

Università degli Studi di Padova
DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"
CORSO DI LAUREA MAGISTRALE IN INFORMATICA



**Il modello Bradley-Terry per l'analisi delle
partite della Serie A italiana di calcio**

Tesi di laurea magistrale

Relatore

Prof. Annamaria Guolo

Laureando

Federico Perin

ANNO ACCADEMICO 2022-2023

ABSTRACT

Viviamo nell'era dei cosiddetti *Big Data*, dove grazie all'interconnessione, un grande flusso di informazioni e di dati può essere ricavato da ogni possibile attività.

Non fa eccezione il calcio in cui da un paio d'anni, le società calcistiche si affidano a sistemi di analisi per produrre tattiche di gioco ma anche per effettuare *scouting* di giocatori emergenti. Nel calcio moderno, perciò, numerose statistiche ad esempio il possesso della palla, il numero di tiri effettuati da una squadra ecc. vengono raccolte durante una partita di calcio.

Tale fatto scaturisce l'attenzione su un ulteriore tematica d'analisi: dato che si hanno a disposizione un gran numero di dati sulle prestazioni delle squadre nelle loro partite, è possibile individuare quali variabili vanno ad influenzare in modo significativo il successo o il fallimento sportivo delle singole squadre?

Da questo quesito nasce la tesi qui presentata. L'obiettivo è quello di presentare un'analisi che risponda a tale quesito, attraverso l'utilizzo di tecniche di *data mining*, in particolare lo sfruttamento di un modello di confronto a coppie per le partite di calcio in grado di tenere conto delle variabili esplicative specifiche per le partite. Il modello scelto per l'analisi sarà il modello *Bradley-Terry* con le sue estensioni. Infine, verrà presentata un'applicazione di metodi di *machine learning* per la predizione dei risultati delle singole partite e l'individuazione delle *features* più importanti.

Lo studio prenderà in considerazione i dati relativi alle partite della Serie A italiana della stagione 2021/2022.

“Il successo è deformante: rilassa, inganna, ci rende peggiori. Al contrario, l’insuccesso è formativo: ci rende stabili, ci avvicina alle nostre convinzioni, ci fa ritornare a essere coerenti.”

— Marcelo Bielsa

RINGRAZIAMENTI

Innanzitutto, vorrei esprimere la mia gratitudine alla Prof. Annamaria Guolo, relatrice della mia tesi, per l’aiuto ed il sostegno fornитomi durante tutto il lavoro.

Desidero ringraziare con affetto i miei genitori per il sostegno, per il grande aiuto che mi hanno dato e per essermi stati vicini in ogni momento durante gli anni di studio.

Voglio inoltre ringraziare i miei amici per questi due bellissimi anni trascorsi assieme. Un ringraziamento speciale per Alberto e Stefano che hanno reso questa fase della mia vita indimenticabile.

Padova, Febbraio 2023

Federico Perin

INDICE

1	Introduzione	1
1.1	Dominio del problema	1
1.2	Applicazione	2
1.3	Tecnologie e Strumenti utilizzati	2
1.3.1	Tecnologie	2
1.3.2	Strumenti	3
1.4	Motivazioni personali	3
1.5	Struttura della tesi	3
2	Serie A 2021/2022 dataset	5
2.1	Serie A 2021/2022	5
2.1.1	Ranking	5
2.2	Costruzione del dataset	5
2.3	Struttura del dataset	7
2.3.1	Dati generali	8
2.3.2	Dati relativi ai tiri	9
2.3.3	Dati relativi al possesso	10
2.3.4	Dati relativi ai passaggi	13
2.3.5	Dati difensivi	15
3	Analisi dei dati	19
3.1	Preprocessing dei dati	19
3.2	Analisi grafica dei dati	19
3.2.1	Relazione tra la variabile risposta e le covariate	20
3.2.2	Analisi possibili interazioni	29
3.3	Ulteriori modifiche del dataset	39
3.3.1	Modifiche per il pacchetto BradleyTerry2	39
3.3.2	Modifiche per il pacchetto BTLLasso	40
4	Il modello Bradley-Terry	43
4.1	Introduzione al Modello Bradley-Terry	43
4.2	Modello Bradley-Terry con categorie di risposta ordinate	44
4.3	Modello Bradley-Terry con effetti dell'ordine	45
4.4	Modello Bradley-Terry con variabili esplicative	46
4.5	Stima e penalizzazione	47
4.5.1	LASSO	49
4.5.2	Scelta del parametro di tuning	50
5	Risultati dei modelli Bradley-Terry	51
5.1	Premesse	51
5.2	BTM con effetto dell'ordine	51
5.3	BTM con covariate specifiche dell'oggetto	53

5.4	Modello Bradley-Terry e LASSO	57
5.5	BTM senza l'intercetta e con LASSO	71
5.6	Predizioni	85
6	Algoritmi di machine learning	89
6.1	Componenti essenziali	89
6.1.1	Distanza di Minkowski	89
6.1.2	Funzione kernel	89
6.1.3	Bootstrap	90
6.1.4	Bagging	90
6.1.5	Boosting	91
6.2	K-Nearest-Neighbors	92
6.3	Support Vector Machine	94
6.4	Decision Tree	95
6.5	Random Forest	96
6.6	AdaBoost	97
7	Risultati dei Algoritmi di machine learning	99
7.1	Premesse	99
7.2	Ulteriori metriche	99
7.3	K-Nearest-Neighbors	100
7.4	Support Vector Machine	102
7.5	Decision Tree	104
7.6	Random Forest	108
7.7	AdaBoost	112
8	Modello Bradley-Terry per il numero di gol di scarto	115
8.1	Premessa	115
8.2	Modello Bradley-Terry con Y=5 e Lasso	115
8.3	Predizioni	125
9	Discussione e comparazione dei risultati	127
9.1	Discussione risultati dei modelli Bradley-Terry	127
9.2	Discussione risultati dei modelli di machine learning	129
9.3	Confronto	130
10	Conclusioni	131
11	Codice in R	133
11.1	extractRowsAway	133
11.2	createYFull	134
11.3	extractData	135
11.4	extractAll	136
11.5	colLabel	137
11.6	rowLabel	137
11.7	Librerie	137
12	Codice in Python	139
12.1	createTable	139
12.2	Librerie	140

INDICE

ix

Bibliografia

141

Sitografia

143

ELENCO DELLE FIGURE

2.1	Grafico della classifica del campionato italiano della Serie A 2021/2022. Le barre di color azzurro indicano i punti che la squadra ha guadagnato in partite casalinghe. Le barre di color rosa indicano i punti che la squadra ha guadagnato in partite fuori casa.	6
2.2	Logo di FBref.	6
2.3	Rappresentazione del fuorigioco	9
2.4	In rosso l'area di rigore in un campo da calcio.	11
2.5	In rosso la mediana nel campo da calcio.	11
2.6	In rosso il centrocampo nel campo da calcio.	12
2.7	In rosso la trequarti dell'avversario nel campo da calcio.	13
2.8	Esecuzione di un passaggio filtrante	14
2.9	Esecuzione di un cambio di gioco	15
2.10	Rappresentazione di un cross	16
2.11	Rappresentazione di un contrasto in scivolata	17
3.1	Barplot della distribuzione della variabile di risposta Res	20
3.2	Barplot della distribuzione della variabile di risposta per squadraRes .	21
3.3	Barplot della distribuzione del numero di gol a partita. In azzurro vengono indicati i gol per la squadra in casa mentre in rosa i gol per la squadra ospite	22
3.4	Mosaicplot che mostra la distribuzione degli esiti rispetto alle partite giocate in casa e fuori casa	22
3.5	Boxplot della distribuzione della variabile Poss rispetto ai valori della variabile risposta Res	23
3.6	Boxplot della distribuzione della variabile SoT rispetto ai valori della variabile risposta Res	24
3.7	Boxplot della distribuzione della variabile G/Sh rispetto ai valori della variabile risposta Res	24
3.8	Boxplot della distribuzione della variabile Saves rispetto ai valori della variabile risposta Res	25
3.9	A sinistra il boxplot della variabile numerica PAtt rispetto ai valori della variabile risposta Res e a destra il boxplot della variabile numerica PCmp% rispetto ai valori della variabile risposta Res	26
3.10	Boxplot della distribuzione della variabile ToDefPen rispetto ai valori della variabile risposta Res	27
3.11	Boxplot della distribuzione della variabile ToAttPen rispetto ai valori della variabile risposta Res	28
3.12	A sinistra il boxplot della variabile numerica F1s rispetto ai valori della variabile risposta Res e a destra il boxplot della variabile numerica F1d rispetto ai valori della variabile risposta Res	29

3.13 Boxplot della distribuzione della variabile <code>Int</code> rispetto ai valori della variabile risposta <code>Res</code>	30
3.14 Boxplot della distribuzione della variabile <code>TklWin</code> rispetto ai valori della variabile risposta <code>Res</code>	30
3.15 Boxplot della distribuzione della variabile <code>Recov</code> rispetto ai valori della variabile risposta <code>Res</code>	31
3.16 Grafico delle correlazioni di ogni coppia di variabili	32
3.17 Scatterplot della distribuzione della variabile <code>Sh</code> rispetto ai valori della variabile <code>ToAttPen</code>	33
3.18 Scatterplot della distribuzione della variabile <code>Sh</code> rispetto ai valori della variabile <code>G/Sh</code>	34
3.19 Scatterplot della distribuzione della variabile <code>Sh</code> rispetto ai valori della variabile <code>Poss</code>	35
3.20 Scatterplot della distribuzione della variabile <code>ToMid3rd</code> rispetto ai valori della variabile <code>LPAtt</code>	36
3.21 Scatterplot della distribuzione della variabile <code>ToMid3rd</code> rispetto ai valori della variabile <code>PCmp%</code>	37
3.22 Scatterplot della distribuzione della variabile <code>TotDist</code> rispetto ai valori della variabile <code>PCmp%</code>	37
3.23 Scatterplot della distribuzione della variabile <code>PAtt</code> rispetto ai valori della variabile <code>PCmp%</code>	38
3.24 Scatterplot della distribuzione della variabile <code>ToDefPen</code> rispetto ai valori della variabile <code>ToAttPen</code>	39
 5.1 Barplot che indica per ogni squadra l'abilità stimata dal modello (4.9). A fianco al grafico viene riportato lo <i>Standard Error</i> (SE), il <i>Quasi Standard Error</i> (QSE) e il <i>Quasi Variance</i> (QV). Nel grafico viene indicato con un asterisco le squadre con un piazzamento stimato diverso da quello reale, anche esso riportato a destra del grafico.	52
5.2 Barplot che indica per ogni squadra l'abilità stimata dal modello (5.1). A fianco al grafico vengono riportati i relativi <i>Standard Error</i> (SE), <i>Quasi Standard Error</i> (QSE) e <i>Quasi Variance</i> (QV). Nel grafico viene indicato con un asterisco le squadre con un piazzamento stimato diverso da quello reale, anche esso riportato a destra del grafico	54
5.3 Barplot che indica per ogni squadra l'abilità stimata dal modello (4.12). Viene indicato con un asterisco le squadre con un piazzamento stimato diverso da quello reale anche esso riportato a destra del grafico.	58
5.4 Grafico dell'andamento delle prestazioni del modello (4.12) su tutti i trenta valori assunti dal parametro di tuning, indicato con il simbolo λ , durante l'applicazione di K-Fold Cross Validation. L'andamento viene valutato in termini di Ranked Probability Score (RPS). La linea rossa tratteggiata indica il parametro di tuning λ ottimo da utilizzare.	62
5.5 Grafico che riporta l'andamento stimato dal modello (4.12) della stima del possesso della palla per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	63
5.6 Grafico che riporta l'andamento stimato dal modello (4.12) della stima del numero di tiri in porta per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	63

5.7 Grafico che riporta l'andamento stimato dal modello (4.12) della stima del numero di passaggi corti tentati per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	64
5.8 Grafico che riporta l'andamento stimato dal modello (4.12) della stima della percentuale di passaggi corti riusciti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	65
5.9 Grafico che riporta l'andamento stimato dal modello (4.12) della stima della percentuale di passaggi medi riusciti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	65
5.10 Grafico che riporta l'andamento stimato dal modello (4.12) della stima della percentuale di passaggi lunghi riusciti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	66
5.11 Grafico che riporta l'andamento stimato dal modello (4.12) della stima del numero di tocchi fatti nell'area di rigore avversaria per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	66
5.12 Grafico che riporta l'andamento stimato dal modello (4.12) della stima del numero di falli subiti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	67
5.13 Grafico che riporta l'andamento stimato dal modello (4.12) della stima del numero di falli fatti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	68
5.14 Grafico che riporta l'andamento stimato dal modello (4.12) della stima del numero di fuorigioco fatti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	68
5.15 Grafico che riporta l'andamento stimato dal modello (4.12) della stima del numero di cross fatti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	69
5.16 Grafico che riporta l'andamento stimato dal modello (4.12) della stima del numero di contrasti vinti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	69
5.17 Grafico che riporta l'andamento stimato dal modello (4.12) della stima del numero di recuperi per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	70
5.18 Grafico che riporta l'importanza delle covariate rispetto alle norme L2 al variare del parametro di tuning λ secondo le stime del modello (4.12). La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	71

5.19 Grafico che riporta l'andamento stimato dal modello (5.2) della stima del possesso della palla per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	74
5.20 Grafico che riporta l'andamento stimato dal modello (5.2) della stima del numero di tiri in porta per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	75
5.21 Grafico che riporta l'andamento stimato dal modello (5.2) della stima della percentuale di passaggi corti riusciti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	76
5.22 Grafico che riporta l'andamento stimato dal modello (5.2) della stima della percentuale di passaggi medi riusciti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	76
5.23 Grafico che riporta l'andamento stimato dal modello (5.2) della stima della percentuale di passaggi lunghi riusciti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	77
5.24 Grafico che riporta l'andamento stimato dal modello (5.2) della stima del numero di tocchi fatti nell'area di rigore avversaria per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	77
5.25 Grafico che riporta l'andamento stimato dal modello (5.2) della stima del numero di falli fatti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	78
5.26 Grafico che riporta l'andamento stimato dal modello (5.2) della stima del numero di falli subiti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	79
5.27 Grafico che riporta l'andamento stimato dal modello (5.2) della stima del numero di fuorigioco fatti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	79
5.28 Grafico che riporta l'andamento stimato dal modello (5.2) della stima del numero di cross fatti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	80
5.29 Grafico che riporta l'andamento stimato dal modello (5.2) della stima del numero di contrasti vinti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	80
5.30 Grafico dell'andamento delle prestazioni del modello (5.2) su tutti i trenta valori assunti dal parametro di tuning, indicato con il simbolo λ , durante l'applicazione di K-Fold Cross Validation. L'andamento viene valutato in termini di Ranked Probability Score (RPS). La linea rossa tratteggiata indica il parametro di tuning λ ottimo da utilizzare.	81

5.31 Grafico a effetto stella che riporta il contributo medio di una covariata sull'abilità di una singola squadra secondo il modello (5.2).	82
5.32 Grafico a effetto stella che riporta il contributo medio di una covariata sull'abilità di una singola squadra secondo il modello (5.2).	83
5.33 Grafico che riporta il contributo medio di una covariata sull'abilità di una singola squadra secondo il modello 5.2.	84
5.34 Grafico che riporta l'importanza delle covariate rispetto alle norme L2 al variare del parametro di tuning λ	85
5.35 La prima tabella indica le predizioni di 80 partite fatte dal modello (4.1), la seconda dal modello (5.1), la terza dal modello (4.12), la quarta dal modello (5.2) e la quinta dai <i>bookmakers</i> . La classe 1 indica la vittoria della squadra in casa, la classe 2 indica il pareggio tra le due squadre, la classe 3 indica la vittoria della squadra ospite. Con True si indicano le classificazioni effettive mentre con Predicted le classificazioni effettuate.	86
5.36 La prima tabella indica le sensibilità delle predizioni del modello (4.1), la seconda del modello (5.1), la terza del modello (4.12), la quarta del modello (5.2) e la quinta dei <i>bookmakers</i> . La classe 1 indica la vittoria della squadra in casa, la classe 2 indica il pareggio tra le due squadre, la classe 3 indica la vittoria della squadra ospite.	87
5.37 La prima tabella indica la precisione delle predizioni del modello (4.1), la seconda del modello (5.1), la terza del modello (4.12), la quarta del modello (5.2) e la quinta dei <i>bookmakers</i> . La classe 1 indica la vittoria della squadra in casa, la classe 2 indica il pareggio tra le due squadre, la classe 3 indica la vittoria della squadra ospite.	87
5.38 La prima tabella indica le specificità delle predizioni del modello (4.1), la seconda del modello (5.1), la terza del modello (4.12), la quarta del modello (5.2) e la quinta dei <i>bookmakers</i> . La classe 1 indica la vittoria della squadra in casa, la classe 2 indica il pareggio tra le due squadre, la classe 3 indica la vittoria della squadra ospite.	88
6.1 Esempio grafico della funzione kernel γ che mappa i punti di uno spazio d'input in uno <i>feature space</i> di dimensione maggiore e linearmente separabile rispetto a quello originale.	90
6.2 Esempio grafico della procedura di Bootstrap. Da un <i>dataset</i> di dati iniziale vengono estratti, con reinserimento, gli elementi del <i>dataset</i> per formare k campioni avventi tutti lo stesso numero di elementi.	91
6.3 Esempio grafico della procedura di Bagging. L'algoritmo inizia generando k campioni avventi tutti lo stesso numero di elementi, attraverso il Bootstrapping, a partire dai dati originali ricevuti in input. Successivamente ogni campione viene assegnato a un solo classificatore per effettuare il <i>training</i> del classificatore. Infine le predizioni dei classificatori istruiti vengono tutte aggregate per formare un'unica predizione finale.	92
6.4 Esempio grafico della procedura di Boosting. L'algoritmo parte da un classificatore debole iniziale il quale viene addestrato su tutti i dati. Successivamente, dopo che il classificatore debole ha effettuato delle predizioni, vengono aggiornati i pesi di quei esempi che sono stati erroneamente classificati. Viene allenato un nuovo modello debole sulla base di quanto fatto dal classificatore precedente, ripetendo la procedura di allenamento e aggiornamento dei pesi.	93

6.5 Esempio grafico dell'algoritmo K-Nearest-Neighbors. L'istanza da classificare è indicata con il punto interrogativo (?). Il primo passo dell'algoritmo è quello di calcolare tutte le distanze. Dopo di che, considera solo i $k = 3$ punti più vicini all'istanza (?). Infine l'algoritmo classifica con la classe B l'istanza (?).	94
6.6 Esempio grafico dell'algoritmo Support Vector Machine. I punti sulle linee tratteggiate indicano i vettori di supporto, mentre la retta al centro indica l'iperpiano ottimo di separazione.	95
6.7 Esempio grafico dell'algoritmo Random Forest. L'algoritmo partendo dai dati iniziali, tramite il Bootstrap, crea una serie di campioni di esempi di training, in cui ogni campione ha un sottoinsieme di attributi. Ogni campione viene utilizzato per addestrare un'albero di decisione. Una volta che sono stati costruiti gli alberi, viene fatta la predizione per ogni modello e, attraverso un algoritmo di mediazione, ad esempio il <i>voting</i> , viene prodotta la predizione finale.	97
6.8 Esempio grafico dell'algoritmo AdaBoost. Quest'algoritmo utilizza la tecnica di Boosting per creare un modello di classificazione accurato a partire da un insieme di classificatori deboli. Ad ogni iterazione, i pesi assegnati alle osservazioni vengono modificati dando maggior peso alle istanze che sono state classificate erroneamente per procedere con l'addestramento di un nuovo classificatore debole.	98
 7.1 Grafico dell'andamento della media dell'accuratezza per ogni valore dell'iperparametro <code>n_neighbors</code> e per ogni tipo di metrica di distanza utilizzata durante l'applicazione della Cross Validation con 10 fold per il modello K-Nearest-Neighbors. Ogni punto è un classificatore con un certo numero di vicini. La linea blu indica l'andamento con la distanza di Manhattan, mentre la linea arancione l'andamento con la distanza euclidea.	100
7.2 Tabella di confusione del modello K-Nearest-Neighbors con <code>n_neighbors</code> = 30 e <code>p</code> = 2. La classe 0.0 indica la vittoria della squadra in casa, la classe 1.0 indica il pareggio tra le due squadre, la classe 2.0 indica la vittoria della squadra ospite.	101
7.3 Grafico delle misurazione registrate durante la fase di predizione del modello K-Nearest-Neighbors con <code>n_neighbors</code> = 30 e <code>p</code> = 2.	102
7.4 Grafico dell'andamento della media dell'accuratezza per ogni valore dell'iperparametro C e per ogni funzione kernel utilizzata durante l'applicazione della Cross Validation con 10 fold per il modello Support Vector Machine. Ogni punto è un classificatore con un certo valore di C. La linea blu indica l'andamento con la funzione linear kernel mentre la linea verde l'andamento con la funzione RBF e la linea arancione l'andamento con la funzione polynomial kernel.	103
7.5 Tabella di confusione del modello Support Vector Machine con C = 1.4 e <code>kernel</code> = linear. La classe 0.0 indica la vittoria della squadra in casa, la classe 1.0 indica il pareggio tra le due squadre, la classe 2.0 indica la vittoria della squadra ospite.	103
7.6 Grafico delle misurazione registrate durante la fase di predizione del modello Support Vector Machine con C = 1.4 e <code>kernel</code> = linear.	104

- 7.7 Il grafico in alto a sinistra indica l'andamento della media dell'accuratezza per ogni valore dell'iperparametro `criterion`. Il grafico in alto a destra indica l'andamento della media dell'accuratezza per ogni valore dell'iperparametro `min_samples_split`. Il grafico in basso indica l'andamento della media dell'accuratezza per ogni valore dell'iperparametro `max_depth`. Entrambi i grafici sono l'applicazione della Cross Validation con 10 fold per il modello Decision Tree. 105
- 7.8 Tabella di confusione del modello Decision Tree con `max_depth` = 3, `min_samples_split` = 3 e `criterion` = entropy. La classe 0.0 indica la vittoria della squadra in casa, la classe 1.0 indica il pareggio tra le due squadre, la classe 2.0 indica la vittoria della squadra ospite. 106
- 7.9 Grafico delle misurazioni registrate durante la fase di predizione del modello Decision Tree con `max_depth` = 3, `min_samples_split` = 3 e `criterion` = entropy. 106
- 7.10 L'albero di decisione del modello Decision Tree con `max_depth` = 3 e `criterion` = entropy. Per ogni nodo non foglia c'è un test che indica quale ramo scegliere asseconda del valore contenuto nell'attributo testato. Il parametro `entropy` indica l'entropia misurata. Il parametro `samples` indica il numero di istanze che soddisfano i test precedenti. Nel parametro `value` vengono riportati il numero di istanze presenti per ognuna delle tre classi. Il parametro `class` indica la classe di maggioranza nel nodo. Il colore indica la classe di maggioranza con una tonalità differente asseconda della frequenza della classe di maggioranza. 107
- 7.11 Il grafico in alto a sinistra indica l'andamento della media dell'accuratezza per ogni valore dell'iperparametro `criterion`. Il grafico in alto a destra indica l'andamento della media dell'accuratezza per ogni valore dell'iperparametro `min_samples_split`. Il grafico al centro indica l'andamento della media dell'accuratezza per ogni valore dell'iperparametro `max_depth`. Il grafico in basso indica l'andamento della media dell'accuratezza per ogni valore dell'iperparametro `n_estimators`. Tutti i grafici sono l'applicazione della Cross Validation con 10 fold per il modello Random Forest. 109
- 7.12 Tabella di confusione del modello Random Forest con `criterion` = gini, `max_depth` = 5, `n_estimators` = 102 e `min_samples_split` = 9. La classe 0.0 indica la vittoria della squadra in casa, la classe 1.0 indica il pareggio tra le due squadre, la classe 2.0 indica la vittoria della squadra ospite. 110
- 7.13 Grafico delle misurazioni registrate durante la fase di predizione del modello Random Forest con `criterion` = gini, `max_depth` = 5, `n_estimators` = 102 e `min_samples_split` = 9. 110
- 7.14 Il grafico riporta per ogni *feature* la sua importanza basata sull'impurità. 111
- 7.15 Grafico dell'andamento della media dell'accuratezza per ogni valore dell'iperparametro `n_estimators` e per ogni valore dell'iperparametro `learning_rate` utilizzati durante l'applicazione della Cross Validation con 10 fold per il modello AdaBoost. Ogni punto utilizza un certo numero classificatori mentre il colore indica il peso dell'*learning rate*. 112
- 7.16 Tabella di confusione del modello AdaBoost con `n_estimators` = 19 e `learning_rate` = 0.5. La classe 0.0 indica la vittoria della squadra in casa, la classe 1.0 indica il pareggio tra le due squadre, la classe 2.0 indica la vittoria della squadra ospite. 113

7.17 Grafico delle misurazione registrate durante la fase di predizione del modello AdaBoost con <code>n_estimators</code> = 19 e <code>learning_rate</code> = 0.5.	113
8.1 Barplot che indica per ogni squadra l'abilità stimata dal modello (4.12) con $Y = 5$. Viene indicato con un asterisco le squadre con un piazzamento stimato diverso da quello reale anche esso riportato a destra del grafico.	116
8.2 Grafico che riporta l'andamento stimato dal modello (4.12) con $Y = 5$ della stima del numero di tiri in porta per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	119
8.3 Grafico che riporta l'andamento stimato dal modello (4.12) con $Y = 5$ della stima del numero di passaggi corti tentati per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	120
8.4 Grafico che riporta l'andamento stimato dal modello (4.12) con $Y = 5$ della stima della percentuale di passaggi corti riusciti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	120
8.5 Grafico che riporta l'andamento stimato dal modello (4.12) con $Y = 5$ della stima della percentuale di passaggi medi riusciti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	121
8.6 Grafico che riporta l'andamento stimato dal modello (4.12) con $Y = 5$ della stima del numero di falli subiti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	122
8.7 Grafico che riporta l'andamento stimato dal modello (4.12) con $Y = 5$ della stima del numero di cross fatti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	123
8.8 Grafico che riporta l'andamento stimato dal modello (4.12) con $Y = 5$ della stima del numero di recuperi per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.	123
8.9 Grafico dell'andamento delle prestazioni del modello (4.12) con $Y = 5$, su tutti i trenta valori assunti dal parametro di tuning, indicato con il simbolo λ , durante l'applicazione di K-Fold Cross Validation. L'andamento viene valutato in termini di Ranked Probability Score (RPS). La linea rossa tratteggiata indica il parametro di tuning λ ottimo da utilizzare.	124
8.10 Grafico che riporta l'importanza delle covariate rispetto alle norme L2 al variare del parametro di tuning λ	124

8.11 Tabella di confusione che indica le predizioni su 80 partite, fatte dal modello (4.12) con $Y = 5$. La classe 1 indica la vittoria della squadra in casa con due gol di scarto, la classe 2 indica la vittoria della squadra in casa con un gol di scarto, la classe 3 indica il pareggio tra le due squadre, la classe 4 indica la vittoria della squadra ospite con un gol di scarto e la classe 5 indica la vittoria della squadra ospite con due gol di scarto. Con True si indicano le classificazioni effettive mentre con Predicted le predizioni effettuate.	125
8.12 Grafico delle misurazioni riguardanti le predizioni fatte dal modello (4.12) con $Y = 5$ su 80 partite. La classe 1 indica la vittoria della squadra in casa con due gol di scarto, la classe 2 indica la vittoria della squadra in casa con un gol di scarto, la classe 3 indica il pareggio tra le due squadre, la classe 4 indica la vittoria della squadra ospite con un gol di scarto e la classe 5 indica la vittoria della squadra ospite con due gol di scarto. Con precision si indicano le misurazioni della precisione per ogni categoria. Con recall si indicano le misurazioni della sensibilità per ogni categoria. Con specificity si indicano le misurazioni della specificità per ogni categoria.	126
9.1 Grafico a barre in cui viene riportata la Area Under the Curve (AUC) registrata durante la fase di predizione dei algoritmi K-Nearest-Neighbors (K-NN), Support Vector Machine (SVM), Decision Tree, Random Forest e AdaBoost.	129

ELENCO DELLE TABELLE

2.1 La tabella mostra un estratto del <i>dataset</i> utilizzato i cui dati sono stati ricavati da FBref.	7
2.2 La tabella riassuntiva delle variabili presenti nel <i>dataset</i>	17
2.3 Tabella corrispondenza tra nomi originali e nomi nel <i>dataset</i>	18
4.1 Tipi di covariate e possibili parametrizzazioni applicabili al parametro abilità γ	48
5.1 Stime delle covariate con relativo <i>Standard Error</i> (SE), stimate dal modello 5.1.	55
5.2 Stime delle covariate stimate dal modello (4.12).	59
5.3 Stime delle covariate stimate dal modello (4.12).	60
5.4 Stime delle covariate stimate dal modello (5.2).	72
5.5 Stime delle covariate stimate dal modello (5.2).	73
8.1 Stime delle covariate stimate dal modello (4.12) con $Y = 5$	117

8.2 Stime delle covariate stimate dal modello (4.12) con $Y = 5$	118
--	-----

1 | INTRODUZIONE

Il seguente capitolo descrive il problema scientifico affrontato in questa tesi e fornire un'visione della letteratura di riferimento. Inoltre, il capitolo illustra le tecnologie utilizzate nella tesi, le motivazioni personali alla base della sua realizzazione e infine la struttura.

1.1 Dominio del problema

Negli ultimi anni lo sport del calcio è stato oggetto di un grande processo di rinnovamento tecnologico. Infatti, grazie alla spinta dell'evoluzione dei sistemi di comunicazioni, ora da una singola partita è possibile ottenere una grande quantità di informazioni in modo semplice, dato che lo sviluppo delle piattaforme di *streaming* permette di usufruire di contenuti calcistici in ogni momento a livello globale. Questo rinnovamento tecnologico del calcio, con la conseguente disponibilità di dati riguardanti partite e giocatori, è dovuto anche all'introduzione di sensori e sistemi di tracciamento sofisticati all'interno del campo da calcio, come, ad esempio, la *Goal-Line Technology* (vedi **glt**) oppure la Video Assistant Referee (VAR) (vedi **var**). Recentemente, durante la World Cup Qatar 2022, è stato introdotto il fuorigioco semi-automatico (vedi **offside**) composto da una grande quantità di sensori e telecamere per il tracciamento.

Data la disponibilità di una notevole quantità di dati e informazioni relative alle partite e ai giocatori, sono nati nuovi obiettivi di studi. Tra questi, vi è lo *scouting* di giocatori emergenti (**vilela2018towards**) oppure la scelta del ruolo più adatto per il giocatore in base alle sue caratteristiche (**razali2017predicting**) oppure quanto un giocatore è soggetto ad infortuni (**theron2020use**). Oltre a questi, diversi studi si concentrano sull'analisi di informazioni relative alle squadre e alle partite nel loro complesso, ad esempio, il lavoro di (**ley2019ranking**) che utilizza dieci modelli statistici basati sulla forza stimata sulle partite di calcio, analizza il campionato inglese di calcio (Premier League) per produrre un nuovo *ranking*.

Questa tesi si concentrerà sull'analisi delle partite, ossia l'interesse è l'identificazione dei fattori associati all'esito della partita. Infatti, molto spesso ci si pone l'interrogativo se una statistica, quale ad esempio il possesso della palla, o il numero di falli fatti o il numero di tiri fatti, sia rilevante per l'esito della partita, e in caso affermativo, con quale peso sia associata alla vittoria o al pareggio o alla sconfitta. Dunque, l'analisi sarà condotta sulle partite del campionato italiano della Serie A della stagione 2021/2022, utilizzando le statistiche più rilevanti, raccolte durante le partite di calcio. I dati utilizzati sono stati reperiti dal sito web **fbref**, il quale mette a disposizione un'enorme quantità di statistiche riguardanti le partite delle maggiori leghe di calcio di più stagioni.

Fondamentale è la scelta delle metodologie da utilizzare per le analisi. Si è scelto l'utilizzo del modello Bradley-Terry (**bradley1952rank**) per l'interpretazione dei dati, l'individuazione di possibili legami tra le statistiche registrate durante una partita e l'esito della partita e infine, per scopi di predizione. Successivamente sono state utilizzate anche tecniche di *machine learning* per implementare modelli matematici più complessi in grado di ottenere previsioni più accurate.

1.2 Applicazione

bradley1952rank hanno sviluppato un modello statistico utile per i confronti a coppie. L'obiettivo è quello, per ogni confronto, di stabilire quale dei due oggetti confrontati sia il migliore sulla base di tratti latenti non osservati. Un esempio di tratto latente è l'effetto dell'ambiente dove avviene il confronto, che nell'ambito calcistico può essere tradotto come il vantaggio di giocare una partita in casa.

Successivamente il modello Bradley-Terry è stato modificato con diverse estensioni. Un'importante estensione si deve a **davidson1970extending**, il quale ha introdotto il pareggio nel confronto a coppie, elemento fondamentale per la nostra ricerca dato che il modello originario sviluppato da Bradley e Terry è binario (vittoria/sconfitta). In seguito grazie ai lavori di **francis2010** e di **Turner2012Firth** è stata introdotta e approfondita l'inclusione di covariate per la valutazione nei confronti, ovvero l'utilizzo di attributi che descrivono i soggetti che eseguono i confronti tra oggetti. Nel nostro contesto, i soggetti sono rappresentati dalle partite di calcio, mentre gli oggetti sono rappresentati dalle squadre di calcio. Successivi lavori da parte di **thurner2000policy** e di **mauerer2015modeling** introducono le covariate specifiche dell'oggetto e le covariate specifiche del soggetto e dell'oggetto.

Con l'introduzione delle covariate nei modelli di comparazioni a coppie aumentò la complessità dei modelli. Pertanto, sono state proposte in letteratura soluzioni basate sui metodi di regolarizzazione per ridurre la complessità dei modelli. Si veda ad esempio, **schauberger2019btllasso**, in cui svolgono l'analisi delle partite del campionato tedesco (Bundesliga) applicando il metodo LASSO come metodo di regolarizzazione. Nell'ambito delle predizioni degli esiti delle comparazioni attraverso il modello Bradley-Terry si riporta l'interessante lavoro svolto da **kang2015poisson** in cui vengono svolte predizioni sulle partite del videogioco League of Legends (LOL). Per quanto riguarda i modelli predittivi di *machine learning* **xu2021prediction** applica i metodi Decision Tree e Random Forest per la predizione degli esiti delle partite di calcio della Bundesliga.

1.3 Tecnologie e Strumenti utilizzati

Nella seguente sezione saranno illustrate le tecnologie e gli strumenti utilizzati durante il lavoro di tesi.

1.3.1 Tecnologie

Le tecnologie utilizzate in questa tesi sono descritte di seguito.

* **R (R-language)** è un linguaggio di programmazione per il calcolo statistico e l'analisi grafica. È stato sviluppato nel 1993 da Ross Ihaka e Robert Gentleman ed è diventato uno strumento molto popolare per l'analisi dei dati in molti campi, inclusa la scienza dei dati, l'economia, la genetica e la biologia computazionale. R offre un'ampia gamma di funzionalità per il trattamento dei dati, l'analisi statistica e la creazione di grafici e altre rappresentazioni visuali dei dati. Ad oggi è supportato dalla R Core Team e dalla R Foundation for Statistical Computing. È distribuito come software *open source* e può essere facilmente esteso attraverso il *download* di pacchetti di funzionalità aggiuntive sviluppati da una vasta comunità di utenti.

Le librerie utilizzate sono indicate nell'Appendice 11.7.

- * **Python** ([van2003introduction](#)) è un linguaggio di programmazione general-purpose, interpretato e ad alto livello. È stato sviluppato da Guido van Rossum negli anni '90 ed è mantenuto dalla Python Software Foundation. Il linguaggio Python è utilizzato in molti ambiti, come il web development, il *machine learning* e l'automazione. È distribuito come software *open source* e viene fornito con una grande quantità di librerie standard che espandono le sue funzionalità base.
- Le funzioni utilizzate sono indicate nell'Appendice [12.2](#).

1.3.2 Strumenti

Gli strumenti utilizzati in questa tesi sono descritti di seguito.

- * **RStudio** (si veda [rstudio](#)) è un ambiente di sviluppo integrato (IDE) per il linguaggio di programmazione R. Fornisce un insieme di strumenti per facilitare la scrittura, il debugging e il *testing* del codice R. RStudio include anche funzionalità per la visualizzazione e l'analisi dei dati, come il supporto per i grafici interattivi e la possibilità di eseguire il codice R direttamente nell'editor di testo. RStudio è distribuito come software *open source*. In questa tesi è stato utilizzato per implementare il modello Bradley-Terry in R.
- * **PyCharm** (si veda [pycharm](#)) è un ambiente di sviluppo integrato (IDE) per il linguaggio di programmazione Python. Offre una serie di strumenti per facilitare la scrittura, il debugging e il *testing* del codice Python. PyCharm include anche funzionalità per l'integrazione con altri strumenti e servizi comuni nello sviluppo web, come il supporto per il versionamento del codice con Git e il supporto per i *framework* di sviluppo web come Django. PyCharm è sviluppato da JetBrains. In questa tesi è stato utilizzato per implementare i modelli di *machine learning* in Python.

1.4 Motivazioni personali

Durante il mio percorso di studio ho frequentato alcuni corsi legati al mondo dell'intelligenza artificiale e dello studio dei dati. Oltre a ciò, sono un appassionato dello sport del calcio. Nel seguire questo sport, a volte mi imbatto in articoli in cui sono riportate analisi dei dati e l'applicazione di algoritmi di *machine learning* che provano a predire le classifiche finali dei maggiori campionati europei in fase di svolgimento. Inoltre, in seguito alla pandemia da COVID19, sempre più club calcistici hanno iniziato ad analizzare dati e statistiche per migliorare le loro prestazioni in campo e nello *scouting*. Perciò, su spinta delle mie passioni e dalle recenti applicazioni precedentemente descritte, ho individuato nell'identificazione dei fattori associati all'esito di una partita, un campo di ricerca innovativo e interessante come lavoro di tesi di laurea magistrale.

1.5 Struttura della tesi

La struttura della tesi è riportata di seguito.

Il secondo capitolo descrive la raccolta dati e la struttura del dataset.

Il terzo capitolo descrive l'analisi grafica dei dati e il *preprocessing* dei dati.

Il quarto capitolo descrive il modello Bradley-Terry e le sue estensioni che sono state utilizzante durante l'analisi.

Il quinto capitolo illustra i risultati registrati con il modello Bradley-Terry e le sue estensioni. Inoltre vengono riportate le predizioni eseguite dai vari modelli Bradley-Terry confrontate con le predizioni dei *bookmakers*.

Il sesto capitolo descrive gli algoritmi di apprendimento automatico che sono stati utilizzanti durante l'analisi.

Il settimo capitolo riporta le predizioni calcolate dagli algoritmi di apprendimento automatico.

l'ottavo capitolo descrive una nuova applicazione di un modello BT già utilizzato nel Capitolo 5, con una variabile risposta non più a tre ma a cinque categorie.

Il nono capitolo contiene una descrizione sui risultati riportati dai Capitoli 5, 7 e 8.

Il decimo capitolo riporta un riassunto di quanto è stato svolto, sottolineando possibili sviluppi del lavoro di tesi.

2 | SERIE A 2021/2022 DATASET

Nel seguente capitolo verrà descritta la raccolta dati effettuata per costruire il dataset riguardante le partite di calcio della Serie A italiana della stagione 2021/2022 e la struttura di tale dataset.

2.1 Serie A 2021/2022

L'analisi effettuata ha preso in considerazione le partite della Serie A italiana della stagione 2021/2022. La Serie A è un torneo che comprende 20 squadre sparse per tutta l'Italia, alcune anche della stessa città, come ad esempio Milan e Inter per Milano. Tale torneo è organizzato con una struttura *Double-Round-Robin*, dove ogni squadra affronta due volte le altre 19 avversarie del torneo. Vi è quindi una partita di andata e una di ritorno. In base al sorteggio necessario alla creazione del calendario delle partite si decide quale delle due partite sarà giocata in casa oppure fuori casa (in casa dell'avversario).

Il torneo della stagione 2021/2022 è iniziato il 22 Agosto con Inter - Genoa e si è concluso il 22 Maggio con le partite Salernitana - Udinese e Venezia - Cagliari, per un totale 380 partite giocate, suddivise in 38 turni, ciascuno composto da 10 partite.

2.1.1 Ranking

Le squadre di calcio sono classificate in base all'ordine dei punti che hanno totalizzato al termine della stagione. In un torneo calcistico, per ogni partita, la squadra vincitrice guadagna tre punti, la squadra sconfitta guadagna nessun punto, mentre, in caso di pareggio, entrambe le squadre guadagnano un punto. Nel torneo della Serie A chi guadagna più punti vince il campionato, mentre chi si classifica tra le ultime tre retrocede alla lega inferiore, la Serie B. Il posto delle tre squadre retrocesse verrà preso da tre squadre della Serie B che hanno guadagnato la promozione alla Serie A.

La classifica della stagione 2021/2022 è riportata nella Figura 2.1.

2.2 Costruzione del dataset

Al giorno d'oggi, nelle partite di calcio professionistico viene raccolta un'enorme quantità di variabili. Ad esempio, per ogni squadra è noto il tempo in percentuale del possesso della palla e il numero di tiri in porta in una determinata partita. L'obbiettivo principale di questo lavoro è determinare l'influenza che queste variabili hanno sull'esito della partita.

A tale scopo, sono state raccolte un gran numero di variabili che si suppone essere associate all'esito della partita.

Tali dati sono stati offerti dal sito web **fbref**, un sito web dedicato al tracciamento delle statistiche relative ai calciatori e alle squadre di calcio di tutto il mondo.

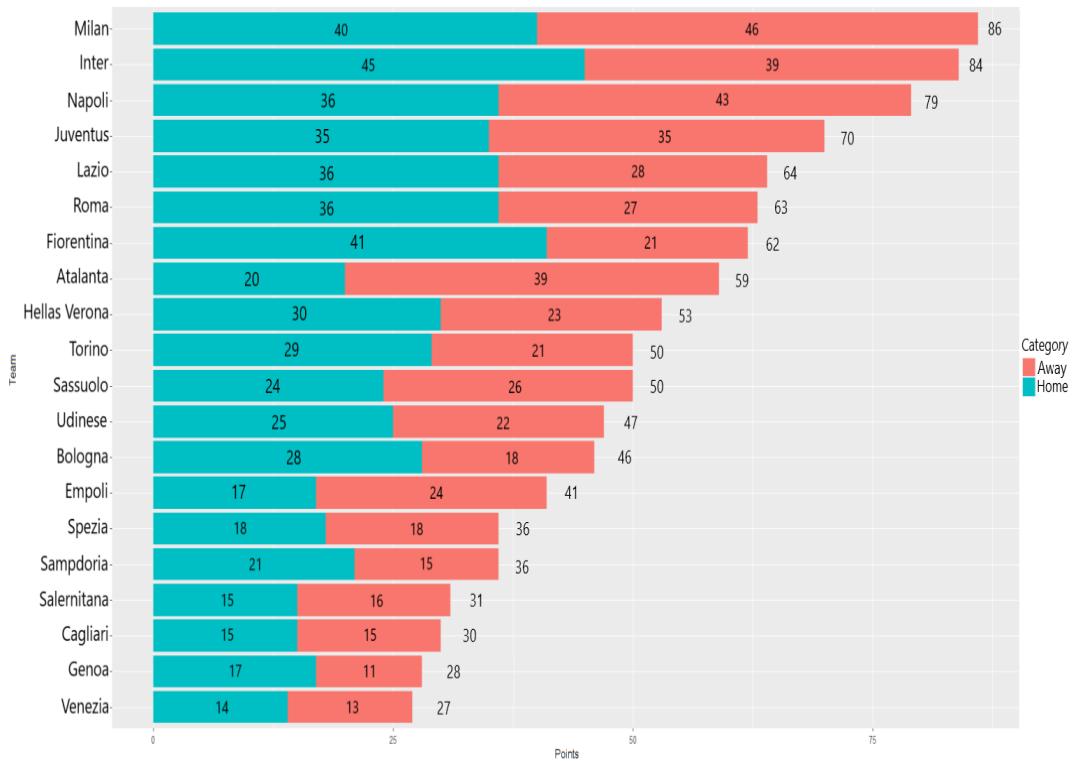


Figura 2.1: Grafico della classifica del campionato italiano della Serie A 2021/2022. Le barre di color azzurro indicano i punti che la squadra ha guadagnato in partite casalinghe. Le barre di color rosa indicano i punti che la squadra ha guadagnato in partite fuori casa.

Il sito web **fbref** mette a disposizione i dati sotto forma di tabelle che possono essere modificate per mantenere solo i dati di nostro interesse.

Dunque, per ogni squadra che ha partecipato alla stagione 2021/2022 di Serie A,



Figura 2.2: Logo di FBref.
Source: <https://fbref.com>

sono state esportate le variabili di interesse per ogni partita giocata, selezionando le macro-aree opportune e adattando le tabelle per ottenere solo i dati utili. Le varie tabelle hanno composto un file Excel divenuto il *dataset* per le analisi svolte nelle tesi.

2.3 Struttura del dataset

Il *dataset* risultante dalla raccolta dati è composto da 760 righe e 35 colonne. Ogni riga riguarda una specifica partita di calcio giocata dalla squadra indicata nella colonna **Team** contro la squadra indicata nella colonna **Vs**. Ogni riga contiene informazioni riguardanti solo la squadra indicata in **Team** fatta eccezione per la data della partita (**Date**), il turno (**Round**) e gli spettatori (**Spec**). Quindi, per ogni partita esistono due righe, una per ciascuna squadra coinvolta. Come risultato finale, ogni squadra appare nella colonna **Team** 38 volte e, siccome il numero totale di squadre è 20, si ottengono 760 righe. La Tabella 2.1 mostra un breve estratto dei dati riguardanti le prime tre partite della stagione.

Date	AtHome	Res	GF	GA	Team	Vs	Poss	...
21/08/2021	TRUE	1	4	0	Inter	Genoa	0,59	...
...
22/08/2021	TRUE	1	2	0	Napoli	Venezia	0,56	...
...
23/08/2021	FALSE	1	1	0	Milan	Sampdoria	0,51	...
...
21/08/2021	FALSE	-1	0	4	Genoa	Inter	0,41	...
...
22/08/2021	FALSE	-1	0	2	Venezia	Napoli	0,44	...
...
23/08/2021	1	TRUE	1	0	1	Sampdoria	Milan	0,49
...

Tabella 2.1: La tabella mostra un estratto del *dataset* utilizzato i cui dati sono stati ricavati da FBref.

Come scritto precedentemente all'interno del *dataset* sono presenti 35 colonne. Oltre alle già citate **Date**, **Round** e **Spec** che hanno solo un valore di completezza dei dati, le restanti 32 colonne sono le possibili variabili che possono influenzare l'esito della partita. Le covariate sono state raggruppate nelle seguenti cinque macro-aree:

- * dati generali,
- * dati relativi ai tiri,
- * dati possesso,
- * dati passaggi,

* dati difensivi,
che sono illustrate di seguito.

2.3.1 Dati generali

In questo gruppo sono presenti le variabili legate a statistiche che non fanno parte di una precisa macroarea ma che descrivono più genericamente la partita giocata.
Le possibili covariate sono le seguenti:

- * **AtHome**: indica se la squadra specificata della variabile **Team** gioca nel proprio stadio, quindi in casa oppure fuori casa. Per indicare se la squadra gioca in casa viene messo come valore TRUE altrimenti FALSE.
Come mostrato dalla Figura 2.1, la quale indica quanti punti sono stati vinti in casa per ogni squadra, ci sono 11 squadre che hanno avuto un leggero vantaggio nel giocare in casa le partite di calcio rispetto a altre sei squadre che hanno avuto l'effetto opposto, mentre le rimanenti tre hanno avuto un effetto nullo.
- * **Res**: indica se la squadra specificata della variabile **Team** ha vinto, pareggiato o perso la partita. Per indicare se ha vinto viene inserito il valore 1, se ha pareggiato 0, altrimenti se ha perso -1. **Res** sarà la variabile risposta.
- * **GF**: indica il numero di gol fatti dalla squadra specificata della variabile **Team**. È stata inserita perché può permettere di valutare la qualità della fase offensiva della squadra e quindi ci si aspetta che possa essere utile ai fini dell'analisi.
- * **GA**: Indica il numero di gol subiti dalla squadra specificata della variabile **Team** e quindi fatti dalla squadra indicata nella variabile **Vs**.
Essa può essere utile perché subire pochi gol incide positivamente sull'esito della partita, limitando l'esposizione della squadra ad uno sbilanciamento in attacco per recuperare lo svantaggio e quindi rischiando maggiormente di subire ulteriori gol dagli avversari. Inoltre, è un fatto riconosciuto che aver la miglior difesa del campionato è associato ad una maggiore probabilità di vittoria del campionato (vedi **catenaccio**).
- * **Team**: indica il nome della squadra a cui i dati della riga fanno riferimento.
- * **Vs**: indica il nome della squadra avversaria.
- * **Fld**: indica il numero di falli fatti dai giocatori della squadra specificata della variabile **Team**.
Questa variabile è stata inserita per capire se una squadra adotta un gioco più fisico/tattico. In questo caso sarà più propensa a interrompere il gioco della squadra avversaria e a commettere più falli. Si vuole perciò capire come questa variabile possa essere associata all'esito della partita, ricordando però che una squadra che commette molti falli è più soggetta a ricevere cartellini gialli o rossi che condizionano la prestazione dei giocatori.
- * **Fls**: indica il numero di falli subiti dai giocatori della squadra specificata della variabile **Team** da parte della squadra avversaria specificata della variabile **Vs**.
Si è deciso di inserire questa covariata perché un alto numero di falli può portare a molte interruzioni della manovra di gioco e quindi permettere alla squadra avversaria di riorganizzarsi.

- * Off: indica il numero di volte che la squadra specificata della variabile Team è finita in fuorigioco. Un calciatore si trova in posizione di fuorigioco quando una qualsiasi parte del suo corpo, fatta eccezione per braccia e mani, si trova nella metà campo avversaria ed è più vicina alla linea di porta avversaria, sia rispetto al pallone che rispetto al penultimo giocatore difendente avversario, portiere compreso nel caso in cui un compagno di questi è più vicino alla linea di porta. Una rappresentazione grafica del fuorigioco è mostrata nella Figura 2.3.

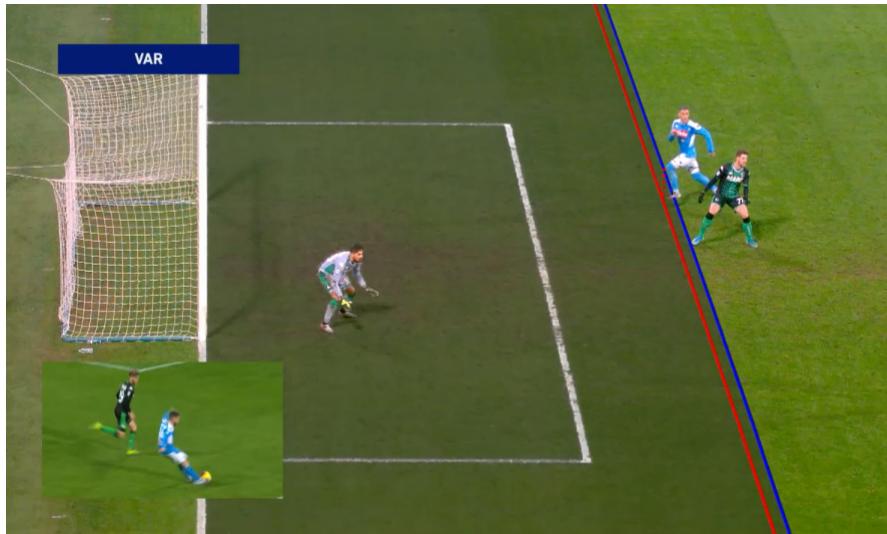


Figura 2.3: Rappresentazione del fuorigioco

Source: <https://sport.sky.it/calcio/2021/10/05/fifa-figc-var-fuorigioco>

Questa variabile è stata inserita perché, se una squadra viene colta molte volte in fuorigioco allora il suo gioco sarà interrotto generando un vantaggio alla squadra avversaria che farà ripartire la sua azione a proprio favore.

