

# Progetto A\_Clus - Documentazione

A cura di: Vito Stefano  
Lorenzo Gelao

4 aprile 2025

## Indice

<b>1</b>	<b>Introduzione</b>	<b>4</b>
1.1	Agglomerative Clustering . . . . .	4
1.1.1	Framework Analitico . . . . .	4
<b>2</b>	<b>Guida all'installazione</b>	<b>5</b>
2.1	Installazione JDK . . . . .	5
2.2	Variabili di ambiente . . . . .	5
2.3	Installazione MySQL . . . . .	6
2.4	Schermata iniziale . . . . .	6
2.5	Download dei file . . . . .	7
2.6	Configurazione del account MySQL . . . . .	7
2.7	Schermata finale . . . . .	8
<b>3</b>	<b>Eseguire A-CLus Base</b>	<b>9</b>
3.1	Operazioni preliminari . . . . .	9
3.2	Avvio dell'applicazione . . . . .	9
3.3	Avvio dell'ambiente di sviluppo . . . . .	10
3.4	Interfaccia iniziale . . . . .	10
3.5	Selezione delle operazioni disponibili . . . . .	11
3.6	Generazione nuovo dendrogramma . . . . .	11
3.7	Inserimento Profondità . . . . .	11
3.7.1	Elaborazione con Single-link . . . . .	12
3.8	Inserimento nome file . . . . .	12
3.8.1	Elaborazione con Average-link . . . . .	13
3.9	Caricamento dendrogramma esistente . . . . .	13
3.10	Inserimento nome archivio con estensione . . . . .	13
<b>4</b>	<b>Test A-CLus Base</b>	<b>14</b>
4.1	Gestione di tabelle inesistenti . . . . .	14
4.2	Validazione delle selezioni di menù . . . . .	14
4.2.1	Controllo sull'esistenza degli archivi . . . . .	15
4.2.2	Validazione parametri di profondità . . . . .	15
4.2.3	Controllo selezioni metodologiche . . . . .	15
4.2.4	Verifica di compatibilità dimensionale . . . . .	16
4.3	Gestione connettività client-server . . . . .	16
4.3.1	Elaborazione su dataset vuoti . . . . .	17
<b>5</b>	<b>Modifiche Implementative</b>	<b>17</b>
5.1	Adattamenti nella Versione Base . . . . .	17
5.1.1	Modifiche ai Requisiti Originali . . . . .	17
5.2	Evoluzione Architetturale nella Versione Estesa . . . . .	17
5.2.1	Riprogettazione dello State Management . . . . .	17
5.2.2	Modello di Gestione Errori Rafforzato . . . . .	18
5.2.3	Architettura Client-Server Ottimizzata . . . . .	18
<b>6</b>	<b>Eseguire A-CLus Esteso</b>	<b>19</b>
6.1	Avvio dall'ambiente di sviluppo . . . . .	19
6.2	Guida operativa al bot Telegram . . . . .	19
6.3	Test A-CLUS ESTESO . . . . .	21

<b>7</b>	<b>Cambiamenti</b>	<b>21</b>
7.1	Cambiamenti Motivati nella Versione Base Rispetto i Requisiti del Progetto . . . . .	21
7.2	Cambiamenti Motivati nella Versione Estesa Rispetto alla Base .	22
<b>8</b>	<b>Macchina a stati</b>	<b>24</b>

## 1 Introduzione

Il progetto in questione verte sull'argomento dell'*Agglomerative Clustering*, una tecnica di clusterizzazione basata sui metodi Single distance e Average Distance.

Il progetto in questione è suddiviso in una parte client e una server, che comunicando tra loro, generano il dendrogramma, permettendo inoltre di visualizzare e memorizzare tali risultati o di caricarne dei precedenti.

È inoltre possibile visualizzare nuovamente dei file caricati in passato per visualizzare i cluster e i dendrogrammi associati.

### 1.1 Agglomerative Clustering

L'algoritmo di clustering utilizzato in tale progetto, come si può desumere dal nome, sfrutta il concetto di clustering agglomerativo. In pratica, tratta ciascun cluster in maniera separata dagli altri, unendo progressivamente quelli più vicini, in base a due criteri principali, nel nostro caso.

Il principale vantaggio rispetto ad altri algoritmi di clustering, come il k-means, è che in questo modo non è necessario specificare in anticipo la quantità di cluster da analizzare.

I criteri utilizzati nel progetto A-CLus sono i seguenti:

1. **Single-Link:** tale criterio determina la distanza minima tra i punti dei vari cluster

$$D(C1, C2) = \min_{(t1 \in C1, t2 \in C2)} (dist(t1, t2)) \quad (1)$$

Durante l'anno accademico 2024/2025, l'oggetto di ricerca è stato incentrato su "A-CLus", una piattaforma con architettura client-server dedicata all'analisi dei dati mediante algoritmi di clustering gerarchico agglomerativo. La componente server esegue le operazioni di clustering impiegando le metodologie Single Link o Average Link per il calcolo delle distanze inter-cluster e la successiva costruzione del dendrogramma. L'applicativo client, implementato in linguaggio Java, offre agli utenti diverse funzionalità: il caricamento o la creazione di istanze HierarchicalClusterMiner, la rappresentazione grafica del dendrogramma e l'archiviazione dei risultati per analisi successive. È inoltre disponibile la funzione di importazione di file precedentemente salvati, consentendo agli utenti di riesaminare i cluster e i relativi dendrogrammi.

#### 1.1.1 Framework Analitico

La tecnica di clustering agglomerativo implementata nel sistema "A-CLus" costituisce una metodologia avanzata per l'identificazione di correlazioni latenti nei dataset.

