Bahria University,

Karachi Campus



COURSE: Data Mining

TERM: SPRING 2021, CLASS: BSE- 6 (A)

PROJECT NAME

Sentiment Analysis

Dr.Salahuddin / Engr. Ramsha Mashood

Signed Remarks: Score:

***GROUP MEMBERS LIST:***

***<WITH TEAM LEAD>***

Moeez Ahmed Shah

1. Table of Contents

[**1.** **INTRODUCTION & PROBLEM** 2](#_Toc75836450)

[**2.** **PROJECT SCOPE** 3](#_Toc75836451)

[**3.** **TECHNOLOGY** 4](#_Toc75836452)

[**4.** **FUNCTIONALITIES** 5](#_Toc75836453)

[**5.** **CODE, INTERFACES AND EXPLANATION** 6](#_Toc75836454)

[**6.** **CONCLUSION** 35](#_Toc75836455)

# **INTRODUCTION & PROBLEM**

Humans work and follow their own instincts and emotions when it comes to any decision-making situation, be it any life-threatening situation or just buying any product. Emotions are complex conscious experiences and a human always goes on what these emotions and experiences suggests them to do. So in order to persuade a person, to get his/her attention, and to make him/her to partake in any activity you need to have the better understanding of their emotions, thinking, and opinions about that particular aspect, how does he/she feels about it.

Sentiment Analysis tool is a power innovative AI based project for all the businesses and brands, who are looking forward to measure and get to know of the attitudes, feelings and emotions of their customers regarding their brand/product. It can be done through social media data, survey responses or other hubs of user-generated content. Through these analyzations of customer’s feeling and emotions brands and businesses can get an inside view of consumer’s requirements, need and demands and behavior, which leads them to serve their audience with the improved products and services.

Fundamental task is to figure out the polarity of customer’s reviews and texts; whether it is positive, negative or neutral. Inducement of sentiment classification beyond polarity is even more helping in understanding the audience i.e., predicting the emotional states on deeper level such as enjoyment, sadness, anger or disgust through the words used in texts. And its advancement beyond polarity will make it a more significant tool to be used on even wider scale in future.

# **PROJECT SCOPE**

We can analyze and predict the customer’s/buyer’s feelings and thoughts about a certain product/brand through their reviews, As a result:

* Brands and organizations can work on the betterment and improvement of their goods and services
* Consumers and audience can be satisfied according to their needs.
* A better understanding of people’s mindset and their present needs.
* Better understanding of reviews in order to assess the future required improvements.

Also, we may analyze tweets and other social media activities in order to create mood-based stuffs for our consumers.

# **TECHNOLOGY**

We have used several of the technologies in here, including:

* Google Collab (For python coding)

A lot of libraries, including:

* Pandas
* Numpy
* Wordcloud
* Regex (re)
* Emoji
* Contractions
* Nltk (For tokenize, stemming, stopwords, and lemmetization)
* String
* Random
* Scikit-learn
* CountVectorizer (From sklearn)
* Logistic regression
* Accuracy score
* Confusion matrix

# **FUNCTIONALITIES**

* Given just the words in class we are going to analyze if the review provided is positive or negative.
* Judging how positive or negative some text is using natural language processing.
* We are going to use logistic regression in order to do so.
* It can perform sentiment on tweets, random sentences, complex sentences, and also reviews.

# **CODE, INTERFACES AND EXPLANATION**

**Text Normalization:**

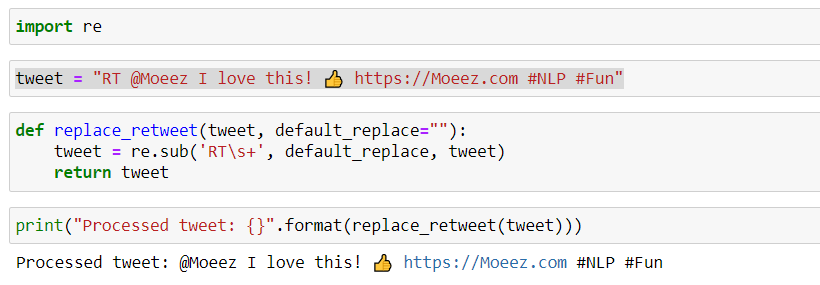
Basically, reducing randomness in a particular piece of text. Example:

* Moeez is so nice to everyone. (Positive sentiment and so = really or very)
* Moeez is sooooo nice to everyone. (Positive sentiment, however, sooooo in this case is different from so for the machine).

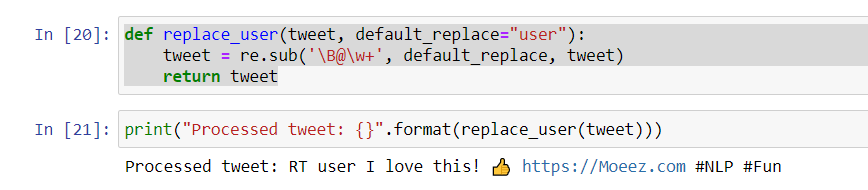
Using text mining I can reduce this to.

* Moeez so nice.
* Moeez so nice. (For a positive sentiment)

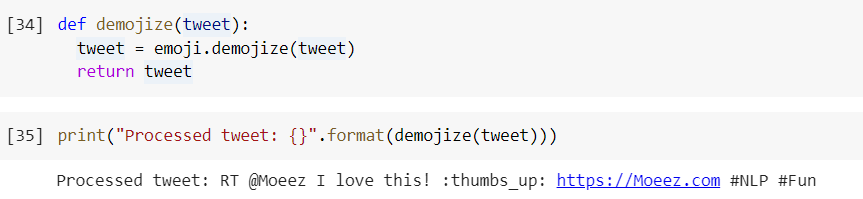
Performing normalization by creating my very own method replace\_retweet that uses regex:



Replaced RT = Retweet with default\_replace that is nothing in this case, in order to decrease the randomness.



