Page No.
Date

Assignment C4.

★ Title :- Twitter Data Analysis.

Use twitter data for sentiment analysis. the dataset has 31962 tweets. Identify the tweets which are hate tweets & which are not.

★ Objective - Twitter Data Sentiment Analysis using Naive Bayes classifier.

★ outcomes :- Learn steps to do sentiment Analysis.

★ Theory -

Sentiment analysis is a process of determining whether a piece of writing (product / movie review, tweet etc) is positive, negative & neutral. It can be used to identify the customers or followers attitude towards a brand through the use of a variable such as context, tone, emotion etc.

Steps to perform sentiment analysis.

① Gather relvent tweets from twitter.

② Preprocessing :- preprocessing of a text data is an essential steps as to make the raw text ready for mining i.e. it became easier to extract information from the text & apply machine learning to it. If we skip this step then there is a higher chance that you are working with noisy & inconsistent data.

Page No.
Date

(3) Feature extraction - selection of useful words from the tweets is called as feature extraction. In feature extraction method, extract this aspect from pre-processed twitter dataset.

1. there are 3 different types of features namely unigram, bigram, n-gram features.
2. parts of speech tag such as like adjectives, verbs, adverbs, & noun are good indicates of subjectively & sentiment.
3. negation is another important but difficult feature to interpret. The presence of negation usually changes the polarity of the sentiment.

(4) Feature selection - correct feature selection technique are used in sentiment analysis that has got a significant role for identifying relevent attributes & increasing classification accuracy.

(5) Classification - for classification of tweets & naive bayes algorithm -

Naive bayes is a probablistic classifier inspired by bayes algorithm, under a simple assumption which is attribute of are conditionaly independent.

$$P(c|x) = \frac{P(x|c)\ P(c)}{P(x)}$$

The classification is conducted by deriving the maximum posterior which is maximal $P(C_i|x)$. with the above assumption applying to Bayes theorem. This assumption greatly reduce the computational cost by only counting the class distribution Even though than the assumption is not valid in most cases since the attribute are dependent surprisingly Naive Bayes has able to perform impressively

b. Svm classifier -

The Support vector machine (SVM) is known as to Perform well in sentiment analysis, Support vector machine analyzes the data, define the decision boundry it uses the kernal for computation which are Performed in input space. The input data are two sets of vector of size m each. Then every data which is represented as vector is classified into a clas

Conclusion -

Here, we are able to predict the tweet is positive negative or neutral.

```
!pip install gensim
!pip install wordcloud
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: gensim in ./.local/lib/python3.6/site-packages (3.8.3)
Requirement already satisfied: numpy>=1.11.3 in ./.local/lib/python3.6/site-packages (fr
Requirement already satisfied: smart-open>=1.8.1 in ./.local/lib/python3.6/site-packages
Requirement already satisfied: scipy>=0.18.1 in ./.local/lib/python3.6/site-packages (fr
Requirement already satisfied: six>=1.5.0 in /usr/lib/python3/dist-packages (from gensim
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (from smart-op
WARNING: You are using pip version 20.2.1; however, version 20.2.4 is available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade pip' co
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: wordcloud in ./.local/lib/python3.6/site-packages (1.8.0)
Requirement already satisfied: numpy>=1.6.1 in ./.local/lib/python3.6/site-packages (fro
Requirement already satisfied: pillow in ./.local/lib/python3.6/site-packages (from word
Requirement already satisfied: matplotlib in ./.local/lib/python3.6/site-packages (from
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in /usr/lib/pyth
Requirement already satisfied: cycler>=0.10 in /usr/lib/python3/dist-packages (from matp
Requirement already satisfied: kiwisolver>=1.0.1 in ./.local/lib/python3.6/site-packages
Requirement already satisfied: certifi>=2020.06.20 in ./.local/lib/python3.6/site-packag
Requirement already satisfied: python-dateutil>=2.1 in /usr/lib/python3/dist-packages (f
WARNING: You are using pip version 20.2.1; however, version 20.2.4 is available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade pip' co
```

```
pip install nltk
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: nltk in ./.local/lib/python3.6/site-packages (3.5)
Requirement already satisfied: tqdm in ./.local/lib/python3.6/site-packages (from nltk)
Requirement already satisfied: joblib in /usr/lib/python3/dist-packages (from nltk) (0.1
Requirement already satisfied: regex in ./.local/lib/python3.6/site-packages (from nltk)
Requirement already satisfied: click in ./.local/lib/python3.6/site-packages (from nltk)
WARNING: You are using pip version 20.2.1; however, version 20.2.4 is available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade pip' co
Note: you may need to restart the kernel to use updated packages.
```