2.3.2 Dati relativi ai tiri

In questo gruppo sono presenti le variabili collegate alla fase offensiva della squadra in esame.

- * Sh: indica il numero di tiri totali fatti dalla squadra specificata della variabile Team. Quindi vengono conteggiati il numero di tiri in porta più i tiri fuori dalla porta.
Una squadra che effettua tanti tiri ha più probabilità di segnare un gol. Occorre però capire quanto è precisa una squadra nel centrare la porta.
- * SoT: Indica il numero di tiri in porta totali fatti dalla squadra specificata della variabile Team.
Una squadra con un alto valore di tiri in porta è più probabile che possa segnare un gol. SoT permette di capire quanto è precisa in combinazione con Sh la squadra di calcio nel centrare la porta.
- * G/Sh: indica la proporzione tra gol e tiri fatti dalla squadra specificata della variabile Team.

Questo può permettere di capire quanto la produzione di tiri della squadra è efficace o meno. Con Sh e SoT si riesce a valutare quanto sia offensiva la squadra, cioè se essa gioca costantemente in attacco o utilizza la tattica "difesa e contropiede". Inoltre, permette di capire quanto la squadra sia precisa nell'effettuare i tiri in porta.

2.3.3 Dati relativi al possesso

In questo gruppo sono contenute le variabili collegate al possesso della palla.

- * Poss: indica la quantità di tempo (in percentuale) di possesso palla durante una partita di calcio per la squadra specificata della variabile Team. Nel gioco del calcio, con il termine "possesso palla" si intende un'azione manovrata di due o più giocatori che riescono a passarsi la palla evitando i contrasti degli avversari. Durante la partita, ogni volta che una squadra ha il dominio della palla si dice che questa squadra è in fase di "possesso palla", quindi in questa variabile viene indicato quanto questa fase è durata nell'intera partita.

Il metodo più comune utilizzato per calcolare il possesso palla di una squadra si basa sull'utilizzo di tre cronometri, uno per ciascuna formazione più uno per i tempi morti. Quando un giocatore della squadra A tocca un pallone che prima era in possesso della squadra B, il cronometro della squadra A parte e quello della squadra B si ferma e così via. Il terzo cronometro registra il tempo in tutte le situazioni di palla inattiva, ad esempio, rimesse laterali, calci di punizione ecc... I tempi vengono poi trasformati in percentuali. Per una registrazione più sofisticata, si possono utilizzare ventidue cronometri, uno per ogni giocatore.

La variabile è stata inserita perché, la supremazia nel possesso palla è solitamente desiderabile e utile, dati i seguenti vantaggi:

- spingere l'avversario a muoversi verso la palla per allontanarlo dalla difesa della propria porta per poi sorprenderlo negli spazi lasciati incustoditi.
- modulare il ritmo della gara, ad esempio, se una squadra sta vincendo con un gol di scarto, "congela" il risultato mantenendo il possesso della palla in modo da non ricevere attacchi da parte della squadra avversaria.

Il possesso palla però non garantisce la vittoria. Produrre un possesso palla "sterile", cioè senza che questo porti alla produzione di azioni offensive, può esporre la squadra in possesso della palla a contropiedi nel caso in cui la palla venga persa e quindi all'alto rischio di subire gol perché sbilanciata e non ben posizionata. Vedremo di seguito quali variabili possono essere utili per capire se il possesso palla fatto dalla squadra è "sterile" oppure no.

- * ToDefPen: indica il numero di tocchi fatti dai giocatori della squadra specificata della variabile Team nella propria area di rigore.

Questa variabile è stata inserita perché può essere utile per capire come venga gestito il possesso della palla. Se vi è un alto numero di tocchi, vuol dire che la squadra subisce molto la pressione della squadra avversaria, viceversa cerca di fare un gioco più offensivo. Questa variabile, in combinazione con le variabili ToDef3rd, ToMid3rd, ToAtt3rd e ToAttPen permette di capire se il possesso della palla fatto della squadra sia utile e porti benefici ai fini del risultato oppure sia sterile. Inoltre, si vuole capire in che misura come ToDefPen influenza il risultato della partita con un alto o un basso valore di numero di tocchi nella propria area di rigore, la cui area è indicata nella Figura 2.4.

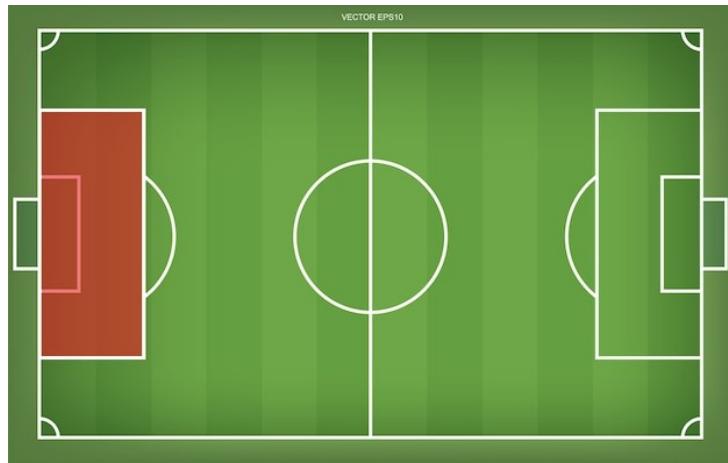


Figura 2.4: In rosso l'area di rigore in un campo da calcio.

Source: <https://it.freepik.com/foto-vettori-gratuito/campo-da-calcio>

- * ToDef3rd: indica il numero di tocchi fatti dai giocatori della squadra specificata della variabile Team nella propria mediana o trequarti difensiva. Questa variabile è stata inserita perché può essere utile per capire come venga gestito il possesso della palla. Se vi è un alto numero di tocchi, vuol dire che la squadra cerca di mantenere il possesso palla creando poche azioni offensive, viceversa cerca di fare un gioco più offensivo. Questa variabile, in combinazione con ToDefPen, ToMid3rd, ToAtt3rd e ToAttPen, permette di capire se il possesso della palla fatto della squadra sia utile e porti benefici ai fini del risultato oppure sia sterile. Inoltre, si vuole capire in che misura ToDef3rd influenza il risultato della partita con un alto o un basso valore di numero di tocchi nella propria mediana la cui area, è indicata nella Figura 2.5.

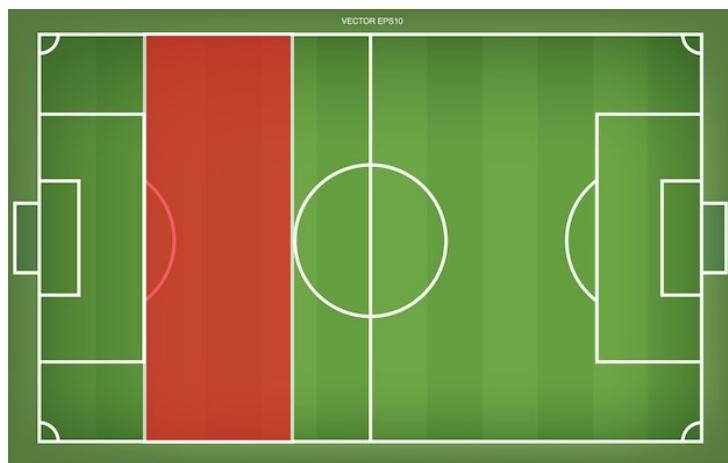


Figura 2.5: In rosso la mediana nel campo da calcio.

Source: <https://it.freepik.com/foto-vettori-gratuito/campo-da-calcio>

- * **ToMid3rd:** indica il numero di tocchi fatti dai giocatori della squadra specificata della variabile **Team** a centrocampo.

Questa variabile è stata inserita perché può essere utile per capire come venga gestito il possesso della palla. Se vi è un alto numero di tocchi, vuol dire che la squadra cerca di mantenere il possesso palla cercando di creare delle azioni offensive, viceversa cerca di fare un gioco più difensivo. Questa variabile, in combinazione con le variabili **ToDefPen**, **ToDef3rd**, **ToAtt3rd** e **ToAttPen**, permette di capire se il possesso della palla fatto dalla squadra sia utile e porti benefici ai fini del risultato oppure sia sterile. Inoltre, si vuole capire in che misura **ToMid3rd** influenza il risultato della partita con un alto o un basso valore di numero di tocchi a centrocampo la cui area, è indicata nella Figura 2.6.

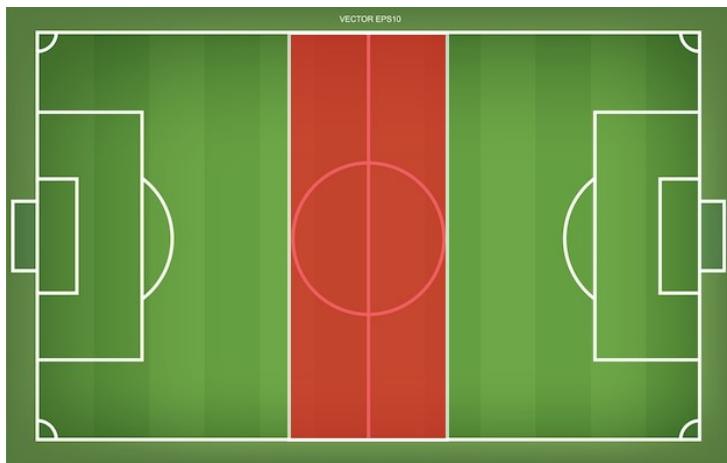


Figura 2.6: In rosso il centrocampo nel campo da calcio.

Source: <https://it.freepik.com/foto-vettori-gratuito/campo-da-calcio>

- * **ToAtt3rd:** indica il numero di tocchi fatti dai giocatori della squadra specificata della variabile **Team** nella trequarti dell'avversario.

Questa variabile è stata inserita perché può essere utile per capire come venga gestito il possesso della palla. Se vi è un alto numero di tocchi, vuol dire che la squadra cerca di mantenere il possesso palla per effettuare una pressione sulla squadra avversaria affinché si possano creare degli spazi per delle azioni offensive, viceversa cerca di fare un gioco molto più difensivo. Questa variabile, in combinazione con le variabili **ToDefPen**, **ToDef3rd**, **ToMid3rd** e **ToAttPen**, permette di capire se il possesso della palla fatto dalla squadra sia utile e porti benefici ai fini del risultato oppure sia sterile. Inoltre, si vuole capire in che misura **ToAtt3rd** influenza il risultato della partita con un alto o un basso valore di numero di tocchi nella trequarti dell'avversario la cui area, è indicata nella Figura 2.7.

- * **ToAttPen:** indica il numero di tocchi fatti dai giocatori della squadra specificata della variabile **Team** a nell'area di rigore dell'avversario.

Questa variabile è stata inserita perché può essere utile per capire come venga gestito il possesso della palla. Se vi è un alto numero di tocchi, vuol dire che la squadra cerca di mantenere il possesso palla applicando un'alta pressione sulla squadra avversaria affinché si possano creare molte occasioni da gol in area,

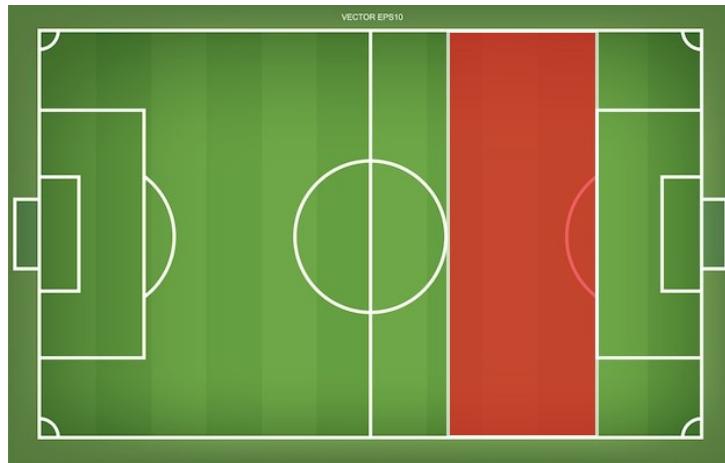


Figura 2.7: In rosso la trequarti dell'avversario nel campo da calcio.

Source: <https://it.freepik.com/foto-vettori-gratuito/campo-da-calcio>

viceversa o la squadra subisce troppo la pressione dell'avversario oppure tende ad avere un gioco molto difensivo. Questa variabile, in combinazione con le variabili ToDefPen, ToDef3rd, ToMid3rd e ToAtt3rd permette di capire se il possesso della palla fatto della squadra sia utile e porti benefici ai fini del risultato oppure sia sterile. Inoltre, si vuole capire in che misura ToAttPen influenza il risultato della partita con un alto o un basso valore di numero di tocchi nell'area di rigore dell'avversario.

- * **ToDist:** Indica la distanza totale, espressa in metri, in cui un giocatore della squadra specificata della variabile **Team** si è mosso con la palla in qualsiasi direzione, controllandola con i piedi.
Questa variabile è stata inserita perché permette di comprendere se il possesso della palla sia stato statico, ovvero i giocatori si sono mossi poco senza avanzare, oppure no. Sarà di interesse analizzare se un alto valore di metri percorsi con palla al piede possa essere utile ad ottenere la vittoria.

2.3.4 Dati relativi ai passaggi

In questo gruppo vi sono raggruppate le variabili collegate ai passaggi della palla.

- * **PAtt:** Indica il numero di tutti i passaggi tentati dai giocatori della squadra specificata della variabile **Team**.
Utile a capire quanto la squadra sia incline a tentare i passaggi.
- * **PCmp%:** Indica la percentuale di passaggi riusciti ai giocatori della squadra specificata della variabile **Team**.
È stata inserita perché permette di capire quanti passaggi siano andati a buon fine tra tutti quelli tentati e quindi ricavare la precisione dei giocatori della squadra.
- * **SPAtt:** Indica il numero di passaggi corti tentati dai giocatori della squadra specificata della variabile **Team**. Per passaggi corti si intendono tutti quelli

effettuati all'interno di una lunghezza tra i tre e quattordici metri.

È stata inserita per capire se un alto numero di passaggi corti possa essere determinanti ai fini dell'esito della partita.

- * **SPCmp%:** Indica la percentuale di passaggi corti riusciti ai giocatori della squadra specificata della variabile **Team**.

È stata inserita perché permette di capire quanti passaggi andati a buon fine tra tutti quelli tentati e quindi ricavare la precisione dei giocatori della squadra.

- * **MPAtt:** Indica il numero di passaggi medi tentati dai giocatori della squadra specificata della variabile **Team**. Per passaggi medi si intendono tutti quelli effettuati all'interno di una lunghezza tra i tredici e ventisette metri. Questi passaggi possono essere considerati come passaggi filtranti, cioè non diretti al proprio compagno di squadra ma verso un'area del campo dove il compagno di squadra deve andare a prendere la palla. Spesso questi passaggi vengono fatti per sorprendere la difesa avversaria ed evitare che la palla venga intercettata. Nella Figura 2.8 viene mostrato l'esecuzione di un passaggio filtrante.

È stata inserita per capire se un alto numero di passaggi medi possa essere

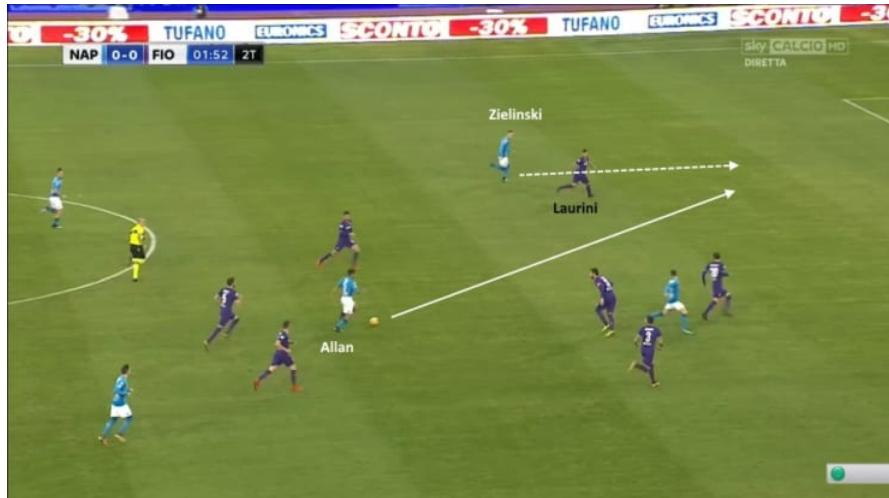


Figura 2.8: Esecuzione di un passaggio filtrante

Source: <https://www.ilmisterone.com/2019/01/16/passaggi-filtranti/>

determinante ai fini dell'esito della partita.

- * **MPCmp%:** Indica la percentuale di passaggi medi riusciti ai giocatori della squadra specificata della variabile **Team**.

È stata inserita perché permette di capire quanti passaggi siano andati a buon fine tra tutti quelli tentati e quindi ricavare la precisione dei giocatori della squadra.

- * **LPAtt:** Indica il numero di passaggi lunghi tentati dai giocatori della squadra specificata della variabile **Team**. Per passaggi lunghi si intendono tutti quelli effettuati all'interno di una lunghezza superiore ai ventisette metri. Questi passaggi possono essere considerati come lanci lunghi per cambi di gioco o per lanciare le punte, cioè i giocatori che giocano come attaccanti, in profondità. Una

rappresentazione di passaggio lungo è mostrata nella Figura 2.9.

È stata inserita per capire se un alto numero di passaggi lunghi possa essere

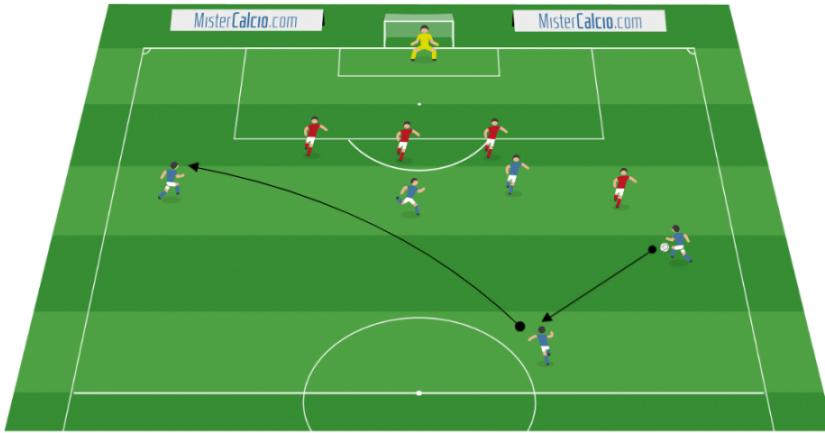


Figura 2.9: Esecuzione di un cambio di gioco

Source: <https://www.mistercalcio.com/tattica/il-cambio-di-gioco/>

determinante ai fini dell'esito della partita.

- * **LPCmp%:** Indica la percentuale di passaggi lunghi riusciti ai giocatori della squadra specificata della variabile **Team**.
È stata inserita perché permette di capire quanti passaggi sono andati a buon fine tra tutti quelli tentati e quindi ricavare la precisione dei giocatori della squadra.
- * **Crs:** Indica il numero di cross effettuati dalla squadra specificata della variabile **Team**. Un cross (in italiano traversone) è un tipo di passaggio medio o lungo, solitamente effettuato sulle fasce laterali dell'area avversaria o comunque vicino all'area avversaria, che permette al compagno di squadra posizionato vicino alla porta avversaria di colpire la palla al volo di testa oppure di piede per segnare un possibile gol. Quindi, se eseguito correttamente, il cross può diventare un assist, cioè l'ultimo passaggio per la realizzazione del gol.

Una rappresentazione di cross è mostrata nella Figura 2.10.

2.3.5 Dati difensivi

In questo gruppo sono contenute le variabili collegate alla fase difensiva.

- * **Saves:** Indica il numero di parate fatte del portiere della squadra specificata della variabile **Team**.
È stata inserita perché permette di valutare se la squadra subisce tanti tiri dagli avversari, così come la qualità del portiere nel salvare la squadra da un possibile gol subito.
- * **Int:** Indica il numero di intercettazioni fatte dai giocatori della squadra specificata della variabile **Team**. Per intercettazione della palla si intende l'intercettazione di un passaggio della squadra avversaria entrando in possesso del pallone andando ad interrompere il passaggio avversario.

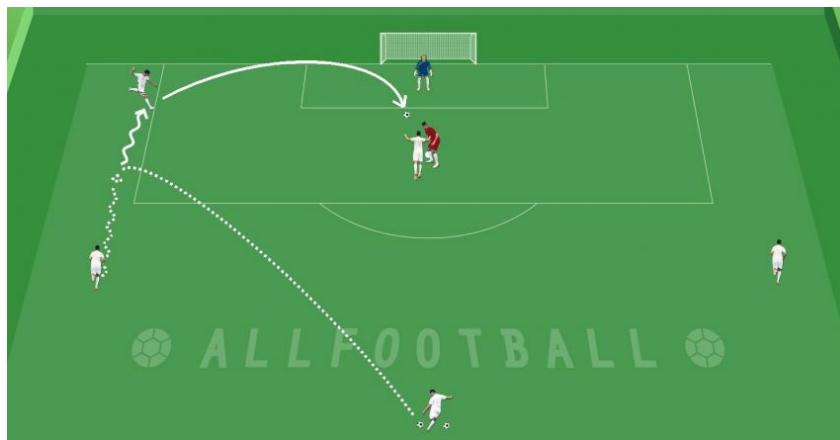


Figura 2.10: Rappresentazione di un cross

Source: <http://www.allfootball.it/blog/calcio-vincere-allenando-i-dettagli/27-2-2017/calcio-la-marcatura-a-uomo-su-cross-laterale/>

- * **TklWin:** Indica il numero di contrasti vinti dai giocatori della squadra specificata della variabile **Team**. Per contrasto si intende il tentativo da parte di un giocatore difendente di sottrarre il possesso della palla all'avversario. Quindi chi ha in possesso la palla viene attaccato da chi ne è privo. Se si riesce a prendere il pallone all'avversario allora si avrà vinto il contrasto. I contrasti vengono effettuati anche per allontanare l'avversario dalle zone pericolose. La Figura 2.11 mostra un contrasto di gioco.
Visto che tale intervento senza palla modifica il gioco dell'avversario, si è deciso di inserire i contrasti vinti come variabile.
- * **Recov:** Indica il numero di palle vaganti recuperate dalla squadra specificata della variabile **Team**. Per palle vaganti si intendono quei palloni che, a seguito di un contrasto di gioco, non sono stati recuperati dalla squadra che ha effettuato il contrasto ma chi ha subito il contrasto, ne ha comunque perso il controllo. Quindi nessuno ha in possesso il pallone e la palla viene detta vagante.
Dato che questa variabile sembra essere legata al possesso del pallone, potrebbe essere interessante per l'analisi.

Nella Tabella 2.2 è riassunto l'insieme delle variabili presenti e le loro macro-aree di appartenenza.

Di seguito nella Tabella 2.3 è mostrato per ogni variabile il nome che ha all'interno del *dataset*.



Figura 2.11: Rappresentazione di un contrasto in scivolata

Source: <https://www.ilmisterone.com/2022/01/24/partita-solo-tackle/>

Statistiche generali	Tiri	Possesso	Passaggi	Difensive
AtHome	Sh	Poss	PAtt	Saves
Res	SoT	ToDefPen	PCmp%	Int
GF	G/Sh	ToDef3rd	SPAtt	TklWin
GA		ToMid3rd	SPCmp%	Recov
Team		ToAtt3rd	MPAtt	
VS		ToAttPen	MPCmp%	
Fls		ToDist	LPAtt	
Fld			LPCmp%	
Off			Crs	

Tabella 2.2: La tabella riassuntiva delle variabili presenti nel *dataset*.

Originale	Rinominate
AtHome	AtHome
Res	Res
GF	GF
GA	GF
Team	Team
VS	Vs
Poss	Poss
Sh	Sh
SoT	SoT
G/Sh	G.Sh
Saves	Saves
PAtt	PAtt
PCmp%	PCmp.
SPAtt	SPAtt
SPCmp%	SPCmp.
MPAtt	MPAtt
MPCmp%	MPCmp.
LPAtt	LPAtt
LPCmp%	LPCmp.
ToDefPen	ToDefPen
ToDef3rd	ToDef3rd
ToMid3rd	ToMid3rd
ToAtt3rd	ToAtt3rd
ToAttPen	ToAttPen
ToDist	ToDist
Fls	Fls
Fld	Fld
Off	Off
Crs	Crs
Int	Int
TklWin	TklWin
Recov	Recov

Tabella 2.3: Tabella corrispondenza tra nomi originali e nomi nel dataset

3 | ANALISI DEI DATI

Nel seguente capitolo verrà illustrata la fase di *preprocessing* e le analisi grafiche dei dati. Le analisi verranno svolte usando il linguaggio di programmazione di (**R-language**).

3.1 Preprocessing dei dati

Dopo aver importato il *dataset*, il primo step da effettuare durante il *preprocessing* è individuare e risolvere possibili anomalie nei dati.

Il dataset non ha valori mancanti. Questo è stato possibile grazie a **fbref** che ha messo a disposizione dati quasi sempre completi; in quei rari casi di mancanza di dati sono stati reperiti manualmente da altre fonti altrettanto attendibili.

Sono state inoltre tolte le variabili **Date** e **Round**.

Il passo successivo è stato controllare che le variabili fossero interpretate correttamente. Infatti, **Team** e **Vs** vengono interpretate erroneamente come tipo **character**. Perciò, **Team** e **Vs** devono essere interpretate come un fattore cioè come un valore non numerico espresso in termini verbali, ad esempio una categoria. Quindi, nel nostro contesto ogni squadra sarà un livello del fattore. Analogamente, **AtHome** è stata trasformata in un fattore a due livelli. Invece, **Res** è stato trasformato in un fattore ordinato con i livelli: -1 = sconfitta < 0 = pareggio < 1 = vittoria, dove la vittoria ha la massima preferenza.

3.2 Analisi grafica dei dati

In questa sezione attraverso il supporto di grafici, si analizzerà graficamente i dati disponibili e le loro relazioni per avere una prima visione dei dati raccolti. Si valuteranno le relazioni tra covariate e la variabile di risposta, le relazioni tra due covariate. Tutto ciò per individuare quali covariate possano essere significative per la variabile risposta e quali interazioni emergono dall'analisi grafica.

Come primo passo, è stata valutata la distribuzione della variabile risposta **Res**, come è mostrato in Figura 3.1. Si può notare come le classi sembrino ben distribuite, dato che abbiamo 196 pareggi e 282 vittorie e altrettante sconfitte. Si ha quindi un campione abbastanza ampio, distribuito e privo di classi povere.

La Figura 3.2 mostra la distribuzione delle vittorie, dei pareggi e delle sconfitte per ogni squadra.

Successivamente si è valutata la distribuzione dei gol ovvero, la frequenza del numero di gol a partita distinguendo i gol per la squadra in casa e per quella ospite. La Figura 3.3 mostra la distribuzione del numero di gol. Il grafico 3.3 riporta che la squadra in casa ha segnato in più partite un gol e due gol rispetto alla squadra ospite. Inoltre, l'evento in cui la squadra in casa non segna nessun gol ha una frequenza nettamente inferiore a quella della squadra ospite. Tuttavia, si nota una certa tendenza delle partite a finire con pochi gol, al massimo due gol per squadra.

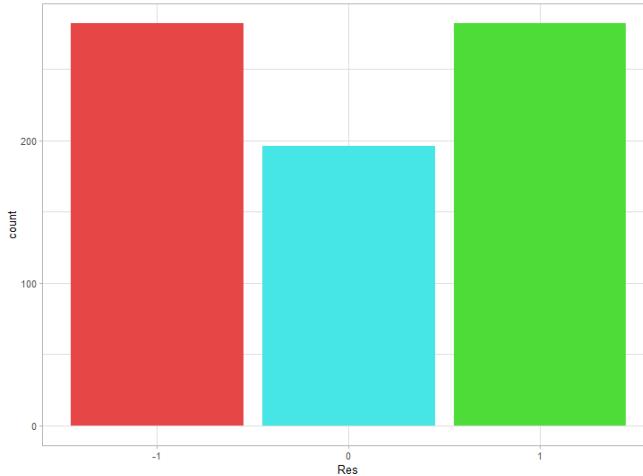


Figura 3.1: Barplot della distribuzione della variabile di risposta Res

3.2.1 Relazione tra la variabile risposta e le covariate

La prima relazione che si analizza riguarda la variabile categorica AtHome. Nella Figura 3.4 viene riportato il mosaicplot tra la variabile risposta Res e AtHome. Tale grafico è un particolare tipo di diagramma a barre impilate che mostra la relazione che c'è tra due fattori. Il numero di colonne è uguale al numero livelli della variabile inserita sull'asse orizzontale. L'altezza delle barre in verticale, invece, è proporzionale al numero di osservazioni della variabile inserita sull'asse verticale per ciascun livello della variabile nell'asse orizzontale. In sostanza, il mosaicplot è una rappresentazione grafica di una tabella di contingenza che permette un confronto visivo tra gruppi. Nella Figura 3.4 c'è una leggera variazione dei risultati tra la squadra che gioca in casa e l'avversaria, infatti per le squadre che giocano in casa, c'è una maggior presenza di vittorie e di minor sconfitte. Naturalmente non c'è alcuna variazione per il pareggio dato che entrambe le squadre lo ottengono.

Nella Figura 3.5 viene riportato il boxplot della distribuzione della variabile Poss rispetto ai valori della variabile risposta Res. Il boxplot è un grafico che consente di visualizzare il centro e la distribuzione dei dati. Inoltre, può essere uno strumento visivo per la verifica della normalità o per l'identificazione di possibili outlier. Dal grafico si nota che Poss sembra essere significativa per l'esito. Infatti, i valori crescono dal boxplot della sconfitta al boxplot della vittoria. C'è una buona distribuzione dei dati perché la lunghezza dei baffi per ogni boxplot è simmetrica. Si segnala che la mediana della sconfitta è più vicina al 3° quantile mentre la mediana della vittoria è più vicina al 1° quantile. Non sono presenti outliers.

Nella Figura 3.6 viene riportato il boxplot della distribuzione della variabile SoT rispetto ai valori della variabile risposta Res. Valori più alti sono presenti nella vittoria mentre valori molto più bassi sono presenti nella sconfitta. C'è una buona distribuzione dei valori nella vittoria dato che i baffi sono simmetrici, viceversa per gli altri due boxplot non c'è simmetria, infatti, il baffo inferiore è molto più corto rispetto al baffo superiore, segno che la maggior parte dei valori sono bassi e simili tra loro. Inoltre, alcuni outliers si discostano dalla distribuzione di tutti e tre i boxplot, questo perché ci sono state squadre che hanno tirato molte volte in porta. Le mediane dei boxplot



Figura 3.2: Barplot della distribuzione della variabile di risposta per squadraRes

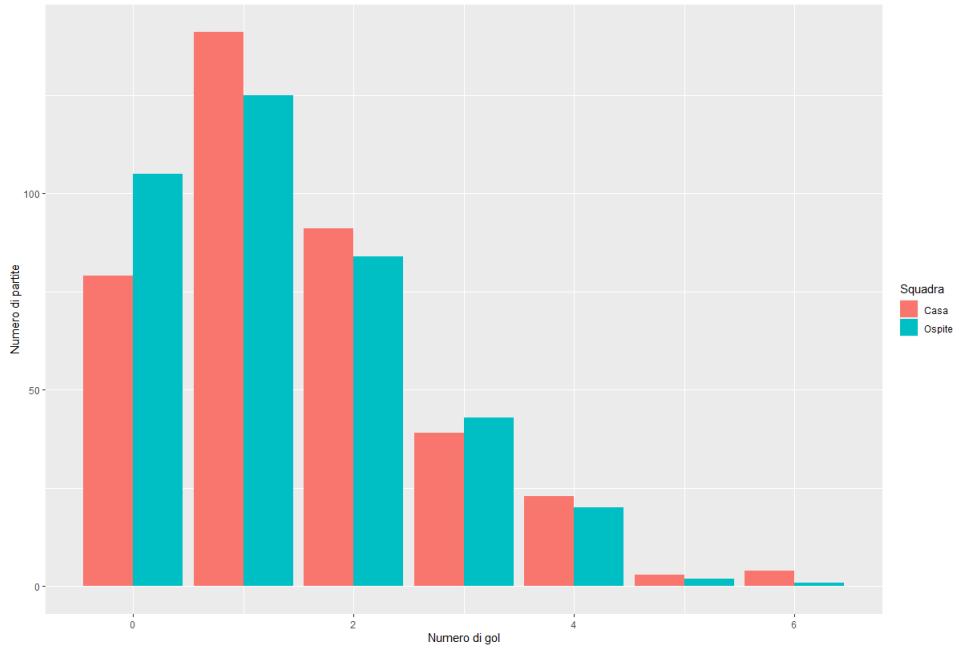


Figura 3.3: Barplot della distribuzione del numero di gol a partita. In azzurro vengono indicati i gol per la squadra in casa mentre in rosa i gol per la squadra ospite

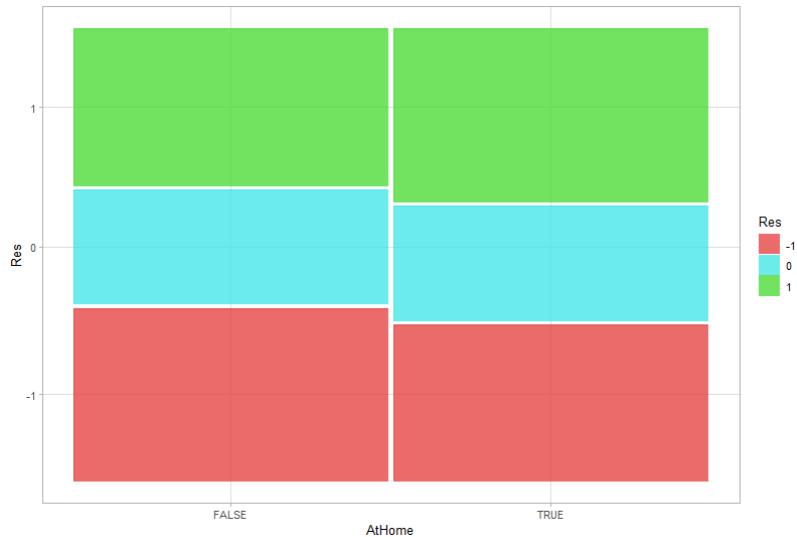


Figura 3.4: Mosaicplot che mostra la distribuzione degli esiti rispetto alle partite giocate in casa e fuori casa

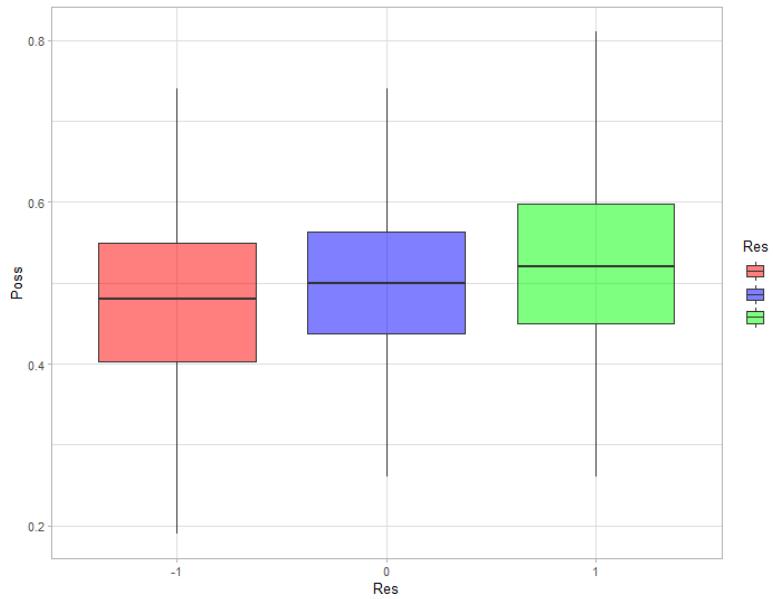


Figura 3.5: Boxplot della distribuzione della variabile **Poss** rispetto ai valori della variabile risposta **Res**

pareggio e vittoria non sono equidistanti dai quantili ma più vicine al 1° quantile. Il boxplot della sconfitta ha una bassa varianza. In conclusione, avere un valore alto di tiri in porta sembra essere utile ai fini della vittoria.

Per la relazione tra la variabile risposta **Res** e la variabile **Sh**, si ha un boxplot molto simile al boxplot mostrato nella Figura 3.5. Il grafico di **Sh** rispetto al grafico di **Poss** ha degli outliers e la mediana della sconfitta non è equidistante dai quantili ma più vicina al 1° quantile.

Nella Figura 3.7 viene riportato il boxplot della distribuzione della variabile **G/Sh** rispetto ai valori della variabile risposta **Res**. Si nota che ci sono valori molto bassi ma leggermente più alti per la vittoria. La distribuzione non è buona perché i baffi sono asimmetrici, infatti, tutti i valori sono concentrati in basso e pochi verso il baffo superiore, segno che la maggior parte dei valori sono bassi e simili tra loro. C'è una bassa varianza tra i valori e la presenza di outliers perché alcune squadre sono riuscite a ottenere il massimo da ogni tiro. I risultati mostrati, nonostante la distribuzione, sono comunque coerenti dato che non ci si aspetta dal rapporto tiri-gol un numero alto ma comunque una tendenza che favorisca la vittoria.

Nella Figura 3.8 viene riportato il boxplot della distribuzione della variabile **Saves** rispetto ai valori della variabile risposta **Res**. Come si può notare sembra che **Saves** sia poco significativa ai fini del risultato. Infatti, c'è poca variazione tra un boxplot e l'altro perché sembra che avere un alto numero di parate non è determinante a fini del risultato.

La Figura 3.9 viene riportato a sinistra il boxplot della variabile numerica **PAtt** rispetto ai valori della variabile risposta **Res** e a destra il boxplot della variabile numerica **PCmp%** rispetto ai valori della variabile risposta **Res**. Per entrambi sembra significativo l'elevato numero di passaggi tentati ma soprattutto quelli completati ai fini della vittoria. Nel grafico a sinistra, nel secondo e terzo boxplot il baffo superiore è

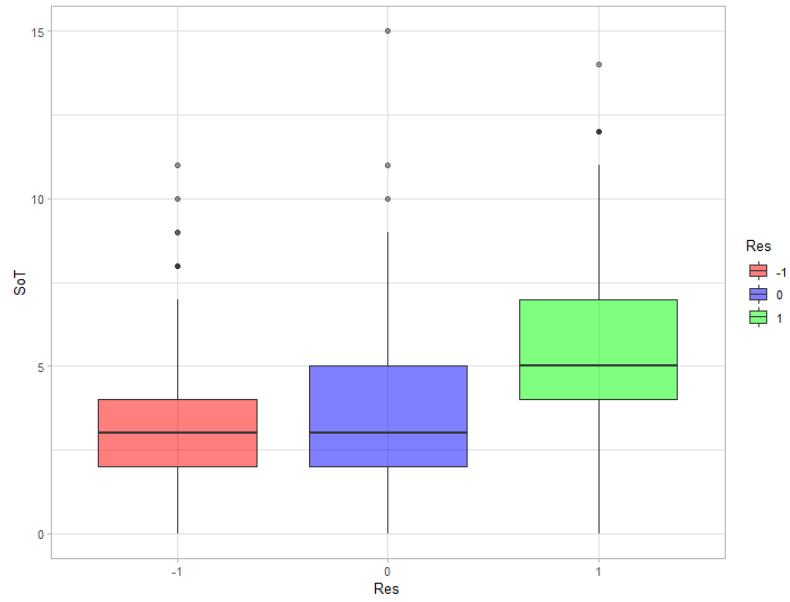


Figura 3.6: Boxplot della distribuzione della variabile SoT rispetto ai valori della variabile risposta Res

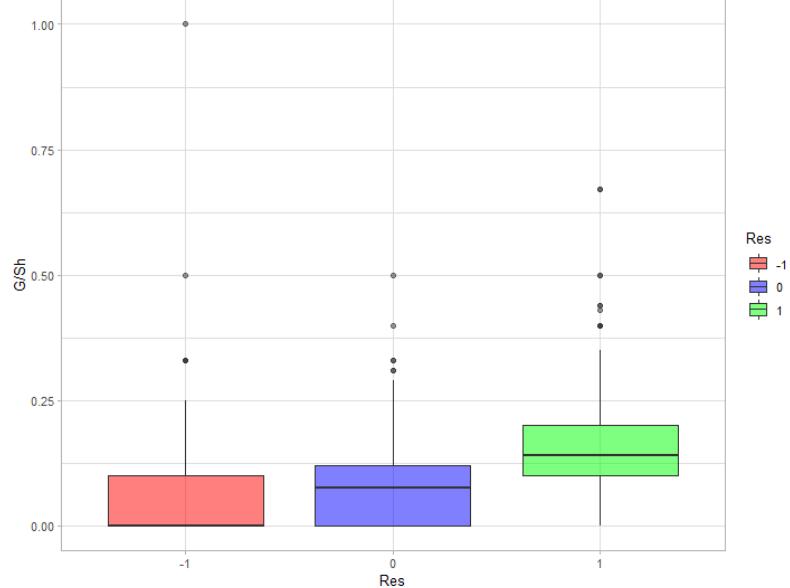


Figura 3.7: Boxplot della distribuzione della variabile G/Sh rispetto ai valori della variabile risposta Res

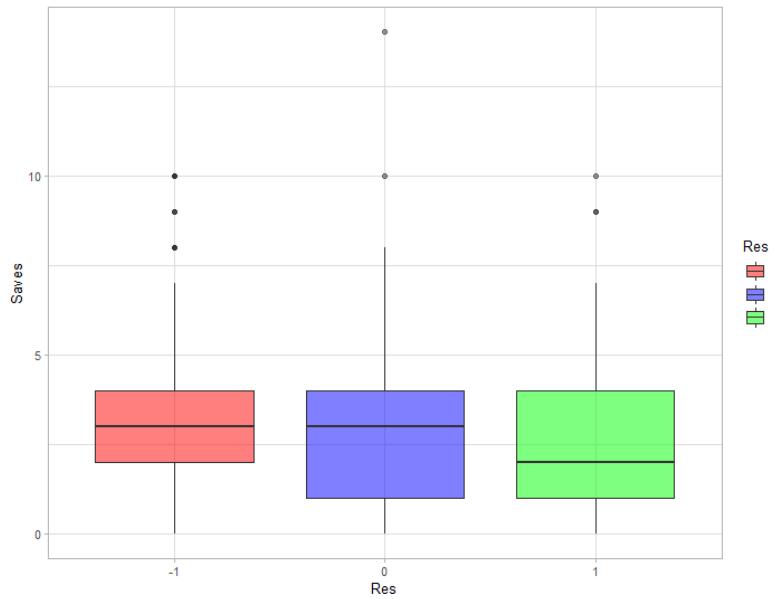


Figura 3.8: Boxplot della distribuzione della variabile **Saves** rispetto ai valori della variabile risposta **Res**

più lungo rispetto al baffo inferiore, segno che molti valori sono bassi e simili tra loro, viceversa il primo boxplot ha una buona distribuzione perché i baffi sono simmetrici. Il boxplot della vittoria ha una maggiore varianza rispetto agli altri due e in più ha valori più alti; sia la mediana del boxplot della vittoria e sia quella del pareggio sono più vicine al 1° quantile, viceversa quella della sconfitta. I dati nel primo boxplot sembrano essere coerenti con l'esito della partita perché, maggiore è numero di passaggi che si prova ad effettuare, maggiori sono le possibilità di vittoria. Occorre però sapere quanto è precisa la squadra e questo lo si può scoprire con la variabile **PCmp%**. Nel grafico a destra, si notano valori alti e molti outliers con valori bassi dovuti al fatto che ci sono state partite dove alcune squadre sono state poco precise nei passaggi. I baffi superiori di tutti e tre i boxplot sono molto meno lunghi rispetto ai baffi inferiori segno che molti valori sono alti e simili tra loro, inoltre, le varianze dei box sembrano essere uguali tra di loro. Sorprendentemente l'andamento invece di essere sempre crescente, prima scende da sconfitta a pareggio e poi sale da pareggio a vittoria.

Per la relazione tra la variabile risposta e la variabile **SPAtt**, si ha un grafico molto simile al grafico a sinistra della Figura 3.9. Il grafico di **SPAtt** rispetto al grafico di **PAtt** ha un maggior numero di outliers soprattutto per la sconfitta rispetto al grafico **PAtt** inoltre, c'è una minor varianza per tutti i tre boxplot oltre a valori più bassi in generale, questo è naturale perché **PAtt** contiene tutti i passaggi tentati e non solo quelli corti.

Per la relazione tra la variabile risposta e la variabile **SPCmp%**, si ha un grafico molto simile al grafico a destra della Figura 3.9. Nel grafico di **SPCmp%** rispetto al grafico di **PCmp%**, il boxplot della sconfitta ha una maggior varianza, viceversa per la vittoria, che ha una minor varianza.

Per la relazione tra la variabile risposta **Res** e la variabile **MPAtt**, si ha un grafico molto simile al grafico a sinistra della Figura 3.9. Nel grafico di **MPAtt** rispetto al grafico

di PAtt, il boxplot della sconfitta ha una maggior varianza. In generale i valori sono più bassi rispetto al grafico di PAtt ma questo è naturale perché PAtt contiene tutti i passaggi tentati e non solo quelli medi.

Per la relazione tra la variabile risposta e la variabile MPCmp%, si ha un grafico molto simile al grafico a destra della Figura 3.9. Il grafico di MPCmp% rispetto al grafico di PCmp% ha valori più alti e molti più outliers inoltre, i baffi inferiore dei boxplot della sconfitta e della vittoria sono più corti.

Per la relazione tra la variabile risposta e la variabile LPAtt, si ha un grafico molto simile al grafico a sinistra della Figura 3.9. Nel grafico di LPAtt rispetto al grafico di PAtt, il boxplot della sconfitta ha valori più bassi rispetto agli boxplot del pareggio e della vittoria inoltre, il boxplot del pareggio ha una maggior varianza dei valori mentre il boxplot della vittoria ha una minor varianza.

In generale i valori sono più bassi rispetto al grafico di PAtt ma questo è naturale perché PAtt contiene tutti i passaggi tentati e non solo quelli lunghi.

Per la relazione tra la variabile risposta e la variabile LPCmp%, si ha un grafico molto simile al grafico a destra della Figura 3.9. Il grafico di LPCmp% rispetto al grafico di PCmp% ha valori più bassi, la distribuzione dei valori per il boxplot della sconfitta è ben equilibrata perché i baffi sono della stessa lunghezza e in più la mediana è equidistante dai due quantili. Analogamente anche il boxplot del pareggio ha una distribuzione equilibrata ma con più varianza e con una mediana equidistante dai quantili.

Nella Figura 3.10 viene riportato il boxplot della distribuzione della variabile ToDefPen

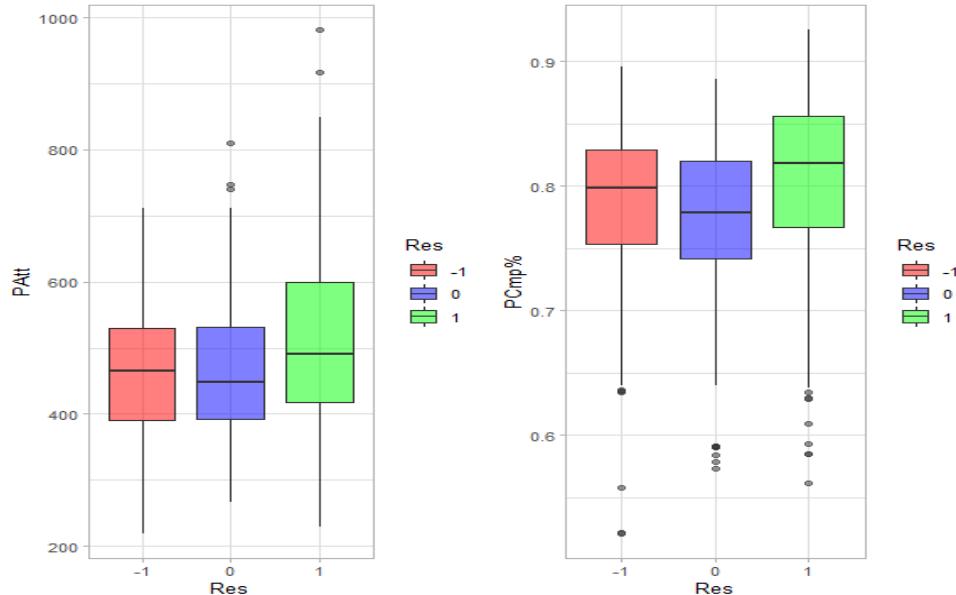


Figura 3.9: A sinistra il boxplot della variabile numerica PAtt rispetto ai valori della variabile risposta Res e a destra il boxplot della variabile numerica PCmp% rispetto ai valori della variabile risposta Res

rispetto ai valori della variabile risposta Res. Da ciò si può ipotizzare che ToDefPen non è significativa per la variabile risposta. Prima di escluderla si andrà ad analizzare se c'è qualche interazione con altre variabili che la fanno diventare significativa.

Nella Figura 3.11 viene riportato il boxplot della distribuzione della variabile ToAttPen

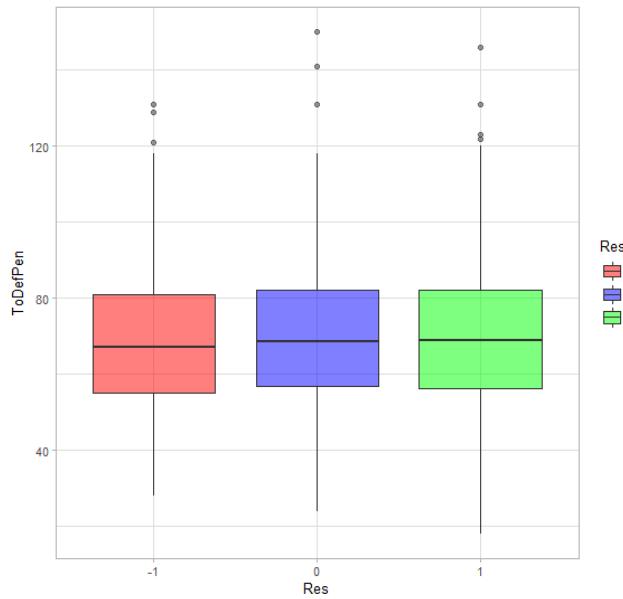


Figura 3.10: Boxplot della distribuzione della variabile `ToDefPen` rispetto ai valori della variabile risposta `Res`

rispetto ai valori della variabile risposta `Res`. Contrariamente quanto visto con la Figura 3.10 qui si nota una certa variazione tra i boxplot infatti, i valori crescono dal boxplot della sconfitta fino al boxplot della vittoria. C'è una maggior varianza per il boxplot della vittoria rispetto agli altri due boxplot. Per tutti e tre i boxplot i baffi inferiori sono leggermente meno lunghi rispetto ai baffi superiori, segno che i valori sono bassi e simili tra loro, infatti, ci sono alcuni outliers sopra al baffo superiore, segno che alcune squadre in qualche partita si sono particolarmente rese note nel produrre un quantitativo di tocchi maggiore rispetto alla distribuzione, ciò però non sembra influenzare l'esito. Le mediane sono equidistanti.

