SS20

universität wien

K Means

SCIENTIFIC DATA MANAMEMENT SS20

JOANNA KALETA
JAN DUBIŃSKI
VINCENT REINER
MARCEL ANDREU Y CASADESUS

Implementation description

The purpose of this section is to introduce the reader to the methods and approaches applied in this work. Kmeans algorithm was implemented with Python 3 in Google Collaboratory. The source code is available at https://github.com/joaxkal/SDM/.

Update strategies

According to the requirements we implemented two different update strategies:

1. Lloyd – the algorithm updates each round.
2. MacQueen – the algorithm updates immediately after assignment of each point.

Since both methods have been introduced in the lecture, we omitted the descriptions here.

Initialization strategies

We implemented three different initialization strategies mentioned in paper [1]. Short description of each strategy is provided below.

1. Forgy's (*forgy*)
   In this strategy each datapoint is assigned randomly to one of the clusters. Based on the assignment initial centroids are determined. This strategy has no scientific theoretical basis.

2. Kmeans++ (*kmpp*)
   This strategy chooses the first center randomly and the each next center is chosen with probability proportional to its squared distance from the point's closest existing cluster center. More detailed information can be found in paper [2].

   Figure below presents the algorithm of kmeans++ initialization strategy. For the k-means problem we are given an integer k representing the number of centers C and a set of n data points $X \epsilon R^d$. Let $D(x)$ denote the shortest distance from a data point to the closest center we have already chosen.

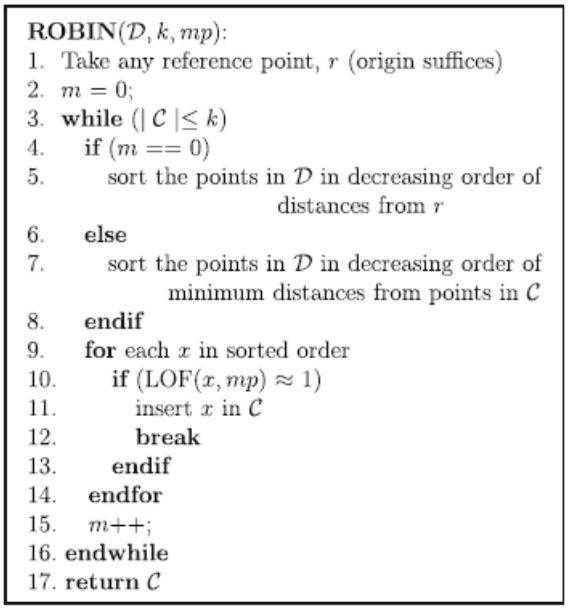
   Kmeans++

   1. Take one center c_1, chosen uniformly at random from $\mathcal{X}$.
   2. Take a new center c_i, choosing $x \in \mathcal{X}$ with probability $\frac{D(x)^2}{\sum_{x \in \mathcal{X}} D(x)^2}$.
   3. Repeat Step 2. until we have taken k centers altogether.

3. ROBIN (*robin*)
   The ROBIN (ROBust Initialization) strategy uses a local outlier factor which allows to avoid determining outlier points as centroids. Outliers are those points whose density is very different compared to neighbour densities. If a point is in a low density neighbourhood compared to all its neighbours, then its score is low and hence its LOF value is high. A point that belongs to a cluster has LOF value approximately equal to 1. In the proposed algorithm we do not compute the density or LOF value of all the points, since that is computationally expensive. We compute LOF value on demand for just one datapoint at a time. Moreover ROBIN ensures that the seeds are as far apart as possible. For any reference point r, ROBIN first sorts the points in decreasing order of their distances from r. In this sorted order, it selects the first point that is not an outlier as validated through its LOF value. The subsequent seed points

are obtained in a similar manner, by first sorting the points in decreasing order of their minimum distance to seed centers already in the set *C*. Detailed information can be found in paper [3].

The algorithm of ROBIN strategy is presented in the figure. In the figure *D* is the dataset; *k* is the number of clusters or seeds desired, and *mp* is the number of neighbours to consider while computing the LOF. *C* is the set of centroids.



,

Data preprocessing

Conducted tests of the kmeans algorithm included also three different data preprocessing approaches:

1. Raw data - no preprocessing
2. MinMaxScaler - Transformation of features by scaling each feature to a given range.
   This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one. For this purpose, we used Sklearn implementation sklearn.preprocessing.MinMaxScaler.
3. StandardScaler - Standardization of features by removing the mean and scaling to unit variance
   The standard score of a sample *x* is calculated as *z = (x - u) / s* where *u* is the mean of the training samples and *s* is the standard deviation of the training samples. For this purpose, we used Sklearn implementation sklearn.preprocessing.StandardScaler.

Test datasets

According to the assignment requirements tests were conducted on two datasets:

1. Skin dataset – The skin dataset is collected by randomly sampling B,G,R values from face images of various age groups, race groups, and genders. Total learning sample size is 245 057; out of which 50 859 is the skin samples and 194 198 is non-skin samples.

2. HTRU_2 dataset – Dataset contains pulsar candidates collected during the HTRU survey which must be classified in to pulsar and non-pulsar classes to aid discovery. Dataset consists of 17 898 instances. Each candidate is described by 8 continuous variables and a single class variable.

Results and discussion

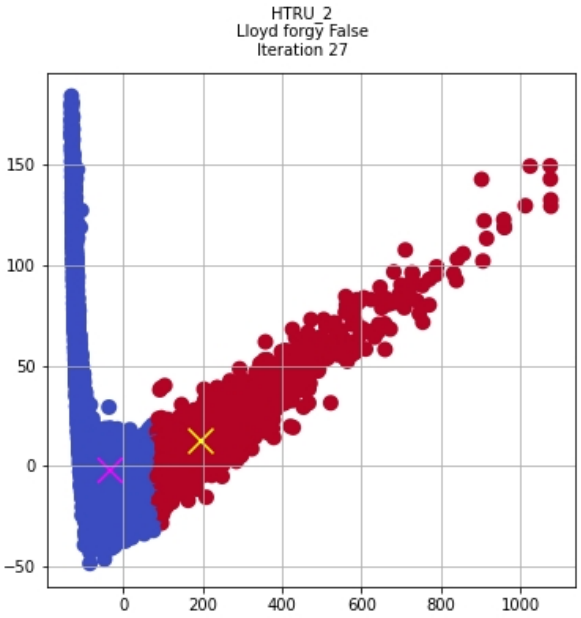
This part of the document describes the obtained results. Firstly, we present sample outcomes of clustering visualised using PCA. Secondly, we include a table with mean results for 100 runs of each combination of algorithm, initialization strategy and normalization scheme. Then, the most important metrics: NMI, number of iterations required to converge and time to converge are presented in form of bar plots. We also discuss our observations and present the outcome of the kmeans algorithm visualised using PCA.

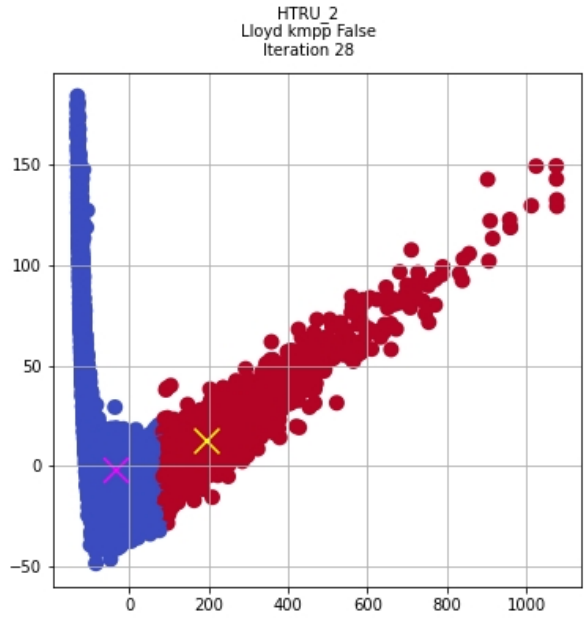
Scatter plots

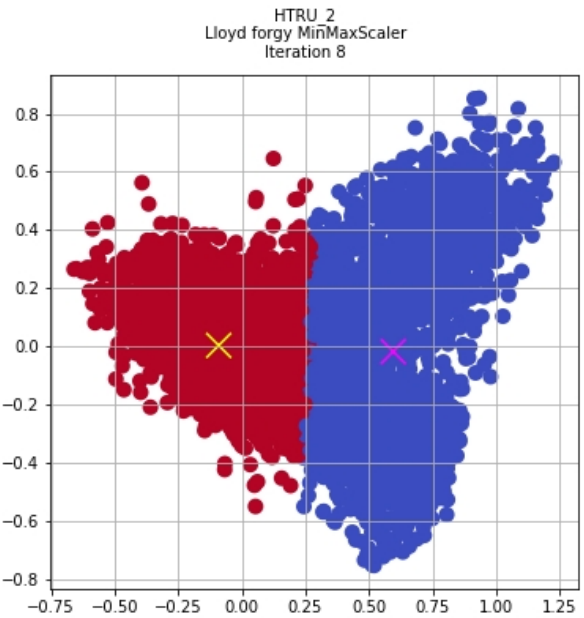
The plots presented below include clustering results from all combinations of update strategies, initialization strategies and data preprocessing methods for both datasets. As we can notice immediately, data normalization had a significant impact on HTRU_2 dataset. The shapes of PCA scatter plots vary a lot between three different data preprocessing approaches. On the contrary the skin dataset does not show such a tendency - we assume that skin dataset was already normalized. This assumption explains why after data normalization application only for the HTRU_2 dataset the great quality improvement of clustering was observed.

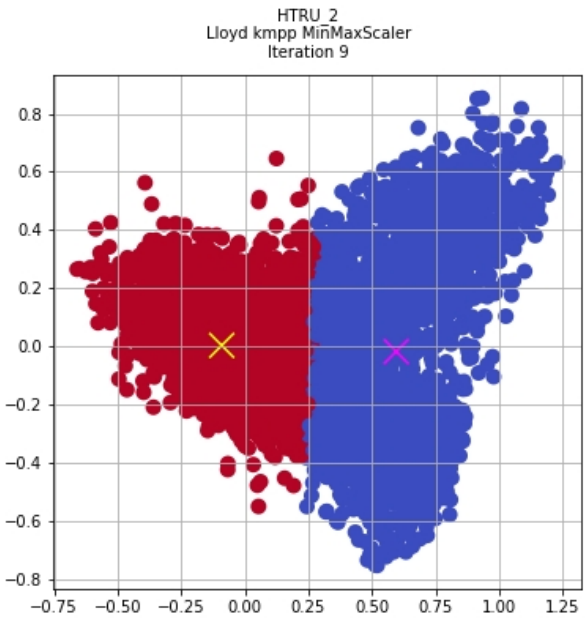
However we did not observe a significant difference in clustering after different update and initialization strategies were used. We assume that the algorithm in vast majority of cases found the same local minimum.