Created very own method in order to replace user tagged name that = Moeez, in my case with a default replace of user.



Used emoji library in order to decode emotes.



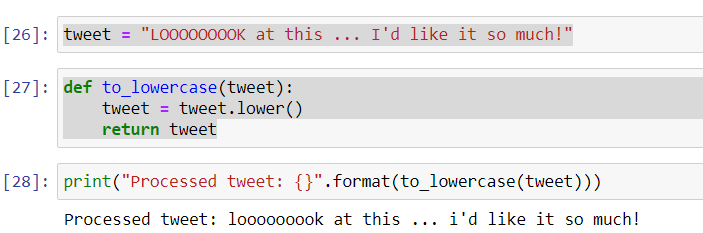
Replaced URL because it`s not important at all for performing sentiment.

Further Normalization:

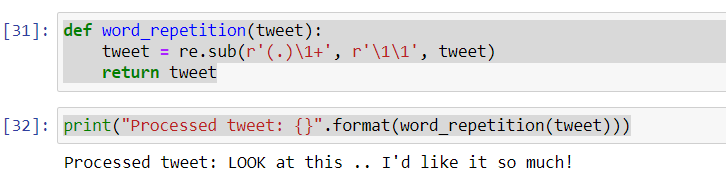
Moeez: Sooo HAPPY! Won’t stop buzzing!!!!

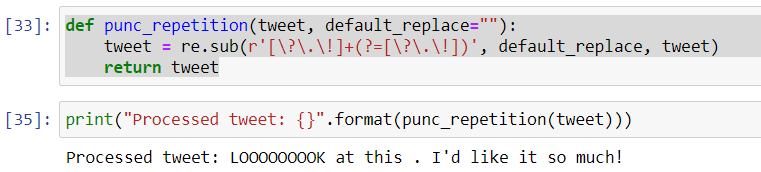
After normalization: So happy!! Will not stop buzzing!

Removed letters repetition, better if words in lowercase. Wont = will not and same with many punctuations marks repetition.

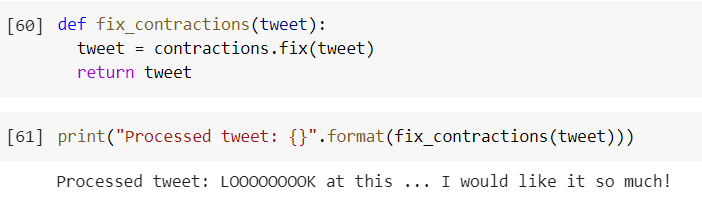


Changed sentence to lowercase.



Stopped the repetition of word.   


Stopped the repetition of punctuations.



Used contractions library in order to fix contractions.

**Further Normalization:**

Performing Tokenization;

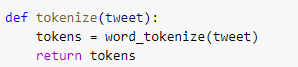
That is a way to basically separate text into smaller chunks in order for computer to learn and to understand it.

Three things here are to be taken care of: Punctuation, Stop words (like I am, and etc.), and numbers.

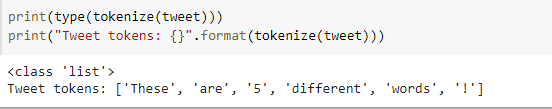
Creating a sentence, a tweet:



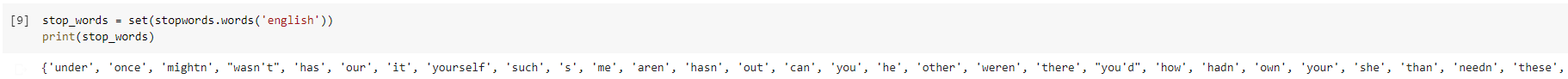
Creating a tokenize function to get tokens of the tweet:



Tokenized version:

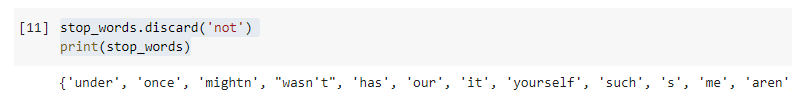


Getting a list of stop words:

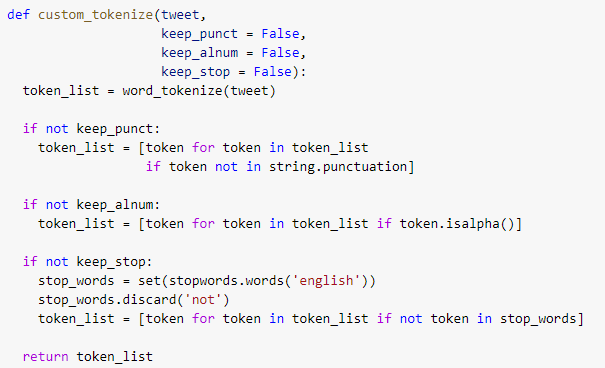


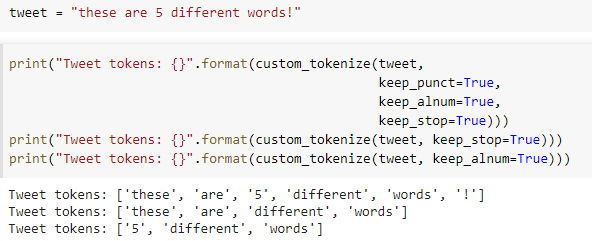
And a lot more ..