Per la relazione tra la variabile risposta e la variabile `ToDef3rd`, si ha un grafico molto simile a quello mostrato nella Figura 3.11. Il grafico di `ToDef3rd` rispetto al grafico di `ToAttPen` ha un minore numero di outliers soprattutto per il boxplot del pareggio, tale boxplot ha inoltre una varianza simile al boxplot della sconfitta. Il boxplot della vittoria invece, ha una distribuzione ben equilibrata.

Per la relazione tra la variabile risposta e la variabile `ToMid3rd`, si ha un grafico molto simile a quello mostrato nella Figura 3.11. Il grafico di `ToMid3rd` rispetto al grafico di `ToAttPen` ha un minore numero di outliers e la varianza del boxplot della sconfitta è molto simile alla mediana del boxplot del pareggio ma con la mediana più vicina al 3° quantile.

Per la relazione tra la variabile risposta e la variabile `ToAtt3rd`, si ha un grafico molto simile a quello mostrato nella Figura 3.11. Il grafico di `ToAtt3rd` rispetto al grafico di `ToAttPen` ha una minor varianza in generale per tutti e tre i boxplot e una distribuzione sbilanciata verso valori più bassi dato che tutti i baffi inferiori sono più corti rispetto ai baffi superiori. L'andamento però rimane lo stesso presente nella Figura 3.11.

Nella Figura 3.12 vengono riportati a sinistra il boxplot della variabile numerica `F1s` rispetto ai valori della variabile risposta `Res` e a destra il boxplot della variabile

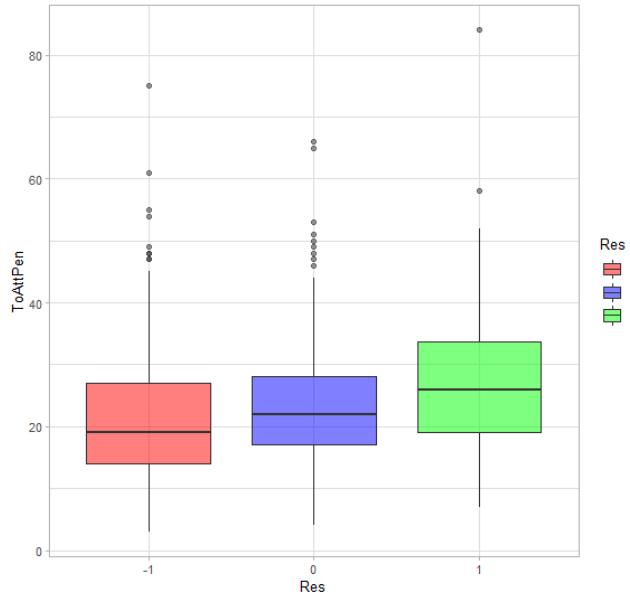


Figura 3.11: Boxplot della distribuzione della variabile `ToAttPen` rispetto ai valori della variabile risposta `Res`

numerica `F1d` rispetto ai valori della variabile risposta `Res`. Nel boxplot a sinistra si può notare che i valori più alti sono nel boxplot del pareggio e della vittoria ma nel boxplot del pareggio ci sono più valori alti in assoluto. Ciò fa ipotizzare che subire molti falli può impedire la vittoria alla squadra che li subisce. Per quanto riguarda la distribuzione sembra essere buona; c'è una minor varianza per quanto riguarda il boxplot della sconfitta.

Nel secondo boxplot si hanno valori più alti nel boxplot della vittoria e una maggior varianza rispetto al boxplot della sconfitta. Sembra perciò che dal grafico si può intuire che se la squadra non commette dei falli allora sarà più soggetta a perdere.

Per la relazione tra la variabile risposta `Res` e la variabile `Off`, si ha un grafico molto simile a quello mostrato nella Figura 3.8. Il grafico di `Off` rispetto al grafico di `Saves` ha un numero minore di valori per il boxplot della sconfitta rispetto agli altri due boxplot inoltre, le mediane dei boxplot della sconfitta e del pareggio sono attaccate al 1° quantile.

Per la relazione tra la variabile risposta `Res` e la variabile `Crs`, si ha un grafico molto simile a quello mostrato nella Figura 3.13. Nel grafico di `Crs` rispetto al grafico di `Saves`, il boxplot della sconfitta ha maggior varianza e il baffo inferiore dei boxplot della sconfitta e della vittoria sono più corti rispetto ai baffi superiori.

Nella Figura 3.13 viene riportato il boxplot della distribuzione della variabile `Int` rispetto ai valori della variabile risposta `Res`. Sorprendentemente, valori più alti sono registrati nel boxplot della sconfitta, anche se la mediana risulta essere più vicina al 1° quantile sottolineando che c'è un maggior numero di valori bassi piuttosto che alti. Le mediane dei restanti boxplot invece, sono ben equilibrate ma il boxplot del pareggio risulta avere meno varianza. Sembra perciò che effettuare troppe intercettazioni dei passaggi avversari contrariamente da quanto si pensi sia controproducente per la vittoria. Si segnala inoltre la presenza di alcuni outliers con valori alti di intercettazioni

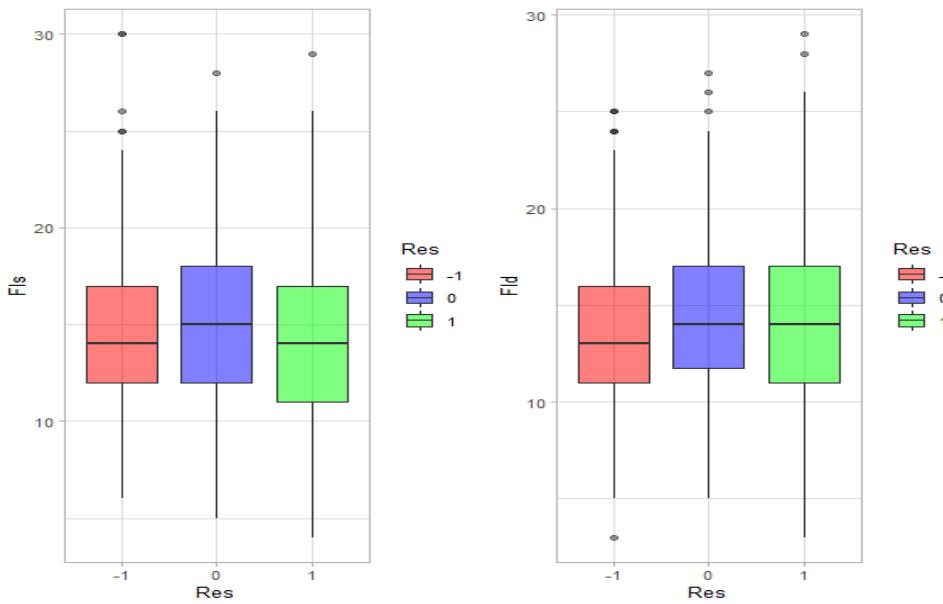


Figura 3.12: A sinistra il boxplot della variabile numerica $F1s$ rispetto ai valori della variabile risposta Res e a destra il boxplot della variabile numerica $F1d$ rispetto ai valori della variabile risposta Res

che si discostano dalle distribuzioni.

Nella Figura 3.14 viene riportato il boxplot della distribuzione della variabile $TklWin$ rispetto ai valori della variabile risposta Res . Come si può notare, vincere più contrasti possibili evita di subire una sconfitta. Infatti, ci sono valori più alti nei boxplot del pareggio e della vittoria rispetto al boxplot della sconfitta. Nello specifico però si nota che: nella distribuzione c'è una maggior presenza di valori alti nella vittoria rispetto al pareggio, graficamente lo si vede dalla mediana che nel boxplot del pareggio è più vicina al 1° quindi ha valori più bassi e lo si nota anche dal baffo inferiore che è meno lungo rispetto a quello superiore viceversa. La mediana del boxplot della vittoria risulta più vicina al 3° oltre ad avere il baffo superiore più corto rispetto a quello inferiore. C'è inoltre qualche outliers con valori più alti di contrasti vinti ma sembrano non influenzare la classificazione.

Infine nella Figura 3.15 viene riportato il boxplot della distribuzione della variabile $Recov$ rispetto ai valori della variabile risposta Res . Per entrambi i boxplot la distribuzione sembra più sbilanciata verso valori bassi quindi ad una loro maggior presenza. Infatti, entrambi i baffi inferiori sono più corti rispetto a quelle superiori. Per quanto riguarda la mediana sembra equidistante dai quantili per entrambi i tre boxplot. Si nota che il boxplot del pareggio presenta minor varianza rispetto agli altri due boxplot ma valori più alti soprattutto nei confronti del boxplot della vittoria. Perciò, un eccessivo numero di recuperi non porta alla vittoria. Inoltre, ci sono numerosi outliers.

3.2.2 Analisi possibili interazioni

Per concludere l'attività di prepossessing, non resta che analizzare le relazioni tra le covariate per individuare possibili interazioni tra di loro che possono influenzare la

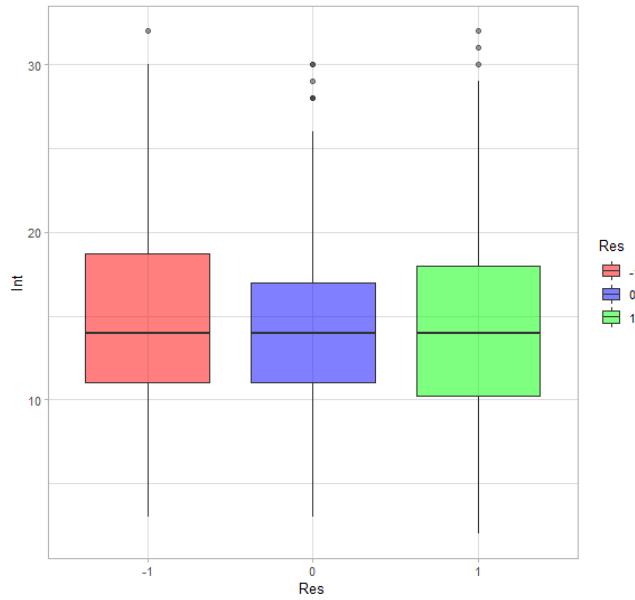


Figura 3.13: Boxplot della distribuzione della variabile **Int** rispetto ai valori della variabile risposta **Res**

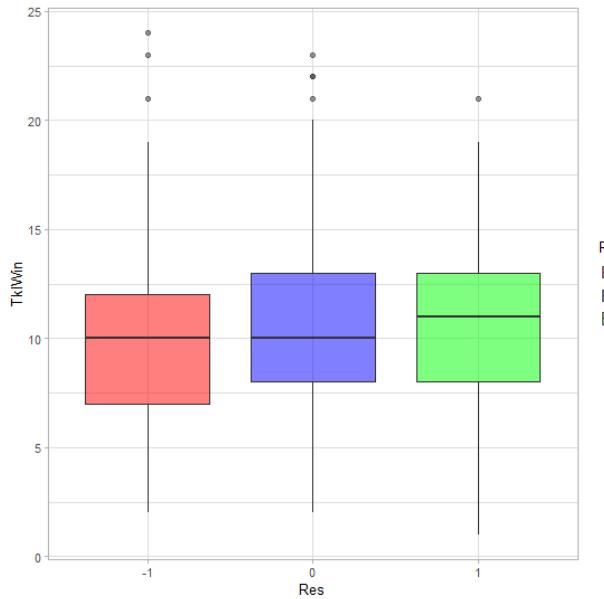


Figura 3.14: Boxplot della distribuzione della variabile **TkWn** rispetto ai valori della variabile risposta **Res**

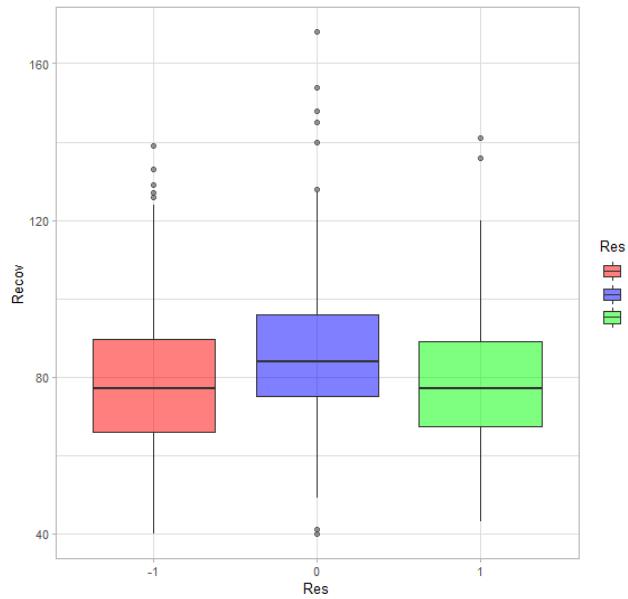


Figura 3.15: Boxplot della distribuzione della variabile `Recov` rispetto ai valori della variabile risposta `Res`

variabile risposta. Chiaramente dato che ci sono più di trenta variabili e dunque, un grandissimo numero di combinazioni, non si sono esaminate tutte le relazioni ma sono state selezionate solo alcune per l'analisi, basandosi su teorie calcistiche esaminate durante la fase di studio del problema.

Per l'analisi delle interazioni si sono utilizzati i grafici di dispersione. Un grafico di dispersione mostra la relazione tra due variabili continue. A tali grafici si è inserito una terza variabile, la variabile risposta `Res`, dove ogni punto è colorato in tre possibili colori che rappresentano una delle tre categorie di `Res`. Di conseguenza il grafico permette di visualizzare se le categorie sono ben separate e quindi se un'interazione può spiegare l'andamento dei punti della variabile risposta.

Inoltre, è stato utilizzato l'indice di correlazione, che indica la forza dell'associazione lineare espressa in valori compresi tra -1 e 1. Tale misura permette di escludere da subito alcune relazioni tra variabili se l'indice è troppo alto o basso, infatti, le relazioni troppo forti vanno escluse perché possono generare il fenomeno della collinearità. La collinearità è quel fenomeno che va a nasconde il legame tra le variabili e la variabile risposta, a causa di un legame troppo forte tra le covariate.

Nella Figura 3.16 viene mostrato il valore della correlazione per ogni possibile relazione tra variabili numeriche. Si nota che ci sono molte relazioni che hanno un valore di correlazione molto vicino a 1, in basso a sinistra del grafico. Ad esempio notiamo che la variabile `SPCmp%` ha una relazione molto forte con la variabile `PCmp%` (correlazione = 0.82), ciò è coerente perché, la variabile `SPCmp%` contiene solo i passaggi corti completati mentre `PCmp%` contiene tutti i tipi di passaggi completati, ne consegue che la ridondanza dei dati causa questa alta correlazione. Analogamente la stessa motivazione la si può applicare tra la variabile `PAtt` e la variabile `SPAtt` (correlazione = 0.91). Perciò tale motivazione è applicabile a tutte le variabili relative ai passaggi completati o relative ai passaggi tentati. Di seguito si riporteranno le interazioni che sono state individuate

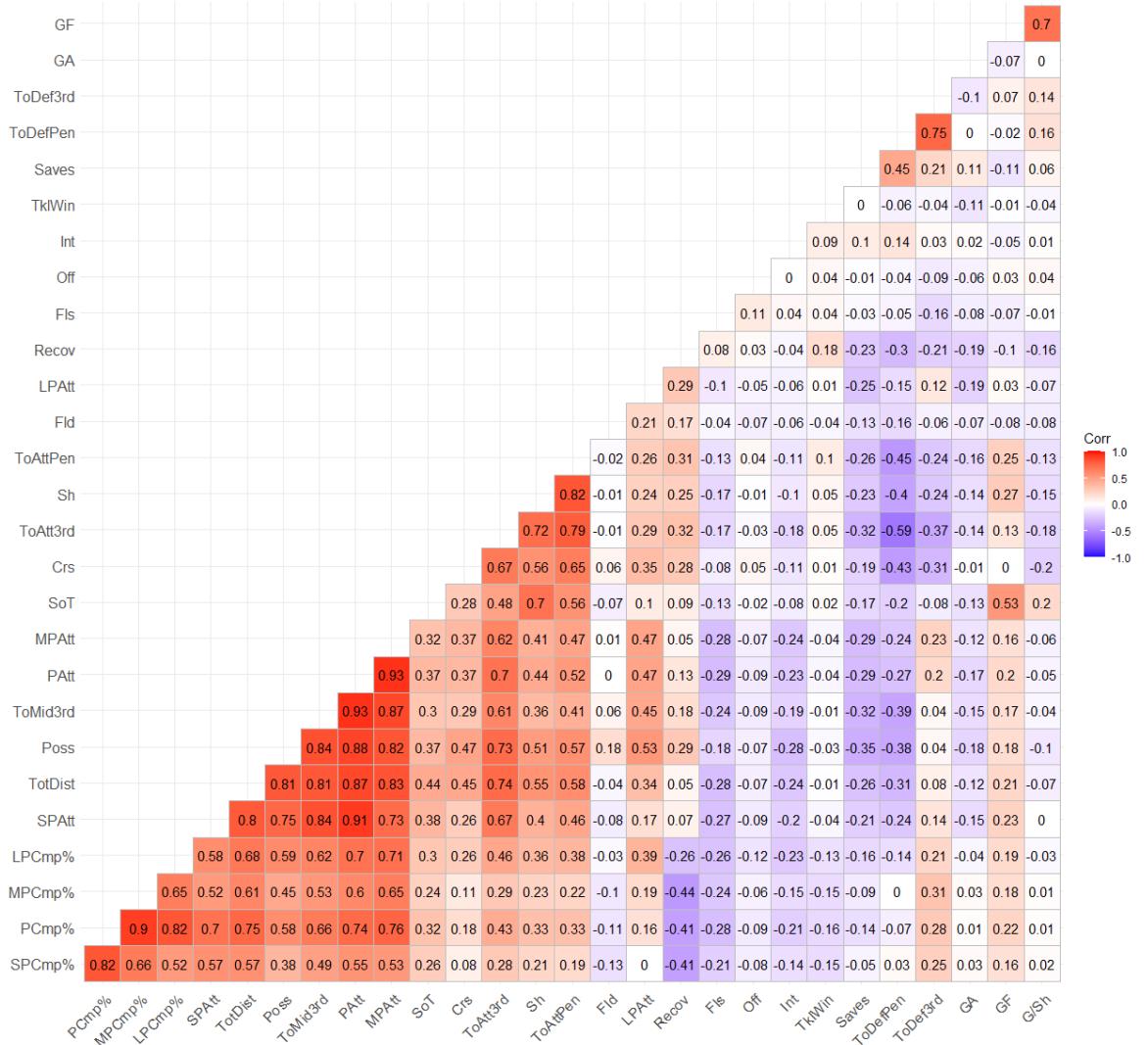


Figura 3.16: Grafico delle correlazioni di ogni coppia di variabili

come significative.

Sono state individuate le seguenti tre interazioni con la variabile Sh:

- * Interazione tra la variabile Sh e la variabile ToAttPen. È ragionevole ipotizzare che il numero di tocchi fatti nell'area di rigore avversaria possano creare azioni da tiro. È quindi possibile che tra le due variabili possa esserci una relazione. La Figura 3.17 mostra una relazione positiva tra le due variabili infatti, quando aumenta la variabile Sh aumenta anche la variabile ToAttPen e viceversa. Sono distinguibili tre differenti gruppi che rappresentano le tre categorie della variabile risposta. Inoltre, la correlazione tra le due variabili non è troppo alta (0.72). Ne consegue che un'interazione tra la variabile Sh e la variabile ToAttPen sembra essere significativa rispetto alla variabile risposta.

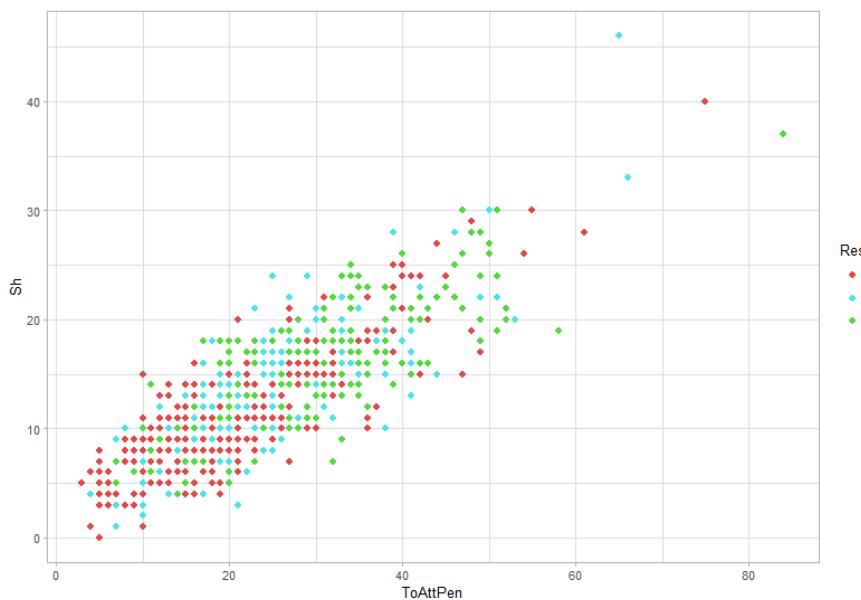


Figura 3.17: Scatterplot della distribuzione della variabile Sh rispetto ai valori della variabile ToAttPen

- * Interazione tra la variabile Sh e la variabile G/Sh. È ragionevole ipotizzare che ci sia un legame naturale tra tiri fatti e rapporto tiri-gol. La Figura 3.18 mostra una relazione negativa tra le due variabili infatti, quando aumenta la variabile Sh diminuisce anche la variabile G/Sh e viceversa. Sono distinguibili tre differenti gruppi che rappresentano le tre categorie della variabile risposta infatti, i punti della categoria vittoria sono più in alto mentre i punti delle categorie pareggio e sconfitta più in basso. Inoltre, la correlazione tra le due variabili non è bassa (-0.15). Ne consegue che un'interazione tra la variabile Sh e la variabile G/Sh sembra essere significativa rispetto alla variabile risposta.
- * Interazione tra la variabile Sh e la variabile Poss. Generalmente è possibile ipotizzare che il possesso della palla possa favorire la squadra nell'effettuare i tiri. Infatti, la Figura 3.19 mostra una relazione positiva tra le due variabili,

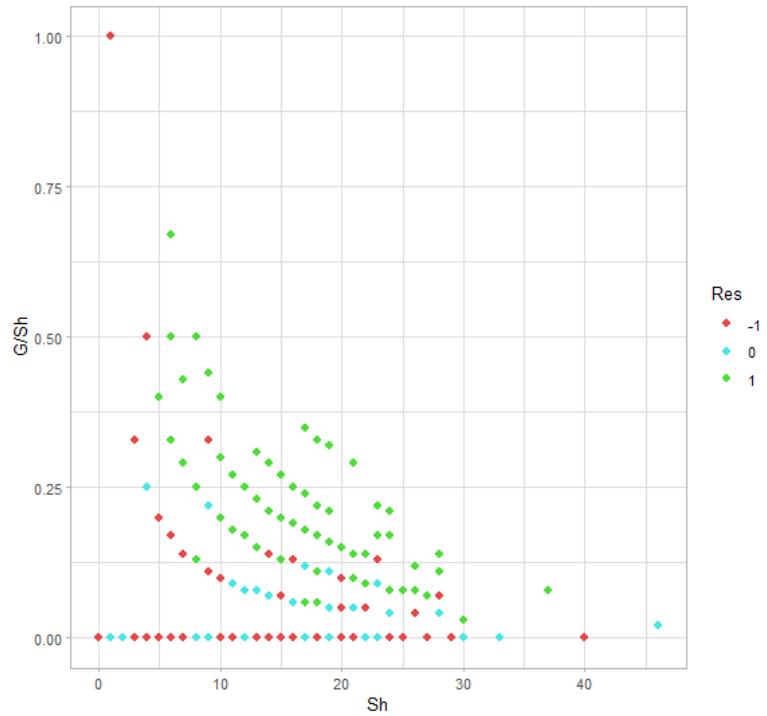


Figura 3.18: Scatterplot della distribuzione della variabile Sh rispetto ai valori della variabile G/Sh

quando aumenta la variabile **Sh** aumenta anche la variabile **Poss** e viceversa. Sono distinguibili tre differenti gruppi che rappresentano le tre categorie della variabile risposta. Inizialmente i vari punti sono mescolati tra di loro ma, con l'avanzamento emergono le direzioni di ogni categoria infatti, i punti della categoria vittoria vanno più verso destra mentre i punti della categoria sconfitta si spostano verso l'alto. I punti della categoria pareggio invece, si muovono in mezzo ai punti delle altre due categorie. La correlazione tra le due variabili non è alta (0.51). Ne consegue che un'interazione tra la variabile **Sh** e la variabile **Poss** sembra essere significativa rispetto alla variabile risposta.

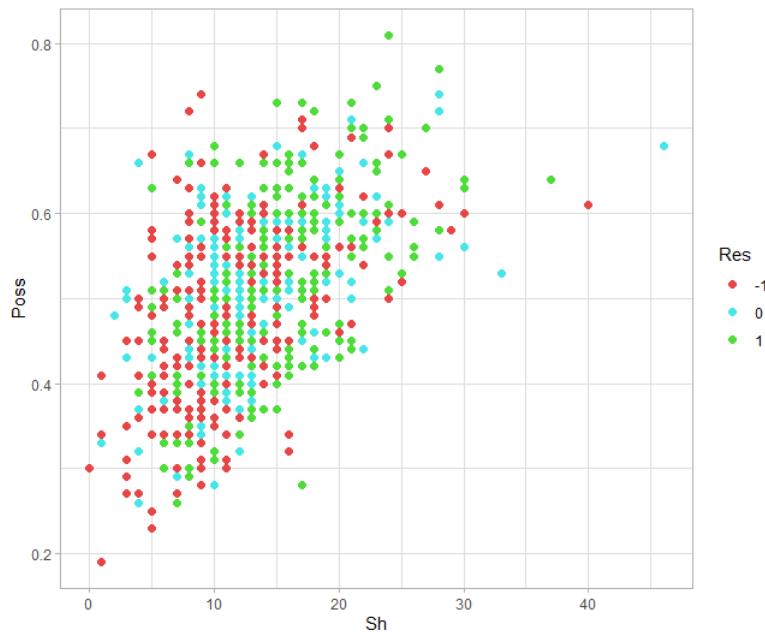


Figura 3.19: Scatterplot della distribuzione della variabile **Sh** rispetto ai valori della variabile **Poss**

Sono state individuate le seguenti tre interazioni con la variabile **ToMid3rd**:

- * Interazione tra la variabile **ToMid3rd** e la variabile **LPAtt**. Si suppone che tra le due variabili ci sia una relazione perché molti lanci lunghi per le punte partono proprio del centrocampo. La Figura 3.20 mostra un andamento leggermente a "nuvola" ma comunque, è possibile individuare una relazione positiva tra le due variabili. Infatti, quando aumenta la variabile **ToMid3rd** aumenta anche la variabile **LPAtt** e viceversa. Sono distinguibili tre differenti gruppi che rappresentano le tre categorie della variabile risposta. Inizialmente i vari punti sono mescolati tra di loro ma, successivamente i punti della categoria vittoria vanno molto in alto mentre i punti della categoria sconfitta rimangono molto più bassi muovendosi verso destra, invece i punti della categoria pareggio anche essi vanno verso destra ma rimanendo più alti rispetto ai punti della categoria sconfitta. La correlazione tra le due variabili non è alta (0.45). Ne consegue che un'interazione tra la variabile **ToMid3rd** e la variabile **LPAtt** sembra essere significativa rispetto alla variabile risposta.

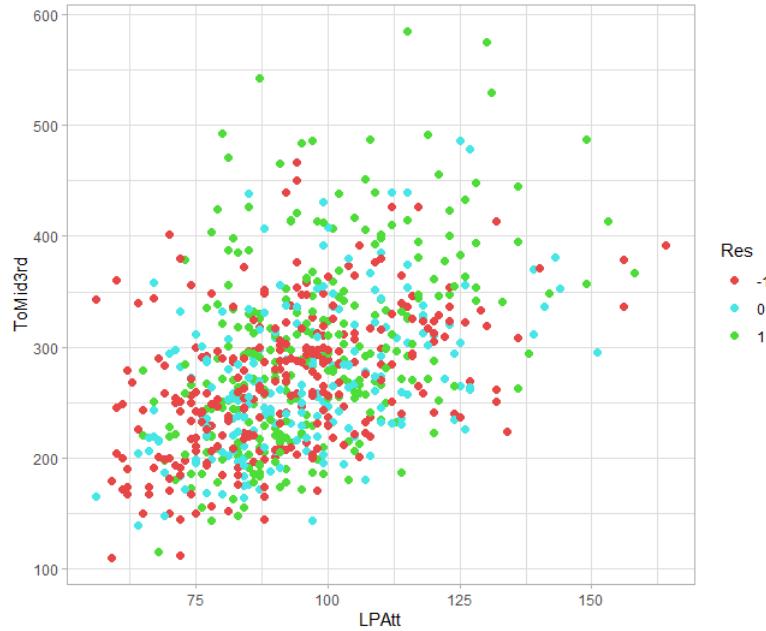


Figura 3.20: Scatterplot della distribuzione della variabile ToMid3rd rispetto ai valori della variabile LPAtt

- * Interazione tra la variabile ToMid3rd e la variabile PCmp%. Per le stesse ragioni illustrate nel punto precedente si ipotizza una relazione tra le variabili. La Figura 3.21 mostra una relazione positiva tra le due variabili infatti, quando aumenta la variabile ToMid3rd aumenta anche la variabile PCmp% con un'andamento simile ad una funzione esponenziale. Sono distinguibili tre differenti gruppi che rappresentano le tre categorie della variabile risposta, dove i punti più in alto sono della categoria del pareggio, leggermente più sotto ci sono i punti della vittoria che però verso la fine del grafico raggiungono i valori più alti e infine, i punti della sconfitta. La correlazione tra le due variabili non è alta (0.66). Ne consegue che un'interazione tra la variabile ToMid3rd e la variabile PCmp% sembra essere significativa rispetto alla variabile risposta.

Infine sono state individuate le seguenti interazioni:

- * Interazione tra la variabile TotDist e la variabile PCmp%. Naturalmente per effettuare i passaggi e completarli è possibile farlo solo se ci si muove con la palla. La Figura 3.22 mostra una relazione positiva tra le due variabili infatti, quando aumenta la variabile TotDist aumenta anche la variabile PCmp% con un'andamento simile ad una funzione esponenziale. Sono distinguibili tre differenti gruppi che rappresentano le tre categorie della variabile risposta, dove i punti più in alto sono della categoria del pareggio, leggermente più sotto ci sono i punti della vittoria e infine i punti della sconfitta. La correlazione tra le due variabili non è troppo alta (0.75). Ne consegue che un'interazione tra la variabile TotDist e la variabile PCmp% sembra essere significativa rispetto alla variabile risposta.
- * Interazione tra la variabile PAtt e la variabile PCmp%. Data la loro naturale

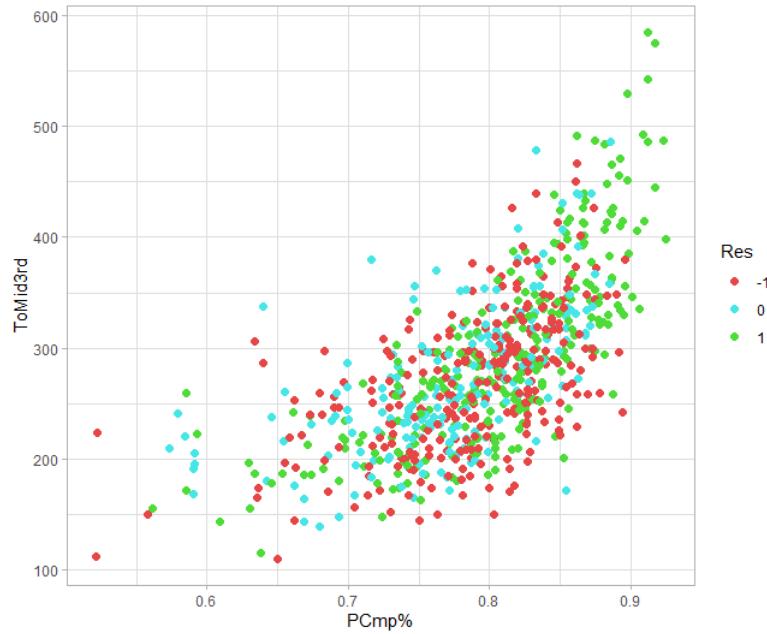


Figura 3.21: Scatterplot della distribuzione della variabile **ToMid3rd** rispetto ai valori della variabile **PCmp%**

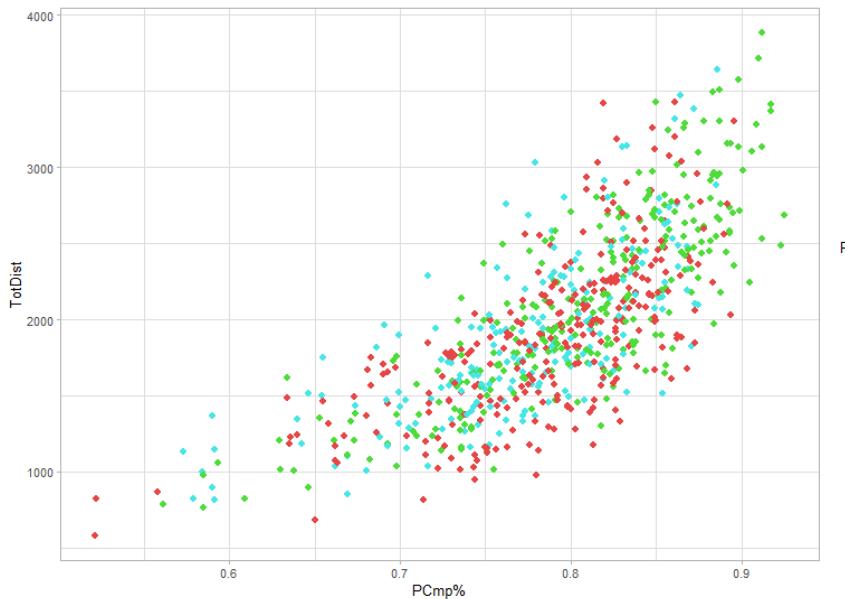


Figura 3.22: Scatterplot della distribuzione della variabile **TotDist** rispetto ai valori della variabile **PCmp%**

correlazione si ipotizza che ci sia un'interazione. La Figura 3.23 mostra una relazione positiva tra le due variabili infatti, quando aumenta la variabile PAtt aumenta anche la variabile PCmp% con un'andamento simile ad una funzione esponenziale. Sono distinguibili tre differenti gruppi che rappresentano le tre categorie della variabile risposta, dove i punti più in alto sono della categoria del pareggio, leggermente più sotto ci sono i punti della vittoria e infine i punti della sconfitta. La correlazione tra le due variabili non è troppo alta (0.74). Ne consegue che un'interazione tra la variabile PAtt e la variabile PCmp% sembra essere significativa rispetto alla variabile risposta.

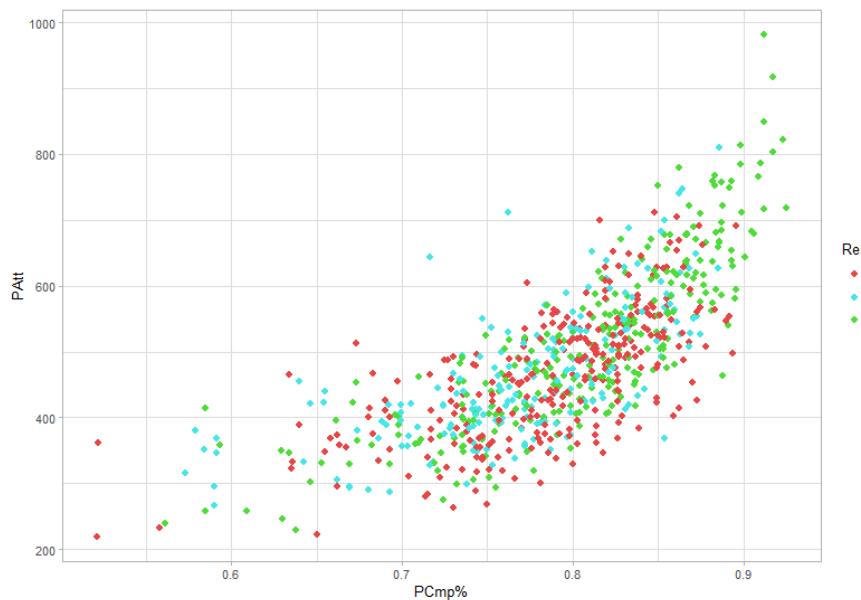


Figura 3.23: Scatterplot della distribuzione della variabile PAtt rispetto ai valori della variabile PCmp%

- * Interazione tra la variabile ToDefPen e la variabile ToAttPen. Come ci si può aspettare la Figura 3.24 mostra una relazione negativa tra le due variabili, quando aumenta la variabile ToDefPen diminuisce anche la variabile ToAttPen e viceversa. Sono distinguibili tre differenti gruppi che rappresentano le tre categorie della variabile risposta infatti, i punti della categoria vittoria sono quelli più distanti dallo zero mentre i punti delle categorie pareggio e sconfitta sono più vicini allo zero. Inoltre, la correlazione tra le due variabili non è bassa (-0.45). Ne consegue che un'interazione tra la variabile ToDefPen e la variabile ToAttPen sembra essere significativa rispetto alla variabile risposta.

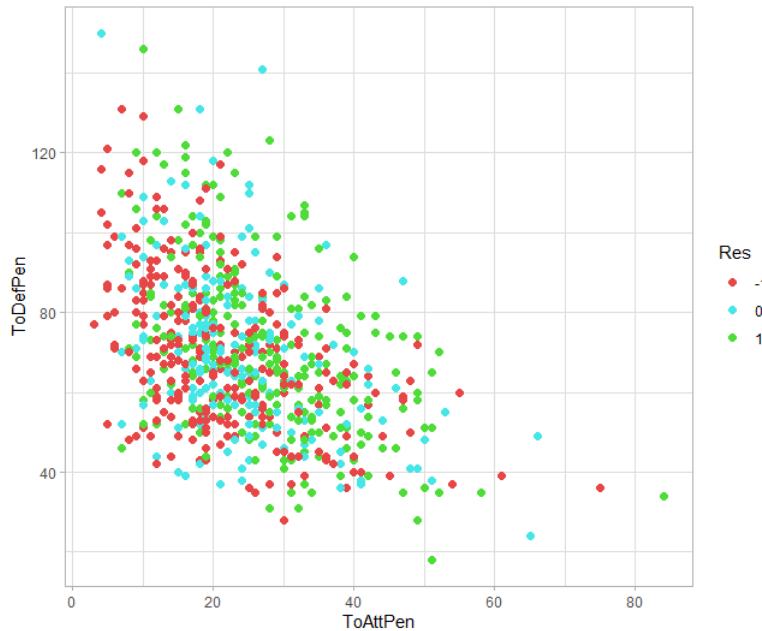


Figura 3.24: Scatterplot della distribuzione della variabile `ToDefPen` rispetto ai valori della variabile `ToAttPen`

3.3 Ulteriori modifiche del dataset

Nelle sezioni precedenti si è descritto come si è costruito il *dataset* e come esso è stato strutturato. Tale struttura ha il vantaggio di rendere il *dataset* di facile interpretazione, ma deve essere riadattato per poter utilizzare le funzioni del pacchetto **bt2** e del pacchetto **btl**.

Innanzitutto la variabile risposta `Res` viene modificata in modo tale che valgano le seguenti condizioni

$$Res = \begin{cases} 1 & \text{se la squadra in casa batte la squadra ospite,} \\ 2 & \text{se la partita termina con un pareggio,} \\ 3 & \text{se la squadra ospite batte la squadra in casa.} \end{cases} \quad (3.1)$$

La variabile risposta `Res` rimane ancora un fattore ordinato.

Di seguito vengono presentate le modifiche effettuate specifiche per i due pacchetti.

3.3.1 Modifiche per il pacchetto BradleyTerry2

Innanzitutto il modello richiede che le due variabili `Team` e `Vs` siano di tipo fattore oppure che costituiscano un `data.frame`. Un `data.frame` è una raccolta di vettori di osservazioni che devono avere tutti la stessa lunghezza ma possono essere di tipo diverso: variabili nominali (fattori) o variabili numeriche. Le variabili `Team` e `Vs` sono state trasformate in `data.frame` in modo da poter inserire al loro interno tutte le variabili descritte nella sezione precedente. Inoltre, i valori della variabile `AtHome` sono stati convertiti in 1 se `TRUE` mentre in 0 se `FALSE`.

Per poter creare i due `data.frame` occorre raccogliere le informazioni sulle partite giocate fuori casa dalle squadre indicate nella variabile `Team`. Con la funzione [11.1](#) viene implementato ciò in R. La definizione è la seguente.

Innanzitutto viene creato un vettore vuoto per ogni variabile presente nel *dataset*, ad eccezione di `AtHome` che verrà gestita in un modo diverso. Viene creato il vettore `del` che tiene traccia di quali osservazioni saranno da eliminare. Si eliminano le osservazioni delle partite giocate fuori casa dalle squadre indicate nella variabile `Team` perché dopo le modifiche saranno ridondanti. La variabile `k` è l'indice usato per scorre il *dataset* per trovare i dati dell'avversario. La variabile `z` è l'indice usato per inserire un nuovo elemento nel vettore `del`. Il funzionamento è il seguente.

Il primo ciclo `for` scorre tutto il *dataset* alla ricerca delle righe con i dati delle partite giocate in casa dalla squadra indicata in `Team` infatti, al suo interno il primo costrutto `if` controlla se la partita è in casa per `Team` se sì, parte un secondo ciclo `for` che anche esso scorre tutto il *dataset* per cercare la riga con la partita giocata della squadra indicata in `Vs`. All'interno del secondo ciclo `for` c'è un costrutto `if` che controlla se la *j*-esima riga si riferisce alla stessa partita indicata nella *i*-esima riga, se sì allora si salvano tutti i dati nei vettori e si incrementa l'indice `k`. Se il primo `if` da esito negativo allora si andrà a inserire l'indice dell'*i*-esima riga nel vettore `del` perché contiene informazioni di una partita giocata fuori casa dalla squadra indicata in `Team` e viene incrementato l'indice di uno `z`.

Grazie a questa funzione un nuovo *dataset* con 380 righe viene creato eliminando tutte quelle righe con valore `FALSE` su `AtHome`. Perciò ogni partita nel dataset viene descritta da una sola specifica riga.

3.3.2 Modifiche per il pacchetto BTLLasso

Il pacchetto richiede una specifica lista strutturata nel seguente modo:

- * `Y3`: Oggetto risposta per il pacchetto con tre categorie di risposta, inoltre include i seguenti attributi:
 - `response`: Variabile risposta di tipo fattore ordinato.
 - `first.object`: Vettore che indica il nome della squadra che gioca in casa.
 - `second.object`: Vettore che indica il nome della squadra che gioca fuori casa.
 - `subject`: Vettore che indica a quale giornata appartiene ogni partita.
 - `subject.names`: Vettore di tipo fattore che indica l'identificativo di ogni giornata.
 - `object.names`: Vettore di tipo fattore che indica l'identificativo di ogni squadra.
 - `m`: Indica il numero di giornate presenti.
 - `n`: Indica il numero di squadre presenti.
 - `k`: Indica il numero di categorie presenti.
 - `with.order`: Vettore di tipo logico che indica per ogni partita se considerare l'effetto dell'ordine.
- * `Z1`: Matrice che contiene tutti i dati sulle variabili raccolte suddivise per squadra e per partita.

Con la funzione 11.2 vengono raccolte le informazioni per produrre l'elemento Y3. Nella funzione ci sono tre cicli, il primo per scorre le giornate, il secondo per scorrere le partite della i-esima giornata, il terzo per trovare l'altra osservazione della j-esima partita, dato che si ricorda per ogni partita ci sono due osservazioni. Quindi man mano vengono raccolte le varie informazioni vengono ritornate attraverso *side effects*.

Per creare la matrice Z1 viene utilizzata la funzione 11.4, la quale estrae tutte le variabili indicate nell'attributo `covs`. Tale funzione estrae una alla volta le variabile attraverso la funzione 11.3. La funzione 11.3 scorre tutto il *dataset* raccogliendo per ogni squadra tutti i valori registrati in ogni partita per la variabile indicata nell'attributo `cov`. Una volta estratti tutti i dati della z-esima variabile viene creata una nuova matrice contenente i dati raccolti precedentemente delle altre variabili e quelli della z-esima variabile, inserendo `n` colonne dove `n` indica il numero di squadre presenti. Perciò, viene creata per ogni squadra una colonna per ogni per ogni variabile raccolta. Infine viene creata per ogni colonna la sua etichetta, nella forma *variabile.squadra*. Per creare le etichette viene utilizzata la funzione 11.5.

4 | IL MODELLO BRADLEY-TERRY

Nel seguente capitolo verranno introdotti differenti modelli per il confronto a coppie, iniziando con il modello Bradley-Terry versione base fino a presentare tutte le sue estensioni usate per l'analisi trattata. Infine, verrà illustrata la penalizzazione applicata.

4.1 Introduzione al Modello Bradley-Terry

Il modello Bradley-Terry (**bradley1952rank**) è un modello probabilistico che permette di predire il risultato di un confronto a coppie. Un confronto a coppie è un processo di comparazione tra una serie di oggetti dove ogni oggetto viene confrontato in coppia con un altro oggetto determinando per ogni confronto, se l'oggetto è preferibile all'altro. Formalmente, dato un set di n oggetti $\{\alpha_1, \dots, \alpha_n\}$, un set di n parametri $\{\gamma_1, \dots, \gamma_n\}$ che rappresentano ciascuno l'abilità/forza dell'i-esimo oggetto, la variabile casuale associata al risultato del confronto a coppie $Y_{i,j}$ con $i < j \in \{1, \dots, n\}$, la probabilità che il risultato sia $\alpha_i \succ \alpha_j$ è

$$P(\alpha_i \succ \alpha_j) = P(Y_{i,j} = 1) = \frac{\exp(\gamma_i - \gamma_j)}{1 + \exp(\gamma_i - \gamma_j)}. \quad (4.1)$$

Il risultato $\alpha_i \succ \alpha_j$ può essere letto come "l'oggetto α_i è preferito all'oggetto α_j ", " α_i batte l'oggetto α_j " oppure " α_i è migliore dell'oggetto α_j ". La variabile casuale $Y_{i,j}$ è di tipo binario, cioè $Y_{i,j} = 1$ se l'oggetto α_i è preferito all'oggetto α_j e $Y_{i,j} = 0$ se l'oggetto α_j è preferito all'oggetto α_i . I parametri γ_i sono stimati dal modello attraverso la massima verosimiglianza. È necessario imporre un vincolo per identificare gli oggetti. Tali vincoli possono essere il vincolo di somma $\sum_{i=1}^n \gamma_i = 0$ oppure il vincolo dell'oggetto di riferimento. Per il vincolo dell'oggetto di riferimento si intende che viene fissato $\gamma_i = 0$ per un oggetto $\alpha_i \in \{1, \dots, n\}$, mentre il valore dei parametri γ_j degli altri oggetti α_j sarà la differenza rispetto all'oggetto di riferimento α_i .

Il modello può essere alternativamente espresso in forma di logit lineare

$$\text{logit}(\alpha_i \succ \alpha_j) = \log \left(\frac{P(\alpha_i \succ \alpha_j)}{P(\alpha_j \succ \alpha_i)} \right) = \log \left(\frac{\exp(\gamma_i)}{\exp(\gamma_j)} \right) = \gamma_i - \gamma_j. \quad (4.2)$$

Il modello descritto è chiamato *modello non strutturato*. Il modello non strutturato non considera covariate e, in generale, non presta alcuna attenzione all'eterogeneità causata dai soggetti dei confronti. Per la nostra analisi, vengono considerati un numero di oggetti pari a $n=20$, cioè il numero di squadre partecipanti alla Serie A.

4.2 Modello Bradley-Terry con categorie di risposta ordinate

In molti contesti di comparazione tra oggetti, è possibile che sia richiesto di dare una scala di preferenza tra un oggetto e un altro. In tal caso, la variabile casuale presenta K possibili categorie di risposta con $K > 2$. Le scelte di preferenza devono avere un ordine, dal risultato meno gradevole al più gradevole per ogni oggetto. Avere K categorie di risposta ordinate con $K > 2$ è di interesse per le comparazioni calcistiche. Infatti, non è sufficiente stimare la probabilità di vittoria o di sconfitta ma deve essere obbligantemente preso in considerazione anche il pareggio come risultato. Inoltre, anche l'ordine delle K categorie di risposta è importante, infatti, un oggetto preferisce il pareggio piuttosto che la sconfitta. Perciò il modello (4.2) con una variabile risposta binaria non è adeguato.

Modelli che consentono un numero generale di categorie K sono stati proposti da **bradley1952rank** e **tutz1986bradley**. In particolare, **tutz1986bradley** mostrò come due modelli per l'analisi di dati ordinati possono essere adattati per i confronti a coppie.

Il primo modello presentato è detto a *collegamento cumulativo* il quale sarà usato per l'analisi. Il modello sfrutta la rappresentazione tramite variabili latenti. In generale, data la variabile continua casuale latente $Z_{i,j}$, sia K il numero di gradi della scala di preferenza e siano $\theta_1 < \theta_2 < \dots < \theta_{K-1}$ le soglie tale che $Y_{i,j} = k$ quando $\theta_{k-1} < Z_{i,j} < \theta_k$. Allora:

$$P(Y_{i,j} \leq k) = \frac{\exp(\theta_k + \gamma_i - \gamma_j)}{1 + \exp(\theta_k + \gamma_i - \gamma_j)}, \quad (4.3)$$

con $k \in \{1, \dots, K\}$ che indica le possibili categorie di risposta. I parametri θ_k rappresentano le cosiddette soglie per le singole categorie di risposta, cioè sono i migliori valori in cui dividere le categorie. Tali soglie vengono stimate dai dati. In generale vi è imposta una simmetria del modello, in modo che valga: $P(Y_{i,j} = k) = P(Y_{j,i} = K - k + 1)$. Pertanto le soglie sono ristrette a $\theta_k = -\theta_{K-k}$ e se, K è dispari, $\theta_{K/2} = 0$, per garantire che le probabilità siano simmetriche, cioè il risultato opposto abbia la stessa probabilità di verificarsi. Per garantire che le probabilità siano non negative per le singole categorie di risposta si impone il seguente vincolo $-\infty = \theta_0 < \theta_1 < \dots < \theta_{K-1} < \theta_K = \infty$. Dato che la soglia per l'ultima categoria è fissata a $\theta_K = \infty$, allora $P(Y_{i,j} \leq K) = 1$. La probabilità di una singola categoria di risposta può essere derivata dalla differenza tra categorie adiacenti, come segue,

$$P(Y_{i,j} = k) = P(Y_{i,j} \leq k) - P(Y_{i,j} \leq k-1).$$

Il modello ha anche la seguente rappresentazione logit lineare

$$\text{logit}(Y_{i,j} \leq k) = \theta_k + \gamma_i - \gamma_j. \quad (4.4)$$

Il secondo modello invece proposto da **agresti1992analysis** è detto *modello a categorie adiacenti*. In questo caso il collegamento è applicato alle probabilità di risposte adiacenti piuttosto che alle probabilità cumulative, riducendosi così al modello Bradley-Terry quando sono consentite solo due categorie. Quando sono consentite solo tre categorie, il modello coincide con quello di **davidson1970extending** che verrà presentato di seguito.

Il modello a categorie adiacenti è più semplice da interpretare rispetto ai modelli a collegamenti cumulativi poiché la probabilità si riferisce a un determinato risultato anziché a raggruppamenti di risultati.

