

---

# DATA MINING PROJECT

---

-CLARA-



BUZATU RARES TUDOR  
ARTIFICIAL INTELLIGENCE  
UPB

## Table of Contents

Abstract.....	2
Introduction.....	2
Importance and practical applications .....	2
Algorithm.....	3
K-median .....	3
Partitioning Around Medoids (PAM).....	3
CLARA .....	4
Datasets.....	4
Small size dataset .....	4
Medium size dataset.....	5
Large size dataset.....	6
Implementation .....	6
Results .....	7
Small dataset .....	7
Cluster plot .....	7
Execution time .....	7
Confusion matrix.....	7
Interpretation .....	8
Medium dataset .....	8
Cluster plot .....	8
Execution time .....	8
Confusion matrix.....	8
Interpretation .....	9
Large dataset .....	9
Cluster plot .....	9
Execution time .....	9
Confusion matrix.....	10
Interpretation .....	10
Conclusions.....	10
Bibliography .....	10

## Abstract

This project consists of an analysis of the algorithm Clustering Large Applications (CLARA). In this paper both a speed and an accuracy comparison will be made, but the main comparison is the one between this algorithm and its previous version which was not working fast enough for big datasets. After describing the algorithm particularities, some tests will be made in order to see the results. There will be used three datasets of different sizes and from different fields in order to better analyze how good this method works.

## Introduction

As the name suggest, CLARA is an algorithm used for clustering and was invented by Kaufman and Rousseeuw in 1990 because of the need of a clustering algorithm for big datasets. [1] The principal particularity of this algorithm is represented by its speed when talking about large data. Clara is based on k-medoids algorithm, but improves k-medoids by clustering not the entire dataset, but just a part of it. [2]

## Importance and practical applications

The fields of use for this algorithm are extremely various. Basically, any field of work is good as long as the dataset contains a large number of entries or a large number of attributes. Some examples of fields are [3]:

- Multivariate surveys
- Recognizing communities: social media
- Image segmentation
- Anomaly detection
- Crime analysis
- Identifying fake news
- Pattern recognition
- Data compression
- Computer graphics
- Machine learning.
- Bioinformatics

Being a clustering algorithm, its purpose is just to group similar entries close to one another forming a cluster and not to classify elements from a dataset (see figure 1). The data from datasets is almost unlabeled, that's why it needs to be clustered and not classified, but the results from this paper were manually labeled in order to see how accurate the algorithm is.

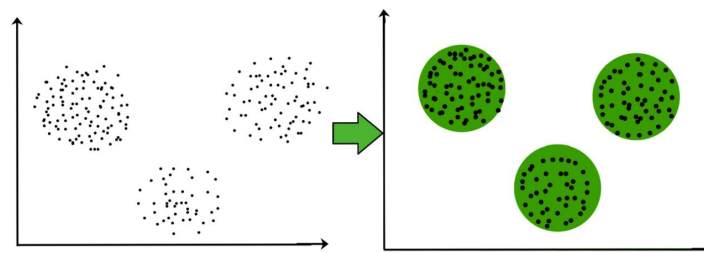


Figure 1: Before clustering vs After clustering <sup>1</sup>

<sup>1</sup> <https://www.geeksforgeeks.org/clustering-in-machine-learning>

## Algorithm

CLARA is an improved version of k-median, so, in order to describe CLARA, one should first understand how k-median works.

### K-median

K-median is an algorithm very similar to k-means. Its purpose is to form clusters from the given data. Compared to k-means, this algorithm takes the median of each cluster in order to determine its center, instead of taking its mean, the resulting cluster being named medoid, instead of centroid. This means, that always its center will be one of the elements from the dataset, opposed to k-means which its center is not always included in the dataset. From a mathematical point of view, taking the median, instead of the mean, reflects in minimizing the error over all clusters with respect to the 1-norm distance metric, as opposed to the squared 2-norm distance metric [1]. Another difference from k-means is the way the algorithm treats outliers: k-medoids are less influenced by outliers, than k-centroids:

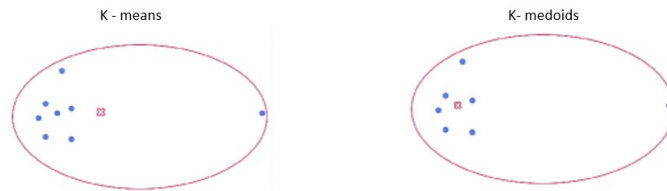


Figure 2: Outliers: k-means vs k-medoids <sup>1</sup>

To understand this is easier to work in an 1 dimensional space (left-right). When a new entry is added for k-medoids it does not matter how far is that value, it matters just its position with respect to the center, where when speaking about k-means it matters not just the position, but the value as well.

