

theoretical1

January 30, 2023

1 Tweet classification with naive bayes

For this notebook we are going to implement a naive bayes classifier for classifying positive or negative based on the words in the tweet. Recall that for two events A and B the bayes theorem says

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where $P(A)$ and $P(B)$ is the **class probabilities** and $P(B|A)$ is called **conditional probabilities**. this gives us the probability of A happening, given that B has occurred. So as an example if we want to find the probability of “is this a positive tweet given that it contains the word”good” ” we will obtain the following

$$P(\text{"positive"}|\text{"good" in tweet}) = \frac{P(\text{"good" in tweet}|\text{"positive"})P(\text{"positive"})}{P(\text{"good" in tweet})}$$

This means that to find the probability of “is this a positive tweet given that it contains the word”good” ” we need the probability of “good” being in a positive tweet, the probability of a tweet being positive and the probability of “good” being in a tweet.

Similarly if we want to obtain the opposite “is this a negative tweet given that it contains the word”boring” ” we get

$$P(\text{"negative"}|\text{"boring" in tweet}) = \frac{P(\text{"boring" in tweet}|\text{"negative"})P(\text{"negative"})}{P(\text{"boring" in tweet})}$$

where we need the probability of “boring” being in a negative tweet, the probability of a tweet negative being and the probability of “boring” being in a tweet.

We can now build a classifier where we compare those two probabilities and whichever is the larger one it’s classified as

if $P(\text{"positive"}|\text{"good" in tweet}) > P(\text{"negative"}|\text{"boring" in tweet})$

Tweet is positive

else

Tweet is negative

Now let's expand this to handle multiple features and put the Naive assumption into bayes theroem. This means that if features are independent we have

$$P(A, B) = P(A)P(B)$$

This gives us:

$$P(A|b_1, b_2, \dots, b_n) = \frac{P(b_1|A)P(b_2|A)\dots P(b_n|A)P(A)}{P(b_1)P(b_2)\dots P(b_n)}$$

or

$$P(A|b_1, b_2, \dots, b_n) = \frac{\prod_i^n P(b_i|A)P(A)}{P(b_1)P(b_2)\dots P(b_n)}$$

So with our previous example expanded with more words "is this a positive tweet given that it contains the word"good" and "interesting" " gives us

$$P(\text{"positive"}|\text{"good", "interesting" in tweet}) = \frac{P(\text{"good" in tweet}|\text{"positive"})P(\text{"interesting" in tweet}|\text{"positive"})P(\text{"positive"})}{P(\text{"good" in tweet})P(\text{"interesting" in tweet})P(\text{"positive"})}$$

As you can see the denominator remains constant which means we can remove it and the final classifier end up

$$y = \underset{A}{\operatorname{argmax}} P(A) \prod_i^n P(b_i|A)$$

The dataset that you will be working with can be downloaded from the following link: <https://uppsala.instructure.com/courses/66466/files>

```
[ ]: #stuff to import
import pandas as pd
import numpy as np
import random
import sklearn
from sklearn.model_selection import train_test_split
```

Load the data, explore and pre-processing

```
[ ]: tweets=pd.read_csv('twitter_sentiment_analysis.csv',encoding='latin',
                        names = ['sentiment','id','date','query','user','tweet'])
tweets
```

```
[ ]:      sentiment      id      date      query \
0          0  1467810369  Mon Apr 06 22:19:45 PDT 2009  NO_QUERY
1          0  1467810672  Mon Apr 06 22:19:49 PDT 2009  NO_QUERY
```

```

2          0 1467810917 Mon Apr 06 22:19:53 PDT 2009 NO_QUERY
3          0 1467811184 Mon Apr 06 22:19:57 PDT 2009 NO_QUERY
4          0 1467811193 Mon Apr 06 22:19:57 PDT 2009 NO_QUERY
...
1599995    4 2193601966 Tue Jun 16 08:40:49 PDT 2009 NO_QUERY
1599996    4 2193601969 Tue Jun 16 08:40:49 PDT 2009 NO_QUERY
1599997    4 2193601991 Tue Jun 16 08:40:49 PDT 2009 NO_QUERY
1599998    4 2193602064 Tue Jun 16 08:40:49 PDT 2009 NO_QUERY
1599999    4 2193602129 Tue Jun 16 08:40:50 PDT 2009 NO_QUERY

