Names: Tyler Hudson (tkh51)

0) Make sure you have nltk and pandas installed.

Make sure you ran nltk.download already run and installed all. These were done in processingstringandwordclouds.py

Go to naivebayes.py

Open up conda prompt in a windows OS or terminal if you are using mac.

pip install scikit-learn

pip install spacy

pip install the long https line that is given

Also run the python -m spacy download en\_core\_web\_sm line from conda prompt

1) Obviously run the importations and lemmatizer tokenizer and stop\_words objects on the first 20 lines of NaiveBayes.py

I do not need you to copy/paste anything from questions 0 and 1, they are guidelines.

2) Import traintextMining.csv into QuoraQ object. Copy/paste the script.

QuoraQ = pd.read\_csv("C:/Users/tyler/OneDrive/Documents/school2k20/Fall2020/CIS3339/Data/traintextMining.csv", sep= ',')

3) Use the tokenizer function as I have done. Create the cv object with the countvectorizer function. Create the stopwords object sw. Use BernoulliNB() to create the object naive\_bayes as I have done on NaiveBayes.py

Use the fit and predict function to modify it as I have done. Create the y\_train object as I have done that is going to have the target 0 and 1 values. Use cv.fit\_transform to create the X\_train\_cv object. Copy/paste the script that does this (aside from your explanation of course).

Explain what X\_train\_cv is and represents.

def my\_tokenizer(doc):

tokens = lemmatizer(doc)

[token.lemma\_ for token in tokens]

return tokenizer.tokenize(str(tokens))

sw=my\_tokenizer(str(stop\_words))

cv = CountVectorizer(lowercase=True, stop\_words=sw,binary=True,tokenizer=my\_tokenizer)

X\_train\_cv = cv.fit\_transform(QuoraQ.question\_text)

Names=cv.get\_feature\_names()

dict1=cv.vocabulary\_

y\_train=QuoraQ.target

naive\_bayes = BernoulliNB()

naive\_bayes.fit(X\_train\_cv, y\_train)

trainedpred=naive\_bayes.predict(X\_train\_cv)

y\_train==trainedpred

X\_train\_cv is a code in which generates and lists a large amount of methods the user can use for a large amount of things. Before running this code, the user needs to first identify “cv” (CountVectorizer) to the system in order to run “X\_train\_cv”.

4) Compute the accuracy\_score, precision\_score and recall score (recall is sensitivity). You need to make some changes for instance accuracy\_score(py\_test,predictions) to accuracy\_score(y\_train,predictions) Note that there are multiple predictions object, every time you run a line that has predictions = something, you are recreating the object. Make sure y\_train and predictions have the same length so that you are comparing what you observed with what you predicted. This means the predictions object you need is around line 61 which operates on X\_train\_cv.

print('Accuracy score: ', accuracy\_score(y\_train, predictions))

sum(y\_train==predictions)/len(predictions)

Accuracy Score: 0.9491895261845387

Out[54]: 0.9491895261845387

print('Precision score: ', precision\_score(y\_train, predictions))

sum(y\_train[predictions==1]==1)/len(y\_train[predictions==1])

Precision score: 0.6306905370843989

Out[56]: 0.6306905370843989

print('Recall score: ', recall\_score(y\_train, predictions))

sum(predictions[y\_train==1]==1)/len(predictions[y\_train==1]==1)

Recall score: 0.43511248345831494

Out[58]: 0.43511248345831494