Documentazione progetto di Ingegneria della conoscenza

Beat Saber PP Maximizer

AA 2025-26

https://github.com/MMauro11/Beat-Saber-PP-Maximizer---IngegneriaDellaConoscenza

Introduzione

Beat Saber è un videogioco ritmico in realtà virtuale (VR), il giocatore impugna due spade laser virtuali e deve recidere una serie di blocchi cubici che scorrono verso di lui a ritmo di musica. panorama competitivo di Beat Saber gravita attorno a **ScoreSaber**, una piattaforma di ranking globale gestita dalla community ma riconosciuta come standard *de facto*. ScoreSaber assegna a ogni giocata completata con successo un punteggio in **PP (Performance Points)**. Questo sistema di punteggio è non lineare e non noto al pubblico, viene anche modificato saltuariamente dagli sviluppatori. La somma pesata dei PP delle migliori giocate di un utente determina il suo **Rank** globale (es. #1, #100, #5000).

Sommario

In seguito ad una accurata selezione di features durante la costruzione del dataset, il sistema esamina i dati storici delle performance dei giocatori in tutto il range della leaderboard globale e le caratteristiche strutturali delle mappe di gioco per determinare la tipologia di mappa, un fattore che determina le skill necessarie per vincere la partita. In seguito è stata costruita una Knowledge Base che è stata utilizzata in particolare per inferire nuove features molto utili alle fasi successive. In seguito è stato quindi utilizzato l'apprendimento supervisionato per compiti di regressione e prevedere accuratamente il valore PP (Performance Points): il punteggio che si ottiene in base alla performance della partita e che serve a scalare la leaderboard competitiva. Insieme queste tecniche hanno permesso infine di elaborare un **piano di allenamento** (training plan) ottimizzato per ogni giocatore (indicando il proprio rank attuale) e in base alla previsione della performance del giocatore tale piano è ottimizzato per massimizzare il guadagno potenziale di punti competitivi PP, suggerendo una sequenza di mappe selezionate in cui ci si aspetta che quel giocatore riesca a dare la migliore performance anche in mappe non facili da affrontare per lui.

Elenco argomenti di interesse

- **Costruzione del dataset, Features selection, preprocessing:** Utilizzo di query ad API selezionando apposite features per la costruzione del dataset su cui si baserà il KBS.
- **Apprendimento Non Supervisionato:** Utilizzo di clustering per la categorizzazione automatica delle mappe di gioco, prescindendo dall'utilizzo di etichette predefinite.
- **Rappresentazione della Conoscenza e Ragionamento:** Integrazione di un motore inferenziale logico (Prolog) finalizzato alla deduzione di nuove caratteristiche semantiche non esplicite nei dati grezzi.
- **Apprendimento Supervisionato:** Sviluppo e validazione di modelli predittivi (Regressione) per la stima quantitativa del punteggio competitivo ottenibile in base alla performance dei giocatori.
- **Problemi di Soddisfacimento dei Vincoli:** Modellazione e risoluzione di un problema di ottimizzazione combinatoria per la generazione di piani di allenamento personalizzati.

Sezione 1) Costruzione del dataset e preprocessing

Librerie Utilizzate: `pandas`, `numpy`, `requests` (per le chiamate API).

Acquisizione Dati e pulizia

Il processo di acquisizione dati è stato condotto interrogando le API pubbliche di **ScoreSaber** e **BeatSaver**. Inizialmente, è stato raccolto un campione di giocatori distribuiti lungo tutto il range di Rank della leaderboard globale per garantire variabilità. Per ogni giocatore, è stato poi estratto lo storico delle migliori performance. Successivamente, per ogni mappa presente nello storico, è stata interrogata l'API di BeatSaver per ottenerne i metadati tecnici (BPM, durata, note). È stato quindi creato un dataset unificato effettuando un'operazione di *join* tra le performance dei giocatori e i metadati delle mappe, utilizzando il `songHash` come chiave primaria.

Decisioni di Progetto

Durante questa fase, sono state adottate una serie di decisioni progettuali mirate a garantire la qualità e la coerenza del dataset per le fasi successive:

1. **Filtro Difficoltà "Expert":** È stato deciso di filtrare il dataset mantenendo esclusivamente le istanze relative a mappe giocate alla difficoltà "Expert" (valore numerico 7 nelle API). Questa scelta è stata cruciale per la normalizzazione dei dati. Confrontare punteggi ottenuti su difficoltà diverse (es. "Hard" vs "Expert+") avrebbe introdotto un bias che avrebbe reso impossibile isolare l'influenza delle caratteristiche strutturali (come BPM e note) sulla performance del giocatore. Normalizzando sulla difficoltà "Expert", i dati sono stati resi comparabili.
2. **Rimozione Dati Inconsistenti:** È stata effettuata una pulizia dei dati eliminando le righe che presentavano valori mancanti (NaN) in feature critiche per l'analisi, garantendo l'integrità del training set.
3. **Filtro Failure Cases (PP = 0):** Sono state rimosse dal dataset di addestramento tutte le registrazioni in cui i PP assegnati erano pari a 0. Nel contesto di ScoreSaber, un valore di 0 PP indica tipicamente un fallimento del livello (non completato) o un punteggio così basso da non essere considerato classificabile. Questi dati rappresentano degli *outlier* o rumore che avrebbero inquinato l'addestramento dei modelli di regressione, i quali avrebbero faticato a generalizzare una funzione di predizione coerente.

