Social Media Sentiment Analysis

Final Report for Capstone-2 Project

Ojasvi Kapoor

Admission No.: 17SCSE121002

Under the Supervision of

Mr. S.Kalidass

School of Computing Science and Engineering Greater Noida, Uttar Pradesh Winter 2019-2020

Table of Contents:-

Abstract

Sentiment Analysis is the process of 'computationally' determining whether a piece of writing is positive, negative or neutral. It's also known as **opinion mining**, deriving the opinion or attitude of a speaker.

Use of Sentiment Analysis

- **Business: In marketing field companies use it to develop their strategies, to understand customers' feelings towards products or brand, how people respond to their campaigns or product launches and why consumers don't buy some products.**
- **Politics: In political field, it is used to keep track of political view, to detect consistency and inconsistency between statements and actions at the government level. It can be used to predict election results as well!**

- **Public Actions: Sentiment analysis also is used to monitor and analyse social phenomena, for the spotting of potentially dangerous situations and determining the general mood of the blogosphere.**

Introduction

(I) Overall Description

Social media platforms like Twitter, Facebook, YouTube, Reddit generate huge amounts of big data that can be mined in various ways to understand trends, public sentiments and opinions. Social media data today has become relevant for branding, marketing, and business as a whole. A sentiment analyser learns about various sentiments behind a "content piece" (could be IM, email, tweet or any other social media post) through machine learning and predicts the same using AI.Twitter data is considered as a definitive entry point for beginners to practice sentiment analysis machine learning problems. Using Twitter dataset, one can get captivating blend of tweet contents and other related metadata such as hashtags, retweets, location, users and more which pave way for insightful analysis. Twitter dataset consists of 31,962 tweets and is 3MB in size. Using Twitter data you can find out what the world is saying about a topic whether it is movies, sentiments about US elections or any other trending topic like predicting who would win the FIFA world cup 2018.Working with the twitter dataset will help you understand the challenges associated with social media data mining and also learn about classifiers in depth. The foremost problem that you can start working on as a beginner is to build a model to classify tweets as positive or negative.

(II) Purpose

The purpose of Social Media Sentiment Analysis can be understood with the help of the following points:

- It Provides a gathered information about the audience's choice.
- It. categorize the feeds into positive , negative or neutral search.
- It can be used in various fields such as business,politics etc.
- It is also be helpful to overview the opinions(general opinions),
- It helps companies to Know what users really want as a product.

(III) Motivations and Scope

Sentiment Analysis is the process of ‘computationally’ determining whether a piece of writing is positive, negative or neutral. It’s also known as **opinion mining**, deriving the opinion or attitude of a speaker.

It can provide the knowledge Of categorized data at the time of data mining..It can be helpful for politics/business etc.We can easily determine the opinions of the public opinion etc.

Literature Survey

- **The role of Social Media Sentiment Analysis::**

Social media platforms like Twitter, Facebook, YouTube, Reddit generate huge amounts of big data that can be mined in various ways to understand trends, public sentiments and opinions. Social media data today has become relevant for branding, marketing, and business as a whole. A sentiment analyser learns about various sentiments behind a “content piece” (could be IM, email, tweet or any other social media post) through machine learning and predicts the same

Problem Statement

The **problem** in **sentiment analysis** is classifying the polarity of a given **text** at the document,sentence, or feature/aspect level . whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive,negative, or neutral .

Proposed Methodology

The proposed method uses Naïve Bayes and Levenshtein algorithm to determine the emotion into different categories from given social media news data. This method provides excellent performance for real time news data on social media and also provides the better result in terms of accuracy.

Source code:

jupyter Sentiment analysis (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

In [12]:
```
#Data Analysis
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#Data Preprocessing and Feature Engineering
from textblob import TextBlob
import re
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

#Model Selection and Validation
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.metrics import confusion_matrix, classification_report,accuracy_score
```

In [13]:
```
train_tweets=pd.read_csv('d:ojaswi/test_tweets.txt')
test_tweets=pd.read_csv('d:ojaswi/train_tweets.txt')
```

In [14]:
```
sns.barplot(x='id',data = train_tweets)
```

Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2b19ff98a48>
```

jupyter Sentiment analysis (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

In [15]:
```
def form_sentence(tweet):
    tweet_blob = TextBlob(tweet)
    return ' '.join(tweet_blob.words)

print(form_sentence(train_tweets['tweet'].iloc[10]))
print(train_tweets['tweet'].iloc[10])
```

