Relazione progetto python

Corso di Data Science

A.A. 2023/2024

Docenti:
Prof. Domenico Ursino
Dott. Michele Marchetti

Studenti:
Civitarese Andrea
Longarini Lorenzo
Pasquini Gioele
Ramovini Loris

Sommario

1 Introduzione

In questo progetto si andranno ad analizzare due dataset, il primo riguardante i dati di clienti contattati in una campagna di marketing di una banca portoghese, il secondo riguardante l'andamento di alcuni indici finanziari nel tempo; i due dataset verranno descritti più nello specifico nel seguito della relazione.
La scelta di due dataset è dovuta alle diverse analisi che dovevamo effettuare, ossia classificazione, clustering e serie temporali. Difatti, il primo dataset si presta particolarmente alla classificazione e al clustering poiché contiene diversi attributi per ogni cliente, mentre il secondo contiene dati con un andamento temporale necessari alla creazione di modelli per la predizione di serie temporali.
Per l'analisi di questi dataset è stato utilizzato il linguaggio di programmazione Python, in particolare sono state utilizzate le librerie Numpy, StatsModels, MatPlotLib, SciKit Learn, Pandas e Seaborn.
Abbiamo sfruttato i blocchi note Jupyter per effettuare l'analisi poiché permettono di scrivere passo dopo passo le operazioni da effettuare, senza rieseguire il codice precedente, in modo da rendere l'analisi più flessibile ai risultati parziali che otteniamo.
Il codice utilizzato per il progetto è visualizzabile su GitHub al seguente link.

2 Dataset

In questa sezione andremo a descrivere il contenuto dei due dataset scelti, analizzando nello specifico il significato singoli attributi.

2.1 Bank marketing

Il dataset è presente su Kaggle al seguente link.
In particolare, il dataset contiene dati riguardanti i clienti contattati da una banca portoghese durante una campagna di marketing telefonica.
Sono presenti 45.211 clienti, dei quali 39.922 hanno risposto positivamente alla campagna sottoscrivendo il prodotto offerto dalla banca, ossia un deposito a termine prefissato; i restanti 5.289, invece, non hanno sottoscritto l'offerta.
Le colonne presenti nel dataset sono descritte nella Tabella 1**Error! Reference source not found.** dove, inoltre, sono visibili il conteggio di valori non nulli e la tipologia di dato.

#	Column	Non-Null Count	Dtype
0	Age	45211 non-null	int64
1	Job	45211 non-null	object
2	Marital Status	45211 non-null	object
3	Education	45211 non-null	object
4	Credit	45211 non-null	object
5	Balance (euros)	45211 non-null	int64
6	Housing Loan	45211 non-null	object
7	Personal Loan	45211 non-null	object
8	Contact	45211 non-null	object
9	Last Contact Day	45211 non-null	int64
10	Last Contact Month	45211 non-null	object
11	Last Contact Duration	45211 non-null	int64
12	Campaign	45211 non-null	int64
13	Pdays	45211 non-null	int64
14	Previous	45211 non-null	int64
15	Poutcome	45211 non-null	object
16	Subscription	45211 non-null	int64

Tabella 1: Colonne del dataset bank marketing

Gli attributi che compongono il dataset, riferiti alla singola persona contattata, sono:

0. **Age**: età;
1. **Job**: impiego;
2. **Marital Status**: stato coniugale;
3. **Education**: livello di istruzione;
4. **Credit**: se la persona ha del debito insoluto con la banca;
5. **Balance (euros)**: bilancio medio annuo stimato;
6. **Housing loan**: se la persona ha acceso un mutuo per una casa;
7. **Personal loan**: se la persona ha acceso un mutuo per un investimento privato;
8. **Contact**: tipologia di contatto utilizzata (telefono o cellulare);
9. **Last Contact Day**: giorno di ultimo contatto nella campagna corrente;
10. **Last Contact Month**: mese di ultimo contatto nella campagna corrente;

11. **Last Contact Duration**: durata, in secondi, dell'ultimo contatto telefonico avvenuto;
12. **Campaign**: numero di contatti avvenuti durante la campagna con il cliente, incluso l'ultimo;
13. **Pdays**: numero di giorni passati dall'ultimo contatto in campagne precedenti prima di questa;
14. **Previous**: numero di contatti avvenuti con il cliente prima di questa campagna;
15. **Poutcome**: Esito della precedente campagna di marketing con questo cliente;
16. **Subscription**: se la campagna corrente ha avuto successo con il cliente.

2.2 Andamento indici finanziari

Il dataset è stato ottenuto da Kaggle al seguente link.
Contiene l'andamento temporale degli indici di Oro, Argento, Barrick Gold Corp e S&P 500; quest'ultimo è un indice che riassume l'andamento delle 500 aziende a maggior capitalizzazione quotate alla borsa americana.
I dati (inizialmente divisi in 4 file diversi, ciascuno per ogni indice finanziario) contengono il prezzo di chiusura giornaliero dell'indice e vanno da settembre 2017 a settembre 2020.
La struttura del dataframe finale, una volta riunito il contenuto dei file, è mostrata in Figura 1, dove sono visibili le prime 5 righe.

	Date	GLD	SPX	BARR	SLV
0	2017-08-22	1291.0	2452.51	16.5072	17.060
1	2017-08-23	1294.7	2444.04	16.6639	17.126
2	2017-08-24	1292.0	2438.97	16.6932	17.046
3	2017-08-25	1297.9	2443.05	16.7814	17.132
4	2017-08-28	1315.3	2444.24	17.3003	17.529

Figura 1: Struttura del dataset degli indici finanziari.

3 Classificazione

La classificazione in ambito machine learning è un processo mediante il quale si assegna una categoria o una classe a un'istanza o a un'osservazione in base alle sue caratteristiche.
L'obiettivo della classificazione è quello di creare un modello che possa generalizzare le relazioni tra le caratteristiche delle istanze e le rispettive classi, in modo che possa classificare correttamente nuove istanze non viste in precedenza.
Durante la fase di addestramento, il modello di classificazione impara dai dati di addestramento, cercando di trovare i pattern e le relazioni tra le caratteristiche e le classi di appartenenza.
Una volta addestrato, il modello viene valutato utilizzando il set di test, dove le previsioni del modello vengono confrontate con le vere etichette delle osservazioni.
Infine, il modello addestrato può essere utilizzato per classificare nuove istanze di dati, dove le caratteristiche delle istanze vengono inserite nel modello e viene prodotta una previsione sulla classe di appartenenza.
I modelli di classificazione utilizzati nel progetto sono stati implementati tramite la libreria SciKitLearn.

3.1 ETL e analisi descrittiva

Il dataset utilizzato per la classificazione è quello descritto nel paragrafo 2.1 contenente i dati di una campagna di marketing di una banca portoghese.
Per l'ETL è stata utilizzata la libreria Pandas di Python.
La prima verifica effettuata è stata quella dei valori nulli, ossia andare a vedere se erano presenti delle righe nel dataset che al loro interno contenevano valori nulli.
Come si può vedere dalla tabella riportata nel paragrafo 2.1, non sono presenti valori nulli per alcuna delle righe del dataset.
Abbiamo poi scelto come etichetta di target per la classificazione il valore dell'attributo Subscription, ossia siamo andati a costruire dei modelli di classificazione che, in base al valore di tutti gli altri attributi del dataset, predicessero se il cliente sottoscriverà il prodotto pubblicizzato oppure no.
Come si può vedere dalla Figura 2, all'interno del dataset, l'88,3% dei clienti ha accettato di sottoscrivere il prodotto offerto (etichetta 1), mentre solo l'11,7% non ha accettato (etichetta 0).
In termini numerici, ci sono 39.922 clienti che hanno accettato e 5.289 clienti che hanno rifiutato.
Per questo andremo poi a prendere solo una porzione del dataset, in modo da ottenere un train set bilanciato tra le due classi.

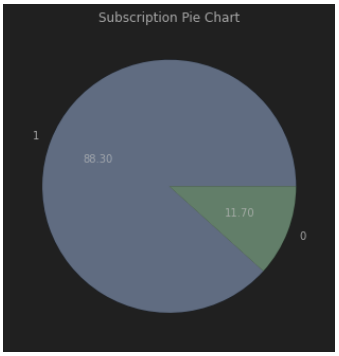

Figura 2: Grafico a torta dei clienti in base all'esito della campagna pubblicitaria.

Un'altra analisi descrittiva effettuata è stata quella riguardo il lavoro dei clienti contattati, mostrata in Figura 3. Come possiamo notare, la maggior parte dei clienti contattati lavora nell'ambito del

management o sono dei tecnici. Porzioni più piccole del dataset sono composte da imprenditori o cosiddetti “colletti blu”, ossia operai. Le altre professioni compongono fette minori dei dati disponibili. Possiamo notare quindi come la campagna pubblicitaria abbia interessato tutte le fasce di lavoratori, dai più ai meno facoltosi.

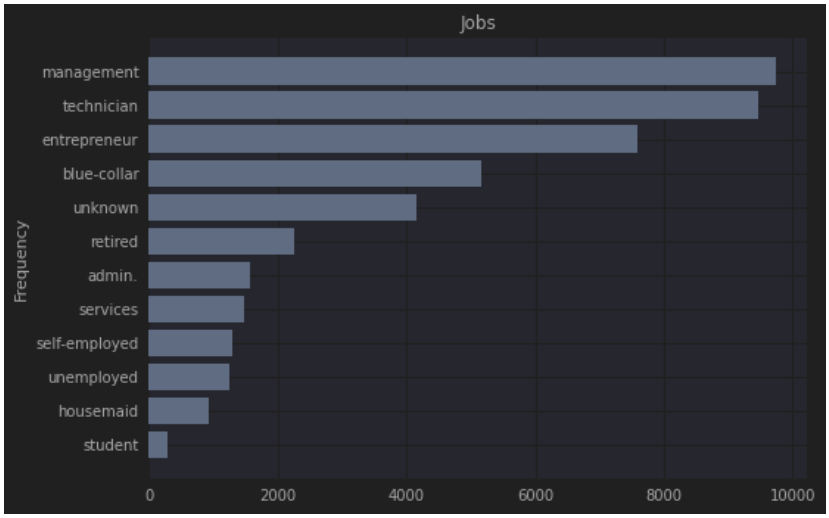

Figura 3: Impiego dei clienti contattati nella campagna pubblicitaria.

Abbiamo poi analizzato la tipologia di educazione in base all’impiego, il risultato è mostrato nella Figura 4. Si evidenzia il fatto che, nella categoria dei lavoratori nel campo del management e degli imprenditori la maggior parte è laureata, mentre nelle categorie dei tecnici, operai, servizi e le altre tipologie di impiego, il livello di istruzione sia più basso.

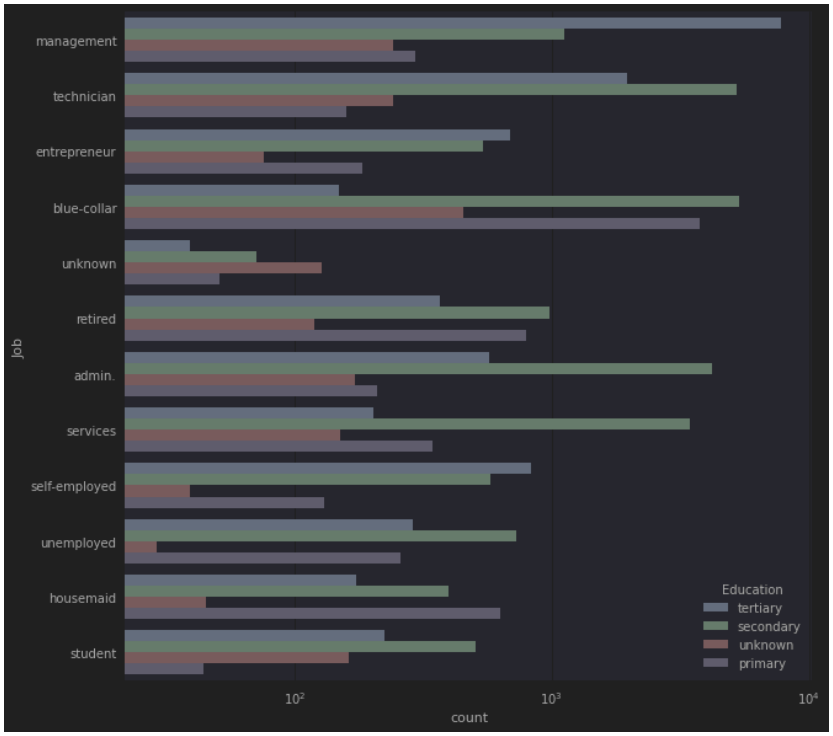

Figura 4: Livello di educazione in base all'impiego.

Sempre in relazione al livello di istruzione, abbiamo graficato, in base a ogni livello di quest’ultimo attributo, il numero di successi e insuccessi della campagna pubblicitaria. Il risultato è riportato in Figura 5.

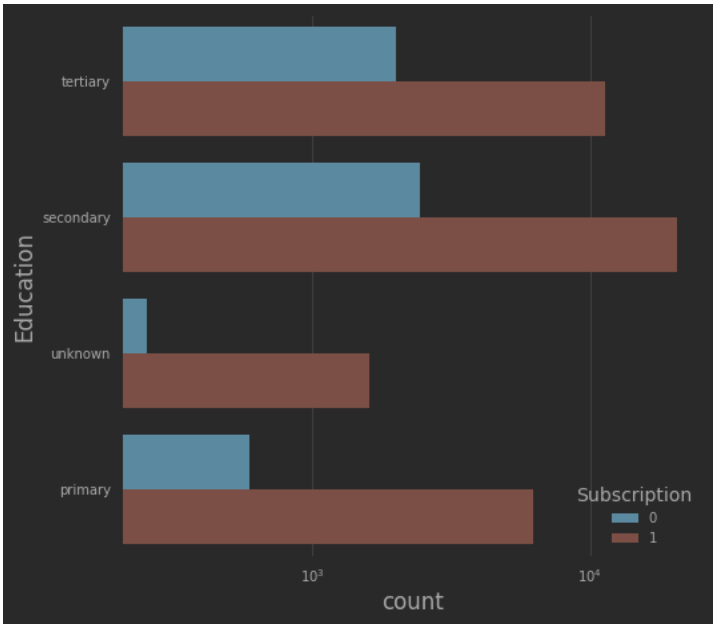

Figura 5: Livello di istruzione ed esito della campagna pubblicitaria.

Sempre riguardo le sottoscrizioni, il loro numero in base all'età è descritto nel grafico riportato in Figura 6, dove si può notare come la campagna si sia concentrata su una fascia d'età dai 30 ai 50 anni.