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

import warnings


train = pd.read_csv('train_tweet.csv')
test = pd.read_csv('test_tweets.csv')

print(train.shape)
print(test.shape)
```

```
print(test.shape)
```

```
(31962, 3)
(17197, 2)
```

```
train.head()
```

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

```
test.head()
```

	id	tweet
0	31963	#studiolife #aislife #requires #passion #dedic...
1	31964	@user #white #supremacists want everyone to s...
2	31965	safe ways to heal your #acne!! #altwaystohe...
3	31966	is the hp and the cursed child book up for res...
4	31967	3rd #bihday to my amazing, hilarious #nephew...

```
train.isnull().any()
test.isnull().any()
```

```
id       False
tweet    False
dtype: bool
```

```
# checking out the negative comments from the train set

train[train['label'] == 0].head(10)
```

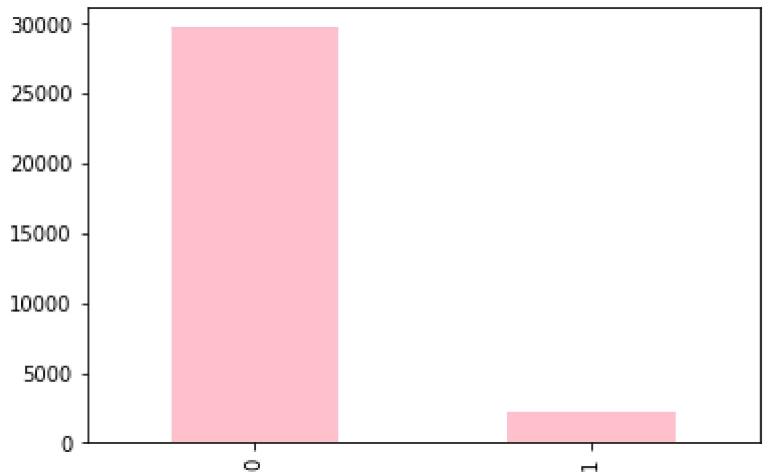
	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

```
# checking out the postive comments from the train set

train[train['label'] == 1].head(10)
```

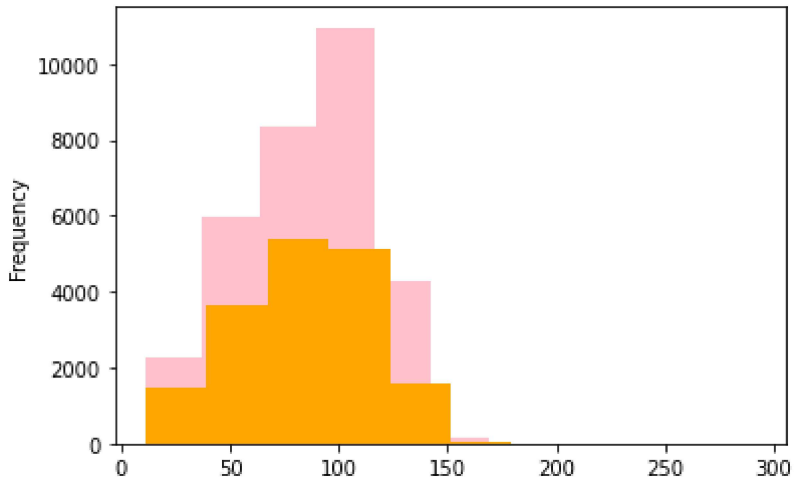
	id	label	tweet
13	14	1	@user #cnn calls #michigan middle school 'buil...
14	15	1	no comment! in #australia #opkillingbay #se...
17	18	1	retweet if you agree!
23	24	1	@user @user lumpy says i am a . prove it lumpy.
34	35	1	it's unbelievable that in the 21st century we'...
56	57	1	@user lets fight against #love #peace
68	69	1	ð□□©the white establishment can't have blk fol...
77	78	1	@user hey, white people: you can call people '...
82	83	1	how the #altright uses & insecurity to lu...
111	112	1	@user i'm not interested in a #linguistics tha...

