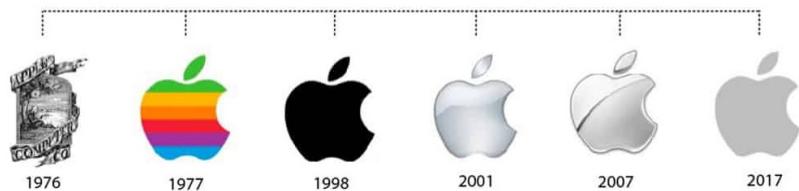


UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione



PROGETTO DI DATA SCIENCE

Q Tecniche di Machine Learning per l'Analisi di Dati Sportivi e Finanziari: Studio su Partite ATP e Azioni Apple



Docenti

Ursino Domenico,
Marchetti Michele

A cura di

De Grazia Davide,
Piergallini Enrico,
Visi Andrea

1 DATASET	1
1.1 La scelta dei dataset	1
1.1.1 Dataset delle partite di tennis ATP (2004-2024)	1
1.1.2 Dataset delle azioni Apple (1984-2024)	2
2 ANALISI DESCRITTIVA	3
2.1 Introduzione	3
2.2 Librerie utilizzate	3
2.3 Risultati dell'analisi	4
2.4 Correlazioni tra le variabili del dataset	11
2.4.1 Matrice di correlazione di Pearson	11
2.4.2 Matrice di correlazione di Spearman	12
3 CLASSIFICAZIONE	13
3.1 Introduzione ai Task di Classificazione sviluppati	13
3.1.1 Passaggi comuni nei diversi task	13
3.2 T1 - Classificare il vincitore di una partita disputata tra due giocatori	14
3.2.1 Preparazione dei dati per la classificazione del vincitore	14
3.2.2 Preparazione del dataset per l'addestramento	17
3.2.3 Addestramento e valutazione dei modelli di classificazione	18
3.3 T2 - Classificare il risultato di una partita	21
3.3.1 Preparazione dei dati	21
3.3.2 Preparazione del dataset per l'addestramento	25
3.3.3 Addestramento e valutazione modelli	26
3.3.4 Risultati e osservazioni:	27
3.4 T3 - Classificare la categoria di un torneo	30
3.4.1 Filtraggio e Preprocessing del Dataset	30
3.4.2 Bilanciamento delle classi per categoria di torneo	31
3.4.3 Valutazione dei modelli	31
3.5 T4 - Classificare un giocatore in base alla sua superficie preferita	33
3.5.1 Preparazione dei dati	33
3.5.2 Generazione e filtraggio del dataframe finale dei giocatori	34
3.5.3 Calcolo della superficie preferita	34
3.5.4 Bilanciamento del dataset con SMOTE	36
3.5.5 Addestramento e valutazione dei modelli	36

3.5.6	Valutazione delle performance	36
3.6	Conclusioni e sviluppi futuri	39
3.6.1	Valutazione Complessiva dei Task di Classificazione	39
3.6.2	SviluppiFuturi	40
4	CLUSTERING	41
4.1	Cos'è il clustering?	41
4.1.1	Approfondimento: K-Means	41
4.1.2	Approfondimento: DBSCAN	42
4.1.3	Approfondimento: HDBSCAN	42
4.2	ETL sui dati	43
4.2.1	Scelta delle feature di interesse	44
4.2.2	Creazione delle feature	44
4.3	Analisi esplorativa dei dati	47
4.4	Analisi delle relazioni tra le variabili	50
4.4.1	Pairplot	50
4.4.2	Matrice di correlazione	51
4.4.3	Identificazione delle correlazioni forti	52
4.5	Applicazione della PCA	54
4.5.1	Standardizzazione dei dati	54
4.5.2	Applicazione della PCA	54
4.5.3	Varianza spiegata dalle componenti	55
4.5.4	Matrice delle componenti	55
4.6	Clustering con K-Means	55
4.6.1	Calcolo del WCSS e del Silhouette Score	55
4.6.2	Analisi del metodo del gomito e del silhouette score	56
4.6.3	Visualizzazione dei cluster	57
4.6.4	Caratteristiche dei cluster	58
4.6.5	Distribuzione dei silhouette score per cluster	60
4.7	Clustering con DBSCAN	61
4.7.1	Ottimizzazione dei parametri di DBSCAN	61
4.7.2	Visualizzazione dei cluster identificati	63
4.8	Clustering con HDBSCAN	64
4.8.1	Applicazione di HDBSCAN	64
4.8.2	Visualizzazione dei risultati	65
4.8.3	Confronto tra HDBSCAN e DBSCAN	66
5	FORECASTING	67
5.1	Introduzione	67
5.1.1	Motivazioni della scelta del dataset	67
5.2	Analisi preliminare	68
5.2.1	Andamento del prezzo nel tempo	68
5.2.2	Andamento del cambio giornaliero	69
5.2.3	Andamento del volume nel tempo	70
5.3	Forecasting sul prezzo di chiusura con il modello ARIMA	71
5.4	Forecasting sull'andamento del volume con il modello SARIMAX	76

Elenco delle figure

2.1	Distribuzione delle quote assegnate ai giocatori (Odd_1 e Odd_2)	4
2.2	Distribuzione dei ranking dei giocatori (Rank_1 e Rank_2)	5
2.3	Classifica dei 20 giocatori con il maggior numero di vittorie nel circuito ATP (2004-2024)	5
2.4	Distribuzione dei punti ATP dei giocatori (Pts_1 e Pts_2)	6
2.5	Distribuzione del numero di partite per superficie (sinistra) e per round (destra)	7
2.6	Andamento della percentuale di vittorie annue per Federer, Nadal e Djokovic	7
2.7	Confronto della percentuale di vittorie per anno tra Jannik Sinner e Carlos Alcaraz	8
2.8	Confronto vittorie in termini assoluti per anno tra Jannik Sinner e Carlos Alcaraz .	8
2.9	Distribuzione dei tornei ATP ospitati dai paesi	9
2.10	Distribuzione mondiale dei tornei ATP (conteggio tornei per paese)	10
2.11	Distribuzione mondiale dei tornei ATP classificati per prestigio	10
2.12	Matrice di correlazione di Pearson per le variabili del tennis	11
2.13	Matrice di correlazione di Spearman per le variabili del tennis	12
3.1	Matrice di correlazione calcolata con il coefficiente di Pearson e visualizzata tramite Seaborn	16
3.2	Matrice di confusione e metriche prodotte dal modello Random Forest	20
3.3	Matrice di confusione e metriche prodotte dal modello SVM	20
3.4	Importanza delle feature per il modello Random Forest	21
3.5	Importanza delle feature per il modello Gradient Boosting	21
3.6	Matrice di correlazione calcolata con il coefficiente di Pearson e visualizzata tramite Seaborn	25
3.7	Matrice di confusione e metriche: Logistic Regression	27
3.8	Matrice di confusione e metriche: SVM	28
3.9	Matrice di confusione e metriche: KNN	29
3.10	Ranking medio degli iscritti per categoria di torneo	31
3.11	Matrice di confusione e metriche: Random Forest	32
3.12	Matrice di confusione e metriche: SVM	33
3.13	Matrice di correlazione delle feature relative al task in questione	35
3.14	Valori di output dopo l'applicazione di SMOTE su train set e test set	36
3.15	Risultati: Gradient Boosting	37
3.16	Risultati: Logistic Regression	38
3.17	Risultati: Random Forest	38

4.1	Feature scelte per il clustering dei tennisti	46
4.2	Feature scelte per il clustering dei tennisti - 2	46
4.3	Distribuzione del numero di vittorie per giocatore	47
4.4	Relazione tra miglior ranking e percentuale di vittorie	48
4.5	Relazione tra miglior ranking e numero di vittorie	48
4.6	Distribuzione delle vittorie per superficie	49
4.7	Top 10 giocatori per numero totale di partite giocate	50
4.8	<i>Pairplot</i> delle feature numeriche scelte	51
4.9	Matrice di correlazione tra le feature	52
4.10	Correlazioni forti identificate nella correlation matrix	53
4.11	Varianza cumulativa rispetto al # di componenti	54
4.12	Grafici del WCSS (a sinistra) e del Silhouette Score (a destra) in funzione di k	56
4.13	Specifiche del grafico del WCSS (a sinistra), del silhouette score (al centro) e della riduzione percentuale del WCSS (a destra) in funzione del numero di cluster.	56
4.14	Confronto doppio tra andamento del parametro WCSS e del silhouette score.	57
4.15	Tasso di variazione percentuale del WCSS.	57
4.16	Distribuzione dei cluster nello spazio PCA bidimensionale.	57
4.17	Distribuzione tridimensionale dei cluster nello spazio PCA.	58
4.18	Distribuzione dei campioni nei tre cluster.	59
4.19	Caratteristiche principali dei cluster normalizzate - Heatmap.	59
4.20	Caratteristiche principali dei cluster normalizzate - Radar.	60
4.21	Distribuzione dei silhouette score per cluster.	60
4.22	Heatmap dello <i>Silhouette Score</i> per diverse combinazioni di eps e min_samples. Le caselle contornate in nero indicano i parametri ottimali.	62
4.23	Heatmap del numero di cluster generati per diverse combinazioni di eps e min_samples. Le caselle contornate in nero indicano i parametri ottimali.	62
4.24	Cluster identificati da DBSCAN visualizzati in 2D).	63
4.25	Cluster identificati da DBSCAN visualizzati in 3D.	64
4.26	Clustering con HDBSCAN visualizzato in 2D.	65
4.27	Clustering con HDBSCAN visualizzato in 3D.	66
5.1	Andamento dei prezzi delle azioni Apple (1984-2024).	68
5.2	Andamento corretto dei prezzi delle azioni Apple: prezzo di Chiusura, apertura, massimo e minimo Giornaliero (1984-2024).	69
5.3	Andamento della volatilità (1984-2024).	70
5.4	Andamento corretto della volatilità (1984-2024).	70
5.5	Andamento del volume (Vol.) delle Azioni Apple nel tempo (1984-2024).	71
5.6	Risultati dell'analisi delle componenti della serie temporale delle azioni Apple.	72
5.7	Confronto tra i dati prima e dopo la differenziazione.	73
5.8	Autocorrelazione e autocorrelazione parziale.	73
5.9	Specifiche del modello ARIMA.	74
5.10	Confronto tra valori reali e predetti	75
5.11	Confronto tra valori reali e predetti fino al 12/12/2022.	76
5.12	Andamento del volume delle azioni Apple con aggregazione bimestrale.	77
5.13	Forecasting del volume sugli ultimi 2 anni con modello SARIMAX.	78

CAPITOLO 1

DATASET

1.1 La scelta dei dataset

Nel presente progetto ci siamo avvalsi di due dataset principali, ciascuno scelto con l'obiettivo di rispondere a specifiche esigenze analitiche e metodologiche. Questi dataset rappresentano due scenari distinti ma complementari per l'applicazione di tecniche di analisi dati, classificazione, clustering e forecasting.

1.1.1 Dataset delle partite di tennis ATP (2004-2024)

Il primo dataset contiene informazioni dettagliate sulle partite disputate nel circuito ATP tra il 2004 e il 2024. Si tratta di un dataset estremamente ricco, che copre quasi due decenni di competizioni professionali e include variabili che descrivono sia le caratteristiche della partita che il profilo dei giocatori. Tra le colonne principali troviamo:

- **Tournament:** il nome del torneo.
- **Date:** la data della partita.
- **Series:** la categoria del torneo (es. Grand Slam, ATP 1000, ATP 500).
- **Surface:** la superficie di gioco (terra battuta, erba, cemento, ecc.).
- **Player_1 e Player_2:** i due giocatori in competizione.
- **Winner:** il vincitore della partita.
- **Rank_1 e Rank_2:** il ranking ATP dei due giocatori.
- **Odd_1 e Odd_2:** le quote dei bookmaker per entrambi i giocatori.

1.1.2 Dataset delle azioni Apple (1984-2024)

Il secondo dataset è costituito dai dati storici delle azioni Apple, che coprono un periodo esteso dal 1984 al 2024. Questo dataset contiene informazioni sul comportamento finanziario delle azioni, con variabili come:

- **Date:** la data di riferimento.
- **Price:** il prezzo di chiusura delle azioni.
- **Open:** il prezzo di apertura, il massimo e il minimo raggiunti durante la giornata.
- **High:** il prezzo massimo raggiunto durante la giornata.
- **Low:** il prezzo minimo raggiunto durante la giornata.
- **Vol.:** il volume di scambi giornaliero.
- **Change %:** la variazione percentuale rispetto al giorno precedente.

CAPITOLO 2

ANALISI DESCRITTIVA

2.1 Introduzione

Prima di procedere con le fasi principali della relazione, verranno analizzati i dati preliminari attraverso un'analisi descrittiva. Questa sezione si focalizza sull'esplorazione del dataset relativo alle partite di tennis ATP per ottenere i primi insight.

Il tennis, uno degli sport più seguiti a livello mondiale, rappresenta una ricca fonte di informazioni che possono essere analizzate per approfondire le performance dei giocatori, le dinamiche di gioco e l'evoluzione dei tornei. L'obiettivo di questa analisi descrittiva è esplorare le peculiarità dei dati, evidenziare pattern rilevanti e fornire una base solida per le fasi successive dell'analisi.

2.2 Librerie utilizzate

L'analisi è stata condotta utilizzando il linguaggio Python e le seguenti librerie:

- **Pandas**: utilizzata per la manipolazione e l'analisi dei dati tabulari. Ha permesso di caricare il dataset, esplorarlo e trasformarlo rimuovendo valori nulli o non validi.
- **Matplotlib e Seaborn**: usate per creare grafici e visualizzazioni efficaci. Matplotlib ha fornito strumenti flessibili per histogrammi, scatter plot e grafici a linee, mentre Seaborn ha aggiunto una componente estetica e intuitiva, utile per heatmap e grafici di densità.
- **NumPy**: essenziale per operazioni matematiche e manipolazioni di array numerici, come il calcolo di medie e deviazioni standard.
- **SciPy**: utilizzata per analisi statistiche avanzate, come il calcolo delle correlazioni tra variabili e la verifica di ipotesi.
- **Folium**: integrata per creare mappe interattive in caso di analisi geografiche delle partite (non utilizzata nella sezione presente ma disponibile per espansioni future).

I dati sono stati pre-elaborati per rimuovere eventuali valori mancanti, normalizzare le variabili principali e garantire l'integrità delle analisi.

2.3 Risultati dell’analisi

Di seguito verranno riportate le analisi che sono state condotte insieme ai relativi risultati ottenuti.

Distribuzione delle quote (Odd_1 e Odd_2)

Le quote assegnate dai bookmaker (**Odd_1** per il primo giocatore e **Odd_2** per il secondo) rappresentano la probabilità percepita di vittoria di ciascun giocatore in una partita; più una quota relativa a un giocatore è bassa e più la probabilità che il giocatore vinca è alta. Analizzare la distribuzione di queste quote fornisce informazioni sulla competitività dei match e sulle aspettative dei bookmaker.

I grafici in Figura 2.1 mostrano la distribuzione delle quote **Odd_1** (a sinistra) e **Odd_2** (a destra). Entrambe le distribuzioni sono positivamente asimmetriche, con la maggior parte dei valori concentrati tra 1 e 5. I picchi principali sono attorno a valori bassi, indicando che spesso uno dei giocatori è nettamente più favorito rispetto all’altro.

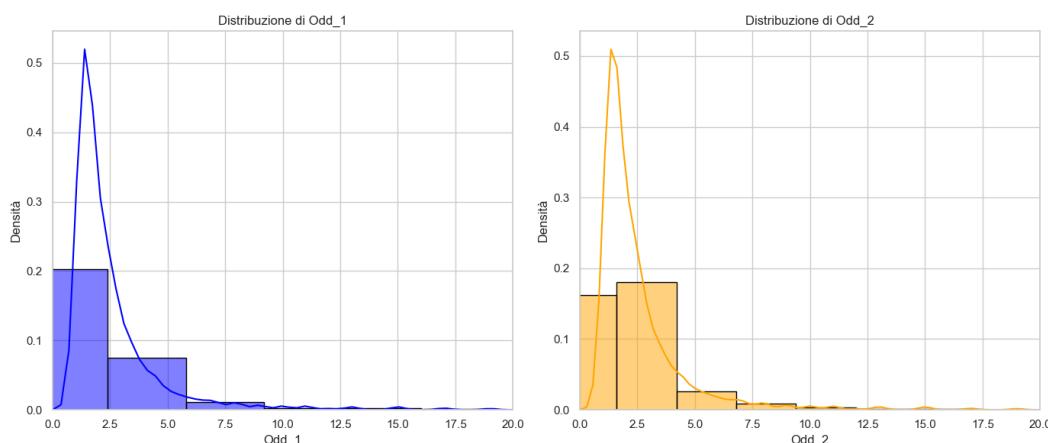


Figura 2.1: Distribuzione delle quote assegnate ai giocatori (**Odd_1** e **Odd_2**).

Distribuzione dei ranking (Rank_1 e Rank_2)

Il ranking ATP (**Rank_1** per il primo giocatore e **Rank_2** per il secondo) è un indicatore della posizione di un giocatore nella classifica mondiale, basato sulle sue performance nei tornei. Analizzare la distribuzione dei ranking consente di comprendere la distribuzione delle abilità tra i giocatori e la loro competitività.

I grafici presenti in Figura 2.2 rappresentano la distribuzione dei ranking del primo giocatore (Rank_1, a sinistra) e del secondo giocatore (Rank_2, a destra). Entrambe le distribuzioni mostrano un forte picco per valori di ranking bassi (tra 1 e 100), seguito da un rapido decremento con l’aumentare del valore di ranking.

La concentrazione elevata per valori di ranking bassi suggerisce che la maggior parte delle partite coinvolge giocatori di alto livello, classificati nelle prime 100 posizioni. La coda lunga della distribuzione indica la presenza di un numero più ridotto di giocatori con ranking elevati (oltre 200), che partecipano a tornei con minore frequenza.

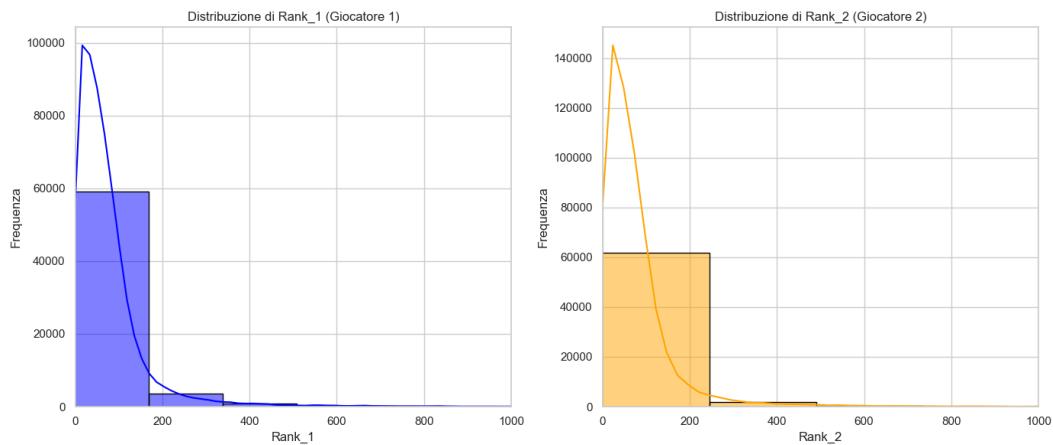


Figura 2.2: Distribuzione dei ranking dei giocatori (**Rank_1** e **Rank_2**).

Top 20 giocatori più vincenti

La classifica dei giocatori con il maggior numero di vittorie fornisce una visione complessiva delle performance individuali nel corso degli anni. Questa analisi permette di identificare i giocatori più dominanti nel circuito ATP tra il 2004 e il 2024.

Il grafico in Figura 2.3 mostra i 20 giocatori con il maggior numero di vittorie, ordinati in ordine decrescente. Roger Federer guida la classifica, seguito da Novak Djokovic e Rafael Nadal. La distribuzione evidenzia una significativa differenza tra i primi tre classificati e gli altri giocatori, con un calo graduale del numero di vittorie.

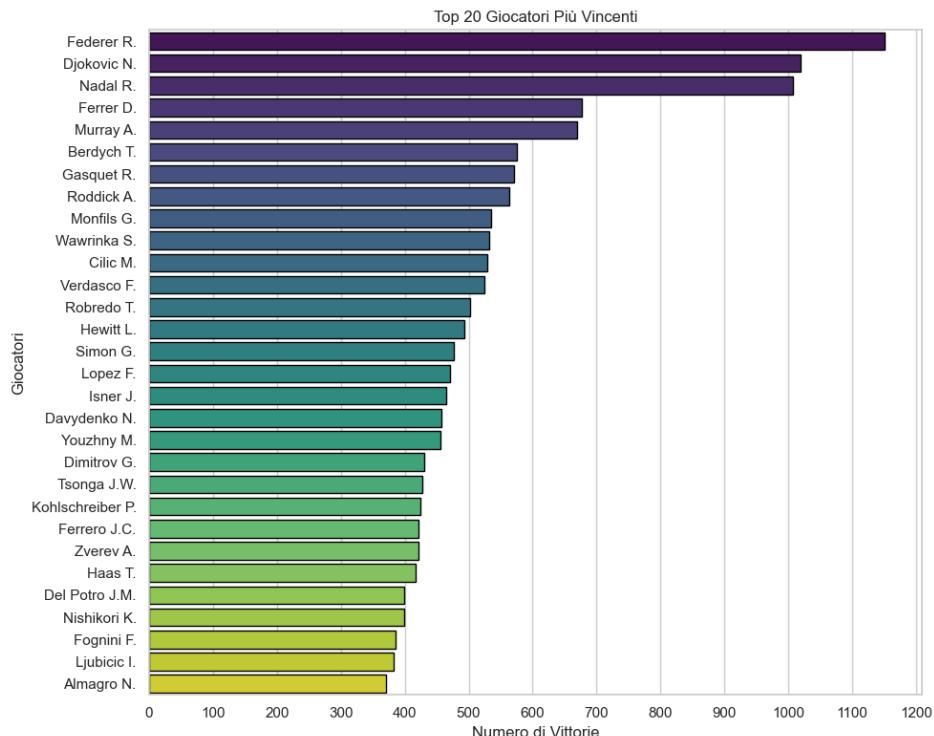


Figura 2.3: Classifica dei 20 giocatori con il maggior numero di vittorie nel circuito ATP (2004-2024).

Distribuzione dei punti ATP (Pts_1 e Pts_2)

I punti ATP (Pts_1 per il primo giocatore e Pts_2 per il secondo) rappresentano il totale dei punti accumulati dai giocatori nel ranking mondiale, basato sulle performance nei tornei. Analizzare la distribuzione di questi punti fornisce informazioni sul livello complessivo dei partecipanti. I grafici della Figura 2.4 mostrano la distribuzione dei punti ATP del primo giocatore (Pts_1, a sinistra) e del secondo giocatore (Pts_2, a destra). Entrambe le distribuzioni sono fortemente asimmetriche, con la maggior parte dei giocatori che accumula meno di 2500 punti. Sono presenti code lunghe che indicano un numero limitato di giocatori con punteggi molto elevati.

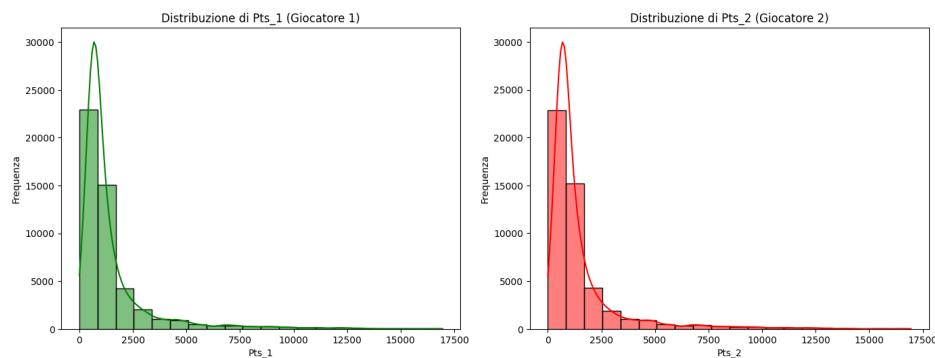


Figura 2.4: Distribuzione dei punti ATP dei giocatori (Pts_1 e Pts_2).

Distribuzione delle partite per superficie e round

Il tennis si gioca su superfici diverse (Hard, Clay, Grass), ognuna delle quali influenza lo stile di gioco e le performance dei giocatori. Inoltre, analizzare il numero di partite per round (ad esempio, primi turni, semifinali, finali) offre una prospettiva sulla struttura dei tornei e sul numero di partite giocate nei vari stadi delle competizioni. In Figura 2.5 sono evidenziati questi due aspetti:

- Il grafico a sinistra mostra il numero di partite giocate su ciascuna superficie. La superficie Hard domina con il maggior numero di partite, seguita da Clay e Grass.
- Il grafico a destra evidenzia invece il numero di partite giocate nei diversi round dei tornei. Il primo turno registra il maggior numero di partite, con una riduzione progressiva nei round successivi.

Andamento percentuale di vittorie per anno (dei "big 3")

L’analisi della percentuale di vittorie per anno offre una visione dell’andamento della performance dei giocatori di tennis nel corso del tempo. In questo grafico vengono confrontati tre dei giocatori più influenti della storia del tennis: Roger Federer, Rafael Nadal e Novak Djokovic.

La Figura 2.6 mostra un grafico a linee che illustra l’evoluzione della percentuale di vittorie annue per ciascun giocatore. Federer raggiunge il picco del 95,2% nel suo anno migliore, Nadal segue con il 95,5%, mentre Djokovic raggiunge il massimo al 94,0%. Si osservano fluttuazioni significative nel tempo, riflettendo l’alternanza di dominanza tra i tre giocatori.

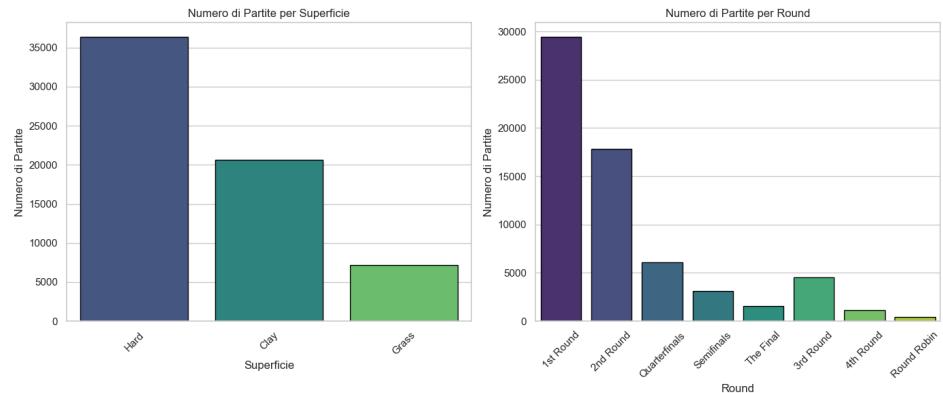


Figura 2.5: Distribuzione del numero di partite per superficie (sinistra) e per round (destra).

- Roger Federer presenta una forte crescita nella prima metà degli anni 2000, raggiungendo il suo massimo storico intorno al 2006, seguita da una graduale stabilizzazione.
- Rafael Nadal mostra un picco nel 2018, coincidente con la sua eccellente stagione sulla terra rossa, dimostrando la sua superiorità su questa superficie.
- Novak Djokovic raggiunge un’alta percentuale di vittorie a partire dal 2011, con un andamento costantemente elevato negli anni successivi, a dimostrazione della sua costanza e adattabilità su tutte le superfici.

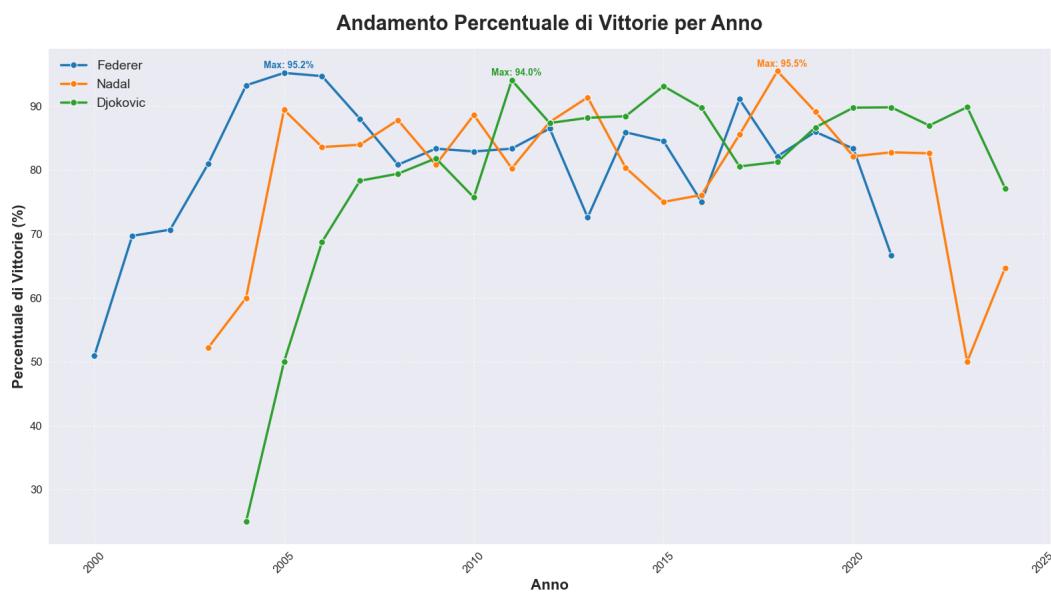


Figura 2.6: Andamento della percentuale di vittorie annue per Federer, Nadal e Djokovic.

Confronto sulle vittorie tra Sinner e Alcaraz per anno

Il confronto tra Jannik Sinner e Carlos Alcaraz evidenzia due giovani protagonisti del tennis mondiale negli ultimi anni. Analizzare la loro percentuale di vittorie nel tempo consente di valutare il loro sviluppo e la loro competitività nel circuito ATP.

Il grafico a linee (Figura 2.8) mostra l’andamento della percentuale di vittorie per anno di Sinner (linea blu) e Alcaraz (linea arancione). Entrambi i giocatori mostrano una crescita

significativa, con alcune variazioni che riflettono le differenze di performance in determinati anni. In particolare Sinner mostra una crescita costante nella percentuale di vittorie dal 2019 al 2024, segnalando un continuo miglioramento nella sua performance. Alcaraz registra un’ascesa molto rapida, raggiungendo il picco nel 2023, seguito da un leggero calo nel 2024.

