Politecnico di Torino

Politecnico di Torino

Corso di laurea magistrale in Ingegneria Informatica

A.a. 2020/2021

Sessione di Laurea Dicembre 2021

Enabling multidimensional time series clustering using signal embeddings

Clustering di time series multivariate utilizzando document encoder convoluzionali e tecniche NLP come metodi di embedding

Relatori:

Giordano Danilo

Cagliero Luca

Candidato:

Drago Attilio

Matricola: s265715

Abstract

L'obiettivo del lavoro di tesi è l'implementazione e la validazione di nuovi approcci per il clustering di time series multivariate. L' approccio sviluppato utilizza tecniche di machine learning per creare sottospazi di embedding che si adattino alle caratteristiche specifiche di ogni dataset considerato. Tramite l'utilizzo di document encoder basati su reti neurali e algoritmi di natural language processing riadattati per il task oggetto di studio, puntiamo a creare vettori di embedding che siano in grado di sintetizzare in maniera accurata le caratteristiche delle multivariate time series per poter poi impiegare i principali algoritmi di clustering. Gli esperimenti preliminari sono condotti su insiemi di dati provenienti da sensori posti su dei volontari durante lo svolgimento di alcune attività e pertanto l'obiettivo è raggruppare le time series nelle rispettive attività. Le metodologie sviluppate hanno dimostrato di riuscire ad ottenere un buon grado di incorporazione delle time series in un unico vettore permettendo l'applicazione di metriche standard per poterne valutare la similarità e conseguentemente il raggruppamento in cluster, raggiungendo inoltre risultati di notevole interesse in termini di random index e silhouette.

Sommario

Con "*time series analysis*" si intende lo studio di problemi nei quali i dati sono organizzati in sequenze temporali. Grandi volumi di queste sequenze di dati appaiano in ogni campo di ricerca e industriali quali astronomia, biologia, meteorologia, medicina, finanza, robotica, ingegneria e altri. Tra le varie tecniche applicate per l'analisi di questa tipologia di dati il clustering è sicuramente una delle più studiate.

Con time series si intende un set di dati ordinati in maniera temporale in un certo periodo di tempo e possono esserci due tipi di time series: in uno le series temporali sono indipendenti e quindi una sola serie è utilizzata per misurare un dato fenomeno, nell'altro vengono registrate più series temporali insieme per monitorare lo stesso fenomeno.

Esistono di conseguenza due principali tecniche di analisi per le time series: analisi di time series univariate, nelle quali è soltanto una variabile a variare nel tempo e quindi ad essere analizzata e analisi di time series multivariate nella quali si analizzano più serie alla volta.

Con il *clustering* possiamo identificare e riassumere interessanti pattern e correlazioni nei dati oggetto di studio senza una conoscenza a priori del loro raggruppamento. Questo raggruppamento è basato solo sulla misura della somiglianza dei campioni del dataset preso in esame e l'obiettivo è collocare nello stesso insieme gli elementi più simili e in altri insiemi i campioni meno simili.

Nell'ambito univariato esistono già modelli solidi e ampiamente affermati, tuttavia nell'analisi multivariata le principali metodologie di clustering applicate alle univariate time serie risultano inefficienti, poco accurate o spesso impossibili da estendere. Una delle maggiori criticità dell'analisi univariata è che analizzando le time series separatamente si perdono delle correlazioni tra i dati deteriorando pertanto la qualità dell'analisi e conseguentemente i risultati ottenibili.

Per questi motivi, in questo lavoro proponiamo lo studio di nuovi metodi di *embedding* di time series multivariate che siano in grado di integrarsi validamente nel processo di clustering ottenendo buoni risultati nel nuovo spazio.

A differenza degli approcci già esistenti, il nostro sviluppa metodologie che, sfruttando le potenzialità dei nuovi approcci di machine learning, siano in grado di creare spazi vettoriali attraverso approcci "*data-driven*", ossia che si adattino alle caratteristiche intrinseche dell'insieme di dati su cui vengono applicati.

In questo lavoro di tesi l'obiettivo principale è quindi raggruppare le time series multivariate; per farlo proponiamo delle metodologie per l'elaborazione di una rappresentazione unificata delle serie temporali multivariate in uno spazio *vettoriale high-dimensional* tramite tecniche che sfruttino le reti neurali convoluzionali e gli algoritmi di *document embedding* in modo da poter valutare in modo quantitativo la similarità tra serie multivariate e poter infine applicare algoritmi di clustering tradizionali direttamente sull'encoding delle serie multivariate.

Andando più nel dettaglio studieremo due principali approcci: uno basato su *document encoder* implementati tramite reti neurali convoluzionali e un altro basato sull'algoritmo *doc2vec*, un algoritmo molto diffuso in ambito del document embedding per l'analisi di dati testuali.

La tesi è suddivisa in tre parti principali riassumibili come segue.

La prima è una breve ricapitolazione degli argomenti teorici utili a capire il lavoro. La seconda presenta le metodologie utilizzate, i dataset e come sono stati processati. Nella terza vengono esposti gli esperimenti e risultati ottenuti e infine le conclusioni e possibili sviluppi futuri.

Non abbiamo svolto un importante lavoro di preprocessamento sui dati in quanto il nostro obiettivo è stato in larga parte studiare il comportamento degli algoritmi di *embedding* applicati alle time series multivariate. Le uniche operazioni svolte sui dati sono state il *windowing*, ovvero la divisione dell'intere time series in finestre temporali più brevi e, eventualmente, operazioni di normalizzazione per l'approccio con reti neurali o discretizzazione per l'approccio con algoritmi *di natural language processing*. In seguito, abbiamo studiato la distribuzione dei dati tra le varie time series appena suddivise.

La parte successiva è il nucleo della ricerca. Dopo la breve fase di preprocessamento e analisi passiamo ad allenare i modelli per *l'embedding*. In questa fase avviene il tuning dei parametri caratteristici dei modelli presi in esame con l'ausilio di grafici e metriche ad-hoc per lo scopo. Generiamo successivamente dei vettori di embedding, creando di fatto un nuovo dataset che utilizzeremo per l'addestramento di classici algoritmi di clustering. In questa fase finale facciamo un ulteriore tuning di questi modelli validando i risultati con gli indici di validità più comunemente utilizzati nell'ambito del clustering.

Gli approcci utilizzati richiedono di un dataset abbastanza vasto di time series in quanto è necessaria una buona variabilità dei dati perché questi raggiungano risultati soddisfacenti e, soprattutto, attendibili e generali. Pertanto, abbiamo scelto tra dataset che presentassero un numero elevato di samples dai quali estratte un numero soddisfacente di time series. Il dataset scelto contiene misurazioni provenienti da sensori tri-assiali posti in diverse parti del corpo di alcuni volontari durante lo svolgimento di alcune attività, quali corsa, passeggio ecc.

Successivamente alla fase di preprocessamento dei dati, il flusso si divide in due parti, una inerente alle metodologie che sfruttano reti neurali convoluzionali, l'altra agli algoritmi doc2vec.

Nel primo approccio viene allenato un document encoder tramite un approccio unsupervised avendo come obiettivo quello di minimizzare l'errore di ricostruzione dell'output del decoder. Dopo la fase di allenamento si utilizza l'output dell'encoder appena addestrato per generare un

nuovo dataset costituito dai vettori di embedding delle time series. Come funzione di perdita viene utilizzata una combinazione di diverse funzioni di perdita, ognuna con un proprio peso.

I vettori estratti vengono poi clusterizzati utilizzando algoritmi di clustering standard quali: K-Means, clustering gerarchico agglomerativo, DBscan, K-Medoids e due diverse distanze, euclidea e del coseno. Per ognuno di questi viene fatto il tuning dei principali parametri, tramite grid-search.

L'altro approccio è basato su un adattamento di doc2vec.

Doc2vec è un algoritmo non supervisionato che ha lo scopo di generare vettori per frase/paragrafi/documenti. L'algoritmo è un adattamento di word2vec, un altro famoso algoritmo pensato per generare vettori di embedding per le parole. I vettori generati da doc2vec possono quindi essere utilizzati per misurare la somiglianza tra frasi/paragrafi/documenti.

Per quanto riguarda l'approccio con algoritmi doc2vec utilizziamo invece le due principali implementazioni chiamate PV-DBOW e PV-DM.

I nuovi vettori di embedding vengono quindi clusterizzati coi medesimi algoritmi introdotti precedentemente.

Ogni fase prevede il tuning degli iperparametri di tutti i modelli menzionati.

La qualità dei cluster generati ad ogni step viene validata quantitativamente tramite tre metriche: *random index, adjusted random index e silhouette*. Le prime due misurano la qualità del raggruppamento utilizzando informazioni esterne, ossia a quale cluster effettivamente appartengano le time series. Questi indici variano tra 0 (raggruppamenti totalmente sbagliati) a 1 (tutti gli elementi raggruppati perfettamente). Il secondo misura quanto gli elementi appartenenti al medesimo cluster siano simili tra di loro (coesione) rispetto agli elementi appartenenti ad altri cluster (separazione) senza utilizzare informazioni esterne, variando in un range compreso tra -1 e 1.

Per entrambi gli approcci i risultati ottenuti sono notevoli poiché hanno dimostrato di raggiungere valori molto alti di random index (circa 0.8) e di adjusted random index (fino a 0.5).

Detto ciò, potrebbe essere interessante approfondire le metodologie presentate in questo progetto tesi in lavori futuri, utilizzando un numero maggiore di dataset e testando anche le performance di algoritmi di clustering più complessi in modo da avere una panoramica generalizzata sui punti di forza e debolezza di questi approcci.

.....1
Abstract.....3
Sommario.....3
Introduzione.....11
IoT, industria 4.0 e BigData.....11
PARTE I.....11
Time series analysis.....11
Univariate Time series.....11
Multivariate Time series.....11
Machine learning.....12
Tipi di problemi e task.....12
Apprendimento supervisionato.....13
Reti neurali nell' apprendimento supervisionato.....14
Convolutional neural network.....18
Apprendimento non supervisionato.....19
Algoritmi di Clustering.....19
Validare i risultati di un algoritmo di cluster.....23
Reti neurali convoluzionali nell'apprendimento non supervisionato.....24
Algoritmi di Natural Language Processing.....26
Time series clustering.....29
Clustering per time series multivariate.....30
PARTE II.....30
Metodo proposto.....30
Obiettivi della tesi.....30
Metodologie applicate.....31
Piattaforma e librerie usate.....32
Dataset utilizzati.....33
Analisi del dataset.....33

Pipeline del lavoro 39
Approccio con autoencoder 39
Approccio Sig2vec....42
Analisi per curse of dimensionality 45
Analisi per K selection....46
Clustering e validazione....47
PARTE III....48
Sig2vec 48
Distanza Euclidea....51
Coseno 63
Convolutional neural network autoencoder 71
Euclidean 72
Cosine 92
Risultati DBSCAN....104
Comparazione con altri paper 105
Conclusioni e sviluppi futuri....111
Bibliografia 113

Indice delle tabelle

Tabella 1 Risultati medi al variare dell'algoritmi doc2vec con distanza euclidea....48
Tabella 2 Risultati medi al variare dell'algoritmi doc2vec con distanza del coseno 48
Tabella 3 Risultati medi al variare della grandezza del vettore di embedding con distanza euclidea49
Tabella 4 Risultati medi al variare della grandezza del vettore di embedding con distanza del coseno 49
Tabella 5 Risultati medi al variare della grandezza della finestra con distanza euclidea....49
Tabella 6 Risultati medi al variare della grandezza della finestra con distanza del coseno 49
Tabella 7 Risultati medi al variare dell'algoritmo di clustering con distanza euclidea 49
Tabella 8 Risultati medi al variare dell'algoritmo di clustering con distanza del coseno 50
Tabella 9 Risultati migliori per sig2vec con distanza euclidea....61
Tabella 10 Risultati migliori per DBSCAN per sig2vec con distanza euclidea....63
Tabella 11 Risultati migliori per sig2vec con distanza del coseno 69

Tabella 12 Risultati migliori per DBSCAN per sig2vec con distanza del coseno 71
Tabella 13 Risultati migliori per CNN con vettori concatenati e distanza euclidea 82
Tabella 14 Risultati migliori per CNN con secondo embedding e distanza euclidea 92
Tabella 15 Risultati migliori per CNN con vettori concatenati e distanza del coseno 98
Tabella 16 Risultati migliori per CNN con secondo embedding e distanza del coseno 104
Tabella 17 Risultati migliori per ogni metodologia 105

Indice delle figure

Figura 1 Grafici underfitting, right fitting e overfitting 13
Figura 2 Esempio di piano con support vector machine 14
Figura 3 Architetture di Reti neurali artificiali 15
Figura 4 Schematizzazione neurone aritficiale 15
Figura 5 Esempio di linear rectifier 16
Figura 6 Esempio softmax 17
Figura 7 Visualizzazione del processo di minimizzazione della funzione di costo 18
Figura 8 Esempio di layers di una rete neurale convoluzionale 19
Figura 9 Esempio di punti clusterizzati su un piano bidimensionale 20
Figura 10 Successione degli step dell'algoritmo di K-means 21
Figura 11 Rappresentazione a dentogramma del risultato di un algoritmo di clustering agglomerativo. 22
Figura 12 Rappresentazione di uno step del DBSCAN 23
Figura 13 Risultato del clustering con DBSCAN 23
Figura 14 Esempio architettura autoencoder 25
Figura 15 Esempio embedding in word2vec 27
Figura 16 Schematizzazione CBOW e Skip-Gram 27
Figura 17 Modello PV-DM estensione del modello CBOW 28
Figura 18 Model PV-DBOW 29
Figura 19 Embedding con autoencoder 31
Figura 20 Clustering dopo Embedding con autoencoder 32
Figura 21 Fase di embedding e clustering per sig2vec 32
Figura 22 Risultato comando describe 33
Figura 23 Risultato comando unique 34
Figura 24 Matrice di correlazione fra dimensioni 34
Figura 25 Grafici dei primi 4 secondi di ogni di attività svolta da un volontario 37
Figura 26 Distribuzione dati 37

Figura 27 Distribuzione valori dopo windowing 38
Figura 28 Training dell'autoencoder 41
Figura 29 Fasi di embedding per l'approccio con CNN 42
Figura 30 Fasi di discretizzazione e windowing per sig2vec 44
Figura 31Fase di embedding e clustering per sig2vec 44
Figura 32 Esempio grafico per analisi CUD 45
Figura 33 Grafico della K selection 47
Figura 34 Grafico delle distanze rispetto al quarto punto più vicino estratto da un nostro esperimento 48
Figura 35 Correlazione tra iperparemetri per distanza Euclidea e del coseno 50
Figura 36 Grafici a barre risultati metriche al variare delle combinazioni fra gli iperparametri 51
Figura 37 Curse of dimensionality Sig2vec con distanza euclidea 61
Figura 38 K-selection Sig2vec con distanza euclidea 61
Figura 39 Grafici risultati DBSCAN al variare dei parametri per sig2vec con distanza euclidea 62
Figura 40 Grafici a barre sig2vec con distanza del coseno al variare delle combinazioni fra iperparametri 64
Figura 41 Curse of dimensionality Sig2vec con distanza del coseno 69
Figura 42 K-selection Sig2vec con distanza del cosen 70
Figura 43 Grafici a barre dei risultati ottenuti con l'approccio CNN al variare delle combinazioni di iperparametriche e distanze utilizzate 72
Figura 44 Silhouette per cluster per Sig2vec 106
Figura 45 Pair confusion matrix per Sig2vec 106
Figura 46 Confusion matrix per Sig2vec 106
Figura 47 Silhouette per cluster per CNN con vettori concatenati 107
Figura 48 Pair confusion matrix per CNN con vettori concatenati 107
Figura 49 Confusion matrix per CNN con vettori concatenati 107
Figura 50 Silhouette per cluster per CNN con secondo embedding 108
Figura 51 Confusion matrix per CNN con secondo embedding 108
Figura 52 Pair confusion matrix per CNN con secondo embedding 108
Figura 53 Pair confusion matrix per ANN con layer bidirezionali 109
Figura 54 Silhouette per ANN con layer bidirezionali 109
Figura 55 Confusion matrix per ANN con layer bidirezionali 109
Figura 56 Silhouette per ANN con con soli fully connected layer 110
Figura 57 Pair confusion matrix per ANN con con soli fully connected layer 110
Figura 58 Pair confusion matrix per ANN con con soli fully connected layer 110

Introduzione

IoT, industria 4.0 e BigData

Oggigiorno stiamo assistendo ad una nuova rivoluzione industriale e il concetto di "Industria 4.0" è sempre più diffuso. Questo termine si riferisce all'applicazione di nuove tecnologie e metodi al mondo industriale e non solo, con il duplice obiettivo di migliorare la qualità del lavoro e produzione. Tra le nuove tecnologie e metodologie è necessario inserire alcune Tecniche di Data Mining, Internet of Things, soluzioni Cloud e anche i concetti di manutenzione predittiva e Smart Manufacturing.

Il termine comune di questi nuovi approcci è la grande quantità di dati che abbiamo oggi a disposizione e detto ciò è necessario sviluppare tecniche e approcci sempre più intelligenti ed automatizzati per processare queste grandi quantità di dati anche detti "*Big Data*". La qualità dei processi e la capacità di gestione di queste informazioni è concretamente diventata la vita e la morte dei nuovi modelli di business.

PARTE I

Time series analysis

Con *time series* si intende un set di dati ordinati in maniera temporale in un certo periodo di tempo.

Il data mining temporale o sequenziale si occupa di problemi dove i dati sono organizzati in questo tipo di sequenze (ad esempio, stock, segnali audio o video, segnali provenienti da sensori etc.). Grandi volumi di sequenze di serie temporali appaiono in quasi tutte le discipline quali astronomia, biologia, meteorologia, medicina, finanza, robotica, ingegneria e altri. Tale diffusione delle serie temporali ha generato un sostanziale interesse nell' indicizzazione, la classificazione, il clustering e la predizione di tali dati. Tra tutte le tecniche applicate ai dati di serie temporali, il clustering è il più utilizzato in quanto non si basa su costosa supervisione umana o annotazione di dati che richiede tempo e forza lavoro. Con il clustering, possiamo identificare e riassumere interessanti pattern e correlazioni nei dati sottostanti.

Negli ultimi decenni, il raggruppamento di sequenze di serie temporali ha ricevuto nuovamente notevole interesse, non solo come un potente metodo esplorativo autonomo, ma anche come fase di pre-elaborazione dei dati o subroutine per altre attività.

Univariate Time series

Definiamo una *univariate time series* (d'ora in poi chiamata UTS) una sequenza di dati definita come sopra in cui solo una variabile varia nel tempo come, ad esempio, i dati raccolti da un sensore che misura la temperatura di una stanza ogni secondo. Pertanto, ogni secondo, avremo solo un valore unidimensionale, che è la temperatura.

Multivariate Time series

Definiamo una *multivariate time series* (d'ora in poi chiamata MTS) una sequenza temporale in cui più di una variabile varia nel tempo. Possiamo intendere una MTS come un insieme di segnali

(UTS), ad esempio, provenienti da più sensori posti in una stanza. Quindi, in ogni istante temporale preso in considerazione avremo N valori quali, ad esempio temperatura, pressione, umidità di una stanza.

Machine learning

Con il termine *machine learning* (in Italiano "apprendimento automatico") ci si riferisce ad un ramo fortemente in evoluzione di algoritmi computazionali ispirati all'intelligenza umana che imparano dall'ambiente circostante.

Spesso confuso col termine "intelligenza artificiale", il machine learning è invece soltanto una sotto branca di quest'ultima.

Sono considerati il cavallo di battaglia nella nuova era dei cosiddetti big data. Le tecniche basate sull'apprendimento automatico vengono applicate con successo in diversi campi che vanno dal riconoscimento di modelli, visione artificiale, ingegneria dei veicoli, finanza, intrattenimento e biologia computazionale alle applicazioni biomediche e mediche.

La fase di machine learning e quindi di allenamento di un tale modello è generalmente posizionata nella fase finale di una pipeline di *data mining* e rappresenta lo step tramite il quale si caratterizzano i dati contenuti all'interno di un dataset e se ne estraggono dei pattern.

Al termine di tale step un esperto di dominio dovrebbe analizzare i risultati e verificarne l'effettiva valenza per il task in questione.

Tipi di problemi e task

Grazie alla grande disponibilità di dati e potenza computazionale disponibile ad oggi l'utilizzo di algoritmi di machine learning è sempre più preminente nell'industria. I campi di applicazione sono innumerevoli:

- Funzionalità di social media: i social notano e registrano attività, chat, i mi piace e i commenti e il tempo che un utente dedica a tipi specifici di post. L'apprendimento automatico impara dall' esperienza e crea suggerimenti di amici e di pagine per il profilo dell'utente.
- Raccomandazione prodotti: utilizzando l'apprendimento automatico e l'intelligenza artificiale, i siti web tengono traccia del comportamento, dei precedenti acquisti, delle ricerche, alla cronologia del carrello di utente e forniscono consigli sui prodotti.
- Riconoscimento d'immagini
- Sentiment analysis
- Predizione guasti e manutenzione

Questa rappresenta solo una minima parte delle applicazioni industriali dei processi di elaborazione dei dati.

Gli approcci di machine learning sono tradizionalmente suddivisi in tre grandi categorie, a seconda della natura del dataset o del "feedback" disponibile per il sistema di apprendimento:

- Apprendimento supervisionato (*supervised learning*): al modello vengono presentati input di esempio e gli output desiderati, anche *detti "ground truth"*, e l'obiettivo è quello di apprendere una regola generale che associ gli input agli output.

- Apprendimento non supervisionato (un*supervised learning*): nessuna etichetta viene assegnata all'algoritmo di apprendimento, lasciandolo da solo per trovare la struttura nel suo input. L'apprendimento non supervisionato può essere un obiettivo in sé (scoprire pattern nascosti nei dati) o un mezzo verso un fine (apprendimento delle funzionalità).
- Apprendimento per rinforzo (*reinforcement learning)*: un agente interagisce con un ambiente dinamico in cui deve raggiungere un determinato obiettivo (come guidare un veicolo o giocare una partita contro un avversario). Mentre naviga nel suo spazio, al programma viene fornito un feedback analogo ai premi, che cerca di massimizzare.

Apprendimento supervisionato

L'apprendimento supervisionato (SL) è l'attività di machine learning che mira all'apprendimento di una funzione in grado di associare un input a un output in base a coppie input-output di esempio. Il modello deduce una funzione dai dati di addestramento etichettati e dovrà essere in grado di etichettare dati futuri correttamente. Ciò richiede che l'algoritmo di apprendimento generalizzi dai dati di addestramento a situazioni non viste prima in un modo "ragionevole". Questa qualità statistica di un algoritmo viene misurata attraverso il cosiddetto "*bias"*. Il bias è una misura dell'errore e permette quindi di quantificare quanto il modello addestrato differisce rispetto ad un modello "perfetto".

Questo bias, se elevato, può essere dovuto a due principali cause:

- *Overfitting*: il modello statistico è troppo complesso si adatta ai dati osservati (il campione) probabilmente dovuto ad un numero eccessivo di parametri rispetto al numero di osservazioni. In questo caso si osservano ottimi risultati sui dati di training e scarse prestazioni su nuovi dati mai visti.
- *Underfitting*: Il modello ha un numero di parametri troppo basso e quindi la funzione di mappatura imparata risulta essere troppo semplice per etichettare correttamente i dati.

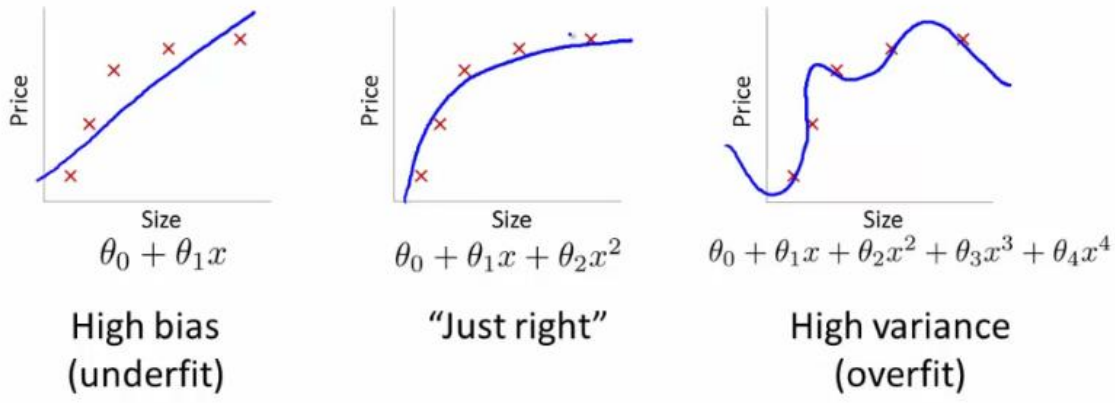


Figura 1 Grafici underfitting, right fitting e overfitting

Pertanto questi tipi di modelli soffrono di diverse criticità che devono essere individuate, analizzate e risolte al fine di ottenere risultati soddisfacenti.

I principali task degli algoritmi supervisionati sono:

- Classificazione: l'obiettivo del task è categorizzare, e quindi assegnare una label, un gruppo di appartenenza ai dati.
- Regressione: l'obiettivo del task è predire un valore numerico associato ad un dato di input.

I più comuni algoritmi supervisionati sono:

- *Decision Tree(classificazione)*
- *Support vector machine (classificazione)*
- *Linear regression (regressione)*
- *Artificial Neural Network(classificazione, regressione e altro)*

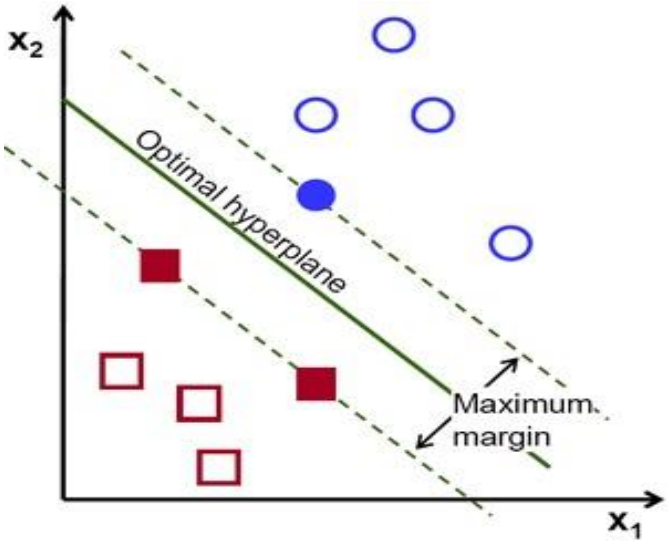


Figura 2 Esempio di piano con support vector machine

Per ognuno di questi esistono varianti che li adattano anche ad altri task o ne aumentano la complessità parametrica.

In questo studio abbiamo utilizzato per una buona parte del lavoro le reti neurali pertanto è doveroso un approfondimento.

Reti neurali nell' apprendimento supervisionato

Le reti neurali artificiali (ANN) sono un sottoinsieme degli algoritmi di machine learning e nello specifico rientrano nella tipologia di algoritmi di deep learning. La loro struttura e il funzionamento si ispirano al cervello umano, emulando i segnali e il modo tramite il quale i neuroni si organizzano per apprendere un nuovo task.

Una rete neurale è composta da più strati neuronali e nella sua implementazione più semplice presenta uno strato di input, uno o più strati intermedi detti *"hidden layers"* e uno strato di output.

Ogni strato è formato da uno o più neuroni, ogni neurone è connesso ad ogni altro neurone dello strato successivo (questa è una semplificazione, in realtà questa architettura di rete è una possibile variante detta "*fully connected neural network"*).

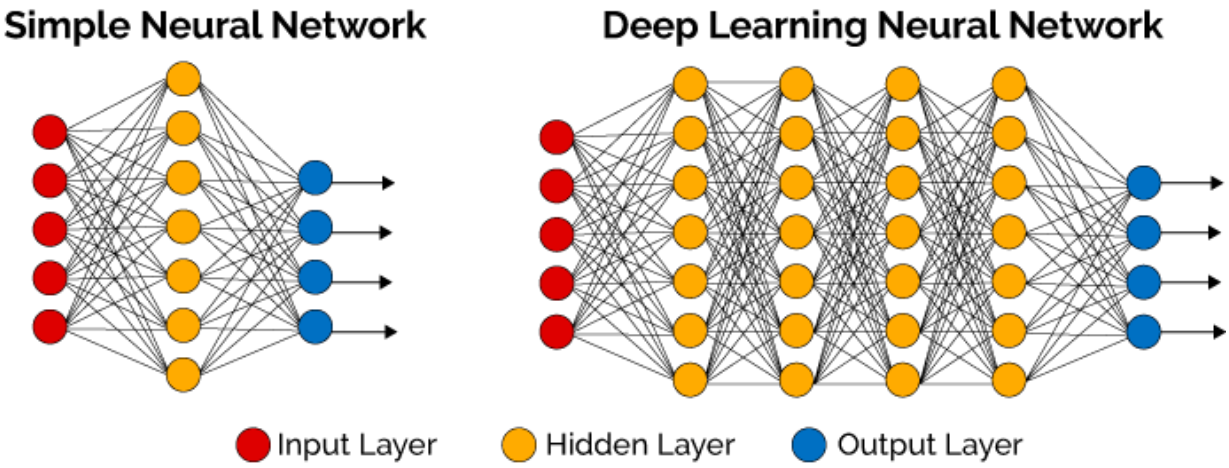


Figura 3 Architetture di Reti neurali artificiali

Il collegamento fra un neurone ed un altro è modellato da un parametro chiamato "peso" (*weight*), che rappresenta la "forza" con la quale il neurone è connesso all'altro. In ambito biologico il peso rappresenta una sinapsi, ovvero quella struttura anatomica che consente la comunicazione fra più neuroni.

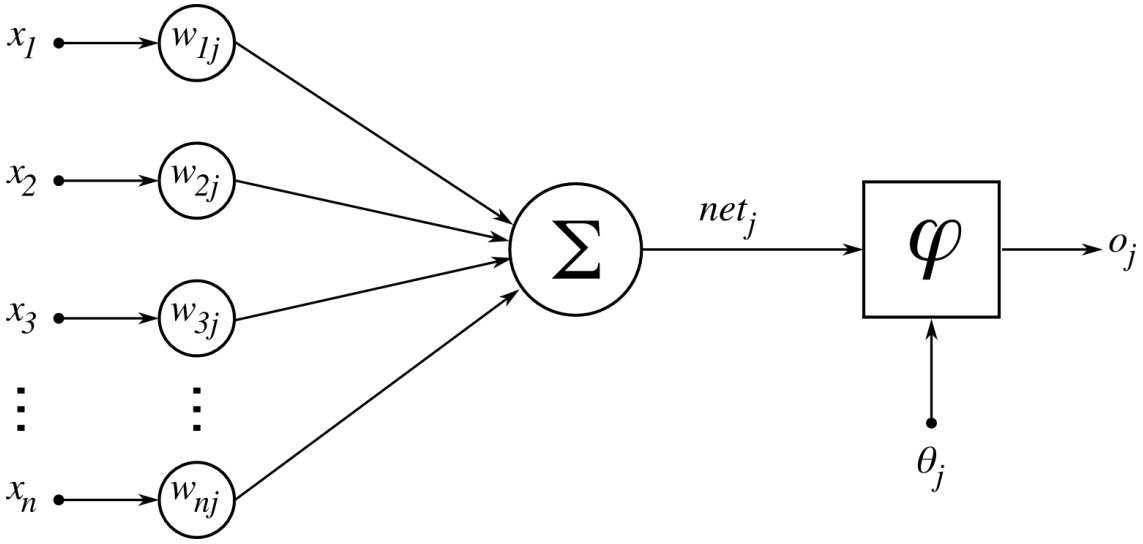


Figura 4 Schematizzazione neurone artificiale

Ogni neurone può essere inteso come un regressore lineare che riceve come input gli output di tutti i neuroni a lui connessi moltiplicati per il relativo peso più un *bias*. La seguente formula è esplicativa di tale comportamento:

$$u = \sum_{i=1}^{n} w_i x_i$$

o in notazione matriciale

$$f(x) = b + W^T X$$

dove W è il vettore dei pesi e X il vettore delle variabili di input

Una volta assegnato il valore del neurone si può applicare una funzione di attivazione che ne determina l'output. Le funzioni di attivazione sono funzioni derivabili che simulano il "trigger" che avviene in un neurone umano quando il neurone raggiunge un certo stato di eccitazione.

Semplificando, se il valore del neurone supera una certa soglia, il neurone si attiva e l'output viene inviato ai neuroni successivi.

In realtà esistono molte funzioni di attivazione con comportamenti anche molto diversi fra loro, una delle più utilizzate è la *"linear rectifier"* (anche detta *RELU*) che ha la seguente equazione:

$$f(x) \; = \; x+ = \; max(0, x)$$

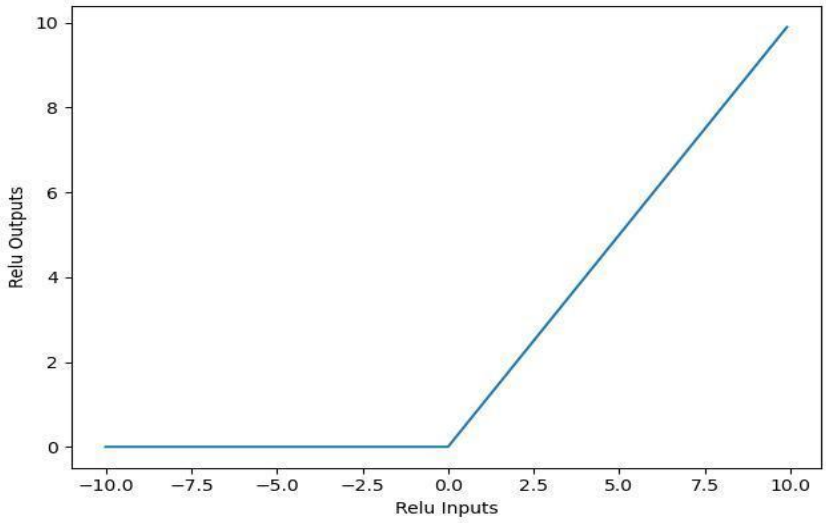


Figura 5 Esempio di linear rectifier

Altra funzione da menzionare è la funzione "*softmax*", utilizzata spesso come funzione di attivazione dell'ultimo layer, quello di output, che permette di modellare un distribuzione di probabilità di un certo input nell' appartenere ad ognuna delle classi prese in considerazione.

$$\sigma(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} \text{ per } j = 1, \dots, K$$

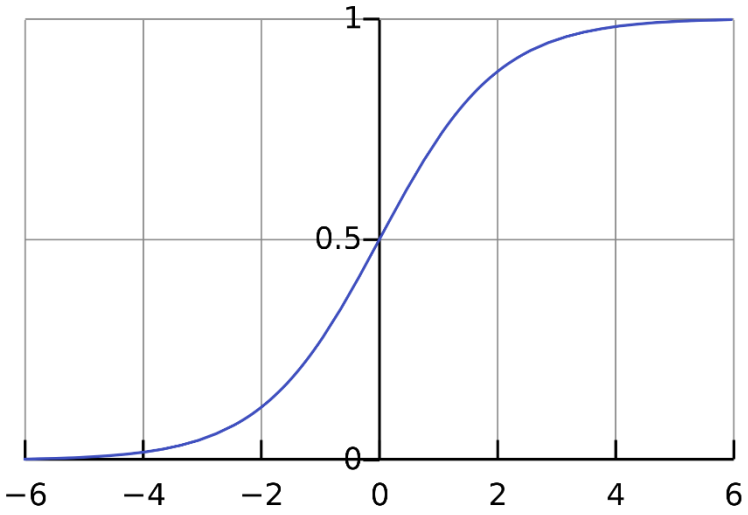


Figura 6 Esempio softmax

Funzione di Loss e training

Dopo aver introdotto la struttura della rete neurale passiamo a spiegare come avviene il processo di apprendimento.

Come già detto, nel processo di apprendimento supervisionato l'obiettivo è quello di creare una funzione di mapping che sia in grado di predire il corretto output per sample mai visti prima. L'apprendimento di questa funzione avviene secondo un processo di try and error.

La fase di training comporta la regolazione dei pesi (e delle soglie opzionali) della rete per migliorare l'accuratezza (o anche altre metriche) del risultato. Questo viene fatto riducendo al minimo gli errori osservati.

Nella pratica questo viene fatto definendo una funzione di costo (o di loss) che viene valutata periodicamente durante l'apprendimento.

$$Crossentropy = H(p,q) = -\sum_i p_i \log q_i = -y \log \hat{y} - (1-y) \log(1-\hat{y})$$

$$p \in \{y, 1-y\} \; e \; q \in \{\hat{y}, 1-\hat{y}\}$$

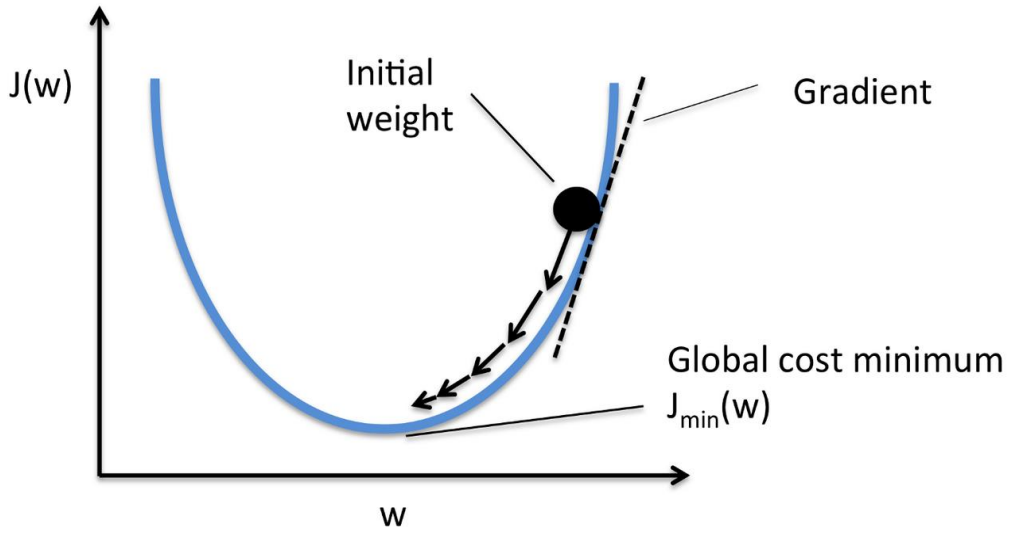


Figura 7 Visualizzazione del processo di minimizzazione della funzione di costo. I pesi variano in direzione del gradiente con l'obiettivo di portare il punto verso un minimo (preferibilmente globale) della funzione.

In questo caso abbiamo mostrato l'equazione della *categorical cross-entropy* , una funzione di perdità molto diffusa nell' ambito della classificazione.

Ad ogni step di training un batch di sample viene prelevato dal dataset e passato alla rete neurale.

L'output generato dalla rete neurale viene passato alla funzione di costo insieme all'output desiderato e vengono così confrontati.

L'obiettivo è quello di minimizzare questo output e quindi la funzione di loss aggiustando ad ogni iterazione i pesi che compongono la rete fino a raggiungere il valore desiderato dopo un certo tempo (definito in epoche).

L' aggiornamento dei pesi e la conseguente minimizzazione della funzione di loss avviene secondo un metodo di ottimizzazione per funzioni differenziabili iterativo chiamato *stochastic gradient descent (SGD)* definito come segue:

$$Loss \;=\; Q(w) = \frac{1}{n}\sum_{i=1}^{n} Q_i(w)$$

$$w \coloneqq w - \eta \nabla Q(w) = w - \eta \frac{1}{n}\sum_{i=1}^{n} \nabla Q_i(w)$$

Dove *w* indica il peso, ∇ è l'operatore di gradiente e η è il learning rate, un parametro che indica il fattore di sconto o la grandezza dello step ad ogni iterazione mentre ci si muove verso il minimo della funzione.

Convolutional neural network

Una rete neurale convoluzionale (CNN) è un tipo di rete neurale artificiale comunemente utilizzata per analizzare immagini e video, sistemi di raccomandazione, elaborazione del linguaggio naturale, serie temporali.

Le reti convoluzionali si ispirano ai processi biologici in quanto il modello di connettività tra i neuroni assomiglia all'organizzazione della corteccia visiva animale. I singoli neuroni corticali rispondono agli stimoli solo in una regione ristretta del campo visivo nota come campo recettivo

In una rete neurale convoluzionale, gli hidden layers includono strati che eseguono operazioni di convoluzione. Tipicamente questo include un livello che esegue un prodotto scalare del kernel di convoluzione con la matrice di input del livello. Questo prodotto è solitamente il prodotto interno e la sua funzione di attivazione è comunemente la *RELU*. Quando il kernel di convoluzione scorre lungo la matrice di input, l'operazione di convoluzione genera una mappa detta *features map* che a sua volta contribuisce all'input del livello successivo. Questo processo è simile alla risposta di un neurone nella corteccia visiva quando sottoposto ad uno specifico stimolo.

Le reti convoluzionali possono includere livelli di pooling locali e/o globali insieme ai tradizionali livelli convoluzionali. I livelli di pooling riducono le dimensioni dei dati combinando gli output dei cluster di neuroni a uno strato in un singolo neurone nel livello successivo eseguendo operazioni di estrazione del massimo, minimo o media di un cluster di valori.

Ciascun neurone in una rete neurale calcola un valore di output applicando una funzione specifica ai valori di input ricevuti dal campo recettivo nel livello precedente.

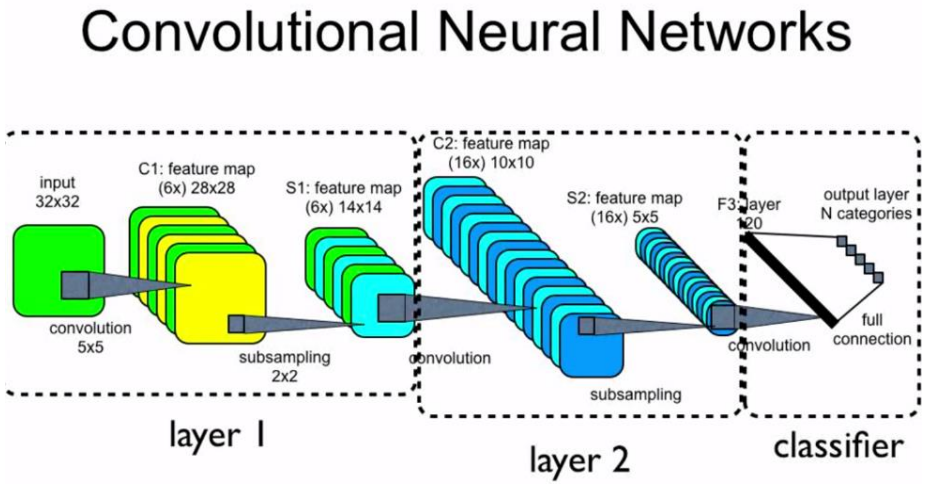


Figura 8 Esempio di layers di una rete neurale convoluzionale

Apprendimento non supervisionato

L'apprendimento non supervisionato utilizza algoritmi di apprendimento automatico per analizzare e raggruppare set di dati non etichettati. Questi algoritmi scoprono modelli nascosti o cluster di dati senza la necessità dell'intervento umano. La sua capacità di scoprire somiglianze e differenze nelle informazioni lo rende la soluzione ideale in ambito specialmente industriale per l'analisi esplorativa dei dati, la segmentazione dei clienti e il riconoscimento delle immagini, la previsione di guasti dei macchinari.

Algoritmi di Clustering

Il clustering è una tecnica di data mining che raggruppa dati non etichettati in base alle loro somiglianze o differenze. Gli algoritmi di clustering vengono utilizzati per elaborare oggetti dati grezzi e non classificati in gruppi rappresentati da strutture o modelli nelle informazioni. Gli

algoritmi di clustering possono essere classificati in alcuni tipi, specificamente esclusivi, sovrapposti, gerarchici e probabilistici.

Esistono diversi algoritmi di clustering e la loro efficacia dipende molto dai tipi di dati che utilizziamo e dalla forma dei cluster. Un altro parametro cruciale nella scelta e nel tuning di un algoritmo di clustering è la metrica utilizzata per calcolare le distanze reciproche (intesa anche come similarità/dissimilarità) tra i punti e/ o tra i cluster all'interno di uno spazio multidimensionale.

Inoltre gli algoritmi di clustering possono essere suddivisi in ulteriori due categorie:

Il clustering esclusivo: stabilisce che un punto dati può esistere solo in un cluster.

Il clustering sovrapposto: consente ai punti di appartenere a più cluster con gradi di appartenenza separati.E' anche detto *"soft" o fuzzy*, un esempio è il fuzzy k-means.

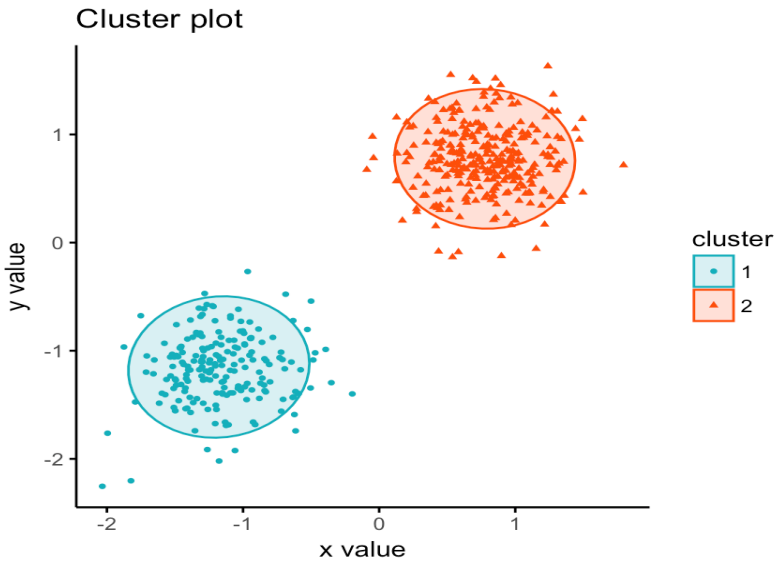


Figura 9 Esempio di punti clusterizzati su un piano bidimensionale

KMeans e K-Medoids

L'algoritmo di K-Means è un esempio comune di un metodo di clustering esclusivo in cui i punti dati vengono assegnati a K cluster, dove K rappresenta il numero di cluster. L'assegnazione si basa sulla distanza dal centroide di ciascun gruppo. I punti più vicini a un dato centroide verranno raggruppati nella stessa categoria.

Un valore K più grande sarà indicativo di raggruppamenti più piccoli con più granularità mentre un valore K più piccolo avrà raggruppamenti più grandi e meno granularità.

Il valore di K deve essere inserito dal programmatore, pertanto è un parametro da ottimizzare nel caso non si conosca il numero effettivo di cluster da generare.

