

```
# This Python 3 environment comes with many helpful analytics  
libraries installed# It is defined by the kaggle/python Dock  
er image: https://github.com/kaggle/docker-python# For exampl  
e, here's several helpful packages to load
```

```
import numpy as np # linear algebraimport pandas as pd # data  
processing, CSV file I/O (e.g. pd.read_csv)
```

```
# Input data files are available in the read-only "../input/"  
directory# For example, running this (by clicking run or pre  
ssing Shift+Enter) will list all files under the input direct  
ory
```

```
import osfor dirname, _, filenames in os.walk('/kaggle/input  
'):
```

```
    for filename in filenames:
```

```
        print(os.path.join(dirname, filename))
```

```
# You can write up to 20GB to the current directory (/kaggle/  
working/) that gets preserved as output when you create a ver  
sion using "Save & Run All" # You can also write temporary fi  
les to /kaggle/temp/, but they won't be saved outside of the  
current session
```

```
/kaggle/input/train-sentiment-analysiscsv/train.sentiment analysis.csv
```

```
/kaggle/input/test-1-sentimentanalysis/test sentiment analysis.csv
```

## Twitter Sentiment Analysis

This kernel is the solution for the challenge launched by School of AI - Algiers, which consist of building a system that can classify tweets as Sad or Happy.

### 1. Solution

We will start by reading some tweets so we can understand our data better. We will then try to transform our tweets into something usable by different ML models, where we are going to choose the more efficient. We will finally fine tune our model and then test it to see its efficiency on new data.

## Update

After getting some comments on the School of AI - Algiers group, especially from Belkacem, I updated the following:

I used lemmatization instead of steaming I also noticed that I was mistaken when I stopped the max\_features parameter at 20000 while doing GridSearch, I should have tested a bigger one, because

if it stopped at 20000 (which is the max), it may get better using a bigger one. I just added None (no limit).

## Content

Loading the data

Visualize the tweets

Emoticons

Most used words

Stop words

Stemming

Prepare the data

Bag of Words

Building the pipeline

Select a model

Fine tune the model

Testing the model

Test your tweet

## Load the data

In [2]:

```
import numpy as npimport pandas as pd

# This is for making some large tweets to be displayedpd.options.display.max_colwidth = 100

# I got some encoding issue, I didn't knew which one to use !
# This post suggested an encoding that worked!# https://stackoverflow.com/questions/19699367/unicodedecodeerror-utf-8-codec-cant-decode-bytetrain_data = pd.read_csv("/kaggle/input/train-sentiment-analysiscsv/train.sentiment analysis.csv")
```

In [3]:

```
train_data
```

Out[3]:

	ItemID	Sentiment	SentimentText
0	1	0	is so sad for my APL friend.....
1	2	0	I missed the New Moon trailer...
2	3	1	omg its already 7:30 :0

	ItemID	Sentiment	SentimentText
3	4	0	.. Omgaga. Im sooo im gunna CRY. I've been at this dentist since 11.. I was suposed 2...
4	5	0	i think mi bf is cheating on me!!! T_T
...	...	...	...
99984	99996	0	@Cupcake seems like a repeating problem hope you're able to find something.
99985	99997	1	@cupcake__ arrrr we both replied to each other over different tweets at the same time , i'll se...
99986	99998	0	@CuPcAkE_2120 ya i thought so
99987	99999	1	@Cupcake_Dollie Yes. Yes. I'm glad you had more fun with me.
99988	100000	1	@cupcake_kayla haha yes you do

99989 rows x 3 columns

## Visualize the tweets

*From the tweets above, we can already make some remarks about the data:*

*We can see that there is some garbage like '&amp;', '&lt;' (which are basically used in HTML) that aren't gonna help us in our classification*

*In twitter, people mention their friends with tags like @username, there is a lot of them in our data. I was discussing with a friend about the usefulness of tags in our classification, for him, people tend to mention more friends when they are happy, but I think that people may mention people because they made bad things. When we face this kind of uncertainty, it's better to try the different options and evaluate which will do well, this is what we are gonna do.*

In [4]:

*# We will now take a look at random tweets# to gain more insights*

```
rand_indexes = np.random.randint(1,len(train_data),50).tolist()
train_data["SentimentText"][rand_indexes]
```

Out[4]:

```
56509                                     @BDEugenio why would you do such a thing
and leave me

62551    @blakeleray There's no way You can't see what messages she reads.On her page you jus
t see what ...

10702                                     &quot;When i love you a little less than be
fore...&quot;;

65372    @brenda_song HI Brenda! How are you? I start exams tomorow. btw ur Amazing! L
ove ya!  Xxxxxx

30032                                     @adamsmith I saw a lobster-colored woman at church this morning and tho
ught of you.

46789                                     @AprilRainer aaaaaahhhhhhhh is that new tonight? my tv guide
says its not
```

51235 @ashleyann2009 i loveeeers you

34862 @alagu There is hi chnce dat TN nuclear facility may be visible now .. As GooG update d images - ...

90181 @babyguuuurl awww reunited and it f eel soo good!

74312 @caramelflavored hahah the p owers of dana

82923 @caseyahf Maybe? No? K. I'll just sit here alone. And hum to myself

52396 @avidbookreader Cronenberg directed The Thing? No, he didn't. John C arpernter did.

17436 @\_\_sugar aww what's wrong?

32217 @Ali\_Sweeney Thanks for sharing your pics from the photo shoot. Loo ked like fun!

85588 @chrisfreeman LOL I'm following u now too! Night new friiiend!