```
train['label'].value_counts().plot.bar(color = 'pink', figsize = (6, 4))
```

```
<AxesSubplot:>
```



```
# checking the distribution of tweets in the data

length_train = train['tweet'].str.len().plot.hist(color = 'pink', figsize = (6, 4))
length_test = test['tweet'].str.len().plot.hist(color = 'orange', figsize = (6, 4))
```



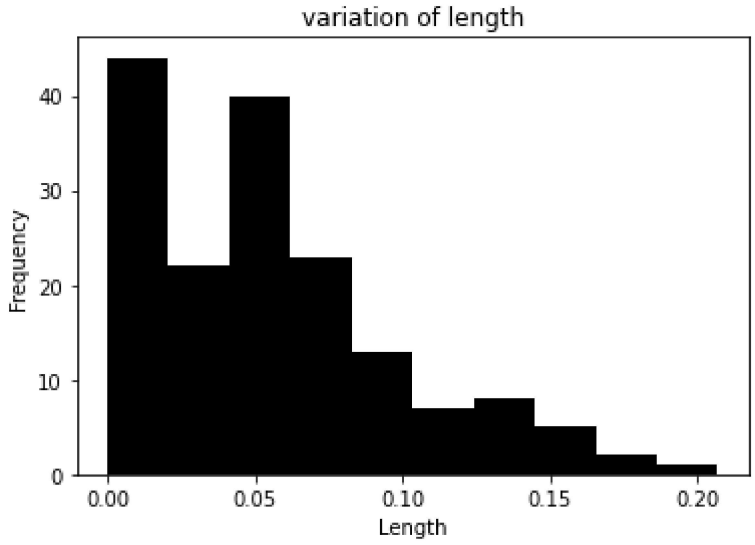

```
# adding a column to represent the length of the tweet

train['len'] = train['tweet'].str.len()
test['len'] = test['tweet'].str.len()

train.head(10)
```

	id	label	tweet	len
0	1	0	@user when a father is dysfunctional and is s...	102
1	2	0	@user @user thanks for #lyft credit i can't us...	122
2	3	0	bihday your majesty	21
3	4	0	#model i love u take with u all the time in ...	86
4	5	0	factsguide: society now #motivation	39
5	6	0	[2/2] huge fan fare and big talking before the...	116
6	7	0	@user camping tomorrow @user @user @user @use...	74
7	8	0	the next school year is the year for exams.ð□□...	143
8	9	0	we won!!! love the land!!! #allin #cavs #champ...	87
9	10	0	@user @user welcome here ! i'm it's so #gr...	50

