Progetto Data Technology e Machine Learning

Matteo Colella, Matteo Angelo Costantini, Dario Gerosa

13/02/2018

Indice

Capitolo 1

Introduzione

In questo elaborato verrà mostrato il processo di integrazione di tre dataset appartenenti allo stesso dominio applicativo in modo da generare un terzo dataset a partire dal quale verranno applicate delle tecniche di clustering tramite l'algoritmo *K-Means* al fine di evidenziare delle strutture intrinseche presenti nei dati. I dataset utilizzati riguardano i risultati di 50 000 partite al videogioco competitivo *DOTA 2*.

DOTA 2 è un videogame in cui due squadre di cinque giocatori ciascuna si scontrano con l'obiettivo di distruggere le torri della squadra avversaria e infine la base nemica. All'inizio di una partita ogni giocatore deve scegliere un *champion* (personaggio) da utilizzare per tutta la durata della partita. Ogni campione ha delle abilità e statistiche che lo caratterizzano e, per poter vincere, i giocatori devono cercare di scegliere i campioni in modo da andare a comporre una squadra il più equilibrata possibile.

Il dataset principale contiene una tupla per ogni giocatore e per ogni partita per un totale di 500 000 tuple e 71 attributi riguardanti le performace del giocatore. Il secondo dataset contiene 50 000 tuple, una per ogni partita giocata, e 14 attributi tra cui la durata della partita e la squadra vincitrice. Infine, il terzo dataset contiene i 110 nomi dei personaggi del videogame associati ai loro rispettivi identificatori.

La relazione è suddivisa in due parti. Nel capitolo relativo a Data Technology vengono analizzati i dataset utilizzati e verranno valutate due dimensioni di qualità; successivamente viene descritto dettagliatamente il processo di integrazione dei tre dataset, verranno analizzati i dataset ottenuti alla fine del processo, verranno ricalcolate le due dimensioni di qualità già calcolate in precedenza evidenziandone le differenze rispetto alle stesse prima del processo di integrazione; infine verranno effettuate delle analisi descrittive sui dataset (prima e dopo l'integrazione). Nel capitolo relativo a Machine

Learning, dopo aver presentato il problema, vengono analizzate le feature del dataset evidenziandone in particolare le correlazioni al fine di scegliere le feature più importanti; successivamente vengono descritti i quattro esperimento condotti sui dataset integrati e verranno analizzati i risultati ottenuti; infine l'ultima sezione viene dedicata alle conclusioni.

Capitolo 2

Data Technology

I dataset iniziali sono stati reperiti da `kaggle.com` e sono stati costruiti sfruttando le API offerte dalla piattaforma *Steam*, sviluppatrice del videogioco. Tramite questa API è possibile scaricare informazioni riguardanti un match conoscendone l'id. A causa delle limitazioni delle API non è però possibile ottenere gli id di match con certe caratteristiche (ad esempio tutti i match giocati in una cerca giornata, o quelli riguardanti un certo livello dei giocatori), ma si può solo visitare il grafo dei giocatori, ottenendo gli id delle partite che hanno giocato e gli id dei giocatori nella stessa partita. Questo porta sicuramente ad uno sbilanciamento del dataset limitando le informazioni a partite relative a giocatori con lo stesso livello di abilità.

I dataset scelti per l'integrazione sono tre e tutti in formato csv, per questo motivo abbiamo deciso di rappresentarli attraverso un modello relazionale. Per la manipolazione abbiamo deciso di utilizzare il DBMS `SQLite` e la libreria `pandas` di Python. In particolare abbiamo deciso di utilizzare `SQLite` in quanto per le necessità del progetto e per la mole contenuta di dati da manipolare abbiamo ritenuto che sarebbe stato troppo dispendioso in termini di tempo l'utilizzo di altri DBMS più completi (Postgres, MySQL...) per avere minimi vantaggi in termini prestazionili. Infatti, il DBMS è stato utilizzato solo a fini esplorativi ed è risultati comunque estremamente veloce sia nell'importazioni che nelle query. Tutto il lavoro di integrazione è stato invece svolto utilizzando `pandas`.

2.1 Descrizione dei dataset

2.1.1 Dataset matches

Questo è il dataset principale contenente 500 000 istanze. Ogni istanza contiene informazioni relative alle performance di un *champion* in uno specifico *match*. Gli attributi inizialmente presenti in questo dataset sono 71, per praticità elenchiamo di seguito solo quelli di interesse, ovvero gli attributi rimasti dopo la prima fase di integrazione descritta successivamente.

- match_id : l'identificatore della partita;
- player_slot : l'identificatore del giocatore all'interno della partita. I giocatori della squadra *Radiant* hanno valori che vanno da 0 a 4, mentre i giocatori della squadra *Dire* hanno valori che vanno da 128 a 132.
- hero_id : l'identificatore del *champion*;
- gold : l'ammontare delle monete del giocatore alla fine della partita;
- gold_spent : l'ammontare delle monete spese durante la partita;
- xp_creep : l'esperienza guadagnata dal giocatore uccidendo i *creep*;
- xp_hero : l'esperienza base guadagnata dal giocatore;
- xp_roshan : l'esperienza guadagnata dal *Roshan The Immortal*;
- xp_other : l'esperienza guadagnata in altri modi;
- kills : numero di uccisioni in quella partita;
- deaths : numero di morti del giocatore in quella partita;
- assists : numero di partecipazioni alle uccisioni;
- lasthits : numero di *creep* uccisi;
- hero_damage : danno inflitto dal giocatore;
- hero_healing : danni curati dal giocatore;
- tower_damage : danni inflitti alle torri;

match_id	player_slot	hero_id	gold	gold_spent	xp_creep	xp_hero	xp_roshan	...
0	0	86	3261	10960	5440.0	8840.0	NaN	...
0	1	51	2954	17760	8440.0	14331.0	2683.0	...
0	2	83	110	12195	8112.0	6692.0	NaN	...
0	3	11	1179	22505	14230.0	8583.0	894.0	...
0	4	67	3307	23825	14325.0	15814.0	NaN	...
0	128	106	476	12285	12259.0	8502.0	NaN	...
0	129	102	317	10355	9417.0	5201.0	NaN	...
0	130	46	2390	13395	13396.0	6853.0	NaN	...
0	131	7	475	5035	4038.0	4798.0	NaN	...
0	132	73	60	17550	10471.0	6659.0	NaN	...

...	xp_other	kills	deaths	assists	last_hits	hero_damage	hero_healing	tower_damage
...	83.0	9	3	18	30	8690	218	143
...	671.0	13	3	18	109	23747	0	423
...	453.0	0	4	15	58	4217	1595	399
...	293.0	8	4	19	271	14832	2714	6055
...	62.0	20	3	17	245	33740	243	1833
...	1.0	5	6	8	162	10725	0	112
...	1.0	4	13	5	107	15028	764	0
...	244.0	4	8	6	208	10230	0	2438
...	27.0	1	14	8	27	4774	0	0
...	933.0	1	11	6	147	6398	292	0

Tabella 2.1: dataset matches

2.1.2 Dataset matches_stats

Questo dataset contiene per ogni *match* la sua durata e la squadra vincitrice. In particolare utilizziamo il primo attributo per calcolare delle feature in relazione al tempo giocato. Mentre il secondo viene utilizzato per calcolare la percentuale di vittoria di ogni campione. Come per la tabella precedente, anche in questo caso vengono riportati solo gli attributi di interesse. Sono quindi stati omessi 14 attributi.

match_id	duration	radiant_win
0	2375	True
1	2582	False
2	2716	False
3	3085	False
4	1887	True
5	1574	True
6	2124	True
7	2328	True
8	2002	False
9	2961	False

Tabella 2.2: dataset matches_stats

- match_id : l'identificatore del match giocato;
- duration : la durata in secondi del match;
- radiant_win : indica se la squadra *radiant* ha vinto o meno la partita.

2.1.3 Dataset hero_names

Questo dataset associa ad ogni identificatore numerico del campione sia il suo nome in inglese sia un identificatore leggibile. Utilizziamo questo dataset per associare il nome del campione ad ogni istanza del dataset principale.

name	hero_id	localized_name
npc_dota_hero_antimage	1	Anti-Mage
npc_dota_hero_axe	2	Axe
npc_dota_hero_bane	3	Bane
npc_dota_hero_bloodseeker	4	Bloodseeker
npc_dota_hero_crystal_maiden	5	Crystal Maiden
npc_dota_hero_drow_ranger	6	Drow Ranger
npc_dota_hero_earthshaker	7	Earthshaker
npc_dota_hero_juggernaut	8	Juggernaut
npc_dota_hero_mirana	9	Mirana
npc_dota_hero_morphling	10	Morphling

Tabella 2.3: dataset hero_names

- name: identificatore testuale del campione;
- hero_id: identificatore numerico;
- localized_name: nome in inglese;

2.2 Analisi di dimensioni di qualità (prima dell'integrazione)

Sono state effettute le seguenti due misure di qualità:

- Completezza
- Consistenza

Di seguito verranno descritti i passi svolti per l'analisi delle due misure in relazione ai dataset iniziali.

2.2.1 Completezza

Per valutare la completezza dei dataset di partenza abbiamo calcolato le percentuali di valori nulli per ogni attributo del dataset, per ogni tupla e sul totale di valori per ognuno dei dataset di partenza. Tutte le operazioni sono state svolte utilizzando la libreria `pandas` di Python. Di seguito i valori riportati per ogni tabella.

attribute	# NaN	% NaN
match_id	0	0.0
account_id	0	0.0
hero_id	0	0.0
player_slot	0	0.0
gold	0	0.0
gold_spent	0	0.0
kills	0	0.0
deaths	0	0.0
assists	0	0.0
denies	0	0.0
last_hits	0	0.0
stuns	0	0.0
hero_damage	0	0.0
hero_healing	0	0.0
tower_damage	0	0.0
item_0	0	0.0
item_1	0	0.0
item_2	0	0.0
item_3	0	0.0
item_4	0	0.0
item_5	0	0.0
level	0	0.0
leaver_status	0	0.0
xp_hero	1883	0.38
xp_creep	68	0.01
xp_roshan	320438	64.00
xp_other	21036	4.21
gold_other	94897	18.98
gold_death	6299	1.26
gold_buyback	352859	70.57
gold_abandon	479366	95.87
gold_sell	102593	20.52
gold_destroying_structure	19675	3.94
gold_killing_heros	1565	0.31
gold_killing_creeps	294	0.059
gold_killing_roshan	240264	48.05
gold_killing_couriers	403021	80.60
unit_order_none	499994	99.99
unit_order_move_to_position	42	0.01
unit_order_move_to_target	54550	10.91
unit_order_attack_move	113271	22.65
unit_order_attack_target	105	0.02
unit_order_cast_position	2269	0.45
unit_order_cast_target	16674	3.33
unit_order_cast_target_tree	35712	7.14
unit_order_cast_no_target	892	0.18
unit_order_cast_toggle	401211	80.24
unit_order_hold_position	137277	27.46
unit_order_train_ability	73	0.01
unit_order_drop_item	288667	57.73
unit_order_give_item	394631	78.93
unit_order_pickup_item	244658	48.93
unit_order_pickup_rune	74517	14.90
unit_order_purchase_item	57	0.01
unit_order_sell_item	111866	22.37
unit_order_disassemble_item	485446	97.09
unit_order_move_item	4264	0.85
unit_order_cast_toggle_auto	463670	92.73
unit_order_stop	412425	82.49
unit_order_taunt	500000	100.0
unit_order_buyback	352233	70.45
unit_order_glyph	273230	54.65
unit_order_eject_item_from_stash	468736	93.75
unit_order_cast_rune	499991	99.99
unit_order_ping_ability	160852	32.17
unit_order_move_to_direction	496449	99.29
unit_order_patrol	500000	100.0
unit_order_vector_target_position	500000	100.0
unit_order_radar	500000	100.0
unit_order_set_item_combine_lock	500000	100.0
unit_order_continue	500000	100.0

Tabella 2.4: Completezza di matches

matches

Completezza della tabella:

- Numero di tuple: 500 000
- Numero di attributi: 71
- Numero di valori null: 11 038 020
- Numero di valori della tabella: 35 500 000
- Completezza tabella: 68,91%
- Completezza tabella utilizzata: 95,07%

Attributo	# NaN	% NaN
match_id	0	0.00
player_slot	0	0.00
hero_id	0	0.00
gold	0	0.00
gold_spent	0	0.00
xp_creep	68	0.01
xp_hero	1883	0.38
xp_roshan	320438	64.09
xp_other	21036	4.21
kills	0	0.00
deaths	0	0.00
assists	0	0.00
last_hits	0	0.00
hero_damage	0	0.00
hero_healing	0	0.00
tower_damage	0	0.00

Tabella 2.5: Completezza di matches ridotta ai soli attributi di interesse

heroes_names

Attributo	# NaN
name	0
hero_id	0
localized_name	0

Tabella 2.6: Completezza di heroes_names

matches_stats

attributo	# NaN
match_id	0
start_time	0
duration	0
tower_status_radiant	0
tower_status_dire	0
barracks_status_dire	0
barracks_status_radiant	0
first_blood_time	0
game_mode	0
radiant_win	0
negative_votes	0
positive_votes	0
cluster	0

Tabella 2.7: Completezza di matches_stats

Osservazioni

Come si può notare sono presenti un gran numero di attributi con valore nulla, ma se restringiamo il campio ai soli attributi di nostro interesse (quelli mostratai in tabella 2.1) il numero di valori null dimunisce sensibilemente e il contributo maggiore è dovuto ai valori mancanti per l'attributo `xp_roshan` che contribuisce per il 64% dei null di tutta la tabella. Un'altra osservazione che abbiamo potuto fare è relativa alla natura di questi valori: abbiamo infatti notato che le API utilizzate non includevano gli attributi relativi all'esperienza ottenuta qualora il suo valore fosse stato 0 e che quindi tutti i valori null sono in realtà 0. Questa ipotesi è anche confermata dal fatto che, per ognuno dei quattro attributi, non sono presenti valori pari a 0 in tutto il dataset. Per questo motivo abbiamo deciso di correggere i valori mancanti assegnandogli 0 durante la fase di integrazione dei dataset.

