Aarifah Ullah

Prof. Kasia Hitczenko

CSCI 4907.82

3 February 2024

CSCI 4907.82 Natural Language Understanding Homework 1

Part 1: Preparing Data

1. What are the dimensions of your feature matrix?
    a. The matrix is 5000 X 9084, meaning there are 5000 tweets and 9084 unique words that appear within those tweets. This makes sense as the csv file is after all named "Tweets_5K.csv"
2. What is the value of X[1460][1460]?
    a. The value is 0.
3. What does this value mean? What feature does the 1460th column represent?
    a. This value is the count of the 1460th word in the matrix in the 1460th document of the matrix. So, the 1460th word is "bristol" and we are asking how many times it appears in document 1460. This is 0 because that document says, "Happy Mother's Day!"

Part 2: Implementing Naive Bayes

1. Calculate whether the Naive Bayes classifier you trained would classify the following tweet as positive, neutral, or negative: "Happy birthday Annemarie". You should do this by hand and show your work, but you can use code to get the relevant probabilities (basically, the goal of this question is to make sure that you understand how Naive Bayes works)
    a.
2. Report the model's accuracy on the unseen test set. How does this compare to a classifier that always outputs the most frequent category in the training set (what is that classifier's accuracy)?
    a. The naive bayes with just the preprocessing steps had an F1 score of .5554. An F1 score closer to 1 indicates a very good classifier while closer to zero is a poor classifier. The Naive Bayes classifier seems to do okay-ish.

       The accuracy is .54575 or 54.575%, meaning the classifier gets the right answer about half the time. Compared to a classifier that just always outputs the most frequent category, this Naive Bayes classifier has a lower accuracy. This is because the other classifier just answers the most frequent class and is right about most of the time, but it is a poor classifier since it does not know much about the data. (It's precision and recall are perhaps very low).

Part 3: Implementing Logistic Regression

1. In class, we learned how to do binary logistic regression between 2 options. Here, there are three possible classifications, so sklearn uses a "one vs. rest" scheme where it learns a

binary logistic regression model for each possible label (i.e., one logistic regression which learns to separate positive from non-positive tweets, a second that learns to separate negative from non-negative tweets, and a final logistic regression that learns to separate neutral from non-neutral tweets). How many parameters did your multiclass logistic regression model learn?

    a. Parameters and features are different. Parameters refers to the document's attributes that set it apart and help to place that document into groups known as features. Because we implemented a bag of words representation, the words themselves are what defines the documents, as in the document can be described by all the words it has. So, in a binary logistic model using the bag of words representation, it learns the whole list of words as the parameters. Because the multinomial logistic regression model employed binary logistic regression model three times (positive v. rest, negative v. rest, neutral v. rest), it still learned the same number of parameters, but repeated its steps three times.

2. Report the model's accuracy on the unseen test set. How does this compare to a classifier that always outputs the most frequent category in the training set and the Naive Bayes classifier from Part 2?
    a. The logistic regression model has an F1 score of 55.54% and an accuracy of .58 or 58%, meaning that the model correctly assesses the class 58% of the time. This is a little better than the naive bayes classifier that had an accuracy rate of 54.575% using just basic pre-processing steps and the bag of words model. Compared toa classifier that outputs the most frequent category, the logistic regression model is less accurate for the same reasons mentioned when describing the naive bayes classifier. The other classifier is not very good but it has a high accuracy rate when it guesses the most frequent class all the time.

Part 4: Implementing More Elaborate Pre-Processing

1. How does this impact the performance of your Naive Bayes and Logistic Regression classification results?
    a. More pre-processing improves results. After implementing each processing step, I re-ran both models and observed how both the accuracy and F1 score improved. The final accuracy and logistic numbers are

Part 5: Your turn!

1. Propose and implement at least one addition to the workflow that you think will improve performance (but please still use Naive Bayes or Logistic Regression). Specifically, you can add a pre-processing step, add a feature (or class of features), etc. Explain why you chose this feature and why you think it might help performance. Report the classifier's accuracy on the test set. Did this help as expected? Note: You will not be graded on whether your change actually improved performance.
    a. My suggestion to improve performance is to make a separate list of stop words that add onto Spacy's standard list of stop words. This would help eliminate words more related to tweets that are perhaps taking up space in the 1000 most frequent words lists and are not giving us clues on the actual sentiment.

