

FUZZY RECOMMENDER SYSTEM

**Fuzzy Clustering per il Supporto alle Decisioni in
Sistemi di Raccomandazione**

PANORAMICA

01

Contesto e
Obiettivo del
Progetto

02

Fondamenti
Teorici

03

Analisi
Esplorativa
Dataset

04

Implementazione
del Sistema

05

Risultati
Sperimentali

06

Conclusioni e
Sviluppi Futuri

Contesto e Obiettivo del Progetto

01

Insiemi Fuzzy e Sistemi di Raccomandazione

I sistemi di raccomandazione sono strumenti fondamentali per filtrare informazioni e suggerire contenuti rilevanti agli utenti.

Gli insiemi fuzzy permettono di modellare l'incertezza e la gradualità tipiche delle preferenze umane.

L'integrazione tra insiemi fuzzy e sistemi di raccomandazione consente una rappresentazione più realistica delle preferenze degli utenti.

01

Obiettivo del Progetto

Esplorare l'utilizzo del fuzzy clustering nei sistemi di raccomandazione per contenuti audiovisivi.

Estendere metodologicamente l'approccio in "User based Collaborative Filtering using fuzzy C-means" di Koohi e Kiani. (2016).

Fornire un framework riproducibile per valutare il clustering fuzzy, tramite metriche quantitative, qualitative e analisi visive.

01

Fondamenti Teorici

02

Logica Fuzzy e Clustering

Gli insiemi fuzzy generalizzano gli insiemi classici: l'appartenenza non è più "tutto o niente", ma graduata tra 0 e 1.

La funzione di appartenenza esprime similarità, incertezza o preferenza in modo flessibile.

Il clustering fuzzy (es. Fuzzy C-Means) permette a ciascun elemento di appartenere a più cluster con diversi gradi di appartenenza.

Il parametro di fuzzificazione controlla la "morbidezza" della partizione: più alto è, più le appartenenze sono distribuite.

02

Sistemi di Raccomandazione

I sistemi di raccomandazione guidano l'utente verso contenuti rilevanti in grandi spazi di opzioni.

Principali approcci:

- Filtraggio demografico: usa attributi dell'utente.
- Filtraggio collaborativo: si basa sulle valutazioni degli utenti.
- Raccomandazione basata sul contenuto: sfrutta le caratteristiche degli oggetti.
- Approcci ibridi: combinano più strategie.

Il filtraggio collaborativo è tra i più diffusi: utenti simili ricevono raccomandazioni simili.

02

Clustering nei sistemi di raccomandazione

Le misure di similarità tradizionali possono essere inefficaci su grandi dataset o in presenza di dati sparsi. Il clustering organizza gli utenti in gruppi omogenei, facilitando la raccomandazione.

Tecniche comuni:

- K-Means: partizioni nette, semplice ma poco flessibile.
- SOM: reti neurali che preservano la topologia dei dati.
- Fuzzy Clustering: consente appartenenze multiple, utile per utenti con gusti complessi.

Il clustering migliora la scalabilità, riduce la sparsità e rafforza la coerenza delle raccomandazioni.

02

Analisi Esplorativa Dataset

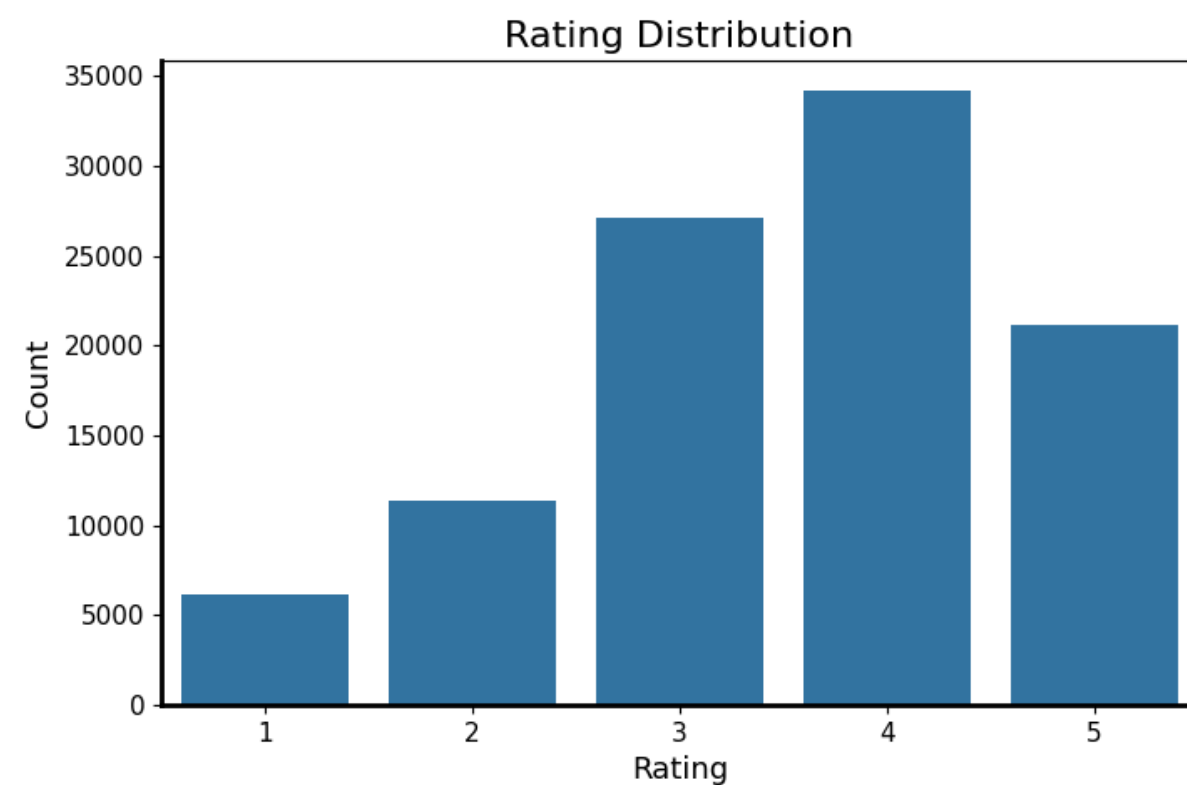
03

Descrizione Dataset

Utilizzato il dataset MovieLens 100k: 100.000 valutazioni, 943 utenti, 1682 film, 6% densità.

Ogni valutazione è un intero da 1 a 5; disponibili anche informazioni su utenti e film.

Le valutazioni sono fortemente sbilanciate verso i valori alti (bias positivo).



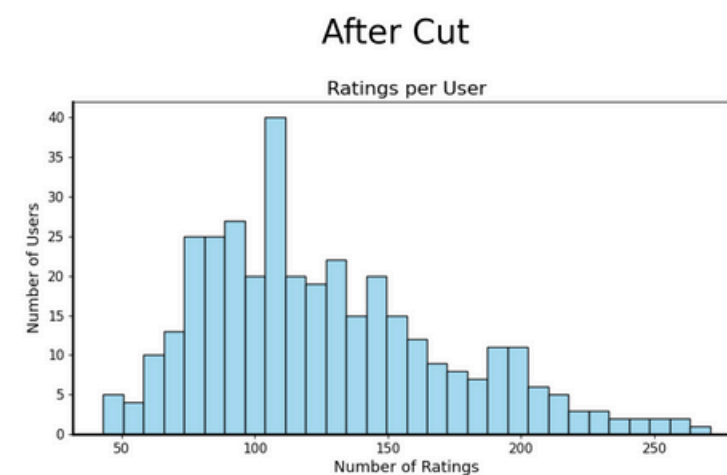
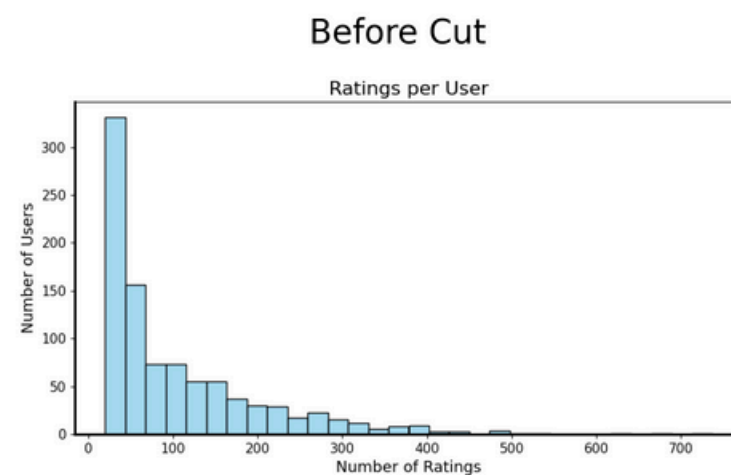
03

