Assignment 3

Start Assignment

Due Friday by 11:59pm **Points** 100 **Submitting** a website url
Available after Oct 14 at 12am

Note: I'm giving you longer than usual to do this assignment because your midterm is coming up, but I believe it will help you appreciate PCA, Explicit Euler, and data cleaning.

Please put your answers to this assignment in the same GitHub repository that you used for the first assignment.

In the repository, be sure to include runnable source code, and a single consolidated (one file) report (readme) that has answers to all the questions, along with code, and relevant output (e.g. graphs). I would prefer for the consolidated report to be in pdf format, but markdown (md), word (docx), and html are also fine. Failure to provide a single-file report will incur a 5 point penalty.

As always, reasonable discussions about programming issues (or making a report) are encouraged, both in the relevant **forum** and in person with your classmates. Do not, however, share code with your classmates; the work you submit must be your own. (i.e. Learn from each other, but don't cheat.) For questions involving fine details of your implementation, please meet with the TF or professor. Please do not spend hours stuck on an exercise; if you get stuck and have tried to get unstuck, ask the TF or professor a question and move on to something else until you get an answer.

Exercise 1. (25 points)

Use the requests module (or urllib) to use the Entrez API (see **slides 5**) to identify the PubMed IDs for 1000 Alzheimers papers from 2022 and for 1000 cancer papers from 2022. **(9 points)**

Note: To search for a disease and a publication year, structure the term like: **Alzheimers+AND+2022[pdat]** (Here [pdat] indicates that this is a publication year, and the AND (has to be all caps) means both conditions should apply.)

Use the Entrez API via requests/urllib to pull the metadata for each such paper found above (both cancer and Alzheimers) (and save a JSON file storing each paper's title, abstract, and the query that

found it that is of the general form: **(12 points)**

```
{
    "32008517": {
        "ArticleTitle": "Deep Learning for Alzheimer's Disease Classification using Texture Featu
res",
        "AbstractText": "We propose a classification method for Alzheimer's disease (AD)...",
        "query": "Alzheimer"
    }, ...
}
```

Here 32008517 would be the PubMed ID of one of the 2000 papers, specifically one that came from searching for Alzheimer's papers (it won't be in your data set because it was published in 2019). You should include the full AbstractText; I'm abridging here for clarity.

There are of course many more papers of each category, but is there any overlap in the two sets of papers that you identified? **(3 points)**

Hint: To do this, you'll probably want to look at one of the XML responses with a text editor so that you understand how it is structured.

Hint: Some papers like 32008517 (https://pubmed.gov/32008517) have multiple AbstractText fields (e.g. when the abstract is structured). Be sure to store all parts. You could do this in many ways, from using a dictionary or a list or simply concatenating with a space in between. Discuss any pros or cons of your choice in your readme (**1 point**).

Caution: the PubMed API allows a rate of at most one query at a time and no more than 3 per second unless you have an API key. To be safe, use

```
time.sleep(1)
```

after each query to the PubMed API.

Note: This doesn't require 2002 separate queries. You can get the metadata for many articles at a time by using a comma separated list of ids. While GET queries have a total line length limit, you could use a POST query instead and get the information for all the papers in one pass. (We can use POST instead of GET here in part because this is *not* a RESTful API.)

Note: BioPython provides functions for accessing the PubMed API. Do not use them; use the **requests** module to do an HTTP or HTTPS request directly on a URL that you specify with the parameters that you specify. Why? Because this approach is general and will work in many contexts whereas BioPython only works for PubMed and only from Python.

NOTE: sometimes papers (e.g. 31842501 (https://pubmed.gov/31842501)) have italics in their title or abstract. If so, using .text won't work well; use ET.tostring(item, method="text").decode() instead.

Exercise 2. (25 points)

Machine learning and data visualization strategies generally work best on data that is numeric, but exercise 1 gave us text data, and indeed text is common. Fortunately, modern NLP algorithms powered by machine learning trained on massive datasets exist that can take words (e.g. word2vec) or titles and abstracts (e.g. SPECTER) and return a vector of numbers in a way that similar items are given similar vectors. Since we have titles and abstracts, let's use SPECTER.

In particular, for each paper identified from exercise 1, compute the SPECTER embedding (a 768-dimensional vector). Keep track of which papers came from searching for Alzheimers, which came from searching for cancer. (**5 points**) If you are familiar with SPECTER and wish to do it another way, that's great, if not here's one approach based on https://github.com/allenai/specter (https://github.com/allenai/specter) :

Install pytorch (a deep learning library) by following the instructions here: https://pytorch.org/get-started/locally/ (https://pytorch.org/get-started/locally/)

Install the huggingface transformers module: **pip install transformers**

(🤗 huggingface (https://huggingface.co) provides access to a number of pre-trained NLP language models.)