```

```

          user          tweet
0  _TheSpecialOne_  @switchfoot http://twitpic.com/2y1zl - Awww, t...
1    scotthamilton  is upset that he can't update his Facebook by ...
2      mattycus    @Kenichan I dived many times for the ball. Man...
3      ElleCTF     my whole body feels itchy and like its on fire
4      Karoli      @nationwideclass no, it's not behaving at all...
...
1599995 AmandaMarie1028 Just woke up. Having no school is the best fee...
1599996   TheWDBboards  TheWDB.com - Very cool to hear old Walt interv...
1599997      bpbabe     Are you ready for your MoJo Makeover? Ask me f...
1599998   tinydiamondz  Happy 38th Birthday to my boo of alll time!!! ...
1599999   RyanTrevMorris happy #charitytuesday @theNSPCC @SparksCharity...

```

[1600000 rows x 6 columns]

```
[ ]: tweets = tweets.sample(frac=1)
      tweets = tweets[:200000]
      print("Dataset shape:", tweets.shape)
```

Dataset shape: (200000, 6)

```
[ ]: tweets['sentiment'].unique()
```

```
[ ]: array([4, 0])
```

Currently (0 = negative and 4 = positive) changing the notation to (0 = negative and 1 = positive)

```
[ ]: tweets['sentiment']=tweets['sentiment'].replace(4,1)
      tweets
```

```
[ ]:
      sentiment      id      date      query \
807355          1 1468973998 Tue Apr 07 04:39:45 PDT 2009 NO_QUERY
714751          0 2259219365 Sat Jun 20 17:12:46 PDT 2009 NO_QUERY
656892          0 2240723278 Fri Jun 19 10:05:11 PDT 2009 NO_QUERY
1164846         1 1979796158 Sun May 31 04:32:20 PDT 2009 NO_QUERY
1571157         1 2188719371 Mon Jun 15 22:25:37 PDT 2009 NO_QUERY

```

```

...
1369467      1  2050811765  Fri Jun 05 19:20:09 PDT 2009  NO_QUERY
246461      0  1982247677  Sun May 31 10:49:39 PDT 2009  NO_QUERY
1105450      1  1971291151  Sat May 30 06:44:29 PDT 2009  NO_QUERY
870490      1  1678389499  Sat May 02 06:32:54 PDT 2009  NO_QUERY
1549944      1  2183625334  Mon Jun 15 14:45:49 PDT 2009  NO_QUERY

      user      tweet
807355      jenibake      @SwandaG if you find one, please pass it on
714751  OhDearItsCarmen      its like im always sick
656892      kellyhuckvale  @OfficialBB i dont want either of them to sre...
1164846      Shanniece      @KtotheCdawg omg yes  thatd be heaps cool
1571157      InnocentVice      Ok... going to bed.

...
1369467  JennifferJanna  Looking forward to 100 followers on Follow Fri...
246461  ImGingerSmith  @Wordlily I've done many of those things ...
1105450  hyperjetrod      more new followerscan they keep up
870490  DiyanaRazinah      A cup of tea cures emoness
1549944  u_foundme  @therealsavannah http://twitpic.com/7hviu - we...

[200000 rows x 6 columns]

```

Removing the unnecessary columns.

```

[ ]: tweets.drop(['date', 'query', 'user'], axis=1, inplace=True)
     tweets.drop('id', axis=1, inplace=True)
     tweets.head(10)

```

```

[ ]:      sentiment      tweet
807355      1      @SwandaG if you find one, please pass it on
714751      0      its like im always sick
656892      0  @OfficialBB i dont want either of them to sre...
1164846      1      @KtotheCdawg omg yes  thatd be heaps cool
1571157      1      Ok... going to bed.
81838      0      I wish I could b with my mommy on mothers day
615370      0  @Shnicky_lover aww chick thats not so good. W...
1356171      1      @aikigirl77 there you go again!
832990      1  @_AislinnTighee wow your coool ;) lalalala th...
395097      0  @sergeantkero sometimes growing up is scary ,...

