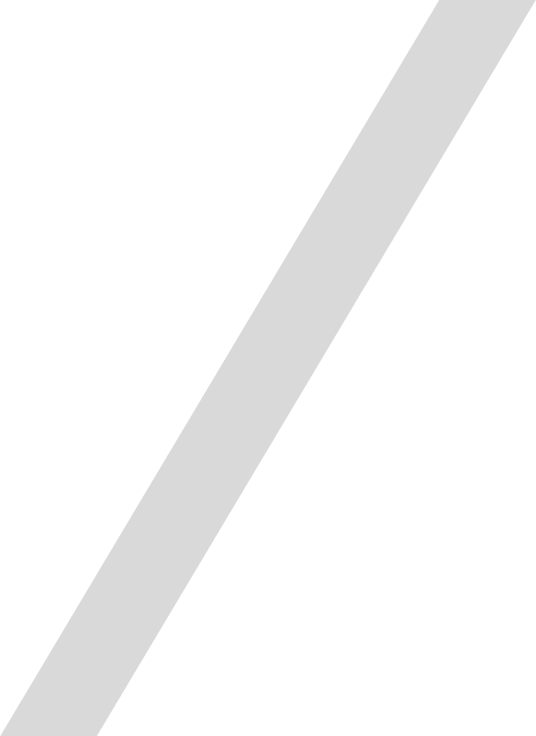
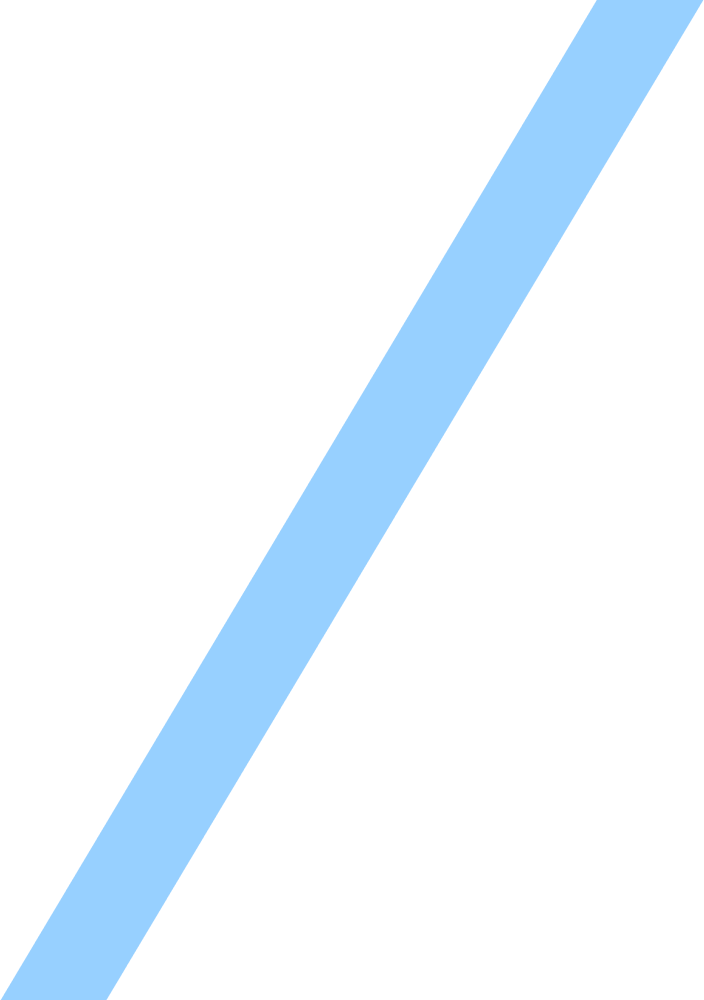
|  |
| --- |
|  |

