Twitter Sentimental Analysis

Submitted in partial fulfilment of the

requirements of the degree of

Bachelor of Engineering

by

Masood Khan (3117021)

Ashhar Shaikh (3117052)

Faraz Shaikh (3117053)

Needa Shaikh (3117057)

Supervisor (s):

Prof. Waheeda Dhokley

Computer Engineering/M.H Saboo Siddik College of Engineering.

Mumbai University

2019-20

CERTIFICATE

This is to certify that the project entitled **"Twitter Sentimental Analysis"** is a bonafide work of **"Masood Khan (3117021)", "Ashhar Shaikh(3117052)", "Faraz Shaikh(3117053)", "Needa Shaikh (3117057)"** submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of **"Undergraduate"** in **"Bachelor of Computer Engineering".**

Prof. Waheeda Dhokley

Supervisor

Dr. Zainab Pirani

Head of

Department

ACKNOWLEDGEMENT

It is our esteemed pleasure to present the project report on the topic “Twitter Sentimental Analysis”.

We would firstly like to thank our Head of the Department Dr Zainab Pirani for encouraging

and motivating us with her guidance and total support for our work.

We shall also like to thank Mrs. Waheeda Dhokley for working as our guide and making our path to integrity much simpler.

We also thank all the teachers who constantly motivated us and provided us their precious

knowledge about the procedures carried out for making a report along with technical

knowledge they have availed.

We would also like to thank our principal Dr. Ganesh Kame for providing us this opportunity of integrating our own project and constantly supporting us throughout the

process.

It would also be pleasure thanking all the staff, be it teaching or non-teaching who always

understood by us and never made any problem tread our way.

ABSTRACT

As humans, we are able to classify text into positive/negative subconsciously. For example, the sentence “The kid had a gorgeous smile on his face”, will most likely give us a positive sentiment. In layman’s terms, we kind of arrive to such conclusion by examining the words and averaging out the positives and the negatives. For instance, the words “gorgeous” and “smile” are more likely to be positive, while words like “the”, “kid” and “face” are really neutral. Therefore, the overall sentiment of the sentence is likely to be positive.

A common use for this technology comes from its deployment in the social media space to discover how people feel about certain topics, particularly through users’ word-of-mouth in textual posts, or in the context of Twitter, their *tweets*.

Thesis Approval for Project Report Approval for B. E.

This project report entitled Twitter Sentimental Analysis by **Masood Khan, Ashhar Shaikh, Faraz Shaikh, Needa Shaikh** is approved for the degree of Bachelor in Computer Engineering.

Examiners

1. Prof. Waheeda Dhokley

Date: 23 April 2020

Place: Mumbai

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

Masood Khan (3117021)

Ashhar Shaikh (3117052)

Faraz Shaikh (3117053)

Needa Shaikh (3117057)

Date: 23 April 2020

Place: Mumbai

Chapter 1)

Introduction

Social media has profound impact in capturing the potential customers and thus there are a lot of consulting firms that operate in the digital strategy space. Whether it is to design a marketing campaign or look at the effect of marketing campaigns on user engagement or sentiment, it is a very valuable tool.

Natural Language Processing (NLP) is a hotbed of research in data science these days and one of the most common applications of NLP is sentiment analysis. From opinion polls to creating entire marketing strategies, this domain has completely reshaped the way businesses work, which is why this is an area every data scientist must be familiar with.

Thousands of text documents can be processed for sentiment (and other features including named entities, topics, themes, etc.) in seconds, compared to the hours it would take a team of people to manually complete the same task.

We will do so by following a sequence of steps needed to solve a general sentiment analysis problem. We will start with preprocessing and cleaning of the raw text of the tweets. Then we will explore the cleaned text and try to get some intuition about the context of the tweets. After that, we will extract numerical features from the data and finally use these feature sets to train models and identify the sentiments of the tweets.

We have worked upon sentimental analysis of twitter comments and predicting results upon classification as 0,1,2,3 as positive, negative, neutral and can't say respectively. The Machine Learning Algorithm used is Support Vector Machine (SVM) .We have demonstrated all plots such as Word Cloud, Count plot, Training vs Testing Accuracy plot and metrics count including confusion metrics.

Chapter 2)

Sentimental Analysis

Sentiment analysis is a text analysis method that detects polarity (e.g. a positive or negative opinion) within text, whether a whole document, paragraph, sentence, or clause.

Understanding people's emotions is essential for businesses since customers are able to express their thoughts and feelings more openly than ever before. By automatically analyzing customer feedback, from survey responses to social media conversations, brands are able to listen attentively to their customers, and tailor products and services to meet their needs.

For example, using sentiment analysis to automatically analyze 4,000+ reviews about your product could help you discover if customers are happy about your pricing plans and customer service.

Why Perform Sentiment Analysis?

It's estimated that 80% of the world's data is unstructured, in other words it's unorganized. Huge volumes of text data (emails, support tickets, chats, social media conversations, surveys, articles, documents, etc), is created every day but it's hard to analyze, understand, and sort through, not to mention time-consuming and expensive.

