Università Politecnica delle Marche

Dipartimento di Ingegneria dell'Informazione

Facoltà di Ingegneria Informatica e dell'Automazione

Analisi dei clienti ideale di una compagnia e del consumo elettrico domestico mediante l'ecosistema Python

Docenti:
Domenico Ursino
Michele Marchetti

Studenti:
Silvia Ciuffreda
Luca Liberatore
Beatrice Moliterno

ANNO ACCADEMICO 2022/2023

Indice

Elenco delle figure

1 Introduzione

L'elaborato riporta i risultati dell'analisi dei dati di due dataset: il primo concerne informazioni sulla clientela ideale di una generica compagnia comprendente tre anni, dal 2012 al 2014; il secondo include i dati del consumo elettrico domestico a San Jose nel periodo 2020-2022.

La ragione che ha portato alla scelta di due dataset è dovuta al fatto che si effettuano analisi di tipo differente, di fatto quello comprendente i dati dei clienti di un'azienda viene utilizzato per effettuare Clustering e Classificazione, mentre il secondo, dato la presenza di dati giornalieri del consumo elettrico domestico, viene utilizzato per l'analisi delle serie temporali, al fine di estrarre previsioni future.

Seguendo le orme della Data Science attuale si utilizzerà il linguaggio di programmazione Python per lo svolgimento di analisi sopracitate; in particolare, si farà uso di librerie come *Pandas* per la fase di ETL, *Seaborn* e *Matplotlib* per la visualizzazione dei dati ed, infine, *Scikit-learn* per il Machine Learning.

2 Customer Personality Analysis

In questa sezione viene presentato il dataset riguardante la fidelizzazione di clienti ideali di una generica compagnia food&drink, disponibile al seguente link: `https://www.kaggle.com/datasets/imakash3011/customer-personality-analysis`, sui cui si effettueranno dapprima operazioni di ETL e visualizzazione dei dati, per poi concentrarsi ampiamente sull'applicazione di tecniche di clustering e classificazione. In tal senso, si avrà modo di comprendere come si segmenta la domanda sui vari prodotti disponibili, consentendo alla compagnia di svolgere azioni mirate a favorire l'acquisto di un determinato prodotto da parte di una specifica porzione di clienti.

Di seguito si riporta una breve descrizione degli attributi presenti (1).

Customer Personality Analysis	
ID	Customer's unique identifier
Year Birth	Customer's birth year
Education	Customer's education level
Marital Status	Customer's marital status
Income	Customer's yearly household income
Kidhome	Number of children in customer's household
Teenhome	Number of teenagers in customer's household
Dt Customer	Date of customer's enrollment with the company
Recency	Number of days since customer's last purchase
Complain	1 if the customer complained in the last 2 years, 0 otherwise
MntWines	Amount spent on wine in last 2 years
MntFruits	Amount spent on fruits in last 2 years
MntMeatProducts	Amount spent on meat in last 2 years
MntFishProducts	Amount spent on fish in last 2 years
MntSweetProducts	Amount spent on sweets in last 2 years
MntGoldProds	Amount spent on gold in last 2 years
NumDealsPurchases	Number of purchases made with a discount
AcceptedCmp1	1 if customer accepted the offer in the 1st campaign, 0 otherwise
AcceptedCmp2	1 if customer accepted the offer in the 2nd campaign, 0 otherwise
AcceptedCmp3	1 if customer accepted the offer in the 3rd campaign, 0 otherwise
AcceptedCmp4	1 if customer accepted the offer in the 4th campaign, 0 otherwise
AcceptedCmp5	1 if customer accepted the offer in the 5th campaign, 0 otherwise
Response	1 if customer accepted the offer in the last campaign, 0 otherwise
NumWebPurchases	Number of purchases made through the company's website
NumCatalogPurchases	Number of purchases made using a catalogue
NumStorePurchases	Number of purchases made directly in stores
NumWebVisitsMonth	Number of visits to company's website in the last month

Figura 1: Customer Personality Analysis

2.1 ETL

Una volta scaricato il dataset e caricato su *Jupyter Notebook* sono state effettuate operazioni di pulizia e di visualizzazione dei dati mediante le librerie *pandas, matplotlib e seaborn.*

Innanzitutto, il file *.csv* appena caricato, viene prontamente visualizzato per confermare se tutto fosse andato a buon fine, mostrandone le prime cinque righe (2).

```
#importo il dataset
data=pd.read_csv('marketing_campaign.csv',header=0,sep='\t')
data.head()
```

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebVisitsMonth	AcceptedCmp3	Acce
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2012	58	635	...	7	0	
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2014	38	11	...	5	0	
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-2013	26	426	...	4	0	
3	6182	1984	Graduation	Together	26646.0	1	0	10-02-2014	26	11	...	6	0	
4	5324	1981	PhD	Married	58293.0	1	0	19-01-2014	94	173	...	5	0	

Figura 2: Dataset iniziale

Si può osservare da subito come, essendo le colonne un numero considerevole, non possono essere visualizzate tutte all'interno della stessa schermata. Si è deciso quindi, anche con la motivazione di snellire l'analisi, di eliminare alcune colonne ritenute poco significative per i passi successivi; sono state rimosse, dunque, tutte le colonne riguardanti le campagne pubblicitarie.

Successivamente, si è verificata la lunghezza del dataframe dimostrando l'univocità dell'attributo "ID" ed inoltre si è voluto constatare la presenza o meno di eventuali valori nulli.

Come riportato in Figura 3, il dataset riporta un totale di 24 valori nulli esclusivamente per l'attributo "Income". Essendo un numero esiguo, rispetto alla totalità delle righe, esse sono state semplicemente eliminate dal dataframe.

```
#verifico la lumghezza attuale del dataframe
len(data)
```

```
2240
```

```
#assegno l'identificatore verificando prima che sia univoco
data['ID'].is_unique
```

```
True
```

```
data = data.set_index('ID')
```

```
#controllo i valori nulli
data.isnull().sum()
```

```
Year_Birth             0
Education              0
Marital_Status         0
Income                24
Kidhome                0
Teenhome               0
Dt_Customer            0
Recency                0
MntWines               0
MntFruits              0
MntMeatProducts        0
MntFishProducts        0
MntSweetProducts       0
MntGoldProds           0
NumDealsPurchases      0
NumWebPurchases        0
NumCatalogPurchases    0
NumStorePurchases      0
NumWebVisitsMonth      0
Complain               0
dtype: int64
```

Figura 3: Lunghezza df e Valori Nulli

Un'operazione di ETL riguarda la conversione di formato del campo concerne la data di iscrizione del cliente presso la compagnia, "DT Customer" in un attributo di tipo *datetime*. A tal proposito, il dataset non fornisce di default l'età di ciascun cliente fidelizzato, ma soltanto il proprio anno di nascita.

Considerando l'età un attributo fondamentale ai fini del caso di studio, si è deciso di ricavarla dai campi presenti nel dataset. Supponendo come anno di riferimento il 2014, il campo "Età" si è ottenuto come differenza tra il 2014 e la propria data di nascita. In questo modo, dunque, si ha la disponibilità delle fasce d'età di ciascun compratore, ritenuta fondamentale per le analisi che seguiranno.

Il passo successivo considera tre features dei dati su cui fare trasformazione, al fine di rendere il dataset più comprensibile nonchè facilmente utilizzabile nella successiva fase di analisi. I campi di cui si sta parlando sono i seguenti:

- *Education*: il grado di istruzione dei clienti
- *Marital Status*: lo stato sociale dei clienti
- *Teenhome* e *kidhome*: il numero di figli dei clienti

Si è partiti dal campo "Education" in cui, effettuando un semplice *value_counts()* è emersa la distinzione di 5 tipologie di istruzione, come mostrato nella Figura 4.

```
#controllo il titolo di studio
data["Education"].value_counts()
```

```
Graduation    1116
PhD            480
Master         365
2n Cycle       198
Basic           54
Name: Education, dtype: int64
```

Figura 4: Conteggi dei valori univoci del campo "Education"

Al fine di semplificare le cose si è deciso di creare una nuova colonna, "Education2", che suddividesse l'istruzione in due semplici insiemi distinti: *Base* per il tipo di istruzione "Basic" ed *Avanzata* per tutte le restanti categorie.

Una rappresentazione tabellare di quanto detto è presente nella Figura 5.

ID	Education	Education2
5524	Graduation	Avanzata
2174	Graduation	Avanzata
4141	Graduation	Avanzata
6182	Graduation	Avanzata
5324	PhD	Avanzata
7446	Master	Avanzata
965	Graduation	Avanzata
6177	PhD	Avanzata
4855	PhD	Avanzata
5899	PhD	Avanzata
387	Basic	Base

Figura 5: Education2

Per quanto riguarda il numero dei figli di ciascun cliente, il dataset mette a disposizione due campi a riguardo, "Teenhome" e "Kidhome", che esprimono rispettivamente il numero di adolescenti e il numero di bambini in casa. Da tali colonne sono stati creati due nuovi campi: "No_Child" rappresenta il numero totale di figli in casa, ignorando la distinzione tra bambini e adolescenti; "Child" il cui valore è un booleano e indica se il cliente abbia o meno figli in casa.

L'ultima trasformazione effettuata riguarda lo stato sociale dei clienti. Il dataset originario permette di distinguere diversi valori per il campo "Marital Status", come mostrato nella Figura 6.

```
#controllo lo stato civile
data["Marital_Status"].value_counts()
```

```
Married     857
Together    572
Single      470
Divorced    231
Widow        76
Alone         3
Absurd        2
YOLO          2
Name: Marital_Status, dtype: int64
```

Figura 6: Conteggi dei valori univoci del campo "Marital Status"

Anche in questo caso, per rendere più leggibile il dataset e semplificare l'analisi, si è deciso di raggruppare i dati in due insiemi: "Single" ed "In couple", rinominando semplicemente la colonna di interesse. In Figura 7 vengono elencati i passaggi effettuati ed, inoltre, viene visualizzato il dataframe a seguito del processo di trasformazione.

```
#creo una nuova colonna "Child" con valore booleano
data['Child'] = np.where((data['Kidhome']+data['Teenhome'])> 0, True, False)
#creo una colonna che indica il numero di figli
data["No_Child"] = data["Kidhome"]+data["Teenhome"]

#suddivido lo stato civile in: single/coppia
data['Marital_Status']=data['Marital_Status'].replace({'Divorced':'Single','Single':'Single','Married':'In couple',
                                                       'Together':'In couple','Absurd':'Single','Widow':'Single',
                                                       'YOLO':'Single', 'Alone':'Single'})

#visualizzo se tutto ok
data[["Marital_Status", "Child", "Teenhome", "Kidhome", "No_Child"]].head(10)
```

ID	Marital_Status	Child	Teenhome	Kidhome	No_Child
5524	Single	False	0	0	0
2174	Single	True	1	1	2
4141	In couple	False	0	0	0
6182	In couple	True	0	1	1
5324	In couple	True	0	1	1
7446	In couple	True	1	0	1
965	Single	True	1	0	1
6177	In couple	True	0	1	1
4855	In couple	True	0	1	1
5899	In couple	True	1	1	2

Figura 7: Marital Status and Child

Infine, è stato creato un ulteriore campo dal nome "Tot_Mnt", il quale rappresenta il totale speso dai clienti nella compagnia, ovvero la somma dei singoli prodotti acquistati.

2.2 Data Visualization

Terminata la fase di preprocessing e pulizia dei dati, in questa sezione si è svolta un'analisi qualitativa e descrittiva del dataset, al fine di comprendere meglio il suo contenuto. Il primo grafico che si è ricavato, riportato in Figura 8 riguarda la rappresentazione, tramite *histplot*, del conteggio dei clienti particolarizzato per età di ciascuno di essi.

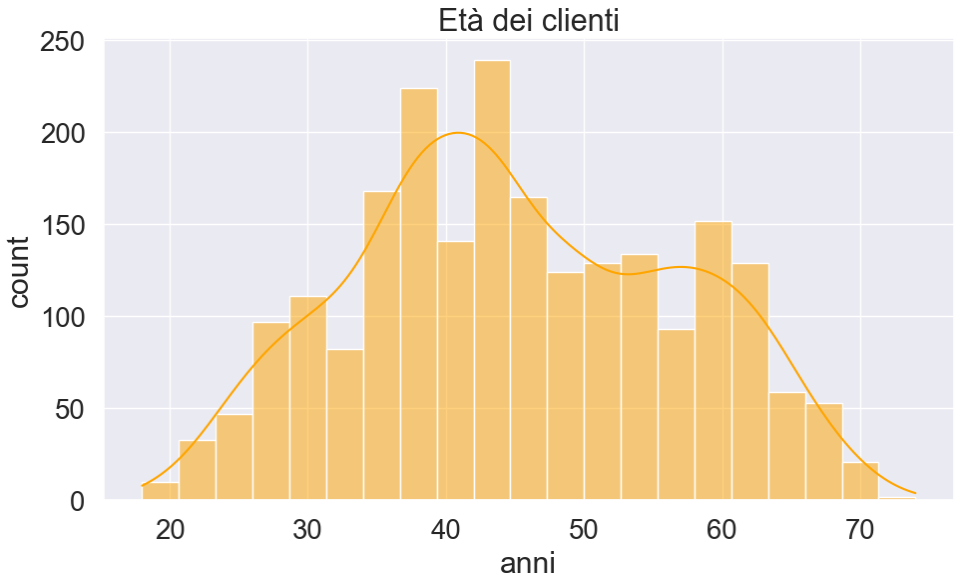

Figura 8: Età dei clienti

Da tale grafico si può osservare come i maggiori clienti si concentrano nella fascia d'età intermedia compresa tra i 35 e i 45 anni; al contrario, i più giovani e gli anziani sembrano essere meno interessati alla fidelizzazione e, dunque, alle promozioni che può offrire l'azienda.

In secondo luogo, è stato realizzato un grafico a torta che mostra la ripartizione tra i clienti che hanno figli e i clienti che non hanno figli, sulla base del campo "Child". Osservando la Figura 9, si nota chiaramente che la maggior parte dei clienti fidelizzati ha almeno un figlio, mentre solo il 28.56% sono quelli che non ne hanno.

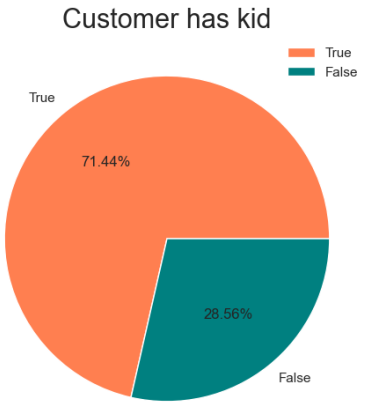

Figura 9: Child

Probabilmente più interessante è il *countplot* riportato nella Figura 10, il quale mostra i conteggi del numero categorico di figli dei clienti della compagnia, ciascuno raffigurato mediante l'utilizzo di barre.

Il grafico mostra come la maggior parte dei clienti abbia un solo figlio, seguito dai clienti che non possiedono figli e dai clienti che possiedono due figli. Infine, i clienti con ben tre figli sono presenti in minore quantità.

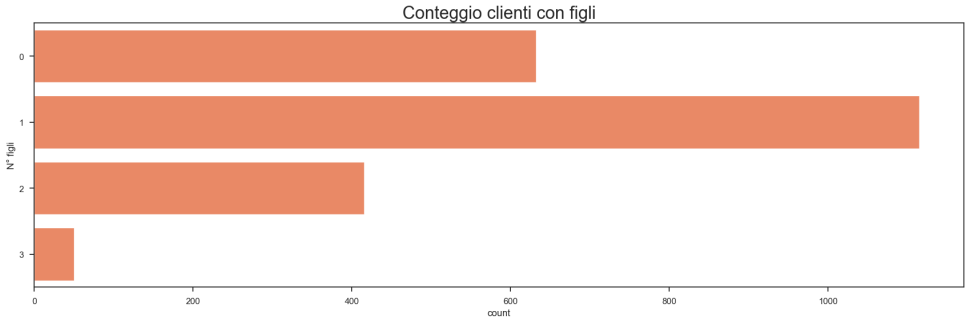

Figura 10: N° of Childs

Successivamente, è stato realizzato un grafico ad anello inerente la suddivisione dei clienti per stato sociale (11): si osserva come quasi il 65% dei clienti ha una relazione di coppia mentre il 35% dei clienti risulta single.

