SkillCraftI

Report per l'Esame di Fondameti di Machine Learning

Francesco Moretti

130301
Corso di Laurea
260877@studenti.unimore.it

Abstract

Abstract

Indice

1 Introduzione

Il mondo dei videogiochi è in grossa crescita negli ultimi anni, siamo arrivati al punto che alcuni possono essere già definiti come veri e propri Esports, questo ha contributo a rendere il mondo dei videogiochi sempre più competitivo e dinamico.

I giochi multiplayer dividono spesso i giocatori in rank o gradi, questo è necessario perché le partite siano bilanciate, ovvero affinché ogni giocatore trovi un avversario circa al suo livello. Per implementare questo sistema sono solitamente usati dei sistemi a punti che premiano le vittorie e puniscono le sconfitte.
Questo però non è un metodo propriamente efficace per dividere i giocatori in base alle loro capacità, un esempio lampante di questo è il fenomeno dello smurfing. Uno smurf account, usato appunto per fare smurfing, significa un account creato con lo scopo di nascondere la reale identità del giocatore.
Le ragioni dietro questa pratica possono essere diverse, da giocatori professionisti che vogliono usare un altro account per fare pratica e nascondere le proprie strategie, al comportamento ben più nocivo del "newb bashing", ovvero giocatori di livello medio-alto che si fanno un nuovo account per giocare contro i principianti e sembrare quindi dei fuoriclasse durante le partite.

La domanda a cui cercherò di rispondere in questo report è se esiste un metodo per creare i rank dei videogiochi migliore rispetto ai sistemi a punti, che possa sostituire il sistema attuale o almen integrarsi ad esso per fare il modo di piazzare un giocatore nel suo rank il più in fretta possibile.

Per farlo userò come esempio videogioco StarCraftII, un gioco di strategia in tempo reale a tema fantascientifico e anche uno dei principali Esports. Starcraft II usa un sistema a punti per dividere i sui giocatori in 7 diversi rank, che sono dal più basso al più alto: Bronzo, Argento, Oro, Platino, Diamante, Master e Grand Master. Oltre a questi nel dataset che userò è stato aggiunto anche un ottavo rank, quello dei giocatori professionisti, questo perché non tutti i giocatori a Grand Master sono abbastanza forti per competere nei tornei più importanti, ma solo una parte di essi ne è capace.

2 Analisi dei dati

In questa sezione saranno analizzati i dati di SkillCraftI per cercare di capire la struttura dei dati e correggere eventuali problemi che il dataset potrebbe avere. Per fare questo saranno usate le librerie Python Matplotlib e Seaborn, che permettono di graficare i dati per renderli più leggibili e interpretabili.