Diversamente da tecniche alternative come il k-means, l'approccio agglomerativo elimina la necessità di predefinire il numero di raggruppamenti. La procedura inizializza ciascun elemento come cluster individuale e procede con l'unificazione sequenziale dei cluster con maggiore affinità, applicando metodologie quali:

Con il procedere delle aggregazioni tra cluster, si sviluppa un dendrogramma che illustra la gerarchia delle aggregazioni. Il procedimento continua fino al raggiungimento di un cluster unificato o fino a una soglia di profondità stabilita dall'utilizzatore. La visualizzazione mediante dendrogramma rende la metodologia particolarmente comprensibile, facilitando l'esplorazione strutturale dei dati a diversi livelli di granularità. Inoltre, questa strategia non è influenzata dalla configurazione iniziale dei centroidi, riducendo così la probabilità di risultati inconsistenti e garantendo una rappresentazione più accurata dell'organizzazione interna del dataset. RiprovaClaude può commettere errori. Verifica sempre le risposte con attenzione.

## 2 Guida all'installazione

Prima di essere in grado di eseguire il programma, è necessario eseguire il file `risorse.bat` contenuto nella cartella Risorse.

Una volta eseguito il file, si aprirà una pagina di Powershell e seguirà un download.



Figura 1: Schermata di download

Al termine del download del file compresso delle risorse, verranno estratti i file necessari per l'esecuzione del programma.

Per il progetto in questione è necessario installare il Java Developer Kit (JDK) nella versione 22.0.1 e il software di gestione del database MySQL nella sua versione 8.0.39.

La prima scheda di installazione che apparirà è quella del JDK.

Verrà chiesto se si vuole eseguire l'installazione del JDK tramite permessi di amministratore, cliccare su Sì.

### 2.1 Installazione JDK

Una volta confermata l'esecuzione come amministratore, si procederà all'installazione del JDK.

Cliccare su **Avanti**, nuovamente **Avanti** e infine, nel caso in cui l'installazione sia andata a buon fine, si dovrà cliccare sul tasto **Chiudi**.

### 2.2 Variabili di ambiente

è consigliato creare una variabile di ambiente per il JDK. Per farlo, si dovrà cercare la voce **variabili di ambiente** nella barra di ricerca di Windows e cliccare su Modifica le variabili di ambiente per il tuo account.

Successivamente si dovrà cliccare sul pulsante **Variabili di ambiente** e si aprirà una schermata con le variabili di ambiente.

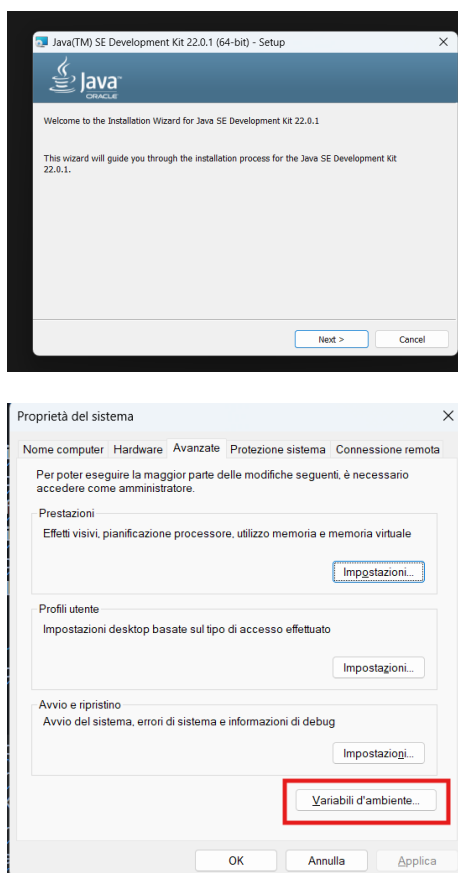


Figura 2: Schermata delle variabili di ambiente

Nella schermata, si dovrà cliccare sulla variabile **PATH** e modificarla in modo da aggiungere il percorso del JDK, inserendo il percorso in cui è stato installato il JDK.

Una volta completata l'installazione del JDK, si procederà automaticamente con l'installazione del MySQL.

## 2.3 Installazione MySQL

Anche qui sarà necessario fornire i permessi di amministratore, cliccare dunque su Sì.

## 2.4 Schermata iniziale

Dopodiché si aprirà la schermata di installazione di MySQL.

Selezionare il tipo di setup **Full** e cliccare su **Next**. Tale scelta permetterà di installare tutti i componenti necessari per l'esecuzione del programma.

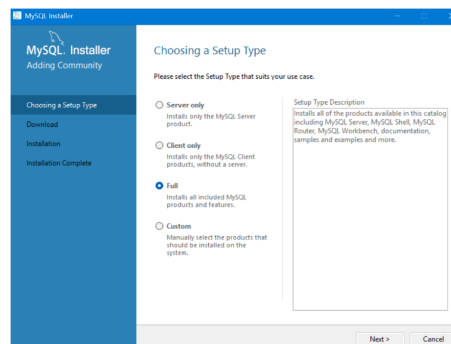


Figura 3: Scelta del setup

## 2.5 Download dei file

Si passerà alla schermata di download dei file necessari per l'installazione, cliccare su **Execute**.

### Avvertenza

Tale schermata potrebbe richiedere un po' di tempo per il download dei file, a seconda della velocità della connessione internet.

Dopo aver scaricato i file, si dovrà cliccare nuovamente su **Execute**.