|  |
| --- |
|  |

|  |  |
| --- | --- |
|  |  |



Twitter Sentimental Analysis

Report

# ****Data Set Description****

|  |
| --- |
|  |

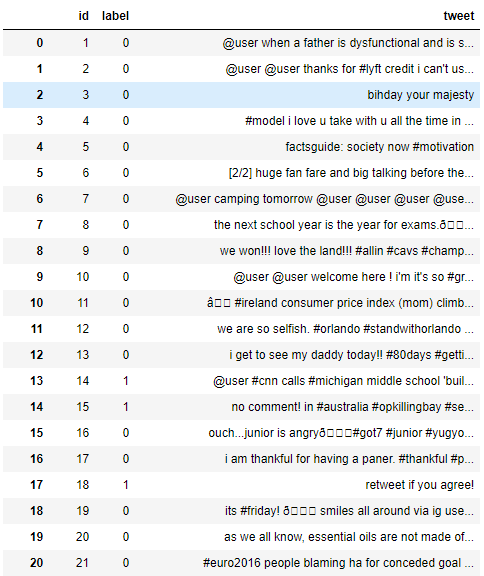
* id : The id associated with the tweets in the given dataset.
* tweets : The tweets collected from various sources and having either positive or negative sentiments associated with it.
* label : A tweet with **label ‘0’**is of**positive sentiment** while a tweet with **label ‘1’**is of**negative sentiment.**

**Importing the necessary packages**

**Reading the train.csv Pandas file**

* In the first line we read the train.csv file using Pandas.
* In the second line as a safe backup we keep a copy of our original train.csv file. **We make a copy of train data so that even if we have to make any changes in this dataset we would not lose the original dataset.**

**Overview of the training dataset**



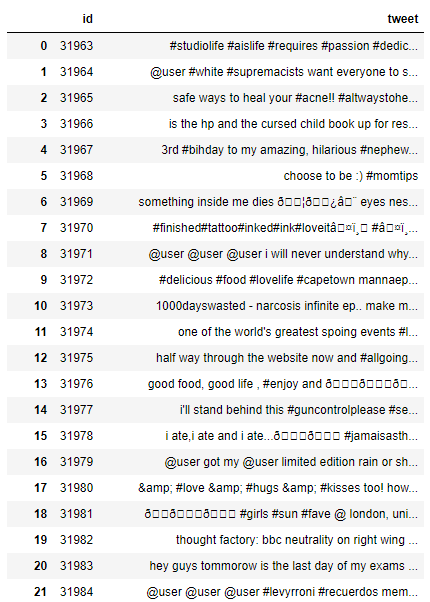


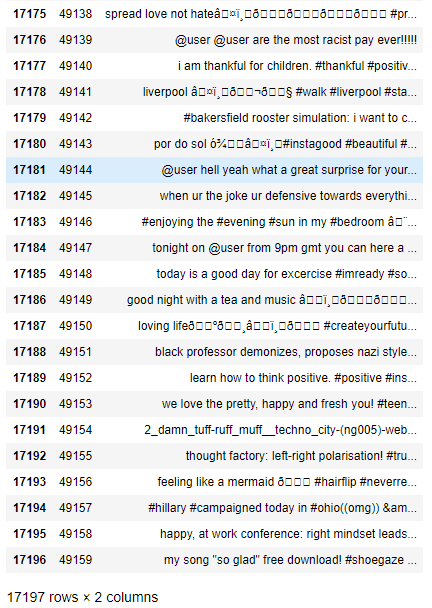
As we have 3 attributes present in our dataset and a total of 31962 labeled tweets , ‘1’ standing for tweets with negative sentiment and ‘0’ for tweets with positive sentiments.

**Reading the test.csv Pandas file**

* In the first line we read the test.csv file using Pandas.
* In the second line as a safe backup we keep a copy of our original test.csv file.**We make a copy of test data so that even if we have to make any changes in this dataset we would not lose the original dataset.**

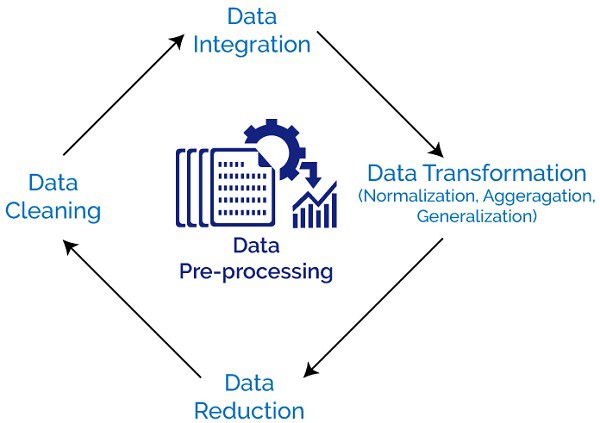
**Overview of the test dataset**





As we can see we have 2 **attributes** present here that is **‘id’** and **‘tweets’.** This is the dataset on which we are going to test our Machine Learning models so it is unlabeled.