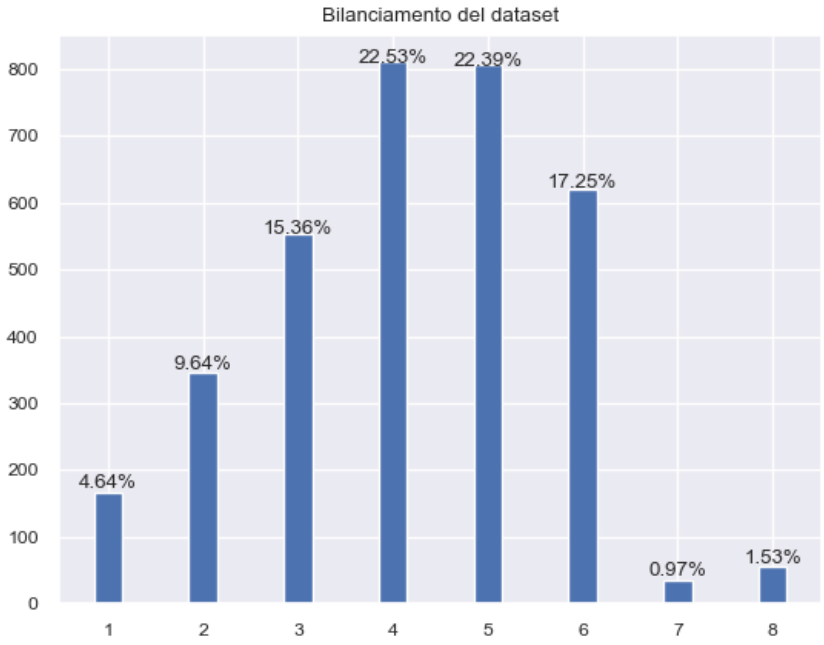
2.1 Bilanciamento del dataset

Le leghe di starcraft sono per loro natura sbilanciati, dal sito Liquipedia [1], un'enciclopedia online a tema videogiochi, si può leggere come sono strutturare le leghe del gioco e gli obbiettivi che si sono posti gli sviluppatori quando le hanno create.

La distribuzione a cui puntavano gli sviluppatori è come segue:

- Grand Master 1000 giocatori
- Master 2%
- Diamante 18%
- Platino 20%
- Oro 20%
- Argento 20%
- Bronzo 20%

Nella realtà le cifre sono destinate ad essere leggermente diverse, ma la proporzione è a grandi line corretta.
In più oltre, alle leghe del gioco, nel data set possiamo trovare anche i giocatori professionisti. Questa ulteriore divisione è stata aggiunta perché, anche se i grand master possono essere considerati un elite tra i giocatori, il divario le loro capacità e quelle dei professionisti è considerevole. Nel database Skillcraft invece le proporzioni sono indicate nel grafico a seguire

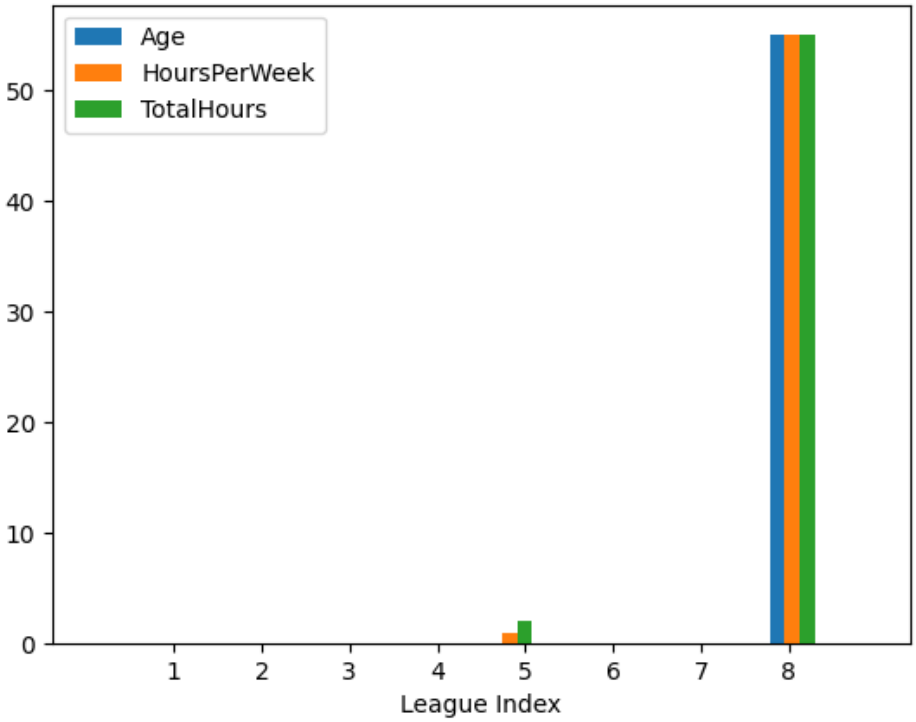


Come previsto il dataset è sbilanciato, questo comporta che bisognerà prestare attenzione a come vengono raggruppati i sample durante l'addestramento dei modelli. Oltre a questo, l'unica altra anomalia che è possibile individuare è che al contrario delle aspettative ci sono più professionisti che Grand Master, questo non è rispecchiato nella realtà ed è dovuto al modo in cui sono stati campionati i dati, ma ai fini della classificazione non dovrebbe creare grossi problemi.