2.2.2 Consistenza

Durante il processo di integrazione ci siamo accorti che dopo aver effettuato la join tra il dataset `heroes_names` contenente il nome dei campioni con il dataset dei match, ottenevamo solo 499 963 match anziché 500 000 a causa di una discrepanza tra gli id dei campioni contenuti in `heroes_names` e quelli contenuti in `matches`. Indagando ulteriormente abbiamo notato che per quanto riguarda la prima tabella il range dei valori variava tra 1 e 113, mentre nella seconda tabella tra 0 e 112. Inizialmente questo ci ha portato ha pensare che gli id fossero semplicemente shiftati di un valore. Per confermare quest'ipotesi abbiamo quindi provato ad effettuare una join incrementando di uno il valore degli id della tabella `heroes_stats`. Il risultato della join è stato però differente da quanto ci aspettavamo, 482 312 corrispondenze, meno che nel caso precedente. Questo ha portato ad escludere l'ipotesi dello shift degli id. Analizzando ulteriormente i due dataset abbiamo notato che in entrambi i dataset non era mai presente l'id 24. Abbiamo quindi ipotizzato che l'eroe identificato con l'id 113 non sia presente nel dataset `matches` perché il campione associato è stato rilasciato in un periodo successivo a quello di svolgimento dei match contenuti nel dataset. Analogamente, ipotizziamo che il campione con id 0 non sia presente nella tabella `heroes_names` perché è stato rimosso dal gioco.

Non sono state quindi effettuate modifiche agli id ed abbiamo deciso di utilizzare i 499 963 match anziché 500 000 dato che il numero di tuple perse è irrisorio rispetto alla dimensione del dataset.

2.3 Processo di integrazione

Il processo di integrazione è stato effettuato importando i dataset in Python e utilizzando la libreria `pandas` e si è suddiviso nelle seguenti fasi di integrazione:

1. Eliminati alcuni attributi del dataset `matches`.
2. Eliminati alcuni attributi del dataset `matches_stats`.
3. Calcolate feature aggiuntive per il dataset `matches` sfruttando le informazioni di `matches_stats`.
4. Rimossi dalla tabella `matches` tutti gli attributi utilizzati per il calcolo delle feature del passo precedente.

5. Aggiunti i nomi dei campioni nella tabella `matches`.
6. Raggruppamento per *champion* dei dati di `matches` in un nuovo dataset `heroes_stats`.
7. Normalizzazione delle features.
8. Inversione dei valori relativi all'attributo `deaths`.

Fase 1: Questa prima fase è stata eseguita subito dopo aver scaricato i dataset in modo da limitare la quantita' di dati da dover manipolare inutilmente. Le feature iniziali si possono trovare nella tabella `matches` dei dataset di partenza e non le abbiamo elencate in questa relazione. Il risultato di questa fate è mostrato in tabella 2.1.

Fase 2: Analogamente alla fase precedente, abbiamo eliminato da `matches_stats` tutte le feature che non intendiamo utilizzare nelle fasi successive. Anche queste feature si possono trovare nella rispettiva tabella nei dataset iniziali. Il risultato di questa fase è mostrato in tabella 2.2.

Fase 3: In questa fase sono stati calcolati gli attributi `gold_over_duration`, `won`, `xp_over_duration` e `picks`. Il primo attributo indica il numero medio di monete guadagnate al secondo ed è stato calcolato dividendo la somma degli attributi `gold` e `gold_spent` per il numero di secondi del match (valore presente nella tabella `matches_stats`). Il secondo attributo fa riferimento all'esperienza guadagna in gioco dal giocatore ed è stata calcolata in modo analogo all'attributo precendente: abbiamo sommato i valori di esperienza dei quattro attributi relativi (descritti nella sottosezione 2.1.1) e li abbiamo divisi per la durata della partita. Infine abbiamo aggiunto un attributo che indica se il giocatore ha vinto o meno la partita. Per fare questo abbiamo utilizzato l'attributo `radiant_win` del dataset `matches_stats` per sapere se la squadra *Radiant* ha vinto o meno la partita e abbiamo poi controllato la squadra di appartenza del giocatore (attributo `player_slot` di `matches`) per sapere se ha vinto o meno.

Fase 4: In questa fase abbiamo rimosso tutti gli attributi intermedi utilizzati per calcolare le feature della *Fase 3* e che non hanno più alcuna utilità.

Fase 5: Dopo aver rimosso gli attributi inutili abbiamo effettuato una join tra la tabella `matches` ottenuta al passo successivo con la tebella `heroes_names` per aggiungere i nomi dei campioni al dataset principale. Abbiamo deciso di utilizzare i nomi locallizati

in inglese e non gli identificatori testuali o numerici per riferirci al campione per una questione di praticità.

Fase 6: Una volta ottenuti tutti gli attributi di interesse abbiamo raggruppato in base al campione le tuple di `matches` e calcolato le statistiche medie degli attributi a disposizione per ogni campione ed è stato aggiunto l'attributo `picks` ovvero la percentuale di volte in cui un certo campione viene scelto da un giocatore. In questo modo abbiamo creato il dataset `heroes_stats`.

Fase 7: A questo punto abbiamo deciso di nornalizzare le feature in modo da lavorare con attributi con valori nello stesso range.

Fase 8: In quest'ultima fase abbiamo sostituito i valori dell'attributo `deaths` con il valore ottenuto calcolando $1 - deaths$. Abbiamo preso questa decisione in quanto questo attributo è l'unico che a valori maggiori fa corrispondere una performance del campione minore. Per tutti gli altri attributi a valori più alti corrispondono prestazioni migliori.

2.4 Descrizione dei dataset finali

2.4.1 Dataset matches_stats

Questo è il dataset principale dopo il processo di integrazione e contiene 499 963 match. Non è quello su cui verrà fatto Machine Learning, ma è quello che ha permesso di creare il dataset su cui faremo il clustering, ovvero `heroes_stats_normalized`. Il dataset contiene 11 attributi, tra cui il nome del campione ed è riportato in tabella 2.8. Gli attributi sono i seguenti:

- hero_name : il nome del campione;
- kills : numero di uccisioni in quella partita;
- deaths : numero di morti del giocatore in quella partita;
- assists : numero di partecipazioni alle uccisioni;
- lasthits : numero di *creep* uccisi;

- hero_damage : danno inflitto dal giocatore;
- hero_healing : danni curati dal giocatore;
- tower_damage : danni inflitti alle torri;
- gold_over_duration : monete medie guadagnate al secondo;
- xp_over_duration : esperienza media guadagnata al secondo;
- won : il giocatore ha vinto la partita;

hero_name	kills	deaths	assists	last_hits	hero_damage	...
Rubick	9	3	18	30	8690	...
Rubick	4	14	15	42	9757	...
Rubick	2	4	18	143	8208	...
Rubick	4	7	16	20	6273	...
Rubick	14	7	23	66	17818	...

...	hero_healing	tower_damage	gold_over_duration	xp_over_duration	won
...	218	143	5.987789	6.047579	True
...	325	280	3.200608	4.691658	False
...	0	151	5.091923	7.961724	False
...	0	309	5.391186	5.889596	True
...	0	403	7.090737	10.257404	True

Tabella 2.8: dataset matches_stats

2.4.2 Dataset heroes_stats

Questo è il dataset ottenuto aggregando le informazioni di `matches` e contiene le statistiche medie per ogni campione. Abbiamo utilizzato questo dataset per l'analisi descrittiva dei dati in quanto i valori non sono ancora stati normalizzati e rappresentano la media effettiva del rispettivo attributo. Il dataset contiene 13 attributi tra cui il nome del campione, ed è riportato in tabella 2.9. Gli attributi sono i seguenti:

- hero_name : il nome del campione;
- kills : numero medio di uccisioni del *champion*;
- deaths : numero medio di morti del *champion*;
- assists : numero medio di partecipazioni alle uccisioni;
- lasthits : numero medio di *creep* uccisi;

- hero_damage : danno medio inflitto dal giocatore;
- hero_healing : danni medio curati dal giocatore;
- tower_damage : danni medi inflitti alle torri;
- gold_over_duration : monete medie guadagnate al secondo;
- xp_over_duration : esperienza media guadagnata al secondo;
- won_perc : percentuale di partite vinte dal *champion*;
- lost_perc : percentuale di partite perse dal *champion*;
- picks : percentuale di volte in cui il *champion* è stato usato in una partita.

hero_name	kills	deaths	assists	last_hits	hero_damage	hero_healing	...
Abaddon	5.005740	6.278550	12.647432	84.188822	9660.720846	2242.337462	...
Alchemist	6.679019	7.173369	11.647562	265.160847	14438.700906	10.312735	...
Ancient Apparition	4.826151	7.492818	13.356286	48.930994	8888.788094	213.265956	...
Anti-Mage	7.340996	5.449979	6.722222	298.673585	10162.878033	135.156769	...
Axe	8.525321	9.365138	10.498153	136.269724	11945.333188	14.217779	...
Bane	4.766941	7.721112	11.811986	32.764591	7838.933803	167.141794	...
Batrider	6.055977	8.263757	14.039848	103.942125	10802.548387	75.833966	...
Beastmaster	5.455987	7.912768	13.011895	111.848533	10093.486122	123.457573	...
Bloodseeker	9.657307	9.190122	9.746279	165.670162	13487.634641	740.771651	...
Bounty Hunter	6.731047	8.261446	10.033417	40.083910	9326.336228	771.840130	...

...	tower_damage	gold_over_duration	xp_over_duration	won_perc	lost_perc	picks
...	1009.606949	5.819416	6.967741	30.543807	69.456193	0.662049
...	2585.172758	11.168695	9.629661	27.089484	72.910516	1.964745
...	298.453280	4.936875	5.418059	26.047683	73.952317	1.350700
...	3081.401767	8.875521	9.685556	23.893146	76.106854	1.879339
...	453.616605	5.989381	7.394753	22.647251	77.352749	0.920268
...	381.018801	4.624201	5.624299	24.441833	75.558167	0.510638
...	268.393738	5.581703	6.986072	21.726755	78.273245	0.210816
...	1491.495638	5.975999	7.124649	26.804124	73.195876	0.252219
...	1449.624831	6.830154	8.259095	23.815968	76.184032	0.591244
...	430.565141	6.581745	6.141938	25.732372	74.267628	1.358701

Tabella 2.9: dataset heroes_stats

2.4.3 Dataset heroes_stats_normalized

Questo è il dataset su cui abbiamo effettuato il clustering e contiene gli stessi attributi della tabella `heroes_stats`, ma con i valori normalizzati. Il risultato è riportato in tabella

2.10. Essendo gli attributi quelli del dataset precedente con i valori normalizzati, le descrizioni sono le stesse della sottosezione 2.4.2.

hero_name	kills	deaths	assists	last_hits	hero_damage	hero_healing	...
Abaddon	0.251889	0.806403	0.550658	0.203026	0.276095	0.444800	...
Alchemist	0.414291	0.618982	0.470296	0.875475	0.520827	0.001890	...
Ancient Apparition	0.234459	0.552074	0.607631	0.072017	0.236556	0.042163	...
Anti-Mage	0.478540	0.979947	0.074435	1.000000	0.301816	0.026664	...
Axe	0.593486	0.159916	0.377916	0.396546	0.393115	0.002665	...
Bane	0.228712	0.504257	0.483511	0.011946	0.182782	0.033010	...
Batrider	0.353821	0.390600	0.662570	0.276425	0.334581	0.014892	...
Beastmaster	0.295588	0.464115	0.579951	0.305803	0.298262	0.024342	...
Bloodseeker	0.703352	0.196573	0.317486	0.505791	0.472113	0.146838	...
Bounty Hunter	0.419340	0.391084	0.340564	0.039143	0.258968	0.153003	...