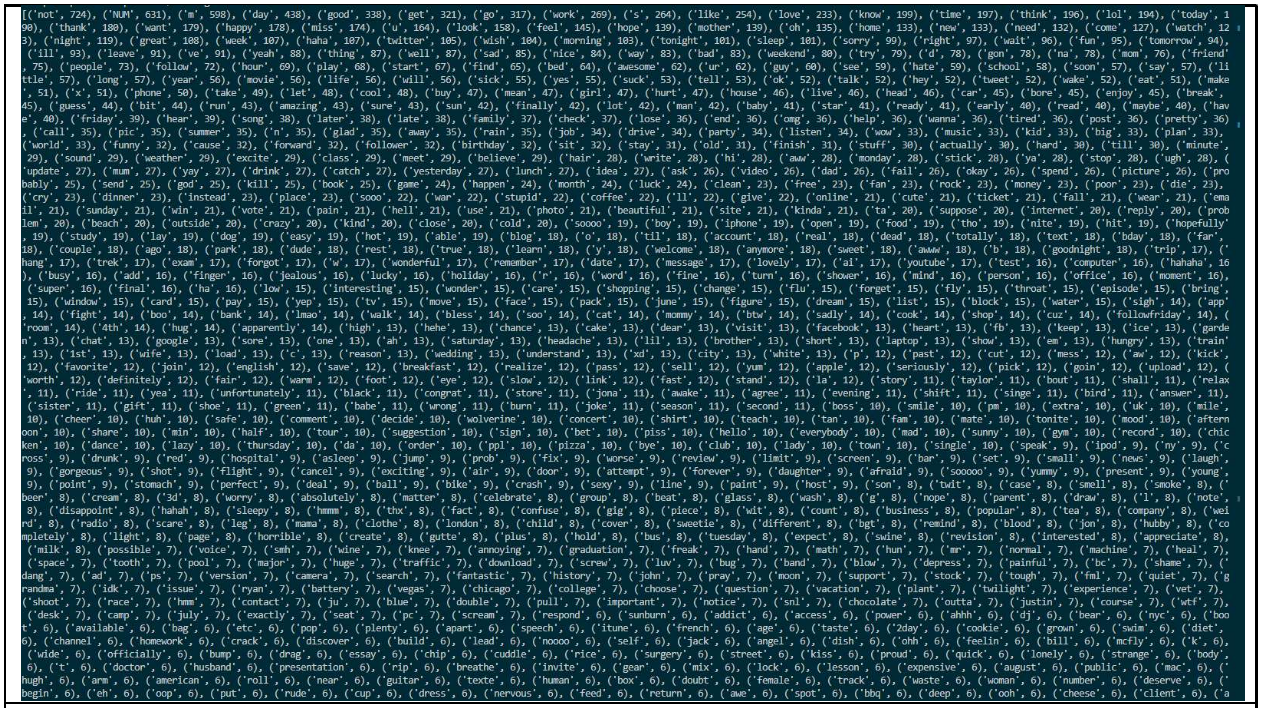
b. After generating a list of the 1000 most frequent words that occur after more elaborate pre-processing including removal of stop words and lemmatization, it is clear to see that some "words" remain.

c. For example:

   i. "ve" is not a word but most likely a conjunction that got separated from the root word with the removal of punctuation "ve" is one of the most frequent words, in the top of the list with a count of 91.

   ii. "s" ~ 264, "m" ~ 598 most likely from "I'm," "d" ~78

d. I made a custom list of common leftover words I saw and include that along with my stop word function on line 103.I also used CountVectorizer()'s own stopword feature to handle any missed word .isstop did not handle.

