AIMm: Artificial Intelligence for Matchmaking

Giuseppe Sabetta
Saverio D'Avanzo

13 Febbraio 2025

1 Introduzione

Lo Skill-Based-Matchmaking, maggiormente conosciuto come SBMM, è un paradigma molto presente nella maggior parte dei videogiochi online.
Esso permette di poter giocare con persone al proprio livello, senza alcun tipo di problema di sbilanciamento delle abilità (skill), migliorando la **giocabilità** e il **divertimento** dei giocatori.
Purtroppo non sempre gli algoritmi usati per l'SBMM riescono a dare una buona giocabilità e prevedono uno scarso bilanciamento o un bilanciamento dei giocatori in una lobby così elevato da non avere la possibilità di migliorarsi, avendo tutti i player la stessa strategia, le stesse meccaniche e statistiche troppo vicine.
Quando il **bilanciamento** è troppo alto o troppo basso, la giocabilità si **riduce di colpo**. Per giocabilità si intende la dinamicità classica dei videogiochi che, fino a diversi anni fa, contenevano giocatori di qualsiasi tipo, potendo giocare e divertirsi, indipendentemente dalle proprie abilità. Attualmente, la distribuzione dei player in base alla loro competitività nelle statistiche presenta troppi picchi su principianti e su esperti.
La **giocabilità** è la possibilità di avere un bilancio ma non eccessivo al punto di avere giocatori quasi "identici", garantendo lobby variegate ma con abilità che si rispecchiano abbastanza, rendendo più dinamica e simpatica l'esperienza.
Il compito di **AIMm** è quello di garantire e spostare i vari giocatori in delle lobby variegate ma bilanciate al punto giusto, riuscendo a non avere esperienze negative per i principianti (che devono contrastare le abilità e strategie degli esperti), ma neanche dei più esperti (che devono contrastare persone con quasi le stesse abilità, non consentendo **dinamicità**).

2 Business Understanding

Il matchmaking tradizionale spesso porta a esperienze **frustranti** a causa di squilibri nelle abilità dei giocatori. AIMM mira a migliorare questo aspetto.
Il lavoro di AIMm è molto semplice: data una quantità indeterminata di giocatori in coda, in attesa di entrare in una lobby, l'algoritmo deve poter selezionare diversi giocatori da inserire in diverse lobby e quest'ultime devono avere le combinazioni di giocatori migliori al fine di avere una giocabilità elevata ed un bilanciamento non troppo elevato (si esclude il bilanciamento basso poichè irrosorio per l'esperienza del principiante).
Ovviamente, il bilanciamento non ha solo la variabile dell'ottimalità dell'algoritmo, bensì anche del quantitativo di giocatori in coda e la qualità dei giocatori in coda: in caso di poche persone, le lobby potrebbero essere gestite semplicemente al fine di riempire il vuoto, risultando negativo per i principianti e troppo positivo per gli esperti.
Il rischio di quest'algoritmo, nel caso peggiore, è di risultare in un bilanciamento eccessivamente basso o alto, uniformandosi agli algoritmi troppo statici e "spenti" attuali. **Un altro rischio** possibile è che, se il bilanciamento segue la visione avuta nelle fasi di design, ci possono essere dei ritardi nel ricercare una partita e ciò si alimenterà ancora di più in mancanza di giocatori (o mancanza di giocatori in grado di bilanciare possibili lobby). Non ci sarà il rischio di non trovare mai una partita poichè l'algoritmo si basa sul garantire il **bilanciamento dove possibile**, quindi dipendendo dalle variabili citate precedenemente.
Analizziamo i **business criteria** in soldoni:

- Non basare la selezione del giocatore in una lobby in base a semplici attributi, ma studiando, attraverso formule matematiche date da varie **euristiche** con piena conoscenza del dominio ap-

plicativo, tutti gli attributi che creeranno la vera e propria distanza di statistica. Queste euristiche permetteranno di valutare **ogni statistica** nel suo insieme, tenendo in considerazione che una partita non deve fare la differenza al livello di selezione del giocatore nella lobby.

- Maggiore **soddisfazione** nell'esperienza dell'utente, indipendentemente dalle sue skills.

