Code	**CLUSTERING.PY**
Author	Nathaniel Heatwole, PhD (heatwolen@gmail.com) (GitHub) (LinkedIn)
Summary	Uses k-means and k-nearest neighbors (KNN) clustering to identify the parts of a snowman (from scratch and using the built-in functionality in sklearn).
Methods/ process	k-means clustering: - Unsupervised learning method that assigns points to cluster with nearest centroid (cluster boundaries are linear). - Minimizes within cluster sum of squared distance (WCSS), or variance in position about cluster centroids. Steps: 1. Randomly select locations for the initial cluster centroids. 2. Assign points to cluster with nearest centroid, and recompute cluster centroids. 3. Repeat the previous step for some fixed number of iterations. 4. Repeat the entire process for many different sets of random initial centroids. 5. Choose the cluster centroids yielding the lowest WCSS (sum over all clusters). k-nearest neighbors clustering (KNN): - Supervised learning method that assigns clusters using plurality vote of k-nearest neighbor points. - Cluster boundaries do not have a particular parametric/functional form. Steps: 1. Compute distance to all points in the training data. 2. Sort this vector of distances (ascending). 3. Assign cluster using plurality vote of k-nearest neighbor points. 4. Repeat this process for all points in the test data.
Training data	Randomly generated points (synthetic data) in the general shape of a showman (three circles of different radii stacked on top of one another).
Output	Plots: - Training data (snowman) - k-means clusters (with cluster centroids) - k-means accuracy (correct/misclassified) - KNN cluster regions Summary: - k-means accuracy (misclassification rate) - k-means centroids (from scratch, sklearn) - KNN distribution (% in each cluster)
Result	The results from scratch and using sklearn align closely. Both k-means and KNN identify the different parts/regions well, although k-means misclassifies some points.