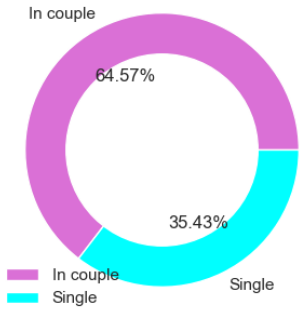

Figura 11: Marital Status

Un'ulteriore visualizzazione di rilevata importanza riguarda il guadagno dei clienti. Al fine di capire come si collocano i guadagni dei singoli clienti, è stato realizzato un *histplot*, il quale rappresenta la distribuzione dei guadagni contando il numero di osservazioni che rientrano in bin discreti (Figura 12). Si può dedurre, dunque, come la maggior parte dei guadagni tende a concentrarsi intorno al valore medio, lasciando intendere che la maggioranza della clientela abbia un guadagno annuo tutto sommato normale. Al contrario, si rilevano pochi individui che possiedono un guadagno annuo inferiore a 20.000$, così come quelli con un guadagno annuale di oltre 80.000$.

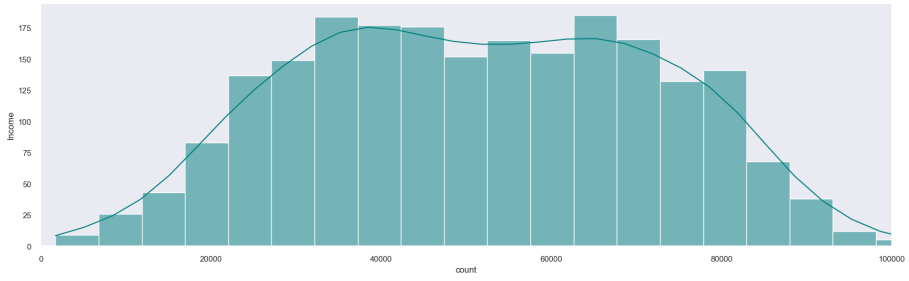

Figura 12: Income

Giunti a questo punto, nell'ultima parte della visualizzazione dei dati si sono voluti mostrare dei grafici più specifici. Ad esempio, il grafico in Figura 13 mette in relazione i valori della spesa totale per cliente

con quelli dei campi "Child" e "No_Child" (rispettivamente presenza di figli o meno a casa e numero totale di figli a casa) per evidenziare eventuali informazioni rilevanti per il caso di studio.

Come emerge chiaramente da tale grafico, la clientela che non possiede figli tende a spendere molto di più rispetto a chi non ha figli a carico; tale risultato sembra ragionevole, in quanto con la presenza di figli si è portati ad avere un occhio di riguardo per quanto riguarda la spesa complessiva.

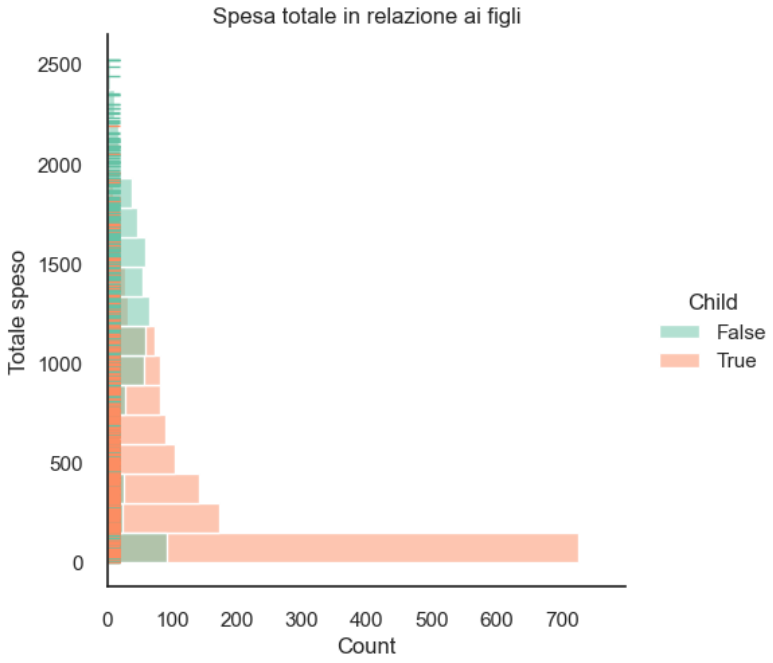

Figura 13: Conteggio del totale speso in relazione ad avere figli o meno

Quanto appena indotto appare ancora più evidente all'interno di uno *scatterplot*, riportato in Figura 14, in cui lungo l'asse delle ascisse vengono riportati i guadagni dei clienti e lungo quello delle ordinate il totale speso, il tutto particolarizzato per il numero di figli. Inoltre, all'aumento del numero di figli a carico, la spesa da parte dei clienti fidelizzati tende ad essere sempre minore.

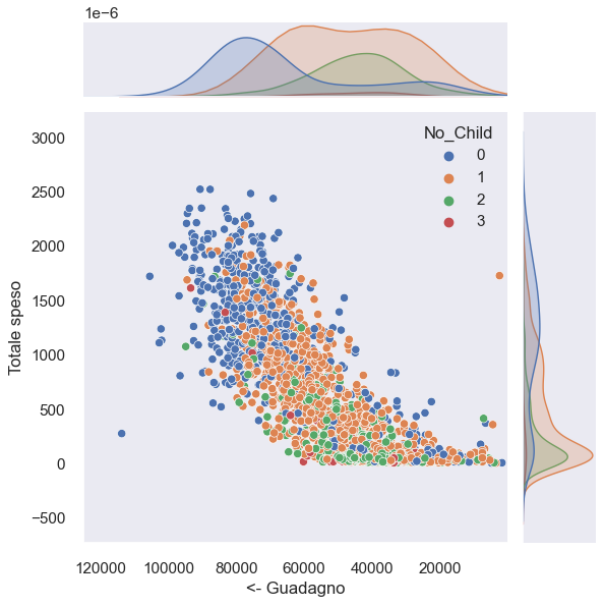

Figura 14: Scatter Plot per Guadagno e Totale speso

3 Clustering

Una volta svolte le analisi di tipo descrittivo sul dataset, al fine di comprenderne meglio le caratteristiche intrinseche, si procede con il clustering, una tecnica di apprendimento non supervisionata che permette di raggruppare elementi simili in insiemi di analoghe caratteristiche. Tutto questo viene svolto per ottenere una profilazione dei clienti, con lo scopo di trarne un vantaggio economico per la compagnia, in quanto permetterebbe di svolgere azioni di marketing mirate.

3.1 Preparazione del dataframe

Prima di procedere alle operazioni di clustering, sono state svolte alcune operazioni necessarie sul dataset per renderlo conforme alle necessità degli algoritmi che verranno utilizzati.

Innanzitutto, sono state eliminate quelle colonne ridondanti non utili per le operazioni di clustering, le quali sono state rimpiazzate da altre, secondo le metodologie illustrate nella fase di ETL; si sta parlando, quindi, dei campi "Education", "kidhome", "Teenhome". In aggiunta, sono stati rimossi l'attributo che fa riferimento alla data di iscrizione dei clienti "Dt_customer" e la colonna con valori booleani che indica se il cliente possiede figli o meno, perché ridondante data la presenza della *features* "No_Child", contenente il numero totale dei figli.

Successivamente, dato che il modello di Machine Learning non accetta variabili categoriche, si è proceduto alla trasformazione dei valori degli attributi "Education2" e "Marital Status" in valori di tipo numerico, 0 ed 1.

Inoltre, sono stati esclusi dall'algoritmo tutte quelle righe dei clienti con guadagni annui eccessivamente alti rispetto agli altri, considerati dei veri e propri outlier, i quali probabilmente sarebbero stati un ostacolo per la buona riuscita delle tecniche di Machine Learning.

Come ultimo passo, avendo constatato che i valori degli attributi appartenessero a scale di grandezze diverse, è stata effettuata un'operazione di standardizzazione, necessaria al fine di poter dare in input, ai modelli di clustering, dati con metriche simili. Per quest'ultima operazione si è scelto di utilizzare lo *ScandardScaler* del modello preprocessing di ScikitLearn.

La Figura 15 mostra il dataset al termine delle operazioni di preparazione, idoneo per l'applicazione delle successive tecniche di clustering.

```
data_cluster = pd.DataFrame(StandardScaler().fit_transform(data_outline), columns=data_outline.columns)
data_cluster.head()
```

	Marital_Status	Income	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	NumDealsPurchases	NumW
0	1.347625	0.314651	0.310830	0.974566	1.548614	1.748400	2.449154	1.480301	0.849556	0.361479	
1	1.347625	-0.254877	-0.380600	-0.874776	-0.638664	-0.731678	-0.652345	-0.635399	-0.735767	-0.168834	
2	-0.742046	0.965354	-0.795458	0.355155	0.568110	-0.175957	1.336263	-0.149031	-0.039771	-0.699147	
3	-0.742046	-1.206087	-0.795458	-0.874776	-0.563241	-0.667380	-0.506392	-0.586763	-0.755100	-0.168834	
4	-0.742046	0.322136	1.555404	-0.394659	0.417263	-0.217292	0.150396	-0.003121	-0.561768	1.422105	

Figura 15: Normalizzazione del dataframe

3.2 Clustering bidimensionale

Il primo tipo di clustering che si vuole effettuare prende in considerazione solamente due *features* specifiche del dataset. Basandosi sull'osservazione della heatman in Figura 16, sono stati scelti gli attributi "Tot_Mnt" e "Income", rispettivamente il totale speso e il guadagno annuo dei clienti, in quanto soggetti a una correlazione molto elevata rispetto alla combinazione tra gli attributi restanti. Tale osservazione, in realtà, si percepiva già nello Scatter Plot nella sezione della visualizzazione dati, in Figura 14, in cui si mostrava come questi due attributi fossero abbastanza omogenei nella spazio.

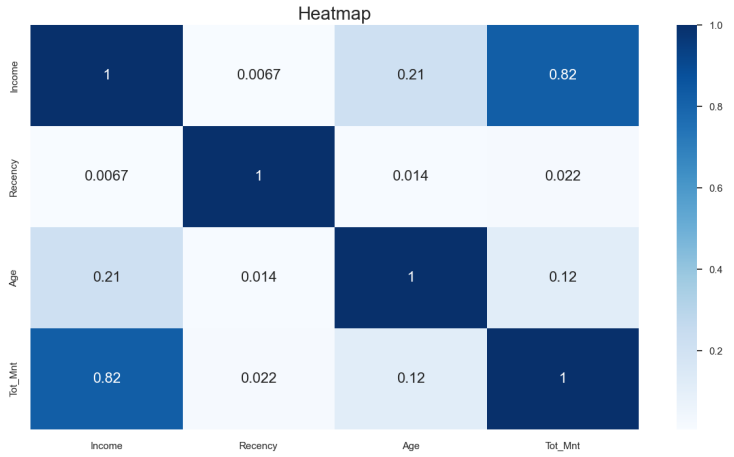

Figura 16: Heatmap

3.2.1 K-Means

Una volta scelti i due attributi su cui effettuare la prima analisi di clustering, si è scelto di utilizzare l'algoritmo K-Means. La caratteristica fondamentale di tale tecnica è che essa richiede di conoscere in anticipo il parametro *k*, ovvero il numero di cluster in cui si intende segmentare i dati. Al fine di individuare tale informazione, quindi, è stato utilizzato, tramite la libreria *Yellowbrick*, il cosiddetto metodo dell' *Elbow Method*, un'euristica che ricava, tramite iterazioni del K-Means, il parametro *k* ottimale. Il grafico ottenuto è mostrato nella Figura 17: esso individua un "gomito" in corrispondenza di *k=3*.

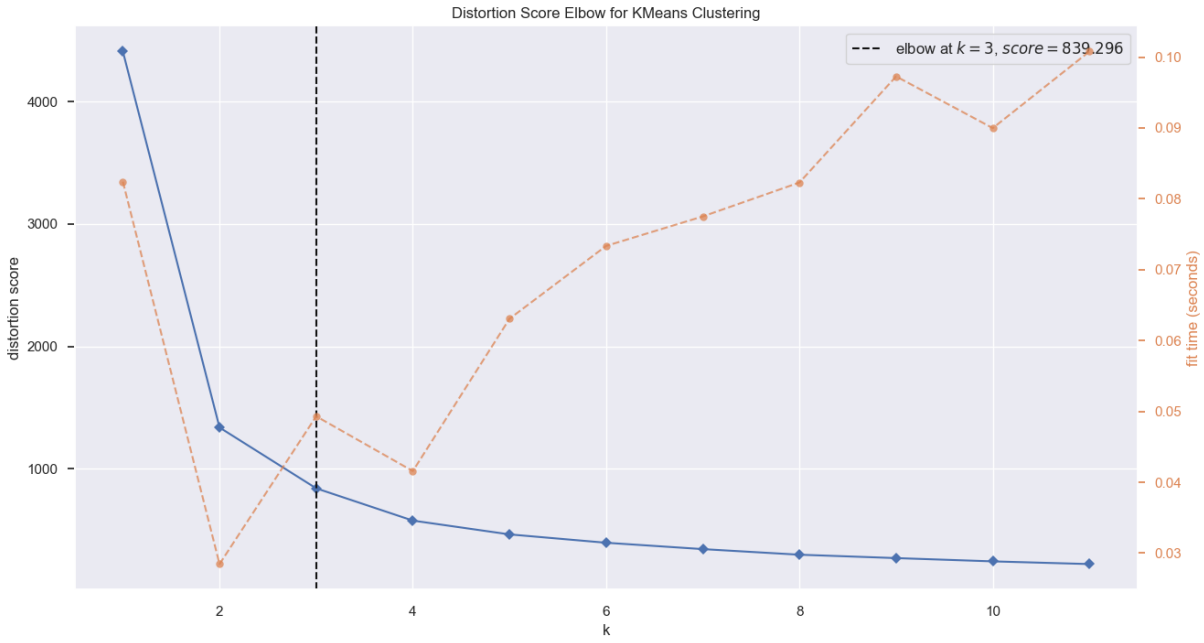

Figura 17: Elbow Method

Dunque, a questo punto si esegue il K-Means con numero di cluster preimpostato a 3. Il risultato del clustering è mostrato in Figura 18.

Come è possibile notare, i tre cluster sono distribuiti lungo il grafo in forma ascendente, ognuno centrato nel proprio centroide, evidenziato con una stella nera. Da questa preliminare rappresentazione, inoltre, si può distinguere come ogni cluster contenga un sottoinsieme differente di clienti: il cluster A è formato da clienti che guadagnano poco e allo stesso tempo spendono anche poco; il cluster intermedio, cioè il C, è costituito da clienti che guadagnano nella media e allo stesso tempo spendono, in prevalenza,

nella media; il cluster B, invece, comprende la clientela che guadagna molto e allo stesso tempo spende anche molto all'interno della compagnia.

Tuttavia, si ricorda che sugli assi non sono riportati i valori assoluti dei guadagni e della spesa totale bensì i valori normalizzati a seguito della precedente operazione di standardizzazione.