Dataset Generati e Descrizione delle Feature

Durante le varie fasi del progetto sono stati prodotti e raffinati diversi dataset intermedi. Di seguito viene riportato l'elenco dei file generati con la descrizione puntuale delle feature in essi contenute.

1. `players_scores.csv`

Contiene i dati grezzi relativi alle performance dei giocatori scaricati da ScoreSaber.

- **player_id:** Identificativo univoco del giocatore sulla piattaforma (chiave primaria).
- **player_rank:** Posizione del giocatore nella classifica globale al momento dell'estrazione.

- **songHash:** Hash univoco che identifica la mappa giocata.
- **base_score:** Punteggio numerico assoluto ottenuto nella partita.
- **accuracy:** Precisione del taglio in percentuale (valore da 0 a 100).
- **pp:** Punti competitivi (Performance Points) assegnati per quella specifica giocata.

2. `maps_data.csv`

Contiene i metadati tecnici delle mappe scaricati da BeatSaver.

- **songHash:** Hash univoco della mappa (chiave di join).
- **bpm:** Battiti per minuto della traccia musicale.
- **duration:** Durata della mappa in secondi.
- **notes:** Numero totale di blocchi colpibili presenti nella mappa.

3. `map_performance.csv`

Risultato del join tra `players_scores.csv` e `maps_data.csv`. Questo è il dataset base su cui sono state applicate le pulizie iniziali. Contiene tutte le feature sopra elencate combinate in un'unica struttura.

4. `maps_clustered.csv`

Dataset generato al termine della Fase di Clustering (Sezione 2). Oltre alle feature precedenti, include le etichette assegnate dall'algoritmo K-Means.

- **map_cluster:** Identificativo numerico del cluster di appartenenza (0, 1, 2, 3).
- **cluster_name:** Etichetta semantica del cluster ("HighSpeed", "Warm-up", "Balanced", "Extreme"), assegnata in base all'analisi dei centroidi.

5. `maps_enhanced.csv` (Dataset Finale)

Dataset finale utilizzato per l'addestramento dei modelli di Machine Learning. È stato arricchito con le feature logiche inferite tramite Prolog (Sezione 3).

- **avg_score_in_cluster:** Punteggio medio storico ottenuto dal giocatore specifico su mappe appartenenti a quel determinato cluster. Questa feature cattura l'abilità settoriale del giocatore.
- **avg_acc_in_cluster:** Accuratezza media storica del giocatore su quel determinato cluster.

Sezione 2) Apprendimento Non Supervisionato

Sommario

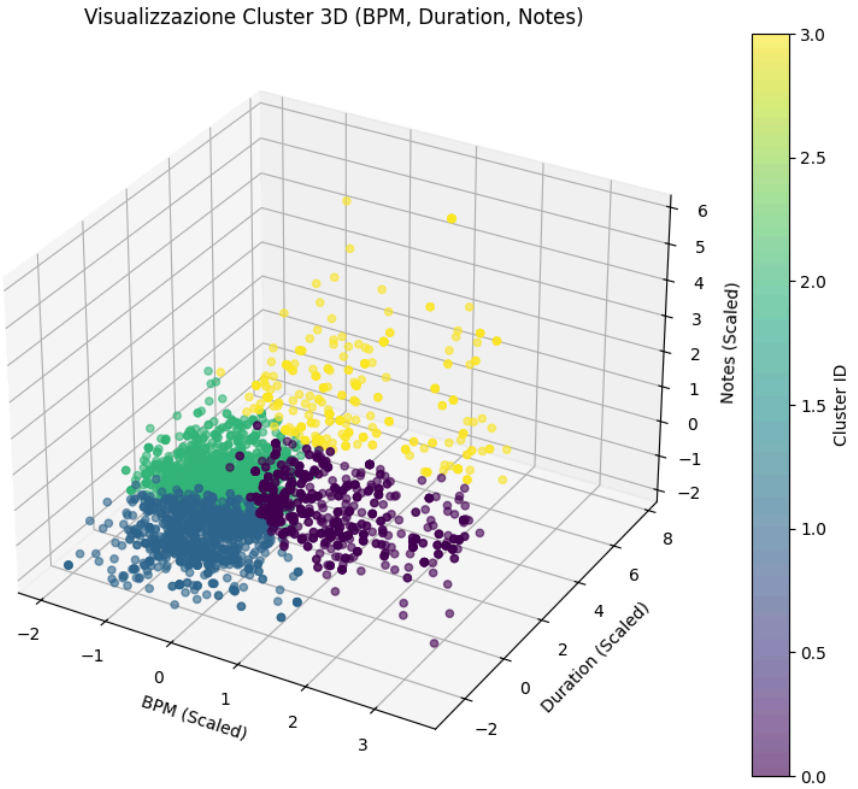
Per comprendere la struttura latente del dominio e categorizzare le mappe in modo oggettivo, sono state applicate tecniche di apprendimento non supervisionato. L'obiettivo era raggruppare le mappe non in base a etichette soggettive (come il genere musicale), ma in base alle loro caratteristiche tecniche fisiche.

