UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

DOCUMENTAZIONE PROGETTO ICON

A.A. 2024/25

Nome: Davide Grazioso

Matricola: 758355

Mail: d.grazioso@studenti.uniba.it

Link repository: https://github.com/davidegrazioso/ProgettoIcon

Nome progetto: PokéJob

Introduzione

Il progetto PokéJob propone di classificare, consigliare, confrontare e ottimizzare i Pokémon (o in alcuni casi un team di Pokémon) utilizzando tecniche di machine learning, di clustering e risolvendo un CSP. Ci sono 3 obbiettivi principali:

1) Creare un Pokémon e classificarlo come Pokémon leggendario o meno
2) Raccomandare all'utente Pokémon in base alle caratteristiche inserite da tastiera
3) Predire il risultato di una battaglia fra 2 Pokémon preesistenti nel set
4) Ottimizzare un team di Pokémon in base ai vincoli inseriti dall'utente

I dataset utilizzati sono disponibili a questo link:
https://www.kaggle.com/datasets/abcsds/Pokémon

Uno (Pokémon.csv) ha al suo interno i Pokémon (dalla prima alla sesta generazione) con le loro caratteristiche, identificati dal loro numero del pokedex mentre l'altro (combats.csv) ha al suo interno alcuni combattimenti fra Pokémon presenti nel primo dataset con i loro rispettivi vincitori (non sono presenti scontri fra tutti i Pokémon del set "Pokémon.csv"

Per quanto riguarda i punti 1 e 3 verranno testate diverse tecniche di apprendimento supervisionato per risolvere problemi di classficazione (1) e regressione (2) (K-Nearest Neighbors, Random Forest, Gaussian Naive Bayes, Decision Tree e Support vector machine; di questi algoritmi verrà utilizzata la variante "regressor", ove disponibile, per risolvere il problema di regressione), in base a chi tra queste mostrerà una maggiore accuratezza verrà decisa la tecnica da utlizzare.

Per quanto riguarda il punto 2, invece, dopo un'attenta EDA (exploratory data analysis) verrà determinato il numero ideale di cluster da utilizzare. Nell'EDA viene usato il metodo del gomito insieme al Silhouette Score per capire il numero ideale di cluser, che verranno poi utilizzati per raccomandare il numero ideale di Pokémon.

Per quanto riguarda il punto 4 verrà risolto un constraint satisfaction problem in modo da consigliare all'utente il team di Pokémon ottimizzato in base ai vincoli imposti

Sommario

Ho deciso di dividere il progetto in 6 moduli: Preprocessing, analisi, classificazione, raccomandazione, regressione e ottizzazione. Ogni modulo ha una sua importanza per quanto riguarda il progetto nella sua completezza.

Argomenti di interesse

- **Preprocessing dei dati:** Questo è il primo modulo, la correttezza di questo modulo gioca un ruolo cruciale in quato si andrà a "pulire" il dataset in modo che i moduli successivi trovino un dataset più "pulito" e pronto all'uso, in questa fase vengono gestiti i valori mancanti, la trasformazione delle variabili e il merging di 2 dataset

- **Analisi dei dati:** L'analisi dei dati si concentra sull'esplorazione del dataset, al fine di comprendere le caratteristiche principali, identificare eventuali pattern e anomalie, e supportare le fasi successive di modellazione e predizione. L'EDA è fondamentale per:

    1. Capire come sono distribuite le variabili
    2. Identificare correlazioni e outlier (valori anomali che potrebbero influire in maniera negativa quando si vanno a costruire modelli).
    3. Fornire basi per la selezione delle variabili e la progettazione del modello predittivo.
    4. Processo di Analisi Esplorativa
    5. Visualizzazione grafica dei dati

- **Classificazione dei Pokémon leggendari:** Il modulo di classificazione è incentrato sulla classificazione di un Pokémon, in particolare andare a vedere tramite le sue statistiche di base, la generazione in cui è collocato e il suo tipo se questo Pokémon è leggendario o meno. Vengono implementati e addestrati diversi modelli (K-Nearest Neighbors, Random Forest, Gaussian Naive Bayes, Decision Tree e Support vector machine) in modo da poterli valutare tutti insieme e decidere di conseguenza quale andare ad utilizzare secondo accuracy, precision, recall e F1-score

- **Raccomandazione di 5 Pokémon:** Il modulo raccomandazione è incentrato sulla raccomandazione di 5 pokémon in base alle caratteristiche inserite dall'utente. Il modello si basa sull'algoritmo del K-means e quindi sul clustering. L'indicazione del numero ideale di cluster viene analizzata nella fase di analisi tramite metodo del gomito e Silhouette Score

- **Predizione risultato di una battaglia:** Il modulo regressione è incentrato sulla predizione dell'esito della battaglia fra 2 Pokémon, l'utente da in input 2 Pokémon da far scontrare tra di loro e come output avrà il vincitore dello scontro, per predire il

risultato vengono implementati e addestrati diversi modelli (K-Nearest Neighbors regressor, Random Forest regressor, Decision Tree regressor e Support vector machine regressor) in modo da poterli valutare tutti insieme e decidere di conseguenza quale andare ad utilizzare secondo Precision Learn, Precision Validation, Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE)

- **Costruzione di un team ottimizzato:** Questo modulo è incentrato sulla costruzione di un team ottimizzato e quindi la risoluzione di un constraint satisfaction problem. Questo team deve essere costruito in base a vari vincoli; i vincoli possono essere di due tipi: dettati dall'utente oppure imposti di default.

Preprocessing dei dati

Sommario

Il preprocessing dei dati si divide in 2 macro aree, la prima consiste nell'andare a lavorare sul database Pokémon.csv in modo da andare a eliminare colonne superflue e fare l'encoding di alcuni dati in modo da avere un database più chiaro e meno grezzo. La secondo macroarea è invece basata sul merging del dataset visto prima (Pokémon.csv) e il dataset combat.csv. il quale contiene dati su vari combattimenti fra Pokémon (Attenzione, non tutti Pokémon presenti nel set Pokémon.csv si sono scontrati fra loro)

Strumenti utlizzati:

- **Pandas**: utilizzata per visualizzazione e manipolazione dei dati da vari database

Decisioni di progretto

Sono state fatte diverse decisioni di progretto durante la realizzazione:

- **Selezione delle colonne:** è stata eliminata la colonna "Type 2" dal database Pokémon.csv in quanto non ritenuta utile per nessuno scopo e, inoltre, questa colonna presentava diversi valori nulli che avrebbero potuto portare a disagi. Per tutti questi motivi si è decisa l'eliminazione

- **Encoding:** I dati presenti nella collonna "Legendary" sono stati convertiti in valori numerici, adesso i valori sono 0 e 1, rispettivamente "false" e "true"

- **Calcolo percentuali vittoria:** Si è deciso di andare a fare un lavoro sul database "combats.csv", per ogni Pokémon presente nel dataset si è andato a vedere quante volte compariva come primo o secondo, in seguito è stata calcolata la percentuale di vittoria per ogni Pokémon.

- **Merging:** Si è deciso di andare ad unire i 2 database in un unico database che andremo a chiamare "dataset.csv", in questo dataset saranno presenti tutti i dati contenuti nei due database più tre colonne, l'utilizzo di questo set servirà nel modulo "predizione risultato di una battaglia", le colonne sono rispettivamente chiamate:

    - **NBR_COMBATS:** numero di combattimenti del Pokémon
    - **NB_VICOTRIES:** numero di vittorie del Pokémon
    - **WINNING_PERCENTAGE:** percentuale di vittoria

Si è deciso di prendere queste decisioni in modo da avere un dataset pulito e ordinato che ci sarà utile quando andremo a fare l'analisi dei dati e nei succesivi moduli.

Inizialmente i dataset si presentavano così:

Dataset Pokémon.csv (prima):

#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	Charmander	Fire		309	39	52	43	60	50	65	1	False
5	Charmeleon	Fire		405	58	64	58	80	65	80	1	False
6	Charizard	Fire	Flying	534	78	84	78	109	85	100	1	False
6	CharizardMega Charizard X	Fire	Dragon	634	78	130	111	130	85	100	1	False
6	CharizardMega Charizard Y	Fire	Flying	634	78	104	78	159	115	100	1	False
7	Squirtle	Water		314	44	48	65	50	64	43	1	False
8	Wartortle	Water		405	59	63	80	65	80	58	1	False
9	Blastoise	Water		530	79	83	100	85	105	78	1	False
9	BlastoiseMega Blastoise	Water		630	79	103	120	135	115	78	1	False
10	Caterpie	Bug		195	45	30	35	20	20	45	1	False

Dataset Pokémon.csv (dopo, succesivamente chiamato Pokémon_coded):

#	Name	Type	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
1	Bulbasaur	Grass	318	45	49	49	65	65	45	1	0
2	Ivysaur	Grass	405	60	62	63	80	80	60	1	0
3	Venusaur	Grass	525	80	82	83	100	100	80	1	0
3	VenusaurMega Venusaur	Grass	625	80	100	123	122	120	80	1	0
4	Charmander	Fire	309	39	52	43	60	50	65	1	0
5	Charmeleon	Fire	405	58	64	58	80	65	80	1	0
6	Charizard	Fire	534	78	84	78	109	85	100	1	0
6	CharizardMega Charizard X	Fire	634	78	130	111	130	85	100	1	0
6	CharizardMega Charizard Y	Fire	634	78	104	78	159	115	100	1	0
7	Squirtle	Water	314	44	48	65	50	64	43	1	0
8	Wartortle	Water	405	59	63	80	65	80	58	1	0
9	Blastoise	Water	530	79	83	100	85	105	78	1	0
9	BlastoiseMega Blastoise	Water	630	79	103	120	135	115	78	1	0
10	Caterpie	Bug	195	45	30	35	20	20	45	1	0