Preprocessamento del Dataset

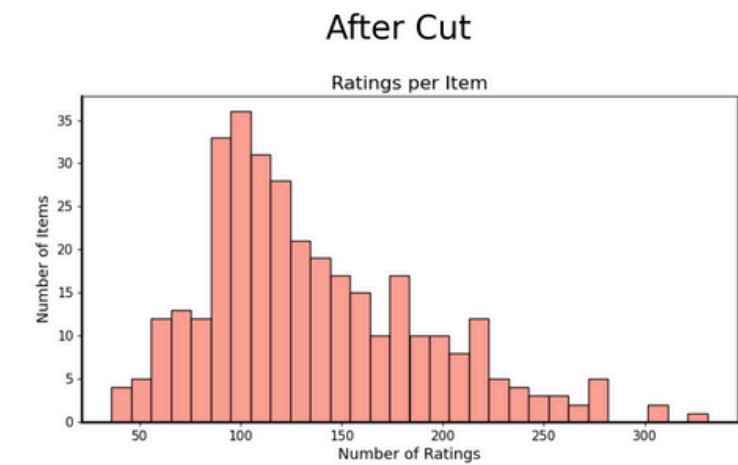
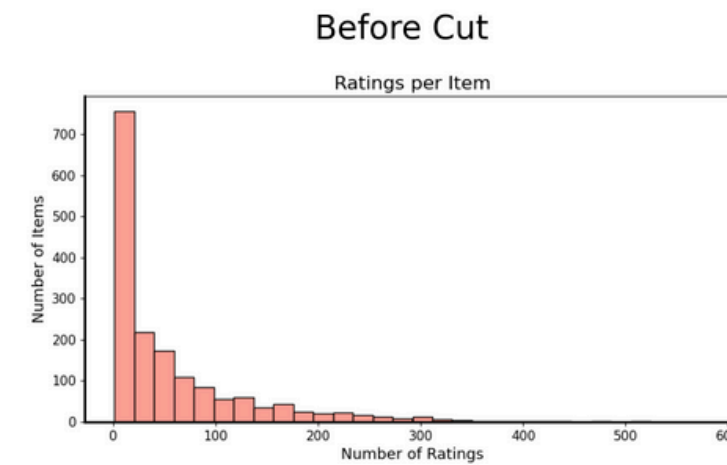
Mantenuti solo utenti e film con almeno 100 valutazioni.

Il filtraggio riduce il dataset a 45.926 valutazioni, 364 utenti e 338 film, 37% densità.

