Università degli Studi di Milano Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di Laurea Triennale

???

Relatore:
XXX YYY

Candidato:
XXX YYY

Correlatore:
XXX YYY

ANNO ACCADEMICO 2023/2024

Indice

Elenco delle figure

1. Abstract

Silhouette é una metrica spesso utilizzata per individuare la combinazione di iperparametri di un algoritmo di clustering non supervisionato che risulti nel clustering piú vicino possibile alla vera struttura di cluster dei dati in analisi. Verrá introdotta Silhouette e come calcolarla, alcune proprietá 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`).

2. Introduzione

Clustering è suddividere un dataset di un certo numero di elementi in sottoparti chiamate cluster sulla base della loro affinità.

Il problema è che diversi algoritmi di clustering non sono in grado di fornire una metrica oggettiva per determinare quanto il clustering che hanno indotto sia effettivamente rappresentativo della struttura del dataset, o se sia semplicemente un raggruppamento arbitrario. Inoltre, diversi algoritmi come K-Means richiedono il numero di cluster come iperparametro, rendendo il discorso ancora più complesso, perché in genere nel clustering non supervisionato non vi è a disposizione alcuna ”ground truth”.

La metrica Silhouette, introdotta per la prima volta in [11], 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’?

Articoli citati: [2], [1], [7], [8], [3].
Articoli citati: [4], [6], [5], [9], [13].

Identifying heterogeneous subgroups of systemic autoimmune diseases by applying a joint dimension reduction and clustering approach to immunomarkers					
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					
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					
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					
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					
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

3. Metodi

3.1 Silhouette

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. Per comodità, si assuma che gli attributi siano tutti dati numerici (altezze, lunghezze, capacità, ecc...). Per ogni elemento, tutti i valori di ciascun attributo sono noti.

A partire da tale dataset 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 un valore indicato con $d(i, j)$ che rappresenta il grado di "dissomiglianza" fra l'elemento i e l'elemento j del dataset.

Tale grado di dissomiglianza è calcolato mediante una **funzione di distanza**, usando come input i valori degli attributi di i e di j. Un esempio di funzione di distanza è la **distanza Euclidea**, definita come segue:

$$d(i,j) = \sqrt{\sum_{m=1}^{M} (f_{i,m} - f_{j,m})^2} = \sqrt{(f_{i,1} - f_{j,1})^2 + \cdots + (f_{i,M} - f_{j,M})^2} \tag{3.1}$$

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.

Altri esempi di distanze sono la **distanza di Manhattan** e la **distanza di Minkowski** (dal punto di vista di Silhouette, quale funzione di distanza venga usata è irrilevante).

"1"	"2"	"3"	"4"	"5"	"6"
0	0.54	0.51	0.65	0.14	0.62
0.54	0	0.3	0.33	0.61	1.09
0.51	0.3	0	0.24	0.51	1.09
0.65	0.33	0.24	0	0.65	1.17
0.14	0.61	0.51	0.65	0	0.62
0.62	1.09	1.09	1.17	0.62	0

Tabella 3.1: Matrice delle distanze per il dataset `iris`. Per questioni di spazio sono presenti solamente i primi 6 elementi.

Una volta nota la matrice delle distanze, si supponga di applicare un algoritmo di clustering (K-Means, ad esempio) per suddividere il dataset in un certo numero di cluster, sia questo K. Per ciascun cluster, è interamente noto sia il numero di suoi elementi, sia a quale cluster ciascun elemento del dataset è stato assegnato.

Per poter calcolare il coefficiente di Silhouette, è prima necessario introdurre due quantità per ciascun elemento i del dataset, indicate rispettivamente con $a(i)$ e $b(i)$.

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:

$$a(i) = \frac{1}{|A|-1} \sum_{j \in \{A-\{i\}\}} d(i,j) \tag{3.2}$$

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, dato che è quello i cui elementi sono i più vicini ad i.

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

$$b(i) = min_{C \neq A} D(i,C) \tag{3.3}$$

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:

$$s(i) = \frac{b(i)-a(i)}{max\{a(i),b(i)\}} \tag{3.4}$$

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:

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

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:

0	1	2
1	2	3
3	1	0.85
3	1	0.82
3	1	0.83
3	1	0.81
3	1	0.85
3	1	0.75
3	1	0.82
3	1	0.85
3	1	0.75
3	1	0.83

0	1	2
1	2	3
3	1	0.85
1	2	0.85
1	2	0.38
2	1	0.85
1	2	0.59
1	2	0.37
1	2	0.59
1	2	0.28
1	3	0.27
1	2	0.34
1	2	0.58

0	1	2
1	2	3
1	2	0.63
2	1	0.5
1	2	0.23
2	1	0.61
2	1	0.36
2	1	0.56
2	1	0.54
1	2	0.47
2	1	0.56
2	1	0.44
2	1	0.56

Tabella 3.2: Valori di $s(i)$, cluster e neighboring cluster per i primi 10 elementi dei tre cluster. Si noti come i valori di $s(i)$ del primo cluster siano più alti ed il neighboring cluster sia sempre lo stesso, mentre gli altri due cluster hanno valori più variegati.

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

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 $[0,1]$.

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 $s(i)$, ordinato dalla colonna più alta a quella più bassa. Tale

grafico può essere arricchito riportando informazioni ulteriori per ciascun elemento come, ad esempio: in quale cluster si trova, qual'è il suo neighboring cluster, il suo valore di $s(i)$.

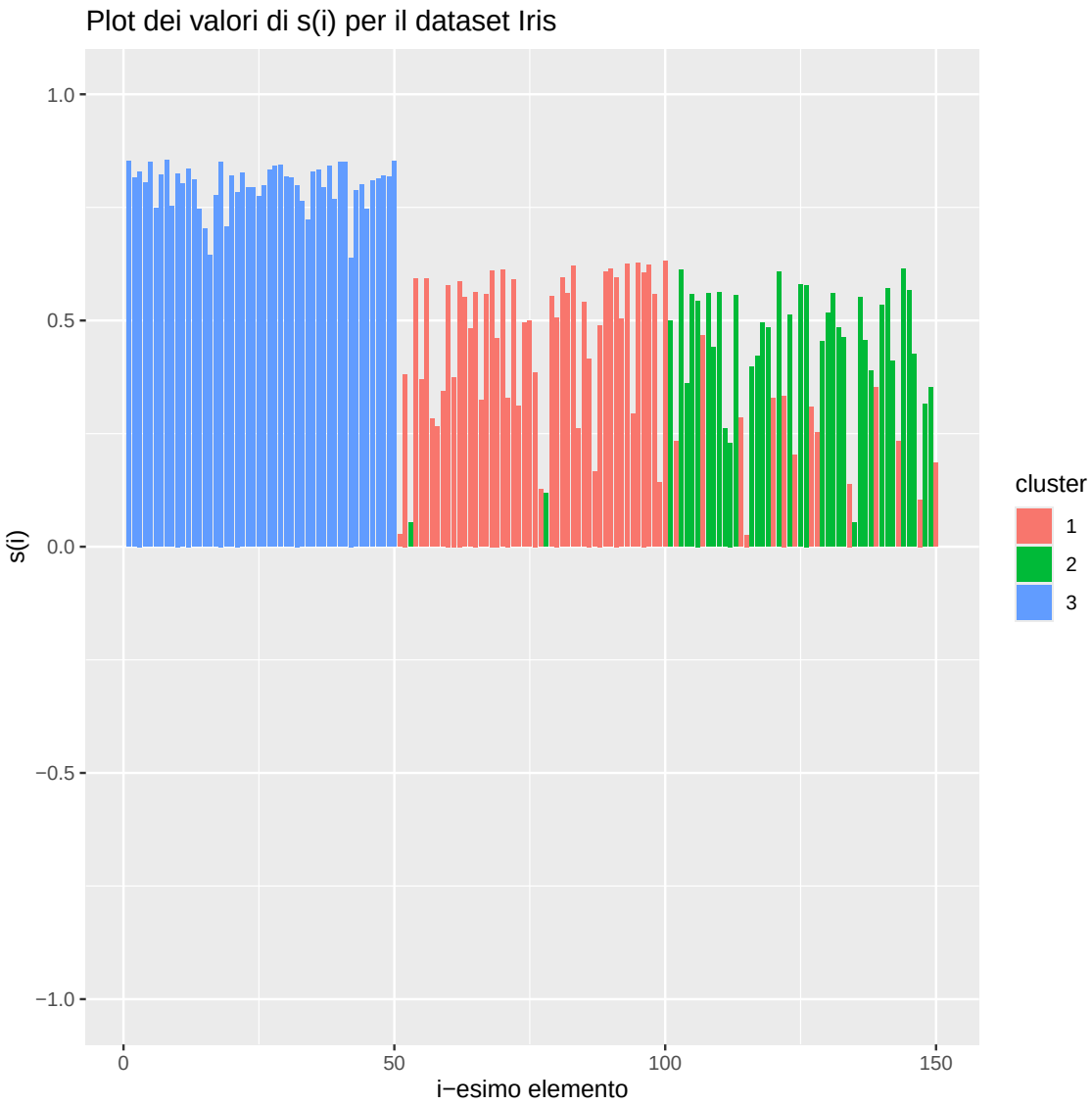


Figura 3.1: Silhouette plot per il dataset `iris`.

Il vantaggio di Silhouette è che non dipende da quale algoritmo è stato usato per effettuare il clustering. Per tale motivo, può essere usato per valutare "a posteriori" il risultato dell'algoritmo, provando a modificare il valore degli iperparametri per valutare quale combinazione di iperparametri restituisce il risultato più coerente. Questo riesce particolarmente bene negli algoritmi in cui il numero di cluster figura fra gli iperparametri, come K-Means.

Per esempio, si supponga che un dataset abbia effettivamente 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.

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.

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. L'idea è quella di testare diverse combinazioni di iperparametri dell'algoritmo di clustering e scegliere la combinazione che restituisce la Silhouette media complessiva più grande: tale combinazione sarà quella che restituisce il clustering che meglio interpreta il dataset.

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.

3.2 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, é necessario scegliere un pacchetto fra quelli disponibili.

Ho cercato quante piú implementazioni di Silhouette possibili utilizzando il sito `https://rdrr.io` e scegliendone quante più possibili che provenissero dal repository CRAN. In particolare, sono stati scelti `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`.

Per comparare le performance delle diverse implementazioni di Silhouette ho eseguito due test, uno chiamato "sanity check" ed uno chiamato "matrice binaria". Il codice per la generazione dei test è liberamente disponibile. Ho organizzato il codice seguendo le linee guida riportate in [10]. Ho anche tenuto un lab notebook per tenere traccia dei risultati mano mano che venivano generati, facendo riferimento a [12]

Il test Sanity check prevede di utilizzare due dataset creati artificialmente, il primo dove la struttura di cluster è estremamente evidente ed il secondo dove la struttura di

cluster è completamente assente, applicare su questi l'algoritmo di clustering K-Means e calcolare sul risultato di quest'ultimo la Silhouette media.

Lo scatter plot dei due dataset è riportato in Figure 3.3. Li ho costruiti utilizzando la funzione `rnorm`, che genera dei punti casuali a partire da una distribuzione normale bivariata. Il primo è stato costruito usando due distribuzioni normali fuse insieme, entrambe con una deviazione standard molto bassa, di modo che i punti siano concentrati attorno alla media. Il secondo è costituito da una sola distribuzione normale centrata in $(0,0)$ con una deviazione standard molto ampia, di modo che i punti siano molto dispersi.

Il test matrice binaria inizia con una matrice di dimensione $2ntimesn$, costituita per metà da 0 e per metà da 1. Su tale matrice viene applicato K-Means con iperparametro $K = 2$ 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. Tale test viene ripetuto esattamente $2n$ volte, ogni volta con una riga diversa.

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 piano piano perdano si riducano così come il dataset perde coesione. Ci si aspetta che i punti sul grafico approssimino una distribuzione lineare.

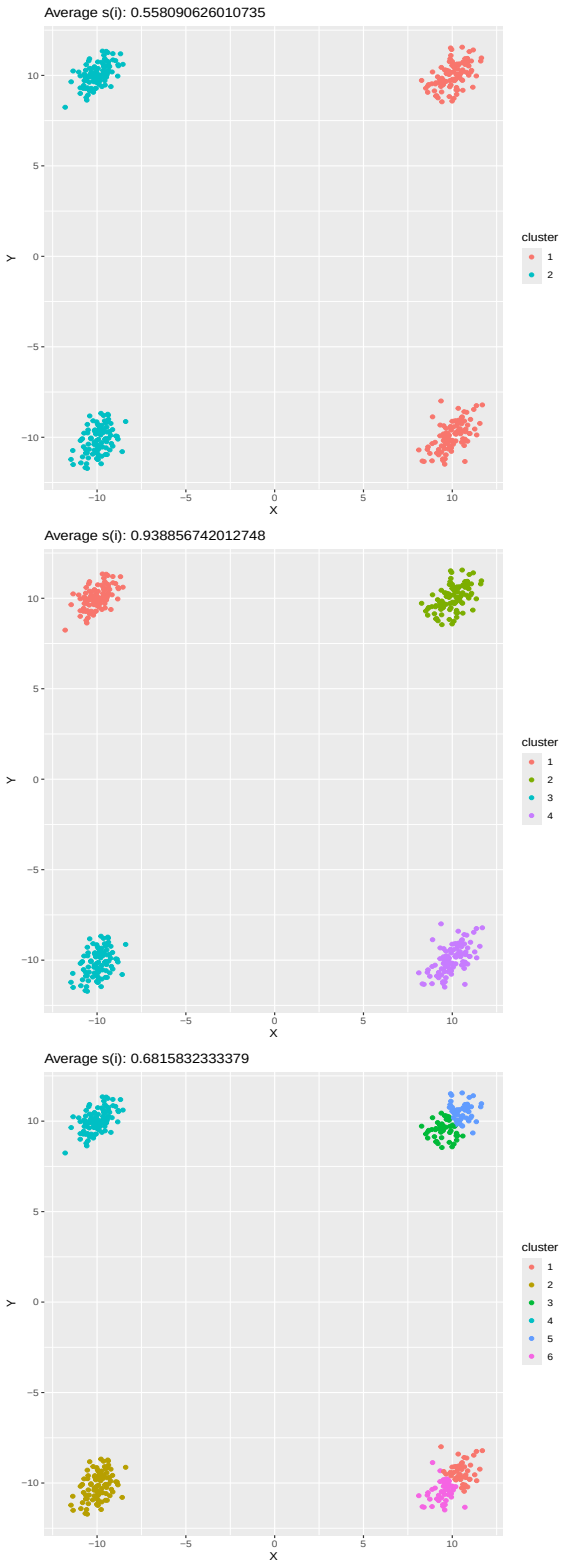


Figura 3.2: Clustering con K-Means usando K = 2, 4, 6 per un dataset auto generato in cui la struttura di cluster è ben visibile.

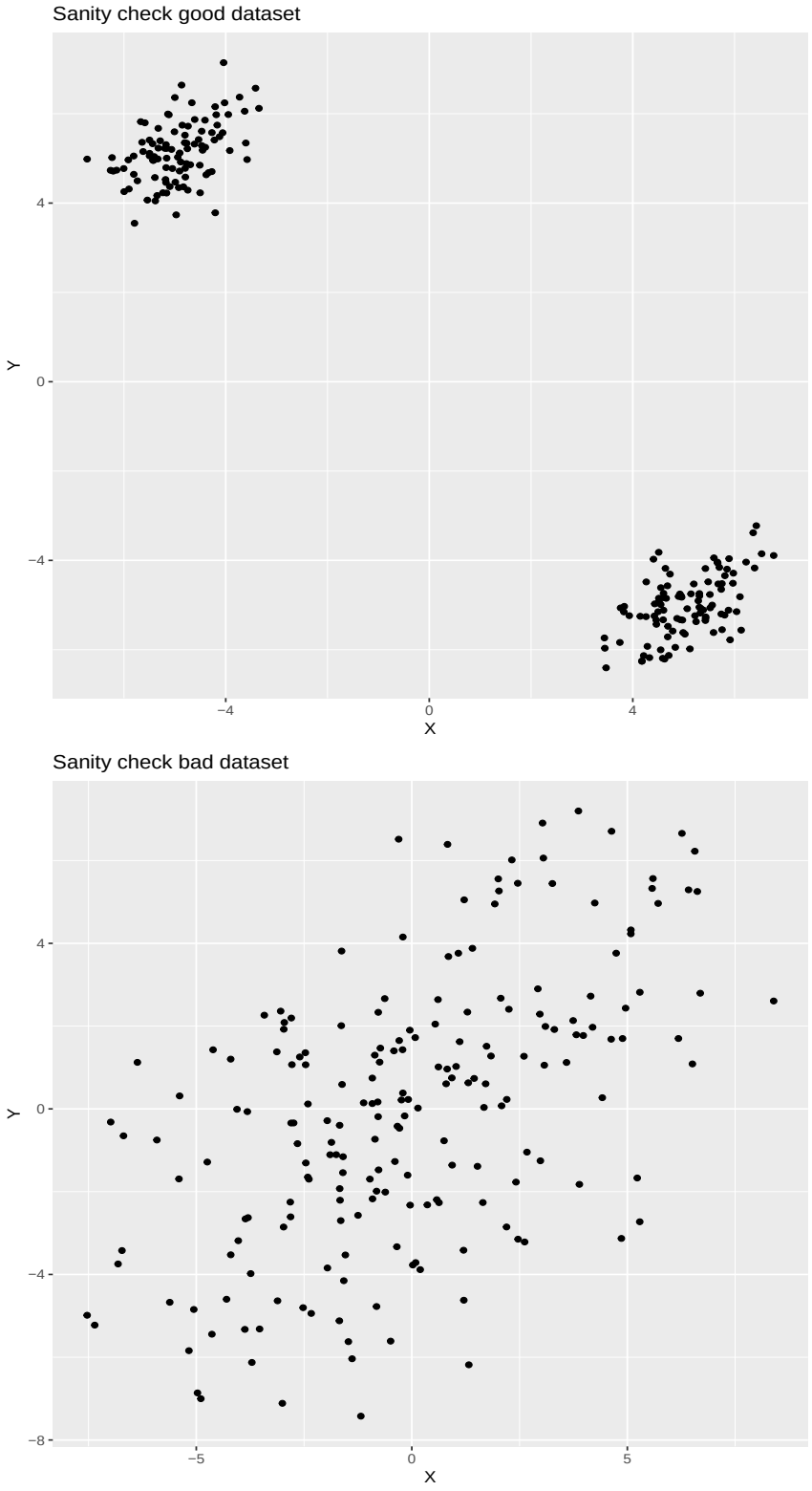


Figura 3.3: Plot dei dataset per il test sanity check. Il primo presenta una struttura di cluster ben definita, mentre nel secondo la struttura di cluster è completamente assente.

4. Risultati

4.1 Risultati dei test

I risultati per il test sanity check sono riportati di seguito. Il test sanity check non é 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 é un risultato atteso, perché il test era appositamente costruito per escludere pacchetti problematici.