Found a stop word as not, that is surely providing a lot of sentiment, so removing it from the list:



Creating a function to handle all of those conditions:





**Stemming:**

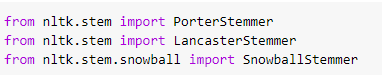
Let’s consider 3 words such as; Manager, Management, and Managing.

With the help of stemming, we can consider all these 3 words as same as they have the same starting stem. It is basically a process for reducing words to their root/original form.

Over stemming: Happens when too much of a word is chopped off; a very common example would be universal, university, and universe; which definitely carry different meanings but same stem as univers.

Under stemming: Happens when word is not chopped off enough.

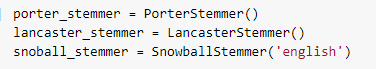
Importing libraries for stemming from NLTK:



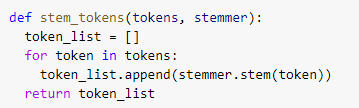
Creating tokens, that needs to be stemmed:



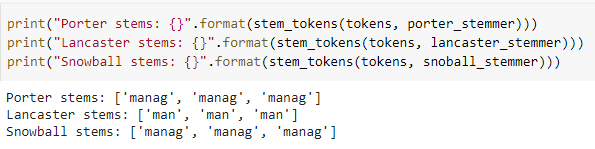
Assigning variables to stemmers:



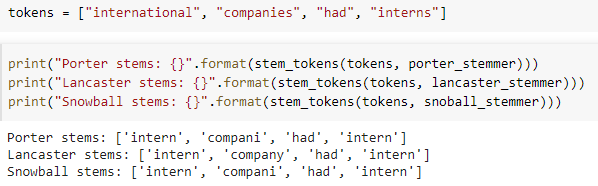
Creating a method:



Stemmed using 3 different ways:



Let’s try stemming on another example:



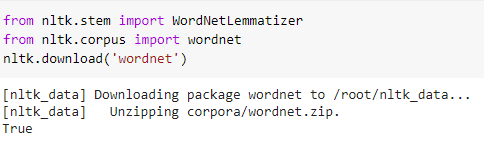
Here we see over stemming (as international and interns have different meaning but same stem), so in order to fix that we`ll use another technique:

We are using Lemmatization in order to fix this issue.

**Lemmatization:**

It serves the same purpose as the stemming does but it makes use of word context. It follows dictionary-based approach.

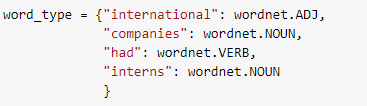
Using a wordnet library that is very important:



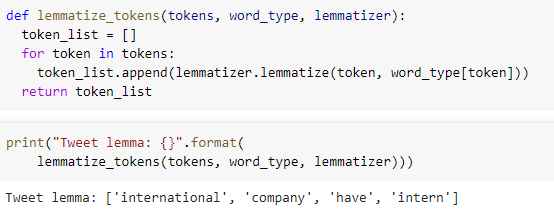
Taking the same example as before, lets see how Lemmatization performs it:



We are also including the parts of speech now:

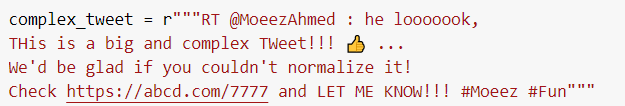


Creating a method for lemmatization:

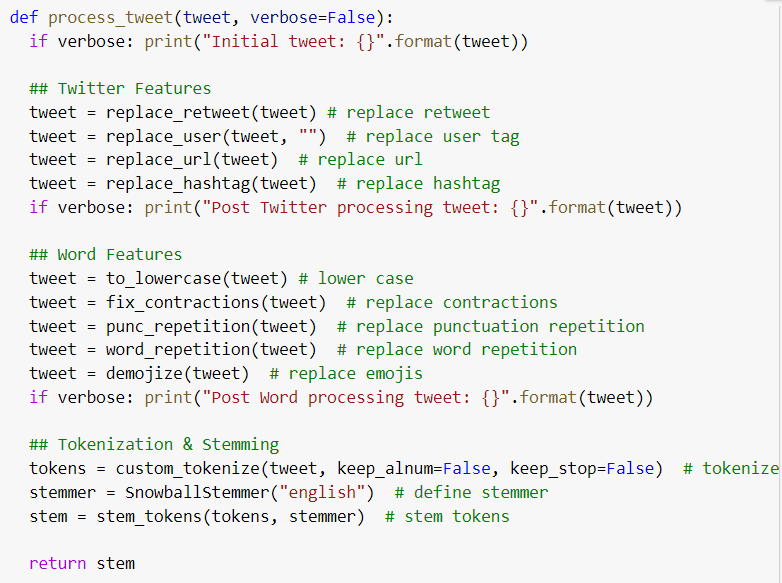


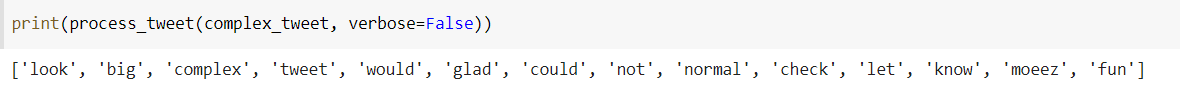
However, we are gonna stick with Stemming as its faster and better for sentiment analysis than Lemmatization.

