Università degli Studi di Palermo

Fondamenti di Scienza dei Dati: Progetto Gruppo G

Alex Di Corrado, Giampiero Fuschi, Alessandro Botta, Riccardo Schiera, Luca Buccheri

11 Luglio 2024

Indice

1 Obiettivo

A partire dal dataset contenente l'evoluzione della simulazione di una rete idrica, si vuole effettuare la profilazione dei nodi della rete tramite PCA e Clustering K-Means. Si procederà raggruppando prima le misure per nodo (mediando in modo da assegnare ad ogni nodo un'unica riga di misura); successivamente verranno adoperate alcune tecniche di data transformation e data reduction, per potere gestire i dati in modo più efficiente ed ottenere informazioni più veritiere; infine, si eseguiranno PCA e Clustering K-Means, mostrando graficamente i risultati dell'applicazione di queste tecniche

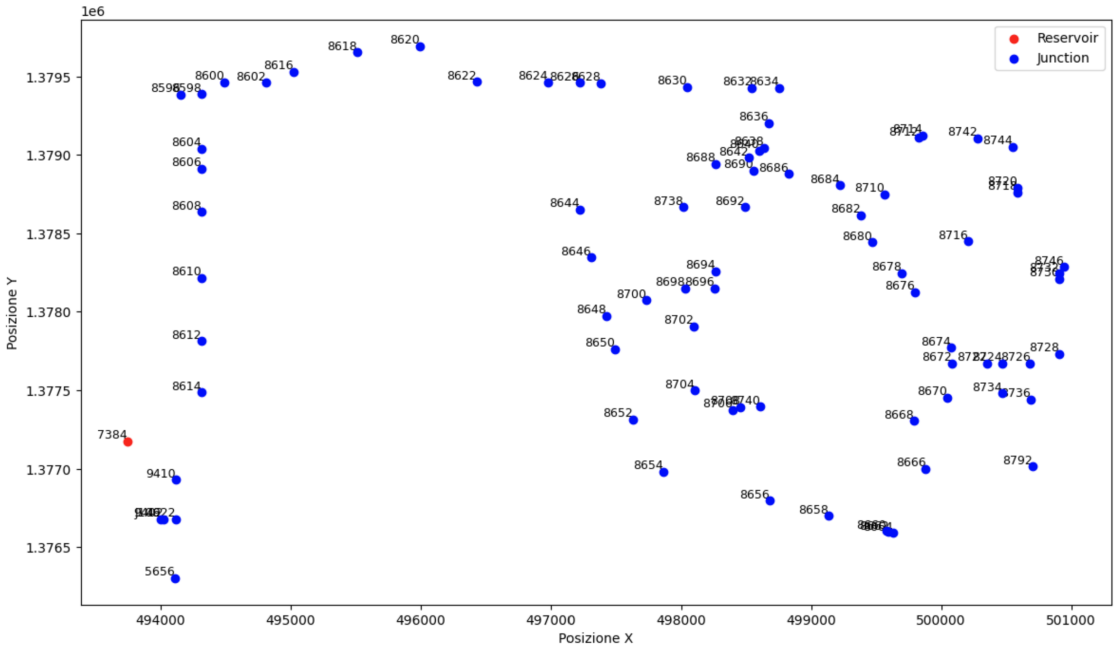

Figura 1: Disposizione dei Nodi Idrici nella Rete Simulata

2 Introduzione e Descrizione del Dataset

Introduzione alla Pipeline del Progetto. Nella pipeline di un progetto di analisi dei dati, la prima fase sarebbe quella di raccolta dei dati (Data Collection). Tuttavia, in questo caso, è stato già fornito un dataset, permettendo di partire direttamente dalla fase di pre-elaborazione dei dati (Data Pre-Processing). Inoltre, verrà saltata la fase di elaborazione analitica (Analytical Processing), poiché non rientra negli obiettivi del progetto e tutte le osservazioni del caso saranno fatte "in loco".