Strumenti:

Librerie utilizzate: `scikit-learn` (KMeans), `kneed` (per il metodo del gomito), `matplotlib` (grafici 3D e curve), `pandas`, `numpy`.

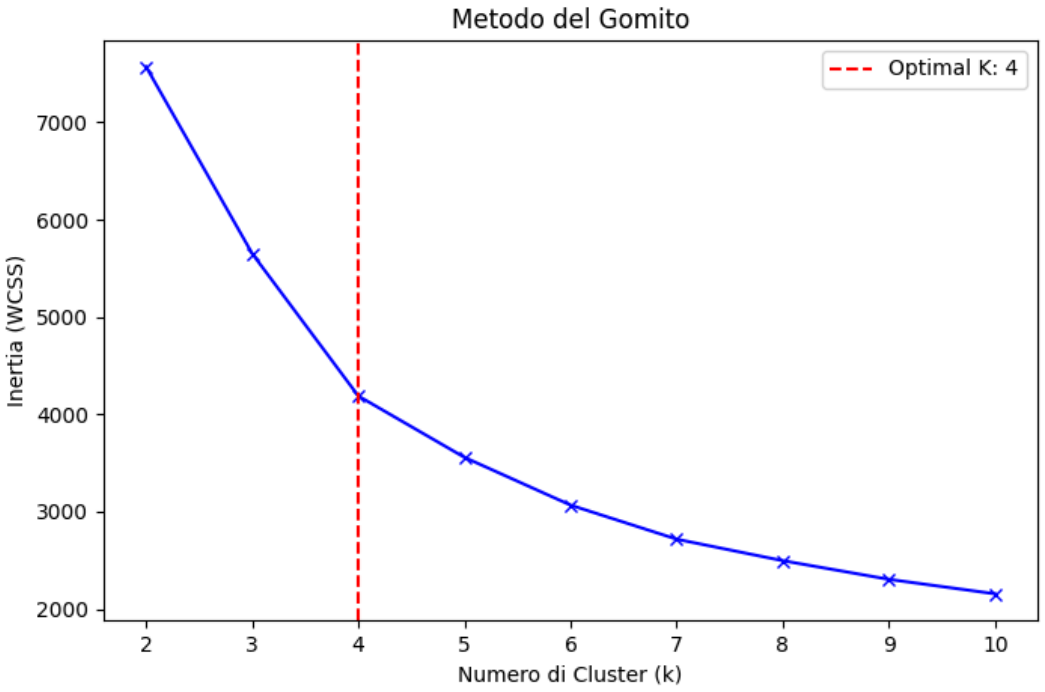
Clustering K-Means

È stato scelto l'algoritmo **K-Means** per la sua efficienza e interpretabilità. Il clustering è stato eseguito sulle feature strutturali: `bpm` (velocità), `duration` (resistenza) e `notes` (densità). Per determinare il numero ottimale di cluster (k)



Decisioni

È stato utilizzato il **Metodo del Gomito (Elbow Method)**. L'algoritmo è stato eseguito per valori di k da 2 a 10, calcolando per ogni iterazione la *Within-Cluster Sum of Squares* (WCSS/Inerzia). L'analisi del grafico risultante, supportata dalla libreria `kneed`, ha identificato in modo chiaro k=4 come il punto di gomito ottimale, dove il guadagno marginale in termini di riduzione della varianza inizia a decrescere significativamente.



