Tesina di:

Data Science

Analisi su dataset per Seire temporali, classificazione e clustering

Gabriel Piercecchi

Tosca Pierro

Università Politecnica delle Marche

Facoltà di Ingegneria

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

---

Tesina di:

Data Science

Esame per il corso tenuto dal Prof. Domenico Ursino,
durante l'anno accademico 2024-2025

9 CFU

Professori:
Domenico Ursino
Michele Marchetti

Redazione del documento a cura di:

- **Piercecchi Gabriel** (Matr. 1120541) - s1120541@studenti.univpm.it
- **Pierro Tosca** (Matr. 1120542) - s1120542@studenti.univpm.it

---

Anno Accademico 2024-2025

Anno II

Indice

1. Introduzione

In questa tesina sono stati analizzati due dataset provenienti da ambiti molto diversi, al fine di applicare metodi statistici e di clustering per l'estrazione di informazioni significative.

Il primo dataset è dedicato ai Pokémon di *Pokémon GO*. Qui l'attenzione è rivolta all'analisi delle statistiche di base dei Pokémon, con lo scopo di determinare quali siano più efficaci in situazioni di PVE (Player versus Environment) e PVP (Player versus Player). Utilizzando tecniche di clustering, si è cercato di raggruppare i Pokémon in base alle loro caratteristiche, evidenziando i possibili cluster che potrebbero indicare performance simili in contesti competitivi.

Il secondo dataset, invece, riguarda i *Crimini* registrati a Los Angeles nel corso degli anni compresi tra il 2020 e il 2024. L'obiettivo principale è stato studiare l'andamento temporale e la distribuzione geografica degli eventi criminali. Attraverso l'analisi dei dati, si sono voluti individuare eventuali trend e correlazioni, esplorando come la frequenza e la localizzazione dei crimini si siano evolute nel tempo.

Questi risultati potranno fornire spunti utili per l'elaborazione di strategie di intervento e prevenzione in ambito di sicurezza urbana.

Riassumendo, nei capitoli successivi verranno esplicati in maniera dettagliati la metodologia impiegata, le analisi condotte e i risultati ottenuti, aprendo la strada a possibili sviluppi futuri e applicazioni pratiche.

Clustering e Classification

2. Clustering

In questo capitolo è stato analizzato il dataset relativo a Pokémon GO *(disponibile al seguente indirizzo* `https://www.kaggle.com/datasets/shreyasur965/pokemon-go/data` *) che raccoglie informazioni dettagliate su* 1007 *Pokémon. Il dataset include 24 attributi per ciascun Pokémon, tra cui statistiche di battaglia (quali* `base_attack`*,* `base_defense` *e* `base_stamina`*), tipi, rarità, metodi di acquisizione (wild, egg, raid, research, ecc.) e informazioni sulle mosse (fast e charged). Tali dati hanno rappresentato una risorsa preziosa per approfondire le potenzialità dei Pokémon in contesti di combattimento, sia in modalità PVE che PVP.*

2.1 Analisi del Dataset

La prima parte dell'analisi si è concentrata su una dettagliata esplorazione del dataset. Di seguito, per una migliore comprensione, vengono riportate le sue colonne con allegata descrizione:

- **pokemon_id**: Identificativo univoco per ciascun Pokémon.
- **pokemon_name**: Nome del Pokémon.
- **base_attack**: Statistica base dell'attacco del Pokémon.
- **base_defense**: Statistica base della difesa del Pokémon.
- **base_stamina**: Statistica base della resistenza (HP) del Pokémon.
- **type**: Tipo/i del Pokémon.
- **rarity**: Classificazione della rarità del Pokémon.
- **charged_moves**: Elenco delle mosse caricate disponibili per il Pokémon.
- **fast_moves**: Elenco delle mosse veloci disponibili per il Pokémon.
- **candy_required**: Numero di caramelle necessarie per l'evoluzione del Pokémon.
- **distance**: Distanza di camminata necessaria per ottenere caramelle con il Pokémon come compagno.
- **max_cp**: Potere di Combattimento massimo (CP) che il Pokémon può raggiungere.
- **attack_probability**: Probabilità che il Pokémon attacchi in battaglia.
- **base_capture_rate**: Probabilità base di catturare il Pokémon.
- **base_flee_rate**: Probabilità base che il Pokémon fugga dall'incontro.
- **dodge_probability**: Probabilità che il Pokémon schivi in battaglia.
- **max_pokemon_action_frequency**: Frequenza massima delle azioni del Pokémon in battaglia.
- **min_pokemon_action_frequency**: Frequenza minima delle azioni del Pokémon in battaglia.
- **found_egg**: Booleano che indica se il Pokémon può essere trovato nelle uova.
- **found_evolution**: Booleano che indica se il Pokémon può essere ottenuto tramite evoluzione.
- **found_wild**: Booleano che indica se il Pokémon può essere trovato nel mondo selvaggio.

- **found_research**: Booleano che indica se il Pokémon può essere ottenuto tramite attività di ricerca.
- **found_raid**: Booleano che indica se il Pokémon può essere incontrato nelle incursioni (raid).
- **found_photobomb**: Booleano che indica se il Pokémon può apparire nei photobomb (foto imprevisti).

2.1.1 ETL

Una volta scaricato il dataset in formato `.CSV` e caricato su Jupyter Notebook, sono state eseguite operazioni di pulizia e visualizzazione dei dati utilizzando le librerie *pandas*, *matplotlib* e *seaborn*.

In particolare, sono stati applicati dei vincoli di lettura da parte del framework per due colonne specifiche (**fast_moves** e **charged_moves**) al fine di prevenire eventuali problematiche future.

```
converters = {
    'fast_moves': str,
    'charged_moves': str
}
```

Successivamente, per verificare che il caricamento fosse avvenuto senza problemi, sono stati eseguiti i seguenti comandi:

```
file = 'pokemon.csv'
pokemon = pd.read_csv(file, converters=converters)
pokemon.head()
```

I quali hanno restituito come output le prime 5 istanze del dataset (Figura 2.1).

	pokemon_id	pokemon_name	base_attack	base_defense	base_stamina	type	rarity	charged_moves	fast_moves	candy_required	...	base_flee_rate	dodge_probab
0	1	Bulbasaur	118	111	128	['Grass', 'Poison']	Standard	['Sludge Bomb', 'Seed Bomb', 'Power Whip']	['Vine Whip', 'Tackle']	NaN	...	-1.0	
1	2	Ivysaur	151	143	155	['Grass', 'Poison']	Standard	['Sludge Bomb', 'Solar Beam', 'Power Whip']	['Razor Leaf', 'Vine Whip']	25.0	...	-1.0	
2	3	Venusaur	198	189	190	['Grass', 'Poison']	Standard	['Sludge Bomb', 'Petal Blizzard', 'Solar Beam']	['Razor Leaf', 'Vine Whip']	100.0	...	-1.0	
3	4	Charmander	116	93	118	['Fire']	Standard	['Flame Charge', 'Flame Burst', 'Flamethrower']	['Ember', 'Scratch']	NaN	...	-1.0	
4	5	Charmeleon	158	126	151	['Fire']	Standard	['Fire Punch', 'Flame Burst', 'Flamethrower']	['Ember', 'Fire Fang']	25.0	...	-1.0	

5 rows × 24 columns

Figura 2.1: Output che verifica il corretto carimento del file `pokemon.csv`

Modifca campi NaN

Confermato il corretto caricamento del dataset, si è proceduto alla gestione delle istanze contenenti campi *NaN*. In particolare, si è scelto di sostituire tali eventuali valori con la media dei valori adiacenti. Tale approccio risulta particolarmente utile quando i dati sono ordinati in maniera logica (ad esempio per ID o per una sequenza temporale) e permette di preservare

la coerenza dei dati, limitando l'effetto degli outlier che potrebbero influenzare una media globale.

Per verificare se, ed in quali colonne fossero presenti campi vuoti critici, è stato utilizzato il comando:

```
pokemon.isnull().sum()
```

Il conteggio è risultato il seguente:

```
pokemon_id                      0
pokemon_name                    0
base_attack                     0
base_defense                    0
base_stamina                    0
type                            0
rarity                          0
charged_moves                   0
fast_moves                      0
candy_required                536
distance                        0
max_cp                          0
attack_probability            103
base_capture_rate             103
base_flee_rate                103
dodge_probability             103
max_pokemon_action_frequency  103
min_pokemon_action_frequency  103
found_egg                     263
found_evolution               263
found_wild                    263
found_research                263
found_raid                    263
found_photobomb               263
dtype: int64
```

La scelta di imputare i campi NaN utilizzando la media dei valori adiacenti è stata motivata dalla semplicità d'implementazione e dalla leggerezza computazionale del metodo. Alternative come l'imputazione tramite media globale o mediana, seppur valide, non garantiscono lo stesso livello di coerenza locale e potrebbero essere maggiormente influenzate dalla presenza di outlier.

Verificata la presenza dei campi *NaN*, si è proceduto prima ad eliminare eventuali duplicati presenti nel dataset e poi a sostituire tali valori *NaN*.

Questo ha portato a una drastica diminuzione dei campi *NaN*, come mostrato nel conteggio seguente:

```
pokemon_id          0
pokemon_name        0
base_attack         0
base_defense        0
base_stamina        0
```

```
type                              0
rarity                            0
charged_moves                     0
fast_moves                        0
candy_required                    2
distance                          0
max_cp                            0
attack_probability                1
base_capture_rate                 1
base_flee_rate                    1
dodge_probability                 1
max_pokemon_action_frequency      1
min_pokemon_action_frequency      1
found_egg                         1
found_evolution                   1
found_wild                        1
found_research                    1
found_raid                        1
found_photobomb                   1
dtype: int64
```

2.1.2 Visualizzazione del Dataset

Fatto ciò, è stato possibile ottenere una prima visualizzazione quantitativa e qualitativa dei dati presenti all'interno del dataset: individuare i pokémon più forti. L'output è mostrato in Figura 2.2:

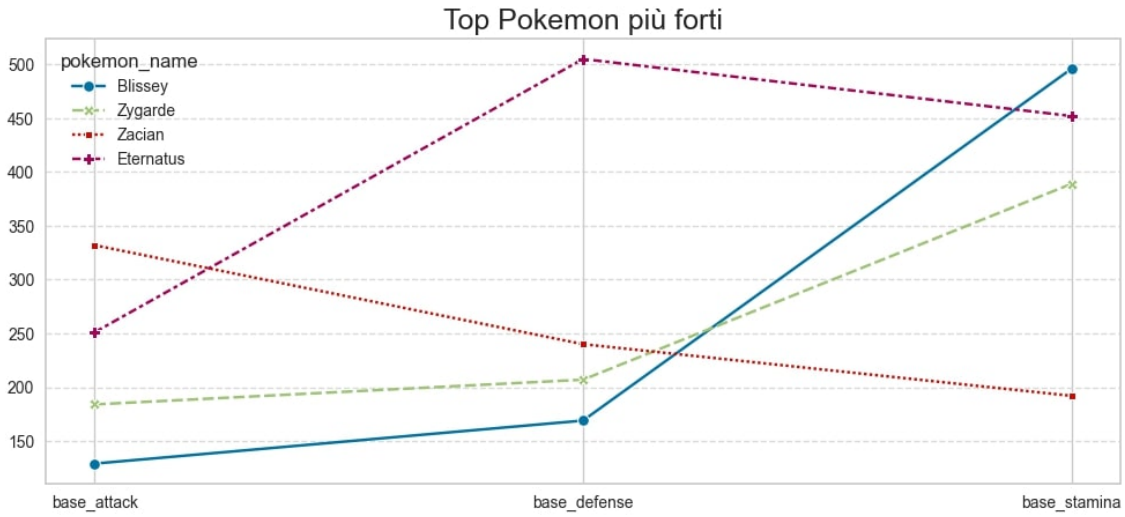


Figura 2.2: I pokemon più forti sono quindi *Blissey*, *Zygarde*, *Zacian* e *Eternatus*

Dall'analisi è emerso che, sebbene tutti e quattro i Pokémon presentino un attacco base "medio-basso", si distinguono significativamente per le altre due statistiche base. In particolare, nella difesa base, solo *Eternatus* mostra un livello "alto", mentre gli altri tre hanno un livello "basso". Per quanto riguarda la stamina base, invece, a parte *Zacian*, che ha un livello "basso", tutti gli altri Pokémon presentano un livello "alto".

Il fatto che l'attacco sia "medio-basso" per tutti suggerisce che l'efficacia in battaglia non dipenda solo dall'offensiva, ma anche dalla capacità di resistere agli attacchi e di sostenere scontri prolungati. Infatti, *Eternatus*, con la sua alta difesa, potrebbe essere impiegato come Pokémon da tank, mentre *Zacian*, con bassa stamina, potrebbe risultare più vulnerabile in combattimenti estesi e richiedere strategie basate su attacchi rapidi.

Top 10 Pokémon per attributo base

Continuando l'analisi, sono stati cercati i 10 pokémon più forti per ogni attributo base con il seguente output mostrato in Figura 2.3.

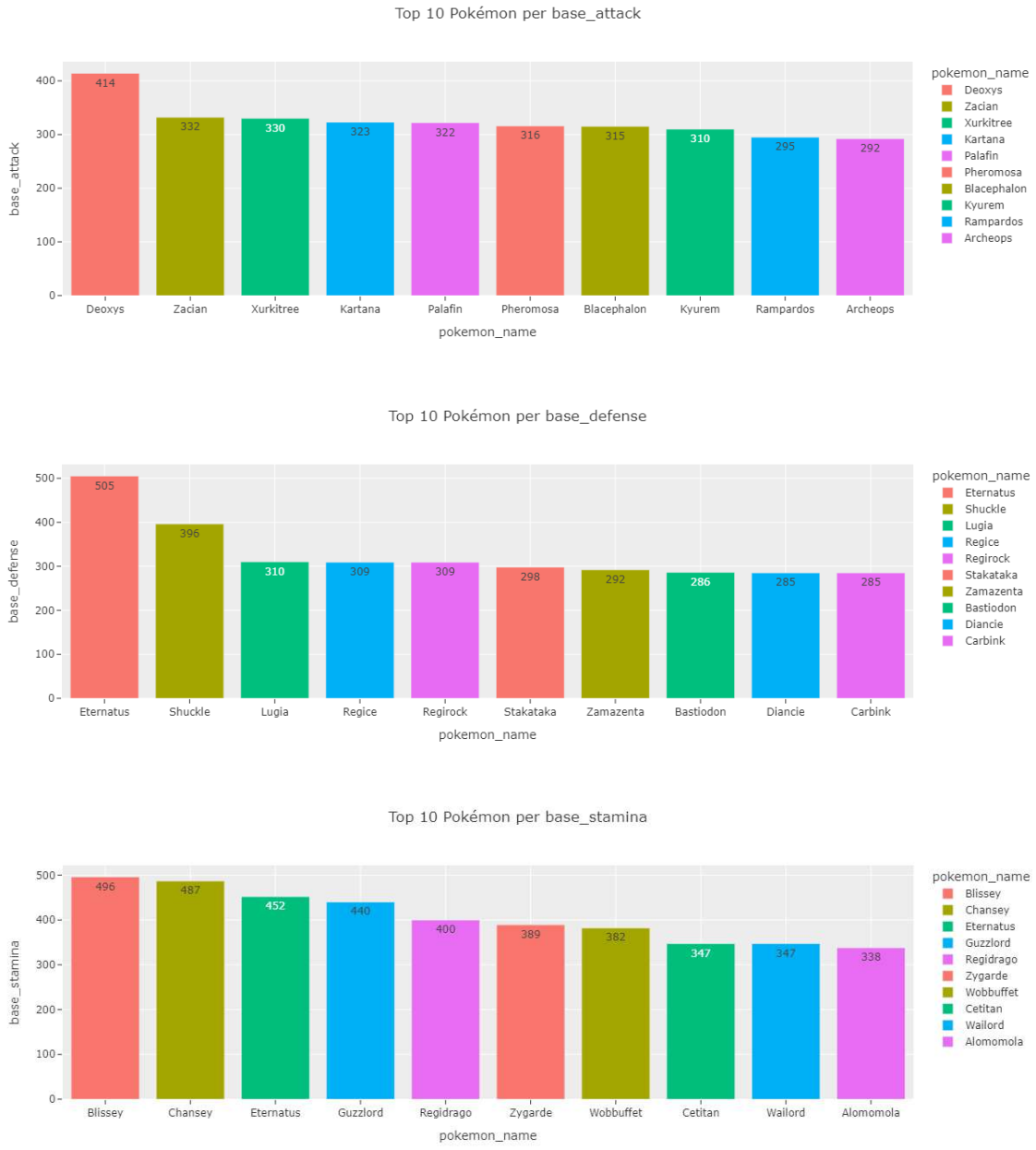


Figura 2.3: I primi 10 pokemon più forti per ogni statistica base

Dall'analisi è stato possibile osservare che il Pokémon con l'attacco base più elevato,

Deoxys, non apparisse tra i quattro Pokémon più forti mostrati precedentemente. Questo è dovuto al fatto che, per entrare nella lista dei Pokémon più forti, fosse necessario avere una somma delle statistiche base superiore a 750. Ciò significa che, sebbene *Deoxys* possegga l'attacco più alto, non ottiene un punteggio sufficiente nelle altre due statistiche base.

Grafici relativi all'andamento delle statistiche

Come se non bastasse, sono stati creati dei grafici relativi alle statistiche base sopracitate (**base_attack**, **base_defense** e **base_stamina**) per osservare la loro distribuzione all'interno del Dataset (Figura 2.4). Nel dettaglio, per ogni attributo sono stati generati tre tipi di grafico:

1. **Istogramma**: mostra la frequenza (conteggio) dei valori presenti nella variabile. L'istogramma fornisce una visione generale della distribuzione dei dati e della loro concentrazione in determinate aree.
2. **Q-Q plot**: confronta i quantili della distribuzione dei dati con i quantili di una distribuzione normale (gaussiana). Se i punti del grafico seguono una linea retta, significa che i dati sono distribuiti in maniera simile a una distribuzione normale. Eventuali deviazioni dalla retta potrebbero indicare la presenza di outlier o una distribuzione non normale.
3. **Boxplot**: mostra la mediana, il primo e il terzo quartile, e i "baffi" (whiskers), che indicano la dispersione dei dati. In questo caso, i punti al di fuori dei baffi vengono mostrati come outlier.