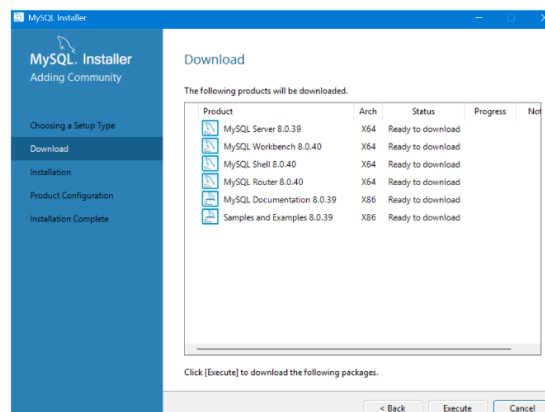


Figura 4: Download MySQL

## 2.6 Configurazione del account MySQL

Una volta terminato il download, si procederà con la configurazione dell'account MySQL. Si dovrà quindi scegliere una password, necessaria per accedere al database.

Si giungerà infine alla schermata di applicazione della configurazione del database. Cliccare su **Execute** per applicare la configurazione.

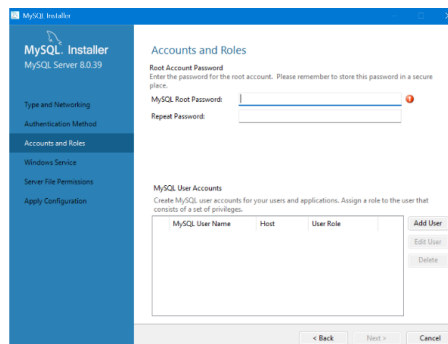


Figura 5: Configurazione account MySQL

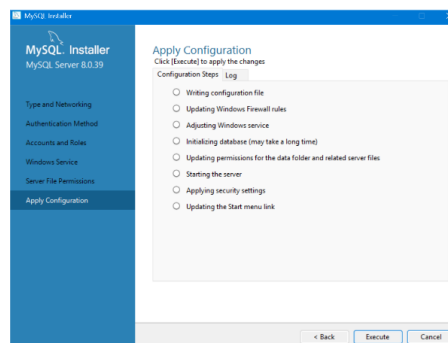


Figura 6: Completamento installazione

Una volta terminata anche l'installazione del MySQL, si dovrà cliccare su **Finish** per chiudere l'installler.

#### Avvertenza

Per un corretto funzionamento sia del JDK che del MySQL, è consigliato riavviare il computer.

## 2.7 Schermata finale

Al termine dell'installazione il programma `resources.bat` mostrerà la seguente schermata.



Figura 7: Completamento installazione



## 3 Eseguire A-CLus Base

### 3.1 Operazioni preliminari

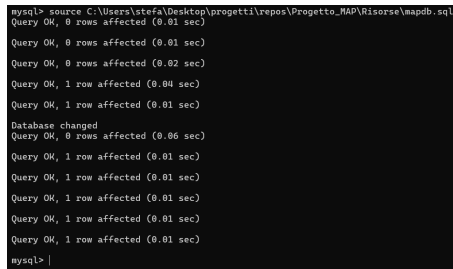
Prima di poter eseguire il programma, occorre importare il database di A-CLus in MySQL.

Tale file è presente nella cartella **A-CLus/Risorse** e si chiama **mapdb.sql**.

Per importare il database, è necessario semplicemente accedere al MySQL tramite root e digitare il seguente comando **source** seguito dal percorso del file **mapdb.sql**.

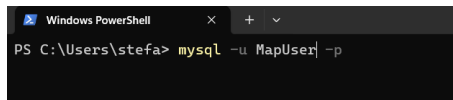
Tale percorso si può ottenere semplicemente trascinando il file del database sul terminale di MySQL.

Alle fine dell'importazione, il terminale di MySQL dovrebbe mostrare un messaggio simile al seguente:



```
mysql> source C:\Users\stefa\Desktop\progetti\repos\Progetto_MAP\Risorse\mapdb.sql
Query OK, 0 rows affected (0.01 sec)
Query OK, 0 rows affected (0.02 sec)
Query OK, 1 row affected (0.04 sec)
Query OK, 1 row affected (0.01 sec)
Database changed
Query OK, 0 rows affected (0.06 sec)
Query OK, 1 row affected (0.01 sec)
Query OK, 1 row affected (0.01 sec)
Query OK, 1 row affected (0.01 sec)
Query OK, 1 row affected (0.01 sec)
Query OK, 1 row affected (0.01 sec)
mysql>
```

È possibile verificare la corretta importazione anche uscendo da MySQL e accedere con le credenziali dell'utente MapUser, come mostrato nella figura seguente:



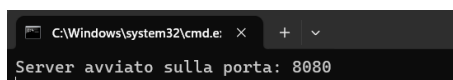
```
Windows PowerShell
PS C:\Users\stefa> mysql -u MapUser -p
```

La password dell'utente MapUser è **map**. Se viene confermato l'accesso al database, significa che l'importazione è avvenuta con successo.

### 3.2 Avvio dell'applicazione

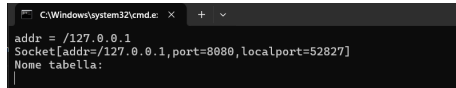
Per inizializzare correttamente la versione standard di A-CLus, attenersi alla seguente sequenza operativa:

1. Accedere alla directory **A-CLus\_Base/Bat**
2. Eseguire il file **Start\_Server.bat** facendo un doppio click su di esso, assicurandosi che venga mostrato il seguente messaggio a finestra



```
C:\Windows\system32\cmd.e
Server avviato sulla porta: 8080
```

3. Dopo aver verificato il corretto funzionamento del server, eseguire il file `Start_Client.bat` facendo un doppio click su di esso; se non ci sono problemi con il server, apparirà la seguente schermata:



```
C:\Windows\system32\cmd.e: x + v
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=52827]
Nome tabella:
```

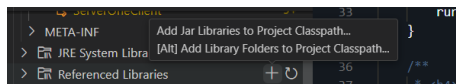
#### Avvertenza

È necessario mantenere attivo il terminale del server per garantire la comunicazione tra le componenti client e server.