...	tower_damage	gold_over_duration	xp_over_duration	won_perc	lost_perc	picks
...	0.225532	0.207991	0.298929	0.914078	0.085922	0.134519
...	0.661543	1.000000	0.732669	0.613589	0.386411	0.455325
...	0.028733	0.077323	0.046419	0.522963	0.477037	0.304108
...	0.798866	0.660475	0.741777	0.335541	0.664459	0.434292
...	0.071671	0.233156	0.368507	0.227161	0.772839	0.198109
...	0.051581	0.031028	0.080025	0.383271	0.616729	0.097232
...	0.020414	0.172795	0.301915	0.147088	0.852912	0.023397
...	0.358887	0.231174	0.324496	0.588765	0.411235	0.033593
...	0.347300	0.357640	0.509345	0.328827	0.671173	0.117082
...	0.065292	0.320860	0.164370	0.495534	0.504466	0.306078

Tabella 2.10: dataset heroes_stats_normalized

2.5 Analisi di dimensioni di qualità (dopo l'integrazione)

Dopo il processo di integrazione le misure di completezza e consistenza risultano perfette al 100%. Infatti, come descritto nella sezione 2.2.1, dopo aver valutato la natura dei valori null presenti nella tabella, abbiamo deciso di sostituirli con il valore 0 e quindi siamo riusciti ad ottenere una completezza totale della tabella. Tutti gli attributi calcolati successivamente, durante il processo di integrazione, utilizzavano quindi le informazione di dataset completi ed è stato dunque possibile calcolarne il valore per ogni tupla.

Per quanto riguarda la consistenza, dopo aver eliminato le tuple di `matches` contenenti un eroe non disponibile nella tabella `heroes_names` abbiamo portato la consistenza al 100% e da quel momento non abbiamo introdotto modifiche agli identificatori e le uniche

modifiche alle tuple effettuate riguardano l'aggiunta di nuovi attributi calcolati sulla base di quelli disponibili e quindi, anche in questo caso, abbiamo mantenuto la consistenza del dataset.

Attributo	# NaN	Percentuale
hero_name	0	0.0
kills	0	0.0
deaths	0	0.0
assists	0	0.0
last_hits	0	0.0
hero_damage	0	0.0
hero_healing	0	0.0
tower_damage	0	0.0
gold_over_duration	0	0.0
xp_over_duration	0	0.0
won	0	0.0

Tabella 2.11: Completezza di matches

Completezza della tabella matches:

- Numero di tuple: 499 963
- Numero di attributi: 11
- Numero di valori null: 0
- Numero di valori della tabella: 5 499 593
- Completezza tabella: 100,0%

Attributo	# NaN	Percentuale
hero_name	0	0.0
kills	0	0.0
deaths	0	0.0
assists	0	0.0
last_hits	0	0.0
hero_damage	0	0.0
hero_healing	0	0.0
tower_damage	0	0.0
gold_over_duration	0	0.0
xp_over_duration	0	0.0
won_perc	0	0.0
lost_perc	0	0.0
picks	0	0.0

Tabella 2.12: Completezza di heroes_stats

Completezza della tabella heroes_stats:

- Numero di tuple: 110

- Numero di attributi: 13
- Numero di valori null: 0
- Numero di valori della tabella: 1 430
- Completezza tabella: 100,0%

Attributo	# NaN	Percentuale
hero_name	0	0.0
kills	0	0.0
deaths	0	0.0
assists	0	0.0
last_hits	0	0.0
hero_damage	0	0.0
hero_healing	0	0.0
tower_damage	0	0.0
gold_over_duration	0	0.0
xp_over_duration	0	0.0
won_perc	0	0.0
lost_perc	0	0.0
picks	0	0.0

Tabella 2.13: Completezza di heroes_stats_normalized

Completezza della tabella heroes_stats_normalized:

- Numero di tuple: 110
- Numero di attributi: 13
- Numero di valori null: 0
- Numero di valori della tabella: 1 430
- Completezza tabella: 100,0%

2.6 Analisi descrittive dei dati

E' stata effettuata un'analisi descrittiva sui dataset, individuando alcuni indici statistici riguardanti i dati a disposizione. L'obiettivo delle analisi fatte è stato quello di cercare delle caratteristiche riguardanti i vari campioni. I risultati ottenuti sono mostrati nei grafici di questa sezione.

2.6.1 Percentuale di vittorie

La prima statistica che abbiamo calcolato riguarda la percentuale di vittorie di ogni campione. Come si può vedere in figura 2.1 la maggior parte dei campioni sono bilanciati e vincono in media la metà delle partite. Come ci aspettavamo, si possono anche notare, però alcuni campioni leggermente sbilanciati che raggiungono percentuali di vittoria di quasi il 60% e del 45%. Tra i primi troviamo i cosiddetti campioni *in meta*, ovvero quei personaggi che in un determinato periodo risultano sbilanciati rispetto agli altri e vengono giocati maggiormente e dei campioni probabilmente meno popolari che vengono utilizzati solo da giocatori più capaci e che quindi portano ad avere percentuali di vittoria maggiori rispetto alla media.

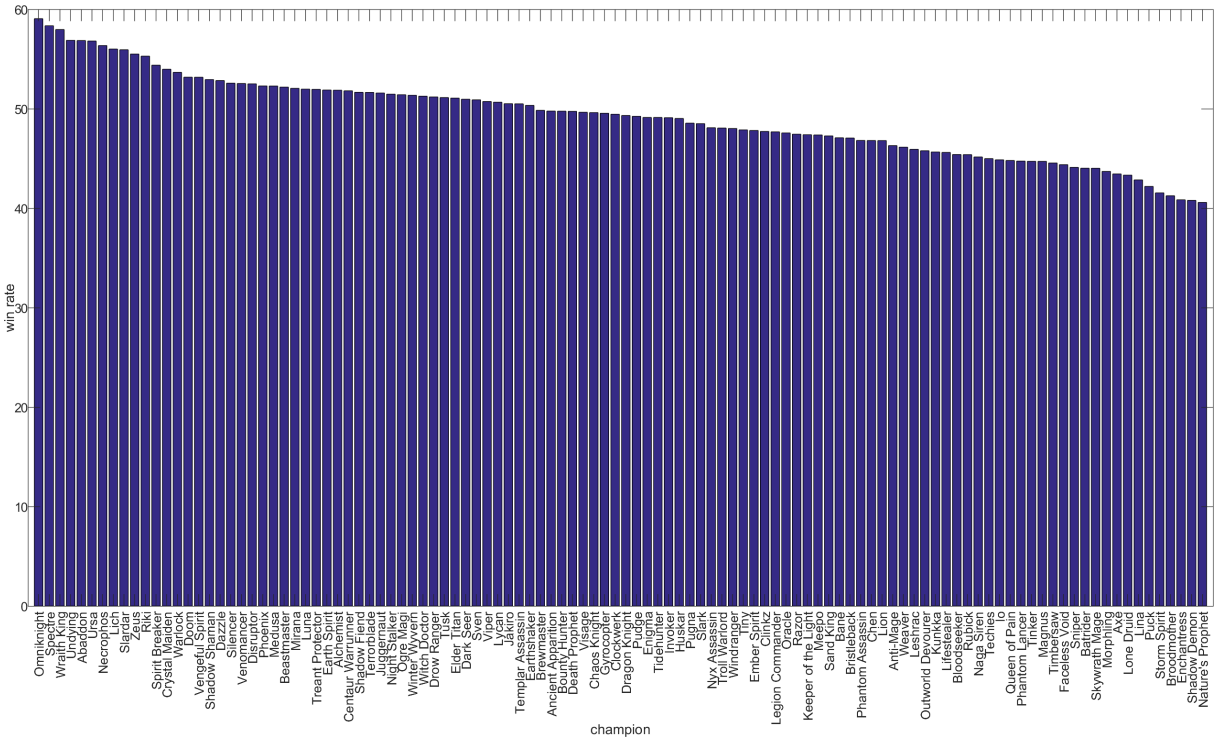

Figura 2.1: Percentuale di vittoria in relazione al campione

2.6.2 Percentuale di vittorie in funzione del team

Successivamente ci siamo chiesti se il team potesse influire sulla percentuale di vittoria. Questa domanda deriva dal fatto che la mappa non è perfettamente simmetrica e questo potrebbe portare un particolare campione ad avere dei vantaggi a giocare in una squadra piuttosto che nell'altra. Le statistiche di vittoria in funzione del team si possono vedere nelle figure 2.2 e 2.3. Come si può vedere dai grafici, per la maggior

parte dei campioni non vi è differenza nella vittoria giocando in un team piuttosto che nell'altro. Anche in questo caso, però ci sono alcune eccezioni: per 7 campioni su 110 si ha un significativo vantaggio a giocare una partita nella squadra *Radiant*, mentre per 4 campioni su 110 si ha un vantaggio a giocare nella squadra dei *dire* con la differenza maggiore di circa il 10% per il campione *Chen*.

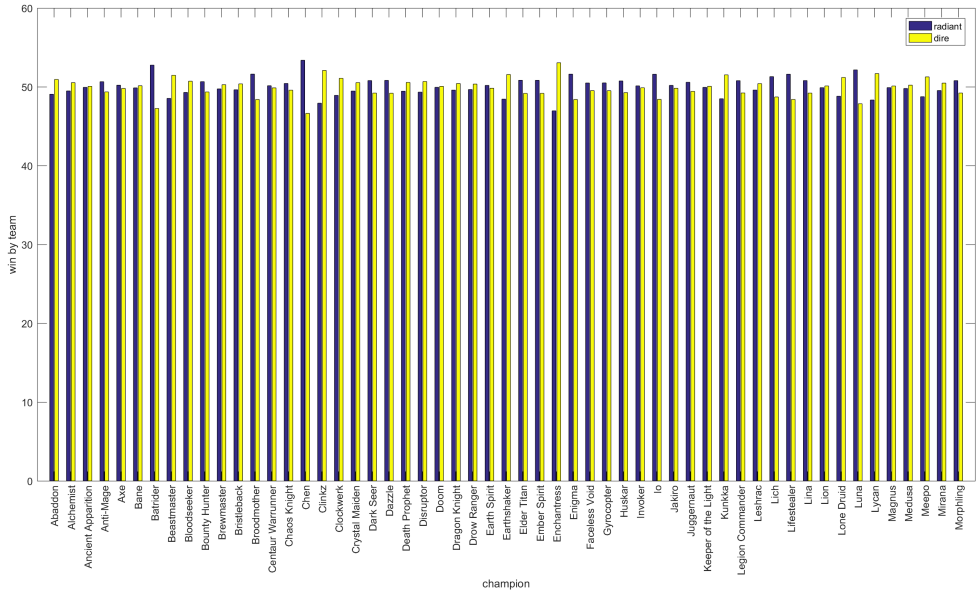

Figura 2.2: Percentuale di vittoria di ogni campione in funzione del team

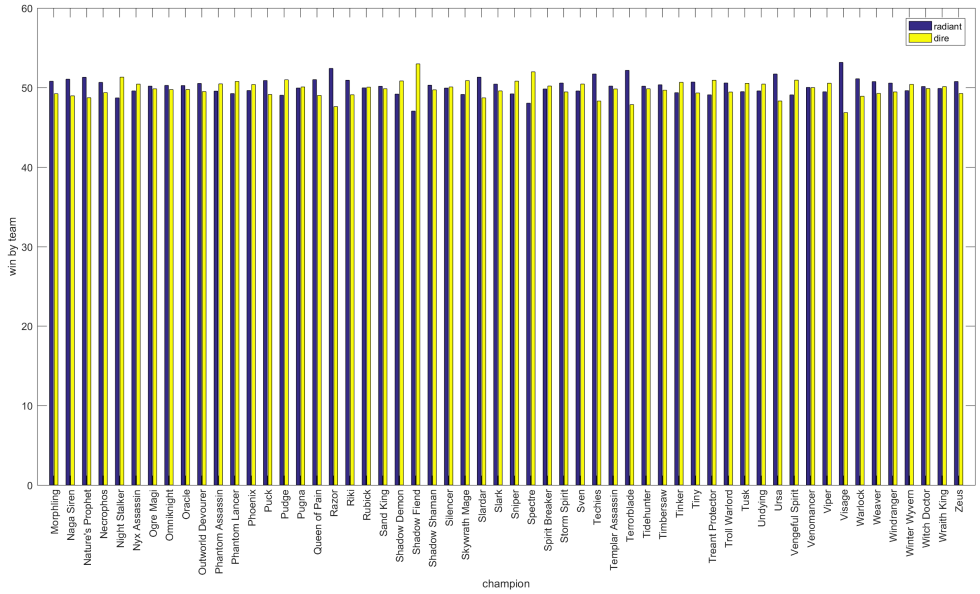

Figura 2.3: Percentuale di vittoria di ogni campione in funzione del team

2.6.3 Percentuale di scelta di un campione

Abbiamo poi calcolato la percentuale di *pick* di un campione, ovvero la probabilità che un campione venga scelto da almeno un giocatore in una partita. Come si può vedere in figura 2.4 i campioni *Windranger* e *Shadow Fiend* vengono scelti molto più frequentemente degli altri, ma contrariamente a quanto ci aspettavamo, non sono quelli con una percentuale di vittoria più alta. Si ha poi un secondo gruppo di 25-30 campioni particolarmente popolari e un ultimo gruppo di campioni meno giocati.