Analizzando questo output, si è osservato che:

- Trattando il grafico `base_attack`:
  1. *L'istogramma* mostra una distribuzione che appare relativamente normale, con una maggiore concentrazione di Pokémon attorno ai valori medi (intorno a 166.27). Tuttavia, è evidente una certa dispersione verso i valori più elevati, come indicato dal massimo di 414. Ciò suggerisce che ci siano alcuni Pokémon con attacchi significativamente superiori alla media.
  2. *Il Q-Q plot* conferma che la distribuzione dell'attacco segue una tendenza simile a una distribuzione normale, con lievi scostamenti nei valori estremi, indicanti la presenza di outlier.
  3. *Il boxplot*, invece, mostra chiaramente che la maggior parte dei valori si concentra tra il primo quartile (119) e il terzo quartile (211), con alcuni outlier sopra il terzo quartile, rappresentando Pokémon con attacchi estremamente elevati.
- Trattando il grafico `base_defense`:
  1. *L'istogramma* rivela una distribuzione che si concentra attorno ai valori medi (143.82), ma con una certa variabilità, come suggerito dalla deviazione standard di 52.02. Questo è visibile anche nel boxplot, che mostra un ampio intervallo interquartile tra il primo quartile (103) e il terzo quartile (179), con valori estremamente alti fino a 505, che possono essere considerati outlier.
  2. *Il Q-Q plot*, pur mostrando un buon allineamento con la distribuzione normale, evidenzia deviazioni nei quantili più estremi, confermando la presenza di outlier.
  3. *Il boxplot* mostra che la maggior parte dei valori si concentra tra 103 (primo quartile) e 179 (terzo quartile), ma ci sono alcuni valori molto elevati (oltre il terzo quartile), indicati come outlier, che rappresentano Pokémon con difese eccezionalmente alte, come quelli più rari o leggendari.
- Trattando il grafico `base_stamina`:
  1. *La distribuzione* è simile a quella di attacco e difesa, con la maggior parte dei Pokémon che hanno valori di resistenza attorno alla media di 155.28. L'istogramma mostra una distribuzione che tende a concentrarsi intorno ai valori centrali, ma con una certa dispersione verso i valori più alti (massimo di 496).
  2. *Il Q-Q plot* suggerisce anch'esso che la distribuzione segue abbastanza da vicino una normale, con lievi deviazioni nei quantili estremi.
  3. *Il boxplot* mostra che la maggior parte dei Pokémon si concentra tra il primo quartile (116) e il terzo quartile (190), con alcuni outlier nei valori più elevati di

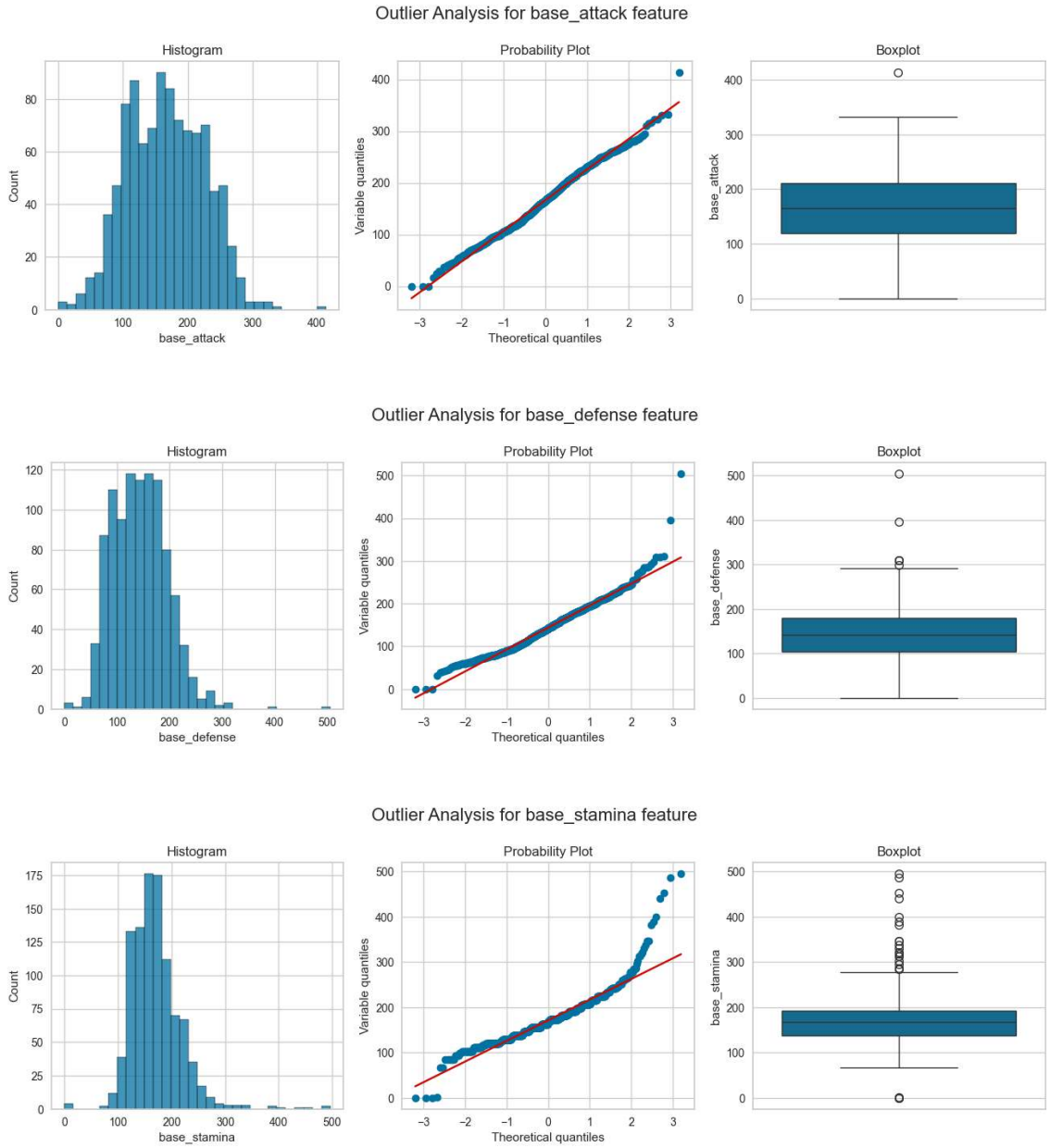


Figura 2.4: Grafici relativi alle statistiche base di ogni pokemon, In ordine: `base_attack`, `base_defense` e `base_stamina`

resistenza.

In generale, questi grafici hanno confermato che le statistiche di base di attacco, difesa e resistenza sono distribuite in modo relativamente simile, con una concentrazione centrale attorno ai valori medi, ma anche con una certa variabilità che evidenzia la presenza di Pokémon con caratteristiche fisiche significativamente più forti in alcuni casi.

La presenza di outlier, come osservato nei boxplot e nei Q-Q plot, potrebbe essere indicativa di Pokémon speciali o rari, con performance eccezionali in alcune di queste statistiche.

Top 10 pokémon sommati gli attributi base

Continuando ad analizzare i dati relativi alle statistiche di attacco, difesa e resistenza dei Pokémon, è stata effettuata una simulazione per determinare i dieci Pokémon più forti sulla base della somma di queste qualità. In particolare, sono stati considerati i parametri `base_attack`, `base_defense` e `base_stamina`, e per ciascun Pokémon è stata calcolata la somma di queste statistiche per ottenere un punteggio complessivo, `total_stats`.

L'analisi mostrata dalla Figura 2.5 ha evidenziato che *Eternatus* è il Pokémon con la somma più alta di queste statistiche, risultando il più forte tra tutti. I top 10 Pokémon rimanenti sono stati ordinati in base al valore di `total_stats`, con una rappresentazione visiva dei risultati tramite un grafico a barre.

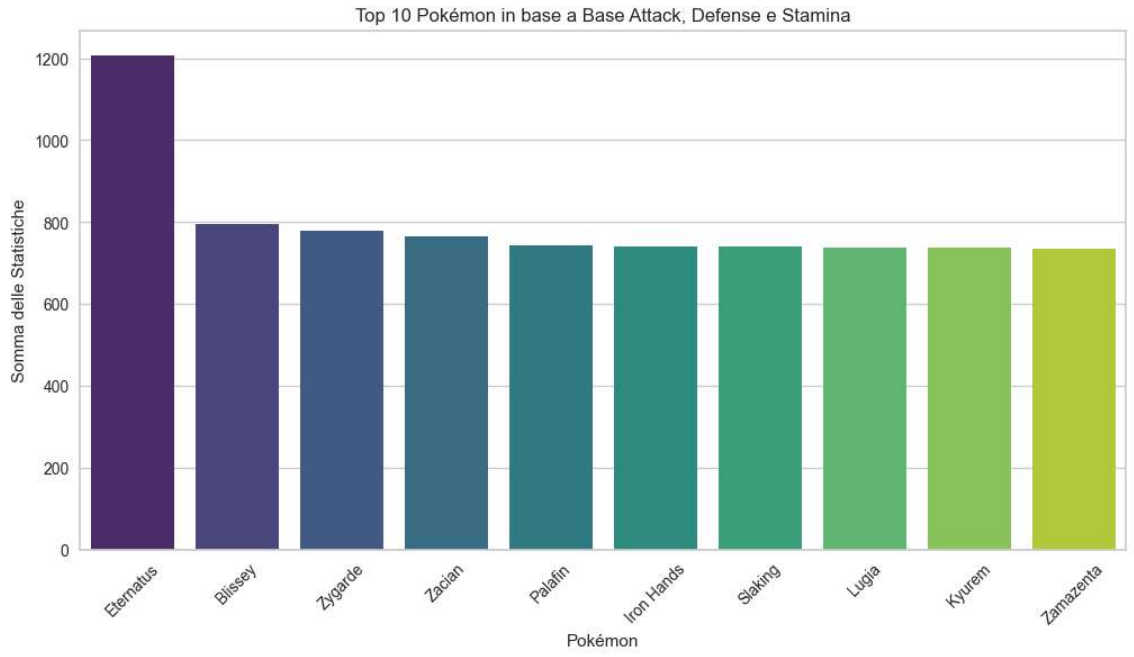


Figura 2.5: I 10 pokémon più forti data la somma delle loro statistiche

In generale, il grafico fornisce una visione chiara della distribuzione delle statistiche totali tra i Pokémon, con una netta separazione tra i primi dieci classificati e quelli successivi. Questo tipo di analisi ha evidenziato i Pokémon con le migliori performance complessive in base alle tre qualità selezionate.

Top 10 del tipo dei pokémon più forti

L'ultima analisi effettuata all'interno del dataset si è concentrata sulla determinazione del tipo dei Pokémon più forti (`type`), utilizzando la somma delle loro statistiche di base, ovvero `base_attack`, `base_defense` e `base_stamina`.

Per ottenere questo risultato, è stata calcolata la somma di queste statistiche per ogni Pokémon, definendo un nuovo valore `TOTAL`. Successivamente, i Pokémon sono stati ordinati in ordine decrescente rispetto a questa somma, così da identificare quelli con le performance complessive più elevate da cui ricavarne il tipo.

Dall'analisi mostrata in Figura 2.6 è emerso che il Pokémon più forte tra tutti appartenesse ai tipi *Poison* e *Dragon*, indicando che questi tipi vantano almeno un Pokémon con un punteggio complessivo estremamente elevato rispetto agli altri.

Inoltre, il grafico fornisce una chiara rappresentazione della forza relativa dei Pokémon più potenti per ciascun tipo, evidenziando come il tipo *Dragon* compaia ben quattro volte (quasi il 50% della top 10), racchiudendo così la maggior parte dei Pokémon più forti.

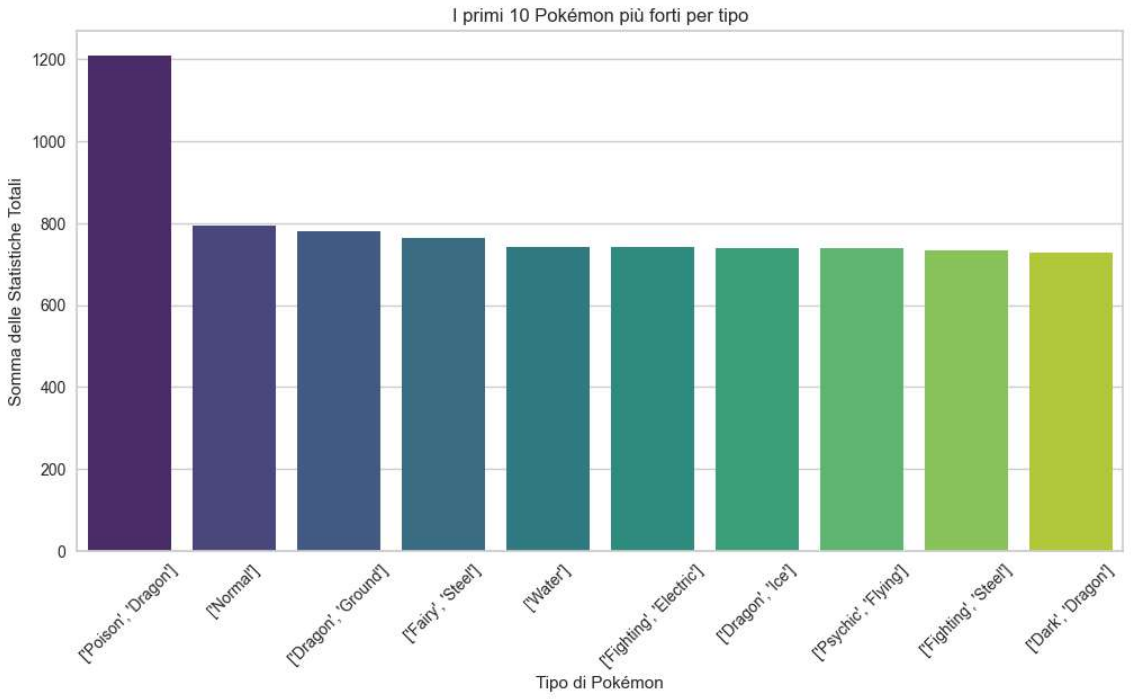


Figura 2.6: Il tipo dei pokémon più forti data la somma delle loro statisitche

Questo mostra come alcuni tipi di Pokémon abbiano una naturale predisposizione a eccellere nelle statistiche complessive, probabilmente a causa del loro design bilanciato tra attacco, difesa e stamina. Inoltre, evidenzia come il bilanciamento tra i diversi tipi nel gioco possa non essere uniforme, portando alcuni di essi a dominare le classifiche di forza complessiva.

2.2 Tecniche di clustering

Terminata l'analisi e visualizzazione del dataset, la seconda parte del capitolo è stata dedicata all'applicazione di diverse tecniche di clustering per raggruppare i Pokémon in base alle loro

statistiche di battaglia. In particolare, sono stati utilizzati i seguenti approcci:

- **K-Means:** L'algoritmo K-Means è stato impiegato per identificare gruppi omogenei. Tramite il *metodo del gomito*[1] è stato determinato il numero ottimale di cluster (in questo caso 4), scelta confermata dall'analisi della *silhouette score*. I risultati sono stati visualizzati tramite grafici $3D$ che mostrano la distribuzione dei Pokémon in funzione di `base_attack`, `base_defense` e `base_stamina`.
- **DBSCAN:** Per confrontare i risultati ottenuti con K-Means, è stato applicato anche l'algoritmo DBSCAN, che consente di identificare cluster basati sulla densità e di rilevare eventuali *outlier*. Il parametro `eps` è stato individuato mediante l'analisi del *k-distance plot*[2].

Scendendo nel dettaglio, il codice sviluppato segue una struttura ordinata:

1. Preprocessing dei dati: rimozione dei NaN, standardizzazione delle feature e analisi delle distribuzioni.
2. Applicazione dei metodi di determinazione del numero ottimale di cluster (metodo del gomito e silhouette analysis).
3. Implementazione dei clustering (K-Means e DBSCAN) e visualizzazione dei risultati tramite grafici 3D e plot comparativi.

2.2.1 ETL per il clustering

Come accennato precedentemente, la prima parte del codice è stata dedicata al preprocessing dei dati. Infatti, dopo aver verificato e rimosso le istanze contenenti valori mancanti nelle feature selezionate (ovvero `base_attack`, `base_defense` e `base_stamina`), si è proceduto con la standardizzazione dei dati.

Questa operazione è risultata fondamentale per garantire che ciascuna feature contribuisse in modo equilibrato al processo di clustering, evitando che differenze nelle scale potessero distorcere i risultati. Utilizzando lo `StandardScaler` di *scikit-learn*, le feature sono state trasformate in modo da avere una media pari a 0 e una deviazione standard pari a 1. In questo modo, le statistiche di battaglia dei Pokémon sono state riportate su una scala comune, facilitando l'analisi dei raggruppamenti basati sulle loro prestazioni relative.

Successivamente, è stata condotta un'analisi della correlazione tra le feature numeriche del dataset. In particolare, sono state selezionate tutte le colonne numeriche per calcolare la matrice di correlazione. Questo passaggio è stato fondamentale per identificare le relazioni lineari tra le statistiche di battaglia dei Pokémon e per comprendere come queste interagiscono tra loro. Infine, la matrice di correlazione è stata visualizzata attraverso una heatmap, che offre una rappresentazione visiva immediata delle relazioni esistenti, facilitando l'individuazione di pattern e possibili ridondanze nelle variabili.

L'output viene mostrato in Figura 2.7.

[1] Il metodo a gomito è una tecnica utilizzata per determinare il numero ottimale di cluster (k) in un algoritmo di clustering come K-Means.

[2] Il K-Distance Plot aiuta a scegliere il parametro eps (epsilon), che rappresenta il raggio massimo di vicinanza tra due punti affinché possano essere considerati nello stesso cluster in DBSCAN.

