# Test Data Generation from Sample Population using Data Mining methods*

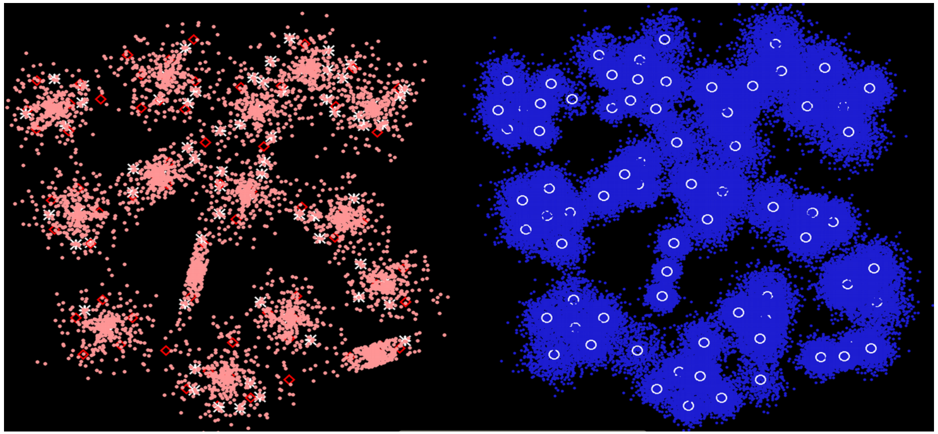
Supervisor:
**dr. Kiss, Attila**

Written by:
**Fazekas, Bálint**

*This work was supported by EFOP-3.6.3- VEKOP-16- 2017-00002 project.

# Motivation

- Legal use of a truthful dataset
- Generation of large datasets based on a small sample
- Simulate databases
- Recreate, and further expand a dataset



Sample dataset | Regenerated dataset

# Defining the problem

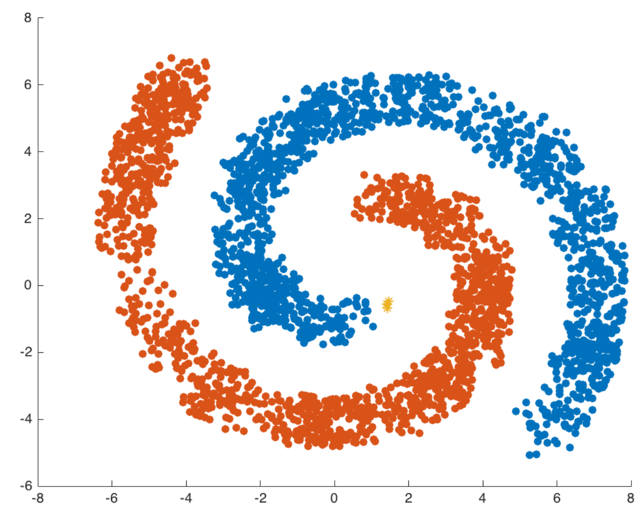
- Data regeneration:

  *"Upon observing a given sample dataset, we would like to identify smaller sub-clusters, and based on the statistical properties (such as the mean and the distribution) of these clusters we want to regenerate a similar dataset with possibly different number of data points."*

# Clustering methods [1]

- Hierarchical methods:
  - Agglomerative
  - Divisive

  <u>Similarity measures</u>:
  - Single-link
  - Complete-link
  - Average-link

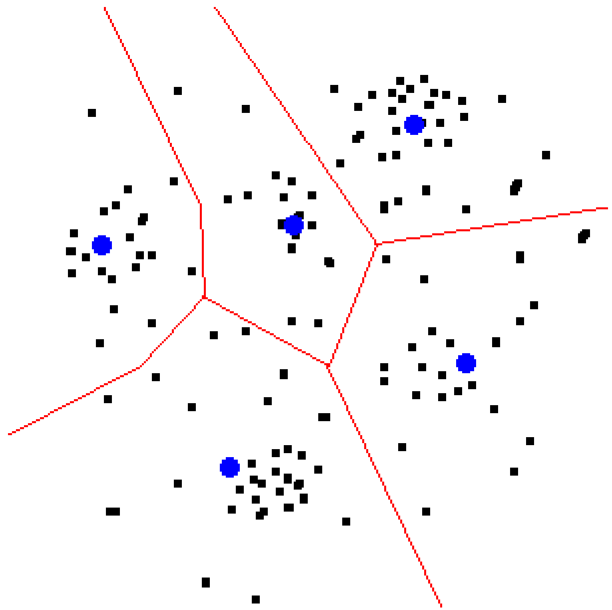




[1] Rokach, Lior, and Oded Maimon. "CLUSTERING METHODS."