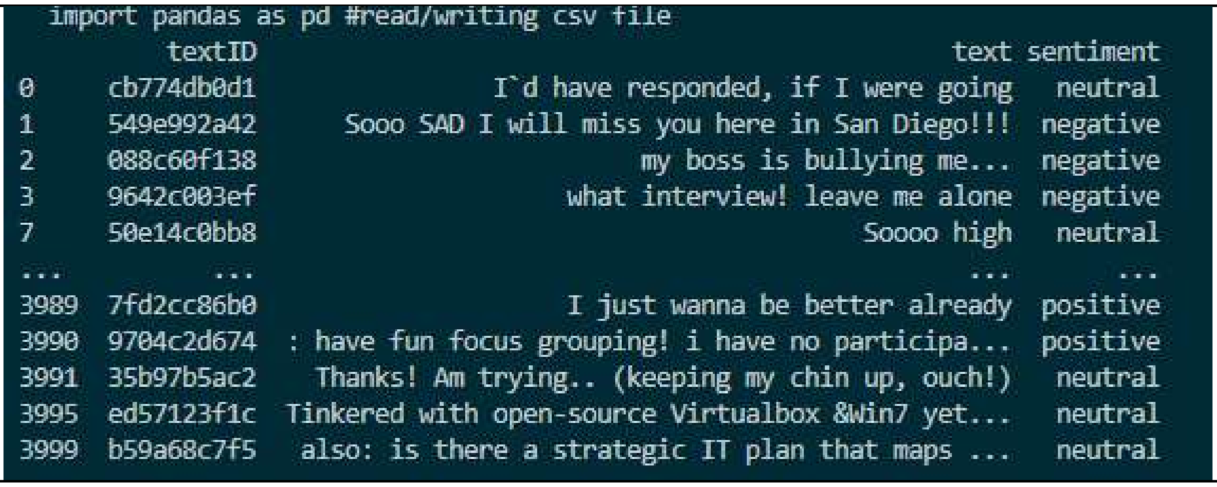


```
[('not', 724), ('NUM', 631), ('m', 598), ('day', 438), ('good', 338), ('get', 321), ('go', 317), ('work', 269), ('s', 264), ('like', 254), ('love', 233), ('know', 199), ('time', 197), ('think', 196), ('lol', 194), ('today', 1
90), ('thank', 180), ('want', 179), ('happy', 178), ('miss', 174), ('u', 164), ('look', 158), ('feel', 145), ('hope', 139), ('mother', 139), ('oh', 135), ('home', 133), ('new', 133), ('need', 132), ('come', 127), ('watch', 12
3), ('night', 119), ('great', 108), ('week', 107), ('haha', 107), ('twitter', 105), ('wish', 104), ('morning', 103), ('tonight', 101), ('sleep', 101), ('sorry', 99), ('right', 97), ('wait', 96), ('fun', 95), ('tomorrow', 94),
 ('ill', 93), ('leave', 91), ('ve', 91), ('yeah', 88), ('thing', 87), ('well', 87), ('sad', 85), ('nice', 84), ('way', 83), ('bad', 83), ('weekend', 80), ('try', 79), ('d', 78), ('gon', 78), ('na', 78), ('mom', 76), ('friend'
, 75), ('people', 73), ('follow', 72), ('hour', 69), ('play', 68), ('start', 67), ('find', 65), ('bed', 64), ('awesome', 62), ('ur', 62), ('guy', 60), ('see', 59), ('hate', 59), ('school', 58), ('soon', 57), ('say', 57), ('li
ttle', 57), ('long', 57), ('year', 56), ('movie', 56), ('life', 56), ('will', 56), ('sick', 55), ('yes', 55), ('suck', 53), ('tell', 53), ('ok', 52), ('talk', 52), ('hey', 52), ('tweet', 52), ('wake', 52), ('eat', 51), ('make
', 51), ('x', 51), ('phone', 50), ('take', 49), ('let', 48), ('cool', 48), ('buy', 47), ('mean', 47), ('girl', 47), ('hurt', 47), ('house', 46), ('live', 46), ('head', 46), ('car', 45), ('bore', 45), ('enjoy', 45), ('break',
45), ('guess', 44), ('bit', 44), ('run', 43), ('amazing', 43), ('sure', 43), ('sun', 42), ('finally', 42), ('lot', 42), ('man', 42), ('baby', 41), ('star', 41), ('ready', 41), ('early', 40), ('read', 40), ('maybe', 40), ('hav
e', 40), ('friday', 39), ('hear', 39), ('song', 38), ('later', 38), ('late', 38), ('family', 37), ('check', 37), ('lose', 36), ('end', 36), ('omg', 36), ('help', 36), ('wanna', 36), ('tired', 36), ('post', 36), ('pretty', 36)
, ('call', 35), ('pic', 35), ('summer', 35), ('n', 35), ('glad', 35), ('away', 35), ('rain', 35), ('job', 34), ('drive', 34), ('party', 34), ('listen', 34), ('wow', 33), ('music', 33), ('kid', 33), ('big', 33), ('plan', 33),
('world', 33), ('funny', 32), ('cause', 32), ('forward', 32), ('follower', 32), ('birthday', 32), ('sit', 32), ('stay', 31), ('old', 31), ('finish', 31), ('stuff', 30), ('actually', 30), ('hard', 30), ('till', 30), ('minute',
 29), ('sound', 29), ('weather', 29), ('excite', 29), ('class', 29), ('meet', 29), ('believe', 29), ('hair', 28), ('write', 28), ('hi', 28), ('aww', 28), ('monday', 28), ('stick', 28), ('ya', 28), ('stop', 28), ('ugh', 28), (
'update', 27), ('mum', 27), ('yay', 27), ('drink', 27), ('catch', 27), ('yesterday', 27), ('lunch', 27), ('idea', 27), ('ask', 26), ('video', 26), ('dad', 26), ('fail', 26), ('okay', 26), ('spend', 26), ('picture', 26), ('pro
