Ingegneria della Conoscenza applicata al videogioco FIFA 23

Michelangelo Balzano

Gabriele Grossano

a.a. 2022/2023

link repository GitHub:
`https://github.com/michelangelobalzano/progetto-Icon`

Indice

1. Introduzione al progetto

1.1. FIFA 23

FIFA 23 è un videogioco di calcio sviluppato da EA Sports e disponibile su tutte le principali piattaforme di videogiochi della serie di videogiochi chiamata, appunto, FIFA. Al suo interno ci sono le squadre dei principali campionati di calcio di tutto il mondo. Ogni squadra ha ovviamente la sua lista di calciatori, tutti con caratteristiche differenti.

1.2. Il dataset

Il dataset di cui disponiamo è stato trovato su Kaggle e contiene le informazioni su tutti i calciatori presenti sul gioco. Oltre al nome del calciatore e alla squadra di appartenenza, ci sono tanti attributi riguardanti le qualità tecniche. Al calciatore è associato un ***overall***, ovvero un punteggio complessivo, che rappresenta il calciatore in base alle sue qualità.

Il nostro progetto si concentra principalmente sull'overall dei calciatori che, come tutti gli altri attributi riguardanti la qualità, è un numero, che può assumere un valore tra 0 e 100.

1.3. Task effettuate

1.3.1 Apprendimento supervisionato

Il primo compito riguarda l'**apprendimento supervisionato**. In particolare, ci siamo concentrati sulla predizione dell'overall di un calciatore a partire dalle caratteristiche tecniche specifiche. Abbiamo confrontato differenti modelli di apprendimento supervisionato: la ***regressione lineare***, gli ***alberi decisionali***, il ***knn*** e il ***random forest***. Per minimizzare problemi di sovradattamento dei modelli abbiamo inoltre applicato la tecnica di convalida incrociata ***k-fold cross validation*** su tutti i modelli appena citati. Inoltre, per tutti i modelli, tranne che per quello della regressione lineare, sono stati testati valori differenti per i propri iperparametri più incisivi. La metrica di valutazione e di confronto dei modelli utilizzata è l'***RMSE***.

1.3.2 CSP

Il secondo compito è la risoluzione di un ***Costraints Satisfaction Problem (CSP)***. L'obiettivo è di trovare la migliore formazione possibile, dato un modulo e un budget massimo, in base all'overall dei calciatori. In particolare, si tratta di un problema di ottimizzazione, dove si cerca di massimizzare l'overall medio della formazione. Ad ogni modulo corrispondono undici ruoli differenti che l'algoritmo deve rispettare per cercare il miglior giocatore per ruolo.

Il nostro dataset si presta bene a tale problema di ricerca in quanto per ogni calciatore c'è l'attributo *Positions Played* contenente le posizioni che lo stesso può occupare in campo.

1.3.3 Apprendimento non supervisionato

Il terzo compito riguarda l'apprendimento non supervisionato. Si cerca di trovare dei possibili cluster di raggruppamento dei calciatori in base alle caratteristiche tecniche. Gli algoritmi di clustering utilizzati sono il ***k-means clustering*** e l'***hierarchical clustering***. L'attività di clustering è stata preceduta da un'attività di ***Principal Component Analysis (PCA)*** per semplificare il dataset in poche componenti e da un'analisi delle posizioni e dei reparti di gioco. Successivamente, sulla base delle componenti principali, sono stati effettuati calcoli sui calciatori più simili ad alcuni calciatori di esempio, come Leo Messi.

1.3.4 Knowledge base

L'ultimo compito riguarda la realizzazione di una ***knowledge base*** contenente un insieme di fatti e regole riguardanti i calciatori di FIFA da poter utilizzare per effettuare delle ***query***. I fatti sono stati estratti dal dataset in python e scritti in Prolog. L'interrogazione della base di conoscenza avviene attraverso Python.

1.4. Strumenti utilizzati

Il progetto è stato sviluppato in **Python** e, per la parte riguardante la base di conoscenza, in **Prolog**. Come ambiente di sviluppo è stato utilizzato **Visual Studio Code**. Il controllo di versione è stato effettuato tramite **GitHub**. Le librerie Python utilizzate sono le seguenti:

- **scikit-learn**: per i metodi che implementano i modelli di apprendimento e altre funzioni utili allo scopo;
- **pandas**: per gestire il dataset in formato "csv";
- **numpy**: per metodi riguardanti calcoli matematici;
- **matplotlib**: per la creazione e visualizzazione di grafici.
- **pyswip**: per interagire con file Prolog attraverso Python;
- **unidecode**: per la rimozione degli accenti dalle stringhe del dataset, nella task KB;
- **prettytable**: per la stampa di tabelle utilizzate per la stampa di una singola formazione nella task del CSP;
- **scipy**: per la realizzazione del dendogramma dell'hierarchical clustering nella task del clustering.

2. Apprendimento supervisionato

L'obiettivo è di prevedere il valore di overall di un calciatore a partire dai valori delle singole caratteristiche tecniche.

2.1. Preprocessing

Prima di iniziare ad effettuare le predizioni, abbiamo svolto le seguenti attività di ***pre-processing*** dei dati:

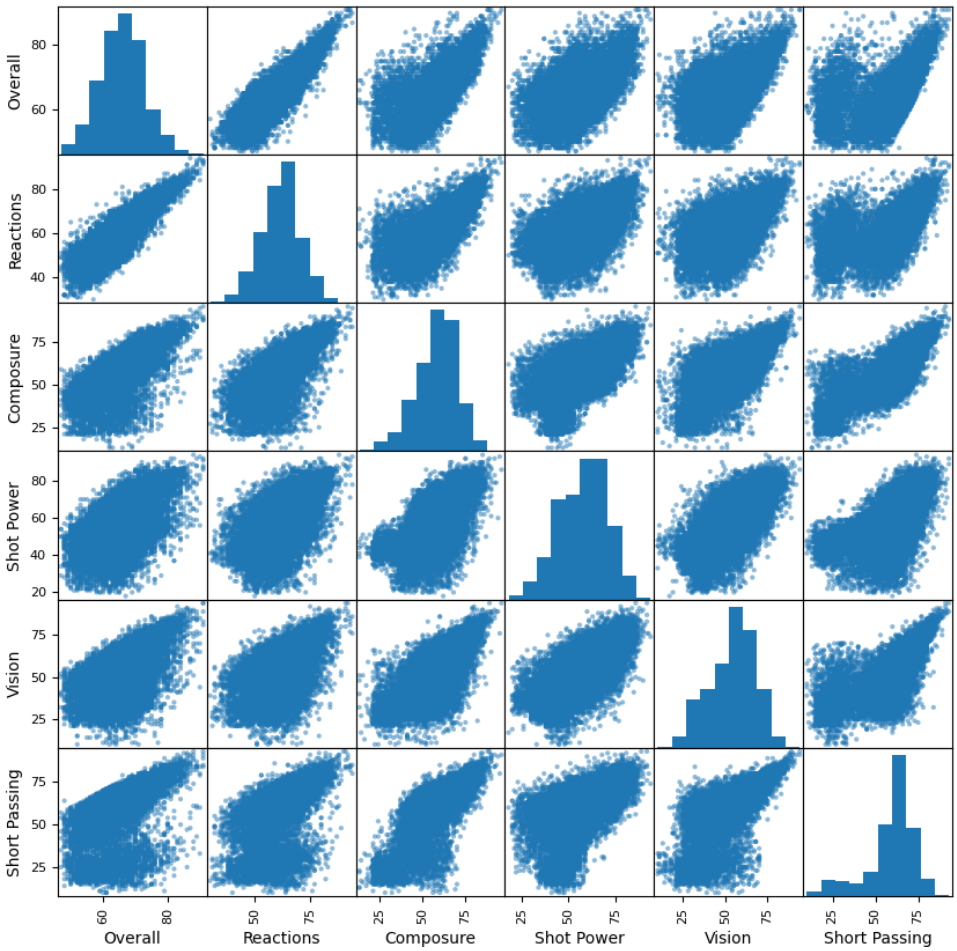
1. **Verifica dati mancanti**: il primo passo è quello di verificare se nel dataset ci sono dei valori mancanti per uno o più attributi e per ogni calciatore. Il nostro dataset è una tabella 18539 x 89, ovvero 18539 calciatori e 89 attributi per calciatore, ordinati proprio per overall. Successivamente abbiamo stampato le informazioni di ciascun attributo e, per fortuna, tutti gli attributi sono di tipo "not-null", dunque non ci sono valori mancanti.
2. **Rimozione attributi superflui**: una tupla del dataset, che rappresenta il singolo calciatore, contiene tante informazioni che non sono utili al nostro scopo come il nome e cognome, la squadra di appartenenza, il suo valore economico, il suo stipendio, ecc. Tali colonne vanno rimosse lasciando soltanto quelle riguardanti le caratteristiche tecniche. Gli attributi rimasti sono i seguenti:
   [*Overall, Crossing, Finishing, Heading Accuracy, Short Passing, Volleys, Dribbling, Curve, Freekick Accuracy, LongPassing, BallControl, Acceleration, Sprint Speed, Agility, Reactions, Balance, Shot Power, Jumping, Stamina, Strength, Long Shots, Aggression, Interceptions, Positioning, Vision, Penalties, Composure, Marking, Standing Tackle, Sliding Tackle, Goalkeeper Diving, Goalkeeper Handling, Goalkeeper Kicking, Goalkeeper Positioning, Goalkeeper Reflexes*].

3. **Ricerca di correlazioni**: prima di cominciare con l'apprendimento, abbiamo calcolato il coefficiente di correlazione tra tutte le coppie di attributi, tra quelli rimasti, in modo tale da capire quanto l'uno incide sull'altro, creando una matrice di correlazioni. Ovviamente, ciò che ci interessa è la colonna della matrice corrispondente all'overall in modo da visualizzare la correlazione di tutti gli altri attributi con lo stesso.

Overall	1.000000
Reactions	0.872789
Composure	0.700583
Shot Power	0.552518
Vision	0.523290
Short Passing	0.520784
LongPassing	0.507192
BallControl	0.457705
Curve	0.416575
Crossing	0.395435
Aggression	0.393433
Long Shots	0.392187
Stamina	0.376994
Dribbling	0.376185
Freekick Accuracy	0.374162
Volleys	0.367168
Strength	0.353857
Positioning	0.348038

Heading Accuracy	0.342452
Finishing	0.331967
Penalties	0.331834
Marking	0.313975
Interceptions	0.311216
Jumping	0.282737
Standing Tackle	0.266683
Agility	0.254826
Sliding Tackle	0.240499
Sprint Speed	0.213367
Acceleration	0.198758
Balance	0.142749
Goalkeeper Positioning	-0.008318
Goalkeeper Reflexes	-0.013110
Goalkeeper Handling	-0.013655
Goalkeeper Diving	-0.016593
Goalkeeper Kicking	-0.016764

Inoltre, abbiamo selezionato i migliori 5 attributi, in base al loro valore di correlazione, e abbiamo visualizzato graficamente la loro incisione sull'overall. Più un valore è incisivo, più tende ad essere direttamente proporzionale al valore sul quale incide, quindi, il suo grafico tenderà ad una retta del tipo $y = x$.
Il grafico dimostra la forte correlazione che c'è tra l'overall e l'attributo *Reactions*.
Dall'analisi delle correlazioni, inoltre, si evince come gli attributi riguardanti i portieri (*Goalkeeper Positioning, Goalkeeper Reflexes, Goalkeeper Handling, Goalkeeper Diving, Goalkeeper Kicking*) abbiano una correlazione negativa, ovvero inferiore allo 0. Questo risultato afferma che tali caratteristiche diminuiscono all'aumentare dell'overall, e viceversa.