**Now combining all of this together in order to normalize a very complex tweet.**

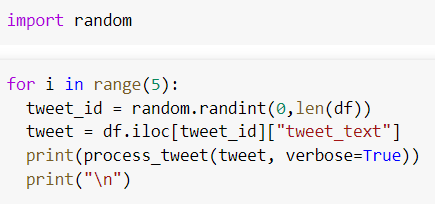


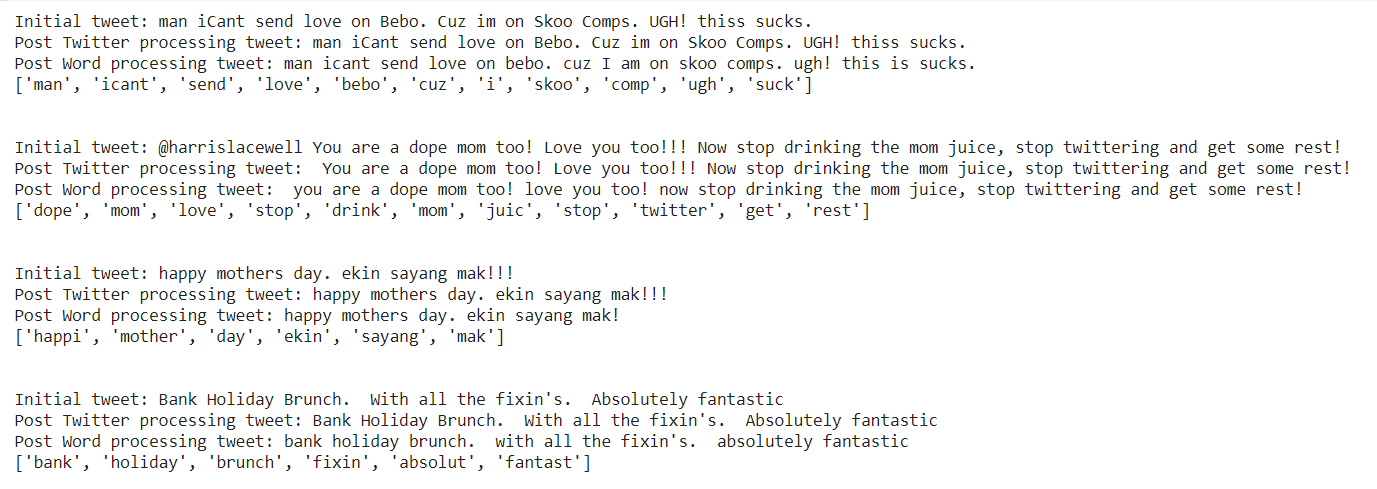
Creating a function to combine all.





Doing normalization on some random tweets.





**Text Representation:**

Converting all the text into number in order for computer and model to process it is one of the most crucial steps.

We are going to work on three methods in order for text vectorization including; Positive/negative frequencies, Bag of words, and TF-IDF.

Text representation aims at presenting text as a vector so that this vector can be used by computer and machine learning models. The better the vector and features representatives, the better the model that will be used and better it will perform.

**Positive and Negative Frequencies:**

Let’s suppose 4 tweets:

Tweet 1 (positive): I am glad I got hired.

Tweet 2 (positive): This is great.

Tweet 3 (negative): This is bad.

Tweet 4 (negative): I am sad I got fired.

The tweets look almost similar and identical to each other; however, their sentiment is different. We can create a frequency table in order to represent it, and count how many times each word appears in positive and negative sentiment.



Now consider another new tweet: I got fired, this is bad.

In order to represent it, we can create a vector.

X = [ pos freq, neg freq ]

X = [ Sum of freq(w, 1), Sum of freq(w, 0) ]



5 positive and 7 negatives, so it becomes.

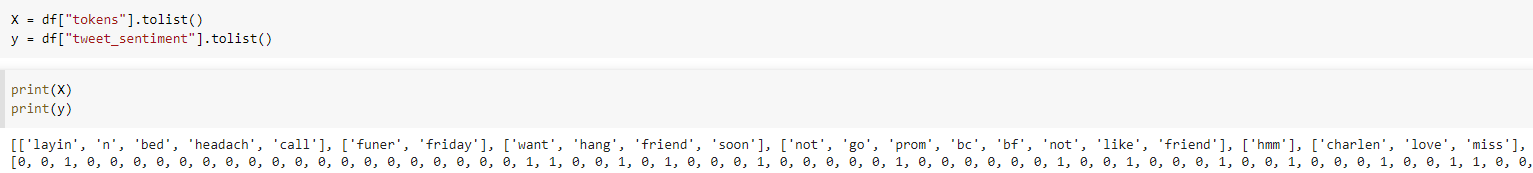
X = [5,7]

Through coding:

Introducing a new column as tokens and tweet\_sentiment (positive = 1 and negative = 0):

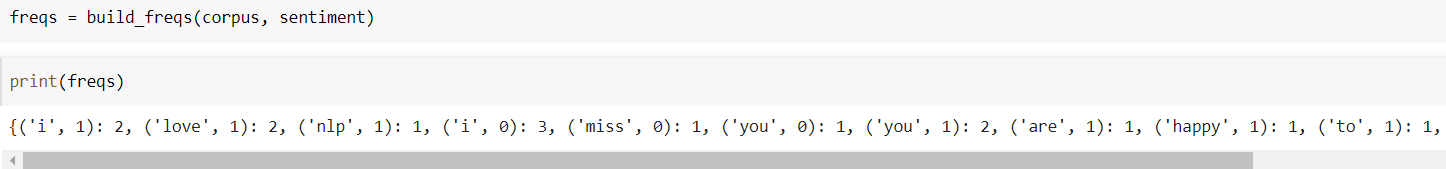


Converting them to list:

