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. Compiti effettuati

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 squadra titolare possibile per una nazionale presa in input o del mondo intero in base all'overall dei calciatori. La formazione si basa su un numero finito di moduli di gioco (ad esempio il 4-3-3) preso anch'esso in input. 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.

1.4. Strumenti utilizzati

Il progetto è stato sviluppato in python. Per quanto riguarda la task dell'apprendimento supervisionato, le librerie 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.

Per la task della risoluzione del CSP abbiamo utilizzato le seguenti librerie:

- **pyswip**: per interagire con file di Prolog;
- **pandas**: per gestire il dataset in formato "csv";
- **unidecode**: per gestire i nomi dei calciatori con accenti non leggibili dal Prolog;

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 **preprocessing** 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. Il risultato ottenuto è quello mostrato in Figura 1.

Overall	1.000000	Heading Accuracy	0.342452
Reactions	0.872789	Finishing	0.331967
Composure	0.700583	Penalties	0.331834
Shot Power	0.552518	Marking	0.313975
Vision	0.523290	Interceptions	0.311216
Short Passing	0.520784	Jumping	0.282737
LongPassing	0.507192	Standing Tackle	0.266683
BallControl	0.457705	Agility	0.254826
Curve	0.416575	Sliding Tackle	0.240499
Crossing	0.395435	Sprint Speed	0.213367
Aggression	0.393433	Acceleration	0.198758
Long Shots	0.392187	Balance	0.142749
Stamina	0.376994	Goalkeeper Positioning	-0.008318
Dribbling	0.376185	Goalkeeper Reflexes	-0.013110
Freekick Accuracy	0.374162	Goalkeeper Handling	-0.013655
Volleys	0.367168	Goalkeeper Diving	-0.016593
Strength	0.353857	Goalkeeper Kicking	-0.016764
Positioning	0.348038		

Figura 1: Correlazioni tra gli attributi e l'overall.

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 $y = x$ (Figura 2).
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 abbiano una correlazione negativa, ovvero inferiore allo 0. Questo risultato afferma che tali caratteristiche diminuiscono all'aumentare dell'overall, e viceversa.