# Clustering methods [1]

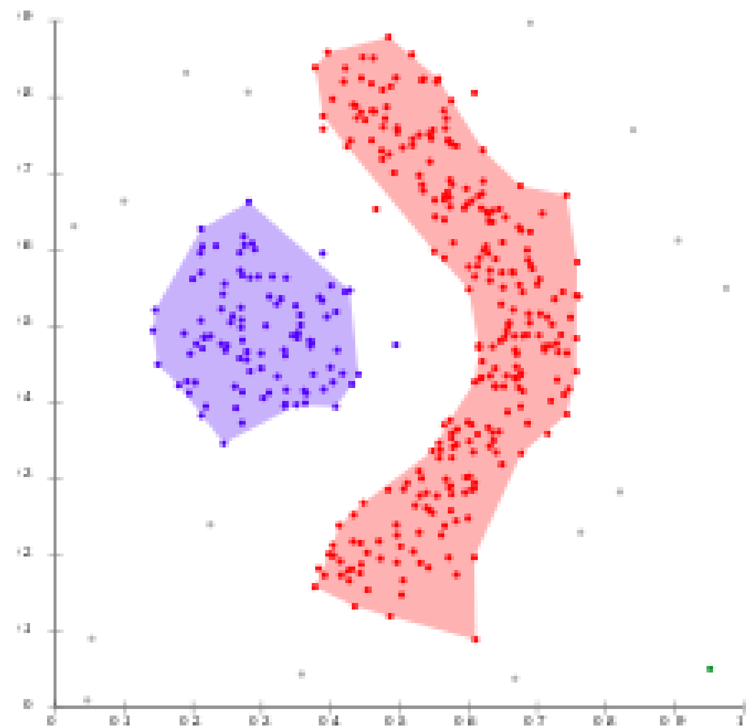
- Partitioning methods:
  - Divide space of the dataset into smaller ranges
  - Attempts to minimize the error of a centroid of a cluster



[1] Rokach, Lior, and Oded Maimon. "CLUSTERING METHODS."

# Clustering methods [1]

- Density based methods:
  - Considers the distances between the individual data points.
  - Creates a chain of data points that will become a cluster



[1] Rokach, Lior, and Oded Maimon. "CLUSTERING METHODS."

# Clustering methods

Which approach is better?

None, all have strengths and weaknesses.

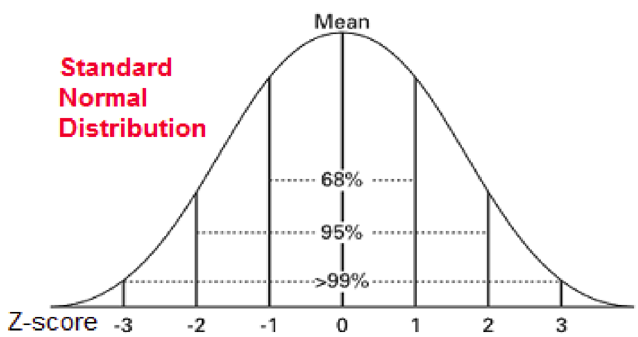
To solve the “data regeneration” problem, both approaches are needed.

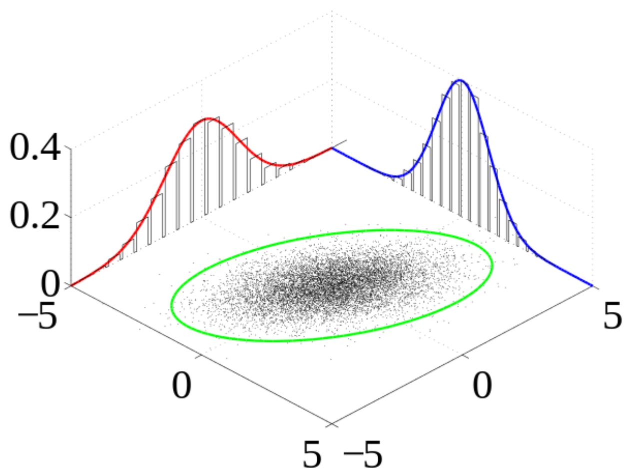
# Definitions

- Point: $p \in \mathbb{R}^n$, where $n \in \mathbb{N}$, and $n > 0$. A point only has position.
- Vector: similar to a point, but also has magnitude and direction.

  We are *vectorizing* the dataset so we can:
- Calculate distance
- Centroid
- Distribution

# Definitions



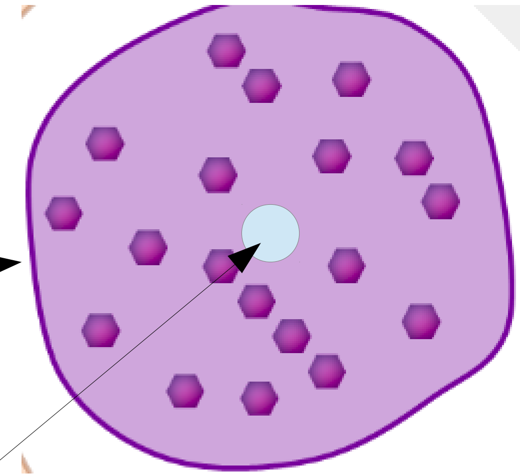




- Normal Distribution: the data points of a given dataset deviate around a mean value, and are usually most populated around the mean.
- Covariate (bivariate) distribution: it is possible for a dataset with data points of higher dimensions, to have different distributions in each dimensions.