Sia θ il parametro stimato dai dati che indica quanto è auspicabile la non preferenza. Allora (**davidson1970extending**)

$$P(Y_{i,j} = 2|Y_{i,j} \neq 0) = \frac{\exp(\gamma_i - \gamma_j)}{1 + \exp(\gamma_i - \gamma_j)}, \quad (4.5)$$

$$P(Y_{i,j} = 1) = \frac{\theta \sqrt{\exp(\gamma_i) * \exp(\gamma_j)}}{\exp(\gamma_i) + \exp(\gamma_j) + \theta \sqrt{\exp(\gamma_i) * \exp(\gamma_j)}}, \quad (4.6)$$

$$P(Y_{i,j} = 0|Y_{i,j} \neq 1) = \frac{\exp(\gamma_j - \gamma_i)}{1 + \exp(\gamma_j - \gamma_i)}. \quad (4.7)$$

Si è riportato la modellazione di tutti e tre i possibili risultati, con γ_n che rappresenta la forza degli oggetti in comparazione. La probabilità che l'oggetto α_i batta l'oggetto α_j è rappresentata da (4.5), mentre la probabilità che l'oggetto α_j batta l'oggetto α_i è rappresentata da (4.7). Sia (4.5) sia (4.7) rimangono uguali alla probabilità (4.2) descritta precedentemente. Invece, per la probabilità che l'oggetto α_i pareggi con l'oggetto α_j (4.6), viene aggiunto il parametro θ . Il parametro θ rappresenta, quindi, quanto sia auspicabile il pareggio.

4.3 Modello Bradley–Terry con effetti dell'ordine

Nel modello descritto nella sezione 4.2 è necessario imporre la simmetria tra le categorie di risposta. Purtroppo, la simmetria imposta risulta essere non adeguata in alcuni contesti. Tra questi vi è anche il calcio poiché l'ordine degli oggetti (le squadre) conta. Infatti, in una partita di calcio, la prima squadra che viene indicata tra le due squadre è quella che gioca in casa, per cui ci si attende crei un vantaggio sull'avversario. Perciò, il presupposto che le categorie di risposta siano simmetriche non vale più.

Un possibile modello riadattato al problema esposto è il seguente:

$$P(\alpha_i \succ \alpha_j) = P(Y_{i,j} = 1) = \frac{\exp(\delta + \gamma_i - \gamma_j)}{1 + \exp(\delta + \gamma_i - \gamma_j)}. \quad (4.8)$$

L'effetto dell'ordine come, ad esempio, il vantaggio di giocare in casa in ambito calcistico, viene trattato come un parametro δ . Se $\delta > 0$, viene attribuito un vantaggio all'oggetto α_i , aumentando la probabilità che vinca il confronto o, nel caso di categorie di risposta ordinate, di avere un risultato superiore rispetto all'oggetto α_j . Chiaramente il valore di δ deve essere stimato dai dati.

Invece un modello con categorie di risposta ordinate con l'effetto dell'ordine è il seguente

$$P(Y_{i,j} \leq k) = \frac{\exp(\delta + \theta_k + \gamma_i - \gamma_j)}{1 + \exp(\delta + \theta_h + \gamma_i - \gamma_j)}. \quad (4.9)$$

Il modello (4.8) e il modello (4.9) hanno anche la seguente rappresentazione logit lineare rispettivamente,

$$\text{logit}(\alpha_i \succ \alpha_j) = \delta + \gamma_i - \gamma_j, \quad (4.10)$$

$$\text{logit}(\alpha_i \succ \alpha_j) = \delta + \theta_h + \gamma_i - \gamma_j. \quad (4.11)$$

Perciò si fissa che la prima squadra che viene indicata tra le due squadre è quella che gioca in casa.

4.4 Modello Bradley–Terry con variabili esplicative

In precedenza, è stato descritto un modello che valutasse il grado di preferenza per un oggetto α_i rispetto a un oggetto α_j , senza considerare nessuna covariata. Spesso, però, si è interessati a capire quali elementi possono essere associati al risultato della comparazione. Prima di esporre il modello con covariate, è necessario fare una distinzione tra soggetti e oggetti e successivamente distinguere i tre tipi di covariate di un confronto a coppie, ovvero, le covariate specifiche al soggetto x_p , le covariate specifiche all'oggetto z_i e infine le covariate specifiche al soggetto e all'oggetto z_{pi} per i soggetti p , con $p = 1, \dots, m$ e gli oggetti α_i , con $i = 1, \dots, n$.

Gli oggetti sono le entità che vengono confrontate in un confronto a coppie. I soggetti invece, sono le unità che stabiliscono la preferenza tra gli oggetti in un confronto a coppie. Nel calcio gli oggetti sono le squadre di calcio, mentre i soggetti sono le partite di calcio tramite le quali avviene la comparazione tra le squadre. Nell'analisi trattata il numero di soggetti sarà pari a $m=380$, cioè il numero di partite giocate nel campionato di Serie A.

Di seguito vengono illustrate le tre tipologie di covariate in un confronto a coppie:

- * **covariate specifiche del soggetto.** Caratterizzano i soggetti che eseguono i confronti tra oggetti e quindi queste covariate variano solo tra soggetti. Ad esempio, nel calcio, covariate specifiche del soggetto possono essere il numero di spettatori o le condizioni meteo. Sia x_p un vettore di covariate specifiche del soggetto, β_i il peso stimato delle covariate per ogni oggetto α_i e β_{i0} l'intercetta. Allora l'abilità γ_{pi} dell'oggetto α_i nel soggetto p sarà

$$\gamma_{pi} = \beta_{i0} + x_p^T \beta_i.$$

Con l'inclusione di covariate specifiche del soggetto, il modello è in grado di spiegare l'eterogeneità tra i soggetti. Le covariate specifiche del soggetto nei confronti a coppie sono state considerate, ad esempio da **francis2010** e **Turner2012Firth**.

- * **covariate specifiche dell'oggetto.** Caratterizzano gli oggetti che vengono confrontati. Non variano tra i soggetti, ma tra gli oggetti. Nel caso del calcio, una covariata specifica dell'oggetto può essere il valore di mercato della rosa della squadra di calcio. Un loro utilizzo lo si può trovare in **schauberger2017**. Sia z_i un vettore di covariate specifiche all'oggetto, τ il peso uguale per tutti gli oggetti e β_{i0} l'intercetta. Allora l'abilità γ_i dell'oggetto α_i sarà

$$\gamma_{pi} = \gamma_i = \beta_{i0} + z_i^T \tau.$$

Il peso τ è un parametro globale che insieme a z_i rappresenta l'abilità spiegata delle covariate, mentre β_{i0} rappresenta la parte dell'abilità non spiegata dalle covariate. Nell'analisi in esame questo tipo di covariate non verrà usato.

- * **covariate specifiche del soggetto e dell'oggetto.** Questi tipi di covariate possono variare sia per oggetti e sia per i soggetti. Nel calcio, ad esempio il possesso palla, è una covariata che varia per ogni singola squadra e per ogni singola partita. Tali variabili vengono approfondite in **thurner2000policy** e in **mauerer2015modeling**. Sia z_{pi} un vettore di covariate specifiche del soggetto e dell'oggetto, η_i il peso stimato delle covariate per ogni oggetto, β_{i0} l'intercetta. Allora l'abilità γ_{pi} dell'oggetto α_i nel soggetto p sarà

$$\gamma_{pi} = \beta_{i0} + z_{pi}^T \eta_i.$$

Contrariamente alle covariate specifiche al soggetto, le covariate specifiche al soggetto e all'oggetto posso essere modellate con un effetto globale

$$\gamma_{pi} = \beta_{i0} + z_{pi}^T \tau,$$

dove τ rappresenta il peso stimato delle covariate. Come si può notare, il parametro τ non ha alcun indice, questo perché l'effetto della covariate è uguale su tutti gli oggetti.

Il parametro β_{i0} nelle specificazioni precedenti è l'intercetta specifica dell'oggetto. Tale parametro spiega la maggior parte della forza dell'oggetto. Infatti, le covariate possono essere viste come estensioni contenenti effetti aggiuntivi dell'abilità dell'oggetto che non sono spiegati dall'intercetta. In tal senso, gli effetti della covariata possono aiutare a spiegare i risultati imprevisti di un soggetto che non possono essere completamente spiegati esclusivamente dall'intercetta (**cattelan2012models**) e (**schauburger2017**). Nella Sezione 4.3 viene presentato l'effetto dell'ordine degli oggetti in competizione. Invece dell'effetto d'ordine globale δ , che è uguale per tutti gli oggetti, è possibile specificare l'effetto d'ordine specifico per ogni oggetto α_i , quindi δ_i . Nella Tabella 4.1 vengono riassunti tutti i tipi di covariate e tutte le possibili parametrizzazioni che possono essere applicate.

Quindi, il parametro abilità γ_{pi} di un oggetto α_i con $i = 1, \dots, n$ su un soggetto p , con $p = 1, \dots, m$ non è altro che una combinazione lineare dei parametri precedentemente spiegati. Da ciò si ottiene il modello capace di utilizzare le covariate. Tale modello viene chiamato modello strutturato e fa parte dei *generalized linear models* (GLMs). Aggiungendo al modello 4.9 le covariate di tipo specifiche del soggetto e dell'oggetto al modello e l'effetto dell'ordine con effetto specifico dell'oggetto si ha

$$P(Y_{p(i,j)} \leq k) = \frac{\exp(\delta_i + \theta_k + \beta_{i0} - \beta_{j0} + x_{pi}^T \eta_i - x_{pj}^T \eta_j)}{1 + \exp(\delta_i + \theta_k + \beta_{i0} - \beta_{j0} + x_{pi}^T \eta_i - x_{pj}^T \eta_j)}, \quad (4.12)$$

con $i < j \in \{1, \dots, 20\}$ e $p \in \{1, \dots, 380\}$.

Come si può vedere, il parametro abilità γ_i è stato sostituito da β_{i0} e $x_{pi}^T \eta_i$. Analogamente anche per γ_j .

4.5 Stima e penalizzazione

È importante considerare che, con l'inserimento di un elevato numero di covariate, si ha un aumento di complessità del modello. Dato che si utilizza un modello lineare, un

Tipo di covariate	Tipo di effetto	$\gamma_{pi} =$	$\gamma_{pj} =$	$\gamma_{p(ij)} = \gamma_{pi} - \gamma_{pj}$
Intercetta	Spec. dell'oggetto	β_{i0}	β_{j0}	$\beta_{i0} - \beta_{j0}$
Effetto dell'ordine	Globale	$+ \delta$		$+ \delta$
Effetto dell'ordine	Spec. dell'oggetto	$+ \delta_i$		$+ \delta_i$
Spec. del soggetto x_p	Spec. dell'oggetto	$+ x_p^T \beta_i$	$+ x_p^T \beta_j$	$+ x_p^T (\beta_i - \beta_j)$
Spec. dell'oggetto z_i	Globale	$+ z_i^T \tau$	$+ z_{si}^T \tau$	$+ (z_i - z_j)^T \tau$
Spec. del sogg. e dell'ogg. z_{pi}	Globale	$+ z_{pi}^T \tau$	$+ z_{pj}^T \tau$	$+ (z_{pi} - z_{pj})^T \tau$
Spec. del sogg. e dell'ogg. z_{pi}	Spec. dell'oggetto	$+ x_{pi}^T \eta_i$	$+ x_{pj}^T \eta_i$	$+ x_{pi}^T \eta_i - x_{pj}^T \eta_j$

Tabella 4.1: Tipi di covariate e possibili parametrizzazioni applicabili al parametro abilità γ .

eccessivo livello di complessità può portare a problemi di identificabilità ed efficienza. Infatti, se si include una covariata specifica del soggetto e dell'oggetto equivale a inserire n covariate dove n è il numero di oggetti in considerazione. Nel nostro caso abbiamo 26 covariate di tipo specifiche del soggetto e dell'oggetto da inserire, ciò significa che se abbiamo 20 squadre abbiamo 520 parametri da stimare, un numero chiaramente troppo grande. Inoltre, la complessità è aumentata dalla presenza di una intercetta per ogni oggetto. La soluzione alla gestione di modelli complessi è l'utilizzo di metodi di *shrinkage*, che includono termini di penalizzazione nelle procedure di stima. I metodi di *shrinkage* (**copas1983regression**) regolarizzano il processo di stima spingendo le stime dei parametri verso zero. L'inclusione della penalizzazione dei termini potrebbe migliorare o leggermente peggiorare il modello, ma la variabilità associata alle stime sarà minore. C'è perciò un *trade-off* di cui occuparsi, infatti più è forte la penalità inserita, più i parametri saranno vicini a zero e quindi meno informazioni si avranno sui parametri, di conseguenza sarà elevata la varianza a causa della perdita di informazioni. Ovviamente più informazioni vengono perse meno complesso sarà il modello ma allo stesso tempo sarà poco preciso. Non si massimizzerà la log verosimiglianza ma la log verosimiglianza penalizzata

$$L(\varepsilon)^p = L(\varepsilon) - \lambda P(\varepsilon),$$

dove $L(\varepsilon)$ è la log verosimiglianza con ε che rappresenta il vettore contenente tutti i parametri del modello e $P(\varepsilon)$ è un termine di penalizzazione. Il parametro λ è il parametro di tuning che stabilisce quanto forte deve essere la penalizzazione sui parametri.

Per eseguire la penalizzazione è necessario trasformare in scale comparabili tutte le covariate, per evitare che la penalizzazione influisca in modo diverso sui parametri.

Oltre a una riduzione di complessità del modello, si vogliono ottenere i seguenti due obiettivi:

- * Eseguire una selezione delle covariate spingendo a zero quelle non significative,
- * Valutare se vi è la formazione di un *cluster* di valori di una covariata su più squadre, in modo tale da utilizzare un effetto globale piuttosto che un effetto specifico dell'oggetto.

Quello che si intende per *cluster* di valori è che, durante la penalizzazione può accadere che una covariata ha come valore del parametro lo stesso per tutti gli oggetti in esame, perciò non occorre considerare n volte la covariata ma soltanto una volta, riducendo così la complessità. Per soddisfare questi punti, come metodo di penalizzazione verrà applicato il *LASSO* (**tibshirani1996regression**).

4.5.1 LASSO

Il metodo *Least Absolute Shrinkage and Selection Operator* detto *LASSO* (**tibshirani1996regression**) è un metodo di penalizzazione che permette di eseguire una selezione delle covariate. La selezione è possibile perché la penalizzazione applicata spinge i parametri ad essere uguali a zero. Si ha la seguente penalizzazione

$$L(\varepsilon)^P = L(\varepsilon) + \lambda \sum_{j=1}^p |\beta_j|,$$

dove $\lambda \sum_{j=1}^p |\beta_j|$ è il fattore di penalizzazione che include una norma L_1 dei parametri. Grazie alla norma L_1 è possibile eseguire la selezione delle covariate.

Sono state utilizzate solo alcune modalità di penalizzazione tra quelle disponibili; quindi, verranno esposte solo quelle effettivamente utilizzate. Si veda ad esempio, **schauberger2019btlasso** per una trattazione delle varie penalizzazioni esistenti.

Le applicazioni del *LASSO* sono le seguenti:

- * **Penalizzazione all'effetto partita in casa**

Si è applicata la seguente penalizzazione su 20 parametri che rappresentano l'effetto di giocare una partita in casa

$$P_\delta(\delta_1, \dots, \delta_n) = \sum_{i < j}^n |\delta_i - \delta_j|.$$

Come si può notare l'effetto partita in casa δ_i è un parametro con effetto specifico all'oggetto. La penalizzazione risultante è data dai valori delle differenze assolute tra tutti i parametri. Tale tipo di penalizzazione permette di formare clusters di oggetti, nel nostro caso squadre, con un valore dell'abilità simile.

È possibile applicare una penalizzazione sul valore assoluto del parametro ma, dato che non vi sono dubbi che l'effetto casa sia determinante per l'esito di una partita di calcio (**lago2016home**), non verrà applicata nessun'altra penalizzazione.

- * **Penalizzazione alla covariata specifica del soggetto e dell'oggetto**
Si è applicata la seguente penalizzazione sui parametri della c-esima covariata

$$P_{\eta_c}(\eta_{1,1}, \dots, \eta_{n,m}) = \sum_{p=1}^m \sum_{i < j}^n |\eta_{ip} - \eta_{jp}| + \sum_{p=1}^m \sum_{i < j}^n |\eta_{ip}|.$$

Rispetto alla penalizzazione precedente è stata aggiunta una penalizzazione al valore assoluto delle covariate. Questo perché non sappiamo in anticipo se una variabile è associata alla risposta oppure no. Perciò con la penalizzazione al valore assoluto possiamo fare selezione delle covariate.

Le penalizzazioni illustrate precedentemente se combinate permettono di ottenere il parametro

$$P(\varepsilon) = \sum_{c=1}^C P_{\eta_c} + P_\delta,$$

con $C = 26$ che indica il numero di covariate descritte nel Capitolo 2.

4.5.2 Scelta del parametro di tuning

Un punto cruciale per le tecniche di *shrinkage* è la determinazione del parametro di *tuning* ottimo λ , cioè il grado di penalizzazione che fornisce il miglior trade off tra complessità e precisione del modello. Per farlo ci si affiderà alla K-Fold Cross-Validation Ranked Probability Score (RPS). Il RPS ([gneiting2007strictly](#)) per categorie di risposte ordinate $y \in \{1, \dots, K\}$ misura quanto siano buone le previsioni espresse come distribuzioni di probabilità rispetto ai valori osservati. Sia K il numero di categorie della variabile risposta y , il RPS è così espresso

$$RPS(y, \pi(k)) = \sum_{k=1}^K \pi(k) - \mathbb{1}(y \leq k)^2$$

dove $\pi(k)$ rappresenta la probabilità cumulativa $\pi(k) = P(y \leq k)$ mentre $\mathbb{1}$ è una funzione che restituisce 1 se il parametro in ingresso è vero, 0 altrimenti. A differenza delle altre possibili misure dell'errore, ad esempio la devianza, il RPS tiene conto dell'ordine di preferenza.

5 | RISULTATI DEI MODELLI BRADLEY-TERRY

In questo capitolo vengono presentate le stime e i risultati ottenuti dai modelli Bradley-Terry (BTM) descritti nel Capitolo 4. Inoltre, sarà riportata l'applicazione del metodo LASSO con relativi risultati. Infine si riporteranno le predizioni sugli esiti delle partite prodotte dai modelli per essere poi confrontate con le predizioni dei bookmakers.

5.1 Premesse

I risultati che verranno esposti non tengono in considerazione le variabili esplicative del numero di gol fatti GF e dei gol subiti GA. Questo perché provocano la non convergenza del modello. Infatti, le librerie usate non sono in grado di interpretare correttamente i dati. Una possibile soluzione è allargare il numero di categoria K inserendo per ogni possibile risultato finale una categoria. Data l'elevata complessità che raggiunge il modello esteso Bradley-Terry, non sono state inserite le interazioni illustrate nel Capitolo 3.

5.2 BTM con effetto dell'ordine

Le analisi dello studio iniziano con l'applicazione del modello (4.9). Tale modello presenta una struttura abbastanza semplice, in cui la stima dell'abilità delle squadre tiene conto solo degli esiti osservati delle varie partite e del vantaggio di giocare in casa. La stima dei parametri soglia θ_1 e θ_2 è pari rispettivamente, a -0.669 e 0.669 mentre il parametro δ globale per tutte le squadre è di 0.099 con uno *standard error* (SE) di 0.126. Si nota che la possibilità di giocare in casa è effettivamente un vantaggio in quanto la stima del parametro è positiva. Nella Figura 5.1 vengono riportati i risultati ottenuti in ordine dell'abilità stimata. Inoltre viene riportato lo *Standard Error* (SE) il *Quasi Standard Error* (QSE) (`firth2004quasi`) e il *Quasi Variance* (QV) (`firth2004quasi`) per ogni squadra.

Nonostante, la semplicità del modello, viene offerta una stima delle abilità delle squadre che rispecchia molto il piazzamento mostrato nella Figura 2.1. Infatti, solo quattro squadre hanno un piazzamento diverso da quello reale. L'Udinese e il Sassuolo hanno il piazzamento invertito con una stima dell'abilità che è molto simile. Ciononostante, il risultato è comunque soddisfacente dato che nella stagione in esame il distacco tra Udinese e Sassuolo è stato solo di tre punti. Anche Genoa e Salernitana hanno un piazzamento differente da quello reale.

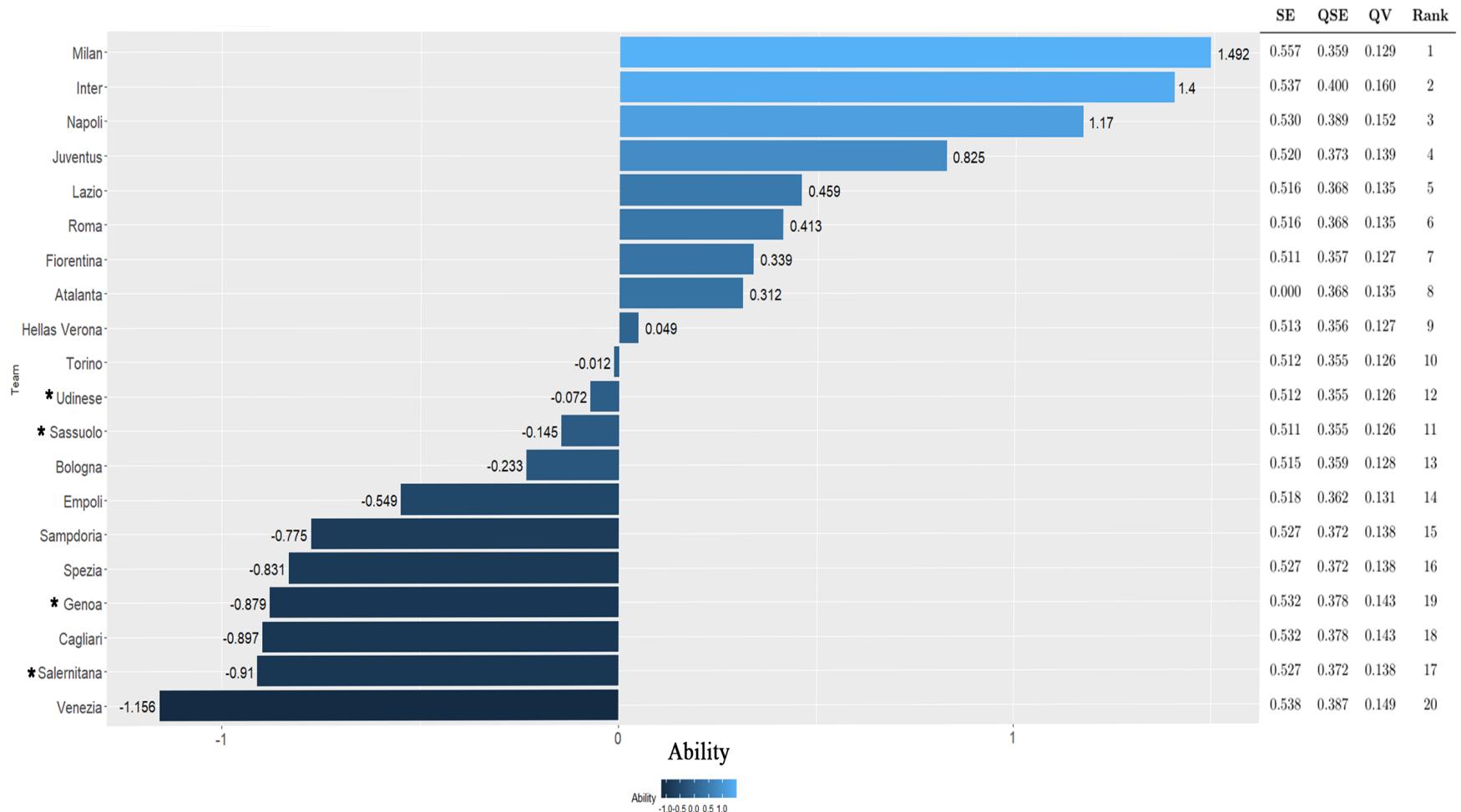


Figura 5.1: Barplot che indica per ogni squadra l'abilità stimata dal modello (4.9). A fianco al grafico viene riportato lo *Standard Error* (SE), il *Quasi Standard Error* (QSE) e il *Quasi Variance* (QV). Nel grafico viene indicato con un asterisco le squadre con un piazzamento stimato diverso da quello reale, anche esso riportato a destra del grafico.

Per quanto riguarda il Genoa tale risultato può essere spiegato dal fatto che all'inizio del campionato ha avuto un buon andamento (vedi **storyGenoa**) e dall'ottenimento di punti contro Juventus, Inter, Roma e Atalanta, cioè squadre considerate tra le più forti del campionato. Per quanto riguarda la stima al ribasso della Salernitana è determinata dal suo pessimo andamento per la maggior parte del campionato fatta eccezione per l'ultima parte, dove sono stati guadagnati la maggior parte dei punti, tanto da permettere alla squadra di guadagnare all'ultima giornata la salvezza (vedi **storySal**).

Il *Quasi Variance* (QV) (**firth2004quasi**) è un metodo che fornisce un'approssimazione della varianza, ed è utilizzato per confrontare livelli differenti di un fattore. Il tipo fattore è stato illustrato nel Capitolo 3. Il QV è stato introdotto da **firth2004quasi** per risolvere il problema della categoria di riferimento. Tale problema si riferisce al fatto che risulta essere semplice confrontare un livello qualsiasi del fattore con il suo livello di riferimento ma confrontare tra loro due livelli entrambi non di riferimento non è possibile. Grazie al QV cioè è possibile, infatti permette di confrontare tra di loro diversi livelli che non sono di riferimento con il vantaggio di non dover riportare tutta la matrice delle varianze e delle covarianze per effettuare i confronti. Nel nostro caso abbiamo la variabile **team** di tipo fattore con la squadra Atalanta come livello di riferimento. Grazie al QV ci viene fornita il QSE, una stima dello SE che verrà utilizzata per confrontare le abilità stimate dei diversi livelli, ovvero le squadre, per poter dedurre se la differenza di abilità tra due squadre sia significativa dal punto di vista statistico. Con il QSE le squadre vengono trattate come unità indipendenti. Esempio di applicazioni del QSE e del QV sul BTM è possibile trovarli in **firth2004quasi** e in **turner2012bradley**.

Perciò, confrontiamo le stime dei valori delle abilità delle squadre classificate nelle prime due posizioni, rispettivamente il Milan e Inter. Il QSE per Milan è di 0.359, mentre per l'Inter è di 0.400. La differenza assoluta tra le loro abilità è pari a $|1.492 - 1.4| = 0.092$. Applicando il calcolo pitagorico è possibile calcolare lo QSE, cioè un SE approssimato, relativo alla differenza tra abilità. Questo risulta essere $(0.359^2 + 0.400^2)^{\frac{1}{2}} = 0.537 > 0.092$. Perciò, la differenza in termini di abilità tra le due squadre non è significativa da un punto di vista statistico. Infatti, le due squadre hanno una differenza di soli due punti.

5.3 BTM con covariate specifiche dell'oggetto

Si consideri l'estensione del modello Bradley-Terry con covariate specifiche dell'oggetto. Il modello applicato è il seguente

$$P(Y_{p(i,j)} \leq k) = \frac{\exp(\delta + \theta_k + \beta_{i0} - \beta_{j0} + x_{pi}^T \tau - x_{pj}^T \tau)}{1 + \exp(\delta + \theta_k + \beta_{i0} - \beta_{j0} + x_{pi}^T \tau - x_{pj}^T \tau)}, \quad (5.1)$$

dove l'effetto dell'ordine δ , cioè il vantaggio di giocare la partita in casa, ha ancora un effetto globale per tutte le squadre, mentre x_{pi}^T è il vettore con tutti i valori delle ventisei covariate per l'i-esima squadra e per la p-esima partita. Il parametro τ è il peso medio stimato di ogni covariata. Le covariate, perciò, sono specifiche del soggetto e dell'oggetto ma con un effetto specifico dell'oggetto.

La stima dei parametri soglia θ_1 e θ_2 è pari, rispettivamente a -1.113 e 1.113, mentre il parametro δ globale per tutte le squadre è salito a 0.27, con uno SE di 0.142. Nella Figura 5.2 e nella Tabella 5.1 vengono riportate le stime delle abilità delle squadre con i relativi SE, QSE e QV, e le stime di ogni covariata sul modello con relativo SE.

CAPITOLO 5. RISULTATI DEI MODELLI BRADLEY-TERRY

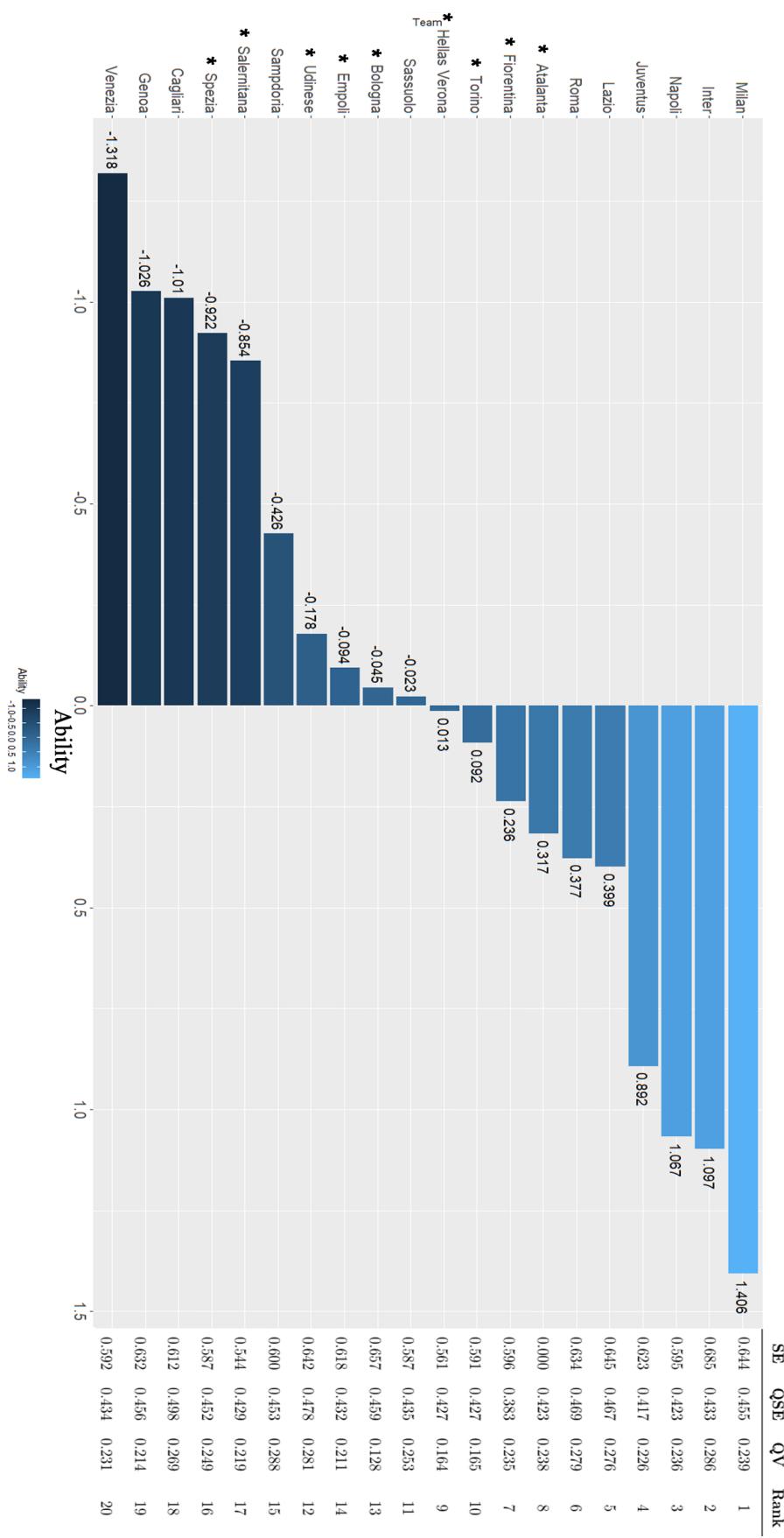


Figura 5.2: Barplot che indica per ogni squadra l'abilità stimata dal modello (5.1). A fianco al grafico vengono riportati i relativi *Standard Error* (SE), *Quasi Standard Error* (QSE) e *Quasi Variance* (QV). Nel grafico viene indicato con un asterisco le squadre con un piazzamento stimato diverso da quello reale, anche esso riportato a destra del grafico

Covariata	Stima	SE
ToMid3rd	1.57	0.025
G/Sh	1.135	0.317
Sh	0.787	0.085
SoT	0.536	0.324
PCmp%	0.534	0.300
ToDefPen	0.375	0.027
ToDef3rd	0.347	0.026
ToAtt3rd	0.283	0.025
Saves	0.280	0.312
Fls	0.138	0.204
Fld	0.100	0.204
TklWin	0.082	0.049
LPAtt	0.078	0.049
Poss	0.032	0.169
ToAttPen	0.027	0.044
TotDist	-0.039	0.001
Off	-0.054	0.144
PAtt	-0.080	0.053
Int	-0.082	0.057
SPCmp%	-0.100	0.136
Crs	-0.199	0.062
LPCmp%	-0.309	0.380
Recov	-0.512	0.030
SPAtt	-0.650	0.053
MPCmp%	-0.748	0.126
MPAtt	-1.011	0.050

Tabella 5.1: Stime delle covariate con relativo *Standard Error* (SE), stimate dal modello 5.1.

Dai risultati si nota che alcune variabili esplicative sono fortemente associate all'esito della partita. Come ci si aspetta le variabili esplicative legate ai tiri quindi, tiri **Sh**, tiri in porta **SoT** e il rapporto gol/tiri **G/Sh** hanno un peso stimato molto alto e positivo. Sono perciò fortemente decisive per aumentare la probabilità di vittoria. Da notare che sia **G/Sh** e sia **Sh** hanno un alto SE, quindi un'elevata variabilità. Sarà interessante, perciò, analizzare nel prossimo modello, il peso di queste covariate per ogni singola squadra. Sorprendentemente la variabile esplicativa **ToMid3rd** ovvero, il numero di tocchi con la palla fatti a centrocampo ha una forte associazione positiva con la probabilità di vittoria. Le altre covariate legate ai tocchi nelle altre zone del campo quindi **ToDefPen**, **ToDef3rd**, **ToAtt3rd** e **ToAttPen** hanno un'associazione positiva seppur minore rispetto a **ToMid3rd**. Sembra perciò che avere il controllo del centrocampo sia fondamentale per costruire azioni da gol, ma anche per mantenere un risultato positivo dalla partita. Anzi, mantenere il pallone in zone difensive con meno transizioni in zone d'attacco sembra che dia maggior probabilità di vittoria. Infatti, si può notare che un elevato numero di tocchi in area di rigore avversaria **ToAttPen** aumenti di molto poco la probabilità di vittoria. A sostegno di ciò, si consideri che il campionato italiano è spesso considerato un campionato difensivista e tattico (vedi **speculazione**), dove si spinge l'avversario a sbilanciarsi per poi attaccarlo in contropiede.

Un aspetto difensivo chiave sembra essere rappresentato dalle parate fatte **Saves**. Inoltre, anche il numero di contrasti vinti **TklWin** è positivo quando associato alla probabilità di vittoria. Sorprendentemente, le altre variabili esplicative difensive rispettivamente, numero di intercetti **Int** e numero di recuperi **Recov** hanno un'associazione negativa con la probabilità di vittoria. Al contrario di quanto si pensi il possesso della palla non sembra essere un elemento chiave per la vittoria. Infatti, la sua stima fa aumentare di molto poco la probabilità di vittoria. Analogamente, anche la distanza percorsa con la palla **TotDist** non sembra essere un elemento chiave per la vittoria, anzi va a diminuire la probabilità di vittoria. Perciò sembra che stia emergendo dall'analisi, una tendenza ad avere il controllo del gioco nei momenti giusti e nelle zone giuste del campo per aver maggior probabilità di vittoria.

Per quanto riguarda l'aggressività della squadra, la stima del modello indica che commettere falli **F1d** aumenti le probabilità di vittoria, d'altra parte subire falli **F1s** è più conveniente.

Si nota che subire un fuorigioco **Off** ha un impatto negativo sulle probabilità di vittoria. Per quanto riguarda le covariate legate ai passaggi notiamo che solo la percentuale dei passaggi completati **PCmp%** e il numero di lanci lunghi tentati **LPAtt** hanno un'associazione positiva con la probabilità di vittoria, le restanti covariate invece presentano un'associazione negativa. Un abuso di passaggi filtrati **MPAtt** o di cross **Crs** sembra essere controproducente per la vittoria. Una buona precisione sui passaggi **PCmp%** e tentare i cambi di gioco **LPAtt** invece, suggeriscono la possibilità di maggiori probabilità di vittoria.

Come fatto nella sezione precedente è possibile anche qui confrontare tra loro le squadre utilizzando i loro QSE relativi alla loro abilità stimata. Confrontando ancora le prime due squadre cioè Milan e Inter, la loro differenza assoluta di abilità è pari $|1.406 - 1.097| = 0.309$ e il relativo QSE è pari a $(0.455^2 + 0.433^2)^{\frac{1}{2}} = 0,628 > 0.309$. Si ottiene per ciò che la differenza di abilità tra le due squadre è ancora non significativa anche con l'effetto delle covariate.

5.4 Modello Bradley-Terry e LASSO

Nella sezione precedente si sono presentati i risultati ottenuti nel modello Bradley-Terry con l'inserimento di covariate con effetto specifico dell'oggetto. È però di interesse per le nostre analisi capire in che modo ogni singola covariata sia determinante nell'esito della partita a seconda della squadra in esame. Per esempio, è possibile che il possesso della palla possa essere determinante per una squadra mentre per un'altra no. A tale scopo si applicherà il modello (4.12) utilizzando covariate specifiche del soggetto e dell'oggetto. Ovviamente con l'inserimento di questo tipo di covariate il modello sarà estremamente complesso, essendo basato su 520 covariate. Di conseguenza sarà applicata una selezione delle covariate operata attraverso il metodo *LASSO* illustrato nel Capitolo 4. Sempre attraverso il *LASSO* sarà di interesse individuare clusters di squadre che per una certa covariata hanno un effetto simile. Allo stesso tempo si cercherà di individuare quali squadre invece si discostano maggiormente da questi clusters.

Purtroppo, non è stato possibile riportare gli SE delle stime a causa dell'elevata complessità del procedimento di calcolo. Questo è possibile solo attraverso la procedura di tipo *bootstrap* (**henderson2005bootstrap**), molto onerosa in termini computazionali, soprattutto con un numero elevato di covariate.

Nella Figura 5.3 e nelle Tabelle 5.2 e 5.3 vengono riportate le stime dei parametri delle abilità e delle covariate per ogni singola squadra. Si noti che non tutte le covariate hanno un'unica stima per tutte le squadre, ma in alcuni casi ci sono più stime per alcune covariate. Perciò, per ogni stima del parametro di una covariata verrà indicata quale squadra ha tale valore stimato. Nell'analisi dei risultati spesso si farà un confronto con i risultati ottenuti con il modello della sezione precedente.

Nella Figura 5.3 si può notare che le abilità stimate tramite *LASSO* sono quasi sempre in linea con il piazzamento reale, migliorando perciò le prestazioni del modello rispetto al modello con effetto specifico dell'oggetto stimato nella sezione precedente.

Purtroppo, l'abilità dell'Atalanta viene sovrastimata nonostante al termine della stagione si sia classificata dietro a Roma e Fiorentina. Tale fenomeno può essere spiegato dal fatto che l'Atalanta per larga parte della stagione militava tra il terzo e il quarto posto, ma nell'ultima parte della stagione l'Atalanta è crollata di prestazione (vedi **storyAta**). Si nota che l'abilità della Sampdoria viene sottostimata, infatti in generale, non ha avuto un buon rendimento soprattutto verso la fine della stagione (vedi **storySamp**). Nella Tabella 5.2 e nella Tabella 5.3 alcune variabili esplicative sono state portate a zero, quindi eliminate per effetto della regolarizzazione tramite *LASSO*, mentre altre hanno diversi valori a seconda della squadra in considerazione.

Tra le covariate eliminate c'è il numero di passaggi tentati PAtt che nella Tabella 5.1 del modello precedente aveva un valore stimato quasi nullo. Sorprendentemente anche la percentuale di passaggi tentati PCmp% viene eliminata dal modello nonostante per il modello precedente avesse un valore stimato alto del parametro. Anche il numero di tocchi nella tre quarti di difesa ToDef3rd viene tolta dal modello nonostante un valore stimato alto nella Tabella 5.1. Infine l'ultima variabile esplicativa eliminata interamente del modello è la distanza percorsa con la palla TotDist rimanendo in linea con quanto visto nella Tabella 5.1. Anche qui viene confermato che giocare le partite in casa Home ha un effetto positivo stimato pari a 0.310. Per quanto riguarda invece il possesso della palla Poss, come visto dal precedente modello viene stimato con un peso nullo per la maggior parte delle squadre ad eccezione di Lazio e Torino, il quale ha una stima positiva.

Il risultato della stima legata alla Lazio è un risultato in realtà non è sorprendente,

CAPITOLO 5. RISULTATI DEI MODELLI BRADLEY-TERRY

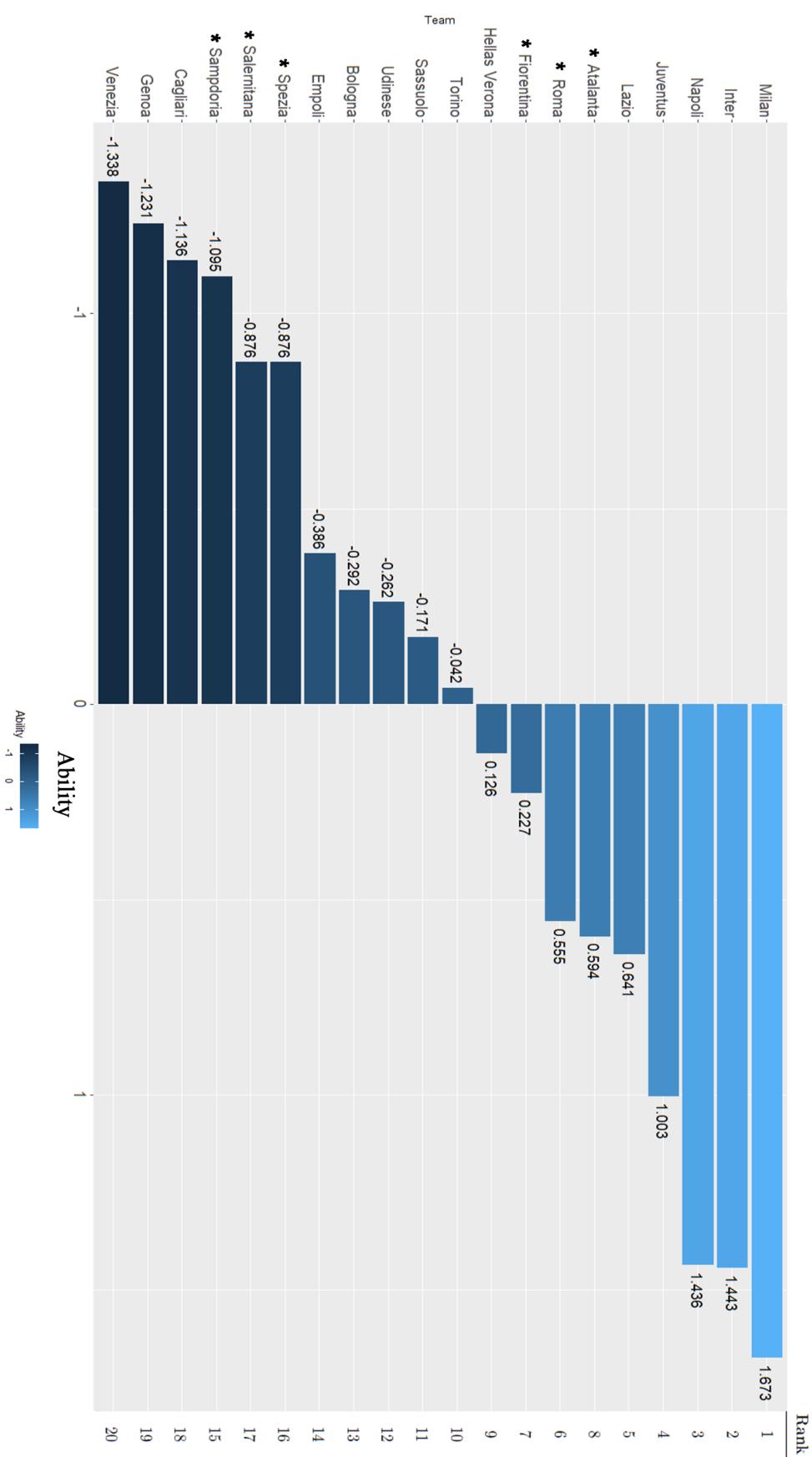


Figura 5.3: Barplot che indica per ogni squadra l'abilità stimata dal modello (4.12). Viene indicato con un asterisco le squadre con un piazzamento stimato diverso da quello reale anche esso riportato a destra del grafico.

Covariata	Stima	Squadra
Home	0.310	Tutti
Poss	0.239	Lazio
Poss	0.171	Torino
Poss	0.000	Tutti tranne Lazio e Torino
Sh	0.520	Tutti
SoT	0.596	Atalanta, Cagliari, Empoli, Genoa, Verona, Juventus, Lazio, Milan, Napoli, Salernitana, Sampdoria, Sassuolo, Spezia, Torino, Venezia
SoT	0.495	Inter, Roma
SoT	0.361	Bologna
SoT	0.263	Fiorentina
SoT	0.007	Udinese
G/Sh	1.107	Tutti
Saves	0.260	Tutti
PAtt	0.000	Tutti
PCmp%	0.000	Tutti
SPAtt	0.124	Napoli
SPAtt	0.000	Tutti tranne Napoli
SPCmp%	0.067	Tutti tranne Genoa
SPCmp%	-0.235	Genoa
MPAtt	-0.058	Tutti
MPCmp%	-0.246	Tutti tranne Bologna e Genoa
MPCmp%	-0.255	Bologna e Genoa
LPAtt	0.077	Tutti
LPCmp%	0.199	Hellas Verona
LPCmp%	0.000	Tutti tranne Bologna e Verona
LPCmp%	-0.303	Bologna

Tabella 5.2: Stime delle covariate stimate dal modello (4.12).

Covariata	Stima	Squadra
ToDefPen	0.135	Tutti
ToDef3rd	0.000	Tutti
ToMid3rd	0.147	Tutti
ToAtt3rd	-0.154	Tutti
ToAttPen	0.000	Tutti tranne Atalanta
ToAttPen	-0.311	Atalanta
TotDist	0.000	Tutti
Fls	0.219	Bologna
Fls	0.012	Tutti tranne Bologna, Napoli, Genoa e Salernitana
Fls	-0.001	Napoli
Fls	-0.030	Genoa, Salernitana
Fld	0.100	Spezia
Fld	0.015	Tutti tranne Spezia e Udinese
Fld	-0.005	Udinese
Off	0.055	Hellas Verona
Off	0.002	Tutti tranne Verona, Inter, Juventus, Milan e Napoli
Off	-0.097	Inter, Juventus, Milan e Napoli
Crs	0.000	Torino
Crs	-0.180	Tutti tranne Milan, Roma, Torino, Atalanta e Napoli
Crs	-0.359	Roma
Crs	-0.391	Milan
Crs	-0.639	Napoli
Crs	-0.671	Atalanta
Int	0.012	Tutti
TklWin	0.225	Empoli
TklWin	0.086	Tutti tranne Empoli
Recov	-0.120	Genoa
Recov	-0.132	Tutti tranne Udinese e Genoa
Recov	-0.189	Udinese

Tabella 5.3: Stime delle covariate stimate dal modello (4.12).

infatti il **sarrismotr**, neologismo per indicare il gioco applicato dall'allenatore Maurizio Sarri, allenatore della Lazio nella stagione 2021/2022, ha tra le sue caratteristiche il mantenimento del possesso della palla, oltre a una propensione offensiva (vedi **sarrismo**). Analogamente anche il gioco del Torino si fonda sul possesso palla, ma con minor propensione offensiva (vedi **torino**).

Come era atteso, il numero di tiri **Sh**, il numero di tiri in porta **SoT**, il rapporto gol/tiri **G/Sh** e il numero di parate **Saves** sono fortemente associate all'aumento della probabilità di vittoria. Si nota che per **SoT** ci sono ben cinque stime, ciò poteva essere atteso dato che nella Tabella 5.1 era stato stimato un SE pari a 0.324 che giustifica la variazione di stima da squadra a squadra.

Per quanto riguarda le variabili legate ai passaggi non ancora illustrate, abbiamo che, il numero di passaggi corti tentati **SPAtt** non risulta essere associato alla probabilità di vittoria per le squadre ad eccezione del Napoli, per la quale invece la stima risulta positiva. La percentuale di passaggi corti completati **SPCmp%**, invece, presenta una stima del parametro molto bassa per tutte le squadre ad eccezione del Genoa, squadra per la quale il valore stimato è associato ad una diminuzione della probabilità di vittoria. Il numero di passaggi medi tentati **MPAtt** diminuisce le probabilità di vittoria per tutte le squadre. Analogamente anche per la percentuale di passaggi medi riusciti **MPCmp%** ha il parametro stimato fortemente negativo.

Si nota che il numero di passaggi lunghi tentati **LPAtt** ha la stessa stima calcolata con il modello precedente per tutte le squadre. È interessante notare come la percentuale di passaggi lunghi riusciti **LPCmp%** per la maggior parte delle squadre non ha alcuna associazione con l'esito della partita. Per l'Hellas Verona è associato ad un aumento della probabilità di vittoria, al contrario al Bologna è associato ad una diminuzione delle probabilità di vittoria. Infine, si nota che al crescere del numero di cross **Crs** si ha una diminuzione della probabilità di vittoria, principalmente per Atalanta e Napoli. Resta escluso il Torino, con una stima pari a 0.

Per quanto riguarda le variabili legate al possesso, al crescere del numero di tocchi in area di rigore **ToDefPen** e a centrocampo **ToMid3rd** si ha una crescita della probabilità di vittoria. Viceversa, il numero di tocchi fatti nella tre quarti avversaria **ToAtt3rd** e nell'area di rigore avversaria **ToAttPen** è associato ad una riduzione della probabilità di vittoria.

Al subire falli **F1s** viene associato ad un aumento della probabilità di vittoria per molte squadre, specialmente per il Bologna, mentre la relazione si inverte per Napoli, Genoa e Salernitana. Per quanto riguarda l'effettuare falli **F1d** si associa una leggera probabilità di vittoria per la maggior parte delle squadre, soprattutto per lo Spezia. Tendenza inversa per l'Udinese.

Il numero di fuorigioco **Off** in generale ha una associazione non rilevante con l'esito della partita. Curiosamente per le quattro squadre con la maggior abilità stimata, cioè Milan, Inter, Napoli e Juventus, **Off** ha un impatto negativo sull'esito della partita. Tale risultato può essere spiegato dal fatto che le squadre più forti ad esempio Milan, Inter ecc., creano più azioni d'attacco, mentre le squadre meno forti per difendersi da esse, utilizzano la trappola del fuorigioco per fermarle.

Per quanto riguarda le variabili esplicative difensive, il numero di intercetti **Int** e il numero di contrasti vinti **TklWin** sono associati ad un aumento della probabilità di vittoria. Viceversa, il numero di recuperi **Recov** si associa ad una diminuzione della probabilità di vittoria a tutte le squadre. Anche qui rispetto al modello precedente è cambiato la stima delle soglie θ_1 e θ_2 che valgono rispettivamente -1.075 e 1.075.

Si ricorda scelto per ottenere i risultati illustrati precedentemente si è scelto il parametro di tuning ottimo, indicato con il simbolo λ , attraverso la procedura di K-Fold Cross

Validation spiegata nel Capitolo 4. Nella Figura 5.4 è mostrato l'andamento delle prestazioni del modello su tutti i valori assunti dal parametro di tuning λ durante l'operazione di K-Fold Cross Validation.

