A REPORT

ON

“SENTIMENT ANALYSIS”

By

Sarath Chandra Vadavalli AP21110011318
-------------------------------- --------------------

Prepared in the partial fulfillment of the
Summer Research Internship

UNDER

Dr.Sanjay Kumar sir

SRM UNIVERSITY, AP

(July, 2023)

Acknowledgements

We would like to express our special thanks to SANJAY KUMAR sir for giving us guidance and support in order to complete the research internship successfully.

All our team members discussed together, shared our ideas and completed this research. The interaction among our team members has made it wonderful and impressive.

We have learnt many new concepts in this course of research. We thank SRM University for giving us this wonderful opportunity to do a research internship.

V SARATH CHANDRA
B.Tech, 3nd Year, CSE

Abstract

Sentiment analysis is a crucial natural language processing task that aims to determine the emotion or feeling expressed in text data. With the widespread use of social media platforms, sentiment analysis has become increasingly important for businesses, researchers, and individuals seeking to gauge public opinion and sentiment towards products, services, events, and various social issues. This research presents a comprehensive comparative study of sentiment analysis techniques applied to social media texts.

The primary objective of this study is to analyze and compare the effectiveness of various sentiment analysis methods, including traditional machine learning approaches. A diverse dataset of real-world social media texts is collected and preprocessed to ensure the representation of different sentiments and linguistic variations.

The research evaluates the performance of classical It also uses different natural language processing (NLP) techniques like tokenization, lemmatization, stemming etc. The library which is used to pre-process the textual data is Natural language toolkit (nltk). The experimental results show whether a particular text is of positive, negative or neutral sentiment as well as predicts the label of text which indicates whether is depressed tweet or not.

Contents

Introduction

Social media has become a vital platform for different individuals, organizations, and governments worldwide to communicate with each other and express their views. They also offer a unique opportunity to gain insights into individuals' mental health and emotional well-being. One of such social media platform is '**twitter**'. It serves as a vast repository of user-generated content that can be used for sentiment analysis and understanding mental health trends.

Depression is a serious mental health issue that affects millions of people worldwide, with its prevalence continuing to rise in the digital age. The study of depression through Twitter analysis has gained significant attention from researchers, mental health professionals, and policymakers. By analyzing the language used in tweets, sentiment analysis can provide valuable information about the emotional state of users and detect signs of depression.

This research aims to explore and analyze tweets related to depression to better comprehend the emotional expressions and sentiments prevalent in such discussions. It tries to identify link between social media usage and mental health, contributing to the broader discourse on the implications of digital connectivity on well-being. Through sentiment analysis, we try to identify patterns and trends associated with depressive state on Twitter. The insights gained from this analysis could help mental health experts, support organizations and policymakers devise targeted interventions and support systems for those experiencing depression.

In this study, we will employ various sentiment analysis techniques with machine learning algorithms to assess their effectiveness in detecting and categorizing depression-related sentiments in tweets. We will collect a dataset of tweets, annotated with depression-related labels, to train and evaluate the models. The insights gained from this research can help in building automated systems that can assist in identifying potential cases of depression early on, thereby improving mental health outreach and support.

Description

Dataset:

A dataset is a collection of data organized in a specific way for analysis or processing. It's like a bunch of information that we can study. For example, in sentiment analysis, we need data to understand people's feelings and opinions.

Different models use datasets for different purposes. One example is NLP (Natural Language Processing) model. These models use large datasets of text documents to learn about language patterns and meanings. They can do things like analyzing sentiment (positive or negative feelings), translating languages, or identifying named entities (like names of people or places).

- When it comes to Twitter, there are specific datasets that people use. Here are some examples:

  1. **Sentiment140:** This dataset contains 1.6 million tweets that are labeled as positive or negative. It's widely used for sentiment analysis on Twitter.
  2. **Tweepy:** A Python library helps us access the Twitter API. With Tweepy, we can collect our own Twitter datasets by filtering and retrieving specific data.
  3. **Kaggle:** Kaggle is a platform where people share datasets and compete in machine learning challenges. You can find Twitter sentiment analysis datasets on Kaggle that are contributed by the community or part of specific challenges.

These datasets are helpful because they provide us with a lot of Twitter data to work with and train our models. Overall, datasets are important because they provide us with the information we need to analyze and understand different aspects of Twitter, like sentiment or other patterns.