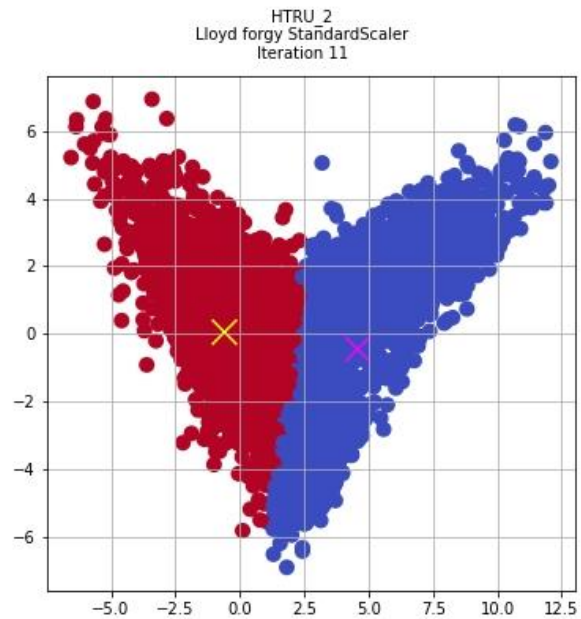
The detailed measurement results like mean number of iterations, total time and NMI were presented in the next section of the paper. To make the results more comparable we also measured the same values for Sklearn implementation of Kmeans algorithm.