Valutazione: Analisi semantica dei cluster

Una volta ottenuti i 4 cluster, sono stati analizzati i valori dei loro centroidi per assegnare un'etichetta semantica a ciascuno, rendendo i risultati interpretabili per il sistema decisionale:

1. **Cluster 0 - HighSpeed:** Caratterizzato da BPM molto elevato ma durata contenuta. Queste mappe richiedono riflessi estremamente rapidi e capacità di lettura veloce, ma meno resistenza a lungo termine.
2. **Cluster 1 - Warm-up:** Mappe con valori bassi sia di densità di note che di durata. Sono ideali per il riscaldamento o per sessioni defaticanti.
3. **Cluster 2 - Balanced:** Mappe con valori medi ed equilibrati su tutte le feature. Rappresentano lo standard dell'esperienza di gioco a livello "Expert".
4. **Cluster 3 - Extreme:** Mappe con densità di note e durata molto elevate. Rappresentano la sfida fisica maggiore, richiedendo un'alta resistenza cardiovascolare e muscolare (Stamina).

Questa categorizzazione ha generato una nuova feature categorica, `map_cluster`, che è stata aggiunta al dataset per arricchire le fasi successive.

Sezione 3) Rappresentazione della Conoscenza e Ragionamento

Sommario

Per potenziare le capacità predittive del sistema, è stato integrato un modulo di ragionamento logico basato su **Prolog**. L'intuizione alla base è che il solo "Rank" di un giocatore è un indicatore troppo aggregato: due giocatori con lo stesso Rank potrebbero avere profili di abilità molto diversi (es. uno specialista in velocità vs uno specialista in resistenza).

Strumenti:

Librerie Utilizzate**:** `pyswip`: permette di utilizzare prolog in python con un bridge, `pandas`.

Knowledge Base e Regole

È stata costruita una Knowledge Base traducendo i dati del dataset in fatti Prolog ad esempio:

```
played(PlayerID, ClusterName, Score, Accuracy).
```

Successivamente, è stato definito un set di regole logiche per inferire conoscenza non esplicitamente presente nei dati grezzi. In particolare, sono state create regole per calcolare la performance media storica (sia in termini di punteggio assoluto che di precisione) di un giocatore specifica per ciascun cluster di mappe.

Ecco le regole implementate per l'inferenza delle medie:

Per calcolare lo score medio di un giocatore su un cluster specifico:

```
average_score_cluster(Player, Cluster, AvgScore) :-
    findall(Score, played(Player, Cluster, Score, _), ScoreList),
    length(ScoreList, Count),
    Count > 0,
    sum_list(ScoreList, Sum),
    AvgScore is Sum / Count.
```

Per calcolare l'accuracy media di un giocatore su un cluster specifico:

average_accuracy_cluster(Player, Cluster, AvgAcc) :-

findall(Acc, played(Player, Cluster, _, Acc), AccList),

length(AccList, Count),

Count > 0,

sum_list(AccList, Sum),

AvgAcc is Sum / Count.

`findall`*(Acc, played(Player, Cluster, _, Acc), AccList)* trova ogni mappa in un determinato cluster giocata da un giocatore e aggiunge la accuracy *Acc* corrispondente ad una lista *AccList*.

length(AccList, Count) conta il numero di elementi della lista

Count>0 è un check per controllare che la lista non sia vuota

sum_list(AccList, Sum) somma tutti i valori nella lista

AvgAcc is Sum / Count divide Sum per il numero degli elementi della lista quindi è di fatto la media delle accuracy per quel cluster di mappe giocata da quel giocatore.

Interrogazione della Kb

Utilizzando la libreria **pyswip**, è stato interrogato il motore Prolog direttamente dallo script Python. Per ogni coppia (Giocatore, Cluster) presente nel dataset, sono state lanciate le query `average_score_cluster` e `average_accuracy_cluster`. I risultati di queste inferenze sono stati utilizzati per arricchire il dataset con due nuove feature:

- `avg_score_in_cluster`: Il punteggio medio storico del giocatore in quella specifica tipologia di mappa.
- `avg_acc_in_cluster`: La precisione media storica del giocatore in quella specifica tipologia di mappa.

Sezione 4) Apprendimento Supervisionato

Sommario

L'obiettivo di questa fase è stato l'addestramento di un modello di regressione in grado di predire con precisione i **PP** (Performance Points) che un giocatore otterrà.

Iterazione 1: Dati Grezzi

In una prima fase sperimentale, è stato tentato l'addestramento dei modelli utilizzando esclusivamente le feature grezze della mappa (`bpm`, `notes`, `duration`) e il `player_rank`. Le

metriche di valutazione ottenute non sono state soddisfacenti, indicando che queste sole variabili non erano sufficienti a catturare la complessa relazione tra l'abilità del giocatore e la difficoltà della mappa. Questo ha confermato la necessità di un approccio più sofisticato basato sul ragionamento logico.

Iterazione 2: Dati Arricchiti

Si è quindi proceduto utilizzando il dataset arricchito dalle feature logiche inferite nel capitolo precedente (`avg_score_in_cluster`, `avg_acc_in_cluster`). L'inclusione di queste informazioni contestuali ha permesso al modello di migliorare la capacità di generalizzazione.

