Please include your Name and USC ID in the first cell of the Jupyter Notebook file for each assignment. Name your jupyter file as Lastname\_Firstname\_HW(i).ipynb, for example, John Doe’s HW3 should look like Doe\_John\_HW3.ipynb. Similar to HW2, a single ipynb file is the deliverable for this homework.

"""

At this point I am assuming that:

--you are comfortable writing and testing python code for the purposes of these homeworks.

--you are comfortable with statistical/ML tools like sklearn and numpy.

--if given reasonable problems involving featurization of raw data, training, testing, sampling etc., you

can figure out reasonable solutions to these problems.

MAX SCORE POSSIBLE: 100 points

Good luck!

"""

As a first step, you must import sklearn for this homework

import sklearn

"""

[40 points] Replace 'pass' below as appropriate. Make sure to read the notes in the function header.

<The main goal of this function is to ensure you can use the 'tfidf' function offered in the sklearn

lib: https://scikit-learn.org/stable/modules/generated/sklearn.feature\_extraction.text.TfidfVectorizer.html>

"""

def encode\_text\_tfidf\_vectors(in\_folder='/train/', subject=False): # appropriately replace this ‘in\_folder’ path with your own after you've unzipped the input file.

"""

1. The goal of this function is to take the path to a folder containing your text files, read in/encode the text files as tf-idf vectors

and labels and output the tf-idf matrix X and the labels-vector y. You may use the sklearn tf-idf vectorizer

with default parameters. Initially, the goal should be to get the full system working rather than 'optimize'

the tf-idf and its parameters for best performance.

2. Note that the text in each input file has already been 'preprocessed' (in this case, each unique word has been assigned to a unique number.

This makes your job easier, not harder, since it means we won't have to process the 'raw' text and perform NLP-based preprocessing

like tokenization/word segmentation etc.). Make sure to read the readme file to understand what's going on here, including

on how the name of the file tells you whether a file is 'legit' or 'spam'

Hint: A useful package here is https://docs.python.org/2/library/os.html#os.listdir, which comes with python

and allows you to directly list/manipulate files in folders etc. Other ways are fine too, as long as your

approach is programmatic (i.e. you are not allowed to ‘manually’ input a long list of file-named directly in this exercise, since this is obviously not scalable/appropriate)

3. If subject is false, then that means you should only produce the tf-idf of the 'main body'. If subject is true,

then you should produce the tf-idf of only the subject (see extra credit question at the end of this program). Make sure to write the code accordingly.

[At no point during this homework will we be producing the

tf-idf of the subject+main body combined. That is, for a single file, either the subject OR the main body but not BOTH will always be used.]

Hint: You will need to parse the contents of the file accordingly. The package 're' (also included in the python

system lib) is your friend as it allows you to split strings (and file contents) using regular expressions!

"""

X = None

y = None

pass # put your code here. You can define helper functions if you want.

return X, y

"""

[40 points] Train a classifier on the train dataset that is able to predict spam. You must:

--Try at least five different classifiers

--Try at least three different tf-idf 'settings' i.e. try to play with the parameters in the sklearn tf-idf vectorizer to achieve

optimal performance (measured using 'accuracy' i.e. the percentage of correct classifications).

--Use some kind of 'validation' mechanism to ensure that you truly select the best classifier and don't overfit. There

are several ways of achieving this including (i) cross validation (if you are able to do it), or (ii) split your 'original' training

data into a 'training' and 'validation' set (e.g., maybe using an 80/20 or 90/10 split). Use the 'training' portion

for training your classifier, and your 'validation' portion for getting an unbiased estimate of classifier performance.

Make sure to use stratified sampling (the code from the previous HW is your friend here)!

You MUST NOT use the test dataset at all for this part of the experiment. The deliverable for this exercise should

be both the code that you write below as well as a short summary showing results for the different things you tried. The summary may include tables, text, figures etc. but we must be able to read and understand it, and it should be within this ipynb just like previous homeworks. Shorter is better,

but not at the cost of readability/clarity. Be liberal in your use of tables/diagrams. For example, you could produce

a table with 5 X 3 rows, showing performance on the validation set for each of the fifteen things you tried.

[20 points] Using the best and second-best classifiers you got from above, apply them once each on the entire test dataset.

If you run into the out-of-vocabulary problem i.e. there are words in your test data that are not in your training data,

you can delete the word, although in practice we would be taking a more sophisticated approach.

What are the accuracy measures that you are getting for both? Is the difference greater, smaller or equal compared to performance

difference on validation set/cross-validation? What would be the accuracy for a classifier that labeled the data 'randomly'?

(Hint: for the last question, use a bernoulli distribution i.e. toss a coin, with spam coming up with probability p and non-spam

with probability 1-p. Use the number of spam/legit samples in the training data to estimate the ideal value for p.)

"""

"""

[10 points Extra Credit] Using the subject tf-idf rather than main-body, re-train your classifier/model of choice above (you

do not have to try out fifteen different things for this exercise), and apply it to the test set (again, using subject tf-idf

as features). Does the performance improve?

HINT: The very first function you wrote is your friend here!

"""