Perceptron
Programming Project 4

Your source file is a survey of 600 college students from a major university, in which they answered questions about their interest and usage of alcohol. There are 86 questions that were asked of each student, and a final prediction stored in the variable `DSMDX`. A `1` indicates that they are abusing alcohol, and a `0` indicates that they have not begun down that path. The original survey data was graded on a 5 point scale, and details of both the questions and the response meanings can be found in the file `feature_dictionary`. These response values were converted to a smaller set of response values based on the following table:

Original Score	**New Score**
0	0
1	0
2	0
3	0
4	1
5	1
99	-1

Spit Spot – Clean the Data. As one might expect, this data is dirty, and the conversion didn't quite work cleanly. You will see values that are not simply ± 1 or 0. You will first need to clean the data so that it reflects that format in some intelligent way. For example, you may want to think about the best way to handle the value 3, and how that might differ from handling the value -745, for example. Furthermore, you will need to come up with some sort of strategy for handling "missing" values (that were described as 99 in the `feature_dictionary`). The reason for this is that the simplified neural network that you will be writing won't be able to handle inputs that are not ± 1. You could use a number of strategies, each with its own particular consequences. Here are a few different strategies for handling them:

- You could drop (or ignore) the rows where wizards that have weird data.
- You could drop (or ignore) the columns that have weird data.
- You could set all of the values to 0, 1, or randomly choose which one to pick for each missing value.
- You could do detailed statistics to try to figure out the best way to fill in these values (over-achiever alert, but actually the right thing to do).
- You could choose to make your input values decimal numbers in the range $[0, 1]$ and run the standard algorithm on that. For this, you'd have to decide how to fill in the decimal value – randomly, using some fixed linear equation weighted to 0 or 1, setting it to 0.5, etc.

Once you have done some basic data engineering (or data wrangling as some people like to call it), you now have a well-formatted piece of data. Congratulations! You should save this Python script as a separate `.py` file from the script you will write for prediction.

Prediction – Neural Network for One. Your task, if you choose to accept getting positive points for this project, is to implement a single perceptron neural network that will take 86 input parameters and predict whether each stuent has a problem with alcohol abuse. You have only a subset of the total data, which is the train/test data, and I have held back the validation data so that I can see how well you did! You will have to decide what percentage of the test/train data you will allocate to each, and it may help to think back on the class discussion on cross validation if you want to build a better classifier. For a perceptron, you know that there are four parameters that you can adjust/tweak to encourage different behavior from your neural network:

- The threshold value t,
- The step size s of weight updates,
- The initial weights on the edges of the perceptron, and
- How many rounds of training you perform in the event that the perceptron does not *stabilize*.

You will need to develop the best classifier that you can with this information. Your output will be the final list of the 86 weights along with the threshold value t, put into a file called `weights.txt` with the following format:

```
<value of t>
<edge weight 1>
<edge weight 2>
...
<edge weight 86>
```

For this to work correctly, you need to make sure that you don't change the ordering of variables in your dataset. (If you opted to drop columns while cleaning, you will have to set the edge weight of that corresponding feature to 0.)

Turn In Materials

What you must submit on Canvas:

1. The `.py` file used to clean the dirty data.
2. The `.py` file used to create the `weights.txt` file requested above.
3. The output file `weights.txt`.