### 3.3 Avvio dell'ambiente di sviluppo

Per aprire correttamente il codice sorgente di A-CLus, è necessario seguire le seguenti operazioni nell'ordine in cui sono presentate:

1. Aprire la cartella del progetto `A-CLus_base`
2. Incorporare nelle librerie del progetto `Server` il file `mysql_connector.jar` presente nella cartella `A-CLus/Risorse`. Nel caso si stia eseguendo il progetto in Visual Studio Code, è possibile eseguire i seguenti passaggi:
  - (a) Aprire il progetto `A-CLus_Base`
  - (b) Recarsi nella sezione `Referenced Libraries`, in asso a sinistra
  - (c) Posizionare il cursore sulla cartella e selezionare `Add JAR/Folder Classpath`, che apparirà sulla destra



3. Avviare `MultiServer.java`
4. Controllare che il server sia in esecuzione correttamente
5. Aprire il progetto `MainTest.java` presente nella cartella `A-CLus_Base/Client`

### 3.4 Interfaccia iniziale

L'interfaccia iniziale di A-CLus è composta da tre sezioni principali:

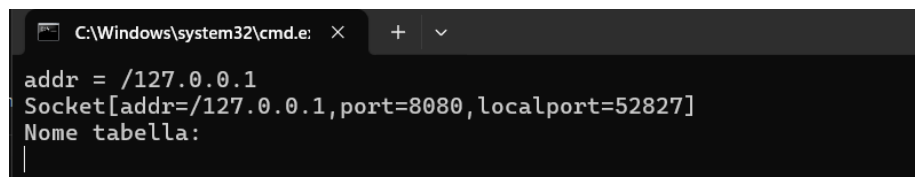
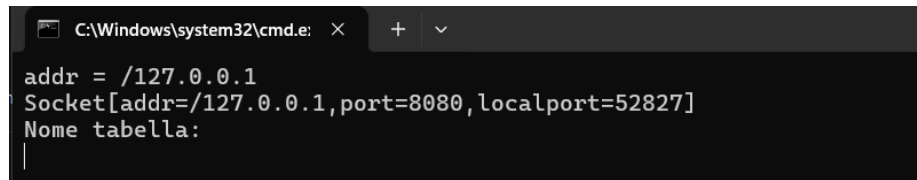


Figura 8: Interfaccia iniziale di A-CLus

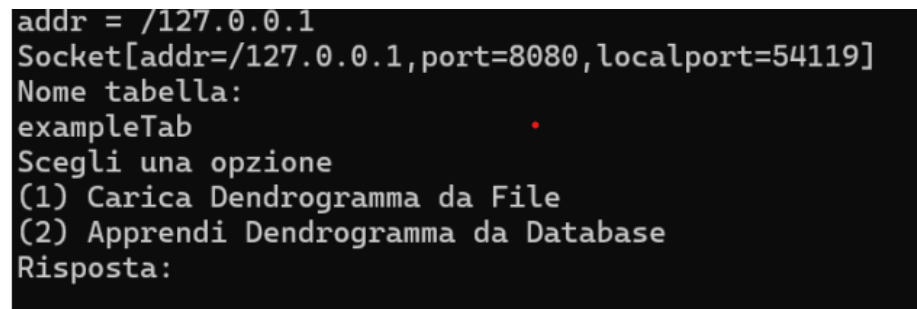
L'interfaccia iniziale di A-CLus è composta da tre sezioni principali:



```
C:\Windows\system32\cmd.exe
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=52827]
Nome tabella:
|
```

Figura 9: Interfaccia iniziale di A-CLus

### 3.5 Selezione delle operazioni disponibili

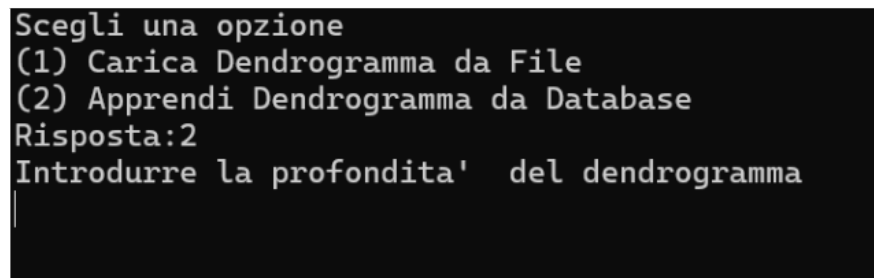


```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=54119]
Nome tabella:
exampleTab
Scegli una opzione
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:
|
```

Dopo l'inserimento della tabella, verrà proposto un menu con due alternative:

1. **Carica Dendrogramma da File:** permette di importare un dendrogramma precedentemente memorizzato
2. **Apprendi Dendrogramma da Database:** consente di generare un nuovo dendrogramma analizzando i dati presenti nella tabella selezionata

### 3.6 Generazione nuovo dendrogramma



```
Scegli una opzione
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:2
Introdurre la profondita' del dendrogramma
|
```

Selezionando l'opzione 2, il sistema richiederà l'inserimento del parametro di profondità desiderato per il dendrogramma.

