# Project

- **Madhur Devkota**

Adopting Machine Learning algorithms for unsupervised classification of water quality data

## 1. Data used

- Four major DOM compositions:
  C1 - terrestrial humic-like, C2 - microbial humic-like, C3 - soil-derived humic-like
- Concentration of nutrients:
  Phosphate ($PO_4^{3-}$) in µgm/L, Nitrate ($NO_3^-$) µgm/L, Ammonium ($NH_4^+$) µgm/L, Nitrite ($NO_2^-$) µgm/L
- Absorption data:
  a254 - UV absorption coefficient at 254 nm, is directly proportional to aromaticity
  a440 - UV absorption coefficient at 440 nm, is directly proportional to Dissolved Organic carbon concentration
- Absorption ratio:
  SR - a ratio of average absorption coefficient from 275 to 295 nm and from 350 to 400 nm wavelengths
  FIX - fluorescence index; a ratio of emission wavelengths at 470 nm divided by emission at 520 nm acquired at the excitation wavelength of 370 nm (McKnight et al., 2001). It is used as an indicator of the source of Dissolved Organic Matter (DOM).
  HIX - humification index; a ratio of peak integrated area of emission spectra between 300 & 345 nm divided by the sum of peak integrated area of emission spectra between 300–345 nm & 435–480 nm at 254 nm excitation (Ohno, 2002). It indicates the degree of humification of natural organic matter.
  BIX - biological index; a ratio of emission intensity at 380 nm divided by the maximum emission
  intensity maximum observed between 420 & 435 nm, obtained at the excitation 310 nm (Parlanti et al., 2000). It is an indicator of the freshness of DOM.
- Land use Land cover data:
  The land use land cover (LULC) data of the Hydrological Unit Code (HUC) 12 watershed is calculated for each sample using GIS tools. The multiple LULC categories were combined into 8 basic categories, namely, water, urban, barren, forest, rangeland, hay pasture, agriculture, and wetland. Their respective area in km$^2$ and their ratio.
- Water Body type:
  It is categorical data indicating water body type, namely, coastal (MS Sound), river, and lake.

  The total observation is 189 with the total original column being 31.

## 2. Data Processing

Following data processing tasks were conducted:

- Getting spatial information:
  The original data have 189 observations with  10 columns. ArcGIS (window-based GIS app) is used to obtain LULC for the watershed that lies on each observation.

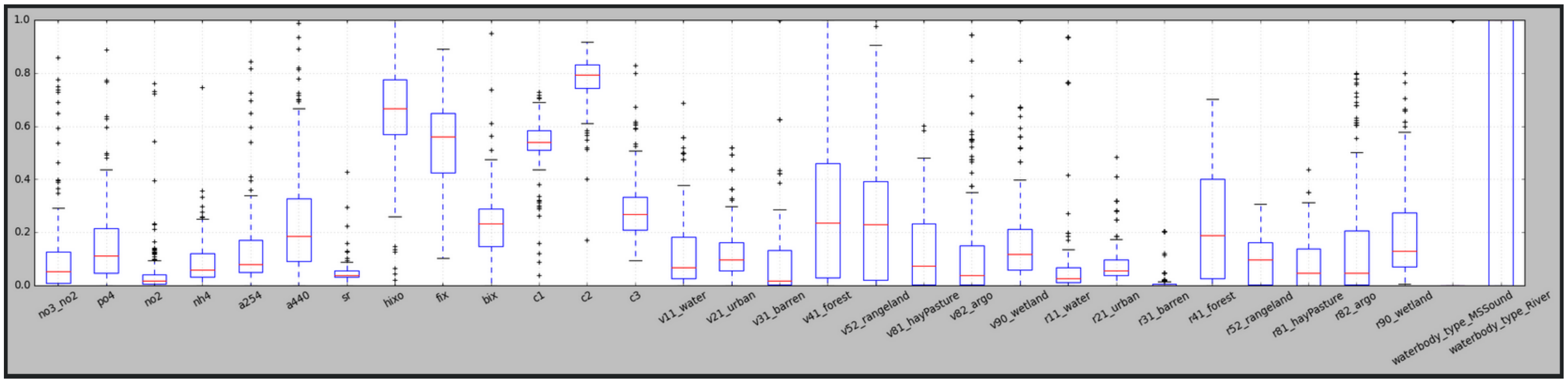
- Data Cleaning and Imputation

There were missing values for multiple features and well outliers (i.e., extremely high values, and negative values for positive features). Outliers were also considered as missing values.

The *MissForest* library is used for imputation to fill in the missing values that use Random Forest in an iterated fashion. The candidate column is selected from all the columns having the least number of missing values. In the first step, all the other columns, i.e., non-candidate columns having missing values, are filled with the mean for the numerical columns and the categorical columns. After that, imputer fits a random forest model with the candidate columns as the outcome variable(target variable) and remaining columns as independent variables and then fills the missing values in the candidate column using the predictions from the fitted Random Forest model. Then the imputer moves on, and the next candidate column is selected with the second least number of missing values, and the process repeats itself for each column with the missing values.
The advantages of *MissForest* over KNN Imputer are:

- is not sensitive to outliers or extreme values
- doesn’t require extensive data preparation — as a Random Forest algorithm can determine which features are important
- doesn’t require any tuning — unlike ‘K’ in K-Nearest Neighbors
- can automatically handle categorical data types without any datatype conversion

After imputation, the data is normalized using MinMaxScaler for the given range (0,1). Following is the distribution (box plot) of the features after imputation and normalization:

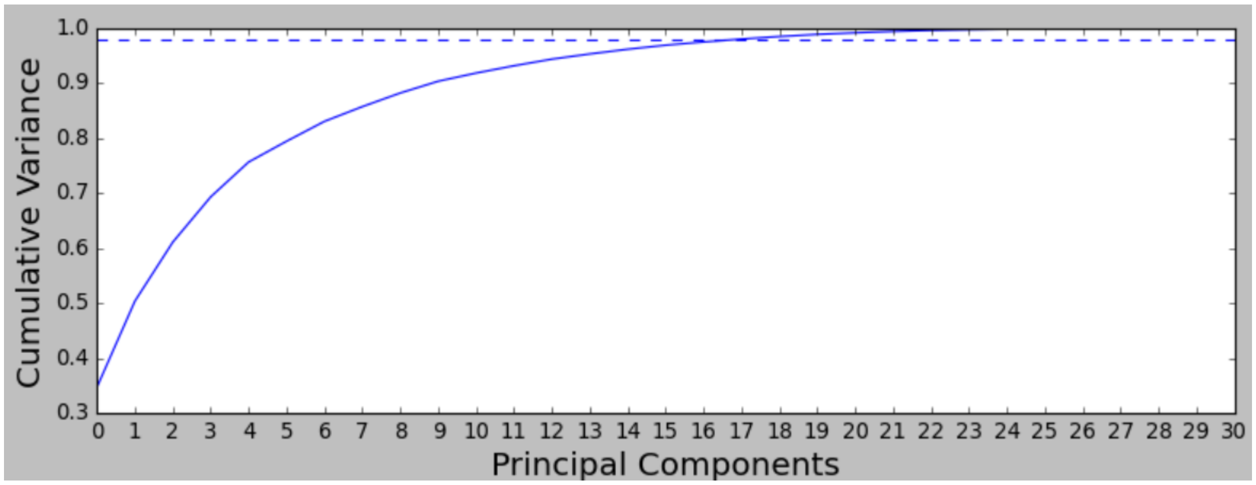


Even after removing outliers and imputing, and normalizing, there are ample extreme values. Different algorithms are tested to address these extreme values.

## 3. Methodology

- **Preparation prior to adopting ML models**

  Principal Components were calculated using all 31 features. Out of 31 features, the first 18 Principal Components (PC) were considered which contained more than 98% of the variation as shown in the plot below:



- **Adopting ML models**

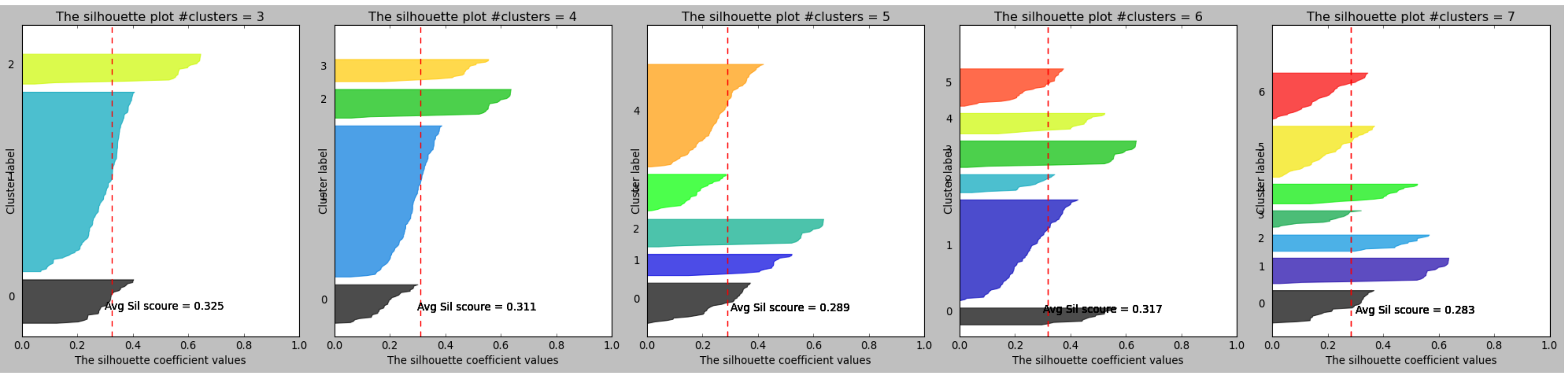
  Various Machine Learning unsupervised models were adopted, and hyper-parameter tuning is conducted to find the best hyperparameters for each model.