Overall
Reactions
Composure
Shot Power
Vision
Short Passing

4. **Rimozione dei portieri**: a causa delle correlazioni negative degli attributi riguardanti le caratteristiche tecniche dei portieri, abbiamo capito che, per i giocatori di questo ruolo, dovrebbe essere effettuata un'attività di apprendimento separato.
   Non essendo giocatori di movimento, i portieri hanno caratteristiche tecniche totalmente diverse: le caratteristiche riguardanti le abilità con i piedi sono molto basse. Analogamente, per tutti gli altri ruoli, le caratteristiche dei portieri hanno valori molto bassi. Abbiamo deciso di rimuovere dal dataset tutti i portieri e tutte le caratteristiche che li riguardano.
5. **Suddivisione dei dati**: prima di scegliere il modello di apprendimento supervisionato più adatto, abbiamo suddiviso il dataset in due sottoinsiemi: un **training set** per effettuare appunto training, e un **test set** per applicare il modello scelto ed effettuare le predizioni. In particolare la suddivisione è dell'80% per gli esempi di addestramento e il rimanente 20% per i test, ovviamente scelti in modo casuale.

2.2. Valutazione dei modelli

Come metodo di valutazione dell'errore dei modelli utilizzati abbiamo scelto l'**RMSE**, ovvero la radice dell'errore quadratico medio. Tale misura è molto utilizzata nei problemi di regressione. Esso rappresenta, in poche parole, la distanza che c'è tra il vettore di valori previsti e il vettore dei valori osservati.

Per ogni modello di apprendimento testato è stato elaborato un grafico che rappresenta il valore dell'RMSE calcolato sul training set, l'RMSE calcolato con la k-fold cross validation, e infine, una volta addestrato il modello, l'RMSE calcolato sul test set.

2.3. Regressione lineare

Il primo modello testato è quello della **regressione lineare**. Gli step che abbiamo effettuato sono i seguenti:

1. **Addestramento del modello**: in questa fase addestriamo il modello utilizzando il training set in modo da migliorarlo. L'RMSE ottenuto è di circa *2.18*, non male;
2. **Validazione del modello**: per convalidare i modelli e ridurre il problema del sovradattamento, abbiamo utilizzato la **k-fold cross validation** con il valore k impostato a 10. Il modello valuta quindi 10 fold differenti. Successivamente abbiamo calcolato la media dei 10 valori ottenuti da ciascun fold e il risultato ottenuto è il linea con il precedente, ovvero *2.19*;
3. **Valutazione del modello**: dopo aver addestrato il modello, l'abbiamo applicato al test set, ottenendo, più o meno, sempre lo stesso risultato: *2.17*.

I tre risultati ottenuti dalle tre fasi dell'apprendimento sono molto simili tra loro. In questo caso, abbiamo un modello ben bilanciato che non presenta problemi di sovradattamento e generalizza bene su dati che non ha mai visto, ovvero quelli del test set.

2.4. Albero decisionale

Il secondo modello analizzato è quello dell'**albero decisionale**. Abbiamo deciso di valutare questo modello con differenti valori per l'attributo *max_ depth*. Questo attributo indica la profondità massima dell'albero. I valori di *max_ depth* testati sono:

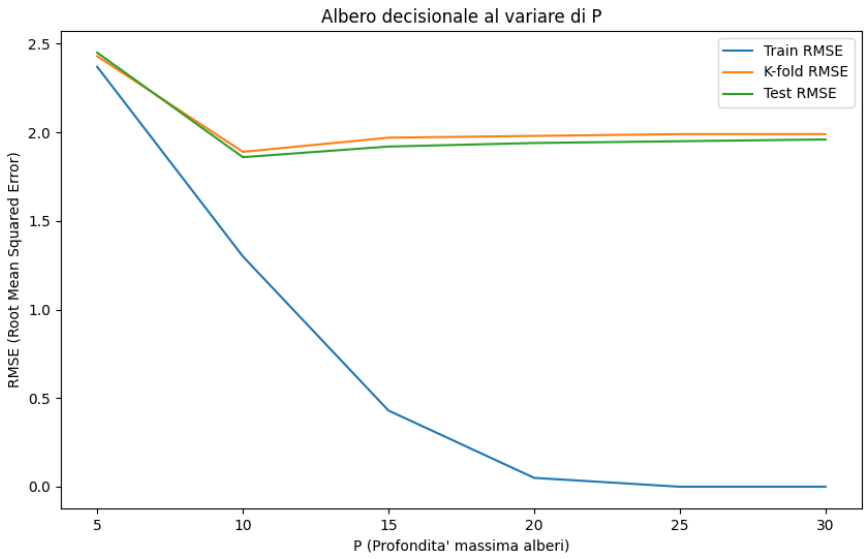
$$5, 10, 15, 20, 25, 30$$

Tutte e tre le fasi dell'apprendimento sono state svolte per ogni profondità massima selezionata.

Nella fase di training abbiamo notato che all'aumentare della profondità massima, l'RMSE tende a *0*. Potrebbe essere un modello perfetto oppure potrebbe verificarsi un eccessivo sovradattamento.

Successivamente abbiamo effettuato la validazione (sempre con la *k-fold cross validation*) e la valutazione del modello alle diverse profondità massime. La profondità massima dell'albero che ci fa ottenere il miglior punteggio è *10*, con un RMSE uguale a circa *1.8*.

Abbiamo effettuato un ulteriore test senza imporre un limite di profondità notando che da una profondità massima di 15 in poi, l'RMSE rimane costante (circa 1.9).



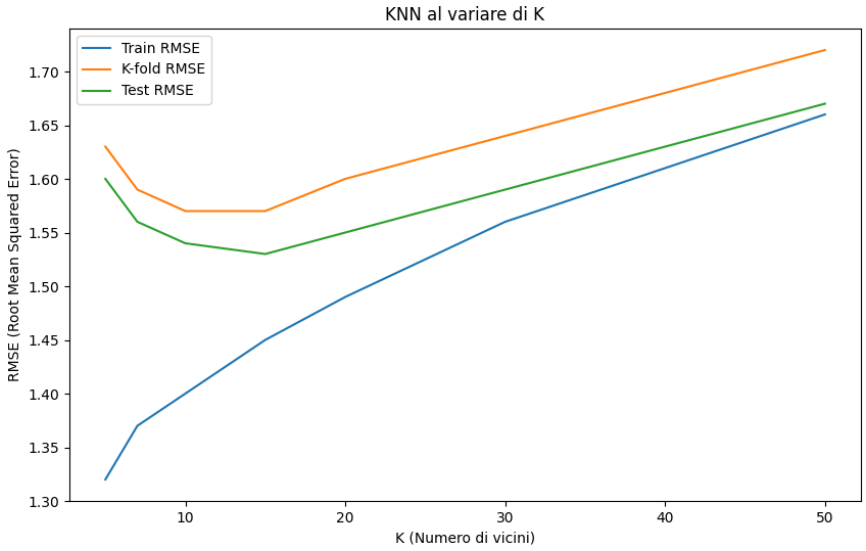
2.5. KNN

Il terzo modello che testiamo è un modello del ***case-based reasoning***, ovvero il ***kNN***. Per il knn abbiamo valutato differenti valori per l'attributo *n_neighbors* che indica il numero di vicini da considerare. I valori valutati sono i seguenti:

$$5, 7, 10, 15, 20, 30, 50$$

Non abbiamo considerato valori minori di 5 in quanto rendono il modello molto sensibile al rumore, facendosi influenzare da dati anormali, producendo un RMSE alto. Come numero di vicini massimo ci siamo fermati a 50 per non semplificare troppo il modello. Infatti, l'RMSE del modello sul training set tende ad aumentare man mano che il numero di vicini cresce. Questo si verifica in quanto il modello diventa sempre più semplice, fino a diventare poco adattabile ai dati.

Analizzando i risultati ottenuti con la *k-fold cross validation* e la valutazione sul test set, sembra che il miglior valore per il numero di vicini sia intorno a $10 - 15$, dove otteniamo un RMSE di circa 1.53.



2.6. Random forest

L'ultimo modello è quello del **random forest**. Questo modello ha differenti iperparametri che determinano di molto il risultato dell'apprendimento. In particolare, abbiamo valutato diversi valori per gli iperparametri *n_estimators* e *max_depth*, che rappresentano rispettivamente il numero di alberi della foresta e la profondità massima del singolo albero, come già visto per il modello dell'albero decisionale.

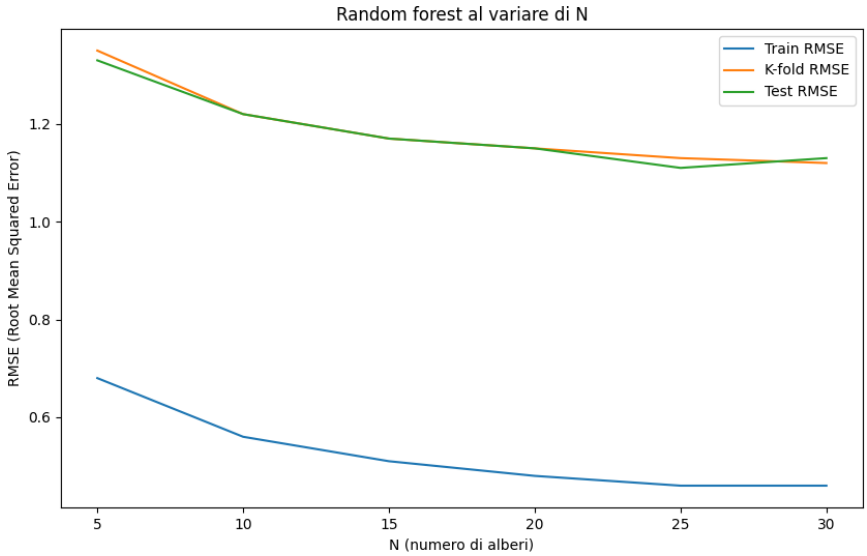
Il random forest ha un altro importante iperparametro: *max_features*, che indica il numero massimo di caratteristiche che vengono considerate per la suddivisione in ogni nodo dell'albero. In altre parole, limita il sottoinsieme delle caratteristiche che ogni albero può utilizzare per prendere decisioni durante la costruzione. Per questo iperparametro, dopo vari test, abbiamo deciso di mantenere il suo valore fisso ad 8. Valori maggiori vanno ad aumentare l'RMSE oltre che il costo computazionale.

Innanzitutto abbiamo provato diversi valori per *n_estimators*, tra i quali

$$5, 10, 15, 20, 25, 30$$

senza imporre un limite sulla profondità massima. Ci siamo fermati a 30 come valore massimo perché valori più alti impiegano un elevato tempo di calcolo.

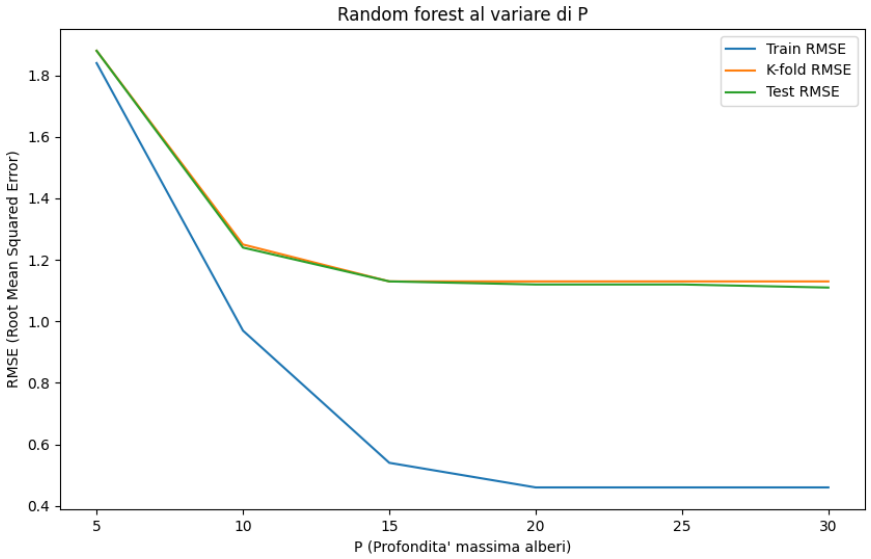
Il miglior risultato l'abbiamo ottenuto proprio con *n_estimators* uguale a $25 - 30$ e ce l'aspettavamo poiché il modello del random forest aumenta l'adattabilità ai dati all'aumentare del numero di alberi.