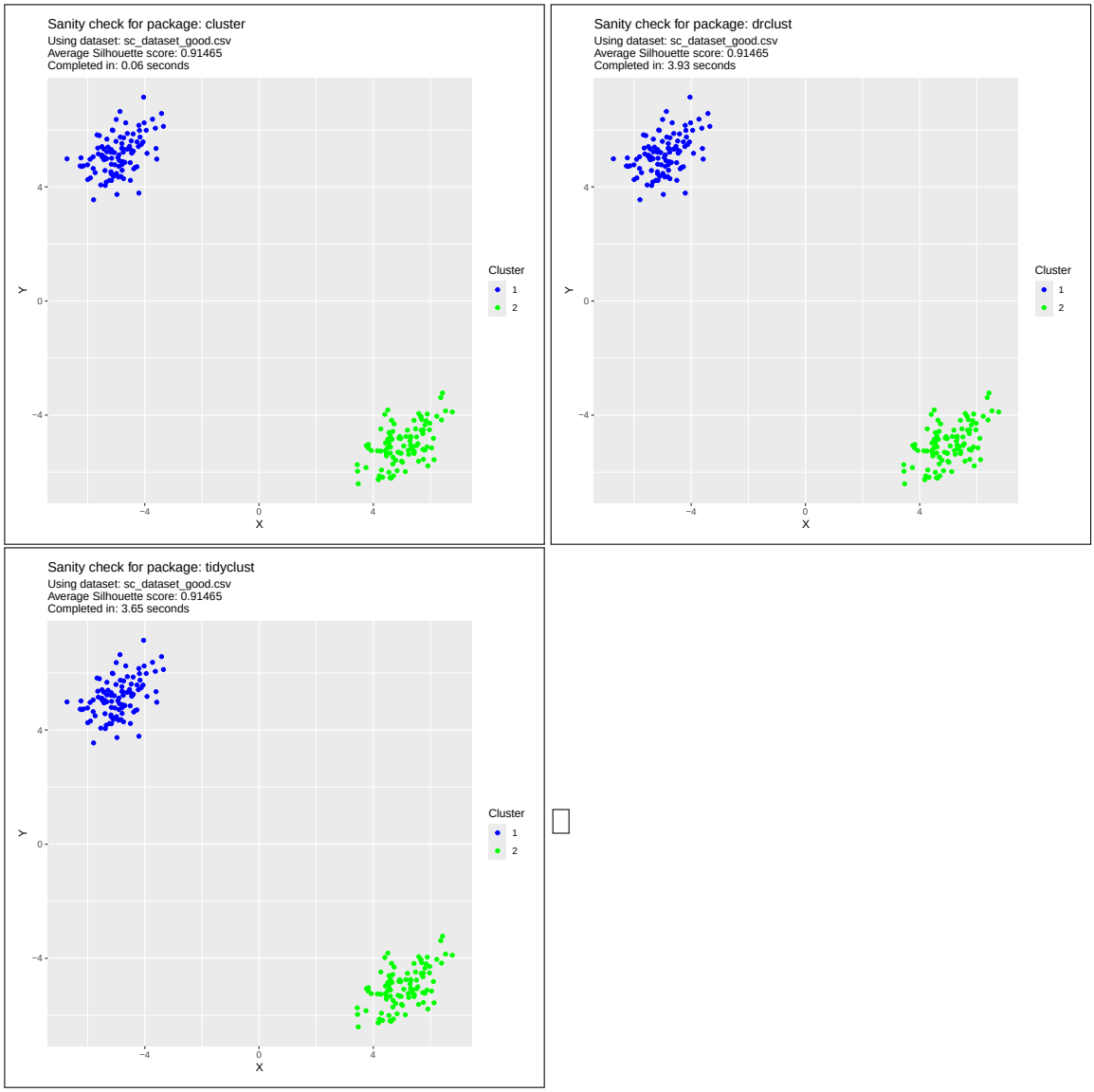


Figura 4.1: Plot dei test sanity check usando il dataset con struttura di cluster ben visibile.

I risultati per il test matrice binaria sono riportati di seguito. Il test é stato piú informativo del precedente, perché i valori restituiti avevano delle differenze evidenziabili.

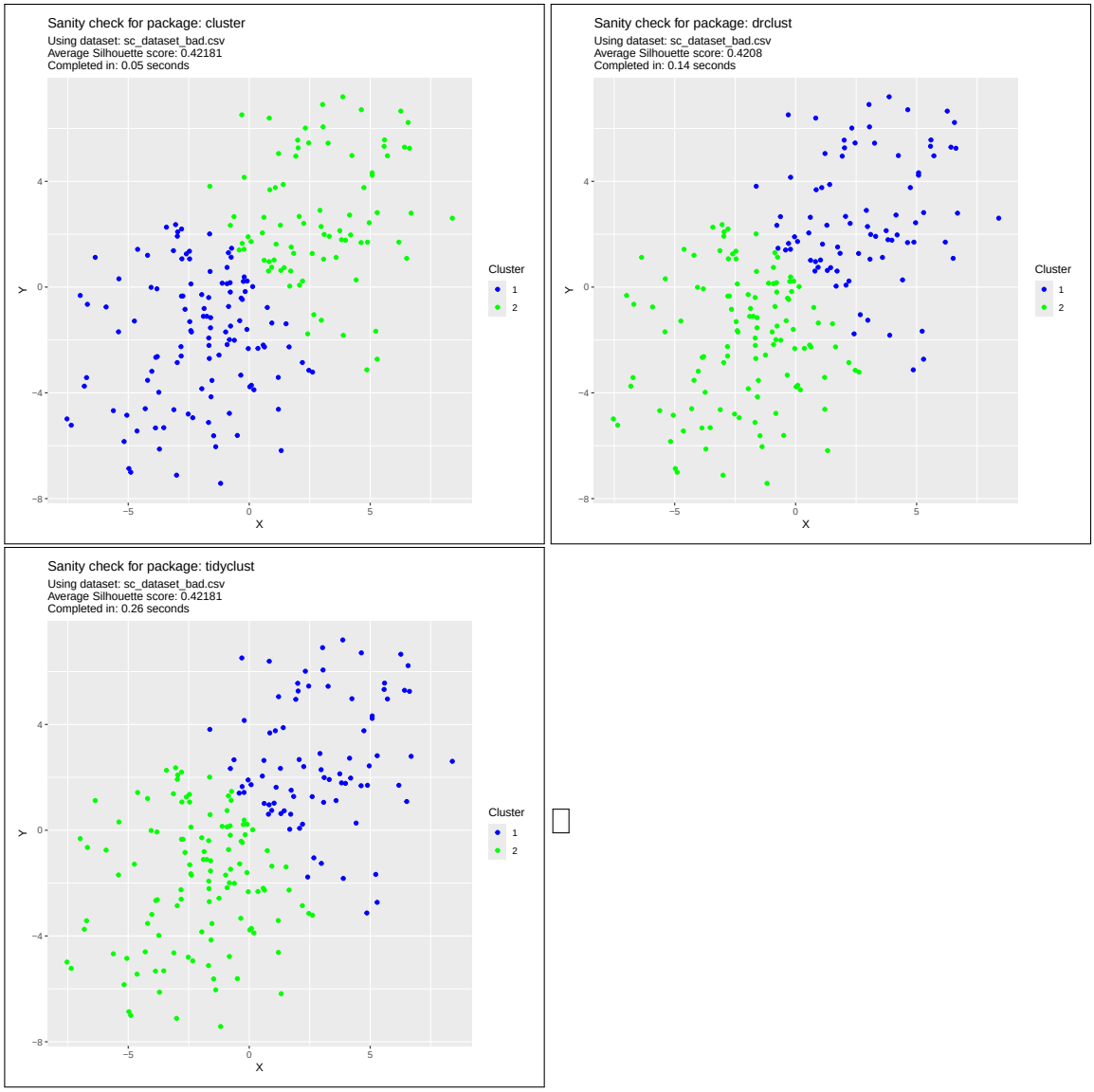


Figura 4.2: Plot dei test sanity check usando il dataset con struttura di cluster assente.

In particolare, `Kira` é 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.

4.2 Risultati delle applicazioni su EHR

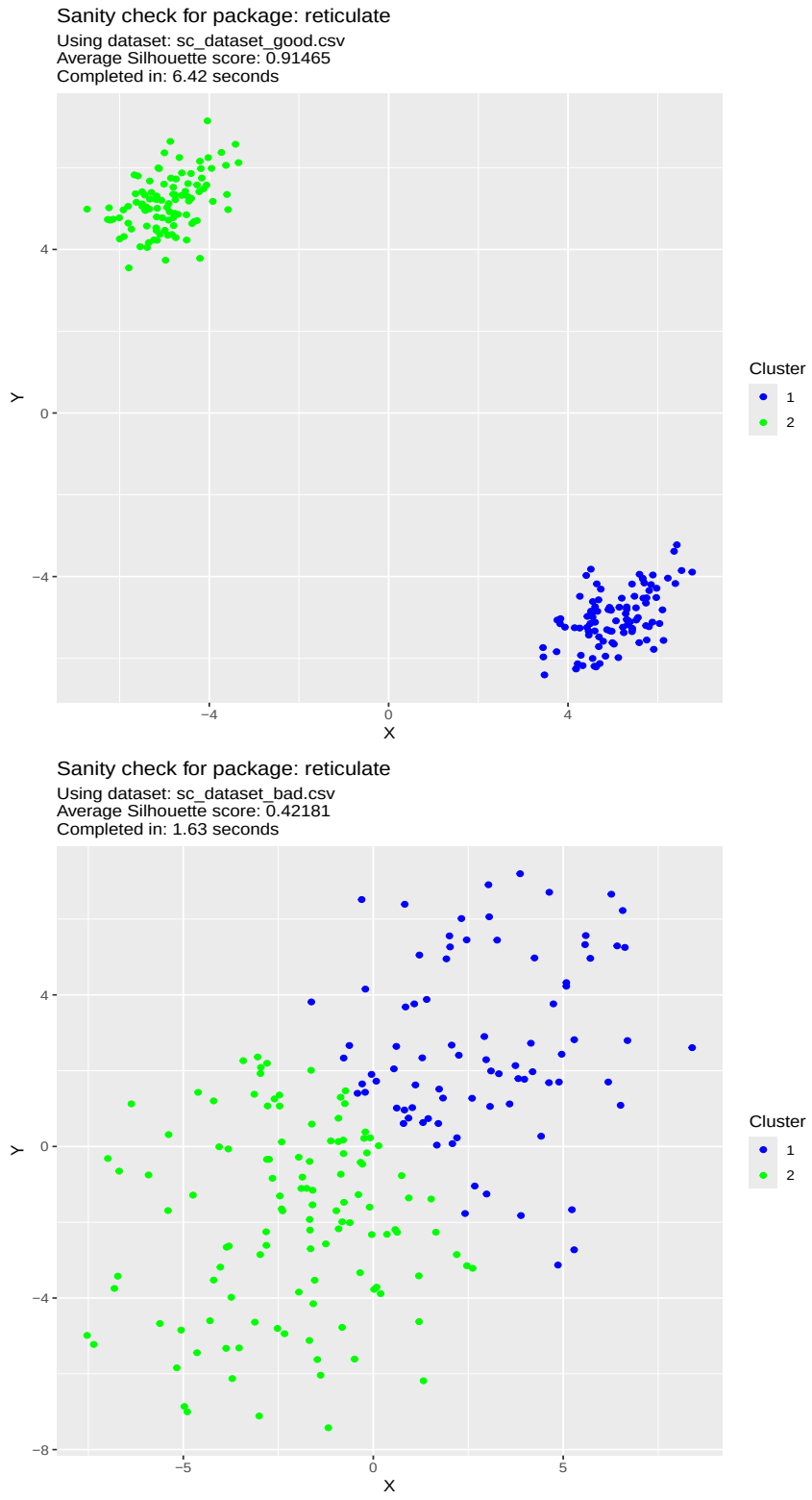


Figura 4.3: Plot del test sanity check usando `scikit-learn` attraverso `reticulate`.

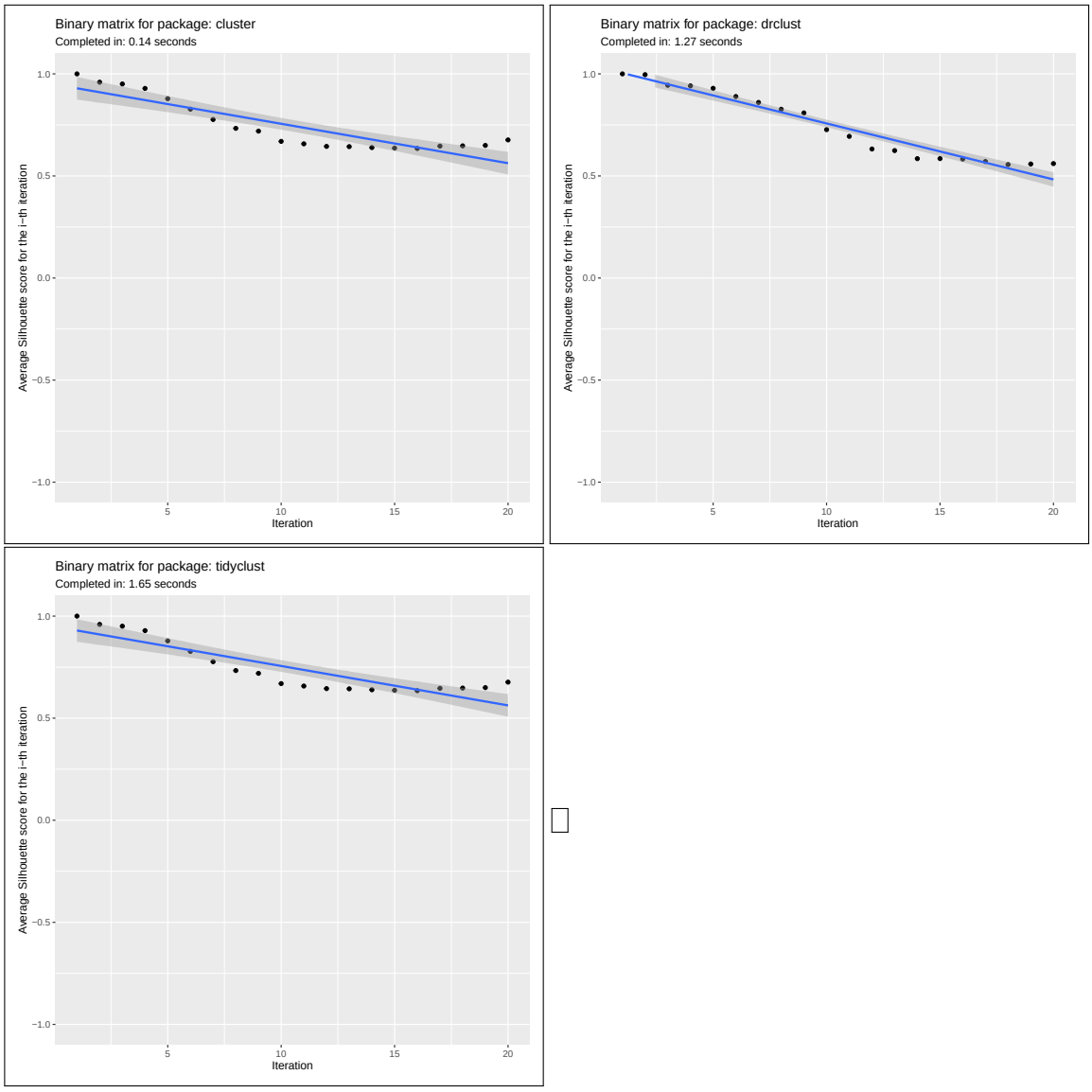


Figura 4.4: Plot dei test matrice binaria.

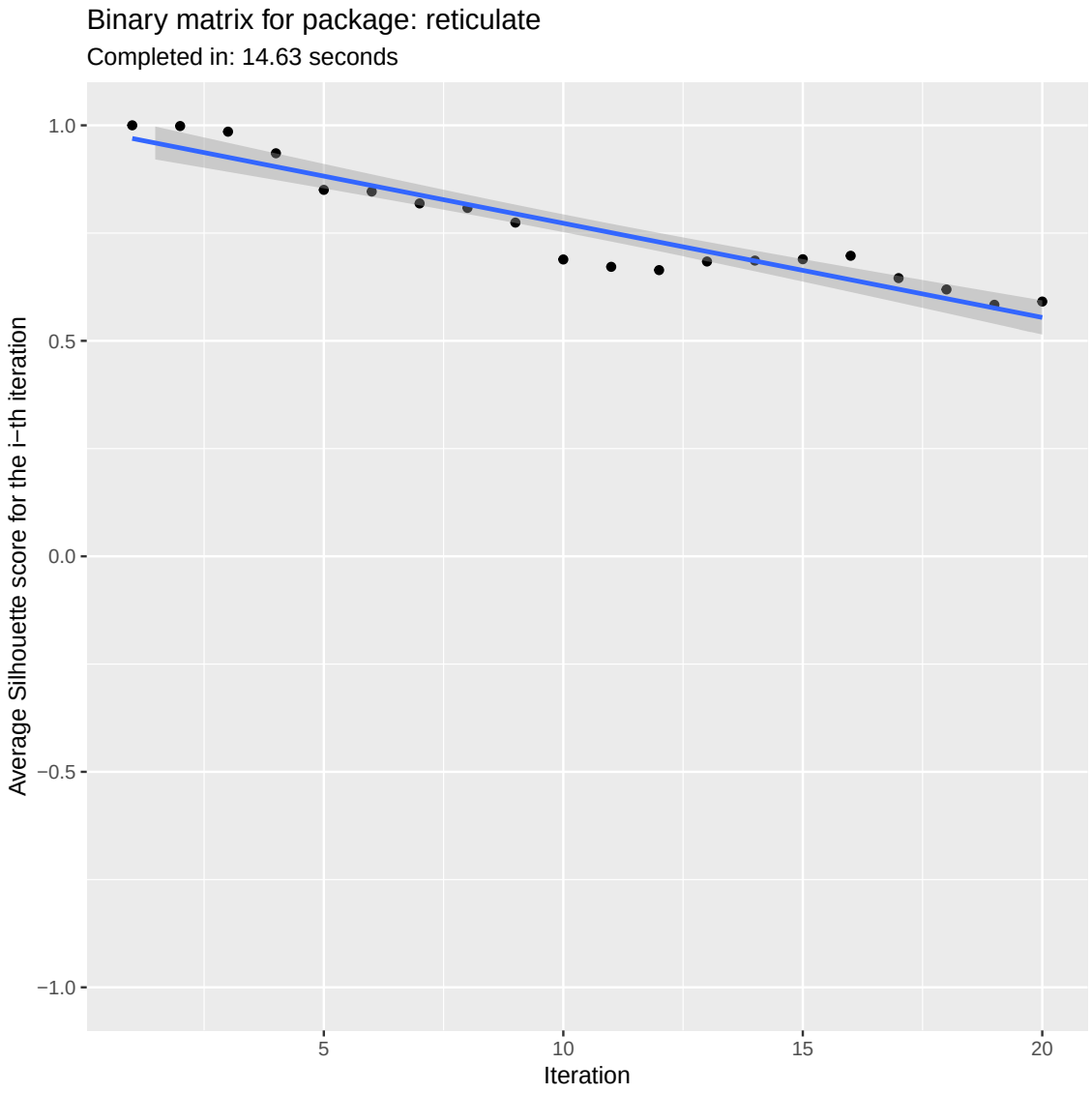


Figura 4.5: Plot del test matrice binaria per il pacchetto `scikit-learn` (attraverso `reticulate`)

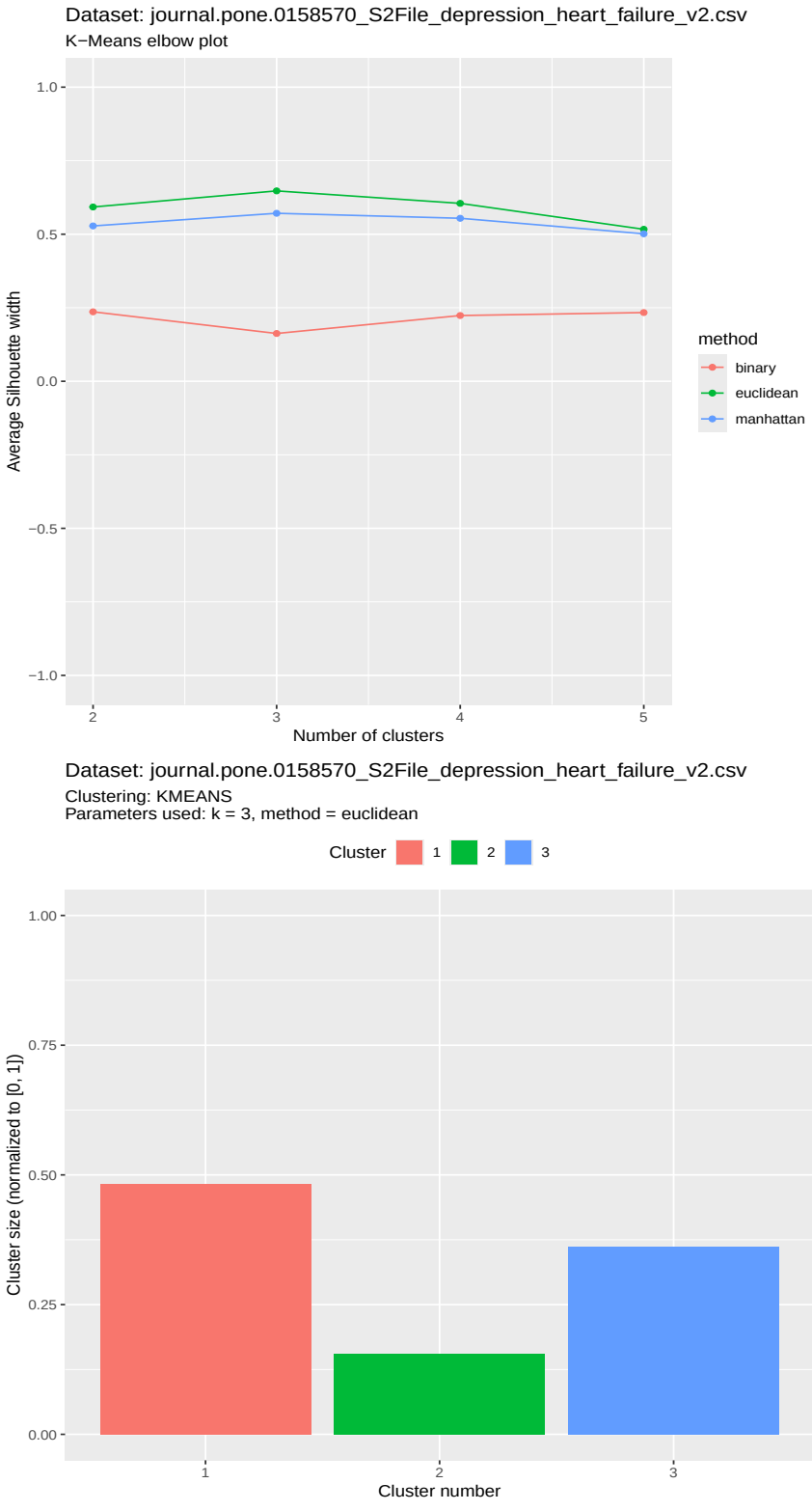


Figura 4.6: Risultati dell'algoritmo K-Means per il dataset `results_journal.pone.0158570_S2File_depression_heart_failure_v2.csv`