# Definitions

- Normal distribution:

$$s_x = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$



- Centroid:

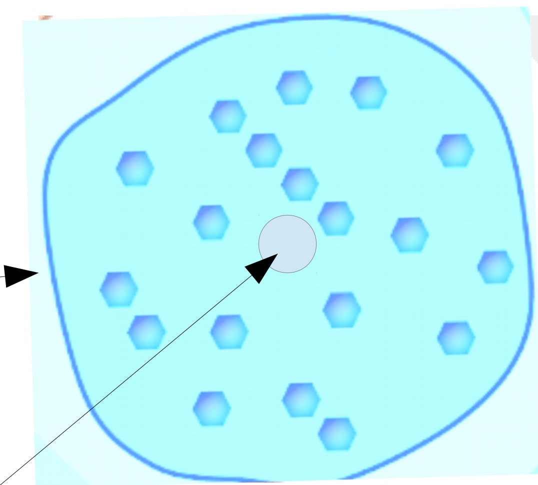
  The “mean” of a set of points (with any number of dimensions)

$$\bar{v} = \frac{Vec(\sum_{i=1}^{n} a_i, \sum_{i=1}^{n} b_i, ..., \sum_{i=1}^{n} z_i)}{n}$$

# Definitions

- Normal distribution:

$$s_x = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$



- Centroid:

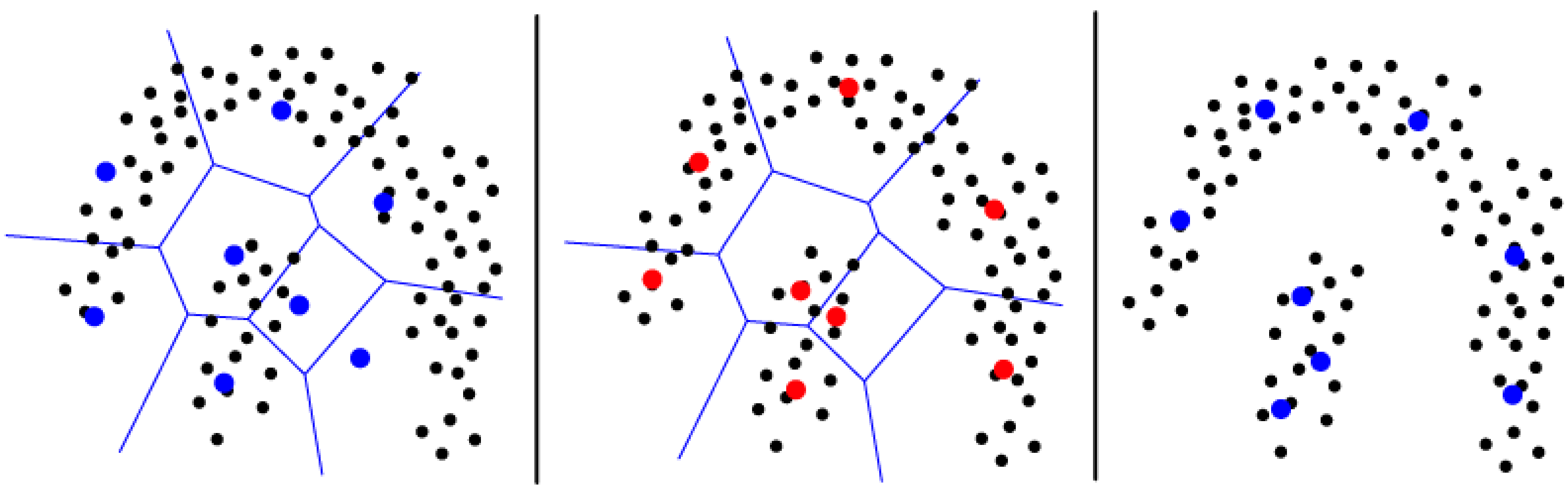
  The “mean” of a set of points (with any number of dimensions)

$$\bar{v} = \frac{Vec(\sum_{i=1}^{n} a_i, \sum_{i=1}^{n} b_i, \ldots, \sum_{i=1}^{n} z_i)}{n}$$

# Clustering Algorithms

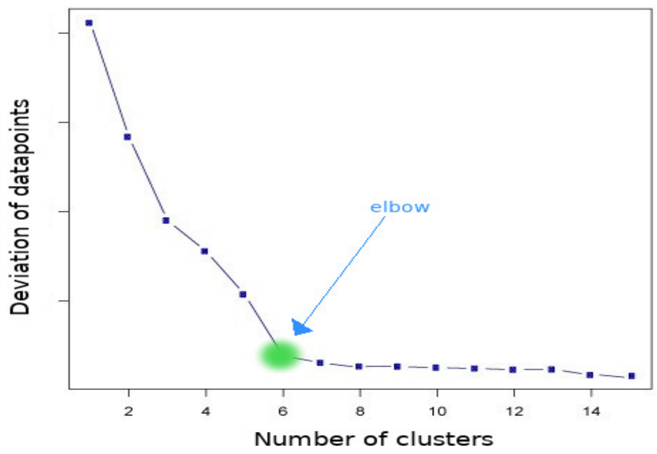
*K-means:* only needs the “k” number of clusters to create, given a dataset.