Dati i business criteria, parliamo ora degli **obiettivi tecnici**, sfruttando l'occasione per spiegare meglio le parole chiave del progetto:

- L'algoritmo dovrà calcolare la **distanza di statistica**, determinante per capire le differenze tra un giocatore e l'altro
- L'algoritmo presenterà una funzione per calcolare la **giocabilità** della lobby, ovvero la **soddisfacibilità** dei giocatori all'interno di una determinata lobby, usando formule e calcoli dati da euristica e conoscenze del dominio applicativo. Per soddisfacibilità si intende una **previsione dell'esperienza** di ogni utente all'interno di una lobby in base alla distanza di statistica.
- L'algoritmo dovrà presentare un giusto **bilanciamento**, non eccessivamente alto, nè eccessivamente basso. Il bilanciamento, in teoria, è una caratteristica dell'algoritmo che dipende dal numero di persone, dalla loro qualità e la qualità dell'algoritmo usato per selezionarli e inserirli in determinate lobby. In pratica, **è la media delle giocabilità** delle varie lobby prese in considerazione.
- La giocabilità e il bilanciamento sono valori in percentuale. Il risultato finale che vorremo è un **65% di bilanciamento**, il giusto per definire la giocabilità media delle lobby prese in considerazione.

Costi-Benefici:

- **Maggiori dati ed esperienze** garantiranno bilanciamento e giocabilità migliorati.

3 Data Understanding

Per costruire il sistema, abbiamo utilizzato un dataset esistente in rete con esperienze reali sul videogioco sparatutto FPS *Call of Duty: Modern Warfare (2019)*. Purtroppo, il dataset utilizzato era l'unico accessibile per le statistiche che l'algoritmo deve gestire per operazioni di selezione e inserimento dei giocatori all'interno delle lobby. Proprio per questo, abbiamo utilizzato il dataset riguardante questo videogioco, nonostante alcune **problematiche** come esperienze surreali per correlazione tra le varie informazioni nelle statistiche o inconsistenza dei dati, o addirittura problemi nelle variabili dipendenti da altre variabili (errori nei calcoli).
Le colonne di questo dataset sono le seguenti:

- *index*: indice dell'esperienza / giocatore con le sue statistiche, partendo da 1.
- *name*: nome del giocatore
- *wins*: vittorie di un determinato giocatore. Ci aspettiamo qualsiasi valore intero non negativo. In generale nessun valore potrebbe risultare irrisorio. Questa informazione sarà anormale solo se inconsistente rispetto a tutte le altre informazioni di statistica di un determinato giocatore.
- *losses*: sconfitte di un determinato giocatore. Ci aspettiamo qualsiasi valore intero non negativo. Sull'anormalità dell'informazione presa in considerazione, si effettua lo stesso ragionamento delle vittorie.
- *kills*: uccisioni in totale effettuate da un certo giocatore. Ci aspettiamo qualsiasi valore intero non negativo.
- *deaths*: morti in totale avute da un certo giocatore. Ci aspettiamo qualsiasi valore intero non negativo.
- *kdRatio*: rapporto delle uccisioni e le morti
- *killstreak*: massimo di uccisioni effettuate in una partita da un certo giocatore. Ci aspettiamo un valore intero non negativo non eccessivamente alto, con un massimo di 100.

