Università Politecnica delle Marche

Facoltà di Ingegneria

Dipartimento di Ingegneria dell'Informazione

Studenti:

Bedetta Alessandro
Ascani Christian

Docenti:

Ursino Domenico

A.A. 2022-2023

Sommario

1. Introduzione

La relazione che segue illustra il progetto da noi svolto nell'ambito della clusterizzazione/classificazione e dell'analisi di serie temporali.

Il linguaggio di programmazione utilizzato nel corso di tutto lo sviluppo è Python, il quale negli anni si è stabilito come standard per quanto riguarda la data science; ciò è dovuto al fatto che esso offre innumerevoli librerie (come Pandas, SKLearn e StatsModel) le quali rappresentano lo stato dell'arte e sono accessibili gratuitamente.

Nel progetto è stato fatto ampio uso di questi tool.

2. Datasets

Dopo aver cercato a lungo un dataset su cui basare lo svolgimento del progetto, siamo giunti alla conclusione che, data la forte diversità fra i task da risolvere, un solo dataset non sarebbe stato sufficiente. Per questo motivo abbiamosi è deciso di usarne due, uno per l'analisi di serie temporali e uno per classificazione/clustering.

I dataset in questione sono i seguenti:

https://www.kaggle.com/datasets/varpit94/amazon-stock-data per analisi legate alle serie temporali.

https://www.kaggle.com/datasets/kidoen/bank-customers-data per la clusterizzazione e la classificazione.

Il primo dei due è chiamato "*Amazon Stock Data*" e si tratta di un dataset contenente l'andamento in borsa dei titoli di Amazon a partire dal 15 maggio 1997 fino ad oggi.

Nel dettaglio le colonne componenti il dataset sono le seguenti sette.

Date: Data di riferimento in formato "yyyy-mm-dd".
Open: Prezzo della prima transizione del giorno.
High: Prezzo massimo nel giorno di riferimento.
Low: Prezzo minimo nel giorno di riferimento.
Close: Prezzo dell'ultima transizione del giorno.
Adj Close: Prezzo di chiusura aggiustato per riflettere eventuali azioni fatte dalla corporate.
Volume: Numero di unità scambiate nella giornata.

La natura di questo dataset lo rende perfetto per operare analisi di serie temporali, in quanto la maggior parte dei campi sono rilevamenti fatti con cadenza di tempo pressoché costante (i fine settimana non sono presenti nei giorni).

Il secondo dataset è chiamato "Bank Customers Data" e contiene una serie di dati relativi ai clienti di una banca portoghese. Questi dati sono stati

raccolti durante una campagna di marketing diretta. Lo scopo alla base del dataset è quello di riuscire a capire, in base al profilo di un cliente, se quest'ultimo potrebbe affidare alla banca un deposito di lungo termine o meno.

Come nel caso precedente il dataset è in formato di file .csv e le colonne al suo interno sono le seguenti.

Age: Campo intero che indica l'età della persona.
Job: Campo stringa contenente l'occupazione della persona.
Marital: Campo stringa indicante lo stato maritale della persona.
Education: Stringa contente il grado di istruzione raggiunto dalla persona
Default: Campo booleano che indica se il conto della persona è in default
Balance : Campo numerico che indica il credito bancario del cliente
Hougsing Loan: Campo booleano il quale indica se la persona ha ricevuto un prestito per l'acquisto di un immobile.
Loan: Variabile booleana che indica se o meno la persona ha ricevuto un prestito di ogni genere.
Contact: Stringa che indica il tipo di recapito per il contatto con la persona.
Day: Giorno del contatto (sondaggio) con il cliente.
Month: Mese del contatto (sondaggio) con il cliente.
Duration: Durata del contatto (sondaggio) con il cliente (in secondi).
Campaign: Numero di tentativi fatti per contattare il cliente ai fini del sondaggio nella campagna attuale.
Pdays: Numero di giorni passati dall'ultimo contatto con il cliente.
Previous: Numero di chiamate effettuate al cliente prima della campagna attuale.
Poutcome: Risultato dell'ultima campagna pubblicitaria (successo o fallimento).

3. Serie Temporali

La prima analisi da noi svolta è stata quella relativa alle serie temporali, basata come detto in precedenza sul dataset "*Amazon Stock Data*". Lo scopo generale dietro queste analisi è stato quello di riuscire ad osservare l'andamento di queste serie di misurazioni e fare previsioni su di esse dopo una fase iniziale di pre-processing.

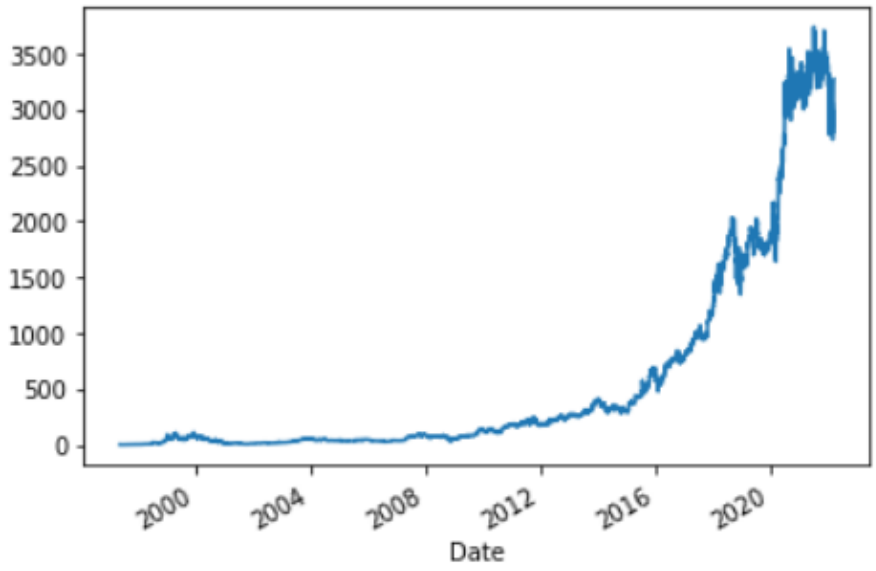


Figura 1: Serie originale

3.1 Processing Iniziale

Inizialmente, utilizzando il metodo predefinito di Pandas *to_datetime*, il campo "Date" è stato convertito in formato datetime, più facile da manipolare. In aggiunta, è stato trasformato il riferimento temporale nell'indice del dataset tramite la funzione *set_index*. Successivamente, è stata considerata solamente la colonna "Open" poiché l'analisi si concentra sul prezzo di apertura del titolo azionario; comunque, la differenza tra il prezzo di apertura e quello di chiusura non è sufficientemente importante da giustificare l'utilizzo di entrambi i campi; nella Figura 2 si può verificare che il delta tra le curve sia limitato.

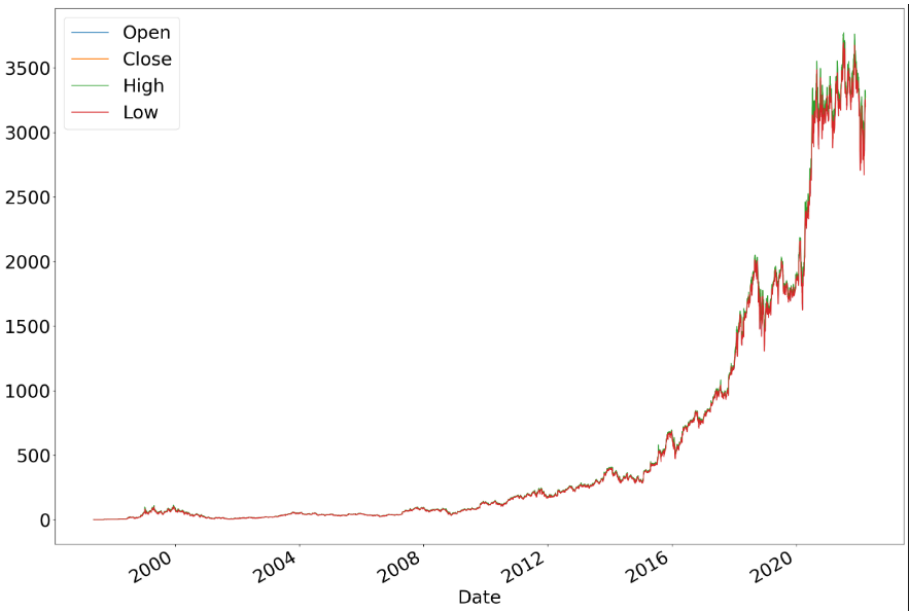


Figura 2: Comparazione delle colonne del dataset

Dall'analisi delle colonne si può notare che i giorni di apertura del mercato azionario sono solitamente cinque (dal lunedì al venerdì), escluse alcune festività (1° gennaio, 4 luglio, ecc).

Data la grande mole di osservazioni, abbiamo deciso di restringere il dataset all'intervallo di date a partire dal 2 Gennaio 2020 al 24 Marzo 2022; questa scelta è dovuta al fatto che le nostre analisi riguardano l'andamento futuro del mercato azionario e di conseguenza avere dati troppo arretrati rischia solo di inquinare le previsioni, alla luce anche dei fatti più recenti e della situazione economica meno tradizionale.

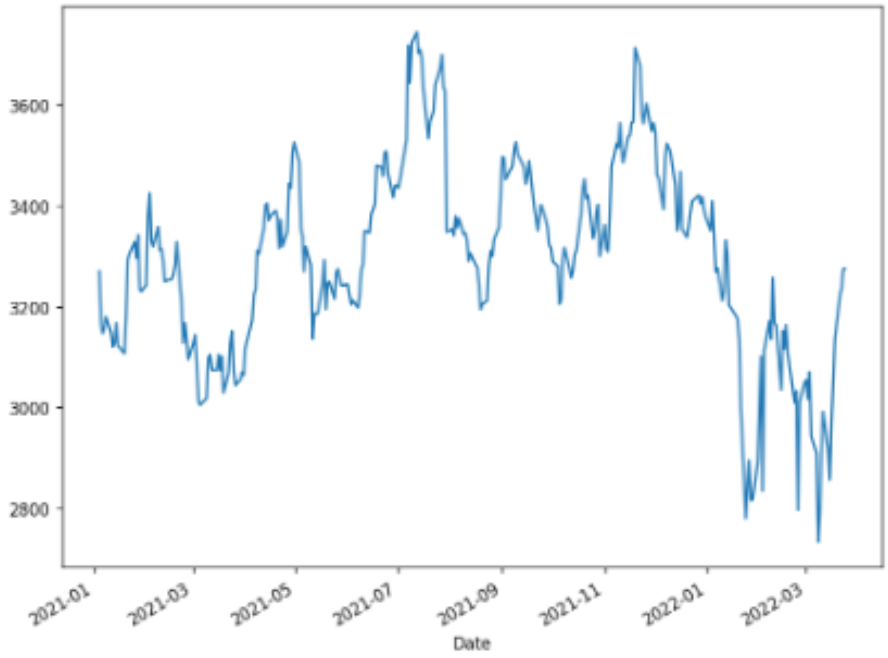


Figura 3: Serie estratta

Sono state eseguite solamente le operazioni riportate di filtraggio e di trasformazione dei dati perché il dataset di partenza era già di per sé pulito e pronto all'uso (il relativo coefficiente di usability riportato da Kaggle è 10). Nella Figura 4, si riporta un campione estratto dalla colonna "Open".

Date	
2020-01-02	1875.000000
2020-01-03	1864.500000
2020-01-06	1860.000000
2020-01-07	1904.500000
2020-01-08	1898.040039
2020-01-09	1909.890015
2020-01-10	1905.369995

Figura 4: Esempio del dataset

3.2 Studio delle caratteristiche della serie

Per prima cosa, si è proceduto con la decomposizione della serie in modo da capire l'andamento delle tre componenti principali: trend, stagionalità e residui. Questo ha permesso di visualizzare le caratteristiche isolate della serie, sia attraverso la decomposizione additiva (Figura 5) sia con quella moltiplicativa (Figura 6). Riguardo ai residui, quelli relativi alla decomposizione additiva sono meno regolari, per cui questa è preferibile a quella moltiplicativa.