Dataset combats.csv:

First_pokemon	Second_pokemon	Winner
266	298	298
702	701	701
191	668	668
237	683	683
151	231	151
657	752	657
192	134	134
73	545	545
220	763	763
302	31	31
442	130	130
701	624	701

Dataset dopo il merge (dataset.csv):

	#	Name	Type	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary	First_pokemon	Second_pokemon	NBR_COMBATS
0	1	Bulbasaur	Grass	318	45	49	49	65	65	45	1	0	37.0	37.0	133.0
1	2	Ivysaur	Grass	405	60	62	63	80	80	60	1	0	46.0	46.0	121.0
2	3	Venusaur	Grass	525	80	82	83	100	100	80	1	0	89.0	89.0	132.0
3	3	VenusaurMega Venusaur	Grass	625	80	100	123	122	120	80	1	0	89.0	89.0	132.0
4	4	Charmander	Fire	309	39	52	43	60	50	65	1	0	70.0	70.0	125.0
5	5	Charmeleon	Fire	405	58	64	58	80	65	80	1	0	55.0	55.0	112.0
6	6	Charizard	Fire	534	78	84	78	109	85	100	1	0	64.0	64.0	118.0
7	6	CharizardMega Charizard X	Fire	634	78	130	111	130	85	100	1	0	64.0	64.0	118.0
8	6	CharizardMega Charizard Y	Fire	634	78	104	78	159	115	100	1	0	64.0	64.0	118.0
9	7	Squirtle	Water	314	44	48	65	50	64	43	1	0	115.0	115.0	133.0
10	8	Wartortle	Water	405	59	63	80	65	80	58	1	0	119.0	119.0	139.0
11	9	Blastoise	Water	530	79	83	100	85	105	78	1	0	114.0	114.0	135.0

Type	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary	First_pokemon	Second_pokemon	NBR_COMBATS	NB_VICTOIRES	WINNING_PERCENTAGE
Grass	318	45	49	49	65	65	45	1	0	37.0	37.0	133.0	37.0	0.2781954887218045
Grass	405	60	62	63	80	80	60	1	0	46.0	46.0	121.0	46.0	0.38016528925619836
Grass	525	80	82	83	100	100	80	1	0	89.0	89.0	132.0	89.0	0.6742424242424242
Grass	625	80	100	123	122	120	80	1	0	89.0	89.0	132.0	89.0	0.6742424242424242
Fire	309	39	52	43	60	50	65	1	0	70.0	70.0	125.0	70.0	0.56
Fire	405	58	64	58	80	65	80	1	0	55.0	55.0	112.0	55.0	0.49107142857142855
Fire	534	78	84	78	109	85	100	1	0	64.0	64.0	118.0	64.0	0.5423728813559322
Fire	634	78	130	111	130	85	100	1	0	64.0	64.0	118.0	64.0	0.5423728813559322
Fire	634	78	104	78	159	115	100	1	0	64.0	64.0	118.0	64.0	0.5423728813559322
Water	314	44	48	65	50	64	43	1	0	115.0	115.0	133.0	115.0	0.8646616541353384
Water	405	59	63	80	65	80	58	1	0	119.0	119.0	139.0	119.0	0.8561151079136691
Water	530	79	83	100	85	105	78	1	0	114.0	114.0	135.0	114.0	0.8444444444444444

Valutazione:

Per quanto concerne la valutazioe a noi basta sapere quanti valori mancanti ci sono e se il merge sia andato a buon fine, il resto è ininfluente o comunque molto poco influente per i nostri scopi e per l'analisi, è stata creata una funzione apposita che controlla che non ci siano valori mancanti nel nuovo dataset e che il merge sia andato a buon fine:

```
Preprocessing completato. Dataset salvato in 'datasets/dataset.csv'.

--- Analisi del nuovo dataset ---

Dimensioni: 800 righe, 1 colonne

Valori mancanti:
Nessun valore mancante
```

Analisi dei dati

Sommario

Si è deciso di creare il modulo analisi in modo da poter avere una panoramica molto completa del dataset analizzato nel preprocessing. Si è deciso di optare per l'EDA (Exploratory data analysis) in modo esplorare e comprendere il dataset e, grazie alle varie librerie utilizzate, riusciremo ad avere un'interfaccia grafica

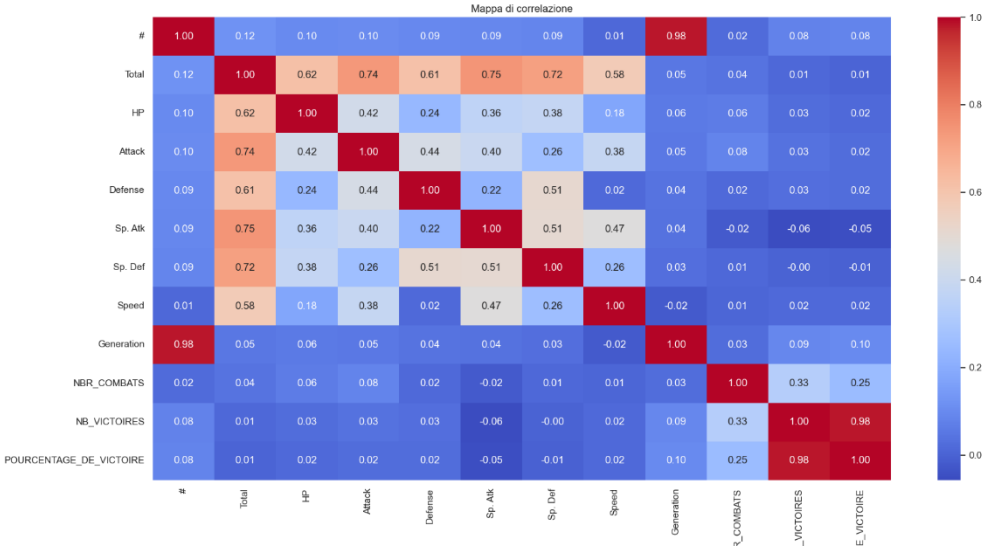
Strumenti utilizzati

- **Matplotlib:** Matplotlib è una libreria per la creazione di grafici
- **Pandas:** collegamento alla sezione precedente
- **Seaborn:** Seaborn è una libreria di visualizzazione dati basata su Matplotlib. Fornisce un'interfaccia di alto livello per disegnare grafici statistici accattivanti e informativi.