Strumenti utilizzati

Librerie Utilizzate**:** `scikit-learn`, `matplotlib`, `pandas`, `numpy`, `joblib` (`serializzazione e salvataggio del modello`).

Modelli

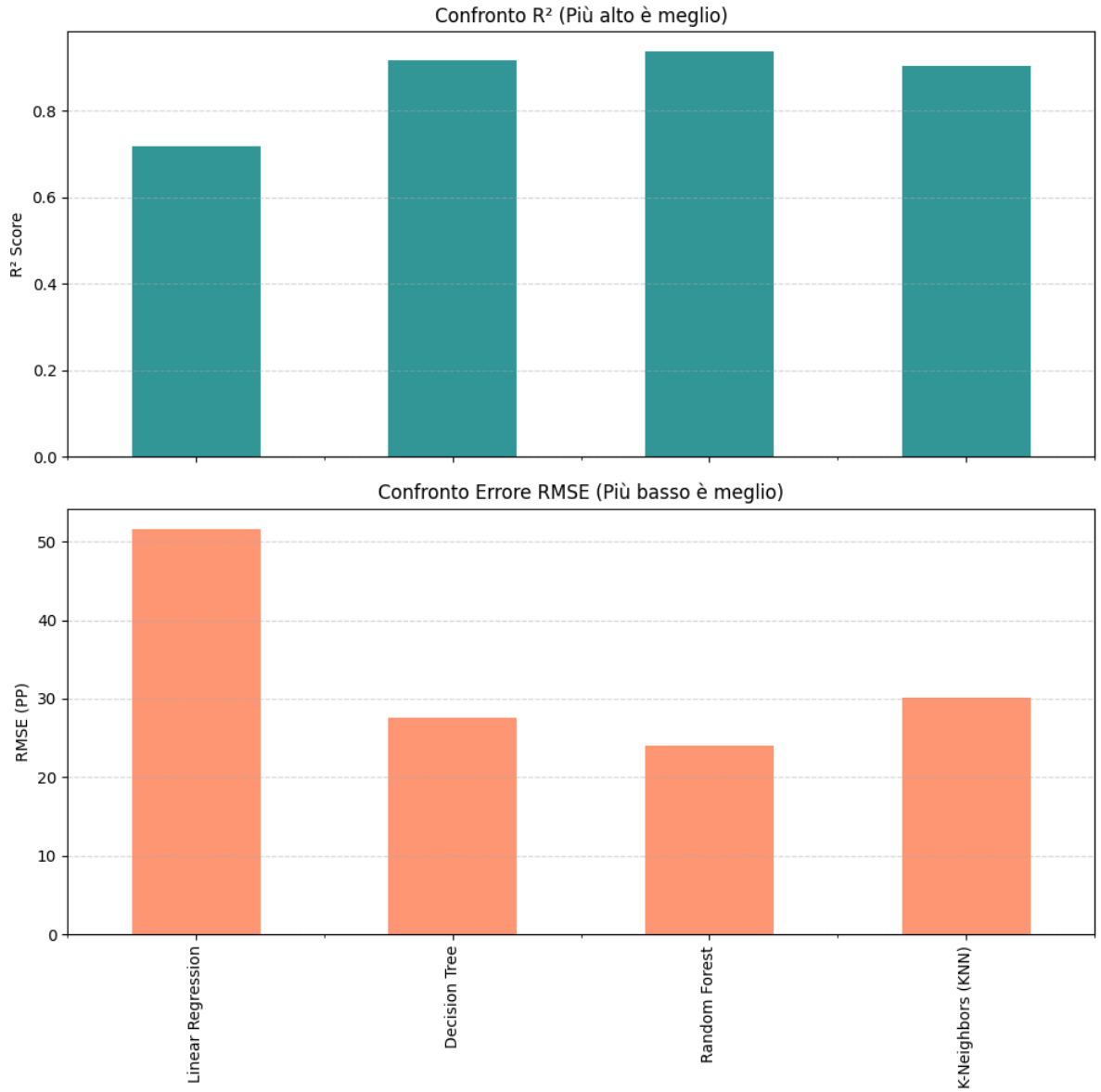
I modelli confrontate per ottenere il migliore regressore sono stati:

- Linear Regression
- Decision Tree Regressor
- Random Forest Regressor
- K-Nearest Neighbors (KNN)

Decisioni di Progetto

È stato condotto un confronto sistematico tra diversi algoritmi di regressione, utilizzando una **10-Fold Cross-Validation** per garantire la robustezza statistica dei risultati ed evitare l'overfitting.