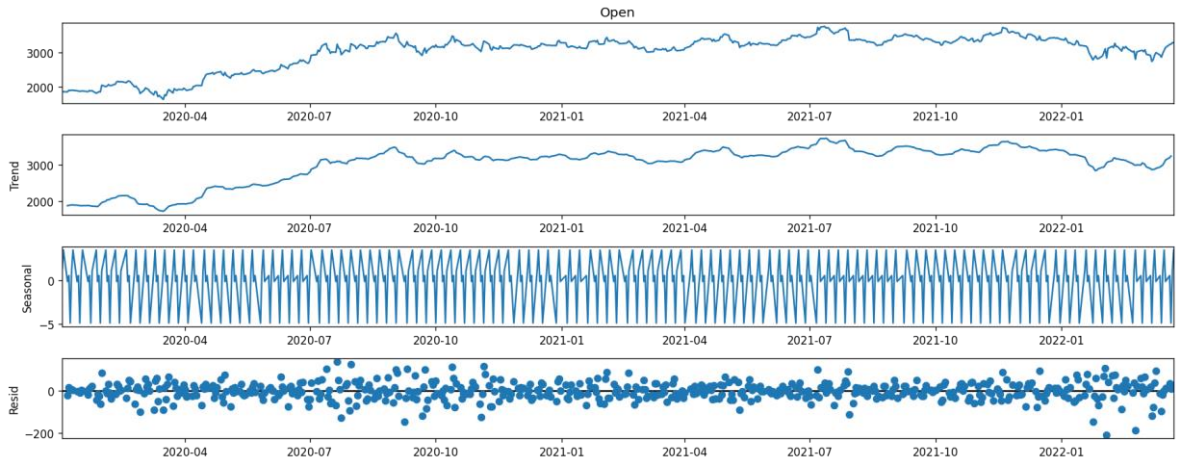


Figura 5: Decomposizione additiva

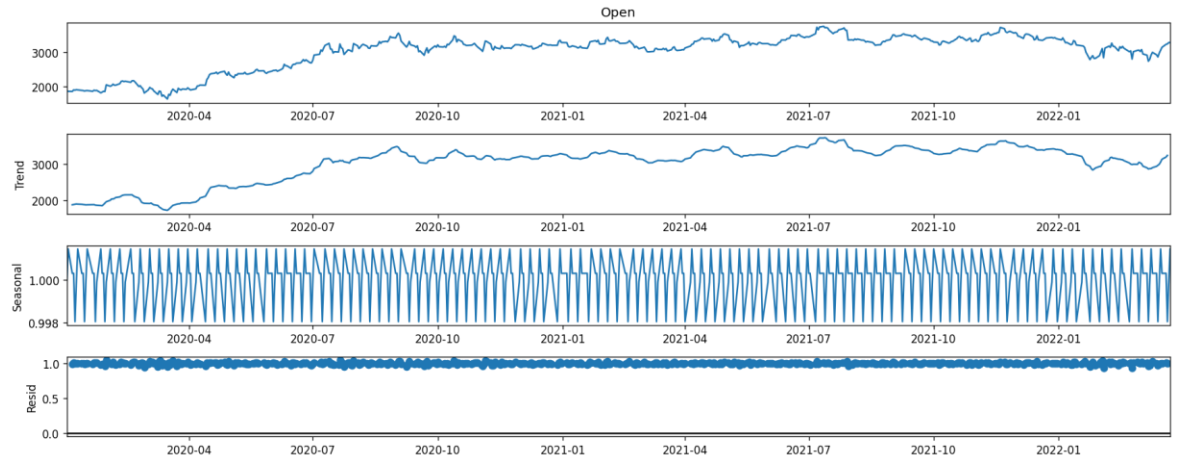


Figura 6: Decomposizione moltiplicativa

Avendo fatto la decomposizione, si notano in maniera chiara sia una tendenza verso i valori più alti sia una certa stagionalità. Focalizzandoci sul trend, si può eseguire uno studio più approfondito tramite un'operazione di detrendizzazione, così da visualizzare la serie priva della componente più evidente. Infatti, grazie alla funzione detrend, messa a disposizione dalla libreria "scipy", la serie "detrendizzata" (Figura 7) mostra che, anche non essendo uguali in valore assoluto, il prezzo nell'aprile 2020 e quello nel marzo 2022 indicano una situazione economica simile: infatti, nel primo caso la flessione era dovuta all'impatto iniziale della pandemia, mentre nel secondo caso il comportamento è riconducibile sia alla guerra sia al rialzo dei tassi di interesse.

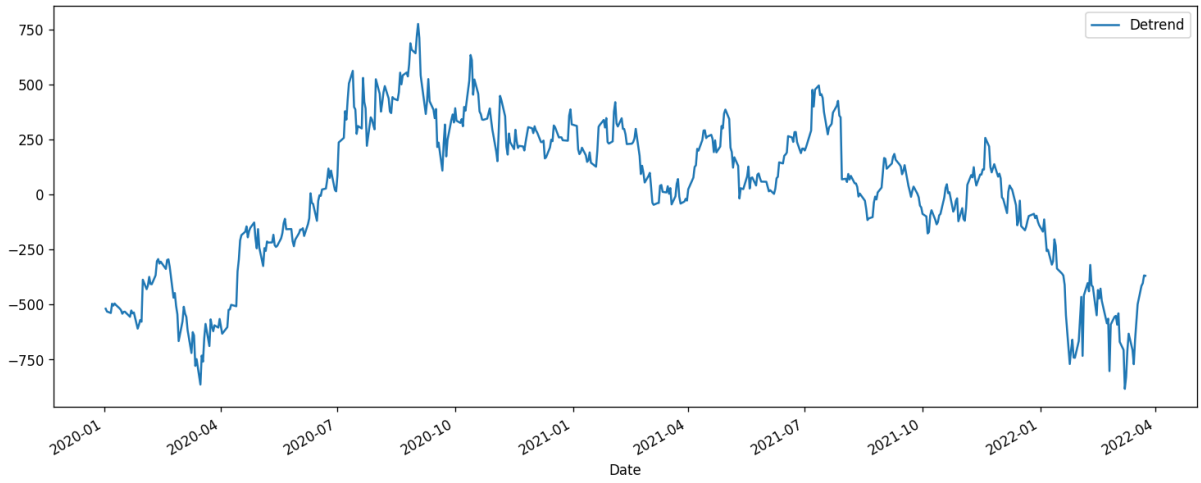


Figura 7: Serie senza trend

3.3 Analisi della stazionarietà e stima dei parametri

Dato che l'analisi di una serie temporale risulta essere più semplice se questa è stazionaria, è necessario procedere con la verifica di tale proprietà. Infatti, in una serie stazionaria i valori non dipendono dal tempo e, quindi, media e varianza risultano essere costanti. Al fine di ciò, viene utilizzato l'Augmented Dickey Fuller test, in cui si considera la serie non stazionaria (ipotesi nulla) e viene calcolato il p-value per confermare o meno tale ipotesi:

se il p-value > 0.05 la serie è non stazionaria, altrimenti l'ipotesi nulla viene rifiutata e si asserisce che la serie sia stazionaria.

Nel caso della serie integrale, il test mostra che l'ADF statistics è pari a -2.343, mentre che il p-value è pari a 0.158, per cui si può affermare che la serie è non stazionaria.

Dunque, viene eseguita una differenziazione della serie, di primo e di secondo ordine. Fatto ciò, si esegue l'ADF test su queste ultime due serie ricavate, ottenendo:

- serie differenziata I ordine: ADF statistics: -25.371, p-value: 0.0;
- serie differenziata II ordine: ADF statistics: -11.750, p-value; $1.2*10^{-21}$;

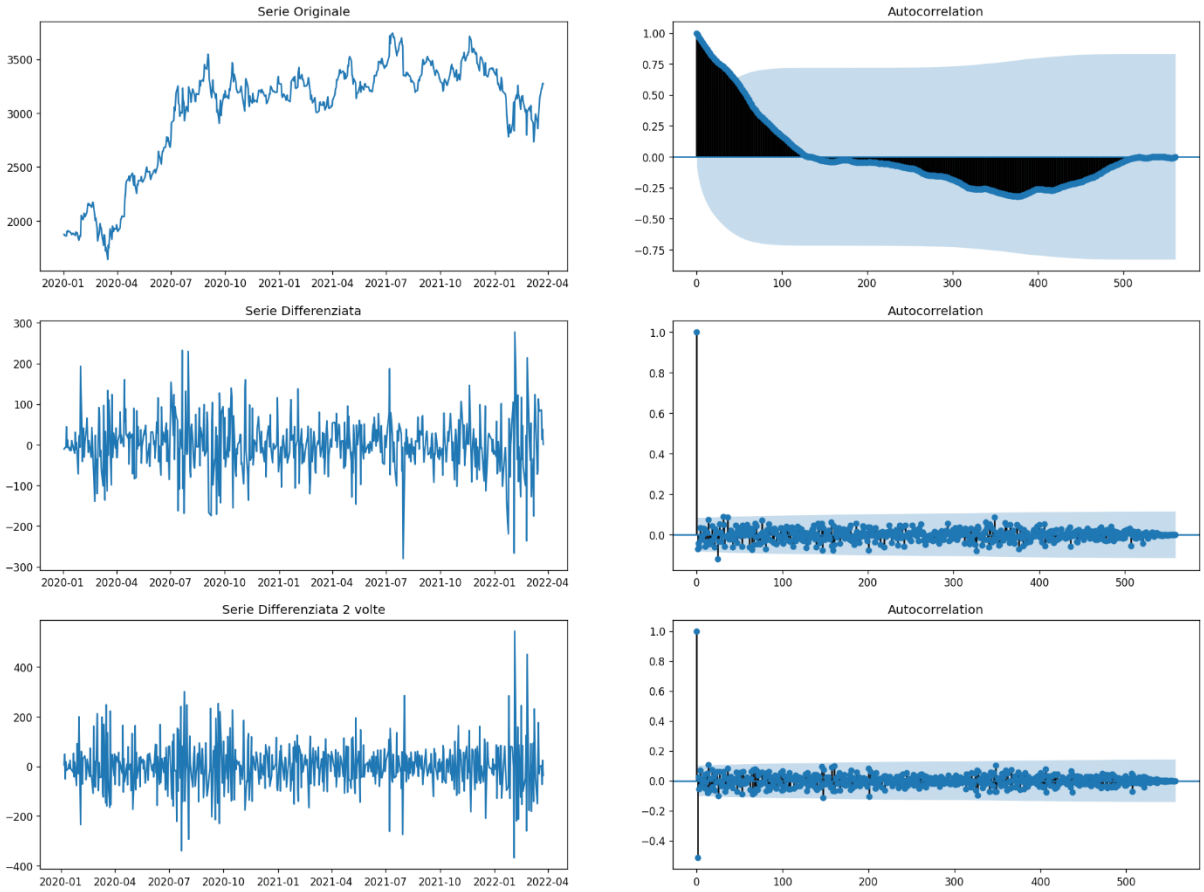


Figura 8: Augmented Dickey Fuller test

Da questi test si può considerare la differenziazione del I ordine, in quanto ulteriori differenziazioni potrebbero causare la perdita di caratteristiche importanti della serie, snaturandola.

Per la predizione di una serie temporale attraverso i suoi valori passati, si vogliono testare approcci differenti: prima si vuole creare un modello ARIMA e poi si vuole aggiungere la componente stagionale, in modo da ottenere un modello SARIMA. Generalmente, un modello ARIMA è caratterizzato da tre

parametri, ovvero **p** (Auto Regressive), **q** (Moving Average) e **d** (Integrated), i quali possono essere calcolati tramite alcuni test, come l'ADF test.

Avendo già eseguito questo test, si può asserire che $d = 1$, in quanto il p-value corrispondente alla serie differenziata una sola volta è minore di 0.05.

Per valutare p e q, si devono considerare rispettivamente l'autocorrelazione parziale e l'autocorrelazione. Dal grafico del PACF (Partial Auto Correlation Function), si può notare che solamente il *lag(1)* è sopra il livello di significatività, per cui si fissa $p = 1$. In modo simile, dal grafico dell'ACF (Auto Correlation Function), si ricava che è necessario un solo un termine per rimuovere ogni autocorrelazione nella serie stazionaria, perciò $q = 1$.