- 1st phase: placing “k” number of points, the markers of a cluster
- 2nd phase: fine-adjusting the position of the markers



# Elbow method

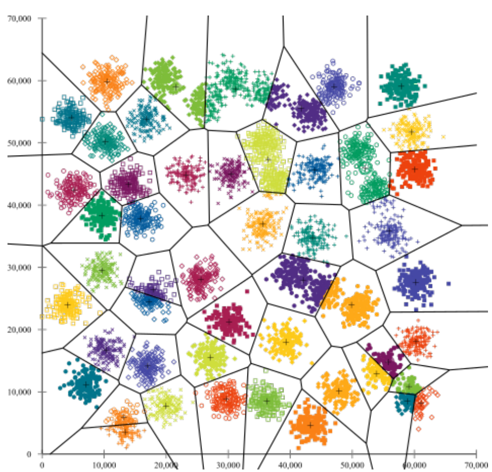
- Attempts to find the correct number of clusters for the k-means algorithm
- Calculates the average difference of squares between the data points and the markers
- Has to run the algorithm many times with increasing number of “k”

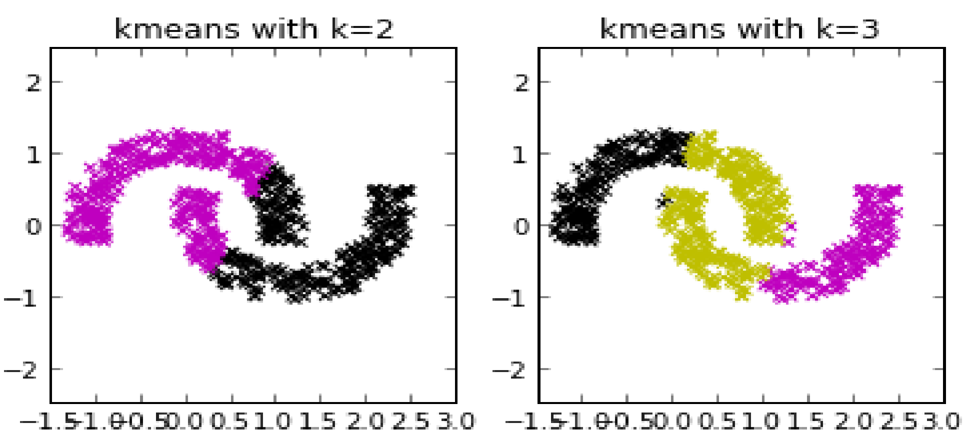


# K-means algorithm

- Problems with the k-means algorithm:
  - Can not identify clusters which have a concave shape!
  - What should be the “k” parameter?
  - Might not find the correct centroids

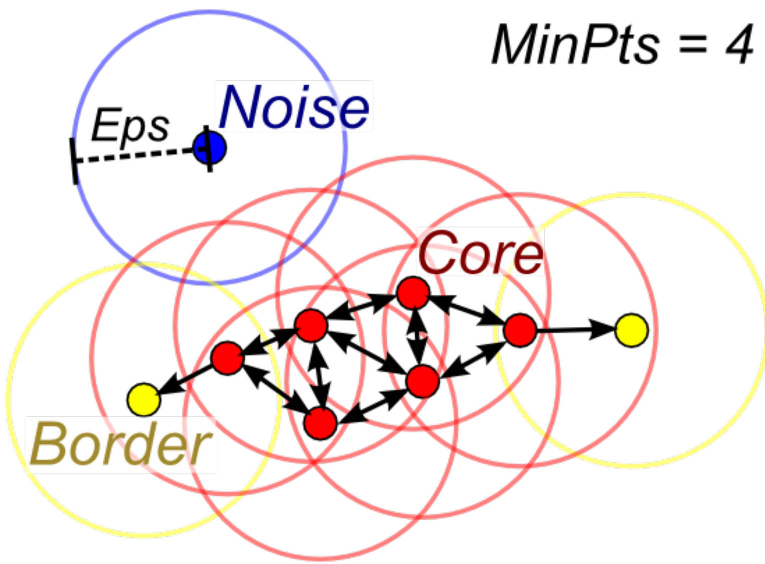
⇒ Possibly won’t give an intuitive result (set of sub-clusters)





# DBSCAN algorithm

- (Agglomerative) Density based algorithm.
- Needs a <u>minimum distance</u> and a <u>minimum number of points</u> as parameters.
- Only one phase; iterates through all the points, while creating clusters.