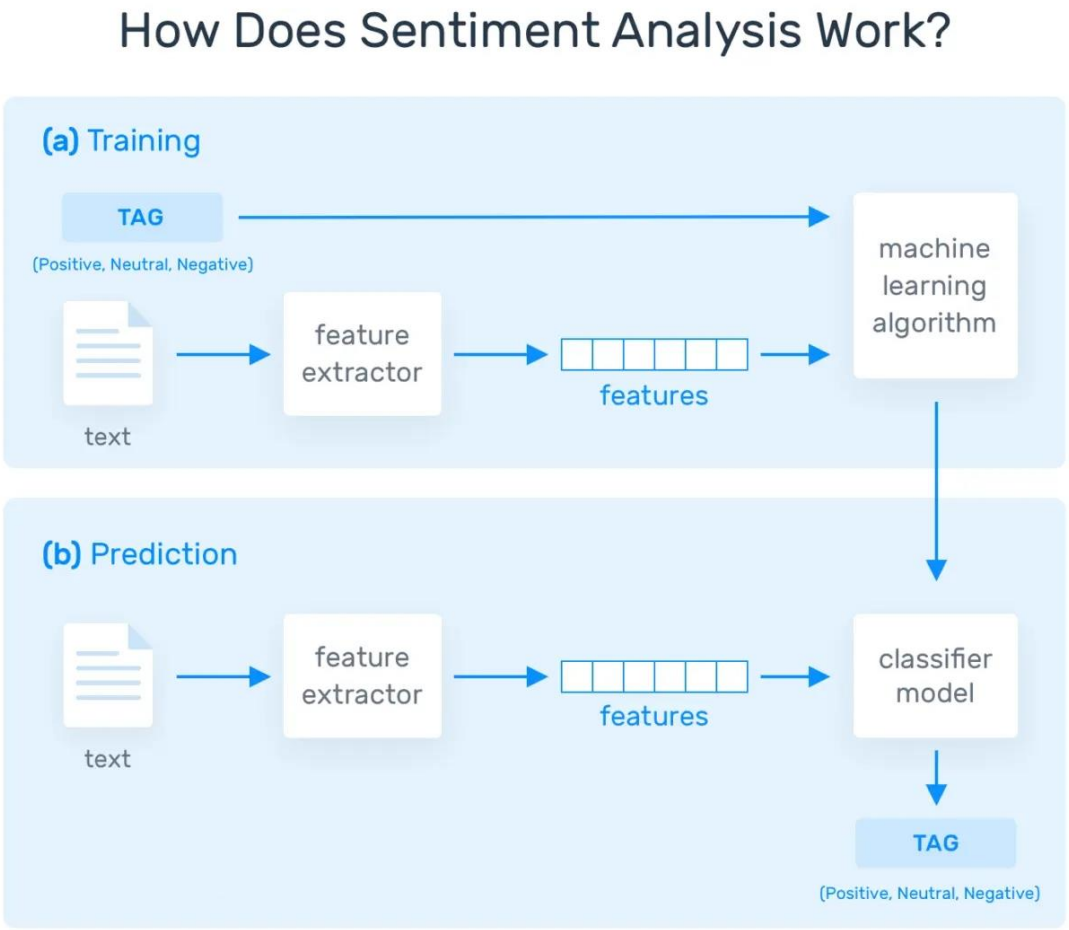
Sentiment analysis, however, helps businesses make sense of all this unstructured text by automatically tagging it.

Benefits of sentiment analysis include:

Sorting Data at Scale Can you imagine manually sorting through thousands of tweets, customer support conversations, or surveys? There's just too much data to process manually. Sentiment analysis helps businesses process huge amounts of data in an efficient and cost-effective way.

Real-Time Analysis Sentiment analysis can identify critical issues in real-time, for example is a PR crisis on social media escalating? Is an angry customer about to churn? Sentiment analysis models can help you immediately identify these kinds of situations and gauge brand sentiment, so you can take action right away.

Consistent criteria It's estimated that people only agree around 60-65% of the time when determining the sentiment of a particular text. Tagging text by sentiment is highly subjective, influenced by personal experiences, thoughts, and beliefs. By using a centralized sentiment analysis system, companies can apply the same criteria to all of their data, helping them improve accuracy and gain better insights.



How Accurate Is Sentiment Analysis?

Here's what sentiment analysis is: it's a tremendously difficult task even for human beings. That said, sentiment analysis classifiers might not be as precise as other types of classifiers. Remember that inter-annotator agreement is pretty low and that machines learn from the data they are fed with (see above).
That said, you might be saying, is it worth the effort? The answer is simple: it sure is worth it! Chances are that sentiment analysis predictions will be wrong from time to time, but by using sentiment analysis you will get the opportunity to get it right about 70-80% of the times you submit your texts for classification. If you or your company have not used sentiment analysis before, then you'll see some improvement really quickly. For typical use cases, such as ticket routing, brand monitoring, and VoC analysis (see below), this means you will save a lot of time and money -which you are likely to be investing in in-house manual work nowadays,- save your teams some frustration, and increase your (or your company's) productivity.

Chapter 3)

The Training and Prediction Processes.

In the training process (a), our model learns to associate a particular input (i.e. a text) to the corresponding output (tag) based on the test samples used for training. The feature extractor transfers the text input into a feature vector. Pairs of feature vectors and tags (e.g. positive, negative, or neutral) are fed into the machine learning algorithm to generate a model.
In the prediction process (b), the feature extractor is used to transform unseen text inputs into feature vectors. These feature vectors are then fed into the model, which generates predicted tags (again, positive, negative, or neutral).

Feature Extraction from Text

The first step in a machine learning text classifier is to transform the text extraction or text vectorization, and the classical approach has been bag-of-words or bag-of-ngrams with their frequency.
More recently, new feature extraction techniques have been applied based on word embeddings (also known as word vectors). This kind of representations makes it possible for words with similar meaning to have a similar representation, which can improve the performance of classifiers.

Classification Algorithms

The classification step usually involves a statistical model like Naïve Bayes, Logistic Regression, Support Vector Machines, or Neural Networks:

Naïve Bayes: a family of probabilistic algorithms that uses Bayes's Theorem to predict the category of a text.

Linear Regression: a very well-known algorithm in statistics used to predict some value (Y) given a set of features (X).

Support Vector Machines: a non-probabilistic model which uses a representation of text examples as points in a multidimensional space. Examples of different categories (sentiments) are mapped to distinct regions within that space. Then, new texts are assigned a category based on similarities with existing texts and the regions they're mapped to.

Deep Learning: a diverse set of algorithms that attempt to mimic the human brain, by employing artificial neural networks to process data.

Chapter 4)

Data Analysis and Data Wrangling.

Import Packages

In [219]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [220]:
```
from scipy.stats import pearsonr
import sklearn.ensemble
from nltk.corpus import stopwords
from textblob import TextBlob
from nltk.stem import PorterStemmer
from textblob import Word
```

In [221]:
```
df = pd.read_csv('train.csv')
df1 = pd.read_csv('test.csv')# use / not this \

# df = pd.merge(
#             df,
#             df1,
#             how='outer',
#             on='tweet_id',
#             indicator=True
# )
```

In [222]:
```
df.head(10)
```

Out[222]:

	tweet_id	tweet	sentiment
0	1701	#sxswnui #sxsw #apple defining language of tou...	1
1	1851	Learning ab Google doodles! All doodles should...	1
2	2689	one of the most in-your-face ex. of stealing t...	2
3	4525	This iPhone #SXSW app would b pretty awesome i...	0
4	3604	Line outside the Apple store in Austin waiting...	1
5	966	#technews One lone dude awaits iPad 2 at Apple...	1
6	1395	SXSW Tips, Prince, NPR Videos, Toy Shopping Wi...	1
7	8182	NU user RT @mention New #UberSocial for #iPhon...	1
8	8835	Free #SXSW sampler on iTunes {link} #FreeMusic	2
9	883	I think I might go all weekend without seeing ...	2

In [223]:
```
df1.head()
```

Out[223]:

	tweet_id	tweet
0	7506	Audience Q: What prototyping tools do you use?...
1	7992	At SXSW? Send Your Best Photos & Videos to...
2	247	@mention and here's a pic of you winning your...
3	7688	Google Marissa Mayer: mobile phone as a cursor...
4	3294	#SXSW Google maps is even cooler than I thought

