ECE 457B, Fall 2025, Assignment 3
Due: Sun, Nov 02, 11:59pm ET
Absolutely *no* extensions!

Submission:

> You do not submit any of your code for this assignment. Rather, you should submit a single pdf file with your responses to the problems in this assignment, which are based on observations from running code that you write/modify. So your submission instruction: submit your single pdf file to the appropriate dropbox on Learn.
>
> We may carry out plagiarism-checking on your submissions. You should credit all software/people as appropriate. It does not have to be elaborate. E.g., "Credit: Alice Wong, and ChatGPT."

The intent of this assignment is for you to (i) experience some challenges with real-world datasets, and, (ii) progress further on your experimentation with techniques we have learned in Chapters 02 and 03. You are allowed to use any Python packages you like for this assignment. You may want to look through ch02.py and ch03.py for what they import.

To begin, read in the training and test datasets for the fetch_20newsgroups from Scikit-learn:

```
import sklearn.datasets as datasets

d_train = datasets.fetch_20newsgroups(subset='train', remove=('headers', \
'footers', 'quotes'))

d_test = datasets.fetch_20newsgroups(subset='test', remove=('headers', \
'footers', 'quotes'))
```

There are two main fields in each of d_train and d_test we care about: data and target. (If you want to look at a mapping of target to what each target really is, look at the field target_names.) The field data is a bunch of text, including characters such as `'\n'` and punctuation. The field target is the newsgroup to which the text was posted.

We want to see how well we are able to classify the text, based on the frequency of occurrence of certain words in it, into a target. For this, you would first (a) *tokenize* words in each data field, then, (b) count the number of occurrences of words from a *dictionary* among those words in the data field. We adopt each word in the dictionary as a feature, and the number of occurrences of a word in the data field as the value for that entry along that feature.

For example, suppose the data field is:

> This is a test.
> This is a test only.
> If this weren't a test, it would be something else.

And suppose the dictionary comprises the four words "test", "this", "fortune" and "only". Then, we have four features, each corresponding to a word in the dictionary. With the following values, based on the number of occurrences.

test	this	fortune	only
3	3	0	1

To tokenize words in a data field of each entry, here is an approach:

- Remove `'\n'`s. You should be able to do this using the `replace()` for strings.
- Remove punctuation marks. You could do this by importing `string` and then `...translate(str.maketrans(...))` with `string.punctuation`.
- Remove strings that contain digits. You could do this using regular expressions.
- Convert the data field to lower case. You should be able to do this with the `lower()` method for strings.
- Remove *stop words*. Observe that stop words do not appear in the dictionary. You could import `get_stop_words` from `stop_words`.

Each problem (1) and (2) below is worth 50 points.

1. Adopt as dictionary the words in keywords769.txt on Learn. For each of (i) logistic regression with one-vs-rest, (ii) logistic regression with one-vs-one, and, (iii) support vector machine, first learn with the training data. Then, check the `accuracy_score` for the test data. For logistic regression, use the lbfgs solver with max_iter set to 1000. (You should observe that a max_iter of a smaller value of 100 causes complaints from the call.) For the support vector machine, adopt the linear kernel. Try different values for the regularization parameter: $C = 1000, 100$ and 1.

   Does any of the learning approaches yield a good accuracy score? Does the value of C seem to impact how well you are able to learn? Based on all the inputs and data you have, reason about the cause(s) for what you observe.

2. Repeat #1 above, but this time with the much smaller keywords35.txt as your dictionary. Compare the quality of the learning outcome between #1 above and this exercise. Again try all three learning approaches, each with the three different values for the regularization parameter C. Reason about why it performs better/worse. Notwithstanding whether it performs better/worse than #1 above, in an absolute sense, is the learning effective under any of the learning schemes and parameters?