Università degli Studi di Milano Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea in Informatica

Studio e approfondimento del coefficiente Silhouette per la valutazione interna di risultati di clustering non-supervisionato

Relatore: Dott. Davide Chicco
Co-relatore: Prof.ssa Francesca Gasparini

Relazione della prova finale di:
Federico Salvioni
Matricola 845029

Anno Accademico 2023-2024

Indice

Elenco delle figure

Abstract

Silhouette è una metrica spesso utilizzata per individuare la combinazione di iperparametri di un algoritmo di clustering non supervisionato che risulti nel clustering più vicino possibile alla vera struttura di cluster dei dati in analisi. Verrà introdotta Silhouette e come calcolarla, alcune proprietà matematiche ed alcune applicazioni concrete su dataset di EHR (*Electronic Health Records*). Verranno inoltre analizzati alcuni pacchetti per il linguaggio `R` che implementano Silhouette per determinare quale sia il migliore, comparandone le performance sia fra di loro sia rispetto all'implementazione di `scikit-learn` (`Python`).

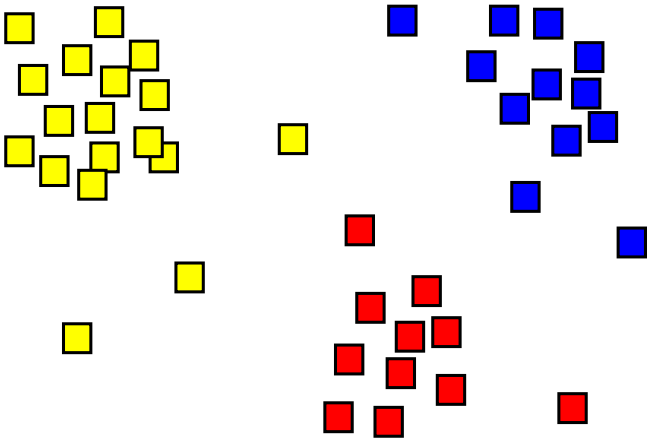
1. Introduzione

Con il termine **cluster analysis**, o semplicemente **clustering**, si intende *"una tecnica di classificazione statistica atta a scoprire se gli individui di una popolazione ricadono all'interno di certi gruppi mediante comparazioni quantitative di loro molteplici caratteristiche"* [1]. Informalmente, il clustering consiste nel (tentare di) suddividere un insieme di dati in gruppi, chiamati appunto **cluster**, di modo che ciascun gruppo contenga elementi simili fra di loro ma dissimili rispetto a quelli degli altri gruppi.

La definizione del concetto di "somiglianza" sta nelle mani di chi compie il clustering. In genere, l'analisi dei dati si occupa di dati numerici (altezze, lunghezze, età, ampiezze), pertanto il modo più naturale per formalizzare la somiglianza è dato da una **funzione di distanza**. Matematicamente, una qualsiasi funzione $d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ è considerabile una funzione di distanza se possiede (almeno) le seguenti proprietà:

- Per ogni $x \in \mathbb{R}$, $d(x,x) = 0$;
- Per ogni $x, y \in \mathbb{R}$, se $x \neq y$ allora $d(x,y) > 0$;
- Per ogni $x, y \in \mathbb{R}$, $d(x,y) = d(y,x)$ (**simmetria**);
- Per ogni $x, y, z \in \mathbb{R}$, $d(x,z) \leq d(x,y) + d(y,z)$ (**disuguaglianza triangolare**).

Un **algoritmo di clustering** non è altro che un algoritmo che abbia in input un insieme di dati e che abbia in output il risultato del clustering. Algoritmi di questo tipo fanno uso di una funzione di distanza per raggruppare gli elementi in cluster senza che sia necessario farlo manualmente.



Gli algoritmi di clustering si dividono in due macrocategorie: **algoritmi supervisionati** e **algoritmi non supervisionati**. Nei primi, viene usato un insieme di dati per generalizzarne le proprietà, cercando di predire il risultato per dati futuri, mentre nei secondi si cerca di catturare le proprietà dei dati in sé e per sé.

Una volta applicato un algoritmo di clustering su un certo insieme di dati, è ragionevole chiedersi se il risultato del clustering effettivamente rispecchi la struttura dei dati o se l'algoritmo abbia errato. Negli algoritmi di tipo supervisionato questo è semplice, perché è possibile comparare i dati ottenuti con il risultato atteso, similmente a come viene fatto in un modello di regressione.

Negli algoritmi di tipo non supervisionato questo è molto più difficile, perché non c'è differenza fra l'insieme di dati utilizzato per costruire il modello e l'insieme di dati usato per testare il modello: sono lo stesso insieme. Una possibile soluzione al problema è data da delle statistiche, il cui valore

[1] `https://www.merriam-webster.com/dictionary/cluster%20analysis`

rappresenta l'accuratezza del modello. Alcuni esempi sono: **Calinksi-Harabasz** [2], **Davies-Bouldin** [6] e **Dunn Index** [8].

La statistica di interesse per questa tesi è Silhouette [19]. Tale statistica, nelle parole dell'autore, si propone di rispondere alle seguenti domande:

- Il clustering è di buona qualità? In altre parole, gli elementi di uno stesso cluster sono fra di loro "vicini" ed al contempo "lontani" dagli elementi di tutti gli altri cluster?
- Quali sono gli elementi ben classificati, ovvero quelli che probabilmente si trovano nel cluster "giusto"?
- Quali sono gli elementi che è difficile stabilire con certezza in quali cluster vadano collocati, ovvero quelli che stanno "nel mezzo" fra più cluster?
- Il numero di cluster scelto è effettivamente rappresentativo del dataset o è 'artificioso'?

Identifying heterogeneous subgroups of systemic autoimmune diseases by applying a joint dimension reduction and clustering approach to immunomarkers [4]					
Patologie	**N. pazienti**	**Features**	**Algoritmi**	**Software**	**Metriche**
Lupus Eritematoso Sistemico, Artrite Reumatoide, Sindrome di Sjogren	11923	biomarcatori (tradotti in variabili categoriali)	Multiple Correspondence Analysis K-means (clustering e dimensionality reduction insieme)	clustrd (R)	Silhouette
Association of comorbid-socioeconomic clusters with mortality in late onset epilepsy derived through unsupervised machine learning [15]					
Patologie	**N. pazienti**	**Features**	**Algoritmi**	**Software**	**Metriche**
Epilessia (tardiva)	11307	Fattori di rischio, comorbiditá (indice di Charlson)	Agglomerative Hierarchical Clustering	scikit-learn (Python), Python 3.6.3, Stata 16.1	Silhouette, Davies-Bouldin
Exploration of critical care data by using unsupervised machine learning [14]					
Patologie	**N. pazienti**	**Features**	**Algoritmi**	**Software**	**Metriche**
	1503	Valori di test di routine di laboratorio (BUN, creatinina, glucosio, ...)	Kmeans	R 3.5.2	Total within-cluster variation, Silhouette, Gap statistic
Identifying and evaluating clinical subtypes of Alzheimer's disease in care electronic health records using unsupervised machine learning [1]					
Patologie	**N. pazienti**	**Features**	**Algoritmi**	**Software**	**Metriche**
Malattia di Alzheimer	10065	sintomi tipici, comorbiditá, dati demografici	K-Means, Kernel K-means, Affinity Propagation, Latent Class Analysis		Silhouette, Coefficiente di Jaccard
Utilization of Deep Learning for Subphenotype Identification in Sepsis-Associated Acute Kidney Injury [5]					
Patologie	**N. pazienti**	**Features**	**Algoritmi**	**Software**	**Metriche**
Sepsi	4001	Segni vitali, test di laboratorio	K-Means	scikit-learn (Python), matplotlib (Python), SAS 9.4, R 3.4.3	Silhouette, Davies-Bouldin, Calinksi-Harabasz

Tabella 1.1: Riassunto degli articoli scientifici che applicano Silhouette su EHR

Silhouette Index as clustering evaluation tool [7]				
Sintesi	**Algoritmi**	**Dataset**	**Software**	**Metriche**
È possibile utilizzare Silhouette come ”stopping rule” direttamente all'interno di un algoritmo di clustering?	RPCT	Dataset artificiale, generato mediante `cluster.Gen`, FCPS Dataset artificiale costituito da tre distribuzioni uniformi	clusterSim (R)	Adjusted Rand Index
Performance evaluation of the Silhouette index [22]				
Sintesi	**Algoritmi**	**Dataset**	**Software**	**Metriche**
Qual'è il modo corretto per calcolare la Silhouette media?	Complete-linkage, single-linkage	Quattro dataset artificiali con una struttura di cluster ben definita, `Iris`, `Wine`		
A comparison of clustering quality indices using outliers and noise [12]				
Sintesi	**Algoritmi**	**Dataset**	**Software**	**Metriche**
Quali sono le prestazioni di Silhouette rispetto ad altre metriche?	K-means, model-based clustering, hierarchical clustering, algoritmo ”random”	`clear` (struttura di cluster è ben definita), `out` (parte dei dati sono noise), `noi` (parte delle dimensioni sono noise)		Calinski-Harabasz, C-index, Davies-Bouldin, Gamma, ARI
Condensed Silhouette: An Optimized Filtering Process for Cluster Selection in K-Means [17]				
Sintesi	**Algoritmi**	**Dataset**	**Software**	**Metriche**
Nel caso di algoritmi di clustering partizionale, è possibile semplificare ulteriormente Silhouette?	K-Means	7 microarray gene expressions (migliaia di features), 7 dataset reali famosi (come Iris)	Scikit-Learn (Python)	
CUBOS: An Internal Cluster Validity Index for Categorical Data [11]				
Sintesi	**Algoritmi**	**Dataset**	**Software**	**Metriche**
È possibile estendere Silhouette per utilizzarla su valori discreti?	IDC	K-Modes	UCI	

Tabella 1.2: Riassunto degli articoli scientifici che studiano Silhouette da un punto di vista teorico

2. Metodi e strumenti utilizzati

2.1 Dataset

Per quanto riguarda l'applicare algoritmi di clustering a dataset reali, come dataset ho utilizzato delle cartelle cliniche elettroniche (EHR). Questi dataset, in formato `.csv` (*comma separated value*), sono liberamente utilizzabili. Per comodità, ho indicato tali dataset con la malattia a cui si riferiscono: `Diabetes`, `Neuroblastoma`, `CardiacArrest`, `HeartFailure`, `Sepsis`. Nell'intestazione dei plot è comunque possibile leggere il nome completo del rispettivo dataset.

Per quanto riguarda la valutazione delle performance delle implementazioni di Silhouette ho utilizzato due dataset da me generati, indicati con `sc_dataset_good` e `sc_dataset_bad`. Li ho costruiti utilizzando la funzione `rnorm`, che genera dei punti casuali a partire da una distribuzione normale bivariata. Dataset simili compaiono anche nella descrizione della Silhouette.

`sc_dataset_good` è costituito da due distribuzioni normali fuse insieme, entrambe con una deviazione standard molto bassa, di modo che i punti siano concentrati attorno alla media. `sc_dataset_bad` è costituito da una sola distribuzione normale bivariata centrata in $(0, 0)$ e con una deviazione standard molto ampia, di modo che i punti siano molto dispersi. I loro scatter plot sono riportati in Figura 2.1 e Figura 2.2 rispettivamente.

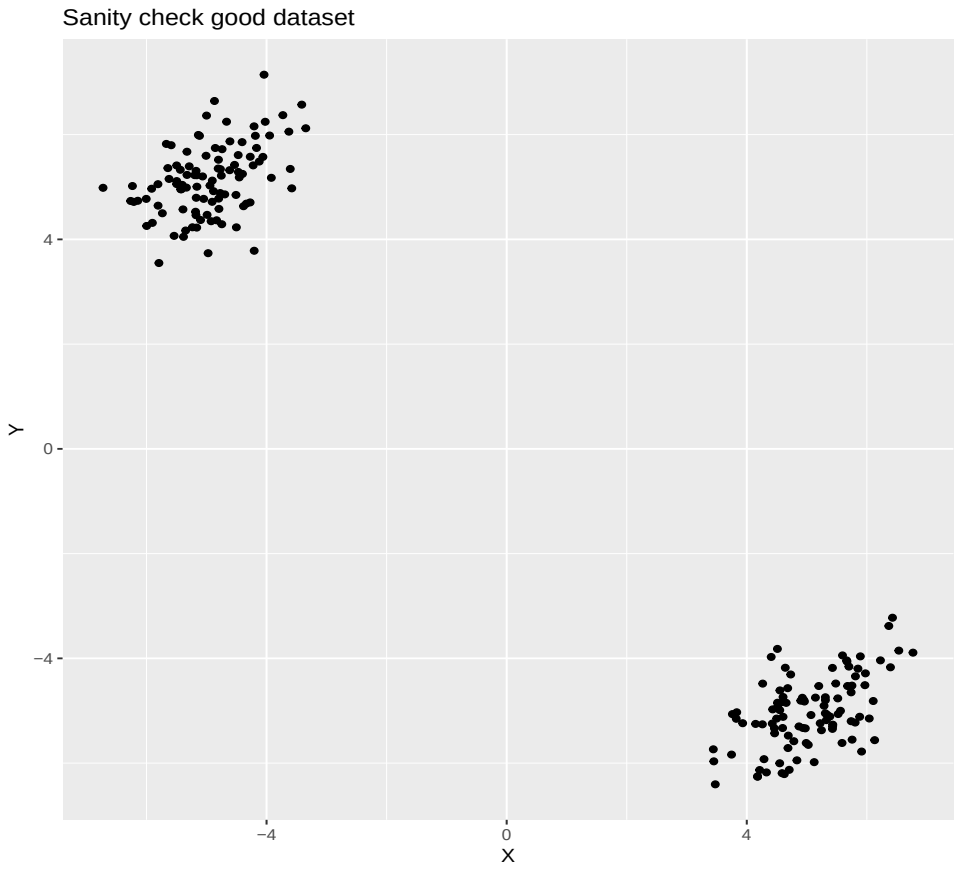


Figura 2.1: Scatter plot di `sc_dataset_good`. Si noti la struttura di cluster ben definita.

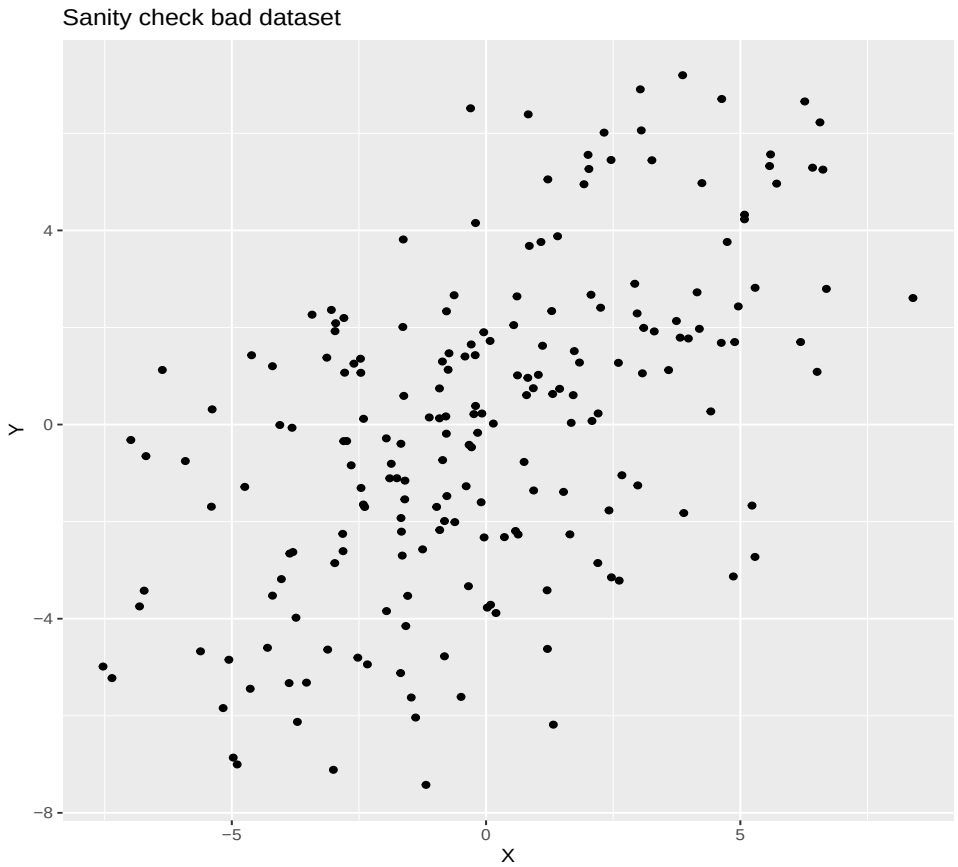


Figura 2.2: Scatter plot di `sc_dataset_bad`. Si noti l'assenza della struttura di cluster.

Infine, diversi esempi si rifanno ad un dataset molto famoso chiamato `iris` [10]. Tale dataset è costituito da 150 elementi estratti da una popolazione di fiori, ciascuno contenente cinque attributi, in ordine: lunghezza del sepalo (in centimetri), larghezza del sepalo, lunghezza del petalo, larghezza del petalo e specie del fiore (*Iris setosa*, *Iris virginica* oppure *Iris versicolor*). Dato che l'ultimo attributo non è numerico, viene in genere scartato. Nella Tabella 2.1 sono riportati alcuni elementi di tale dataset.

Sepal.Length	**Sepal.Width**	**Petal.Length**	**Petal.Width**	**Species**
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa

Tabella 2.1: Primi elementi del dataset `iris`, così come compare nel linguaggio `R`.

2.2 Funzioni di distanza

Si supponga di avere a disposizione un dataset di dimensione $N \times M$, dove N indica il numero degli elementi e M è il numero di attributi. Si assuma, per semplicità, che tutti gli attributi siano tutti dati numerici (altezze, lunghezze, capacità, ecc...) e che ogni valore di ogni attributo sia noto.

Per ciascuna coppia di elementi i e j del dataset è possibile calcolare la loro distanza mediante una funzione di distanza. La funzione di distanza più intuitiva è la **distanza Euclidea**, che equivale alla distanza fra due punti nel piano cartesiano n-dimensionale, definita come segue:

$$d(i,j) = \sqrt{\sum_{m=1}^{M} (f_{i,m} - f_{j,m})^2} = \sqrt{(f_{i,1} - f_{j,1})^2 + \cdots + (f_{i,M} - f_{j,M})^2} \tag{2.1}$$

Dove $f_{i,m}$ e $f_{j,m}$ indicano il valore del m-esimo attributo per, rispettivamente, l'i-esimo ed il j-esimo elemento del dataset.

Un'altro esempio di distanza è la **distanza di Manhattan**, calcolata come la somma fra le differenze in modulo di ciascuna coppia di attributi:

$$d(i,j) = \sum_{m=1}^{M} |f_{i,m} - f_{j,m}| = |f_{i,1} - f_{j,1}| + \cdots + |f_{i,M} - f_{j,M}| \tag{2.2}$$

Ultimo esempio di distanza è la **distanza di Jaccard**, calcolata a partire dal numero di elementi che due insiemi A e B hanno in comune ed al numero di elementi che non hanno in comune:

$$d(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|} \tag{2.3}$$

A partire da un dataset di N elementi e da una funzione di distanza è possibile costruire quella che viene chiamata **matrice delle distanze**. Tale matrice ha dimensione $N \times N$ e, in ciascuna cella (i,j), è presente la distanza $d(i,j)$ fra i due elementi. La Tabella 2.2 contiene un esempio di matrice delle distanze.

0.5385165						
0.5099020	0.3000000					
0.6480741	0.3316625	0.2449490				
0.1414214	0.6082763	0.5099020	0.6480741			
0.6164414	1.0908712	1.0862780	1.1661904	0.6164414		
0.5196152	0.5099020	0.2645751	0.3316625	0.4582576	0.9949874	
0.1732051	0.4242641	0.4123106	0.5000000	0.2236068	0.7000000	0.4242641

Tabella 2.2: Matrice delle distanze per il dataset `iris`, scartando l'ultimo attributo in quanto non numerico, usando come funzione di distanza la distanza Euclidea. Per questioni di spazio sono presenti solamente i primi 7 elementi. Si noti come la matrice sia riportata solo per metà, perché per definizione una matrice delle distanze è sempre simmetrica.

La distanza Euclidea è la funzione di distanza che viene utilizzata di default nella maggior parte delle implementazioni di default degli algoritmi di clustering. Inoltre, nella maggior parte dei casi, un algoritmo di clustering non necessita di utilizzare una funzione di distanza specifica. Pertanto, se non specificato diversamente, quando ci si riferisce ad una funzione di distanza generica si intende la distanza Euclidea.

2.3 Algoritmi di clustering

Gli algoritmi di clustering sono innumerevoli, ed è impossibile testarli tutti. In questa tesi ne ho scelti quattro: **K-Means** [16] e la sua variante **K-Medians**, **DBSCAN** [9] e **HDBSCAN** [3].

2.3.1 Clustering partizionale: K-Means e K-Medians

K-Means e **K-Medians** sono esempi di algoritmi di clustering **partizionale**, ovvero che suddividono l'insieme di dati in un certo numero di cluster operando diversi "raffinamenti" spostando uno o più elementi da un cluster all'altro fino a raggiungere la precisione desiderata. L'algoritmo può essere descritto come segue:

1. Sia scelga un intero k. Tale valore sarà il numero di cluster;
2. Si scelgano k elementi qualsiasi del dataset, detti **seed**. Tali seed fungeranno da **centroidi** iniziali, ovvero da elementi che rappresentano il "baricentro" o il "punto medio" di ciascun cluster;
3. Per ciascun elemento del dataset che non è un centroide, si calcoli la distanza fra tale elemento e tutti i centroidi. L'elemento viene assegnato alla partizione il cui centroide ha la più piccola distanza da questo;
4. Per ogni cluster se ne ricalcolino i centroidi, operando la media aritmetica dei suoi valori;
5. Se è stato raggiunto un criterio di terminazione, l'algoritmo termina. Altrimenti, si riprende dal punto 3.

Si noti come l'algoritmo non specifichi un criterio di terminazione. Un criterio molto semplice consiste nel fissare un ε e valutare di quanto si discosta il nuovo valore dei centroidi (calcolato al punto 4) dal valore precedente: se questo scostamento è inferiore ad ε, l'algoritmo termina. Un criterio simile prevede di fissare un ε e di terminare l'algoritmo se il numero di elementi che vengono assegnati ad un cluster diverso alla fine della corrente iterazione è inferiore ad ε.

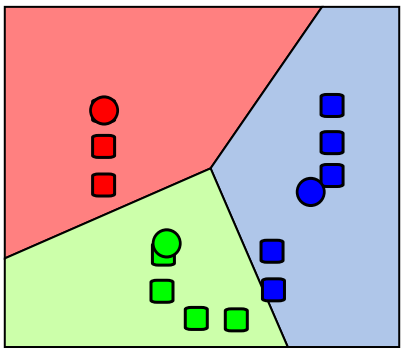


Figura 2.3: Applicazione dell'algoritmo K-Means ad un ipotetico dataset; i tre colori indicano i tre cluster individuati dall'algoritmo (By I, Weston.pace, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=2463085).

L'algoritmo K-Medians è una variante di K-Means che usa le mediane dei cluster come centroidi anziché le loro medie. Per tal motivo, mentre K-Means può utilizzare sostanzialmente qualsiasi funzione di distanza, K-Medians utilizza sempre la distanza di Manhattan.

Sebbene K-Means e K-Medians siano efficienti (se il numero di iterazioni è piccolo, il tempo di esecuzione è quasi-lineare) e anche estremamente semplici, tanto da comparire come subroutine in algoritmi più complessi, presentano dei problemi. Ad esempio, dato che ogni elemento ha lo stesso peso nel computo dei centroidi, anche un solo elemento che abbia valori anomali può destabilizzare completamente il risultato finale. Inoltre, il parametro k che determina il numero di cluster potrebbe non avere alcuna relazione con l'effettivo numero di cluster (se esistono) nell'insieme di dati, e quindi un valore scorretto di k porta a risultati che non rispecchiano per nulla la vera struttura di cluster. Infine, il raggruppamento di più elementi sulla base di una distanza genera dei cluster di forma ellittica, ma non tutti i dataset hanno una struttura di cluster con questa forma.

2.3.2 Clustering per densità: DBSCAN

DBSCAN è un esempio di algoritmo di clustering **per densità**, ovvero che costruisce i cluster a partire da come gli elementi di un dataset sono aggregati. In questo senso, i cluster figurano come regioni di spazio densamente popolate, di forma del tutto arbitraria, separate da spazio poco popolato.

DBSCAN prevede che vengano innanzitutto scelti due valori, un intero chiamato MinPts ed un numero reale positivo ε. A partire da questi, per ogni elemento p del dataset è possibile definire un insieme $N_\varepsilon(p)$, chiamato **ε-vicinato** (**ε-neighbourhood**). Tale insieme contiene tutti i punti q che hanno distanza da p inferiore a ε:

$$N_\varepsilon(p) = \{q | d(p,q) \leq \varepsilon\}$$

Ogni elemento p del dataset viene classificato sulla base del numero di elementi di $N_\varepsilon(p)$:

- Se $N_\varepsilon(p)$ ha almeno MinPts elementi, si dice che p è un **core point**;
- Se $N_\varepsilon(p)$ ha meno di MinPts elementi ma p si trova nell'ε-vicinato di un altro elemento, allora si dice che p è un **border point**;
- Se un elemento non non è né un core point né un border point, è detto **noise point**.

Si dice che un elemento q è **direttamente raggiungibile** da p se p è un core point e q si trova nell'ε-vicinato di p. Se un elemento r è direttamente raggiungibile da q e q è direttamente raggiungibile da un elemento p, allora si dice che r è **indirettamente raggiungibile** da p (si noti come la raggiungibilità non sia una proprietà necessariamente simmetrica).

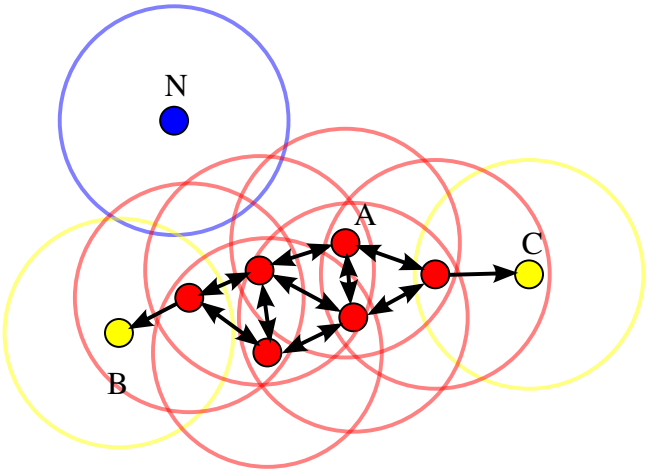


Figura 2.4: Ipotetica classificazione di alcuni elementi, usando MinPts = 4. Gli elementi in rosso sono dei core point, perché hanno 4 o più elementi nel loro ε-vicinato. Gli elementi in giallo sono invece border point, perché hanno meno di 4 elementi nel loro ε-vicinato ma sono nell'ε-vicinato di almeno un core point. Infine, gli elementi in blu sono dei noise point, perché oltre ad avere meno di 4 elementi nel loro ε-vicinato non sono nell'*epsilon*-vicinato di alcun core point (By Chire - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=17045963).

Quando DBSCAN viene invocato viene inizializzato un cluster C, dopodiché viene costruito l'ε-vicinato di ogni elemento p che non sia stato ancora ispezionato. Se tale insieme ha meno di MinPts elementi, allora p è certamente un noise point: questo perché è troppo isolato per poter essere un core point e, non essendo ancora stato ispezionato, non può trovarsi nell'ε-vicinato di nessun altro punto.

