UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea Magistrale in
Ingegneria Informatica e dell'Automazione

Analisi sul costo della vita nel mondo e sulle azioni di Netflix tramite l'utilizzo di librerie di Python

Studenti:
Daniele Pallini 1107326
Matteo Abbruzzetti 1108842

Docenti:
Domenico Ursino
Gianluca Bonifazi
Michele Marchetti

Anno Accademico 2022-2023

Indice

Capitolo 1

Introduzione

In questo documento verranno presentate le analisi di due dataset: il primo contiene dati riguardanti il costo della vita in diverse città situate in tutto il mondo; il secondo, invece, contiene valori storici delle azioni della società di distribuzione in streaming di film e serie televisive, *Netflix*, dal 2012 al 2022.
I dataset sono presenti e visualizzabili sul sito *Kaggle* ai seguenti link:

- Global cost of living: https://www.kaggle.com/datasets/mvieira101/global-cost-of-living?select=cost-of-living.csv
- Netflix: https://www.kaggle.com/datasets/whenamancodes/netflix-stock-market-analysis-founding-years

Sono stati scelti due dataset diversi in quanto si effettueranno due tipi di analisi differenti: il dataset relativo alle città e al loro costo della vita è stato scelto per la parte di Clustering e Classificazione, mentre quello contenente informazioni relative alle azioni di Netflix servirà per l'analisi delle serie temporali.

Le analisi sopracitate saranno realizzate attraverso il linguaggio di programmazione *Python*. Il motivo principale per cui si è scelto questo linguaggio è che può essere arricchito e semplificato attraverso librerie specializzate in questi tipi di analisi come *Pandas*, *Seaborn*, *Matplotlib*, *Scikitlearn* e *Statsmodels*.

Capitolo 2

Dataset

2.1 Dataset sul costo della vita globale

Come già accennato nel capitolo introduttivo, il dataset utilizzato per clustering e classificazione è quello contenente informazioni riguardanti il costo della vita globale. Raccoglie una collezione di dati su quasi 5000 città situate in tutto il mondo. I dati presenti in esso sono stati raccolti dal sito *Numbeo*, il database più grande al mondo riguardo il costo della vita. Il dataset è strutturato in 4873 righe, che rappresentano le città, e in 59 colonne, che sono i loro attributi. Nella seguente tabella sono riportati gli attributi e una loro breve descrizione:

Lista degli attributi		
Nome	Tipo	Descrizione
id	int	Codice identificativo della città
city	string	Nome della città
country	string	Nome della nazione
x1	float	Costo di un pasto economico (in dollari)
x2	float	Costo di un pasto per 2 in un ristorante di media qualità (in dollari)
...	...	...
...	...	...
x55	float	Tasso di interesse su un mutuo (in percentuale)
data_quality	int	Qualità del dato, 0 se Numbeo considera che servano più contributori per aumentare la qualità del dato, 1 altrimenti

La tabella del dataset è, quindi, formata da 55 attributi che indicano il costo (in dollari americani, USD) di 55 diversi prodotti e forniture, come ad esempio pane, pollo, carburante, connessione a Internet, appartamenti e molto altro.

2.1.1 ETL

Il dataset, inizialmente, risultava "sporco" e pieno di dati mancanti. Si è resa necessaria una fase di ETL (Extract, Transform, Load), nella quale si sfruttano le funzioni messe a disposizione dalla libreria **Pandas** per effettuare le operazioni di ripulitura del dataset. Il primo step è stato quello di eliminare le righe in cui erano presenti delle città duplicate, alcune colonne relative a prodotti o forniture poco interessanti o contenenti troppi valori NULL e la colonna *data_ quality*, che non risulta utile per il nostro scopo. Successivamente, è stata impostata la colonna *city* come indice e sono state rinominate le colonne dei prodotti, in modo da identificarle in base al contenuto. Infine, si è deciso di eliminare i valori nulli rimanenti rimpiazzandoli con la media dei corrispettivi prodotti (Figura 2.1). Questa operazione è stata eseguita sulle colonne che avevano relativamente pochi valori nulli, in modo tale da non falsificare i risultati delle analisi. Le colonne con un alto numero di valori nulli sono state, invece, rimosse.

```
df.isnull().sum()
```

```
country                0
Pasto Economico        0
Pasto per 2            0
Pasto McDonald         0
Birra Locale           0
Birra Estera           0
Cappuccino             0
Cola                   0
Acqua                  0
Latte                  0
Pane                   0
Riso                   0
Uova                   0
Formaggio Locale       0
Pollo                  0
Manzo                  0
Mele                   0
Banane                 0
Arance                 0
```

Figura 2.1: Valori nulli per prodotto

Il dataset finale, su cui si è svolta l'analisi, è formato da 4816 righe, ossia le città considerate, e 34 colonne, di cui una relativa al nome della città, una riguardo la nazione di appartenenza e 32 contenenti il costo medio dei prodotti nella città di riferimento. In Figura 2.2 è riportato l'elenco dei nomi assegnati alle colonne dei prodotti considerati.

```
df.rename(columns =
          {'x1':'Pasto Economico', 'x2':'Pasto per 2','x3':'Pasto McDonald', 'x4':'Birra Locale',
           'x5':'Birra Estera','x6':'Cappuccino', 'x7':'Cola','x8':'Acqua', 'x9':'Latte',
           'x10':'Pane', 'x11':'Riso','x12':'Uova', 'x13':'Formaggio Locale','x14':'Pollo',
           'x15':'Manzo','x16':'Mele', 'x17':'Banane','x18':'Arance','x19':'Pomodori',
           'x20':'Patate','x21':'Cipolle','x22':'Lattuga','x23':'Acqua (1.5 lt)','x24':'Vino',
           'x25':'Birra Locale (Supermarket)','x26':'Birra Estera (Supermarket)',
           'x27':'Sigarette','x33':'Carburante','x36':'Utenze Domestiche (85 mq)',
           'x38':'Internet','x41':'Cinema','x44':'Jeans'}, inplace = True)
```

Figura 2.2: Elenco dei prodotti considerati

2.1.2 Analisi Descrittiva

Il primo passo è stato cercare di avere un'idea della tipologia di dataset considerato per le attività di clustering e classificazione. Per questo motivo, è stata effettuata una breve analisi descrittiva.
In Figura 2.3 è rappresentato un boxplot relativo al prezzo di un paio di jeans. Ogni box racchiude i valori delle varie città e paesi appartenenti allo stesso continente. Il prezzo tra i diversi continenti risulta notevolmente diverso; infatti, si passa da un prezzo medio di 25$ in Africa ai quasi 75$ in Europa. Da evidenziare è anche la presenza di *outliers*, ovvero valori fuori scala, in 3 continenti. Allo stesso modo, in Figura 2.4 si rappresenta la situazione relativa al prezzo di un pasto per 2 persone. Si può notare come l'andamento complessivo risulti simile al precedente, la differenza tra i vari continenti ha una fisionomia paragonabile a quella vista in Figura 2.3. Tuttavia, in questo caso risulta esserci un numero maggiore di outliers.

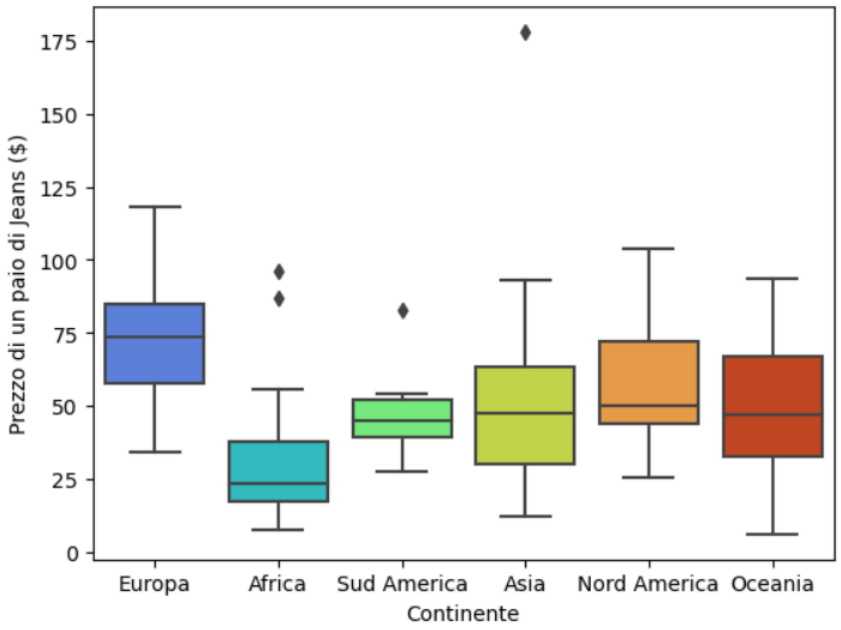

Figura 2.3: Boxplot relativo al prezzo dei jeans nei vari continenti

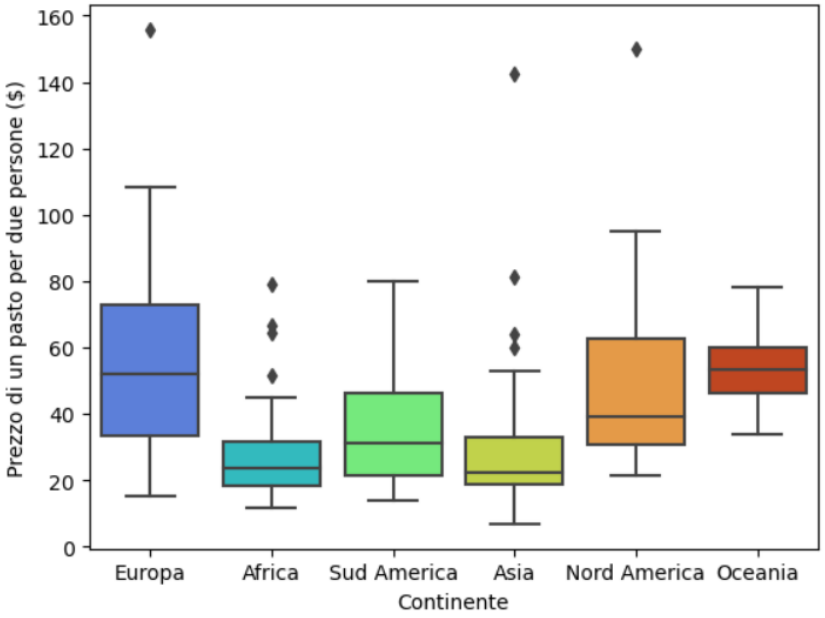

Figura 2.4: Boxplot relativo al prezzo di un pasto per 2 persone nei vari continenti

È stata posta, poi, l'attenzione sulle singole città. In Figura 2.5 sono rappresentate le 10 città a livello globale aventi il prezzo più alto per 1 kg di pollo. Da evidenziare, in questo caso, è la presenza di molte città appartenenti alle stesse nazioni, fattore che verrà analizzato più in dettaglio nelle operazioni di clustering e classificazione.
In Figura 2.6 è, invece, presente la classifica delle città più costose al mondo per prezzo di un kg di pane. A differenza di quanto visto per il pollo, una particolarità interessante è che tutte e 10 le città elencate appartengono agli Stati Uniti d'America.
Gli attributi *pollo* e *pane* saranno ripresi nelle operazioni relative al clustering bidimensionale, in quanto considerati beni di prima necessità e quindi buoni indicatori del costo della vita nelle città.