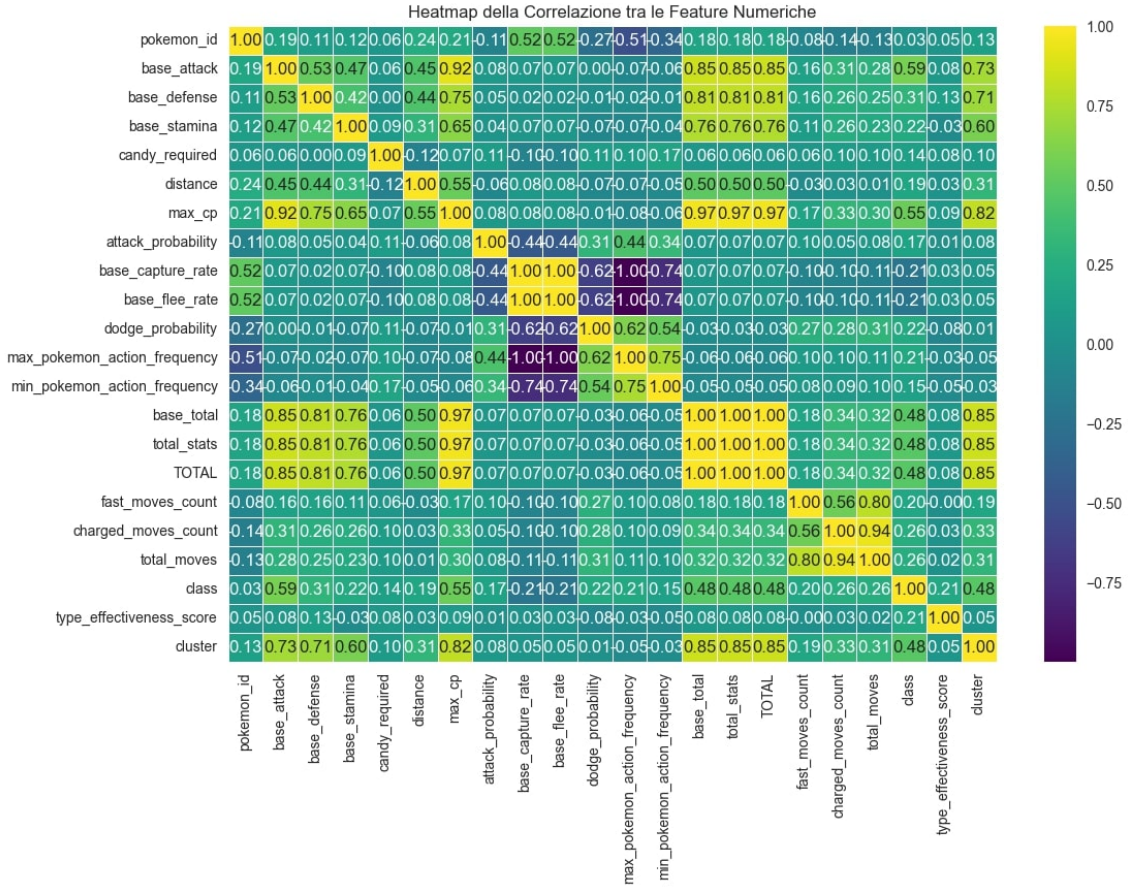


Figura 2.7: Heatmap della matrice di correlazione tra le feature numeriche

Infine, è stato eseguito un confronto delle distribuzioni delle feature prima e dopo il ridimensionamento. Il ridimensionamento, realizzato tramite standardizzazione, ha garantito che ciascuna feature avesse una media pari a 0 e una deviazione standard pari a 1, rendendo i dati comparabili per le successive analisi di clustering. L'analisi visiva tramite KDE plot ha permesso di verificare l'efficacia della procedura, evidenziando come le distribuzioni originali siano state trasformate in distribuzioni normalizzate.

Grazie a questo si è ottenuto il seguente confronto mostrato in Figura 2.8.

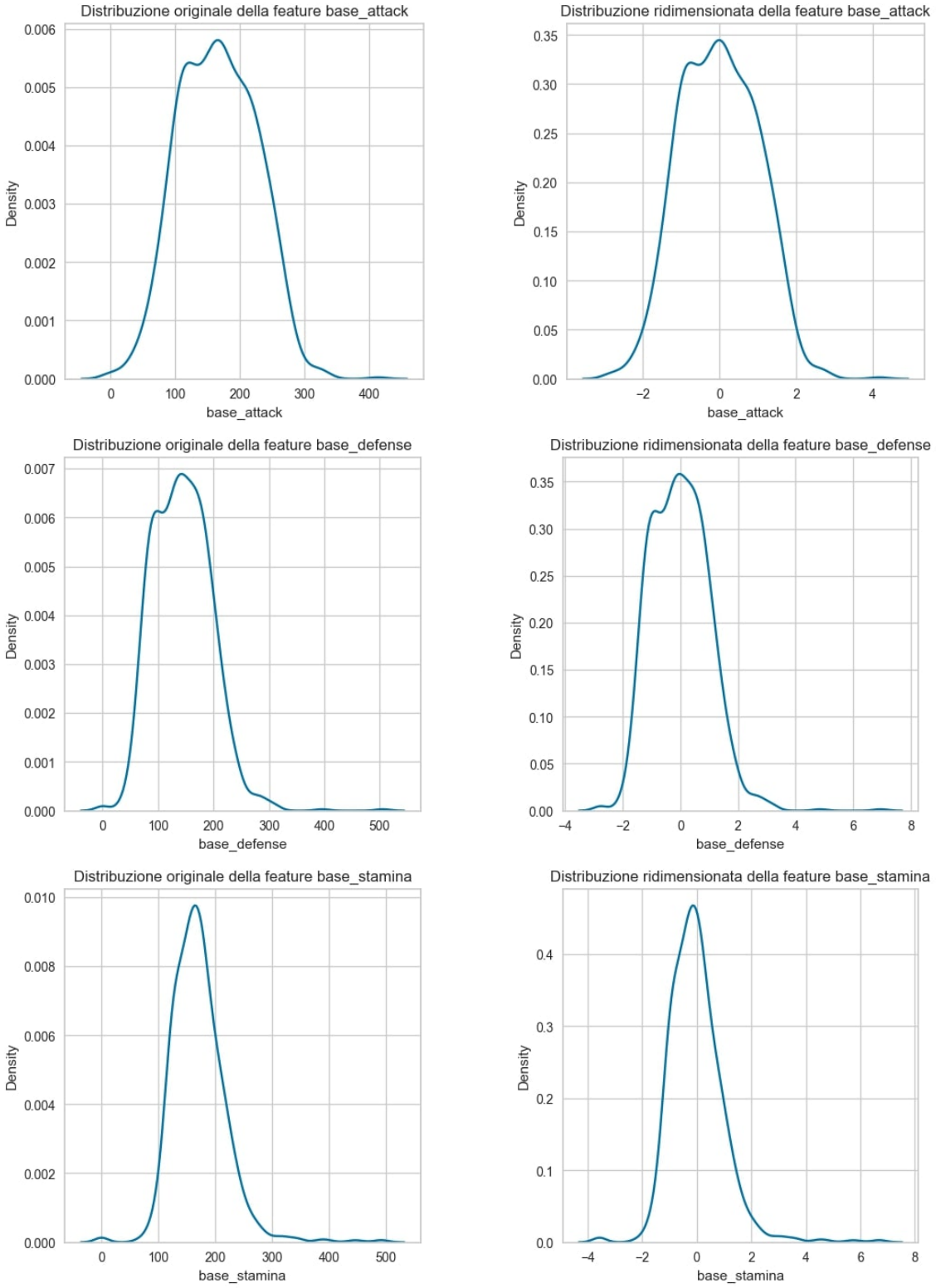


Figura 2.8: Confronto *pre* e *post* ridimensionamento delle statistiche base dei Pokémon (`base_attack`, `base_defense` e `base_stamina`)

2.2.2 Applicazione dei metodi

Terminata la fase di ETL e l'analisi qualitativa e quantitativa dei dati, si è passati all'applicazione dei metodi per determinare il numero ottimale di cluster.

Metodo del gomito

In particolare, il primo metodo adottato è stato il *metodo del gomito* (Figura 2.9). In questo approccio, l'algoritmo K-Means viene eseguito per un range di valori di k (da 1 a 10) e, per ciascun valore, si calcola l'*inertia*, ovvero la somma delle distanze quadrate tra i punti e il centro del cluster (**W***ithin-***C***luster* **S***um of* **S***quares*). Tracciando l'inertia in funzione del numero di cluster, si osserva il punto in cui la riduzione dell'inertia inizia a diminuire in modo significativo: questo punto, il cosiddetto "gomito", suggerisce il numero ottimale di cluster da utilizzare per il clustering.

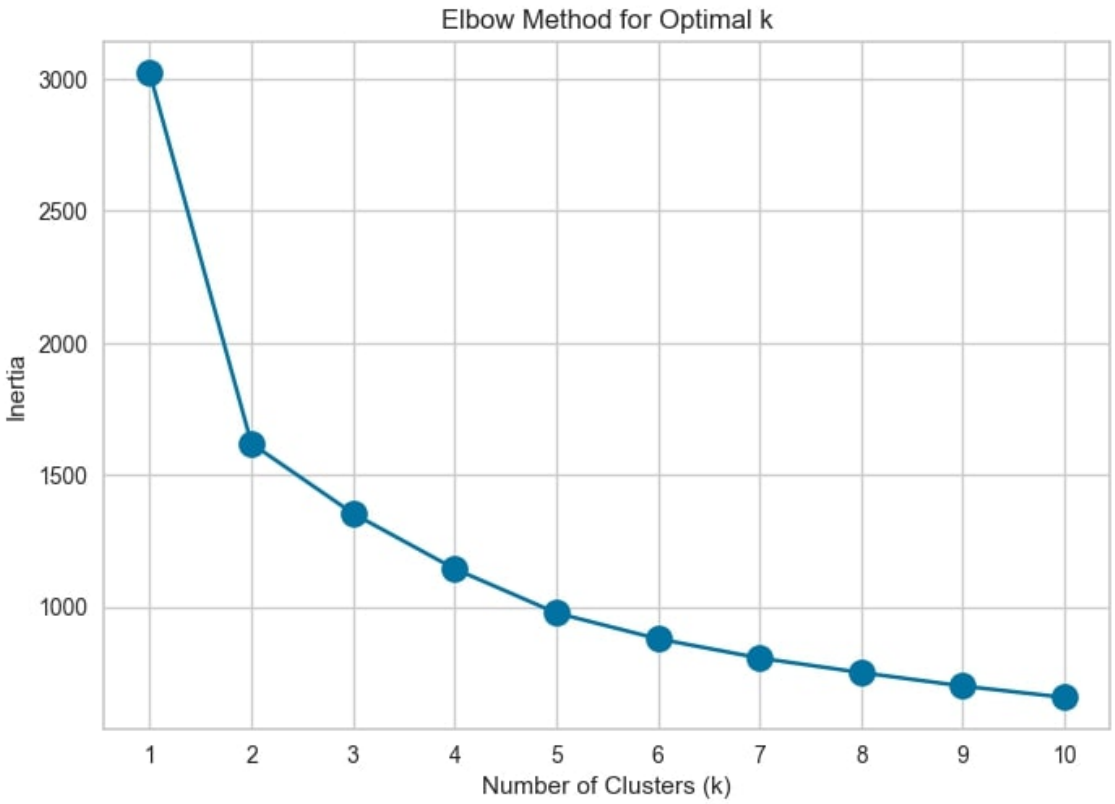


Figura 2.9: Grafico generato dal metodo del gomito

Sulla base dell'analisi del gomito illustrata nel grafico e nella tabella, dopo $k = 4$, non vi è alcuna diminuzione significativa dell'inerzia (meno del 10% della diminuzione dell'inerzia dopo il numero del cluster $= 4$). Pertanto, il numero ottimale di cluster dall'analisi del gomito è 4.

A valle dell'analisi condotta con il metodo del gomito, si è quindi proceduto all'applicazione di K-Means con $k = 4$. L'algoritmo suddivide i Pokémon in quattro gruppi sulla base delle loro statistiche di attacco, difesa e resistenza, consentendo di evidenziare eventuali somiglianze o differenze significative tra le diverse specie.

Come evidenziato in Figura 2.10, la rappresentazione 3D non solo suddivide chiaramente i Pokémon in quattro gruppi distinti, ma rivela anche differenze sostanziali nelle loro statistiche di attacco (*attack*), difesa (*defense*) e resistenza (*stamina*). In particolare:

- **Cluster Giallo**: molto compatto, posizionato nella regione con valori più bassi di Attacco, Difesa e Resistenza. Questo suggerisce che raggruppa Pokémon con statistiche complessivamente inferiori, probabilmente Pokémon base o meno competitivi in battaglia.
- **Cluster Blu**: Cluster denso ma più distribuito rispetto al giallo, con valori intermedi nelle tre statistiche. Potrebbe rappresentare Pokémon con un bilanciamento tra attacco, difesa e resistenza, adatti a strategie versatili.
- **Cluster Verde**: Il cluster più disperso, con punti distribuiti in un'ampia area. Questo suggerisce un gruppo eterogeneo di Pokémon, che potrebbero avere combinazioni di statistiche meno uniformi rispetto agli altri cluster. Potrebbe indicare Pokémon con specializzazioni differenti o una difficoltà nel clustering per questa categoria.
- **Cluster Viola**: Cluster ben definito e meno disperso rispetto al verde, con una leggera distribuzione su tutta l'area (*estremizzata da un singolo pokémon con la resistenza più elevata di tutti gli altri*). Probabilmente include Pokémon con statistiche più elevate, ma comunque distinguibili dagli altri gruppi.

Questa analisi non solo conferma la validità dell'approccio di clustering per la classificazione dei Pokémon, ma fornisce anche una chiara suddivisione basata sulle loro capacità di combattimento. L'eterogeneità tra i gruppi evidenzia come alcune categorie di Pokémon siano più adatte a strategie offensive, mentre altre possano eccellere in ruoli difensivi o di supporto.

Un aspetto particolarmente interessante è la distribuzione dei cluster: mentre il cluster giallo raccoglie Pokémon dalle statistiche più basse e quello blu rappresenta un gruppo più equilibrato e versatile, il cluster verde mostra una maggiore dispersione, suggerendo che i Pokémon in esso contenuti abbiano caratteristiche meno uniformi o che possano appartenere a sottogruppi distinti. Il cluster viola, invece, appare più compatto e identificabile, indicando Pokémon con valori elevati ma ben differenziati dagli altri gruppi.

Questa segmentazione ha suggerito l'esistenza di archetipi ben definiti all'interno del dataset, con potenziali implicazioni nell'ottimizzazione delle squadre e nell'analisi delle sinergie tra Pokémon.

Metodo della silhouette

Passando al secondo metodo adottato, la *silhouette* è una metrica che permette di valutare la bontà di un raggruppamento, misurando quanto ciascun punto sia ben assegnato al proprio cluster rispetto ai cluster vicini. Il valore della silhouette varia da -1 a 1: un valore più alto indica una migliore separazione tra i cluster, mentre valori prossimi allo zero o negativi suggeriscono una suddivisione meno efficace. Calcolando la silhouette per diversi valori di k (numero di cluster), è possibile identificare quello che massimizza questa metrica, fornendo un ulteriore criterio per stabilire la suddivisione ottimale dei dati.

Infatti, il grafico risultante (Figura 2.11) mostra come la silhouette tenda a diminuire con l'aumentare di k, evidenziando che un valore elevato di silhouette non garantisce necessariamente una separazione ottimale dei cluster. In questo caso, l'analisi ha portato

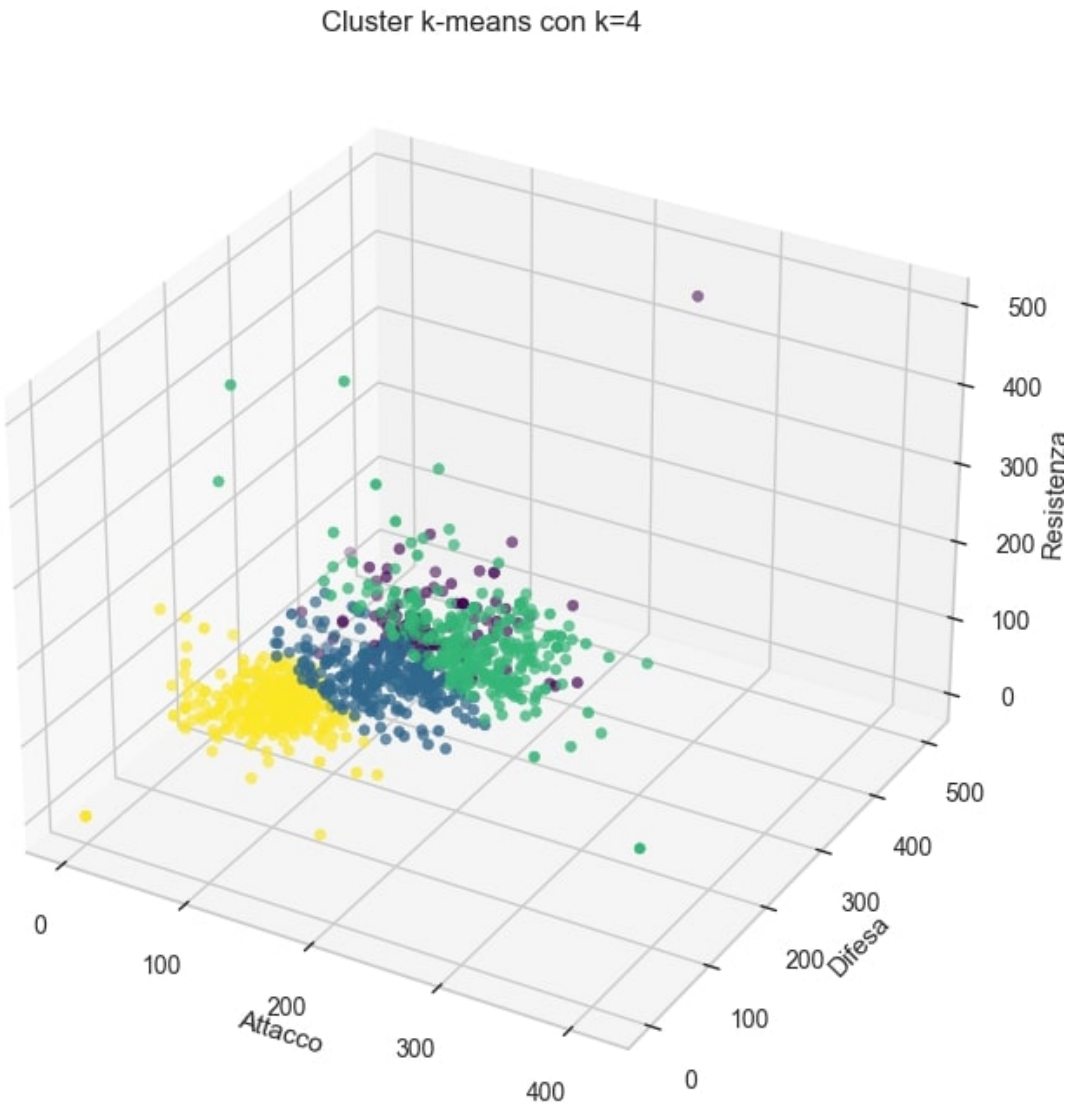


Figura 2.10: Grafico tridimensionale ponendo $k = 4$

all'individuazione di **2 cluster ottimali** con una media della silhouette score pari a 0.426. Questo valore suggerisce che, pur essendo la suddivisione in 2 gruppi la soluzione migliore tra quelle testate, la separazione tra i cluster non è estremamente netta, segnalando una certa sovrapposizione tra i gruppi.

