Università degli studi di
Milano-Bicocca

F1801Q127

Modelli probabilistici per le decisioni

Predicting soccer results

Studenti:	*Matricole:*
Basso Matteo	807628
Ferri Marco	807130

Giugno 2019

Indice

1 Abstract

Il calcio è da anni lo sport più giocato e diffuso in svariati paesi del mondo. Moltissime persone seguono settimanalmente, con grande attenzione e fervore, ogni singola partita nella speranza di veder vincere la propria squadra del cuore.

In questo contesto si sono inevitabilmente sviluppati un gran numero di diversi business di enorme valore economico, fra cui ad esempio quello delle scommesse. Saper prevedere l'esito di una partita, sia in termini di vittoria che di goal fatti, rappresenta per moltissimi un argomento di particolare interesse. Durante questo studio, ci si è posti l'obiettivo di sviluppare un modello per la predizione del risultato delle partite di calcio in maniera automatica. Tale predizione deve essere basata sulla formazione delle due squadre considerate per ciascuna partita, nonché sulle caratteristiche dei giocatori coinvolti.

Il dominio di riferimento è stato modellato attraverso l'utilizzo di una Rete Bayesiana attraverso la quale fare le dovute inferenze e conseguentemente predire il risultato di una partita fra due squadre di calcio. Il medesimo approccio potrebbe essere riutilizzato in altri sport o giochi che prevedano l'interazione fra più squadre di cui si conoscono le statistiche di ciascun giocatore.

2 Introduzione

Viene qui presentata una visione generale del progetto, ovvero il dominio di riferimento e gli obiettivi che esso si pone, le scelte di design per la creazione del dataset ed eventuali ipotesi o assunzioni fatte durante lo sviluppo dell'elaborato.

2.1 Dominio di riferimento

Il calcio rappresenta, soprattutto negli ultimi anni, lo sport maggiormente diffuso in vari paesi del mondo. Molte persone seguono con grande attenzione tutte le partite cercando di capire preventivamente il vincitore, per piacere personale oppure per giocare nel mercato delle scommesse.

A seguito di questo fenomeno sono stati creati svariati portali che permettessero alle persone di entrare sempre di più in questo mondo. Un aspetto fondamentale per ogni piattaforma di scommesse è senz'altro la possibilità di predire il vincitore di un match attraverso la simulazione di partite di calcio con particolari formazioni.

2.2 Obiettivi dell'elaborato

Questo elaborato sarà suddiviso per argomenti, secondo un approccio di indagine incrementale e coerente con quanto praticamente svolto.

Verranno innanzitutto descritte le modalità di acquisizione dei dati dalla sorgente, a cui sarà associata anche una breve descrizione di quanto a disposizione. Quindi, si esploreranno i criteri e le assunzioni che hanno portato alla creazione del dataset su cui sviluppare il modello di predizione. Con questo, si intende dire che verranno inizialmente presentate alcune analisi qualitative sui dati e successivamente descritti i procedimenti svolti per l'integrazione e l'estrazione delle informazioni più rilevanti.

In seguito all'esportazione del dataset saranno descritte le modalità di suddivisione del dataset su cui fare inferenze, in particolare attraverso l'utilizzo delle Reti Bayesiane. Diversi modelli di generazione della rete verranno presentati e per ognuno saranno analizzati i risultati ed evidenziate le differenze. L'obiettivo del modello è quello di predire il team vincitore di un determinato match, utilizzando le informazioni dei giocatori presenti in campo.

Verrà infine presentata un'interfaccia utente per la simulazione di una partita di calcio, utile a mostrare l'effettivo impiego della rete in un prodotto di natura commerciale e pensato per l'utilizzo sul Web.

3 Descrizione del dataset originale

Di seguito viene presentata la descrizione del dataset originale, prelevato da Kaggle (3) e a sua volta creato attraverso l'unione di dati provenienti da diverse fonti, fra cui la più importante contenente le statistiche del famoso videogioco di calcio EA Sports FIFA. (2). Il contenuto del dataset è rappresentato dalle principali squadre e giocatori di calcio in Europa, con le partite svolte in un periodo temporale di 6 anni.

3.1 Formato

Il dataset consiste di un database SQLite (7), ovvero un file con omonima estensione contenente un database relazionale di rapido utilizzo, in grado di essere condiviso e utilizzato facilmente su un qualsiasi dispositivo.

3.2 Giocatori

I giocatori sono espressi all'interno di due tabelle differenti nello schema del database considerato. La prima relazione denominata *Player* contiene informazioni di carattere generale ed è mostrata in tabella 1 a seguire.

Tabella 1: Player

Campo	Descrizione
id	Id del record
player_api_id	Identificativo univoco del giocatore
player_name	Nome completo del giocatore
player_fifa_api_id	Id utile per reperire le informazioni da fifa
birthday	Data di nascita
height	Altezza in centimetri
weight	Peso in libbre

La seconda tabella, denominata *PlayerAttributes* contiene invece i punteggi associati alle abilità personali di ciascun giocatore. Essa contiene inoltre un campo che indica a quale data tali punteggi si riferiscono, cioè quando sono state effettuate le rilevazioni. É possibile, infatti, che i dati associati ad un giocatore cambino nel tempo; per tale motivo, nel momento in cui si rivelerà necessario recuperare i dati dei giocatori facenti partecipanti ad una partita, sarà necessario selezionare i giusti record dalla tabella *Player* proprio in base alla data della partita.

Nella tabella 2 vengono meglio elencati i campi che descrivono un giocatore. La maggior parte di essi sono espressi su una scala intera da 1 a 100.

Tabella 2: PlayerAttributes

Campo	Descrizione
id	Id del record
player_api_id	Identificativo univoco del giocatore
player_fifa_api_id	Id utile per reperire le informazioni da FIFA
date	Data di riferimento del record
overall_rating	Valutazione generale
potential	Potenziale
preferred_foot	Piede preferito
attacking_work_rate	Punteggio di attacco
defensive_work_rate	Punteggio di difesa
crossing	Cross
finishing	Capacità di finalizzare (segnare un goal)
heading_accuracy	Tiro di testa
short_passing	Passaggio corto
volleys	Tiro al volo
dribbling	Dribbling
curve	Tiro con effetto curvilineo
free_kick_accuracy	Accuratezza dei calci piazzati
long_passing	Passaggio lungo
ball_control	Controllo di palla
acceleration	Accelerazione
sprint_speed	Velocità in scatto
agility	Agilità
reactions	Reattività
balance	Equilibrio
shot_power	Potenza di tiro
jumping	Salto
stamina	Resistenza
strength	Forza
long_shots	Tiri lunghi
aggression	Aggressività
interceptions	Intercettazione
positioning	Posizionamento
vision	Vista
penalties	Falli commessi
marking	Capacità di marcare
standing_tackle	Contrasto in piedi
sliding_tackle	Scivolata
gk_diving	Tuffo (portiere)
gk_handling	Presa (portiere)
gk_kicking	Rinvio (portiere)
gk_positioning	Posizionamento (portiere)
gk_reflexes	Riflessi (portiere)

3.3 Squadre

Le squadre sono descritte analogamente ai giocatori tramite 2 tabelle, *Team* 4 e *TeamAttributes* 3 riportate qui di seguito.

Tabella 3: TeamAttributes

Campo	Descrizione
id	Id del record
team_api_id	Identificativo univoco della sqaudra
team_fifa_api_id	Id utile per reperire le informazioni da FIFA
team_name	Nome completo della squadra
date	Data di riferimento del record
buildUpPlaySpeed	Velocità di gioco
buildUpPlaySpeedClass	Discretizzazione velocità di gioco
buildUpPlayDribbling	Dribbling
buildUpPlayDribblingClass	Discretizzazione dribbling
buildUpPlayPassing	Passaggio
buildUpPlayPassingClass	Discretizzazione passaggio
buildUpPlayPositioningClass	Posizionamento della squadra, definito o libero
chanceCreationPassing	Creazione chance di passaggio
chanceCreationPassingClass	Discretizzazione creazione chance di passaggio
chanceCreationCrossing	Creazione chance di cross
chanceCreationCrossingClass	Discretizzazione creazione chance di cross
chanceCreationShooting	Creazione chance di tiro in porta
chanceCreationShootingClass	Discretizzazione creazione chance di tiro in porta
chanceCreationPositioningClass	Creazione chance di posizionamento
defencePressure	Pressing della difesa
defencePressureClass	Discretizzazione pressing della difesa
defenceAggression	Aggressività della difesa
defenceAggressionClass	Discretizzazione aggressività della difesa
defenceTeamWidth	Copertura della difesa
defenceTeamWidthClass	Discretizzazione copertura della difesa
defenceDefenderLineClass	Tipologia di difesa

Come osservabile, a differenza di quanto avvenga con i giocatori, per le squadre molti attributi appaiono già discretizzati attraverso una suddivisione in classi che descrive a parole la relativa scala da 1 a 100 di ciascun campo.

Tabella 4: Team

Campo	Descrizione
id	Id del record
team_api_id	Identificativo univoco della sqaudra
team_name	Nome completo della squadra
team_fifa_api_id	Id utile per reperire le informazioni da fifa
team_short_name	Nome abbreviato della squadra
team_long_name	Nome completo della squadra

3.4 Partite

La tabella *Match* contiene tutte le informazioni relative alle squadre che hanno giocato una partita l'una contro l'altra, i giocatori coinvolti e il loro posizionamento in campo. I giocatori vengono numerati da 1 a 11 e il portiere viene sempre collocato in posizione [1,1]. Sono presenti inoltre dei valori il cui scopo è quello di indicare alcune specifiche tecniche sull'andamento della partita. Per brevità e mancanza di documentazione verranno semplicemente omessi, poiché poco rilevanti ai fini dello studio. Nella tabella 5 è presentata la descrizione dei vari campi.

Tabella 5: Match

Campo	Descrizione
id	Id del record
country_id	Id della nazione di riferimento
league_id	Id della lega di riferimento
season	Stagione calcistica
date	Data della partita
match_api_id	Id univoco della partita
home_team_api_id	Id del team in casa
away_team_api_id	Id del team in trasferta
home_team_goal	Goal del team in casa
away_team_goal	Goal del team in trasferta
home_player_i	Id dell'i-esimo giocatore in casa
away_player_i	Id dell'i-esimo giocatore in traferta
home_player_Xi	Posizione i-esimo gioc. (casa) sul lato corto del campo
home_player_Yi	Posizione i-esimo gioc. (casa) sul lato lungo del campo
away_player_Xi	Posizione i-esimo gioc. (trasferta) sul lato corto del campo
away_player_Yi	Posizione i-esimo gioc. (trasferta) sul lato lungo del campo

3.5 Qualche numero

Di seguito sono riportati i numeri costituenti il dataset d'origine:

- 25979 partite svoltesi fra il 2010 e il 2016;
- 11060 giocatori in totale;
- 183978 rilevazioni sulle abilità dei giocatori
  ciò significa approssimativamente 16 rilevazioni per ciascun giocatore;
- 299 squadre di 11 diversi campionati Europei
- 1458 rilevazioni sulle caratteristiche delle squadre
  ciò significa approssimativamente 5 per ciascun squadra.

4 Creazione del dataset per le predizioni

Affinché i dati a disposizione possano essere utilizzati per la predizione del risultato di una partita su base statistica, è necessario che essi vengano prima elaborati secondo un insieme di regole su cui costruire i principi per portare a termine una specifica predizione. L'elaborazione stessa è perciò determinata sia attraverso delle assunzioni volte a identificare gli elementi più rilevanti nei dati, sia tramite l'effettiva esecuzione delle operazioni necessarie a trasformare il dataset originale in quello utilizzato per modellare la Rete Bayesiana.

4.1 Ipotesi e assunzioni

Al fine di sviluppare il progetto e procedere con la stesura del seguente elaborato, sono state fatte delle assunzioni che fungono da base portante.