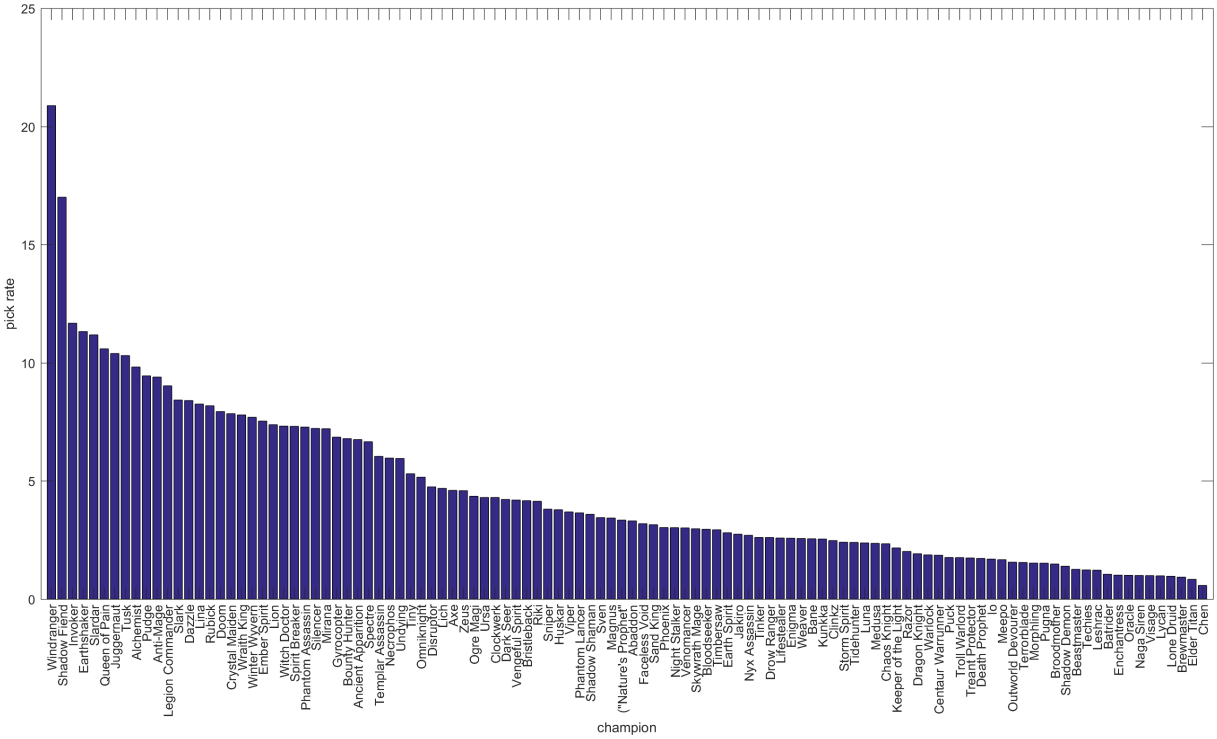

Figura 2.4: Percentuale di scelta di un campione

2.6.4 Durata media delle partite

Un'altra statistica calcolata è la durata media di una partita calcolata utilizzando il dataset `matches_stats` e riportata in figura 2.5 e risulta essere di circa quaranta minuti. In figura 2.6 è riportata invece la durata media di una partita per ogni campione. Come ci aspettavamo, la scelta del campione non influisce nella durata, ma anche in questo caso di sono due accezioni anche se la differenza è comunque ridotta.

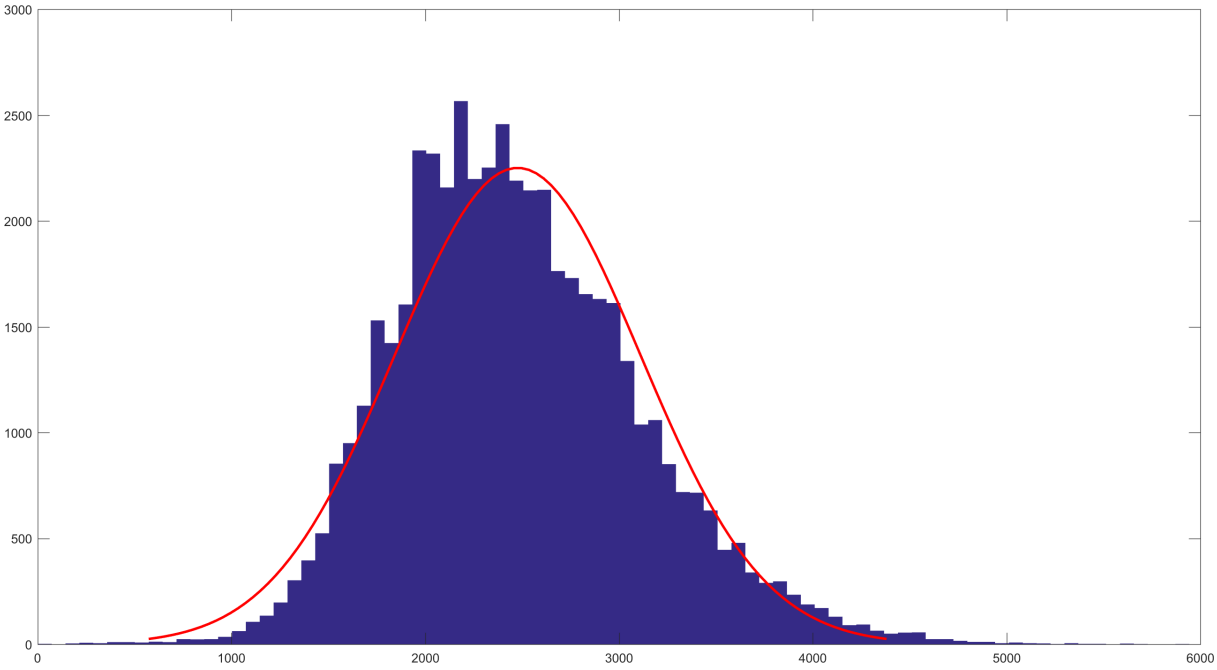

Figura 2.5: Durata media di una partita

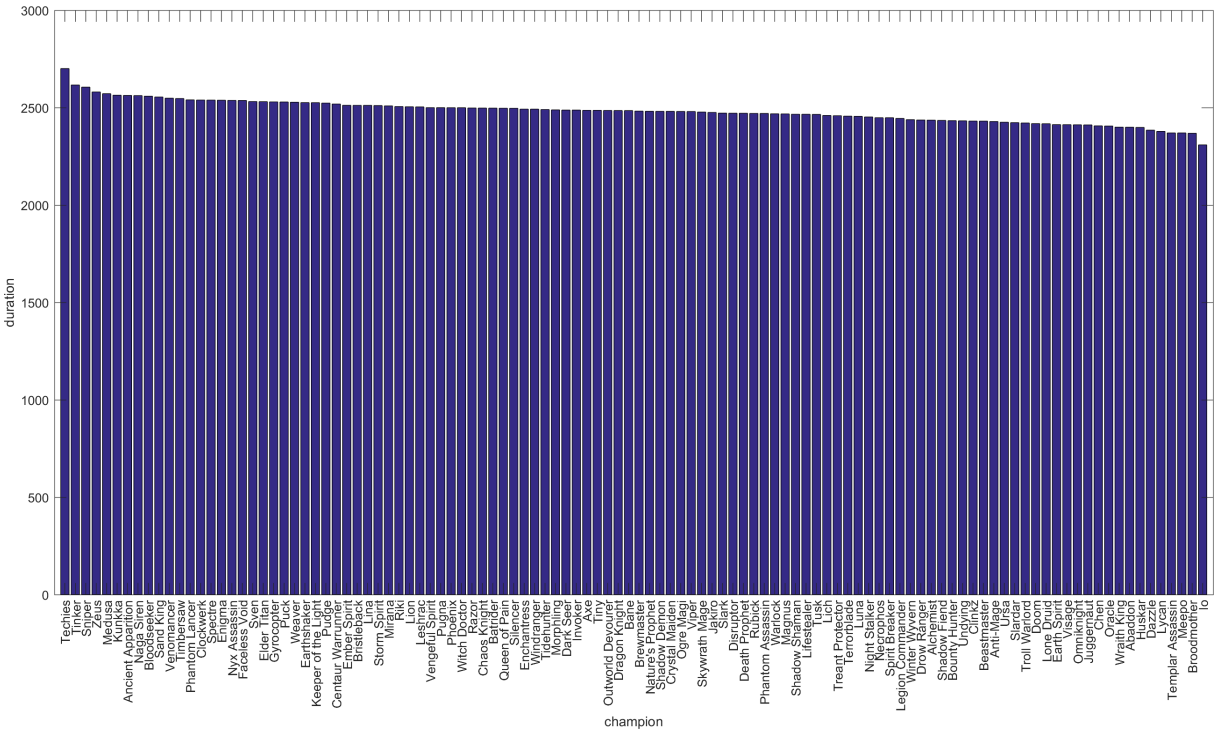

Figura 2.6: Durata media di una partita in funzione del campione scelto

2.6.5 Percentuale di vittoria in funzione della durata di una partita

Abbiamo poi provato a calcolare quale fosse la percentuale di vittoria di ogni campione in funzione della durata di una partita in modo da evidenziare *champion* che tendono a performare meglio durante il *late game* (la fase finale del gioco) e durante l'*early/mid game* (le fasi iniziali e intermedie). Le partite sono state suddivise in bucket di cinque minuti ciascuna dopo aver eliminato una partita outlier della durata di 16 000 secondi. Abbiamo poi calcolato la percentuale di vittoria di ogni campione per ognuno dei bucket calcolati e trovato una curva che meglio approssimasse i valori trovati. Per quest'ultima parte abbiamo utilizzato la libreria `numpy` ed effettuato il fit con un polinomio di terzo grado. Il metodo di ricerca degli outlier non è estremamente affidabile e, come si può vedere in figura 2.7, per alcuni campioni la curva trovata non rispecchia correttamente le percentuali (in particolare si può osservare quella del campione *Spirit Breaker*). Per questo motivo sono stati plottati anche i punti utilizzati per effettuare la regressione.

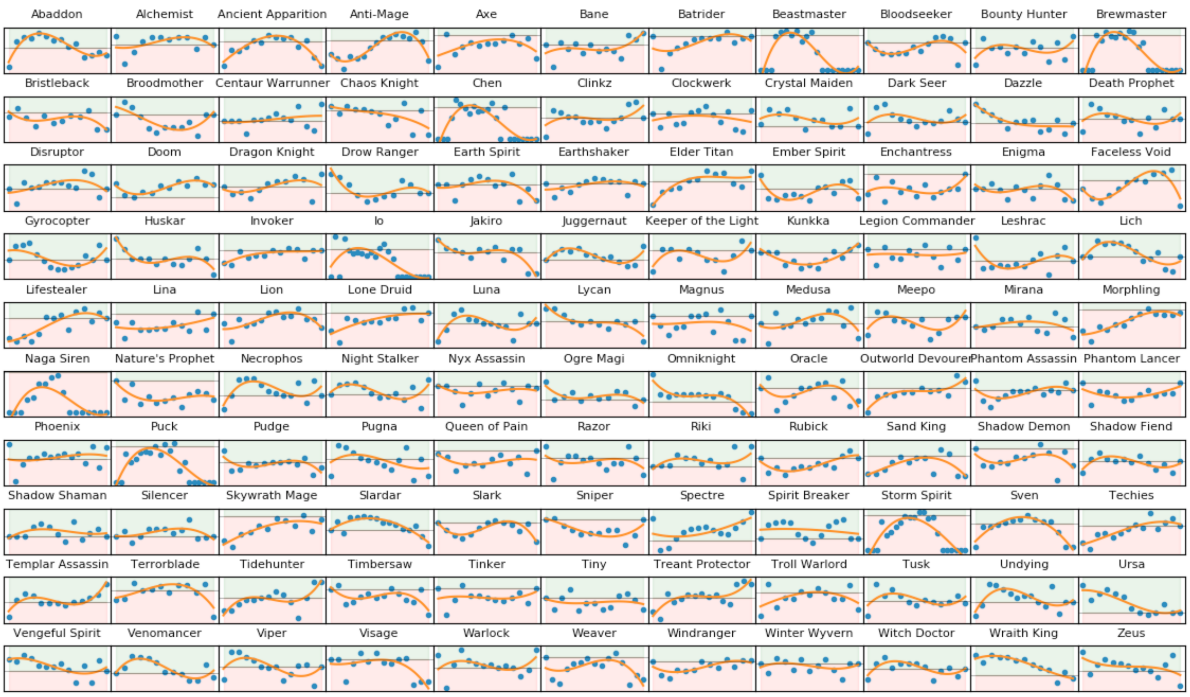

Figura 2.7: Percentuale di vittoria di ogni campione in funzione della durata della partita

Capitolo 3

Machine Learning

3.1 Descrizione del problema

L'obiettivo di questo progetto è quello di cercare, tramite algoritmi di clusterizzazione, in che modo sono strutturati i dati da analizzare. Per completare questo task è stato utilizzato l'algoritmo di unsupervised-learning k-means. Il dataset che è stato utilizzato contiene tutte le caratteristiche dei personaggi del videogioco DOTA2. Di conseguenza, applicare un algoritmo di unsupervised-learning su un dataset di questa tipologia equivale a suddividere in categorie di "forza" i vari champions.
Essendo la natura del gioco quella di rendere il gameplay il più equilibrato possibile, individuare delle categorie di personaggi non è una pratica facile come può sembrare, anche perché la "forza" di un personaggio è stimata osservando diverse caratteristiche.