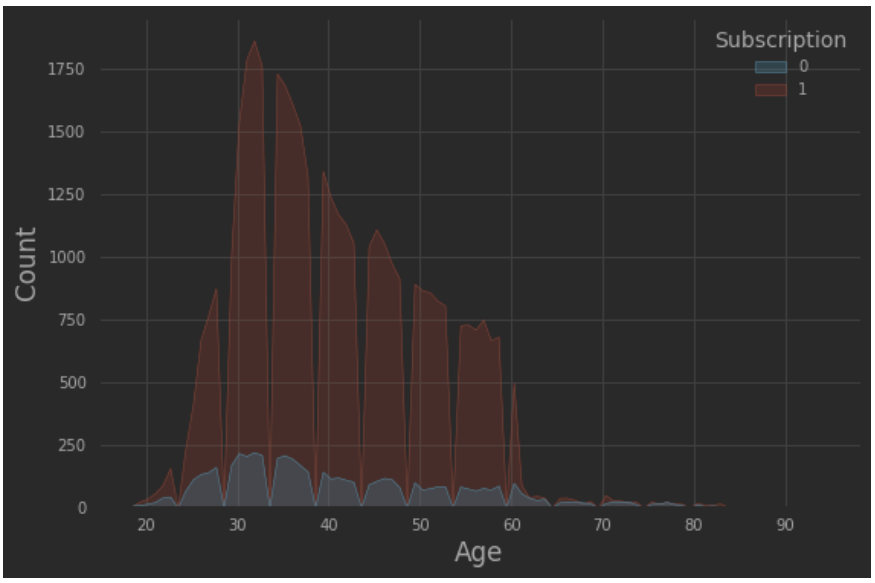

Figura 6: Sottoscrizioni in base all'età.

La Figura 7 rappresenta i risultati della scorsa campagna pubblicitaria per i clienti contattati in questa campagna, emerge che la scorsa campagna non ha avuto un tasso di successo positivo come la corrente; coloro per cui il risultato è '*unknown'*, significa che nella scorsa campagna pubblicitaria non sono stati contattati.

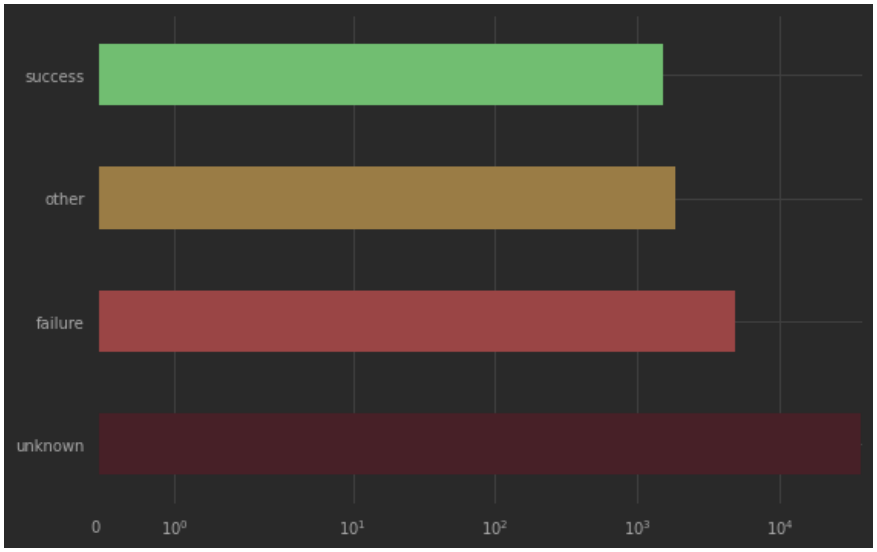

Figura 7: Esito della scorsa campagna pubblicitaria per i clienti contattati.

Terminata questa analisi descrittiva, il passo successivo è stato convertire in attributi discreti in attributi numerici per rendere i dati compatibili con gli algoritmi che saremmo andati ad utilizzare. Ciò è stato fatto tramite la funzione *factorize()* della libreria Pandas.
Una volta fatto questo, abbiamo calcolato la correlazione tra i valori degli attributi presenti nel dataset che viene riportata in Figura 8.

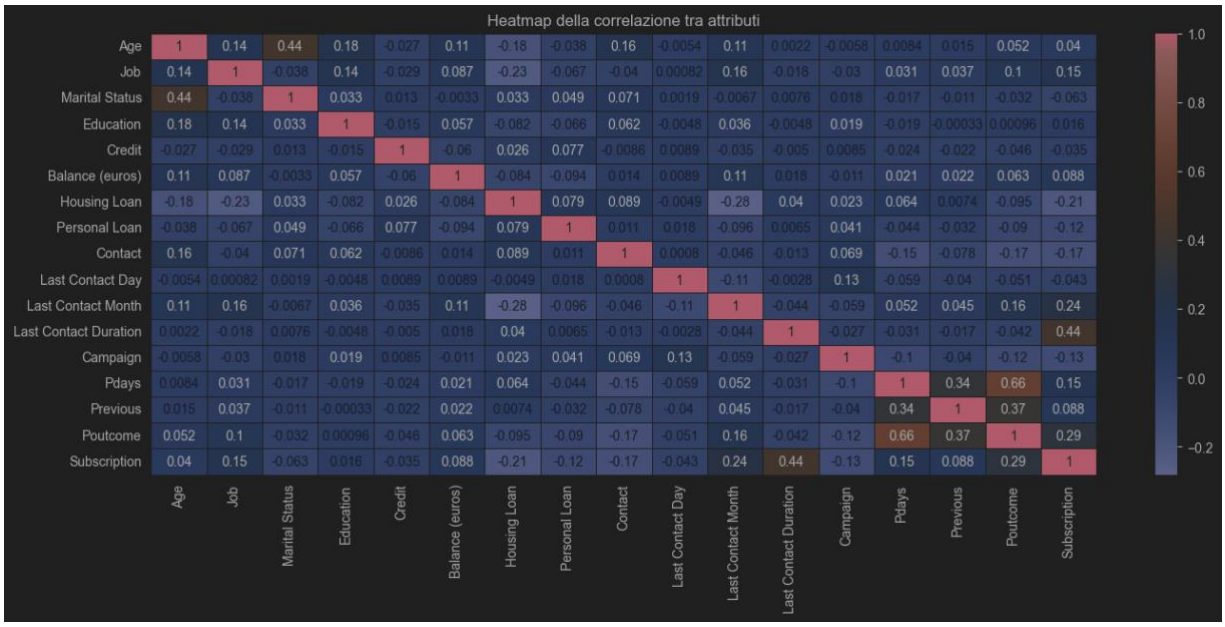

Figura 8: Correlazione tra i valori degli attributi del dataset.

I valori di correlazione sono molto bassi se non per le coppie “Age - Marital status” e “Last Contact Duration – Subscription” che non superano comunque il valore di 0,5 e “Pdays - Poutcome” che ha valore di 0,66. Per questo abbiamo deciso di mantenere tutti gli attributi per effettuare la classificazione.
I valori all’interno del dataset sono stati poi normalizzati prima di essere dati in pasto agli algoritmi, questo per far sì che i diversi attributi mantenessero valori simili ed evitare che alcuni venissero

considerati più importanti di altri dagli algoritmi. Ciò è stato effettuato tramite la classe *StandardScaler* messa a disposizione dalla libreria SciKit Learn.
L'ultimo passo della fase di ETL è stato creare i set di train e test.
Come detto precedentemente, sono solo 5.289 i clienti che hanno rifiutato l'offerta. Per creare un dataset bilanciato tra le due classi abbiamo allora deciso di campionare casualmente 5.289 clienti tra coloro che avevano accettato l'offerta.
Il dataset per i modelli sarà allora composto da tutti i clienti con cui la campagna di marketing ha fallito e il campione estratto casualmente, di uguale numerosità, con cui la campagna ha avuto successo.
Questo dataset è poi stato diviso in due dataset, uno di train e uno di test composti rispettivamente dall'80% e dal 20% dei campioni estratti. La suddivisione è stata effettuata mantenendo la stratificazione delle etichette, ossia in entrambi i subset è stato mantenuto il rapporto di 50/50 tra i clienti che hanno sottoscritto il prodotto e clienti che non lo hanno fatto.

3.2 Esperimenti di classificazione

Passiamo adesso alla classificazione vera e propria. Vedremo una serie di esperimenti effettuati tramite modelli implementati nella libreria SciKit Learn.

3.2.1 Modelli di classificazione

Abbiamo testato sette modelli di classificazione costruiti tramite algoritmi di machine learning; questi modelli sono:

- ***LogisticRegression****;*
- ***DecisionTreeClassifier****;*
- ***SVC****;*
- ***RandomForestClassifier****;*
- ***AdaBoostClassifier****;*
- ***GradientBoostingClassifier****;*
- ***LinearDiscriminantAnalysis****.*

Questi modelli sono stati addestrati con il train set e le loro performance sono state poi verificate tramite il test set. Riportiamo in Figura 9 i valori numerici di accuratezza dei vari modelli.

```
Accuratezza del modello LogisticRegression: 0.8147448015122873
Accuratezza del modello DecisionTreeClassifier: 0.7944234404536862
Accuratezza del modello SVC: 0.832703213610586
Accuratezza del modello RandomForestClassifier: 0.8501890359168242
Accuratezza del modello AdaBoostClassifier: 0.8123818525519849
Accuratezza del modello GradientBoostingClassifier: 0.8388468809073724
Accuratezza del modello LinearDiscriminantAnalysis: 0.8081285444234405
```

Figura 9: Accuratezza dei modelli di classificazione.

In Figura 10 riportiamo anche le matrici di confusione di test dei vari modelli.

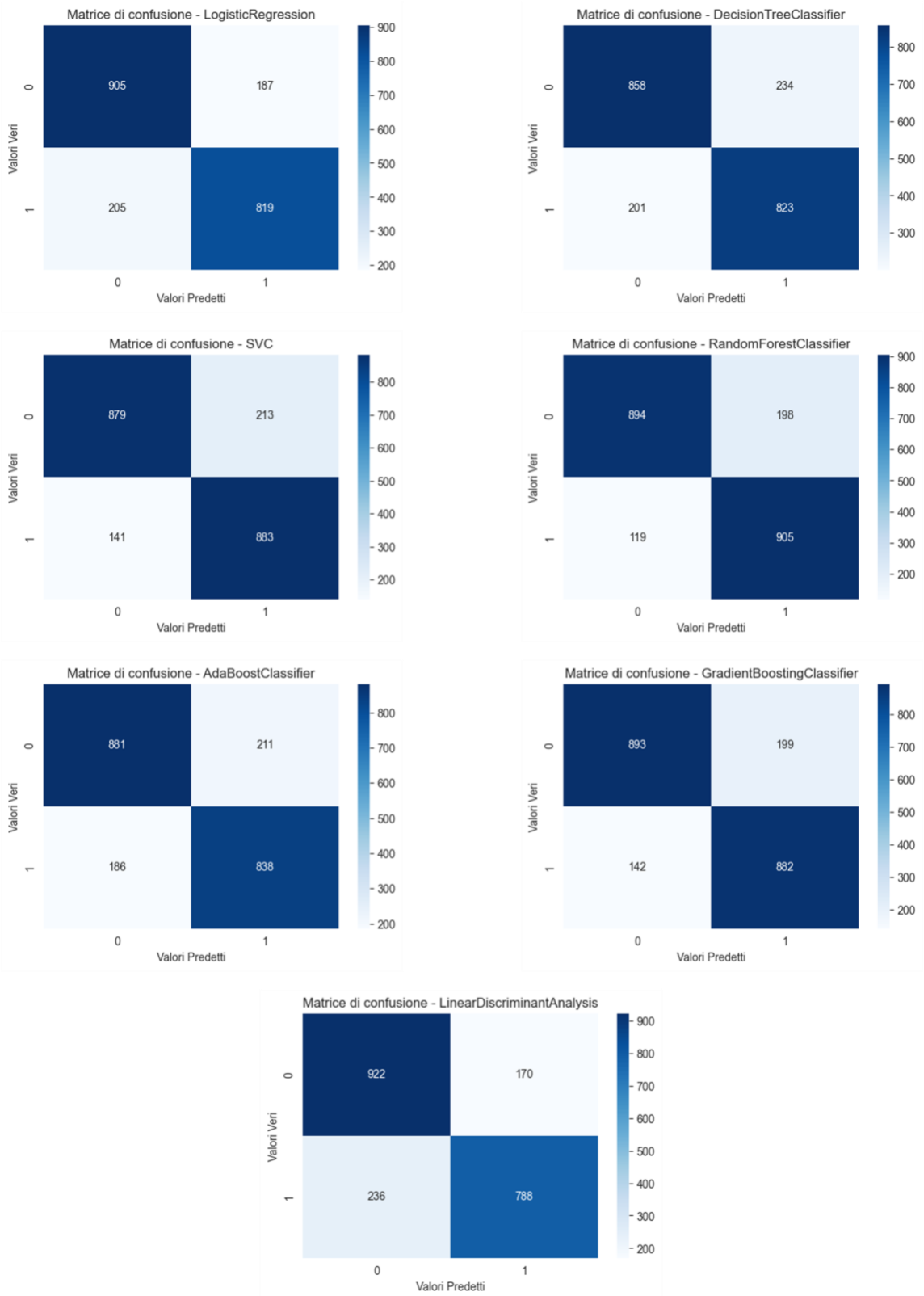

Figura 10: Matrici di confusione dei modelli di classificazione.

La curva ROC (Receiver Operating Characteristic) è un altro metodo comunemente utilizzato per valutare le prestazioni di un modello di classificazione. Rappresenta la relazione tra la sensibilità (True Positive Rate) e la specificità (False Positive Rate) del modello, considerando tutte le possibili

soglie di classificazione. In sostanza, mostra quanto bene il modello può distinguere tra le classi positive e negative. Una curva ROC ideale si avvicina il più possibile all'angolo in alto a sinistra del grafico, indicando una sensibilità elevata e una bassa tasso di falsi positivi.
L'AUC (Area Under the Curve) è un'importante misura di performance per valutare i modelli di classificazione attraverso la curva ROC. Rappresenta l'area sotto la curva ROC e fornisce una misura complessiva della capacità di discriminazione del modello. L'AUC varia da 0 a 1, dove un valore più vicino a 1 indica un modello con migliori capacità discriminanti, mentre un valore più vicino a 0 indica un modello con prestazioni peggiori. In pratica, un'AUC pari a 0,5 corrisponde a una prestazione casuale, mentre un'AUC pari a 1 indica una prestazione perfetta.
In Figura 11 sono riportate le curve ROC e i valori di AUC dei modelli ottenuti.