Have your code load the SPECTER model (the first time you do this, it will take a bit to download the model; it will be stored locally for fast reuse later):

```
from transformers import AutoTokenizer, AutoModel

# load model and tokenizer
tokenizer = AutoTokenizer.from_pretrained('allenai/specter')
model = AutoModel.from_pretrained('allenai/specter')
```

Load the papers dictionary (**3 points**) and then process your dictionary of papers to find the SPECTER embeddings (**2 points**). One (somewhat slow) way to do this is as follows:

```
import tqdm

# we can use a persistent dictionary (via shelve) so we can stop and restart if needed
# alternatively, do the same but with embeddings starting as an empty dictionary
```

```
embeddings = {}
for pmid, paper in tqdm.tqdm(papers.items()):
    data = [paper["ArticleTitle"] + tokenizer.sep_token + get_abstract(paper)]
    inputs = tokenizer(
        data, padding=True, truncation=True, return_tensors="pt", max_length=512
    )
    result = model(**inputs)
    # take the first token in the batch as the embedding
    embeddings[pmid] = result.last_hidden_state[:, 0, :].detach().numpy()[0]

# turn our dictionary into a list
embeddings = [embeddings[pmid] for pmid in papers.keys()]
```

At this point, embeddings[i] is the 768-dim vector for the ith paper.

Apply principal component analysis (PCA) to identify the first three principal components. (**5 points**) I suggest using the sklearn module, e.g.

```
from sklearn import decomposition
pca = decomposition.PCA(n_components=3)
embeddings_pca = pd.DataFrame(
    pca.fit_transform(embeddings),
    columns=['PC0', 'PC1', 'PC2']
)
embeddings_pca["query"] = [paper["query"] for paper in papers.values()]
```

(See the documentation for sklearn.decomposition.PCA (https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html) if needed.)

Plot 2D scatter plots for PC0 vs PC1, PC0 vs PC2, and PC1 vs PC2; color code these by the search query used (Alzheimers vs cancer). (**5 points**) Comment on the separation or lack thereof, and any take-aways from that. (**5 points**)

Exercise 3 (25 points)

Consider the simple, non-spatial SIR model (https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology#The_SIR_model):

$$\frac{ds}{dt} = -\frac{\beta}{N} si$$

$$\frac{di}{dt} = \frac{\beta}{N} si - \gamma i$$

$$\frac{dr}{dt} = \gamma i$$

In this epidemiology model, s is the number of "susceptible" individuals, i is the number of infected

individuals, r is the removed population that can no longer become sick, N is the total population (hence $N = s + i + r$), β is a scale factor measuring how likely an infected person is to make a susceptible person sick, γ measures the rate at which infected individuals are removed from that state.

Write a Python function that uses the Explicit Euler method to plot $i(t)$ given $s(0), i(0), r(0), \beta, \gamma$ and T_{max} (the last time point to compute). (N follows from the formula.) Do not use an integration library; provide your own implementation (**5 points**).

The New Haven population is approximately $N = 134000$. Suppose that on day 0, there was 1 person infected with a new disease in New Haven and everyone else was susceptible (as the disease is new). Suppose further that $\beta = 2$ and $\gamma = 1$. Plot the time course of the number of infected individuals until that number drops below 1 (at which point, we'll assume the disease has run its course). (**5 points**)

For those parameter values, when does the number of infected people peak? (**2 points**) How many people are infected at the peak? (**3 points**). (Have your code identify these things; don't do it manually.)

Unfortunately, for new diseases, we may not know β or γ with much accuracy. Vary these two variables over "nearby" values, and plot on a heat map how the time of the peak of the infection depends on these two variables. (**5 points**). Do the same for the number of individuals infected at peak. (**5 points**)

Hint: if you're using plotnine, consider using geom_raster or geom_tile (https://dputhier.github.io/jgb53d-bd-prog_github/practicals/intro_ggplot/intro_ggplot.html#creating_heatmaps) to make the heat map. Other approaches are fine too.

Exercise 4 (25 points)

Identify a data set online (**5 points**) that you find interesting that could potentially be used for the final project; the main requirements is that there needs to be many (hundreds or more) data items with several identifiable variables, at least one of which could be viewed as an output variable that you could predict from the others.

Describe the dataset (**5 points**) Your answer should address (but not be limited to): how many variables? Are the key variables explicitly specified or are they things you would have to derive (e.g. by inferring from text)? Are any of the variables exactly derivable from other variables? (i.e. are any of

them redundant?) Are there any variables that could in principle be statistically predicted from other variables? How many rows/data points are there? Is the data in a standard format? If not, how could you convert it to a standard format?

Describe the terms of use and identify any key restrictions (e.g. do you have to officially apply to get access to the data? Are there certain types of analyses you can't do?) (**5 points**)

Remember: if you can't find explicit permission to use a given dataset, assume that you cannot do so.

Do data exploration on the dataset, and present a representative set of figures that gives insight into the data. Comment on the insights gained. (**5 points**)

Identify any data cleaning needs (this includes checking for missing data) and write code to perform them. If the data does not need to be cleaned, explain how you reached this conclusion. (**5 points**)

Note: You're not committing to use this dataset for the project, but this will give you one option.