Descrizione del Dataset. Ogni riga del dataset rappresenta le caratteristiche di ciascun nodo della rete idrica a ogni ora, a partire dall'ora zero fino alla 671esima ora. Di seguito vengono elencate le caratteristiche del dataset, accompagnate da una loro descrizione:

Feature	**Descrizione**
hour	L'ora a cui sono stati registrati i dati (formato hh:mm:ss).
nodeID	Identifica il nodo nella rete idrica.
base_demand	Domanda di base al nodo, rappresenta la quantità d'acqua richiesta in condizioni normali.
demand_value	Valore della domanda al nodo in quel momento specifico.
head_value	Valore dell'altezza piezometrica al nodo, che rappresenta l'energia potenziale dell'acqua.
pressure_value	Valore della pressione dell'acqua al nodo.
x_pos	Posizione X del nodo, coordinata in un sistema di riferimento spaziale.
y_pos	Posizione Y del nodo, coordinata in un sistema di riferimento spaziale.
node_type	Indica se il nodo è una giunzione (junction) o un serbatoio (reservoir).
has_leak	Indica se il nodo ha una perdita (True o False).
leak_area_value	Valore dell'area di perdita al nodo.
leak_discharge_value	Valore della portata di perdita al nodo.
leak_demand_value	Valore della domanda di perdita al nodo.
tot_junctions_demand	Domanda totale delle giunzioni nella rete.
tot_leaks_demand	Domanda totale dovuta alle perdite nella rete.
tot_network_demand	Domanda totale della rete idrica.

Tabella 1: Descrizione delle caratteristiche del dataset della rete idrica

3 Data Pre-Processing

Nella pipeline di Data Processing, la fase di Pre-Processing è fondamentale, in quanto si occupa di preparare i dati per la fase di analisi. Ci sono diverse fasi che costituiscono questa seconda "macro-area" della pipeline, e che possono essere prese in considerazione quando si lavora con un set di dati, scegliendo accuratamente quelle più necessarie. Tra queste fasi ci sono la Data Cleaning, la Data Transformation, la Data Integration e la Data Reduction.

3.1 Rimozione Attributi non rilevanti

Osservando il dataset, è stato possibile notare che gli attributi relativi alle perdite sono tutti nulli, poiché la rete idrica non presenta perdite. Di conseguenza, per condurre un'analisi più accurata, è stata presa la decisione di rimuovere questi attributi.

	nodeID	base_demand	demand_value	head_value	pressure_value	x_pos	y_pos	node_type	tot_junctions_demand	tot_network_demand
0	4922	0.006040	0.006040	55.931300	39.167300	494117.01	1376679.17	Junction	0.006040	0.006040
1	5656	0.004547	0.004547	55.931278	39.167278	494110.13	1376302.88	Junction	0.010587	0.010587
2	8596	0.005016	0.005016	55.676176	37.430848	494155.35	1379384.78	Junction	0.015603	0.015603
3	8598	0.006176	0.006176	54.622440	35.426136	494320.25	1379386.96	Junction	0.021779	0.021779
4	8600	0.005701	0.005701	53.582899	34.532899	494495.39	1379463.39	Junction	0.027480	0.027480
...	...	...	...	...	...	...	...	...	...	...
55771	8792	0.006899	0.002156	22.983946	5.232394	500703.60	1377013.26	Junction	0.248834	0.248834
55772	9402	0.003179	0.003179	56.221259	39.457259	494025.60	1376678.69	Junction	0.252013	0.252013
55773	9410	0.007050	0.007050	56.147546	39.383546	494118.61	1376930.61	Junction	0.259063	0.259063
55774	J106	0.002073	0.002073	56.221778	39.671138	494000.44	1376678.87	Junction	0.261136	0.261136
55775	7384	0.000000	-0.261136	56.237282	0.000000	493747.16	1377175.41	Reservoir	0.000000	0.000000

55776 rows × 10 columns

Figura 2: Dataframe prima dell'aggregazione

3.2 Aggregazione

Aggregazione dei Dati per Ridurre la Complessità. Durante questa fase del progetto, si è focalizzati sull'aggregazione dei dati come tecnica di riduzione della mole di input, mantenendo la validità delle analisi. Partendo dal dataset originale, che contiene osservazioni per ogni nodo della rete idrica per ciascuna delle 671 ore considerate, sono stati aggregati i dati con l'obiettivo di ottenere un dataset di dimensioni più contenute.