### 3.7 Inserimento Profondità

Successivamente, l'applicazione proporrà due metodologie di calcolo alternative:

```
Introdurre la profondita' del dendrogramma
3
Distanza: single-link (1), average-link (2):
```

1. **Single-link**: identifica la distanza minima tra cluster, connettendo gli elementi più prossimi tra i gruppi
2. **Average-link**: determina la distanza utilizzando la media tra tutti gli elementi dei cluster considerati

### 3.7.1 Elaborazione con Single-link

```
Distanza: single-link (1), average-link (2):
1
level0:
cluster0:<[1.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]>
cluster3:<[1.0,3.0,4.0]>
cluster4:<[2.0,2.0,0.0]>

level1:
cluster0:<[1.0,2.0,0.0]><[2.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]>
cluster3:<[1.0,3.0,4.0]>

level2:
cluster0:<[1.0,2.0,0.0]><[2.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]><[1.0,3.0,4.0]>

Inserire il nome dell'archivio (comprensivo di estensione):
```

Selezionando l'opzione Single-link (1), verrà elaborato e visualizzato il dendrogramma risultante. L'applicativo richiederà poi di specificare il nome del file per l'archiviazione del dendrogramma, completo di estensione.

## 3.8 Inserimento nome file

```
Inserire il nome dell'archivio (comprensivo di estensione):
Example1.dat
Dendrogramma salvato correttamente.
Premere un tasto per continuare . . . |
```

Indicare il nome desiderato (esempio: `Example1.dat`). Il dendrogramma verrà salvato nella cartella "**saved**" all'interno della directory "**Jar + Bat**" e l'applicazione terminerà l'esecuzione.

### 3.8.1 Elaborazione con Average-link

```
Distanza: single-link (1), average-link (2):
2
level0:
cluster0:<[1.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]>
cluster3:<[1.0,3.0,4.0]>
cluster4:<[2.0,2.0,0.0]>

level1:
cluster0:<[1.0,2.0,0.0]><[2.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]>
cluster3:<[1.0,3.0,4.0]>

level2:
cluster0:<[1.0,2.0,0.0]><[2.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]><[1.0,3.0,4.0]>

Inserire il nome dell'archivio (comprensivo di estensione):
|
```

Selezionando l'opzione Average-link (2), il sistema elaborerà il dendrogramma utilizzando il criterio della distanza media e lo visualizzerà a schermo. Successivamente, verrà richiesto di specificare il nome del file per la memorizzazione del dendrogramma, completo di estensione.

## 3.9 Caricamento dendrogramma esistente

```
Scegli una opzione
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:1
Inserire il nome dell'archivio (comprensivo di estensione):
```

Selezionando l'opzione 1 per importare un dendrogramma preesistente, sarà necessario indicare il nome completo del file archivio, comprensivo di estensione. I file precedentemente salvati sono localizzati nella cartella "saved" all'interno della directory "Jar + Bat".

### 3.10 Inserimento nome archivio con estensione

Dopo l'inserimento del nome dell'archivio, il sistema visualizzerà il dendrogramma caricato e concluderà l'esecuzione.

```
Scegli una opzione
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:1
Inserire il nome dell'archivio (comprensivo di estensione):
Example1.dat
level0:
cluster0:<[1.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]>
cluster3:<[1.0,3.0,4.0]>
cluster4:<[2.0,2.0,0.0]>

level1:
cluster0:<[1.0,2.0,0.0]><[2.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]>
cluster3:<[1.0,3.0,4.0]>

level2:
cluster0:<[1.0,2.0,0.0]><[2.0,2.0,0.0]>
cluster1:<[0.0,1.0,-1.0]>
cluster2:<[1.0,3.0,5.0]><[1.0,3.0,4.0]>

Premere un tasto per continuare . . .
```

## 4 Test A-CLus Base

Durante la fase di testing, sono stati analizzati diversi scenari critici per validare la robustezza dell'applicazione:

### 4.1 Gestione di tabelle inesistenti

```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=54675]
Nome tabella:
example
Errore nel recupero dei dati: Table 'mapdb.example' doesn't exist
Nome tabella:
```

Quando viene specificato un identificativo di tabella non presente nel database, il sistema visualizza un appropriato messaggio diagnostico e offre all'utente la possibilità di inserire una denominazione alternativa.

### 4.2 Validazione delle selezioni di menù

La schermata di selezione del menù principale consente di scegliere tra due opzioni: "Carica Dendrogramma" o "Esegui Clustering".

```
Scegli una opzione
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:3
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:
```

L'inserimento di valori non conformi alle opzioni disponibili (diversi da 1 o 2) viene intercettato, consentendo all'utente di ripetere la selezione senza interruzioni del flusso operativo.

#### 4.2.1 Controllo sull'esistenza degli archivi

```
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:1
Inserire il nome dell'archivio (comprensivo di estensione):
example.dat
File non trovato: example.dat
Premere un tasto per continuare . . .
```

Nel caso di caricamento di un dendrogramma precedentemente salvato, il sistema verifica l'esistenza dell'archivio specificato. In caso negativo, viene presentata una notifica e l'applicazione termina la sessione.

#### 4.2.2 Validazione parametri di profondità

```
(2) Apprendi Dendrogramma da Database
Risposta:2
Introdurre la profondita' del dendrogramma
0
Distanza: single-link (1), average-link (2):
1
E' stato inserito un valore <=0 per la profondita' del Dendrogramma.
Premere un tasto per continuare . . .
```

L'inserimento di valori non ammissibili per il parametro di profondità (zero o negativi) viene rilevato dal sistema che, pur proseguendo con la richiesta del metodo di calcolo, interromperà l'elaborazione notificando l'anomalia.