Dopo aver selezionato il miglior valore per l'attributo *n_ estimators* (30), abbiamo valutato differenti valori per l'attributo *max_ depth*, ovvero

$$5, 10, 15, 20, 25, 30$$

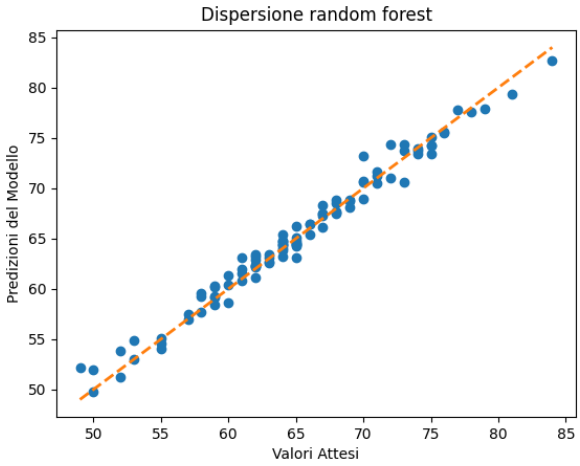
gli stessi utilizzati per *max_ depth* nel modello dell'albero decisionale.



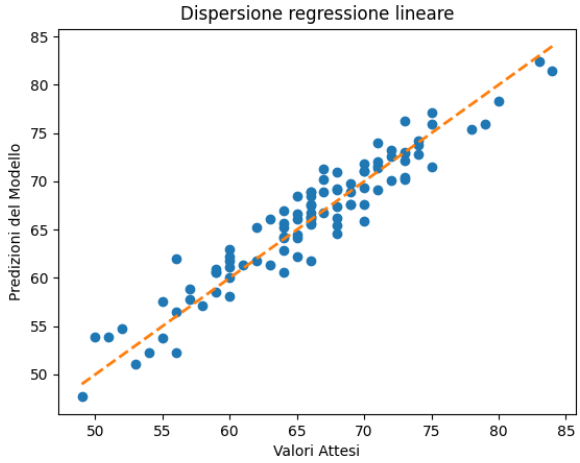
Il miglior modello tra quelli testati risulta essere il random forest in quanto produce l'RMSE più basso. I suoi miglior valori per gli iperparametri di *n_ estimators* e *max_ depth*, tra quelli provati sono $25 - 30$ per *n_ estimators* e $15 - 20$ per *max_ depth*.

2.7. Considerazioni finali

Come ulteriore dimostrazione dell'efficienza del modello, abbiamo realizzato un grafico di dispersione che mostra come le predizioni del modello si allineano rispetto ai valori attesi. Per far ciò abbiamo selezionato 100 esempi casuali dal test set, abbiamo effettuato le previsioni e le abbiamo confrontate con gli attributi target degli esempi. I valori degli iperparametri di *n_estimators* e *max_depth* sono rispettivamente 25 e 20.



Per rendere più chiara la dimostrazione dell'efficienza del random forest, abbiamo prodotto il grafico di dispersione anche per il modello che ha portato i risultati peggiori, ovvero la regressione lineare. Si nota subuto la maggiore dispersione delle predizioni rispetto ai valori attesi confronto al grafico del random forest.



3. CSP

Il CSP affrontato consiste nel cercare di ottenere la miglior formazione possibile per un determinato modulo e che rispetti un budget economico. Un modulo corrisponde ad un vettore di undici posizioni differenti, ognuna del quale deve essere ricoperta da un calciatore. Ogni calciatore è dotato di un attributo *Positions played* che comprende l'insieme delle posizioni che può ricoprire. Inoltre, ad ogni calciatore è associato l'attributo *Value(in euro)*, che rappresenta il suo valore economico.

Vista la grandezza del dataset (18000+ tuple), applicare algoritmi di ricerca che analizzano l'intero spazio delle soluzioni non sarebbe stato ragionevole. Poiché il nostro obiettivo è la massimizzazione di un valore, abbiamo deciso di utilizzare metodi di ***ricerca locale*** che analizzano soltanto una porzione dello spazio delle soluzioni sapendo però che potrebbero convergere ad un massimo locale e non globale.

Il CSP è stato risolto attraverso cinque algoritmi di ricerca locale differenti: *Hill Climbing*, *Tabu Search*, *Simulated Annealing*, *Most Improving Step* e un *algoritmo genetico*. Ogni algoritmo è stato testato 500 volte sullo stesso modulo e con lo stesso budget a disposizione per poi calcolare la media dei risultati ottenuti. Il modulo testato è il *4-3-3*, con un budget di 350 milioni.

La ***funzione di valutazione*** utilizzata si basa sulla media dell'overall dei calciatori. Una singola sostituzione è migliorativa se va ad incrementare l'overall medio della formazione, non peggiorativa se l'overall della sostituzione è uguale, peggiorativa se è inferiore. Inoltre, ogni sostituzione effettuata rispetta il budget. Se una sostituzione fa sforare il budget viene ignorata direttamente.

3.1. Preprocessing

L'attività di preprocessing per il task di risoluzione del CSP consiste nei seguenti passaggi:

1. **verifica dati mancanti**: la verifica è già stata effettuata nell'attività di apprendimento supervisionato. Il dataset non presenta dati mancanti;
2. **rimozione attributi superflui**: per questa task sono necessari soltanto gli attributi del nome del calciatore, le posizioni in cui può giocare, l'overall e il suo valore economico. Tutti gli altri valori sono stati scartati;
3. **conversione euro - milioni di euro**: per rendere più chiare e meno confusionarie le cifre del valore del calciatore, abbiamo effettuato la conversione da euro a milioni di euro (ad esempio: $50000000 \rightarrow 50$);
4. **conversione sigla posizione - nome posizione**: come per il valore economico, vogliamo rendere più chiaro anche la posizione, convertendo le sigle in nomi di posizione (ad esempio: GK $\rightarrow$ Portiere);
5. **creazione vettore calciatori**: una volta terminate tutte le fasi precedenti, è stato creato un vettore contenente tutti i calciatori, presenti una volta per ogni posizione in cui possono giocare, dal quale pescare per effettuare le sostituzioni.

3.2. Hill Climbing

Il metodo Hill Climbing è il più semplice e immediato metodo di ricerca locale. Consiste nel Iterative Best Improvement che cerca di massimizzare il valore dato dalla funzione di valutazione andando a lavorare su un'unica assegnazione iniziale.

Viene effettuato un *random restart* all'inizio, seguito soltanto da passi iterativi di *random walk*. Ad ogni passo e per ogni posizione del modulo si pesca casualmente dal dataset un giocatore che potrebbe sostituire quello selezionato (se le posizioni combaciano e se non è già presente nella formazione).

Nel nostro caso, l'algoritmo effettua la sostituzione se si verifica una delle seguenti due condizioni:

- il budget non viene sforato e l'overall medio viene migliorato;
- il costo della singola assegnazione è inferiore a quello attuale e l'overall della singola assegnazione ha un peggioramento massimo del 3% rispetto all'overall da sostituire (il tasso di peggioramento è stato scelto dopo vari test: il giocatore con overall maggiore, 91, potrebbe essere sostituito al massimo da un 89, a patto che il suo costo sia inferiore).

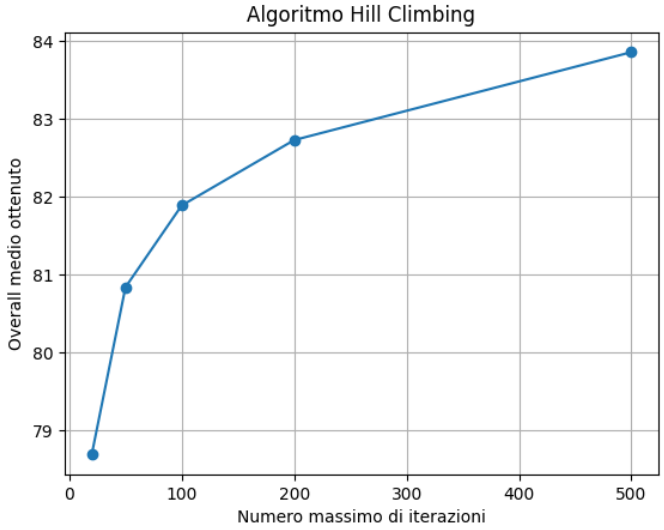
L'Hill Climbing implementato non effettua soltanto mosse migliorative ma anche mosse peggiorative che però vanno a diminuire il costo totale della formazione.

Una volta tentata la sostituzione per tutte le posizioni del modulo, si effettua l'iterazione successiva, fino al raggiungimento del numero massimo di iterazioni.

L'algoritmo è stato testato con un diverso numero di iterazioni massime, in modo da poter studiare come i risultati cambiano all'aumentare del numero di mosse effettuate (random walk). I valori testati sono:

$$20, 50, 100, 200, 500$$

I risultati crescono all'aumentare del numero massimo di iterazioni, arrivando ad ottenere, con un numero di iterazioni massime uguale a 500, ad un overall medio di circa 83.8.



3.3. Tabu Search

Poiché l'algoritmo dell'Hill Climbing, valutando anche soluzioni non migliorative, potrebbe tornare su sostituzioni già effettuate, abbiamo testato anche l'algoritmo del Tabu Search, che mantiene memorizzate le ultime k sostituzioni. Il valore k rappresenta la lunghezza della lista dei tabu e l'abbiamo fissato a 50.

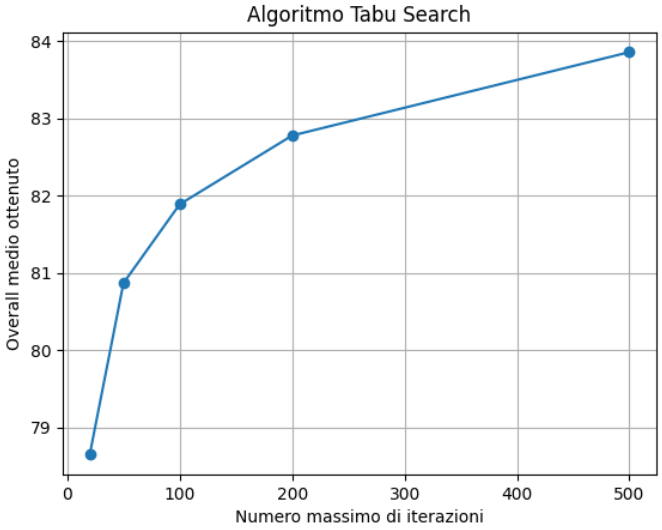
Il cambiamento principale, rispetto all'algoritmo dell'Hill Climbing, sta nel memorizzare all'interno della lista dei tabu le sostituzioni effettuate. Ad ogni nuova sostituzione, oltre che controllare che il calciatore non sia già presente nella formazione, si controlla anche che non sia presente nella lista tabu. Una volta riempita la lista tabu, una nuova sostituzione farebbe cancellare quella più vecchia, mantenendo la lista piena delle ultime 50 sostituzioni.

Anche per il Tabu Search abbiamo valutato differenti valori per il numero massimo di iterazioni, gli stessi utilizzati nell'Hill Climbing:

$$20, 50, 100, 200, 500$$

Infine, la funzione di valutazione è quella dell'Hill Climbing: la sostituzione viene effettuata se si verifica una delle due condizioni discusse precedentemente.

L'andamento è molto simile a quello dell'Hill Climbing. Non c'è stato un miglioramento. Questo è dato dal fatto che già l'Hill Climbing è stato ottimizzato in quanto non permette sostituzioni molto peggiorative e non permette di sforare il budget. Succede quindi raramente di tornare su sostituzioni già effettuate, dunque si ottengono risultati simili.



3.4. Simulated Annealing

Il Simulated Annealing permette di considerare anche sostituzioni molto peggiorative con lo scopo di superare minimi locali, consentendo l'esplorazione di soluzioni che potrebbero portare a un massimo globale più elevato.