Metodo. Nello specifico, le misurazioni sono state raggruppate per nodo e sono state calcolate le medie, consentendo di ottenere una singola riga di misurazioni per ciascun nodo. Questa operazione è stata realizzata utilizzando il metodo groupBy di Pandas.

```python
# Seleziona le colonne numeriche
numeric_columns = dataframe.select_dtypes(include=[np.number]).columns.tolist()

# Colonne per il raggruppamento
grouping_columns = ['nodeID', 'node_type']

# Calcola la media delle colonne numeriche, mantenendo 'node_type'
dataframe = dataframe.groupby(grouping_columns)[numeric_columns].mean().reset_index()

# Visualizza il nuovo DataFrame
print("Dataframe dopo l'aggregazione")
display(dataframe)
```

[7] ✓ 0.0s Python

Nel particolare il processo di aggregazione dei dati inizia con l'input di una lista di colonne da utilizzare per il raggruppamento. Successivamente, considerando esclusivamente le colonne numeriche, il sistema calcola la media e procede con il reset dell'indice per garantirne la corretta reimpostazione

	nodeID	node_type	base_demand	demand_value	head_value	pressure_value	x_pos	y_pos	tot_junctions_demand	tot_network_demand
0	4922	Junction	0.005046	0.005046	56.052966	39.288966	494117.01	1376679.17	0.005046	0.005046
1	5656	Junction	0.004791	0.004791	56.052935	39.288935	494110.13	1376302.88	0.009837	0.009837
2	7384	Reservoir	0.000000	-0.237096	56.067862	0.000000	493747.16	1377175.41	0.000000	0.000000
3	8596	Junction	0.004917	0.004917	55.810883	37.565555	494155.35	1379384.78	0.014754	0.014754
4	8598	Junction	0.005089	0.005089	54.803419	35.607115	494320.25	1379386.96	0.019843	0.019843
...	...	...	...	...	...	...	...	...	...	...
78	8746	Junction	0.005112	0.002077	23.331692	6.549404	500947.50	1378286.76	0.220313	0.220313
79	8792	Junction	0.004904	0.001738	23.544991	5.793439	500703.60	1377013.26	0.222051	0.222051
80	9402	Junction	0.005093	0.005093	56.054445	39.290445	494025.60	1376678.69	0.227145	0.227145
81	9410	Junction	0.004912	0.004912	55.995764	39.231764	494118.61	1376930.61	0.232056	0.232056
82	J106	Junction	0.005040	0.005040	56.054869	39.504229	494000.44	1376678.87	0.237096	0.237096

83 rows × 10 columns

Figura 3: Dataframe dopo l'aggregazione

Come evidente, la mole di dati è stata significativamente ridotta: da 55,776 osservazioni si è passati a sole 83, dove ciascun valore numerico rappresenta la media dei dati corrispondenti registrati nelle altre ore.

3.3 Normalizzazione StandardScaler

Durante questa fase, viene applicata una tecnica di trasformazione dei dati nota come normalizzazione, che consiste nel modellare i dati in forme più adatte per le analisi successive. In particolare, si parla della normalizzazione StandardScaler (Z-Score o Z-Mean), un passaggio essenziale da completare prima di procedere al punto successivo, la PCA (Principal Component Analysis).

Questo processo è definito nel seguente modo:

$$v' = \frac{Y - \text{mean}_A}{\sigma_A}$$

dove Y rappresenta il dato da normalizzare, mean_A è la media dell'attributo A, e σ_A è la deviazione standard dell'attributo A.

In altre parole, ogni variabile di input viene trasformata sottraendo la sua media e dividendo per la deviazione standard. Questo processo sposta la distribuzione in modo che abbia una media di zero e una deviazione standard di uno.

```python
# Crea un oggetto scaler
scaler = StandardScaler()

# Definisci le colonne da normalizzare
columns_to_normalize = [col for col in dataframe.columns if col not in ['x_pos', 'y_pos']
                        and dataframe[col].dtype in [np.float64, np.int64]]

# Normalizza le colonne numeriche
dataframe.loc[:, columns_to_normalize] = scaler.fit_transform(dataframe[columns_to_normalize])

# Visualizza il DataFrame normalizzato
print("DataFrame Normalizzato:")
display(dataframe)
```

[9] ✓ 3.5s Python

Spiegazione del codice. Nel codice fornito, si inizia creando un'istanza dell'oggetto 'StandardScaler', appartenente alla classe 'sklearn.preprocessing', utilizzata per la normalizzazione dei dati. Successivamente, viene creato un elenco delle colonne da normalizzare, escludendo quelle specifiche come le coordinate dei nodi idrici o valori non numerici. Infine, la normalizzazione dei dati avviene mediante l'utilizzo del metodo $.fit\_transform()$ applicato alle colonne specificate.