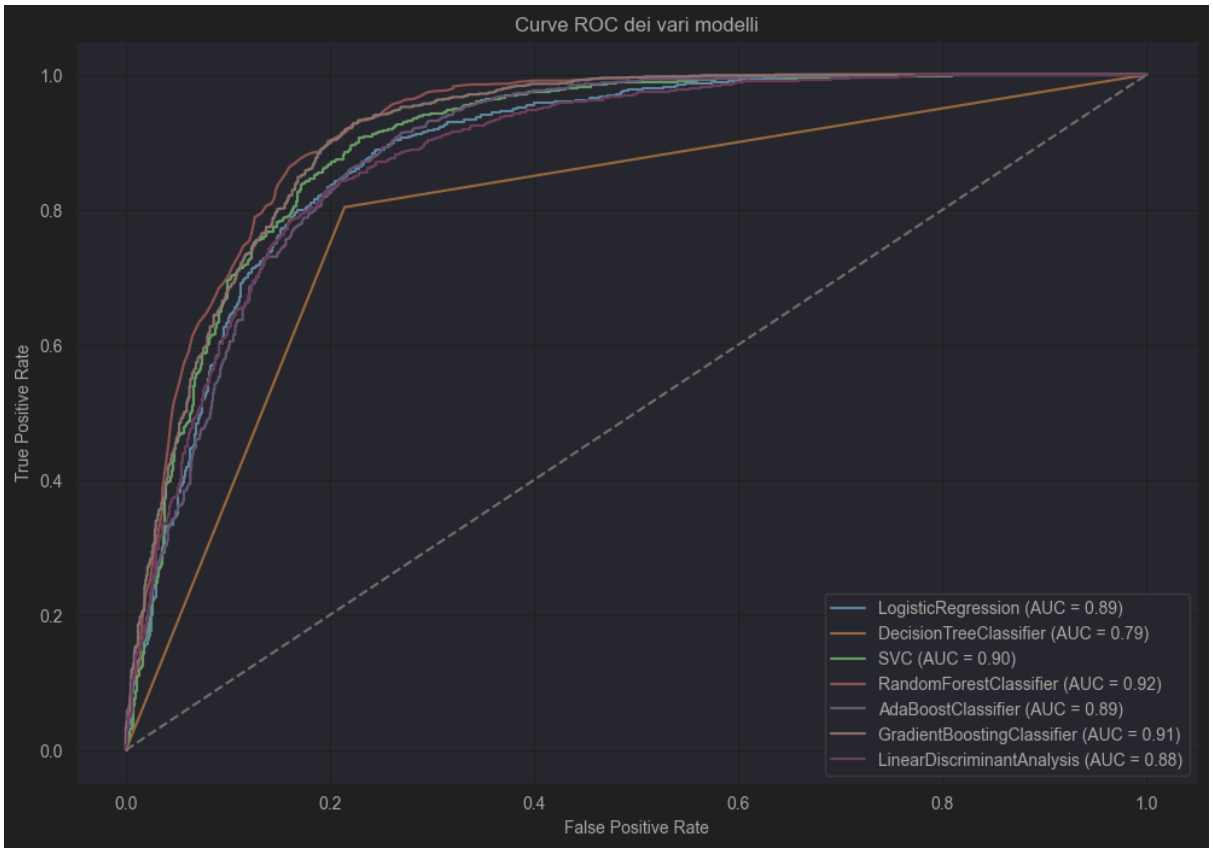

Figura 11: Curve ROC dei modelli di classificazione.

In base ai valori di accuratezza, alle matrici di confusione, alle curve ROC e ai valori di AUC possiamo dire che i migliori modelli ottenuti sono ***SVC***, ***Random Forest Classifier*** e ***Gradient Boosting Classifier***.
Un'altra analisi effettuata sui risultati dei modelli è stata calcolare la correlazione tra le predizioni sui dati di test, riportata in Figura 12.
Possiamo notare come le predizioni tra la maggior parte dei modelli siano altamente correlate (valore di correlazione pari a 0,79 o superiore), questo non vale però per il ***Decision Tree*** per il quale i valori di correlazione con gli altri modelli sono inferiori; difatti questo modello è quello che ha avuto le performance peggiori.

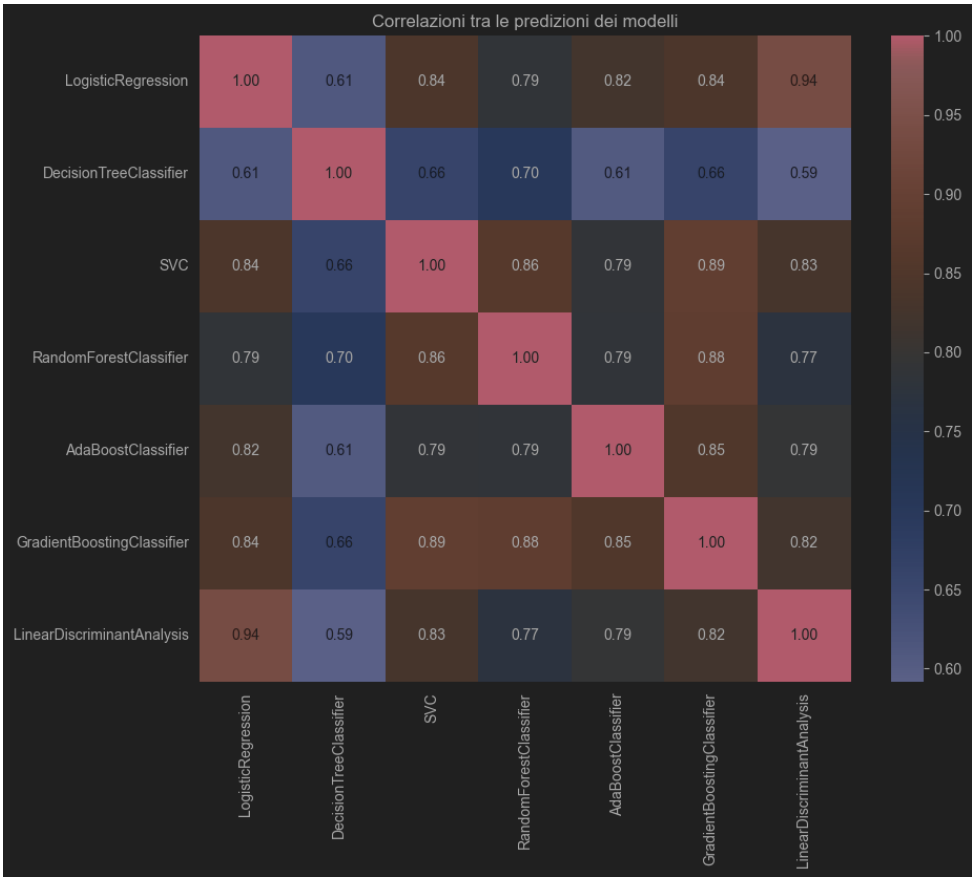

Figura 12: Correlazione tra le predizioni dei vari modelli.

3.2.2 Ricerca dei migliori parametri

Sui tre migliori modelli è stata poi effettuata una GridSearch, ossia una tecnica che effettua il train di questi modelli utilizzando diverse combinazioni di parametri precedentemente specificate.
Le combinazioni di parametri utilizzate sono riportate in Figura 13.

```
parametri_svc = {
    'C': [0.1, 1, 10],
    'gamma': [0.01, 0.1, 1],
    'kernel': ['linear', 'rbf']
}
parametri_random_forest = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['log2', 'sqrt']
}
parametri_gradient_boost = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.1, 0.5, 0.01],
    'max_depth': [2, 3, 4]
}
```

Figura 13: Parametri utilizzati per gli addestramenti con GridSearch.

Il punteggio utilizzato per valutare quale combinazione di parametri di addestramento realizzi il miglior classificatore è l'accuratezza.

I risultati della GridSearch sono riportati in Figura 14.

```
Migliori parametri per SVC: {
    'C': 1,
    'gamma': 0.1,
    'kernel': 'rbf'
}
Miglior punteggio per SVC: 0.8288826288483118

Migliori parametri per RandomForestClassifier: {
    'max_depth': None,
    'max_features': 'log2',
    'min_samples_leaf': 2,
    'min_samples_split': 10,
    'n_estimators': 150
}
Miglior punteggio per RandomForestClassifier: 0.8519271622909236

Migliori parametri per GradientBoostingClassifier: {
    'learning_rate': 0.1,
    'max_depth': 4,
    'n_estimators': 200
}
Miglior punteggio per GradientBoostingClassifier: 0.8515729713739323
```

Figura 14: Risultati GridSearch.

È importante notare che i valori di accuratezza riportati nei risultati della GridSearch sono relativi ai dati di train del modello, poiché questo algoritmo si occupa del solo addestramento.
Successivamente abbiamo verificato le performance dei migliori modelli ottenuti tramite GridSearch e li abbiamo confrontate con i modelli ottenuti inizialmente. I risultati di questo confronto sono riportati in Tabella 2**Error! Reference source not found.**.

Modello	Accuratezza senza GridSearch	Accuratezza con GridSearch
SVC	0.8327	0.8327
Random Forest Classifier	0.8502	0.8502
Gradient Boosting Classifier	0.8388	0.8497

Tabella 2: Confronto tra modelli inziali e modelli ottenuti con GridSearch.

Osserviamo come i modelli ottenuti con il GridSearch per SVC e Random Forest abbiamo le stesse prestazioni, mentre per il Gradient Boosting si è riusciti a trovare una combinazione di parametri di addestramento che ne ha migliorato le performance.

3.2.3 Modello ensemble

L'ultimo esperimento effettuato è stato andare a creare un modello cosiddetto *ensemble*, ossia un modello composto da diversi sotto-modelli di classificazione.
I sotto-modelli utilizzati al suo interno sono i 3 modelli migliori descritti sopra; questi sono stati uniti tramite la classe *VotingClassifier* di SciKit Learn.

Quando un modello ensemble deve effettuare la classificazione di un dato, questo viene dato in pasto ai suoi sotto-modelli e in qualche modo le classificazioni predette vengono aggregate per fornire un unico risultato.
Abbiamo utilizzato un sistema di votazione 'hard', ossia viene predetta la classe con la maggioranza di predizioni dei sotto-modelli.
L'accuratezza del modello ensemble sui dati di test è risultata essere di **0,8469**.
Possiamo notare come il valore di accuratezza sia inferiore sia al modello Random Forest che al modello Gradient Boosting presi singolarmente, tutta via questo modello è sicuramente più affidabile poiché utilizza le predizioni di tre modelli di classificazione diversi.
Riportiamo, inoltre, in Figura 15 la curva ROC, il valore di AUC, la curva Precision-Recall e il valore di Average Precision del modello ensemble.

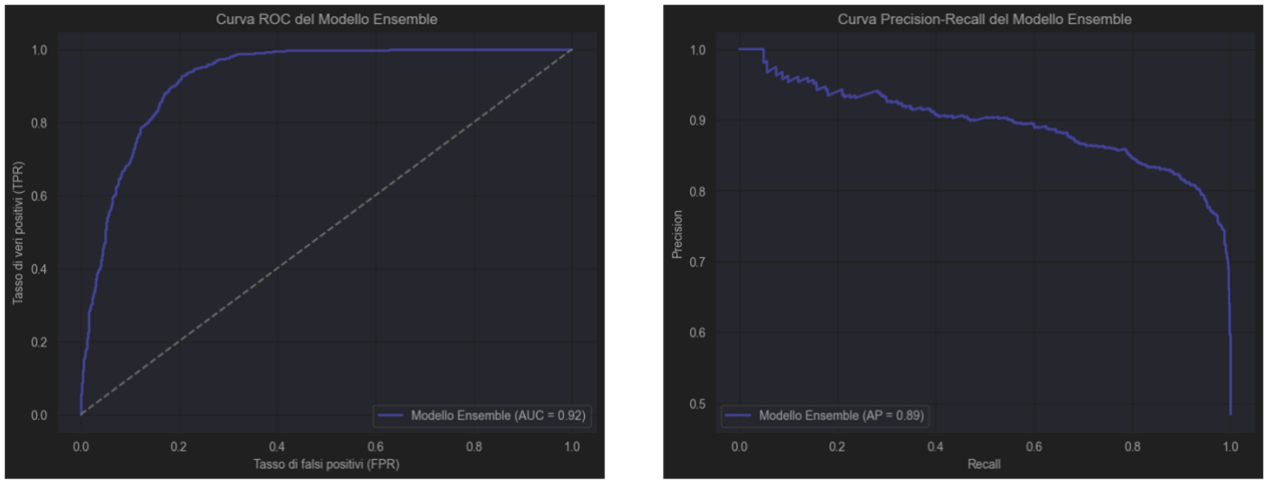

Figura 15: Curve ROC e Precision-Recall del modello ensemble.

L'analisi della curva di apprendimento, come mostrato nella Figura 16, è stata utilizzata per valutare quanto bene il modello è in grado di apprendere dai dati. Questa curva confronta il punteggio di cross validation con il punteggio di addestramento. Quando i due punteggi sono simili, indica un buon apprendimento del modello, poiché il modello è in grado di generalizzare bene dai dati di addestramento ai dati di validazione. In questo caso, possiamo notare un buon apprendimento poiché i due punteggi sono vicini tra loro.

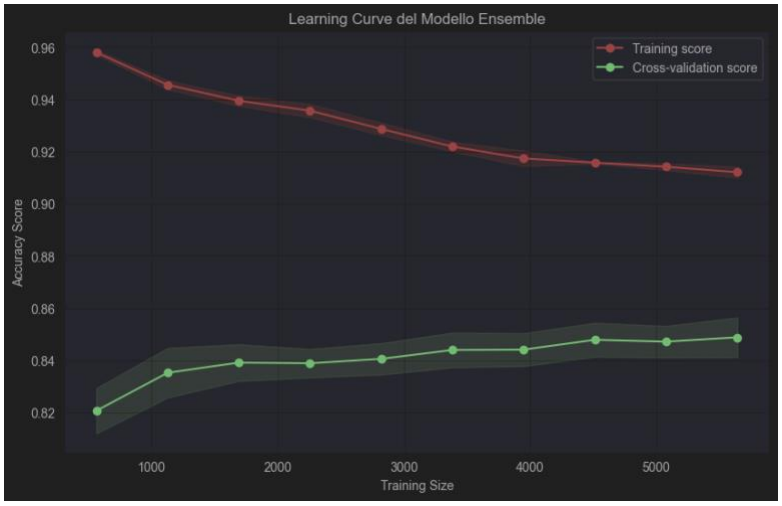

Figura 16: Curva di apprendimento del modello ensemble.

4 Clustering

Il clustering è una tecnica di analisi dei dati utilizzata per raggruppare insiemi di oggetti simili all'interno di un dataset. L'obiettivo è di suddividere i dati in gruppi omogenei in base alle caratteristiche intrinseche degli oggetti, senza conoscere a priori le etichette di classe.
Tra i diversi tipi di clustering, possiamo citare:

- **clustering basato su partizioni**, come il K-Means, dove i dati vengono suddivisi in un numero predeterminato di cluster;
- **clustering gerarchico**, che costruisce una struttura ad albero dei cluster, suddividendo o unendo iterativamente gruppi di dati;
- **clustering basato su densità**, come il DBSCAN, che trova cluster basandosi sulla densità dei dati nello spazio delle feature, piuttosto che su centroidi;
- **clustering basato su modelli probabilistici**, come il Gaussian Mixture Model (GMM), che assume che i dati siano generati da un mix di distribuzioni gaussiane e assegna probabilità a ciascun punto di appartenere a un cluster;
- **clustering basato su vincoli**, dove i cluster sono formati rispettando specifici vincoli imposti dall'utente.

Il clustering trova applicazioni in vari campi, come il marketing, la biologia computazionale e l'analisi delle reti sociali, consentendo di identificare pattern nascosti nei dati e di prendere decisioni informate basate su gruppi omogenei.
In particolare, nel nostro progetto abbiamo scelto di utilizzare due algoritmi per clusterizzare i diversi dati: il k-means e il DBSCAN (implementati tramite la libreria SciKitLearn).
Il dataset da noi utilizzato per il clustering è quello descritto nel paragrafo 2.1; lo stesso utilizzato anche per la classificazione. Avendo il dataset di partenza ben 16 feature, anche eliminandone 8 da ritenute da noi superflue ('Contact', 'Last Contact Day', 'Last Contact Month', 'Last Contact Duration', 'Campaign', 'Pdays'), avremmo avuto un totale di 12 feature: troppe per poter pensare di graficare i cluster che vengono fuori applicando i diversi algoritmi. La soluzione a questo problema sta nell'applicazione di un altro algoritmo, la PCA, che consente ridurre la dimensionalità dei dati, mantenendo al contempo la maggior parte delle informazioni contenute nei dati originali.
In particolare, i due algoritmi presi in considerazione verranno testati sui dati ridotti a 2 e 3 feature, al fine di rendere i plot dei cluster visibili e chiari.