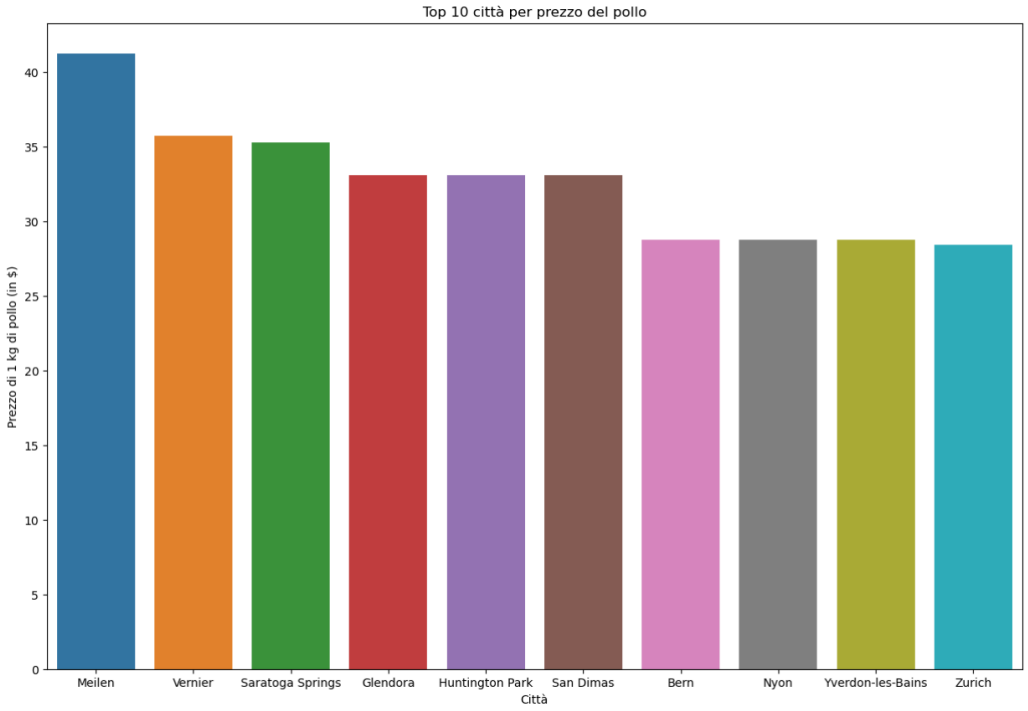

Figura 2.5: Top 10 città per prezzo del pollo

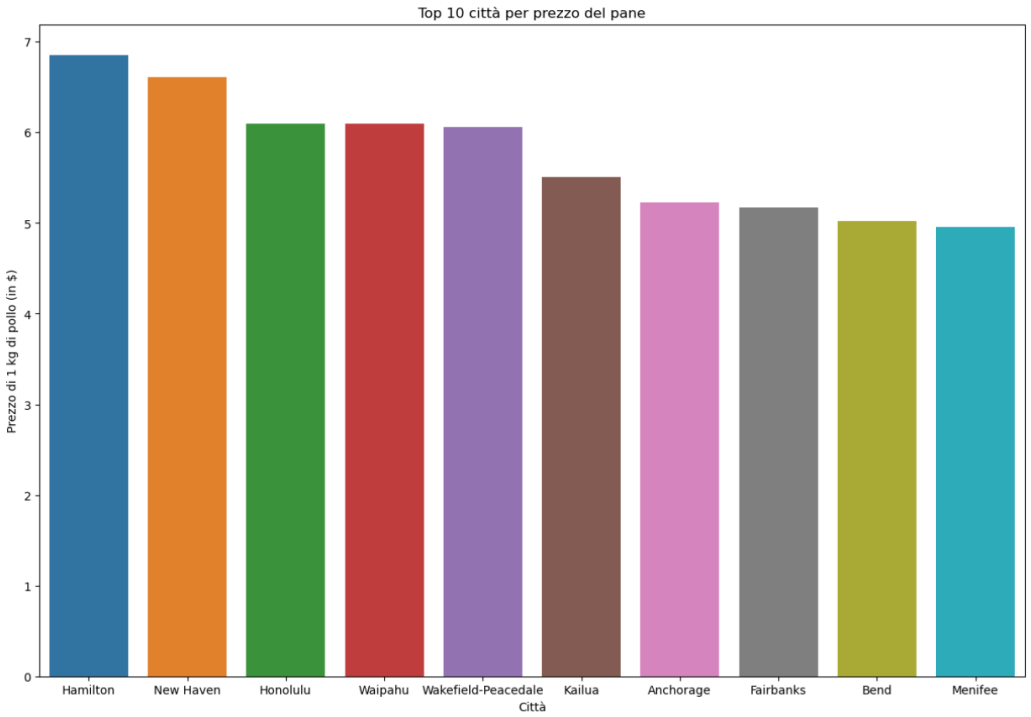

Figura 2.6: Top 10 città per prezzo del pane

2.2 Dataset sull'andamento delle azioni di Netflix

Il dataset scelto per l'analisi delle serie temporali riguarda l'andamento delle azioni in Borsa di Netflix degli ultimi 20 anni (dal 2002 al 2022). Il dataset è costituito da circa 5000 righe, rappresentanti i giorni, e in 7 colonne, corrispondenti agli attributi. Nella seguente tabella sono riportati gli attributi e una loro breve descrizione:

Lista degli attributi	
Nome	Descrizione
Date	Data di riferimento
Open	Prezzo all'apertura dei mercati
High	Prezzo più alto registrato nel giorno di riferimento
Low	Prezzo più basso registrato nel giorno di riferimento
Close	Prezzo alla chiusura dei mercati
Adj Close	Prezzo di chiusura modificato in base alle azioni societarie
Volume	Numero di azioni vendute nel giorno di riferimento

La tabella del dataset è, quindi, formata dalla data di riferimento e da 6 attributi che riassumono l'andamento delle azioni di Netflix nel corso della giornata.

2.2.1 ETL

Si è resa necessaria una breve fase di ETL per preparare il dataset all'analisi. Innanzitutto, si è scelto di concentrarsi sugli anni successivi al 2012, in quanto si è ritenuto fossero troppi 20 anni per un'analisi efficace su questa tipologia di argomento. Inoltre, tramite la funzione di *Pandas* **to_datetime()**, l'attributo *Date* è stato convertito nel tipo *datetime* ed è stato reso indice della tabella. A questo punto, si è deciso di raggruppare i dati per mese, tramite il metodo **resample()**. In questo modo si è riusciti ad avere una frequenza mensile costante, e non una frequenza giornaliera intervallata dai week-end, in cui i dati risultavano assenti. Gli attributi *Open, High, Low, Close e Adj Close* sono stati raggruppati tramite media, in quanto sono corrispondenti a dei prezzi. L'attributo *Volume*, invece, è stato raggruppato tramite somma, in modo da rappresentare la totalità delle azioni vendute nel mese specifico.

Capitolo 3

Classificazione e Clustering

3.1 Clustering

Le attività di clustering sono state eseguite sul dataset relativo al costo della vita. Il clustering consiste in una serie di tecniche e metodi che hanno lo scopo di raggruppare oggetti con caratteristiche simili negli stessi gruppi. In questo modo si riesce ad avere una suddivisione in cluster degli elementi del dataset senza avere una conoscenza approfondita di ognuno di essi.
L'obiettivo della nostra analisi sui cluster è quello di dividere le città in base al costo della vita. I parametri scelti per la clusterizzazione sono stati il costo medio del pollo e del pane. La scelta è motivata dal fatto di aver trovato un buon compromesso tra una correlazione non troppo alta tra i due prodotti, pari a 0.55, e la necessità di studiare il costo della vita su beni di prima necessità, come, nello specifico, il pane e il pollo. Per realizzare le attività di clustering sono stati utilizzati i modelli e i metodi della libreria *Scikitlearn*.

3.1.1 Clustering bidimensionale

Algoritmo K-means

Per effettuare un'attività di clustering sui due parametri sopracitati, in primo luogo è stato applicato l'algoritmo *K-means*. L'algoritmo, tuttavia, richiede che sia noto il parametro k, il quale indica il numero di cluster in cui dividere gli elementi del dataset. Innanzitutto, è stato necessario individuare il valore ideale di k e per farlo si è sfruttato l'*elbow method*. Il metodo itera l'algoritmo K-means in modo tale da calcolare i diversi distortion score al variare del numero k. Tali punteggi, in funzione della variabile k, formano una curva e il punto in cui si trova il "gomito" di essa indica il numero ideale di cluster per il dataset di studio. Nel nostro caso, l'elbow method ha indicato il valore 3 come numero di cluster opportuno in cui suddividere le città del dataset (Figura 3.1).

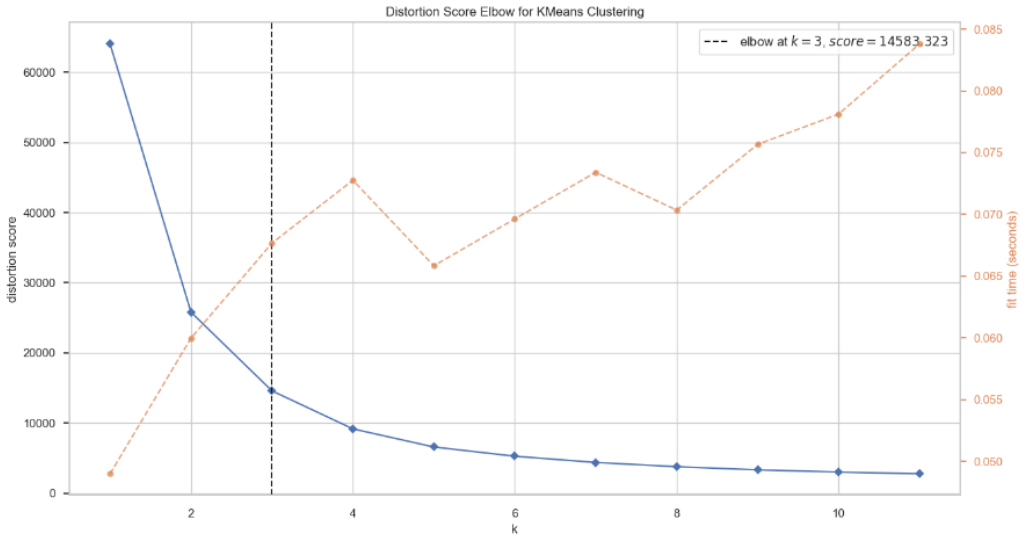

Figura 3.1: Applicazione dell'elbow method

Successivamente, si è generato lo scatterplot scaturito dal risultato della clusterizzazione (Figura 3.2). Ogni città del dataset è stata assegnata ad uno dei tre cluster. Sulla base dei risultati ottenuti, il primo cluster (cluster 0) è stato destinato alle città considerate aventi un costo della vita medio, il secondo cluster (cluster 1) sembra raggruppare le città reputate come soggette ad un costo della vita basso e, infine, il terzo cluster (cluster 2) pare raccogliere quelle dove il costo della vita risulta essere alto.