Si assume che il risultato di una partita, a sua volta determinato dal numero di goal segnati da una squadra e dall'altra, sia influenzato dalla bravura dei giocatori che partecipano alla partita stessa. Nel gioco del calcio, come in molti altri, esiste una netta suddivisione fra i compiti assegnati a ciascun giocatore che vanno a determinare il ruolo di quest'ultimo: **portiere**, **difensore**, **centrocampista** e **attaccante**. É chiaro che ogni ruolo riveste la sua particolare importanza e influisce in maniera diversa sull'esito della partita. Per questo progetto, si è pensato di costruire un modello basato proprio sulle potenzialità delle due squadre sfidanti nei diversi ruoli assunti dai giocatori. Poiché tale informazione è mancante all'interno del dataset originale, il primo step cui si è fatto fronte è stata la determinazione dei ruoli di ciascun giocatore in campo. Tale obiettivo è stato raggiunto attraverso l'ipotesi, più che ragionevole, secondo la quale la posizione di una persona sul campo da gioco determina anche il suo ruolo all'interno della squadra e della partita.

In secondo luogo, si è scelto inoltre di supporre che le caratteristiche del team siano determinate solo dai giocatori che ne fanno parte. Non avendo a disposizione informazioni circa l'appartenenza di un giocatore ad un determinato team nel tempo, non è possibile ottenere una suddivisione dei giocatori in squadre ben formate. Nonostante quest'ultime si potrebbero ricavare direttamente dalle partite svolte, ciò comporterebbe la creazione di dati incoerenti col mondo reale poiché i giocatori di calcio sono soggetti ad un mercato di compra-vendita su base annuale.
Per questo motivo, ai fini della predizione si è scelto di fatto di ignorare l'identità delle due squadre coinvolte in un match, ma semplicemente **limitarsi a considerare i 22 giocatori che hanno preso parte alla partita**, pesando adeguatamente le abilità che li contraddistinguono nel proprio ruolo.

4.2 Scelta del formato CSV

Per la creazione del dataset finale, utile per la modellazione della Rete Bayesiana, è stato scelto di trasformare il dataset in formato CSV, in quanto più semplice e pratico da utilizzare rispetto al database relazionale inizialmente fornito. Come si accennava, affinché possa essere creato il modello e su questo condotto delle previsioni, è necessario che i dati vengano elaborati e aggregati in un'unica tabella. Mentre SQLite consente agevolmente di lavorare sulle tabelle per effettuare *join*, selezioni e proiezioni sul modello relazionale, il DBMS non è particolarmente comodo per conservare i dati una volta terminata l'aggregazione. A tale scopo si è preferito quindi esportare successivamente il risultato ottenuto all'interno di un file CSV, decisamente più portabile e immediatamente leggibile rispetto ad un database.

4.3 Elaborazione dei dati

Avendo come principale fonte d'informazione i dati relativi alle partite calcistiche svoltesi in Europa fra il 2010 e il 2016, e sapendo per ciascuna di essa i giocatori partecipanti (con relative posizioni in campo) e l'esito del match, è chiaro che la priorità durante la fase di elaborazione dei dati è stata quella di unire tutte le informazioni a disposizione in un'unica tabella. Ciò è stato possibile attraverso il *join* delle tabelle *Match* e *PlayerAttributes* utilizzando le chiavi che le mettono in relazione. Successivamente, i dati sono stati aggregati per computare la competitività di una certa squadra in una specifica partita.

4.3.1 Reperimento delle informazioni sui giocatori di una partita

Per ogni partita si hanno a disposizione 22 chiavi esterne, relative ai giocatori della squadra di casa e di quella ospite; con queste, è stato possibile reperire per ciascun giocatore le proprie abilità in quel determinato momento temporale. Tale informazione è reperibile nella tabella *PlayerAttributes*, nella quale per la medesima persona sono contenute più rilevazioni, con date differenti. Chiaramente, la rilevazione corretta delle abilità di un certo giocatore è rappresentata da quella più recente, purché questa sia stata effettuata precedentemente alla data del match in esame.

Per fare un esempio pratico, se in una partita del 3/07/2016 ha partecipato il giocatore con `id 1234` e supponendo che in *PlayerAttributes* si trovino, per tale giocatore, i dati che è possibile osservare nella tabella 6, è chiaro che il dato corretto da associare a tale partita per il giocatore `1234` è quello datato 12/12/2015. Questo rappresenta l'informazione più aggiornata alla data della partita.

Tabella 6: PlayerAttributes WHERE Id = 1234

Id	Name	Date	Overall	Finishing	Marking	...
1234	Gabriel Marquez	12/04/2018	78	72	85	...
1234	Gabriel Marquez	12/12/2015	80	75	82	...
1234	Gabriel Marquez	05/10/2013	75	70	80	...

L'operazione di *join* da eseguire per ciascuna partita ha dovuto coinvolgere ben 22 giocatori, di cui per ognuno è stato necessario scegliere la riga corretta di *PlayerAttributes*, adeguatamente filtrata per data. É da ricordare, come accennato nel capitolo 3.5, che per ciascun giocatore si registrano mediamente 16 rilevazioni in date diverse all'interno della tabella *PlayerAttributes*.

Il primo tentativo di effettuare il join è quindi fallito.
Tentando di reperire tutte le informazioni necessarie attraverso l'esecuzione di un'unica query SQL ci si è accorti che, in questo specifico caso d'uso, 22 join fossero decisamente troppi; questo poiché per ciascuna colonna di *Match* da *joinare* (ognuna delle quali rappresenta la chiave esterna di un giocatore) era necessario computare il prodotto cartesiano fra ciascun match e mediamente 16 record di *PlayerAttributes*. La query, il cui codice è semplificato qui sotto, produceva quindi una crescita di righe esponenziale fra un *join* e il successivo, che rappresenta tutt'oggi un requisito computazionalmente impossibile da gestire per tabelle di grosse dimensioni.

```
SELECT *
FROM Match as m
JOIN Player_Attributes as h1
on h1.player_api_id = match.home_player_1 AND h1.date < m.date
JOIN Player_Attributes as h2
on h2.player_api_id = match.home_player_2 AND h2.date < m.date
...
GROUP BY m.match_api_id
HAVING h1.date = MAX(h1.date)
HAVING h2.date = MAX(h2.date)
...
```

Per ovviare il problema, si è quindi pensato di svolgere un'operazione di *join* alla volta, salvando di volta in volta il risultato in tabelle intermedie per i 22 giocatori. Ciò è stato possibile attraverso lo sviluppo di un programma Javascript per la creazione incrementale delle query e l'interfacciamento diretto con il database SQLite. L'approccio si è rivelato vincente, facendo abbassare drasticamente i tempi di esecuzione da diverse ore a poche decine di secondi.

4.3.2 Ruolo di un giocatore

La tabella risultante dallo step precedente, contenente per ogni partita i dati di ciascun giocatore coinvolto, ne è risultata di circa 17mila record dai 25mila iniziali, a causa di alcuni dati mancanti proprio relativamente ai giocatori partecipanti. A questo punto si è quindi scelto di elaborare proprio le caratteristiche dei giocatori per costruire il dataset da utilizzare per le predizioni.

Come accennato nella sezione 4.1, si è assunto che la potenzialità di vittoria di una certa squadra sia da ricondurre alle caratteristiche della squadra stessa nei diversi ruoli assunti dai giocatori a seconda della loro posizione sul campo da gioco. Tale problema è stato affrontato utilizzando le 44 colonne della tabella *Match* rappresentanti le coordinate `X` e `Y` di ciascuno dei 22 giocatori in campo.

In particolare, considerando il campo da calcio come un piano e rivolgendosi con le spalle alla porta, si è potuto evincere che le `X` rappresentassero la posizione di un giocatore rispetto al lato corto del campo (quindi a destra o sinistra rispetto al portiere), mentre le `Y` fossero rappresentative della distanza fra il portiere e il giocatore considerato, quindi la posizione di quest'ultimo rispetto al lato lungo del campo. Proprio questa coordinata, sopratutto grazie ad una conoscenza pregressa del dominio di riferimento, si è rivelata essenziale per determinare i ruoli di ciascun giocatore durante il corso di una partita.

Attraverso una scelta empirica e il confronto fra distribuzioni del numero di giocatori nei diversi ruoli, sono state fatte le seguenti assunzioni, che si possono ritrovare espresse graficamente anche nella figura 2:

- La `Y` pari a 1 è sempre occupata dal portiere;
- Le `Y` da 2 a 4 costituiscono i difensori;
- Le `Y` da 5 a 8 sono costituenti dei centrocampisti;
- Le `Y` da 9 a 11 rappresentano gli attaccanti.

Tale assunzione è totalmente plausibile considerata la tipica formazione di una squadra da calcio ed ha portato all'assegnamento di uno dei 4 ruoli disponibili a ciascun giocatore in partita. Nella figura 1 è possibile quindi osservare la distribuzione assunta dal numero di giocatori per ciascun ruolo nelle varie partite presenti nel dataset. Mentre i difensori sono quasi sempre quattro, gli attaccanti sono fra uno e tre per partita mentre i centrocampisti possono essere da tre a cinque.

Le analisi e le elaborazioni sui dati presentate nel seguito di questo capitolo sono state ottenute tutte attraverso l'utilizzo del linguaggio di programmazione R.

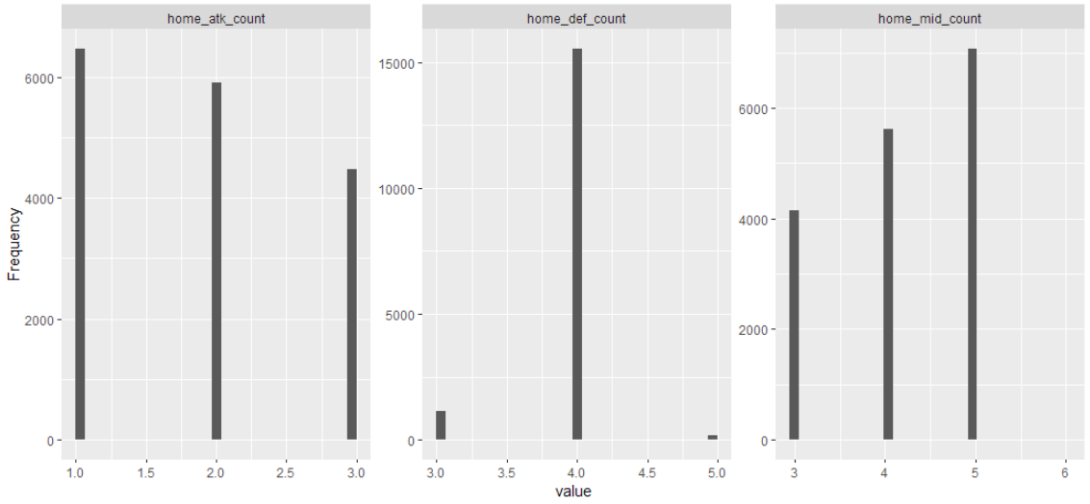


Figura 1: Distribuzione dei ruoli nelle partite

Figura 2: Suddivisione del campo da calcio in ruoli

4.3.3 Potenzialità di una squadra in partita

Se in *PlayerAttributes* (le cui colonne sono meglio descritte nella tabella 2) vengono espresse quasi 40 diverse abilità tipiche dei giocatori di calcio, non tutte sono essenziali per ciascun ruolo. Si trovano ad esempio `finishing` e `shot_power` come caratteristiche degli attaccanti oppure `marking` e `standing_tackle` per i difensori, e ancora altri attributi espressamente dedicati ai portieri.

In questo contesto, si è pensato di computare per ciascuna partita e squadra partecipante diversi punteggi per esprimere la potenza di una certa squadra, sulla base dei 4 ruoli disponibili nel calcio: **portiere**, **difesa**, **centrocampo** e **attacco**. Tali punteggi costituiranno in seguito i nodi della Rete Bayesiana su cui fare le dovute predizioni; si è deciso di calcolarli sfruttando i valori registrati dai giocatori che costituiscono i 4 ruoli della squadra durante una partita.