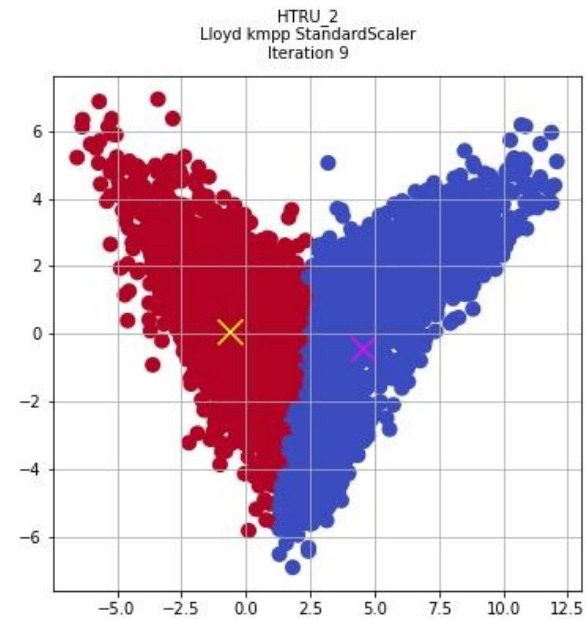

HTRU_2
Lloyd forgy False
Iteration 27

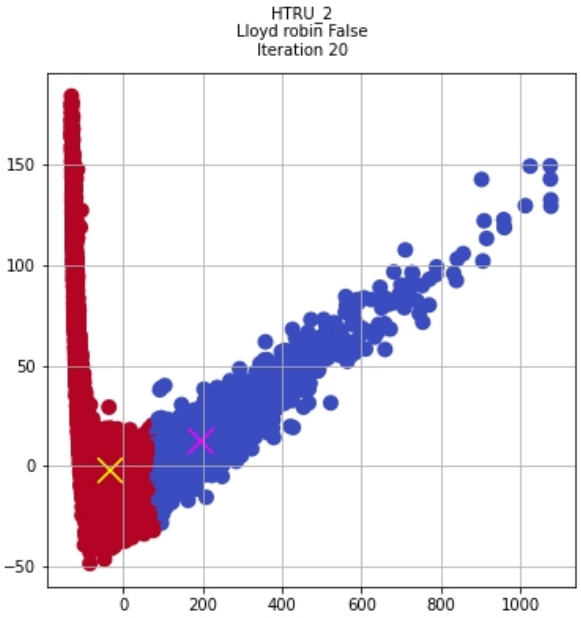

HTRU_2
Lloyd kmpp False
Iteration 28

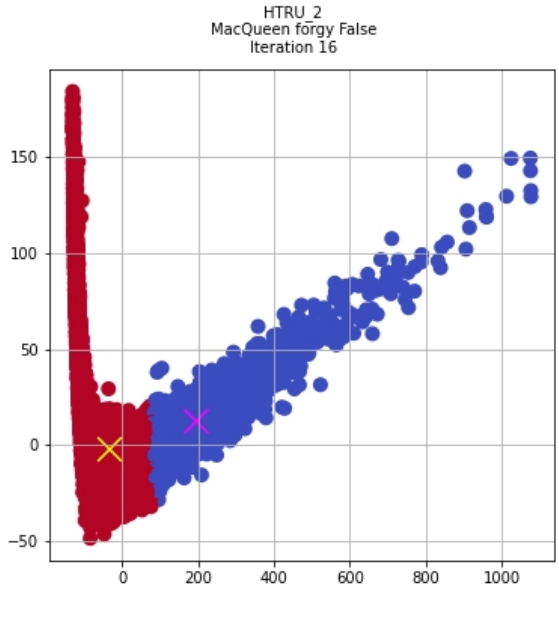

HTRU_2
Lloyd forgy MinMaxScaler
Iteration 8

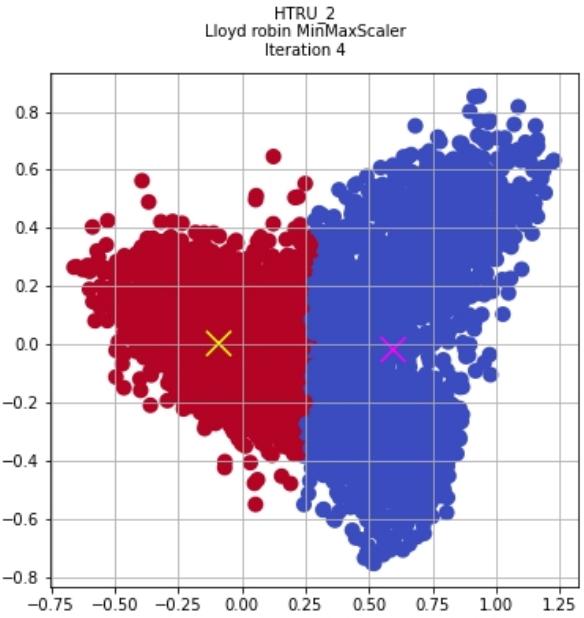

HTRU_2
Lloyd kmpp MinMaxScaler
Iteration 9

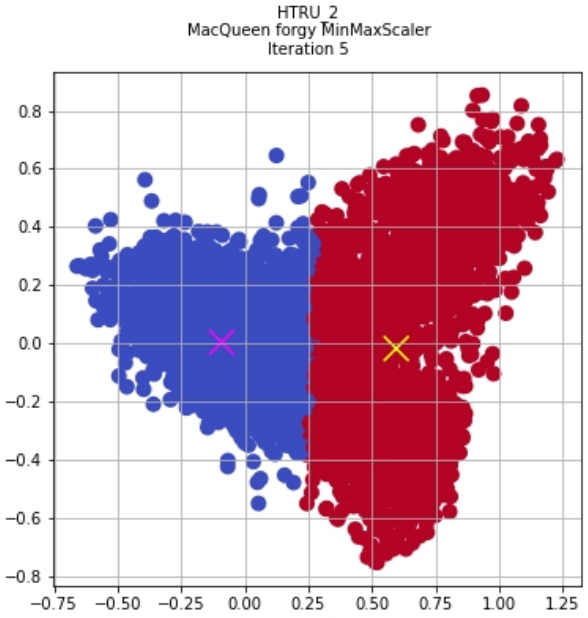

HTRU_2
Lloyd forgy StandardScaler
Iteration 11

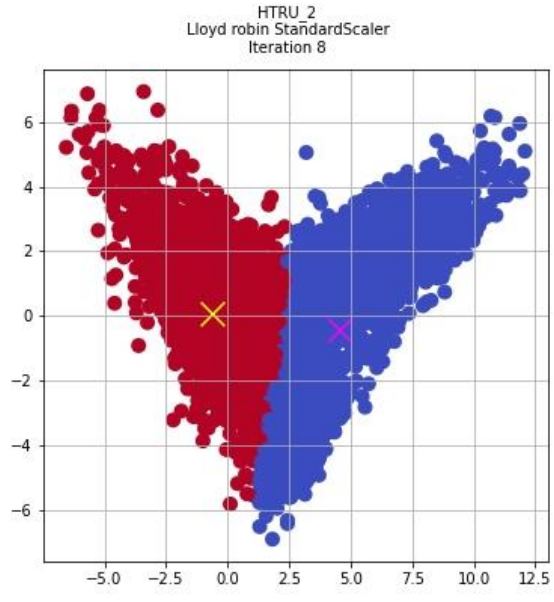

HTRU_2
Lloyd kmpp StandardScaler
Iteration 9

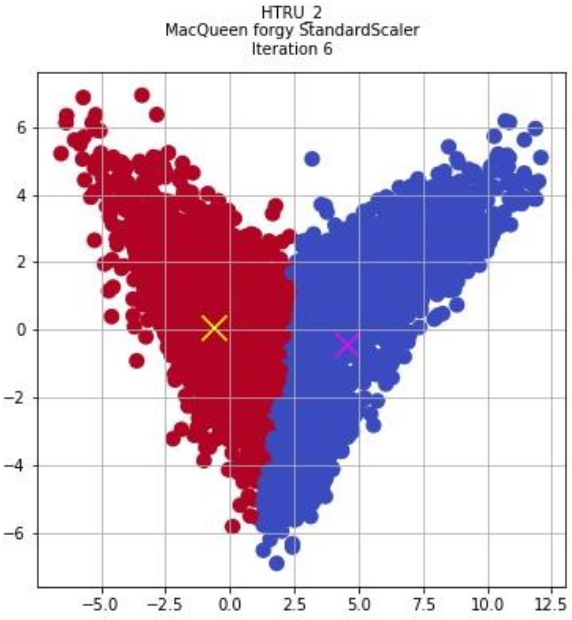

HTRU_2
Lloyd robin False
Iteration 20

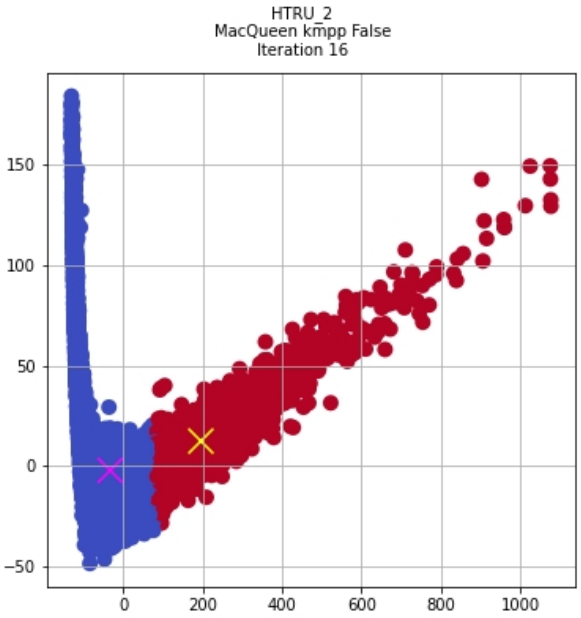

HTRU_2
MacQueen forgy False
Iteration 16

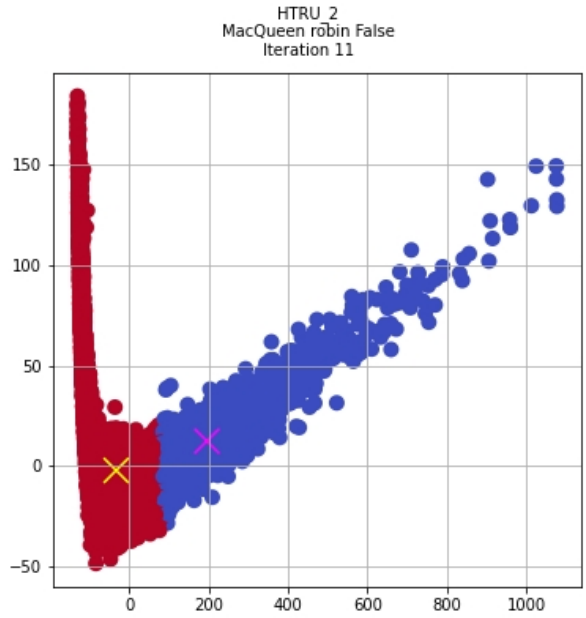

HTRU_2
Lloyd robin MinMaxScaler
Iteration 4

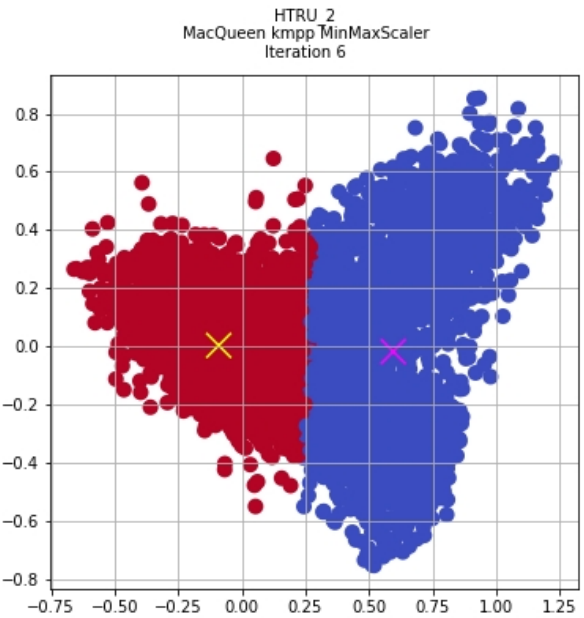

HTRU_2
MacQueen forgy MinMaxScaler
Iteration 5

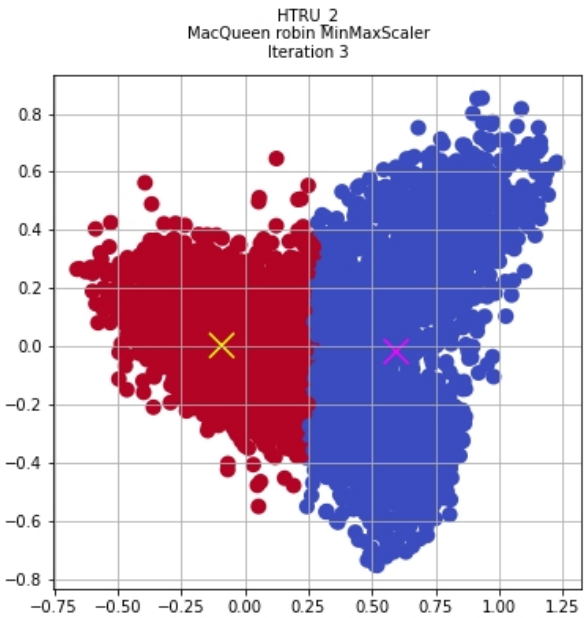

HTRU_2
Lloyd robin StandardScaler
Iteration 8

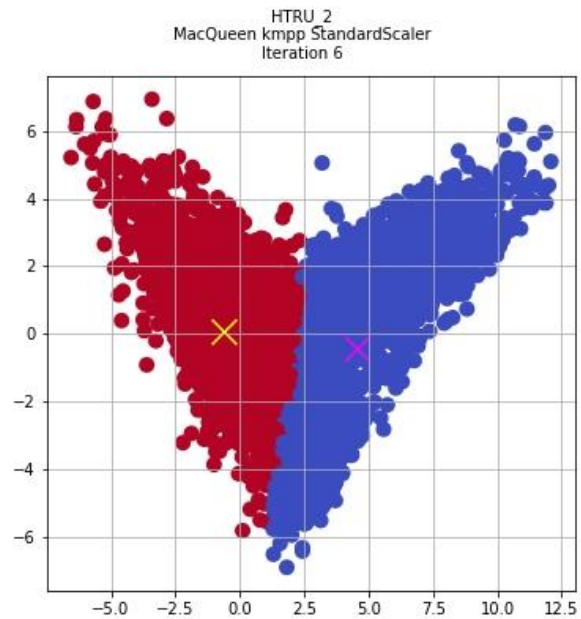

HTRU_2
MacQueen forgy StandardScaler
Iteration 6

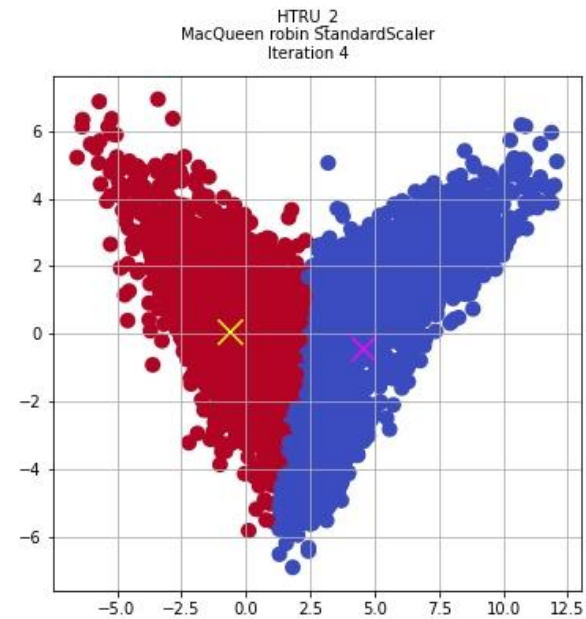

HTRU_2
MacQueen kmpp False
Iteration 16

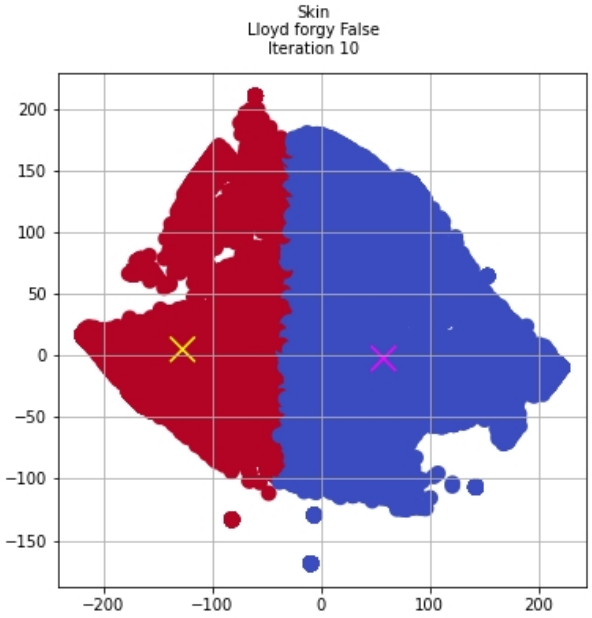

HTRU_2
MacQueen robin False
Iteration 11

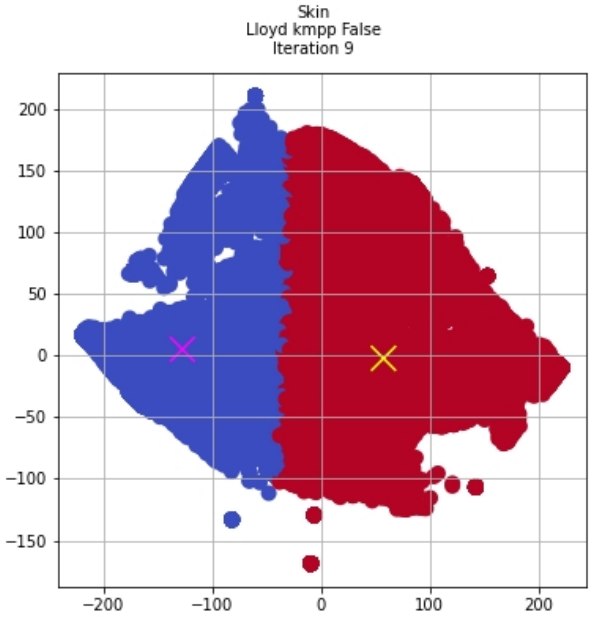

HTRU_2
MacQueen kmpp MinMaxScaler
Iteration 6

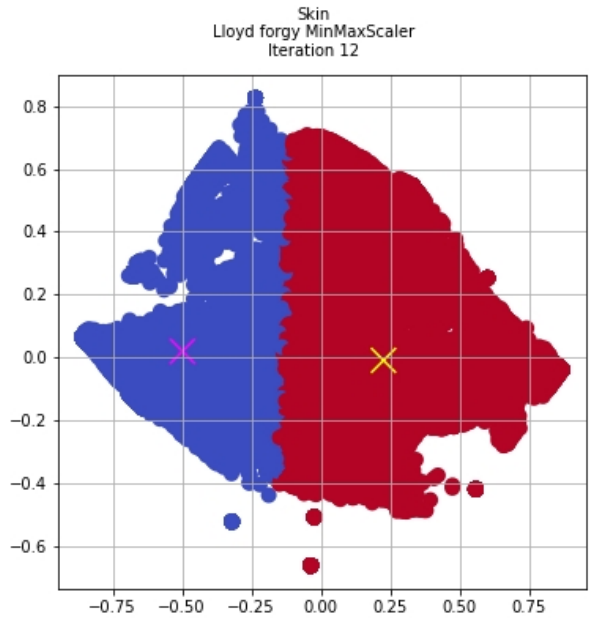

HTRU_2
MacQueen robin MinMaxScaler
Iteration 3

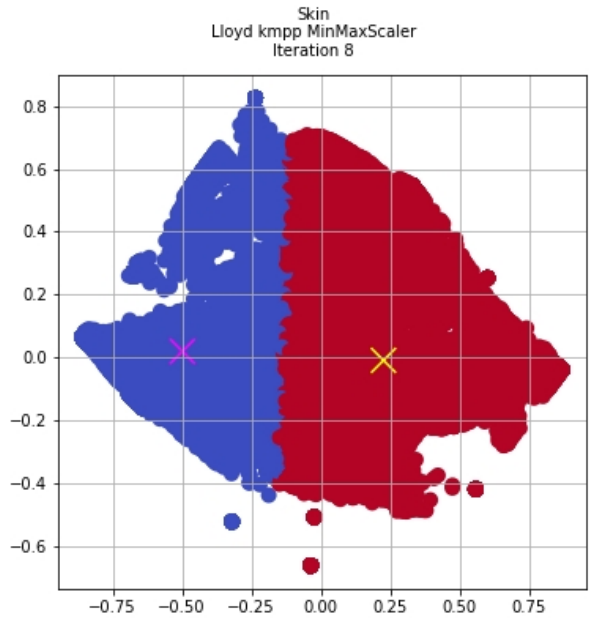

HTRU_2
MacQueen kmpp StandardScaler
Iteration 6

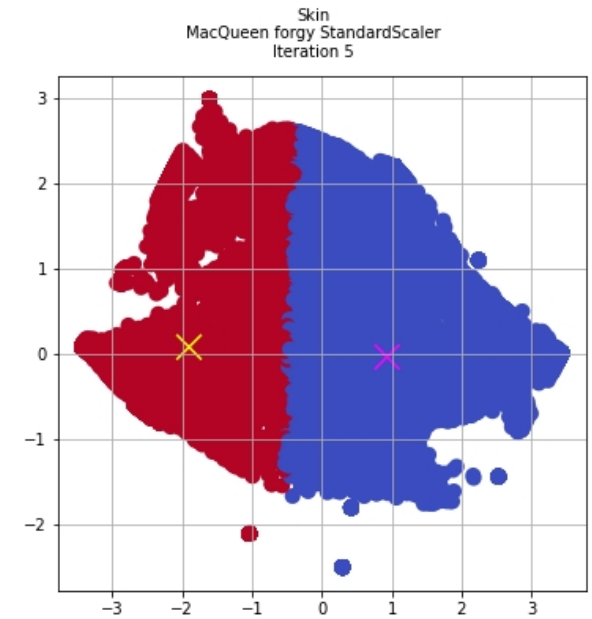

HTRU_2
MacQueen robin StandardScaler
Iteration 4

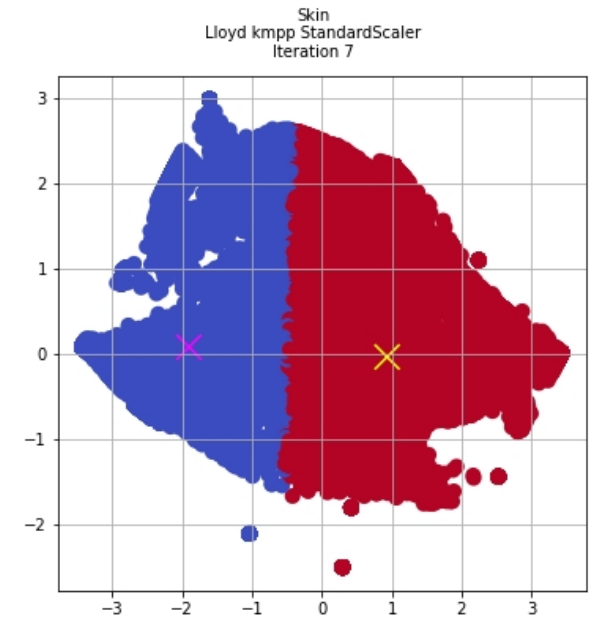

Skin
Lloyd forgy False
Iteration 10

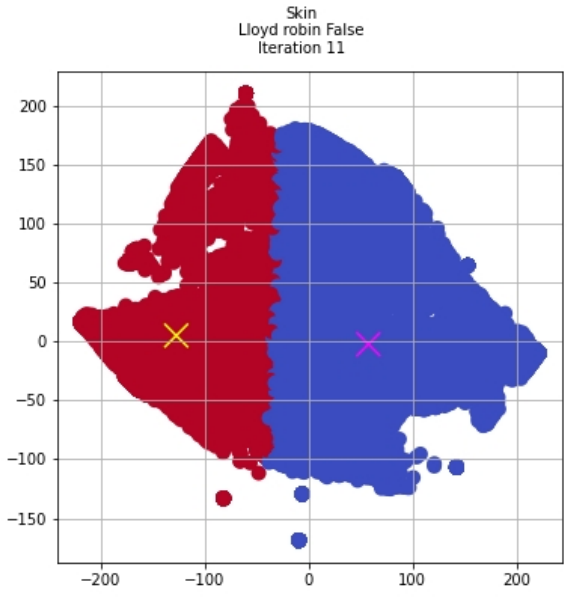

Skin
Lloyd kmpp False
Iteration 9

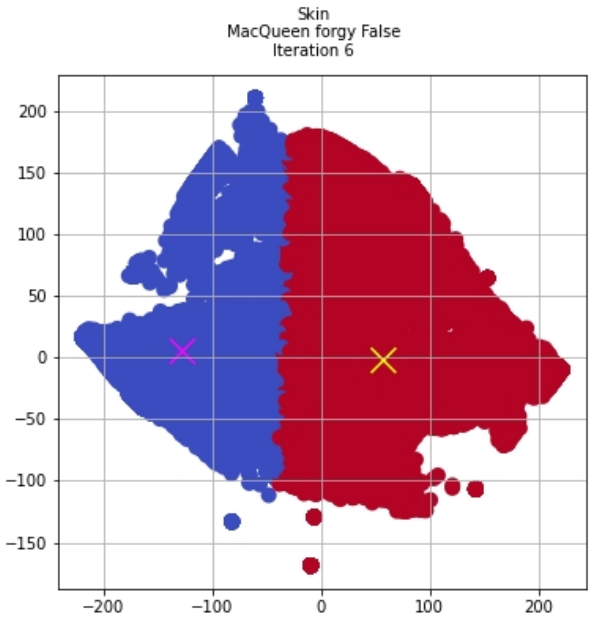

Skin
Lloyd forgy MinMaxScaler
Iteration 12

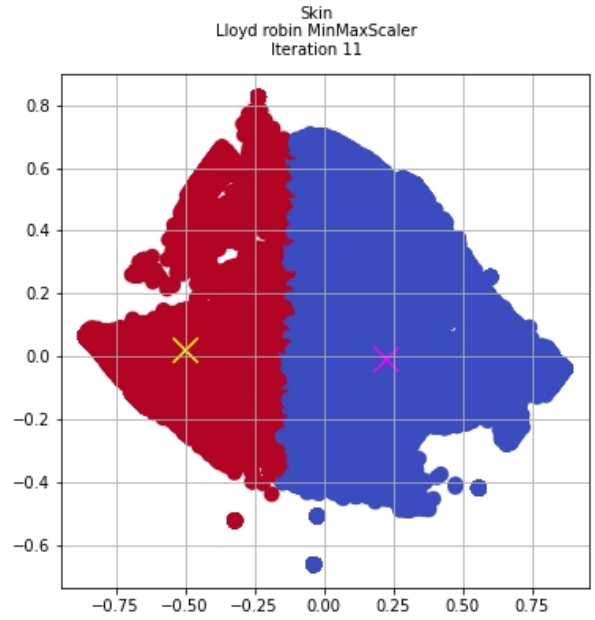

Skin
Lloyd kmpp MinMaxScaler
Iteration 8

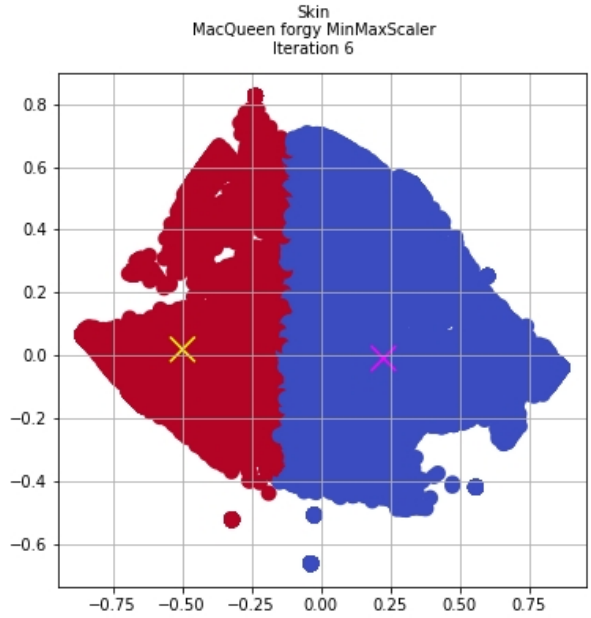

Skin
MacQueen forgy StandardScaler
Iteration 5

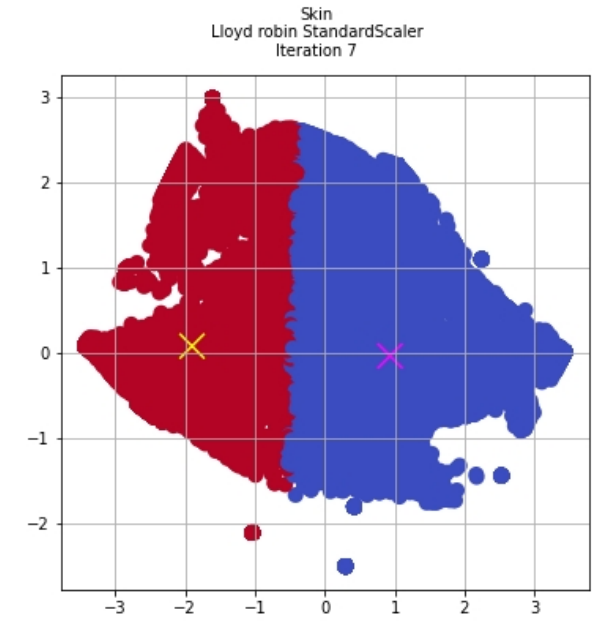

Skin
Lloyd kmpp StandardScaler
Iteration 7

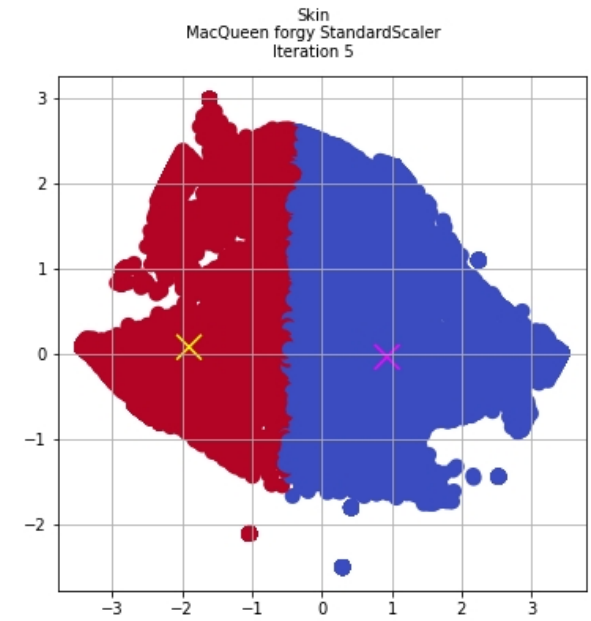

Skin
Lloyd robin False
Iteration 11

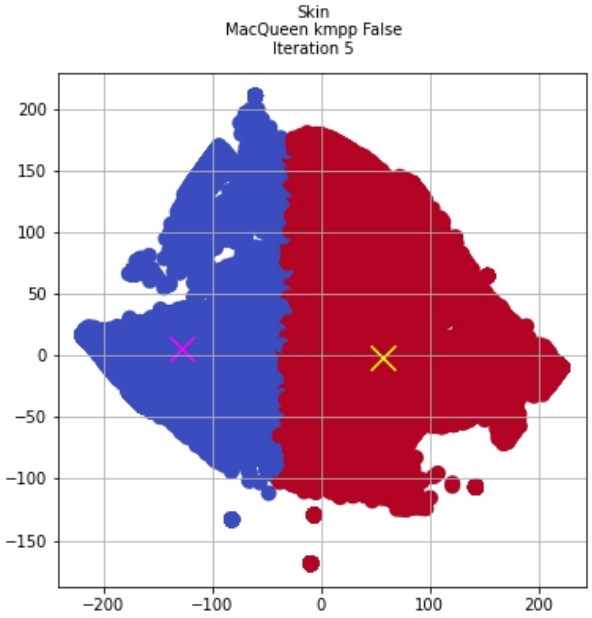

Skin
MacQueen forgy False
Iteration 6

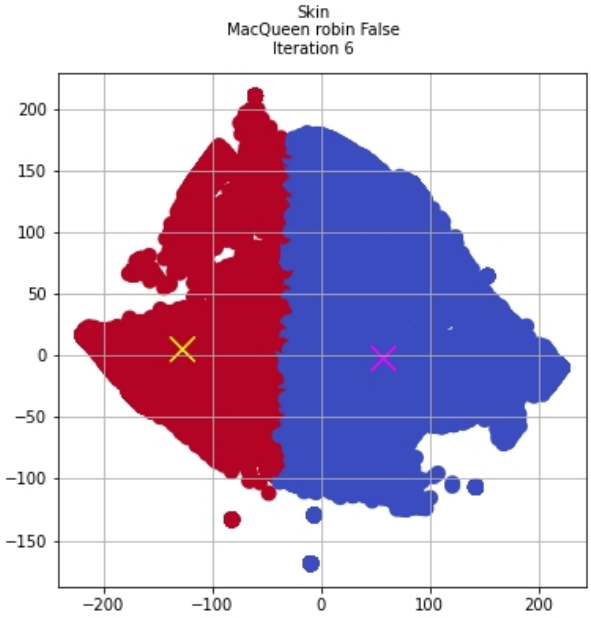

Skin
Lloyd robin MinMaxScaler
Iteration 11

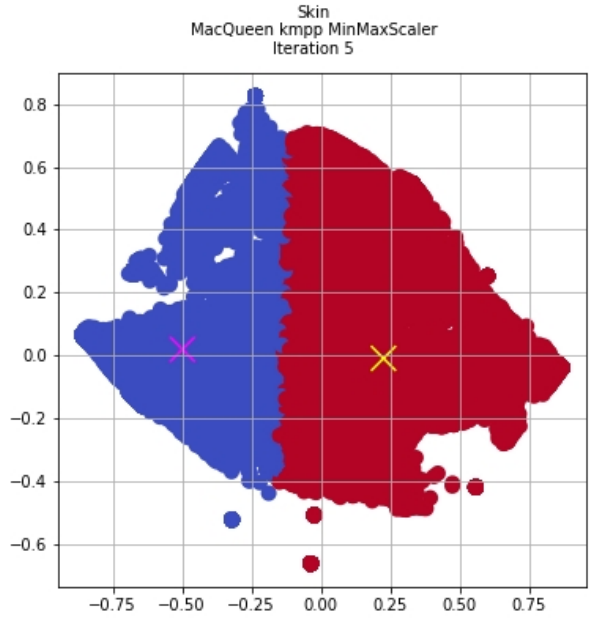

Skin
MacQueen forgy MinMaxScaler