In [224]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1819 entries, 0 to 1818
Data columns (total 2 columns):
tweet_id    1819 non-null int64
tweet       1819 non-null object
dtypes: int64(1), object(1)
memory usage: 28.5+ KB
```

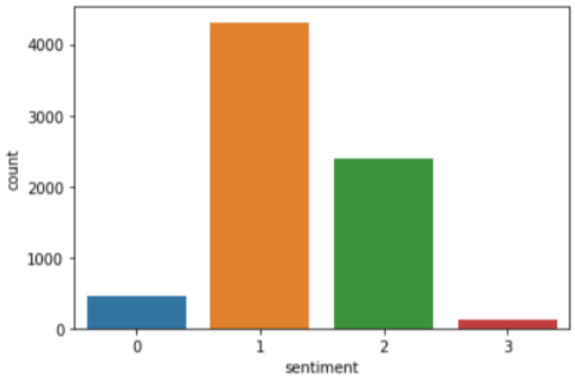
Analyzing The Data

In [225]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7274 entries, 0 to 7273
Data columns (total 3 columns):
tweet_id     7274 non-null int64
tweet        7273 non-null object
sentiment    7274 non-null int64
dtypes: int64(2), object(1)
memory usage: 170.6+ KB
```

In [226]:
```
sns.countplot(x="sentiment",data=df)
#0: Negative, 1: Neutral, 2: Positive, 3: Can't Tell
```

Out[226]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x212bc85bd68>
```



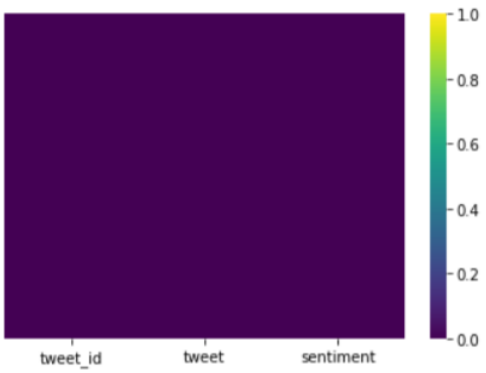
Data Wrangling

In [227]:
```
df.isnull().sum()
```

Out[227]:
```
tweet_id     0
tweet        1
sentiment    0
dtype: int64
```

In [228]:
```
sns.heatmap(df.isnull(),yticklabels=False,cmap="viridis")
#1 Missing Value
```

Out[228]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x212bc65acf8>
```



In [229]:
```
df.drop("tweet_id",axis=1,inplace=True)
```

In [230]:
```
df.head(10)
```

Out[230]:

	tweet	sentiment
0	#sxswnui #sxsw #apple defining language of tou...	1
1	Learning ab Google doodles! All doodles should...	1
2	one of the most in-your-face ex. of stealing t...	2
3	This iPhone #SXSW app would b pretty awesome i...	0
4	Line outside the Apple store in Austin waiting...	1
5	#technews One lone dude awaits iPad 2 at Apple...	1
6	SXSW Tips, Prince, NPR Videos, Toy Shopping Wi...	1
7	NU user RT @mention New #UberSocial for #iPhon...	1
8	Free #SXSW sampler on iTunes {link} #FreeMusic	2
9	I think I might go all weekend without seeing ...	2

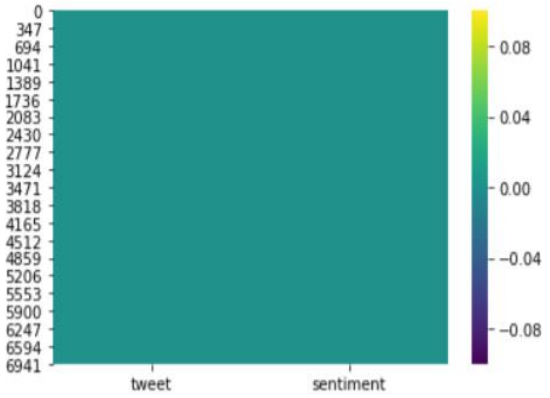
In [231]:
```
df.dropna(inplace=True)
#Dropping The Column
```

In [232]:
```
df.isnull().sum()
```

Out[232]:
```
tweet        0
sentiment    0
dtype: int64
```

In [233]:
```
sns.heatmap(df.isnull(), linecolor="red",cmap="viridis")
# Perfectly Clean Data
```

Out[233]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x212bc4eec50>
```



In [234]:
```
df.head(5)
```

Out[234]:

	tweet	sentiment
0	#sxswnui #sxsw #apple defining language of tou...	1
1	Learning ab Google doodles! All doodles should...	1
2	one of the most in-your-face ex. of stealing t...	2
3	This iPhone #SXSW app would b pretty awesome i...	0
4	Line outside the Apple store in Austin waiting...	1

Chapter 5)

Exploratory Data Analysis.

Removing Punctuations, Numbers, and Special Characters

Punctuations, numbers and special characters do not help much. It is better to remove them from the text just as we removed the twitter handles. Here we will replace everything except characters and hashtags with spaces.

Removing Short Words

We have to be a little careful here in selecting the length of the words which we want to remove. So, I have decided to remove all the words having length 3 or less. For example, terms like “hmm”, “oh” are of very little use. It is better to get rid of them.

Stemming

Stemming is a rule-based process of stripping the suffixes (“ing”, “ly”, “es”, “s” etc) from a word. For example, For example – “play”, “player”, “played”, “plays” and “playing” are the different variations of the word – “play”.

Understanding the common words used in the tweets: WordCloud

Now I want to see how well the given sentiments are distributed across the train dataset. One way to accomplish this task is by understanding the common words by plotting wordclouds.

A wordcloud is a visualization wherein the most frequent words appear in large size and the less frequent words appear in smaller sizes.

Extracting Features from Cleaned Tweets

To analyze a preprocessed data, it needs to be converted into features. Depending upon the usage, text features can be constructed using assorted techniques – Bag-of-Words, TF-IDF, and Word Embeddings. In this article, we will be covering only Bag-of-Words and TF-IDF.

TF-IDF Features

This is another method which is based on the frequency method but it is different to the bag-of-words approach in the sense that it takes into account, not just the occurrence of a word in a single document (or tweet) but in the entire corpus.

TF-IDF works by penalizing the common words by assigning them lower weights while giving importance to words which are rare in the entire corpus but appear in good numbers in few documents.

Let’s have a look at the important terms related to TF-IDF:

- TF = (Number of times term t appears in a document)/(Number of terms in the document)
- IDF = log(N/n), where, N is the number of documents and n is the number of documents a term t has appeared in.
- TF-IDF = TF*IDF

In [235]:
```
df['word_count'] = df['tweet'].apply(lambda x: len(str(x).split(" ")))
df[['tweet','word_count']].head()
```

Out[235]:

	tweet	word_count
0	#sxswnui #sxsw #apple defining language of tou...	12
1	Learning ab Google doodles! All doodles should...	19
2	one of the most in-your-face ex. of stealing t...	23
3	This iPhone #SXSW app would b pretty awesome i...	19
4	Line outside the Apple store in Austin waiting...	15

In [236]:
```
df['char_count'] = df['tweet'].str.len() ## this also includes spaces
df[['tweet','char_count']].head()
```

Out[236]:

	tweet	char_count
0	#sxswnui #sxsw #apple defining language of tou...	89
1	Learning ab Google doodles! All doodles should...	143
2	one of the most in-your-face ex. of stealing t...	132
3	This iPhone #SXSW app would b pretty awesome i...	125
4	Line outside the Apple store in Austin waiting...	77

In [237]:
```
def avg_word(sentence):
  words = sentence.split()
  return (sum(len(word) for word in words)/len(words))