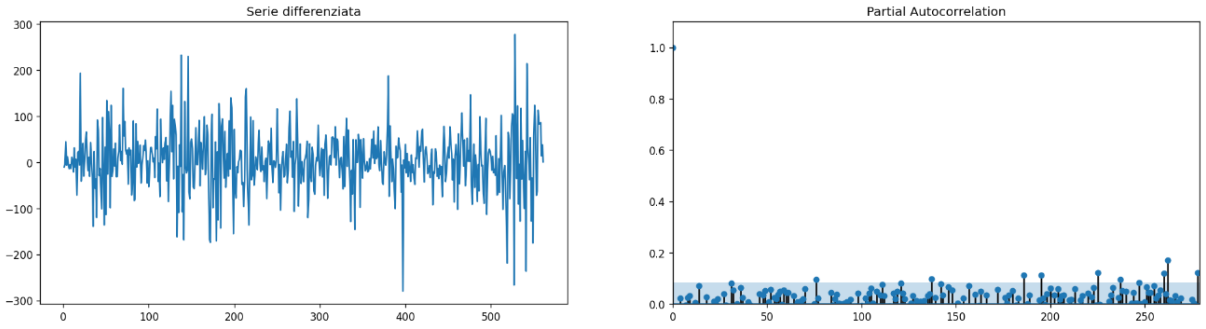


Figura 9: Autocorrelazione parziale

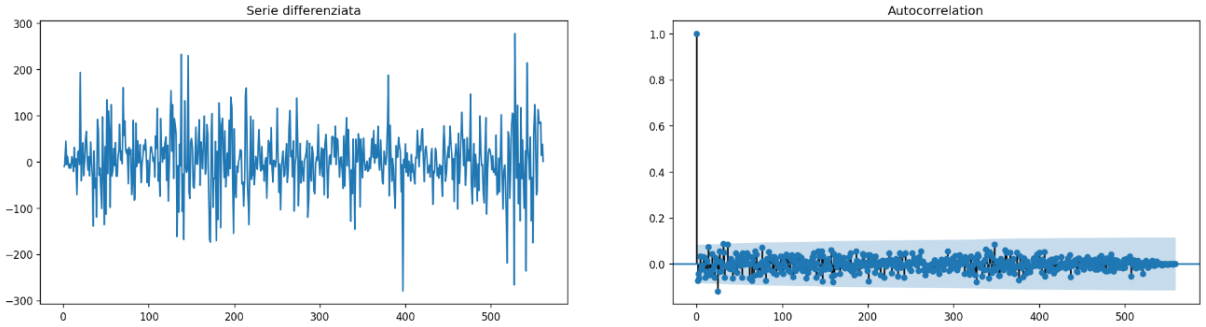


Figura 10: Correlazione parziale

3.4 Modello ARIMA

Dato che sono stati valutati i parametri caratteristici ($p = 1, d = 1, q = 1$), si procede a creare e ad addestrare il modello ARIMA.

ARIMA Model Results

Dep. Variable:	D.Open	No. Observations:	561
Model:	ARIMA(1, 1, 1)	Log Likelihood	-3143.404
Method:	css-mle	S.D. of innovations	65.645
Date:	Fri, 02 Dec 2022	AIC	6294.808
Time:	21:56:40	BIC	6312.127
Sample:	1	HQIC	6301.570

	coef	std err	z	P>|z|	[0.025	0.975]
const	2.4650	2.223	1.109	0.267	-1.892	6.822
ar.L1.D.Open	0.5978	0.301	1.986	0.047	0.008	1.188
ma.L1.D.Open	-0.6779	0.277	-2.446	0.014	-1.221	-0.135

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.6727	+0.0000j	1.6727	0.0000
MA.1	1.4751	+0.0000j	1.4751	0.0000

Figura 11: Modello ARIMA

La prima predizione è stata calcolata tramite il metodo *predict()*, il cui attributo (*dynamic*) che può essere impostato a *True* (ad ogni passo viene eseguita una predizione aggiungendo il valore predetto nel passo precedente, riadattando il modello) o *False* (predizione sequenziale a partire dal valore vero precedente, per cui la predizione è traslata di un passo). I risultati della predizione non sono stati soddisfacenti, come si può anche vedere con *dynamic = False* (Figura 12).

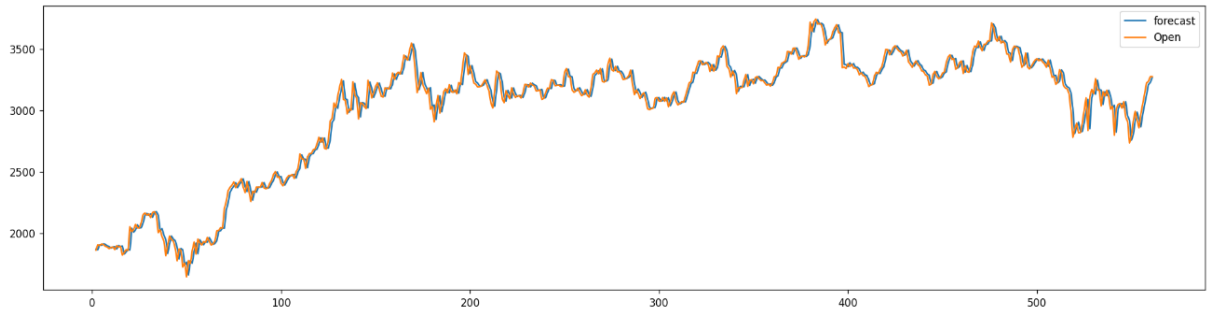


Figura 12: Predizione con dynamic=False

È stata eseguita anche una *cross validation*, dividendo il dataset in training set (90%) e testing set (10%), così da verificare la bontà della previsione; tuttavia, con un intervallo di confidenza pari al 95% la previsione non riflette l'andamento effettivo (Figura 13).

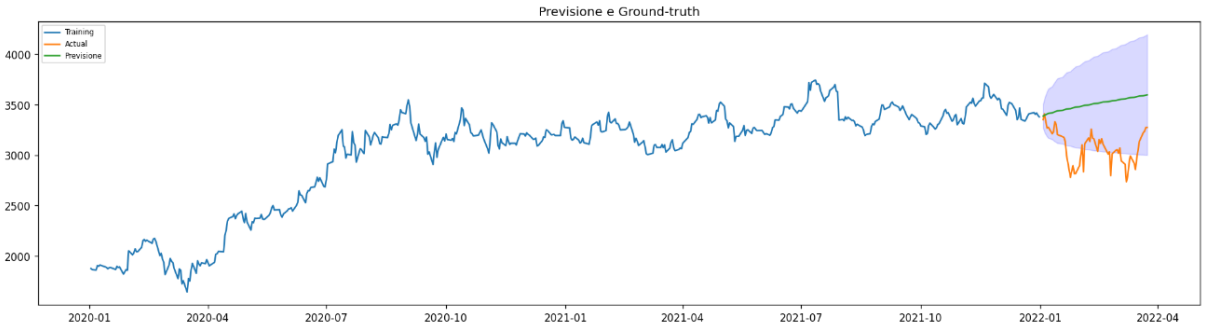


Figura 13: Predizione con cross validation

Si è proceduto con lo studio dei residui (Figura 14), in modo da notare eventuali irregolarità o pattern. Si può notare che questi sono contenuti e la loro media è vicino allo 0, con una bassa varianza, delineando una distribuzione molto simile a quella normale.

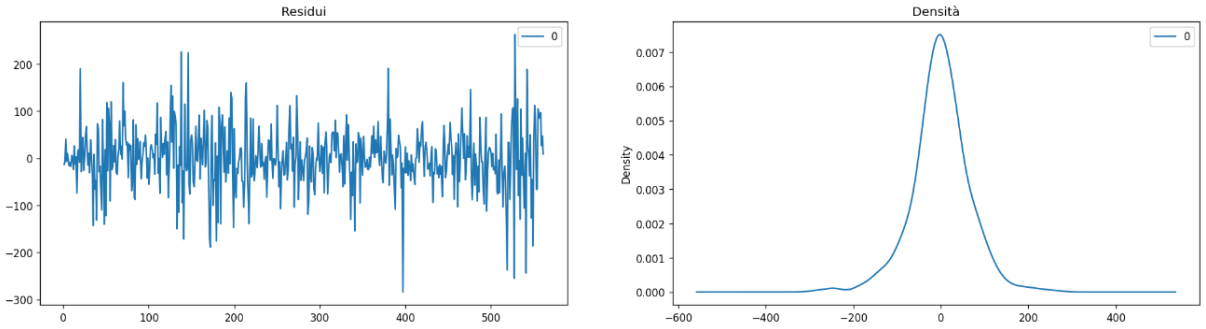


Figura 14: Residui e distribuzione

3.5 Modello SARIMA

Un'estensione del modello ARIMA è data dal modello SARIMA, che riprende l'ARIMA e comprende anche una componente stagionale. Infatti, in molti casi è possibile individuare una certa stagionalità. Attraverso i grafici dell'autocorrelazione e della decomposizione, si può notare una leggera stagionalità ogni 5 lag (equivalenti a 5 giorni). A partire dagli stessi parametri (p, d, q) del modello ARIMA, si aggiungono altri quattro parametri (P, D, Q, m): il parametro **m** indica la stagionalità della serie, i restanti si riferiscono alla componente stagionale. La costruzione e l'addestramento del modello vengono effettuati come in precedenza, attuando la medesima divisione del dataset in training e testing set.

Si considera la configurazione (1,1,1)x(1,3,4,5) e si può osservare in (Figura 15) che la previsione segue il trend; infatti, inizialmente si prevede il primo *bottom*, poi la curva si stabilizza, senza predirre il picco finale.

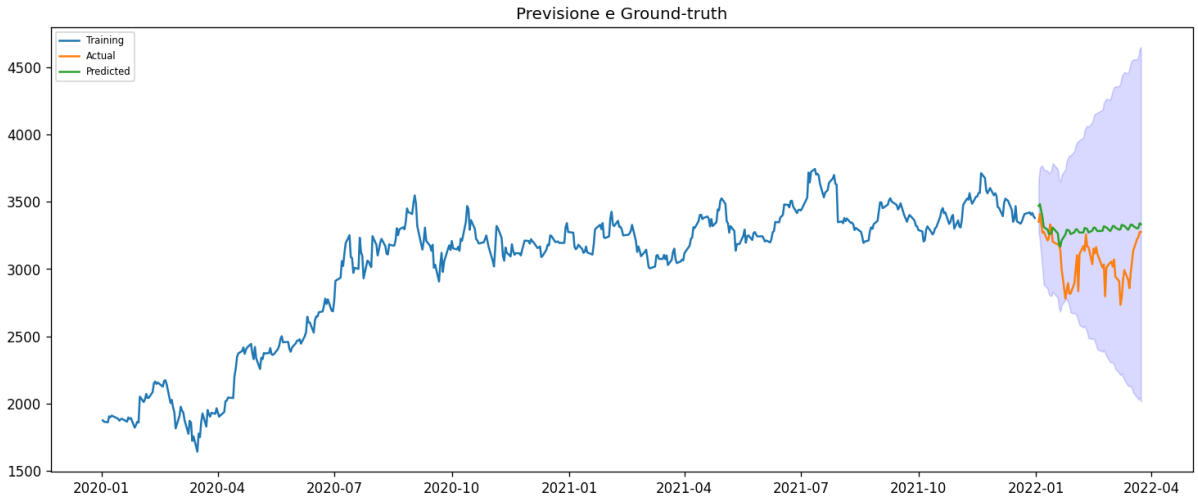


Figura 15: Predizione con SARIMA (1,1,1)x(1,3,5,5)

Successivamente, si può considerare una differente configurazione del modello SARIMA, ovvero (1,1,1)x(10,3,7,5). In questo caso (Figura 16), si nota che, a causa dei termini MA, c'è una fedeltà maggiore verso i movimenti della curva effettiva (picchi più evidenti). D'altra parte, c'è una traslazione dei ribassi e dei rialzi, anche se questi seguono l'andamento della curva, la quale risulta essere compresa nell'intervallo di confidenza (95%).

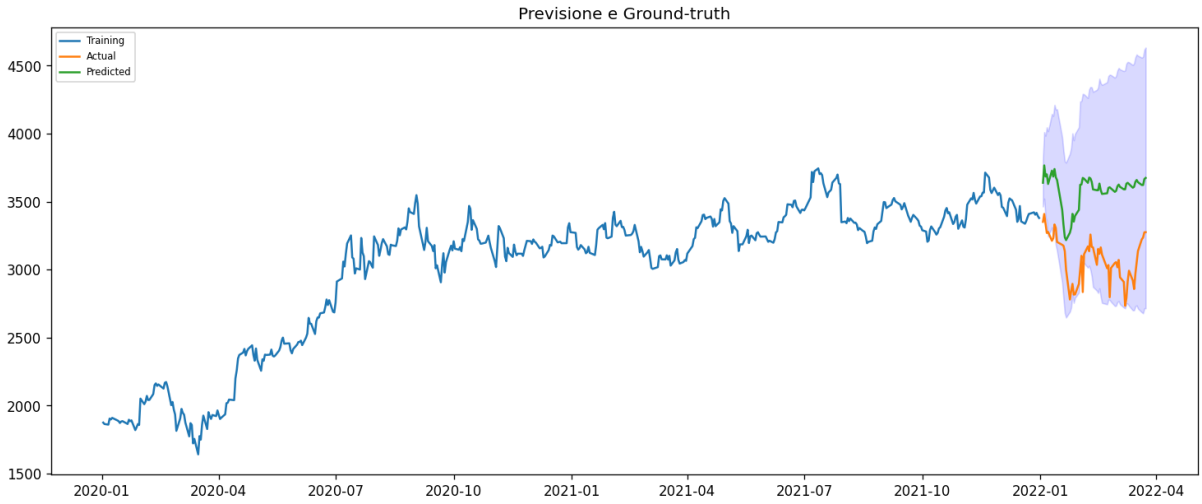


Figura 16: Predizione con SARIMA(1,1,1)x(0,3,7,5)

3.6 Residui e metriche di valutazione

Ottenuti i modelli, si può procedere con l'analisi dei residui. Infatti, è possibile individuare eventuali correlazioni o se sono presenti informazioni che potrebbero essere utili per la previsione. Per il primo modello SARIMA (Figura 17), si nota che i residui si stabilizzano, come dimostrato anche dalla distribuzione leggermente traslata rispetto a quella normale. Tuttavia, si può notare dal correlogramma che sono presenti correlazioni tra i primi residui.