#### 4.2.3 Controllo selezioni metodologiche

L'inserimento di opzioni non valide per la selezione della metodologia di calcolo delle distanze viene gestito permettendo all'utente di riformulare la propria scelta.

```
(2) Apprendi Dendrogramma da Database
Risposta:2
Introdurre la profondita' del dendrogramma
-2
Distanza: single-link (1), average-link (2):
2
E' stato inserito un valore <=0 per la profondita' del Dendrogramma.
Premere un tasto per continuare . . .
```

```
Distanza: single-link (1), average-link (2):
3
Distanza: single-link (1), average-link (2):
```

#### 4.2.4 Verifica di compatibilità dimensionale

```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=54135]
Nome tabella:
exampleTab1
Scegli una opzione
(1) Carica Dendrogramma da File
(2) Apprendi Dendrogramma da Database
Risposta:1
Inserire il nome dell'archivio (comprensivo di estensione):
esempio111.dat
Il numero di esempi nella tabella scelta e' minore del numero di esempi con cui e' stato salvato l'oggetto h-clus
precedentemente serializzato.
Premere un tasto per continuare . . .
```

Il sistema controlla la congruenza tra la cardinalità degli esempi nella tabella corrente e quella memorizzata nell'oggetto serializzato. In caso di incongruenza, viene visualizzato un messaggio esplicativo e l'esecuzione viene terminata.

#### 4.3 Gestione connettività client-server

```
C:\WINDOWS\system32\cmd. × + v
addr = /127.0.0.1
Connection refused: connect
Premere un tasto per continuare . . . |
```

L'avvio del client in assenza del componente server attivo genera un tentativo di connessione all'indirizzo locale (127.0.0.1) che, non trovando un endpoint disponibile, risulta in un rifiuto della connessione con relativa notifica.



```
addr = /127.0.0.1
Socket[addr=/127.0.0.1,port=8080,localport=52441]
Nome tabella:
exampleTab1
Errore nel recupero dei dati: La tabella exampleTab1 è vuota.
Nome tabella:
|
```

#### 4.3.1 Elaborazione su dataset vuoti

Quando l'utente tenta di eseguire un'analisi di clustering su una tabella priva di contenuti, il sistema identifica la condizione e fornisce un'appropriata segnalazione.

## 5 Modifiche Implementative

### 5.1 Adattamenti nella Versione Base

#### 5.1.1 Modifiche ai Requisiti Originali

- **Metodo `getLength()` in `ClusterSet`:**
  - **Funzionalità:** Restituisce la lunghezza dell'array `C` contenente i cluster
  - **Integrazione:** Utilizzato in `loadDendrogramFromFile()` per verificare la consistenza dimensionale tra dataset corrente e dendrogramma caricato
- **Metodo `getLevel0Length()` in `HierarchicalClusterMiner`:**
  - **Funzionalità:** Fornisce la cardinalità dei cluster al livello base del dendrogramma
  - **Scopo:** Garantire la coerenza tra struttura gerarchica e dati correnti
- **Metodo `getDepth()` in `HierarchicalClusterMiner`:**
  - **Funzionalità:** Determina la profondità effettiva del dendrogramma
  - **Utilizzo:** Previene caricamento di strutture incompatibili con il dataset selezionato

### 5.2 Evoluzione Architetture nella Versione Estesa

#### 5.2.1 Riprogettazione dello State Management

- **Decentralizzazione dello Stato:**
  - Rimozione della persistenza di `Data` lato server
  - Migrazione della gestione stato verso il client Telegram
  - Introduzione del pattern `StateContext` per isolare le sessioni utente
- **Implementazione del Contesto di Stato:**

- **StateHistory**: Stack per tracciare la cronologia degli stati
  - \* Supporto operazioni di rollback controllato
  - \* Meccanismo di transizione stati tramite **push/pop**
- **allowBack**: Flag per abilitare/disabilitare le transizioni inverse
- Separazione in fasi operative:
  - \* **preOperation**: Inizializzazione stato
  - \* **postOperation**: Gestione transizioni e validazione input

### 5.2.2 Modello di Gestione Errori Rafforzato

- **Gestione Centralizzata delle Eccezioni**:
  - **Pre-Operation**: Rollback automatico allo stato precedente
  - **Post-Operation**: Riterazione della fase corrente con notifica utente
- **Meccanismi di Validazione**:
  - Controllo consistenza nomi file
  - Verifica intervalli valori numerici
  - Convalida formale degli input testuali

### 5.2.3 Architettura Client-Server Ottimizzata

- **Comunicazione Stateless**:
  - Sessioni brevi per richiesta/risposta
  - Eliminazione lock su risorse condivise
- **Design Estensibile**:
  - Separazione chiara tra logica di business e gestione interfaccia
  - Astrazione dei canali di comunicazione (Socket/Telegram API)
- **Modello Sicurezza**:
  - Isolamento sessioni utente tramite ID univoci
  - Crittografia end-to-end per dati sensibili

Tabella 1: Confronto architetturale versione Base vs Estesa

Caratteristica	Base	Estesa
Gestione stato	Server-side	Client-side
Persistenza dati	Sessionale	Transazionale
Modello sicurezza	Base	Rafforzato
Concorrenza	Multi-thread	Event-driven

Questi adattamenti hanno permesso di:

- Supportare multipli client simultanei

- Garantire scalabilità orizzontale
- Migliorare l'efficienza nella gestione risorse
- Ottenere maggiore flessibilità architetturale

## 6 Eseguire A-CLus Esteso

Per avviare la versione estesa del progetto A-CLus, è necessario seguire i seguenti passaggi:

1. Accedere alla directory `A-CLUS_Esteso/Jar + Bat`
2. All'interno della suddetta cartella, eseguire il file `Start Server.bat` con doppio click, mantenendo attiva la finestra del prompt dei comandi
3. Procedere con l'avvio del file `Start Client.bat` tramite doppio click, il quale genererà un'ulteriore finestra di comando
4. Aprire Telegram e cercare "A-CLus" utilizzando il tag "`@A_Clus_bot`" oppure cliccando sul link diretto

### Avvertenza

È necessario mantenere attivi i terminali del server e del client per garantire la comunicazione tra tutte le componenti del sistema.