bably', 25), ('send', 25), ('god', 25), ('kill', 25), ('book', 25), ('game', 24), ('happen', 24), ('month', 24), ('luck', 24), ('clean', 23), ('free', 23), ('fan', 23), ('rock', 23), ('money', 23), ('poor', 23), ('die', 23),
('cry', 23), ('dinner', 23), ('instead', 23), ('place', 23), ('sooo', 22), ('war', 22), ('stupid', 22), ('coffee', 22), ('ll', 22), ('give', 22), ('online', 21), ('cute', 21), ('ticket', 21), ('fall', 21), ('wear', 21), ('ema
il', 21), ('sunday', 21), ('win', 21), ('vote', 21), ('pain', 21), ('hell', 21), ('use', 21), ('photo', 21), ('beautiful', 21), ('site', 21), ('kinda', 21), ('ta', 20), ('suppose', 20), ('internet', 20), ('reply', 20), ('prob
lem', 20), ('beach', 20), ('outside', 20), ('crazy', 20), ('kind', 20), ('close', 20), ('cold', 20), ('soooo', 19), ('boy', 19), ('iphone', 19), ('open', 19), ('food', 19), ('tho', 19), ('nite', 19), ('hit', 19), ('hopefully'
, 19), ('study', 19), ('lay', 19), ('dog', 19), ('easy', 19), ('hot', 19), ('able', 19), ('blog', 18), ('o', 18), ('til', 18), ('account', 18), ('real', 18), ('dead', 18), ('totally', 18), ('text', 18), ('bday', 18), ('far',
18), ('couple', 18), ('ago', 18), ('park', 18), ('dude', 18), ('rest', 18), ('true', 18), ('learn', 18), ('y', 18), ('welcome', 18), ('anymore', 18), ('sweet', 18), ('awww', 18), ('b', 18), ('goodnight', 18), ('trip', 17), ('
hang', 17), ('trek', 17), ('exam', 17), ('forgot', 17), ('w', 17), ('wonderful', 17), ('remember', 17), ('date', 17), ('message', 17), ('lovely', 17), ('ai', 17), ('youtube', 17), ('test', 16), ('computer', 16), ('hahaha', 16
), ('busy', 16), ('add', 16), ('finger', 16), ('jealous', 16), ('lucky', 16), ('holiday', 16), ('r', 16), ('word', 16), ('fine', 16), ('turn', 16), ('shower', 16), ('mind', 16), ('person', 16), ('office', 16), ('moment', 16),
 ('super', 16), ('final', 16), ('ha', 16), ('low', 15), ('interesting', 15), ('wonder', 15), ('care', 15), ('shopping', 15), ('change', 15), ('flu', 15), ('forget', 15), ('fly', 15), ('throat', 15), ('episode', 15), ('bring',
 15), ('window', 15), ('card', 15), ('pay', 15), ('yep', 15), ('tv', 15), ('move', 15), ('face', 15), ('pack', 15), ('june', 15), ('figure', 15), ('dream', 15), ('list', 15), ('block', 15), ('water', 15), ('sigh', 14), ('app'
, 14), ('fight', 14), ('boo', 14), ('bank', 14), ('lmao', 14), ('walk', 14), ('bless', 14), ('soo', 14), ('cat', 14), ('mommy', 14), ('btw', 14), ('sadly', 14), ('cook', 14), ('shop', 14), ('cuz', 14), ('followfriday', 14), (
'room', 14), ('4th', 14), ('hug', 14), ('apparently', 14), ('high', 13), ('hehe', 13), ('chance', 13), ('cake', 13), ('dear', 13), ('visit', 13), ('facebook', 13), ('heart', 13), ('fb', 13), ('keep', 13), ('ice', 13), ('garde
n', 13), ('chat', 13), ('google', 13), ('sore', 13), ('one', 13), ('ah', 13), ('saturday', 13), ('headache', 13), ('lil', 13), ('brother', 13), ('short', 13), ('laptop', 13), ('show', 13), ('em', 13), ('hungry', 13), ('train'