	nodeID	node_type	base_demand	demand_value	head_value	pressure_value	x_pos	y_pos	tot_junctions_demand	tot_network_demand
0	4922	Junction	0.191298	0.192524	2.001247	2.177364	494117.01	1376679.17	-2.099026	-2.099026
1	5656	Junction	-0.269696	0.182775	2.001244	2.177361	494110.13	1376302.88	-2.022352	-2.022352
2	7384	Reservoir	-8.912215	-9.045456	2.002484	-1.139960	493747.16	1377175.41	-2.179789	-2.179789
3	8596	Junction	-0.041938	0.187591	1.981140	2.031849	494155.35	1379384.78	-1.943658	-1.943658
4	8598	Junction	0.268582	0.194158	1.897462	1.866490	494320.25	1379386.96	-1.862209	-1.862209
...	...	...	...	...	...	...	...	...	...	...
78	8746	Junction	0.309387	0.079230	-0.716523	-0.586968	500947.50	1378286.76	1.346168	1.346168
79	8792	Junction	-0.065576	0.066307	-0.698807	-0.650797	500703.60	1377013.26	1.373984	1.373984
80	9402	Junction	0.275585	0.194306	2.001370	2.177488	494025.60	1376678.69	1.455495	1.455495
81	9410	Junction	-0.051777	0.187383	1.996496	2.172534	494118.61	1376930.61	1.534102	1.534102
82	J106	Junction	0.180286	0.192291	2.001405	2.195539	494000.44	1376678.87	1.614768	1.614768

83 rows × 10 columns

Figura 4: Dataset normalizzato

3.4 Principal Component Analysis (PCA)

Importanza e scopo. Anche questo passaggio risulta estremamente importante dal punto di vista della riduzione dei dati (*data reduction*). Questa fase non riduce direttamente il numero di osservazioni, ma piuttosto estrae le componenti principali attraverso la PCA (*Principal Component Analysis* o Analisi delle Componenti Principali).

PCA. La PCA è una tecnica statistica utilizzata per ridurre la dimensionalità dei dati, permettendo di combinare variabili fortemente correlate tra loro, caratterizzate da una covarianza prossima a 1. Il processo include diversi passaggi:

1. Normalizzazione delle osservazioni.
2. Calcolo della matrice di correlazione.
3. Estrazione degli autovalori e degli autovettori tramite SVD (*Singular Value Decomposition*).
4. Ordinamento degli autovettori in base agli autovalori, selezionando le prime k componenti principali.

Il primo passaggio è stato già completato nel punto precedente utilizzando *StandardScaler*. Di solito, i passaggi successivi della PCA vengono eseguiti automaticamente dalle librerie Python. Tuttavia, per scopi analitici, è stata decisa la visualizzazione della matrice di correlazione, calcolata esclusivamente sulle feature di tipo numerico.