df['avg_word'] = df['tweet'].apply(lambda x: avg_word(x))
df[['tweet','avg_word']].head()
```

Out[237]:

	tweet	avg_word
0	#sxswnui #sxsw #apple defining language of tou...	6.500000
1	Learning ab Google doodles! All doodles should...	6.578947
2	one of the most in-your-face ex. of stealing t...	5.000000
3	This iPhone #SXSW app would b pretty awesome i...	5.631579
4	Line outside the Apple store in Austin waiting...	4.500000

In [238]:
```
stop = stopwords.words('english')
df['stopwords'] = df['tweet'].apply(lambda x: len([x for x in x.split() if x in stop]))
df[['tweet','stopwords']].head()
```

Out[238]:

	tweet	stopwords
0	#sxswnui #sxsw #apple defining language of tou...	2
1	Learning ab Google doodles! All doodles should...	4
2	one of the most in-your-face ex. of stealing t...	7
3	This iPhone #SXSW app would b pretty awesome i...	4
4	Line outside the Apple store in Austin waiting...	4

In [239]:
```
print(stop)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "yo
u'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'sh
e', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'thei
r', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'thes
e', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',
'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'becaus
e', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between',
'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down',
'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'ther
e', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'othe
r', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll',
'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't",
'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't",
'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'should
n', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

In [240]:
```
df['hastags'] = df['tweet'].apply(lambda x: len([x for x in x.split() if x.startswith('#'
)]))
df[['tweet','hastags']].head()
```

Out[240]:

	tweet	hastags
0	#sxswnui #sxsw #apple defining language of tou...	3
1	Learning ab Google doodles! All doodles should...	2
2	one of the most in-your-face ex. of stealing t...	1
3	This iPhone #SXSW app would b pretty awesome i...	3
4	Line outside the Apple store in Austin waiting...	1

In [241]:
```
df.head(5)
```

Out[241]:

	tweet	sentiment	word_count	char_count	avg_word	stopwords	hastags
0	#sxswnui #sxsw #apple defining language of tou...	1	12	89	6.500000	2	3
1	Learning ab Google doodles! All doodles should...	1	19	143	6.578947	4	2
2	one of the most in-your-face ex. of stealing t...	2	23	132	5.000000	7	1
3	This iPhone #SXSW app would b pretty awesome i...	0	19	125	5.631579	4	3
4	Line outside the Apple store in Austin waiting...	1	15	77	4.500000	4	1

In [242]:
```
df['numerics'] = df['tweet'].apply(lambda x: len([x for x in x.split() if x.isdigit()]))
df[['tweet','numerics']].head()
#Total Number Present
```

Out[242]:

	tweet	numerics
0	#sxswnui #sxsw #apple defining language of tou...	0
1	Learning ab Google doodles! All doodles should...	0
2	one of the most in-your-face ex. of stealing t...	0
3	This iPhone #SXSW app would b pretty awesome i...	0
4	Line outside the Apple store in Austin waiting...	0

In [243]:
```
df['upper'] = df['tweet'].apply(lambda x: len([x for x in x.split() if x.isupper()]))
df[['tweet','upper']].head()
#Upper Case Characters Presnt in Datset
```

Out[243]:

	tweet	upper
0	#sxswnui #sxsw #apple defining language of tou...	0
1	Learning ab Google doodles! All doodles should...	0
2	one of the most in-your-face ex. of stealing t...	2
3	This iPhone #SXSW app would b pretty awesome i...	1
4	Line outside the Apple store in Austin waiting...	1

In [244]:
```
df.tail(5)
```

Out[244]:

	tweet	sentiment	word_count	char_count	avg_word	stopwords	hastags	numerics	upper
7269	@mention Google plze Tammi. I'm in middle of ...	1	16	93	5.200000	4	1	0	1
7270	RT @mention □÷¼ Are you all set? □÷_ {link} □÷...	1	15	91	5.133333	2	5	0	1
7271	RT @mention Aha! Found proof of lactation room...	1	22	140	5.409091	5	1	0	2
7272	We just launched our iPad app at #SXSW! Get al...	1	18	92	4.166667	6	1	0	2
7273	The next fin serv battle is vs Apple, GOOG, Mo...	1	23	137	5.000000	4	2	0	2

Chapter 6)

Data Preprocessing and Cleaning.

The preprocessing of the text data is an essential step as it makes the raw text ready for mining, i.e., it becomes easier to extract information from the text and apply machine learning algorithms to it. If we skip this step then there is a higher chance that you are working with noisy and inconsistent data. The objective of this step is to clean noise those are less relevant to find the sentiment of tweets such as punctuation, special characters, numbers, and terms which don't carry much weightage in context to the text.

In one of the later stages, we will be extracting numeric features from our Twitter text data. This feature space is created using all the unique words present in the entire data. So, if we preprocess our data well, then we would be able to get a better quality feature space.