K-means ha problemi quando i cluster differiscono molto in termini di grandezza, densità o presentano forme non globulari, inoltre non ha alcuna gestione degli *outliers*.

Un algoritmo con un approccio simile è il K-Medoids nel quale però il baricentro di un cluster non è un punto artificiale, ma un punto dell'insieme di dati.

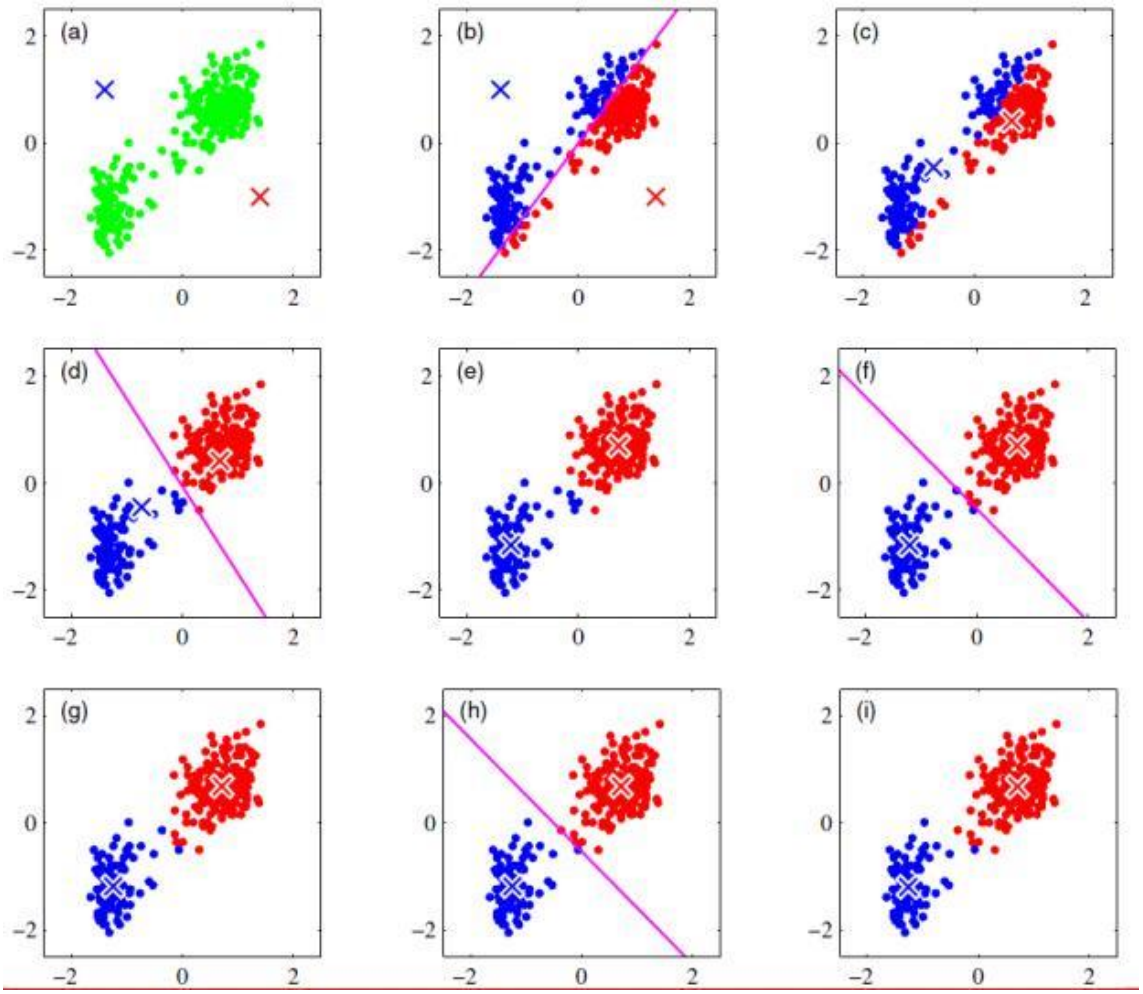


Figura 10 Successione degli step dell'algoritmo di K-means. Notiamo come i cluster formati abbiamo una forma regolare.

Agglomerative clustering

Il clustering gerarchico è un algoritmo di clustering non supervisionato che può essere di due tipo: agglomerativo o divisivo. Il clustering agglomerativo è considerato un approccio "dal basso verso l'alto".I punti del dataset, inizialmente isolati, vengono raggruppati insieme in modo iterativo sulla base della somiglianza fino a quando non viene raggiunto un cluster. Esistono quattro diversi metodi usati per definire la similarità tra i cluster e che sono comunemente usati:

1. Ward linkage: in questo metodo la distanza tra due cluster è definita dall'aumento della somma dei quadrati dopo che i cluster sono stati uniti.
2. Average linkage: questo metodo definisce dalla distanza media tra due punti in ciascun cluster
3. Complete (o massimo) linkage: questo metodo è definito dalla distanza massima tra due punti in ciascun cluster
4. Single (o minimo) linkage: questo metodo è definito dalla distanza minima tra due punti in ciascun cluster.

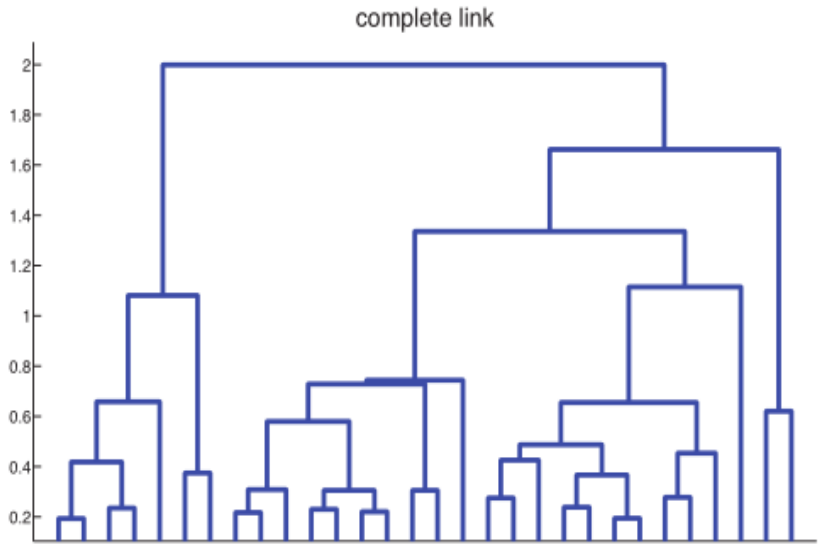


Figura 11 Rappresentazione a dentogramma del risultato di un algoritmo di clustering agglomerativo.

La distanza euclidea è la metrica più comune utilizzata per calcolare queste distanze; tuttavia, nella letteratura sul clustering sono citate anche altre metriche, come la distanza di Manhattan.

DBSCAN

E'un algoritmo di clustering non parametrico basato sulla densità: dato un insieme di punti in uno spazio, raggruppa i punti che sono molto ravvicinati (punti con alta densità di vicini) mentre contrassegna come valori anomali(outliers) i punti che si trovano da soli in regioni a bassa densità (i cui vicini più vicini sono troppo lontani). DBSCAN è uno degli algoritmi di clustering più comuni e anche il più citato nella letteratura scientifica.

DBSCAN richiede due parametri:

- *eps*: La distanza massima tra due campioni affinché uno sia considerato come nelle vicinanze dell'altro.
- *minpoints*: numero minimo di punti necessari per formare una regione densa.

Un punto viene detto "core" se ha più di un numero specificato di punti (*MinPts*) all'interno del raggio eps. Questi sono punti che si trovano all'interno di un cluster.

Un punto di confine ha meno di MinPts all'interno di Eps, ma è un vicino di un punto core.

Un punto di outliers è qualsiasi punto che non sia un punto centrale o un punto di confine.

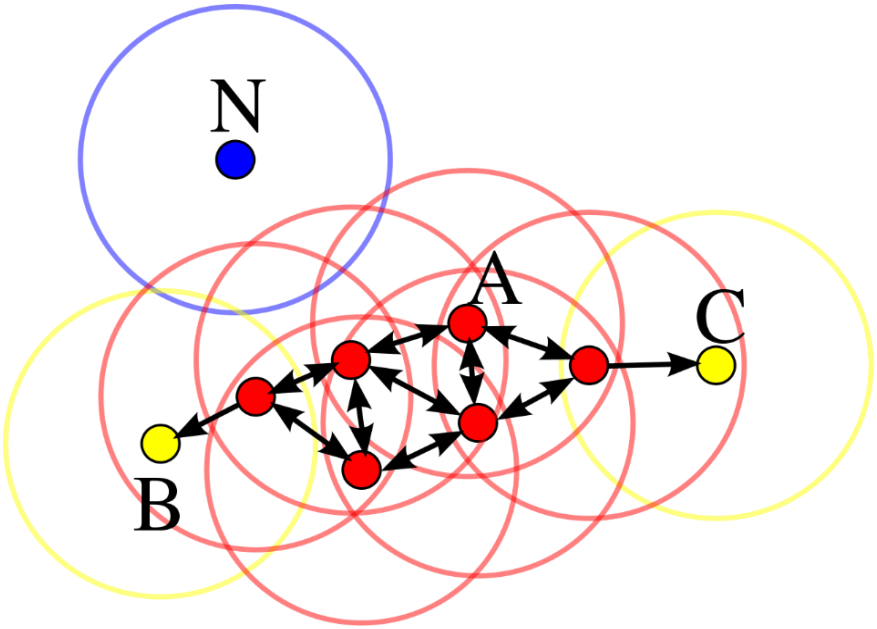


Figura 12 Rappresentazione di uno step del DBSCAN, il raggio della circonferenza corrisponde al parametro eps. I punti rossi sono i punti appartenenti al cluster A. I punti gialli sono i punti di confine mentre il punto blu è un outlier.

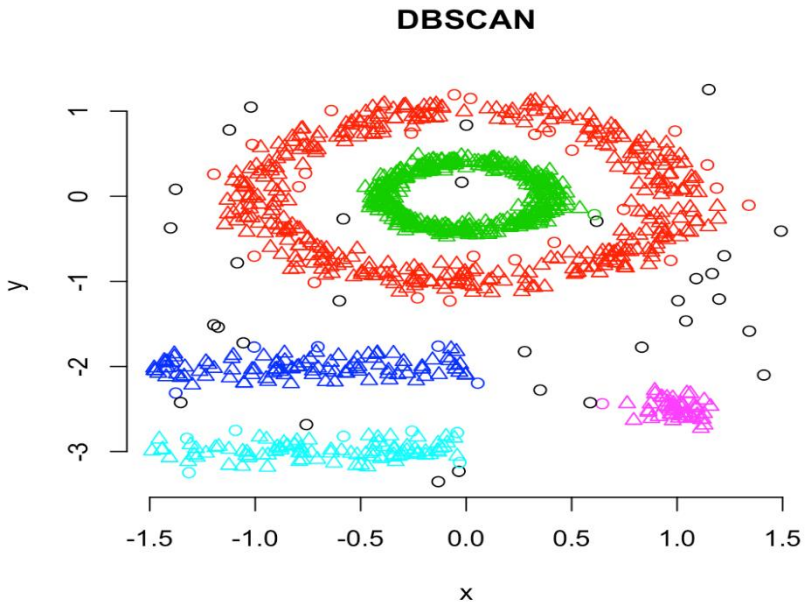


Figura 13 Risultato del clustering con DBSCAN. Notiamo come le forme siano molto più arbitrarie a differenza del KMeans.

Validare i risultati di un algoritmo di cluster

Per la classificazione supervisionata abbiamo una varietà di misure per valutare quanto sia buono il nostro modello quali accuratezza, precisione, richiamo.

Per l'analisi dei cluster valutarne la bontà è invece meno banale.

Generalmente gli approcci più diffusi si dividono in due categorie di valutazioni che sia basano su approcci diversi in base al tipo di dati che abbiamo:

- *Internal evaluation*: si basa su indici interni usati per misurare la bontà della struttura di un clustering indipendentemente dalle informazioni esterne. Esempi tipici sono: la coesione dei cluster, la separazione dei cluster, *Silhouette* index.
  A seguire alcune equazioni per il calcolo della silhouette:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j}^{C} d(i,j)$$

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

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

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

- *External evaluation*: si basa su indici che tengono conto anche di informazioni esterne quali, ad esempio delle label fornite. Alcuni esempi *sono RandIndex(RI), Adjusted RandIndex(ARI).*

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

$$ARI = \frac{\sum_{ij} \left(\frac{n_{ij}}{2}\right) - \left[\sum_i \left(\frac{a_i}{2}\right) \sum_j \left(\frac{b_j}{2}\right)\right] / \left(\frac{n}{2}\right)}{\frac{1}{2}\left[\sum_i \left(\frac{a_i}{2}\right) + \sum_j \left(\frac{b_j}{2}\right)\right] - \left[\sum_i \left(\frac{a_i}{2}\right) \sum_j \left(\frac{b_j}{2}\right)\right] / \left(\frac{n}{2}\right)}$$

Reti neurali convoluzionali nell'apprendimento non supervisionato

Gli Auto Encoder

Un autoencoder è un algoritmo di apprendimento non supervisionato basato su reti neurali che applica la retro propagazione. L'obiettivo di un autoencoder è ricostruire un input creando un spazio di embedding che sia in grado di rappresentare in maniera efficiente la caratteristiche di un dato e conseguentemente fare features reduction. L'architettura è divisa in tre parti principali:

- *L'encoder*: ha il compito di mappare l'input nello spazio di embedding. Può essere implementato come semplici multiplayer perceptron in cascata o tramite architetture più complesse ad esempio utilizzando strati convoluzionali
- *Bottleneck* o *latent space*: è la parte dell'autoencoder che codifica lo spazio di embedding. In questo stadio l'input è mappato come un punto dello spazio di embedding e quindi si trova nella sua forma compressa
- *Decoder*: ha il compito di ricostruire l'input a partire da un punto dello spazio di embedding e restituirlo come output

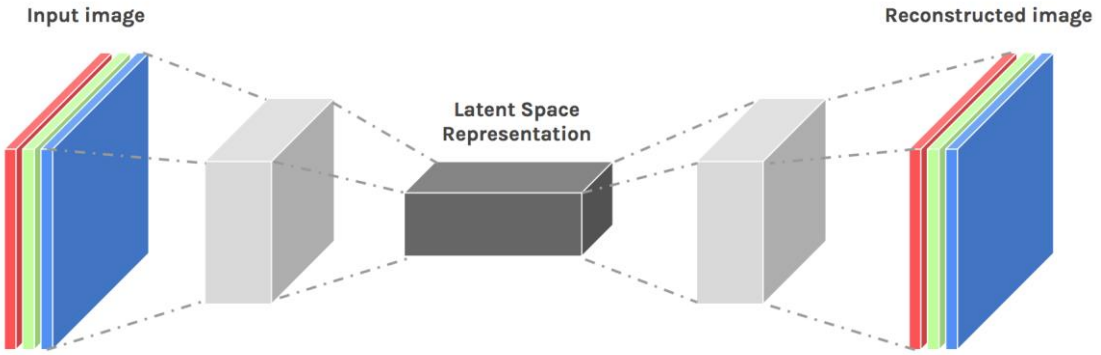


Figura 14 Esempio architettura autoencoder

Durante la fase di training l'autoencoder viene addestrato a ricostruire un input (un'immagine, un segnale etc.) con l'obiettivo di minimizzare l'errore di ricostruzione espresso tramite funzioni di loss basate su distanze.

Matematicamente possiamo esprimere un autoencoder nel seguente modo:

sia $x \epsilon R^d$ il vettore di input di dimensione d, il decoder cerca di apprendere una funzione mapping:

$$h(\cdot) : h(x) \rightarrow y \epsilon Y^m \; con \; m \leq d \; e \; Y \; lo \; spazio \; di \; embedding$$

infine un decoder invece crea la funzione di mapping:

$$f(\cdot): \; f(y) \rightarrow \; x' \epsilon R^d$$

Definita una funzione di perdita come ad esempio la funzione di squared error:

$$L(x, x') = |x - x'|^2$$

l'obiettivo è minimizzare la distanza fra $x \; e \; x'$:

$$min\big(L(x, x')\big) = min(|x - x'|^2) = min(|x - f(h(x)|^2)$$

Funzioni di loss basate su distanze

Le funzioni di loss basate su distanze sono utilissime ed essenziali nei problemi di regressione, in cui è appunto necessario misurare un errore tra un risultato predetto e quello aspettato. In seguito un elenco delle funzioni che utilizzeremo nel nostro lavoro:

Mean absolute error

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |Y_i - \widehat{Y}_i|$$

Mean squared error

$$MSE = \frac{1}{n}\sum_{i=1}^{n} \big(Y_i - \widehat{Y}_i\big)^2$$

Huber loss

$$L_\delta(x) = f(x) = \begin{cases} \frac{1}{2}x^2, |x| \leq \delta \\ \delta(|x| - \frac{1}{2}\delta), \ altrimenti \end{cases}$$

Log-Cosh loss

$$logc = \sum_{i=1}^{n} log(cosh(Y_i - \widehat{Y}_i))$$

Algoritmi di Natural Language Processing

Natural Language Processing (NLP) è una branca della linguistica, dell'informatica e dell'intelligenza artificiale che si occupa di sviluppare metodologie per elaborare e analizzare grandi quantità di dati del linguaggio naturale. L'obiettivo è un trovare modello in grado di "capire" i contenuti dei documenti. Le applicazioni più comuni spesso riguardano il riconoscimento vocale e la generazione di testi, traduzione testi, sentiment analysis etc.

Ad oggi le tecniche più comuni di NLP combinano algoritmi informatici con modelli di machine learning e deep learning per estrarre, classificare ed etichettare automaticamente elementi di testo e dati vocali e quindi assegnare una probabilità statistica e creare uno spazio di embedding a ciascun possibile significato di tali elementi.

In linea generale una pipeline di NLP consiste nel effettuare una prima fase di preprocessing dei vocaboli durante la quale avvengono dei processi di trasformazione quali:

- *Tokenization*: la tokenization (o tokenizzazione) è un modo per separare una parte di testo in unità più piccole chiamate token.I token possono essere parole, caratteri o parole secondarie. Spesso questa fase è seguita dalla rimozione delle stop word.
- *Stemming*: è il processo di trasformazione delle parole nelle loro rispettive radici.
- Creazione di un vocabolario
- *Encoding*: poiché gli algoritmi di machine learning possono lavorare solo su numeri è necessario codificare le parole del vocabolario in un vettore di *embedding*. Le parole vengono inizialmente codificate come one-hot encoding per essere passate poi al modello per l'embedding.

Word2Vec

Word2vec è un approccio basato su una rete neurale a soli due strati che elabora le parole trasformandole in vettori. L'input è un corpus di parole di grandi dimensioni e l'output è uno spazio vettoriale, di solito centinaia di dimensioni, con ogni parola univoca nel corpus rappresentata con quello spazio vettoriale generato. È usato per ricostruire i contesti linguistici delle parole in numeri. I vettori di parole sono posizionati nello spazio vettoriale in modo tale che le parole che condividono contesti comuni si trovino in stretta vicinanza l'una dall'altra in quello spazio multidimensionale, in parole povere le parole che hanno quasi lo stesso significato saranno messe insieme. Questo modello cattura le somiglianze sintattiche e semantiche tra le parole.

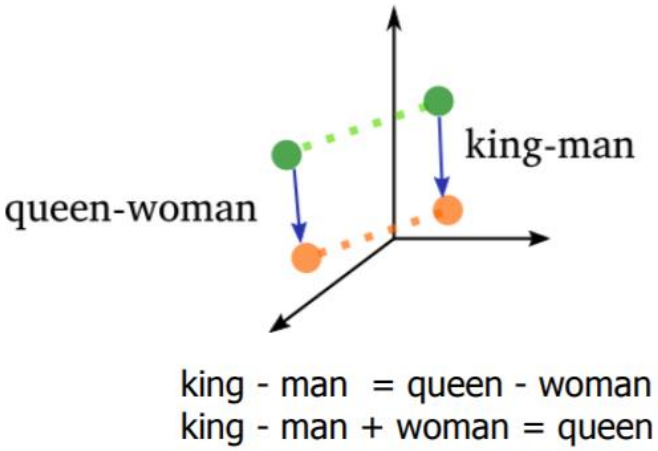


Figura 15 Esempio embedding in word2vec

Ciò può avvenire attraverso due approcci:

- *CBOW(Continuous bag of words)* : ad ogni step l'algoritmo prova a prevedere la parola target usando le parole di contesto.
- *Skip-Gram*: ad ogni step l'algoritmo prova a prevedere le parole di contesto usando una parola di destinazione.

Con parola target si intende la parola che viene predetta.

Con parola di contesto: ogni altra parola nella frase, a parte la parola di destinazione.

Dimensione dell'embedding: la grandezza del vettore in cui vogliamo che le nostre parole siano codificate.

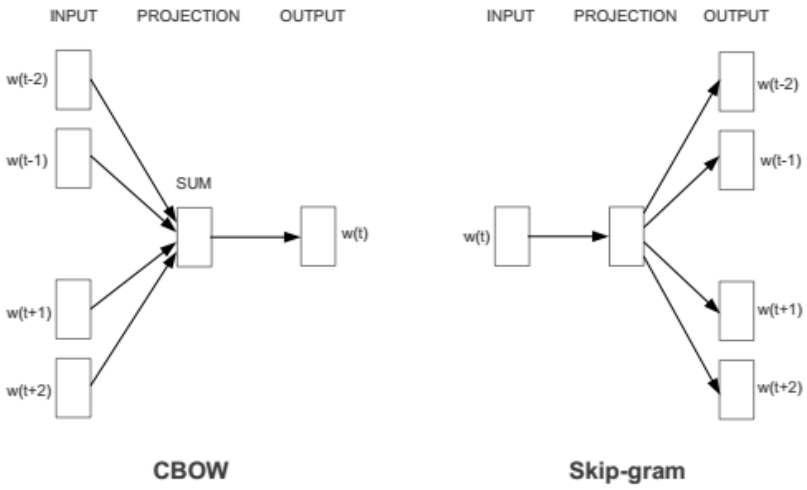


Figura 16 Schematizzazione CBOW e Skip-Gram

Doc2Vec

Dopo aver introdotto cos'è word2vec, sarà più agevole spiegare come funziona doc2vec. L'obiettivo di doc2vec è creare una rappresentazione numerica di un documento, indipendentemente dalla sua lunghezza e non più delle singole parole. Il concetto adottato da doc2vec è semplice, ma intelligente [1] :

si applica il modello word2vec e si aggiunge un altro vettore in input (ID paragrafo) come mostrato nelle figure.

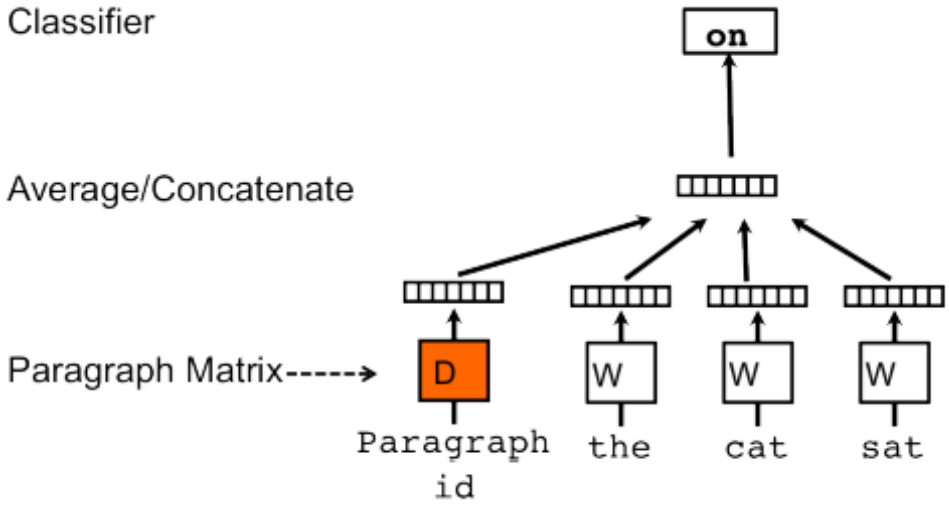


Figura 17 Modello PV-DM estensione del modello CBOW

Ma invece di usare solo parole per prevedere la parola successiva, si utilizza anche un altro vettore di caratteristiche, che è l'id del documento. Quindi, durante l'addestramento dei vettori di parole W, viene addestrato anche il vettore del documento D e, alla fine dell'addestramento, si ottiene una rappresentazione numerica del documento. Il modello sopra è chiamato versione a memoria distribuita del vettore di paragrafo (PV-DM). Agisce come un ricordo che ricorda ciò che manca dal contesto attuale. Mentre i vettori di parole rappresentano il concetto di una parola, il vettore di documenti intende rappresentare il concetto di documento. Come in word2vec, può essere utilizzato un altro algoritmo, simile a skip-gram, chiamato *"Distributed Bag of Words of Paragrafo Vector"* (PV-DBOW).

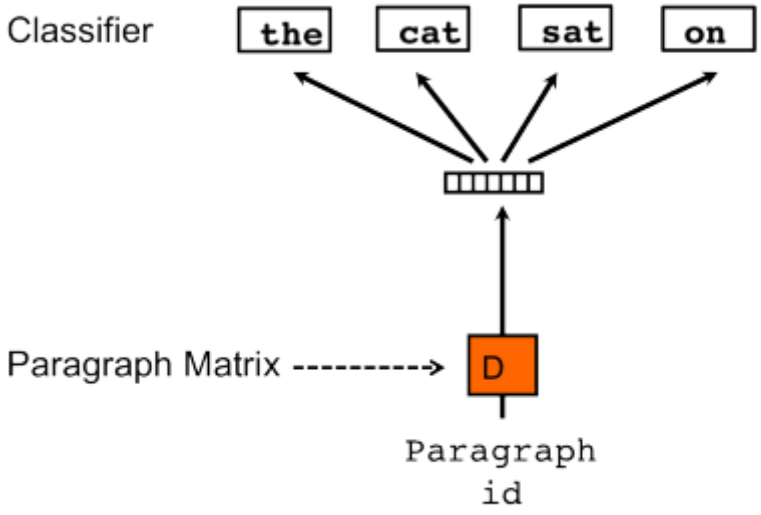


Figura 18 Model PV-DBOW

Questo algoritmo è effettivamente più veloce (al contrario di word2vec) e consuma meno memoria, poiché non è necessario salvare i vettori di parole. Nell'articolo, gli autori affermano di consigliare l'uso di una combinazione di entrambi gli algoritmi, sebbene il modello PV-DM sia superiore. I modelli doc2vec possono essere utilizzati nel seguente modo: per la formazione è necessaria una serie di documenti. Per ogni parola viene generato un vettore di parole W e per ciascun documento viene generato un vettore di documenti D. Il modello allena anche i pesi per uno strato nascosto softmax. Nella fase di inferenza, può essere presentato un nuovo documento e tutti i pesi sono fissati per calcolare il vettore del documento.

Time series clustering

Come illustrato in [2] e [3] il clustering di serie temporali ha avuto molta risonanza ed è stato oggetto di molti studi. In [3] vengono illustrati i progressi del time series clustering negli ultimi 20 anni nello specifico per quanto riguarda le univariate time series. Gaussian mixed models [13], ARIMA [14], e Hidden Markov models [15,16] rappresentano lo stato dell'arte tra gli approcci per il clustering di time series multivariate.

In [4] propongono un metodo iterativo chiamato K-shape che sfrutta la correlazione incrociata per misurare la distanza tra UTS. Questo approccio sembra superare in termini di minor complessità e accuratezza altri metodi allo stato dell'arte.

Un metrica molto comune nel clustering UTS è presentata in [17] è il "Dynamic time wrapping" estesa in diverse varianti anche per il clustering MTS come studiato in [5].

[6] interessante affronta il clustering di UTS utilizzando reti relazionali e algoritmi di community detection.

Nel complesso, quindi, gli approcci per l'UTS clustering sembrano solidi e ben affermati mentre risulta un problema ancora molto aperto quello per un clustering accurato ed efficiente per le MTS.

Clustering per time series multivariate

Comparato al clustering di serie temporali univariate, il clustering di serie temporali multivariate è diventato un argomento di ricerca impegnativo sul data mining delle serie temporali. A causa dell'elevata dimensionalità delle serie temporali multivariate e della maggior parte del lavoro precedente incentrato sul clustering di serie temporali univariate ad oggi non sembrano esserci metodi affermati per la loro gestione.

In [7] si propone un nuovo metodo basato sull'analisi delle componenti principali comuni per ottenere il clustering di serie temporali multivariate in modo più rapido e accurato chiamato Mc2PCA.

Sempre Li et all in [8] mappa un set di dati MTS su una rete multi-relazione (MRN) e in secondo luogo, sfrutta una fattorizzazione a matrice non negativa progettata per identificare cluster di serie temporali. In [9] lo studio sviluppa un nuovo metodo basato su reti complesse per il clustering di serie temporali multivariate (BCNC). BCNC, ispirato a [6] si basa in modo innovativo su una rete di relazioni e si sfrutta l'uso della tecnologia di community detection per ottenere un clustering completo di serie temporali multivariate.

In [10] invece si utilizzano le SOM (self organizing map) , delle reti neurali che sfruttano un approccio non supervisionato per rappresentare dati ad alta dimensionalità in struttura a più bassa dimensionalità, tipicamente in due dimensioni, preservando la struttura topologica dei dati.

Gli approcci mostrati finora quindi seguono un processo per il quale le MTS subiscono delle trasformazioni matematiche (come PCA, SOM etc) in modo da venir proiettate in nuovi spazi vettoriali nei quali è più efficiente o agile l'applicazione di algoritmi di clustering standard.

L'approccio adottato nel nostro studio segue questa linea, ma invece di utilizzare trasformazioni generali seguiremo un approccio basato su reti neurali per trovare un sottospazio vettoriale specifico e quanto più rappresentativo per i dati che analizzeremo.

PARTE II

Metodo proposto

Obiettivi della tesi

La motivazione allo studio del lavoro può essere riassunta come segue:

(1) A causa dell'elevata dimensionalità delle MTS, proponiamo dei metodi di embedding in grado di integrarsi validamente nel processo di clustering ottenendo buoni risultati di clustering nel nuovo spazio vettoriale.
(2) Quando l'MTS è molto lunga, formata da un elevato numero di features e il volume del dataset MTS è molto grande, gli approcci più tradizionali, basati su DTW, diventano estremamente inefficienti. Per questo motivo tenteremo di trovare dell'alternative più efficienti a parità di accuratezza dei risultati.
(3) È ben noto che K-Means, DBSCAN etc. siano algoritmi di clustering semplici ed efficaci, le cui complessità temporali non sono elevate. Ci si aspetta, quindi, che il design dei

metodi proposti permettano di trovare dei vettori di embedding che possano essere utilizzati con buoni risultati da questi algoritmi.

(4) L'approccio sviluppato, a differenza di quelli degli altri paper menzionati in precedenza, non deve utilizzare algoritmi di features reduction generali(es PCA), ma creare sottospazi di embedding che si adattino alle caratteristiche di ogni dataset tramite l'utilizzo di autoencoder e algoritmi di NLP adattati per poter lavorare con i segnali UTS che compongono le MTS.

Metodologie applicate

Nel nostro studio sono state implementati due diversi approcci entrambi col fine di creare vettori di embedding.

Come primo approccio abbiamo implementato un autoencoder convoluzionale che prende in input una MTS.

In dettaglio, l'encoder riceve *m* input di dimensione *n* con *n* uguale alla lunghezza della MTS e *m* è la dimensionalità della MTS, quindi il numero di segnali UTS che la compongono.

Ogni input attraversa una serie di strati convoluzionali unidimensionali a diversa grandezza di kernel e filtri, in questo modo ogni kernel "scorre" sulla UTS facendo windowing ed estrapolando le caratteristiche più salienti per ogni finestra campionata. Ad ogni layer l'input riduce la propria dimensione fino a raggiungere la grandezza del feature vector desiderato. Successivamente viene ricostruito dal decoder e confrontato con l'input.

Il secondo approccio ispira ai lavori [1], [2] e che applicano gli stessi concetti di embedding tramite approcci NLP alle serie nucleotidiche del DNA e a time series univariate.

Questo secondo approccio tratta ogni UTS componente una MTS come fosse un documento ed ogni valore della UTS come una parola. Viene quindi applicato l'algoritmo di doc2vec in modo da estrarre un vettore di embedding da ogni UTS (documenti). Questi vettori vengono poi concatenati per formare un unico vettore di embedding.

In entrambi gli approcci si crea successivamente un nuovo dataset di features vector estratti.

I dataset vengono poi utilizzati per fare il training di classici algoritmi di clustering quali K-Means, K-Medoids, agglomerative clustering e DBSCAN.

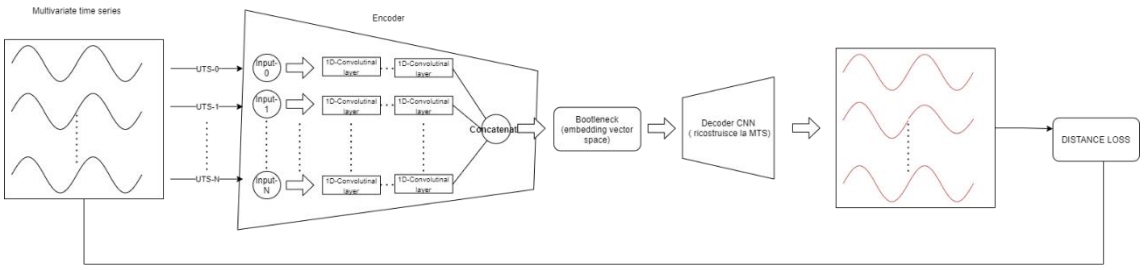


Figura 19 Embedding con autoencoder

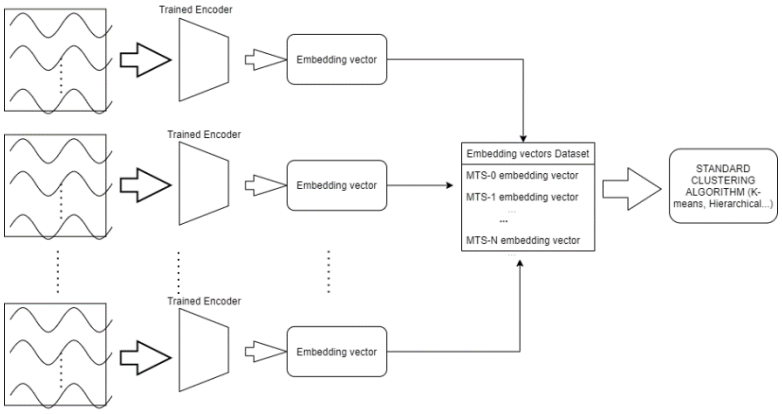


Figura 20 Clustering dopo Embedding con autoencoder

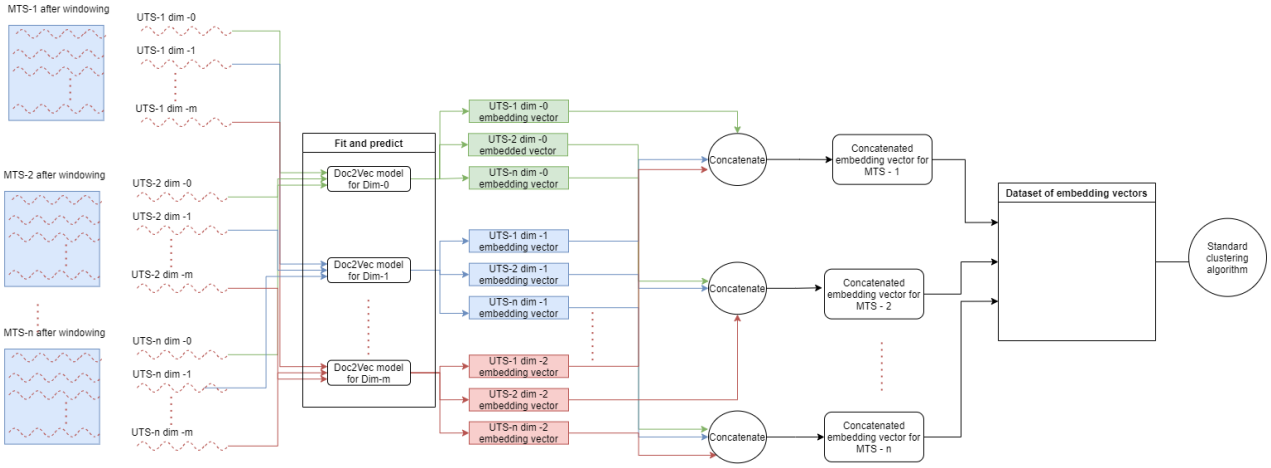


Figura 21 Fase di embedding e clustering per sig2vec

Piattaforma e librerie usate

Gli esperimenti sono stati effettuati sul cluster del centro di ricerca SmartData del Politecnico di Torino. Il cluster assegna agli utenti di 8 CPU e 40 GB di RAM.

E' stato utilizzato il linguaggio di programmazione Python versione 3 nell'ambiente di sviluppo di JupyterLab.

Sono state utilizzate le principali librerie python per il calcolo scientifico quali:

- Numpy: libreria per la gestione efficiente di array e matrici
- Pandas: per la gestione di series e dataframe in maniera efficiente
- Keras/Tensorflow 2: per l'implementazione delle reti neurali e funzioni di loss
- Scikit-learn: librerie per l'implementazione dei principali algoritmi supervised e unsupervised, per il calcolo di matrici di distanze e le principali metriche utilizzate per la validazione dei risultati
- Seaborn e Matplotlib: librerie per la visualizzazione tramite grafici dei dati e dei risultati
- Gensim: libreria per l'implementazione degli algoritmi di NLP quali doc2vec con CBOW e Skip-gram

Dataset utilizzati

I dataset sono stati selezioni da varie fonti. Le principali sono:

- Repository dei dataset UCI Center for Machine Learning and Intelligent Systems dell'università della California:
  https://archive.ics.uci.edu/ml/datasets/
- Repository http://www.time seriesclassification.com/ contenente solo dataset di serie temporali
- Google Dataset

Abbiamo selezionato solo dataset multivariati con un numero di samples tale da poter fare un windowing in grado di estrarre almeno >=20k time series.

In dettaglio abbiamo utilizzato il dataset:

Selfback: https://archive.ics.uci.edu/ml/datasets/selfBACK

Il set di dati SELFBACK [18] è diviso in 9 classi di attività; 6 attività di movimento e 3 attività sedentarie, eseguite da 33 partecipanti. I dati vengono registrati con due accelerometri triassiali con campionamento a 100Hz, montati sul polso laterale dominante e la coscia del partecipante.

Il dataset consta di 4.616.316 samples in totale e non sono presenti valori nulli.

Analisi del dataset

Come mostra il comando *"describe()"* della libreria pandas, il dataset è composto da attributi con valori compresi tra -8 e +8. Da questo intuiamo che potrebbero non servire operazioni di normalizzazione dei dati in quanto tutti appartenenti all'incirca alla stessa scala.

	wx	wy	wz	tx	ty	tz
count	4.616316e+06	4.616316e+06	4.616316e+06	4.616316e+06	4.616316e+06	4.616316e+06
mean	-4.973957e-01	-5.462681e-01	1.744710e-01	-7.950896e-01	-1.489110e-01	-2.449117e-01
std	6.541399e-01	7.150293e-01	4.285635e-01	7.785600e-01	4.848127e-01	6.418388e-01
min	-8.000000e+00	-8.000000e+00	-7.765625e+00	-8.000000e+00	-8.000000e+00	-8.000000e+00
25%	-6.093750e-01	-1.031250e+00	-7.812500e-02	-1.093750e+00	-3.125000e-01	-8.125000e-01
50%	-3.906250e-01	-8.125000e-01	1.093750e-01	-9.062500e-01	-1.250000e-01	-9.375000e-02
75%	-2.343750e-01	1.562500e-02	4.687500e-01	-2.187500e-01	3.125000e-02	1.406250e-01
max	6.250000e+00	7.765625e+00	7.984375e+00	6.187500e+00	7.984375e+00	7.984375e+00

Figura 22 Risultato comando describe

Il comando *"nununique()"* mostra che ogni attributo possiede sempre meno di 1000 valori distinti

```
Person       33
Activity      9
wx          806
wy          643
wz          587
tx          778
ty          956
tz          924
dtype: int64
```

Figura 23 Risultato comando nunique

Questa informazione ci è molto utile durante la fase di discretizzazione dei dati di cui parleremo nel capitolo dedicato all'approccio sig2vec. Dunque approfondiremo questa informazione più avanti.

La matrice mostrata nella figura in seguito mostra la correlazione fra le colonna delle attività(le classi) e le altre colonne attributi. Come possiamo notare non c'è una correlazione netta fra nessun attributo e una classe. Questo ci indica che nessun valore o range di valori di un attributo preso individualmente può essere utilizzato come predittore di una specifica classe e possiamo concludere, dunque, di essere davanti ad un effettivo problema inerente alle time series.

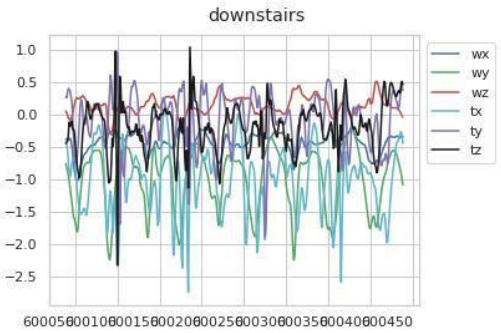
Figura 24 Matrice di correlazione fra dimensioni

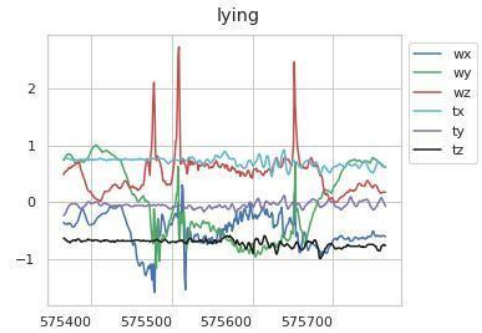
Per continuare l'analisi del dataset abbiamo filtrato i nostri dati rispetto alle attività svolta da un solo soggetto dell'esperimento ed abbiamo estratto i primi 400 (4 secondi) campioni per ogni attività.

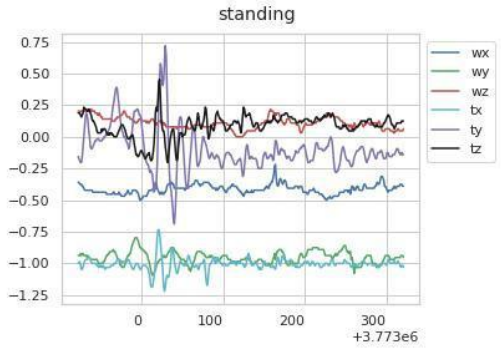
Dalle figure notiamo che già utilizzando 4 secondi si notano pattern ben definiti che ben caratterizzano e permettono di distinguere le attività. Pertanto possiamo utilizzare finestre temporali di 4 secondi per dividere le serie temporali.

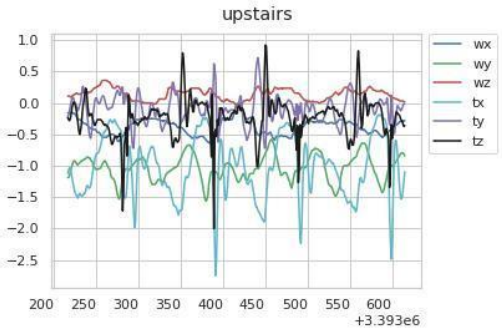
Mostriamo infine un grafico della distribuzione dei valori sull'intero dataset.

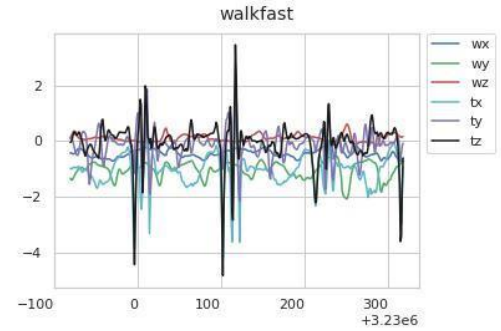
Vediamo come le classi siano distribuite uniformemente ad esclusione della classe identificata con 0 (downstairs) e con 5(upstairs).