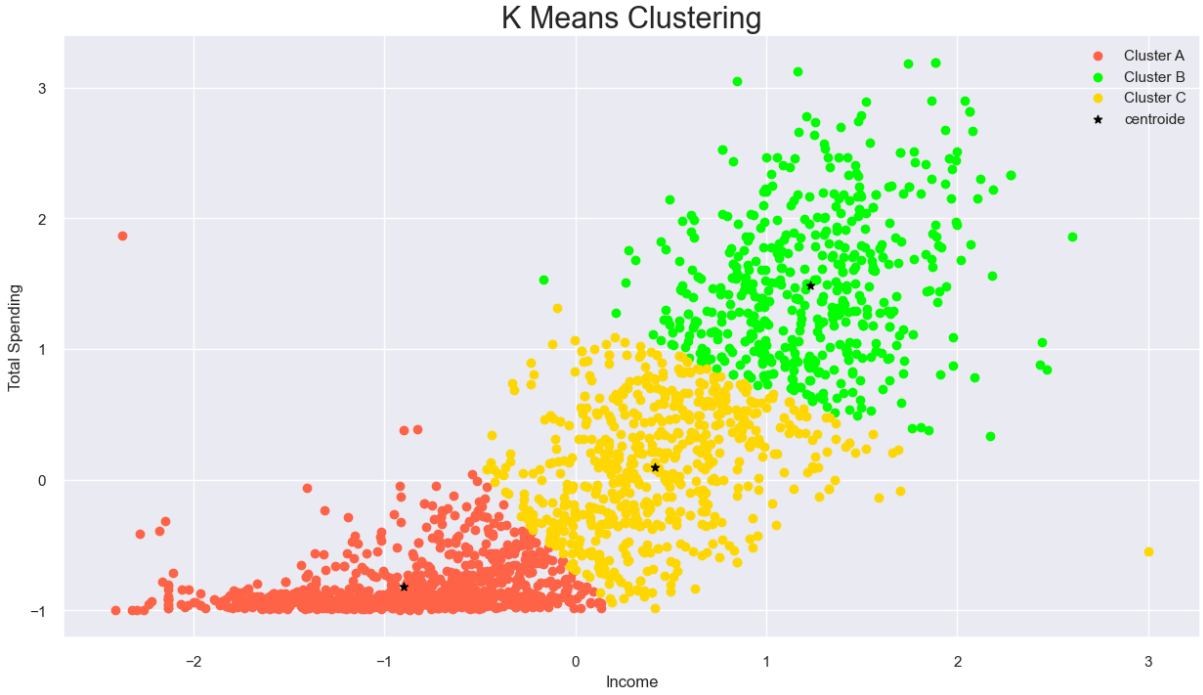

Figura 18: K-Means

Dopodichè, per valutare la bontà dei risultati ottenuti, è stata disegnata la metrica della *Silhouette*. Essa assegna un valore in un range compreso tra -1 e +1, fornendo un'indicazione di quanto un determinato punto appartiene correttamente al cluster a cui è stato classificato.

Il grafico della Silhouette è mostrato in Figura 19. Come si può osservare, si ottiene uno score medio vicino allo 0.5, il quale viene superato da una buona quantità di elementi, a prova del fatto che sono ben rappresentati dal proprio cluster. Fanno eccezione alcuni elementi, seppur pochi, appartenenti al cluster 2, i quali presentare uno score negativo, a indicare che potevano essere rappresentati meglio da uno altro gruppo di cluster.

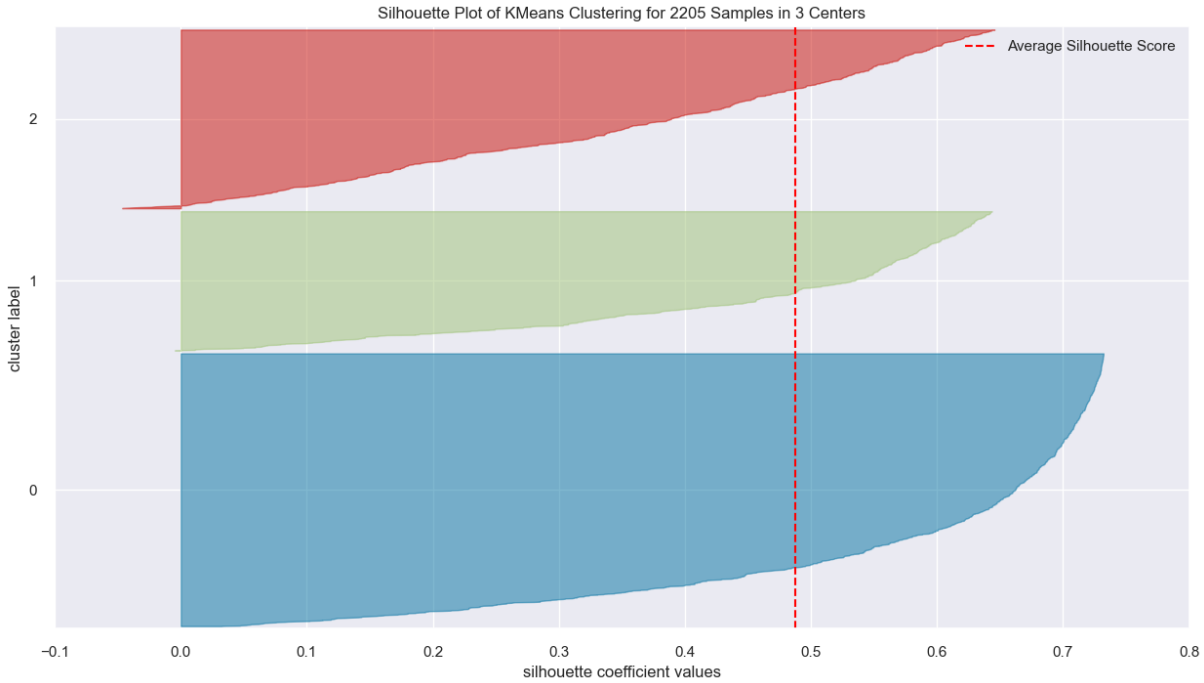

Figura 19: Silhouette K-Means

3.2.2 Clustering gerarchico

In seguito, per avere un termine di paragone, sugli stessi attributi considerati per il K-Means, si è voluto verificare se utilizzando un altro algoritmo di machine learning, il clustering gerarchico in questo caso, avesse prodotto dei risultati analoghi.

In generale, il clustering gerarchico non richiede a priori la definizione del numero di cluster da ricercare e prevede che i gruppi siano annidati e organizzati come un albero gerachico. Esso, inoltre, si basa su un approccio di tipo bottom-up: inizia considerando le singole unità statistiche per poi aggregarle nei vari gruppi secondo determinati requisiti di similarità.

In Figura 20 è mostrato il risultato ottenuto per mezzo del modello *AgglomerativeClustering* di *Sklearn*, avendo definito un numero di cluster pari a 3 ed il criterio di merge "ward", il quale minimizza la devianza totale dal centroide del gruppo da unire.

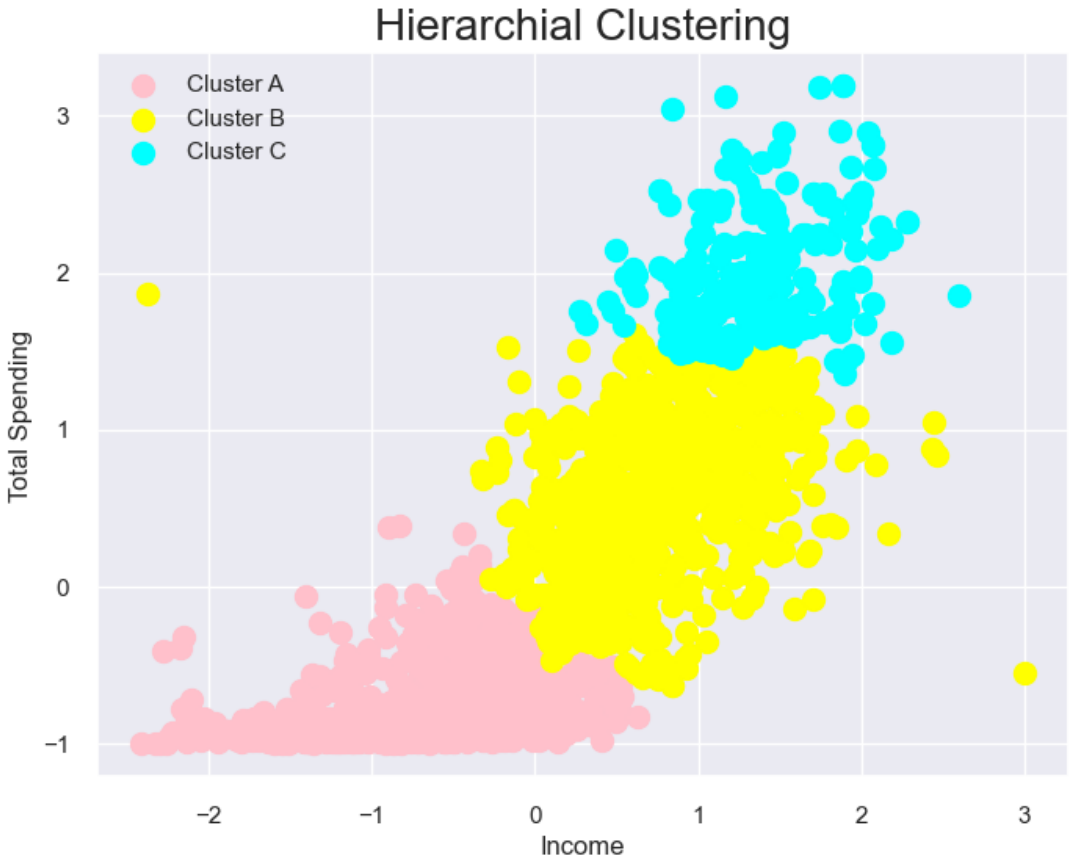

Figura 20: Clustering gerarchico

La suddivisione in cluster ottenuta in seguito all'applicazione del clustering gerarchico non è molto differente da quella individuata dal K-Means; fatta eccezione per alcuni outlier, in particolare collocato in posizione (x=-1.5, y=1.9) appartenerte al cluster intermedio B mentre il K-Means lo classificava nel cluster A.

Infine, anche qui è stata svolta l'analisi sulla bontà dei risultati tramite la Silhoutte, che ha prodotto un risultato analogo al caso precedente e, per tale ragione, viene omessa la sua rappresentazione.

3.3 Clustering multidimensionale - PCA

Si vuole svolgere un'ulteriore piccolo passo in avanti con la volontà di effettuare analisi che non siano più circoscritte a solo due *features*, ma che permettano di avere un quadro generale.

In tal senso si considerano, dunque, tutte le colonne del dataset, utilizzando una tecnica che permetta di ridurre le dimensionalità del dataset ma, al tempo stesso, conservandone tutte le informazioni risalienti: la cosiddetta PCA. Per mezzo dell'utilizzo di questa tecnica, si riesce a condensare l'intero dataset in un dataframe di due sole colonne, alle quali vengono compresse tutte le informazioni originarie.

Si è così ottenuto un array a due sole dimensioni, come esplicitato in Figura 21.

```
#visualizzo l'array a due dimensioni condensate tramite la PCA
pca_array
```

```
array([[ 0.31584517,  2.30838502],
       [-0.25552528, -1.30580489],
       [ 0.96464435, -0.90021833],
       ...,
       [ 0.25961604,  1.70177793],
       [ 0.85051676, -0.5431803 ],
       [ 0.05947926, -1.33144435]])
```

Figura 21: Array PCA

3.3.1 DBSCAN

Una volta aver eseguito la riduzione delle dimensioni tramite la tecnica del PCA, si è passati alla visualizzazione del clustering, costituita dall'utilizzo di un nuovo algoritmo di Machine Learning, il DBSCAN.

Questo modello permette di fare clustering con una tecnica basata sulla densità che risulta differente rispetto al K-Means, il quale parte da un numero predefinito di cluster, e quindi di centroidi, ed evolve assegnano i punti al centroide più vicino. Non avendo la necessità di conoscere in anticipo il numero di cluster, il DBSCAN ha però bisogno di due parametri di input: *epsilon*, cioè la "distanza-soglia" sotto la quale due punti sono considerati vicini, ovvero appartenenti allo stesso cluster; *MinPts*, che rappresenta il numero minimo di punti per formare una regione densa.

Per stimare l'*epsilon* è stato utilizzato il *NearestNeighbors*: l'epsilon ideale è quello che corrisponde al punto della curva in cui la pendenza aumenta drasticamente, in maniera simile ad un andamento esponenziale. Dal grafico di Figura 22 si può osservare come il valore ricavato è pari a *epsilon*=0.2.

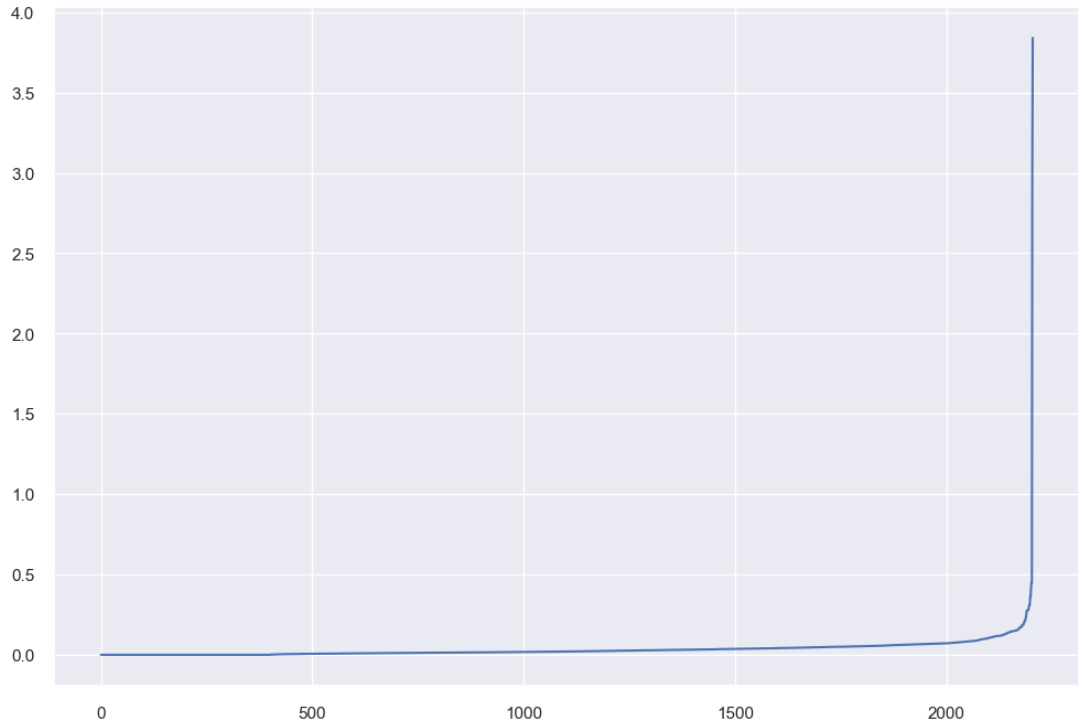

Figura 22: NearestNeighbors

Per quanto riguarda il parametro *MinPts*, in seguito a delle ricerche, è emerso che generalmente si sceglie un valore maggiore o uguale al doppio del numero di features del dataset; a tal proposito, si è posto *MinPts*=7.

Il clustering ottenuto per mezzo dell'algoritmo DBSCAN è mostrato nella Figura 23.