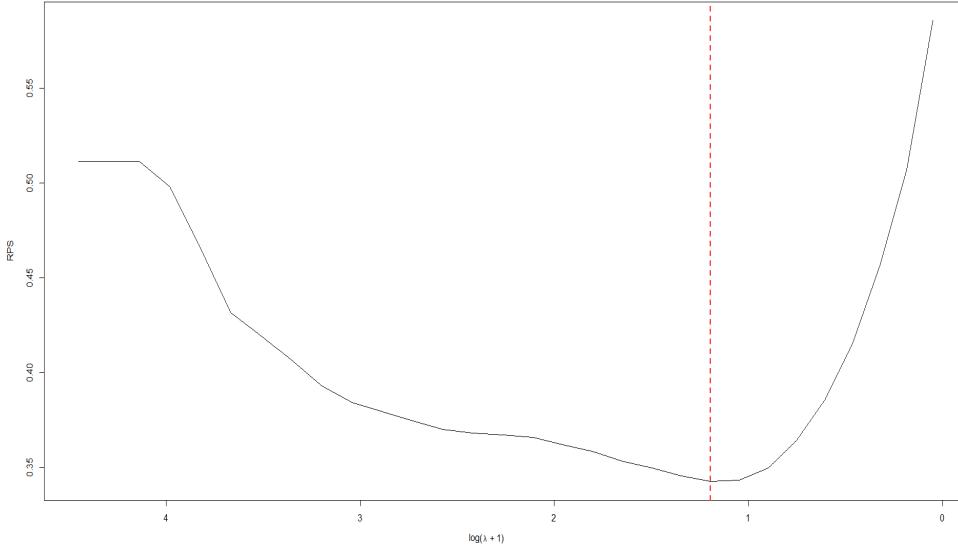


Figura 5.4: Grafico dell'andamento delle prestazioni del modello (4.12) su tutti i trenta valori assunti dal parametro di tuning, indicato con il simbolo λ , durante l'applicazione di K-Fold Cross Validation. L'andamento viene valutato in termini di Ranked Probability Score (RPS). La linea rossa tratteggiata indica il parametro di tuning λ ottimo da utilizzare.

La K-Fold Cross Validation utilizzata prevedeva l'uso di 10 gruppi ($k = 10$) e di trenta valori diversi per λ . Successivamente, i risultati ottenuti dal modello applicando i trenta diversi valori del parametro di tuning, sono stati confrontati in termini di RPS. Si nota che con il diminuire della penalizzazione il modello registra una RPS che migliora fino a quando la penalizzazione diventa troppo debole, causando un peggioramento delle prestazioni. Perciò, il parametro di tuning ottimo λ risulta essere pari a 1.196, come indicato dalla linea tratteggiata di color rosso.

In alcuni casi nelle stime dei parametri delle variabili esplicative c'è un'alta variabilità delle stime tanto da essere negative per alcune squadre mentre per altre nulle o positive. In altri casi invece, si vengono a formare dei clusters per alcune covariate. Questo fenomeno lo si può osservare chiaramente dalla Figura 5.5 alla Figura 5.17. Nei grafici vengono mostrati come cambiano le stime dei parametri associati ad ogni covariata e per ogni squadra al variare del parametro di tuning espresso in scala logaritmica. Ovviamente con un valore alto di penalizzazione si vede all'inizio che tutte le stime sono spinte a zero, ma con il diminuire della penalizzazione si iniziano ad ottenere stime diverse per la stessa covariata. La linea rossa tratteggiata indica il parametro di tuning ottimo utilizzato per ottenere i risultati illustrati precedentemente.

In Figura 5.5 viene mostrato l'andamento relativo alla stima della covariata del possesso della palla Poss, si diversificano i risultati per Lazio e Torino che si discostano nettamente dall'andamento nullo tenuto dalla maggior parte delle squadre.

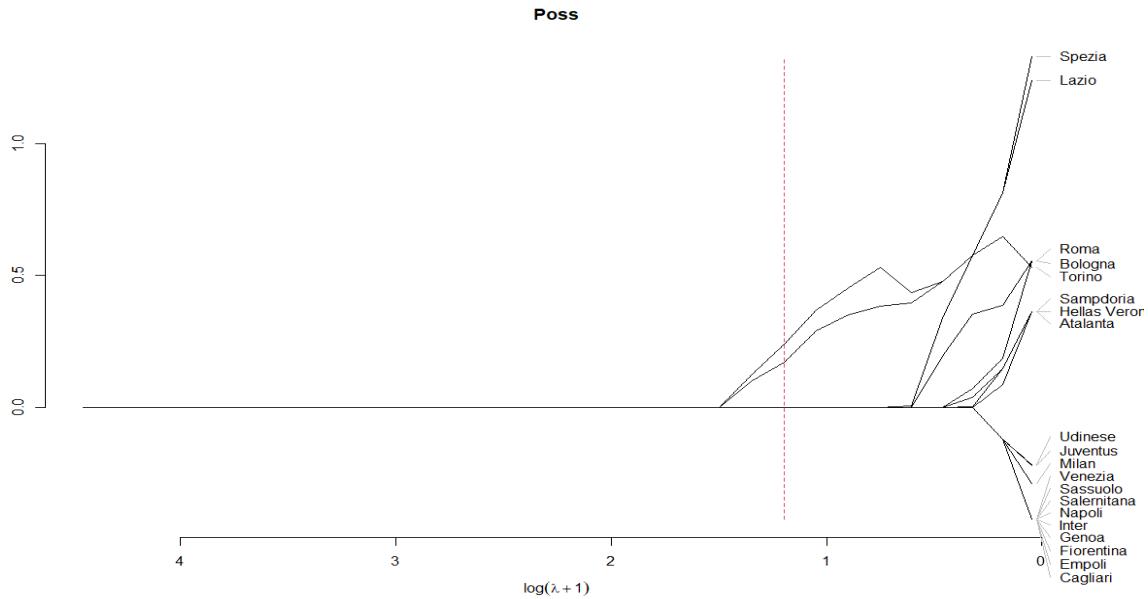


Figura 5.5: Grafico che riporta l’andamento stimato dal modello (4.12) della stima del possesso della palla per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

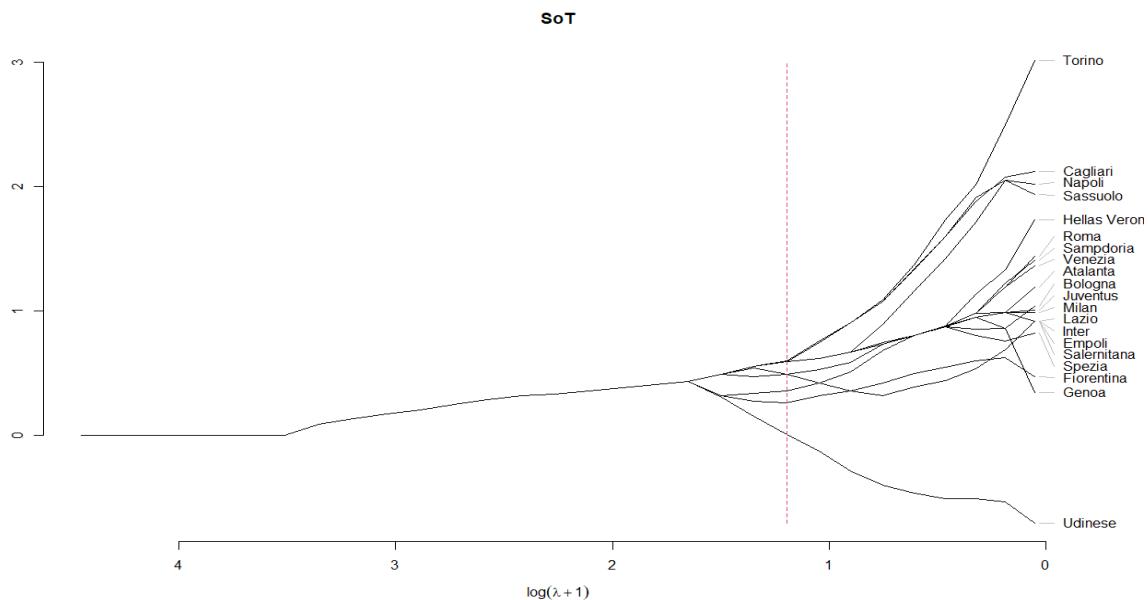


Figura 5.6: Grafico che riporta l’andamento stimato dal modello (4.12) della stima del numero di tiri in porta per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

In Figura 5.6 viene mostrato l'andamento relativo alla stima della covariata del numero di tiri in porta SoT. Si notano cinque clusters con stima positiva. Il cluster con la stima più alta contiene la maggioranza delle squadre, seguito dal secondo cluster per stima contenente Inter e Roma. Il terzo cluster per stima contiene solo il Bologna, anche il quarto cluster per stima contiene solo una squadra, la Fiorentina. Infine, il quinto cluster per stima contiene l'Udinese che ha un valore positivo prossimo a zero.

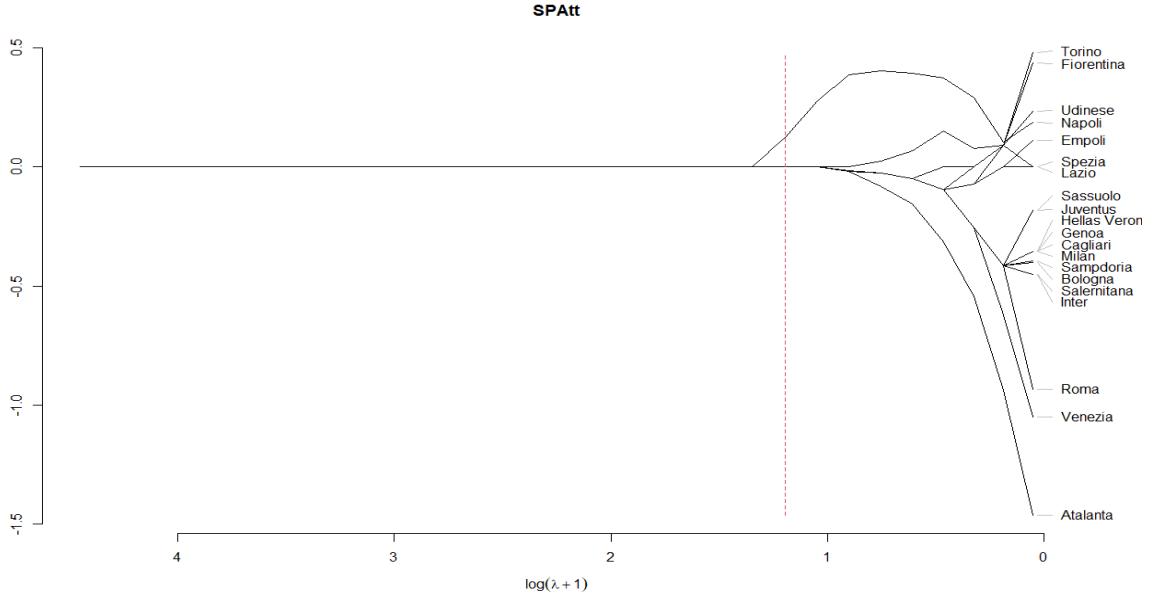


Figura 5.7: Grafico che riporta l'andamento stimato dal modello (4.12) della stima del numero di passaggi corti tentati per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

In Figura 5.7 viene mostrato l'andamento relativo alla stima della covariata del numero di passaggi corti tentati SPAtt. Si nota che il Napoli ha un andamento positivo che si discosta nettamente dall'andamento nullo tenuto dalla maggior parte delle squadre. In Figura 5.8 viene mostrato l'andamento relativo alla stima della covariata della percentuale di passaggi corti riusciti SPCmp%. Si nota che il Genoa ha un andamento negativo che si discosta nettamente dall'andamento leggermente positivo tenuto dalla maggior parte delle squadre.

In Figura 5.9 viene mostrato l'andamento relativo alla stima della covariata della percentuale di passaggi medi riusciti MPCmp%. Si nota che il Genoa e il Bologna hanno un andamento leggermente più negativo rispetto all'andamento, comunque, negativo tenuto dalla maggior parte delle squadre.

In Figura 5.10 viene mostrato l'andamento relativo alla stima della covariata della percentuale di passaggi lunghi riusciti LPCmp%. Sono presenti tre clusters. Il primo contenente solo l'Hellas Verona con un percorso positivo, il secondo che è il cluster più grande contiene quasi tutte le squadre ha un andamento nullo. Infine, il terzo cluster con il Bologna che presenta un andamento negativo.

In Figura 5.11 viene mostrato l'andamento relativo alla stima della covariata del numero di tocchi fatti nell'area di rigore avversari ToAttPen. Si nota che l'Atalanta ha

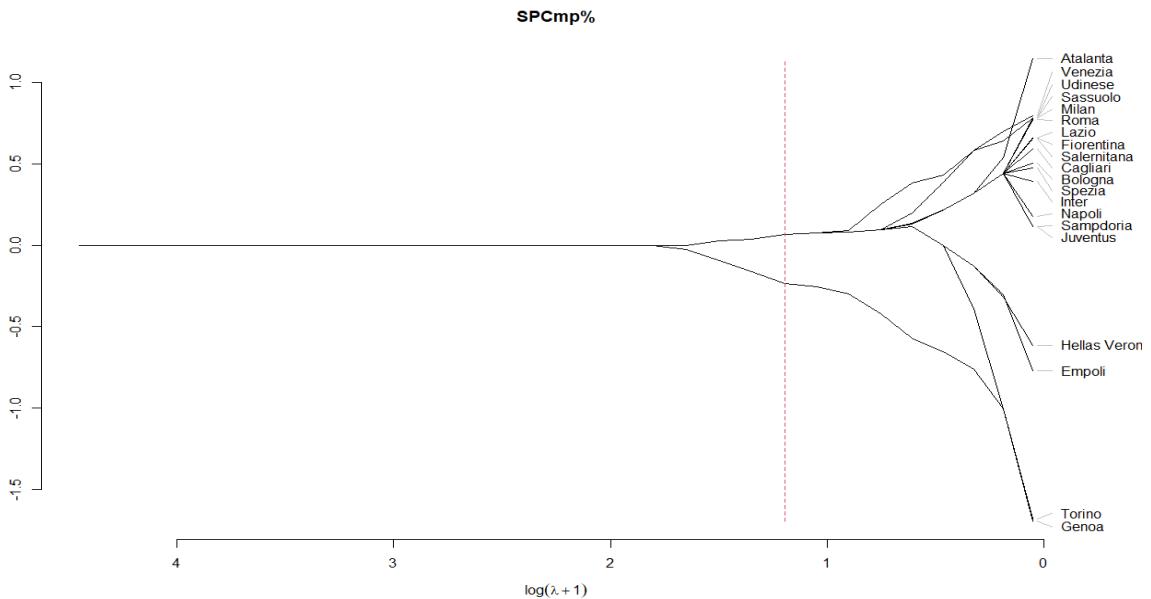


Figura 5.8: Grafico che riporta l'andamento stimato dal modello (4.12) della stima della percentuale di passaggi corti riusciti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

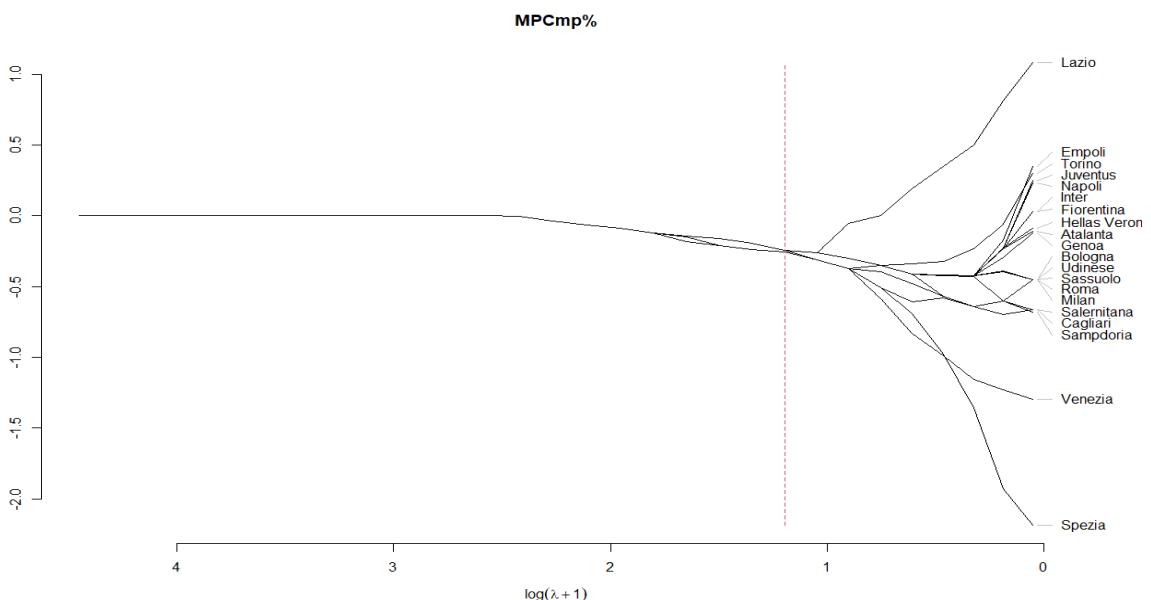


Figura 5.9: Grafico che riporta l'andamento stimato dal modello (4.12) della stima della percentuale di passaggi medi riusciti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

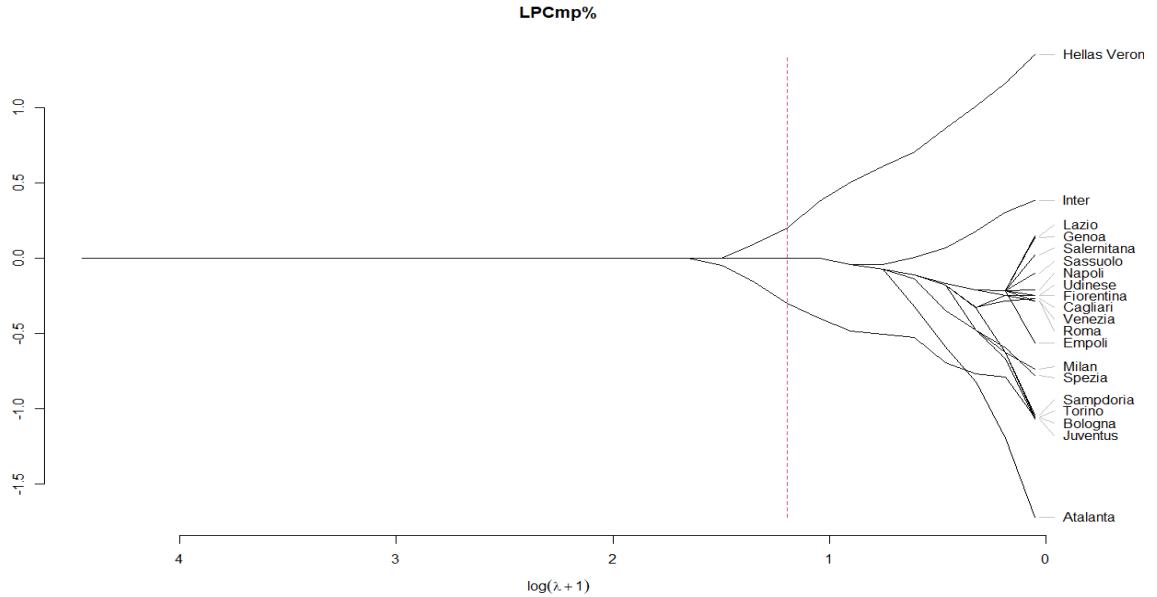


Figura 5.10: Grafico che riporta l'andamento stimato dal modello (4.12) della stima della percentuale di passaggi lunghi riusciti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

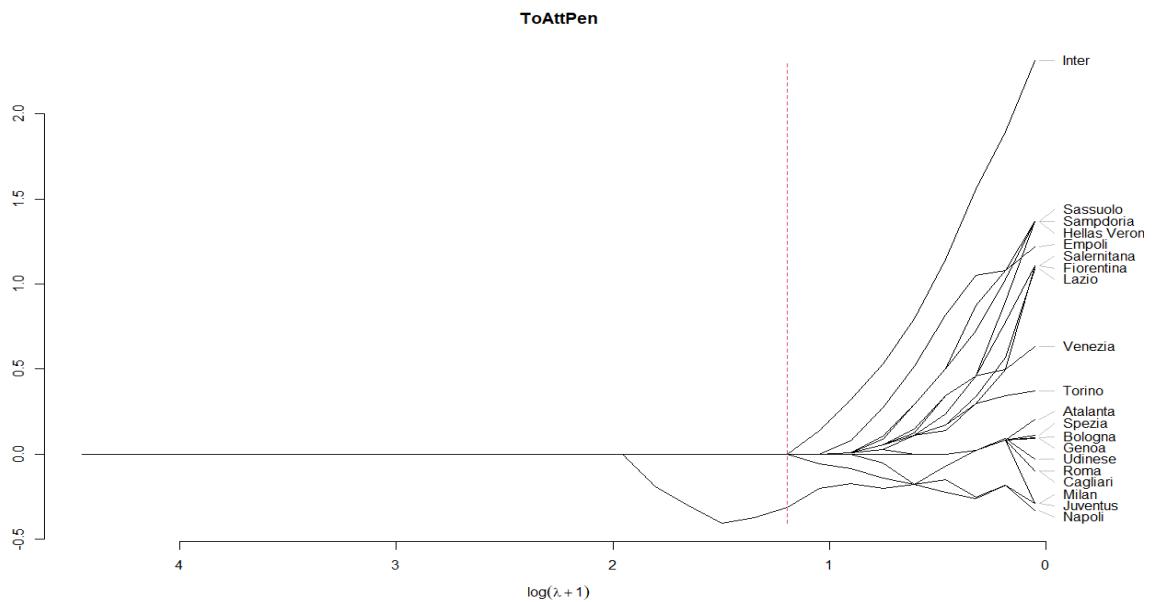


Figura 5.11: Grafico che riporta l'andamento stimato dal modello (4.12) della stima del numero di tocchi fatti nell'area di rigore avversaria per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

un andamento negativo che si discosta nettamente dall'andamento nullo tenuto dalla maggior parte delle squadre.

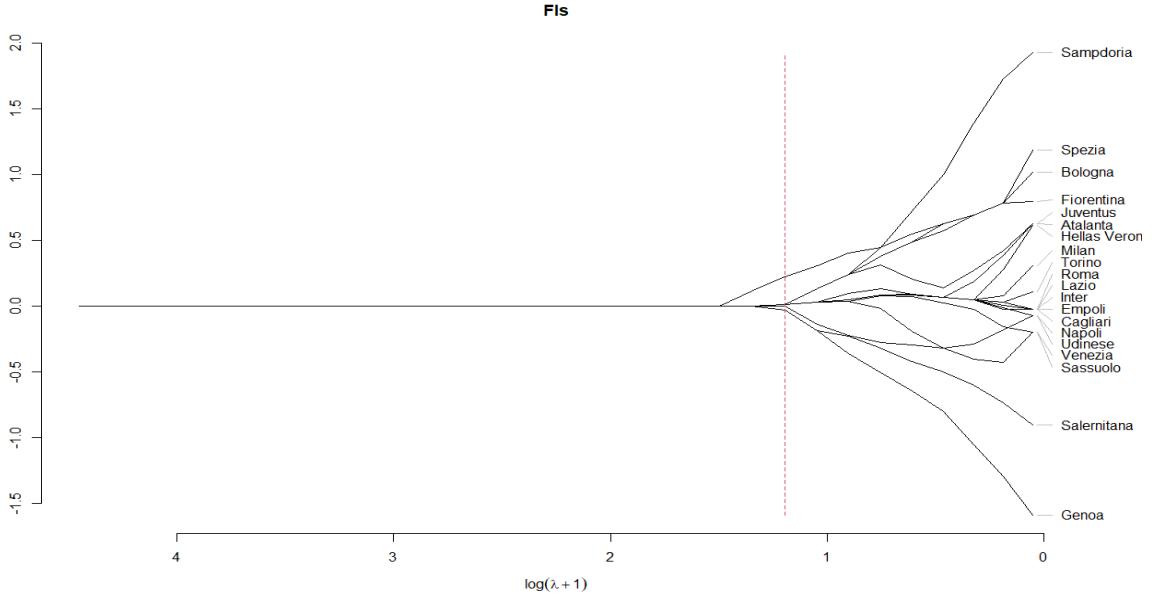


Figura 5.12: Grafico che riporta l'andamento stimato dal modello (4.12) della stima del numero di falli subiti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

In Figura 5.12 viene mostrato l'andamento relativo alla stima della covariata del numero di falli subiti F_{1s} , in cui si notano quattro clusters. Il primo cluster contenente il Bologna che con un percorso positivo è seguito dal cluster più grande che contiene quasi tutte le squadre il quale ha un andamento leggermente positivo. Invece, il cluster che contiene il Napoli ha un andamento leggermente negativo, mentre ancora più negativo è il cluster contenente Genoa e Salernitana.

In Figura 5.13 viene mostrato l'andamento relativo alla stima della covariata del numero di falli fatti, in cui si notano tre clusters. C'è il cluster contenente lo Spezia che ha un percorso positivo, il cluster più grande che contiene quasi tutte le squadre che ha un andamento leggermente positivo. Invece il cluster che contiene l'Udinese ha un andamento leggermente negativo.

In Figura 5.14 viene mostrato l'andamento relativo alla stima della covariata del numero di fuorigioco fatti Off . Ci sono tre clusters. Il primo cluster contenente l'Hellas Verona ha un percorso positivo. Il secondo cluster è il cluster più grande dato che contiene quasi tutte le squadre e ha un andamento leggermente positivo. Infine, il cluster terzo che contiene Milan, Inter, Napoli e Juventus ha un andamento negativo.

In Figura 5.15 viene mostrato l'andamento relativo alla stima della covariata del numero di cross fatti Crs . Ci sono sei clusters. Il primo cluster contenente il Torino ha un andamento nullo. Il secondo cluster è il cluster più grande dato che contiene quasi tutte le squadre e ha un percorso negativo. Ancora più negativi sono i percorsi dei clusters contenenti rispettivamente Roma e Milan, secondi solo ai clusters contenenti Napoli e Atalanta che si discostano nettamente da tutti gli altri clusters.

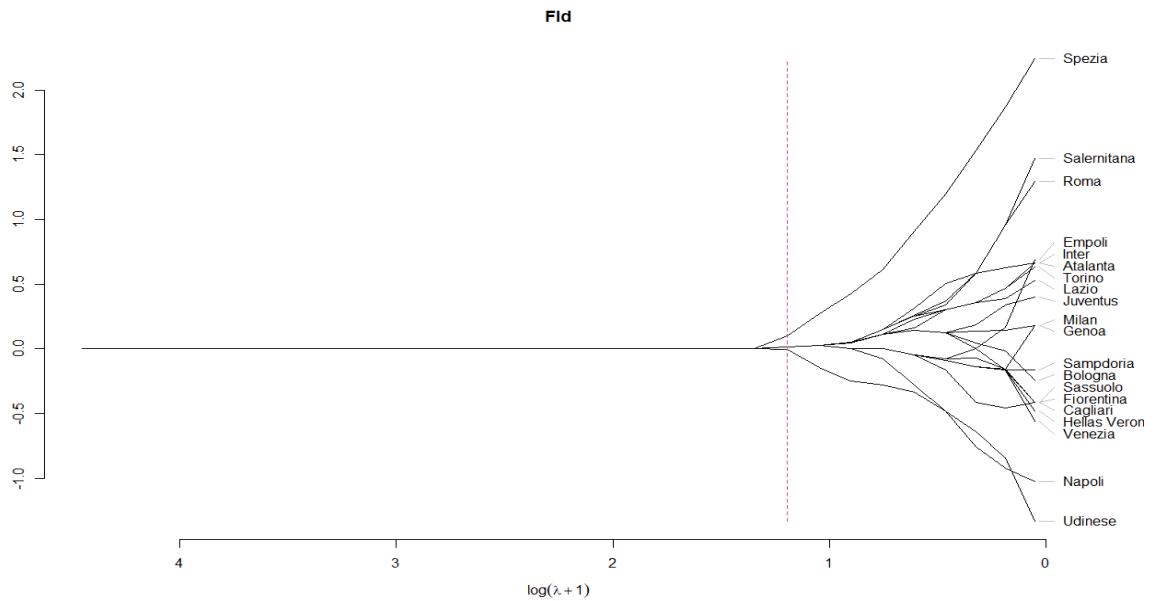


Figura 5.13: Grafico che riporta l'andamento stimato dal modello (4.12) della stima del numero di falli fatti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

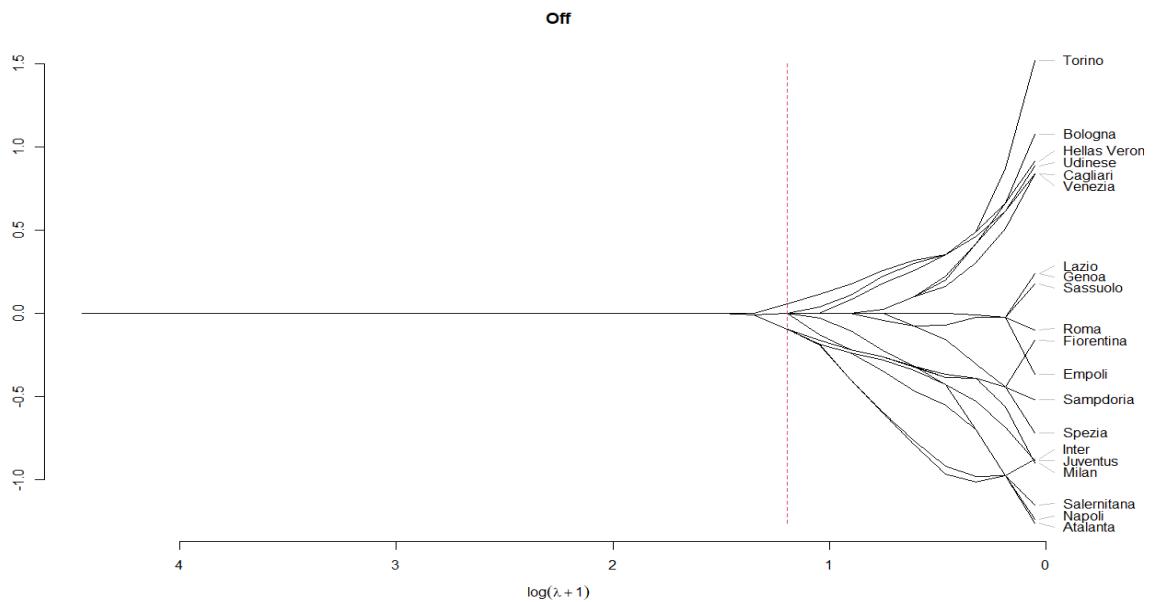


Figura 5.14: Grafico che riporta l'andamento stimato dal modello (4.12) della stima del numero di fuorigioco fatti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

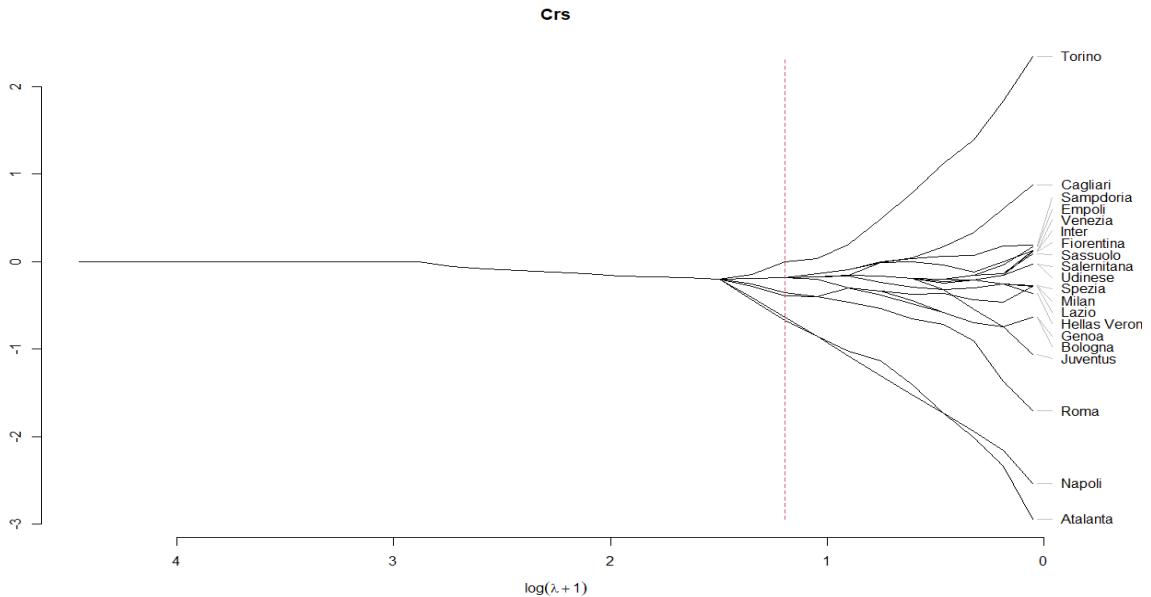


Figura 5.15: Grafico che riporta l'andamento stimato dal modello (4.12) della stima del numero di cross fatti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

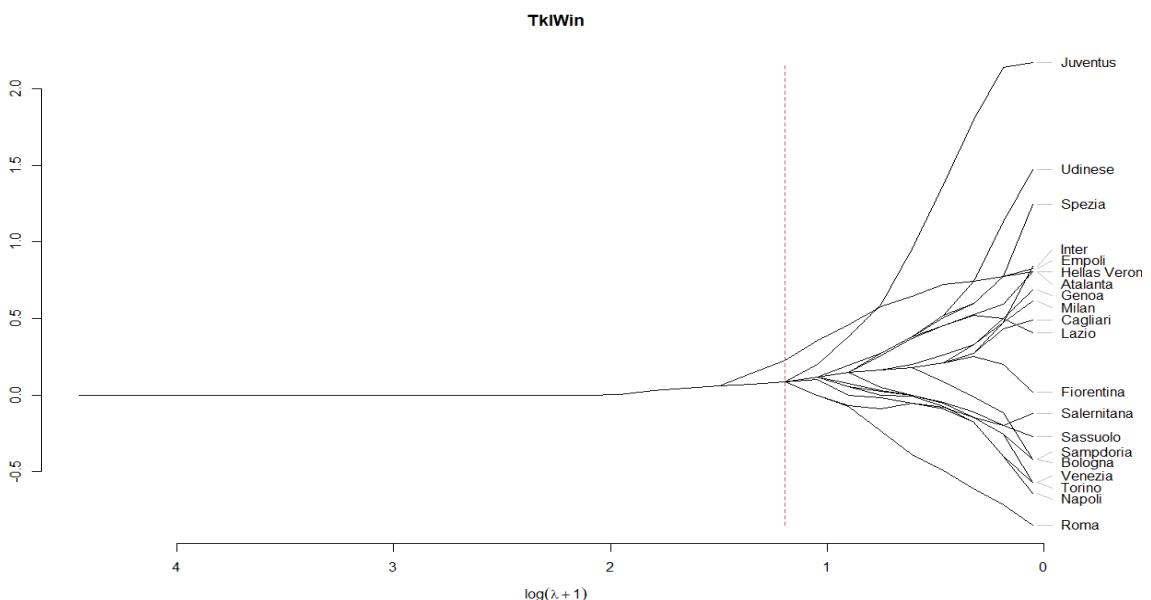


Figura 5.16: Grafico che riporta l'andamento stimato dal modello (4.12) della stima del numero di contrasti vinti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

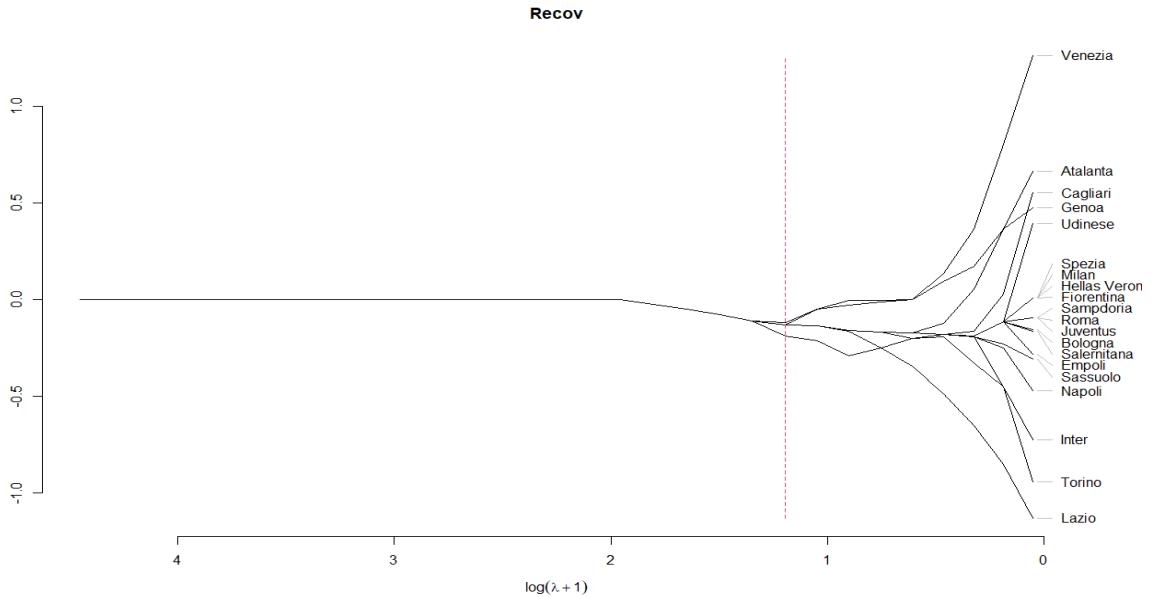


Figura 5.17: Grafico che riporta l’andamento stimato dal modello (4.12) della stima del numero di recuperi per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

In Figura 5.16 viene mostrato l’andamento relativo alla stima della covariata del numero di contrasti vinti **TklWin**, in cui si nota che l’Empoli ha un percorso positivo che si discosta nettamente dall’andamento comunque positivo ma in minor misura tenuto dalla maggior parte delle squadre.

In Figura 5.17 viene mostrato l’andamento relativo alla stima della covariata del numero di recuperi **Recov**. Si nota che l’Udinese ha un percorso leggermente più negativo rispetto all’andamento negativo tenuto dalla maggior parte delle squadre. Viceversa, Genoa che ha un andamento meno negativo rispetto a tutte le altre squadre.

Per riassumere quanto visto finora, la Figura 5.18 mostra i percorsi delle norme L2 che rappresentano l’importanza complessiva dei singoli effetti delle covariate.

Dal grafico si nota che il rapporto gol/tiri **G/Sh** è la variabile che incide maggiormente nella determinazione dell’esito di una partita. Analogamente, il numero di tiri in porta **SoT** e il numero di parate **Saves** sono determinanti sull’esito di una partita, ma con un minor peso rispetto a **G/Sh**. Anche il numero di cross **Crs** è significativamente associato all’esito della partita, ma al contrario di **G/Sh** sappiamo che contribuisce a diminuire le probabilità di vittoria. Si nota anche qui che il possesso della palla ha un ruolo molto marginale nel determinare il risultato di una partita. Viene confermata la tendenza che mantenere il pallone in zone difensive con meno transizioni in zone d’attacco sembra che si associa maggior probabilità di vittoria. Si riconferma importante il numero di lanci lunghi tentati **LPAtt** per l’esito favorevole della partita. Dal grafico si nota che sia il numero di contrasti vinti **TklWin** e il numero di fuorigioco **Off** sono determinanti per l’ottenimento della vittoria. Diversamente, il numero di recuperi **Recov**, la distanza percorsa con la palla **TotDist** e il numero di intercetti **Int** sono poco significativi.

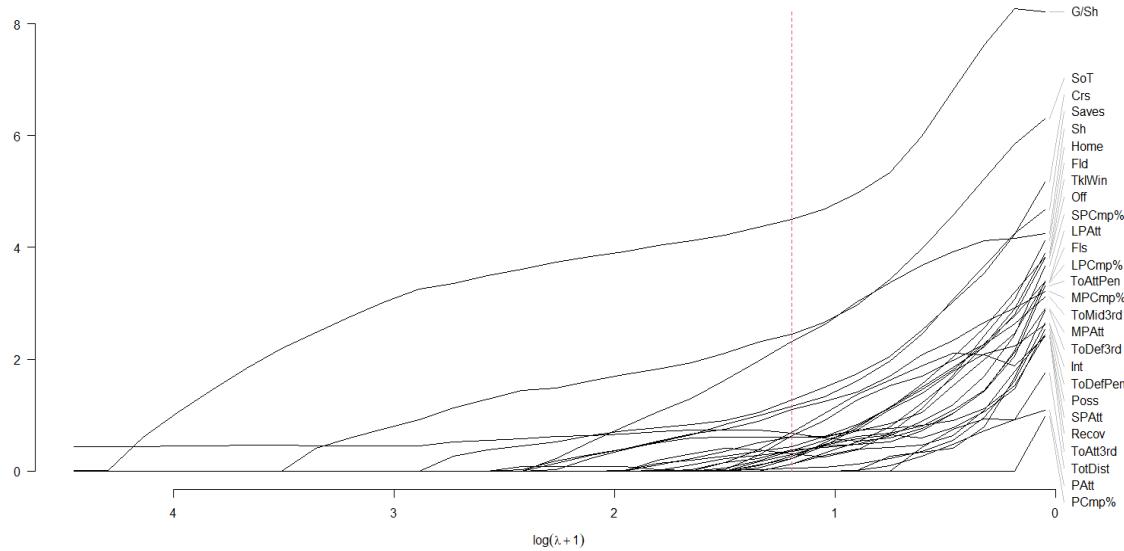


Figura 5.18: Grafico che riporta l'importanza delle covariate rispetto alle norme L2 al variare del parametro di tuning λ secondo le stime del modello (4.12). La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

Infine, notiamo che il numero di falli fatti `f1d` è più determinante di quelli subiti `f1s`. Perciò, la maggior parte delle stime ottenute sembrano essere in linea con i risultati osservati nel precedente modello con covariate specifiche dell'oggetto senza l'applicazione del metodo *LASSO*.

5.5 BTM senza l'intercetta e con LASSO

Come era stato accennato nel Capitolo 4, l'intercetta spiega la maggior parte dell'abilità relativa alla squadra. Per cui le covariate possono essere viste come estensioni contenenti effetti aggiuntivi dell'abilità della squadra che non sono spiegati dall'intercetta. In tal senso, gli effetti della covariata possono aiutare a spiegare i risultati imprevisti di una partita. Nelle tre precedenti applicazione del modello Bradley-Terry è sempre stata inserita un intercetta per ogni squadra. Perciò, di seguito verranno mostrati i risultati relativi a un modello Bradley-Terry della stessa forma del modello (4.12) ma senza le intercette, con lo scopo di capire quale sia l'effetto che ogni variabile esplicativa sull'abilità della squadra senza l'interferenza dell'intercetta. Ovviamente dato il numero elevato di covariate è stata applicata una selezione attraverso il metodo *LASSO*. Il modello applicato è il seguente

$$P(Y_{p(i,j)} \leq k) = \frac{\exp(\delta_i + \theta_k + x_{pi}^T \eta_i - x_{pj}^T \eta_j)}{1 + \exp(\delta_i + \theta_k + x_{pi}^T \eta_i - x_{pj}^T \eta_j)}. \quad (5.2)$$

Nella Tabella 5.4 e nella Tabella 5.5 vengono riportati i risultati nella stessa modalità utilizzata nella precedente sezione.

Covariata	Stima	Squadra
Home	0.270	Tutti
Poss	0.299	Lazio
Poss	0.047	Tutti tranne Lazio
Sh	0.317	Tutti
SoT	0.495	Atalanta, Cagliari, Empoli, Genoa, Verona, Juventus, Lazio, Milan, Napoli, Roma, Salernitana, Sampdoria, Sassuolo, Spezia, Torino e Venezia
SoT	0.438	Inter
SoT	0.399	Bologna, Fiorentina e Udinese
G/Sh	0.867	Tutti
Saves	0.242	Tutti
PAtt	0.000	Tutti
PCmp%	0.000	Tutti
SPAtt	0.000	Tutti
SPCmp%	0.000	Tutti tranne Genoa
SPCmp%	-0.076	Genoa
MPAtt	0.000	Tutti
MPCmp%	-0.230	Udinese
MPCmp%	-0.236	Tutti tranne Udinese
LPAtt	0.178	Tutti
LPCmp%	0.016	Hellas Verona
LPCmp%	0.000	Tutti tranne Hellas Verona
ToDefPen	0.080	Tutti
ToDef3rd	0.024	Tutti

Tabella 5.4: Stime delle covariate stimate dal modello (5.2).

Covariata	Stima	Squadra
ToMid3rd	0.002	Tutti tranne Inter e Sampdoria
ToMid3rd	0.000	Inter e Sampdoria
ToAtt3rd	-0.013	Tutti
ToAttPen	0.035	Tutti tranne Atalanta
ToAttPen	-0.083	Atalanta
TotDist	0.000	Tutti
Fls	0.256	Bologna
Fls	0.088	Tutti tranne Bologna
Fld	0.066	Tutti tranne Udinese
Fld	0.023	Udinese
Off	0.000	Tutti tranne Juventus
Off	-0.085	Juventus
Crs	-0.190	Tutti tranne Atalanta
Crs	-0.464	Atalanta
Int	0.000	Tutti
TklWin	0.117	Empoli
TklWin	0.000	Tutti tranne Empoli
Recov	0.000	Tutti

Tabella 5.5: Stime delle covariate stimate dal modello (5.2).

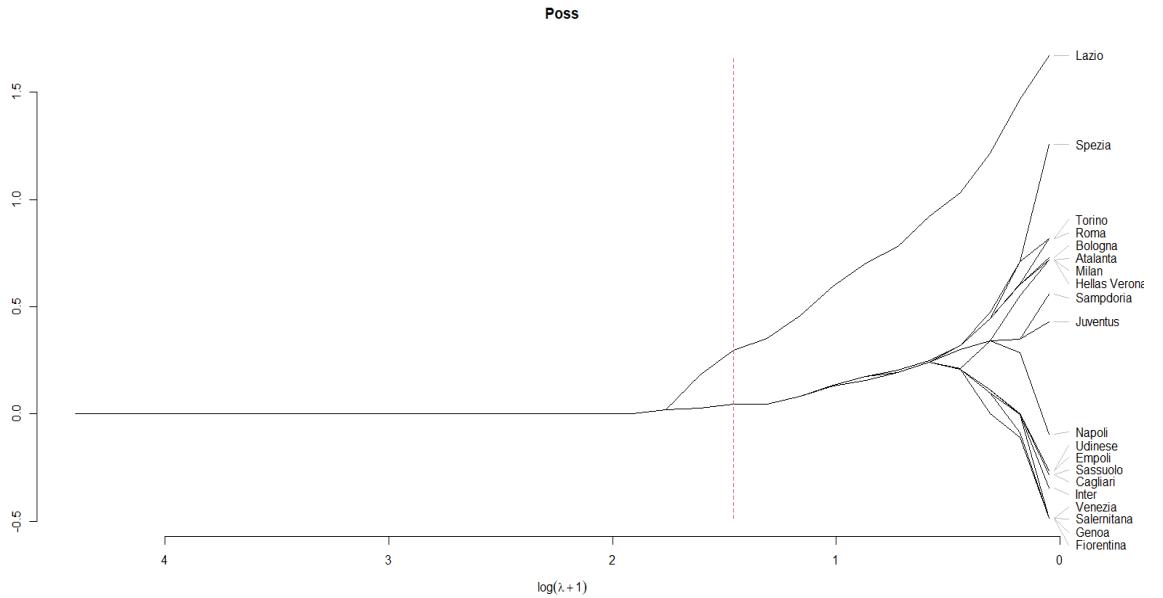


Figura 5.19: Grafico che riporta l'andamento stimato dal modello (5.2) della stima del possesso della palla per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

Anche in questa applicazione sono state eliminate alcune covariate. Come già visto nel modello precedente, vengono confermate l'eliminazione del numero di passaggi tentati **PAtt**, della percentuale dei passaggi completati **PCmp%** e della distanza percorsa con la palla **TotDist**. Viene tolta dal modello la variabile esplicativa del numero di passaggi corti tentati **SPAtt** la quale nel modello precedente andava ad aumentare le probabilità di vittoria solo per il Napoli. Viene eliminata la covariata del numero di passaggi medi tentati **MPAtt** e quella del numero di intercettazioni **Int**. Infine, si rileva l'eliminazione della variabile esplicativa che indica il numero di recuperi **Recov** che nel precedente modello era valutata come una covariata che incideva negativamente sulla probabilità di vittoria.

In questa nuovo tipo di modello la stima del parametro del possesso palla **Poss** ha subito una piccola variazione. Infatti, ora la stima non è più nulla per la maggior parte delle squadre ma è leggermente positiva. Ciononostante, la significatività si riconferma ancora bassa. Si riconferma però significativa solo per il gioco della Lazio, ma non più per il Torino come nello scorso modello. Tale risultato è visibile nella Figura 5.19.

Il numero di tiri **Sh**, il rapporto gol/tiri **G/Sh** e il numero di parate **Saves** sono ancora determinati per aumentare la probabilità di vittoria. Analogamente anche il numero di tiri in porta **SoT** mantiene una stima positiva del parametro, con una variabilità più ristretta rispetto al modello precedente. Infatti nella Figura 5.20 è possibile individuare tre clusters. Il più grande con la maggiore stima contiene la maggior parte delle squadre. Il secondo contiene solo l'Inter e infine il terzo contiene Bologna, Fiorentina e Udinese. Il risultato è visibile nella Figura 5.20.

Nella percentuale di passaggi corti riusciti **SPCmp%** ora viene a crearsi un cluster con un percorso nullo contenente quasi tutte le squadre eccetto il Genoa che è contenuto in

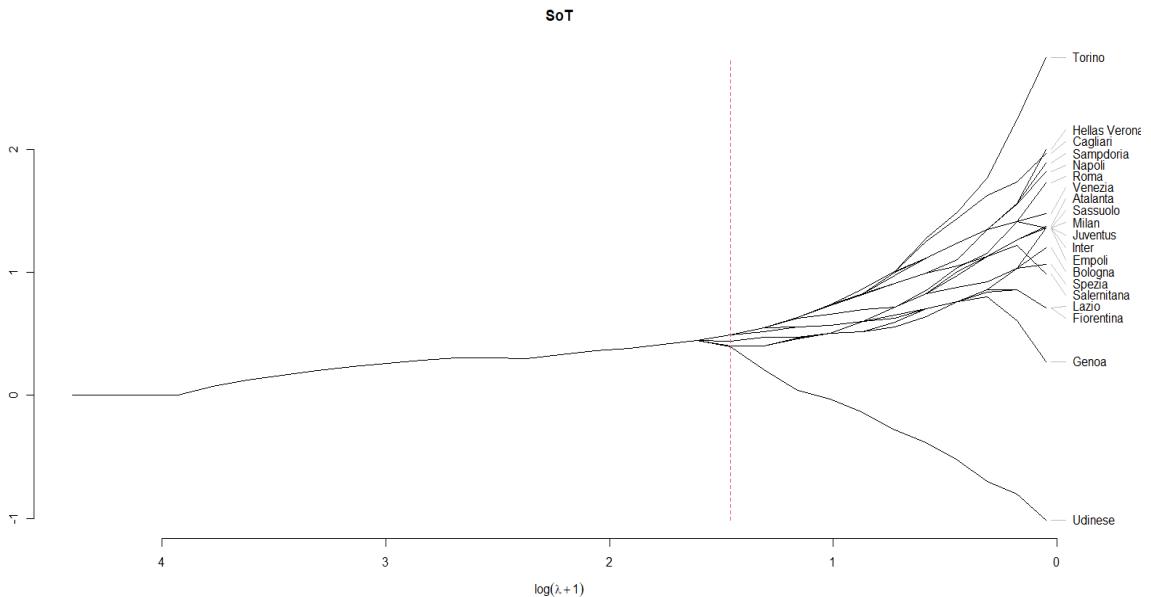


Figura 5.20: Grafico che riporta l'andamento stimato dal modello (5.2) della stima del numero di tiri in porta per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

un cluster con un percorso negativo. Tali risultati sono visibili nella Figura 5.21.