Si parte da una temperatura fissata a 2000. L'algoritmo è stato testato con differenti valori del tasso di raffreddamento. Più la temperatura è alta, più c'è la probabilità di accettare una sostituzione peggiorativa. Con differente tasso di raffreddamento, abbiamo testato differenti velocità di discesa della temperatura. I valori del tasso testati sono:

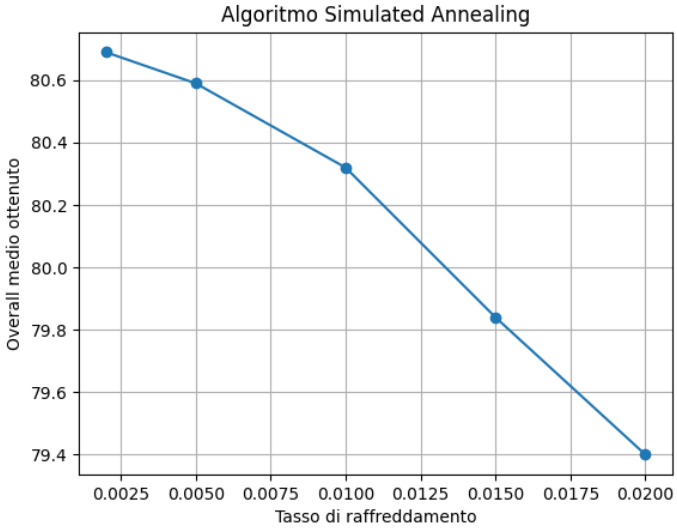
$$0.02, 0.015, 0.01, 0.005, 0.002$$

dove, ad esempio, un tasso di raffreddamento uguale a 0.002, indica che la temperatura, dopo un iterazione, viene ridotta del $0,2\%$, ovvero è uguale al 99.8% di se stessa.

La funzione di valutazione in questo caso è differente perché si basa su una probabilità dipendente dalla temperatura attuale. La sostituzione si effettua se si verificano entrambe le seguenti condizioni:

- il nuovo costo non sfora il budget;
- si verifica una delle seguenti condizioni:
  - l'overall medio della squadra migliora;
  - si verifica l'evento con probabilità dipendente dalla temperatura attuale.

I risultati calano all'aumentare del tasso di raffreddamento, ovvero all'aumentare della velocità di discesa della temperatura. I valori ottenuti

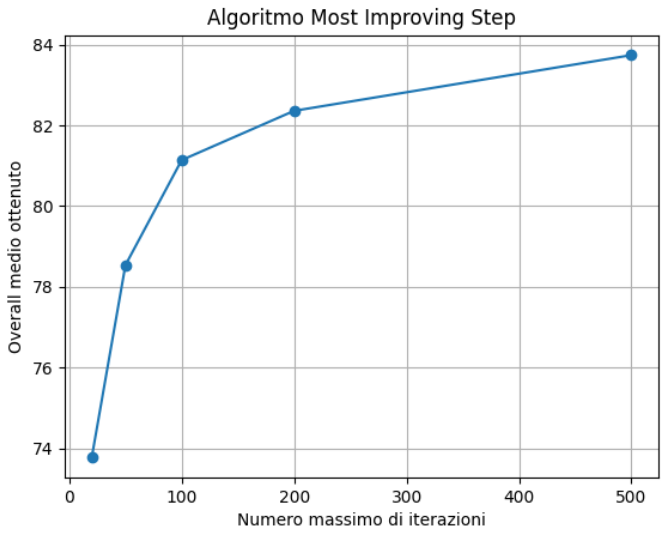


sono quelli aspettati. Infatti, con una velocità di raffreddamento maggiore, vengono effettuate meno sostituzioni perché si raggiunge più in fretta la temperatura 0. Le sostituzioni, tra l'altro, possono anche essere peggiorative. Otteniamo dei risultati più bassi rispetto ai due metodi precedenti (il risultato massimo è di circa 80.7).

3.5. Most Improving Step

Con questo algoritmo abbiamo la possibilità di valutare più sostituzioni ad ogni iterazione dal quale poter scegliere la migliore, ovvero quella che apporta un miglioramento maggiore nell'overall medio della squadra. Per implementare tale algoritmo abbiamo deciso di considerare una nuova formazione attraverso un nuovo *random restart*, dal quale poter scegliere il calciatore che, messo nella squadra di partenza nella relativa posizione, apporta il maggior miglioramento.

L'algoritmo è stato testato sui diversi valori di *max iteration* utilizzati nell'Hill Climbing e Tabu Search. Anche la funzione di valutazione funziona allo stesso modo dei due algoritmi.



Il Most Improving Step ha portato dei risultati simili all'Hill Climbing e al Tabu Search.

3.6. Algoritmo genetico

L'algoritmo genetico è un metodo che valuta una popolazione di individui (formazioni, in questo caso) contemporaneamente in modo da analizzare diverse regioni dello spazio delle soluzioni. La popolazione testata ha un numero di individui fissato a 50. Inoltre, è stato testato con diversi numeri di generazioni, dove ogni generazione corrisponde ad un'iterazione dove la popolazione si evolve. I valori testati sono i seguenti:

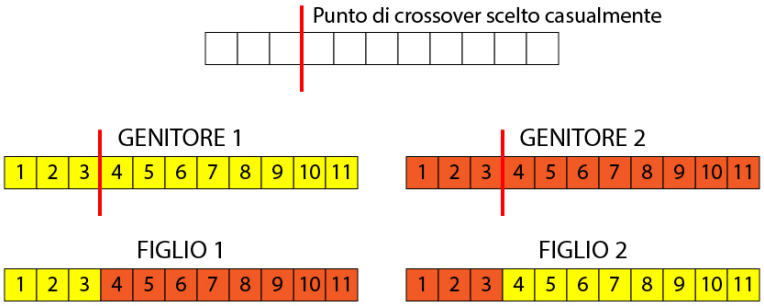
$$20, 50, 100, 200, 500$$

Per ogni nuova generazione, l'algoritmo seleziona due individui genitore con il metodo della ***roulette wheel selection***. Essi vengono selezionati in base ad una probabilità dipendente dal loro ***fitness***. Come fitness abbiamo considerato l'overall medio della formazione e, in caso di budget sforato, il fitness è uguale a 0.

A partire dai due genitori selezionati, si crea la generazione successiva. Fin quando non si raggiunge il numero di individui della popolazione prefissato, l'algoritmo effettua due procedure metafore della genetica: il crossover e la mutazione.

3.6.1 Crossover

Il metodo di crossover adottato è il ***crossover a un punto***. Si seleziona casualmente un punto della formazione in cui applicare la suddivisione e, dati i due genitori, si creano due figli dove vengono mescolate le quattro mezze formazioni.



Il crossover viene accettato solo nel caso in cui le due formazioni figlie ottenute non presentano al loro interno dei calciatori duplicati. Alcuni moduli hanno dei ruoli duplicati (ad esempio, i moduli con difesa a quattro hanno due difensori centrali), dunque potrebbero crearsi formazioni con calciatori ripetuti più di una volta. Il numero massimo di crossover falliti è fissato a 5. Se si raggiunge il numero massimo di fallimenti, il crossover restituisce i genitori stessi.

3.6.2 Mutazione

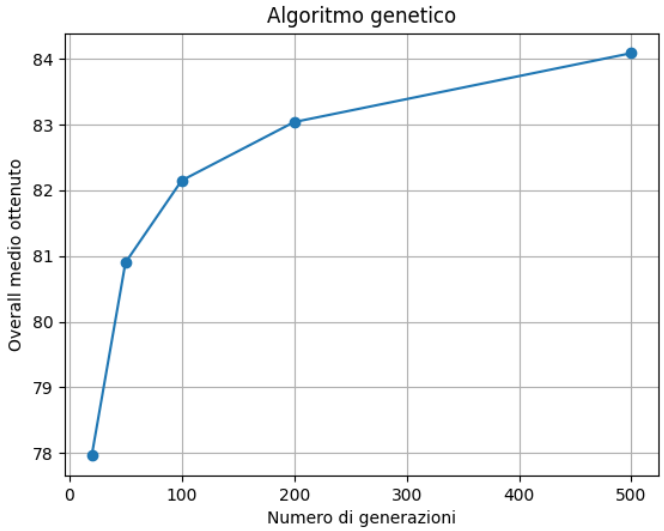
Una volta ottenuti due nuovi individui dal crossover, viene applicata agli stessi la procedura di mutazione. Come mutazione abbiamo utilizzato la sostituzione del calciatore con overall minore dell'individuo. Il calciatore nuovo viene accettato solo se si verifica una delle seguenti due condizioni:

- l'overall migliora;
- il nuovo costo è minore e viene tollerata una diminuzione dell'overall del 3%.

Come per il crossover, anche per la mutazione abbiamo previsto un numero massimo di tentativi falliti fissato a 5.

Una volta effettuato il numero di evoluzioni fissato, si restituisce il miglior individuo appartenente all'ultima generazione.

I risultati del test dell'algoritmo sono riassunti nell'immagine seguente:

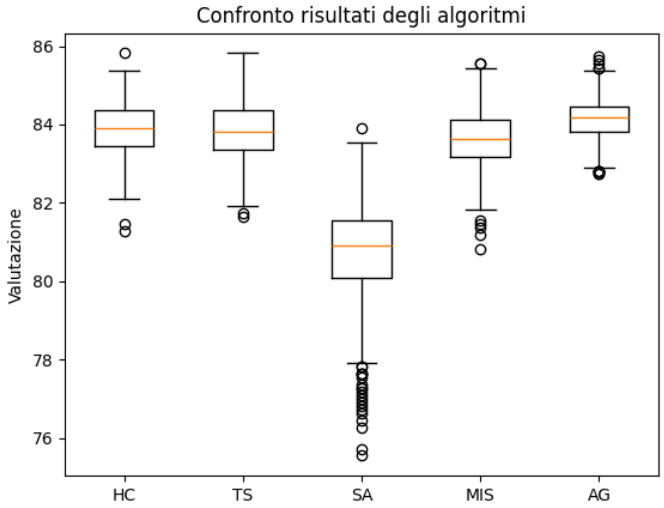


Con l'algoritmo genetico abbiamo ottenuto i risultati migliori con un overall medio di circa 84.2 con un numero di 500 generazioni.

3.7. Considerazioni finali

Come ulteriore prova dell'efficacia dei metodi, abbiamo elaborato un grafico di tipo ***box plot*** per analizzare la distribuzione statistica dei differenti risultati ottenuti. Nel grafico sono presenti i seguenti elementi:

- box: rappresenta il 50% centrale dei dati. La parte inferiore della scatola indica il 25% inferiore dei dati, mentre la parte superiore indica il 25% superiore;
- linea mediana: rappresenta il valore mediano della distribuzione;
- baffi: sono linee che si estendono dalla scatola fino ai valori più estremi che rientrano ancora entro 1,5 volte l'ampiezza della scatola;
- punti: i punti al di fuori dei baffi che indicano valori estremi che potrebbero essere considerati atipici.



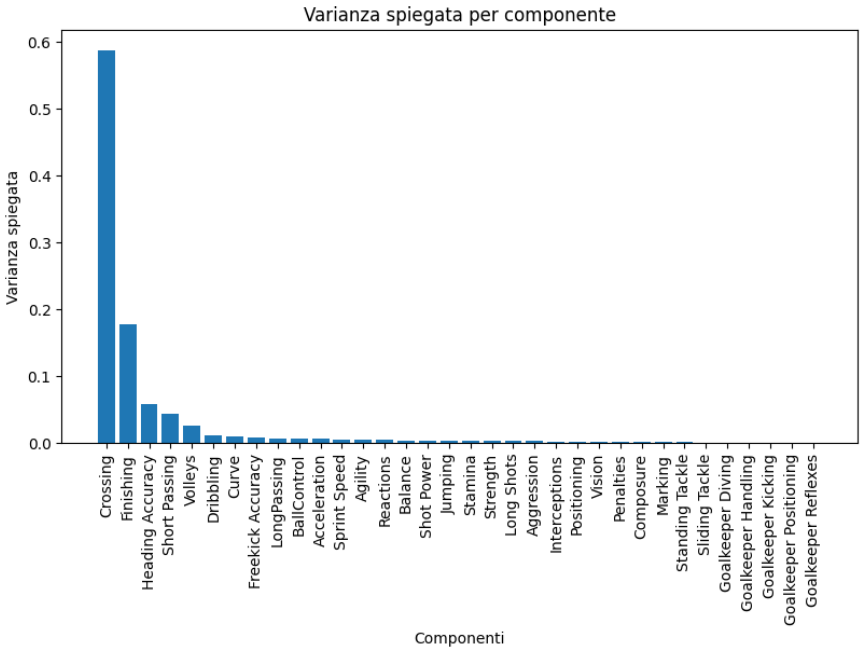
Dal grafico si evince che la migliore media è quella dell'algoritmo genetico, anche se di poco confronto all'Hill Climbing, al Tabu Search e al Most Improving Step. L'algoritmo genetico è inoltre il metodo che ha un'intervallo di distribuzione minore, data la minor ampiezza del box e dei baffi. Al contrario, l'algoritmo con ampiezza maggiore è il Simulated Annealing, che presenta anche la media peggiore.