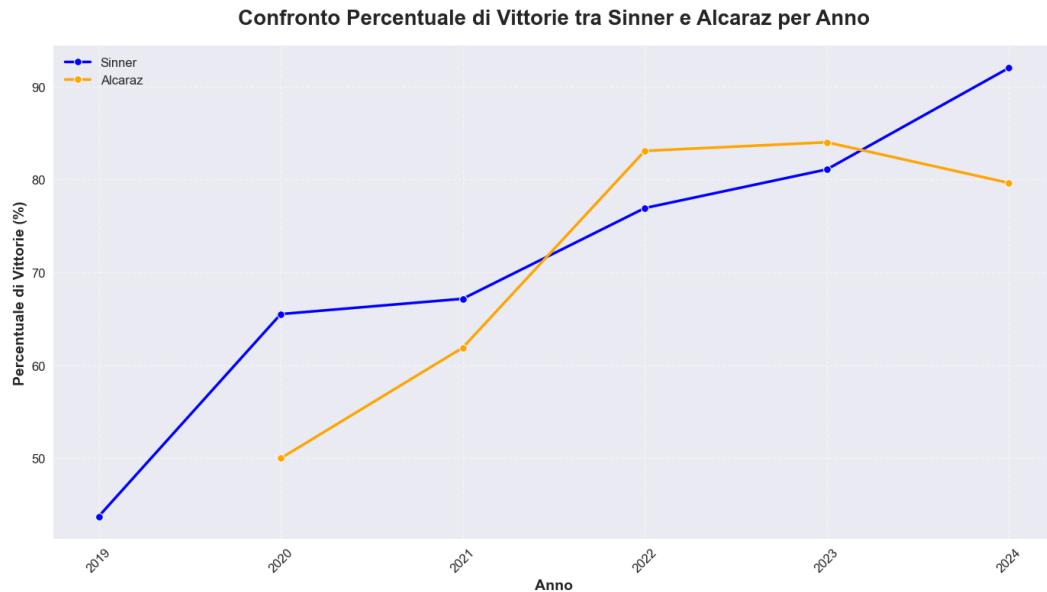


Figura 2.7: Confronto della percentuale di vittorie per anno tra Jannik Sinner e Carlos Alcaraz.

L’istogramma a barre verticali sottostanti (Figura 2.8) mostra, invece, il confronto sulle vittorie tra Sinner e Alcaraz in termini assoluti, quindi analizzando il numero di vittorie fatte dai due tennisti.

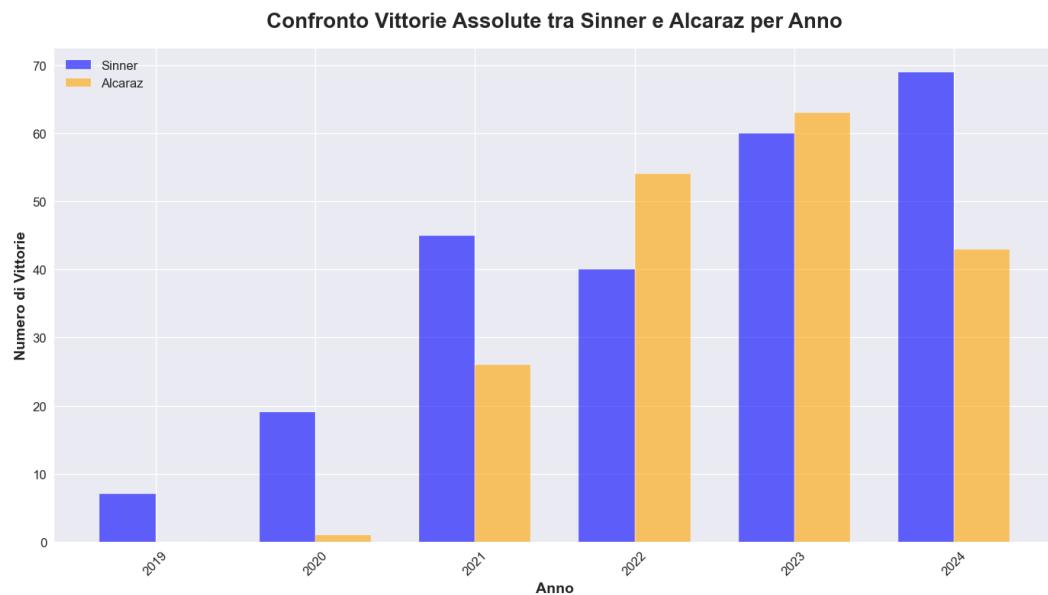


Figura 2.8: Confronto vittorie in termini assoluti per anno tra Jannik Sinner e Carlos Alcaraz.

Distribuzione dei tornei ATP per Paese

La distribuzione geografica dei tornei ATP evidenzia l’importanza di alcune nazioni nel panorama tennistico globale. Questo grafico permette di individuare i Paesi che ospitano il maggior numero di eventi, riflettendo il loro ruolo nel circuito ATP.

Il Paese in cui si svolge una partita non era un campo presente nel dataset originale. Per questa analisi, abbiamo eseguito un ingente passaggio di ETL, mappando ogni torneo al Paese corrispondente e aggiungendo successivamente una colonna al dataset. Questo passaggio ci ha permesso di associare ogni partita al suo Paese e analizzare la distribuzione geografica dei tornei.

Il grafico a barre (Figura 2.9) mostra il numero di tornei ATP ospitati dai diversi Paesi. Gli Stati Uniti d’America dominano con il maggior numero di tornei, seguiti da Francia, Regno Unito e Australia. Paesi europei come Germania, Spagna e Italia seguono, sottolineando l’importanza del tennis anche in Europa.

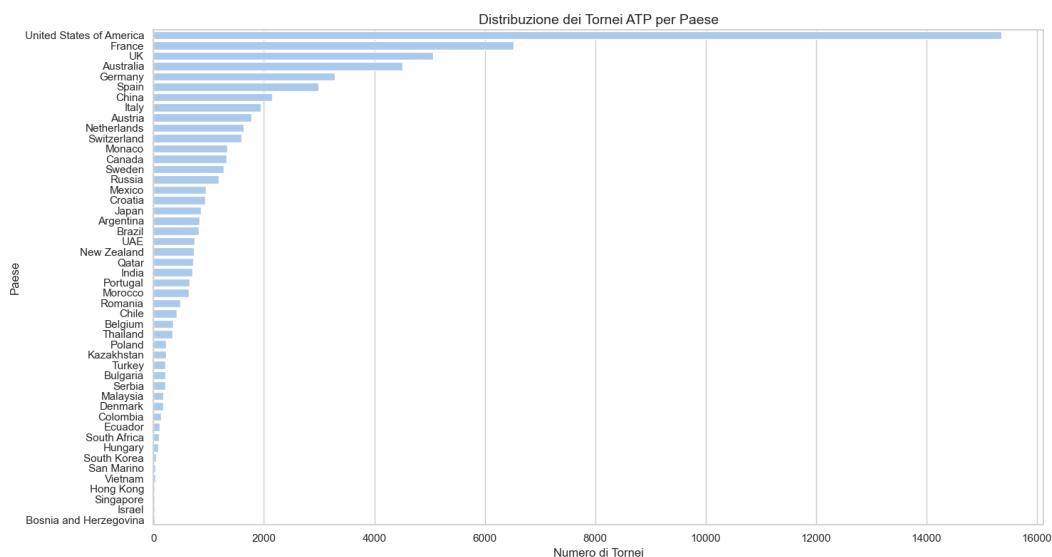


Figura 2.9: Distribuzione dei tornei ATP ospitati dai paesi.

Il grafico sottostante (Figura 2.10) fornisce una rappresentazione visiva dei tornei distribuiti nel mondo. La mappa del mondo evidenzia il numero di tornei ATP ospitati da ciascun Paese, utilizzando una scala cromatica dal giallo chiaro (basso numero di tornei) al rosso intenso (alto numero di tornei). Gli Stati Uniti emergono come il Paese dominante, seguiti da nazioni europee come Francia, Regno Unito e Germania, e, infine, dall’Australia.

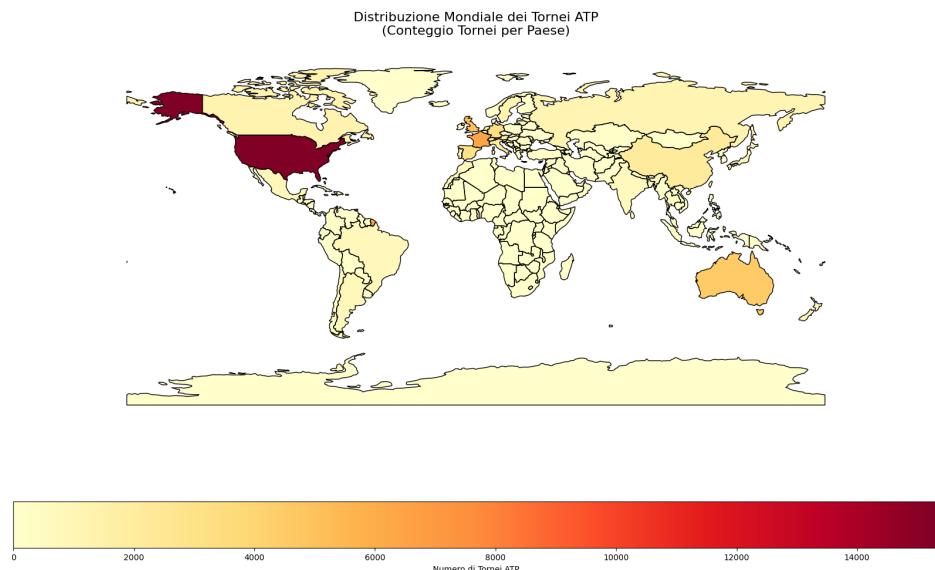


Figura 2.10: Distribuzione mondiale dei tornei ATP (conteggio tornei per paese).

Distribuzione mondiale dei tornei ATP per prestigio

I tornei ATP variano in prestigio e importanza, con categorie che includono Grand Slam, Masters 1000, ATP 500 e ATP 250. Questo grafico (Figura 2.11) illustra la distribuzione globale del torneo più prestigioso ospitato da ciascun Paese.

I Paesi evidenziati in giallo ospitano i 4 tornei del Grand Slam, la categoria più prestigiosa. I Paesi in grigio ospitano tornei Masters 1000, che rappresentano il secondo livello di importanza. I Paesi in marrone ospitano tornei ATP 500, mentre quelli in azzurro ospitano tornei ATP 250, che rappresentano il livello più basso in termini di prestigio.

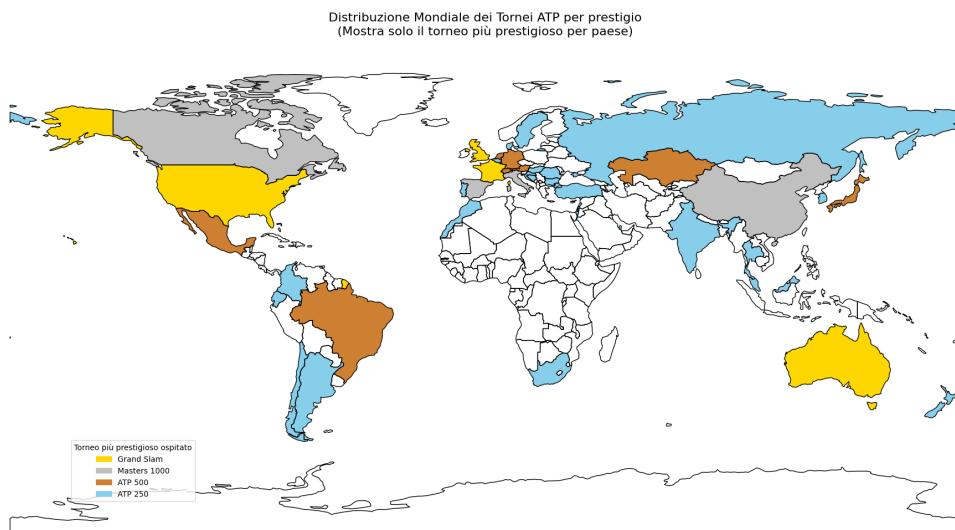


Figura 2.11: Distribuzione mondiale dei tornei ATP classificati per prestigio.

2.4 Correlazioni tra le variabili del dataset

Dopo aver esaminato le principali analisi generali del dataset, passiamo ora a verificare se esistono correlazioni significative tra le diverse feature, un aspetto di fondamentale importanza. Due strumenti che risultano essere molto utili in questo contesto sono la matrice di correlazione di Pearson e quella di Spearman che verranno analizzate nei prossimi paragrafi.

2.4.1 Matrice di correlazione di Pearson

La correlazione di Pearson è una misura statistica che quantifica la relazione lineare tra due variabili. Un valore positivo indica una correlazione diretta, mentre un valore negativo rappresenta una correlazione inversa. Il suo scopo è analizzare la forza e la direzione della relazione tra variabili numeriche, fornendo informazioni cruciali per comprendere i legami tra diversi aspetti del dataset.

Nel nostro caso, la matrice mostra le correlazioni tra i ranking, i punti ATP e le quote dei giocatori nei match. Come è possibile constatare dalla Figura 4.9), i valori di correlazione variano da -1 (correlazione negativa perfetta) a 1 (correlazione positiva perfetta), con 0 che indica l'assenza di correlazione. La scala cromatica evidenzia le correlazioni positive in rosso e quelle negative in blu.

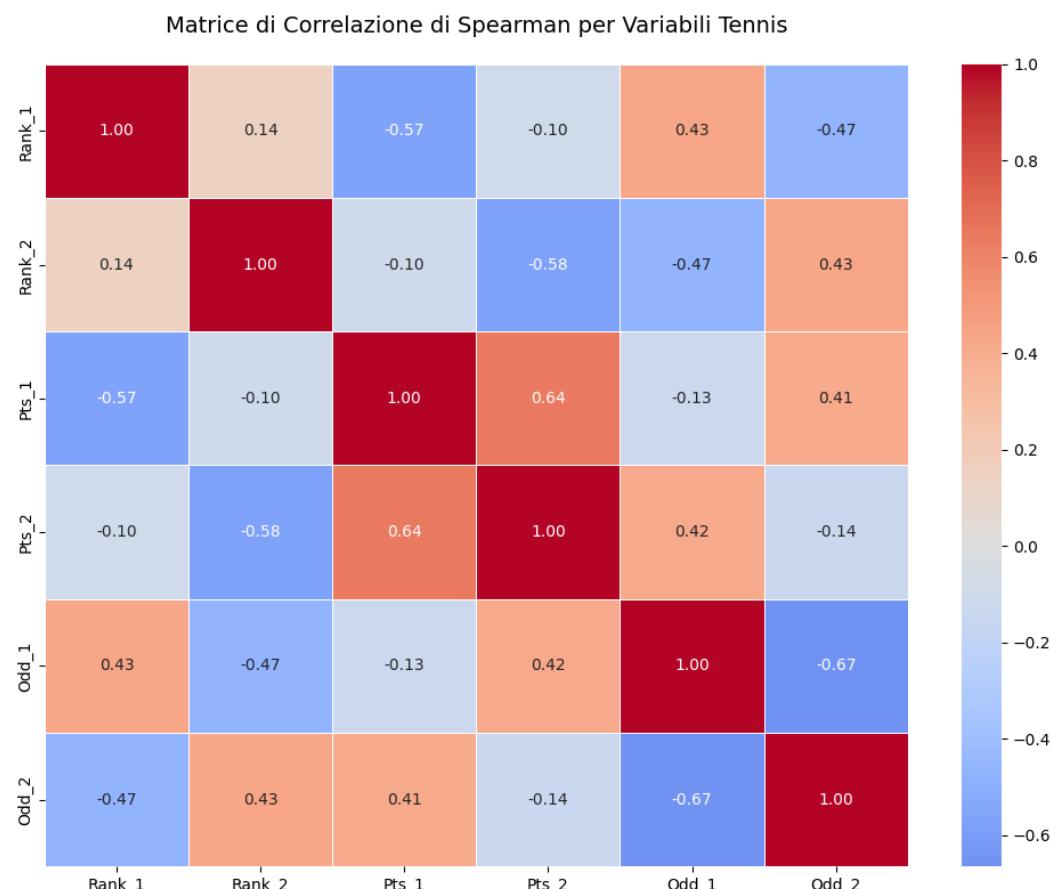


Figura 2.12: Matrice di correlazione di Pearson per le variabili del tennis.

2.4.2 Matrice di correlazione di Spearman

La correlazione di Spearman è una misura non parametrica che valuta la relazione monotona tra due variabili. A differenza della correlazione di Pearson, Spearman è particolarmente utile quando le relazioni tra le variabili non sono lineari, ma presentano un ordinamento coerente. Tuttavia, anche in questo caso l'interpretazione dei risultati è la stessa, ovvero i valori di correlazione variano da -1 a 1, con 0 che indica l'assenza di correlazione.

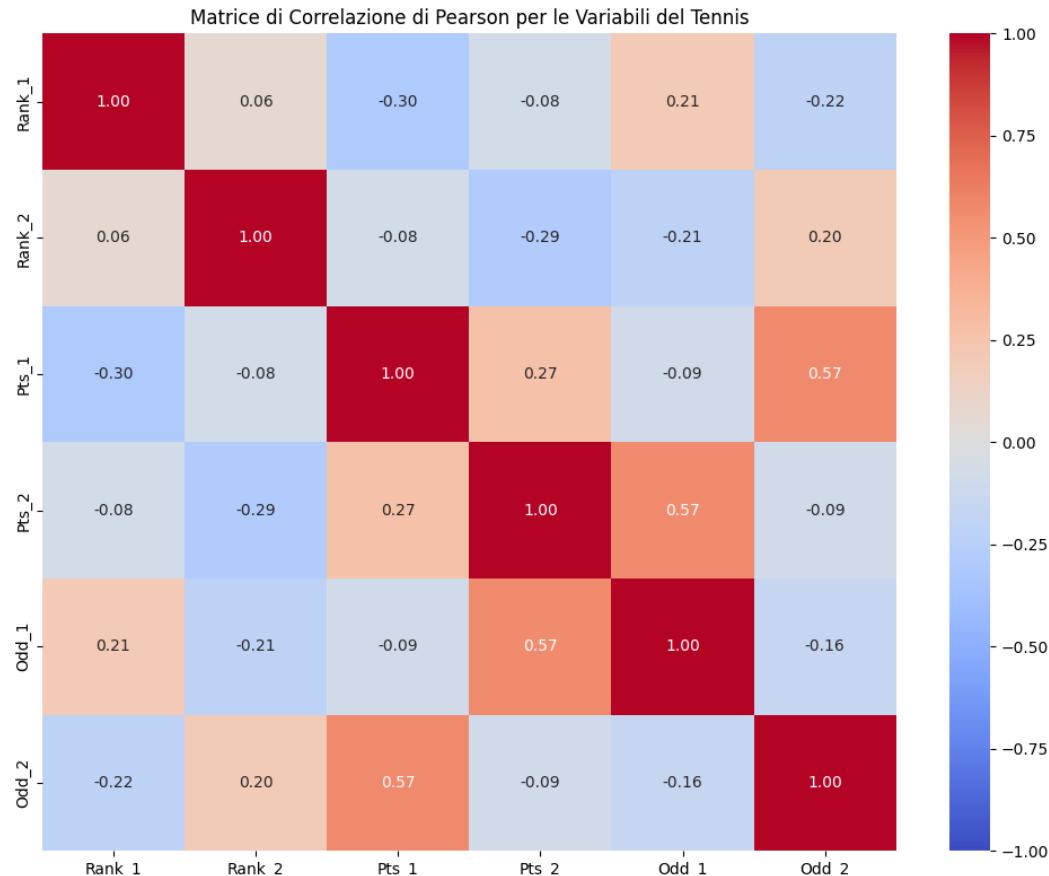


Figura 2.13: Matrice di correlazione di Spearman per le variabili del tennis.

CAPITOLO 3

CLASSIFICAZIONE

3.1 Introduzione ai Task di Classificazione sviluppati

La classificazione rappresenta una delle principali problematiche affrontate nell'ambito dell'apprendimento supervisionato, con applicazioni che spaziano dalla diagnosi medica al riconoscimento di immagini e alla previsione di preferenze degli utenti. Nel seguente studio sono stati condotti quattro diversi task, di seguito elencati:

- **T1 - Classificare il vincitore di una partita disputata tra due giocatori**
Predire quale giocatore vincerà una specifica partita in base a dati pre-partita dei due contendenti.
- **T2 - Classificare il risultato (in numero di set disputati) di una partita tra due giocatori**
Determinare il numero di set giocati (es. 2-0, 2-1) in una partita, a partire dai dati storici dei due giocatori.
- **T3 - Classificare un torneo in base alle categorie disponibili (ATP250, ATP500, Masters1000, Grand Slam)**
Assegnare un torneo alla sua categoria in base alle informazioni generali e storiche.
- **T4 - Classificare un giocatore in base alla sua superficie preferita**
Identificare la superficie preferita di un giocatore (es. terra battuta, erba, cemento) sulla base delle sue prestazioni storiche.

Questo capitolo descrive in dettaglio i passi intrapresi, partendo dalla preparazione dei dati fino all'analisi comparativa dei modelli, evidenziando le principali sfide affrontate e i risultati ottenuti.

3.1.1 Passaggi comuni nei diversi task

Le fasi principali seguite nei task di classificazione includono:

1. **Preparazione dei dati:**
i dati sono stati pre-processati per gestire valori mancanti, normalizzare le feature numeriche e trasformare le variabili categoriali in rappresentazioni numeriche.

2. Bilanciamento del dataset:

a causa della distribuzione iniziale sbilanciata delle classi, è stato applicata la *Synthetic Minority Oversampling Technique (SMOTE)* per generare campioni sintetici nelle classi minoritarie e migliorare l'efficacia dei modelli di classificazione.

3. Confronto di modelli:

sono stati addestrati diversi modelli di classificazione, tra cui Random Forest, Support Vector Machines (SVM), Logistic Regression e Gradient Boosting, per identificare quello con le migliori prestazioni.

4. Valutazione delle prestazioni:

i modelli sono stati valutati utilizzando metriche standard come accuracy, precision, recall, F1-score, oltre all'analisi delle matrici di confusione e delle curve ROC per una comprensione dettagliata delle capacità predittive degli stessi.

La metodologia sopra descritta è stata applicata a ciascun task, con adattamenti specifici per affrontare le peculiarità del problema di classificazione trattato.

3.2 T1 - Classificare il vincitore di una partita disputata tra due giocatori

3.2.1 Preparazione dei dati per la classificazione del vincitore

Questa sezione descrive il processo di preparazione e trasformazione del dataset utilizzato per il task di classificazione del vincitore. I passaggi includono la definizione delle feature principali, il filtraggio dei dati e il calcolo delle feature derivate necessarie per l'addestramento del modello.

Definizione delle Feature Base

Sono state definite le seguenti feature base, che rappresentano le caratteristiche principali dei giocatori prima di una partita:

- **Rank_1 e Rank_2:** Classifica mondiale dei due giocatori al momento della partita.
- **Pts_1 e Pts_2:** Punti ATP dei due giocatori al momento della partita.
- **rank_diff:** Differenza tra le classifiche dei due giocatori ($\text{Rank_1} - \text{Rank_2}$) in quella settimana dell'anno¹.
- **Odd_diff:** Differenza tra le quote di vittoria pre-match dei due giocatori ($\text{Odd_1} - \text{Odd_2}$).

Caricamento e Filtraggio del Dataset

Il dataset iniziale viene caricato da un file *.csv* attraverso la libreria Pandas e successivamente sottoposto a una serie di filtri per garantire la qualità dei dati:

1. Rimozione delle righe contenenti valori NaN.
2. Filtraggio delle righe con quote di vittoria (Odd_1 e Odd_2) negative o nulle.

¹La classifica ATP World Tour viene aggiornata automaticamente il Lunedì di ogni settimana, sulla base dei punti guadagnati o persi (punti da difendere dell'anno precedente nella stessa settimana) da ogni singolo giocatore.

3. Filtraggio delle righe con punti ATP (`Pts_1` e `Pts_2`) pari o inferiori a zero.

Ecco il risultato dell'operazione di filtraggio:

- Righe totali prima del filtraggio: 64.166;
- Righe rimanenti: 48.344.

Calcolo delle feature derivate

Per migliorare le capacità predittive del modello, sono state calcolate le seguenti feature derivate. Le differenze sono state calcolate come valori assoluti per eliminare l'influenza del segno e focalizzarsi sull'entità delle differenze tra i due giocatori:

- **rank_diff**: Differenza assoluta tra le classifiche dei due giocatori ($|Rank_1 - Rank_2|$).
- **rank_ratio**: Rapporto tra le classifiche dei due giocatori ($Rank_1 / Rank_2$).
- **pts_ratio**: Rapporto tra i punti ATP dei due giocatori (Pts_1 / Pts_2).
- **avg_rank**: Media delle classifiche dei due giocatori ($(Rank_1 + Rank_2) / 2$).
- **total_pts**: Somma dei punti ATP dei due giocatori ($Pts_1 + Pts_2$).
- **Odd_diff**: Differenza assoluta tra le quote di vittoria dei due giocatori ($|Odd_1 - Odd_2|$).

Analisi della correlazione tra le Feature

Per comprendere le relazioni tra le feature e identificare eventuali dipendenze lineari, è stata calcolata la matrice di correlazione utilizzando il coefficiente di correlazione di Pearson. Questo coefficiente misura la forza e la direzione della relazione lineare tra due variabili, assumendo valori compresi tra -1 e 1:

- Un valore vicino a 1 indica una forte correlazione positiva.
- Un valore vicino a -1 indica una forte correlazione negativa.
- Un valore vicino a 0 indica l'assenza di una relazione lineare.

La matrice di correlazione è stata calcolata utilizzando il metodo `corr()` di *pandas*, che implementa il coefficiente di Pearson per tutte le feature numeriche. Per facilitare l'interpretazione, questa matrice è stata rappresentata graficamente tramite una heatmap fornita dalla libreria *Seaborn*.

Visualizzazione della matrice di correlazione La *heatmap* della matrice di correlazione è mostrata nella Figura 3.1. In questa rappresentazione, le celle più vicine al rosso indicano una forte correlazione positiva, mentre quelle più vicine al blu indicano una forte correlazione negativa. Le celle neutre rappresentano l'assenza di una relazione lineare significativa tra le feature.

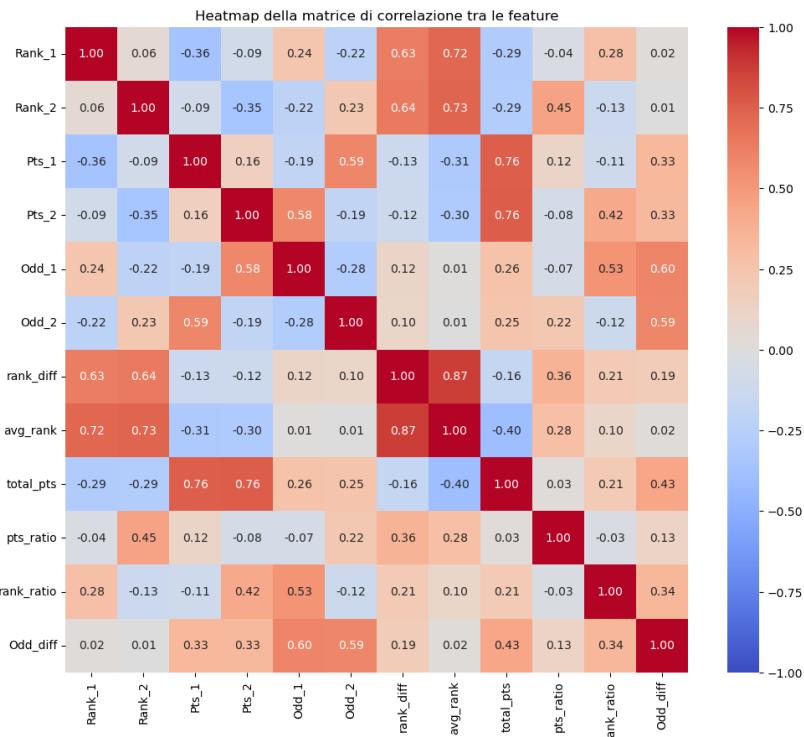


Figura 3.1: Matrice di correlazione calcolata con il coefficiente di Pearson e visualizzata tramite Seaborn.

Osservazioni principali Dall’analisi della matrice di correlazione, sono emersi i seguenti punti:

- **rank_diff** e **avg_rank** presentano una forte correlazione positiva, suggerendo una dipendenza tra la differenza di classifica e la media delle classifiche. Nell’addestramento del modello infatti, le feature **rank_diff** non è stata considerata, in quanto non fornisce ulteriori informazioni rispetto a quelle già garantite da **avg_rank**.
- La **total_pts** mostra una correlazione moderata con **Pts_1** e **Pts_2**, indicando una relazione prevedibile tra il totale dei punti e i punti accumulati dai singoli giocatori.

Questa analisi ha fornito preziose informazioni sulla struttura interna dei dati, consentendo di valutare la ridondanza delle feature e di prendere decisioni informate sulle strategie di selezione delle variabili.

Definizione del Target

La variabile target, che rappresenta il vincitore della partita, è stata calcolata come segue:

```
1      df['target'] = df.apply(lambda row: 1 if row['Winner'].strip()
() == row['Player_1'].strip() else 0, axis=1)
```

- Se il nome del vincitore (`Winner`) corrisponde al nome del giocatore 1 (`Player_1`), la classe è assegnata a **1**.
- In caso contrario, la classe è assegnata a **0**.

La variabile target, dunque, è stata definita come segue:

$$\text{target} = \begin{cases} 1, & \text{se Winner = Player_1} \\ 0, & \text{altrimenti.} \end{cases}$$

Questo processo ha permesso di creare un dataset pronto per l’addestramento dei modelli di classificazione, con feature significative e un target binario ben definito.

3.2.2 Preparazione del dataset per l’addestramento

Per addestrare i modelli di classificazione, è stata effettuata una preparazione del dataset che include la definizione delle feature, la divisione in train/test set, la verifica della distribuzione del target e la scalatura delle feature. Questi passaggi sono descritti di seguito.

Definizione delle feature e del target Le feature utilizzate per l’addestramento del modello sono state selezionate dalla lista `winner_features`, mantenendo le variabili principali che descrivono i due giocatori e il contesto della partita:

- **Rank_1** e **Rank_2**: Classifica mondiale dei due giocatori.
- **Pts_1** e **Pts_2**: Punti ATP dei due giocatori.
- **rank_diff**: Differenza di classifica tra i due giocatori.
- **Odd_diff**: Differenza tra le quote di vittoria dei due giocatori.

La variabile target `y` è stata impostata come una colonna binaria che distingue tra il vincitore (Player 1) e il perdente (Player 2).

Suddivisione del dataset in train/test Il dataset è stato suddiviso in un *training set* e un *test set* utilizzando una proporzione di 80% e 20%, rispettivamente. Questa operazione è stata effettuata con il metodo `train_test_split` di *scikit-learn*, specificando un valore di `random_state=42` per garantire la riproducibilità.

```
1 # Divisione train/test
2 X_train, X_test, y_train, y_test = train_test_split(X, y,
   test_size=0.2, random_state=42)
```

Verifica della distribuzione del target La distribuzione della variabile target è stata analizzata per controllare il bilanciamento tra le due classi:

- **Classe 1 (Player 1 vincitore)**: Numero totale di vittorie del Player 1.
- **Classe 0 (Player 2 vincitore)**: Numero totale di vittorie del Player 2.

Questa analisi ha permesso di identificare eventuali squilibri nelle classi, che potrebbero richiedere tecniche di bilanciamento nei passaggi successivi.

Scaling delle feature Per garantire che le variabili numeriche siano comparabili e migliorare l’efficienza degli algoritmi di apprendimento, è stato applicato il metodo di *standardizzazione*. Questo processo, realizzato con lo `StandardScaler` di *scikit-learn*, ha trasformato ogni feature per avere una media pari a 0 e una deviazione standard pari a 1. Sia il *training set* che il *test set* sono stati scalati utilizzando lo stesso modello di scalatura.

Verifica delle dimensioni dei dati Le dimensioni dei dati risultanti per il *training set* e il *test set* sono state verificate prima di procedere nell’addestramento.

Risultati della preparazione del target Dopo la definizione del target e la suddivisione dei dati in train e test set, i risultati principali sono stati i seguenti:

- **Distribuzione del target nel dataset completo:**

- Vittorie **Player 1**: 24.169;
- Vittorie **Player 2**: 24.175;

La distribuzione è risultata ben bilanciata tra le due classi, con una differenza minima tra il numero di vittorie dei due giocatori.