In [245]:
```
df['tweet'] = df['tweet'].apply(lambda x: " ".join(x.lower() for x in x.split()))
df['tweet'].head()
#Making Everything in LowerCase No Repeatations
```

Out[245]:
```
0    #sxswnui #sxsw #apple defining language of tou...
1    learning ab google doodles! all doodles should...
2    one of the most in-your-face ex. of stealing t...
3    this iphone #sxsw app would b pretty awesome i...
4    line outside the apple store in austin waiting...
Name: tweet, dtype: object
```

In [246]:
```
df['tweet'] = df['tweet'].str.replace('[^\w\s]','')
df['tweet'].head()
#REMOVING THE PUNCTUCATION
```

Out[246]:
```
0    sxswnui sxsw apple defining language of touch ...
1    learning ab google doodles all doodles should ...
2    one of the most inyourface ex of stealing the ...
3    this iphone sxsw app would b pretty awesome if...
4    line outside the apple store in austin waiting...
Name: tweet, dtype: object
```

In [247]:
```
stop = stopwords.words('english')
df['tweet'] = df['tweet'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
df['tweet'].head()
# Removing Stopwords
```

Out[247]:
```
0    sxswnui sxsw apple defining language touch dif...
1    learning ab google doodles doodles light funny...
2    one inyourface ex stealing show yrs rt mention...
3    iphone sxsw app would b pretty awesome didnt c...
4    line outside apple store austin waiting new ip...
Name: tweet, dtype: object
```

In [248]:
```
freq = pd.Series(' '.join(df['tweet']).split()).value_counts()[:10]
#Commonly Used Words And Thier Count
```

In [249]:
```
freq
```

Out[249]:
```
sxsw       7540
mention    5512
link       3427
rt         2344
ipad       1912
google     1862
apple      1729
iphone     1215
store      1188
new         862
dtype: int64
```

In [250]:
```
freq = list(freq.index)
df['tweet'] = df['tweet'].apply(lambda x: " ".join(x for x in x.split() if x not in freq))
df['tweet'].head()
#Removing the Common Words
```

Out[250]:
```
0    sxswnui defining language touch different dial...
1    learning ab doodles doodles light funny amp in...
2    one inyourface ex stealing show yrs quotat sch...
3    app would b pretty awesome didnt crash every 1...
4                          line outside austin waiting
Name: tweet, dtype: object
```

In [251]:
```
freq1 = pd.Series(' '.join(df['tweet']).split()).value_counts()[-10:]
# Rare Words From Dataset
```

In [252]:
```
freq1
```

Out[252]:
```
monocle          1
copypaste        1
268              1
txts             1
latinasintech    1
rainjacket       1
selfrespecting   1
lurking          1
tipped           1
cpap             1
dtype: int64
```

In [254]:
```
df['tweet'][:5].apply(lambda x: str(TextBlob(x).correct()))
#Words Correction analytics and analtycs
```

Out[254]:
```
0    sxswnui defining language touch different dial...
1    learning ab doubles doubles light funny amp in...
2    one inyourface ex stealing show yes quotas sch...
3    pp would b pretty awesome didn crash every 10m...
4                          line outside austin waiting
Name: tweet, dtype: object
```

In [255]:
```
TextBlob(df['tweet'][1]).words
```

Out[255]:
```
WordList(['learning', 'ab', 'doodles', 'doodles', 'light', 'funny', 'amp', 'innovative', 'exc
eptions', 'significant', 'occasions', 'googledoodle'])
```

In [256]:
```
st = PorterStemmer()
df['tweet'][:5].apply(lambda x: " ".join([st.stem(word) for word in x.split()]))
#removal of suffices, like “ing”, “ly”, “s”, etc.
```

Out[256]:
```
0    sxswnui defin languag touch differ dialect bec...
1    learn ab doodl doodl light funni amp innov exc...
2    one inyourfac ex steal show yr quotat school m...
3    app would b pretti awesom didnt crash everi 10...
4                              line outsid austin wait
Name: tweet, dtype: object
```

In [257]:
```
df.head(5)
```

Out[257]:

	tweet	sentiment	word_count	char_count	avg_word	stopwords	hastags	numerics	upper
0	sxswnui defining language touch different dial...	1	12	89	6.500000	2	3	0	0
1	learning ab doodles doodles light funny amp in...	1	19	143	6.578947	4	2	0	0
2	one inyourface ex stealing show yrs quotat sch...	2	23	132	5.000000	7	1	0	2
3	app would b pretty awesome didnt crash every 1...	0	19	125	5.631579	4	3	0	1
4	line outside austin waiting	1	15	77	4.500000	4	1	0	1

Chapter 7)

Advanced Text Processing.

TextBlob aims to provide access to common text-processing operations through a familiar interface. You can treat **TextBlob** objects as if they were Python strings that learned how to do Natural Language Processing.

n-grams:

The **TextBlob.ngrams()** method returns a list of tuples of **n** successive words.

In [258]:
```
TextBlob(df['tweet'][0]).ngrams(2)
#N-grams are the combination of multiple words used together.
```

Out[258]:
```
[WordList(['sxswnui', 'defining']),
 WordList(['defining', 'language']),
 WordList(['language', 'touch']),
 WordList(['touch', 'different']),
 WordList(['different', 'dialects']),
 WordList(['dialects', 'becoming']),
 WordList(['becoming', 'smaller'])]
```

In [259]:
```
tf1 = (df['tweet'][1:2]).apply(lambda x: pd.value_counts(x.split(" "))).sum(axis = 0).reset_
index()
tf1.columns = ['words','tf']
tf1
#Term frequency is simply the ratio of the count of a word present in a sentence, to the len
gth of the sentence.
```

Out[259]:

	words	tf
0	doodles	2
1	light	1
2	funny	1
3	significant	1
4	ab	1
5	exceptions	1
6	googledoodle	1
7	innovative	1
8	learning	1
9	occasions	1
10	amp	1

In [259]:
```
tf1 = (df['tweet'][1:2]).apply(lambda x: pd.value_counts(x.split(" "))).sum(axis = 0).reset_
index()
tf1.columns = ['words','tf']
tf1
#Term frequency is simply the ratio of the count of a word present in a sentence, to the len
gth of the sentence.
```

Out[259]:

	words	tf
0	doodles	2
1	light	1
2	funny	1
3	significant	1
4	ab	1
5	exceptions	1
6	googledoodle	1
7	innovative	1
8	learning	1
9	occasions	1
10	amp	1

In [260]:
```
for i,word in enumerate(tf1['words']):
  tf1.loc[i, 'idf'] = np.log(df.shape[0]/(len(df[df['tweet'].str.contains(word)])))
tf1
#The intuition behind inverse document frequency (IDF) is that a word is not of much use to
 us if it's appearing
#in all the documents.
```

Out[260]:

	words	tf	idf
0	doodles	2	5.800882
1	light	1	4.687232
2	funny	1	5.896192
3	significant	1	8.891924
4	ab	1	2.787131
5	exceptions	1	8.891924
6	googledoodle	1	6.183874
7	innovative	1	7.793312
8	learning	1	6.326975
9	occasions	1	8.891924
10	amp	1	2.349452

In [261]:
```
tf1['tfidf'] = tf1['tf'] * tf1['idf']
tf1
#TF-IDF is the multiplication of the TF and IDF which we calculated above.
```