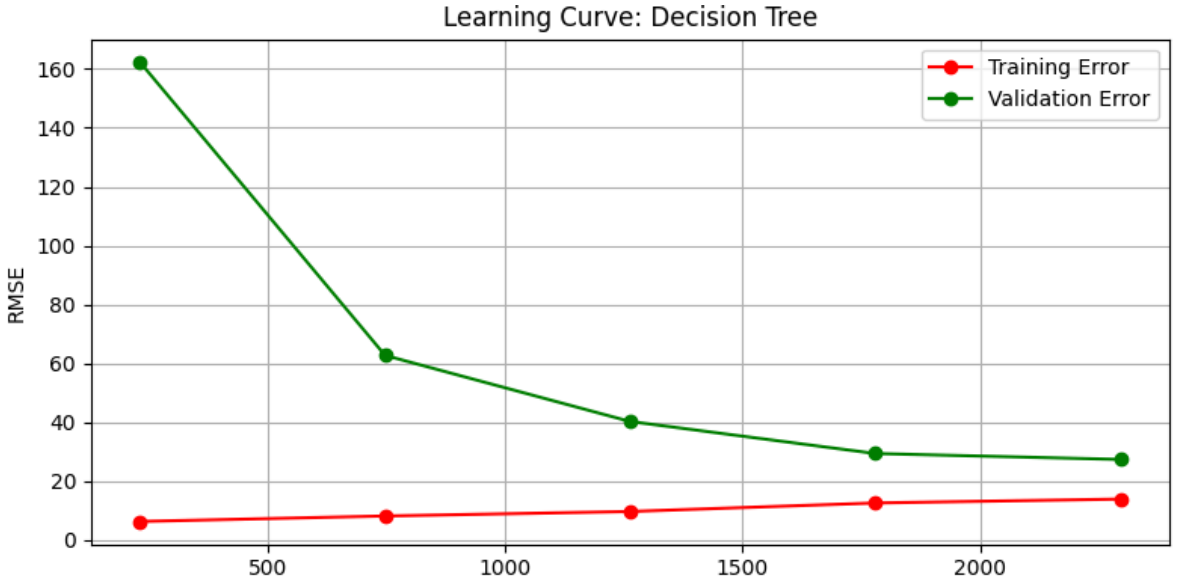
Le metriche utilizzate per il confronto sono state **R^2 (Coefficiente di determinazione)**, per valutare la bontà dell'adattamento, e **RMSE (Root Mean Squared Error)**, per quantificare l'errore medio di predizione.

