University of Canberra
Faculty of Science and Technology

Programming for Data Science G (11521)

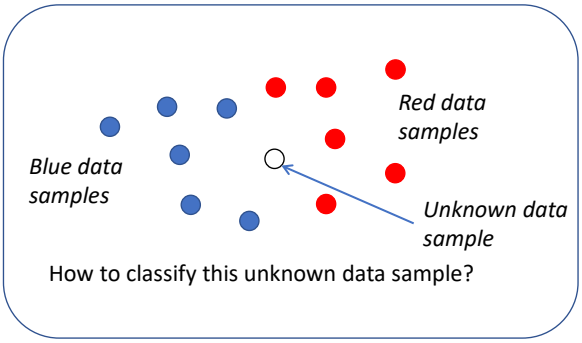
Assignment 1
Classifier and Cluster Analysis in Data Science

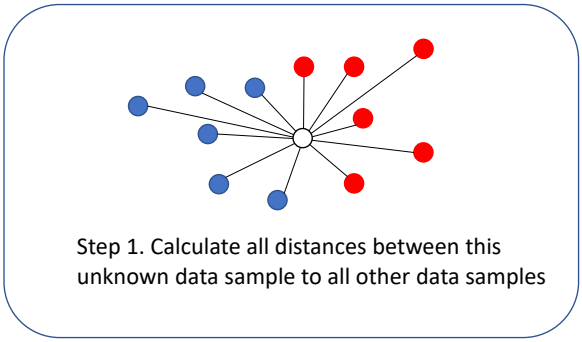
Due dates: 23:59 Sunday (Week 8)
Type: Individual assignment
Mark for assessment: 20
Submission: Submit a .zip file containing all Python files (.py) in your project via Canvas site.
Late submission: 5% of the total mark per day (1 mark per day). Information on how to apply for extension can be found in the unit outline on Canvas.
Remarks: As per unit outline, you will need an aggregate of **at least 40** over all assignments to pass the unit.

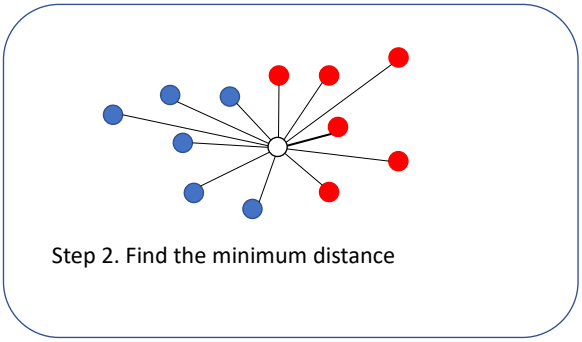
[6 marks] Question 1: Implement a Python program for **Nearest Neighbour Classifier** that can classify an unknown data sample to one of the given classes.

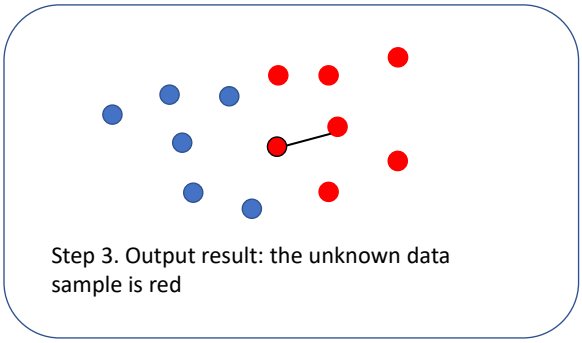
For example, there are 2 classes **Red** and **Blue**, and *x* is an unknown data sample (i.e., we do not know *x* is red or blue). After calculating all distances between *x* and all data samples in the 2 classes, we find a data sample in the Red class that has shortest distance to *x*, so *x* is classified as a red data sample.

Requirements: Your program reads data samples from 2 text files for 2 classes and unknown data samples from another text file, runs the Nearest Neighbour Classifier algorithm as demonstrated in the screenshots below, and outputs all unknown data samples and their classified label to screen and to another text file. Your program should work with any data dimension D > 1 and any number of unknown data samples > 0. For Python programming, use a **tuple** to store a data sample, a **list** to store all data samples, and **modules** to store functions. The main program includes only function calls and does not include any function implementations. Please do not use other versions of Nearest Neighbour Classifier you can find on websites or research articles, and do not import any external packages (except **tkinter**) to this project.







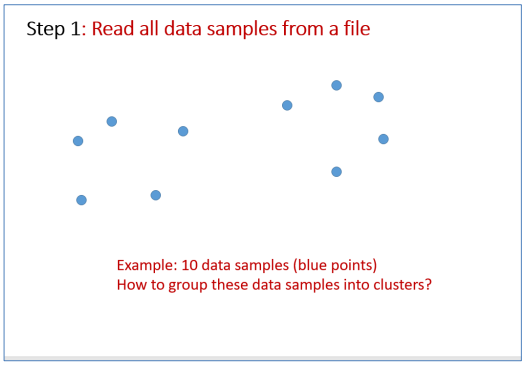


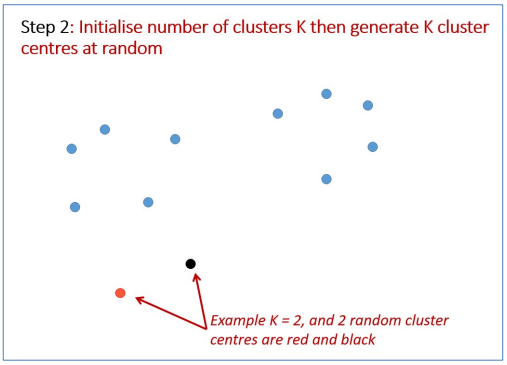
[14 marks] Question 2: Implement a Python program for **K-Means Clustering** that can group data samples to clusters.