- *level*: livello del giocatore, intero non negativo non troppo irrisorio. Avrà un range [1,55] se "Prestige" è maggiore di 10, altrimenti fino a 8000.
- *prestige*: il numero di "prestigio" di livello di un giocatore. Ogni 55 livelli, il giocatore può aumentare di prestigio e resettare il suo livello, per puro intrattenimento (e ricompense). Ci aspettiamo un valore intero non negativo che parte da 0 fino a 11.
- *assists*: numero di assist (assistenza alle uccisioni) effettuati da un determinato giocatore. Ci aspettiamo un numero intero non negativo.
- *shots*: numero di colpi sparati in totale da un giocatore. Ci aspettiamo un valore non negativo, anche irrisorio.
- *hits*: numero di colpi sparati sul corpo da un giocatore. Ci aspettiamo un valore non negativo, anche irrisorio.
- *headshots*: numero di colpi alla testa sparati da un giocatore. Ci aspettiamo un valore non negativo, anche irrisorio.
- *misses*: numero di colpi sparati a vuoto da un giocatore. Ci aspettiamo un valore non negativo, anche irrisorio.
- *xp*: numero di punti esperienza totali del giocatore. Ci aspettiamo qualsiasi valore non negativo.
- *scorePerMinute*: numero di punteggio che riesce a guadagnare il giocatore in un minuto. Ci aspettiamo qualsiasi valore non negativo.
- *averageTime*: tempo medio di gioco in minuti al giorno del giocatore. Ci aspettiamo un valore inferiore a 1440.
- *timePlayed*: tempo totale di gioco in ore del giocatore. Ci aspettiamo qualsiasi valore non negativo.
- *gamesPlayed*: partite giocate da un determinato giocatore. Ci aspettiamo qualsiasi valore non negativo

In questo caso possiamo avere diverse **anomalie**, soprattutto a causa del dataset utilizzato. Le anomalie sono principalmente sui **calcoli** (dove alcuni valori che dipendono da somme tra altri valori di statistica sono sbagliati), sull'**inesistenza di esperienze simili** a player che hanno un livello alto e basso tempo giocato e viceversa, sulla presenza di un **numero eccessivo** di wins rispetto alle losses (o kills, assists e deaths).

4 Data Preparation

Come è stato già accennato, la maggior parte delle problematiche sono sempre dovute ai dati che si hanno a disposizione. Nel nostro caso, conoscendo il dominio applicativo, abbiamo effettuato varie tecniche per eliminare delle esperienze surreali o impossibili, tecniche di data imputation per mitigare problemi su valori mancanti, feature scaling per normalizzare valori con range troppo diversi rispetto agli altri, feature selection per garantire l'accesso alle varie funzioni di selezione dei giocatori all'interno della lobby calcolando solo un certo insieme di statistiche ed informazioni importanti. Vediamo passo dopo passo quali sono state le tecniche utilizzate per migliorare ed analizzare il dataset:

1. Si è iniziata la fase di Data Preparation con una **Feature Selection** di primo acchitto, rimuovendo delle feature che non serviranno per analizzare e modellare sul nostro dataset. Le feature meno importanti sono l'*averageTime*, poichè è già abbastanza sapere il tempo giocato totale e il livello, e il *name*, dato che il gamertag con il tag giocatore definiscono la chiave identificativa di ogni individuo.
2. Solitamente, la fase di feature selection viene seguita per ultima nella sezione di Data Preparation, ma noi abbiamo deciso di effettuarla per prima per rendere più compatto il dataset e restringere il dominio delle feature, dato che alcune di esse erano appunto inutili per la nostra analisi. Subito dopo la feature selection di primo acchitto, abbiamo effettuato una **Pulizia dei Dati** non indifferente, a causa di dati troppo rumorosi, mancanti o surreali.

Conoscendo il dominio applicativo e le possibili esperienze (con i vari range per ogni feature), abbiamo effettuato le seguenti operazioni:
- **Nan/Infinito**: Grazie alle tecniche del Data Imputation e conoscendo il dominio applicativo e come sono state fornite le esperienze del dataset, si è deciso di rendere ogni dato mancante o infinito semplicemente 0. In certi casi, con feature dipendenti da altre feature (ovvero risultati di calcoli tra valori di più feature), si è deciso di rieffettuare il calcolo. In certi casi si è preferito eliminare direttamente esperienze contenenti valori e statistiche effettivamente improbabili o non controllate.
- **Feature Interaction**: certe feature sono eccessivamente accoppiate e la loro separazione non rende l'idea del progetto. Infatti "level" e "prestige" trattano lo stesso argomento, ovvero l'esperienza accumulata. Per eliminare la possibilità di avere troppe feature che interagiscono tra loro, si usa la tecnica di **Feature Interaction**, usando la formula con euristica basata su conoscenza del dominio applicativo, ovvero $\text{level}_a nd_p restige == level + prestige * 55$.
- **Data Balancing**: Si è infine utilizzata una tecnica di data balancing per rimuovere tutte le esperienze eccessivamente sbilanciate, come quelle che contengono un rapporto vittorie/sconfitte maggiori del 100.