Per completare l'analisi visiva della *silhouette* e verificare la qualità del clustering per diversi valori di k, è stato utilizzato il `SilhouetteVisualizer` della libreria *Yellowbrick*. In particolare, il codice usato genera, per ogni valore di k da 2 a 10, un grafico che mostra la distribuzione dei coefficienti di silhouette all'interno di ciascun cluster, oltre alla media complessiva rappresentata da una linea verticale tratteggiata.

Ogni grafico illustra la forma della distribuzione dei coefficienti di silhouette per i vari cluster, aiutando a identificare eventuali sovrapposizioni o punti che potrebbero essere stati assegnati a un cluster non ideale. La linea tratteggiata rossa indica il valore medio della silhouette, fornendo un riferimento rapido per confrontare la coesione e la separazione tra i cluster. Per comodità, in Figura 2.12 viene mostrato solo il primo grafico generato (relativo a $k = 2$), che risulta coerente con l'analisi precedente: l'*average silhouette score* di 0.426

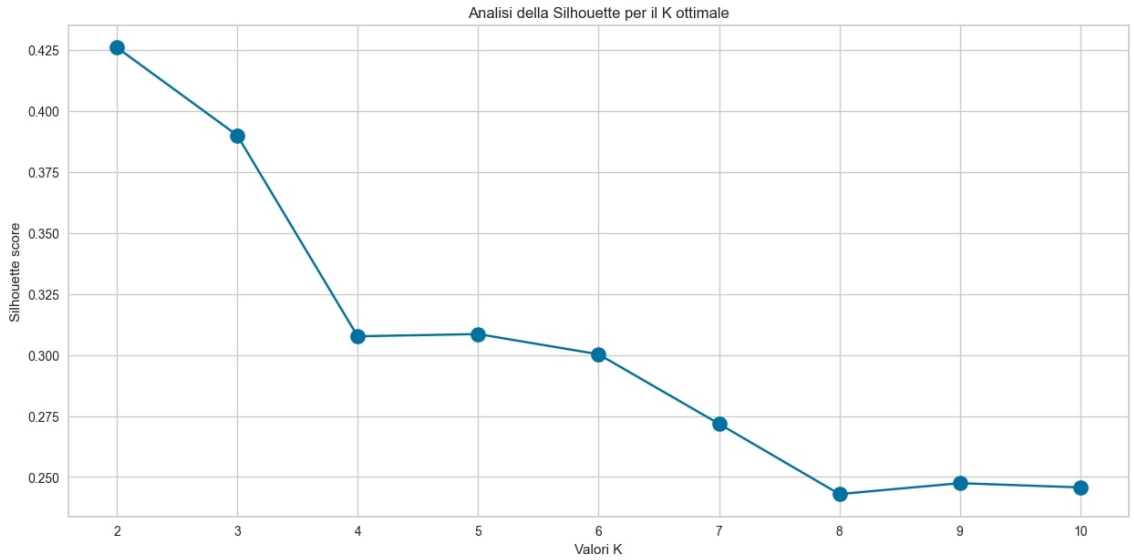


Figura 2.11: Analisi della Silhouette per il K ottimale

conferma infatti che, pur essendo la soluzione migliore fra quelle testate, la separazione tra i cluster non è particolarmente marcata.

Inoltre, da esso è stato possibile determinare come il

- **Cluster Verde** sembrasse contenere un numero maggiore di campioni e mostra una certa variabilità nei valori di silhouette. Ciò suggerirebbe che all'interno di questo cluster convivano Pokémon con diversi gradi di *appartenenza* al gruppo, con alcuni esemplari che potrebbero essere borderline.
- **Cluster Blu** fosse più compatto e meno esteso. Ciò indicherebbe che i Pokémon al suo interno tendono ad avere caratteristiche più simili tra loro, risultando in valori di silhouette generalmente più uniformi.

Il confronto visivo tra i grafici relativi a diversi valori di k ha consentito di valutare più agevolmente la qualità del clustering e di individuare il numero di gruppi che meglio bilanciava coesione interna e separazione esterna.

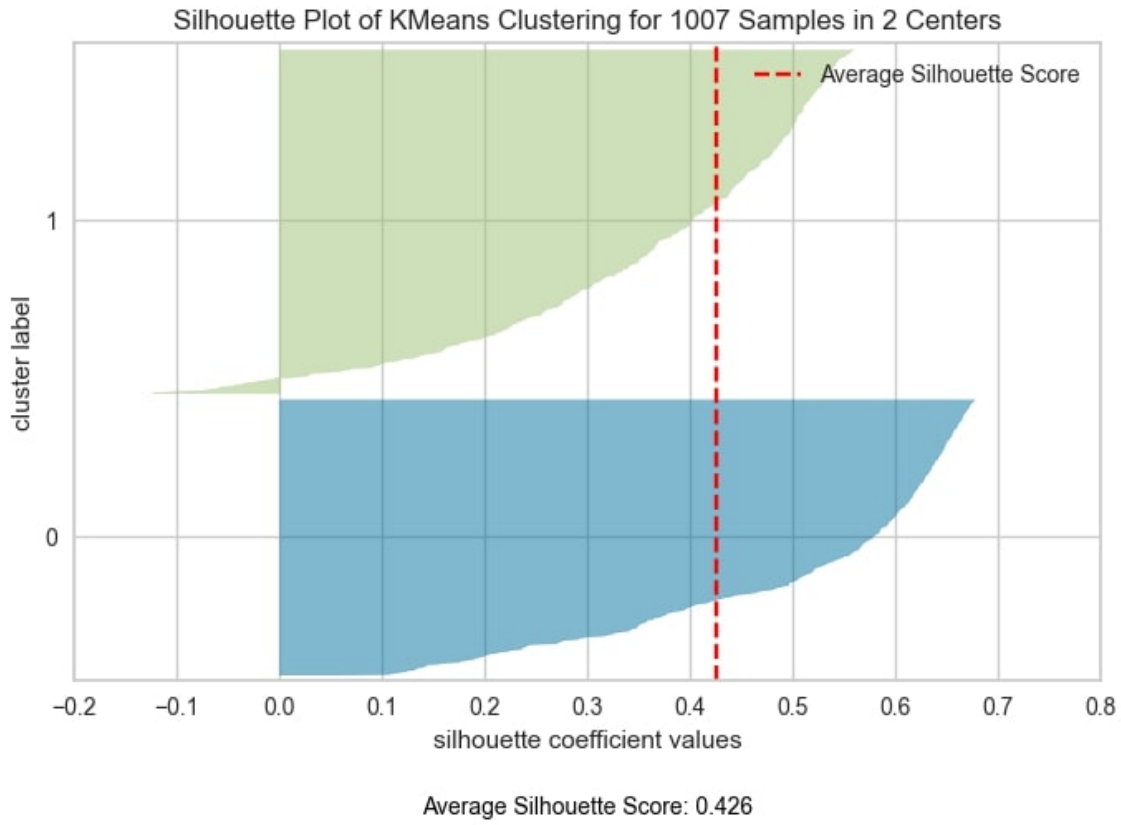


Figura 2.12: Grafico della Silhouette in 2 *centri*

Metodo DBSCAN

Proseguendo con l'analisi dei dati, si è utilizzato l'algoritmo **DBSCAN** (**D***ensity-***B***ased* **S***patial* **C***lustering of* **A***pplications with* **N***oise*). Questo metodo di clustering si basa sulla densità dei punti ed è particolarmente utile quando si vuole identificare strutture di dati che non sono necessariamente separabili in cluster di forma sferica, come avviene con il metodo *K-Means*.

L'algoritmo DBSCAN richiede due parametri fondamentali: ε (epsilon), ovvero la distanza massima tra due punti affinché possano essere considerati parte dello stesso cluster, e $minPts$, il numero minimo di punti richiesto affinché un gruppo di punti sia considerato un cluster denso.

A differenza di K-Means, DBSCAN può identificare punti che non appartengono a nessun cluster, classificandoli come outlier. Per determinare un valore adeguato per ε*, si utilizza un* **k-distance plot***, che consente di individuare il punto di gomito nella curva delle distanze tra i punti e i loro k-esimi vicini.*

Per determinare un valore ottimale di ε, è stato applicato il seguente codice:

```
# Definisce il valore di k (numero di vicini)
k = 4  # Si puo cambiare questo valore in base alle necessita'

# Calcola le distanze dai vicini piu' prossimi
neigh = NearestNeighbors(n_neighbors=k)
neigh.fit(scaled_features)
distances, indices = neigh.kneighbors(scaled_features)

```

```
# La distanza al k-esimo vicino per ogni punto (prendiamo la k-esima distanza da ogni punto)
k_distances = distances[:, -1]  # L'ultima colonna corrisponde alla k-esima distanza

# Ordina le distanze in ordine crescente
k_distances = np.sort(k_distances)

# Calcola la derivata per trovare il punto di massima variazione
derivative = np.diff(k_distances)  # Differenza tra distanze successive
# Trova il punto in cui la derivata e' massima (punto di gomito)
elbow_index = np.argmax(derivative)

# Traccia il k-distance plot
plt.figure(figsize=(8, 6))
plt.plot(np.arange(1, len(k_distances) + 1), k_distances, label='Distanza al k-esimo vicino', color='blue')

# Aggiunge il punto del gomito (dove la derivata e massima)
plt.axvline(x=elbow_index + 1, color='red', linestyle='--', label=f'Punto del gomito (eps={k_distances[elbow_index]:.2f})')

# Aggiunge un'etichetta al punto
plt.text(elbow_index + 1, k_distances[elbow_index] + 0.05, f'eps={k_distances[elbow_index]:.2f}',
         color='red', fontsize=12, ha='center')

# Titolo e etichette
plt.title(f"k-Distance Plot (k={k}) con il punto dell'eps")
plt.xlabel("Punti")
plt.ylabel(f"Distanza al {k}-esimo vicino piu prossimo")
plt.grid(True)
plt.legend()
plt.show()
```

Il **k-Distance Plot** risultante è mostrato in Figura 2.13.

Dall'analisi del grafico, si è osservate come le distanze ai k-esimi vicini crescono lentamente per la maggior parte dei punti, fino a raggiungere un punto di gomito in cui la pendenza aumenta improvvisamente. Questo punto di gomito rappresenta il valore ottimale per ε, che in questo caso è pari a 3.07.

Il punto di gomito viene identificato come la massima variazione nella curva delle distanze, e fornisce un criterio per scegliere ε in modo oggettivo. Questo parametro, come esplicato precedentemente, sarà utilizzato nell'algoritmo DBSCAN per determinare quali punti appartengono allo stesso cluster e quali invece verranno considerati outlier.

Proseguendo con l'analisi, una volta determinato il valore di $\varepsilon = 3.07$, è stato applicato

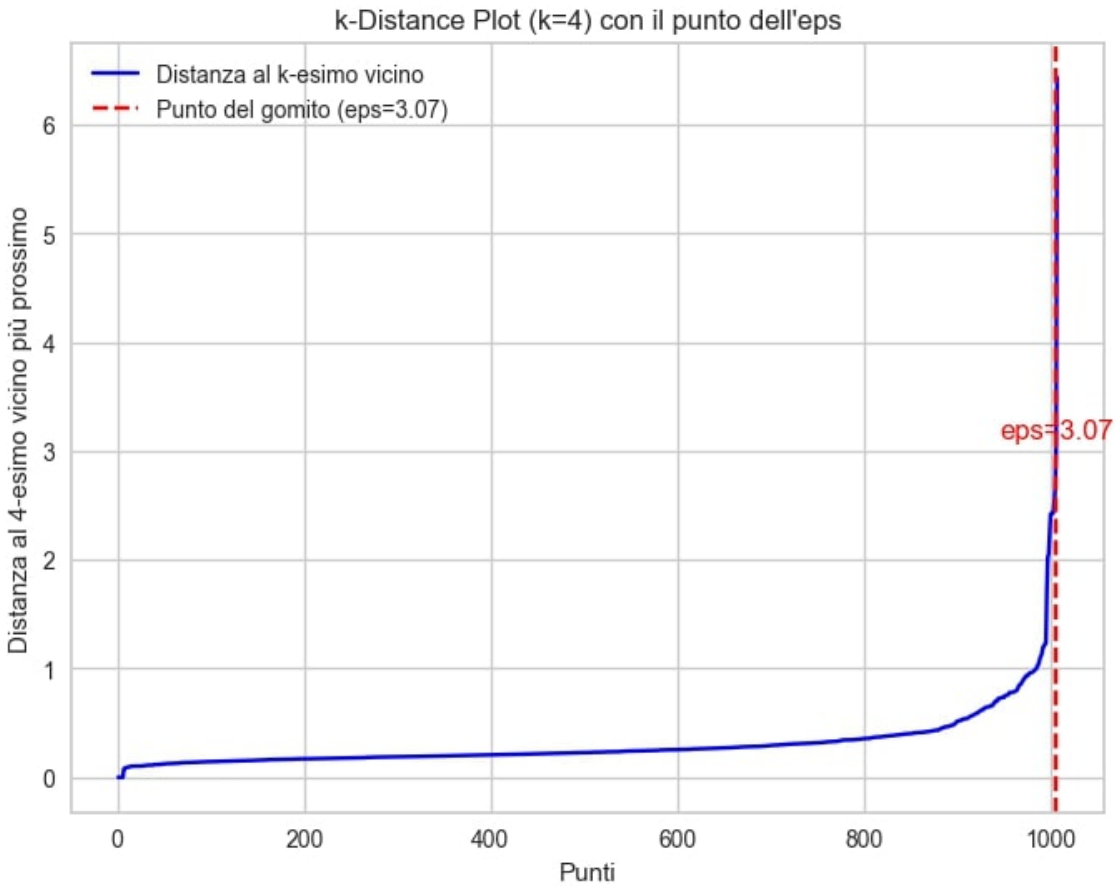


Figura 2.13: Grafico che permette la visualizzazione di ε

l'algoritmo **DBSCAN** al dataset ridimensionato. In questo modo, ogni Pokémon è stato assegnato a un cluster specifico o, nel caso in cui non soddisfi i requisiti minimi di densità (stabiliti da ε e dal numero minimo di punti *minPts*), è stato etichettato come *outlier* (cluster -1).

La Figura 2.14 mostra la distribuzione dei Pokémon nello spazio definito da `base_attack`, `base_defense` e `base_stamina`, con colori diversi in base all'etichetta di cluster. Nello specifico:

- I punti in **Giallo** corrispondono al **cluster** 0, che raccoglie la maggior parte dei Pokémon, suggerendo che il dataset presenti un'unica grande regione densa.
- I punti in **Viola** (**cluster** -1) rappresentano i cosiddetti **outlier**, ossia Pokémon che non rientrano in nessun cluster sufficientemente denso in base ai criteri stabiliti da ε e *minPts*.

Come si può osservare, l'algoritmo ha identificato un unico cluster dominante e un numero esiguo di punti isolati, etichettati come *outlier*. Ciò indica che la maggior parte dei Pokémon presenta caratteristiche (in termini di attacco, difesa e stamina) sufficientemente simili da formare un'unica regione densa, mentre pochi esemplari (praticamente uno solo) mostrano valori anomali o troppo distanti dal gruppo principale.

2.2.3 Confronto dei metodi

Per mettere a confronto i risultati di K-Means con un diverso numero di cluster ($k = 4$ e $k = 2$), è stato realizzato un semplice esperimento di visualizzazione in 3D. In particolare,

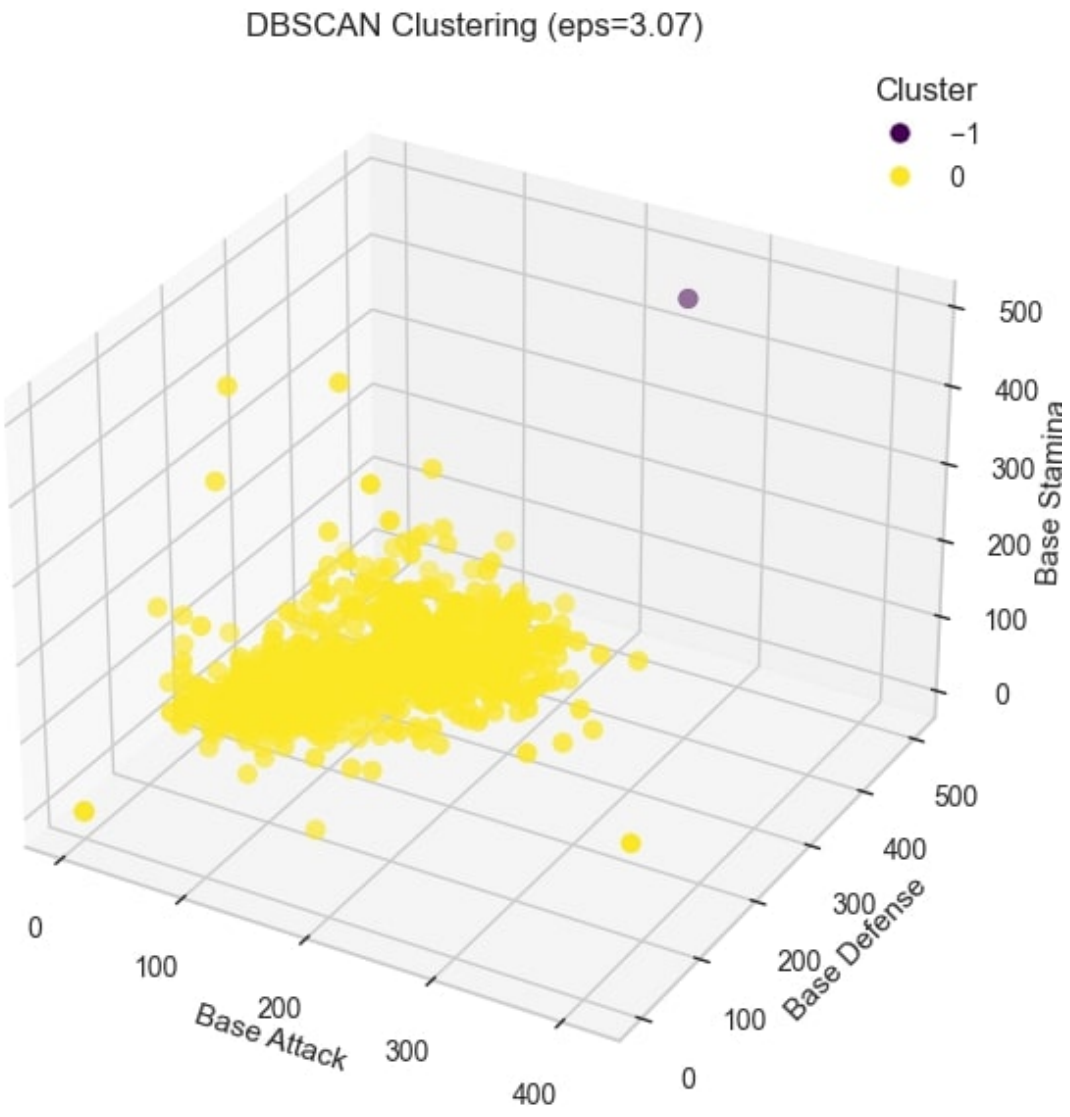


Figura 2.14: DBSCAN Clustering con $\varepsilon = 3.07$

dopo aver applicato K-Means, si è effettuata una riduzione dimensionale a 3 componenti principali (*Principal Component Analysis*, PCA) per consentire la rappresentazione su uno spazio tridimensionale.