Ratings Per User Hist

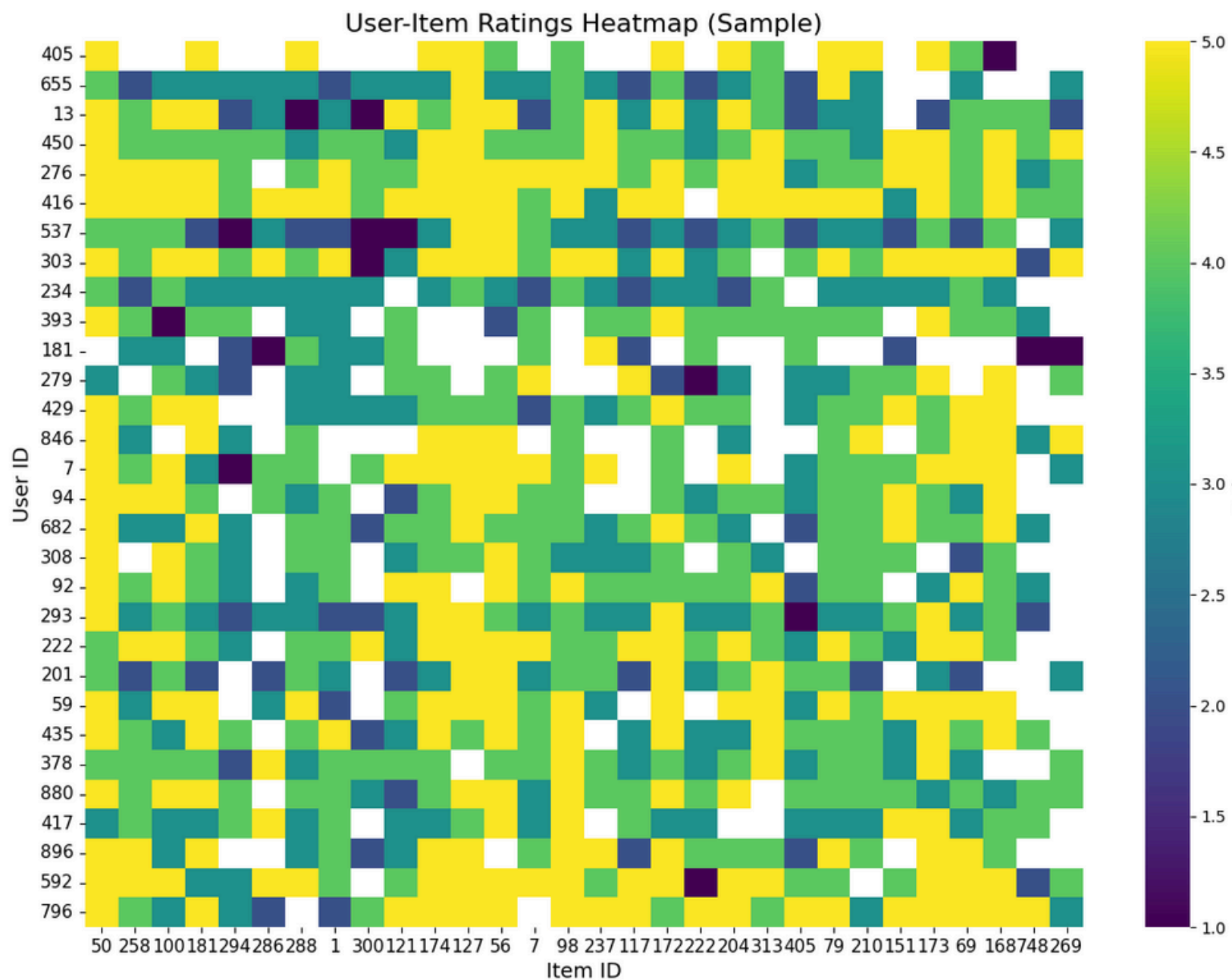


Ratings Per Item Hist

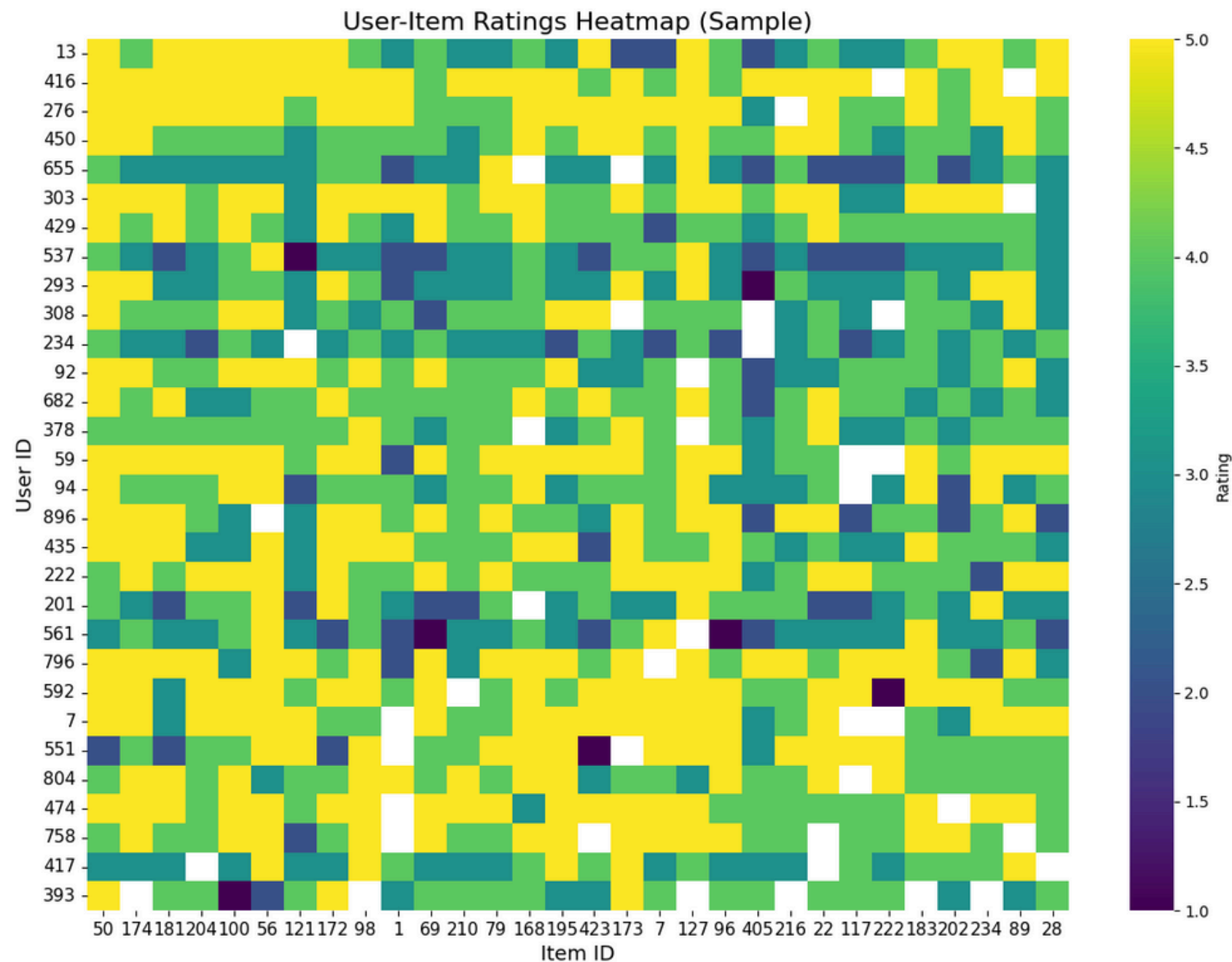


Sparsità Matrice

prima

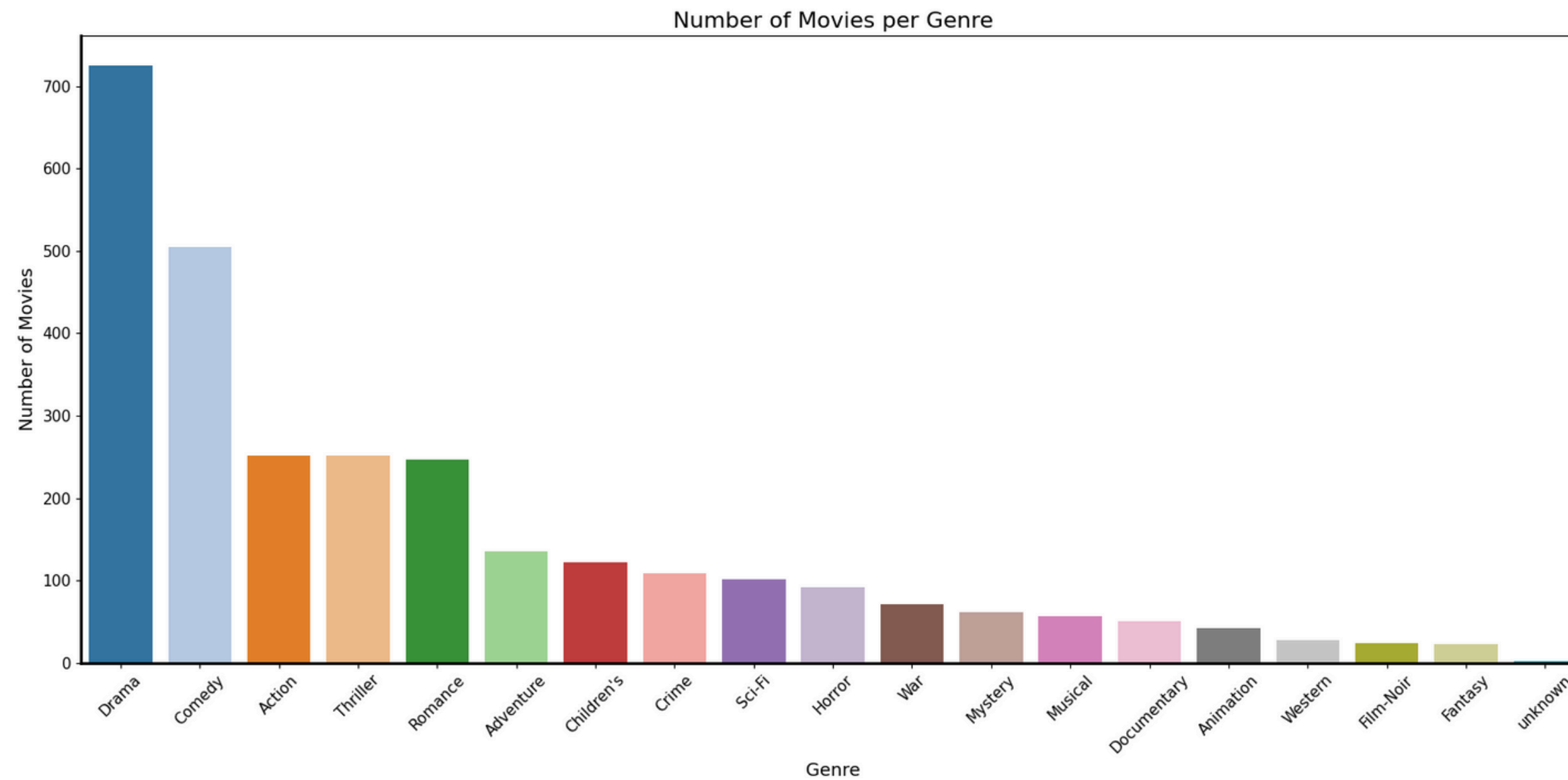


dopo



Distribuzioni

L'analisi dei generi rivela una forte predominanza di alcuni generi.



03

Implementazione del Sistema

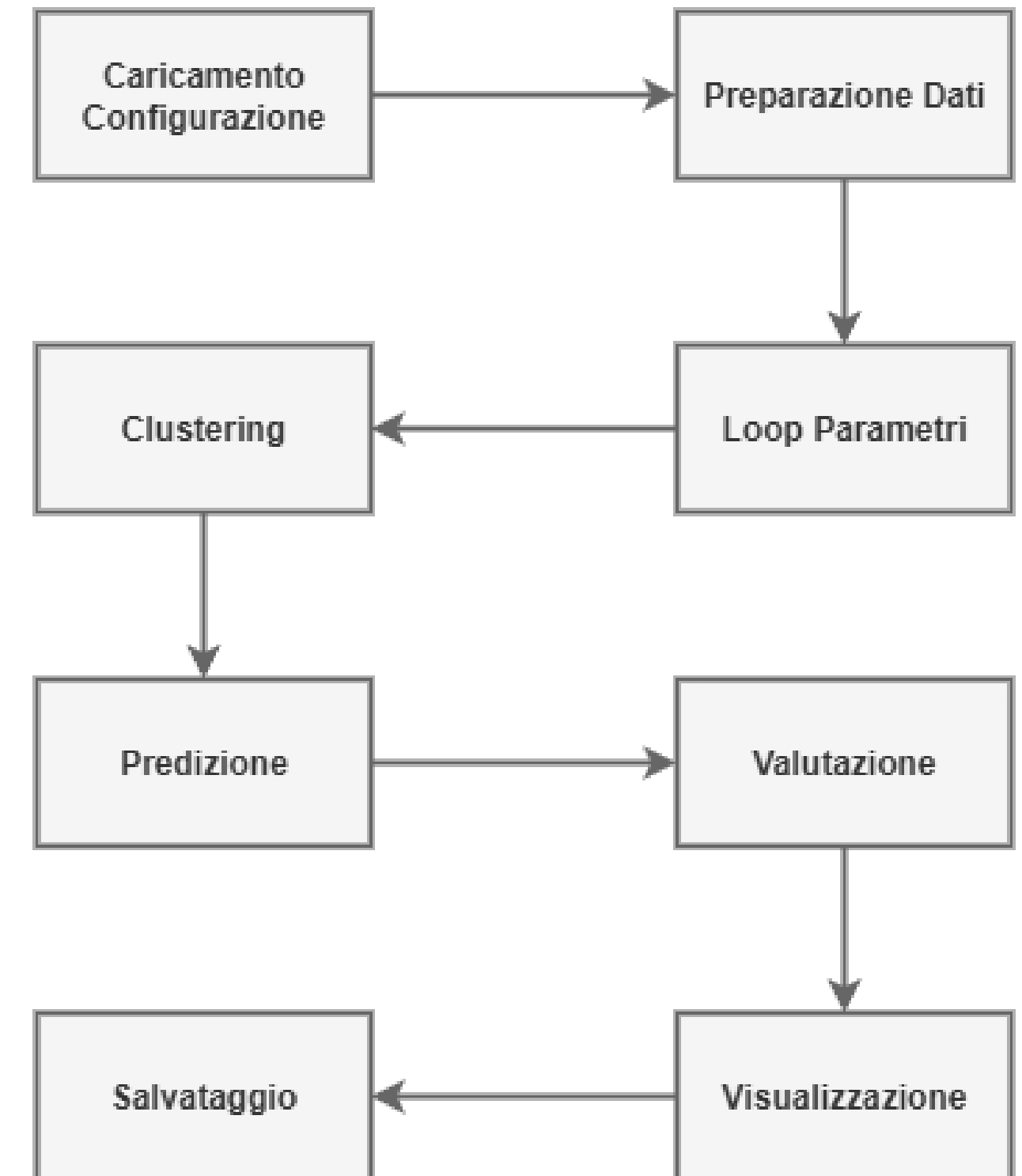
04

Architettura e pipeline sperimentale

Architettura modulare e scalabile organizzata in quattro livelli:

- Livello di Configurazione: gestione centralizzata dei parametri
- Livello di Orchestrazione: coordinamento del flusso di esecuzione
- Livello di Elaborazione: clustering, valutazione e visualizzazione
- Livello di Utilità: funzioni di supporto

Configurazione centralizzata tramite file JSON per facilitare la sperimentazione



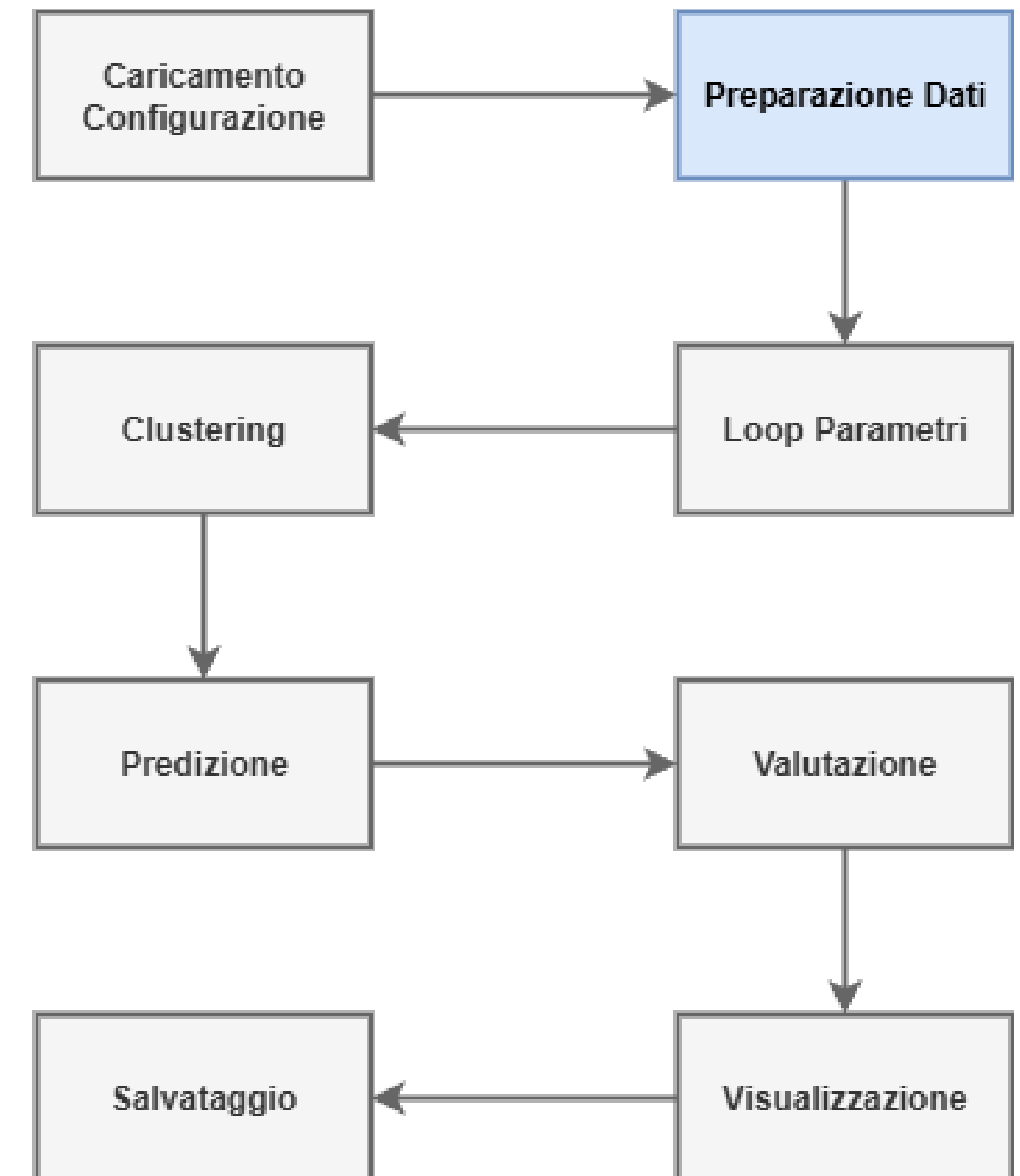
Gestione dati e normalizzazione

DataManager responsabile di caricamento, preprocessamento e normalizzazione

Filtraggio automatico: utenti e item con almeno 100 valutazioni

Split train/test configurabile con allineamento delle matrici

Quattro strategie di normalizzazione implementate.