Se invece l'ε-vicinato di p ha almeno MinPts elementi, allora tale elemento è certamente un core point. Viene allora costruito un cluster C nel quale p viene inserito, dopodiché vengono osservati

tutti gli elementi q che si trovano nell'ε-vicinato di p. Se q non è mai stato ispezionato, si osserva l'ε-vicinato di q a sua volta: se questo contiene più elementi dell'ε-vicinato di p, allora $N_\varepsilon(q)$ e $N_\varepsilon(p)$ vengono uniti in un insieme unico, perché gli elementi dell'ε-vicinato di q sono indirettamente raggiungibili a partire da p. Se q non appartiene ad alcun cluster, allora viene aggiunto al cluster in esame.

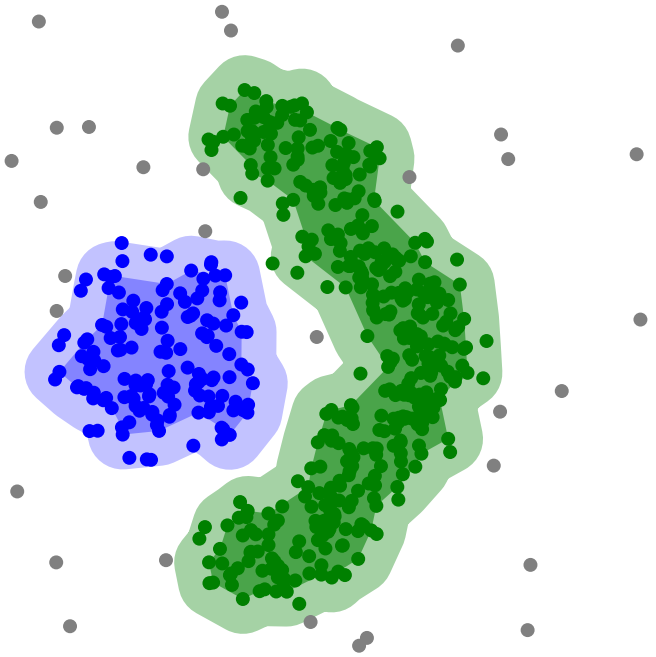


Figura 2.5: Risultato dell'applicazione dell'algoritmo DBSCAN su un ipotetico dataset. Le aree in blu e in verde rappresentano i due cluster, mentre i punti in grigio sono i noise point (By Chire - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=17085332)

Si osservi come la scelta di ε e MinPts influisca di molto sul risultato finale. Infatti, scegliendo un valore di ε o di MinPts troppo piccolo, quasi tutti gli elementi verranno classificati come noise point, e quindi quasi tutti scartati. D'altro canto, un valore di ε o di MinPts troppo grande potrebbe indurre un clustering dove quasi tutti i punti sono inclusi nello stesso cluster.

A differenza di K-Means e K-Medians, che costruiscono sempre cluster di forma ellissoidale, DBSCAN costruisce cluster di qualsiasi forma. Questo è sia un aspetto positivo, perché in questo modo è possibile catturare strutture di cluster molto più variegate, sia negativo, perché la densità di un'area non sottende necessariamente alla presenza di un cluster, e non tutte le aree dense lo sono allo stesso modo. Occorre però evidenziare come, nonostante sia leggermente inferiore a K-Means e K-Medians in termini di tempo di esecuzione ($On\log(n)$, con n numero di elementi del dataset), le sue prestazioni sono state empiricamente dimostrate come molto competitive.

2.3.3 Clustering gerarchico: HDBSCAN

HDBSCAN è un esempio di algoritmo di clustering **gerarchico**, ovvero che costruisce i cluster formando una struttura ad albero [1]

Per K-Means, è stato testato un numero di cluster compreso fra 2 e 6, mentre per K-Medians fra 2 e 10.

Per DBSCAN e HDBSCAN, MinPts è stato scelto nel range dal numero delle dimensioni del dataset al doppio più uno delle dimensioni del dataset.

Per DBSCAN, ε è stato scelto partizionando la distanza massima in parti uguali.

[1] In realtà, HDBSCAN è un algoritmo ibrido fra il paradigma per densità ed il paradigma gerarchico. Un esempio semplice di algoritmo di clustering "puramente" gerarchico é **UPGMA** [21].

2.4 Calcolo di Silhouette

Per poter calcolare Silhouette è prima necessario introdurre due quantità assegnate a ciascun elemento i del dataset, indicate rispettivamente con $a(i)$ e $b(i)$. A partire da queste sará possibile calcolare un valore $s(i)$, e la media di tutti gli $s(i)$ per ciascun i sará il valore di interesse.

Dato un dataset con N elementi e D dimensioni, se ne calcoli la matrice delle distanze e si applichi su questo un algoritmo di clustering. Si supponga che tale algoritmo individui K cluster; per ciascuno di questi, è interamente noto sia il numero di suoi elementi sia a quale cluster ciascun elemento del dataset è stato assegnato.

Preso un elemento i del dataset, sia A il cluster in cui l'algoritmo lo ha riposto. Ammesso che A contenga altri elementi all'infuori di i, è possibile definire $a(i)$ come distanza media fra i e tutti gli elementi di A escluso i stesso:

$$a(i) = \frac{1}{|A|-1} \sum_{j \in \{A-\{i\}\}} d(i,j) \tag{2.4}$$

Tale valore misura quanto un cluster è *coeso*, nel senso che se tale valore è piccolo per tutti gli elementi del cluster, questi si trovano fra loro vicini. Per tale motivo, $a(i)$ viene anche chiamata **distanza intra-cluster**.

Dopodiché, in maniera simile, per un cluster C diverso da A è possibile definire $D(i,C)$ come la distanza media fra i (che appartiene ad A) e gli elementi di C:

$$D(i,C) = \frac{1}{|C|} \sum_{j \in C} d(i,j)$$

Tale valore misura quanto un cluster è *separato*, nel senso che se tale valore è grande per tutti gli elementi del cluster a cui i appartiene, il cluster nel suo complesso si trova molto distante da tutti gli altri. Per tale motivo $D(i,C)$ viene anche chiamata **distanza inter-cluster**.

Assumendo che il numero di cluster sia più di uno, per uno stesso elemento i è possibile calcolare la distanza inter-cluster per ogni possibile cluster C distinto da A. Fra questi $K-1$ cluster, è di particolare interesse il cluster che ha il più piccolo valore di distanza inter-cluster per i, chiamato **neighboring cluster**. Questo perché tale cluster è quello che, se il cluster A non esistesse, sarebbe la miglior scelta per catalogare i, essendo quello con gli elementi più vicini ad i.

Se il neighboring cluster per i è il cluster C', la distanza inter-cluster $D(i,C')$ viene indicata con $b(i)$:

$$b(i) = min_{C \neq A} D(i,C) \tag{2.5}$$

Una volta calcolato $a(i)$ e $b(i)$ per l'elemento i del dataset, è possibile assegnarvi un valore di Silhouette $s(i)$, così calcolato:

$$s(i) = \frac{b(i)-a(i)}{max\{a(i),b(i)\}} \tag{2.6}$$

Se l'elemento i si trova in un cluster che contiene solamente sé stesso, per convenzione il valore $s(i)$ viene posto a 0 (è una scelta arbitraria, ma è anche quella più neutra).

È facile verificare che, per qualsiasi elemento i:

$$-1 \leq s(i) \leq 1$$

Si assuma infatti che $b(i) \geq a(i)$. L'espressione diventa:

$$s(i) = \frac{b(i)-a(i)}{b(i)} = \frac{b(i)}{b(i)} - \frac{a(i)}{b(i)} = -\frac{a(i)}{b(i)}$$

C	N	$s(i)$
3	1	0.85295
3	1	0.81549
3	1	0.82931
3	1	0.80501
3	1	0.84930
3	1	0.74828
3	1	0.82165
3	1	0.85390
3	1	0.75215
3	1	0.82529

C	N	$s(i)$
3	1	0.85209
1	2	0.85209
1	2	0.38118
2	1	0.85209
1	2	0.59294
1	2	0.36885
1	2	0.59221
1	2	0.28232
1	3	0.26525
1	2	0.34419

C	N	$s(i)$
1	2	0.63064
2	1	0.49927
1	2	0.23225
2	1	0.61193
2	1	0.36075
2	1	0.55777
2	1	0.54384
1	2	0.46682
2	1	0.55917
2	1	0.44076

Tabella 2.3: Valori di $s(i)$, cluster (**C**) e neighboring cluster (**N**) per i primi 10 elementi dei tre cluster ottenuti dall'applicare K-Means con $K = 3$ sul dataset `iris`. Si noti come i valori di $s(i)$ del cluster 3 siano più alti ed il neighboring cluster sia sempre lo stesso, mentre gli altri due cluster hanno valori più variegati.

Avendo assunto che $b(i)$ sia maggiore di $a(i)$, tale frazione è una frazione propria, e pertanto il suo valore è racchiuso nell'intervallo $[-1,0]$.

Si assuma invece $a(i) > b(i)$. L'espressione diventa:

$$s(i) = \frac{b(i)-a(i)}{a(i)} = \frac{b(i)}{a(i)} - \frac{a(i)}{a(i)} = \frac{b(i)}{a(i)}$$

Avendo assunto che $a(i)$ sia maggiore di $b(i)$, tale frazione è una frazione propria, e pertanto il suo valore è racchiuso nell'intervallo $[0,1]$.

Inoltre, $s(i)$ non varia se tutte le distanze vengono moltiplicate per una costante q:

$$s(i) = \frac{mb(i)-ma(i)}{max\{ma(i),mb(i)\}} = \frac{m(b(i)-a(i))}{m(max\{a(i),b(i)\})} = \frac{b(i)-a(i)}{max\{a(i),b(i)\}}$$

Per farsi una migliore idea del significato di $s(i)$, può essere utile considerare alcune situazioni estreme.

Quando $s(i)$ è approssimativamente 1 si ha che $b(i)$ è molto più grande di $a(i)$, e quindi la distanza fra i ed i membri del cluster a cui appartiene è molto più piccola della distanza fra i ed i membri degli altri cluster. Questo significa che la scelta di aver posto i in quel cluster è una buona scelta, perché persino la "seconda scelta" è di netto inferiore alla prima.

Quando $s(i)$ è approssimativamente 0 si ha che $b(i)$ e $a(i)$ hanno lo stesso ordine di grandezza, e quindi la distanza fra i ed i membri del cluster a cui appartiene è comparabile a quella fra i ed i membri del suo neighboring cluster. Questo significa che la scelta di aver posto i in quel cluster è inconclusiva, nel senso che se fosse stato invece scelto il neighboring cluster si avrebbe avuto sostanzialmente lo stesso risultato.

Quando $s(i)$ è approssimativamente -1 significa che $a(i)$ è molto più grande di $b(i)$, e quindi la distanza fra i ed i membri del cluster a cui appartiene è molto più grande della distanza fra i ed i membri degli altri cluster. Questo significa che la scelta di aver posto i in quel cluster è discutibile, perché vi sono cluster con cui i ha più in comune rispetto a quello in cui si trova.

I valori $s(i)$ non sono, di per loro, particolarmente informativi. È però possibile costruire un Silhouette plot di ciascun cluster come un bar chart dove ciascuna colonna i-esima ha altezza proporzionale a $s(i)$. Un esempio é riportato in Figura 2.6.

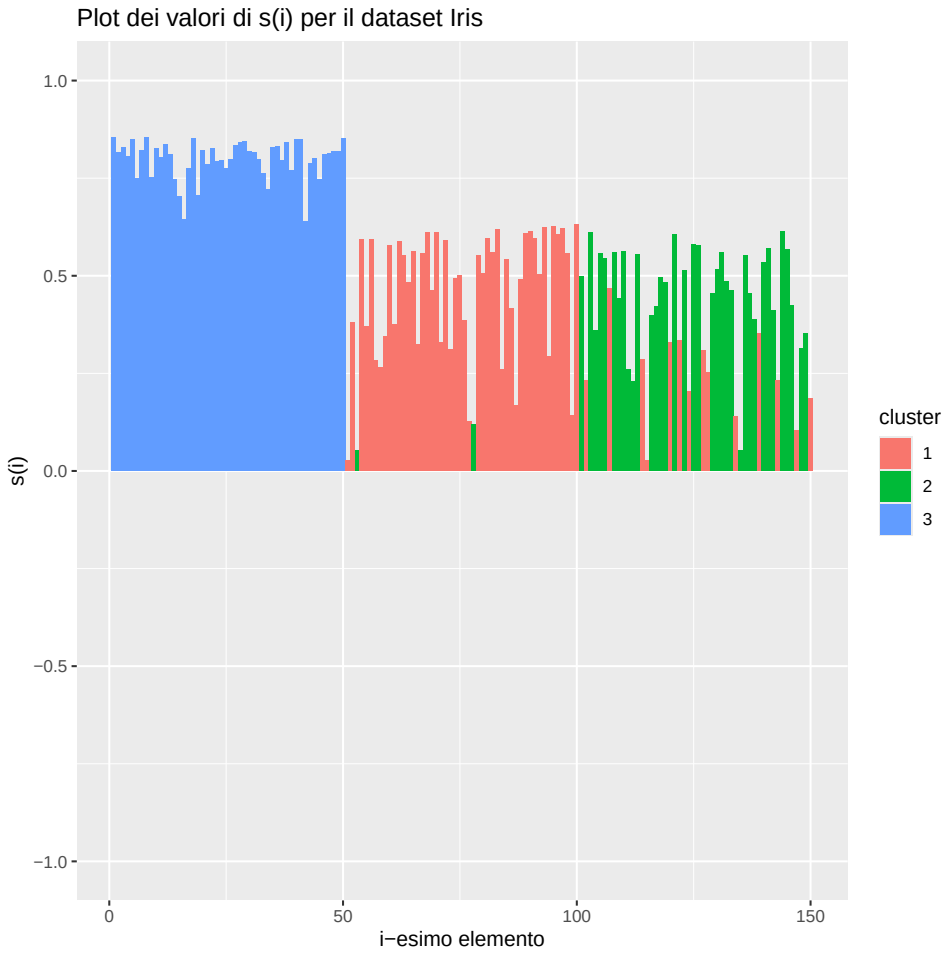


Figura 2.6: Silhouette plot per il dataset `iris`, ottenuto dopo aver applicato K-Means con $K = 3$. Per ciascun elemento i é riportato il valore di $s(i)$ ed il cluster a cui i é stato assegnato.

In generale, l'operato di un algoritmo di clustering può considerarsi ottimale se il valore si $s(i)$ tende ad essere molto alto per tutti gli elementi del dataset. A tale scopo, è possibile calcolare la Silhouette media per un certo cluster C come la media tutti gli $s(i)$ per ciascun elemento i che appartiene a C. Se tale valore medio è alto, il cluster nel suo complesso è ben formato.

Se si ha invece interesse a sapere qual'è il numero ottimale di cluster, è possibile considerare la Silhouette media complessiva come la media di tutti gli $s(i)$ per ogni elemento dell'intero dataset. Piú il valore della Silhouette media complessiva si avvicina ad 1, piú il clustering interpreta correttamente la natura del dataset.

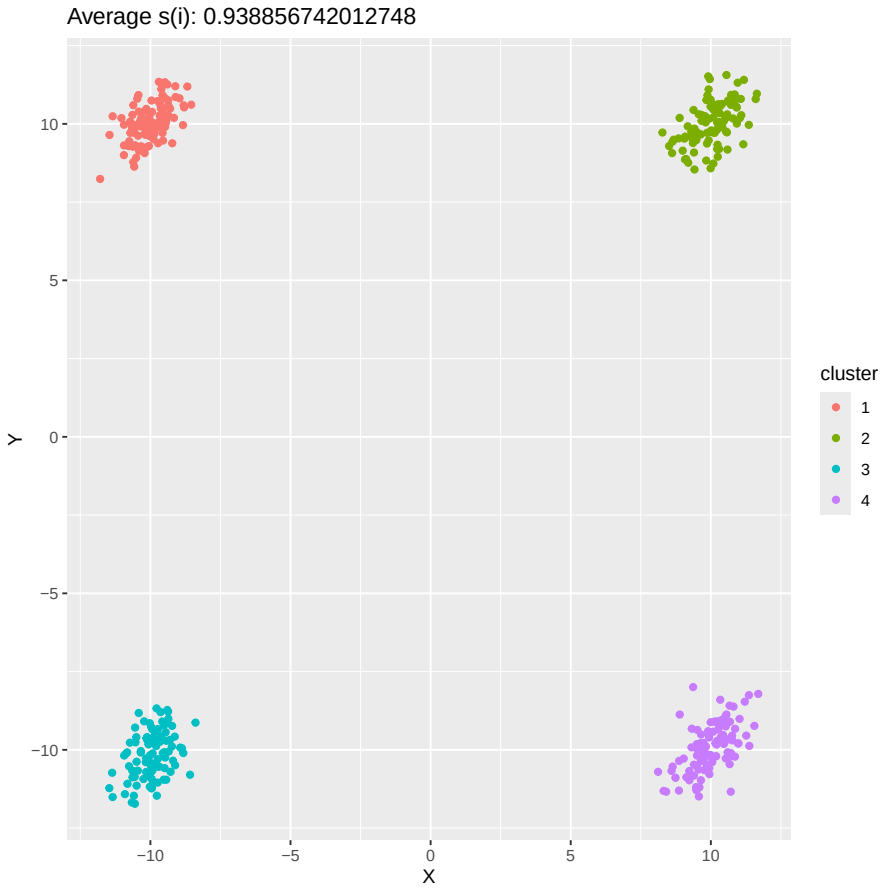


Figura 2.7: Clustering indotto da K-Means con $K = 4$ su un dataset costituito da punti di normali bivariate, dove il numero di cluster rispecchia la naturale struttura del dataset. Si noti il valore della Silhouette media complessiva molto alto.

Tutte le considerazioni fatte per i singoli valori di $s(i)$ si estendono in maniera naturale alla Silhouette media complessiva. Si supponga ad esempio che un dataset abbia delle aree molto dense separate da aree ampie vuote. Operando un clustering in cui il numero di cluster è più basso del numero ”naturale” di cluster, delle aree molto distanti tra loro vengono inglobate in un cluster unico nonostante vi siano considerevoli distanze nel mezzo. Silhouette può evidenziare questa situazione perché il valore di $a(i)$ tende ad essere molto alto, essendo i membri del dataset molto distanti dai loro centroidi. Di conseguenza, la Silhouette media complessiva sará bassa.

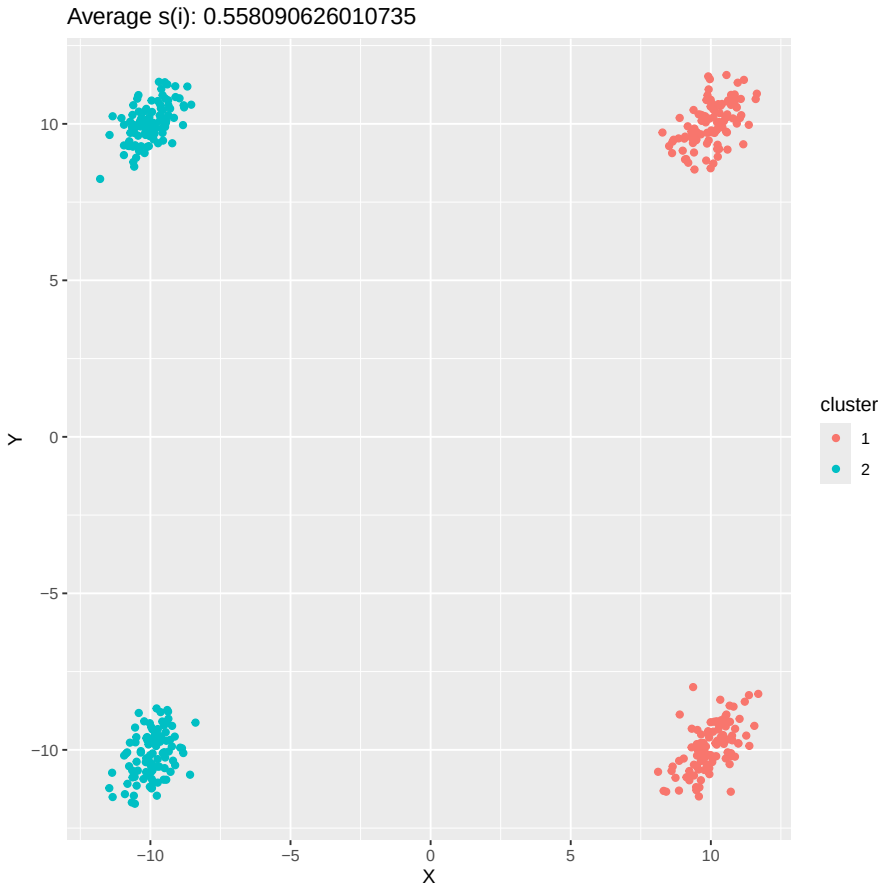


Figura 2.8: Clustering indotto da K-Means con $K = 2$ su un dataset costituito da punti di normali bivariate, dove il numero di cluster é inferiore del numero di cluster naturali. Si noti il valore della Silhouette media complessiva basso.

Si supponga invece di operare un clustering in cui il numero di cluster è più alto del numero ”naturale” di cluster. In tale situazione, anche aree dense vengono spezzate in cluster diversi. Silhouette può evidenziare questa situazione perché il valore di $b(i)$ tende ad essere molto basso, dato che elementi molto vicini vengono separati forzosamente. Anche in questo caso, la Silhouette media complessiva sará bassa.

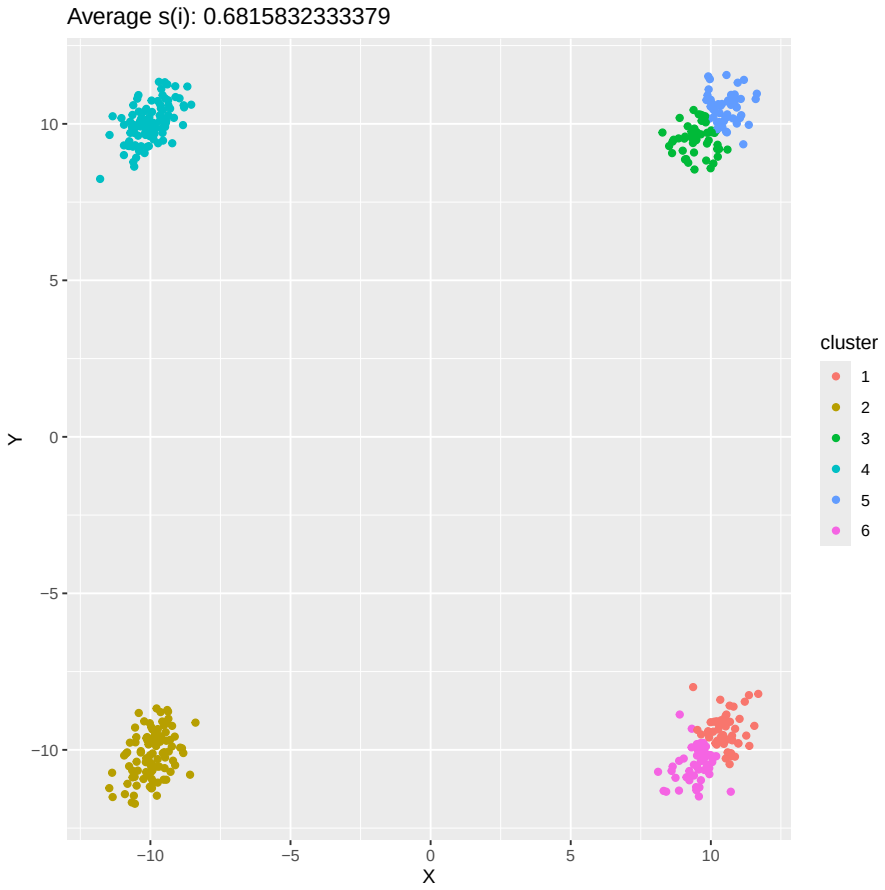


Figura 2.9: Clustering indotto da K-Means con $K = 6$ su un dataset costituito da punti di normali bivariate, dove il numero di cluster é superiore al numero di cluster naturali. Si noti il valore della Silhouette media complessiva basso.

Si noti come un valore della Silhouette media complessiva pari a 0 non significa necessariamente che il clustering non sia andato a buon fine. Può infatti anche indicare che effettivamente il dataset non ha alcuna struttura di clustering naturale, e che quindi l'algoritmo di clustering ha comunque fornito un risultato corretto, dato che effettivamente qualsiasi risultato vale l'altro.

2.5 Test sanity check e matrice binaria

Il linguaggio `R` offre diverse implementazioni del calcolo della Silhouette. A differenza di altri linguaggi come Python, dove esiste un "consensus" (ufficiale o ufficioso) riguardo a quali siano i pacchetti da utilizzare per un determinato scopo, su `R` questo talvolta manca. Pertanto, prima di mettere Silhouette sotto analisi, è necessario scegliere un pacchetto fra quelli disponibili.

Ho cercato quante più implementazioni di Silhouette possibili utilizzando il sito `https://rdrr.io` e scegliendo solamente quelli provenienti dal repository CRAN, di modo da avere la certezza di reperire solamente pacchetti di qualitá. In particolare, ho scelto `cluster`, `drclust`, `tidyclust` e `Kira`. Pacchetti noti come `fpc`, che pure implementano Silhouette, li ho esclusi a priori perché non aggiornati da tempo.

Oltre a questi, come controprova ho utilizzato l'implementazione della Silhouette presente nel pacchetto `scikit-learn` per `Python`. Questo è stato fatto attraverso il pacchetto `reticulate`, che permette di chiamare funzioni `Python` all'interno di codice `R`. Essendo `scikit-learn` considerata una implementazione fidata, é possibile usarla come riferimento per fare dei confronti.

Per comparare le performance delle diverse implementazioni di Silhouette ho eseguito due test, uno chiamato "sanity check" ed uno chiamato "matrice binaria".

Il test sanity check prevede di applicare K-Means con $K = 2$ sui dataset `sc_dataset_good` e `sc_dataset_bad` e calcolare la Silhouette media complessiva per entrambi. L'idea é che una implementazione corretta di Silhouette fornisca un valore molto alto per la Silhouette media complessiva rispetto al primo dataset, dove il numero di cluster scelto rispecchia perfettamente la struttura del dataset, ed un valore molto basso per il secondo, in cui i dati non hanno alcuna struttura.

Il test matrice binaria prevede invece di generare una matrice di dimensione 20×10, costituita per metà da 0 e per metà da 1. Su tale matrice viene applicato K-Means con $K = 2$ e si calcola la Silhouette media complessiva del risultato. Dopodiché, una qualsiasi delle righe viene sostituita con un valore scelto casualmente nell'intervallo $(0, 1)$ e si ripete il procedimento fino a quando tutte le righe hanno subito esattamente una sostituzione.

I $2n$ valori della Silhouette media complessiva così trovati sono poi riportati in un plot; l'idea è che tali valori debbano essere alti nelle prime istanze del test quando gli 0 e gli 1 sono ben separati e, mano a mano che nel dataset vengono introdotti valori casuali, si abbassino sempre piú. Pur ammettendo delle flutturazioni, ci si aspetta che tali valori si distribuiscano linearmente.

3. Risultati ottenuti

3.1 Risultati dei test sanity check su pacchetti R

Il risultati del test sanity check sono riportati nei plot seguenti. Per ciascun pacchetto sono riportati sia i valori della Silhouette media complessiva, sia il tempo di esecuzione necessario. Il colore degli elementi rappresenta il cluster nel quale è stato assegnato. Per ciascun pacchetto il test è stato ripetuto due volte: una volta usando `sc_dataset_good` ed una volta usando `sc_dataset_bad`.