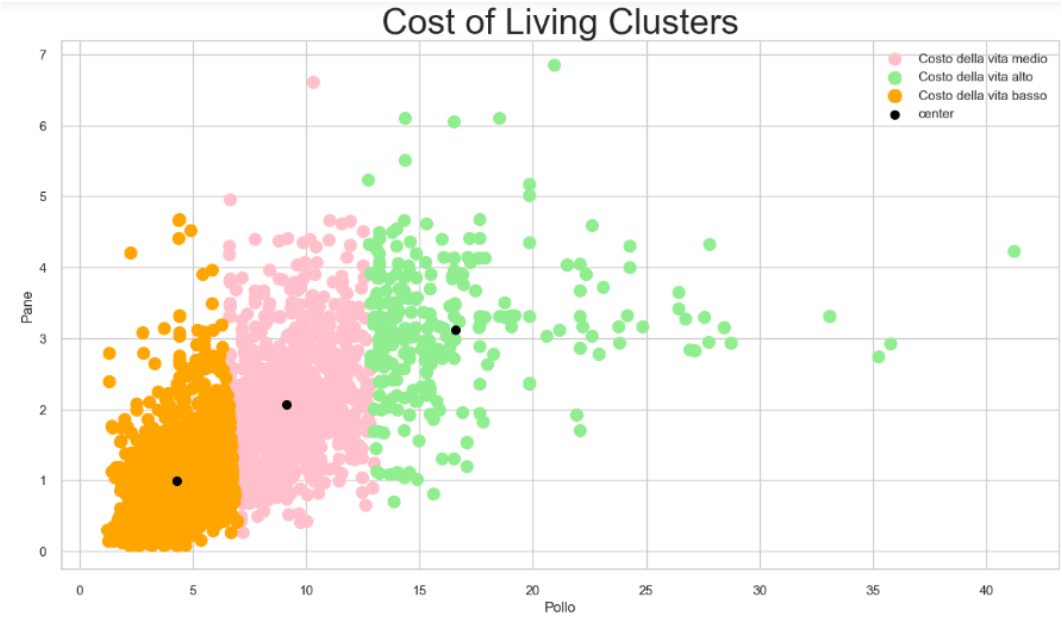

Figura 3.2: Clustering derivato da K-Means

Attraverso il modello K-means, è stato generato anche il grafico relativo alla metrica *silhouette* (Figura 3.3), che associa ad ogni elemento di un cluster un valore compreso tra -1 e 1 in base al livello di similarità che ha con gli altri elementi dello stesso gruppo.

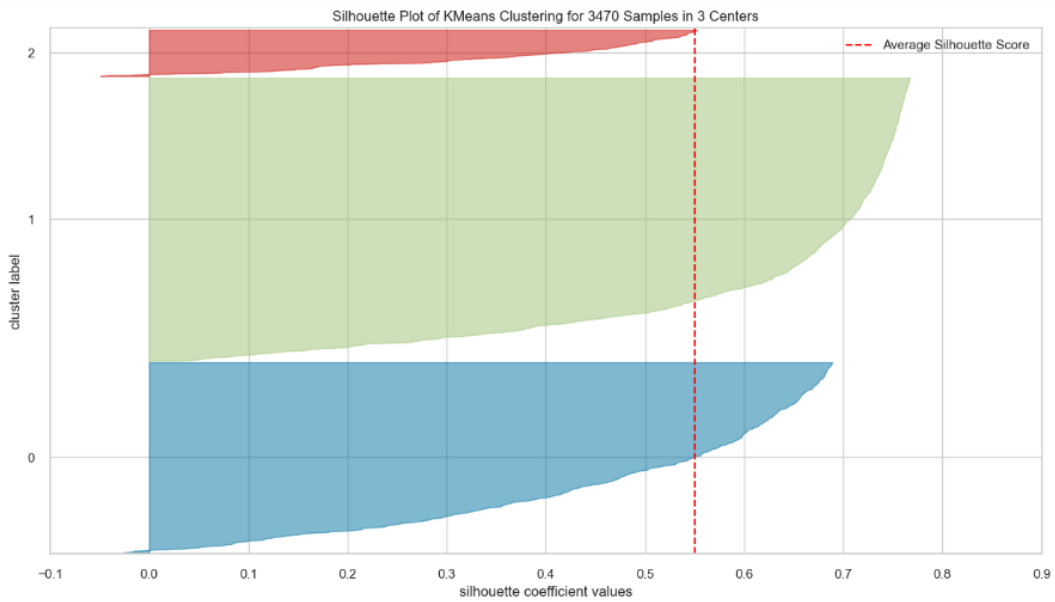

Figura 3.3: Grafico relativo alla silhouette

Come si può evincere dalla Figura 3.3, la media dei valori della metrica silhouette risulta essere circa 0.55, che può essere considerato un valore soddisfacente. Si può, quindi, affermare che gran parte degli elementi sono stati assegnati correttamente al cluster di appartenenza.

Per ricevere un'ulteriore conferma del fatto che l'opzione di dividere il dataset in 3 cluster costituisse la soluzione migliore (come suggerito dall'applicazione dell'elbow method), si è realizzata un tentativo ponendo il numero di cluster uguale a 4. Come da aspettativa, i risultati ottenuti da questa alternativa sono stati meno soddisfacenti. Lo scatterplot (Figura 3.4) che ne deriva posiziona le città in 4 gruppi, che abbiamo etichettato come costo della vita basso (0), medio (1), alto (2) e molto alto (3).

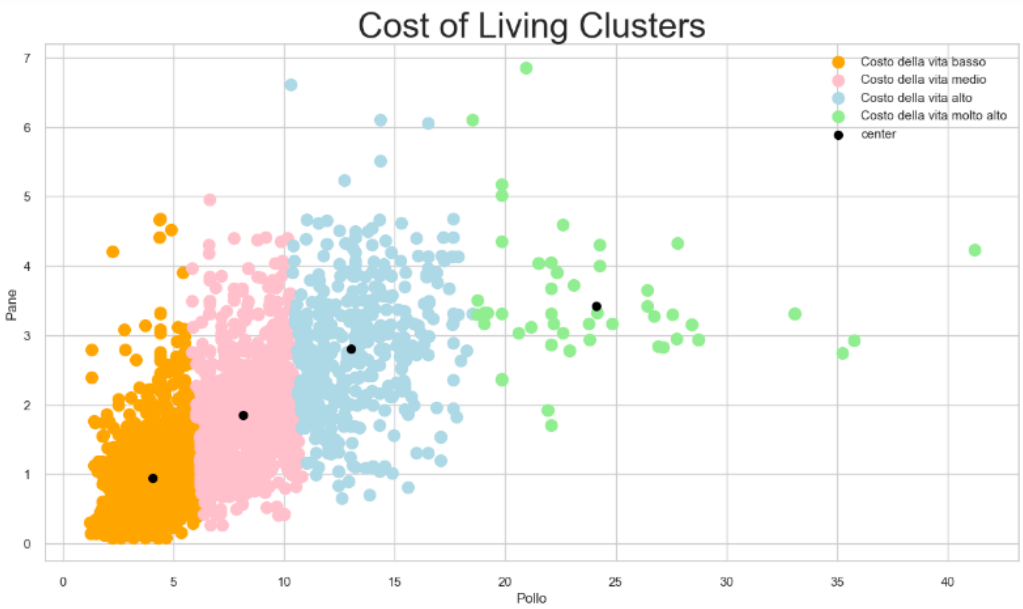

Figura 3.4: Clustering con 4 cluster

Ciò che salta subito all'occhio è che il cluster 3, contenente le città che hanno un costo della vita molto alto, risulta molto meno popolato rispetto agli altri, a riprova del fatto che una suddivisione delle città in un cluster aggiuntivo risulti poco vantaggiosa. Inoltre, anche i risultati del metodo silhouette risultano leggermente più bassi di quelli ottenuti con 3 cluster (Figura 3.5).

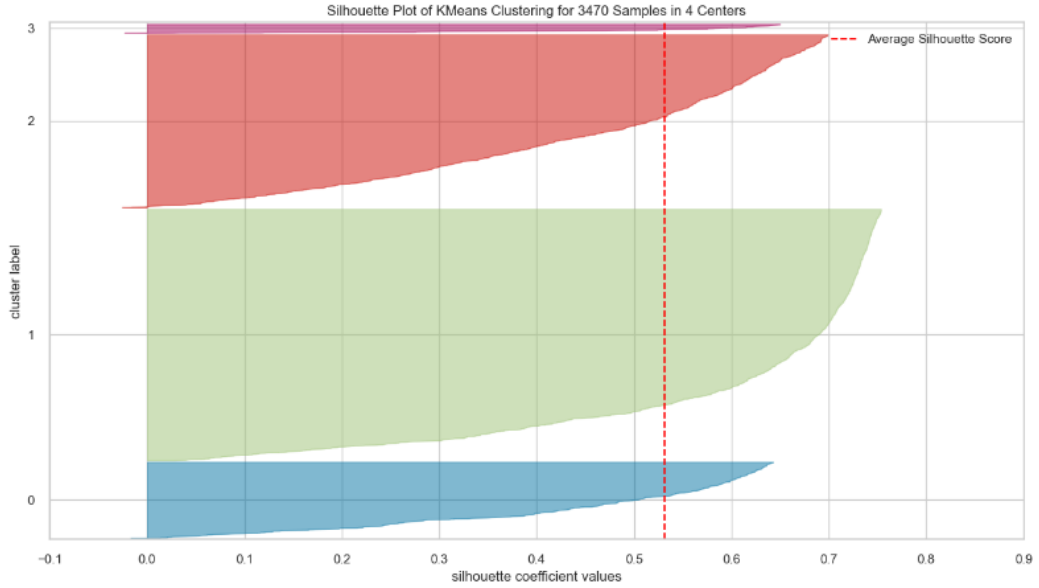

Figura 3.5: Silhouette con 4 cluster

Clustering gerarchico

Per non fare troppo affidamento sui soli risultati dell'algoritmo K-means, si è deciso di realizzare un clustering gerarchico agglomerativo. Tale cluster è realizzato sfruttando un approccio bottom-up in cui si parte da un certo numero di cluster N e si procede all'accoppiamento graduale di cluster a due a due fino a raggiungere il numero di cluster richiesto. Lo scatterplot relativo ai cluster gerarchici è stato generato dal modello *AgglomerativeClustering*. Come si può constatare dal grafico in Figura 3.6, i risultati sono piuttosto simili a quelli ottenuti dal cluster realizzato dall'algoritmo K-means, a ulteriore conferma dell'affidabilità dei cluster generati nel paragrafo precedente.