## K Means Clustering

This is the base model where no hyperparameter tuning is conducted. The clustering is implemented using the following hyperparameter value:

```
sklearn.cluster.KMeans(  init= 'k-means++', max_iter= 1000, n_init= 3*18,
                         algorithm= 'full', random_state= 10  )
```

The Silhouette score for different clusters (from 3 to 7) obtained from K Means clustering is calculated and plotted as:



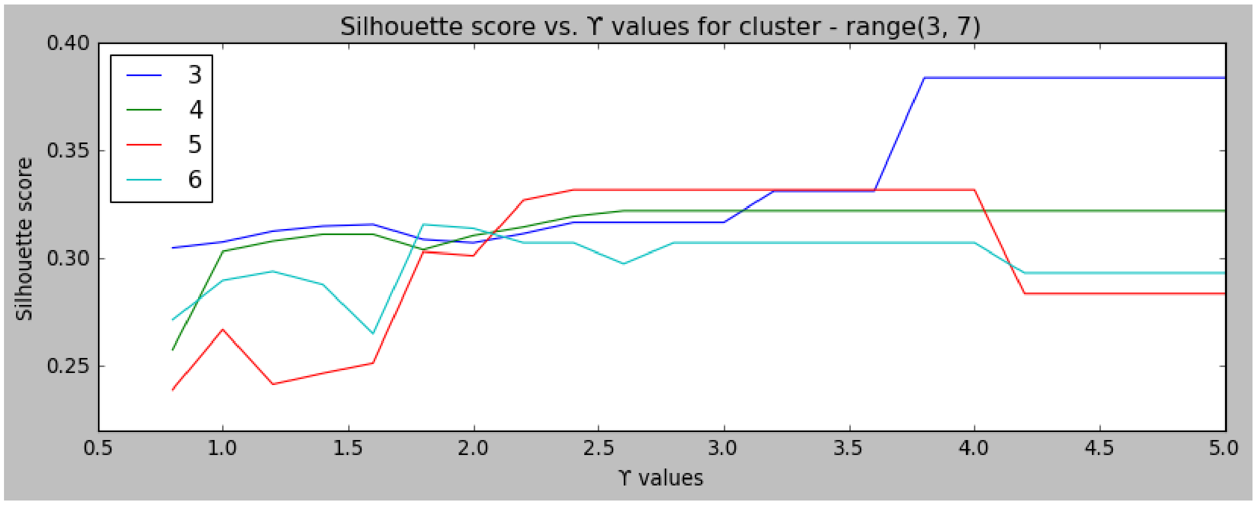
## Spectral Clustering

Spectral clustering is a technique with roots in graph theory, where the approach is used to identify communities of nodes in a graph based on the edges connecting them. The method is flexible and allows us to cluster non graph data as well.

Next Spectral clustering is used with the following hyperparameters:

```
sklearn.cluster.SpectralClustering( eigen_solver= 'arpack', affinity= 'rbf',
                                    assign_labels= 'discretize',
                                    random_state = 10, n_jobs= -1 )
```

The hyperparameter that was tuned is *gamma*. For a different value of *gamma* and a different number of clusters, calculated Silhouette scores are:



## Agglomerative Clustering

This clustering recursively merges pairs of clusters based on linkage distance. Following hyperparameters were tuned for various numbers of clusters:

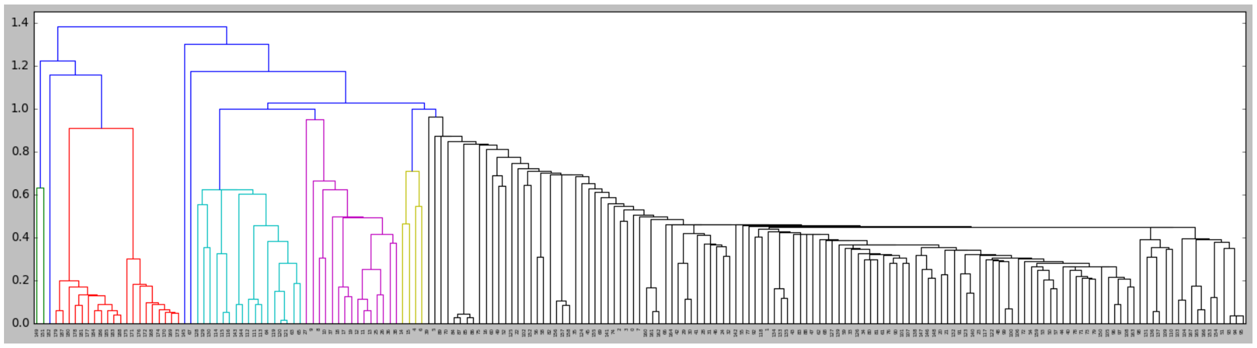
```
sklearn.cluster.AgglomerativeClustering( affinity, linkage )
```

The hyperparameters values for a different number of clusters are:

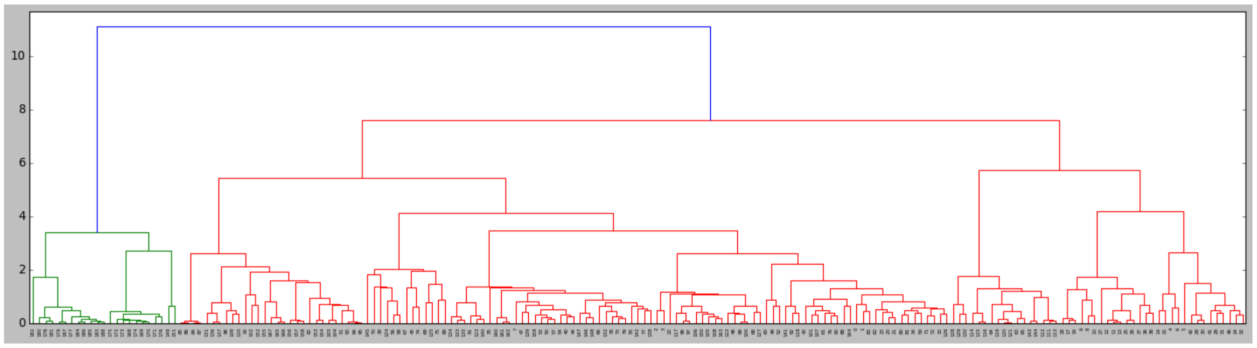
```
linkage = [ 'ward', 'complete', 'average', 'single' ]

affinity = [ 'euclidean', 'l1', 'l2', 'manhattan', 'cosine' ]
```

Following is the dendrogram based on Agglomerate Clustering with default hyperparameter value.



Following is the dendrogram based on Agglomerate clustering using "ward" as the linkage algorithm. Thus, it is apparent that the hyperparameter tuned agglomerate model yields completely different clusters.



## Density-based spatial clustering of applications with noise - DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm that has a high-performance rate for a dataset where clusters have the constant density of data points. One of the significant attributes of this algorithm is noise cancellation.

Advantages of DBSCAN:

- Arbitrarily shaped clusters
- Robust to outliers
- Doesn't require specification of the number of clusters

**DBSCAN is used with the following hyperparameters:**

```
sklearn.cluster.DBSCAN( algorithm= 'brute', metric= 'euclidean',
                        eps, minPoints )
```

The hyperparameters that were tuned are:
eps: specifies how close points should be to each other to be considered a part of a cluster. It means that if the distance between two points is lower or equal to this value (*eps*), these points are considered neighbors.
minPoints: the minimum number of points to form a dense region. For example, if we set the minPoints parameter as 6, then we need at least 6 points to form a dense region.

## Hierarchical Density-based spatial clustering of applications with noise - HDBSCAN

HDBSCAN is a clustering algorithm developed by Campello, Moulavi, and Sander. It extends DBSCAN by converting it into a hierarchical clustering algorithm, and then using a technique to extract a flat clustering based in the stability of clusters. It is also a Density-based spatial clustering of applications with noise that has a high-performance rate for a dataset where clusters have the varying density of data points.

HDBSCAN is robust to data that has:

- Clusters with arbitrary shapes
- Clusters of different sizes
- Clusters with different densities
- Some noise and maybe some outliers

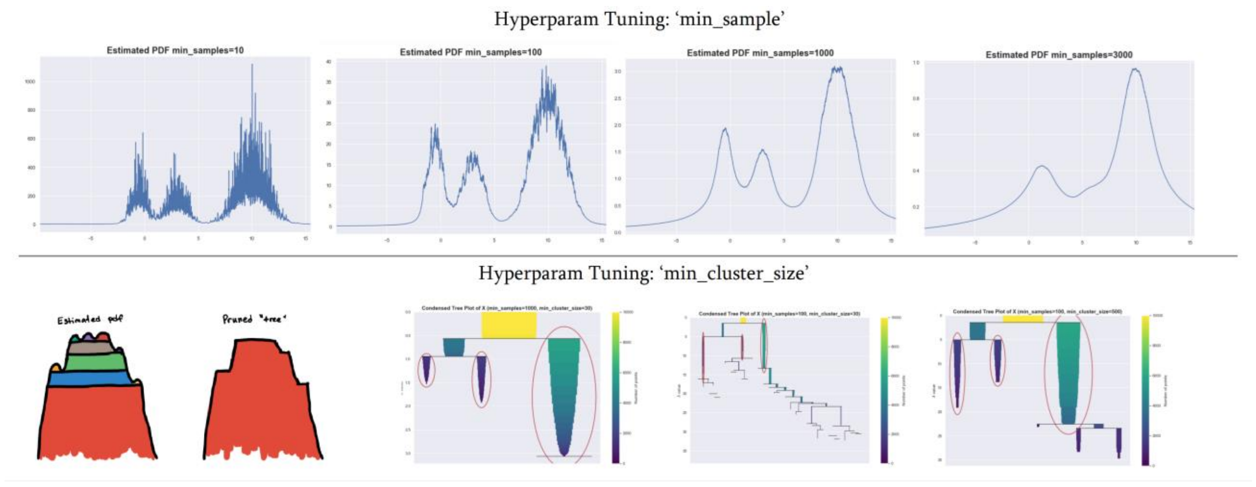
HDBSCAN is used with the following hyperparameters:

```
hdbscan.HDBSCAN(  min_cluster_size, cluster_selection_epsilon,
                  cluster_selection_method  )
```

min_cluster_size – The minimum size of clusters; single linkage splits that contain fewer points than this will be considered points “falling out” of a cluster rather than a cluster splitting into two new clusters.
min_samples - The number of samples in a neighborhood for a point to be considered core points. The simplest intuition for what min_samples does is provide a measure of how conservative one wants the clustering to be. The larger the value of min_samples, the more conservative the clustering – more points will be declared as noise, and clusters will be restricted to progressively more dense areas.
cluster_selection_method – ‘eom’ or ‘leaf’. The method used to select clusters from the condensed tree.

Note: the cluster_selection_method doesn’t impact the metrics but it yields different number of clusters as will be observed later on in the Result section.