Osservando i due grafici (Figura 2.15), si nota che con $k = 4$ l'algoritmo ha individuato quattro gruppi distinti, in linea con i risultati del *metodo del gomito*, che segnalava un punto di flesso significativo a $k = 4$. Con $k = 2$, invece, i Pokémon sono stati suddivisi in due gruppi più ampi, come indicato dal *metodo della silhouette*, il quale suggeriva un numero di cluster più basso per massimizzare la separazione media tra i punti.

La scelta finale tra $k = 4$ e $k = 2$ dipende dall'obiettivo dell'analisi: una partizione più granulare può mettere in risalto differenze sottili tra i Pokémon, mentre una divisione più ampia risulta più semplice da interpretare e offre una distinzione netta.

Per concludere l'analisi, è stato realizzato un confronto tra diversi algoritmi di clustering applicati allo stesso dataset. In particolare, sono stati presi in esame K-Means (con $k = 4$) e DBSCAN (con $\varepsilon = 3.05$ e **min_samples** $= 5$). Per valutare le performance di ciascun algoritmo, è stato calcolato il Silhouette Score, una metrica che misura quanto bene ogni

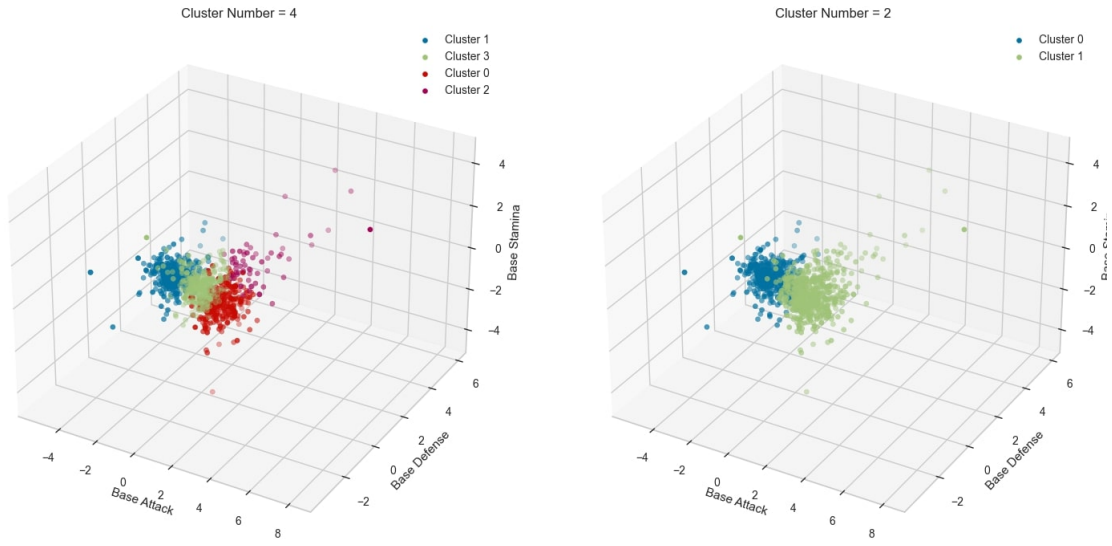


Figura 2.15: Confronto $k = 4$ e $k = 2$

punto si trovi nel proprio cluster rispetto ai cluster vicini.

Dall'output (Figura 2.16), è emerso che DBSCAN presenta un Silhouette Score notevolmente più alto rispetto a K-Means. Ciò indica che, nel contesto di questo dataset e con i parametri scelti, DBSCAN è in grado di individuare gruppi di punti meglio separati tra loro e con maggiore coesione interna. Tuttavia, è importante sottolineare che la scelta dell'algoritmo dipende anche dalla forma e dalla distribuzione dei dati, nonché dal significato pratico dei cluster.

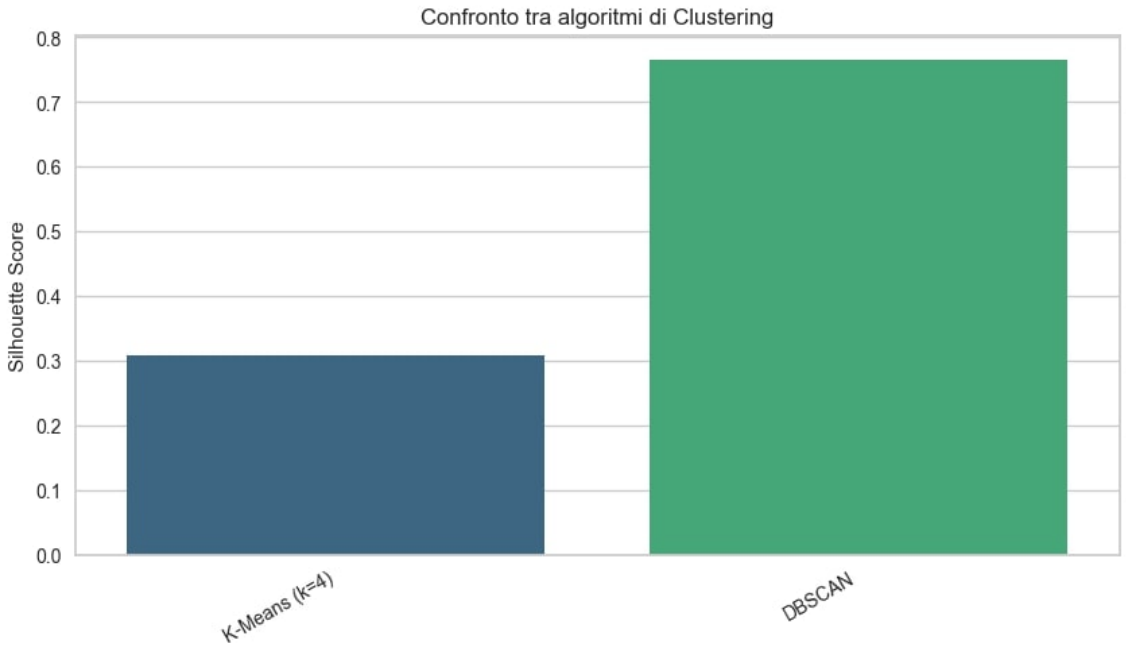


Figura 2.16: Confronto tra gli algoritmi di clustering

In particolare:

- **K-Means (k=4):** Suddivide i dati in quattro cluster distinti, in linea con il punto di

gomito identificato. Tuttavia, se i cluster non sono di forma sferica o presentano densità differenti, l'algoritmo potrebbe non separare adeguatamente alcuni gruppi.

- **DBSCAN ($\varepsilon = 3.07$, min_samples $= 5$):** Si basa sulla densità locale dei punti, permettendo di identificare cluster di forme diverse e rilevando eventuali outlier. In questo caso, l'algoritmo ha ottenuto un Silhouette Score più elevato, segnalando una separazione più netta tra i gruppi individuati.

In conclusione, DBSCAN è sembrato fornire una partizione più coerente e densa dei Pokémon rispetto a K-Means, almeno secondo il criterio della silhouette. È tuttavia fondamentale considerare che i parametri di DBSCAN, in particolare ε e **min_samples**, possono influire notevolmente sul risultato, e che la scelta dell'algoritmo migliore dipende sempre dagli obiettivi e dalle caratteristiche specifiche dell'analisi.

3. Classification

La classificazione è una delle tecniche fondamentali dell'apprendimento automatico e consiste nell'assegnare a ciascun elemento di un dataset una **categoria** o **classe** sulla base delle sue caratteristiche. L'obiettivo è costruire un modello che sia in grado di apprendere dai dati di addestramento e generalizzare le sue previsioni su dati nuovi e mai visti.

Nel contesto di questa analisi, la classificazione è stata utilizzata per determinare la categoria di appartenenza dei Pokémon, basandosi su parametri numerici come *attacco*, *difesa*, *resistenza* e altre statistiche di combattimento.

Questo può essere utile per prevedere, ad esempio, se un Pokémon è più adatto al **PvP** o al **PvE**, oppure per classificare la sua efficacia in battaglia.

3.1 ETL per la classificazione

Prima di poter applicare un algoritmo di classificazione, è stato necessario preparare il dataset, garantendo che tutte le colonne contenessero dati numerici e privi di valori mancanti.

Successivamente, oltre alla conversione e alla gestione delle colonne numeriche, per migliorare la capacità predittiva del modello di classificazione è stato creato un nuovo attributo che rappresenta il numero totale di mosse che ogni Pokémon possiede. Le mosse sono suddivise in due categorie: `fast_moves` e `charged_moves`. Tuttavia, queste colonne contengono stringhe che rappresentano liste di mosse.

Per trasformare questi dati in un formato numerico, è stata definita la funzione `count_moves()`, che tenta di convertire la stringa in una lista e restituisce il numero di elementi. Se la conversione fallisce o il valore è mancante, la funzione restituisce 0. In questo modo, si ottengono due nuove colonne che contengono il conteggio delle mosse per ciascun Pokémon.

Successivamente, è stata creata una nuova feature, `total_moves`, ottenuta sommando il conteggio delle mosse veloci e caricate, fornendo così un ulteriore indicatore della varietà e della capacità offensiva del Pokémon.

Il seguente output prova la riuscita creazione delle nuove colonne:

```
      pokemon_name                        fast_moves  fast_moves_count  \
0        Bulbasaur            ['Vine Whip', 'Tackle']                 2
1          Ivysaur         ['Razor Leaf', 'Vine Whip']                 2
2          Venusaur         ['Razor Leaf', 'Vine Whip']                 2
3        Charmander                ['Ember', 'Scratch']                 2
4        Charmeleon              ['Ember', 'Fire Fang']                 2
...             ...                                 ...               ...
1002        Chi-Yu            ['Incinerate', 'Snarl']                 2
1003  Roaring Moon                          ['Splash']                 1
1004  Iron Valiant                          ['Splash']                 1
1005     Koraidon       ['Rock Smash', 'Dragon Tail']                 2
1006     Miraidon  ['Thunder Shock', 'Dragon Breath']                 2
```

```
                                                charged_moves  charged_moves_count
0              ['Sludge Bomb', 'Seed Bomb', 'Power Whip']                      3
1            ['Sludge Bomb', 'Solar Beam', 'Power Whip']                       3
2         ['Sludge Bomb', 'Petal Blizzard', 'Solar Beam']                      3
3         ['Flame Charge', 'Flame Burst', 'Flamethrower']                      3
4           ['Fire Punch', 'Flame Burst', 'Flamethrower']                      3
...                                                   ...                    ...
1002        ['Dark Pulse', 'Flame Charge', 'Flame Wheel']                      3
1003                                          ['Struggle']                     1
1004                                          ['Struggle']                     1
1005  ['Giga Impact', 'Dragon Claw', 'Close Combat',...                         4
1006  ['Hyper Beam', 'Dragon Pulse', 'Thunder', 'Out...                          4

[1007 rows x 5 columns]
```

Oltre alle statistiche di combattimento già elaborate, è stato introdotto un ulteriore elemento informativo: un punteggio che valuta l'efficacia del tipo del Pokémon. Questo punteggio, che verrà utilizzato come feature nel modello di classificazione, consente di differenziare i Pokémon anche in base alla loro tipologia e alla capacità di contrastare altri tipi.

Per ottenere questo punteggio sono state eseguite le seguenti operazioni:

- **Estrazione dei tipi:** Dato che nella colonna `type` vi erano contenute più tipologie per ciascun Pokémon (ad esempio, `['Grass', 'Poison']`). Per gestire questa situazione, è stata definita la funzione `get_types()` che converte la stringa in una lista di tipi, restituendo tutti i tipi associati al Pokémon.
- **Mappatura dell'efficacia dei tipi:** È stato creato un dizionario, `type_effectiveness`, che associa ad ogni tipo un insieme di coefficienti. Questi coefficienti rappresentano l'efficacia offensiva di un tipo nei confronti degli altri, ad esempio un Pokémon di tipo `Fire` potrebbe essere particolarmente efficace contro un Pokémon di tipo `Grass` ma meno contro uno di tipo `Water`.
- **Calcolo del punteggio:** La funzione `get_type_effectiveness()` itera su tutti i tipi di un Pokémon e moltiplica i coefficienti corrispondenti per ottenere un punteggio complessivo, indicato come `type_importance` o `type_effectiveness`. Questo punteggio fornisce un'indicazione della capacità del Pokémon di fronteggiare un determinato tipo di avversario, e può essere utilizzato come ulteriore feature nel modello di classificazione.

Il punteggio ottenuto, indicato come `type_importance`, ha permesso di dare un valore quantitativo all'efficacia del tipo di ciascun Pokémon in relazione ad altri tipi. Questa feature, combinata con le altre statistiche di combattimento, offre un quadro più completo delle potenzialità offensive e difensive dei Pokémon, contribuendo così a una classificazione più precisa.

3.2 Analisi classificazione PvE/PvP

Nel seguente passaggio, si è proceduto ad assegnare ad ogni Pokémon una classe che indichi se è maggiormente adatto a combattimenti **PvP** (**P***layer* **v***s* **P***layer*) o a quelli **PvE** (**P***layer* **v***s* **E***nvironment*). L'idea alla base di questa classificazione è di utilizzare una combinazione di statistiche di combattimento (come `base_attack`, `base_defense`, `base_stamina`, `max_cp` e `dodge_probability`), insieme ad altre feature derivate (ad esempio il conteggio delle mosse e il punteggio di `type_effectiveness`) per determinare il potenziale utilizzo del Pokémon in contesti diversi.

La funzione `classify_pokemon()` implementa questa logica:

- In primis, viene valutato se il Pokémon possiede almeno due mosse veloci (`fast_moves_count` ≥ 2).
- Successivamente, si verificano delle condizioni specifiche sui parametri di combattimento. Queste condizioni variano in base al valore di `max_cp` e sono impostate in modo da individuare soglie minime per `base_attack`, `base_defense`, `base_stamina` e `dodge_probability` che, se rispettate, indicano un potenziale per il PvP.
- In aggiunta, se il punteggio complessivo dell'efficacia dei tipi (calcolato dalla funzione `get_type_effectiveness()` iterando su tutti i tipi del Pokémon) supera una certa soglia (in questo caso, 25), il Pokémon viene classificato come adatto al PvP.

Se le condizioni sopra non sono soddisfatte, il Pokémon viene classificato come PvE (0), altrimenti come PvP (1).

L'output prodotto, riportato di seguito, mostra le prime righe del dataset con la nuova colonna `class`:

```
Classificazione PvE (0) vs PvP (1):
       pokemon_name  class
0         Bulbasaur      0
1           Ivysaur      0
2          Venusaur      1
3        Charmander      0
4        Charmeleon      0
...             ...    ...
1002         Chi-Yu      0
1003   Roaring Moon      0
1004   Iron Valiant      0
1005      Koraidon       0
1006      Miraidon       0

[1007 rows x 2 columns]
```

Da questo output si evince che alcuni Pokémon, come *Venusaur*, sono stati classificati come adatti per il PvP (valore 1), mentre la maggior parte viene indicata come più adatta al PvE (valore 0).

Per valutare l'equilibrio tra le due classi ottenute, è stata visualizzata la distribuzione delle etichette PvE (0) e PvP (1) nel dataset. In questo modo, si è potuto avere una panoramica immediata della prevalenza di Pokémon classificati in ciascuna categoria:

```
Distribuzione classi PvE (0) vs PvP (1):
 class
0    0.696127
1    0.303873
Name: proportion, dtype: float64
```

Da questo risultato si evince che circa il 69.6% dei Pokémon è classificato come PvE, mentre il 30.4% risulta adatto al PvP. Tale distribuzione suggerisce una prevalenza dei Pokémon orientati al PvE nel dataset, pur evidenziando una quota significativa di esemplari adatti al PvP.

Successivamente, per integrare ulteriormente le informazioni utili alla classificazione, è stato calcolato un punteggio di efficacia totale del tipo per ogni Pokémon, denominato

`type_effectiveness_score`. In pratica, è stata definita la funzione `calculate_type_effectiveness()` che, per ogni Pokémon, itera su tutti i possibili tipi (definiti nel dizionario `type_effectiveness`) e somma i coefficienti di efficacia ottenuti dalla funzione `get_type_effectiveness()`. Questo punteggio rappresenta una misura quantitativa dell'efficacia complessiva del tipo o dei tipi di un Pokémon contro gli altri, e viene aggiunto al dataset come nuova feature. Tale informazione, combinata con le altre statistiche di combattimento, offre un quadro più completo delle potenzialità offensive e difensive, e può essere utilizzata per affinare il modello di classificazione.

L'output ottenuto è stato il seguente:

```
Efficacia totale per ogni Pokémon:
      pokemon_name  type_effectiveness_score
0        Bulbasaur                     22.50
1          Ivysaur                     22.50
2         Venusaur                     22.50
3       Charmander                     22.50
4       Charmeleon                     22.50
...            ...                       ...
1002        Chi-Yu                     23.50
1003  Roaring Moon                     20.75
1004  Iron Valiant                     23.50
1005      Koraidon                     21.75
1006      Miraidon                     21.00

[1007 rows x 2 columns]
```

Dall'output si evince che il punteggio di efficacia dei tipi varia tra i Pokémon, con valori che oscillano, ad esempio, tra 20.75 e 23.50. Questo punteggio permette di confrontare in modo quantitativo l'efficacia di ciascun Pokémon, fornendo un'ulteriore metrica che, insieme alle altre statistiche di combattimento, contribuisce a definire il profilo strategico di ciascun esemplare.

3.3 Modello di classificazione

Una volta integrate tutte le feature rilevanti, si è proceduto con la preparazione dei dati per la fase di classificazione. In particolare, sono state definite le variabili indipendenti X (costituite dalle feature selezionate) e la variabile target y (che rappresenta la classe PvE/PvP). Il dataset è stato suddiviso in set di training e test (80% e 20% rispettivamente) utilizzando una stratificazione per preservare la distribuzione delle classi.