67825 @BoobooBest I can 't find it...

33739 @airbagged im so broke!!! i would oth erwise though

4462 # andyhurleyday

32677 @agoldenberg it was on my micro SD card

54996 @bannersrus Good to know! Clouds and expected rain here Enjoy your beauty of a day tho!!

12267 \*in giddy voice\* OH MY GOSH!!! The cute severe thunderstorm is headed this wa y!!!! BOOM!!!

70629 @Butterfly\_Sing awww poor kya... that sucks do ya plan to get anot her pet later?

6154 #ipv6summit - yes Windows OS has &quot;power shell&quot;!! weeeee - how useless

96497 @crsimp01 Always fun to just get out. Go lf or no golf

28851 @AJMIX969 sorry to hear about ur date, people just suck sometimes.

56590 @BeachNYC09 ye aaaah you do!

80481 @chaoskittypie \*flys to Scotland, goes to lowri's hous e, kills fly\*

45356 @anotherojplease I think Kelz, as well as myself, were doomed from the very beginning, regardles...

32617 @AgingBackwards lol! thanks.. I laughe d! needed that

77399 @C3Mike Its out next week! Codemasters really isn't doing a good job of getting the word out. I'...

2598 i feel like such a loser.  
i hate hills

4478 #asot400 he said video should be up... but it is not. Lovin' the audio, tho!

46550 @aplusk Ashton can you explain to me what R and B means please as I am in the I'm

77978 @CAGEosaurus Hey,  
how are you?

61501 @billmelater Lucky bastard! Tell  
us how it is

90649 @backstreetboys We miss seeing the Panic dance in  
Canada #BSB

55718 @babychoops denver to laguna beach to the hills, at least shes moving in the right direction

92480 @chrisivens we had that yesterday too! some microsoft update got applied automatically and it w...

23843 @abhishek Yes thank God for that - It's Raining and hopefully will continue to do so throughout ...

55262 @BarkingDogShoes I hear ya! I made my own... Wouldn't be able to drag myself out of the house wi...

23988 @1Superstar We're going to be good friends...Love ya!

31277 @aeroplanes IKR? It  
was terrible.

98907 @CoverFX Hello!  
How are you?

73772 @Candypants2 not quitting taking a break that's, personal crap, things have been shaky on my end

29909 @adamlawlz its how u love me now  
by hey monday

80442 @CareerDesign you're so welcome #PersonalBranding  
#FollowFriday

24895 @33girl hehehe Wasn't sure if that would offend or not, purely fun from my perspective! MEMBER!

31320 @AlexanderNixon always friendly ;- ) Late answer but no more computer My mac died... Trying to ...

57752 @AMYburg  
h No problem.

91131 @Chloii14 did Katie tell you about that amazing new show  
how on living?

Name: SentimentText, dtype: object

## Note

you will not have the same results at each execution because of the randomization.  
For me, after some execution, I noticed this:

There is tweets with a url (like tweet 35546): we must think about a way to handle URLs, I thought about deleting them because a domain name or the protocol used will not make someone happy or sad unless the domain name is 'food.com'.

The use of hashtags: we should keep only the words without '#' so words like python and the hashtag '#python' can be seen as the same word, and of course they are. Words like 'as', 'to' and 'so' should be deleted, because they only serve as a way to link phrases and words

## Emoticons

The internet language includes so many emoticons, people also tend to create their own, so we will first analyze the emoticons included in our dataset, try to classify them as happy and said, and make sure that our model know about them.

In [5]:

```
# We are gonna find what emoticons are used in our dataset
import re
retweets_text = train_data.SentimentText.str.cat()
emos = set(re.findall(r"([xX:;][-']?.)", tweets_text))
emos_count = []
for emo in emos:
```

```
    emos_count.append((tweets_text.count(emo), emo))
sorted(emos_count, reverse=True)
```

Out[5]:

```
[(3281, ':/'),
 (2874, 'x '),
 (2626, ': '),
 (1339, 'x@'),
 (1214, 'xx'),
 (1162, 'xa'),
 (984, ';3'),
 (887, 'xp'),
 (842, 'xo'),
 (713, ';)'),
 (483, 'xe'),
 (431, ';I'),
 (353, ';.'),
 (254, 'xD'),
 (251, 'x.'),
```

(245, '::~'),  
(234, 'X '),  
(217, ';t'),  
(209, ';s'),  
(185, ':0'),  
(176, ':3'),  
(166, ';D'),  
(159, ":'"),  
(157, 'XD'),  
(146, 'x3'),  
(142, ':p'),  
(126, ":'("),  
(118, ':@'),  
(117, 'xh'),  
(117, ':S'),  
(109, 'xm'),  
(104, ';p'),  
(104, ';-'),  
(92, ':|'),  
(91, 'x,'),  
(89, ';P'),  
(76, 'xd'),  
(75, ';o'),  
(75, ';d'),  
(71, ':o'),  
(65, 'XX'),  
(63, ':L'),  
(59, 'Xx'),  
(59, ':1'),  
(58, ':]'),  
(57, ':s'),  
(56, ':0'),  
(54, 'X0'),