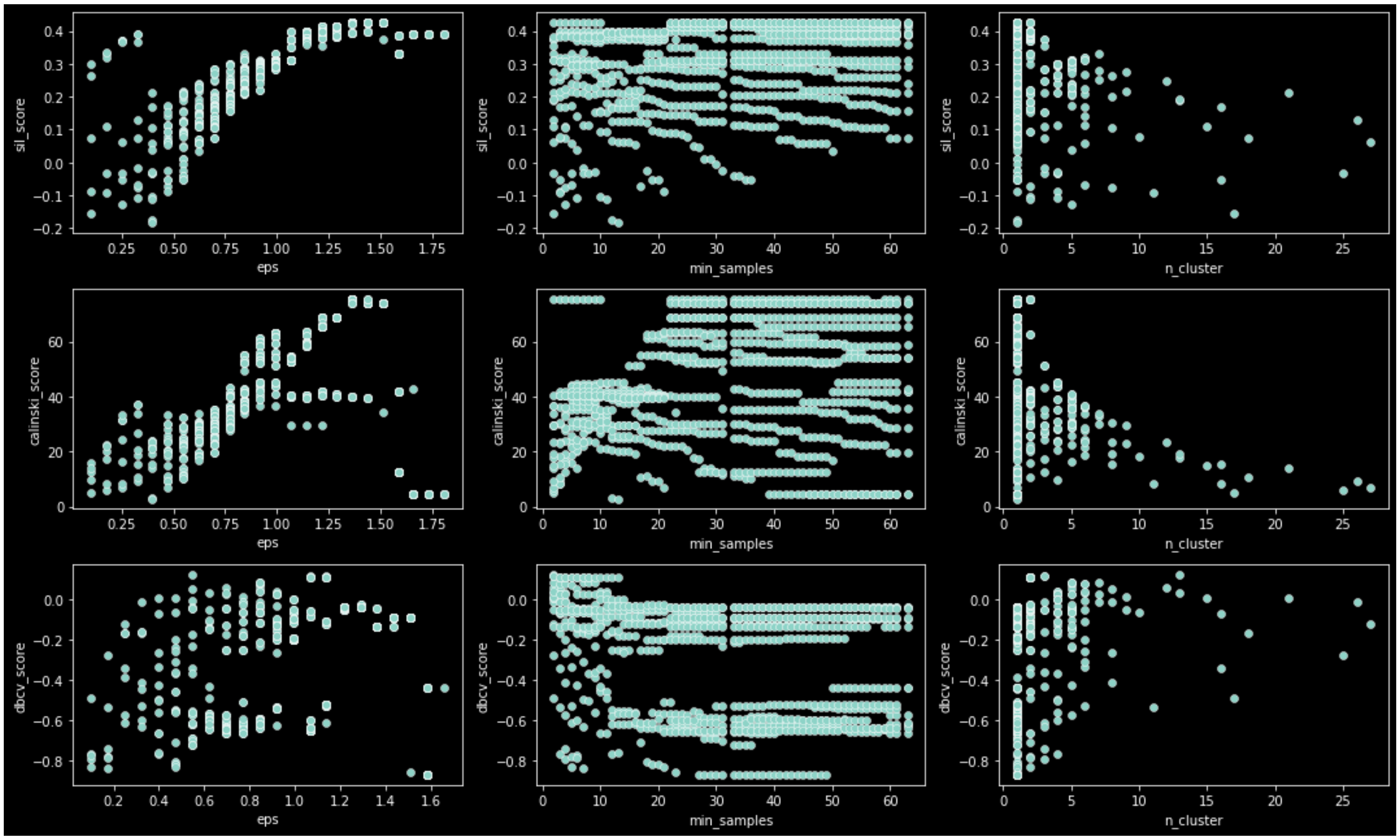
The following plot illustrates the hyperparameter tuning in HDBSCAN:



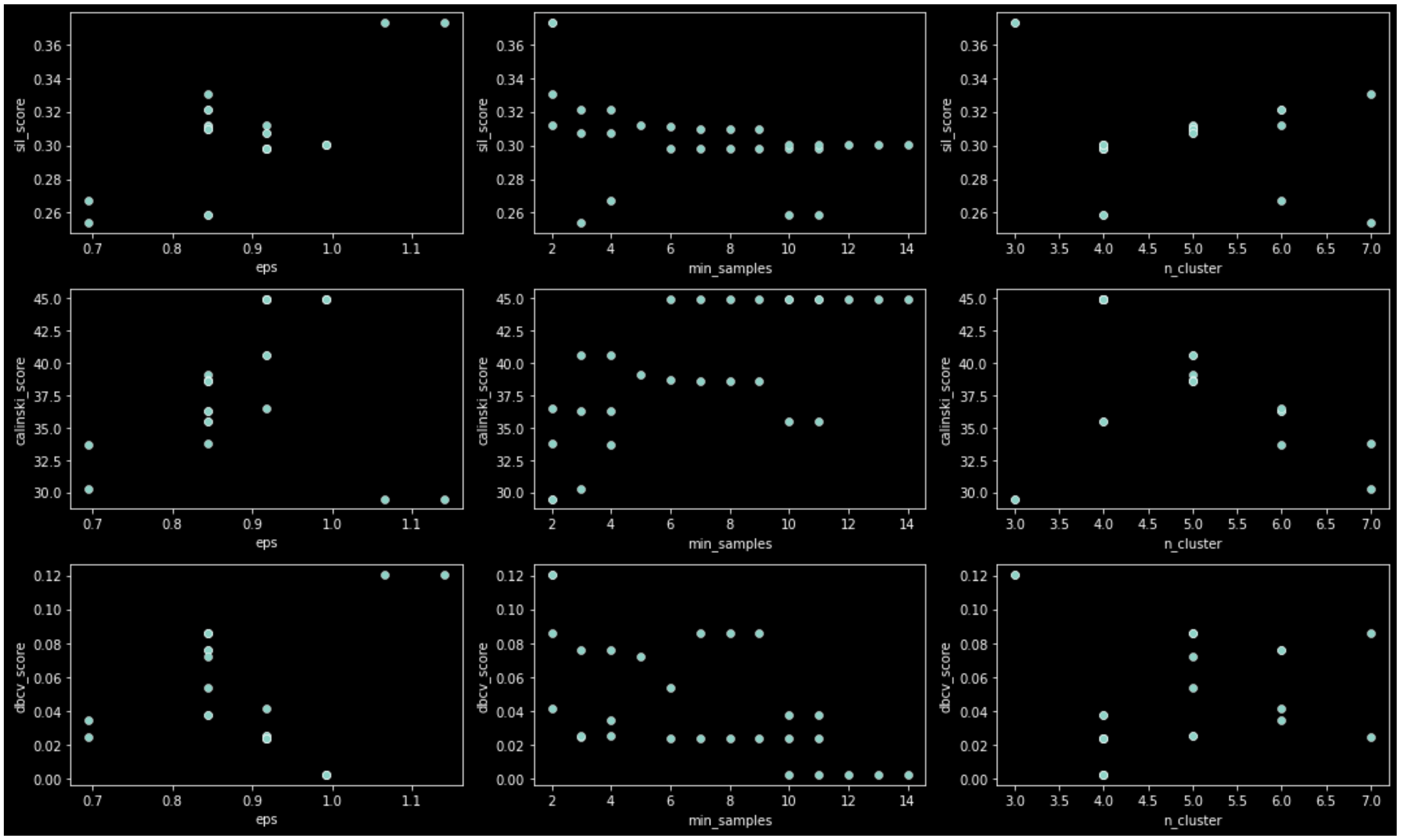
- Hyperparameter Tuning

A heuristic method, namely, “coarse-to-fine” is used for hyperparameter tuning. Firstly, a higher range of each hyperparameter is chosen and multiple metrics for the randomly selected combination of hyperparameters are calculated. The narrower bandwidth of hyperparameters was considered for the next hyperparameter tuning. Either Random Search or Grid Search is calculated depending upon the total number of combinations of hyperparameters.

For example, in the case of DBSCAN, the 1st hyper-parameter tuning would look something like this where three metrics, namely, sil_score, calsinki_score, dbcv_score, were calculated for each combination of hyper-parameters (eps and min_samples).



Based on this result, the next iteration of hyperparameter tuning is conducted (could be Grid or Random Search depending upon the total number of hyperparameters combination). The result of the next hyperparameter tuning resulted as:



Now, these plots help us find the optimum model using the best hyperparameters.

Next, multiple metrics were calculated for all the Unsupervised classification models using best respective hyperparameters values and compared among themselves for different cluster numbers (3, 4, 5, 6, 7). Based on these results, the best model with optimum hyperparameters is chosen.

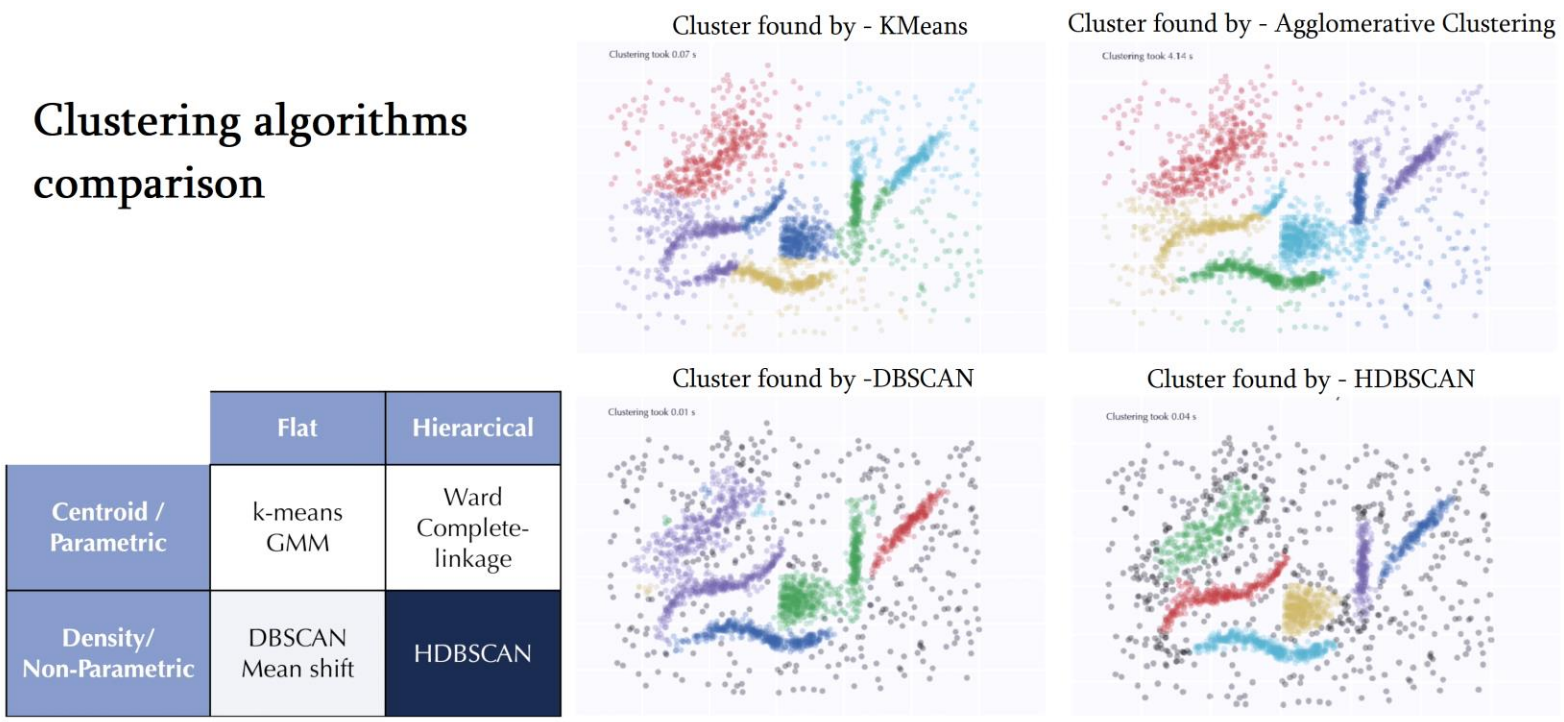
## 4. Result & Conclusion

The optimum hyperparameters for respective models are as follows:

| **Model type** | **Model** | **Best Hyperparameters values** |
|---|---|---|
| Centroid-based, Parametric, Flat | **KMeans** | init = 'k-means++' \| n_init = 54 |
| Graph-based | **Spectral Clustering** | eigen_solver = 'arpack' \| gamma' = 3.7, affinity = 'rbf' \| assign_labels = 'discretize' |
| Centroid-based, Parametric, Hierarchical | **Agglomerative Clustering** | Affinity = 'l2' \| linkage = 'average' |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| Density-based, Non-Parametric, Hierarchical | **DBSCAN** | **eps** | **min_samples** | **n_cluster** | |
| | | 1.15 | 2 | 3 | |
| | | 0.90 | 10 | 4 | |
| | | 0.85 | 7 | 5 | |
| | | 0.85 | 3 | 6 | |
| | | 0.85 | 2 | 7 | |
| | **HDBSCAN** | **min_cluster_size** | **min_samples** | **cluster_selection_method** | **n_cluster** |
| | | 22 | 6 | 'eom' | 3 |
| | | 13 | 6 | 'eom' | 4 |
| | | 9 | 6 | 'eom' | 5 |
| | | 5 | 6 | 'leaf' | 6 |
| | | 5 | 6 | 'leaf' | 7 |