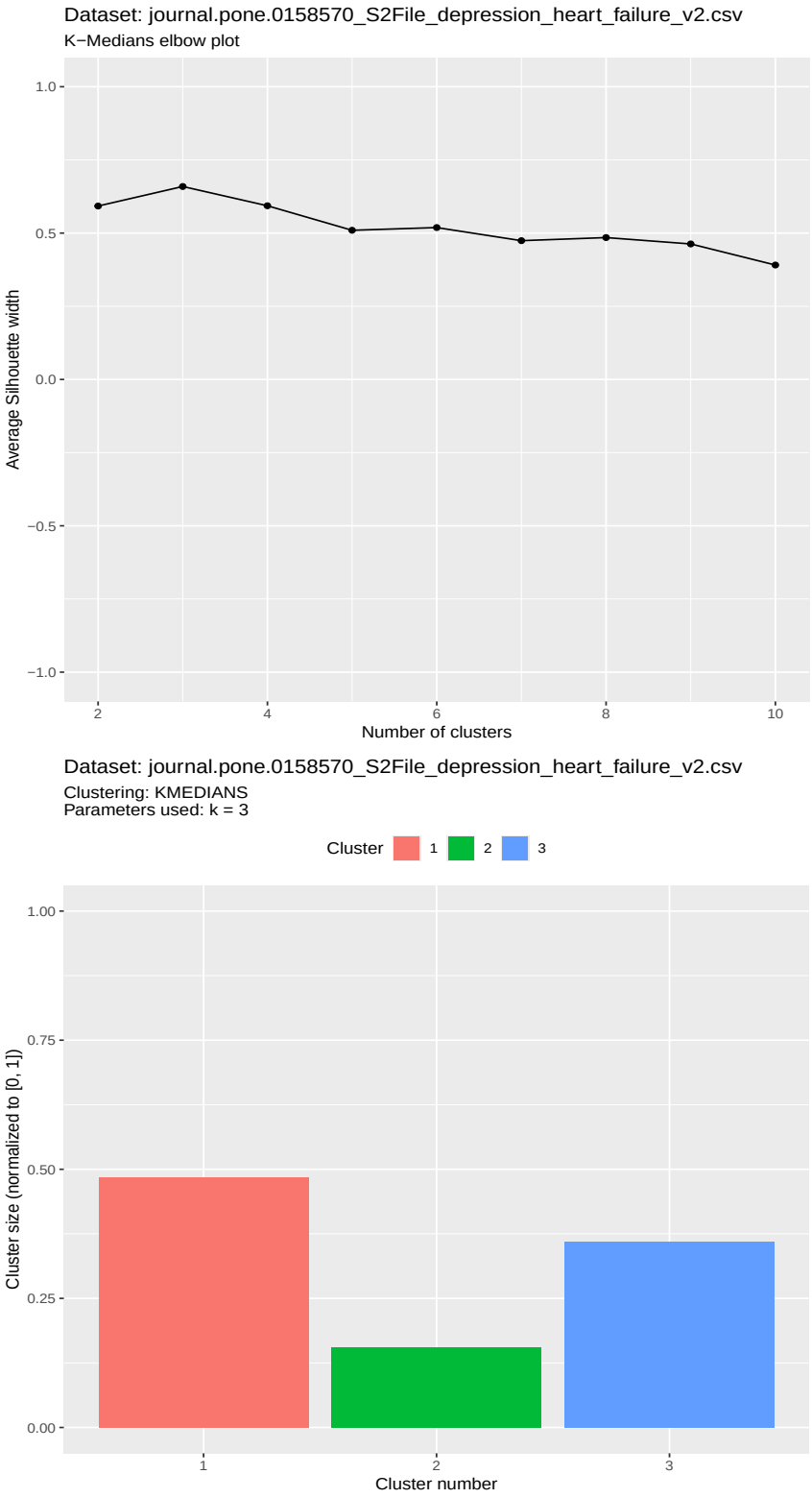


Figura 4.7: Risultati dell'algoritmo K-Medians per il dataset `results_journal.pone.0158570_S2File_depression_heart_failure_v2.csv`

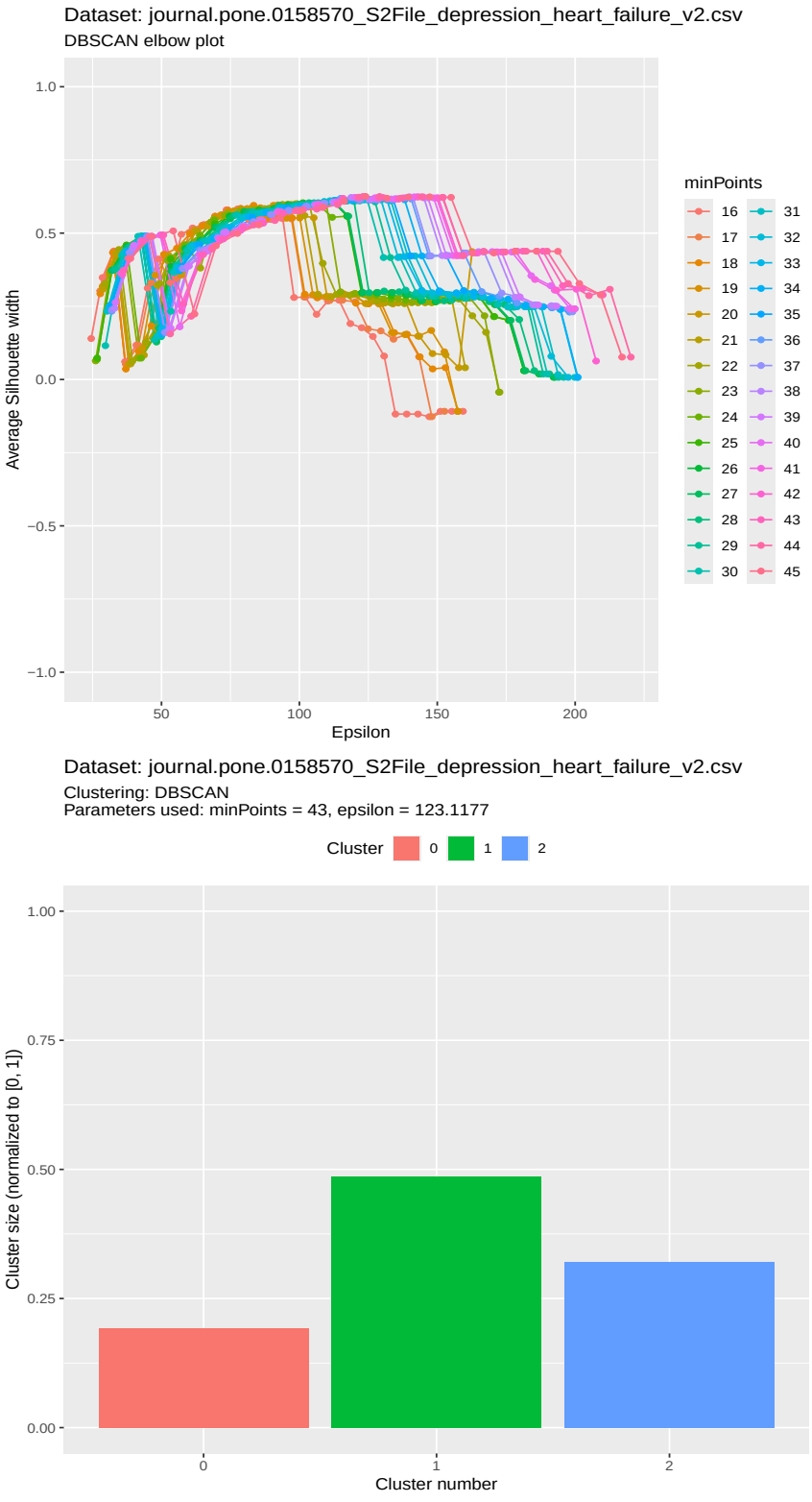


Figura 4.8: Risultati dell'algoritmo DBSCAN per il dataset `results_journal.pone.0158570_S2File_depression_heart_failure_v2.csv`

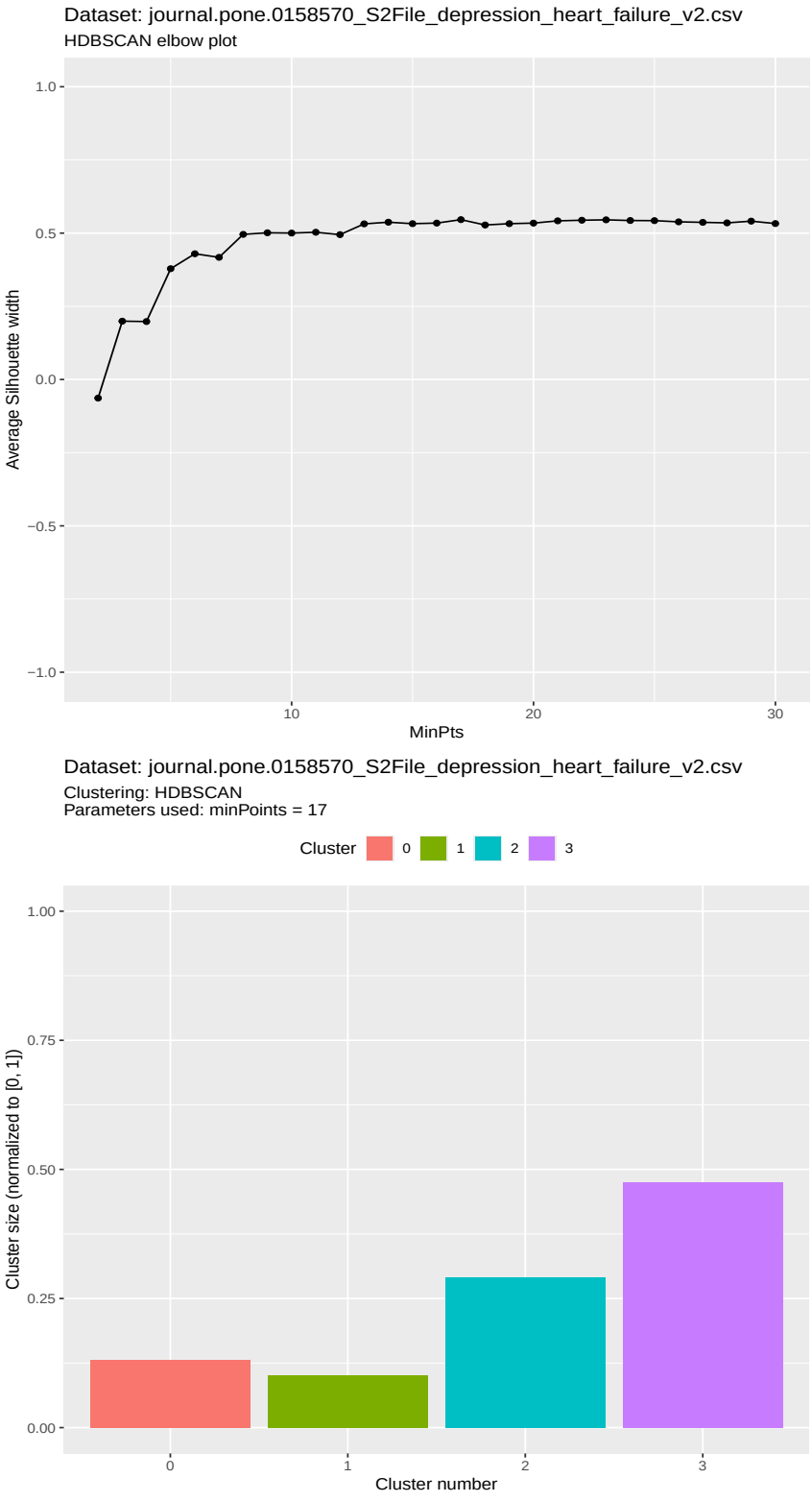
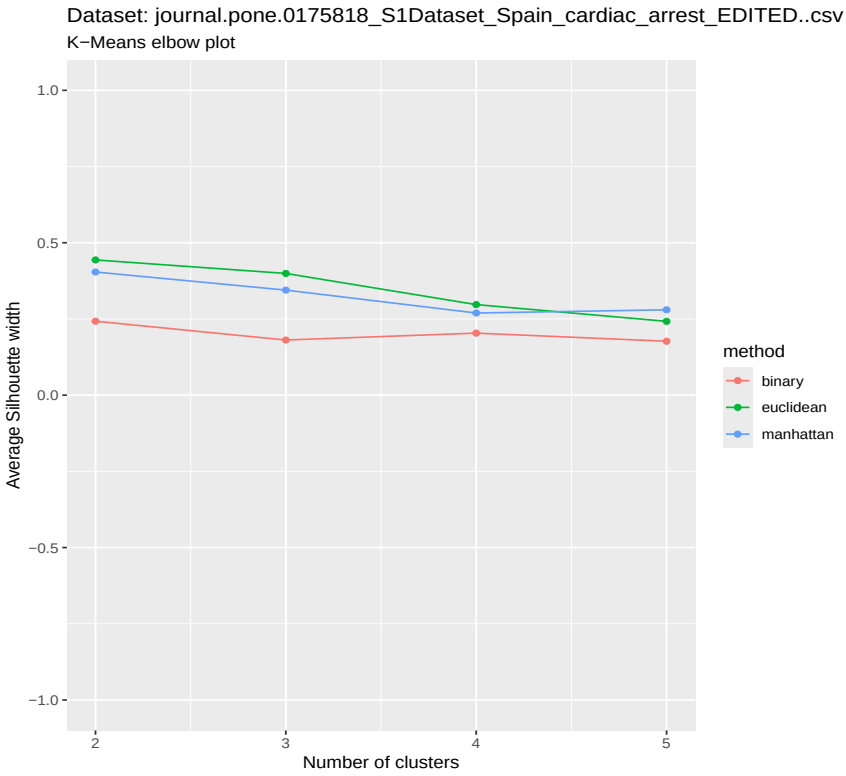


Figura 4.9: Risultati dell'algoritmo HDBSCAN per il dataset `results_journal.pone.0158570_S2File_depression_heart_failure_v2.csv`



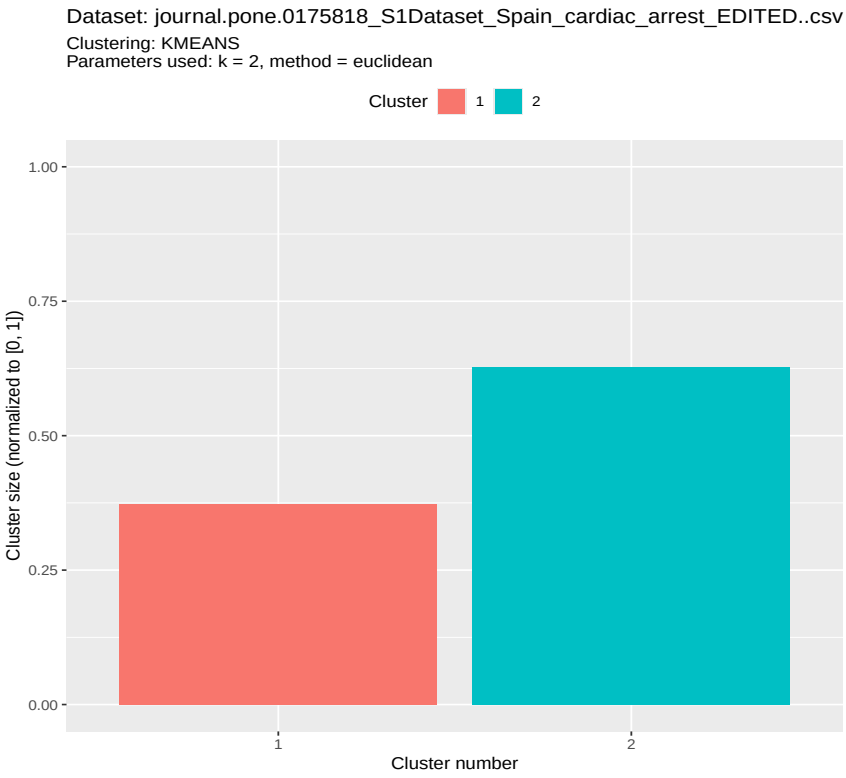


Figura 4.10: Risultati dell'algoritmo K-Means per il dataset `results_journal.pone.0175818_S1Dataset_Spain_cardiac_arrest_EDITED..csv`

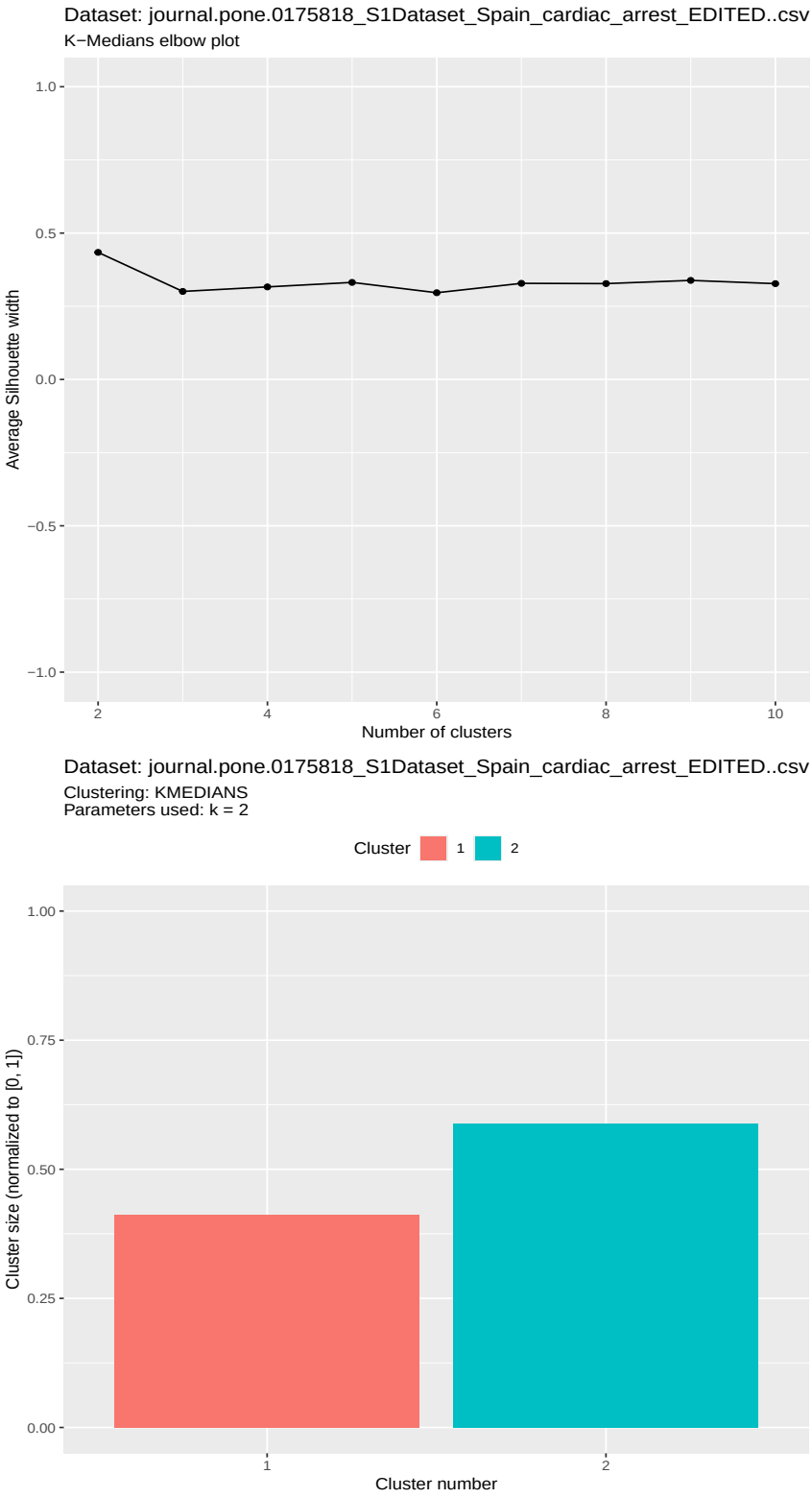


Figura 4.11: Risultati dell'algoritmo K-Medians per il dataset `results_journal.pone.0175818_S1Dataset_Spain_cardiac_arrest_EDITED..csv`