(44, ';;;'),  
(43, ';( '),  
(38, ':-D'),  
(37, 'xk'),  
(36, 'XT'),  
(35, 'x?'),  
(35, 'x)'),  
(34, 'x2'),  
(33, ';/'),  
(32, 'x: '),  
(32, ':\ \'),  
(31, 'x- '),  
(27, 'Xo'),  
(27, 'XP'),  
(27, ':-/'),  
(26, ':-P'),  
(25, ':\*'),  
(23, 'xX'),  
(22, ":' )"),  
(17, 'xP'),  
(16, ':[ '),  
(16, ':-p'),  
(14, 'x]'),  
(14, 'XM'),  
(13, ':-O'),  
(12, 'x( '),  
(12, 'X1'),  
(12, ':x'),  
(11, 'XS'),  
(11, ':1'),  
(10, 'x\*'),  
(10, 'X. '),  
(10, ':b'),



(10, 'T'),  
(9, ''],  
(9, 'I'),  
(8, 'C'),  
(7, ';-('),  
(7, ':-|'),  
(6, 'X,'),  
(6, ':-o'),  
(6, ':-\\'),  
(6, ':-\*'),  
(6, ':-\$'),  
(5, 'XL'),  
(5, ':-d'),  
(5, ':-X'),  
(5, ':-H'),  
(5, ':-?'),  
(5, ':-S'),  
(4, ':-D'),  
(3, ':-Z'),  
(3, ':-E'),  
(3, ':-s'),  
(3, ':-['),  
(3, ':-X'),  
(2, 'X5'),  
(2, 'X-('),  
(2, "X's"),  
(2, ';-;'),  
(2, ':-}'),  
(2, ':-D'),  
(2, ':-;'),  
(2, ":-D"),  
(1, 'x|'),  
(1, "x'd"),

```
(1, "x'D"),
(1, ';-|'),
(1, ';-/'),
(1, ':('),
(1, ':-x'),
(1, ':-h'),
(1, ':-|'),
(1, ':-W'),
(1, ':-$'),
(1, ':('),
(1, ":'[""])
```

We should by now know which emoticons are used (and its frequency) to build two regex, one for the happy ones and another for the sad ones. We will then use them in the preprocessing process to mark them as using happy emoticons or sad ones.

In [6]:

```
HAPPY_EMO = r" ([xX;:]-?[dD])|:-?[\\]|[:;][pP]" SAD_EMO = r"
(:'?[/|\\()]" print("Happy emoticons:", set(re.findall(HAPPY_
EMO, tweets_text))) print("Sad emoticons:", set(re.findall(SAD
_EMO, tweets_text)))
```

```
Happy emoticons: {'xD', ':-D', 'xd', ';-)', 'p', 'p', 'P', 'd', 'D', 'D', 'XD', 'x)', ';-D', ';;)', 'd'}
```

```
Sad emoticons: {'|', ':(', ":'(", ':/'}
```

## Most used words

What we are going to do next is to define a function that will show us top words, so we may fix things before running our learning algorithm. This function takes as input a text and output words sorted according to their frequency, starting with the most used word.

In [7]:

```
nlTK.download('punkt')
```

```
-----NameError
Traceback (most recent call last):<ipython-input-7-9533fb74b295> in <mod
ule>----> 1 nlTK.download('punkt')
```

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

In [8]:

```
import nlTKfrom nlTK.tokenize import word_tokenize
```

```
# Uncomment this line if you haven't downloaded punkt before#  
or just run it as it is and uncomment it if you got an error.  
#nltk.download('punkt')def most_used_words(text):
```

```
    tokens = word_tokenize(text)
```

```
    frequency_dist = nltk.FreqDist(tokens)
```

```
    print("There is %d different words" % len(set(tokens)))
```

```
    return sorted(frequency_dist, key=frequency_dist.__getitem__  
__, reverse=True)
```

In [9]:

```
most_used_words(train_data.SentimentText.str.cat())[ :100]
```

There is 133864 different words

Out[9]:

```
['@',  
'!',  
'.',  
'I',  
,',',  
'to',  
'the',  
'you',  
'?',  
'a',  
'it',  
'i',  
'...',  
,',',  
'and',  
'&',  
'my',  
'for',  
'is',  
'that',  
's',
```

"n't",  
'in',  
'of',  
'me',  
'have',  
'on',  
'quot',  
" 'm",  
'so',  
':',  
'but',  
'#',  
'do',  
'was',  
'be',  
'not',  
'your',  
'are',  
'just',  
'with',  
'like',  
'-',  
'at',  
'too',  
'get',  
'good',  
'u',  
'up',  
'know',  
'all',  
'this',  
'now',  
'no',

'we',  
'out',  
)',  
'love',  
'can',  
'(',  
'what',  
'one',  
'will',  
'lol',  
'go',  
'about',  
'did',  
" 'll",  
'got',  
'amp',  
'there',  
'day',  
'http',  
'see',  
" 're",  
'if',  
'time',  
'they',  
'think',  
'as',  
'when',  
'from',  
'You',  
'It',  
'going',  
'really',  
'am',

```
'work',  
'well',  
'had',  
'would',  
'how',  
'he',  
'here',  
'some',  
'thanks',  
'back',  
'im',  
'haha',  
'or']
```

## stop words

*What we can see is that stop words are the most used, but in fact they don't help us determine if a tweet is happy/sad, however, they are consuming memory and they are making the learning process slower, so we really need to get rid of them.*

In [10]:

```
from nltk.corpus import stopwords  
  
#nltk.download("stopwords")  
  
mw = most_used_words(train_data.SentimentText.str.cat())most_  
words = []for w in mw:  
  
    if len(most_words) == 1000:  
  
        break  
  
    if w in stopwords.words("english"):  
  
        continue  
  
    else:  
  
        most_words.append(w)
```