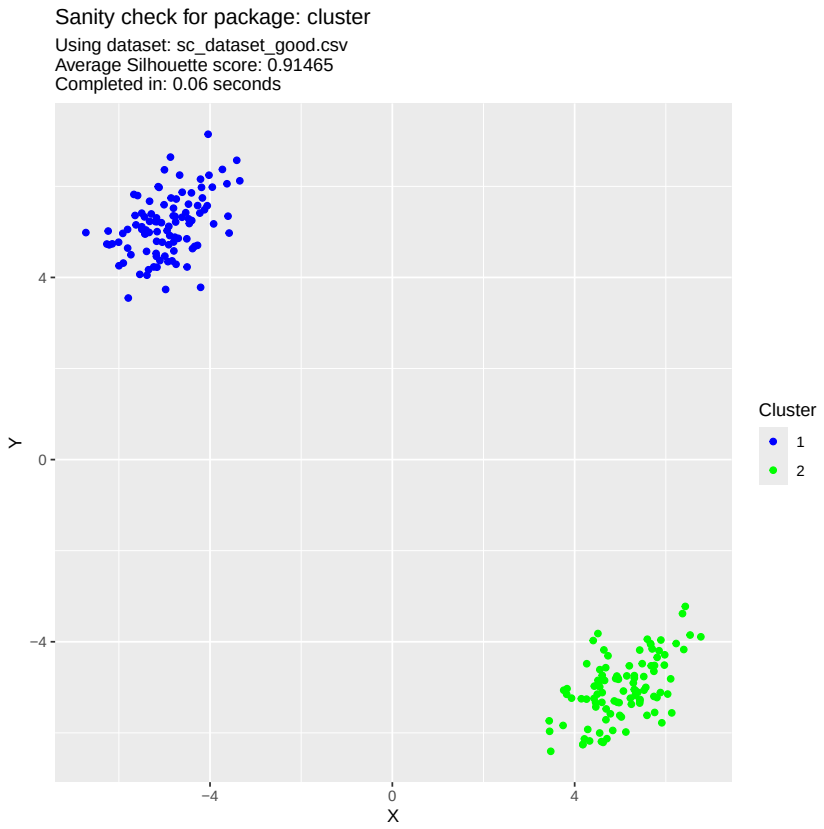


Figura 3.1: Risultato del test sanity check per il pacchetto `cluster`, usando `sc_dataset_good` come dataset.

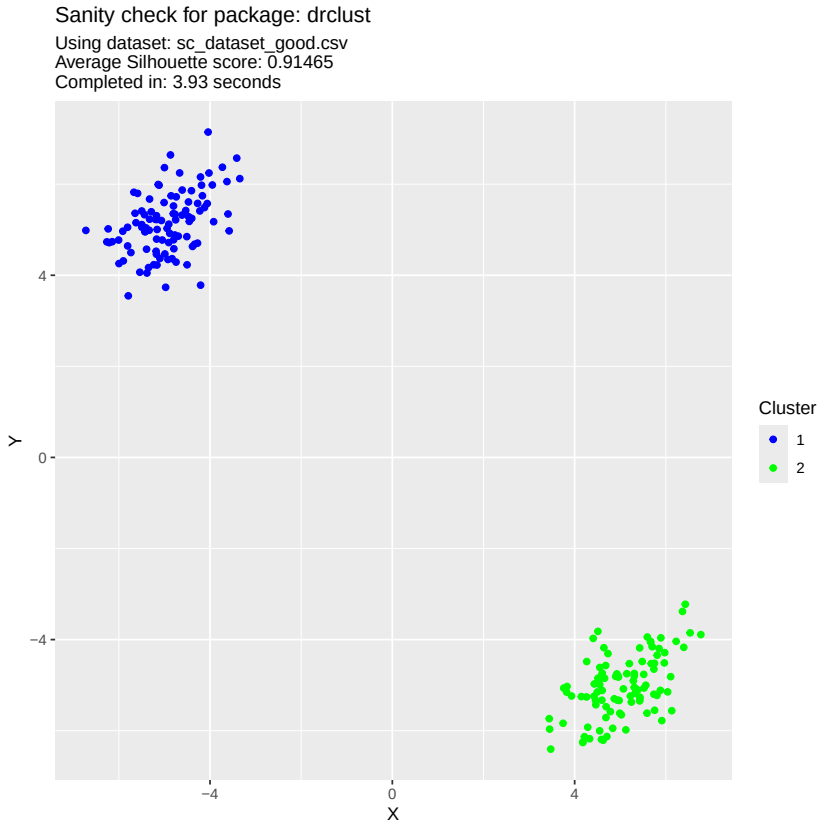


Figura 3.2: Risultato del test sanity check per il pacchetto `drclust`, usando `sc_dataset_good` come dataset.

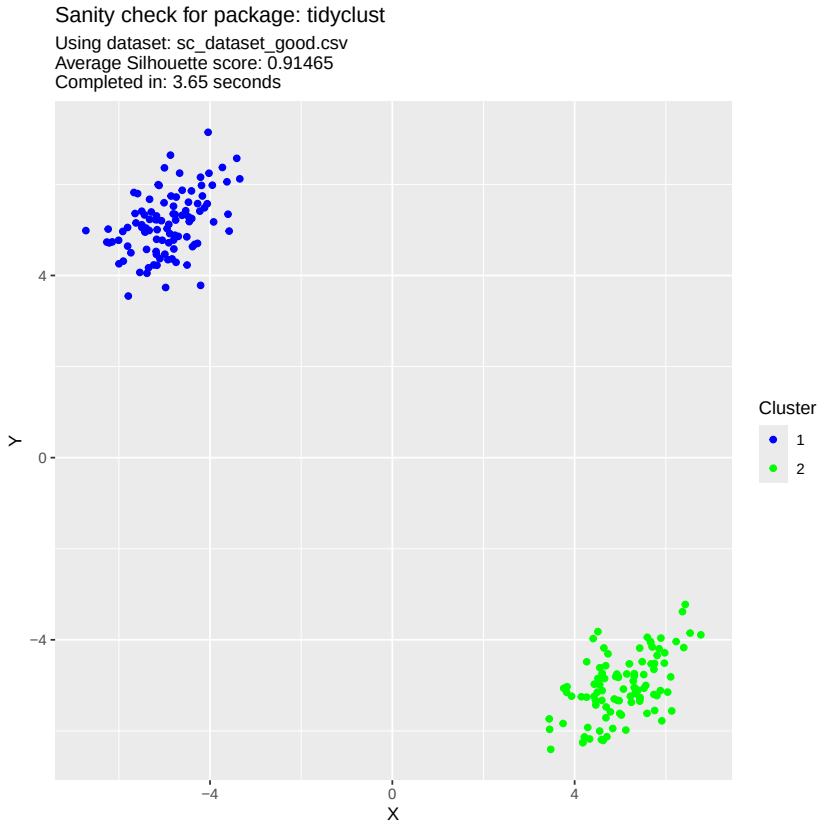


Figura 3.3: Risultato del test sanity check per il pacchetto `tidyclust`, usando `sc_dataset_good` come dataset.

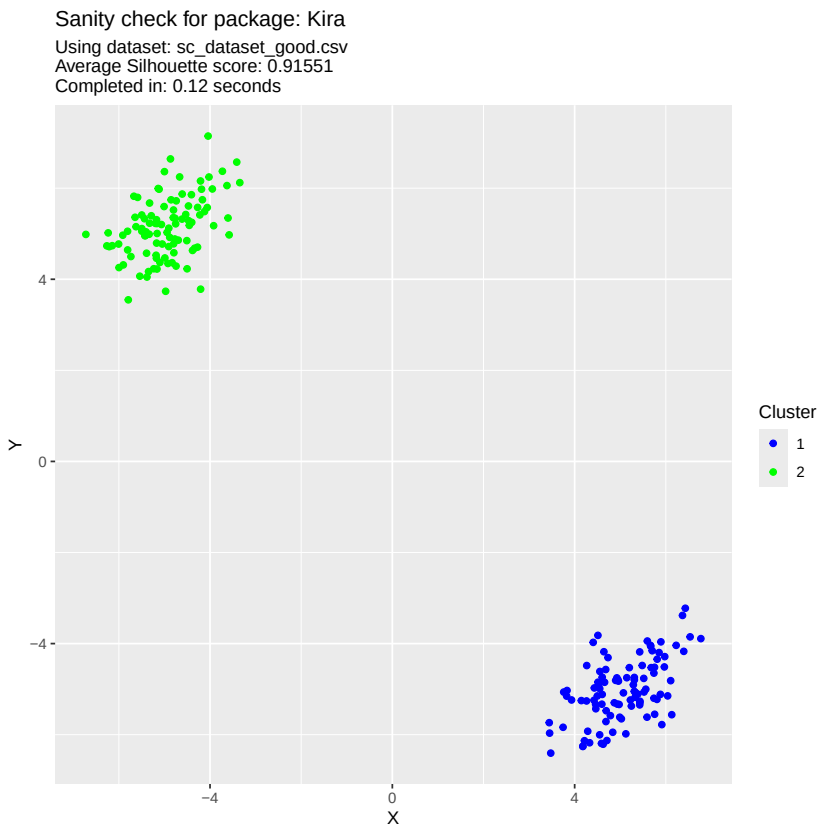


Figura 3.4: Risultato del test sanity check per il pacchetto `kira`, usando `sc_dataset_good` come dataset.

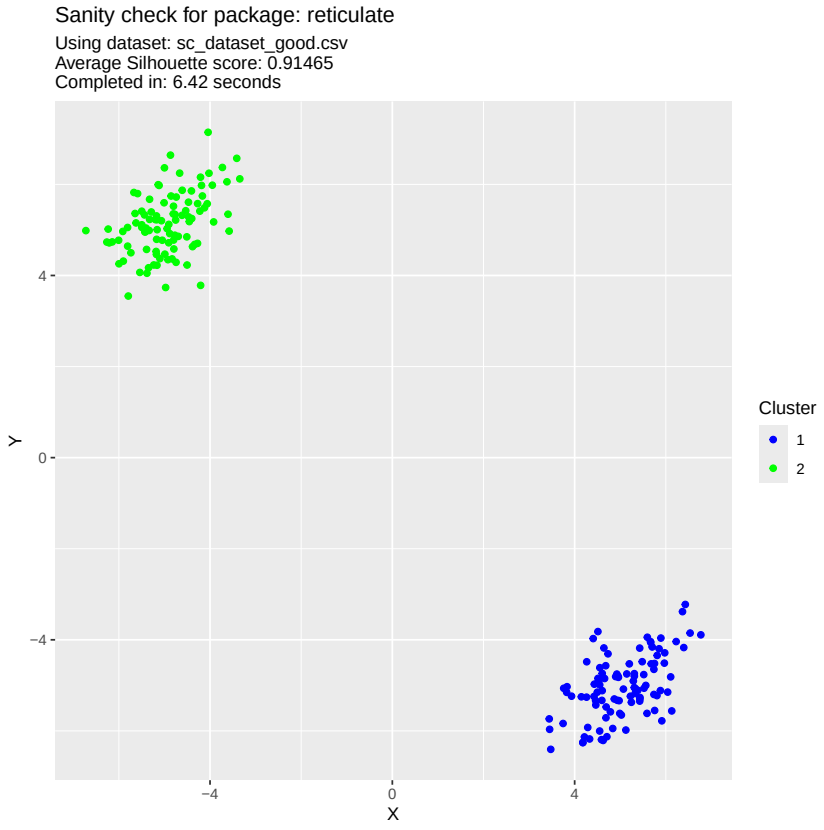


Figura 3.5: Risultato del test sanity check per il pacchetto `scikit-learn` tramite `reticulate`, usando `sc_dataset_good` come dataset.

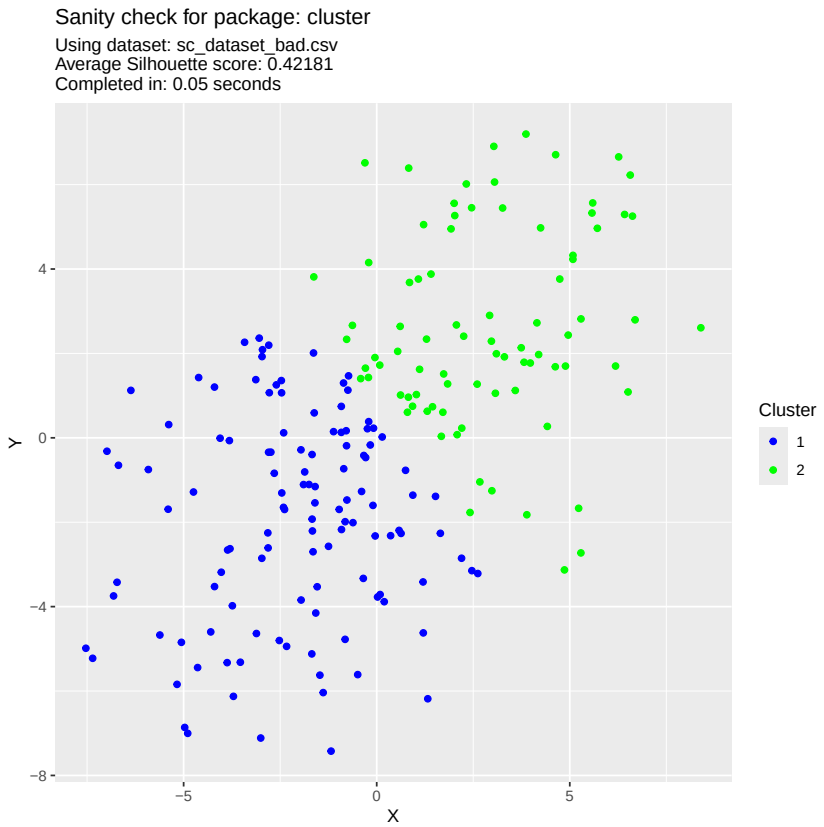


Figura 3.6: Risultato del test sanity check per il pacchetto `cluster`, usando `sc_dataset_bad` come dataset.

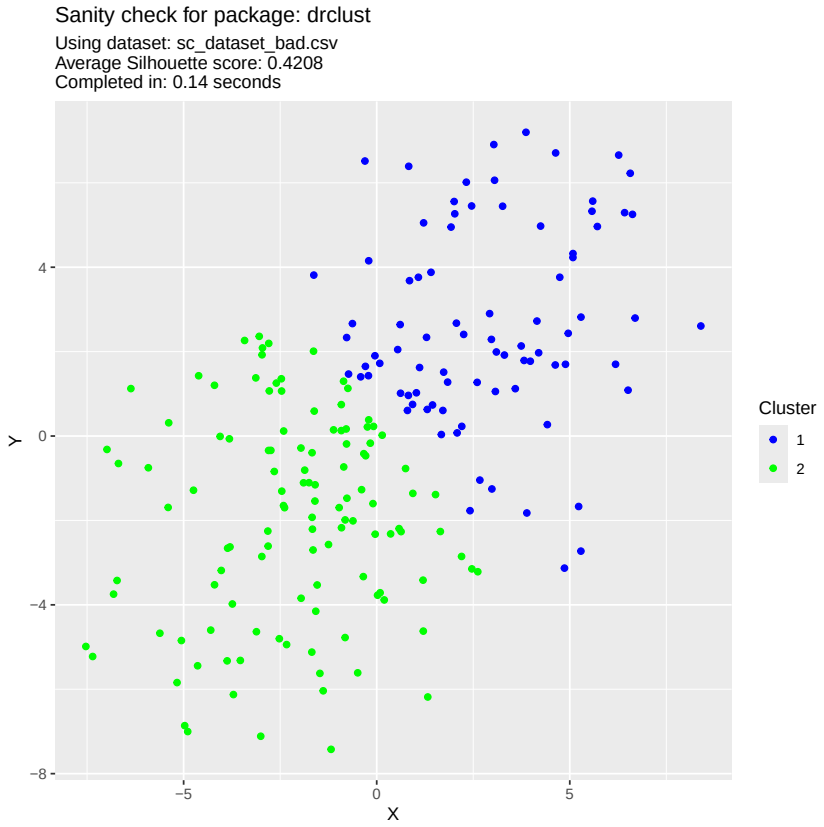


Figura 3.7: Risultato del test sanity check per il pacchetto `drclust`, usando `sc_dataset_bad` come dataset.

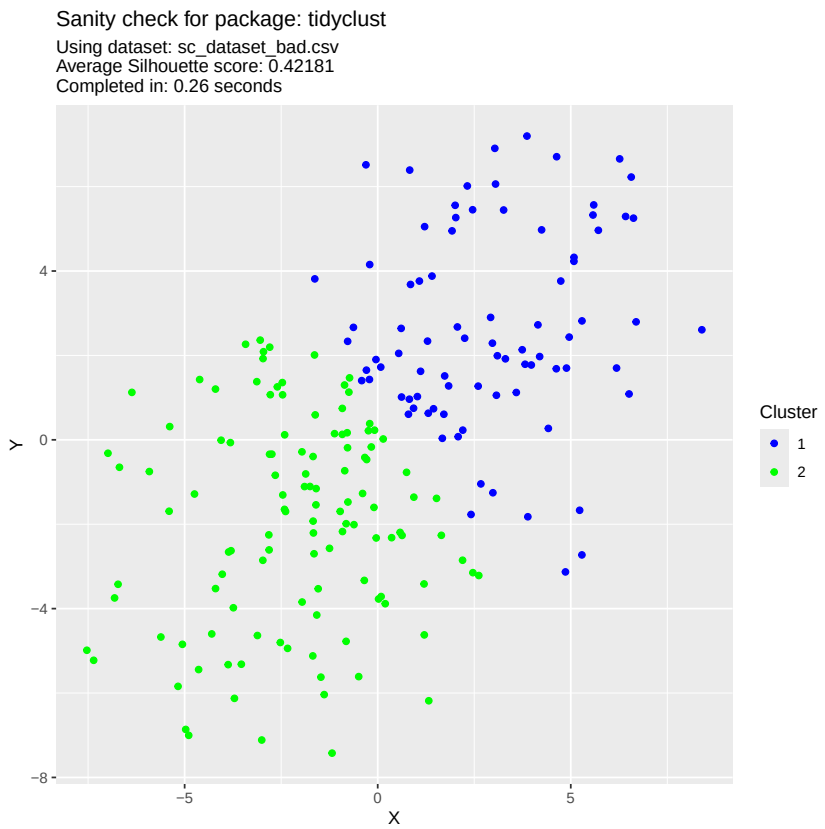


Figura 3.8: Risultato del test sanity check per il pacchetto `tidyclust`, usando `sc_dataset_bad` come dataset.

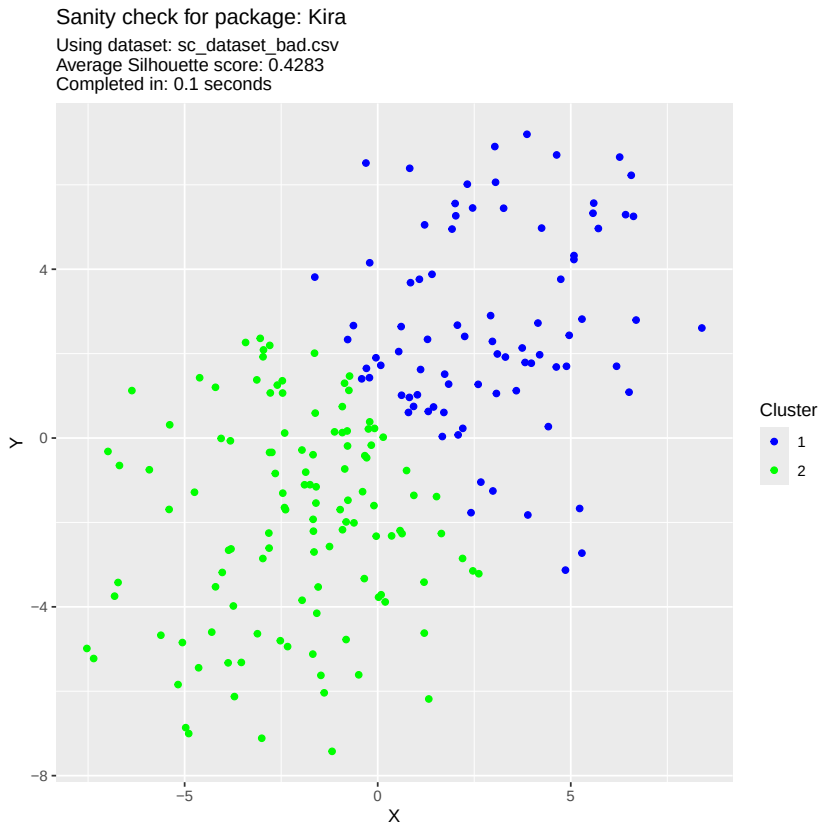


Figura 3.9: Risultato del test sanity check per il pacchetto `kira`, usando `sc_dataset_bad` come dataset.

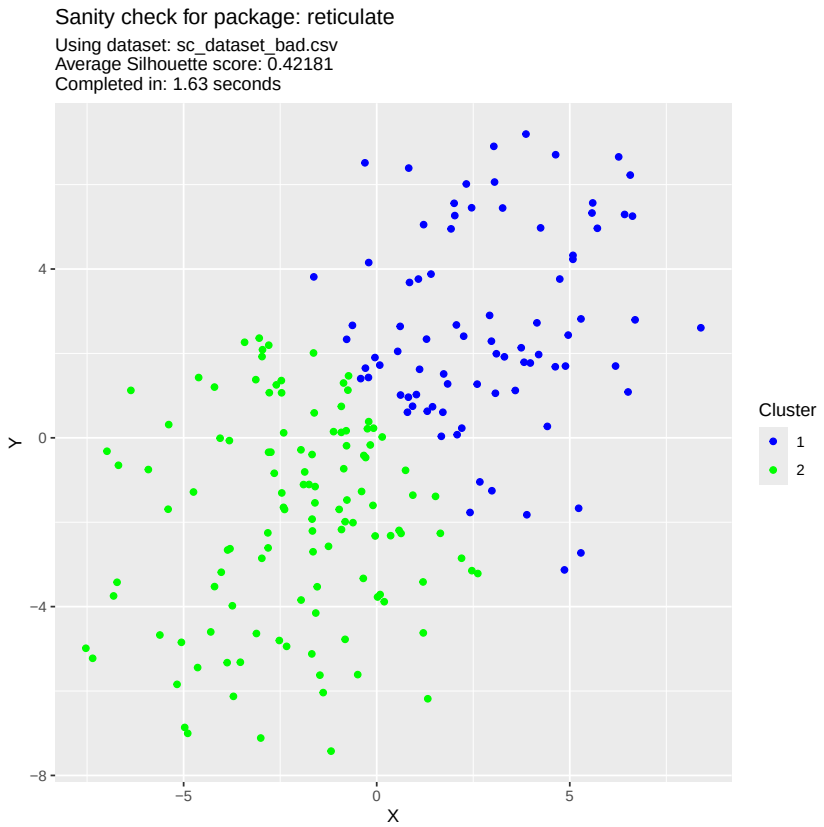


Figura 3.10: Risultato del test sanity check per il pacchetto `scikit-learn` tramite `reticulate`, usando `sc_dataset_bad` come dataset.

3.2 Risultati dei test matrice binaria su pacchetti R

I risultati per il test matrice binaria sono riportati nei plot seguenti. Per ciascun dataset, sono riportati i valori della Silhouette media complessiva per ciascuna delle 20 iterazioni dell'algoritmo. Sull'asse delle ascisse è riportato il numero delle iterazioni, mentre su quello delle ordinate il valore della Silhouette media complessiva. Essendo Silhouette un valore compreso fra -1 e 1, le ordinate sono normalizzate su tale intervallo. Sovrimposta a tale sequenza di punti figura la retta di regressione lineare.

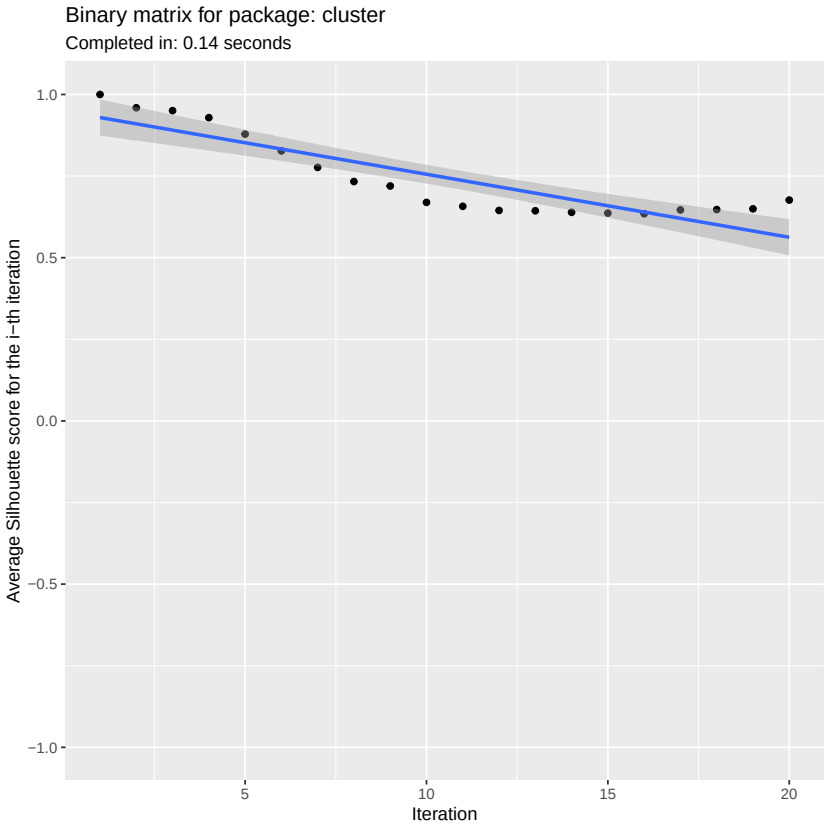


Figura 3.11: Risultato del test matrice binaria per il pacchetto `cluster`.

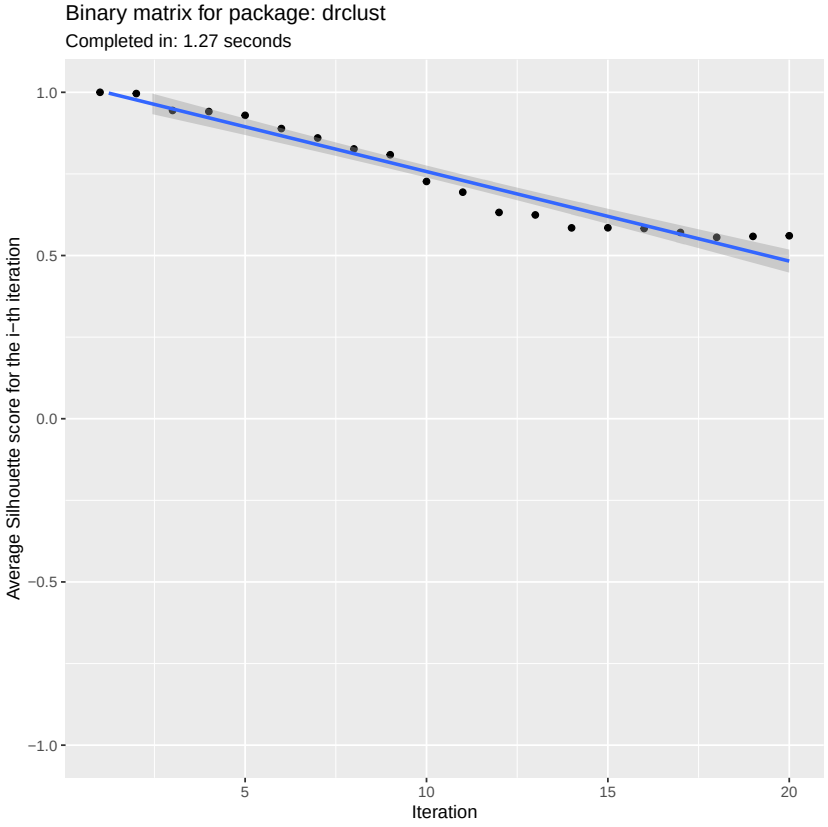


Figura 3.12: Risultato del test matrice binaria per il pacchetto `drclust`.