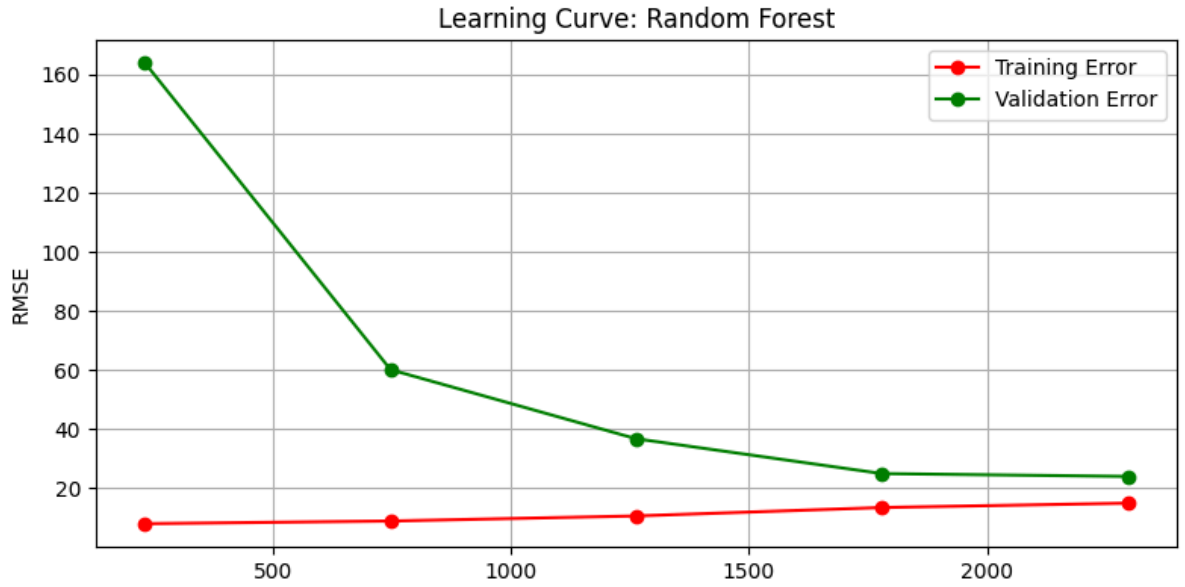


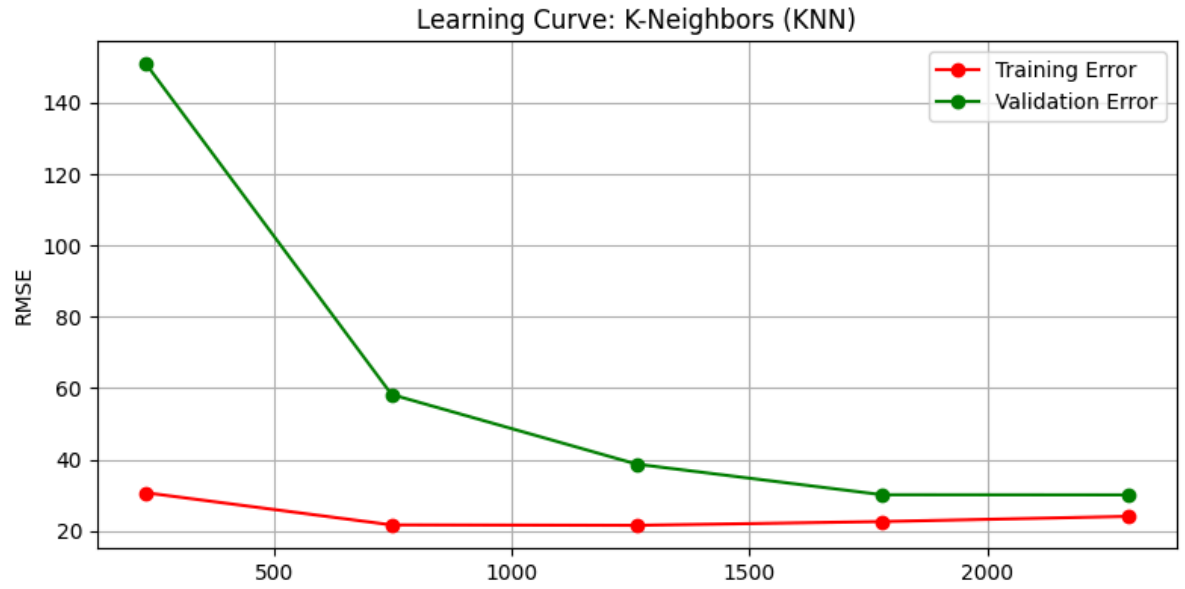
Valutazione

Il modello **Random Forest Regressor** si è rivelato il più performante. Il modello ottimizzato è stato salvato per essere utilizzato come predittore nella fase successiva.


Learning Curve: Linear Regression
Training Error
Validation Error
RMSE
800
700
600
500
400
300
200
100
500
1000
1500
2000


Learning Curve: Decision Tree
Training Error
Validation Error
RMSE
160
140
120
100
80
60
40
20
0
500
1000
1500
2000





Sezione 5) Constraint Satisfaction Problem (CSP)

Sommario

L'ultimo modulo del KBS è il generatore del **Piano di Allenamento** per il giocatore. Questo compito è stato modellato come un Problema di Soddisfacimento dei Vincoli (CSP), dove l'obiettivo è selezionare una sequenza ottimale di 4 mappe che sufficientemente difficili per il giocatore, ma non troppo, per massimizzare i PP che potrebbe ottenere, e rispettando al contempo dei criteri per migliorare il ritmo di allenamento selezionando inizialmente una mappa “di riscaldamento”, ovvero appartenente al cluster *warm-up*, caratterizzate da tracce meno impegnative relativamente

all’intensità, e finendo sempre con una mappa del cluster *Extreme*, ad altissima intensità, per concludere l’allenamento.

Strumenti utilizzati

Librerie Utilizzate: `python-constraint` (Risolutore CSP), `pandas`, `numpy`, `joblib` (per caricare il modello ML).

Backtracking Search: Per risolvere il Constraint Satisfaction Problem l'algoritmo esplora lo spazio delle possibili combinazioni di mappe, verificando passo dopo passo il soddisfacimento dei vincoli. Appena trova un'assegnazione parziale che viola un vincolo (es. due mappe uguali), effettua il backtracking, abbandonando quel ramo di ricerca.

Decisioni di Progetto

Definizione del Problema

- **Variabili:** 4 slot temporali sequenziali (M1, M2, M3, M4) che compongono la sessione di allenamento.dizione del modello ML (le "Top-K" mappe più promettenti per quel giocatore).
- **Vincoli:** È stato imposto un set di vincoli per garantire che il piano generato fosse vario, stimolante e strutturato logicamente:
    1. **Vincolo Unario (Hard):** La prima mappa (M1) deve appartenere obbligatoriamente al cluster "Warm-up", per garantire un riscaldamento adeguato.
    2. **Vincolo Unario (Hard):** L'ultima mappa (M4) deve appartenere al cluster "Extreme", per fornire una sfida finale impegnativa.
    3. **Vincolo Globale:** `AllDifferent`. Tutte le mappe nel piano devono essere distinte tra loro per evitare ripetizioni monotone.