There is 133864 different words

In [11]:

```
# What we did is to filter only non stop words.# We will now  
get a look to the top 1000 wordssorted(most_words)
```

Out[11]:

```
['!',  
  
'#',  
  
$',  
  
%',  
  
&',  
  
",  
  
"d",  
  
"ll",  
  
"m",  
  
"re",  
  
"s",  
  
"ve",  
  
(,  
  
)',  
  
*',  
  
*hugs*,  
  
*sigh*,  
  
+',  
  
,',  
  
-',  
  
--',  
  
,',  
  
..',  
  
...',  
  
,',  
  
'1',  
  
'10',  
  
'100',  
  
'12',  
  
'1st',
```

'2',  
'20',  
'2nd',  
'3',  
'30',  
'30SECONDSTOMARS',  
'4',  
'5',  
'6',  
'7',  
'8',  
':',  
';',  
'=',  
'?',  
'@',  
'A',  
'AND',  
'Ah',  
'AlexAllTimeLow',  
'All',  
'Also',  
'Alyssa\_Milano',  
'Am',  
'And',  
'Are',  
'As',  
'At',  
'Aw',  
'Awesome',  
'Aww',  
'Awww',  
'BSB',



'Birthday',  
'But',  
'Ca',  
'Can',  
'Chris',  
'Come',  
'Congrats',  
'Cool',  
'D',  
'DM',  
'DO',  
'Damn',  
'Day',  
'Did',  
'Do',  
'Enjoy',  
'FF',  
'Follow',  
'FollowFriday',  
'For',  
'Friday',  
'Get',  
'Glad',  
'Go',  
'God',  
'Good',  
'Got',  
'Great',  
'Had',  
'Haha',  
'Happy',  
'Have',  
'He',

'Hello',  
'Hey',  
'Hi',  
'Hope',  
'How',  
'I',  
'IS',  
'IT',  
'If',  
'Im',  
'In',  
'Is',  
'It',  
'Its',  
'July',  
'June',  
'Just',  
'Keep',  
'LA',  
'LMAO',  
'LOL',  
'LOVE',  
'Let',  
'Like',  
'Lol',  
'London',  
'Love',  
'ME',  
'MY',  
'Maybe',  
'Me',  
'Monday',  
'Morning',

'My',  
'NO',  
'NOT',  
'New',  
'Nice',  
'Night',  
'No',  
'Not',  
'Now',  
'O',  
'OK',  
'OMG',  
'Of',  
'Oh',  
'Ok',  
'On',  
'Once',  
'One',  
'Only',  
'Or',  
'Please',  
'Poor',  
'Really',  
'S',  
'SO',  
'Saturday',  
'See',  
'She',  
'So',  
'Sorry',  
'Sounds',  
'Still',  
'Sunday',

'THAT',  
'THE',  
'TO',  
'TV',  
'Tell',  
'Thank',  
'Thanks',  
'That',  
'The',  
'Then',  
'There',  
'They',  
'This',  
'To',  
'Too',  
'Twitter',  
'U',  
'UK',  
'US',  
'Very',  
'Was',  
'We',  
'Welcome',  
'Well',  
'What',  
'When',  
'Where',  
'Who',  
'Why',  
'Will',  
'Wish',  
'Would',  
'Wow',

'XD',  
'YAY',  
'YES',  
'YOU',  
'Yay',  
'Yeah',  
'Yep',  
'Yes',  
'You',  
'Your',  
'[',  
'],  
'able',  
'absolutely',  
'account',  
'actually',  
'add',  
'afternoon',  
'ago',  
'agree',  
'ah',  
'ahh',  
'aint',  
'air',  
'album',  
'almost',  
'alone',  
'along',  
'alot',  
'already',  
'alright',  
'also',  
'always',

'amazing',  
'amp',  
'andyclemmenssen',  
'annoying',  
'another',  
'answer',  
'anymore',  
'anyone',  
'anything',  
'anyway',  
'aplusk',  
'app',  
'apparently',  
'appreciate',  
'around',  
'ashleytisdale',  
'ask',  
'asked',  
'asleep',  
'ass',  
'aw',  
'awake',  
'away',  
'awesome',  
'aww',  
'awww',  
'awwww',  
'b',  
'babe',  
'baby',  
'babygirlparis',  
'back',  
'backstreetboys',

'bad',  
'band',  
'bday',  
'beach',  
'beat',  
'beautiful',  
'bed',  
'beer',  
'behind',  
'believe',  
'best',  
'bet',  
'better',  
'big',  
'billyraycyrus',  
'birthday',  
'bit',  
'bitch',  
'black',  
'blog',  
'blue',  
'body',  
'boo',  
'book',  
'books',  
'bored',  
'boring',  
'bought',  
'bout',  
'box',  
'boy',  
'boys',  
'bradiwebbstack',

'break',  
'breakfast',  
'bring',  
'bro',  
'broke',  
'broken',  
'brother',  
'btw',  
'business',  
'busy',  
'buy',  
'c',  
'ca',  
'cake',  
'call',  
'called',  
'came',  
'camera',  
'cant',  
'car',  
'care',  
'case',  
'cat',  
'catch',  
'cause',  
'chance',  
'change',  
'chat',  
'check',  
'chocolate',  
'city',  
'class',  
'close',



