COMP9318 Project Report

Team member：

Z5151539 Tianyu Deng

Z5230310 Tian Liu

Part 1:

The purpose of part1 is to do PQ method via the given data, P, init_centroids and maximum number of iterations. According to the PQ method, we use K-means with L1 distance. Just like the normal K-means algorithm.

- Every iteration we use numpy operations to compute the **absolute value** of each data point and all centroids and then deicide which cluster the point belongs to.
- Then, recompute the centroids if each cluster, here we used median instead of mean because of the L1 distance. Apart from that, we also check some **bad centroids** which those don't have any points in its cluster. This could optimize a little bit of kmeans.
- After the max_iter iterations, we get a **codebooks**, and then do one more kmeans algorithms, set the codebooks as init_centroids parameter and last we get the **codes**.

Part 2:

The purpose of part2 is to do queries about some points and return an array contains the candidates for each query through codes and codebooks given by part1.

In order to efficiently extend Algorithm 3.1 with P > 2.

- First for the codes, we convert it to a dictionary, the keys are the code and values are index that corresponding to the code. This could help the program run faster when we find the candidates.
- After that, we do query iterations, we will never combine all points since it's very time-consuming, so we do generate P numpy arrays which contain the distance of the #P part of query data and all centroids.
- Following we sorted the P numpy arrays, and extend the Algorithm 3.1 to high dimension(i.e. P > 2).The main problem is that we avoid to compute some useless codes.

- Similar to 3.1, imagine we have a high dimension cube and we check every points if one dimension of this points is 0 (eg. (0,2,3,4) the first dimension is 0), we directly add it to pqueue, otherwise we check whether a point at this dimension minus 1 has been traversed, if so add this to pqueqe, else not and go to next iteration(eg. (1,2,3,4) at first dimension we check (0,2,3,4)).
- Add the index of these codes to candidates, the method is we can retrieve the centroids of each block by mapping through the P sorted numpy arrays (i.e. distance matrix of each block) via using the inverted index.
- Until we find T or greater than T codes and return index of these codes. And do next query.