VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

MAGISTRO BAIGIAMASIS DARBAS

Ansamblio klasterizavimo metodų taikymas bendrojo kraujo tyrimo rezultatams, naudojant „Python“ biblioteką „OpenEnsembles“

Applying Ensemble Clustering Methods to a Complete Blood Count Test Results Using OpenEnsembles: A Python Resource for Ensemble Clustering

GINTARĖ PETRIŠIŪNAITĖ

Vilnius 2020

MATEMATIKOS IR INFORMATIKOS FAKULTETAS
STATISTINĖS ANALIZĖS KATEDRA

Darbo vadovas: Doc. dr. Viktor Skorniakov _____

Darbas apgintas

Registravimo NR. _____

Ansamblio klasterizavimo metodų taikymas bendrojo kraujo tyrimo rezultatams, naudojant „Python“ biblioteką „OpenEnsembles”

Santrauka

Siekiant įveikti atskirų klasterizavimo algoritmų trūkumus ir pagerinti klasterizavimo rezultatus yra taikomas ansamblių klasterizavimo metodas. Šis metodas sujungia skirtingų klasterizavimo algoritmų rezultatus ir gali pateikti pagerintą galutinį klasterizavimo rezultatą. Šiame darbe ansamblių klasterizavimas taikomas bendro kraujo tyrimo parametrams. Pašalinus išskirtis Tukey metodu parametrams buvo apskaičiuotos referencinės ribos, kurios naudojamos klasterizavimo rezultatų validacijai. Ansambliai buvo sukurti naudojant „Python“ biblioteką „OpenEnsembles“. Tankio funkcijas naudojantys klasterizavimo algoritmai tiksliausiai aptiko išskirčių klasterius. Geriausi klasterių ansambliai buvo sukurti naudojant „co-occurrence linkage“ ansamblio kūrimo būdą.

Raktiniai žodžiai : Ansamblio klasterizavimo metodai, Python, OpenEnsembles.

Applying Ensemble Clustering Methods to a Complete Blood Count Test Results Using OpenEnsembles: A Python Resource for Ensemble Clustering

Abstract

Clustering algorithms have their individual limitations. In order to overcome limitations of individual clustering algorithms and improve clustering results ensemble clustering method can be applied. Ensemble clustering method combines results of cluster partitions of different clustering algorithms and can produce the final clustering solution of an improved quality. This thesis focuses on applying ensemble clustering methods to a Complete Blood Count test parameters. Reference intervals were calculated and used for cluster validation after outlier removal, using Tukey fences. Ensembles were created using a Python library OpenEnsembles. Density based clustering algorithms were able to detect outlier clusters. Ensembles created using co-occurrence linkage ensemble creation method avoided over-merging or under-merging clusters unlike majority vote and graph closure ensemble creation methods.

Keywords : Ensemble clustering, OpenEnsembles, Python, Clustering algorithms.

Table of Contents

1. Introduction

No single clustering algorithm is capable of correctly identifying the underlying structure of all data sets. Therefore, different clustering algorithms applied to the same data set can produce distinctly different results. In order to overcome limitations of individual clustering algorithms and improve clustering results, ensemble clustering method can be applied. A cluster ensemble combines results of cluster partitions of different clustering algorithms and can produce final clustering solution of an improved quality.[18]

Clustering is used in a broad range of applications to analyze biological data, including anomaly detection. An anomaly (also called an outlier) can be described as an object that has a low affinity to all the clusters. This thesis focuses on cluster ensemble creation for anomaly detection in results of Complete Blood Count(CBC) test parameter. This Complete Blood Count (CBC) test panel measures parameters of red blood cells, white blood cells, hemoglobin, hematocrit and platelets (thrombocytes).[4]

Creating and analyzing ensembles is done using a Python resource OpenEnsembles. The OpenEnsembles library provides a unified interface for applying transformations to data, clustering data, visualizing individual clustering solutions, visualizing and creating the ensemble, calculating validation metrics for a clustering solution.[9] This library is dedicated to clustering ensembles and is built using many open source Python projects, which include: scikit-learn, Pandas, Matplotlib, NetworkX, and NumPy.[22]

The goal of this paper is to create cluster ensembles that are capable of recognizing outliers (anomalies) in the Complete Blood Count test cell parameter groups more accurately than individual clustering algorithms.

Objectives:

- To compare effects of distance metrics between objects on the solution space for DBSCAN, Spectral, AffinityPropagation, Agglomerative, K-Means clustering algorithms.
- To compare effects of linkages between objects (clusters) on the solution space for agglomerative clustering algorithm.
- To identify most effective algorithms for outlier detection evaluating results of individual clustering solutions of Affinity Propagation, Birch, DBSCAN, HDBSCAN, Gaussian Mixture, HDBSCAN, Mean shift, Agglomerative, K-means, and Spectral clustering algorithms.

- From selected most effective algorithms, to create cluster ensembles using majority voting, graph closure, and co-occurrence linkage ensemble creation methods.

Structure. The beginning of the thesis contains analysis and synthesis of previous research and theoretical material about clustering algorithms, their metrics and methods for ensemble creation. Further, in the methodological part, data description, hypotheses, ensemble design, and the chosen validation metrics are presented. The final part includes results presentation.

2. Available Clustering Algorithms

This chapter presents clustering algorithms that are available in the OpenEnsembles library. The list includes: K-means, Agglomerative, Spectral, Birch, GaussianMixture, HDBSCAN, DBSCAN, Affinity Propagation algorithms.[9] These algorithms have their advantages, and depending on a clustering goal, some may perform more accurately than others. For example, there are algorithms designed to perform better with large data sets, are memory and time efficient, other algorithms that are based on density estimation can identify clusters of various shapes and sizes.

2.1 DBSCAN and HDBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm creates clusters as continuous regions of high density. For each object (instance), the algorithm counts how many other objects are located within a tiny distance called epsilon from it. This region is named the object's epsilon neighborhood. If an object has at least minimum sample of objects in its epsilon neighborhood (including itself), then it is considered a core instance.[11] Core instance can also be defined as an object that is located in a dense region. All objects in the neighborhood of a core instance belong to the same cluster. This can include other core instances and, for this reason, a long sequence of neighboring core instances creates a single cluster. An object that is not a core instance and does not have one in its neighborhood is considered an anomaly (an outlier). If the clusters of objects are dense enough this algorithm performs well and is able to separate clusters based on their density regions.[17]

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) algorithm performs clustering over varying epsilon values and integrates the result to find a clustering that gives the best stability over epsilon. Therefore, the algorithm is able to find clusters of varying densities which is not possible using DBSCAN. Objects that do not have neighbors nearby are considered being noise and HDBSCAN labels them by giving such objects a label with value -1.[19]

2.2 Agglomerative Clustering

This algorithm builds a hierarchy of clusters from the bottom up. At each iteration Agglomerative clustering merges the closest pair of clusters (starting from individual objects). The closest pair of clusters is determined by a specified linkage- single, complete, average, or Wards. These linkages are described in detail in the 2.9 subsection about clustering metrics. Agglomerative clustering algorithm can capture clusters of various shapes, and it can be used with any pairwise distance metric. If an algorithm is applied to a very large data set, it is recommended to provide a connectivity matrix in order for the algorithm to perform better. Connectivity matrix is a sparse $m \times m$ matrix that indicates which pairs of objects are neighbors.[17]

2.3 BIRCH

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) algorithm is memory and time efficient and was created to be particularly used for clustering of very large datasets.[10],[21] It can be faster than other algorithms like K-Means, and produce similar results, as long as the number of features is not very large (up to 20). BIRCH incrementally clusters acquired data objects. It keeps a tree of cluster features (information about clusters) which is updated in an iterative fashion.[20] The tree structure that is built during training contains just enough information to rapidly assign each new object to a cluster, while not having to store all the instances in the tree. Therefore, this process enables to use limited memory, while handling large datasets.[17]

2.4 K-Means

The K-Means algorithm clusters data by trying to separate objects into a specified number of clusters that have equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares. Inertia is the sum of squared error for each cluster. The smaller the inertia, the closer together all the objects are and the denser the cluster is. Therefore, lower values are better and zero is optimal.[3]

The k-means algorithm divides a set of N objects X into K disjoint clusters C, each described by the mean of the cluster objects. The means are commonly called the cluster "centroids"; note that they are not, in general, points from X , although they live in the same space.

