TWITTER SENTIMENT ANALYSIS

Introduction

Detecting hate speech in tweets involves classifying tweets as either containing racist or sexist sentiment or not. To accomplish this using Python libraries, we can employ NLP techniques and machine learning algorithms. By analyzing the text content and applying sentiment analysis models, we can train a classifier to distinguish between tweets with hate speech and those without.

Sentiment Analysis

Sentiment analysis, also known as opinion mining, is a branch of natural language processing (NLP) that involves the use of computational techniques to determine and extract subjective information from text data. It aims to analyze and understand the sentiment, emotions, attitudes, and opinions expressed within a given piece of text.

The primary goal of sentiment analysis is to automatically classify the sentiment of a text document, such as a tweet, review, or customer feedback, into different categories, typically positive, negative, or neutral. However, sentiment analysis can also include more fine-grained sentiment classifications, such as very positive, positive, neutral, negative, and very negative.

Sentiment analysis techniques leverage various approaches, including machine learning algorithms, lexicon-based methods, and rule-based systems. These techniques process text data by examining patterns, semantic structures, linguistic features, and context to determine the sentiment orientation.

IMPORT NECESSARRY LIBRARIES

In [6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [104]:
```
from sklearn import model_selection, preprocessing, linear_model, metrics
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer, HashingVectorizer
from sklearn import ensemble
from lightgbm import LGBMClassifier
from sklearn.metrics import roc_auc_score, roc_curve
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from nltk.corpus import stopwords
from textblob import TextBlob
from textblob import Word
#nltk.download('wordnet')
from textblob import TextBlob
from termcolor import colored
from warnings import filterwarnings
filterwarnings('ignore')
import re
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from sklearn import set_config
set_config(print_changed_only = False)
```

In [8]:
```
test_set = pd.read_csv(r"C:\Desktop\Data Analyst Project\Sentiment Analysis\test.csv", encoding = "utf-8",
                       engine = "python",
                       header = 0)
train_set = pd.read_csv(r"C:\Desktop\Data Analyst Project\Sentiment Analysis\train.csv", encoding = "utf-8",
                       engine = "python",
                       header = 0)
```

In [9]:
```
print(colored("\nDATASETS WERE SUCCESFULLY LOADED...", color = "orange", attrs = ["dark", "bold"]))
```

```
DATASETS WERE SUCCESFULLY LOADED...
```

first five rows from train data set

In [10]:
```
train_set.head(n = 5).style.background_gradient(cmap = "PiYG")
```

Out[10]:

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run
1	2	0	@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disapointed #getthanked
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in urð□□±!!! ð□□□ð□□□ð□□□ð□□□ð□□¦ð□□¦ð□□¦
4	5	0	factsguide: society now #motivation

First Five rows from test dataset

In [11]:
```
test_set.head(n=5).style.background_gradient(cmap='PiYG')
```

Out[11]:

	id	tweet
0	31963	#studiolife #aislife #requires #passion #dedication #willpower to find #newmaterialsâ□¦
1	31964	@user #white #supremacists want everyone to see the new â□□ #birdsâ□□ #movie â□□ and hereâ□□s why
2	31965	safe ways to heal your #acne!! #altwaystoheal #healthy #healing!!
3	31966	is the hp and the cursed child book up for reservations already? if yes, where? if no, when? ð□□□ð□□□ð□□□ #harrypotter #pottermore #favorite
4	31967	3rd #bihday to my amazing, hilarious #nephew eli ahmir! uncle dave loves you and missesâ□¦

In [12]:
```
#shape
```

In [13]:
```
format(train_set.shape)
```

Out[13]:
```
'(31962, 3)'
```

In [14]:
```
#format(test_set.shape)
```

In [15]:
```
train_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31962 entries, 0 to 31961
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      31962 non-null  int64 
 1   label   31962 non-null  int64 
 2   tweet   31962 non-null  object
dtypes: int64(2), object(1)
memory usage: 749.2+ KB
```

In [16]:
```
test_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17197 entries, 0 to 17196
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      17197 non-null  int64 
 1   tweet   17197 non-null  object
dtypes: int64(1), object(1)
memory usage: 268.8+ KB
```

In [17]:
```
train_set.groupby("label").count().style.background_gradient(cmap="autumn")
```

Out[17]:

	id	tweet
label		
0	29720	29720
1	2242	2242

In [18]:
```
train_set_len = train_set['tweet'].str.len()
test_set_len = test_set['tweet'].str.len()
print("train data length :" , train_set_len)
print("test data length :" , test_set_len)
```

```
train data length : 0        102
1        122
2         21
3         86
4         39
        ...
31957     68
31958    131
31959     63
31960     67
31961     32
Name: tweet, Length: 31962, dtype: int64
test data length : 0         90
1        101
2         71
3        142
4         93
        ...
17192    108
17193     96
17194    145
17195    104
17196     64
Name: tweet, Length: 17197, dtype: int64
```