2.2 Elementi nulli

Per la buona riuscita dell'addestramento è necessario rimuovere gli elementi nulli e il nostro dataset ne ha alcuni, ora cercheremo di capire quanti sono e come sono distribuiti per trovare una soluzione al problema.

Il grafico sotto rappresenta il numero totale di elementi nulli per ogni rank divisi per la feature a cui appartengono.



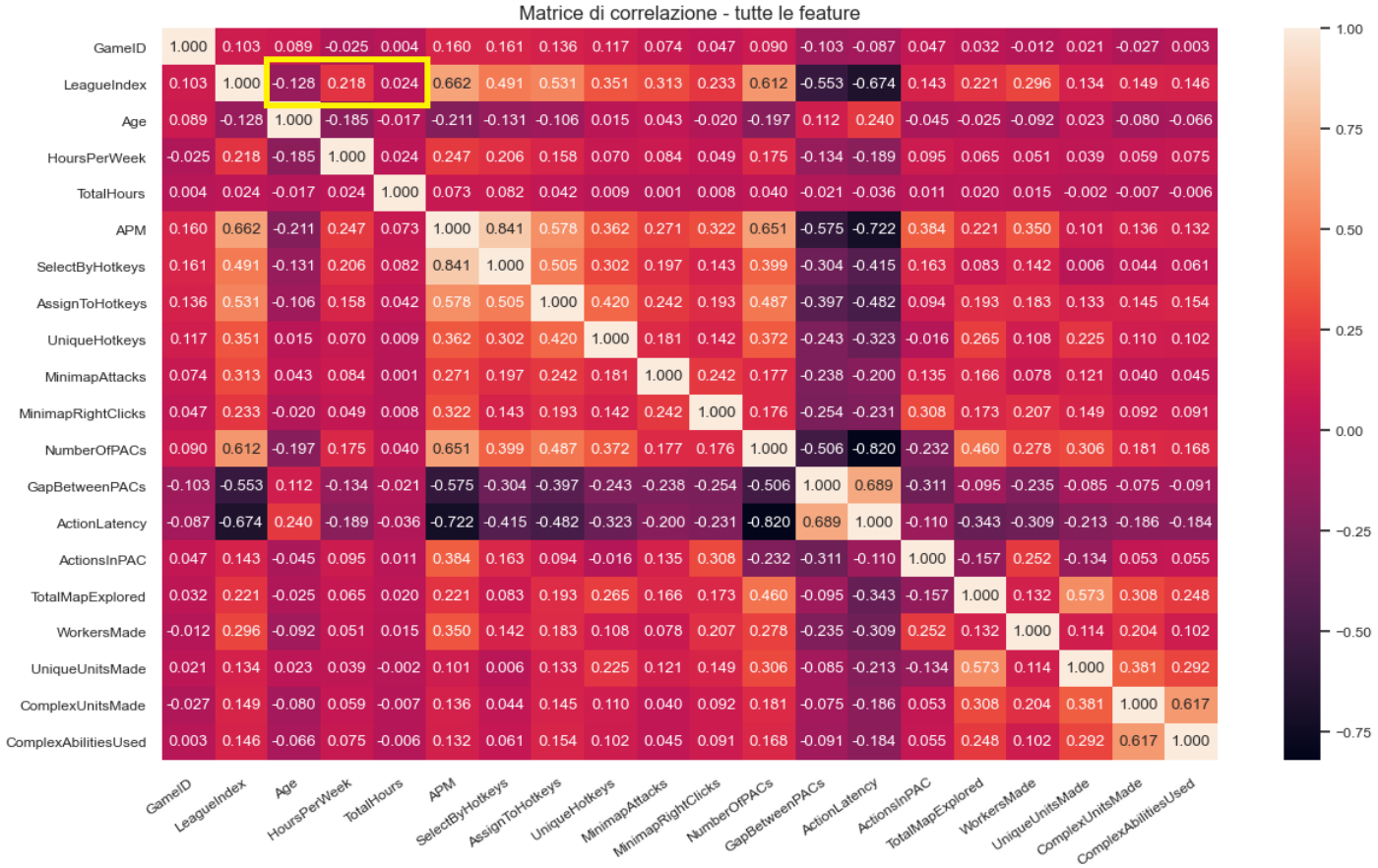
Dal grafico si può vedere che gli elementi nulli sono concentrati in 3 features che rappresentano: Età, ore di gioco settimanali e ore di gioco totali, in più la maggior parte è nella lega 8, ovvero i professionisti.