### Partitioning Around Medoids (PAM)

PAM is a way of implementing k-medoids clustering which uses a greedy search. The basic pseudocode for its implementation is the following [1]:

1. Select k random data points as initial medoids
2. Associate each point to the closest medoid
3. Compute distance cost medoids – points
4. Try other points as medoids
5. Compare costs
6. Keep combination of lowest cost
7. Repeat for all combinations of medoids

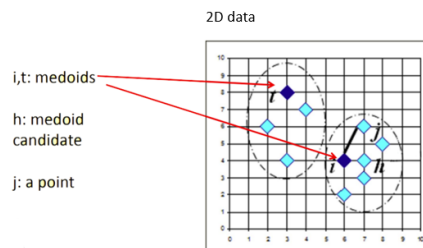


Figure 3: k-medoids <sup>2</sup>

<sup>1</sup>[https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-30164-8\\_426](https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-30164-8_426)

<sup>2</sup><https://www.youtube.com/watch?v=GApaAnGx3Fw>

The complexity of PAM is  $O(k(n-k)^2)$  / iteration. A threshold or a maximum number of iterations is given in order to stop the search.

## CLARA

CLARA is very similar to PAM. The only difference is that CLARA instead of working with the whole dataset, is working with multiple subsets from the big dataset, apply PAM on them and then select the best k-medoids. The pseudocode is the following [4]:

1. Create subsets randomly
2. Compute PAM for each subset
3. Compute dissimilarities of all obs:  
$$Cost(M, D) = \frac{\sum_{i=1}^n dissimilarity(O_i, rep(M, O_i))}{n},$$
4. where  $n$  = number of total entries;  $M$  = set of selected medoids;  $rep(M, O_i)$  returns a medoid in  $M$  which is closest to  $O_i$ ;  $dissimilarity(O_i, O_j)$  is the dissimilarity between objects  $O_i$  and  $O_j$
5. Retain subset with min dissimilarities

The complexity of CLARA is  $O(k(M-k)^2)$  / iteration.

## Datasets

For this project three datasets of different size were used in order to demonstrate the difference between CLARA and PAM: one small size dataset, one medium size dataset and one big size dataset. The datasets come from different fields and have a different number of entries and a different number of variables.

**!Obs:** Two of the three datasets have the entries already labeled. These labels will not be included in the clustering process because clustering work with unlabeled data. The only use for these columns is to help in creating the confusion matrices for a better accuracy understanding.

### Small size dataset

The small dataset is represented by a table which contains information about some wheat's seeds.

Characteristics:

- Name of dataset: "Small\_dataset\_seeds"
- Link to dataset: <https://archive.ics.uci.edu/ml/datasets/seeds> [5]
- Number of entries: 210
- Number of attributes: 7 total / 7 used
- Used attributes:
  - area: area of seed (numeric)
  - perimeter: perimeter of seed (numeric)
  - compactness: how much compact the seed is (numeric)
  - lengthofkernel: the length of the seeds kernel (numeric)
  - widthofkernel: the width of the seeds kernel (numeric)
  - asymmetrycoefficient: how much asymmetric the seed is (numeric)
  - lengthofkernelgroove: length of the kernels groove (numeric)
- Missing values: No
- Duplicate values: No
- Pre-labeled for testing: Yes

Sample of the dataset:

A	B	C	D	E	F	G	H	I
id	area	perimeter	compactness	lengthofkernel	widthofkernel	asymmetrycoefficient	lengthofkernelgroove	seedtype
1	15.26	14.84	0.871	5.763	3.312	2.221	5.22	1
2	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	1
3	14.29	14.09	0.905	5.291	3.337	2.699	4.825	1
4	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	1
5	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	1
6	14.38	14.21	0.8951	5.386	3.312	2.462	4.956	1
7	14.69	14.49	0.8799	5.563	3.259	3.586	5.219	1
8	14.11	14.1	0.8911	5.42	3.302	2.7	5	1
9	16.63	15.46	0.8747	6.053	3.465	2.04	5.877	1
10	16.44	15.25	0.888	5.884	3.505	1.969	5.533	1

Figure 4: Sample of small dataset

The only pre-processing required is normalization of data in order to give to all variables the same weight.