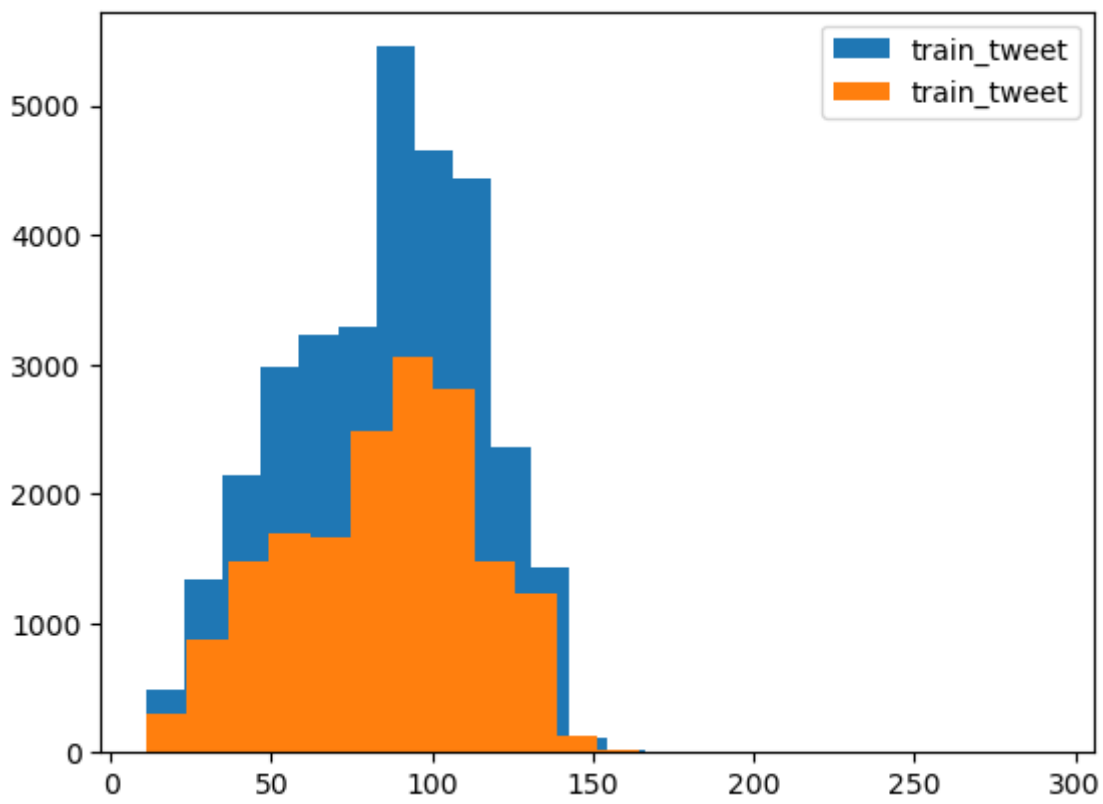
In [19]:
```
pos = 100*len(train_set.loc[train_set['label']==0, 'label'])/len(train_set['label'])
neg=100*len(train_set.loc[train_set['label']==1, 'label'])/len(train_set['label'])
```

In [20]:
```
print(f'Percentage of Negative Sentiment tweets is {pos}')
print(f'Percentage of Postitive Sentiment tweets is {neg}')
print('\nClearly, herre we can see the data ')
```

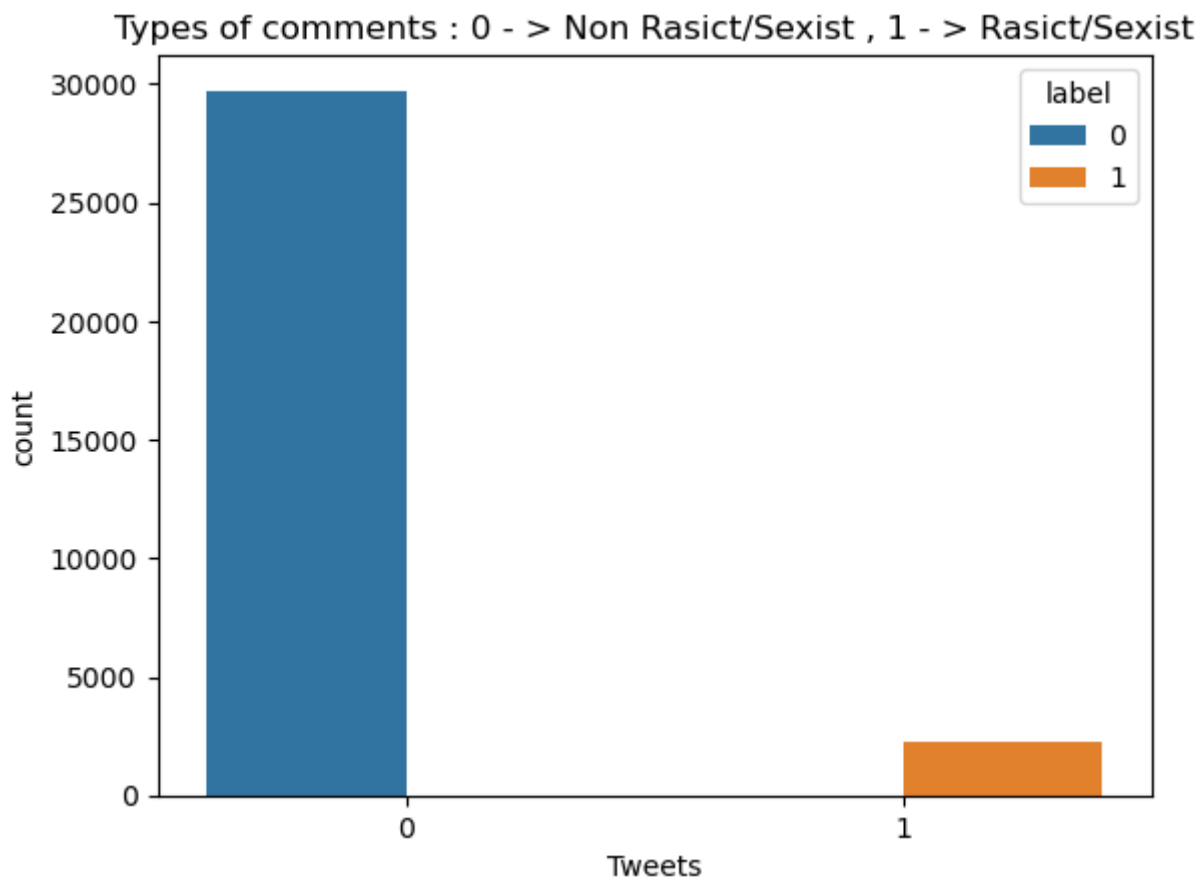
```
Percentage of Negative Sentiment tweets is 92.98542018647143
Percentage of Postitive Sentiment tweets is 7.014579813528565

Clearly, herre we can see the data
```