```

Checking if any null values present

```

[ ]: (tweets.isnull().sum() / len(tweets))*100

[ ]: sentiment      0.0
     tweet          0.0
     dtype: float64

```

Now make a new column for side by side comparison of new tweets vs old tweets

```
[ ]: #converting pandas object to a string type
tweets['tweet'] = tweets['tweet'].astype('str')
```

Check the number of positive vs. negative tagged sentences

```
[ ]: positives = tweets['sentiment'][tweets.sentiment == 1 ]
negatives = tweets['sentiment'][tweets.sentiment == 0 ]

print('Total length of the data is:          {}'.format(tweets.shape[0]))
print('No. of positive tagged sentences is: {}'.format(len(positives)))
print('No. of negative tagged sentences is: {}'.format(len(negatives)))
```

```
Total length of the data is:          200000
No. of positive tagged sentences is:   99820
No. of negative tagged sentences is: 100180
```

```
[ ]: # nltk
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
#Stop Words: A stop word is a commonly used word (such as "the", "a", "an",
↳ "in")
#that a search engine has been programmed to ignore,
#both when indexing entries for searching and when retrieving them as the
↳ result of a search query.
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('omw-1.4')
nltk.download('wordnet')
stopword = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data] /home/rilo3075/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /home/rilo3075/nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /home/rilo3075/nltk_data...
[nltk_data] Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package wordnet to /home/rilo3075/nltk_data...
[nltk_data] Package wordnet is already up-to-date!
```

Data Cleaning

```
[ ]: import warnings
warnings.filterwarnings('ignore')
import re
```

```

import string
import pickle
urlPattern = r"((http://)[^ ]*|(https://)[^ ]*|( www\.)[^ ]*)"
userPattern = '@[^\s]+'
some = 'amp,today,tomorrow,going,girl'
def process_tweets(tweet):
    # Lower Casing
    tweet = re.sub(r"he's", "he is", tweet)
    tweet = re.sub(r"there's", "there is", tweet)
    tweet = re.sub(r"We're", "We are", tweet)
    tweet = re.sub(r"That's", "That is", tweet)
    tweet = re.sub(r"won't", "will not", tweet)
    tweet = re.sub(r"they're", "they are", tweet)
    tweet = re.sub(r"Can't", "Cannot", tweet)
    tweet = re.sub(r"wasn't", "was not", tweet)
    tweet = re.sub(r"don\x89\u00at", "do not", tweet)
    tweet = re.sub(r"aren't", "are not", tweet)
    tweet = re.sub(r"isn't", "is not", tweet)
    tweet = re.sub(r"What's", "What is", tweet)
    tweet = re.sub(r"haven't", "have not", tweet)
    tweet = re.sub(r"hasn't", "has not", tweet)
    tweet = re.sub(r"There's", "There is", tweet)
    tweet = re.sub(r"He's", "He is", tweet)
    tweet = re.sub(r"It's", "It is", tweet)
    tweet = re.sub(r"You're", "You are", tweet)
    tweet = re.sub(r"I'M", "I am", tweet)
    tweet = re.sub(r"shouldn't", "should not", tweet)
    tweet = re.sub(r"wouldn't", "would not", tweet)
    tweet = re.sub(r"i'm", "I am", tweet)
    tweet = re.sub(r"I\x89\u00am", "I am", tweet)
    tweet = re.sub(r"I'm", "I am", tweet)
    tweet = re.sub(r"Isn't", "is not", tweet)
    tweet = re.sub(r"Here's", "Here is", tweet)
    tweet = re.sub(r"you've", "you have", tweet)
    tweet = re.sub(r"you\x89\u00ave", "you have", tweet)
    tweet = re.sub(r"we're", "we are", tweet)
    tweet = re.sub(r"what's", "what is", tweet)
    tweet = re.sub(r"couldn't", "could not", tweet)
    tweet = re.sub(r"we've", "we have", tweet)
    tweet = re.sub(r"it\x89\u00as", "it is", tweet)
    tweet = re.sub(r"doesn\x89\u00at", "does not", tweet)
    tweet = re.sub(r"It\x89\u00as", "It is", tweet)
    tweet = re.sub(r"Here\x89\u00as", "Here is", tweet)
    tweet = re.sub(r"who's", "who is", tweet)
    tweet = re.sub(r"I\x89\u00ave", "I have", tweet)
    tweet = re.sub(r"y'all", "you all", tweet)
    tweet = re.sub(r"can\x89\u00at", "cannot", tweet)