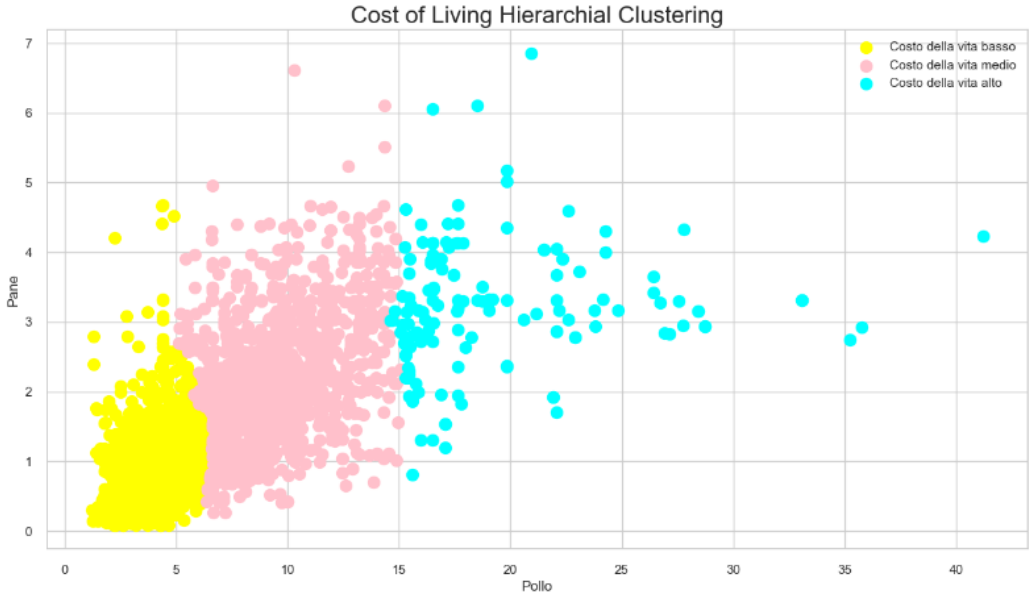

Figura 3.6: Clustering gerarchico

3.1.2 Clustering bidimensionale con normalizzazione

L'attività di clustering eseguita sui parametri *pollo* e *pane* ha generato un problema, derivante dal fatto che il prezzo del pollo incidesse maggiormente rispetto a quello del pane (in quanto il primo è significativamente più alto). Come si può notare, ad esempio, in Figura 3.2, le città hanno il relativo costo della vita che dipende fortemente dal prezzo del pollo. Esistono città, considerate con costo della vita medio, che hanno un prezzo del pane sei volte maggiore rispetto ad una città definita altamente costosa, perché il suo prezzo del pollo è più alto di pochi dollari. Questa problematica è evidenziata dai grafici delle distribuzioni dei cluster, realizzati sfruttando i metodi delle librerie *Matplotlib* e *Seaborn*, visualizzabili in Figura 3.7.

Per cercare di bilanciare i due parametri, si è effettuata un'operazione di normalizzazione su di essi. La funzione *StandardScaler* di Scikitlearn permette di eseguire facilmente questa azione; vengono calcolate media e deviazione standard delle due feature e utilizzate per assegnare dei nuovi valori ai campi dei due parametri, in modo che la media sia uguale a 0 e la deviazione pari a 1 per entrambi (pane e pollo). A seguito di tale operazione, si può notare come i grafici delle distribuzioni dei cluster del pane (Figura 3.8) siano migliorati notevolmente, evidenziando una netta delimitazione dei diversi cluster in base prezzo di tale prodotto.

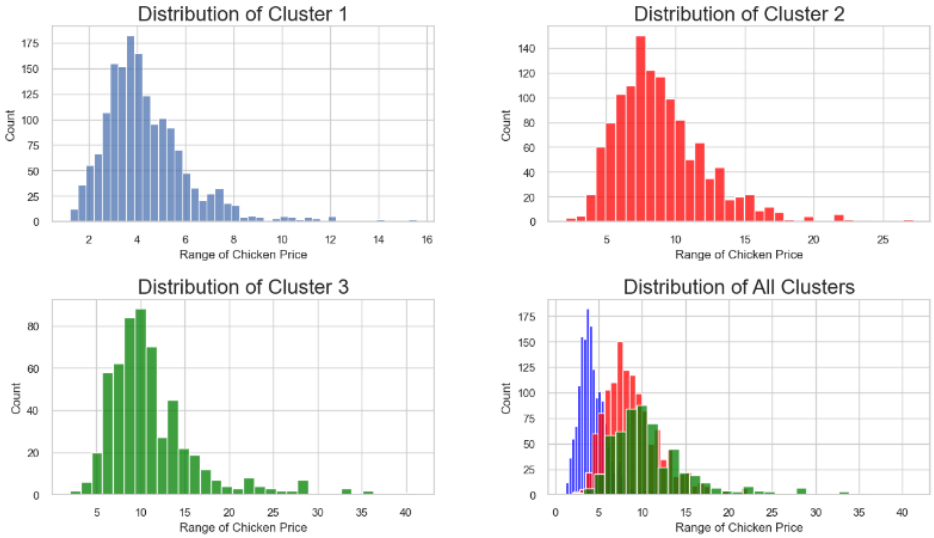

Figura 3.7: Distribuzioni dei cluster relative al prezzo del pane

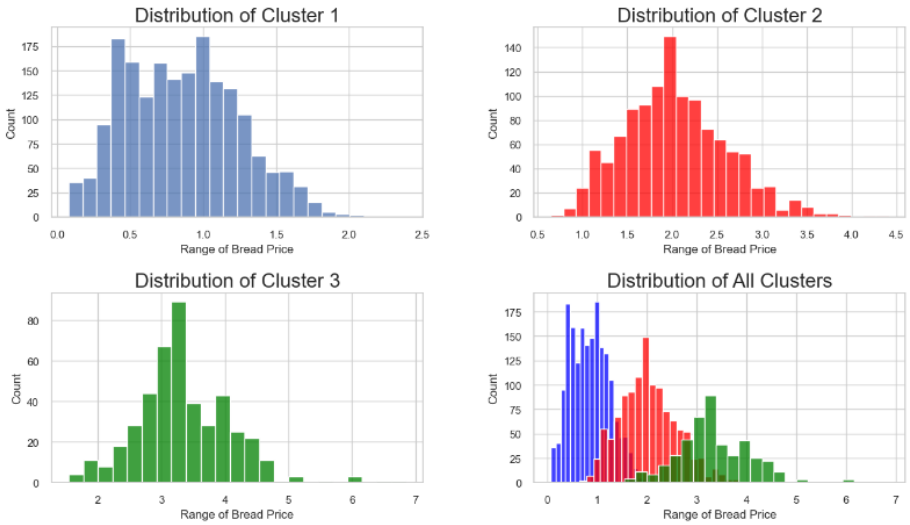

Figura 3.8: Distribuzioni dei cluster relative al prezzo del pane normalizzato

Di conseguenza, anche i grafici relativi allo scatterplot generato tramite l'algoritmo K-means (Figura 3.9) e quello prodotto con il metodo dell'AgglomerativeClustering (Figura 3.10) risultano dipendenti da entrambe le misure.

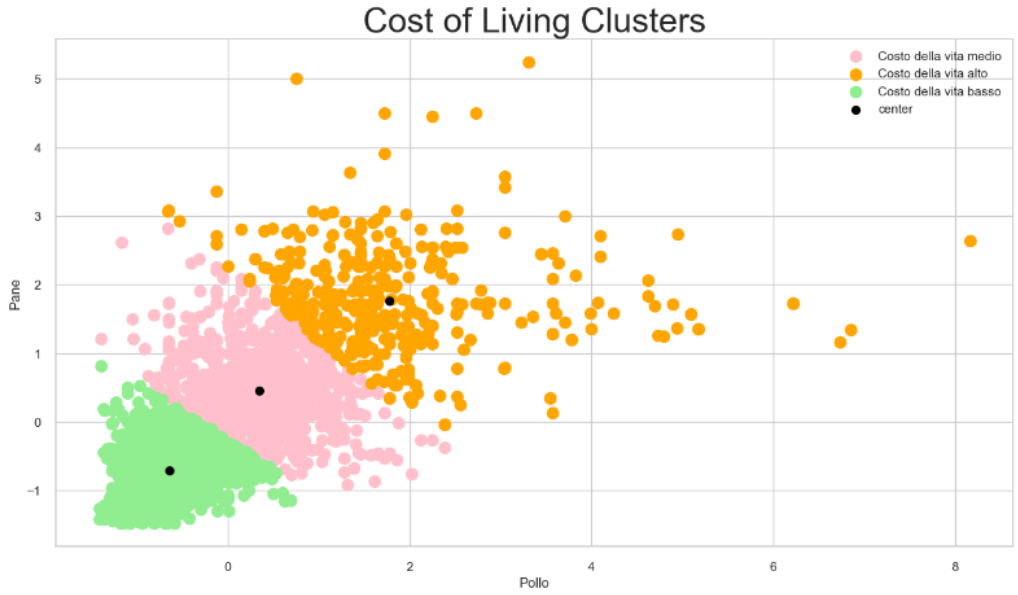

Figura 3.9: Clustering normalizzato derivato da K-means

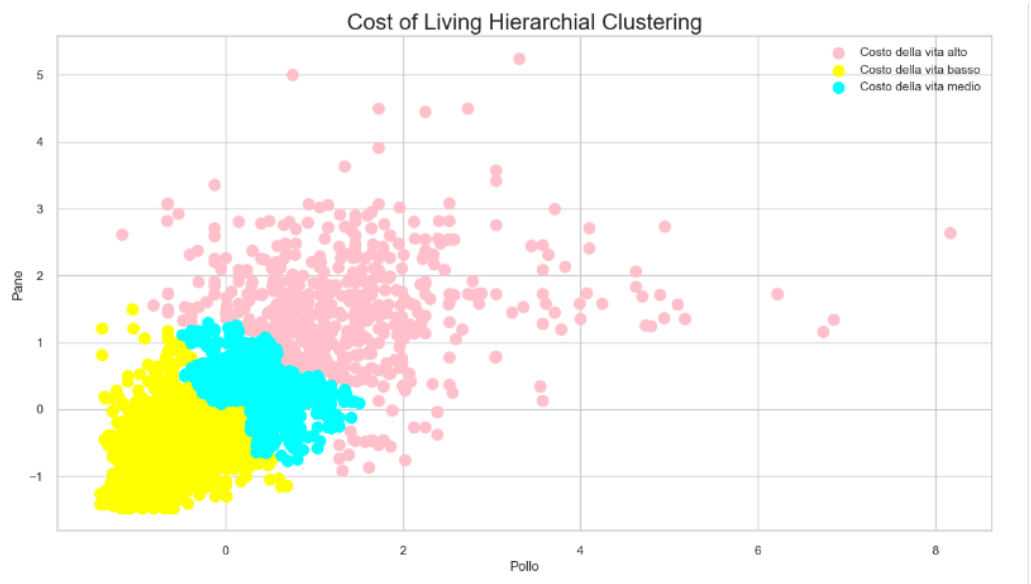

Figura 3.10: Clustering gerarchico normalizzato

L'unica nota negartiva individuata a seguito della normalizzazione è nel grafico relativo alla silhouette (Figura 3.11). Infatti, si può notare come la media

dei valori della metrica silhouette, che si aggira intorno ad un valore di 0.46, sia inferiore rispetto alla media calcolata senza normalizzazione.

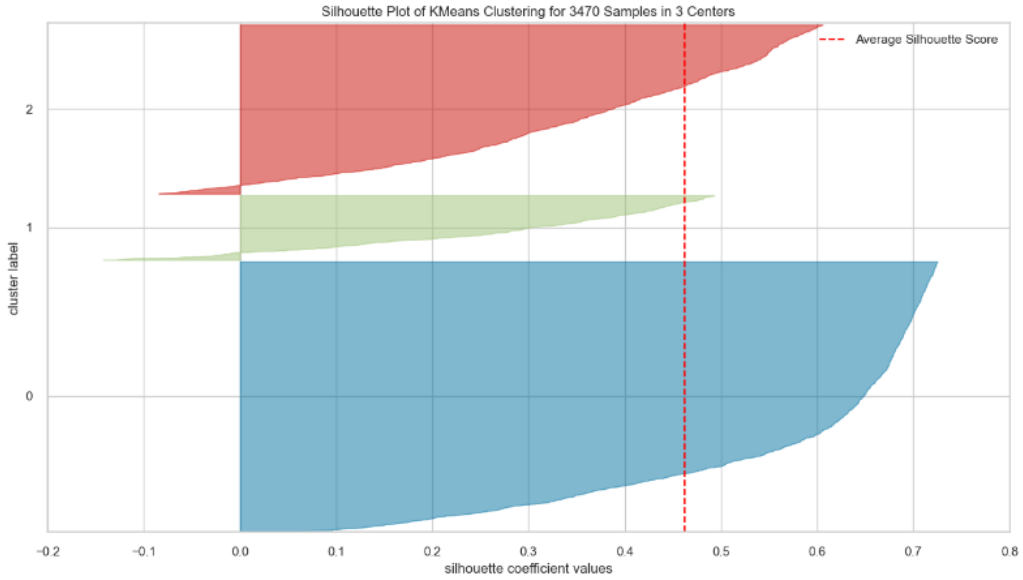

Figura 3.11: Silhouette con parametri normalizzati

3.1.3 PCA

Fino ad ora sono stati analizzati cluster scaturiti dallo studio di due features. Il passo successivo è stato quello di effettuare un clustering di dimensione N attraverso il metodo della *PCA* (*Principal Component Analysis*). La PCA è una tecnica usata per ridurre la dimensionalità di un insieme di dati complesso; si cerca di trovare una combinazione lineare di queste variabili che possa spiegare la maggior parte della varianza nei dati.
A differenza di quanto succede nel clustering bidimensionale, nella PCA la normalizzazione non risulta essere la strategia vincente, come si può subito notare dal grafico della silhouette (Figura 3.12), il quale risulta avere una media dei valori decisamente troppo bassa (intorno a 0.25) e con molti valori negativi.