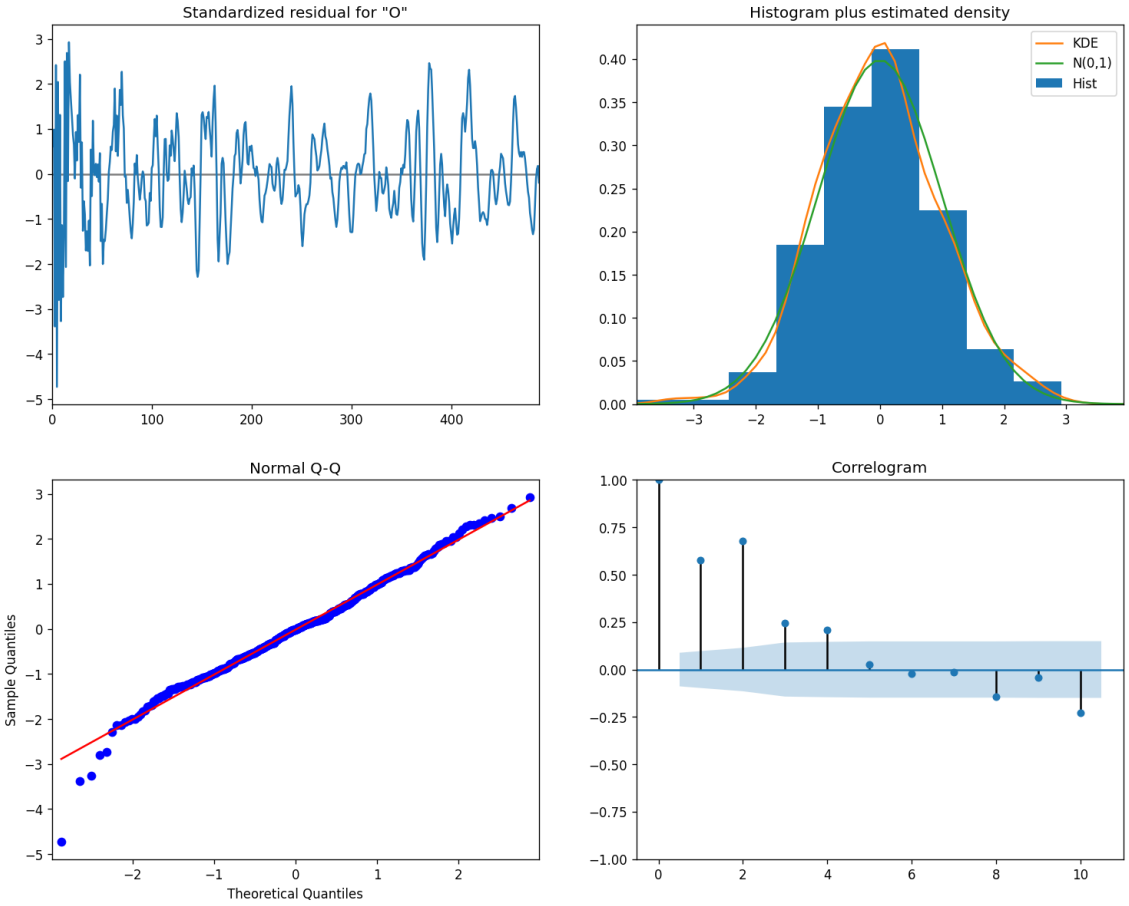


Figura 17: Analisi residui SARIMA(1,1,1)x(1,3,4,5)

Riguardo al secondo modello SARIMA, si può notare una situazione molto simile alla precedente (Figura 18). A differenza, però, della configurazione (1,1,1)x(1,3,4,5), in questo caso il correlogramma non mostra particolari anomalie e la distribuzione è molto più simile ad una Gaussiana.

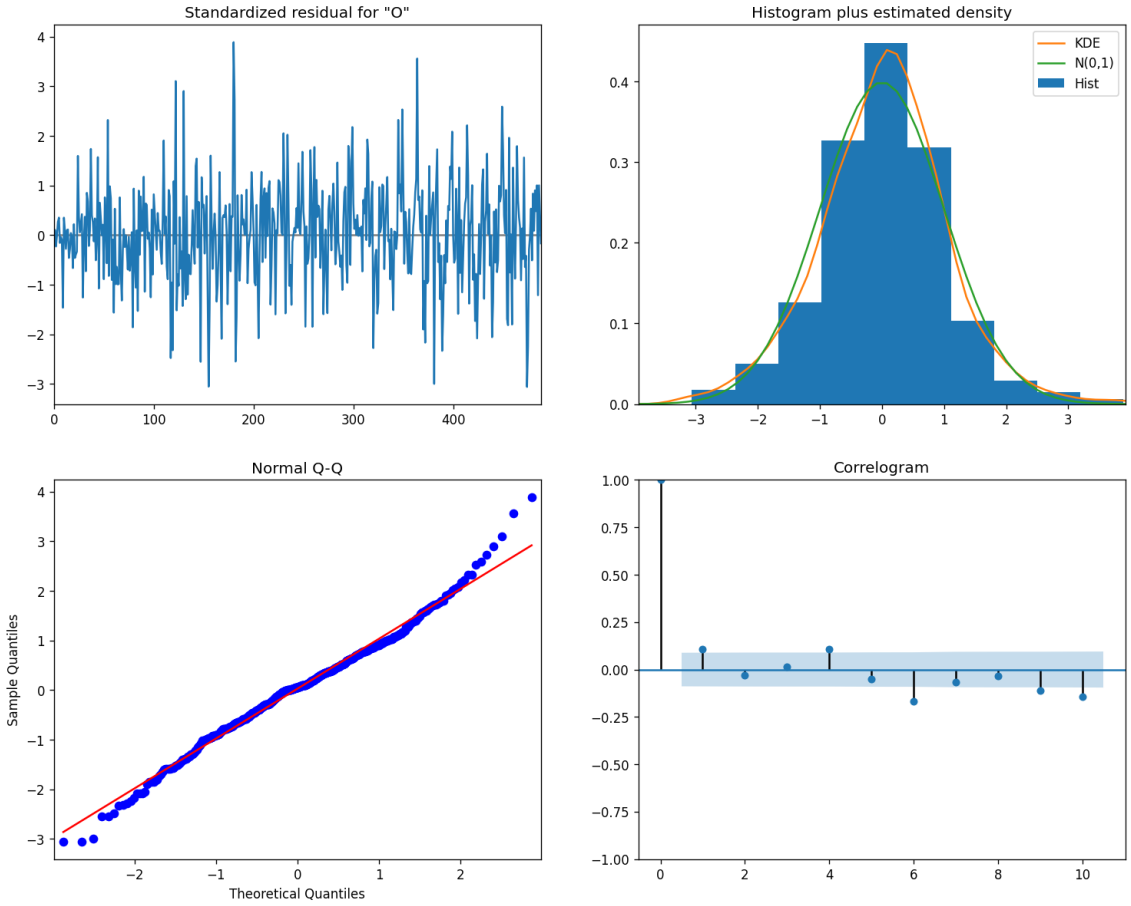


Figura 18: Analisi residui SARIMA(1,1,1)x(0,3,7,5)

Per misurare la bontà della previsione, sono state utilizzate alcune metriche, ovvero il MAPE (Mean Absolute Percentage Error), il ME (Mean Error), il MAE (Mean Absolute Error), l'MPE (Mean Percentage Error), l'RMSE (Root Mean Squared Error), l'ACF (autocorrelazione dell'errore per il lag(*1*)), la correlazione tra la serie e la previsione, il MINMAX error.

Considerando il modello SARIMA(1,1,1)x(1,3,4,5), come si può osservare in Figura 19, si può riscontrare che il MAPE è pari a 0.077, l'RMSE è uguale 277.5, mentre la CORR si attesta a 0.31.

```
'mape': 0.07797024499053874,
'me': 227.01054481909944,
'mae': 230.80803057015663,
'mpe': 0.07682720749149671,
'rmse': 277.4973658586348,
'corr': 0.31332432850601427,
'acf1': 0.7676403166664119,
'minmax': 0.06992885469805399
```

Figura 19: SARIMA (1,1,1)x(1,3,5,5)

Invece, riguardo al SARIMA (1,1,1)x(0,3,7,5), si hanno i valori di MAPE e di RMSE rispettivamente pari a 0.167 e a 532.2, mentre la CORR è uguale a 0.451.

```
'mape': 0.16788869219540933,
'me': 508.3705594456423,
'mae': 508.3705594456423,
'mpe': 0.16788869219540933,
'rmse': 532.2024485574955,
'corr': 0.45098910329983416,
'acf1': 0.7248844300916056,
'minmax': 0.14155438300875933
```


Figura 20: SARIMA(1,1,1)x(0,3,7,5)

Dunque, dalle metriche di valutazione si può asserire che il primo modello risulta essere più preciso in generale, mentre nel secondo modello c'è una correlazione maggiore tra la serie originale e la previsione.

In conclusione, dall'analisi di questa serie temporale si è ottenuto che, come supponibile, prevedere l'andamento di un titolo azionario in borsa è altamente complicato e che la conseguente previsione risulta essere poco precisa. Infatti, soprattutto nell'ultimo anno, prima a causa della pandemia poi a causa della guerra, ci sono stati stravolgimenti nello scenario economico globale. Di conseguenza, c'è una maggiore volatilità e, per la presenza di fattori esogeni, si potrebbe pensare di raffinare un'analisi di questo tipo con un modello di tipo SARIMAX. Ad esempio, si potrebbero considerare altri indici economici, come i tassi di interesse, l'oro, il petrolio, l'inflazione, il VIX.

4. Classificazione e Clustering

4.1 Analisi descrittiva

Il primo step in questa fase del progetto ha riguardato lo svolgimento di analisi descrittive, ovvero tutte quelle analisi atte ad illustrare la natura del dataset, mostrandone la composizione e evidenziando eventuali correlazioni.

Ci siamo, quindi, concentrati inizialmente su questa fase di Data Understanding, andando ad esaminare il dataset sotto vari punti di vista. Di seguito verranno riportate alcune delle analisi eseguite al fine di avere una migliore visione del dataset.

La prima è stata un'analisi relativa alla durata media delle chiamate pubblicitarie su base tempo (divisione per mesi). La colonna “duration” rappresenta un'informazione di fondamentale importanza, in quanto, nella stragrande maggioranza dei casi, se un cliente rimane al telefono in ascolto di un messaggio pubblicitario implica che c’è da parte sua un certo interesse verso il servizio offerto. Il calcolo della durata media ha mostrato che nei mesi precedenti l’inverno si ha un leggero innalzamento di quest’ultimo valore, con un picco in corrispondenza di dicembre; ciò potrebbe essere spiegato considerando che il periodo intorno al Natale è da sempre caratterizzato da un aumento della spesa media (Figura 21).

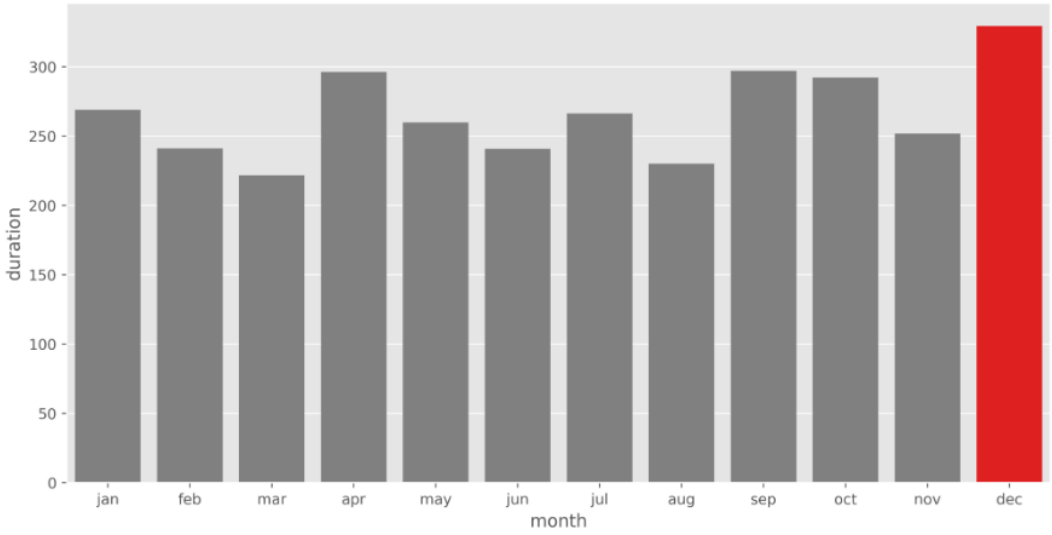


Figura 21: Durata media per mese

Un secondo gruppo di analisi ha riguardato le caratteristiche generali delle persone interessate dalla campagna pubblicitaria. Sono riportati dei pie-chart che rappresentano diverse caratteristiche dei clienti, come ad esempio settore di lavoro, fascia di età, stato maritale e livello di educazione. Questa caratterizzazione dei record presenti nel dataset consente di avere una visione più chiara delle persone interessate dal sondaggio, e permette di capire quali sono i campi più rilevanti al fine di portare avanti la classificazione e la segmentazione.