The comparison of the models is illustrated in the diagram:



| | Flat | Hierarcical |
|---|---|---|
| Centroid / Parametric | k-means GMM | Ward Complete-linkage |
| Density/ Non-Parametric | DBSCAN Mean shift | HDBSCAN |

HDBSCAN has both the advantage of Density, non-parametric based clustering, and hierarchical clustering. It also doesn’t force extreme/outliers into clusters and marks them as noises unlike other clustering models used in this study.

The metrics used in each clustering model are –

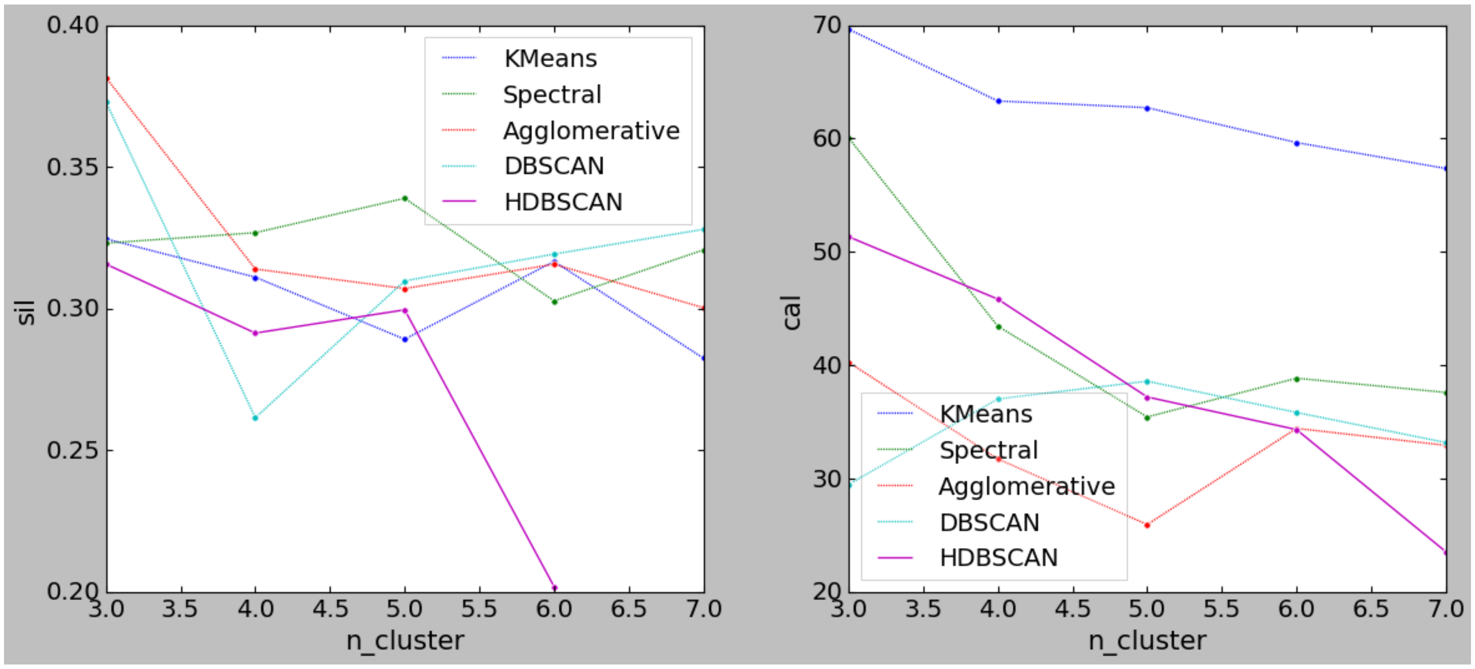
- Silhouette score - Itis calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample i.e., b is the distance between a sample and the nearest cluster that the sample is not a part of.
  The Silhouette Coefficient for a sample is (b - a) / max(a, b).

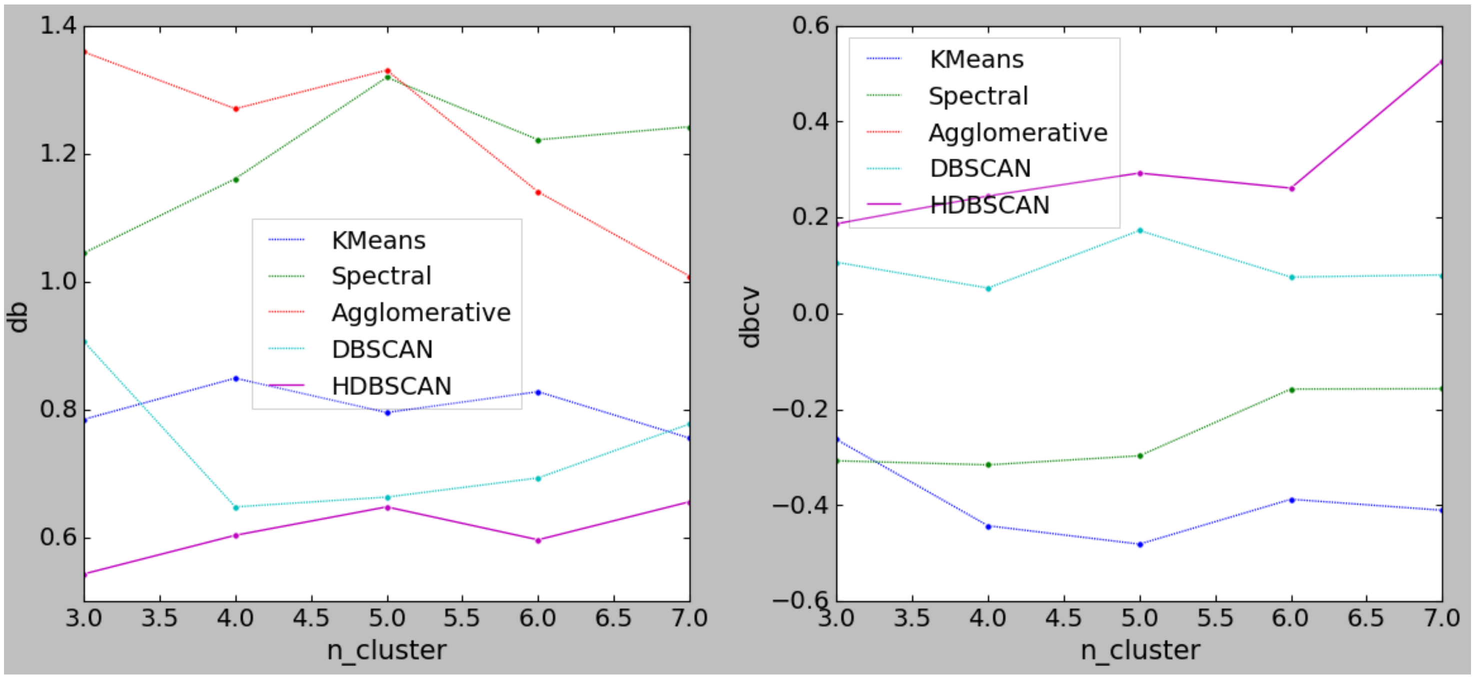
- Calinski and Harabasz score - The score is defined as the ratio between the within-cluster dispersion and the between-cluster dispersion.

- The inverse of Davies-Bouldin score - The score is defined as the average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances. Thus, clusters that are farther apart and less dispersed will result in a better score. The minimum score is zero, with lower values indicating better clustering. Thus, for the inverse of the Davies-Bouldin score, the max the score, the better the clustering.

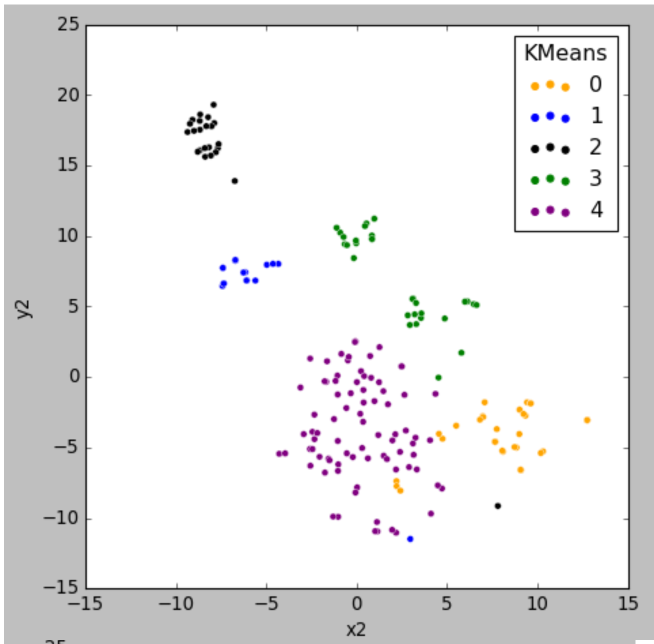
- Density-Based Clustering Validation (DBCV) score - DBCV works for density-based clustering algorithms precisely because it takes noise into account and captures the shape property of clusters via densities and not distances. As the paper explains, the final result of DBCV is a weighted sum of “Validity Index” values of clusters. This produces a score between -1 to 1, with the larger the value the better the clustering solution.