'club',  
'coffee',  
'cold',  
'come',  
'comes',  
'coming',  
'comment',  
'computer',  
'concert',  
'congrats',  
'cool',  
'cos',  
'could',  
'country',  
'couple',  
'course',  
'coz',  
'crap',  
'crazy',  
'cream',  
'cry',  
'crying',  
'cut',  
'cute',  
'cuz',  
'da',  
'dad',  
'damn',  
'dance',  
'date',  
'day',  
'days',  
'dead',

'deal',  
'dear',  
'def',  
'definitely',  
'didn't',  
'die',  
'died',  
'different',  
'dinner',  
'doesn't',  
'dog',  
'done',  
'don't',  
'dream',  
'dreams',  
'drink',  
'drive',  
'dude',  
'due',  
'dunno',  
'earlier',  
'early',  
'easy',  
'eat',  
'eating',  
'eh',  
'either',  
'else',  
'em',  
'email',  
'end',  
'enjoy',  
'enjoyed',

'enjoying',  
'enough',  
'especially',  
'etc',  
'even',  
'evening',  
'ever',  
'every',  
'everyone',  
'everything',  
'exactly',  
'exam',  
'exams',  
'except',  
'excited',  
'exciting',  
'eye',  
'eyes',  
'face',  
'facebook',  
'fact',  
'fail',  
'fair',  
'fall',  
'family',  
'fan',  
'fans',  
'far',  
'fast',  
'favorite',  
'feel',  
'feeling',  
'feels',

'fell',  
'felt',  
'figure',  
'film',  
'finally',  
'find',  
'fine',  
'finish',  
'finished',  
'first',  
'fix',  
'flight',  
'follow',  
'followers',  
'followfriday',  
'following',  
'food',  
'forever',  
'forget',  
'forgot',  
'forward',  
'found',  
'free',  
'friday',  
'friend',  
'friends',  
'front',  
'fuck',  
'fucking',  
'full',  
'fun',  
'funny',  
'future',

'game',  
'gave',  
'get',  
'gets',  
'getting',  
'girl',  
'girls',  
'give',  
'giving',  
'glad',  
'go',  
'god',  
'goes',  
'goin',  
'going',  
'gon',  
'gone',  
'good',  
'goodnight',  
'gorgeous',  
'got',  
'great',  
'green',  
'gt',  
'guess',  
'guy',  
'guys',  
'ha',  
'haha',  
'hahah',  
'hahaha',  
'hair',  
'half',

'hand',  
'hang',  
'happen',  
'happened',  
'happens',  
'happy',  
'hard',  
'hate',  
'havent',  
'head',  
'hear',  
'heard',  
'heart',  
'hehe',  
'hell',  
'hello',  
'help',  
'hey',  
'hi',  
'high',  
'hit',  
'hmm',  
'hold',  
'home',  
'hope',  
'hopefully',  
'hoping',  
'horrible',  
'hot',  
'hour',  
'hours',  
'house',  
'http',

'hug',  
'huge',  
'huh',  
'hun',  
'hungry',  
'hurt',  
'hurts',  
'iPhone',  
'ice',  
'idea',  
'idk',  
'ill',  
'im',  
'inaperfectworld',  
'indeed',  
'info',  
'instead',  
'interesting',  
'internet',  
'invite',  
'iphone',  
'iremember',  
'ive',  
'jealous',  
'job',  
'join',  
'keep',  
'keeps',  
'kid',  
'kidding',  
'kids',  
'kill',  
'kind',

'kinda',  
'knew',  
'know',  
'knows',  
'lady',  
'lame',  
'laptop',  
'last',  
'late',  
'lately',  
'later',  
'laugh',  
'learn',  
'least',  
'leave',  
'leaving',  
'left',  
'less',  
'let',  
'lets',  
'life',  
'like',  
'liked',  
'lil',  
'line',  
'link',  
'list',  
'listen',  
'listening',  
'little',  
'live',  
'living',  
'lmao',



'lol',  
'long',  
'longer',  
'look',  
'looked',  
'looking',  
'looks',  
'lose',  
'lost',  
'lot',  
'lots',  
'love',  
'loved',  
'lovely',  
'loves',  
'lt',  
'luck',  
'lucky',  
'lunch',  
'luv',  
'mad',  
'made',  
'make',  
'makes',  
'making',  
'man',  
'many',  
'mate',  
'matter',  
'may',  
'maybe',  
'mean',  
'means',

'meant',  
'meet',  
'meeting',  
'mention',  
'message',  
'met',  
'might',  
'mind',  
'mine',  
'minute',  
'minutes',  
'miss',  
'missed',  
'missing',  
'mom',  
'moment',  
'monday',  
'money',  
'month',  
'months',  
'mood',  
'morning',  
'move',  
'movie',  
'movies',  
'moving',  
'much',  
'mum',  
'music',  
'musicmonday',  
'must',  
'myspace',  
'myweakness',

'n',  
"n't",  
'na',  
'name',  
'near',  
'need',  
'needed',  
'needs',  
'never',  
'new',  
'news',  
'next',  
'nice',  
'night',  
'nite',  
'nope',  
'nothing',  
'number',  
'office',  
'oh',  
'ok',  
'okay',  
'old',  
'omg',  
'one',  
'ones',  
'online',  
'open',  
'order',  
'others',  
'outside',  
'p',  
'page',

'pain',  
'parents',  
'part',  
'party',  
'pass',  
'past',  
'pay',  
'people',  
'perfect',  
'person',  
'phone',  
'photo',  
'photos',  
'pic',  
'pick',  
'pics',  
'picture',  
'pictures',  
'place',  
'plan',  
'plans',  
'play',  
'played',  
'playing',  
'please',  
'plus',  
'point',  
'pool',  
'poor',  
'post',  
'posted',  
'power',  
'ppl',