4.1 Esperimenti con K-means

Il processo di clusterizzazione nel K-means inizia assegnando casualmente dei centroidi (punti rappresentativi) ai cluster. Dopodichè, ogni punto nel dataset viene assegnato al cluster il cui centroide è più vicino. Successivamente, per ogni cluster, si calcola il nuovo centroide come la media aritmetica dei punti assegnati ad esso. Questo passaggio viene ripetuto iterativamente finché i centroidi non convergono o finché il numero massimo di iterazioni è stato raggiunto.
L'algoritmo cerca di minimizzare la somma delle distanze quadrate tra ciascun punto nel cluster e il centroide del cluster stesso.
Un aspetto importante del K-Means è che richiede la specifica del numero di cluster desiderati, denotato come K. La scelta di K può influenzare significativamente i risultati del clustering e può variare a seconda delle caratteristiche dei dati e degli obiettivi dell'analisi.

Il K-Means è computazionalmente efficiente e può essere implementato in modo relativamente semplice. Tuttavia, è sensibile alla scelta iniziale dei centroidi e può convergere verso un minimo locale anziché verso la soluzione ottimale. Pertanto, spesso viene eseguito più volte con diverse inizializzazioni per mitigare questo problema.

4.1.1 K-means con dati a 3 dimensioni

Per prima cosa, abbiamo utilizzato la PCA per portare i dati già normalizzati da 12 a 3 feature. Successivamente, abbiamo applicato il “metodo del gomito”, ossia una tecnica utilizzata per determinare il numero ottimale di cluster in un dataset. Esso si basa sull'osservazione della variazione dell'inertia (o somma delle distanze quadrate tra i punti e i centroidi dei cluster) al variare del numero di cluster. In sostanza, si esegue il clustering su un dataset per un range di valori di K (nel nostro caso da 1 a 10) e si calcola l'inertia per ciascun valore di K. Successivamente, si traccia un grafico dell'inertia rispetto al numero di cluster (riportato in Figura 17): l'idea è che l'inertia diminuisca man mano che aumenta il numero di cluster, ma a un certo punto il tasso di diminuzione si riduce drasticamente, formando un "gomito" nel grafico. Il punto in cui si verifica questo gomito è considerato il numero ottimale di cluster da utilizzare per la suddivisione dei dati. Nel nostro caso, il numero di cluster ottimale corrisponde a 4.

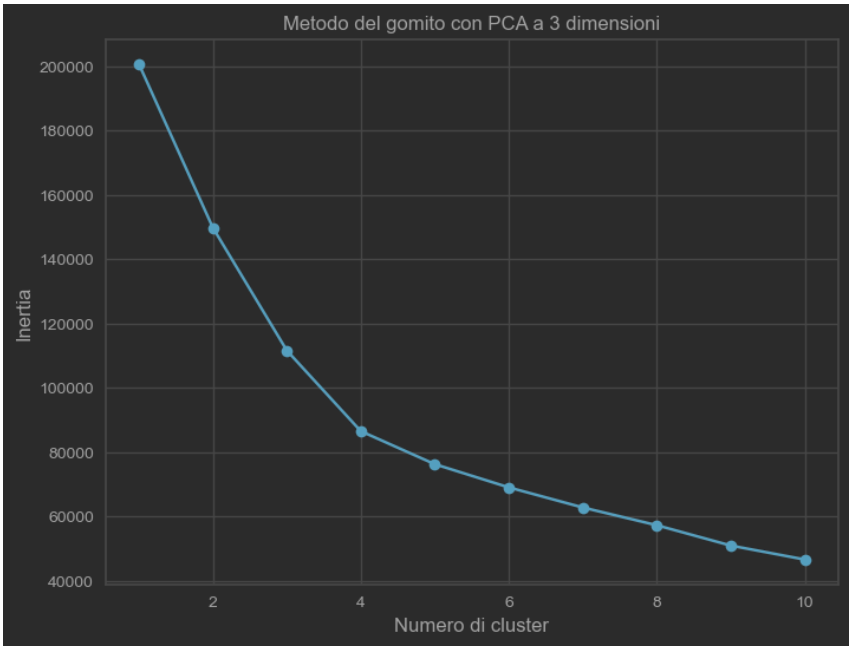

Figura 17: Metodo del gomito per K-means con dati a 3 dimensioni.

Una volta determinato il numero ottimale di cluster, è stato possibile utilizzare l’algoritmo, che ha restituito i 4 cluster riportati in **Error! Reference source not found.** (per una totale rappresentazione dei 4 cluster, vengono riportate tutte le possibili viste a 3 dimensioni).

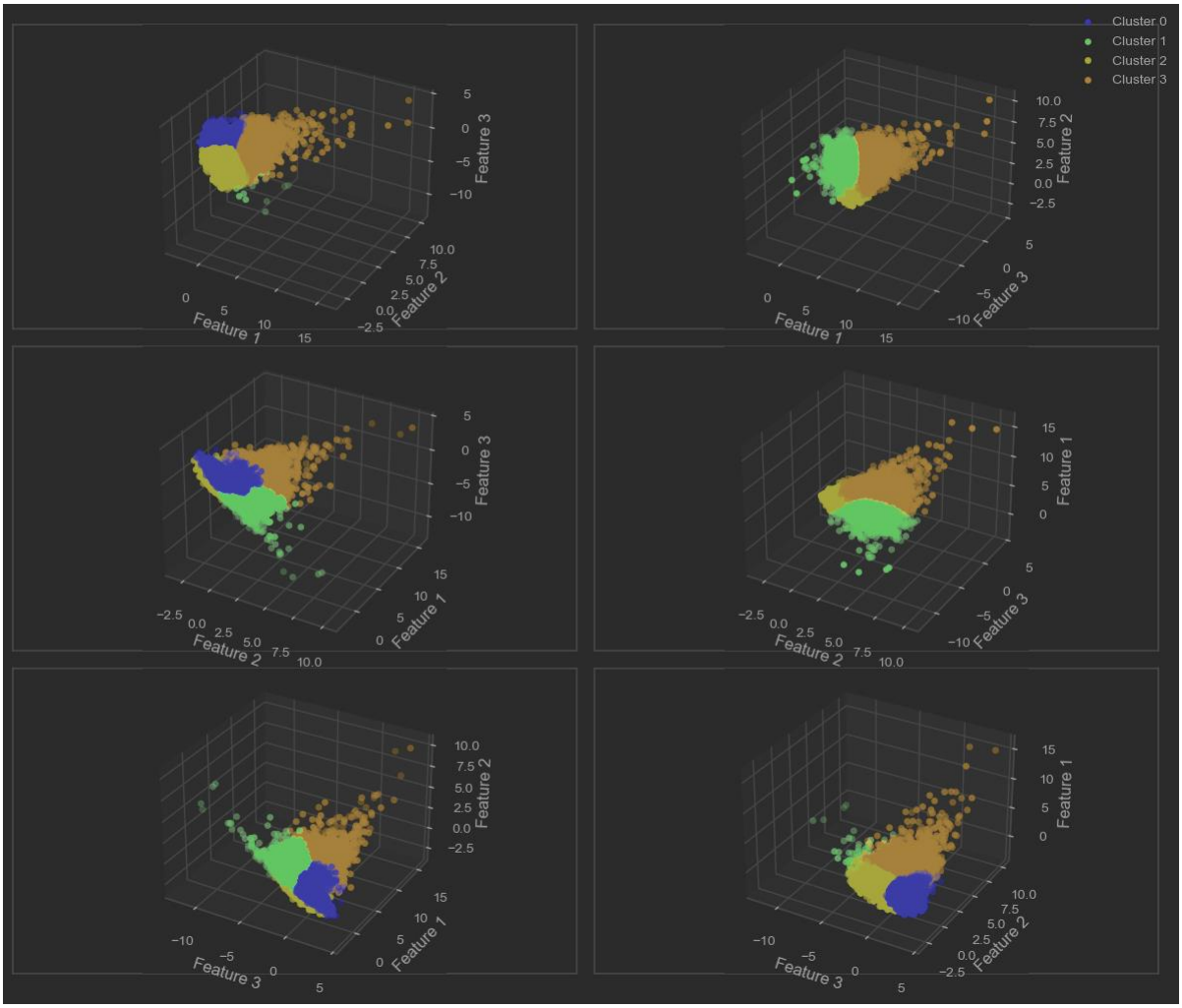

Figura 18: Risultato K-means con dati a 3 dimensioni.

Il grafico della Silhouette è mostrato in Figura 19. Come si può osservare, si ottiene uno score medio vicino allo 0.33, 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 ai cluster 1, 2 e 3, i quali presentano score negativo, a indicare che potevano essere rappresentati meglio da uno altro gruppo di cluster.

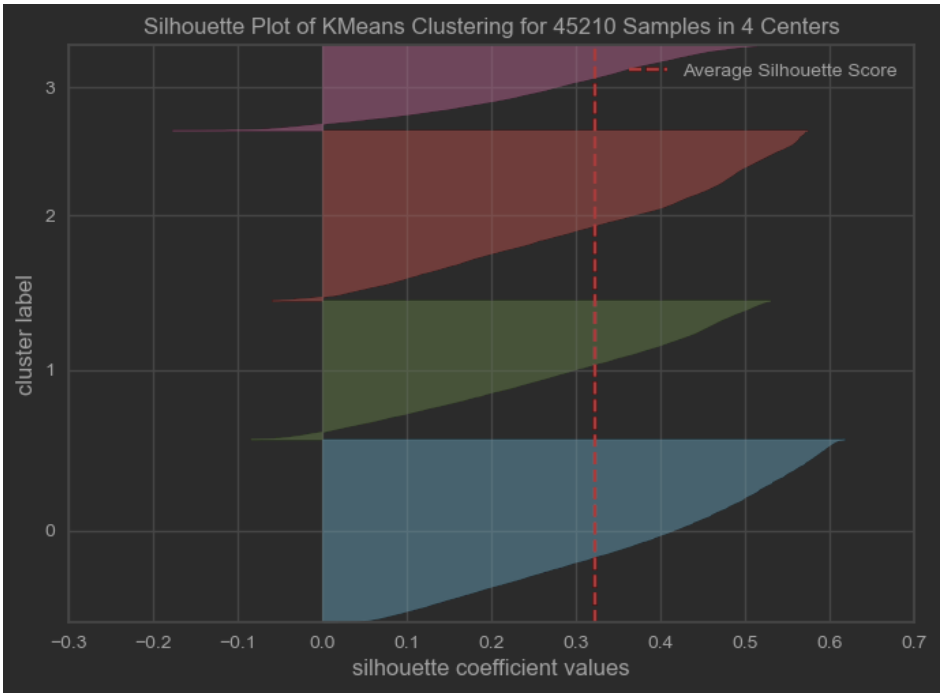

Figura 19: Silhouette K-means con dati a 3 dimensioni.

4.1.2 K-means con dati a 2 dimensioni

Successivamente, è stato provato l'algoritmo K-means sui dati portati a 2 dimensioni dalla PCA. Come nel caso precedente, abbiamo utilizzato il metodo del gomito per poter capire quale fosse il numero ottimale di cluster: come si può vedere nella Figura 20, il gomito questa volta è in corrispondenza del 3, dunque, scegliamo k uguale a 3.

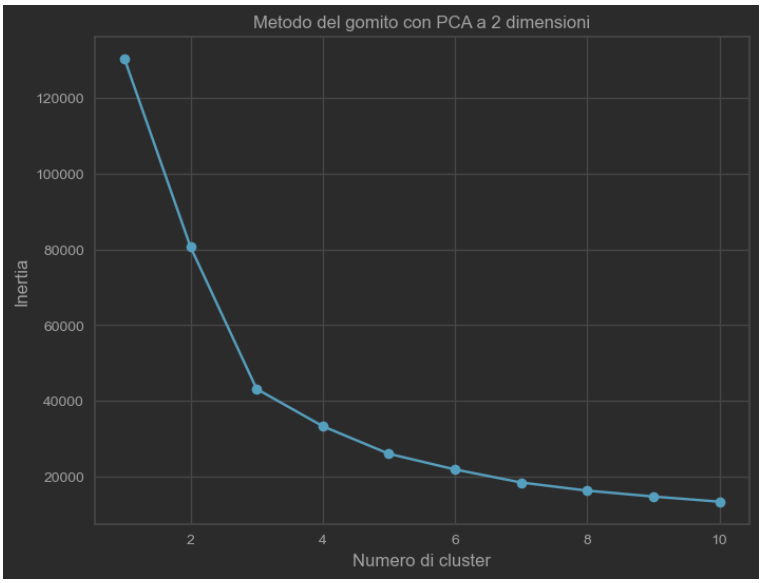

Figura 20: Metodo del gomito per K-means con dati a 2 dimensioni.

Una volta scelto il numero di cluster ottimale, è stato possibile utilizzare l'algoritmo, che ha restituito i 3 cluster riportati in Figura 21.

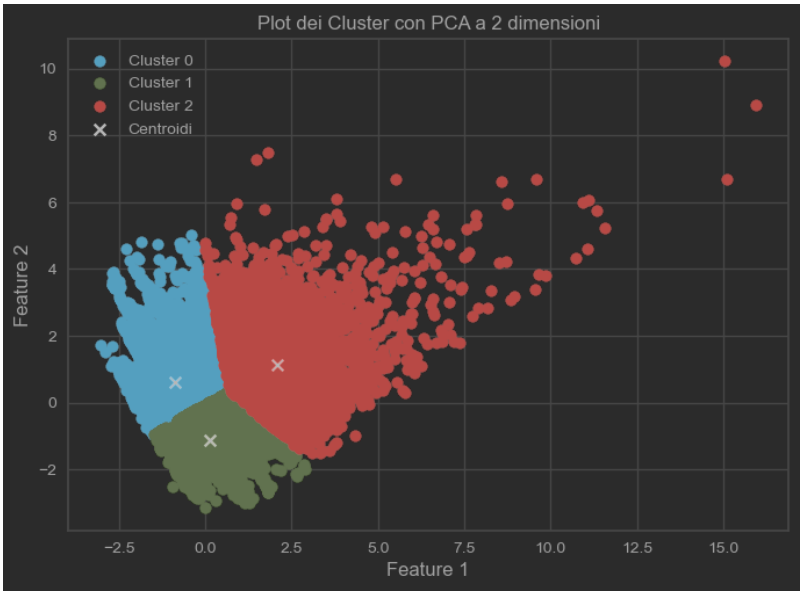

Figura 21: Risultato K-means con dati a 2 dimensioni.

Il grafico della Silhouette è mostrato in Figura 22. Come si può osservare, si ottiene uno score medio dello 0.49, 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, il quale presenta score negativi, a indicare che potevano essere rappresentati meglio da uno altro gruppo di cluster.