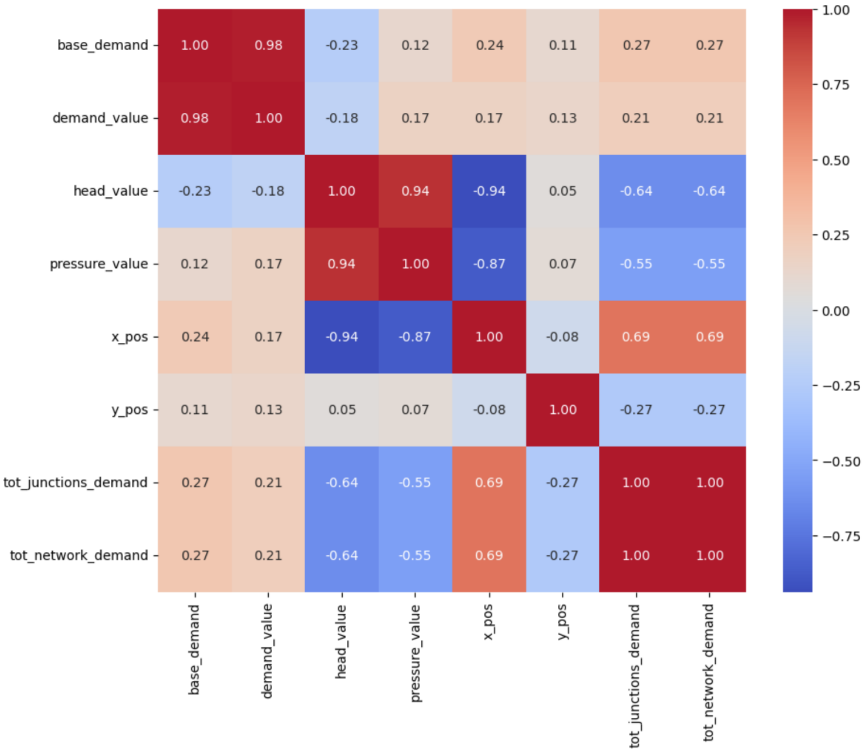

Figura 5: Matrice di Correlazione

Si notano delle forti correlazioni tra le seguenti variabili:

- **base_demand e demand_value**: questo vuol dire che la domanda attesa in condizioni normali si avvicina molto alla domanda media effettivamente richiesta dai nodi nel nostro dataset
- **head_value e pressure_value**: ciò accade perché le due grandezze fisiche sono strettamente legate fra loro. Infatti, la pressione idrostatica è responsabile della variazione di energia potenziale dell'acqua in movimento
- **tot_junctions_demand e tot_network_demand**: questo perché il numero di reservoir è molto inferiore rispetto al numero di junctions. In effetti, abbiamo un solo reservoir su un numero totale di nodi pari a 83
- **x_pos e pressure_value (ed head_value)**: qua abbiamo una covarianza molto vicina a -1, ciò vuol dire che i due attributi crescono in modo proporzionalmente inverso. Ciò ha senso perché all'aumentare della x_pos ci allontaniamo dall'unico reservoir della rete, dunque la pressione dell'acqua diminuisce.

Per applicare la PCA in Python, è possibile importare la classe `PCA` da `sklearn.decomposition` e utilizzare il metodo `fit()`, che calcola autovalori e autovettori:

```python
# Calcola gli autovalori e gli autovettori
pca = PCA()
pca.fit(numeric_df)
```

Python

A questo punto è possibile mostrare lo scree plot, analizzando la relazione tra il numero di componenti principali e la varianza spiegata:

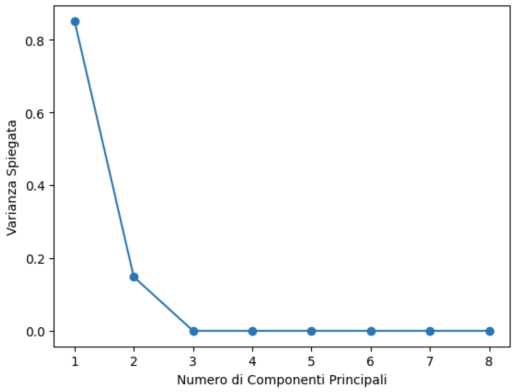

Figura 6: Scree Plot

Elbow method. Per ottenere il k ottimale, utilizzando l'*elbow method*, si cerca il punto di gomito, dopo il quale si nota che aumentare il numero di componenti principali non incrementa significativamente la ricchezza del dataset. Nel caso analizzato, con $k = 2$ si cattura la maggior parte della varianza, ma viene scelto $k = 3$ poiché vi è comunque un significativo incremento della ricchezza del dataset, che non disturba e può fornire ulteriori spunti da un punto di vista analitico.

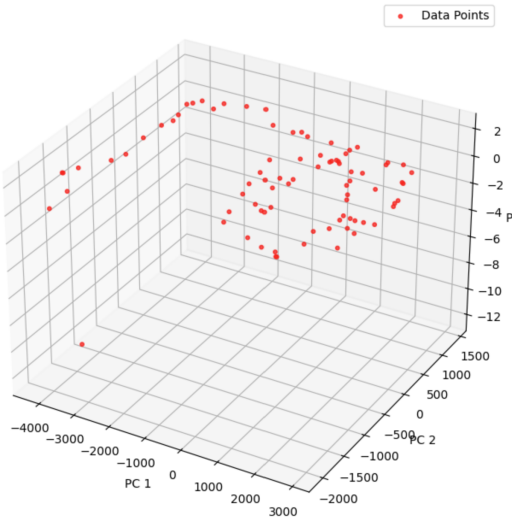

Figura 7: 3D Scatterplot

3.5 Clustering K-Means

A questo punto si procede con il clustering, utilizzando l'algoritmo k-means come tecnica di *data reduction*. L'algoritmo k-means è un metodo di clustering:

- **Partizionale**: suddivide i dati in sottoinsiemi (cluster) non sovrapposti, assicurando che ogni oggetto appartenga esattamente a un cluster. Questo metodo è ideale per dataset di grandi dimensioni e richiede di specificare a priori il numero di cluster.
- **Basato sui Centroidi (Center-Based)**: ogni cluster è rappresentato dal suo centroide, che è la media dei punti nel cluster.