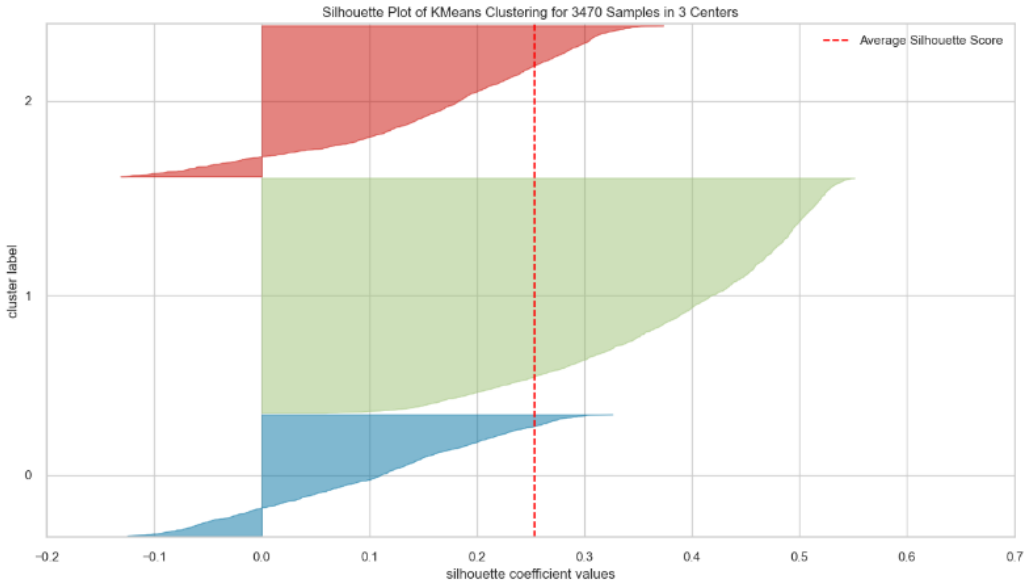

Figura 3.12: Silhouette della PCA con dataset normalizzato

Motivo per cui si è deciso di effettuare una PCA sul dataset originale.
Anche in questo caso, il primo step è stato trovare il numero di cluster ideale tramite l'applicazione dell'elbow method (Figura 3.13), che è risultato essere nuovamente pari a 3.

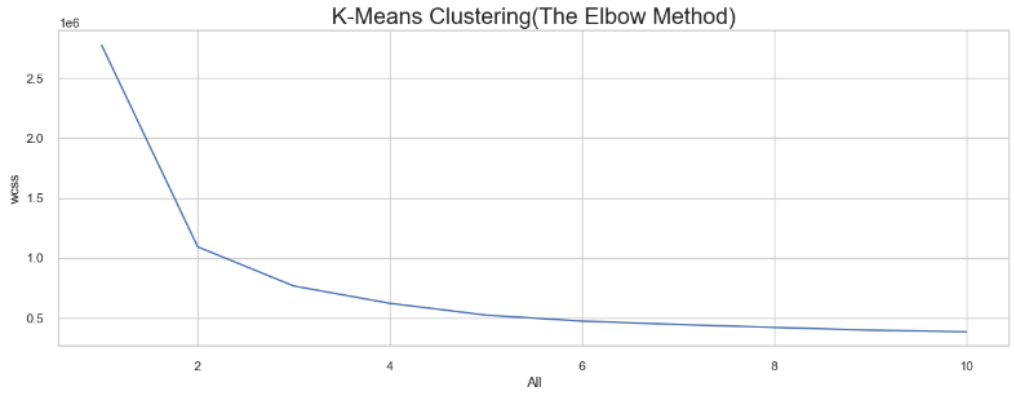

Figura 3.13: Applicazione dell'elbow method nella PCA

Si può, quindi, produrre lo scatterplot relativo alla clusterizzazione multidimensionale formato da tre cluster, che si è deciso che identificano le città con basso, medio e alto costo della vita (Figura 3.14).

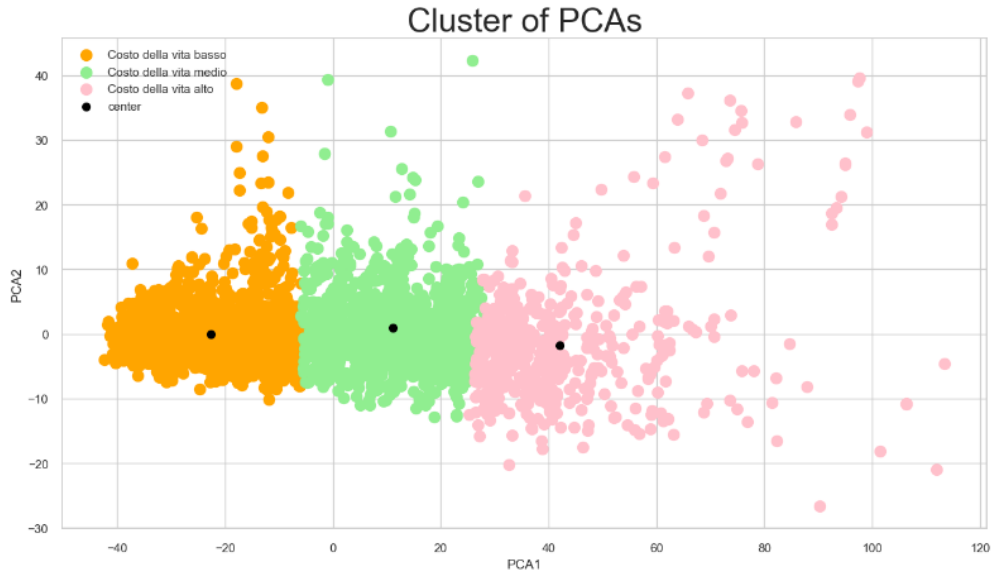

Figura 3.14: Clustering della PCA derivato da K-Means

Per evidenziare il fatto che lo score medio della silhouette del dataset normalizzato sia decisamente troppo basso, si riporta anche il grafico relativo alla silhouette senza normalizzazione (Figura 3.15). Lo score medio, in questo caso, risulta essere migliore, raggiungendo un valore di circa 0.42. Inoltre, anche il numero dei valori negativi diminuisce notevolmente rispetto alla versione normalizzata.

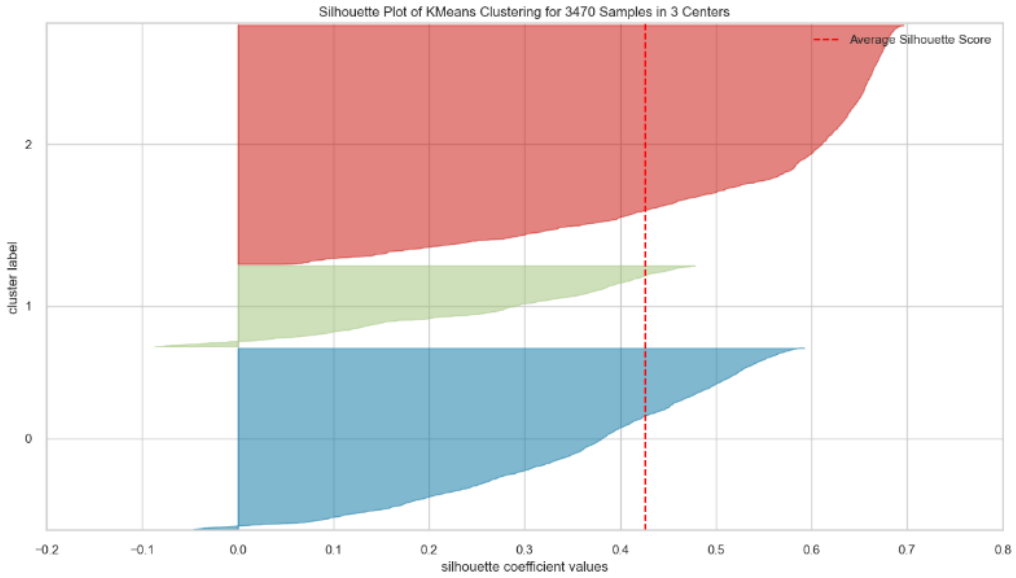

Figura 3.15: Silhouette della PCA

L'ultima attività di clustering eseguita è stata l'applicazione dell'algoritmo

DB-Scan. Tuttavia, questa operazione non ha portato a risultati soddisfacenti; per questo motivo, si è scelto di non riportarli.

3.2 Classificazione

Lo studio del dataset sul costo della vita è stato ulteriormente approfondito attraverso l'utilizzo di tecniche di classificazione. In questa sezione, infatti, verranno illustrati i risultati ottenuti tramite l'applicazione di modelli predittivi per definire la classe di appartenenza di un elemento, in base alle sue caratteristiche.
Le caratteristiche del dataset non si prestano benissimo a un'analisi di questo tipo, tuttavia si è optato per una classificazione binaria dipendente dal prezzo del pane. Questa decisione è giustificata dal fatto che il pane viene considerato un prodotto di prima necessità e, di conseguenza, un buon indicatore del costo della vita di una località. La delimitazione delle due categorie sarà, quindi, indicata dal prezzo medio del pane: se pari o minore a 1.30$, la città rientrerà nella prima classe (economica); se superiore a 1.30$, la città farà parte della seconda classe (onerosa). Questo valore è stato scelto in modo da avere due gruppi bilanciati, costituiti rispettivamente da 1730 e 1740 elementi.
Una volta definite le categorie, sono stati scelti gli algoritmi di classificazione da applicare per le previsioni:

- LogisticRegression
- DecisionTreeClassifier
- SupportVectorClassifier (SVC)
- RandomForestClassifier

Prima di poter mettere all'opera questi algoritmi, è stato necessario dividere il dataset in due parti, una utilizzata per l'addestramento e l'altra destinata ai test. Il dataset è stato quindi suddiviso in questo modo: 80% train-set e 20% test-set. Gli algoritmi eseguiti sotto queste condizioni hanno portato ai seguenti valori di accuratezza:

- LogisticRegression: **0.84**
- DecisionTreeClassifier: **0.85**
- SVC: **0.82**
- RandomForestClassifier: **0.89**

In Figura 3.16 è rappresentato un grafico relativo all'accuratezza degli algoritmi eseguiti con la corrispondente deviazione standard.