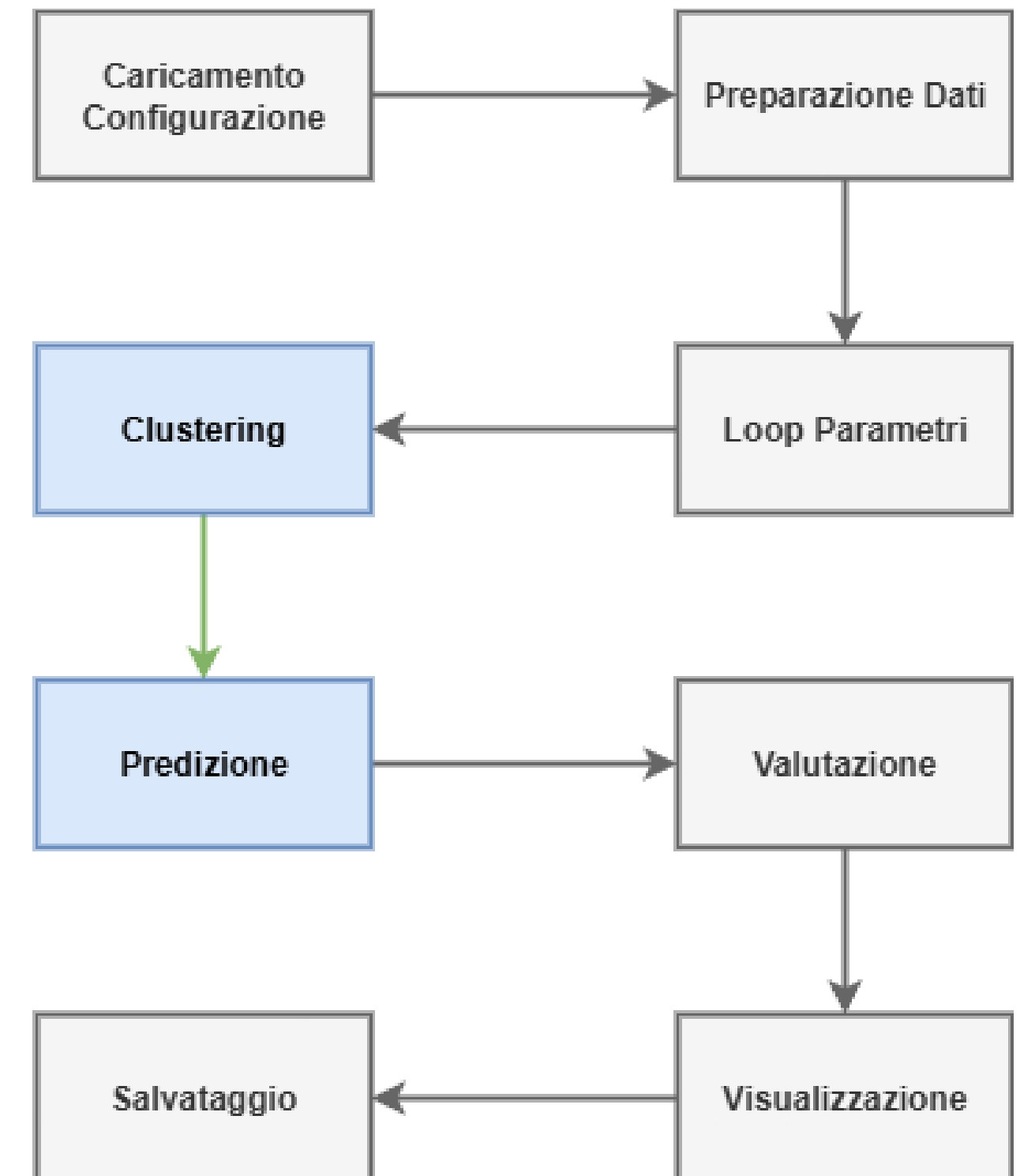
Algoritmi di clustering e predizione

Fuzzy C-Means (FCM): algoritmo principale con membership graduali

K-Means: baseline per confronto con clustering hard

Funzione obiettivo FCM: minimizzazione della somma pesata delle distanze

Predizione rating tramite media pesata dei centroidi



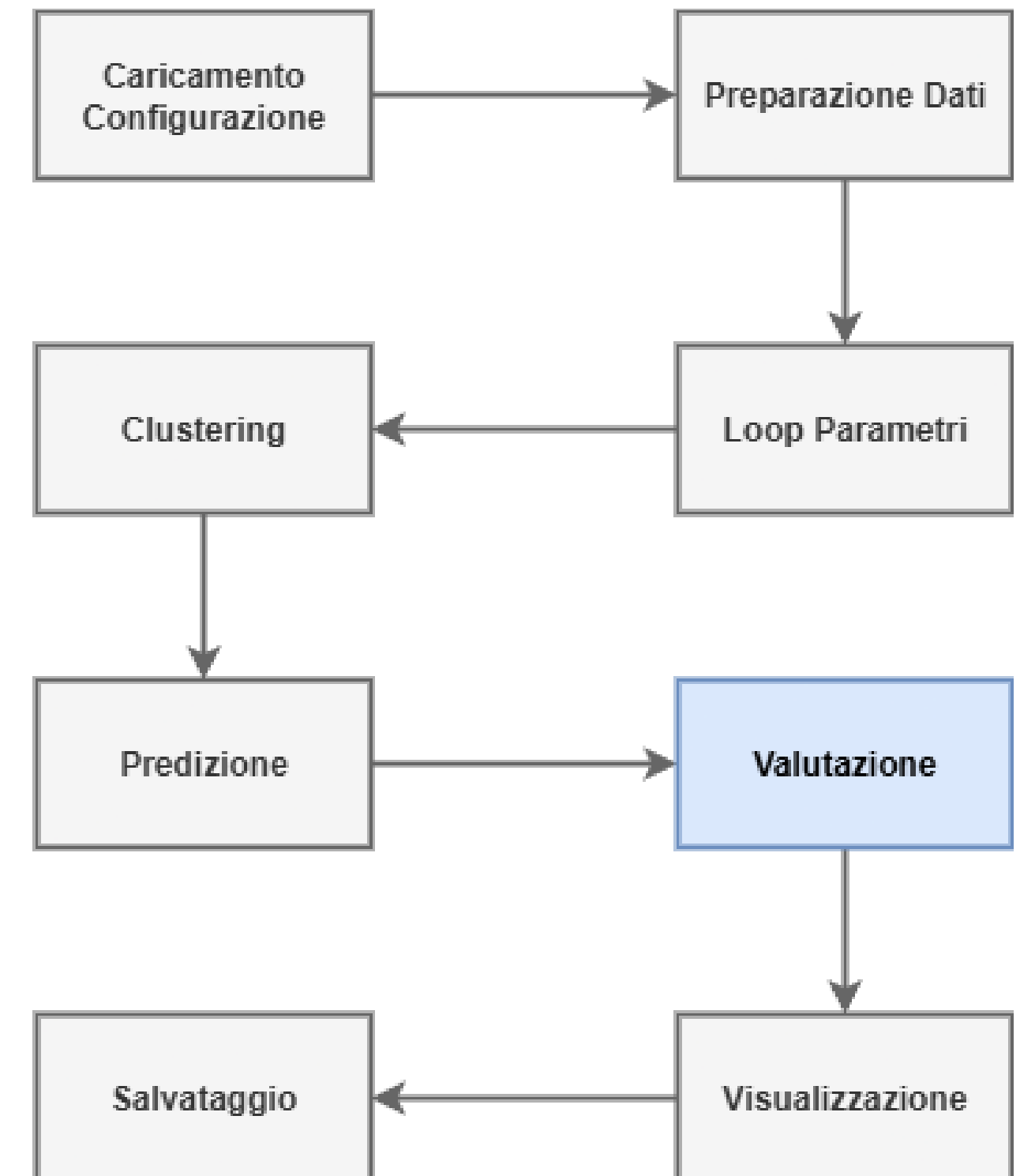
Defuzzificazione e selezione vicini

Due strategie di defuzzificazione:

- Metodo del Massimo: assegna utente al cluster con membership massima
- Center of Gravity (COG): calcola indice continuo come media pesata

Selezione vicini opzionale:

- Nessuna selezione: considera tutti gli utenti del cluster
- Correlazione di Pearson: filtra basandosi su similarità (threshold 0.5)

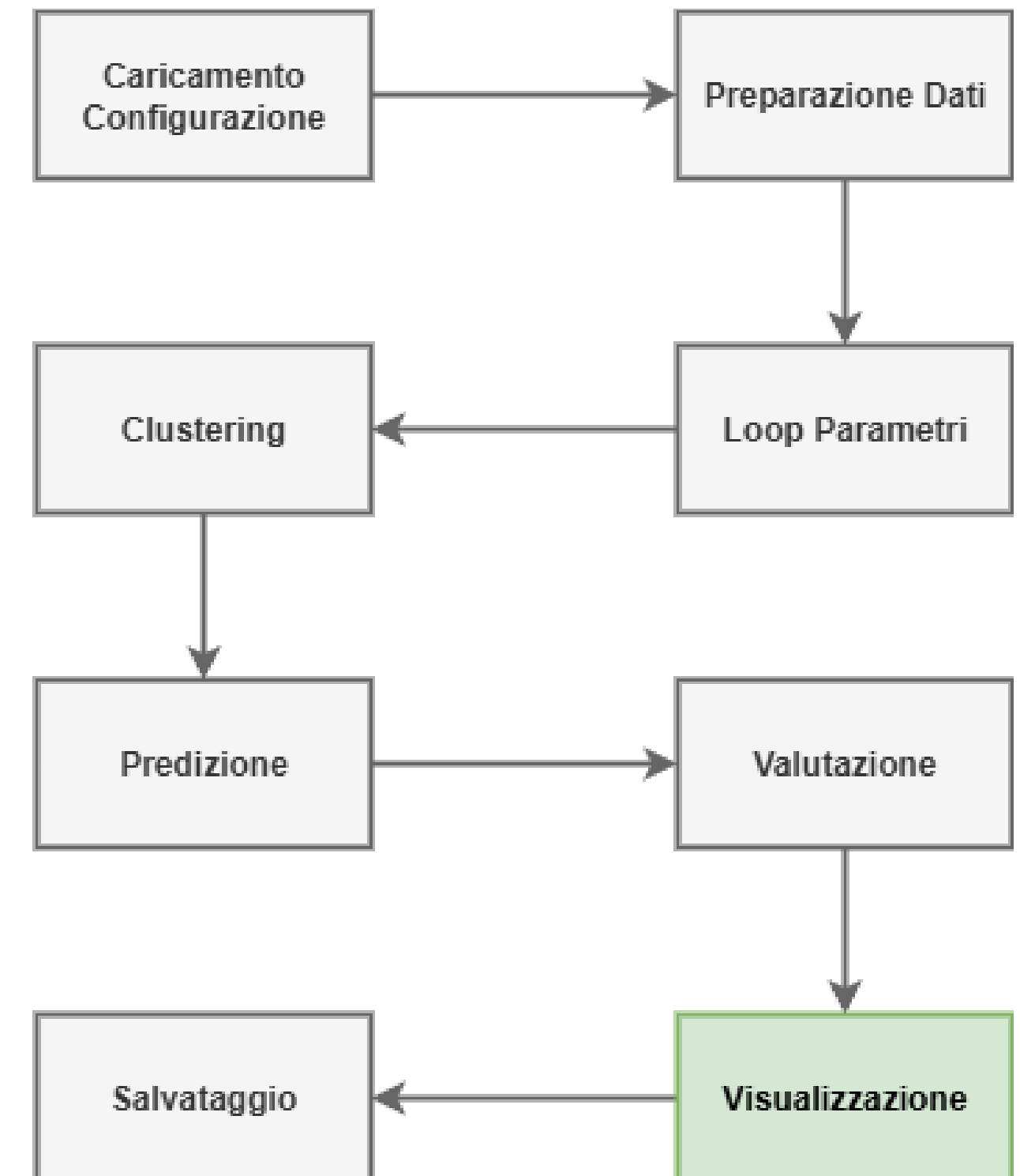


Metriche e Visualizzazione

Metriche

- Predittive: RMSE, MAE
- Qualitative: Precision, Recall, Accuracy, F1-score

Moduli di visualizzazione: heatmap, boxplot, barplot, lineplot, violinplot, histogram ecc...



Risultati Sperimentali

05

Run KMeans – Train

Simple Centering risulta la normalizzazione migliore per RMSE e MAE in training.

I valori di errore (RMSE ~ 0.69 , MAE ~ 0.51) sono costanti al variare del numero di cluster (2-5).