# Problems with DBSCAN

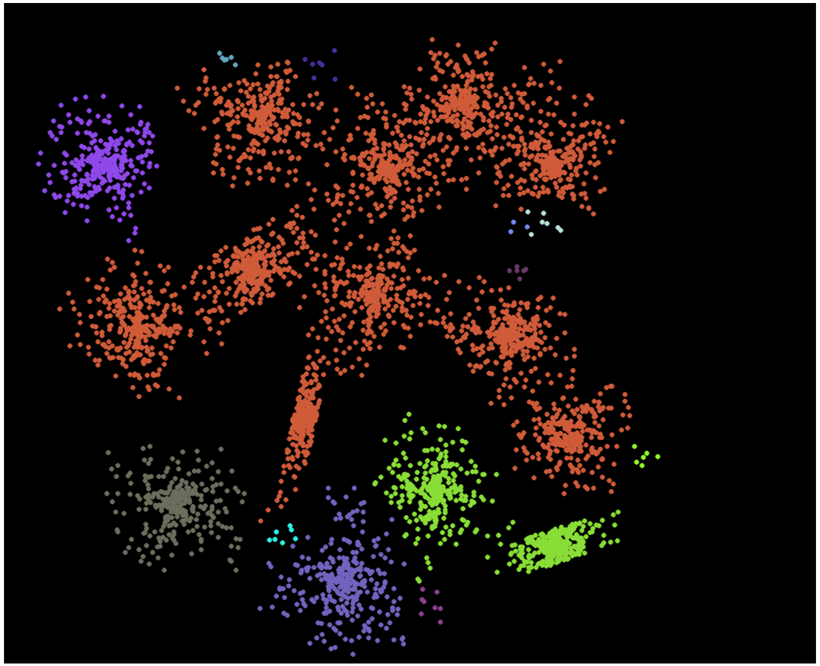
- What should be the minimum distance between points?
  - How should it be determined? ×
- What should be the minimum number of points in a cluster? (What do we consider as a cluster, and what becomes noise?)

Heuristics are needed in order to attempt the assumption of these parameters.

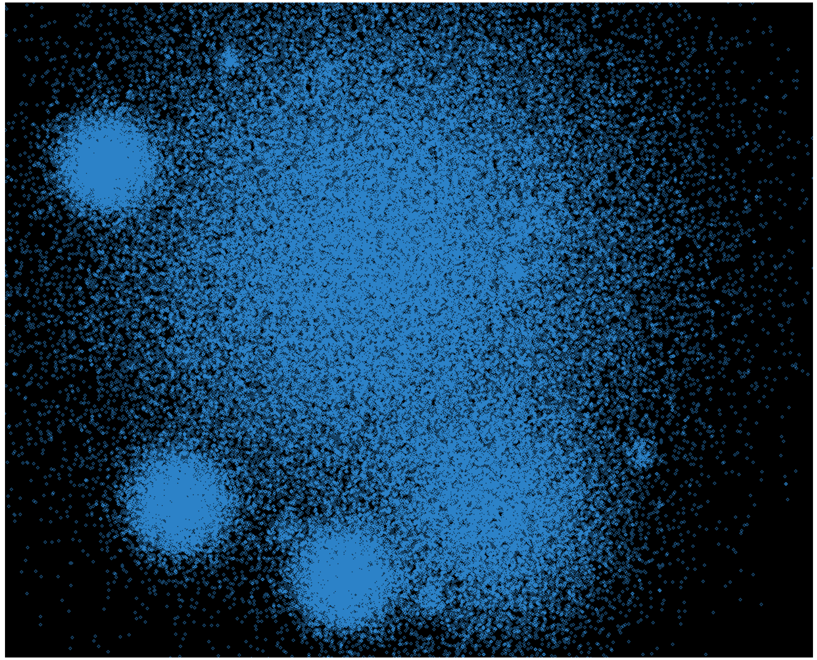
For now, we assume that both are correctly set.

# Problems with DBSCAN

- How do we analyze the result clusters?



**Original dataset clustered by DBSCAN**



**Regenerated dataset only using DBSCAN (50x data points)**

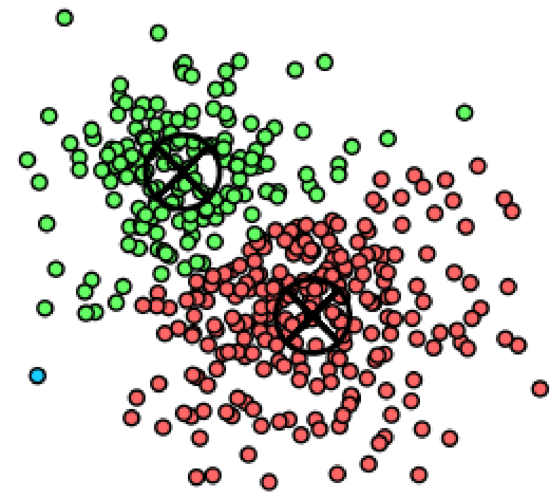
# Hybrid algorithm

- The DBSCAN is able to find the clusters correctly, in an "intuitive" fashion.
  - But it is not able to find a "good" *mean* to regenerate the data.
- The k-means is able to partition any dataset into "k" regions.
  - But neighboring clusters might interfere to find a good mean

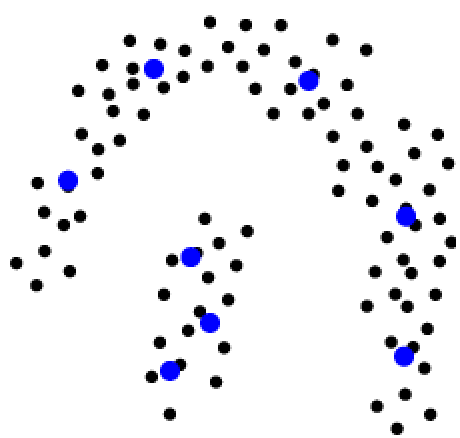
- *Proposal*: cluster the original dataset with the DBSCAN, and apply the k-means algorithm to the resulting sub-clusters!