Inizialmente, si era pensato di assegnare a ciascun giocatore della partita un determinato punteggio calcolato attraverso una media pesata delle abilità più rilevanti per il ruolo che si sta giocando. Questo avrebbe comportato la necessità di aggregare per ciascun giocatore diverse abilità a seconda del suo ruolo in partita, per successivamente andare a costituire un'ulteriore aggregazione di tutti i giocatori in una certa posizione al fine di computare le potenzialità della squadra, per ciascun ruolo, durante la partita.

Seppur il calcolo sia logicamente sensato, conoscendo il dominio si è tuttavia ragionato sul fatto che ciascun giocatore del calcio moderno è sempre intrinsecamente legato ad un ruolo, che assume in tutte le partite da egli giocate. Seguendo questo ragionamento si è conseguentemente provato a verificare l'attendibilità del punteggio `overall_rating`, già espresso all'interno del dataset per ciascun giocatore, che dovrebbe rappresentare il punteggio generale del giocatore con particolare attenzione al proprio ruolo.

É stata pertanto condotta un'analisi di correlazione fra `overall_rating` e gli altri attributi di *PlayerAttribute* per dimostrare che il primo fosse in grado di adattarsi efficacemente al tipico ruolo del giocatore considerato. Tale analisi, presentata nella sezione successiva, ha chiaramente evidenziato che l'attributo `overall_rating` della tabella *PlayerAttribute* è sufficiente a descrivere efficaciemente un giocatore, pertanto si è scelto di utilizzare questa colonna come unica caratteristica per costituire i punteggi aggregati di difesa, centrocampo e attacco.

4.3.4 Correlazione fra ruolo e `overall_rating`

Per analizzare la correlazione fra ruolo di un giocatore e `overall_rating`, al fine di comprendere se quest'ultimo sia adatto a descrivere efficamente un giocatore, si è innanzitutto tentato di suddividere i giocatori in ruoli, sulla base dei valori assunti in determinate abilità e studiando la distribuzione dei valori assunti da quest'ultime.

Facendo riferimento a `gk_diving` e la figura 3, si è facilmente evidenziata una distribuzione nettamente suddivisa fra valori sopra 50 (su una scala di 100) e sotto 40, a scindere inequivocabilmente i giocatori portieri da tutti gli altri. Per questi, è chiaramente evidente in figura 4 che `overall_rating` sia totalmente dipendente dalle abilità legate al portiere e dalla capacità di reagire rapidamente.

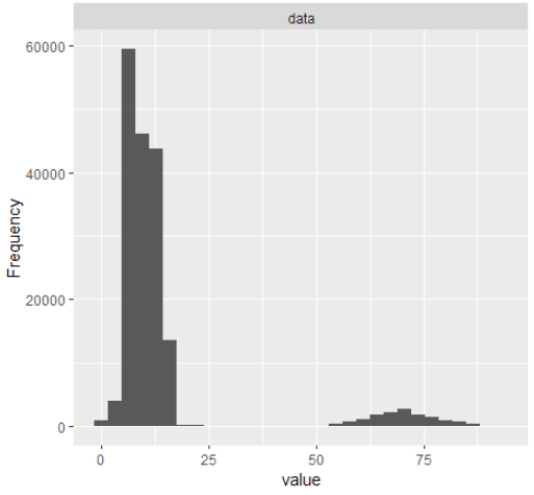


Figura 3: Distribuzione di `gk_diving`

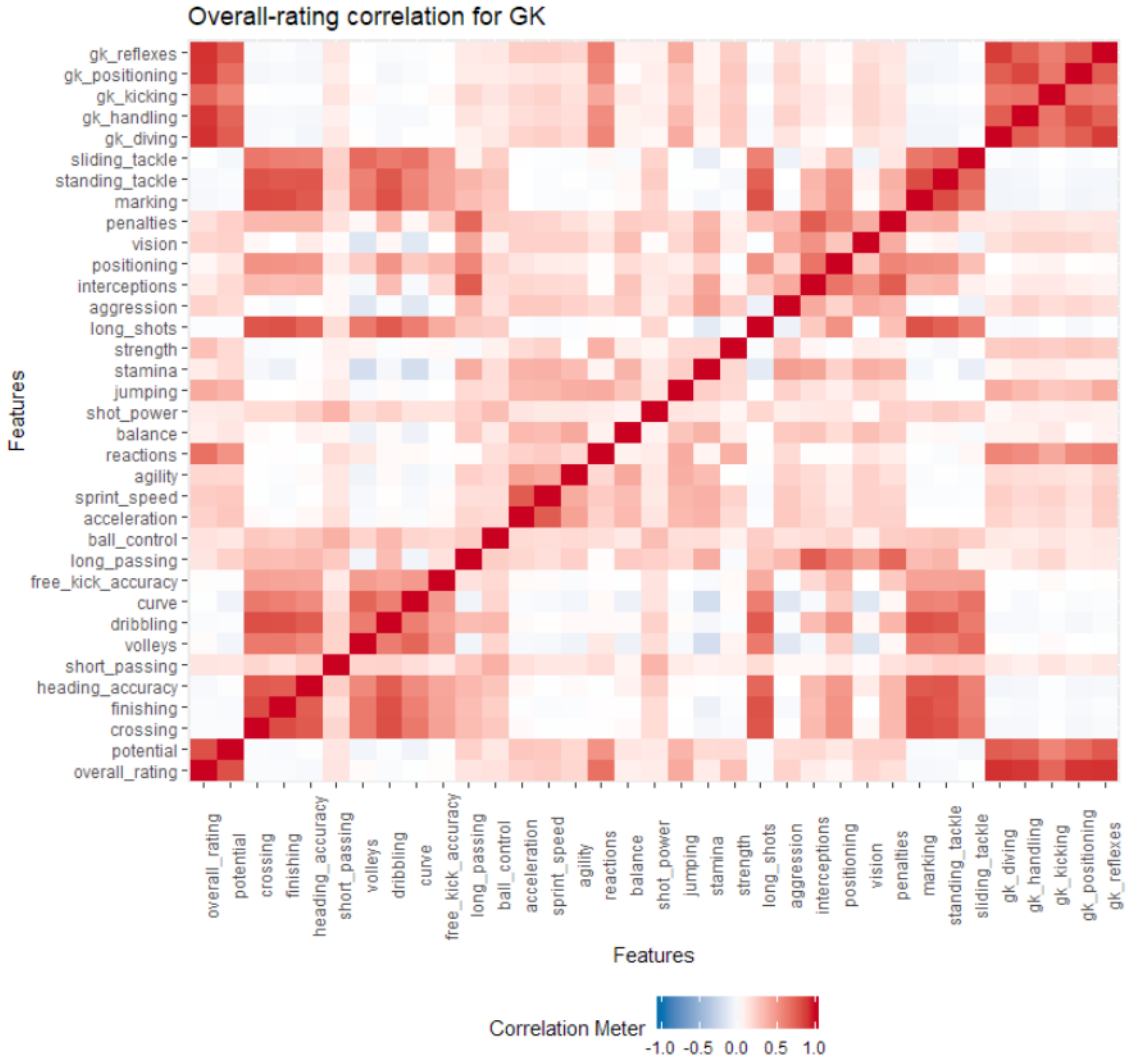


Figura 4: Correlazione fra `overall_rating` (prima riga dal basso) e le abilità di un portiere (a destra)

Studiando invece la correlazione fra le diverse abilità, si è evinto che due fra gli attributi meno correlati fra loro fossero `marking` e `finishing`, associati rispettivamente al ruolo difensivo e quello d'attacco. Sfruttando l'incorrelazione di -0.62 qui mostrata, si è quindi potuto sfruttare i due campi per scindere il dataset dei giocatori in attaccanti e difensori, per studiarne il comportamento di `overall_rating`.

```
cor(PlayerAttributes[PlayerAttributes$gk_diving <= 40, ]$marking,
PlayerAttributes[PlayerAttributes$gk_diving <= 40, ]$finishing,
use ="pairwise.complete.obs");  RESULT = -0.6204823
```

In figura 5 è possibile osservare la matrice di correlazione applicata sui difensori, cioè i giocatori non-portieri con un punteggio minimo di 50 su `marking` e massimo di 50 su `finishing`. Si evince come la maggior parte della correlazione di `overall_rating` (prima riga dal basso) la si ottenga con le abilità tipiche della difesa come `marking`, `sliding_tackle` e `interceptions`. Le abilità del portiere sono totalmente ininfluenti ed in generale anche le caratteristiche d'attacco non sono molto rilevanti.

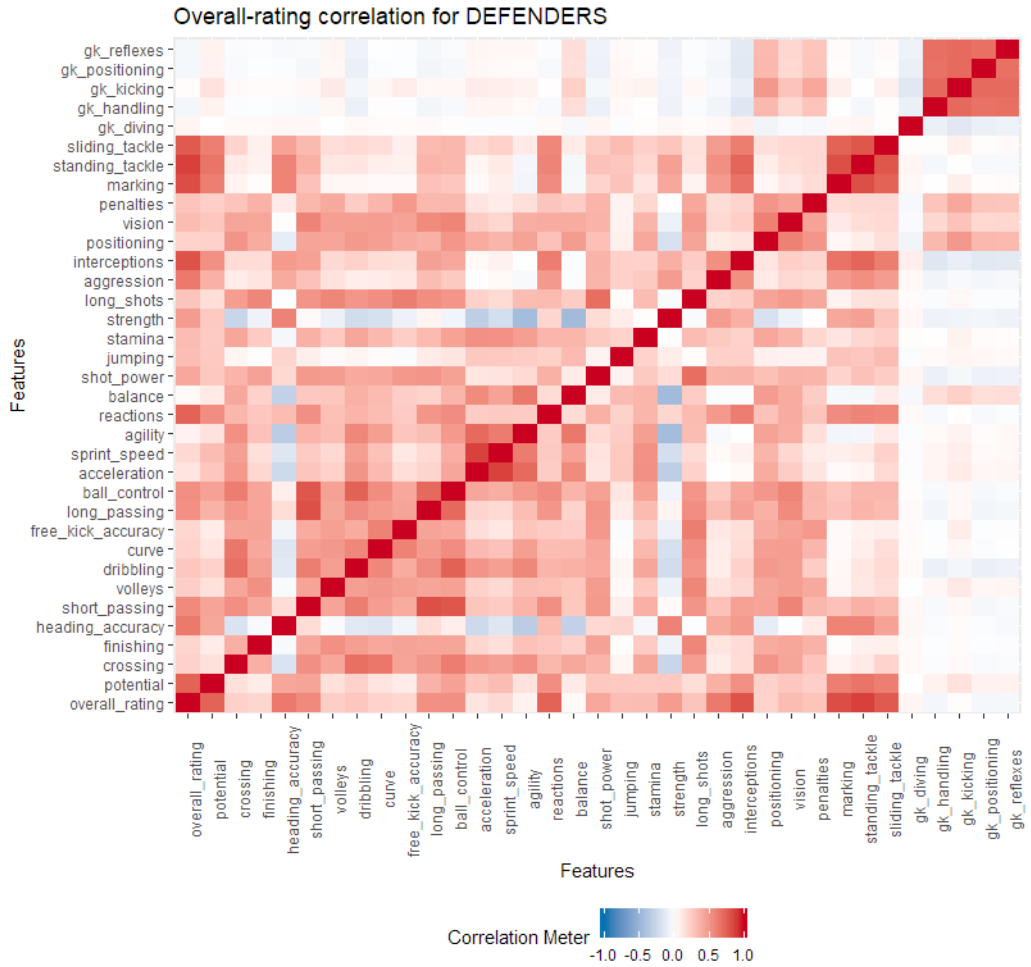


Figura 5: Correlazione fra `overall_rating` (prima riga dal basso) e le abilità di un difensore

Infine, nella figura 6 viene presentata la correlazione fra `overall_rating` e le abilità di un attaccante, cioè un giocatore non-portiere con un punteggio massimo di 50 su `marking` e minimo di 50 su `finishing`. Questa volta si nota che le abilità difensive

non hanno alcuna influenza sul punteggio globale, a differenza di quanto avvenga invece per le abilità più legate ai tiri in porta, al dribbling e alla velocità di corsa, che erano decisamente meno influenti per i difensori.