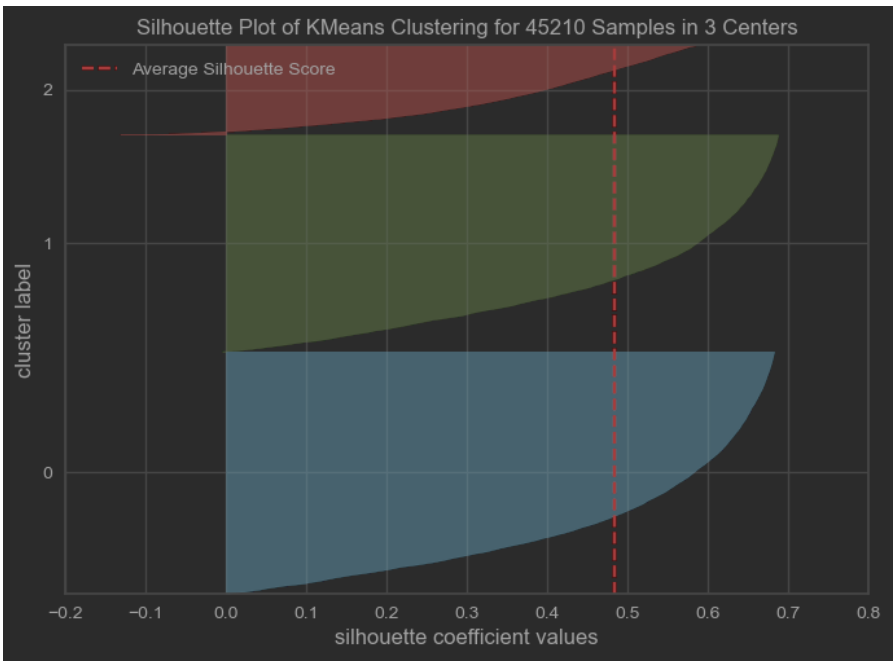

Figura 22: Silhouette K-means con dati a 2 dimensioni.

Il risultato ottenuto riducendo il dataset a 2 feature risulta essere migliore del risultato ottenuto riducendo, tramite PCA, il dataset a 3 feature, sia in termini di valor medio di silhouette, sia in termini di numerosità di elementi con silhouette negativa. Questi sono i motivi per i quali abbiamo deciso di effettuare l'analisi delle caratteristiche dei cluster ottenuti solo di quest'ultimo esperimento.

4.1.2.1 Analisi dei cluster

La prima analisi effettuata sui tre cluster (ottenuti con l'algoritmo K-means applicato ai dati ridotti a dimensionalità 2 tramite PCA) riguarda la numerosità dei cluster. I risultati sono riportati in Figura 23: il cluster 0 risulta essere il meno numeroso, mentre i cluster 1 e 2 risultano essere numericamente maggiori.

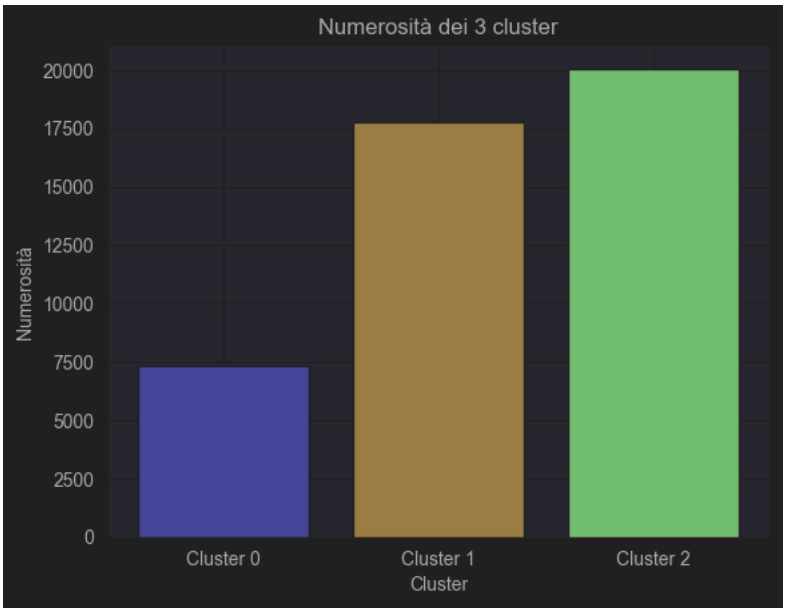

Figura 23: Numerosità dei cluster.

Per quanto riguarda le diverse feature che caratterizzano i cluster, abbiamo scelto di analizzarne le più significative.
Per prima cosa, abbiamo analizzato l'età dei campioni appartenenti ad ogni cluster, il risultato è riportato in Figura 24. Come si può notare dall'immagine, una caratteristica dei campioni appartenenti al cluster 2 è l'età più elevata, contro l'età più bassa che contraddistingue gli elementi appartenenti al cluster 1.

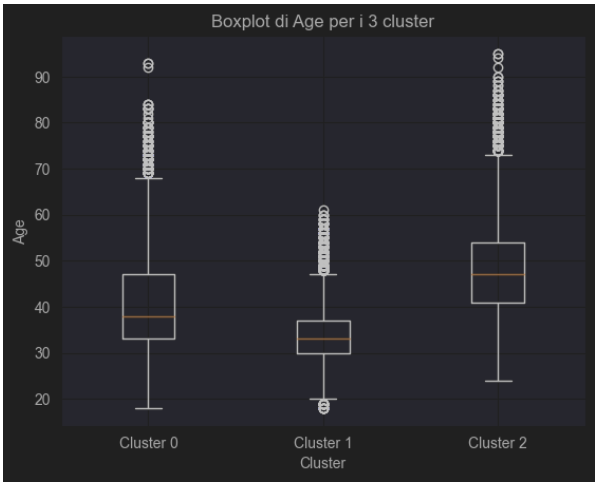

Figura 24: Età dei cluster.

In seconda istanza, abbiamo scelto di analizzare lo stato civile dei diversi soggetti. In particolare, come è possibile osservare nella Figura 25, nel cluster 1 non sono presenti soggetti divorziati, mentre

nel cluster 2 sono quasi assenti soggetti single e la maggior parte di essi è sposata. Il cluster 0 risulta essere un misto tra i tre stati civili possibili.

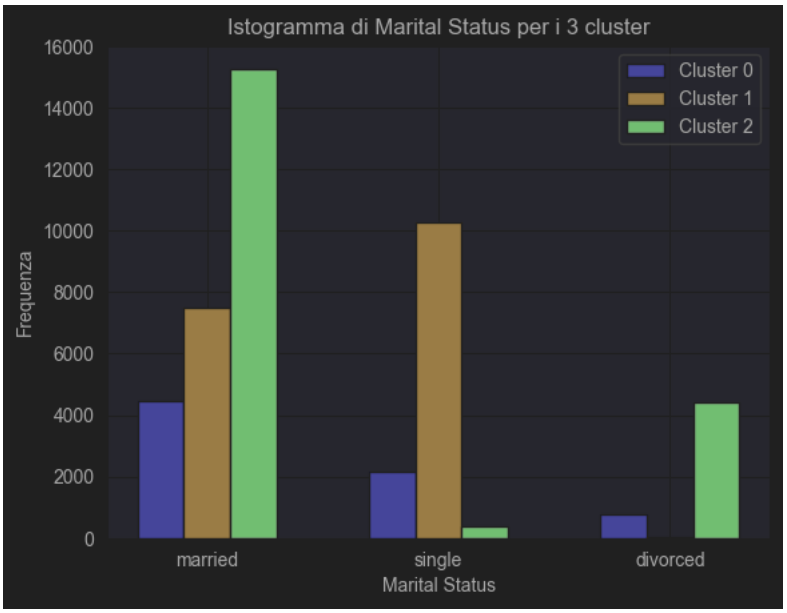

Figura 25: Stato civile nei cluster.

Riguardo le diverse tipologie di impiego dei nostri campioni, mostrate in figura Figura 26, per quanto riguarda il cluster 1 possiamo affermare che non sono presenti oggetti in pensione, le occupazioni più frequenti di questo cluster sono quelli relativi alle etichette “management” e “technician”. Nel cluster 2, invece, troviamo principalmente operai. Infine, nel cluster 0, risulta essere presente un misto di tutte le tipologie di professioni.

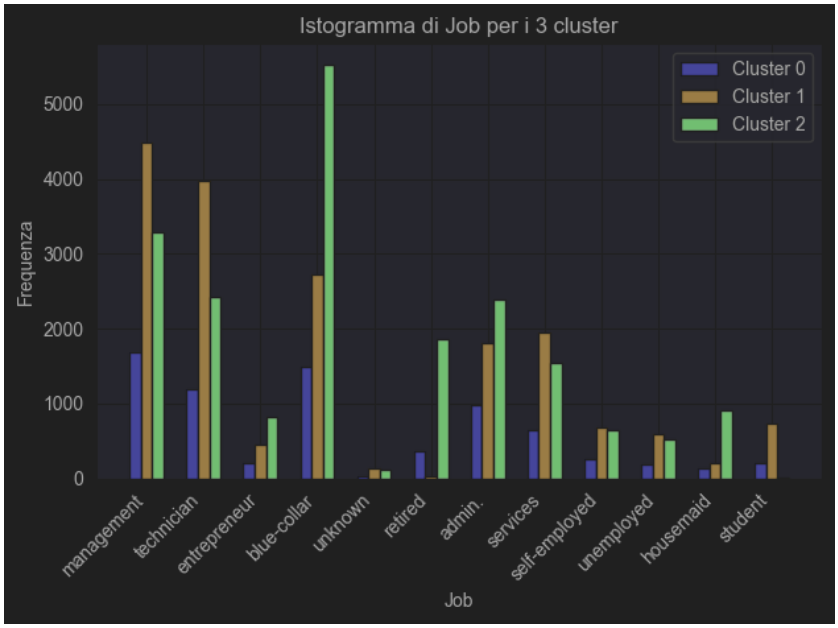

Figura 26: Impiego nei cluster.

Analizzando la feature riguardante l’istruzione dei diversi soggetti, abbiamo notato che nel cluster 1 sono presenti principalmente persone che hanno frequentato superiori ed università, mentre nel cluster 2 spiccano i soggetti che hanno frequentato le superiori. Inoltre, è interessante notare come,

nei soggetti che hanno il livello più basso di istruzione, la maggioranza rappresentata dal cluster 2. Per quanto riguarda il cluster 0, come sempre, risulta essere un compromesso tra gli altri due cluster.

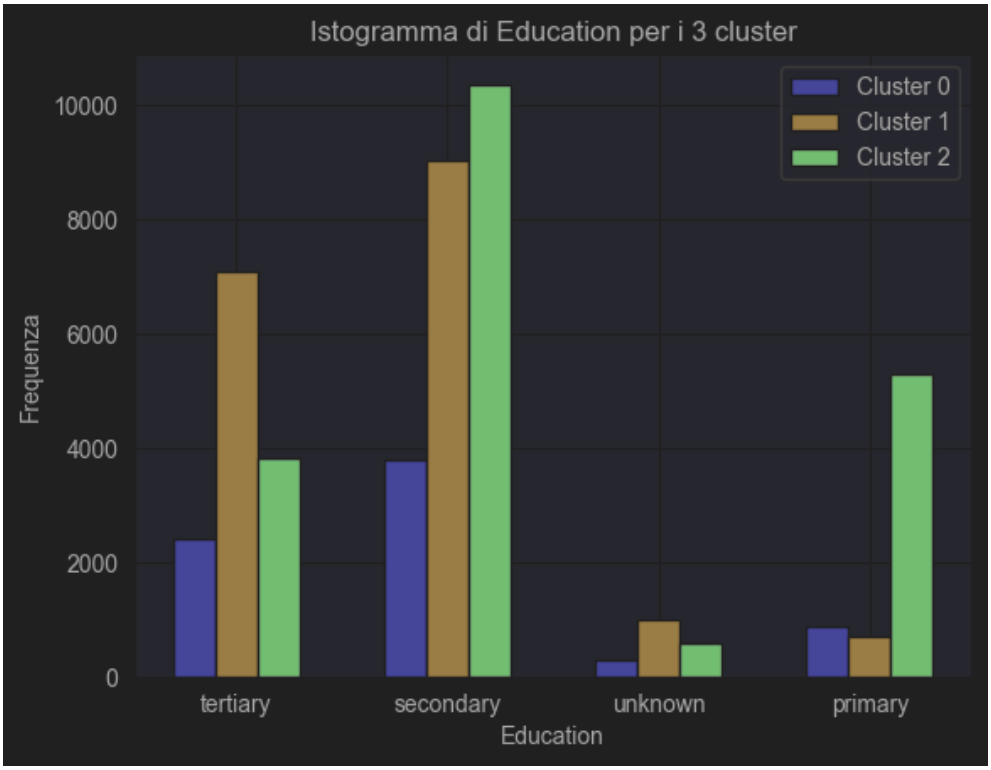

Figura 27: Livello di educazione nei cluster.

Sul debito riguardo i soggetti contattati dalla banca possiamo affermare che, interpretando i grafici riportati in Figura 28, la presenza di mutui personali e di mutui per le case risultano essere non influenti alla clusterizzazione, in quanto non sono presenti cluster caratterizzati dalla presenza di un mutuo e cluster caratterizzati dall'assenza di quest'ultimo, ma tutti i cluster presentano circa lo stesso rapporto tra numero di persone che hanno e non hanno mutui personali e mutui per la casa.

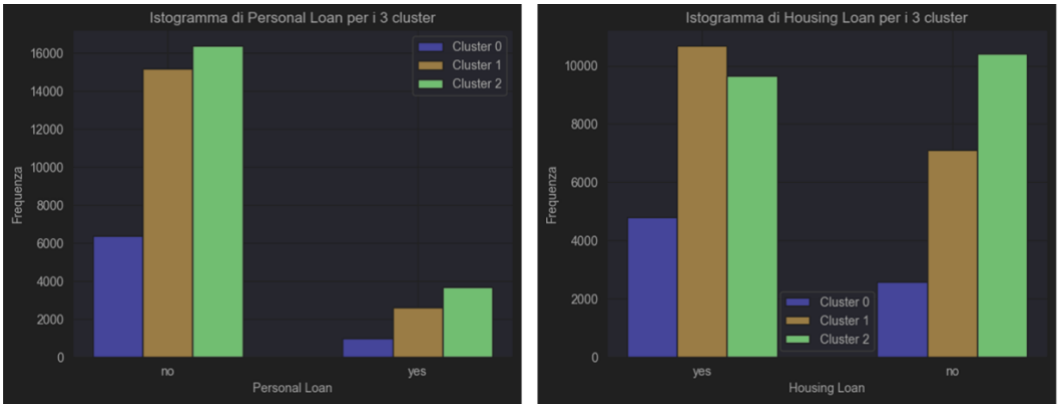

Figura 28: Presenza di mutui personali o per la casa nei cluster.

Infine, l'ultima feature analizzata è "subscription". Dai risultati riportati in Figura 29, si può evincere che in tutti e tre i cluster la maggior parte dei soggetti hanno accettato di sottoscrivere l'offerta per la quale sono stati contattati. Nel cluster 0, però, il rapporto tra il numero di utenti che ha rifiutato ed il numero di utenti che ha accettato l'offerta è molto maggiore rispetto agli altri cluster.