Out[261]:

	words	tf	idf	tfidf
0	doodles	2	5.800882	11.601763
1	light	1	4.687232	4.687232
2	funny	1	5.896192	5.896192
3	significant	1	8.891924	8.891924
4	ab	1	2.787131	2.787131
5	exceptions	1	8.891924	8.891924
6	googledoodle	1	6.183874	6.183874
7	innovative	1	7.793312	7.793312
8	learning	1	6.326975	6.326975
9	occasions	1	8.891924	8.891924
10	amp	1	2.349452	2.349452

In [262]:
```
df_copy=df
```

In [263]:
```
df_copy.head()
```

Out[263]:

	tweet	sentiment	word_count	char_count	avg_word	stopwords	hastags	numerics	upper
0	sxswnui defining language touch different dial...	1	12	89	6.500000	2	3	0	0
1	learning ab doodles doodles light funny amp in...	1	19	143	6.578947	4	2	0	0
2	one inyourface ex stealing show yrs quotat sch...	2	23	132	5.000000	7	1	0	2
3	app would b pretty awesome didnt crash every 1...	0	19	125	5.631579	4	3	0	1
4	line outside austin waiting	1	15	77	4.500000	4	1	0	1

In [264]:
```
# df.drop(["word_count","char_count","avg_word","hastags","numerics","upper","stopwords"],in
place=True,axis=1)
```

In [265]:
```
df.head(5)
```

Out[265]:

	tweet	sentiment	word_count	char_count	avg_word	stopwords	hastags	numerics	upper
0	sxswnui defining language touch different dial...	1	12	89	6.500000	2	3	0	0
1	learning ab doodles doodles light funny amp in...	1	19	143	6.578947	4	2	0	0
2	one inyourface ex stealing show yrs quotat sch...	2	23	132	5.000000	7	1	0	2
3	app would b pretty awesome didnt crash every 1...	0	19	125	5.631579	4	3	0	1
4	line outside austin waiting	1	15	77	4.500000	4	1	0	1

In [266]:
```
df.to_csv('newtwitter.csv')
```

Chapter 8)

MODEL .

Understanding the common words used in the tweets: WordCloud

Now I want to see how well the given sentiments are distributed across the train dataset. One way to accomplish this task is by understanding the common words by plotting wordclouds.

A wordcloud is a visualization wherein the most frequent words appear in large size and the less frequent words appear in smaller sizes.

Let’s visualize all the words our data using the wordcloud plot.

MODEL

In [267]:
```
!pip install WordCloud
```

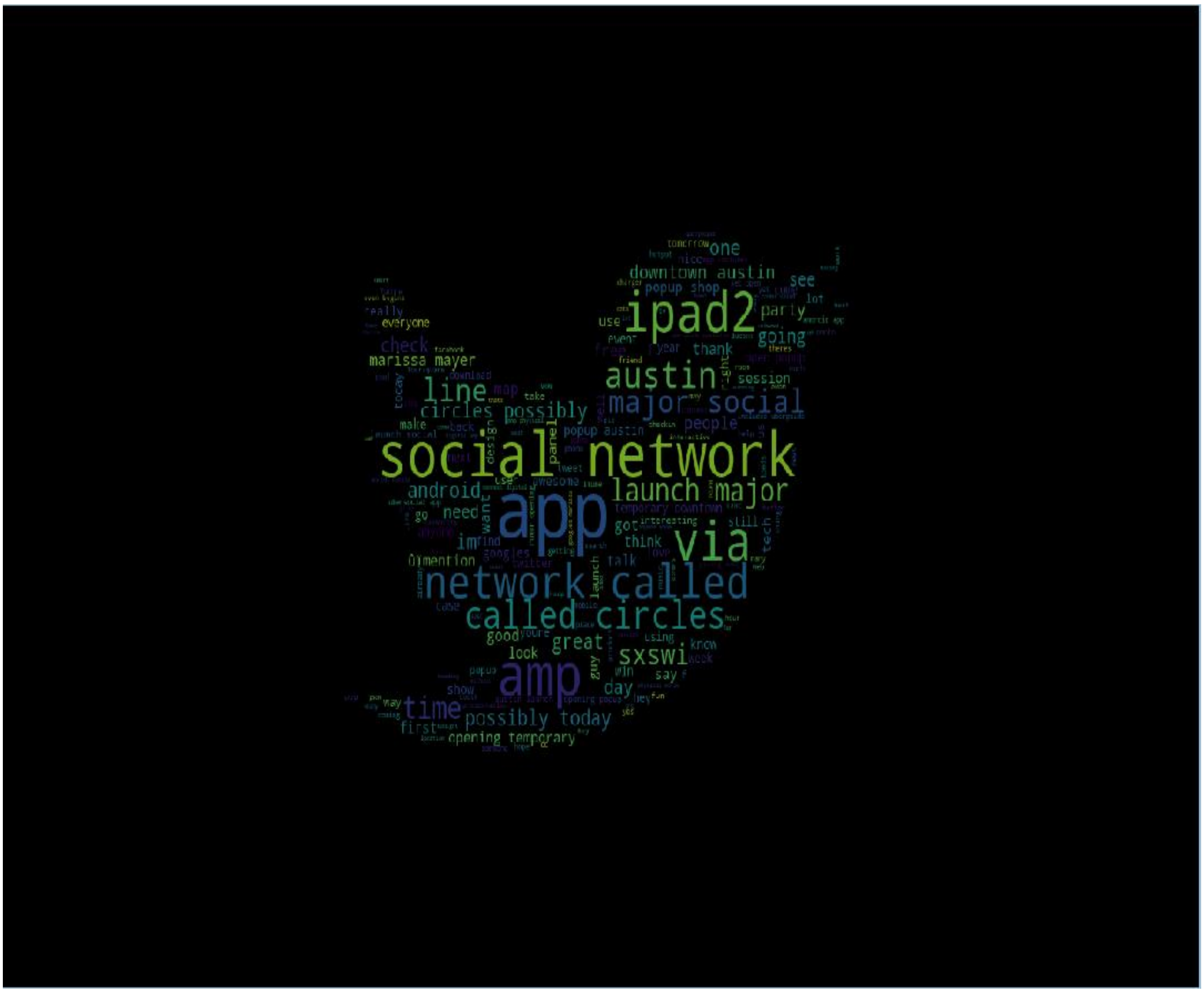
```
Requirement already satisfied: WordCloud in c:\users\shaikh\anaconda3\lib\site-packages (1.5.
0)
Requirement already satisfied: numpy>=1.6.1 in c:\users\shaikh\anaconda3\lib\site-packages (f
rom WordCloud) (1.17.4)
Requirement already satisfied: pillow in c:\users\shaikh\anaconda3\lib\site-packages (from Wo
rdCloud) (6.2.1)
```