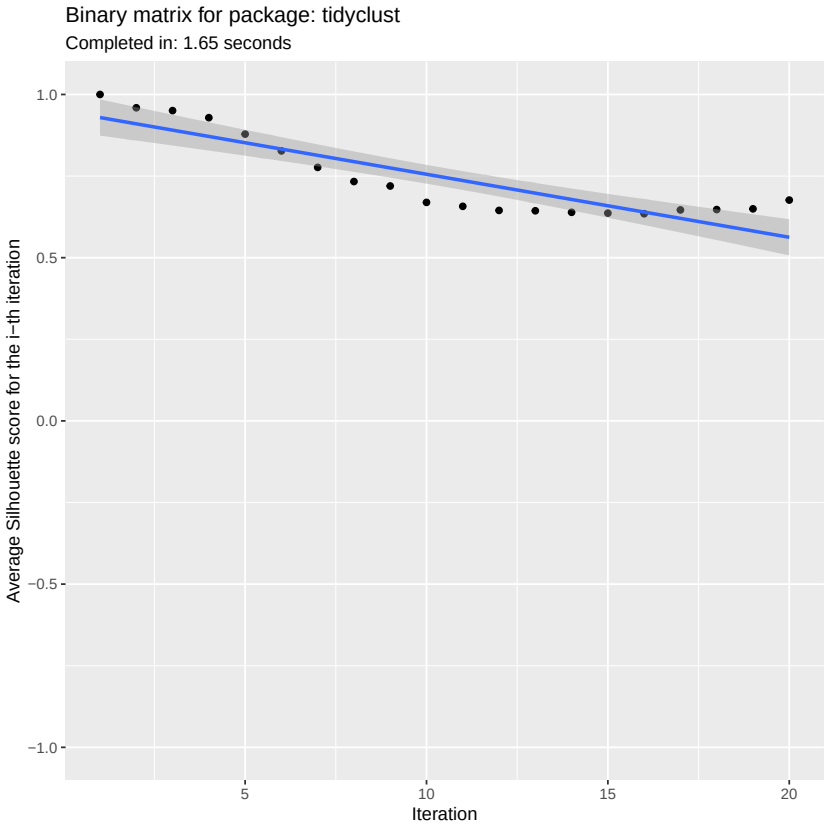


Figura 3.13: Risultato del test matrice binaria per il pacchetto `tidyclust`.

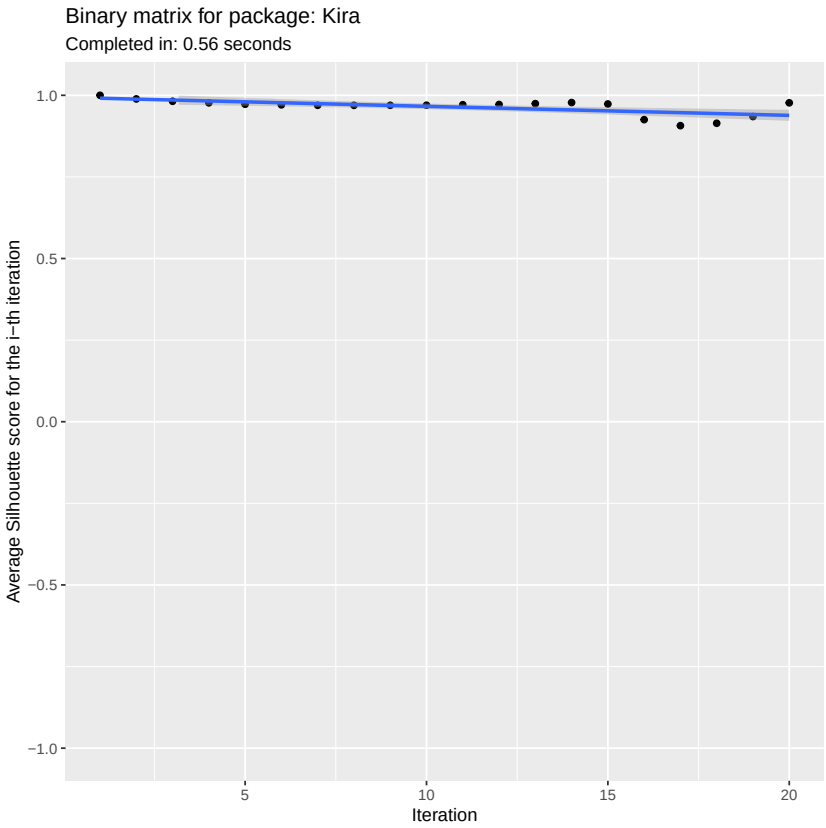


Figura 3.14: Risultato del test matrice binaria per il pacchetto `kira`.

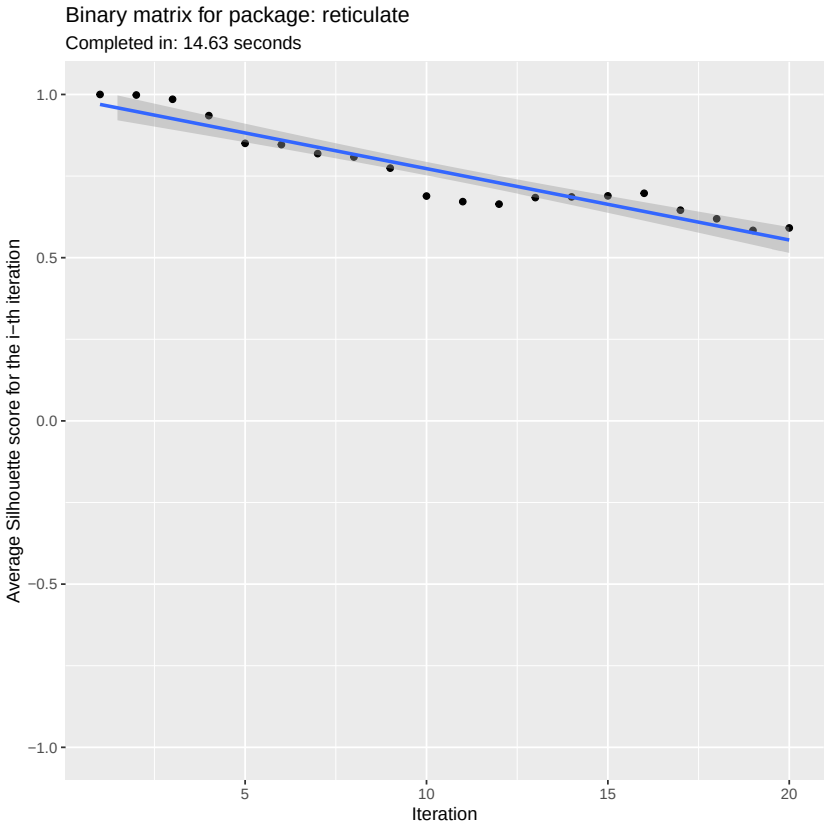


Figura 3.15: Risultato del test matrice binaria per il pacchetto `scikit-learn` tramite `reticulate`.

3.3 Risultati delle applicazioni di algoritmi di clustering su EHR

I risultati dell'applicazione dei quattro algoritmi di clustering sui dataset EHR citati sono riportati di seguito. Ciascuna combinazione algoritmo/dataset è formata da due plot.

Il primo plot è una o più linee spezzate dove ciascun punto ha per ascissa il valore di un certo parametro e per ordinata il valore della Silhouette media complessiva del clustering operato usando tale parametro. Nel caso in cui i parametri dell'algoritmo siano due, sono riportate più rette, e ciascuna retta rappresenta la scelta di uno dei due parametri.

Il secondo plot è un grafico a barre dove ciascuna colonna rappresenta, in percentuale, il numero di elementi che l'algoritmo ha assegnato a tale cluster.