## Medium size dataset

The medium size dataset is represented by a table which contains information about the lifestyle of some people in order to identify their weight category.

- Name of dataset: "Medium\_dataset\_obesity"
- Link to dataset:  
<https://archive.ics.uci.edu/ml/datasets/Estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition+> [6]
- Number of entries: 2111
- Number of attributes: 16 total / 8 used
- Used attributes:
  - Height: Height in meters (numeric)
  - Weight: Weight in kilograms (numeric)
  - family\_history\_with\_overweight: Yes or No (boolean)
  - FAVC: Frequent consumption of high caloric food (boolean)
  - CAEC: Consumption of food between meals (text ordinal)
  - CH2O: Consumption of water daily (ordinal)
  - SCC: Calories consumption monitoring (boolean)
  - FAF: Physical activity frequency (ordinal)
- Missing values: No
- Duplicate values: No
- Pre-labeled for testing: Yes

Sample of the dataset:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	
1	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	MTRANS	NObesidad		
2	Female	21	1.62	64	yes	no		2	3	Sometimes	no		2	no	0	1	no	Public_Transportation	Normal_Weight
3	Female	21	1.52	56	yes	no		3	3	Sometimes	yes		3	yes	3	0	Sometimes	Public_Transportation	Normal_Weight
4	Male	23	1.8	77	yes	no		2	3	Sometimes	no		2	no	2	1	Frequently	Public_Transportation	Normal_Weight
5	Male	27	1.8	87	no	no		3	3	Sometimes	no		2	no	2	0	Frequently	Walking	Overweight_Level_I
6	Male	22	1.78	89.8	no	no		2	1	Sometimes	no		2	no	0	0	Sometimes	Public_Transportation	Overweight_Level_II
7	Male	29	1.62	53	no	yes		2	3	Sometimes	no		2	no	0	0	Sometimes	Automobile	Normal_Weight
8	Female	23	1.5	55	yes	yes		3	3	Sometimes	no		2	no	1	0	Sometimes	Motorbike	Normal_Weight
9	Male	22	1.64	53	no	no		2	3	Sometimes	no		2	no	3	0	Sometimes	Public_Transportation	Normal_Weight
10	Male	24	1.78	64	yes	yes		3	3	Sometimes	no		2	no	1	1	Frequently	Public_Transportation	Normal_Weight

Figure 5: Sample of medium dataset

The pre-processing of this dataset required to transform all text variables into numerical values. For all variables a high numerical value was assigned for things that are helping getting more weight and small values for things that are helping losing weight. The labels column was converted to three numeric classes for a better visualization of the clustering. The classes are transformed in this order to match the clustering order:

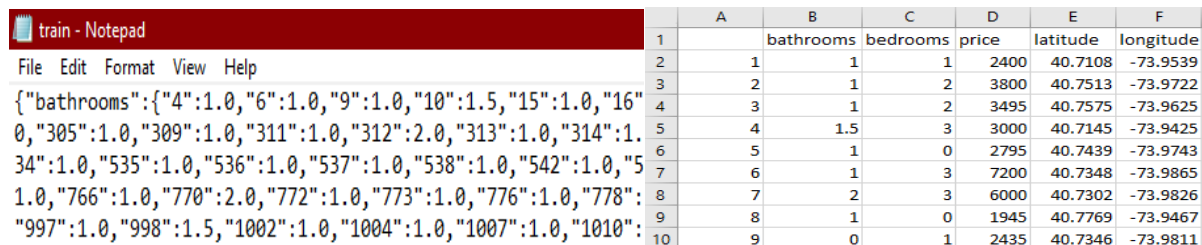
1. Insufficient weight
2. Overweight + Obesity
3. Normal

## Large size dataset

The large dataset is represented by a table which contains information about house renting