'pretty',  
'prob',  
'probably',  
'problem',  
'profile',  
'proud',  
'put',  
'question',  
'quite',  
'quot',  
'r',  
'radio',  
'rain',  
'raining',  
'random',  
'rather',  
'read',  
'reading',  
'ready',  
'real',  
'really',  
'reason',  
'red',  
'remember',  
'reply',  
'rest',  
'ride',  
'right',  
'rock',  
'room',  
'run',  
'running',  
'sad',

'sadly',  
'safe',  
'said',  
'save',  
'saw',  
'say',  
'saying',  
'says',  
'scared',  
'school',  
'season',  
'second',  
'see',  
'seeing',  
'seem',  
'seems',  
'seen',  
'self',  
'send',  
'sense',  
'sent',  
'serious',  
'seriously',  
'set',  
'sexy',  
'shall',  
'shame',  
'share',  
'sharing',  
'shit',  
'shopping',  
'short',  
'show',

'shows',  
'shut',  
'sick',  
'side',  
'sign',  
'silly',  
'since',  
'single',  
'sis',  
'sister',  
'site',  
'sitting',  
'sleep',  
'sleeping',  
'slow',  
'small',  
'smile',  
'sold',  
'someone',  
'something',  
'sometimes',  
'somewhere',  
'son',  
'song',  
'songs',  
'soo',  
'soon',  
'sooo',  
'soooo',  
'sorry',  
'sort',  
'sound',  
'sounds',

'special',  
'squarespace',  
'start',  
'started',  
'starting',  
'stay',  
'still',  
'stop',  
'store',  
'story',  
'stuck',  
'study',  
'stuff',  
'stupid',  
'suck',  
'sucks',  
'summer',  
'sun',  
'sunday',  
'sunny',  
'super',  
'support',  
'supposed',  
'sure',  
'sweet',  
'sweetie',  
'ta',  
'take',  
'taken',  
'taking',  
'talk',  
'talking',  
'tea',



'team',  
'tell',  
'test',  
'text',  
'thank',  
'thanks',  
'thats',  
'thing',  
'things',  
'think',  
'thinking',  
'tho',  
'though',  
'thought',  
'three',  
'thx',  
'tickets',  
'til',  
'till',  
'time',  
'times',  
'tired',  
'today',  
'together',  
'told',  
'tomorrow',  
'tonight',  
'took',  
'top',  
'totally',  
'touch',  
'tour',  
'town',

'train',  
'tried',  
'trip',  
'true',  
'try',  
'trying',  
'turn',  
'tv',  
'tweet',  
'tweeting',  
'tweets',  
'twitter',  
'two',  
'type',  
'u',  
'ugh',  
'understand',  
'unfortunately',  
'update',  
'updates',  
'upset',  
'ur',  
'us',  
'use',  
'used',  
'using',  
'usually',  
'version',  
'via',  
'video',  
'vip',  
'visit',  
'voice',

'vote',  
'w/',  
'wait',  
'waiting',  
'wake',  
'walk',  
'wan',  
'want',  
'wanted',  
'wants',  
'warm',  
'watch',  
'watched',  
'watching',  
'water',  
'way',  
'wear',  
'weather',  
'website',  
'wedding',  
'week',  
'weekend',  
'weeks',  
'weird',  
'welcome',  
'well',  
'went',  
'whatever',  
'whats',  
'white',  
'whole',  
'win',  
'wine',

'wish',  
'wit',  
'without',  
'wo',  
'woke',  
'woman',  
'wonder',  
'wonderful',  
'wont',  
'word',  
'words',  
'work',  
'worked',  
'working',  
'works',  
'world',  
'worry',  
'worse',  
'worst',  
'worth',  
'would',  
'wow',  
'write',  
'writing',  
'wrong',  
'x',  
'xD',  
'xoxo',  
'xx',  
'xxx',  
'ya',  
'yay',  
'yea',

```
'yeah',
'year',
'years',
'yep',
'yes',
'yesterday',
'yet',
'yo',
'youtube',
'yup',
'|',
'~',
'💎']
```

## Stemming

*You should have noticed something, right? There are words that have the same meaning, but written in a different manner, sometimes in the plural and sometimes with a suffix (ing, es ...), this will make our model think that they are different words and also make our vocabulary bigger (waste of memory and time for the learning process). The solution is to reduce those words with the same root, this is called stemming.*

In [12]:

```
# I'm defining this function to use it in the # Data Preparation Phase
from nltk.stem.snowball import SnowballStemmer
from nltk.stem import WordNetLemmatizer
```

```
#nltk.download('wordnet')
def stem_tokenize(text):
```

```
    stemmer = SnowballStemmer("english")
```

```
    stemmer = WordNetLemmatizer()
```

```
    return [stemmer.lemmatize(token) for token in word_tokenize(text)]
```

```
def lemmatize_tokenize(text):
```

```
    lemmatizer = WordNetLemmatizer()
```

```
    return [lemmatizer.lemmatize(token) for token in word_tokenize(text)]
```

will stop here, but you can visualize tweets more and more to gain insights and take decisions about how to transform your data.

## Prepare the data

In this phase, we will transform our tweets into a more usable data by our ML models.

## Bag of Words

We are going to use the Bag of Words algorithm, which basically takes a text as input, extract words from it (this is our vocabulary) to use them in the vectorization process. When a tweet comes in, it will vectorize it by counting the number of occurrences of each word in our vocabulary.