Preprocessing:

It is a crucial step in sentiment analysis, playing a fundamental role in improving the accuracy and reliability of sentiment classification models. Some techniques are:

1. **Text Cleaning:** To eliminate irrelevant characters and symbols, the text data underwent thorough cleaning.
2. **Tokenization:** The process of tokenization involved splitting the text into individual words or tokens.
3. **Stop Word Removal:** Common words that lack significant sentiment information, such as "a," "the," and "is," were removed from the dataset.
4. **Normalization:** Normalization techniques were applied to ensure consistency and reduce the dimensionality of the data.
5. **Handling Contractions:** Contractions were expanded to their full forms to ensure consistency in word representation. For example, "can't" was expanded to "cannot" and "I've" was expanded to "I have."
6. **Removing HTML Tags:** As the dataset contained customer reviews from an e-commerce platform, some text entries contained HTML tags.
7. **Removing Irrelevant Information:** The dataset also included mentions, hashtags, and usernames from social media platforms.
8. **Spell Checking and Correction:** To address any spelling errors or inconsistencies, we employed the PySpellChecker library.

By employing these preprocessing techniques, we prepared the text data for further feature extraction and sentiment classification tasks, ensuring the accuracy and reliability of subsequent analyses.

Different Approaches:

Here are some basic approaches to performing sentiment analysis:

1. **Rule-based Approach:** This approach uses predefined rules and patterns to determine sentiment. Rules can be created manually or using sentiment lexicons or dictionaries. Words or phrases are matched against these rules to determine their sentiment.

2. **Machine Learning Approach:** Machine learning techniques are used to train models for sentiment analysis. The models learn from labeled data, where each data point has a sentiment label (positive, negative, neutral). The trained model can then predict the sentiment of new, unseen data.

3. **Lexicon-based Approach:** Lexicon-based approaches rely on sentiment lexicons, which are lists of words or phrases with assigned sentiment polarities. The sentiment of a text is calculated by looking up the words in the lexicon and aggregating their sentiment values.

4. **Hybrid Approaches:** Hybrid approaches combine multiple techniques to improve sentiment analysis. For example, a hybrid approach may use both rule-based and machine-learning methods to achieve better accuracy.

These approaches are used to analyze the sentiment expressed in text data, such as social media posts, customer reviews, or news articles. By understanding the sentiment, we can gain insights into people's opinions and attitudes towards different subjects.

NLP:

The preprocessing of text is known as natural language processing (NLP). It includes six steps: tokenization, lemmatization, stemming, part-of-speech tagging, named entity recognition, and chunking.

Tokenization- It is like breaking a text into smaller pieces called tokens. Tokens can be words, sentences, paragraphs, or even individual characters. It's an important step in NLP because it helps us understand and analyze the text better.

- There are mainly four types of tokenization. They are:
  1. **Word Tokenization:** This splits the text into separate words. For example, the sentence "I love ice cream" would be divided into ["I", "love", "ice", and "cream"].
  2. **Sentence Tokenization:** This separates the text into individual sentences. For example, the paragraph "I love ice cream. It's so delicious. would become ["I love ice cream.", "It's so delicious."].
  3. **Character Tokenization:** This breaks the text down into individual letters or characters. For example, the word "tokenization" would be split into ["t", "o", "k", "e", "n", "i", "z", "a", "t", "i", "o", "n"].
  4. **Subword Tokenization:** This method divides the text into meaningful subword units.

Tokenization may also involve additional steps like removing punctuation, making every character of the text into lowercase, handling contractions (like "don't" to "do not"), removing common words (stopwords), and more. To tokenize text, we can use special tools or libraries designed for NLP, such as NLTK or the tokenizer module in the Hugging Face's Transformers library.

Lemmatization- It is a process used in NLP to make words simpler. It helps by converting different forms of a word into its basic form, called the lemma. The lemma is like the "dictionary version" of a word. For example,

Original Sentence: "The cats are chasing mice."

Lemmatized Sentence: "The cat is chase mouse."

Lemmatization always produces valid words because it considers the context of the sentence. We use specific tools or libraries in NLP for this purpose.

