University of San Diego
SHILEY-MARCOS SCHOOL OF ENGINEERING

Assignment 3.1 Exercises

For this assignment, you will conduct programming tasks in Google Colab and answer provided questions in a Google Doc or MS Word document. You will submit **2** documents to Blackboard:

1. A PDF file converted from your Assignment 3.1 Short Answer Template Word Doc (document is linked in Blackboard assignment prompt).
    - This template has a copy of all short answer questions for this assignment.
2. A PDF file converted from Google Colab.

Drum, Text, and Animal Data
You will work with three data sets for this assignment, as you did in assignment module 1.

Drum Data: These data are in drum_data.csv. These are the spectrum (frequencies) of sounds from a digital drum musical instrument. The task is to identify which type of drum produced the sound.

Text Data: These data are in count_hamilton.csv. These data are the word counts of the Federalist papers, important documents in American history which have disputed authorship. The task is to identify the author of these documents.

Animal Data: These data are from an animal shelter. These are records of animals coming in for treatment, with a few characteristics of the animals and information on the outcome. The task is to predict the outcome.

Written Questions (Complete these in the Assignment 3.1 Short Answer Template, located in your assignment prompt)
(short answer, 2-3 sentences):

- For the animal shelter dataset, the classification results using k-nearest neighbor are not as good as the other datasets. Why do you think this is the case for this data? Explain in terms of inductive bias, and remember that you used OneHotEncoding, making the data more high-dimensional and sparse.

- Which distance metric seemed to perform the best on the test dataset, Euclidean or Cosine distance? How large are the text feature vectors (how many features/words are in the dataset)? Do you think high-dimensional data like this are better suited for the Euclidean or Cosine distance, based on your understanding of the methods and the results you have obtained here?

- On the audio (Drum) dataset, you should have used nearest neighbor, perceptron, and your own handwritten implementation of the perceptron. Which classifier performs best, or are they all giving about the same results? If there is a clear winner, explain why this classifier might be a good match for the audio dataset.

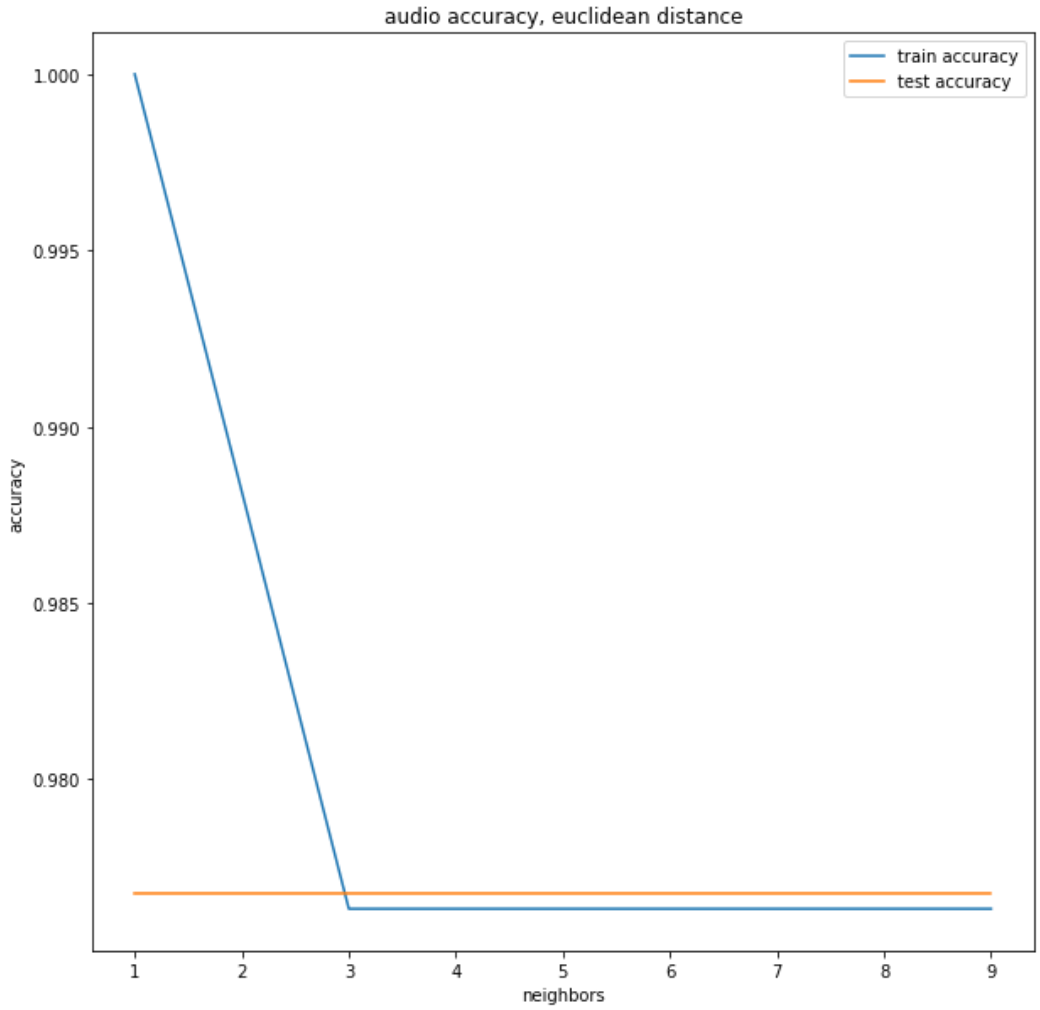
Programming Section (Complete this section in Google Colab)
One key to designing a k-nearest neighbor model is a good choice of the distance metric. By working through these examples (in order), you will overcome a few challenges in applying distances to datasets. Use scikit-learn's neighbors.KNeighborsClassifier to train a classifier on each of the provided datasets.

For all three datasets, because they are small in size, you will use most of the data for training. For all model, do a cross-validation split using:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.10, random_state=42)
```

Load the Drum sounds dataset. Remember that these data are vectors, and each value is the amount of power (signal intensity) at that particular frequency. Use the distance metric euclidean.

Next, staying with the *Euclidean* metric, increase the number of neighbors from1 to 9, only using odd numbers. Save the accuracy at each value of k, and you will be plotting them later. Change the distance metric to *Manhattan* distance. Again, vary the number of numbers and plot results for accuracy using *both* distance metrics on the Drum dataset. Plot both the training and test set errors as below.



Next, you will try the animal shelter data. To convert these categorical features to a vector, use enc = OneHotEncoder(sparse='False'). Read the documentation on what oneHotEncoder is doing, and note how large the transformed data are. Plot a graph of accuracy using k varying from 1 to 9 and using Cosine as the distance.

Finally, load the text data using count_hamilton.csv. You will again use k-nearest neighbors to build a classifier, varying the number of neighbors from 1 to 9. Use both Cosine and Euclidean distance to plot accuracy as the number of neighbors increases.

Perceptron Implementation

Load the audio (Drum) dataset. Make sure you have plus and minus ones for labels. For example, you can modify the labels like this:

```
y=le.transform(labels)
y[y==0]=-1
```

Write the following functions:

predict_perceptron(X,w)

This takes a dataset and a weight vector and returns all the predictions that would be made by a perceptron.

train_perceptron(X,y,w)

Returns w, the trained weight vector. This will take the form of a for loop over all the data points, checking against the pseudocode in the text.

Train and test your perceptron using the audio (Drum) dataset. In addition, use scikit-learn's perceptron classifier and compare. Verify that both your implementation and the built-in one are obtaining similar accuracy values.