# ****Data Pre-Processing****



Steps of data pre-processing

Let’s begin with the pre-processing of our dataset.

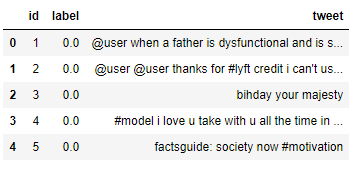
## STEP — 1 :

**Combine the train.csv and test.csv files.**

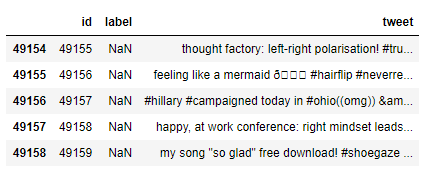
Pandas**dataframe.append()** function is used to append rows of other dataframe to the end of the given dataframe, returning a new dataframe object.

**Overview of the combined train and test dataset.**

Type **combine.head()** in the cell and you get the following result.



Again type combine.tail() in the cell and you get the following result.



Test.csv appended with the train.csv file

Columns not in the original dataframes are added as new columns and the new cells are populated with NaN value.

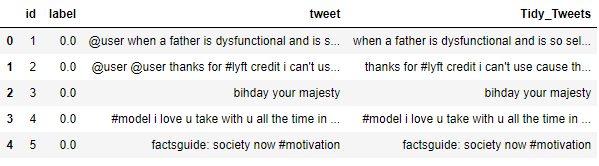
## STEP — 2

**Removing Twitter Handles(@User)**

In our analysis we can clearly see that the Twitter handles do not contribute anything significant to solve our problem. So it’s better if we remove them in our dataset.

Given below is a user-defined function to remove unwanted text patterns from the tweets. It takes two arguments, one is the original string of text and the other is the pattern of text that we want to remove from the string. The function returns the same input string but without the given pattern. We will use this function to remove the pattern ‘@user’ from all the tweets in our data.

Here NumPy Vectorization**‘np.vectorize()’**is used because it is much more faster than the conventional for loops when working on datasets of medium to large sizes.



After removing the twitter handles.

## STEP — 3

**Removing Punctuation, Numbers, and Special Characters**

Punctuation, numbers and special characters do not help much. It is better to remove them from the text just as we removed the twitter handles. Here we will replace everything except characters and hashtags with spaces.



## STEP — 4

**Removing Short Words**

We have to be a little careful here in selecting the length of the words which we want to remove. So, I have decided to remove all the words having length 3 or less. These words are also known as**Stop Words.**

For example, terms like “hmm”, “and”, “oh” are of very little use. It is better to get rid of them.

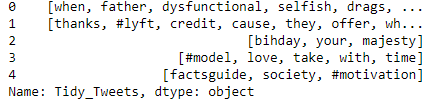


## STEP — 5

**Tokenization**

Now we will tokenize all the cleaned tweets in our dataset. Tokens are individual terms or words, and tokenization is the process of splitting a string of text into tokens.

**Here we tokenize our sentences because we will apply Stemming from the “NLTK” package in the next step.**

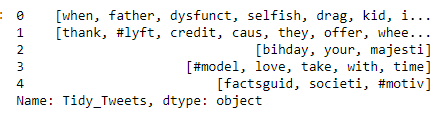


## STEP — 6

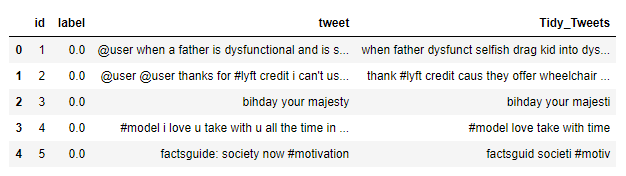
**Stemming**

Stemming is a rule-based process of stripping the suffixes (“ing”, “ly”, “es”, “s” etc) from a word.

For example — “play”, “player”, “played”, “plays” and “playing” are the different variations of the word — “play”



Now let’s stitch these tokens back together

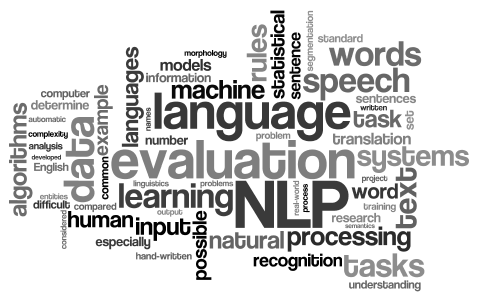


So finally these are the basic steps to follow when we have to Pre-Process a dataset containing textual data.