```
1000dayswasted narcosis infinite ep make me aware grinding neuro bass lifestyle
1000dayswasted - narcosis infinite ep.. make me aware.. grinding neuro bass #lifestyle
```

In [16]:
```
def no_user_alpha(tweet):
    tweet_list = [ele for ele in tweet.split() if ele != 'user']
    clean_tokens = [t for t in tweet_list if re.match(r'[^\W\d]*$', t)]
    clean_s = ' '.join(clean_tokens)
    clean_mess = [word for word in clean_s.split() if word.lower() not in stopwords.words('english')]
    return clean_mess
print(no_user_alpha(form_sentence(train_tweets['tweet'].iloc[10])))
print(train_tweets['tweet'].iloc[10])
```

```
['narcosis', 'infinite', 'ep', 'make', 'aware', 'grinding', 'neuro', 'bass', 'lifestyle']
1000dayswasted - narcosis infinite ep.. make me aware.. grinding neuro bass #lifestyle
```

In [17]:
```
def normalization(tweet_list):
        lem = WordNetLemmatizer()
        normalized_tweet = []
        for word in tweet_list:
            normalized_text = lem.lemmatize(word,'v')
            normalized_tweet.append(normalized_text)
        return normalized_tweet

tweet_list = 'I was playing with my friends with whom I used to play, when you called me yesterday'.split()
print(normalization(tweet_list))
```

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

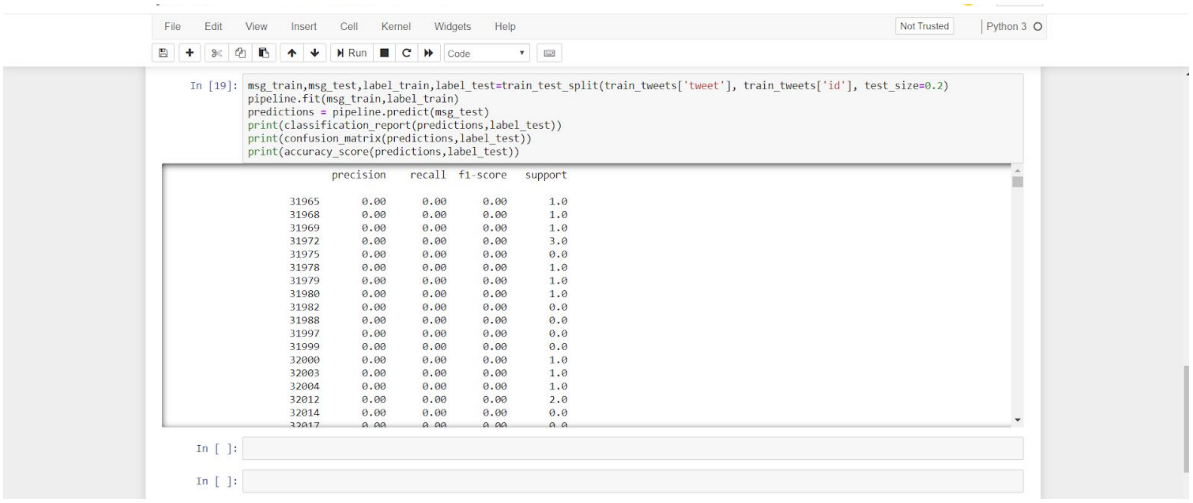
Run Code