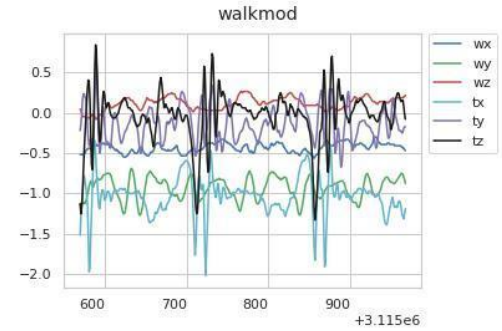

downstairs
wx
wy
wz
tx
ty
tz

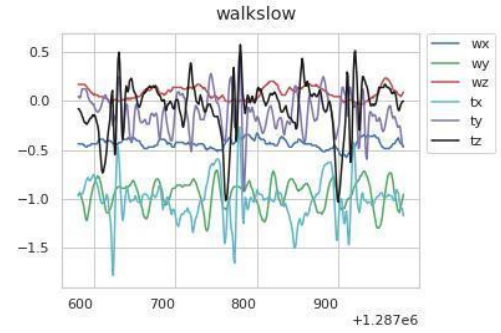

lying
wx
wy
wz
tx
ty
tz

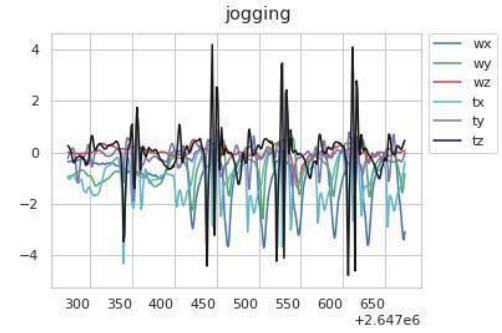

standing
wx
wy
wz
tx
ty
tz
+3.773e6


upstairs
wx
wy
wz
tx
ty
tz
+3.393e6


walkfast
wx
wy
wz
tx
ty
tz
+3.23e6


walkmod
wx
wy
wz
tx
ty
tz
+3.115e6


walkslow
wx
wy
wz
tx
ty
tz
+1.287e6


jogging
wx
wy
wz
tx
ty
tz
+2.647e6

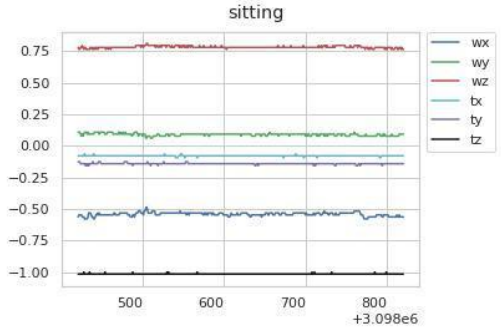

sitting
wx
wy
wz
tx
ty
tz
+3.098e6

Figura 25 Grafici dei primi 4 secondi per ogni di attività svolta da un volontario. Vediamo come siano già ben definiti dei pattern che identificano l'azione in svolgimento

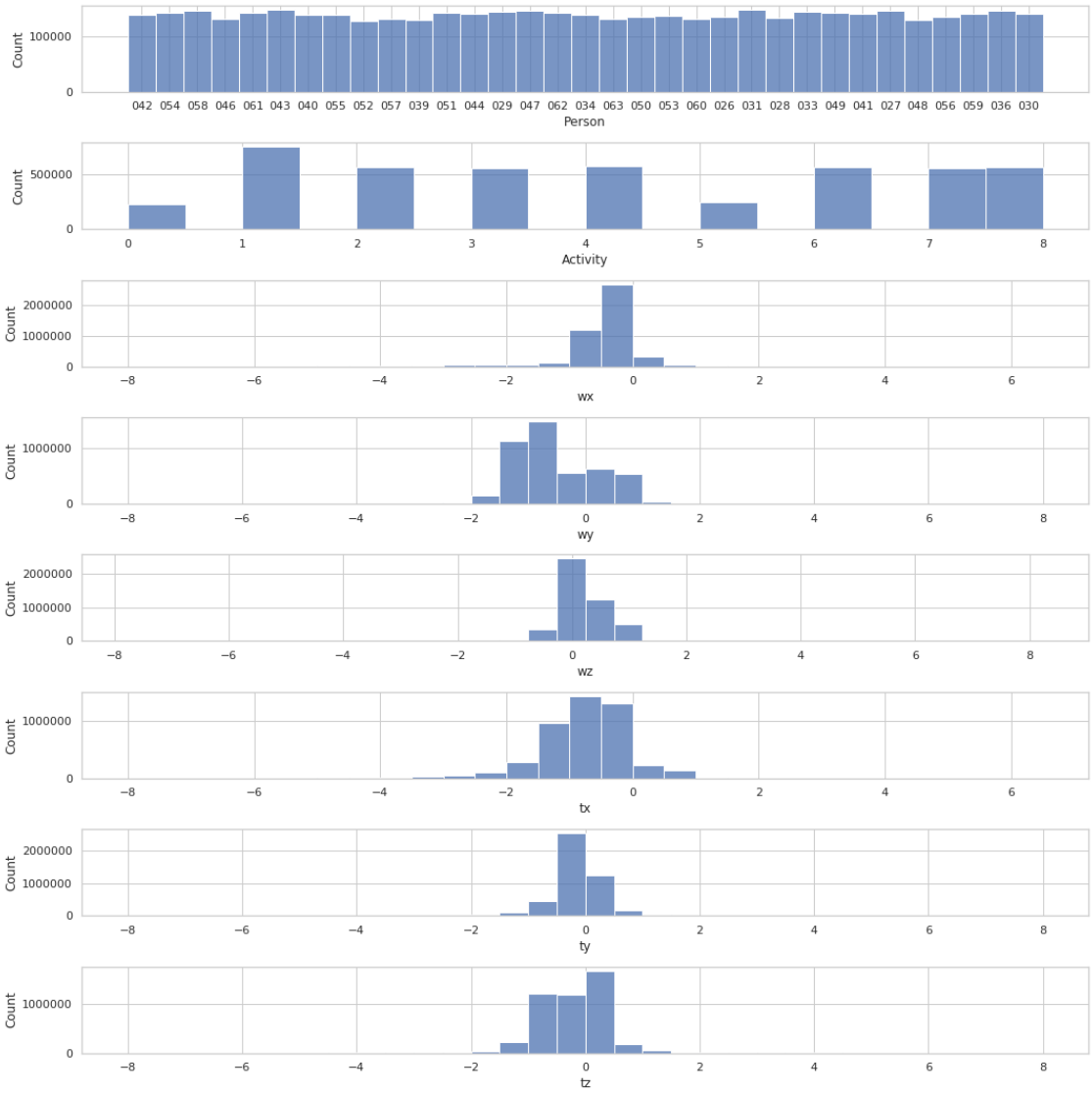


Figura 26 Distribuzione dati

Nella figura successiva invece mostriamo la distribuzione dei valori nelle time series costruite dopo l'operazione di windowing. Anche in questo caso i valori risultano distribuiti uniformemente tra tutte le time series.

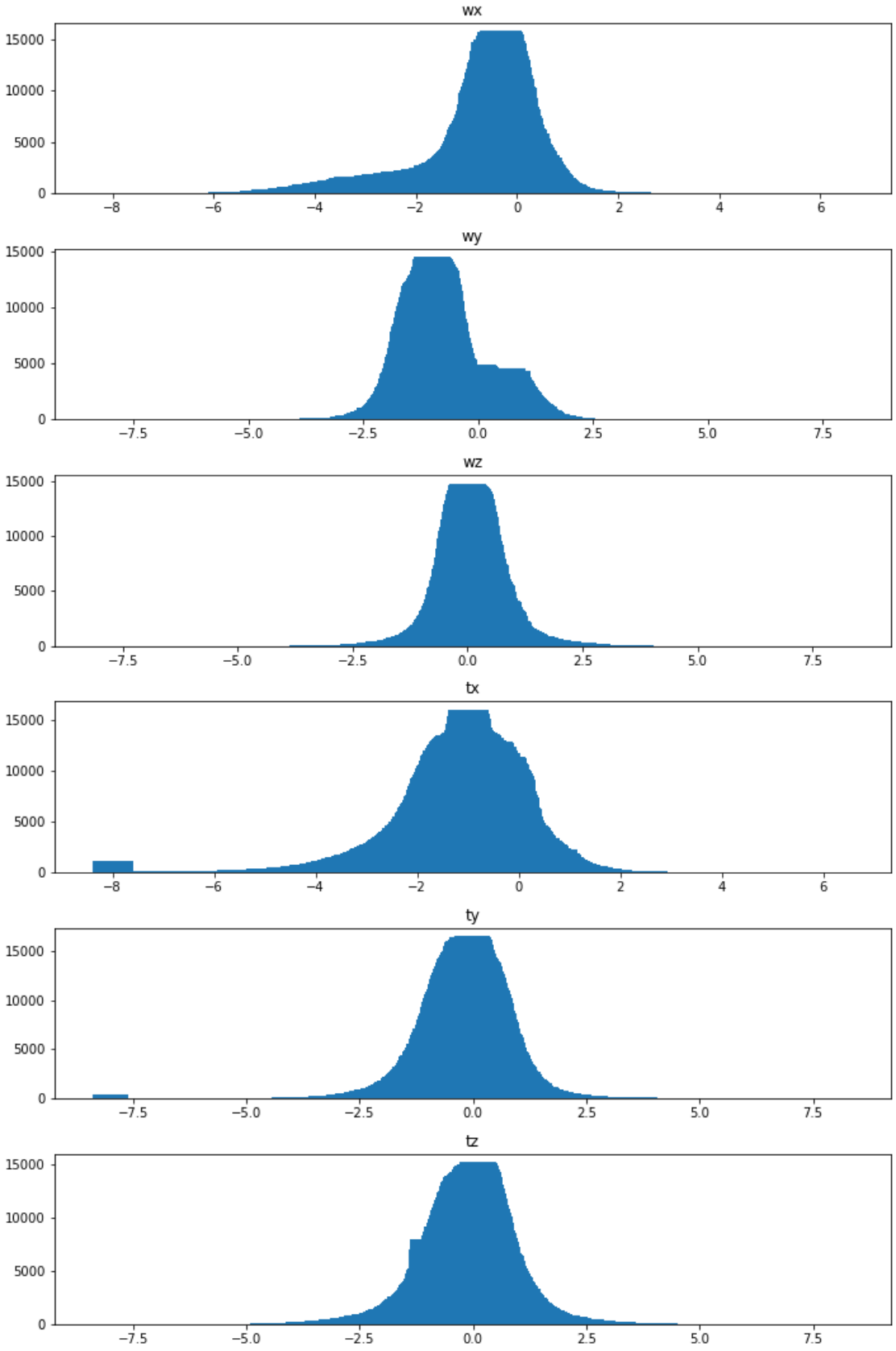


Figura 27 Distribuzione valori dopo windowing

Pipeline del lavoro

La pipeline di lavoro segue questo schema:

1. Dopo aver caricato e analizzato i dati, vengono predisposte le due architetture di embedding, l'autoencoder convoluzionale e il modello gensim per doc2vec.
2. Il primo step è il windowing sulle time series. A seconda del dataset abbiamo utilizzato lunghezze delle finestre e slide (spostamento della finestra) diversi scelti in base all'analisi dei pattern mostrati in precedenza.
3. Per entrambi i modelli effettuiamo una grid search durante il training in modo da trovare la lunghezza migliore del vettore di embedding di output. Questa fase viene analizzata tramite l'analisi della *"curse of dimensionality"* che approfondiremo più avanti.
4. Successivamente, scelta la migliore grandezza, costruiamo un dataset in cui mappiamo ogni MTS con il rispettivo vettore di embedding utilizzando uno dei due modelli appena allenati per predire gli embedding vector a partire dalle singole MTS.
5. Il nuovo dataset viene utilizzato come input degli algoritmi di clustering sui quali facciamo tuning dei parametri caratteristici. Viene anche utilizzato l'algoritmo supervised support vector machine per studiare la qualità dell'embedding quando applicato ad un algoritmo supervisionato.
6. Infine vengono utilizzate le metriche citate nel capitolo precedente per validare i risultati degli algoritmi di machine learning.

Per entrambi gli approcci vengono utlizzate due distanze sia per calcolare le distanze reciproche fra i vettori di embedding sia per il calcolo delle varie metriche per valutare la bontà dei risultati:

- Distanza euclidea:

$$\sqrt{\sum_{k=1}^{n}(p_k - q_k)^2}$$

- Distanza del coseno

$$1 - \frac{\boldsymbol{A} \cdot \boldsymbol{B}}{|\boldsymbol{A}||\boldsymbol{B}|}$$

$$\boldsymbol{A}\ e\ \boldsymbol{B}\ vettori\ di\ embedding$$

Approccio con autoencoder

Come introdotto in precedenza l' autoencoder ha l'obiettivo di creare una rappresentazione compressa dell'input di partenza. Questo avviene allenando il modello a creare uno spazio vettoriale di embedding che approssimi la distribuzione dei dati appartenenti al dataset di train.

Il processo di apprendimento avviene minimizzando l'errore di costruzione tra l'input e l'output generato.

Nel nostro caso abbiamo implementato una architettura di encoder formata all'incirca da 300mila parametri per l'encoder e circa 150mila parametri per il decoder.

Le architetture utilizzate nelle sperimentazioni sono state scelte dopo un tuning durante il quale abbiamo provato diverse configurazioni facendo variare il numero di parametri, i tipi di layer

utilizzati e le funzioni di attivazione. Pertanto i risultati riportati in questo paper sono inerenti alle sole architetture citate che abbiamo ritenuto essere le migliori.

Architettura dell'encoder

L'architettura dell'encoder è formata da una successione di blocchi costituiti da un layer convoluzionale unidimensionale, un layer di batch normalization e la funzione di attivazione Relu.

Un layer di batch normalization è una tipologia di strato che è dimostrato renda il processo di training della rete neurale più veloce e stabile.

Se denotiamo *B* come un mini-batch di grandezza *m* di dati del nostro dataset, possiamo definire la media empirica (μ) e la varianza empirica (σ) come:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2$$

Per uno strato della rete con input d-dimensionale, $\{x=(x^1,\ldots, x^d)\}$, ogni dimensione del suo input viene quindi normalizzata (cioè ricentrata e ridimensionata) separatamente:

$$\widehat{x_i^{(k)}} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{\sigma_B^{(k)^2} + \epsilon}} \text{ dove } k \in [1, d] \text{ e } i \in [1, m]$$

$\mu_B^{(k)}$ and $\sigma_B^{(k)}$ sono rispettivamente la media e la varianza per dimensione.

Per ogni blocco di layer vengono settati diverse grandezze dei filtri, features maps generate e valore dello stride (spostamento).

Abbiamo evitato di usare layer di pooling in quanto abbiamo testato empiricamente delle perfomance peggiori.

Infine sono stati inseriti dei fully connected layers alternati da funzioni RELU.

Architettura del decoder

Il decoder presenta quattro "*residual block*" iniziali [19], seguiti da blocchi costituiti da layer convoluzionali unidimensionali trasposti [20] (un tipo di layer che esegue l'operazione inversa della convoluzione, di fatto aumentando le dimensioni dell'output) e layer di batch normalization seguiti da una funzione di attivazione di tipo RELU (quindi una architettura quasi speculare all'encoder)

Al termine di tutti i blocchi vi è uno strato convoluzionale trasposto finale.

A seconda del tipo di normalizzazione che viene effettuata in fase di preprocessing l'ultimo strato può essere seguito da una RELU (nel caso di una normalizzazione minmax, quindi con valori compresi tra 0 e 1) o lineare (nel caso in cui i dati non vengano normalizzati o venga applicata una standardizzazione rispetto a media e varianza).

In output il decoder emette quindi una unica UTS di lunghezza uguale a quella di input.

L’obiettivo di tutto il processo è di allenare un buon encoder che crei uno spazio di embedding che sia grado di conservare le caratteristiche importanti per il clustering delle time series seppur in uno spazio a dimensionalità ridotta. Al termine della fase di allenamento, infatti, metteremo da parte il decoder e utilizzeremo i vettori prodotti dall’encoder a partire dagli input per creare il dataset da utilizzare con gli algoritmi di clustering.

Funzioni di loss

Per il nostro esperimento abbiamo utilizzato una funzione di loss customizzata che è la somma pesata delle 4 funzioni: *MSE, MAE, Logcosh e HuberLoss*.

Definizione parametri

Per la fase di training abbiamo utilizzato l’ottimizzatore Adam con learning rate a 0.01 e un learning rate scheduler che riduce il learning rate di un valore pari a 0.9 per ogni epoca dopo la decima.

Il processo di apprendimento dura 50 epoche, durante ogni ciclo viene allenato un autoencoder a codificare una UTS appartenente alla MTS fino a creare N modelli capaci di codificare tutte le N UTS delle MTS.

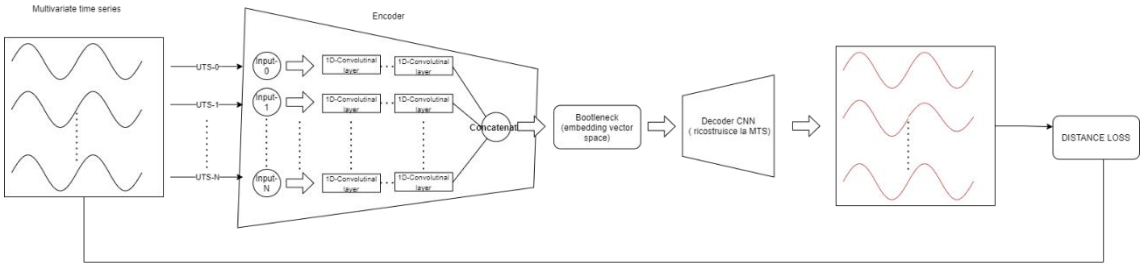


Figura 28 Training dell'autoencoder

Ogni ciclo completo di allenamento avviene all’interno di una grid-search implementata tramite ciclo for.

Definita con TSLENGTH la lunghezza della UTS estratta dal windowing, i parametri che vengono valutati della grid-search sono:

- Grandezza del vettore di embedding (Embedding_size) : 800, 400, 150 , TSLENGTH/2, TSLENGTH/8, TSLENGTH/10, 15, 5 features
- Normalizzazione dei dati: min-max, standard scaler, nessuna normalizzazione.

Per quanto riguarda la funzione di loss customizzata, abbiamo assegnato un peso di 0.25 ad ognuna delle 4 funzioni di loss che la compongono.

Fase di embedding

La fase di embedding prevede due sottofasi.

Una volta allenato l’autoencoder (quindi tutti gli N sotto modelli) a codificare tutte le UTS componenti le MTS estraiamo i features vector del dataset invocando la funzione *“predict()”* di ogni modello sulle rispettive UTS creando di fatto un nuovo dataset di features vector per ogni UTS.

A questo punto il flusso si divide in due flussi paralleli:

1. Nel primo flusso concateniamo gli N feature vector ottenuti da ogni MTS formando un unico vettore di grandezza *N*Embedding_size* e passiamo quindi alla fase finale di clustering.
2. Nel secondo flusso utilizziamo un ulteriore autoencoder che riceve in input gli N features vector ed esegue un ulteriore fase di train durante la quale viene allenato ad apprendere una nuova rappresentazione che sia in grado di comprimere i features vector di grandezza *N*Embedding_size* in unico vettore di grande embedding size features. In questo modo dovremmo ottenere una rappresentazione compressa di tutta la MTS senza bisogno di concatenare i singoli features vector delle UTS. Successivamente a questa fase si procede con la fase di clustering.

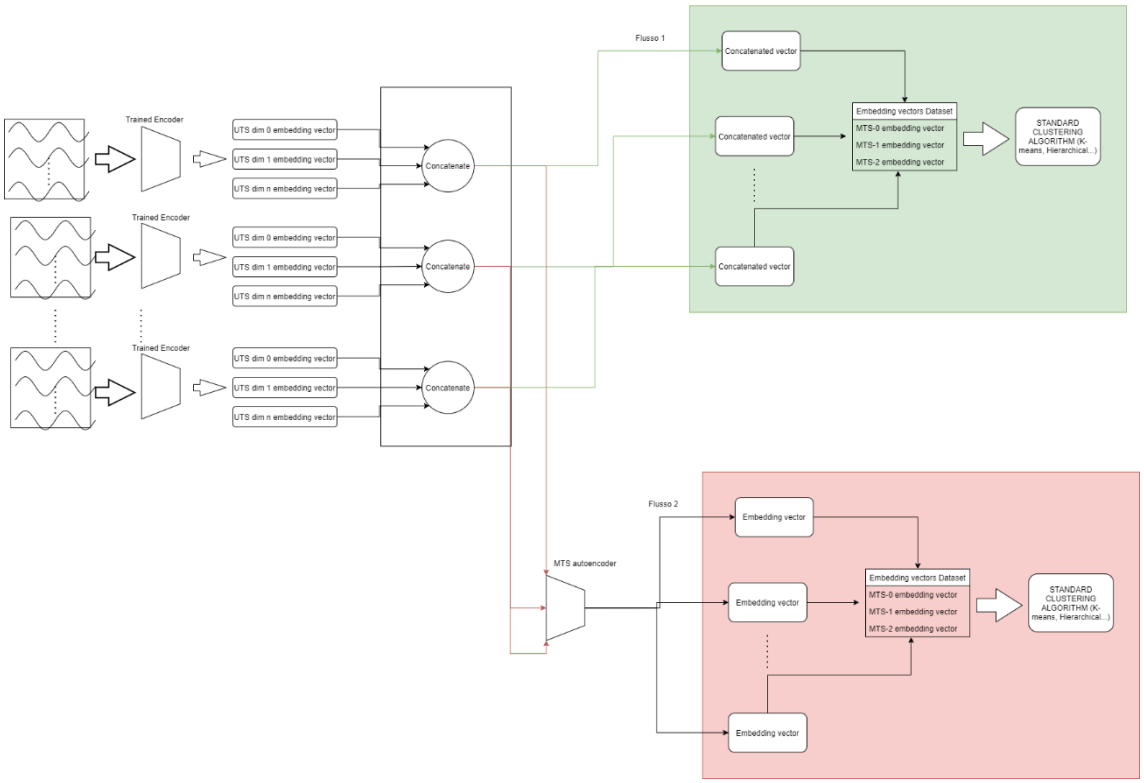


Figura 29 Fasi di embedding per l'approccio con CNN

Architettura autoencoder per MTS

Per il secondo flusso abbiamo inserito una architettura aggiuntiva, formata da una serie di fully connected layer alternati da una funzione di attivazione RELU che prende in input gli N future vector concatenati e tenta di ricostruire la MTS di partenza. L'architettura è composta da circa 10.000 parametri da apprendere.

Approccio Sig2vec

Nell'approccio sig2vec trattiamo l'embedding delle MTS come un problema di tipo NLP.

Una volta ottenuto un dataset di MTS di stessa lunghezza tramite windowing, come già spiegato in precedenza, consideriamo ogni MTS come un documento.

Per meglio spiegare l'approccio possiamo immaginare una MTS come un testo composto da frasi in diverse lingue. Ogni UTS componente la MTS rappresenta un documento in una determinata

lingua (quindi trattiamo ogni UTS di una dimensione come se avesse una "semantica", una "grammatica" e un significato diverso dalle altre UTS appartenenti alla stessa MTS). I valori che assume in ogni istante di tempo una UTS sono le parole.

Quindi presa l'UTS della dimensione *i* di una MTS considereremo il valore all'istante $t_{i,0}$ come una parola , il valore alle $t_{i,1}$ come la parola successiva e così via. L'insieme di queste parole è un documento in una determinata lingua, più documenti in N (con N dimensionalità della MTS) di lingue diverse formano l'intera MTS.

Applicheremo l'algoritmo di doc2vec ad ognuna delle UTS (e quindi ad ogni documento) estraendo un vettore di embedding per ogni UTS e infine concateneremo tutti i vettori in un unico vettore di embedding che utilizzeremo per il fitting degli algoritmi di clustering.

Si pone però il problema di discretizzare i valori all'interno di UTS.

Una UTS è formata da valori potenzialmente continui, mentre il vocabolario di una lingua è formato da un numero finito di vocaboli per questo è necessario, eventualmente, discretizzare i dati.

Definizione approccio sig2vec

Gli step che seguiremo in questo approccio sono:

1. Analisi delle UTS nella loro intera lunghezza: questa fase precede quella di windowing e consiste nel prendere in considerazione l' i-esima dimensione (*i-esima* UTS) dell'intera MTS e contare il numero di valori distinti che appaiono in quella dimensione. Se il numero di valori distinti è superiore ad una soglia (nel nostro caso abbiamo messo come soglia un massimo di 1000 valori distinti che corrisponderebbero a 1000 parole distinte nel nostro vocabolario per la UTS *i-esima*) si procede con lo step 2 di discretizzazione altrimenti si procede direttamente con la fase 3. Questo procedimento viene applicato a tutte le dimensioni della MTS.
2. Discretizzazione: si applica iterativamente l'algoritmo di K-means su tutti i valori che compongono l'intera una UTS presa in considerazione facendo variare il valore K (numero di cluster) e utilizzando una metrica quale SSE o Silhouette per valutare la qualità dei cluster generati. Si fa il grafico detto "scree plot" e si ricerca il *"gomito"* o "*ginocchio*" per individuare il numero di cluster ottimale. Il numero di cluster corrisponde in sostanza al numero di parole distinte che vogliamo nel nostro vocabolario. Infine si sostituiscono i valori della UTS col la label (o il baricentro unidimensionale) del cluster al quale viene assegnato dall'algoritmo K-means appena allenato.
3. Windowing: facciamo windowing delle time series con lo stesso approccio presentato al capitolo precedente.
4. Train dei modelli *doc2vec*: in questa fase alleniamo l'algoritmo doc2vec su ogni dimensione in maniera separata in modo da generare N modelli doc2vec allenati a predire i vettori di embedding per ogni specifica dimensione delle MTS.
5. Concatenazione: in questa fase i vettori vengono concatenati in modo da ottenere un unico vettore di lunghezza *N*Embedding_size* e avere un solo vettore di embedding per ogni MTS.
6. Generazione dataset: si genera un dataset composto da tutti i vettori appena generati.

7. Clustering: i vettori vengono clusterizzati utilizzando diversi algoritmi di clustering e validati tramite le metriche introdotte in precedenza

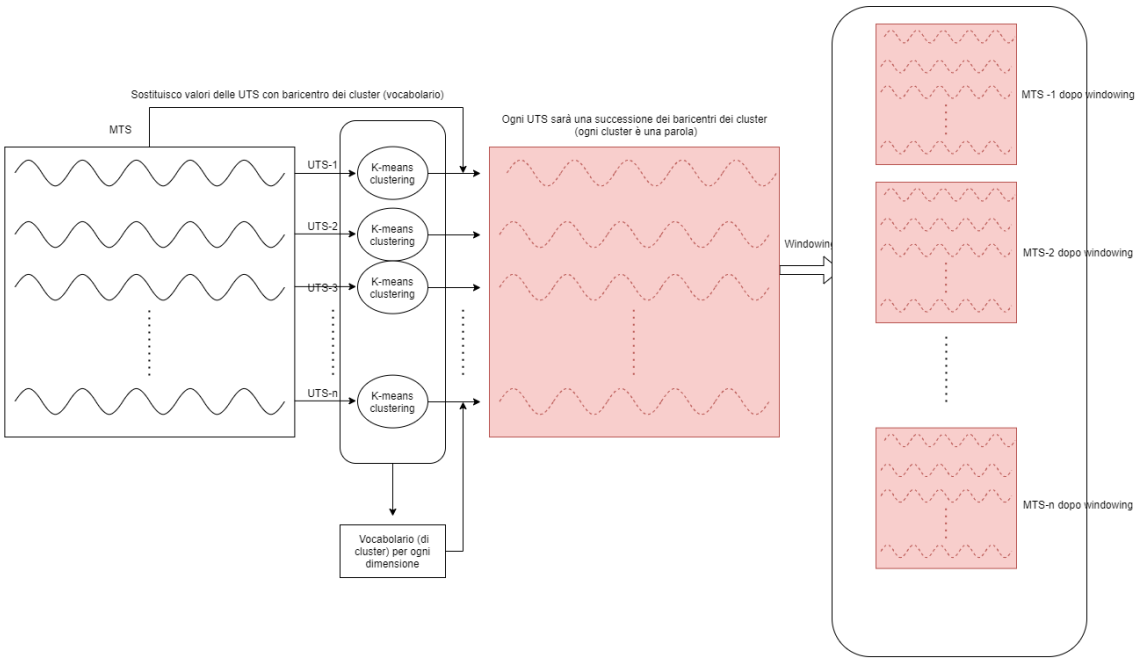


Figura 30 Fasi di discretizzazione e windowing per sig2vec

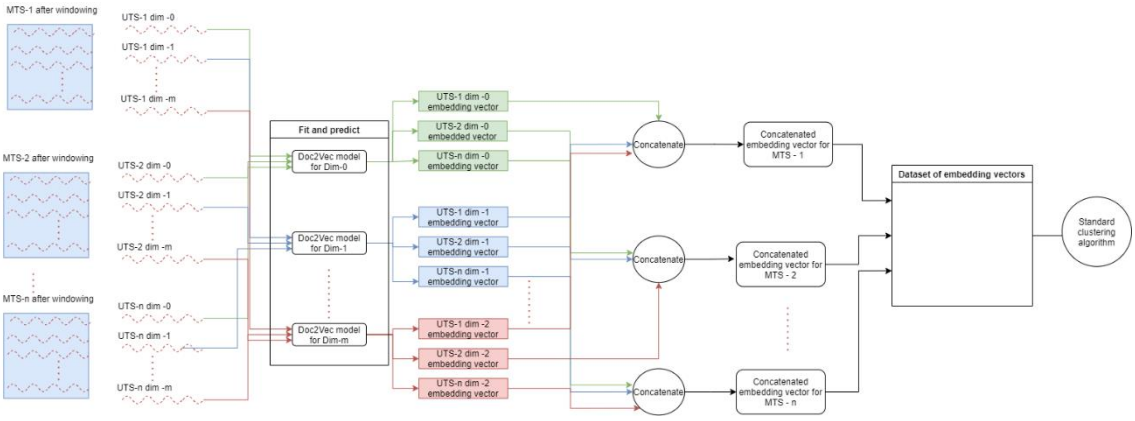


Figura 31Fase di embedding e clustering per sig2vec

Definizione parametri

Per tutta la fase di embedding abbiamo implementato una grid search su questi parametri:

Definita con TSLENGTH la lunghezza della UTS estratta col windowing, i parametri che vengono valutati della grid-search sono:

- Grandezza del vettore di embedding (Embedding_size) : 800,400,150,TSLENGTH/2, TSLENGTH/8, TSLENGTH/10, 15, 5
- Algoritmo di allenamento di doc2vec: PV-DM o PV-DBOW

- Winsize: parametro che definisce il range di parole immediatamente precedenti e immediatamente successive da considerare per il la predizione della parola successiva. Valori possibili: 2,5,8,10,50.

Analisi per curse of dimensionality

La scelta della grandezza del vettore di embedding è guidata dall'analisi della curse of dimensionality. Questa avviene al termine di ogni ciclo completo di training; una volta che l'autoencoder MTS o i modelli doc2vec sono stati allenati si genera il dataset di vettori di embedding.

Successivamente si calcola la matrice di distanza fra ogni embedding vector e tutti gli altri, utilizzando una specifica metrica.

Dopo la creazione della matrice delle distanze si definisce una metrica che serva a valutare la "variabilità" dei valori che compongono i features vector. Si fa ciò perché empiricamente si nota che col crescere della grandezza dei features vector la varianza dei valori diminuisce fino a rendere le distanze fra diversi vettori irrilevanti ai fini della classificazione.

Questo è detto fenomeno di Hughes che mostra che all'aumentare del numero di features, aumentano anche le prestazioni del classificatore fino a raggiungere il numero ottimale di features. L'aggiunta di più features ridurrà quindi le prestazioni del classificatore non rendendo più le distanze fra buona parte dei punti del dataset rilevanti.

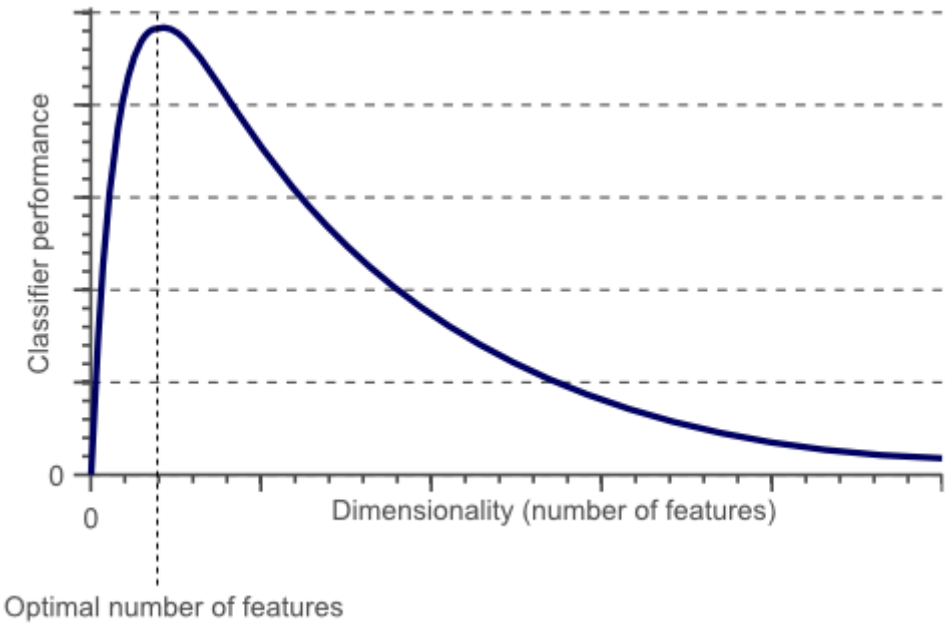


Figura 32 Esempio grafico per analisi CUD

Per l'analisi di procede in questo modo:

- Si pongono in ordine crescente sull'asse delle ascisse le lunghezze dei features vector
- Si pongono sull'asse delle ordinate il risultato ottenuto applicando una delle metriche come definite in precedenza per ogni lunghezza del features vector

- Si sceglie il valore di lunghezza del vettore di embedding tale per cui la metrica sia in una fase intermedia del proprio valore.

Utilizziamo tale grandezza per generare i features vector da utilizzare nella fase successiva, ovvero quella di clustering.

La formula utilizzata per il calcolo della curse of dimensionality è la seguente:

$$CUD = \frac{MaxDist - MinDist}{MinDist}$$

Analisi per K selection

Con l'analisi per la selezione del numero di cluster studiamo la qualità dei raggruppamenti al variare di K utilizzando i vettori di embedding generati. Questa analisi ci aiuta a capire ulteriormente se la grandezza dei vettori e i parametri scelti siano utili al nostro obiettivo di clusteriing dei dati.

Questa analisi prevede la conoscenza del numero esatto di cluster del nostro dataset e segue questi step:

1. Si applica l'algoritmo di K-Means sui features vector generati facendo variare il parametro K (numero di cluster da un valore piccolo, tipicamente 2 o 3, fino ad un valore scelto arbitrariamente maggiore del numero effettivo di cluster). Noi abbiamo scelto un range compreso tra 3 e il numero effettivo di cluster + 10.
2. Per ogni valore di K si calcola la qualità dei cluster utilizzando una metrica predefinita, noi abbiamo scelto SSE(sum of squared error).
3. Si crea un grafico mettendo sull'asse delle ascisse il valore di K crescente e sulle ordinate il valore dell'SSE corrispondente.
4. Si cerca il gomito del grafico, se questo appare nei dintorni del numero reale di cluster o comunque si nota un trend discendente del grafico in quel punto si può concludere che i parametri scelti potrebbero essere validi.

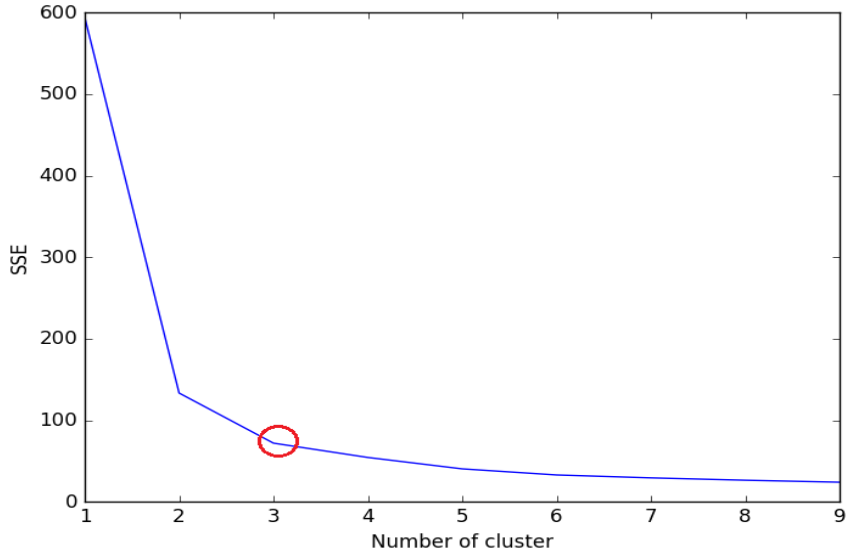


Figura 33 Grafico della K selection. In rosso è segnato il gomito del grafico ed indica quale potrebbe essere il numero ideale di cluster.

Clustering e validazione

L'ultimo step prevede l'applicazione di alcuni algoritmi di clustering sui nuovi dataset appena generati. Gli algoritmi saranno allenati sull'intero dataset e i risultati validati sui medesimi dati tramite le metriche già illustrati nei capitoli precedenti, ovvero:

- Silhouette
- Random Index
- Adjusted Random index

Gli algoritmi che abbiamo usato sono:

K-Means nell'implementazione originale che utilizza la distanza euclidea come metrica per il calcolo dei baricentri dei cluster. L'algoritmo è stato settato per raggiungere un massimo di 500 iterazioni e facendo 20 re-inizializzazioni casuali dei baricentri dei cluster

Agglomerative Clustering utilzzando tre metodi di linkage: ward, average, e single(minumum)

DBSCAN: i parametri su cui abbiamo effettuato la gridsearch sono eps e minpoints. Il range di minpoints è lo stesso per tutti i modelli e tutti i dataset di embedding vector e cioè: 2,3,5,10,50,100

In alcuni casi non è stato possibile raccogliere i risultati peri valori 50 e 100 in quanto l'algoritmo non riusciva a creare un numero sufficiente di cluster (tutti i punti diventavano outliers).

1. Il range dei valori per il parametro Eps è variabile e viene assegnato in base al modello utilizzato per l'embedding e i features vector studiando il grafico della distanza di un punto dal suo 4° più vicino:
2. Per ogni punto si calcola la distanza dal suo 4° punto più vicino
3. Sull'asse delle ordinate si pone la distanza del punto dal suo 4° vicino
4. Si pongono sulle ascisse di un grafico i punti ordinati in base alla distanza dal 4° vicino
5. Si cerca nel grafico, anche in questo caso, il "gomito" o "ginocchio" in modo da determinare eps.

Il range di valori di eps non deve superare il valore dell'ordinata in cui si verifica il gomito.

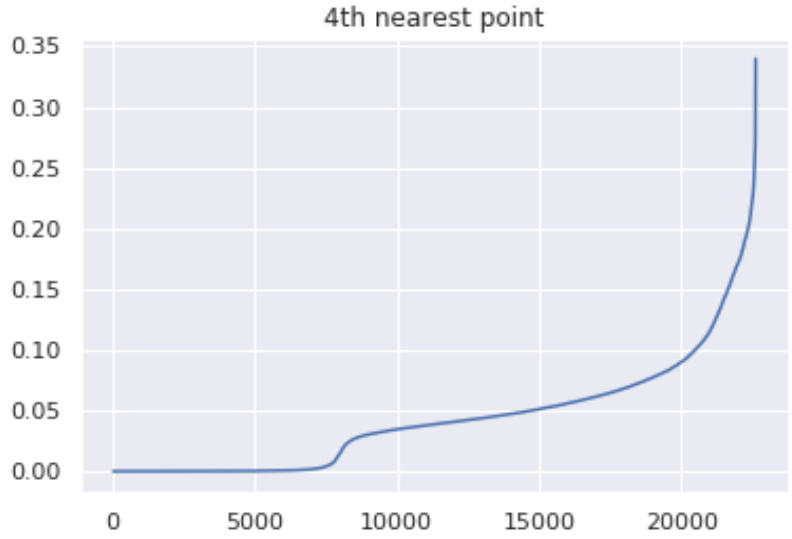


Figura 34 Grafico delle distanze rispetto al quarto punto più vicino estratto da un nostro esperimento. Notiamo come il gomito sia nei dintorni del valore 0.1

Ad eccezione del Kmeans e dell'agglomerative clustering versione Ward linkage che permettono solo l'utilizzo della distanza euclidea come metrica per ogni implementazione degli algoritmi di clustering abbiamo utilizzato sia la distanza del coseno sia la distanza euclidea.

PARTE III

In questa sezione presentiamo i risultati sperimentali e li confronteremo inoltre coi risultati ottenuti da altri paper.

A causa del notevole numero combinazioni di parametri non includeremo tutti i grafici e risultati testati, ma mostreremo e commenteremo soltanto i più significativi e i migliori.

Sig2vec

Cominciamo col presentare i risultati dell'approccio *sig2vec*. Vediamo nelle seguenti tabelle come varino le metriche prese sotto esame al variare dei tre parametri utilizzati.

ALGORITHM_EUCLIDEAN	*ARI*	*RI*	*Silhouette*
0	**0.28662**	**0.63973**	0.08362
1	0.24429	0.59282	**0.1355**

Tabella 1 Risultati medi al variare dell'algoritmi doc2vec con distanza euclidea

ALGORITHM_COSINE	*ARI*	*RI*	*Silhouette*
0	**0.26962**	**0.60338**	0.07132
1	0.1099	0.46325	**0.09624**

Tabella 2 Risultati medi al variare dell'algoritmi doc2vec con distanza del coseno

VECTOR_SIZE_EUCLIDEAN	*ARI*	*RI*	*Silhouette*
5	0.23368	0.5957	**0.24544**
15	0.27754	**0.62177**	0.13571
40	0.28459	0.62614	0.10001
50	**0.28487**	0.6239	0.09638
150	0.27385	0.6254	0.05602
200	0.27859	0.64987	0.05774
400	0.26488	0.61716	0.07028
800	0.27376	0.62363	0.06789

Tabella 3 Risultati medi al variare della grandezza del vettore di embedding con distanza euclidea

VECTOR_SIZE_COSINE	*ARI*	*RI*	*Silhouette*
5	0.18951	0.55831	**0.1304**
15	0.22997	**0.56381**	0.0802
40	0.22543	0.55265	0.05191
50	**0.24416**	0.55712	0.05741
150	0.21445	0.55527	0.06435
200	0.20836	0.55411	0.07885
400	0.19495	0.54289	0.0875
800	0.19573	0.54268	0.08906

Tabella 4 Risultati medi al variare della grandezza del vettore di embedding con distanza del coseno

WINSIZE_EUCLIDEAN	*ARI*	*RI*	*Silhouette*
2	0.25634	0.61067	0.10611
8	0.27431	0.62196	**0.11214**
10	0.27902	0.62886	0.09468
50	**0.28608**	**0.64143**	0.08452

Tabella 5 Risultati medi al variare della grandezza della finestra con distanza euclidea

WINSIZE_COSINE	*ARI*	*RI*	*Silhouette*
2	0.17626	0.52851	0.071
8	0.19967	0.53604	**0.09062**
10	0.24281	0.57825	0.08742
50	**0.27039**	**0.60391**	0.06931

Tabella 6 Risultati medi al variare della grandezza della finestra con distanza del coseno

CLUSTERING MODEL EUCLIDEAN	*ARI*	*RI*	*Silhouette*
Agglomerative_avg	0.00971	0.19079	**0.22905**
Agglomerative_min	-0.00088	0.15855	0.19699
Agglomerative_ward	0.41654	0.83898	0.0618
K_means_elkan	0.43549	0.85541	0.052
K_means_full	**0.43662**	**0.85586**	0.05175
K_medoids	0.33083	0.83748	0.02221

Tabella 7 Risultati medi al variare dell'algoritmo di clustering con distanza euclidea

CLUSTERING MODEL COSINE	*ARI*	*RI*	*Silhouette*
Agglomerative_avg	0.25948	0.63344	**0.15505**
Agglomerative_min	-0.00197	0.16003	0.01001
K_medoids	**0.37885**	**0.86532**	0.07582

Tabella 8 Risultati medi al variare dell'algoritmo di clustering con distanza del coseno

Nei grafici sopra abbiamo riportato i risultati medi ottenuti per ogni parametro scelto nei nostri esperimenti utilizzando rispettivamente la metrica euclidea e del coseno per calcolare le distanze fra i vettori di embedding.

Cominciamo col notare come l'algoritmo di clustering utilizzato faccia da forte discriminante e che sia nella versione con distanza euclidea che coseno, features vectors di dimensioni comprese tra 5 e 50 features per dimensione hanno risultati migliori per ogni metrica.

Gli algoritmi agglomerativi nelle versioni average e minimum risultano meno performanti in entrambe le implementazioni. Per quanto riguarda gli altri parametri non si notano forti variazioni del valore medio al variare dell'algoritmo doc2vec, lunghezza dell'embedding vector e grandezza della finestra.

A conferma di ciò mostriamo in seguito la matrice di correlazione fra i parametri e le varie metriche.

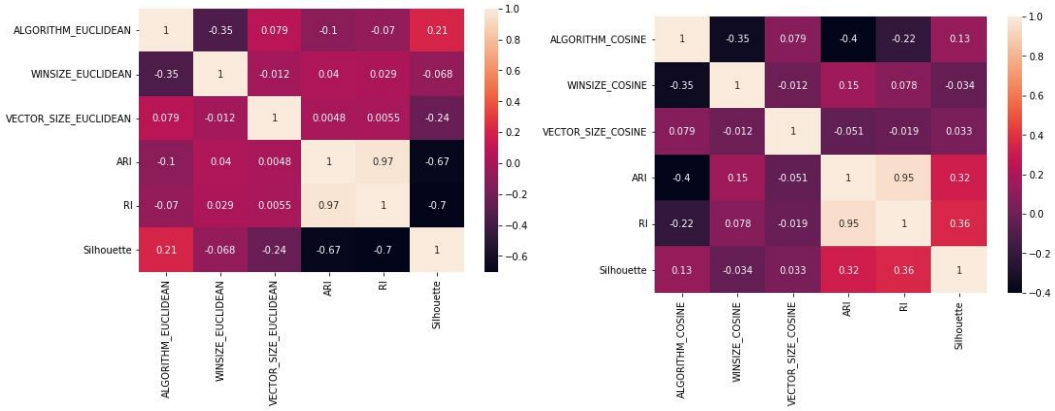


Figura 35 Correlazione tra iperparemetri per distanza Euclidea e del coseno

Notiamo come effettivamente anche da questa matrice non si rilevino importanti correlazioni fra parametri e risultati se non di lievi entità, mentre nell'implementazione con distanza euclidea random index, e adjusted random index abbiano una correlazione fortemente negativa con la silhouette.

Mostriamo ora i grafici dei risultati raggiunti da ogni metrica ed ogni combinazione di parametri. I risultati sono ordinati in ordine crescente in modo da poter individuare agevolmente la migliore combinazione per ogni metrica.