Successivamente, è stato addestrato un modello di **Random Forest** con 200 alberi ($\texttt{n\_estimators} = 200$) utilizzando il set di training. Il modello è stato poi testato sul set di test e le performance sono state valutate tramite l'accuratezza, la matrice di confusione e il report di classificazione.

L'output ottenuto è il seguente:

```
Accuracy: 0.9851485148514851

Confusion Matrix:
 [[141   0]
 [  3  58]]
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       141
           1       1.00      0.95      0.97        61

    accuracy                           0.99       202
   macro avg       0.99      0.98      0.98       202
weighted avg       0.99      0.99      0.99       202
```

Da questo output si evince che il modello ha raggiunto un'accuratezza molto elevata ($\approx 98.5\%$), confermando l'efficacia della Random Forest nel distinguere tra Pokémon adatti al PvE e al PvP. La matrice di confusione mostra che su 202 campioni di test, 141 Pokémon sono stati correttamente classificati come PvE e 58 come PvP, con soli 3 errori (probabilmente casi borderline). Il report di classificazione evidenzia alti valori di precisione, **recall** e **F1-score** per entrambe le classi, attestando la robustezza del modello.

3.4 Analisi delle Feature Importanti e delle Predizioni

Successivamente, per completare l'analisi del modello di classificazione, è stata effettuata una valutazione delle feature più importanti, al fine di identificare quali variabili abbiano il maggior impatto nella predizione della classe PvE/PvP.

L'output ottenuto è stato il seguente:

```
Feature Importances:
                         feature  importance
0                    base_attack    0.331750
5                         max_cp    0.176875
15      type_effectiveness_score    0.114000
1                   base_defense    0.088842
9              dodge_probability    0.078276
2                   base_stamina    0.057577
14                  total_moves    0.021471
12              fast_moves_count    0.017966
8                 base_flee_rate    0.016397
10  max_pokemon_action_frequency    0.016104
6             attack_probability    0.016015
7              base_capture_rate    0.015000
13           charged_moves_count    0.014586
3                candy_required    0.013353
11  min_pokemon_action_frequency    0.013349
4                       distance    0.008440
```

Dall'output si è evinto che le feature più rilevanti per il modello sono **base_attack**, **max_cp** e **type_effectiveness_score**. Queste variabili hanno un impatto significativo nella determinazione della classe, evidenziando come le statistiche di combattimento e il punteggio di efficacia dei tipi siano fondamentali per discriminare tra Pokémon adatti al PvP e al PvE.

Infine, è stata eseguita un'analisi della distribuzione delle predizioni sul dataset di test, per verificare se il modello mantenesse una distribuzione coerente con quella del dataset originale.

Dato il seguente output:

```
Distribuzione delle predizioni:
0    0.712871
1    0.287129
Name: proportion, dtype: float64
```

Da questo risultato si è osservato che circa il 71.3% dei campioni è stato classificato come PvE e il 28.7% come PvP, una distribuzione che rispecchia in maniera coerente la composizione originale del dataset. Questo conferma l'affidabilità del modello nel mantenere la proporzione delle classi, rendendo l'approccio proposto efficace sia in termini di accuratezza che di capacità predittiva.

Questi risultati congiunti, relativi all'importanza delle feature e alla distribuzione delle predizioni, offrono una visione completa del funzionamento del modello di classificazione, fornendo spunti utili per ulteriori sviluppi e ottimizzazioni nel contesto della strategia di gioco e dell'analisi delle squadre di Pokémon.

Time series

4. Time series

Questa analisi esamina il dataset relativo ai crimini registrati a Los Angeles nel periodo compreso tra il 2020 e il 2024 (disponibile al seguente indirizzo:`https://www.kaggle.com/datasets/ishajangir/crime-data/data`*), con l'obiettivo di individuare tendenze significative.*

Attraverso un'analisi approfondita dei dati, sono state esplorate la distribuzione geografica dei crimini e la loro evoluzione nel tempo e le categorie di reato più frequenti. Inoltre, è stato analizzato il ruolo di variabili come l'orario e il giorno della settimana in cui si verificano i reati.

4.1 Analisi del dataset

Le colonne contenute del dataset sono le seguenti:

- **DR_NO**: Numero univoco di riferimento per il crimine riportato.
- **Date Rptd**: La data in cui il crimine è stato segnalato.
- **DATE OCC**: La data in cui l'incidente è accaduto.
- **TIME OCC**: L'orario in cui l'incidente è avvenuto.
- **AREA**: Codice identificativo dell'area in cui si è verificato il crimine.
- **AREA NAME**: Nome dell'area in cui si è verificato il crimine.
- **Rpt Dist No**: Numero di distretto di riferimento per il rapporto del crimine.
- **Part 1-2**: Indica se il crimine appartiene alla categoria di crimine Part 1 o Part 2 (classificazione dei crimini).
- **Crm Cd**: Codice del crimine associato all'incidente.
- **Crm Cd Desc**: Descrizione del crimine.
- **Mocodes**: Codici dei modelli del crimine, utilizzati per classificare i crimini.
- **Vict Age**: Età della vittima al momento dell'incidente.
- **Vict Sex**: Sesso della vittima.
- **Vict Descent**: Discendenza o etnia della vittima.
- **Premis Cd**: Codice del luogo in cui si è verificato il crimine (ad esempio, residenza, strada, etc.).
- **Premis Desc**: Descrizione del tipo di luogo dove è avvenuto il crimine.
- **Weapon Used Cd**: Codice che identifica il tipo di arma utilizzata nell'incidente (se applicabile).
- **Weapon Desc**: Descrizione dell'arma utilizzata nel crimine.
- **Status**: Stato del crimine (ad esempio, risolto, aperto, in corso).
- **Status Desc**: Descrizione dello stato del crimine.
- **Crm Cd 1**: Codice del crimine principale associato all'incidente.
- **Crm Cd 2**: Codice del secondo crimine associato, se presente.
- **Crm Cd 3**: Codice del terzo crimine associato, se presente.
- **Crm Cd 4**: Codice del quarto crimine associato, se presente.

- **LOCATION**: Indica l'indirizzo o la descrizione generale del luogo dove si è verificato l'incidente.
- **Cross Street**: Incrocio più vicino al luogo in cui è avvenuto il crimine.
- **LAT**: Latitudine della posizione geografica dell'incidente.
- **LON**: Longitudine della posizione geografica dell'incidente.

Non sono state utilizzate tutte le colonne per analizzare il dataset, ma c'è stata una scelta sulle colonne da utilizzare in base all'analisi che sono state svolte.

4.1.1 ETL

Per l'analisi del dataset è stato necessario prima importarlo attraverso il seguente codice:

```
1     file = 'Crime_Data_from_2020_to_Present.csv' crime = pd.
   read_csv(file)
2     print(crime)
```

Una volta importato, è stato utilizzato `print(crime)` per visualizzare il file *.csv* e verificare che fosse stato importato correttamente.

L'analisi dei dati è iniziata con la verifica della presenza di valori nulli nelle diverse tabelle del dataset, al fine di identificare eventuali incongruenze o dati mancanti che potrebbero influenzare le analisi successive. Il codice usato è stato il seguente:

```
1     crime. isnull().sum()
2     crime.dropna()
```

Successivamente, è stata effettuata la trasformazione dei tipi di dato delle varie colonne, garantendo uniformità nei formati e consentendo una gestione più coerente delle informazioni.

Infine, è stata eseguita la visualizzazione della tipologia di dato associata a ciascuna colonna, fornendo una panoramica chiara della struttura del dataset e facilitando le operazioni di analisi e manipolazione dei dati.

4.1.2 Visualizzazione del dataset

Dopo una prima analisi quantitativa e qualitativa dei dati presenti nel dataset, è stata avviata un'analisi più approfondita dei crimini in diverse situazioni.

In particolare, l'attenzione è stata rivolta alla distribuzione degli eventi in base al quartiere, al sesso della vittima e alle diverse tipologie di reati.

Il grafico 4.1 mostra la distribuzione del numero di crimini per quartiere a Los Angeles. La rappresentazione utilizza un grafico a barre per indicare la quantità di crimini registrati in ciascun quartiere, mentre una linea con marker sovrapposta evidenzia l'andamento della distribuzione tra le diverse aree.

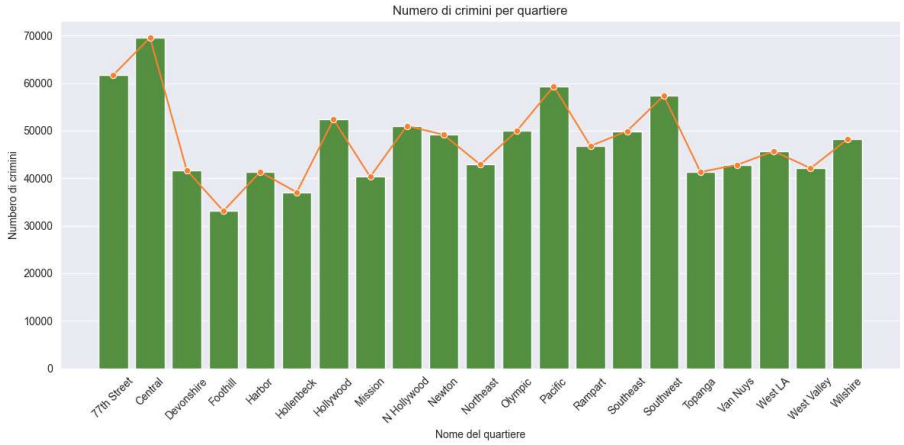
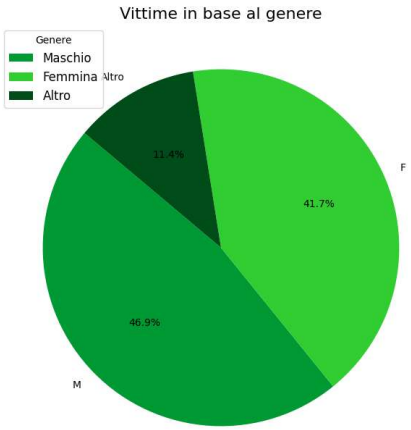


Figura 4.1: Crimini distribuiti per area



Il grafico 4.2 è un diagramma a torta che rappresenta la distribuzione delle vittime di crimini in base al genere. Le diverse sezioni del grafico mostrano la percentuale di vittime di sesso maschile, femminile e di genere non specificato (indicato con "X").

Figura 4.2: Vittime in base al genere

Il grafico 4.3 rappresenta la distribuzione dei 15 tipi di crimine più frequenti nel dataset. Nel dettaglio, l'asse orizzontale (X) mostra le categorie di crimine, mentre l'asse verticale (Y) indica il numero di occorrenze di ciascun tipo.

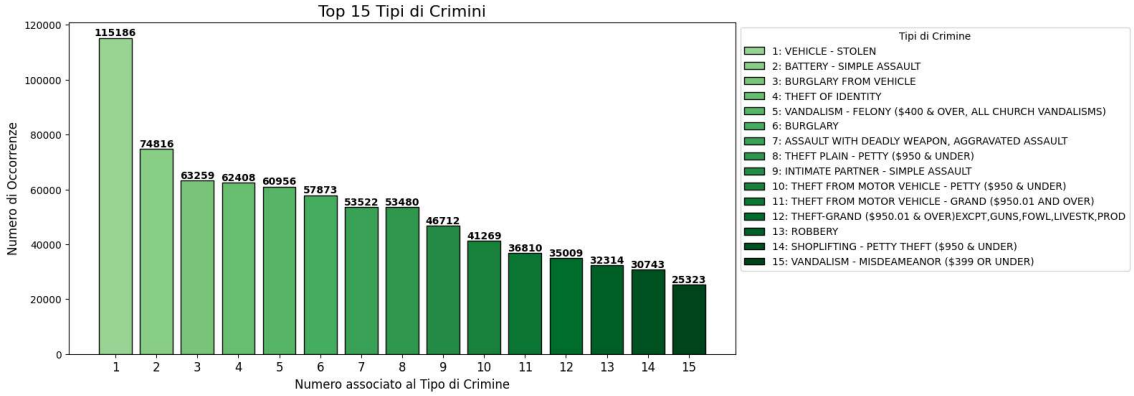


Figura 4.3: Crimini in base alla categoria

Il grafico 4.4 mostra la distribuzione geografica dei crimini a Los Angeles, rappresentando

le loro posizioni in base alle coordinate di latitudine e longitudine.

Per garantire un'analisi più accurata ed evitare distorsioni dovute a valori anomali, i dati sono stati filtrati utilizzando il metodo dei quartili. In altre parole, sono stati esclusi i valori estremi della latitudine che si discostano significativamente dalla distribuzione centrale, migliorando così la qualità della visualizzazione.

Nel caso in cui, dopo il filtro, siano ancora disponibili dati validi, il grafico a dispersione visualizza ogni crimine come un punto colorato.

L'asse orizzontale rappresenta la latitudine, mentre l'asse verticale indica la longitudine. Una legenda laterale aiuta a identificare la corrispondenza tra i colori e i diversi quartieri.

Se invece il filtro elimina tutti i dati disponibili, viene restituito un messaggio che segnala l'assenza di dati validi, suggerendo un'ulteriore verifica delle coordinate registrate nel dataset.

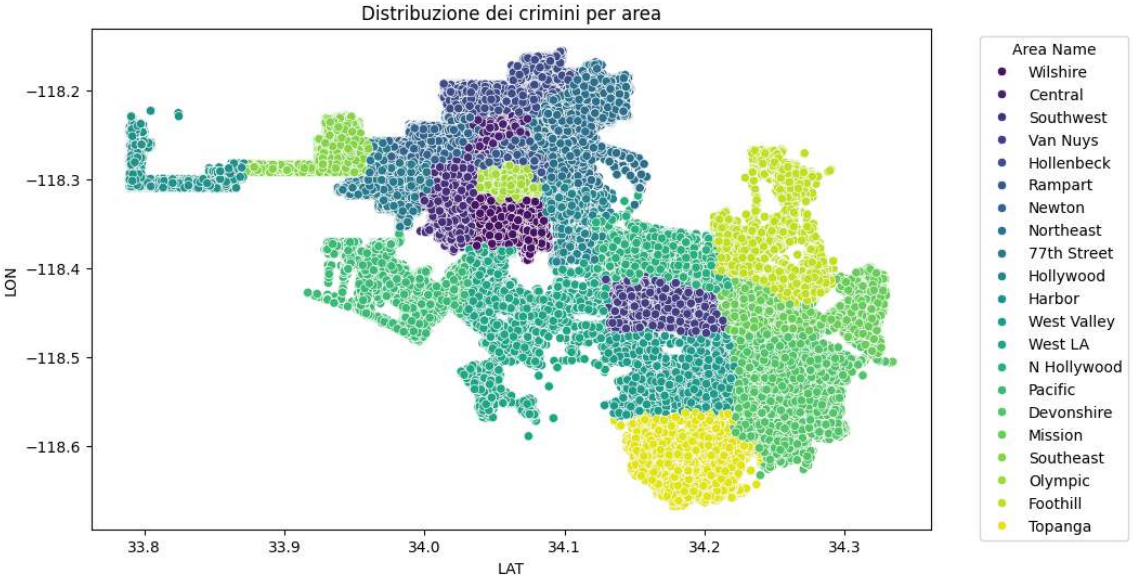


Figura 4.4: Distribuzione dei crimini sulla mappa

4.2 Analisi serie temporale

Una volta completate le analisi geografiche e sulle caratteristiche del dataset, è stata analizzata l'evoluzione temporale dei crimini: il numero di crimini per ciascun giorno è stato conteggiato utilizzando la funzione `value_counts()` per ottenere il numero di crimini per ogni data, seguita dal metodo `sort_index()` per ordinare i risultati cronologicamente.

Infine, è stato creato un grafico a linea (Figura 4.5) per visualizzare l'andamento dei crimini nel tempo. Sull'asse delle `X` sono state posizionate le date, mentre sull'asse delle `Y` il numero di crimini registrati. La linea rende evidente l'andamento giornaliero dei crimini, con i punti marcati da un cerchio che evidenziano ogni valore specifico.

! Come si evince dalla Figura 4.5, il numero di crimini registrati nel 2024 risulta inferiore rispetto agli anni precedenti. Tuttavia, non è possibile determinare con certezza se tale diminuzione sia effettivamente dovuta a una riduzione dell'attività criminale, evento piuttosto raro, oppure a una diminuzione nel numero di crimini effettivamente registrati nel dataset. Per approfondire questa osservazione, si è proceduto con un'analisi della serie temporale, applicando la stazionarizzazione del dataset (nella Sezione 4.3) e conducendo un'ulteriore valutazione escludendo i dati relativi all'anno 2024 (nella Sezione 4.4).

Il grafico 4.6 mostra la distribuzione dei crimini nell'arco della giornata, suddividendo le occorrenze in quattro fasce orarie:

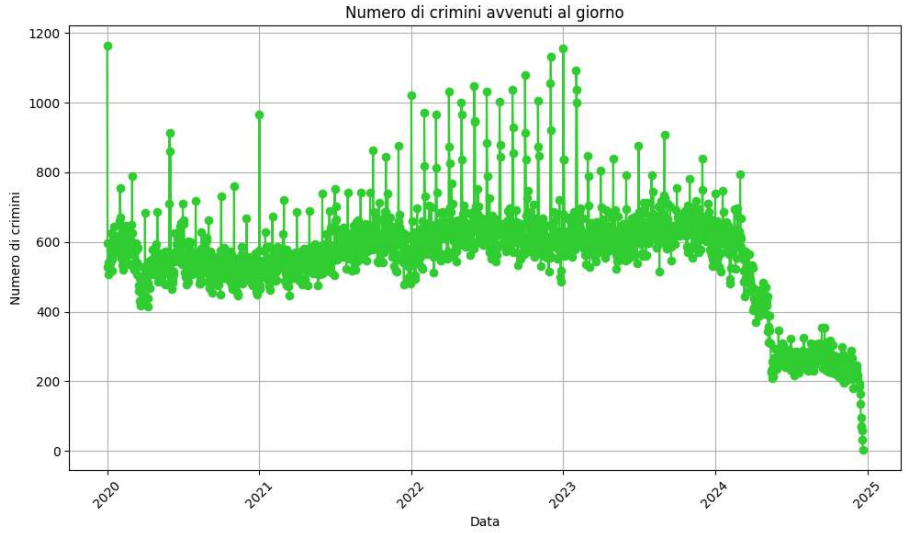


Figura 4.5: Crimini giornalieri

- Mattinata
- Pomeriggio
- Serata
- Notte

L'asse delle ascisse rappresenta le diverse fasce orarie della giornata, mentre l'asse delle ordinate indica il numero di crimini registrati in ciascun intervallo di tempo.