# Hybrid algorithm

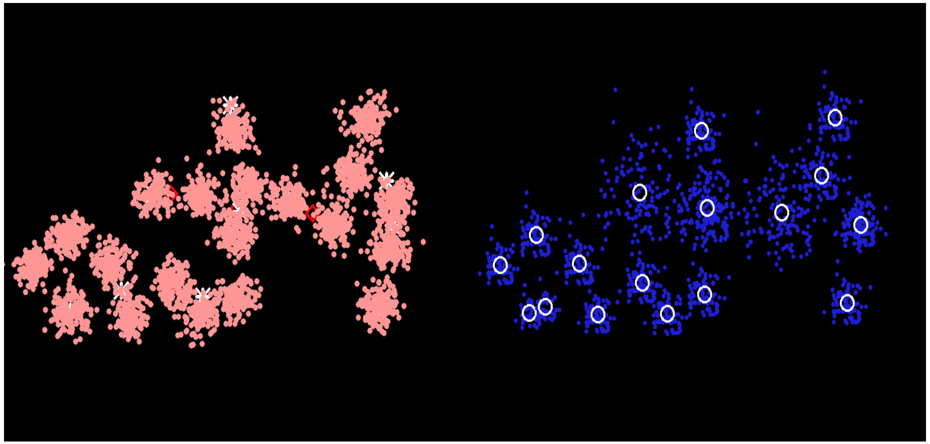
- Clustering a convex shaped cluster (blob) using the k-means algorithm will not “ruin” the regeneration of a sub-cluster.



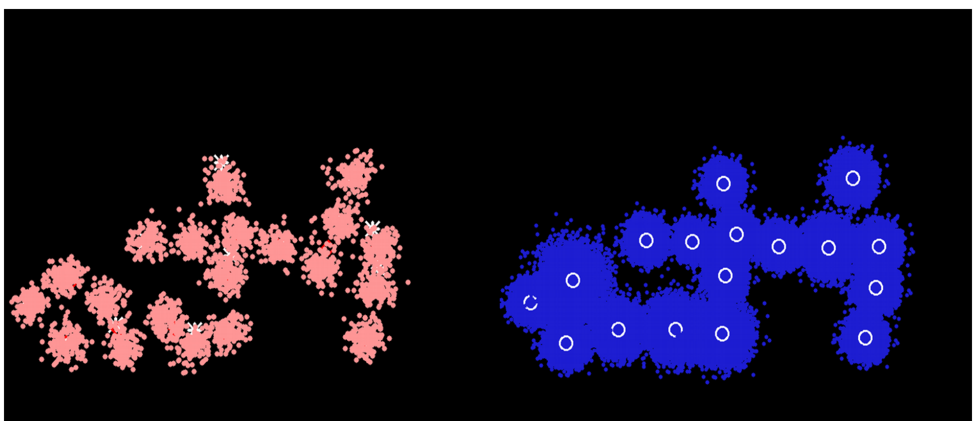
- Clustering a concave shaped cluster will only result in a more detailed (sub-clustered) cluster.



# Results of the hybrid algorithm



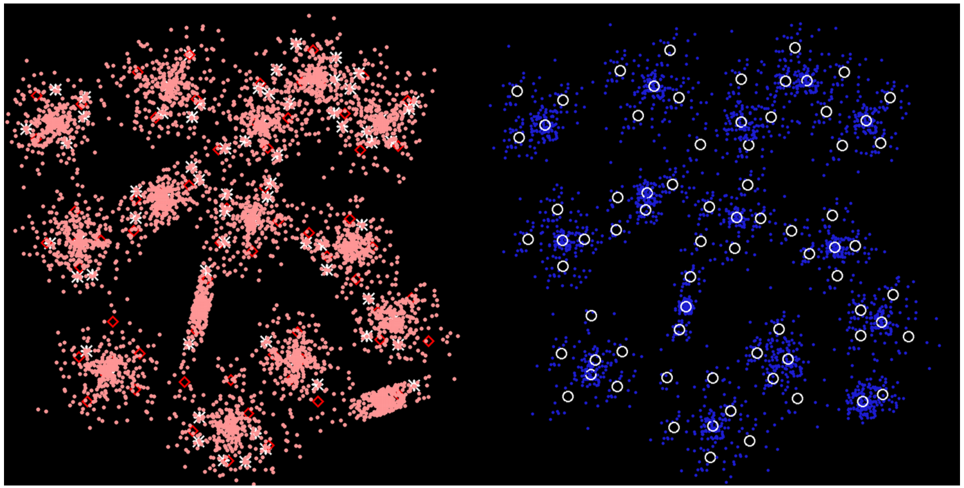
- Data regeneration with the hybrid algorithm – reducing the original number of data points.