Iteration 6

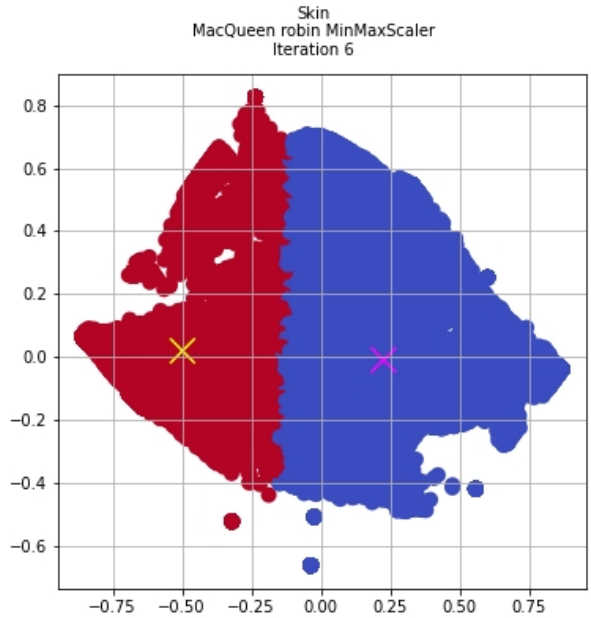

Skin
Lloyd robin StandardScaler
Iteration 7

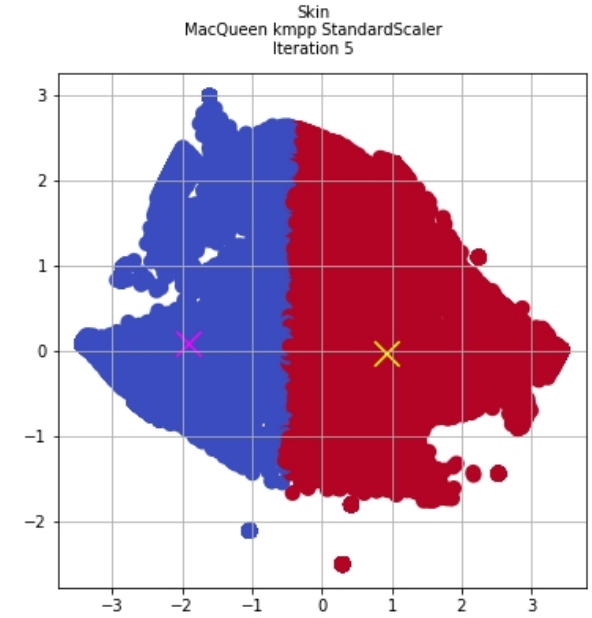

Skin
MacQueen forgy StandardScaler
Iteration 5

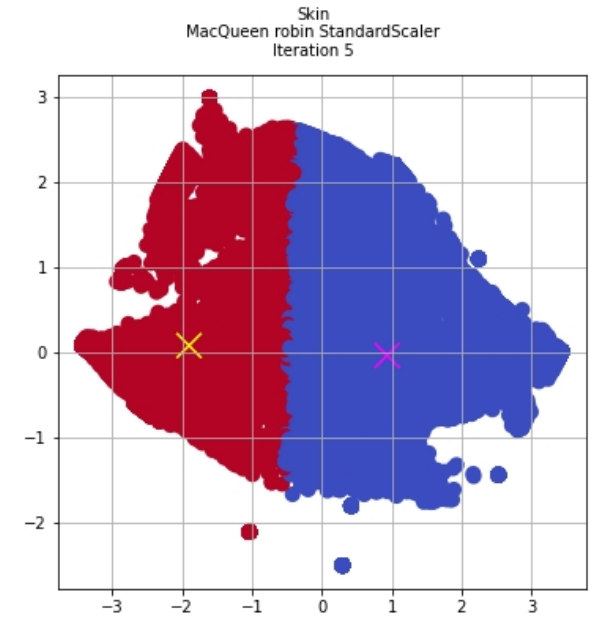

Skin
MacQueen kmpp False
Iteration 5


Skin
MacQueen robin False
Iteration 6


Skin
MacQueen kmpp MinMaxScaler
Iteration 5


Skin
MacQueen robin MinMaxScaler
Iteration 6


Skin
MacQueen kmpp StandardScaler
Iteration 5


Skin
MacQueen robin StandardScaler
Iteration 5

Results

Best results are marked by red colour. Explanation of column labels is presented below:

- NMI is the Normalized Mutual Info score calculated for our implementation of the algorithm
- Sklearn NMI is the reference value obtained by the state-of-the-art implementation of kmeans from sklearn Python library
- time is the required time in seconds for our implementation to converge
- iterations is the number of iterations before our algorithm converges

Notice that the *False* value in the column 'normalize' means that no data normalization scheme was used. In other cases the method of normalization was given.

	info				mean				std			
	dataset	**algorithm**	**init method**	**normalize**	**NMI**	**sklearn NMI**	**time**	**iterations**	**NMI**	**sklearn NMI**	**time**	**iterations**
1	**skin**	**MacQueen**	**robin**	**False**	0,023287	0,023405	26,76465	6	2,44E-17	0,00011335	0,193888	0
2	**skin**	**MacQueen**	**robin**	**MinMaxScaler**	0,023287	0,023405	26,87574	6	2,44E-17	0,00011336	0,464629	0
3	**skin**	**MacQueen**	**robin**	**StandardScaler**	0,014582	0,014596	24,25887	**5**	2,79E-17	2,47E-05	0,367869	0
4	**skin**	**MacQueen**	**kmpp**	**False**	**0,023397**	0,023443	23,00387	**5**	6,97E-18	8,75E-05	0,177137	0
5	**skin**	**MacQueen**	**kmpp**	**MinMaxScaler**	0,023302	0,023443	25,30393	5,85	3,85E-05	8,76E-05	1,152596	0,411329
6	**skin**	**MacQueen**	**kmpp**	**StandardScaler**	0,014584	0,014593	22,89722	**5**	1,74E-17	2,84E-05	0,284431	0
7	**skin**	**MacQueen**	**forgy**	**False**	0,023287	0,023426	22,36986	6,2	2,44E-17	9,99E-05	1,423918	0,402015
8	**skin**	**MacQueen**	**forgy**	**MinMaxScaler**	0,023287	0,023426	22,41239	6,22	2,44E-17	9,99E-05	1,413058	0,416333
9	**skin**	**MacQueen**	**forgy**	**StandardScaler**	0,014582	0,014592	**19,54**	5,13	2,79E-17	3,02E-05	1,376499	0,366667
10	**skin**	**Lloyd**	**robin**	**False**	0,023287	0,023405	39,78104	11	2,44E-17	0,00011335	1,883536	0