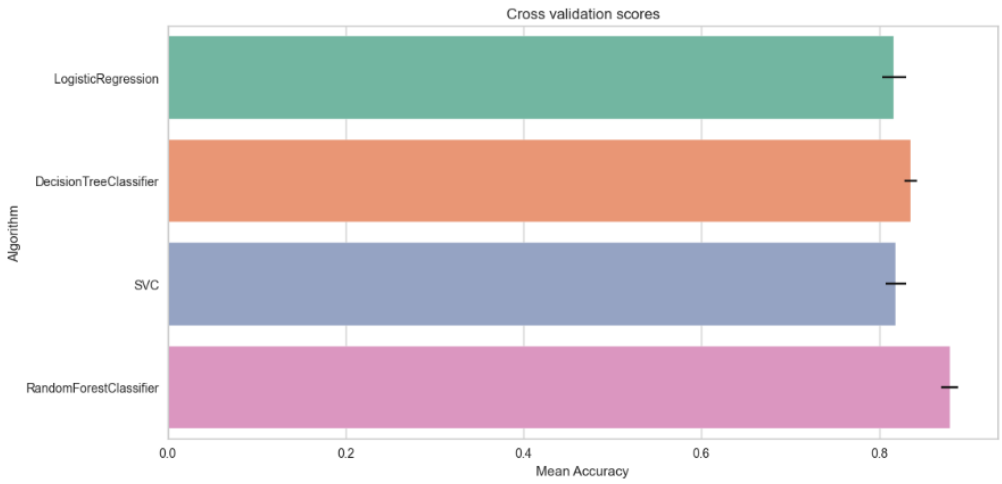

Figura 3.16: Accuratezza media degli algoritmi

Le matrici di confusione corrispondenti ad ogni classificatore sono visualizzabili in Figura 3.17.

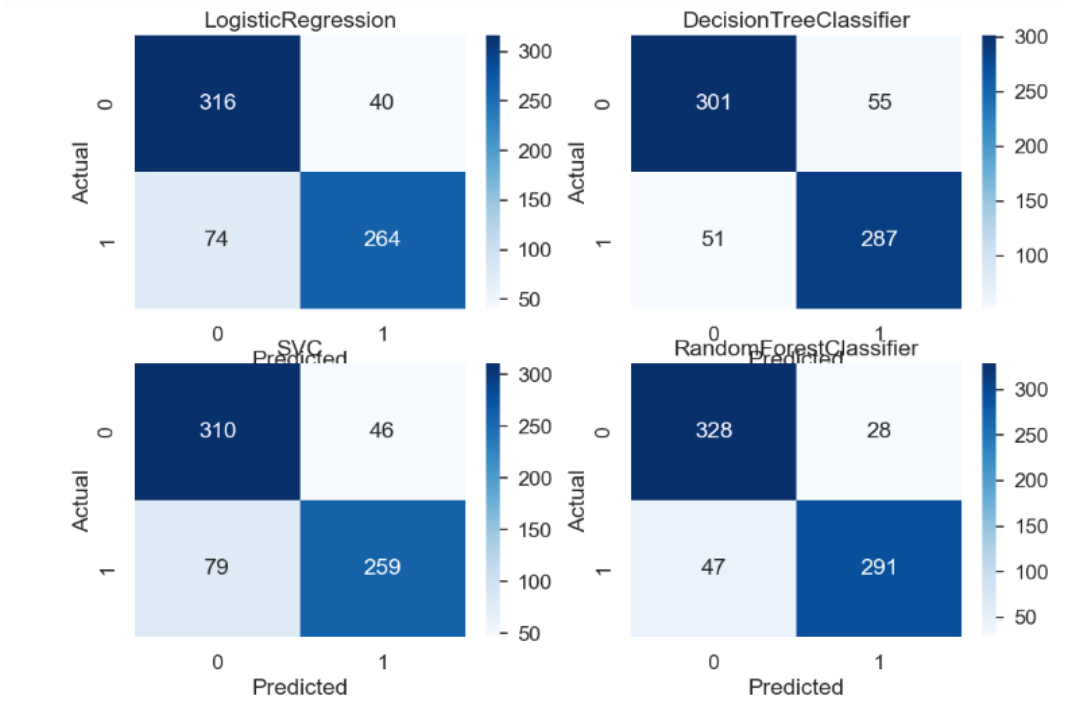

Figura 3.17: Matrici di confusione degli algoritmi

Dai risultati si può dedurre che le prestazioni migliori sono state ottenute dall'applicazione dell'algoritmo RandomForest. A questo proposito, il report sulle

prestazioni del RandomForestClassifier è visibile nella seguente tabella:

	Precision	**Recall**	**F1-Score**	**Support**
Economica	0.87	0.92	0.90	356
Onerosa	0.85	0.77	0.81	338
Accuracy	-	-	0.82	694
Macro AVG	0.89	0.89	0.89	694
Weighted AVG	0.89	0.89	0.89	694

Da questi parametri si può notare come il modello riesca a classificare con maggiore correttezza gli elementi appartenenti alla classe Economica (F1-Score pari a 0.90), pur mantenendo valori elevati anche per la classe Onerosa (F1-Score di 0.81).
Inoltre, dal momento che si sta facendo riferimento ad un classificatore binario, si può rappresentare il grafico della **curva ROC**. La curva ROC esprime la relazione tra la sensibilità, ossia la capacità del modello di identificare correttamente le istanze della classe positiva, e la specificità, ovvero la capacità di identificare le istanze della classe negativa, al variare della soglia di classificazione. La sensibilità è rappresentata dal True Positive Rate, mentre la specificità dal False Positive Rate. Dalla Figura 3.18 si può notare come anche il metodo della curva ROC confermi che il RandomForestClassifier risulta essere l'algoritmo più adatto, in quanto quello che più si avvicina all'angolo in alto a sinistra.

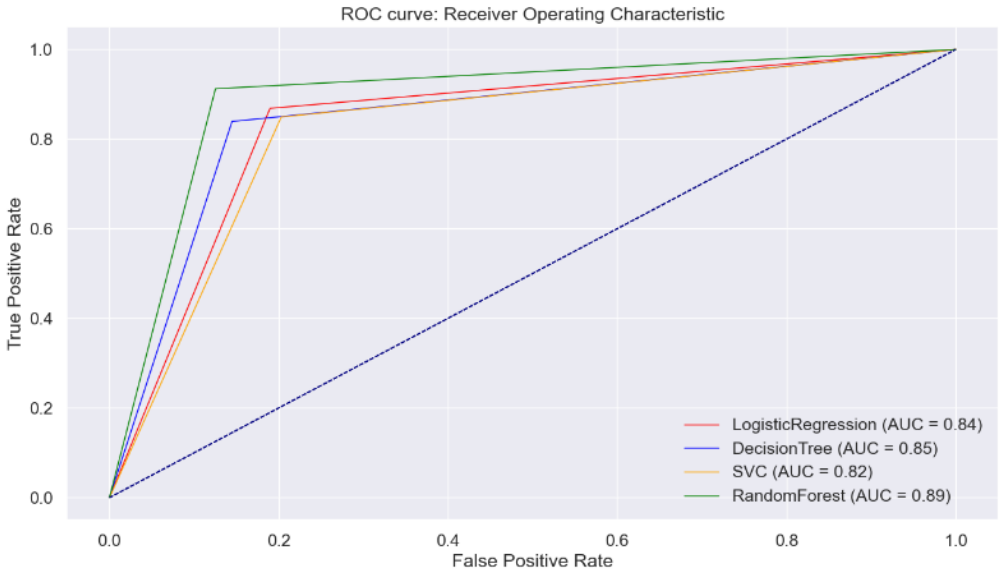

Figura 3.18: Curva ROC

Si può definire un livello di correlazione tra due classificatori in base alla similarità delle loro predizioni. In Figura 3.19 si riporta la heatmap relativa alla correlazione tra i classificatori applicati in questa sede.

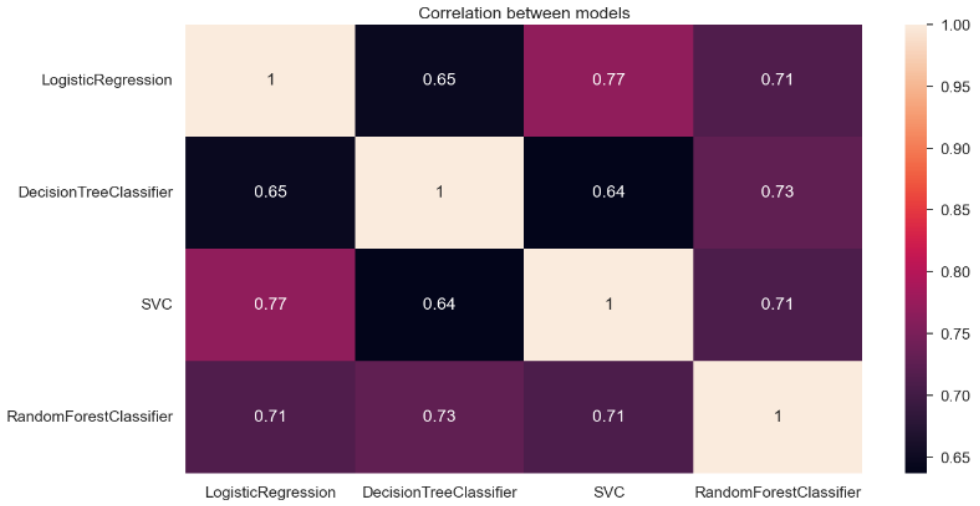

Figura 3.19: Correlazione fra i classificatori

Sulla base di questo grafico sono stati scelti i classificatori da utilizzare per l'analisi effettuata tramite *Grid Search*. Per questa particolare tecnica di classificazione sono stati scelti DecisionTreeClassifier e RandomForestClassifier, i quali hanno un valore di correlazione non troppo elevato. La tecnica di Grid Search si basa sul calcolo dei valori ottimali degli iperparametri; vengono inseriti dei valori per i parametri che si vogliono ottimizzare e si provano tutte le possibili combinazioni. Gli iperparametri selezionati sono mostrati in Figura 3.20.

```
DT_param = {"max_depth": [2,3,8,10],
            "max_features": [0.3, 0.7, 1],
            "min_samples_split": [2, 3, 10],
            "min_samples_leaf": [1, 3, 10],
            "criterion": ["gini"]}

RF_param = {"max_depth": [None],
            "max_features": [0.3, 0.7, 1],
            "min_samples_split": [2, 3, 10],
            "min_samples_leaf": [1, 3, 10],
            "bootstrap": [False],
            "n_estimators" :[100,300],
            "criterion": ["gini"]}
```

Figura 3.20: Iperparametri scelti per la Grid Search

La suddivisione del dataset è stata effettuata tramite la *k-cross fold validation*, operazione che divide il dataset in k parti di uguale dimensioni e sceglie uno di

questi gruppi come test set, mentre tutti gli altri formano il training set. Questa operazione viene iterata k volte scegliendo come test set un gruppo diverso ad ogni passo.
Terminato l'addestramento, sono stati eseguiti i modelli e sono stati valutati i risultati ottenuti sia applicando che non applicando la Grid Search:

Accuratezza	**DecisionTreeClassifier**	**RandomForestClassifier**
Senza GridSearch	0.836	0.879
Con GridSearch	0.856	0.903

L'accuratezza di entrambi i modelli risulta essere maggiore dopo aver effettuato un addestramento applicando la Grid Search, raggiungendo anche valori superiori al 90% nel caso del RandomForestClassifier.
Infine, è stato applicato il metodo dell'*Ensemble* che consiste nell'utilizzare più classificatori contemporaneamente al fine di ottenere migliori prestazioni. L'Ensemble è stato eseguito unendo i due classificatori usati anche nella precedente analisi, DecisionTree e RandomForest. L'accuratezza derivata dalla combinazione dei due classificatori risulta essere pari a 0.886, valore leggermente inferiore rispetto a quella del RandomForestClassifier con Grid Search; tuttavia la valutazione dovrebbe risultare maggiormente affidabile in quanto derivante da più classificatori.