Stemming- It helps by chopping off the ends of words to find their core form, called the stem. The stem is like the "base" or "root" version of a word.

For example,

Original Word: "Running"

Stemmed Word: "Run"

Stemming is often used when we want a quick and simple way to find the core forms of words. But sometimes, we need more accurate results, which is when we might use lemmatization instead.

Part-of-speech tagging- It is the process of labeling each word in a sentence with its grammatical part of speech. An individual word can exhibit different parts of speech in different sentences according to its context. NLP takes the context into consideration and assigns the tag.

For example,

First Sentence: The watch is so expensive --> Here, watch is in '**noun**' form.

Second Sentence: They are watching a movie --> Here, watch is in '**verb**' form.

Named entity recognition- It is the process of identifying and categorizing named entities, such as people, places, and organizations in text.

For example,

Sentence: Sundar Pichai is the CEO of Google company --> It identifies the name '**Sundar Pichai**' and the company '**Google**'.

Chunking- Chunking is used to collect the individual piece of information and grouping them into bigger pieces of sentences.

Steps in algorithm:

Explanation of how the algorithm works in sentiment analysis:

1. **Data Preparation:** First, we clean and prepare the text data by removing unnecessary words like "and," "the," and punctuation marks. We want to focus on important key words that express sentiment.

2. **Classification:** When we have a new, unlabeled document, we assign a sentiment label (positive, negative or neutral) for each given text in the document.

3. **Training:** We use a set of labeled data where each document is labeled with a sentiment (positive or negative). We fit this data into our Machine learning model to train it.

4. **Testing:** We evaluate how well the model is predicting the output on unseen data. We compare the predicted sentiment labels to the actual labels and measure metrics like accuracy, precision, and recall.

5. **Creating confusion matrix:** A confusion matrix is a matrix that summarizes the performance of a machine learning model on a set of test data. So, we will check the accuracy of the Naive Bayes classifier using the Confusion matrix.

These are the steps to be followed for any machine learning model. In this project, we use four types of algorithms to predict the label of given text data. They are:

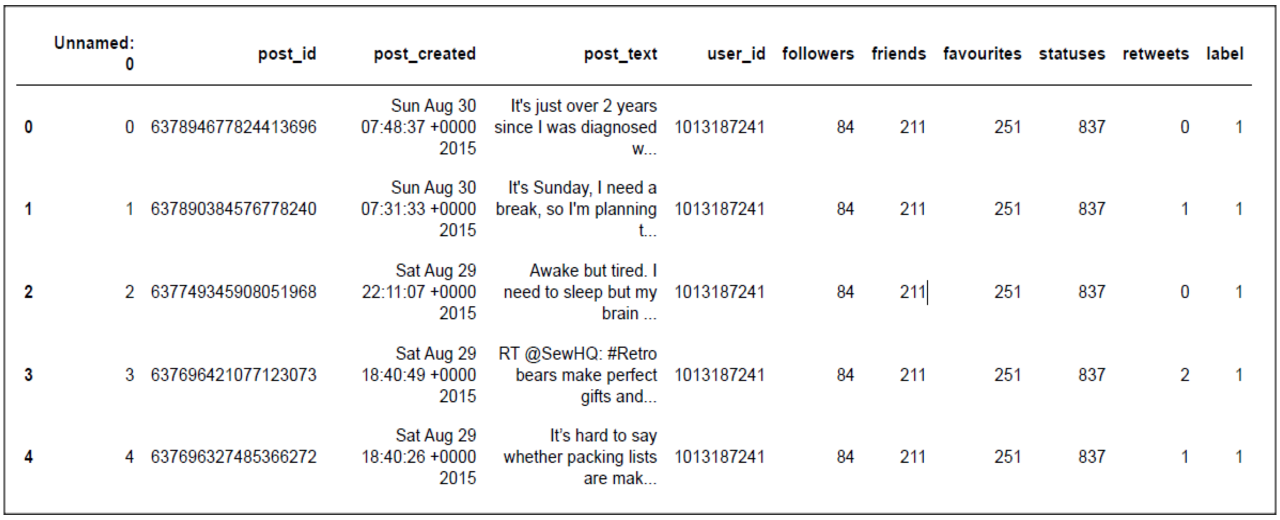
- Logistic Regression
- Naïve Bayes
- Support vector machine (SVM)
- Random Forest