```

```

tweet = re.sub(r"would've", "would have", tweet)
tweet = re.sub(r"it'll", "it will", tweet)
tweet = re.sub(r"we'll", "we will", tweet)
tweet = re.sub(r"wouldn\u0027t", "would not", tweet)
tweet = re.sub(r"We've", "We have", tweet)
tweet = re.sub(r"he'll", "he will", tweet)
tweet = re.sub(r"Y'all", "You all", tweet)
tweet = re.sub(r"Weren't", "Were not", tweet)
tweet = re.sub(r"Didn't", "Did not", tweet)
tweet = re.sub(r"they'll", "they will", tweet)
tweet = re.sub(r"they'd", "they would", tweet)
tweet = re.sub(r"DON'T", "DO NOT", tweet)
tweet = re.sub(r"That\u0027s", "That is", tweet)
tweet = re.sub(r"they've", "they have", tweet)
tweet = re.sub(r"i'd", "I would", tweet)
tweet = re.sub(r"should've", "should have", tweet)
tweet = re.sub(r"You\u0027re", "You are", tweet)
tweet = re.sub(r"where's", "where is", tweet)
tweet = re.sub(r"Don\u0027t", "Do not", tweet)
tweet = re.sub(r"we'd", "we would", tweet)
tweet = re.sub(r"i'll", "I will", tweet)
tweet = re.sub(r"weren't", "were not", tweet)
tweet = re.sub(r"They're", "They are", tweet)
tweet = re.sub(r"Can\u0027t", "Cannot", tweet)
tweet = re.sub(r"you\u0027ll", "you will", tweet)
tweet = re.sub(r"I\u0027d", "I would", tweet)
tweet = re.sub(r"let's", "let us", tweet)
tweet = re.sub(r"it's", "it is", tweet)
tweet = re.sub(r"can't", "cannot", tweet)
tweet = re.sub(r"don't", "do not", tweet)
tweet = re.sub(r"you're", "you are", tweet)
tweet = re.sub(r"i've", "I have", tweet)
tweet = re.sub(r"that's", "that is", tweet)
tweet = re.sub(r"i'll", "I will", tweet)
tweet = re.sub(r"doesn't", "does not", tweet)
tweet = re.sub(r"i'd", "I would", tweet)
tweet = re.sub(r"didn't", "did not", tweet)
tweet = re.sub(r"ain't", "am not", tweet)
tweet = re.sub(r"you'll", "you will", tweet)
tweet = re.sub(r"I've", "I have", tweet)
tweet = re.sub(r"Don't", "do not", tweet)
tweet = re.sub(r"I'll", "I will", tweet)
tweet = re.sub(r"I'd", "I would", tweet)
tweet = re.sub(r"Let's", "Let us", tweet)
tweet = re.sub(r"you'd", "You would", tweet)
tweet = re.sub(r"It's", "It is", tweet)
tweet = re.sub(r"Ain't", "am not", tweet)

```

```

tweet = re.sub(r"Haven't", "Have not", tweet)
tweet = re.sub(r"Could've", "Could have", tweet)
tweet = re.sub(r"youve", "you have", tweet)
tweet = re.sub(r"donâ&t", "do not", tweet)

tweet = re.sub(r"some1", "someone", tweet)
tweet = re.sub(r"yrs", "years", tweet)
tweet = re.sub(r"hrs", "hours", tweet)
tweet = re.sub(r"2morow|2moro", "tomorrow", tweet)
tweet = re.sub(r"2day", "today", tweet)
tweet = re.sub(r"4got|4gotten", "forget", tweet)
tweet = re.sub(r"b-day|bday", "b-day", tweet)
tweet = re.sub(r"mother's", "mother", tweet)
tweet = re.sub(r"mom's", "mom", tweet)
tweet = re.sub(r"dad's", "dad", tweet)
tweet = re.sub(r"hahah|hahaha|hahahaha", "haha", tweet)
tweet = re.sub(r"lmao|lolz|rofl", "lol", tweet)
tweet = re.sub(r"thanx|thnx", "thanks", tweet)
tweet = re.sub(r"goood", "good", tweet)
tweet = re.sub(r"some1", "someone", tweet)
tweet = re.sub(r"some1", "someone", tweet)
tweet = tweet.lower()
tweet=tweet[1:]
# Removing all URLs
tweet = re.sub(urlPattern, '', tweet)
# Removing all @username.
tweet = re.sub(userPattern, '', tweet)
#remove some words
tweet= re.sub(some, '', tweet)
#Remove punctuations
tweet = tweet.translate(str.maketrans("", "", string.punctuation))
#tokenizing words
tokens = word_tokenize(tweet)
#tokens = [w for w in tokens if len(w)>2]
#Removing Stop Words
final_tokens = [w for w in tokens if w not in stopword]
#reducing a word to its word stem
wordLemm = WordNetLemmatizer()
finalwords=[]
for w in final_tokens:
    if len(w)>1:
        word = wordLemm.lemmatize(w)
        finalwords.append(word)
return ' '.join(finalwords)