Capitolo 4

Time Series Analysis

Nel passaggio all'analisi delle serie temporali è stato necessario cambiare anche il set di dati da analizzare. In questa sezione, infatti, il dataset preso in considerazione è quello relativo all'andamento delle azioni di *Netflix* introdotto nel Capitolo 2. L'analisi è stata condotta per lo più applicando metodi relativi alla libreria *Statsmodels* di *Python* che fornisce classi e funzioni per stimare modelli matematici e condurre esplorazioni di dati e test statistici.

4.1 Preparazione del modello ARIMA

Per iniziare, è stata realizzata una rappresentazione grafica dell'andamento del volume delle azioni di Netflix negli ultimi 10 anni (Figura 4.1). Il grafico è stato generato raggruppando i dati per mese, come già accennato nel paragrafo relativo all'ETL.

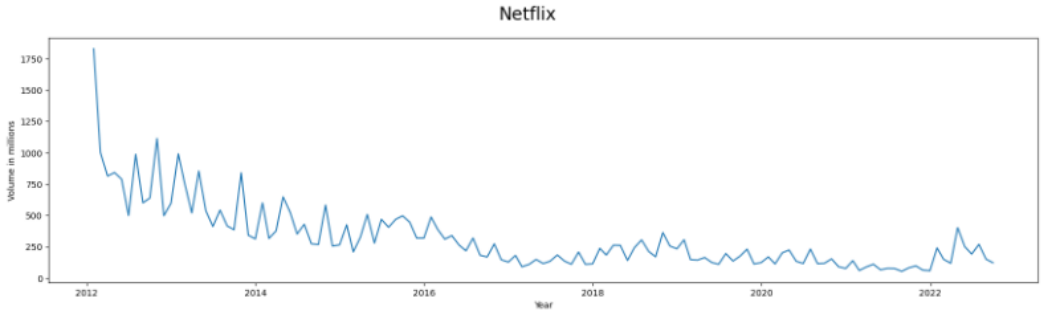

Figura 4.1: Andamento del volume delle azioni di Netflix

Già ad un primo impatto, la serie non risulta essere caratterizzata da stagionalità; di conseguenza, può essere definita stazionaria.

4.1.1 Test sulla stazionarietà della serie

Si è deciso comunque di effettuare un test sulla stazionarietà della serie. È stato quindi eseguito l'*Augmented Dickey Fuller Test* (ADF test), che, attraverso la determinazione del p-value, riesce a stabilire se la serie temporale d'interesse è stazionaria o meno. L'ipotesi nulla dell'ADF test è che la serie sia non stazionaria, se il p-value risulta minore di 0.05 si rifiuta l'ipotesi e si valuta la serie come stazionaria. Dai risultati del test si ottiene un **p-value = 0.031052**, a conferma del fatto che la serie in analisi sia stazionaria.

4.1.2 Autocorrelazione e Autocorrelazione Parziale

A questo punto, si generano i grafici di autocorrelazione e autocorrelazione parziale (Figura 4.2). Questi grafici sono necessari per stimare i parametri del modello *ARIMA* che, attraverso l'indagine di serie storiche, tenterà di effettuare una previsione della serie.

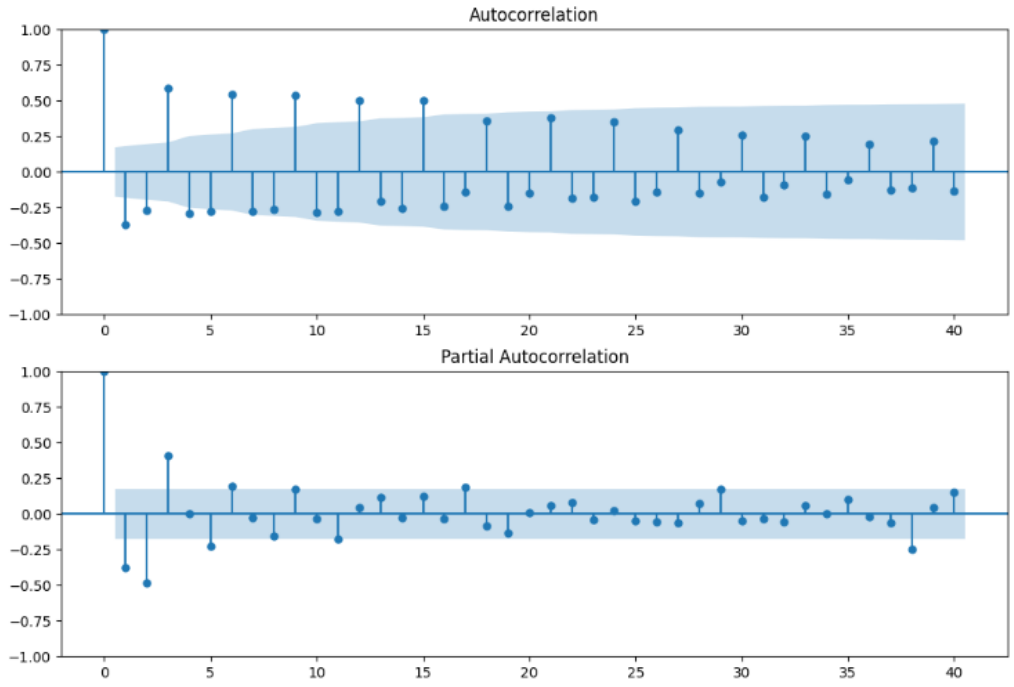

Figura 4.2: Autocorrelazione e Autocorrelazione Parziale

4.1.3 Stima dei parametri del modello ARIMA

I parametri che caratterizzano un modello ARIMA sono i seguenti:

- **p**: numero di parametri autoregressivi (AR)

- **d**: ordine di differenziazione (I)
- **q**: numero di parametri a media mobile (MA)

È necessario effettuare una stima dei tre parametri per poter mettere in atto il modello.
Il parametro **d** indica il numero di differenziazioni necessarie affinché la serie sia stazionaria; dal momento che la serie è risultata essere stazionaria sin da subito, non si è resa necessaria nessuna differenziazione; di conseguenza, si può imporre il parametro **d** pari a 0.
I parametri **p** e **q** possono essere stimati analizzando i grafici di autocorrelazione e autocorrelazione parziale (Figura 4.2). Osservando il grafico di autocorrelazione parziale, si riesce a stimare il parametro **p** in base al numero di lag che si trovano al di sopra dell'indice di significatività. In questo caso, il valore del parametro **p** ideale è stato individuato essere pari a 3. Per realizzare una stima del parametro **q**, invece, è necessario osservare il grafico di autocorrelazione; da esso, però, non è stato possibile individuare un valore accettabile di **q**.
Per eseguire la stima dei valori ideali dei parametri, un'alternativa è messa a disposizione dalla libreria di Python *pmdarima*. Questa libreria mette a disposizione il metodo *auto_arima*, che indica la migliore combinazione dei parametri (p,d,q)(P,D,Q)[m], fissandone alcuni. Ai soliti tre parametri, ne vengono aggiunti altri per la componente stagionale: **P**, **Q** e **D**, che indicano rispettivamente il numero di parametri autoregressivi stagionali, il numero di parametri a media mobile stagionali e l'ordine di differenziazione stagionale, e **m**, che rappresenta la stagionalità. I valori fissati sono stati $\mathbf{p} = 3$, $\mathbf{d} = 0$ e $\mathbf{m} = 12$ e il set di parametri considerato migliore da auto_arima è il seguente:

$$\text{ARIMA}(3,0,0)(2,0,0)[12]$$

4.2 Risultati delle previsioni

Individuati i valori più adatti per tutti i parametri, è stato utilizzato il modello per effettuare previsioni. Il modello a cui si fa riferimento è in realtà una variante dell'ARIMA che, con l'aggiunta della stagionalità e di variabili esogene, è diventato un modello SARIMAX (Seasonal ARIMA with eXogenous factors).
La serie è stata suddivisa in training e testing, rispettivamente 80% e 20%, e attraverso l'applicazione del metodo *forecast()* è stata realizzata una prima previsione (Figura 4.3). Portando l'attenzione soltanto sugli anni interessati dalla predizione (Figura 4.4), si palesa il fatto che, a partire da Gennaio 2022, il forecast risulta discostarsi dai dati reali del volume in quanto, quest'ultimi, hanno registrato un picco in quel periodo temporale.

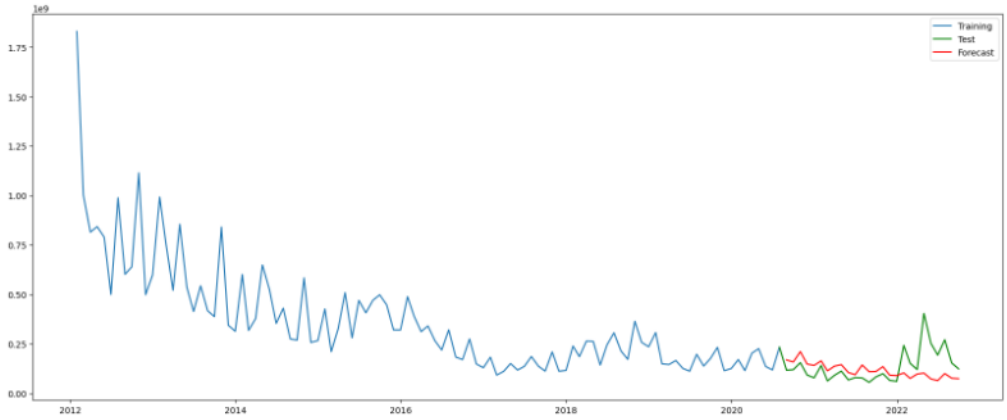

Figura 4.3: Predizione derivata dall'applicazione di forecast()

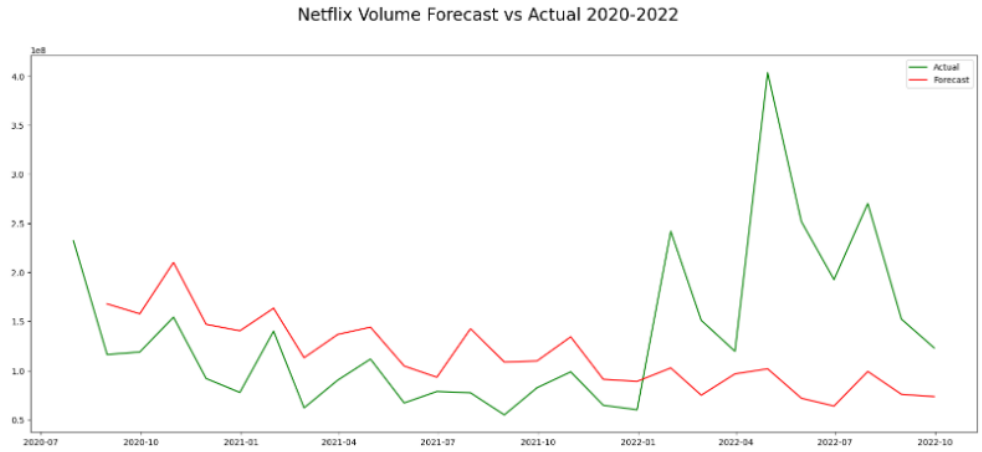

Figura 4.4: Differenza tra predizione e dati reali

Per individuare i motivi che hanno causato questo improvviso aumento, sono stati investigati i dati relativi ai prezzi delle azioni. Come si può osservare dal grafico relativo ai prezzi di chiusura (Figura 4.5), il periodo in cui è stato registrato un aumento del volume corrisponde ad un calo significativo dei prezzi delle azioni di Netflix. Questo inaspettato calo registrato nel mese di Gennaio 2022 ha portato ad un aumento delle vendite e, di conseguenza, del volume complessivo.