Questo consente di identificare rapidamente i momenti della giornata in cui i crimini sono più frequenti: per esempio, secondo la seguente analisi, è possibile notare una frequenza di crimini maggiori nel pomeriggio.

Il grafico 4.7 che si genera mostra il trend dei crimini annuali nel corso degli anni. L'asse `X` rappresenta gli anni, mentre l'asse `Y` mostra il numero totale di crimini registrati per ciascun anno. Ogni punto della linea rappresenta il numero di crimini in un anno specifico, e la linea collega questi punti per evidenziare l'andamento nel tempo.

Questo grafico permette di osservare come la criminalità si sia evoluta nel periodo analizzato, mostrando chiaramente eventuali aumenti, diminuzioni o fluttuazioni nel numero di crimini. La visualizzazione di questo trend temporale aiuta a identificare pattern di lunga durata.

Successivamente, il grafico 4.8 evidenzia il confronto stagionale della criminalità per anno. Sull'asse `X` sono rappresentati i mesi dell'anno, numerati da 1 a 12, mentre sull'asse `Y` viene indicato il numero totale di crimini registrati in ciascun mese. La linea rappresenta il numero di crimini in un mese specifico, e ciascun anno è rappresentato con una linea di un colore diverso, permettendo di confrontare l'andamento stagionale della criminalità tra gli anni.

Grazie a questo grafico, è possibile osservare come la criminalità vari durante l'anno per ciascun anno specifico.

4.3 Analisi sulle previsioni

Dopo aver esaminato i dati sia dal punto di vista temporale che spaziale, è stata iniziata un'analisi predittiva per elaborare previsioni partendo dall'esame del grafico 4.9. Quest'ultimo

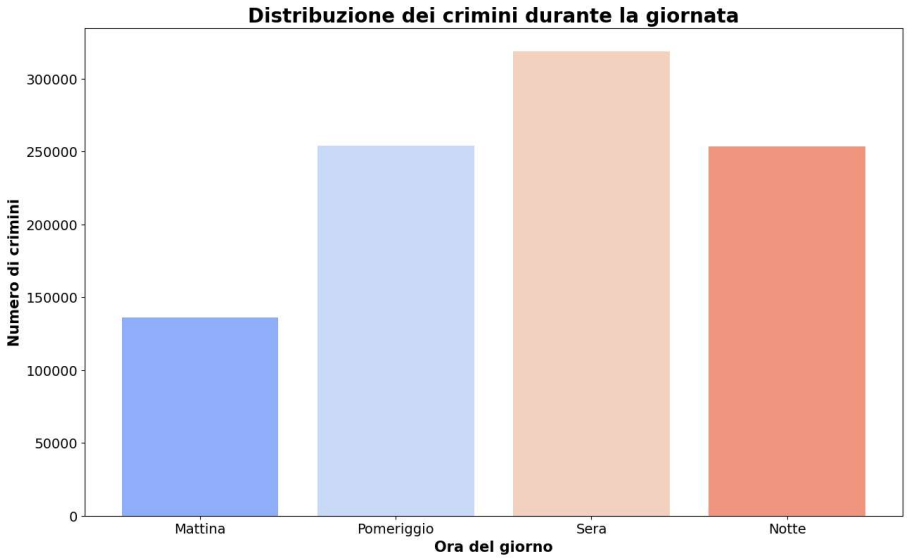


Figura 4.6: Distribuzione dei crimini durante la giornata

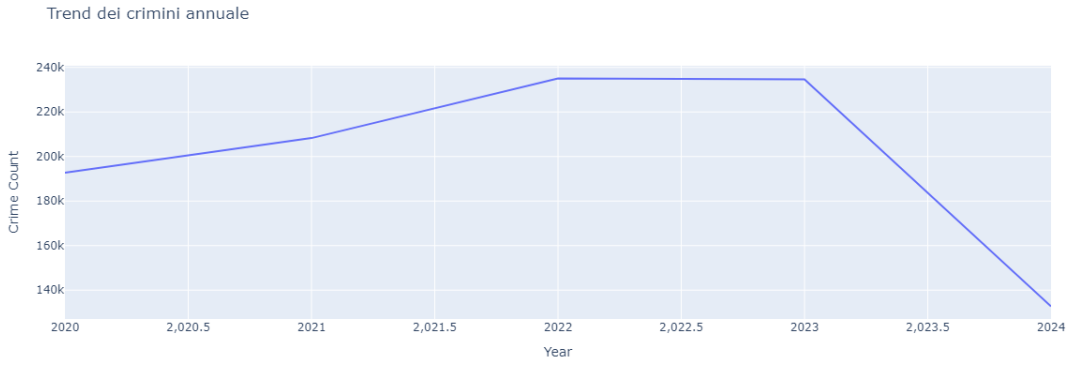


Figura 4.7: Trend dei crimini annuale

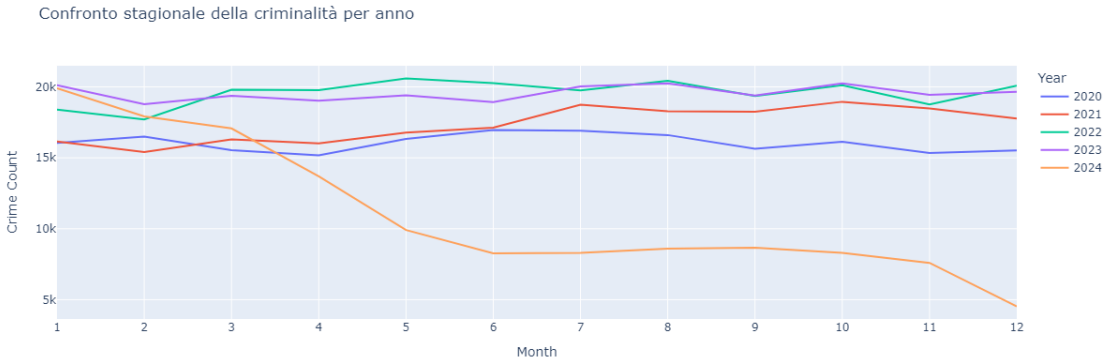


Figura 4.8: Confronto stagionale della criminalità per anno

infatti rappresenta la serie temporale dei crimini detrended, ossia una versione della serie originaria a cui è stato rimosso il trend per garantire una maggiore stabilità nel tempo.

Per realizzare tale grafico, viene applicato il test di Dickey-Fuller aumentato (ADF) per valutare la stazionarietà della serie temporale, ovvero se la media e la varianza rimangono costanti nel tempo.

Se il *p-value* del test risulta superiore a 0.05, la serie viene considerata non stazionaria e, di conseguenza, è necessario applicare delle trasformazioni.

Per eliminare il trend, viene calcolata una media mobile con una finestra di sette giorni, corrispondente a un'intera settimana. Questa media mobile rappresenta la componente di trend della serie.

Successivamente, il trend stimato viene sottratto dalla serie originale per ottenere una nuova serie più stabile nel tempo, ma che mantiene il livello medio dei dati originali.

Il grafico risultante mostra questa serie *detrended*, nella quale l'andamento generale è stato reso più uniforme, evidenziando meglio eventuali variazioni stagionali o anomalie.

Dopo la rimozione del trend, il test di Dickey-Fuller viene nuovamente applicato per verificare se la nuova serie possa essere considerata stazionaria. Se il valore di *p-value* risulta inferiore a 0.05, significa che la trasformazione ha reso i dati adatti per eventuali analisi predittive, come l'applicazione di modelli ARIMA, che richiedono una serie stazionaria per fornire previsioni affidabili.

Verifica della stazionarietà:

```
Statistiche ADF: 0.11051571951615193
P-value: 0.9667717605849461
Numero di lags usati: 25
Numero di osservazioni: 1791
Valori critici:
        1%: -3.434
        5%: -2.863
        10%: -2.568
```

Dopo aver reso la serie stazionaria:

```
Statistiche ADF (Serie Detrended): -12.89207456452746
P-value (Serie Detrended): 4.426937985861253e-24
La serie detrended è stazionaria. Il valore di d è 1.
```

Fatto questo, si è proceduto alla creazione dei grafici ACF e PACF per l'identificazione dei parametri `q` e `p`. L'ACF è utile per determinare il parametro `q` del modello ARIMA, che rappresenta l'ordine della componente di Media Mobile (MA).

La funzione di autocorrelazione parziale (PACF) misura la correlazione tra un'osservazione e un suo ritardo, escludendo l'influenza dei ritardi intermedi. Questo consente di individuare le correlazioni dirette senza interferenze da parte di altre osservazioni passate.

Il grafico 4.10, relativo alla funzione di autocorrelazione (ACF), mostra il grado di correlazione tra un'osservazione della serie temporale e le sue osservazioni precedenti a diversi ritardi (*lag*).

Il grafico 4.11, relativo alla PACF, è utile per determinare il parametro `p` del modello ARIMA, che rappresenta l'ordine della componente autoregressiva (AR).

Una volta scelti i valori p e q, entrambi impostati a 3, è stato costruito il modello ARIMA (Figura 4.12) per prevedere il numero di crimini giornalieri.

Dopo aver identificato i parametri p, d e q, si è proceduto a suddividere il dataset in due parti: una per l'addestramento e l'altra per il test.

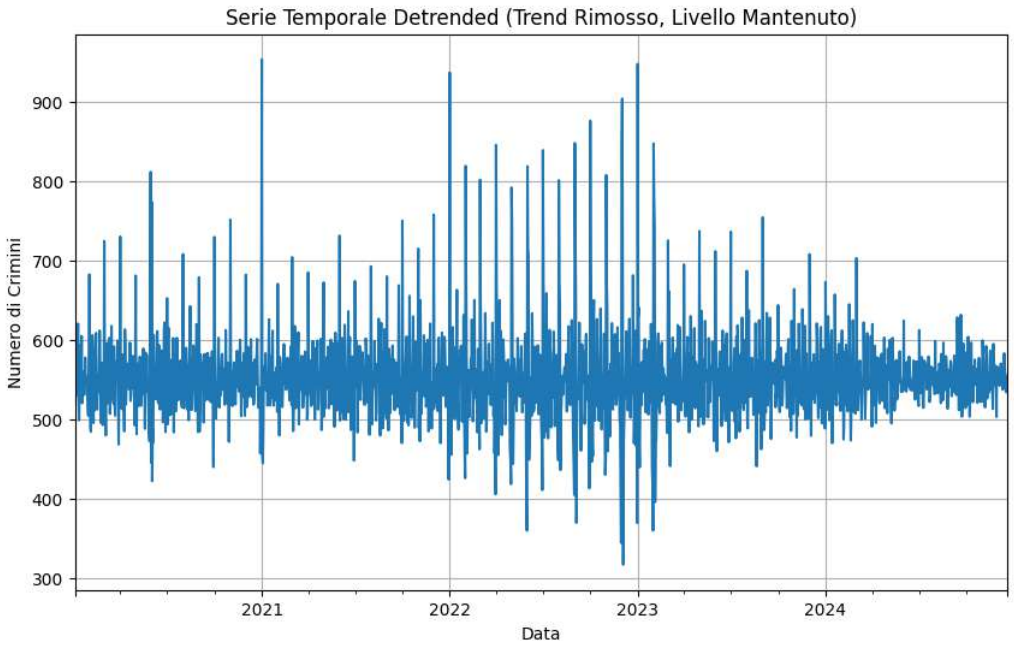


Figura 4.9: Serie temporale dei crimini detrended

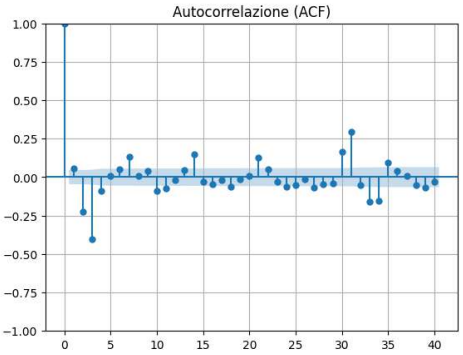


Figura 4.10: Autocorrelazione

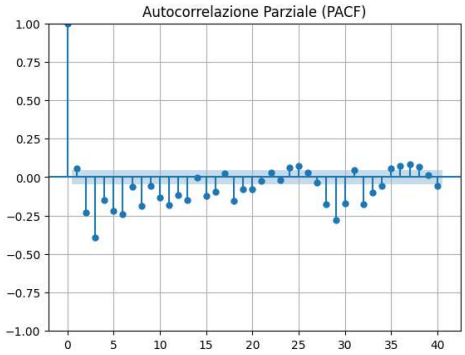


Figura 4.11: Autocorrelazione parziale

Per l'addestramento del modello, il dataset è stato suddiviso in due parti: l'80% dei dati è stato utilizzato per la fase di training, mentre il restante 20% è stato destinato alla fase di testing. Il modello ARIMA è stato addestrato con i parametri $p = 3$, $d = 1$ e $q = 3$.

```
Errore Quadratico Medio (MSE): 934.0806930443164
Media giornaliera dei crimini: 552.21
Errore Quadratico Medio (RMSE): 30.56
```

Successivamente, il modello è stato impiegato per effettuare previsioni sul periodo di test, confrontando i risultati ottenuti con i dati reali mediante un grafico. Infine, sono stati calcolati gli errori, tra cui il Mean Squared Error (MSE) e il Root Mean Squared Error (RMSE), al fine di valutare l'accuratezza delle previsioni.

L'analisi del modello, basata sull'errore quadratico medio (MSE) e sull'errore quadratico medio della radice (RMSE), evidenzia una certa precisione nelle previsioni relative ai crimini

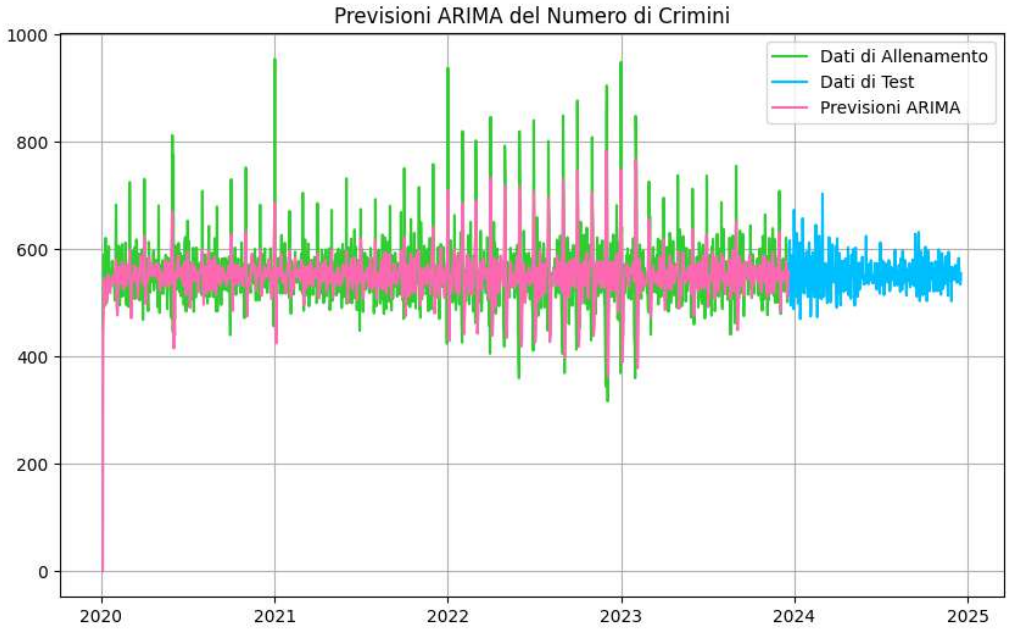


Figura 4.12: Previsioni ARIMA del Numero di Crimini

giornalieri. In particolare, l'errore quadratico medio (MSE) è pari a 934.08, un valore che indica l'entità complessiva degli errori di previsione. Sebbene tale valore possa sembrare elevato, è importante considerare che il MSE è sensibile agli outlier, ossia a previsioni significativamente distanti dai valori reali.

La media giornaliera dei crimini, che si attesta a 552.21, fornisce un riferimento utile per valutare l'accuratezza del modello. In relazione a questa media, l'MSE suggerisce che il modello, pur presentando degli errori, si mantiene in un intervallo di valori ragionevole. Inoltre, l'errore quadratico medio della radice (RMSE), pari a 30.56, indica che la previsione si discosta in media di circa 31 crimini dalla media giornaliera. Questo errore, pur non trascurabile, può essere considerato accettabile in contesti simili, pur lasciando spazio a eventuali miglioramenti nella precisione delle previsioni. In generale, i risultati suggeriscono che il modello è sufficientemente preciso

L'analisi è proseguita con lo studio della stagionalità (Figura 4.13) sia in modalità additiva che moltiplicativa, evidenziando una chiara stagionalità annuale. Tuttavia, a causa delle limitazioni nelle capacità computazionali, si è deciso di utilizzare un periodo stagionale di $m = 90$ giorni nel parametro `seasonal_order()`.

4.3.1 ARIMA

Per approfondire ulteriormente l'analisi dei crimini e delle relative previsioni, è stato deciso di concentrarsi su un singolo anno, il 2022, in quanto rappresenta l'anno con il maggior numero di crimini.

Nel corso dell'analisi del 2022, è stato impiegato il modello ARIMA. Il dataset è stato suddiviso in due parti: l'80% dei dati è stato utilizzato per il training del modello, mentre il restante 20% è stato riservato per il test, sul quale è stata successivamente effettuata la previsione.