Si noti come non vi è stato necessario ricorrere a tecniche di up/downsampling per questo specifico task, in quanto i dati all’interno delle due classi apparivano già bilanciati nel dataset iniziale.

- **Dimensioni dei dataset di train e test:**

- *Training Set*: 38.675 righe e 12 feature.
- *Test Set*: 9.669 righe e 12 feature.

La suddivisione ha rispettato la proporzione 80%-20%, garantendo un adeguato numero di campioni per l’addestramento e la valutazione.

3.2.3 Addestramento e valutazione dei modelli di classificazione

Per il task di classificazione, sono stati utilizzati diversi modelli per confrontare le loro prestazioni. I modelli selezionati includono algoritmi noti per la loro efficacia nella classificazione e la loro interpretabilità.

Modelli considerati I classificatori considerati sono stati implementati utilizzando la libreria *scikit-learn* e includono:

- **Random Forest**: Un algoritmo basato su alberi decisionali.
- **Support Vector Machine (SVM)**: Un classificatore che utilizza kernel non lineari per separare i dati in classi distinte.
- **Gradient Boosting**: Un metodo basato su alberi incrementali che ottimizza la funzione di perdita durante l’addestramento.

Ogni classificatore è stato configurato con i seguenti parametri principali:

- **Random Forest**: `n_estimators=100, random_state=42`.
- **SVM**: `kernel='rbf', C=10.0, gamma='scale', random_state=42`.
- **Gradient Boosting**: `random_state=42`.

Pipeline di addestramento e valutazione Il processo seguito per ciascun modello è descritto nei passaggi seguenti:

1. **Addestramento del modello:**

Ogni classificatore è stato addestrato utilizzando il *training set* (X_{train} , y_{train}).

2. **Predizione sui dati di test:**

Dopo l'addestramento, i modelli sono stati utilizzati per effettuare predizioni sul *test set* (X_{test}).

3. **Valutazione delle prestazioni:**

Le prestazioni dei modelli sono state valutate tramite:

- **Report di classificazione:** Include metriche come precision, recall, F1-score e accuracy per ciascuna classe.
- **Matrice di Confusione:** Mostra i valori veri positivi, falsi positivi, veri negativi e falsi negativi per analizzare la distribuzione delle predizioni.

4. **Analisi dell'importanza delle feature:**

Per i modelli basati su alberi (Random Forest e Gradient Boosting), è stata analizzata l'importanza delle feature, evidenziando il contributo di ciascuna variabile al processo decisionale del modello.

Dopo l'addestramento, è stata utilizzata la proprietà `feature_importances_` dei modelli per ottenere un valore numerico di importanza per ciascuna variabile.

Successivamente, i dati sono stati ordinati in ordine decrescente di importanza per evidenziare le feature più significative.

È stato creato quindi un grafico a barre utilizzando la libreria *Seaborn*, che mostra l'importanza relativa di ciascuna feature. Le feature con maggiore importanza sono state collocate in alto, permettendo una rapida identificazione delle variabili più rilevanti.

Visualizzazione dei risultati Le valutazioni sono state supportate da visualizzazioni grafiche:

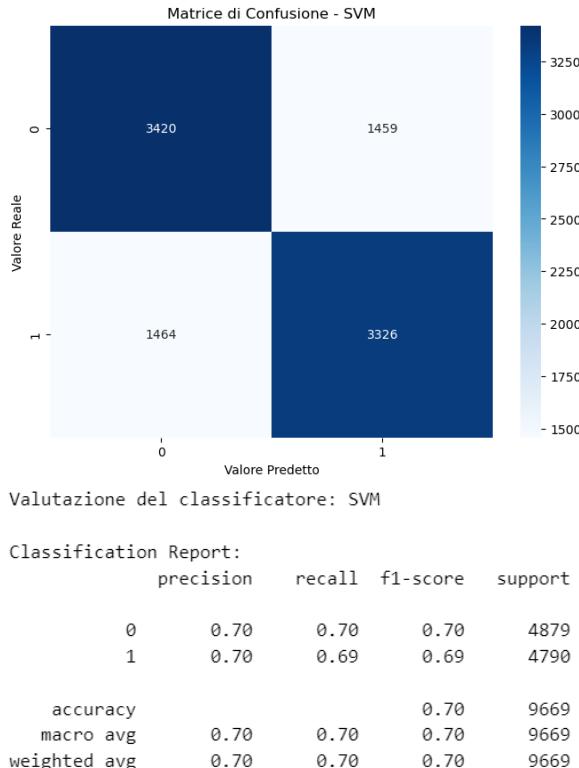
- **Matrici di confusione:** Visualizzate come heatmap per ciascun classificatore, come mostrato nella Figura 3.3.
- **Importanza delle feature:** Per Random Forest e Gradient Boosting, sono stati generati grafici a barre (Figura 3.4) che mostrano l'importanza relativa delle feature.

Risultati e Osservazioni Di seguito vengono riportati i risultati principali emersi dall'analisi delle prestazioni dei modelli di classificazione:

• **Random Forest:**

Questo modello ha mostrato ottime prestazioni grazie alla capacità di gestire feature non lineari e interazioni complesse tra le variabili. L'importanza delle feature è stata distribuita in modo equilibrato, evidenziando il contributo significativo di variabili come `pts_ratio` e `rank_ratio`.

Inoltre, le metriche di precision, recall e F1-score hanno evidenziato una distribuzione bilanciata tra le classi, rendendo il modello adatto a dataset bilanciati e complessi.

**Figura 3.2:** Matrice di confusione e metriche prodotte dal modello Random Forest**Figura 3.3:** Matrice di confusione e metriche prodotte dal modello SVM

- **Support Vector Machine (SVM):**

L'SVM ha fornito buone prestazioni, in particolare sui dataset normalizzati, sfruttando il kernel non lineare `rbf` per separare efficacemente le classi. Tuttavia, l'efficienza

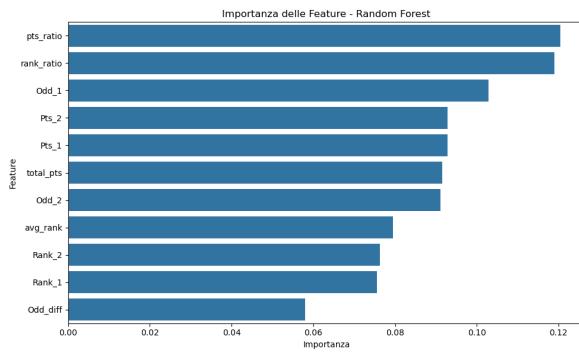


Figura 3.4: Importanza delle feature per il modello Random Forest.

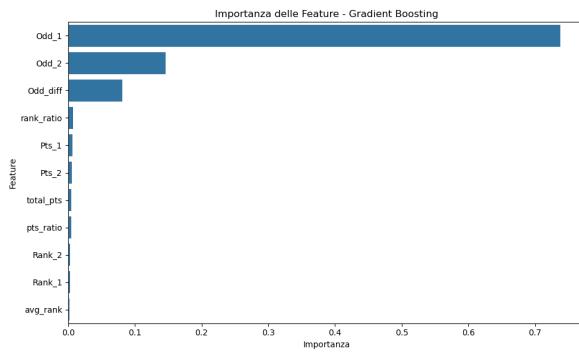


Figura 3.5: Importanza delle feature per il modello Gradient Boosting.

dell'SVM è stata maggiormente influenzata dalla scelta dei parametri C e gamma, evidenziando la necessità di una loro ottimizzazione accurata. Questo modello si è rivelato particolarmente utile quando le relazioni tra le feature e il target sono complesse ma ben definite.

- **Gradient Boosting:**

Il Gradient Boosting si è distinto per la capacità di ottimizzare le predizioni incrementali, migliorando la funzione di perdita ad ogni iterazione. Le feature relative alle quote di vittoria (Odd_1, Odd_2, Odd_diff) hanno dominato l'importanza, indicando che questo modello si concentra maggiormente su feature fortemente correlate al target.

L'elevata accuratezza ottenuta ha confermato l'efficacia del Gradient Boosting per dataset con pattern dominanti, ma il suo comportamento potrebbe risentire di feature altamente ridondanti o troppo dominanti.

Questi risultati sottolineano le differenze principali tra i modelli, offrendo spunti per la scelta del classificatore più adatto in base alle caratteristiche del dataset e agli obiettivi del task.

3.3 T2 - Classificare il risultato di una partita

3.3.1 Preparazione dei dati

Questa sezione descrive il processo di preparazione e trasformazione del dataset delle partite ATP. L'obiettivo è costruire un dataframe ottimizzato per la predizione del numero di set con cui finirà una partita, selezionando feature significative e garantendo la qualità dei dati.

Definizione delle feature base

Le seguenti feature sono state selezionate come caratteristiche principali dei giocatori prima della partita:

- **Rank_1** e **Rank_2**: Posizione in classifica ATP dei due giocatori al momento della partita.
- **Odd_1** e **Odd_2**: Quote di vittoria pre-match dei due giocatori.
- **Pts_1** e **Pts_2**: Punti ATP dei due giocatori.

Feature derivate

Per migliorare la capacità predittiva del modello, sono state introdotte nuove feature estratte dal dataset iniziale. Il compito iniziale di predire il numero di set in una partita si è rivelato particolarmente complesso a causa della variabilità del punteggio e delle numerose dinamiche di gioco che influenzano l'andamento di un incontro.

Per affrontare questa sfida, è stato condotto un approfondito lavoro di ingegneria delle feature al fine di estrarre informazioni più rilevanti e migliorare le prestazioni del modello. Le feature derivate sono state progettate per catturare aspetti chiave come la differenza di esperienza tra i giocatori, la loro performance storica negli scontri diretti e la capacità di sostenere match lunghi. Di seguito le principali feature calcolate:

- **prob_terzo_set_p1** e **prob_terzo_set_p2**: Probabilità che il giocatore 1 o il giocatore 2 vinca un set decisivo, calcolata sulla base degli incontri disputati negli scorsi match². Ecco la funzione utilizzata per calcolare questi valori:

```

1      def calcola_probabilità_giocatori(df, n_bins):
2          """
3              Calcola per ogni giocatore la probabilità aggregata di
4                  andare al terzo set
5                  in base al ranking medio degli avversari nelle partite
6                  passate.
7
8          Parameters:
9              df (DataFrame): Il DataFrame contenente i dati
10                 delle partite.
11              n_bins (int): Numero di intervalli per suddividere
12                 i ranking.
13
14          Returns:
15              dict: Un dizionario con i giocatori come chiavi e
16                  le probabilità di andare al terzo set.
17          """
18
19          # Calcolo della media del ranking degli avversari e
20          # probabilità di terzo set
21          prob_terzo_set = {}
22
23
24          # Per ogni giocatore come Player 1
25          for player in df['P1_name'].unique():
26              player_data = df[df['P1_name'] == player]
27              if len(player_data) > 0:

```

²I valori descritti sono stati calcolati su tutto il dataset iniziale, perciò su un periodo di 24 anni

```

20             avg_rank_opponent = player_data['ATP_rank_p2'].
21             mean()
22             prob_set_3 = player_data['target_sets'].mean()
23             prob_terzo_set[player] = (avg_rank_opponent,
24             prob_set_3)
25
26             # Per ogni giocatore come Player 2
27             for player in df['P2_name'].unique():
28                 player_data = df[df['P2_name'] == player]
29                 if len(player_data) > 0:
30                     avg_rank_opponent = player_data['ATP_rank_p1'].
31                     mean()
32                     prob_set_3 = player_data['target_sets'].mean()
33                     if player in prob_terzo_set:
34                         # Media delle probabilità già calcolate
35                         prob_terzo_set[player] = (
36                             (prob_terzo_set[player][0] +
37                             avg_rank_opponent) / 2,
38                             (prob_terzo_set[player][1] + prob_set_3
39                             ) / 2,
39                         )
39             else:
40                 prob_terzo_set[player] = (avg_rank_opponent
41                 , prob_set_3)
42
43             return prob_terzo_set

```

- **long_matches_p1 e long_matches_p2:** Numero di partite "lunghe" (con più di 20 giochi) disputate dai due giocatori negli ultimi mesi. Nel pezzo di codice che segue viene descritta la funzione utilizzata per il calcolo di questa feature:

```

1     def calcola_partite_lunghes(df, n=12):
2     """
3         Calcola il numero di partite "lunghe" (>20 giochi)
4         giocate da ciascun giocatore negli ultimi n mesi.
5     """
6         # Assicura che la colonna 'Date' sia in formato
7         # datetime
8         df['Date'] = pd.to_datetime(df['Date'])
9         cutoff_date = df['Date'].max() - pd.DateOffset(months=n
10     )
11
12         # Calcola il numero totale di giochi per ogni partita
13         df['total_games'] = df['Score'].apply(
14             lambda score: sum([int(x.split('-')[0]) + int(x.
15             split('-')[1]) for x in score.split() if '-' in x])
16         )
17
18         # Filtra solo le partite lunghe
19         long_matches = df[(df['total_games'] > 20) & (df['Date'
20             ] >= cutoff_date)]
21
22         # Conta le partite lunghe per ogni giocatore
23         long_matches_p1 = long_matches.groupby('P1_name').size
24         ()

```

```

19     long_matches_p2 = long_matches.groupby('P2_name').size()
20     ()
21     # Combina i risultati in un dizionario
22     partite_lunghe = long_matches_p1.add(long_matches_p2,
23     fill_value=0).to_dict()
24     # Aggiungi valori di default per tutti i giocatori
25     all_players = pd.concat([df['P1_name'], df['P2_name']]).unique()
26     partite_lunghe_full = {player: partite_lunghe.get(
27     player, 0) for player in all_players}
28
return partite_lunghe_full

```

- **H2H_p1** e **H2H_p2**: Storico degli scontri diretti (Head-to-Head), indicando il numero di vittorie di ciascun giocatore nelle precedenti sfide.
- **form_diff**: Differenza di forma tra i due giocatori basata sui risultati recenti (Rapporto vittorie/sconfitte nelle ultime 5 partite disputate da ogni giocatore).
- **Surface**: Tipo di superficie su cui si gioca la partita.
- **Court**: Tipologia del campo (indoor o outdoor).

L'inclusione di queste feature ha permesso di arricchire il dataset con informazioni fondamentali per migliorare la capacità predittiva del modello, rendendolo più robusto nel distinguere le caratteristiche dei giocatori e nel determinare le loro probabilità di vittoria in termini di set.

Filtraggio del dataset

Per migliorare la qualità dei dati, sono stati applicati i seguenti filtri:

1. Rimozione delle righe con valori mancanti nelle colonne principali.
2. Esclusione delle partite in cui i ranking dei giocatori non erano disponibili.
3. Filtraggio delle righe con quote di vittoria non valide.
4. Rimozione delle partite senza un valore calcolabile per le feature H2H e long matches.
5. Rimozione delle partite al meglio dei 5 set, in questo modo si è ridotto il problema ad un task di classificazione binaria (2 set o 3 set).

Correlazione feature

Anche per questo task, è stato eseguito un controllo sulla possibile presenza di correlazione (Figura 3.6) tra le feature selezionate, sia quelle base che quelle derivate.

Si noti come **ATP_rank_p1** e **ATP_rank_p2** sono moderatamente correlate con **rank_diff**, suggerendo una connessione prevedibile tra il ranking individuale dei giocatori e la loro differenza di ranking. **Surface** e **Court** mostrano una bassa correlazione con le altre feature, indicando che queste variabili potrebbero avere un'influenza limitata nella predizione del numero di set. **form_diff** ha una correlazione moderata con **rank_diff**, suggerendo che i giocatori con una forma recente migliore tendono ad avere un ranking relativamente più alto.

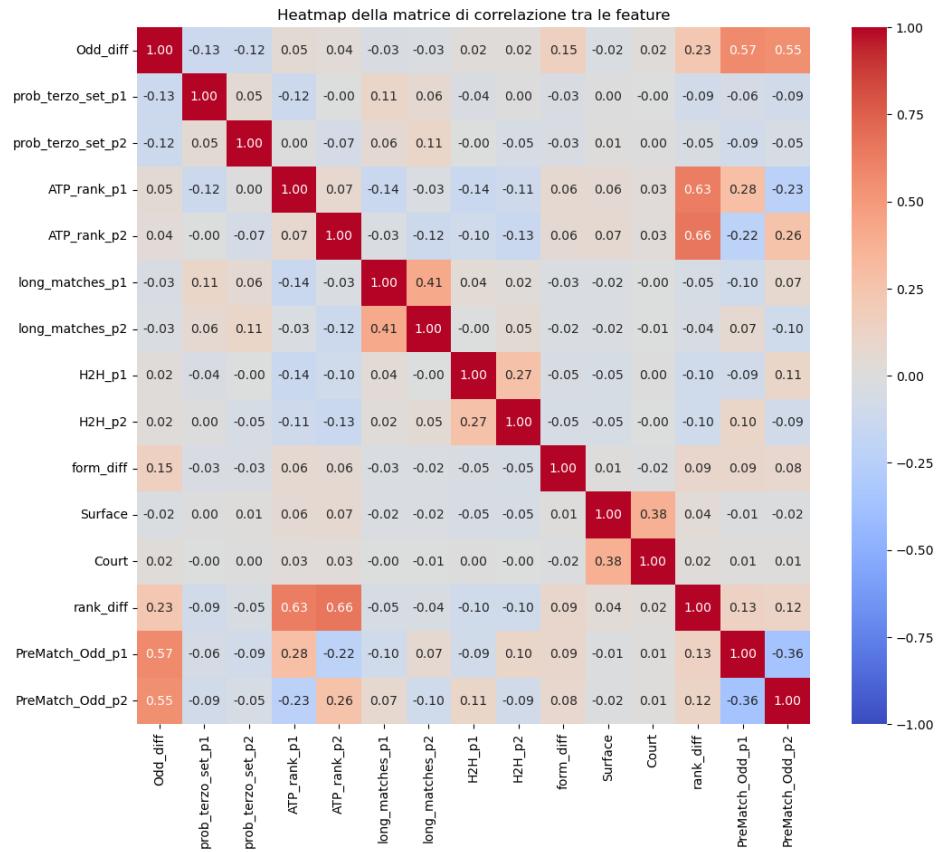


Figura 3.6: Matrice di correlazione calcolata con il coefficiente di Pearson e visualizzata tramite Seaborn.

3.3.2 Preparazione del dataset per l’addestramento

La preparazione del dataset per l’addestramento è stata eseguita seguendo una serie di passaggi fondamentali per garantire la qualità e la rappresentatività dei dati:

Mappatura delle variabili categoriche Le colonne categoriche Surface e Court sono state convertite in valori numerici utilizzando una mappatura definita manualmente:

- **Surface:** Hard: 0, Clay: 1, Grass: 2, Carpet: 3
- **Court:** Indoor: 0, Outdoor: 1

Divisione in train e test set Il dataset è stato suddiviso in un training set e un test set con una proporzione di 80%-20%, utilizzando una suddivisione stratificata per mantenere la distribuzione originale delle classi.

Bilanciamento delle classi Il dataset presentava un iniziale squilibrio tra le classi, con una netta prevalenza di partite concluse in due set rispetto a quelle concluse in tre set. Tale squilibrio avrebbe potuto influire negativamente sulle performance dei modelli di classificazione, portandoli a favorire la classe maggioritaria. Per affrontare questa problematica, è stato applicato un processo di sovraccampionamento sulla classe minoritaria utilizzando la funzione `resample()` della libreria *scikit-learn*.

L'obiettivo principale del sovraccampionamento è stato quello di creare un dataset bilanciato, dove entrambe le classi fossero rappresentate in modo equo. Dopo il sovraccampionamento, la distribuzione delle classi nel set di addestramento è risultata la seguente:

- **Classe 0 (2 Set)**: 20.100 campioni.
- **Classe 1 (3 Set)**: 20.100 campioni.

Questo approccio ha consentito di migliorare l'efficacia degli algoritmi di apprendimento automatico, garantendo che le predizioni per entrambe le classi non fossero influenzate dal predominio della classe maggioritaria. Inoltre, ha reso le metriche di valutazione, come precision e recall, più rappresentative delle reali performance dei modelli.

3.3.3 Addestramento e valutazione modelli

Per il task di classificazione, sono stati considerati diversi modelli di apprendimento automatico, al fine di confrontare le loro prestazioni nella predizione del numero di set in una partita. I modelli selezionati includono algoritmi noti per la loro capacità di generalizzazione e versatilità.

Modelli considerati: I classificatori utilizzati sono:

- **Logistic Regression**: Algoritmo lineare con bilanciamento delle classi e solver saga.
- **SVM (Support Vector Machine)**: Classificatore lineare con kernel lineare e bilanciamento delle classi.
- **Random Forest**: Algoritmo basato su alberi decisionali, configurato con 100 stimatori.
- **XGBoost**: Modello di boosting graduale configurato per ottimizzare la funzione di perdita logloss.
- **LightGBM**: Modello di boosting leggero ed efficiente.
- **K-Nearest Neighbors (KNN)**: Classificatore basato sulla vicinanza di 5 punti.
- **Decision Tree**: Albero decisionale standard.
- **Extra Trees**: Versione estesa di Random Forest con maggiore randomizzazione.

Pipeline di addestramento: Per ogni modello, è stato seguito il processo di addestramento sui dati bilanciati (`x_train_balanced` e `y_train_balanced`). L'addestramento è stato effettuato con le seguenti configurazioni:

- Bilanciamento delle classi utilizzando il parametro `class_weight='balanced'` ove disponibile.
- Impostazione di una riproducibilità uniforme utilizzando il valore `random_state=42`.

Tutti i modelli sono stati addestrati utilizzando la funzione `fit()` e memorizzati in un dizionario per la successiva valutazione.

Valutazione dei modelli: La valutazione è stata condotta sui dati di test (`x_test` e `y_test`), utilizzando le seguenti metriche:

- **Matrice di Confusione:** Visualizza la distribuzione delle predizioni corrette e degli errori tra le due classi (2 Set e 3 Set). È stata rappresentata graficamente per ciascun modello, evidenziando le prestazioni nella classificazione.
- **Report di Classificazione:** Include metriche come precision, recall, F1-score e accuracy per ciascuna classe.

Ogni modello è stato valutato mediante:

1. Predizione dei valori di test (`y_pred`).
2. Calcolo della matrice di confusione con la funzione `confusion_matrix()` e successiva visualizzazione con `ConfusionMatrixDisplay`.
3. Generazione del report di classificazione utilizzando `classification_report()`.

3.3.4 Risultati e osservazioni:

I risultati dell’addestramento e della valutazione dei modelli sono stati analizzati attraverso le matrici di confusione e i report di classificazione. Le performance dei modelli principali sono riportate nelle Figure 3.7, 3.8 e 3.9. Di seguito, sono riportate le osservazioni più significative.

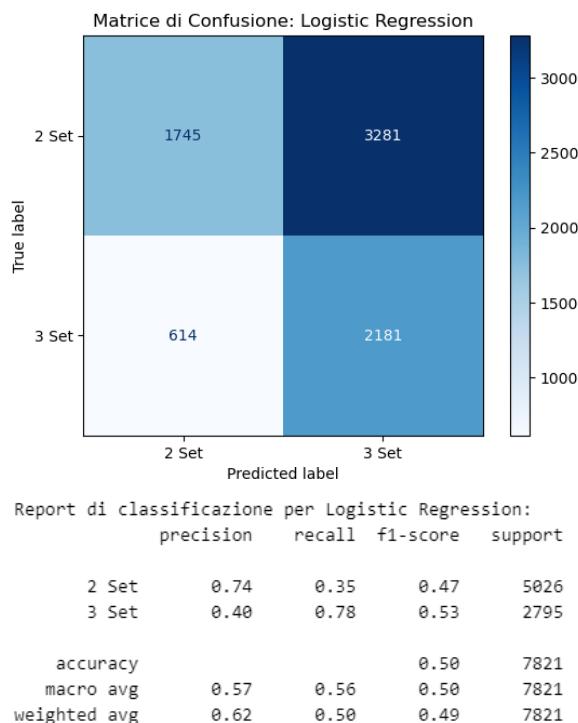


Figura 3.7: Matrice di confusione e metriche: Logistic Regression

Logistic Regression: La regressione logistica ha mostrato una scarsa capacità di distinguere tra le due classi (2 Set e 3 Set). In particolare, la recall della classe 2 Set è risultata inferiore rispetto alla classe 3 Set, indicando che il modello fatica a identificare correttamente alcune

partite concluse in due set. Questo risultato può essere attribuito alla natura lineare del modello, che potrebbe non essere sufficientemente flessibile per catturare tutte le relazioni non lineari nei dati.

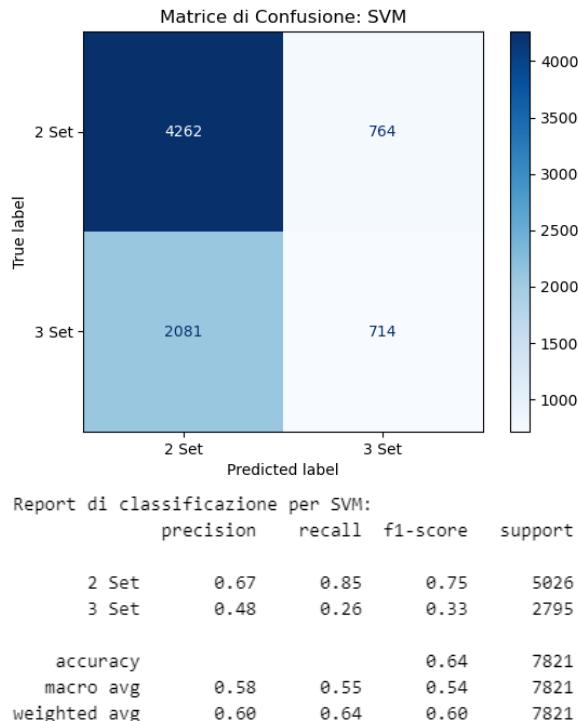


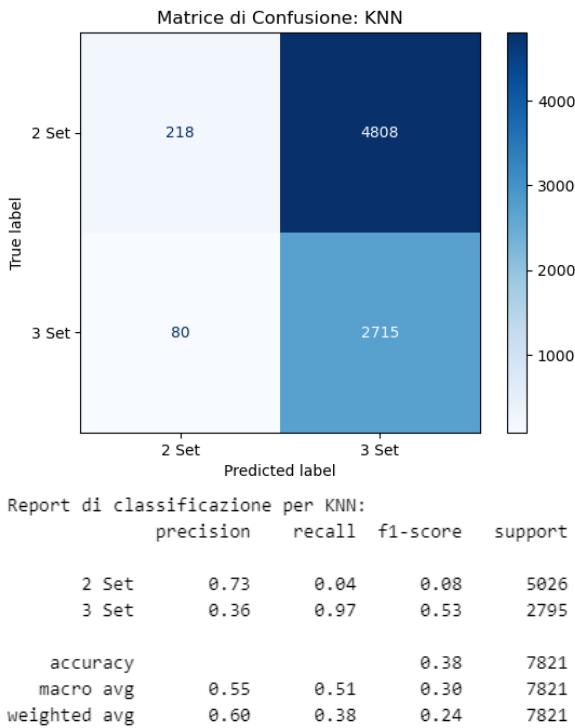
Figura 3.8: Matrice di confusione e metriche: SVM

Support Vector Machine (SVM): Il modello SVM con kernel lineare ha ottenuto buoni risultati complessivi, con una accuracy del 68% e una precision simile per entrambe le classi. Tuttavia, è evidente che il modello tende a sovrastimare la classe maggioritaria (*3 Set*), come indicato dal numero maggiore di falsi negativi per la classe *2 Set*. Questo comportamento può essere giustificato dalla natura del kernel lineare, che potrebbe non essere ideale per dataset con confini di decisione complessi.

K-Nearest Neighbors (KNN): Il modello KNN ha ottenuto performance inferiori rispetto agli altri modelli, con una accuracy complessiva del 55% e una precision per la classe *2 Set* significativamente più bassa. Il numero elevato di falsi negativi evidenzia che il modello fatica a distinguere efficacemente le partite concluse in due set. Questo risultato è attribuibile alla sensibilità del modello ai dati rumorosi e alla mancanza di una struttura decisionale robusta.

Osservazioni finali:

- **Influenza dei dati bilanciati:** I modelli hanno beneficiato del sovraccampionamento, che ha ridotto il bias verso la classe maggioritaria (*2 Set*). Tuttavia, questo approccio potrebbe aver introdotto una lieve ridondanza nei dati, influenzando i risultati.
- **Natura del task:** La classificazione del numero di set in una partita di tennis è intrinsecamente complessa, in quanto dipende da molteplici fattori dinamici (es. performance

**Figura 3.9:** Matrice di confusione e metriche: KNN

individuale, superfici, stato fisico). Questa complessità giustifica la presenza di alcune discrepanze nei risultati.

- **Scalabilità dei modelli:** Modelli come SVM e Logistic Regression si sono dimostrati scalabili ed efficienti per il dataset bilanciato, mentre il KNN ha evidenziato limiti in termini di accuratezza e capacità predittiva.
- **Limiti del dataset per questo task specifico:** Il dataset contiene informazioni statiche relative alle partite, come il ranking ATP, le quote pre-match e le statistiche aggregate. La mancanza di dati dinamici (es. statistiche in tempo reale, dettagli sulle performance durante la partita) ha limitato la possibilità di creare feature più specifiche o predittive.
- **Contesto del task:** La classificazione del numero di set dipende da molteplici fattori intangibili e difficili da quantificare, come la forma mentale dei giocatori, le condizioni meteorologiche e i dettagli tattici. Questi aspetti non possono essere rappresentati adeguatamente con i dati disponibili.
- **Trade-off tra complessità e valore aggiunto:** Approfondire ulteriormente il feature engineering avrebbe richiesto un aumento significativo della complessità computazionale senza garantire un miglioramento sostanziale nelle prestazioni del modello, data la natura del task e le limitazioni dei dati.
- **Generalizzabilità:** L'introduzione di feature altamente specifiche potrebbe portare a un modello meno generalizzabile, soprattutto quando applicato a dati nuovi o diversi da quelli del dataset di addestramento.

3.4 T3 - Classificare la categoria di un torneo

Per questo task, si è scelto di provare a classificare la categoria del torneo sulla base dei 4 possibili casi già presenti nel dataset:

- ATP 250;
- ATP 500;
- Masters 1000;
- Grand Slam;

3.4.1 Filtraggio e Preprocessing del Dataset

Per preparare il dataset alla modellazione, sono state eseguite le seguenti operazioni di filtraggio e preprocessing:

Filtraggio dei Dati di Base

- Sono stati rimossi i record con valori non validi di **Rank** e **Pts** (Rank_1, Rank_2, Pts_1, Pts_2), riducendo il dataset da *original_rows* a *rows_after* righe.
- Distribuzione originale delle categorie (**Series**) e dei turni (**Round**) è stata analizzata per identificare categorie non rilevanti.

Filtraggio delle categorie e dei round

- Sono stati rimossi i record relativi alle categorie **Series** non rilevanti (International, International Gold, Masters, Masters Cup), con una riduzione di *rows_before_series* - *rows_after_series* righe.
- È stato eliminato il Round Robin dai turni, con una riduzione di *rows_before_round* - *rows_after_round* righe.
- Dopo i filtri, sono state mantenute *rows_after_round* righe totali, rappresentando il $(\text{len}(df)/\text{original_rows}) * 100$ % dei dati originali.