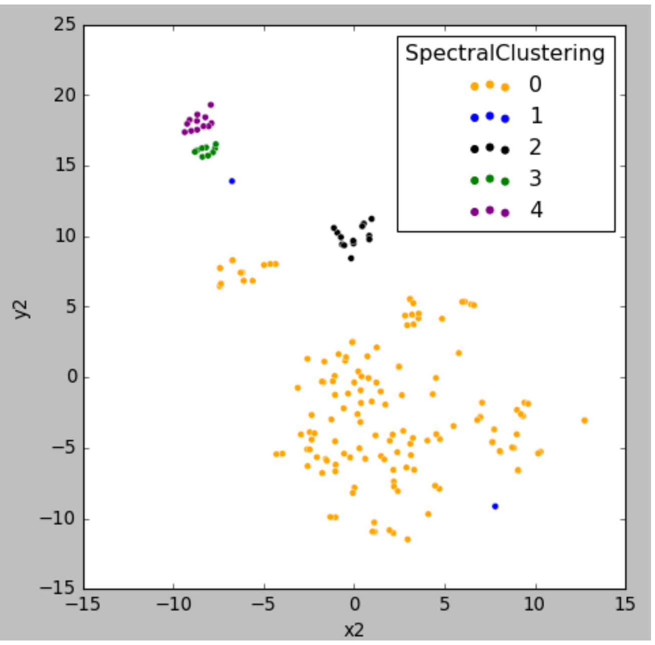
The plots of the metrics for hyperparameter-tuned each model are:

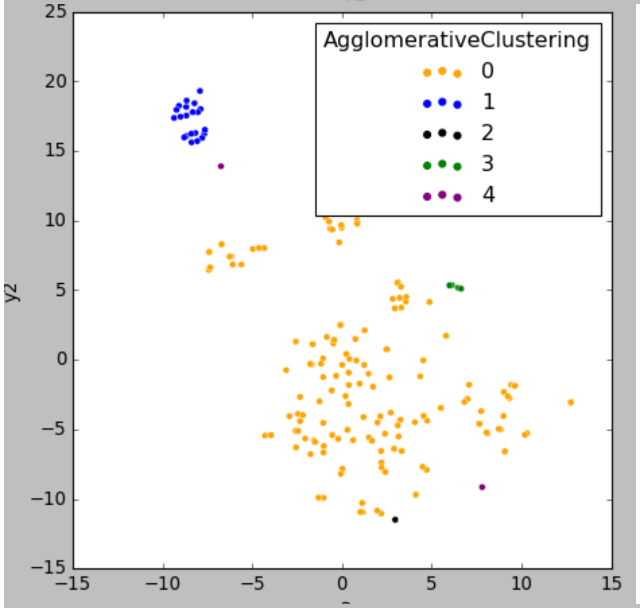




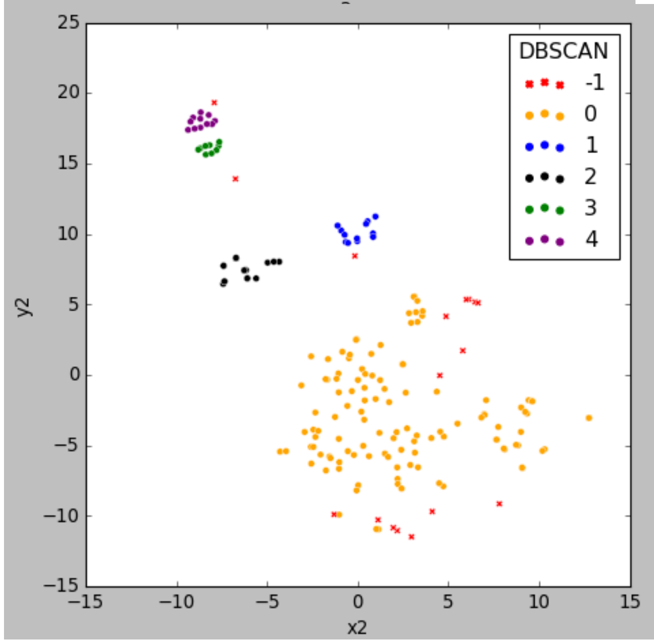
The HDBSCAN performed best if we considered the DBCV metric.

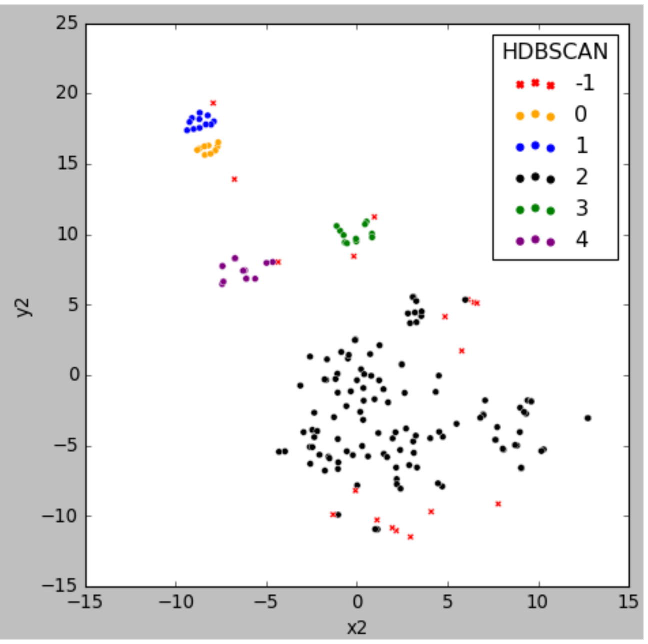




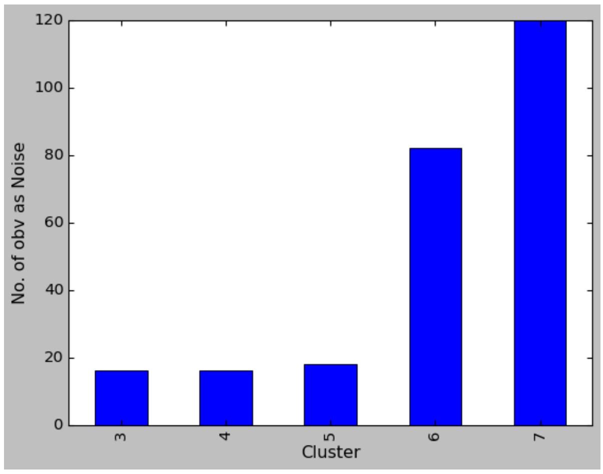


The clustering of the 18 Principal Components of the data using t-SNE for cluster number = 5, for different algorithms.





The HDBSCAN performed best if we considered the DBCV metric for 5 or 7 clusters. When cluster number = 7, the noise increased by a huge amount (120 in cluster 7 vs 18 in cluster 5). Thus, 5 Clusters classification is chosen for optimized Classification/clustering.

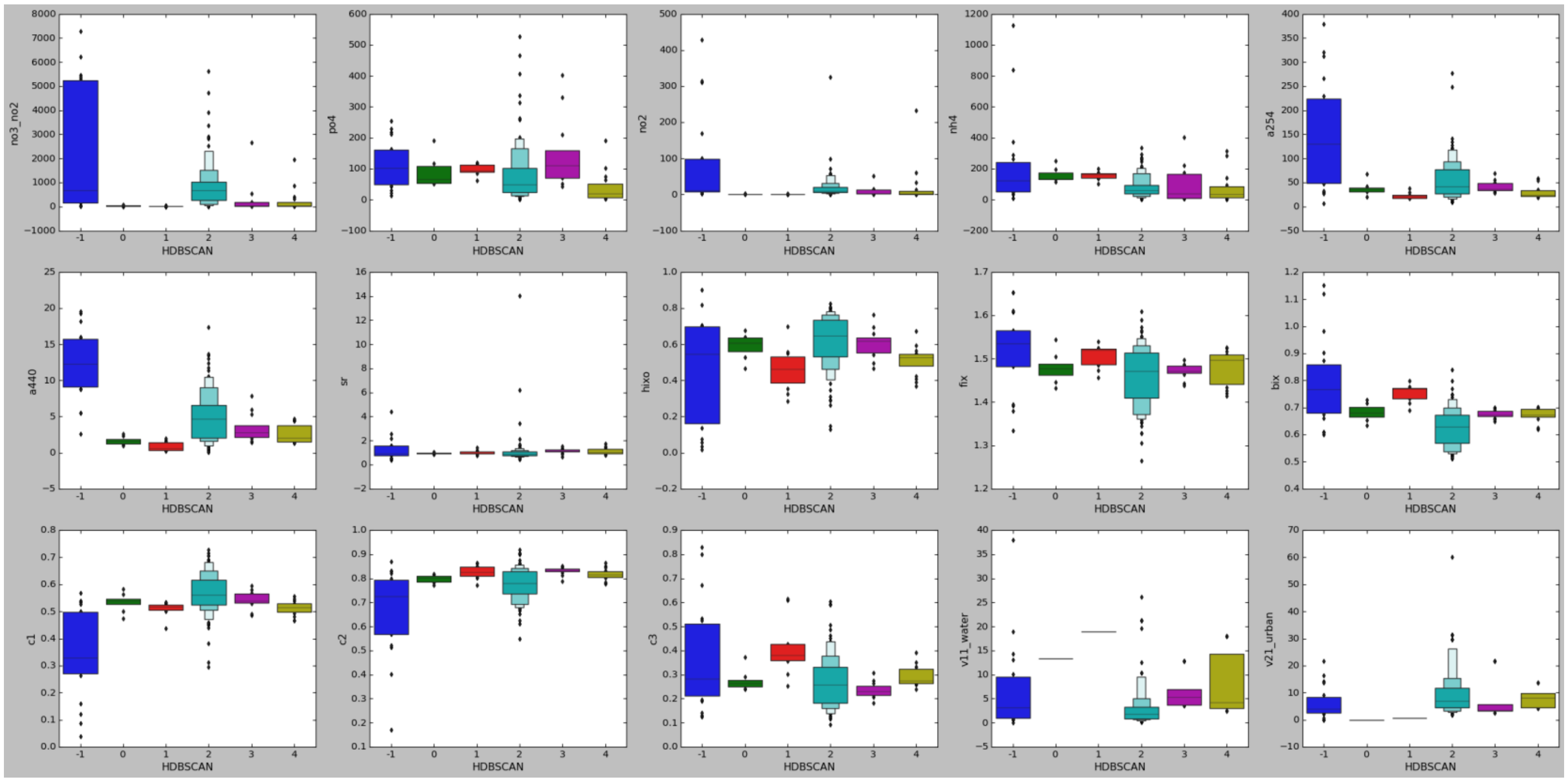


The number of observations designated as noise for various clustering is shown as a bar plot. →