Implementation

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

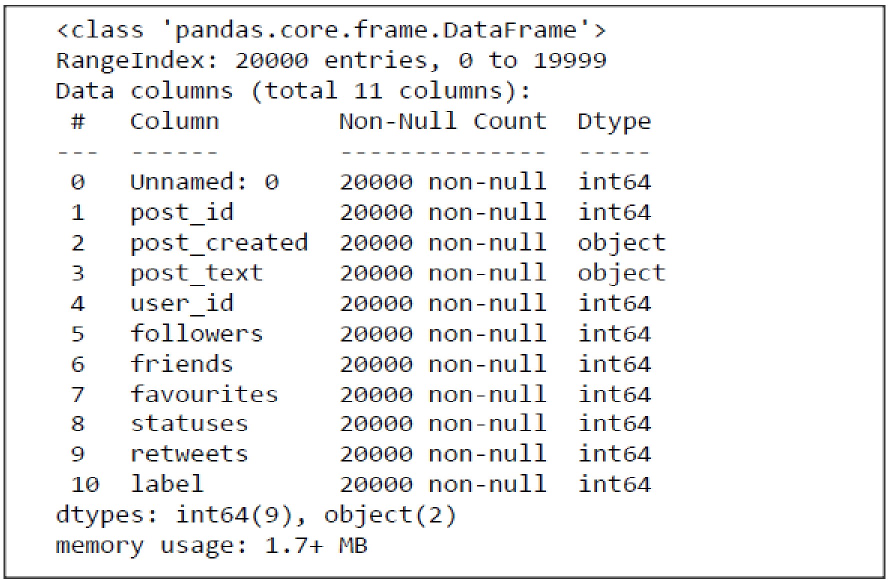
Extracting data:

```
df = pd.read_csv('Twitter.csv')
df.head()
```



	Unnamed: 0	post_id	post_created	post_text	user_id	followers	friends	favourites	statuses	retweets	label
0	0	637894677824413696	Sun Aug 30 07:48:37 +0000 2015	It's just over 2 years since I was diagnosed w...	1013187241	84	211	251	837	0	1
1	1	637890384576778240	Sun Aug 30 07:31:33 +0000 2015	It's Sunday, I need a break, so I'm planning t...	1013187241	84	211	251	837	1	1
2	2	637749345908051968	Sat Aug 29 22:11:07 +0000 2015	Awake but tired. I need to sleep but my brain ...	1013187241	84	211	251	837	0	1
3	3	637696421077123073	Sat Aug 29 18:40:49 +0000 2015	RT @SewHQ: #Retro bears make perfect gifts and...	1013187241	84	211	251	837	2	1
4	4	637696327485366272	Sat Aug 29 18:40:26 +0000 2015	It's hard to say whether packing lists are mak...	1013187241	84	211	251	837	1	1

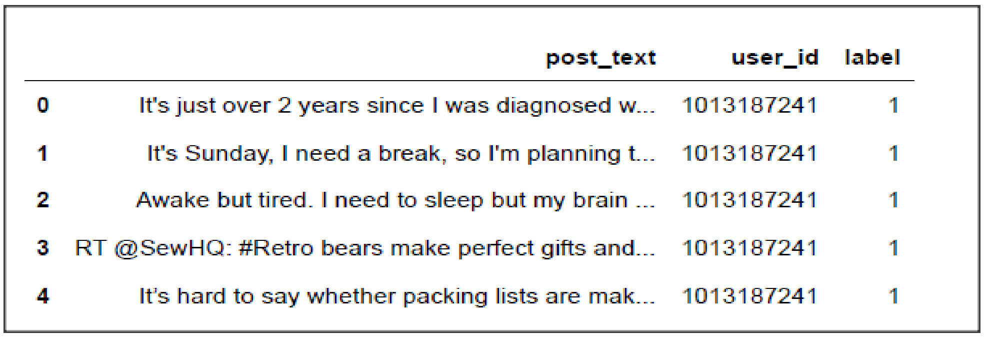
```
df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Unnamed: 0    20000 non-null  int64 
 1   post_id       20000 non-null  int64 
 2   post_created  20000 non-null  object
 3   post_text     20000 non-null  object
 4   user_id       20000 non-null  int64 
 5   followers     20000 non-null  int64 
 6   friends       20000 non-null  int64 
 7   favourites    20000 non-null  int64 
 8   statuses      20000 non-null  int64 
 9   retweets      20000 non-null  int64 
 10  label         20000 non-null  int64 
dtypes: int64(9), object(2)
memory usage: 1.7+ MB
```