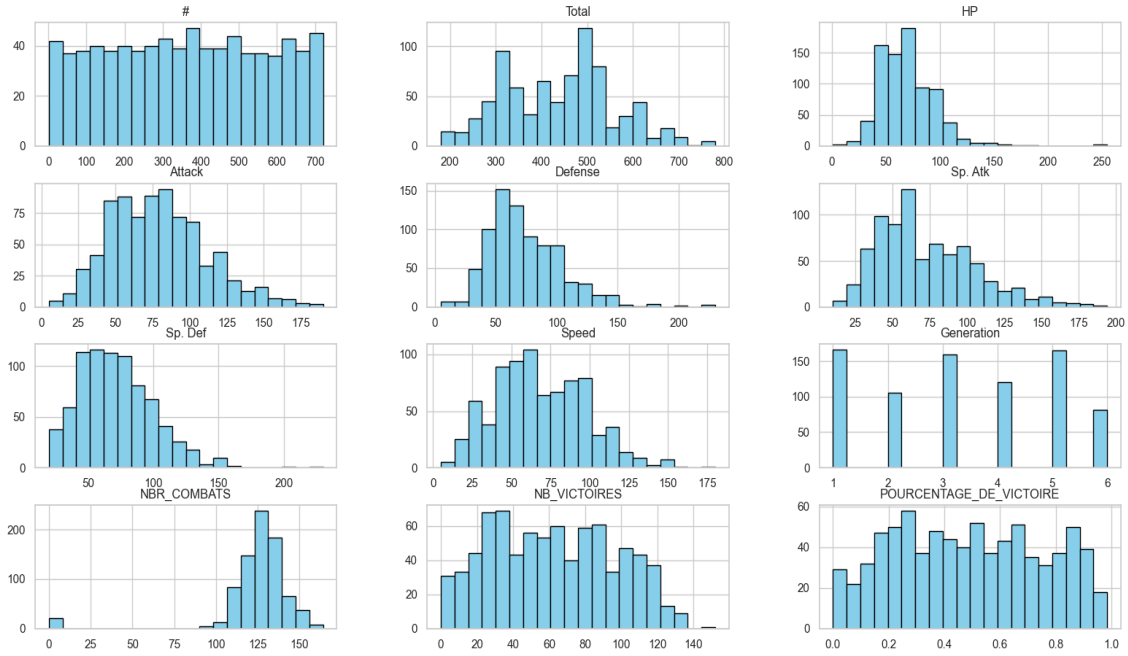
Decisioni di progetto

Si è deciso di fare un'EDA per capire al meglio il dataset nella sua completezza

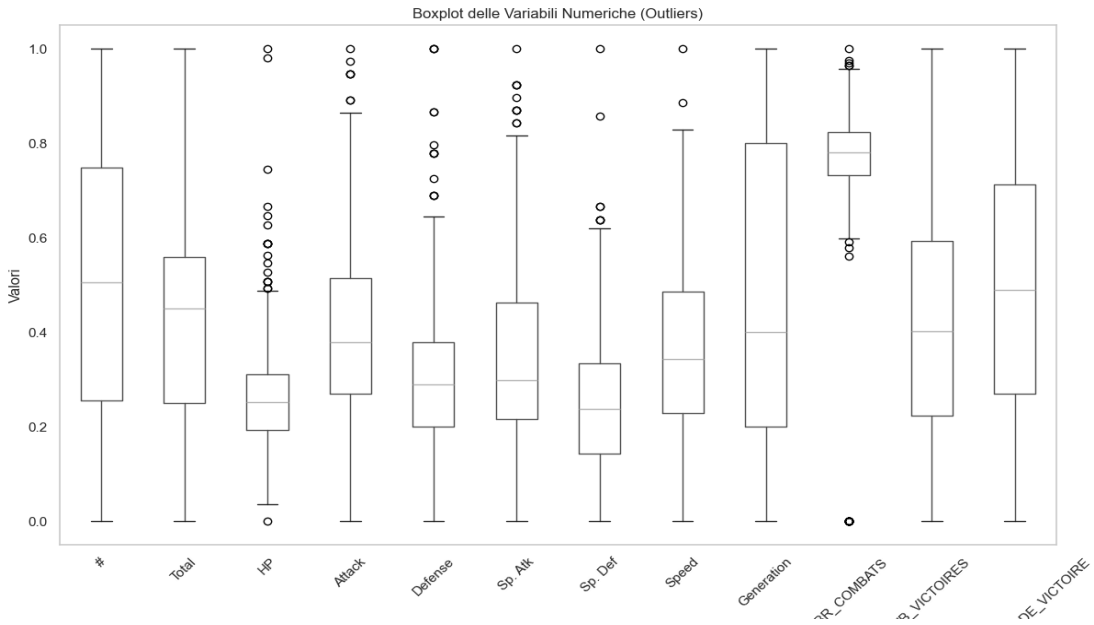
- **Heatmap della matrice delle correlazioni**: indica il grado di correlazione tra 2 variabili, i risultati sono compresi fra -1 e 1, un risultato uguale a 1 indica che la variabili sono proporzionali fra di loro (quando una aumenta, l'altra aumenta), un risultato uguale a -1 indica che la variabili sono inversamente proporzionali fra di loro (quando una diminuisce, l'altra diminuisce), un risultato uguale a 0 inidica la totale assenza di correlazione fra 2 variabili



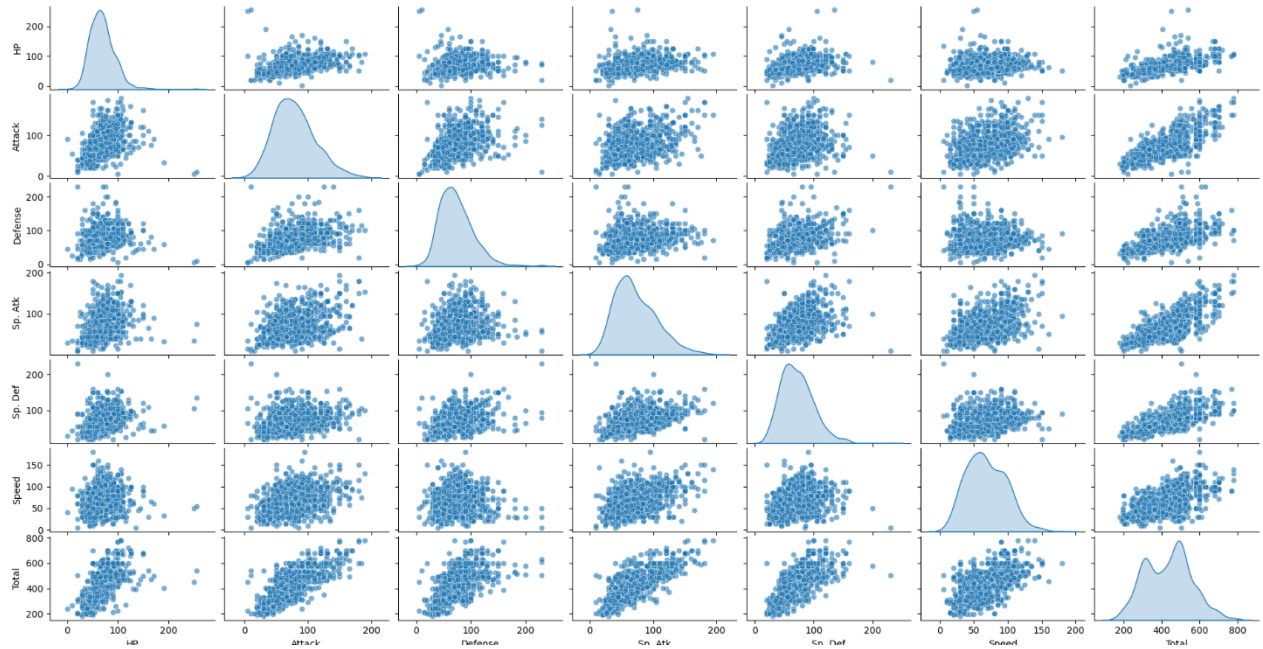
- **Distribuzione delle variabili numeriche:** indica in che modo le variabili sono distribuite nel dataset e la loro variabilità:

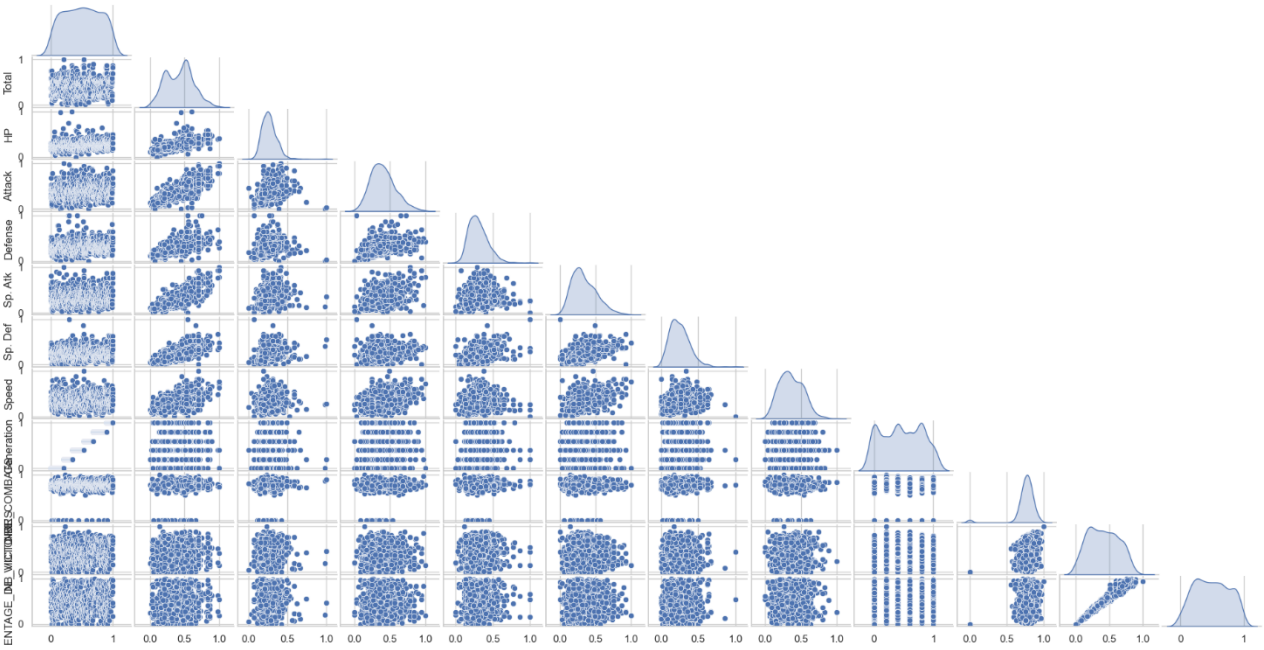


- **Identificazione Outliers:** Gli outlier sono un aspetto importante dell'analisi dei dati. È fondamentale identificarli e valutarne l'impatto sulle analisi statistiche.