```
train.groupby('label').describe()
```

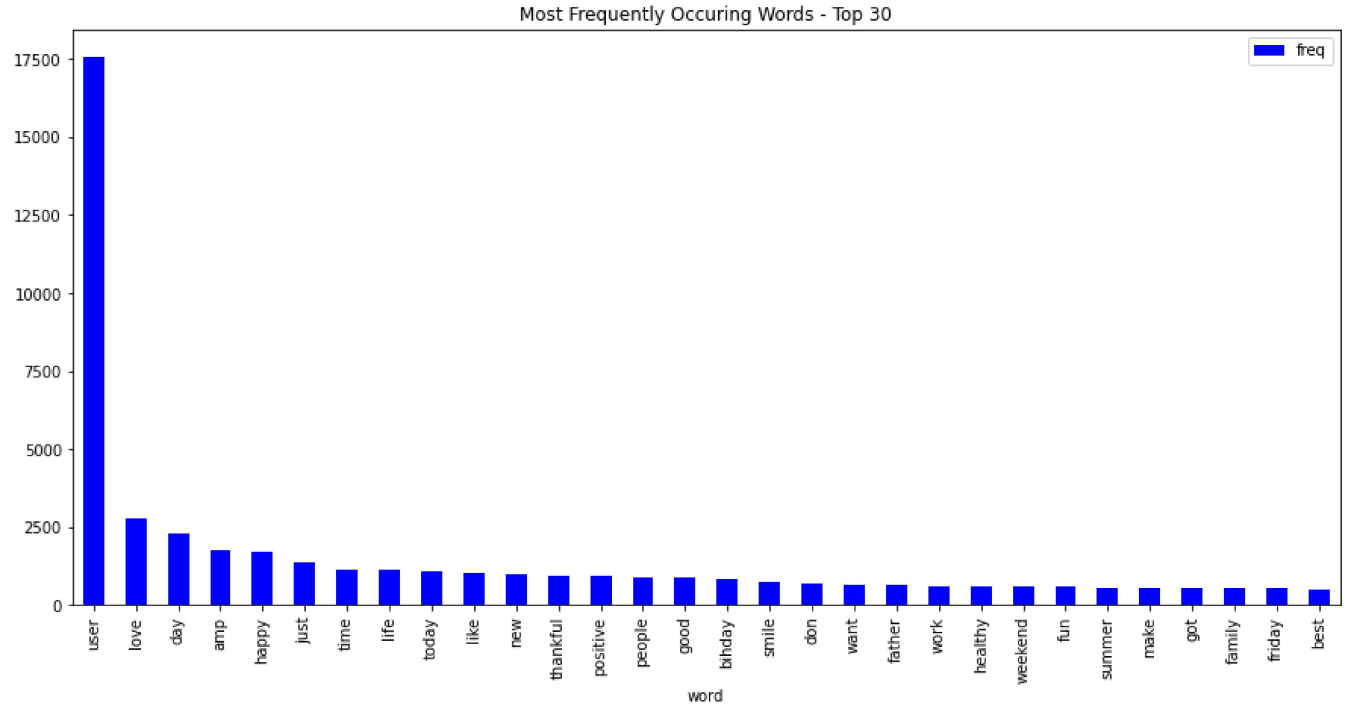
	id								len
	count	mean	std	min	25%	50%	75%	max	count
label									
0	29720.0	15974.454441	9223.783469	1.0	7981.75	15971.5	23965.25	31962.0	29720
1	2242.0	16074.806075	9267.955758	14.0	8075.25	16095.0	24022.00	31961.0	2242

```
train.groupby('len').mean()['label'].plot.hist(color = 'black', figsize = (6, 4),)
plt.title('variation of length')
plt.xlabel('Length')
plt.show()
```



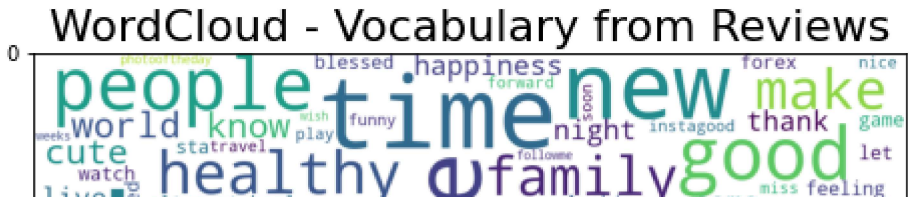

```
from sklearn.feature_extraction.text import CountVectorizer


cv = CountVectorizer(stop_words = 'english')
words = cv.fit_transform(train.tweet)

sum_words = words.sum(axis=0)

words_freq = [(word, sum_words[0, i]) for word, i in cv.vocabulary_.items()]
words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)

frequency = pd.DataFrame(words_freq, columns=['word', 'freq'])

frequency.head(30).plot(x='word', y='freq', kind='bar', figsize=(15, 7), color = 'blue')
plt.title("Most Frequently Occuring Words - Top 30")
```