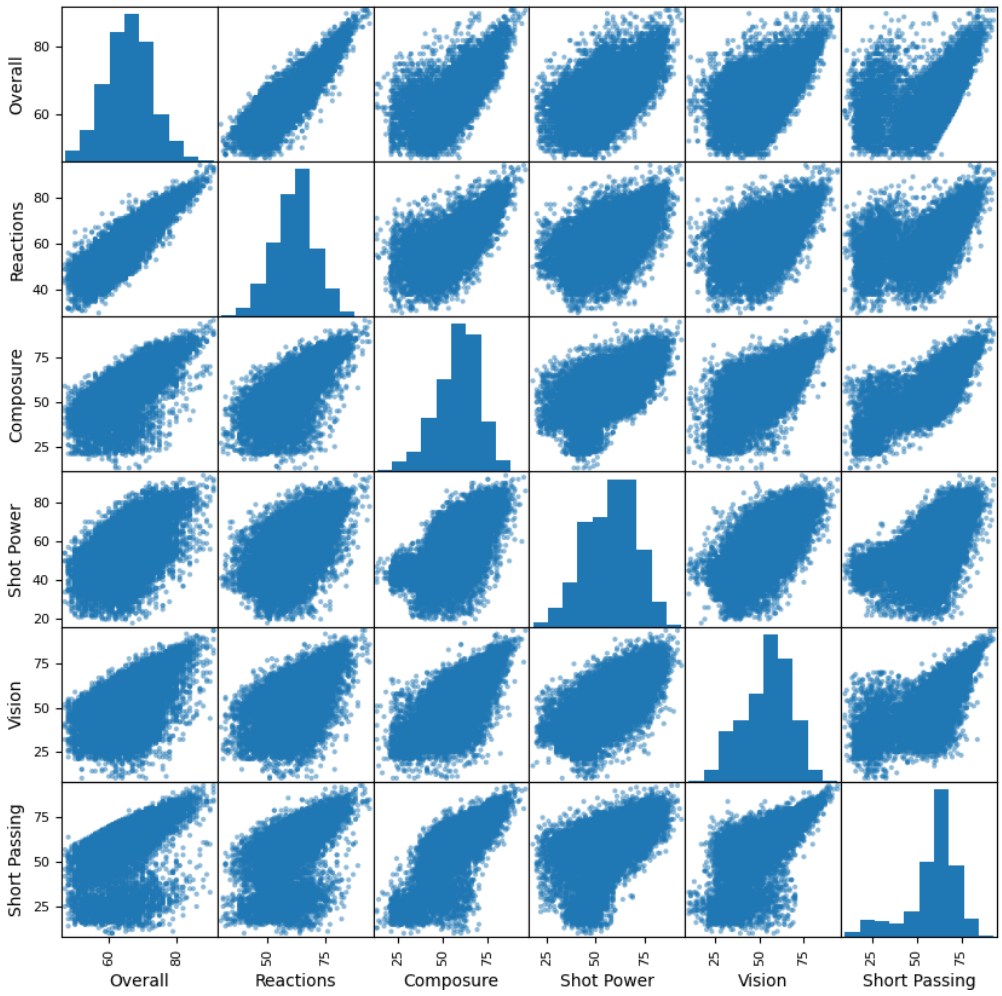

Figura 2: Matrice delle corellazioni che considera l'overall e i 5 attributi con correlazione maggiore.

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 apprendimento, 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.

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*.

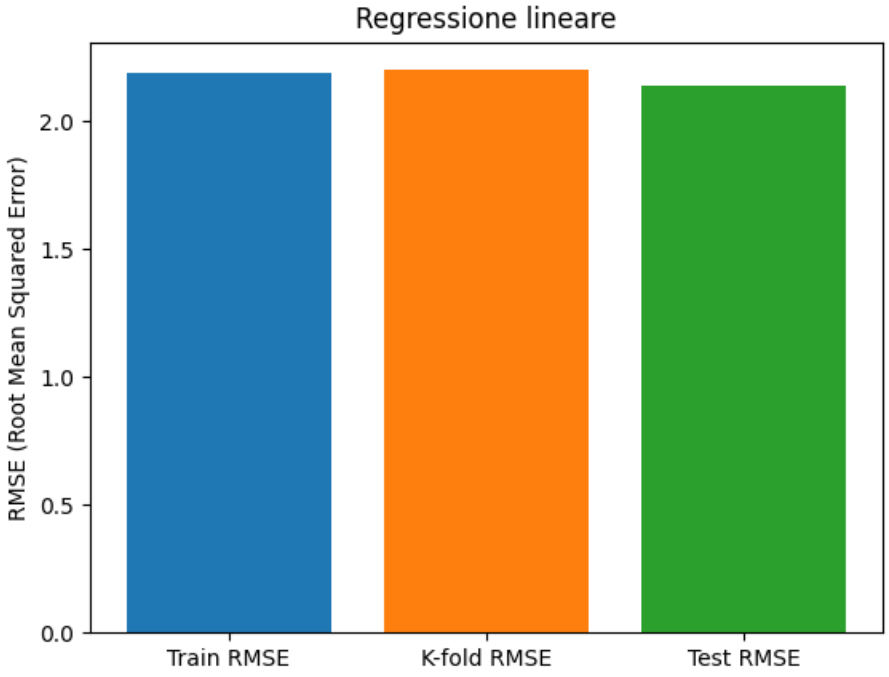

Figura 3: Risultati del modello di regressione lineare.

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 (linea blu Figura 4), 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 (linee arancione e verde Figura 4). La profondità massima dell'albero che ci fa ottenere il miglior punteggio è *10*, con un RMSE uguale a circa *1.9*.

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). I risultati ottenuti sono descritti dalla Figura 5.