In [21]:
```
#data Exploration
```

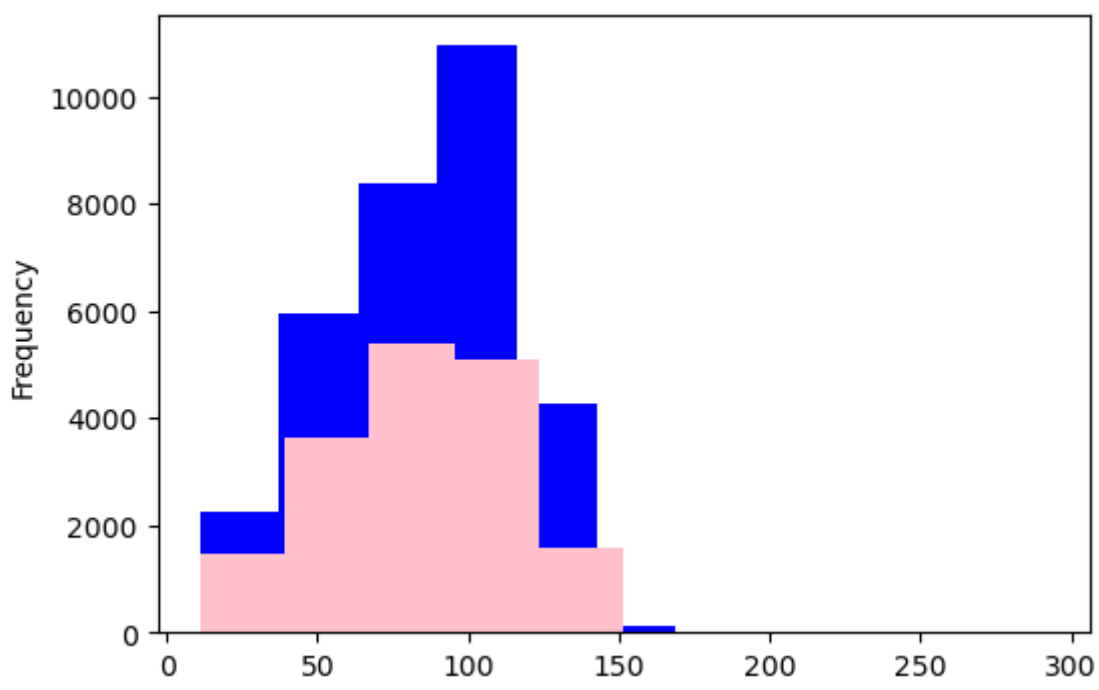
In [22]:
```
plt.hist(train_set_len, bins=22, label ='train_tweet')
plt.hist(test_set_len, bins=22, label = 'train_tweet')
plt.legend()
plt.show()
```



In [23]:
```
sns.countplot(data=train_set, x='label', hue='label')
plt.title('Types of comments : 0 - > Non Rasict/Sexist , 1 - > Rasict/Sexist')
plt.xlabel('Tweets')
plt.show()
```



In [24]:
```
clength_train = train_set['tweet'].str.len().plot.hist(color = 'blue', figsize = (6, 4))
length_test = test_set['tweet'].str.len().plot.hist(color = 'pink', figsize = (6, 4))
```