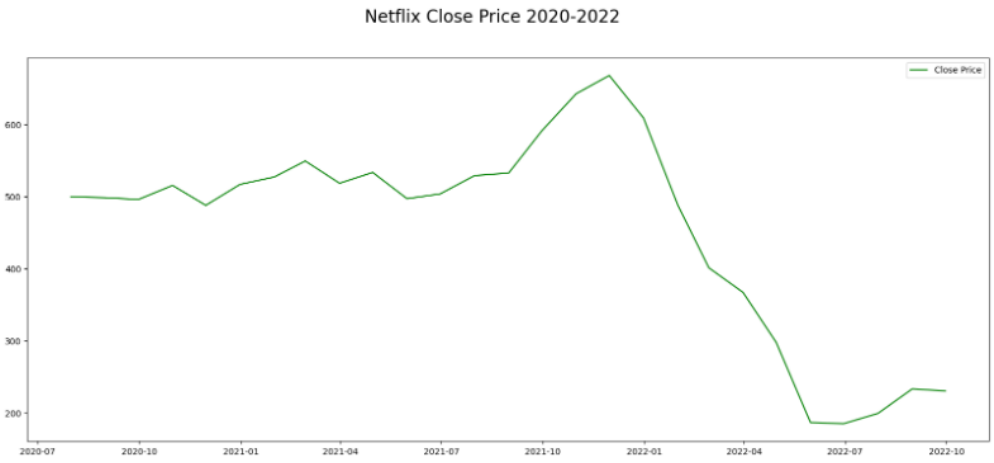

Figura 4.5: Andamento del prezzo alla chiusura dei mercati tra il 2020 e il 2022

Tuttavia, l'andamento della predizione antecedente il mese di Gennaio 2022 risultava essere in linea con quello reale. Da ciò si può dedurre che il modello si è dimostrato capace di generare previsioni realistiche, allontanandosi dai valori effettivi solo in presenza di fenomeni imprevedibili.
Statsmodels mette a disposizione il metodo *plot_ diagnostics* in grado di stampare grafici (Figura 4.6) relativi alla qualità del modello SARIMAX realizzato.

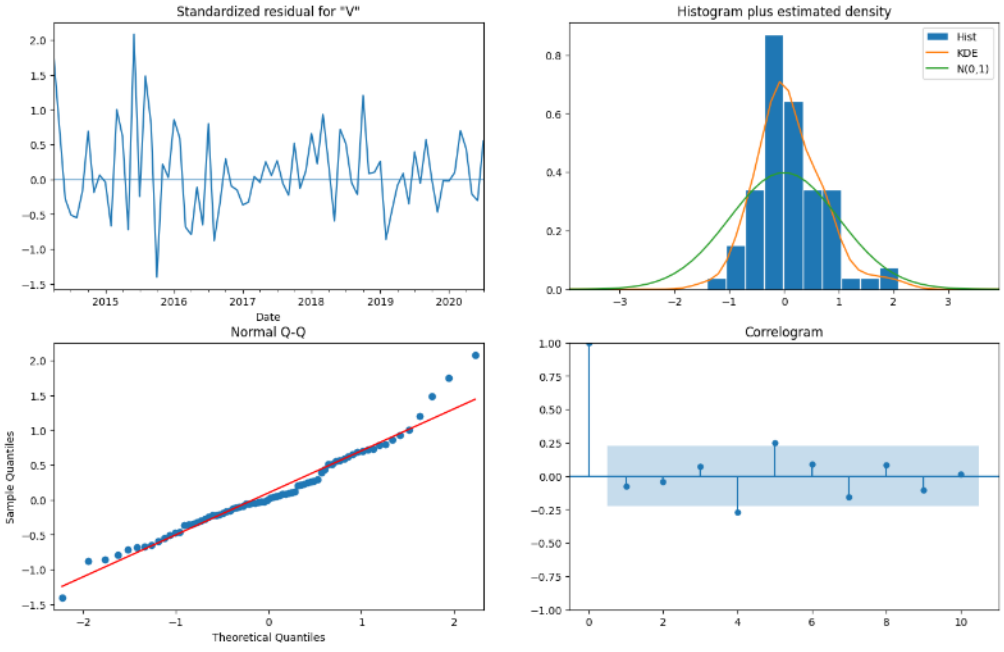

Figura 4.6: Statistiche relative al modello SARIMAX

Dai residual plots si può osservare come i residui oscillino attorno ad una media nulla, indicatore di una buona capacità predittiva del modello. Le principali metriche di valutazione, invece, sono state riportate di seguito:

- **MAPE**: 0.5172804775071107,
- **MAE**: 71282210.10901056,
- **MPE**: 0.1369858980980398,
- **RMSE**: 95315886.5374028,
- **NRMSE**: 0.27344229575616225

L'ultima tipologia di analisi realizzata nello studio della serie temporale è stata la previsione dei valori fuori dal campione. Per questa analisi sono stati utilizzati due diversi metodi: *predict()*, che si basa su un modello addestrato sui dati storici, e *get_forecast()*, che tiene conto di trend e stagionalità. Il primo è un metodo che genera una previsione cosiddetta "quick and dirty", ovvero eseguita rapidamente, senza soffermarsi sui dettagli e sulla qualità. Con questo metodo si è voluto generare velocemente una previsione dei successivi 16 mesi, anche se non perfetta. Ciò che si è ottenuto dalla sua applicazione è il grafico in Figura 4.7.

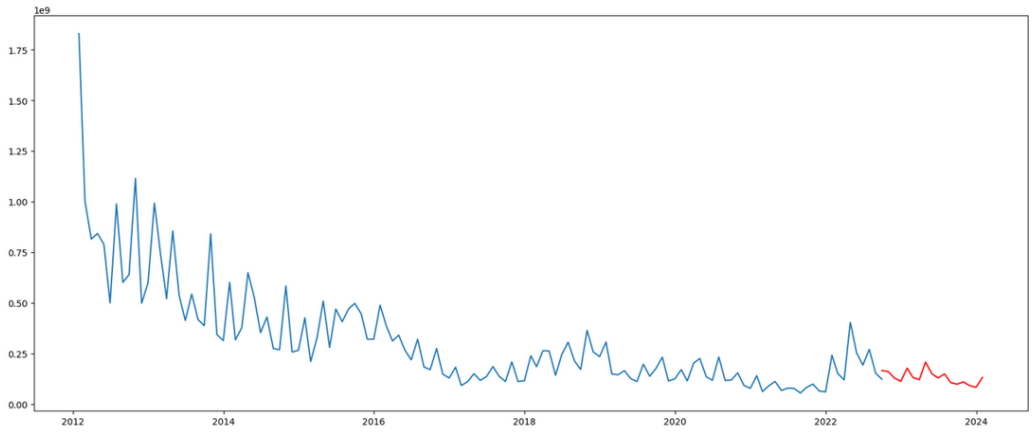

Figura 4.7: Predizione dei valori fuori campione con il metodo predict()

Infine, con l'applicazione del metodo *get_forecast*, è stata generata la previsione riportata in Figura 4.8. Essa è in grado di catturare tendenze e stagionalità con maggiore precisione rispetto al metodo *predict()*.

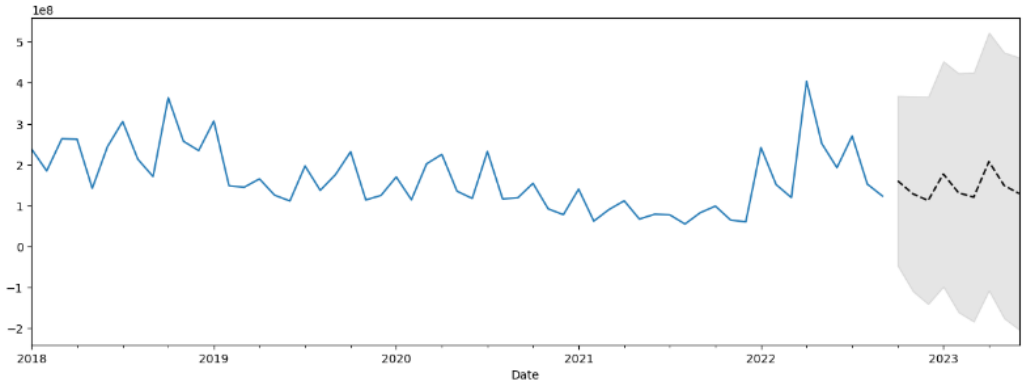

Figura 4.8: Predizione dei valori fuori campione con il metodo get_forecast()

È difficile determinare quale delle due previsioni sia più accurata, ma il loro utilizzo combinato garantisce una maggiore completezza riguardo la previsione di valori fuori campione.

Capitolo 5

Conclusioni

Classificazione e clustering sono stati eseguiti analizzando il dataset relativo al costo della vita globale.
Riguardo le attività di clustering, sono stati sfruttati i metodi messi a disposizione dalla libreria di *Python* chiamata *Scikitlearn*. Il primo passo è stato definire un clustering bidimensionale che dipendesse dai prezzi di pane e pollo. Si è proseguito trovando il numero ideale di cluster attraverso l'applicazione dell'*elbow method* e rappresentandoli attraverso una serie di grafici quali scatterplot e silhouette. Si è cercato di migliorare i risultati ottenuti prima cambiando il numero di cluster e poi normalizzando i parametri relativi a pane e pollo. La prima variante ha portato a risultati peggiori, mentre la seconda ha visto un netto miglioramento dei risultati. Per concludere la parte sui cluster, è stata effettuata la PCA per non limitare l'analisi al solo clustering bidimensionale.
La classificazione è stata eseguita dividendo le città in base al costo del pane. Sono stati utilizzati quattro diversi algoritmi di classificazione e si è potuto stabilire che il *RandomForestClassifier* è risultato essere quello più accurato in questo specifico caso di studio. È stato anche implementato un modello addestrato tramite *Grid Search* utilizzando i classificatori *DecisionTree* e *RandomForest*.
L'analisi della serie temporale ha avuto come protagonista il dataset relativo all'andamento delle azioni di Netflix. Le operazioni eseguite per preparare il modello sono state, in ordine, il test sulla stazionarietà, la stima dei parametri usando i grafici di autocorrelazione e autocorrelazione parziale e la stima dei parametri attraverso l'applicazione di *auto_arima*. A questo punto, è stato possibile eseguire dei test per verificare la capacità del modello di prevedere correttamente i valori della serie associata al testing set. Inoltre, sono stati analizzati i grafici dei residui e le metriche di valutazione al fine di stabilire la qualità del modello SARIMAX realizzato. In conclusione, sono state realizzate due previsioni dei valori fuori campione utilizzando i metodi *predict()* e *get_forecast()*.

Elenco delle figure