```
Text(0.5, 1.0, 'Most Frequently Occuring Words - Top 30')
```

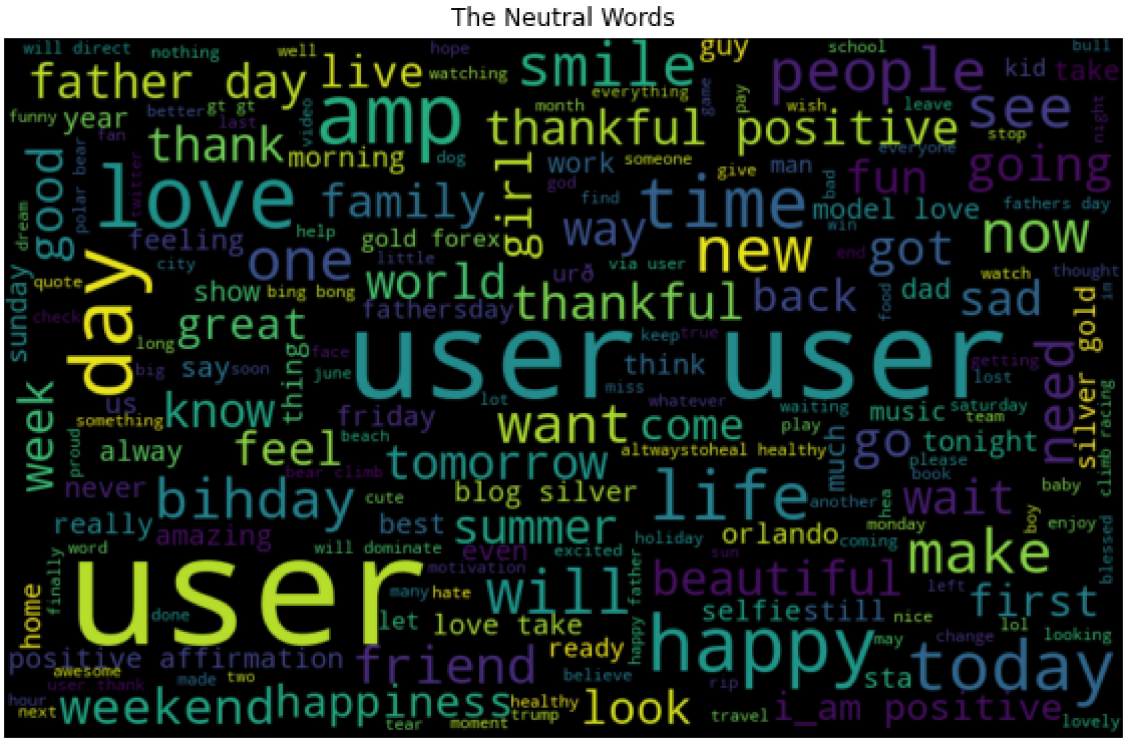



```
from wordcloud import WordCloud

wordcloud = WordCloud(background_color = 'white', width = 1000, height = 1000).generate_from_

plt.figure(figsize=(10,8))
plt.imshow(wordcloud)
plt.title("WordCloud - Vocabulary from Reviews", fontsize = 22)
```