Esistono diverse soluzioni per gestire gli elementi nulli di un dataset, si possono eliminare le righe che ne contengono uno, ma controllando i dati dei professionisti si può notare che le tre colonne identificate prima sono tutte nulle in questa lega, di conseguenza eliminare

le righe con elementi nulli vorrebbe dire perdere completamente i dati sui professionisti.

Un'altra alternativa è quella di riempire i dati nulli con un valore che potrebbe estimarli, come la media degli altri valori di quella feature in quella lega, ma come visto prima non ci sono altri valori su cui costruire la stima e di conseguenza questa strada è impraticabile.

L'unica alternativa rimasta è eliminare le colonne con i valori nulli, ma come possiamo misurare l'effetto che questa decisione avrà sulla classificazione? Con una mappa di correlazione!
La mappa di correlazione rappresenta la relazione lineare tra i suoi elementi, da essa è possibile capire quanto due feature sono legate tra di loro e con la classe che vogliamo predire.



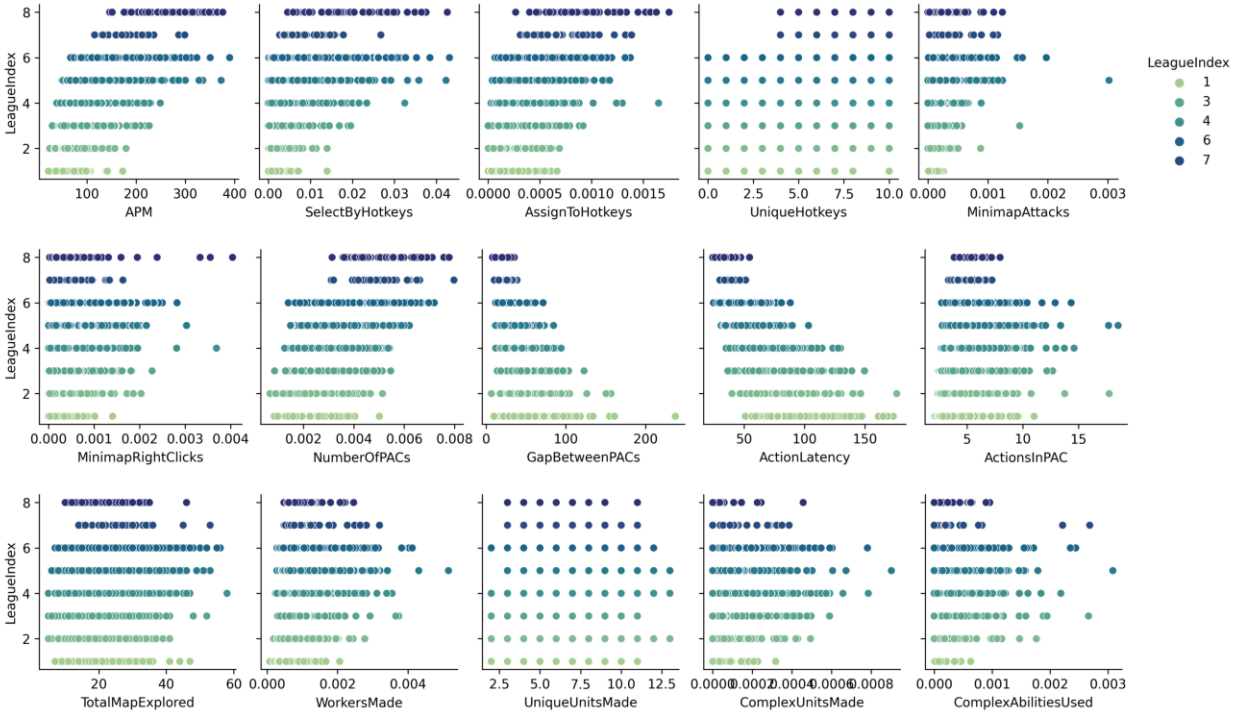
I valori evidenziati in giallo sono gli indici di correlazione delle feature con valori nulli, fortunatamente sono tendenzialmente bassi, quindi togliendoli non andremo ad influire sulla classificazione, ma potremmo addirittura migliorare le prestazioni di certi algoritmi di classificazione.