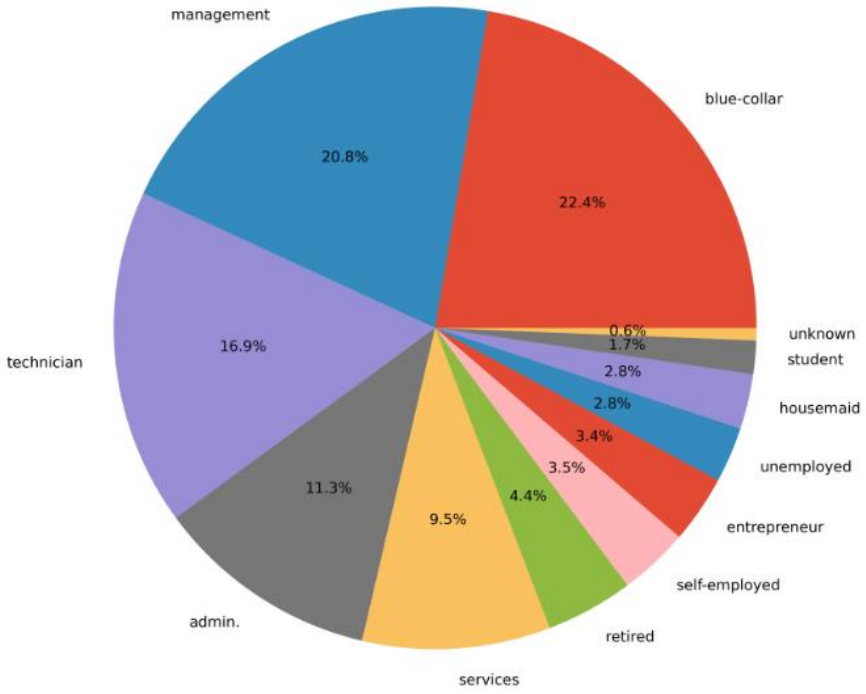


Figura 22: Distribuzione delle occupazioni

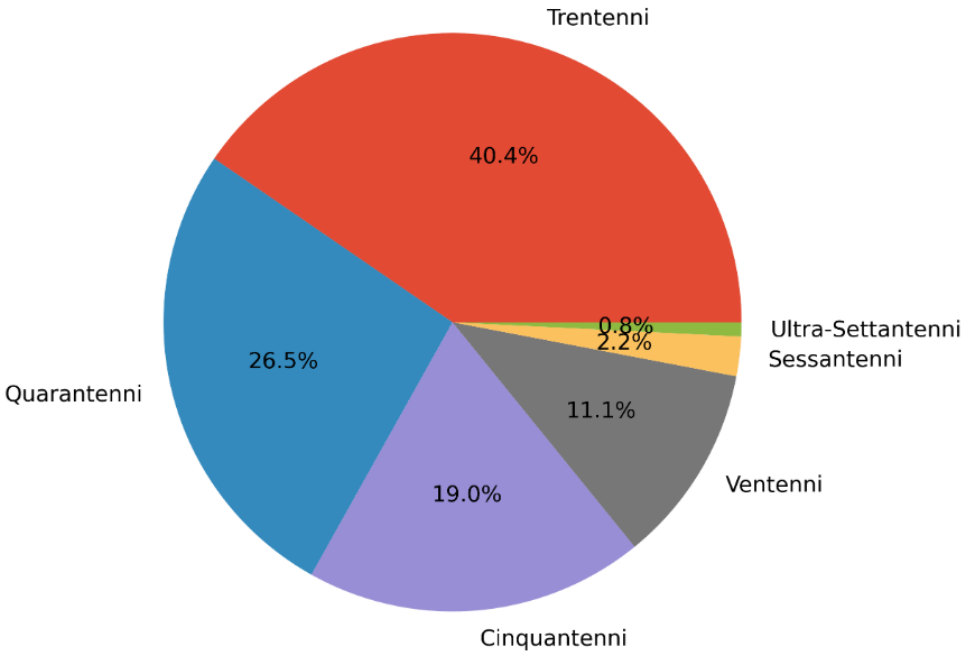


Figura 23: Distribuzione delle fasce di età

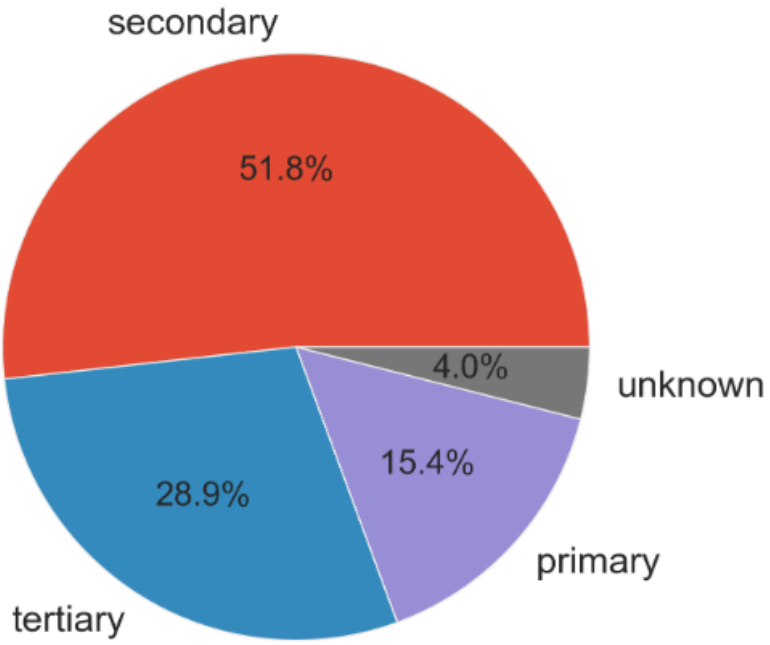


Figura 24: Distribuzione del livello di educazione

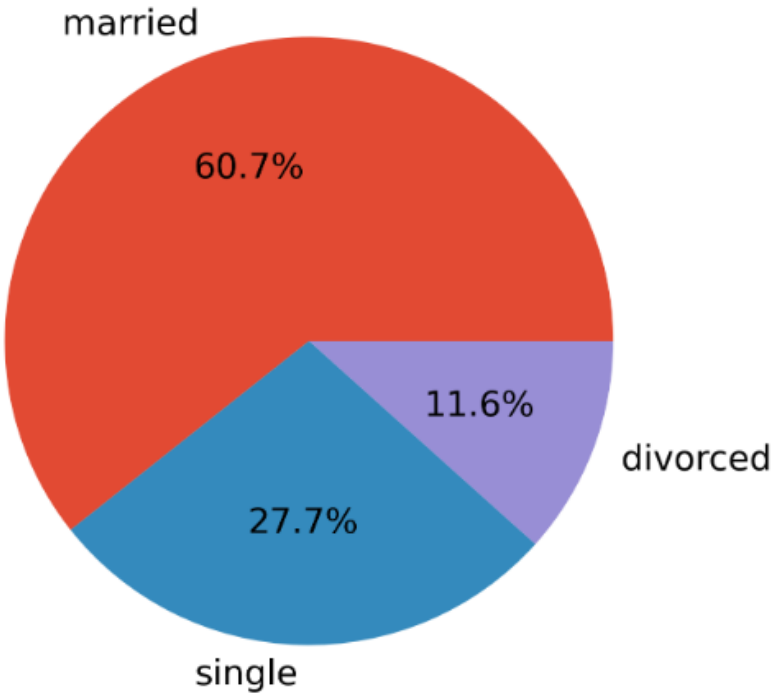


Figura 25: Distribuzione degli stati maritali

Un ultimo esempio che può risultare utile alla comprensione del dataset, riguarda l'analisi svolta in merito all'ammontare dei bilanci dei clienti. Di fatti, in questa fase il focus è stato quello di osservare l'ammontare del credito bancario medio al variare di alcune caratteristiche delle persone interessate. Il dato più sorprendente emerso da tale analisi ha riguardato le persone di età più avanzata. Infatti, sembra che i clienti di terza età siano quelli che presentano un bilancio medio più alto di tutte le altre categorie;

ciò si può evincere dai grafici a barre riportati, dove è chiaro come le persone con età compresa fra i 71 e gli 80 anni siano quelle aventi conti in banca più ricchi (Figura 26); tale comportamento si conferma osservando il bilancio medio per occupazione (Figura 27), dove può essere riscontrato che le persone ormai ritirate dal mondo del lavoro (spesso pensionati) abbiano una media più alta di tutte le altre classi.

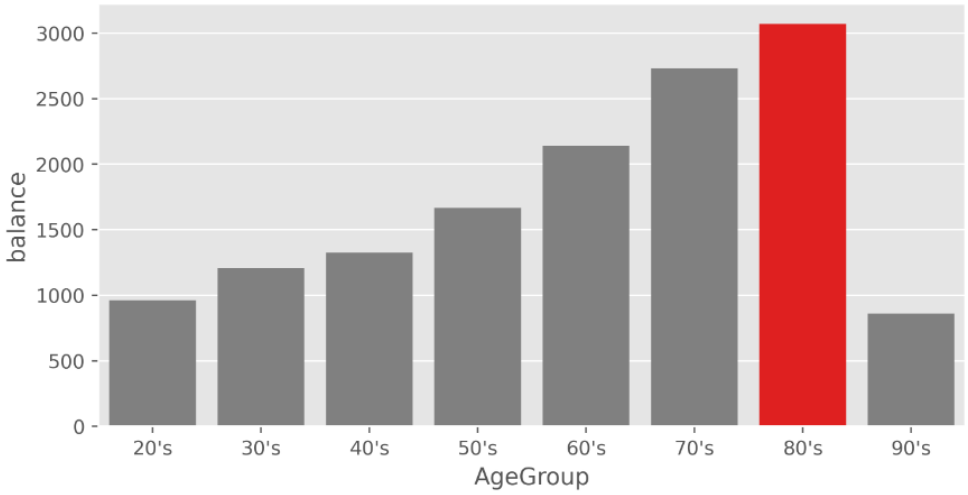


Figura 26: Bilancio medio per fascia di età

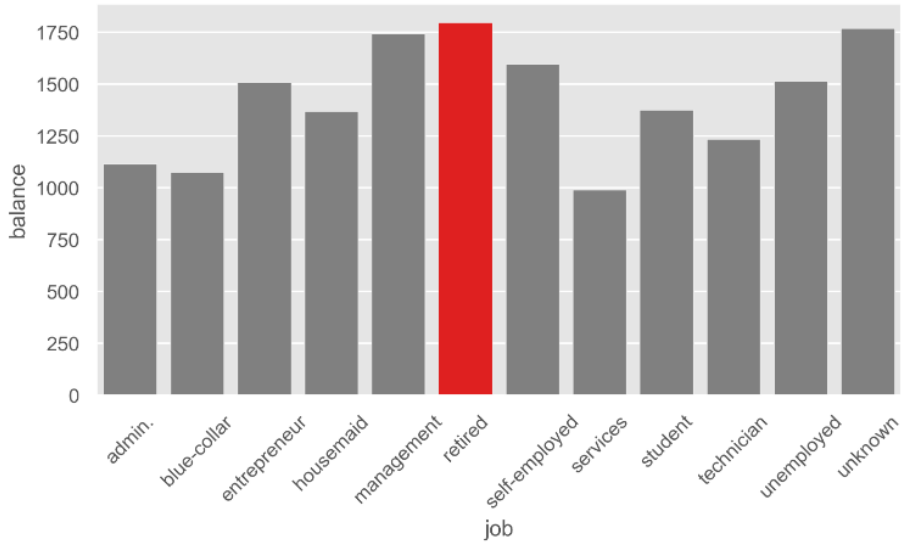


Figura 27: Bilancio medio per occupazione

4.2 Classificazione

Il principale scopo della classificazione è quello di assegnare una categoria ad un insieme di elementi dati in input. Si tratta di un task che, in base alle tecniche usate, può ricadere nel supervised o nell'unsupervised learning.