```
Text(0.5, 1.0, 'WordCloud - Vocabulary from Reviews')
```



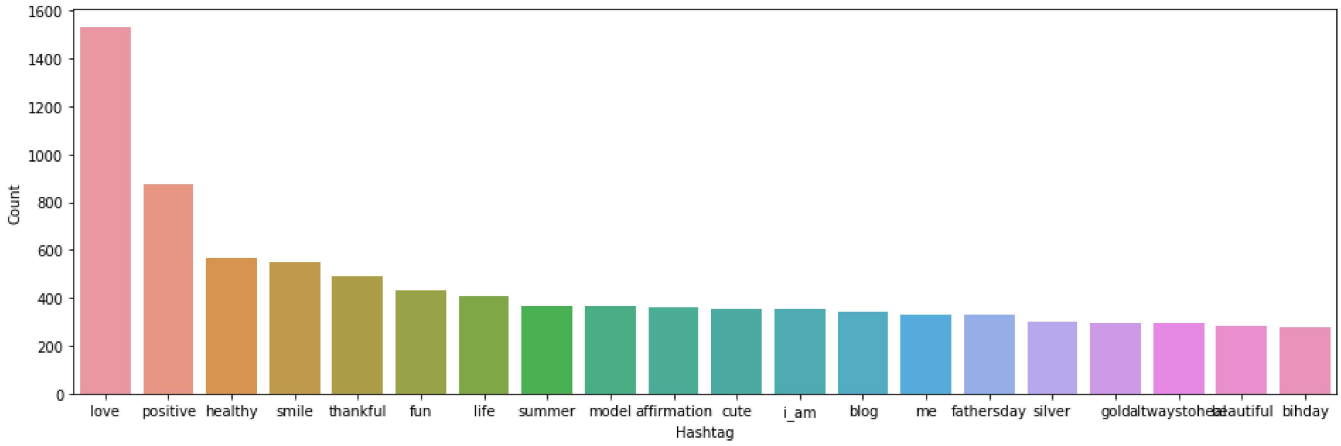

```
normal_words =' '.join([text for text in train['tweet'][train['label'] == 0]])

wordcloud = WordCloud(width=800, height=500, random_state = 0, max_font_size = 110).generate(
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title('The Neutral Words')
plt.show()
```



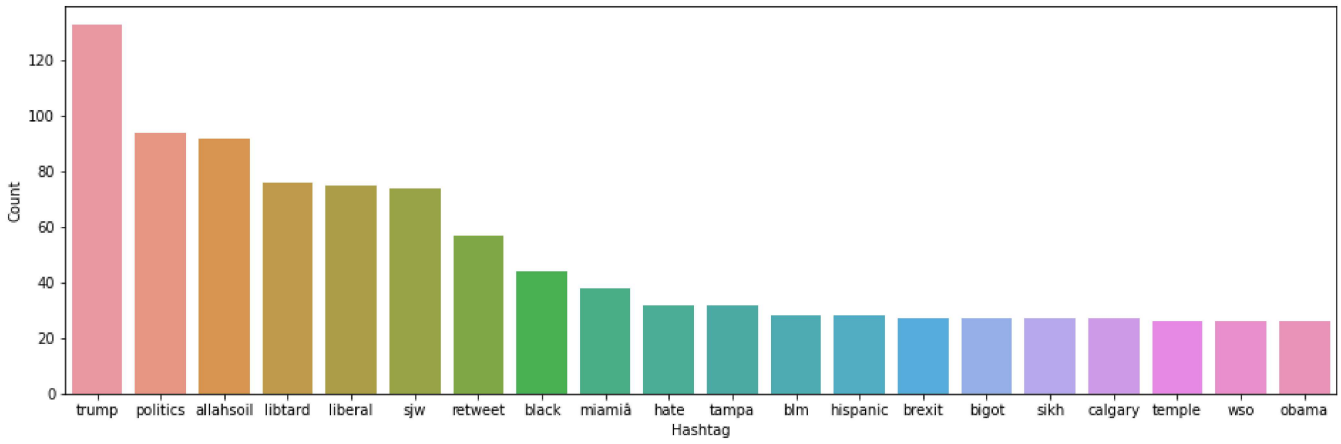

```
negative_words =' '.join([text for text in train['tweet'][train['label'] == 1]])

wordcloud = WordCloud(background_color = 'cyan', width=800, height=500, random_state = 0, max
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title('The Negative Words')
plt.show()
```

The Negative Words


```
# collecting the hashtags
import re
def hashtag_extract(x):
    hashtags = []

    for i in x:
        ht = re.findall(r"#(\w+)", i)
        hashtags.append(ht)

    return hashtags


# extracting hashtags from non racist/sexist tweets
HT_regular = hashtag_extract(train['tweet'][train['label'] == 0])

# extracting hashtags from racist/sexist tweets
HT_negative = hashtag_extract(train['tweet'][train['label'] == 1])

# unnesting list
HT_regular = sum(HT_regular,[])
HT_negative = sum(HT_negative,[])


a = nltk.FreqDist(HT_regular)
d = pd.DataFrame({'Hashtag': list(a.keys()),
                  'Count': list(a.values())})