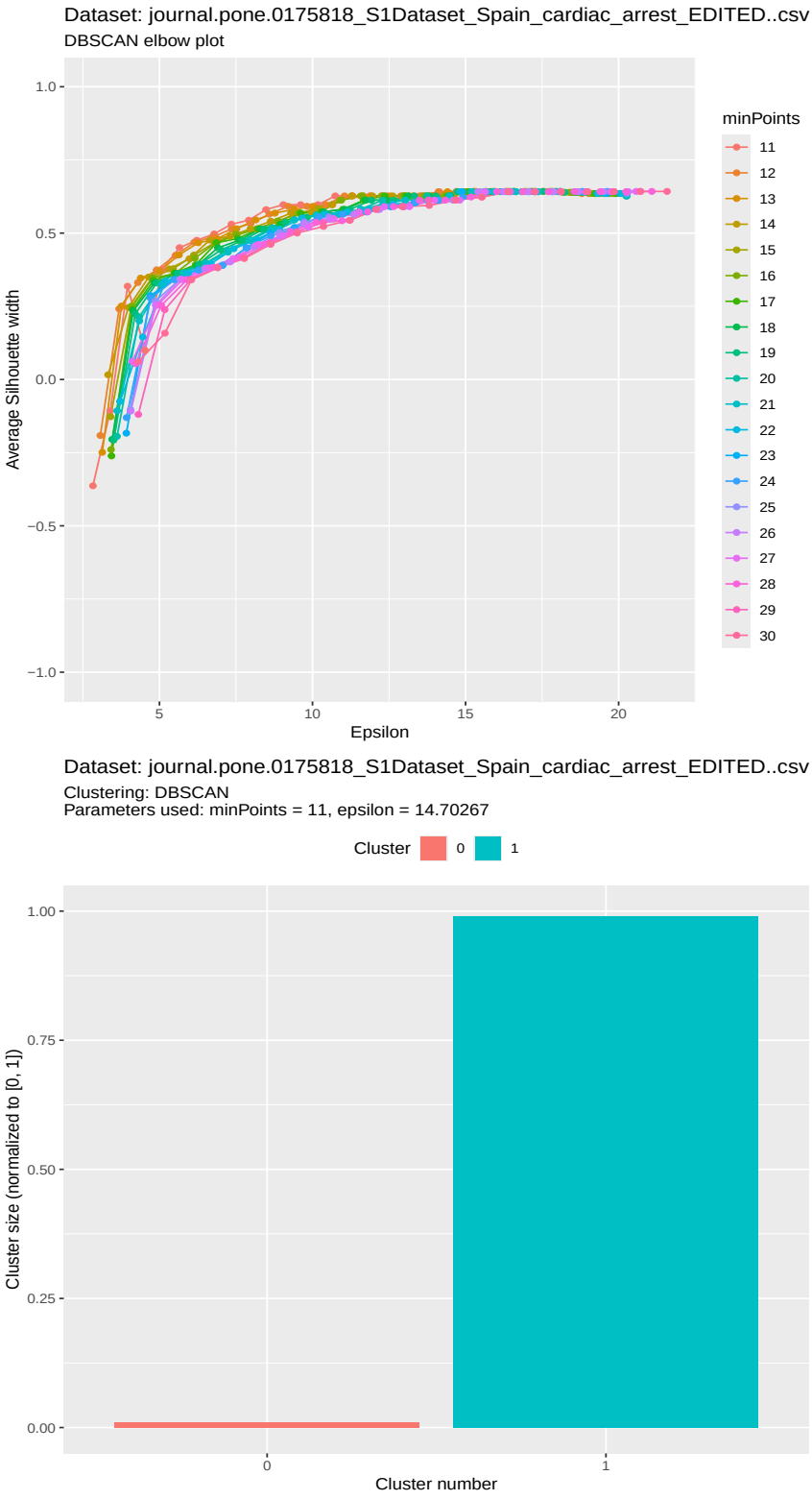


Figura 4.12: Risultati dell'algoritmo DBSCAN per il dataset `results_journal.pone.0175818_S1Dataset_Spain_cardiac_arrest_EDITED..csv`

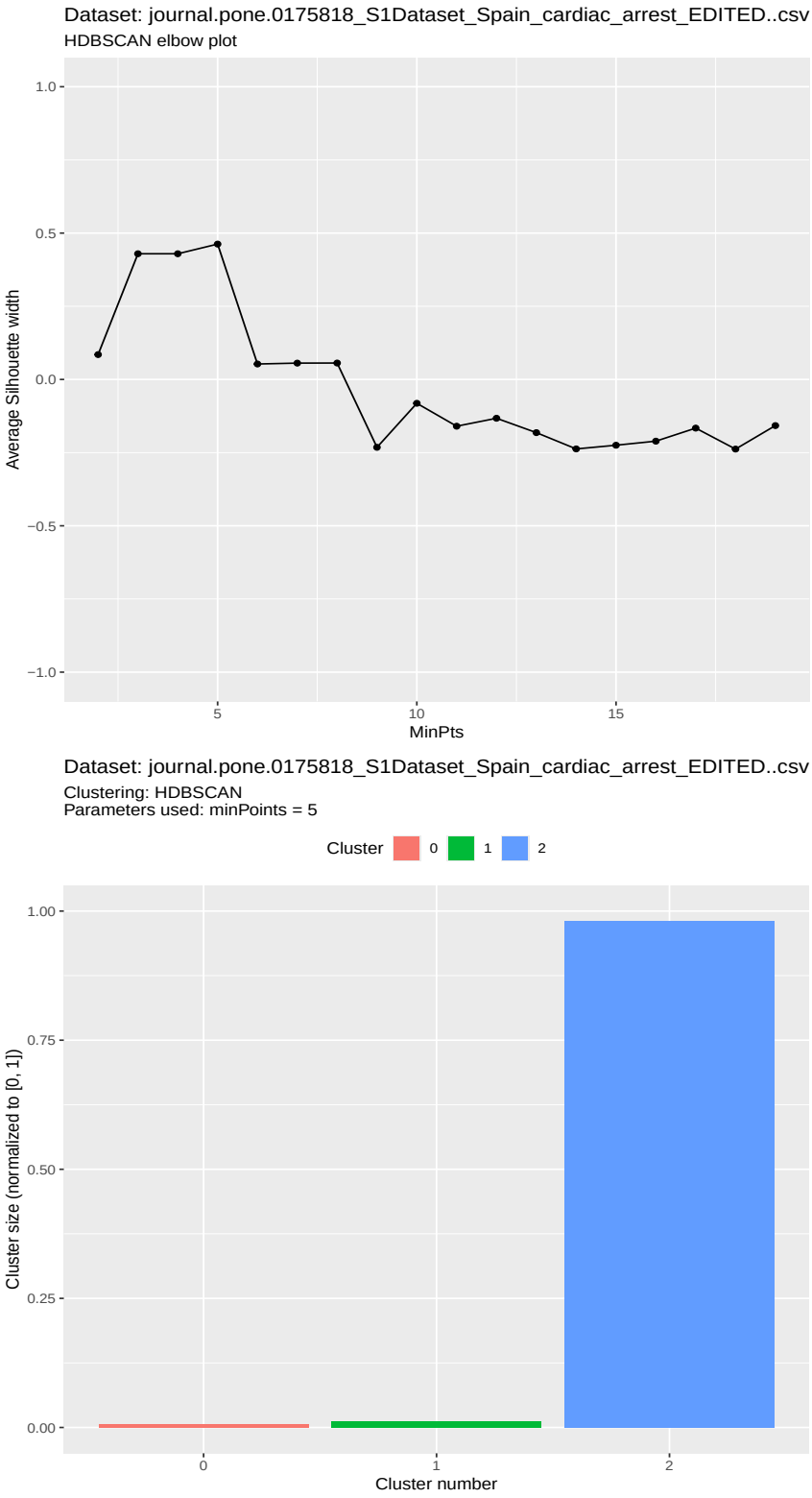


Figura 4.13: Risultati dell'algoritmo HDBSCAN per il dataset `results_journal.pone.0175818_S1Dataset_Spain_cardiac_arrest_EDITED..csv`

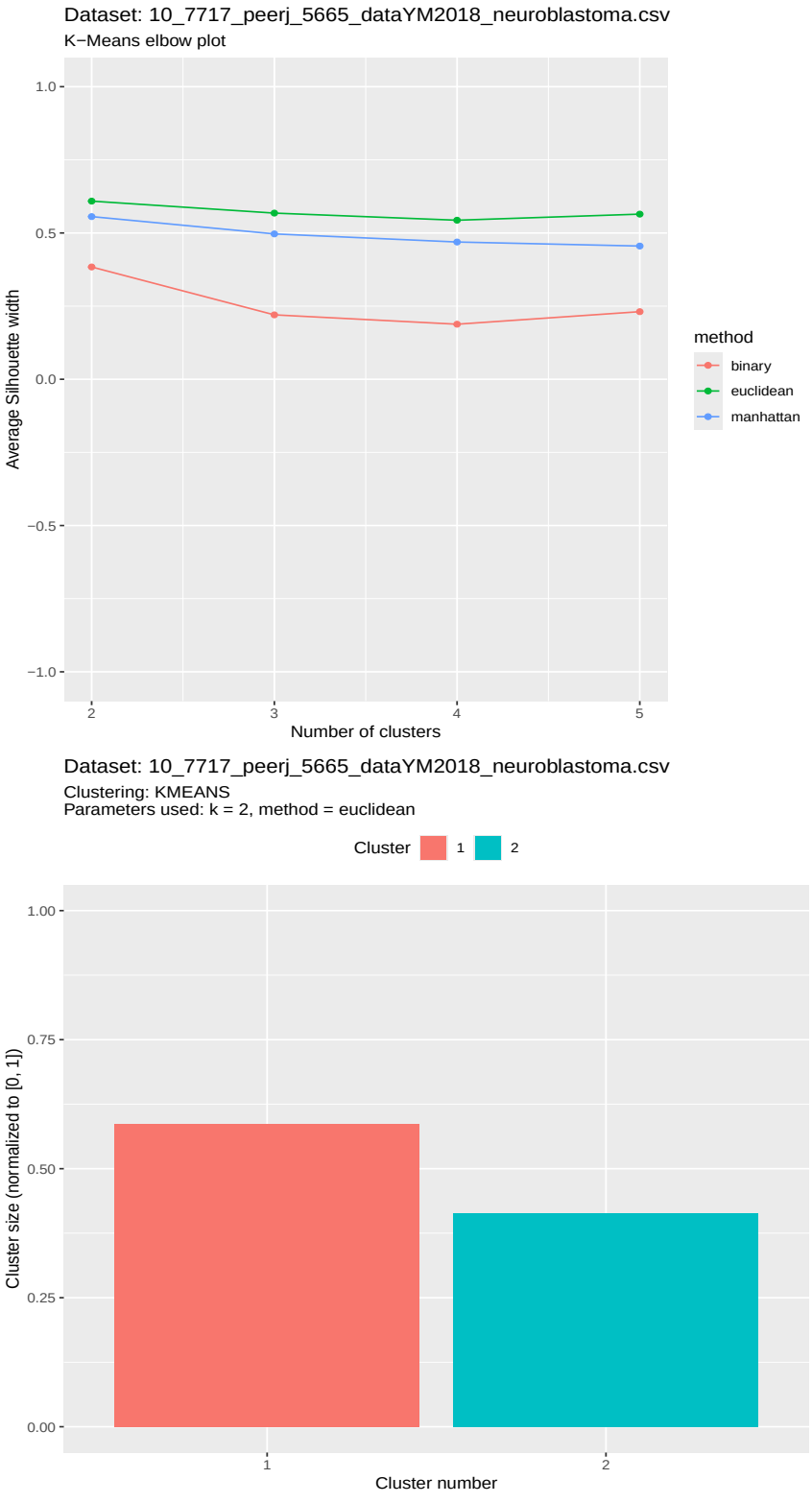


Figura 4.14: Risultati dell'algoritmo K-Means per il dataset `results_10_7717_peerj_5665_dataYM2018_neuroblastoma.csv`

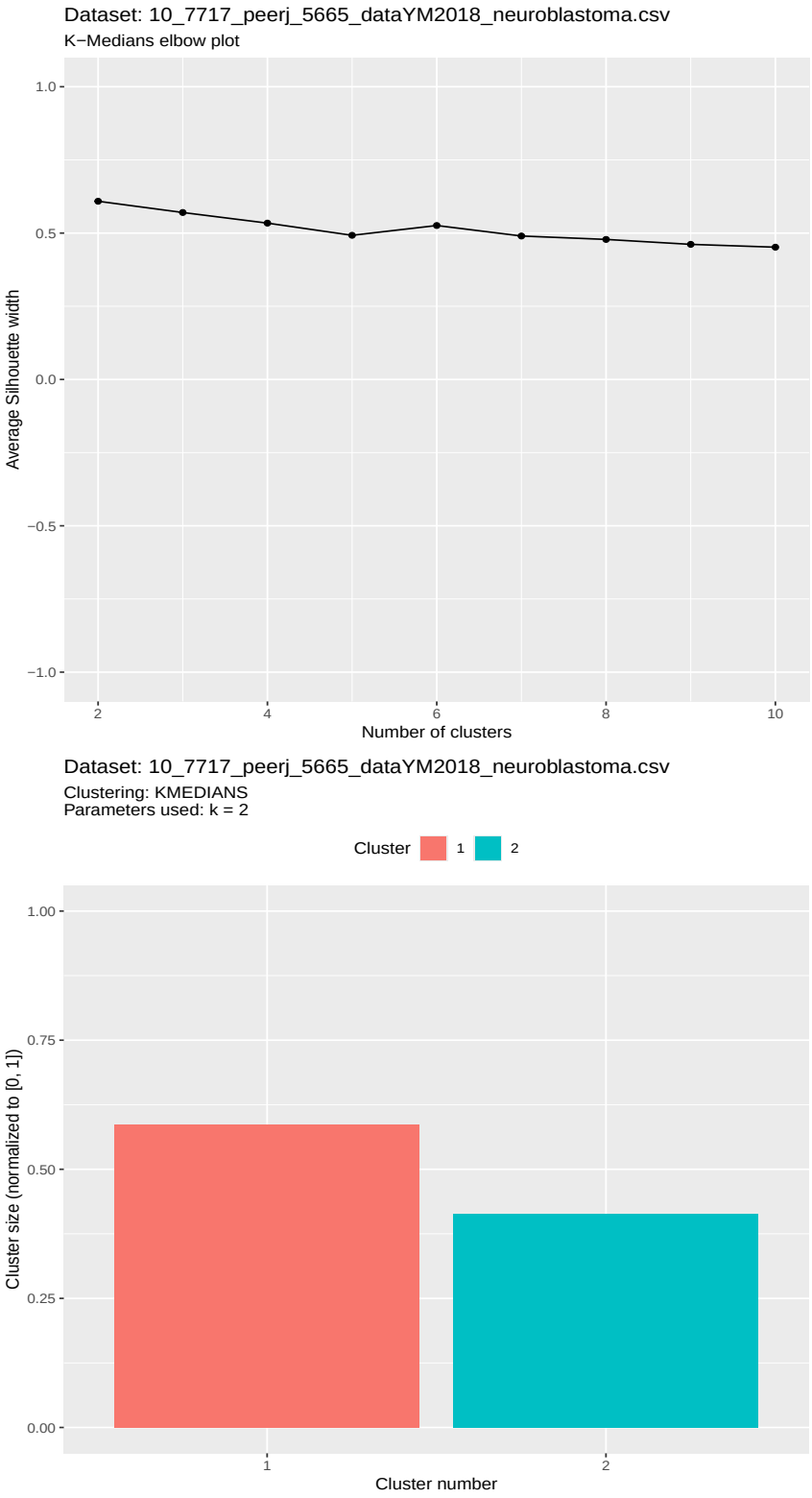


Figura 4.15: Risultati dell'algoritmo K-Medians per il dataset `results_10_7717_peerj_5665_dataYM2018_neuroblastoma.csv`

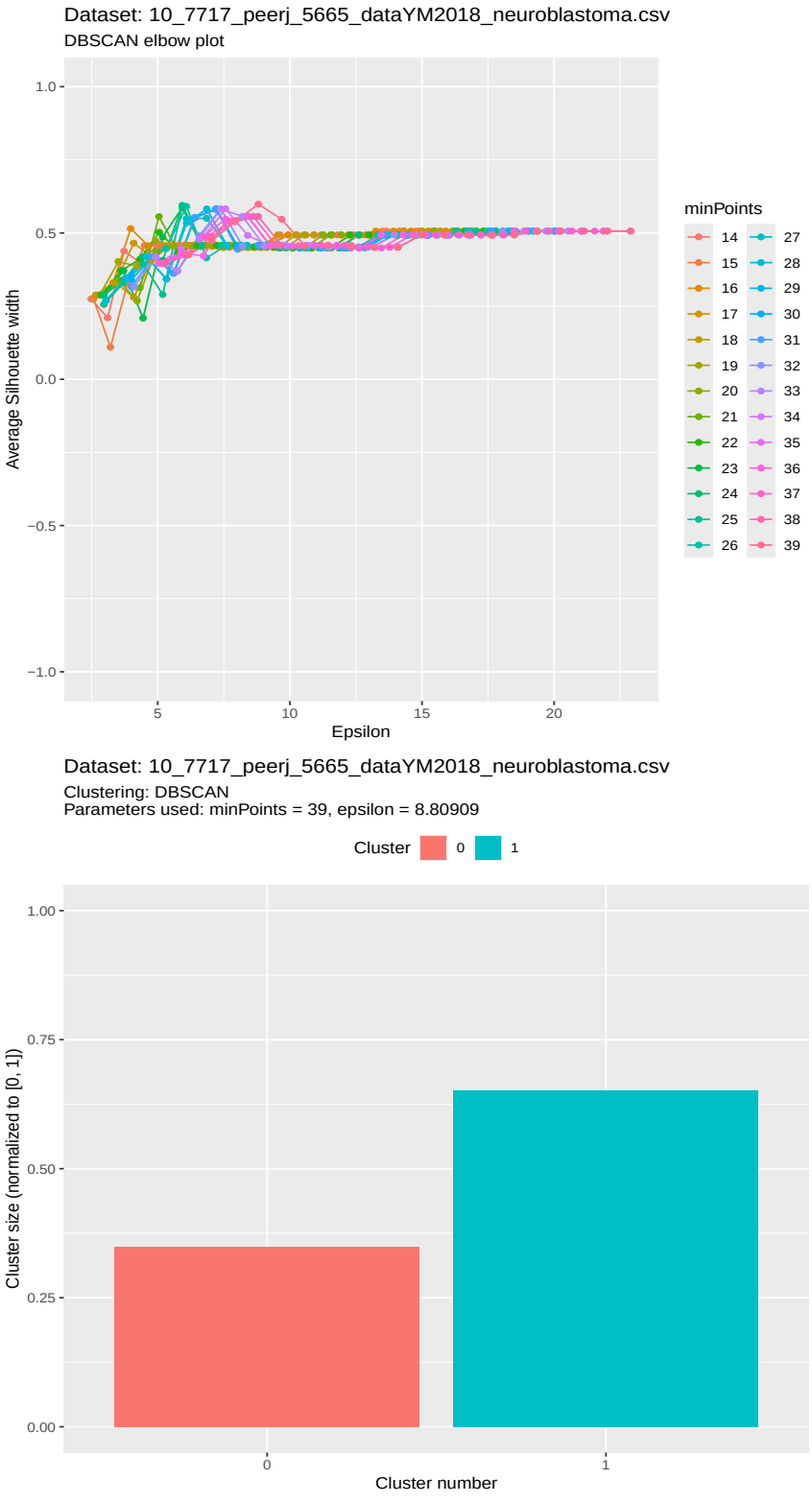


Figura 4.16: Risultati dell'algoritmo DBSCAN per il dataset `results_10_7717_peerj_5665_dataYM2018_neuroblastoma.csv`