Distanza Euclidea

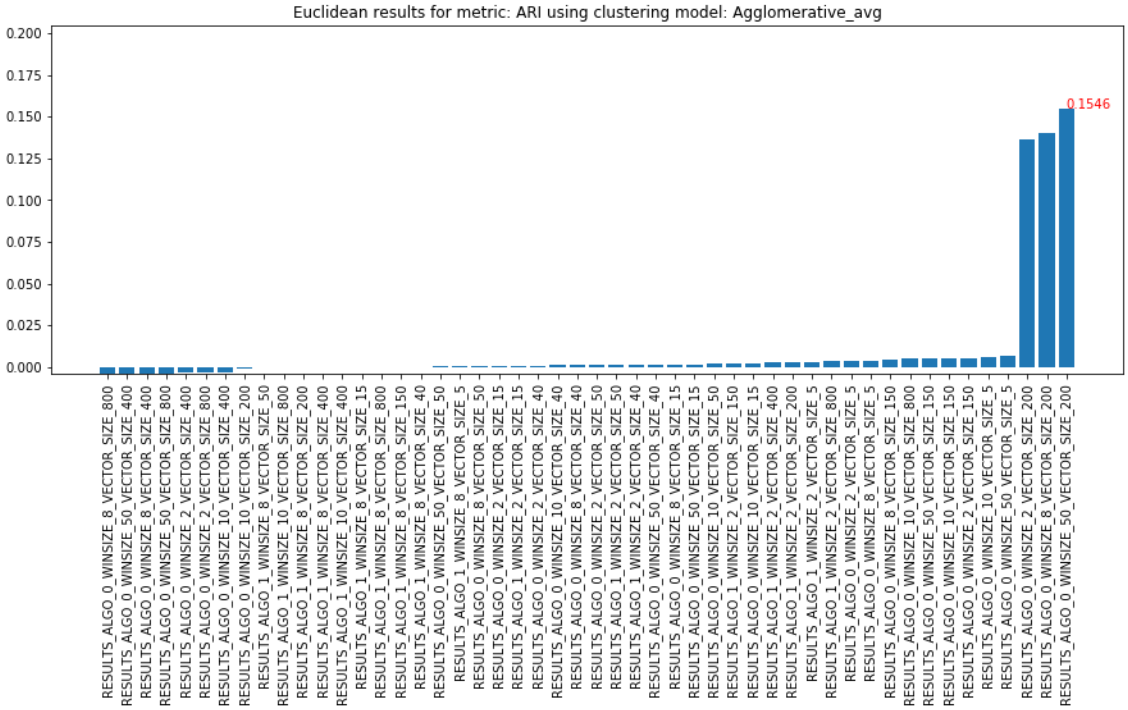
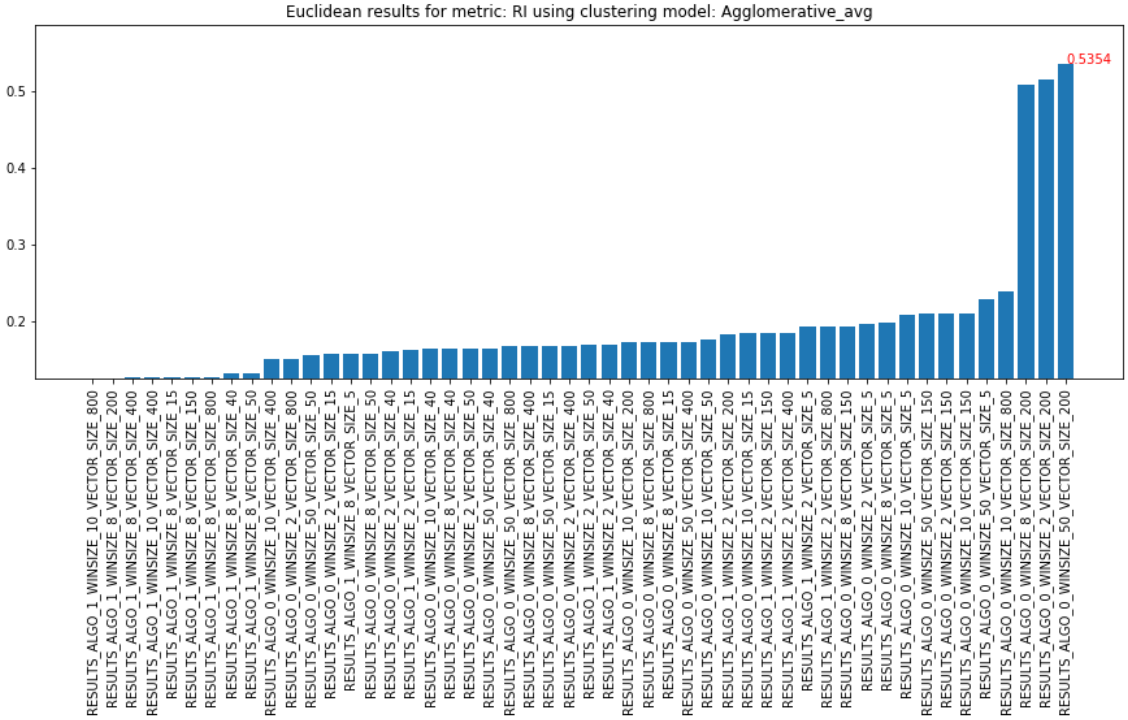
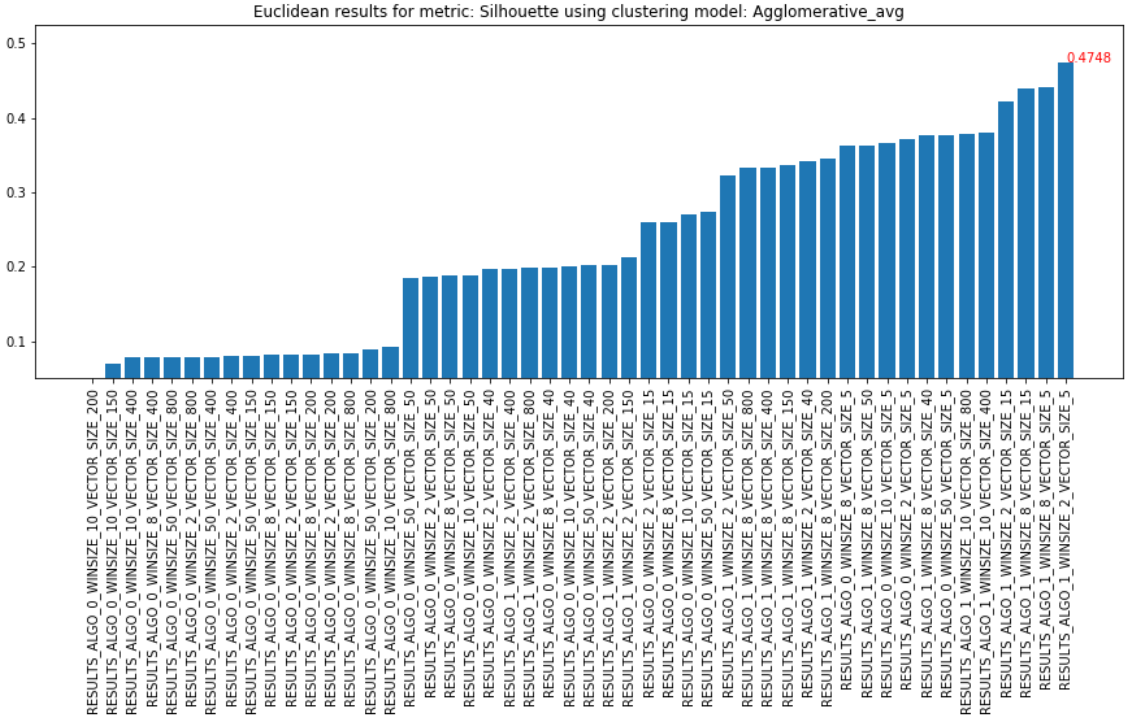
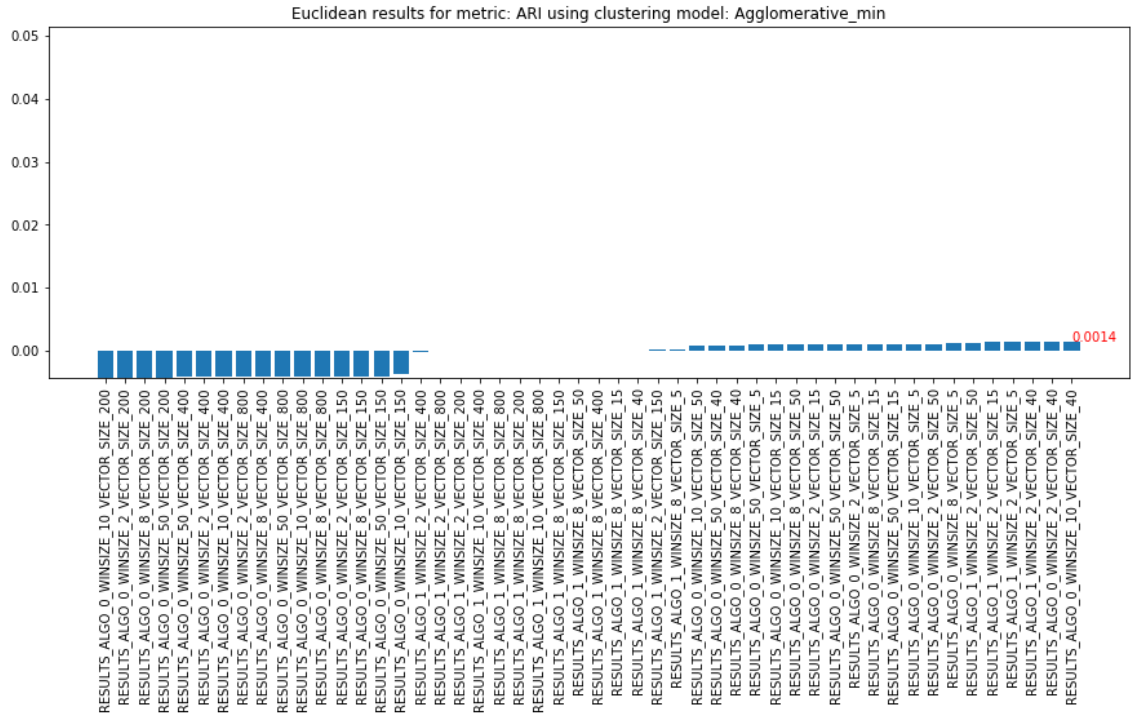
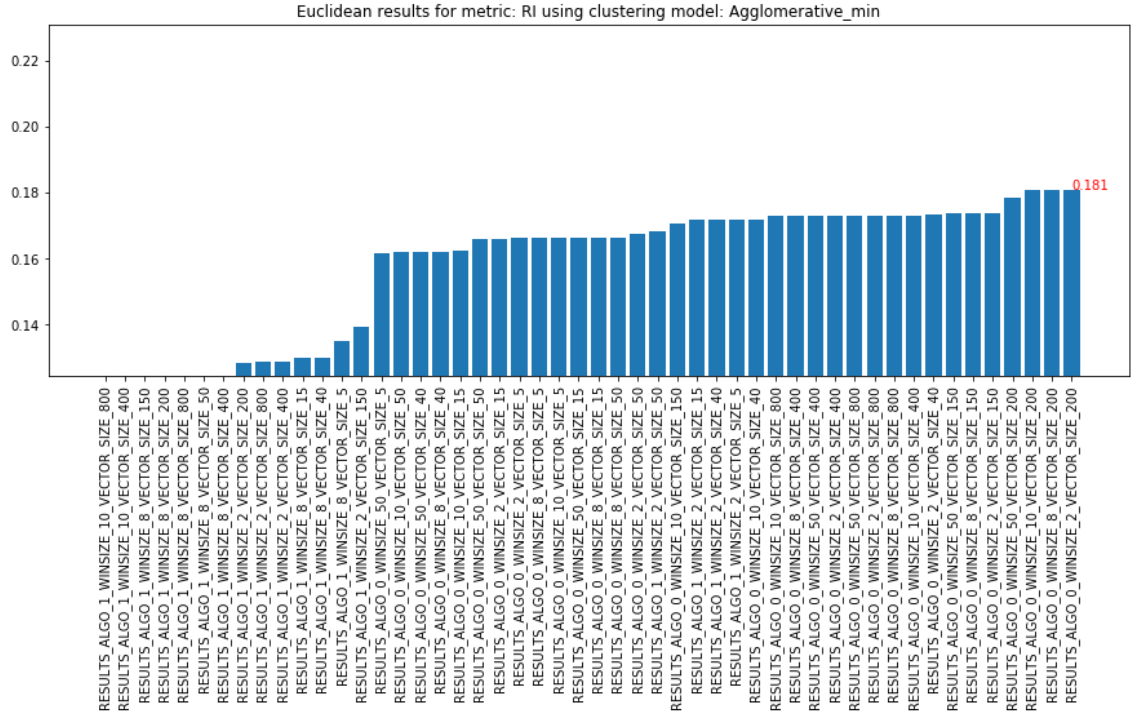


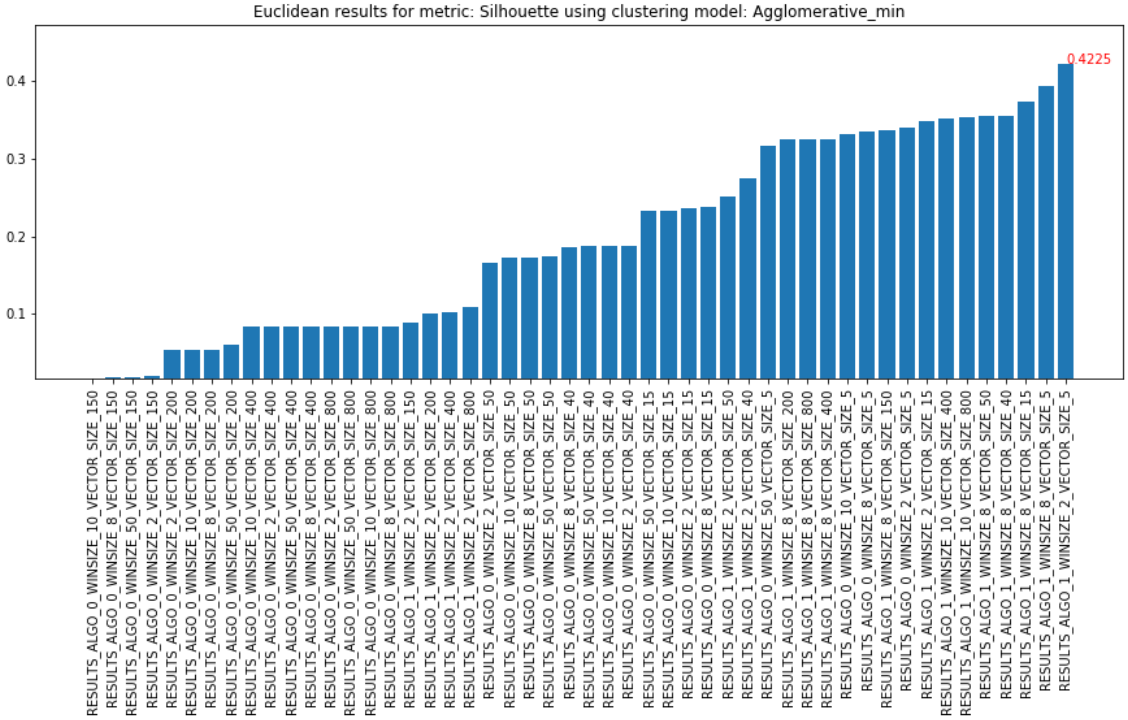
Figura 36 Grafici a barre risultati metriche al variare delle combinazioni fra gli iperparametri

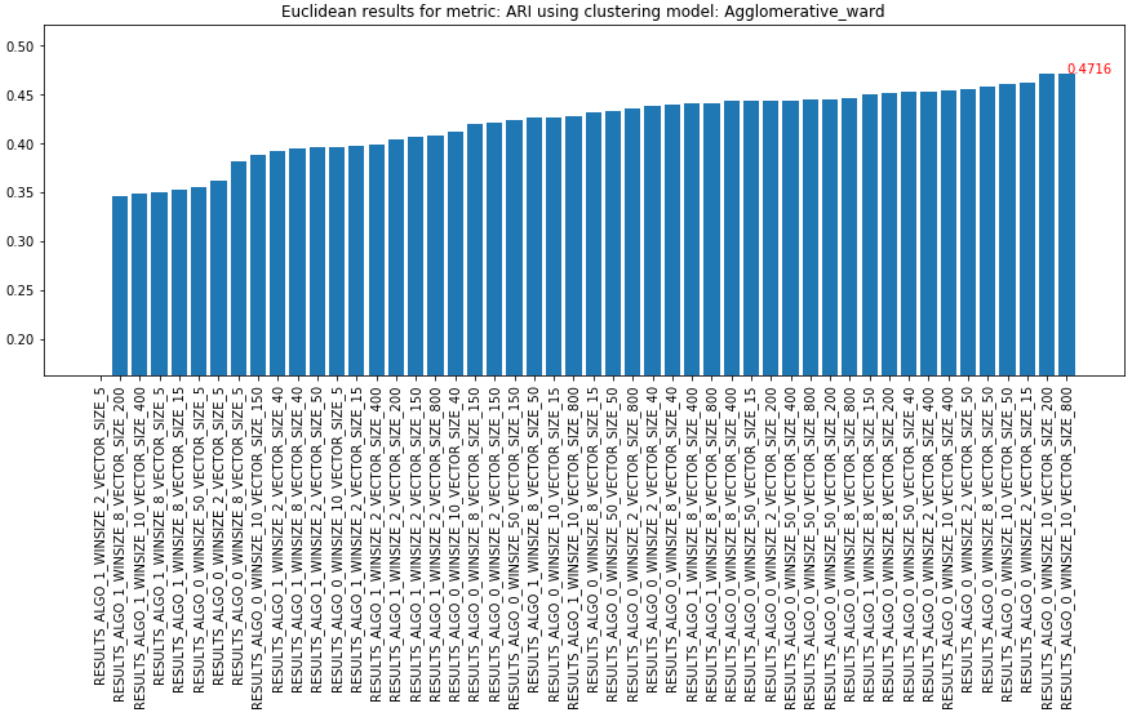

Euclidean results for metric: RI using clustering model: Agglomerative_avg
0.5354
0.5
0.4
0.3
0.2
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200

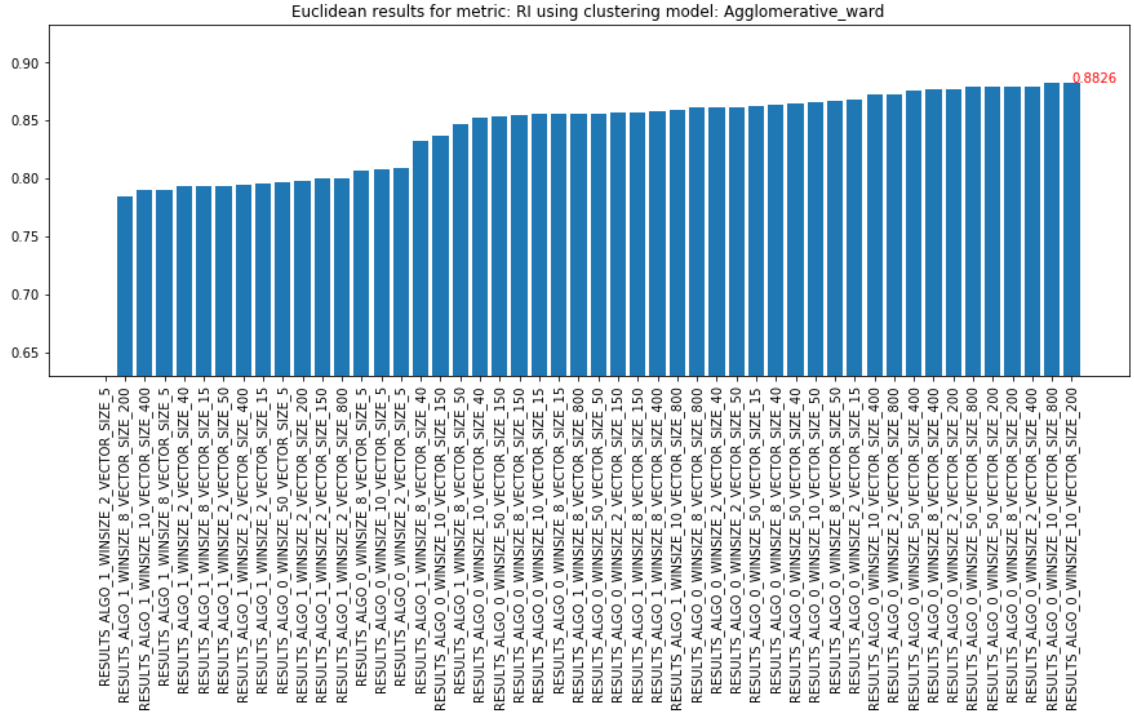

Euclidean results for metric: Silhouette using clustering model: Agglomerative_avg
0.4748
0.5
0.4
0.3
0.2
0.1
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_5

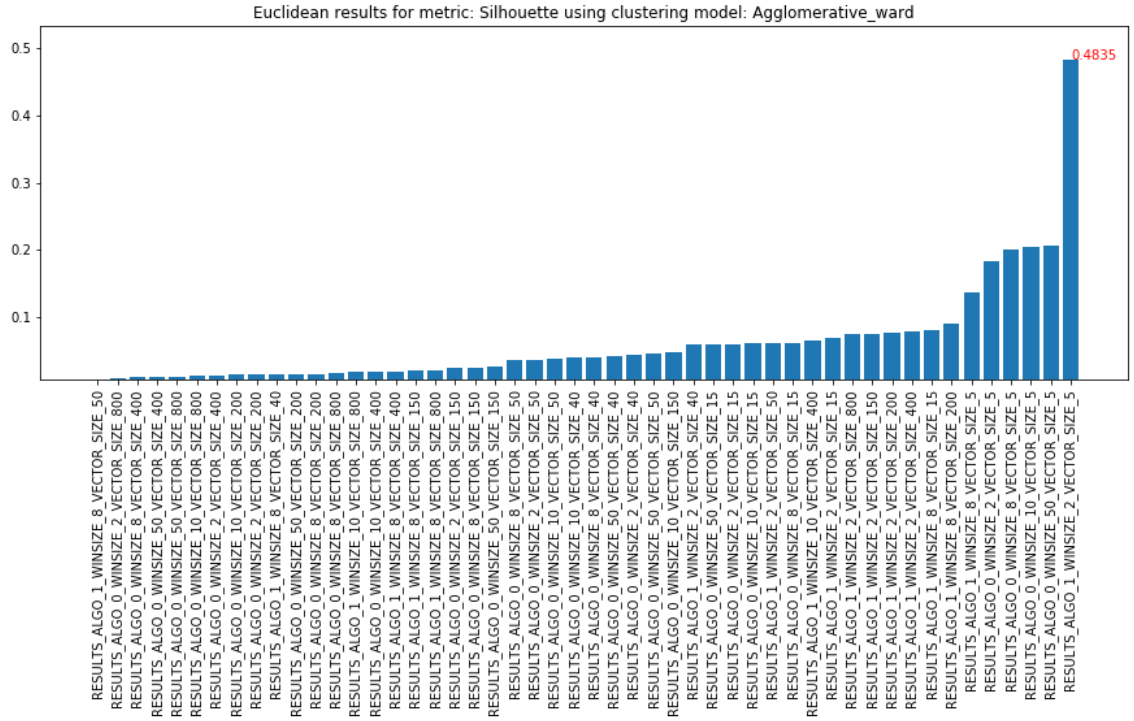

Euclidean results for metric: ARI using clustering model: Agglomerative_min
0.05
0.04
0.03
0.02
0.01
0.00
0.0014
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_40

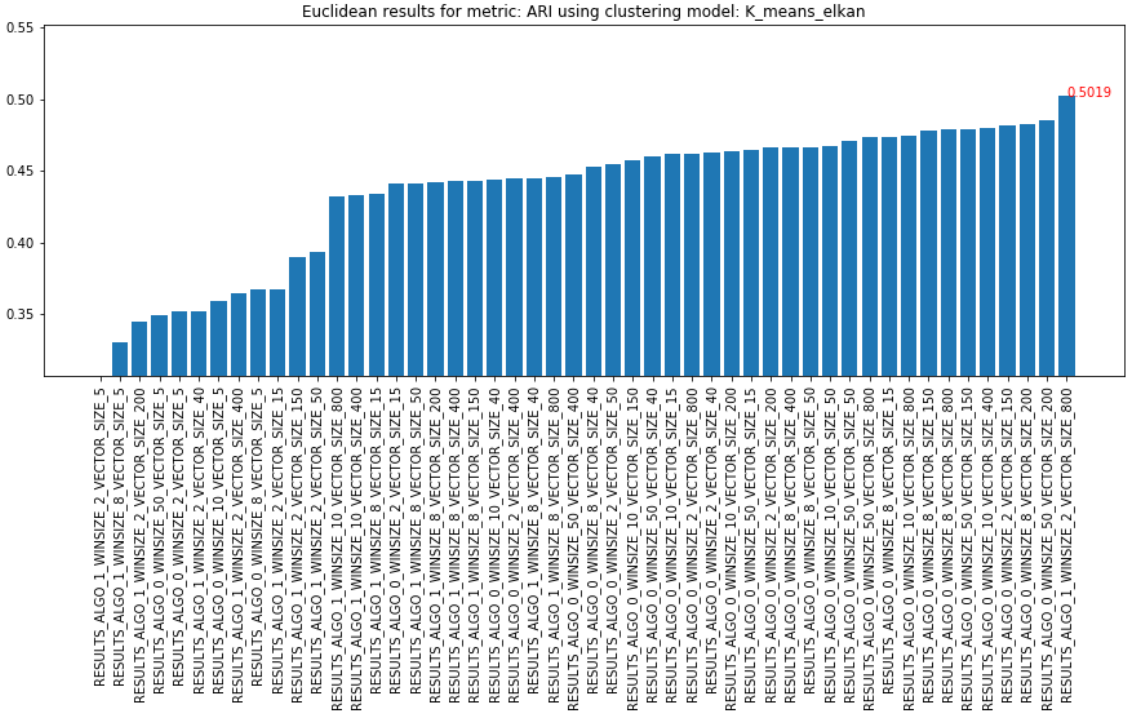

Euclidean results for metric: RI using clustering model: Agglomerative_min
0.22
0.20
0.18
0.16
0.14
0.181
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_200

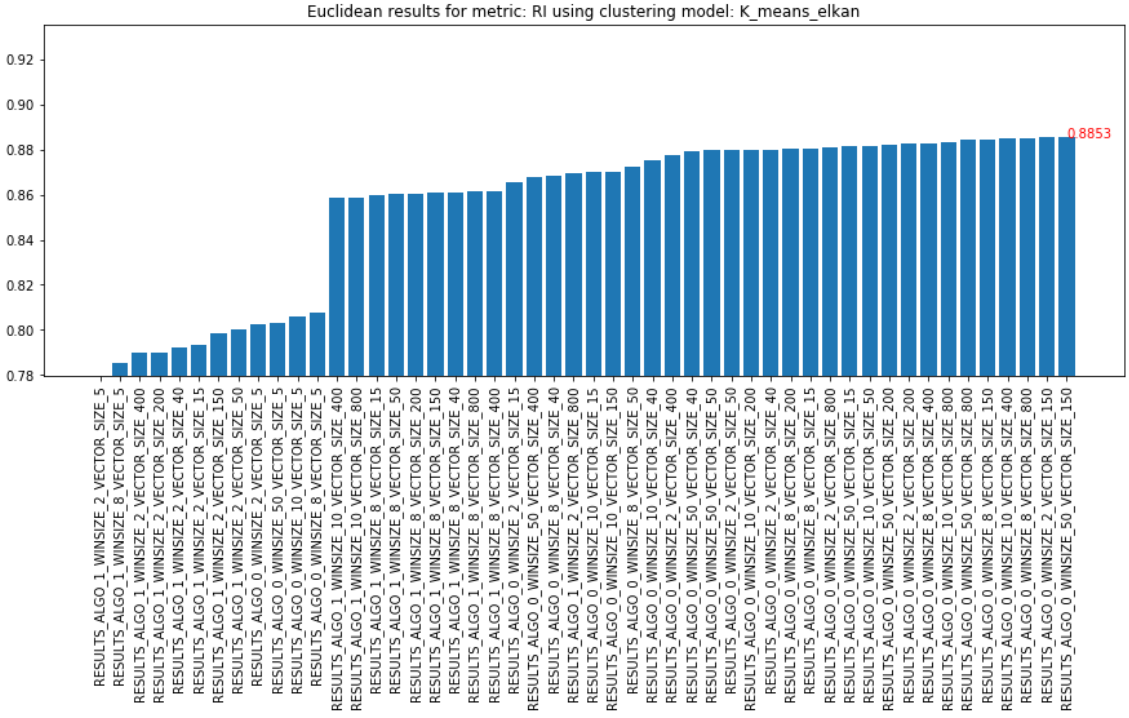

Euclidean results for metric: Silhouette using clustering model: Agglomerative_min
0.4225
0.4
0.3
0.2
0.1
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_5

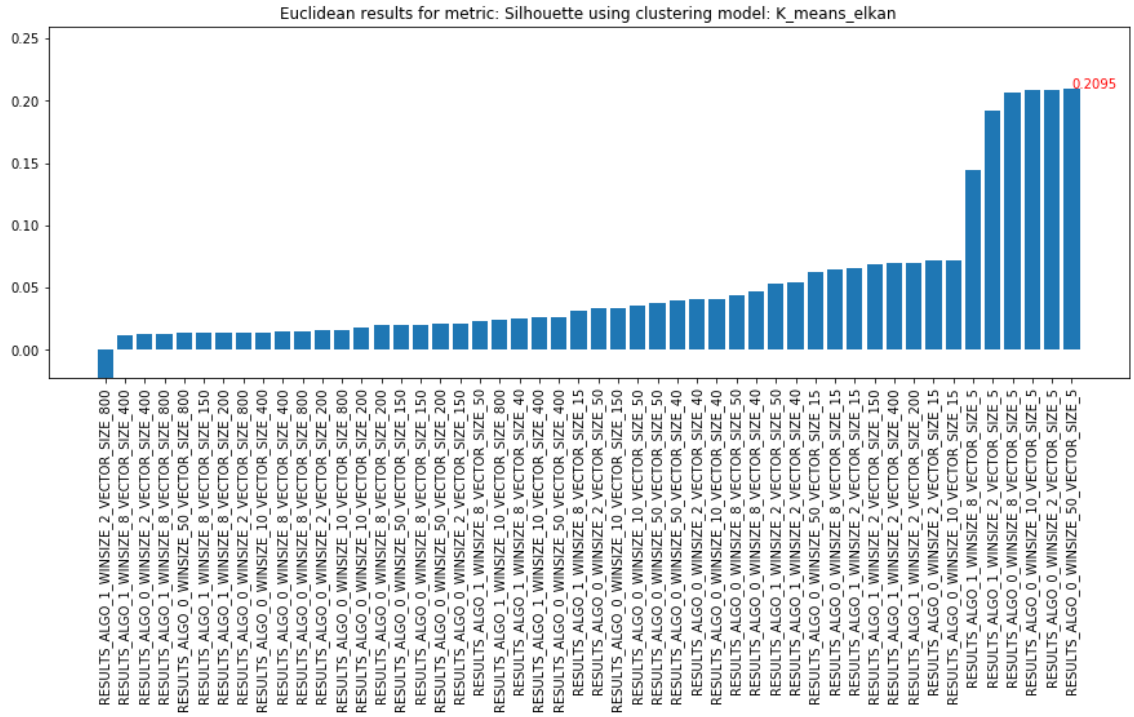

Euclidean results for metric: ARI using clustering model: Agglomerative_ward
0.4716
0.50
0.45
0.40
0.35
0.30
0.25
0.20
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800

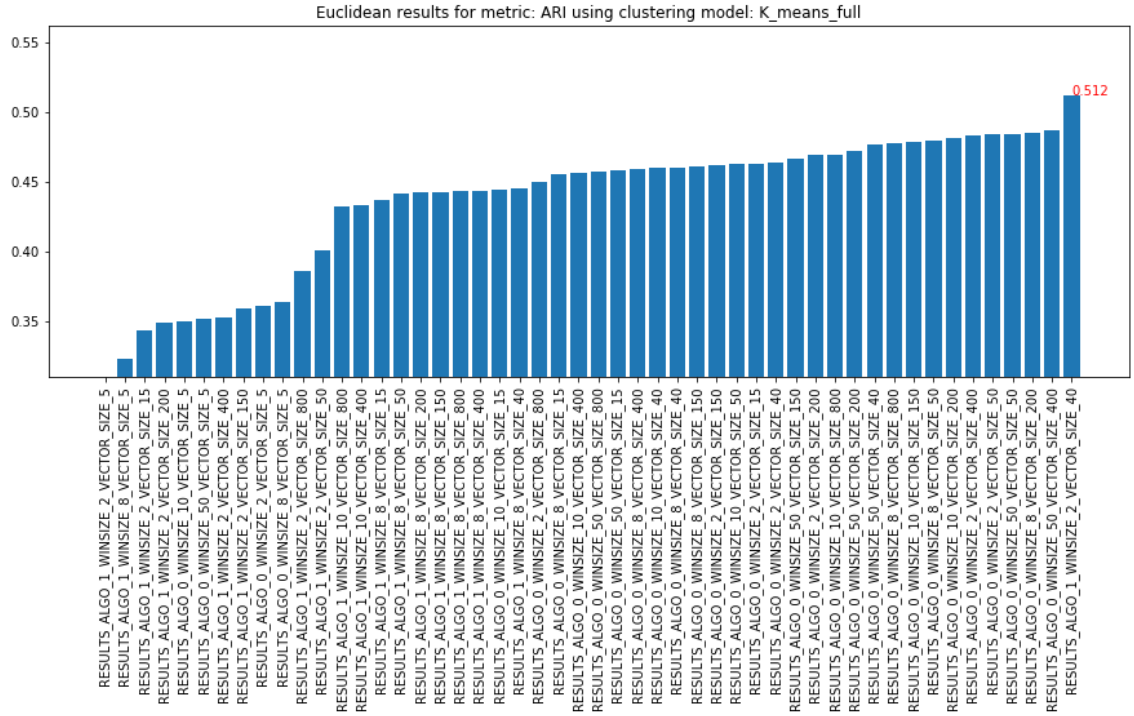

Euclidean results for metric: RI using clustering model: Agglomerative_ward
0.8826
0.90
0.85
0.80
0.75
0.70
0.65
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_200

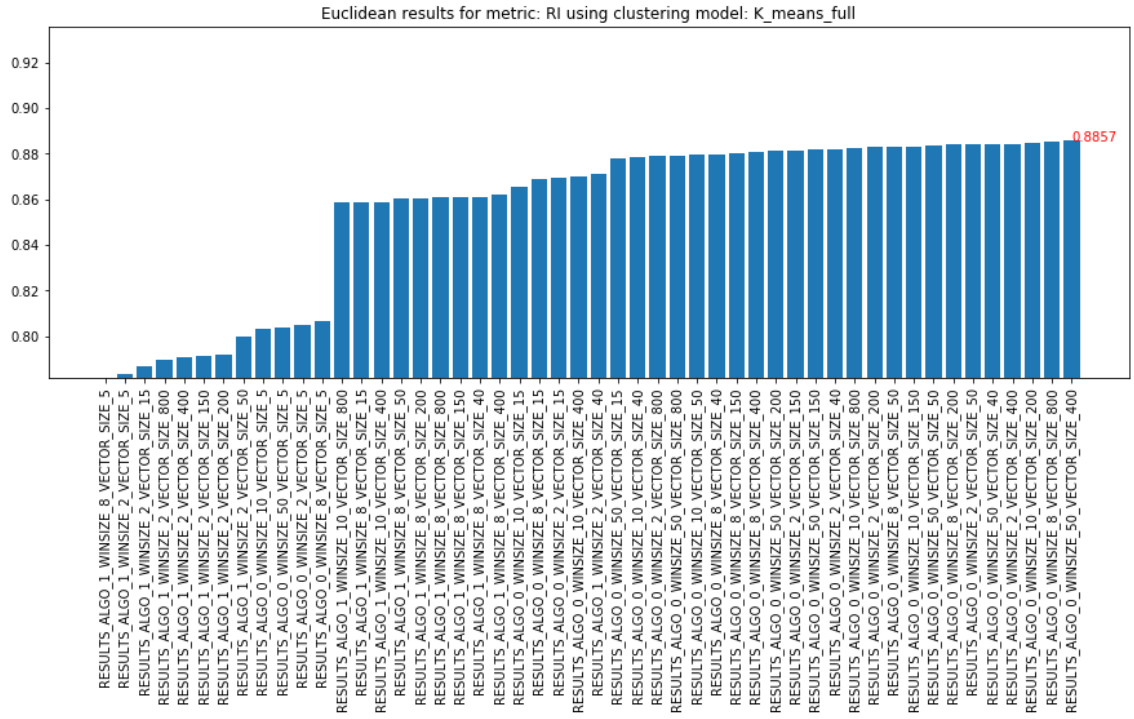

Euclidean results for metric: Silhouette using clustering model: Agglomerative_ward
0.4835
0.5
0.4
0.3
0.2
0.1
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_5

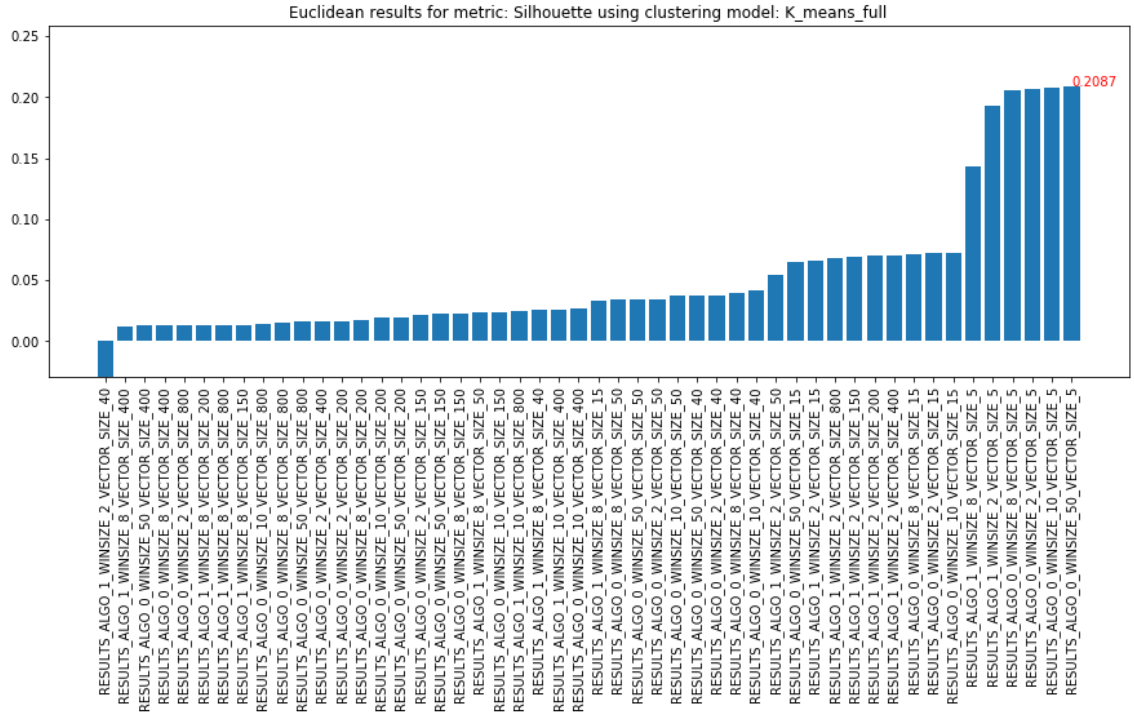

Euclidean results for metric: ARI using clustering model: K_means_elkan
0.55
0.50
0.45
0.40
0.35
0.5019
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800

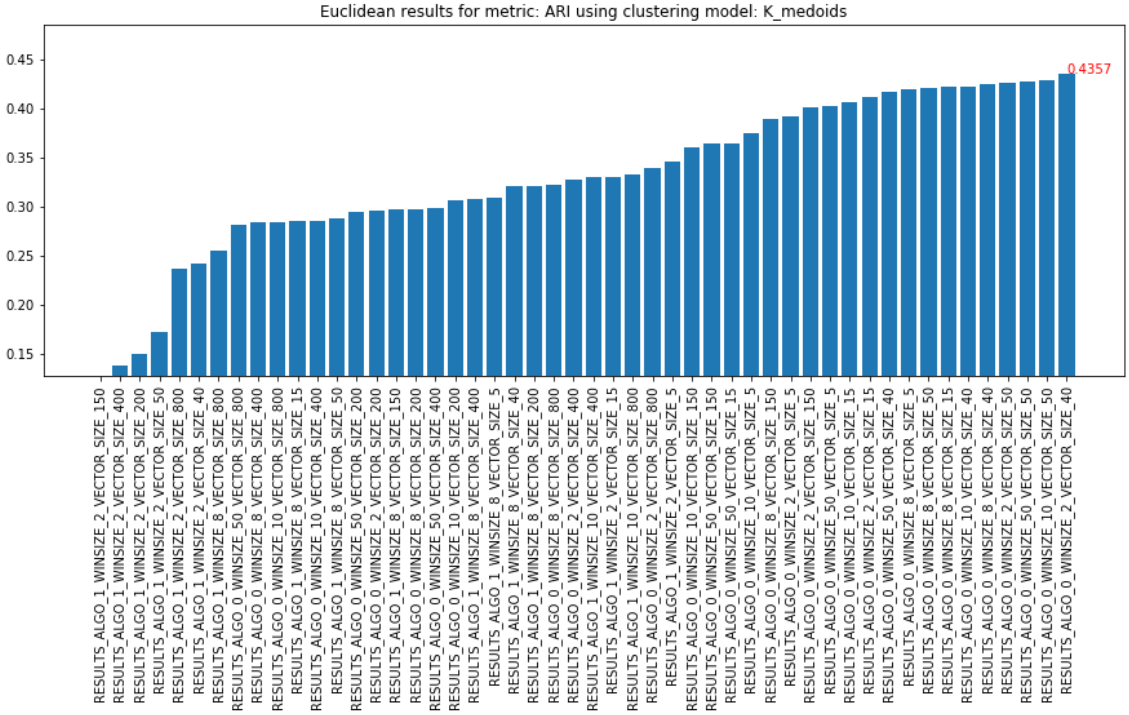

Euclidean results for metric: RI using clustering model: K_means_elkan
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0.8853
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_150

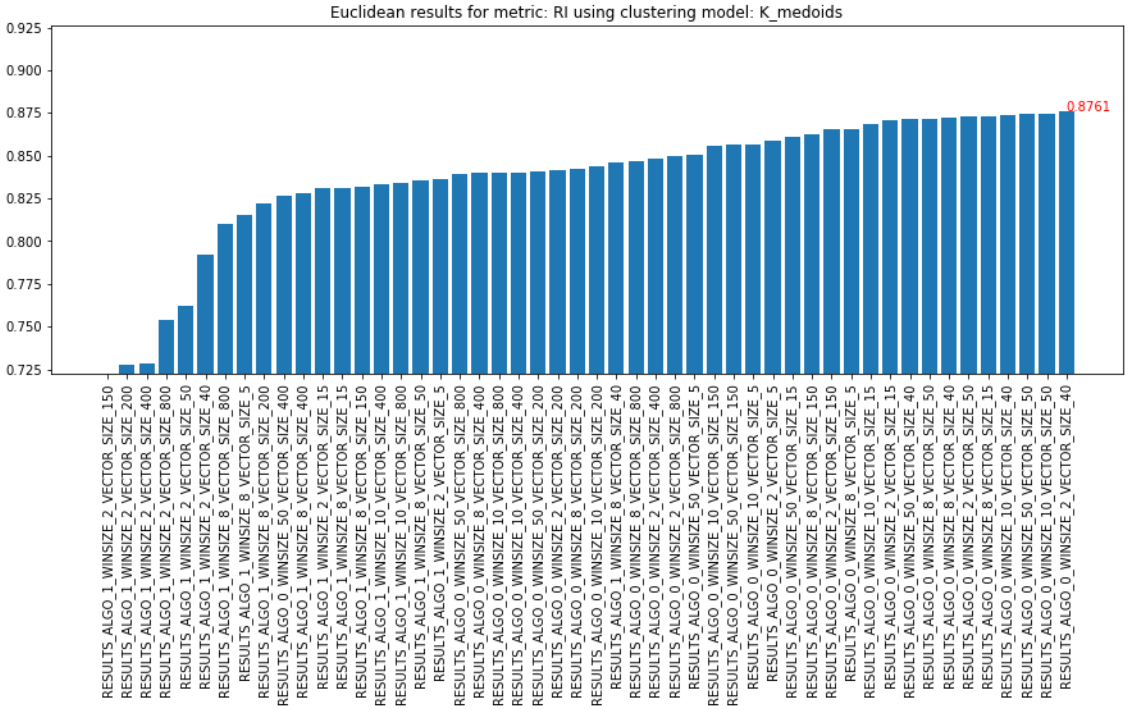

Euclidean results for metric: Silhouette using clustering model: K_means_elkan
0.2095

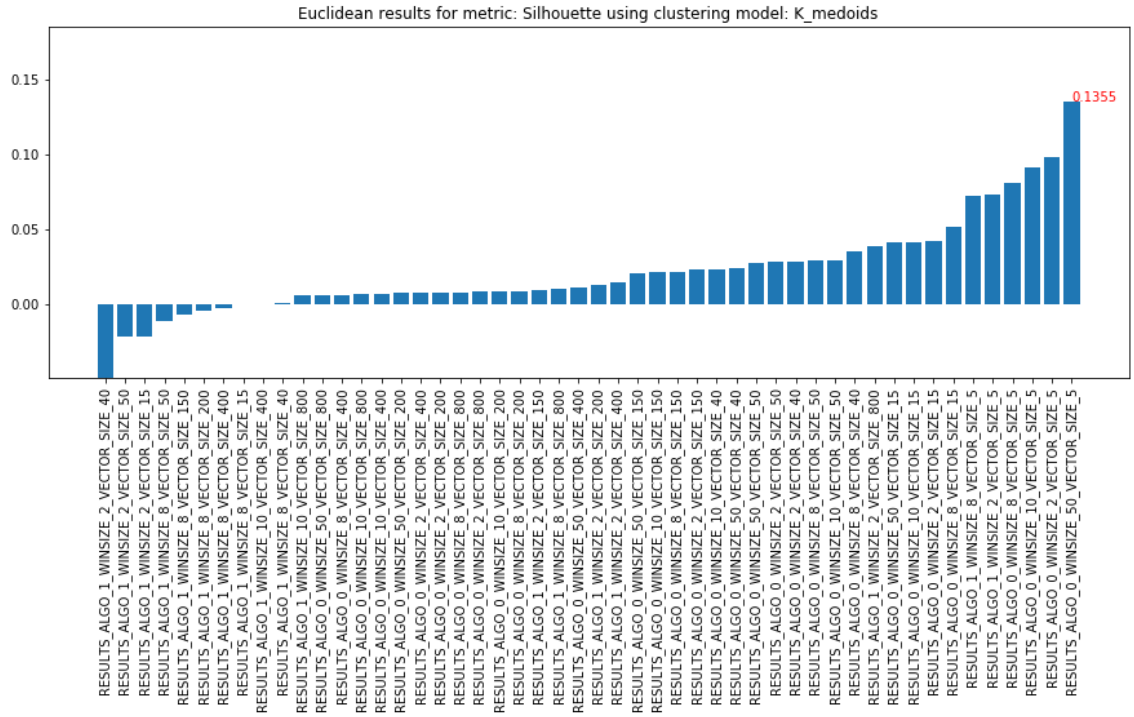

Euclidean results for metric: ARI using clustering model: K_means_full
0.512


Euclidean results for metric: RI using clustering model: K_means_full
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.8857
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_50_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_400


Euclidean results for metric: Silhouette using clustering model: K_means_full
0.25
0.20
0.15
0.10
0.05
0.00
0.2087
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_5


Euclidean results for metric: ARI using clustering model: K_medoids
0.4357


Euclidean results for metric: RI using clustering model: K_medoids
0.8761



Notiamo delle forti variazioni di perfomance per determinate combinazioni di parametri e al variare dell'algoritmo di clustering utlizzato in riferimento a determinate metriche. Un esempio molto chiaro è il parametro lunghezza del vettore di embedding, che col valore nel range 200-800 ha spesso dei valori di silhouette molto bassi con ogni altra configurazione di parametri ed addirittura negativi nella configurazione con agglomerative clustering e linkage minimum per la metrica ARI.