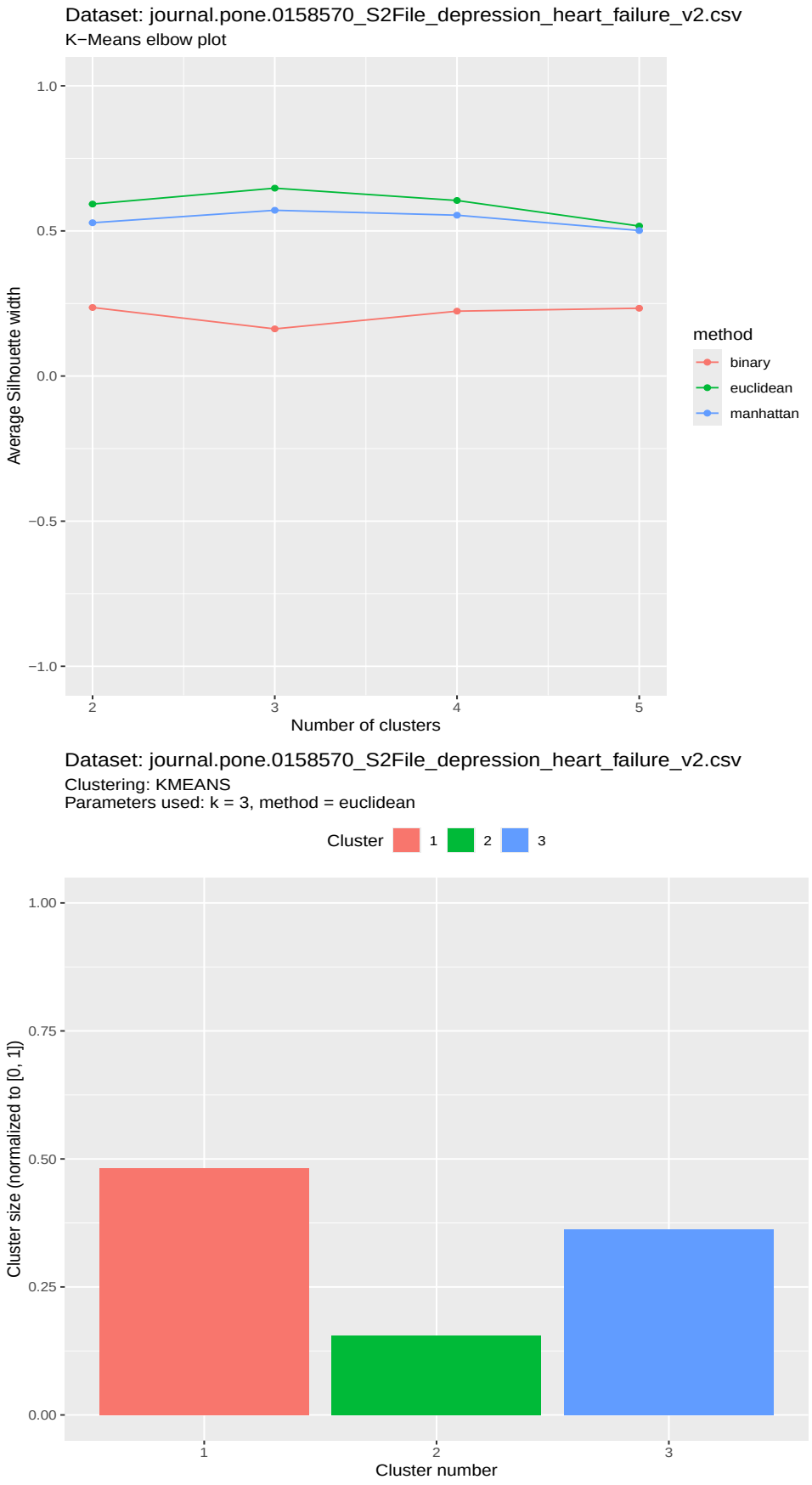


Figura 3.16: Risultati dell'algoritmo K-Means per il dataset `HeartFailure`

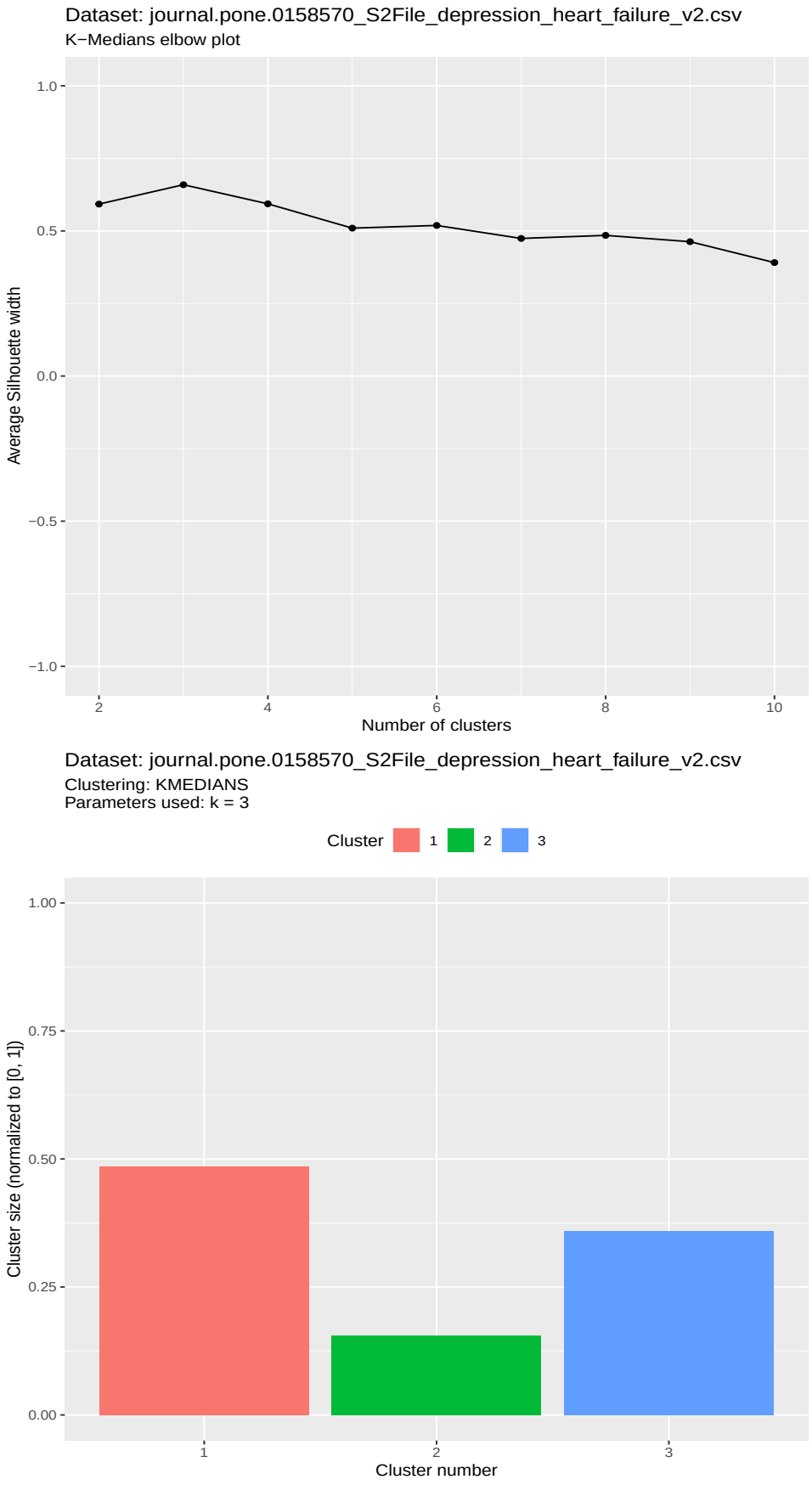


Figura 3.17: Risultati dell'algoritmo K-Medians per il dataset `HeartFailure`

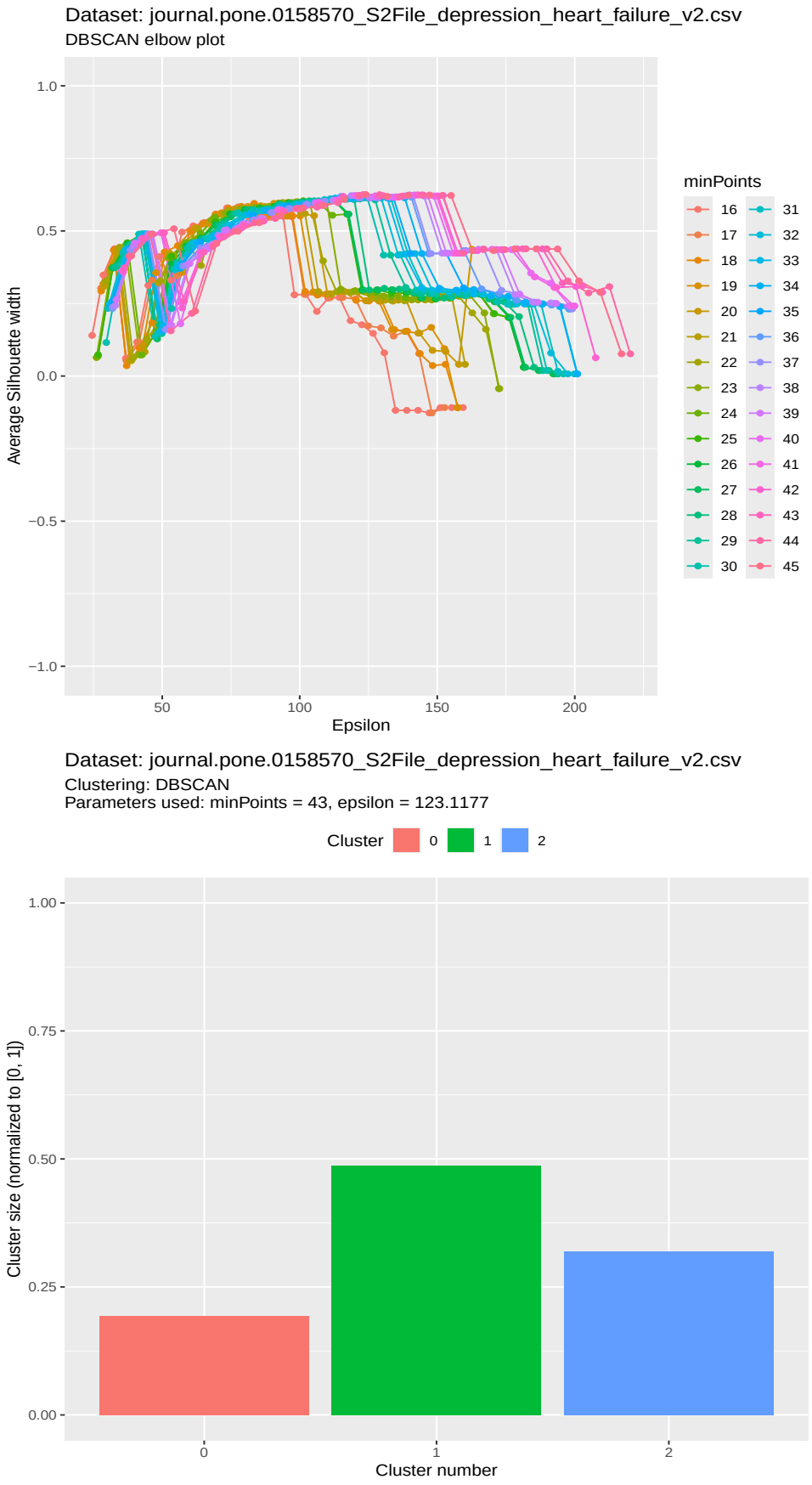


Figura 3.18: Risultati dell'algoritmo DBSCAN per il dataset `HeartFailure`

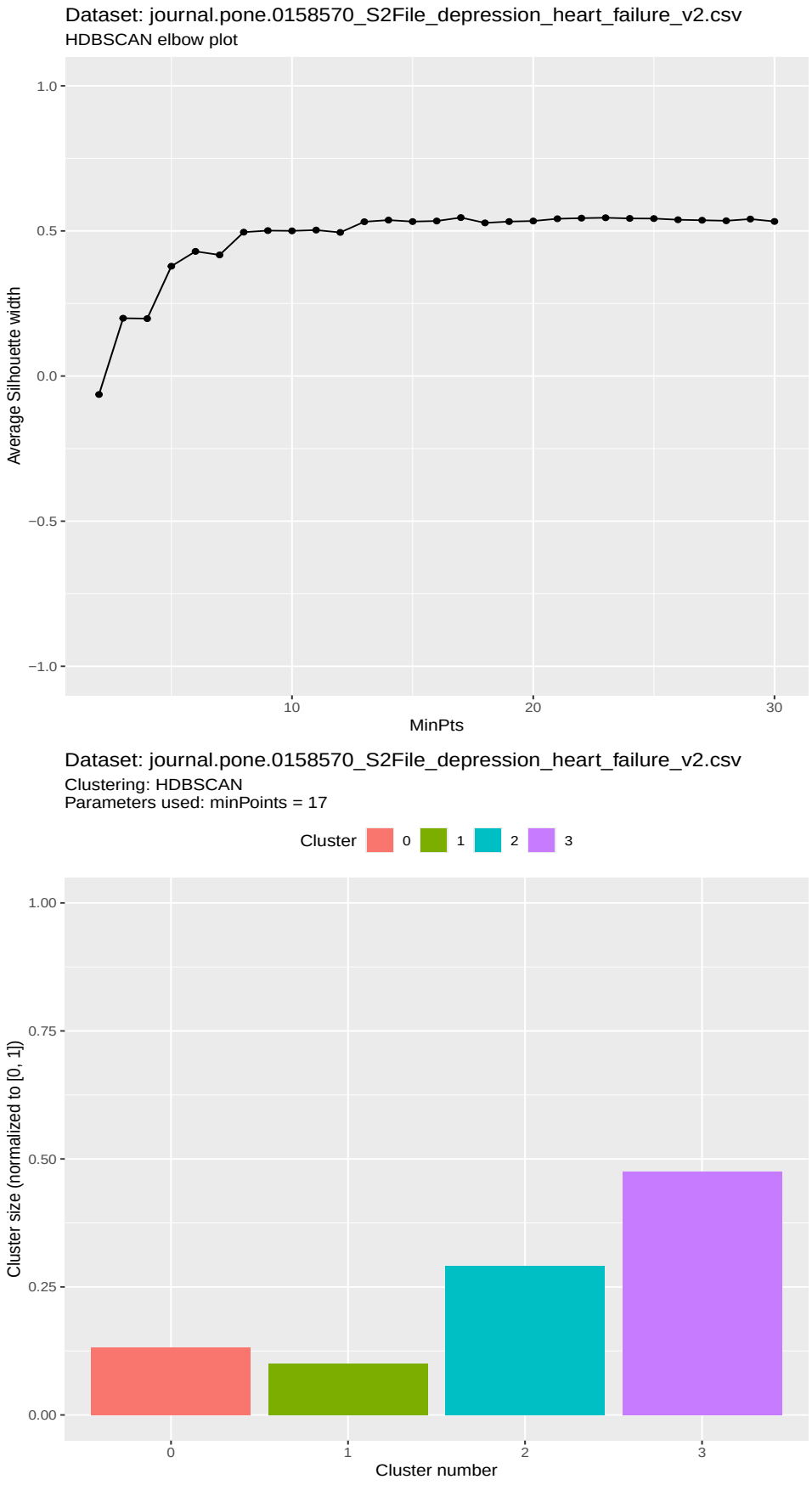


Figura 3.19: Risultati dell'algoritmo HDBSCAN per il dataset `HeartFailure`

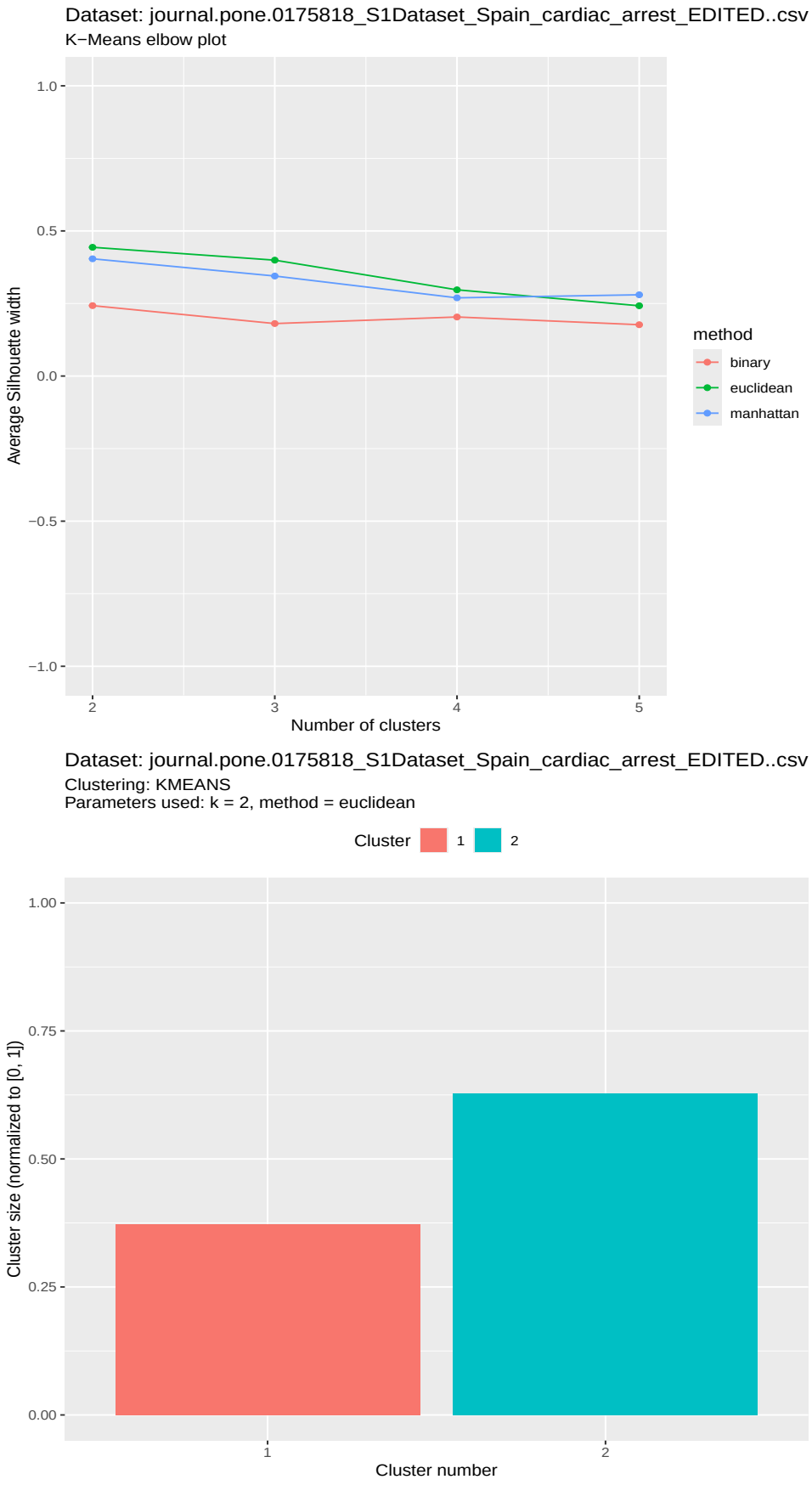


Figura 3.20: Risultati dell'algoritmo K-Means per il dataset `CardiacArrest`

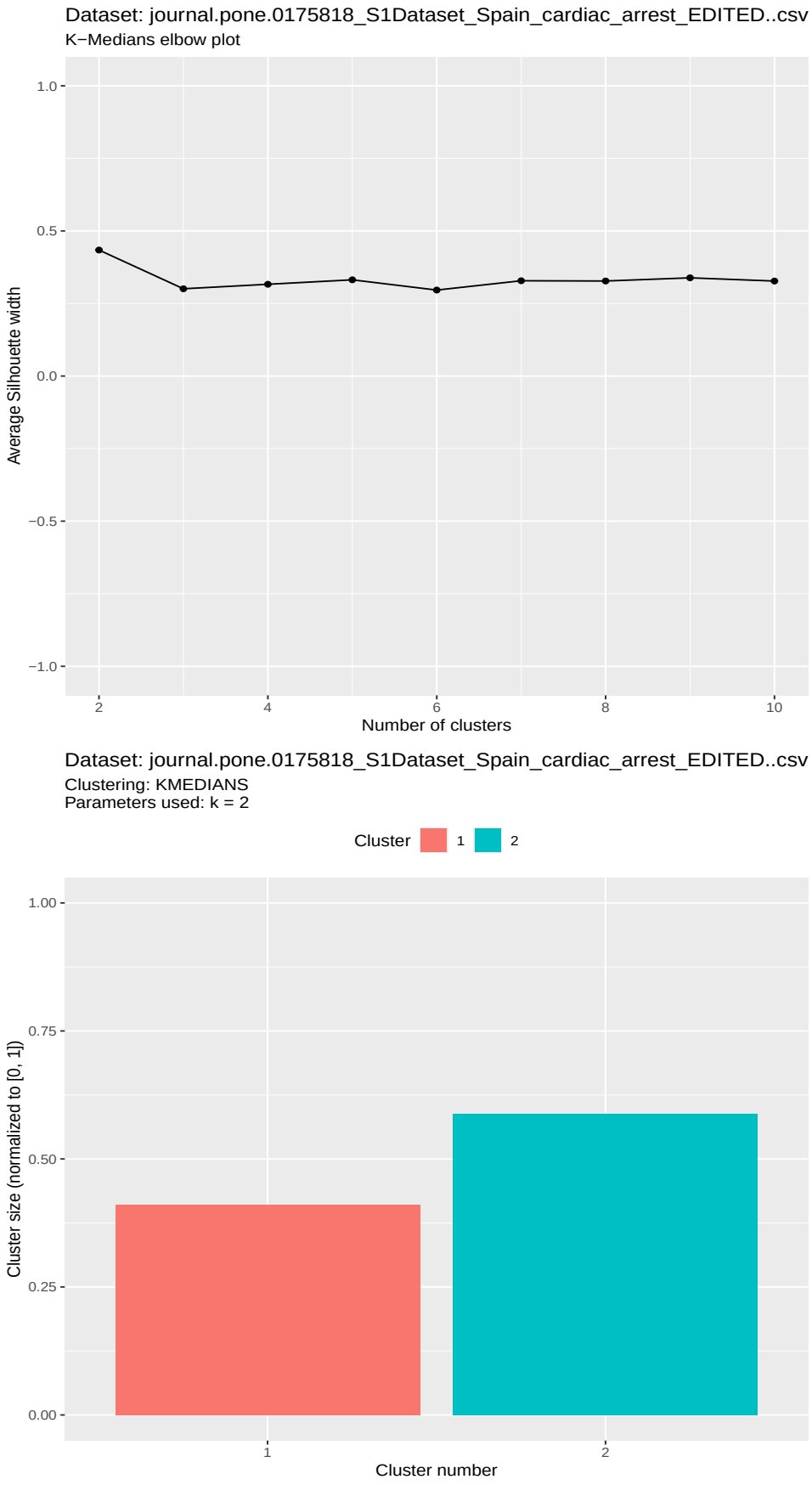


Figura 3.21: Risultati dell'algoritmo K-Medians per il dataset `CardiacArrest`

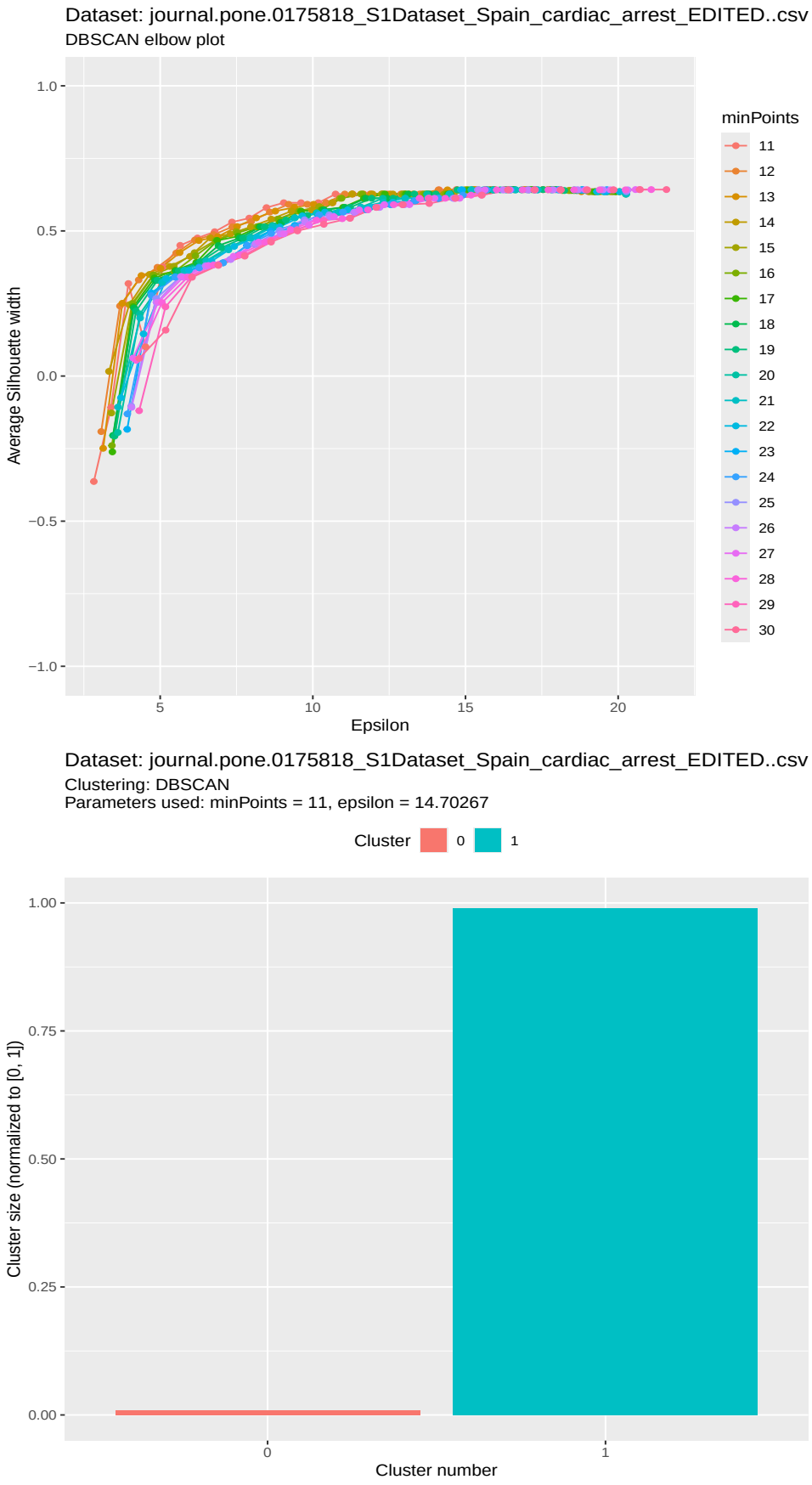


Figura 3.22: Risultati dell'algoritmo DBSCAN per il dataset `CardiacArrest`

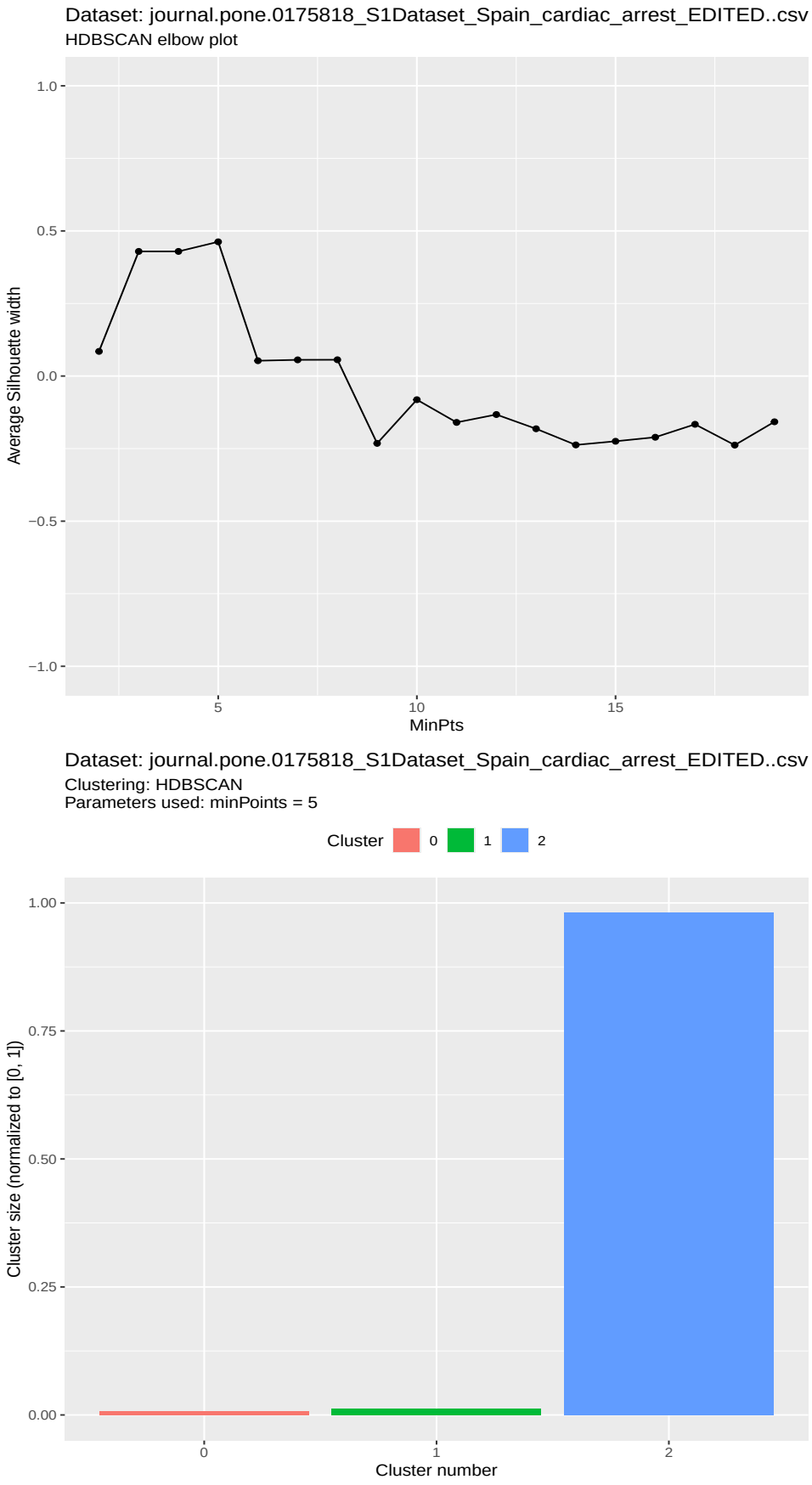


Figura 3.23: Risultati dell'algoritmo HDBSCAN per il dataset `CardiacArrest`

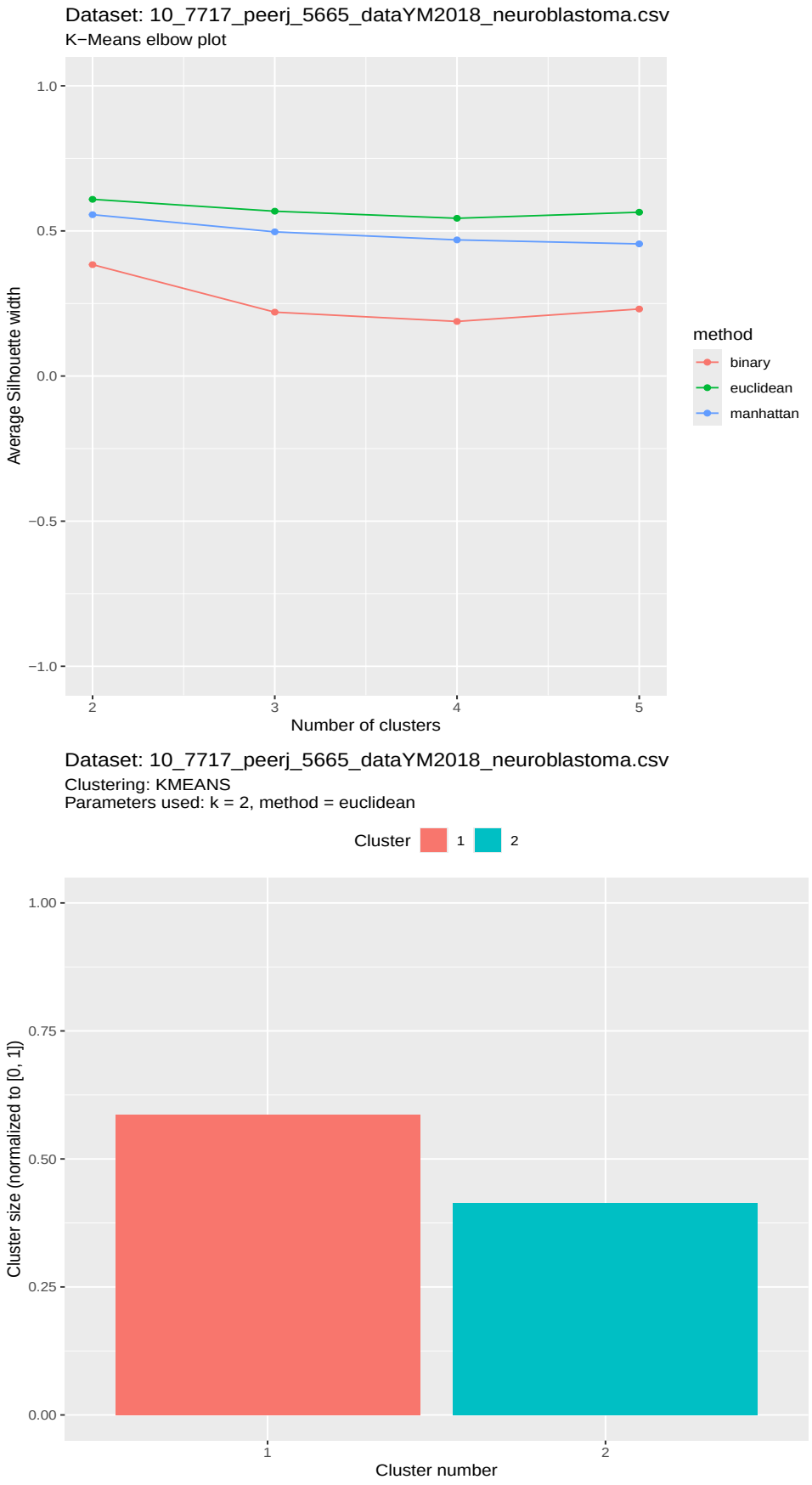


Figura 3.24: Risultati dell'algoritmo K-Means per il dataset `Neuroblastoma`

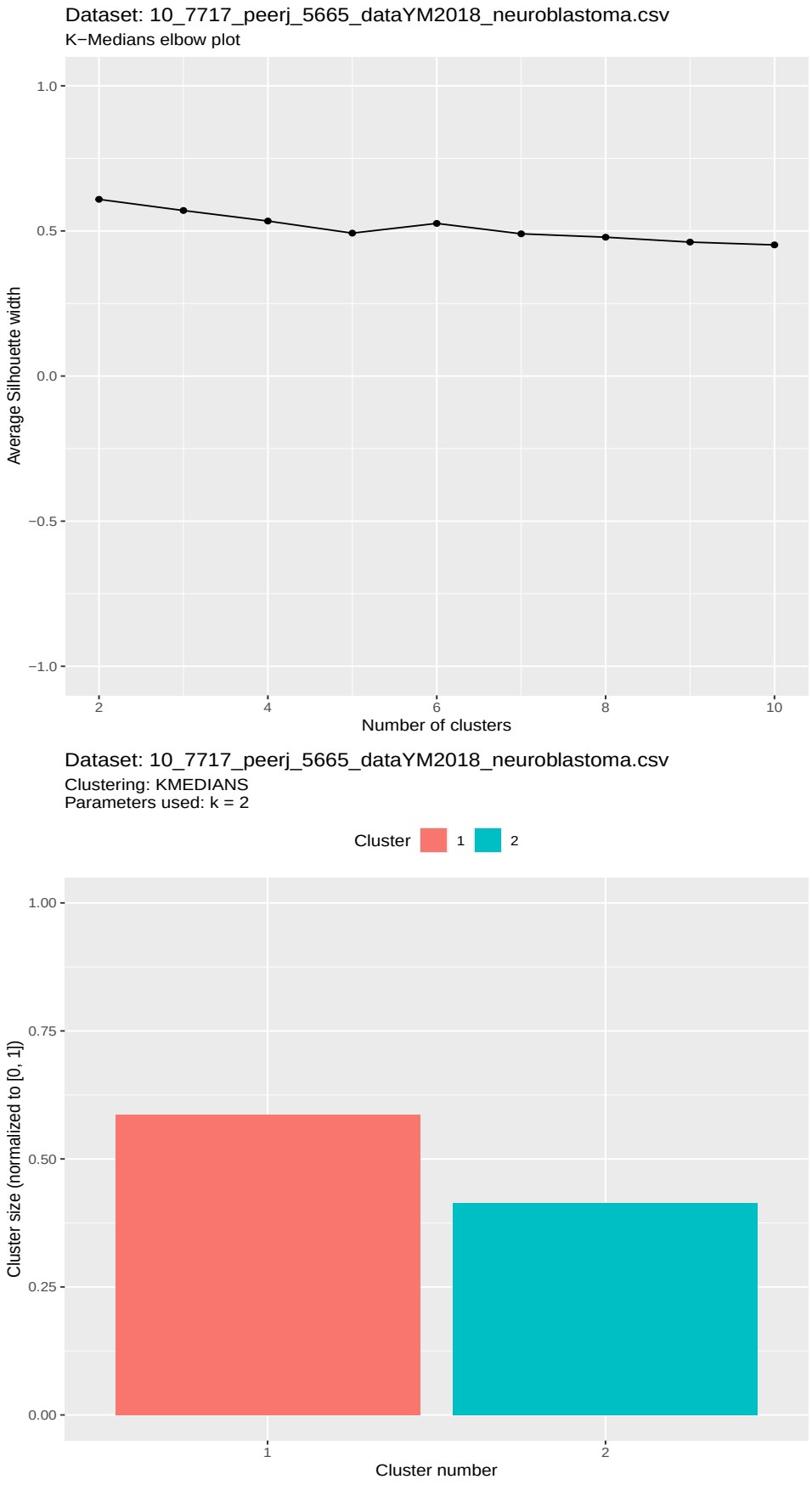


Figura 3.25: Risultati dell'algoritmo K-Medians per il dataset `Neuroblastoma`

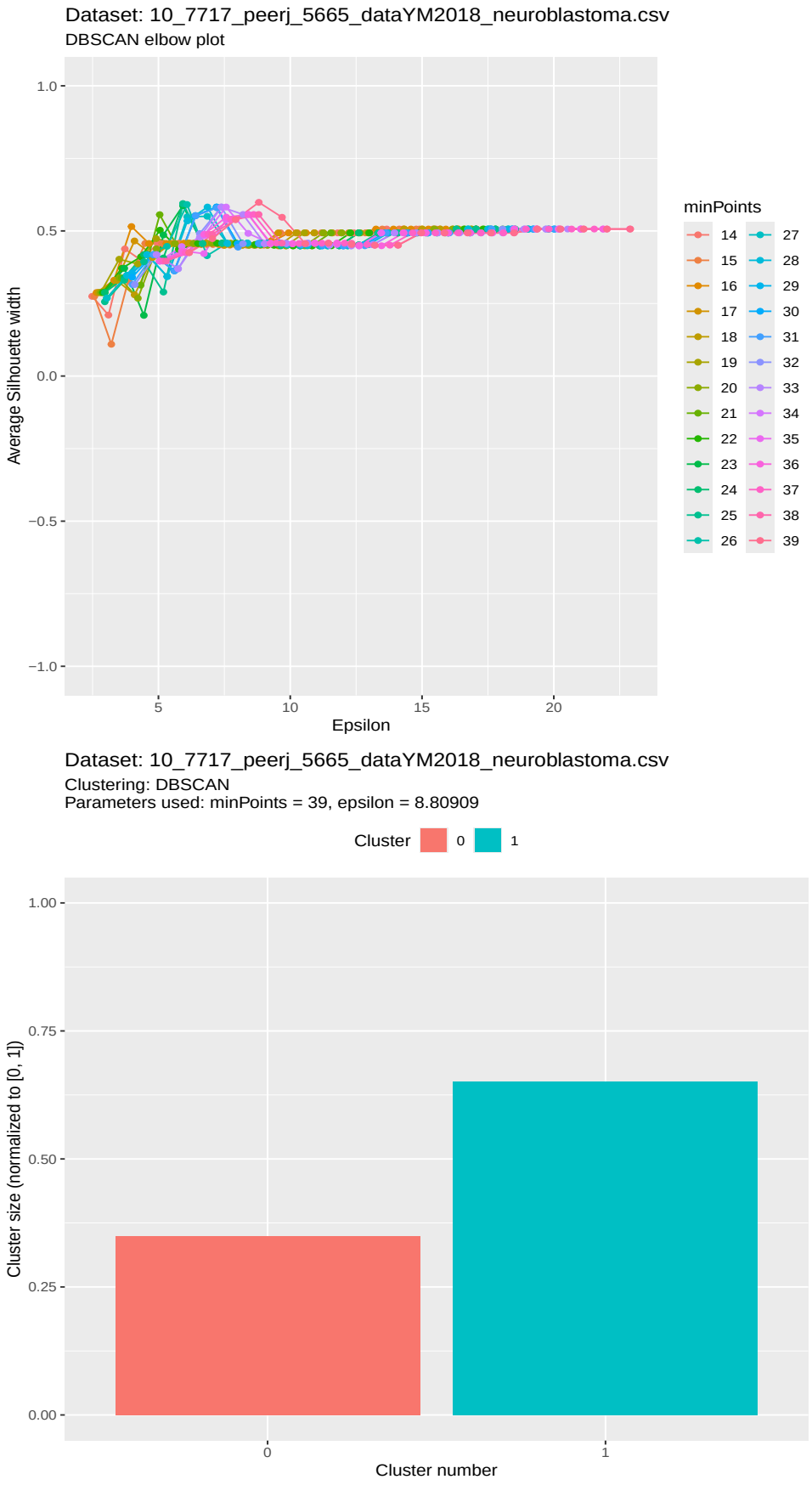


Figura 3.26: Risultati dell'algoritmo DBSCAN per il dataset `Neuroblastoma`

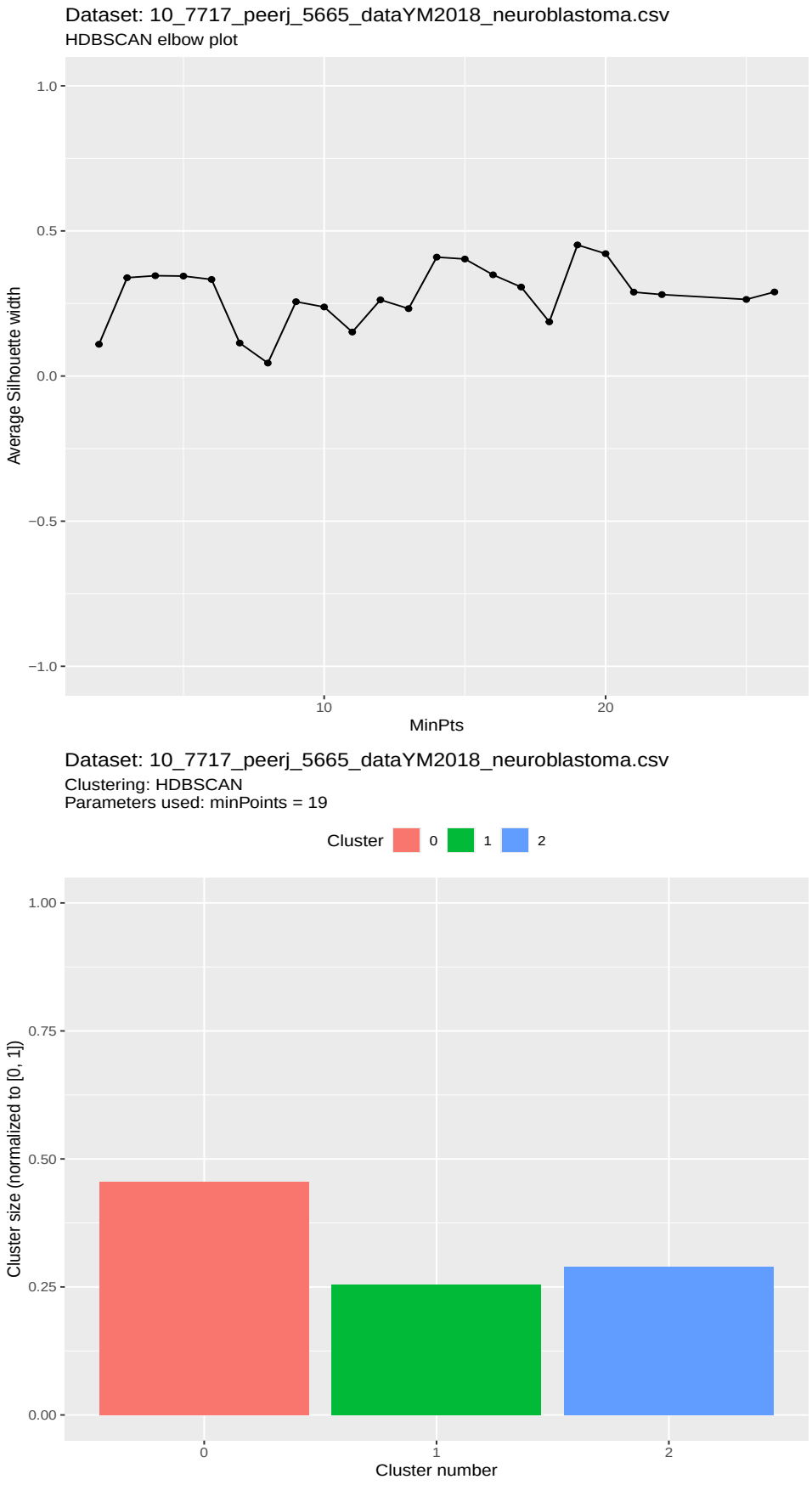


Figura 3.27: Risultati dell'algoritmo HDBSCAN per il dataset `Neuroblastoma`

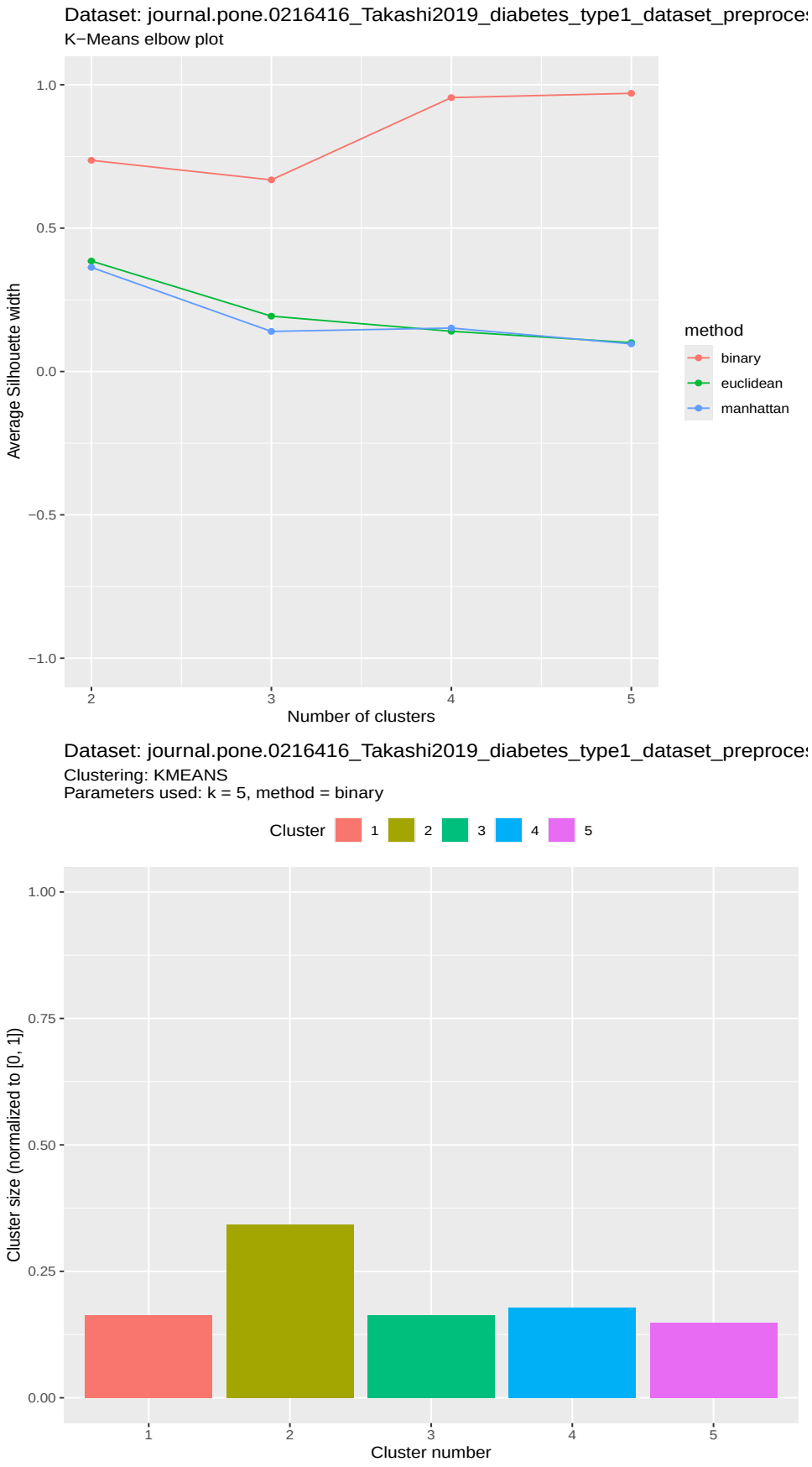


Figura 3.28: Risultati dell'algoritmo K-Means per il dataset `Diabetes`

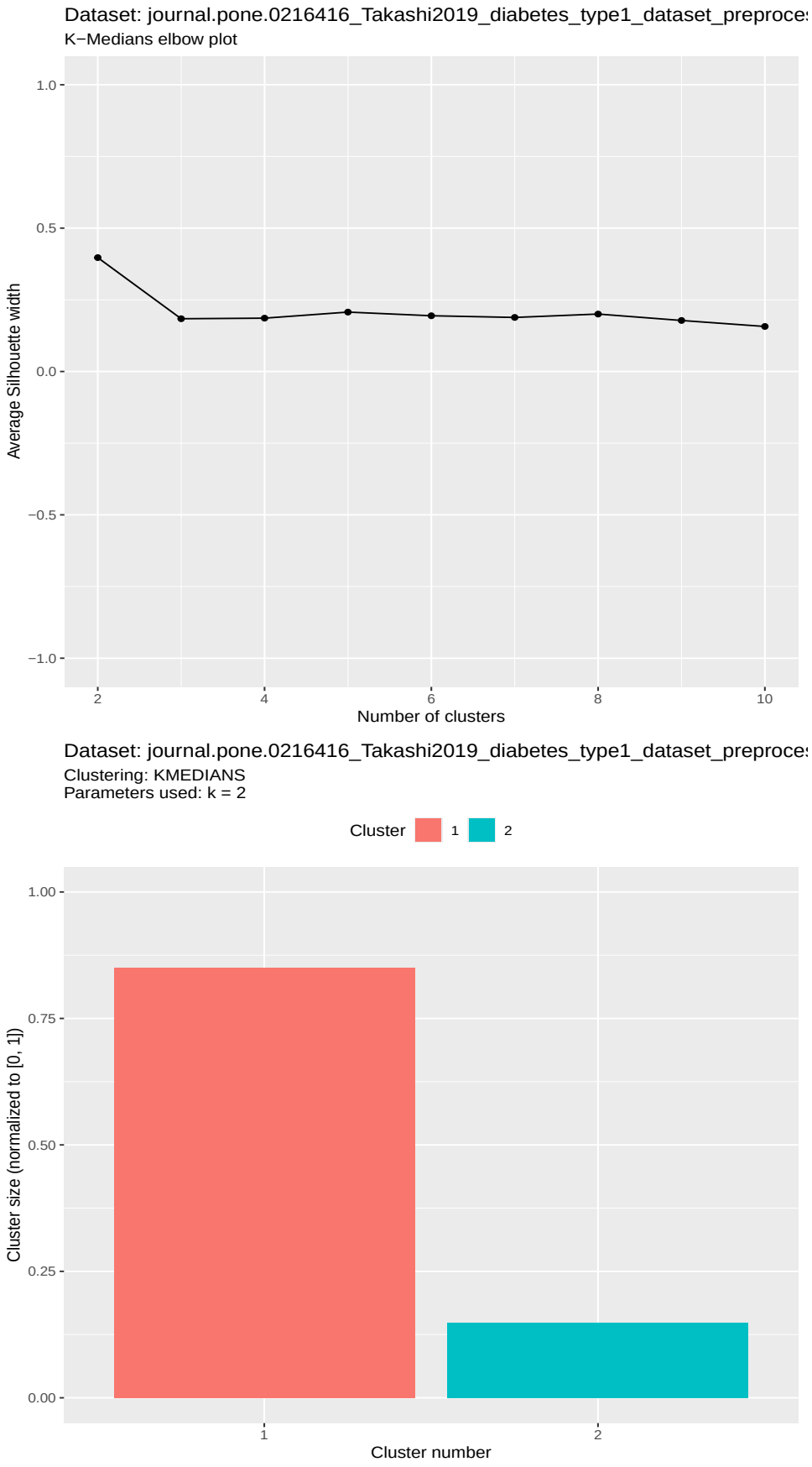


Figura 3.29: Risultati dell'algoritmo K-Medians per il dataset `Diabetes`