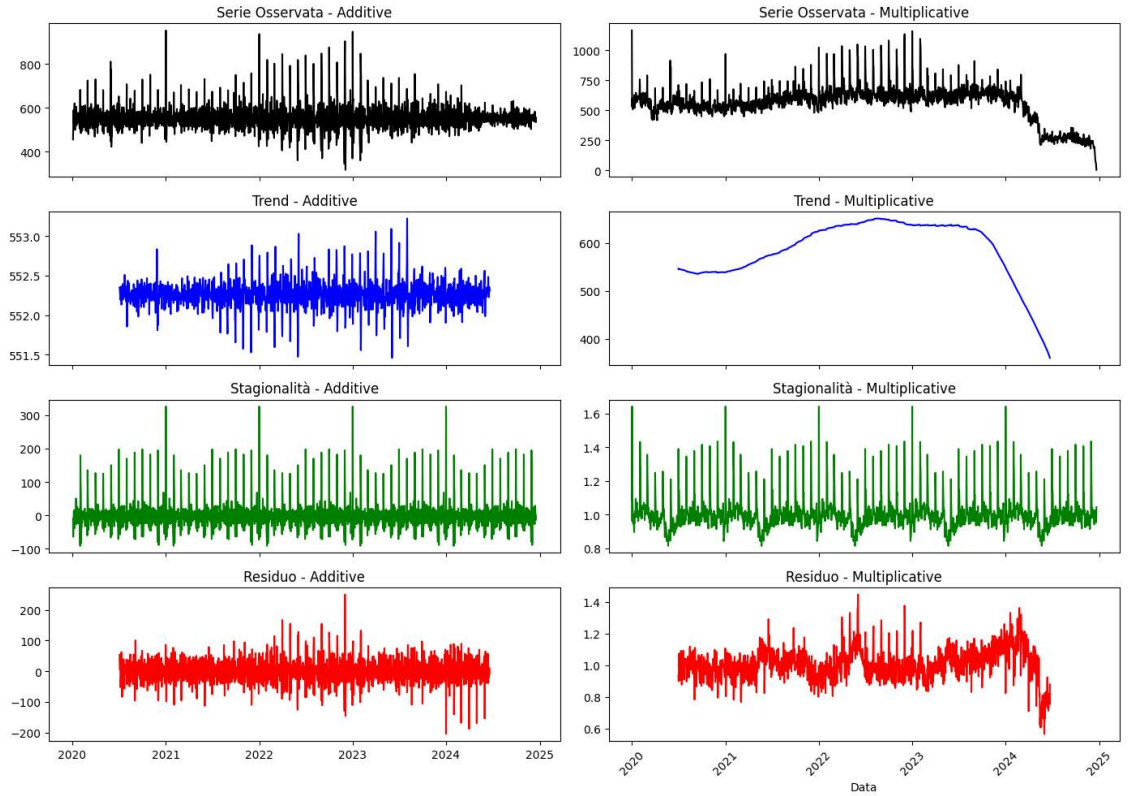


Figura 4.13: Stagionalità

Il grafico 4.14 rappresenta la previsione del numero di crimini nel periodo di test, suddiviso in tre componenti principali. La serie dei dati di allenamento che rappresenta il numero di crimini registrati nel periodo utilizzato per addestrare il modello ARIMA, evidenziando l'andamento storico della variabile. La serie dei dati di test mostra i crimini reali nel periodo successivo alla fine dei dati di allenamento, utilizzato per verificare l'accuratezza delle previsioni. Infine, la serie delle previsioni ARIMA rappresenta i valori predetti dal modello ARIMA per il periodo di test, evidenziando la capacità del modello di adattarsi ai dati osservati.

I parametri di performance ottenuti mostrano che l'Errore Quadratico Medio (MSE) è pari a $17,028.83$, indicando una discrepanza significativa tra i valori reali e quelli previsti dal modello. La media giornaliera dei crimini è di 552.26, rappresentando il valore medio del numero di crimini registrato quotidianamente nel dataset. Inoltre, l'Errore Quadratico Medio (RMSE) è di 130.49, una misura dell'errore in termini di unità originali, che indica un errore medio di circa 130 crimini per giorno tra i dati reali e quelli previsti dal modello.

Di seguito l'output completo:

```
Errore Quadratico Medio (MSE): 17028.826738830954
Media giornaliera dei crimini: 552.26
Errore Quadratico Medio (RMSE): 130.49
```

4.3.2 SARIMAX

Per quanto riguarda SARIMAX (rappresentato in Figura 4.15), sebbene le divisioni del dataset siano state le stesse di quelle utilizzate nel modello ARIMA, i risultati ottenuti presentano delle differenze significative. Ciò è dovuto alle differenze nella struttura e nella capacità dei due

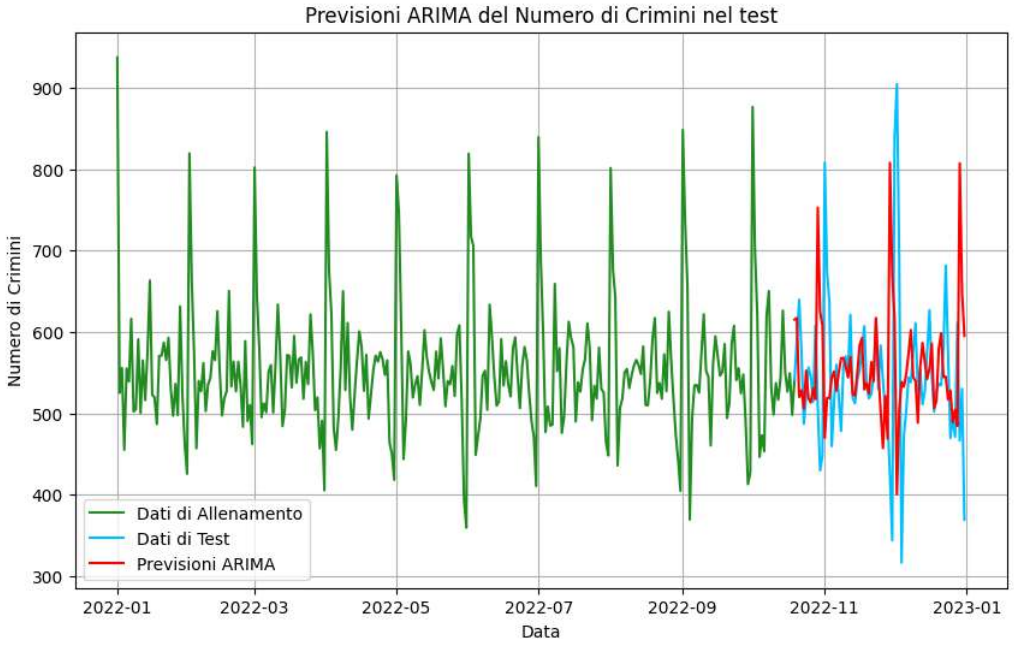


Figura 4.14: Analisi ARIMA 2022

modelli di trattare la stagionalità e altri fattori esterni, pur utilizzando parametri simili come l'ordine autoregressivo (AR), l'ordine differenziale (I) e l'ordine della media mobile (MA).

Nel caso di ARIMA, la versione stagionale SARIMA viene gestita tramite il parametro `seasonal_order`, che consente al modello di adattarsi ai cicli periodici presenti nella serie temporale. Tuttavia, ARIMA si concentra esclusivamente sulla stagionalità, senza considerare altri fattori esterni che potrebbero influenzare i dati.

Al contrario, SARIMAX estende il concetto di stagionalità, offrendo anche la possibilità di includere variabili esogene, ovvero fattori esterni che non sono direttamente legati alla stagionalità ma che possono comunque influenzare la serie temporale, come ad esempio eventi esterni. Sebbene anche SARIMAX utilizzi il parametro `seasonal_order` per gestire la stagionalità, la possibilità di includere variabili esogene rende questo modello più flessibile rispetto ad ARIMA. Di conseguenza, la presenza di questi fattori aggiuntivi può spiegare le discrepanze nei risultati tra i due modelli, anche se entrambi trattano la stagionalità in modo simile:

```
Errore Quadratico Medio (MSE): 19273.084647230207
Media giornaliera dei crimini: 552.21
Errore Quadratico Medio (RMSE): 138.83
```

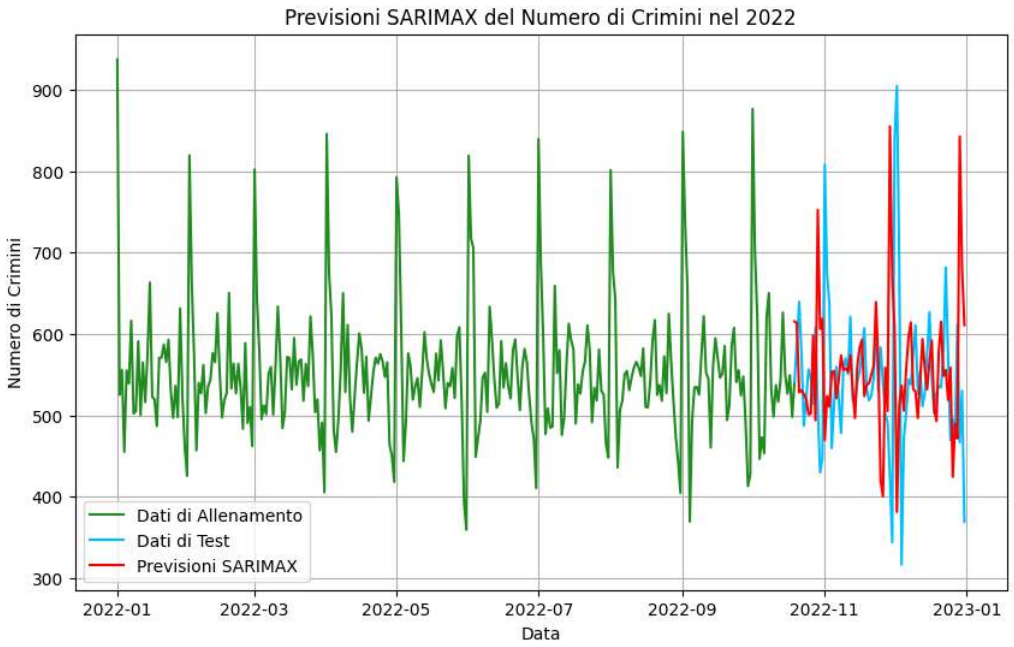


Figura 4.15: Analisi SARIMAX 2022

4.3.3 Forecast

Il grafico 4.16 visualizza le previsioni per il numero di crimini nell'arco dei successivi 30 giorni, ottenute tramite il modello ARIMA costruito e addestrato sui dati storici.

La linea verde rappresenta la serie storica dei crimini, mostrando l'andamento del numero di crimini per ciascun giorno nel dataset utilizzato per l'addestramento. La linea rossa indica le previsioni del modello, prolungando la serie storica oltre l'ultimo dato disponibile. L'area colorata in lavanda rappresenta l'intervallo di confidenza per le previsioni, indicando il range all'interno del quale ci si aspetta che i valori reali possano cadere, con un livello di confidenza specificato dal modello. Questo intervallo aiuta a comprendere l'incertezza delle previsioni, evidenziando che i valori futuri potrebbero variare, ma con una certa probabilità, all'interno di questo range.

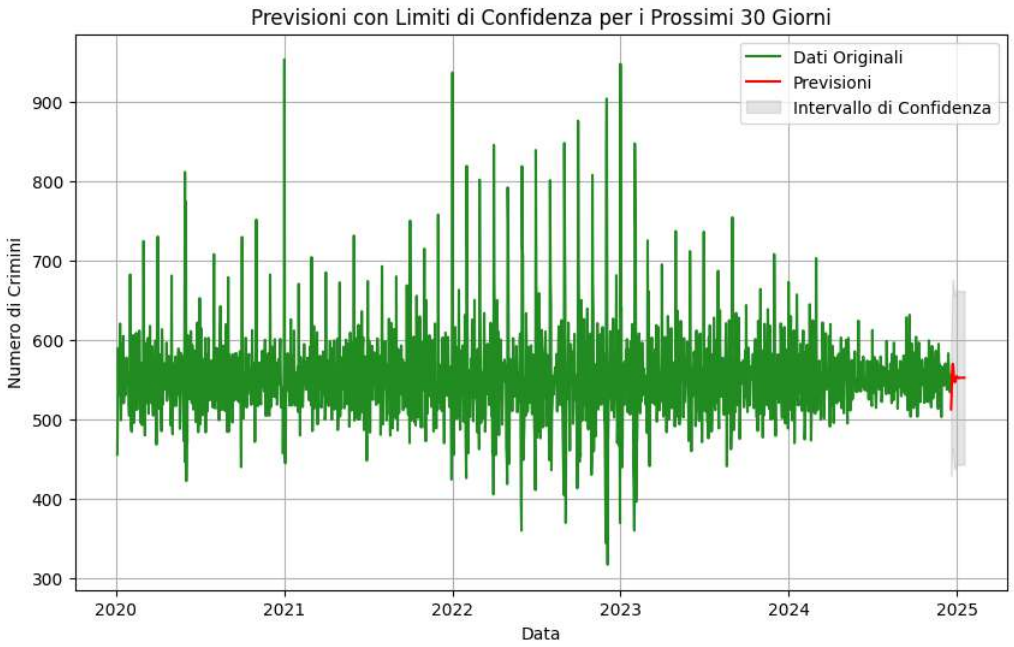


Figura 4.16: Forecast

4.4 Analisi del dataset senza il 2024

Dopo aver escluso l'anno 2024, i crimini sono stati nuovamente confrontati in base agli anni (Figura 4.17).

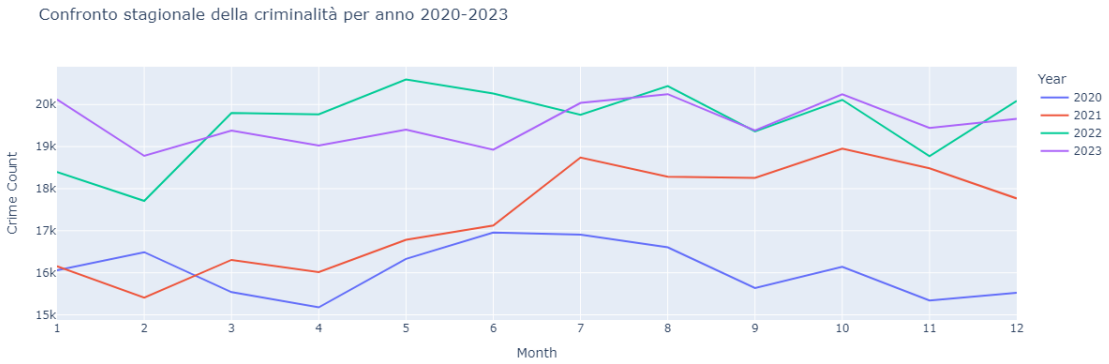


Figura 4.17: Confronto dei crimini negli anni 2020-2023

Estato dimostrato che il dataset è stazionario e, pertanto, non è stato necessario applicare una differenziazione (Figura 4.18). Successivamente, le analisi sono state ripetute sui dati ottenuti, con risultati più preformanti rispetto ai precedenti:

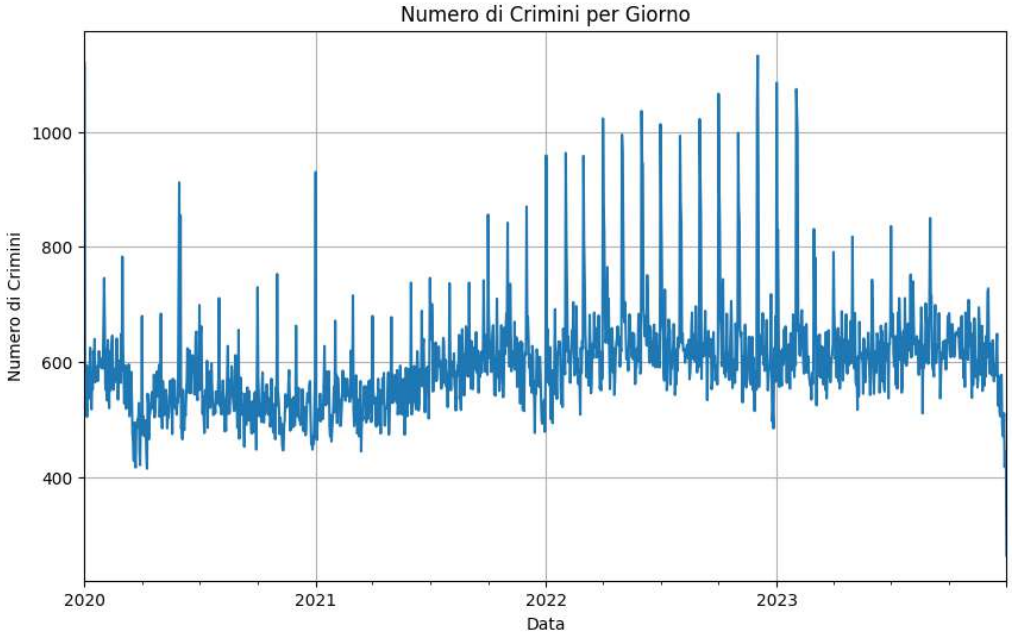


Figura 4.18: Crimini 2020-2023

```
Statistiche ADF: -3.8266508924514913
P-value: 0.0026461136388890453
Numero di lags usati: 22
Numero di osservazioni: 1438
Valori critici:
        1%: -3.435
        5%: -2.864
        10%: -2.568
La serie è già stazionaria.
Il valore di d è 0.
```

Infine, è possibile osservare la previsione per l'anno 2024 nella Figura 4.19.

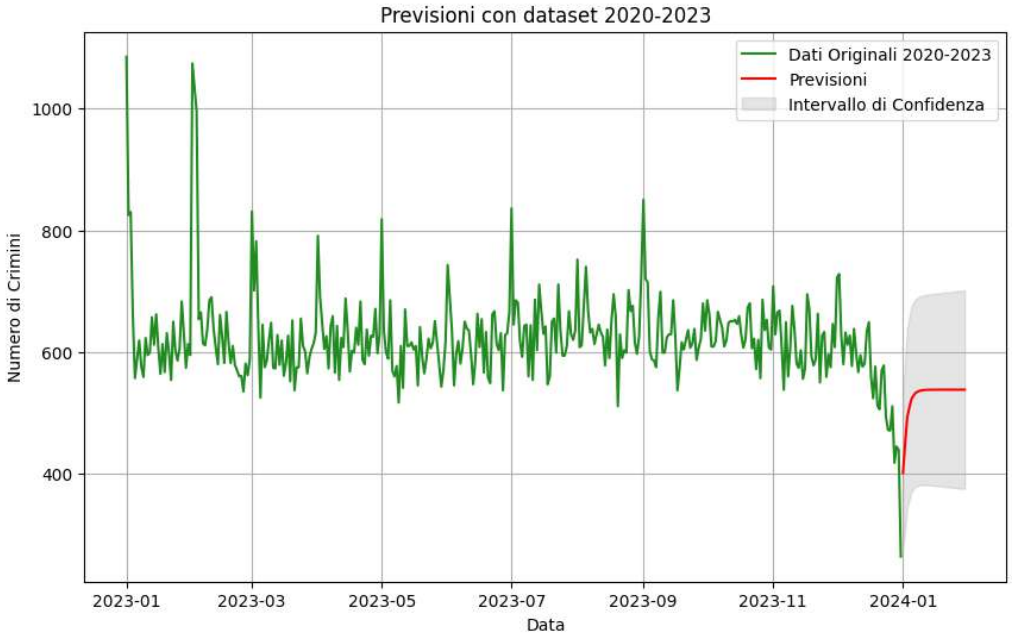


Figura 4.19: Forecast