3.2 Analisi del dataset

Il dataset utilizzato per il clustering è composto da 13 features di cui una non è stata utilizzata in quanto contiene il nome del personaggio. Le altre 12 features rappresentano:

- hero_name : il nome del campione;
- kills : numero medio di uccisioni del *champion*;
- deaths : numero medio di morti del *champion*;
- assists : numero medio di partecipazioni alle uccisioni;
- lasthits : numero medio di *creep* uccisi;

- hero_damage : danno medio inflitto dal giocatore;
- hero_healing : danni medio curati dal giocatore;
- tower_damage : danni medi inflitti alle torri;
- gold_over_duration : monete medie guadagnate al secondo;
- xp_over_duration : esperienza media guadagnata al secondo;
- won_perc : percentuale di partite vinte dal *champion*;
- lost_perc : percentuale di partite perse dal *champion*;
- picks : percentuale di volte in cui il *champion* è stato usato in una partita.

Per eseguire una clusterizzazione più fedele possibile tutte le features sono state normalizzate, ed è stata analizzata la correlazione pairwise. In figura 3.1 è visualizzabile la matrice di correlazione. Come si può notare dalla matrice, la feature `gold_over_duration` e `xp_over_duration` presentano un indice di correlazione di 0.91, mentre la feature `won_perc` e `lost_perc` presentano un indice di correlazione di -1. Per questo motivo e per diminuire la dimensionalità del dataset è stato deciso di eliminarle. Sono stati però eseguiti esperimenti anche con queste features per avere una più completa osservazione dei risultati.

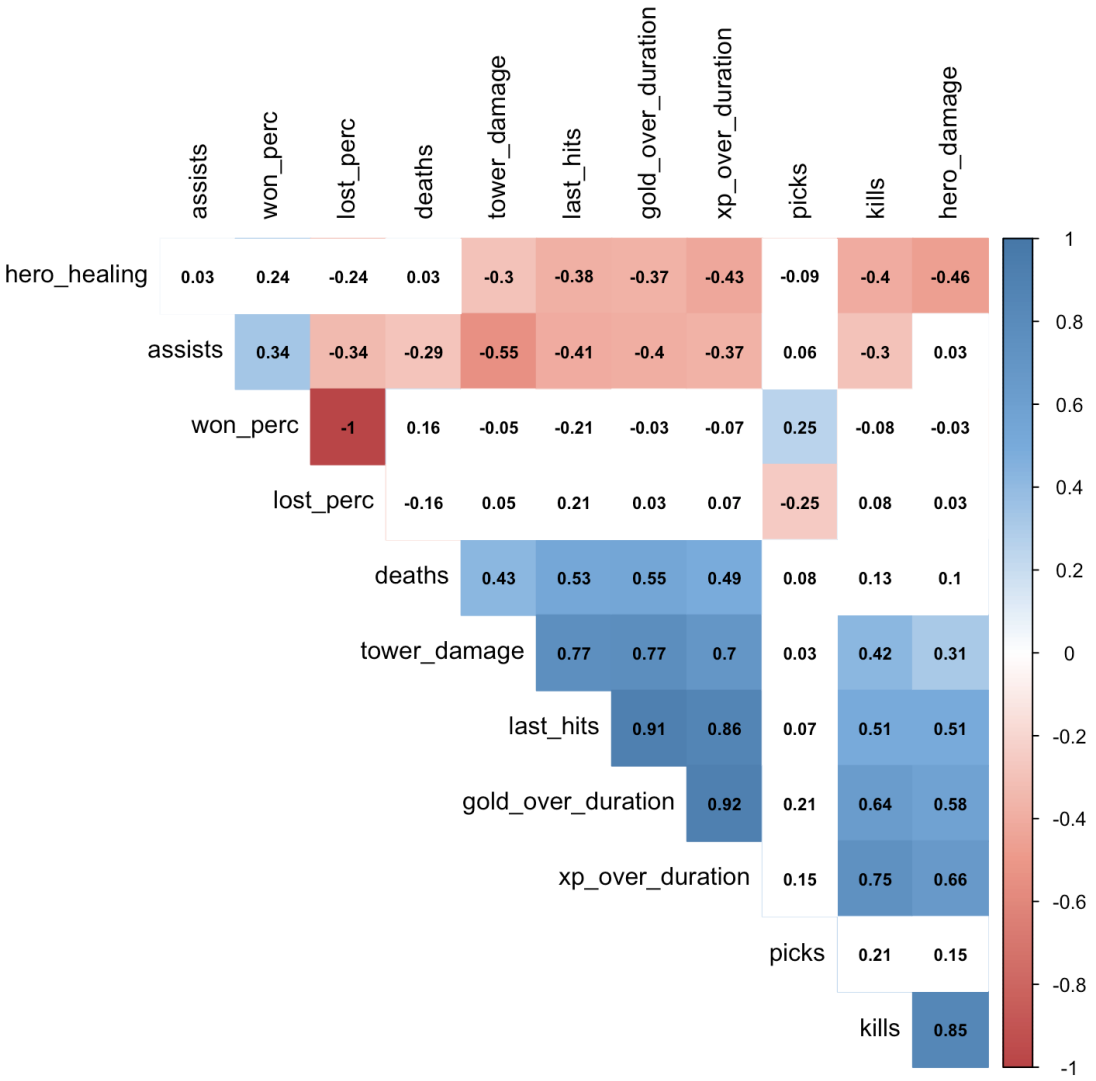

Figura 3.1: Matrice di correlazione delle feature del dataset

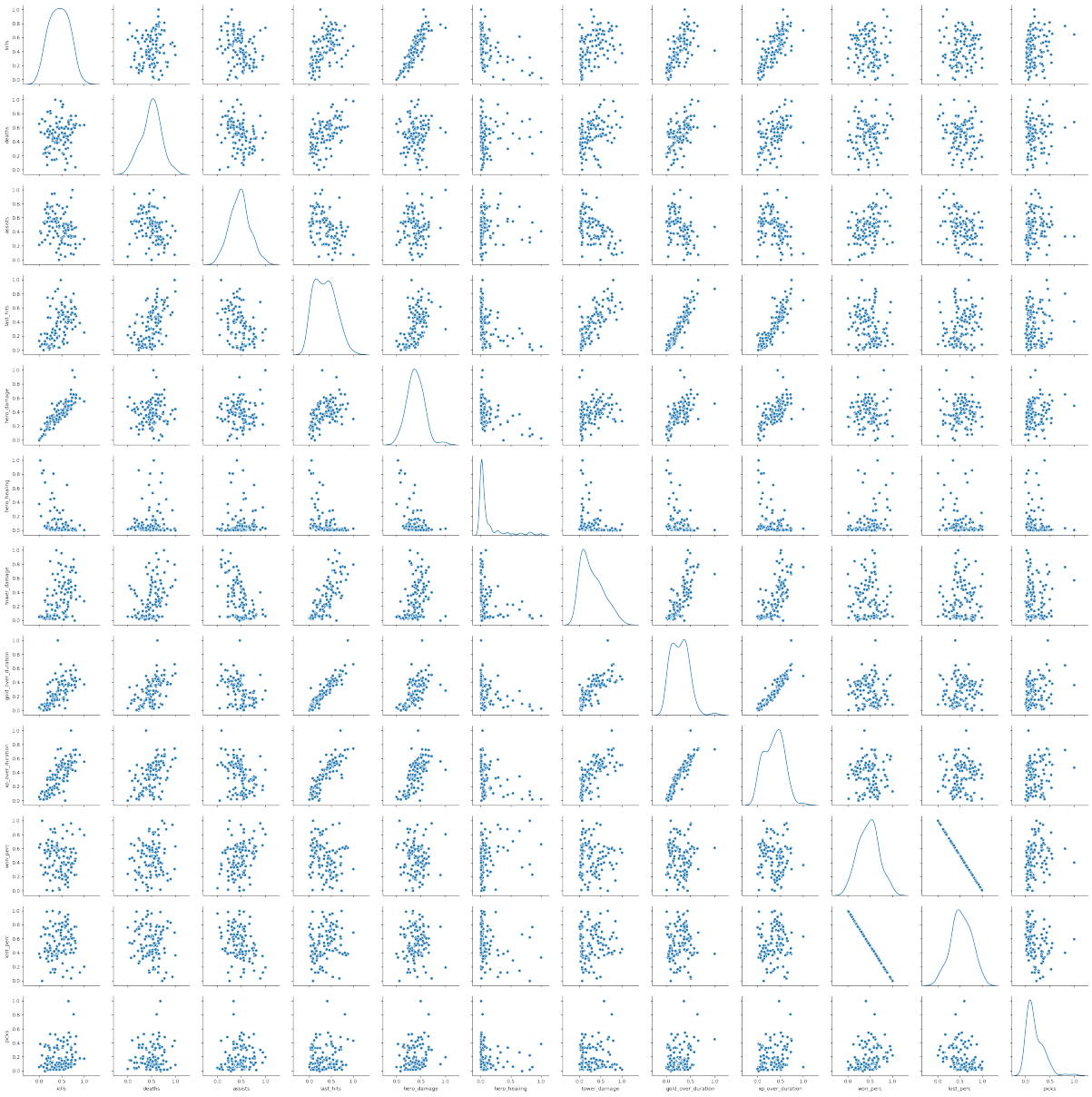

Figura 3.2: Plot delle feature a coppie

3.3 Esperimenti condotti

3.3.1 Utilizzo di tutte le feature eccetto lost_perc

Un primo esperimento è stato svolto effentuando il clustering utilizzando tutte le feature ad eccezione di `lost_perc` in quanto perfettamente correlata a `win_perc` essendo stata calcolata in funzione di quest'ultima. In figura 3.3 si può osservare come il numero

	1	2	3
1	0	0,81	1,17
2	0,81	0	0,70
3	1,17	0,70	0

	1	2	3
distanza	0,28	0,22	0,25

Tabella 3.1: Nella prima tabella la casella (i,j) rappresenta la distanza tra il centroide i e il centroide j. Nella seconda invece troviamo la somma normalizzata delle distanze di tutti i punti di un cluster dal proprio centroide

di cluster ottimale sia tre. Si può inoltre notare che già da quattro cluster in poi, la precisione del modello diminuisce sensibilmente per poi riamuntare leggermente solo da ventisette cluster in poi, valore eccessivamente alto soprattutto considerando le 110 istanze del dataset.
Il KMeans ha individuato due cluster che contengono ognuno circa il 40% e il 50% delle istanze, mentre il terzo cluster contiene solo il restante 10%, come si può notare dal grafico delle silhouette in figura 3.5. In figura 3.6 si possono vedere i valori dei centroidi di ogni cluster. In particolare si può notare che il terzo cluster è caratterizzato da valori relativamente bassi per la maggior parte degli attributi ad eccezione di `hero_healing` e di `won_perc`. Questo ci suggerisce che il terzo cluster contiene campioni il cui ruolo principale è quello di *support*, ovvero tendono ad assistere gli altri giocatori durante i combattimenti piuttosto che a combattere contro gli avversari. Per quanto riguarda gli altri due cluster, invece, possiamo vedere una divisione in *"statistiche più alte"* e *"Statistiche più basse"*.
Dalla matrice di dissimilarità 3.4 possiamo notare come il primo cluster sia molto dissimile dagli altri due, entrambi caratterizzati per statistiche relativamente basse (ad eccezione di cure e vittorie).
Un'altra considerazione che possiamo fare è che, utilizzare dei *support* favorisce il gioco di squadra e aumenta di conseguenza la percentuale di vittorie.