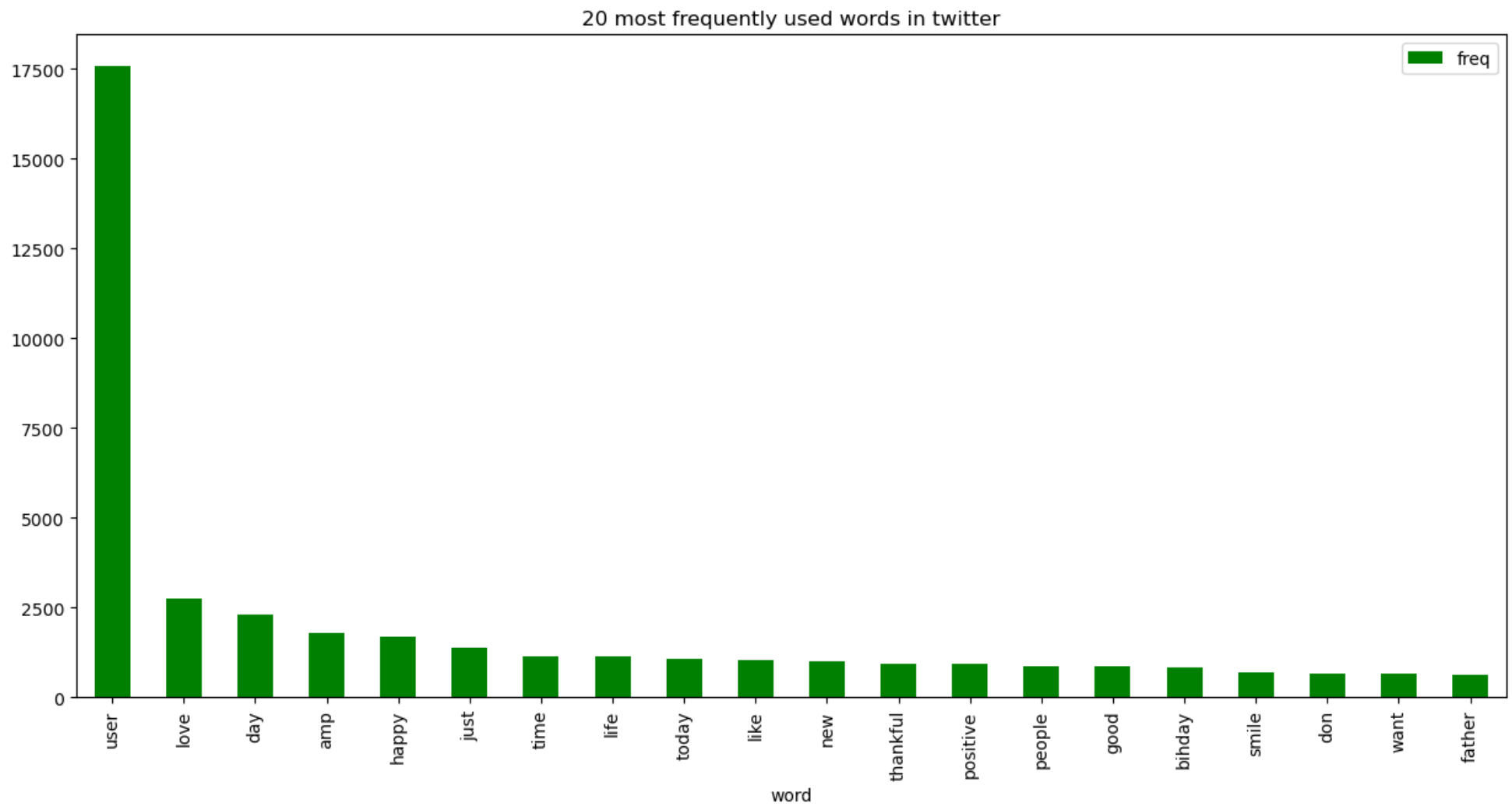
In [25]:
```
c=CountVectorizer(stop_words='english')
word=c.fit_transform(train_set.tweet)
summation=word.sum(axis=0)
print(summation)
```

```
[[ 51  28   2 ... 272   1   2]]
```

In [26]:
```
freq=[(word,summation[0,i]) for word,i in c.vocabulary_.items()]
freq=sorted(freq,key=lambda x:x[1],reverse=True)
frequency = pd.DataFrame(freq, columns=['word', 'freq'])
print(frequency)
```

```
           word   freq
0          user  17577
1          love   2749
2           day   2311
3           amp   1776
4         happy   1686
...         ...    ...
41099       isz      1
41100  airwaves      1
41101    mantle      1
41102   shirley      1
41103   chisolm      1

[41104 rows x 2 columns]
```

In [27]:
```
#most frequentlyy used words
df=frequency.head(20).plot(x='word', y='freq', kind='bar', figsize=(15, 7), color = 'green')
plt.title("20 most frequently used words in twitter")
plt.show()
```



In [28]:
```
#Count number of words
def num_of_words(df):
    df['word_count'] = df['tweet'].apply(lambda x : len(str(x).split(" ")))
    print(df[['tweet','word_count']].head())
```

In [29]:
```
num_of_words(train_set)
```

```
                                               tweet  word_count
0   @user when a father is dysfunctional and is s...          21
1  @user @user thanks for #lyft credit i can't us...          22
2                                bihday your majesty           5
3  #model   i love u take with u all the time in ...          17
4             factsguide: society now    #motivation           8
```

In [30]:
```
num_of_words(test_set)
```

```
                                               tweet  word_count
0  #studiolife #aislife #requires #passion #dedic...          12
1   @user #white #supremacists want everyone to s...          20
2  safe ways to heal your #acne!!    #altwaystohe...          15
3  is the hp and the cursed child book up for res...          24
4    3rd #bihday to my amazing, hilarious #nephew...          18
```

In [31]:
```
#Count number of characters
def num_of_chars(train_set):
    train_set['char_count'] = train_set['tweet'].str.len() ## this also includes spaces
    print(train_set[['tweet','char_count']].head())
```