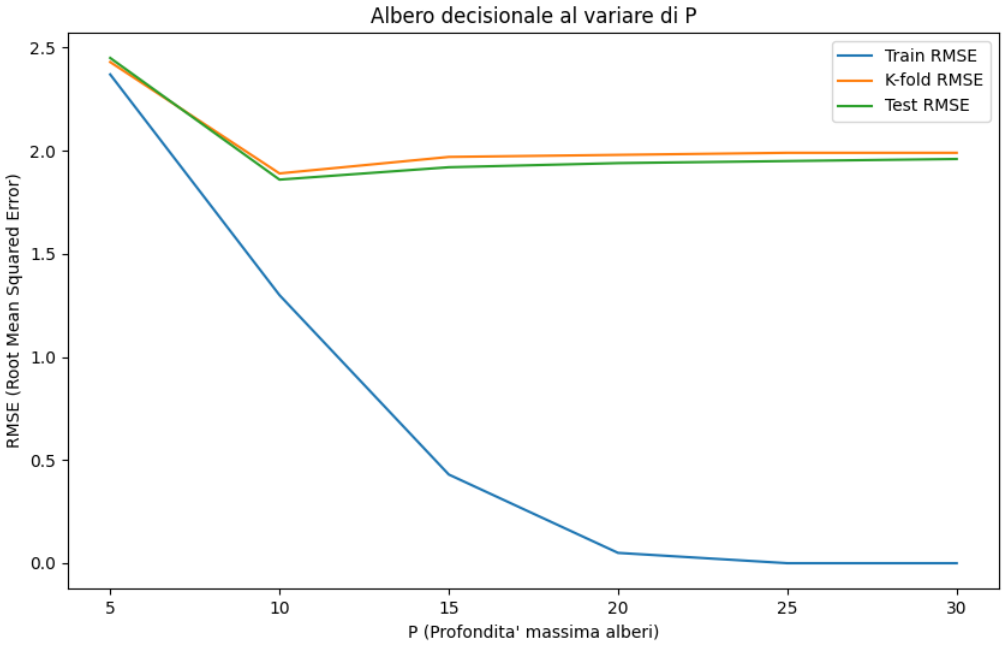

Figura 4: Confronto tra l'RMSE ottenuto in fase di training, le medie della k-fold cross validation applicata al training set e i risultati della valutazione, tutti applicati alle diverse profondità massime selezionate.

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. Ciò infatti provoca un alto RMSE. 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 (Figura 5).

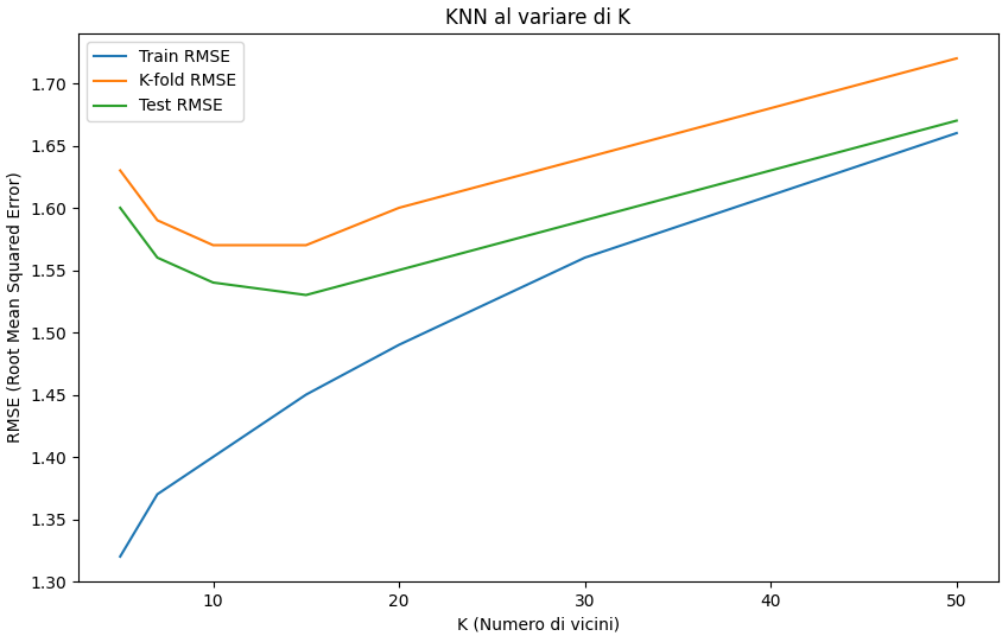

Figura 5: Confronto tra l'RMSE ottenuto in fase di training, le medie della k-fold cross validation applicata al training set e i risultati della valutazione, tutti applicati al diverso numero di vicini.

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 30 e ce l'aspettavamo poiché il modello del random forest aumenta l'adattabilità ai dati all'aumentare del numero di alberi.

