Tyler Angert
CS378
HW3 Writeup: k-means clustering

1. **Introduction**
    1. This assignment implements the k-means clustering algorithm in Python from scratch.
2. **Datasets chosen**
    1. I chose the Iris and Seeds data set from the UCI machine learning repository. They are both entirely Integer/Real number based and have no categorical attributes (besides the labels), so no discretization had to be done in order to process the data.
    2. The Iris data set describes the physical features of various kinds of Iris plants, such as sepal length and width, and petal length and width.
    3. The Seeds data set describes the geometrical properties of kernels for three different kinds of wheat. The attributes are area, perimeter, compactness, kernel length, kernel width, asymmetry coefficient, and length of kernel groove.
3. **Algorithm implementation**
    1. I took this from an object-oriented approach and created **DataPoint** and **Centroid** objects that stored relevant metadata to evaluate the algorithm.
        1. The **DataPoint** class looks like this:

```
class DataPoint:
    def __init__(self, data):
        self.data = data
        self.current_centroid = None
        self.previous_centroid = None
```

This class stores the data, aka the “coordinate” of the point, its current centroid it belongs to, and the centroid it previously belonged to. This way, in order to check convergence, the algorithm simply compares to see if every DataPoint’s current and previous centroid are different. If *at least* one data point has a different current/previous centroid, then the algorithm continues to another iteration. If all DataPoints have the same current/previous centroid, the algorithm has converged and we have our final clusters.

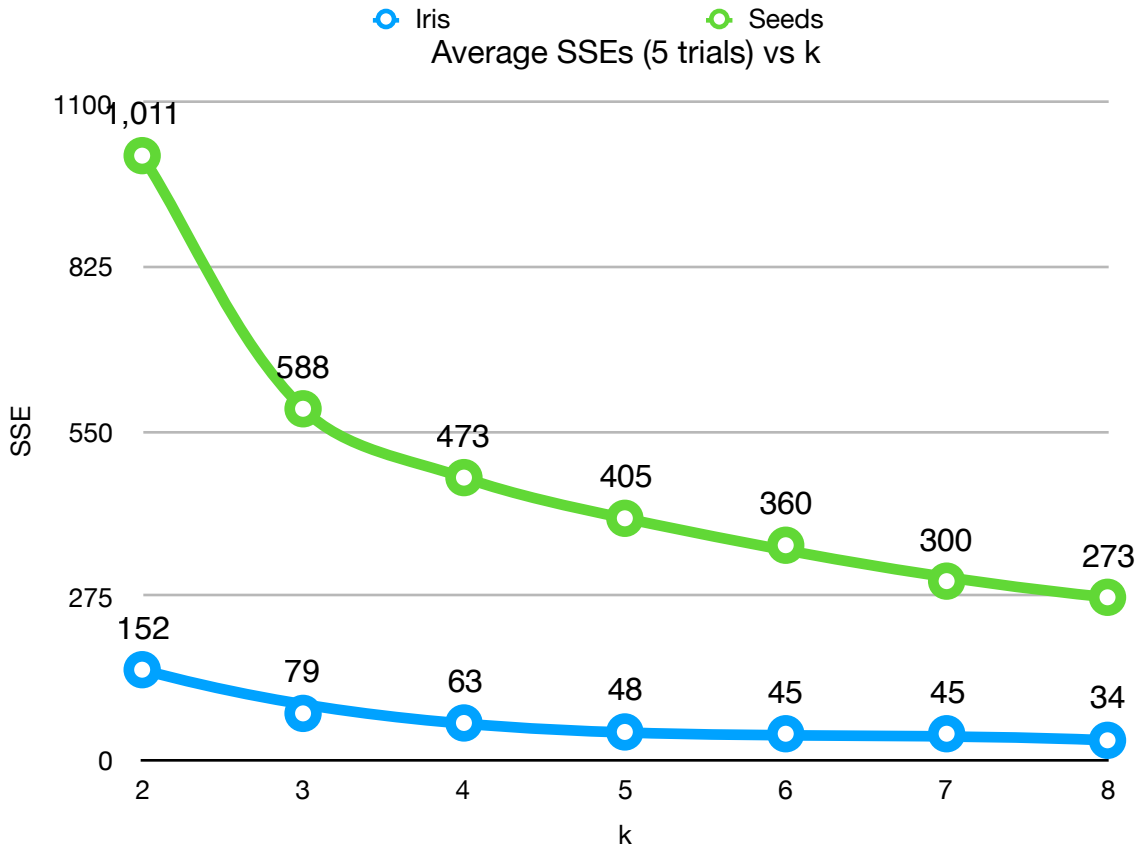
2. The **Centroid** class looks like this:

```
class Centroid:
    def __init__(self, data, cluster):
        self.data = data
        self.cluster = cluster
```

This class simply stores the coordinate of the centroid and the label of the cluster, such as 0, 1, 2, etc.

I used **euclidean distance** as my distance metric. No significant preprocessing was done besides converting all attributes to Floats in order to do numerical comparisons easily.

4. Insights, results, conclusions



This was a really interesting algorithm to implement because of the nuances in certain edge cases. For example, I tracked down a bug that was giving me null centroids because I realized, during the initial randomization of the centroids, sometimes *two* of the centroids would have the same location— causing one of them to have no points. The elbow rule is also interesting to investigate, especially since on the seeds data set, the SSE seemed to steadily decrease even with values of k > 10, whereas the Iris set flat lines around 4/5.