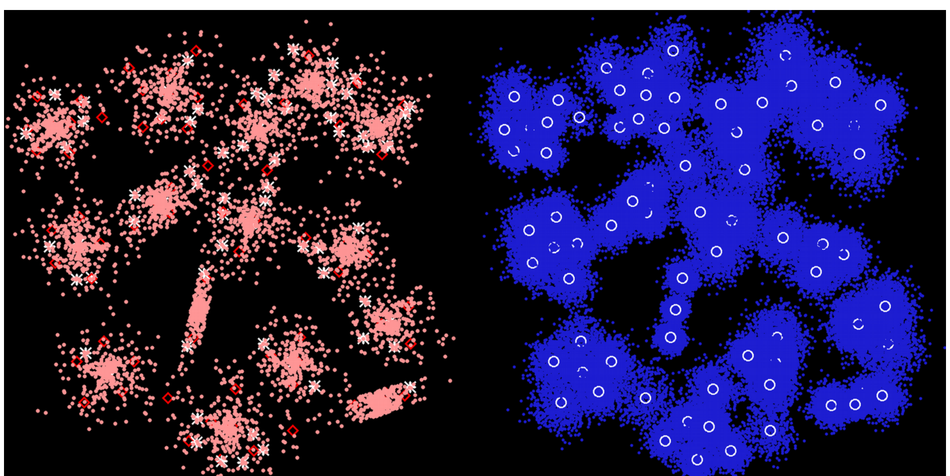
- Data regeneration with the hybrid algorithm – increasing the number of data points 50 times.

~ 5,000 data points in the original dataset

# Results of the hybrid algorithm



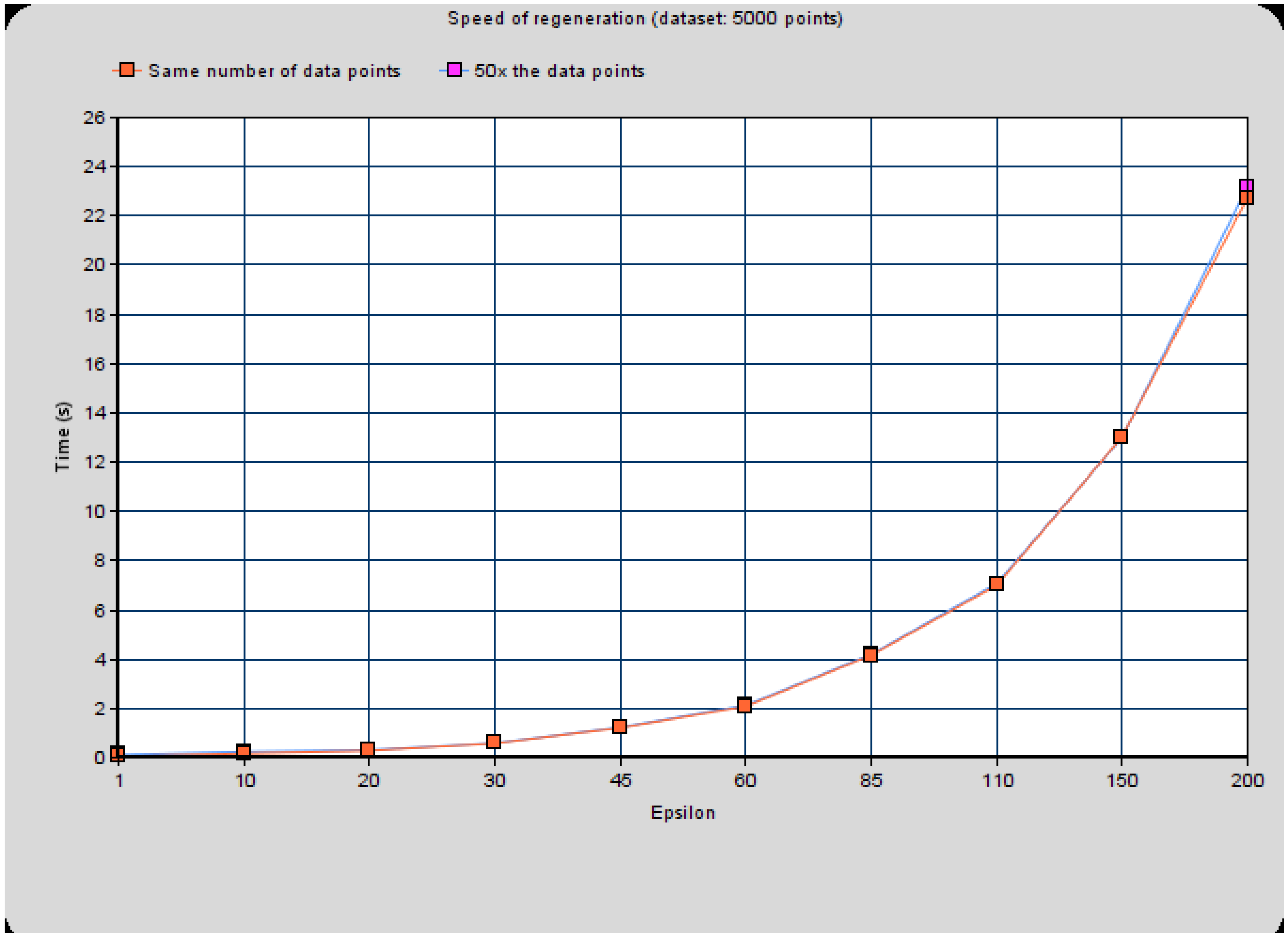
- Data regeneration with the hybrid algorithm – reducing the original number of data points.



