Parshvanath Charitable Trust's
A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

Department of Information Technology

Academic Year: 2018-19 **Name of Student: Pratiksha patil**
Semester: VII **Class / Branch: IT**

Project Title:

Group No: 14

Group Members: Riddhi Prajapati
Pratiksha Patil
Aafreen Shaikh

Guide: Prof. Sunil N. Sushir

Co Guide: -

In [42]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#for sentiment
import nltk
from wordcloud import WordCloud,STOPWORDS
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer, ENGLISH_STOP_WORDS

#For Model
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score, roc_auc_score


import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
train  = pd.read_csv("C:\\Users\\Admin\\Downloads\\train_E6oV3lV.csv (1)\\train_E6oV3lV.csv")
```

In [3]:
```
train.shape
```

Out[3]: (31962, 3)

Firstly, we have imported all the required libraries for the manipulation, visualization of the data set. As mentioned above the required libraries for sentiment classification and modelling are imported as well. Then the data set is imported in python i.e. csv file.

```
In [4]: df = train
        df
```

Out[4]:

	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
5	6	0	[2/2] huge fan fare and big talking before the...
6	7	0	@user camping tomorrow @user @user @user @use...
7	8	0	the next school year is the year for exams.ð...
8	9	0	we won!!! love the land!!! #allin #cavs #champ...
9	10	0	@user @user welcome here ! i'm it's so #gr...

Remove twitter handlers i.e., @user

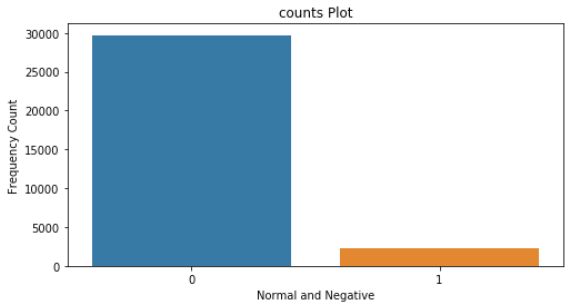
```
In [5]: train['cleaned_tweet'] = train.tweet.apply(lambda x: ' '.join([word for word in x.split() if not word.startswith('@')]))
```

Here the tweets of users are displayed randomly from the entire data set. Label 0 is allotted to positive tweets whereas label 1 is allotted to negative tweets.

```
In [5]: train['cleaned_tweet'] = train.tweet.apply(lambda x: ' '.join([word for word in x.split() if not word.startswith('@')]))
```

```
In [25]: plt.figure(figsize=(8,4))
         ax = sns.countplot(x="label", data=df)
         plt.title('counts Plot')
         plt.xlabel('Normal and Negative')
         plt.ylabel('Frequency Count')
```