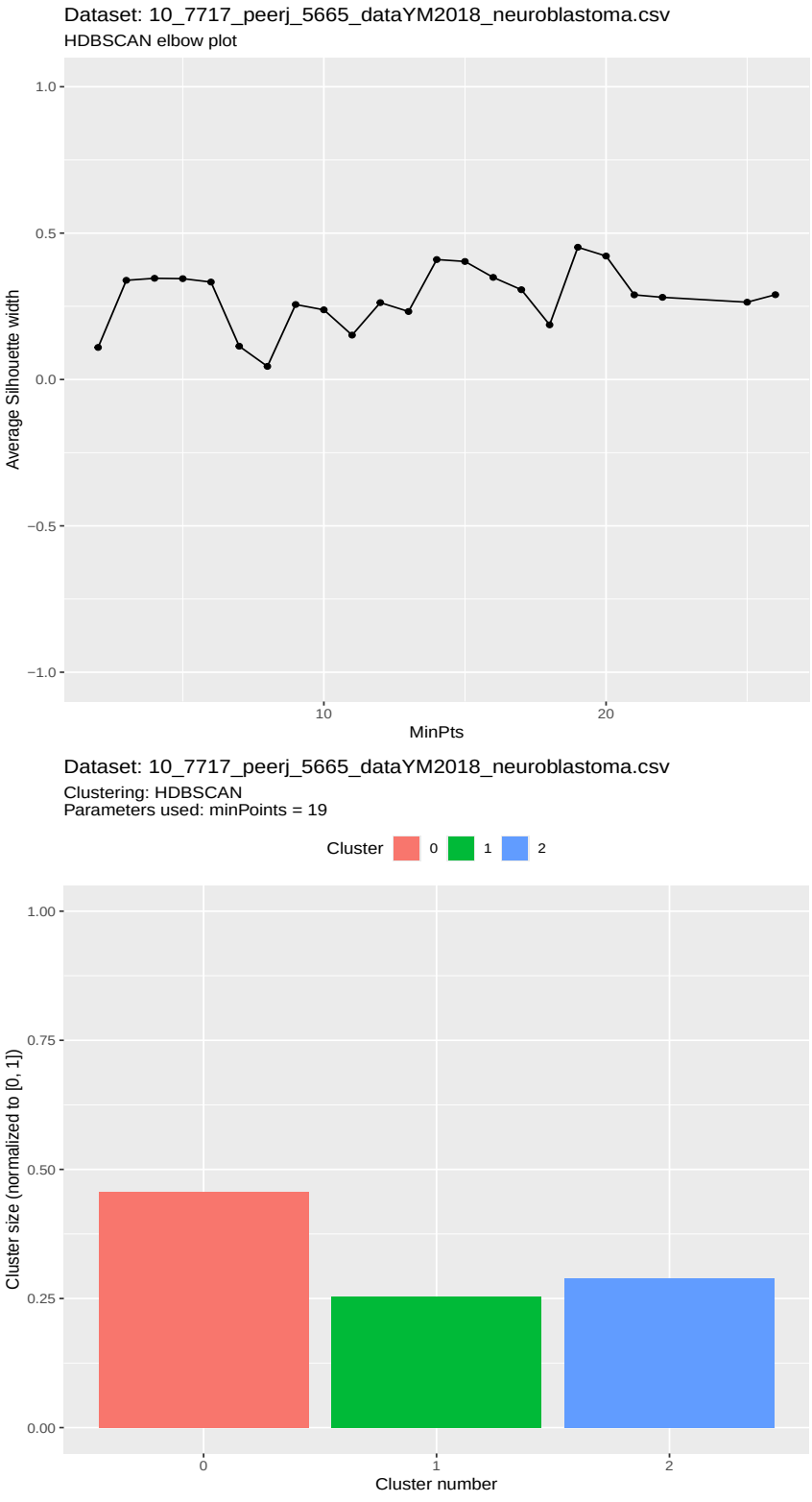


Figura 4.17: Risultati dell'algoritmo HDBSCAN per il dataset `results_10_7717_peerj_5665_dataYM2018_neuroblastoma.csv`

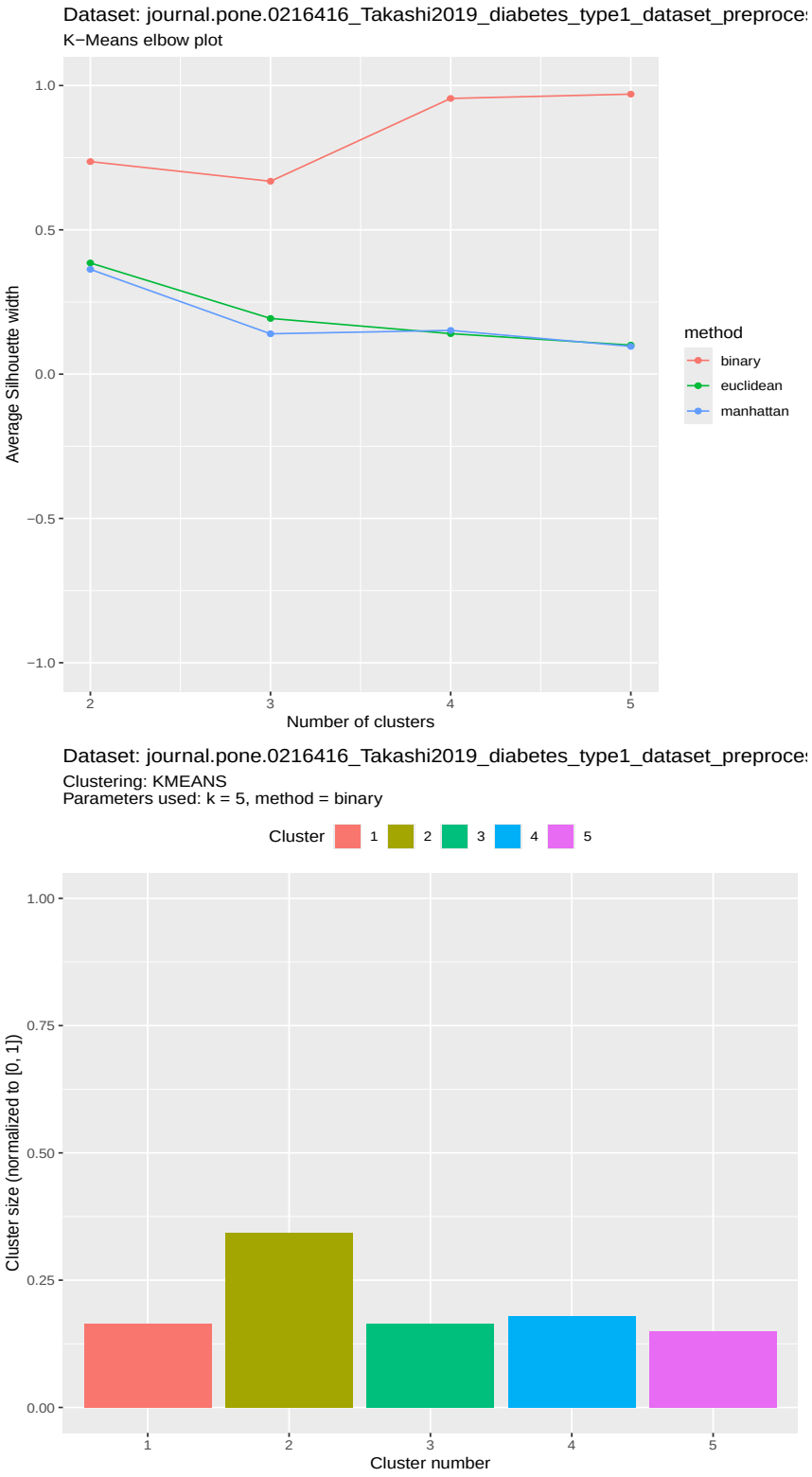


Figura 4.18: Risultati dell'algoritmo K-Means per il dataset `results_journal.pone.0216416_Takashi2019_diabetes_type1_dataset_preprocessed.csv`

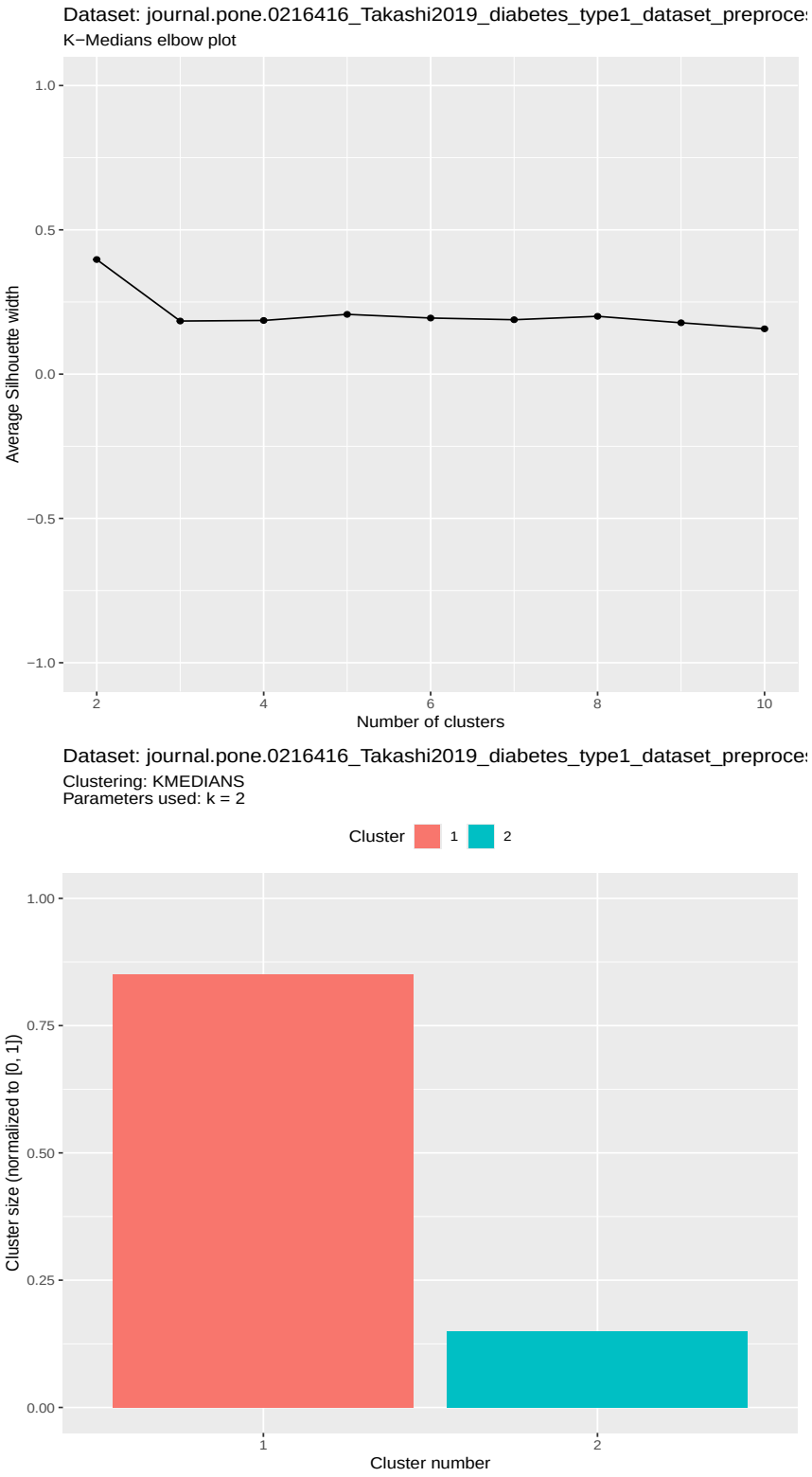


Figura 4.19: Risultati dell'algoritmo K-Medians per il dataset `results_journal.pone.0216416_Takashi2019_diabetes_type1_dataset_preprocessed.csv`

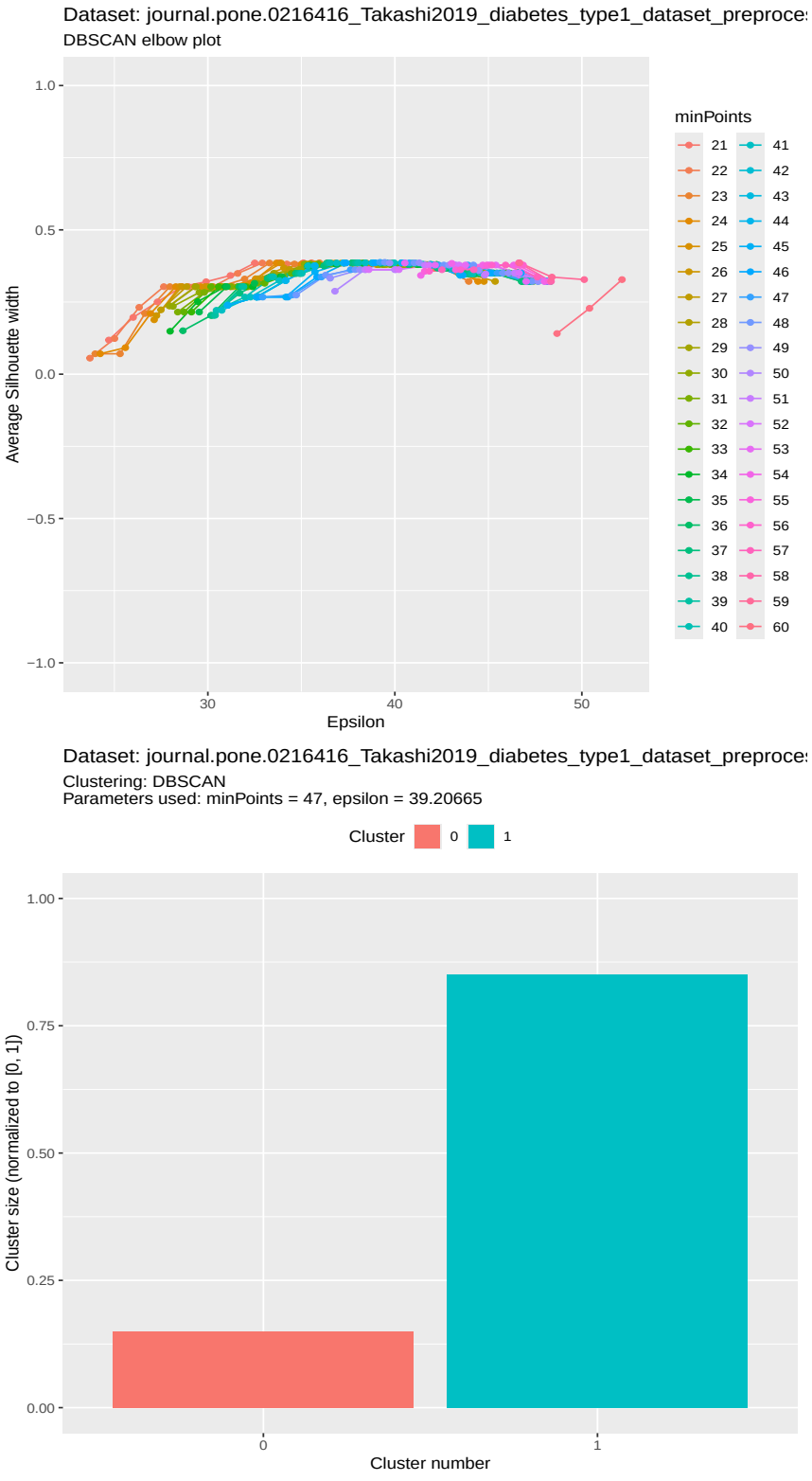


Figura 4.20: Risultati dell'algoritmo DBSCAN per il dataset `results_journal.pone.0216416_Takashi2019_diabetes_type1_dataset_preprocessed.csv`

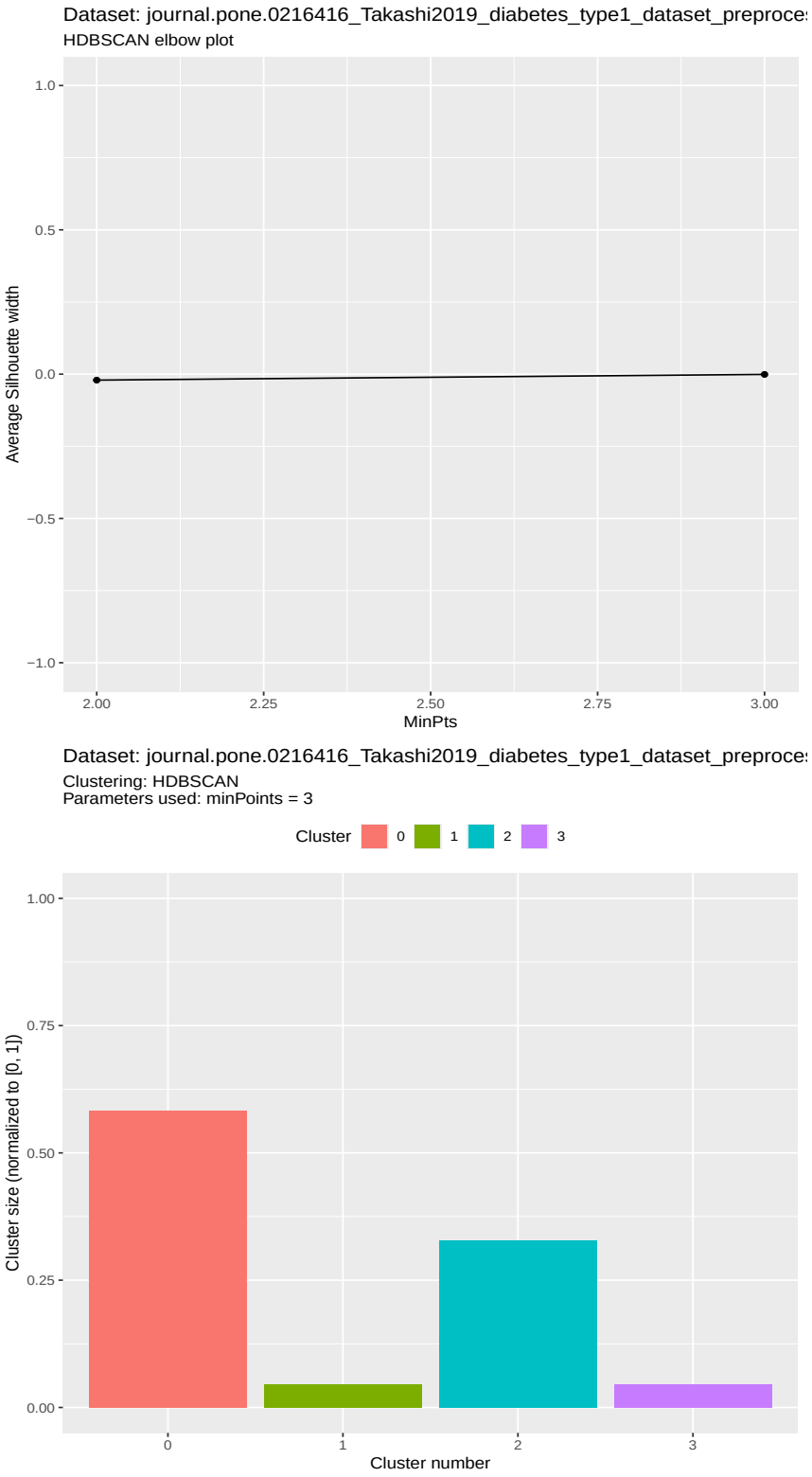


Figura 4.21: Risultati dell'algoritmo HDBSCAN per il dataset `results_journal.pone.0216416_Takashi2019_diabetes_type1_dataset_preprocessed.csv`