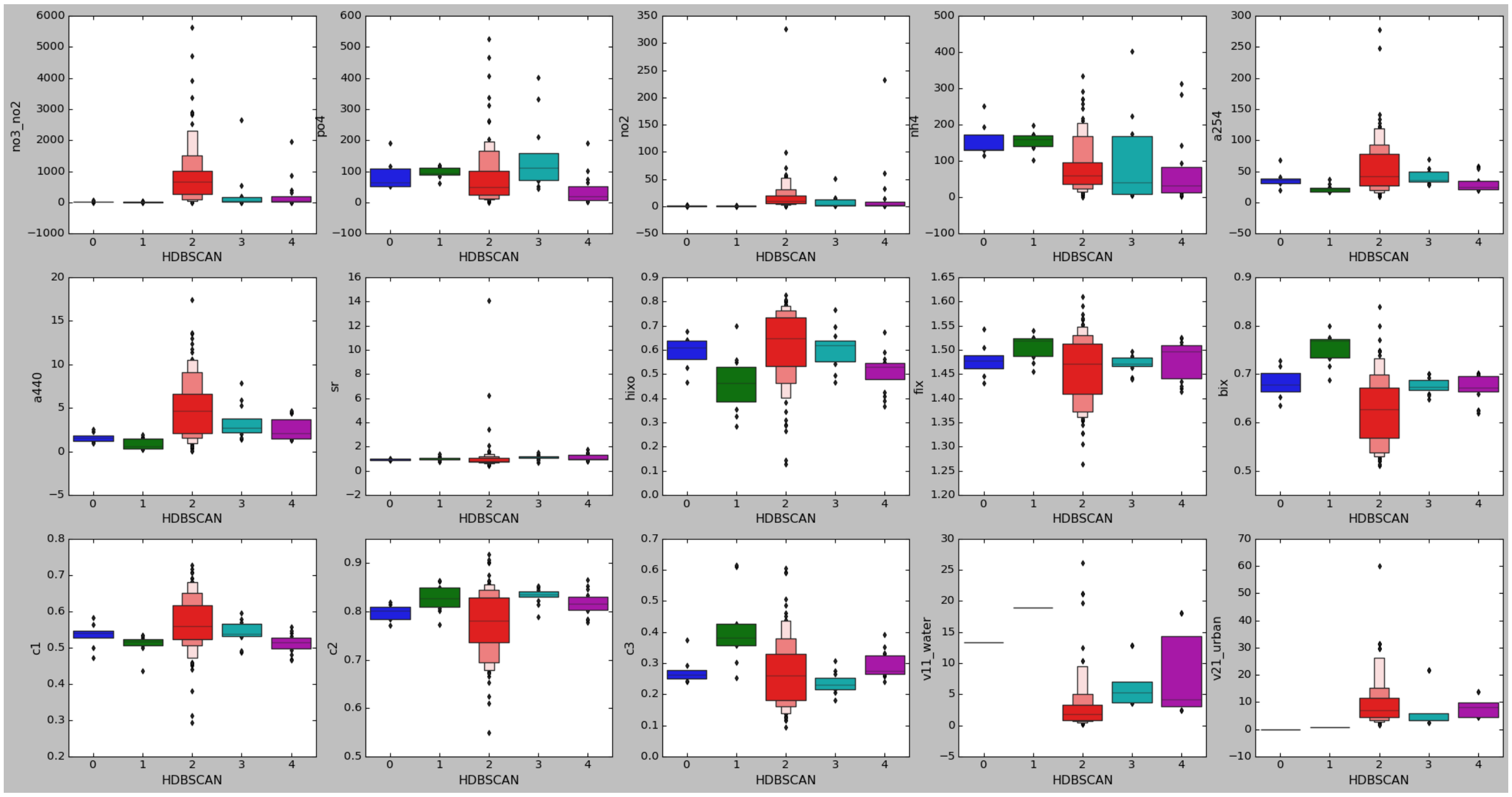
The other metrics (Silhouette, Calinski-and-Harabasz, Davies-Bouldin) used in this study are suitable for KMeans, Spectral, Agglomerate clustering but not for density-based clustering (DBSCAN & HDBSCAN). DBCV is suitable for density-based clustering as it accounts for varying density and noises. The distribution of our data is different (shown later on the t-SNE plot), thus classifying HDBSCAN is relevant than other classification models.

The boxen-plot distribution of the water chemistry parameter when used HDBSCAN with cluster number = 5 are:

With Noise data →

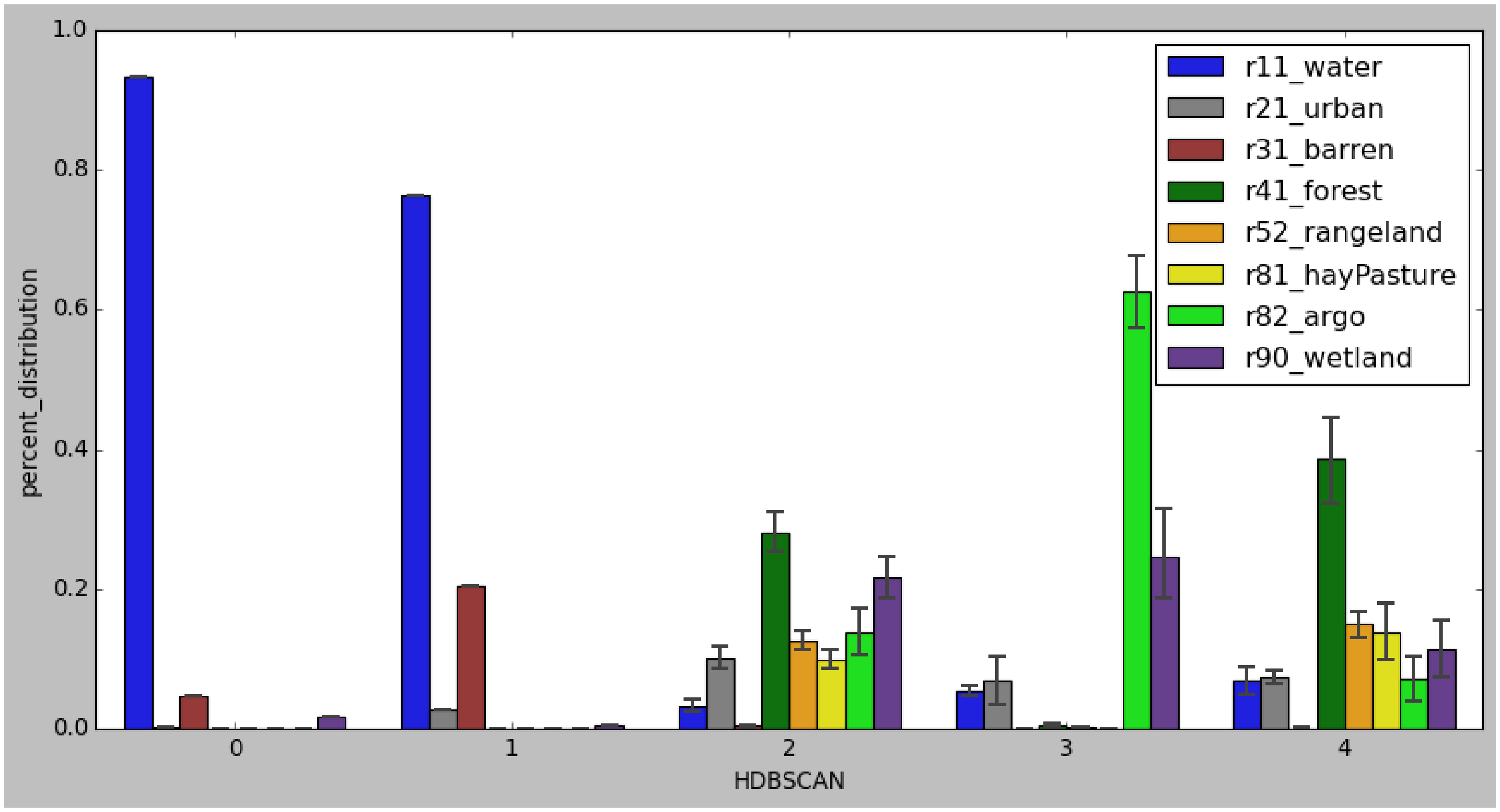


Without Noise data →

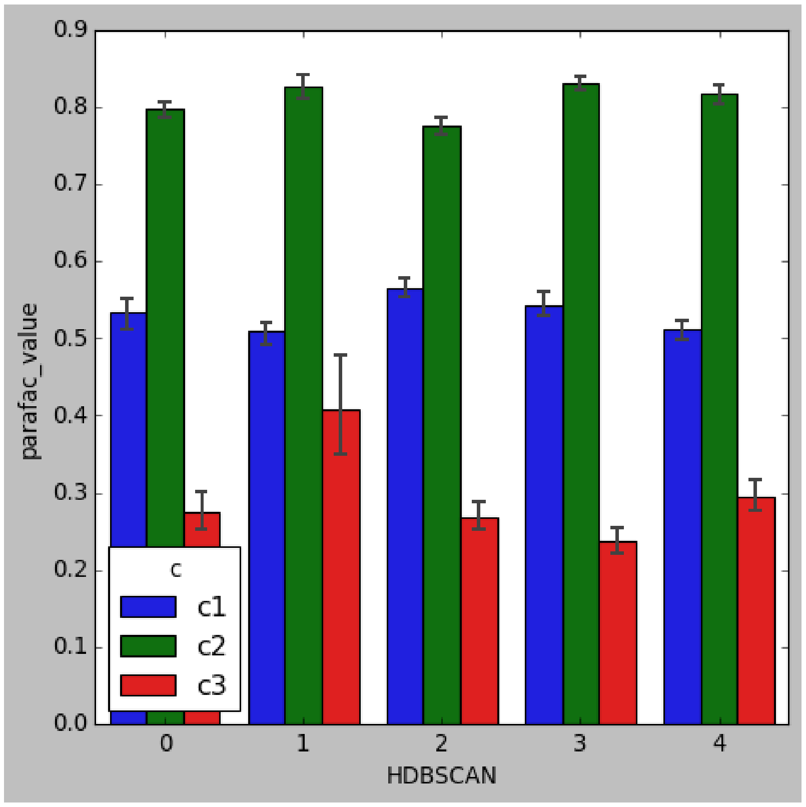


From the two images, it is apparent that the ranges of the noise are much higher than the parameters' values. So, ignoring the noise and forcefully clustering it with the nearest cluster is a bad idea. Thus, HDBSCAN points out noise and doesn't cluster it to any clusters.

The percent distribution of the land use land cover for the different clusters is shown using the following bar plot.