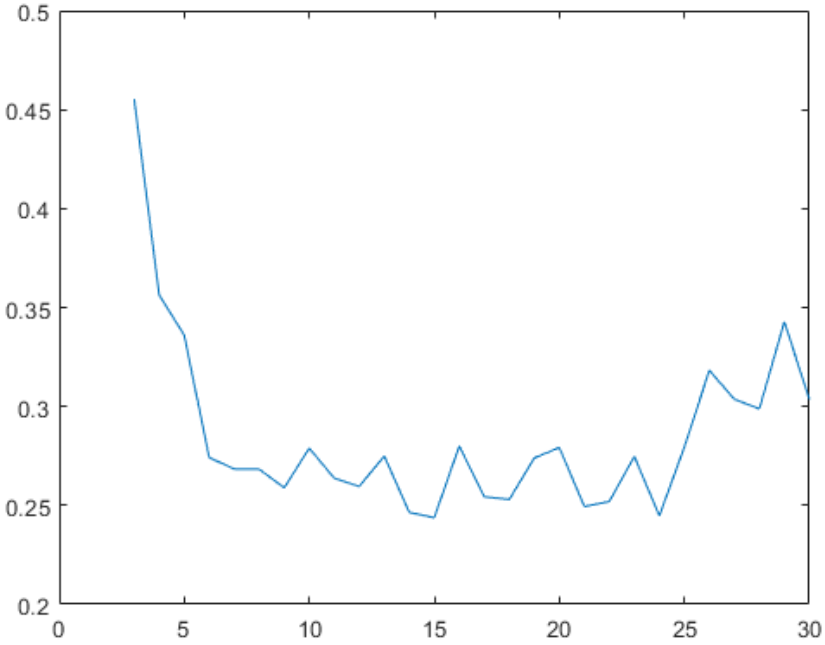

Figura 3.3: Silhouette media al variare del numero di cluster utilizzando tutte le feature tranne lost_perc

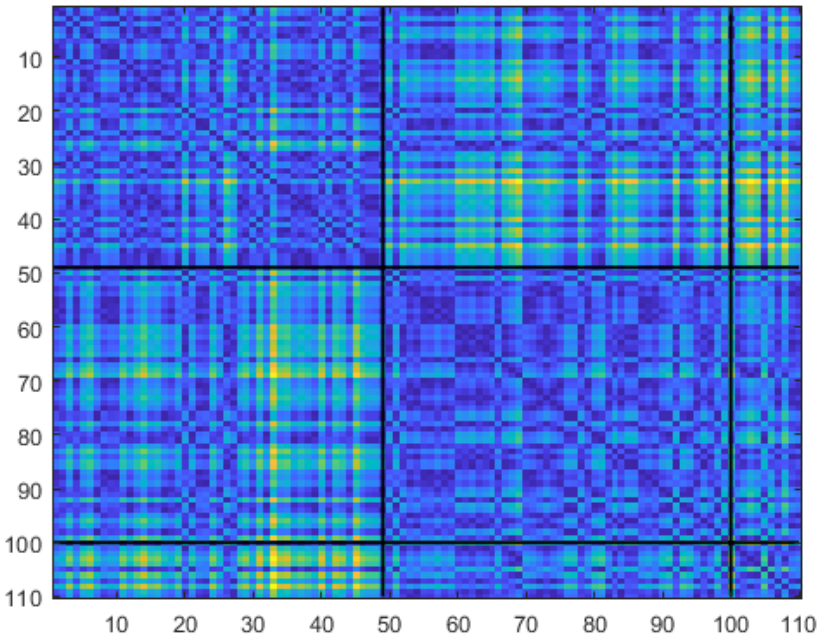

Figura 3.4: Matrice di dissimilarità del clustering ottimo utilizzando tutte le feature tranne lost_perc

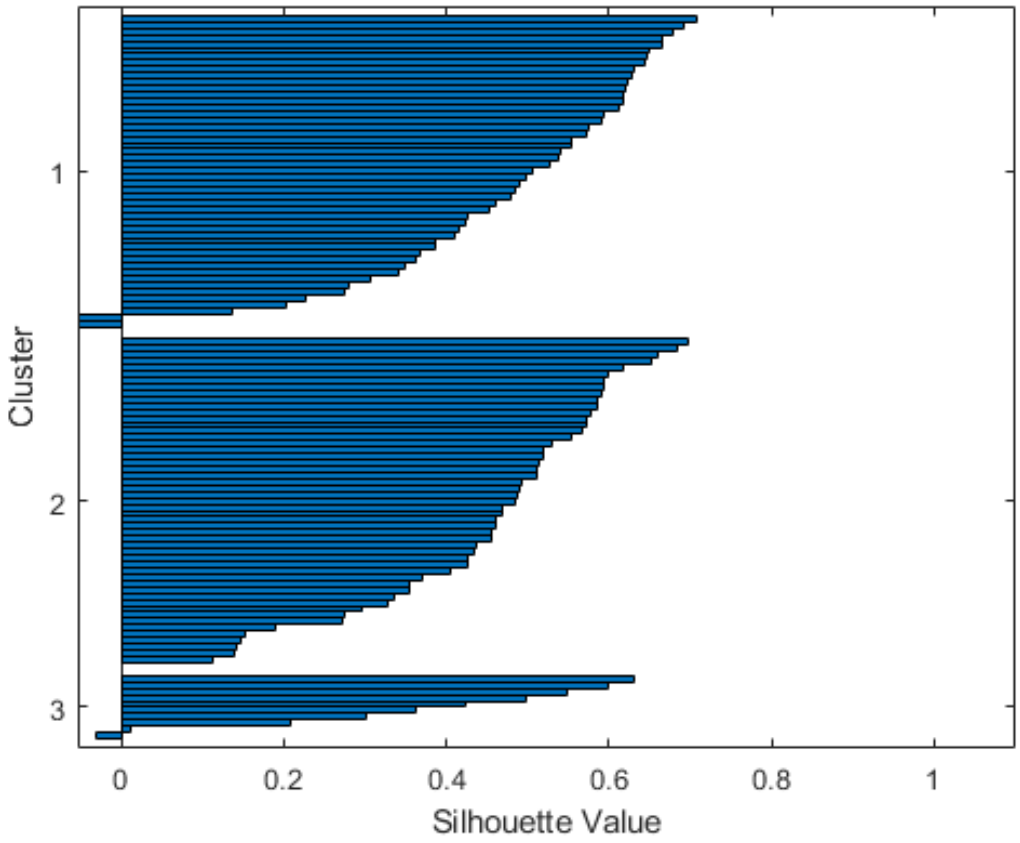

Figura 3.5: Silhouette del clustering ottimo utilizzando tutte le feature tranne lost_perc

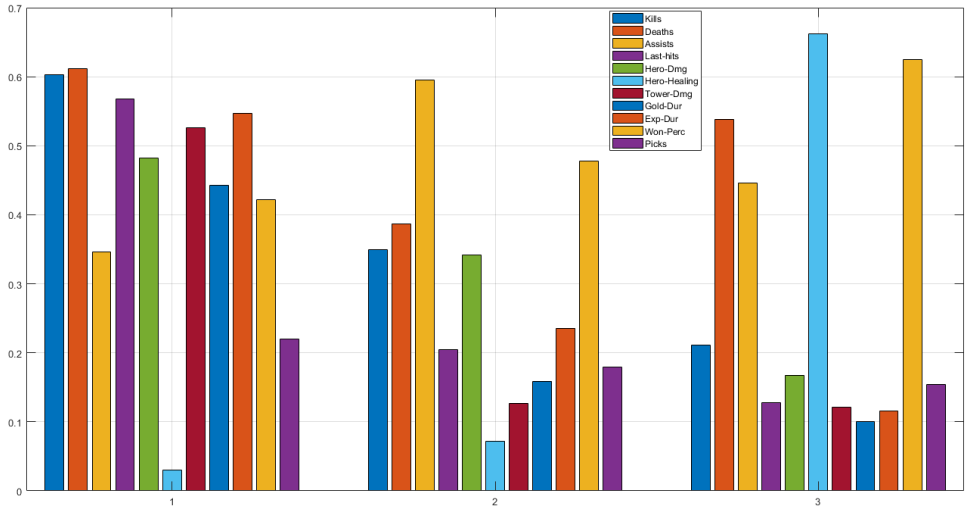

Figura 3.6: Barplot delle istanze clusterizzate utilizzando tutte le feature tranne lost_perc

3.3.2 Utilizzo di tutte le feature eccetto gold_over_duration e lost_perc

	1	2	3
1	0	0,70	0,77
2	0,70	0	1,12
3	0,77	1,12	0

	1	2	3
distanza	0,22	0,25	0,27

Tabella 3.2: Nella prima tabella la casella (i,j) rappresenta la distanza tra il centroide i e il centroide j. Nella seconda invece troviamo la somma normalizzata delle distanze di tutti i punti di un cluster dal proprio centroide

Il secondo esperimento è stato svolto eliminando anche `gold_over_duration`, oltre a `lost_perc` in quanto correlata sia a `xp_over_duration` che alla somma di `kills` e `last_hits` (il modo più efficace di guadagnare monete è infatti uccidendo i campioni avversari e i *creeps*).
Rimuovendo due feature il numero di cluster ottimo rimane di tre, ma si si può notare che la precisione della clusterizzazione con quattro e cinque gruppi è aumentata sensibilmente mentre rimane bassa per clusterizzazioni con più di cinque gruppi.
Rispetto alla clusterizzazione precedente, come si può notare sia dalla silhouette 3.9 e dal barplot in figura 3.10, non ci sono grosse differenze e continua ad essere identificato il gruppo dei *support* con circa il 10% delle istanze.
Analoghe considerazioni possono essere fatte per la matrice di dissimilarità.

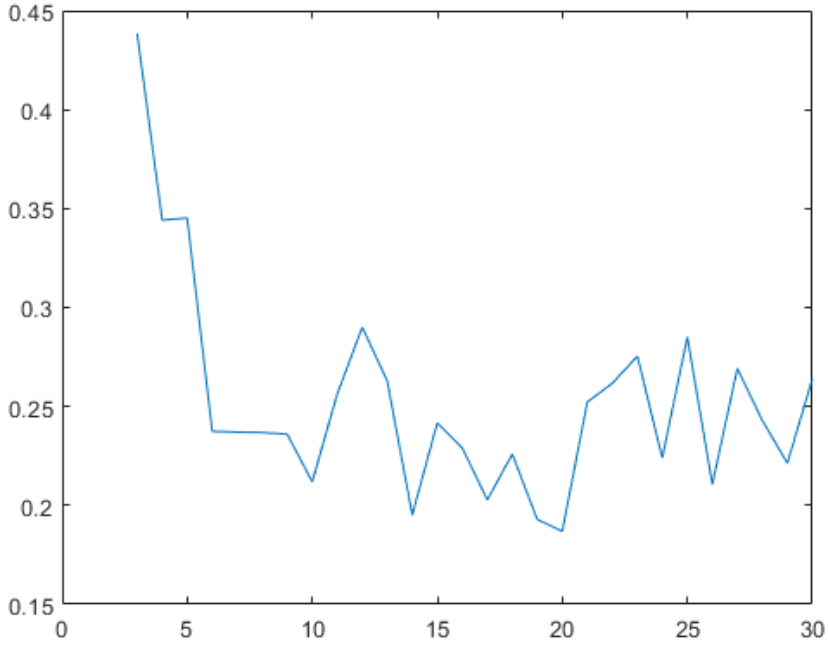

Figura 3.7: Silhouette media al variare del numero di cluster utilizzando tutte le feature tranne gold_over_duration e lost_perc

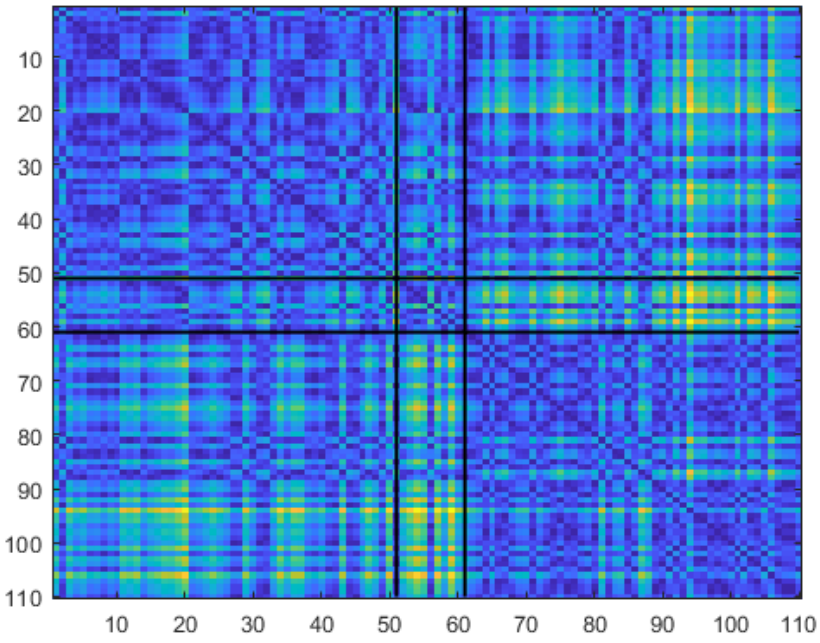

Figura 3.8: Matrice di dissimilarità del clustering ottimo utilizzando tutte le feature tranne gold_over_duration e lost_perc