5 Modeling

Per l'iniziale concezione del matchmaking che avevamo, vedevamo il problema come una semplice **classificazione** (supervisionato), in cui ogni lobby avrebbe rappresentato una classe distinta e i giocatori sarebbero stati assegnati di conseguenza. Tuttavia, questa impostazione si è rivelata poco scalabile e **inadatta al contesto** poichè non è nota nè la quantità di giocatori, nè la qualità di essi e nemmeno in quale esatto momento, essendo il matchmaking basato sulla selezione real-time da una coda. Il numero di lobby, infatti, non è un valore fisso e prestabilito, ma varia dinamicamente in base al numero di giocatori disponibili e alla loro distribuzione di abilità. Un modello di classificazione tradizionale avrebbe richiesto una struttura rigida con classi predefinite, risultando inefficace per gestire una variabilità così elevata.
Per affrontare questa sfida, è stato adottato un approccio basato sul **clustering**, con l'obiettivo di raggruppare i giocatori in base a caratteristiche simili, creando così lobby abbastanza bilanciate. La tecnica scelta è stata K-Means, un algoritmo di clustering che suddivide i dati in gruppi omogenei ottimizzando la varianza interna ai cluster. Per determinare il numero ottimale di cluster, è stato utilizzato il metodo del gomito (Elbow Method), che analizza l'andamento della somma degli errori quadratici (SSE) rispetto al numero di cluster, individuando il punto oltre il quale aggiungere ulteriori cluster non comporterebbe un miglioramento significativo.
Prima di applicare il clustering, è stato necessario effettuare un pre-processing delle caratteristiche dei giocatori. I dati sono stati normalizzati mediante *StandardScaler*, in modo da garantire che tutte le feature avessero la stessa scala e che il clustering non fosse influenzato da grandezze numericamente dominanti. Inoltre, sono stati introdotti dei **pesi differenziati** sulle feature, con una divisione ponderata per enfatizzare maggiormente alcuni aspetti ritenuti più significativi nel matchmaking, principalmente quelli rilevanti l'abilità del giocatore, riducendo il peso, di conseguenza, delle feature riguardanti il tempo di gioco. Avremo così un bilancio di 70% feature riguardanti le skill e 30% feature riguardanti il tempo di gioco
Una volta individuati i cluster, il passaggio successivo è stato quello di **assegnare nuovi giocatori** a lobby appropriate. Per fare ciò, è stato adottato un metodo basato su **K-Nearest Neighbors** (KNN), che assegna ogni nuovo giocatore al cluster più affine in base alla somiglianza con i giocatori già presenti. L'algoritmo utilizza la metrica della distanza per confrontare le caratteristiche del nuovo giocatore con quelle dei membri dei cluster preesistenti, garantendo un'assegnazione coerente con la distribuzione dei gruppi già formati. Grazie a questa tecnica, il matchmaking si adatta **dinamicamente** a nuovi ingressi, permettendo una gestione flessibile e bilanciata delle lobby.
Per facilitare l'interpretazione dei cluster, è stata inoltre applicata **l'Analisi delle Componenti Principali** (PCA), una tecnica di riduzione della **dimensionalità** che permette di visualizzare i dati in uno spazio bidimensionale. Questo passaggio ha consentito di rappresentare in modo intuitivo la distribuzione dei giocatori, confermando la coerenza della segmentazione effettuata dal modello di clustering.
Infine, una volta formate le lobby, è stata implementata una suddivisione dei giocatori in **squadre**, con l'obiettivo di garantire partite **equilibrate**. I giocatori all'interno di ogni lobby sono stati suddivisi in due squadre attraverso un'assegnazione basata su criteri di bilanciamento e casualità, con l'obiettivo di mantenere la competitività delle partite senza creare dislivelli eccessivi.
L'approccio adottato, combinando K-Means per il clustering, KNN per l'assegnazione dei nuovi giocatori

e PCA per la visualizzazione, ha permesso di sviluppare un sistema di matchmaking **dinamico**, **scalabile** e **adattabile** alle variazioni del pool di giocatori, offrendo un'esperienza bilanciata e coinvolgente.
Dato l'algoritmo **non supervisionato** del clustering e non essendo un semplice problema di classificazione, non possiamo addestrare il nostro modello portando delle esperienze in un cluster o meno: queste decisioni, che effettuerà il nostro model in tempo reale, sono dipendenti dal numero e dalla qualità delle statistiche dei giocatori che arrivano alla coda del matchmaking. Pertanto, si utilizzerà (e si analizzerà nella sezione successiva) tramite formule di studio della giocabilità.