In [32]:
```
num_of_chars(train_set)
```

```
                                               tweet  char_count
0   @user when a father is dysfunctional and is s...         102
1  @user @user thanks for #lyft credit i can't us...         122
2                                bihday your majesty          21
3  #model   i love u take with u all the time in ...          86
4             factsguide: society now    #motivation          39
```

In [33]:
```
num_of_chars(test_set)
```

```
                                               tweet  char_count
0  #studiolife #aislife #requires #passion #dedic...          90
1   @user #white #supremacists want everyone to s...         101
2  safe ways to heal your #acne!!    #altwaystohe...          71
3  is the hp and the cursed child book up for res...         142
4    3rd #bihday to my amazing, hilarious #nephew...          93
```

In [34]:
```
#Number of stopwords
set(stopwords.words('english'))
```

Out[34]:
```
{'a',
 'about',
 'above',
 'after',
 'again',
 'against',
 'ain',
 'all',
 'am',
 'an',
 'and',
 'any',
 'are',
 'aren',
 "aren't",
 'as',
 'at',
 'be',
 'because',
```

In [35]:
```
stop = stopwords.words('english')
```

In [36]:
```
def stop_words(df):
    df['stopwords'] = df['tweet'].apply(lambda x: len([x for x in x.split() if x in stop]))
    print(df[['tweet','stopwords']].head())
```

In [37]:
```
stop_words(train_set)
```

```
                                               tweet  stopwords
0   @user when a father is dysfunctional and is s...         10
1  @user @user thanks for #lyft credit i can't us...          5
2                                bihday your majesty          1
3  #model   i love u take with u all the time in ...          5
4             factsguide: society now    #motivation          1
```

In [38]:
```
stop_words(test_set)
```

```
                                               tweet  stopwords
0  #studiolife #aislife #requires #passion #dedic...          1
1   @user #white #supremacists want everyone to s...          4
2  safe ways to heal your #acne!!    #altwaystohe...          2
3  is the hp and the cursed child book up for res...          8
4    3rd #bihday to my amazing, hilarious #nephew...          4
```

Number of special characters

In [39]:
```
def hash_tags(df) :
    df['hashtags'] = df['tweet'].apply(lambda x: len([x for x in x.split() if x.startswith('#')]))
    print(df[['tweet', 'hashtags']].head())
```

In [40]:
```
hash_tags(train_set)
```

```
                                               tweet  hashtags
0   @user when a father is dysfunctional and is s...         1
1  @user @user thanks for #lyft credit i can't us...         3
2                                bihday your majesty         0
3  #model   i love u take with u all the time in ...         1
4             factsguide: society now    #motivation         1
```

In [41]:
```
hash_tags(test_set)
```

```
                                               tweet  hashtags
0  #studiolife #aislife #requires #passion #dedic...         7
1   @user #white #supremacists want everyone to s...         4
2  safe ways to heal your #acne!!    #altwaystohe...         4
3  is the hp and the cursed child book up for res...         3
4    3rd #bihday to my amazing, hilarious #nephew...         2
```

number of numerics

In [45]:
```
def num_numerics(df):
    df['numerics'] = df['tweet'].apply(lambda x: len([x for x in x.split() if x.isdigit()]))
    print(df[['tweet', 'numerics']].head())
```

In [46]:
```
num_numerics(train_set)
```

	tweet	numerics
0	@user when a father is dysfunctional and is s...	0
1	@user @user thanks for #lyft credit i can't us...	0
2	bihday your majesty	0
3	#model i love u take with u all the time in ...	0
4	factsguide: society now #motivation	0

In [47]:
```
num_numerics(test_set)
```

	tweet	numerics
0	#studiolife #aislife #requires #passion #dedic...	0
1	@user #white #supremacists want everyone to s...	0
2	safe ways to heal your #acne!! #altwaystohe...	0
3	is the hp and the cursed child book up for res...	0
4	3rd #bihday to my amazing, hilarious #nephew...	0

In [ ]:
```
```

Clean and Process Dataset

In [48]:
```
#convert upper case to lower case
```

In [49]:
```
train_set["tweet"]= train_set["tweet"].apply(lambda x: " ". join (x.lower() for x in x.split()))
test_set["tweet"] = test_set["tweet"].apply(lambda x: " ". join (x.lower() for x in x.split()))
```

In [50]:
```
print(colored("\nDELETED SUCCESFULLY...", color = "green", attrs = ["dark", "bold"]))
```

```
DELETED SUCCESFULLY...
```

In [51]:
```
#delete punctuations
```

In [58]:
```
train_set["tweet"]=train_set["tweet"].str.replace('[^\w\s]', '')
test_set["tweet"]=test_set["tweet"].str.replace('[^\w\s]','')
train_set['tweet'] = train_set['tweet'].str.replace('\d','')
test_set['tweet'] = test_set['tweet'].str.replace('\d', '')
```

In [59]:
```
#delete stopwords from tweet
```

In [60]:
```
sw = stopwords.words("english")
train_set['tweet'] = train_set['tweet'].apply(lambda x: " ".join (x for x in x.split() if x not in sw))
test_set['tweet'] = test_set['tweet'].apply(lambda x: " ".join (x for x in x.split() if x not in sw))
print(colored("\nSTOPWORDS DELETED SUCCESFULLY...", color = "green", attrs = ["dark", "bold"]))
```

```
STOPWORDS DELETED SUCCESFULLY...
```

