Programming Project 1 – Prototype Selection for nearest neighbor

1. **A Short, high-level description of your idea for prototype selection:**
   Data under each label are condensed into $\frac{M}{number\ of\ labels}$ clusters using k-means algorithms. The new pool of data that contains the centroid of each clusters will form the reduced dataset for 1-NN algorithm.

2. **Concise and unambiguous pseudocode:**
   Step 1: Separate the training dataset based on the label.
   Step 2: For each dataset under a label, run k-means algorithm and reduce it to $\frac{M}{number\ of\ labels}$ clusters. Here, MiniBatchKMeans from sklearn was used. The centroid of each cluster is your output from this step.
   Step 3: Merge all the outputs of k-means algorithm for each label (centroid for each cluster) to create a condensed data set with M examples.
   Step 4: Feed the condensed data into kNN algorithm with k =1 and fit the test data to get your result. Here, KNeighborsClassifier from sklearn was used.

   Summary:
   Begin{
   Split dataset based on number of labels
   For each data under label:
   {
   fit kmeans (number_of_clusters = $\frac{M}{number\ of\ labels}$);
   condensed_dataset[label] = cluster_centroids;
   }
   merge condensed_dataset for each labels into one dataset with M-examples;
   fit kNeighboursClassifier with k =1 on the condensed data;
   predict against the test set;
   }End

3. **Experimental results:**
   Number of trials: 10
   The confidence level is given by t-distribution,

$$Confidence\ Interval = \bar{X} \pm \frac{t * \sigma}{\sqrt{n}}$$

   where,
   $\bar{X} = mean\ of\ the\ same\ set$
   $t = t\ score$
   $\sigma = standard\ deviation$
   $n = number\ of\ samples$

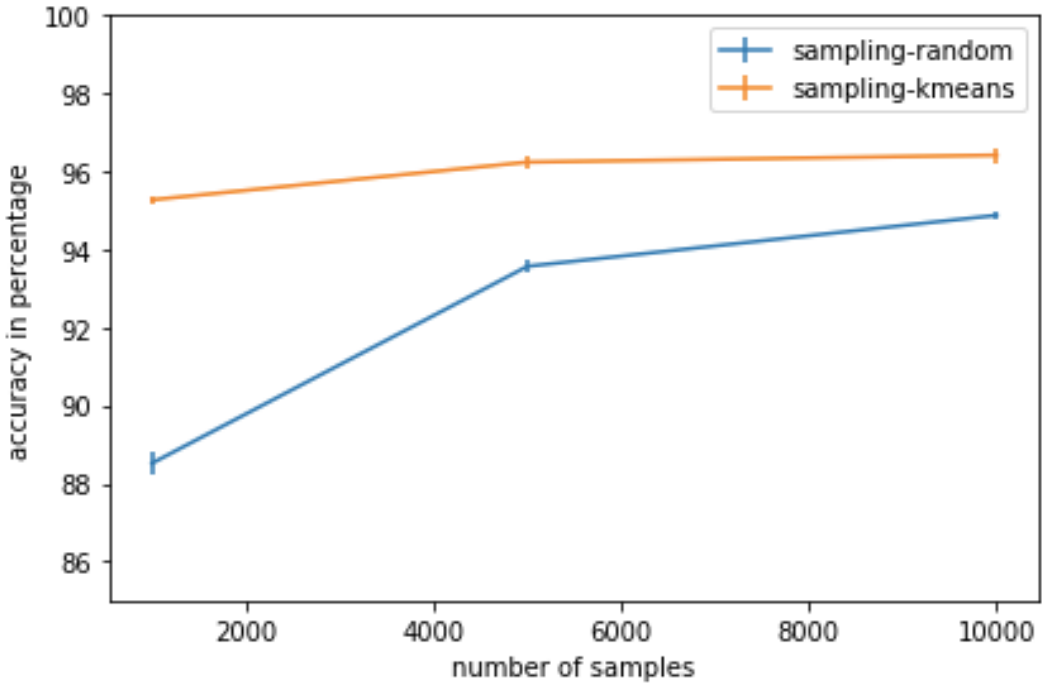
The t-score is calculated by looking at the standard t-distribution table after calculating degrees of freedom and a. Here,

$$degrees\ of\ freedom = sample\ size - 1$$

$$a = \frac{1 - Confidence\ level}{2}$$

Below is the result of the experiment with 95% confidence.

Number of Samples	Random Selection	K-means selection
1000	88.518 ± 0.269	95.274 ± 0.1100
5000	93.568 ± 0.147	96.240 ± 0.1543
10000	94.875 ± 0.104	96.419 ± 0.1948



4. **Critical evaluation:**
   Yes. It is a clear improvement over the random selection. Though the accuracy is nearly equal to using the full data set (around 97%), there is a scope of improvement since the state of art accuracy is over 99.5% using neural networks. I believe clustering could beat the accuracy of full data set by very careful clustering near the boundaries. If I have had more time, I would have devised a metric to calculate distance between each centroid and thus provided different weightage. I would have then measured its performance on the train data set and would have finally applied it on test set.