Per la variabile esplicativa della percentuale di passaggi medi riusciti MPCmp% ora viene a crearsi un cluster con un percorso fortemente negativo contenente quasi tutte le squadre eccetto l'Udinese dove si distingue per avere un percorso leggermente meno negativo. Tali risultati sono visibili nella Figura 5.22.

Il numero di passaggi lunghi tentati è ancora una covariata con una stima del parametro che aumenta la probabilità di vittoria. Si nota che la stima della covariata della percentuale di passaggi lunghi riusciti LPCmp% ha un cluster con un percorso nullo contenente quasi tutte le squadre eccetto l'Hellas Verona, che si distingue per un percorso positivo. Tali risultati sono visibili nella Figura 5.23.

Sia il numero di tocchi fatti in area di rigore ToDefPen sia il numero di tocchi fatti nella trequarti di difesa ToDef3rd sono associate ad un aumento della probabilità di vittoria. Nella stima del parametro della covariata che indica il numero di tocchi fatti a centrocampo ToMid3rd, viene a crearsi un cluster con un percorso poco significativo contenente quasi tutte le squadre eccetto Inter e Sampdoria, le quali formano un cluster con un percorso non significante.

Il numero di tocchi fatti nella trequarti offensiva ToAtt3rd si conferma essere associato ad una riduzione della probabilità di vittoria.

Per la stima della variabile esplicativa che indica il numero di tocchi fatti nell'area di rigore avversaria ToAttPen, c'è un cluster con un percorso negativo contenente quasi tutte le squadre eccetto l'Atalanta che si distingue per un percorso positivo. Tali risultati sono visibili nella Figura 5.24.

Per quanto riguarda l'aggressività della squadra, il numero di falli fatti Fld si associa ad un aumento della probabilità di vittoria per tutte le squadre. Come mostrato della

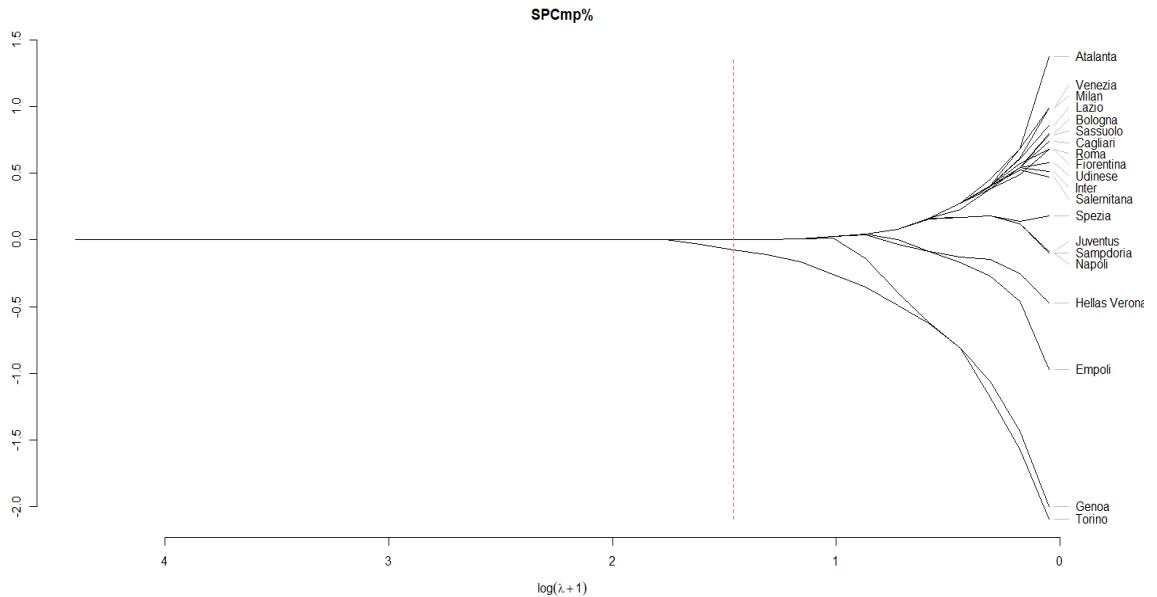


Figura 5.21: Grafico che riporta l'andamento stimato dal modello (5.2) della stima della percentuale di passaggi corti riusciti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

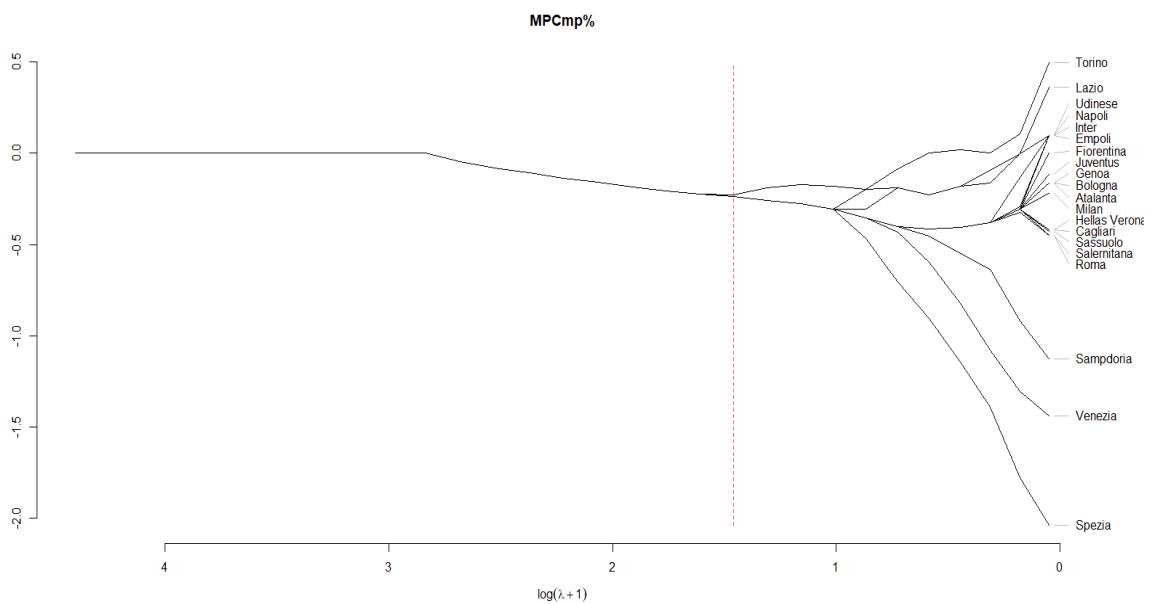


Figura 5.22: Grafico che riporta l'andamento stimato dal modello (5.2) della stima della percentuale di passaggi medi riusciti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

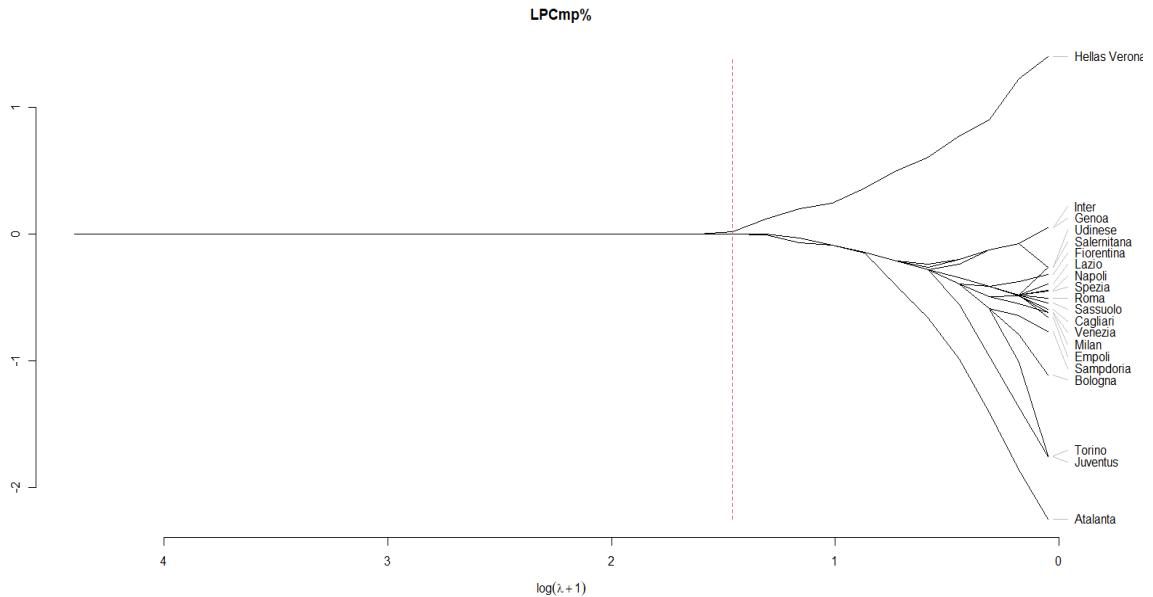


Figura 5.23: Grafico che riporta l'andamento stimato dal modello (5.2) della stima della percentuale di passaggi lunghi riusciti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

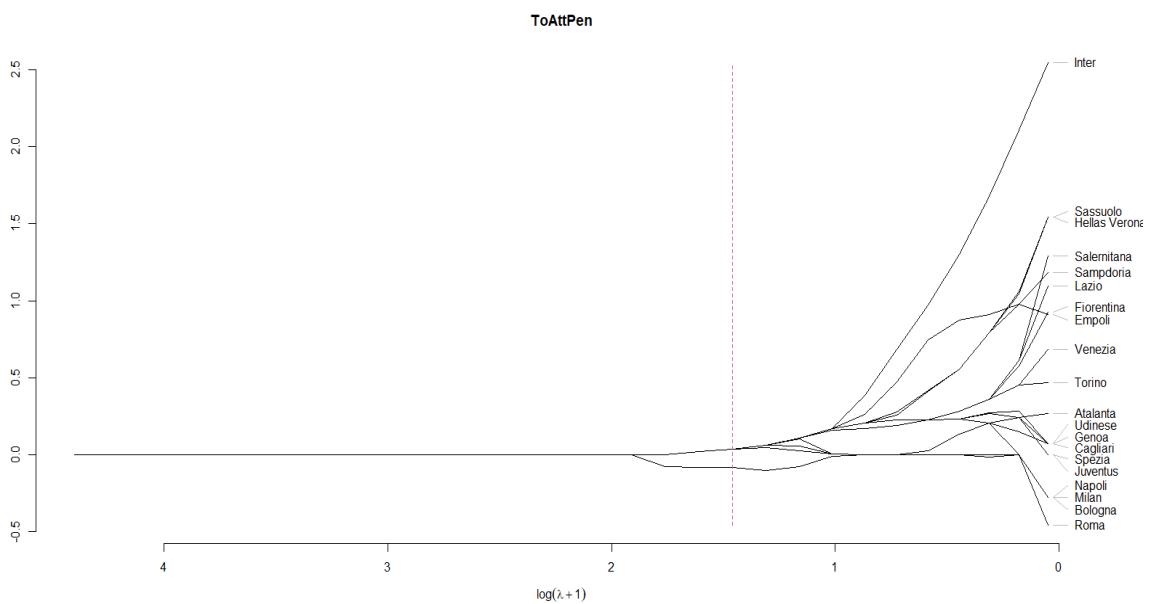


Figura 5.24: Grafico che riporta l'andamento stimato dal modello (5.2) della stima del numero di tocchi fatti nell'area di rigore avversaria per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

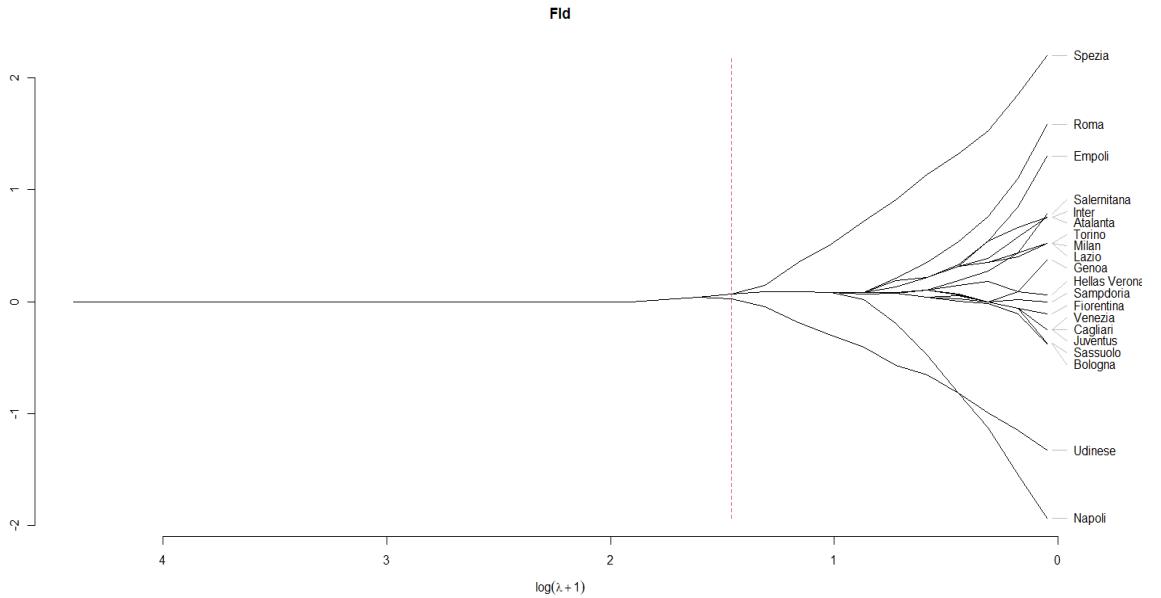


Figura 5.25: Grafico che riporta l'andamento stimato dal modello (5.2) della stima del numero di falli fatti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

Figura 5.25 però, la stima per l'Udinese è minore rispetto a tutte le altre squadre. Analogamente anche il numero di falli subiti F1s si associa ad una aumento della probabilità di vittoria di tutte le squadre, in particolare il Bologna si distingue con una stima maggiore come mostrato nella Figura 5.26. Nella stima della covariata che indica il numero di fuorigioco fatti Off, viene a crearsi un cluster con un percorso nullo contenente quasi tutte le squadre eccetto la Juventus, la quale forma un cluster con un percorso negativo. Tali risultati sono visibili nella Figura 5.27. La stima della variabile esplicativa del numero di cross fatti Crs si conferma essere ancora determinante per diminuire la probabilità di vittoria. L'Atalanta inoltre si distingue dalle altre squadre con un percorso ancora più negativo rispetto, come mostrato nella Figura 5.28. Si nota che nella stima della covariata che indica il numero di contrasti vinti TklWin, viene a crearsi un cluster con un percorso nullo contenente quasi tutte le squadre eccetto l'Empoli, il quale forma un cluster con un percorso positivo. Tali risultati sono visibili nella Figura 5.29. Infine, viene confermato che giocare le partite in casa Home ha un effetto positivo stimato in 0.270, mentre è cambiata la stima delle soglie θ_1 e θ_2 che valgono rispettivamente -0.803 e 0.803 .

Nella Figura 5.30 viene mostrato l'andamento delle prestazioni del modello su tutti i valori assunti dal parametro di tuning λ durante l'operazione di K-Fold Cross Validation. Anche qui, la K-Fold Cross Validation utilizzata prevedeva l'uso di 10 gruppi ($k = 10$) e di trenta valori diversi per λ . Successivamente, i risultati ottenuti dal modello applicando i trenta diversi valori del parametro di tuning, sono stati confrontati in termini di RPS. Come indicato dal grafico con la linea tratteggiata in rosso, il parametro di tuning ottimo λ è pari a 1.458.

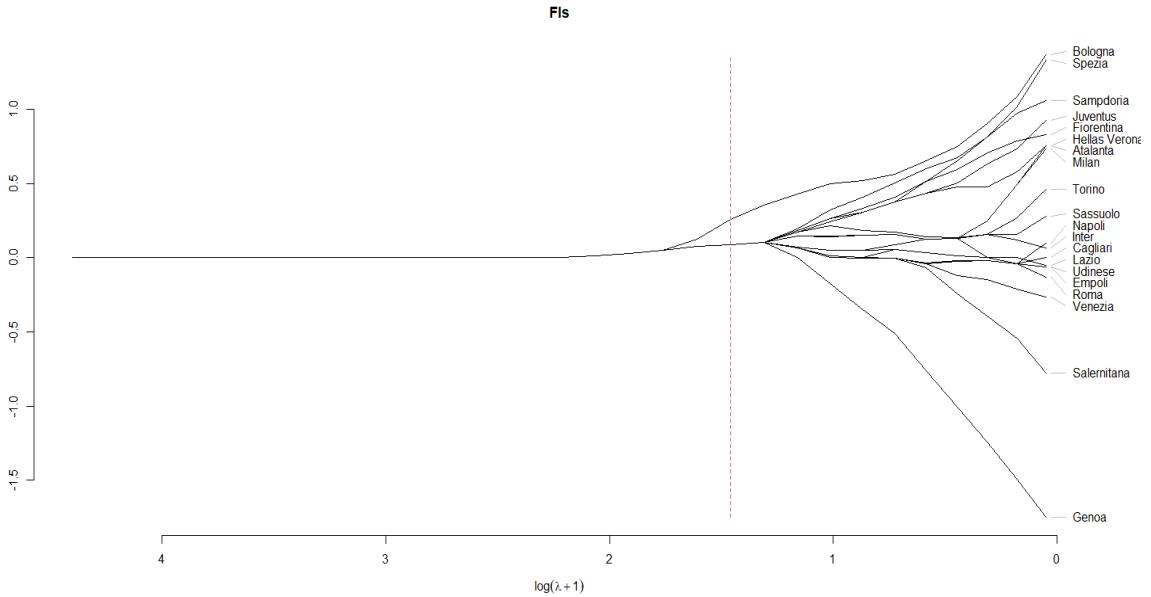


Figura 5.26: Grafico che riporta l'andamento stimato dal modello (5.2) della stima del numero di falli subiti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

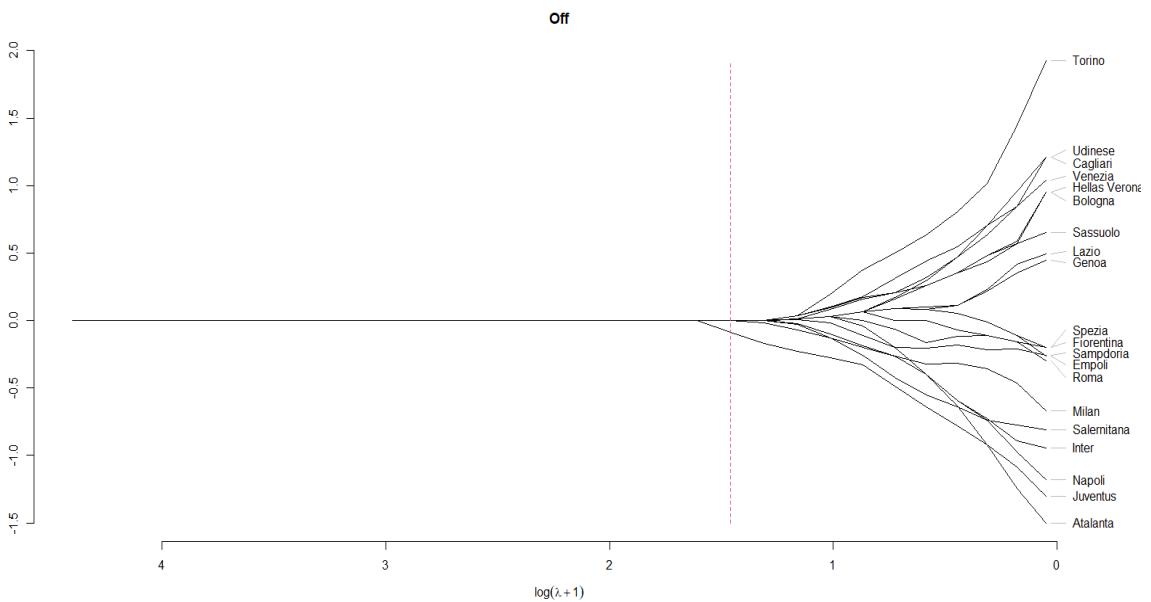


Figura 5.27: Grafico che riporta l'andamento stimato dal modello (5.2) della stima del numero di fuorigioco fatti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

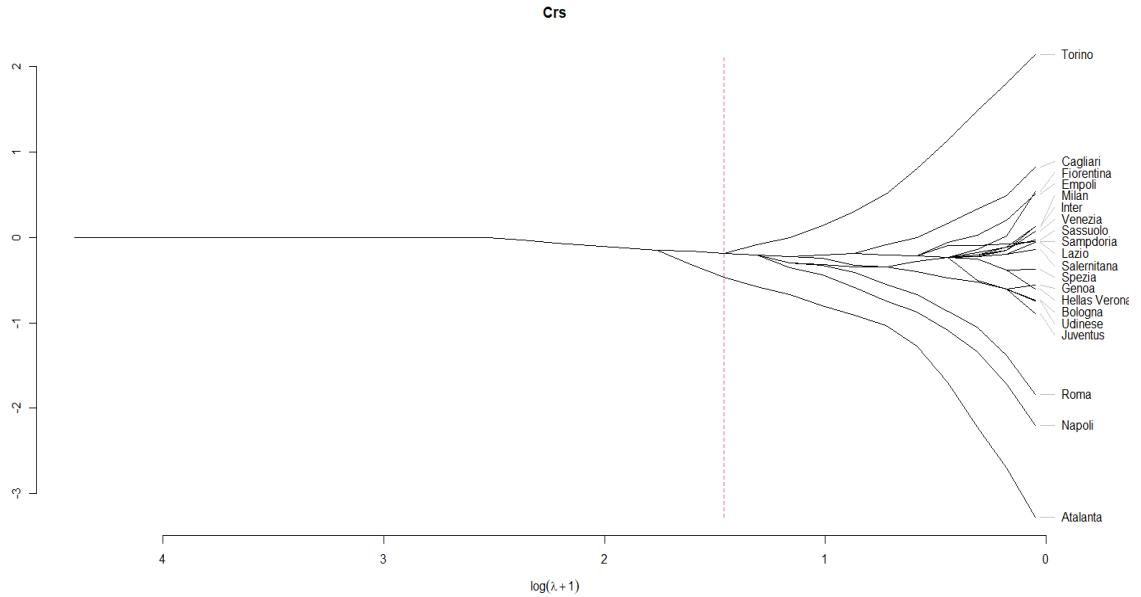


Figura 5.28: Grafico che riporta l’andamento stimato dal modello (5.2) della stima del numero di cross fatti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

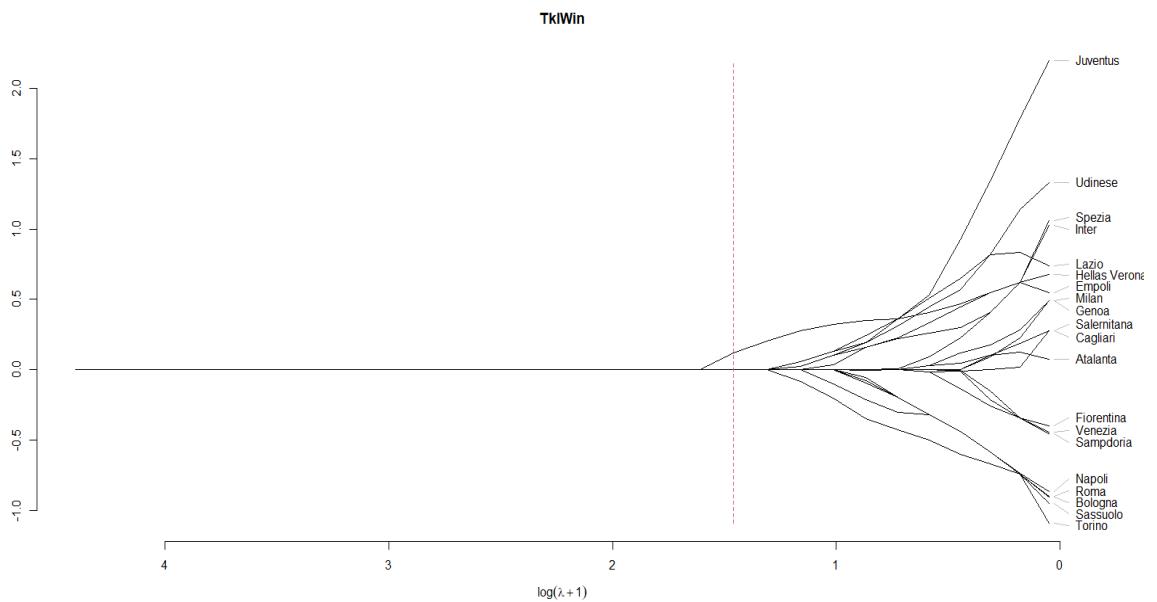


Figura 5.29: Grafico che riporta l’andamento stimato dal modello (5.2) della stima del numero di contrasti vinti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

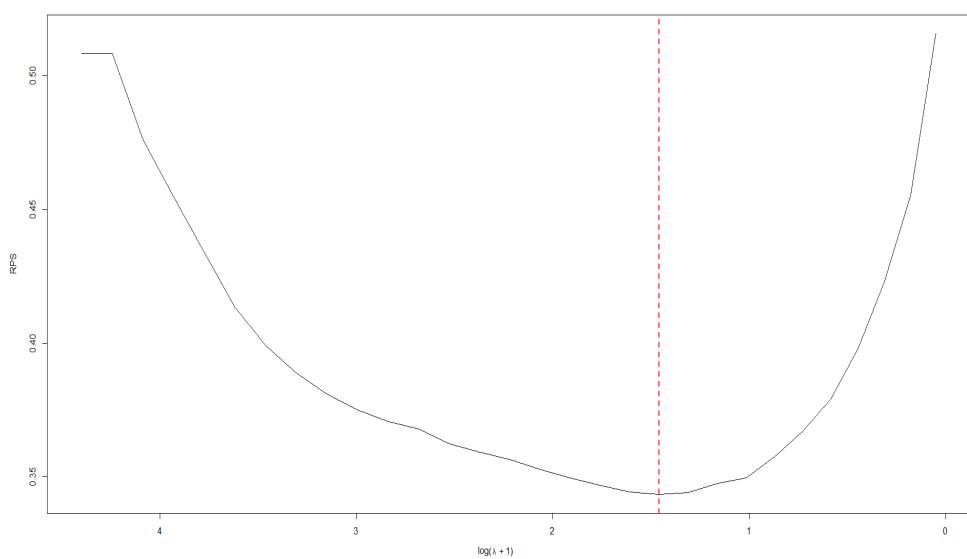


Figura 5.30: Grafico dell'andamento delle prestazioni del modello (5.2) su tutti i trenta valori assunti dal parametro di tuning, indicato con il simbolo λ , durante l'applicazione di K-Fold Cross Validation. L'andamento viene valutato in termini di Ranked Probability Score (RPS). La linea rossa tratteggiata indica il parametro di tuning λ ottimo da utilizzare.

Un’ulteriore analisi che può essere condotta è la valutazione dell’effetto medio dei valori assunti della covariata per ogni partita e per ogni squadra, insieme alle stime dei singoli parametri per squadra. Si utilizzeranno i grafici a *effetto stella* proposti da **tutz2013visualization**. In questi grafici è possibile visualizzare i valori medi per squadra e per covariata moltiplicati per le rispettive stime riportate precedentemente. Quindi verrà illustrato graficamente il contributo medio di una variabile esplicativa sull’abilità di una singola squadra. Il grafico funziona nel seguente modo: esso mostra il prodotto esponenziale tra la media dei valori assunti da una covariate e le sue stime per ogni squadra. Per ogni grafico, viene creato un cerchio con raggio $\exp(0) = 1$ il quale rappresenta il caso con stima nulla. I valori oltre il cerchio indicano che la covariata ha effetto positivo in media sulla squadra. Viceversa, i valori all’interno del cerchio indicano che la variabile esplicativa si associa a effetti negativi in media sulla squadra. Nella Figura 5.31 nella Figura 5.32 e nella Figura 5.33 vengono mostrati i grafici a *effetto stella*. Nella Figura 5.31 si possono vedere tutte le variabili esplicative in cui la maggioranza delle squadre ha ricevuto una stima dei parametri delle variabili esplicative uguale a zero. Si possono comunque notare alcune squadre distinguersi in alcune covariate. Ad esempio la Lazio si differenzia dalle altre squadre con il possesso palla **Poss** mentre l’Empoli con il numero di contrasti vinti **TklWin**.

Per la Figura 5.32 abbiamo due particolari grafici. Entrambi rappresentano l’effetto

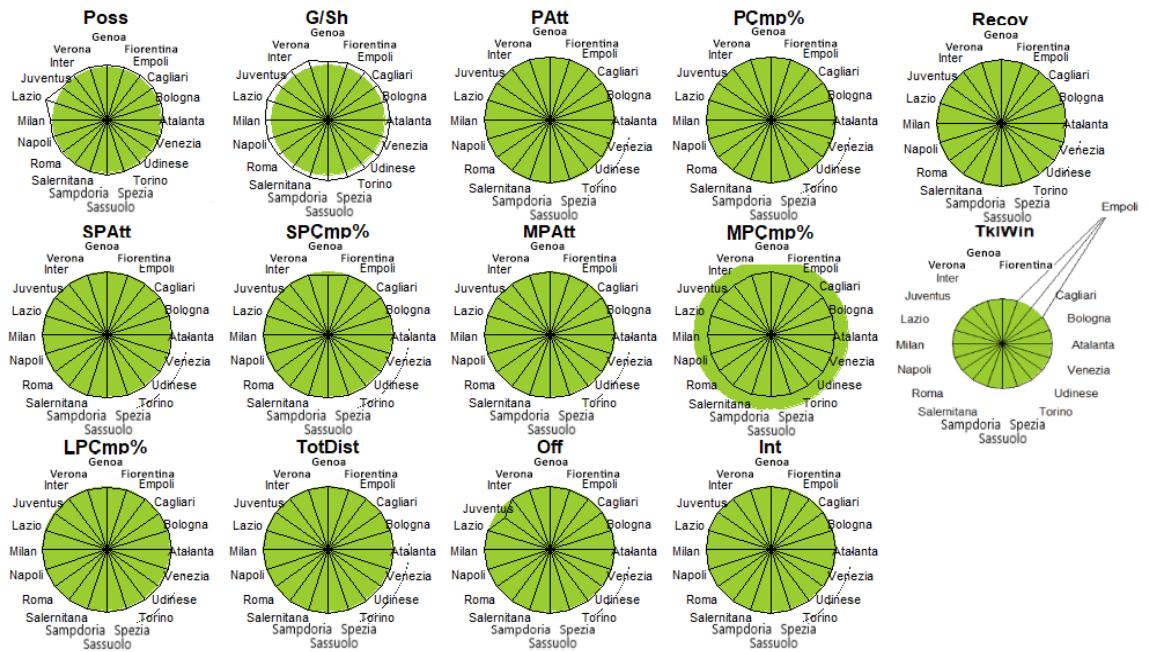


Figura 5.31: Grafico a effetto stella che riporta il contributo medio di una covariata sull’abilità di una singola squadra secondo il modello (5.2).

negativo delle covariate che indicano, rispettivamente, il numero di tocchi nella trequarti avversaria fatti **ToAtt3rd** e il numero di cross fatti **Crs**. Notiamo che a subire più gli effetti negativi sono Inter e Atalanta per entrambe le variabili esplicative. Infine, risultati più interessanti si hanno nella Figura 5.33. Innanzitutto si vede che nel numero di tiri Sh l’Inter ha un grosso beneficio. In minor misura ne beneficiano Milan, Roma,

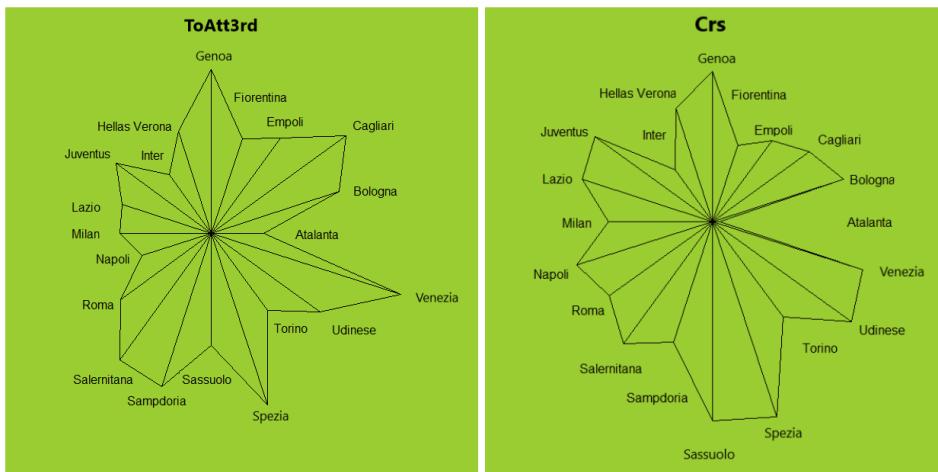


Figura 5.32: Grafico a effetto stella che riporta il contributo medio di una covariata sull'abilità di una singola squadra secondo il modello (5.2).

Atalanta, Napoli, Juventus e Sassuolo. Analoghi risultati sono visibili con il numero di tiri in porta SOT, con l'aggiunta della Lazio tra le squadre che ricevano più benefici. In generale, nel grafico relativo al numero di parate Saves, tutte le squadre ottengo benefici. Stesso risultato si rileva per il numero di passaggi lunghi tentati LPAtt. Nel grafico del numero di tocchi in area di rigore ToDefPen c'è un particolare beneficio ottenuto dal Venezia, così come Inter, Lazio, Empoli e Sassuolo. Analoghi risultati anche per il numero di tocchi nella trequarti difensiva ToDef3rd, ma con la differenza di minori benefici per il Venezia. Pertanto, si nota la tendenza delle squadre italiane ad attuare tattiche che prediligono di giocare nella propria metà campo. I tocchi a centrocampo ToMid3rd vengono in media effettuati molto dalle squadre, tranne per Inter e Sampdoria per le quali l'effetto è nullo. Analogo effetto anche per il numero di tocchi fatti in area di rigore ToAttPen, con l'unica differenza che si registra ora un effetto positivo per Inter e Sampdoria e un effetto negativo solo per Atalanta. In generale i falli subiti F1s portano benefici alle squadre, come si era notato dalle stime del modello. Per i falli fatti, invece abbiamo che l'Udinese ha minori benefici rispetto a tutte le altre squadre.

Dai grafici *effetto stella* emerge la tendenza da parte delle squadre italiane a ricorrere alla cosiddetta **costrdalbasso**. Ossia la costruzione dell'azione partendo dal proprio portiere e l'utilizzo di lanci lunghi. Inoltre, sapendo che ai parametri del numero di tocchi in area di rigore ToDefPen, del numero di tocchi nella trequarti difensiva ToDef3rd, del numero di tocchi a centrocampo ToMid3rd e del numero di passaggi lunghi tentati LPAtt sono associati degli aumenti della probabilità di vittoria, è possibile affermare che la costruzione dal basso offre maggiori probabilità di vittoria.

Infine, come fatto nella sezione precedente, si analizzano i percorsi delle norme L2 che rappresentano l'importanza complessiva dei singoli effetti delle covariate. Tali percorsi sono visibili nella Figura 5.34.

Gli andamenti ottenuti nella Figura 5.34 sono molto simili a quelli visti nella Figura 5.18, con un aumento di importanza per la covariata ToAttPen in termini di diminuzione della probabilità di vittoria. Pertanto, quanto ricavato del modello (4.12) ora trova conferma anche nel modello (5.2).

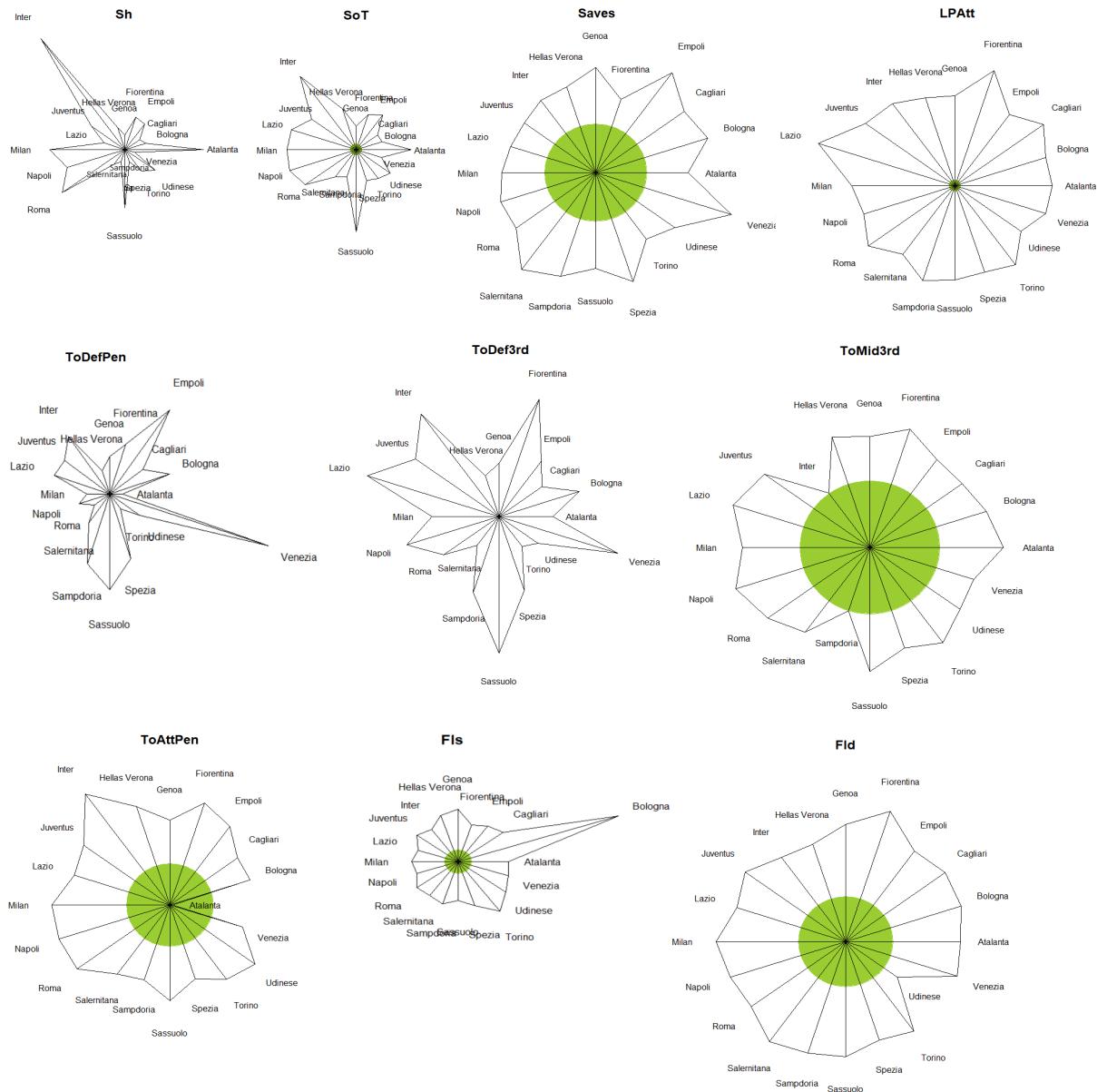


Figura 5.33: Grafico che riporta il contributo medio di una covariata sull'abilità di una singola squadra secondo il modello 5.2.

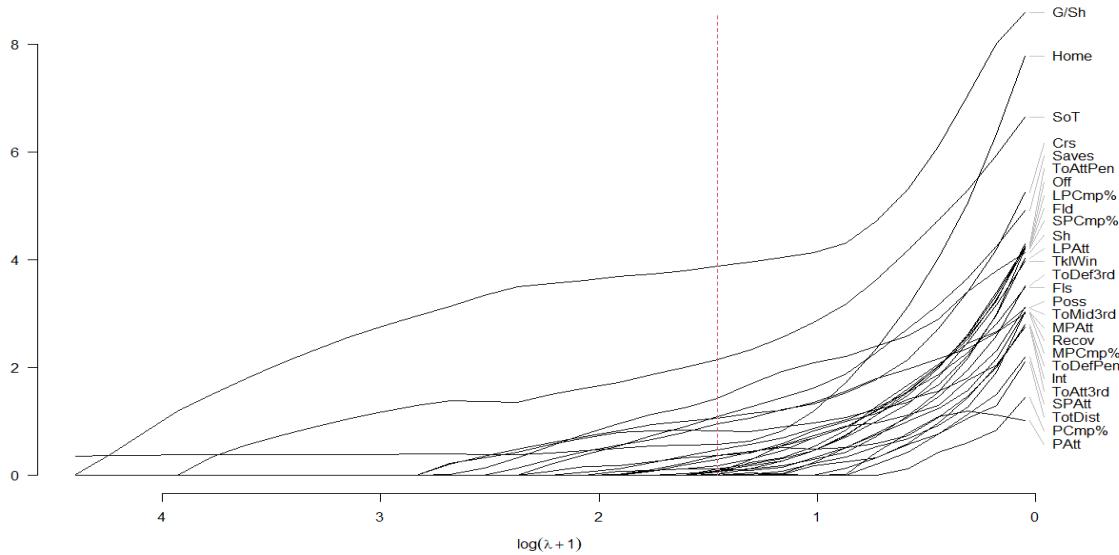


Figura 5.34: Grafico che riporta l'importanza delle covariate rispetto alle norme L2 al variare del parametro di tuning λ

5.6 Predizioni

In questa sezione si vuole valutare le prestazioni dei quattro modelli presentati nelle precedenti sezioni. I modelli vengono valutati in base alle loro capacità di predizioni dell'esito di una partita, sulla base delle informazioni (covariate) disponibili. Per predizione si intende che il modello stabilisce l'esito di una partita senza conoscerne il risultato reale. Per rendere più interessante il confronto si è aggiunto un quinto elemento nel confronto, ossia le predizioni fatte dai *bookmakers*, ad esempio Bet365, William Hill ecc... I dati dei *bookmakers* sono stati presi da **bet**, il quale fornisce la media delle probabilità dei *bookmakers* per ogni risultato, su un gran numero di campionati di calcio, tra cui la Serie A italiana. Si è quindi preso come predizione il risultato più probabile secondo i *bookmakers*.

Le predizioni dei modelli sono state eseguite nel seguente modo: il *dataset* è stato diviso in modo casuale in due parti chiamate solitamente *training set* e *test set*. Il *training set* contiene quasi l'80% delle 38 giornate, ossia 30 giornate per un totale di 300 partite. Invece il *test set* contiene circa il restante 20% ossia 8 giornate per un totale di 80 partite. Il *training set* è utilizzato per stimare i parametri del modello mentre il *test set* è utilizzato per fare predizione. Perciò una parte delle osservazioni è stata utilizzata per allenare il modello, mentre la restante parte per predire l'esito delle restanti osservazioni.

Prima di discutere delle misurazioni e delle predizioni ottenute, è importante tener presente che i modelli (4.1), (5.1), (4.12) e (5.2) utilizzano informazioni e statistiche che sono disponibili solo dopo il termine delle partite, cioè non disponibili per i *bookmakers*. Infatti, i *bookmakers* calcolano le loro predizioni prima che le partite comincino. Certamente i quattro modelli non sono utilizzabili per poter fare predizioni,

ma l'obiettivo del confronto è quello di capire se le informazioni sono state impiegate nel modo opportuno per acquisire maggior conoscenza. Ossia se i modelli ottengono delle prestazioni peggiori rispetto ai *bookmakers*, nonostante abbiano più informazioni sulle partite allora, non sono state utilizzate nel modo corretto le informazioni, viceversa se le prestazioni dei modelli sono migliori rispetto ai *bookmakers* allora le informazioni sono state utilizzate correttamente.

La capacità predittiva di un modello sarà valutata come segue:

- * **Accuratezza.** Indica il rapporto tra il numero di istanze classificate correttamente e il numero totale delle osservazioni in esame.
- * **Sensibilità.** Indica il rapporto tra il numero di istanze identificate correttamente con la categoria k e il numero totale delle osservazioni di categoria k con $k \in \{1, \dots, K\}$.
- * **Precisione.** Misura il grado di correttezza del sistema. Indica il rapporto tra il numero di istanze identificate correttamente con la categoria k e il numero totale delle osservazioni classificate con la categoria k con $k \in \{1, \dots, K\}$.
- * **Specificità.** Indica il rapporto tra il numero di predizioni identificate correttamente con una categoria diversa dalla categoria k e il numero totale delle osservazioni classificate con una categoria diversa dalla categoria k con $k \in \{1, \dots, K\}$.

Nella Figura 5.35 sono mostrate le classificazioni ottenute sulle 80 partite del *test set* per ogni modello e per la predizione dei *bookmakers*. Dai risultati ottenuti, l'accuratezza

Modello (4.1)				Modello (5.1)				Modello (4.12)				Modello (5.2)				Bookmakers			
True	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True		
Predicted	1	2	3	Sum	Predicted	1	2	3	Sum	Predicted	1	2	3	Sum	Predicted	1	2	3	Sum
1	13	0	0	13	1	13	0	0	13	1	15	0	0	15	1	15	0	0	15
2	13	19	8	40	2	12	20	12	44	2	0	10	0	10	2	0	8	0	8
3	3	4	20	27	3	4	3	16	23	3	14	13	28	55	3	14	15	28	57
Sum	29	23	28	80	Sum	29	23	28	80	Sum	29	23	28	80	Sum	29	23	28	80

Figura 5.35: La prima tabella indica le predizioni di 80 partite fatte dal modello (4.1), la seconda dal modello (5.1), la terza dal modello (4.12), la quarta dal modello (5.2) e la quinta dai *bookmakers*. La classe 1 indica la vittoria della squadra in casa, la classe 2 indica il pareggio tra le due squadre, la classe 3 indica la vittoria della squadra ospite. Con **True** si indicano le classificazioni effettive mentre con **Predicted** le classificazioni effettuate.

dei quattro modelli è rispettivamente 0.65, 0.6125, 0.6625 e 0.6375, mentre per i *bookmakers* è di 0.55. Si può subito notare che tutti e quattro i modelli sono migliori delle predizioni dei *bookmakers*. In particolare, il modello (4.12) risulta essere quello che produce più predizioni corrette. Sorprendentemente il modello (4.1) che utilizza solo le abilità medie delle squadre e quindi senza l'utilizzo delle variabili esplicative risulta essere migliore di tutti eccetto del modello (4.12). In particolare, si conferma quanto enunciato nel Capitolo 4 riguardo al ruolo dell'intercetta e delle covariate, infatti il modello senza intercetta (5.2) risulta essere leggermente peggiore del modello con l'intercetta (4.12). A tal proposito si può confermare l'esistenza di una differente relazione delle variabili esplicative da squadra a squadra. Infatti, tra i quattro modelli confrontati, il modello (5.1) ottiene le peggiori prestazioni dato che ha ignorato le diverse relazioni delle variabili esplicative con ogni singola squadra.

Nella Figura 5.36 vengono mostrate le misurazioni della sensibilità per ognuna delle tre categorie per i quattro modelli e per i *bookmakers*.

Come si può notare i *bookmakers* hanno molte difficoltà a identificare le partite che

Modello (4.1)			Modello (5.1)			Modello (4.12)		
1	2	3	1	2	3	1	2	3
0.4482759	0.8260870	0.7142857	0.4482759	0.8695652	0.5714286	0.5172414	0.4347826	1.0000000
Modello (5.2)			Bookmakers					
1	2	3	1	2	3	0.5172414	0.3478261	1.0000000
0.8181818	0.0000000	0.6538462	0.8181818	0.0000000	0.6538462			

Figura 5.36: La prima tabella indica le sensibilità delle predizioni del modello (4.1), la seconda del modello (5.1), la terza del modello (4.12), la quarta del modello (5.2) e la quinta dei *bookmakers*. La classe 1 indica la vittoria della squadra in casa, la classe 2 indica il pareggio tra le due squadre, la classe 3 indica la vittoria della squadra ospite.

terminano con un pareggio. Infatti, vediamo che la sensibilità è pari a zero. Questo perché ci sono zero partite identificate con il pareggio, ma dai dati osservati si sa che ci sono delle partite che terminano con tale esito. Tuttavia, sono molto affidabili nell'identificare la vittoria della squadra in casa contro la squadra ospite. I modelli (4.12) e (5.2) sanno identificare correttamente la vittoria della squadra ospite sulla squadra che gioca in casa. Infatti, la sensibilità associata è pari a uno. Infine, notiamo che i modelli (4.1) e (5.1) fanno registrare delle buone prestazioni sulla sensibilità delle partite di classe pareggio infatti, sbagliano a identificare rispettivamente quattro e tre partite su 20. Viceversa, i modelli (4.12) e (5.2) non hanno buone prestazioni sulla sensibilità della classe pareggio infatti, molte partite di classe pareggio non vengono identificate correttamente con tali dai due modelli. Per tutti i modelli Bradley-Terry calcolati la sensibilità della classe vittoria della squadra in casa è discreta.

Nella Figura 5.37 vengono mostrate le misurazioni della precisione per ognuna delle tre categorie per i quattro modelli e per i *bookmakers*. Come già riportato, i *bookmakers* hanno molte difficoltà a etichettare correttamente le partite che terminano con un pareggio. Infatti, sono state classificate zero partite con la classe pareggio. Inoltre, la precisione registrata nelle classi vittoria della squadra in casa e vittoria della squadra ospite è discreta. I modelli (4.12) e (5.2) sono estremamente precisi nell'etichettare le partite con le classi vittoria della squadra in casa e pareggio infatti, non hanno commesso alcun errore di classificazione. Purtroppo, le prestazioni dei modelli (4.12) e (5.2) calano per quanto riguarda la precisione sulla classe vittoria della squadra ospite.

Modello (4.1)			Modello (5.1)			Modello (4.12)		
1	2	3	1	2	3	1	2	3
1.0000000	0.4750000	0.7407407	1.0000000	0.4545455	0.6956522	1.0000000	1.0000000	0.5090909
Modello (5.2)			Bookmakers					
1	2	3	1	2	3	0.5625000	0.0000000	0.5312500
1.0000000	1.0000000	0.4912281	0.5625000	0.0000000	0.5312500			

Figura 5.37: La prima tabella indica la precisione delle predizioni del modello (4.1), la seconda del modello (5.1), la terza del modello (4.12), la quarta del modello (5.2) e la quinta dei *bookmakers*. La classe 1 indica la vittoria della squadra in casa, la classe 2 indica il pareggio tra le due squadre, la classe 3 indica la vittoria della squadra ospite.

Modello (4.1)			Modello (5.1)			Modello (4.12)		
1	2	3	1	2	3	1	2	3
1.0000000	0.6315789	0.8653846	1.0000000	0.5789473	0.8653846	1.0000000	1.0000000	0.4807692
Modello (5.2)			Bookmakers					
1	2	3	1	2	3	0.5531915	1.0000000	0.7222222
1.0000000	1.0000000	0.4423077						

Figura 5.38: La prima tabella indica le specificità delle predizioni del modello (4.1), la seconda del modello (5.1), la terza del modello (4.12), la quarta del modello (5.2) e la quinta dei *bookmakers*. La classe 1 indica la vittoria della squadra in casa, la classe 2 indica il pareggio tra le due squadre, la classe 3 indica la vittoria della squadra ospite.