, 13), ('1st', 13), ('wife', 13), ('load', 13), ('c', 13), ('reason', 13), ('wedding', 13), ('understand', 13), ('xd', 13), ('city', 13), ('white', 13), ('p', 12), ('past', 12), ('cut', 12), ('mess', 12), ('aw', 12), ('kick',
 12), ('favorite', 12), ('join', 12), ('english', 12), ('save', 12), ('breakfast', 12), ('realize', 12), ('pass', 12), ('sell', 12), ('yum', 12), ('apple', 12), ('seriously', 12), ('pick', 12), ('goin', 12), ('upload', 12), (
'worth', 12), ('definitely', 12), ('fair', 12), ('warm', 12), ('foot', 12), ('eye', 12), ('slow', 12), ('link', 12), ('fast', 12), ('stand', 12), ('la', 12), ('story', 11), ('taylor', 11), ('bout', 11), ('shall', 11), ('relax
', 11), ('ride', 11), ('yea', 11), ('unfortunately', 11), ('black', 11), ('congrat', 11), ('store', 11), ('jona', 11), ('awake', 11), ('agree', 11), ('evening', 11), ('shift', 11), ('singe', 11), ('bird', 11), ('answer', 11),
 ('sister', 11), ('gift', 11), ('shoe', 11), ('green', 11), ('babe', 11), ('wrong', 11), ('burn', 11), ('joke', 11), ('season', 11), ('second', 11), ('boss', 10), ('smile', 10), ('pm', 10), ('extra', 10), ('uk', 10), ('mile',
 10), ('cheer', 10), ('huh', 10), ('safe', 10), ('comment', 10), ('decide', 10), ('wolverine', 10), ('concert', 10), ('shirt', 10), ('teach', 10), ('tan', 10), ('fam', 10), ('mate', 10), ('tonite', 10), ('mood', 10), ('aftern
oon', 10), ('share', 10), ('min', 10), ('half', 10), ('tour', 10), ('suggestion', 10), ('sign', 10), ('bet', 10), ('piss', 10), ('hello', 10), ('everybody', 10), ('mad', 10), ('sunny', 10), ('gym', 10), ('record', 10), ('chic
ken', 10), ('dance', 10), ('lazy', 10), ('thursday', 10), ('da', 10), ('order', 10), ('ppl', 10), ('pizza', 10), ('bye', 10), ('club', 10), ('lady', 10), ('town', 10), ('single', 10), ('speak', 9), ('ipod', 9), ('ny', 9), ('c
ross', 9), ('drunk', 9), ('red', 9), ('hospital', 9), ('asleep', 9), ('jump', 9), ('prob', 9), ('fix', 9), ('worse', 9), ('review', 9), ('limit', 9), ('screen', 9), ('bar', 9), ('set', 9), ('small', 9), ('news', 9), ('laugh',
 9), ('gorgeous', 9), ('shot', 9), ('flight', 9), ('cancel', 9), ('exciting', 9), ('air', 9), ('door', 9), ('attempt', 9), ('forever', 9), ('daughter', 9), ('afraid', 9), ('sooooo', 9), ('yummy', 9), ('present', 9), ('young',
 9), ('point', 9), ('stomach', 9), ('perfect', 9), ('deal', 9), ('ball', 9), ('bike', 9), ('crash', 9), ('sexy', 9), ('line', 9), ('paint', 9), ('host', 9), ('son', 8), ('twit', 8), ('case', 8), ('smell', 8), ('smoke', 8), ('
beer', 8), ('cream', 8), ('3d', 8), ('worry', 8), ('absolutely', 8), ('matter', 8), ('celebrate', 8), ('group', 8), ('beat', 8), ('glass', 8), ('wash', 8), ('g', 8), ('nope', 8), ('parent', 8), ('draw', 8), ('l', 8), ('note',
 8), ('disappoint', 8), ('hahah', 8), ('sleepy', 8), ('hmmm', 8), ('thx', 8), ('fact', 8), ('confuse', 8), ('gig', 8), ('piece', 8), ('wit', 8), ('count', 8), ('business', 8), ('popular', 8), ('tea', 8), ('company', 8), ('wei
rd', 8), ('radio', 8), ('scare', 8), ('leg', 8), ('mama', 8), ('clothe', 8), ('london', 8), ('child', 8), ('cover', 8), ('sweetie', 8), ('different', 8), ('bgt', 8), ('remind', 8), ('blood', 8), ('jon', 8), ('hubby', 8), ('co