Minimization of the inertia (within-cluster sum-of-squares criterion) is expressed:

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2)$$

In conclusion, K-means clustering method is based on the idea that a center can represent a cluster. However, it performs poorly when the goal is to identify elongated clusters, or clusters with irregular shapes or sizes.[17]

2.5 Gaussian Mixture

Gaussian Mixture algorithm clusters data in a very similar way to K-means. Gaussian Mixture algorithm accounts for the variance. Therefore, it can detect even oblong clusters (unlike K-means).[7]

2.6 Mean-shift

Mean-Shift clustering seeks to discover clusters in a smooth density of objects. It is a centroid based algorithm, which works by updating candidates for centroids to be the mean of the points (objects) within a region. This algorithm begins by putting a circle on each object, then, for each circle, it calculates the mean of all the instances located within it, and it shifts the circle so that it is centered on the mean. It automatically sets the number of clusters. Next, it iterates this mean-shift step until all the circles stop moving.[17]

The centroid candidate is updated according to the following equation: $\mathrm{x}_i^{t+1} = m(x_i^t)$

Samples within a given distance around an object x_i , where m is the mean shift vector that is computed for each centroid using the following equation, effectively updating a centroid to be the mean of the samples within a neighborhood $N(x_i)$ of an element x_i, containing:

$$m(x_i) = \frac{\sum_{x_j \in N(x_i)} K(x_j - x_i) x_j}{\sum_{x_j \in N(x_i)} K(x_j - x_i)}$$

In the next stage, the candidates for centroids are filtered to eliminate near-duplicates and then a final set of centroids is formed. [8]

This algorithm has similar features to the DBSCAN and HDBSCAN algorithms because it relies on the local density estimation and is able to find any number of clusters of any shape.

2.7 Affinity Propagation

Affinity Propagation chooses the number of clusters based on the data provided. The algorithm uses a voting system, where objects vote for similar objects to be their representatives. When the algorithm converges, each object and its voters form a cluster. Affinity propagation can detect any number of clusters of different sizes and shapes, however, this algorithm has a computational complexity not suited for large datasets. [17]

2.8 Spectral Clustering

Spectral clustering is a technique which has emerged from graph theory, where the communities of nodes are identified in a graph based on the edges connecting them. The algorithm uses information from the eigenvalues (spectrum) of special matrices built from the graph or the data set. [12] Spectral clustering can capture complex cluster structures but it does not scale well to large number of objects, and it does not perform well when the clusters have significantly different sizes. The algorithm reduces dimensionality by taking a similarity matrix between the objects and creating a low-dimensional embedding from it. After that, it uses another clustering algorithm in this low-dimensional space (Scikit-Learn's implementation uses K-Means). [8]

2.9 Clustering Metrics

Algorithms can be grouped by the metrics they use to produce results. These metrics include a number of clusters K, distances between objects and linkages (distances between clusters).[2]

The following metrics before clustering can be specified for these algorithms.

- Linkages: Agglomerative.
- Distances: K-means, HDBSCAN, DBSCAN, Spectral, AffinityPropagation, Agglomerative, Mean-shift.
- K: K-means, agglomerative, spectral, Birch, GaussianMixture.

2.9.1 Linkages

Linkages metric is used in Agglomerative clustering algorithm and it specifies distances between clusters.

The following is the list of available linkages:

- Single linkage returns the minimum distance between the two points that belong to clusters A and B.[5]

$$d(A, B) = \min_{\overrightarrow{x} \in A, \overrightarrow{y} \in B} \| \overrightarrow{x} - \overrightarrow{y} \|$$

  Single linkage is fast, can perform well on non-globular data, and can handle quite complicated cluster shapes. However, it performs poorly in the presence of noise.

- Complete linkage calculates the maximum distance distance between the two points that belong to clusters A and B.[5]

$$d(A, B) = \max_{\overrightarrow{x} \in A, \overrightarrow{y} \in B} \| \overrightarrow{x} - \overrightarrow{y} \|$$

- Average linkage is determined, by calculating the average distance between all the possible pairs of two objects in A and B clusters.[5]

$$d(A,B) = \frac{1}{A}\frac{1}{B}\sum_{x\in A}\sum_{x\in B} d(x,y)$$

Average and complete linkage perform well on cleanly separated globe shaped (spherical) clusters. [13]

2.9.2 Distance metrics

Distances between objects are implemented by K-means, HDBSCAN, DBSCAN, Spectral, AffinityPropagation, Agglomerative, Mean-shift clustering algorithms.[8] Very often the measure of distance is the Euclidean metric which belongs to the Minkowski metric group. In addition, city-block(Manhattan), Euclidean and Chebyshev distances belong to the Minkowski metric group.[3]

OpenEnsembles library allows to choose from a range of distances. Most frequently used are:

- Minkowski distance is a metric on Euclidean space, and it is also considered as a generalisation of Euclidean distance, Manhattan distance and Chebyshev distance.

$$D_{ij} = \left(\sum_{m=1}^{M} |x_{im} - x_{jm}|^p\right)^{\frac{1}{p}}$$

- Euclidean distance is a straight-line distance between two points.

$$D_{ij} = \left(\sum_{m=1}^{M} |x_{im} - x_{jm}|^2\right)^{\frac{1}{2}}$$

- Chebyshev distance is a special case of Minkowski distance as $p \to \infty$.

$$D_{ij} = \max_{1\le m\le M} |x_{im} - x_{jm}|$$

- Manhattan (city-block) distance distance between two points measured along axes at right angles. [3]

$$D_{ij} = \sum_{m=1}^{M} |x_{im} - x_{jm}|$$

3. Ensemble Creation Methods

OpenEnsembles library allows creation of ensembles using majority voting, co-occurrence linkage and graph closure methods. For all of them the only parameter that must be specified is the threshold value. [1]

3.1 Majority voting, co-occurrence linkage and graph closure

Majority voting clusters together instances that have a high co-occurrence.

Graph closure uses co-occurrence matrices in order to create graphs, and clusters the data based on cliques. Co-occurrence matrices portray how frequently a pair of objects has been assigned to the same cluster.

Co-occurrence linkage implements a specific clustering algorithm, hierarchical (Agglomerative) clustering, by treating the co-occurrence matrix as a pairwise distance matrix. The clustering ends when there is no element on the matrix with a value that is larger then the threshold value.[16]

3.2 Cluster Validation Methods

The term cluster validation is used to design the procedure of evaluating the goodness of clustering algorithm results. Clustering validation statistics can be categorized into internal and external validation metrics. Internal cluster validation uses the internal information of the clustering process to evaluate the goodness of a clustering structure without reference to external information. External cluster validation has the results of a cluster analysis to an externally known result, such as externally provided cluster class labels. It measures the extent to which cluster labels match externally supplied class labels. Since we know the "true" cluster number in advance, this approach can be used for determining the most suitable clustering algorithm for a specific data set. [1]

4. Methodology

This chapter presents data, hypotheses, explanation of ensembles creation and formation of validation metrics that were used in evaluating clustering results.