11	**skin**	**Lloyd**	**robin**	**MinMaxScaler**	0,023287	0,023405	40,8114	11	2,44E-17	0,00011336	1,960368	0
12	**skin**	**Lloyd**	**robin**	**StandardScaler**	0,014582	0,014596	29,79864	7	2,79E-17	2,47E-05	1,359991	0
13	**skin**	**Lloyd**	**kmpp**	**False**	**0,023397**	0,023443	34,96081	9	6,97E-18	8,75E-05	2,073807	0
14	**skin**	**Lloyd**	**kmpp**	**MinMaxScaler**	0,023302	0,023443	38,79959	9,72	3,85E-05	8,76E-05	1,967996	0,711805
15	**skin**	**Lloyd**	**kmpp**	**StandardScaler**	0,014584	0,014593	30,88943	6,8	1,74E-17	2,84E-05	1,282678	0,402015
16	**skin**	**Lloyd**	**forgy**	**False**	0,023287	0,023426	36,84472	10,84	2,44E-17	9,99E-05	2,26492	0,748331
17	**skin**	**Lloyd**	**forgy**	**MinMaxScaler**	0,023287	0,023426	36,49745	10,97	2,44E-17	9,99E-05	2,629242	0,968754
18	**skin**	**Lloyd**	**forgy**	**StandardScaler**	0,014582	0,014592	26,88721	7,41	2,79E-17	3,02E-05	2,104445	0,792579

	info				min				max			
	dataset	**algorithm**	**init method**	**normalize**	**NMI**	**sklearn NMI**	**time**	**iterations**	**NMI**	**sklearn NMI**	**time**	**iterations**
1	**skin**	**MacQueen**	**robin**	**False**	0,023287	0,023164	26,26249	6	0,023287	0,0235275	27,36079	6
2	**skin**	**MacQueen**	**robin**	**MinMaxScaler**	0,023287	0,023164	26,4553	6	0,023287	0,0235275	31,1229	6
3	**skin**	**MacQueen**	**robin**	**StandardScaler**	0,014582	0,01452	23,74157	5	0,014582	0,01465316	27,41495	5
4	**skin**	**MacQueen**	**kmpp**	**False**	0,023397	0,023164	22,61966	5	0,023397	0,0235275	23,41871	5
5	**skin**	**MacQueen**	**kmpp**	**MinMaxScaler**	0,023287	0,023164	20,34326	4	0,023397	0,0235275	28,70198	7
6	**skin**	**MacQueen**	**kmpp**	**StandardScaler**	0,014584	0,01452	22,46239	5	0,014584	0,01465316	24,34043	5
7	**skin**	**MacQueen**	**forgy**	**False**	0,023287	0,023164	21,13042	6	0,023287	0,02352636	26,0592	7
8	**skin**	**MacQueen**	**forgy**	**MinMaxScaler**	0,023287	0,023164	21,0534	6	0,023287	0,02352636	25,83536	7
9	**skin**	**MacQueen**	**forgy**	**StandardScaler**	0,014582	0,01452	18,65634	5	0,014582	0,01464466	25,74427	7
10	**skin**	**Lloyd**	**robin**	**False**	0,023287	0,023164	36,94242	11	0,023287	0,0235275	43,22176	11
11	**skin**	**Lloyd**	**robin**	**MinMaxScaler**	0,023287	0,023164	37,45357	11	0,023287	0,0235275	44,98955	11
12	**skin**	**Lloyd**	**robin**	**StandardScaler**	0,014582	0,01452	27,49551	7	0,014582	0,01465316	33,57685	7