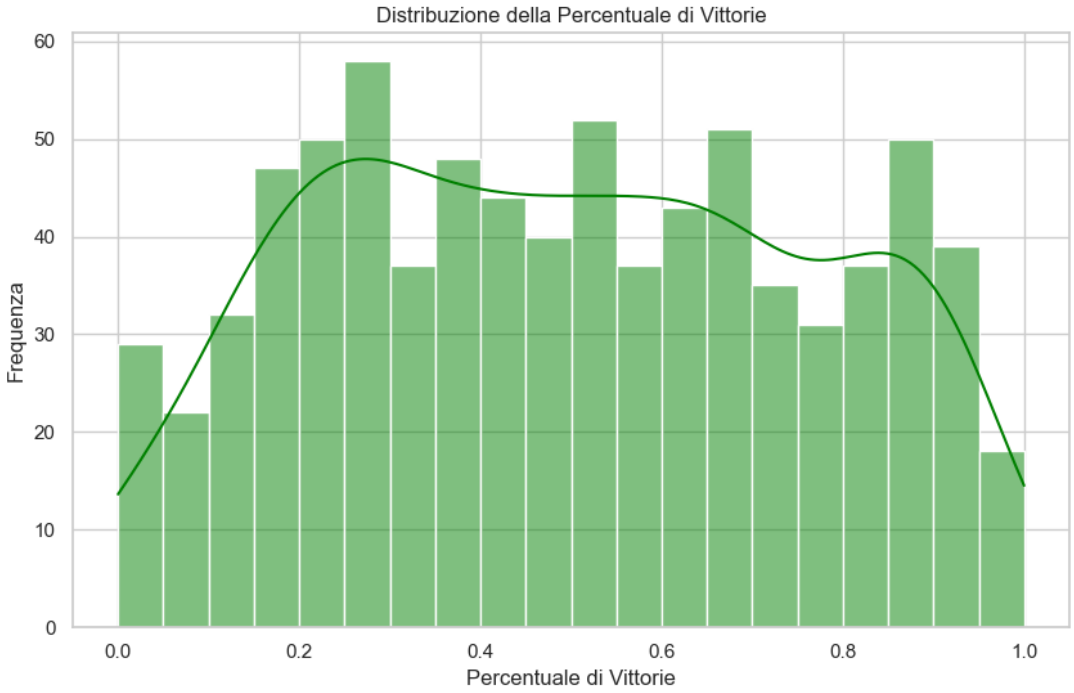


- **Relazioni fra variabili numeriche:** Uno degli aspetti chiave dell'EDA è l'analisi delle relazioni tra le variabili. Quando si tratta di variabili numeriche, queste relazioni possono essere di grande interesse per individuare pattern, trend e correlazioni.

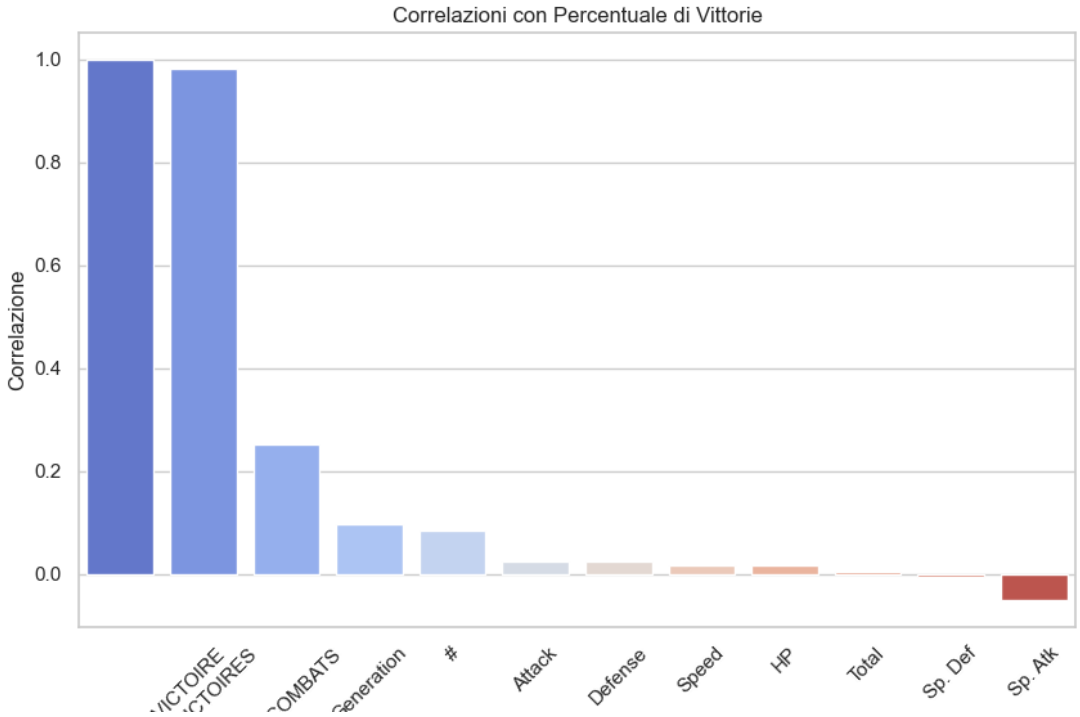




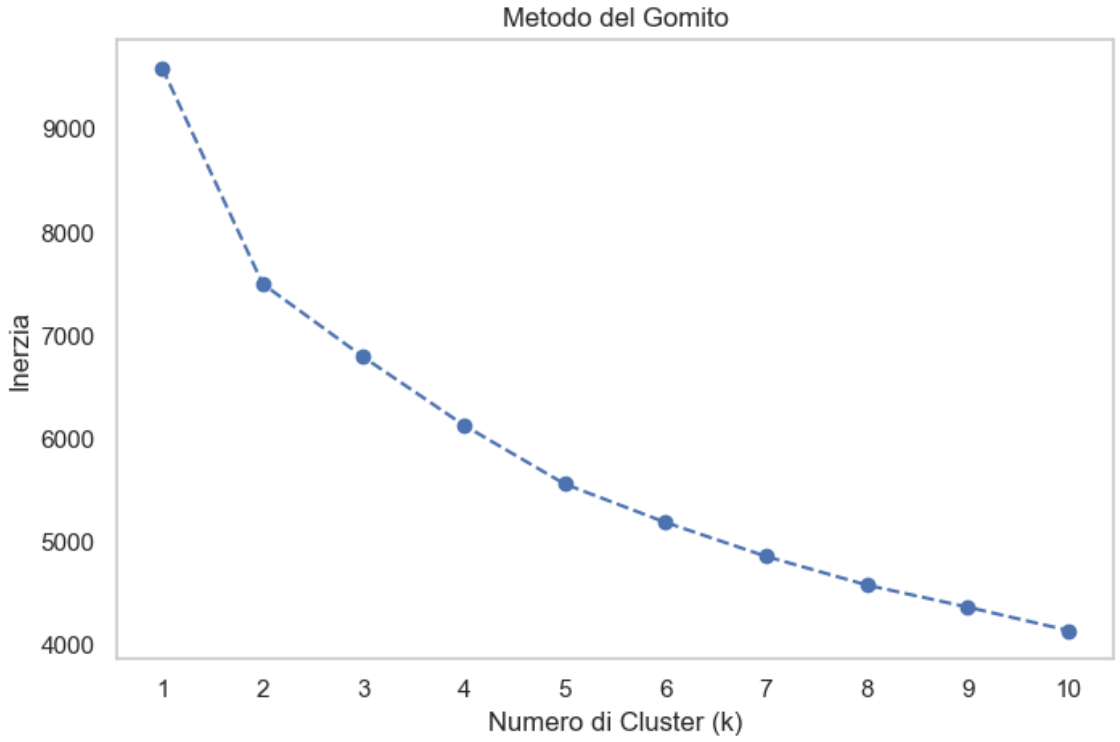
- **Analisi variabile target:** La variabile target, a volte chiamata variabile dipendente o output, rappresenta ciò che vogliamo prevedere o spiegare attraverso il nostro modello

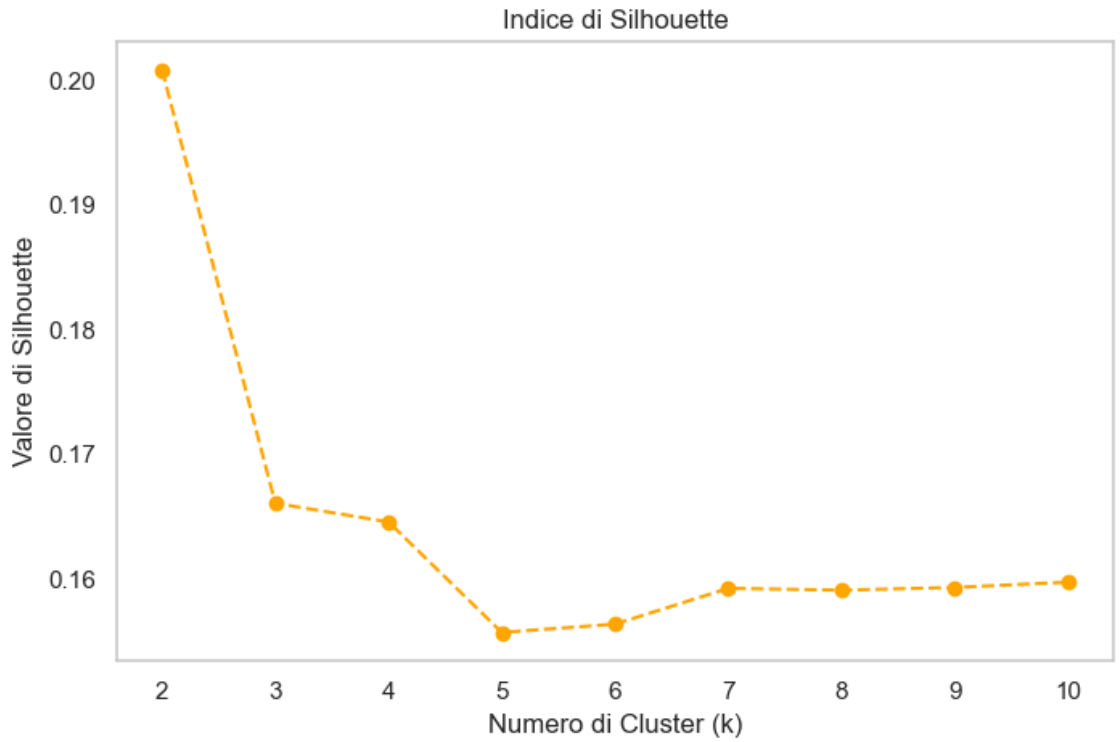


- **Analisi delle Correlazioni tra la Variabile Target e le Altre Variabili:** Questa analisi ci permette di identificare quali variabili sono più influenti sulla variabile target e, di conseguenza, di selezionare le caratteristiche più rilevanti per il nostro modello



- **Metodo del gomito e silhouette score:** Il metodo del gomito e lo silhouette score sono due tecniche molto utilizzate nell'analisi dei cluster per determinare il numero ottimale di cluster in un dataset. Quando si applica un algoritmo di clustering, come K-means, è fondamentale scegliere il numero corretto di cluster per ottenere una suddivisione dei dati significativa e utile.