```

```

[ ]: abbreviations = {
    "$" : " dollar ",

```


"€" : " euro ",
"4ao" : "for adults only",
"a.m" : "before midday",
"a3" : "anytime anywhere anyplace",
"aamof" : "as a matter of fact",
"acct" : "account",
"adih" : "another day in hell",
"afaic" : "as far as i am concerned",
"afaict" : "as far as i can tell",
"afaik" : "as far as i know",
"afair" : "as far as i remember",
"afk" : "away from keyboard",
"app" : "application",
"approx" : "approximately",
"apps" : "applications",
"asap" : "as soon as possible",
"asl" : "age, sex, location",
"atk" : "at the keyboard",
"ave." : "avenue",
"aymm" : "are you my mother",
"ayor" : "at your own risk",
"b&b" : "bed and breakfast",
"b+b" : "bed and breakfast",
"b.c" : "before christ",
"b2b" : "business to business",
"b2c" : "business to customer",
"b4" : "before",
"b4n" : "bye for now",
"b@u" : "back at you",
"bae" : "before anyone else",
"bak" : "back at keyboard",
"bbbg" : "bye bye be good",
"bbc" : "british broadcasting corporation",
"bbias" : "be back in a second",
"bbl" : "be back later",
"bbs" : "be back soon",
"be4" : "before",
"bfn" : "bye for now",
"blvd" : "boulevard",
"bout" : "about",
"brb" : "be right back",
"bros" : "brothers",
"brt" : "be right there",
"bsaaw" : "big smile and a wink",
"btw" : "by the way",
"bwl" : "bursting with laughter",
"c/o" : "care of",

"cet" : "central european time",
"cf" : "compare",
"cia" : "central intelligence agency",
"csl" : "can not stop laughing",
"cu" : "see you",
"cul8r" : "see you later",
"cv" : "curriculum vitae",
"cwot" : "complete waste of time",
"cya" : "see you",
"cyt" : "see you tomorrow",
"dae" : "does anyone else",
"dbmib" : "do not bother me i am busy",
"diy" : "do it yourself",
"dm" : "direct message",
"dwh" : "during work hours",
"e123" : "easy as one two three",
"eet" : "eastern european time",
"eg" : "example",
"embm" : "early morning business meeting",
"encl" : "enclosed",
"encl." : "enclosed",
"etc" : "and so on",
"faq" : "frequently asked questions",
"fawc" : "for anyone who cares",
"fb" : "facebook",
"fc" : "fingers crossed",
"fig" : "figure",
"fimh" : "forever in my heart",
"ft." : "feet",
"ft" : "featuring",
"ftl" : "for the loss",
"ftw" : "for the win",
"fwiw" : "for what it is worth",
"fyi" : "for your information",
"g9" : "genius",
"gahoy" : "get a hold of yourself",
"gal" : "get a life",
"gcse" : "general certificate of secondary education",
"gfn" : "gone for now",
"gg" : "good game",
"gl" : "good luck",
"glhf" : "good luck have fun",
"gmt" : "greenwich mean time",
"gmta" : "great minds think alike",
"gn" : "good night",
"g.o.a.t" : "greatest of all time",
"goat" : "greatest of all time",