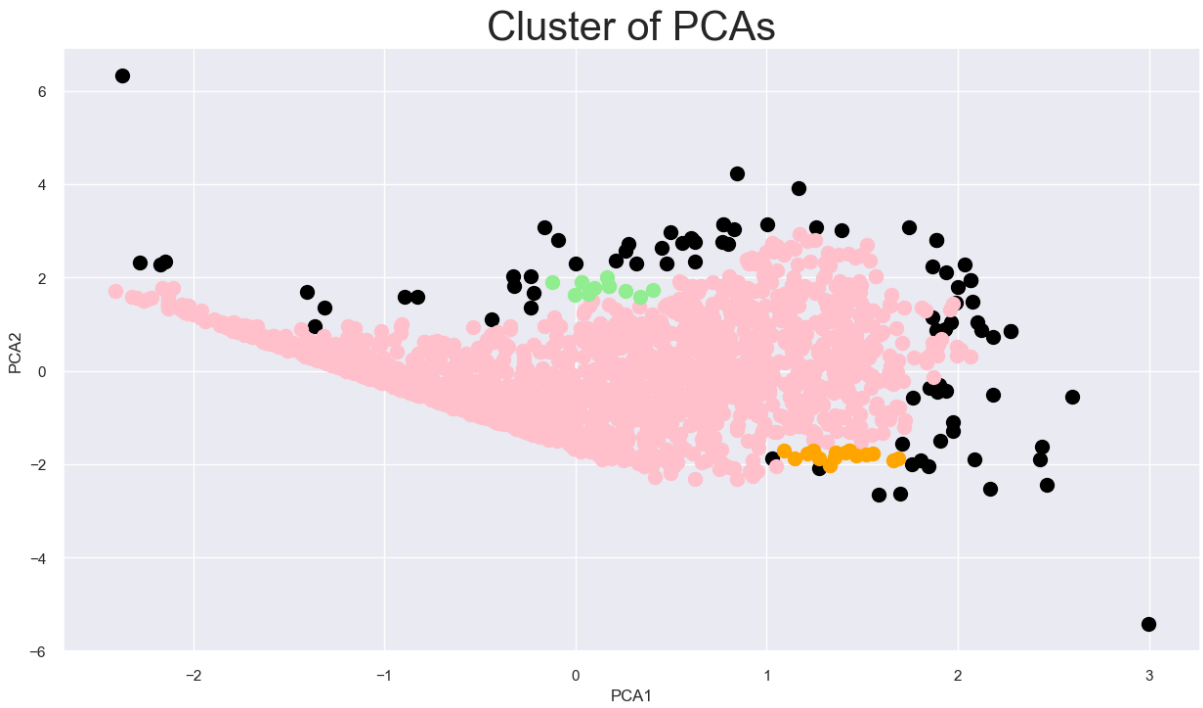

Figura 23: DBSCAN

Il risultato del DBSCAN mostra tre cluster, di cui uno occupa quasi totalmente lo scatterplot. Oltre a ciò, è possibile notare la molteplicità dei punti che non sono stati inseriti all'interno di nessun cluster e, perciò, etichettati come outlier, rappresentati con punti di colore nero.

Alla luce di ciò, appare chiaro come il DBSCAN non sia l'algoritmo idoneo per questa tipologia di trattazione, risultando magari migliore per distribuzioni a spirale o per dataset i cui elementi abbiano una densità pressochè omogenea.

3.3.2 K-Means

A seguito dei risultati insoddisfacenti ottenuti dal DBSCAN, si è passati alla visualizzazione del clustering del dataset elaborato tramite la PCA costruita per mezzo dell'algoritmo K-Means.

Nella Figura 24 è riportato l'elbow method, il quale suggerisce di considerare 3 cluster.

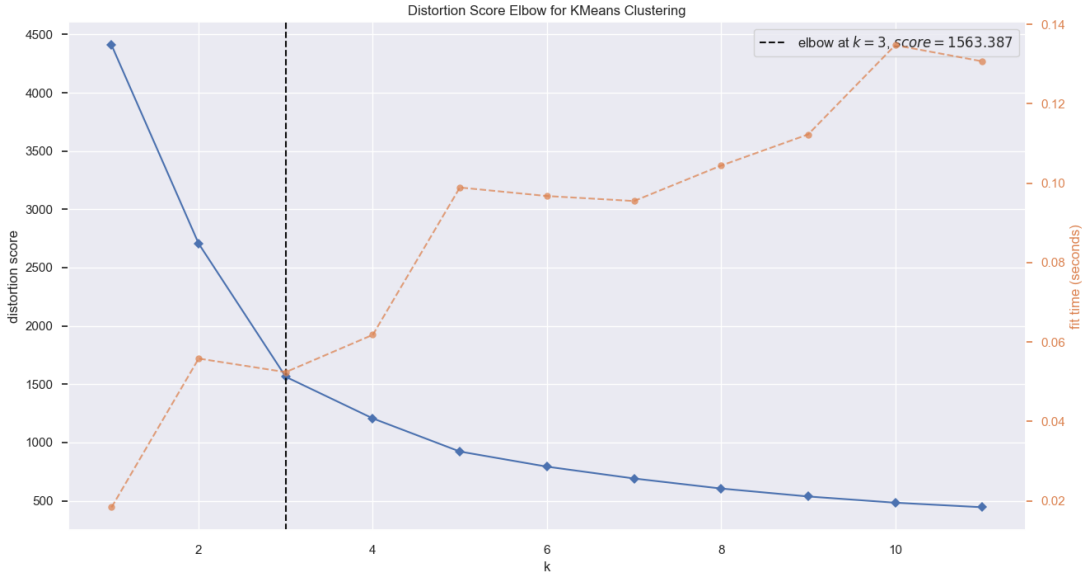

Figura 24: Metodo del gomito

La rappresentazione del K-Means sulla PCA viene mostrata in Figura 25. Come è possibile notare, i cluster individuati dall'algoritmo appaiono densi e ben distinti tra loro, a prova del fatto che il K-Means sia riuscito a distribuire ciascun elemento all'interno dei cluster, fatta eccezione per alcuni outlier.

Per confermare la bontà del modello, in Figura 26 è rappresentata la Silhouette. Diversamente dal K-Means precedente effettuato soltanto su due specifiche features del dataset, si ha uno score leggermente più basso, ma che risulta comunque accettabile. Inoltre, si può notare la presenza di elementi del cluster 1, seppur minimi, con uno score negativo, i quali potevano essere rappresentati meglio da un altro cluster.

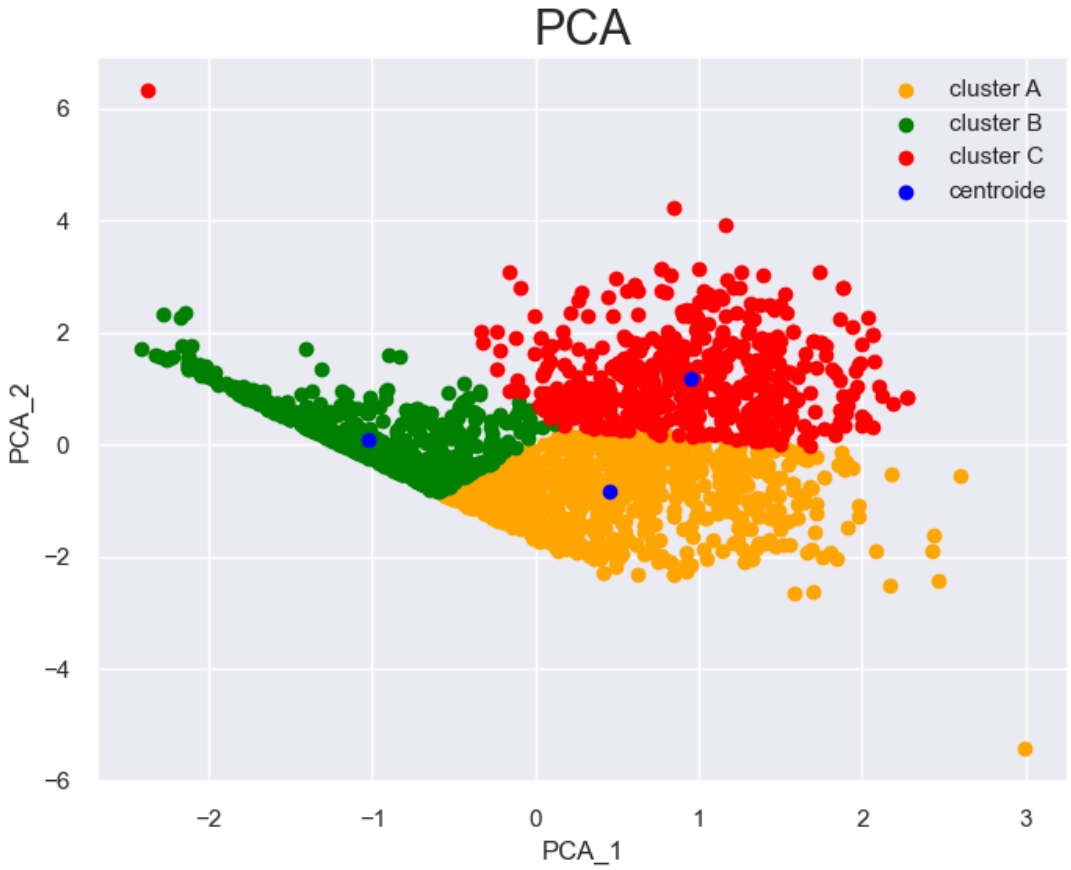

Figura 25: K-Means su PCA

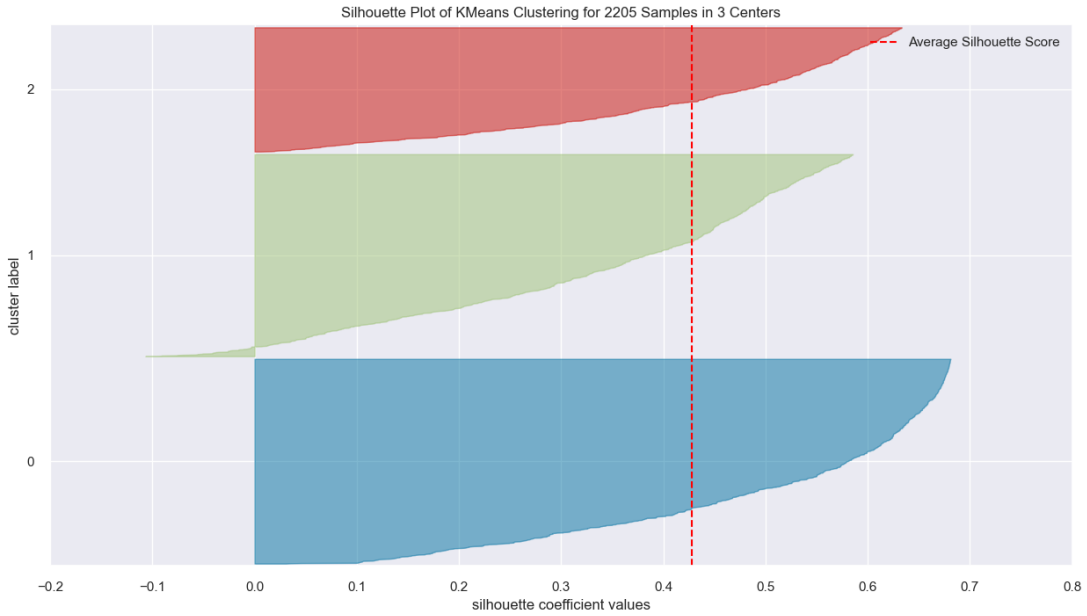

Figura 26: Silhouette

3.4 Interpretazione dei risultati dei cluster - K-means

Giunti a questo punto, il lavoro è stato proseguito concentrandosi in maniera approfondita sulle caratteristiche interessate dai singoli cluster.

Lo scopo di questa procedura, svolta tramite l'esplorazione dei dati, è quello di riuscire ad effettuare una vera e propria profilazione dei clienti della compagnia, la quale, attraverso comportamenti differenti da parte dei gruppi dei clienti, può agire con azioni pubblicitarie e offerte mirate verso determinate porzioni di clientela.

Innanzitutto, nel grafico a barre in 27 si può osservare come la dimensione dei cluster sia abbastanza distribuita, nonostante non tutti siano popolati dalla stessa quantità di clienti; infatti, mentre i cluster 0 e 1 possiedono una dimensione simile, il cluster 2 è caratterizzato da una quantità minore di clienti.

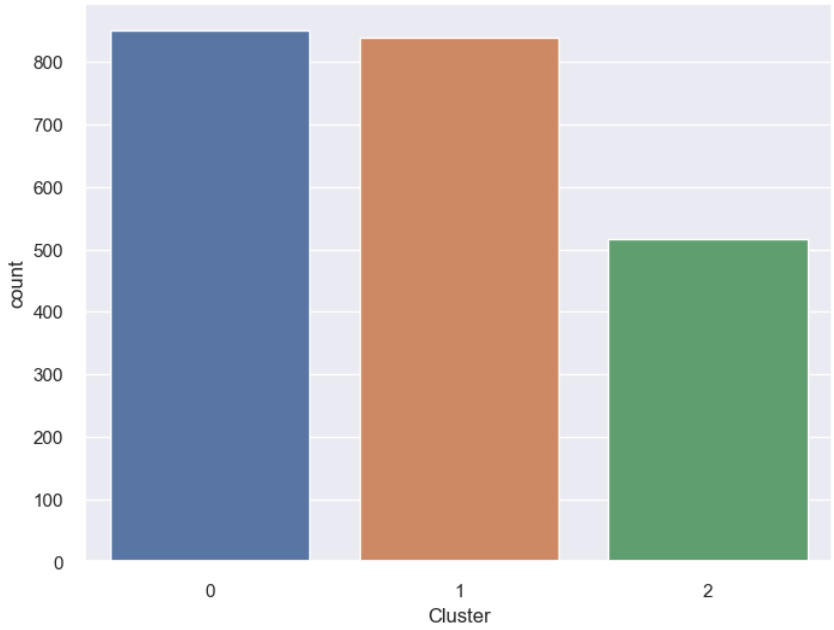

Figura 27: Numerosità Cluster

Successivamente, è stata effettuata l'analisi sulla distribuzione dell'età mediante boxplot, illustrata nella Figura28), da cui si evince che la mediana del cluster 0 si aggira intorno ai 40 anni, mentre per i restanti due cluster la mediana è compresa tra i 47-49 anni.

In linea generale, dunque, la compagnia sembra essere attirata da una clientela piuttosto giovanile. Inoltre, i dati sono abbastanza simmetrici, in quanto la mediana dei rispettivi boxplot è collocata esattamente al centro del contenitore. Fa eccezione il cluster 2, la cui mediana è posizionata leggermente al di sotto ma tale aspetto può essere ignorato in quanto comunque vicino al 50° percentile.

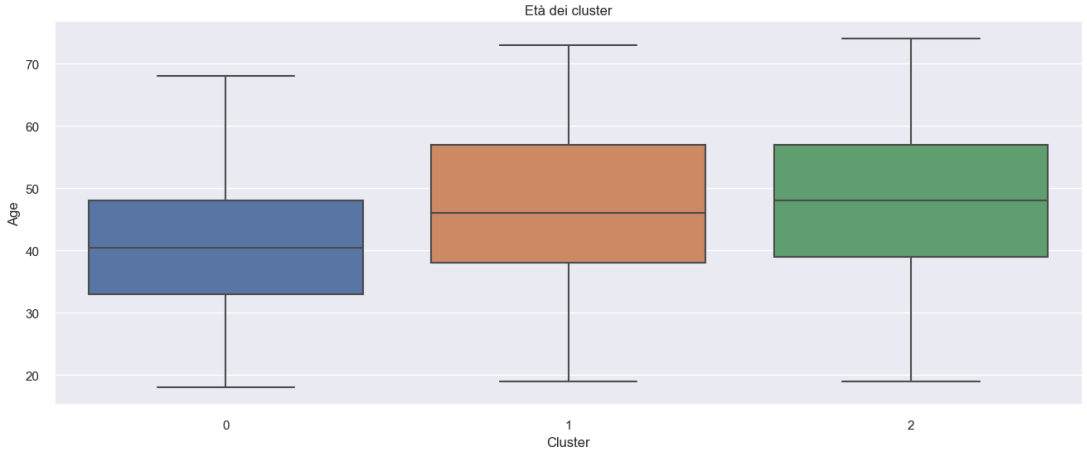

Figura 28: Età Cluster

In seguito, è stato analizzato lo stato civile dei cluster; il valore "0" corrisponde ai clienti in coppia, mentre il valore "1" corrisponde ai clienti single. Dalla Figura 29 si evince che tutti i cluster hanno una distribuzione che si concentra prevalentemente in "0".