In seguito mostriamo un riepilogo dei risultati migliori pero ogni metrica:

	AdjustedRI
Clustering model	K_means_full
ARI	0.511961
RI	0.871453
Silhouette	-0.0291136
Parameters set	ALGO_1_WINSIZE_2_VECTOR_SIZE_40
	RandomIndex
Clustering model	K_means_full
ARI	0.486845
RI	0.88567
Silhouette	0.0130669
	Silhouette
Parameters set	ALGO_0_WINSIZE_50_VECTOR_SIZE_400
Clustering model	Agglomerative_ward
ARI	0.163625
RI	0.630104
Silhouette	0.483477

Parameters set	ALGO_1_WINSIZE_2_VECTOR_SIZE_5

Tabella 9 Risultati migliori per sig2vec con distanza euclidea

Notiamo che per due casi (ARI e RI) la combinazione formata da algoritmo con grandezza dell'embedding vector di 40 features e ampiezza della finestra 2 o 50 si hanno i risultati migliori seppur con un bassissimo valore di silhouette. La figura successiva mostra il grafico dell'evoluzione della curse of dimensionality al crescere del valore di grandezza del vettore di embedding. Il grafico sembra confermare la nostra supposizione circa la migliore lunghezza di vettore di embedding in quanto mostra il ginocchio in prossimità del range 15-40.

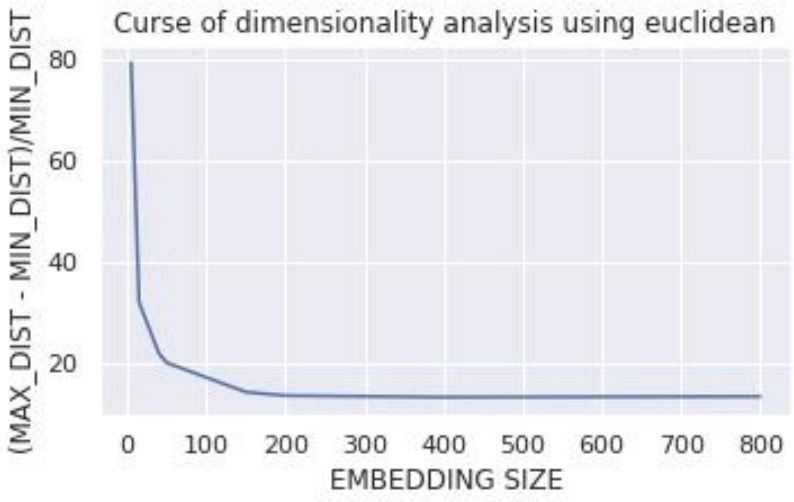


Figura 37 Curse of dimensionality Sig2vec con distanza euclidea

La figura successiva mostra inoltre l'andamento del valore di SSE al crescere di K, numero dei cluster per la configurazione Algo 1, winsize 5 vectorsize 40:

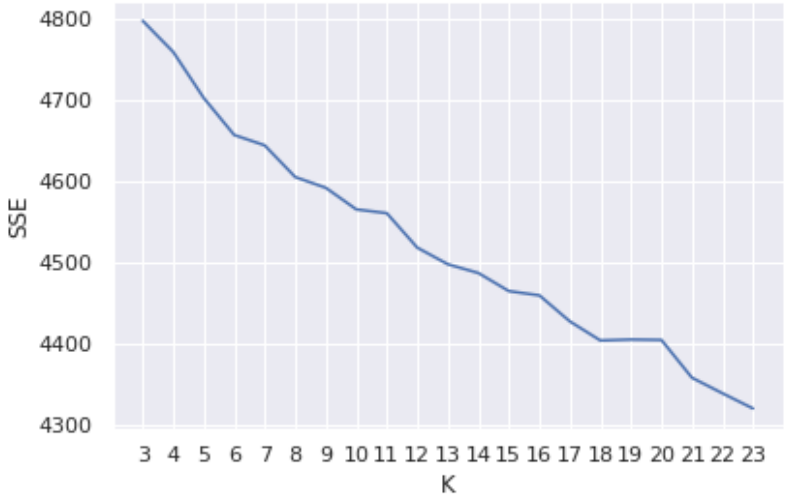


Figura 38 K-selection Sig2vec con distanza euclidea

Passiamo pertanto a mostrare i risultati ottenuti con l'algoritmo di DBSCAN utilizzando la combinazione sovracitata.

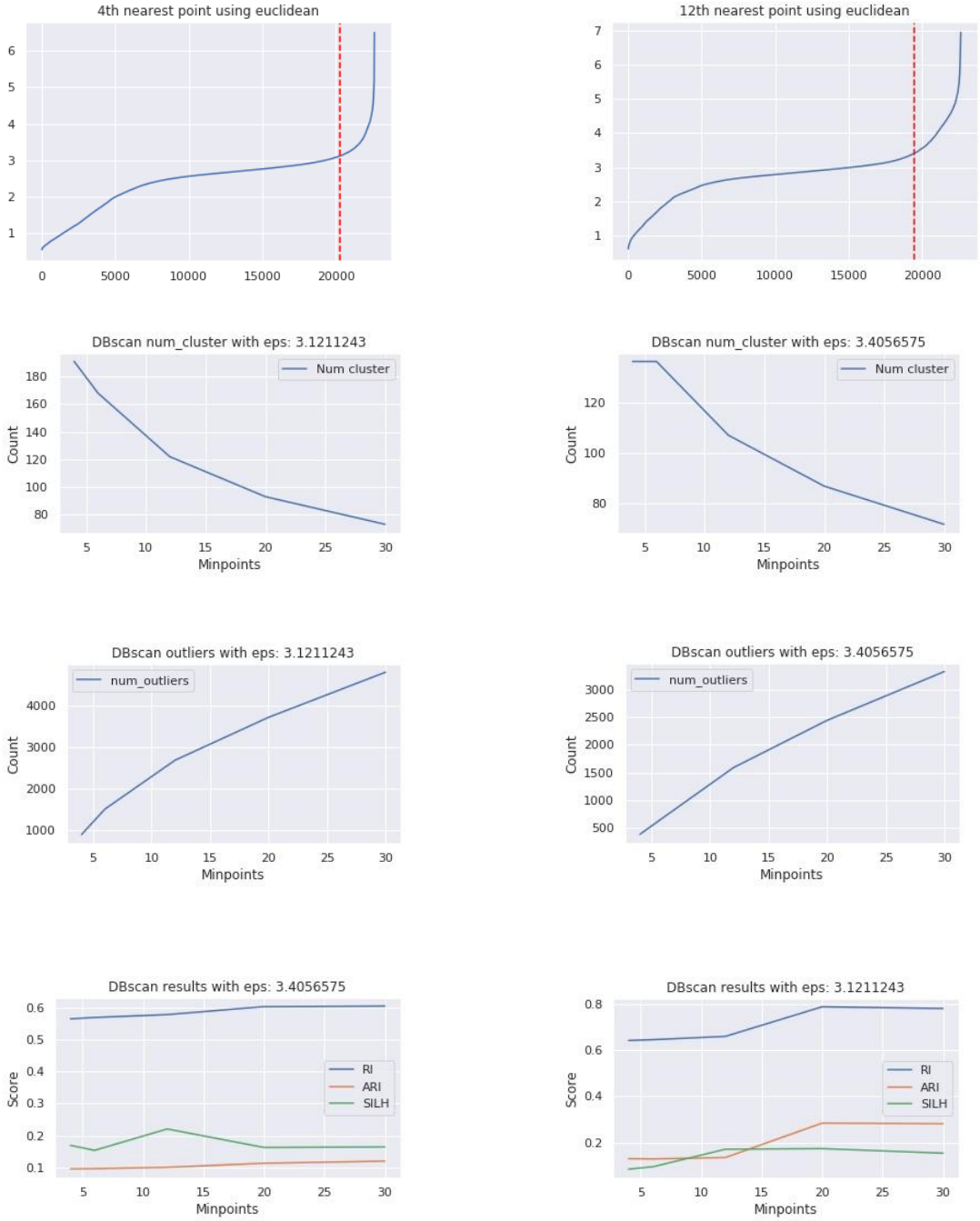


Figura 39 Grafici risultati DBSCAN al variare dei parametri per sig2vec con distanza euclidea

Notiamo come in questa configurazione anche il DBscan riesca a raggiungere buoni risultati senza mai però convergere verso il corretto numero di cluster da formare, ovvero 9, ma rimanendo ben sopra quel valore. Riportiamo per completezza i migliori risultati raggiunti dal DBscan rispetto ad ogni metrica su tutte le possibili combinazioni di parametri.

Parameters set	*ARI*	*RI*	*SILHOUETTE*	*NUM_CLUSTER*	*NUM_OUTLIERS*	
eps:3.14 minpoints: 20	**0.283**	0.788	0.175	96	3709	ALGO_0_WINSIZE_50_VECTOR_SIZE_40
eps:3.14 minpoint:20	0.283	**0.788**	0.175	96	3721	ALGO_0_WINSIZE_8_VECTOR_SIZE_40
eps:1.059 minpoints:6	0.159	0.623	**0.488**	174	4241	ALGO_1_WINSIZE_2_VECTOR_SIZE_5

Tabella 10 Risultati migliori per DBSCAN per sig2vec con distanza euclidea

Coseno

Continuiamo con la presentazione degli esperimenti e dei risultati ottenuti tramite la distanza del coseno. In questo secondo scenario abbiamo utilizzato la metrica del coseno sia per calcolare le distanze fra i features vectors durante il training degli algoritmi di cluster, sia per la creazione dei grafici (tranne per il K-selection) e la computazione della metrica silhouette. In questo caso non abbiamo potuto utilizzare gli algoritmi di KMeans (in quanto non compatibile con metriche diverse da quella euclidea) e agglomerative con ward linkage. Per questi motivi abbiamo sostituito il KMeans con l'algoritmo Kmedoids che invece è più flessibile con le metriche utilizzabili).

Mostriamo come prima i grafici a barre per ogni metrica ed ogni combinazione di parametri.

Notiamo come prima delle migliori perfomance con gli algoritmi KMedoids per "piccole" lunghezze dei features vectori, mentre vettori con 200 features sembrano performare meglio unicamente con agglomerative minumum linkage.

Mostriamo nella tabella seguente i migliori risultati per ogni metrica.

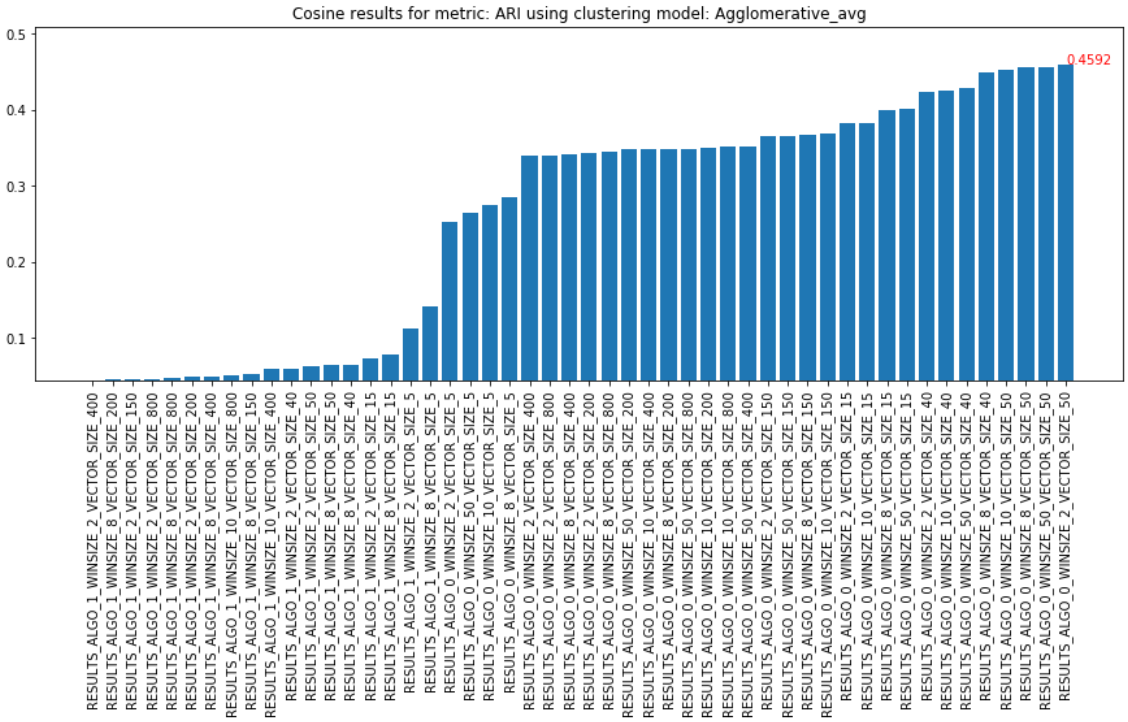
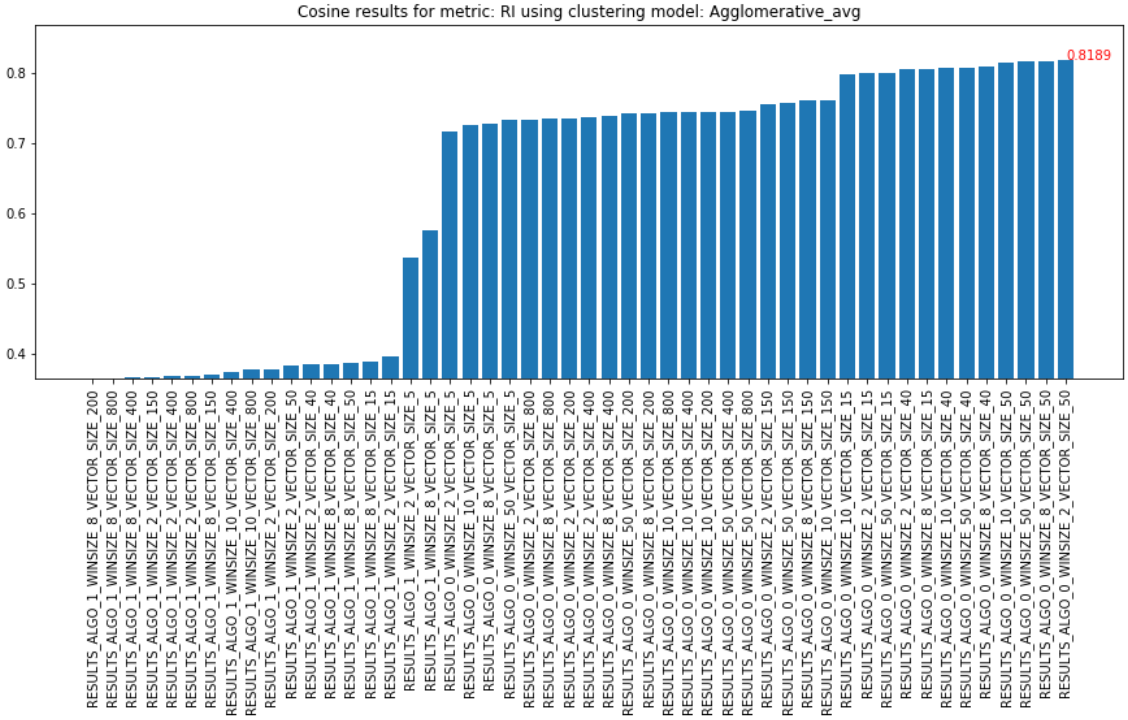
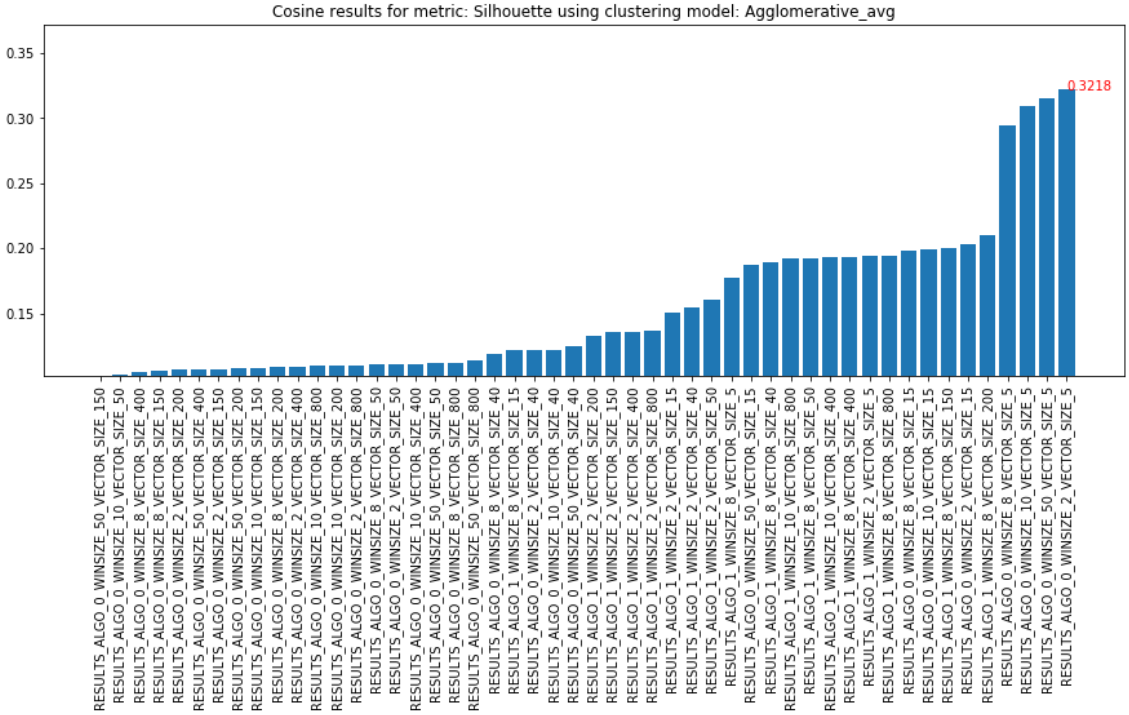
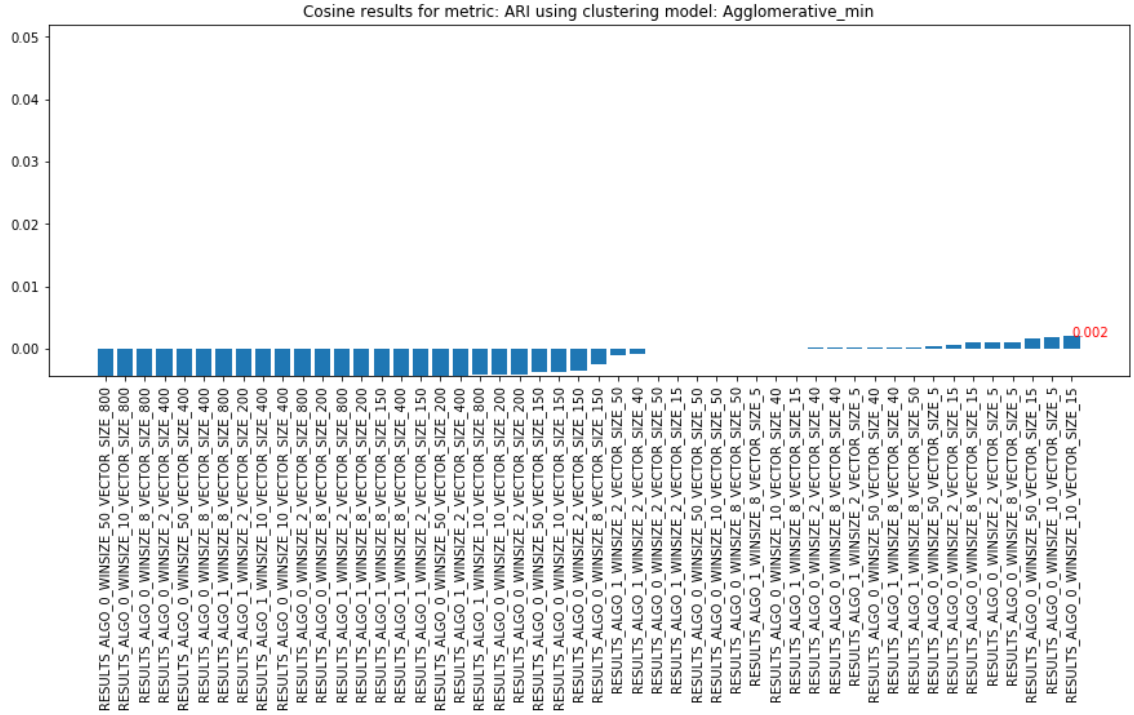
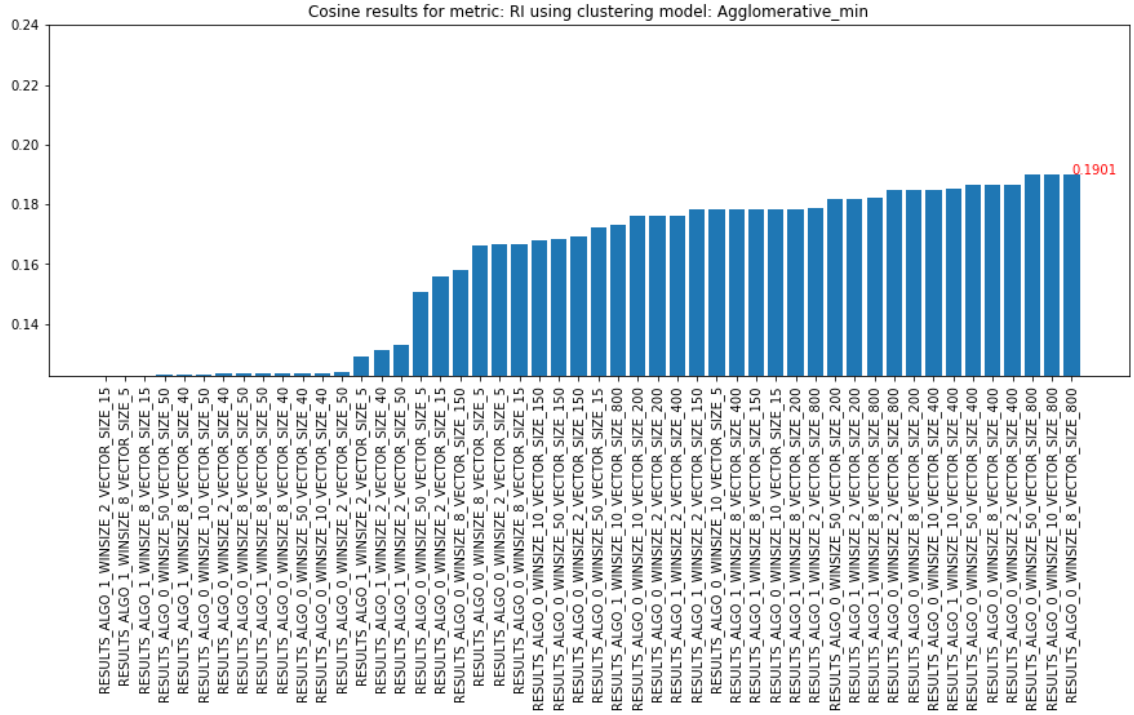


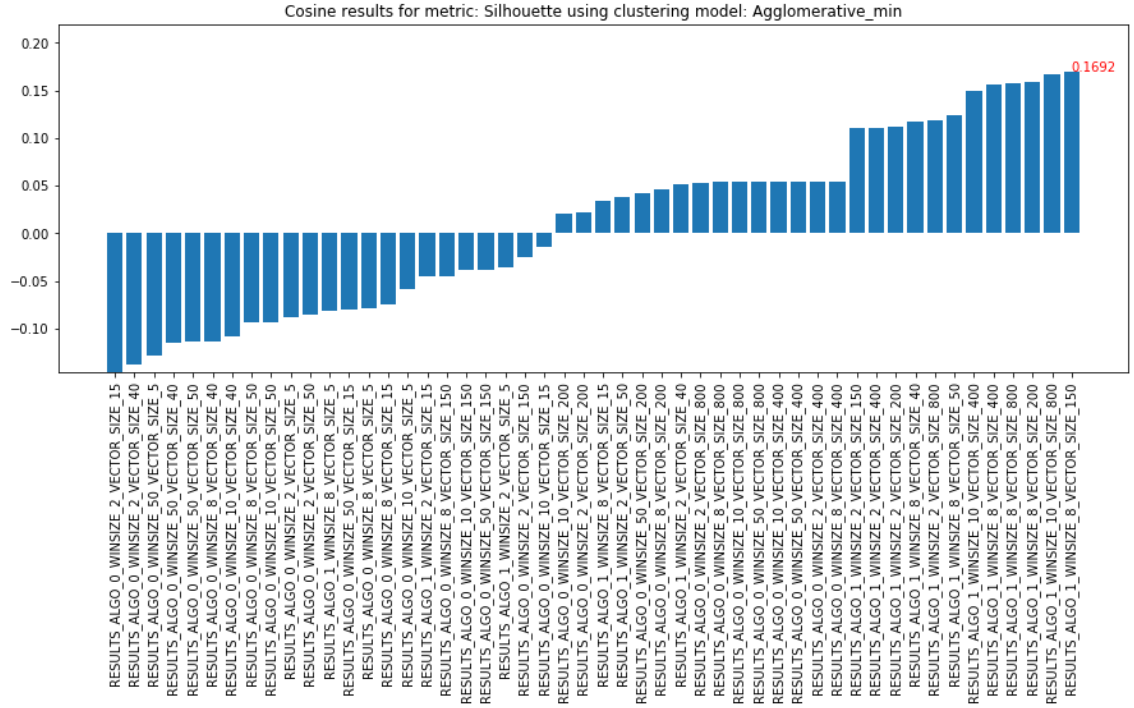
Figura 40 Grafici a barre sig2vec con distanza del coseno al variare delle combinazioni fra iperparametri

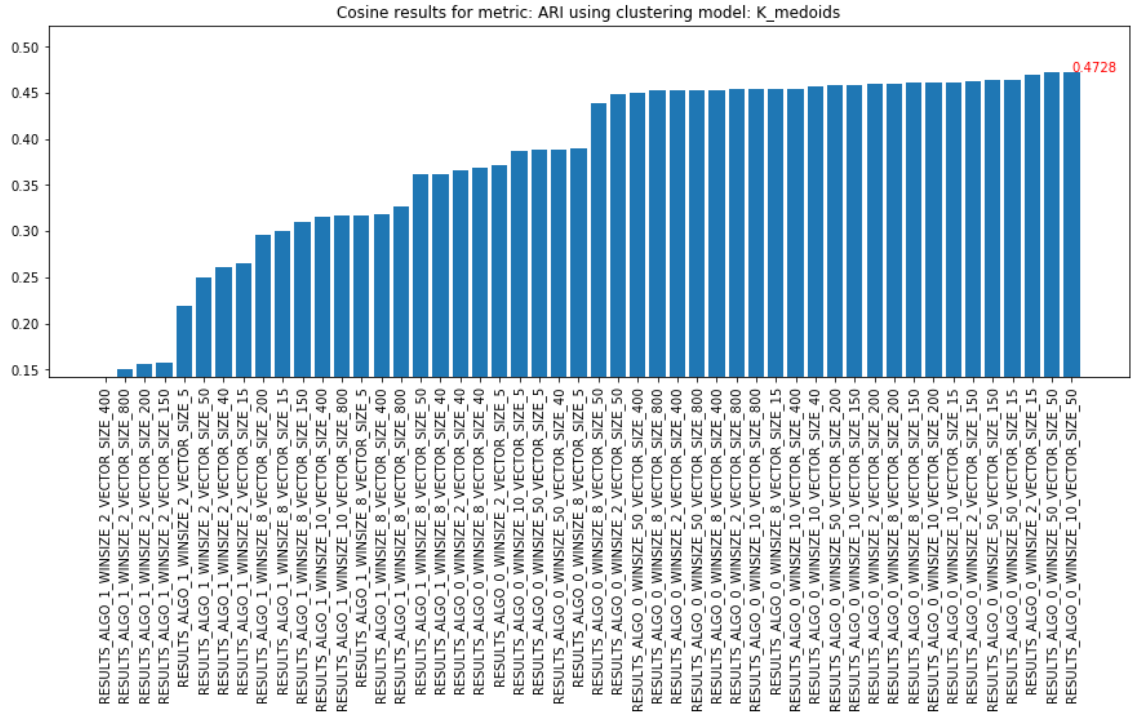

Cosine results for metric: RI using clustering model: Agglomerative_avg
0.8189
0.8
0.7
0.6
0.5
0.4
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50

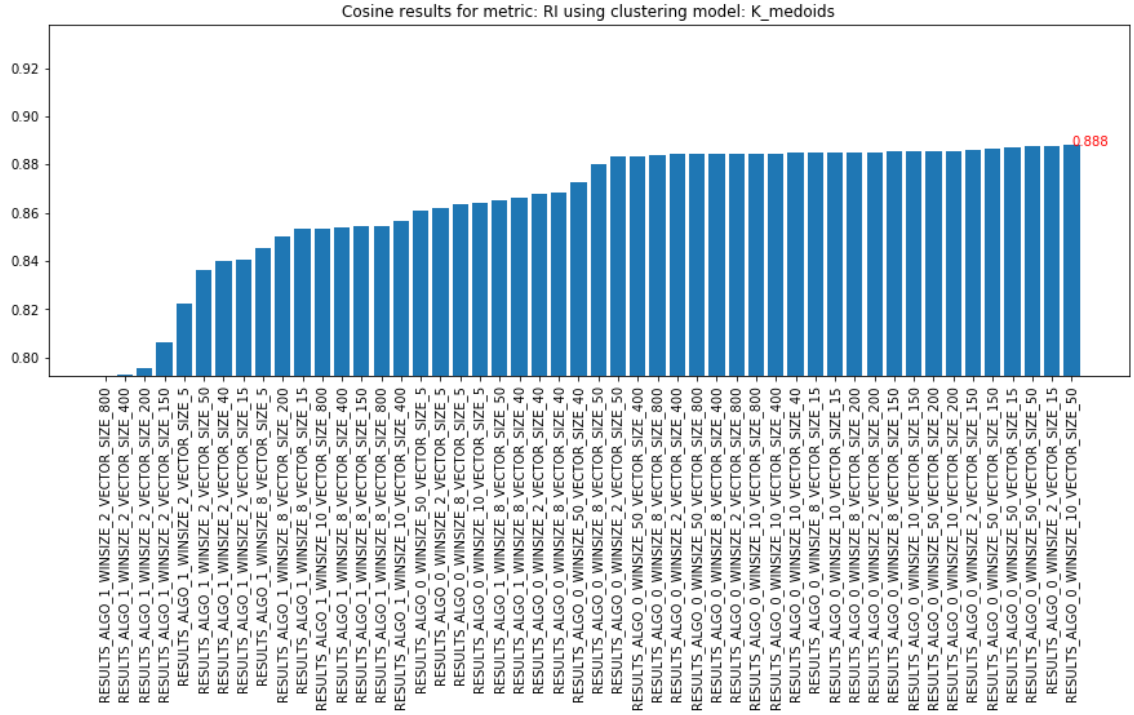

Cosine results for metric: Silhouette using clustering model: Agglomerative_avg
0.3218
0.35
0.30
0.25
0.20
0.15
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_5

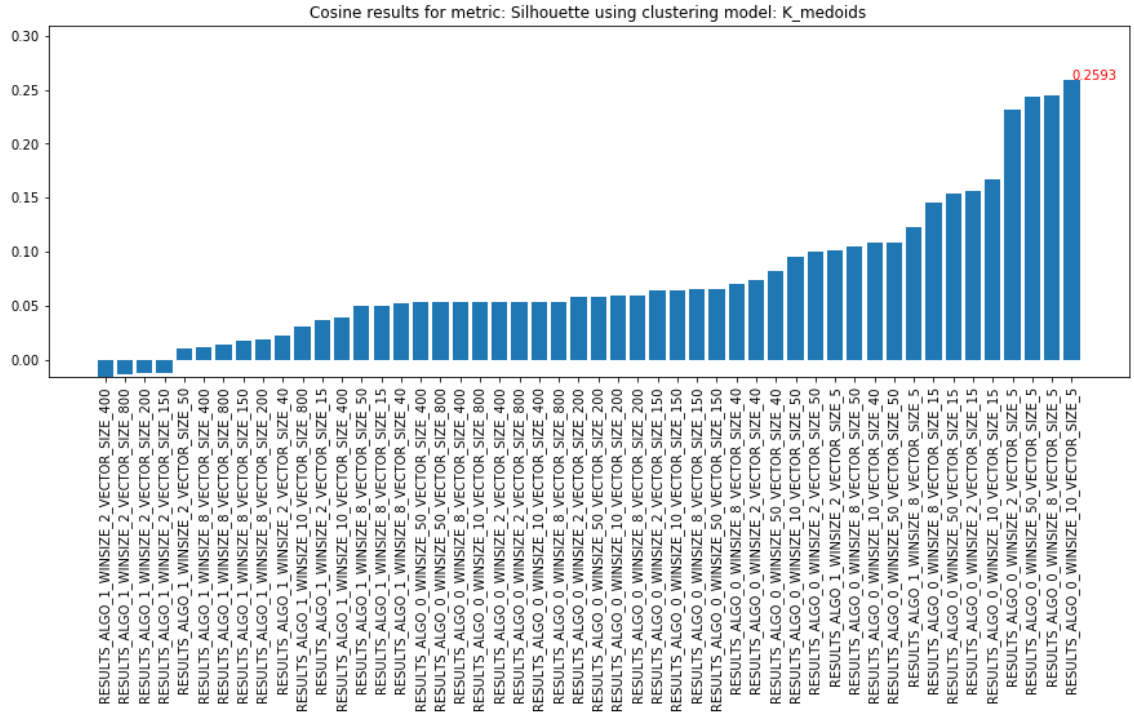

Cosine results for metric: ARI using clustering model: Agglomerative_min
0.05
0.04
0.03
0.02
0.01
0.00
0.002
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_800


Cosine results for metric: RI using clustering model: Agglomerative_min
0.24
0.22
0.20
0.18
0.16
0.14
0.1901
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_40
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_40
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_50
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_5
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_15
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_200
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_5
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_150
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_15
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_2_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_8_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_200
RESULTS_ALGO_1_WINSIZE_10_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_2_VECTOR_SIZE_400
RESULTS_ALGO_0_WINSIZE_50_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_10_VECTOR_SIZE_800
RESULTS_ALGO_0_WINSIZE_8_VECTOR_SIZE_800


Cosine results for metric: Silhouette using clustering model: Agglomerative_min
0.1692


Cosine results for metric: ARI using clustering model: K_medoids
0.4728





	AdjustedRI
Clustering model	K_medoids
ARI	0.47285
RI	0.887955
Silhouette	0.0955702
Parameters set	ALGO_0_WINSIZE_10_VECTOR_SIZE_50

	RandomIndex
Clustering model	K_medoids
ARI	0.47285
RI	0.887955
Silhouette	0.0955702
Parameters set	ALGO_0_WINSIZE_10_VECTOR_SIZE_50
	Silhouette
Clustering model	Agglomerative_avg
ARI	0.253147
RI	0.716212
Silhouette	0.321833
Parameters set	ALGO_0_WINSIZE_2_VECTOR_SIZE_5

Tabella 11 Risultati migliori per sig2vec con distanza del coseno

Anche in questo caso una combinazione risulta performare bene per due metriche, Adjusted random index e random index. Pertanto procediamo a mostrare i risultati per il DBSCAN con la configurazione algoritmo doc2vec 0, ampiezza della finestra delle parole 10 e lunghezza del vettore di embedding 50 features.

Come ulteriore controprova di questa scelta mostriamo anche in questo caso il grafico della curse of dimensionality utilizzando la distanza del coseno fra le features e il grafico dell'andamento del residual sum of squared error.

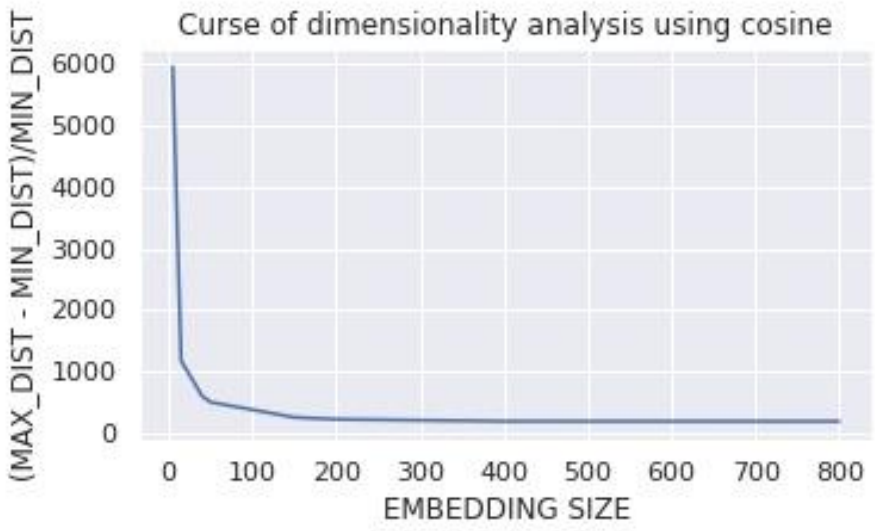


Figura 41 Curse of dimensionality Sig2vec con distanza del coseno

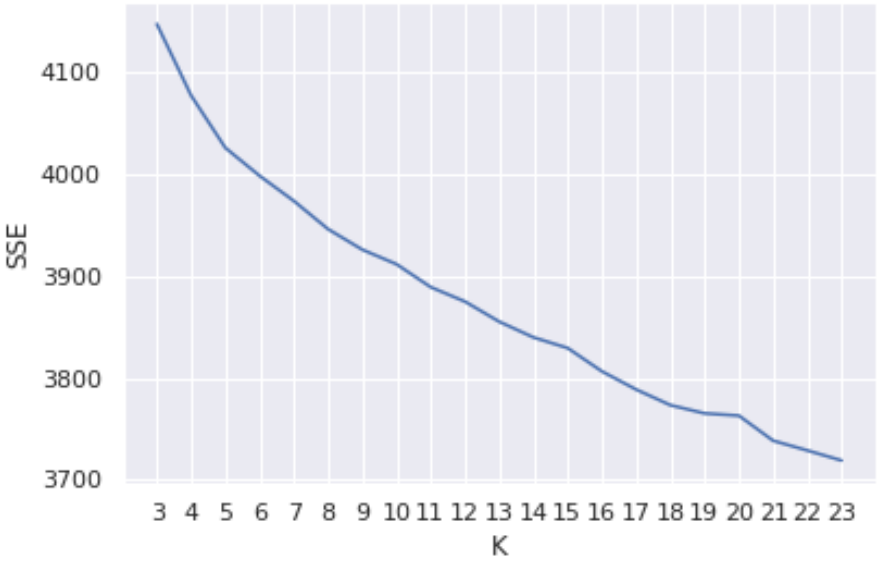
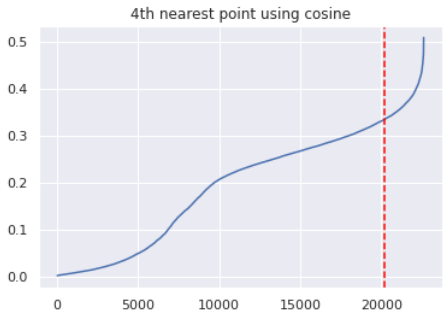
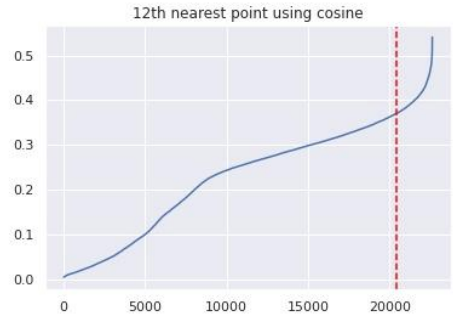
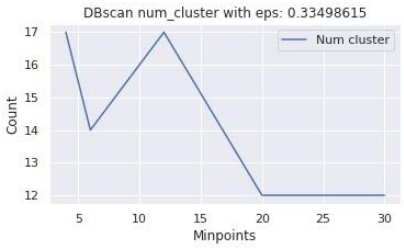
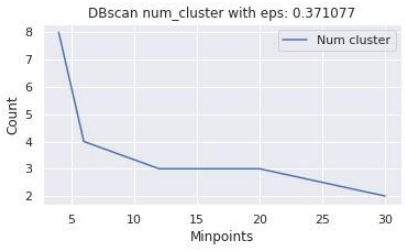


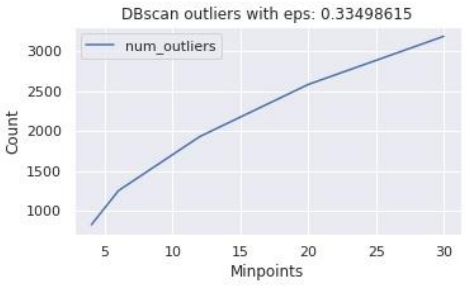
Figura 42 K-selection Sig2vec con distanza del coseno

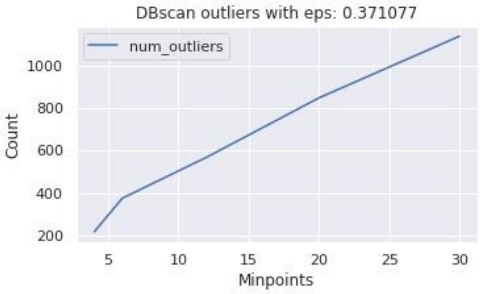


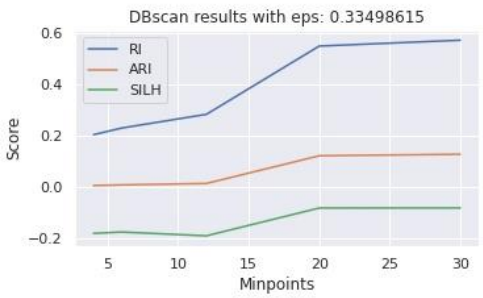


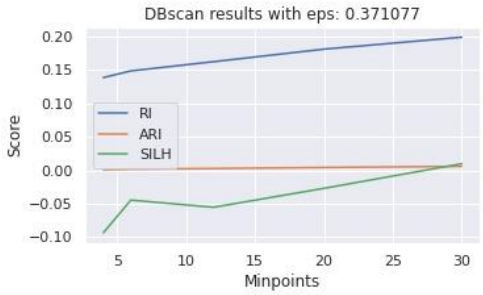












Notiamo che anche in questo caso, seppur DBSCAN riesca a raggiungere risultati discretamente positivi non riesce comunque a generare il corretto numero di cluster del dataset.

Proseguiamo mostrando i migliori risultati per il DBSCAN.

Parameters set	*ARI*	*RI*	*SILHOUETTE*	*NUM CLUSTER*	*NUM OUTLIERS*	
eps:0.13768624 minpoints: 20	0.3213	0.8416	0.0310	121	4887	ALGO_0_WINSIZE_50_VECTOR_SIZE_15
eps:0.1376 minpoints: 20	0.3213	0.8416	0.0310	121	4887	ALGO_0_WINSIZE_50_VECTOR_SIZE_15
eps:0.0676 minpoints: 30	0.2988	0.7998	0.309	85	3148	ALGO_0_WINSIZE_2_VECTOR_SIZE_5

Tabella 12 Risultati migliori per DBSCAN per sig2vec con distanza del coseno

Riassumendo, i risultati ottenuti mostrano che in entrambi i casi analizzati vettori di embedding relativamente corti permettono di ottenere migliori risultati. In entrambi i casi, una finestra di grandezza compresa tra 2 e 50 e l'algoritmo sembra rendere al meglio sia con algoritmi partizionali. DBscan al contrario in pochi casi è in grado di generare il corretto numero di cluster ottenendo tuttavia cattive performance. In conclusione abbiamo ottenuto un punteggio massimo compreso tra 0.48 e 0.51 per quanta riguarda l'adjusted random index, 0.89 per il random index e 0.14 per la silhouette con le configurazioni analizzate.

Convolutional neural network autoencoder

Procediamo adesso con la presentazione dei dati ottenuti tramite gli autoencoder convoluzionali.

Come presentato in precedenza i parametri che abbiamo ottimizzato sono la lunghezza del vettore di embedding e l'applicazione o meno di normalizzazione dei dati tramite tecnica min-max e standardizzazione. Inoltre abbiamo testato diverse architetture di reti neurali sia relativamente all'organizzazione degli strati convoluzionali (inception, residual etc.) sia sul numero di parametri liberi, grandezza dei filtri e stride degli strati convoluzionali. Abbiamo infine studiato due approcci nell'utilizzo dei vettori di embedding negli algoritmi di clustering, uno che prevede la

concatenazione dei vettori, l'altro un re-embedding utilizzando una ulteriore rete neurale di soli fully connected layers.

Mostriamo quindi i risultati ottenuti come nella sezione precedente.