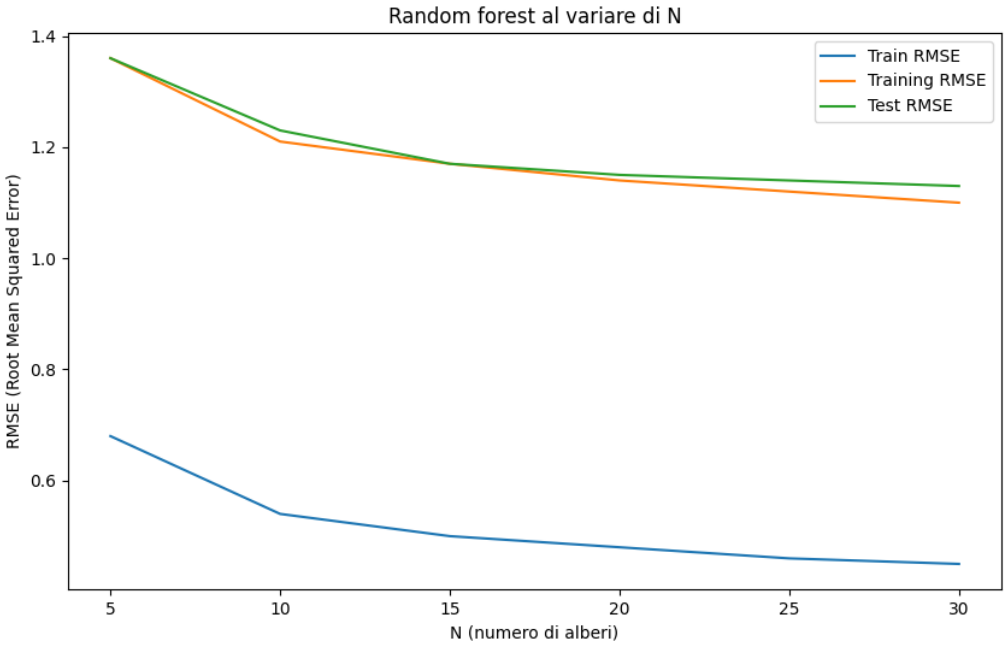

Figura 6: Confronto tra l'RMSE ottenuto in fase di training, le medie della k-fold cross validation applicata al training set e i risultati della valutazione, tutti applicati al diverso numero di alberi della foresta.

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 (Figura 7).

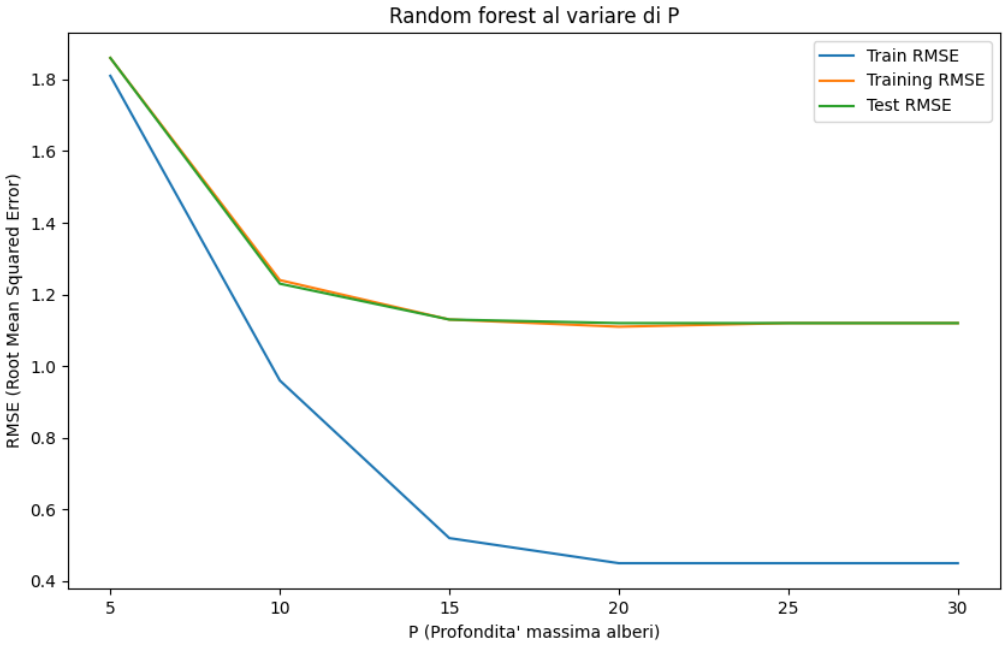

Figura 7: Confronto tra l'RMSE ottenuto in fase di training, le medie della k-fold cross validation applicata al training set e i risultati della valutazione, tutti applicati alle diverse profondità massime selezionate.