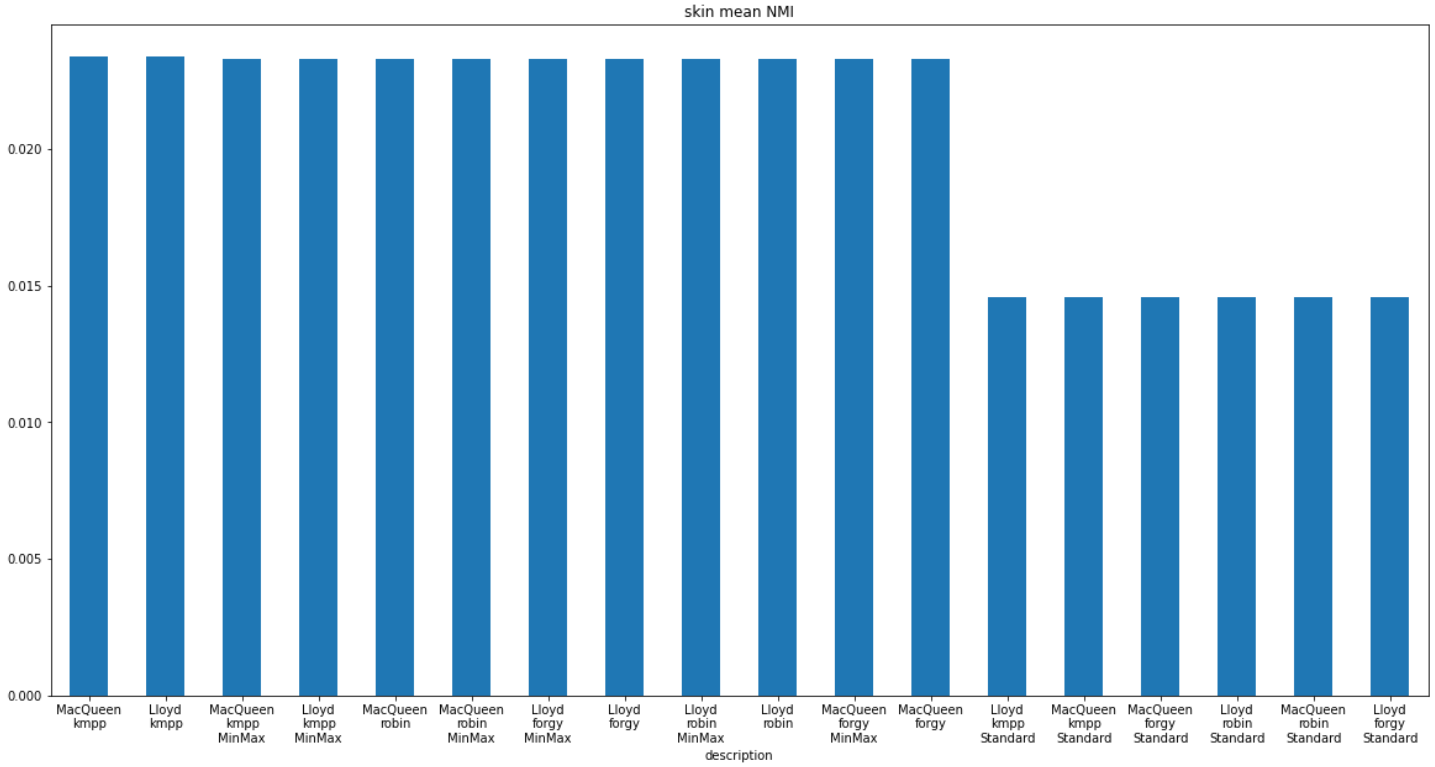
13	**skin**	**Lloyd**	**kmpp**	**False**	0,023397	0,023164	31,35023	9	0,023397	0,0235275	39,41904	9
14	**skin**	**Lloyd**	**kmpp**	**MinMaxScaler**	0,023287	0,023164	33,4115	8	0,023397	0,0235275	43,09892	11
15	**skin**	**Lloyd**	**kmpp**	**StandardScaler**	0,014584	0,01452	27,88782	6	0,014584	0,01465316	33,44086	7
16	**skin**	**Lloyd**	**forgy**	**False**	0,023287	0,023164	31,67455	10	0,023287	0,02352636	44,20876	13
17	**skin**	**Lloyd**	**forgy**	**MinMaxScaler**	0,023287	0,023164	33,01059	10	0,023287	0,02352636	44,65348	14
18	**skin**	**Lloyd**	**forgy**	**StandardScaler**	0,014582	0,01452	23,583	6	0,014582	0,01464466	34,02425	10

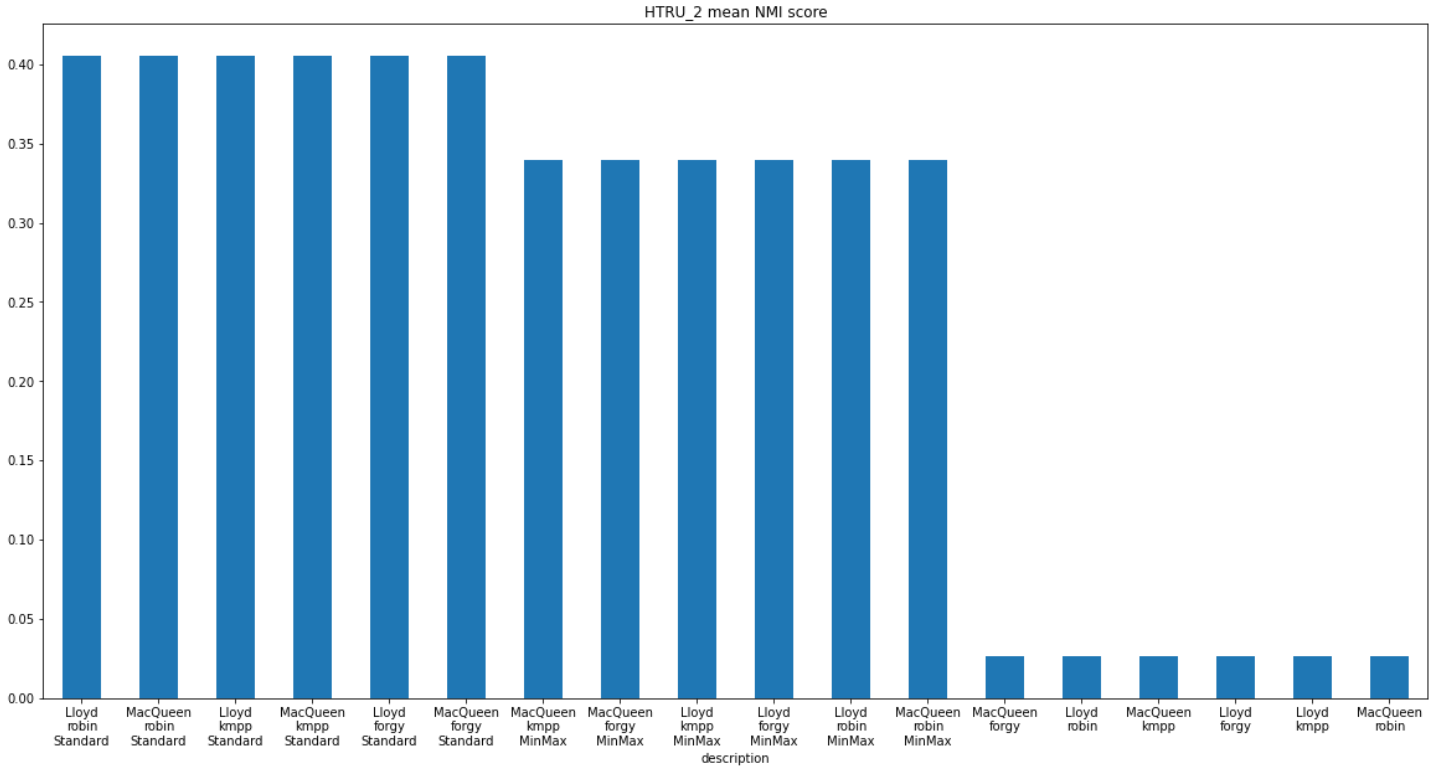
	info				**mean**				**std**			
	dataset	**algorithm**	**init method**	**normalize**	**NMI**	**sklearn NMI**	**time**	**iterations**	**NMI**	**sklearn NMI**	**time**	**iterations**
1	**HTRU_2**	**Lloyd**	**robin**	**False**	0,026497	0,026457	4,951487	20	1,39E-17	7,21E-05	0,087408	0
2	**HTRU_2**	**Lloyd**	**robin**	**MinMaxScaler**	0,339838	0,339707	1,988489	4	2,23E-16	0,000206	0,046369	0
3	**HTRU_2**	**Lloyd**	**robin**	**StandardScaler**	**0,405581**	0,405406	2,854693	8	7,25E-16	0,000224	0,059554	0
4	**HTRU_2**	**Lloyd**	**kmpp**	**False**	0,026497	0,026464	6,092178	27,44	1,39E-17	8,43E-05	0,142809	0,498888
5	**HTRU_2**	**Lloyd**	**kmpp**	**MinMaxScaler**	0,339838	0,339689	2,819113	10,28	2,23E-16	0,000213	0,275651	1,436044
6	**HTRU_2**	**Lloyd**	**kmpp**	**StandardScaler**	0,405379	0,405397	3,623909	14,6	0,00023	0,000227	0,912939	4,988877
7	**HTRU_2**	**Lloyd**	**forgy**	**False**	0,026497	0,026475	5,794192	27,7	1,39E-17	9,93E-05	0,19594	0,810287
8	**HTRU_2**	**Lloyd**	**forgy**	**MinMaxScaler**	0,339838	0,33968	2,280988	9,18	2,23E-16	0,000216	0,315554	1,665939
9	**HTRU_2**	**Lloyd**	**forgy**	**StandardScaler**	0,405287	0,405425	2,866554	12,18	0,000222	0,000219	0,958379	5,280764
10	**HTRU_2**	**MacQueen**	**robin**	**False**	0,026497	0,026457	3,317213	11	1,39E-17	7,21E-05	0,058071	0
11	**HTRU_2**	**MacQueen**	**robin**	**MinMaxScaler**	0,339838	0,339707	1,797136	**3**	2,23E-16	0,000206	0,044621	0
12	**HTRU_2**	**MacQueen**	**robin**	**StandardScaler**	**0,405581**	0,405406	2,138194	4	7,25E-16	0,000224	0,052696	0
13	**HTRU_2**	**MacQueen**	**kmpp**	**False**	0,026497	0,026464	3,883137	15,44	1,39E-17	8,43E-05	0,119234	0,498888
14	**HTRU_2**	**MacQueen**	**kmpp**	**MinMaxScaler**	0,339838	0,339689	2,175284	6,11	2,23E-16	0,000213	0,186524	0,723278
15	**HTRU_2**	**MacQueen**	**kmpp**	**StandardScaler**	0,405379	0,405397	2,586312	8,24	0,00023	0,000227	0,373544	1,995551
16	**HTRU_2**	**MacQueen**	**forgy**	**False**	0,026497	0,026475	3,747652	16,04	1,39E-17	9,93E-05	0,097452	0,281411
17	**HTRU_2**	**MacQueen**	**forgy**	**MinMaxScaler**	0,339838	0,33968	**1,672864**	5,39	2,23E-16	0,000216	0,178766	0,737111
18	**HTRU_2**	**MacQueen**	**forgy**	**StandardScaler**	0,405208	0,405425	1,933362	6,62	0,000181	0,000219	0,351855	1,790999
	info				**min**				**max**			
	dataset	**algorithm**	**init method**	**normalize**	**NMI**	**sklearn NMI**	**time**	**iterations**	**NMI**	**sklearn NMI**	**time**	**iterations**
1	**HTRU_2**	**Lloyd**	**robin**	**False**	0,026497	0,026438	4,758122	20	0,026497	0,026763	5,214553	20
2	**HTRU_2**	**Lloyd**	**robin**	**MinMaxScaler**	0,339838	0,339386	1,899932	4	0,339838	0,339838	2,120898	4
3	**HTRU_2**	**Lloyd**	**robin**	**StandardScaler**	0,405581	0,405121	2,732157	8	0,405581	0,405581	3,054487	8
4	**HTRU_2**	**Lloyd**	**kmpp**	**False**	0,026497	0,026438	5,832662	27	0,026497	0,026763	6,47075	28
5	**HTRU_2**	**Lloyd**	**kmpp**	**MinMaxScaler**	0,339838	0,339386	2,359205	8	0,339838	0,339838	3,628357	15
6	**HTRU_2**	**Lloyd**	**kmpp**	**StandardScaler**	0,405121	0,405121	2,484668	9	0,405581	0,405581	4,57673	19
7	**HTRU_2**	**Lloyd**	**forgy**	**False**	0,026497	0,026438	5,48325	27	0,026497	0,026763	6,512727	31
8	**HTRU_2**	**Lloyd**	**forgy**	**MinMaxScaler**	0,339838	0,339386	1,8872	7	0,339838	0,339838	3,927813	18
9	**HTRU_2**	**Lloyd**	**forgy**	**StandardScaler**	0,405121	0,405121	1,895506	7	0,405581	0,405581	5,450547	27
10	**HTRU_2**	**MacQueen**	**robin**	**False**	0,026497	0,026438	3,201954	11	0,026497	0,026763	3,444361	11
11	**HTRU_2**	**MacQueen**	**robin**	**MinMaxScaler**	0,339838	0,339386	1,710599	3	0,339838	0,339838	1,914661	3
12	**HTRU_2**	**MacQueen**	**robin**	**StandardScaler**	0,405581	0,405121	2,032012	4	0,405581	0,405581	2,296646	4
13	**HTRU_2**	**MacQueen**	**kmpp**	**False**	0,026497	0,026438	3,665674	15	0,026497	0,026763	4,188885	16
14	**HTRU_2**	**MacQueen**	**kmpp**	**MinMaxScaler**	0,339838	0,339386	1,824852	5	0,339838	0,339838	2,515428	8
15	**HTRU_2**	**MacQueen**	**kmpp**	**StandardScaler**	0,405121	0,405121	2,087945	6	0,405581	0,405581	3,148082	10