OK, so now we are done with our Data Pre-Processing stages.

Let’s move on to our next step that is **Data Visualisation.**

# Data Visualisation



So Data Visualisation is one of the most important steps in Machine Learning projects because it gives us an approximate idea about the dataset and what it is all about before proceeding to apply different machine learning models.

## WordCloud

One of the popular visualisation techniques is **WordCloud.**



**A WordCloud is a visualisation wherein the most frequent words appear in large size and the less frequent words appear in smaller sizes.**

So, in Python we have a package for generating **WordCloud**.

Let’s dive into the code to see how can we generate a **WordCloud.**

**Importing packages necessary for generating a WordCloud**

## Generating WordCloud for tweets with label ‘0’.

**Store all the words from the dataset which are non-racist/sexist.**

The code to generate the required **WordCloud**.

**Each line has been properly commented for a better understanding.**



Generated WordCloud

We can see most of the words are positive or neutral. With happy, smile, and love being the most frequent ones. Hence, most of the frequent words are compatible with tweets in positive sentiment.

## Generating WordCloud for tweets with label ‘1‘.

**Store all the words from the dataset which are non-racist/sexist.**

The code to generate the required **WordCloud**.

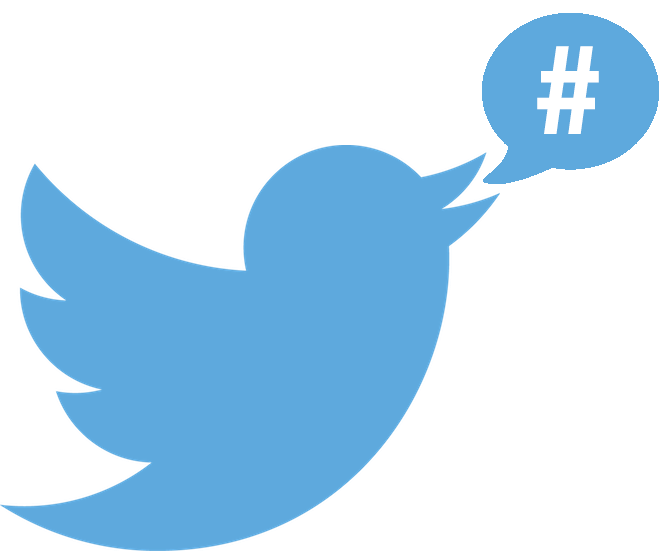
**Each line has been properly commented for a better understanding.**



Generated WordCloud

We can clearly see, most of the words have negative connotations. So, it seems we have a pretty good text data to work on.

## Understanding the impact of Hashtags on tweets sentiment



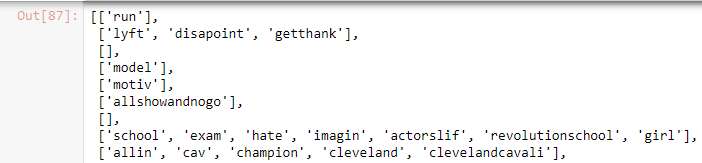
Hash-tagging on Twitter can have a major impact when it comes to your follower count by using general and non-specific hashtags. If you hashtag general words, like **#creative,** or events, like **#TIFF,** that are going on, it is more likely that your tweet will reach beyond your follower list.

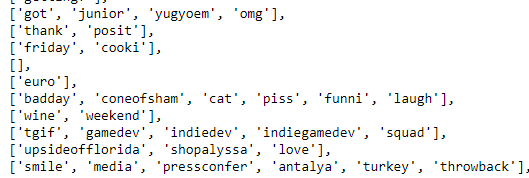
So we will look how we can extract the hashtags and see which hashtags fall into which category.