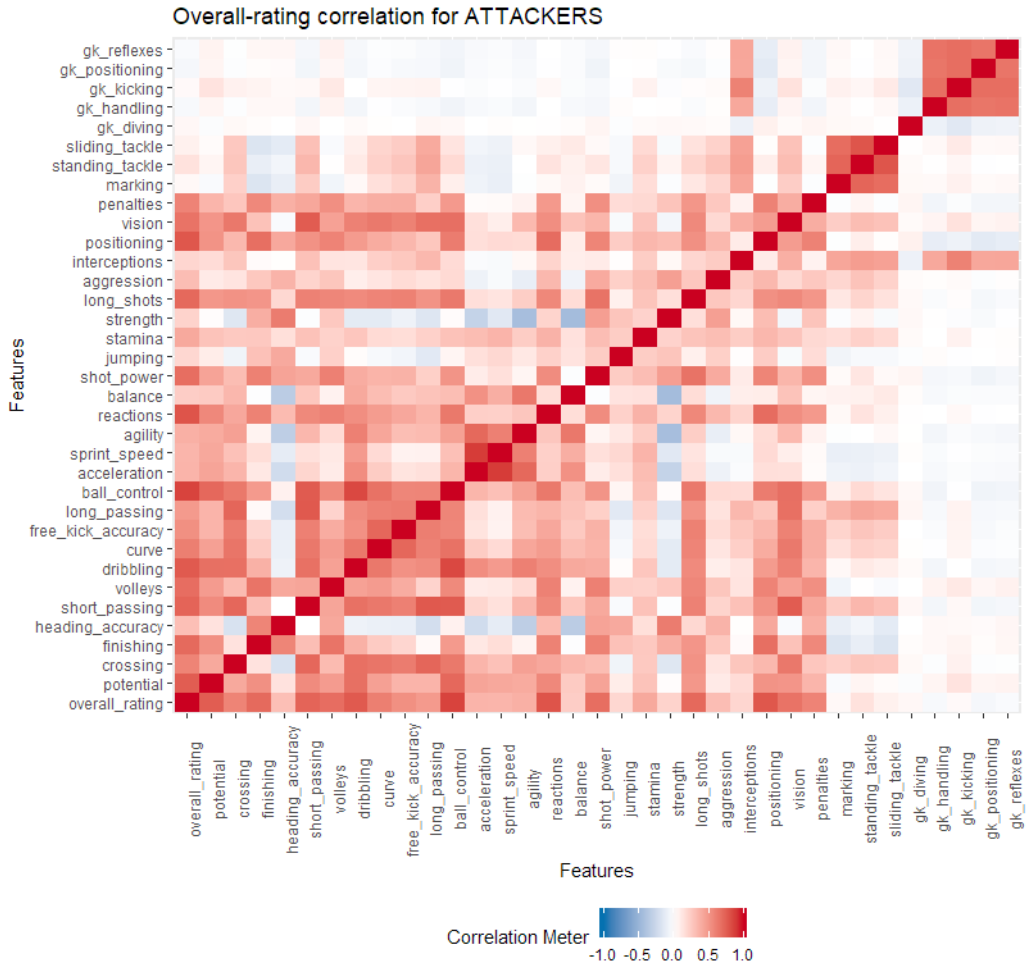


Figura 6: Correlazione fra `overall_rating` (prima riga dal basso) e le abilità di un attaccante

4.3.5 Calcolo e discretizzazione dei punteggi generali per la squadra

Tramite le precedenti considerazioni si è dimostrato che è possibile utilizzare il solo attributo `overall_rating` per descrivere efficacemente un giocatore. Per calcolare quindi i punteggi ottenuti da una certa squadra, composta dagli 11 giocatori, in una certa partita si è quindi pensato di mediare fra loro gli `overall_rating` dei giocaori facenti parte dei 4 diversi ruoli del calcio.

Così facendo si sono ottenuti 4 valori continui, da 1 a 100, rappresentanti le caratteristiche generali della squadra sulla base dei giocatori coinvolti nella partita. Successivamente verrà dimostrato che, per il caso in esame, considerare anche il numero di giocatori in un determinato ruolo è ininfluente.

Al fine di sottoporre queste informazioni ad una Rete Bayesiana, si è quindi proceduto con una discretizzazione dei valori ottenuti, che erano compresi fra 1 e 100. Inizialmente, si è scelto di utilizzare 4 intervalli di discretizzazione; più avanti in questo elaborato (nella sezione 6.1.4) si mostrerà come questa scelta possa avere un'influenza sulle performance del modello di predizione.

I dati ottenuti sono stati ripuliti delle informazioni non più rilevanti, come i dati sui singoli giocatori, e quindi esportati in formato CSV per essere sottoposti alla Rete Bayesiana con lo scopo di fare previsioni sul risultato di una partita e inferenze di diverso genere sui possibili valori assunti dai 4 punteggi per ciascuna squadra.

5 Rete Bayesiana

Lo scopo di questo progetto è la predizione del risultato di una partita di calcio. Sfruttando i dati ottenuti dal dataset d'origine attraverso l'elaborazione esplicata nel dettaglio alla sezione 4.3, è possibile costruire un modello di apprendimento che utilizzi le caratteristiche tecniche delle due squadre in competizione per predire, su base statistica, chi sarà il vincitore.

A questo scopo di è scelto di utilizzare una Rete Bayesiana, un modello grafico probabilistico che rappresenta un insieme di variabili stocastiche con le loro dipendenze condizionali attraverso l'uso di un grafo aciclico diretto (DAG). Nel caso in esame, le variabili stocastiche sono rappresentate proprio dai punteggi assunti dalla squadra in casa e quella in trasferta nei 4 ruoli tipici del calcio.

In questo capitolo vi si farà riferimento con la seguente nomenclatura:
`HOME_GK`, `HOME_DEF`, `HOME_MID` e `HOME_ATK` per la squadra che gioca in casa e
`AWAY_GK`, `AWAY_DEF`, `AWAY_MID` e `AWAY_ATK` per la squadra ospite. La variabile target, cioè quella da predire, verrà invece identificata come `WINNER`.

5.1 Assunzioni

La principale assunzione su cui si basa l'intero lavoro di modellazione con le Reti Bayesiane è costituito dal fatto che le due squadre che si sfidano in una partita sono fra loro causalmente indipendenti. Ciò significa che le variabili stocastiche della squadra in casa, e quindi i nodi che compongono la Rete Bayesiana, non devono poter essere raggiungibili attraverso un cammino orientato da parte dei nodi costituenti la squadra in trasferta, e viceversa.

In altre parole, se ciascuna squadra è rappresentata da un arbitrario numero di nodi, questi costituiranno una sorta di componente connessa che interagirà con le altre solamente attraverso dei nodi intermedi con soli archi entranti. Il risultato della partita è un tipico nodo di collegamento fra le due squadre, i cui punteggi che le caratterizzano nei diversi ruoli non si influenzano fra una squadra l'altra.

5.2 Software

Il linguaggio di programmazione utilizzato per la modellazione della Rete Bayesiana su cui effettuare le predizioni è R(5), un ambiente software gratuito per computazione di natura statistica. Più nello specifico, la libreria utilizzata per lo scopo del progetto è `bnlearn`(1), pensata proprio per lavorare con modelli di apprendimento basati sulle Reti Bayesiane.

`bnlearn` mette a disposizione una moltitudine di istruzioni sia per strutturare la Rete che per svolgere predizioni e inferenze di qualsiasi tipo. Fra le funzionalità più vantaggiose, la libreria è in grado di generare anche la struttura fisica della rete basandosi sui dati conosciuti e vantando la capacità di sfruttare più che una decina di algoritmi di apprendimento della struttura. Inoltre, `bnlearn` è in grado di lavorare anche con dataset continui.

La libreria è il principale punto di riferimento per l'utilizzo delle Reti Bayesiane in R ed è stata creata da Marco Scutari, italiano specialista del settore oggi occupato all'Istituto Dalle Molle di Studi sull'Intelligenza Artificiale di Lugano.

Altre librerie utilizzate durante lo sviluppo ed in particolare per la fase di pre-analisi ed elaborazione descritta nel precedente capitolo sono `DataExplorer`, `arules` e `caret` per la visualizzazione e lavorazione dei dati.

5.3 Progettazione di una rete

Come precedentemente accennato, con `bnlearn` è possibile modellare una Rete Bayesiana sia progettandola in autonomia che facendola generare automaticamente alla libreria con la possibilità di sfruttare una moltitudine di approcci diversi.

Gli step per la progrettazione di una Rete sono in primis la modellazione della sua struttura e poi la specifica dei parametri, ossia delle probabilità condizionate di ciascun nodo (CPT). Una peculiarità delle Reti Bayesiane, se confrontate ad altri modelli con cui è possibile fare Machine Learning, è quella di essere completamente visibile agli occhi del programmatore e soprattutto semanticamente significativa, specie per quanto riguarda il DAG rappresentante la struttura della rete stessa. É importante che il grafo della Rete non sia particolarmente complesso, sia per facilitarne la lettura che per evitare un peggioramento delle performance incontrollato.

5.4 Generazione della struttura

Nel caso in esame, si è deciso di sfruttare in primis una rete auto generata dai dati a disposizione. Ciò ha il vantaggio di essere immediatamente funzionante e tendenzialmente cablata per avere delle prestazioni quasi ottimali. La generazione delle struttura avviene tramite un approccio data-driven, sfruttando il dataset su cui si vuole fare predizioni e inferenze. All'approccio numerico basato sui dati, segue spesso un *tuning* effettuato sulla conoscenza del dominio di riferimento, come possono essere ad esempio determinate assunzioni fatte a priori.

5.4.1 Algoritmo di apprendimento

`bnlearn` mette a disposizione del programmatore svariati metodi per l'apprendimento della struttura della Rete: *constraint-based*, *score-based* e *hybrid*. Partendo da una condizione iniziale di indecisione relativo al metodo migliore da utilizzare per il progetto in esame, si è scelto di utilizzare un approccio empirico basato sulla sperimentazione. Provando svariati algoritmi, si è notato che molti di questi lasciavano all'interno della rete dei nodi completamente isolati, motivo per cui sono stati scartati a priori.

Fra i metodi che sono sembrati più "corretti" nella generazione della Rete è emerso *hill-climbing* (HC), un algoritmo *score-based* di ricerca greedy che parte da una condizione di rete senza archi, a cui vengono progressivamente aggiunti, rimossi o invertiti fino alla generazione di una rete che non riesce a ottenere un punteggio migliore. Quest'ultimo è calcolato attraverso una funzione di ottimizzazione applicata alla rete stessa ed utilizzata per apprendere la struttura sub-ottimale(8).

Algorithm 2.2 Hill Climbing Algorithm

1. Choose a network structure G over $\mathbf{V}$, usually (but not necessarily) empty.
2. Compute the score of G, denoted as $Score_G = \text{Score}(G)$.
3. Set $maxscore = Score_G$.
4. Repeat the following steps as long as $maxscore$ increases:
   (a) For every possible arc addition, deletion or reversal not resulting in a cyclic network:
      i. compute the score of the modified network G^*, $Score_{G^*} = \text{Score}(G^*)$.
      ii. if $Score_{G^*} > Score_G$, set $G = G^*$ and $Score_G = Score_{G^*}$.
   (b) update $maxscore$ with the new value of $Score_G$.
5. Return G.