Ingegneria delle Feature

- **Feature binarie:** Sono state aggiunte le variabili `is_best_of_5` e `is_grand_slam` per identificare i match giocati al meglio dei 5 set e quindi nei tornei del Grande Slam.
- **Ranking e differenze:** Sono state introdotte le variabili `winner_rank` (ranking del vincitore), `rank_diff` (differenza tra i ranking) e `avg_rank` (media dei ranking).
- **Feature derivate dai turni:** È stata mappata la variabile `Round` in numeri interi (`round_number`) e calcolate le variabili `has_3rd_round` (flag per i tornei con terzo turno) e `draw_size` (dimensione del tabellone), basandosi sul tipo di torneo e sul turno.

Nella figura che segue (Figura 3.10, è stato controllato il ranking medio per ogni categoria di torneo.

```
Ranking medio per categoria del torneo:
Series
ATP250           92.214895
ATP500           62.068706
Grand Slam       68.620612
Masters 1000     48.742143
Name: avg_rank, dtype: float64
```

Figura 3.10: Ranking medio degli iscritti per categoria di torneo

3.4.2 Bilanciamento delle classi per categoria di torneo

Per garantire che il modello non sia influenzato da una distribuzione squilibrata delle categorie di torneo, il dataset è stato separato in base alla tipologia di evento e riequilibrato tramite tecniche di upsampling e downsampling.

Separazione delle Categorie di Torneo Il dataset è stato suddiviso nelle seguenti categorie:

- **ATP 250:** Tornei della categoria ATP 250.
- **ATP 500:** Tornei della categoria ATP 500.
- **Masters 1000:** Tornei della serie Masters 1000.
- **Grand Slam:** Tornei dei quattro Slam.

Bilanciamento delle classi Poiché le categorie non erano equidistribuite, è stata applicata una strategia combinata di **downsampling** e **upsampling** per garantire una distribuzione equilibrata delle classi:

- La dimensione target è stata definita in base alla categoria con il minor numero di esempi (`n_samples`).
- La classe maggioritaria (**ATP 250**) è stata sottocampionata (*downsampling*) riducendo il numero di esempi a `n_samples`.
- Le categorie ATP 500, Masters 1000 e Grand Slam sono state sovraccampionate (*upsampling*) fino a raggiungere la dimensione target `n_samples`.

3.4.3 Valutazione dei modelli

I risultati dell’addestramento e della valutazione dei modelli utilizzati anche nei precedenti task sono stati analizzati attraverso le matrici di confusione e i report di classificazione. Di seguito, sono riportate le osservazioni più significative.

Random Forest:

- **Accuracy:** 78%
- **Prestazioni per le classi:** La classe *Grand Slam* (2) è stata classificata con precisione e recall perfetti (1.00), dimostrando che il modello riconosce con precisione i pattern distintivi di questa categoria. Le prestazioni per le altre classi, come *ATP500* e *ATP250*, risultano invece più basse (Figura 3.11).

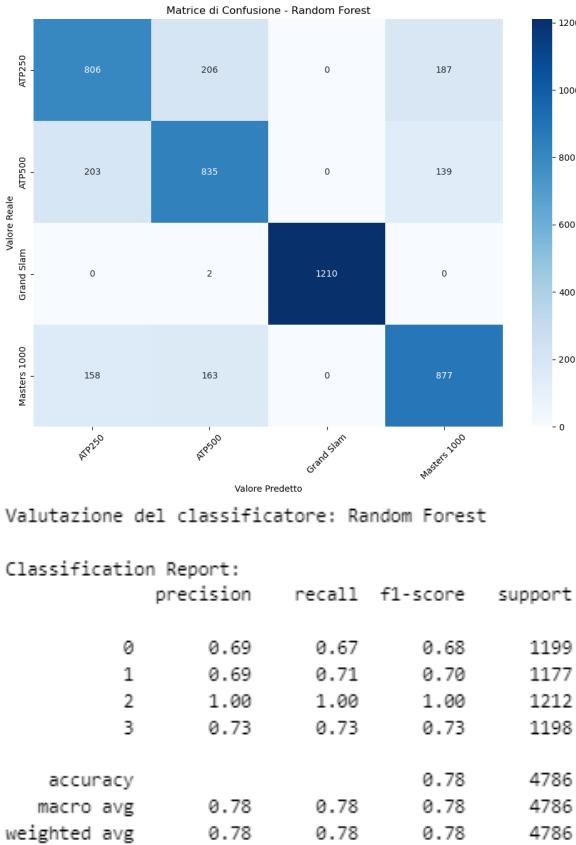


Figura 3.11: Matrice di confusione e metriche: Random Forest

SVM (Support Vector Machine):

- **Accuracy:** 67%
- **Prestazioni per le classi:** Sebbene la classe *Grand Slam* (2) sia stata classificata correttamente con precision e recall pari a 1.00, le performance per le categorie *ATP250* (0) e *ATP500* (1) sono significativamente inferiori, con precision e recall rispettivamente del 54% e 53% (Figura 3.12).

Osservazioni sui Risultati: I risultati meno accurati per le categorie *ATP250* e *ATP500* possono essere attribuiti alla mancanza di diversità nei dati di queste categorie rispetto ai *Grand Slam*. In particolare:

- Le classi non *Grand Slam* presentano caratteristiche più simili tra loro, rendendo difficile per i modelli distinguerele in modo netto.
- Il dataset a disposizione non contiene elementi aggiuntivi, come il *prize money*, il livello di competizione o il numero di top players presenti, che potrebbero fornire informazioni utili per differenziare le categorie.

Limiti del Dataset: L'assenza di feature contestuali, come dettagli sul montepremi (*prize money*) o sul prestigio del torneo, limita le capacità predittive dei modelli per le categorie *ATP250*, *ATP500* e *Masters 1000*. Questo suggerisce che ulteriori miglioramenti richiederebbero l'inclusione di variabili aggiuntive per caratterizzare meglio i tornei.

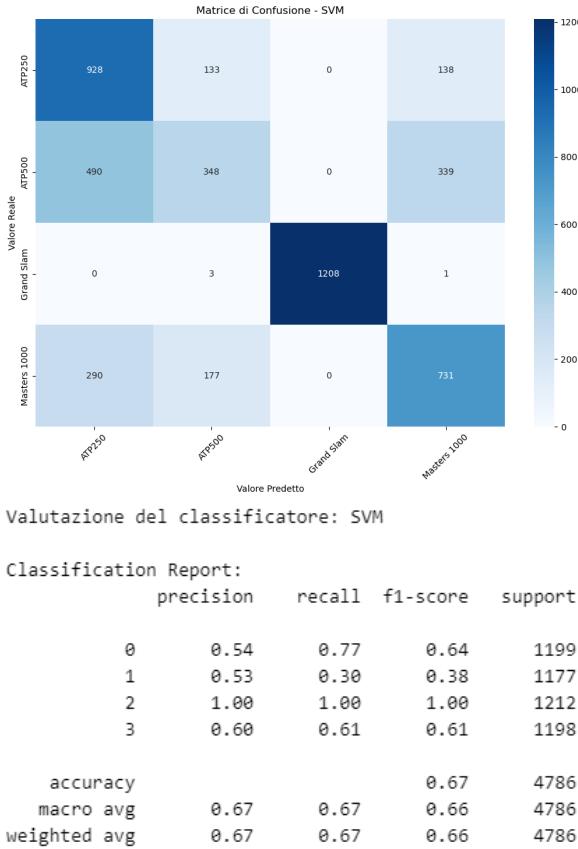


Figura 3.12: Matrice di confusione e metriche: SVM

3.5 T4 - Classificare un giocatore in base alla sua superficie preferita

Per il quarto ed ultimo task di classificazione si è voluto provare a classificare un giocatore in base alle sue caratteristiche, in 3 possibili classi corrispondenti alla sua superficie preferita di gioco:

- Clay: giocatore più adatto alla terra rossa.
- Hard: giocatore che predilige la superficie cemento.
- Grass: giocatore che performa meglio su erba naturale.

3.5.1 Preparazione dei dati

Dopo le operazioni di ETL già ampiamente descritte nei task di classificazione precedenti, in quest'ultimo c'è stato un cambio di prospettiva, passando ad un elenco di giocatori sulle righe del dataframe. In questo modo, l'attenzione del task si è spostata su dati storici ed aggregati relativi ai match di quest'ultimi, e non sui match stessi come nei modelli precedenti.

Per ottenere un elenco unico di giocatori presenti nel dataset, sono stati eseguiti i seguenti passaggi:

1. I nomi dei giocatori sono stati estratti dalle colonne P1_name e P2_name.
2. Sono stati rimossi i duplicati e ordinati alfabeticamente per ottenere un set unico.
3. È stato creato un dataframe, `players_df`, contenente un totale di 1358 giocatori unici.

Questo nuovo dataframe servirà per analisi successive, come lo studio delle prestazioni individuali o la creazione di feature aggiuntive.

3.5.2 Generazione e filtraggio del dataframe finale dei giocatori

Per arricchire l'analisi e costruire un dataset utile per le successive fasi di modellazione, è stato generato un dataframe con informazioni estese sui giocatori e applicati filtri basati su criteri specifici.

Generazione del dataframe finale Il dataframe finale dei giocatori è stato creato tramite la funzione `generate_features_and_target()`, che calcola le seguenti feature:

- **Vittorie per superficie:** Numero di vittorie su terra (`Clay_wins`), erba (`Grass_wins`) e cemento (`Hard_wins`).
- **Vittorie per tipo di campo:** Numero di vittorie su campi indoor e outdoor (`Indoor_wins`).
- **Partite giocate:** Numero di partite giocate per superficie (`Surface_matches`).
- **Anni attivi:** Calcolo degli anni di attività professionale.
- **Superficie preferita:** Identificazione della superficie con il maggior numero di vittorie.

Il dataframe risultante contiene 1358 righe e 14 colonne.

Filtraggio basato su partite giocate Un ulteriore filtraggio è stato applicato per mantenere solo i giocatori che hanno disputato almeno 30 partite complessive tra terra, erba e cemento. Il filtro è stato implementato tramite la seguente condizione:

$$\text{Clay_matches} + \text{Hard_matches} + \text{Grass_matches} \geq 30$$

In questo modo, si è garantito che nel dataset di addestramento fossero presenti solo giocatori con una mole di dati (quindi di partite giocate) significativa ai fini del task condotto.

3.5.3 Calcolo della superficie preferita

La funzione `calculate_preferred_surface()` è stata utilizzata per determinare la superficie preferita di ciascun giocatore nel dataset. Questo calcolo si basa su una combinazione normalizzata di vittorie e punti ottenuti su tre tipi di superfici: terra battuta (`Clay`), cemento (`Hard`) ed erba (`Grass`). Di seguito viene illustrato il funzionamento della funzione.

1. Somma normalizzata di vittorie e punti Per ogni superficie, è stata calcolata una somma normalizzata che combina il numero di vittorie e i punti accumulati³, al fine di ottenere un punteggio rappresentativo per ciascun giocatore:

$$\text{Clay_score} = \frac{\text{Clay_wins} + \text{Clay_points}}{250}$$

³Il calcolo dei punti è stato effettuato assegnando un valore specifico a ogni turno del torneo, in base alla categoria dell'evento (*Grand Slam, Masters 1000, ATP500, ATP250*). Questa struttura riflette il sistema ufficiale di punteggio ATP, che premia i giocatori in base alla difficoltà e al prestigio del torneo. Integrare i punti accumulati consente di catturare non solo il numero di vittorie, ma anche il contesto competitivo in cui esse sono state ottenute. Ad esempio, una vittoria in un *Grand Slam* ha un peso significativamente maggiore rispetto a una in un *ATP250*, evidenziando i giocatori che eccellono nei tornei più importanti. Questo approccio migliora l'accuratezza del modello, poiché i punti rappresentano una metrica aggregata della qualità delle performance su diverse superfici.

$$\text{Hard_score} = \frac{\text{Hard_wins} + \text{Hard_points}}{250}$$

$$\text{Grass_score} = \frac{\text{Grass_wins} + \text{Grass_points}}{250}$$

Questa normalizzazione garantisce che il contributo di vittorie e punti sia proporzionale e comparabile tra le superfici.

2. Determinazione della superficie preferita Per ciascun giocatore, la superficie preferita è stata identificata come quella con il punteggio massimo tra Clay_score, Hard_score e Grass_score:

$$\text{Preferred_surface} = \text{argmax}(\text{Clay_score}, \text{Hard_score}, \text{Grass_score})$$

Feature selezionate Per ridurre la ridondanza e mantenere le informazioni più rilevanti, al termine di un'analisi qualitativa svolta attraverso l'uso della matrice di correlazione (Figura 3.13 sono state selezionate le seguenti feature:

- **Clay_ratio, Hard_ratio, Grass_ratio:** Rapporto tra le vittorie e le partite giocate su terra, cemento ed erba. Queste metriche normalizzano le performance dei giocatori su ciascuna superficie.
- **Clay_wins, Grass_wins, Hard_wins:** Numero totale di vittorie ottenute su terra, erba e cemento. Questi valori riflettono il successo complessivo su ciascuna superficie.

Queste feature combinano quantità assolute (numero di vittorie) e relative (percentuali), migliorando la rappresentazione delle performance dei giocatori su diverse superfici.

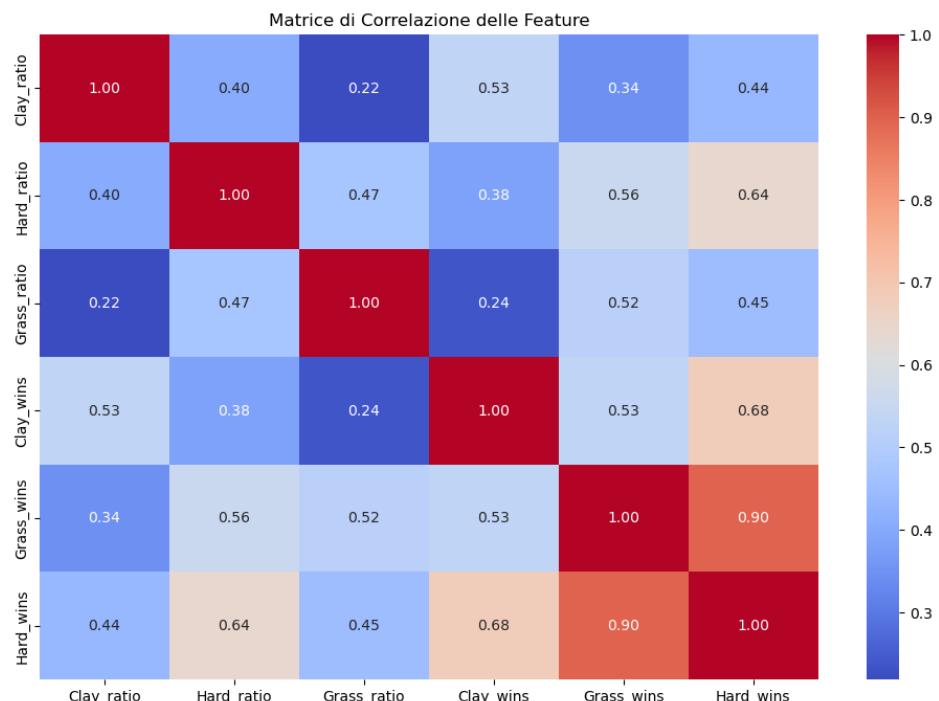


Figura 3.13: Matrice di correlazione delle feature relative al task in questione

3.5.4 Bilanciamento del dataset con SMOTE

Per gestire lo squilibrio delle classi nel dataset, è stata applicata la SMOTE (Synthetic Minority Oversampling Technique), tecnica che genera nuovi campioni sintetici per le classi meno rappresentate.

Identificazione delle classi minoritarie La distribuzione iniziale delle classi è stata analizzata per identificare la classe minoritaria e stabilire un numero desiderato di campioni per ogni classe

Applicazione di SMOTE

- **Training set:** È stato applicato SMOTE al *training set* per generare campioni sintetici, aumentando la dimensione delle classi meno rappresentate.
- **Test set:** Un'operazione simile è stata effettuata sul *test set*, garantendo che il bilanciamento fosse mantenuto anche nei dati di validazione.

Distribuzione delle Classi dopo SMOTE Di seguito si riportano i valori (Figura 3.14) di output della distribuzione finale dei set di training e test, dopo l'applicazione di SMOTE.

- **Training set:** Ogni classe ha raggiunto una distribuzione bilanciata di 1000 campioni.
- **Test set:** Ogni classe ha raggiunto una distribuzione bilanciata di 250 campioni.

```
Classe minoritaria identificata: 2
Distribuzione dei target nel test set (non modificato): Counter({1: 72, 0: 29, 2: 1})
Distribuzione dei target nel test set dopo SMOTE: Counter({1: 250, 0: 250, 2: 250})
Distribuzione dei target nel train set dopo SMOTE: Counter({0: 1000, 1: 1000, 2: 1000})
```

Figura 3.14: Valori di output dopo l'applicazione di SMOTE su train set e test set

3.5.5 Addestramento e valutazione dei modelli

Sono stati testati diversi modelli di machine learning per valutare le loro performance nella classificazione delle superfici preferite. I modelli considerati includono tecniche lineari, basate su alberi e metodi di vicinato.

Modelli considerati I seguenti modelli sono stati implementati:

- **Random Forest:** Un ensemble di alberi decisionali per catturare relazioni non lineari.
- **Gradient Boosting:** Un metodo iterativo per ottimizzare le predizioni.
- **Support Vector Machine (SVM):** Un modello a margine massimo con probabilità.
- **Logistic Regression:** Un modello lineare per la classificazione.
- **K-Nearest Neighbors (KNN):** Un metodo basato sulle distanze tra i punti.

3.5.6 Valutazione delle performance

Nelle figure che seguono (Figure 3.15, 3.16, 3.17) sono descritti i risultati principali dei classificatori utilizzati:

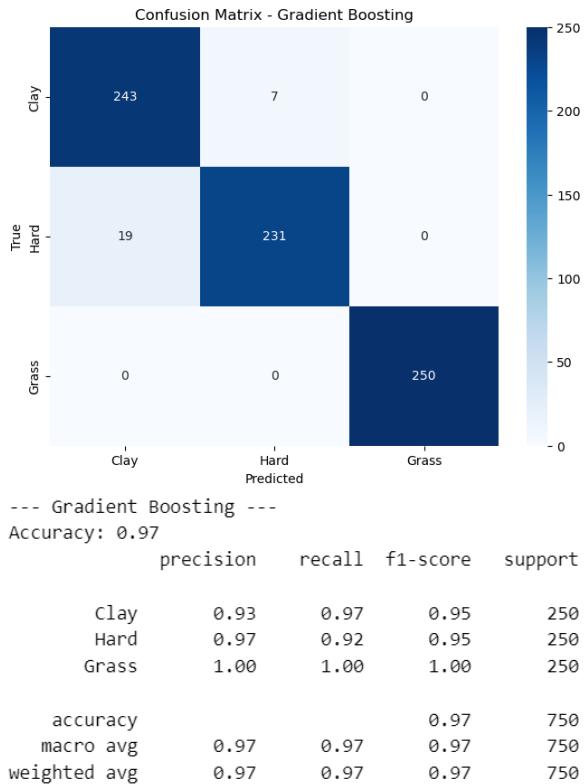


Figura 3.15: Risultati: Gradient Boosting

Gradient Boosting

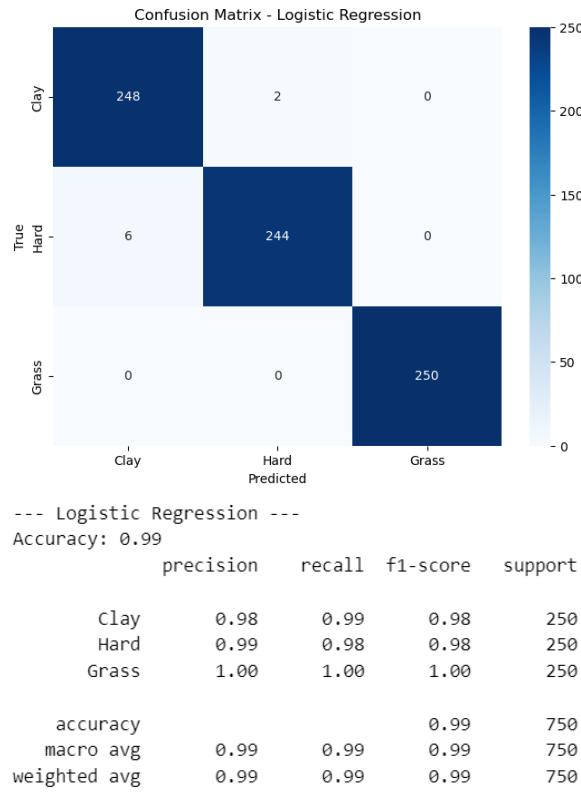
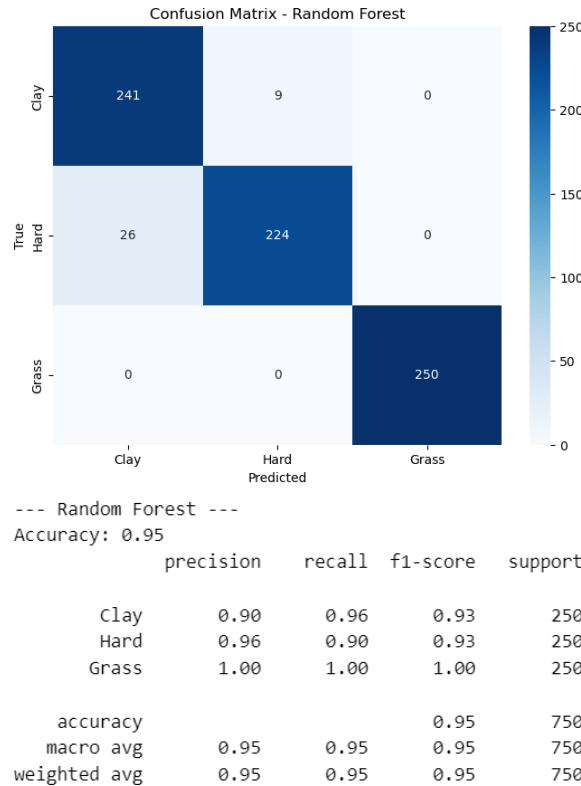
- **Accuracy:** 97%
- **Prestazioni per classe:** Precision e recall elevati per tutte le superfici, con valori perfetti per la classe Grass.
- **Osservazioni:** Ha evidenziato una leggera confusione tra Clay e Hard, ma nel complesso è uno dei modelli più affidabili.

Logistic Regression

- **Accuracy:** 99%
- **Prestazioni per classe:** Ottime performance su tutte le superfici, con precision e recall quasi perfetti.
- **Osservazioni:** La semplicità del modello e il bilanciamento dei dati hanno contribuito a risultati eccellenti.

Random Forest

- **Accuracy:** 95%
- **Prestazioni per classe:** Performance equilibrate con lievi errori nella distinzione tra Clay e Hard.
- **Osservazioni:** La robustezza del modello si conferma, anche se meno precisa rispetto al Gradient Boosting.

**Figura 3.16:** Risultati: Logistic Regression**Figura 3.17:** Risultati: Random Forest

Osservazioni finali

- I modelli hanno ottenuto elevate accuratezze grazie al bilanciamento dei dati tramite SMOTE e alla riduzione della ridondanza nelle feature.
- **Gradient Boosting** e **Random Forest** si confermano robusti per la gestione di relazioni non lineari, mentre **Logistic Regression** ha beneficiato della semplicità del task.
- Gli errori riscontrati riguardano principalmente la confusione tra superfici simili, come Clay e Hard, a causa della loro maggiore somiglianza nei pattern dei dati.

3.6 Conclusioni e sviluppi futuri

Nel corso di questo lavoro, sono stati affrontati diversi task di classificazione, ciascuno con obiettivi specifici e strategie mirate per ottimizzare i risultati. Di seguito, una sintesi complessiva delle analisi svolte e delle principali osservazioni.

3.6.1 Valutazione Complessiva dei Task di Classificazione

T1 - Classificare il vincitore di una partita disputata tra due giocatori Il primo task ha riguardato la predizione del vincitore di una partita, basandosi su feature come ranking ATP, punti accumulati e differenze di ranking. I risultati ottenuti hanno evidenziato come modelli basati su ensemble (ad esempio, Random Forest) siano particolarmente efficaci nel catturare le relazioni non lineari tra le variabili. Tuttavia, la presenza di dati con distribuzioni non bilanciate ha richiesto tecniche di filtraggio e preprocessing approfonditi.

T2 - Classificare il risultato (in numero di set disputati) di una partita tra due giocatori Il secondo task ha affrontato un problema più complesso: prevedere il numero di set in una partita. Questo task si è rivelato sfidante a causa dell'elevata variabilità dei dati e della mancanza di feature più dettagliate sullo stile di gioco dei giocatori o sulle condizioni ambientali. Nonostante ciò, i modelli hanno mostrato risultati accettabili, evidenziando la necessità di ulteriori dati per migliorare la precisione delle predizioni.

T3 - Classificare un torneo in base alle categorie disponibili (ATP250, ATP500, Masters1000, Grand Slam) Il terzo task ha riguardato la classificazione delle categorie di tornei (ATP250, ATP500, Masters 1000, Grand Slam). Qui, il bilanciamento delle classi è stato cruciale per evitare il predominio delle categorie più rappresentate. Modelli come Gradient Boosting e Random Forest hanno nuovamente dimostrato la loro efficacia, sebbene la mancanza di feature aggiuntive, come il montepremi o il livello di partecipazione dei top players, abbia limitato le capacità predittive.

T4 - Classificare un giocatore in base alla sua superficie preferita Il quarto ed ultimo task si è concentrato sull'identificazione della superficie preferita dei giocatori (terra, erba o cemento). Grazie a feature costruite ad hoc, come il rapporto vittorie/partite per superficie e il calcolo dei punti in base ai round e alla categoria dei tornei, i modelli hanno raggiunto performance eccellenti. Logistic Regression e Gradient Boosting si sono dimostrati particolarmente efficaci, con accuracy superiori al 95%.

3.6.2 Sviluppi futuri

Per migliorare ulteriormente i risultati, si propongono i seguenti sviluppi:

- **Espansione del dataset:** Integrare feature aggiuntive come il montepremi, statistiche in tempo reale e condizioni ambientali per arricchire le analisi.
- **Ottimizzazione dei modelli:** Esplorare architetture avanzate, come le reti neurali profonde, per affrontare task più complessi.

CAPITOLO 4

CLUSTERING

4.1 Cos'è il clustering?

Il clustering è una tecnica di apprendimento automatico non supervisionato che mira a raggruppare un insieme di dati in gruppi o cluster, in modo tale che i punti all'interno dello stesso cluster siano più simili tra loro rispetto a quelli di cluster diversi. Questa tecnica è ampiamente utilizzata in settori come il marketing, la biologia, l'analisi delle immagini e molto altro. Gli obiettivi principali del clustering includono:

- Scoprire strutture nascoste nei dati.
- Ridurre la dimensionalità dei dati.
- Identificare pattern utili per analisi successive.

Tipologie di clustering

Esistono vari tipi di algoritmi di clustering, ognuno con specifiche caratteristiche e applicazioni:

- **Clustering Partizionale:** Suddivide i dati in un numero predefinito di cluster, ottimizzando una funzione obiettivo. Esempio: K-Means.
- **Clustering Gerarchico:** Costruisce una struttura gerarchica di cluster, che può essere rappresentata da un dendrogramma. Esempio: Agglomerative Clustering.
- **Clustering Basato sulla Densità:** Identifica cluster basandosi sulle regioni ad alta densità di punti nei dati. Esempio: DBSCAN, HDBSCAN.
- **Clustering Basato su Modelli:** Assume che i dati siano generati da una combinazione di modelli probabilistici e raggruppa i punti in base a tali modelli. Esempio: Gaussian Mixture Model (GMM).

4.1.1 Approfondimento: K-Means

K-Means è uno degli algoritmi di clustering più popolari e semplici. Funziona suddividendo i dati in un numero predefinito k di cluster. L'algoritmo segue i seguenti passi:

1. Inizializza k centroidi casualmente.
2. Assegna ciascun punto al cluster con il centroide più vicino.
3. Aggiorna i centroidi come la media dei punti assegnati a ciascun cluster.
4. Ripeti i passi 2 e 3 fino a convergenza.

Vantaggi:

- Semplice e veloce.
- Efficace per cluster sferici e ben separati.

Svantaggi:

- Richiede il numero di cluster k come input.
- Sensibile ai centroidi iniziali e ai valori anomali.

4.1.2 Approfondimento: DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) è un algoritmo di clustering basato sulla densità che identifica cluster come regioni ad alta densità di punti. Funziona secondo i seguenti principi:

- Un punto è considerato **core** se contiene almeno un numero minimo di punti `min_samples` entro una distanza `eps`.
- I punti vicini a un punto core sono inclusi nel suo cluster.
- I punti che non appartengono a nessun cluster sono considerati rumore.

Vantaggi:

- Non richiede il numero di cluster a priori.
- Gestisce cluster di forma arbitraria e identifica il rumore.

Svantaggi:

- Sensibile alla scelta dei parametri `eps` e `min_samples`.
- Non funziona bene con cluster di densità variabile.

4.1.3 Approfondimento: HDBSCAN

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) è un’estensione di DBSCAN che migliora le sue capacità:

- Utilizza una struttura gerarchica per identificare cluster con densità variabile.
- Trova automaticamente il numero ottimale di cluster.

Vantaggi:

- Adatto a dati con densità variabile.
- Identifica il rumore in modo efficace.

Svantaggi:

- Più complesso e computazionalmente intensivo rispetto a DBSCAN.
- Può essere sensibile alla scelta del parametro `min_cluster_size`.

Il clustering è una tecnica potente per esplorare e analizzare dati non etichettati. Ogni algoritmo ha vantaggi e svantaggi che lo rendono adatto a specifici scenari. La scelta dell'algoritmo migliore dipende dalla natura dei dati e dagli obiettivi dell'analisi.

4.2 ETL sui dati

In questa sezione analizzeremo il codice creato per il preprocessing dei dati. Si è deciso di effettuare clustering su singoli giocatori.

Il codice inizia con l'importazione delle librerie necessarie per la gestione e l'analisi dei dati, per poi proseguire caricando il dataset utilizzando pandas. Il file `atp_tennis.csv` contiene informazioni sulle partite di tennis, come punteggi, classifiche e punti dei giocatori.

Una volta stampata un'anteprima del dataset e le sue informazioni generali (numero di righe, colonne e tipi di dati) per identificare eventuali problemi, come valori mancanti o formati errati, si prosegue con la pulizia preliminare dei dati:

1. Identificazione e rimozione di valori non validi:

```
1 data[numerical_columns] = data[numerical_columns].replace(0, np.nan)
2 data_cleaned = data.dropna(subset=numerical_columns)
3
```

- Le colonne numeriche `Rank_1`, `Rank_2`, `Pts_1` e `Pts_2` vengono esaminate per valori non validi (`NaN` o `0`).
- Gli zeri vengono convertiti in `NaN` e le righe contenenti valori non validi vengono rimosse.

2. Verifica dei valori non validi:

```
1 invalid_counts_before = data[numerical_columns].isna().sum()
   + (data[numerical_columns] == 0).sum()
2 invalid_counts_after = data_cleaned[numerical_columns].isna().sum()
```

Il codice calcola e stampa il numero di valori non validi prima e dopo la pulizia.