Previsione e Risoluzione del CSP

Il processo di risoluzione avviene secondo un flusso strutturato che trasforma l'input utente (il Rank) in un piano di allenamento concreto.

1. **Nearest Neighbors:** Poiché il modello ML necessita non solo del Rank, ma anche delle performance medie storiche (`avg_score`, `avg_acc`). Nel caso in cui si inserisca il rank di un utente di cui non si hanno informazioni sulle performance nel sistema, ho implementato una tecnica di imputazione basata sui vicini. Il sistema scansiona il dataset alla ricerca di giocatori con un Rank simile a quello inserito (range ±10%). Calcola quindi la media delle loro statistiche (`avg_score`, `avg_acc`) e le attribuisce all'utente corrente. Questo permette di stimare con buona approssimazione il profilo di abilità atteso per quel livello di classifica.
2. **Predizione:** Una volta costruito il profilo dell'utente, il sistema utilizza il modello **Random Forest** per effettuare una predizione su tutte le mappe disponibili nel dataset. Per ogni mappa candidata, il modello stima i **PP** che l'utente otterrebbe giocandola.
3. **Ottimizzazione dei Domini:** Invece di fornire al risolutore CSP tutte le migliaia di mappe disponibili (il che renderebbe la ricerca esponenzialmente lenta), i domini delle variabili vengono potati.

- Per la variabile M1 (Warm-up), vengono selezionate solo le migliori 10 mappe del cluster "Warm-up" (quelle con i PP predetti più alti).
- Per la variabile M4 (Extreme), vengono selezionate le migliori 10 mappe del cluster "Extreme".
- Per le variabili centrali (M2, M3), vengono selezionate le migliori 30 mappe in generale.

4. **Risoluzione (Backtracking Search):** Il problema viene infine passato al risolutore `python-constraint`, che utilizza un algoritmo di backtracking Search per risolvere il CSP. Tra tutte le soluzioni valide trovate, il sistema seleziona quella che massimizza la somma totale dei PP previsti, restituendo all'utente il piano di allenamento ottimale.

```
Inserisci il tuo Rank attuale (es. 1500): 2500

Profilo Stimato per Rank 2500:
   Avg Score atteso: 1487629
   Avg Acc attesa:   0.91
   Calcolo predizioni ML per tutte le mappe candidate...

Avvio Risolutore CSP (Backtracking)...
   Ricerca soluzioni...
   Trovati 81200 piani validi. Ottimizzazione (Max PP)...

==================================================
   PIANO DI ALLENAMENTO PERSONALIZZATO (Rank 2500)
==================================================
[Mappa_1_Start]
   Mappa:   A1315B48BEFE9C6... (Hash)
   Tipo:    Warm-up (Cluster 1)
   BPM:     170 | Note: 849
   Est. PP: 345.54
------------------------------
[Mappa_2]
   Mappa:   3C37D1264FC2754... (Hash)
   Tipo:    Warm-up (Cluster 1)
   BPM:     165 | Note: 776
   Est. PP: 345.54
------------------------------
[Mappa_3]
   Mappa:   5C6D84C52B7B1B3... (Hash)
   Tipo:    Warm-up (Cluster 1)
   BPM:     200 | Note: 295
   Est. PP: 345.54
------------------------------
[Mappa_4_End]
   Mappa:   9A972C45D803012... (Hash)
   Tipo:    Extreme (Cluster 3)
   BPM:     175 | Note: 2339
   Est. PP: 344.26
------------------------------
TOTALE PP PREVISTI: 1380.89
==================================================
```

Conclusioni

Il progetto ha dimostrato come l'integrazione di tecniche diverse di Intelligenza Artificiale possano risolvere problemi complessi in domini reali, anche in campo videoludico.

Sviluppi Futuri

Un possibile sviluppo futuro del sistema consiste nell'implementazione di una funzionalità di predizione puntuale on-demand. Questa funzione permetterebbe all'utente di inserire specificamente l'**ID univoco di una mappa** (songHash) e l'**ID di un giocatore** (o il suo Rank). Il sistema, interrogando il modulo di clustering per identificare la tipologia della mappa inserita e recuperando il profilo storico del giocatore, restituirebbe una stima precisa della performance attesa (PP) per quella specifica combinazione, offrendo uno strumento utile per valutare la fattibilità e il potenziale di guadagno di una mappa prima ancora di giocarla.

Riferimenti Bibliografici

[1] D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. 3e, Cambridge University Press

[2] https://scikit-learn.org/stable/getting_started.html