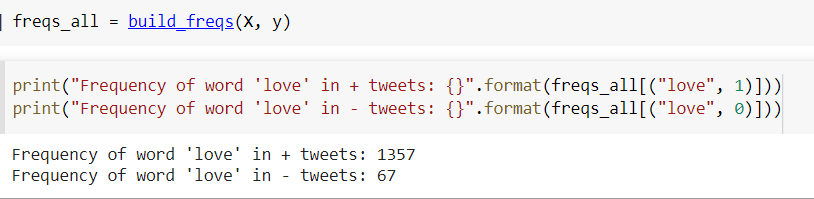


Creating a build\_freqs function used to build a dictionnary with the word and sentiment as index and the count of occurence as value:

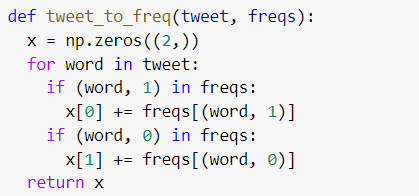


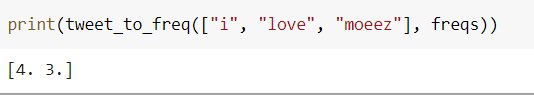


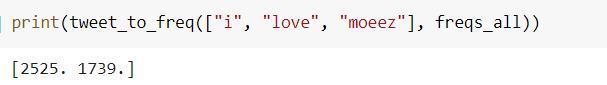
Checking sentiment of love:



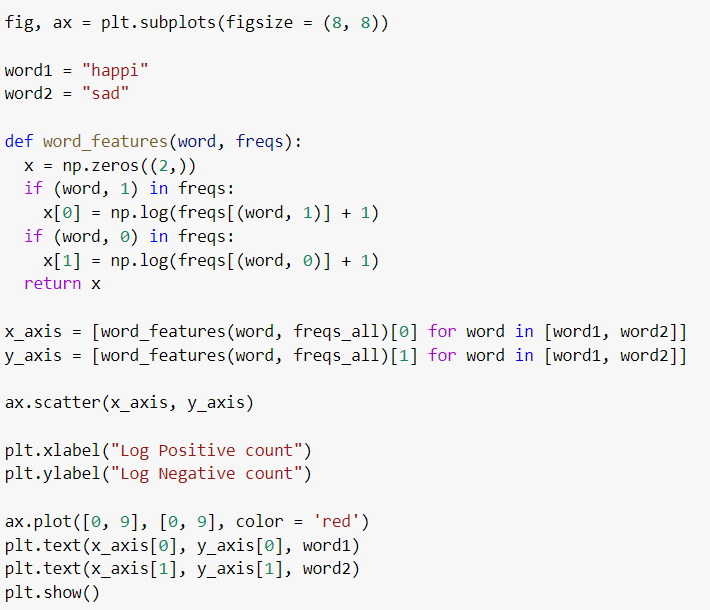
Creating a tweet\_to\_freqs function used to convert tweets to a 2-d array by using the frequency dictionary.

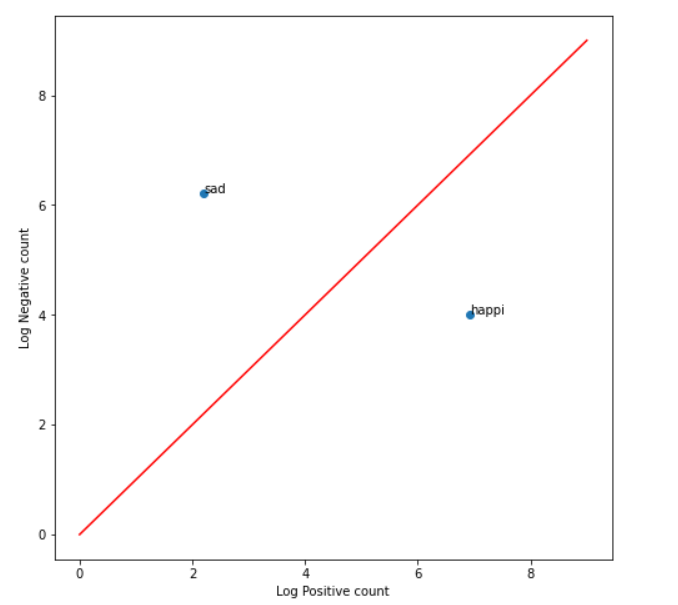






Plotting word vectors in a chart to see where they locate:





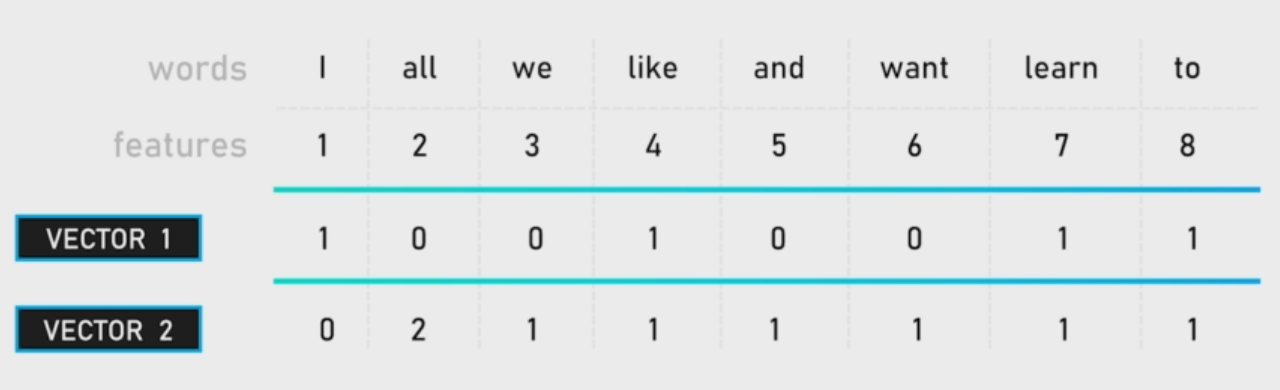
**Bag of Words Representation:**

Used to extract features from a document. First it puts all the text inside a bag and then assigns an array in order to see which word appears in the tweet and how many times. An example;