- Name of dataset: "Big\_dataset\_renting"
- Link to dataset: <https://www.kaggle.com/c/two-sigma-connect-rental-listing-inquiries> [7]
- Number of entries: 49352
- Number of attributes: 5 total / 3 used
- Used attributes:
  - bathrooms: number of bathrooms (numeric)
  - bedrooms: number of bedrooms (numeric)
  - price: price of house (numeric)
  - Missing values: No
- Duplicate values: No
- Pre-labeled for testing: No

Sample of the dataset:



	A	B	C	D	E	F
1		bathrooms	bedrooms	price	latitude	longitude
2	1	1	1	2400	40.7108	-73.9539
3	2	1	2	3800	40.7513	-73.9722
4	3	1	2	3495	40.7575	-73.9625
5	4	1.5	3	3000	40.7145	-73.9425
6	5	1	0	2795	40.7439	-73.9743
7	6	1	3	7200	40.7348	-73.9865
8	7	2	3	6000	40.7302	-73.9826
9	8	1	0	1945	40.7769	-73.9467
10	9	0	1	2435	40.7346	-73.9811

Figure 6: Data before vs after processing

The pre-processing of this required parsing the original "train.json" file which is represented in Fig. 6 (left). The data was parsed and split in columns in form of a Python dictionary and then stored in a .csv file (Fig. 6 right). After that, the data was normalized in order to equalize the weight of attributes.

## Implementation

For the algorithm's implementation and testing programming language, "R" language was used, along with "R Studio" IDE. This environment was chosen because R is a specialized language for data processing / analyzing and has already built function for clustering. Three scripts were created, one for each dataset. The scripts are very similar between each other. The main operations in a script are:

- Load the dataset in a data frame variable

- Normalize each column using the function “scale” from R
- Select the columns used for clustering using function “subset” from R
- Start a timer
- Execute CLARA / PAM using predefined functions from R
- Stop the timer
- Print the execution times
- Plot the clusters

## Results

Here are the results for all datasets:

### Small dataset

For the small dataset there were considered 3 clusters, because the labels of the data had already 3 classes.

### Cluster plot

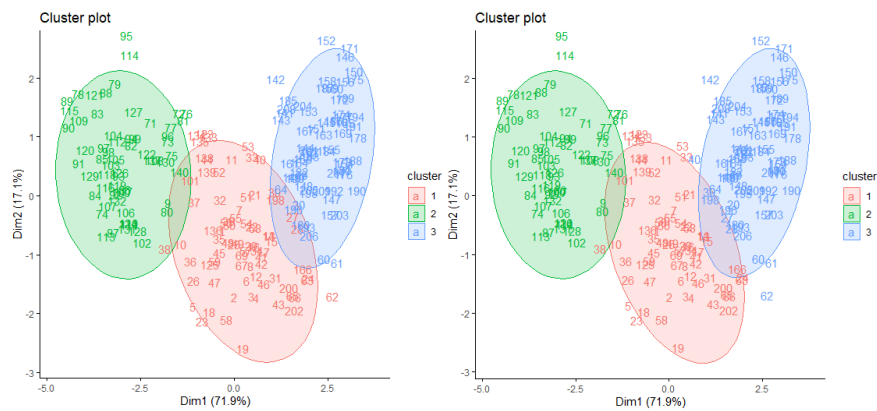


Figure 7: PAM's cluster (left) vs CLARA's cluster (right)

### Execution time

Algorithm	Time of execution
PAM	0.00700
CLARA	0.00899

### Confusion matrix

PAM	1	2	3	CLARA	1	2	3
1	64	1	5	1	59	1	10
2	9	61	0	2	9	61	0
3	4	0	66	3	3	0	67



## Interpretation

The cluster's plot looks identical for both algorithms. The execution time is slightly smaller on PAM, but at that scale, the difference is insignificant. Looking at the confusion matrix, CLARA gives slightly worse results, but again, the difference between the two is really small.

## Medium dataset

For the medium dataset there were considered 3 clusters for a better visualization of the clusters. The labels column was already split in 3 categories.

