K-means

Group 04:

1. Kummer Lorenz, student ID – 00947188
2. Sidak Kevin, student ID – 01249373
3. Higgs Sam, student ID – 11844757
4. Setezhev Grigory, student ID – 11808684

Introduction

The K-means algorithm is used in the solution of the clustering problem. It aims to partition *n* given data points (or observations) into *k* different clusters. This algorithm dates back to 1957 by Stuart Lloyd and 1965 by E. W. Forgy.

Programming setup

Python (version 3) was used as a programming language for this assignment. It was chosen for it's ease of use and comprehensive list of available packages which could simplify the development of the program. Package-management system **pip** allowed us to use the following packages:

1. NumPy – library which supports multi-dimensional arrays and matrices and also provides the functions to operate on them. We also used it as a tool to import data from TXT and CSV files.
2. Scikit-learn – machine learning library which gave us the ability to calculate the Normalized Mutual Information score.
3. Matplotlib – plotting library which was used by us to visualize the clustering result.

Project structure

The project has the following structure:

1. kmeans.py – main class, where the data is read, pre- and post processed, metrics are calculated, execution time is measured and the results are presented using matplot.py.
2. setup.py – class which is required for setuptools working.

3. init_strategies.py – class which describes different techniques for the initial definition of clusters in k-means algorithm.
4. update_strategies.py – class where different algorithms to update cluster's centroids are described.
5. mq_run_script.py – separate script which is used for MacQueen update strategy.
6. experiment.py – benchmark tool which is used for calculation the average results among 100 runs.

We decided to use the strategy pattern for the different initialization and update strategies so we can easily switch out different algorithm during runtime, allowing for an easier setup of the experiments.

Initialization techniques

Our team implemented three different algorithms for initial definition of clusters in k-means algorithm:

1. Random initialization: given *n* data points, we choose random *k* among them and define them as initial our cluster centroids.
2. Farthest point initialization: at first, a point of the point cloud is selected at random and added to the set of initial centroids. Then from all points in the point cloud that have not already been chosen as centroids, the point that is farthest away from the set of centroids is chosen as new centroid and added to the set. This step is repeated until k centroids are found
3. Pre-Sampled Clustering Initialization: This initialization technique first choses k random points of the point cloud as initial centroids. Then an appropriate step size is chosen based on the average distance of all points to each other. Now in each iteration it increases the distance by one step size from the k centroids we selected randomly and adds all points which are in this distance/radius. This step is repeated until we completed all iterations or we want to add a point to a cluster while it is already part

of another cluster. Finally we calculate the new center for each cluster and return it.

Updating techniques

We have implemented two different algorithms for updating the clusters:

1. Lloyd algorithm – cluster centroids are updated after all point's affiliation with clusters are recalculated.
2. MacQueen algorithm – cluster centroids are updated after every single one recalculation of point's cluster membership.

Dataset description

Two datasets were used in our k-means implementation:

1. Skin Segmentation Data Set – RGB values from face images of various age, race groups and genders. They are taken from FERET and PAL databases. 50859 instances are skin samples, and 194198 are non-skin samples (artificially generated values).
2. HTRU2 Data Set - a sample of pulsar candidates collected during the High Time Resolution Universe Survey. There are 8 continuous variables describing each instance in dataset. 1639 instances are real pulsar examples and 16259 are fake.

Metrics

For the evaluation of the results, two metrics were used:

1. **Normalized Mutual Information (NMI)** measures the similarity between two labels of the same data. If $|U_i|$ is the number of samples in cluster U_i, and $|V_j|$ is the number of samples in cluster V_j, their mutual information equals

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j-1}^{|V|}\frac{|U_i \cap V_j|}{N}\log\frac{N|U_i \cap V_j|}{|U_i||V_j|}$$

   Then, mutual information is normalized.
   NMI score has values between 0 and 1, where 0 means "no mutual information" and 1 mean "perfect correlation". Ideal k-means algorithm would provide the result with NMI score equals 1.

2. **Within-cluster sum of squares (WCSS)** is the sum of squared deviations from each observation and the cluster centroid. If ($\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$) is the set of observations, and $\mathbf{S} = \{S_1, S_2, ..., S_k\}$ are the clusters with $\boldsymbol{\mu}_i$ - the mean of points in S_i, then WCSS is:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i-1}^{k} |S_i| \operatorname{Var} S_i$$

The lower the metric, the better, because it means that the cluster is more compact.