In [61]:
```
train_set = train_set.drop("id", axis=1)
test_set = test_set.drop("id", axis=1)
print(colored("\n 'ID' Columns Dropped Successfully", color="green", attrs=["dark", "bold"]))
```

```
 'ID' Columns Dropped Successfully
```

CountVectorization

CounterVectorization is a SciKitLearn library takes any text document and returns each unique word as a feature with the count of number of times that word occurs

In [62]:
```
corpus = [
          'This is the first document.',
          'This document is the second document.',
          'And this is the third one.',
          'Is this the first document?',
         ]

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)
print(vectorizer.get_feature_names())
```

```
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
```

In [63]:
```
print(X.toarray())
```

```
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```

In [64]:
```
vectorizer2 = CountVectorizer(analyzer='word', ngram_range=(2, 2))
X2 = vectorizer2.fit_transform(corpus)
print(vectorizer2.get_feature_names())
```

```
['and this', 'document is', 'first document', 'is the', 'is this', 'second document', 'the first', 'the second', 'the third',
'third one', 'this document', 'this is', 'this the']
```

In [65]:
```
print(X2.toarray())
```

```
[[0 0 1 1 0 0 1 0 0 0 0 1 0]
 [0 1 0 1 0 1 0 1 0 0 1 0 0]
 [1 0 0 1 0 0 0 0 1 1 0 1 0]
 [0 0 1 0 1 0 1 0 0 0 0 0 1]]
```

Hashing Vectorizer

Hashing Vectorizer converts text to a matrix of occurrences using the hashing trick it converts a collection of text documents to a matrix of token occurrences.

In [66]:
```
corpus = [
          'This is the first document.',
          'This document is the second document.',
          'And this is the third one.',
          'Is this the first document?',
         ]
vectorizer = HashingVectorizer(n_features=2**4)
X = vectorizer.fit_transform(corpus)
print(X.shape)
```

```
(4, 16)
```

Lower Casing

Another pre-processing step which we will do is to transform our tweets into lower case. This avoids having multiple copies of the same words. For example, while calculating the word count, ‘Lower’ and ‘lower’ will be taken as different words.

In [67]:
```
def lower_case(df):
    df['tweet'] = df['tweet'].apply(lambda x: " ".join(x.lower() for x in x.split()))
    print(df['tweet'].head())
```

In [68]:
```
lower_case(train_set)
```

```
0    user father dysfunctional selfish drags kids d...
1    user user thanks lyft credit cant use cause do...
2                                    bihday majesty
3               model love u take u time urð ðððð ððð
4                      factsguide society motivation
Name: tweet, dtype: object
```

In [69]:
```
lower_case(test_set)
```

```
0    studiolife aislife requires passion dedication...
1    user white supremacists want everyone see new ...
2    safe ways heal acne altwaystoheal healthy healing
3    hp cursed child book reservations already yes ...
4    rd bihday amazing hilarious nephew eli ahmir u...
Name: tweet, dtype: object
```

frequent words removal

In [70]:
```
freq = pd.Series(' '.join(train_set['tweet']).split()).value_counts()[:10]
freq
```

Out[70]:
```
user     17473
love      2648
ð         2516
day       2230
â         1867
happy     1663
amp       1588
u         1141
im        1139
time      1110
dtype: int64
```

In [71]:
```
freq=list(freq.index)
```

In [72]:
```
def frequent_words_removal(df):
    df['tweet'] = df['tweet'].apply(lambda x: " ".join(x for x in x.split() if x not in freq))
    print(df['tweet'].head())
```

In [73]:
```
frequent_words_removal(train_set)
```

```
0    father dysfunctional selfish drags kids dysfun...
1    thanks lyft credit cant use cause dont offer w...
2                                      bihday majesty
3                              model take urð ðððð ðð
4                        factsguide society motivation
Name: tweet, dtype: object
```

In [74]:
```
frequent_words_removal(test_set)
```

```
0    studiolife aislife requires passion dedication...
1    white supremacists want everyone see new birds...
2    safe ways heal acne altwaystoheal healthy healing
3    hp cursed child book reservations already yes ...
4    rd bihday amazing hilarious nephew eli ahmir u...
Name: tweet, dtype: object
```

In [75]:
```
#Rare words removal
```

In [76]:
```
freq= pd.Series(' '.join(train_set['tweet']).split()).value_counts()[-10:]
freq
```

Out[76]:
```
socalled                  1
haleððââðð                 1
becauseyouturnedintoarat  1
cryingforever             1
anitgay                   1
threads                   1
destroyingpotential       1
onlyrelatives             1
myfamilysucks             1
chisolm                   1
dtype: int64
```

In [77]:
```
freq = list(freq.index)
```

In [78]:
```
def rare_words_removal(df):
    df['tweet'] = df['tweet'].apply(lambda x: " ".join(x for x in x.split() if x not in freq))
    print(df['tweet'].head())
```

In [79]:
```
rare_words_removal(train_set)
```