Valori non validi (`NaN` o `0`) prima della pulizia:

<code>Rank_1</code>	14
<code>Rank_2</code>	12
<code>Pts_1</code>	15652
<code>Pts_2</code>	15653
<code>dtype</code> :	<code>int64</code>

Valori non validi (`NaN` o `0`) dopo la pulizia:

<code>Rank_1</code>	0
---------------------	---

```

Rank_2      0
Pts_1       0
Pts_2       0
dtype: int64

Dati puliti: righe rimosse dove le colonne numeriche
avevano NaN o 0
Righe originali: 64166, Righe dopo pulizia: 48513

```

3. Verifica della colonna Score:

```

1 score_pattern = r'^(\d+-\d+\s?)+$'
2 data_cleaned = data_cleaned[data_cleaned['Score'].notna()]
3 data_cleaned = data_cleaned[data_cleaned['Score'].str.strip()
                               () != ""]
4 data_cleaned = data_cleaned[data_cleaned['Score'].str.match
                               (score_pattern)]

```

- La colonna Score viene controllata per valori mancanti o vuoti.
- Viene utilizzata un'espressione regolare per verificare che i punteggi siano in un formato valido (esempio: 6–4 6–3).
- Le righe con punteggi non validi vengono rimosse.

Controllo della colonna 'Score':

Numero di valori non validi (NaN o vuoti): 0
 Numero di punteggi con formato non valido: 0
 Righe dopo la pulizia della colonna 'Score': 48513

4.2.1 Scelta delle feature di interesse

Questa sezione descrive il processo di selezione, creazione e filtraggio delle caratteristiche utilizzate per il clustering dei tennisti basato sui dati delle partite. Verranno analizzate le trasformazioni applicate ai dati grezzi per generare informazioni rilevanti come ranking, punti, prestazioni per superficie e vittorie nei tornei principali.

4.2.2 Creazione delle feature

1. Ranking massimo e medio

Il ranking massimo e medio viene calcolato per ciascun giocatore considerando le colonne Rank_1 e Rank_2. Ovviamente è stato necessario analizzare entrambe le colonne per poi unire i risultati di ogni giocatore in un valore univoco.

```

1 data_cleaned['Max_Rank_1'] = data_cleaned.groupby('Player_1')
2 ['Rank_1'].transform('min')
3 data_cleaned['Max_Rank_2'] = data_cleaned.groupby('Player_2')
4 ['Rank_2'].transform('min')
5 data_cleaned['Avg_Rank_1'] = data_cleaned.groupby('Player_1')
6 ['Rank_1'].transform('mean')
7 data_cleaned['Avg_Rank_2'] = data_cleaned.groupby('Player_2')
8 ['Rank_2'].transform('mean')

```

I dati di entrambe le colonne sono aggregati per ottenere un unico valore di ranking massimo e medio per ciascun giocatore.

2. Punteggio massimo e medio

Il punteggio massimo e medio viene calcolato analogamente al ranking, usando le colonne Pts_1 e Pts_2. L'unica differenza da notare sta nel fatto che il valore dei dati ottenuti sono inversi rispetto ai dati precedenti: un punteggio massimo molto alto corrisponde ad un rank massimo molto basso.

3. Numero totale di partite giocate e di vittorie ottenute

Il numero totale di partite giocate da ciascun giocatore è calcolato sommando la presenza nelle colonne Player_1 e Player_2.

```
1 total_matches = pd.concat([data_cleaned['Player_1'],
                             data_cleaned['Player_2']]).value_counts()
```

Il numero totale di vittorie è calcolato utilizzando la colonna Winner.

```
1 total_wins = data_cleaned['Winner'].value_counts()
```

Percentuale di vittorie: calcolata come il rapporto tra Total_Wins e Total_Matches:

```
1 win_percentage['Win_Percentage'] = win_percentage['Total_Wins']
                                    / win_percentage['Total_Matches'] * 100
```

4. Performance per superficie

Le vittorie per superficie (terra, erba, cemento) sono calcolate utilizzando una tabella di contingenza:

```
1 surface_wins_1 = pd.crosstab(data_cleaned['Player_1'],
                                data_cleaned['Surface'])
2 surface_wins_2 = pd.crosstab(data_cleaned['Player_2'],
                                data_cleaned['Surface'])
```

Sono state, dunque, contate, per ognuna delle tre superfici di gioco, le vittorie ottenute da ogni giocatore. Di seguito, le **percentuali di vittorie per superficie** sono state calcolate dividendo le vittorie totali per superficie per il numero totale di partite giocate su ognuna di esse.

5. Prestazioni nei tornei principali

Si è deciso di analizzare anche le partite vinte nei Grandi Slam e nei Masters 1000, calcolate separatamente e poi aggregate per ciascun giocatore. Sapevamo, tuttavia, di andare incontro a molti valori nulli.

Unione delle feature

Tutte le feature calcolate vengono unite in un unico dataset chiamato player_summary, che rappresenta il dataset finale per il clustering. Le prime 5 righe del dataset sono visualizzabili nelle Figure 4.1 e 4.2.

```

1 player_summary = max_rank.merge(avg_rank, on='Player', how='left')
2 player_summary = player_summary.merge(max_points, on='Player',
3     how='left')
...

```

Player	Max_Rank	Avg_Rank	Max_Points	Avg_Points	Total_Matches_x	Total_Wins	Win_Percentage	Wins_Clay	Wins_Grass	Wins_Hard
Hajek J.	79.0	79.000000	506.0	506.000000	2	1.0	50.000000	2.0	0.0	0.0
Acasuso J.	20.0	47.969072	1640.0	885.463918	194	103.0	53.092784	113.0	0.0	81.0
Adaktusson J.	265.0	265.000000	133.0	133.000000	1	NaN	NaN	0.0	0.0	1.0
Agamenone F.	136.0	140.000000	407.0	399.800000	5	3.0	60.000000	5.0	0.0	0.0
Agassi A.	5.0	12.513514	2275.0	1830.540541	37	26.0	70.270270	0.0	4.0	33.0

Figura 4.1: Feature scelte per il clustering dei tennisti

Win_Percentage_Clay	Win_Percentage_Hard	Win_Percentage_Grass	Grand_Slam_Wins	Masters_1000_Wins
1.000000	0.000000	0.000000	NaN	NaN
0.582474	0.417526	0.000000	8.0	1.0
0.000000	1.000000	0.000000	NaN	NaN
1.000000	0.000000	0.000000	0.0	0.0
0.000000	0.891892	0.108108	10.0	0.0

Figura 4.2: Feature scelte per il clustering dei tennisti - 2

Dopo una prima analisi esplorativa del nuovo dataset, è emerso che alcune statistiche, come le percentuali di vittorie su determinate superfici o in specifici tornei, potevano risultare poco significative se calcolate su un numero esiguo di partite giocate. Per evitare che tali dati ininfluenti compromettessero la qualità delle analisi, si è deciso di escludere dal dataset i giocatori con meno di 30 partite disputate. Questo filtraggio ha comportato una riduzione del numero di giocatori analizzati, ma ha garantito maggiore coerenza e affidabilità ai risultati successivi:

Numero di giocatori prima del filtraggio: 1369

Numero di giocatori dopo il filtraggio: 446

Le feature calcolate (ranking, punti, performance per superficie e vittorie nei tornei principali) costituiscono una base solida per applicare tecniche di clustering, fornendo una rappresentazione dettagliata delle caratteristiche di ciascun giocatore.

Questo step è essenziale per garantire la qualità dei dati prima di procedere con il clustering.

4.3 Analisi esplorativa dei dati

Prima di procedere con l’analisi vera e propria, sono stati descritti e analizzati alcuni aspetti principali del dataset relativo ai giocatori di tennis, utilizzando visualizzazioni grafiche per comprendere meglio le caratteristiche e le relazioni tra le variabili.

Distribuzione del numero di vittorie

Per esplorare la distribuzione del numero totale di vittorie per giocatore, è stato creato un istogramma con una curva di densità stimata (*Kernel Density Estimate, KDE*). Questo grafico, mostrato in Figura 4.3, consente di identificare come le vittorie siano distribuite tra i giocatori:

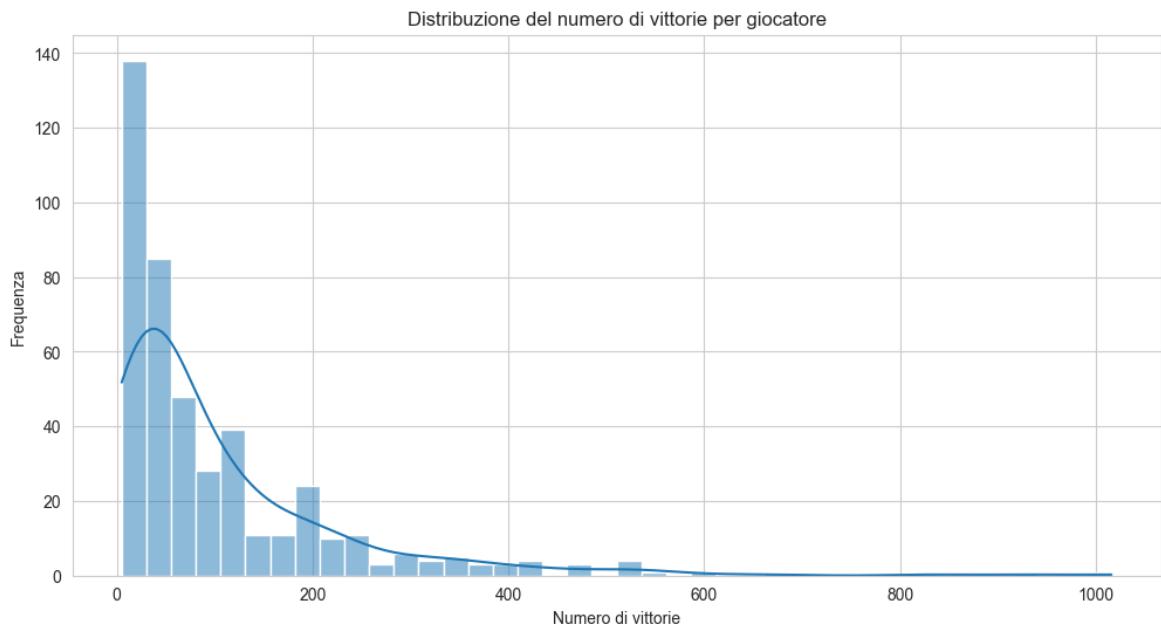


Figura 4.3: Distribuzione del numero di vittorie per giocatore

La distribuzione evidenzia che la maggior parte dei giocatori ha un numero limitato di vittorie, mentre solo pochi riescono a raggiungere un numero considerevole.

Relazione tra Ranking e Vittorie

Sono state analizzate due relazioni principali:

- La relazione tra il miglior ranking raggiunto da un giocatore e la sua percentuale di vittorie (Figura 4.4).
- La relazione tra il miglior ranking raggiunto e il numero totale di vittorie (Figura 4.5).

Queste relazioni sono state rappresentate utilizzando grafici esagonali (*hexbin*) per evidenziare le densità delle osservazioni.

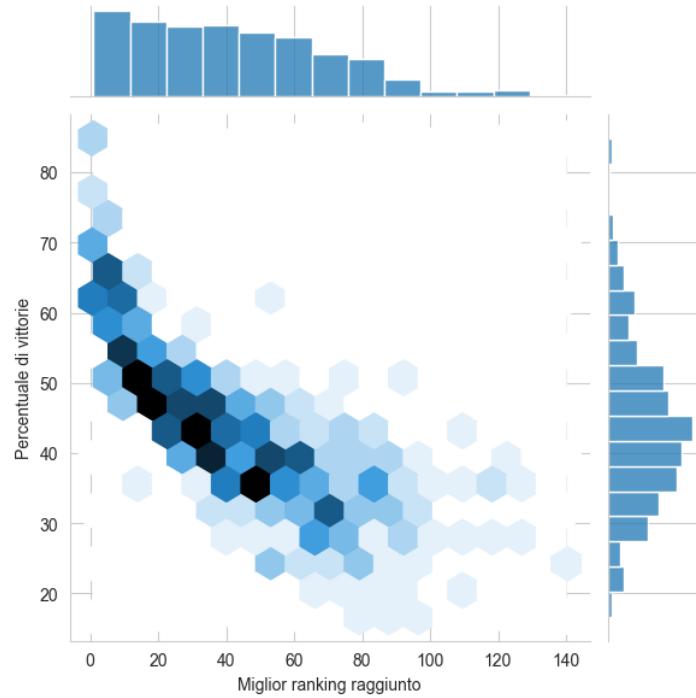


Figura 4.4: Relazione tra miglior ranking e percentuale di vittorie

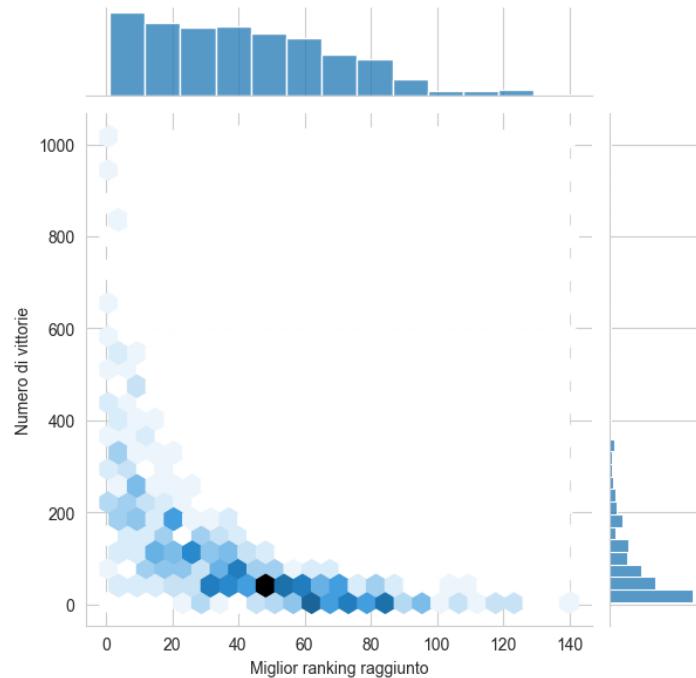


Figura 4.5: Relazione tra miglior ranking e numero di vittorie

Da questi grafici si osserva che i giocatori con un miglior ranking tendono ad avere percentuali di vittorie e numeri di vittorie complessivamente più alti.

Distribuzione delle vittorie per superficie

Per esplorare la distribuzione delle vittorie dei giocatori sulle diverse superfici (terra rossa, erba e cemento), è stato creato un *boxplot*, mostrato in Figura 4.6. Inoltre, è stato calcolato il numero totale di partite giocate su ciascuna superficie, come riportato di seguito:

- **Terra rossa:** 28.704 partite.
- **Erba:** 10.165 partite.
- **Cemento:** 53.158 partite.

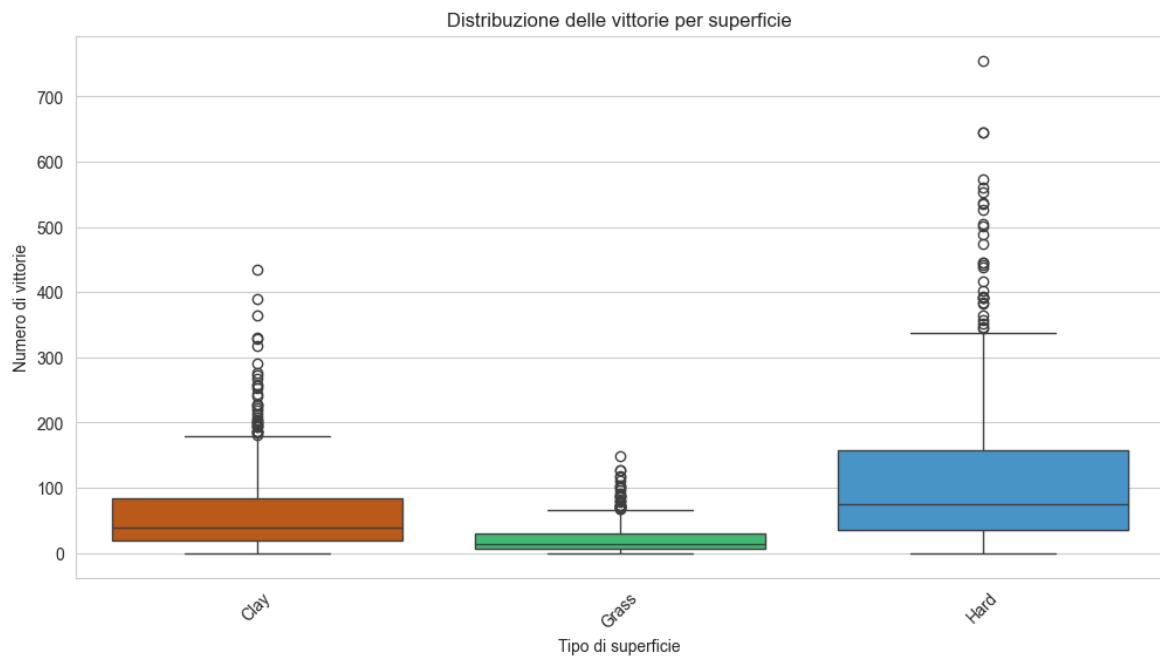


Figura 4.6: Distribuzione delle vittorie per superficie

Dal grafico si nota come i giocatori tendano a ottenere un numero maggiore di vittorie su superfici specifiche, probabilmente a causa di preferenze o abilità tecniche: nello specifico, c'è un elevato numero di vittorie su cemento, rispetto alle altre due tipologie di superficie.

Top 10 giocatori per numero di partite

Infine, è stato analizzato il numero totale di partite giocate dai tennisti, evidenziando i primi 10 giocatori in termini di match disputati. L'istogramma risultante (Figura 4.7) utilizza una paletta di colori dal più scuro al più chiaro per rappresentare visivamente la classifica.

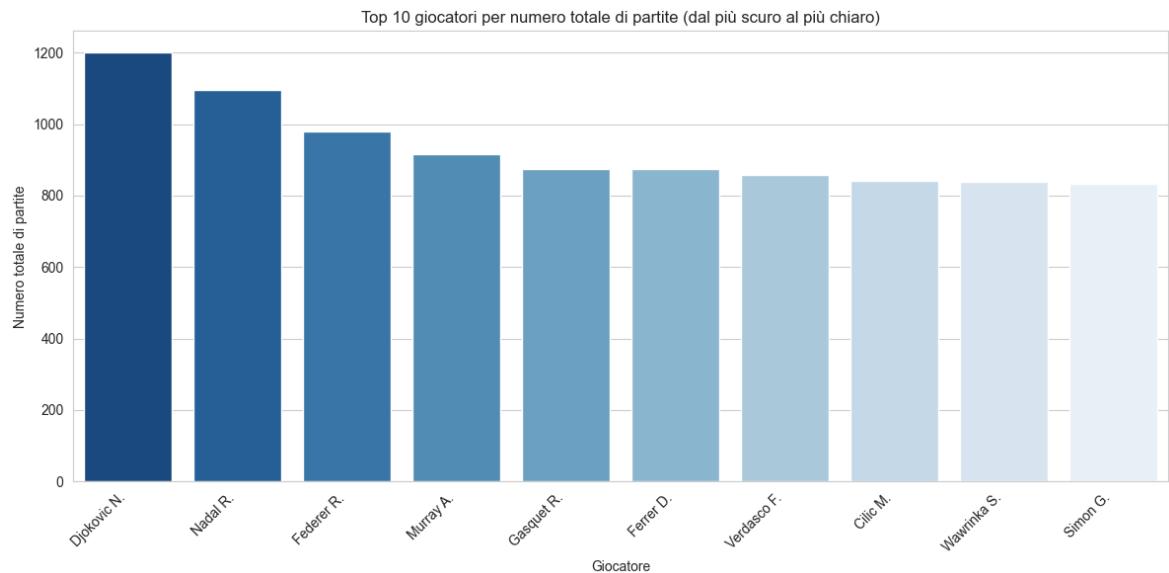


Figura 4.7: Top 10 giocatori per numero totale di partite giocate

Questo grafico mette in evidenza i top 10 giocatori che stanno avendo o hanno avuto una carriera particolarmente lunga e ricca di incontri.

4.4 Analisi delle relazioni tra le variabili

Per esplorare le relazioni tra le feature nel dataset, sono stati utilizzati due principali strumenti grafici e statistici per identificare eventuali correlazioni rilevanti.

4.4.1 Pairplot

Un *pairplot* è una visualizzazione che mostra le relazioni a coppie tra tutte le variabili numeriche di un dataset attraverso grafici a dispersione (*scatterplot*). Lungo la diagonale vengono rappresentati gli istogrammi, che forniscono una panoramica delle distribuzioni di ciascuna variabile (Figura 4.8).

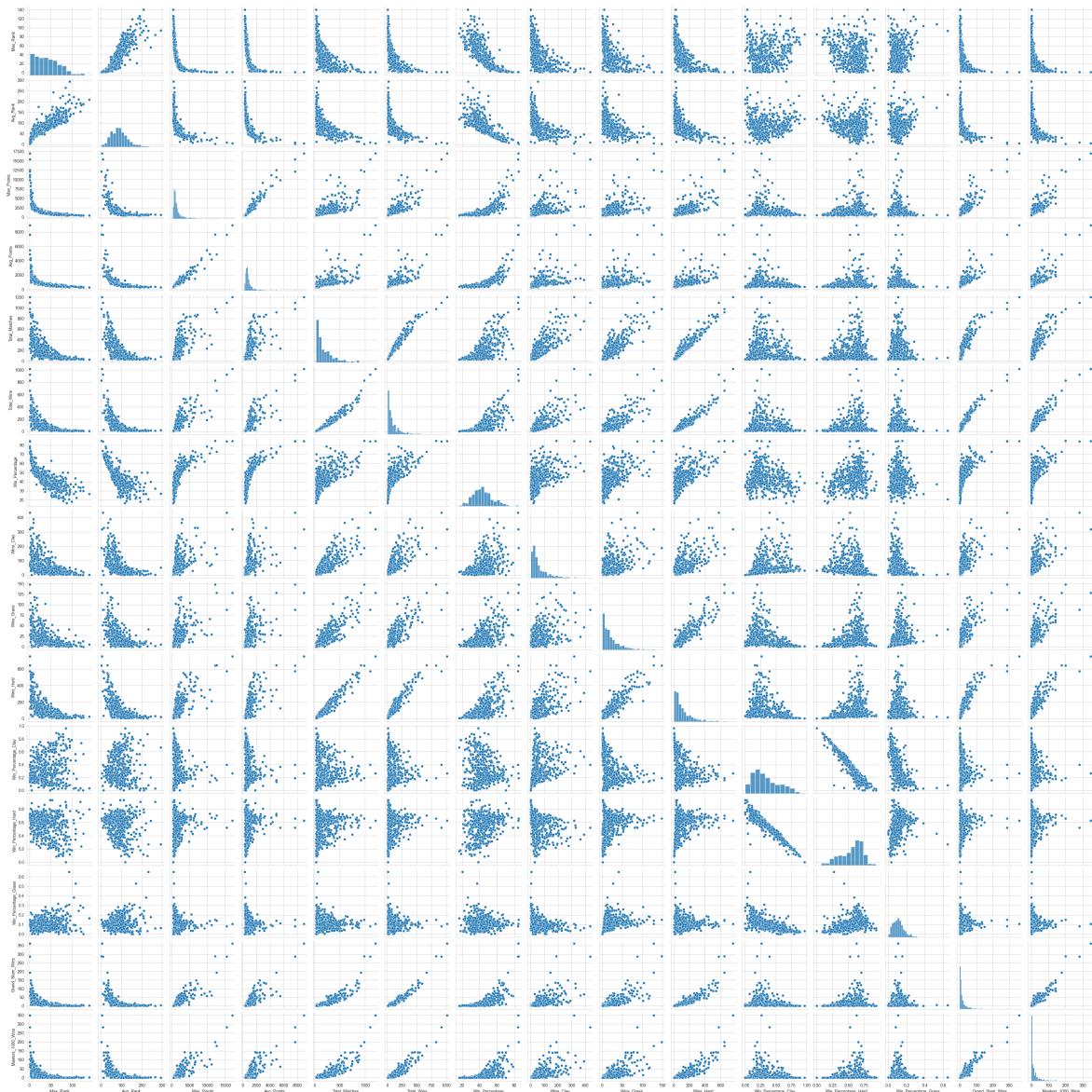
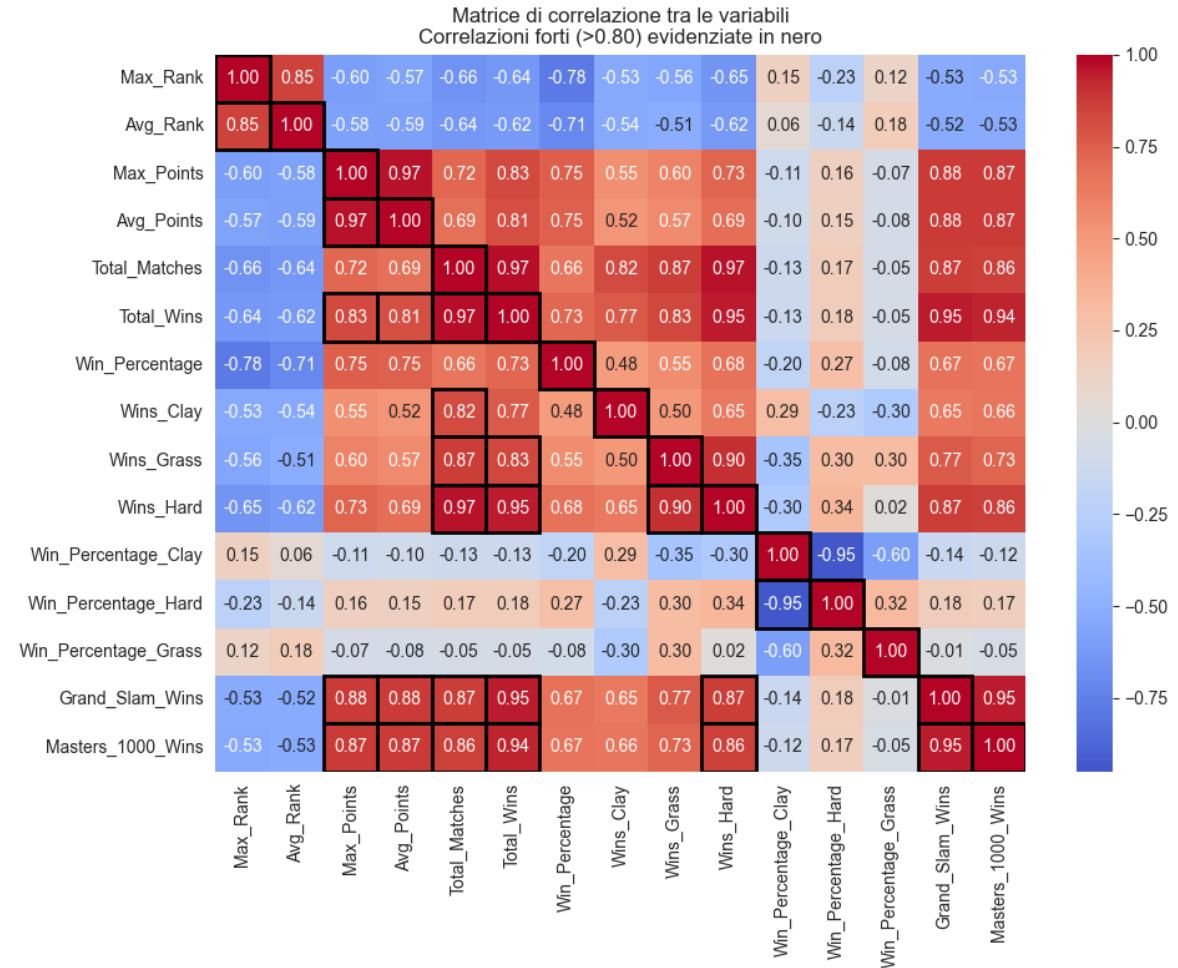


Figura 4.8: Pairplot delle feature numeriche scelte

Dall’osservazione di questo grafico è possibile individuare eventuali relazioni lineari, non lineari, o cluster naturali presenti tra le feature presentate nella Sezione 5.2.

4.4.2 Matrice di correlazione

La matrice di correlazione è un importante strumento statistico che misura la relazione lineare tra due variabili. In questa analisi, sono state calcolate le correlazioni tra tutte le feature (Figura 4.9). La correlazione varia da -1 (relazione inversa perfetta) a +1 (relazione diretta perfetta), mentre valori vicini a 0 indicano l’assenza di correlazione.

**Figura 4.9:** Matrice di correlazione tra le feature

La matrice di correlazione evidenzia:

- Correlazioni deboli (valori vicini a 0).
- Correlazioni forti, con valori assoluti maggiori di 0.80, che sono state evidenziate in nero.

4.4.3 Identificazione delle correlazioni forti

Per identificare correlazioni significative che potrebbero semplificare l'analisi o il modello di clustering, sono state conteggiate e analizzate tutte le correlazioni forti, definite come $|correlazione| > 0.80$. Di seguito vengono riportati i principali risultati:

- **Numero totale di correlazioni possibili:** 105 (calcolato come combinazione delle variabili).
- **Numero di correlazioni forti:** 38.
- **Percentuale di correlazioni forti:** 36.19%.

In Figura 4.10 sono mostrate le coppie di variabili che mostrano correlazioni forti:

```

Correlazioni forti (>0.80):
Max_Rank - Max_Rank: 1.00
Avg_Rank - Max_Rank: 0.85
Avg_Rank - Avg_Rank: 1.00
Max_Points - Max_Points: 1.00
Avg_Points - Max_Points: 0.97
Avg_Points - Avg_Points: 1.00
Total_Matches - Total_Matches: 1.00
Total_Wins - Max_Points: 0.83
Total_Wins - Avg_Points: 0.81
Total_Wins - Total_Matches: 0.97
Total_Wins - Total_Wins: 1.00
Win_Percentage - Win_Percentage: 1.00
Wins_Clay - Total_Matches: 0.82
Wins_Clay - Wins_Clay: 1.00
Wins_Grass - Total_Matches: 0.87
Wins_Grass - Total_Wins: 0.83
Wins_Grass - Wins_Grass: 1.00
Wins_Hard - Total_Matches: 0.97
Wins_Hard - Total_Wins: 0.95
Wins_Hard - Wins_Grass: 0.90
Wins_Hard - Wins_Hard: 1.00
Win_Percentage_Clay - Win_Percentage_Clay: 1.00
Win_Percentage_Hard - Win_Percentage_Clay: -0.95
Win_Percentage_Hard - Win_Percentage_Hard: 1.00
Win_Percentage_Grass - Win_Percentage_Grass: 1.00
Grand_Slam_Wins - Max_Points: 0.88
Grand_Slam_Wins - Avg_Points: 0.88
Grand_Slam_Wins - Total_Matches: 0.87
Grand_Slam_Wins - Total_Wins: 0.95
Grand_Slam_Wins - Wins_Hard: 0.87
Grand_Slam_Wins - Grand_Slam_Wins: 1.00
Masters_1000_Wins - Max_Points: 0.87
Masters_1000_Wins - Avg_Points: 0.87
Masters_1000_Wins - Total_Matches: 0.86
Masters_1000_Wins - Total_Wins: 0.94
Masters_1000_Wins - Wins_Hard: 0.86
Masters_1000_Wins - Grand_Slam_Wins: 0.95
Masters_1000_Wins - Masters_1000_Wins: 1.00

```

Figura 4.10: Correlazioni forti identificate nella correlation matrix

Questi risultati permettono di comprendere meglio le relazioni all'interno del dataset, identificando le variabili che potrebbero essere ridondanti o che richiedono ulteriori analisi. Questo passo è fondamentale per ottimizzare successivamente le tecniche di clustering applicate al dataset. Infatti, prima di procedere con le tecniche di clustering, si sono sfruttati i risultati ottenuti per ridurre le feature da 15 a 6, basandosi sulle correlazioni forti individuate. Le feature selezionate per l'analisi rimangono, dunque:

- **Avg_Rank:** ranking medio del giocatore.
- **Win_Percentage:** percentuale di vittorie.
- **Wins_Clay:** numero di vittorie su terra battuta.
- **Win_Percentage_Hard:** percentuale di vittorie su superficie dura.
- **Win_Percentage_Grass:** percentuale di vittorie su erba.
- **Masters_1000_Wins:** numero di vittorie nei tornei Masters 1000.