4.1 Data description and preparation

The dataset was collected in Vilniaus Centro Poliklinika during the time period between June and December in 2017. In the dataset, there are results of venous and capillary prophylactic Complete Blood Count test parameters and the age of a patient in days at the time test was performed. The Complete Blood Count test measured parameters of red blood cells (erythrocytes, mean corpuscular volume of erythrocytes, erythrocyte partition width, hemoglobin, mean hemoglobin in erythrocyte and mean erythrocyte hemoglobin concentration), white blood cells (leukocytes, neutrophils, lymphocytes, monocytes, eosinophils and basophils), hematocrit, and platelets, which help with clotting. The test was performed on 3688 patients aged from 14 days to 18 years old. Test results were divided into groups according to the patients age. There were only 32 patients of age that is less than 2 months old patients, therefore this age group was excluded from ensemble creation because of not sufficient number of instances. Therefore, patients test results were divided into the following age groups: [2 months - 6 months), [6 months - 2 years), [2 years - 6 years), [6 years - 12 years), [12 years - 18 years). In addition, patients of [12 years - 18 years) age group were split into groups of men and women. The table shows the number of patients in each age group. The 4.1 table below shows the amount of patients in each age group. Ensembles were chosen to be created for white blood cells called granulocytes (neutrophils, eosinophils, basophils, and red blood cell parameters (hemoglobin, erythrocytes, and mean corpuscular volume of erythrocytes).

Table 4.1: Number of patients in each age group

Age group	Number of Patients
[2m.-6m.)	184
[6m.-2y.)	607
[2y.-6y.)	1322
[6y.-12y.)	904
[12y.-18y.) women	368
[12y.-18y.) men	271

4.2 Creation of the external validation metrics

This section describes cluster validation metrics creation. For clustering results validation reference interval were calculated after removing outliers. The calculation of outlier values and their removal was done using the Tukey Fences Method.[15] The Tukey method calculates outliers based on the interquartile range. For example, if the first quartile Q1 (0.25) and third quartile Q3 (0.75) are the lower and upper quartiles, respectively, then an outlier can be defined as a value outside the range:

$$Q_1 - k(Q_3 - Q_1), Q_3 + k(Q_3 - Q_1)$$

K is a positive constant. A k = 1.5 was used in calculations as suggested by John Tukey. (Using a large constant such as k = 3, only very significant outliers would be recognized). 4.2 table shows Tukey Fences values for granulocytes. Granulocytes had outliers only above upper Tukey fence, while for all granulocytes the lower fence was 0.

Table 4.2: Outlier values for granulocytes

	Eosinophils ($10^9/l$)		Neutrophils ($10^9/l$)		Basophils ($10^9/l$)	
Age group	Lower Tukey fence	Upper Tukey fence	Lower Tukey fence	Upper Tukey fence	Lower Tukey fence	Upper Tukey fence
[2m.-6m.)	0	0.795	0	4.4995	0	0.08
[6m.-2y.)	0	0.68	0	7.95	0	0.07
[2y.-6y.)	0	0.78	0	11.85575	0	0.095
[6y.-12y.)	0	0.835	0	9.81675	0	0.095
[12y.-18y.) women	0	0.5	0	7.8275	0	0.07
[12y.-18y.) men	0	0.63	0	6.5	0	0.07

The 4.3 table below shows values of lower and upper Tukey fences for red blood cell parameters.

Table 4.3: Outlier values for red blood cell parameters

	Hemoglobin (g/l)		Erythrocytes ($10^{12}/l$)		Mean Corpuscular Erythrocyte Cell Volume (fl)	
Age group	Lower Tukey fence	Upper Tukey fence	Lower Tukey fence	Upper Tukey fence	Lower Tukey fence	Upper Tukey fence
[2m.-6m.)	95.5	139.5	3.3205	5.5795	65.8425	85.1375
[6m.-2y.)	95	143	3.735	5.775	64.5	82.1
[2y.-6y.)	103.5	147.5	3.965	5.765	67.8	83
[6y.-12y.)	111.5	155.5	4.085	5.885	69.2625	87.7675
[12y.-18y.) women	110	158	3.86	5.7	73.6	95.2
[12y.-18y.) men	117.5	177.5	4.2725	6.2125	73.325	92.725

After the outleirs removal below the lower Tukey Fences and above the upper Tukey Fences the 0.95 reference interval for was calculated using nonparRI function from R package referenceIntervals. The clustering results are evaluated according to the reference values in the tables below.

For granulocytes all values only above 97.5% are considered anomalies because there were outliers present only above the upper Tukey fence (Table 4.4).

For red blood cell parameters values below 2.5% and above 97.5% are considered anomalies as there were outliers below the lower and above the upper Tukey fences (Table 4.5).

Table 4.4: Reference intervals for granulocytes

	Eosinophils ($10^9/l$)		Neutrophils ($10^9/l$)		Basophils ($10^9/l$)	
Age group	2.50%	97.50%	2.50%	97.50%	2.50%	97.50%
[2m.-6m.)	0.04525	0.71475	0.708	3.774	0.01	0.07
[6m.-2y.)	0	0.5865	0.753	7.333	0.01	0.06425
[2y.-6y.)	0	0.67	1.52	10.68522	0.01	0.08
[6y.-12y.)	0.01	0.7275	1.28275	8.807	0.01	0.08
[12y.-18y.) women	0.015	0.44	1.4735	7.0065	0.01	0.06
[12y.-18y.) men	0.02	0.57675	1.2975	5.70375	0.01	0.06

Table 4.5: Reference intervals for red blood cells parameters

	Hemoglobin (g/l)		Erythrocytes ($10^{12}/l$)		Mean Corpuscular Erythrocyte Cell Volume (fl)	
Age group	2.50%	97.50%	2.50%	97.50%	2.50%	97.50%
[2m.-6m.)	101	134	3.5775	5.29	68.9	81.8
[6m.-2y.)	103	135	4.11	5.479	66.4925	79.315
[2y.-6y.)	111	142	4.24675	5.54	69.2	80.9
[6y.-12y.)	118	150	4.36	5.67	71.34	85.3
[12y.-18y.) women	113	151	4.14025	5.4595	77.025	90.66
[12y.-18y.) men	127	168	4.556	5.89	75.08	89.4

4.3 Creation of cluster ensembles

Ensembles were chosen to be created for white blood cells called granulocytes (neutrophils, eosinophils, basophils) and red blood cell parameters (hemoglobin, erythrocytes, mean corpuscular volume of erythrocytes). Firstly, all individual clustering algorithms were tested on the CBC test parameter subgroups. After that, the algorithms that were able to detect outliers were used for creating clustering ensembles using majority voting, co-occurrence linkage and graph closure ensemble creation methods with different threshold values.

4.4 Hypotheses

- Density based algorithms such as HDBSCAN and Mean Shift will effectively identify outliers.
- Parametric centroid based clustering algorithms, such as K-Means, Gaussian Mixture will not be able to identify outliers.
- Cluster ensemble of selected algorithms will be more effective in anomaly detection than solutions of individual algorithms.
- Minkowski family distance metrics (Euclidean, Manhattan, Chebyshev) will have a similar effect on the solution space.

5. Results

This chapter presents the comparison of individual clustering algorithms, comparison of distance metrics on the solution space, and cluster ensemble results.