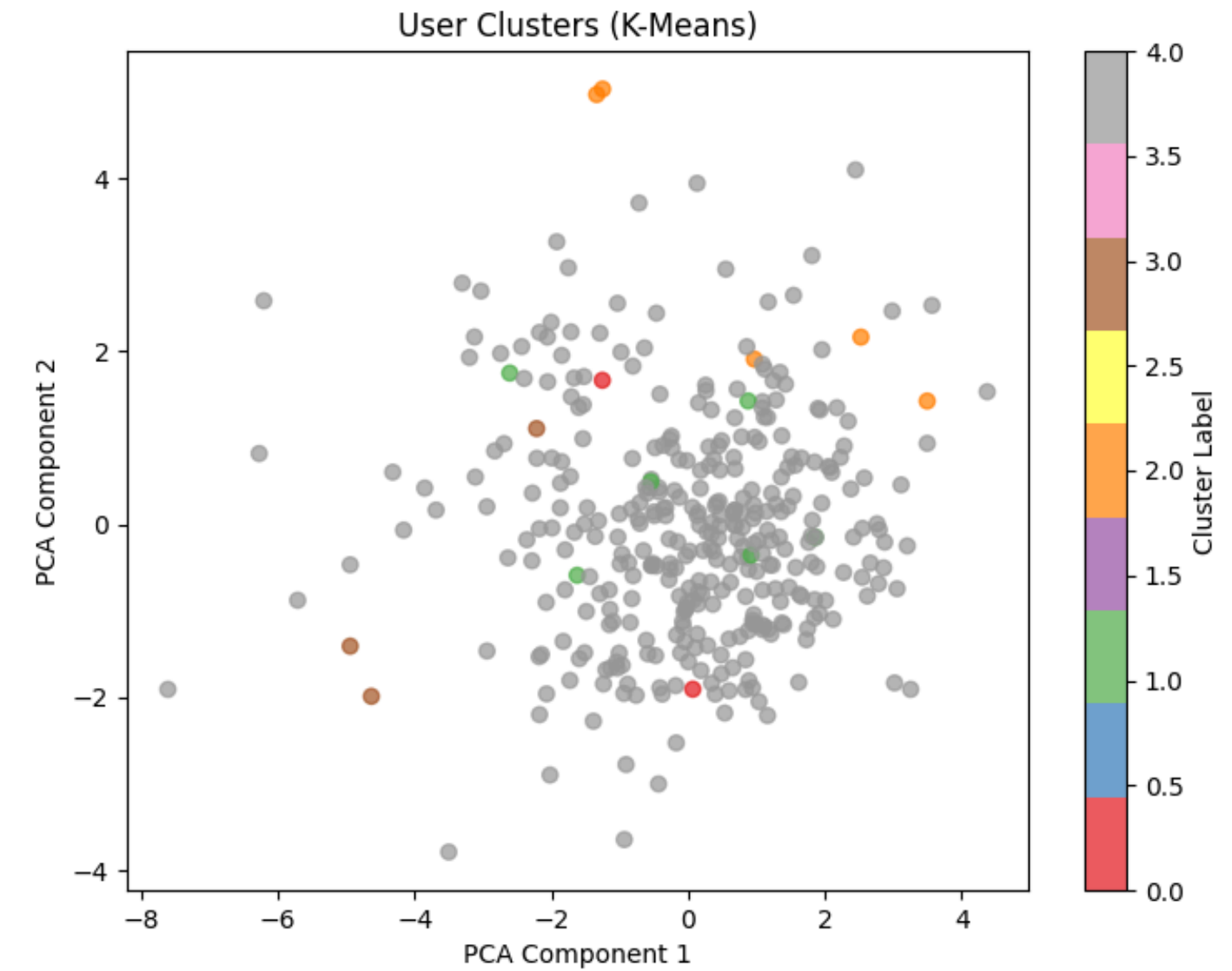
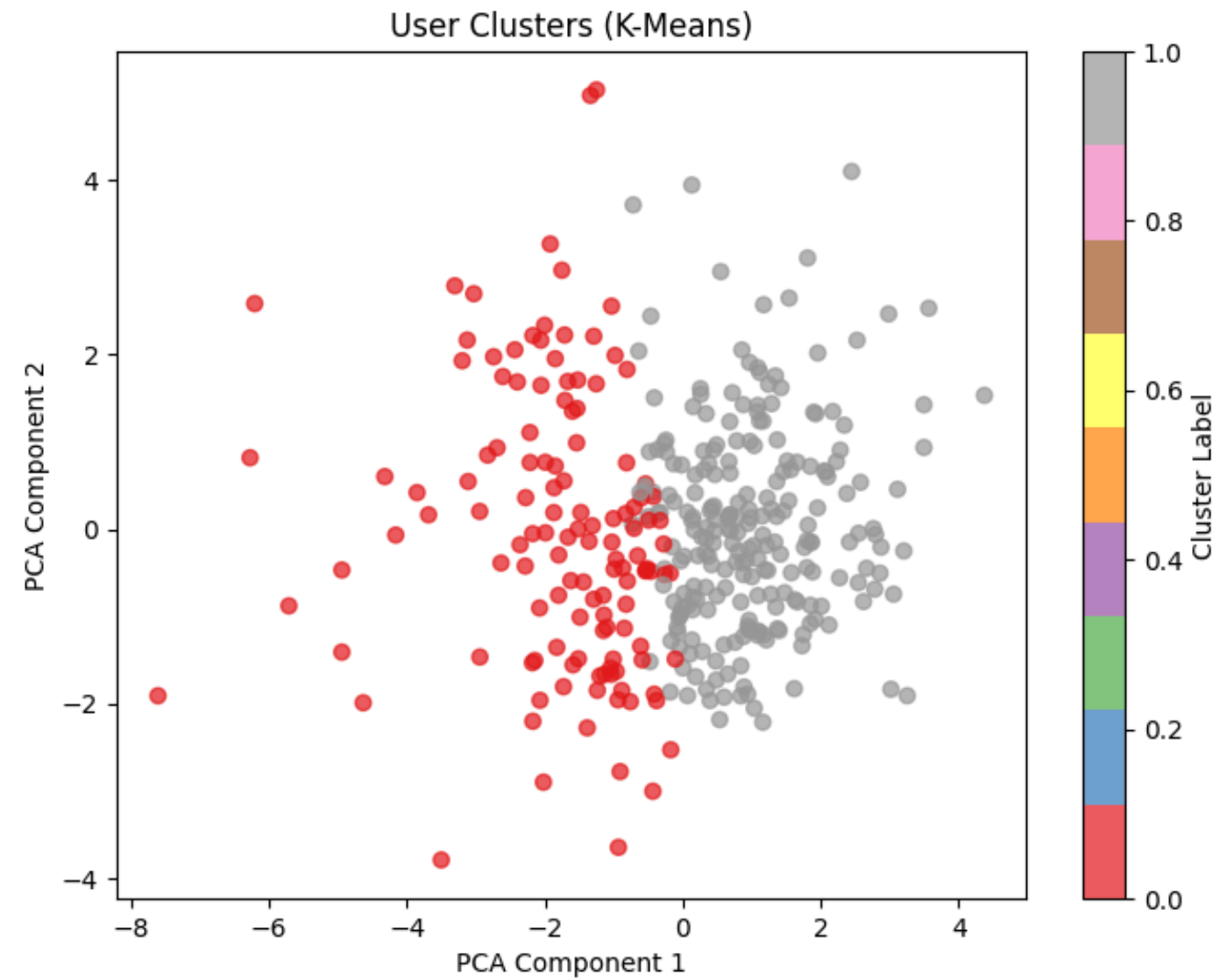
Il clustering non produce separazioni significative tra i profili utente: utenti attivi hanno gusti simili.

Precision e accuracy in training intorno al 59%, recall molto basso ($\sim 14\%$).

F1-score basso (~ 0.21) a causa del recall limitato

05

Run KMeans – Train



Run KMeans – Test

Simple Centering domina anche in test per RMSE (0.68) e precision/accuracy (60.5%).

Z-score per utente ottiene il miglior MAE (0.54).