4.5 Applicazione della PCA

Per migliorare l'efficacia del clustering e ridurre ulteriormente la dimensionalità del dataset, è stata applicata la **Principal Component Analysis (PCA)** sulle feature rilevanti. I passi effettuati sono mostrati di seguito.

4.5.1 Standardizzazione dei dati

Prima di applicare la PCA, le feature sono state standardizzate utilizzando lo *StandardScaler* per garantire che ogni variabile contribuisca equamente all'analisi, indipendentemente dalla sua scala originale.

4.5.2 Applicazione della PCA

La **Principal Component Analysis** è stata applicata per trasformare il dataset originale in un nuovo spazio di dimensioni ridotte. Questa tecnica permette di ridurre la dimensionalità mantenendo la maggior parte della varianza spiegata: nel nostro caso si è deciso di mantenere l'80% di questa, come mostrato in Figura 4.11.

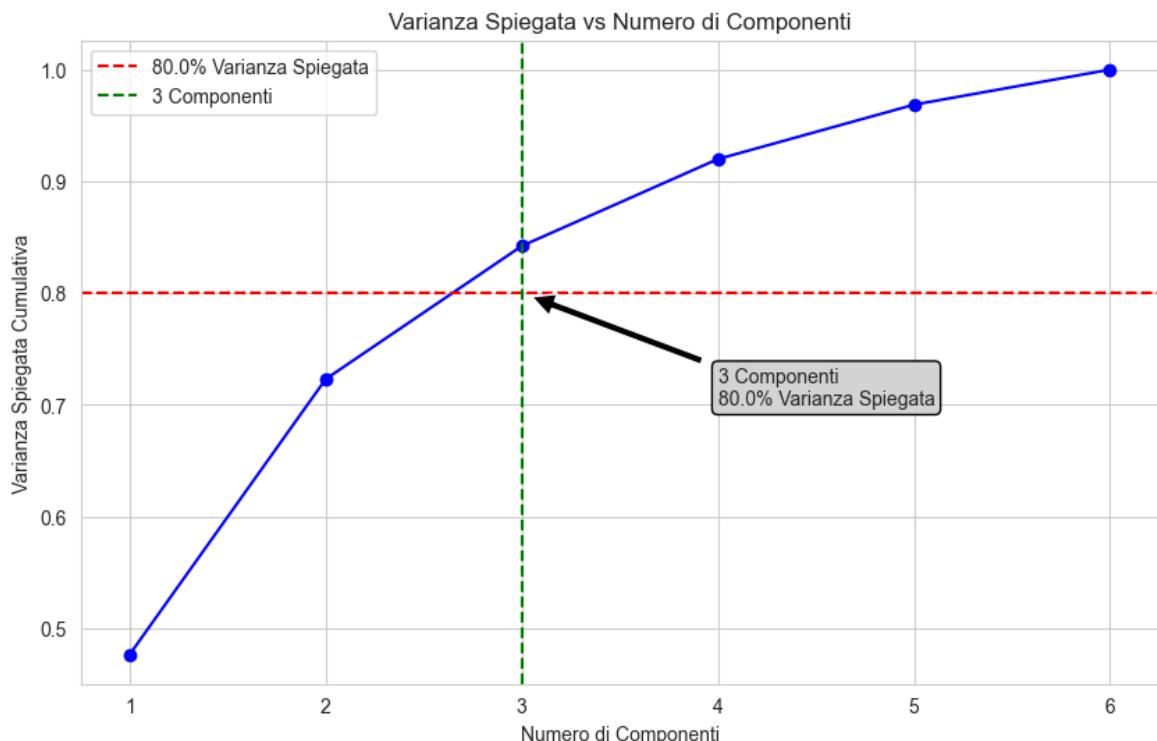


Figura 4.11: Varianza cumulativa rispetto al # di componenti

Il grafico rappresenta la varianza spiegata cumulativa in funzione del numero di componenti principali. La linea rossa indica una soglia dell'80% di varianza spiegata, mentre la linea verticale verde evidenzia il numero di componenti necessarie per raggiungere questa soglia. Nel nostro caso, dunque, il numero ottimale è pari a **3 componenti principali** per trattenere, nello specifico, l'**84.21% della varianza**.

4.5.3 Varianza spiegata dalle componenti

La varianza spiegata da ognuna delle 6 componenti principali è riportata nella Tabella 4.1.

Tabella 4.1: Varianza spiegata da ogni componente principale

Componente Principale	Varianza Spiegata (%)
PC1	47.6%
PC2	24.7%
PC3	11.9%
PC4	7.8%
PC5	4.9%
PC6	3.1%

4.5.4 Matrice delle componenti

La matrice delle componenti principali, riportata nella Tabella 4.2, mostra i pesi associati a ciascuna variabile in ognuna delle 6 componenti principali. Essa è fondamentale per comprendere come le feature originali contribuiscono a ogni componente principale.

Tabella 4.2: Matrice delle componenti principali

Componente	Avg_Rank	Win_Percentage	Wins_Clay	Win_Percentage_Hard	Win_Percentage_Grass	Masters_1000_Wins
PC1	-0.492943	0.508953	0.467994	0.073825	-0.147509	0.501743
PC2	-0.051064	0.203203	-0.307200	0.706875	0.593235	0.100646
PC3	0.239731	-0.146994	0.365312	-0.424512	0.712412	0.315798
PC4	0.640939	-0.194751	0.183103	0.393381	-0.336794	0.499563
PC5	0.379051	0.657283	-0.495103	-0.365967	-0.047982	0.207216
PC6	-0.377421	-0.456201	-0.523931	-0.161206	-0.055228	0.588127

Nota: i valori rappresentano i coefficienti delle feature nelle componenti principali.

4.6 Clustering con K-Means

Il clustering tramite l’algoritmo **K-Means** è stato applicato al dataset per segmentare i giocatori in gruppi coerenti basati su caratteristiche specifiche. Questa sezione descrive i passaggi seguiti, le metriche calcolate e le visualizzazioni create per interpretare i risultati.

4.6.1 Calcolo del WCSS e del Silhouette Score

Per determinare il numero ottimale di cluster, sono state calcolate due metriche fondamentali:

- **WCSS (Within-Cluster Sum of Squares):** misura la compattezza dei cluster, ovvero quanto vicini sono i punti al centroide del proprio cluster. Un valore più basso indica cluster più compatti.
- **Silhouette Score:** misura la separazione tra i cluster, calcolando la differenza tra la coesione interna e la distanza tra cluster. Un valore più alto indica cluster ben separati.

La Figura 4.12 mostra il grafico del **metodo del gomito** per il WCSS e il Silhouette Score per ciascun valore di k (numero di cluster).

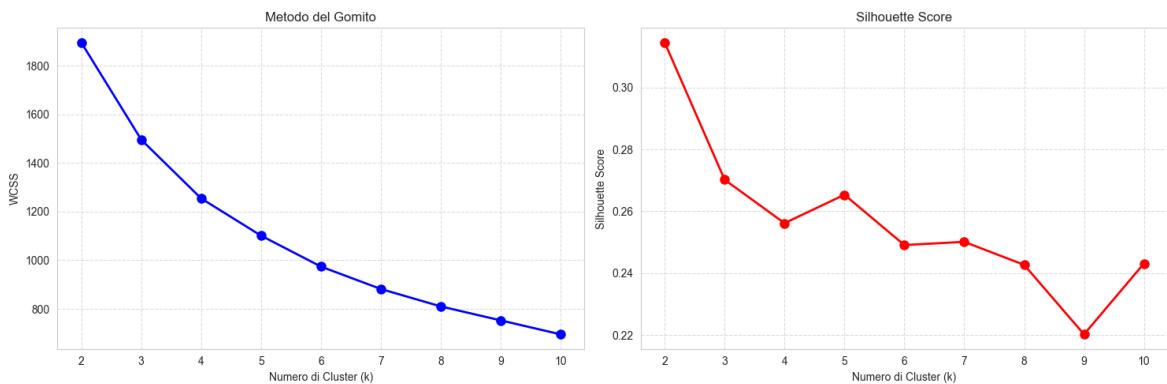


Figura 4.12: Grafici del WCSS (a sinistra) e del Silhouette Score (a destra) in funzione di k .

4.6.2 Analisi del metodo del gomito e del silhouette score

Analizziamo più specificatamente i risultati appena presentati attraverso la Figura 4.13.

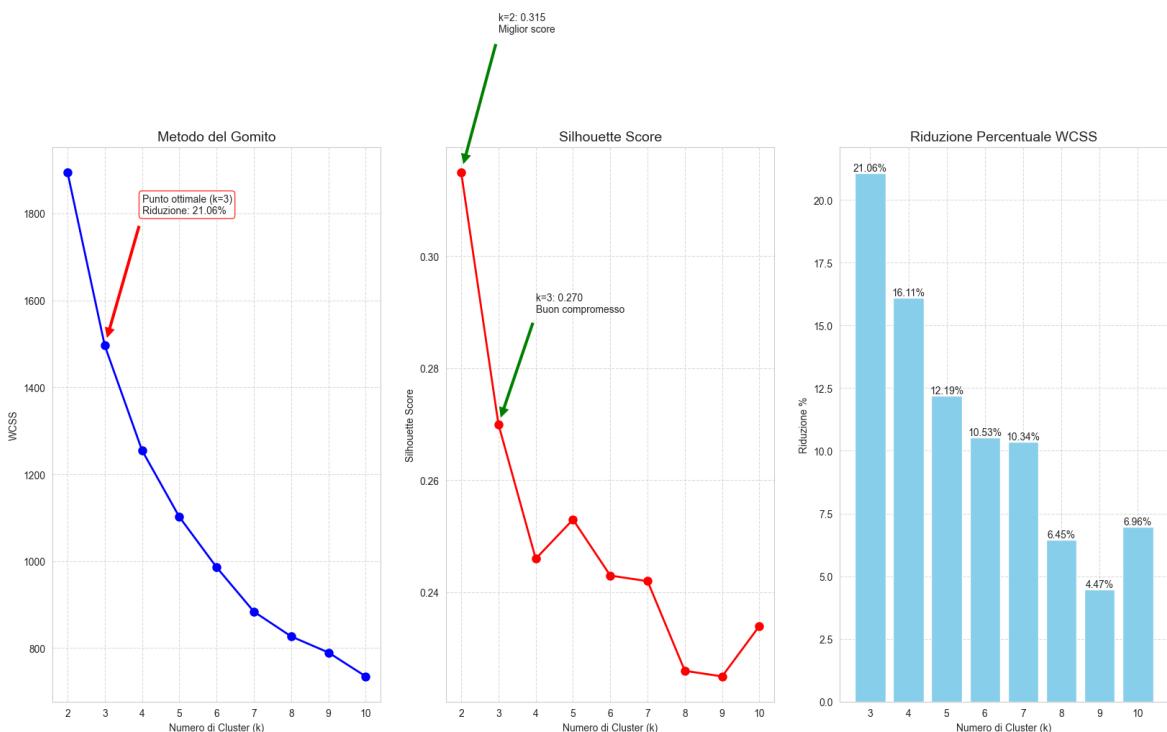


Figura 4.13: Specifica del grafico del WCSS (a sinistra), del silhouette score (al centro) e della riduzione percentuale del WCSS (a destra) in funzione del numero di cluster.

Dai grafici è possibile osservare che:

- Il metodo del gomito suggerisce $k = 3$ come punto ottimale, poiché si osserva una significativa riduzione del WCSS tra $k = 2$ e $k = 3$, seguita da una diminuzione meno marcata per valori di k superiori.
- Il silhouette score è massimo per $k = 2$, indicando una separazione netta tra i cluster. Tuttavia, per $k = 3$, si osserva un buon compromesso tra coesione e separazione.

La scelta ricade, dunque, su **3 cluster**. Tale scelta viene motivata anche dai grafici mostrati nelle Figure 4.14 e 4.15.

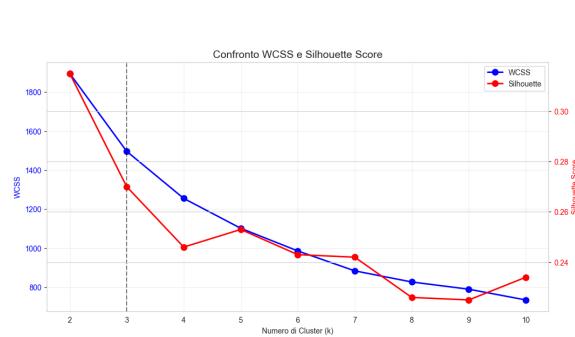


Figura 4.14: Confronto doppio tra andamento del parametro WCSS e del silhouette score.

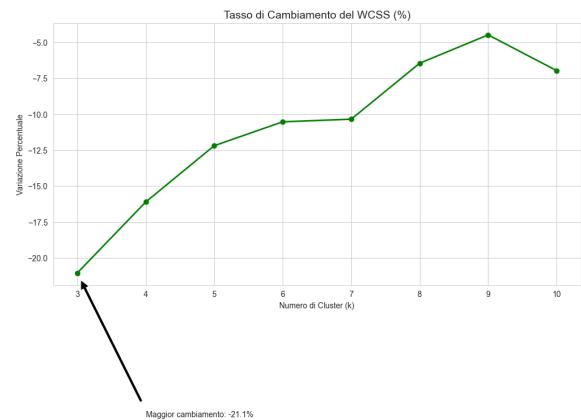


Figura 4.15: Tasso di variazione percentuale del WCSS.

4.6.3 Visualizzazione dei cluster

Visualizzazione 2D

I cluster rilevati sono stati inizialmente visualizzati in uno scatter plot bidimensionale, come mostrato in Figura 4.16. I punti rappresentano i giocatori, mentre i colori indicano i cluster assegnati. I centroidi dei cluster sono evidenziati con una croce nera.

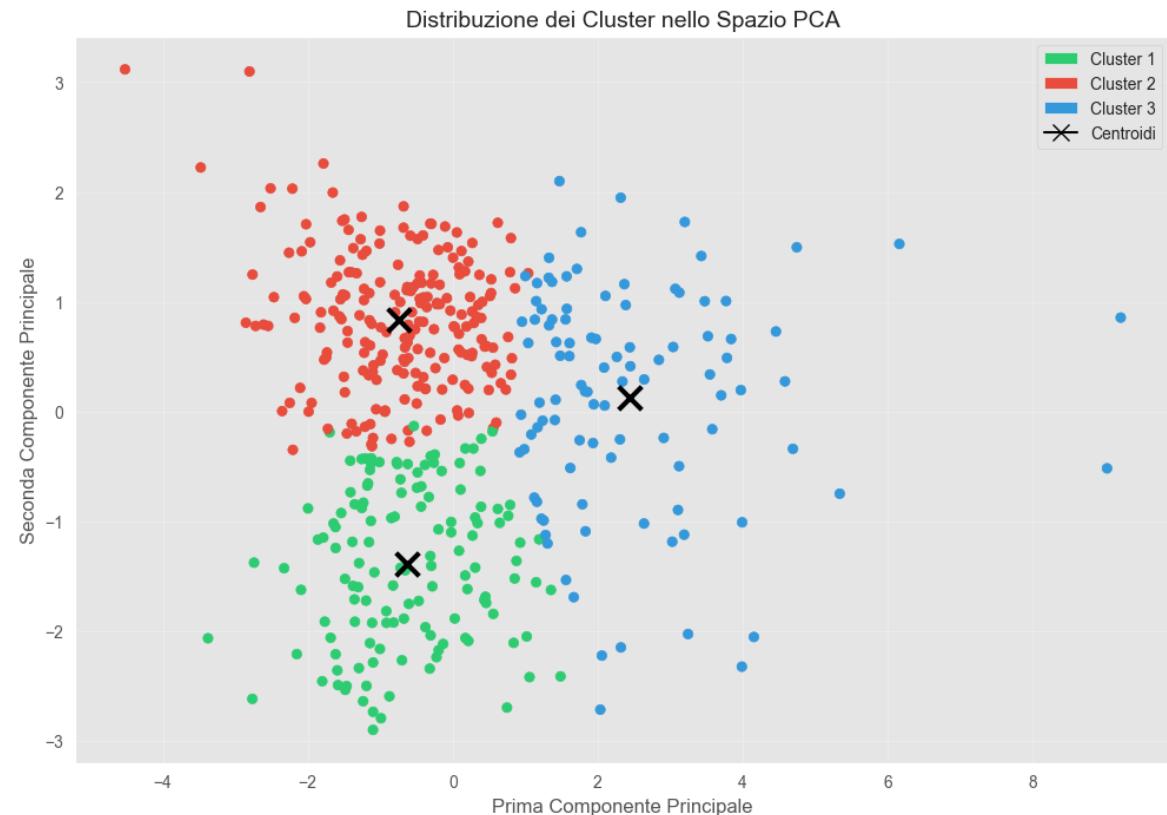


Figura 4.16: Distribuzione dei cluster nello spazio PCA bidimensionale.

Visualizzazione 3D

Per fornire una rappresentazione più dettagliata, i cluster sono stati visualizzati in uno spazio tridimensionale utilizzando le tre componenti principali. La Figura 4.17 mostra questa distribuzione.

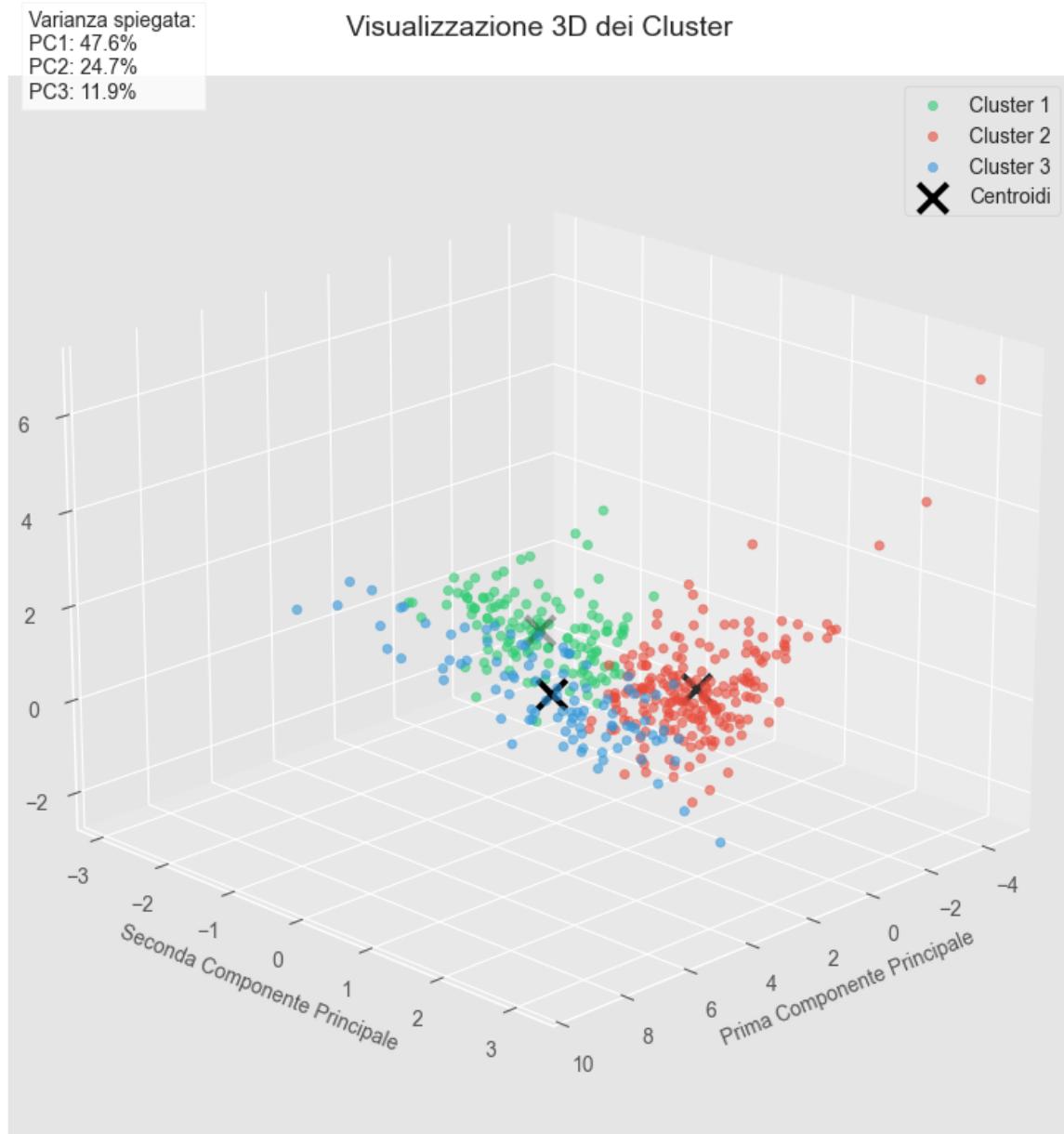


Figura 4.17: Distribuzione tridimensionale dei cluster nello spazio PCA.

4.6.4 Caratteristiche dei cluster

Le caratteristiche principali dei cluster sono state sintetizzate in alcuni diagrammi. Anzitutto, vengono mostrate le dimensioni dei tre cluster nella Figura 4.18.

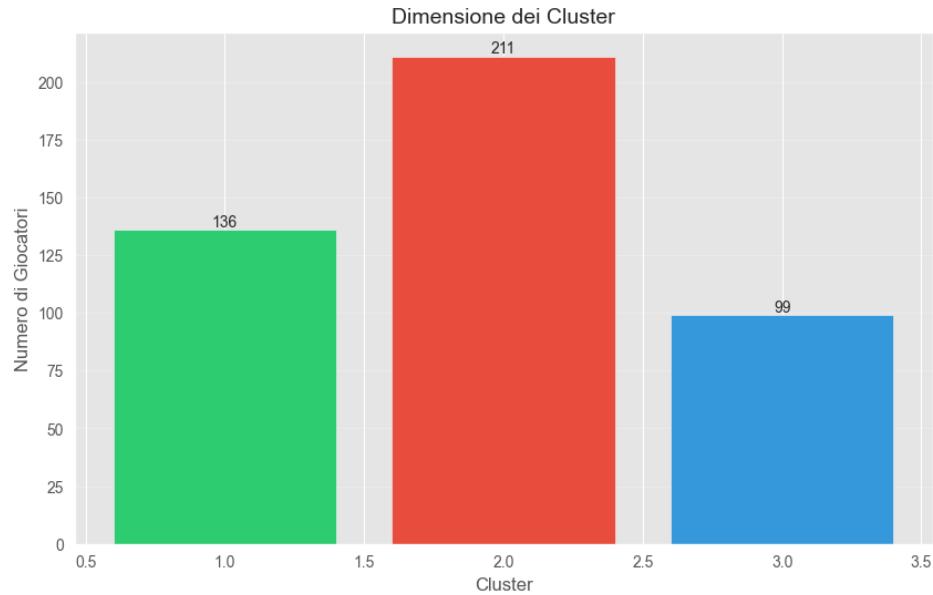


Figura 4.18: Distribuzione dei campioni nei tre cluster.

Inoltre, le caratteristiche relative alle sei feature selezionate a seguito della scrematura iniziale sono state analizzate attraverso una heatmap (Figura 4.23), che evidenzia i valori normalizzati per ciascuna feature in relazione ai diversi cluster. Per completare l’analisi, è stato realizzato un grafico a radar (Figura 4.20) che illustra in modo chiaro e visivo l’importanza relativa di ciascuna feature nel determinare i cluster, fornendo una rappresentazione intuitiva della distribuzione e della predominanza delle caratteristiche nei vari gruppi.

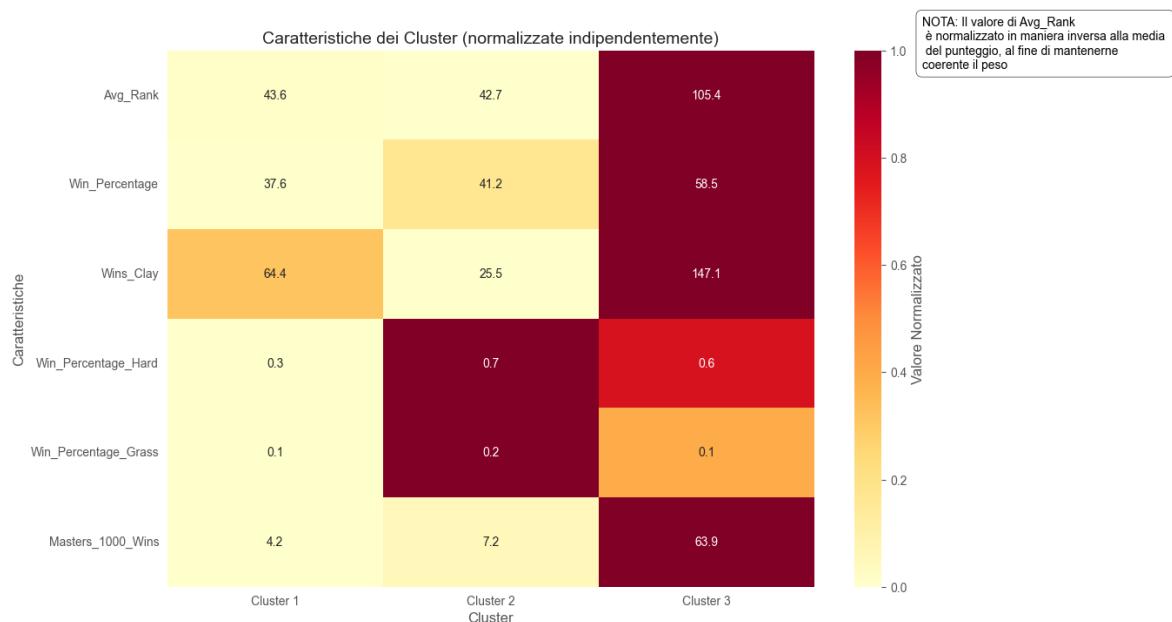


Figura 4.19: Caratteristiche principali dei cluster normalizzate - Heatmap.

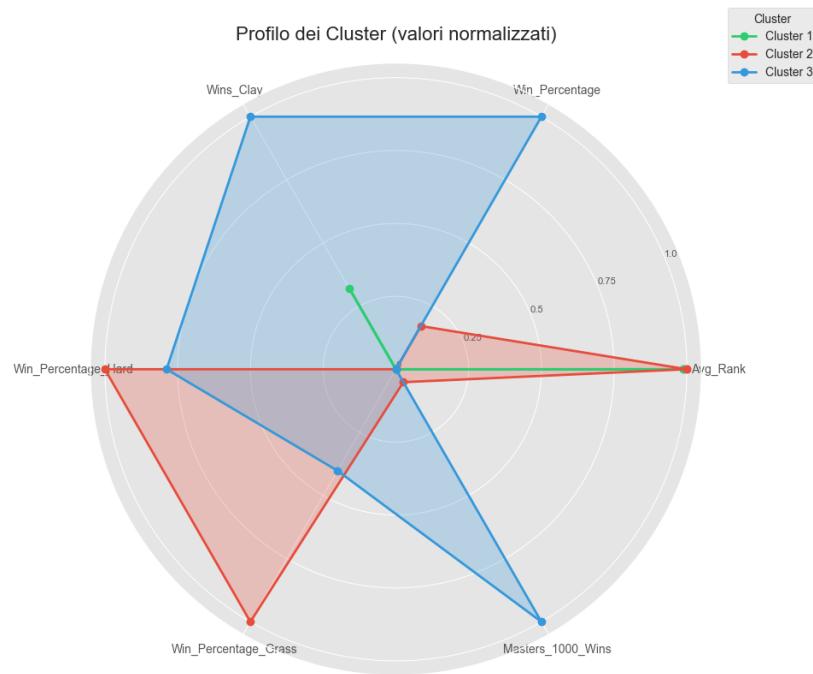


Figura 4.20: Caratteristiche principali dei cluster normalizzate - Radar.

4.6.5 Distribuzione dei silhouette score per cluster

Per concludere l'analisi dell'algoritmo di clustering KMeans, la Figura 4.21 mostra la distribuzione dei silhouette score per ciascun cluster. La linea verticale tratteggiata rappresenta il silhouette score medio, il cui valore è pari a 0.2702.

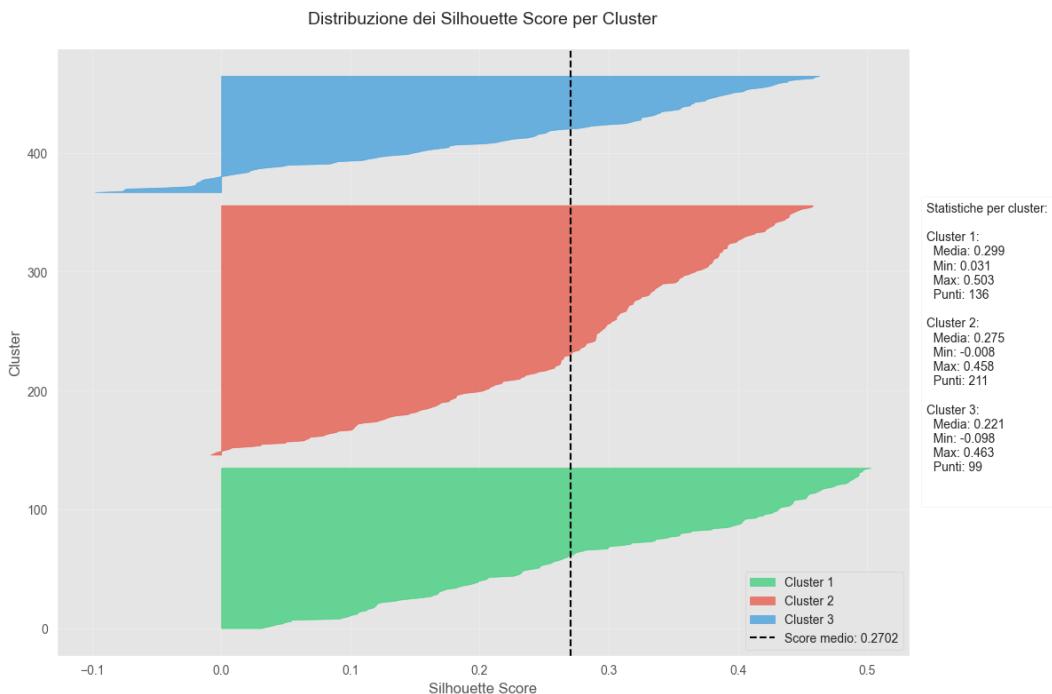


Figura 4.21: Distribuzione dei silhouette score per cluster.

4.7 Clustering con DBSCAN

In questa sezione viene illustrato l'approccio al clustering mediante **DBSCAN** (*Density-Based Spatial Clustering of Applications with Noise*), un algoritmo non supervisionato basato sulla densità dei dati. Questo metodo consente di individuare cluster di forma arbitraria, identificando e gestendo efficacemente i punti rumorosi. Di seguito sono descritti i principali passaggi e risultati ottenuti applicando DBSCAN al dataset considerato.

4.7.1 Ottimizzazione dei parametri di DBSCAN

Per determinare i valori ottimali dei parametri `eps` (raggio di ricerca locale) e `min_samples` (numero minimo di punti per definire un cluster come tale), è stata condotta una ricerca su griglia (*grid search*). Per ogni combinazione di `eps` e `min_samples`, sono stati calcolati il numero di cluster generati, il numero di punti rumorosi e lo *silhouette score*, una metrica che misura la coesione e separazione dei cluster.