```
['I', 'be', 'play', 'with', 'my', 'friends', 'with', 'whom', 'I', 'use', 'to', 'play,', 'when', 'you', 'call', 'me', 'yesterday']
```

In [18]:
```
pipeline = Pipeline([
    ('bow',CountVectorizer(analyzer = 'char_wb')),  # strings to token integer counts
    ('tfidf', TfidfTransformer()),  # integer counts to weighted TF-IDF scores
    ('classifier', MultinomialNB()),  # train on TF-IDF vectors w/ Naive Bayes classifier
])
```

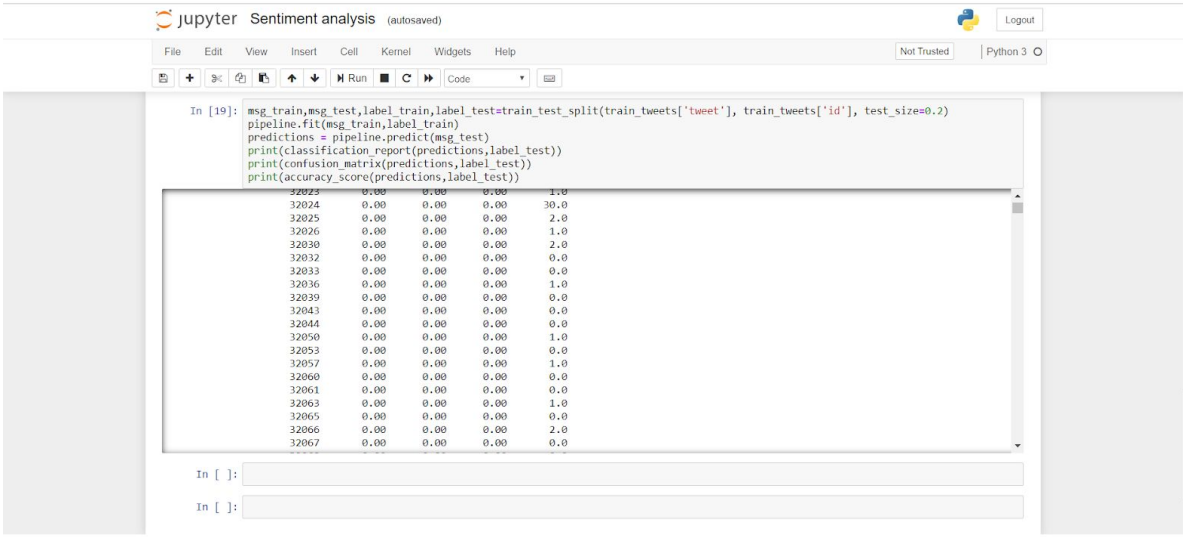
In [19]:
```
msg_train,msg_test,label_train,label_test=train_test_split(train_tweets['tweet'], train_tweets['id'], test_size=0.2)
pipeline.fit(msg_train,label_train)
predictions = pipeline.predict(msg_test)
print(classification_report(predictions,label_test))
print(confusion_matrix(predictions,label_test))
print(accuracy_score(predictions,label_test))
```

```
              precision    recall  f1-score   support

       31965       0.00      0.00      0.00       1.0
       31968       0.00      0.00      0.00       1.0
       31969       0.00      0.00      0.00       1.0
       31972       0.00      0.00      0.00       3.0
       31975       0.00      0.00      0.00       0.0
       31978       0.00      0.00      0.00       1.0
       31979       0.00      0.00      0.00       1.0
       31980       0.00      0.00      0.00       1.0
       31982       0.00      0.00      0.00       0.0
       31988       0.00      0.00      0.00       0.0
       31997       0.00      0.00      0.00       0.0
       31999       0.00      0.00      0.00       0.0
```



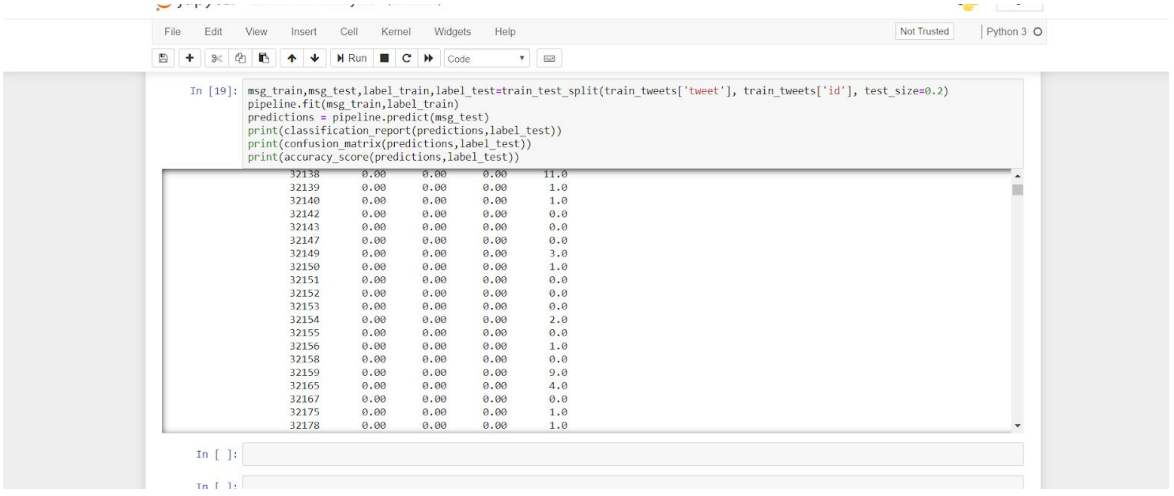
```
In [19]: msg_train,msg_test,label_train,label_test=train_test_split(train_tweets['tweet'], train_tweets['id'], test_size=0.2)
pipeline.fit(msg_train,label_train)
predictions = pipeline.predict(msg_test)
print(classification_report(predictions,label_test))
print(confusion_matrix(predictions,label_test))
print(accuracy_score(predictions,label_test))
```