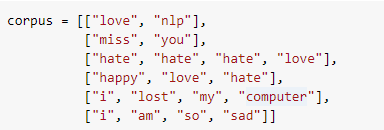
Tweet 1: I like to learn.

Tweet 2: We all like and all want to learn.

Creating the table:

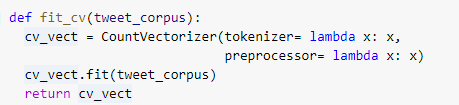


Creating corpus:



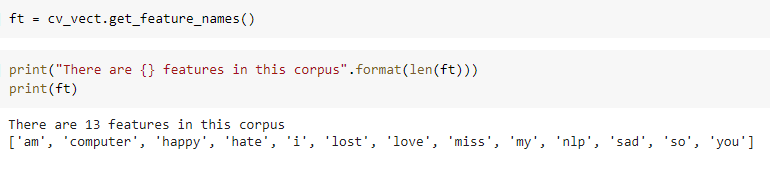


Creating a fit\_cv function used to build the Bag-of-Words vectorizer with the corpus:



Using the fit\_cv function to fit the vectorizer on the corpus:



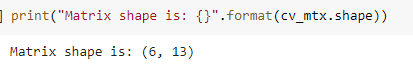


And here we have the matrix of 13 features.

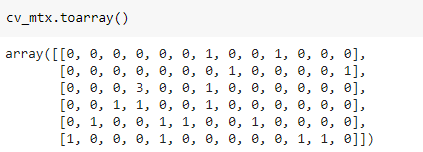
Converting the corpus to a matrix by using the vectorize.



Printing the shape of the matrix:



It looks like this after converting it into an array:



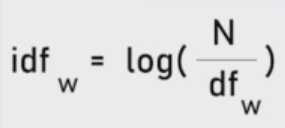
**TF-IDF:**

Term Frequency – Inverse Document Frequency.

Is actually one of the most widely used technique to represent text.

Tf is computed by counting number of times a word is repeated in document divided by the total number of words in the document.

IDF is computer by a formula of:



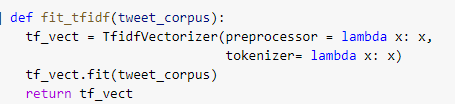
Corpus of tweet tokens:



Using library:

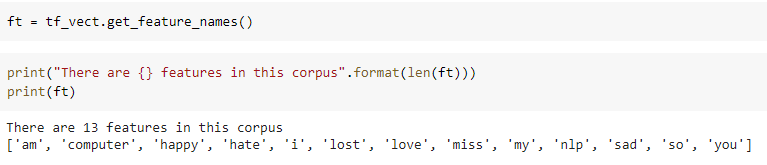


Creating a fit\_tfidf function used to build the TF-IDF vectorizer with the corpus.

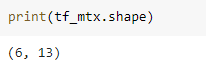


Using the fit\_cv function to fit the vectorizer on the corpus, and transform the corpus.

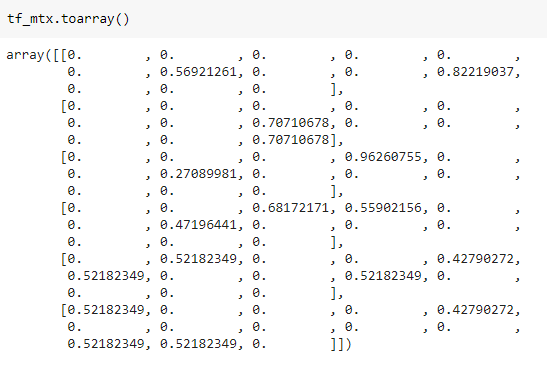




Printing the shape of the matrix:



Converting it into array and printing:

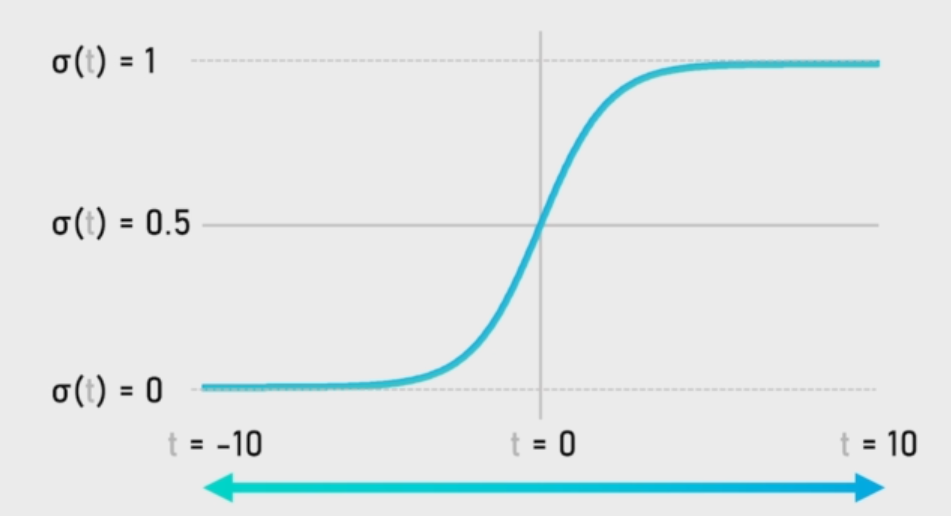


**Sentiment Analysis:**

First of all we understood the nature of the tweet, after that we performed tokenization, stemming and lemmatization in order to clean up our tweets. After that we performed vectorization in order to convert the list of words into an array of numbers and realized why is it one of the most crucial step for our sentiment to work correctly and for computers. Now, it`s time to construct the sentiment model of our application.