# selecting top 20 most frequent hashtags
d = d.nlargest(columns="Count", n = 20)
plt.figure(figsize=(16,5))
ax = sns.barplot(data=d, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.show()
```




```
a = nltk.FreqDist(HT_negative)
d = pd.DataFrame({'Hashtag': list(a.keys()),
                  'Count': list(a.values())})

# selecting top 20 most frequent hashtags
d = d.nlargest(columns="Count", n = 20)
plt.figure(figsize=(16,5))
ax = sns.barplot(data=d, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.show()
```




```
# tokenizing the words present in the training set
tokenized_tweet = train['tweet'].apply(lambda x: x.split())
```

```
# importing gensim
import gensim

# creating a word to vector model
model_w2v = gensim.models.Word2Vec(
            tokenized_tweet,
            size=200, # desired no. of features/independent variables
            window=5, # context window size
            min_count=2,
            sg = 1, # 1 for skip-gram model
            hs = 0,
            negative = 10, # for negative sampling
            workers= 2, # no.of cores
            seed = 34)

model_w2v.train(tokenized_tweet, total_examples= len(train['tweet']), epochs=20)
```

```
(6109997, 8411580)
```

```
model_w2v.wv.most_similar(positive = "dinner")
```

```
[('spaghetti', 0.6450440287590027),
 ('#prosecco', 0.6306897401809692),
 ('4pm', 0.6025007963180542),
 ('bay.', 0.5962221026420593),
 ('#wanderlust', 0.5905485153198242),
 ('podium', 0.5890529751777649),
 ('spa', 0.5868688225746155),
 ('#trailrunning', 0.5861507654190063),
 ('#demoday', 0.5850203037261963),
 ('dialogue', 0.5843442678451538)]
```

```
model_w2v.wv.most_similar(positive = "cancer")
```

```
[('harassment', 0.7236580848693848),
 ("society's", 0.7194166779518127),
 ('law.', 0.7146366834640503),
 ('ownership', 0.7116885781288147),
 ('speeches', 0.7088077068328857),
 ('inflict', 0.7077646851539612),
 ('level.', 0.706573486328125),
 ('champion,', 0.7058138251304626),
 ('#merica', 0.704941987991333),
 ('targeted', 0.7042679786682129)]
```

```
model_w2v.wv.most_similar(positive = "apple")
```

```
[('mytraining', 0.7012615203857422),
 ('"mytraining"', 0.700869083404541),
 ('training"', 0.6904847621917725),
 ('app,', 0.6299394369125366),
 ('"my', 0.6201056241989136),
```

```
       ('humans.', 0.5938655138015747),
       ('ta', 0.591256856918335),
       ('app', 0.5735200643539429),
       ('bees', 0.5694879293441772),
       ("others'", 0.5666430592536926)]
```

```
model_w2v.wv.most_similar(negative = "hate")
```

```
     [('#apple', 0.00541564030572772),
      ('#hype', -0.03733295947313309),
      ('#yay', -0.04639853164553642),
      ('#games', -0.04658985137939453),
      ('#fundraising', -0.058874525129795074),
      ('#wednesdaywisdom', -0.06913787871599197),
      ('members', -0.07289311289787292),
      ('#grateful', -0.0768258273601532),
      ('you?', -0.08171205222606659),
      ('hu', -0.08282715827226639)]
```

```
from tqdm import tqdm
tqdm.pandas(desc="progress-bar")
from gensim.models.doc2vec import LabeledSentence


def add_label(twt):
    output = []
    for i, s in zip(twt.index, twt):
        output.append(LabeledSentence(s, ["tweet_" + str(i)]))
    return output