For example, we have this two tweets: "I learned a lot today" and "hahaha I got you".

*tweet / words* I learned a lot today hahaha got  
you

1.

*first* 1 1 1 1 1 0 0 0

2.

•

*second* 1 0 0 0 0 1 1 1

•

In [13]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

## Building the pipeline

It's always a good practice to make a pipeline of transformation for your data, it will make the process of data transformation really easy and reusable. We will implement a pipeline for transforming our tweets to something that our ML models can digest (vectors)

In [14]:

```
from sklearn.base import TransformerMixin, BaseEstimator  
from sklearn.pipeline import Pipeline
```

In [15]:

*# We need to do some preprocessing of the tweets. # We will delete useless strings (like @, # ...) # because we think that they will not help in determining if the person is Happy/Sad*

```
class TextPreProc(BaseEstimator,TransformerMixin):

    def __init__(self, use_mention=False):

        self.use_mention = use_mention


    def fit(self, X, y=None):

        return self


    def transform(self, X, y=None):

        # We can choose between keeping the mentions

        # or deleting them

        if self.use_mention:

            X = X.str.replace(r"@[a-zA-Z0-9_]* ", " @tags ")

        else:

            X = X.str.replace(r"@[a-zA-Z0-9_]* ", "")


        # Keeping only the word after the #

        X = X.str.replace("#", "")

        X = X.str.replace(r"[-.\n]", "")

        # Removing HTML garbage

        X = X.str.replace(r"&\w+;", "")

        # Removing links

        X = X.str.replace(r"https?:\/\/\S*", "")

        # replace repeated letters with only two occurrences
```

```
# heeeellllloooo => heelloo
```

```
X = X.str.replace(r"(\.)\1+", r"\1\1")
```

```
# mark emoticons as happy or sad
```

```
X = X.str.replace(HAPPY_EMO, " happyemoticons ")
```

```
X = X.str.replace(SAD_EMO, " sademoticons ")
```

```
X = X.str.lower()
```

```
return X
```

In [16]:

```
# This is the pipeline that will transform our tweets to some  
# thing eatable. # You can see that we are using our previously  
# defined stemmer, it will take care of the stemming process.  
# For stop words, we let the inverse document frequency do the  
# job  
from sklearn.model_selection import train_test_split
```

```
sentiments = train_data['Sentiment'] tweets = train_data['SentimentText']
```

```
# I get those parameters from the 'Fine tune the model' part  
vectorizer = TfidfVectorizer(tokenizer=lemmatize_tokenize, ngram_range=(1,2))  
pipeline = Pipeline([
```

```
    ('text_pre_processing', TextPreProc(use_mention=True)),
```

```
    ('vectorizer', vectorizer),])
```

```
# Let's split our data into learning set and testing set  
# This process is done to test the efficiency of our model at the end.  
# You shouldn't look at the test data only after choosing the final model  
learn_data, test_data, sentiments_learning, sentiments_test = train_test_split(tweets, sentiments, test_size=0.3)
```

```
# This will transform our learning data from simple text to vector  
# by going through the preprocessing transformer.  
learning_data = pipeline.fit_transform(learn_data)
```

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:17: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:23: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:25: FutureWarning: The default value of regex will change from True to False in a future version.



/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:27: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:30: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:32: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:33: FutureWarning: The default value of regex will change from True to False in a future version.

## Select a model

*When we have our data ready to be processed by ML models, the question we should ask is which model to use?*

*The answer varies depending on the problem and data, for example, it's known that Naive Bias has proven good efficacy against Text Based Problems.*

*A good way to choose a model is to try different candidate, evaluate them using cross validation, then chose the best one which will be later tested against our test data.*

In [17]:

```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import BernoulliNB, MultinomialNB
```

```
lr = LogisticRegression()
bnb = BernoulliNB()
mnb = MultinomialNB()
```

```
models = {
    'logistic regression': lr,
    'bernoulliNB': bnb,
    'multinomialNB': mnb,
}
```

```
for model in models.keys():
    scores = cross_val_score(models[model], learning_data, sentiments_learning,
                              scoring="f1", cv=10)

    print("===", model, "===")

    print("scores = ", scores)

    print("mean = ", scores.mean())

    print("variance = ", scores.var())
```

```
models[model].fit(learning_data, sentiments_learning)
```

```
print("score on the learning data (accuracy) = ", accuracy_score(models[model].predict(learning_data), sentiments_learning))
```

```
print("")
```

/opt/conda/lib/python3.7/site-packages/sklearn/linear\_model/\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG)

/opt/conda/lib/python3.7/site-packages/sklearn/linear\_model/\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG)

/opt/conda/lib/python3.7/site-packages/sklearn/linear\_model/\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG)

/opt/conda/lib/python3.7/site-packages/sklearn/linear\_model/\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG)

/opt/conda/lib/python3.7/site-packages/sklearn/linear\_model/\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG)

/opt/conda/lib/python3.7/site-packages/sklearn/linear\_model/\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG)

/opt/conda/lib/python3.7/site-packages/sklearn/linear\_model/\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG)

/opt/conda/lib/python3.7/site-packages/sklearn/linear\_model/\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG)

/opt/conda/lib/python3.7/site-packages/sklearn/linear\_model/\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG)

/opt/conda/lib/python3.7/site-packages/sklearn/linear\_model/\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG)