Conclusione: il miglior modello tra quelli testati è il random forest in quanto ha 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*.

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 (Figura 8). I valori degli iperparametri di *n_ estimators* e *max_ depth* sono rispettivamente 25 e 20.

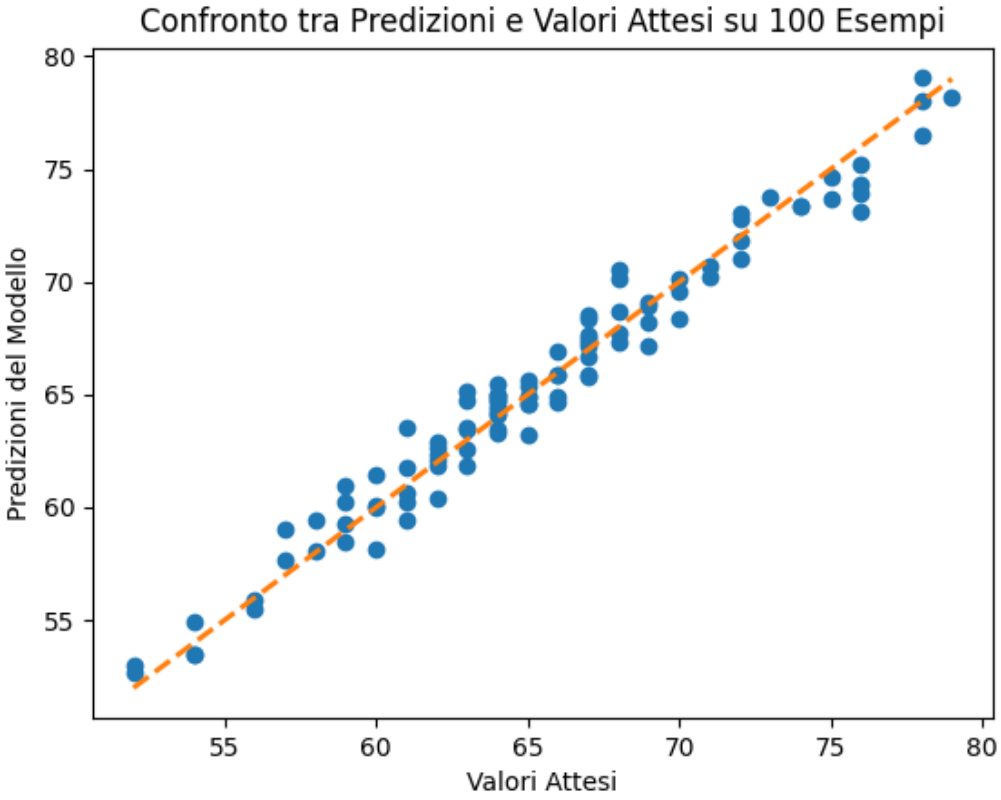

Figura 8: Grafico di dispersione delle previsioni confronto ai valori attesi per 100 esempi del test set.

3. CSP

Il CSP affrontato consiste nel cercare la miglior formazione per una nazionale presa in input o di tutto il mondo. Il software, dopo aver chiesto di quale nazionale effettuare il calcolo, chiederà anche il modulo di gioco, ovvero la disposizione dei ruoli della squadra sul campo. Ad ogni modulo corrisponde un fatto nella knowledge base.

Esempio 3.1. *Per il modulo 4-3-3, si ha il seguente fatto*

$$modulo(433, [gk, lb, cb, cb, rb, cdm, cm, cm, lw, rw, st]).$$

dove le sigle che seguono il nome del modulo indicano i nomi dei ruoli che ricoprono i calciatori, cioè goalkeeper (portiere) left/right back (terzino destro e sinistro), central back (difensore centrale), central defensive midfielder (mediano), central midfielder (centrocampista centrale), left/right winger (ala destra e sinistra) e striker (prima punta).