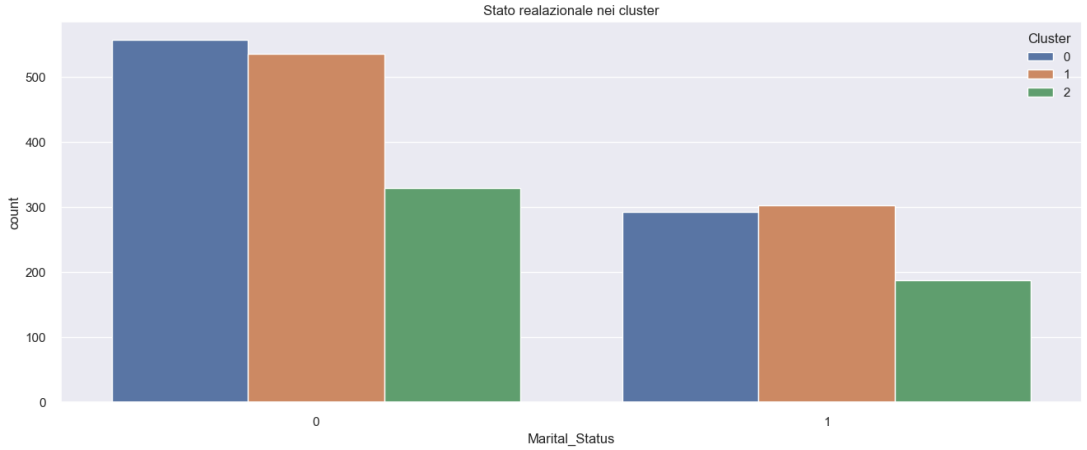

Figura 29: Marital Status Cluster

E' stato analizzato il numero di figli di ciascun cluster e dalla Figura 30 si può dedurre che il cluster 0 e il cluster 1 corrispondono ad una clientela che ha prevalentemente un figlio, mentre il cluster 2 corrisponde ad una clientela che tende a non avere figli.

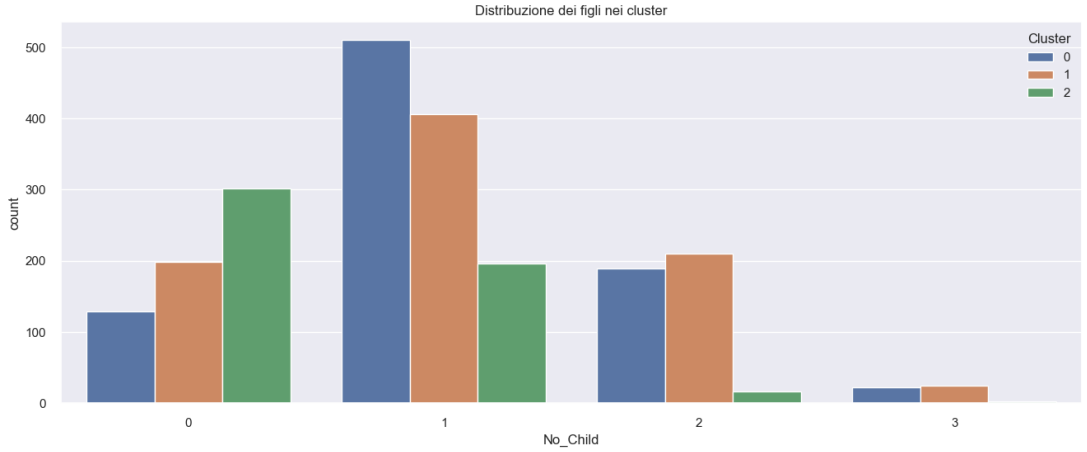

Figura 30: Figli Cluster

Una sostanziale differenza tra la ripartizione dei diversi cluster si può notare al livello di spesa totale da parte dei clienti. Come è osservabile nei boxplot in Figura31, il cluster 0 comprende clienti che spendono decisamente poco nei prodotti della compagnia, il cluster 1 è costituito da clienti che ricoprono una spesa nella media; il cluster 2, invece, comprende tutta la parte di clientela, probabilmente maggiormente interessante per la compagnia, in quanto tende a spendere molto di più della media.

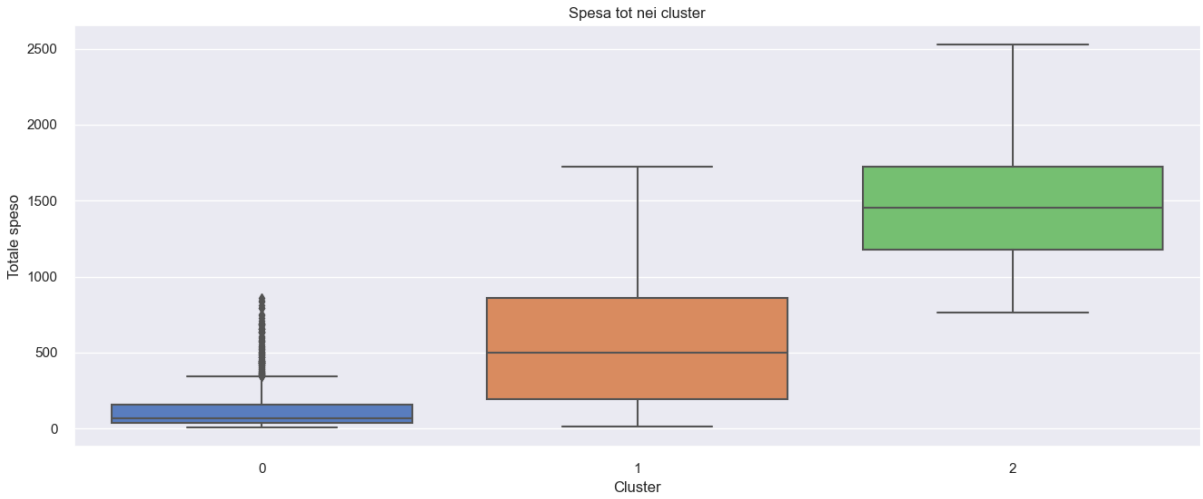

Figura 31: Totale speso Cluster

In ultima analisi, è stato visualizzato lo scatterplot raffigurante la spesa totale in funzione del guadagno annuo dei clienti, tramite il quale è possibile osservare, in Figura32, il cluster 0 corrisponde a clienti che guadagnano e spendono poco, il cluster 1 corrisponde a clienti che guadagnano tanto e spendono nella media ed, infine, il cluster 2 corrisponde a clienti che guadagnano e spendono tanto.

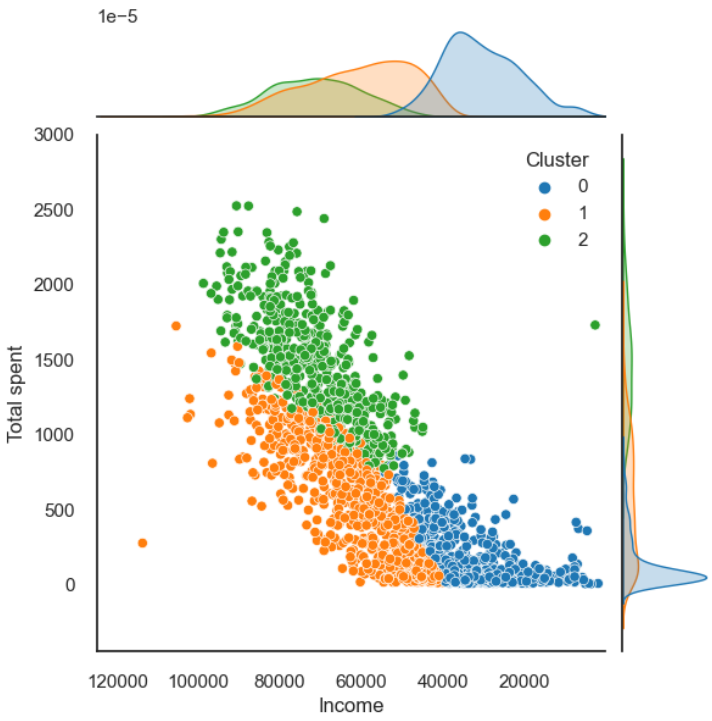

Figura 32: Scatterplot clustering

3.5 Profilazione dei cluster

Al termine dell'esplorazione dei dati, ci si è ritenuti soddisfatti concerne la comprensione di come l'algoritmo K-Means abbia classificato i clienti nei diversi cluster nonchè l'osservazione delle differenti abitudini di acquisto da parte dei clienti stessi.

Sulla base delle caratteristiche indicative dei tratti personali dei clienti, si è deciso di terminare l'analisi di clustering tracciando, in qualche modo, un profilo dei cluster formati al fine di giungere a una conclusione sui clienti definiti "oro" e su chi, invece, merita maggiore attenzione da parte del team di marketing del punto vendita.

Cluster 0: Cliente bronzo

- L'età si aggira intorno ai 40 anni;
- In coppia con prevalentemente un figlio;
- Guadagno basso e spesa bassa.

Cluster 1: Cliente argento

- L'età si aggira intorno ai 47 anni;
- In coppia con prevalentemente un figlio;
- Guadagno elevato e spesa nella media.

Cluster 2: Cliente oro

- L'età si aggira intorno ai 49 anni;
- In coppia con prevalentemente nessun figlio;
- Guadagno elevato e spesa alta.

4 Classificazione

In questo capitolo sarà implementata la tecnica di classificazione, un metodo di learning supervisionato che classifica i dati in categorie rilevanti, già conosciute a priori. Questo metodo ha lo scopo di delineare classi per assegnare un nuovo elemento alla classe appropriata.

In tale progetto è stata effettuata una classificazione binaria considerando la categoria "Child", la quale descrive la presenza o meno di un figlio all'interno di ciascun nucleo familiare preso in considerazione nella campagna di marketing.

4.1 Il preprocessing dei dati

La prima fase consiste nella manipolazione dei dati a disposizione tramite semplici operazioni di preprocessing, fondamentali per il raggiungimento degli obiettivi.

Innanzitutto, sono state eliminate tutte le colonne non significative per la classificazione e sono state prese in considerazioni gli attributi: "Income", Recency", "MntWines", "MntFruits", "MntMeatProducts", "MntFishProducts", "MntSweetProducts", "MntGoldProds", "NumDealsPurchases", "NumWebPurchases", "NumCatalogPurchases", "NumWebVisitsMonth", "Child", "TotMnt" e "Age".

In seguito è stata valutata la correlazione tra ciascuna coppia di variabili tramite la heatmap in Figura 33 e sono state eliminate tutte le categorie con correlazione maggiore di 0.7 al fine di evitare che l'operato dei classificatori possa risultare banale, in quanto si concentrerebbe soltanto sugli attributi che possiedono una forte correlazione trascurando, di fatto, tutti gli altri. Di conseguenza, sono state rimosse le variabili "TotMnt" e "NumCatalogPurchases".

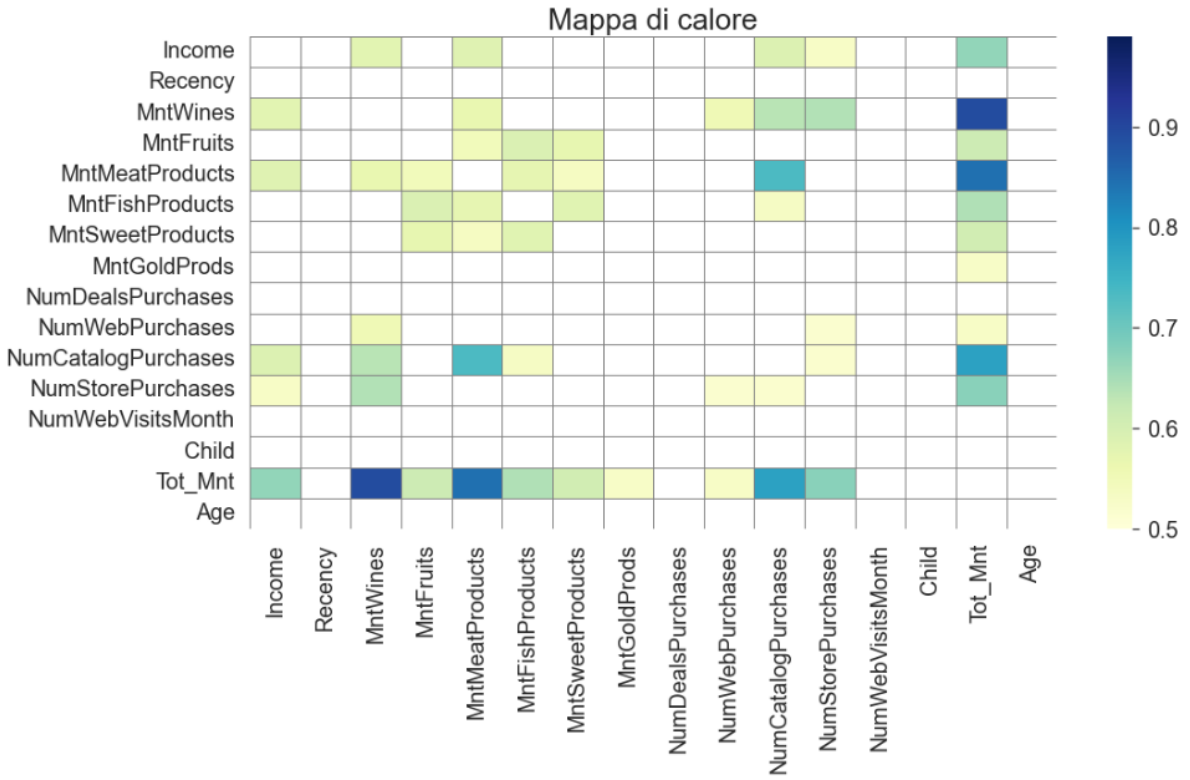

Figura 33: L'analisi della correlazione

Successivamente, sono state trasformate tutte le variabili categoriche in numeriche, aggiungendo una colonna per ogni possibile valore associato alla categoria stessa. In particolare, gli elementi di ciascuna colonna sono stati valorizzati con 0 o 1, in funzione della presenza o meno dell'attributo categorico nella riga del dataframe. Dunque, "MaritalStatus" e "Education2" sono state divise rispettivamente in "MaritalStatusInCouple" e "MaritalStatusSingle", "Education2Avanzata" e "Education2Base"; data la

forte correlazione tra ciascuna coppia di categorie creata, si è scelto di eliminare "MaritalStatusInCouple" e "Education2Base".

Il passo successivo è stato quello di normalizzare le colonne del dataset che presentavano ordini di grandezza differenti, in modo tale che il calcolatore non si concentrasse su una colonna piuttosto che su un'altra. Dunque, gli elementi sono stati trasformati in valori compresi tra 0 e 1.

Infine, è stato controllato il bilanciamento del dataset rispetto alla categoria "Child" e si è dedotto che circa il 30% degli utenti presi in considerazione in questa campagna di marketing non ha figli, mentre il restante 70% ha prole (Figura 34).

```
Childness: 633 With children: 1583
```

Figura 34: Output del codice per il controllo del bilanciamento del dataset

Dunque, è stato bilanciato il dataset concatenando, in un nuovo dataframe, gli utenti senza figli (633 elementi) e un sottoinsieme casuale degli utenti con figli, di lunghezza pari a 633. Conseguentemente, la lunghezza del nuovo dataframe è 1266.

4.2 Il training e i primi risultati della classificazione

La classificazione si sviluppa in due passi:

- al sistema sono forniti dati di training già categorizzati o etichettati, in modo tale che esso possa sviluppare una comprensione delle diverse categorie (il sistema viene addestrato);
- al sistema sono forniti dati sconosciuti, ma simili (dati di testing) e, basandosi sulla comprensione che esso ha precedentemente sviluppato a partire dai dati di training, classificherà i nuovi dati.