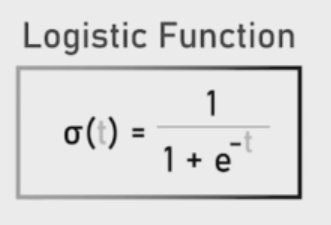
**Logistic Regression:**

The statistical model we will use to predict sentiment is called Logistic regression. Regression is the way to estimate the relationship between a dependent variable and one or more explanatory variables. A simple example can be the relationship between Moeez`s weight which is a dependent variable and a 50/50 mix of how much I eat and how much exercise/sports I do (that are the explanatory variables). Logistic regression however, is a particular form of regression because it can be used to model a binary dependent variable. Just like a positive or a negative sentiment for instance. Let me demonstrate it using a graph:



On vertical axis is the prediction for the dependent variable.

Logistic Function:



1 meaning the event happened and 0 meaning it didn’t for the value of alpha. Results above 50% can be considered as positive and results below 50% can be considered as negative.

Suppose 2 tweets:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tweet | Vectors | t | Alpha | Sentiment |
| Tweet 1 | [1450, 2214] | -2.97 | 0.0487 | Negative |
| Tweet 2 | [3561, 1930] | 4.77 | 0.9915 | Positive |

Alpha value of tweet 2 is higher than 0.5, hence positive. However, tweet 1 alpha value is less than 0.5 (that is 0.0487) hence, it`s negative sentiment.

The logistic function needs a single input to output the probability.

**Estimation of Parameters:**

We need to find beta (learned parameters) in order to feed it to our logistic regression algorithm in order to find the accurate sentiment.

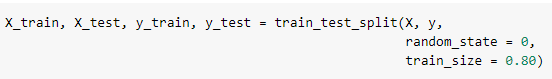
**Training/Testing split phase:**

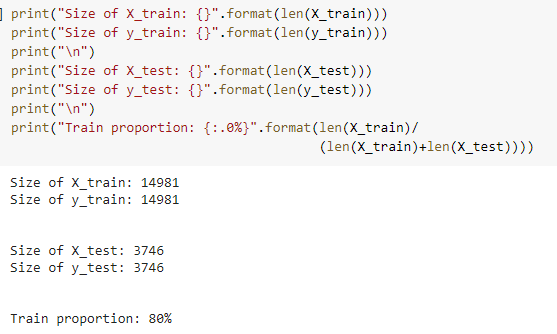
Dataset is split into two distinct sets, a train set used during the training phase (usually representing around 70-80% of the dataset), and a test set used during the testing phase, usually representing the remaining 20 percent. But why do we need these two stages with different datasets? It`s because the training phase review different examples present in the dataset and try to fit vector features as well as possible with tweet sentiments by using beta coefficients. The objective of training is to actually find the optimal set of weights and biases, so the model produces a lower loss across all training examples. Testing data is used to measure whether the model performs. The goal of the model is to be used on new data, data that has never been encountered, if the model was trained on all the data, there will be no way to easily verify the effectiveness. However, by keeping some data on the side, it becomes possible to use the train model parameters to make the prediction on this data. More interestingly, it is possible to check and evaluate the predictions made by the model on the test set because the true predictions for the test set is already actually known.

Importing train\_test\_split:

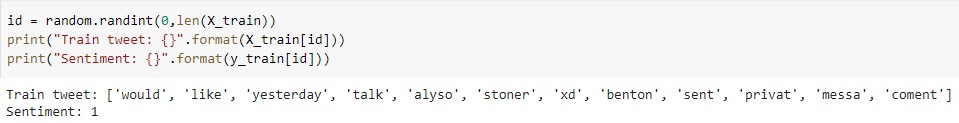


Using it to train the model, keeping train size = 80%





Printing random tweets just to verify that everything goes as expected:

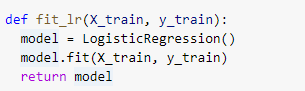


**Importing Logistic Regression model from Scikit-Learn:**



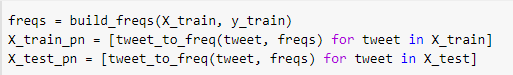
Fitting the model means finding the optimal set of Beta parameters, so the relationship between X\_train and y\_train is as accurately described as possible.

Creating fit\_lr method used to fit the logistic regression model on y and X training data.

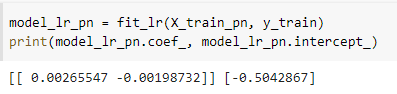


Pos/Nev Frequency:

Building dictionary based on the training set and not the entire set-in order to prevent the data leakage. As the model needs to be trained on the training data only and is not supposed to access any of the test data.

