! These actually work!!! As expected

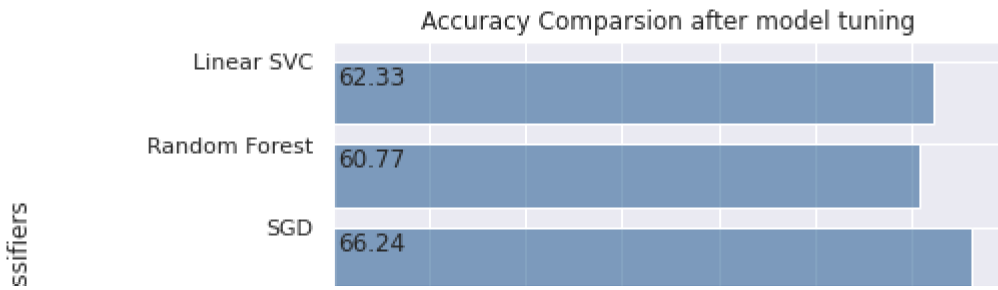
```
# print classification reports
```

```
print("\nclassification reports: ")
print( "LogisticRegression: " )
print( classification_report(y_test, lr.predict(X_test), digits=4))
print( "MultinomialNB: " )
print( classification_report(y_test, nb_clf.predict(X_test), digits=4))
print( "SGDClassifier: " )
print( classification_report(y_test, sgd_clf.predict(X_test), digits=4))
print( "Random Forest: " )
print( classification_report(y_test, forest.predict(X_test), digits=4))
print( "SVC: " )
print( classification_report(y_test, svc.predict(X_test), digits=4))
```

```
classification reports: 
LogisticRegression: 
              precision    recall  f1-score   support

           1     0.6684    0.8002    0.7284      3514
           2     0.3813    0.0671    0.1142      1460
           3     0.4792    0.2073    0.2894      1944
           4     0.4304    0.2499    0.3162      3613
           5     0.7125    0.9260    0.8054     10439

    accuracy                         0.6620     20970
   macro avg     0.5344    0.4501    0.4507     20970
weighted avg     0.6118    0.6620    0.6122     20970

MultinomialNB: 
              precision    recall  f1-score   support

           1     0.6479    0.7610    0.6999      3514
           2     0.3100    0.0212    0.0397      1460
           3     0.4005    0.0880    0.1442      1944
           4     0.3522    0.2380    0.2841      3613
           5     0.6855    0.9111    0.7824     10439

    accuracy                         0.6317     20970
   macro avg     0.4792    0.4039    0.3901     20970
weighted avg     0.5692    0.6317    0.5718     20970

SGDClassifier: 
              precision    recall  f1-score   support

           1     0.6588    0.8011    0.7230      3514
           2     0.3309    0.0315    0.0575      1460
           3     0.5050    0.2068    0.2934      1944
           4     0.4511    0.2209    0.2965      3613
           5     0.7025    0.9417    0.8047     10439

    accuracy                         0.6624     20970
   macro avg     0.5297    0.4404    0.4350     20970
weighted avg     0.6077    0.6624    0.6040     20970

Random Forest: 