"goi" : "get over it",
"gps" : "global positioning system",
"gr8" : "great",
"gratz" : "congratulations",
"gyal" : "girl",
"h&c" : "hot and cold",
"hp" : "horsepower",
"hr" : "hour",
"hrh" : "his royal highness",
"ht" : "height",
"ibrb" : "i will be right back",
"ic" : "i see",
"icq" : "i seek you",
"icymi" : "in case you missed it",
"idc" : "i do not care",
"idgadf" : "i do not give a damn fuck",
"idgaf" : "i do not give a fuck",
"idk" : "i do not know",
"ie" : "that is",
"i.e" : "that is",
"ifyp" : "i feel your pain",
"IG" : "instagram",
"iirc" : "if i remember correctly",
"ilu" : "i love you",
"ily" : "i love you",
"imho" : "in my humble opinion",
"imo" : "in my opinion",
"imu" : "i miss you",
"iow" : "in other words",
"irl" : "in real life",
"j4f" : "just for fun",
"jic" : "just in case",
"jk" : "just kidding",
"jsyk" : "just so you know",
"l8r" : "later",
"lb" : "pound",
"lbs" : "pounds",
"ldr" : "long distance relationship",
"lmao" : "laugh my ass off",
"lmfao" : "laugh my fucking ass off",
"lol" : "laughing out loud",
"ltd" : "limited",
"ltns" : "long time no see",
"m8" : "mate",
"mf" : "motherfucker",
"mfs" : "motherfuckers",
"mfw" : "my face when",

"mofo" : "motherfucker",
"mph" : "miles per hour",
"mr" : "mister",
"mrw" : "my reaction when",
"ms" : "miss",
"mte" : "my thoughts exactly",
"nagi" : "not a good idea",
"nbc" : "national broadcasting company",
"nbd" : "not big deal",
"nfs" : "not for sale",
"ngl" : "not going to lie",
"nhs" : "national health service",
"nrn" : "no reply necessary",
"nsfl" : "not safe for life",
"nsfw" : "not safe for work",
"nth" : "nice to have",
"nvr" : "never",
"nyc" : "new york city",
"oc" : "original content",
"og" : "original",
"ohp" : "overhead projector",
"oic" : "oh i see",
"omdb" : "over my dead body",
"omg" : "oh my god",
"omw" : "on my way",
"p.a" : "per annum",
"p.m" : "after midday",
"pm" : "prime minister",
"poc" : "people of color",
"pov" : "point of view",
"pp" : "pages",
"ppl" : "people",
"prw" : "parents are watching",
"ps" : "postscript",
"pt" : "point",
"ptb" : "please text back",
"pto" : "please turn over",
"qpsa" : "what happens",
"ratchet" : "rude",
"rbtl" : "read between the lines",
"rlrt" : "real life retweet",
"rofl" : "rolling on the floor laughing",
"roflol" : "rolling on the floor laughing out loud",
"rotflmao" : "rolling on the floor laughing my ass off",
"rt" : "retweet",
"ruok" : "are you ok",
"sfw" : "safe for work",

```

    "sk8" : "skate",
    "smh" : "shake my head",
    "sq" : "square",
    "srsly" : "seriously",
    "ssdd" : "same stuff different day",
    "tbh" : "to be honest",
    "tbs" : "tablespoonful",
    "tbsp" : "tablespoonful",
    "tfw" : "that feeling when",
    "thks" : "thank you",
    "tho" : "though",
    "thx" : "thank you",
    "tia" : "thanks in advance",
    "til" : "today i learned",
    "tl;dr" : "too long i did not read",
    "tldr" : "too long i did not read",
    "tmb" : "tweet me back",
    "tntl" : "trying not to laugh",
    "ttyl" : "talk to you later",
    "u" : "you",
    "u2" : "you too",
    "u4e" : "yours for ever",
    "utc" : "coordinated universal time",
    "w/" : "with",
    "w/o" : "without",
    "w8" : "wait",
    "wassup" : "what is up",
    "wb" : "welcome back",
    "wtf" : "what the fuck",
    "wtg" : "way to go",
    "wtpa" : "where the party at",
    "wuf" : "where are you from",
    "wuzup" : "what is up",
    "wywh" : "wish you were here",
    "yd" : "yard",
    "ygtr" : "you got that right",
    "ynk" : "you never know",
    "zzz" : "sleeping bored and tired"
}

```

```

[ ]: def convert_abbrev_in_text(tweet):
    t=[]
    words=tweet.split()
    t = [abbreviations[w.lower()] if w.lower() in abbreviations.keys() else w
    ↪for w in words]
    return ' '.join(t)

```

Text processing completed

```
[ ]: tweets['processed_tweets'] = tweets['tweet'].apply(lambda x: process_tweets(x))
tweets['processed_tweets'] = tweets['processed_tweets'].apply(lambda x:
↳convert_abbrev_in_text(x))
print('Text Preprocessing complete.')
tweets
```