4. Apprendimento non supervisionato

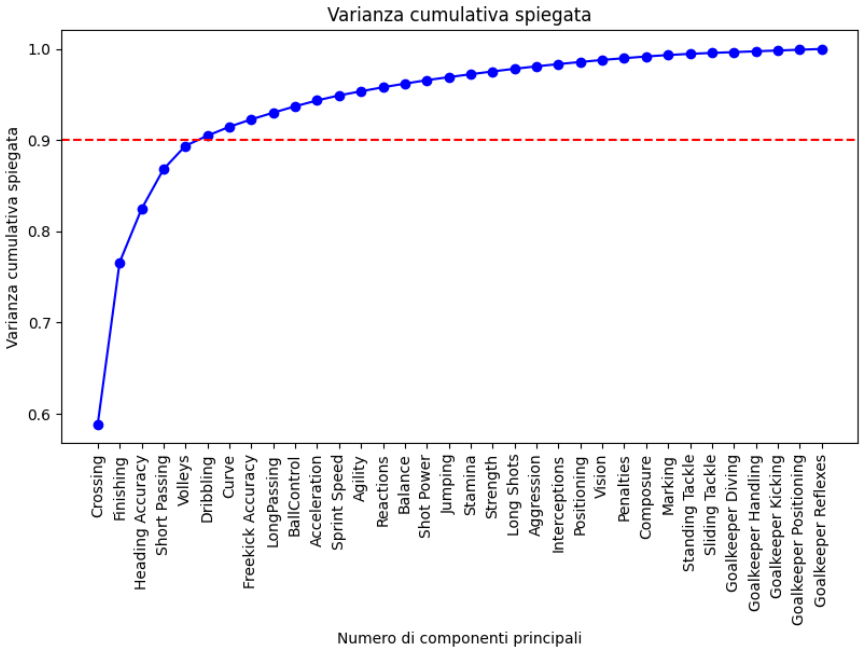
L'obiettivo è di raggruppare i calciatori in cluster in modo tale che calciatori nello stesso cluster si assomiglino, calciatori in cluster differenti siano molto diversi. La similarità o diversità di due calciatori è data dalle caratteristiche tecniche. Abbiamo selezionato le seguenti caratteristiche:
[*Crossing, Finishing, Heading Accuracy, Short Passing, Volleys, Dribbling, Curve, Freekick Accuracy, LongPassing, BallControl, Acceleration, Sprint Speed, Agility, Reactions, Balance, Shot Power, Jumping, Stamina, Strength, Long Shots, Aggression, Interceptions, Positioning, Vision, Penalties, Composure, Marking, Standing Tackle, Sliding Tackle, Goalkeeper Diving, Goalkeeper Handling, Goalkeeper Kicking, Goalkeeper Positioning, Goalkeeper Reflexes.*]

4.1. PCA

Visto il grande numero di attributi da considerare (34), abbiamo svolto un'attività di ***Principal Component Analysis (PCA)***, con lo scopo di ottenere un minor numero di variabili, meno correlate tra loro. Una volta applicato il modello PCA al dataset, abbiamo ottenuto i seguenti risultati:



Successivamente, abbiamo calcolato quali e quanti attributi sarebbero necessari per avere una varianza cumulativa del 90%.

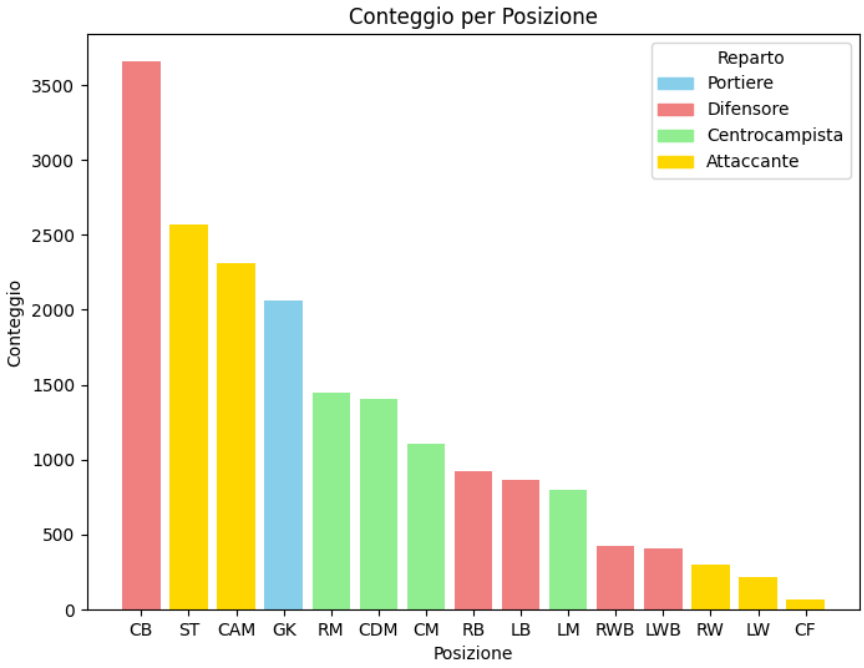


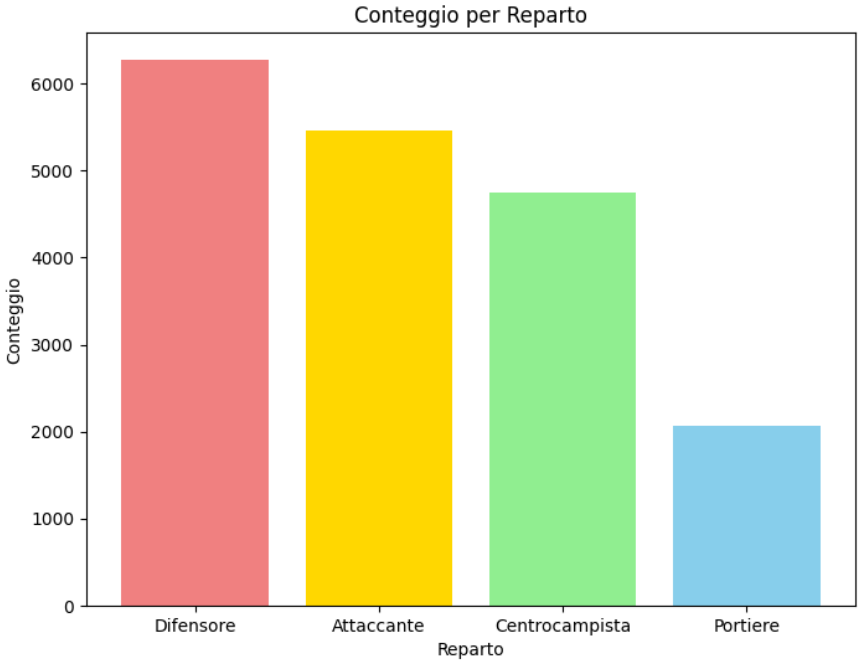
Considerando i primi 6 attributi con varianza spiegata maggiore, si riesce ad ottenere una varianza superiore 90%. Dunque, abbiamo applicato la PCA al dataset con un numero di componenti finali fissato a 6.

4.2. Analisi dei reparti

In generale, calciatori simili tecnicamente, giocano nello stesso reparto. I reparti del calcio sono 4: portiere, difensore, centrocampista e attaccante. Ad ogni reparto, corrispondono una serie di ruoli. Ad esempio, al reparto difesa, appartengono i ruoli *CB* (difensore centrale), *LB/RB* (terzino sinistro e destro), *LWB/RWB* (ala difensiva sinistra e destra).

Come prima cosa, abbiamo effettuato il conteggio dei calciatori per posizione e per reparto. Il ruolo con più calciatori nel dataset è *CB*, ovvero il difensore centrale, quello con meno occorrenze è *CF* (seconda punta). Il reparto con più occorrenze è il centrocampo, quello con meno occorrenze è la porta.

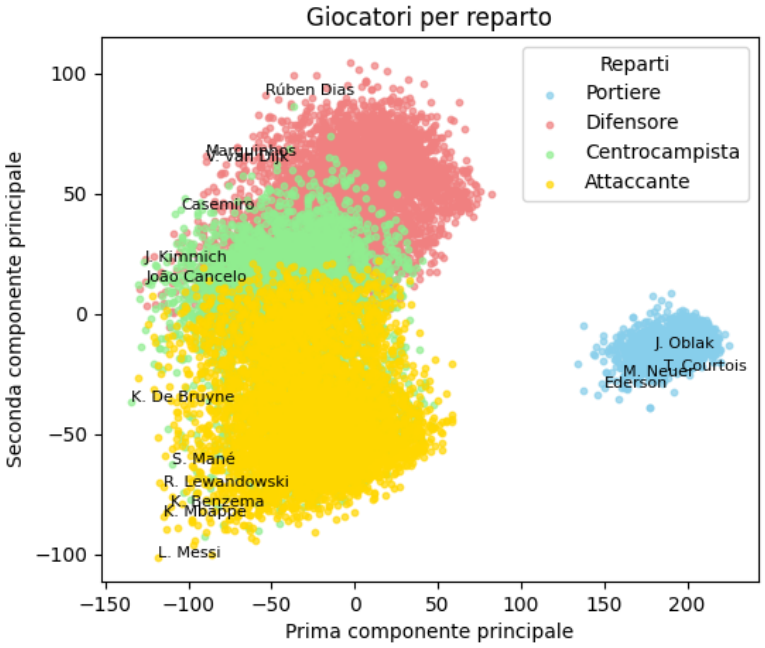


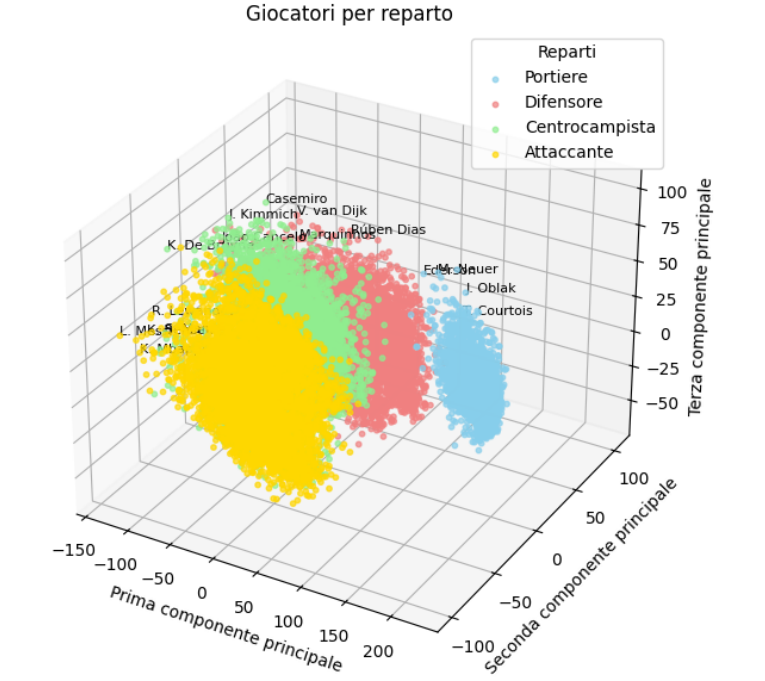


Andando ad utilizzare le prime due componenti principali, che insieme ci permettono di ottenere una varianza cumulativa del 77%, abbiamo realizzato un grafico bidimensionale, in modo da rappresentare la posizione di tutti i calciatori del dataset in base alle due componenti. Per ogni reparto, inoltre, sono stati rappresentati i 4 migliori calciatori.