In [268]:
```
from wordcloud import WordCloud
from PIL import Image
import requests
```

In [269]:
```
all_words = ' '.join([text for text in df['tweet']])
```

In [270]:
```
mask = np.array(Image.open(requests.get('https://lofrev.net/wp-content/photos/2016/07/twitte
r_logo.jpg', stream=True).raw))
```

In [271]:
```
def generate_wordcloud(all_words, mask):
    word_cloud = WordCloud(width = 900, height = 600, background_color='black', mask=mask).g
enerate(all_words)
    plt.figure(figsize=(20,18),facecolor = 'white', edgecolor='blue')
    plt.figure
    plt.imshow(word_cloud)
    plt.axis('off')
    plt.tight_layout(pad=0)
    plt.show()
generate_wordcloud(all_words,mask)
```


social network
app
ipad2
via
network called
called circles
amp
austin
major
social
line
launch
major
sxswi
time
possibly today
circles possibly
android
downtown austin
popup shop
marissa mayer
check
party
going
people
great
opening temporary

In [209]:
```
df_copy.head()
```

Out[209]:

	tweet	sentiment	word_count	char_count	avg_word	stopwords	hastags	numerics	upper
0	sxswnui defining language touch different dial...	1	12	89	6.500000	2	3	0	0
1	learning ab doodles doodles light funny amp in...	1	19	143	6.578947	4	2	0	0
2	one inyourface ex stealing show yrs quotat sch...	2	23	132	5.000000	7	1	0	2
3	app would b pretty awesome didnt crash every 1...	0	19	125	5.631579	4	3	0	1
4	line outside austin waiting	1	15	77	4.500000	4	1	0	1

In [214]:
```
df_copy['tweet'][:5].apply(lambda x: TextBlob(x).sentiment)
```

Out[214]:
```
0          (0.15, 0.65)
1    (0.38125, 0.89375)
2           (0.0, 0.0)
3          (0.625, 1.0)
4           (0.0, 0.05)
Name: tweet, dtype: object
```

In [215]:
```
df_copy.head()
```

Out[215]:

	tweet	sentiment	word_count	char_count	avg_word	stopwords	hastags	numerics	upper
0	sxswnui defining language touch different dial...	1	12	89	6.500000	2	3	0	0
1	learning ab doodles doodles light funny amp in...	1	19	143	6.578947	4	2	0	0
2	one inyourface ex stealing show yrs quotat sch...	2	23	132	5.000000	7	1	0	2
3	app would b pretty awesome didnt crash every 1...	0	19	125	5.631579	4	3	0	1
4	line outside austin waiting	1	15	77	4.500000	4	1	0	1

In [216]:
```
df.head()
```

Out[216]:

	tweet	sentiment	word_count	char_count	avg_word	stopwords	hastags	numerics	upper
0	sxswnui defining language touch different dial...	1	12	89	6.500000	2	3	0	0
1	learning ab doodles doodles light funny amp in...	1	19	143	6.578947	4	2	0	0
2	one inyourface ex stealing show yrs quotat sch...	2	23	132	5.000000	7	1	0	2
3	app would b pretty awesome didnt crash every 1...	0	19	125	5.631579	4	3	0	1
4	line outside austin waiting	1	15	77	4.500000	4	1	0	1

In [217]:
```
df_copy['sentiment'] = df['tweet'].apply(lambda x: TextBlob(x).sentiment[0] )
df_copy[['tweet','sentiment']].head()
```

Out[217]:

	tweet	sentiment
0	sxswnui defining language touch different dial...	0.15000
1	learning ab doodles doodles light funny amp in...	0.38125
2	one inyourface ex stealing show yrs quotat sch...	0.00000
3	app would b pretty awesome didnt crash every 1...	0.62500
4	line outside austin waiting	0.00000

In [218]:
```
df_copy.sentiment.value_counts()
```

Out[218]:
```
 0.000000    3442
 0.500000     270
 0.400000     232
 0.200000     190
 0.250000     165
             ...
 0.378788       1
 0.471429       1
 0.621429       1
-0.088889       1
-0.111111       1
Name: sentiment, Length: 446, dtype: int64
```

In [143]:
```
df_copy['sentiment']
```

Out[143]:
```
0       0.15000
1       0.38125
2       0.00000
3       0.62500
4       0.00000
         ...
7269    0.05000
7270    0.00000
7271   -0.02500
7272    0.32500
7273    0.00000
Name: sentiment, Length: 7273, dtype: float64
```

Chapter 9)

TESTING.

In [272]:
```
from sklearn.metrics import classification_report
y_true = df['sentiment']
y_pred = df_copy['sentiment']
#0: Negative, 1: Neutral, 2: Positive, 3: Can't Tell
target_names = ['0', '1', '2','3']
# print(classification_report(y_true, y_pred, target_names=target_names)
```

In [273]:
```
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7273 entries, 0 to 7273
Data columns (total 9 columns):
tweet         7273 non-null object
sentiment     7273 non-null int64
word_count    7273 non-null int64
char_count    7273 non-null int64
avg_word      7273 non-null float64
stopwords     7273 non-null int64
hastags       7273 non-null int64
numerics      7273 non-null int64
upper         7273 non-null int64
dtypes: float64(1), int64(7), object(1)
memory usage: 888.2+ KB
```

In [274]:
```
y_true.head(20)
```

Out[274]:
```
0     1
1     1
2     2
3     0
4     1
5     1
6     1
7     1
8     2
9     2
10    3
11    2
12    2
13    1
14    1
15    1
16    1
17    2
18    1
19    1
Name: sentiment, dtype: int64
```

In [275]:
```
y_pred = y_pred.abs()
```

In [276]:
```
y_pred = y_pred.round(decimals=0)
```

In [277]:
```
y_pred
```

Out[277]:
```
0       1
1       1
2       2
3       0
4       1
       ..
7269    1
7270    1
7271    1
7272    1
7273    1
Name: sentiment, Length: 7273, dtype: int64
```

In [278]:
```
print(classification_report(y_true, y_pred, target_names=target_names))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       456
           1       1.00      1.00      1.00      4310
           2       1.00      1.00      1.00      2382
           3       1.00      1.00      1.00       125

    accuracy                           1.00      7273
   macro avg       1.00      1.00      1.00      7273
weighted avg       1.00      1.00      1.00      7273
```

Chapter 10)

FINAL SVM MODEL.

What is Support Vector Machine?

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.

SVM works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable. A separator between the categories is found, then the data are transformed in such a way that the separator could be drawn as a hyperplane.