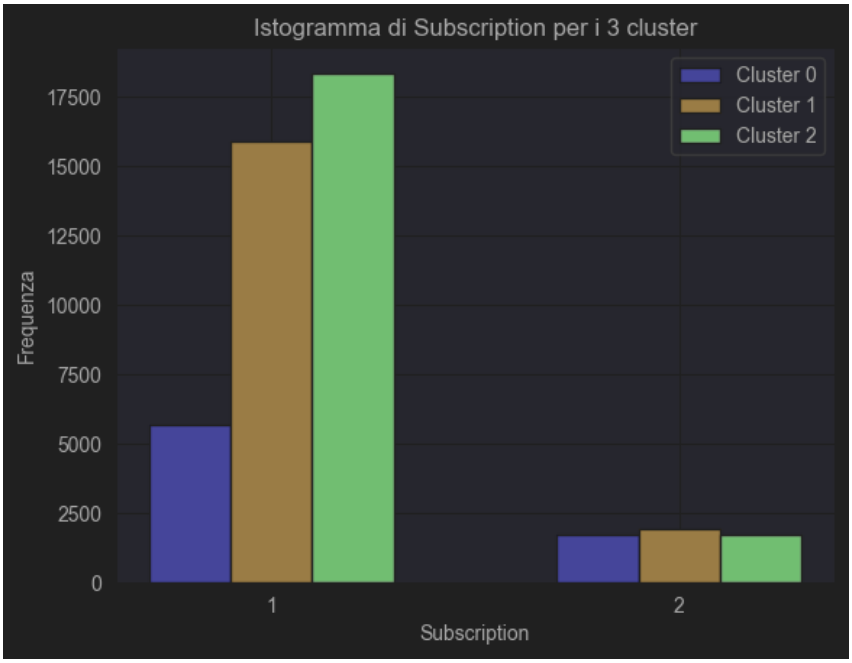

Figura 29: Sottoscrizione o rifiuto dell'offerta nei cluster.

Alla luce di tutto ciò, possiamo assegnare ai diversi cluster le seguenti caratteristiche:

- **Cluster 1**: cluster a cui sono associati i soggetti più giovani, dunque principalmente single/sposati, con un livello di istruzione più alto, e che occupano posizioni lavorative di "alto livello";
- **Cluster 2**: cluster a cui sono associati i clienti più anziani, sposati/divorziati, principalmente operai o comunque soggetti con un livello di istruzione medio/basso;
- **Cluster 0**: cluster che risulta essere un misto tra i due precedenti, ossia clienti di mezza età, distribuiti tra le varie posizioni lavorative, con livello di istruzione medio. Una particolarità di questo cluster è che i soggetti al loro interno sono quelli che risultano essere meno propensi all'accettazione delle offerte.

4.2 Esperimenti con DBSCAN

Il DBSCAN è un algoritmo di clustering che raggruppa i dati in base alla densità degli stessi nello spazio delle feature. L'algoritmo è composto da diversi passi:

1. si seleziona un punto casuale non ancora assegnato a un cluster (punto di partenza);
2. si identificano tutti i punti nel vicinato di questo punto, definito da un raggio ε.
3. se il numero di punti nel vicinato supera una soglia minima predefinita (MinPts), il punto è considerato un "punto centrale" e quindi parte di un cluster;
4. se un punto è un "punto centrale", tutti i punti nel suo vicinato diventano parte del suo stesso cluster;
5. questo processo si ripete ricorsivamente fino a quando non si esauriscono tutti i punti vicini;
6. i punti che non soddisfano i criteri di densità (non sono "punti centrali" e non appartengono al vicinato di alcun "punto centrale") vengono considerati rumore e non assegnati a nessun cluster.

L’algoritmo DBSCAN, a differenza del K-means, è in grado di identificare cluster di forma arbitraria e può gestire efficacemente cluster di diverse forme e dimensioni.
Tuttavia, richiede una buona scelta dei parametri, come il raggio ε e il numero minimo di punti MinPts, che possono variare a seconda delle caratteristiche dei dati e dei requisiti dell'applicazione.

4.2.1 DBSCAN con 3 dimensioni

Come detto in precedenza nella sezione 4.2, il DBSCAN prende in input due parametri fondamentali: ε e MinPts.
Per la scelta del parametro ε ottimale, abbiamo scelto di utilizzare il NearestNeighbors: ciò che viene fatto è calcolare la distanza media tra i punti utilizzando un raggio di ricerca crescente, per poi tracciare un grafico della distanza media rispetto al raggio di ricerca. Si cerca il punto in cui la curva presenta un "gomito", indicando un cambiamento significativo nella distanza media tra i punti. Questo valore di distanza corrisponderà a un valore ragionevole per ε in DBSCAN.

Abbiamo testato il NearestNeighbors con diversi valori di numero di vicini, all’inizio bassi (sull’ordine delle decine), riscontrando che i valori di ε che venivano fuori erano molto piccoli e molto simili tra di loro. Successivamente, abbiamo testato il metodo con valori di numeri di vicini più alti (sull’ordine delle centinaia), per riscontrare, alla fine dei conti, lo stesso risultato. Infine, abbiamo scelto di provare con range di numeri di vicini ancora più alti (sull’ordine delle migliaia) per capire che non era possibile avere un ε ottimale, in quanto il gomito che corrisponde al valore di epsilon ottimale non è presente nel nostro grafico. Nella Tabella 3 vengono riportate le diverse distanze medie calcolate.

Num Neighbors	Avg Distances
100	0.06203444
5100	0.06203444
10000	0.06203444
15000	0.06203444
20000	0.06203444
25000	0.06203444

Tabella 3: Distanze medie in base al numero di vicini considerati.

Abbiamo ottenuto questo risultato perché i dati che abbiamo a disposizione sono molto densi, dunque, cercare di trovare diversi cluster con un algoritmo density-based come DBSCAN su un dataset di questo tipo non è producente.
Per completezza, riportiamo il risultato del clustering effettuato con l’algoritmo DBSCAN sul dataset ridotto a 3 dimensioni, al quale abbiamo passato come argomenti ε = 0.9 e min_pts = 50, scelti arbitrariamente.
I risultati sono riportati in Figura 30: come nel caso del K-means a 3 dimensioni, per una totale rappresentazione dei cluster, vengono riportate tutte le possibili viste a 3 dimensioni.

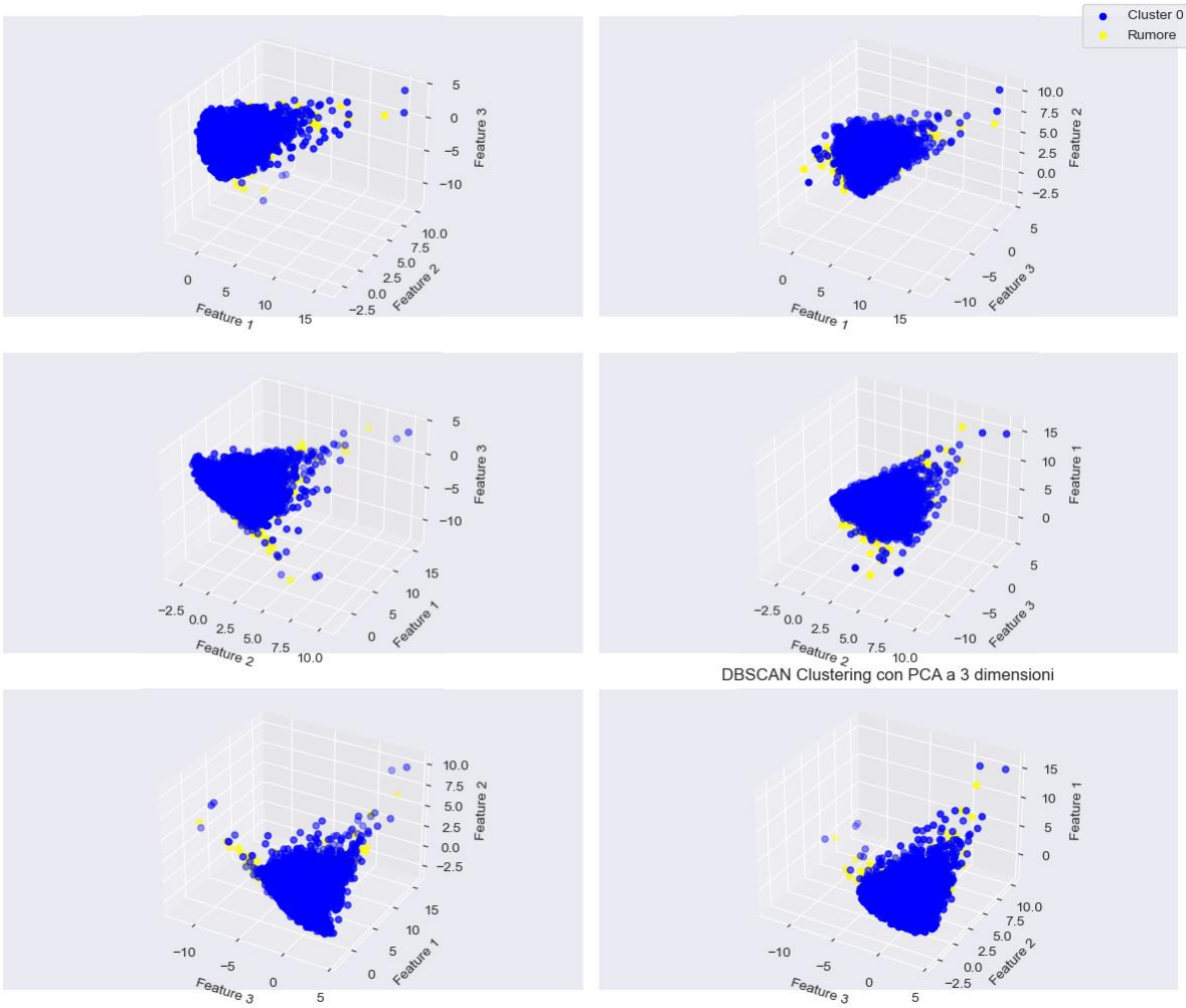

Figura 30: Risultato DBSCAN con dati a 3 dimensioni.

È importante notare come il DBSCAN riesce a trovare un solo cluster, prendendo come rumore i punti che non risultano essere nell'area densa: ciò è dovuto proprio al fatto che è stato scelto un algoritmo non adatto per il tipo di dati presenti nel dataset.

4.2.2 DBSCAN con 2 dimensioni

Lo stesso discorso fatto nella sezione 4.2.1, vale per il DBSCAN applicato ai dati ridotti a 2 feature: i valori delle distanze medie risultano essere costanti, come è possibile vedere in Tabella 4, dunque il picco non è presente e non è stato possibile scegliere un epsilon ottimale.

Num Neighbors	Avg Distances
100	0.012767077
5000	0.012767077
10000	0.012767077
15000	0.012767077
20000	0.012767077
30000	0.012767077
40000	0.012767077

Tabella 4: Distanze medie rispetto al numero di vicini considerati.

Come nel caso precedente, abbiamo settato arbitrariamente i due parametri ε con 0.6 e min_pts con 50.
I risultati sono riportati in Figura 31.

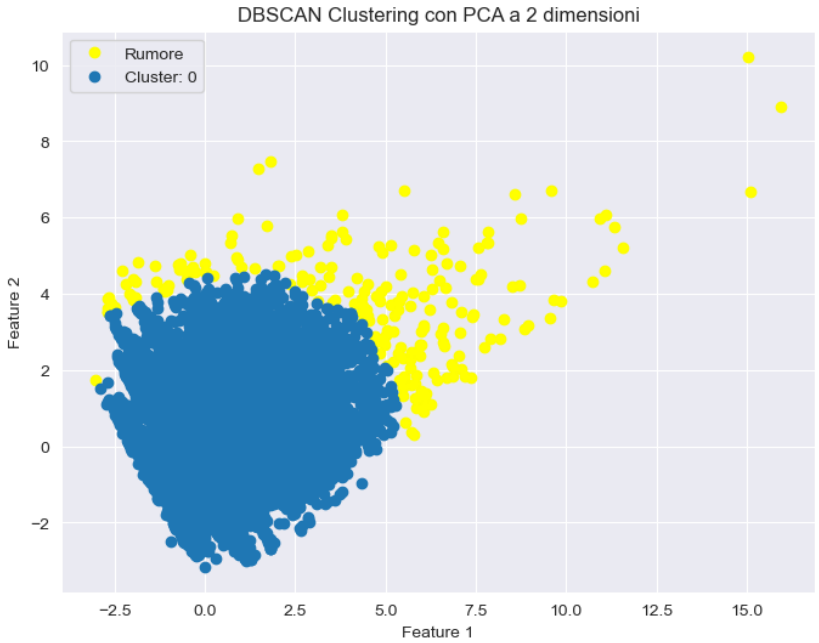

Figura 31: Risultato DBSCAN a 2 dimensioni.

Come nel caso precedente, l'algoritmo, a causa dei dati molto fitti, non riesce a trovare pattern tra i dati, restituendo un solo cluster che corrisponde alla parte più densa e considerando i punti al di fuori di questa come rumore.

5 Forecasting

Il forecasting è uno strumento che fornisce una visione proattiva del futuro; esso si basa sull'analisi dei dati storici per individuare modelli e tendenze che possono essere proiettati nel tempo.
Per fare previsioni, vengono impiegati modelli matematici e statistici che aiutano a interpretare i dati e a identificare schemi ricorrenti. Questi modelli sono spesso addestrati utilizzando dati storici e poi testati su dati non visti per valutarne l'accuratezza.
Il forecasting ha una vasta gamma di campi di applicazione in cui riveste un ruolo cruciale per guidare decisioni e pianificazione. Tra i principali settori, possiamo citare:

- **Economia e Finanza**: il forecasting è molto utilizzato per prevedere i movimenti dei mercati azionari, dei tassi di interesse, dei tassi di cambio e altro ancora. Inoltre, le aziende utilizzano il forecasting per prevedere le vendite future, identificare trend di mercato, valutare l'efficacia delle campagne pubblicitarie e pianificare le strategie di marketing.
- **Scienza e Tecnologia**: Nel campo della ricerca scientifica e tecnologica, il forecasting è utilizzato per prevedere lo sviluppo di nuove tecnologie, l'andamento delle epidemie e delle pandemie, il cambiamento climatico e altro ancora.
- **Meteorologia**: La previsione del tempo è uno dei campi più noti di forecasting. Utilizzando dati storici e modelli meteorologici complessi, gli scienziati possono fare previsioni sul tempo futuro per aiutare le persone e le aziende a prepararsi a eventi meteorologici estremi.

Per l'implementazione del modello di forecasting è stata utilizzata la libreria Statsmodel.
In questo capitolo verranno descritte tutte le soluzioni da noi adottate per fare previsioni sui prezzi di diversi indici: particolarmente enfasi verrà posta sulla predizione dell'indice dell'oro, mentre degli altri indici verranno riportati solo i risultati.