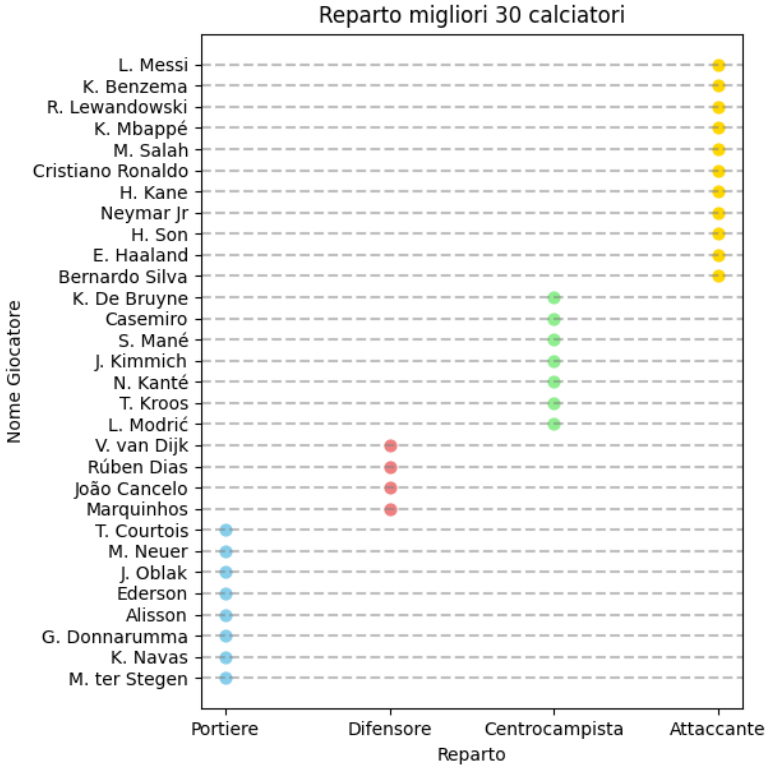
Con la stessa tecnica, abbiamo realizzato un grafico tridimensionale, che considera le prime tre componenti principali, con la quale si raggiunge una varianza cumulativa dell'83%.

Si può subito notare come i portieri abbiano valori totalmente differenti ai calciatori "di movimento". L'altra cosa da notare è la sovrapposizione dei reparti di movimento: i centrocampisti arrivano a ricoprire tutta la sezione degli attaccanti e parte della sezione dei difensori. Essi, infatti, si differenziano in "difensivi", se tendono ad assomigliare a difensori, come *Casemiro* e *J. Kimmich* mostrati nel grafico, "offensivi", se tendono ad assomigliare agli attaccanti, come *S. Mané* e *K. De Bruyne*. Esiste poi un sottogruppo di difensori, che è dotato di buone qualità tecniche, paragonabili a quelle dei centrocampisti, come ad esempio *Joao Cancelo*.


Giocatori per reparto
Reparti
Portiere
Difensore
Centrocampista
Attaccante
Rúben Dias
Marquinhos
V. van Dijk
Casemiro
J. Kimmich
João Cancelo
K. De Bruyne
S. Mané
R. Lewandowski
K. Benzema
K. Mbappé
L. Messi
J. Oblak
T. Courtois
M. Neuer
Ederson
Seconda componente principale
Prima componente principale


Giocatori per reparto
Reparti
Portiere
Difensore
Centrocampista
Attaccante
Casemiro
J. Kimmich
V. van Dijk
Rúben Dias
J. Oblak
Courtois
Terza componente principale
Seconda componente principale
Prima componente principale

Selezionando i primi 30 calciatori del dataset, ovvero i 30 più forti, abbiamo prodotto un grafico in cui sono rappresentati da un pallino e posizionati sull'asse relativo al proprio reparto.



4.3. Metriche di valutazione

Prima di parlare degli algoritmi utilizzati, descriviamo le metriche di confronto degli algoritmi di clustering utilizzate. Ogni metrica utilizzata effettua una comparazione tra i reparti di gioco dei calciatori, che costituiscono le **etichette** reali, e le **assegnazioni** effettuate dagli algoritmi, ovvero i reparti al quale sono stati classificati.

4.3.1 Rand Index (RI)

L'Indice di Rand è una misura di similarità tra due raggruppamenti, le etichette e le assegazioni in questo caso. Può essere influenzato dalla casualità, il che significa che potrebbe dare risultati significativi anche quando si confrontano raggruppamenti casuali. L'RI restituisce un valore compreso tra 0 e 1, dove 0 indica nessuna similarità e 1 indica una corrispondenza perfetta tra i raggruppamenti.

4.3.2 Adjusted Rand Index (ARI)

L'ARI è una versione corretta dell'Indice di Rand che tiene conto della casualità che potrebbe verificarsi anche con raggruppamenti casuali. Esso modifica l'Indice di Rand per correggere i risultati che potrebbero essere ottenuti casualmente e normalizza il punteggio in modo che il risultato casuale abbia uno score vicino a zero. Restituisce un valore compreso tra -1 e 1, dove -1 indica che i raggruppamenti sono completamente diversi, 0 indica un risultato casuale e 1 indica una corrispondenza perfetta.

4.3.3 Fowlkes-Mallows Index (FMI)

L'FMI considera i *True Positive (TP)*, *False Positive (FP)* e i *False Negative (FN)*. Esso misura la precisione, ovvero i veri positivi su tutti i positivi calcolati (falsi positivi compresi), e il richiamo, ovvero i veri positivi su tutti i positivi reali (falsi negativi compresi). La formula è la seguente:

$$\sqrt{\frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}}$$

Può variare da 0 a 1, dove 0 indica nessuna corrispondenza, 1 indica una corrispondenza perfetta.

4.4. K-means clustering

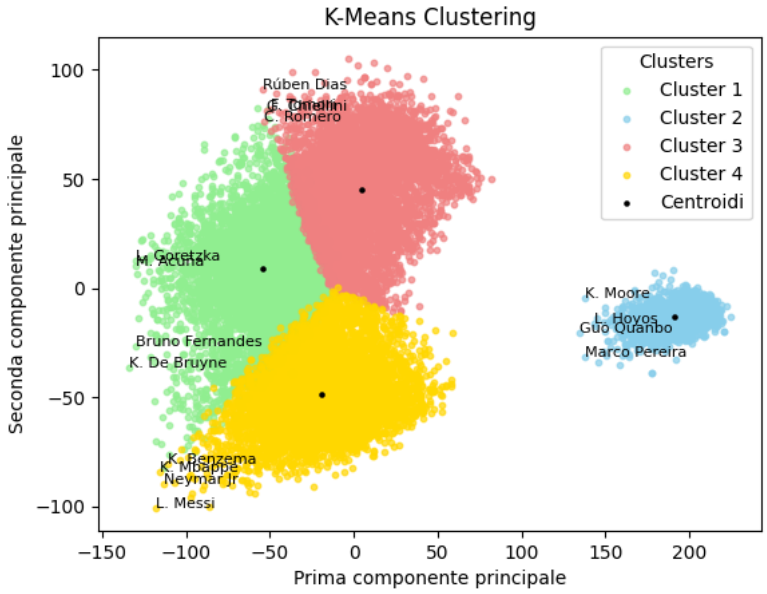
Il primo algoritmo di clustering applicato al dataset creato con la PCA è il ***k-means clustering***. Il numero di clustering è fissato a 4, come i reparti di gioco. In questo modo possiamo confrontare il grafico bidimensionale e tridimensionale dei reparti con quelli prodotti dal k-means.

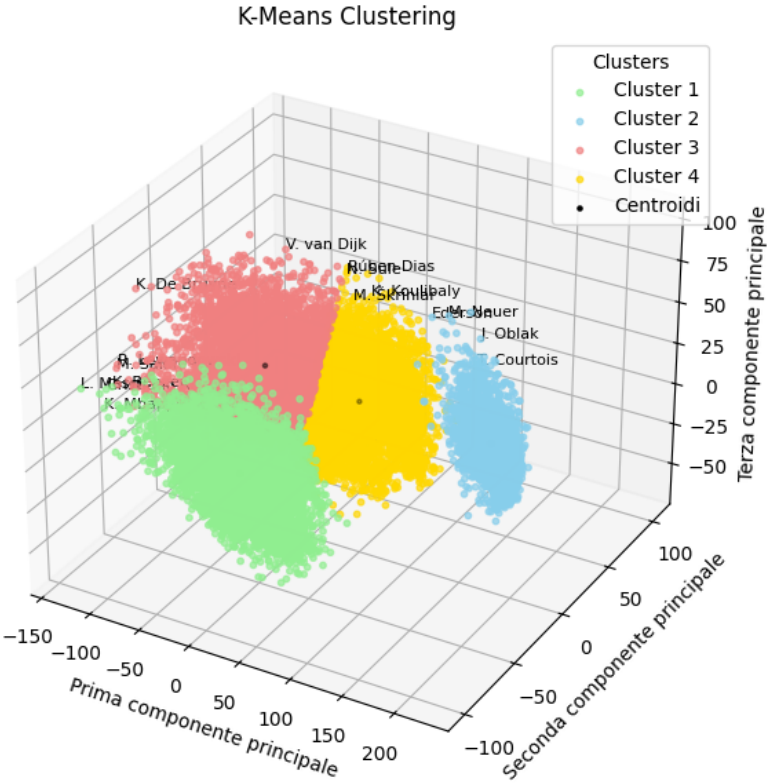
L'algoritmo funziona bene nel classificare il reparto dei portieri, ma per il gruppo dei calciatori di movimento tende a creare cluster di uguale dimensione. Questo è causato dal fatto che viene effettuato iterativamente il calcolo dei quattro centroidi per poi raggruppare gli elementi in base alla distanza euclidea con il centroide.

Indice	Punteggio
Rand Index (RI)	0.76
Adjusted Rand Index (ARI)	0.41
Fowlkes Mallows Index (FMI)	0.57

Un RI di 0.76 può essere considerato abbastanza alto e suggerisce una buona similarità tra i raggruppamenti ottenuti dall'algoritmo e quelli reali di riferimento. Però, allo stesso tempo, un ARI di 0.4 è relativamente basso e potrebbe suggerire che parte della somiglianza tra i raggruppamenti potrebbe essere attribuita alla casualità.

Un FMI di 0.55, infine, indica una corrispondenza moderata tra i cluster predetti e le etichette di classe reali. È superiore a 0.5, suggerendo che c'è una certa coerenza nei risultati della predizione. Questo punteggio suggerisce che il modello è bilanciato tra la capacità di predire correttamente i veri positivi e la capacità di evitare i falsi positivi e falsi negativi.


K-Means Clustering
Clusters
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Centroidi
Rúben Dias
C. Romero
M. Acuña
Bruno Fernandes
K. De Bruyne
K. Benzema
K. Mbappé
Neymar Jr
L. Messi
K. Moore
L. Hoyos
Guo Quanbo
Marco Pereira
Seconda componente principale
Prima componente principale


K-Means Clustering
Clusters
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Centroidi
V. van Dijk
Rúben Dias
K. Koulibaly
M. Skriniar
J. Oblak
T. Courtois
Prima componente principale
Seconda componente principale
Terza componente principale

4.5. Hierarchical Clustering

Il clustering basato sui centroidi del k-means tende a formare cluster di uguale dimensione. Un approccio di clustering differente lo si ha con il ***Clustering gerarchico*** (*hierarchical clustering).* I calciatori vengono organizzati in un albero chiamato **dendogramma** sulla base delle loro somiglianze in modo iterativo. In particolare, abbiamo utilizzato l'approccio **agglomerativo**, dove si parte dalla lista dei calciatori e si uniscono progressivamente.