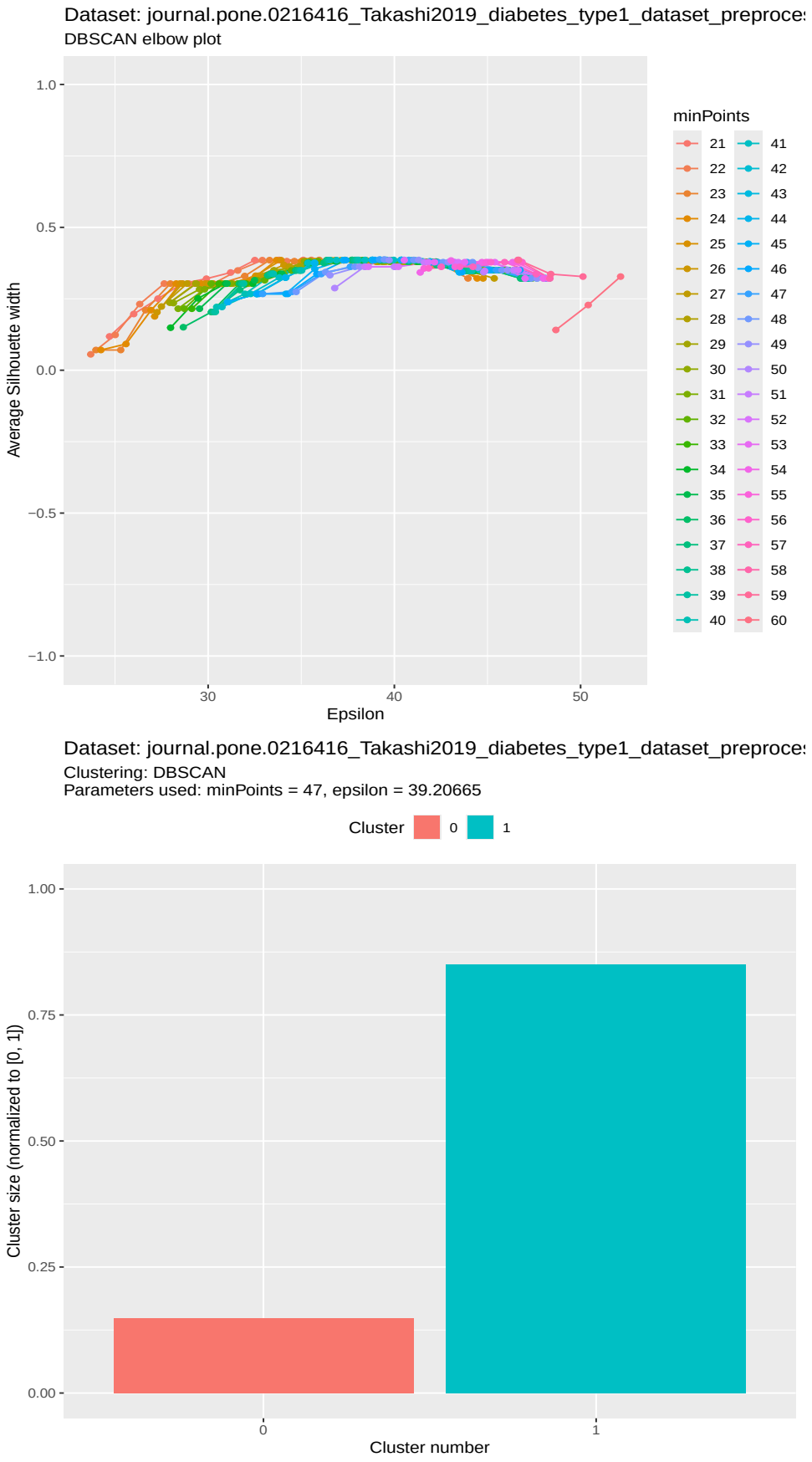


Figura 3.30: Risultati dell'algoritmo DBSCAN per il dataset `Diabetes`

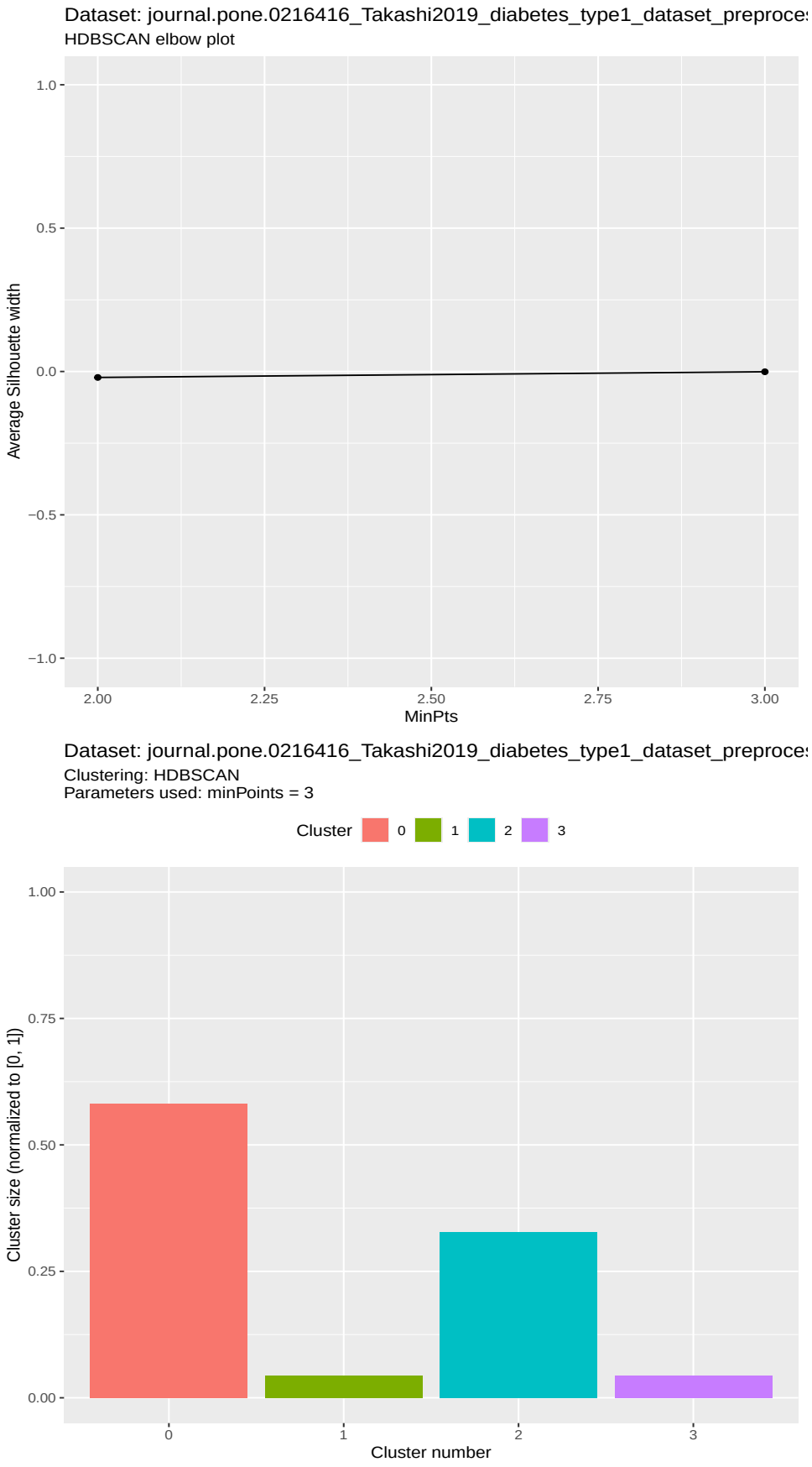


Figura 3.31: Risultati dell'algoritmo HDBSCAN per il dataset `Diabetes`

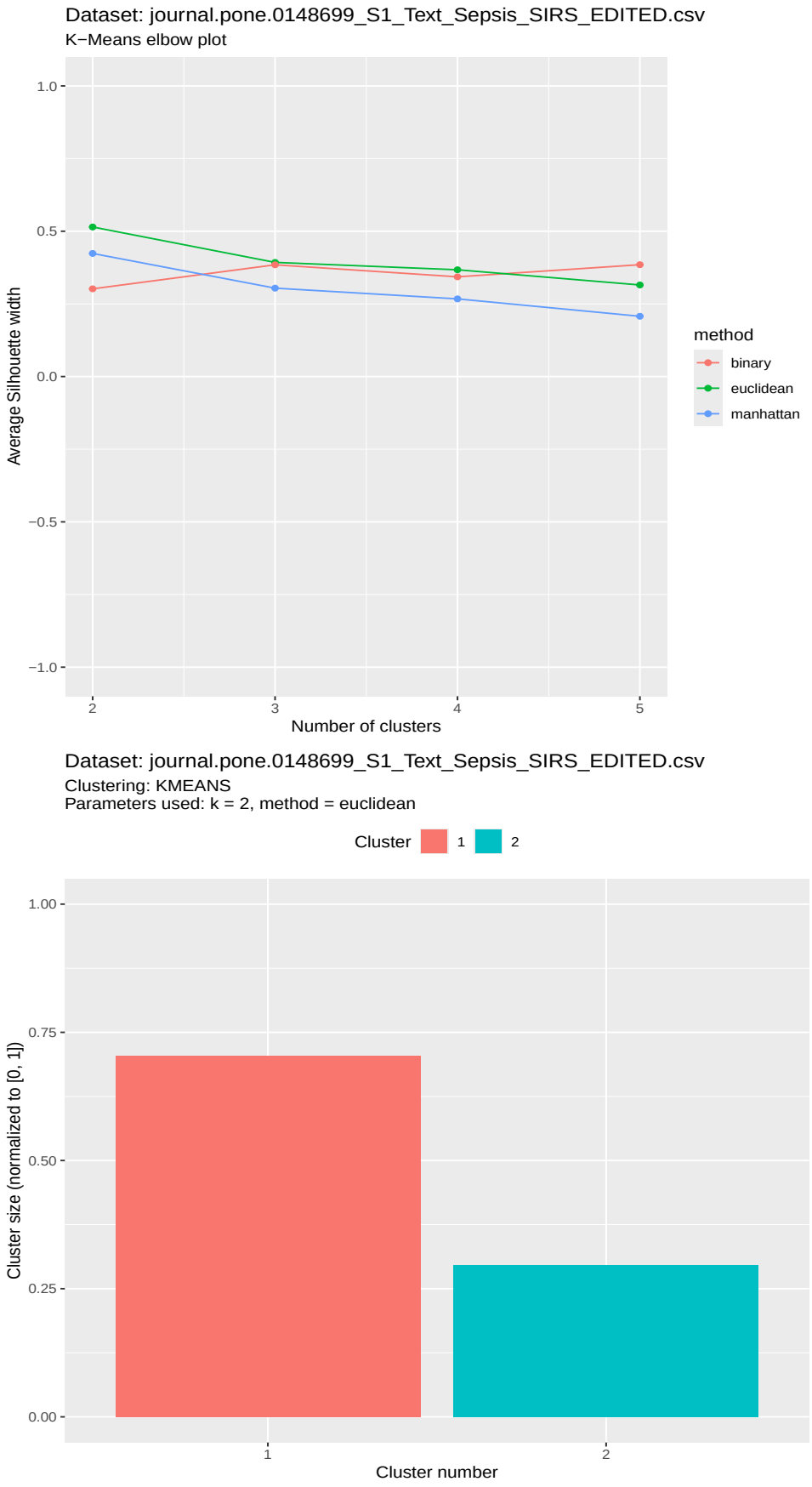


Figura 3.32: Risultati dell'algoritmo K-Means per il dataset `Sepsis`

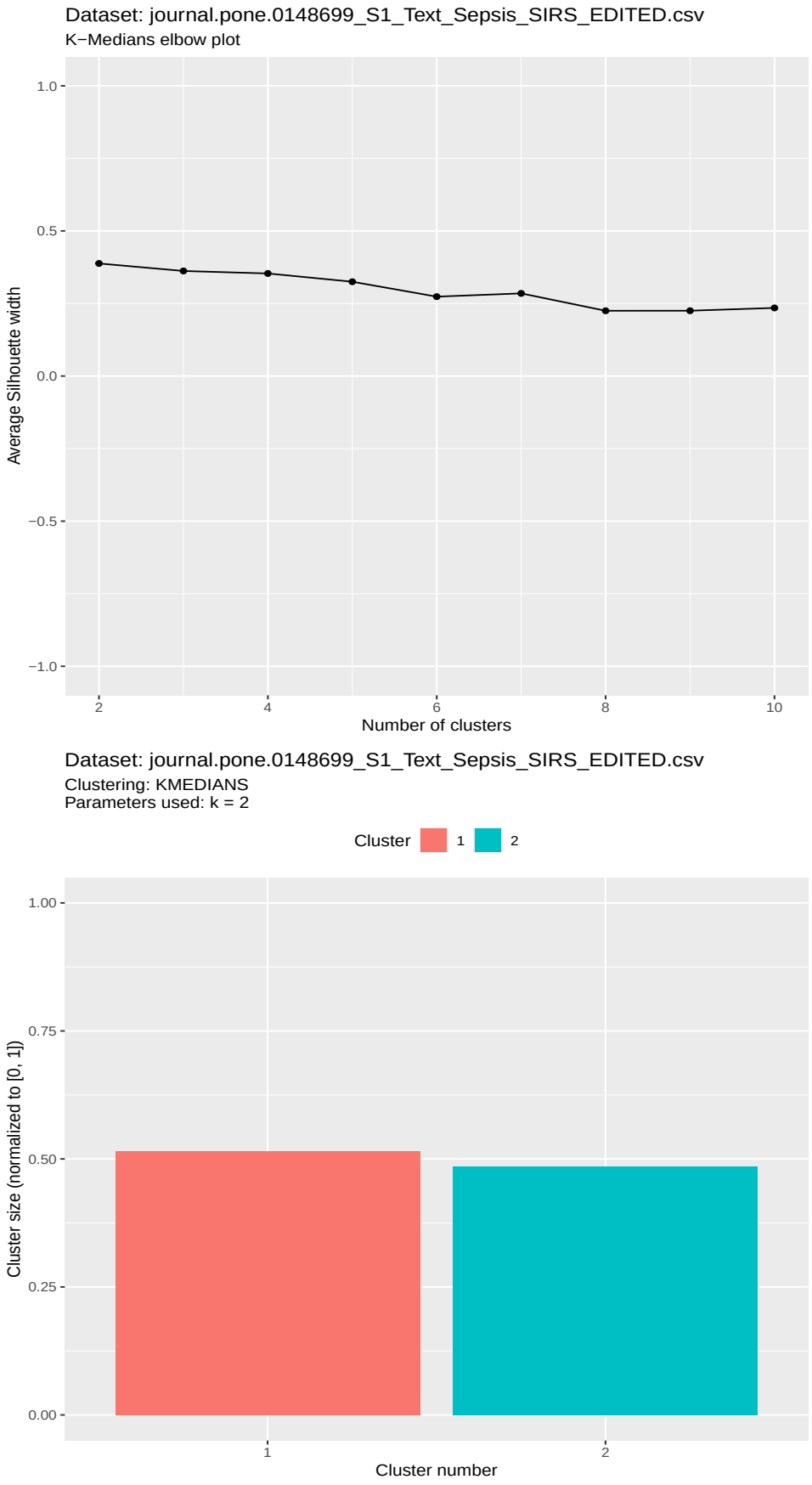


Figura 3.33: Risultati dell'algoritmo K-Medians per il dataset `Sepsis`

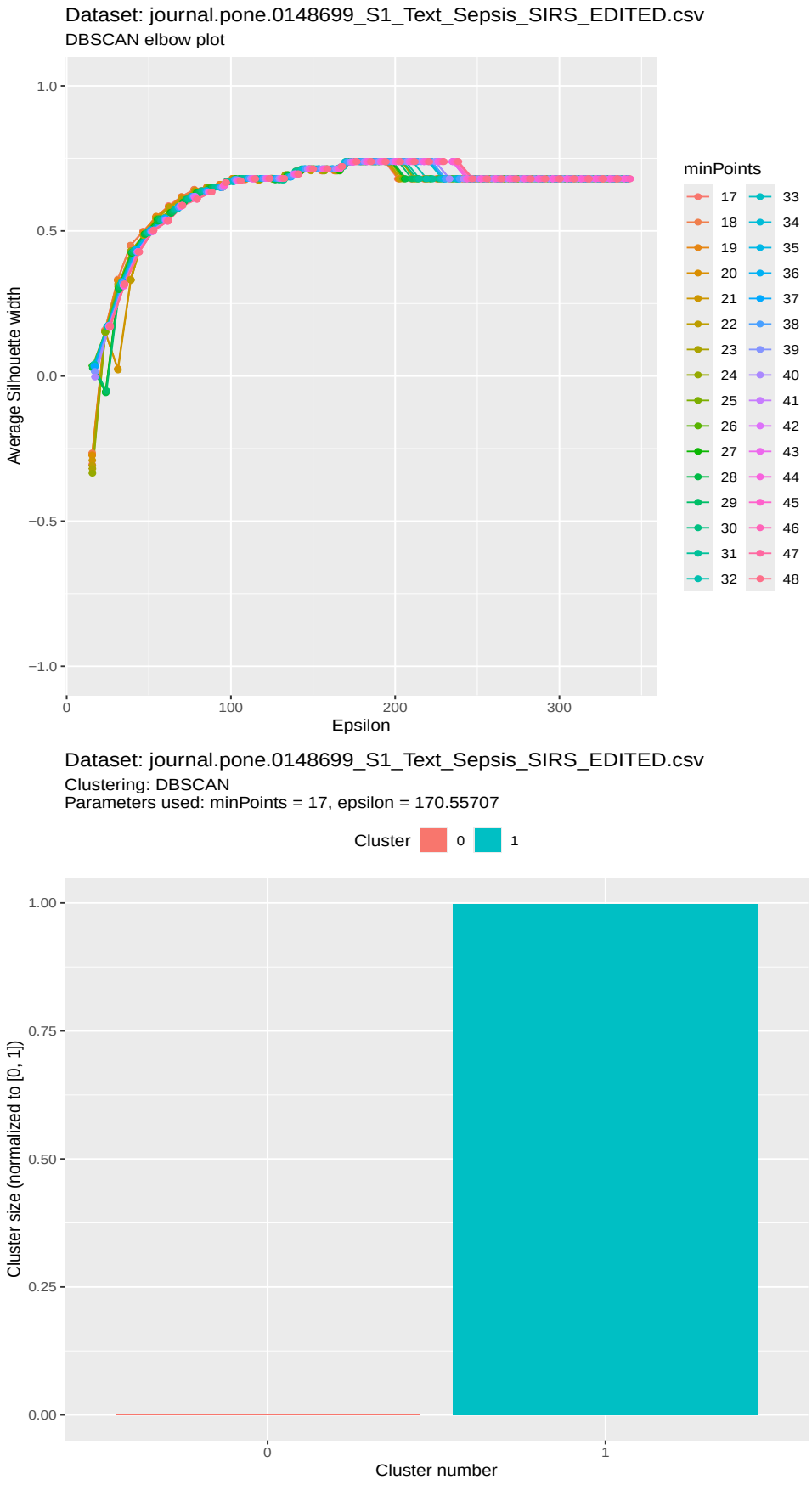


Figura 3.34: Risultati dell'algoritmo DBSCAN per il dataset `Sepsis`

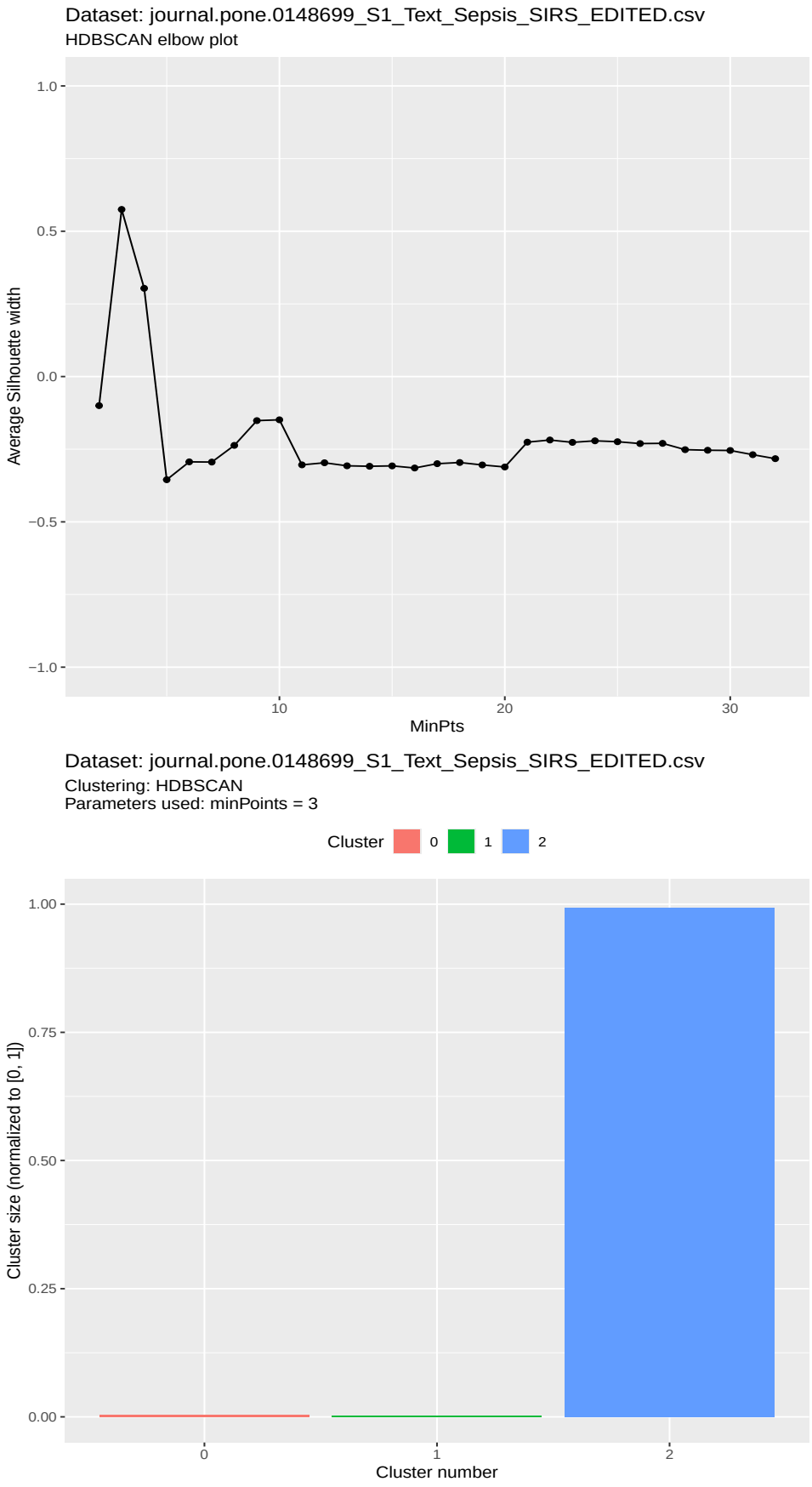


Figura 3.35: Risultati dell'algoritmo HDBSCAN per il dataset `Sepsis`

4. Discussione

4.1 Silhouette

A partire dall'idea di base di Silhouette, è possibile costruirne infinite varianti. Gli autori citano:

- Per determinare il miglior numero di cluster non è strettamente necessario utilizzare la Silhouette media complessiva. Sarebbe infatti possibile anche combinare gli $s(i)$ in modo diverso;
- La Silhouette media complessiva può essere essa stessa usata come funzione obiettivo da massimizzare direttamente all'interno di un algoritmo di clustering, anziché effettuare una valutazione a posteriori;
- Se l'algoritmo di clustering si basa sulla costruzione di centroidi o sulla elezione di rappresentanti, si potrebbe usare la distanza da tali centroidi o rappresentanti come grado di dissomiglianza anziché calcolare $a(i)$ o $D(i,C)$ per ogni i-esimo elemento, semplificando il procedimento. Naturalmente, questo approccio renderebbe Silhouette dipendente dal tipo di algoritmo usato.