Valutazione:

L'analisi dei dati attraverso questo processo offre utili intuizioni per lo sviluppo di modelli predittivi, facilitando:

- **L'identificazione di variabili chiave e input significativi** per i modelli di classificazione. La rimozione di variabili irrilevanti e la selezione di dati rilevanti consentono di focalizzarsi sulle variabili che realmente influenzano il modello, migliorandone l'efficacia complessiva.
- **L'ottimizzazione del pre-processing per la gestione e la pulizia dei dati**, inclusa la gestione dei valori mancanti e degli outlier. Queste operazioni garantiscono che i dati siano preparati in modo ottimale per la modellazione, migliorando la coerenza e la qualità dei dati stessi.
- **Una comprensione più profonda dei dati** L'EDA permette di ridurre al minimo il rischio di errori durante la fase di modellazione predittiva e di migliorare le prestazioni dei classificatori, ottimizzando le operazioni di pre-processing e aumentando l'affidabilità del modello.

Classificazione dei Pokémon leggendari

Sommario

Questo modulo si concentra sulla predizione dello stato di un Pokémon inserito dall'utente, se leggendario o meno. All'utente verrà chiesto di inserire le statistiche di un nuovo Pokémon e, grazie a un modello di apprendimento supervisionato, verrà valutato se il nuovo Pokémon inserito sia leggendario o meno. Sono stati confrontati diversi modelli di apprendimento supervisionato (K-Nearest Neighbors, Naive Bayes, Random Forest, Decision Tree, Support vector machine) e sono stati confrontati tra di loro, si è deciso di optare per il modello con l'accuratezza maggiore

Strumenti utilizzati

Pandas: Vedi sezione precendete

Matplotlib: Vedi sezione precedente

Seaborn: Vedi sezione precendete

Scikit-Learn: Utilizzato per implentare i modelli di apprendimento supervisionato

Decisioni di progetto:

- **Target e set di addestramento:** Definita come target la colonna "Legendary". Questa colonna è stata separata dal resto del dataset e assegnata alla variabile Y, mentre tutte le altre colonne (eccetto Name, First_Pokémon, Second_Pokémon, NBR_victories, #, NB_victories) sono state utilizzate come caratteristiche di input e assegnate alla variabile X.

- **Modulazione**: Si è deciso di dividere tutto in due moduli "addestramento" e "legendary_function", nel modulo addestramento avviene l'addestramento dei modelli mentre nel modulo legendary_function viene definita la funzione che permette all'utente di inserire da tastiera i dati e di ricevere output)

- **Modelli utlizzati:** Sono stati implementati diversi modelli di apprendimento supervisionato (K-Nearest Neighbors, Naive Bayes, Random Forest, Decision Tree, Support vector machine) e sono stati valutati per capire quale di questi si adatterebbe meglio al nostro set

    - **K-Nearest Neighbors:** L'algoritmo KNN non richiede un vero processo di addestramento come gli altri modelli di machine learning. Piuttosto, è un metodo lazy learning: Durante la fase di training, si limita a memorizzare l'intero

dataset, Quando viene richiesto di effettuare una previsione, il modello: Calcola la distanza tra l'entità da classificare (query point) e tutti i punti del dataset di riferimento, identifica i K punti più vicini al query point, Determina la categoria dell'entità basandosi sulla maggioranza delle classi tra i K vicini (nel caso della classificazione) o calcola una media dei valori dei vicini (per la regressione).

- **Gaussian Naive Bayes:** è una variante dell'algoritmo Naive Bayes, specificamente progettata per gestire dati continui assumendo che le caratteristiche seguano una distribuzione normale**.** L'algoritmo Naive Bayes si basa sul Teorema di Bayes, che afferma:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

dove:
P(C|X) è la probabilità che un'entità appartenga alla classe C, dato il vettore di caratteristiche X (probabilità a posteriori).
P(X|C) è la probabilità di osservare X dato che l'entità appartiene alla classe C (probabilità condizionata).
P(C) è la probabilità di osservare la classe C indipendentemente dai dati (probabilità prior).
P(X) è la probabilità marginale di osservare X.
L'algoritmo utilizza il principio della massima verosimiglianza per assegnare una classe: **C=argmax P(C|X)**

- **Random forest:** Il Random Forest è un algoritmo di machine learning supervisionato basato sull'approccio ensemble, particolarmente adatto a gestire problemi complessi di classificazione e regressione. In questo contesto è stato utilizzato per capire se un Pokémon è leggendario o meno grazie a caratteristiche come HP,Attack,Defense e altre statistiche chiave. Questo modello si distingue per la capacità di catturare relazioni non lineari e identificare schemi nascosti nei dati. Ogni albero viene costruito utilizzando un campionamento casuale dei dati di addestramento (metodo bootstrap). Per ogni divisione di un albero, viene scelto casualmente un sottoinsieme delle caratteristiche disponibili, garantendo diversità tra gli alberi. Per la classificazione, ogni albero "vota" per una classe, e la classe con il maggior numero di voti viene assegnata all'osservazione.

- **Decision tree**: Un Decision Tree suddivide iterativamente il dataset in base a criteri che massimizzano l'informazione ottenuta ad ogni livello. In questo caso, utilizza caratteristiche come HP, Attack, Defense, Type e altre per creare le divisioni.

- **Radice**:
  La radice dell'albero è la prima decisione, basata sulla caratteristica che meglio separa i dati. Ad esempio, può iniziare con una domanda come: "Il totale delle statistiche del Pokémon è maggiore di 600?"
- **Nodi interni**:
  Ogni nodo rappresenta una decisione successiva basata su un'altra caratteristica. Ad esempio:
  "Se il totale è maggiore di 600, il tipo è Drago o Normale?"
- **Foglie**:
  Ogni foglia rappresenta una classe predetta (leggendario o non leggendario).

il Decision Tree Classifier è stato scelto per predire se un Pokémon è leggendario sfruttando:

- **Dati Numerici**: Statistiche come HP, Attack e Speed.
- **Dati Categoriali**: Tipo (ad esempio, "Drago" o "Acqua").
- **Interazione tra Caratteristiche**: Ad esempio, la combinazione di un alto totale di statistiche e il tipo "Drago" è un indicatore chiave di Pokémon leggendari.

- **SVM**: Il Support Vector Machine (SVM) si distingue per la sua capacità di trovare confini decisionali ottimali tra classi di dati. SVM potrebbe essere utilizzato per classificare Pokémon come leggendari o non leggendari, analizzando caratteristiche come HP, Attack, Defense, e altre statistiche chiave. La forza dell'SVM risiede nella sua capacità di gestire sia problemi lineari che non lineari, grazie all'uso di trasformazioni matematiche chiamate kernel. SVM può essere utilizzato per predire se un Pokémon è leggendario sfruttando:

  - **Separazione Complessa**: Le statistiche dei Pokémon (ad esempio, Total, Attack, Defense) possono non avere confini netti tra classi. L'SVM, grazie ai kernel, può modellare confini non lineari per distinguere i leggendari dai non leggendari.
  - **Support Vector come Indicatore Chiave**: I Pokémon vicini al confine decisionale possono fornire intuizioni utili. Ad esempio, Pokémon quasi leggendari con caratteristiche simili ai leggendari.
  - **Robustezza al Rumore**: La capacità dell'SVM di concentrarsi sui support vector riduce l'impatto di Pokémon rumorosi o fuori distribuzione nel dataset.
  - **Scalabilità a Dati Categoriali e Numerici**: Combinando caratteristiche numeriche (statistiche) e categoriali (tipo, generazione) trasformate in variabili dummy, l'SVM offre una soluzione flessibile.

- **Configuarazione dei parametri**: I modelli testati includono K-Nearest Neighbors (KNN), Decision Tree, Random Forest, Gaussian Naive Bayes, Support Vector Machine (SVM). Ogni modello è stato configurato e ottimizzato utilizzando **GridSearchCV** per determinare i migliori parametri e garantire risultati ottimali.

    - **K-Nearest Neighbors (KNN)**
      Parametri Testati:
        - **n_neighbors**: Numero di vicini da considerare.
        - [5, 6, 7, 8, 9, 10]
        - **weights**: Metodo di ponderazione dei vicini.
        - ['uniform', 'distance']

      metric: Metrica utilizzata per calcolare la distanza.