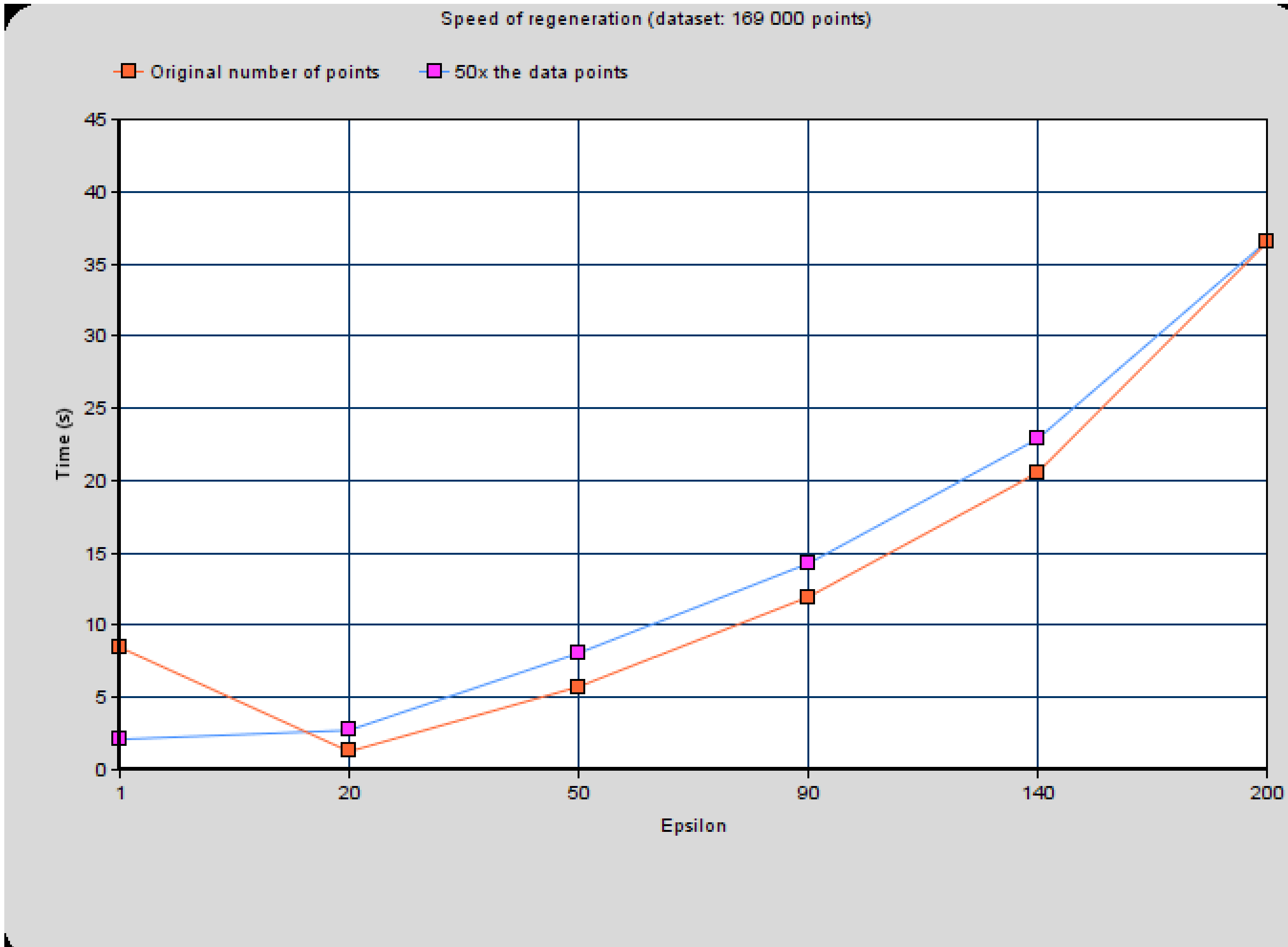
- Data regeneration with the hybrid algorithm – increasing the number of data points 50 times.

~ 169,000 data points in the original dataset

# Speed of the hybrid algorithm



# Speed of the hybrid algorithm



# Further ideas and enhancements

- Extend the algorithm to n > 2 dimensions.
- Find a metric to numerically represent non-numerical data, such as *e-mail address, texts, etc.*
- Find a way to correctly assume the “good” parameters of the DBSCAN algorithm.

# Thank You for your attention

This work was supported by EFOP-3.6.3- VEKOP-16- 2017-00002 project.