In [279]:
```
df_copy.head()
```

Out[279]:

	tweet	sentiment	word_count	char_count	avg_word	stopwords	hastags	numerics	upper
0	sxswnui defining language touch different dial...	1	12	89	6.500000	2	3	0	0
1	learning ab doodles doodles light funny amp in...	1	19	143	6.578947	4	2	0	0
2	one inyourface ex stealing show yrs quotat sch...	2	23	132	5.000000	7	1	0	2
3	app would b pretty awesome didnt crash every 1...	0	19	125	5.631579	4	3	0	1
4	line outside austin waiting	1	15	77	4.500000	4	1	0	1

In [280]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
# Create feature vectors
vectorizer = TfidfVectorizer(
                             min_df = 5,
                             max_df = 0.8,
                             sublinear_tf = True,
                             use_idf = True
                            )
train_vectors = vectorizer.fit_transform(df_copy['tweet'])
test_vectors = vectorizer.transform(df_copy['tweet'])
```

In [281]:
```
train_vectors
```

Out[281]:
```
<7273x2018 sparse matrix of type '<class 'numpy.float64'>'
        with 46863 stored elements in Compressed Sparse Row format>
```

In [282]:
```
df_copy
```

Out[282]:

	tweet	sentiment	word_count	char_count	avg_word	stopwords	hastags	numerics	upper
0	sxswnui defining language touch different dial...	1	12	89	6.500000	2	3	0	0
1	learning ab doodles doodles light funny amp in...	1	19	143	6.578947	4	2	0	0
2	one inyourface ex stealing show yrs quotat sch...	2	23	132	5.000000	7	1	0	2
3	app would b pretty awesome didnt crash every 1...	0	19	125	5.631579	4	3	0	1
4	line outside austin waiting	1	15	77	4.500000	4	1	0	1
...	...	...	...	...	...	...	...	...	...
7269	plze tammi im middle craziness everything sooo...	1	16	93	5.200000	4	1	0	1
7270	¼ set _ _ edchat musedchat sxswi newtwitter	1	15	91	5.133333	2	5	0	1
7271	aha found proof lactation room excuse quotmoth...	1	22	140	5.409091	5	1	0	2
7272	launched app get details first edition free	1	18	92	4.166667	6	1	0	2
7273	next fin serv battle vs goog mobile operators ...	1	23	137	5.000000	4	2	0	2

7273 rows × 9 columns

In [283]:
```
import time
from sklearn import svm
from sklearn.metrics import classification_report
# Perform classification with SVM, kernel=linear
classifier_linear = svm.SVC(kernel='linear')
t0 = time.time()
classifier_linear.fit(train_vectors, df_copy['sentiment'])
t1 = time.time()
y_pred = classifier_linear.predict(test_vectors)
t2 = time.time()
time_linear_train = t1-t0
time_linear_predict = t2-t1
```

In [284]:
```
import warnings
warnings.filterwarnings('always')
```

In [285]:
```
# results
print("Training time: %fs; Prediction time: %fs" % (time_linear_train, time_linear_predict))
report = classification_report(df_copy['sentiment'], y_pred, output_dict=True)
```

```
Training time: 4.295119s; Prediction time: 2.671734s
```

```
C:\Users\Shaikh\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1437: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no p
redicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Shaikh\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1437: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no p
redicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\Shaikh\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1437: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no p
redicted samples.
  'precision', 'predicted', average, warn_for)
```

In [286]:
```
report
```

Out[286]:
```
{'0': {'precision': 0.8863636363636364,
  'recall': 0.2565789473684211,
  'f1-score': 0.3979591836734694,
  'support': 456},
 '1': {'precision': 0.7451379885163919,
  'recall': 0.9334106728538283,
  'f1-score': 0.8287156246781336,
  'support': 4310},
 '2': {'precision': 0.7876004592422503,
  'recall': 0.5759865659109992,
  'f1-score': 0.6653734238603297,
  'support': 2382},
 '3': {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 125},
 'accuracy': 0.7578715798157569,
 'macro avg': {'precision': 0.6047755210305696,
  'recall': 0.44149404653331215,
  'f1-score': 0.4730120580529832,
  'support': 7273},
 'weighted avg': {'precision': 0.755092924873162,
  'recall': 0.7578715798157569,
  'f1-score': 0.733968544720633,
  'support': 7273}}
```

In [ ]:
```
#Since Classification Report could not be generated due ti MetricWarning we use sklearn.metr
ics to calculate performance measure of our model
```

In [287]:
```
from sklearn.metrics import f1_score, precision_recall_fscore_support, accuracy_score
precision,recall,_,_ = precision_recall_fscore_support(
                                                        df_copy['sentiment'],
                                                        y_pred,
                                                        average='weighted',
                                                        warn_for=tuple()
                                                    )

print("F1 Score: ",f1_score(df_copy['sentiment'],
                            y_pred, average='weighted',
                            labels=np.unique(y_pred)),
      "\nPrecision:",precision,
      "\nRecall:",recall,
      "\nAccuracy Score:", accuracy_score(df_copy['sentiment'],y_pred))
```

```
F1 Score:  0.7468037529033524
Precision: 0.755092924873162
Recall: 0.7578715798157569
Accuracy Score: 0.7578715798157569
```

In [288]:
```
Ser = pd.Series(y_pred).append(pd.Series([3]))
Ser.value_counts()
```

Out[288]:
```
1    5399
2    1742
0     132
3       1
dtype: int64
```

In [ ]:
```

```

In [ ]:
```
df_copy['sentiment'].value_counts()
```

In [ ]:

In [ ]:
```
df_copy['sentiment'].value_counts()
```

In [ ]:

In [ ]:
```
df_copy
```

In [296]:
```
type(y_pred[0])
```

Out[296]:
```
numpy.int64
```

In [294]:
```
# Confusion Matrix
from sklearn.metrics import confusion_matrix

confusion_matrix(df_copy['sentiment'], y_pred, labels=[0, 1, 2,3])
```

Out[294]:
```
array([[ 117,  268,   71,    0],
       [   9, 4023,  278,    0],
       [   4, 1006, 1372,    0],
       [   2,  102,   21,    0]], dtype=int64)
```

Chapter 11)

Conclusion :

Thus we have successfully implemented Twitter Sentimental Analysis Using Support Vector Machine and have achieved an accuracy of 75.78% . Importing Packages, Data Analyzing, Data Wrangling, Exploratory Data Analysis , Data Preprocessing And Cleaning, Advanced Text Processing, Model , Testing and Final SVM Model are the procedures followed in this project.

References

1) https://pandas.pydata.org/docs/

2) https://numpy.org/doc/1.18/

3) https://scikit-learn.org/stable/modules/svm.html

4) https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

5) https://scikit-learn.org/stable/modules/model_evaluation.html