mpletely', 8), ('light', 8), ('page', 8), ('horrible', 8), ('create', 8), ('gutte', 8), ('plus', 8), ('hold', 8), ('bus', 8), ('tuesday', 8), ('expect', 8), ('swine', 8), ('revision', 8), ('interested', 8), ('appreciate', 8),
 ('milk', 8), ('possible', 7), ('voice', 7), ('smh', 7), ('wine', 7), ('knee', 7), ('annoying', 7), ('graduation', 7), ('freak', 7), ('hand', 7), ('math', 7), ('hun', 7), ('mr', 7), ('normal', 7), ('machine', 7), ('heal', 7),
 ('space', 7), ('tooth', 7), ('pool', 7), ('major', 7), ('huge', 7), ('traffic', 7), ('download', 7), ('screw', 7), ('luv', 7), ('bug', 7), ('band', 7), ('blow', 7), ('depress', 7), ('painful', 7), ('bc', 7), ('shame', 7), ('
dang', 7), ('ad', 7), ('ps', 7), ('version', 7), ('camera', 7), ('search', 7), ('fantastic', 7), ('history', 7), ('john', 7), ('pray', 7), ('moon', 7), ('support', 7), ('stock', 7), ('tough', 7), ('fml', 7), ('quiet', 7), ('g
randma', 7), ('idk', 7), ('issue', 7), ('ryan', 7), ('battery', 7), ('vegas', 7), ('chicago', 7), ('college', 7), ('choose', 7), ('question', 7), ('vacation', 7), ('plant', 7), ('twilight', 7), ('experience', 7), ('vet', 7),
('shoot', 7), ('race', 7), ('hmm', 7), ('contact', 7), ('ju', 7), ('blue', 7), ('double', 7), ('pull', 7), ('important', 7), ('notice', 7), ('snl', 7), ('chocolate', 7), ('outta', 7), ('justin', 7), ('course', 7), ('wtf', 7),
 ('desk', 7), ('camp', 7), ('july', 7), ('exactly', 7), ('seat', 7), ('pc', 7), ('scream', 7), ('respond', 6), ('sunburn', 6), ('addict', 6), ('access', 6), ('power', 6), ('ahhh', 6), ('dj', 6), ('bear', 6), ('nyc', 6), ('boo
t', 6), ('available', 6), ('bag', 6), ('etc', 6), ('pop', 6), ('plenty', 6), ('apart', 6), ('speech', 6), ('itune', 6), ('french', 6), ('age', 6), ('taste', 6), ('2day', 6), ('cookie', 6), ('grown', 6), ('swim', 6), ('diet',
6), ('channel', 6), ('homework', 6), ('crack', 6), ('discover', 6), ('build', 6), ('lead', 6), ('noooo', 6), ('self', 6), ('jack', 6), ('angel', 6), ('dish', 6), ('ohh', 6), ('feelin', 6), ('bill', 6), ('mcfly', 6), ('k', 6),
 ('wide', 6), ('officially', 6), ('bump', 6), ('drag', 6), ('essay', 6), ('chip', 6), ('cuddle', 6), ('rice', 6), ('surgery', 6), ('street', 6), ('kiss', 6), ('proud', 6), ('quick', 6), ('lonely', 6), ('strange', 6), ('body',
 6), ('t', 6), ('doctor', 6), ('husband', 6), ('presentation', 6), ('rip', 6), ('breathe', 6), ('invite', 6), ('gear', 6), ('mix', 6), ('lock', 6), ('lesson', 6), ('expensive', 6), ('august', 6), ('public', 6), ('mac', 6), ('
hugh', 6), ('arm', 6), ('american', 6), ('roll', 6), ('near', 6), ('guitar', 6), ('texte', 6), ('human', 6), ('box', 6), ('doubt', 6), ('female', 6), ('track', 6), ('waste', 6), ('woman', 6), ('number', 6), ('deserve', 6), ('
begin', 6), ('eh', 6), ('oop', 6), ('put', 6), ('rude', 6), ('cup', 6), ('dress', 6), ('nervous', 6), ('feed', 6), ('return', 6), ('awe', 6), ('spot', 6), ('bbq', 6), ('deep', 6), ('ooh', 6), ('cheese', 6), ('client', 6), ('a
```