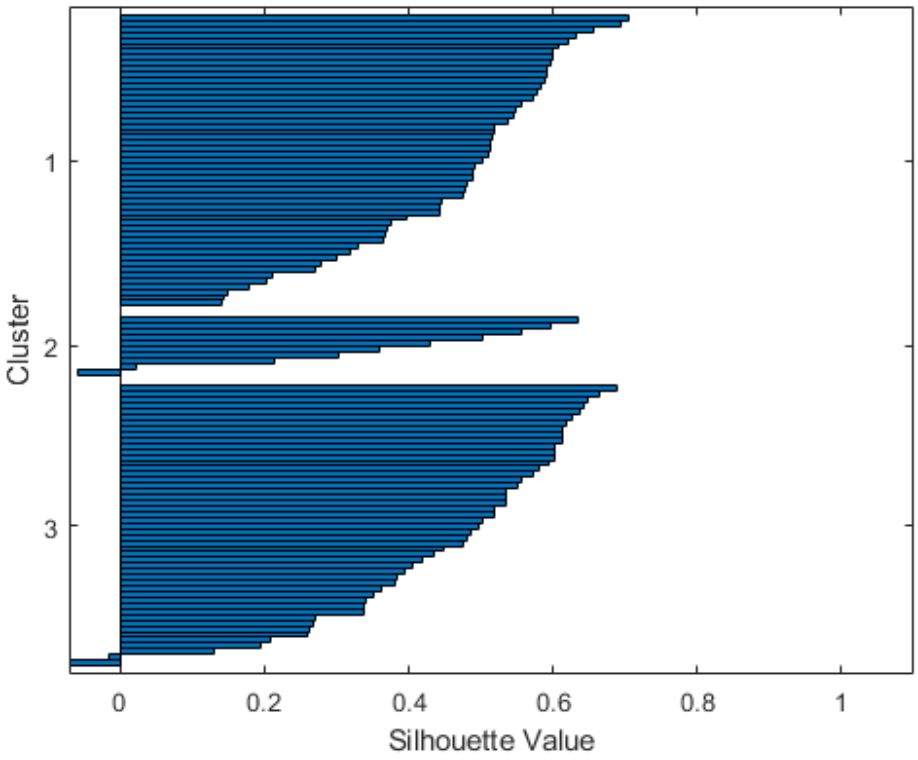

Figura 3.9: Silhouette del clustering ottimo utilizzando tutte le feature tranne gold_over_duration e lost_perc

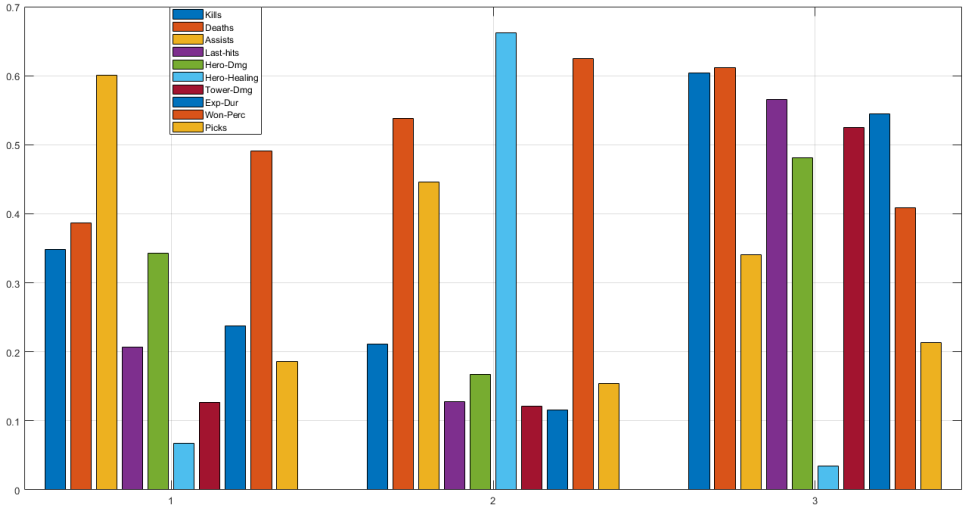

Figura 3.10: Barplot delle istanze clusterizzate utilizzando tutte le feature tranne gold_over_duration e lost_perc

3.3.3 Utilizzo delle prime sette feature

	1	2	3	4
1	0,00	0,94	0,56	0,50
2	0,94	0,00	0,68	1,07
3	0,56	0,68	0,00	0,84
4	0,50	1,07	0,84	0,00

	1	2	3	4
distanza	0,15	0,16	0,1	0,13

Tabella 3.3: Nella prima tabella la casella (i,j) rappresenta la distanza tra il centroide i e il centroide j. Nella seconda invece troviamo la somma normalizzata delle distanze di tutti i punti di un cluster dal proprio centroide

Un'altro esperimento è stato fatto utilizzando le sole prime sette feature, ovvero escludendo le feature che rappresentano i risultati delle partite o le preferenze dei giocatori. Le sette feature utilizzate sono: `kills`, `deaths`, `assists`, `last_hits`, `hero_damage`, `hero_healing`, `tower_damage`. La prima cosa che possiamo notare (figura 3.11) è che in questo caso, il numero di cluster ottimo è aumentato a quattro, anche se si continua ad avere un buon raggruppamento utilizzando tre cluster.
Analizzando i valori dei centroidi riportati in figura 3.14 possiamo notare che ancora una volta viene identificato il cluster dei *support*, mentre quello che prima era il cluster caratterizzato da statistiche più alte è stato diviso in due cluster di dimensione minore. In questi due cluster potremmo identificare due categorie di campioni, gli *split pusher* e i *carry*, i primi caratterizzati da elevati danni alle torri, e i secondi da danni ed uccisioni elevati.
Dalla matrice di dissimilarità in figura 3.12 possiamo notare che i cluster 1 e 4 in cui abbiamo identificato *carry* e *split pusher* si presentano piuttosto simili, così come i cluster 2 e 3.

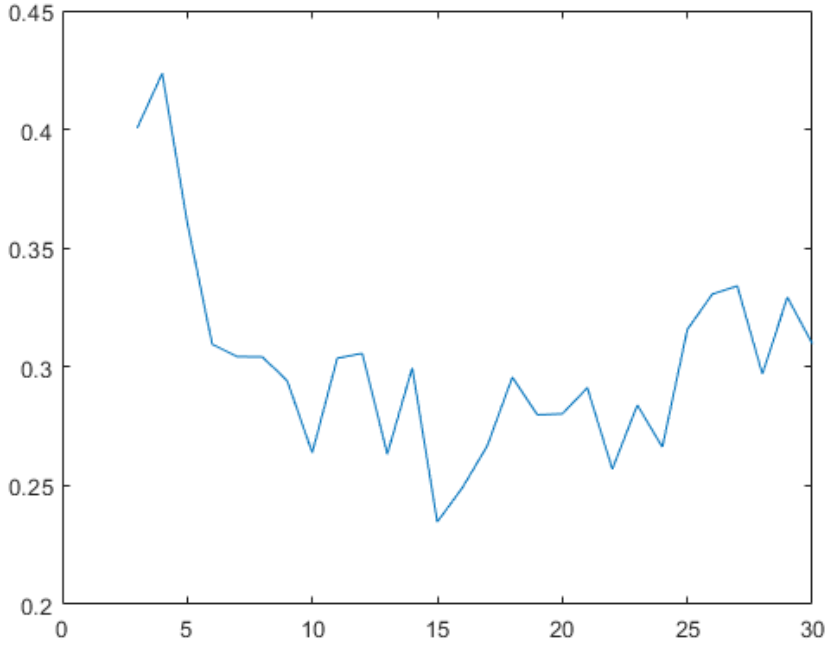

Figura 3.11: Silhouette media al variare del numero di cluster ottimo utilizzando le prime sette feature

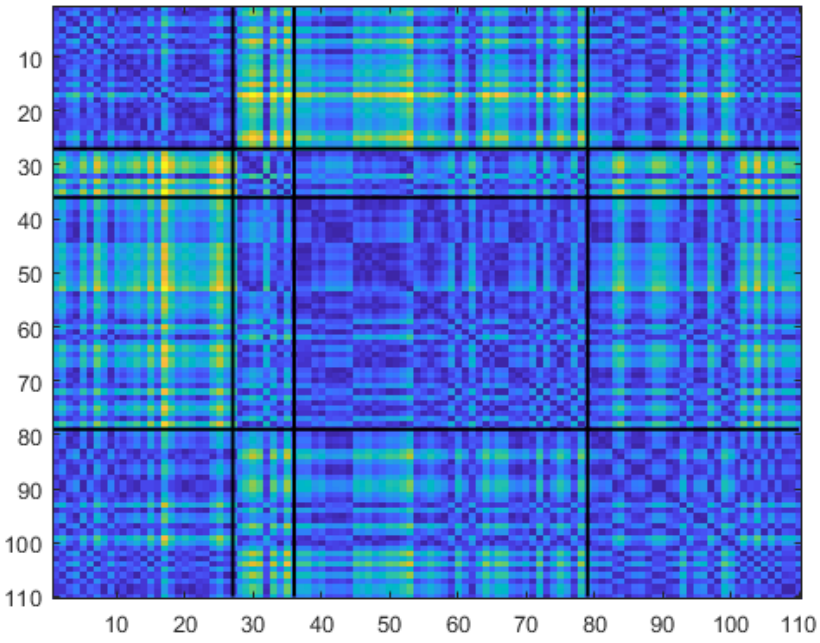

Figura 3.12: Matrice di dissimilarità del clustering ottimo utilizzando le prime sette feature

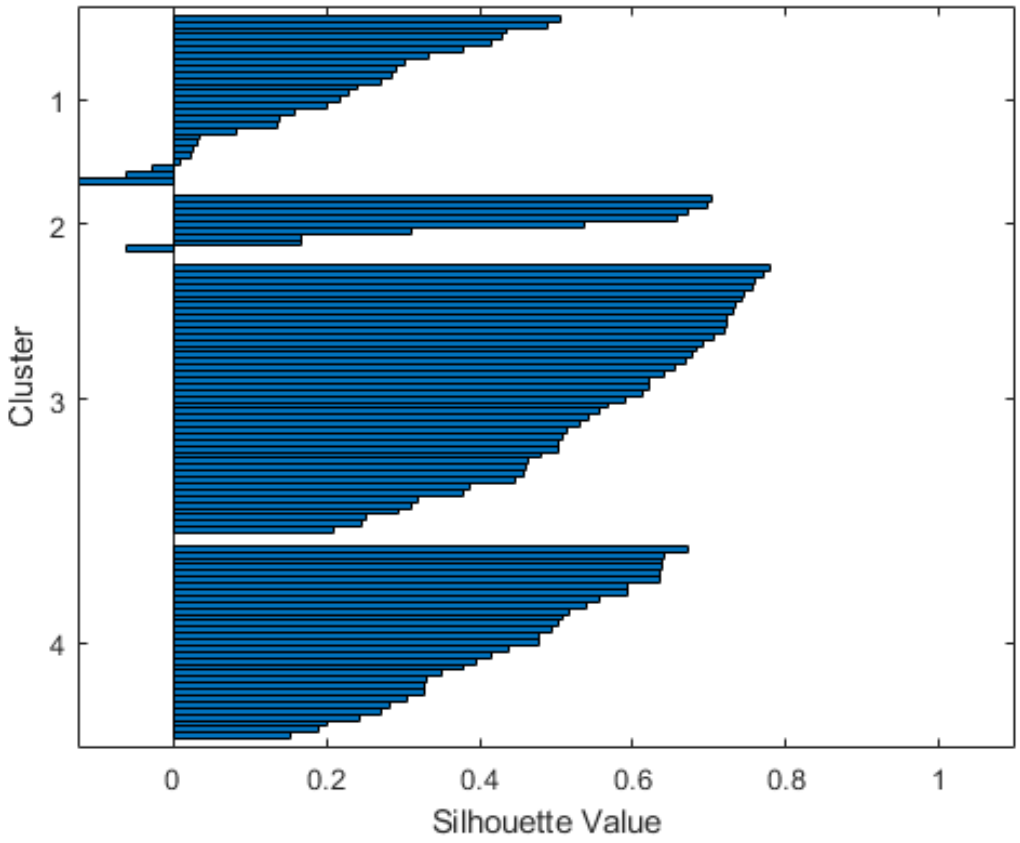

Figura 3.13: Silhouette del clustering ottimo utilizzando le prime sette feature

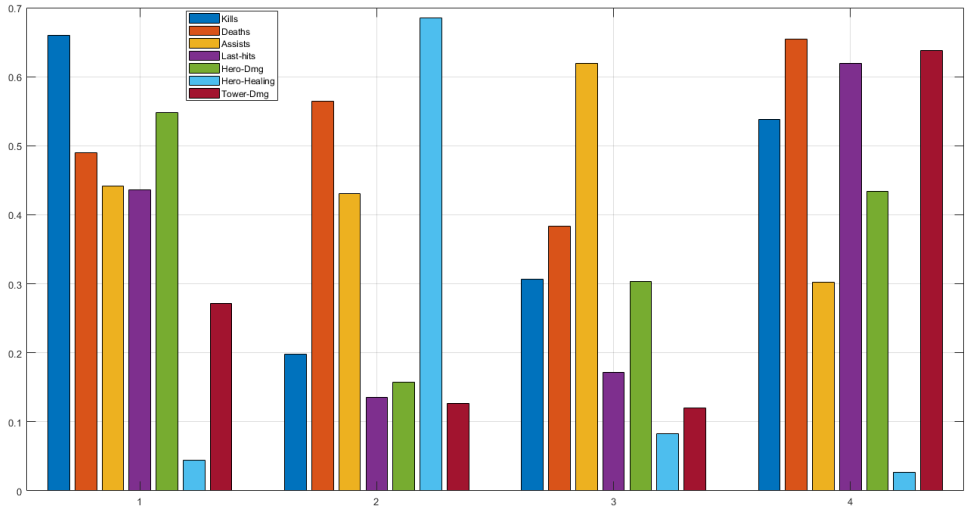

Figura 3.14: Barplot delle istanze clusterizzate utilizzando le prime sette feature