16	**HTRU_2**	**MacQueen**	**forgy**	**False**	0,026497	0,026438	3,480426	15	0,026497	0,026763	4,089144	17
17	**HTRU_2**	**MacQueen**	**forgy**	**MinMaxScaler**	0,339838	0,339386	1,288954	4	0,339838	0,339838	2,43711	9
18	**HTRU_2**	**MacQueen**	**forgy**	**StandardScaler**	0,405121	0,405121	1,521081	5	0,405581	0,405581	3,182558	13

Bar Plots

This fragment presents bar plots for mean NMI, time and number of iterations before convergence for each combination of algorithm, initialization method and data normalization scheme. The results are plotted from best to worst.





The best result for skin dataset was achieved for Lloyd/MacQueen with kmpp initalization and no data normalization. This approach performed best with NMI equal to 0.023397. On the HTRU_2 dataset Lloyd/MacQueen with robin and StandardScaler achieved best results with NMI equal to 0.405581.

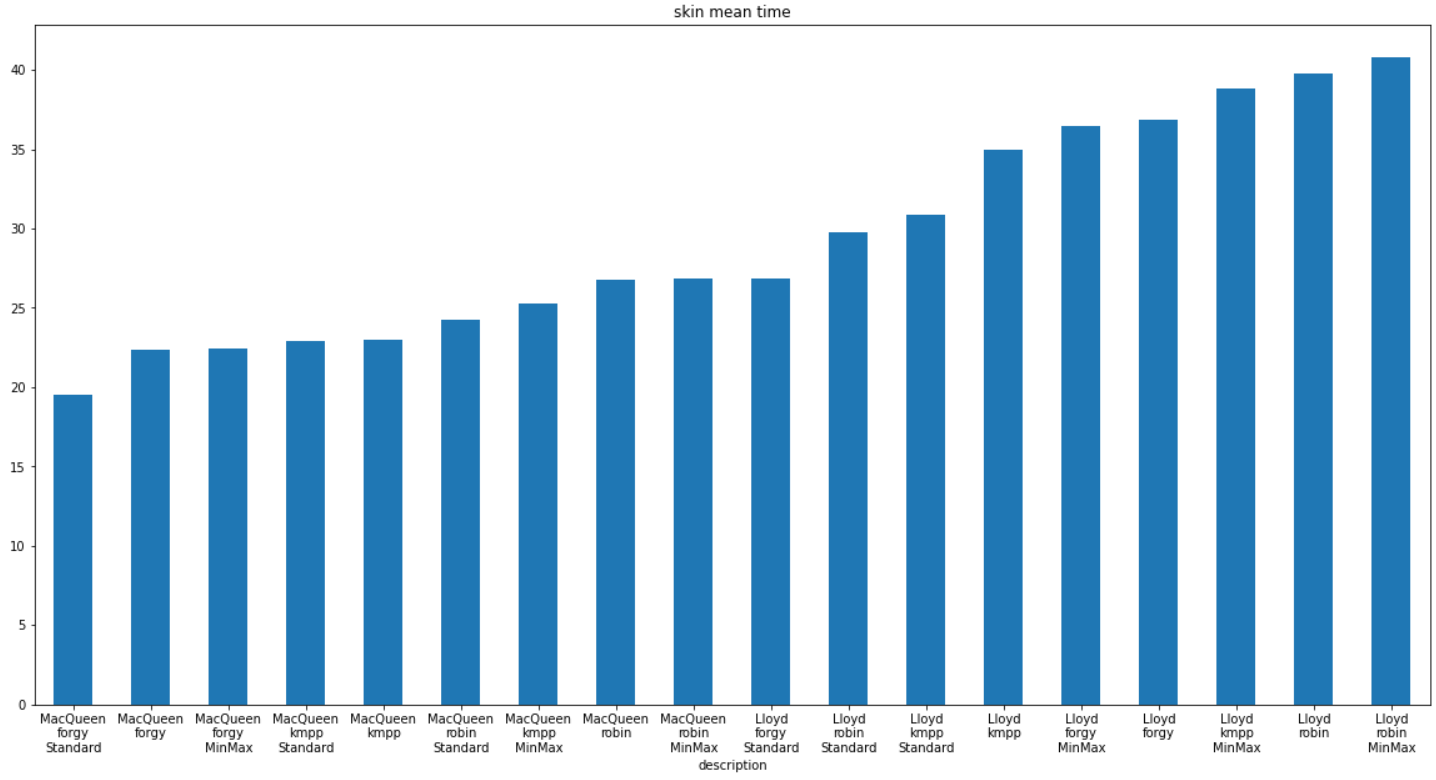
From examining the mean NMI score of different approaches we can arrive to the following conclusions: Most importantly, it is not the algorithm or the initialization method which has the greatest influence on the NMI. It is the preprocessing of data. This is mostly visible on the HTRU_2 dataset when applying data normalization resulted in a tenfold increase in NMI when compared to using raw data. This effect can be attributed to the fact clusters created by KMeans are N-dimensional hyperspheres in the feature space that are easily effected by the difference in value ranges of individual features. On the HTRU_2 dataset best results were achieved using StandardScaler. However, on the skin dataset the algorithm achieves best results on the raw data or data transformed with MinMaxScaler. Applying the Standard Scaler lowered the achieved NMI.

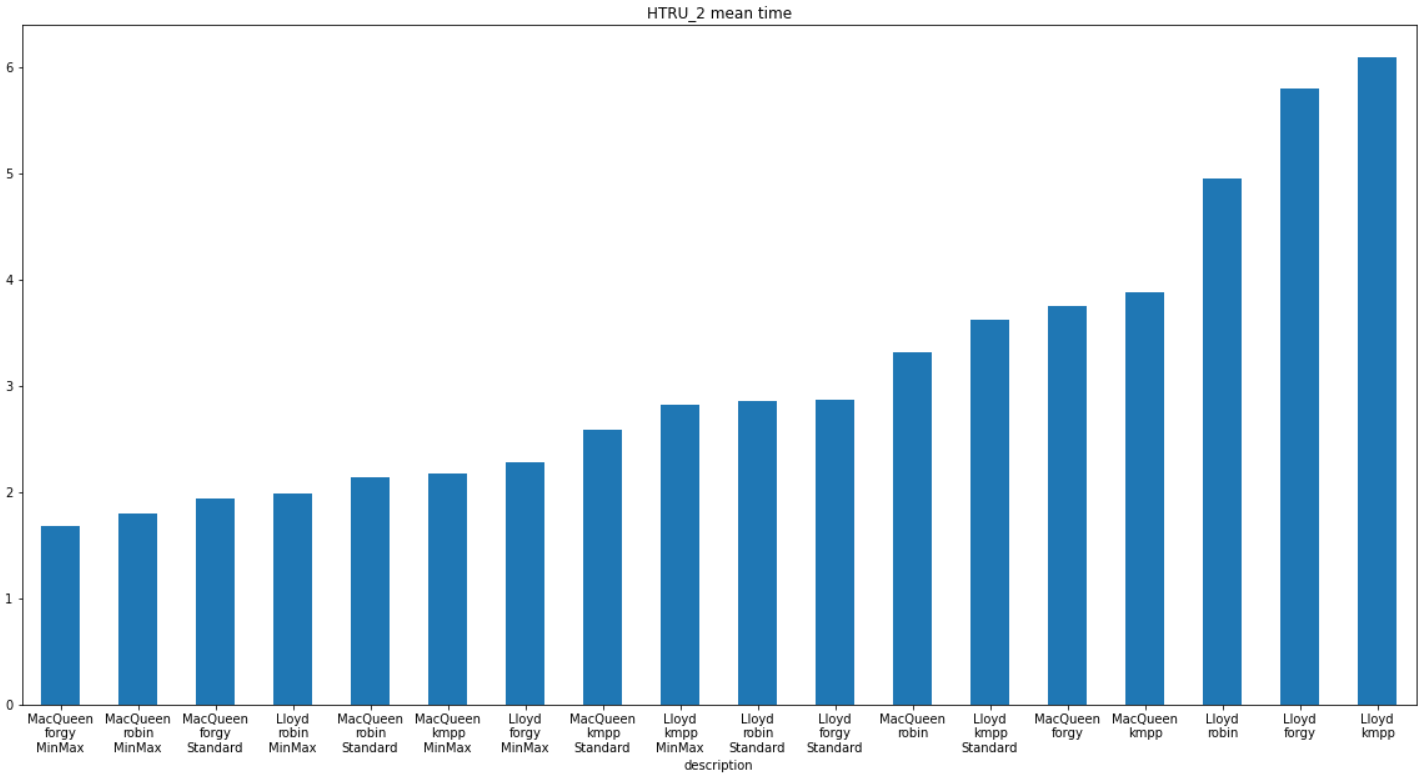
When it comes to the algorithm selection MacQueen provides no visible advantage over Lloyd. However, we can conclude that on the large skin dataset kmpp initialization achieved better results than kmpp and on the small HTRU_2 dataset robin performed better. Both approaches tend to perform better than forgy although the difference is minimal when compared to the difference related to data preprocessing. It is no surprise that the more complicated initialization schemes perform better than a simple approach like forgy.