5.1 Comparison of individual clustering algorithms

Many clustering algorithms are unable to recognize unusual shapes of anomaly clusters. That is especially true with parametric clustering algorithms that have an implicit assumption about cluster shapes. For example, K-means and Gaussian Mixture clustering algorithms make an assumption that clusters form a shape of a Gaussian ball (sphere). As outlier clusters do not form clusters of spherical shapes such algorithms are not suitable for outlier detection. However, non-parametric density based clustering algorithms such as Mean-shift algorithm, DBSCAN, and HDBSCAN and allow to discover clusters of unusual shapes.[14] This implies that they are more suitable when clustering is used for anomaly detection. Therefore, density based clustering algorithms are used to form cluster ensembles for outlier detection. However, the DBSCAN algorithm preformed poorly as it merged all objects into one cluster and finding a suitable epsilon value for a dataset difficult. Therefore, HDBSCAN had an advantage because it is able to use different epsilon values when clustering. Birch, Agglomerative, and Affinity propagation algorithms also merged results into one cluster. Spectral clustering produced clusters of sphere shapes and therefore, is also not suitable for outlier detection.

5.2 Comparison of distance metrics effects on the solution space

Co-occurrence matrices portray the similarities of clustering solutions for the same algorithm with different distance metrics.

Figure 5.1 shows the distances effect on solution space when clustering red blood cell parameters using HDBSCAN, [2m.-6m.) age group. Euclidean and Minkowski distances produce exactly the same results. Therefore, in the ensemble creation will be used only Euclidean distance.[6]

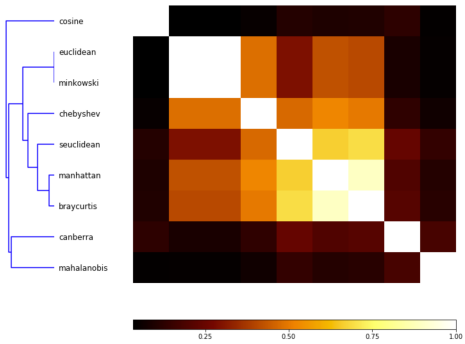

Figure 5.1: Co-occurrence matrix (Distances effect on solution space when clustering red blood cell parameters using HDBSCAN, [2m.-6m.) age group)

5.3 Ensemble results

For ensemble creation the best performing algorithms were chosen to be used. Mean-shift algorithm and HDBSCAN clustering algorithm solutions with Euclidean, Manhattan and Chebyshev distance metrics were combined to form ensembles. Mean-shift algorithm distinguished less outliers when compared with HDBSCAN. HDBSCAN solutions with Manhattan distances distinguished the higher amount of instances as outliers, compared to HDBSCAN solutions that used Euclidean or Chebyshev distances. Ensemble creation was most accurate when performed using co-occurrence linkage ensemble creation method. Majority vote and Graph Closure was consistently under-merging clusters of outliers such that there was one major cluster and many small clusters that were unmerged outliers. Choosing a suitable threshold value was done by evaluating whether the clusters were under-merged or over-merged. Ensemble results were best produced with co-occurrence linkage and threshold values from 0.5 to 0.9. Frequently used threshold values were 0.6, 0.7, 0.9. However, in some cases 0.9 threshold value over-merged results into a one large cluster. Therefore, in those cases lower threshold values (0.5, 0.6, 0.7) produced better results by not over-merging clusters.

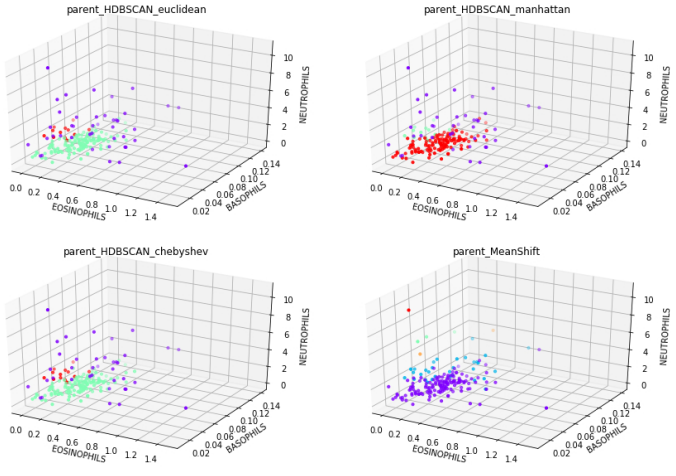

Figure 5.2: Granulocytes clustering results, [2m.-6m.) age group

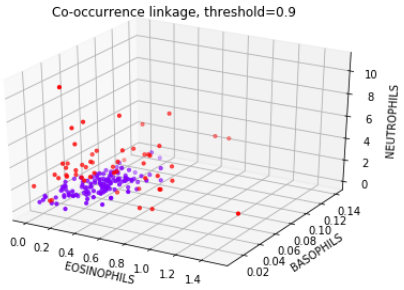

Figure 5.3: Co-occurrence linkage ensemble

Figure 5.2 shows results of individual clustering algorithms of granulocytes in [2m.-6m.) age group. The outlier values of HDBSCAN algorithm solutions were assigned value -1 and are portrayed in purple color. Meanwhile, outliers from Mean-shift algorithm cluster were assigned value 1, portrayed in light blue color (Figure 5.2). Individual clustering algorithms of HDBSCAN with Euclidean distance, Manhattan distance, Chebyshev distance, and Mean-shift algorithm distinguished 42, 41, 38, 32 instances as outliers, respectively. According to validation metrics (reference values) this age group has 27 outliers, out of which 22 (81.48%) were identified by the co-occurence linkage ensemble with threshold value 0.9. In total, the co-occurence linkage ensemble (Figure 5.3) distinguished 54 instances as outliers which are portrayed in red color.

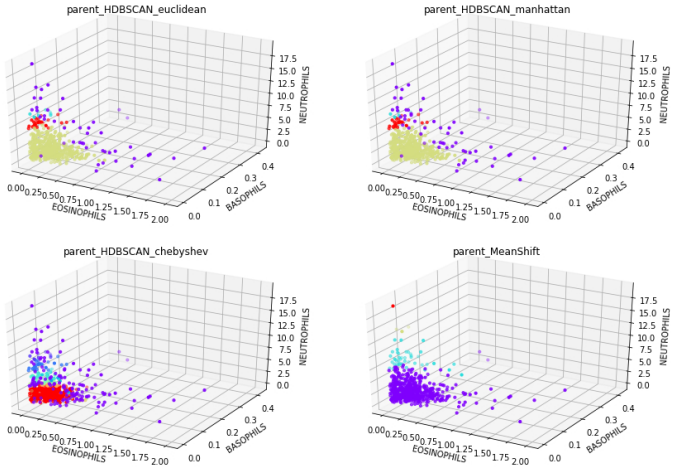

Figure 5.4: Granulocytes clustering results, [6m.-2y.) age group

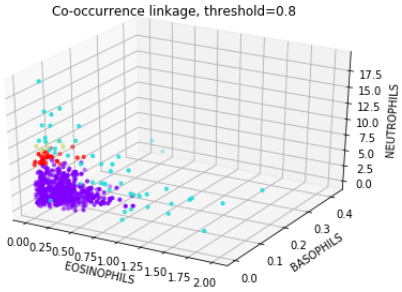

Figure 5.5: Co-occurrence linkage ensemble

Figure 5.4 shows results of individual clustering algorithms of granulocytes in [6m.-2y.) age group. The outlier values of HDBSCAN algorithm solutions were assigned value -1 and are portrayed in purple color (Figure 5.4). Meanwhile, outliers from Mean-shift algorithm cluster were assigned value 1, portrayed in light blue color (Figure 5.4). Individual clustering algorithms of HDBSCAN with Euclidean distance, Manhattan distance, Chebyshev distance, and Mean-shift algorithm distinguished 49, 58, 139, 46 instances as outliers, respectively. According to validation metrics (reference values) this age group has 123 outliers, out of which 47 (38.21%) were identified by the co-occurence linkage ensemble with threshold value 0.8. In total, the ensemble (Figure 5.5) distinguished 49 instances as outliers which are portrayed in light blue color.