Text Preprocessing complete.

```
[ ]:      sentiment                                tweet \
807355      1      @SwandaG if you find one, please pass it on
714751      0                                its like im always sick
656892      0  @OfficialBB i dont want either of them to sre...
1164846      1      @KtotheCdawg omg yes  thatd be heaps cool
1571157      1                                Ok... going to bed.
...
1369467      1  Looking forward to 100 followers on Follow Fri...
246461      0  @Wordlily I&quot;ve done many of those things ...
1105450      1                                more new followerscan they keep up
870490      1                                A cup of tea cures emoness
1549944      1  @therealsavannah http://twitpic.com/7hv1u - we...

                                processed_tweets
807355                                swandag find one please pas
714751                                t like im always sick
656892      officialbb dont want either sree neeeeds go
1164846      ktothecdawg oh my god yes thatd heap cool
1571157                                going bed
...
1369467  ooking forward 100 follower follow friday happ...
246461  wordlily iquoteve done many thing mostly wantin...
1105450                                ore new followerscan keep
870490                                cup tea cure emoness
1549944      therealsavannah weird definitely funny

[200000 rows x 3 columns]
```

```
[ ]: #removing shortwords
tweets['processed_tweets']=tweets['processed_tweets'].apply(lambda x: " ".
↳join([w for w in x.split() if len(w)>3]))
tweets.head(5)
```

```
[ ]:      sentiment                                tweet \
807355      1      @SwandaG if you find one, please pass it on
714751      0                                its like im always sick
656892      0  @OfficialBB i dont want either of them to sre...
1164846      1      @KtotheCdawg omg yes  thatd be heaps cool
```

| | | |
|---------|------------|-------------------------------|
| 1571157 | 1 | Ok... going to bed. |
| | | processed_tweets |
| 807355 | | swandag find please |
| 714751 | | like always sick |
| 656892 | officialbb | dont want either sree neeeeds |
| 1164846 | | ktothecdawg thatd heap cool |
| 1571157 | | going |

Now lets split the data into a training set and a test set using scikit-learns `train_test_split` function https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

```
[ ]: tweets_data = tweets["processed_tweets"]
      tweets_labels = tweets["sentiment"]

      #Split data into train_tweets, test_tweets, train_labels and test_labels
      train_tweets, test_tweets, train_labels, test_labels = \
          train_test_split(tweets_data, tweets_labels)
```

What we need to build our classifier is “probability of positive tweet” $P(\text{pos})$, “probability of negative tweet” $P(\text{neg})$, “probability of word in tweet given tweet is positive” $P(w|\text{pos})$ and “probability of word in tweet given tweet is negative” $P(w|\text{neg})$. Start by calculating the probability that a tweet is positive and negative respectively

```
[ ]: pos, neg = train_labels.value_counts()
      P_pos = pos / len(train_labels)
      P_neg = neg / len(train_tweets)
```

For $P(w|\text{pos})$, $P(w|\text{neg})$ we need to count how many tweets each word occur in. Count the number of tweets each word occurs in and store in the word counter. An entry in the word counter is for instance `{‘good’: ‘Pos’:150, ‘Neg’: 10}` meaning good occurs in 150 positive tweets and 10 negative tweets. Be aware that we are not interested in calculating multiple occurrences of the same word in the same tweet. Also we change the labels from 0 for “Negative” and 1 for “Positive” to “Neg” and “Pos” respectively. For each word convert it to lower case. You can use Python’s `lower`. Another handy Python string method is `split`.

```
[ ]: new_train_labels = train_labels.replace(0, "Neg", regex=True)
      final_train_labels = new_train_labels.replace(1, "Pos", regex=True)
      word_counter = {}
      for (tweet, label) in zip(train_tweets, final_train_labels):
          tweet = list(set(tweet.split()))
          for word in tweet:
              word = word.lower()
              if word in word_counter:
                  word_counter[word][label] += 1
              else:
                  word_counter[word] = {"Pos":0, "Neg":0}
                  word_counter[word][label] +=1
```

```
# ... Count number of tweets each word occurs in and store in word_counter
↳where an entry looks like ex. {'word': 'Pos':98, 'Neg':10}
```