```
0    father dysfunctional selfish drags kids dysfun...
1    thanks lyft credit cant use cause dont offer w...
2                                      bihday majesty
3                              model take urð ðððð ðð
4                        factsguide society motivation
Name: tweet, dtype: object
```

In [80]:
```
rare_words_removal(test_set)
```

```
0    studiolife aislife requires passion dedication...
1    white supremacists want everyone see new birds...
2    safe ways heal acne altwaystoheal healthy healing
3    hp cursed child book reservations already yes ...
4    rd bihday amazing hilarious nephew eli ahmir u...
Name: tweet, dtype: object
```

Spelling Correction

In [81]:
```
def spell_correction(df):
    return df['tweet'][:5].apply(lambda x: str(TextBlob(x).correct()))
```

In [82]:
```
spell_correction(train_set)
```

Out[82]:
```
0    father dysfunctional selfish drags kiss dysfun...
1    thanks left credit can use cause dont offer wh...
2                                     midday majesty
3                             model take or ðððð ðð
4                        factsguide society motivation
Name: tweet, dtype: object
```

In [83]:
```
spell_correction(test_set)
```

Out[83]:
```
0    studiolife dislike requires passion education ...
1    white supremacists want everyone see new birds...
2    safe ways heal acne altwaystoheal healthy healing
3    he cursed child book reservations already yes ...
4    rd midday amazing hilarious nephew epi their u...
Name: tweet, dtype: object
```

Tokenization

In [84]:
```
def tokens(df):
    return TextBlob(df['tweet'][1]).words
```

In [85]:
```
tokens(train_set)
```

Out[85]:
```
WordList(['thanks', 'lyft', 'credit', 'cant', 'use', 'cause', 'dont', 'offer', 'wheelchair', 'vans', 'pdx', 'disapointed', 'ge
tthanked'])
```

In [86]:
```
tokens(test_set
      )
```

Out[86]:
```
WordList(['white', 'supremacists', 'want', 'everyone', 'see', 'new', 'birdsâ', 'movie', 'hereâs'])
```

Stemming

In [87]:
```
st = PorterStemmer()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1616\2044030079.py in <module>
----> 1 st = PorterStemmer()

NameError: name 'PorterStemmer' is not defined
```

In [ ]:
```
def stemming(df):
    return df['tweet'][:5].apply(lambda x: " ".join([st.stem(word) for word in x.split()]))
```

In [ ]:
```
stemming(train_set)
```

In [ ]:
```
stemming(test_set)
```

In [ ]:
```
#Lemmatization
#Lemmatization is the process of converting a word to its base form. The difference between stemming and lemmatization is, lemma
#



    #lemmatization(train)
    #lemmatization(test)
```

In [ ]:
```
def lemmatization(df):
        df['tweet'] = df['tweet'].apply(lambda x: " ".join([Word(word).lemmatize() for word in x.split()]))
        print(df['tweet'].head())
```

In [ ]:
```
lemmatization(train_set)
```

In [ ]:
```
lemmatization(test_set)
```

N-Grams

N-grams are the combination of multiple words used together. Ngrams with N=1 are called unigrams. Similarly, bigrams (N=2), trigrams (N=3) and so on. Unigrams do not usually contain as much information as compared to bigrams and trigrams. The basic principle behind n-grams is that they capture the language structure, like what letter or word is likely to follow the given one. The longer the n-gram (the higher the n), the more context you have to work with. Optimum length really depends on the application – if your n-grams are too short, you may fail to capture important differences. On the other hand, if they are too long, you may fail to capture the “general knowledge” and only stick to particular cases.

In [ ]:
```
def combination_of_words(df):
    return (TextBlob(df['tweet'][0]).ngrams(2))
```

In [ ]:
```
combination_of_words(train_set)
```

In [ ]:
```
combination_of_words(test_set)
```

Term Frequent

Term frequency is simply the ratio of the count of a word present in a sentence, to the length of the sentence.

In [ ]:
```
def term_frequency(df):
    tf1 = (df['tweet'][1:2]).apply(lambda x: pd.value_counts(x.split(" "))).sum(axis = 0).reset_index()
    tf1.columns = ['words','tf']
    return tf1.head()
```

In [88]:
```
term_frequency(train_set)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1616\744514178.py in <module>
----> 1 term_frequency(train_set)

NameError: name 'term_frequency' is not defined
```

In [89]:
```
term_frequency(test_set)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1616\1344080375.py in <module>
----> 1 term_frequency(test_set)