Gli algoritmi (riportati di seguito) utilizzati ricadono tutti nella categoria del supervised learning:

- Logistic Regression
- Decision Tree
- Support Vector Classifier (SVC)
- Random Forest
- AdaBoost
- Gradient Boosting
- Classificazione conseguente a Linear Discriminative Analysis

La classificazione da noi desiderata riguarda l'attributo *term_deposit* presente nel dataset; infatti, dato il profilo di un cliente (valori delle feature), l'obiettivo è stato quello di capire se quest'ultimo avrebbe potuto affidare alla banca un deposito sul lungo periodo. Di conseguenza, il task svolto è stato un task di classificazione binaria fra la label "Si" (il cliente potrebbe effettuare un deposito sul lungo periodo) e la label "No" (il cliente non affiderà alla banca un deposito di lungo periodo).

Come già detto in precedenza, gli algoritmi utilizzati sono tutti metodi di learning supervisionato, di conseguenza la qualità dei modelli finali dipende fortemente dalla composizione e dalla bontà del dataset di training. A tal proposito, un problema venutosi a evidenziare fin dalle primissime fasi è stato quello del class-imbalance; ciò vuol dire che nel dataset di partenza era presente un fortissimo sbilanciamento fra la classe "Si" e la classe "No". Nella maggior parte dei casi tutto questo può portare ad avere modelli finali caratterizzati dalla dominanza di un bias che farebbe tendere la classificazione verso la classe maggioritaria. Ad esempio, se dovessimo avere un dataset composto da 999 elementi di classe 1 ed un solo elemento di classe 2, il classificatore prodotto dal training su tale dataset etichetterebbe in maniera indistinta tutti gli input come appartenenti alla classe 1.

Nel nostro caso il training di modelli sul dataset di partenza non ha prodotto risultati così estremi; al contrario, questi ultimi sono stati fin da subito abbastanza soddisfacenti, mostrando un accuracy generale intorno al 90%.

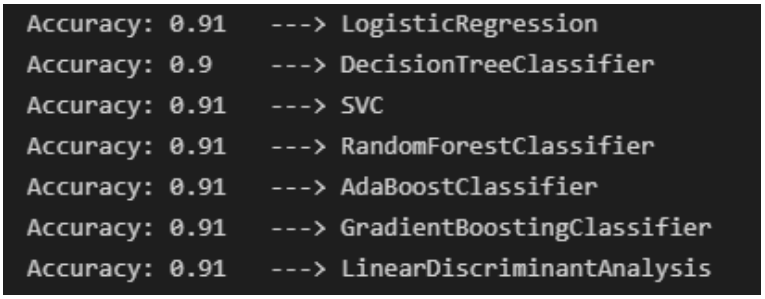
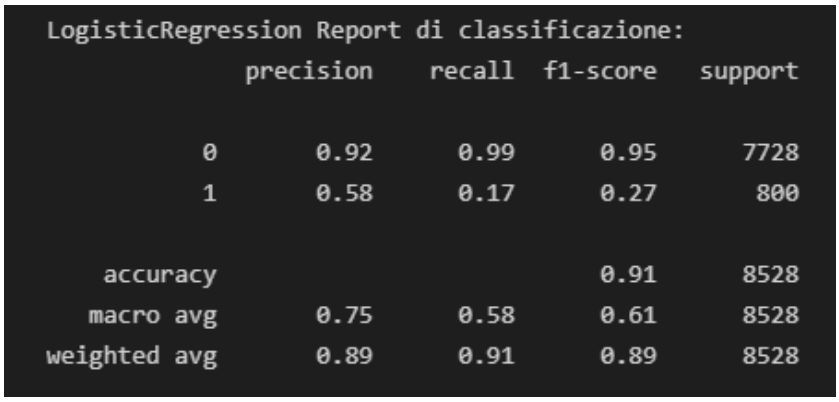


Figura 28: Accuratezze ottenute con i principali classificatori

Il problema legato al class-imbalance è stato più palese dopo aver calcolato le metriche di precision e recall per singola classe utilizzando la Logistic Regression. Infatti, facendo ciò si è visto che il valore della precision per la classe 1 (Si) era parecchio deludente, aggirandosi intorno al 58%; questo indica che quando il classificatore etichetta con classe 1, sbaglia quasi la metà delle volte. Tuttavia, la metrica più grave era la recall, pari al 17%; ciò vuol dire che sul totale dei clienti "interessanti", solo il 17% veniva identificato correttamente come tale. Al contrario, entrambe le metriche relative alla classe maggioritaria erano soddisfacenti e abbondantemente sopra al 90%.



LogisticRegression Report di classificazione:

	precision	recall	f1-score	support
0	0.92	0.99	0.95	7728
1	0.58	0.17	0.27	800
accuracy			0.91	8528
macro avg	0.75	0.58	0.61	8528
weighted avg	0.89	0.91	0.89	8528

Figura 29: Precision e Recall per Logistic Regression

4.3 Approccio al class-imbalance

Al fine di avere un dataset più bilanciato è necessario adottare tecniche apposite finalizzate ad avere un equilibrio migliore fra classi. Il primo approccio da noi adottato è quello del *metacost*.

Metacost

Si tratta di un meta-algoritmo che funziona da estensione ad approcci preesistenti ed il suo obiettivo è quello di rietichettare il dataset in modo che questo torni ad essere bilanciato. Il primo step nell'applicazione del meta-cost consiste nel generare *m* training set a partire da quello iniziale con la tecnica del bootstrap (estrazione casuale con reinserimento). Fatto ciò, vengono addestrati *m* modelli utilizzando l'algoritmo scelto sugli *m* dataset generati; ogni elemento *x* del dataset iniziale viene etichettato da tutti gli *m*

modelli, generando m predizioni genericamente diverse. Vengono calcolate le probabilità finali per ogni classe $P(j|x)$ come somma del numero di volte che x è stato etichettato come appartenente alla classe j. Fatto ciò, calcolando il costo della riassegnazione dell'elemento x alla classe i (usando la formula sotto per ogni classe i) viene trovata la nuova etichetta di x:

$$x\text{'s class} = \text{argmin}_i \sum_j P(j|x)C(i,j)$$

Ovvero x verrà rietichettato come di classe i, dove quest'ultima minimizza la sommatoria dei prodotti dei costi legati al confondere i con ognuna delle altre classi per le probabilità di tutte le altre classi. Rietichettati tutti gli elementi del dataset di partenza, il metacost ritorna il modello M ottenuto applicando l'algoritmo scelto in partenza sul nuovo dataset rietichettato.

Scelta un'implementazione del metacost, sono stati scelti dei costi di errore pari a 4 nel caso di classe 1 (Sì) e pari a 1 nel caso di classe 0 (No); il valore 4 fa sì che l'errore nel caso di classe 1 venga giudicato più pesante in fase di training.

I risultati in termini di accuracy, precision e recall sono riportati sotto e, come è possibile osservare, i miglioramenti sono notevoli.

	precision	recall	f1-score	support
0	1.00	0.96	0.98	38678
1	0.73	1.00	0.84	3961
accuracy			0.97	42639

Figura 30: Metriche dopo l'applicazione del Metacost

Bilanciamento manuale

Un altro approccio adottato è rappresentato dal *sampling* manuale. Per primo, è stato eseguito un undersampling della classe maggioritaria, rispettando la distribuzione del dataset, in modo da diminuirla a circa un terzo della dimensione originale. Successivamente, è stato applicato un oversampling alla classe minoritaria, in modo che le due classi avessero lo stesso numero di elementi.

A questo punto, sono stati addestrati i modelli e sono state calcolate le stesse metriche viste in precedenza. Generalmente, i risultati sono sufficientemente buoni, ma le performance migliori si hanno con il Random Forest (Figura 31): l'accuratezza si assesta su 0.95, la precision e la recall si trovano tra 0.92 e 0.98.

	precision	recall	f1-score	support
0	0.97	0.92	0.94	2306
1	0.93	0.98	0.95	2448
accuracy			0.95	4754

Figura 31: Metriche dopo bilanciamento manuale

Le matrici di confusione (Figura 32) mostrano che i modelli addestrati e testati hanno un comportamento più o meno simile, anche se in generale gli alberi di classificazione (dunque anche il Random Forest) predicono in maniera migliore. Nella matrice di confusione, si rappresentano sulle righe i valori attuali e sulle colonne i valori predetti.

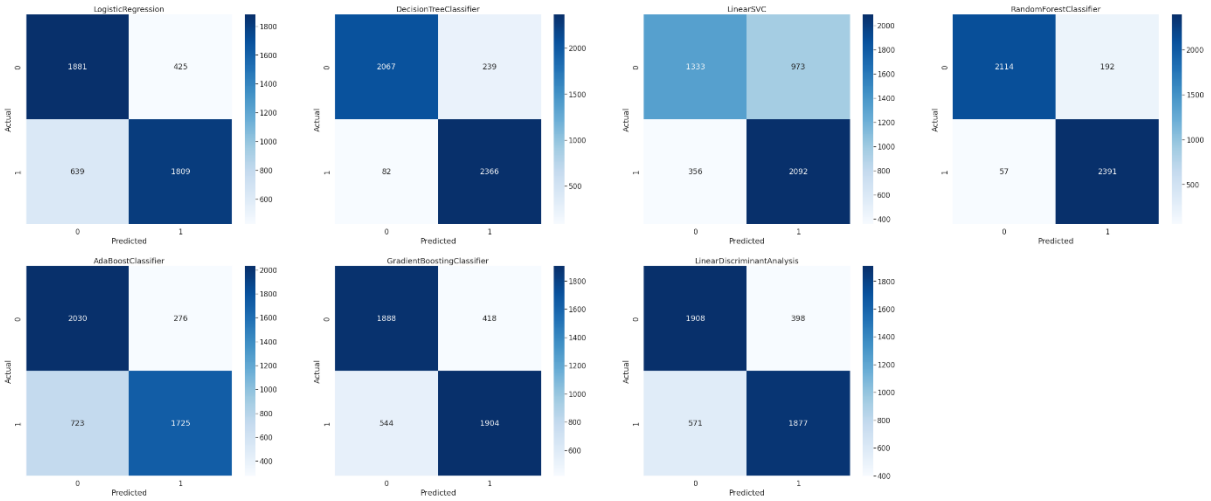


Figura 32: Matrici di confusione per i vari modelli

Un altro metodo utile a valutare i modelli ottenuti è dato dalle curve ROC (Figura 33). In generale, la curva ROC mostra le performance di un modello di classificazione considerando tutte le possibili soglie di classificazione. Tale curva rappresenta due parametri: *True Positive Rate* e *False Positive Rate*.

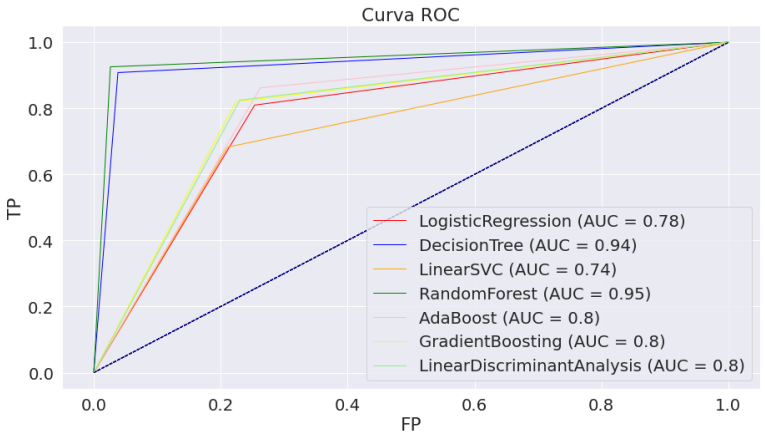


Figura 33: Curve ROC per i vari modelli

In seguito, si è verificata la presenza di correlazione tra i modelli di predizione (Figura 34). In dettaglio, gli alberi sono correlati con il Random Forest, mentre è presente una certa correlazione tra il modello di regressione logistica, il classificatore Gradient Boosting e LDA.