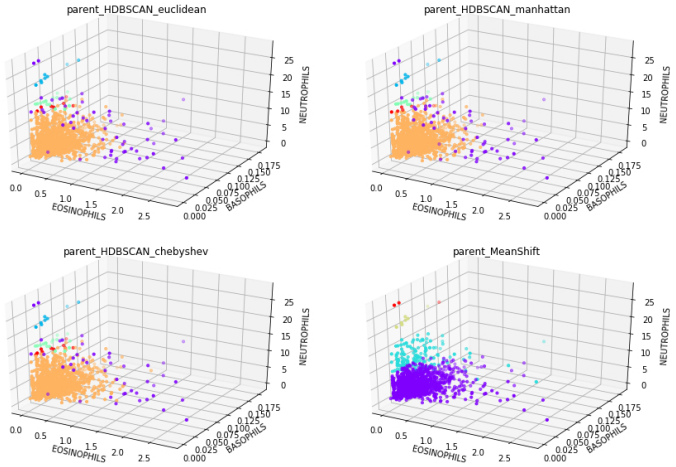

Figure 5.6: Granulocytes clustering results, [2y.-6y.) age group

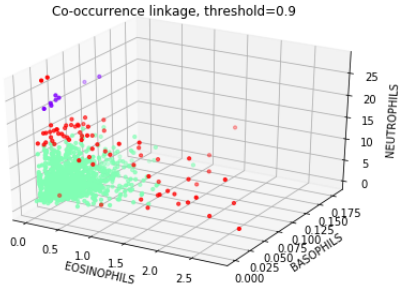

Figure 5.7: Co-occurrence linkage ensemble

The outlier values of HDBSCAN algorithm solutions were assigned value -1 and are portrayed in purple color (Figure 5.6). Meanwhile, outliers from Mean-shift algorithm cluster were assigned value 1, portrayed in light blue color (Figure 5.6). Individual clustering algorithms of HDBSCAN with Euclidean distance, Manhattan distance, Chebyshev distance, and Mean-shift algorithm distinguished 55, 61, 45, 158 instances as outliers, respectively. According to validation metrics (reference values) this age group has 205 outliers, out of which 70 (34.14%) were identified by the co-occurence linkage ensemble with threshold value 0.9. In total, the ensemble (Figure 5.7) distinguished 73 instances as outliers which are portrayed in red color.

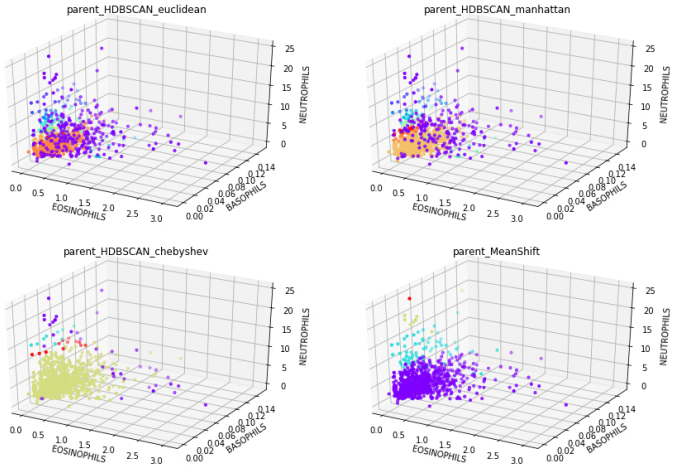

Figure 5.8: Granulocytes clustering results, [6y.-12y.) age group

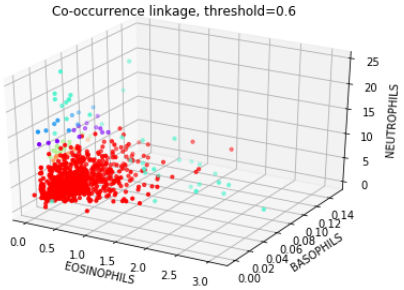

Figure 5.9: Co-occurrence linkage ensemble

The outlier values of HDBSCAN algorithm solutions were assigned value -1 and are portrayed in purple color (Figure 5.8). Meanwhile, outliers from Mean-shift algorithm cluster were assigned value 1, portrayed in light blue color (Figure 5.8). Individual clustering algorithms of HDBSCAN with Euclidean distance, Manhattan distance, Chebyshev distance, and Mean-shift algorithm distinguished 268, 208, 38, 72 instances as outliers, respectively. According to validation metrics (reference values) this age group has 160 outliers, out of which 32 (20%) were identified by the co-occurence linkage ensemble with threshold value 0.6. In total, the ensemble (Figure 5.9) distinguished 37 instances as outliers which are portrayed in light blue color.

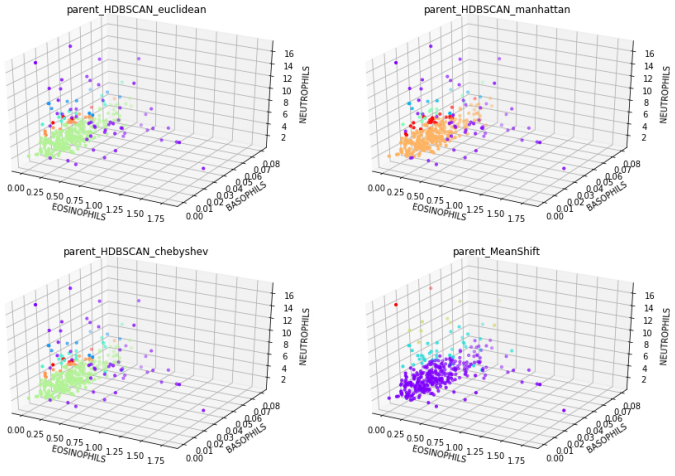

Figure 5.10: Granulocytes clustering results, women [12y.-18y.) age group

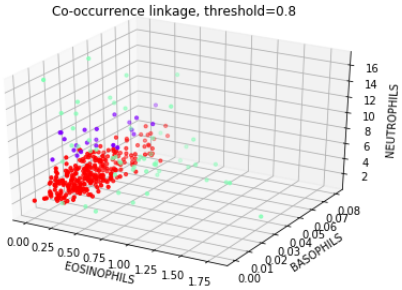

Figure 5.11: Co-occurrence linkage ensemble

The outlier values of HDBSCAN algorithm solutions are assigned value -1 and are portrayed in purple color (Figure 5.10). Individual clustering algorithms of HDBSCAN with Euclidean distance, Manhattan distance, Chebyshev distance, and Mean-shift algorithm distinguished 33, 79, 32, 29 instances as outliers, respectively. According to validation metrics (reference values) this age group has 72 outliers, out of which 45 (62.5%) were identified by the co-occurence linkage ensemble with threshold value 0.8. In total, the co-occurrence linkage ensemble (Figure 5.11) distinguished 55 instances as outliers which are portrayed in light green color.