## Cluster plot

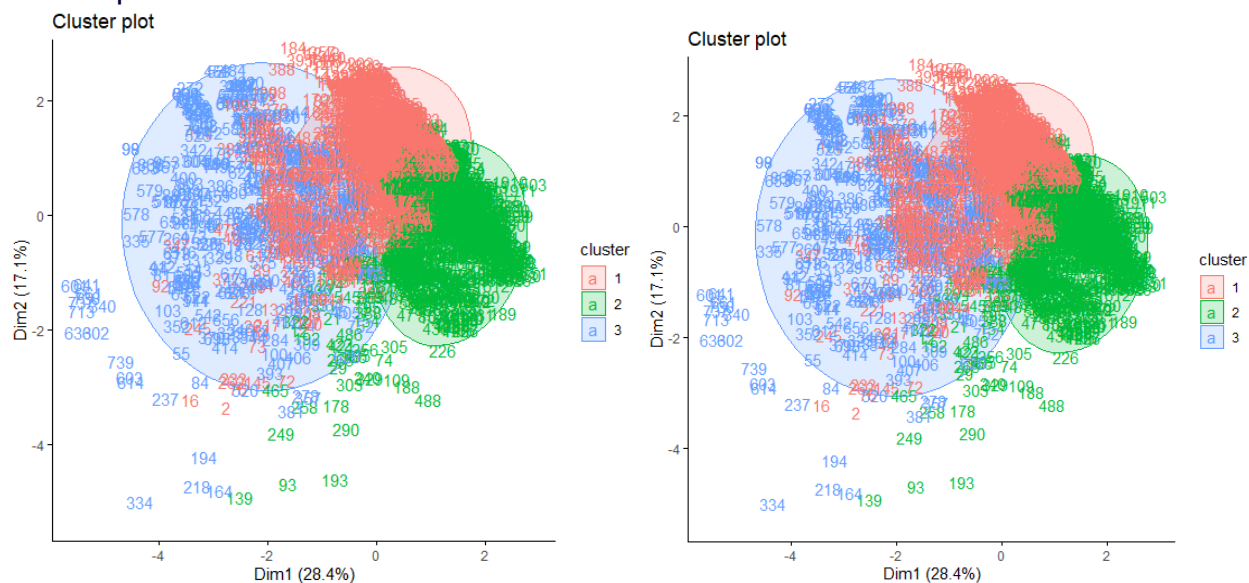


Figure 8: PAM's cluster (left) vs CLARA's cluster (right)

## Execution time

Algorithm	Time of execution
PAM	0.6810021
CLARA	0.0149970

## Confusion matrix

PAM	1	2	3	CLARA	1	2	3
1	72	54	146	1	97	29	146
2	566	557	105	2	589	532	107
3	123	35	129	3	129	26	132

## Interpretation

Like the previous dataset, the clusters generated by the two algorithms look the same. The time of execution is in CLARA's favor this time and the difference between them is larger than the previous case. Looking at the confusion matrix, the results are pretty similar, thing observed from looking at the clusters.

## Large dataset

In order to keep consistency between testing, three clusters will be considered as well for this dataset. This dataset is not prior labeled, so the confusion matrix cannot be computed, but a version of confusion matrix can be produced between the labels given by PAM and by CLARA. The clusters and the confusion matrix were extracted from a 10000 entries subset. The whole dataset (49352 entries) was not used at a time, because PAM would take too long to compute.