By comparing the achieved NMI results with the sklearn NMI results we can observe that in most cases sklearn provides better results, but the difference is minor.

The variance of the results is neglectable, often equal to 0. In our interpretation it is due to the fact that different approaches fall into different local minimums that are the same for each of the 100 runs of a given approach.

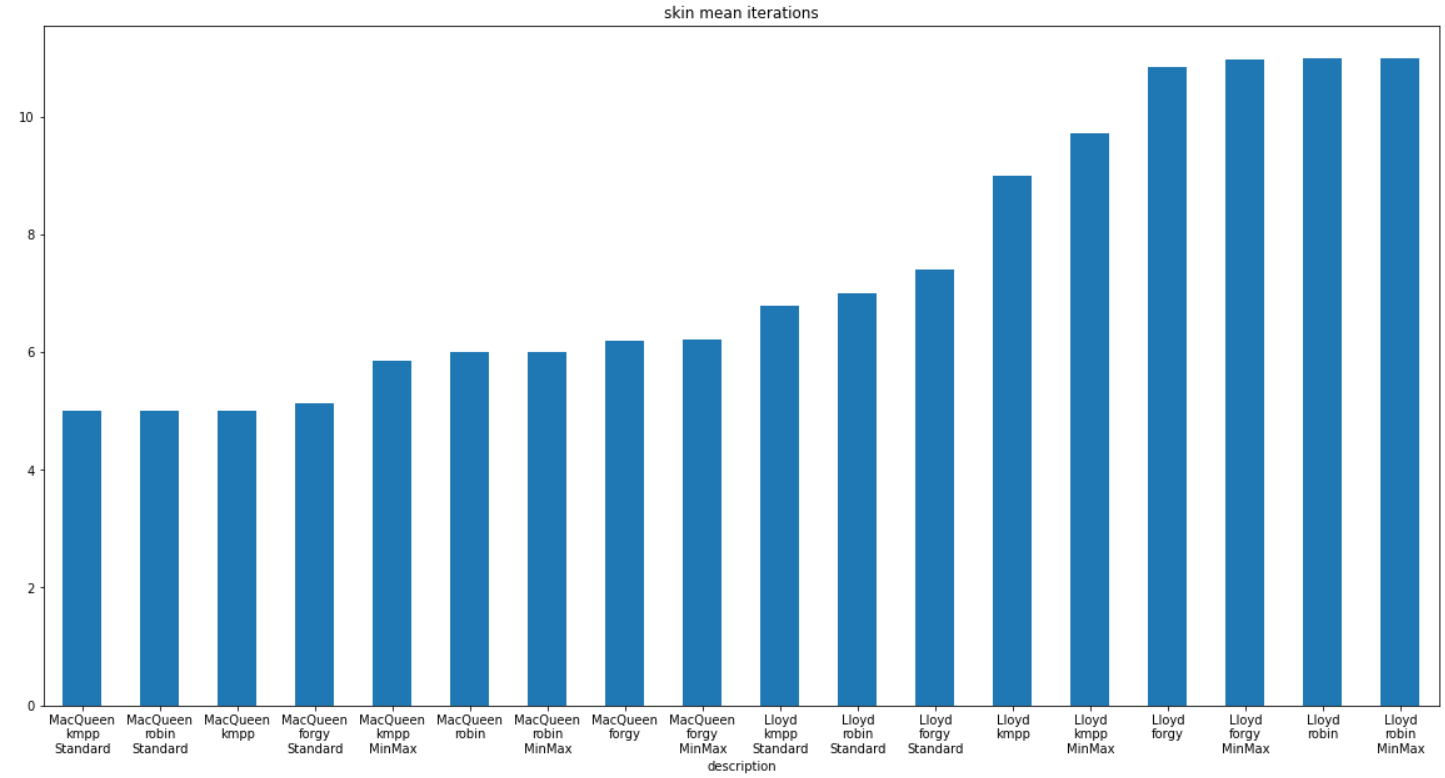
The extremely low NMI values for the skin dataset show that kmeans is a really simple algorithm and is not sutiable for complicated problems.

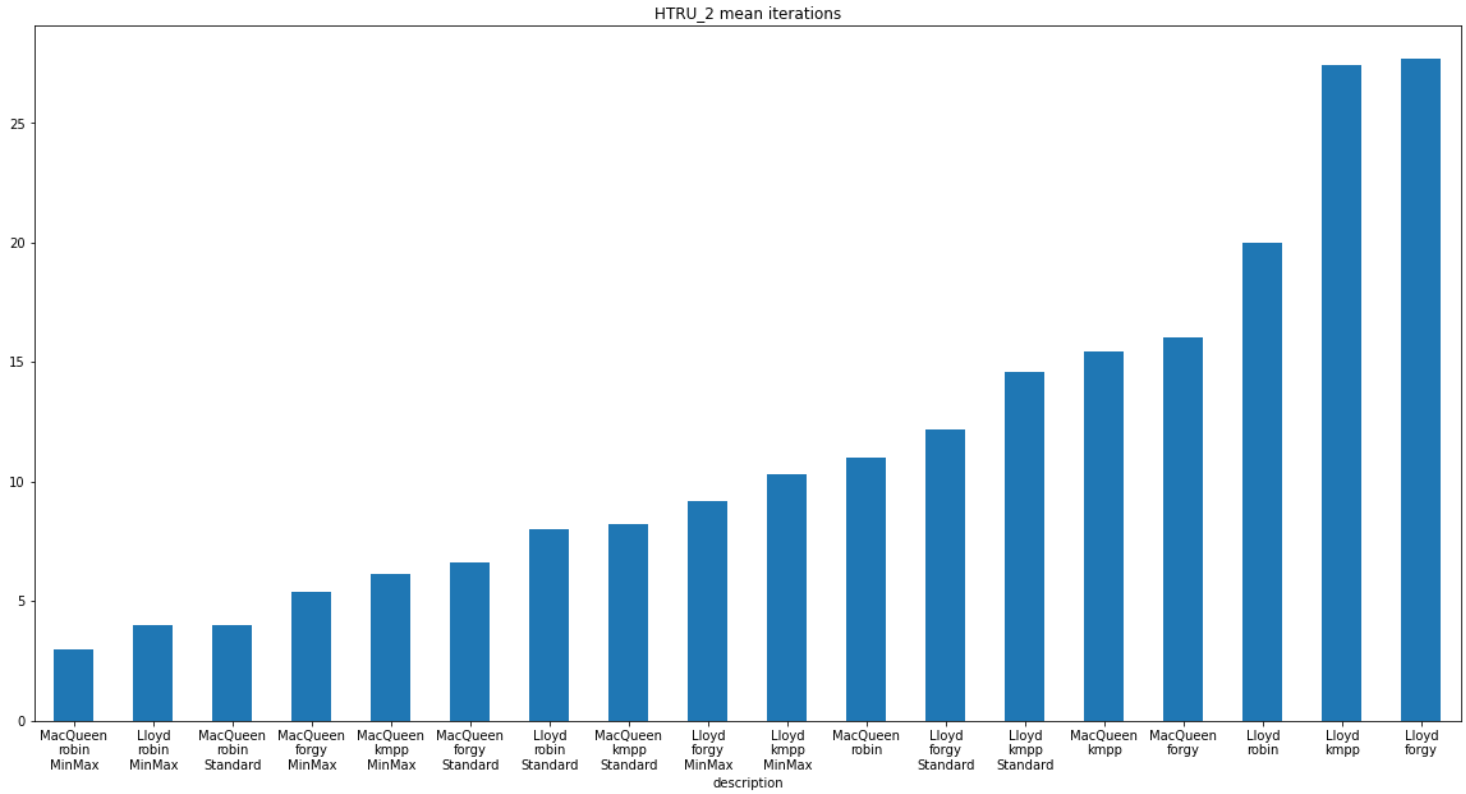




The fastest convergence time for the skin dataset was achieved using MacQueen with forgy and StandardScaler and was equal to 19.54 seconds. For th HTRU dataset MacQueen with forgy and MinMaxScaler was the fastest and converged in 1.672864 seconds.

After examining the results we can arrive to the following conclusions: Firstly, MacQueen is faster than Lloyd. This is most visible for larger datasets like the skin dataset where Lloyd always takes more time than MacQueen. Secondly, forgy initialization tends to result in slightly faster convergence time. This is due to the fact that using this method calculating initial centroids is extremely fast.





For the skin dataset MacQueen with robin + StandardScaler / kmpp + Standard / kmpp + NoScaler converged with minimal number of iterations equal to 5. For the HTRU_2 dataset MacQueen with robin and MinMaxScaler converged with the minimal number of iterations equal to 3.

From the presented results we can arrive to the following conclusions: Firstly, in general MacQueen converged in fewer iterations than Lloyd. This is due to the fact that MacQueen updates centroids every time a point changes its assignment. We can also see that forgy tends to require more iterations to converge than more elaborate initialization schemes.

For the HTRU_2 dataset which benefits heavily from standardization we can osberve that normalized data allows for convergence in fewer iterations than raw data.

Summary

By completing this assignment we have arrived to important conclusions related to kmeans and working with data. First and foremost it is crucial to always pre-process your data before applying the algorithm. The details of the chosen algorithm and initialization impact had minor impact on NMI scores and the clusterization quality when compared to the effect of data pre-processing.

Secondly, kmeans is a really simple algorithm in its nature and is not suitable for complicated problems which was proven by poor results achieved on the skin dataset.

When it comes to kmeans the following observation can be made: Firstly, MacQueen is generally better and faster than Lloyd. Even if it achieves similar results in terms of NMI its convergence is much faster. Secondly, we can achieve better results when using any non-trivial initialization scheme rather than forgy. However the choice between kmpp and robin does not influence the results greatly. Rather than investing the majority of worktime on tweaking the algorithm and initialization method It is reasonable to go with MacQueen and kmpp and focus on data preprocessing.

Bibliography

[1] M. Emre Celebi, Hassan A. Kingravi, Patricio A. Vela,

A comparative study of efficient initialization methods for the k-means clustering algorithm,

https://doi.org/10.1016/j.eswa.2012.07.021.

(http://www.sciencedirect.com/science/article/pii/S0957417412008767)

[2] David Arthur, Sergei Vassilvitskii,

k-means++: The Advantages of Careful Seeding

(https://theory.stanford.edu/~sergei/papers/kMeansPP-soda.pdf)

[3] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, Mohammed J. Zaki,

Robust partitional clustering by outlier and density insensitive seeding,

https://doi.org/10.1016/j.patrec.2009.04.013.

(http://www.sciencedirect.com/science/article/pii/S0167865509000956)