Result of k-means algorithm

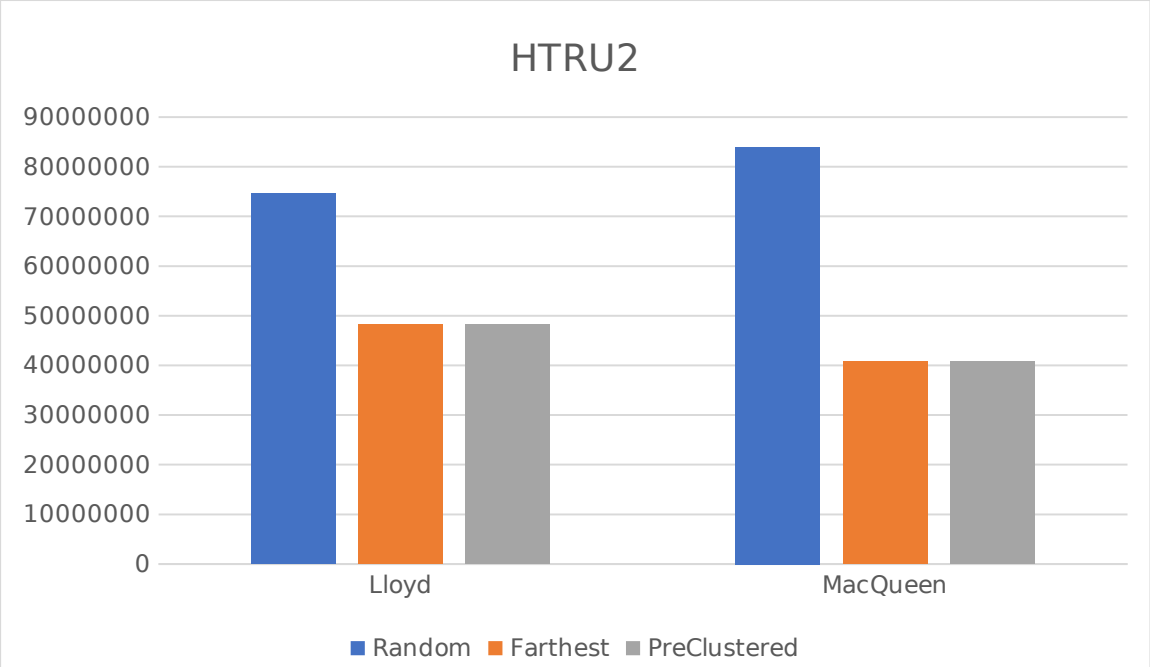
We evaluated six possible combinations of k-means algorithm (three initialization strategies multiply by two update strategies) for each data set. As result metrics NMI and WCSS were chosen as an average of 100 runs of each combination.

Results using the HTRU2 data set:

WCSS	Random
Lloyd	74544592.5012591
MacQueer	84000670.7533754

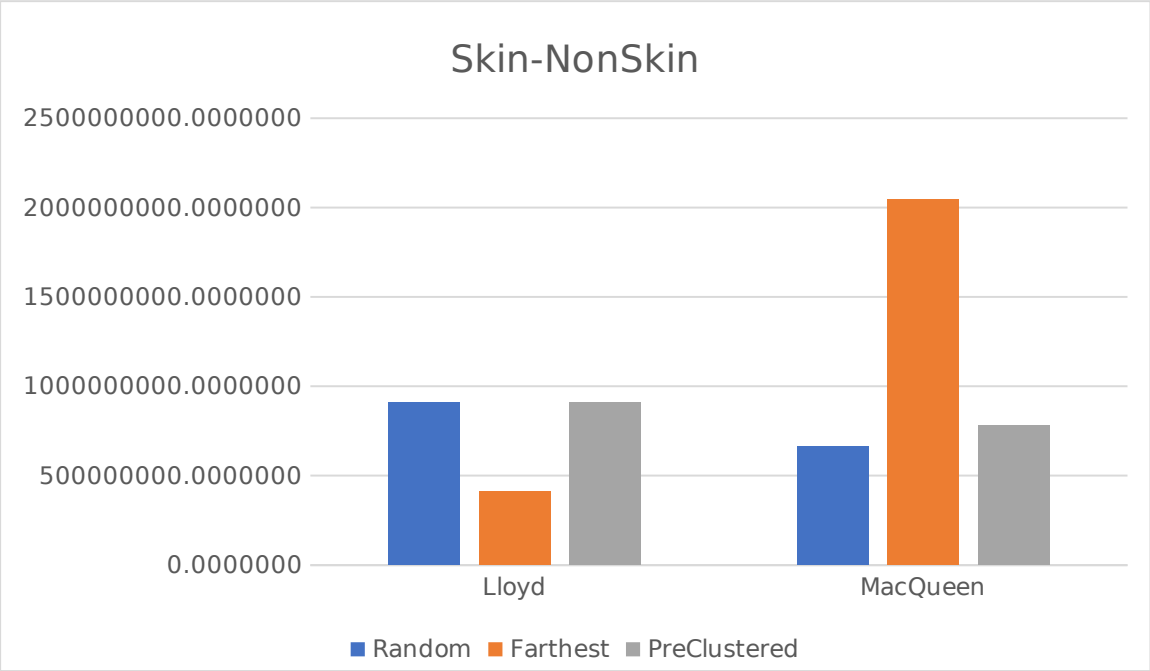
NMI	Random	Farthest
Lloyd		
MacQueen		

Let's visualize results:



Reults using the Skin-NonSkin dataset:

WCSS	Random
Lloyd	910993566.646362
MacQueen	661299441.787527



Conclusion

If we compare different algorithms by using WCSS metric, we may notice that MacQueen update strategy performs better than Lloyd, because the resulting clusters are more compact. Comparing different initialization algorithms, we conclude that random initialization leads to worst results, while “Farthest” and “PreClustered” have better performance, but their score is identical.