Portfolio Component: Searching for Similarity

A. **kNN and Decision Trees**
   a. Regression:
      i. KNN Regression approximates the association between independent variables and continuous variables, stores all available cases and predicts the target (numeric) based on a similarity measure. It computes the Euclidean from query to labeled examples, orders the labeled examples by increasing distance, finds k of nearest neighbors based on the root-mean-square deviation, and calculates the inverse distance weighted average with k-nearest neighbors.
      ii. Decision Tree Regression builds models in tree structure form. It divides the dataset into smaller subsets (regions), while creating a decision tree, which results in a tree with decision and leaf nodes. Within each region, it wants to minimize the residual sum of squares. All predictors are examined to see if they're able to make good splits in the data. It will continue to split it until a stopping threshold is reached.
   b. Classification:
      i. kNN Classification is a supervised, instance-based learning algorithm which classifies a selected k number of neighbors near select data points to be classified. Essentially the class at which the most neighbors near the data point belong will be the class the data point will be classified to. It won't actually build a model, unlike previously discussed linear models, but rather build training observations in memory. In the case of Classification, it will solely use the majority class of the nearest neighbors in its predictions, rather than the averages; this is the main difference between the Classification and Regression variants of kNN.
      ii. Decision Tree Classification uses the same aforementioned greedy algorithm, but uses class counts in the subsets/regions rather than the residual sum of squares. Additionally, other measures - outside of accuracy - must be used for splitting regions, as accuracy itself is insufficient at measuring the purity of said regions.

B. **How the 3 clustering methods of step 3 work**
   a. KMeans Clustering: This is an unsupervised learning algorithm which uses observations as input and groups those inputs into k clusters. The algorithm starts with k centroids, similar to data points, which are typically placed

randomly somewhere inside the ranges of the data. We then assign each data point to a singular cluster depending on the shortest distance to that cluster. Once all data points are assigned a cluster, we compute a new mean, or a new value for the centroids. We do this by adding all the distances to that centroid and dividing it by the number of observations in that cluster. We keep computing new means until the means don't change, meaning we have found an optimal value of the centroids for that particular number of clusters. The true challenge of this algorithm is finding k, the number of clusters, that should be used in the algorithm. This k value is the main hyper-parameter for this algorithm.

b. Hierarchical Clustering: This is another unsupervised algorithm for clustering that starts out by treating each observation as its own separate cluster. Afterwards, the algorithm will identify which two clusters are the closest to each other, and then merge those two clusters into just one cluster. The algorithm will continue to identify and merge the most similar clusters until there is just one cluster left. Typically, one will tell the algorithm to stop merging at a particular number of clusters to actually have useful data. The output of this algorithm is a dendrogram which is a tree-like diagram that shows this process in action with each observation starting as a separate entity and then with each iteration merging eventually to one giant cluster. It should be noted that euclidean distance is the typical measure for how similar two clusters are to each other, however, the main difference of this algorithm to others is the choice of linkage. Linkage is exactly from where the distance between the clusters is computed. There are many linkage criteria, some of which include single, complete, average, and ward linkage.

c. Model-Based Clustering: This clustering algorithm is based on the Gaussian Mixture Model and holds three very important assumptions on the data. The first is that we already know how many clusters we need, second is that each observation has a singular and determinate probability of belonging to a cluster, and lastly that each cluster has observations that follow a normal distribution. Knowing all this, the algorithm simply needs to know what the centroids or means for the clusters are and what observation each cluster belongs to. These two questions can be solved by the second part of the algorithm, the Expectation-Maximization Algorithm, which is an algorithm that assigns each cluster to its most likely observation and then updates the cluster means variances based on the observations. One of the benefits of this model is that it can also give specific details as to what it believes the shape of the data is after it is given its input.

C. How PCA and LDA work, and why they might be useful techniques for machine learning

PCA works by reducing the dimensions of the data that we have. In order to do this though, it must first transform the data into a new coordinate space while also reducing the number of axes. It works by taking a data set that contains many variables into a smaller set that still contains most of the information vital for the data set. PCA is useful in understanding important variables while also picking out outliers in the data set.

LDA works in a similar way by splitting up a larger data set into two smaller sets. It does this by calculating how separable the different classes are and then seeing how much variance each class has. A good projection will separate the classes well.