La Tabella 4.3 mostra un esempio delle combinazioni calcolate e dei rispettivi risultati.

Tabella 4.3: Esempio di combinazioni di parametri DBSCAN e risultati

eps	min_samples	Numero Cluster	Punti Rumorosi	Silhouette Score	Note
0.55	6	4	389	0.331	-
0.55	7	3	404	0.334	-
0.55	8	1	-	-	Escluso (1 solo cluster)
...

Sono state generate due heatmap (Figura 4.22 e Figura 4.23) per rappresentare visivamente i risultati ottenuti dalla grid search. La prima evidenzia lo *silhouette score* in funzione dei parametri, mentre la seconda mostra il numero di cluster ottenuti. Si è esplorato uno spazio di ricerca che, per il valore di `eps`, passa da 0.1 a 3.0, con intervalli di 0.05, e per il valore di `min_samples` spazia da 3 a 15.

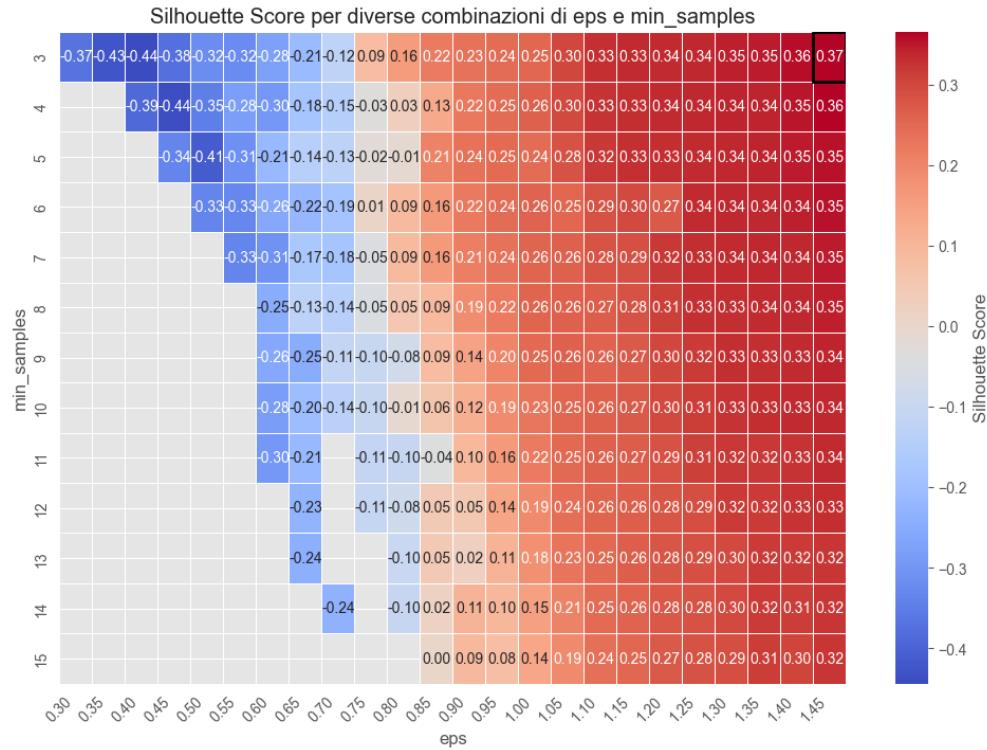


Figura 4.22: Heatmap dello *Silhouette Score* per diverse combinazioni di *eps* e *min_samples*. Le caselle contornate in nero indicano i parametri ottimali.

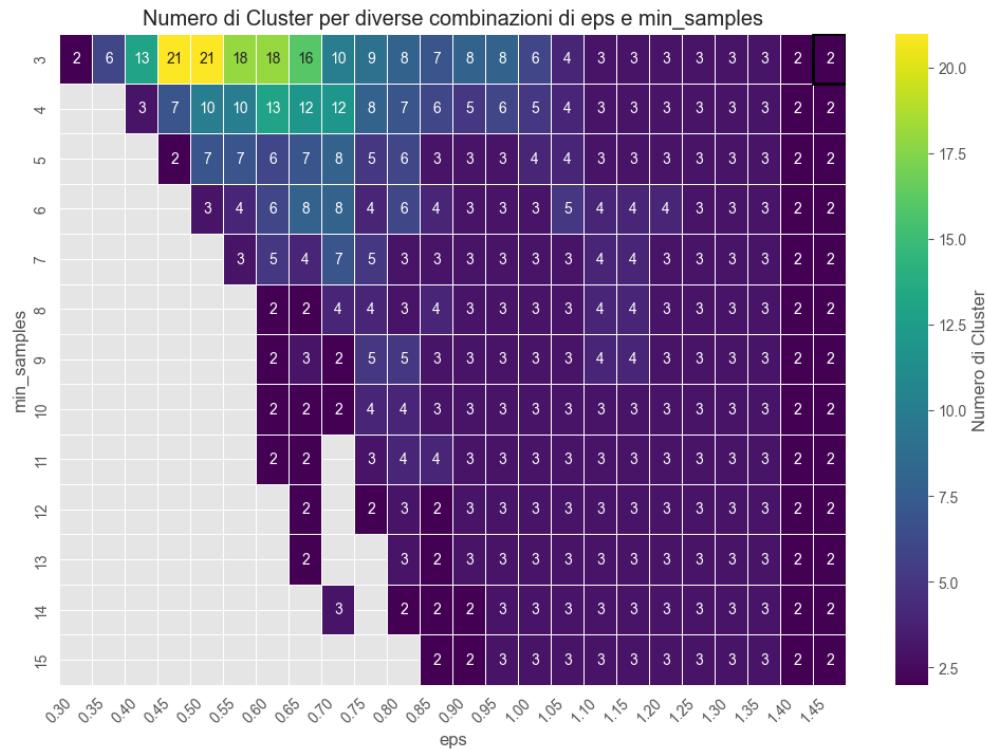


Figura 4.23: Heatmap del numero di cluster generati per diverse combinazioni di *eps* e *min_samples*. Le caselle contornate in nero indicano i parametri ottimali.

4.7.2 Visualizzazione dei cluster identificati

Una volta selezionati i parametri ottimali ($\text{eps}=0.1\cdot 1.45$ e $\text{min_samples}=3$), l'algoritmo DBSCAN è stato applicato al dataset normalizzato. I risultati del clustering sono stati visualizzati sia in due che in tre dimensioni, utilizzando la *Principal Component Analysis* (PCA) per ridurre le dimensioni del dataset.

Visualizzazione 2D

La Figura 4.24 mostra i cluster rilevati in due dimensioni. I punti rumorosi (indicati come "Rumore") sono rappresentati con una croce azzurra, mentre i cluster sono distinti da colori diversi.

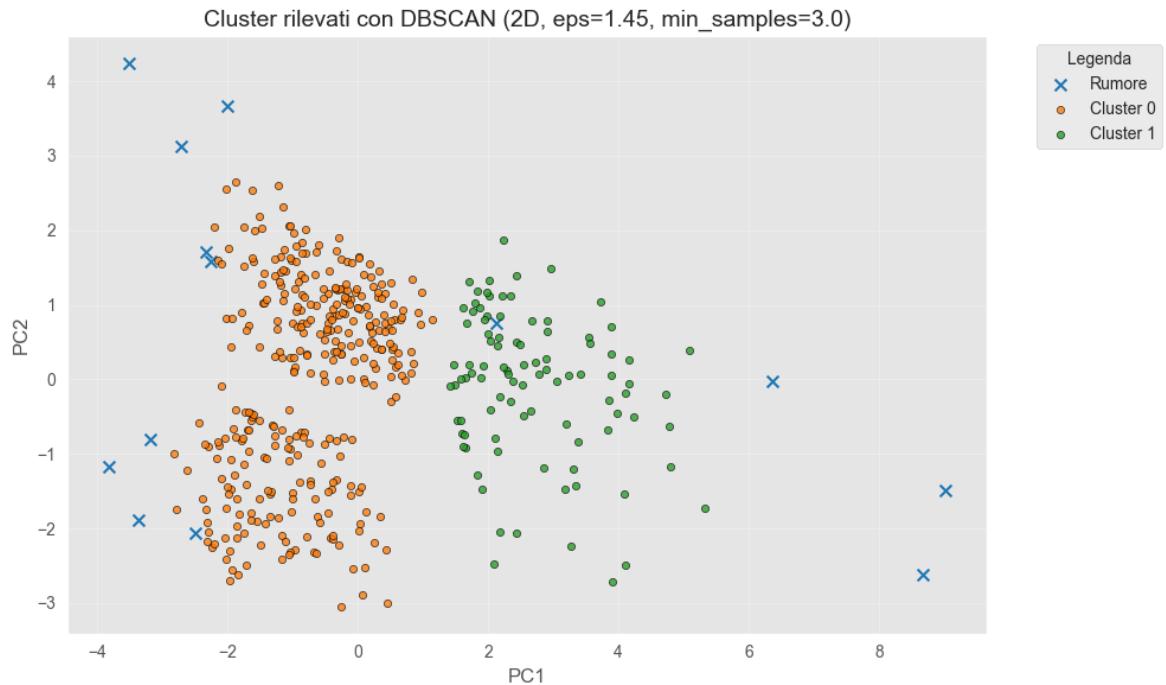


Figura 4.24: Cluster identificati da DBSCAN visualizzati in 2D).

Visualizzazione 3D

La Figura 4.25 presenta i cluster nello spazio tridimensionale, anch'esso ottenuto tramite PCA. Ogni cluster, anche qui, è evidenziato da un colore diverso, con una rappresentazione più evidente dei punti rumorosi.

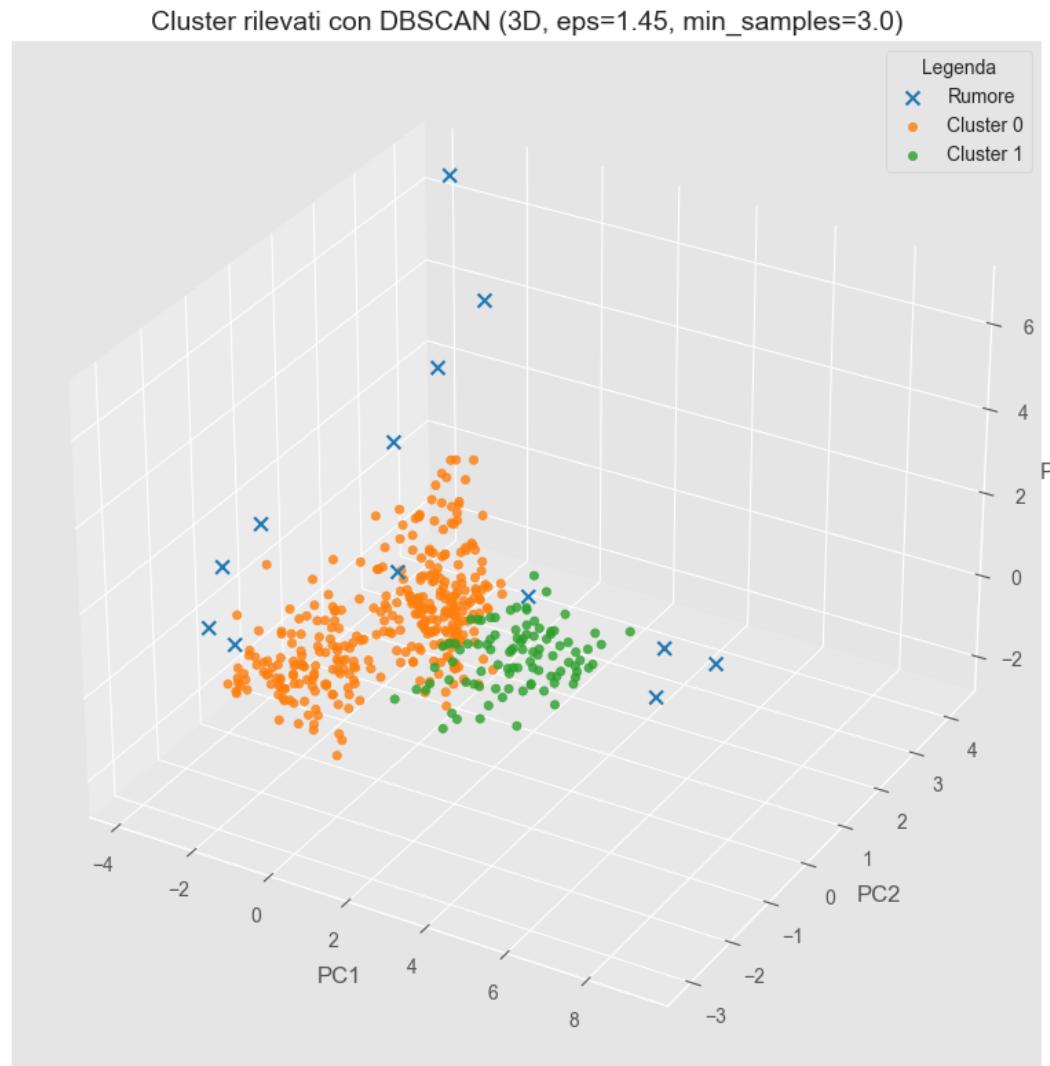


Figura 4.25: Cluster identificati da DBSCAN visualizzati in 3D.

4.8 Clustering con HDBSCAN

In questa sezione viene applicato il metodo di clustering **HDBSCAN** (*Hierarchical Density-Based Spatial Clustering of Applications with Noise*) e vengono confrontati i risultati con quelli ottenuti utilizzando **DBSCAN**. HDBSCAN, un'estensione di DBSCAN, permette di gestire in modo dinamico la densità dei cluster, adattandosi meglio a dataset eterogenei. Inoltre, HDBSCAN identifica cluster con densità variabile e gestisce in modo efficace i punti rumorosi.

4.8.1 Applicazione di HDBSCAN

L'algoritmo HDBSCAN è stato configurato con i seguenti parametri:

- `min_cluster_size = 10`: Numero minimo di punti per formare un cluster.
- `min_samples = 5`: Numero minimo di punti per considerare un punto come core.
- `metric = euclidean`: Distanza utilizzata per il clustering.

L'output di HDBSCAN è un array di etichette, dove:

- I cluster vengono numerati a partire da 0.
- Il rumore è etichettato con -1.
- Per migliorare la leggibilità, le etichette dei cluster sono state incrementate di 1, mantenendo invariato il valore -1 per il rumore.

4.8.2 Visualizzazione dei risultati

I risultati sono stati visualizzati utilizzando la *Principal Component Analysis* (PCA) per ridurre le dimensioni a 2D e 3D.

Visualizzazione 2D

La Figura 4.26 mostra i cluster identificati in uno spazio bidimensionale. I punti rumorosi (etichettati come "Rumore") sono rappresentati in nero con una croce, mentre i cluster validi sono distinti da colori diversi.

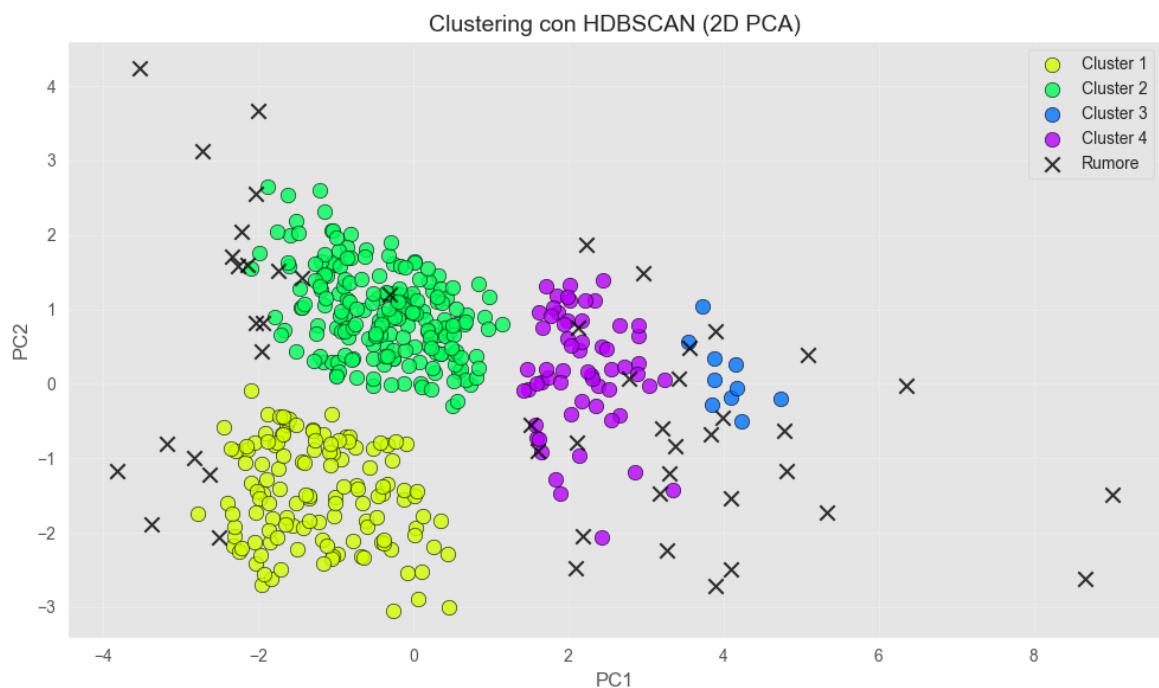


Figura 4.26: Clustering con HDBSCAN visualizzato in 2D.

Visualizzazione 3D

La Figura 4.27 rappresenta i cluster nello spazio tridimensionale. Ogni cluster è evidenziato da un colore distinto, mentre i punti rumorosi sono rappresentati in nero.

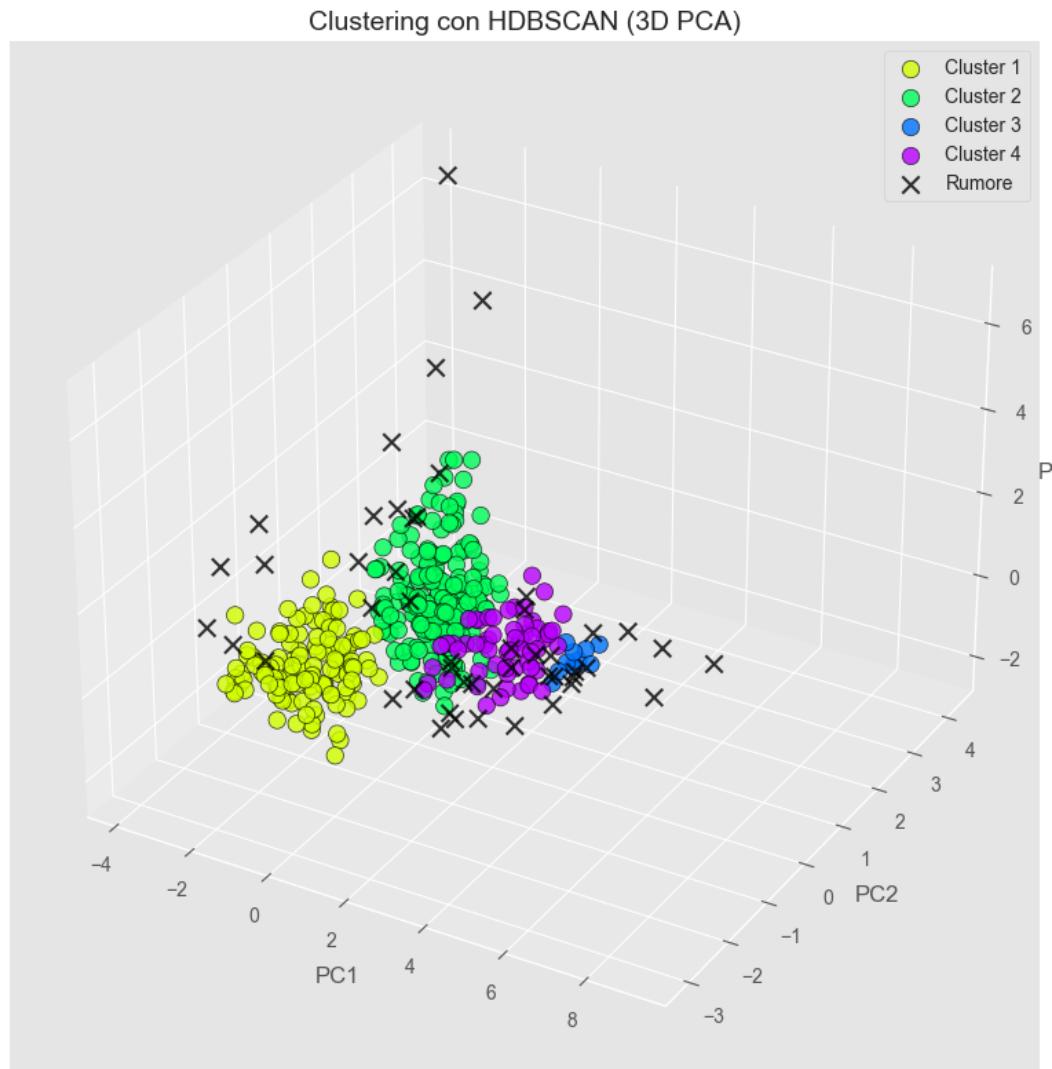


Figura 4.27: Clustering con HDBSCAN visualizzato in 3D.

4.8.3 Confronto tra HDBSCAN e DBSCAN

Il confronto tra HDBSCAN e DBSCAN ha evidenziato differenze significative nelle prestazioni dei due algoritmi:

- **Identificazione dei cluster:** HDBSCAN è risultato più efficace nel rilevare cluster di densità variabile, mentre DBSCAN richiede una densità uniforme dei dati.
- **Configurazione dei parametri:** HDBSCAN richiede meno tuning dei parametri rispetto a DBSCAN, poiché calcola automaticamente le soglie di densità.
- **Qualità dei cluster:** HDBSCAN ha prodotto cluster più coerenti nei dati analizzati, come confermato dall'analisi visiva.

CAPITOLO 5

FORECASTING

5.1 Introduzione

In questo capitolo ci concentreremo sullo studio delle serie temporali, un'area fondamentale dell'analisi dei dati che consente di osservare e modellare fenomeni che evolvono nel tempo. L'obiettivo principale è quello di sfruttare i dati a disposizione per effettuare una previsione accurata sull'andamento di alcune variabili chiave.

Per questa analisi, abbiamo utilizzato un dataset contenente i dati delle azioni Apple, che copre un periodo significativo dal 1984 al 2024. La lunghezza e la qualità del dataset ci hanno permesso di sviluppare modelli predittivi robusti, in grado di fornire previsioni affidabili sia sul prezzo di chiusura delle azioni che sul volume degli scambi per i periodi successivi.

5.1.1 Motivazioni della scelta del dataset

L'utilizzo del dataset relativo alle azioni Apple per il forecasting e le serie temporali è preferibile rispetto ai risultati delle partite di tennis ATP per diversi motivi:

- **Continuità temporale:** i dati delle azioni sono registrati con regolarità (es. giornaliera), mentre i risultati ATP sono distribuiti in modo irregolare nel tempo.
- **Volume sufficiente di dati:** le azioni Apple offrono migliaia di osservazioni su lunghi periodi (1984-2024), mentre i risultati ATP sono più limitati e frammentati.
- **Variabili significative:** il dataset Apple contiene variabili numeriche utili per il forecasting (es. prezzo, volume, variazione), mentre i dati ATP sono principalmente categorici.
- **Trend e stagionalità:** le serie temporali finanziarie presentano trend chiari e stagionalità, ideali per il forecasting, mentre i dati ATP non mostrano pattern temporali strutturati.
- **Adattabilità ai modelli di serie temporali:** i dati finanziari sono un caso d'uso classico per modelli come ARIMA e SARIMAX, mentre i risultati ATP sono più adatti per classificazione o clustering.

5.2 Analisi preliminare

Prima di procedere con l'applicazione dei modelli di forecasting, è fondamentale osservare il comportamento dei dati nel tempo. A tal fine, abbiamo realizzato una serie di grafici per visualizzare l'andamento di tre metriche fondamentali: il prezzo delle azioni, il cambioamento percentuale giornaliero e il volume di scambio. Queste visualizzazioni forniscono una panoramica iniziale e aiutano a identificare eventuali trend, stagionalità o anomalie nei dati.

5.2.1 Andamento del prezzo nel tempo

Il grafico (Figura 5.1) mostra l'evoluzione del prezzo di chiusura delle azioni Apple (**Price**) dal 1984 al 2024. Questo consente di evidenziare eventuali trend di lungo termine, come periodi di crescita o crisi, e di comprendere le dinamiche storiche del mercato.

I grafici seguenti mostrano l'andamento delle principali metriche di prezzo: **chiusura**, **apertura**, **massimo giornaliero** e **minimo giornaliero**, dal 1984 al 2024. Questi grafici forniscono una visione completa delle dinamiche del mercato, consentendo di identificare trend, stagionalità e anomalie. Come è evidente, i grafici sono pressoché identici ed evidenziano una crescita significativa a lungo termine, con accelerazioni notevoli in corrispondenza di eventi di mercato chiave (a fine 2024 Apple risulta essere l'azienda con la capitalizzazione di mercato più alta al mondo).

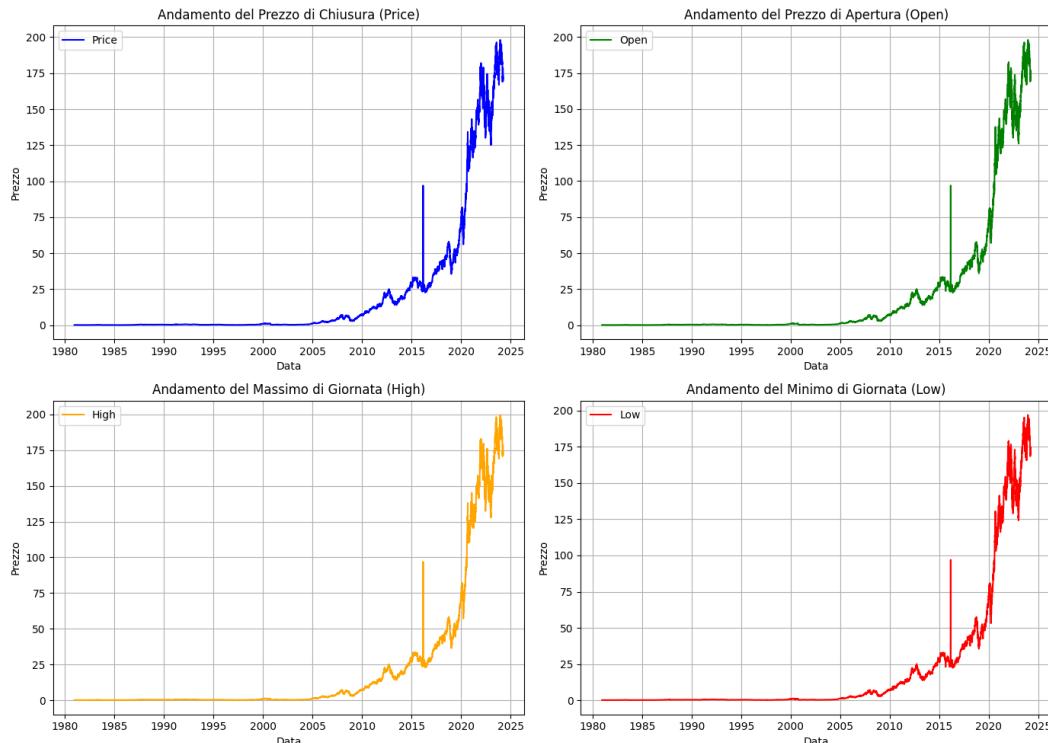


Figura 5.1: Andamento dei prezzi delle azioni Apple (1984-2024).

Durante l'analisi preliminare dei dati, è stato individuato un picco insolito relativo alla data 27 febbraio 2016, con i seguenti valori anomali:

02/27/2016, 96.95, 96.95, 96.95, 96.95, , 300.12%

Questa osservazione presenta diverse incongruenze che suggeriscono un errore nei dati:

- Le altre fonti non riportano questo valore anomalo per la stessa data.

- Il 27 febbraio 2016 era un sabato, giorno in cui il mercato azionario non opera, escludendo quindi la possibilità che si tratti di un dato valido.

In base a queste evidenze, abbiamo deciso di procedere con l'eliminazione di questa osservazione dal dataset, garantendo così l'integrità dei dati e la correttezza delle analisi successive (Figura 5.2).

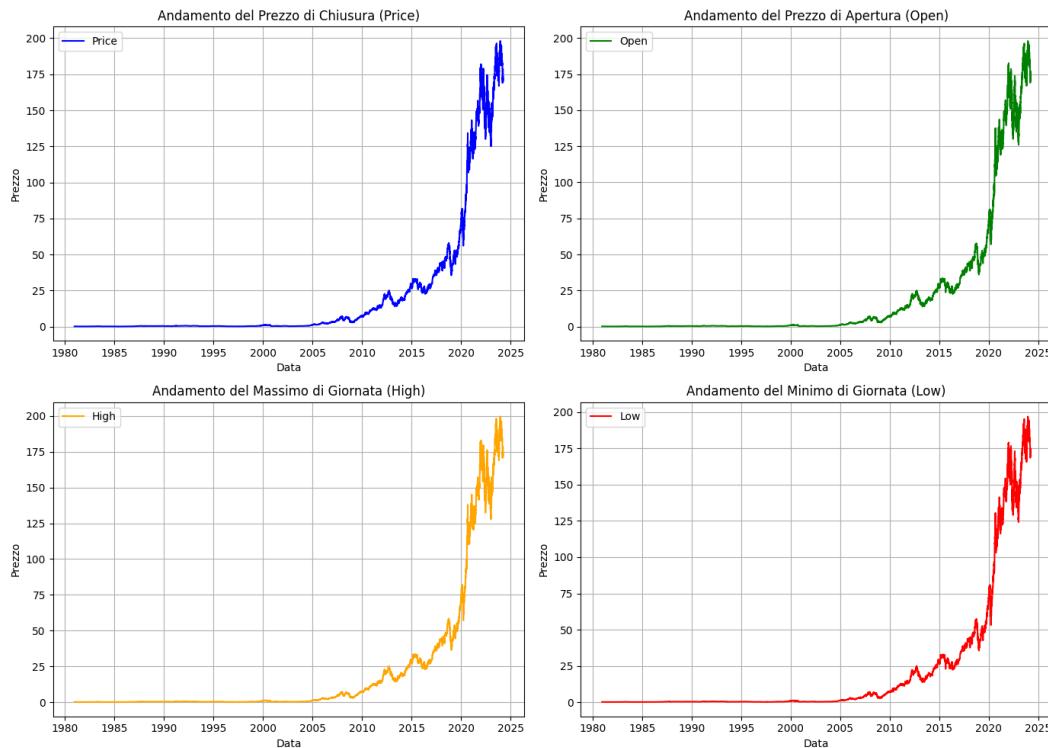


Figura 5.2: Andamento corretto dei prezzi delle azioni Apple: prezzo di Chiusura, apertura, massimo e minimo Giornaliero (1984-2024).

5.2.2 Andamento del cambio giornaliero

Il grafico in Figura 5.3 rappresenta l'andamento del cambiamento percentuale giornaliero (Change %) delle azioni Apple dal 1984 al 2024. Questa metrica misura la variazione percentuale del prezzo di chiusura rispetto al giorno precedente e offre una visione chiara della volatilità del mercato in diversi periodi storici. Questa analisi preliminare è fondamentale per identificare eventuali anomalie che potrebbero influire sulla qualità del forecasting e per comprendere il contesto storico della serie temporale.