Per definire il modello possono essere utilizzati molteplici algoritmi che la libreria Scikitlearn mette a disposizione insieme a molte altre funzioni. Quelli che sono stati utilizzati in questo progetto sono i seguenti modelli:

- LogisticRegression
- DecisionTreeClassifier
- SVC
- RandomForestClassifier
- XGBClassifier

Come precedentemente anticipato, la classificazione è stata effettuata considerando la categoria "Child". Innanzitutto, è stato istanziato un random state pari a 42 in modo che i parametri iniziali dei diversi test effettuati sul classificatore siano sempre gli stessi. Per poter allenare il modello è stato necessario dividere il dataset in train e test e per tale ragione l'80% del dataset è stato utilizzato per il train e il 20% per il test.

Tra i modelli precedentemente elencati è possibile misurarne la qualità attraverso vari indici; in questo progetto è stata presa in considerazione l'accuratezza, ovvero il rapporto tra il numero di dati classificati correttamente e il numero di dati classificati. I risultati di accuratezza ottenuti dai diversi modelli sono i seguenti:

- LogisticRegression: 0.81
- DecisionTreeClassifier: 0.85
- SVC: 0.83
- RandomForestClassifier: 0.89
- XGBClassifier: 0.9

É possibile visualizzare la bontà della classificazione tramite il barplot in (Figura 35), in cui è evidenziata la variazione e la deviazione standard della cross validation dei vari algoritmi.

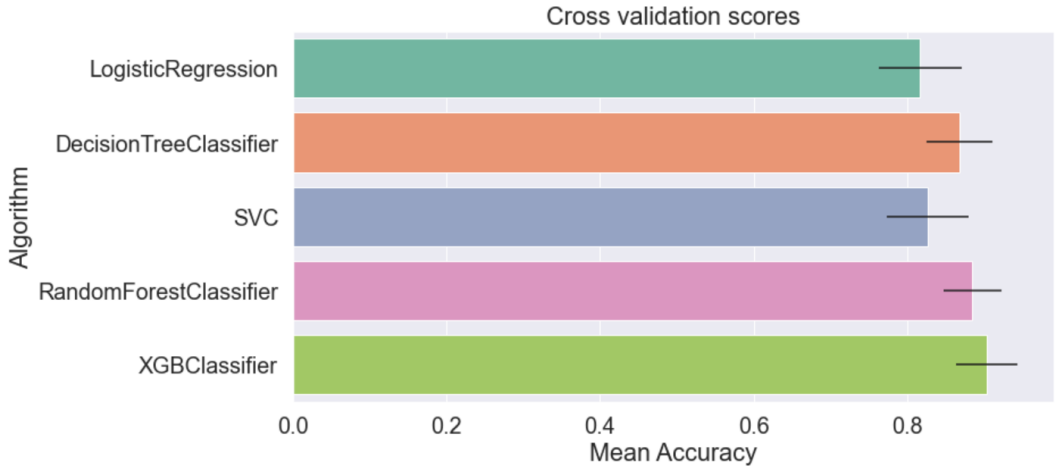

Figura 35: La cross validation

In particolare, si può notare che il classificatore con una "cross validation accuracy" migliore è l'XGB, con una deviazione standard di circa 0.1; segue il Random Forest, il quale presenta anch'esso una deviazione standard di circa 0.1.

Inoltre, sono state visualizzate le matrici di confusione di ciascun classificatore (Figura 36), ovvero delle tabelle in cui sono riportate le previsioni in corrispondenza delle colonne e lo stato effettivo in corrispondenza delle righe; tramite questo strumento è possibile valutare le prestazioni dei modelli considerati. Tutte le previsioni corrette si trovano sulla diagonale della tabella, dove vi sono True Positive e True Negative, mentre le previsioni errate si trovano esternamente alla diagonale e sono i False Positive e i False Negative.

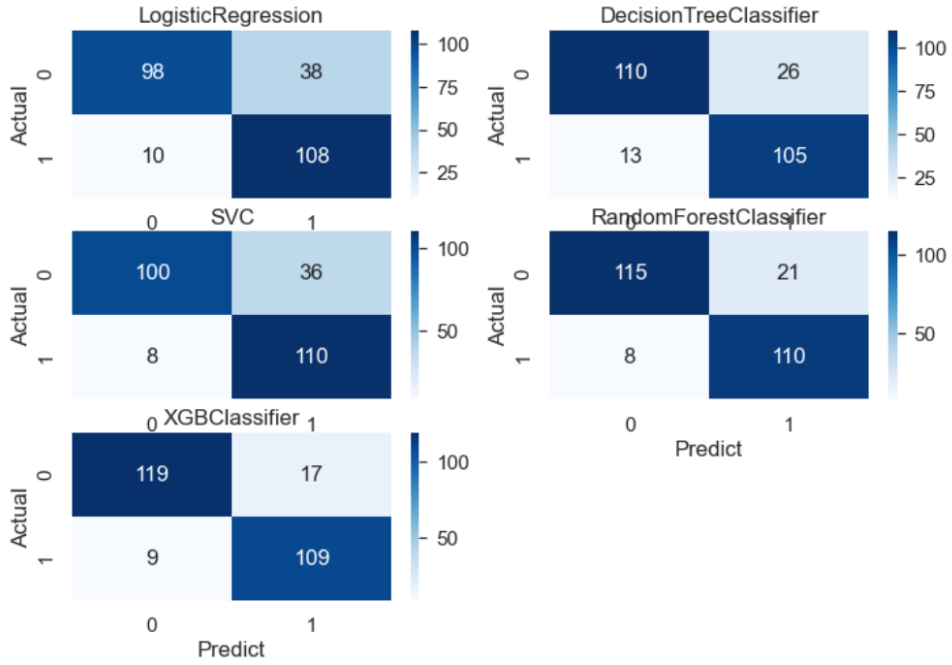

Figura 36: Le matrici di confusione

Dalle matrici di confusione relative ai risultati di tale classificazione si evince come tutti gli algoritmi abbiano adoperato una buona classificazione, con alti valori di True Positive e True Negative. In supporto a tale strumento, è riportato il classification report (Figura 37), il quale mostra, per ogni classificatore, ulteriori metriche di valutazione delle performance.

LogisticRegression Classification Report:

	precision	recall	f1-score	support
0	0.89	0.75	0.82	136
1	0.76	0.90	0.82	118
accuracy			0.82	254
macro avg	0.83	0.82	0.82	254
weighted avg	0.83	0.82	0.82	254

DecisionTreeClassifier Classification Report:

	precision	recall	f1-score	support
0	0.88	0.89	0.88	136
1	0.87	0.86	0.86	118
accuracy			0.87	254
macro avg	0.87	0.87	0.87	254
weighted avg	0.87	0.87	0.87	254

SVC Classification Report:

	precision	recall	f1-score	support
0	0.88	0.79	0.83	136
1	0.79	0.87	0.83	118
accuracy			0.83	254
macro avg	0.83	0.83	0.83	254
weighted avg	0.84	0.83	0.83	254

RandomForestClassifier Classification Report:

	precision	recall	f1-score	support
0	0.89	0.92	0.90	136
1	0.90	0.86	0.88	118
accuracy			0.89	254
macro avg	0.89	0.89	0.89	254
weighted avg	0.89	0.89	0.89	254

XGBClassifier Classification Report:

	precision	recall	f1-score	support
0	0.92	0.90	0.91	136
1	0.89	0.91	0.90	118
accuracy			0.91	254
macro avg	0.90	0.91	0.91	254
weighted avg	0.91	0.91	0.91	254

Figura 37: Il classification report

Per completare l'analisi dei risultati si è ricorso all'utilizzo della curva ROC (Receiver Operating Characteristics), una tecnica statistica che misura l'accuratezza di un test diagnostico lungo tutto il range dei valori possibili e utilizzabile solo in caso di classi binarie. Questo grafico mostra il trade-off tra Sensitività e Specificità mano a mano che si varia la soglia o la regola scelta per classificare un record. L'area sottostante alla curva ROC, chiamata AUC (Area Under the Curve), è una misura di accuratezza e in base a tale valore il test si definisce:

- non informativo, se
$$AUC = 0.5$$
- poco accurato, se
$$0.5 < AUC \leq 0.7$$

- moderatamente accurato, se
$$0.7 < AUC < 0.9$$
- altamente accurato, se
$$0.9 \leq AUC < 1$$
- perfetto, se
$$AUC = 1$$

La (Figura 38) mostra le curve relative ai vari modelli. I valori di AUC risultano in media moderatamente accurati e i valori più alti si registrano in corrispondenza di XGB e Random Forest.

Figura 38: Le curve ROC

4.3 La Grid Search

Un fattore importante nelle prestazioni di questi modelli sono i loro iperparametri, ovvero variabili che l'utente specifica in genere durante la creazione del modello stesso. Una volta impostati valori appropriati per questi iperparametri, le prestazioni di un modello possono migliorare in modo significativo.

La Grid Search è una tecnica di ottimizzazione che tenta di calcolare i valori ottimali degli iperparametri.

Per determinare i classificatori da utilizzare in tale algoritmo, si è fatto ricorso alla heatmap (Figura 39) in modo da valutare le correlazioni tra i diversi modelli. I modelli con correlazione più alta restituiscono una predizione simile e per tale ragione sono stati scelti Decision Tree e SVC, che hanno una correlazione pari a 0.62, quindi predicono in modo pressochè differente. Gli iperparametri utilizzati sono mostrati in Figura 40.

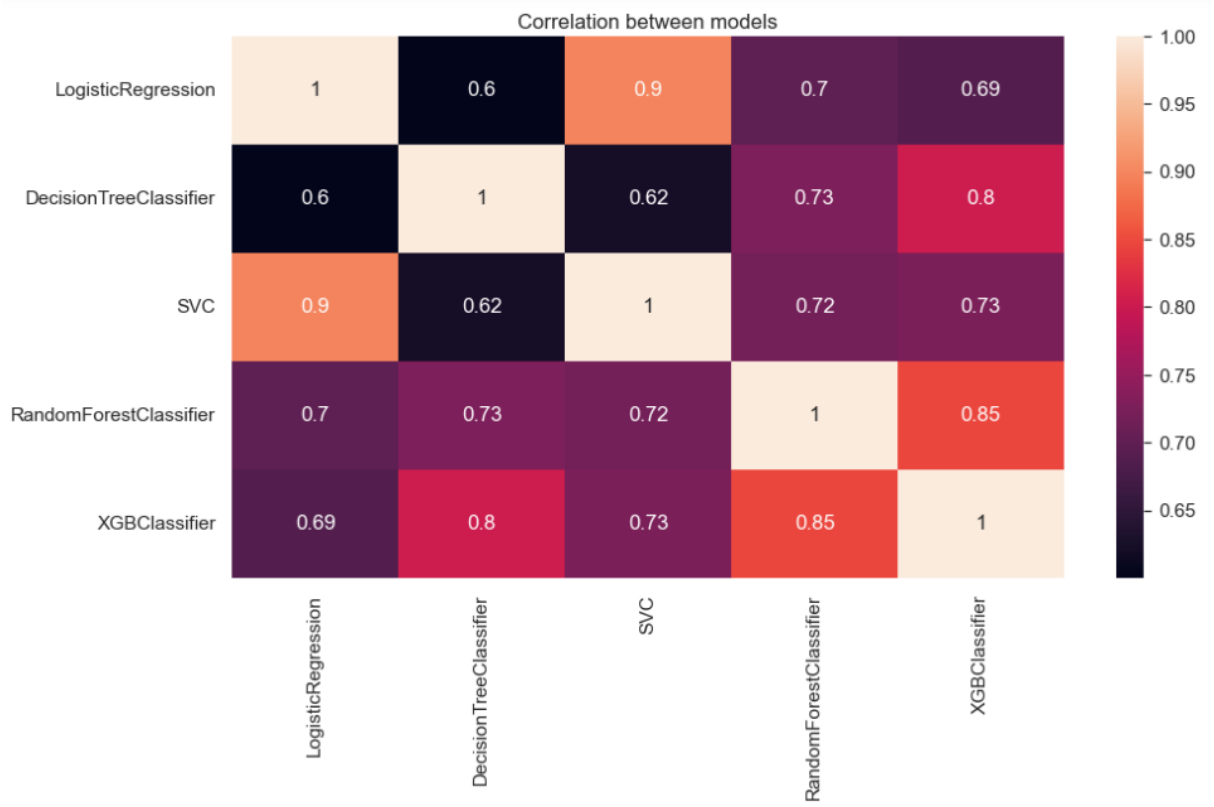

Figura 39: L'analisi delle correlazioni dei modelli di classificazione utilizzati

```
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"]
           }

SVC_param = {
    'C': [0.1,1, 10, 100],
    'gamma': [1,0.1,0.01,0.001],
    'kernel': ['rbf', 'poly', 'sigmoid'],
        }
```

Figura 40: Gli iperparametri di Decision Tree e SCV

Una volta eseguito l'algoritmo, i risultati ottenuti sono riportati in Figura 41. Si può notare che si ha un leggero miglioramento in corrispondenza di Decision Tree e un peggioramento in corrispondenza di SVC.

```
Decision Tree and SVC score without GridSearchCV:  0.873 0.904
Decision Tree and SVC score with GridSearchCV:  0.88 0.897
```

Figura 41: Il confronto dei risultati senza e con GridSearch

4.4 Il model ensemble

Si è voluto effettuare un "ensemble" dei modelli, il quale utilizza più classificatori contemporaneamente per ottenere delle permormance migliori, rispetto ai risultati restituiti dai singoli modelli.

E' stato effettuato un ensemble dei modelli Decision Tree e SVC, gli stessi utilizzati per la Grid Search. L'accuracy è pari a 0.89763, quindi prossimo a quello ottenuto con Grid Search, ma più affidabile, dato l'utilizzo di più classificatori contemporaneamente.

In Figura 42 è riportata la curva ROC e Figura 43 è riportata la curva precision-recall. In Figura 44 si può vedere quanto potrebbe imparare questo modello e, dato che il cross validation score e il training score sono vicini, è possibile dedurre che il modello riesce a riconoscere quello che gli viene passato in input.