6 Evaluation

La fase di evaluation rappresenta un momento fondamentale per verificare la validità e la coerenza del sistema di matchmaking sviluppato, in linea con gli obiettivi tecnici e di business definiti nelle fasi precedenti. In particolare, l'analisi si è concentrata su due aspetti principali: la coesione interna dei cluster formati e l'equità delle partite generate, elementi essenziali per garantire una buona giocabilità e una distribuzione equilibrata delle abilità dei giocatori.
Per quanto riguarda la **Coesione** dei cluster, è stata adottata una metrica di **giocabilità** basata sulla varianza delle feature chiave. In sostanza, per ciascun cluster è stata calcolata la deviazione standard media delle suddette feature, in modo da quantificare la dispersione interna. Tale valore, una volta normalizzato rispetto alla deviazione standard media calcolata sull'intero dataset, è stato tradotto in una percentuale inversa, secondo la formula:

$$Giocabilità(\%) = 100 - \left(\frac{std_{cluster}}{std_{global}} \times 100 \right)$$

Questo procedimento permette di attribuire una percentuale di giocabilità ad ogni cluster, dove un valore elevato indica una bassa varianza e, di conseguenza, una maggiore coesione interna. I risultati sono stati mostrati sia tramite output **testuale** che mediante un grafico **a barre**, nel quale è stata tracciata una **linea di soglia** (al 40%) per evidenziare eventuali cluster potenzialmente sbilanciati. Tale analisi ha fornito una conferma quantitativa della capacità del modello di clustering di raggruppare in modo omogeneo i giocatori, garantendo lobby con distribuzioni di skill equilibrate.

Parallelamente, è stata condotta una valutazione della **Fairness** delle partite, finalizzata a verificare l'equilibrio tra le squadre all'interno di ciascuna lobby. L'approccio utilizzato si basa sul calcolo di un fairness score, ottenuto misurando il rapporto tra la media delle skill dei due team formati. In dettaglio, per ogni lobby i giocatori sono stati mescolati casualmente e suddivisi in due squadre; successivamente, per ciascun team è stata calcolata la media dei valori relativi alle feature di skill. Il fairness score, definito come il rapporto tra la media del team con il punteggio maggiore e quella del team con il punteggio minore, consente di evidenziare il livello di equilibrio: un valore prossimo a 1.4 (o inferiore) è considerato **auspicabile**, mentre valori superiori indicano una possibile asimmetria che **potrebbe compromettere** la competitività della partita. Ovviamente certi risultati, in base al dataset utilizzato e gli obiettivi del progetto, anche se superano di poco la soglia, sono ancora più accettati perchè rispecchiano quel minimo di dinamicità maggiore rispetto al paradigma solitamente utilizzato in rete. I risultati, stampati a video e rappresentati graficamente mediante un diagramma a barre in cui le barre sono colorate in verde (se il punteggio è accettabile) o in rosso (se supera la soglia critica), hanno permesso di identificare con chiarezza le partite che potrebbero risultare sbilanciate, suggerendo eventuali interventi correttivi. In effetti, **grazie a questa fase**, abbiamo modificato la proporzione nel modeling da una suddivisione per peso **60-40** ad una suddivisione dei pesi **70-30**, aumentando ancor di più il peso delle skill, poichè parte importante del bilancio delle statistiche.
In sintesi, l'analisi effettuata dimostra che il sistema di matchmaking sviluppato riesce a formare cluster di giocatori omogenei, garantendo lobby bilanciate dal punto di vista delle statistiche di skill, e che la successiva suddivisione in squadre permette di mantenere un livello di fairness adeguato. Questi risultati, ottenuti grazie a un'attenta valutazione sia quantitativa che visiva, confermano l'efficacia dell'approccio adottato nel garantire una buona giocabilità e un'esperienza di gioco soddisfacente per la maggior parte degli utenti, in piena coerenza con gli obiettivi di AIMm.
NOTA BENE: Esistono le seguenti implicazioni che ci fanno capire meglio la relazione tra giocabilità e le lobby, per studiare di più quest'ultime:

- Giocabilità alta e Lobby Fair: Se la lobby che fa parte di un certo cluster giocabile è altrettanto fair, si ha un perfetto bilanciamento. Attenzione ai valori perchè se sono troppo precisi entrambi (verso il

100 giocabilità e verso l'1 la fairness), si può incappare in uno dei problemi definiti nell'introduzione

- Giocabilità bassa e Lobby Fair: nonostante la bassa giocabilità all'interno del cluster, si riesce comunque a trovare fairness nella lobby grazie agli algoritmi utilizzati. La bassa giocabilità del cluster, dipendendo sempre da quale cluster e quali esperienze contiene, è giustificata.
- Giocabilità alta e Lobby Unfair: Dato che la lobby fa parte del cluster giocabile, se la lobby è sbilanciata vuol dire che all'interno del cluster ci sono diversi stili di gioco e di giocatore, con diverse abilità. Ciò implica, nonostante questa "mancata fairness", che si ha comunque un bilanciamento che, paradossalmente, può portare a risultati migliori e più dinamici, con maggiore intrattenimento.
- Giocabilità bassa e Lobby Unfair: Se si ha questa situazione, molto probabilmente è un problema di mancate esperienze o troppa varianza all'interno del cluster che non permette nemmeno all'algoritmo KNN di trovare lobby giuste.

7 Deployment

La fase di deployment ha rappresentato l'ultima tappa del ciclo di vita del progetto **AIMm**, con l'obiettivo di integrare il modello di matchmaking, ritenuto il migliore sulla base delle analisi di coesione e fairness, in un ambiente di produzione in tempo reale. Il sistema è stato implementato in maniera modulare, in modo da potersi integrare agevolmente con l'infrastruttura esistente del videogioco, garantendo una gestione dinamica dei giocatori in coda e l'assegnazione immediata alle lobby in base alle valutazioni effettuate in fase di modeling.
Nel contesto del deployment, è stata posta particolare attenzione alla scalabilità e alla continuità del servizio. Il modello ha bisogno di analizzare il dataset del prodotto videoludico in cui "si trova" e in cui viene dispiegato, operando in tempo reale, assegnando i giocatori alle lobby sulla base di metriche di giocabilità e fairness, in modo da assicurare partite bilanciate e competitive. Al fine di monitorare costantemente le performance del sistema, sono stati implementati meccanismi di logging e monitoraggio, capaci di raccogliere dati quali tempi di risposta, composizione delle lobby e distribuzione degli score di fairness. Questi strumenti consentono di rilevare eventuali anomalie o variazioni nei parametri di performance, permettendo interventi correttivi tempestivi: gli interventi possono essere effettuati guardando direttamente grafici e punteggi globali, che guardando le anomalie presenti nelle vere e proprie esperienze e nelle lobby; il primo rappresenta un intervento molto più efficace, astraendo il problema, cercando di testare e gestire in maniera generale la situazione del modello del dataset, mentre il secondo intervento è più drastico, cercando di capire il problema nella sua natura, ma senza una risoluzione immediata. Pertanto quest'ultimo intervento è possibile solo da un esperto nel settore dell'analisi dei dati e nel settore videoludico.
I feedback ottenuti, sia in termini quantitativi che qualitativi, hanno permesso di affinare ulteriormente il modello, garantendo così un continuo miglioramento dell'esperienza di gioco. L'approccio iterativo adottato consente di integrare nuove informazioni e aggiornamenti, mantenendo il sistema allineato alle esigenze degli utenti e alle evoluzioni del contesto applicativo.

8 Conclusioni

AIMM propone un sistema innovativo per il matchmaking basato su algoritmi di clustering, migliorando il bilanciamento delle partite non troppo invasivo e l'esperienza dei giocatori.