Figura 7: Algoritmo Hill Climbing per l'apprendimento della struttura di una BN

5.4.2 Autogenerazione

Il codice per la generazione e visualizzazione della struttura della Rete Bayesiana con `bnlearn` è decisamente semplice, e porta al risultato mostrato nella figura 8:

```
dag = hc(dataset);
plot(dag);
```

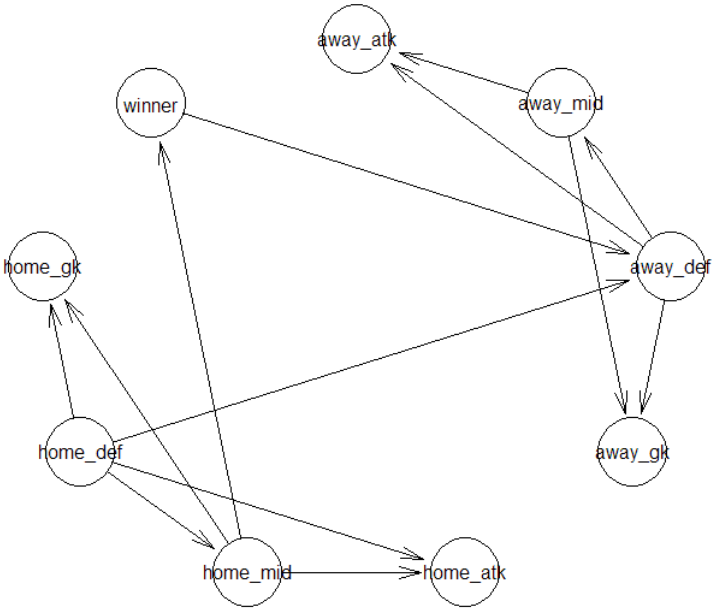


Figura 8: Rete autogenerata con algoritmo HC

Nonostante la rete generata (figura 8) appaia semplice e piuttosto sensata, è doveroso far notare che alcuni archi contraddicono le assunzioni fatte nella sezione 5.1 per cui è stato detto che le due squadre in competizione devono essere composte da nodi causalmente indipendenti. Come si vede nell'immagine, `HOME_MID` ha un arco entrante in `AWAY_DEF` e come se non bastasse anche `WINNER`, la variabile target, va a finire dentro il medesimo nodo. Per questo motivo, si è dimostrato necessario introdurre un intervento manuale per la generazione della rete.

5.4.3 Vincoli di generazione

Attraverso `bnlearn` è possibile specificare alcuni vincoli riguardanti la creazione automatica della struttura della Rete, sia in termini di archi da evitare che di archi da includere nella struttura stessa.

Sfruttando un parametro chiamato `blacklist`, è stata fornita alla funzione `hc` una lista di archi da **non** includere nella rete. Così facendo è stato possibile impedire alle due squadre di collegarsi fra loro, come scelto nelle assunzioni.
Di seguito viene mostrato il codice che genera la rete in figura 9.

```
home_attr = c("home_gk", "home_def", "home_mid", "home_atk");
away_attr = c("away_gk", "away_def", "away_mid", "away_atk");
bl = rbind(
           as.matrix(expand.grid(home_attr, away_attr)),
           as.matrix(expand.grid(away_attr, home_attr))
dag = hc(dfull1, blacklist = bl);
```

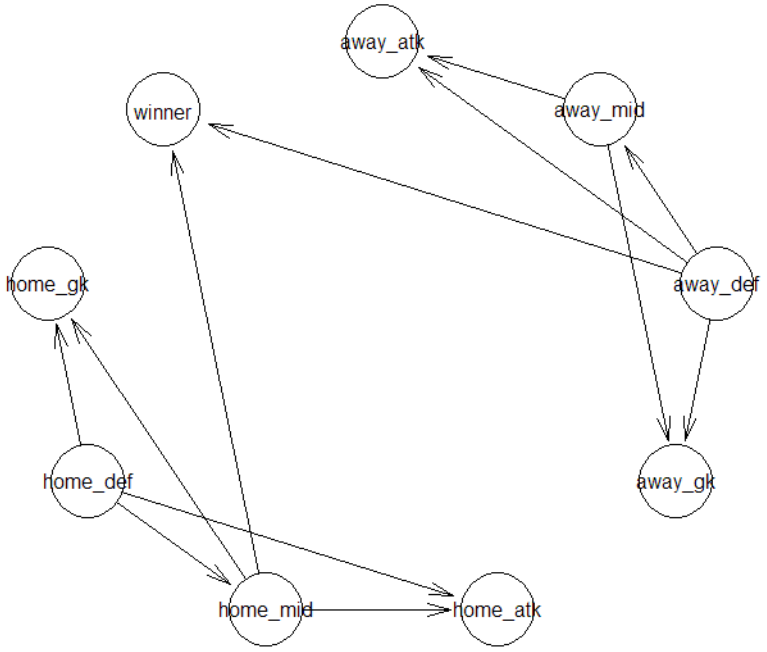


Figura 9: Rete autogenerata con algoritmo HC e blacklist per evitare che le due squadre si connettessero fra loro

Questa volta, le due squadre sono causalmente indipendenti e quindi le assunzioni sono rispettate. Inoltre, è possibile notare che solamente uno dei nodi di entrambe le squadre va a costituire una dipendenza causale diretta con la variabile target, cioè `HOME_MID` e `AWAY_DEF`. Non vi è una vera e propria spiegazione semantica a questo fenomeno, ma intuitivamente si potrebbe pensare che mentre una squadra, quella in casa, punti con questa rete a massimizzare le proprie capacità di possesso palla e attacco (con il centrocampo), l'altra sia prevalentemente interessata a difendersi.

Questo tipo di comportamento è sicuramente originato dal dataset a disposizione, probabilmente anche a causa del fatto che la distribuzione della variabile `WINNER` è mal condizionata: circa nel 50% dei casi vince la squadra in casa, mentre solamente il 25% delle partite è vinta da quella in trasferta, con la rimanente porzione di pareggi.

Quest'ultima intuizione potrebbe essere confermata dal fatto che, riducento il problema ternario `WINNER`=*home/away/draw* ad un problema binario `WINNER`=*home/not home*, la rete cambia nuovamente e questa volta sono entrambi i centrocampi ad influenzare direttamente il target. Lo si può vedere in figura 10.

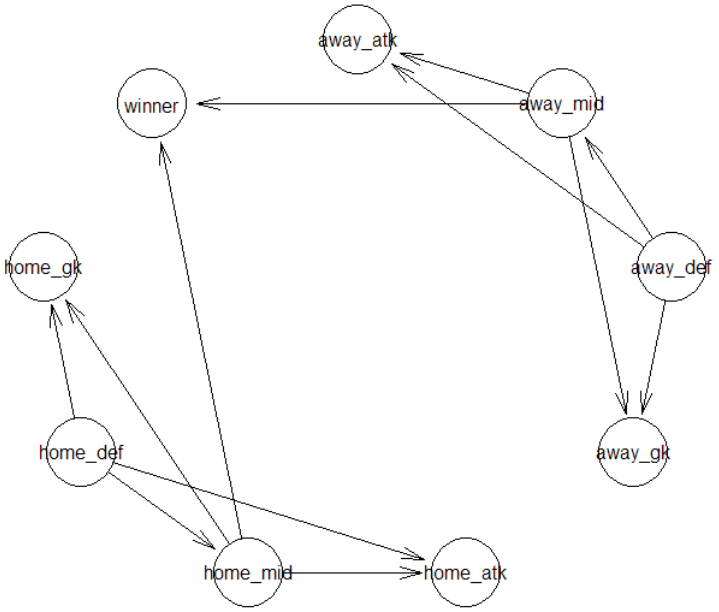


Figura 10: Rete autogenerata con algoritmo HC e blacklist sul problema binario

Si vedranno più avanti in questo elaborato le implicazioni sulle performance derivate dalla considerazione di un problema binario piuttosto che ternario, o viceversa.

5.5 Generazione delle CPT

Appresa (o manualmente generata) la struttura della Rete, è quindi necessario fissare i parametri, cioè le probabilità condizionate di ciascun nodo. Grazie alla proprietà di Markov per cui un nodo è influenzato solo localmente dai nodi facenti parte del suo Markov Blanket, calcolare le CPT è un'operazione relativamente rapida se, come in questo caso, i nodi non hanno un numero particolarmente elevato di archi entranti. Questa proprietà rappresenta un considerevole vantaggio, perché significa che la complessità di calcolare i parametri della rete non cresce con la dimensione della rete ma solamente con il numero di nodi localmente dipendenti.

Anche in questo caso, `bnlearn` fornisce un'istruzione decisamente comoda per la stima dei parametri basandosi sui dati, qui mostrata:

```
fitted = bn.fit(dag, dtrain);
```

A differenza di quanto avviene per la generazione della struttura, l'apprendimento dei parametri dovrebbe essere eseguito solamente sul training set.

5.6 Predizione e inferenza

Una volta generate sia la struttura che i parametri della Rete Bayesiana è finalmente possibile effettuare predizioni e inferenze con il modello appreso. Ancora una volta, le istruzioni sono di semplice utilizzo.

La funzione `predict(object, node, data, method)` permette di generare un insieme di predizioni per una data variabile e successivamente verificare quanto predetto sia coerente con il test set.

```
pred = predict(fitted, "winner", dtest);
cat("Accuracy:", mean(pred == dtest$winner, na.rm = TRUE));
```

La funzione `cpquery(fitted, event, evidence, cluster, method)` consente invece di effettuare inferenze, calcolando la probabilità che un evento si verifichi date le evidenze, sfruttando il modello appreso.

```
res = cpquery(
      fitted,
      event = (winner == "home"),
      evidence = ((home_mid == "terrible") & (away_def == "good")));
print(res);
```

Per completezza, si cita qui anche la funzione `cpdist`, con firma identica a quella di `cpquery`, ma volta alla generazione di campioni casuali date le evidenze e in accordo con le CPT dei nodi di volta in volta campionati.

5.6.1 Qualche esempio

A scopo dimostrativo, si presentano in questa sede alcune delle inferenze che è possibile produrre attraverso la libreria `bnlearn` e l'ultima rete mostrata. I seguenti esempi sono stati generati attraverso l'interfaccia grafica che sarà spiegata nel capitolo 7.

```
P(winner = home | home_mid = terrible, away_mid = good) = 0.184466
P(winner = home | home_mid = terrible, away_mid = terrible) = 0.3825503
P(winner = home | home_mid = terrible, away_mid = excellent) = 0.1244541
P(winner = home | home_gk = very good, away_gk = terrible) = 0.6796657
P(winner = home | home_atk = good, away_gk = excellent) = 0.3128295
P(winner = home | home_atk = good, away_gk = good) = 0.4764493
P(winner = not home | home_mid = terrible, away_mid = excellent) = 0.908642
```

Dai risultati, è possibile facilmente notare come la Rete sia in grado di stimare la probabilità che la squadra di casa sia vincente, sulla base dei valori attribuiti alle diverse variabili secondo la discretizzazione che sarà spiegata nel capitolo 6.1.4. Gli esempi considerano il problema binario, per cui la squadra che gioca in casa può vincere o non vincere (ma senza sapere se si tratti di un pareggio o una sconfitta).

6 Performance

Quando si modellano delle Reti Bayesiane e si sceglie di auto-generare la struttura della rete o comunque provare diverse possibili strutture, è essenziale sfruttare un approccio di tipo sperimentale per determinare, al netto delle assunzioni, quale possa rappresentare la soluzione migliore dal punto di vista delle prestazioni ottenute.

Queste prevedono la necessità di tenere in considerazione diverse metriche standard, che esprimono la bontà di un modello di apprendimento e quindi di predizione sui risultati futuri. Ciò è possibile suddividendo in due (o più) parti il dataset che si vuole fornire all'algoritmo di stima dei parametri della Rete. Ricavando un training set su cui addestrare il modello predittivo e successivamente un test set tramite il quale provare il modello generato, è possibile quindi dedurre le metriche di valutazione.