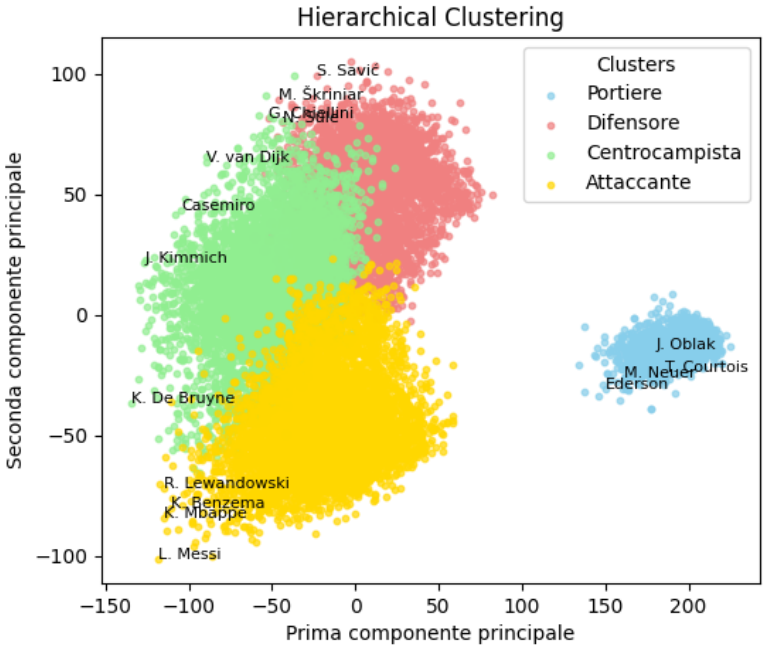
Come metodo di misurazione della distanza abbiamo utilizzato quello più comune, ovvero il metodo ***euclideo***. Tale scelta è data dai vari test effettuati con gli altri metodi che non hanno portato grossi miglioramenti.

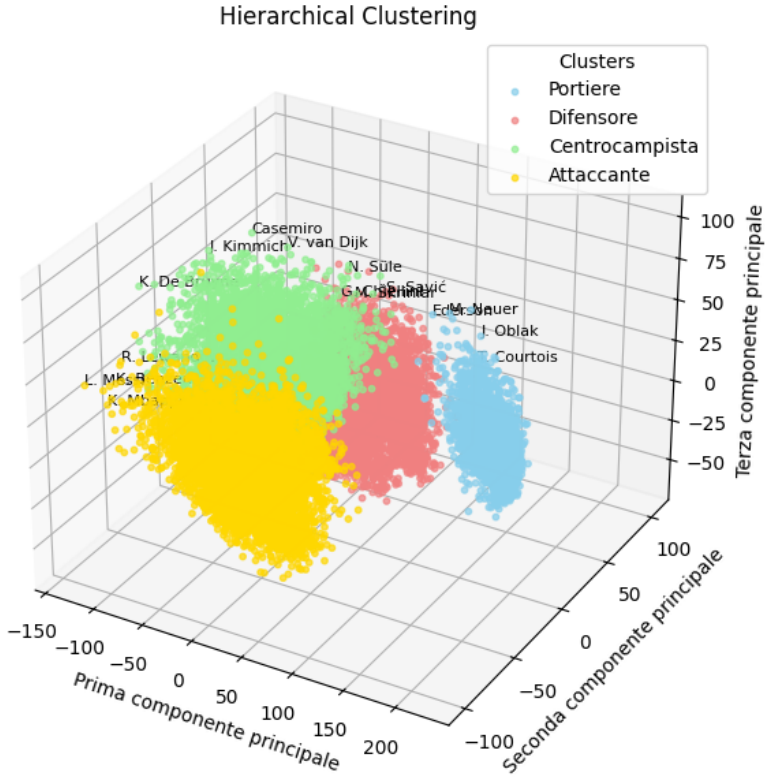
Come metodo di collegamento (***linkage***) dei cluster, abbiamo utilizzato il metodo ***ward***. Tale metodo, durante il processo di clustering agglomerativo, sceglie di unire i due cluster che portano alla minima variazione complessiva nella somma delle differenze quadrate. In altre parole, cerca di formare cluster in modo che la somma delle differenze quadrate tra i punti di ciascun cluster e il loro centroide sia minima.

Indice	Punteggio
Rand Index (RI)	0.75
Adjusted Rand Index (ARI)	0.37
Fowlkes Mallows Index (FMI)	0.55

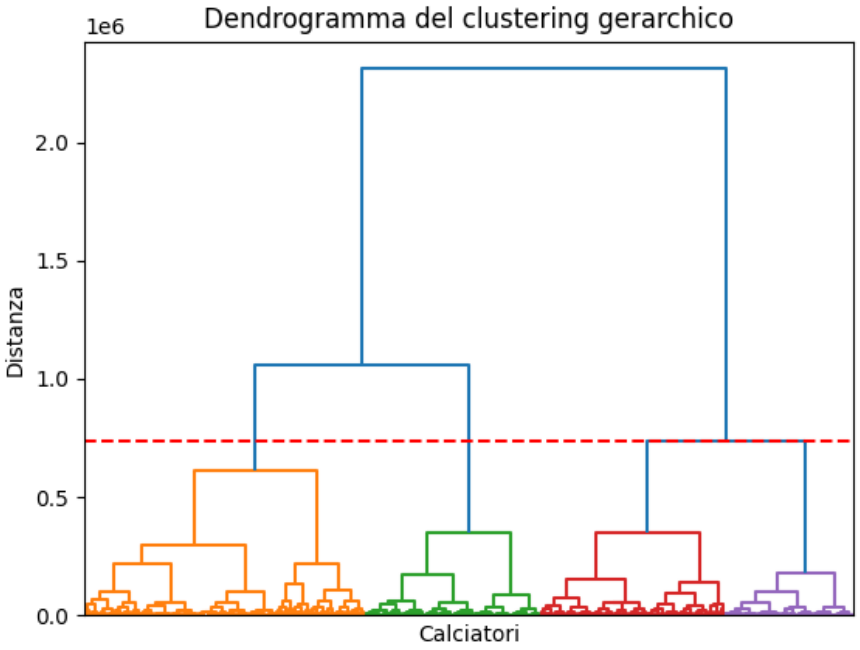
I punteggi ottenuti sono leggermente inferiori a quelli del k-means clustering. Nonostante le varie metriche di distanza e metodi di collegamento testati, non si riescono ad ottenere punteggi migliori.

La differenza più sostanziosa sta nell'ARI. Questo indica che, nell'hierarchical clustering, c'è ancora più casualità nella classificazione dei calciatori rispetto al k-means clustering, nonostante l'algoritmo non si basi sulla distanza dai centroidi.


Hierarchical Clustering
Clusters
Portiere
Difensore
Centrocampista
Attaccante
S. Savić
M. Škriniar
N. Süle
V. van Dijk
Casemiro
J. Kimmich
K. De Bruyne
R. Lewandowski
K. Benzema
K. Mbappé
L. Messi
J. Oblak
T. Courtois
M. Neuer
Ederson
Seconda componente principale
Prima componente principale


Hierarchical Clustering
Clusters
Portiere
Difensore
Centrocampista
Attaccante
Casemiro
V. van Dijk
N. Süle
J. Oblak
Courtois
Terza componente principale
Prima componente principale
Seconda componente principale

Per l'algoritmo dell'hierarchical clustering è stato possibile anche realizzare un grafico differente, il ***dendogramma***. Il grafico mostra come l'algoritmo parte dalla base, ovvero dall'elenco dei calciatori, e li unisca a due a due fino ad arrivare ad ottenere 4 cluster.



4.6. Considerazioni finali

Il dataset è composto da un numero di calciatori elevato e, i sui calciatori meno forti, che sono meno conosciuti, si hanno dei valori più distorti per gli attributi riguardanti caratteristiche tecniche. Tagliando il dataset e considerando soltanto i 200 calciatori più forti si sono ottenuti risultati migliori. L'immagine che segue mostra i punteggi ottenuti dal k-means clustering prima, quelli dell'hierarchical clustering dopo.

Indice	Punteggio
Rand Index (RI)	0.81
Adjusted Rand Index (ARI)	0.52
Fowlkes Mallows Index (FMI)	0.66

Indice	Punteggio
Rand Index (RI)	0.8
Adjusted Rand Index (ARI)	0.51
Fowlkes Mallows Index (FMI)	0.65

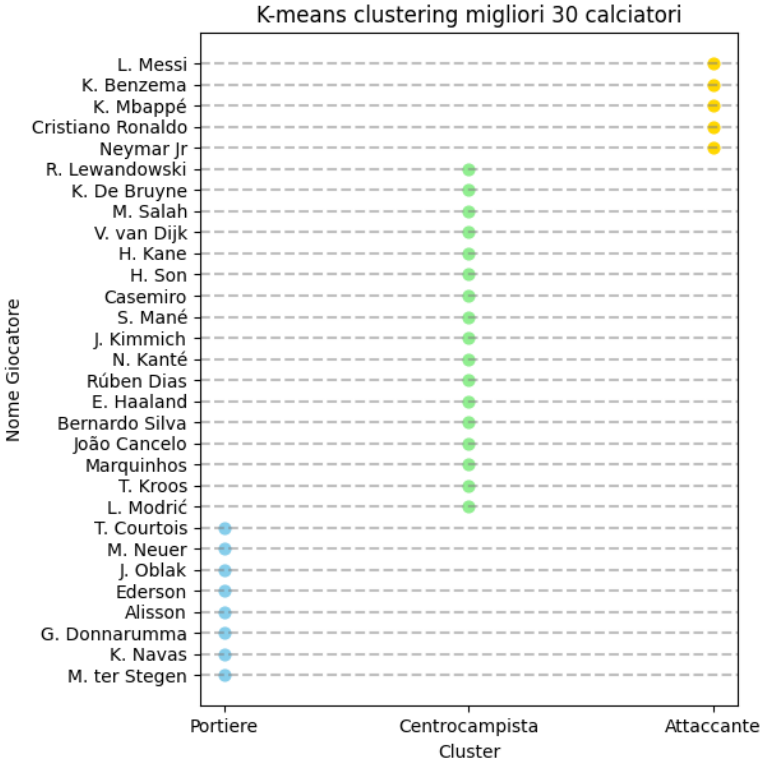
I punteggi sono migliorati ma non sono perfetti. Un'altra causa della corrispondenza imperfetta tra reparto e classificazione potrebbe essere nella diversità dei vari ruoli dello stesso reparto. Ad esempio, esistono dei centrocampisti più difensivi, altri più offensivi, che potrebbero essere classificati male. Un altro motivo potrebbe essere nelle caratteristiche individuali dove il singolo calciatore spicca. Ad esempio, di solito, i calciatori più bravi tecnicamente o a calciare le punizioni, sono i centrocampisti. Tuttavia, alcuni tiratori si possono trovare anche in reparti differenti, come in difesa o attacco.

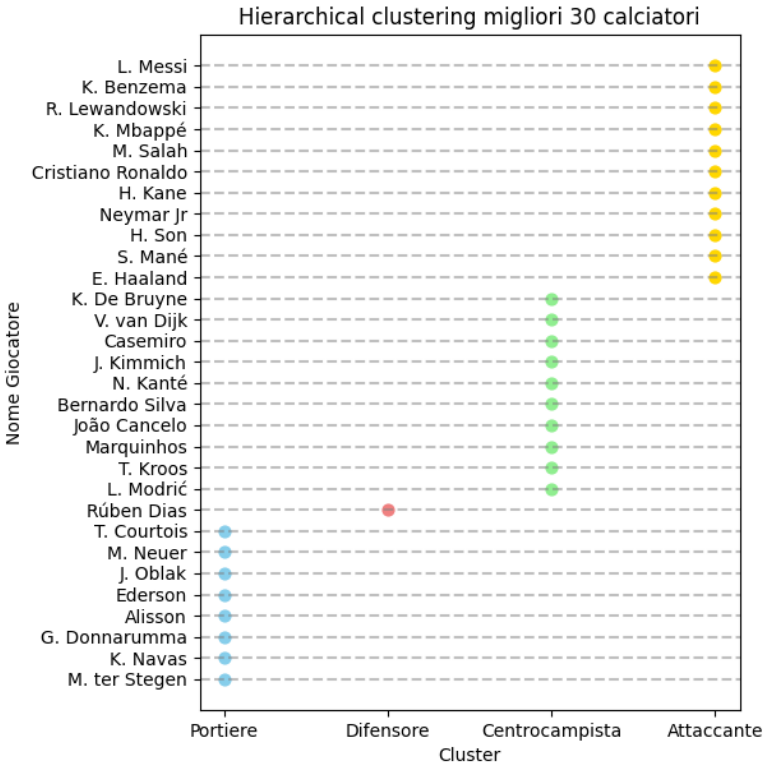
Vista la vastità del dataset e delle diversità che non possono essere considerate dall'algoritmo, i risultati ottenuti possono essere ritenuti soddisfacenti.