Let's work with a smaller subset of words just to save up some time. Find the 1500 most occurring words in tweet data.

```
[ ]: nr_of_words_to_use = 1500
popular_words = sorted(word_counter.items(), key=lambda x: x[1]['Pos'] +
↳x[1]['Neg'], reverse=True)
popular_words = [x[0] for x in popular_words[:nr_of_words_to_use]]
```

Now lets compute $P(w|pos)$, $P(w|neg)$ for the popular words

```
[ ]: P_w_given_pos = {}
P_w_given_neg = {}
total_words = 0
for word in popular_words:
    total_words += word_counter[word]["Pos"] + word_counter[word]["Neg"]
for word in popular_words:
    word_occurance = word_counter[word]["Pos"] + word_counter[word]["Neg"]

    P_pos_given_word = word_counter[word]["Pos"] / word_occurance
    P_neg_given_word = word_counter[word]["Neg"] / word_occurance
    P_word = word_occurance / total_words

    P_w_given_pos[word] = (P_pos_given_word * P_pos) / P_pos
    P_w_given_neg[word] = (P_neg_given_word * P_neg) / P_neg

    # Calculate the two probabilities
```

```
[ ]: classifier = {
    'basis' : popular_words,
    'P(pos)' : P_pos,
    'P(neg)' : P_neg,
    'P(w|pos)' : P_w_given_pos,
    'P(w|neg)' : P_w_given_neg
}
```

Train and predict Write a `tweet_classifier` function that takes your trained classifier and a tweet and returns whether it's about Positive or Negative using the popular words selected. Note that if there are words in the basis words in our classifier that are not in the tweet we have the opposite probabilities i.e $P(w_1 \text{ occurs}) * P(w_2 \text{ does not occur}) * \dots$ if w_1 occurs and w_2 does not occur. The function should return whether the tweet is Positive or Negative. i.e 'Pos' or 'Neg'.

```
[ ]: def tweet_classifier(tweet, classifier_dict):
    """ param tweet: string containing tweet message
```



```

        param classifier: dict containing 'basis' - training words
                                'P(pos)' - class probabilities
                                'P(neg)' - class probabilities
                                'P(w|pos)' - conditional probabilities
                                'P(w|neg)' - conditional probabilities

        return: either 'Pos' or 'Neg'
    """
    prob_pos = classifier_dict["P(pos)"]
    prob_neg = classifier_dict["P(neg)"]
    tweet = tweet.split()
    for word in tweet:
        if word in classifier_dict["basis"]:
            prob_pos *= classifier_dict["P(w|pos)"][word]
            prob_neg *= classifier_dict["P(w|neg)"][word]

    return "Neg" if prob_pos < prob_neg else "Pos"
# ... Code for classifying tweets using the naive bayes classifier

```

```

[ ]: def test_classifier(classifier, test_tweets, test_labels):
    total = len(test_tweets)
    correct = 0
    for (tweet,label) in zip(test_tweets, test_labels):
        predicted = tweet_classifier(tweet,classifier)
        if predicted == label:
            correct = correct + 1
    return(correct/total)

```

```

[ ]: new_test_labels = test_labels.replace(0, "Neg", regex=True)
    final_test_labels = new_test_labels.replace(1, "Pos", regex=True)

```

```

[ ]: acc = test_classifier(classifier, test_tweets, final_test_labels)
    print(acc)
    print(f"Accuracy: {acc:.4f}")

```

0.71674

Accuracy: 0.7167

Optional work In basic sentiment analysis classifications we have 3 classes “Positive”, “Negative” and “Neutral”. Although because it is challenging to create the “Neutral” class. Try to improve the accuracy by filtering the dataset from the perspective of removing words that indicate neutrality.

```

[ ]: #This is an attempt, doesn't seem to do much better

classifier2 = classifier.copy()
classifier2["basis"] = classifier["basis"].copy()

```

```

for word in classifier2["basis"]:
    word_pos = classifier2["P(w|pos)"][word] * classifier2["P(pos)"]
    word_neg = classifier2["P(w|neg)"][word] * classifier2["P(neg)"]
    if word_pos <= (word_neg + word_neg*0.2) and word_pos > (word_neg -
↪word_neg*0.2):
        classifier2["basis"].remove(word)

acc = test_classifier(classifier2, test_tweets, final_test_labels)
print(acc)
print(f"Accuracy: {acc:.4f}")

```

0.71846

Accuracy: 0.7185