Al fine di valutare diversi modelli da adottare per questo progetto, le analisi preliminari sulle varie possibilità sono state effettuate attraverso una suddivisione del dataset in 75% per il tranining set e 25% per il test set; successivamente, una k-fold cross validation è stata utilizzata per ottenere valutazioni più accurate.

6.1 Ricerca della soluzione migliore

La ricerca del modello di apprendimento migliore, sia in termini di elaborazione dei dati che generazione della Rete Bayesiana, ha seguito un'evoluzione graduale volta ad individuare la soluzione più accurata per le predizioni sul test set. Questo tipo di valutazione ha coinvolto diversi aspetti che sono di seguito presentati.

6.1.1 Tempi

Un importante aspetto da considerare quando si ha a che fare con modelli di apprendimento, e più in generale con qualsiasi tipo di Machine Learning, è il tempo necessario per imparare i parametri del modello e successivamente stimare il target di nuove istanze. In questo caso specifico, sia l'algoritmo per la generazione della Rete che per la stima dei parametri si sono rivelati decisamente performanti.

L'apprendimento della struttura della Rete è quasi istantaneo per tutti gli algoritmi provati, con tempi di esecuzione prossimi allo zero. L'apprendimento dei parametri della Rete, e quindi delle CPT, richiede invece tempi differenti a seconda della complessità della struttura. I grafi presentati nel capitolo precedente hanno richiesto sempre meno di un secondo per la stima delle CPT. Tale complessità aumenta non con la dimensione della Rete ma con l'aumentare della dipendenza causale (locale) delle variabili stocastiche, quindi nel momento in cui si tenta di stimare i parametri di una Rete nella quale i nodi che ne fanno parte hanno un considerevole numero di genitori; verrà presentato in seguito un esempio di questo tipo di Rete.

Anche i tempi per effettuare predizioni e inferenze sul modello già appreso sono pressoché insignificanti per tutte le alternative provate.

Alla luce di queste considerazioni, il tempo è una metrica piuttosto ininfluente per il caso in esame. L'unica accortezza necessaria è quella di evitare la generazione di strutture particolarmente dense di connessioni, pratica che rappresenta comunque una cattiva scelta anche dal punto di vista dell'accuratezza del modello che ne risulterebbe.

6.1.2 Problema ternario o binario

Come accennato nel precedente capitolo, il dataset utilizzato per apprendere la struttura e i parametri della Rete Bayesiana presenta una cattiva distribuzione per la variabile target `WINNER`. Come mostrato in figura 11, la maggior parte delle partite considerate sono state vinte da parte della squadra in casa, mentre la rimanente parte è quasi equamente suddivisa fra pareggi e vittoria per la squadra in trasferta.

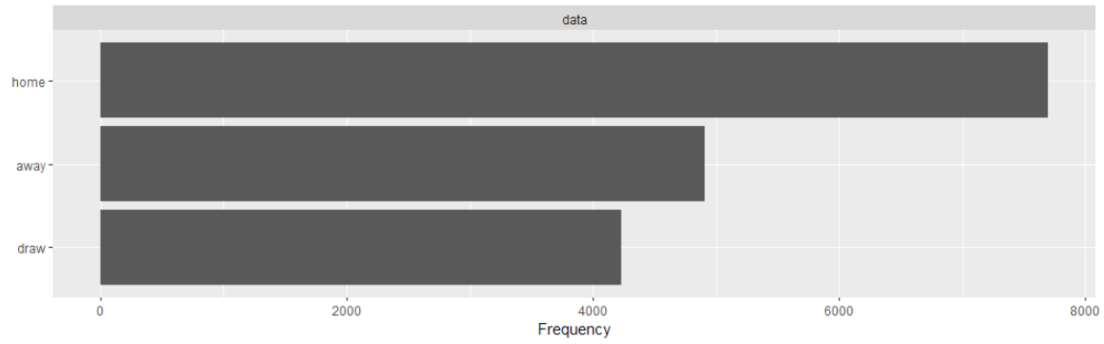


Figura 11: Distribuzione della variabile target

Considerata tale distribuzione e il dominio del problema, è facile aspettarsi che per un modello di predizione possa essere complesso determinare quali siano i casi nei quali a vincere è la squadra ospite o, ancor peggio, si ottenga un pareggio. Il calcio infatti non è chiaramente una scienza esatta, motivo per cui le statistiche delle due squadre non bastano autonomamente per determinare l'esito della partita, specialmente qualora queste abbiano caratteristiche simili; in egual modo, è difficile determinare le motivazioni per cui due squadre sulla carta molto diverse abbiano potuto terminare la partita con un egual numero di goal.

Per questi motivi, ci si aspetta che il modello di Rete Bayesiana abbia delle cattive performance proprio per quanto riguarda la predizione di un modello a tre classi (`WINNER` = *home/away/draw*), ed in particolare con quanto ottenuto nelle due classi in minoranza. A dimostrazione di ciò, vengono presentate in tabella 7 le metriche

di *precision*, *recall* e *F1* calcolate come media di una 10-fold cross validation sulla Rete Bayesiana che era stata presentata in figura 10.

Tabella 7: 10 fold cross validation sul problema ternario

Classe	Precision	Recall	F1
HOME	0.5238946	0.849281	0.6477723
AWAY	0.482222	0.4263695	0.4516147
DRAW	0	0	0

Nonostante l'*accuracy* del modello sia del 0.5126, che non è un risultato particolarmente cattivo per un problema ternario di questo tipo, è possibile notare in tabella come i dati per la classe *draw* siano pari a zero, indice del fatto che il modello non è mai in grado di predire una partita con risultato di pareggio.

Alla luce di questa considerazione, si è deciso di considerare per le prossime valutazioni solamente il problema di natura binaria in grado di predire se il vincitore di una partita sia la squadra in casa o meno; in quest'ultimo caso, non sarà dato sapere se il match sia più incline a concludersi con un pareggio o con la vittoria della squadra ospite. Come si vedrà nella sezione 6.2 dedicata alla cross fold validation, il modello addestrato sul problema binario sarà in proporzione meno accurato, ma privo di mal condizionamenti causati da una cattiva distribuzione della variabile target nel dataset.

6.1.3 Struttura della rete

Per quanto riguarda la struttura della rete, si è scelto di considerare quella in figura 10 come punto di partenza, generata attraverso l'algoritmo *hill climbing* sul problema binario, assicurandosi che le due squadre non avessero collegamenti diretti.

Poiché con tale configurazione le uniche variabili collegate al nodo target `WINNER` sono risultate `HOME_MID` e `AWAY_MID`, si è pensato di costituire una rete con un numero maggiore di collegamenti al nodo su cui effettuare le predizioni. Aumentando il numero di archi della Rete, aumenta conseguentemente anche il tempo necessario per la stima dei parametri.

Fra le varie alternative, nessuna è risultata in termini di *accuracy* più performante rispetto alla Rete originale, che si attesta intorno al 60%. Per questo motivo, si è dedotto che la rete ottenuta attraverso *hill climbing* con un ridotto numero di archi fosse la migliore, e pertanto anche la semplicità della Rete stessa sia da preferire quando si ha a che fare con modelli Bayesiani.

Nella figura 12 è possibile osservare un esempio di Rete fra quelle testate, in cui tutti i nodi sono collegati alla variabile target secondo un'assunzione per cui ognuno di questi dovrebbe avere influenza sul risultato di una partita. Questa Rete si è rivelata tuttavia considerevolmente peggiore, con un'*accuracy* intorno al 55% e dei tempi di stima delle CPT decisamente più importanti, con alcune righe addirittura impossibili da calcolare.

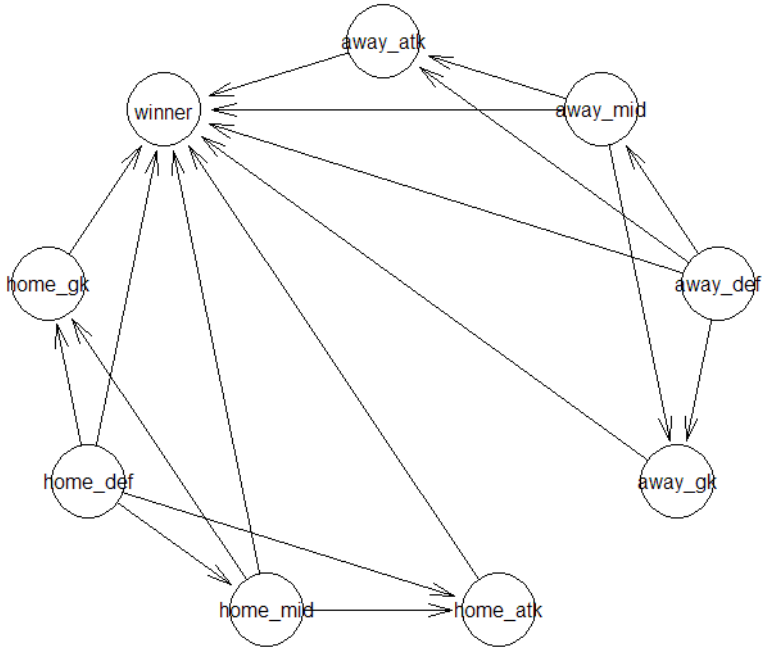


Figura 12: Rete che collega tutte le variabili al target

Un ulteriore tentativo lo si è fatto tentando di aggiungere due nuove feature per ciascuna squadra, che raggruppassero rispettivamente il potenziale difensivo (portiere + difesa + mid) e offensivo (mid + attacco) delle squadre. Avendo provato a generare una Rete Bayesiana con questi nuovi nodi, si sono comunque ottenute performance totalmente simili alle precedenti, motivo per cui si è poi deciso di tornare alla configurazione della Rete originale.

6.1.4 Discretizzazione delle features

Nonostante l'argomento di discretizzazione sia già stato affrontato nel capitolo 4.3.5, poiché si tratta di un'operazione da eseguire sui dati prima di fornirli alla Rete, tale argomento è tornato in auge per fare le ultime considerazioni sulle performance del modello. Infatti, il modo in cui vengono discretizzati i dati assume un ruolo fondamentale per l'intero processo di predizione, in quanto il modello sia in fase di apprendimento che di predizione sfrutta i valori assunti dalle variabili considerate ed espressi secondo la discretizzazione scelta.

Ci sono diversi metodi per discretizzare e per operare questo tipo di elaborazione si è utilizzato il package `arules` di R, che fra le più rilevanti mette a disposizione la discretizzazione *fixed* e quella *frequency-based*. Mentre la prima consente di specificare dei valori che costituiscono gli estremi degli intervalli di discretizzazione, il secondo metodo permette di effettuare una separazione automatica basata sulla distribuzione dei valori assunti dalla variabile che si sta discretizzando. L'obiettivo è quello di creare intervalli che contengano un numero bilanciato di campioni del dataset.

Considerata la distribuzione delle feature scelte per costruire la Rete Bayesiana, visibile nell'immagine 13, si è inizialmente pensato di discretizzare i valori in 4 intervalli. Particolare importanza è stata data alla scelta di un numero di intervalli pari, per evitare che troppe feature finissero nel valore centrale.