### 6.1 Avvio dall'ambiente di sviluppo

La procedura di avvio del progetto A-CLus può essere eseguita anche direttamente dall'ambiente di sviluppo, in maniera analoga alla versione base.

1. Aprire il progetto `A-CLus_Esteso`
2. Eseguire il file `MultiServer.java` presente nella cartella `server/src/server` per avviare il server
3. Eseguire il file `Main.java` presente nella cartella `client/src/client` per avviare il client
4. Aprire Telegram e cercare "A-CLus" utilizzando il tag "`@A_Clus_bot`" oppure cliccando sul link diretto per accedere al bot

### 6.2 Guida operativa al bot Telegram

Una volta assicurati che il server e il client siano in esecuzione, è possibile interagire con il bot Telegram.

La schermata iniziale del bot si presenta con un messaggio di benvenuto. È possibile sia scrivere `/inizia` che cliccare sul bottone per cominciare l'interazione con il bot.

Una volta iniziata l'interazione, verrà chiesto all'utente di selezionare il database da cui prendere i dati. In alternativa, si può schiacciare il bottone indietro per terminare l'interazione.



Figura 10: Interfaccia iniziale del bot Telegram



Figura 11: Interfaccia di scelta del database



Figura 12: Selezione dell'azione da eseguire

Una volta scelto il database, sarà possibile scegliere l'azione da eseguire. In questo caso, si può scegliere di eseguire un clustering nuovo, oppure di caricare un dendrogramma precedentemente salvato. In alternativa, si può schiacciare il bottone indietro per tornare alla schermata di selezione del database.

Nel caso si scelga di caricare un risurrisultato precedentemente calcolato, il bot chiederà di inserire il nome del file con cui è stato salvato, e restituirà il dendrogramma contenuto nel file.

Nel caso, invece, che si scelga di eseguire un nuovo clustering, il bot chiederà informazioni riguardo la profondità del dendrogramma da calcolare, e il metodo di calcolo delle distanze da utilizzare.



Figura 13: Informazioni per il calcolo del dendrogramma

Una volta inseriti i dati, il bot calcolerà il dendrogramma in base ad essi, e chiederà se si vuole salvare il dendrogramma in un file. Di seguito, chiederà di inserire il nome del file in cui salvare il dendrogramma.



Figura 14: Schermata finale

### 6.3 Test A-CLUS ESTESO

Durante la fase di testing della versione estesa, sono stati verificati i seguenti scenari:

- **Resilienza della connessione Telegram:** Il sistema gestisce correttamente disconnessioni temporanee e riprende l'elaborazione al ripristino della connettività.
- **Elaborazione concorrente di richieste:** La capacità di processare simultaneamente richieste da diversi utenti Telegram è stata validata anche in condizioni di carico elevato.
- **Consistenza tra piattaforme:** I risultati ottenuti tramite interfaccia Telegram sono identici a quelli generati dall'applicazione desktop.
- **Persistenza dei dati:** I dendrogrammi salvati attraverso l'interfaccia Telegram possono essere recuperati sia dalla versione estesa che da quella base.
- **Gestione di input non validi:** Il bot implementa controlli di validazione che prevengono errori di sistema in caso di input malformati.

## 7 Cambiamenti

### 7.1 Cambiamenti Motivati nella Versione Base Rispetto i Requisiti del Progetto

1. Metodo `getLength()` in `ClusterSet` / Metodo `getLevel0Length()` in `HierarchicalClusterMiner`:
  - **Descrizione `getLength()`:** Questo metodo restituisce la lunghezza dell'array `C` in `ClusterSet`.
  - **Descrizione `getLevel0Length()`:** Questo metodo restituisce la lunghezza del set di cluster al livello 0 del dendrogramma.
  - **Motivazione:** Abbiamo aggiunto questi metodi per ottenere il numero di elementi presenti nel `ClusterSet` al livello 0 del dendrogramma e per passare tale informazione al metodo `loadDendrogramFromFile()` in `ServerOneClient`. Questo permette di verificare qui che il numero di esempi nella tabella scelta non sia minore del numero di esempi con cui è stato salvato il dendrogramma precedentemente.
2. Metodo `getDepth()` in `HierarchicalClusterMiner`:
  - **Descrizione `getDepth()`:** Questo metodo restituisce la profondità del dendrogramma in `HierarchicalClusterMiner`.

- **Motivazione:** Abbiamo aggiunto questo metodo per verificare nel metodo `loadDendrogramFromFile()` in `ServerOneClient` che la profondità del dendrogramma caricato da file non sia maggiore del numero di esempi nella tabella scelta dall'utente.

## 7.2 Cambiamenti Motivati nella Versione Estesa Rispetto alla Base

### 1. Rimossa responsabilità gestione macchina a stati nel server.

Nella versione base del programma, il metodo `run` di `ServerOneClient` memorizzava l'oggetto `Data`, che rappresentava la tabella contenente il dataset scelto dall'utente. Questo oggetto era creato alla prima richiesta dal client, nel momento in cui l'utente specificava quale tabella del database utilizzare. I dati erano poi conservati all'interno del thread avviato per ciascuna connessione tra client e server, e mantenuti attivi grazie a un ciclo `while true`, che proseguiva fino alla disconnessione del client.