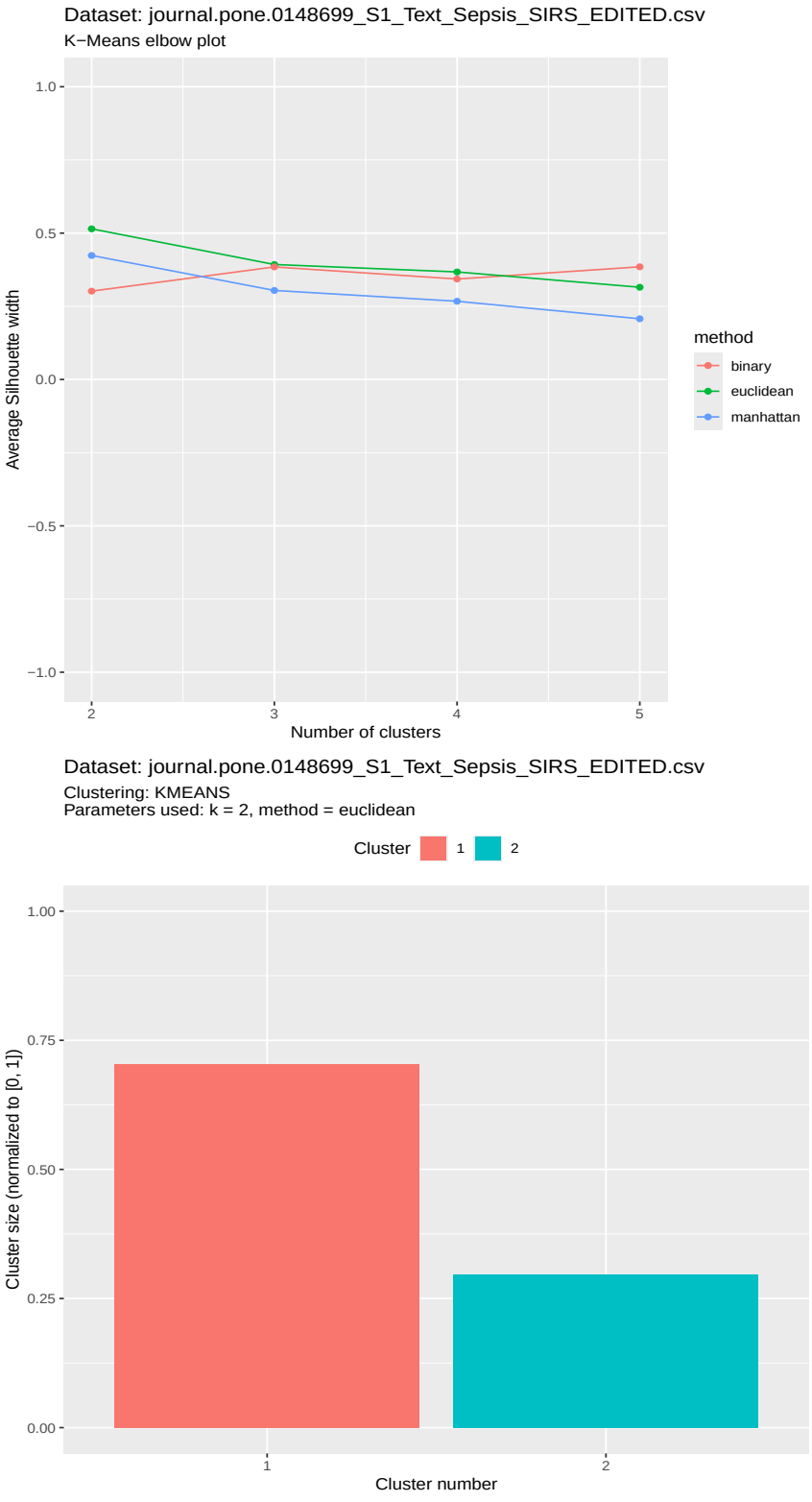
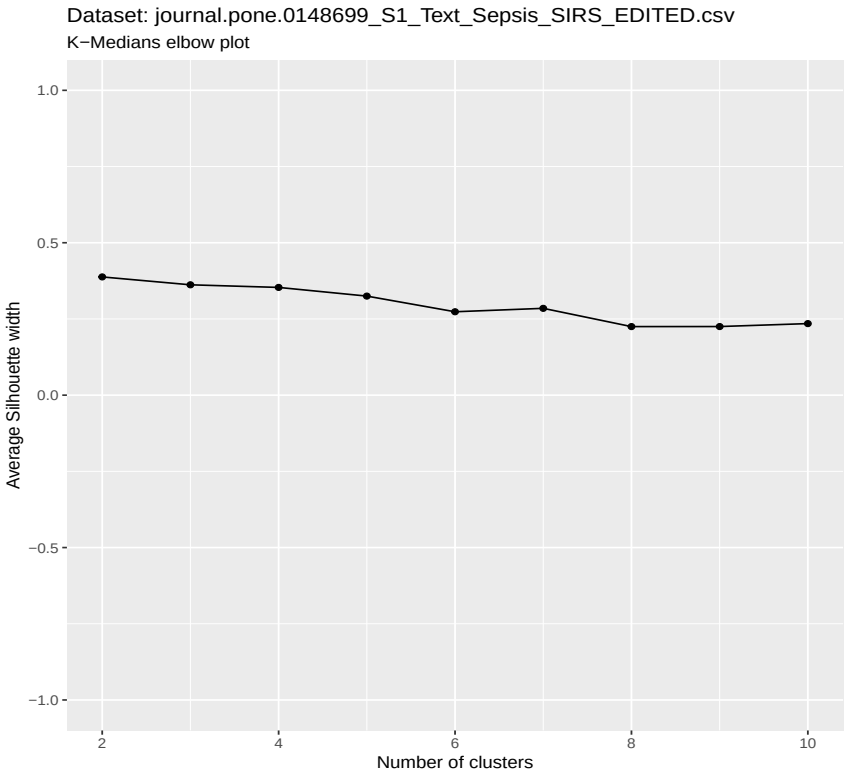


Figura 4.22: Risultati dell'algoritmo K-Means per il dataset `results_journal.pone.0148699_S1_Text_Sepsis_SIRS_EDITED.csv.csv`



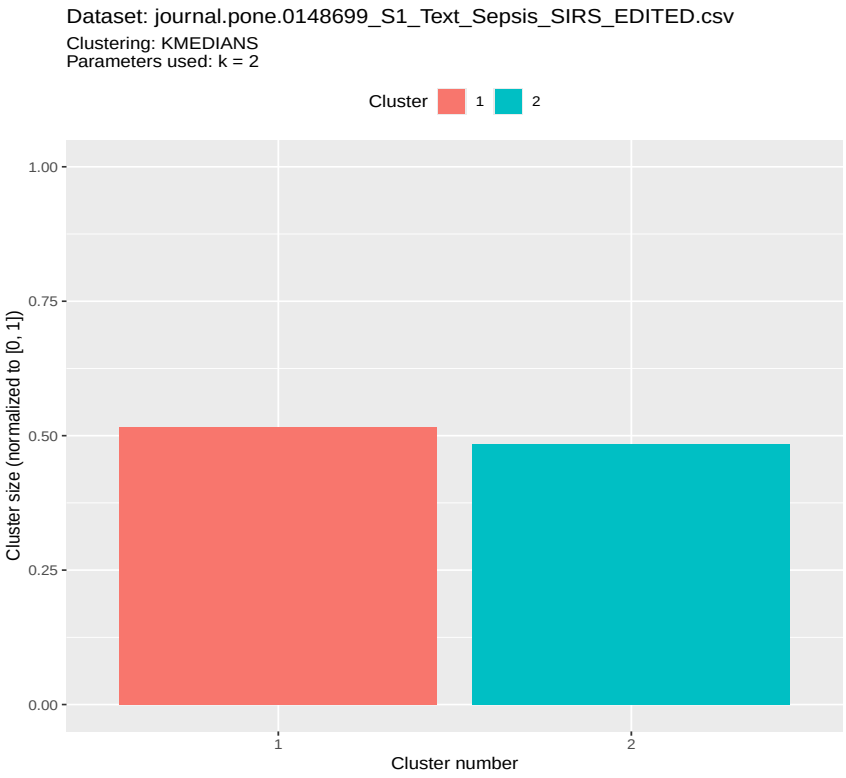


Figura 4.23: Risultati dell'algoritmo K-Medians per il dataset `results_journal.pone.0148699_S1_Text_Sepsis_SIRS_EDITED.csv.csv`

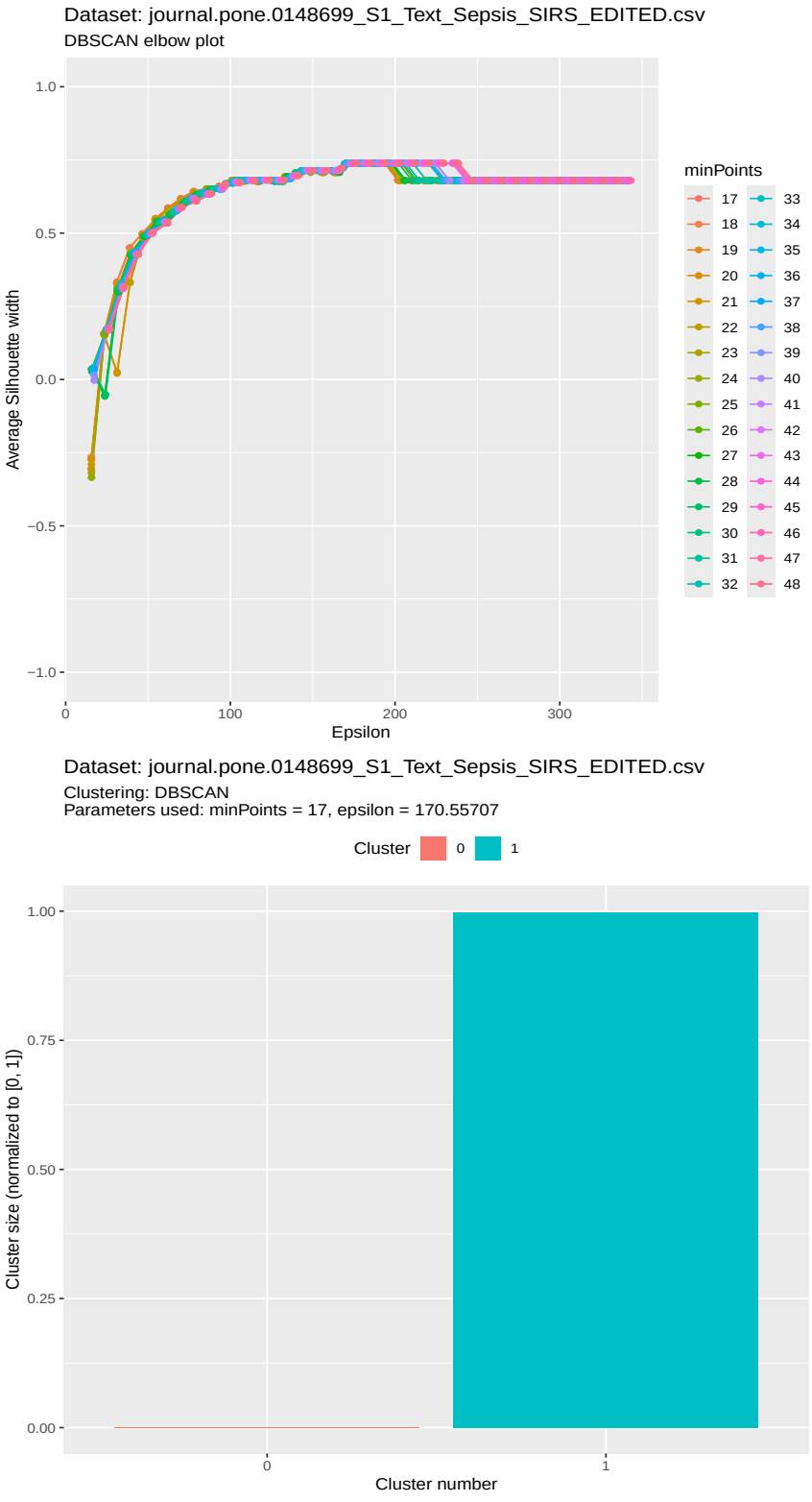


Figura 4.24: Risultati dell'algoritmo DBSCAN per il dataset `results_journal.pone.0148699_S1_Text_Sepsis_SIRS_EDITED.csv.csv`

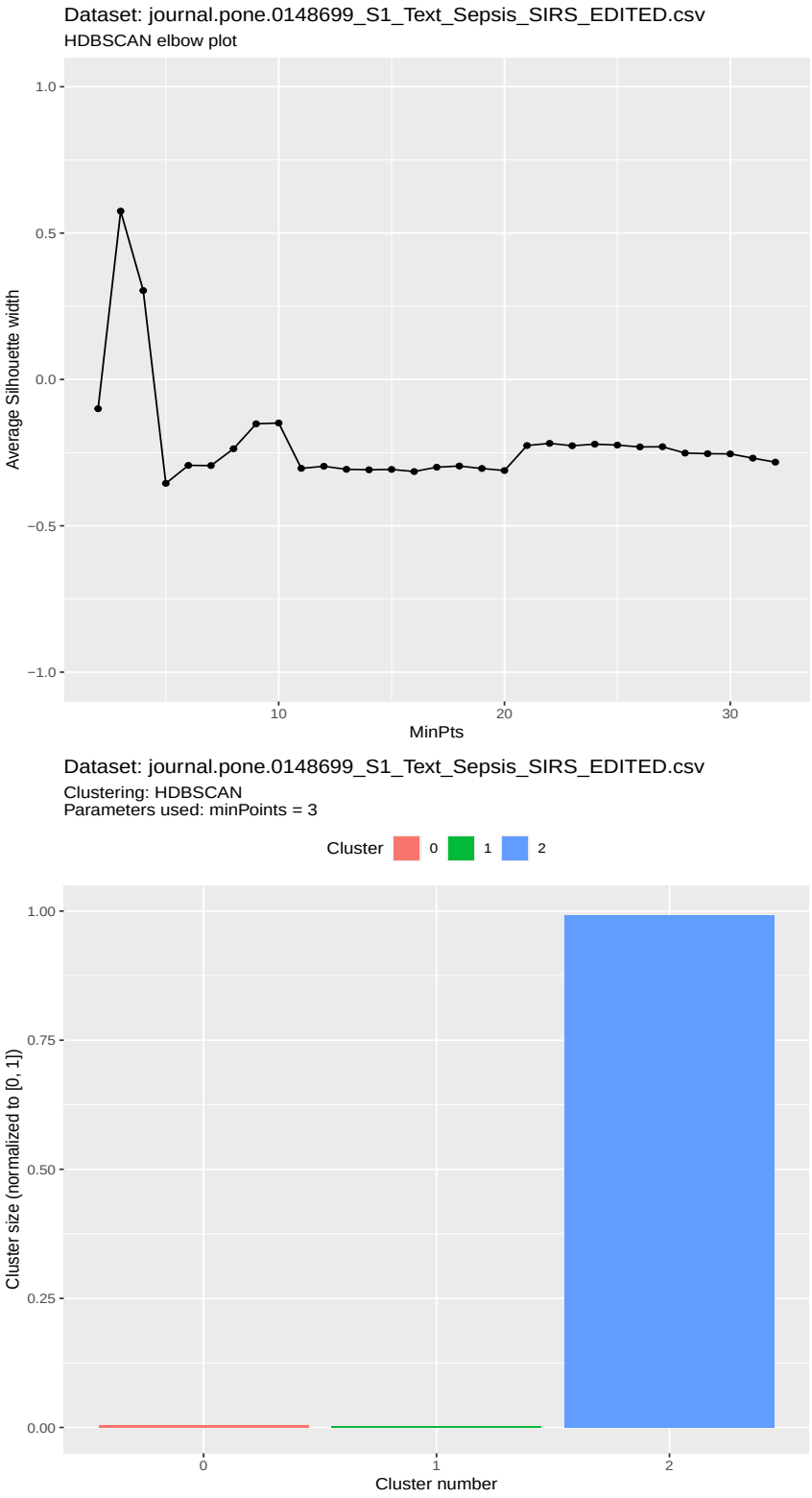


Figura 4.25: Risultati dell'algoritmo HDBSCAN per il dataset `results_journal.pone.0148699_S1_Text_Sepsis_SIRS_EDITED.csv.csv`

5. Discussione

5.1 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.

5.2 Pacchetti

Alla luce dei due test considerati, il pacchetto `Kira` é 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 é stato preferito `cluster`, perché fra i due era quello con il tempo di esecuzione piú basso. Il pacchetto `scikit-learn` attraverso `reticulate` non é 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 é inferiore, sia perché il pacchetto `cluster` si trova spesso giá incluso nelle installazioni di `R` (garanzia di affidabilitá) sia perché é 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 Figure 5.1 è presentata la differenza fra `cluster` e `scikit-learn` sul test matrice binaria. Come è possibile notare, la differenza fra i due è contenuta.

5.3 EHR

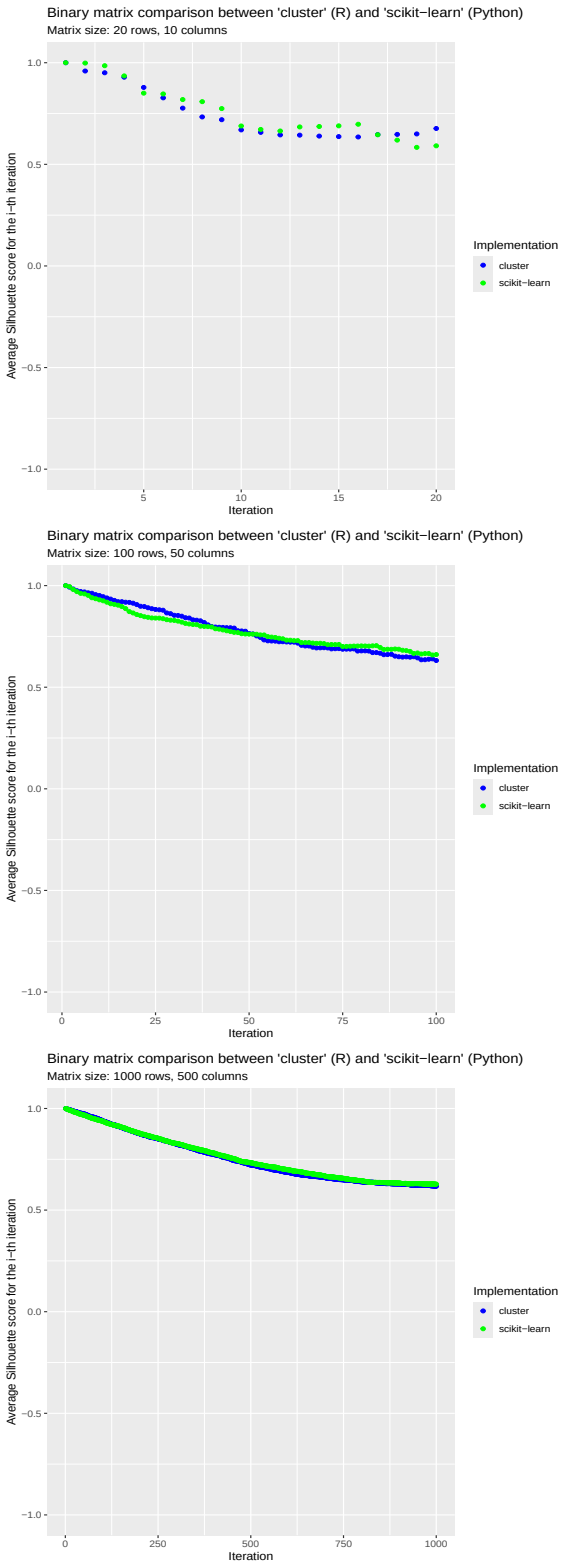


Figura 5.1: Differenze fra il test matrice binaria tra `cluster` (pacchetto R) e `scikit-learn` (pacchetto Python).