              precision    recall  f1-score   support
```

```
           1     0.6490    0.6713    0.6600      3514
           2     0.2812    0.0123    0.0236      1460
           3     0.4367    0.0514    0.0920      1944
           4     0.3326    0.0435    0.0769      3613
           5     0.6101    0.9685    0.7486     10439

    accuracy                         0.6077     20970
   macro avg     0.4619    0.3494    0.3202     20970
weighted avg     0.5299    0.6077    0.5067     20970

SVC:
              precision    recall  f1-score   support

           1     0.6602    0.7271    0.6920      3514
           2     0.2690    0.1555    0.1970      1460
```

```
# Create, fit and predict the star ratings using the Machine Learning classifiers
acc_score=[]
refined_classifiers = [lr, nb_clf, sgd_clf, forest, svc]
for index,clf in enumerate(refined_classifiers):

    y_pred = clf.predict(X_test)
    acc= accuracy_score(y_test, y_pred)
    acc_score.append(acc*100)

    if index == len(refined_classifiers) - 1:
      cm=confusion_matrix(y_test,y_pred)

x = ["Logistic Regression","Multinomial Naive Bayes","SGD", "Random Forest", "Linear SVC"]
acc = acc_score
fig, ax = plt.subplots()
width = 0.75 # the width of the bars
ind = np.arange(len(acc))  # the x locations for the groups
ax.barh(ind, acc, width, color=(0.2, 0.4, 0.6, 0.6))
ax.set_yticks(ind+width/2)
ax.set_yticklabels(x, minor=False)
for i, v in enumerate(acc):
    temp = float(v)
    ax.text(0.5, i+0.1,str(round(v,2)))
plt.title('Accuracy Comparsion after model tuning')
plt.xlabel('Accuracy')
plt.ylabel('Classifiers')
#plt.show()
plt.savefig(os.path.join('test.png'), dpi=300, format='png', bbox_inches='tight')
```




```
# Plotting confusion matrix
def plot_confusion_matrix(cm, title='Confusion matrix'):
    plt.figure( figsize=(9,4))
    plt.imshow(cm, interpolation='nearest')
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(set(y)))
    plt.xticks(tick_marks, set(y), rotation=45)
    plt.yticks(tick_marks, set(y))
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

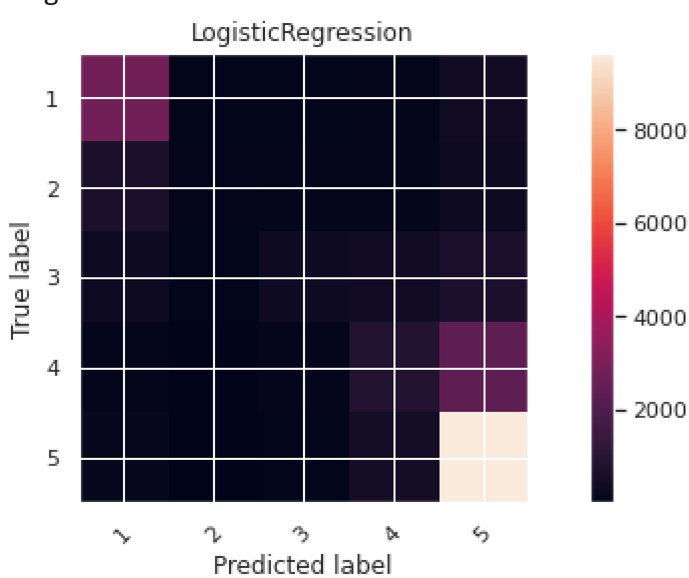
rarely used has already stopped working. Not happy with this product. Works! Great and much cheaper