```
              precision    recall  f1-score   support

       31965       0.00      0.00      0.00       1.0
       31968       0.00      0.00      0.00       1.0
       31969       0.00      0.00      0.00       1.0
       31972       0.00      0.00      0.00       3.0
       31975       0.00      0.00      0.00       0.0
       31978       0.00      0.00      0.00       1.0
       31979       0.00      0.00      0.00       1.0
       31980       0.00      0.00      0.00       1.0
       31982       0.00      0.00      0.00       0.0
       31988       0.00      0.00      0.00       0.0
       31997       0.00      0.00      0.00       0.0
       31999       0.00      0.00      0.00       0.0
       32000       0.00      0.00      0.00       1.0
       32003       0.00      0.00      0.00       1.0
       32004       0.00      0.00      0.00       1.0
       32012       0.00      0.00      0.00       2.0
       32014       0.00      0.00      0.00       0.0
```



```
In [19]: msg_train,msg_test,label_train,label_test=train_test_split(train_tweets['tweet'], train_tweets['id'], test_size=0.2)
pipeline.fit(msg_train,label_train)
predictions = pipeline.predict(msg_test)
print(classification_report(predictions,label_test))
print(confusion_matrix(predictions,label_test))
print(accuracy_score(predictions,label_test))
```

```
       32024       0.00      0.00      0.00      30.0
       32025       0.00      0.00      0.00       2.0
       32026       0.00      0.00      0.00       1.0
       32030       0.00      0.00      0.00       2.0
       32032       0.00      0.00      0.00       0.0
       32033       0.00      0.00      0.00       0.0
       32036       0.00      0.00      0.00       1.0
       32039       0.00      0.00      0.00       0.0
       32043       0.00      0.00      0.00       0.0
       32044       0.00      0.00      0.00       0.0
       32050       0.00      0.00      0.00       1.0
       32053       0.00      0.00      0.00       0.0
       32057       0.00      0.00      0.00       1.0
       32060       0.00      0.00      0.00       0.0
       32061       0.00      0.00      0.00       0.0
       32063       0.00      0.00      0.00       1.0
       32065       0.00      0.00      0.00       0.0
       32066       0.00      0.00      0.00       2.0
       32067       0.00      0.00      0.00       0.0
```



```
In [19]: msg_train,msg_test,label_train,label_test=train_test_split(train_tweets['tweet'], train_tweets['id'], test_size=0.2)
pipeline.fit(msg_train,label_train)
predictions = pipeline.predict(msg_test)
print(classification_report(predictions,label_test))
print(confusion_matrix(predictions,label_test))
print(accuracy_score(predictions,label_test))
```

```
       32138       0.00      0.00      0.00      11.0
       32139       0.00      0.00      0.00       1.0
       32140       0.00      0.00      0.00       1.0
       32142       0.00      0.00      0.00       0.0
       32143       0.00      0.00      0.00       0.0
       32147       0.00      0.00      0.00       0.0
       32149       0.00      0.00      0.00       3.0
       32150       0.00      0.00      0.00       1.0
       32151       0.00      0.00      0.00       0.0
       32152       0.00      0.00      0.00       0.0
       32153       0.00      0.00      0.00       0.0
       32154       0.00      0.00      0.00       2.0
       32155       0.00      0.00      0.00       0.0
       32156       0.00      0.00      0.00       1.0
       32158       0.00      0.00      0.00       0.0
       32159       0.00      0.00      0.00       9.0
       32165       0.00      0.00      0.00       4.0
       32167       0.00      0.00      0.00       0.0
       32175       0.00      0.00      0.00       1.0
       32178       0.00      0.00      0.00       1.0
```

Output

jupyter Sentiment analysis (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

In [19]:
```
msg_train,msg_test,label_train,label_test=train_test_split(train_tweets['tweet'], train_tweets['id'], test_size=0.2)
pipeline.fit(msg_train,label_train)
predictions = pipeline.predict(msg_test)
print(classification_report(predictions,label_test))
print(confusion_matrix(predictions,label_test))
print(accuracy_score(predictions,label_test))
```