5.1 ETL e analisi descrittiva

Il dataset utilizzato per la classificazione è quello descritto nel paragrafo 2.2 contenente i dati relativi all'andamento temporale degli indici di Oro, Argento, Barrick Gold Corp e S&P 500.
Per l'ETL è stata utilizzata la libreria Pandas di Python.
La prima cosa fatta è stata leggere i 4 file (uno per ciascun indice) e caricare il loro contenuto in 4 diversi dataframe. I dati utili ai fini dell'analisi sono il valore in dollari di ciascun indice e le relative date (si è scelto di prendere come riferimento il periodo 21/08/2017 e 21/08/2020).
Per poter inserire i dati all'interno di un unico dataframe, si è scelto di utilizzare la colonna "Date" come indice, per poi avere solo quattro colonne, ciascuna contenente il valore in dollari dei diversi indici.
Nella Figura 32 sono riportate le prime cinque righe del dataframe totale.

	GLD	SPX	BARR	SLV
Date				
2017-08-22	1291.0	2452.51	16.5072	17.060
2017-08-23	1294.7	2444.04	16.6639	17.126
2017-08-24	1292.0	2438.97	16.6932	17.046
2017-08-25	1297.9	2443.05	16.7814	17.132
2017-08-28	1315.3	2444.24	17.3003	17.529

Figura 32: Prime cinque righe del dataframe totale.

Una volta ottenuto il dataframe totale, essendo i dati provenienti da quattro file diversi e avendo utilizzato la data come indice, è stato necessario un controllo sulla presenza di valori di tipo nullo. Nella Figura 33 è riportato il conteggio dei valori di tipo nullo nel dataframe.

```
GLD     0
SPX    16
BARR   16
SLV    11
```

Figura 33: Conteggio valori nulli nel dataframe totale.

La presenza di valori nulli rende necessario il rimpiazzamento di tali valori: per fare ciò e per cercare di modificare il meno possibile il dataframe, abbiamo scelto di tener d'occhio le caratteristiche principali di quest'ultimo, come media, deviazione standard, massimo e minimo, prima e dopo l'utilizzo della funzione "fillna".
Come si può notare nella Figura 34, le differenze tra le caratteristiche del dataframe prima e dopo il rimpiazzamento dei valori nulli sono di fatto impercettibili: dunque, il dataframe risulta essere pronto per le previsioni.

```
maindf.describe()
```

	GLD	SPX	BARR	SLV
count	759.000000	743.000000	743.000000	748.000000
mean	1405.848419	2846.261023	16.046517	16.575697
std	186.315921	226.333878	4.703624	2.132119
min	1176.700000	2237.400000	9.666000	11.772000
25%	1281.300000	2688.700000	12.871200	15.242500
50%	1324.950000	2821.930000	14.123100	16.463300
75%	1503.750000	2978.735000	17.760450	17.279250
max	2061.500000	3397.160000	30.130000	29.261000

```
maindf = maindf.fillna(axis=0, method='ffill')
maindf.describe()
```

	GLD	SPX	BARR	SLV
count	759.000000	759.000000	759.000000	759.000000
mean	1405.848419	2846.759908	16.037577	16.573859
std	186.315921	227.016344	4.687174	2.123482
min	1176.700000	2237.400000	9.666000	11.772000
25%	1281.300000	2688.700000	12.882800	15.244900
50%	1324.950000	2820.400000	14.123100	16.460600
75%	1503.750000	2979.075000	17.760900	17.282500
max	2061.500000	3397.160000	30.130000	29.261000

Figura 34: Confronto tra caratteristiche del dataframe prima (sinistra) e dopo (destra) il rimpiazzamento dei valori nulli.

Prima di procedere col forecasting, avendo ora a disposizione il dataframe sul quale lavorare, abbiamo deciso di studiare l'andamento delle diverse serie temporali. I diversi andamenti sono riportati nella Figura 35.

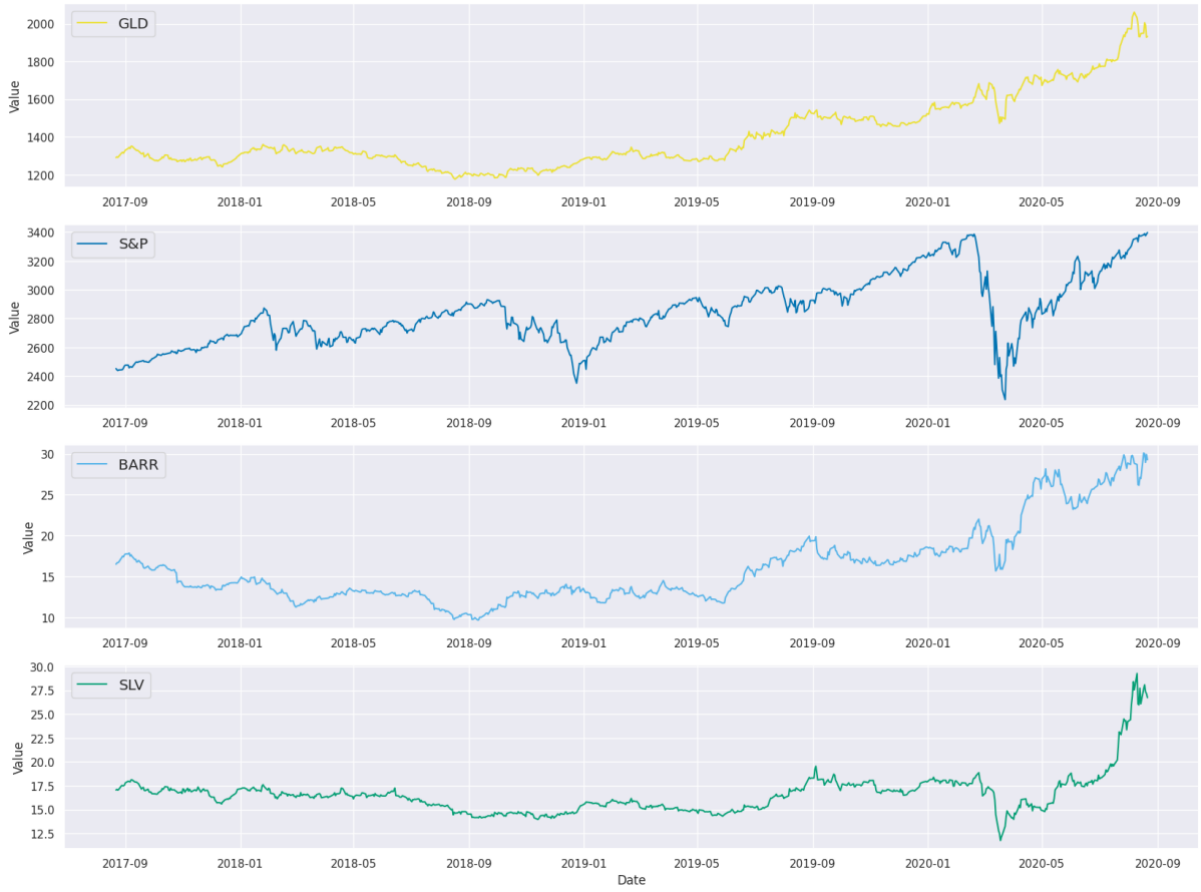

Figura 35: Andamento temporale dei diversi indici.

Dalla rappresentazione grafica degli andamenti nel tempo dei diversi indici, è ben visibile l'effetto che ha avuto il primo lockdown causato dal covid (periodo fine gennaio-inizio febbraio del 2020): tutti gli indici hanno subito un forte calo, ad eccezione dell'oro. È importante notare anche come, nel periodo che va da aprile 2020 in poi, gli effetti del lockdown vanno scemando, facendo impennare tutti gli indici analizzati.

5.2 Forecasting prezzi oro con modello ARIMA

Inizialmente, abbiamo scelto di realizzare le previsioni utilizzando il modello ARIMA.
Il modello ARIMA (AutoRegressive Integrated Moving Average) è un modello composto da tre componenti principali:

- Componente autoregressiva (AR), che cattura le relazioni lineari tra le osservazioni passate e l'osservazione corrente;
- Componente di media mobile (MA), che tiene conto degli errori residui del modello. Si basa sull'idea che ci siano correlazioni tra l'errore attuale e gli errori passati.
- Differenziazione integrata (I), che si occupa della differenziazione della serie temporale per renderla stazionaria. In particolare, una serie stazionaria ha una media e una varianza costanti nel tempo, semplificando così la modellazione.

Il modello ARIMA è definito tramite tre parametri principali: p, d e q, che rappresentano rispettivamente l'ordine dell'auto regressione, l'ordine della differenziazione integrata e l'ordine

della media mobile. La scelta di questi parametri dipende dall'analisi della serie temporale e può essere determinata utilizzando tecniche come l'autocorrelazione e l'autocorrelazione parziale.
Prima di tutto, abbiamo a disposizione un dataframe di dimensione (759, 4): dunque abbiamo scelto di dividerlo in due dataframe diversi, uno di train di dimensione (700, 4) e uno di test di dimensione (59, 4).
Sul dataframe di train, abbiamo effettuato diverse prove per scegliere i parametri ottimali.
Per prima cosa, abbiamo calcolato il p-value del dataframe di train contenente solo i dati riguardanti l'oro. Il p-value calcolato risulta essere 0.93: essendo minore di 0.05, la serie è considerata non stazionaria.
Dovendo lavorare con serie stazionarie, risulta necessario differenziare il dataframe. Utilizzando le funzioni di libreria, abbiamo differenziato e calcolato nuovamente il p-value; questa volta risulta essere 1.97e-09, dunque minore di 0.05. La serie differenziata risulta essere stazionaria. In questo modo abbiamo a disposizione già il parametro d, che dovrà essere 1, in quanto corrisponde al numero di differenziazioni necessarie a rendere la serie stazionaria.
Per poter calcolare i parametri p e q, studiamo i grafici di autocorrelazione e autocorrelazione parziale del dataframe di train: il parametro p si ricava contando il numero di valori al di fuori del limite di significatività nel grafico di autocorrelazione (riportato in Figura 36), mentre il parametro q si ricava nello stesso modo ma prendendo in considerazione il grafico di autocorrelazione parziale (riportato in Figura 37).

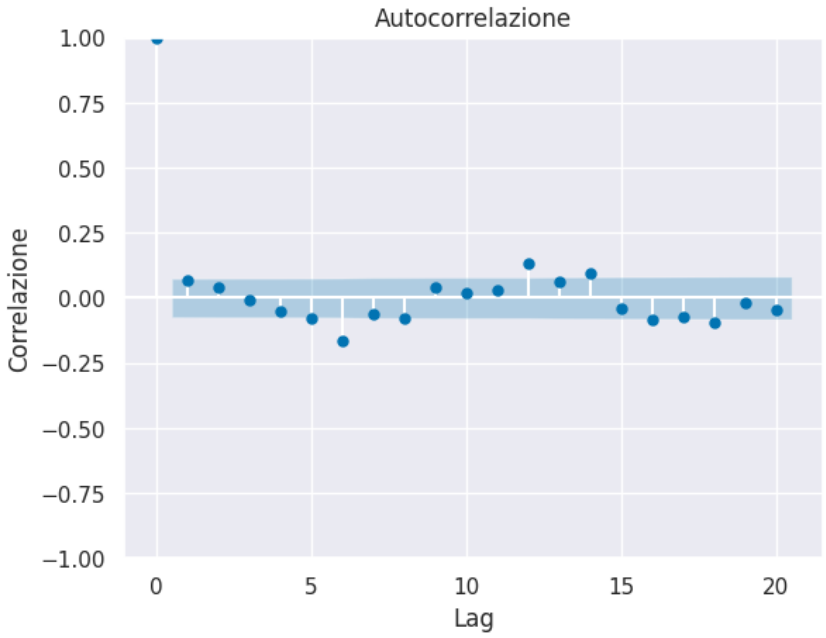

Figura 36: Grafico autocorrelazione.

Nel grafico di autocorrelazione, contiamo il numero di valori al di fuori del limite di significatività: il parametro p sarà 4.

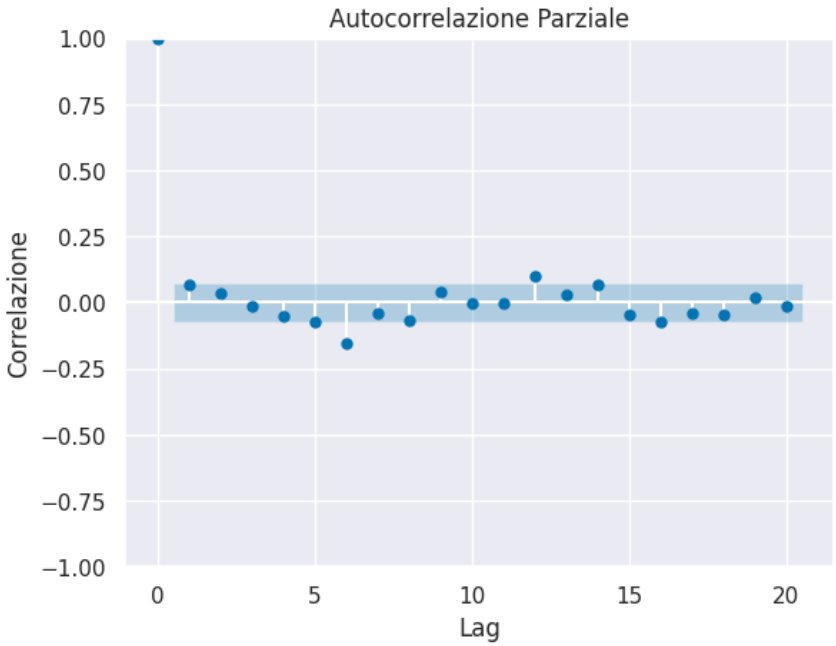

Figura 37: Grafico autocorrelazione parziale.

Nel grafico di autocorrelazione parziale, contiamo il numero di valori al di fuori del limite di significatività: il parametro q sarà 2.
Una volta settati i parametri p, d e q, è possibile addestrare il modello.
In Figura 38 è riportato il risultato dell'addestramento del modello ARIMA con la terna di parametri (4,1,2).

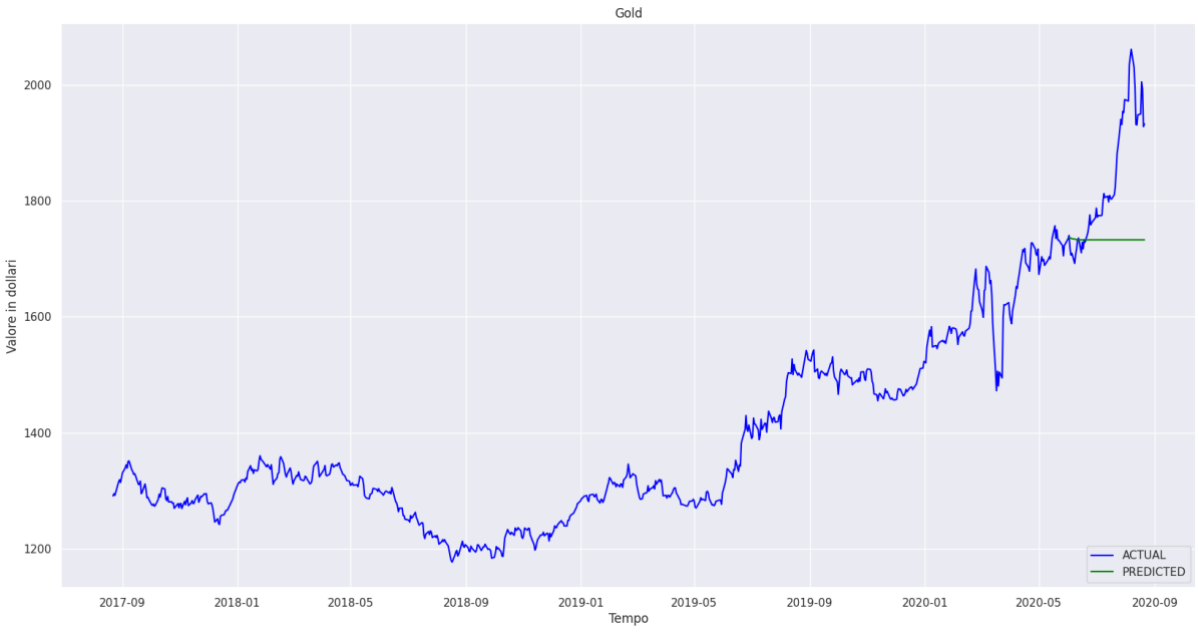

Figura 38: Confronto tra serie temporale reale e previsioni oro con modello ARIMA.