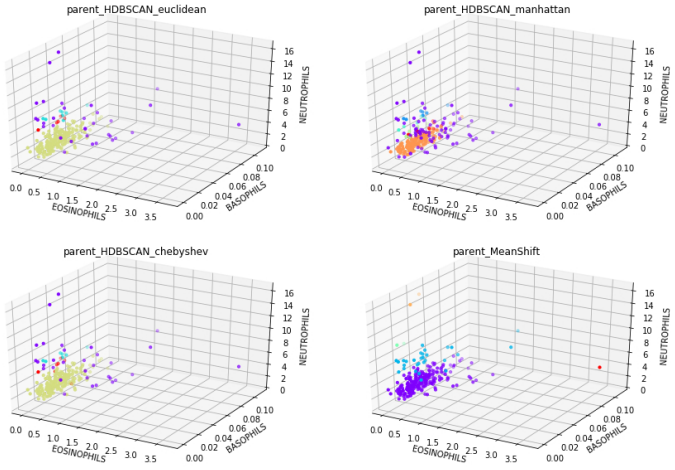

Figure 5.12: Granulocytes clustering results, men [12y.-18y.) age group

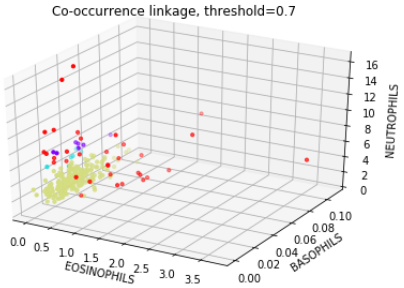

Figure 5.13: Co-occurrence linkage ensemble

The outlier values of HDBSCAN algorithm solutions are assigned value -1 and are portrayed in purple color (Figure 5.12). Individual clustering algorithms of HDBSCAN with Euclidean distance, Manhattan distance, Chebyshev distance, and Mean-shift algorithm distinguished 55, 57, 50, 40 instances as outliers, respectively. According to validation metrics (reference values) this age group has 49 outliers, out of which 28 (57.14%) were identified by the co-occurence linkage ensemble with threshold value 0.7. In total, the co-occurrence linkage ensemble (Figure 5.13) distinguished 34 instances as outliers which are portrayed in red color.

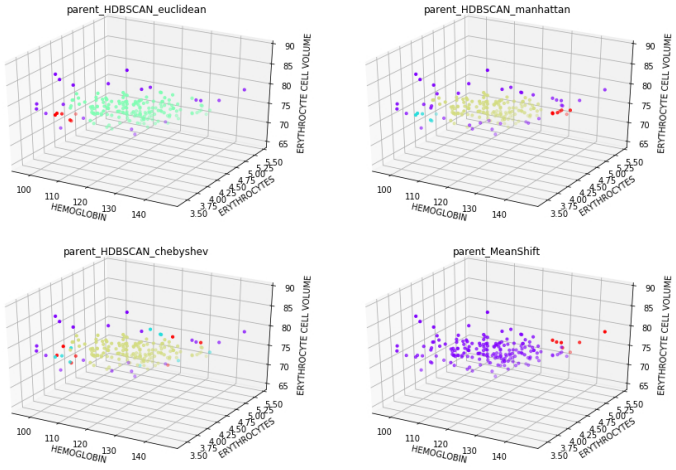

Figure 5.14: Parameters of red blood cells clustering results, [2m.-6m.) age group

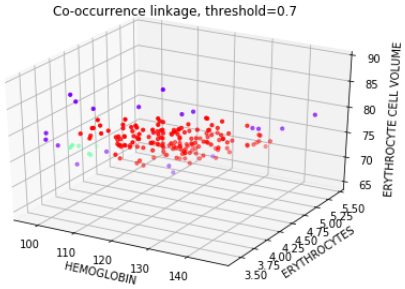

Figure 5.15: Co-occurrence linkage ensemble

The outlier values from HDBSCAN algorithm solutions are assigned value -1 and are illustrated in purple color (Figure 5.14). Mean-shift algorithm assigned value 1 to outliers and portrayed in color red. Individual clustering algorithms of HDBSCAN with Euclidean distance, Manhattan distance, Chebyshev distance, and Mean-shift algorithm distinguished 20, 37, 20, 7 instances as outliers, respectively. According to validation metrics (reference values) this age group has 20 outliers, out of which 16 (80%) were identified by the co-occurence linkage ensemble with threshold value 0.7. In total, the ensemble (Figure 5.15) distinguished 19 instances as outliers.

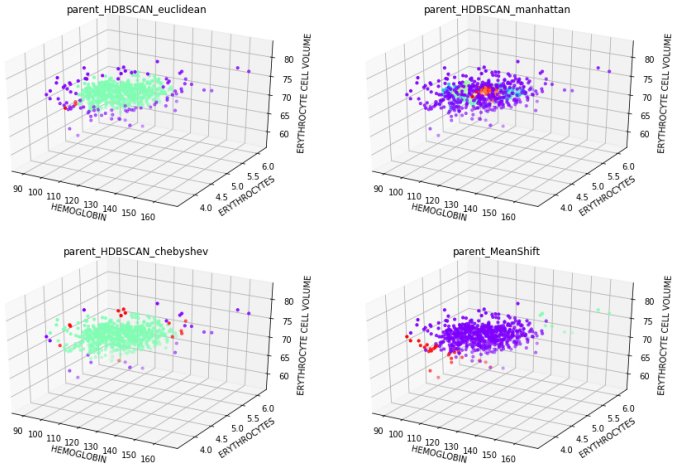

Figure 5.16: Parameters of red blood cells clustering results, [6m.-2y.) age group

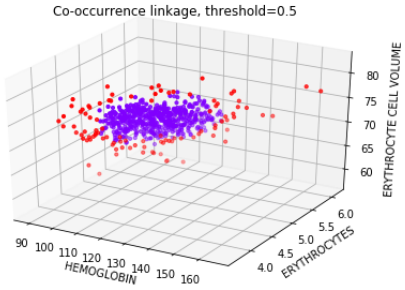

Figure 5.17: Co-occurrence linkage ensemble

The outlier values from HDBSCAN algorithm solutions are assigned value -1 and are illustrated in purple color (Figure 5.16). Individual clustering algorithms of HDBSCAN with Euclidean distance, Manhattan distance, Chebyshev distance, and Mean-shift algorithm distinguished 93, 353, 23, 8 instances as outliers, respectively. According to validation metrics (reference values) this age group has 72 outliers, out of which 67 (93.05%) were identified by the co-occurence linkage ensemble with threshold value 0.5. In total, the ensemble (Figure 5.17) distinguished 98 instances as outliers and portrayed then in red color.