        - ['euclidean', 'manhattan', 'chebyshev']

      Migliori Parametri:

        - **n_neighbors**: 6
        - **weights**: distance
        - **metric**: manhattan

      Risultati: Accuratezza media: **93.125%**

    - **Decision tree**

      Parametri Testati:

        - **criterion**: Funzione di misurazione della purezza del nodo.
        - ['gini', 'entropy']
        - **max_depth**: Profondità massima dell'albero.
        - [5, 10, 15, None]
        - **min_samples_split**: Numero minimo di campioni richiesti per dividere un nodo.
        - [2, 5, 10]
        - **min_samples_leaf**: Numero minimo di campioni per foglia.
        - [1, 2, 4]

      Migliori Parametri:

        - **criterion**: entropy
        - **max_depth**: 10
        - **min_samples_split**: 2
        - **min_samples_leaf**: 1

      Risultati: Accuratezza media: **93.75%**

- **Random forest:**

  Parametri Testati:

  - **n_estimators**: Numero di alberi nella foresta.
  - [100, 200, 300]
  - **max_depth**: Profondità massima degli alberi.
  - [10, 20, None]
  - **min_samples_split**: Numero minimo di campioni richiesti per dividere un nodo.
  - [2, 5]
  - **min_samples_leaf**: Numero minimo di campioni per foglia.
  - [1, 2]
  - **criterion**: Funzione di misurazione della purezza del nodo.
  - ['gini', 'entropy']

  Migliori Parametri:

  - **criterion**: entropy
  - **max_depth**: 10
  - **min_samples_split**: 2
  - **min_samples_leaf**: 1
  - **n_estimators**: 100

  Risultati: Accuratezza media: **93.125%**

- **Gaussian Naive Bayes**

  Parametri Testati:

  - **var_smoothing**: Parametro che aggiunge una piccola quantità di varianza per stabilizzare i calcoli.
  - np.logspace(0, -9, num=10)
  - **Migliori Parametri:**
  - **var_smoothing**: 10−610^{-6}10−6

  Risultati: Accuratezza media: **30%**

- **Support Vector Machine (SVM)**

  Parametri Testati:

  - **kernel**: Funzione del kernel.
  - ['linear', 'poly', 'rbf', 'sigmoid']
  - **C**: Penalizzazione degli errori.
  - [0.1, 1, 10, 100]
  - **gamma**: Parametro per kernel RBF e sigmoid.
  - ['scale', 'auto', 0.1, 0.01]

- **degree**: Grado del polinomio (per kernel polinomiale).
- [2, 3, 4]

Migliori Parametri:

- **kernel**: rbf
- **C**: 10
- **gamma**: scale

Risultati: Accuratezza media: **92.5%**

La scelta del miglior modello dipende dal contesto e dai dati disponibili. Per questo problema specifico, il **Decision tree** ha fornito la migliore accuratezza, con un'accuratezza media del **93.75%**. Tuttavia, ogni modello testato ha le sue qualità:

KNN: Buono per problemi di classificazione con confini lineari.

Random Forest: Buon compromesso tra accuratezza e generalizzazione.

Gaussian Naive Bayes: Rapido ma limitato a dati lineari.

SVM: Efficace per problemi con confini complessi e non lineari.

L'approccio ottimizzato tramite GridSearchCV ha permesso di identificare i migliori parametri per ogni modello, garantendo una migliore generalizzazione e prestazioni sul dataset di Pokémon.

L'accuratezza media di 93.75% indica che il modello ha classificato correttamente circa il 93.75% degli esempi di test su tutte le 15 iterazioni di cross-validation. Questo è un buon risultato, indicando che il modello sta generalizzando bene e non si adatta troppo ai dati di addestramento (evitando l'overfitting). In altre parole, il decision tree sembra essere in grado di fare previsioni molto accurate sui dati di test che non ha mai visto prima.

Valutazione

La valutazione è stata eseguita secondo le seguenti metriche:

- **Accuratezza media (Accuracy):** Indica la percentuale di esempi correttamente classificati rispetto al totale ed è valutata tramite questa formula

$$\text{Accuracy} = \frac{\text{True Positives (TP)} + \text{True Negatives (TN)}}{\text{Total Examples}}$$

- **Precisione media (Precision):** Indica la proporzione di esempi classificati come positivi che sono effettivamente positivi ed è valutata tramite questa formula:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

- **Richiamo medio (Recall):** Indica la proporzione di esempi positivi effettivi che sono stati correttamente identificati ed è valutato tramite questa formula:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

- **F1-Score medio:** È la media armonica di precisione e richiamo, bilanciando i due aspetti ed è valutato secondo questa formula:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **ROC AUC:** Indica la capacità del modello di distinguere tra classi. La curva ROC (Receiver Operating Characteristic) mostra il rapporto tra il Tasso di Veri Positivi (True Positive Rate) e il Tasso di Falsi Positivi (False Positive Rate)

Esempio di visualizzazione:

```
Decision Tree - Accuracy: 0.9437, Precision: 0.6250, Recall: 0.7692, F1 Score: 0.6897, ROC AUC: 0.8642
Random Forest - Accuracy: 0.9437, Precision: 0.6667, Recall: 0.6154, F1 Score: 0.6400, ROC AUC: 0.7941
Naive Bayes - Accuracy: 0.3000, Precision: 0.0976, Recall: 0.9231, F1 Score: 0.1765, ROC AUC: 0.5840
KNN - Accuracy: 0.9313, Precision: 1.0000, Recall: 0.1538, F1 Score: 0.2667, ROC AUC: 0.5769
SVM - Accuracy: 0.9250, Precision: 0.5714, Recall: 0.3077, F1 Score: 0.4000, ROC AUC: 0.6436
```

Esempio di utlizzo del modulo:

```
Inserisci i dettagli del Pokémon per predire se è leggendario o no:
Tipo (es. Grass, Fire, Water) o premi 0 per vedere tutti i tipi disponibili: normal
HP: 120
Attack: 120
Defense: 120
Special Attack: 120
Special Defense: 120
Speed: 120
Generation: 4

Il Pokémon è leggendario!
```

Raccomandazione di 5 Pokémon

Sommario

Il modulo suggerisce Pokémon simili in base alle statistiche fornite dall'utente, sfruttando un algoritmo di clustering per identificare somiglianze nei dati. Questo consente di ottenere suggerimenti personalizzati e pertinenti. Il sistema utilizza l'algoritmo **K-Means** per raggruppare i Pokémon in cluster, basandosi su statistiche come HP, Attack, Defense, Sp. Atk, Sp. Def, Speed e Total. Successivamente, calcola la distanza tra le statistiche normalizzate fornite dall'utente e quelle dei Pokémon all'interno del cluster corrispondente. i valori numerici delle statistiche richieste (HP, Attack, Defense, Sp. Atk, Sp. Def, Speed). Ogni valore deve essere compreso tra **1 e 255** per essere valido.
Il numero di cluster è stato ottimizzato a 4 utilizzando metodi statistici come il "gomito" e il silhouette score (analizzati nel modulo "Analisi dei dati"), garantendo un equilibrio tra accuratezza e semplicità. Dopo aver identificato il cluster di appartenenza, il sistema calcola la distanza euclidea tra le statistiche fornite dall'utente e quelle dei Pokémon nel cluster. I 5 Pokémon con la distanza più breve vengono raccomandati.

Strumenti utilizzati

- **Pandas:** vedi sezione precendente
- **Sklearn:** Utlizzato per l'implementazione di algoritmi di apprendimento non supervisionato (Kmeans)
- **Numpy:** Utilizzato per fare calcoli numerici complessi

Decisioni di progetto

- **Algoritmo di Similarità** Per calcolare la similarità tra le statistiche fornite dall'utente e i Pokémon nel dataset, è stato scelto l'algoritmo di clustering **K-Means**. Questo approccio divide i Pokémon in cluster basati sulle loro statistiche numeriche (HP, Attack, Defense, Sp. Atk, Sp. Def, Speed, Total). Successivamente, viene calcolata la distanza euclidea tra le statistiche dell'utente e quelle dei Pokémon nel cluster, per determinare i Pokémon più simili. I Pokémon con la distanza minima sono quelli raccomandati.
- **Filtraggio Dinamico delle Statistiche in Base ai Dati Utente** Il sistema è progettato per adattarsi ai dati forniti dall'utente. Se l'utente non fornisce un valore per una determinata statistica (ad esempio, HP o Attack), il sistema rimuove quella colonna dal calcolo delle distanze. Questo rende il sistema flessibile e in grado di gestire situazioni in cui alcune informazioni sono mancanti o non necessarie, migliorando la precisione e la pertinenza delle raccomandazioni.
- **Selezione dei Pokémon Raccomandati** Una volta calcolata la distanza tra le statistiche dell'utente e quelle dei Pokémon, il sistema ordina i Pokémon per similarità crescente. I primi 5 Pokémon con la distanza più bassa vengono suggeriti

all'utente. In questo modo, il sistema assicura che i suggerimenti siano i più simili possibile alle preferenze espresse dall'utente.

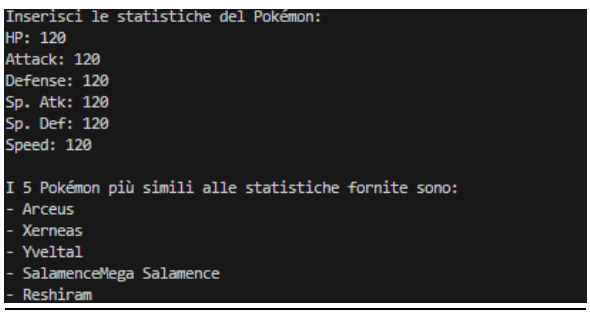
- **Clustering per Ottimizzare la Ricerca** Per migliorare l'efficienza del sistema, i Pokémon sono stati raggruppati in 4 cluster tramite l'algoritmo K-Means, scelto in base ai risultati ottenuti dal metodo del gomito e dal punteggio silhouette. Questo approccio riduce il numero di confronti da fare durante la ricerca dei Pokémon più simili, migliorando le performance del sistema.
- **Dinamicità del Sistema** Il sistema è progettato per essere dinamico e adattarsi facilmente a nuovi dati. Le scelte progettuali, come l'uso del clustering e della normalizzazione, permettono al sistema di fare raccomandazioni precise anche con input variabili. Il filtraggio delle colonne in base ai dati disponibili consente di personalizzare l'esperienza dell'utente.