Possiamo notare, dai risultati, che il modello non riesce a predire il prezzo dell'oro.
Un'ulteriore prova effettuata è stata, dunque, quella di utilizzare la funzione auto_arima della libreria pmdarina sul dataframe di train per verificare che i parametri scelti siano effettivamente

quelli ottimali: abbiamo notato come, effettivamente, i parametri da noi scelti non risultavano essere quelli ottimali. Infatti, la funzione restituisce come parametri ottimali p, d e q rispettivamente 2, 1 e 2.
È importante specificare che è possibile che i valori di p e q identificati tramite l'analisi dei grafici di autocorrelazione e autocorrelazione parziale siano diversi da quelli ottenuti utilizzando la funzione auto_arima o altri metodi automatizzati di selezione del modello, poichè l'analisi visiva dei grafici può fornire solo indicazioni approssimative sui valori ottimali di p e q, mentre i metodi automatizzati utilizzano algoritmi più sofisticati per trovare il modello migliore in base a criteri statistici come AIC o BIC.
I risultati delle previsioni del modello ARIMA con i parametri ottimali sono riportati in Figura 39.

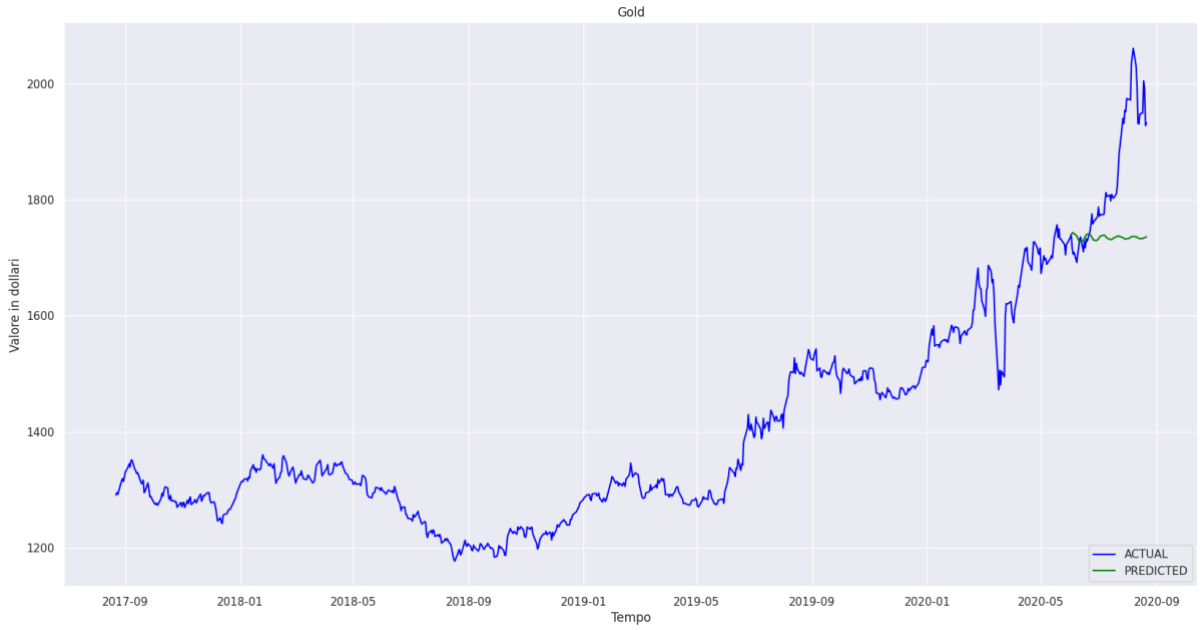

Figura 39: Confronto tra serie temporale reale e previsioni oro con modello ARIMA (parametri ottimali).

Neanche con i parametri ottimali, purtroppo, è stato possibile arrivare a risultati soddisfacenti.

5.3 Forecasting prezzo oro con modello SARIMAX

A questo punto, un'ulteriore possibile soluzione era quella di ricorrere alle variabili esogene, avendo a disposizione l'andamento temporale di ben quattro indici nello stesso periodo: importante è considerare che le serie risultano essere correlate, in quanto si prende in considerazione il periodo Covid, che caratterizza tutte le serie con picchi e depressioni negli stessi mesi.

Il modello SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors) è un'estensione del modello ARIMA che tiene conto delle componenti stagionali nei dati e consente l'inclusione di variabili esogene nel modello.
Come nell'ARIMA classico, il modello SARIMAX comprende le stesse tre componenti principali, ai quali si vanno ad aggiungere:

- **Componenti stagionali**, che possono essere simili alle componenti ARIMA, ma applicate su periodi stagionali specifici (come settimane, mesi, trimestri, ecc.), consentendo di catturare i modelli ciclici nel tempo;
- **Variabili esogene**: fattori esterni che possono influenzare la serie temporale ma non sono direttamente modellati dalla serie stessa. Queste variabili possono fornire informazioni aggiuntive al modello per migliorare le previsioni.

Come per il modello ARIMA, SARIMAX prende in input la stessa terna di variabili del modello ARIMA p,d e q.
Per la scelta di questi, non esistendo una funzione per il calcolo dei parametri ottimali che tenga in conto anche le variabili esogene, abbiamo optato per una Grid Search: sostanzialmente, abbiamo calcolato il RMSE di più modelli, ognuno ei quali aveva una terna di parametri diversa.
Le diverse terne di parametri testate sono tutte le possibili combinazoni ordinate degli elementi delle liste python riportate nel seguente frammento di codice.

```
p_values = [0, 1, 2, 3, 4, 5, 6, 7]
d_values = [1]
q_values = [0, 1, 2, 3, 4, 5, 6, 7]
```

La terna di parametri ottimali, tenuto conto anche della presenza di variabili esogene, risulta essere (2, 1, 6). Il riusltato di tale modello è riportato in Figura 40, e risulta avere un RMSE di 33.19.

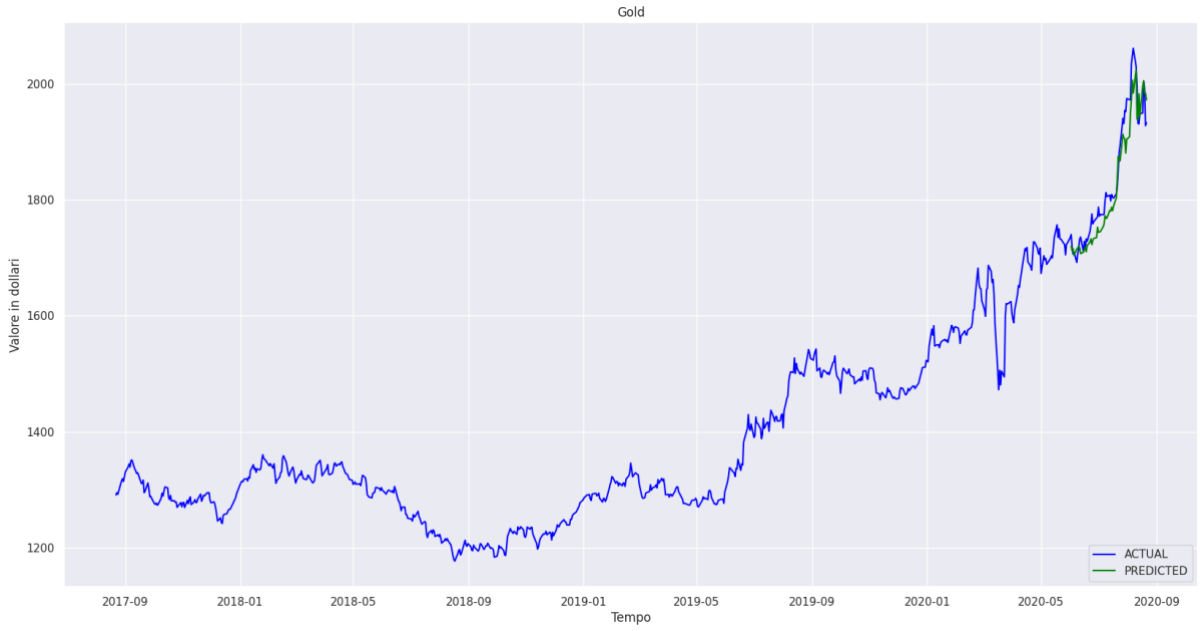

Figura 40: Confronto tra serie temporale reale e previsioni oro con modello SARIMAX.

Nella Figura 41, invece, è riportato uno zoom dei risultati sul periodo di interesse.

Figura 41: Zoom del confronto tra serie temporale reale e previsioni oro con modello SARIMAX.

I risultati risultano essere nettamente migliori, ciò a prova del fatto che effettivamente le serie sono correlate, come dedotto in principio.
In realtà, osservando meglio gli andamenti temporali riportati nel capitolo 5.1, abbiamo ipotizzato che tra i quattro indici, quello che risulta avere un comportamento "più anomalo" o comunque "meno correlato" rispetto agli altri risulta essere S&P: abbiamo provato a stimare il prezzo dell'oro considerando come variabili esogene solo il prezzo di Barrick e Silver. I risultati di questa prova sono riportati in Figura 42.

Figura 42: Risultato previsioni oro con modello SARIMAX (senza S&P come variabile esogena).

Il modello, in questo modo, riesce a prevedere meglio il prezzo dell'oro; infatti, il RMSE ora risulta essere di 32.25, continuando a predire picchi e depressioni effettivamente presenti nella serie reale.

5.4 Forecasting prezzo argento con modello SARIMAX

Allo stesso modo, abbiamo cercato di predire il prezzo dell'argento, questa volta provando direttamente con il modello SARIMAX.
La terna di parametri è stata ricavata con la stessa Grid Search illustrata nel capitolo precedente, considerando, però, come variabili esogene il prezzo di oro, barrick e S&P: quella ottimale risulta essere (7, 1, 5).
I risultati ottenuti sono riportati nella Figura 43: il modello ha un RMSE di 2.39.

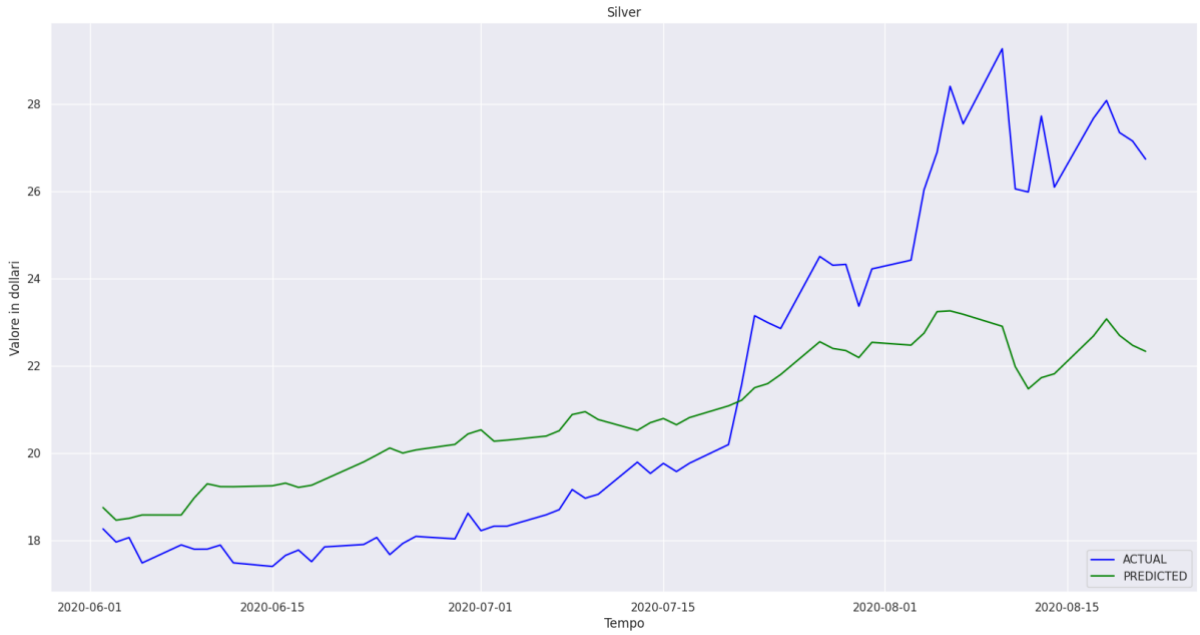

Figura 43: Risultato previsione argento con modello SARIMAX.

I risultati del modello non sono soddisfacenti come nel caso precedente.

5.5 Forecasting prezzo Barrick con modello SARIMAX

Per quanto riguarda il prezzo di Barrick, la terna ottimale, considerando i prezzi di S&P, oro e argento, risulta essere (2, 1, 2). Le previsioni del modello addestrato con tali parametri è riportato nella Figura 44. Il RMSE, questa volta, è di 1.92.

Figura 44: Risultato previsione Barrick con modello SARIMAX.

Questa volta, i risultati non sono molto soddisfacenti: nel periodo 01/08/2020-15/08/2020 l'andamento reale risulta avere un calo, mentre il modello predice un picco.
A questo punto, abbiamo pensato di considerare come variabile esogena soltanto il prezzo dell'argento, in quanto le serie temporali di Barrick e Argento risultano avere gli andamenti più simili.
I risultati di questa prova sono riportati in Figura 45: la terna di parametri scelta tramite la Grid Search questa volta è (3, 1, 4).

Figura 45: Risultato previsioni Barrick con modello SARIMAX (solo argento come variabile esogena).

Il RMSE è leggermente migliorato (1.85), ma il problema rimane lo stesso del caso precedente.

5.6 Forecasting prezzo S&P 500 con modello SARIMAX

Come ultima analisi, abbiamo cercato di predire il prezzo di S&P500 considerando i prezzi di tutti gli altri indici come variabili esogene: la terna di parametri ottimali restituita dalla Grid Search è (0, 1, 1), il RMSE è di 82.19. Il risultato è riportato nella Figura 46.

Figura 46: Risultato previsione S&P con modello SARIMAX.

Osservando i risultati, deduciamo che il modello riesce a predire i prezzi se consideriamo una certa fascia di tolleranza però non riesce a predire i picchi o i cali presenti nella serie temporale reale.