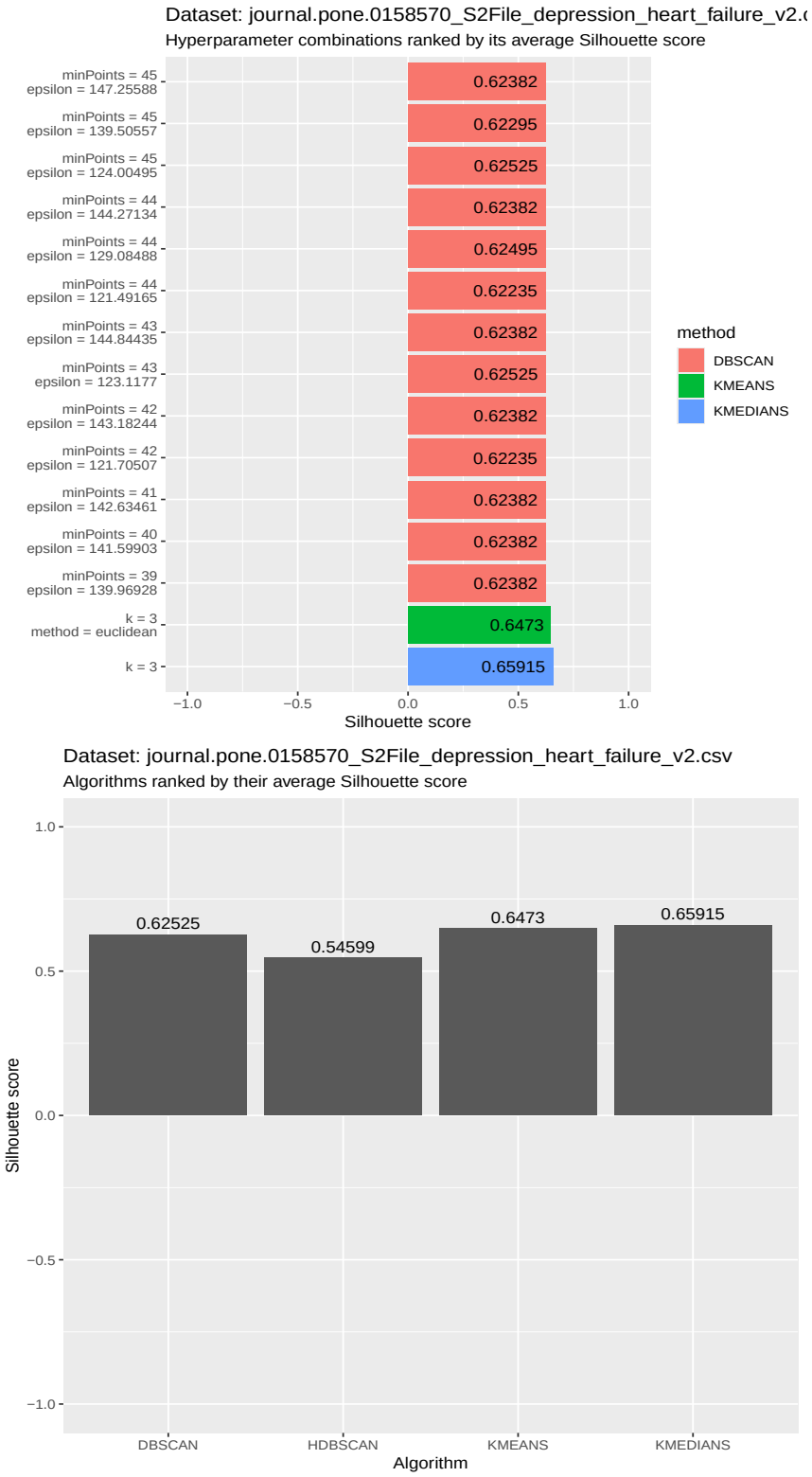


Figura 5.2: Riassunto dei risultati dei vari algoritmi per il dataset `results_journal.pone.0158570_S2File_depression_heart_failure_v2.csv`

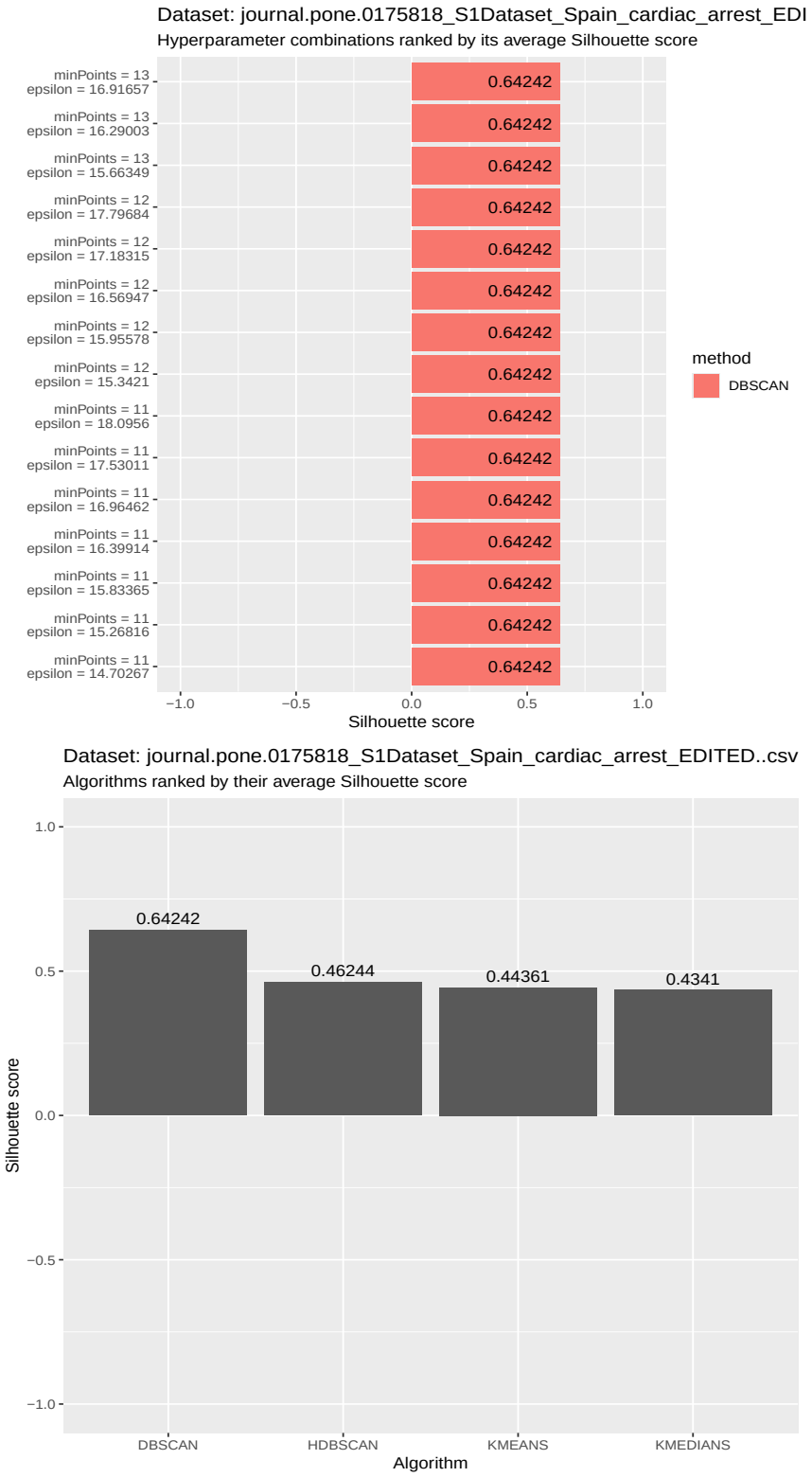


Figura 5.3: Riassunto dei risultati dei vari algoritmi per il dataset `results_journal.pone.0175818_S1Dataset_Spain_cardiac_arrest_EDITED..csv`

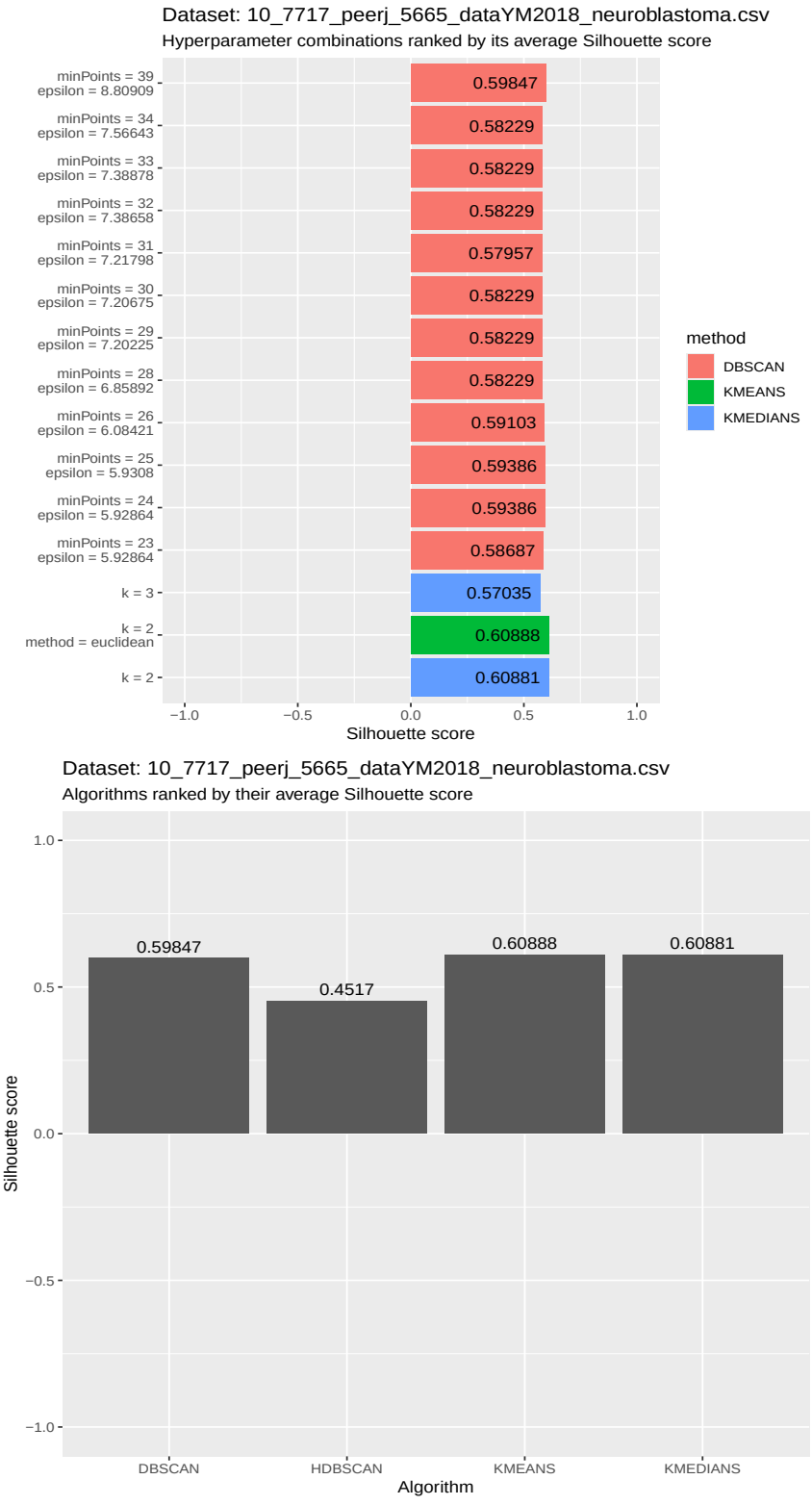


Figura 5.4: Riassunto dei risultati dei vari algoritmi per il dataset `results_10_7717_peerj_5665_dataYM2018_neuroblastoma.csv`

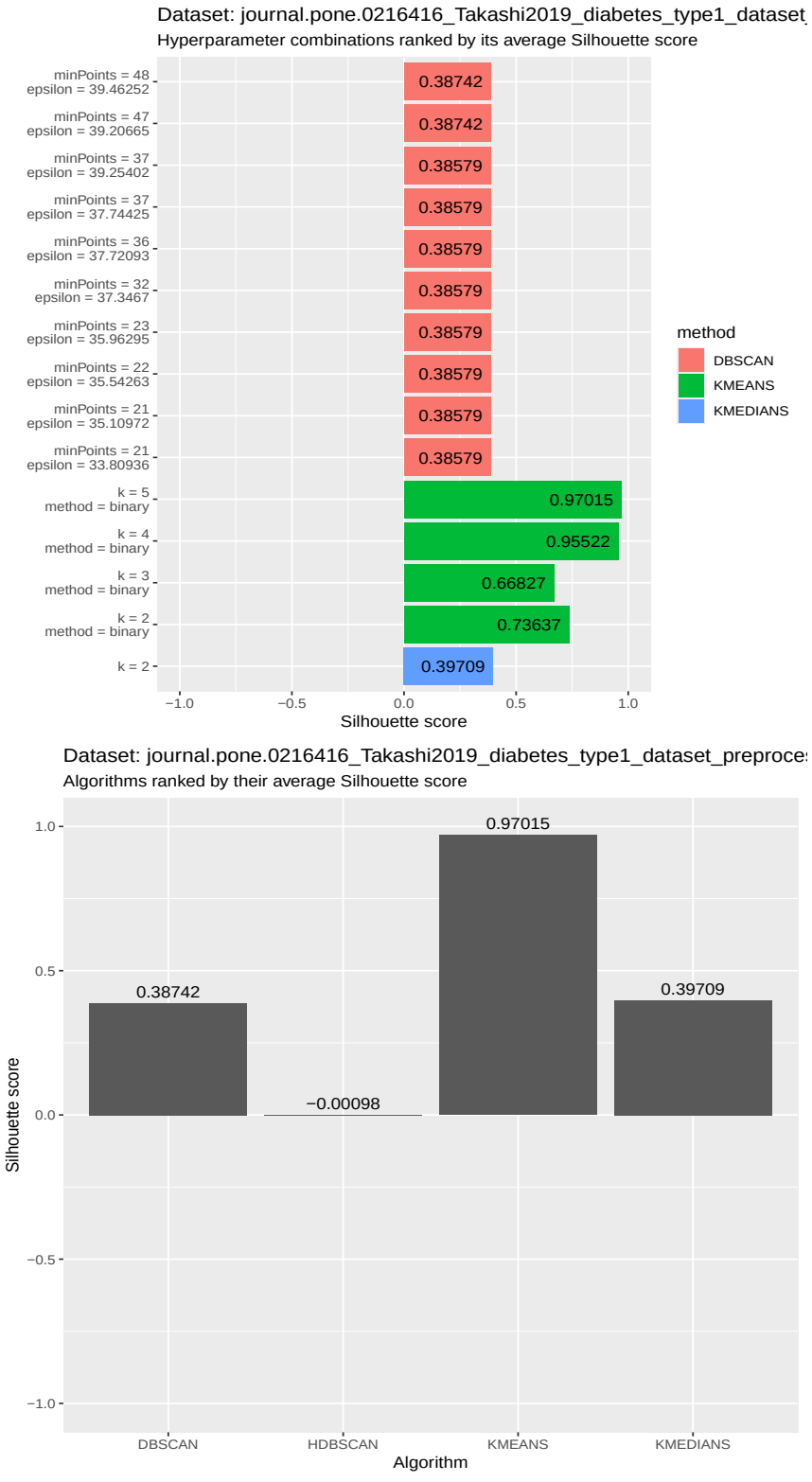


Figura 5.5: Riassunto dei risultati dei vari algoritmi per il dataset `results_journal.pone.0216416_Takashi2019_diabetes_type1_dataset_preprocessed.csv`

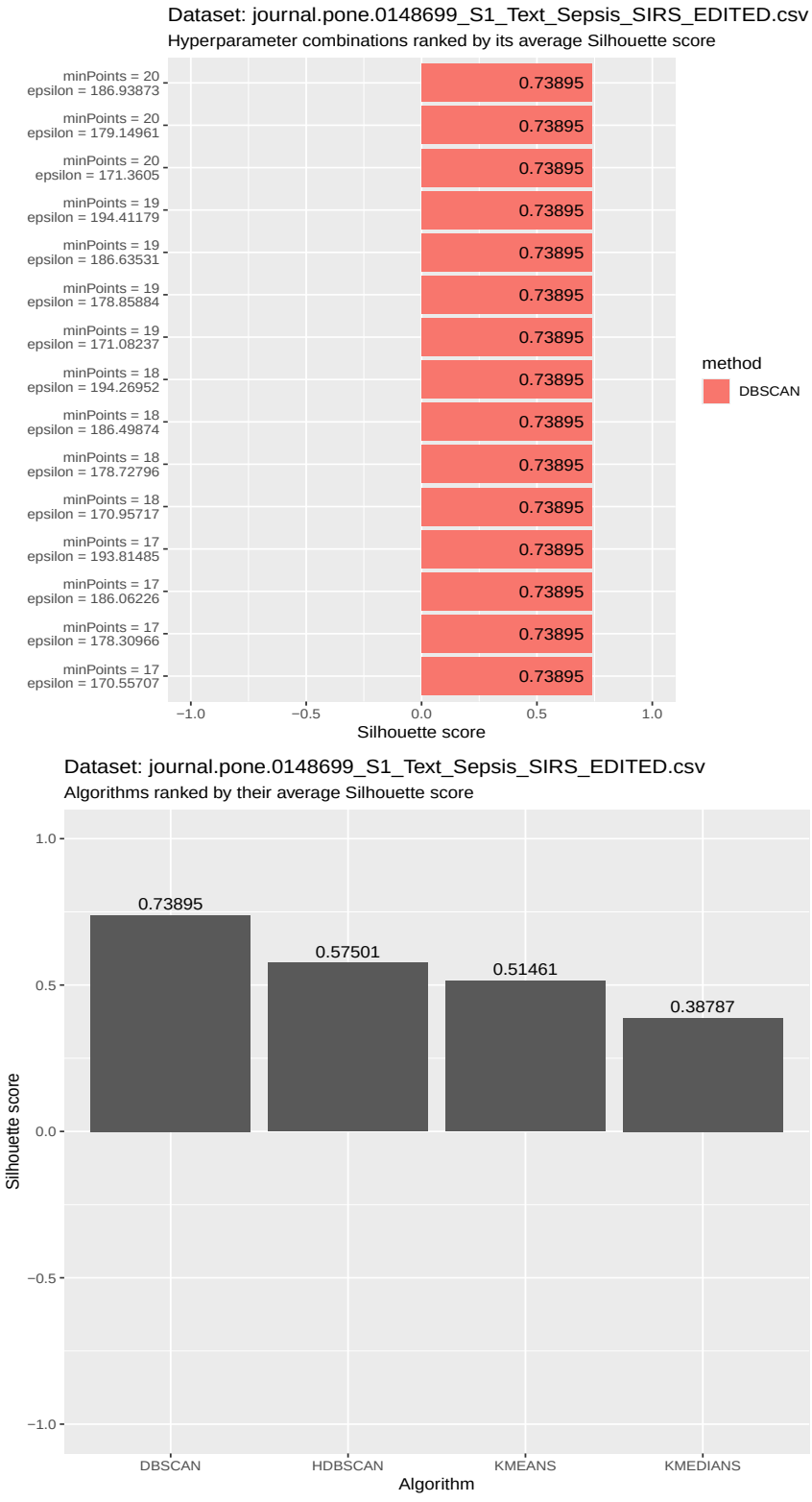


Figura 5.6: Riassunto dei risultati dei vari algoritmi per il dataset `results_journal.pone.0148699_S1_Text_Sepsis_SIRS_EDITED.csv.csv`