```
def create_and_print_confusion_matrix(y_test, predicted, title):
    cm = confusion_matrix(y_test, predicted)
    np.set_printoptions(precision=2)
    plt.figure()
    plot_confusion_matrix(cm, title)
    plt.show()
```

bought 2 **CARDINAL** of these cables for an iphone 5s **CARDINAL** , and iphone 5.
Works great, ve

```
create_and_print_confusion_matrix(y_test, lr.predict(X_test), "LogisticRegression")
```

```
<Figure size 432x288 with 0 Axes>
```

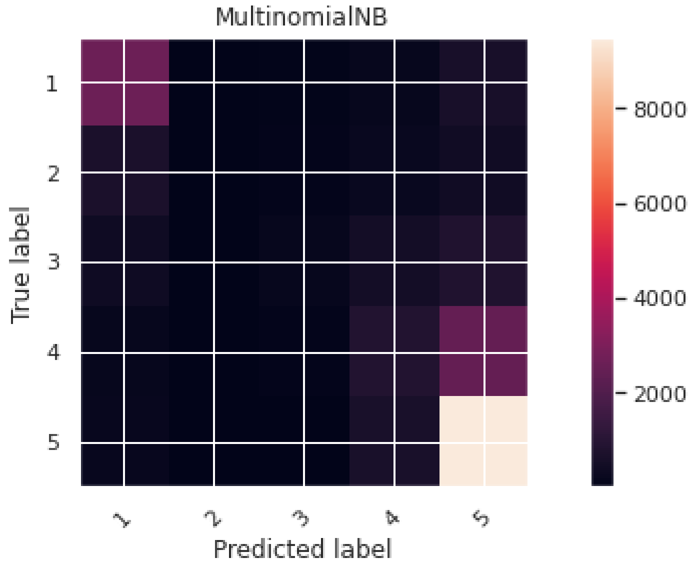


they don't last (they're the same junk cables you one can find on Amazon **ORG** for $ 2-$5 **MONEY**). I

```
# Create MultinomialNB confusion matrix.
create_and_print_confusion_matrix(y_test, nb_clf.predict(X_test),"MultinomialNB")
```

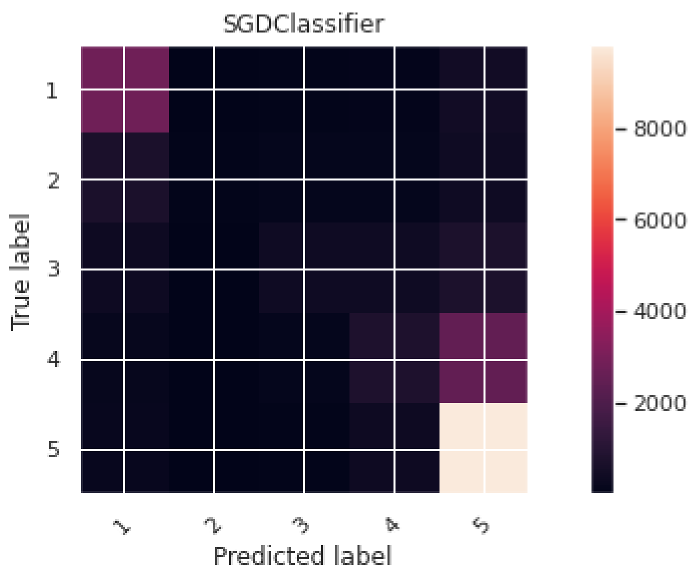
```
create_and_print_confusion_matrix(y_test, nb_clf.predict(X_test), "MultinomialNB")
```

```
<Figure size 432x288 with 0 Axes>
```



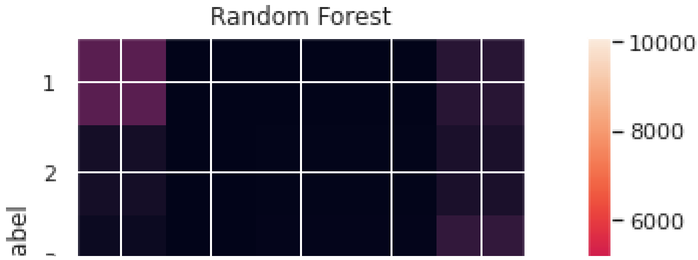

```
# Create SGDClassifier confusion matrix.
create_and_print_confusion_matrix(y_test, sgd_clf.predict(X_test), "SGDClassifier")
```