For example, you are given a set of data samples to group them into 2 clusters. The K-means clustering algorithm generates 2 cluster centres at random, groups data samples that are nearest to the first cluster centre to form a cluster then do the same with the second one to form another cluster. The algorithm will generate new cluster centres by averaging data samples in the same cluster. If the difference between the 2 old cluster centres and the 2 new cluster centres are not significant, the algorithm will stop, otherwise it removes the old cluster centres and re-groups data samples for the new cluster centres as seen above to form new clusters. The process repeats until the difference between the old and new cluster centres is not significant.

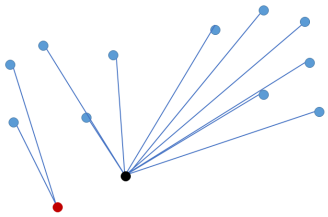
Requirements: Your program reads data samples from a text file, runs K-means Clustering algorithm as demonstrated in the screenshots below, and outputs all data samples with cluster centres to screen as below. Your program should work with any data dimension D > 1 and any number of clusters K > 1. For Python programming, use **tkinter** to display data samples and cluster centres on a canvas, a **tuple** to store a data sample or a cluster centre, a **list** to store all data samples or all cluster centres, and **modules** to store functions. The main program includes only function calls and does not include any function implementations. Please do not use other versions of K-Means Clustering that you can find on websites or research articles to implement this project. Please do not import any external packages (except **tkinter**) to this project.

The screenshots below explain how K-means Clustering algorithm works.

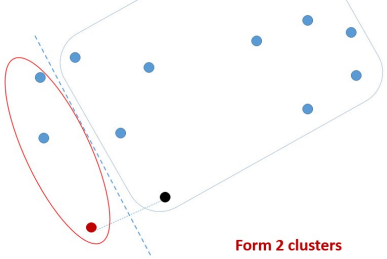




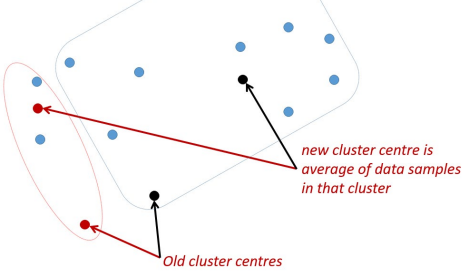
Step 3: For each data sample, find its nearest cluster centre



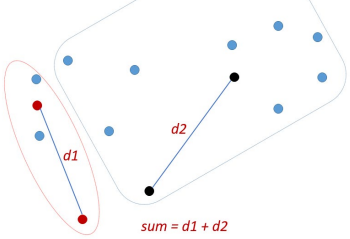
Step 4: Group data samples nearest to same cluster centre to form N clusters



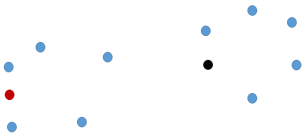
Step 5: Calculate new cluster centre for each cluster



Step 6: For each cluster, calculate distance between old and new cluster centres. Calculate the sum of all these distances

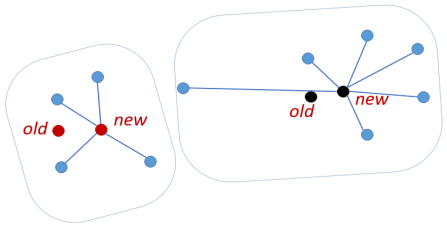


Step 7: If the sum is less than a given threshold: stop and output K cluster centres, else: delete old cluster centres and go to Step 3



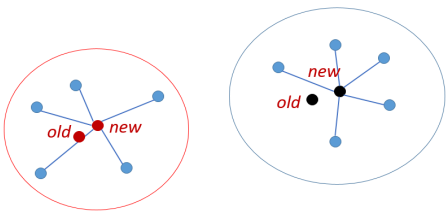
In this example threshold = 0, so sum > threshold: delete old cluster centres and go to step 3

Second round: Step 3 → Step 7



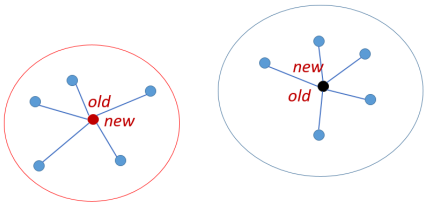
sum > 0: delete old cluster centres and go to step 3

Third round: Step 3 → Step 7



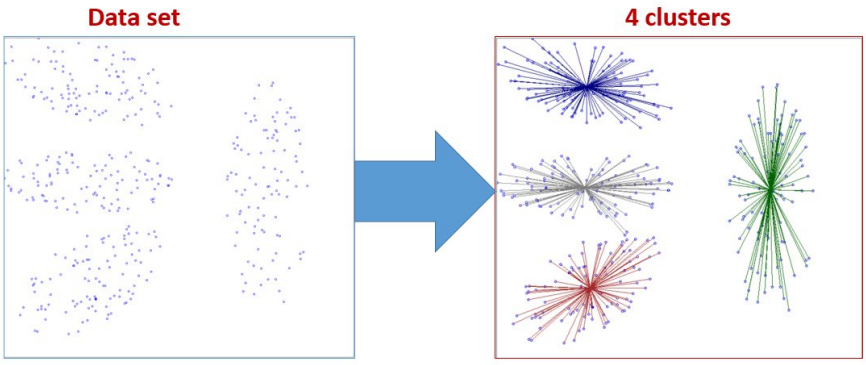
sum > 0: delete old cluster centres and go to step 3

Fourth round: Step 3 → Step 7



sum = 0: stop and output cluster centres

Below is an example of data samples drawn on screen before and after applying K-means clustering.



More details of the above algorithms and demos will be given in lectures and tutorials from Week 2 to Week 7.

-- END --