=== logititc regression ===

scores = [0.80952957 0.80996132 0.80512573 0.80941516 0.81088894 0.8125908  
0.81534023 0.81049212 0.80781759 0.81136638]

mean = 0.810252782485254

variance = 6.657698428255754e-06

/opt/conda/lib/python3.7/site-packages/sklearn/linear\_model/\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG)

score on the learning data (accuracy) = 0.8717996342439136

```
=== bernoulliNB ===
```

```
scores = [0.79156909 0.78775079 0.78224201 0.78791387 0.79228972 0.78485451
0.78859527 0.78725639 0.78480122 0.78928906]
```

```
mean = 0.7876561913042192
```

```
variance = 8.58773568689337e-06
```

```
score on the learning data (accuracy) = 0.9027317407703738
```

```
=== multinomialNB ===
```

```
scores = [0.8087226 0.81302801 0.80400445 0.80567879 0.80681944 0.80842105
0.80549098 0.80649105 0.8067917 0.8118196 ]
```

```
mean = 0.8077267675835677
```

```
variance = 7.260135677129156e-06
```

```
score on the learning data (accuracy) = 0.8981312149959996
```

*None of those models is likely to be overfitting, I will choose the multinomialNB.*

## ***Fine tune the model***

*I'm going to use the GridSearchCV to choose the best parameters to use.*

*What the GridSearchCV does is trying different set of parameters, and for each one, it runs a cross validation and estimate the score. At the end we can see what are the best parameter and use them to build a better classifier.*

In [18]:

```
from sklearn.model_selection import GridSearchCV
```

```
grid_search_pipeline = Pipeline([
```

```
    ('text_pre_processing', TextPreProc()),
```

```
    ('vectorizer', TfidfVectorizer()),
```

```

('model', MultinomialNB()),])

params = [

    {

        'text_pre_processing__use_mention': [True, False],

        'vectorizer__max_features': [1000, 2000, 5000, 10000,
20000, None],

        'vectorizer__ngram_range': [(1,1), (1,2)],

    },]grid_search = GridSearchCV(grid_search_pipeline, param
s, cv=5, scoring='f1')grid_search.fit(learn_data, sentiments_
learning)print(grid_search.best_params_)

```

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:17: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:23: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:25: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:27: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:30: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:32: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:33: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:17: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:23: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:25: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:27: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:30: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:32: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:33: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:17: FutureWarning: The default value of regex will change from True to False in a future version.



















































































































































































```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:32: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:33: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:19: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:23: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:25: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:27: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:30: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:32: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:33: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:17: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:23: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:25: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:27: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:30: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:32: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:33: FutureWarning: The default value of regex will change from True to False in a future version.
```

```
{'text_pre_processing__use_mention': True, 'vectorizer__max_features': None, 'vectorizer__ngram_range': (1, 2)}
```

*Testing our model against data other than the data used for training our model will show how well the model is generalising on new data.*

## Note

*We shouldn't test to choose the model, this will only let us confirm that the chosen model is doing well.*

In [19]:

```
mnb.fit(learning_data, sentiments_learning)
```

Out[19]:



MultinomialNB()

In [20]:

```
testing_data = pipeline.transform(test_data)mnb.score(testing_data, sentiments_test)
```

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:17: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:23: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:25: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:27: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:30: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:32: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:33: FutureWarning: The default value of regex will change from True to False in a future version.

Out[20]:

0.7543754375437544

*Not bad for my first attempt to solve a sentiment analysis problem. I will try to make it better if I got more free time.*

In [21]:

```
# Predicting on the test.csvsub_data = pd.read_csv("/kaggle/input/test-1-sentimentanalysis/test_sentiment_analysis.csv", encoding='ISO-8859-1')sub_learning = pipeline.transform(sub_data.SentimentText)sub = pd.DataFrame(sub_data.ItemID, columns=("ItemID", "Sentiment"))sub["Sentiment"] = mnb.predict(sub_learning)print(sub)
```

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:17: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:23: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:25: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:27: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:30: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:32: FutureWarning: The default value of regex will change from True to False in a future version.

/opt/conda/lib/python3.7/site-packages/ipykernel\_launcher.py:33: FutureWarning: The default value of regex will change from True to False in a future version.



	ItemID	Sentiment
0	1	0
1	2	0
2	3	1
3	4	0
4	5	0
...	...	...
299984	299996	1
299985	299997	1
299986	299998	1
299987	299999	1
299988	300000	1

[299989 rows x 2 columns]

## Test your tweet

*The most exciting part ! Don't be too hard with my classifier...*

In [22]:

```
# Just run it
model = MultinomialNB()
model.fit(learning_data,
sentiments_learning)
tweet = pd.Series([input(),])
tweet = pipeline.transform(tweet)
proba = model.predict_proba(tweet)[0]
print("The probability that this tweet is sad is:", proba[0])
print("The probability that this tweet is happy is:", proba[1])
```

```
-----StdinNotImplemented
Error                                Traceback (most recent call last)
<ipython-input-22-22515163dabf> in <module>
      2 model = MultinomialNB()
      3 model.fit(learning_data, sentiments_learning)
      4 tweet = pd.Series([input(),])
      5 tweet = pipeline.transform(tweet)
      6 proba = model.predict_proba(tweet)[0]
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel/kernelbase.py in raw_input(self, prompt)
53     if not self._allow_stdin:
54         raise StdinNotImplementedError(
55             "raw_input was called, but this frontend does not support input requests."
56         )
57     return self._input_request(str(prompt),
```

```
StdinNotImplementedError: raw_input was called, but this frontend does not support input requests.
```