Anche i modelli (4.1) e (5.1) sono estremamente precisi nell'etichettare le partite con la classe vittoria della squadra in casa infatti, non hanno commesso alcun errore di classificazione. Buone prestazioni si registrano per quanto riguarda le partite che terminano con la vittoria della squadra ospite. Purtroppo, i modelli hanno registrato molti errori di classificazione nella classe pareggio.

Nella Figura 5.38 vengono mostrate le misurazioni della specificità per ognuna delle tre categorie per i quattro modelli e per i *bookmakers*.

Ovviamente in questa misurazione i *bookmakers* ottengono il miglior risultato per quanto riguarda l'identificazione delle partite che non terminano in un pareggio. Si nota che tutti e quattro i modelli Bradley-Terry riescono a predire quando l'esito della partita non è la vittoria della squadra in casa, infatti la misurazione ottenuta è uno. Inoltre, i modelli (4.12) e (5.2) riescono a identificare correttamente le partite che non terminano in un pareggio. D'altra parte, i modelli (4.1) e (5.1) ottengono le migliori misurazione nell'identificare le partite che non terminano con la vittoria della squadra ospite. Viceversa, i modelli (4.12) e (5.2) che ottengono le peggiori prestazioni nell'identificare le partite che non terminano con la vittoria della squadra ospite. In conclusione, dai risultati ottenuti si evince che i modelli mostrano una maggiore accuratezza nelle previsioni rispetto a quanto fornito dai *bookmakers*. Se ne deduce quindi un buon utilizzo delle informazioni a disposizione nei modelli.

6 | ALGORITMI DI MACHINE LEARNING

Questo capitolo illustrerà i metodi di machine learning che sono stati utilizzati per la predizione degli esiti delle partite di calcio della Serie A italiana della stagione 2021/2022. Purtroppo, non è stato possibile applicare metodi di machine learning che corrispondessero al modello Bradley-Terry perché, nonostante esistano metodi in machine learning che forniscono modelli basati sul modello Bradley-Terry, essi non sono in grado di gestire l'esito del pareggio ma solo un esito binario. Ne consegue che tali metodi non sono adatti per contesti come il calcio ma ad altri tipi di sport dove il pareggio non è previsto come il baseball. I metodi di machine learning considerati sono: il K-Nearest-Neighbors (K-NN), la Support Vector Machine (SVM), gli alberi di decisione per la classificazione, la Random Forest e infine l'AdaBoost.

6.1 Componenti essenziali

In questa sezione vengono definite alcune misure e tecniche che sono necessarie per l'applicazione dei metodi di *machine learning* applicati.

6.1.1 Distanza di Minkowski

La distanza di Minkowski (vedi **minkdist**) è una misura utilizzata per la valutazione della distanza ovvero, nel nostro contesto, della somiglianza tra due punti in uno spazio di n -dimensioni. La distanza di Minkowski di ordine d tra due punti $A = (a_1, \dots, a_n)$ e $B = (b_1, \dots, b_n)$ è definita come

$$Dist(A, B) = \left(\sum_{i=1}^n |a_i - b_i|^d \right)^{1/d}.$$

Si sottolinea che per $d = 1$, la distanza utilizzata è la distanza di Manhattan (vedi **manhattan**), ovvero la distanza tra due punti è la somma del valore assoluto delle differenze delle loro coordinate. Quando $d = 2$, è applicata la distanza Euclidea (vedi **euclidea**), secondo la quale la distanza tra due punti è la lunghezza del segmento con agli estremi i due punti d'interesse. Tale misura sarà utilizzata nel metodo K-Nearest-Neighbors (K-NN).

6.1.2 Funzione kernel

Nel contesto dell'apprendimento automatico, la funzione kernel (vedi **kernel**) permette di trasformare uno spazio di input non linearmente separabile in un nuovo spazio delle istanze di input detto *feature space* di dimensione superiore rispetto a quello originale e tale da diventare linearmente separabile. Per spazio linearmente separabile si intende che esiste un iperpiano in grado di separare correttamente i dati in due gruppi distinti. Perciò, aumentando la dimensionalità dello spazio d'interesse è possibile trovare la dimensione opportuna che permetta di separare linearmente i dati. Tale applicazione è chiamata *kernel trick*. Una funzione kernel è una funzione K che per ogni $x, y \in \chi$, dove χ è lo spazio di input di dimensione n , vale

$$K(x, y) = \langle \psi(x), \psi(y) \rangle,$$

con ψ funzione che mappa i punti di uno spazio di dimensione n in uno spazio di dimensione m con $m > n$, e $\langle \cdot \rangle$ prodotto scalare.

Nelle nostre predizioni saranno usati i seguenti kernel:

- * linear kernel, è la funzione precedentemente definita;
- * polynomial kernel, $K(x, y) = (1 + \sum_{i=1}^p x_i y_i)^d$, dove p è il numero di istanze di input presenti in χ e d la dimensione del spazio (l'ordine);
- * Gaussian Radial Basis kernel (RBF), $K(x, y) = \exp(-\gamma \|x - y\|^2)$, con $\gamma = \frac{1}{2\sigma^2}$ e σ parametro libero.

Nella Figura 6.1 è mostrato un esempio di applicazione della funzione kernel.

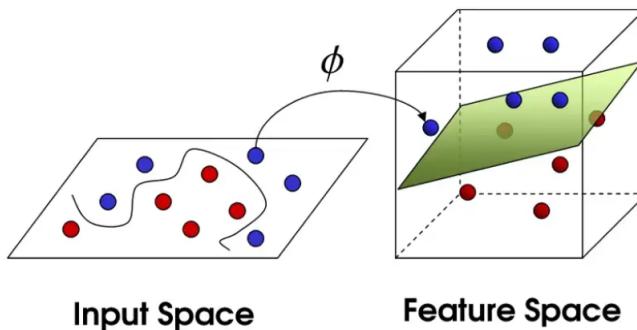


Figura 6.1: Esempio grafico della funzione kernel γ che mappa i punti di uno spazio d'input in uno *feature space* di dimensione maggiore e linearmente separabile rispetto a quello originale.

Source: <https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f>

La funzione kernel sarà utilizzata nella Support Vector Machine (SVM).

6.1.3 Bootstrap

In statistica e nell'apprendimento automatico, per Bootstrap (vedi **bootstrap**) si intende una tecnica di ricampionamento per la generazione di un insieme di campioni di m osservazioni contenute in un dataset di dimensione n . Ogni estrazione è casuale e con reinserimento, cioè un'osservazione può essere presente in più campioni. Tale tecnica è utilizzata per produrre un insieme di campioni che siano il più possibile rappresentativi e indipendenti tra di loro.

Nella Figura 6.2 viene mostrato un esempio della procedura di Bootstrap.

6.1.4 Bagging

Il Bagging (**breiman1996bagging**) detto anche *Bootstrap Aggregation Approach*, è una tecnica di *ensemble learning* di tipo parallelo che dalla mediazione di più predizioni fatte da un insieme di classificatori deboli ottiene un'unica predizione finale. È di tipo parallelo perché va a sfruttare l'indipendenza dei classificatori. La procedura applicata è la seguente:



Figura 6.2: Esempio grafico della procedura di Bootstrap. Da un *dataset* di dati iniziale vengono estratti, con reinserimento, gli elementi del *dataset* per formare k campioni avendo tutti lo stesso numero di elementi.

Source: <https://blog.paperspace.com/bagging-ensemble-methods/>

- * creazione di k campioni utilizzando la tecnica di Bootstrap;
- * per ogni campione, allenare un classificatore;
- * produrre una predizione per ogni classificatore allenato;
- * ottenere una media delle predizioni per avere una predizione finale.

Una tecnica per ottenere una media è, ad esempio, il *voting* secondo il quale la classe più predetta sarà il risultato dalla predizione finale. Inoltre, si utilizza il Bootstrap per rendere i classificatori indipendenti tra di loro. Perciò l'obiettivo del Bagging è quello di creare un classificatore in grado di gestire un'elevata varianza dei dati in modo efficiente grazie al parallelismo.

Nella Figura 6.3 viene illustrato graficamente la procedura di Bagging.

6.1.5 Boosting

Il Boosting (**freund1996experiments**) è una tecnica di *ensemble learning* di tipo sequenziale che sfrutta la dipendenza tra i classificatori usati. L'algoritmo inizialmente allena un classificatore debole con tutto il *dataset* a disposizione. Successivamente, per raffinare la predizione, vengono allenati in sequenza nuovi classificatori che apprendono da tutto ciò che è stato appreso dal classificatore precedente utilizzando ancora l'intero *dataset*.

La procedura completa è la seguente:

- * viene utilizzato l'intero *dataset* per allenare un classificatore debole;
- * vengono ripetuti gli esempi di *training* dando un peso maggiore a quei esempi classificati erroneamente, viceversa per gli esempi classificati correttamente;
- * ripetere per n volte i passi precedenti con un nuovo classificatore con i pesi aggiornati;
- * combinare tutte le ipotesi semplici in un unico classificatore accurato per ottenerne il risultato finale.

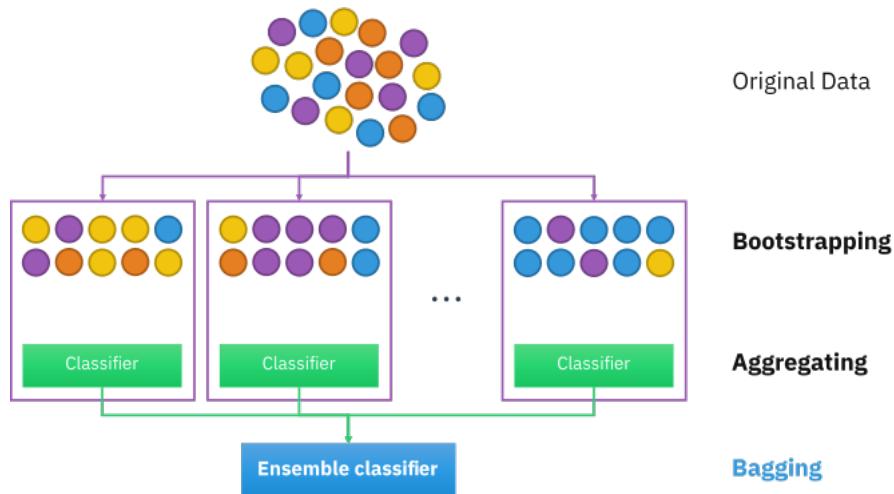


Figura 6.3: Esempio grafico della procedura di Bagging. L’algoritmo inizia generando k campioni avvenuti tutti lo stesso numero di elementi, attraverso il Bootstrapping, a partire dai dati originali ricevuti in input. Successivamente ogni campione viene assegnato a un solo classificatore per effettuare il training del classificatore. Infine le predizioni dei classificatori istruiti vengono tutte aggregate per formare un’unica predizione finale.

Source: <https://www.analyticsvidhya.com/blog/2020/02/what-is-bootstrap-sampling-in-statistics-and-machine-learning/>

Con l’aggiornamento dei pesi si presuppone che i classificatori successivi non andranno a commettere gli stessi errori dei classificatori precedenti.

L’obiettivo del Boosting è concentrare i propri sforzi nel creare un classificatore adatto a gestire un’elevata distorsione, anziché un’elevata varianza dei dati. Infatti, partendo da un classificatore debole e migliorandolo in modo sequenziale, si consente ai classificatori successivi di imparare dagli errori precedentemente commessi, riducendo la distorsione dei dati. Inoltre, il Boosting ha una buona resistenza agli effetti dell’*overfitting*. Purtroppo, il Boosting risulta molto sensibile ai valori anomali. Inoltre, dato che le operazioni di addestramento di ogni classificatore avvengono in modo sequenziale, non sarà possibile utilizzare il parallelismo per risparmiare tempo di calcolo.

Nella Figura 6.4 viene illustrata la procedura di Boosting.

6.2 K-Nearest-Neighbors

Il K-Nearest-Neighbors (K-NN) (**dasarathy1991nearest**) è un algoritmo di apprendimento automatico di tipo supervisionato che permette la classificazione delle istanze ricevute in input. Inoltre, ne esiste una versione per problemi di regressione. Il K-NN assume che tutte le istanze corrispondano a punti in uno spazio di dimensionalità n e utilizza la prossimità tra i vari punti per classificarli, ossia classifica l’istanza con la classe maggiormente presente tra i punti attorno all’istanza stessa, detti punti vicini. Fondamentale, perciò, è l’utilizzo di una qualche tecnica di misurazione della distanza per individuare chi sono i vicini dell’istanza da classificare, ossia la distanza tra il nostro punto d’interesse rispetto a tutti gli altri punti. La misura di distanza maggiormente

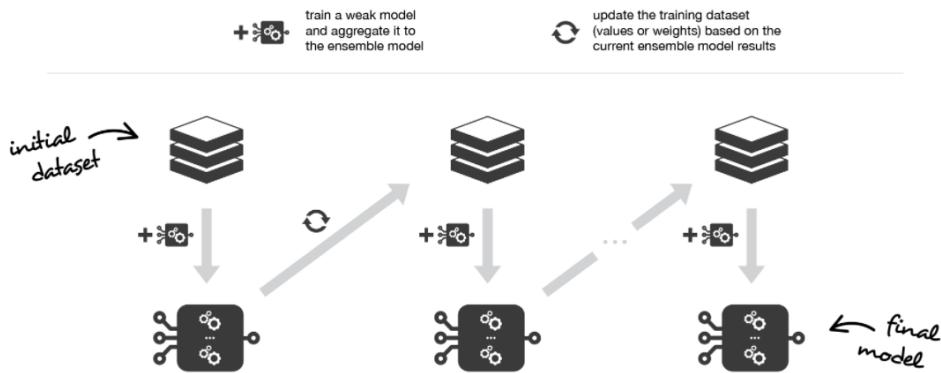


Figura 6.4: Esempio grafico della procedura di Boosting. L'algoritmo parte da un classificatore debole iniziale il quale viene addestrato su tutti i dati. Successivamente, dopo che il classificatore debole ha effettuato delle predizioni, vengono aggiornati i pesi di quei esempi che sono stati erroneamente classificati. Viene allenato un nuovo modello debole sulla base di quanto fatto dal classificatore precedente, ripetendo la procedura di allenamento e aggiornamento dei pesi.

Source: <https://www.section.io/engineering-education/boosting-algorithms-python/>

utilizzata è la distanza di Minkowski definita nella sezione precedente. Misurati tutti i punti, occorre stabilire poi quanti dei punti presenti devono essere considerati vicini. A questo scopo, va fissato il valore di k , iperparametro dell'algoritmo che stabilisce di considerare solo i k punti più vicini all'istanza da classificare. Per esempio, se $k = 3$, si considerano i tre punti più vicini e si classifica l'istanza con la classe più frequente tra i tre punti considerati. È importante scegliere il corretto valore di k , dato il rischio di *overfitting*, nel caso si considerino troppi punti vicini, o *underfitting*, nel caso si considerino pochi punti vicini. Infatti, con valori più bassi di k può verificarsi un'elevata varianza e una distorsione bassa, mentre con valori più grandi di k può verificarsi un'elevata distorsione e una varianza bassa delle predizioni. Una buona soluzione per la scelta dell'iperparametro k , così come dell'ordine p della distanza da utilizzare, è la Cross Validation.

Nella Figura 6.5 è mostrato un esempio di applicazione dell'algoritmo K-NN.

Il K-NN è un algoritmo di classificazione non parametrico, ovvero non fa alcuna assunzione sulla forma della distribuzione dei dati. Inoltre, dato che è un algoritmo di apprendimento supervisionato, le istanze d'input sono nella forma $(x, f(x))$. Nella fase di addestramento si limita soltanto a memorizzare i dati di *training*, dato che li utilizza direttamente per fare predizione. Purtroppo, però, la fase di predizione può essere lenta, poiché è necessario calcolare la distanza di ogni osservazione dall'istanza da classificare, con un costo computazionale crescente se vi sono molti dati.

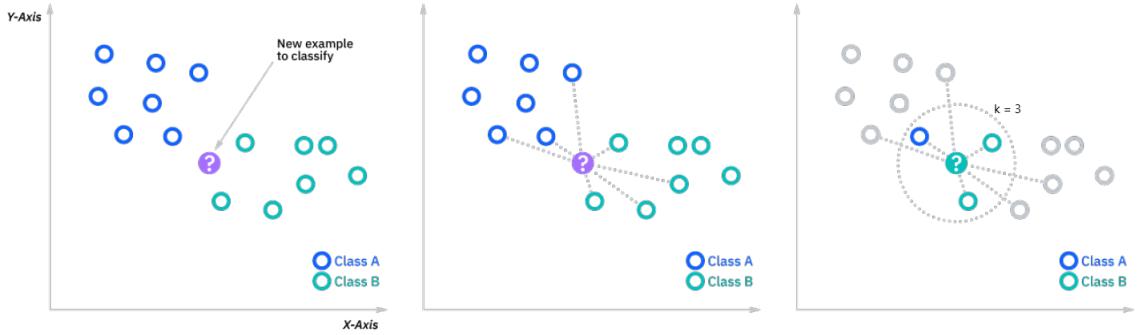


Figura 6.5: Esempio grafico dell’algoritmo K-Nearest-Neighbors. L’istanza da classificare è indicata con il punto interrogativo (?). Il primo passo dell’algoritmo è quello di calcolare tutte le distanze. Dopo di che, considera solo i $k = 3$ punti più vicini all’istanza (?). Infine l’algoritmo classifica con la classe B l’istanza (?).

Source: <https://www.ibm.com/topics/knn#:~:text=The%20k%20nearest%20neighbors%20algorithm%20also%20known%20as%20KNN%20or,of%20an%20individual%20data%20point.>

6.3 Support Vector Machine

La Support Vector Machine (SVM) (**GHOLAMI2017515**) è un algoritmo di apprendimento automatico di tipo supervisionato applicabile in contesti di classificazione. La SVM considera le istanze del *dataset* come punti in uno spazio di dimensionalità n e il suo obiettivo è quello di costruire l’iperpiano ottimo che separi in due classi le osservazioni. L’iperpiano ottimo viene scelto in modo tale da ottenere il maggior margine possibile tra le due classi, ovvero il maggior spazio possibile tra le osservazioni di ciascuna classe e l’iperpiano. Il nome di quest’algoritmo deriva dall’utilizzo dei vettori detti vettori di supporto. Questi vettori sono le istanze che si trovano più vicino all’iperpiano ovvero quelli più difficili da classificare e quindi che danno un grosso contributo alla costruzione dell’iperpiano rispetto alle altre osservazioni. Perciò per massimizzare la distanza tra l’iperpiano e i punti di entrambe le classi, occorre risolvere il seguente problema di ottimizzazione vincolata,

$$\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n (\xi_i) \quad (6.1)$$

$$\begin{cases} y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0, i = \{1, \dots, n\} \end{cases} .$$

Nel problema 6.1 $\|\mathbf{w}\|$ è il vettore direzione e l’iperparametro C è un parametro di regolarizzazione, il quale permette di gestire il *trade-off* tra massimizzazione del margine e penalizzazione, consentendo di controllare la complessità del modello e, quindi, di prevenire l’*overfitting*. Il parametro ξ_i è l’errore commesso. La quantità $\mathbf{w} \cdot \mathbf{x}_i + b$ è la distanza algebrica tra l’iperpiano scelto e il punto più vicino.

Nella Figura 6.6 è mostrato un esempio di applicazione dell’algoritmo SVM. L’algoritmo

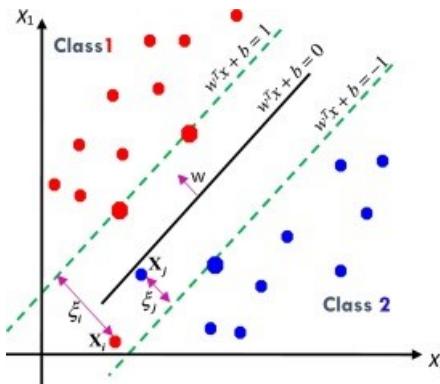


Figura 6.6: Esempio grafico dell'algoritmo Support Vector Machine. I punti sulle linee tratteggiate indicano i vettori di supporto, mentre la retta al centro indica l'iperpiano ottimo di separazione.

Source:

<https://www.sciencedirect.com/science/article/pii/B9780128113189000272>

SVM è in grado di gestire anche spazi d'input non linearmente separabili, grazie all'utilizzo della funzione kernel definita nella sezione precedente. Tramite la Cross Validation si sceglierà il valore più opportuno per l'iperparametro C e il tipo di kernel da applicare.

6.4 Decision Tree

Un Decision Tree (**charbuty2021classification**) è un algoritmo di apprendimento automatico di tipo supervisionato e non parametrico che utilizza una struttura ad albero per produrre le proprie predizioni. Tale albero contiene un insieme di nodi in cui, per ogni nodo, vi è un test su un attributo dell'osservazione da classificare. Perciò, ad ogni nodo, ci sarà una scelta da compiere in base al valore contenuto dell'attributo, che porterà verso un nuovo ramo oppure a una foglia. Le foglie contengono i risultati della classificazione. L'approccio utilizzato per la costruzione dell'albero di decisione è di tipo *greedy*, cioè ogni scelta effettuata su un nodo è l'opzione più conveniente in quel momento. L'albero viene costruito in modalità *top-down* secondo i seguenti passi:

- * viene creata la radice T dell'albero;
- * se le osservazioni dell'insieme D sono tutte della stessa classe k ; allora viene ritornata la radice T classificata con la classe k ;
- * se le osservazioni non hanno attributi che li descrivono, allora viene ritornata la radice T classificata con la classe di maggior presenza tra le osservazioni;
- * viene scelto un attributo a , in base a una specifica regola;
- * viene partizionato D a seconda dei m valori che può assumere l'attributo a ;
- * vengono creati ricorsivamente i sottoalberi dall'albero con radice T senza l'attributo a ripetendo i passi appena descritti.

Un iperparametro di quest'algoritmo è la regola per la decisione di quale attributo testare in un nodo. Nel presente lavoro saranno considerate le seguenti regole:

- * Cross Entropy: È la misura del grado di impurità di un insieme di osservazioni. Sia m il numero di classi e D_k il sottoinsieme di D di osservazioni di classe k , allora l'entropia vale

$$I_E = - \sum_{k=1}^m p_k \log(p_k),$$

dove p_k è la probabilità di estrarre un'osservazione di classe k , ovvero vale $\frac{|D_k|}{|D|}$.

- * Gini Index: È la misura di quanto spesso un'osservazione estratta casualmente dall'insieme delle osservazioni è classificata in modo errato basandosi solo sulla distribuzione delle classi. Sia m il numero di classi e D_k il sottoinsieme di D di osservazioni di classe k , allora l'indice di Gini vale

$$I_G = 1 - \sum_{k=1}^m p_k^2,$$

dove p_k è la probabilità di estrarre un'osservazione di classe k , ovvero vale $\frac{|D_k|}{|D|}$.

Per scegliere l'attributo da valutare ad ogni nodo, si fa riferimento all'*information gain*

$$G(D, a) = I_x - \sum_{v \in V(a)} \frac{|D_a = v|}{D} I_x(D_a = v),$$

dove I_x è la regola scelta e $D_a = v$ è l'insieme delle istanze con valore v nell'attributo a . L'attributo che viene scelto per il test nel nodo è quello con il maggior valore nell'*information gain*.

Il Decision Tree è un modello facile da interpretare e utilizzare, tuttavia, può essere suscettibile all'*overfitting*. È opportuno, perciò, attraverso la Cross Validation, scegliere il valore da assegnare all'iperparametro del limite della massima profondità dell'albero per limitarne la complessità, oltre a scegliere quale regola di decisione utilizzare.

6.5 Random Forest

Il Random Forest (**ho1995random**) è un algoritmo di apprendimento automatico basato sull'utilizzo della tecnica di Bagging applicato ad un insieme di Decision Tree utilizzati per la classificazione. L'obiettivo dell'algoritmo è ottenere una bassa correlazione tra gli alberi e quindi ridurre la varianza. Per tale motivo viene utilizzato il Bagging per addestrare gli alberi di decisione. Tramite il Bootstrap viene quindi creata una serie di campioni di esempi di training, in cui ogni campione ha un sottoinsieme di attributi e viene utilizzato per addestrare un'albero di decisione. Una volta che sono stati costruiti gli alberi, viene fatta la predizione per ogni modello e, attraverso un algoritmo di mediazione, ad esempio il *voting*, viene prodotta la predizione finale. La Random Forest è in grado di ridurre l'*overfitting* che può verificarsi con l'utilizzo di un singolo albero di decisione. Tuttavia, richiede una quantità significativa di risorse computazionali per la costruzione dei singoli alberi di decisione e può essere meno interpretabile rispetto a un singolo albero di decisione.

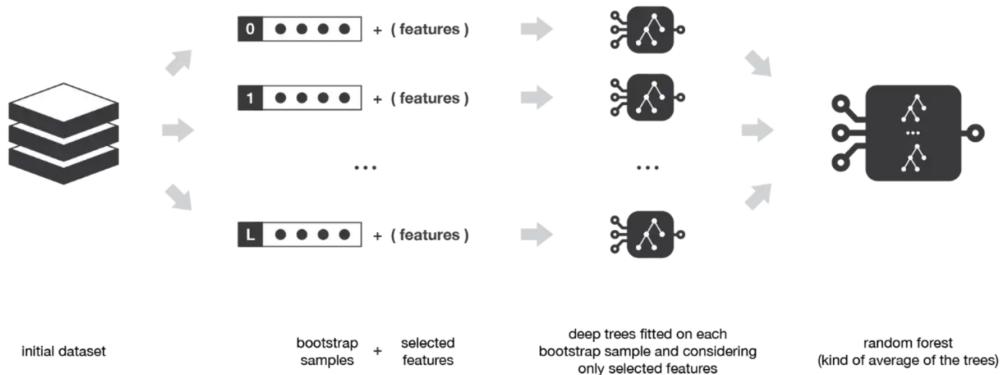


Figura 6.7: Esempio grafico dell'algoritmo Random Forest. L'algoritmo partendo dai dati iniziali, tramite il Bootstrap, crea una serie di campioni di esempi di training, in cui ogni campione ha un sottoinsieme di attributi. Ogni campione viene utilizzato per addestrare un'albero di decisione. Una volta che sono stati costruiti gli alberi, viene fatta la predizione per ogni modello e, attraverso un algoritmo di mediazione, ad esempio il *voting*, viene prodotta la predizione finale.

Source: <https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205>

Nella Figura 6.7 è mostrato un esempio di applicazione dell'algoritmo Random Forest. Gli iperparametri utilizzati sono gli stessi del Decision Tree con l'aggiunta dell'iperparametro che indica il numero di alberi di decisione scelto tramite Cross Validation.

6.6 AdaBoost

L'algoritmo di AdaBoost (Adaptive Boosting) (**auer1995gambling**) è un algoritmo di apprendimento supervisionato che utilizza la tecnica di Boosting per creare un modello di classificazione accurato a partire da un insieme di classificatori deboli. Ad ogni iterazione, i pesi assegnati alle osservazioni vengono modificati dando maggior peso alle istanze che sono state classificate erroneamente per procedere con l'addestramento di un nuovo classificatore debole. L'obbiettivo dei classificatori deboli è quello di massimizzare l'accuratezza del peso, ovvero, minimizzare l'errore ξ_t

$$\xi_t = \sum_i D_t(i)[h_t(x_i) \neq y_i],$$

dove D_t è la distribuzione dei pesi della t -esima iterazione. La funzione $h_t(x_i)$ indica il risultato prodotto dal classificatore debole nell' i -esima istanza mentre y_i è il target dell' i -esima istanza. Perciò la variazione del peso α_t di ogni classificatore viene calcolato tenendo conto dell'errore ξ_t ,

$$\alpha_t = \frac{1}{2} \log \frac{1 - \xi_t}{\xi_t}.$$

Una volta calcolato α_t , i pesi vengono modificati dando più peso alle osservazioni classificate in modo errato. Viceversa per le osservazioni classificate correttamente.

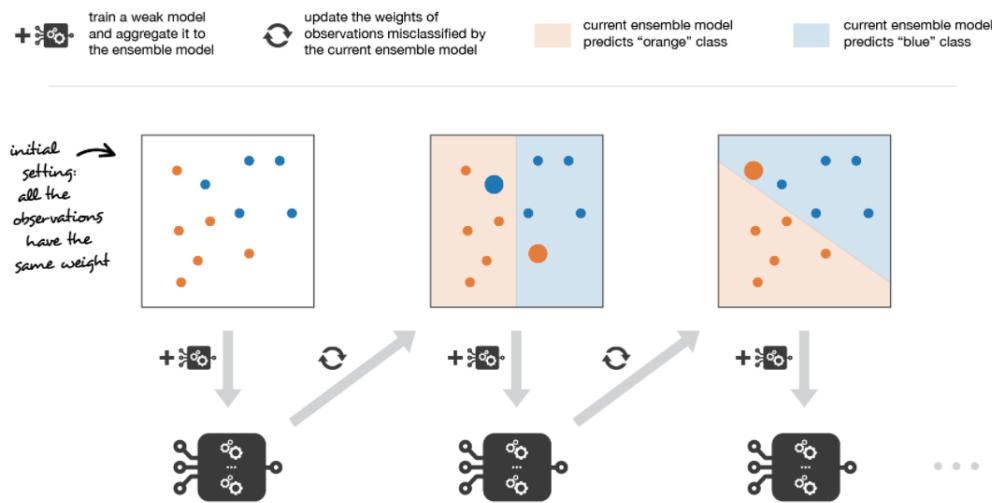


Figura 6.8: Esempio grafico dell'algoritmo AdaBoost. Quest'algoritmo utilizza la tecnica di Boosting per creare un modello di classificazione accurato a partire da un insieme di classificatori deboli. Ad ogni iterazione, i pesi assegnati alle osservazioni vengono modificati dando maggior peso alle istanze che sono state classificate erroneamente per procedere con l'addestramento di un nuovo classificatore debole.

Source: <https://www.section.io/engineering-education/boosting-algorithms-python/>

Nella Figura 6.8 è mostrato un esempio di applicazione dell'algoritmo AdaBoost. Gli iperparametri da gestire in questo algoritmo sono: il numero di classificatori deboli e il *learning rate*. Il *learning rate* indica quanto velocemente procederà l'addestramento. Per regolare la velocità, attraverso il *learning rate* viene indicata quanta porzione dell'aggiornamento dei pesi verrà applicata sui pesi per la prossima iterazione. Ovviamente, questi iperparametri sono scelti attraverso la Cross Validation. L'algoritmo AdaBoost risulta essere robusto contro l'*overfitting*.

7 | RISULTATI DEI ALGORITMI DI MACHINE LEARNING

Questo capitolo illustrerà i risultati ottenuti dai algoritmi K-Nearest-Neighbors (K-NN), Support Vector Machine (SVM), Decision Tree, Random Forest e infine AdaBoost.

7.1 Premesse

Si sottolinea che, purtroppo, i metodi di *machine learning* non consentono un'analisi interpretativa dei dati come i metodi di *data mining*. Per questo verranno presentati solo i risultati delle predizioni con le relative metriche.

Durante la fase di *preprocessing*, i dati sono stati standardizzati attraverso la funzione `StandardScaler()` del linguaggio Python, ovvero tutti i dati per ogni *feature* sono stati centrati per ottenere media 0 e varianza 1. Questa operazione è stata eseguita per poter rendere le *features* comparabili tra loro. Come già verificato nel Capitolo 3 non ci sono valori mancanti. Durante la fase di *feature selection* si utilizzano le stesse *features* del modello Bradley-Terry escludendo ancora il numero di gol fatti dalla squadra in casa e dagli ospiti. Questo perché, durante una prima fase iniziale di verifica degli algoritmi, i metodi Decision Tree e Random Forest, grazie a queste *feature*, ottenevano un'accuratezza pari a 1. Infatti, sapere il numero di gol segnati in una partita indica implicitamente l'esito della partita stessa. Quindi per non rendere inutili le altre *feature*, il numero di gol segnati dalla squadra in casa e dagli ospiti non sono stati presi in considerazione. Ciononostante, si sottolinea la bontà dei due algoritmi che sono riusciti a trovare la correlazione precedentemente illustrata. Quindi le 26 *features* scelte diventeranno 52, una metà delle quali si riferiscono alla squadra in casa, in cui ogni *feature* viene indicata con il prefisso `Home_`, mentre l'altra metà si riferisce alla squadra ospite, in cui ogni *feature* viene indicata con il prefisso `Away_`. L'attività di predizione verrà condotta suddividendo il *dataset* in un insieme di training (80%) e uno di test (20%). La funzione utilizzata per rendere utilizzabile il *dataset* è indicata nella Sezione 12.1.

7.2 Ulteriori metriche

La capacità predittiva di un modello sarà valutata con le metriche illustrate nel Capitolo 5 e con la seguente metrica.

- * **Area Under the Curve.** La Area Under the Curve (AUC) rappresenta l'area al di sotto della curva ROC, che è una rappresentazione grafica del rapporto tra specificità e sensibilità. L'AUC è una metrica che va da 0 a 1, dove un valore più alto indica una migliore prestazione di classificazione. L'AUC permette di confrontare le performance di classificatori diversi, poiché è indipendente dalla soglia di classificazione e fornisce un riassunto delle prestazioni del classificatore.

7.3 K-Nearest-Neighbors

L'algoritmo K-Nearest-Neighbors è risultato essere semplice da applicare, ottenendo complessivamente discreti risultati in fase di predizione. Per scegliere i valori migliori per gli iperparametri si è applicato la K-Fold Cross Validation con $k = 10$. Gli iperparametri valutati sono stati i seguenti:

- * $n_{neighbors}$. Indica il numero massimo di vicini da considerare. Si sono verificati i valori da 3 fino a 163 vicini con un aumento unitario di 1.
- * p . Indica il parametro potenza della distanza di Minkowski. Si sono verificati i valori $p=1$ (Manhattan distance) e $p=2$ (Euclidean distance).

Nella Figura 7.1 viene mostrato l'andamento della Cross Validation per ogni valore degli iperparametri.

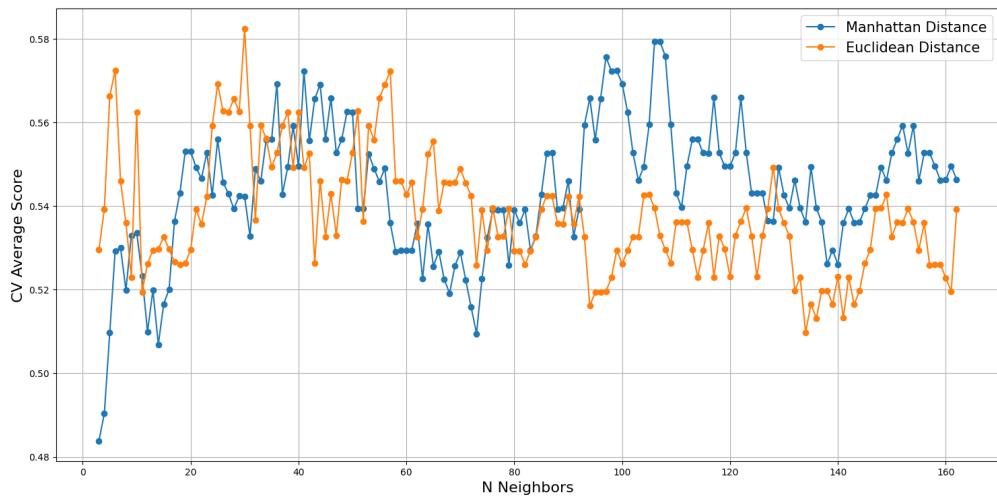


Figura 7.1: Grafico dell'andamento della media dell'accuratezza per ogni valore dell'iperparametro $n_{neighbors}$ e per ogni tipo di metrica di distanza utilizzata durante l'applicazione della Cross Validation con 10 fold per il modello K-Nearest-Neighbors. Ogni punto è un classificatore con un certo numero di vicini. La linea blu indica l'andamento con la distanza di Manhattan, mentre la linea arancione l'andamento con la distanza euclidea.

Quello che si può notare dalla figura è che inizialmente, con pochi vicini, la distanza euclidea risulta essere migliore nella fase di training nel *validation set*, ma, con l'aumentare del numero dei vicini, la distanza di Manhattan ottiene risultati migliori.

Secondo la Cross Validation i valori migliori sono stati: il valore 30 come numero di vicini e la distanza euclidea come metrica della distanza da utilizzare. L'accuratezza ottenuta dal modello migliore nel *validation set* è stata di 0.582.

Nella fase di predizione si sono ottenute le predizioni mostrate nella Figura 7.2 con le relative metriche presentate nella Figura 7.3.

I risultati ottenuti sono discreti. Infatti, l'accuratezza del modello nella fase di predizione è di 0.58. Il modello ha molta difficoltà a riconoscere quando un'osservazione è di classe pareggio dato che la sensibilità è pari 0.16 ovvero, delle 19 osservazioni

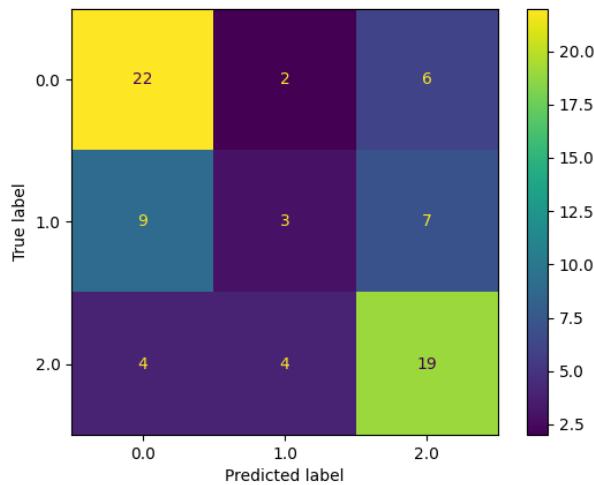


Figura 7.2: Tabella di confusione del modello K-Nearest-Neighbors con $n_{\text{neighbors}} = 30$ e $p = 2$. La classe 0.0 indica la vittoria della squadra in casa, la classe 1.0 indica il pareggio tra le due squadre, la classe 2.0 indica la vittoria della squadra ospite.

effettivamente di classe pareggio solo tre vengono riconosciute come tali. Purtroppo, quando le osservazioni vengono etichettate con il pareggio, molto spesso viene commesso un errore di classificazione, dato che la precisione è pari a 0.33 ovvero, solo tre osservazioni su nove sono effettivamente di classe pareggio. Il modello, commettendo relativamente un numero di errori basso, ha una specificità pari a 0.89. Risultati discreti si ottengono per la classe vittoria della squadra in casa e vittoria della squadra ospite. La precisione del modello nella classe vittoria della squadra in casa è pari a 0.63. Infatti, 13 osservazioni vengono etichettate erroneamente con la classe vittoria della squadra in casa. La specificità risulta pari a 0.71 mentre la sensibilità della classe vittoria della squadra in casa è pari a 0.73 ovvero, otto osservazioni su trenta non sono state identificate di classe vittoria della squadra in casa. La precisione della classe vittoria della squadra ospite è pari 0.59 poiché sono stati commessi tanti errori di classificazione. Infatti, 13 osservazioni su 32 sono state etichettate erroneamente con la classe vittoria della squadra ospite. Invece la sensibilità registrata per la classe vittoria della squadra ospite è pari a 0.70 dato che solo otto osservazioni non sono state riconosciute appartenenti alla classe vittoria della squadra ospite. Analogamente, anche la specificità della classe vittoria della squadra ospite è buona dato che è pari a 0.73. Perciò, il modello riesce ad individuare quasi tutte le osservazioni di classe vittoria della squadra in casa o di classe vittoria della squadra ospite, anche se per entrambe le due classi ha una discreta precisione a causa di molte istanze di classe pareggio etichettate erroneamente.

Risulta non essere particolarmente adatto l'algoritmo K-Nearest-Neighbors per quest'analisi, a causa della grande diversità tra partita e partita. Ciononostante, si ottengono discreti risultati tenendo conto del fatto delle poche osservazioni messe a disposizione.

	precision	recall	specificity
Home team win	0.63	0.73	0.71
Tie	0.33	0.16	0.89
Away team win	0.59	0.70	0.73
accuracy			0.58
AUC Score			0.65

Figura 7.3: Grafico delle misurazione registrate durante la fase di predizione del modello K-Nearest-Neighbors con $n_neighbors = 30$ e $p = 2$.

7.4 Support Vector Machine

L'algoritmo Support Vector Machine ha ottenuto dei buoni risultati durante la fase di predizione. Analogamente all'algoritmo K-Nearest-Neighbors, si è applicata la K-Fold Cross Validation con $k = 10$ per individuare i valori migliori per gli iperparametri.

Gli iperparametri valutati sono stati i seguenti:

- * C . Indica la forza applicata della penalità L2. Si sono verificati valori che vanno da 0.1 a 1.5 con un aumento unitario di 0.1.
- * $kernel$. Indica quale funzione kernel utilizzare. Si sono verificati i valori $kernel = \text{linear}$ ovvero il linear kernel, $kernel = \text{rbf}$ ovvero Gaussian Radial Basis kernel (RBF) e $kernel = \text{poly}$ ovvero il polynomial kernel.

Nella Figura 7.4 viene mostrato l'andamento della Cross Validation per ogni valore degli iperparametri. Dal grafico si nota che, dal punto di vista dell'accuratezza registrata nel *validation set*, il kernel di tipo lineare ottiene le prestazioni migliori. Infatti, vediamo che la linea blu è costantemente al di sopra rispetto alle due restanti a parità del valore in C . Invece, il polynomial kernel non ha buone prestazioni. Infatti, si registra che la linea arancione è costantemente al di sotto delle altre due linee, perché ha valori molto più bassi di accuratezza. Secondo la Cross Validation i risultati migliori sono stati il valore 1.4 per l'iperparametro C e il Linear kernel come funzione kernel da utilizzare. L'accuratezza ottenuta dal modello migliore nel *validation set* è stata di 0.789. Nella fase di predizione si sono ottenute le predizioni mostrate nella Figura 7.5 con le relative metriche presentate nella Figura 7.6. Nonostante le poche osservazioni disponibili e l'elevato numero di *features* utilizzate, i risultati ottenuti sono buoni. Infatti, l'accuratezza delle predizioni è pari a 0.78. Il modello riesce a riconoscere buona parte delle osservazioni di classe pareggio. Infatti, la sensibilità per la classe pareggio è pari a 0.68 ovvero, solo sei osservazioni di classe pareggio su 19 non sono state riconosciute come tali. Tuttavia, la precisione scende a 0.59. Infatti, nove osservazioni sono state etichettate erroneamente dal modello con la classe pareggio. Ciononostante, quest'errore non è molto grande dato che la specificità è pari a 0.84. Risultati migliori si ottengono per la classe vittoria della squadra in casa e vittoria della squadra ospite. La precisione per la classe della vittoria della squadra in casa è pari a 0.92. Infatti, solo due osservazioni sono state etichettate erroneamente dal modello con la classe vittoria della squadra in casa. Di conseguenza anche la specificità ha un valore molto alto ovvero pari a 0.95. Tuttavia, sette osservazioni di classe vittoria della squadra in casa non sono state identificate correttamente, infatti la sensibilità è pari a 0.77. Invece, la

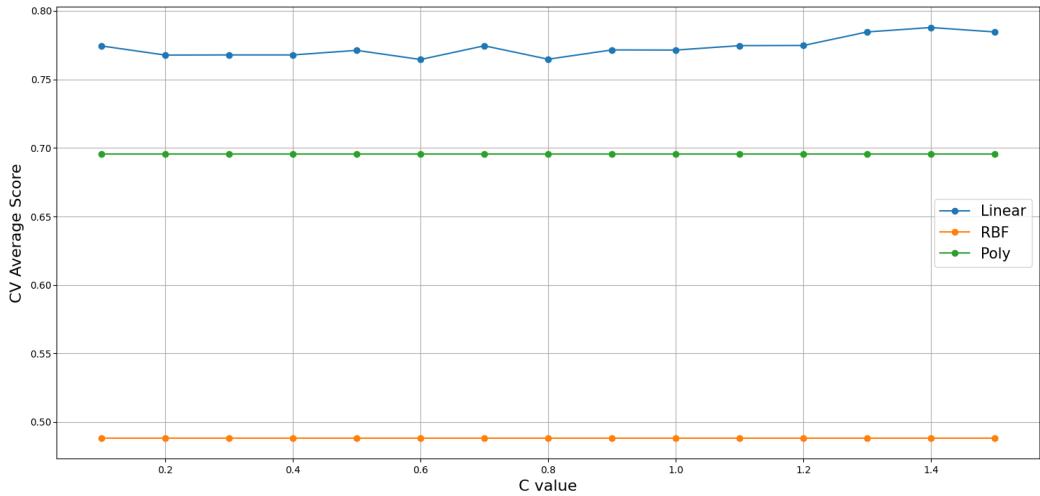


Figura 7.4: Grafico dell’andamento della media dell’accuratezza per ogni valore dell’iperparametro C e per ogni funzione kernel utilizzata durante l’applicazione della Cross Validation con 10 fold per il modello Support Vector Machine. Ogni punto è un classificatore con un certo valore di C. La linea blu indica l’andamento con la funzione linear kernel mentre la linea verde l’andamento con la funzione RBF e la linea arancione l’andamento con la funzione polynomial kernel.

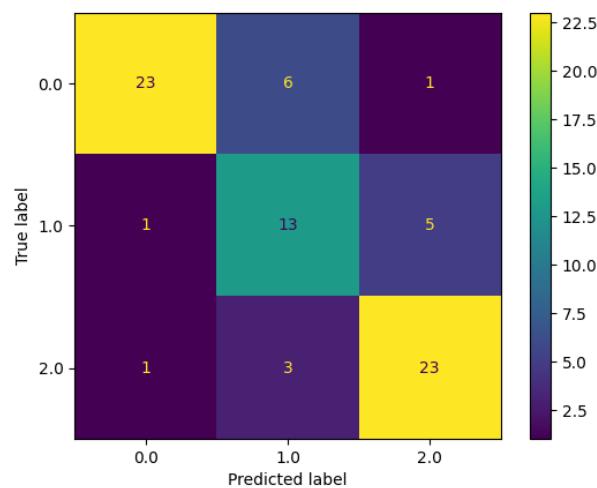


Figura 7.5: Tabella di confusione del modello Support Vector Machine con $C = 1.4$ e $\text{kernel} = \text{linear}$. La classe 0.0 indica la vittoria della squadra in casa, la classe 1.0 indica il pareggio tra le due squadre, la classe 2.0 indica la vittoria della squadra ospite.

	precision	recall	specificity
Home team win	0.92	0.77	0.95
Tie	0.59	0.68	0.84
Away team win	0.79	0.85	0.87
accuracy			0.78
AUC Score			0.82

Figura 7.6: Grafico delle misurazione registrate durante la fase di predizione del modello Support Vector Machine con $C = 1.4$ e $\text{kernel} = \text{linear}$.

sensibilità della classe vittoria della squadra ospite è pari a 0.85 ovvero, solo quattro osservazioni non vengono identificate di classe vittoria della squadra ospite. Si rileva una precisione pari a 0.79 ovvero delle 29 osservazioni classificate con la classe vittoria della squadra ospite solo sei risultano essere di un'altra classe. Buone prestazioni anche per la specificità della classe vittoria della squadra ospite che è pari a 0.87. L'algoritmo Support Vector Machine per quest'analisi si è rivelato particolarmente soddisfacente, nonostante la grande diversità da partita a partita e le poche osservazioni messe disponibile (380 partite).

7.5 Decision Tree

Con l'applicazione dell'algoritmo Decision Tree sono stati registrati dei buoni risultati durante la fase di predizione. Analogamente all'algoritmo K-Nearest-Neighbors, si è applicato la K-Fold Cross Validation con $k = 10$ per individuare i valori migliori per gli iperparametri.

Gli iperparametri valutati sono stati i seguenti:

- * `max_depth`. Indica la profondità massima dell'albero di decisione. Si sono verificati valori che vanno da 3 fino a 52 con un aumento unitario di 1. Si sottolinea che 52 è il numero di *feature* che sono presenti nel *dataset*. Inoltre, questo parametro permette di controllare l'*overfitting* ovvero, se le prestazioni peggiorano durante il training, i rami più profondi dell'albero vengono tagliati.
- * `criterion`. Indica la regola di decisione utilizzata per la creazione dell'albero di decisione. Si sono verificate le regole Gini Index e Cross Entropy.
- * `min_samples_split`. Indica il numero minimo di istanze richieste per dividere un nodo interno. Se la condizione non viene rispettata allora il nodo sarà una foglia attuando una potatura dell'albero di decisione.

Nella Figura 7.7 viene mostrato l'andamento della Cross Validation per ogni valore degli iperparametri. Dal grafico in alto a sinistra si nota che tra le due regole la migliore in termini di accuratezza media registrata nel *validation set* è stata la Cross Entropy. Nel grafico in alto a destra i valori 3,4,6,7 e 9, assunti dall'iperparametro `min_samples_split` hanno fatto registrare le accuratezze medie più alte.

Nel grafico in basso vi è un andamento abbastanza irregolare. Si nota un forte decadimento delle prestazioni con l'aumento del valore del limite della profondità, soprattutto con valori superiori a 5. Secondo la Cross Validation i valori migliori sono stati il

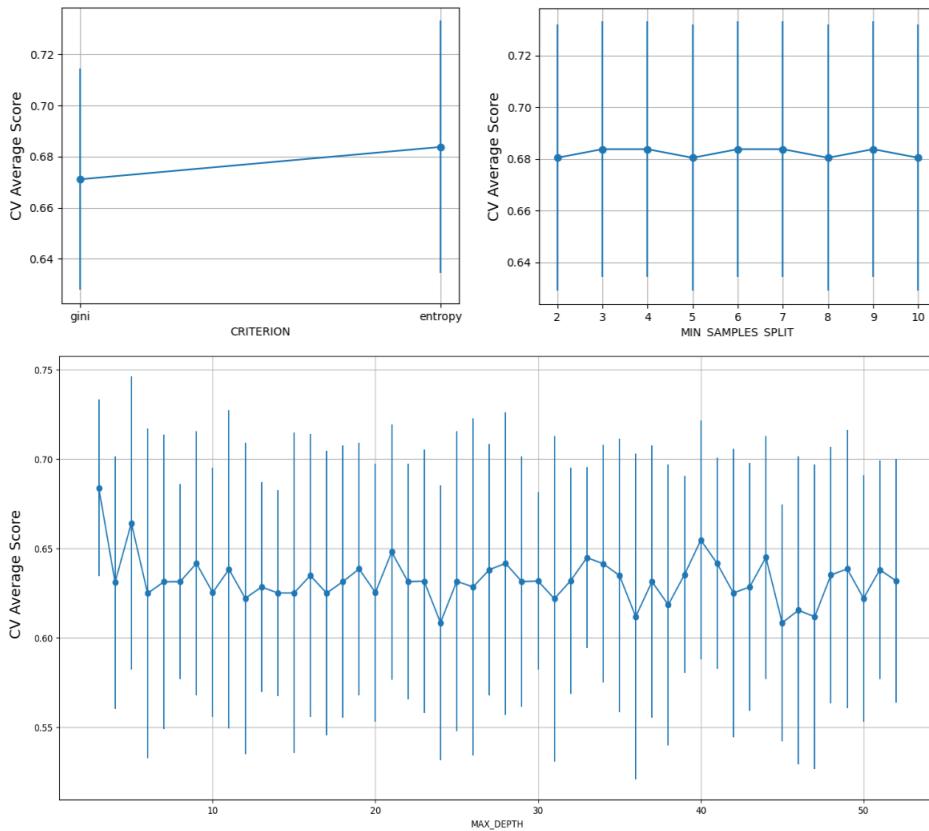


Figura 7.7: Il grafico in alto a sinistra indica l'andamento della media dell'accuratezza per ogni valore dell'iperparametro `criterion`. Il grafico in alto a destra indica l'andamento della media dell'accuratezza per ogni valore dell'iperparametro `min_samples_split`. Il grafico in basso indica l'andamento della media dell'accuratezza per ogni valore dell'iperparametro `max_depth`. Entrambi i grafici sono l'applicazione della Cross Validation con 10 fold per il modello Decision Tree.

valore 3 sia per l'iperparametro `max_depth` e sia per l'iperparametro `min_samples_split` e la Cross Entropy come regola di decisione da utilizzare. L'accuratezza ottenuta dal modello migliore nel *validation set* è stata di 0.684.