Nella versione estesa, invece, il client gestisce gli aggiornamenti (`Updates`) inviati dall'API di Telegram, che contengono i messaggi degli utenti diretti al bot. Teoricamente, il server della versione base potrebbe essere compatibile con il client della versione estesa, poiché è progettato per gestire connessioni multiple tramite `multithreading`: ogni aggiornamento del bot potrebbe quindi attivare un thread separato del server, avviando una nuova connessione `Socket`.

Tuttavia, questa soluzione è adatta solo se la connessione `socket` può rimanere aperta a lungo. Se, per esempio, un utente inviasse un messaggio ora e il successivo solo una settimana dopo, mantenere la connessione aperta risulterebbe inefficiente e costoso, poiché l'unico scopo sarebbe mantenere vivo l'oggetto `Data` nel thread.

Per questa ragione e per garantire al server una struttura generica, capace di interfacciarsi con client di natura diversa, abbiamo deciso di spostare l'oggetto `Data` nel client esteso, insieme ad altre informazioni necessarie per la gestione degli stati. Ad esempio, in futuro potrebbero esistere sia un client web sia un client bot Telegram. Supponiamo che un utente esegua due clustering uno sulla versione web e una sulla versione bot Telegram; mantenere tutti i dati di stato sul server limiterebbe la possibilità di separare l'interazione su dispositivi diversi, oltre a imporre uno schema fisso basato sulla macchina a stati anche per i client che non lo necessitano. Un client senza macchina a stati potrebbe, ad esempio, essere una versione web in cui le operazioni "Carica tabella", "Profondità", "Distanza" e "Clustering" sono eseguite in un'unica richiesta, senza passare attraverso stati successivi.

Nel nuovo client quando riceve un aggiornamento da parte di un nuovo utente, crea un record nel dizionario memorizzando come chiave l'ID Telegram dell'utente e come valore un'istanza della classe `StateContext`, che gestisce la macchina a stati. Quando un utente invia un messaggio e la sua chiave è già presente nel dizionario, il sistema esegue le operazioni sul `StateContext` già esistente per quell'utente. `StateContext` è responsabile della gestione della macchina a stati e contiene uno `Stack` di stati, che

memorizza la cronologia degli stati attraversati dall'utente, più l'ultimo stato attivo in cui applicare gli aggiornamenti. Inoltre, `StateContext` include una classe `Dati` che funge da aggregatore di vari dati condivisi tra gli stati e include anche l'oggetto `Data` (dataset) originale della versione base del server. In questo modo, il sistema può gestire la persistenza e la transizione degli stati per ciascun utente, senza dover mantenere una connessione server-client attiva per tempi prolungati.

La classe `StateContext` contiene i seguenti attributi di istanza:

- **Stack <State> StateHistory:** rappresenta la cronologia degli stati attraversati dall'utente. Di default, lo stack contiene lo stato iniziale passato nel costruttore e viene poi popolato aggiungendo ogni nuovo stato tramite operazioni di `push`. Questo approccio facilita il `rollback` degli stati. Ad esempio, se l'utente è nello stato 3 e lo stack contiene gli stati `[3, 2, 1]`, un `rollback` eseguirà un `pop` dello stato 3, riportando automaticamente l'utente allo stato 2, senza la necessità di gestire manualmente ogni singolo caso di retrocessione.

Ogni stato ha la proprietà `allowBack`. Se impostata su `false` (valore di default), consente il `rollback` dallo stato corrente. Se invece è impostata su `true`, non è possibile tornare indietro da quello stato. Questo comportamento è stato applicato allo stato iniziale, poiché non è consentito fare `rollback` una volta entrati nello stato di avvio. Ad esempio, gli stati `Clustering` e `ShowSavedClustering` offrono opzioni per tornare allo stato principale (ossia `Start`), ma una volta raggiunto lo stato `Start`, il `rollback` non è consentito poiché segna l'inizio di una nuova iterazione.

Ogni stato è suddiviso in due fasi di operazione:

- **Pre-Operation:** viene eseguita una sola volta quando lo stato viene creato o resettato. Nel flusso di lavoro attuale, ogni stato fornisce all'utente una serie di operazioni da eseguire, attende la risposta dell'utente e la valida. La fase di `preOperation` corrisponde all'operazione di chiedere all'utente cosa desidera fare.
- **Post-Operation:** valida la risposta dell'utente e gestisce le transizioni verso altri stati. Passando `StateContext` come parametro al metodo `postOperation`, è possibile richiamare il metodo `changeState` per indicare il nuovo stato da eseguire. Una volta cambiato lo stato, viene eseguita automaticamente la `preOperation` del nuovo stato, permettendo al sistema di rispondere immediatamente all'utente con il nuovo set di operazioni disponibili per quello stato.

Durante l'esecuzione di una delle due operazioni possono verificarsi errori dovuti a input non validi, dati incorretti o problemi nelle risposte socket. Il comportamento in caso di errore varia in base alla fase dell'operazione:

- **Pre-Operation:** un errore in questa fase causa un `rollback` automatico allo stato precedente. Ad esempio, nella fase di pre-operazione del `Clustering`, può verificarsi un errore se l'utente sceglie un nome per il file di salvataggio del clustering che esiste già dallo stato precedente. In tal caso, il blocco `catch` esegue un `rollback` allo stato precedente (con apposito log d'errore), ovvero lo stato `ChooseName`

- **Post-Operation:** in questa fase, un eventuale errore non altera il flusso del programma, ma comporta la ripetizione della post-operazione. Ad esempio, nel metodo ChooseDepth, la post-operazione convalida la profondità scelta dall'utente; se questa profondità non è valida o fuori range, viene generata un'eccezione che porta al riavvio della post-operazione (con apposito log d'errore), consentendo all'utente di inserire una nuova profondità valida.

[illegible]

Pagina 24 di 24