Valutazione:

La precisione del sistema è stata testata attraverso vari scenari di input, confrontando le raccomandazioni con le statistiche inserite dall'utente.

- **Precisione delle Raccomandazioni** I Pokémon suggeriti si sono dimostrati altamente pertinenti e coerenti con le preferenze fornite, grazie all'uso del clustering e alla normalizzazione dei dati.
- **Feedback degli Utenti** Gli utenti che hanno testato il sistema hanno fornito feedback positivo riguardo alla coerenza dei suggerimenti. In particolare, hanno apprezzato che i Pokémon raccomandati riflettessero fedelmente le statistiche inserite e le aspettative, dimostrando un allineamento tra le raccomandazioni e le preferenze.
- **Integrazione del Clustering K-Means** L'uso dell'algoritmo di clustering **K-Means** ha contribuito significativamente alla qualità delle raccomandazioni. Raggruppando i Pokémon in cluster ottimizzati, il sistema riesce a filtrare efficacemente i Pokémon più rilevanti per l'utente, migliorando la precisione del processo decisionale.
- **Robustezza del Sistema**
  Durante la fase di testing, il sistema ha dimostrato una notevole robustezza anche con input incompleti o parzialmente errati. Il filtraggio dinamico delle statistiche assicura che il sistema si adatti in modo flessibile ai dati disponibili, mantenendo alta la qualità delle raccomandazioni.

Esempio di utilizzo



Predizione risultato di una battaglia

Sommario

Il modulo si occupa di prevedere le performance dei Pokémon durante i combattimenti, sfruttando dati statistici e storici. Attraverso tecniche di apprendimento supervisionato, il sistema analizza caratteristiche come attacco, difesa e velocità per stimare l'esito di un combattimento fra due Pokémon. I modelli sviluppati includono algoritmi di regressione (Random Forest regression, Decision Tree regression, KNN regression, SVM regression). Ne è stato scelto uno basandosi sui valori di precision learn e precision palidation. I dati elaborati alimentano una Knowledge Base (KB) in grado di supportare la predizione dei risultati dei combattimenti.

Strumenti utilizzati

- **Scikit-Learn**: Per l'implementazione di algoritmi di regressione e classificazione.
- **Pandas**: vedi sezione precendete
- **NumPy**: vedi sezione precendete
- **Joblib**: Per salvare dei modelli addestrati.

Decisioni di progetto

Durante lo sviluppo del modulo di classificazione e predizione dei combattimenti tra Pokémon, sono state prese le seguenti decisioni progettuali per garantire un sistema robusto ed efficiente:

- **Target e set di addestramento**: Il target scelto è stato il rapporto di vittorie dei Pokémon, calcolato come la percentuale di successi su tutti i combattimenti. Le feature utilizzate includono statistiche come Attacco, Difesa, Attacco Speciale, Difesa Speciale e Velocità.

- **Modulazione**: Si è deciso di dividere tutto in due moduli "battaglia" e "play", nel modulo battaglia avviene l'addestramento dei modelli mentre nel modulo play viene definita la funzione che permette all'utente di inserire da tastiera i dati e di ricevere output)

- **Modelli utlizzati:**

    - **Random forest regressor:** un meta-stimatore che adatta un certo numero di regressori di alberi decisionali su vari sottocampioni del set di dati e usa la media per migliorare l'accuratezza predittiva e controllare l'over-fitting. Gli alberi usano la migliore strategia di suddivisione, ovvero equivalente al passaggio splitter= "best" al sottostante DecisionTreeRegressor. La dimensione del sottocampione è controllata con il max_samples se bootstrap=True(predefinito), altrimenti l'intero set di dati viene usato per costruire ogni albero.
    - **Decision Tree regressor:** La funzione per misurare la qualità di una divisione. I criteri supportati sono "squared_error" per l'errore quadratico medio, che è uguale alla riduzione della varianza come criterio di selezione delle caratteristiche e riduce al minimo la perdita L2 utilizzando la media di ciascun nodo terminale, "friedman_mse", che utilizza l'errore quadratico medio con il punteggio di miglioramento di Friedman per le potenziali divisioni, "absolute_error" per l'errore assoluto medio, che riduce al minimo la perdita L1 utilizzando la mediana di ciascun nodo terminale e "poisson" che utilizza la riduzione della deviazione di Poisson della metà della media per trovare le divisioni
    - **Support Vector Machine (SVM) Regressor:** Una variante delle SVM utilizzata per problemi di regressione. L'obiettivo è trovare una funzione che si adatti ai dati entro una tolleranza (epsilon) e minimizzi l'errore.
    - **K-Nearest Neighbors (KNN) Regressor:** Basato sull'algoritmo KNN**;** L'obiettivo viene previsto tramite interpolazione locale degli obiettivi associati ai vicini più prossimi nel set di addestramento.

- **Configuarazione paramentri iniziali**: sono stati testati i 5 modelli di apprendimento supervisionato e ne è stato ricercato uno migliore con GridSearchCV, siamo partiti con i seguenti parametri iniziali:

    - **K-Nearest Neighbors Regressor (KNN)**

      Parametri Testati:

        - n_neighbors: [3, 5, 7]
        - weights: ['uniform', 'distance']
        - metric: ['euclidean', 'manhattan']

      Migliori Parametri:

        - n_neighbors: 5
        - weights: 'distance'
        - metric: 'manhattan'

Risultati:

- R^2 Training: 1 (overfitting)
- R^2 Validation: -0.2
- Mean Absolute Error (MAE): 32.2593
- Mean Squared Error (MSE): 1405.4382
- Root Mean Squared Error (RMSE): 37.4892

- **Decision Tree Regressor**

Parametri Testati:

- max_depth: [None, 10, 20]
- min_samples_split: [2, 5, 10]
- min_samples_leaf: [1, 2, 4]

Migliori Parametri:

- max_depth: 10
- min_samples_split: 2
- min_samples_leaf: 1

Risultati:

- R^2 Training: 0.40
- R^2 Validation: -0.47
- Mean Absolute Error (MAE): 32.9143
- Mean Squared Error (MSE): 1625.4678
- Root Mean Squared Error (RMSE): 40.3171

- **Random Forest Regressor**

Parametri Testati:

- n_estimators: [50, 100, 200]
- max_depth: [None, 10, 20]
- min_samples_split: [2, 5, 10]
- min_samples_leaf: [1, 2, 4**]**

Migliori Parametri:

- n_estimators: 100
- max_depth: 10
- min_samples_split: 2
- min_samples_leaf: 1

Risultati:

- R^2 Training: 0.51
- R^2 Validation: -0.1
- Mean Absolute Error (MAE): 29.6173
- Mean Squared Error (MSE): 1211.0800
- Root Mean Squared Error (RMSE): 34.8006

- **Support Vector Machine Regressor (SVMR)**

Parametri Testati:

- kernel: ['linear', 'poly', 'rbf']
- C: [0.1, 1, 10]
- epsilon: [0.01, 0.1, 1]

Migliori Parametri:

- kernel: 'rbf'
- C: 10
- epsilon: 0.1

Risultati:

- R^2 Training: 0.00
- R^2 Validation: 0.00
- Mean Absolute Error (MAE): 28.5118
- Mean Squared Error (MSE): 1113.4792
- Root Mean Squared Error (RMSE): 33.3688

In conclusione, abbiamo scelto il **Random Forest Regressor** come modello principale per diversi motivi. Innanzitutto, offre ottime prestazioni generali, combinando più alberi decisionali per ridurre l'overfitting e garantire una buona generalizzazione. Rispetto a modelli

come il **Decision Tree**, che può essere più suscettibile al rumore, o il **KNN**, che non scala bene con dataset complessi e in questo caso genera overfitting, il Random Forest si è dimostrato più robusto. Inoltre, è in grado di catturare relazioni non lineari tra le caratteristiche dei Pokémon, cosa fondamentale per comprendere interazioni complesse tra variabili come attacco, velocità e il fatto di essere leggendario. l'**SVM**, sebbene potente, è meno scalabile e richiede un tuning più sofisticato.

Il Random Forest è anche robusto ai dati rumorosi grazie al processo di aggregazione dei risultati, che rende il modello meno influenzato da outlier rispetto a modelli come il KNN o il Decision Tree. Inoltre, fornisce informazioni utili sull'importanza delle feature, aiutandoci a capire quali caratteristiche influenzano maggiormente le prestazioni.

Infine, si è dimostrato stabile nei risultati, combinando un'alta precisione con metriche di errore contenute, sia nei dati di training che di validazione. Per queste ragioni, il Random Forest rappresenta la scelta ottimale per il nostro problema.

Valutazione

La valutazione del modello è stata eseguita secondo le seguenti metriche

- **Precisione del Modello (R^2 Score)**: Il coefficiente di determinazione misura quanto bene il modello spiega la variabilità dei dati. Un valore più vicino a 1 indica una migliore capacità predittiva del modello. Viene calcolato sia per il set di training (*precision learn*) sia per quello di validazione (*precision validation*).
- **Errore Assoluto Medio (MAE)**: Indica la media degli errori assoluti tra le predizioni del modello e i valori reali. È una misura diretta della precisione delle predizioni, espressa nella stessa unità delle variabili target.
- **Errore Quadratico Medio (MSE)**: Misura l'entità media degli errori elevati al quadrato. Poiché penalizza maggiormente gli errori più grandi, è utile per evidenziare eventuali anomalie o sottostime sistematiche.
- **Radice dell'Errore Quadratico Medio (RMSE)**: È la radice quadrata dell'MSE e fornisce una stima della deviazione standard degli errori residui. È più interpretabile dell'MSE poiché mantiene l'unità delle variabili target.