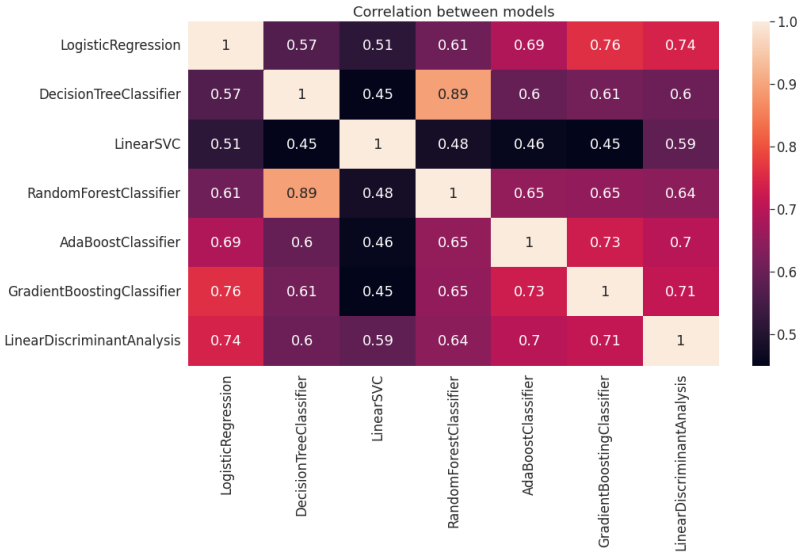


Figura 34: Correlazione tra i modelli di classificazione

4.4 Clustering

La seconda tipologia di task affrontata in questo progetto è stata la clusterizzazione. Ciò indica l'operazione di dividere un insieme di elementi forniti in input in sottogruppi accomunati da determinate caratteristiche. Questa divisione in cluster avviene nello spazio delle features. Quest'ultimo è definito come uno spazio avente come dimensioni le caratteristiche che distinguono gli elementi del nostro dataset (feature). In questa tipologia di task è di fondamentale importanza la scelta di queste feature, ovvero su quali di queste basare la divisione.

Esistono svariati tipi di algoritmi finalizzati a svolgere la clusterizzazione, ma quello scelto è il K-Means. Quest'ultimo ricade nella tipologia di approcci prototype-based, ovvero quelli che si basano sullo spostamento dei centroidi del cluster (punti centrali) e definiscono l'appartenenza di un elemento ad un gruppo in base al centroide più vicino.

Nel caso del K-means si ha un'inizializzazione di partenza dei centroidi, e si presenta un primo problema, ovvero quanti centroidi (quindi cluster) bisogna prevedere. Il metodo più facile per trovare una risposta è chiamato "Elbow Method". Questo metodo consiste nell'applicare l'algoritmo k-means aumentando di volta in volta il numero di cluster usati. Poi, per ognuna delle segmentazioni ottenute viene calcolato il coefficiente medio di distorsione nei cluster ottenuti; si può pensare a questo valore come la distanza media di un punto dal centro del cluster assegnato.

Eseguendo un plot dell'andamento della distorsione in funzione del numero (crescente) di cluster usati, si noterà sia che la distanza media è inizialmente molto alta (pochi cluster) sia che questa ha, nelle prime fasi, una forte tendenza a decrescere. Quando si è raggiunto un numero sufficiente di cluster la distorsione si stabilizza, questo perché i segmenti ottenuti hanno una dimensione talmente piccola che suddividendoli ulteriormente non si hanno più cambiamenti sostanziali del coefficiente di distorsione. Ottenuta questa curva, il numero di cluster adatto è identificato dal punto in cui questa cambia andamento, passando da una forte discesa ad una situazione di maggiore stabilità.

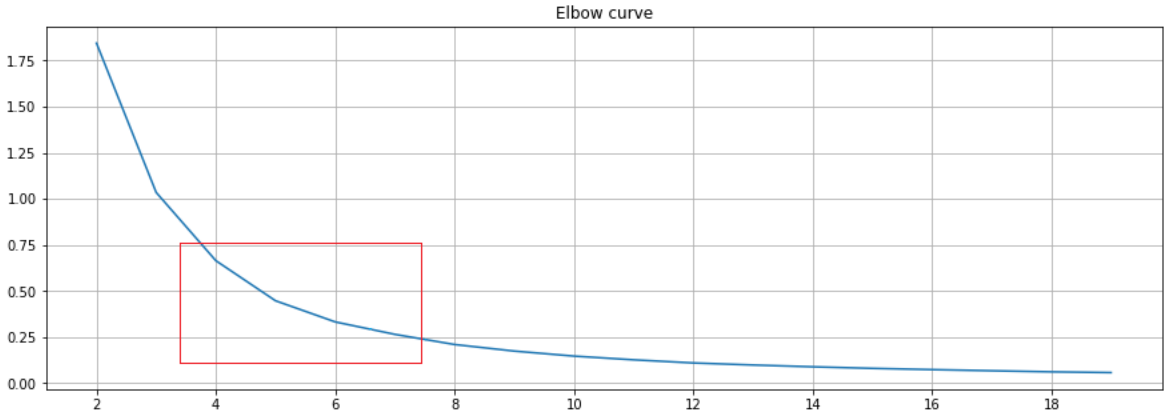


Figura 35: Curva Elbow

Come è possibile osservare il numero di cluster adatto per il nostro dataset sembra essere compreso fra 3 e 7 (area in rosso nella Figura 35).

Una seconda verifica che si può svolgere, per l'identificazione del numero più adatto di sottogruppi, è quella relativa al *coefficiente silhouette* (Figura 36). Questo coefficiente varia fra 1 e -1. Numeri vicini a 1 indicano che gli elementi sono più vicini al centro del loro cluster rispetto ai centri dei cluster circostanti (neighboring clusters). Scorrendo nuovamente attraverso i vari numeri di cluster calcolando di volta in volta, questo coefficiente ci darà una seconda conferma in merito a quanti cluster usare.

Figura 36: Silhouette Coefficient al variare del numero di cluster

L'analisi del coefficiente silhouette indica che, all'aumentare del numero di cluster, si ha un peggioramento quasi lineare di questo valore. Ciò vuol dire che dovremo scegliere un numero di segmenti il più basso possibile. Considerando anche l'analisi fatta sfruttando la curva elbow, si è scelto un numero di cluster **non superiore a 4.**

Avendo effettuato vari test con diversi numeri di cluster, si ottiene una discreta segmentazione del dataset con 4 cluster (Figura 37).

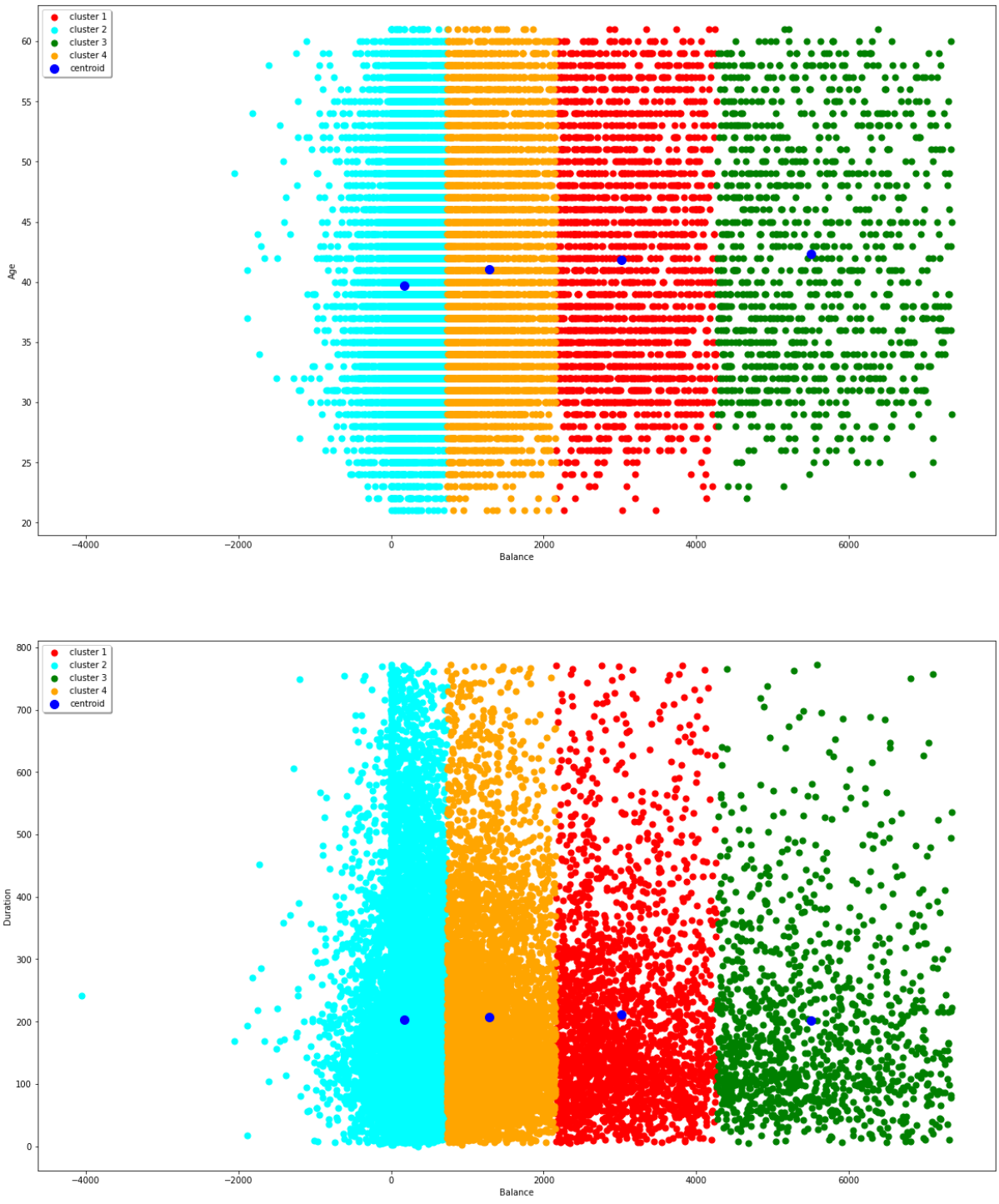
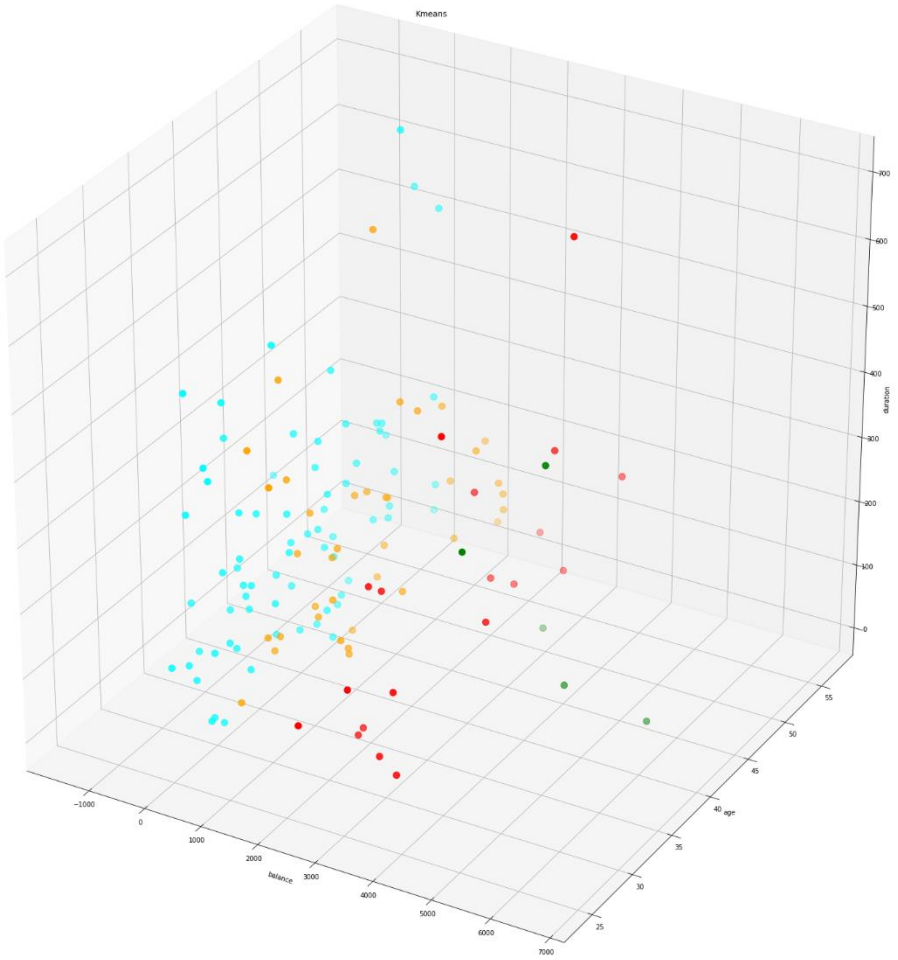


Figura 37: Segmentazione dataset completo

Come si può notare, la qualità della segmentazione non è ottimale e la silhouette si assesta tra 0.5 e 0.6. In particolare, sono state considerate le coppie di feature “age” - “balance” e “balance” – “duration”, poiché con le altre feature categoriche il clustering risultante presentava cluster raggruppati per i valori di quest’ultime. Con tale configurazione, si possono individuare quattro cluster che corrispondono a quattro fasce di reddito. Innanzitutto, il cluster 2 è rappresentativo della fascia povera (intorno allo 0, comprendendo anche valori minori), il cluster 4 rappresenta la fascia medio-bassa (vicino ai 20000 euro), il cluster 3 corrisponde alla fascia più ricca, mentre il cluster 1 rappresenta la fascia medio-alta.