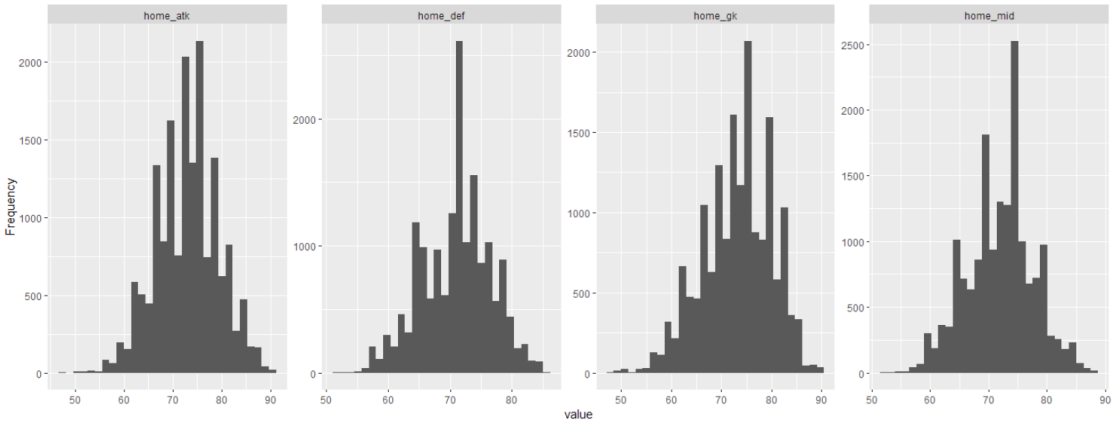


Figura 13: Distribuzione dei valori assunti dalle features

In seguito, poiché si è notato che i valori centrali delle feature fossero assunti da un gran numero di partite si è deciso di aumentare il numero di intervalli con cui discretizzare. Ciò ha portato ad un incremento nell'accuratezza dei modelli di predizione

testati, motivo per cui si è rivelata un'intuizione vincente. Si è quindi scelto di delegare al pacchetto `arules` la scelta degli intervalli in maniera automatica attraverso un algoritmo basato sulla frequenza di distribuzione.

Si mostrano di seguito i limiti degli intervalli adottati per ciascun attributo della Rete e in figura 14 anche la distribuzione assunta delle feature della squadra di casa in seguito alla discretizzazione. Si nota facilmente come il dataset si distribuisca equamente sui 6 intervalli creati, rinominati con delle etichette testuali e semanticamente rilevanti per agevolarne la lettura.

```
Discretization intervals for  home_gk :     48, 66, 71, 74, 77, 80, 90
Discretization intervals for  home_def :    52, 65, 69, 72, 74, 76, 86
Discretization intervals for  home_mid :    52, 66, 70, 72, 74, 77, 88
Discretization intervals for  home_atk :    48, 67, 70, 73, 76, 79, 91
Discretization intervals for  away_gk :     49, 66, 71, 74, 77, 80, 90
Discretization intervals for  away_def :    52, 65, 69, 72, 74, 76, 86
Discretization intervals for  away_mid :    51, 66, 70, 72, 74, 77, 89
Discretization intervals for  away_atk :    46, 67, 70, 73, 76, 79, 92
```

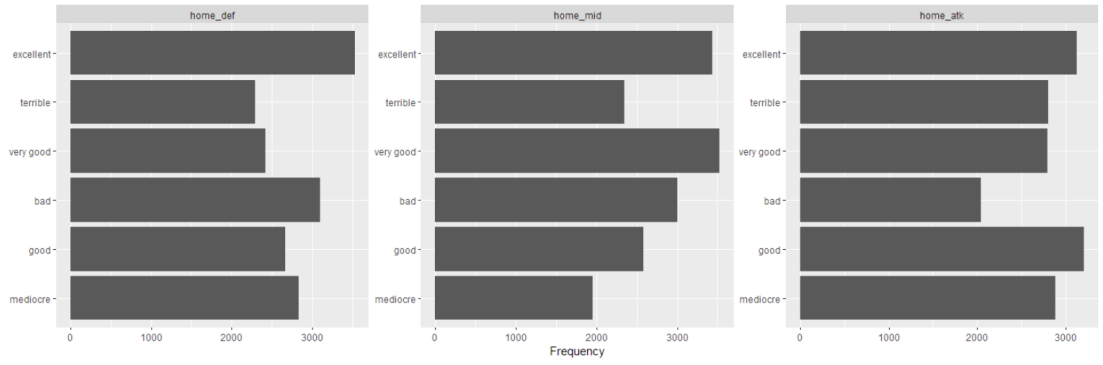


Figura 14: Distribuzione dei valori assunti dalle features dopo la discretizzazione

6.1.5 Esperimento per il calcolo del punteggio squadre

Come ultimo tentativo di *tuning* delle performance della rete si è provato a modificare il calcolo del punteggio assegnato a ciascuna squadra nei diversi ruoli calcistici. Si è rivalutato quanto spiegato nel capitolo 4.3.5, poiché ci si è resi conto che quel calcolo non tenesse conto del numero di giocatori in un certo ruolo ma solamente della media fra gli `overall_rating`.

Al fine di pesare diversamente il punteggio ottenuto da una squadra per un determinato ruolo, ad esempio se questa sceglie di schierare 1 piuttosto che 4 attaccanti, si è deciso di applicare al calcolo della media una modifica che tenesse conto del numero

di giocatori. Per pesare diversamente il beneficio ottenuto dall'aggiunta o la rimozione di giocatori in differenti ruoli, si è pensato di applicare un *reward* logaritmico per ciascuna modifica, considerando la distribuzione media dei giocatori per ruolo (figura 1) come situazione standard per costruire la formazione.

Questa ne è risultata la nuova formula per il calcolo dei punteggi. Si consideri il caso del punteggio d'attacco per la squadra ospite, in cui `overall` è il vecchio valore calcolato (ossia la media dei punteggi ottenuti dagli attaccanti della squadra) e `avg_atk_count` rappresenta il numero medio di attaccanti presenti in tutte le partite del dataset. La costante `K` ha il puro scopo di riscalare il valore del logaritmo ad un valore apprezzabile se si considera che questo verrà sommato ad un valore compreso fra 1 e 100 (con media 70); tale valore è stato impostato a 20.

$$new\_overall = overall + K * log(\frac{away\_atk}{avg\_atk\_count})$$

Nonostante la formula possa rappresentare un buon criterio per assegnare o rimuovere una certa ricompensa in base al numero di giocatori impiegati in un certo ruolo, dal punto di vista pratico si è rivelata di fatto completamente ininfluente per effettuare le predizioni tramite la Rete Bayesiana. Tale ipotesi di *tuning* è stata quindi abbandonata in favore, ancora una volta, di una trattazione più semplice.

6.2 Cross Fold Validation

Scelta la migliore elaborazione sulle feature da considerare e generata la struttura ottimale della Rete Bayesiana, abbiamo quindi applicato tutte le metriche di valutazione al modello appreso da `bnlearn`. In particolare, abbiamo deciso di implementare una cross fold validation sia attraverso una funzione di libreria, sia in maniera personalizzata.

6.2.1 Funzione `bn.cv()`

La libreria `bnlearn` mette a disposizione una comoda funzione `bn.cv(data, bn, loss)` per effettuare la cross fold validation. Questa consente di applicare diverse strategie di validazione e calcolare la *loss* con svariati criteri di valutazione.

Per il progetto in esame, avendo a che fare con variabili di natura discreta si è optato per una *loss* espressa come *Classification Error* calcolata su una 10-fold validation standard. Tale *loss* rappresenta l'errore di predizione per un singolo nodo, risultando quindi particolarmente indicata per valutare le performance della Rete nel predire la variabile target `WINNER`. Per il calcolo, viene sfruttata la proprietà di dipendenza locale della rete, quindi solo i genitori del nodo su cui si sta effettuando la predizione.

In seguito è riportato il codice per effettuare la cross validation e l'output della stessa, che riporta sia la struttura della rete che il valore di *loss* ottenuto. Quest'ultimo è coerente con l'*accuracy* che verrà mostrata nella sezione successiva.

```
cv = bn.cv(method = "k-fold", k = 10, data = dtest, bn = dag,
           loss = "pred", loss.args = list(target = "winner"));
print(cv)

k-fold cross-validation for Bayesian networks

target network structure:
[home_def][away_def][home_mid|home_def][away_mid|away_def]
[winner|home_mid:away_mid][home_gk|home_def:home_mid]
[home_atk|home_def:home_mid][away_gk|away_def:away_mid]
[away_atk|away_def:away_mid]

number of folds:                10
loss function:                  Classification Error
training node:                  winner
expected loss:                  0.3809976
```

6.2.2 Accuracy, Precision e Recall

Poiché la funzione integrata in `bnlearn` per la cross validation prevede come output solamente il valore della *loss*, si è scelto di implementare anche una cross validation personalizzata per calcolare i valori di *accuracy*, *precision*, *recall* e *F1* al fine di valutare la bontà delle predizioni.

Calcolando tali valori su 10 fold e mediandone i risultati, si ottiene un'*accuracy* del 62.57% e le metriche riportate in tabella 8. Queste tengono in considerazione la predizione della variabile `WINNER` con il valore *home*, cioè considerando la squadra che gioca in casa come vincitrice.

Tabella 8: 10 fold cross validation SVM

Classe	Precision	Recall	F1
HOME	0.636251	0.4293271	0.5117034

A differenza di quanto mostrato in tabella 7 per il problema multiclasse, si può questa volta notare un peggioramento delle performance generali *F1* per la classe

home, a fronte però di un risultato non completamente sbilanciato in sfavore di una certa classe, come accadeva per *away* e *draw* sul problema a tre classi.

Dalle metriche misurate si evince un valore accettabile di veri positivi nel considerare la squadra di casa vincente, ma anche una considerevole percentuale di falsi negativi.

Probabilmente ciò è dovuto all'effettiva difficoltà nel riuscire a predire il vincitore di una partita di calcio se ci si limita a osservare le caratteristiche dei giocatori che le compongono. Ciò accade perché molte volte si assiste a partite nelle quali le due squadre sfidanti non sono particolarmente sbilanciate in termini di potenzialità, bensì del tutto comparabili. In questi casi predire un vincitore un'operazione diventa decisamente complessa per un modello di predizione automatizzato tanto quanto lo è per un esperto del settore.

6.2.3 ROC curve e AUC

Come ultima metrica di valutazione si è scelto di considerare anche il grafico della *receiver operating characteristic curve* per mettere in relazione la percentuale di veri e falsi positivi. Questa si può osservare in figura 15.

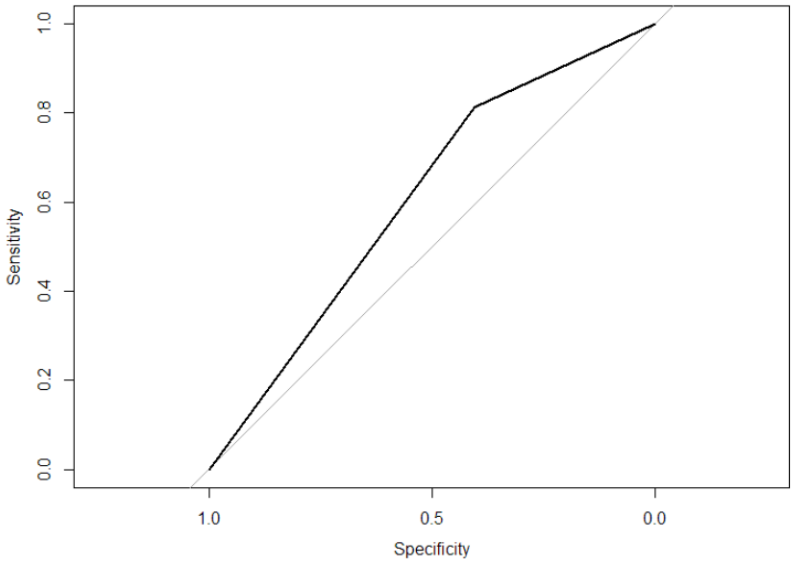


Figura 15: Grafico della ROC curve

Dal grafico della ROC è quindi stato possibile calcolare l'*area under curve*, che determina la bontà generale del modello di predizione.
Il valore dell'AUC è risultato pari a 0.6103608.

7 Web Demo

Al fine di dimostrare l'utilizzo della rete in un applicativo software, viene qui illustrato lo sviluppo e il funzionamento di un'applicazione web che permette la configurazione delle rose dei due team coinvolti in una partita e la conseguente predizione del vincitore. Un ulteriore interfaccia permette anche la visualizzazione della Rete Bayesiana su cui effettuare inferenze specificando le variabili della query che si vuole sottoporre al modello Bayesiano.