```
<Figure size 432x288 with 0 Axes>
```



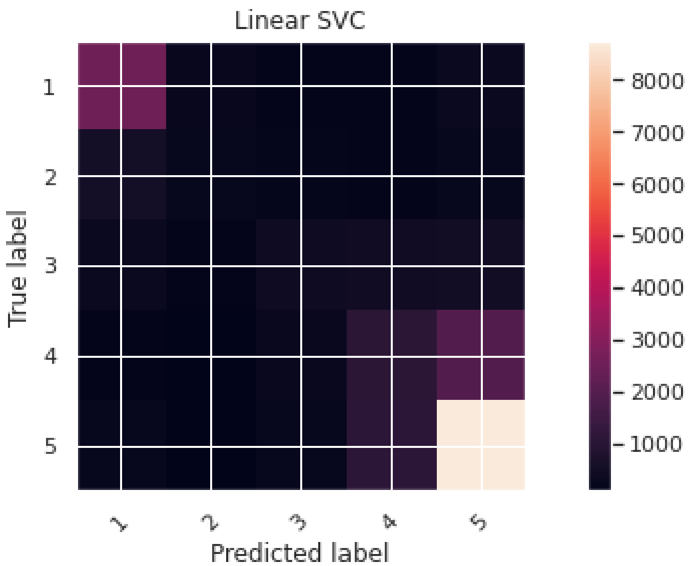

```
create_and_print_confusion_matrix(y_test, forest.predict(X_test), "Random Forest")
```

<Figure size 432x288 with 0 Axes>




```
create_and_print_confusion_matrix(y_test, svc.predict(X_test), "Linear SVC")
```

<Figure size 432x288 with 0 Axes>



REFERENCES:

Band, A. (2020, September 27). Text2emotion: Python package to detect emotions from textual data. Retrieved November 29, 2020, from https://towardsdatascience.com/text2emotion-python-package-to-detect-emotions-from-textual-data-b2e7b7ce1153

Confusion Matrix in Machine Learning. (2020, August 21). Retrieved November 29, 2020, from https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

Joshi, P. (2020, June 25). SpaCy Tutorial: SpaCy For NLP: SpaCy NLP Tutorial. Retrieved November 29, 2020, from https://www.analyticsvidhya.com/blog/2020/03/spacy-tutorial-learn-natural-language-processing/

Kadam, S. (2020, April 18). Generating Word Cloud in Python. Retrieved November 29, 2020, from https://www.geeksforgeeks.org/generating-word-cloud-python/

Logistic Regression 3-class Classifier¶. (n.d.). Retrieved November 29, 2020, from https://scikit-learn.org/stable/auto_examples/linear_model/plot_iris_logistic.html?highlight=logistic+regression

Notebook on nbviewer. (n.d.). Retrieved November 29, 2020, from https://nbviewer.jupyter.org/github/bmabey/hacker_news_topic_modelling/blob/master/HN Topic

Model Talk.ipynb

Rai, A. (2019, January 23). Python: Sentiment Analysis using VADER. Retrieved November 29, 2020, from https://www.geeksforgeeks.org/python-sentiment-analysis-using-vader/

Simplified Text Processing¶. (n.d.). Retrieved November 29, 2020, from https://textblob.readthedocs.io/en/dev/

Sklearn.model_selection.GridSearchCV¶. (n.d.). Retrieved November 29, 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Only had this set a few weeks **DATE** , but seem sturdier than other brands I have tried. Also comes with a 12 month **DATE** warranty i was not aware of, which is good because it seems like the replacement ones I bought in the past wear out far sooner than that.
SIDE NOTE: LIKE THIS IF YOU WANT APPLE THE CHANGE THEIR CHARGERS. I Love apple, but I soon be switching if these new chargers continue to fail so quickly, but until then, these seem to be a good quality cable Good price, fit my case (which most non Apple **ORG** chargers doesnt) and no problems in a month **DATE** of using it daily I tried a lot of different &# 34;alternate&#34 **MONEY** ; iPhone 5 **CARDINAL** cords and all failed - until these - I love them! They're great, solid product! Seems to work really well with my i Devices. No &# 34;Compatibility&#34 **MONEY** ; messages :-) Well made and durable so far. We have 2 **CARDINAL** iPhone **ORG** 6's and and iPad. So far these cords work great on all the devices. Great price for a good product. Will buy again if/when we need additional cords. Well made cable. I give it 4 **CARDINAL** instead of 5 **CARDINAL** because after using it for about 3 weeks **DATE** my iPhone 6 **CARDINAL** is now telling me it is not a compatible device. I