Poiché il dataset contiene una grande quantità di elementi, si è deciso di prendere un campione più piccolo, rispettando la distribuzione originale. Perciò, è stata attuata una riduzione del dataset ed è stato applicato il K-means su 150 punti. Dunque, sono state considerate le stesse coppie di feature viste in precedenza e si sono ottenuti nuovamente quattro cluster, con la silhouette risultante pari a circa 0.63. Come si può osservare in Figura 38, i cluster identificati definiscono gli stessi quattro raggruppamenti di reddito precedentemente descritti. In tal caso, c’è una divisione più netta dei cluster e si può notare che il cluster 2, cioè quello che rappresenta la fascia più povera, è molto più compatto degli altri tre.



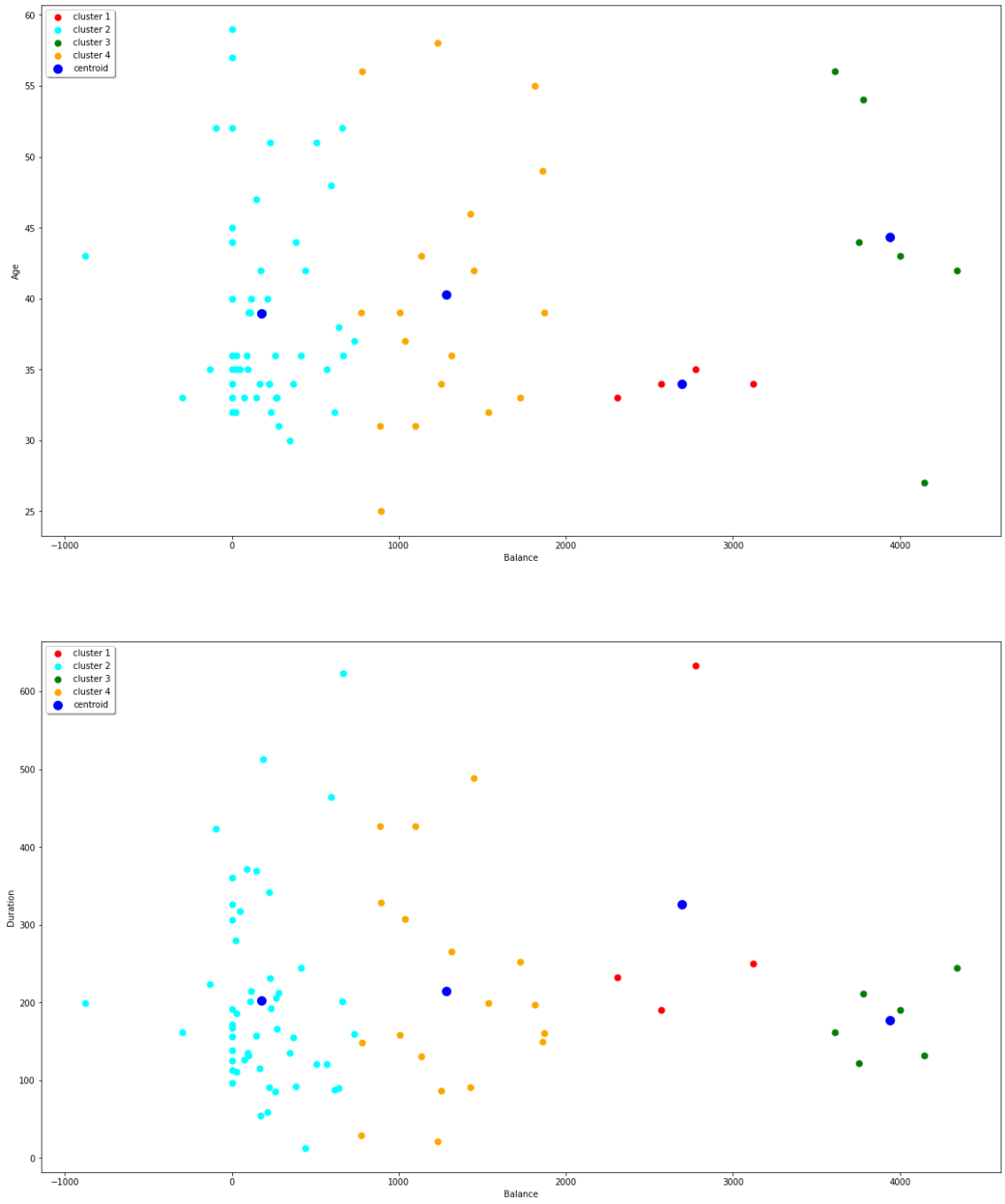


Figura 38: Segmentazione su un campione

Avendo considerato tre feature con tre scale differenti, è stata effettuata una normalizzazione per attenuare l'effetto di sbilanciamento del peso dato ai valori delle feature. Dunque, abbiamo verificato la presenza di differenze tra i cluster prima e dopo la normalizzazione sui 150 elementi. Nella Figura 39, si può notare che i cluster sono molto simili a quelli precedentemente identificati. Infatti, il cluster 2, ovvero quello che rappresenta la fascia più povera, a differenza di quanto accadeva senza normalizzazione, è più distante dal cluster 4, che identifica la fascia medio-bassa. Invece, per quanto riguarda gli altri cluster, c'è una minore separazione, in quanto la

vicinanza tra cluster 1 e cluster 3 è dovuta alla presenza di un maggior numero di elementi in seguito alla normalizzazione. Inoltre, a differenza di quanto visto in precedenza, il cluster rappresentante la fascia più alta risulta essere più compatto. Nella stessa immagine si può individuare una situazione interessante riguardo al rapporto tra “balance” e “duration”; infatti, le fasce centrali (media e bassa) hanno un maggior interesse alla campagna promozionale, testimoniato da una permanenza maggiore al telefono, rispetto agli estremi, poiché nella maggior parte dei casi i più ricchi hanno già depositi a lungo termine e i più poveri non hanno risorse.

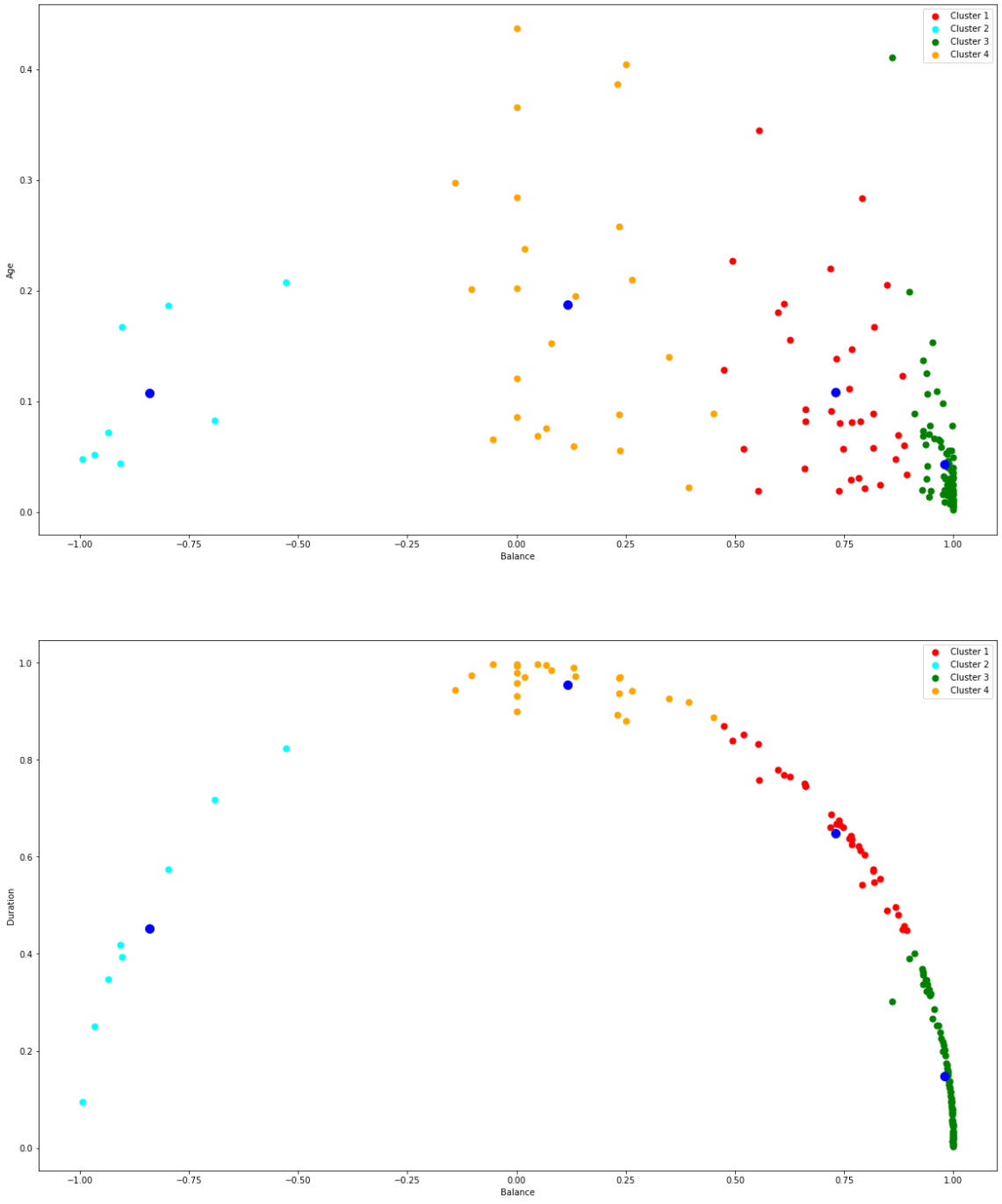


Figura 39: Segmentazione con normalizzazione

Per concludere, si vuole porre l'attenzione sulla composizione del dataset scelto. Le caratteristiche di quest'ultimo non favoriscono l'esecuzione di un task non supervisionato, come il clustering, in quanto, nello spazio delle feature, si ha una distribuzione molto addensata dei sample. Questa osservazione è dimostrata dal fatto che l'esecuzione di algoritmi density based, come il DBSCAN, basandosi sulla sola distanza che separa gli elementi, vanno a raggruppare tutti i sample in un unico cluster.

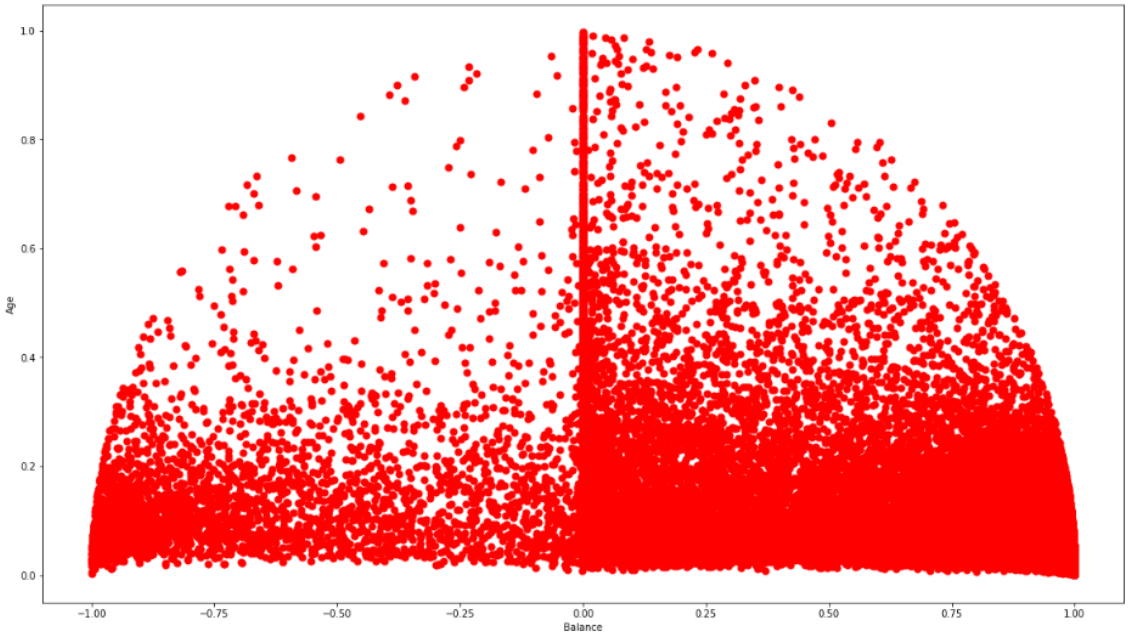


Figura 40: Segmentazione con DBSCAN