Dropping unnecessary columns:

```python
df = df.loc[:, ~df.columns.str.contains('^Unnamed')]
df = df.drop(['post_id','post_created','followers','friends','favourites',
                                         'statuses','retweets'], axis = 1)
```



	post_text	user_id	label
0	It's just over 2 years since I was diagnosed w...	1013187241	1
1	It's Sunday, I need a break, so I'm planning t...	1013187241	1
2	Awake but tired. I need to sleep but my brain ...	1013187241	1
3	RT @SewHQ: #Retro bears make perfect gifts and...	1013187241	1
4	It's hard to say whether packing lists are mak...	1013187241	1

Cleaning the data:

```python
df['post_text'] = df['post_text'].str.lower()
df['post_text'] = df['post_text'].str.replace("\d","")
df['post_text'] = df['post_text'].str.replace("[^\w\s]","")
```

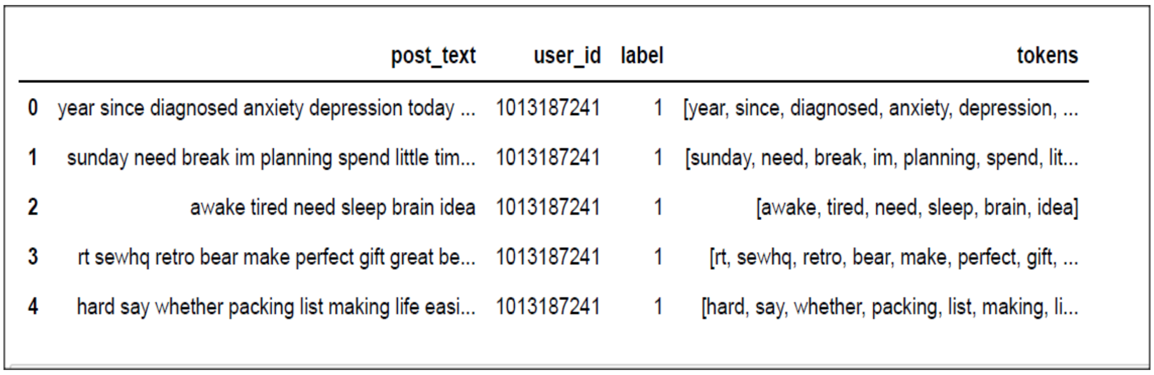
Lemmatization:

```python
from textblob import Word
from nltk.stem import WordNetLemmatizer
def lemmatize_text(text):
    lemmatizer = WordNetLemmatizer()
    return ' '.join([lemmatizer.lemmatize(w) for w in text.split(' ')])

df["post_text"] = df["post_text"].apply(lemmatize_text)
```

Tokenization:

```python
from textblob import TextBlob
df["tokens"] = df["post_text"].apply(lambda x: TextBlob(x).words)
df.head()
```



	post_text	user_id	label	tokens
0	year since diagnosed anxiety depression today ...	1013187241	1	[year, since, diagnosed, anxiety, depression, ...
1	sunday need break im planning spend little tim...	1013187241	1	[sunday, need, break, im, planning, spend, lit...
2	awake tired need sleep brain idea	1013187241	1	[awake, tired, need, sleep, brain, idea]
3	rt sewhq retro bear make perfect gift great be...	1013187241	1	[rt, sewhq, retro, bear, make, perfect, gift, ...
4	hard say whether packing list making life easi...	1013187241	1	[hard, say, whether, packing, list, making, li...