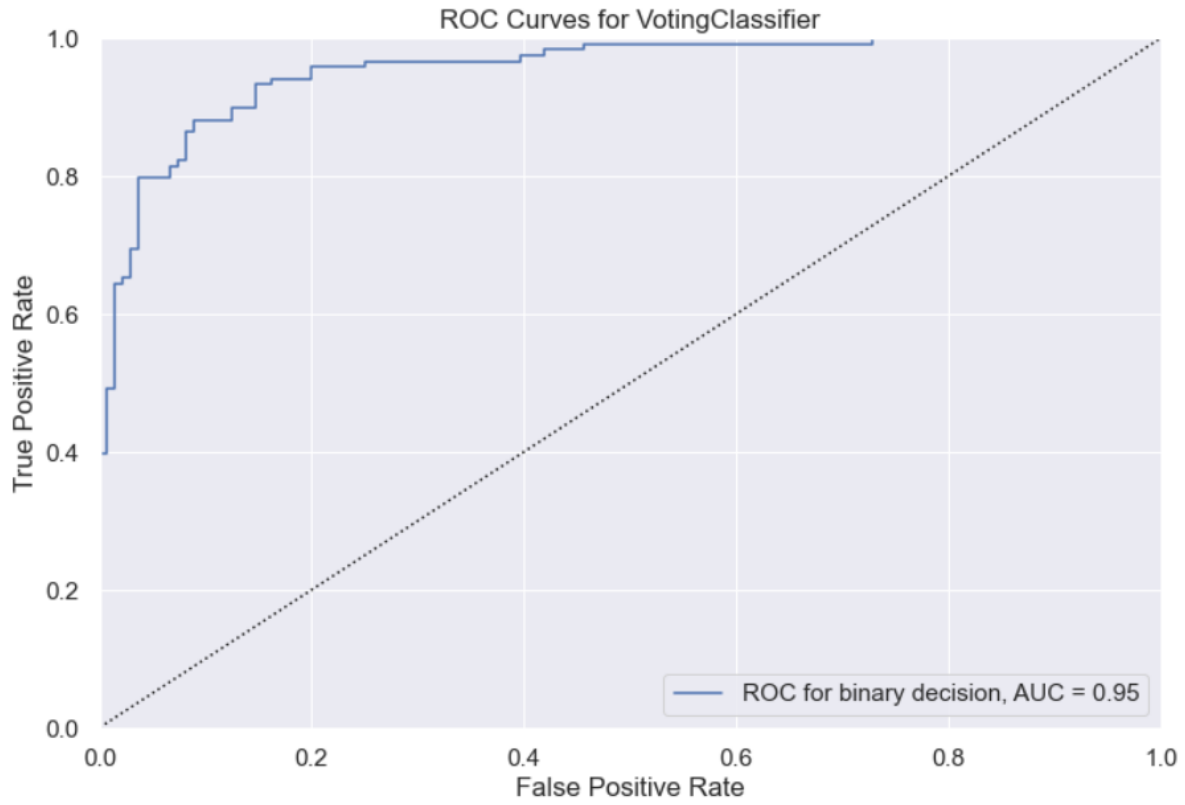

Figura 42: La curva ROC

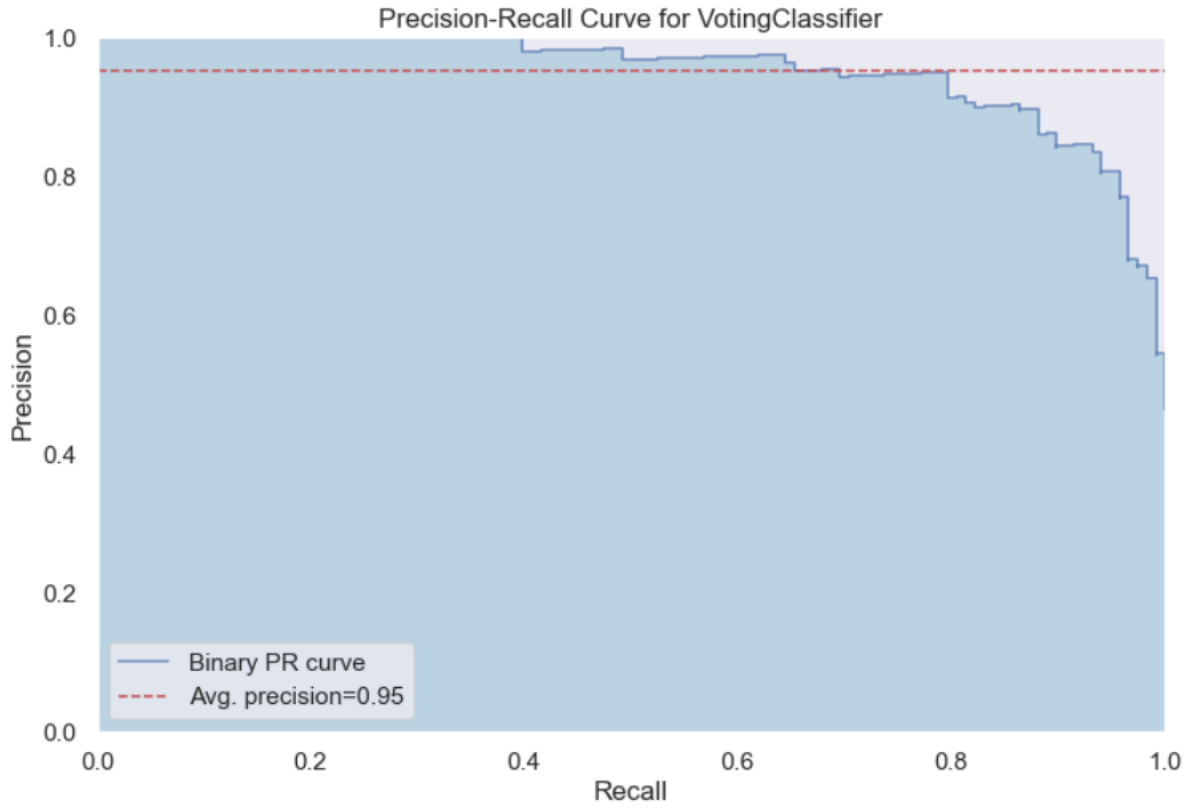

Figura 43: La curva precision-recall

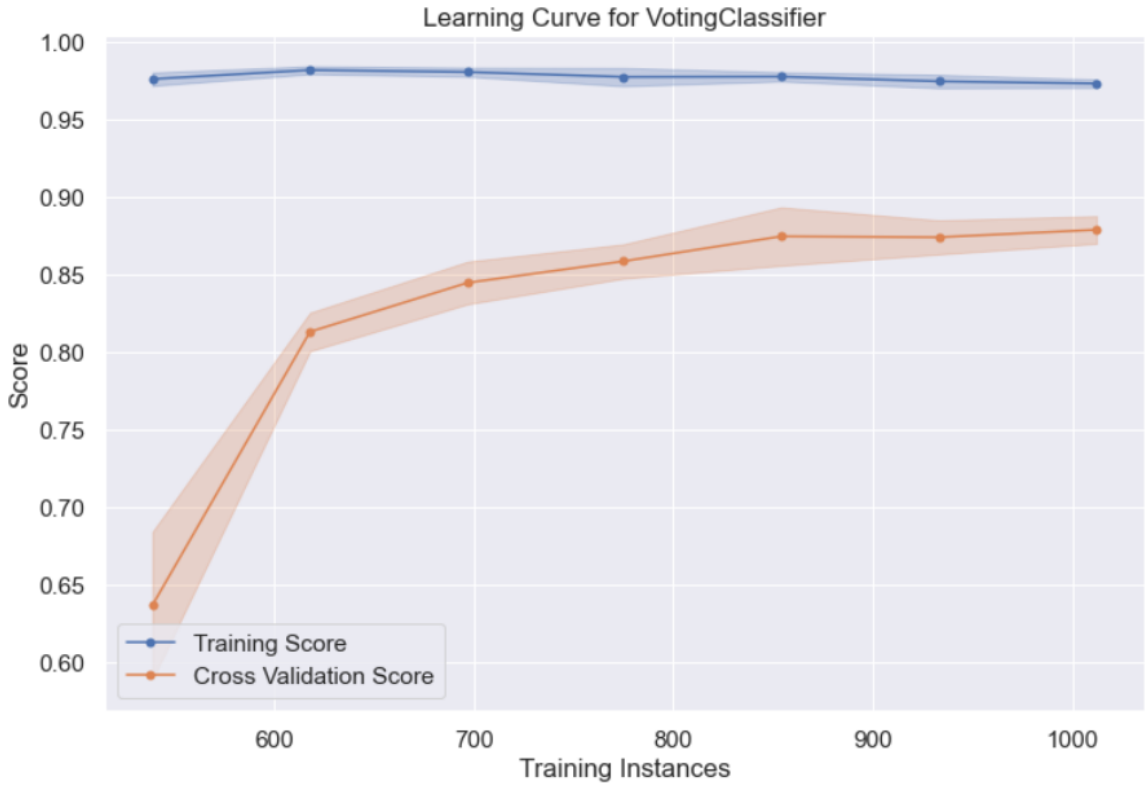

Figura 44: La learning-curve

5 Serie temporali

L'ultima parte del progetto di Python riguarda l'analisi di serie temporali, dal trend alle previsioni di forecasting che possono essere realizzate attraverso specifici algoritmi. Gran parte del lavoro illustrato in tale sezione è stato sviluppato mediante l'utilizzo della libreria *Statsmodels*, la quale consente di esplorare dati, eseguire test statistici e stimare modelli statistici.

In particolare, l'obiettivo prefissato è stato quello di effettuare un'analisi univariata sull'andamento del consumo elettrico domestico in un appartamento a San Jose per due anni e di cui si è provato a prevedere l'andamento futuro; i dati sono stati raccolti da contatori intelligenti e condivisi dalla stessa compagnia energetica.

La serie temporale di riferimento è reperibile al seguente link:

`https://www.kaggle.com/datasets/jaganadhg/house-hold-energy-data` di cui si riporta una breve descrizione degli attributi in esso presenti.

Attributo	Descrizione
TYPE	"Consumo elettrico" per tutte le osservazioni
DATE	Data del consumo elettrico
START TIME	Ora di inizio del consumo
END TIME	Ora di fine del consumo
USAGE	Consumo in kWh
UNITS	Unità di misura. Per tutte le osservazioni è il kWh
COST	Costo del consumo in $
NOTE	Colonna per lo più vuota

A tal proposito, le colonne che sono state utilizzate per le analisi di forecasting sono 'DATE' contenente il trading day di riferimento e la colonna 'USAGE' la quale si riferisce al consumo elettrico alla specifica data.

Gli step che sono stati seguiti nel corso di analisi della serie temporali sono i seguenti:

- ETL - Preparazione serie temporale
- Stazionarità della serie (parametro d)
- Stima parametri p e q
- Applicazione Sarimax e Auto-Arima
- Forecasting

5.1 ETL - Preparazione serie temporale

Prima di iniziare con le analisi vere e proprie, è stata eseguita una breve fase di ETL. Innanzitutto, sono state rimosse tutte le colonne che non risultano utili ai fini dell'elaborato lasciando, quindi, le sole features relative alla data e al consumo di energia elettrica.

Successivamente è stata manipolata la colonna 'DATE' utilizzando la funzione to_datetime per convertire i valori nel tipo datetime ed, in seguito, sono state raggruppate le date ogni 7 giorni, così da avere un andamento temporale che seguisse un evoluzione settimanale.

Tale scelta è giustificata dal fatto che avere dati giornalieri (o addirittura dati a intervalli di 15 minuti rilevati dai contatori intelligenti come nel nostro in questione) possiede delle complicanze nel prevedere il comportamento futuro sulla base del trend osservato.

Infine, è stata applicata la funzione set_index così da trasformare il riferimento temporale nell'indice del dataset.

In questo modo è stata ottenuta una serie che presenta un unico valore di consumo elettrico alla settimana, data dalla somma dei consumi giornalieri della settimana stessa ottenendo un totale di 106 elementi.

Terminata la fase di preparazione, si è ottenuto un dataset pronto per essere analizzato (Figura 45.

DATE	USAGE
2016-10-16	27.84
2016-10-23	48.92
2016-10-30	87.44
2016-11-06	66.76
2016-11-13	78.16

Figura 45: Dataset definitivo con andamento settimanale

L'andamento dell'utilizzo di energia elettrica nel periodo considerato è mostrato nel grafico che segue (Fig. 46)

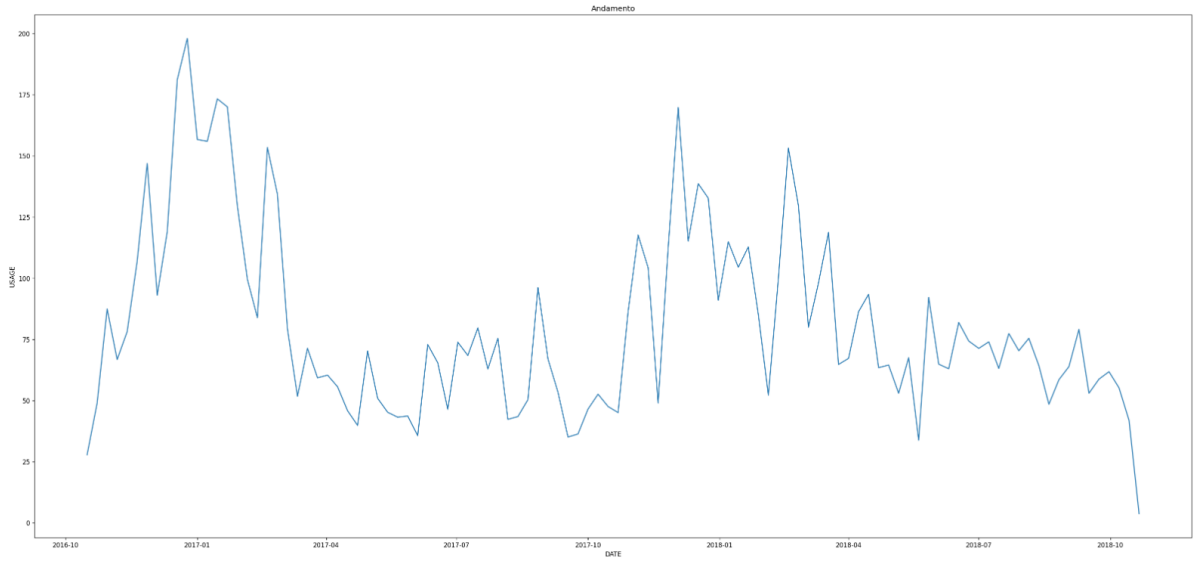

Figura 46: Andamento consumo elettrico

Osservando il grafico emerge subito come la serie presenti un andamento con una certa stagionalità: il consumo energetico è maggiore nei mesi invernale e diminuisce nei mesi estivi.

5.2 Stazionarietà della serie temporale - parametro d

L'obiettivo di tale sezione è testare la stazionarietà della serie temporale. La stazionarietà è un aspetto cruciale di una serie temporale tale per cui i suoi valori non dipendono dal tempo e, in essa, la media e la varianza devono essere costanti nel tempo. In altre parole, una serie temporale è stazionaria se la distribuzione di probabilità della stessa non cambia se essa viene traslata nel tempo.

Per lo studio della stazionarietà è stata utilizzata una delle tecniche di Unit Root Test, Augmented Dickey Fuller test (ADF test), un metodo che parte da un ipotesi nulla: la serie temporale non è stazionaria; dopodichè si calcola il p-value e se quest'ultimo è $<$ **0.05** si rigetta l'ipotesi nulla e si suppone che la serie sia stazionaria.

Com'era chiaramente visibile già dalla Fig.46, la serie temporale considerata non è stazionaria, ed infatti l'ADF test ha restituito un valore del p-value pari a 0.429175, dunque, la serie non è stazionaria. Ha seguito un operazione di differenziazione partendo inizialmente con un solo ordine di differenziazione poichè una serie troppo differenziata potrebbe perdere di significato. Dopodichè, è stato nuovamente eseguito il test sulla stazionarietà ottenendo un esito positivo per cui differenziando una sola volta e, dunque, ponendo *d=1* si è ottenuta una serie effettivamente stazionaria.

5.3 Stima parametri p e q

Per poter prevedere l'andamento futuro della serie con il modello SARIMAX è necessario stimare i parametri d, p, q:

- d determina l'ordine di differenziazione (I)
- p determina il numero di termini autoregressivi (AR)
- q determina il numero di termini di media mobile (MA)

Si ricorda come il parametro d è stato posto uguale a 1, effettuando un solo ordine di differenziazione, come già discusso nel punto precedente. Per quanto riguarda la stima dei parametri p e q, invece, sono stati calcolati i grafici ai autocorrelazione e autoccorelazione parziale (Fig. 47 e Fig. 48).

Il parametro p è stato stimato osservando il grafico di autocorrelazione parziale in base a quanti lag si trovano al di sopra del limite di significatività. In maniera analoga, è stato stimato il parametro q che indica quanti termini sono necessari per rimuovere qualsiasi autocorrelazione nella serie stazionaria;il grafico che si analizza è quello dell'autocorrelazione.