Tutte e tre le varianti sono toccate da articoli citati.

4.2 Pacchetti

Il test sanity check non è stato particolarmente conclusivo, perché tutti e cinque i pacchetti hanno fornito valori molto simili (circa 0.9 per il primo dataset e circa 0.4 per il secondo). Questo è un risultato atteso, perché il test era appositamente costruito per escludere pacchetti problematici.

Il test è stato più informativo del precedente, perché i valori restituiti avevano delle differenze evidenziabili. In particolare, `Kira` è stato il pacchetto con le performance peggiori, perché i valori della Silhouette media complessiva sono rimasti pressoché identici. I pacchetti `cluster`, `drclust` e `tidyclust` hanno invece avuto risultati molto simili. In particolare, `cluster` e `tidyclust` hanno avuto risultati perfettamente identici, segno che probabilmente l'uno usa l'altro come subroutine.

Alla luce dei due test considerati, il pacchetto `Kira` è stato immediatamente escluso, perché i risultati forniti dal test della matrice binaria non sono affatto incoraggianti. Dato che i pacchetti `cluster` e `tidyclust` hanno fornito un risultato identico, fra i due è stato preferito `cluster`, perché fra i due era quello con il tempo di esecuzione più basso. Il pacchetto `scikit-learn` attraverso `reticulate` non è stato preso in considerazione, perché era stato incluso semplicemente come metro di paragone.

Fra `cluster` e `drclust` ho preferito scegliere `cluster`. Questo sia perché il tempo di esecuzione è inferiore, sia perché il pacchetto `cluster` si trova spesso già incluso nelle installazioni di `R` (garanzia di affidabilità) sia perché è l'unico pacchetto il cui input non dipende dall'algoritmo usato. Infatti, `cluster` ha in input il risultato di un qualsiasi algoritmo di clustering ed una matrice delle distanze, mentre gli altri pacchetti richiedono in input espressamente il risultato dell'applicazione di una loro implementazione di un algoritmo.

In Figura 4.1 è presentata la differenza fra `cluster` e `scikit-learn` sul test matrice binaria. Come è possibile notare, la differenza fra i due è contenuta.

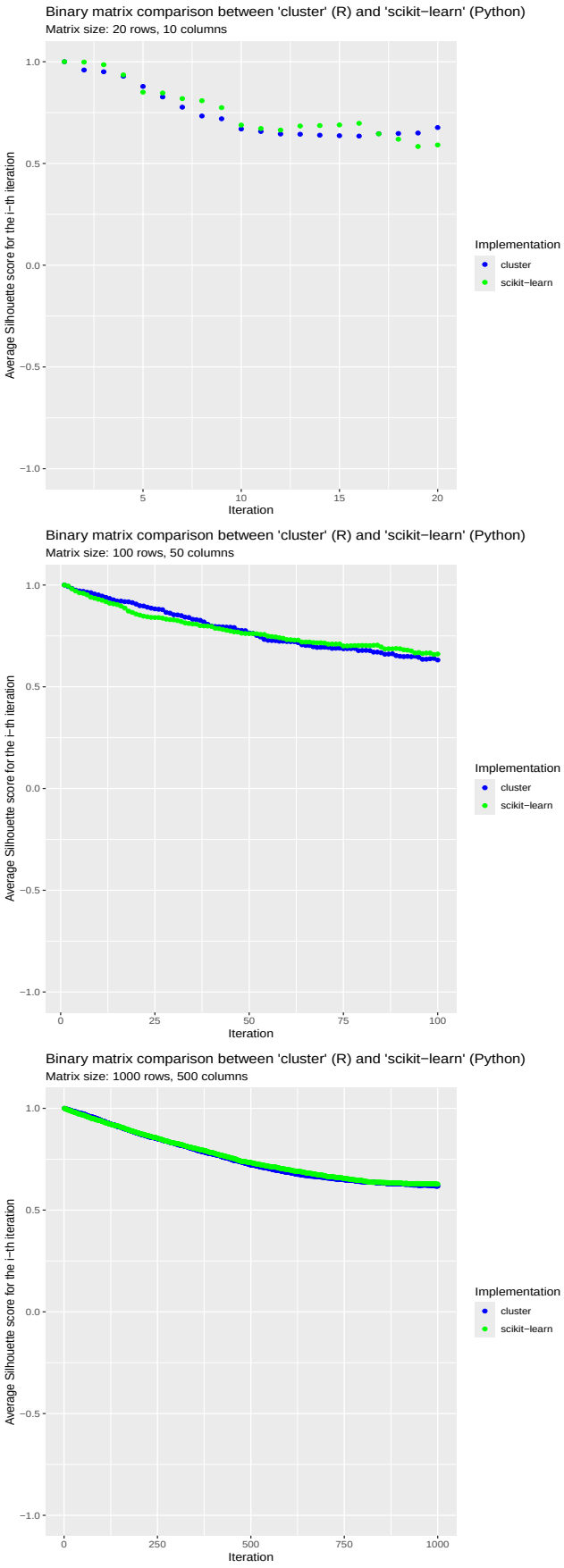


Figura 4.1: Differenze fra il test matrice binaria tra `cluster` (pacchetto R) e `scikit-learn` (pacchetto Python).

4.3 EHR

Come è possibile notare nei plot mostrati in precedenza, l'algoritmo scelto influenza notevolmente il numero di cluster che vengono individuati, pure se in ogni caso si tratta di iperparametri massimizzati usando Silhouette. Questo è in accordo con gli articoli citati. Di seguito sono riportate le performance degli algoritmi di clustering.

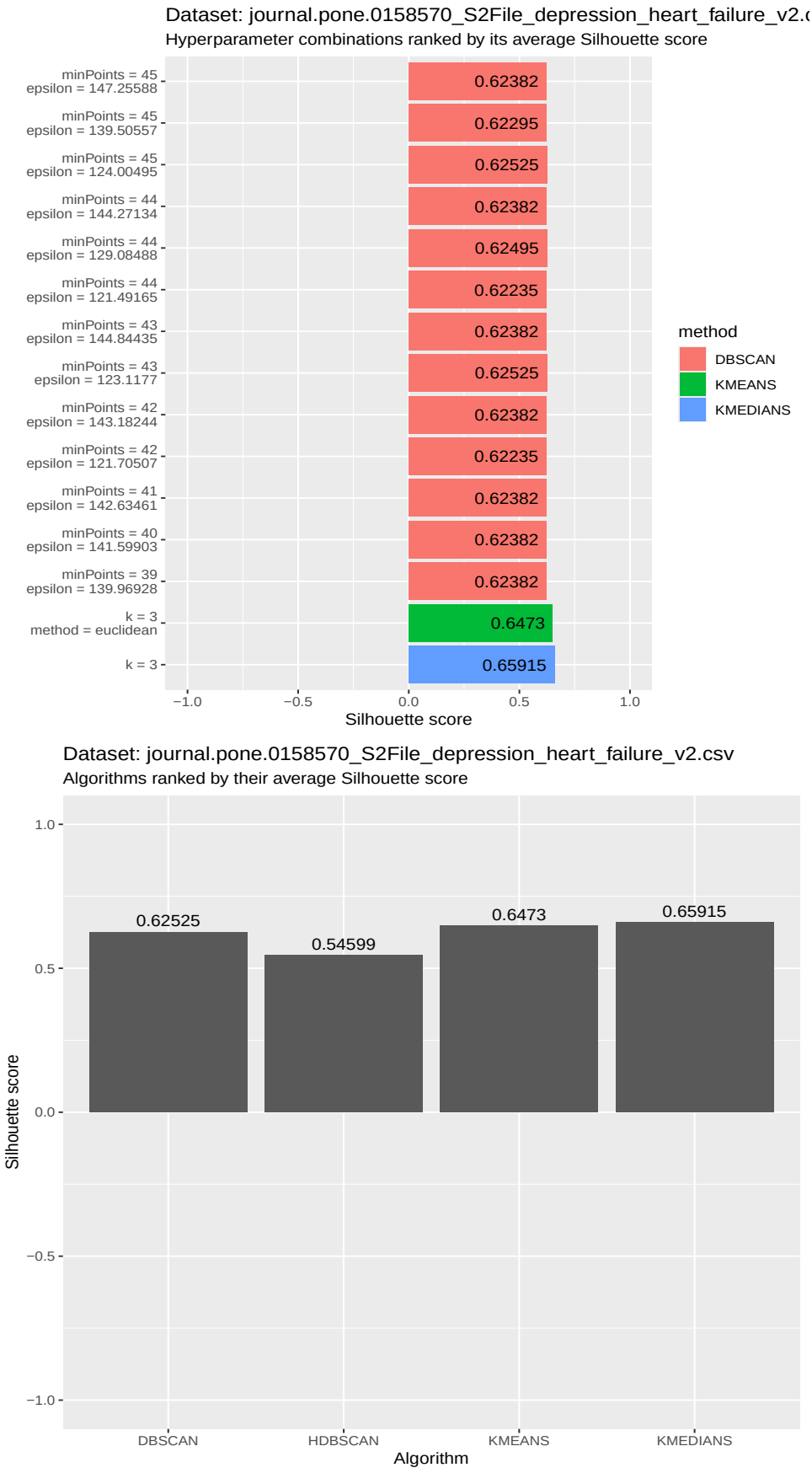


Figura 4.2: Riassunto dei risultati dei vari algoritmi per il dataset `HeartFailure`

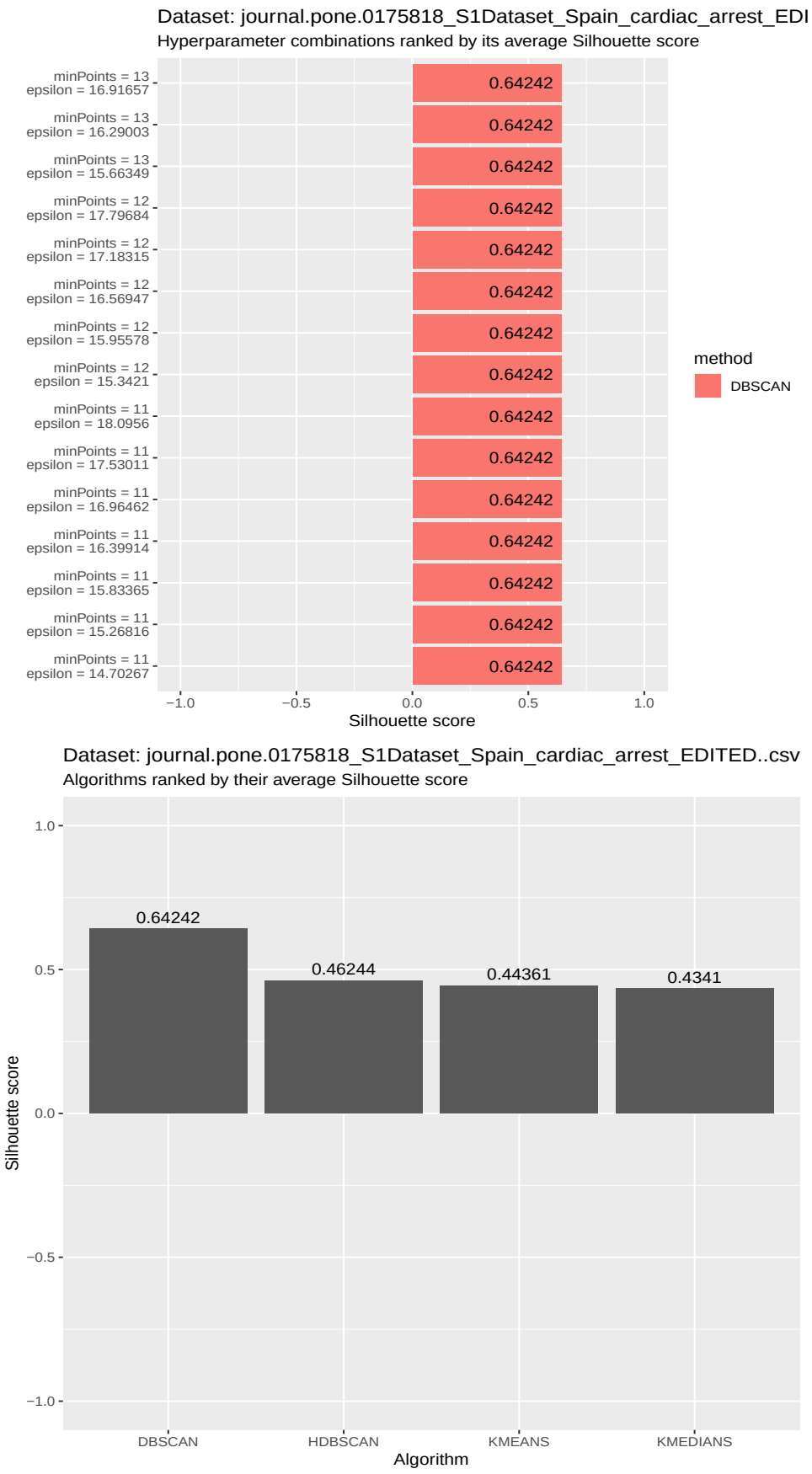


Figura 4.3: Riassunto dei risultati dei vari algoritmi per il dataset `CardiacArrest`

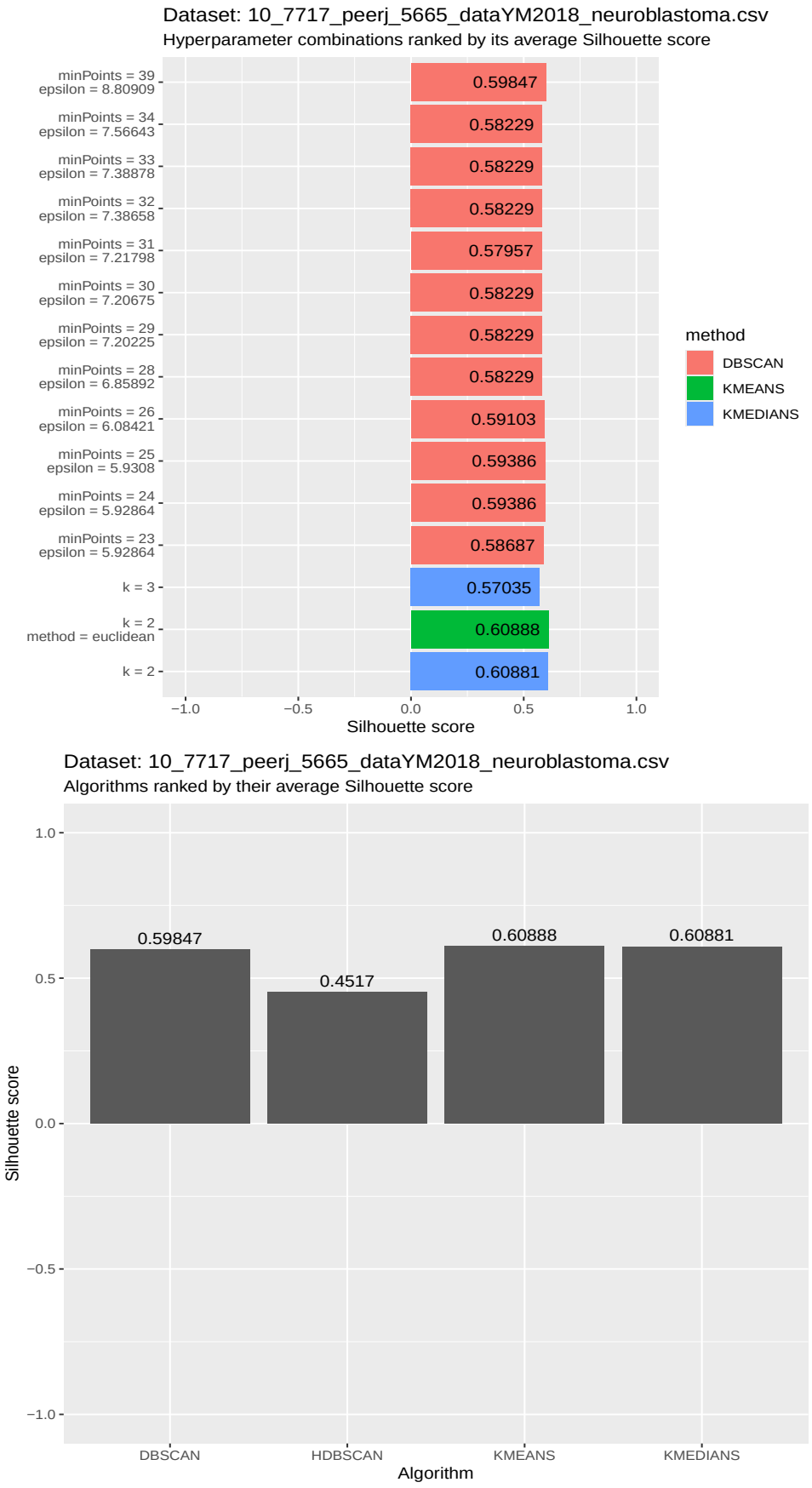


Figura 4.4: Riassunto dei risultati dei vari algoritmi per il dataset `Neuroblastoma`

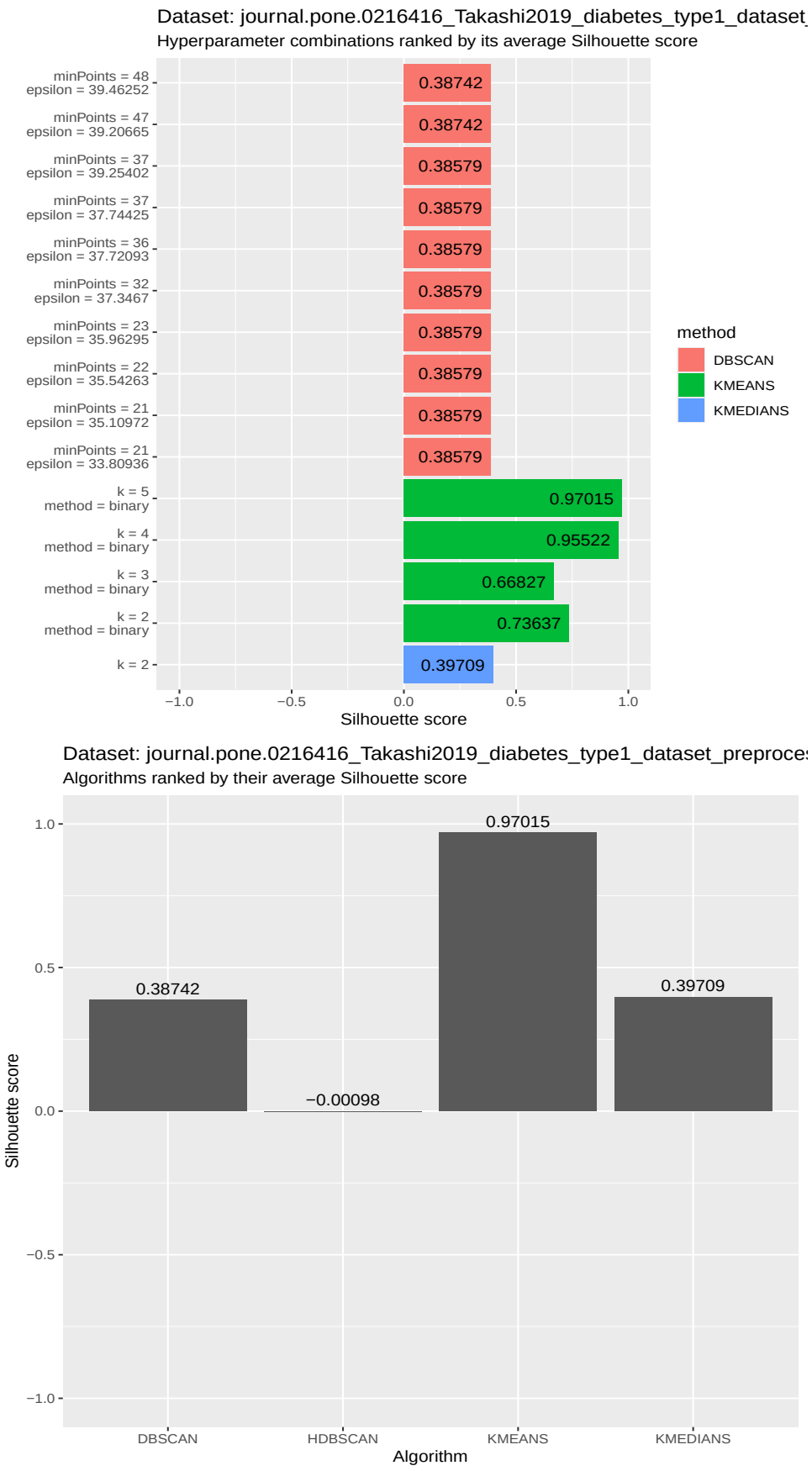


Figura 4.5: Riassunto dei risultati dei vari algoritmi per il dataset `Diabetes`

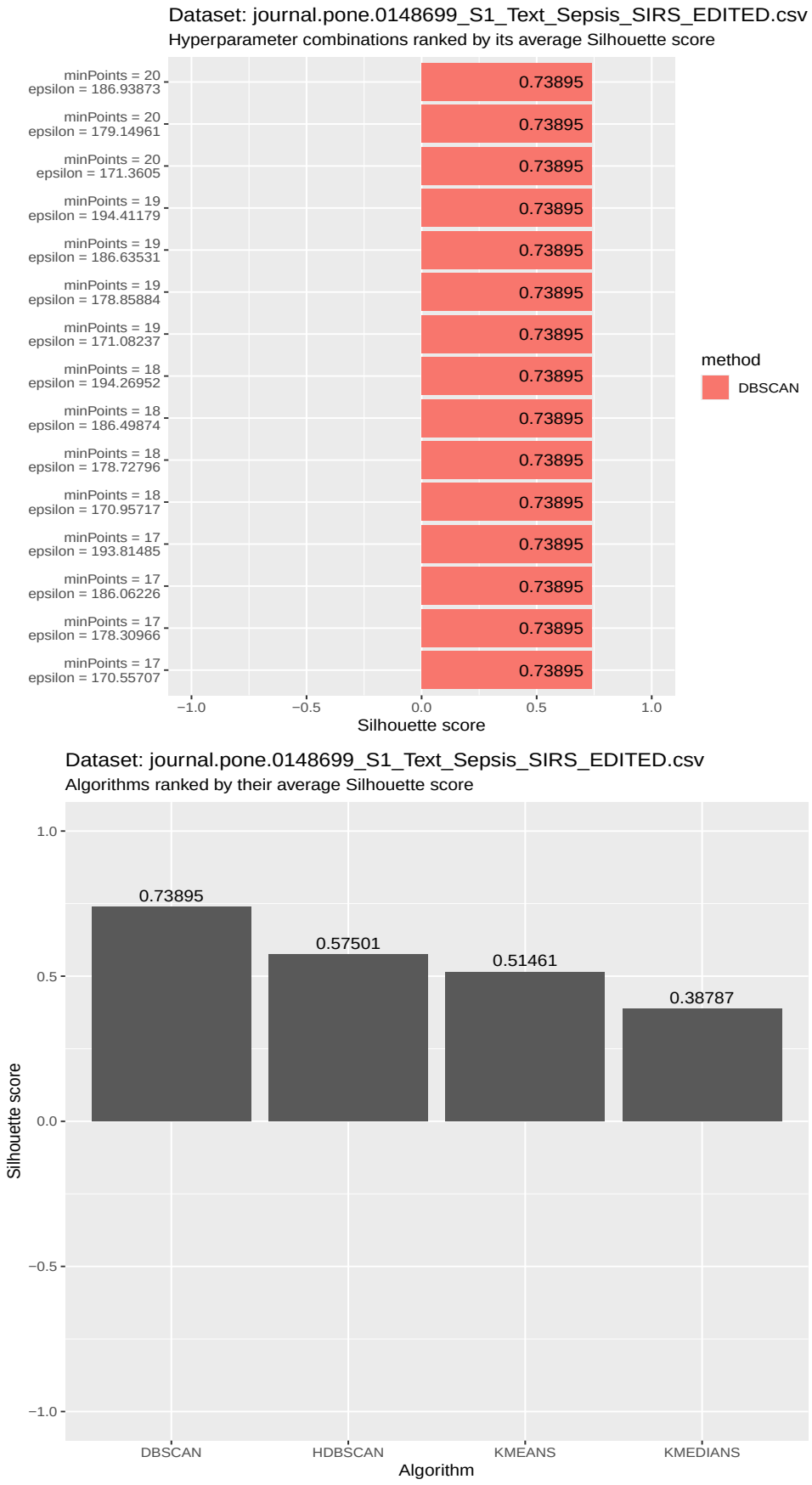


Figura 4.6: Riassunto dei risultati dei vari algoritmi per il dataset `Sepsis`

DBSCAN figura spesso come algoritmo dalle alte performance. Per tale motivo mi sono chiesto se fosse possibile testare le prestazioni di Silhouette su DBSCAN comparandolo con un metodo di

ottimizzazione degli iperparametri alternativo, e vedere se i due risultati sono simili.

Fissato MinPts come il doppio più uno delle dimensioni del dataset, il valore di ε può essere stimato costruendo un **KNN plot**: fissato $k = \text{MinPts} - 1$, lungo l'asse delle ascisse si riportano gli elementi ordinati in ordine crescente per distanza dal loro k-esimo vicino, mentre sull'asse delle ordinate la distanza stessa. In genere, una curva costruita sulla base di questi dati ha inizialmente un andamento stabile per poi avere una crescita rapida: il valore di ε è scelto il punto della curva in cui si ha tale variazione di pendenza.

Come è possibile apprezzare nelle figure successive, i valori di ε così restituiti inducono un clustering che è molto simile a quello fornito utilizzando Silhouette. Questo indica che Silhouette è effettivamente in grado di restituire una combinazione ottimale di iperparametri.