Il funzionamento pratico dell'algoritmo k-means si articola nei seguenti passaggi:

1. **Inizializzazione**: Si seleziona un numero iniziale ottimale di centroidi k utilizzando il *Elbow Method* e il *Silhouette Score*.
2. **Assegnazione dei Punti**: Ogni punto dati viene assegnato al centroide più vicino, formando così k cluster.
3. **Aggiornamento dei Centroidi**: Per ogni cluster, si calcola un nuovo centroide come media dei punti assegnati a quel cluster.
4. **Iterazione**: I passaggi di assegnazione e aggiornamento vengono ripetuti fino a quando i centroidi non subiscono più cambiamenti significativi o fino al raggiungimento di un numero massimo prestabilito di iterazioni.

Si utilizza il metodo Elbow per trovare il k ottimale, osservando il WSS (Within-Cluster Sum of Squares) e il Silhouette Score.

```python
wss = []
list_k = list(range(1, 11))  # Testiamo da 1 a 10 cluster

for k in list_k:
    km = KMeans(n_clusters=k, random_state=42)
    km.fit(reduced_df)
    wss.append(km.inertia_)  # WSS per ogni n_clusters
```

Python

Spiegazione del codice. Viene inizializzata una lista che conterrà tutti i valori dello score WSS in corrispondenza del numero di cluster (valori da 1 a 10 presenti in `list_k`). Successivamente, si itera sulla lista contenente i valori dei cluster e per ogni k si esegue l'algoritmo k-means, valutando la sua coesione interna attraverso il WSS (restituito dall'attributo `.inertia_`).

Analisi del grafico. Analizzando il grafico ottenuto mettendo in relazione il numero k di cluster con il WSS, si ottiene un k ottimale pari a 3. Sebbene sia possibile continuare ad aumentare il valore di k, la maggior parte della minimizzazione del WSS viene raggiunta fermandosi a 3.

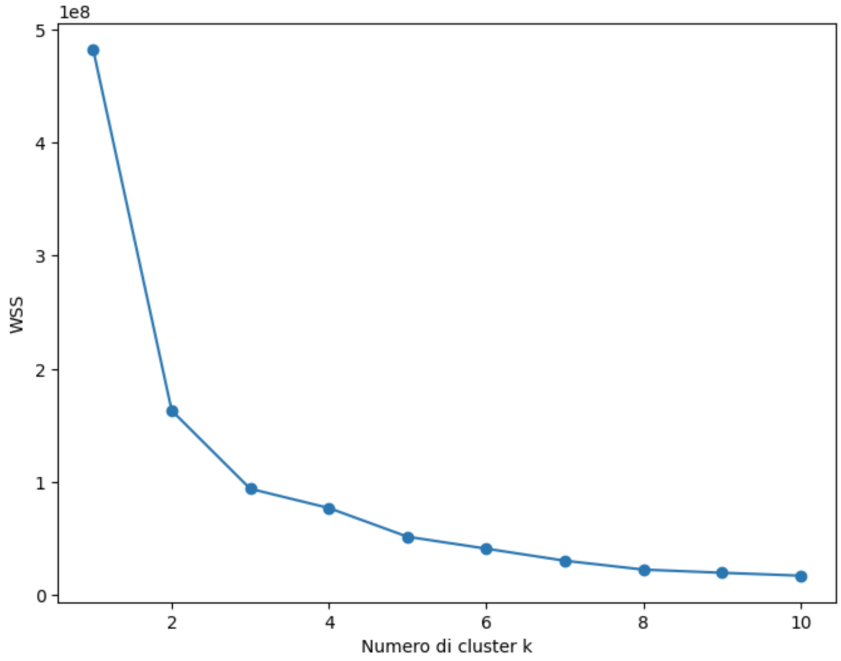

Figura 8: WSS per determinare il numero ottimale di cluster

Tuttavia, utilizzando il coefficiente di silhouette al posto del WSS, si ottiene un risultato leggermente diverso. In questo caso, infatti, si tiene conto non solo della coesione (che deve essere minimizzata) ma anche della separazione (che deve essere massimizzata). Si osserva così che il numero ottimale k di cluster è 2.