```
32023       0.00      0.00      0.00       1.0
32024       0.00      0.00      0.00      30.0
32025       0.00      0.00      0.00       2.0
32026       0.00      0.00      0.00       1.0
32030       0.00      0.00      0.00       2.0
32032       0.00      0.00      0.00       0.0
32033       0.00      0.00      0.00       0.0
32036       0.00      0.00      0.00       1.0
32039       0.00      0.00      0.00       0.0
32043       0.00      0.00      0.00       0.0
32044       0.00      0.00      0.00       0.0
32050       0.00      0.00      0.00       1.0
32053       0.00      0.00      0.00       0.0
32057       0.00      0.00      0.00       1.0
32060       0.00      0.00      0.00       0.0
32061       0.00      0.00      0.00       0.0
32063       0.00      0.00      0.00       1.0
32065       0.00      0.00      0.00       0.0
32066       0.00      0.00      0.00       2.0
32067       0.00      0.00      0.00       0.0
```

In [ ]:

In [ ]:

jupyter Sentiment analysis (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

In [19]:
```
msg_train,msg_test,label_train,label_test=train_test_split(train_tweets['tweet'], train_tweets['id'], test_size=0.2)
pipeline.fit(msg_train,label_train)
predictions = pipeline.predict(msg_test)
print(classification_report(predictions,label_test))
print(confusion_matrix(predictions,label_test))
print(accuracy_score(predictions,label_test))
```

```
32138       0.00      0.00      0.00      11.0
32139       0.00      0.00      0.00       1.0
32140       0.00      0.00      0.00       1.0
32142       0.00      0.00      0.00       0.0
32143       0.00      0.00      0.00       0.0
32147       0.00      0.00      0.00       0.0
32149       0.00      0.00      0.00       3.0
32150       0.00      0.00      0.00       1.0
32151       0.00      0.00      0.00       0.0
32152       0.00      0.00      0.00       0.0
32153       0.00      0.00      0.00       0.0
32154       0.00      0.00      0.00       2.0
32155       0.00      0.00      0.00       0.0
32156       0.00      0.00      0.00       1.0
32158       0.00      0.00      0.00       0.0
32159       0.00      0.00      0.00       9.0
32165       0.00      0.00      0.00       4.0
32167       0.00      0.00      0.00       0.0
32175       0.00      0.00      0.00       1.0
32178       0.00      0.00      0.00       1.0
```

In [ ]:

In [ ]:

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

In [19]:
```
msg_train,msg_test,label_train,label_test=train_test_split(train_tweets['tweet'], train_tweets['id'], test_size=0.2)
pipeline.fit(msg_train,label_train)
predictions = pipeline.predict(msg_test)
print(classification_report(predictions,label_test))
print(confusion_matrix(predictions,label_test))
print(accuracy_score(predictions,label_test))
```

```
    accuracy                           0.00    3440.0
   macro avg       0.00      0.00      0.00    3440.0
weighted avg       0.00      0.00      0.00    3440.0

[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
0.0
```

```
C:\Users\HALLOFFAME\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\metrics\classification.py:1437: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HALLOFFAME\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\metrics\classification.py:1439: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
```

In [ ]:

Social Media Sentiment Analysis Specifications:

1. HARDWARE INTERFACE

- **Processor:** Intel dual core or above
- **Processor Speed:**2.2GHZ or above
- **RAM:** 4 GB RAM or above
- **Hard Disk:** 20 GB hard disk or above
- **OS:** Windows, MacOS, Linux (with specified web browser)

2. SOFTWARE INTERFACE

- **Operating System:**

Windows 10

- **Technology:**

Python/Machine Learning/NLP.

Conclusion:-

It is a very important fact to analyze how people think in different contexts about different things.

References:-

1. *https://link.springer.com/chapter/10.1007/978-3-319-11310-4_76*
2. *http://colah.github.io/posts/2015-08-Understanding-LSTMs/*
3. *https://link.springer.com/chapter/10.1007/978-3-319-11310-4_76*
4. *https://machinelearningmastery.com/diagnose-overfitting-underfitting-lstm-models/*
5. *https://vijaikumar.in/how-to-make-publication-quality-plots-using-python-86bb1c6dabf7memory-networks-keras/*

6. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2992018/
7. https://towardsdatascience.com/twitter-sentiment-analysis-classification-using-nltk-python-fa912578614c