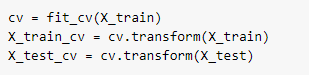


Beta values that we attained are almost really good, means sentiment going to depend on the value of positive and negative vectorization:



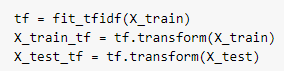
**Bag of words vectorization:**

Using fit\_cv function on training data in order to build bag of words vectorizer.



Now, fitting the logistic regression model on training data by using fit\_lr function. However, there are as many values of Beta as outcomes in the matrix (that is a pretty big number in the current case).

**TF-IDF vectorization:**





**Accuracy Measure:**

Which of the model among the 3 is the best? Going to use performance metrics.

Confusion matrix is the best approach to do so.

**Accuracy =** (TP+TN) / (TP+TN+FN+FP)

Importing accuracy\_score and confusion\_matrix from sklearn.metrics.

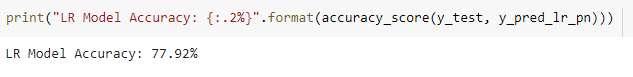


**Pos/Neg Frequencies:**

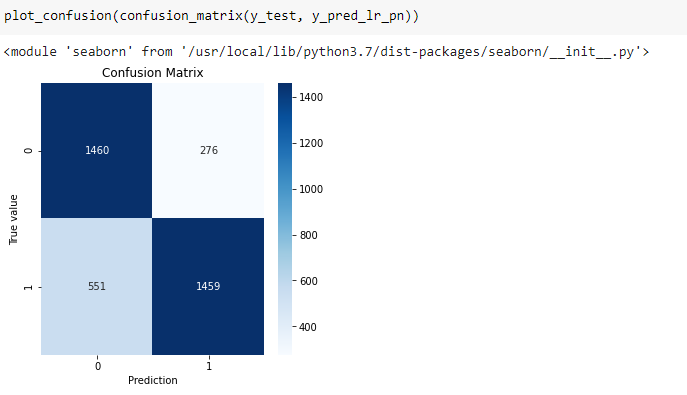
Using X\_test data to predict using positive/negative frequencies.



Printing the accuracy score:



Displaying confusion matrix:

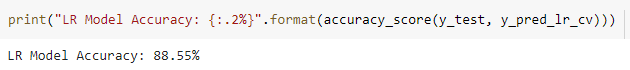


**Bag of words frequencies:**

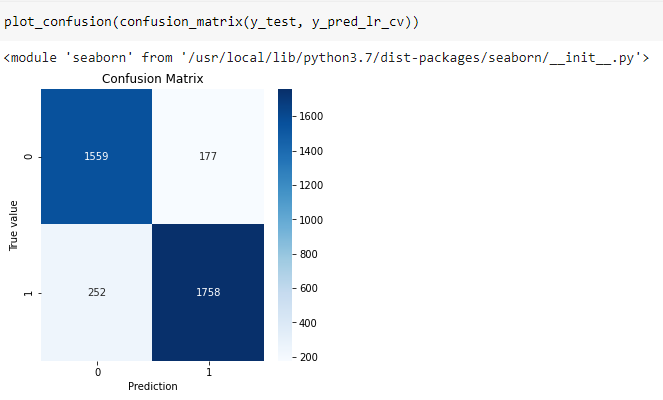
Using the X\_test data to predict using bag of words:



Finding accuracy:



Displaying Confusion Matrix:

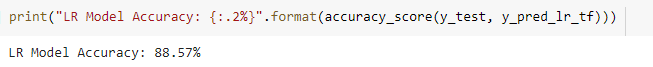


**TF-IDF Frequencies:**

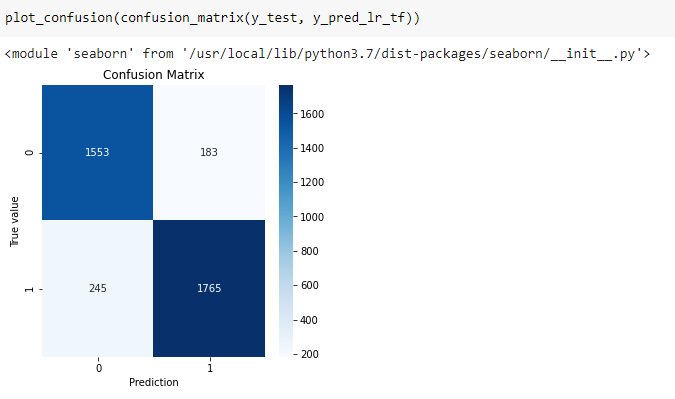
Using the X\_test data to predict using TF-IDF:



Printing Accuracy:



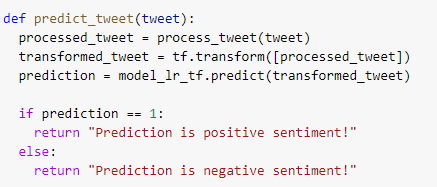
Displaying Confusion matrix:



**Finalizing and predicting my own Tweets:**

Gone through a lot of steps (Cleaned the dataset, how to represent it so it could be used to train the machine learning model) in order to reach this final step in order to predict the sentiment of our provided custom and unknown tweets.

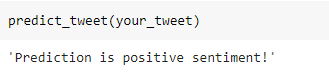
Creating a function used to pre-process, transform and predict tweet`s sentiment:



My custom tweet:



Predicting sentiment:



# **CONCLUSION**

Sentiment analysis or opinion mining is a field of study that analyzes people’s sentiments, attitudes, or emotions towards certain entities. The future of sentiment analysis is basically to dig deeper, much past the number of comments, shares, and like. It aims to reach and truly understand the importance of social media interactions and also what they have to talk about the consumers behind the screens.