1000 Most Frequent Words with Their Counts

2. Finally, take the best performing model and look at the tweets that were incorrectly categorized. Do you observe any patterns in what the model is making mistakes on? What do you think could be done to further improve results? Provide a sample of tweets as well as their true label and their predicted label to justify your answers. Note: You do not need to know how to implement the proposed change.

   a. Logistic regression performed the best with an estimated accuracy of ~60%. It seems to wrongly classify texts that are difficult to tell as a person. It would depend on the context. It also got wrong texts with words and phrases that are used in other classes as well. For example "soooooo..." can be used in a positive sense, but in line 1, the logistic regression model didn't realize it was meant in a negative sense. I would suggest working with more data to improve the results.



```
 import pandas as pd #read/writing csv file
        textID                                                text  sentiment
0      cb774db0d1                 I`d have responded, if I were going    neutral
1      549e992a42       Sooo SAD I will miss you here in San Diego!!!   negative
2      088c60f138                           my boss is bullying me...   negative
3      9642c003ef                      what interview! leave me alone   negative
7      50e14c0bb8                                          Soooo high    neutral
...           ...                                                 ...        ...
3989   7fd2cc86b0                      I just wanna be better already   positive
3990   9704c2d674    : have fun focus grouping! i have no participa...   positive
3991   35b97b5ac2     Thanks! Am trying.. (keeping my chin up, ouch!)    neutral
3995   ed57123f1c  Tinkered with open-source Virtualbox &Win7 yet...    neutral
3999   b59a68c7f5    also: is there a strategic IT plan that maps ...    neutral
```

List of Tweets Logistic Regression Incorrectly Classified

3. Roughly how much time did you spend on this homework?
    a. I have spent about 16 hours completing this homework. I haven't coded in a bit, so it took me some time to set up my computer, including downloading python, etc. I also had some trouble downloading the necessary NLP libraries.

	Naive Bayes		Logisitic Regression	
	Accuracy	F1	Accuracy	F1
Pre-Processing	54.575%	55.54%	58%	55.54%
More Elaborate Pre-Processing	~57.35%	57.38%	59.625%	57.83%
My Own Implemented Suggestion	56.85%	56.93%	58.65%	56.93%

Consulted Sources

When implementing the Bag of Words matrix:

- https://spotintelligence.com/2023/05/17/countvectorizer/
- https://spotintelligence.com/2022/12/20/bag-of-words-python/

Splitting Training and Testing sets:

- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
- https://www.sharpsightlabs.com/blog/scikit-train_test_split/

Naive Bayes Implementation:

- https://scikit-learn.org/stable/modules/naive_bayes.html
- https://www.datacamp.com/tutorial/naive-bayes-scikit-learn
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

Logistic Regression Implementation:

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- https://towardsdatascience.com/logistic-regression-using-python-sklearn-numpy-mnist-handwriting-recognition-matplotlib-a6b31e2b166a

More Pre-Processing:

- https://www.digitalocean.com/community/tutorials/python-remove-spaces-from-string
- https://blog.enterprisedna.co/python-remove-punctuation-from-string/
- https://www.geeksforgeeks.org/python-pos-tagging-and-lemmatization-using-spacy/
- https://realpython.com/natural-language-processing-spacy-python/
- Basics of CountVectorizer | by Pratyaksh Jain | Towards Data Science