# label all the tweets
labeled_tweets = add_label(tokenized_tweet)

labeled_tweets[:6]
```

```
     /usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: DeprecationWarning: Call
       after removing the cwd from sys.path.
     [LabeledSentence(words=['@user', 'when', 'a', 'father', 'is', 'dysfunctional', 'and', 'i
      LabeledSentence(words=['@user', '@user', 'thanks', 'for', '#lyft', 'credit', 'i', "can
      LabeledSentence(words=['bihday', 'your', 'majesty'], tags=['tweet_2']),
      LabeledSentence(words=['#model', 'i', 'love', 'u', 'take', 'with', 'u', 'all', 'the',
      LabeledSentence(words=['factsguide:', 'society', 'now', '#motivation'], tags=['tweet_4
      LabeledSentence(words=['[2/2]', 'huge', 'fan', 'fare', 'and', 'big', 'talking', 'before
```

```
# removing unwanted patterns from the data

import re
import nltk

nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

```
from nltk.stem.porter import PorterStemmer
```

```
[nltk_data] Downloading package stopwords to /home/aarti/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
train_corpus = []

for i in range(0, 31962):
  review = re.sub('[^a-zA-Z]', ' ', train['tweet'][i])
  review = review.lower()
  review = review.split()

  ps = PorterStemmer()

  # stemming
  review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]

  # joining them back with space
  review = ' '.join(review)
  train_corpus.append(review)


test_corpus = []

for i in range(0, 17197):
  review = re.sub('[^a-zA-Z]', ' ', test['tweet'][i])
  review = review.lower()
  review = review.split()

  ps = PorterStemmer()

  # stemming
  review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]

  # joining them back with space
  review = ' '.join(review)
  test_corpus.append(review)


# creating bag of words

from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(max_features = 2500)
x = cv.fit_transform(train_corpus).toarray()
y = train.iloc[:, 1]

print(x.shape)
print(y.shape)
```

```
(31962, 2500)
(31962,)
```

```
# creating bag of words

from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(max_features = 2500)
x_test = cv.fit_transform(test_corpus).toarray()

print(x_test.shape)
```

```
(17197, 2500)
```

```
# splitting the training data into train and valid sets

from sklearn.model_selection import train_test_split

x_train, x_valid, y_train, y_valid = train_test_split(x, y, test_size = 0.25, random_state =

print(x_train.shape)
print(x_valid.shape)
print(y_train.shape)
print(y_valid.shape)
```

```
(23971, 2500)
(7991, 2500)
(23971,)
(7991,)
```

```
# standardization

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x_train = sc.fit_transform(x_train)
x_valid = sc.transform(x_valid)
x_test = sc.transform(x_test)
```

```
/usr/lib/python3/dist-packages/sklearn/utils/validation.py:475: DataConversionWarning: D
  warnings.warn(msg, DataConversionWarning)
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
```

```
model = RandomForestClassifier()
model.fit(x_train, y_train)

y_pred = model.predict(x_valid)

print("Training Accuracy :", model.score(x_train, y_train))
print("Validation Accuracy :", model.score(x_valid, y_valid))

# calculating the f1 score for the validation set
print("F1 score :", f1_score(y_valid, y_pred))

# confusion matrix
cm = confusion_matrix(y_valid, y_pred)
print(cm)
```

```
/usr/lib/python3/dist-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarnir
  from numpy.core.umath_tests import inner1d
Training Accuracy : 0.9944933461265696
Validation Accuracy : 0.951445376048054
F1 score : 0.6008230452674896
[[7311  121]
 [ 267  292]]
```

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(x_train, y_train)

y_pred = model.predict(x_valid)

print("Training Accuracy :", model.score(x_train, y_train))
print("Validation Accuracy :", model.score(x_valid, y_valid))

# calculating the f1 score for the validation set
print("f1 score :", f1_score(y_valid, y_pred))

# confusion matrix
cm = confusion_matrix(y_valid, y_pred)
print(cm)
```

```
Training Accuracy : 0.984773267698469
Validation Accuracy : 0.9410586910274058
f1 score : 0.5915004336513443
[[7179  253]
 [ 218  341]]
```