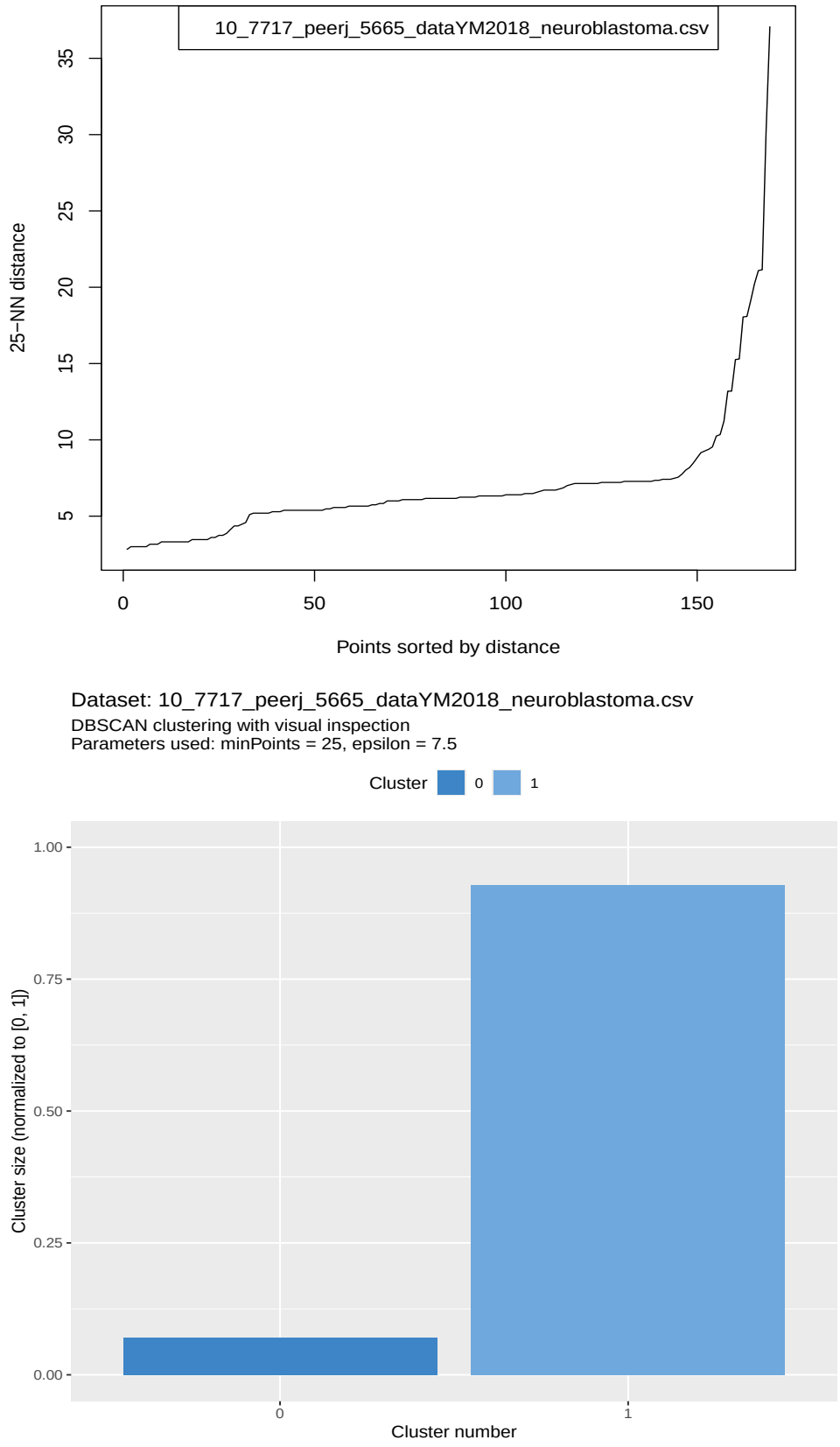


Figura 4.7: Risultati dell'algoritmo DBSCAN per il dataset `HeartFailure`, usando un KNN-plot per stimare ε

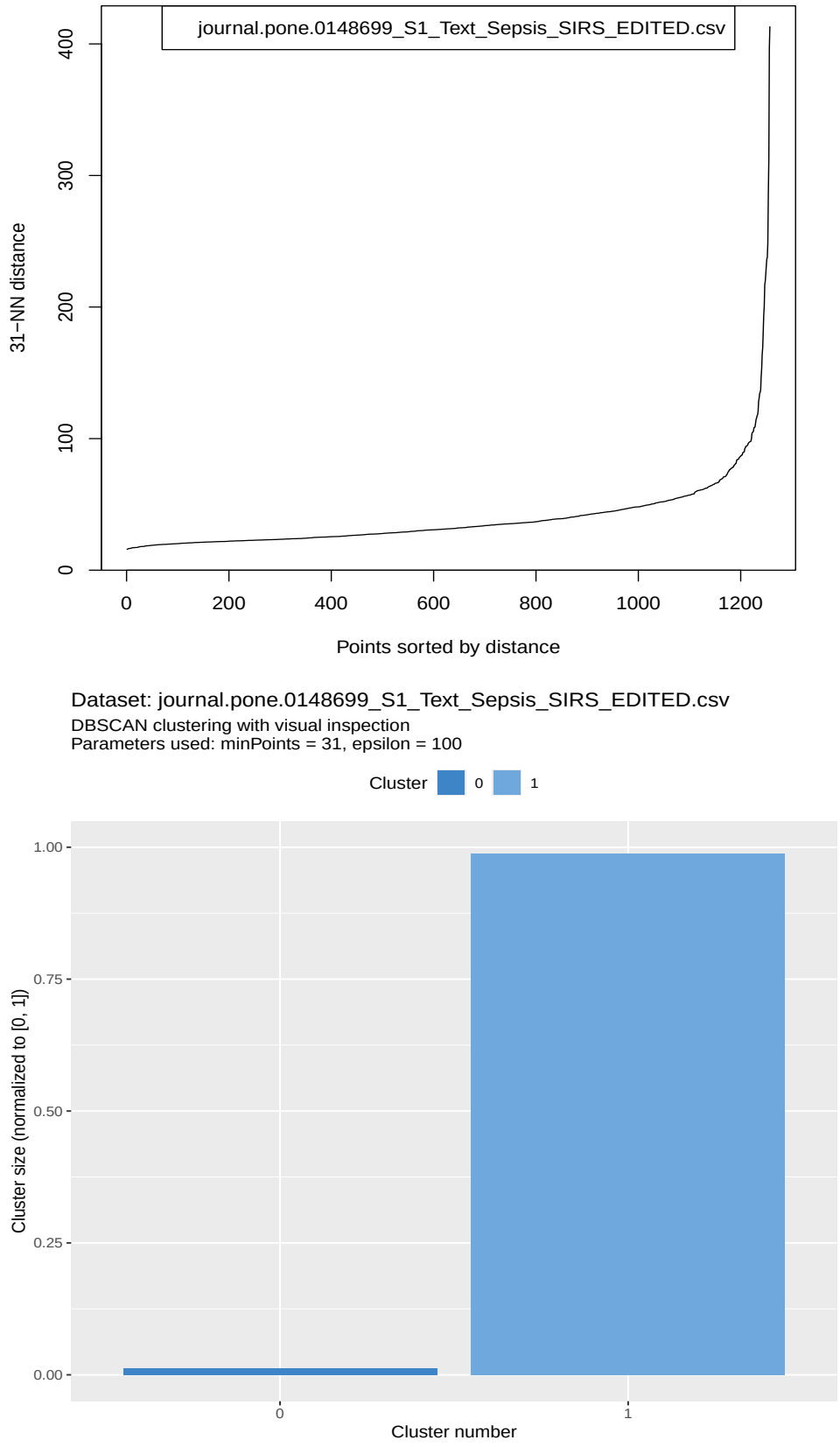


Figura 4.8: Risultati dell'algoritmo DBSCAN per il dataset `CardiacArrest`, usando un KNN-plot per stimare ε

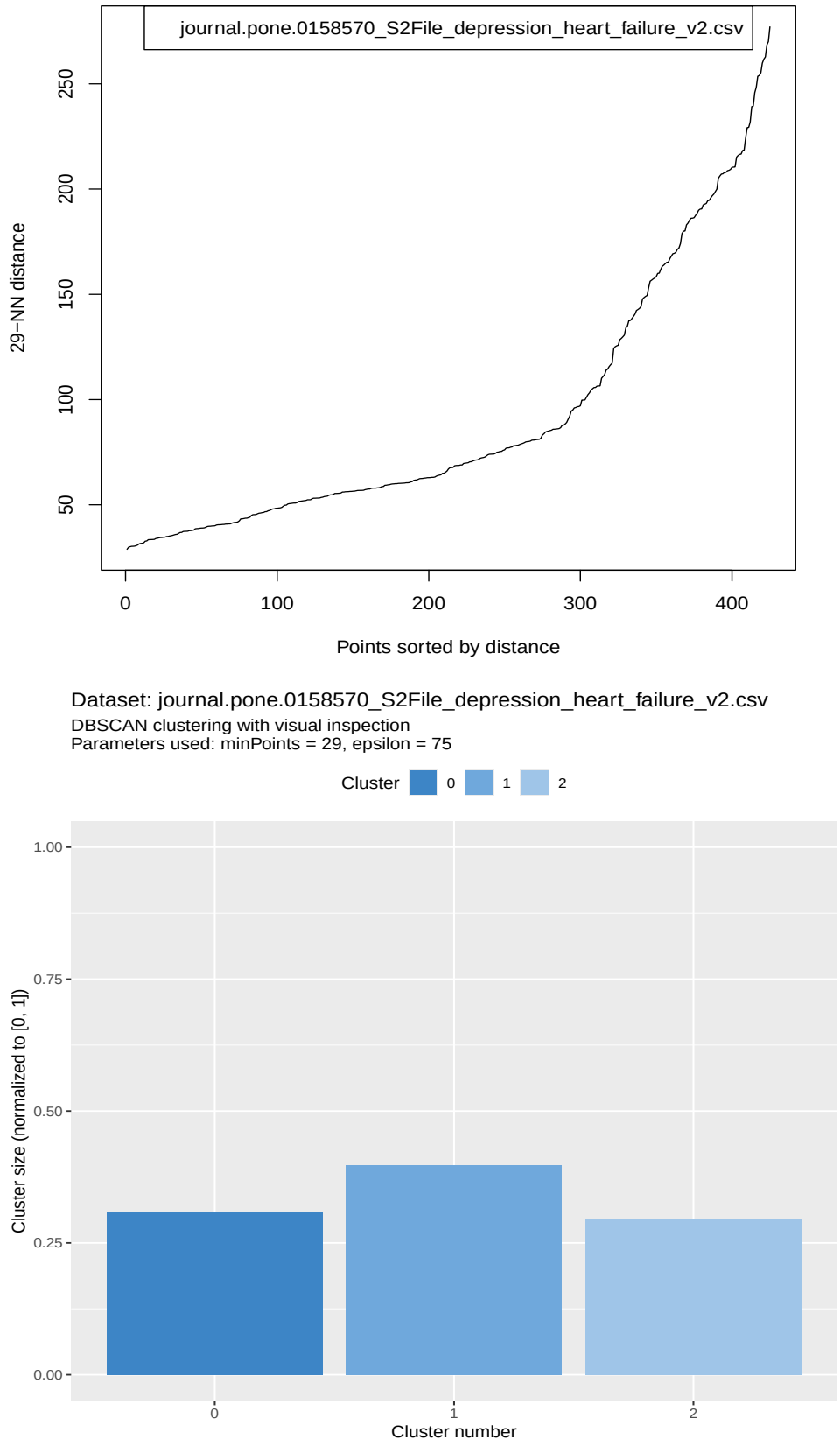


Figura 4.9: Risultati dell'algoritmo DBSCAN per il dataset `Neuroblastoma`, usando un KNN-plot per stimare ε

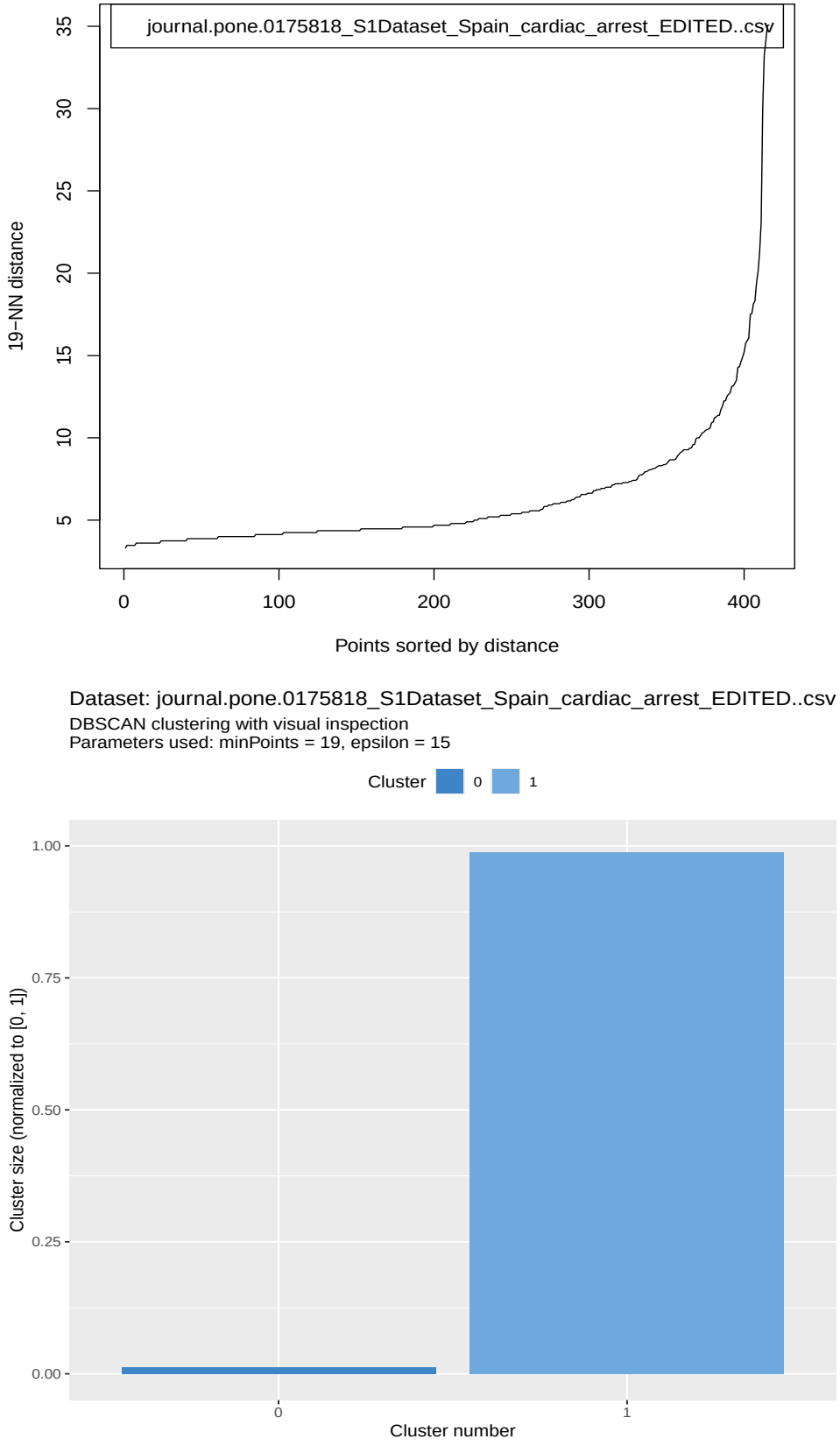


Figura 4.10: Risultati dell'algoritmo DBSCAN per il dataset `Diabetes`, usando un KNN-plot per stimare ε

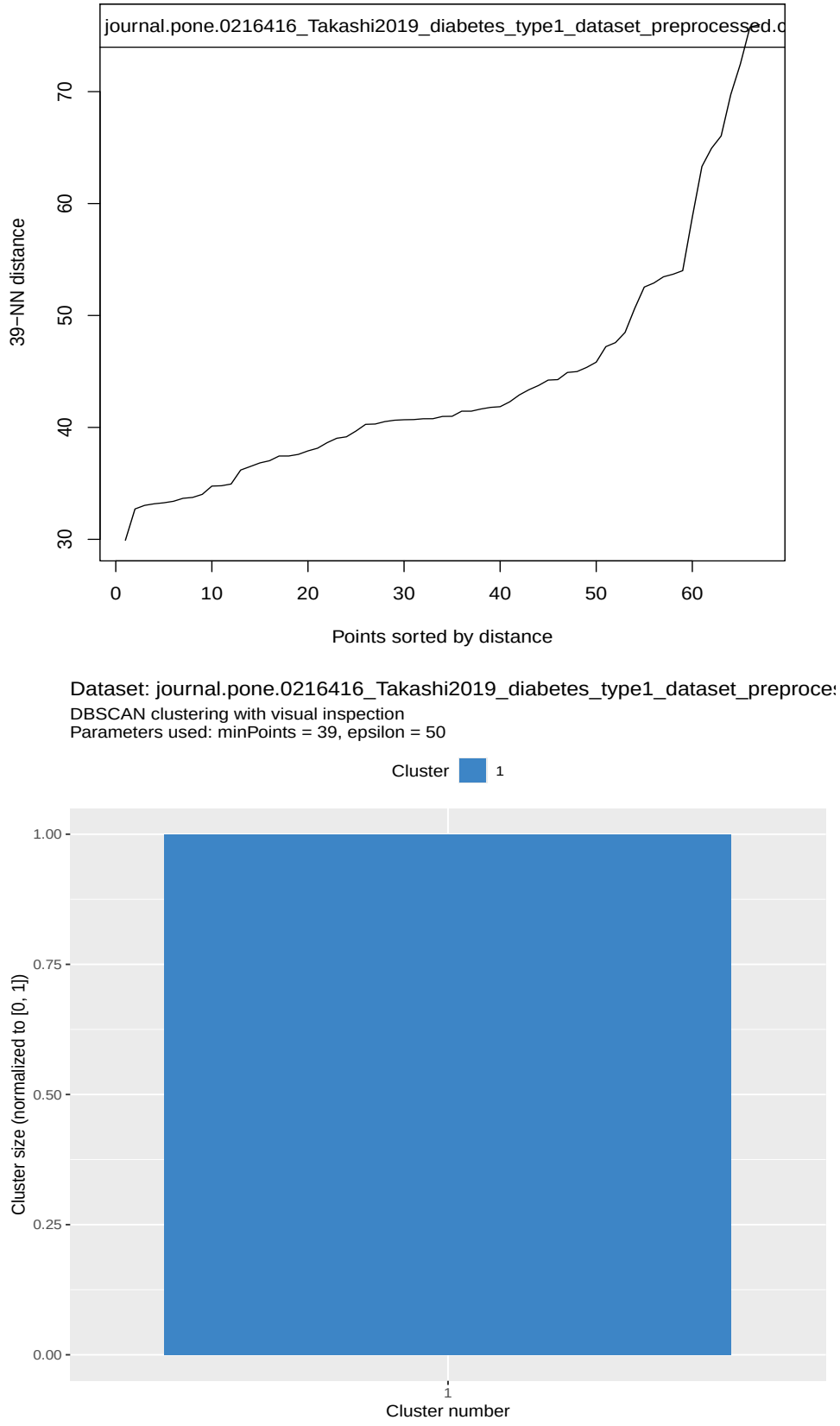


Figura 4.11: Risultati dell'algoritmo DBSCAN per il dataset `Sepsis`, usando un KNN-plot per stimare ε

5. Conclusioni

Data la natura di questa tesi, é evidente come questo non possa essere altro che un lavoro esplorativo. Le idee qui presentate potrebbero essere estese in diversi modi, alcuni riportati di seguito.

Innanzitutto, i test sono stati condotti usando esclusivamente EHR come dataset. Sebbene queste contengano moltissime informazioni, non sono una risorsa omnicomprensiva. Si potrebbe pertanto ripetere i test utilizzando anche o soltanto dataset di dati biomedici che contengono dati che le EHR ignorano, dando un quadro piú completo.

Inoltre, gli algoritmi di clustering sono stati applicati su dataset dove parte degli elementi avevano uno o più attributi di valore ignoto. Non essendo possibile operare il clustering con dati dei quali non è noto il valore, l'approccio che ho utilizzato è semplicemente stato quello di eliminare ogni elemento che avesse almeno un dato mancante. Si potrebbe ripetere gli esperimenti adottando un approccio più conservativo, ad esempio sostituendo tali dati con valori di default ed osservare se si presentano differenze.

Un'alternativa simile è quella di utilizzare tecniche in grado di predire i valori mancanti sulla base di quelli noti, ad esempio calcolando la media di tutti i valori noti per un certo attributo ed utilizzarla al posto dei valori mancanti.

Infine, sarebbe interessante ripetere gli esperimenti operando **dimensionality reduction**, come ad esempio **principal component analysis** [13], ovvero tecniche in grado di accorpare o di scartare del tutto gli attributi che non hanno particolare rilevanza nel risultato finale del clustering. Questo permetterebbe, ammesso di riuscire a ridurre il numero delle dimensioni fino a due o a tre, di poter visualizzare il risultato del clustering in uno scatter plot, a prescindere dal numero originale delle dimensioni.

Disponibilitá del codice

Il codice per la generazione dei test e per l'applicazione degli algoritmi di clustering ai dataset EHR è liberamente disponibile. Il codice in questione puó essere reperito al seguente link: `https://github.com/Sk`

Ho organizzato il codice seguendo le linee guida riportate in [18]. Ho anche tenuto un lab notebook per tenere traccia dei risultati mano mano che venivano generati, facendo riferimento a [20].

Bibliografia

[1] Nonie Alexander et al. «Identifying and Evaluating Clinical Subtypes of Alzheimer's Disease in Care Electronic Health Records Using Unsupervised Machine Learning». In: *BMC Medical Informatics and Decision Making* 21.1 (2021), p. 343. ISSN: 1472-6947. DOI: 10.1186/s12911-021-01693-6. URL: https://doi.org/10.1186/s12911-021-01693-6.

[2] T. Caliński e J Harabasz. «A dendrite method for cluster analysis». In: *Communications in Statistics* 3.1 (1974), pp. 1–27. DOI: 10.1080/03610927408827101. eprint: https://www.tandfonline.com/doi/pdf/10.1080/03610927408827101. URL: https://www.tandfonline.com/doi/abs/10.1080/03610927408827101.

[3] Ricardo J. G. B. Campello, Davoud Moulavi e Joerg Sander. «Density-Based Clustering Based on Hierarchical Density Estimates». In: *Advances in Knowledge Discovery and Data Mining*. A cura di Jian Pei et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 160–172. ISBN: 978-3-642-37456-2.

[4] Chia-Wei Chang et al. «Identifying Heterogeneous Subgroups of Systemic Autoimmune Diseases by Applying a Joint Dimension Reduction and Clustering Approach to Immunomarkers». In: *BioData Mining* 17.1 (2024), p. 36. ISSN: 1756-0381. DOI: 10.1186/s13040-024-00389-7. URL: https://doi.org/10.1186/s13040-024-00389-7.

[5] Kumardeep Chaudhary et al. «Utilization of Deep Learning for Subphenotype Identification in Sepsis-Associated Acute Kidney Injury». In: *Clinical Journal of the American Society of Nephrology* 15.11 (2020). ISSN: 1555-9041. URL: https://journals.lww.com/cjasn/fulltext/2020/11000/utilization_of_deep_learning_for_subphenotype.6.aspx.

[6] David L. Davies e Donald W. Bouldin. «A Cluster Separation Measure». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.2 (1979), pp. 224–227. DOI: 10.1109/TPAMI.1979.4766909.

[7] Andrzej Dudek. «Silhouette Index as Clustering Evaluation Tool». In: *Classification and Data Analysis*. A cura di Krzysztof Jajuga, Jacek Batóg e Marek Walesiak. Cham: Springer International Publishing, 2020, pp. 19–33. ISBN: 978-3-030-52348-0.

[8] J. C. Dunn†. «Well-Separated Clusters and Optimal Fuzzy Partitions». In: *Journal of Cybernetics* 4.1 (1974), pp. 95–104. DOI: 10.1080/01969727408546059. eprint: https://doi.org/10.1080/01969727408546059. URL: https://doi.org/10.1080/01969727408546059.

[9] Martin Ester et al. «A density-based algorithm for discovering clusters in large spatial databases with noise». In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[10] R. A. FISHER. «THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS». In: *Annals of Eugenics* 7.2 (1936), pp. 179–188. DOI: https://doi.org/10.1111/j.1469-1809.1936.tb02137.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1469-1809.1936.tb02137.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x.

[11] Xiaonan Gao e WU Sen. «CUBOS: An Internal Cluster Validity Index for Categorical Data». In: *Tehnicki vjesnik - Technical Gazette* (2019). URL: https://api.semanticscholar.org/CorpusID:198189126.

[12] L. Guerra et al. «A comparison of clustering quality indices using outliers and noise». In: *Intelligent Data Analysis* 16.4 (2012), pp. 703–715. DOI: 10.3233/IDA-2012-0545. eprint: https://doi.org/10.3233/IDA-2012-0545. URL: https://doi.org/10.3233/IDA-2012-0545.

[13] Harold Hotelling. «Analysis of a complex of statistical variables into principal components.» In: *Journal of Educational Psychology* 24 (1933), pp. 498–520. URL: https://api.semanticscholar.org/CorpusID:144828484.

[14] Sookyung Hyun et al. «Exploration of critical care data by using unsupervised machine learning». In: *Computer Methods and Programs in Biomedicine* 194 (2020), p. 105507. ISSN: 0169-2607. DOI: https://doi.org/10.1016/j.cmpb.2020.105507. URL: https://www.sciencedirect.com/science/article/pii/S0169260719310624.

[15] Colin B. Josephson et al. «Association of Comorbid-Socioeconomic Clusters with Mortality in Late Onset Epilepsy Derived Through Unsupervised Machine Learning». In: *Seizure - European Journal of Epilepsy* 111 (), pp. 58–67. ISSN: 1059-1311. DOI: 10.1016/j.seizure.2023.07.016. URL: https://doi.org/10.1016/j.seizure.2023.07.016.

[16] S. Lloyd. «Least squares quantization in PCM». In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489.

[17] Alireza Naghizadeh e Dimitris N. Metaxas. «Condensed Silhouette: An Optimized Filtering Process for Cluster Selection in K-Means». In: *Procedia Computer Science* 176 (2020). Knowledge-Based and Intelligent Information and Engineering Systems: Proceedings of the 24th International Conference KES2020, pp. 205–214. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2020.08.022. URL: https://www.sciencedirect.com/science/article/pii/S1877050920318469.

[18] William Stafford Noble. «A Quick Guide to Organizing Computational Biology Projects». In: *PLOS Computational Biology* 5.7 (lug. 2009), pp. 1–5. DOI: 10.1371/journal.pcbi.1000424. URL: https://doi.org/10.1371/journal.pcbi.1000424.

[19] Peter J. Rousseeuw. «Silhouettes: A graphical aid to the interpretation and validation of cluster analysis». In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. URL: https://www.sciencedirect.com/science/article/pii/0377042787901257.

[20] Santiago Schnell. «Ten Simple Rules for a Computational Biologist's Laboratory Notebook». In: *PLOS Computational Biology* 11.9 (set. 2015), pp. 1–5. DOI: 10.1371/journal.pcbi.1004385. URL: https://doi.org/10.1371/journal.pcbi.1004385.

[21] Robert R. Sokal e Charles Duncan Michener. «A statistical method for evaluating systematic relationships». In: *University of Kansas science bulletin* 38 (1958), pp. 1409–1438. URL: https://api.semanticscholar.org/CorpusID:61950873.

[22] Artur Starczewski e Adam Krzyżak. «Performance Evaluation of the Silhouette Index». In: *Artificial Intelligence and Soft Computing*. A cura di Leszek Rutkowski et al. Cham: Springer International Publishing, 2015, pp. 49–58. ISBN: 978-3-319-19369-4.