```
Out[25]: Text(0,0.5,'Frequency Count')
```



Now here, the all the words from the entire data set are separated and saved in “cleaned tweets” but words starting with @ are not taken. Then a graph is plotted of both the positive and negative content and it is clearly seen that in our data the positive tweets are more in number.

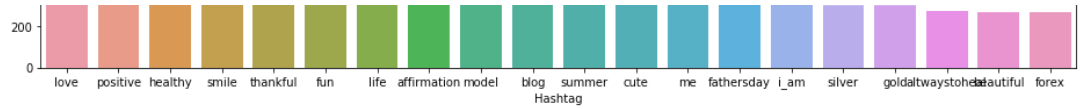
Hashtags

In [6]:
```
#Select all words from normal tweet
normal_words = ' '.join([word for word in train['cleaned_tweet'][train['label'] == 0]])
#Collect all hashtags
pos_htag = [htag for htag in normal_words.split() if htag.startswith('#')]
#Remove hashtag symbol (#)
pos_htag = [pos_htag[i][1:] for i in range(len(pos_htag))]
#Count frequency of each word
pos_htag_freqcount = nltk.FreqDist(pos_htag)
pos_htag_df = pd.DataFrame({'Hashtag' : list(pos_htag_freqcount.keys()),
                            'Count' : list(pos_htag_freqcount.values())})
```

In [7]:
```
#normal_words
```

In [8]:
```
#Select top 20 most frequent hashtags and plot them
most_frequent = pos_htag_df.nlargest(columns="Count", n = 20)
plt.figure(figsize=(16,5))
ax = sns.barplot(data=most_frequent, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.show()
```

Here, again all the words will be splitted again and only those with label 0 will be accepted and saved in “normal tweets”. All the hashtags will be removed and frequency of every word will be calculated. Then command will be given to display n number of most frequent positive words and based on the output a graph will be plotted.



In [9]:
```
#Repeat same steps for negative tweets
negative_words = ' '.join([word for word in train['cleaned_tweet'][train['label'] == 1]])
neg_htag = [htag for htag in negative_words.split() if htag.startswith('#')]
neg_htag = [neg_htag[i][1:] for i in range(len(neg_htag))]
neg_htag_freqcount = nltk.FreqDist(neg_htag)
neg_htag_df = pd.DataFrame({'Hashtag' : list(neg_htag_freqcount.keys()),
                            'Count' : list(neg_htag_freqcount.values())})
```

In [10]:
```
most_frequent = neg_htag_df.nlargest(columns="Count", n = 20)
plt.figure(figsize=(16,5))
ax = sns.barplot(data=most_frequent, x= "Hashtag", y = "Count")
plt.show()
```

The same process is repeated for negative tweets with label 1 as well.

From both plots, we can conclude that hashtags are very important for sentiment analysis and should not be ignored.

Finding common words in both classes of tweets using Visualization

Normal Tweets

In [11]:
```
normal_words = ' '.join([word for word in train['cleaned_tweet'][train['label'] == 0]])
wordcloud = WordCloud(width = 800, height = 500, max_font_size = 110).generate(normal_words)
print('Normal words')
plt.figure(figsize= (12,8))
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.show()
```

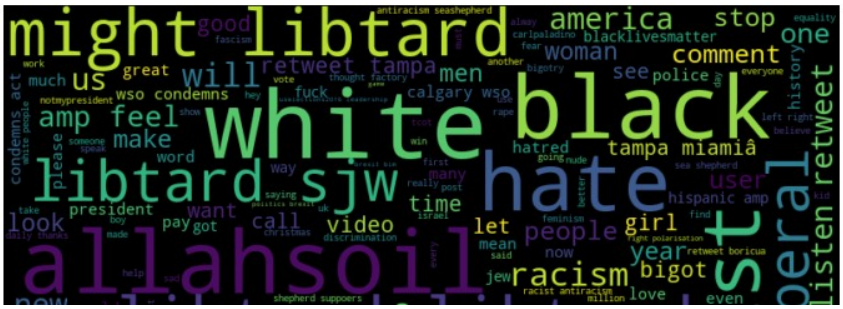
```
Normal words
```

Now a WordCloud will be generated for positive tweets from the entire data set which means data without the removal of @ and hashtags.

Racist/Sexist Tweets

In [12]:
```
negative_words = ' '.join([word for word in train['cleaned_tweet'][train['label'] == 1]])
wordcloud = WordCloud(width = 800, height = 500, max_font_size = 110).generate(negative_words)
print('Negative words')
plt.figure(figsize= (12,8))
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.show()
```

```
Negative words
```



Here also, a WordCloud will be generated for negative tweets from entire data set.

Words used like love, friend, happy are used in normal tweets whereas racist/sexist can be found in words like trump, black, politics etc.

In [13]:
```
train.sample(2)
```

Out[13]:

	id	label	tweet	cleaned_tweet
2590	2591	0	@user lovely to see the @user brighton this m...	lovely to see the brighton this morning. great...
7472	7473	0	@user guitar is life. music is life. my life ...	guitar is life. music is life. my life is note...

In [14]:
```
train.shape
```

Out[14]:
```
(31962, 4)
```

In [15]:
```
X_train, X_val, y_train, y_val = train_test_split(train['cleaned_tweet'], train['label'], random_state = 0)
X_train.shape, X_val.shape
```

Out[15]:
```
((23971,), (7991,))
```

Now here comes the concept where the data is splitted into training and testing data. Training data will be more in number as compared to testing data.

Applying Bag-of-Words

Rescale data using CountVectorizer

CountVectorizer

```
In [16]: vect = CountVectorizer().fit(X_train)
         vect

Out[16]: CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                lowercase=True, max_df=1.0, max_features=None, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)

In [17]: print('Total features =', len(vect.get_feature_names()))
         print(vect.get_feature_names()[::5000])

         Total features = 34478
         ['00', 'btg', 'encouragement', 'ifcarlingdidperfectdays', 'mona', 'rdoequipment', 'technology']

In [18]: X_train_vectorized = vect.transform(X_train)
         X_train_vectorized

Out[18]: <23971x34478 sparse matrix of type '<class 'numpy.int64'>'
                with 266363 stored elements in Compressed Sparse Row format>
```

Here is the CountVectorizer concept where the text is parsed to remove stop words, called tokenization. Then the words need to be encoded as integers or floating point values for use as input to a machine learning algorithm, called feature extraction.