Per visualizzare come sono stati classificati i calciatori attraverso i due algoritmi, abbiamo realizzato i seguenti grafici dove vengono mostrati i 30 calciatori più forti nei cluster corrispondenti, come nell'analisi dei reparti. Il primo grafico si riferisce al k-means clustering, il secondo all'hierarchical clustering.

Il k-means clustering, tra i primi 30 calciatori, non classifica nemmeno un difensore. Tutti i difensori sono stati classificati come centrocampisti. Questo può essere dato dal fatto che, i difensori più forti al mondo, hanno buone qualità tecniche, paragonabili a quelle dei centrocampisti. Ci sono altri errori riguardanti attaccanti classificati come centrocampisti (ad esempio, *R. Lewandowski*).

L'hierarchical clustering effettua molti meno errori sui primi 30 calciatori, anche se sono sempre presenti difensori classificati come centrocampisti. La miglior classificazione è data dal fatto che il clustering gerarchico si basa sulle distanze tra le componenti principali dei calciatori e non dalle distanze dai centroidi.


K-means clustering migliori 30 calciatori
L. Messi
K. Benzema
K. Mbappé
Cristiano Ronaldo
Neymar Jr
R. Lewandowski
K. De Bruyne
M. Salah
V. van Dijk
H. Kane
H. Son
Casemiro
S. Mané
J. Kimmich
N. Kanté
Rúben Dias
E. Haaland
Bernardo Silva
João Cancelo
Marquinhos
T. Kroos
L. Modrić
T. Courtois
M. Neuer
J. Oblak
Ederson
Alisson
G. Donnarumma
K. Navas
M. ter Stegen
Nome Giocatore
Portiere
Centrocampista
Attaccante
Cluster


Hierarchical clustering migliori 30 calciatori
L. Messi
K. Benzema
R. Lewandowski
K. Mbappé
M. Salah
Cristiano Ronaldo
H. Kane
Neymar Jr
H. Son
S. Mané
E. Haaland
K. De Bruyne
V. van Dijk
Casemiro
J. Kimmich
N. Kanté
Bernardo Silva
João Cancelo
Marquinhos
T. Kroos
L. Modrić
Rúben Dias
T. Courtois
M. Neuer
J. Oblak
Ederson
Alisson
G. Donnarumma
K. Navas
M. ter Stegen
Nome Giocatore
Portiere
Difensore
Centrocampista
Attaccante
Cluster

4.7. Analisi similarità

La PCA effettuata nella fase iniziale potrebbe essere utilizzata per cercare all'interno del dataset un elenco dei calciatori più simili ad un determinato calciatore in un modo più rapido. Sulla base dei 6 attributi principali selezionati, dato un calciatore, sono state calcolate le **distanze euclidee** con tutti i calciatori del dataset.

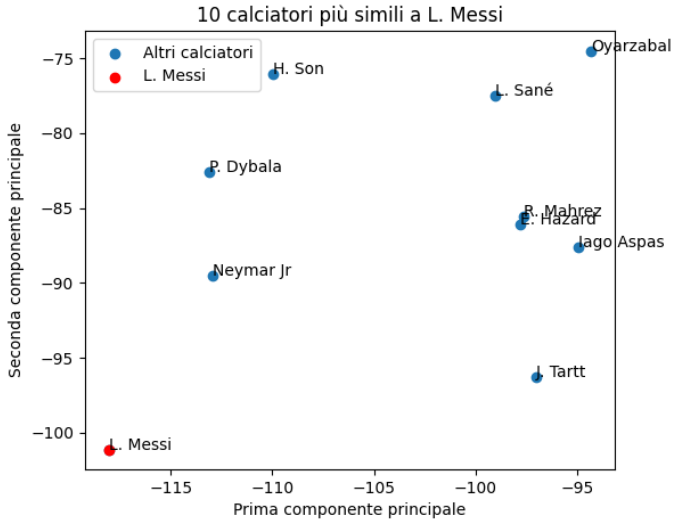
Per distanza euclidea tra due giocatori, si intende la distanza geometrica in uno spazio di 6 dimensioni e dove le coordinate dei calciatori sono le 6 componenti principali selezionate dalla PCA. Se a è il calciatore del quale trovare i più simili, e b è il calciatore da confrontare per il calcolo della distanza, la formula utilizzata è la seguente:

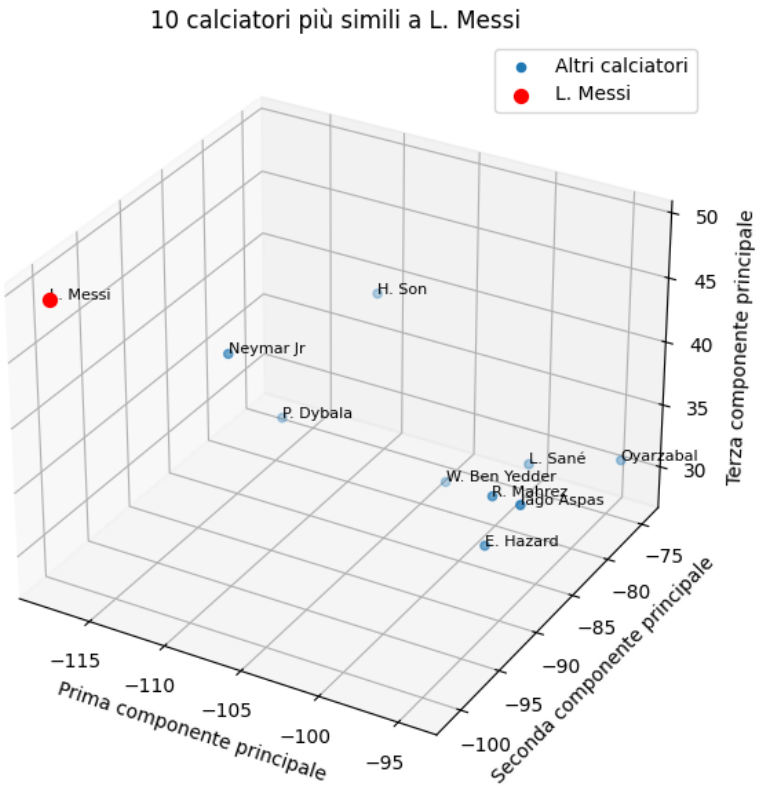
$$\sqrt{(PC1a - PC1b)^2 + (PC2a - PC2b)^2 + ... + (PC6a - PC6b)^2}$$

Successivamente abbiamo ordinato il dataset in ordine crescente per selezionare le prime 10 tuple, ovvero, i 10 calciatori più simili.

Per visualizzare graficamente i risultati dell'analisi della similarità, abbiamo realizzato un grafico simile a quello del k-means, bidimensionale e tridimensionale.

Come calciatore di esempio abbiamo scelto *L. Messi*, rappresentato da un colore differente dagli altri. Il calciatore selezionato è, per distacco, il migliore al mondo, dunque, anche i 10 calciatori più simili tecnicamente sono lontani.


10 calciatori più simili a L. Messi
Altri calciatori
L. Messi
Oyarzabal
H. Son
L. Sané
P. Dybala
R. Mahrez
E. Hazard
Iago Aspas
Neymar Jr
I. Tartt
L. Messi
Seconda componente principale
Prima componente principale
−75
−80
−85
−90
−95
−100
−115
−110
−105
−100
−95


10 calciatori più simili a L. Messi
Altri calciatori
L. Messi
L. Messi
H. Son
Neymar Jr
P. Dybala
L. Sané
Oyarzabal
W. Ben Yedder
R. Mahrez
Iago Aspas
E. Hazard
Terza componente principale
50
45
40
35
30
Prima componente principale
−115
−110
−105
−100
−95
Seconda componente principale
−75
−80
−85
−90
−95
−100

5. KB

Lo scopo dell'ultimo task è quello di realizzare una base di conoscenza da poter interrogare attraverso la formulazione di query.

5.1. Estrapolazione dei fatti dal dataset

Come primo passaggio, considerando i primi 500 calciatori in modo da non produrre un numero troppo grande di regole, abbiamo estrapolato un insieme di 9 fatti per singolo calciatore, da utilizzare per la successiva produzione di regole. I fatti per ogni calciatore sono i seguenti:

- "*overall(Calciatore, Overall)*": indica appunto l'overall del calciatore;
- "*potenziale(Calciatore, Potenziale)*": indica il potenziale, cioè l'overall che potenzialmente potrebbe raggiungere;
- "*nazionalita(Calciatore, Nazionalita)*": la nazionalità;
- "*squadra(Calciatore, Squadra)*": la squadra in cui gioca;
- "*numero(Calciatore, Numero)*": il numero di maglia che indossa nella propria squadra;
- "*ruolo(Calciatore, Ruolo)*": il ruolo principale nel quale gioca;
- "*piede(Calciatore, Piede)*": il piede preferito;
- "*scadenza(Calciatore, Scadenza)*": l'anno di scadenza del contratto per la propria squadra;
- "*inizio(Calciatore, Inizio)*": l'anno nel quale ha iniziato a giocare per la propria squadra attuale;

5.2. Preprocessing

Tutte i dati di tipo stringa, prima di scrivere i fatti su un file Prolog, hanno subito un processo di normalizzazione per poter essere lette senza errori dal linguaggio Prolog. La normalizzazione ha effettuato i seguenti passaggi:

1. trasformazione di tutte le lettere in minuscolo;
2. rimozione di tutti i punti che abbreviano i nomi;
3. rimozione di tutti gli apostrofi;
4. sostituzione di tutti gli spazi con degli underscore;
5. rimozione dei trattini;
6. rimozione di tutti i tipi di accenti (attraverso la libreria *unidecode*).

Una volta creato il file Prolog contenente tutti i fatti, sono state aggiunte 9 clausole, una per ogni tipologia di fatto, per indicare che i fatti dello stesso tipo sono disposti in modo non contiguo, per evitare eventuali errori. I fatti sono del tipo ":- discontiguous overall/2.", ad esempio, per il fatto riguardante l'overall.

5.3. Produzione delle regole

Come ultimo passaggio, abbiamo realizzato un altro file prolog contenente tutte le regole che abbiamo deciso di implementare. Le regole prodotte sono le seguenti:

- "*reparto(nome_reparto, nome_posizione)*": per ogni posizione, in modo da mappare ogni posizione in un determinato reparto;
- "*numero_disponibile(Squadra, Numero)*": determina se un numero di maglia non è utilizzato da nessuno in una determinata squadra;
- "*stesso_reparto(GiocatoreA, GiocatoreB)*": determina se due calciatori hanno ruoli appartenenti allo stesso reparto di gioco;
- "*compagni_di_squadra(GiocatoreA, GiocatoreB)*": determina se due calciatori giocano nella stessa squadra;
- "*compagni_in_nazionale(GiocatoreA, GiocatoreB)*": dando per scontato che nei primi 500 calciatori che abbiamo considerato, siano tutti convocati nella propria nazionale, determina se i due calciatori giocano nella stessa nazionale;
- "*miglior_giocatore_squadra(Calciatore)*": determina se il calciatore è quello migliore, ovvero con overall maggiore, della propria squadra;
- "*miglior_giocatore_squadra_reparto(Calciatore)*": funziona in modo simile alla precedente, determinando però se il calciatore è il migliore del proprio reparto nella propria squadra;
- "*contratto_in_scadenza(Calciatore)*": determina se il calciatore ha un contratto che scade nell'anno corrente, ovvero il 2023;
- "*bandiera(Calciatore)*": determina se il calciatore è una bandiera per la propria squadra, dove per bandiera si intende un calciatore che gioca nella stessa squadra per almeno 10 anni (contando sempre a partire dall'anno corrente, ovvero il 2023);
- "*stesso_piede(GiocatoreA, GiocatoreB)*": determina se due calciatori hanno lo stesso piede preferito;
- "*in_crescita(Calciatore)*": determina se il calciatore ha margine di crescita, ovvero se il suo potenziale è maggiore dell'overall;
- "*doppiamente_compagni(GiocatoreA, GiocatoreB)*": determina se due calciatori sono contemporaneamente compagni di squadra e di nazionale.