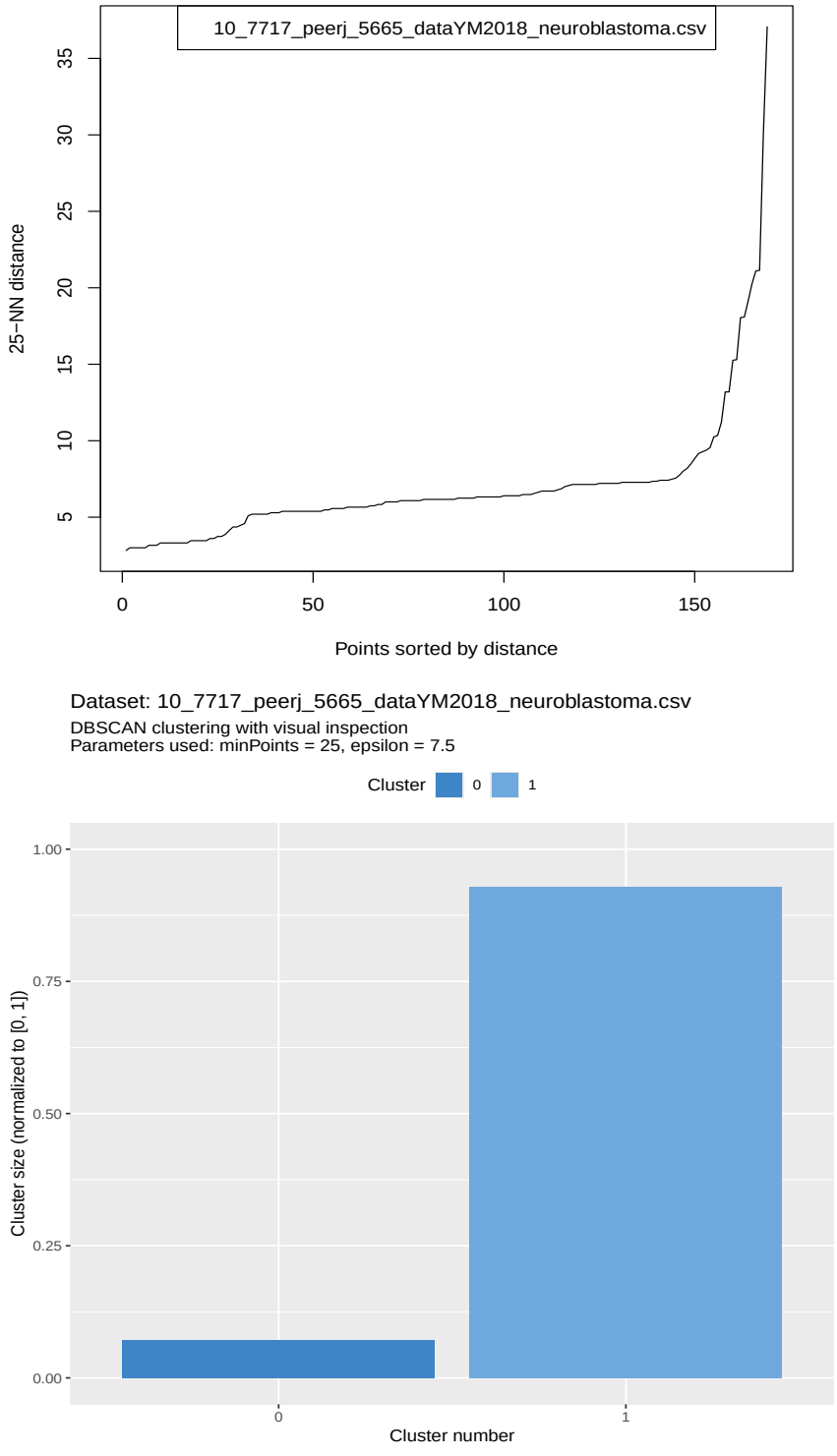


Figura 5.7: Risultati dell'algoritmo DBSCAN mediante KNN-plot per il dataset `results_journal.pone.0158570_S2File_depression_heart_failure_v2.csv`

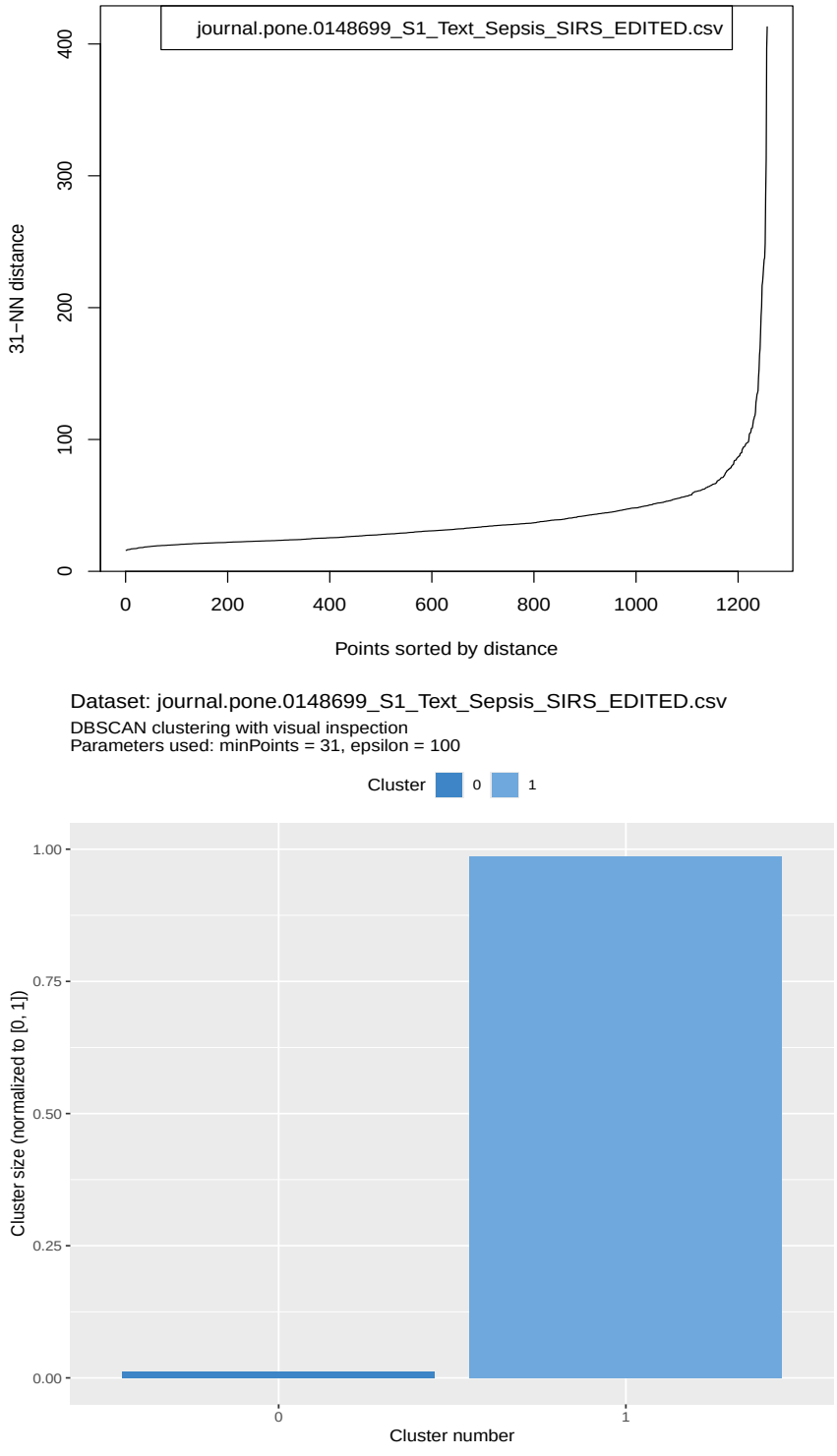


Figura 5.8: Risultati dell'algoritmo DBSCAN mediante KNN-plot per il dataset `results_journal.pone.0175818_S1Dataset_Spain_cardiac_arrest_EDITED..csv`

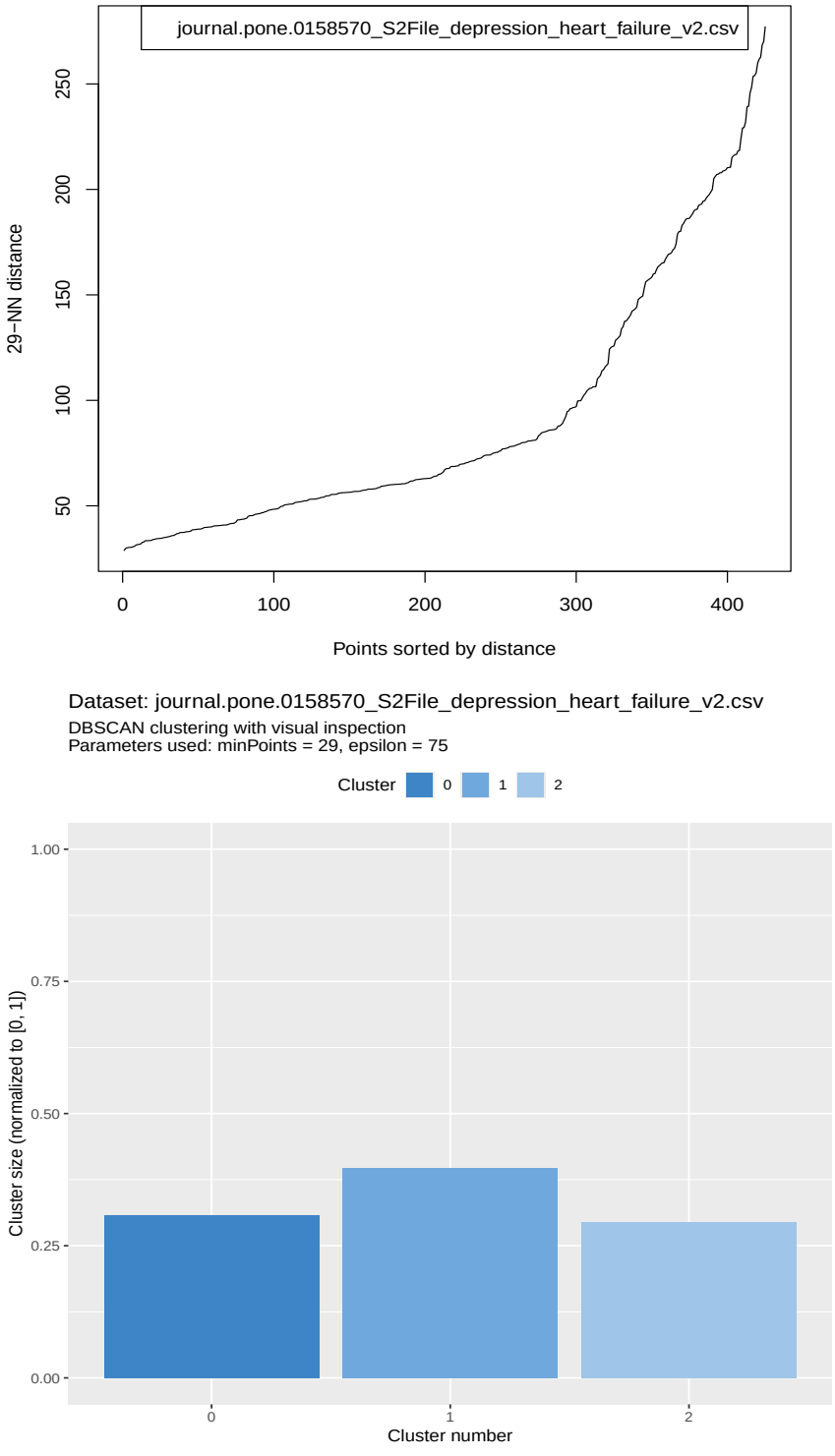


Figura 5.9: Risultati dell'algoritmo DBSCAN mediante KNN-plot per il dataset `results_10_7717_peerj_5665_dataYM2018_neuroblastoma.csv`

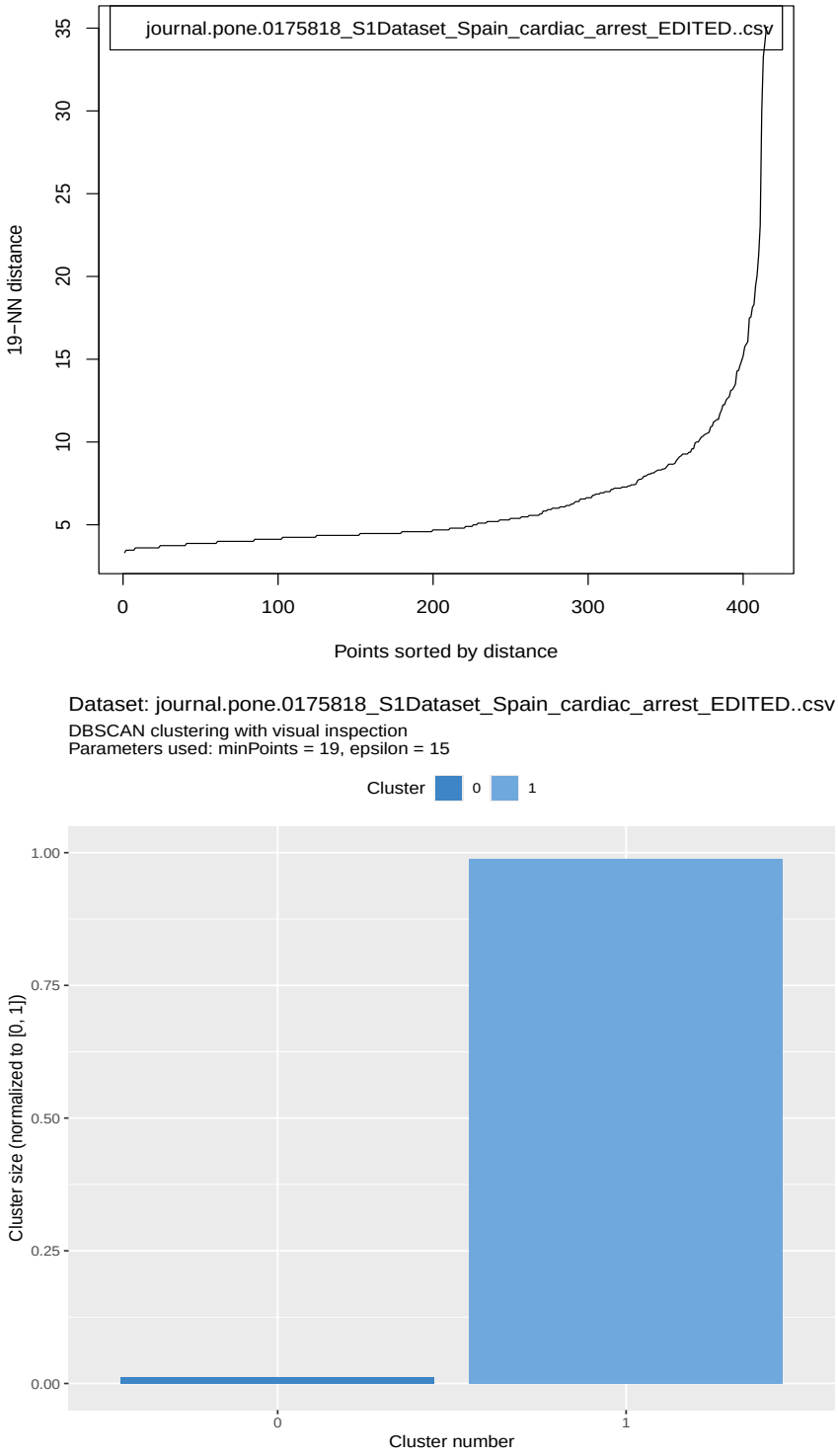


Figura 5.10: Risultati dell'algoritmo DBSCAN mediante KNN-plot per il dataset `results_journal.pone.0216416_Takashi2019_diabetes_type1_dataset_preprocessed.csv`

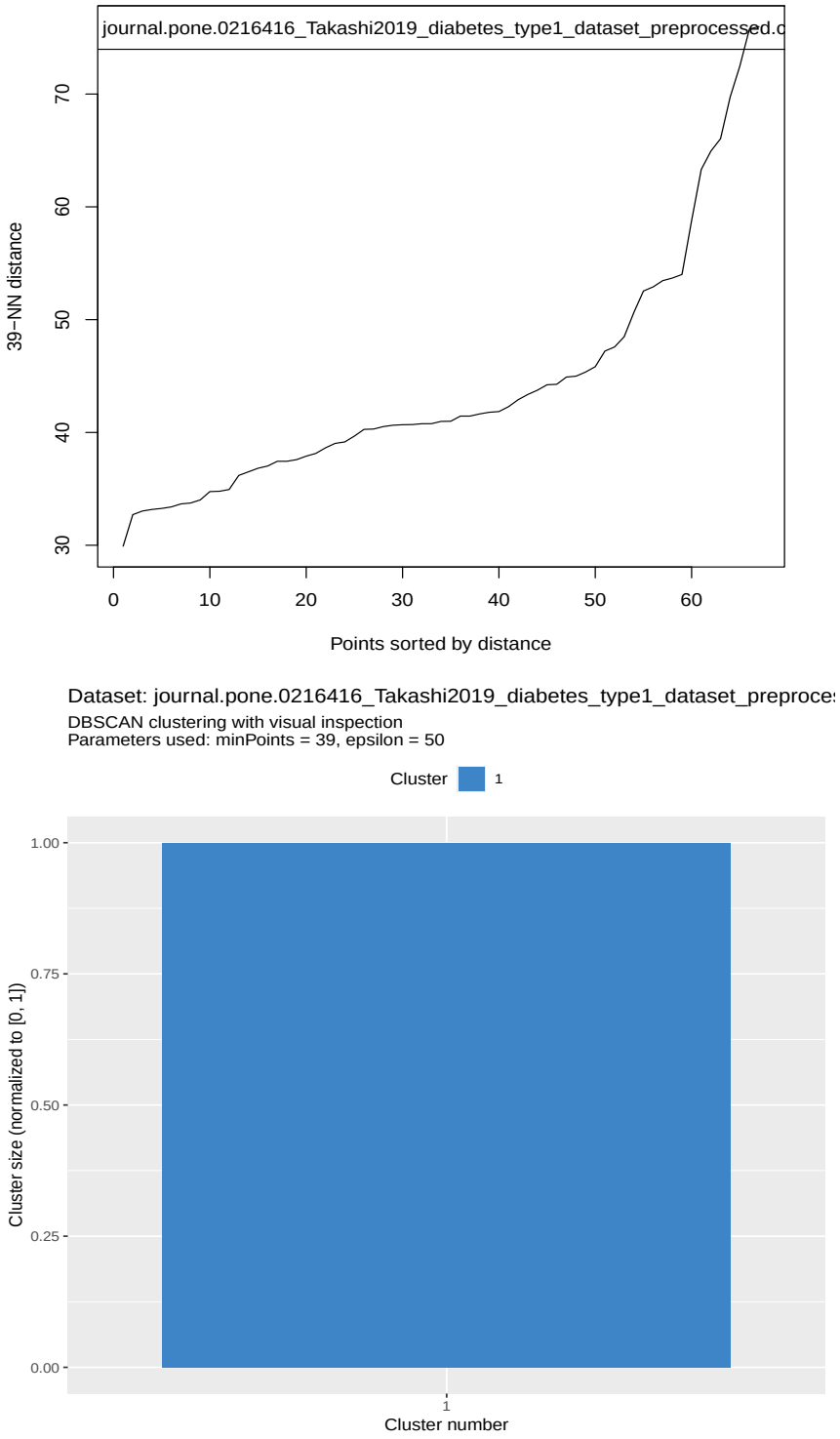


Figura 5.11: Risultati dell'algoritmo DBSCAN mediante KNN-plot per il dataset `results_journal.pone.0148699_S1_Text_Sepsis_SIRS_EDITED.csv.csv`

6. Conclusioni

Gli algoritmi sono stati applicati su dataset ”grezzi”, ovvero senza operare alcun preprocessing che non fosse l'eliminare ogni entry che avesse almeno un dato mancante. Si potrebbe ripetere gli esperimenti adottando un approccio piú conservativo, ad esempio sostituendo tali dati con valori di default ed osservare se si presentano differenze.

Sarebbe inoltre interessante ripetere gli esperimenti operando dimensionality reduction (*principal component analysis*, ecc...), anche e soprattutto per poter visualizzare graficamente il risultato del clustering pur essendo stato questo fatto su dati n-dimensionali.

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] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.