Evaluating polarity and subjectivity:

```
L = []
for i in df["post_text"]:
    blob = TextBlob(i).sentiment
    L.append(blob)

df2 = pd.DataFrame(L)
df2.head()
```

	polarity	subjectivity
0	0.100000	1.000000
1	-0.093750	0.750000
2	-0.400000	0.700000
3	0.900000	0.875000
4	-0.045833	0.370833

Predicting the sentiment:

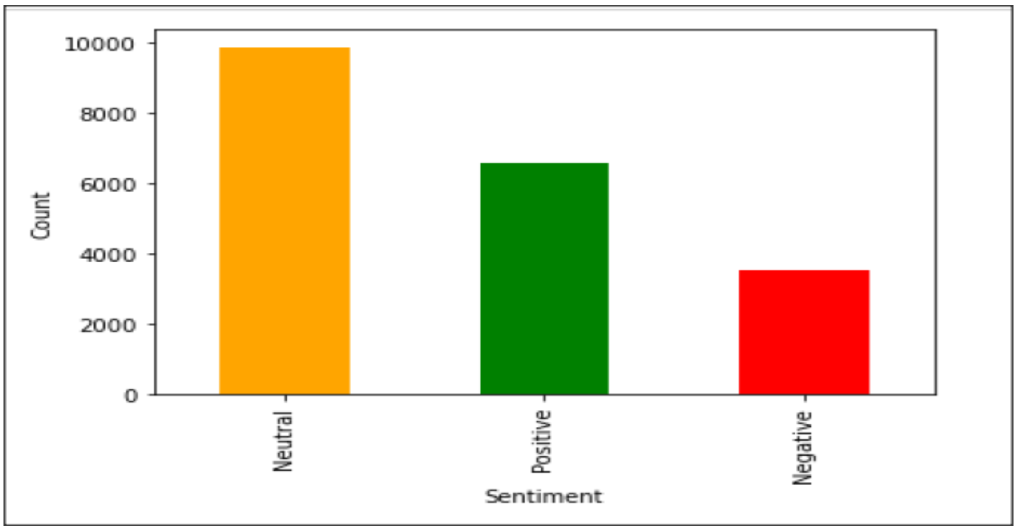
```
dff = pd.concat([df, df2], axis=1)
def get_sentiment(text):
    blob = TextBlob(text)
    polarity = blob.sentiment.polarity
    if polarity < 0:
        sentiment = 'Negative'
    elif polarity > 0:
        sentiment = 'Positive'
    else:
        sentiment = 'Neutral'
    return sentiment

dff["sentiment"] = dff["post_text"].apply(get_sentiment)
dff.head()
```

	post_text	user_id	label	tokens	polarity	subjectivity	sentiment
0	year since diagnosed anxiety depression today ...	1013187241	1	[year, since, diagnosed, anxiety, depression, ...	0.100000	1.000000	Positive
1	sunday need break im planning spend little tim...	1013187241	1	[sunday, need, break, im, planning, spend, lit...	-0.093750	0.750000	Negative
2	awake tired need sleep brain idea	1013187241	1	[awake, tired, need, sleep, brain, idea]	-0.400000	0.700000	Negative
3	rt sewhq retro bear make perfect gift great be...	1013187241	1	[rt, sewhq, retro, bear, make, perfect, gift, ...	0.900000	0.875000	Positive
4	hard say whether packing list making life easi...	1013187241	1	[hard, say, whether, packing, list, making, li...	-0.045833	0.370833	Negative

Analysis of each sentiment:

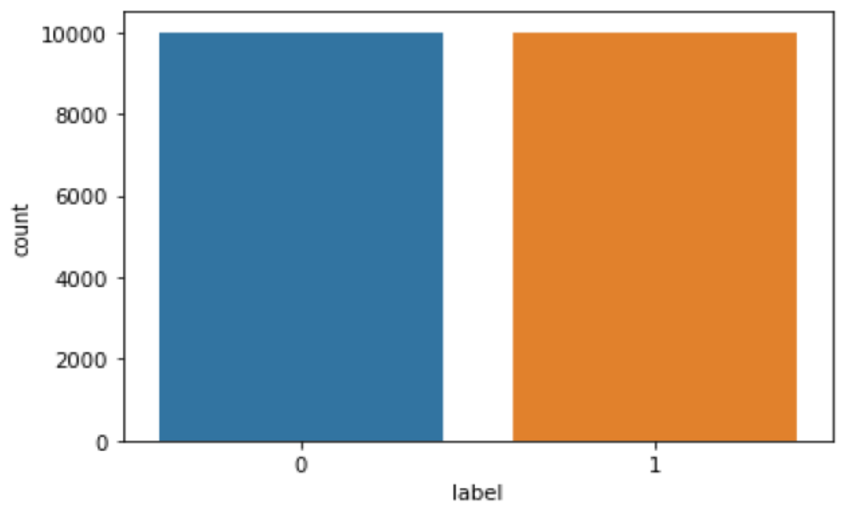
```
Result = dff["sentiment"].value_counts()
plt.xlabel('Sentiment')
plt.ylabel('Count')
result.plot(kind="bar", color = ['orange','green','red']);
```