Valutazione del coefficiente di Silhouette. La valutazione del Silhouette score è analoga a quella del WSS, ma utilizza uno score differente. In questo caso, anziché il WSS, si calcola il coefficiente di Silhouette utilizzando la formula:

$$\frac{b-a}{\max\{a,b\}}$$

dove a rappresenta la distanza media tra un punto e tutti gli altri punti nel suo stesso cluster (coesione interna), mentre b è la distanza media tra il punto e i punti di un altro cluster (separazione da altri cluster). Il coefficiente varia tra -1 e 1, dove un valore elevato indica un cluster ben coeso e separato dagli altri cluster, indicando quindi un'alta similarità intra-cluster e una bassa similarità inter-cluster.

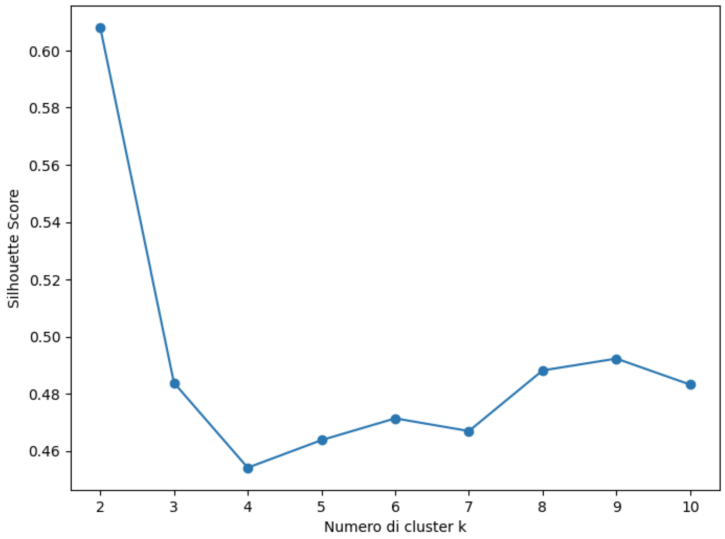

Figura 9: Silhouette Score per determinare il numero ottimale di cluster

Viusalizzazione 3D. A questo punto viene applicato l'algoritmo k-means con $k = 2$ per ottenere una visualizzazione 3D del clustering e dei centroidi.

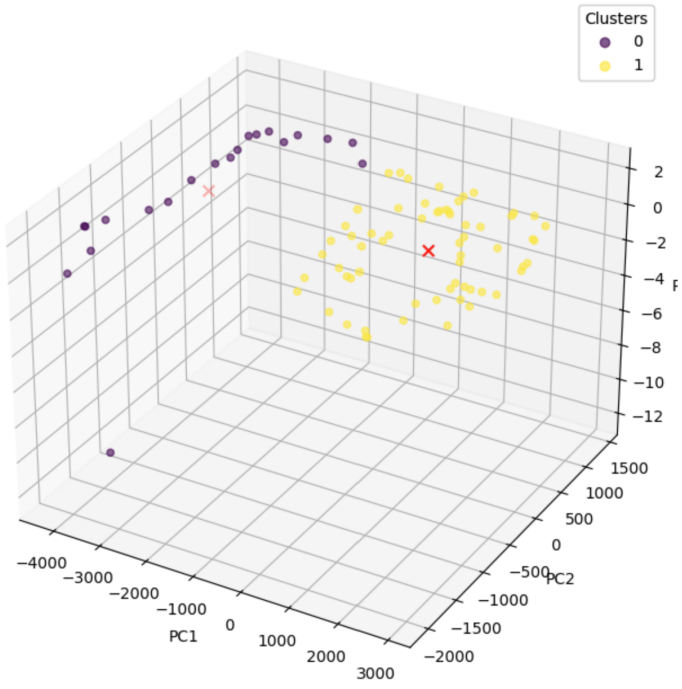

Figura 10: Visualizzazione 3D dei Cluster e Centroidi