3.3.4 Utilizzo delle prime sette feature e cinque cluster

Spinti dalle osservazioni precedenti riguardo le caratteristiche dei cluster, abbiamo provato ad effettuare un ultimo esperimento utilizzando le stesse feature di quello precedente, ma partizionando in cinque gruppi in cui vogliamo provare ad identificare i ruoli principali in cui può giocare un personaggio.
Analizzando i centroidi ottenuti (figura 3.15) possiamo provare ad ipotizzare ed associare un ruolo ad ogni cluster. Come per gli esperimenti precedenti è facile identificare nel quinto cluster la categoria dei *support*, caratterizzati da cure elevate e dal resto delle statistiche relativamente basso. Analogamento il quattro e il secondo cluster possiamo associarli rispettivamente ai *carry* e agli *split pusher*. Per quanto riguarda il terzo cluster possiamo notare che è caratterizzato da un elevato numero di assist e valori particolarmente bassi per il resto delle statistiche. Questo ci porta ad associare al terzo cluster la categoria dei *tank* ovvero quei campioni che non spiccano per danni inflitti, ma caratterizzati da elevata resistenza ai danni che li rende particlarmente adatti ad essere utilizzati come bersaglio e per iniziare i combattimenti (questo porta a giustificare il numero di assist elevato). Per questo cluster ci saremmo aspettati però un maggior numero di morti. Infine abbiamo il primo cluster caratterizzato da valori nella media per tutte le feature ad eccezione delle cure. Avendo statistiche nella media è difficile associare un unico ruolo a questa categoria e abbiamo deciso di associare *fighter* (guerrieri da mischia con statistiche nella media) e *assassini* (personaggi particolarmente vulnerabili ma in grado di effettuare moltissimi danni in poco tempo).

Fatte queste considerazioni, abbiamo assegnato manualmente ad ogni campione una label con il suo ruolo effettivo e ad ogni cluster il ruolo corrispondete in modo da poter verificare la correttezza del clustering. Su 110 campioni, 72 sono stati assegnati al cluster corretto raggiungendo quindi un'accuratezza del 65%.

hero_name	heroclass
Abaddon	Support
Alchemist	Split Pusher
Ancient Apparition	Tank
Anti-Mage	Split Pusher
Axe	Fighter / Assassin
Bane	Tank
Batrider	Tank
Beastmaster	Tank
Bloodseeker	Fighter / Assassin
Bounty Hunter	Tank

Tabella 3.4: Classe associata ad ogni campione

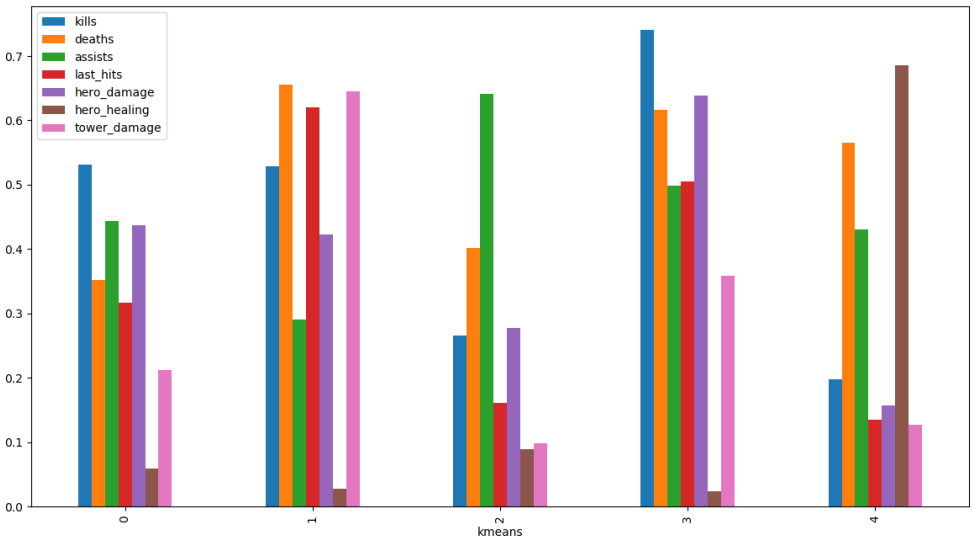

Figura 3.15: Centroidi dopo aver clusterizzato in cinque cluster

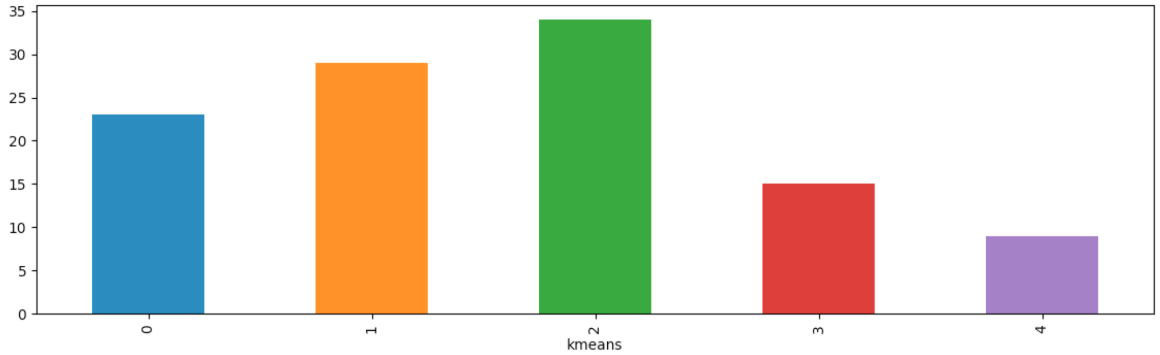

Figura 3.16: Numero di istanze in ogni cluster

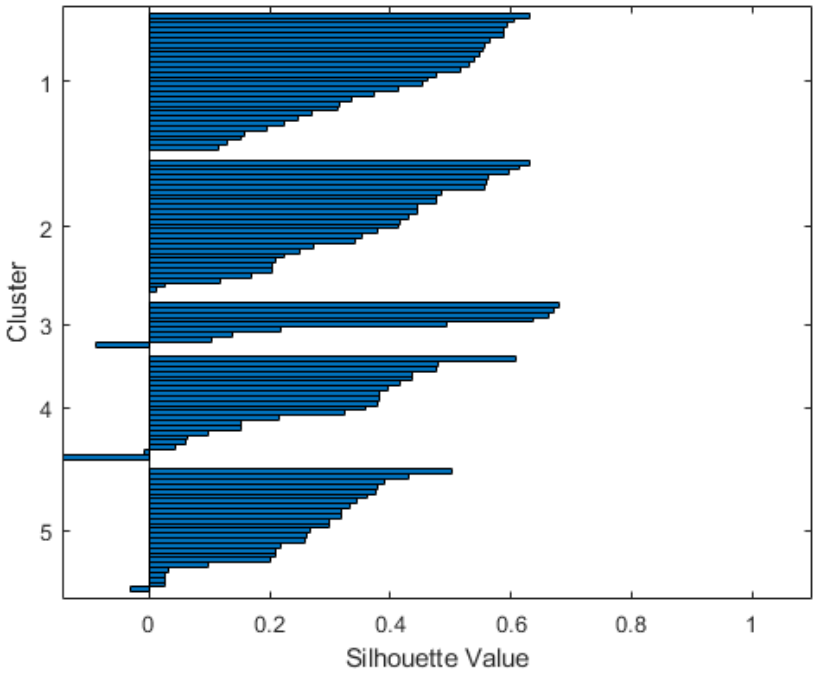

Figura 3.17: Silhouette del clustering utilizzando le prime sette feature e cinque cluster

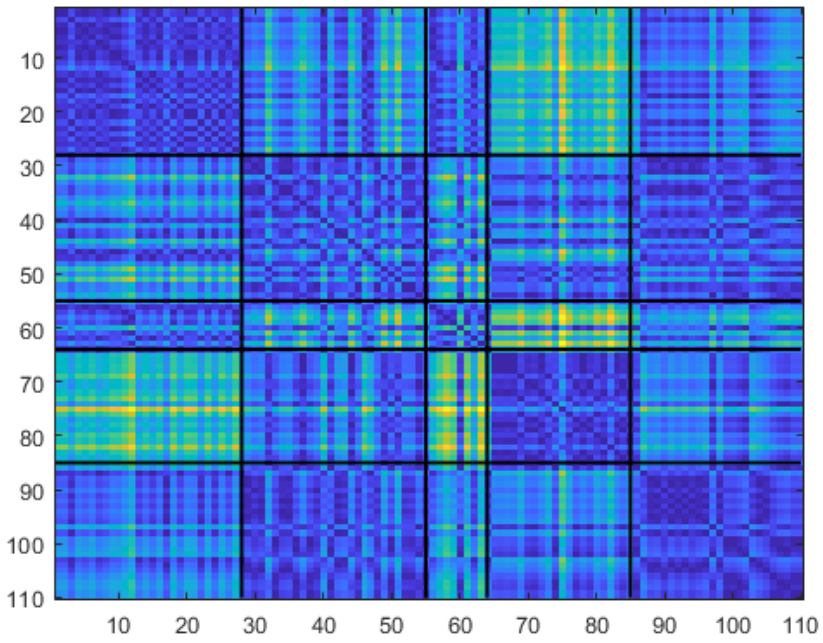

Figura 3.18: Matrice di dissimilarità del clustering utilizzando le prime sette feature e cinque cluster

Capitolo 4

Conclusioni

In questa relazione abbiamo descritto il processo di integrazione di tre dataset riguardanti i risultati di partita al videogame *DOTA 2* per poi applicare delle tecniche di clustering utilizzando l'algoritmo *K-Means*. Per quanto riguarda la parte di Data Technology abbiamo descritto il processo di integrazione dei dataset e analizzato due diverse dimensioni di qualità, sia prima che dopo il processo di integrazione, discutendo e motivando le scelte prese al fine di migliorare la qualità del dataset. Abbiamo poi effettuato delle analisi descrittive sia sui dati a disposizione prima del processo di integrazione, sia su quelli integrati e abbiamo cercato eventuali relazioni tra campioni, squadre e durate delle partite.

Per quanto riguarda la parte di Machine Learning abbiamo cercato eventuali correlazioni sulle feature a disposizione in modo da utilizzare solo quelle di maggior interesse e di ottenere una classificazione migliore. Abbiamo poi effetuato quattro esperimenti di clustering utilizzando l'algoritmo *K-Means* utilizzando diversi sottoinsiemi di feature. Inizialmente pensavamo di ottenere dei cluster che rispecchiassero la forza dei campioni, ma, come descritto nelle osservazioni dei primi tre esperimenti condotti, il raggruppamento sembrava avvenire in base al loro ruolo. Per questo motivo abbiamo deciso di condurre un quarto ed ultimo esperimento imponendo l'uso di cinque cluster per validare la nostra ipotesi. Abbiamo poi assegnato manualmente delle label ad ogni personaggio associandogli i suoi ruoli effettivi in modo da confrontarli con la clusterizzazione ottenendo una percentuale di correttezza del 65%.