Analysis of label in dataset:

```
plt.xlabel('Label')
plt.ylabel('Count')
sns.countplot(x ='label', data = df)
```

```
<AxesSubplot:xlabel='label', ylabel='count'>
```



Splitting the dataset into training and testing data:

```
X = df['post_text']
y = df['label']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2,
                                                        random_state = 0)
```

Count Vectorizer:

```
from sklearn.feature_extraction.text
import CountVectorizer
vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

Logistic Regression model:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import confusion_matrix
lr = LogisticRegression(max_iter = 200)
lr.fit(X_train, y_train)
s1 = lr.score(X_test, y_test)     # s1 = 0.85375
print(s1)                         # Logistic Regression accuracy = 85.375%
y_pred1 = lr.predict(X_test)
cm1 = confusion_matrix(y_test, y_pred1)
print(cm1)
o/p: [[1738 268]
      [317 1677]]
```

Naïve Bayes model:

```
from sklearn.naive_bayes import MultinomialNB
np = MultinomialNB()
np.fit(X_train, y_train)
s2 = np.score(X_test, y_test)     # s2 = 0.8495
print(s2)                         # Naïve Bayes accuracy = 84.95%
y_pred2 = np.predict(X_test)
cm2 = confusion_matrix(y_test, y_pred2)
print(cm2)
```

```
o/p: [[1586 420]
      [182 1812]]
```

Support Vector Machine:

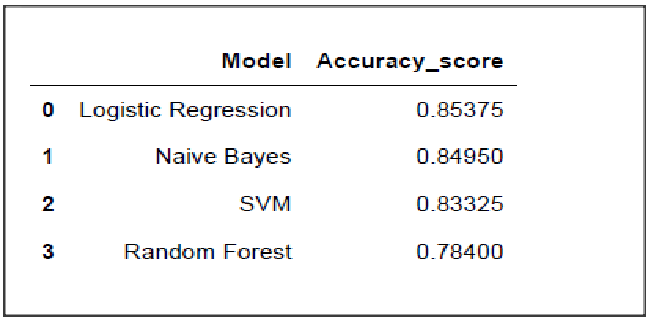
```
from sklearn.svm import SVC
sv = SVC()
sv.fit(X_train, y_train)
s3 = sv.score(X_test, y_test)     # s3 = 0.83325
print(s3)                         # SVM accuracy = 83.325%
y_pred3 = np.predict(X_test)
cm3 = confusion_matrix(y_test, y_pred3)
print(cm3)
o/p: [[1679 327]
      [340 1654]]
```

Random Forest model:

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier();
rf.fit(X_train, y_train)
s4 = sv.score(X_test, y_test)     # s4 = 0.7905
print(s4)                         # Random Forest accuracy = 79.05%
y_pred4 = rf.predict(X_test)
cm4 = confusion_matrix(y_test, y_pred4)
print(cm3)
o/p: [[1402 604]
      [234 1760]]
```

Combining accuracies of all models:

```
models = pd.DataFrame({'Model':['Logistic Regression',
                                'Naive Bayes',
                                'SVM', 'Random Forest'],
                       'Accuracy_score':[s1, s2, s3, s4]})
print(models)
```

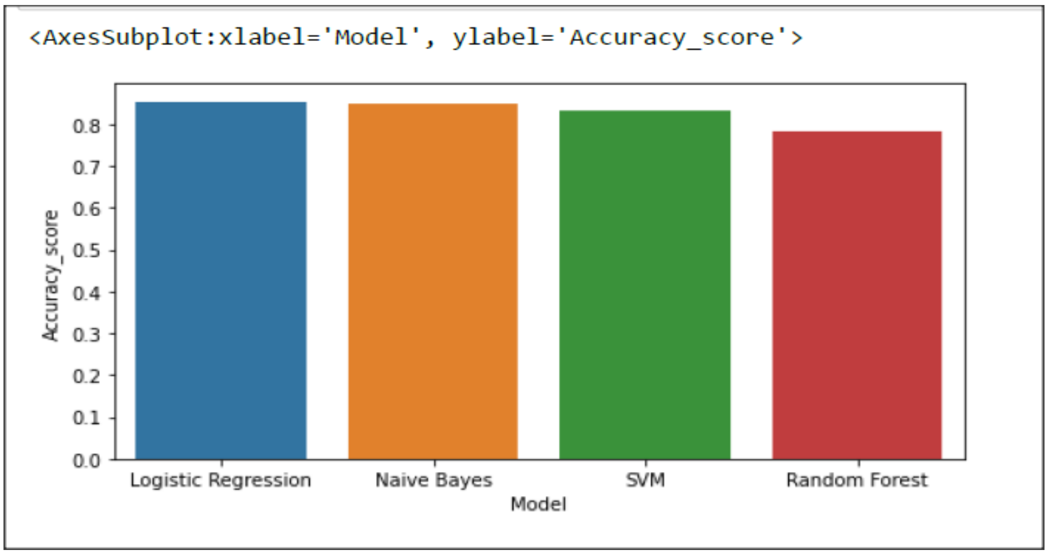


	Model	Accuracy_score
0	Logistic Regression	0.85375
1	Naive Bayes	0.84950
2	SVM	0.83325
3	Random Forest	0.78400

Plotting the accuracies:

```
fig, ax = plt.subplots(figsize=(8, 4))
sns.barplot(x='Model', y='Accuracy_score', data=models)
```

```
<AxesSubplot:xlabel='Model', ylabel='Accuracy_score'>
```



Outcomes:

- The main outcome of this research is that we are able to find out sentiment of given text and find out whether it is depressed tweet or not.
- It is found that highest accuracy is obtained through 'Logistic Regression' model as compared to other models.
- Although its accuracy is 85.3%, it is able to predict most of the results correctly as expected.
- These models can also be used to predict the sentiment of given text by training the them with some part of data.
- NLP is the main technique which has helped to perform sentiment analysis.

Conclusion

Sentiment analysis is an important aspect to understand others' opinions and feelings. This project is able to analyze the sentiments of each tweet expressed in twitter dataset. Also with the help of machine learning models, we are able to predict the depressed tweets. It is going to be very helpful because if someone is suffering from depression, the health experts can identify them and take necessary measures to get out of it.

On summarizing, this project has introduced the concept of sentiment analysis, its objective, application and different NLP techniques to pre-process the data. Owing to numerous challenging research problems and a wide variety of practical applications, sentiment analysis has been a very active research area in several computer science fields.

Not only twitter, sentiment analysis can also be performed on reviews of some situation in any of the social media apps and identify the thoughts of people about it. Future studies can consider performing the sentiment analysis in real time to resemble the current scenarios, and authorities can take quick action. The data can be further refined for better performance. In this way, it is going to be a highly useful technique in all fields.

References

a) Mitchell, T. M. (2018). *Machine Learning.* New Delhi: Mc Graw Hill.
https://www.cin.ufpe.br/~cavmj/Machine%20-%20Learning%20
%20Tom%20Mitchell.pdf

b) Machine Learning tutorial, Simplilearn
https://www.simplilearn.com/tutorials/machine-learning-tutorial

c) Analysis of public sentiment on covid-19 vaccination using Twitter, Research paper (August, 2022)

d) A Quick guide to sentiment analysis, Edureka
https://youtu.be/O_B7XLfx0ic

e) Depression: Twitter Dataset and Feature Extraction, Kaggle
https://www.kaggle.com/datasets/infamouscoder/mental-health-social-media

f) Getting started with Text Preprocessing, Kaggle
https://www.kaggle.com/code/sudalairajkumar/getting-started-with-text-preprocessing

g) NLP Tutorial, Javatpoint
https://www.javatpoint.com/nlp

h) Introduction to nltk and techniques implemented using that library
https://www.geeksforgeeks.org/introduction-to-nltk-tokenization-stemming-lemmatization-pos-tagging/

-------------------------THANK YOU--------------------------