## Cluster plot

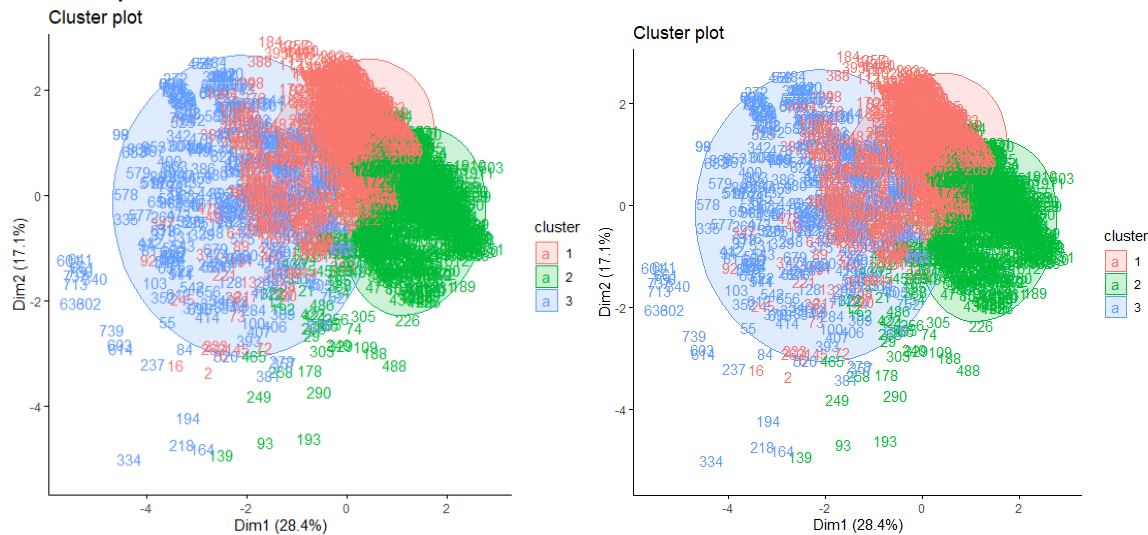


Figure 9: PAM's cluster (left) vs CLARA's cluster (right)

## Execution time

To better understand the difference between PAM and CLARA, several subsets of different lengths from this dataset will be considered. The plot was realized using MATLAB:

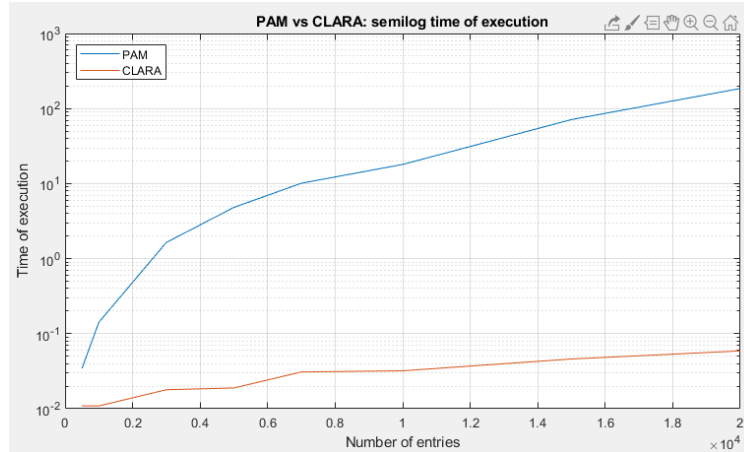


Figure 10: PAM vs. CLARA : semilog time of execution

## Confusion matrix

PAM vs CLARA	1	2	3
1	4999	0	0
2	11	3116	6
3	1	0	1867

## Interpretation

The clusters look identical for the 2 algorithms. The main difference between them is the time of execution. It can be observed that PAM's execution's time increases extremely fast. In fact, its increase is exponential, but it is hard to observe that on a logarithmic scale. The logarithmic scale was chosen to observed just as good both lines. For the CLARA's time of execution, the results are pretty constant and even for a subset of 20000 entries, the computation time remains under 0.1 seconds.

## Conclusions

After analyzing the algorithms both from an accuracy and a time of execution points, the results were pretty conclusive that CLARA is definitely worth using for big datasets, because its accuracy is just as good as PAM's is, but its time of execution remains constant under 0.1 seconds even for big datasets. However, for smaller datasets (under 500 entries), PAM can be used without any problems.

## Bibliography

[1] Kaufman L, Rousseeuw PJ (1990) Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley&Sons, DOI 10.1002/9780470316801

[2] Schubert, Erich; Rousseeuw, Peter J. (2019), Amato, Giuseppe; Gennaro, Claudio; Oria, Vincent; Radovanović, Miloš (eds.), "Faster k-Medoids Clustering: Improving the PAM, CLARA, and CLARANS Algorithms", *Similarity Search and Applications*, Springer International Publishing, arXiv:1810.05691

[3] Claire Whittaker, "7 Innovative Uses of Clustering Algorithms in the Real World", <https://dataflog.com/read/7-innovative-uses-of-clustering-algorithms/6224>

- [4] Alboukadel Kassambara, *“Practical Guide to Cluster Analysis in R”*, Multivariate Analysis I, Edition 1.
- [5] Seeds kernel dataset, <https://archive.ics.uci.edu/ml/datasets/seeds>
- [6] Obesity dataset,  
<https://archive.ics.uci.edu/ml/datasets/Estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition+>
- [7] Rental dataset, <https://www.kaggle.com/c/two-sigma-connect-rental-listing-inquiries>