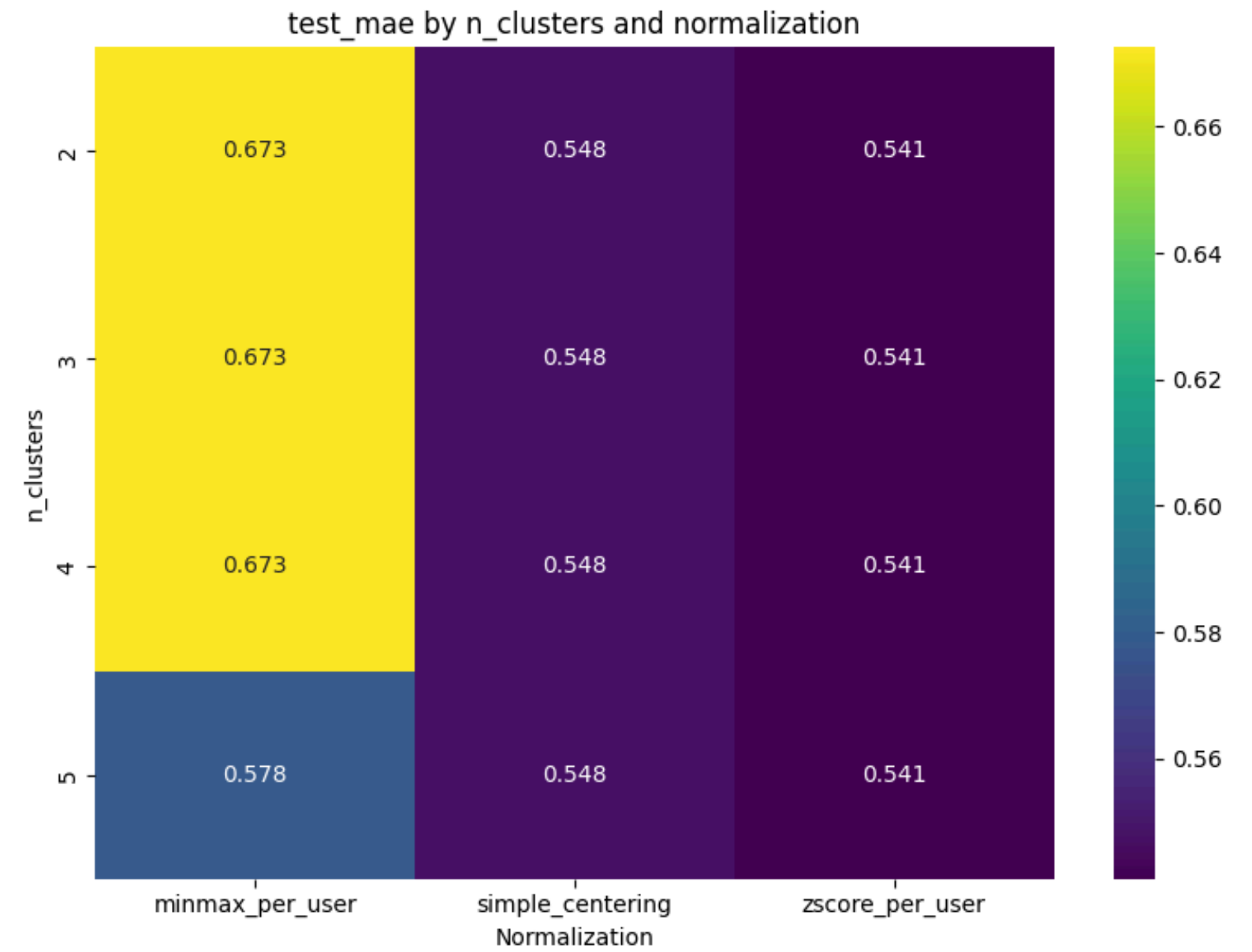
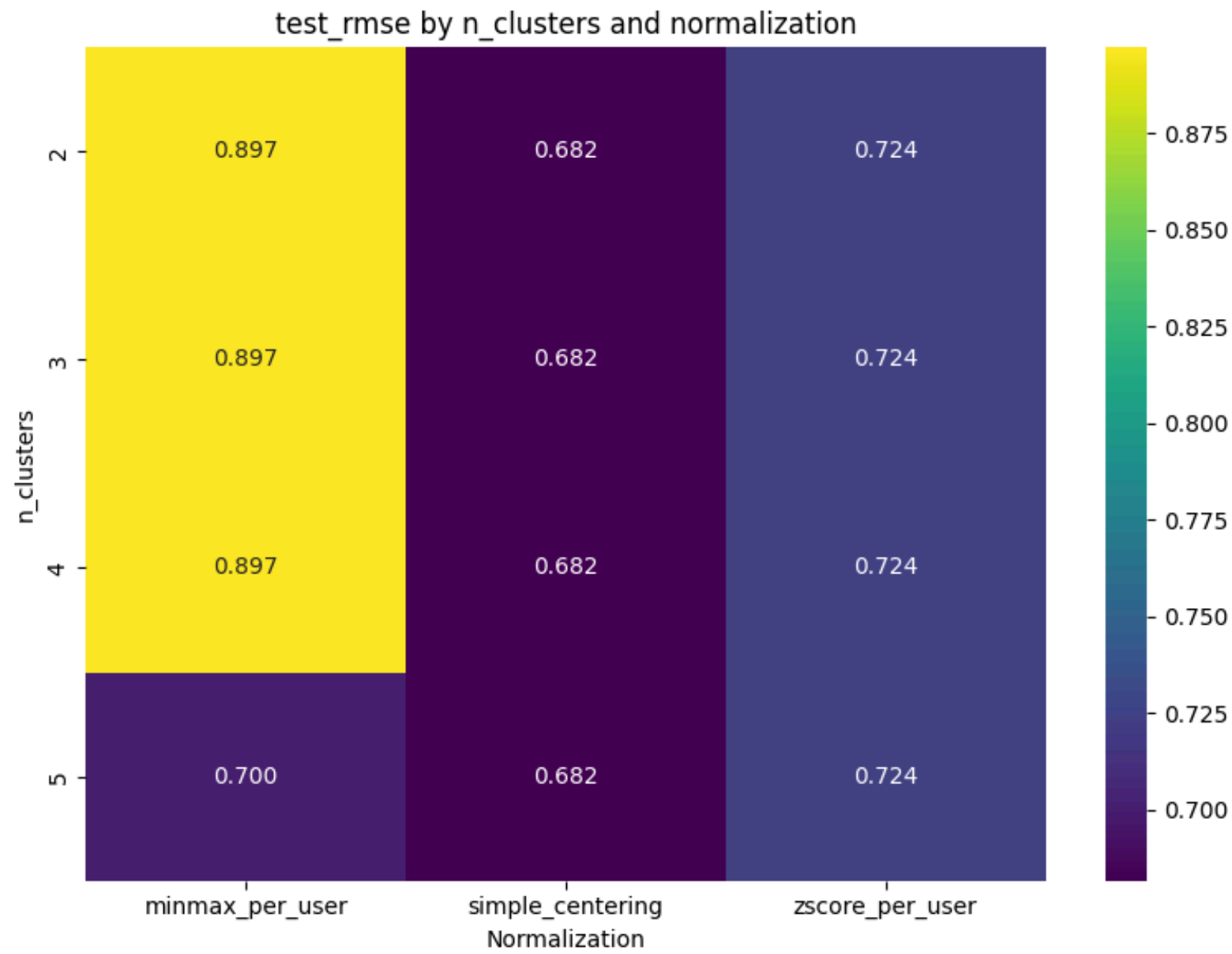
I valori di precision, recall e F1-score migliorano sensibilmente rispetto al training (recall ~51%, F1 ~0.52).

Le performance sono stabili al variare del numero di cluster.

Il sistema generalizza bene su dati non visti.

05

Run KMeans – Test



Run FCM – Train

Simple Centering con 2 cluster domina per RMSE (0.685) e MAE (0.511).

I valori di Avg Max Membership e Avg Entropy sono costanti, indicando una distribuzione uniforme delle membership.