2.3 Relazioni tra le feature

Ora è il momento di controllare come sono fatte le feature, che relazioni hanno tra di loro e in particolare con il target da classificare.

Per farlo useremo due grafici principalmente il pairplot e la matrice di correlazione vista prima, il primo serve a definire il tipo di relazione, mentre la seconda da un'indicazione più quantitativa della relazione.

Il pairplot è essere usato per guardare le relazioni tra tutte le feature, ma nel nostro caso è sconveniente vista la grossa numerosità delle feature, i grafici verrebbero troppo piccoli di disordinati per essere utili. Quindi ho deciso si fare un pairplot con solo le relazioni delle feature con il nostro target, ovvero l'indice della lega.



Le cose che possiamo notare dai due grafici sono principalmente due, la prima è che guardando il pairplot possiamo notare che, alcune features hanno dei pattern nella loro relazione con il target, in particolare l'APM e l'ActionLatency, le due feature con la correlazione maggiore, sembrano seguire un andamento vagamente lineare.
Per quanto riguarda la mappa di correlazione invece possiamo notare che alcune features hanno una relazione forte tra di loro che è un indice di collinearità e multicollinearità, questi fenomeni sono abbastanza pericolosi perché rendono i modelli suscettibili al rumore delle feature.

Analizzando questo caso a fondo però è possibile rendersi conto che queste feature, sebbene collegate, non sono l'una la diretta conseguenza dell'altra. Prendendo ad esempio le APM

e il numero di selezioni fatte con tasti personalizzati, l'usare dei tasti personalizzati, se fatto male può portare anche al calo degli APM dovuto al tempo necessario per correggre gli errori, l'abilità di fare queste operazioni correttamente è un buon indice della qualità di un giocatore.
Visto che anche gli altri casi sono simili a questo, ovvero dove c'è una dipendenza, ma penso che la relazione che sussiste tra di loro sia significativa alla classificazione, ho deciso di lasciare tutte le feature per ora.

Per essere sicuro però ho deciso di inserire tra i modelli il random forest classifier, che è un classificatore che soffre poco a causa della collinearità/multicollinearità, e usarlo un po' come cartina torna sole, se le sue prestazioni finali saranno di molto maggiori rispetto a quelle degli altri modelli sarà necessario tornare indietro e selezionare meglio le feature.

3 Discussione dei modelli

In questa sezione parleremo di modelli, dei quali ho selezionato e perché, dei loro punti di forza e infine dei loro svantaggi.

Il nostro caso è un problema di classificazione con le seguenti caratteristiche:

- Multiclasse, che vuol dire che la variabile da predire può assumere più di due valori possibili.
- Lineare, anche se non tutte le feature hanno questo andamento, quelle principali a grandi linee lo seguono.
- Ci sono diverse feature con un indice di correlazione tendenzialmente basso, per questo potrebbe essere utile scegliere dei modelli che non hanno bisogno di molta feature selection e ridurre di conseguenza il lavoro da fare in preprocessing.

I modelli scelti devono poter funzionare in queste condizioni, in più mi piacerebbe che almeno una parte dei modelli sia leggera in modo da trovare un'effettiva applicazione in caso di una possibile integrazione in un videogioco.

3.1 modello1

sottosezione

4 Dettagli implementativi

sezione

4.1 dettaglio1

sottosezione

5 Risultati e discussione

sezione

5.1 risultato1

sottosezione

References

[1] Battle.net leagues. https://liquipedia.net/starcraft2/Battle.net_Leagues. Accessed: 2021-07-20.