Esempio di visualizzazione:

```
========== Random Forest Regressor ==========
Best Parameters: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 100}
Precision Learn: 0.5610
Precision Validation: -0.0823
Mean Absolute Error (MAE): 29.6140
Mean Squared Error (MSE): 1200.3510
Root Mean Squared Error (RMSE): 34.6461
==================================
========== Decision Tree Regressor ==========
Best Parameters: {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 10}
Precision Learn: 0.4018
Precision Validation: -0.4656
Mean Absolute Error (MAE): 32.9143
Mean Squared Error (MSE): 1625.4678
Root Mean Squared Error (RMSE): 40.3171
==================================
========== SVM Regressor ==========
Best Parameters: {'C': 0.1, 'epsilon': 0.01, 'kernel': 'rbf'}
Precision Learn: -0.0000
Precision Validation: -0.0040
Mean Absolute Error (MAE): 28.5118
Mean Squared Error (MSE): 1113.4792
Root Mean Squared Error (RMSE): 33.3688
==================================
========== KNN Regressor ==========
Best Parameters: {'metric': 'euclidean', 'n_neighbors': 7, 'weights': 'distance'}
Precision Learn: 1.0000
Precision Validation: -0.2672
Mean Absolute Error (MAE): 32.2593
Mean Squared Error (MSE): 1405.4382
Root Mean Squared Error (RMSE): 37.4892
==================================
```

Esempio di utilizzo:

```
Inserisci il nome del primo Pokemon: infernape
Inserisci il nome del secondo Pokemon: arceus
(392) Infernape VS (493) Arceus
Arceus è il vincitore!
```

Costruzione di un team ottimizzato

Sommario

Il modulo di ottimizzazione del team Pokémon consente di costruire squadre personalizzate e ottimizzate basate sulle preferenze dell'utente. Sfruttando le statistiche dei Pokémon e applicando vincoli predefiniti, il sistema garantisce una selezione efficiente e bilanciata. Il programma permette di scegliere le statistiche da ottimizzare (ad esempio, "HP", "Attack", "Defense") e include un filtro per escludere Pokémon leggendari. Dopo la selezione, il dataset viene ridotto ai Pokémon con le migliori prestazioni in base alle statistiche selezionate. La selezione del team migliore avviene sfruttando un approccio multi-threading, che consente di calcolare in parallelo il punteggio di ogni possibile soluzione. Il punteggio di un team è determinato dalla somma delle statistiche ottimizzate dei Pokémon. Un elemento chiave del sistema è l'uso dell'approccio Backtracking per la risoluzione del problema di soddisfazione dei vincoli. Questo metodo consente di esplorare lo spazio delle soluzioni rispettando i vincoli definiti, come la diversità del team e il limite sui tipi.

Strumenti utilizzati

- **Pandas:** Vedi sezione precendente
- **Constraint:** Utilizzata per risolvere constraint satisfaction problem
- **Concurrent:** Utilizzata per sfruttare il multi-threading

Decisioni di progetto

Le principali decisioni progettuali per il sistema di ottimizzazione dei team Pokémon includono:

- **Ottimizzazione delle Statistiche:**
  L'utente può selezionare fino a tre statistiche tra quelle disponibili nel dataset (ad esempio, "HP", "Attack", "Defense") per ottimizzare la squadra. Le statistiche selezionate vengono combinate per calcolare una metrica aggregata (*Optimized_Stat*) per ogni Pokémon. Questa aggregazione permette di identificare i Pokémon con le prestazioni complessive migliori in base alle preferenze dell'utente

- **Filtraggio del Dataset:**
  L'algoritmo prevede un filtro opzionale per escludere i Pokémon leggendari, in base alla scelta dell'utente. Inoltre, il dataset viene ridotto ai 50 Pokémon con il valore più alto di *Optimized_Stat*. Questo filtro migliora l'efficienza dell'elaborazione e garantisce che il sistema operi su un sottoinsieme rilevante di dati

- **Configurazione dei Vincoli:**
  Il sistema grazie alla libreria constraint può modellare due vincoli fondamentali:
  - **Diversità del team**: tutti i Pokémon selezionati devono essere unici.
  - **Limitazione del tipo**: ogni tipo di Pokémon (o combinazione di tipi) non può essere presente in più di due membri del team.
  - **Presenza di leggendari**: vincolo impostato dall'utente, serve a creare un team con la presenza di Pokémon leggendari o senza

  Questi vincoli assicurano che il team sia equilibrato e strategicamente valido.

- **Approccio Multi-threading per l'Ottimizzazione:**
  Il sistema sfrutta ThreadPoolExecutor per eseguire in parallelo la valutazione delle soluzioni. Ogni soluzione rappresenta una combinazione possibile di Pokémon per il team e, il punteggio viene calcolato come somma delle statistiche ottimizzate dei Pokémon selezionati. Questo approccio migliora significativamente le prestazioni computazionali, specialmente in presenza di dataset più grandi o vincoli complessi.

- **Backtracking**:
  La risoluzione del problema CSP avviene attraverso il backtracking, un approccio sistematico che esplora lo spazio delle soluzioni verificando le assegnazioni parziali e rispettando i vincoli definiti. Questo metodo garantisce l'efficienza nella ricerca della soluzione ottimale rispetto al semplice test di tutte le possibili combinazioni (come il metodo Generate-and-Test). Non sono stati utilizzati altri approcci, come la consistenza degli archi o la rappresentazione tramite grafi di ricerca.

Valutazione:

Il sistema di Costruzione di un team ottimizzato è stato valutato attraverso vari scenari di input per verificarne l'efficienza e la capacità di generare squadre bilanciate e competitive. Le metriche adottate includono:

- **Precisione della Selezione:**
  La pertinenza del team generato è stata analizzata confrontando i Pokémon selezionati con le preferenze specificate dall'utente. In particolare, è stata valutata l'efficacia dell'algoritmo nel massimizzare le statistiche indicate come prioritarie dall'utente, mantenendo la coerenza con i vincoli di diversità e tipi.

- **Soddisfazione dell'Utente:**
  Gli utenti hanno fornito feedback qualitativo sulle squadre generate. I risultati mostrano che i team proposti sono equilibrati e rispondono alle esigenze degli utenti in termini di prestazioni e strategia. La possibilità di includere o escludere Pokémon

leggendari è stata particolarmente apprezzata, consentendo maggiore flessibilità nella generazione del team.

- **Efficienza Computazionale:**
  L'utilizzo del multithreading ha permesso di valutare rapidamente un elevato numero di combinazioni possibili. Questo approccio ha garantito tempi di risposta accettabili anche con dataset più grandi e vincoli complessi, dimostrando la scalabilità del sistema.

Il sistema di ottimizzazione del team Pokémon si è dimostrato efficace nel fornire squadre personalizzate e competitive, rispettando le preferenze e i vincoli definiti dall'utente. Grazie all'integrazione di filtri, vincoli strategici e ottimizzazione parallela, il sistema offre una soluzione affidabile per l'ottimizzazione di team Pokémon.

Esempio di utilizzo:

```
Vuoi includere Pokémon leggendari? (si/no): no
Inserisci le statistiche da ottimizzare (es. 'HP', 'Attack', 'Defense', 'Sp. Atk', 'Sp. Def', 'Speed')
Inserisci la statistica 1: hp
Inserisci la statistica 2: Sp. atk
Inserisci la statistica 3: Speed
Il tuo team ottimizzato è:
SceptileMega Sceptile
AlakazamMega Alakazam
Blissey
```

Conclusione

Il progretto PokéJob si dimostra ottimo per l'integrazione di tecinche di apprendimento supervisionato, preprocessing dei dati, analisi dei dati, apprendimento non supervisionato e la risoluzione di un constraint satisfaction problem. Sono stati ottenuti ottimi risultati nel modulo "classificazione dei Pokémon leggendari" con l'utlizzo di diversi algoritmi di classificazione (K-Nearest Neighbors, Random Forest, Gaussian Naive Bayes, Decision Tree e Support vector machine). Nel modulo "raccomandazione di 5 Pokémon" è stato implementato il clustering che stando ai test mostra risultati davvero ottimi. Nel modulo "predizione risultato di una battaglia" vengono utilizzati diversi algoritmi di regressione (Random Forest regression, Decision Tree regression, KNN regression, SVM regression), i quali in alcuni casi (random forest in particolare) hanno mostrato una buona accuratezza in una previsione molto difficile. Anche nel modulo "Costruzione di un team ottimizzato" sono stati riscontrati ottimi risultati.

PokéJob ha ancora ampii margini di miglioramento, ad esempio si potrebbero implementare modelli di clustering più avanzati per avere una predizione ancora più accurata; un altro aspetto sicuramente migliorabile è l'accuratezza delle metriche usate nei due moduli dove si fa uso dell'apprendimento supervisionato in modo da avere previsioni molto più accurate ad esempio andando ad integrare delle tecniche di deep learning, un'altra cosa che potrebbe essere migliorata sarebbe il tempo di computazione durante l'esecuzione del modulo "Costruzione di un team ottimizzato" in quanto allo stato attuale risulta moderatamente elevato (senza il multi-threading i tempi di computazione sarebbero stati esponenzialmente più lunghi)

Bibliografia

[1] Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.