**Function to extract hashtags from tweets**

**A nested list of all the hashtags from the positive reviews from the dataset.**

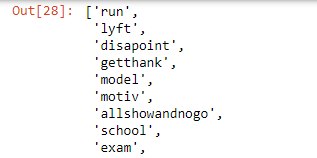
**OUTPUT :**

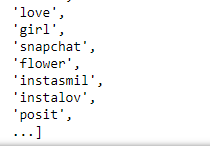




**Here we unnest the list**

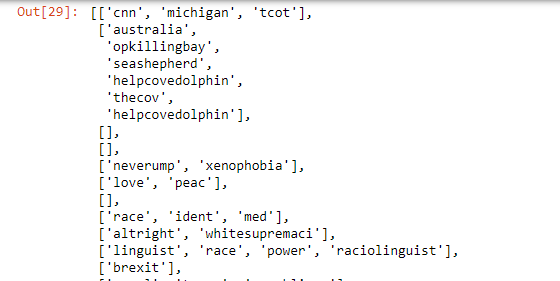
**OUTPUT :**

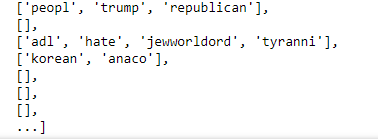




**A nested list of all the hashtags from the negative reviews from the dataset**

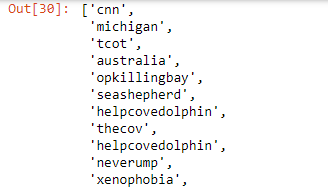
**OUTPUT :**



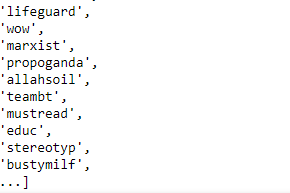


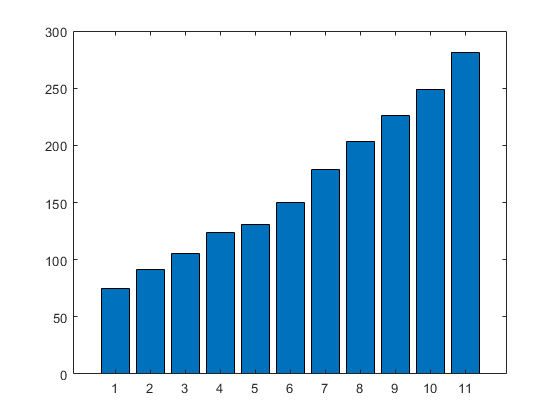
**Here we unnest the list**

**OUTPUT :**



## For Positive Tweets in the dataset



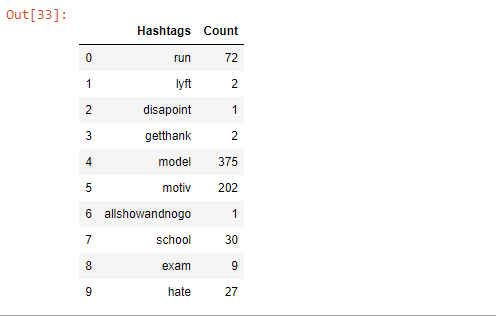


**Counting the frequency of the words having Positive Sentiment**

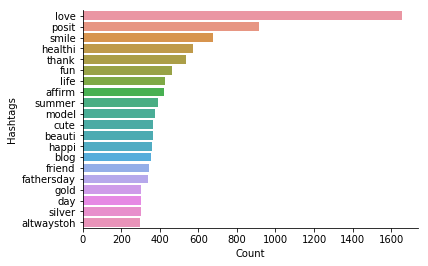
OUTPUT:

FreqDist({'love': 1654, 'posit': 917, 'smile': 676, 'healthi': 573, 'thank': 534, 'fun': 463, 'life': 425, 'affirm': 423, 'summer': 390, 'model': 375, ...})

**Creating a dataframe for the most frequently used words in hashtags**



**Plotting the barplot for the 20 most frequent words used for hashtags**



Count BarPlot

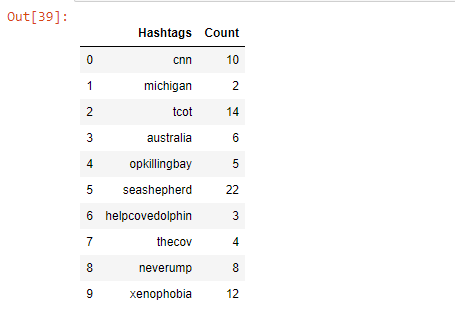
## For Negative Tweets in the dataset

**Counting the frequency of the words having Negative Sentiment**

**OUTPUT :**

FreqDist({'trump': 136, 'polit': 95, 'allahsoil': 92, 'liber': 81, 'libtard': 77, 'sjw': 75, 'retweet': 63, 'black': 46, 'miami': 46, 'hate': 37, ...})

**Creating a dataframe for the most frequently used words in hashtags**



**Plotting the barplot for the 20 most frequent words used for hashtags**