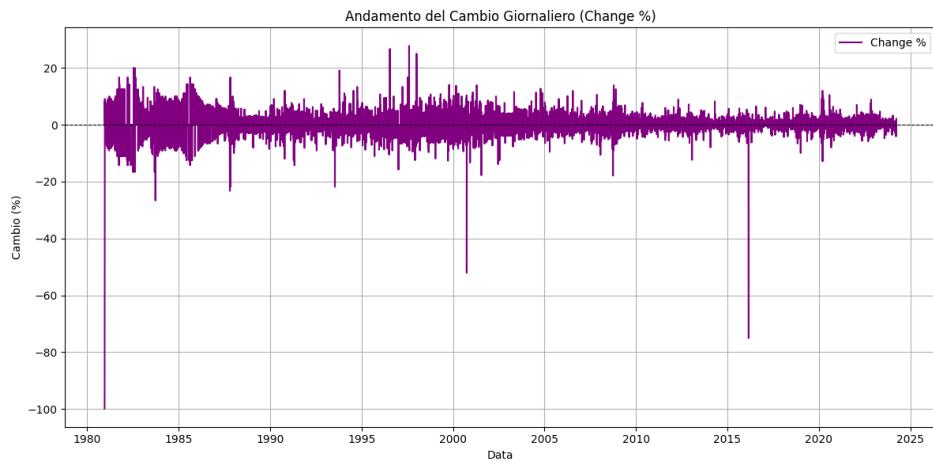


Figura 5.3: Andamento della volatilità (1984-2024).

Analizzando il grafico, possiamo notare alcuni picchi negativi molto pronunciati, come il forte calo del 1987 e del 2015. Durante l’analisi del cambio giornaliero, sono emersi valori estremamente bassi, inferiori a -30% . Tali valori rappresenterebbero situazioni di grave volatilità nel mercato o, più probabilmente, errori nei dati. Dopo un’attenta analisi, abbiamo individuato tre dati relativi ai giorni 12 dicembre 1980, 29 settembre 2000 e 29 febbraio 2016 che non trovano riscontro nella documentazione storica né in altre fonti di report finanziari. Pertanto, li consideriamo errori e li rimuoviamo dal dataset. Il risultato è espresso in Figura 5.4.

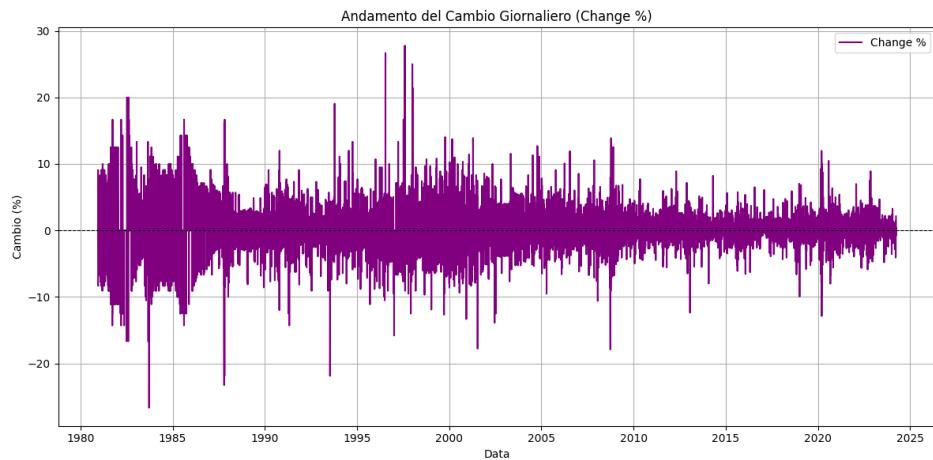


Figura 5.4: Andamento corretto della volatilità (1984-2024).

5.2.3 Andamento del volume nel tempo

Il grafico (Figura 5.5) rappresenta l’andamento del volume di scambi (**Vol.**) delle azioni Apple nel tempo. Questa metrica è fondamentale per analizzare l’attività del mercato e identificare eventuali pattern o anomalie. Analizzando il grafico, emergono alcune osservazioni significative come una crescita generale del volume di scambi aumenta progressivamente nel tempo.

Dopo questa trasformazione, il dataset risultante conserva tutte le informazioni fondamentali per il forecasting, ma in una forma più compatta e maneggevole.

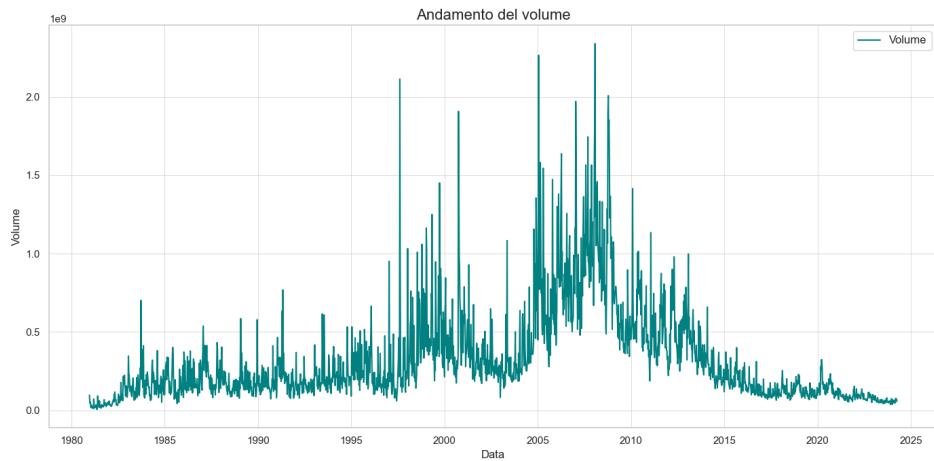


Figura 5.5: Andamento del volume (Vol.) delle Azioni Apple nel tempo (1984-2024).

5.3 Forecasting sul prezzo di chiusura con il modello ARIMA

L’analisi è stata condotta impiegando la libreria `Statsmodels`, concentrandosi inizialmente sui prezzi di chiusura.

Il dataset originario delle azioni Apple copre un periodo esteso, dal 1984 al 2024, e contiene osservazioni giornaliere: questa granularità offre dettagli significativi, ma un volume di dati così ampio può influire negativamente sulle prestazioni dei modelli predittivi, sia in termini di tempi di calcolo che di complessità, senza garantire un incremento proporzionale dell’accuratezza delle previsioni. Per ovviare a tali criticità, si è deciso di ridurre il dataset pur mantenendo l’informazione essenziale al forecasting. In particolare, è stato effettuato un campionamento bisettimanale, ovvero condensando ogni due settimane (10 giorni di dati) in un singolo dato rappresentativo, utilizzando la media. L’implementazione di tale operazione è resa semplice dal metodo `resample()` di Pandas:

```
weekly_avg = df.resample('2W').mean()
```

A seguito di questa riduzione, è stata eseguita una decomposizione della serie temporale, mirata a evidenziare l’eventuale stagionalità, ossia quelle variazioni ricorrenti che si ripresentano, con analoga intensità, negli stessi periodi di anno in anno.

L’analisi (visualizzabile in Figura 5.6) ha evidenziato così le seguenti componenti:

- **Prezzo di chiusura (Price):** Il grafico mostra l’andamento storico del prezzo di chiusura delle azioni Apple dal 1984 al 2024 (già ampiamente descritto nelle sezioni precedenti).
- **Componente di trend:** La componente di trend, estratta dalla serie temporale, indica un chiaro aumento dei prezzi nel tempo. Questo suggerisce una crescita sostenuta e costante delle azioni Apple negli ultimi anni.
- **Componente stagionale:** La componente stagionale mostra variazioni periodiche ricorrenti con una stagionalità regolare su base annuale. Queste oscillazioni sono indicative di fluttuazioni stagionali nei prezzi delle azioni.
- **Componente residuale:** Il residuo rappresenta le variazioni non spiegate dal trend e dalla stagionalità. I residui sono relativamente stabili per la maggior parte del periodo analizzato, ma mostrano un aumento della variabilità negli ultimi anni, suggerendo una maggiore volatilità dovuta all’instabile situazione politica ed economica nel post-Covid.

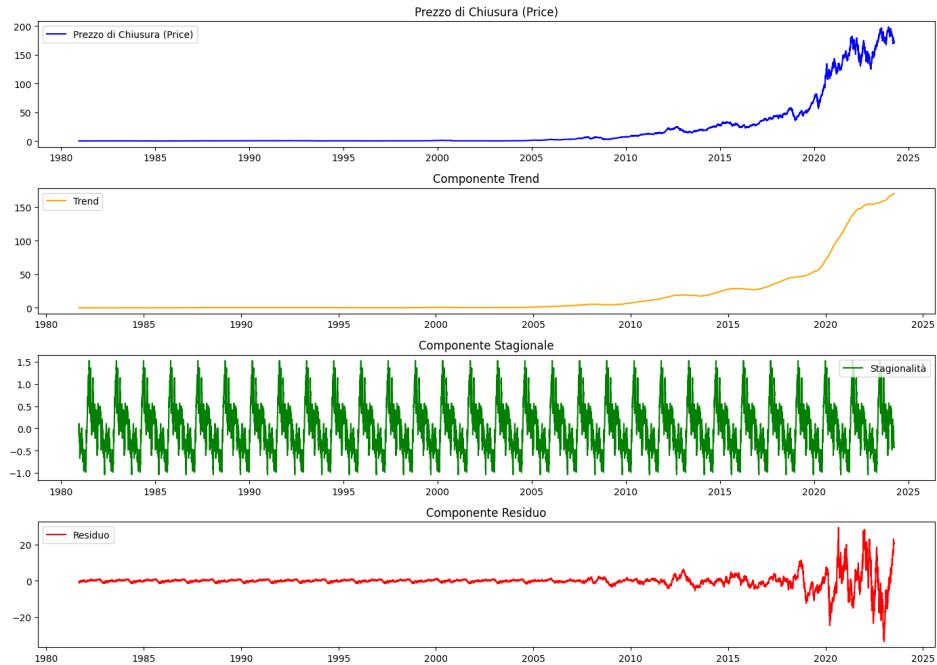


Figura 5.6: Risultati dell’analisi delle componenti della serie temporale delle azioni Apple.

Per raggiungere gli obiettivi prefissati, è stato adottato il modello ARIMA in modo prevedere i prezzi di chiusura delle azioni.

Per l’applicazione del modello ARIMA, è fondamentale verificare la stazionarietà della serie temporale, poiché questo tipo di modello presuppone che i dati siano stazionari. Il test utilizzato per verificare questa proprietà è l’**Augmented Dickey-Fuller (ADF) Test**. L’ADF Test è un test statistico utilizzato per determinare se una serie temporale è stazionaria. Se la serie non è stazionaria, potrebbe essere necessario differenziarla prima di applicare un modello ARIMA.

Il test ADF restituisce diversi valori, ma i più rilevanti per determinare la stazionarietà sono:

1. **Statistic Value:** Il valore della statistica del test.
2. **P-value:** Se il *p-value* è inferiore a una soglia (ad esempio, 0.05), possiamo rifiutare l’ipotesi nulla (H_0), che afferma che la serie ha una radice unitaria (non è stazionaria). In tal caso, possiamo dire che la serie è stazionaria.
3. **Critical Values:** I valori critici del test per vari livelli di significatività (1%, 5%, 10%).

Alla luce di ciò, i risultati del test applicato alla colonna Price sono i seguenti:

Metrica	Valore
ADF Statistic Value	2.39
P-value	0.999
Critic Value (1%)	-3.431
Critic Value (5%)	-2.862
Critic Value (10%)	-2.567

Dal test ADF risulta che il *p-value* è pari a 0.999, un valore di gran lunga superiore alla soglia comune di 0.05. Pertanto, non possiamo rifiutare l’ipotesi nulla (H_0), che afferma che

la serie ha una radice unitaria ed è quindi non stazionaria. Di conseguenza, è necessario applicare la differenziazione alla serie finché non si ottengono dati stazionari. Di seguito vengono riportati i risultati del test in seguito alla prima differenziazione:

Metrica	Valore
ADF Statistic Value	-18.22
P-value	2.38 e-30
Critic Value (1%)	-3.431
Critic Value (5%)	-2.862
Critic Value (10%)	-2.567

Dopo aver differenziato la serie, la statistica del test è diventata molto più negativa, e il p-value estremamente basso conferma che la serie è ora stazionaria. Questi valori indicano che è possibile applicare correttamente un modello ARIMA alla serie differenziata.

In Figura 5.7 è possibile osservare la differenza tra i dati di partenza non stazionari e quelli stazionari, ottenuti dopo la differenziazione.

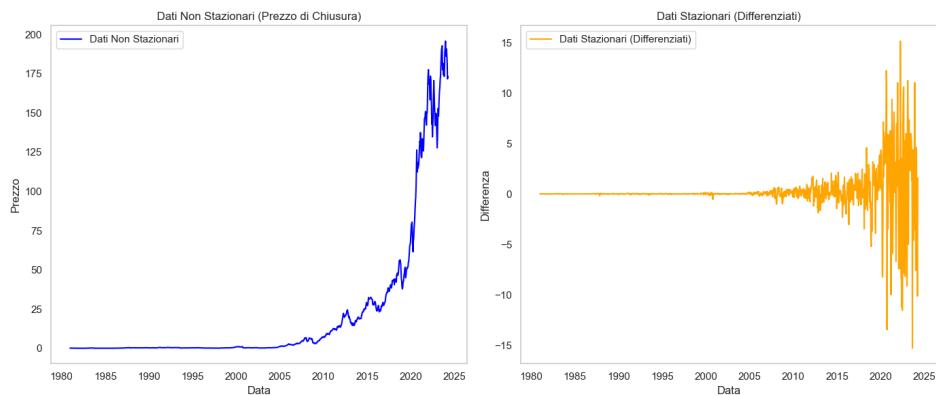


Figura 5.7: Confronto tra i dati prima e dopo la differenziazione.

Ottenuti dei dati stazionari, possiamo procedere con la scelta dei parametri per il nostro modello. È possibile procedere con la determinazione dei parametri del modello ARIMA. Questi parametri sono tre: p , d e q . In Figura 5.8 sono riportati i grafici che mostrano il comportamento delle funzioni di autocorrelazione (ACF) e autocorrelazione parziale (PACF) per i dati stazionari.

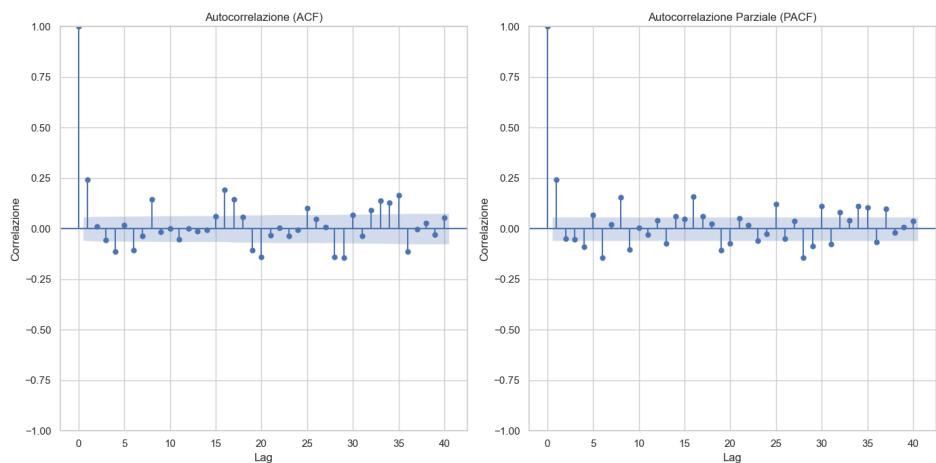


Figura 5.8: Autocorrelazione e autocorrelazione parziale.

Per determinare l'ordine ottimale del nostro modello, abbiamo utilizzato il metodo `auto_arima()`, che seleziona automaticamente la configurazione più adatta in base ai dati in input, minimizzando l'Akaike Information Criterion (AIC). L'AIC è una metrica utilizzata per confrontare modelli statistici, valutandone la qualità della stima considerando sia la bontà di adattamento sia la complessità del modello.

Nel nostro caso, il modello con il valore di AIC più basso corrisponde all'ordine:

$$(p, d, q) = (0, 2, 1)$$

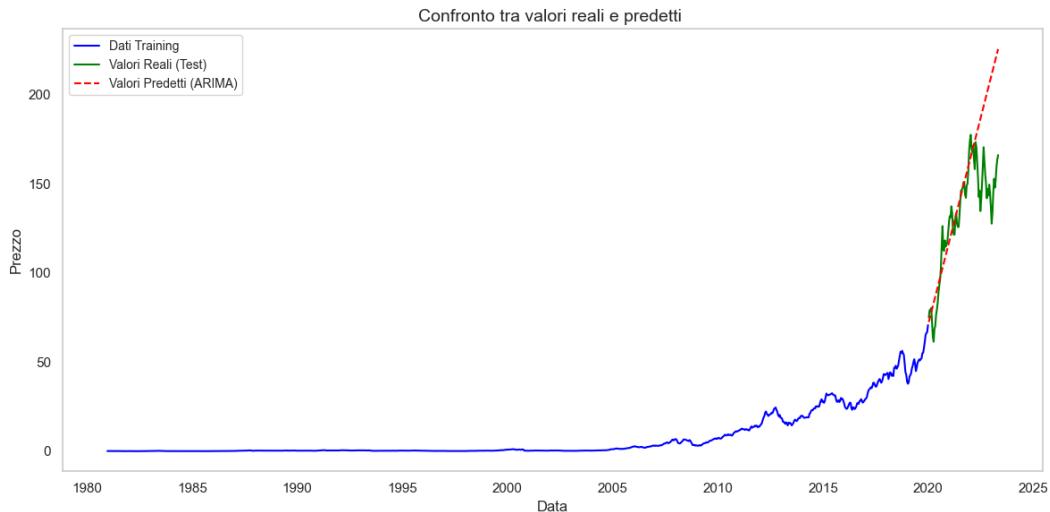
Questi parametri sono stati utilizzati per costruire il modello di previsione finale.

A questo punto, prima di effettuare la predizione, i dati sono stati suddivisi in due insiemi: dati di *training* e dati di *test*. I dati di *training* includono le osservazioni fino al 31 dicembre 2019, mentre i dati di *test* coprono il periodo dal 1 gennaio 2020 al 27 marzo 2024 (ultimo giorno coperto dal dataset). L'obiettivo di questa divisione è stato quello di utilizzare i dati di *training* per addestrare il modello e prevedere i valori futuri fino al 2024, confrontandoli successivamente con i dati di *test* per valutare l'accuratezza della predizione. A questo punto, sui dati di *training*, siamo andati ad applicare il modello per la predizione dei dati. In Figura 5.9 sono osservabili le specifiche del modello di predizione utilizzato.

Dep. Variable:	Price	No. Observations:	9845			
Model:	ARIMA(0, 2, 1)	Log Likelihood	-223.217			
Date:	Mon, 27 Jan 2025	AIC	450.435			
Time:	15:06:56	BIC	464.824			
Sample:	0	HQIC	455.309			
	- 9845					
Covariance Type:	opg					
	coef	std err	z			
ma.L1	-0.9981	0.000	-2303.089			
sigma2	0.0612	0.000	315.047			
			P> z			
			[0.025 0.975]			
ma.L1	-0.9981	0.000	0.000	-0.999	-0.997	
sigma2	0.0612	0.000	315.047	0.000	0.061	0.062
Ljung-Box (L1) (Q):	0.97		Jarque-Bera (JB):	618330.57		
Prob(Q):	0.33		Prob(JB):	0.00		
Heteroskedasticity (H):	2098.99		Skew:	-0.86		
Prob(H) (two-sided):	0.00		Kurtosis:	41.79		

Figura 5.9: Specifiche del modello ARIMA.

Come è possibile constatare in Figura 5.10, il modello ARIMA è efficace nel rappresentare il trend storico lineare e crescente, tuttavia, negli ultimi anni, le predizioni non si allineano perfettamente con i dati reali. Ciò è dovuto alle fluttuazioni improvvise e ad eventi economici straordinari, come la pandemia da Covid-19 e gli sviluppi economici correnti. La pandemia ha infatti causato un'elevata volatilità nei mercati globali e l'incremento dell'inflazione negli Stati Uniti ha rallentato il mercato azionario. Questi fattori combinati hanno complicato la capacità del modello di cogliere le dinamiche complesse e imprevedibili delle azioni (soprattutto di quelle tecnologiche, come Apple).

**Figura 5.10:** Confronto tra valori reali e predetti

Nella tabella sottostante (Tabella 5.1) sono stati inseriti i valori dei prezzi di chiusura predetti per le prime 10 settimane, confrontandoli con quelli reali. Ovviamente, la tabella rappresenta solo una piccola porzione dell'insieme completo delle predizioni, ma serve a fornire una prima impressione generale.

Data	Valori Reali	Valori Predetti
2020-01-12	75.120000	72.543843
2020-01-26	79.075556	74.324352
2020-02-09	79.464000	76.104861
2020-02-23	80.394444	77.885371
2020-03-08	72.466000	79.665880
2020-03-22	64.223000	81.446389
2020-04-05	61.482000	83.226898
2020-04-19	68.616667	85.007408
2020-05-03	70.293000	86.787917
2020-05-17	76.414000	88.568426

Tabella 5.1: Confronto tra valori reali e predetti per le prime 10 settimane

Calcolando il *Mean Squared Error* (MSE) e il *Root Mean Squared Error* (RMSE), emergono valori piuttosto elevati, che evidenziano la scarsa accuratezza del modello nelle previsioni degli ultimi anni:

- **MSE:** 1012,48
- **RMSE:** 31,82

Modificando i parametri del modello *ARIMA*, si osservano variazioni nei risultati, ma l'accuratezza delle previsioni non migliora.

A questo punto, per verificare l'impatto degli anni più recenti caratterizzati da alta volatilità economica (specie nel settore tech), è stato effettuato un test rimuovendo dal dataset i dati relativi agli ultimi due anni. Di conseguenza, il dataset è stato limitato fino al 31 dicembre 2022, partendo dall'ipotesi che questi anni fossero i principali responsabili dell'inaccuratezza del modello. Come evidenziato nel grafico sottostante (Figura 5.11), la

previsione risulta essere infatti molto più accurata rispetto ai grafici precedenti. È stata inoltre aggiunta un’area rosa che rappresenta l’intervallo di confidenza al 95% delle predizioni del modello ARIMA. L’eliminazione dei dati degli ultimi anni ha contribuito a migliorare la precisione delle previsioni, riducendo l’impatto delle fluttuazioni estreme. Ciò viene dimostrato anche dai nuovi valori di MSE e RMSE che sono scesi drasticamente:

- **MSE:** 89,22
- **RMSE:** 9,45

Tuttavia, l’area di confidenza riflette comunque una crescente incertezza con l’aumentare dell’orizzonte temporale, ma il fatto che molti dei valori reali rientrino in tale intervallo conferma una discreta affidabilità del modello nel catturare il trend generale.

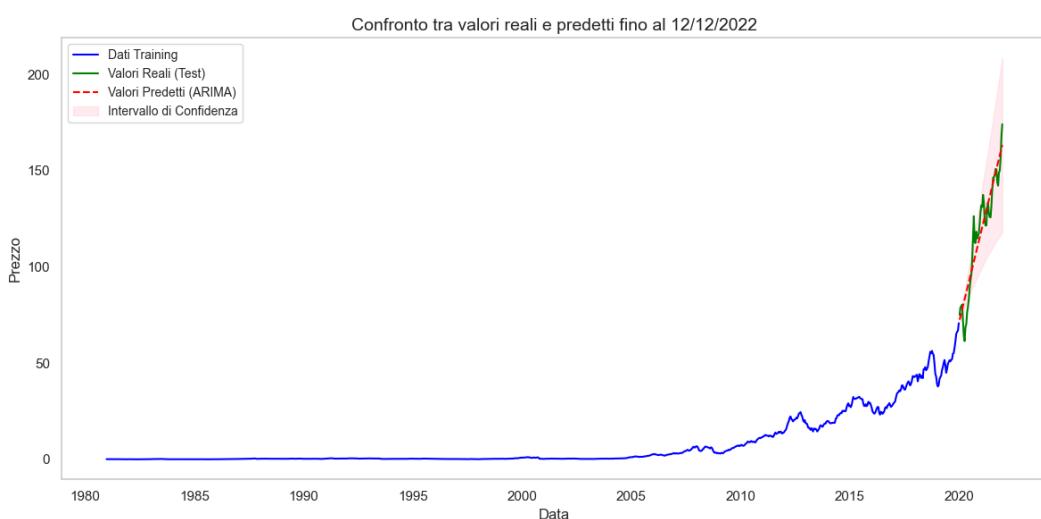


Figura 5.11: Confronto tra valori reali e predetti fino al 12/12/2022.

In conclusione, l’analisi condotta evidenzia i limiti intrinseci dei modelli predittivi applicati al mercato azionario. Nonostante l’uso di tecniche avanzate come l’ARIMA, il mercato azionario si dimostra fortemente influenzato da fattori imprevedibili, quali eventi geopolitici, crisi economiche e decisioni politiche. La volatilità e l’incertezza rendono estremamente complesso, se non impossibile, prevedere con precisione l’andamento futuro dei prezzi.

5.4 Forecasting sull’andamento del volume con il modello SARIMAX

In questa sezione verrà sviluppato un modello di forecasting basato su SARIMAX per stimare l’evoluzione futura del volume (**Vol.**) delle azioni Apple.

La prima fase prevede il preprocessing dei dati. I valori relativi al volume, infatti, sono inizialmente espressi in forma testuale e accompagnati dal suffisso M per indicare i milioni e B per indicare i miliardi. Per poter operare numericamente su questi dati, è stata implementata una funzione che li converte in valori reali, moltiplicando opportunamente per 10^6 o 10^9 a seconda del caso.

Successivamente, analogamente a quanto realizzato in precedenza per il forecasting dei prezzi di chiusura, si è proceduto ad applicare un processo di resampling. Mentre in quella fase era stato scelto un resampling settimanale, in questa analisi si è optato per

un’aggregazione su base bimestrale, ottenuta calcolando la media dei volumi. Tale scelta consente di attenuare le forti oscillazioni giornaliere e di evidenziare le tendenze di lungo periodo.

```
df_monthly = df['Vol.'].resample('2M').mean()
```

Questa scelta è dettata dalla necessità di attenuare le forti oscillazioni presenti nei dati: il resampling mensile consente di ottenere una rappresentazione più *smooth* del trend sottostante, semplificando l’applicazione delle tecniche di forecasting. Di seguito è riportato il grafico ottenuto (Figura 5.12).

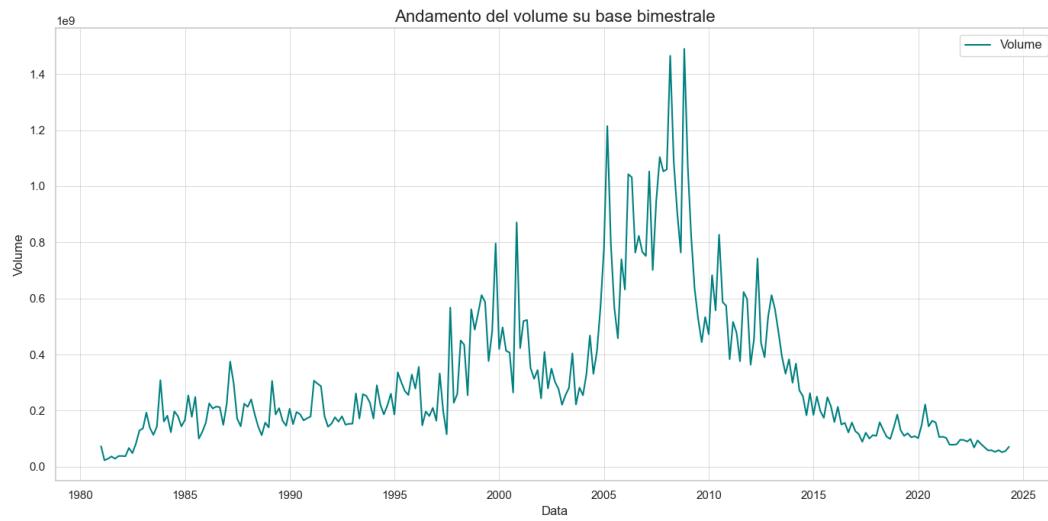


Figura 5.12: Andamento del volume delle azioni Apple con aggregazione bimestrale.

A questo punto si passa alla suddivisione del dataset. Il dataset mensile è stato diviso in due insiemi: un training set, contenente la maggior parte dei dati storici (dal 1984 al 2022), e un test set, costituito dagli ultimi 24 mesi (dal 2022 al 2024), che permette di confrontare le previsioni del modello con i dati reali.

Il modello SARIMAX, in grado di gestire le componenti autoregressive, l’integrazione e la stagionalità tipica dei dati finanziari, è stato scelto per questa analisi. I parametri iniziali, impostati a `order=(1, 1, 1)`, rappresentano una configurazione di partenza che potrà essere ulteriormente ottimizzata. Il risultato finale ottenuto può essere visualizzato in Figura 5.13.

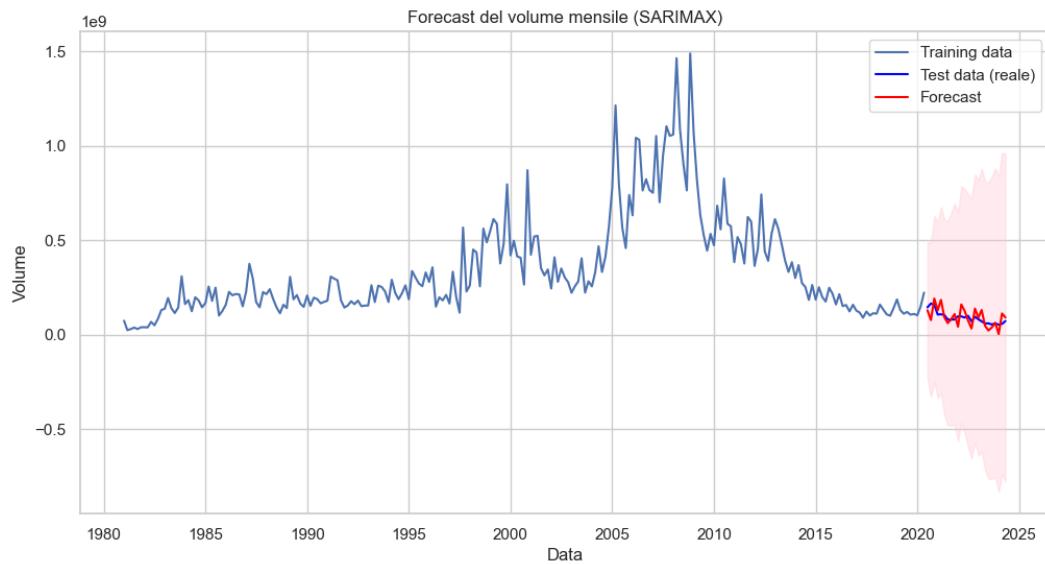


Figura 5.13: Forecasting del volume sugli ultimi 2 anni con modello SARIMAX.

Il modello ottenuto sembra riuscire in modo parziale a catturare sia il trend generale che la stagionalità. Le metriche calcolate sono le seguenti:

- MSE = 374.661.829.900.922
- RMSE = 19.356.183

Sebbene questi valori possano apparire elevati in termini assoluti, essi devono essere interpretati in relazione alla scala dei dati, poiché il volume medio è pari a circa 320 milioni di azioni. In particolare, un RMSE di circa 19,36 milioni corrisponde a un errore relativo intorno al 6% rispetto al valore medio del volume. In questo contesto, l’errore medio è discreto e le performance del modello possono essere ritenute accettabili, pur lasciando margine di miglioramento in termini di precisione.