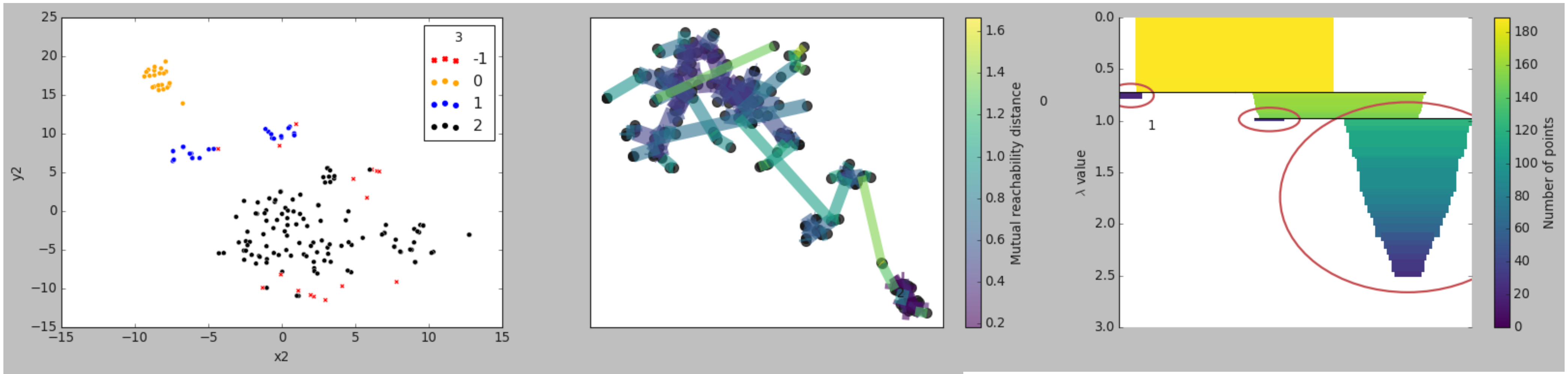


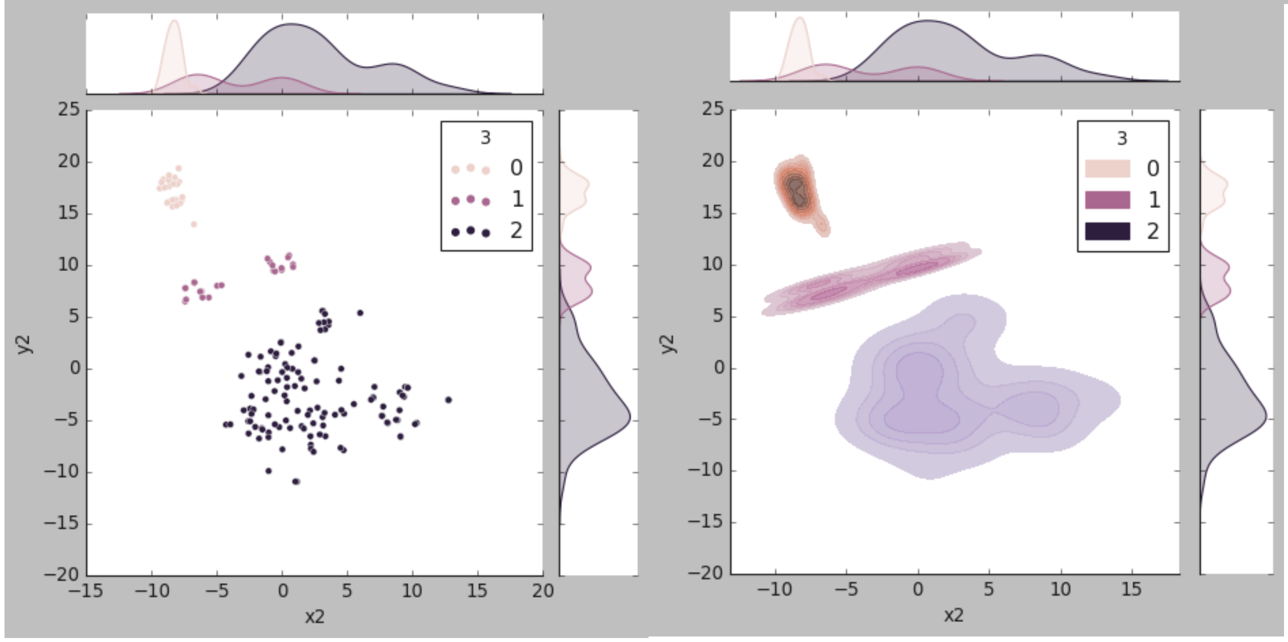
The distribution of the PARAFAC model (parameters C1, C2, C3) for different clusters is shown using a bar plot. →



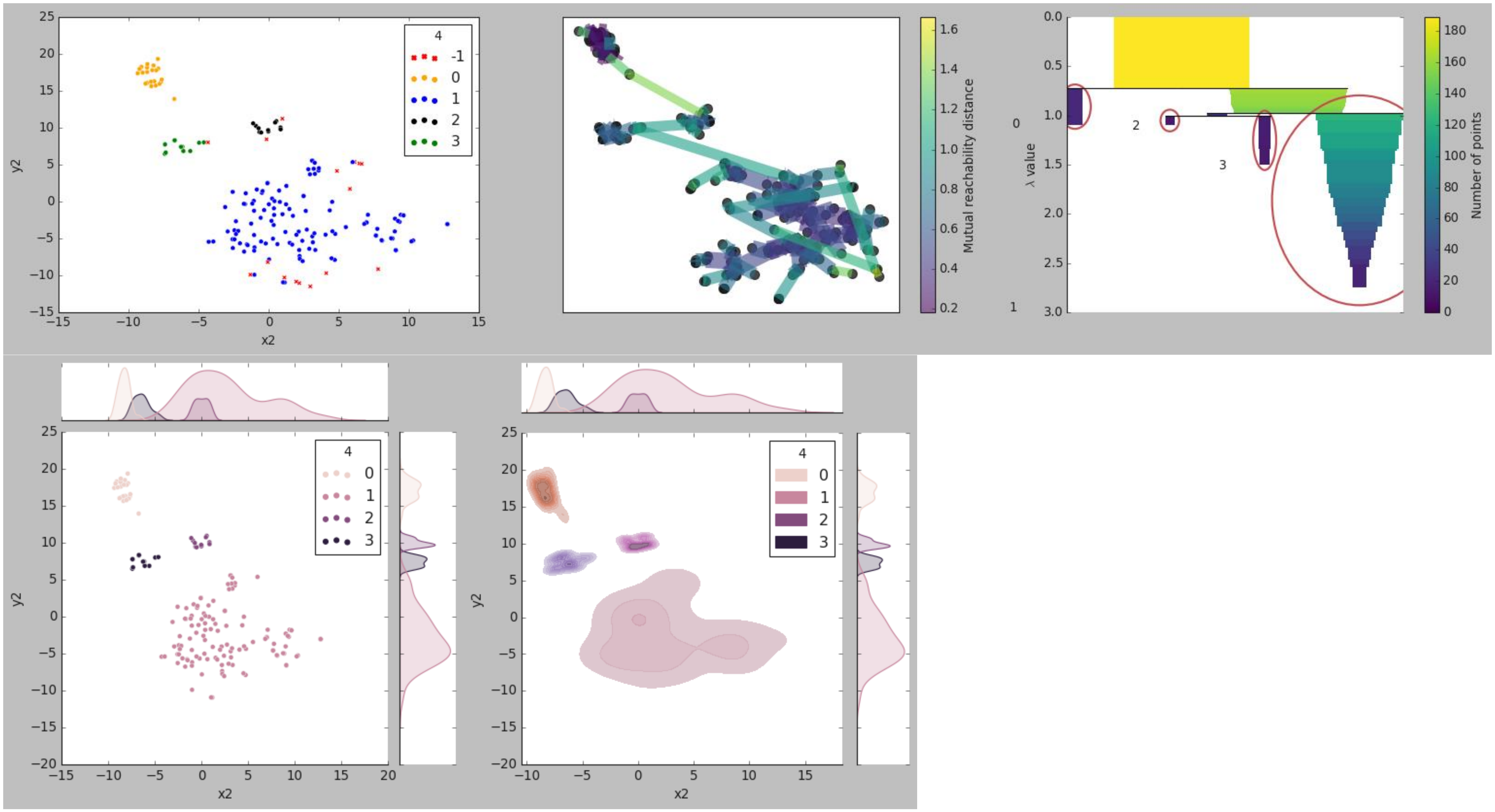
**Scatter plot, Minimum Spanning Tree plot, Condensed Tree plot, Scatter plot with KDE, KDE plot with KDE** for each cluster 3, 4, 5, 6, 7 are:

**# Cluster = 3**



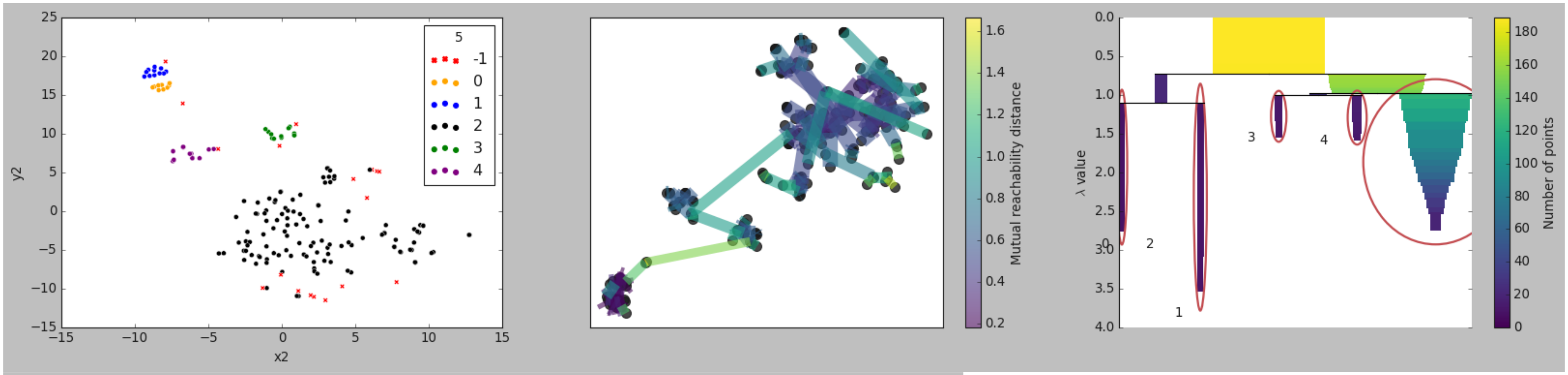


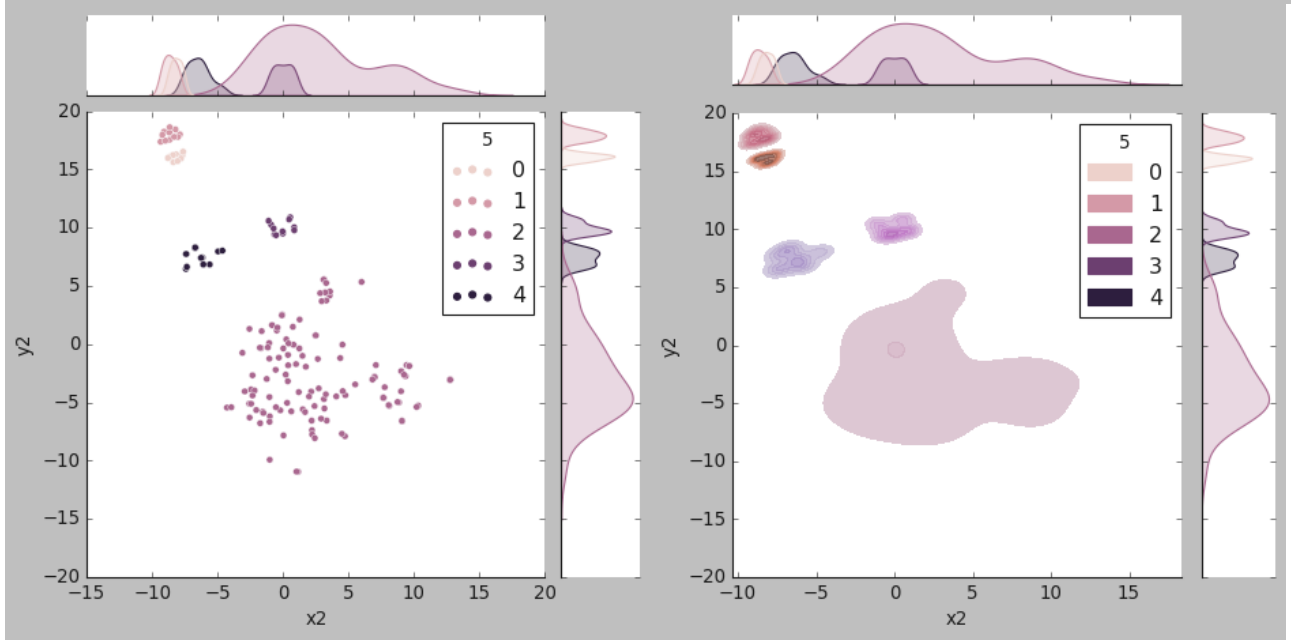
# Cluster = 4



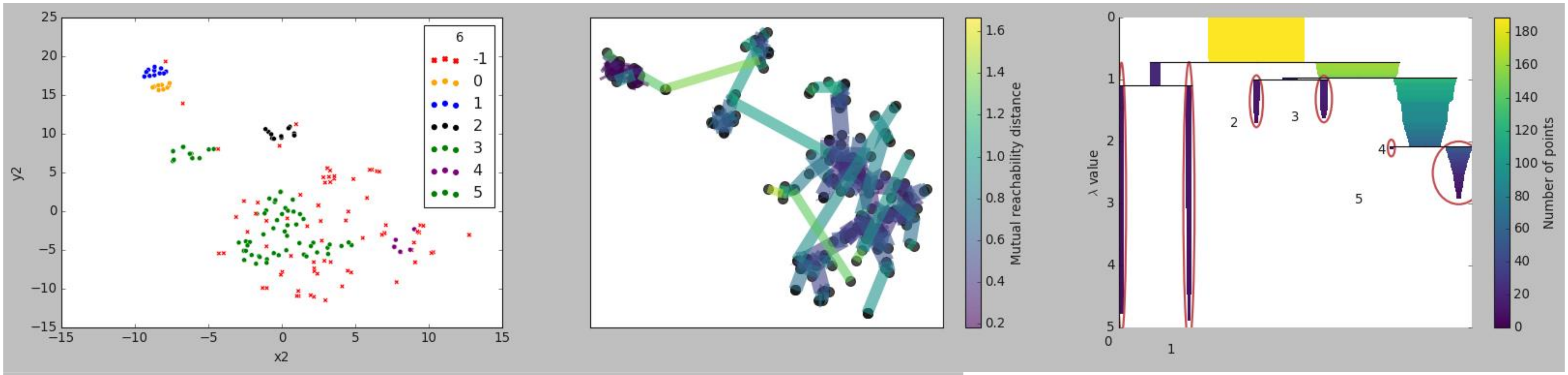
4
-1
0
1
2
3
x2
y2
Mutual reachability distance
λ value
Number of points

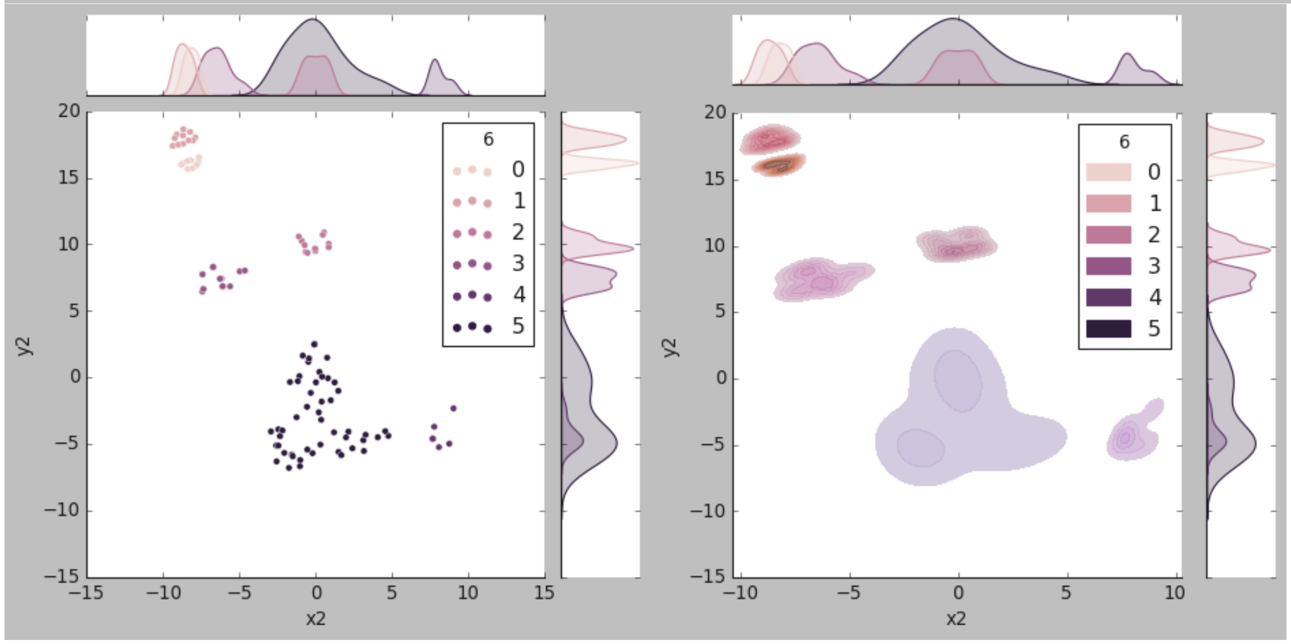
# Cluster = 5


5
-1
0
1
2
3
4
x2
y2
Mutual reachability distance
λ value
Number of points
0
1
2
3
4

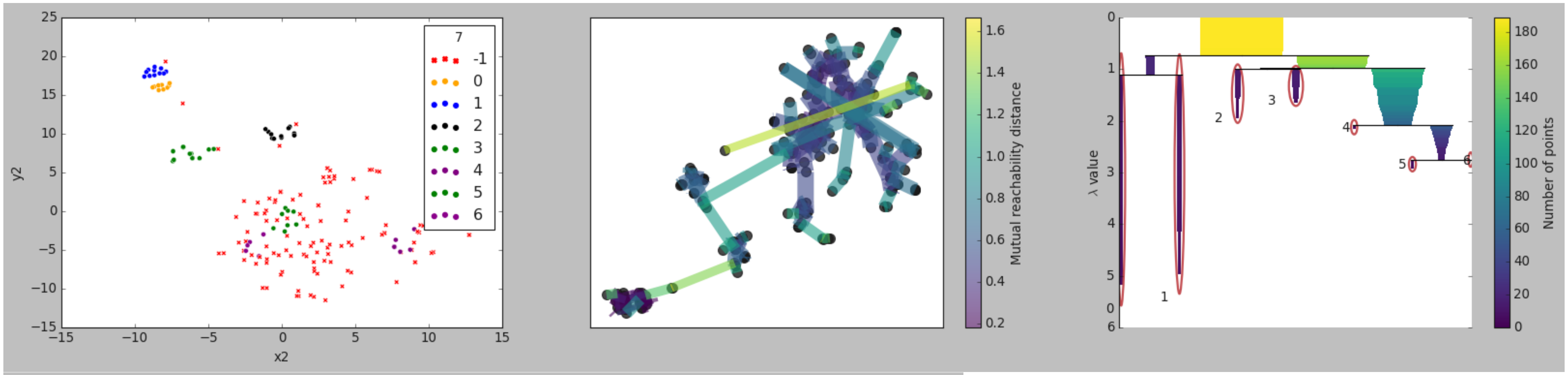

5
0
1
2
3
4
x2
y2

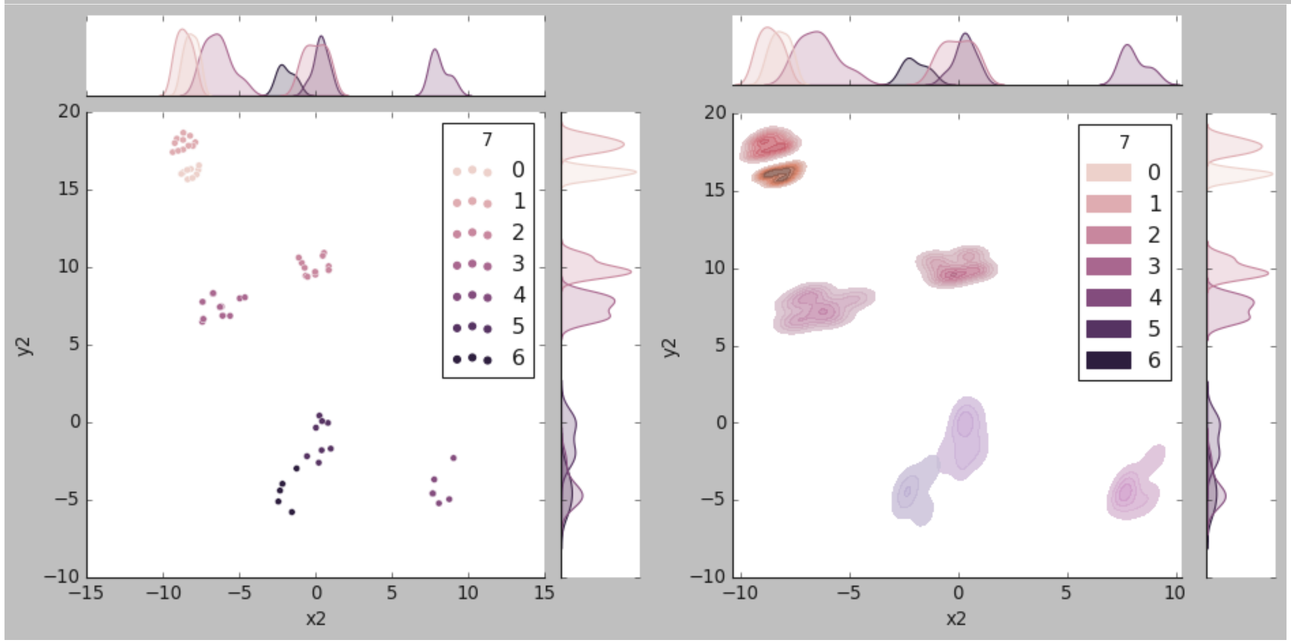
# # Cluster = 6


6
-1
0
1
2
3
4
5
x2
y2
Mutual reachability distance
λ value
Number of points
1
2
3
4
5


6
0
1
2
3
4
5
x2
y2

# # Cluster = 7


7
-1
0
1
2
3
4
5
6
x2
y2
Mutual reachability distance
λ value
Number of points


7
0
1
2
3
4
5
6
x2
y2