Euclidean

Vettori concatenati

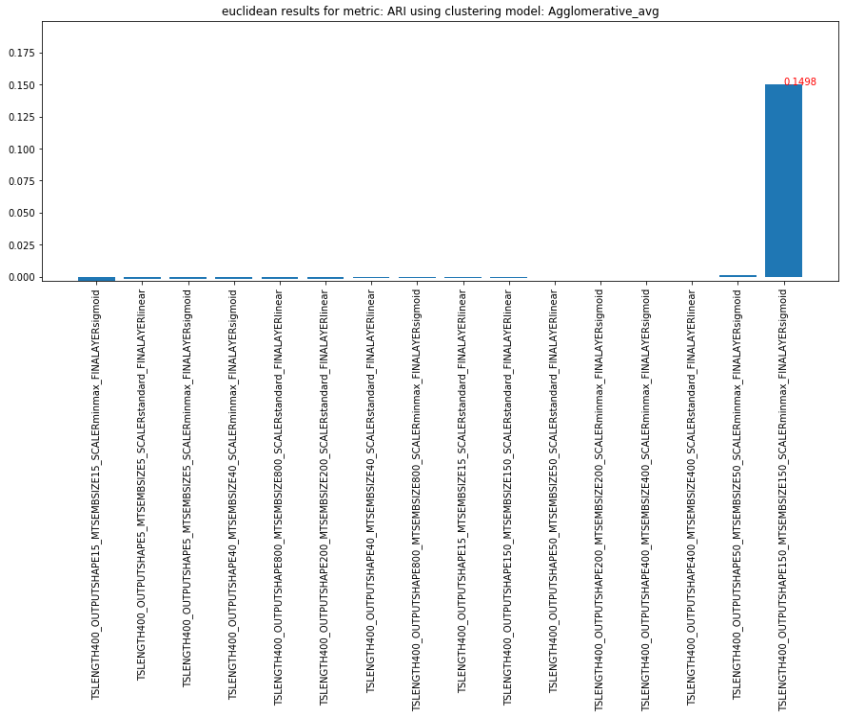
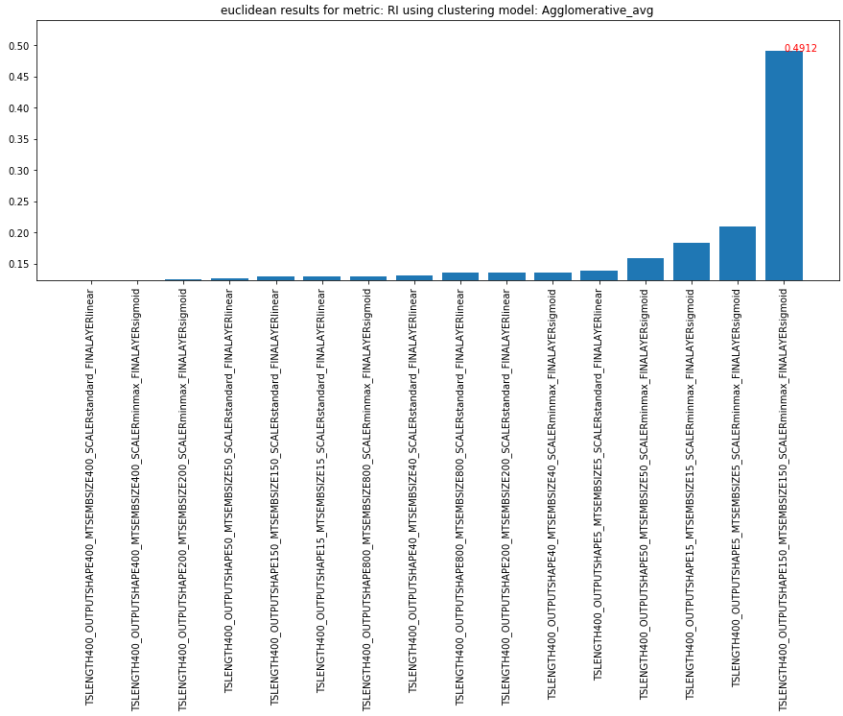
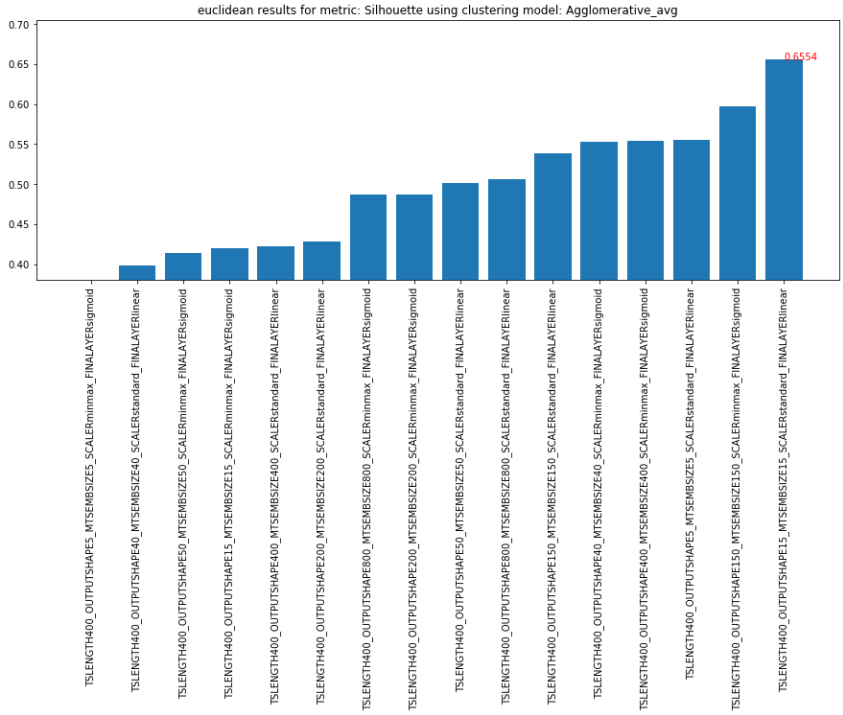
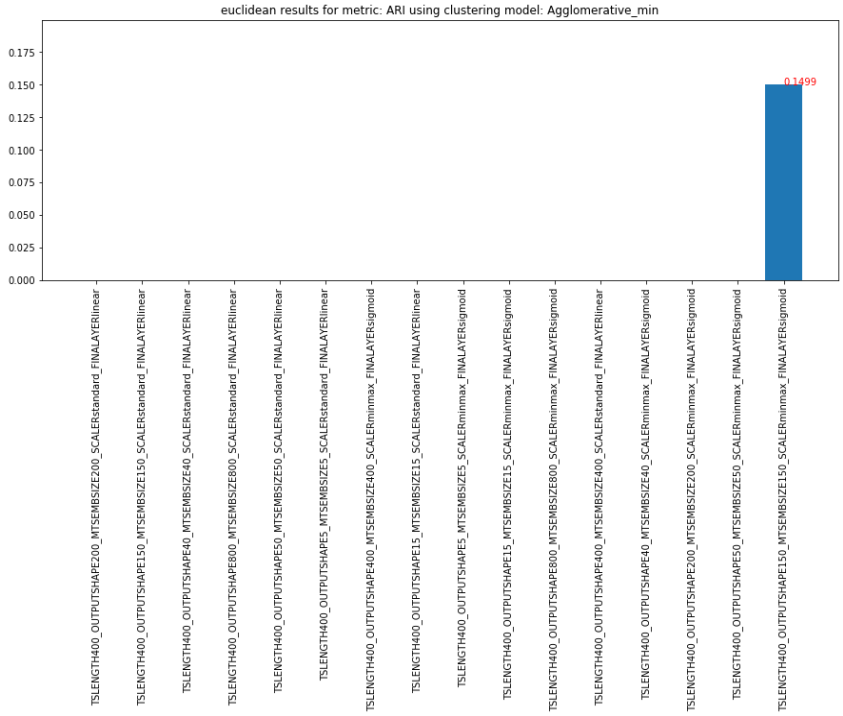
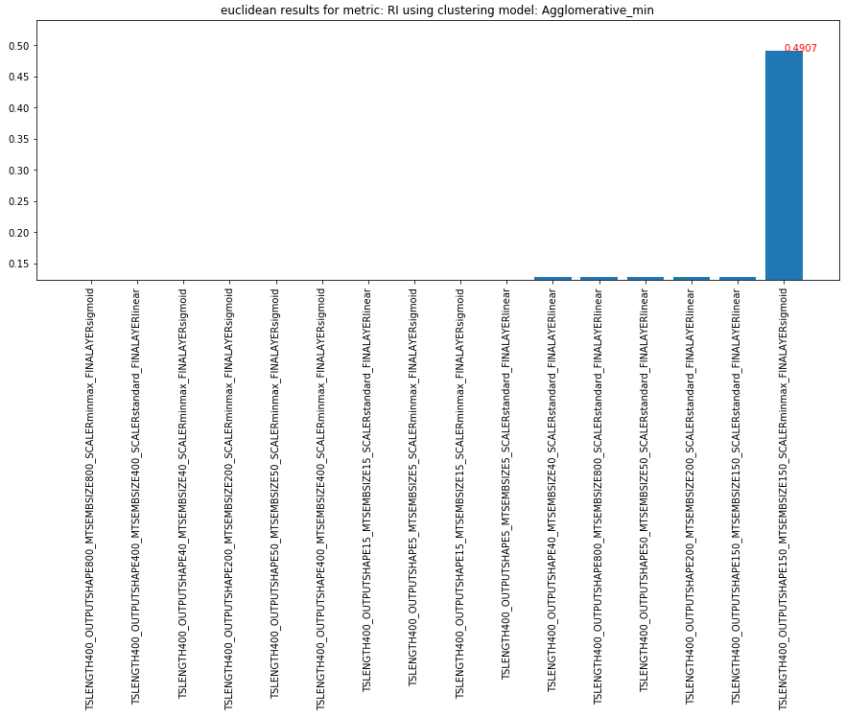


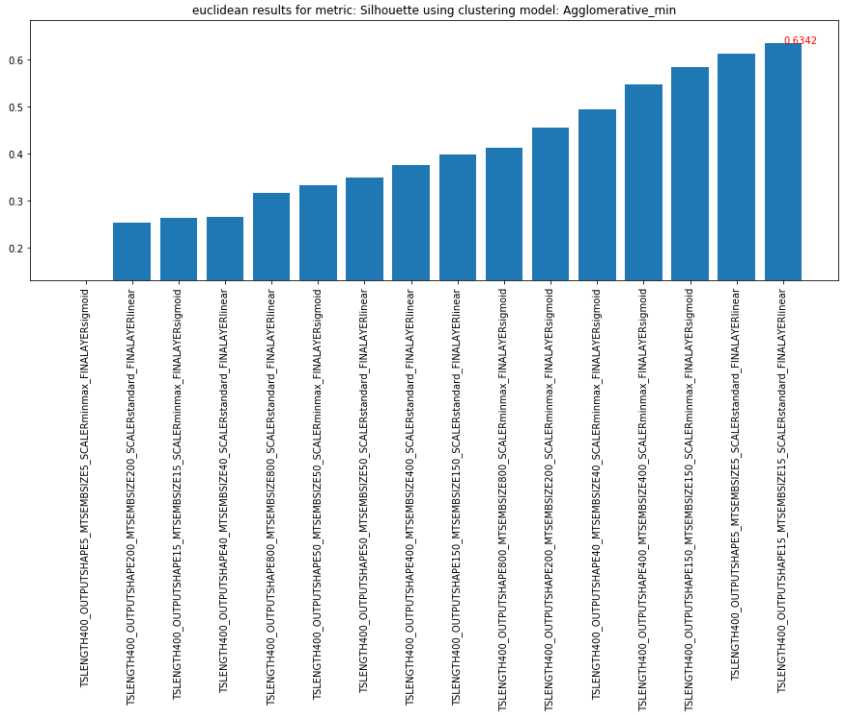
Figura 43 Grafici a barre dei risultati ottenuti con l'approccio CNN al variare delle combinazioni di iperparametri e distanze utilizzate

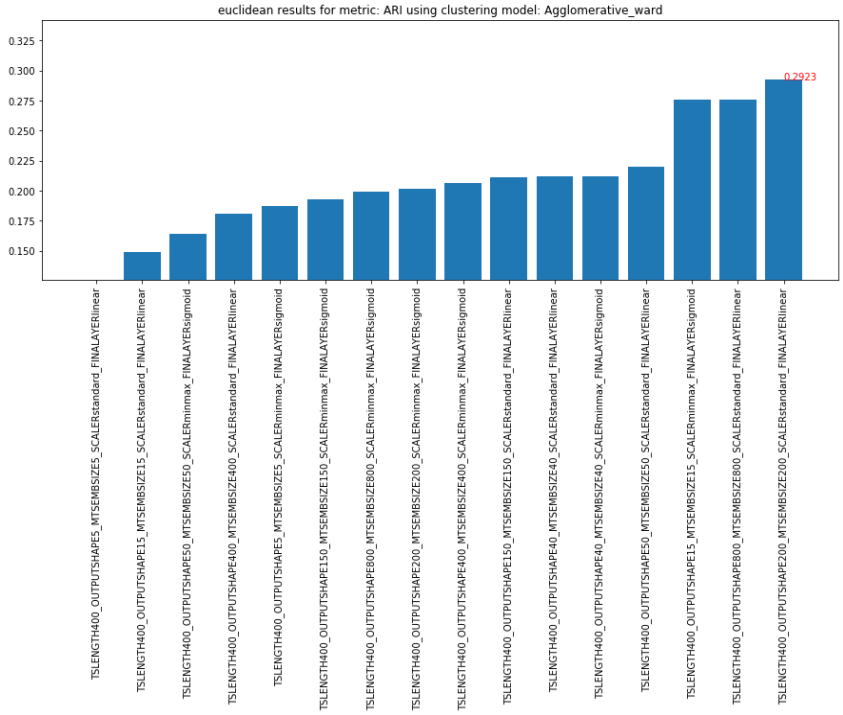

euclidean results for metric: RI using clustering model: Agglomerative_avg
0.4912
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid

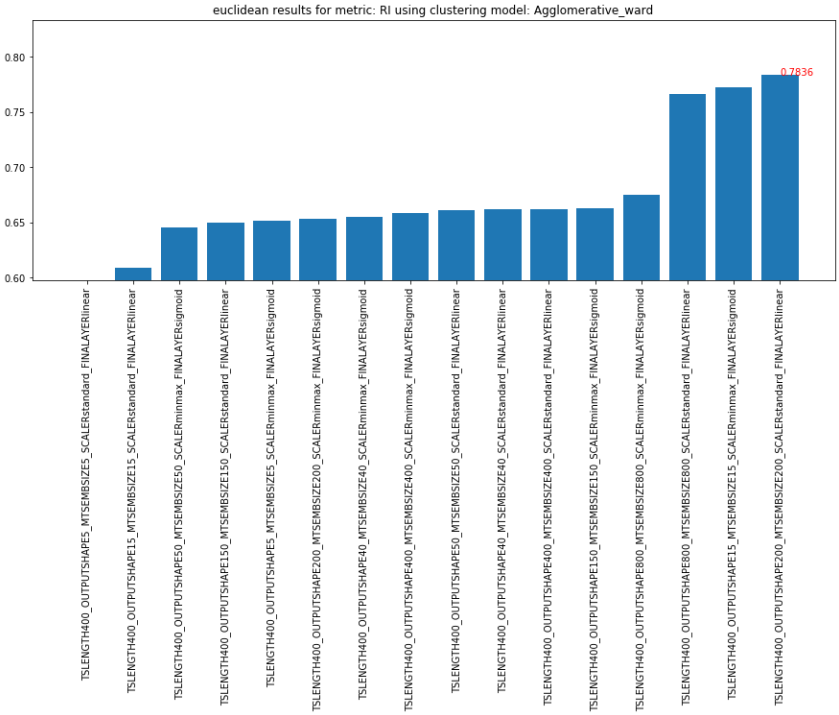

euclidean results for metric: Silhouette using clustering model: Agglomerative_avg
0.6554
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear

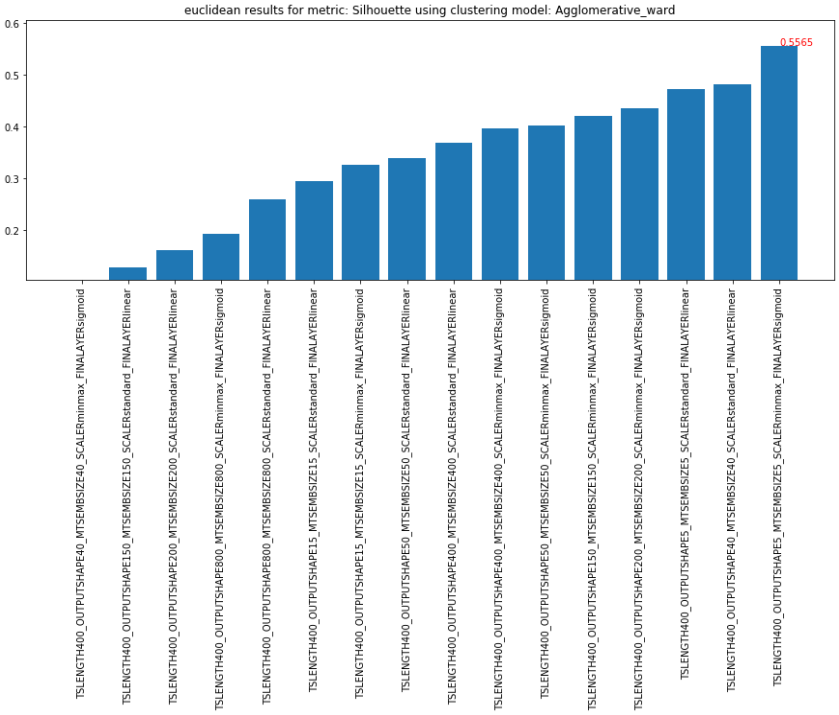

euclidean results for metric: ARI using clustering model: Agglomerative_min
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0.1499
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid

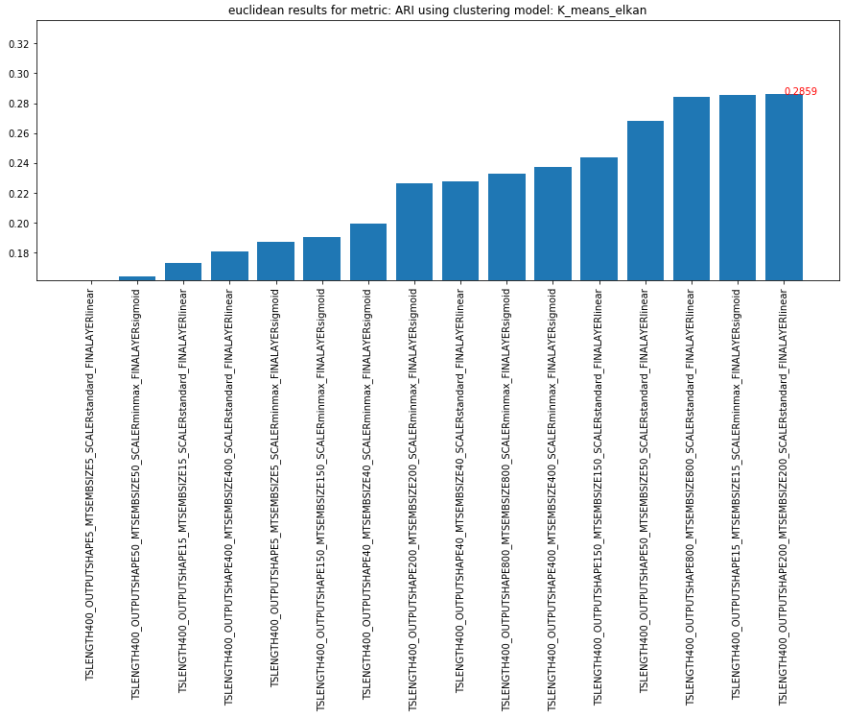

euclidean results for metric: RI using clustering model: Agglomerative_min
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.4907
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid

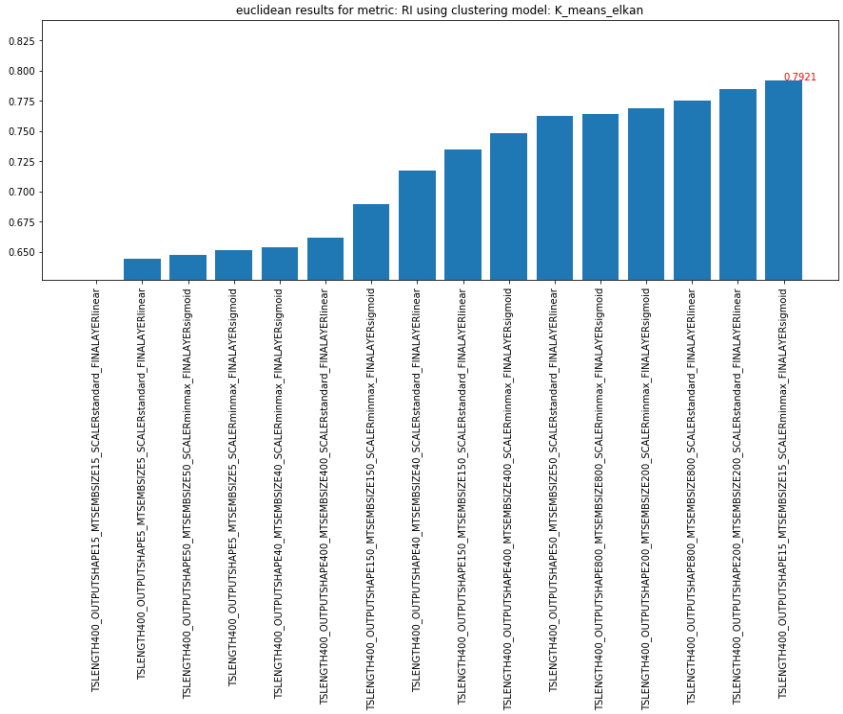

euclidean results for metric: Silhouette using clustering model: Agglomerative_min
0.6342
0.6
0.5
0.4
0.3
0.2
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear

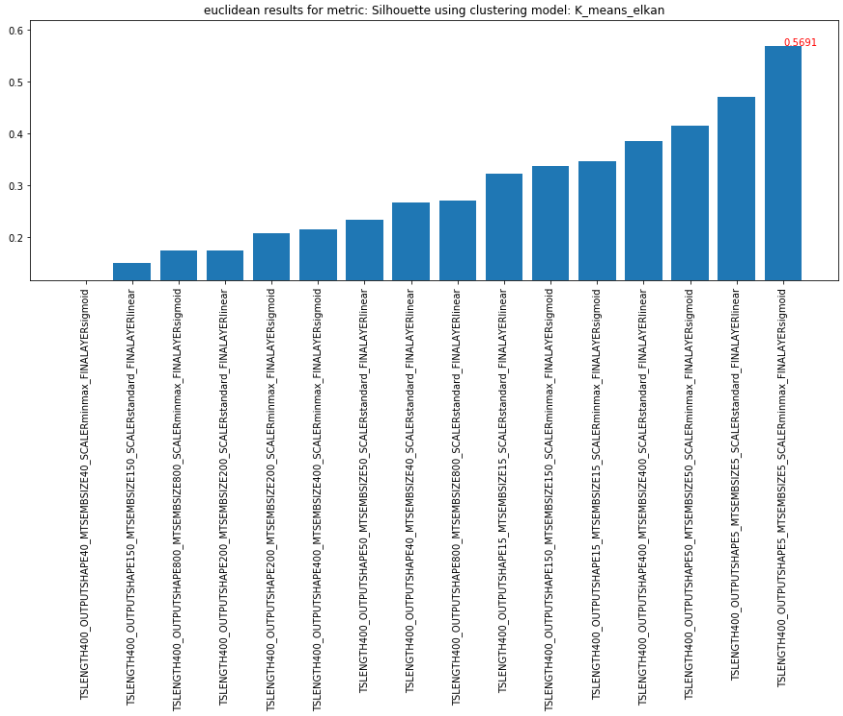

euclidean results for metric: ARI using clustering model: Agglomerative_ward
0.2923
0.325
0.300
0.275
0.250
0.225
0.200
0.175
0.150
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear

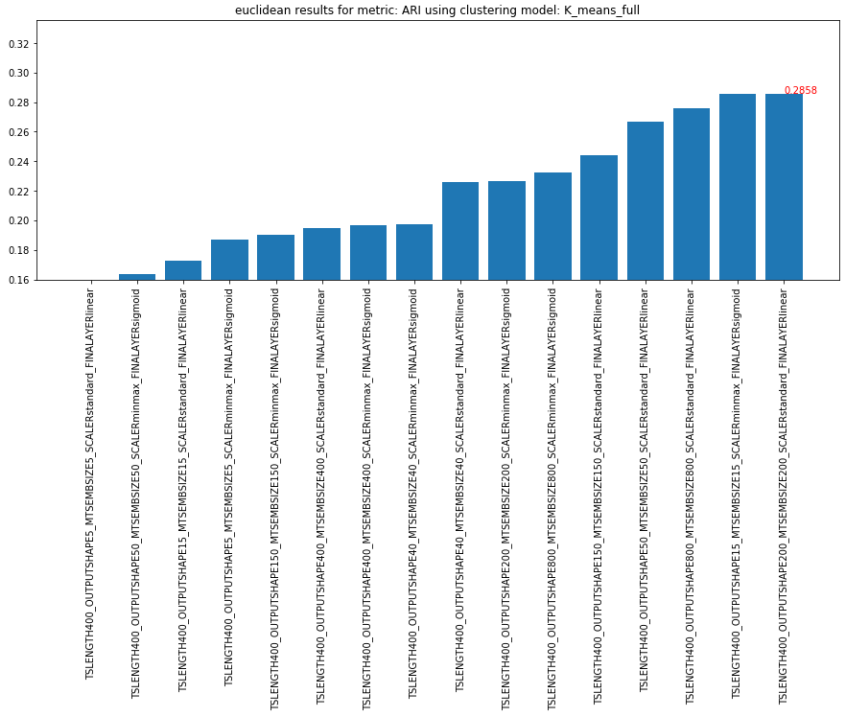

euclidean results for metric: RI using clustering model: Agglomerative_ward
0.80
0.75
0.70
0.65
0.60
0.7836
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear

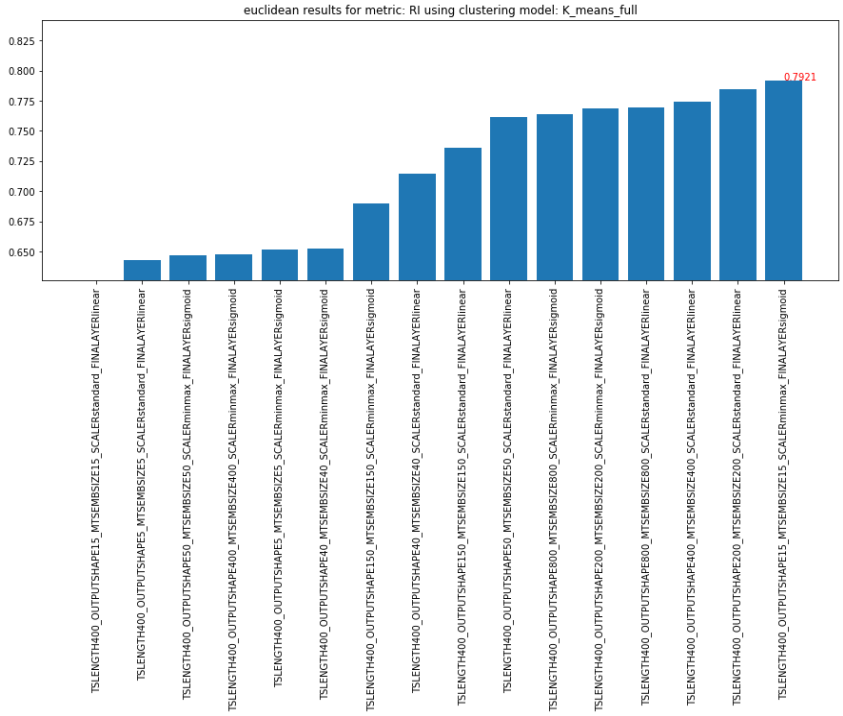

euclidean results for metric: Silhouette using clustering model: Agglomerative_ward
0.6
0.5
0.4
0.3
0.2
0.5565
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid

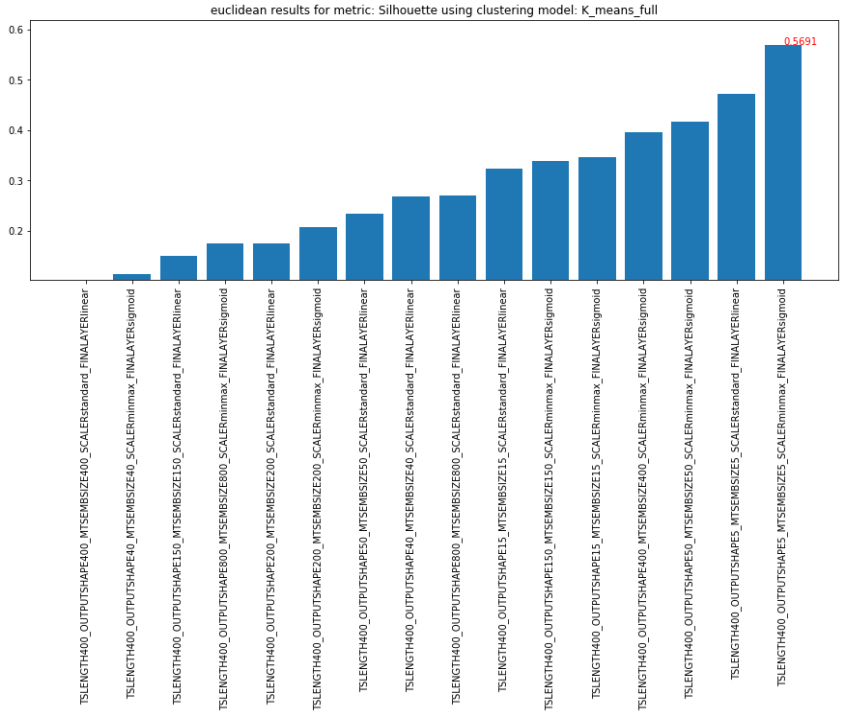

euclidean results for metric: ARI using clustering model: K_means_elkan
0.32
0.30
0.28
0.26
0.24
0.22
0.20
0.18
0.2859
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear

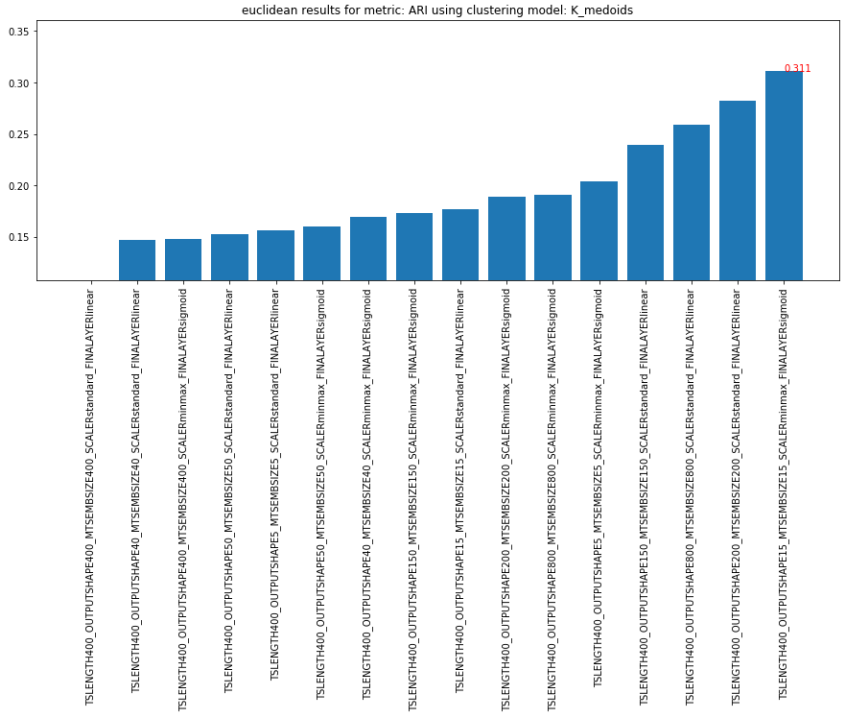

euclidean results for metric: RI using clustering model: K_means_elkan
0.825
0.800
0.775
0.750
0.725
0.700
0.675
0.650
0.7921
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid

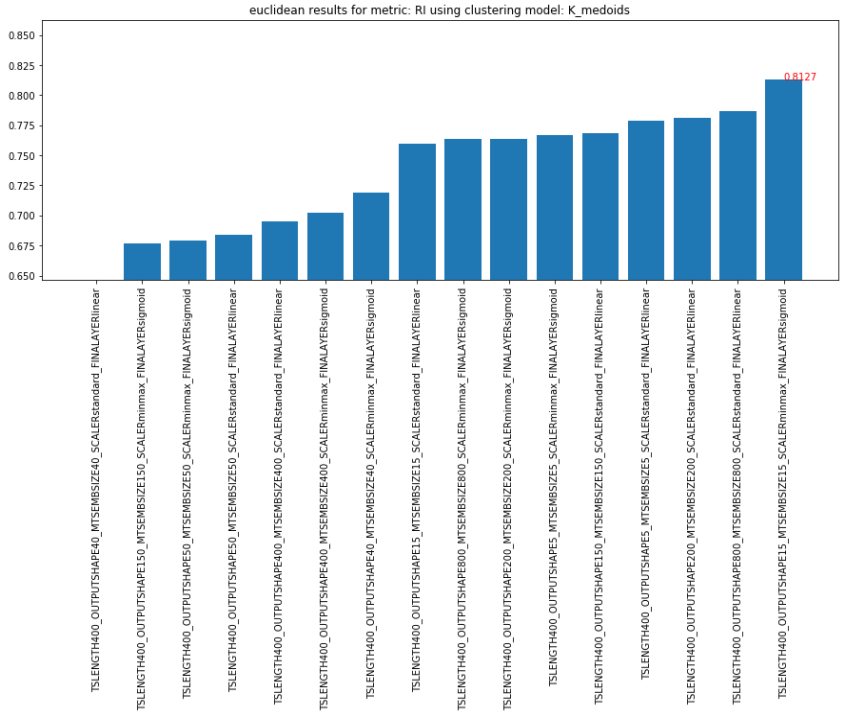

euclidean results for metric: Silhouette using clustering model: K_means_elkan
0.6
0.5
0.4
0.3
0.2
0.5691
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid

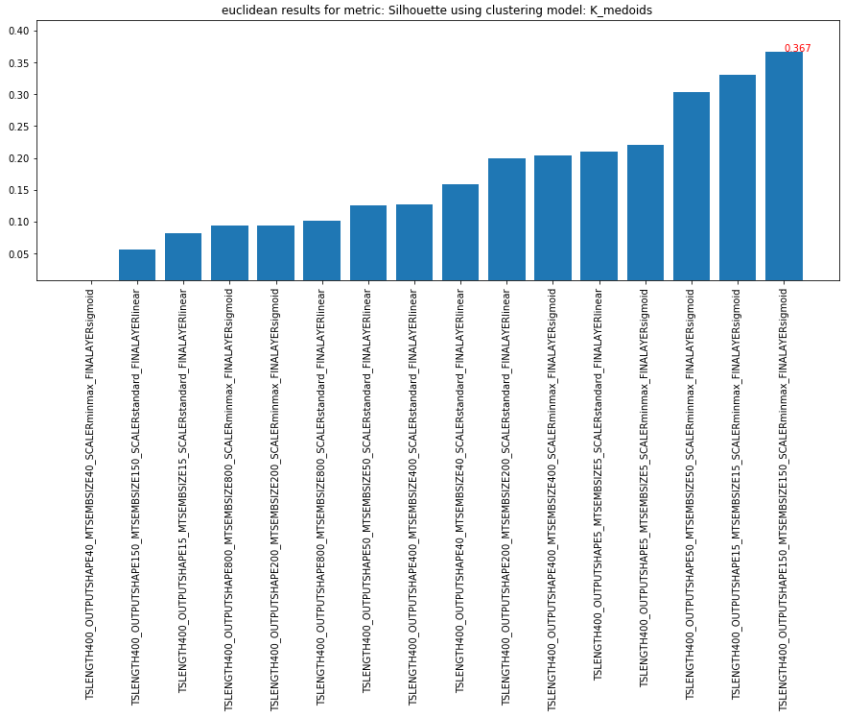

euclidean results for metric: ARI using clustering model: K_means_full
0.32
0.30
0.28
0.26
0.24
0.22
0.20
0.18
0.16
0.2858
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear

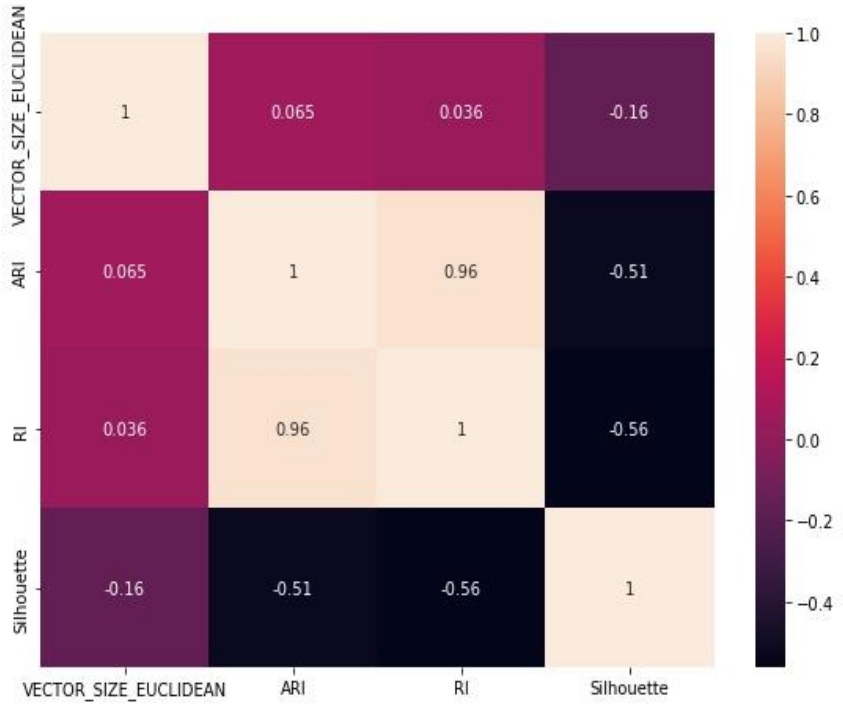

euclidean results for metric: RI using clustering model: K_means_full
0.825
0.800
0.775
0.750
0.725
0.700
0.675
0.650
0.7921
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid


euclidean results for metric: Silhouette using clustering model: K_means_full
0.6
0.5
0.4
0.3
0.2
0.5691
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid


euclidean results for metric: ARI using clustering model: K_medoids
0.35
0.30
0.25
0.20
0.15
0.311
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid


euclidean results for metric: RI using clustering model: K_medoids
0.850
0.825
0.800
0.775
0.750
0.725
0.700
0.675
0.650
0.8127
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid





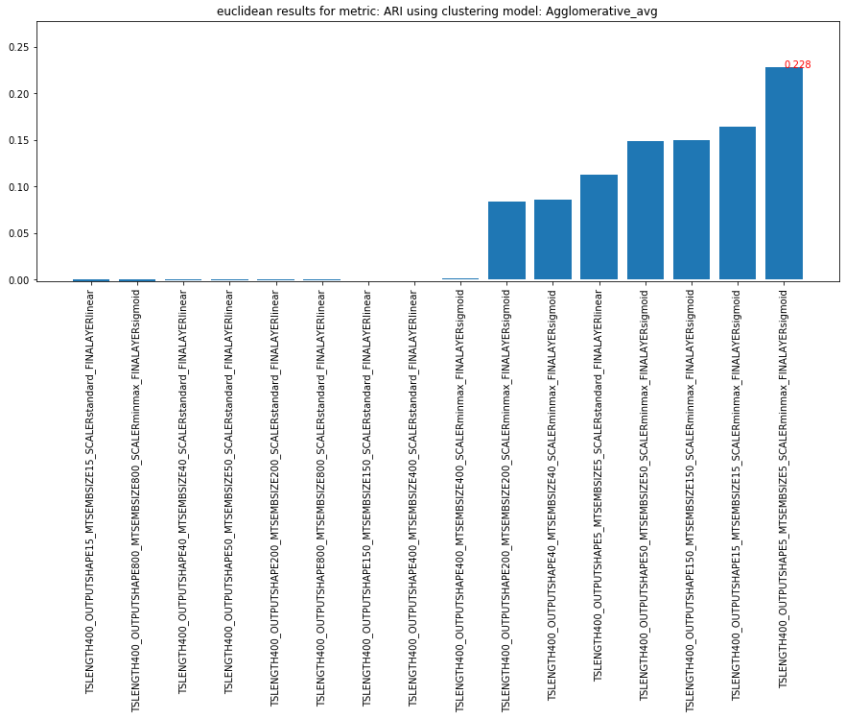
AdjustedRI	
Clustering Model	K_medoids
ARI	0.31100268196457836
RI	0.8127222527640988
Silhouette	0.33008677
Parameters	EMBSIZE: 15 SCALER: minmax
RandomIndex	

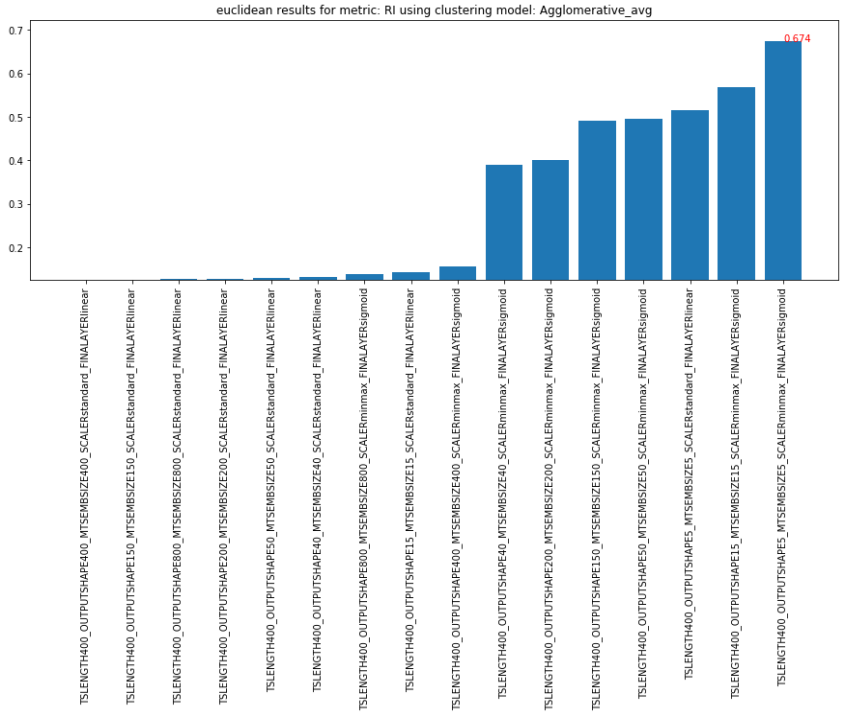
Clustering Model	K_medoids
ARI	0.31100268196457836
RI	0.8127222527640988
Silhouette	0.33008677
Parameters	EMBSIZE: 15 SCALER: minmax
Silhouette	

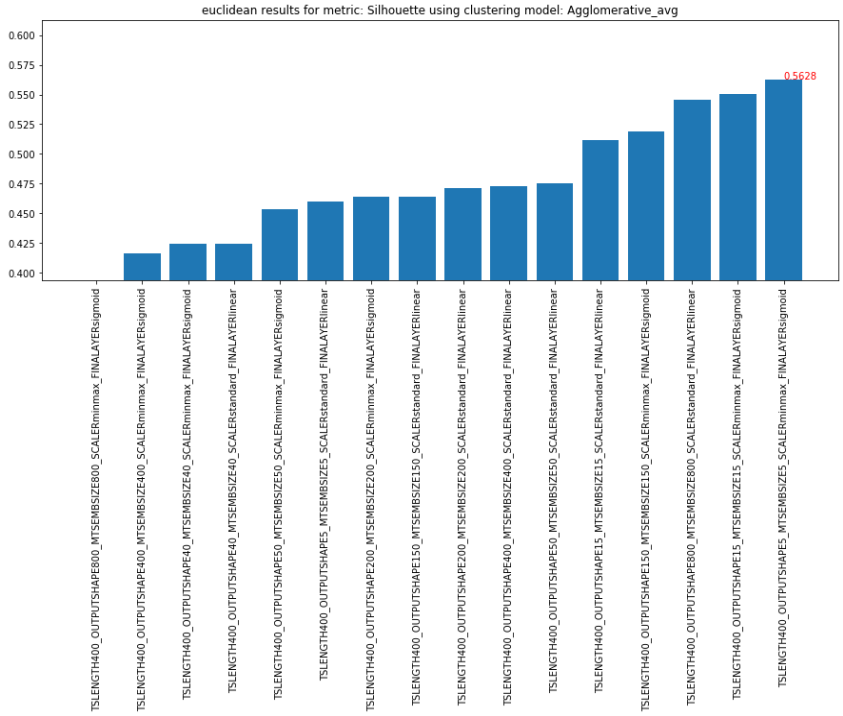
Clustering model	Agglomerative_avg
ARI	-0.0009653209363112494
RI	0.1294089881637509
Silhouette	0.65538186
Parameters	EMBSIZE: 15 SCALER: standard

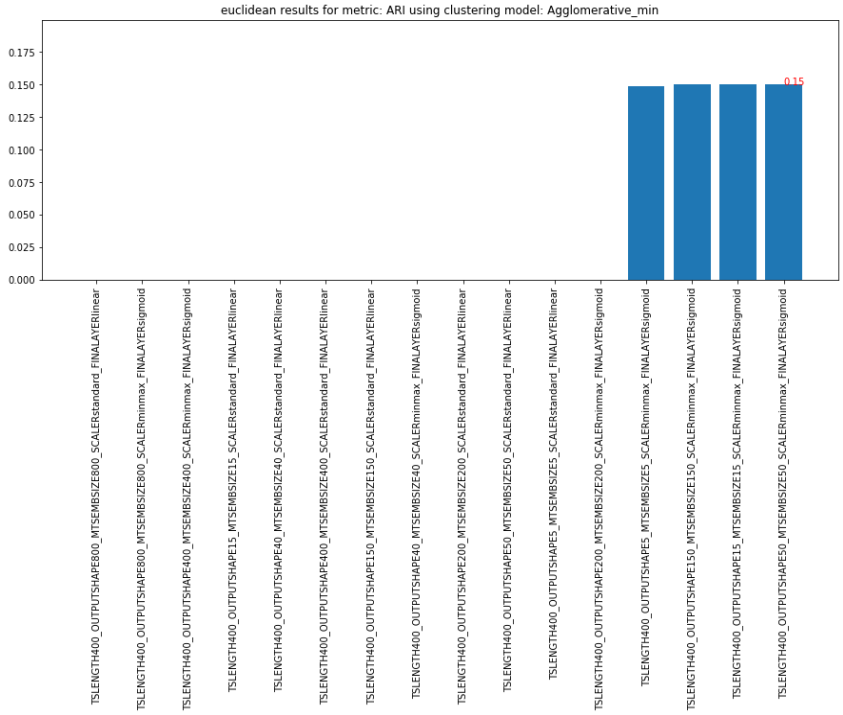
Tabella 13 Risultati migliori per CNN con vettori concatenati e distanza euclidea