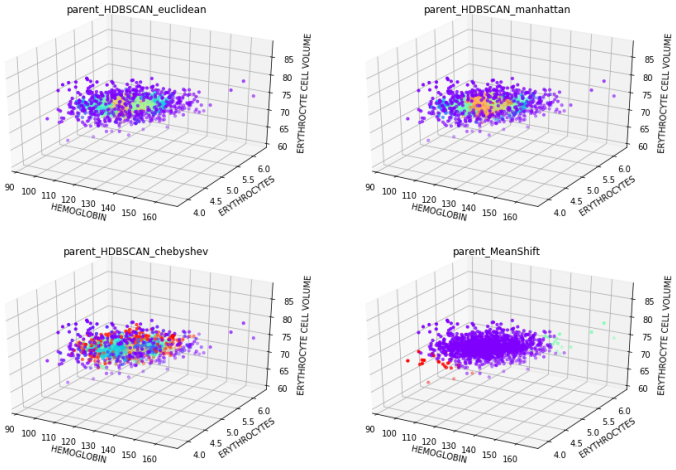

Figure 5.18: Parameters of red blood cells clustering results, [2y.-6y.) age group

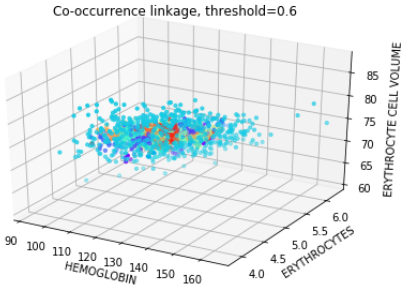

Figure 5.19: Co-occurrence linkage ensemble

The outlier values from HDBSCAN algorithm solutions are assigned value -1 and are illustrated in purple color (Figure 5.18). Individual clustering algorithms of HDBSCAN with Euclidean distance, Manhattan distance, Chebyshev distance, and Mean-shift algorithm distinguished 445, 418, 228, 16 instances as outliers, respectively. According to validation metrics (reference values) this age group has 189 outliers, out of which 167 (88.35%) were identified by the co-occurence linkage ensemble with threshold value 0.6. In total, the ensemble (Figure 5.19) distinguished 461 instances as outliers in light blue color.

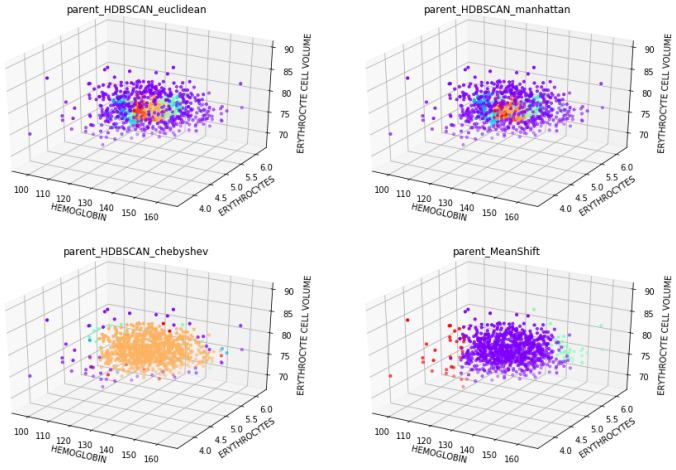

Figure 5.20: Parameters of red blood cells clustering results, [6y.-12y.) age group

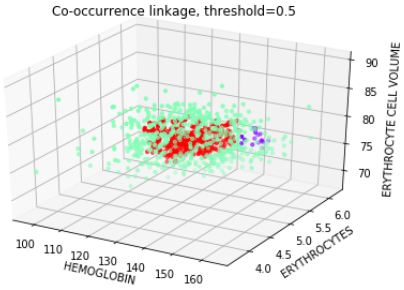

Figure 5.21: Co-occurrence linkage ensemble

Individual clustering algorithms of HDBSCAN with Euclidean distance, Manhattan distance, Chebyshev distance, and Mean-shift algorithm distinguished 457, 480, 48, 37 instances as outliers, respectively. According to validation metrics (reference values) this age group has 111 outliers, out of which 109 (98.19%) were identified by the co-occurrence linkage ensemble with threshold value 0.5. In total, the ensemble (Figure 5.21) distinguished 515 instances as outliers and portrayed them in light green color.

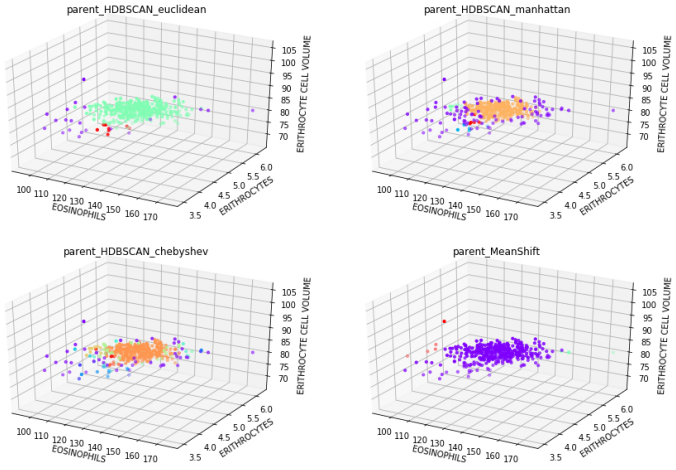

Figure 5.22: Parameters of red blood cells clustering results, women [12y.-18y.) age group

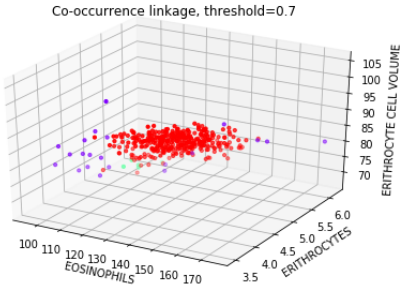

Figure 5.23: Co-occurrence linkage ensemble

Individual clustering algorithms of HDBSCAN with Euclidean distance, Manhattan distance, Chebyshev distance, and Mean-shift algorithm distinguished 83, 152, 19, 8 instances as outliers, respectively. According to validation metrics (reference values) this age group has 56 outliers, out of which 20 (35.71%) were identified by the co-occurrence linkage ensemble with threshold value 0.7. In total, the ensemble (Figure 5.23) distinguished 34 instances as outliers and portrayed them in purple color.

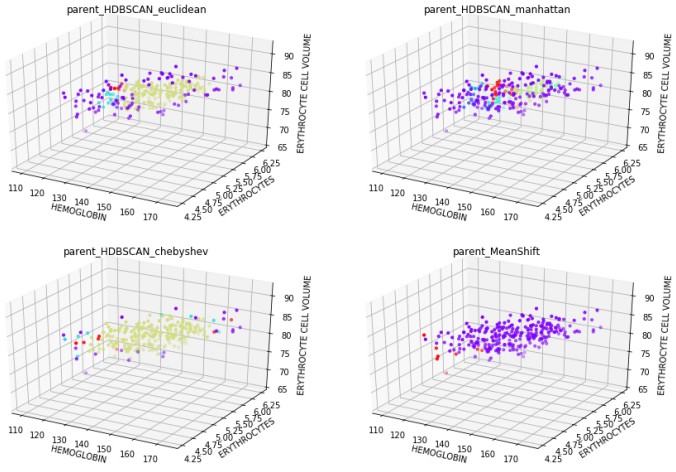

Figure 5.24: Parameters of red blood cells clustering results, men [12y.-18y.) age group

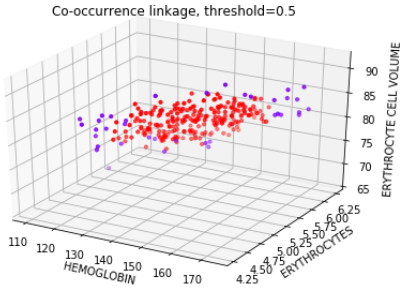

Figure 5.25: Co-occurrence linkage ensemble

Individual clustering algorithms of HDBSCAN with Euclidean distance, Manhattan distance, Chebyshev distance, and Mean-shift algorithm distinguished 22, 98, 43, 3 instances as outliers, respectively. According to validation metrics (reference values) this age group has 24 outliers, out of which 19 (79.16%) were identified by the co-occurrence linkage ensemble with threshold value 0.5. In total, the ensemble (Figure 5.25) distinguished 21 instances as outliers and portrayed them in purple color.