NameError: name 'term_frequency' is not defined
```

Bag of words

In [ ]:
```
```

Bag of Words (BoW) refers to the representation of text which describes the presence of words within the text data. The intuition behind this is that two similar text fields will contain similar kind of words, and will therefore have a similar bag of words. Further, that from the text alone we can learn something about the meaning of the document.

In [90]:
```
bow = CountVectorizer(max_features=1000, lowercase=True, ngram_range=(1,1),analyzer = "word")
train_bow = bow.fit_transform(train_set['tweet'])
train_bow
```

Out[90]:
```
<31962x1000 sparse matrix of type '<class 'numpy.int64'>'
        with 123253 stored elements in Compressed Sparse Row format>
```

Sentiment Analysis

In [91]:
```
def polarity_subjectivity(df):
    return df['tweet'][:5].apply(lambda x: TextBlob(x).sentiment)
```

In [92]:
```
polarity_subjectivity(train_set)
```

Out[92]:
```
0    (-0.5, 1.0)
1     (0.2, 0.2)
2     (0.0, 0.0)
3     (0.0, 0.0)
4     (0.0, 0.0)
Name: tweet, dtype: object
```

In [93]:
```
polarity_subjectivity(test_set)
```

Out[93]:
```
0                                    (0.0, 0.0)
1    (0.06818181818181818, 0.22727272727272727)
2                                    (0.5, 0.5)
3                                    (0.5, 1.0)
4                                  (0.55, 0.95)
Name: tweet, dtype: object
```

In [94]:
```
def sentiment_analysis(df):
    df['sentiment'] = df['tweet'].apply(lambda x: TextBlob(x).sentiment[0] )
    return df[['tweet','sentiment']].head()
```

In [95]:
```
sentiment_analysis(train_set)
```

Out[95]:

	tweet	sentiment
0	father dysfunctional selfish drags kids dysfun...	-0.5
1	thanks lyft credit cant use cause dont offer w...	0.2
2	bihday majesty	0.0
3	model take urð ðððð ððð	0.0
4	factsguide society motivation	0.0

In [96]:
```
sentiment_analysis(test_set)
```

Out[96]:

	tweet	sentiment
0	studiolife aislife requires passion dedication...	0.000000
1	white supremacists want everyone see new birds...	0.068182
2	safe ways heal acne altwaystoheal healthy healing	0.500000
3	hp cursed child book reservations already yes ...	0.500000
4	rd bihday amazing hilarious nephew eli ahmir u...	0.550000

In [97]:
```
#latest condition of dataset
```

In [98]:
```
train_set.head(n=10)
```

Out[98]:

	label	tweet	word_count	char_count	stopwords	hashtags	numerics	sentiment
0	0	father dysfunctional selfish drags kids dysfun...	21	102	10	1	0	-0.5
1	0	thanks lyft credit cant use cause dont offer w...	22	122	5	3	0	0.2
2	0	bihday majesty	5	21	1	0	0	0.0
3	0	model take urð ðððð ððð	17	86	5	1	0	0.0
4	0	factsguide society motivation	8	39	1	1	0	0.0
5	0	huge fan fare big talking leave chaos pay disp...	21	116	6	1	0	0.2
6	0	camping tomorrow dannyâ	12	74	0	0	0	0.0
7	0	next school year year examsð cant think school...	23	143	6	7	0	-0.4
8	0	land allin cavs champions cleveland clevelandc...	13	87	2	5	0	0.0
9	0	welcome gr	15	50	3	1	0	0.8

In [99]:
```
#Divide Dataset
x = train_set['tweet']
y = train_set['label']
train_x, test_x, train_y, test_y = model_selection.train_test_split(x, y, test_size = 0.20, shuffle = True, random_state = 11)
print(colored("\nDIVIDED SUCCESFULLY...", color = "green", attrs = ["dark", "bold"]))
```

```
DIVIDED SUCCESFULLY...
```

VECTORIZE DATA

Word Embeddings or Word vectorization is a methodology in NLP to map words or phrases from vocabulary to a corresponding vector of real numbers which used to find word predictions, word similarities/semantics.

"Count Vectors" method

In [100]:
```
vectorizer = CountVectorizer()
vectorizer.fit(train_x)

x_train_count = vectorizer.transform(train_x)
x_test_count = vectorizer.transform(test_x)

x_train_count.toarray()
```

Out[100]:
```
array([[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]], dtype=int64)
```

Logistic Regression Model

In [101]:
```
log = linear_model.LogisticRegression()
log_model = log.fit(x_train_count, train_y)
accuracy = model_selection.cross_val_score(log_model,
                                           x_test_count,
                                           test_y,
                                           cv = 20).mean()

print(colored("\nLogistic regression model with 'count-vectors' method", color = "red", attrs = ["dark", "bold"]))
print(colored("Accuracy ratio: ", color = "red", attrs = ["dark", "bold"]), accuracy)
```

```
Logistic regression model with 'count-vectors' method
Accuracy ratio:  0.9466624216300941
```

XGBoost model with "count-vectors" method

In [102]:
```
xgb = XGBClassifier()
xgb_model = xgb.fit(x_train_count,train_y)
accuracy = model_selection.cross_val_score(xgb_model,
                                           x_test_count,
                                           test_y,
                                           cv = 20).mean()

print(colored("\nXGBoost model with 'count-vectors' method", color = "red", attrs = ["dark", "bold"]))
print(colored("Accuracy ratio: ", color = "red", attrs = ["dark", "bold"]), accuracy)
```

```
XGBoost model with 'count-vectors' method
Accuracy ratio:  0.9419700235109719
```

Visualization with word cloud

In [103]:
```
#from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
#from PIL import Image

#tw_mask = np.array(Image.open('../input/masksforwordclouds/twitter_mask3.jpg'))

text = " ".join(i for i in train_set.tweet)

wc = WordCloud(background_color = "white",
               width = 600,
               height = 600,
               contour_width = 0,
               contour_color = "red",
               max_words = 1000,
               scale = 1,
               collocations = False,
               repeat = True,
               min_font_size = 1)

wc.generate(text)

plt.figure(figsize = [15, 15])
plt.imshow(wc)
plt.axis("off")
plt.show()
```

Thank You