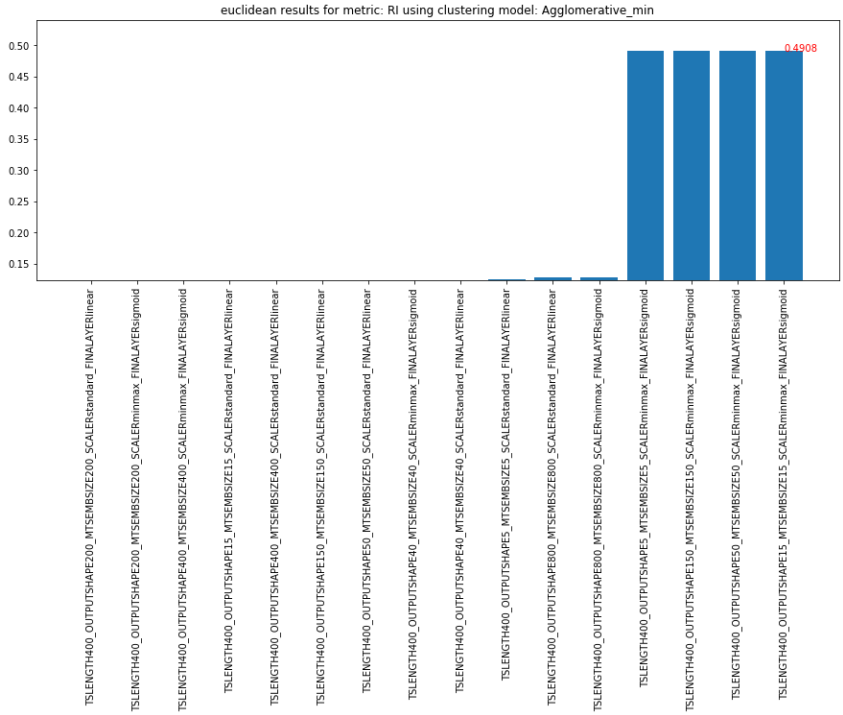
Secondo embedding

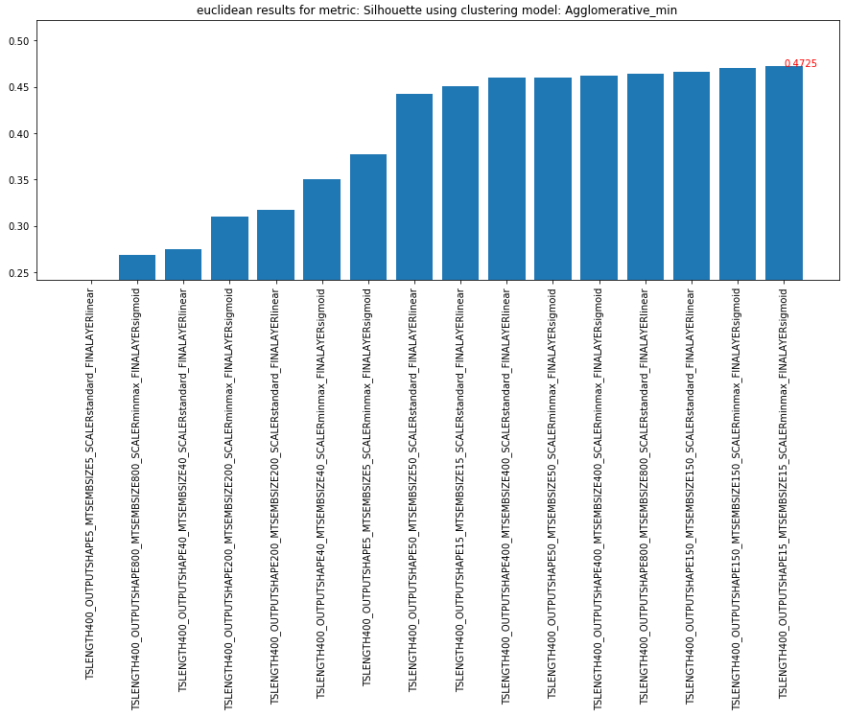


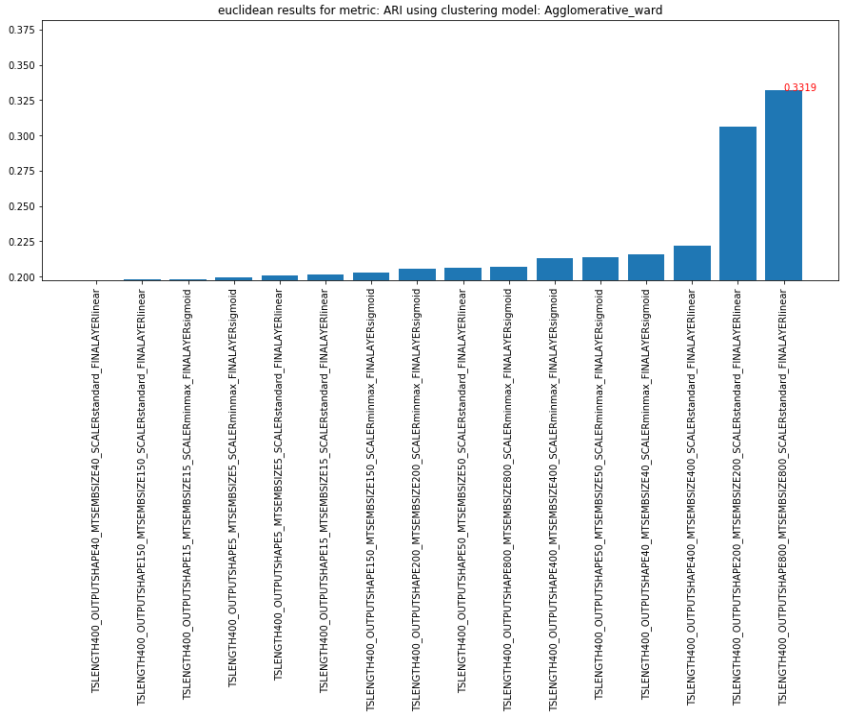


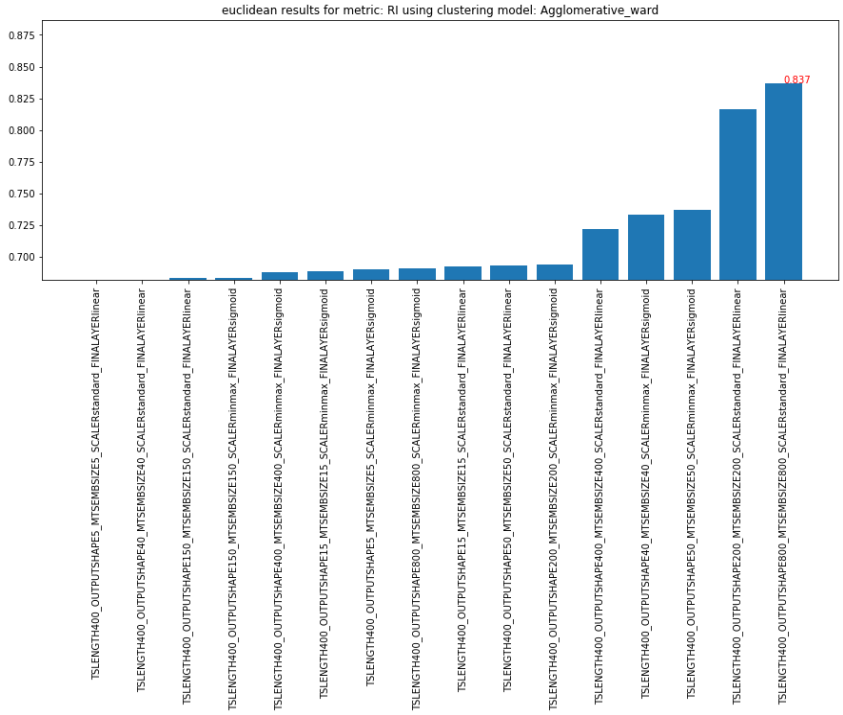

euclidean results for metric: Silhouette using clustering model: Agglomerative_avg
0.600
0.575
0.550
0.525
0.500
0.475
0.450
0.425
0.400
0.5628
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid

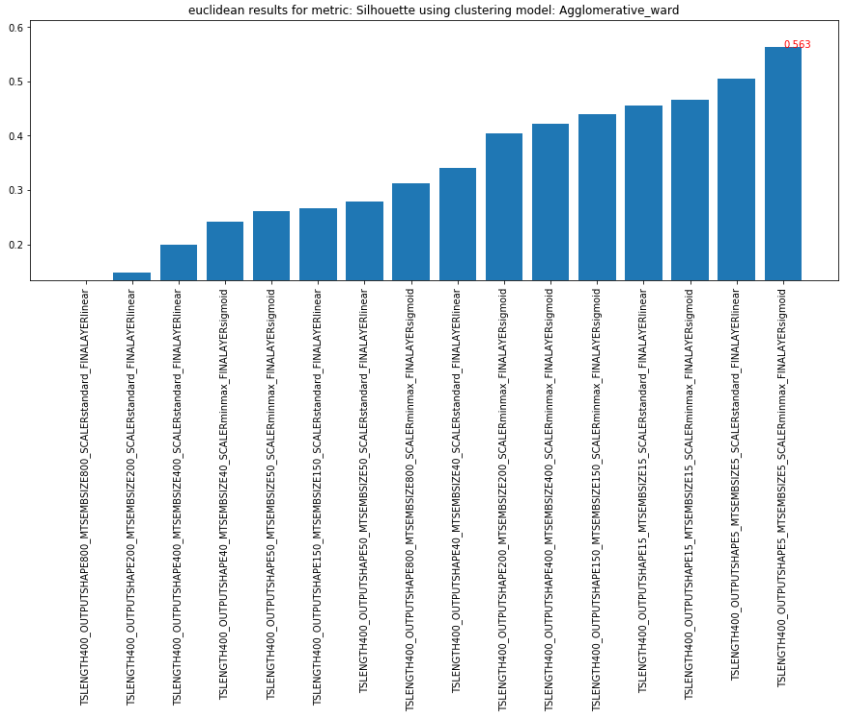

euclidean results for metric: ARI using clustering model: Agglomerative_min
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0.15
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid

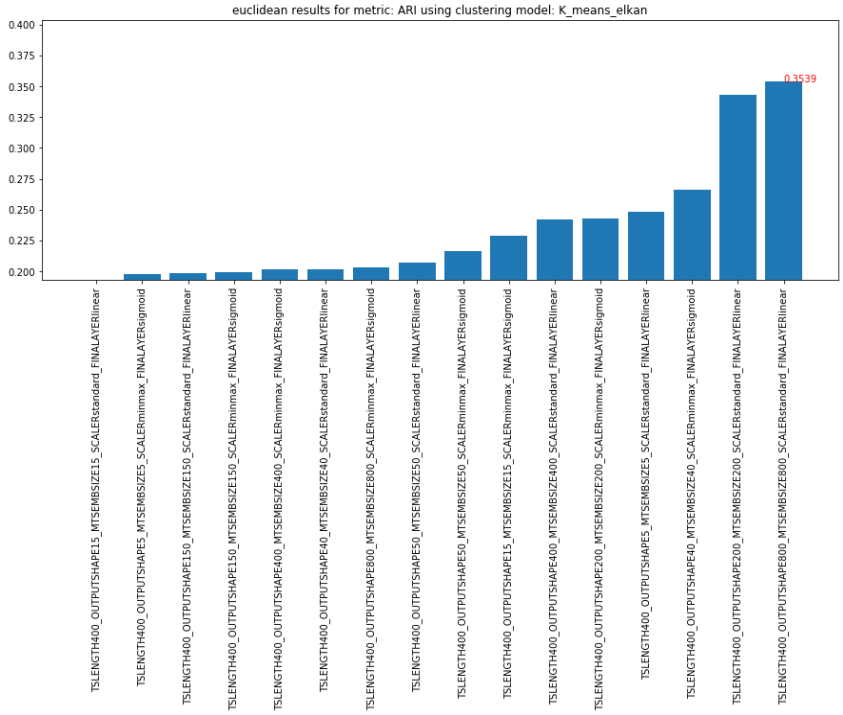

euclidean results for metric: RI using clustering model: Agglomerative_min
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.4908
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid

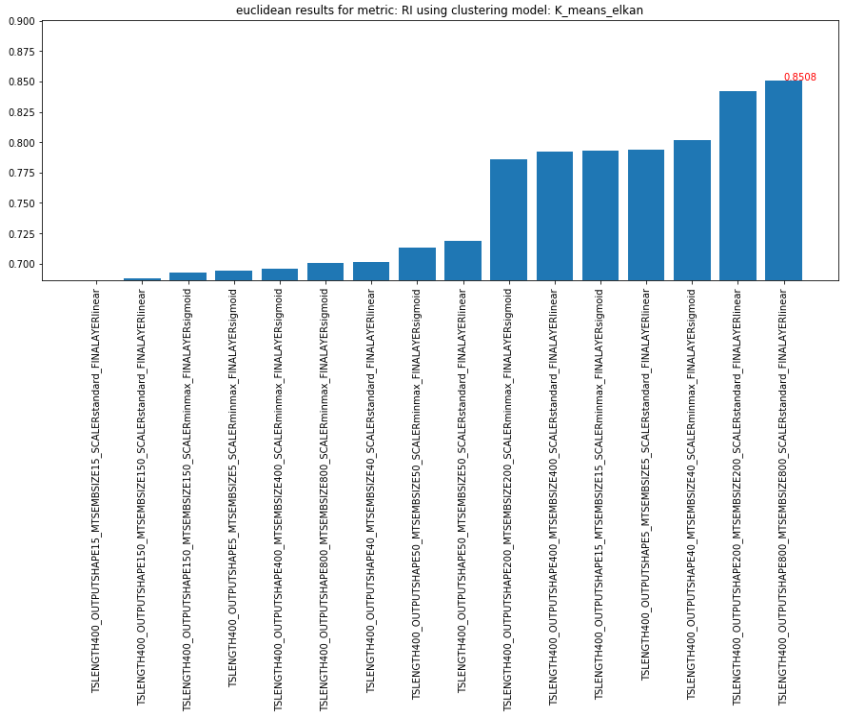

euclidean results for metric: Silhouette using clustering model: Agglomerative_min
0.50
0.45
0.40
0.35
0.30
0.25
0.4725
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid

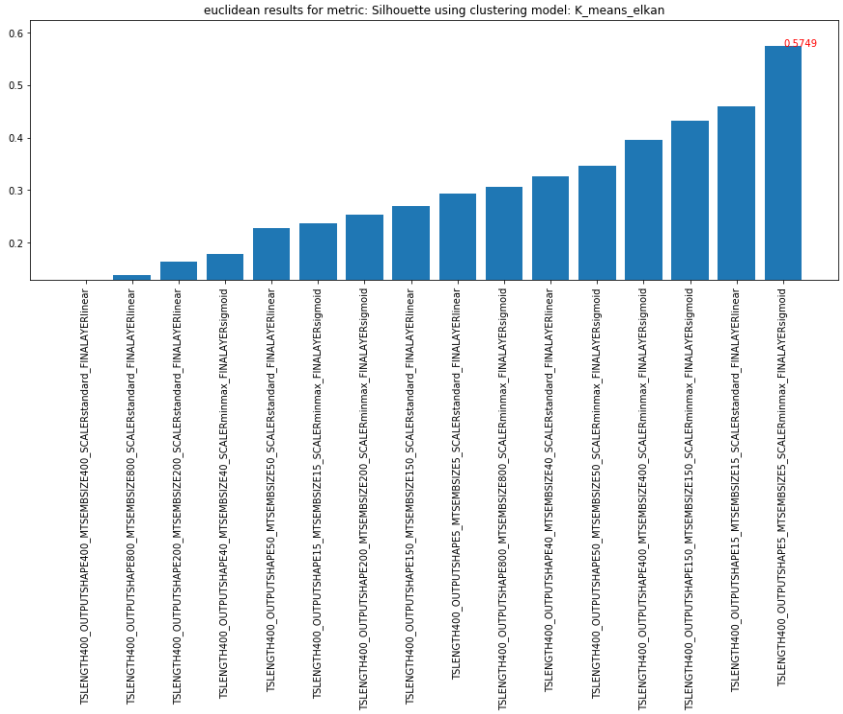

euclidean results for metric: ARI using clustering model: Agglomerative_ward
0.375
0.350
0.325
0.300
0.275
0.250
0.225
0.200
0.3319
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear

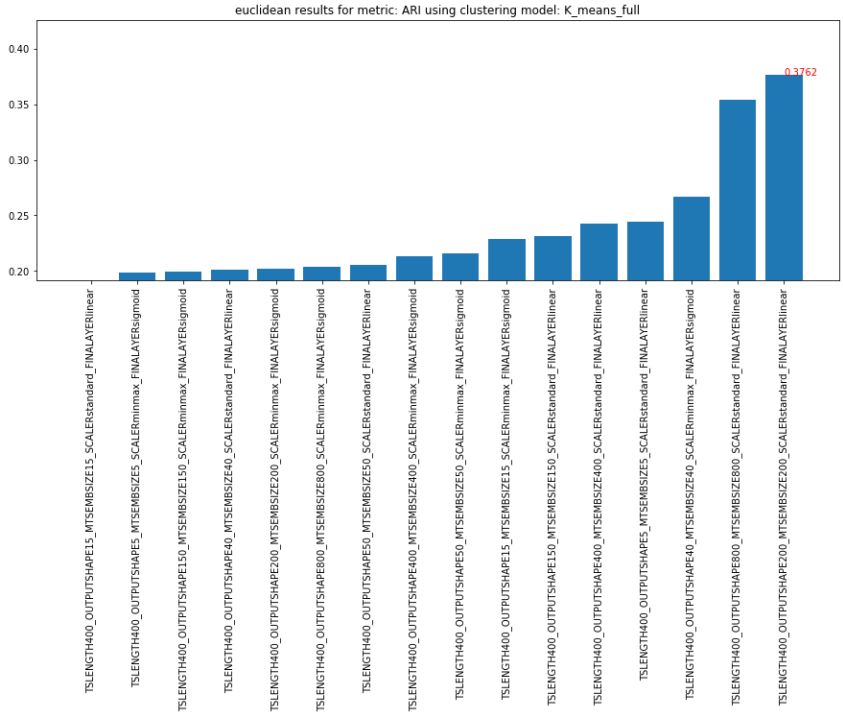

euclidean results for metric: RI using clustering model: Agglomerative_ward
0.875
0.850
0.825
0.800
0.775
0.750
0.725
0.700
0.837
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear

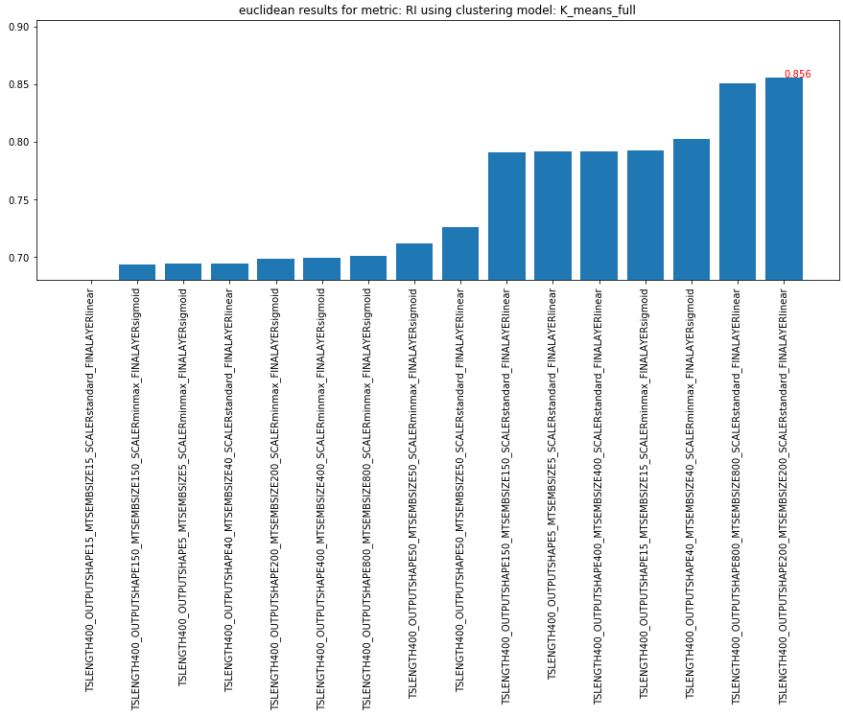

euclidean results for metric: Silhouette using clustering model: Agglomerative_ward
0.6
0.5
0.4
0.3
0.2
0.563
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid

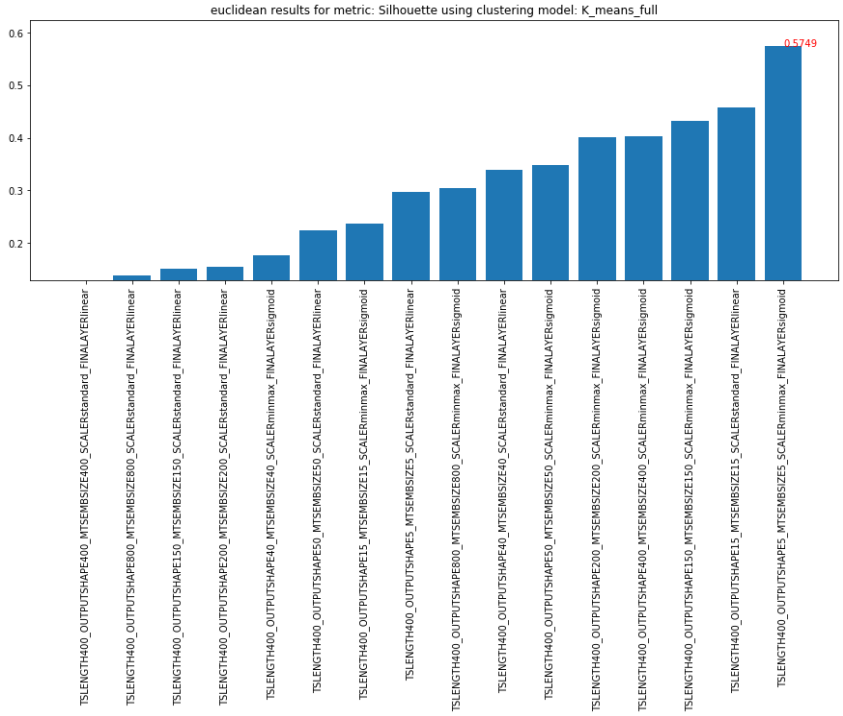

euclidean results for metric: ARI using clustering model: K_means_elkan
0.400
0.375
0.350
0.325
0.300
0.275
0.250
0.225
0.200
0.3539
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear

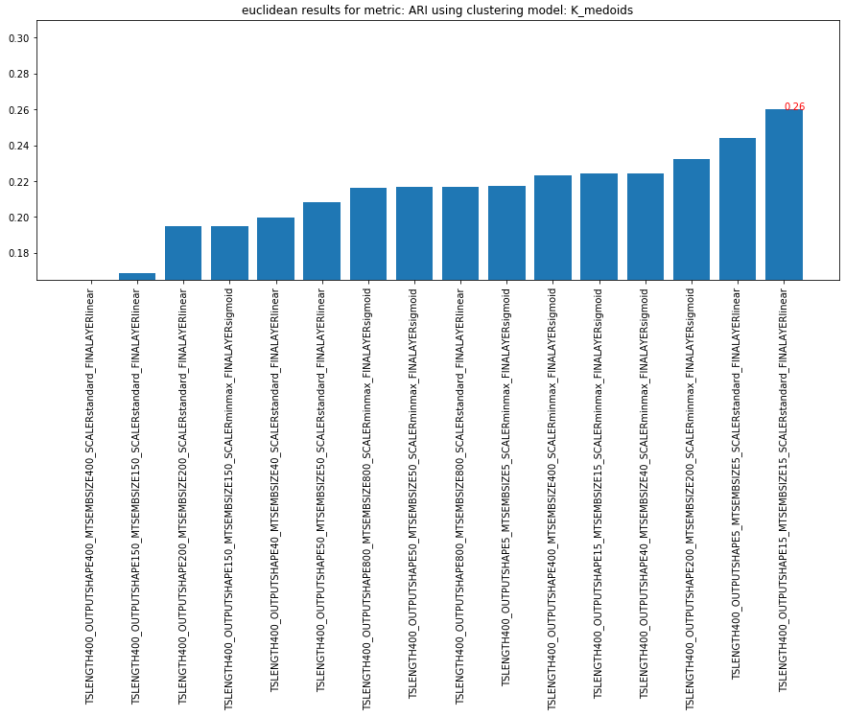

euclidean results for metric: RI using clustering model: K_means_elkan
0.900
0.875
0.850
0.825
0.800
0.775
0.750
0.725
0.700
0.8508
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear

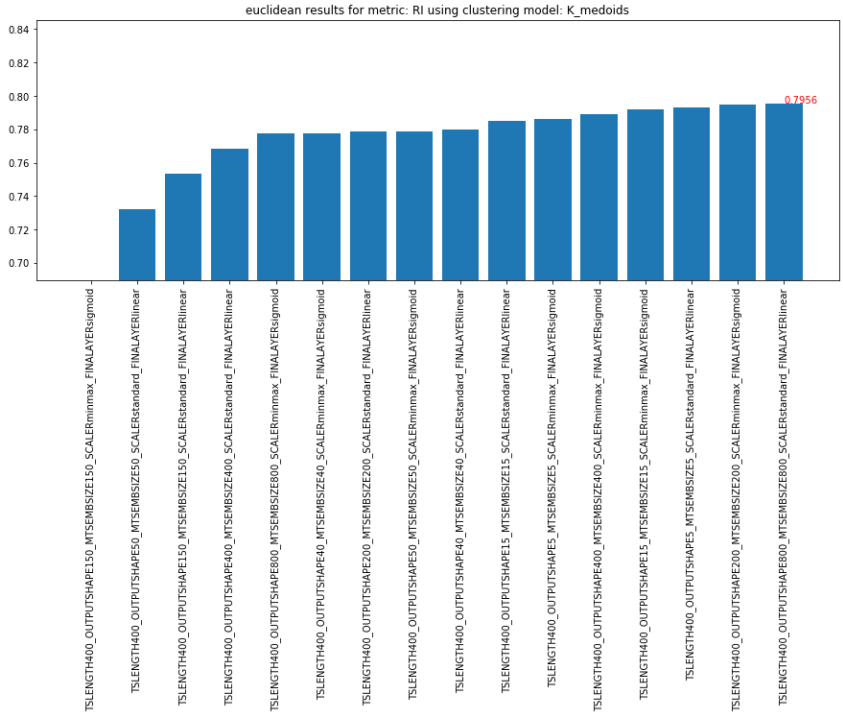

euclidean results for metric: Silhouette using clustering model: K_means_elkan
0.6
0.5
0.4
0.3
0.2
0.5749
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid

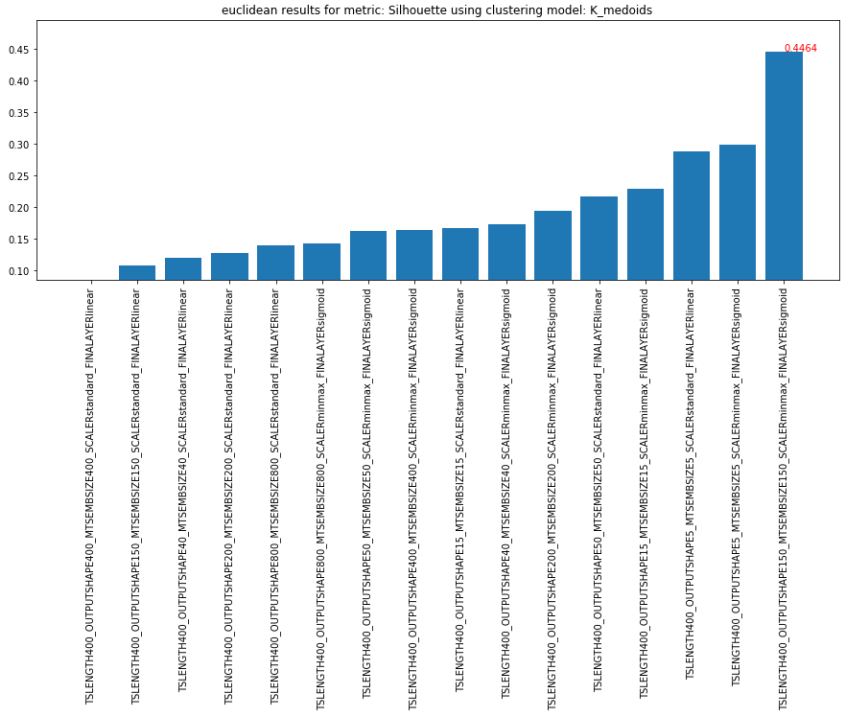

euclidean results for metric: ARI using clustering model: K_means_full
0.40
0.35
0.30
0.25
0.20
0.3762
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear

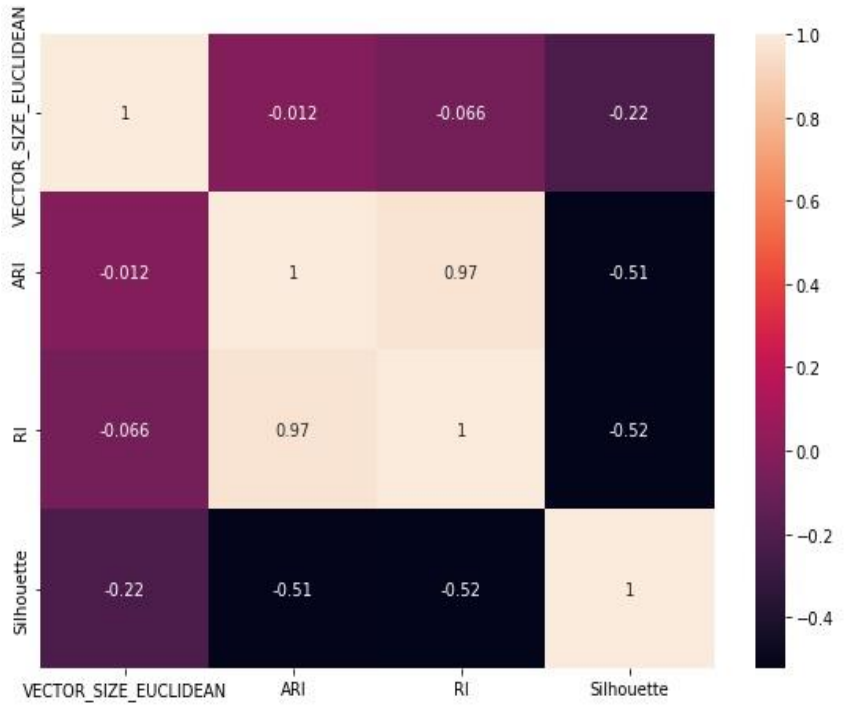

euclidean results for metric: RI using clustering model: K_means_full
0.90
0.85
0.80
0.75
0.70
0.856
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear


euclidean results for metric: Silhouette using clustering model: K_means_full
0.6
0.5
0.4
0.3
0.2
0.5749
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid


euclidean results for metric: ARI using clustering model: K_medoids
0.30
0.28
0.26
0.24
0.22
0.20
0.18
0.26
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear


euclidean results for metric: RI using clustering model: K_medoids
0.84
0.82
0.80
0.78
0.76
0.74
0.72
0.70
0.7956
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear


euclidean results for metric: Silhouette using clustering model: K_medoids
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.4464
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid

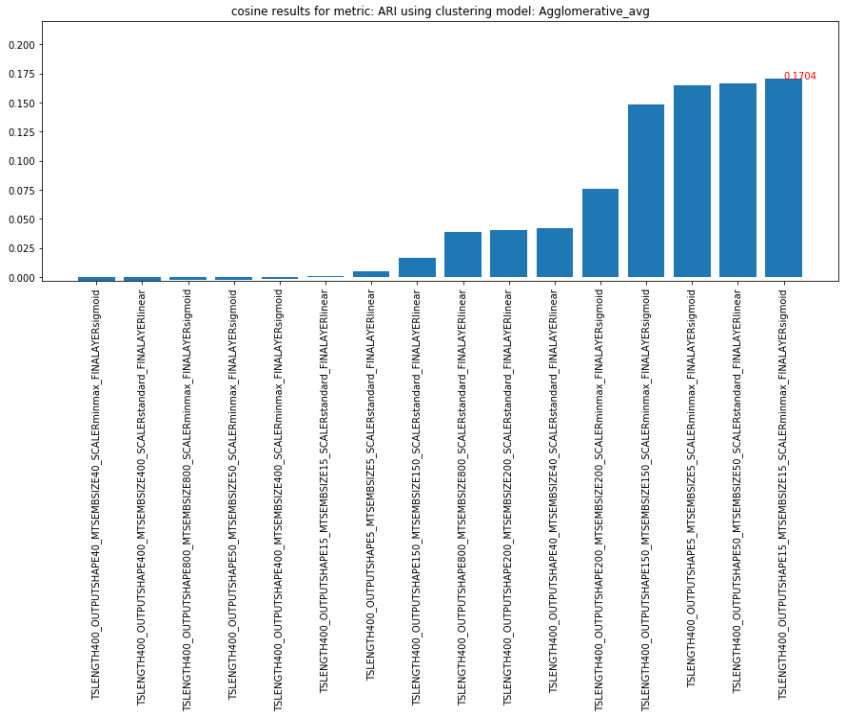


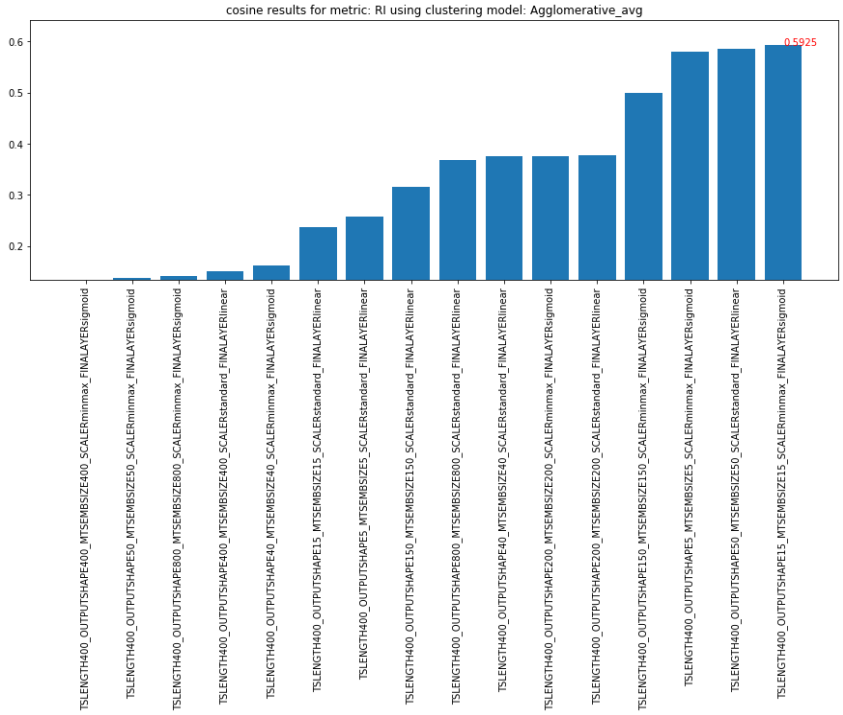
AdjustedRI	
Clustering model	K_means_full
ARI	0.3761783584137743
RI	0.8559804354247745
Silhouette	0.15489413
Parameters	EMBSIZE: 200 SCALER: standard
RandomIndex	
Clustering model	K_means_full
ARI	0.3761783584137743
RI	0.8559804354247745
Silhouette	0.15489413
Parameters	EMBSIZE: 200 SCALER: standard
Silhouette	
Clustering model	K_means_elkan
ARI	0.19803036492730047
RI	0.6940336078315801
Silhouette	0.5748721
Parameters	EMBSIZE: 5 SCALER: minmax

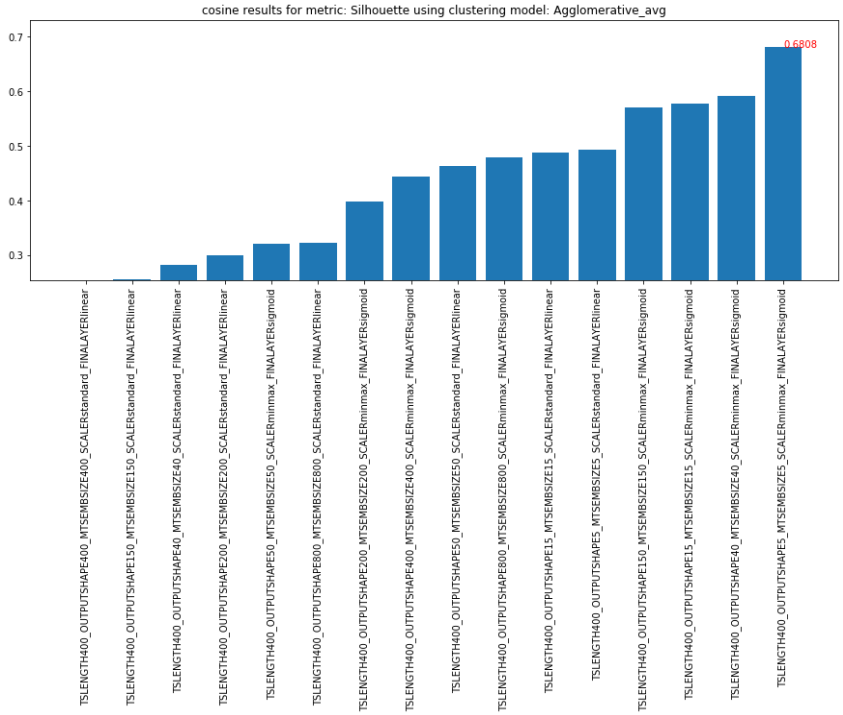
Tabella 14 Risultati migliori per CNN con secondo embedding e distanza euclidea

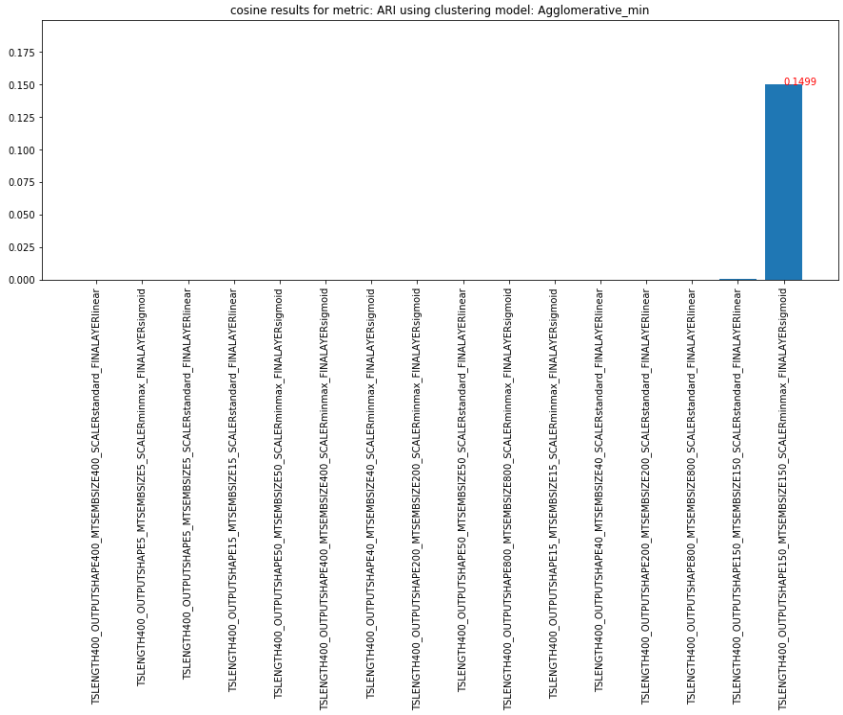
Cosine

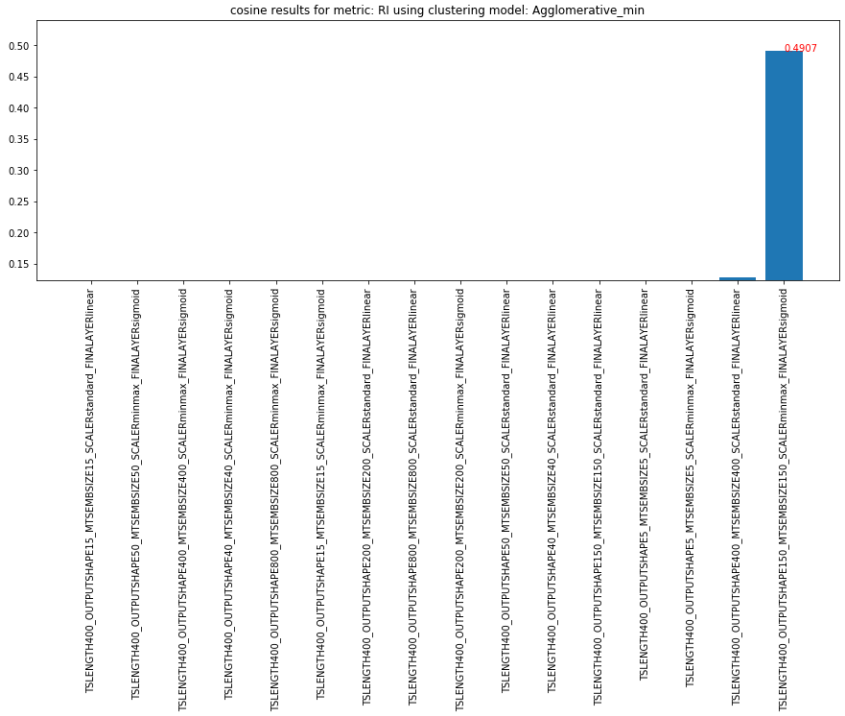
Vettori concatenati

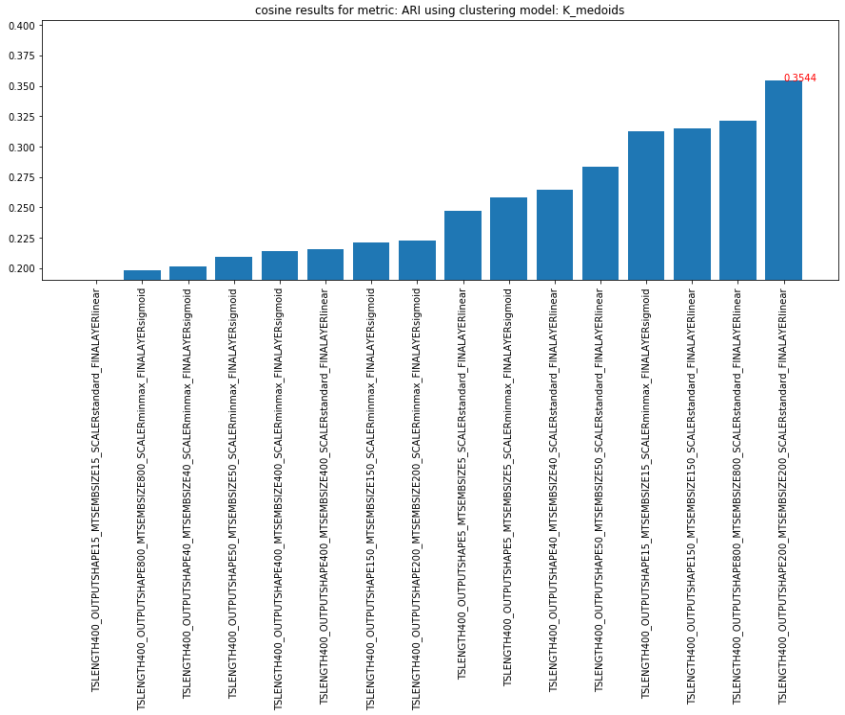


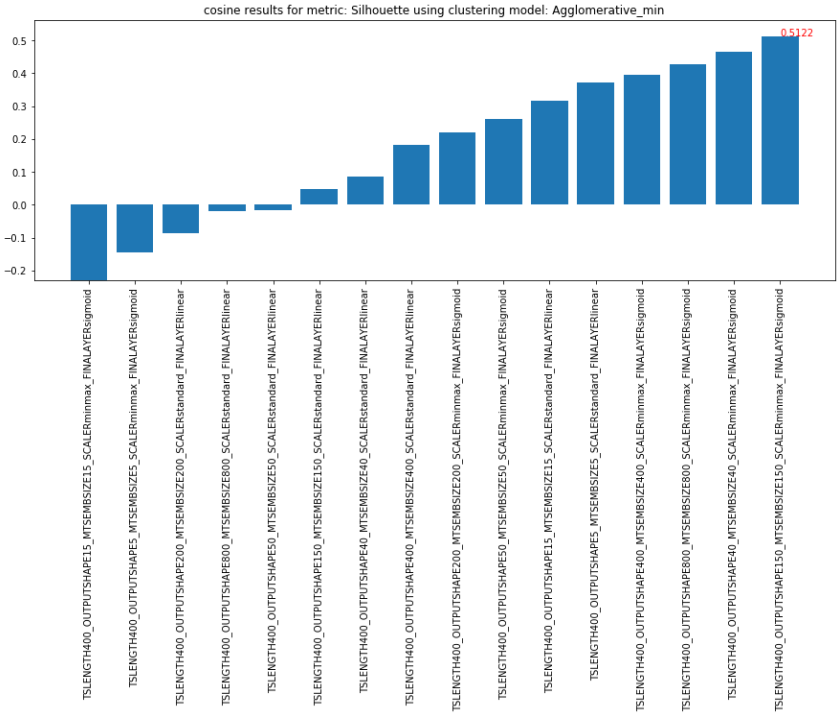


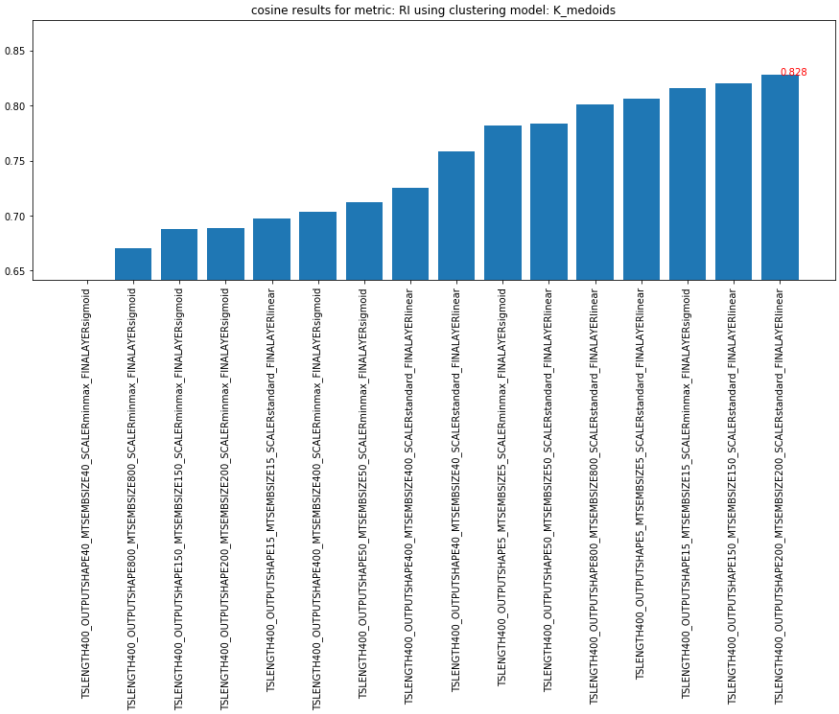

cosine results for metric: Silhouette using clustering model: Agglomerative_avg
0.7
0.6
0.5
0.4
0.3
0.6808
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid

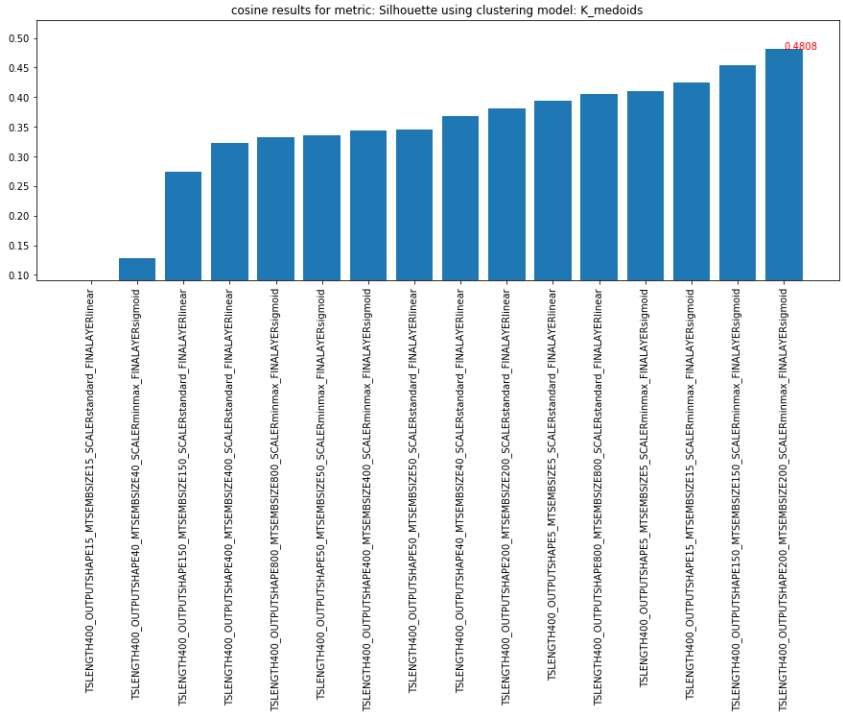

cosine results for metric: ARI using clustering model: Agglomerative_min
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0.1499
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid

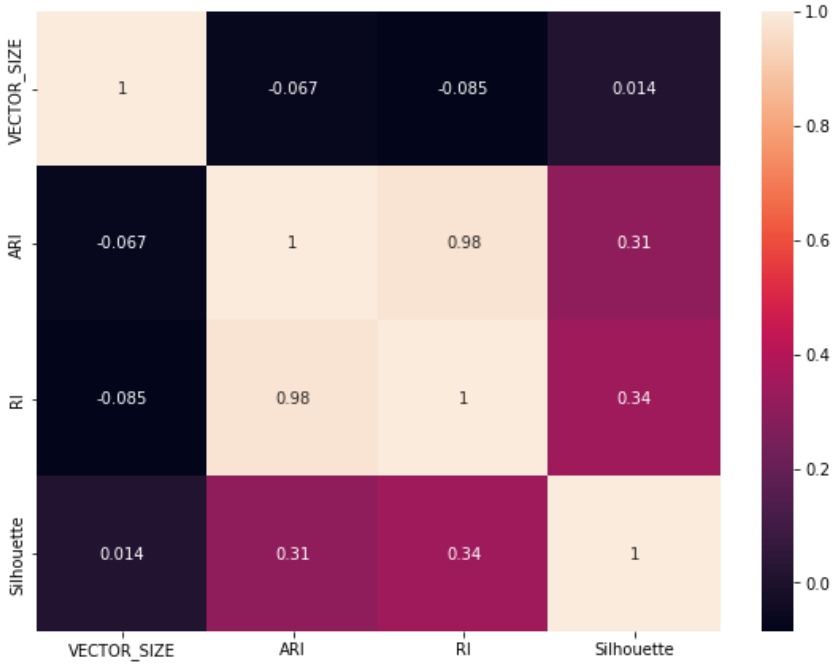

cosine results for metric: RI using clustering model: Agglomerative_min
0.4907
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid


cosine results for metric: ARI using clustering model: K_medoids
0.3544
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear


cosine results for metric: Silhouette using clustering model: Agglomerative_min
0.5122
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid


cosine results for metric: RI using clustering model: K_medoids
0.828
0.85
0.80
0.75
0.70
0.65
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear



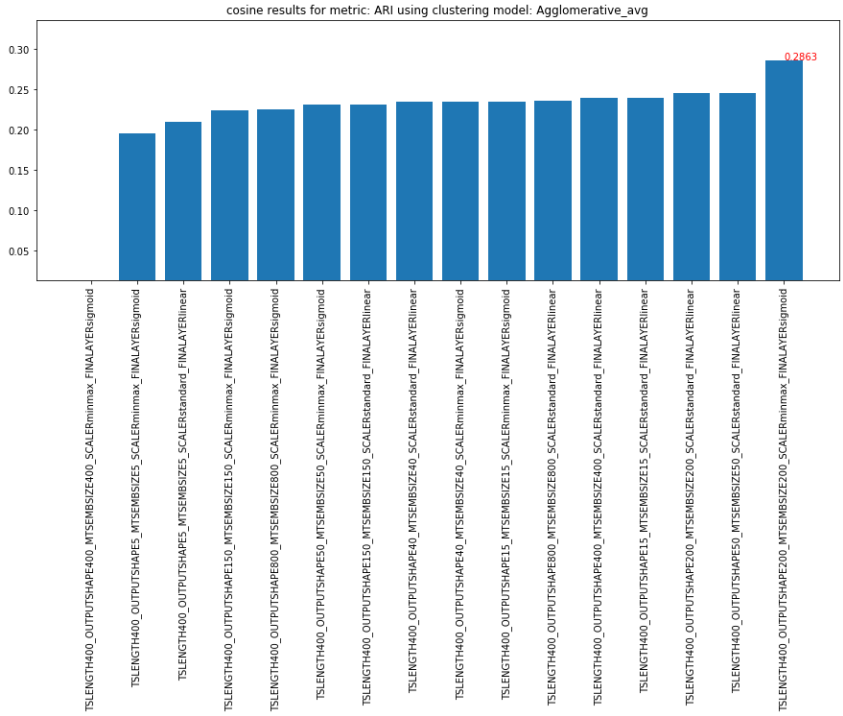


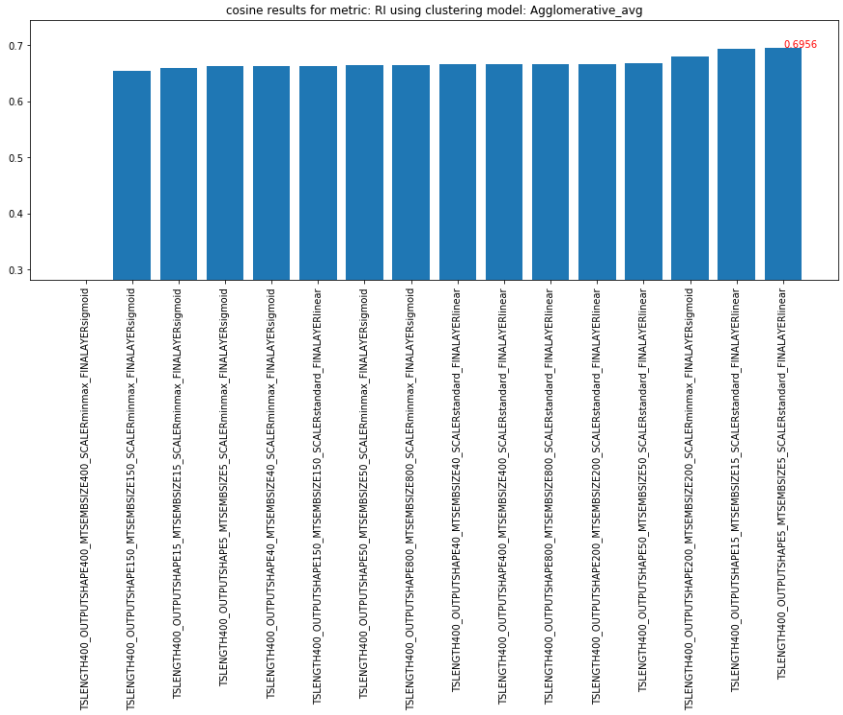
AdjustedRI	
Clustering model	K_medoids
ARI	0.35438466919788086
RI	0.8280336443469372
Silhouette	0.38115513
Parameters	EMBSIZE: 200 SCALER: standard
RandomIndex	
Clustering model	K_medoids

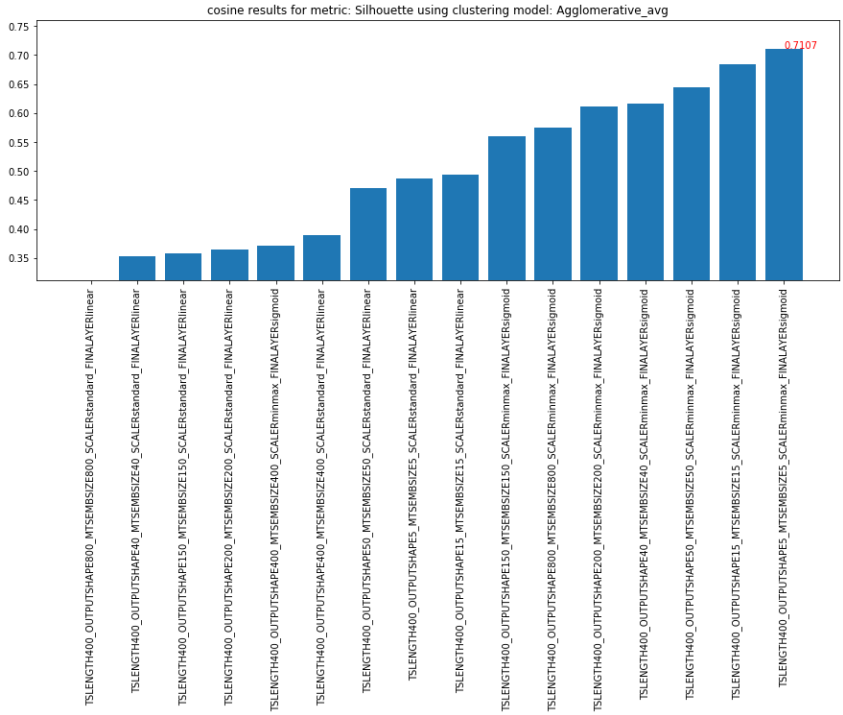
ARI	0.35438466919788086
RI	0.8280336443469372
Silhouette	0.38115513
Parameters	EMBSIZE: 200 SCALER: standard
Silhouette	
Clustering model	Agglomerative avg
ARI	0.16508126485290361
RI	0.5792624458794009
Silhouette	0.6808253
Parameters	EMBSIZE: 5 SCALER: minmax

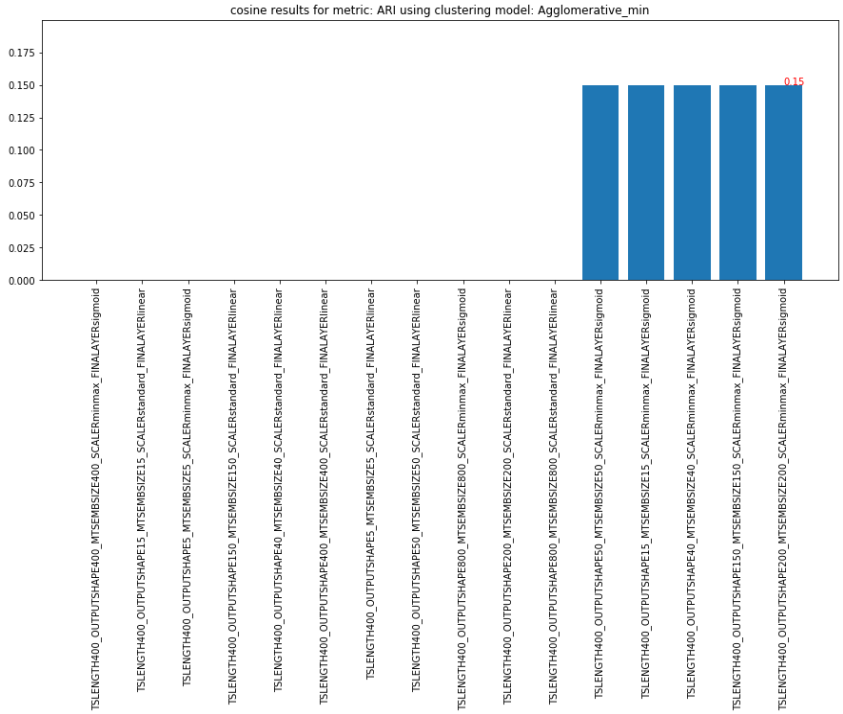
Tabella 15 Risultati migliori per CNN con vettori concatenati e distanza del coseno

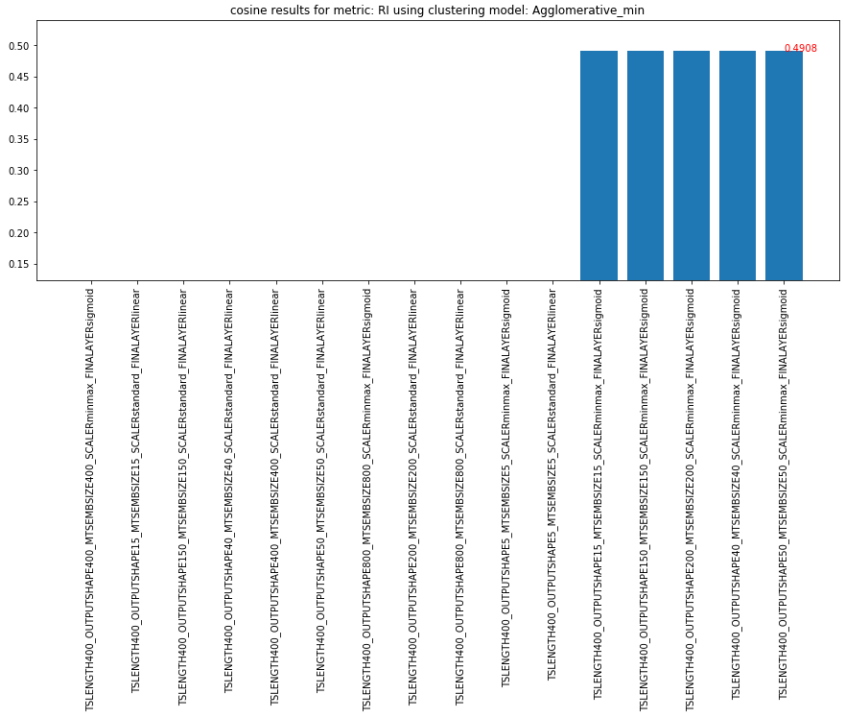
Secondo embedding

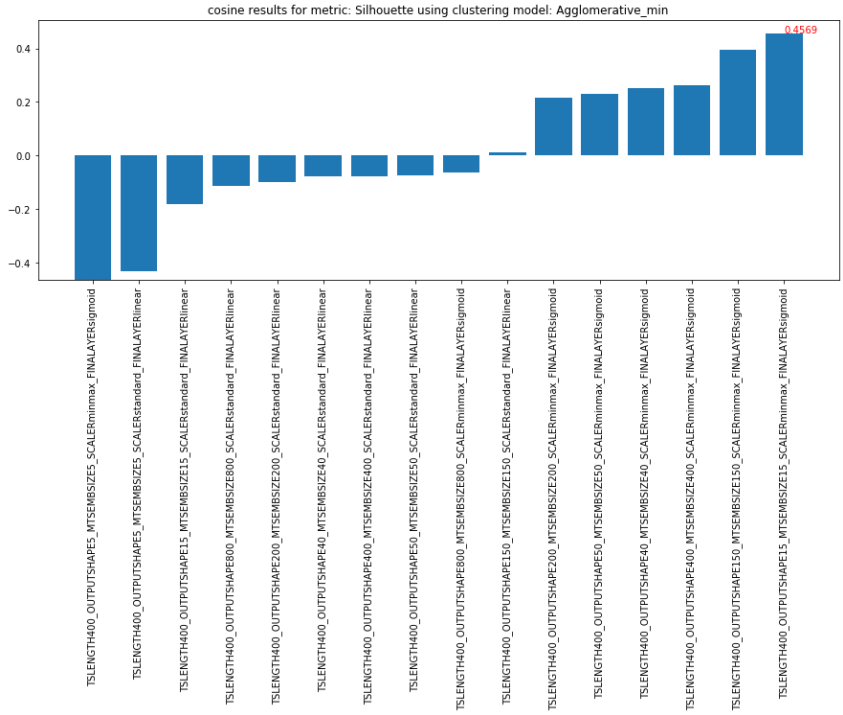


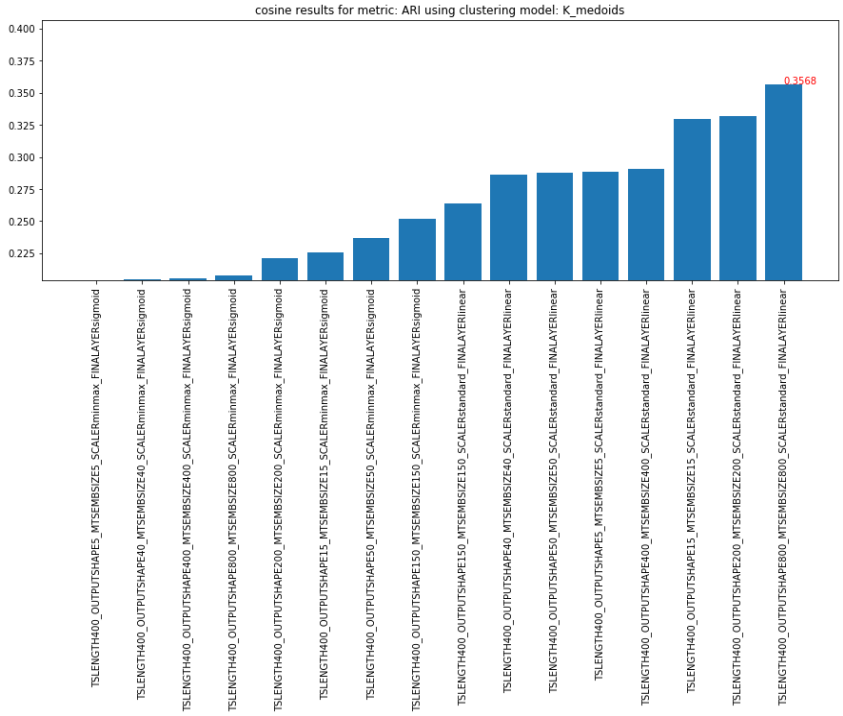


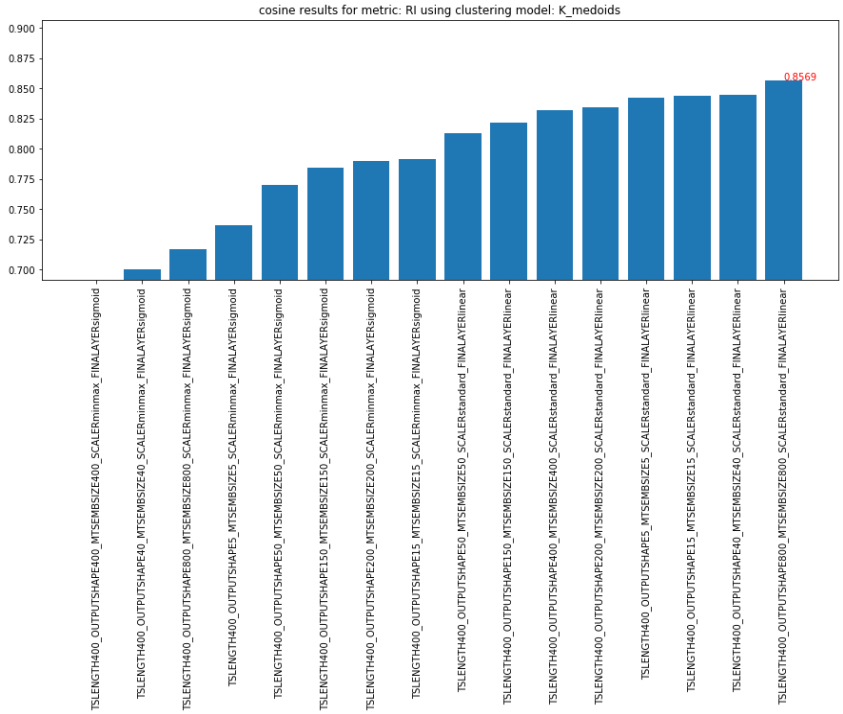

cosine results for metric: Silhouette using clustering model: Agglomerative_avg
0.7107
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid

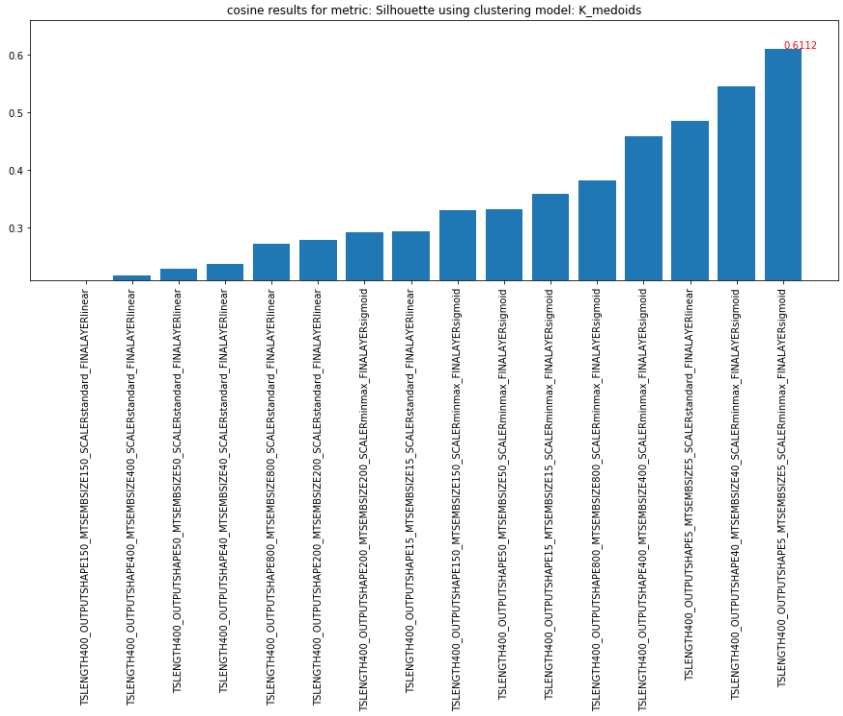

cosine results for metric: ARI using clustering model: Agglomerative_min
0.15
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid

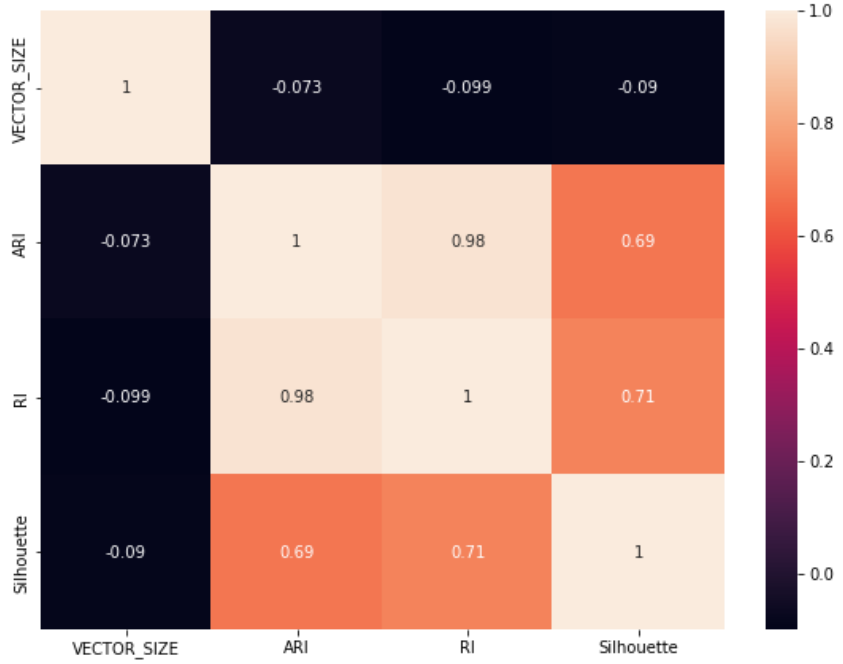

cosine results for metric: RI using clustering model: Agglomerative_min
0.4908
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid


cosine results for metric: Silhouette using clustering model: Agglomerative_min
0.4569
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid


cosine results for metric: ARI using clustering model: K_medoids
0.400
0.375
0.350
0.325
0.300
0.275
0.250
0.225
0.3568
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear


cosine results for metric: RI using clustering model: K_medoids
0.900
0.875
0.850
0.825
0.800
0.775
0.750
0.725
0.700
0.8569
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERminmax_FINALAYERsigmoid
TSLENGTH400_OUTPUTSHAPE50_MTSEMBSIZE50_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE150_MTSEMBSIZE150_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE400_MTSEMBSIZE400_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE200_MTSEMBSIZE200_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE5_MTSEMBSIZE5_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE15_MTSEMBSIZE15_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE40_MTSEMBSIZE40_SCALERstandard_FINALAYERlinear
TSLENGTH400_OUTPUTSHAPE800_MTSEMBSIZE800_SCALERstandard_FINALAYERlinear





AdjustedRI	
Clustering model	K_medoids
ARI	0.35681306298096216
RI	0.8569397760433617
Silhouette	0.27177474
Parameters	EMBSIZE: 800 SCALER: standard

RandomIndex

Clustering model	K_medoids
ARI	0.35681306298096216
RI	0.8569397760433617
Silhouette	0.27177474
Parameters	EMBSIZE: 800 SCALER: standard
Silhouette	
Clustering model	Agglomerative_avg
ARI	0.19500647198172222
RI	0.6623502777585093
Silhouette	0.7107077
Parameters	EMBSIZE: 5 SCALER: minmax

Tabella 16 Risultati migliori per ogni metrica per CNN con secondo embedding e distanza del coseno

Risultati DBSCAN

Vettori concatenati e distanza del coseno						
Parameters set	ARI	RI	SILHOUETTE	NUM_CLUSTER	NUM_OUTLIERS	
eps:0.0893 minpoints:30	0.333	0.734	0.258	3	6343	EMBSIZE:200_SCALER:standard
eps:0.002308 minpoints: 4	0.258	**0.767**	-0.323	252	9525	EMBSIZE:5_SCALER:minmax
eps:0.0079 minpoints:20	0.286	0.680	**0.511**	3	3999	EMBSIZE:5_SCALER:standard

Vettori con secondo embedding e distanza del coseno						
Parameters set	ARI	RI	SILHOUETTE	NUM_CLUSTER	NUM_OUTLIERS	
eps:0.101 minpoints: 30	0.314	0.736	0.174	5	7757	EMBSIZE:800 SCALER:standard
eps:0.076 minpoints:12	0.288	0.756	0.071	39	4398	EMBSIZE:40 SCALER:standard
eps:0.042 minpoints: 20	0.267	0.699	0.668	4	2173	EMBSIZE:200 SCALER:minmax

Comparazione con altri paper

Confrontiamo ora i risultati ottenuti nei due approcci presentati nel nostro lavoro di tesi e i lavori presentati in [11] e in [12]. In questi lavori vengono utilizzate rispettivamente reti neurali con layer bidirezionali e soli strati fully connected per l'embedding delle MTS e Kmeans per il clustering. Implementazioni di tali approcci sono presenti agli URLs github:

https://github.com/FlorentF9/DeepTemporalClustering (BI-ANN)

GitHub - XifengGuo/DEC-keras: Keras implementation for Deep Embedding Clustering (DEC)

Misura	*Sig2vec*	*CNN - Conc*	*CNN–re-emb*	*Bi-ANN*	*DEC*
ARI	**0.49**	0.35	0.38	0.13	0.34
RI	**0.89**	0.83	0.85	0.76	0.84
Silhouette	0.013	0.37	0.27	**0.91**	0.78

Tabella 17 Risultati migliori per ogni metodologia

Come notiamo il nostro metodo Sig2Vec supera in termini di ARI e RI tutti gli altri approcci sul dataset utilizzato.

Mostriamo nelle figure successive, per ogni approccio, la confusion matrix, la pair confusion matrix e il grafico della silhouette per ogni cluster rispetto ai modelli che hanno raggiunto i valori di ARI, RI e silhouette indicati nella tabella appena presentata.

Notiamo come gli approcci da noi proposti distribuiscano i dati su tutte i cluster e non etichettando i dati solo su un sottoinsieme di gruppi o peggio in un unico gruppo, favorendo l'ARI.

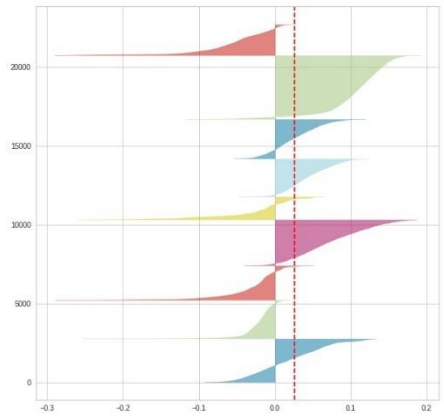


Figura 44 Silhouette per cluster per Sig2vec

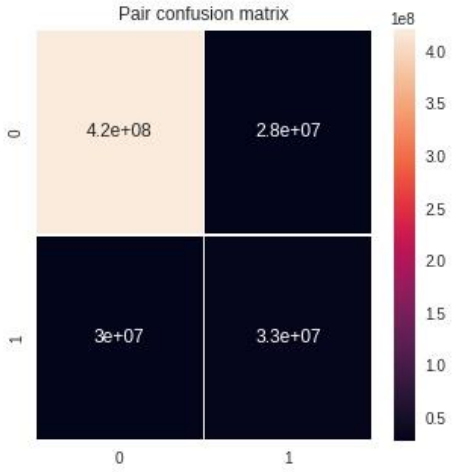


Figura 45 Pair confusion matrix per Sig2vec

Figura 46 Confusion matrix per Sig2vec. Predicted label sull'asse X

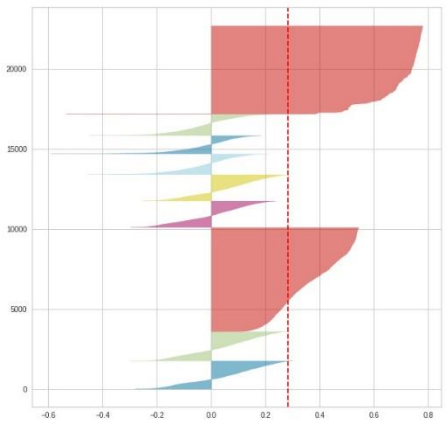


Figura 47 Silhouette per cluster per CNN con vettori concatenati

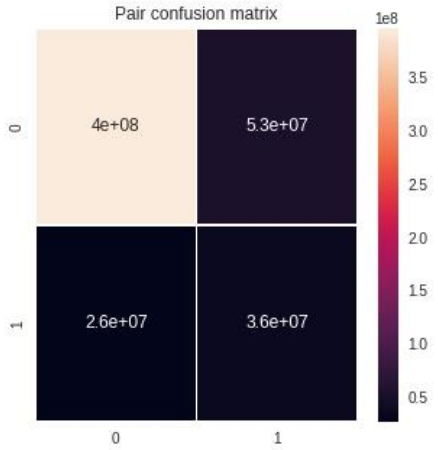


Figura 48 Pair confusion matrix per CNN con vettori concatenati

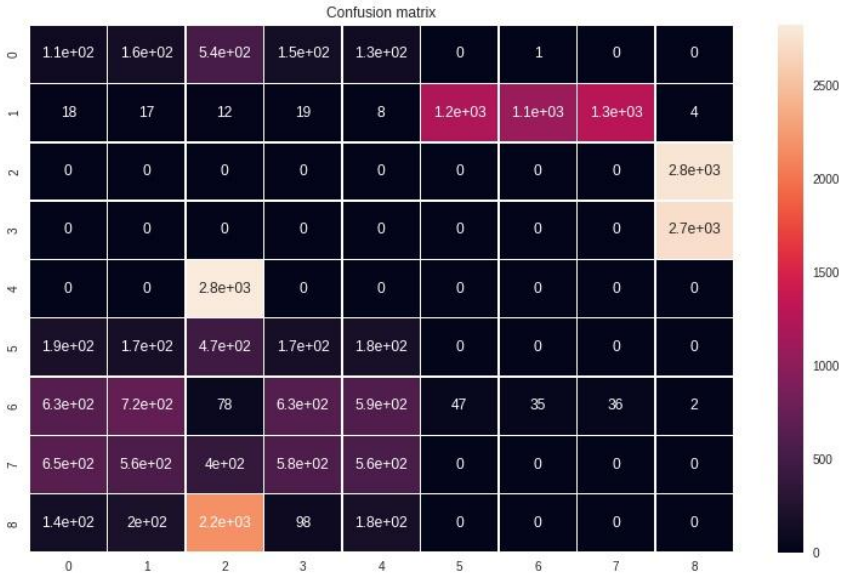


Figura 49 Confusion matrix per CNN con vettori concatenati. Predicted label sull'asse X

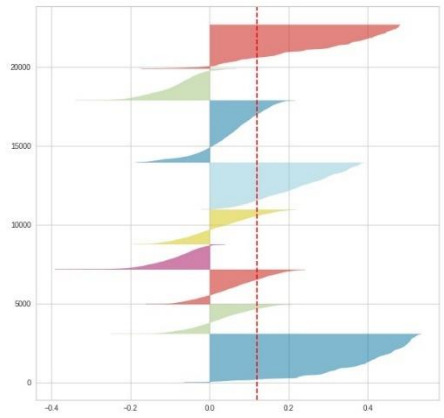


Figura 50 Silhouette per cluster per CNN con secondo embedding

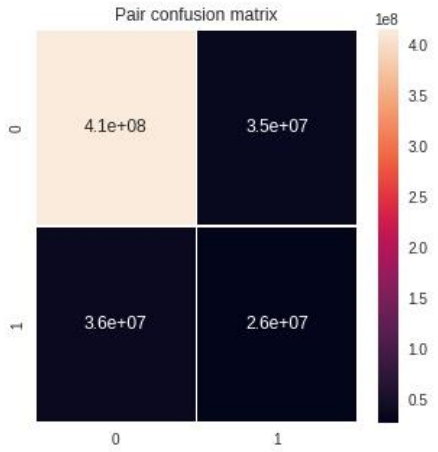


Figura 51 Confusion matrix per CNN con secondo embedding

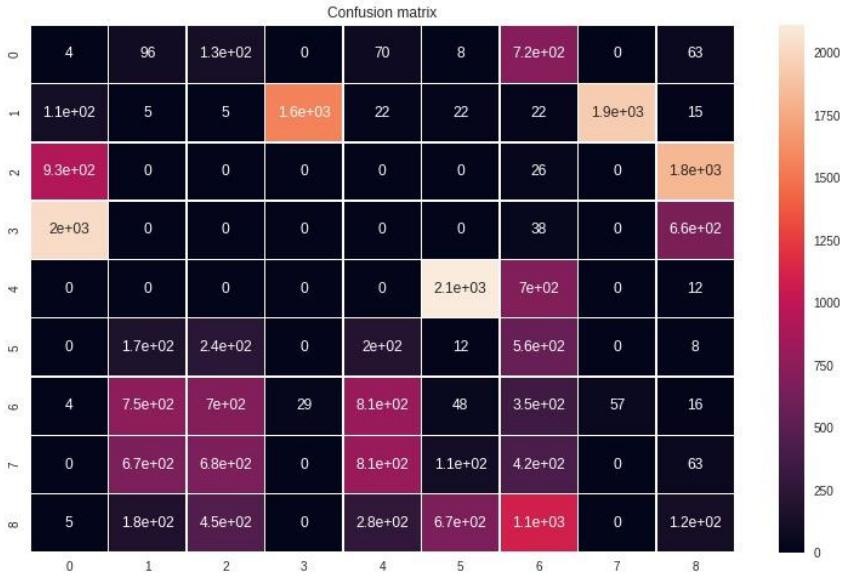
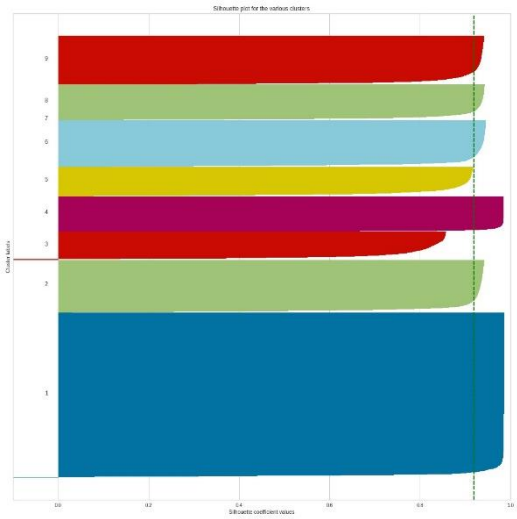


Figura 52 Pair confusion matrix per CNN con secondo embedding, Predicted label sull'asse X



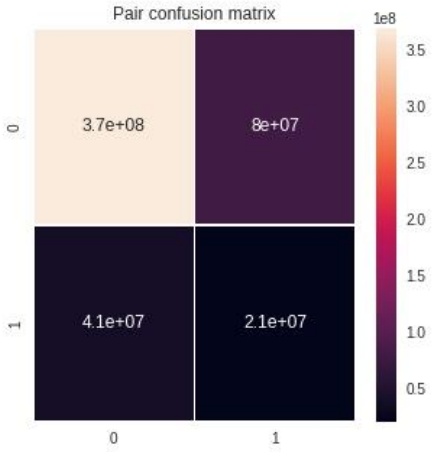


Figura 53 Pair confusion matrix per ANN con layer bidirezionali

Figura 54 Silhouette per ANN con layer bidirezionali

Figura 55 Confusion matrix per ANN con layer bidirezionali. Predicted label sull'asse X

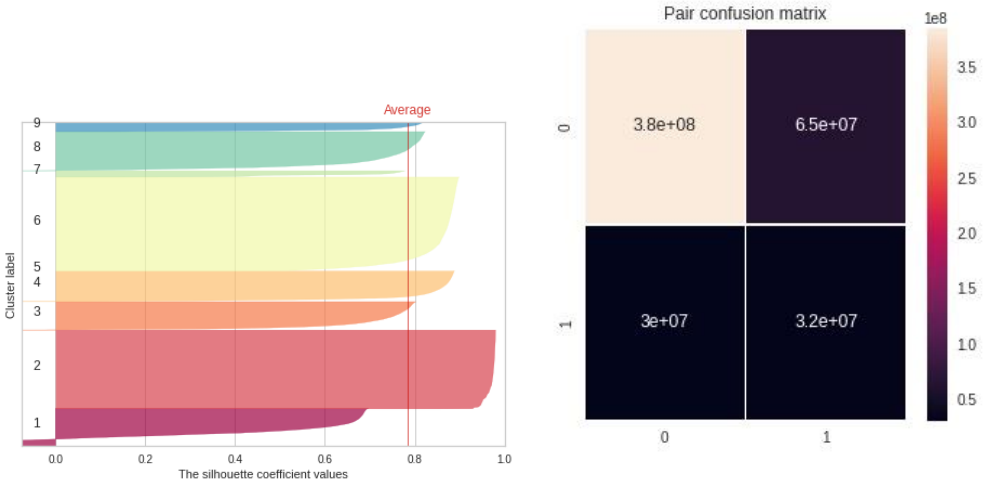


Figura 56 Silhouette per ANN con con soli fully connected layer

Figura 57 Pair confusion matrix per ANN con con soli fully connected layer

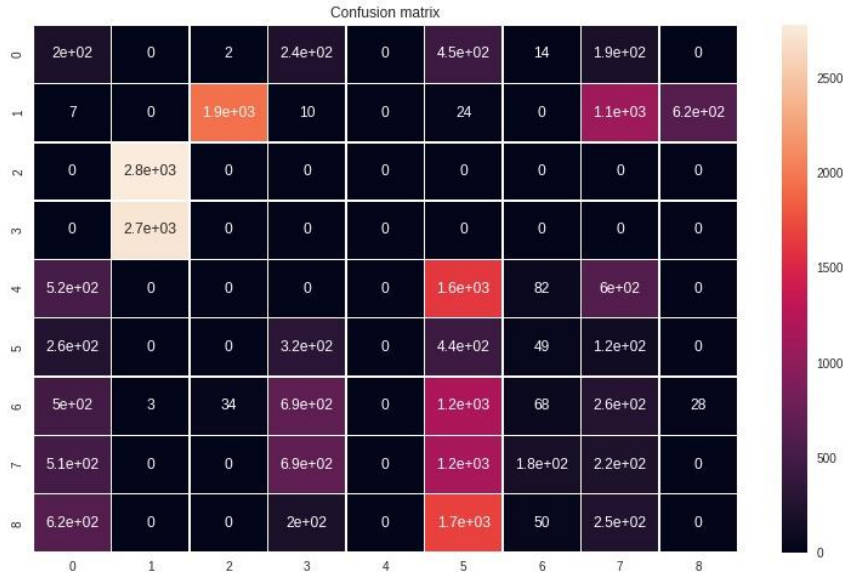


Figura 58 Pair confusion matrix per ANN con con soli fully connected layer. Predicted label sull'asse X

Conclusioni e sviluppi futuri

Entrambi gli approcci mostrano risultati soddisfacenti per le metriche random index e adjusted random index sia utilizzando la distanza euclidea che utilizzando la distanza del coseno e risultano paragonabili o addirittura migliori rispetto agli approcci presentati negli altri paper.

Gli algoritmi partizionali come il k-means e il k-medoids ottengono risultati mediamente migliori per le metriche sopracitate, ma spesso a discapito della silhouette che, tuttavia rimane una metrica meno significativa in termini di validazione dei modelli rispetto alle altre due in quanto basata soltanto su informazioni interne al dataset e non anche su label, come ARI e RI.

Si noti che i risultati migliori si ottengono con features vector relativamente piccoli e ciò determina un forte impatto positivo sulle prestazioni degli algoritmi di clustering che devono in questo modo computare le distanze su relativamente poche features.

Sia per Sig2vec che per il metodo basato su reti neurali convoluzionali giungiamo all'incirca alle stesse conclusioni sia in termini di grandezza dei features vectors che di risultati massimi per ogni metrica. C'è da sottolineare che, tuttavia, le architetture basate su CNN presentano una complessità computazionale maggiore rispetto a Sig2vec in quanto più profonde in numero di layers e conseguentemente presentano un numero maggiore di parametri liberi da addestrare.

In entrambi gli approcci i metodi basati su agglomerative clustering e in particolare l'implementazione con minimum linkage non performano bene tanto quanto gli altri algoritmi mentre DBSCAN non riesce a generare il numero corretto di cluster col dataset selezionato seppur in alcuni casi riesca a raggiungere punteggi accettabili per le metriche di riferimento.

Possiamo concludere che gli studi condotti su questo dataset abbiano mostrato risultati estremamente incoraggianti per entrambi gli approcci ed in particolare Sig2vec sembra performare meglio di approcci sensibilmente più complessi come quelli basati su CNN e reti neurali con layers bidirezionali.

Pertanto, modelli di clustering più evoluti o una selezione di iper-parametri più fine applicati a embedding vectors generati da Sig2vec potrebbero ulteriormente migliorare i risultati ottenuti in questo lavoro di tesi.

Eventuali approfondimenti delle metodologie sviluppate in questo progetto di tesi potrebbero seguire questa direzione e prevedere la valutazione di questi approcci su un numero più ampio di dataset in modo da avere una panoramica generale sugli effettivi punti di forza e/o debolezza di tali approcci. Si potrebbe anche studiare ulteriormente la "qualità" dell'embedding di questi approcci utilizzando algoritmi supervised, più elaborati ed accurati in modo da valutare quanta informazione effettivamente riescano a sintetizzare questi approcci sfruttando la conoscenza delle label anche in fase di training e non solo di validazione.

Bibliografia

[1] Q. Le and T. Mikolov, “Distributed representations of sentences and documents,” *31st Int. Conf. Mach. Learn. ICML 2014*, vol. 4, pp. 2931–2939, 2014.

[2] T. Warren Liao, “Clustering of time series data - A survey,” *Pattern Recognit.*, vol. 38, no. 11, pp. 1857–1874, 2005, doi: 10.1016/j.patcog.2005.01.025.

[3] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah, “Time series clustering - A decade review,” *Inf. Syst.*, vol. 53, pp. 16–38, 2015, doi: 10.1016/j.is.2015.04.007.

[4] J. Paparrizos and L. Gravano, “K-shape: Efficient and accurate clustering of time series,” *Proc. ACM SIGMOD Int. Conf. Manag. Data*, vol. 2015-May, pp. 1855–1870, 2015, doi: 10.1145/2723372.2737793.

[5] M. Shokoohi-Yekta, J. Wang, and E. Keogh, “On the non-trivial generalization of Dynamic Time Warping to the multi-dimensional case,” *SIAM Int. Conf. Data Min. 2015, SDM 2015*, pp. 289–297, 2015, doi: 10.1137/1.9781611974010.33.

[6] L. N. Ferreira and L. Zhao, “Time series clustering via community detection in networks,” *Inf. Sci. (Ny).*, vol. 326, pp. 227–242, 2016, doi: 10.1016/j.ins.2015.07.046.

[7] H. Li, “Multivariate time series clustering based on common principal component analysis,” *Neurocomputing*, vol. 349, pp. 239–247, Jul. 2019, doi: 10.1016/j.neucom.2019.03.060.

[8] H. Li and T. Du, “Multivariate time series clustering based on component relationship networks,” *Expert Syst. Appl.*, vol. 173, p. 114649, Jul. 2021, doi: 10.1016/j.eswa.2021.114649.

[9] H. Li and Z. Liu, “Multivariate time series clustering based on complex network,” *Pattern Recognit.*, vol. 115, p. 107919, Jul. 2021, doi: 10.1016/j.patcog.2021.107919.

[10] Cherif, H. Cardot, and R. Boné, “SOM time series clustering and prediction with recurrent neural networks,” *Neurocomputing*, vol. 74, no. 11, pp. 1936–1944, 2011, doi: 10.1016/j.neucom.2010.11.026.

[11] Kim, J., Moon, N. A deep bidirectional similarity learning model using dimensional reduction for multivariate time series clustering. *Multimed Tools Appl* **80,** 34269–34281 (2021). https://doi.org/10.1007/s11042-020-10476-6

[12] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. ICML 2016

[13] J. D. Banfield, and A. E. Raftery, “Model-based Gaussian and non-Gaussian clustering,” Biometrics, 1993

[14] K. Kalpakis, D. Gada, V. Puttagunta, Distance measures for effective clustering of ARIMA time-series, in: Proceedings 2001 IEEE Interna- tional Conference on Data Mining, 2001, pp. 273–280.

[15] Ghassempour, S., Girosi, F., & Maeder, A. (2014). Clustering multivariate time series using Hidden Markov Models. *International Journal of Environmental Research and Public Health*, *11*(3), 2741–2763. https://doi.org/10.3390/ijerph110302741

[16] Martino, A., Guatteri, G., & Paganoni, A. M. (2020). Hidden Markov Models for multivariate functional data. *Statistics and Probability Letters*, *167*, 108917. https://doi.org/10.1016/j.spl.2020.108917

[17] D. J. Berndt and J. Clifford, “Using dynamic time warping to find patterns in time series.” in KDD workshop, vol. 10, no. 16. Seattle, WA, 1994, pp. 359–370.

[18] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[19] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[20] Zeiler M.D., Fergus R. (2014) Visualizing and Understanding Convolutional Networks. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8689. Springer, Cham. https://doi.org/10.1007/978-3-319-10590-1_53