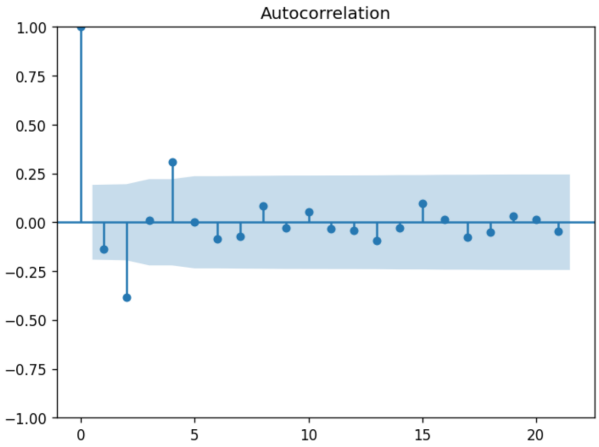

Figura 47: Autocorrelazione - stima parametro q

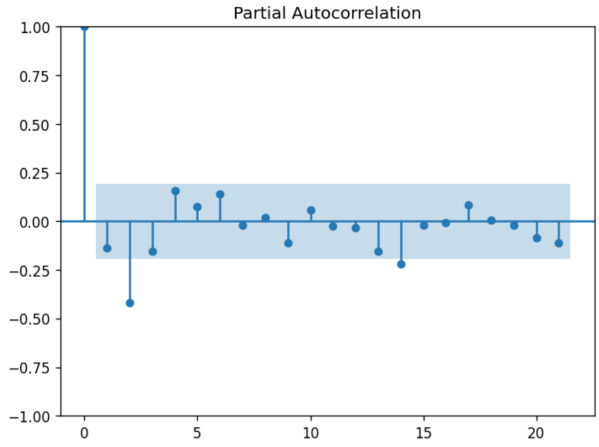

Figura 48: Autocorrelazione parziale - stima parametro p

Dallo studio dei grafici si è deciso, dunque, di porre sia il valore di p che quello di q uguali a 2.

Inoltre, grazie ad una funzione messa a disposizione da statsmodel, *seasonal_ decompose* si riescono ad evidenziare le componenti trend e stagionalità che caratterizzano la serie (Fig. 49).

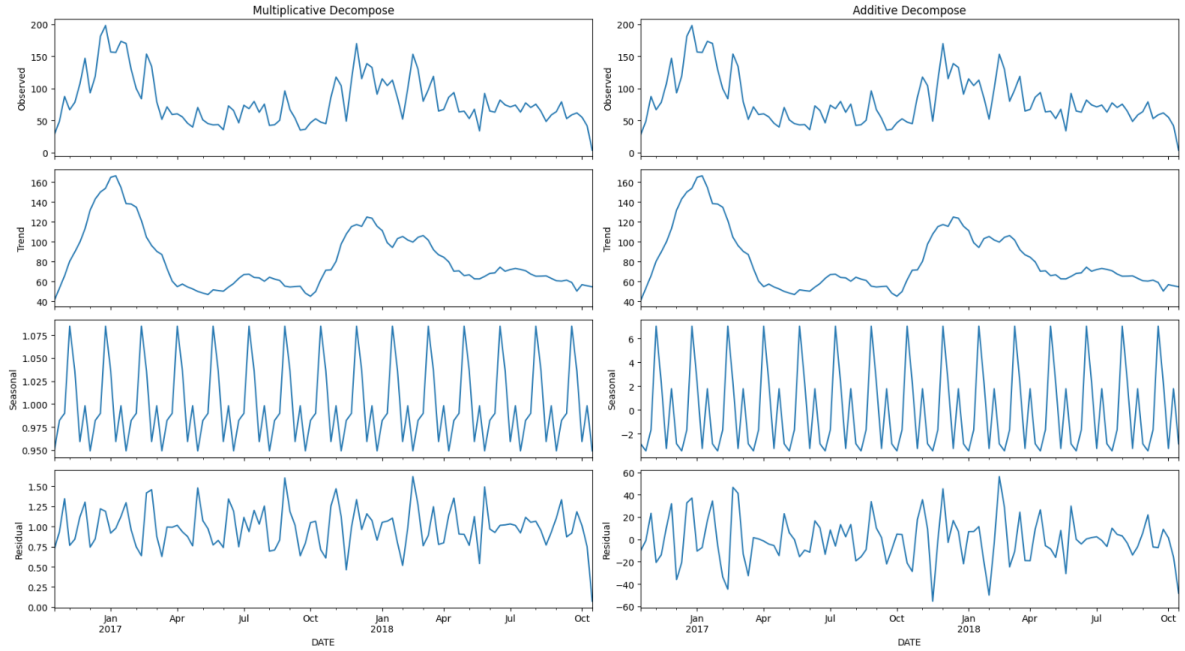

Figura 49: Applicazione della funzione *seasonal_ decompose*

Il grafico mostra come la serie presenti una componente stagionale di cui bisogna tener conto nella costruzione del modello SARIMAX. Sono stati aggiunti, quindi, i parametri *(P, D, Q)m* relativi alla parte stagionale dove *m* indica la periodicità della serie. Nel caso in questione si è deciso di porre *m=52*, data la frequenza a intervalli settimanali cui sono stati raggruppati i dati della serie. Inoltre, la componente stagionale viene implementata in Python semplicemente aggiungendo al modello il parametro *seasonal_ order*.

Riassumendo, il modello SARIMAX che si vuole ottenere è così formato:

- SARIMAX: (2,1,2) X (0,0,1)51

5.4 Applicazione modello SARIMAX

Una volta conclusa la scelta dei parametri, si ha tutto il necessario per allenare il modello SARIMAX ed osservare le proprie caratteristiche. Tramite il sommario, infatti, si è potuto rilevare, oltre al p-value dei coefficienti, il parametro AIC, un criterio che fornisce una misura della qualità di stima di un modello tenendo conto sia della bontà di adattamento che della complessità del modello stesso.

Nel caso di studio, in seguito a svariati esperimenti, si è constatato che i valori più bassi, sia in termini di p-value che di AIC, si ottengono con la configurazione selezionata a partire dall'osservazione dei grafici per cui ci si può ritenere soddisfatti della scelta dei parametri.

Prima di utilizzare tale modello per effettuare previsioni di serie temporali, è necessario assicurarsi che lo stesso modello abbia acquisito informazioni adeguate dai dati. Per verificare ciò, può essere d'aiuto lo studio dei residui al fine di assicurarsi che non ci siano pattern particolari e che abbiano una media nulla e una varianza abbastanza contenuta. In altre parole, se il modello costruito è buono, i suoi residui dovrebbero apparire come rumore bianco.

La Fig. 50 mostra la diagnostica del modello.

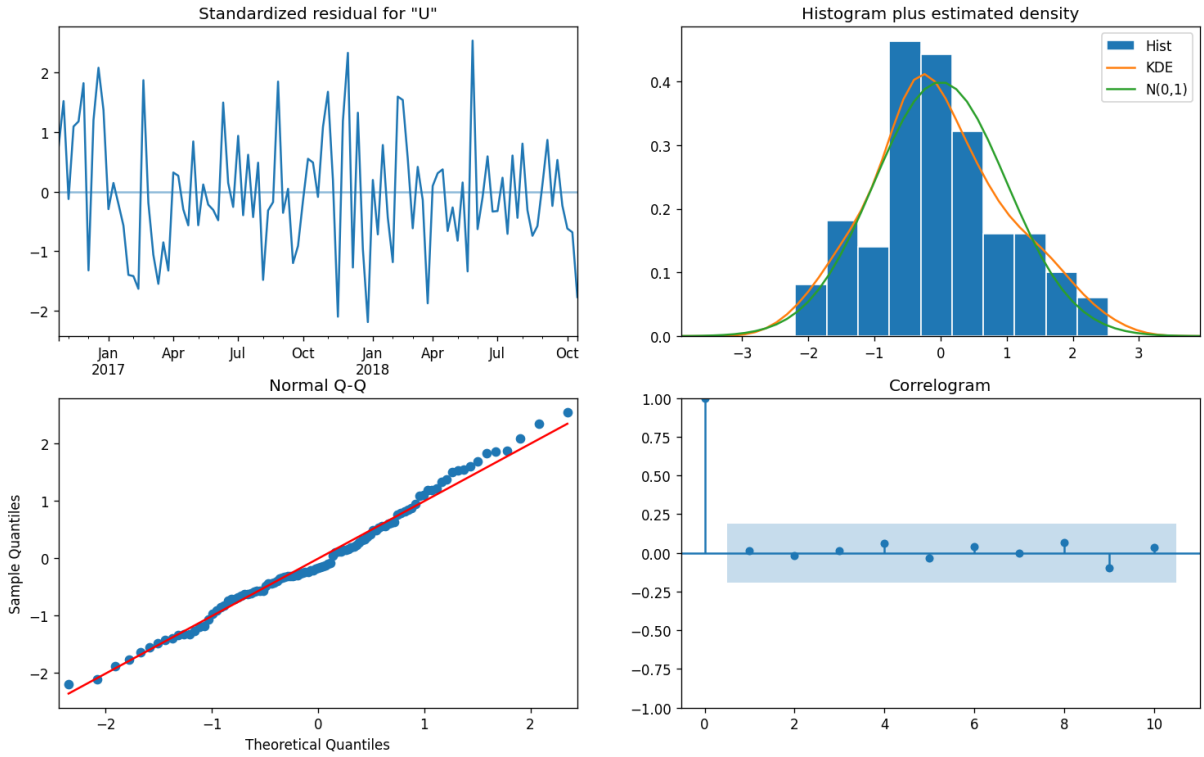

Figura 50: Diagnostica del modello

Dunque, analizzando i grafici (in senso orario dal grafico in alto a sinistra):

- *Residuo standardizzato*: non sono presenti modelli evidenti nei residui per cui si può affermare che la selezione della configurazione discussa precedentemente punta a un buon modello.
- *Istogramma più stima kde*: l'istogramma mostra la distribuzione misurata dei residui mentre la linea arancione mostra la curva KDE (versione smussata dell'istogramma); la linea verde, invece, illustra una distribuzione normale. Per avere un buon modello, la linea arancione dovebbe essere simile a quella verde. Nel caso in questione la densità dei residui appare normalmente distribuita con una media di circa 0 e le due curve sono molto simile, anche se nella parte di destra sembrano leggermente discostarsi, pur seguendo lo stesso andamento.
- *Correlogramma*: tutti i lag sono all'interno della regione di ammissibilità, il che conferma la bontà del modello.
- *Q-Q normale*: la maggior parte dei punti si trovano sulla linea retta, indicando una distribuzione normale dei residui

Alla luce di queste considerazioni, si può confermare che la configurazione indicata dalla ricerca empirica sembra essere un buon modello.

5.5 Modelli di previsione a confronto

Verificata la bontà della configurazione del modello creata a partire dai grafici di autocorrelazione e autocorrelazione parziale, si è passati ad affrontare il tema della previsione sfruttando gli algoritmi messi a disposizione dalla libreria di Python, Statsmodels.

L'idea di base è quella di plottare sullo stesso grafico i forecasting ottenuti dall'allenamento del modello SARIMAX con la stima dei parametri indicata dalla ricerca empirica e la configurazione degli ordini consigliato dal metodo *auto-arima*, appartenente alla libreria *pmdarima* dell'ecosistema Python, il quale determina automaticamente la migliore combinazione dei valori p, q e d necessari alla costruzione del modello sulla base del minor valore della metrica AIC.

Ad ogni modo, il fine ultimo è quello di valutare le differenze e comprendere quale dei due modelli si comporta meglio e per quali ragioni.

Tuttavia, per poter confrontare tra di loro i vari modelli di previsione è stato necessario stabilire un range su tutta la serie temporale da sfruttare per addestrare il modello per poi fare previsione sulla base di quel modello costruito. Solo successivamente le varie previsioni potranno essere confrontate con il reale andamento della serie. Per fare ciò è stato deciso di effettuare uno split del dataset in train e test con rispettiva percentuale di 80-20%.

Una volta aver chiarito le premesse, si è passati all'applicazione della funzione *auto_arima*, la quale è in grado di definire i parametri migliori per il modello in base al training set che gli viene dato in input. Il risultato ottenuto è stato il seguente:

- ARIMA: (0,0,1)(0,1,0)[52]

5.6 Forecasting

Terminata la fase di costruzione dei modelli si hanno in definitiva le seguenti configurazioni: **SARIMAX(2, 1, 2)(0,0,1)[52]** e **ARIMA(0,0,1)(0,1,0)[52]** adattato automaticamente, i quali sono stati confrontati e valutati le loro previsioni sulla base dell'andamento originale della serie. Per rendere possibile ciò, è stato necessario mostrare il plot simultaneo delle previsioni insieme all'andamento originale per permettere il confronto con i risultati. Il numero di step con il quale si fa progredire la previsione è stato settato a 21, corrispondente a 3 settimane, in linea con la frequenza a intervalli settimanali del dataset originale.

Il metodo utilizzato per effettuare previsione è stato *get_forecast* della libreria Statsmodels e in Fig. 51 viene mostrato il forecast con i due modelli in concomitanza all'andamento originale.

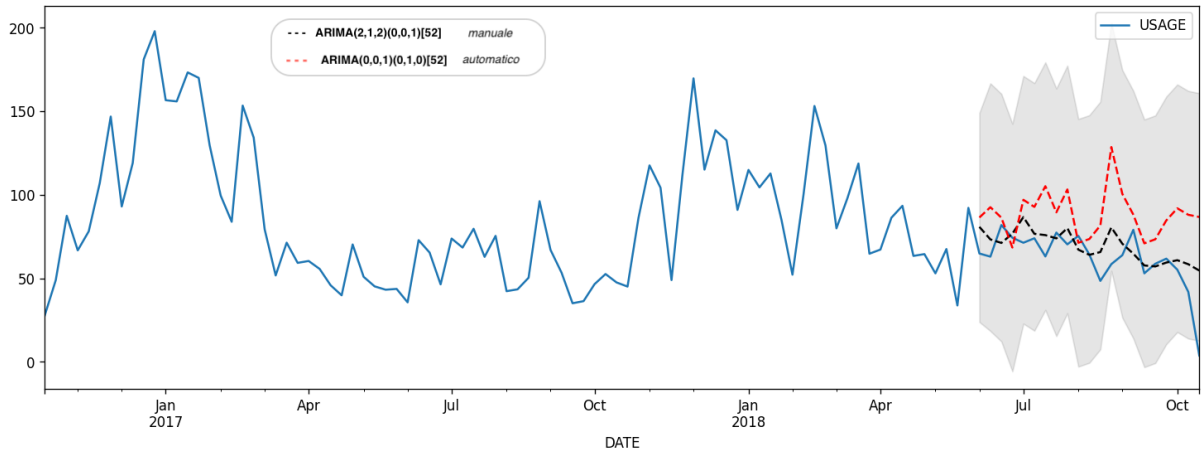

Figura 51: Forecast di modelli a confronto

Come emerge chiaramente dal grafico, il modello adattato manualmente segue molto più fedelmente il set di test effettivo. Al contrario, il modello adattato automaticamente, sembra proseguire il trend in modo molto più grossolano, a prova del fatto che si tratta di un modello più semplice e con ordini dei parametri inferiori.

Infine, sono state riportate alcune metriche statistiche per rendersi conto della bontà e previsione per entrambi i modelli (Fig. 52 e Fig. 53).

```
{'mape - manual': 0.787332467634055,
 'me - manual': 7.253522082076053,
 'mae - manual': 11.150616022078257,
 'mpe - manual': 0.7364053332282933,
 'rmse - manual': 15.467301898620182,
 'corr - manual': 0.5598483071631329,
 'minmax - manual': 0.16329465365003248}
```


Figura 52: Metriche modello manuale SARIMAX(2,1,2)(0,0,1)[52]


```
{'mape - automatic': 1.447903187585798,
 'me - automatic': 26.51572614058215,
 'mae - automatic': 27.47181578900368,
 'mpe - automatic': 1.4351177824814496,
 'rmse - automatic': 34.075504806507354,
 'corr - automatic': -0.000852864487451673,
 'minmax - automatic': 0.29640660819885734}
```


Figura 53: Metriche modello automatico ARIMA(0,0,1)(0,1,0)[52]

Si può notare come i risultati ottenuti del modello manuale siano molto bassi, a conferma della bontà del modello.