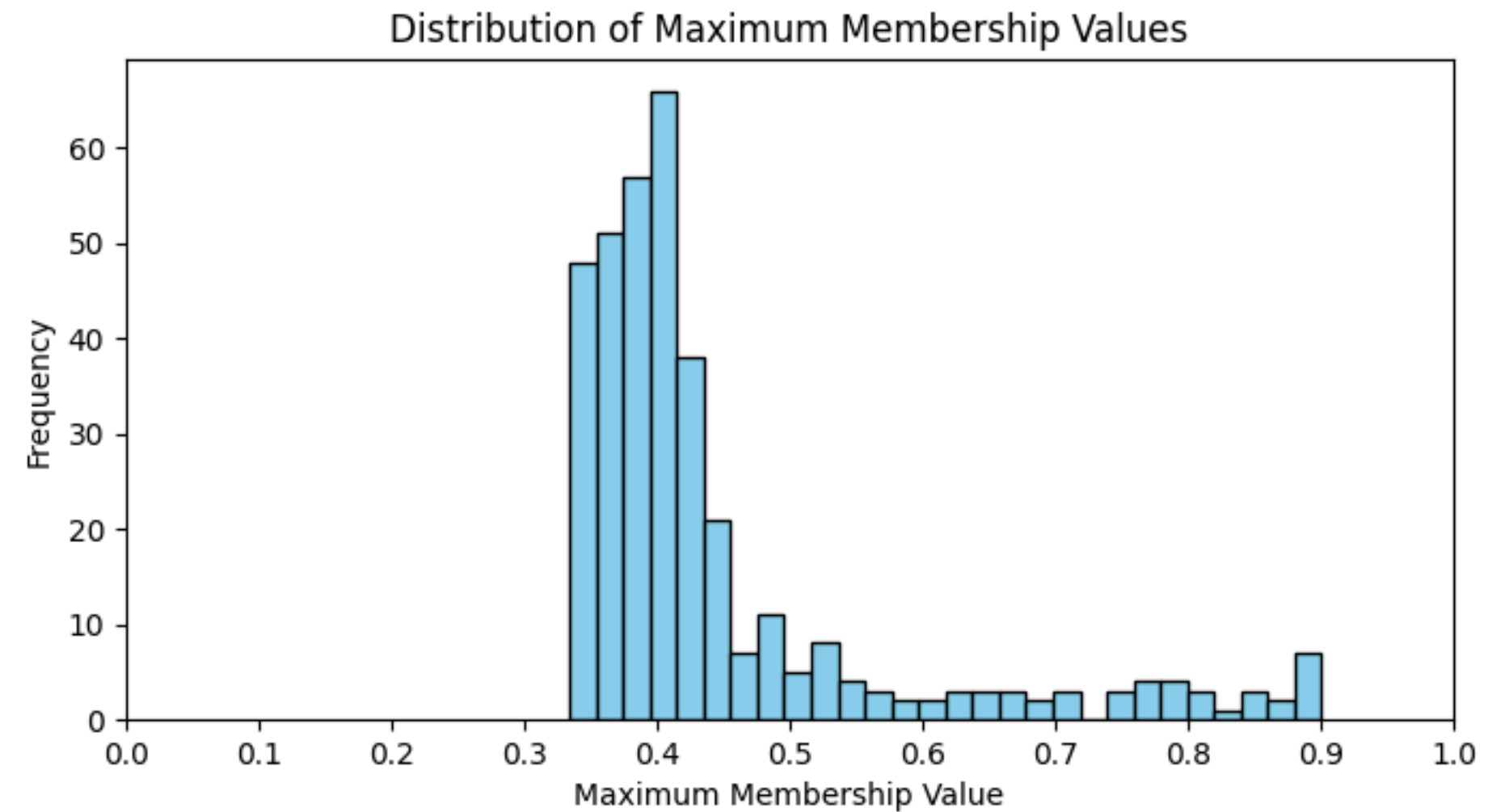
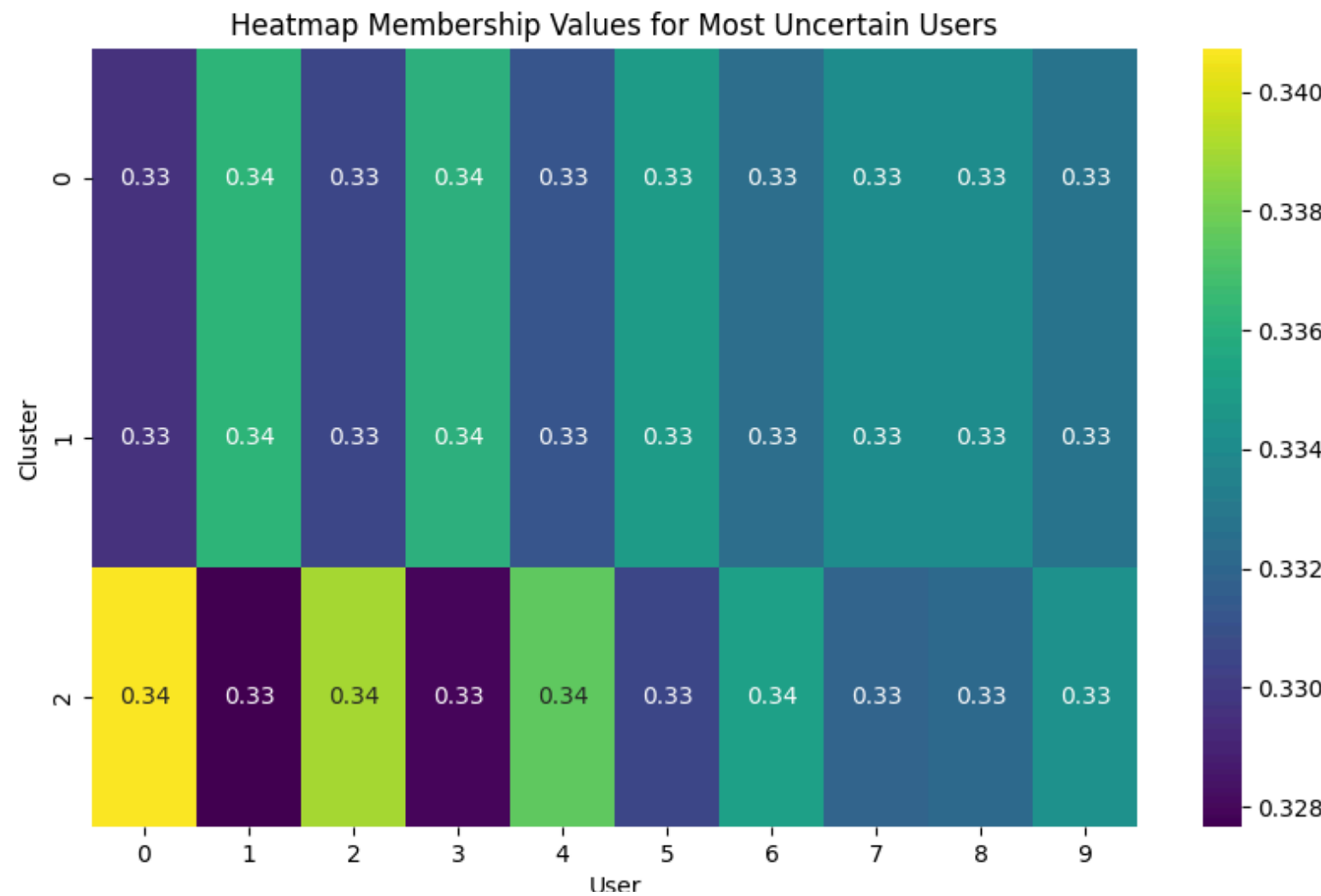
Le metriche di qualità (precision, accuracy) sono simili a K-Means (~59%), recall basso (~14%), F1-score basso (~0.21).

FCM introduce una maggiore flessibilità nella rappresentazione delle preferenze, ma non migliora le metriche rispetto a K-Means.

05

Run FCM – Train

3 Cluster, 1.2 m, Maximum, Pearson



Run FCM- Test

Le performance di FCM in test sono quasi identiche a quelle di K-Means. Il sistema non riesce ad identificare pattern evidenti all'interno dei dati, nè sfumature di preferenze.

Metrica	FCM Train	K-Means Train	FCM Test	K-Means Test
RMSE	0.6851	0.6864	0.6789	0.6817
MAE	0.5107	0.5092	0.5448	0.5409
Precision	58.9%	58.8%	60.5%	60.5%
Recall	13.8%	13.8%	51.2%	51.2%
F1-Score	21.1%	21.1%	52.4%	52.4%

Conclusioni e Sviluppi Futuri

06

Conclusioni

Confrontato Fuzzy C-Means (FCM) e K-Means su MovieLens 100k tramite un framework sperimentale flessibile.

Le prestazioni dei due algoritmi sono risultate molto simili sia per metriche di errore che di qualità delle raccomandazioni.

FCM fatica a individuare cluster realmente significativi: le membership risultano distribuite in modo uniforme e l'entropia è elevata.

L'omogeneità del dataset limita la possibilità di apprendere strutture fuzzy informative.

Il framework sviluppato si è dimostrato efficace e facilmente estendibile ad altri contesti.

06

Sviluppi Futuri

Mantenere la sparsità originale del dataset per valutare l'impatto sulle performance e sulle strutture individuate.

Applicare il framework a dataset di dominio differente (e-commerce, musica, news) per testare la formazione di cluster fuzzy più significativi.

Estendere l'analisi a dataset di dimensioni maggiori (es. MovieLens 1M) per aumentare l'eterogeneità.

Variare sistematicamente altri parametri sperimentali (rumore, test set, min rating) per studiarne l'impatto.

Integrare e confrontare altre tecniche di clustering fuzzy o approcci ibridi per superare i limiti osservati con FCM.

06

Domande?

Grazie dell'attenzione!

LilQuacky/**fuzzy-recommender-system**

Python-based framework for fuzzy clustering-based recommender systems

1

Contributor

0


Issues

0

Stars


0

Forks



LilQuacky/fuzzy-recommender-system: Python-based framework for fuzzy clustering-based recommender...

Python-based framework for fuzzy clustering-based recommender systems - LilQuacky/fuzzy-recommender-system

 GitHub