7.1 Architettura

L'interfaccia web è stata sviluppata utilizzando l'architettura a 3 layer, con separazione di frontend, backend e database.

Il database utilizzato in fase di lettura è quello fornito inizialmente, senza alcuna modifica. Esso consiste quindi in un file SQLite interrogabile e modificabile semplicemente tramite un web server. Questo risulta particolarmente utile per fornire i dettagli dei giocatori ed eventualmente dei team così che l'utente possa visualizzarli e sceglierli attraverso l'opportuna interfaccia.

Per lo sviluppo del backend è stato deciso di utilizzare l'engine Javascript tramite il popolare progetto Node.js (4). Esso è in grado di agire come middleware tra il frontend e il database, separando al meglio le logiche di manipolazione del dato. É inoltre incaricato di chiamare adeguatamente lo script R per la predizione del vincitore della partita e per svolgere le inferenze richieste.

Il frontend è invece sviluppato utilizzando la libreria Javascript React.js (6)

7.2 Interfaccia grafica

Di seguito vengono mostrate brevemente le principali schermate dell'applicazione e il loro funzionamento.

All'avvio dell'app, utilizzando semplicemente un browser web, è possibile notare la schermata principale con 3 tab di selezione: *Home team*, *Away team* e *Results*. Mentre i primi 2 permettono di modificare le squadre, il terzo consente invece di visualizzare i risultati data la configurazione precedente.

La schermata di configurazione della squadra mostrata nella figura 16 sulla sinistra l'immagine di un campo da calcio su cui è possibile spostare i giocatori, scelti da un menù a tendina, per creare la formazione da utilizzare.

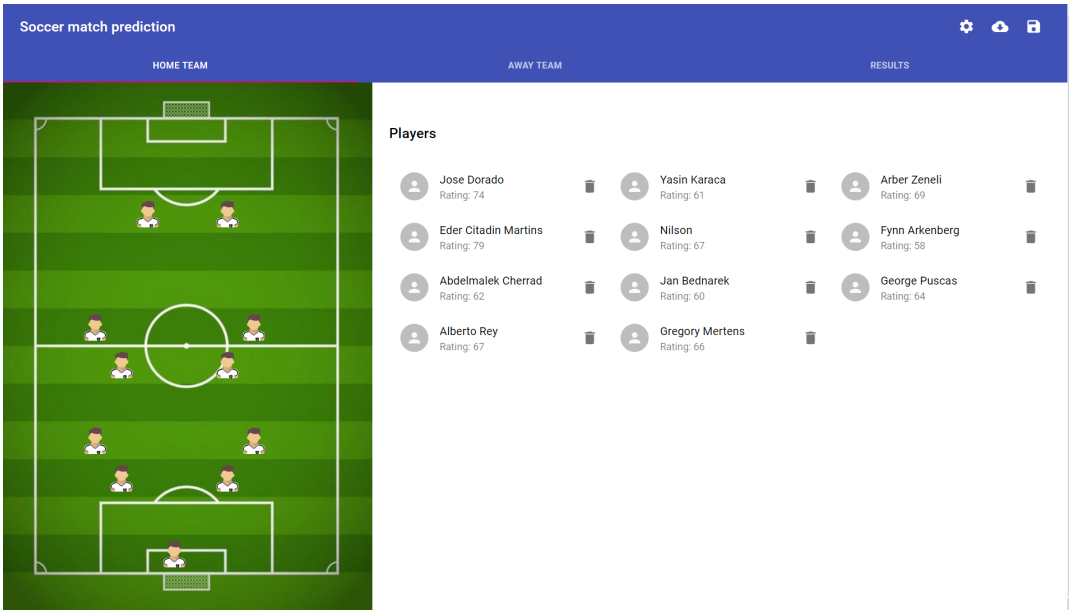


Figura 16: Configurazione team

Ultimata la configurazione, cliccando sul tab dei risultati è possibile vedere in verde la squadra con la maggiore probabilità di vincere ed in rosso l'altra. Nel caso in cui la squadra vincente (verde) non dovesse essere quella che gioca in casa, è opportuno ricordare che poiché il modello considera un output binario, la partita è volta a favore della squadra in trasferta ma senza tuttavia riuscire a specificare se si tratta di una vittoria per quest'ultima, piuttosto che di un pareggio. Un esempio di schermata viene presentato nella figura 17.

Per non dover ricreare configurazioni dall'inizio ogni qualvolta si avvii l'applicazione, nella parte superiore destra dello schermo sono presenti dei comandi per salvare la formazione scelta o caricarne di precedenti, come mostrato nelle figure 18 (a) e 18 (b).

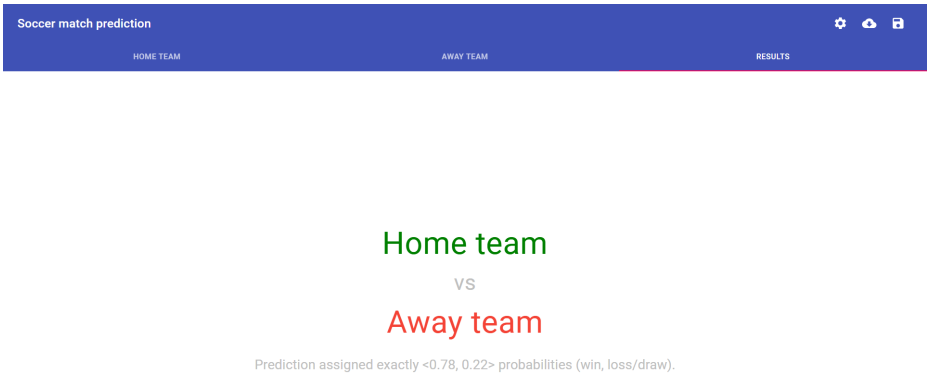
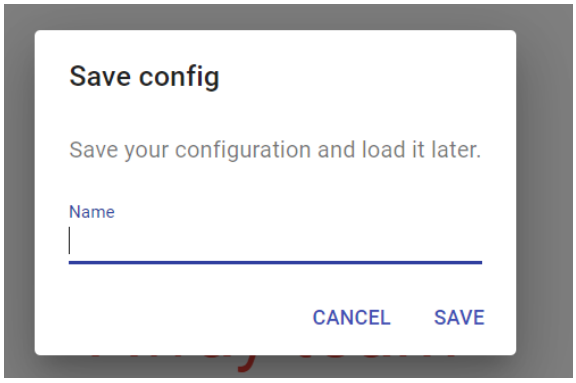
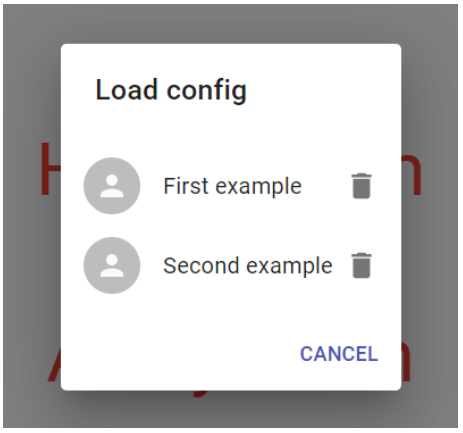


Figura 17: Risultati match



(a) *Salvataggio configurazione*



(b) *Caricamento configurazione*

Figura 18: Salvataggio e caricamento configurazioni

7.3 Predizione del risultato

Per predire il vincitore della partita, come mostrato precedentemente, è stato necessario utilizzare la rete Bayesiana ottenuta a seguito degli esperimenti. Tale rete è stata salvata in un file che è possibile ricaricare nell'ambiente R al bisogno. Il web server Javascript è dunque in grado di chiamare lo script R da riga di comando, fornendogli in input una rappresentazione in formato JSON dei giocatori in campo con le loro caratteristiche. Lo script non deve far altro che manipolare i dati che gli sono

stati forniti secondo quando spiegato nel capitolo 4.3 e successivamente applicare l'inferenza.

Per mostrare inoltre il puro funzionamento della rete è stato sviluppato un semplice form che permette l'inserimento delle evidenze e fornisce in output la probabilità che un certo evento si verifichi, coerentemente a quanto specificato dall'utente.

Premendo il primo pulsante a sinistra nella parte superiore destra dello schermo è possibile aprire la schermata mostrata nella figura 19. Nella parte sinistra dello schermo viene mostrata la topologia della rete utilizzata, con la possibilità di effettuare zoom e muoverla tramite il mouse, per una più agevole visualizzazione.

Nella parte destra vengono invece mostrati una serie di campi di input per configurare evidenze e variabile query. Una volta impostate è possibile premere il tasto *Compute* per calcolarne il risultato, che verrà mostrato sotto al bottone stesso. Nella rete è possibile inoltre visualizzare i nodi delle evidenze di colore rosso e la query di colore verde, per meglio comprendere a quali nodi si riferiscono le scelte dell'utente. É opportuno segnalare inoltre che gli indentificativi dei nodi possono essere visualizzati direttamente sulla rete semplicemente con il passaggio del mouse.

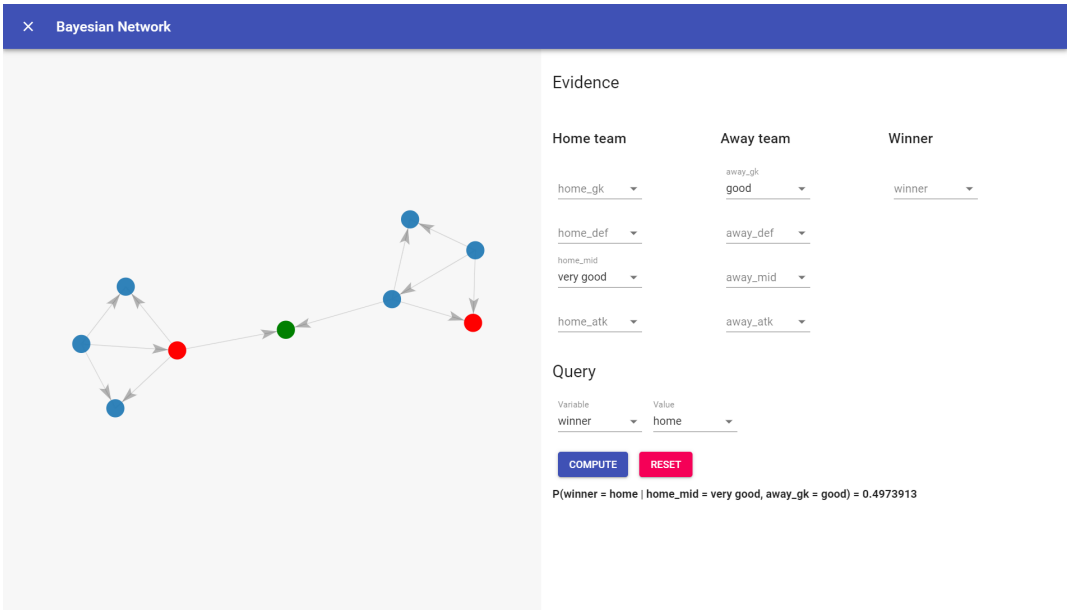


Figura 19: Schermata di inferenza sulla Rete Bayesiana

8 Conclusioni

Lorem ipsum dolor sit amet.

Riferimenti bibliografici

[1] bnlearn - an r package for bayesian network learning and inference. URL: http://www.bnlearn.com/.

[2] Ea sports fifa videogame. URL: https://www.ea.com/it-it/games/fifa.

[3] Kaggle dataset source. URL: https://www.kaggle.com/hugomathien/soccer.

[4] Node.js. URL: https://nodejs.org/it/.

[5] R programming language, free software environment for statistical computing and graphics. URL: https://www.r-project.org/.

[6] React. URL: https://reactjs.org/.

[7] Sqlite. URL: https://www.sqlite.org/index.html.

[8] Marco Scutari. *Measures of Variability for Graphical Models*. PhD thesis, Universita degli Studi di Padova, 2011. URL: http://www.bnlearn.com/about/thesis/thesis.pdf.