6. Conclusions

Outlier clusters often have unusual shapes. Therefore, parametric clustering algorithms such as K-means, Gaussian Mixture models perform poorly in outlier detection because they cluster objects into spherical shapes, or oblong shapes (Gaussian Mixture algorithm). Non parametric density based algorithms preform significantly better in outlier detection, as they recognize unusual cluster shapes and sizes. These algorithms include HDBSCAN and Mean-shift.

HDBSCAN algorithm solutions with Manhattan distance in most cases identified too many objects as outliers, while Mean-shift in majority of cases identified the least amount of outliers. However, HDBSCAN algorithm solutions with Manhattan distance identified outliers that algorithms like HDBSCAN with Euclidean metrics were not able to. HDBSCAN solutions with Euclidean and Chebyshev distances in several solutions detected similar number of objects as outliers. However, HDBSCAN solutions with Euclidean distance usually identified more than HDBSCAN with Chebyshev.

Ensembles created using majority voting and graph closure ensemble creation techniques, with different threshold values, were under-merging outlier clusters. The outliers were unmerged and separated into many clusters. Co-occurrence linkage ensemble creation method was more successful in merging outlier clusters and produced best results with threshold values ranging from 0.5 to 0.9.

References

[1] Cluster validation statistics: Must know methods. https://www.datanovia.com/en/lessons/cluster-validation-statistics-must-know-methods.

[2] Clustering algorithms openensembles. https://naeglelab.github.io/OpenEnsembles/clustering_algorithms.html.

[3] Clustering: Why to use it. https://towardsdatascience.com/clustering-why-to-use-it-16d8e2fbafe.

[4] Complete blood count (cbc) test parameters. https://www.uofmhealth.org/health-library/hw4260.

[5] Distances between clustering, hierarchical clustering 36-350, data mining,p.4.

[6] Documentation openensembles co-occurrence matrix. https://naeglelab.github.io/OpenEnsembles/Examples/Demonstrate_Distance_Affinity_effects.html.

[7] Gaussian mixture models. https://towardsdatascience.com/gaussian-mixture-models-d13a5e915c8e.

[8] K-means documentation from scikit-learn. https://scikit-learn.org/stable/modules/clustering.html#k-means.

[9] Openensembles documentation. https://naeglelab.github.io/OpenEnsembles/index.html.

[10] sklearn birch documentation. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.Birch.html.

[11] sklearn.cluster.dbscan documentation, leland mcinnes, john healy, and steve astels. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html.

[12] W.fleshman-spectral clustering foundation and application. https://towardsdatascience.com/spectral-clustering-aba2640c0d5b.

[13] Distances between clustering, hierarchical clustering. Data Mining, pages 36–350, 2009.

[14] R. J. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In Pacific-Asia conference on knowledge discovery and data mining, pages 160–172. Springer, 2013.

[15] J. W. Foreman. Data smart: Using data science to transform information into insight. pages 337–341, 390–393, 2013.

[16] K. G. M. George Kyriakides. Hands-On Ensemble Learning with Python, 151-170. 2019.

[17] A. Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 237-274. 2019.

[18] R. J. A. K. J. M. M. Jinfeng Yi, Tianbao Yang. Robust ensemble clustering by matrix completion. Data Mining (ICDM), IEEE International Conference, 2013.

[19] L. McInnes, J. Healy, and S. Astels. hdbscan: Hierarchical density based clustering. J. Open Source Software, 2(11):205, 2017.

[20] B. N. O.Nasraoui. Clustering methods for big data analytics. page 119, 2008.

[21] M. L. Tian Zhang, Raghu Ramakrishnan. Birch: An efficient data clustering method for very large databases. pages 103–114, 1996.

[22] Z. R. K. P. S. T.Ronan, S.Anastasio. Openensembles: A python resource for ensemble clustering. Journal of Machine Learning Research, 18:1–6, 2018.

APPENDICES

Ensemble creation example code

In [2]:

```
import scipy as sp
import numpy as np
import pandas as pd
import openensembles as oe
import matplotlib.pyplot as plt
import openensembles as oe
data = pd.read_excel (r'C:/Users/G/Desktop/DUOMENYS/dataf.xlsx')#load data
pirma = data.query('amzius_dienu_sk>=60')
pirma = pirma.query('amzius_dienu_sk<=181')# selecting an age group
```

In [3]:

```
pirma1=pd.DataFrame(pirma[['HGB','RBC','MCV']])
d = oe.data(pirma1,[1,2,3]) #creating a "parent" data object
c  = oe.cluster(d) #instantiate an object so we can get all available algorithms
a = c.algorithms_available()
paramsC = c.clustering_algorithm_parameters()
takesLinkages = paramsC['linkage']
takesDistances = paramsC['distance']
takesK = paramsC['K']
K = range(1,3,1)
linkages = ['single']
distances = ['euclidean','manhattan','chebyshev']
#removing of unwanted algorithms:
algorithmsToRemove =
['Birch','kmeans','spectral','AffinityPropagation','agglomerative','DBSCAN','GaussianMixture']
for algToRemove in algorithmsToRemove:
    del a[algToRemove]
takesLinkages = paramsC['linkage']
takesDistances = paramsC['distance']

# individual clustering algorithms calculations:
c = oe.cluster(d)
for data_source in d.D.keys():
    for algorithm in list(a.keys()):
        if algorithm in takesK:
            for k in K:
                if algorithm in takesDistances:
                    if algorithm in takesLinkages:
                        for linkage in linkages:
                            if linkage == 'ward':
                                out_name = '_'.join([data_source, algorithm, linkage, str(k)])
                                c.cluster(data_source, algorithm, out_name, K=k, Require_Unique= Tr
e, linkage=linkage)
                            else:
                                for dist in distances:
                                    out_name = '_'.join([data_source, algorithm, dist, linkage, str
k)])
                                    c.cluster(data_source, algorithm, out_name, K=k, Require_Unique
True, linkage=linkage, distance=dist)



                    else:
                        for dist in distances:
                            out_name = '_'.join([data_source, algorithm, dist, str(k)])
                            c.cluster(data_source, algorithm, out_name, K=k, Require_Unique= True, d
istance=dist)
                else:
                    out_name = '_'.join([data_source, algorithm, str(k)])
                    c.cluster(data_source, algorithm, out_name, K=k, Require_Unique= True)


        else: # does not take K
            if algorithm in takesDistances:
                for dist in distances:
                    out_name = '_'.join([data_source, algorithm, dist])
                    c.cluster(data_source, algorithm, out_name, Require_Unique= True, distance=
ist)
            else:
                out_name = '_'.join([data_source, algorithm])
```

```
                c.cluster(data_source, algorithm, out_name, Require_Unique= True)
d=pd.DataFrame(c.labels)#creates a dataframe with clustering results
```

In [5]:

```
# CREATION OF AN ENSEMBLE USING MAJORITY VOTING, GRAPH CLOSURE, AND CO-OCCURRENCE LINKAGE
thresholds = [0.5, 0.7] #select threshold values
fig = 0
c_graph_dict = {}
c_majority_vote_dict = {}
c_co_occ_linkage_dict  = {}
for threshold in thresholds:
    c_graph = c.finish_graph_closure(threshold=threshold)
    c_graph_dict[str(threshold)] = c_graph
    c_majority_vote = c.finish_majority_vote(threshold=threshold)
    c_majority_vote_dict[str(threshold)] = c_majority_vote
    c_co_occ_linkage = c.finish_co_occ_linkage(threshold=threshold)
    c_co_occ_linkage_dict[str(threshold)] = c_co_occ_linkage
```