Nella fase di predizione si sono ottenute le predizioni mostrate nella Figura 7.8 con le relative metriche presentate nella Figura 7.9. Nonostante le poche osservazioni disponibili i risultati ottenuti sono buoni. Infatti, l'accuratezza delle predizioni è di 0.71. Il modello ha comunque qualche difficoltà a identificare le osservazioni della classe pareggio. Infatti, la sensibilità è pari a 0.26. Nonostante il modello etichetti un'osservazione con la classe pareggio in poche occasioni, esso sbaglia poco. Ci sono solo due osservazioni etichettate erroneamente con la classe pareggio, quindi la precisione è pari a 0.71. Ovviamente la specificità nella classe pareggio è molto alta dato che il modello poche volte sbaglia ad etichettare con la classe pareggio. Buoni risultati si ottengono sia per la classe vittoria della squadra in casa sia per la classe vittoria della squadra ospite. Infatti, per la prima classe c'è una sensibilità pari a 0.93, ovvero, soltanto due osservazioni di classe vittoria della squadra in casa non sono state identificate

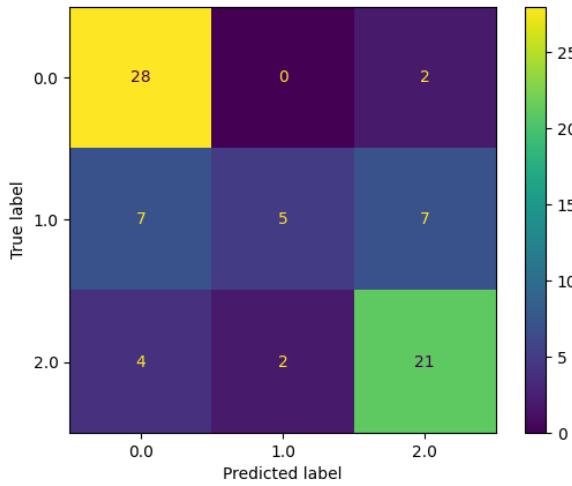


Figura 7.8: Tabella di confusione del modello Decision Tree con `max_depth` = 3, `min_samples_split` = 3 e `criterion` = entropy. La classe 0.0 indica la vittoria della squadra in casa, la classe 1.0 indica il pareggio tra le due squadre, la classe 2.0 indica la vittoria della squadra ospite.

come tali. Le prestazioni però calano nella precisione dato che undici osservazioni sono state etichettate erroneamente con la classe vittoria della squadra in casa, registrando una precisione pari a 0.72. Di conseguenza anche la specificità è calata rispetto alla sensibilità con un valore pari a 0.76. La classe vittoria della squadra ospite ha una sensibilità pari a 0.78. Infatti, sei osservazioni di classe vittoria della squadra ospite non sono state identificate come tali. Anche la precisione rimane in linea con le prestazioni della sensibilità con un valore pari a 0.70, in cui nove osservazioni su trenta sono state etichettate erroneamente con la classe vittoria della squadra ospite. Nonostante ciò, la specificità è pari a 0.82.

Dato che l'algoritmo Decision Tree non è un algoritmo a scatola chiusa (*black box*), è possibile capire quali sono state le sue scelte analizzando l'albero di decisione che ha prodotto per svolgere le sue predizioni. Nella Figura 7.10 viene mostrato l'albero

	precision	recall	specificity
Home team win	0.72	0.93	0.76
Tie	0.71	0.26	0.96
Away team win	0.70	0.78	0.81
accuracy			0.71
AUC Score			0.75

Figura 7.9: Grafico delle misurazioni registrate durante la fase di predizione del modello Decision Tree con `max_depth` = 3, `min_samples_split` = 3 e `criterion` = entropy.

di decisione prodotto e utilizzato durante la fase di predizione. Nell'albero mostrato

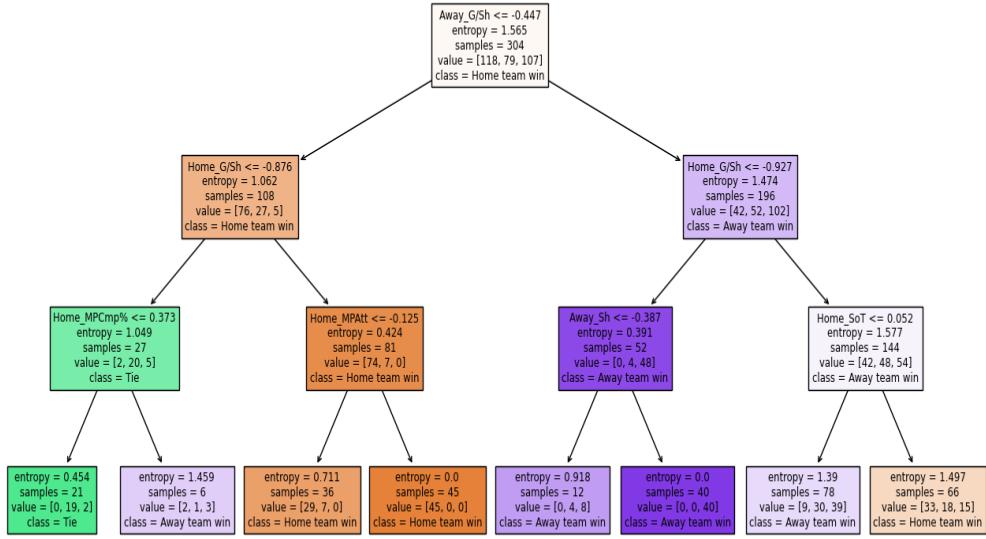


Figura 7.10: L'albero di decisione del modello Decision Tree con `max_depth` = 3 e `criterion` = `entropy`. Per ogni nodo non foglia c'è un test che indica quale ramo scegliere asseseconda del valore contenuto nell'attributo testato. Il parametro `entropy` indica l'entropia misurata. Il parametro `samples` indica il numero di istanze che soddisfano i test precedenti. Nel parametro `value` vengono riportati il numero di istanze presenti per ognuna delle tre classi. Il parametro `class` indica la classe di maggioranza nel nodo. Il colore indica la classe di maggioranza con una tonalità differente asseseconda della frequenza della classe di maggioranza.

In Figura 7.10, per ogni nodo non foglia c'è un test che indica quale ramo scegliere a seconda del valore contenuto nell'attributo testato. Il test consiste semplicemente nel verificare se l'attributo ha un valore minore o uguale oppure maggiore rispetto a una certa soglia. Questo tipo di test permette di gestire attributi con valori continui utilizzando una soglia di decisione scelta opportunamente dell'algoritmo. Per tutti i nodi è presente il parametro `entropy` che indica l'entropia misurata. Il parametro `samples` che indica il numero di istanze che soddisfano i test precedenti. Il parametro `value` che è il numero di istanze presenti per ognuna delle tre classi. Il parametro `class` che indica la classe di maggioranza nel nodo, mentre il colore indica la classe di maggioranza con una tonalità differente a seconda della frequenza della classe di maggioranza. Quando viene raggiunto un nodo foglia si classifica l'osservazione con la classe di maggioranza dato che non ci sono più attributi che permettono di distinguere le istanze.

Osservando l'albero, le *feature* legate ai tiri si confermano di grande importanza non

solo i modelli Bradley-Terry visti nel Capitolo 5, ma anche per il modello Decision Tree.

In conclusione, l'algoritmo Decision Tree ottiene delle buone prestazioni pur rimanendo semplice senza diventare troppo complesso.

7.6 Random Forest

Con l'applicazione dell'algoritmo Random Forest si sono registrati dei buoni risultati durante la fase di predizione. Analogamente all'algoritmo K-Nearest-Neighbors, si è applicato la K-Fold Cross Validation con $k = 10$ per individuare i valori migliori per gli iperparametri.

Gli iperparametri valutati sono stati i seguenti:

- * **criterion**. Indica la regola di decisione utilizzata per la creazione degli alberi di decisione. Si sono verificate le regole Gini Index e Cross Entropy.
- * **max_depth**. Indica la profondità massima degli alberi di decisione utilizzati. Si sono verificati valori che vanno da 3 fino a 52 con un aumento unitario di 1. Si sottolinea che 52 è il numero di *feature* che sono presenti nel *dataset*. Inoltre, questo parametro permette di controllare l'*overfitting* ovvero, se le prestazioni peggiorano durante il training, i rami più profondi dell'albero vengono tagliati.
- * **n_estimators**. Indica il numero massimo di classificatori impiegati per il Bagging. Si sono verificati valori che vanno da 3 fino a 126 classificatori con un aumento unitario di 1.
- * **min_samples_split**. Indica il numero minimo di istanze richieste per dividere un nodo interno. Se la condizione non viene rispettata allora il nodo sarà una foglia attuando una potatura dell'albero di decisione.

Nella Figura 7.11 viene mostrato l'andamento della Cross Validation per ogni valore degli iperparametri. Dal grafico in alto a sinistra si nota che tra le due regole la migliore in termini di accuratezza media registrata nel *validation set* è stata la Cross Entropy. Nel grafico in alto a destra con il valore 7 nell'iperparametro **min_samples_split** il modello ottiene l'accuratezza media più alta. Nel grafico al centro vi è un andamento abbastanza irregolare. Si nota che con il valore 17 dell'iperparametro **max_depth** vi è un aumento improvviso del valore dell'accuratezza media del modello. Infine, anche nel grafico in basso vi è un andamento abbastanza irregolare, nonostante un'iniziale tendenza di aumento dell'accuratezza. Quando l'iperparametro **n_estimators** assume il valore 65 vi è un picco di prestazione dell'accuratezza media.

Secondo la Cross Validation i valori migliori sono stati per l'iperparametro **criterion** la Cross Entropy, il valore 17 per l'iperparametro **max_depth**, per l'iperparametro **min_samples_split** il valore 7 e per l'iperparametro **n_estimators** il valore 65 come numero di classificatori da impiegare. L'accuratezza ottenuta dal modello migliore nel *validation set* è stata di 0.753.

Nella fase di predizione si sono ottenute le predizioni mostrate nella Figura 7.12 con le relative metriche presentate nella Figura 7.13.

Nonostante le poche osservazioni disponibili i risultati ottenuti sono buoni. Infatti, l'accuratezza delle predizioni è di 0.72. Anche in questo modello si sono riscontrate alcune difficoltà nell'identificare le istanze della classe pareggio. La sensibilità rilevata sulla classe pareggio è pari a 0.37, perché solo sette osservazioni su 19 osservazioni di

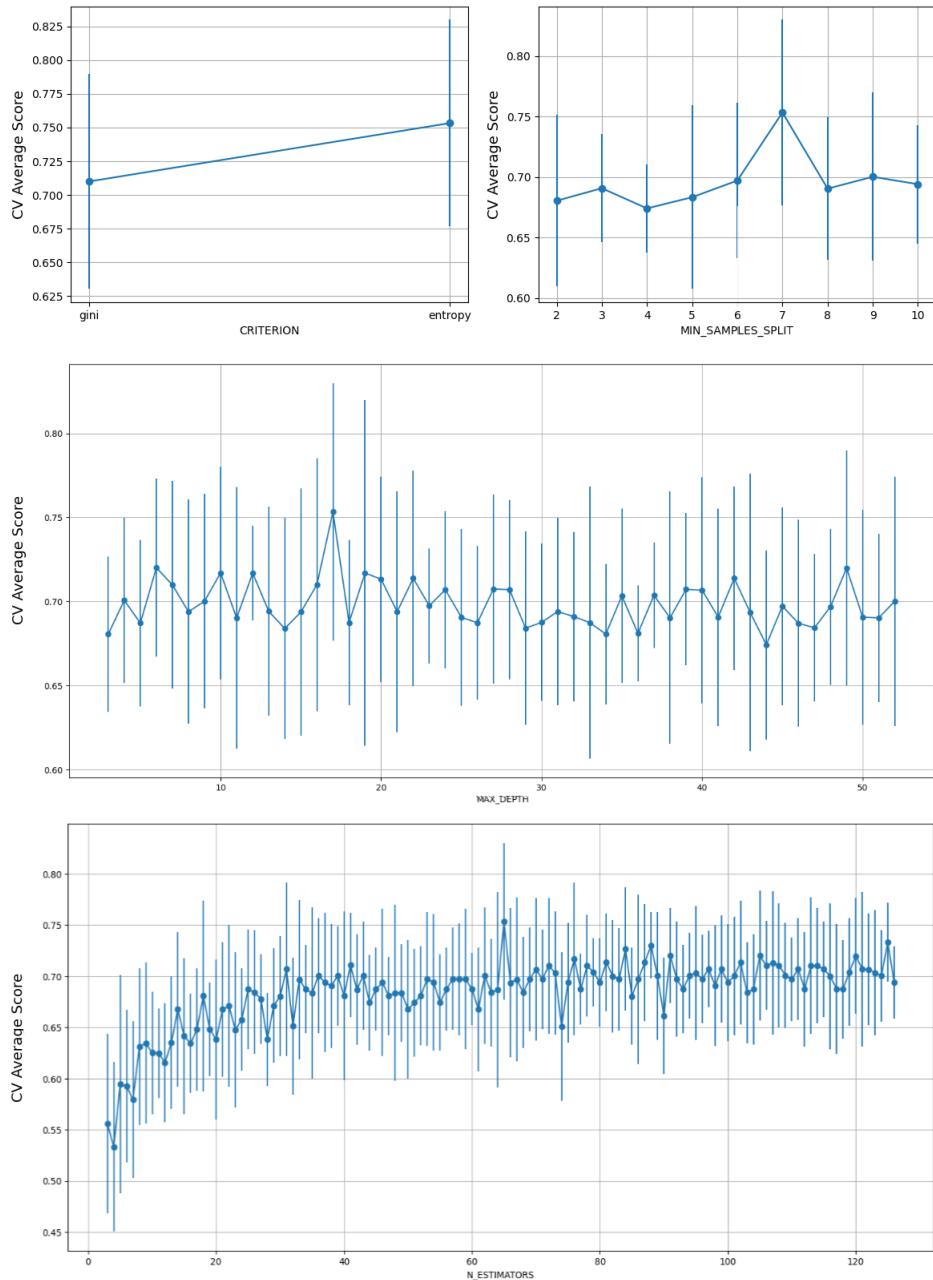


Figura 7.11: Il grafico in alto a sinistra indica l'andamento della media dell'accuratezza per ogni valore dell'iperparametro `criterion`. Il grafico in alto a destra indica l'andamento della media dell'accuratezza per ogni valore dell'iperparametro `min_samples_split`. Il grafico al centro indica l'andamento della media dell'accuratezza per ogni valore dell'iperparametro `max_depth`. Il grafico in basso indica l'andamento della media dell'accuratezza per ogni valore dell'iperparametro `n_estimators`. Tutti i grafici sono l'applicazione della Cross Validation con 10 fold per il modello Random Forest.

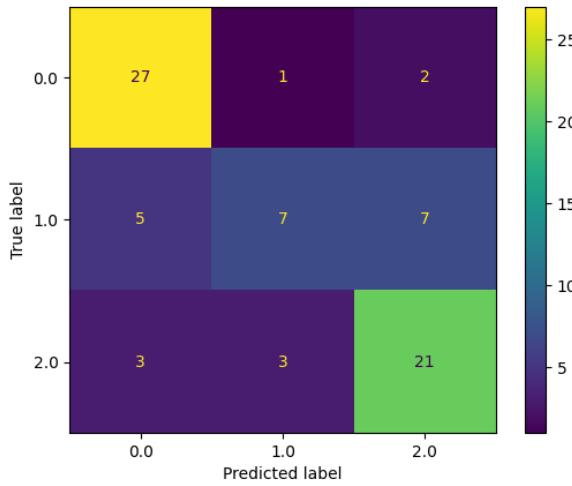


Figura 7.12: Tabella di confusione del modello Random Forest con `criterion = gini`, `max_depth = 5`, `n_estimators = 102` e `min_samples_split = 9`. La classe 0.0 indica la vittoria della squadra in casa, la classe 1.0 indica il pareggio tra le due squadre, la classe 2.0 indica la vittoria della squadra ospite.

classe pareggio sono state identificate con la classe pareggio. Tuttavia, le prestazioni migliorano con la precisione del modello nell’etichettare un’istanza con la classe pareggio, infatti delle undici osservazioni classificate con la classe pareggio solo quattro sono risultate essere di una classe differente. Di conseguenza anche la specificità sulla classe pareggio ha un valore alto pari a 0.92. Si registrano prestazioni migliori con le classi vittoria della squadra in casa e vittoria della squadra ospite. Per la classe vittoria della squadra in casa si ha una sensibilità pari a 0.90 ovvero, solo tre osservazioni su 30 non sono state riconosciute di classe vittoria della squadra in casa. Tuttavia, si registra un calo di prestazioni nella precisione nell’etichettare correttamente le istanze di classe vittoria della squadra in casa. Infatti, si registra un valore pari a 0.77 perché otto osservazioni sono state classificate erroneamente con la classe vittoria della squadra in

	precision	recall	specificity
Home team win	0.77	0.90	0.82
Tie	0.64	0.37	0.92
Away team win	0.70	0.78	0.81
accuracy			0.72
AUC Score			0.77

Figura 7.13: Grafico delle misurazione registrate durante la fase di predizione del modello Random Forest con `criterion = gini`, `max_depth = 5`, `n_estimators = 102` e `min_samples_split = 9`.

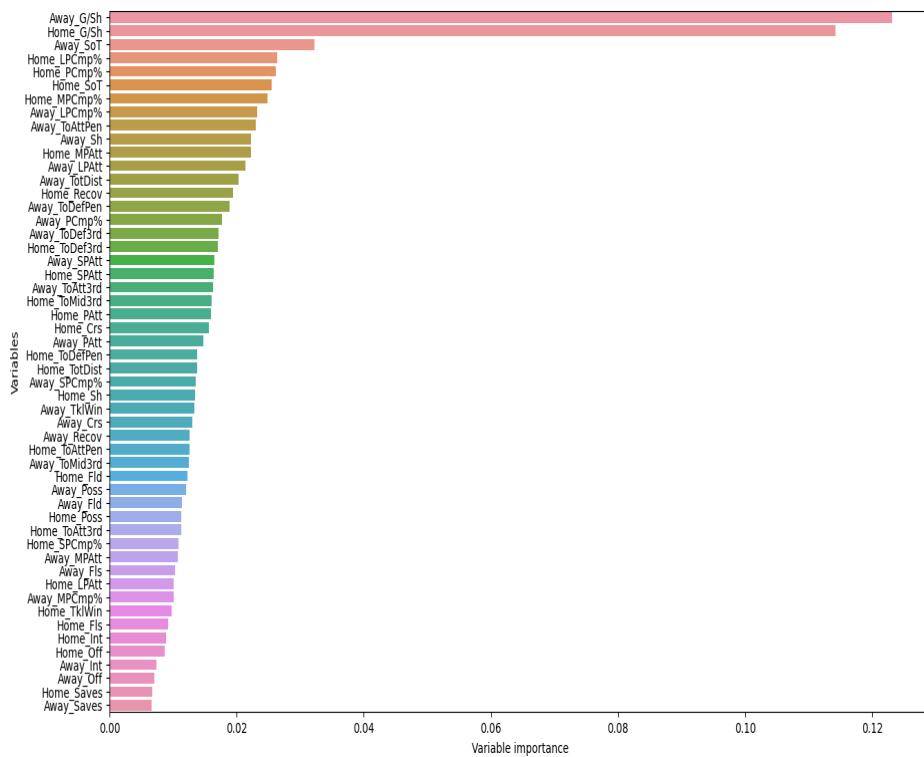


Figura 7.14: Il grafico riporta per ogni *feature* la sua importanza basata sull’impurità.

casa. Di conseguenza anche la specificità sulla classe della vittoria della squadra in casa cala, raggiungendo un valore pari a 0.82. Buone prestazioni del modello si rilevano nell’identificare le istanze di classe vittoria della squadra ospite, con una sensibilità pari a 0.78. Purtroppo, si registra una precisione più bassa ovvero pari a 0.70, con nove osservazioni classificate erroneamente con la classe vittoria della squadra ospite. Infine, la specificità del modello sulla classe vittoria della squadra ospite è pari a 0.81. Dato che l’algoritmo Random Forest utilizza i classificatori Decision Tree è possibile capire quali sono le *features* più importanti secondo il modello per produrre una predizione dell’esito di una partita. Perciò, nella Figura 7.14 viene mostrato per ogni *feature* la sua importanza misurata sull’impurità. Osservando i risultati ottenuti essi sono coerenti con quanto visto nell’albero del Decision tree. Le *features* legate ai tiri si confermano di grande importanza, così come i passaggi completati di tutti i tipi. Invece, poco significativi sono il numero di parate ma anche il numero di fuorigioco, intercetti e contrasti vinti. Perciò, in sostanza sembrano essere poco significative le *features* legate alla difesa, fatta eccezione per il numero di recuperi che sembra essere significativo. Inoltre, le *features* riguardanti i falli quindi numero di falli subiti e numero di falli fatti sono poco importanti per il modello, inoltre come già visto nei modelli BT anche qui il possesso della palla è poco importante. Per quanto riguarda le *features* riguardanti i tocchi nelle diverse zone del campo, esse sono mediamente importanti. Non sorprendentemente, la più importante è il numero di tocchi fatti nell’area di rigore avversaria.

In conclusione, nonostante le poche osservazioni disponibili, l’algoritmo Random

Forest ottiene nel complesso delle buone prestazioni, nonostante qualche difficoltà nell'identificazione dei pareggi.

7.7 AdaBoost

Con l'applicazione dell'algoritmo AdaBoost si sono registrati dei buoni risultati durante la fase di predizione. Analogamente all'algoritmo K-Nearest-Neighbors, si è applicato la K-Fold Cross Validation con $k = 10$ per individuare i valori migliori per gli iperparametri.

Gli iperparametri valutati sono stati i seguenti:

- * `n_estimators`. Indica il numero massimo di classificatori impiegati per il Boosting. Si sono verificati valori che vanno da 3 fino a 126 classificatori con un aumento unitario di 1.
- * `learning_rate`. Indica il peso applicato ad ogni classificatore per ogni iterazione del Boosting. Più precisamente, quanta porzione dell'aggiornamento dei pesi verrà applicata sui pesi per la prossima iterazione. Perciò, l'`learning_rate` permette di controllare quanto velocemente procederà l'addestramento. Si sono verificati valori che vanno da 0.1 fino a 1.0 con un aumento unitario di 0.1. Più l'`learning_rate` è elevato più sarà il contributo di ciascun classificatore e la velocità di addestramento.

Esiste un *trade-off* tra i parametri `learning_rate` e `n_estimators`. Nella Figura 7.15 viene mostrato l'andamento della Cross Validation per ogni valore degli iperparametri. Dal grafico, notiamo un andamento abbastanza irregolare per quasi tutti i tassi d'ap-

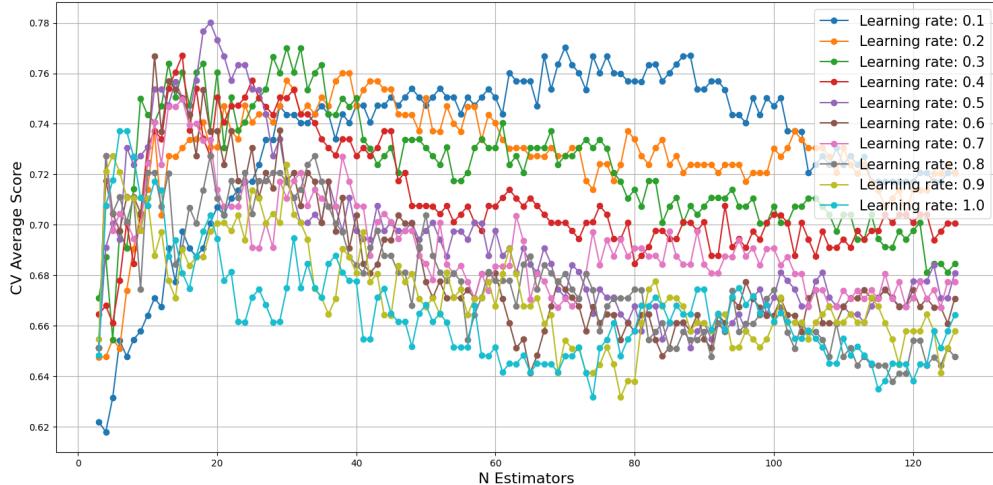


Figura 7.15: Grafico dell'andamento della media dell'accuratezza per ogni valore dell'iperparametro `n_estimators` e per ogni valore dell'iperparametro `learning_rate` utilizzati durante l'applicazione della Cross Validation con 10 fold per il modello AdaBoost. Ogni punto utilizza un certo numero classificatori mentre il colore indica il peso dell'`learning rate`.

prendimento, eccetto per il modello con il tasso d'apprendimento pari a 0.1, il quale

inizialmente ottiene prestazioni migliori con il salire del numero dei classificatori. Tutti i tassi d'apprendimento con un numero elevato di classificatori calano il loro valore di accuratezza nel *validation set*. Nonostante il modello con `learning_rate` pari a 0.1 abbia costantemente valori più alti, il modello con `learning_rate` pari a 0.5 ha segnato un'accuratezza più alta anche se, con l'aumentare del numero di classificatori, le sue prestazioni calano di molto.

Perciò, secondo la Cross Validation i valori migliori sono stati il valore 19 per l'iperparametro `n_estimators` e il valore 0.5 per l'iperparametro `learning_rate`. L'accuratezza ottenuta dal modello migliore nel *validation set* è stata di 0.78.

Nella fase di predizione si sono ottenute le predizioni mostrate nella Figura 7.16 con le relative metriche presentate nella Figura 7.17.

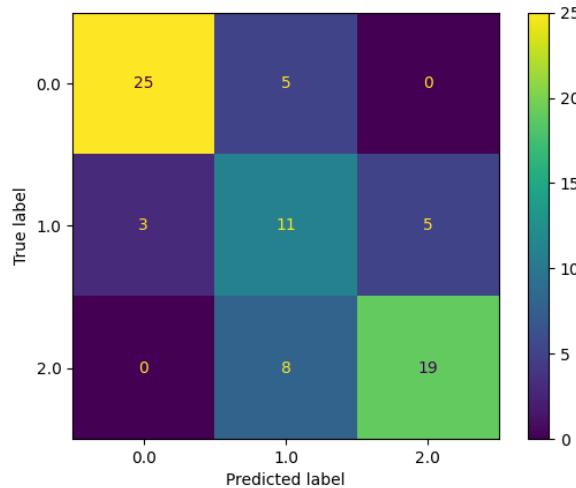


Figura 7.16: Tabella di confusione del modello AdaBoost con `n_estimators` = 19 e `learning_rate` = 0.5. La classe 0.0 indica la vittoria della squadra in casa, la classe 1.0 indica il pareggio tra le due squadre, la classe 2.0 indica la vittoria della squadra ospite.

	precision	recall	specificity
Home team win	0.89	0.83	0.93
Tie	0.46	0.58	0.77
Away team win	0.79	0.70	0.89
accuracy			0.72
AUC Score			0.78

Figura 7.17: Grafico delle misurazioni registrate durante la fase di predizione del modello AdaBoost con `n_estimators` = 19 e `learning_rate` = 0.5.

Nonostante le poche osservazioni disponibili i risultati ottenuti sono buoni. Infatti, l'accuratezza delle predizioni è di 0.72. Anche questo modello ha delle difficoltà a identificare le istanze di classe pareggio. La sensibilità del modello sulla classe pareggio è pari a 0.58 ovvero, undici osservazioni su 19 di classe pareggio sono state identificate come tali. Purtroppo, il modello non è abbastanza soddisfacente nell'etichettare le istanze con la classe pareggio perché la precisione è pari a 0.46. Di conseguenza anche la specificità ne risente, con un valore pari a 0.77. Tuttavia, le prestazioni sono decisamente migliori con le classi vittoria della squadra in casa e vittoria della squadra ospite. Infatti, per la classe vittoria della squadra in casa la sensibilità è pari a 0.83, cioè solo cinque osservazioni su 30 non sono state riconosciute di classe vittoria della squadra. Analogamente, la specificità della classe vittoria della squadra in casa ha un valore molto alto ovvero, 0.93. Inoltre, il modello risulta essere molto preciso nel classificare correttamente le istanze con la classe vittoria della squadra in casa. Infatti, il modello ha commesso solo tre errori ottenendo una precisione pari a 0.89. Buone prestazioni si sono registrate anche nell'identificare le istanze di classe vittoria della squadra ospite da parte del modello. La sensibilità è pari a 0.70, mentre la precisione è pari a 0.79, con solo cinque osservazioni classificate erroneamente con la classe vittoria della squadra ospite. Buona anche la specificità del modello sulla classe vittoria della squadra ospite che risulta essere pari a 0.89.

In conclusione, nonostante le poche osservazioni messe a disposizione, l'algoritmo AdaBoost ottiene nel complesso delle buone prestazioni nonostante qualche difficoltà nell'identificazione dei pareggi.

8 | MODELLO BRADLEY-TERRY PER IL NUMERO DI GOL DI SCARTO

In questo capitolo si illustreranno i risultati ottenuti con il modello Bradley-Terry (4.12) con una variabile risposta a cinque categorie legate al numero di gol segnati dalle squadre durante le partite.

8.1 Premessa

Nel Capitolo 5 si è sottolineato che i risultati ottenuti non tengono in considerazione le variabili esplicative gol fatti GF e gol subiti GA, a causa della non convergenza del modello. Per poter utilizzare queste due variabili esplicative, su ispirazione del lavoro di ricerca di schauberger2017, si è deciso di cambiare il numero di categorie della variabile risposta Y ovvero, di utilizzare cinque categorie invece di tre, in base al numero di gol segnati e subiti, nello specifico:

$$Res = \begin{cases} 1 & \text{se la squadra in casa batte la squadra ospite con due gol di scarto,} \\ 2 & \text{se la squadra in casa batte la squadra ospite con un gol di scarto,} \\ 3 & \text{se la partita termina con un pareggio,} \\ 4 & \text{se la squadra ospite batte la squadra in casa con un gol di scarto,} \\ 5 & \text{se la squadra ospite batte la squadra in casa con due gol di scarto.} \end{cases} \quad (8.1)$$

8.2 Modello Bradley-Terry con $Y=5$ e Lasso

I risultati ottenuti dal modello (4.12) con $Y = 5$ sono presentati nella Figura 8.1 e nelle Tabelle 8.1 e 8.2. Nella Figura 8.1 si può notare che le stime dell'abilità delle squadre sono molto simile a quelle riportate nella Figura 5.3. Infatti, nelle prime e nelle ultime quattro posizioni la stima e il piazzamento rimangono invariati. Cambiano le stime di Atalanta, Lazio e Roma: l'Atalanta aumenta la propria stima e quindi viene ancora più sovrastimata, mentre per Lazio e Roma l'abilità stimata cala. Anche Fiorentina e Hellas Verona hanno un aumento dell'abilità stimata, anche se l'Hellas Verona viene sovrastimata rispetto alla Fiorentina. Infine, anche l'Empoli aumenta la propria abilità stimata, tanto da essere stimato più forte del Bologna.

Le stime dei parametri ottenute sono molto simili alle stime presentate nelle Tabelle 5.2 e 5.3. Viene confermata la poca importanza della percentuale dei passaggi completi PCmp%, del numero di passaggi tentati PAtt, del numero di tocchi nella tre quarti di difesa ToDefPen e della distanza percorsa con la palla TotDist. Anche qui viene confermato, seppur con una leggera diminuzione, il vantaggio nel giocare in casa Home che è pari a 0.272. Per quanto riguarda il possesso palla Poss viene confermato che per la maggior parte delle squadre non ha alcuna influenza sull'esito della partita.

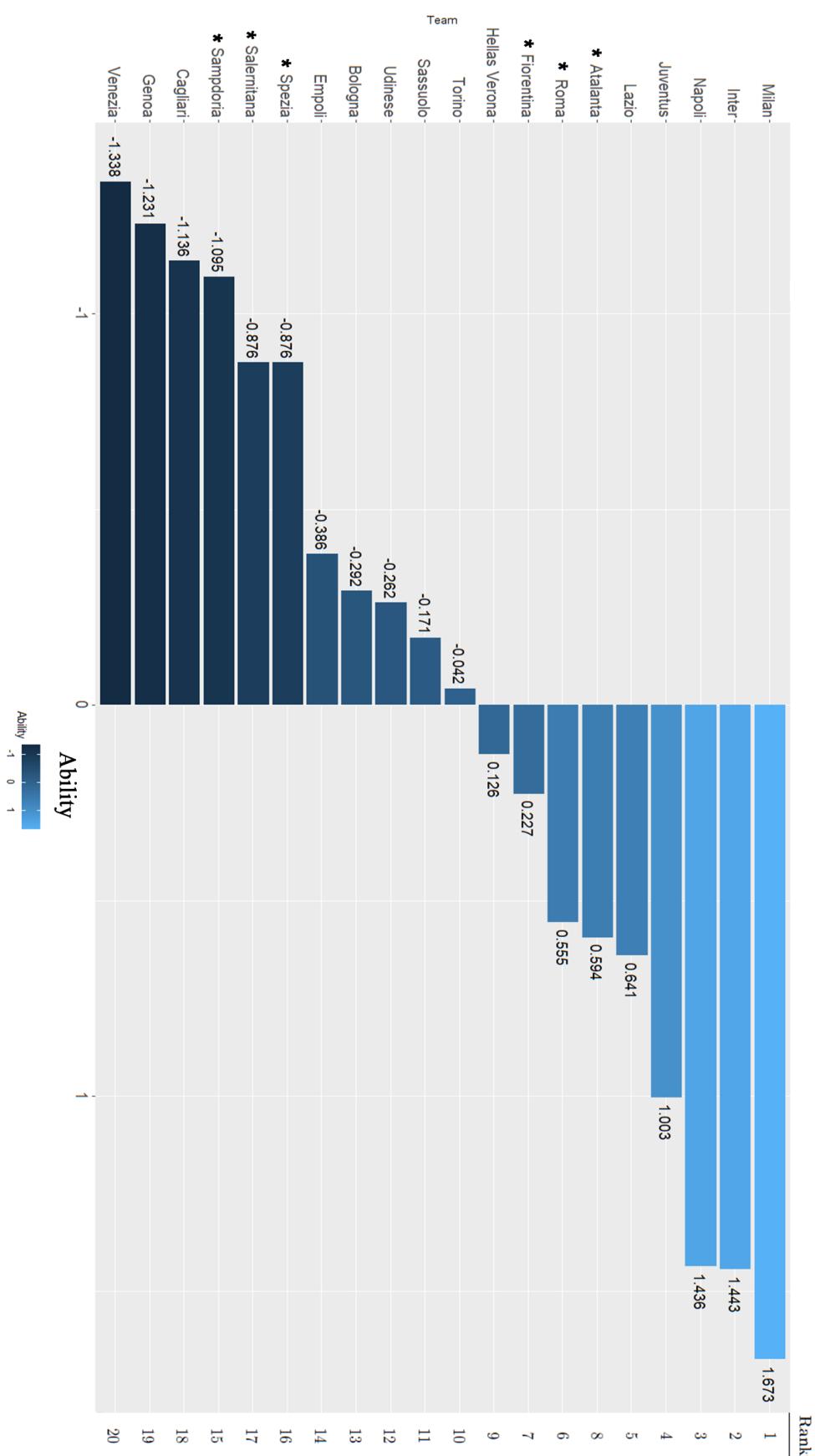


Figura 8.1: Barplot che indica per ogni squadra l'abilità stimata dal modello (4.12) con $Y = 5$. Viene indicato con un asterisco le squadre con un piazzamento stimato diverso da quello reale anche esso riportato a destra del grafico.

Covariata	Stima	Squadra
Home	0.272	Tutti
Poss	0.000	Tutti
Sh	0.446	Tutti
SoT	0.779	Atalanta, Cagliari, Empoli, Genoa, Verona, Inter, Juventus, Lazio, Milan, Napoli, Roma, Salernitana, Sampdoria, Sassuolo, Spezia, Torino, Venezia
SoT	0.393	Fiorentina
SoT	0.384	Bologna e Udinese
G/Sh	1.072	Tutti
Saves	0.373	Tutti
PAtt	0.000	Tutti
PCmp%	0.000	Tutti
SPAtt	0.182	Napoli
SPAtt	0.132	Juventus
SPAtt	0.000	Tutti tranne Napoli e Juventus
SPCmp%	0.047	Udinese
SPCmp%	0.000	Tutti tranne Genoa e Udinese
SPCmp%	-0.030	Genoa
MPAtt	0.000	Tutti
MPCmp%	-0.188	Tutti tranne Bologna, Cagliari, Genoa e Spezia
MPCmp%	-0.201	Bologna, Cagliari e Spezia
MPCmp%	-0.262	Genoa
LPAtt	0.266	Lazio
LPAtt	0.120	Tutti tranne Lazio
LPCmp%	0.236	Hellas Verona
LPCmp%	0.000	Tutti tranne Verona

Tabella 8.1: Stime delle covariate stimate dal modello (4.12) con $Y = 5$.

Covariata	Stima	Squadra
ToDefPen	0.128	Tutti
ToDef3rd	0.000	Tutti
ToMid3rd	0.144	Tutti
ToAtt3rd	-0.104	Tutti
ToAttPen	0.000	Tutti tranne Atalanta
ToAttPen	-0.387	Atalanta
TotDist	0.000	Tutti
Fls	0.270	Sampdoria
Fls	0.238	Bologna
Fls	0.000	Tutti tranne Bologna e Sampdoria
Fld	0.130	Udinese
Fld	0.000	Tutti tranne Udinese
Off	0.178	Hellas Verona
Off	-0.030	Tutti tranne Hellas Verona, Inter e Juventus
Off	-0.034	Inter e Juventus
Crs	0.100	Torino
Crs	-0.226	Tutti tranne Milan, Roma, Torino, Atalanta e Napoli
Crs	-0.333	Milan e Roma
Crs	-0.386	Napoli
Crs	-0.423	Atalanta
Int	0.019	Tutti
TklWin	0.083	Tutti tranne Genoa e Venezia
TklWin	0.000	Genoa e Venezia
Recov	0.000	Venezia e Genoa
Recov	-0.044	Cagliari
Recov	-0.063	Tutti tranne Cagliari, Genoa, Udinese e Venezia
Recov	-0.212	Udinese

Tabella 8.2: Stime delle covariate stimate dal modello (4.12) con $Y = 5$.

A differenza del modello (4.12) con $Y = 3$, però, Lazio e Torino non ricevono alcun beneficio.

Viene confermata l'associazione positiva con la probabilità di vittoria sia del numero di tiri **Sh**, sia del numero di parate **Saves**. Analogamente, anche la stima per il numero di tiri in porta **SoT** è fortemente associata all'aumento della probabilità di vittoria. Si segnalano delle variazioni delle stime del parametro **SoT**. Ora, la maggior parte delle squadre ha una stima del parametro pari a 0.779, ad eccezione di tre squadre. Dalla Figura 8.2 è possibile individuare tre clusters: il più grande contiene la maggioranza delle squadre, un cluster contiene solo la Fiorentina con un percorso leggermente più basso, un intero cluster contiene Bologna e Udinese. Si riconferma la variabile

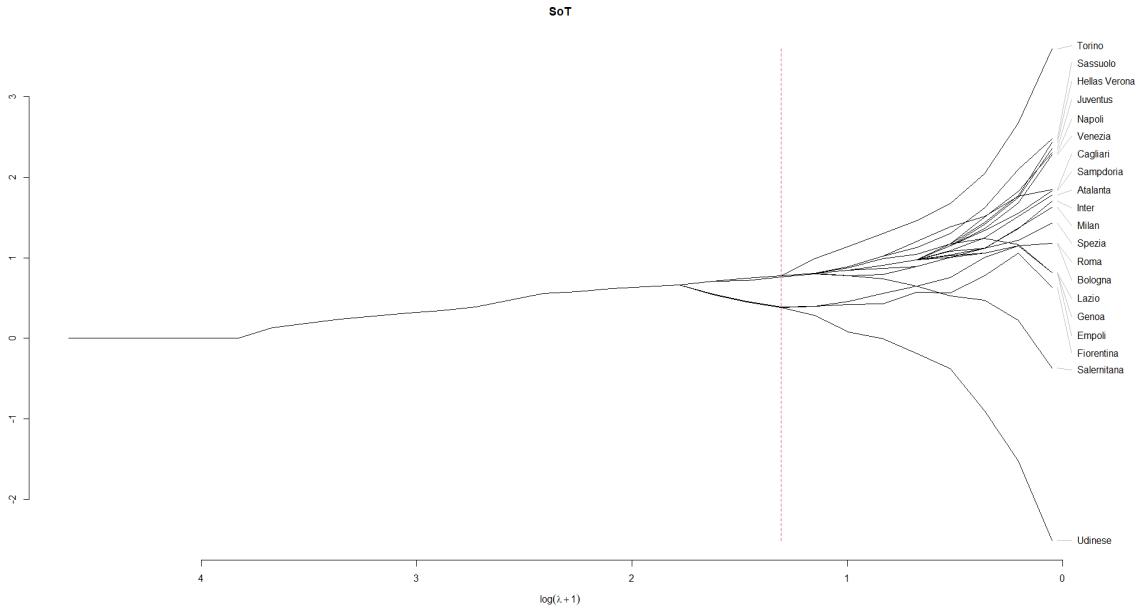


Figura 8.2: Grafico che riporta l'andamento stimato dal modello (4.12) con $Y = 5$ della stima del numero di tiri in porta per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

esplicativa **G/Sh** con la maggiormente associata all'esito della partita.

Per quanto riguarda le variabili legate ai passaggi non ancora illustrate, viene riconfermato che il numero di passaggi corti tentati **SPAtt** non risulta essere associato alla probabilità di vittoria per tutte le squadre, ad eccezione di Napoli e Juventus. Nella Figura 8.3 è possibile notare il cluster con andamento nullo contenente quasi tutte le squadre e più in su i clusters contenenti, rispettivamente, Juventus e Napoli. La percentuale di passaggi corti completati **SPCmp%**, invece, ha un importante aumento della stima per il Genoa, mentre solo per l'Udinese è associato un aumento della probabilità di vittoria. Per le restanti squadre, **SPCmp%** non ha alcuna associazione con l'esito della partita. Tale risultato è presentato nella Figura 8.4 contenente tre clusters, ognuno per una differente stima. Il numero di passaggi medi tentati **MPAtt** non ha alcuna associazione con l'esito della partita. Rimane ancora associata a una diminuzione della probabilità di vittoria la percentuale di passaggi medi riusciti **MPCmp%** per tutte le squadre. Dalla Figura 8.5 è possibile individuare tre clusters. Il cluster con

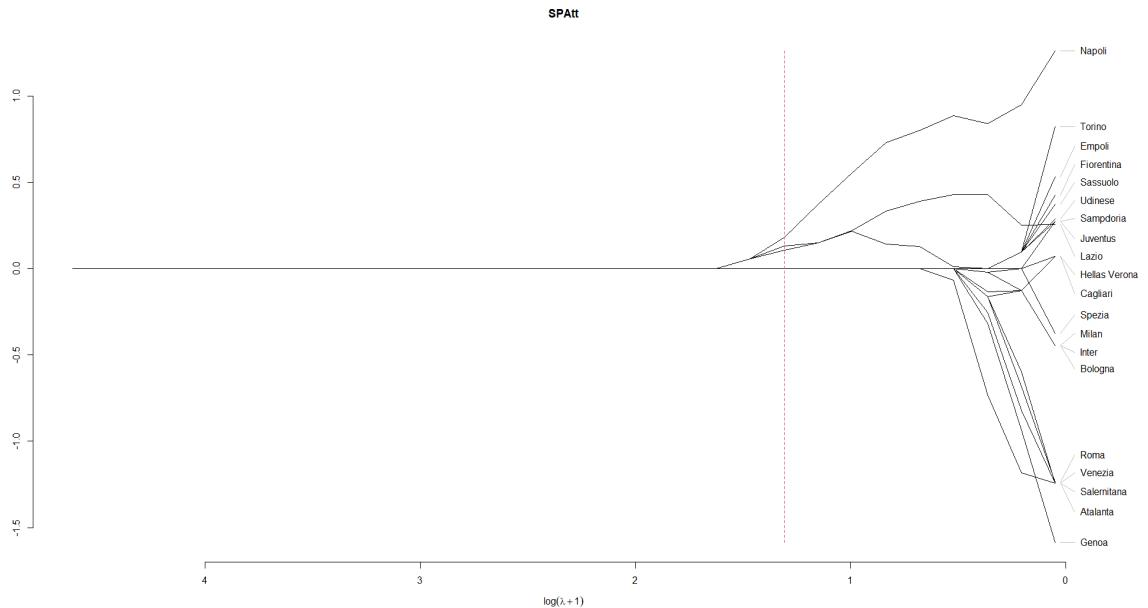


Figura 8.3: Grafico che riporta l'andamento stimato dal modello (4.12) con $Y = 5$ della stima del numero di passaggi corti tentati per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

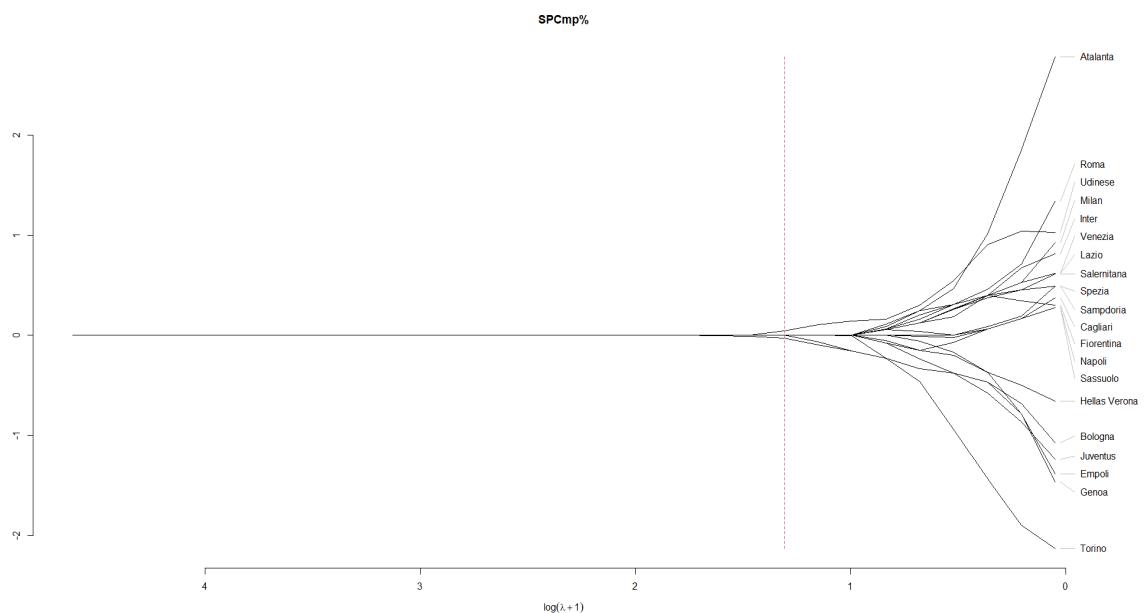


Figura 8.4: Grafico che riporta l'andamento stimato dal modello (4.12) con $Y = 5$ della stima della percentuale di passaggi corti riusciti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

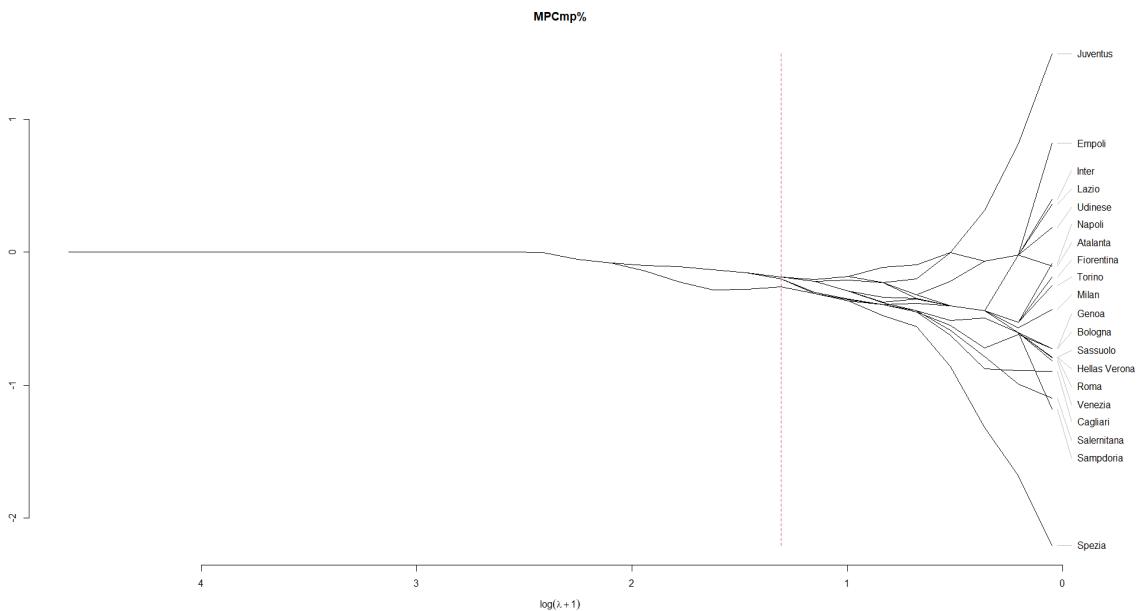


Figura 8.5: Grafico che riporta l'andamento stimato dal modello (4.12) con $Y = 5$ della stima della percentuale di passaggi medi riusciti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

il percorso meno negativo contiene quasi tutte le squadre, il cluster immediatamente sotto contiene le squadre Bologna, Cagliari e Spezia, mentre il cluster con il percorso associato più negativo contiene il Genoa. Per il numero di passaggi lunghi tentati LPAtt c'è un aumento della stima per tutte le squadre, in particolare per la Lazio. Unica variazione che si segnala riguarda la percentuale di passaggi lunghi riusciti LPCmp% che per il Bologna non è più associata con una diminuzione delle probabilità di vittoria. Per quanto riguarda le variabili legate al possesso, non ci sono rilevanti differenze rispetto alle stime ottenute con il modello (4.12) con $Y = 3$.

Si hanno delle differenze con la stima del numero di falli subiti F1s e fatti F1d rispetto alle stime del modello (4.12) con $Y = 3$. Infatti, solo per Bologna e Sampdoria F1s è associato a un aumento della probabilità di vittoria, mentre per le restanti squadre non vi è alcuna associazione. Nella Figura 8.6 è possibile visualizzare l'andamento dei due clusters che contengono, rispettivamente, Bologna e Sampdoria, entrambi con percorsi positivi. È rappresentato anche il cluster contenente quasi tutte le classi, ma con un percorso nullo. Anche per F1d vi è lo stesso esito con la differenza che, al posto di Bologna e Sampdoria, c'è l'Udinese.

Per quanto riguarda il fuorigioco Off, ora solo per l'Hellas Verona vi è una stima positiva. Per tutte le altre squadre la stima è negativa, in particolare per Inter e Juventus.

Ancora una volta il numero di cross Crs si conferma associato a una diminuzione della probabilità di vittoria. In questo caso abbiamo una maggior diminuzione per quasi tutte le squadre. Nella Figura 8.7 è possibile notare il cluster con andamento positivo, il quale contiene il Torino. Sotto al cluster del Torino ci sono quattro cluster tutti associati a percorsi negativi. Il cluster con il percorso meno negativo contiene quasi

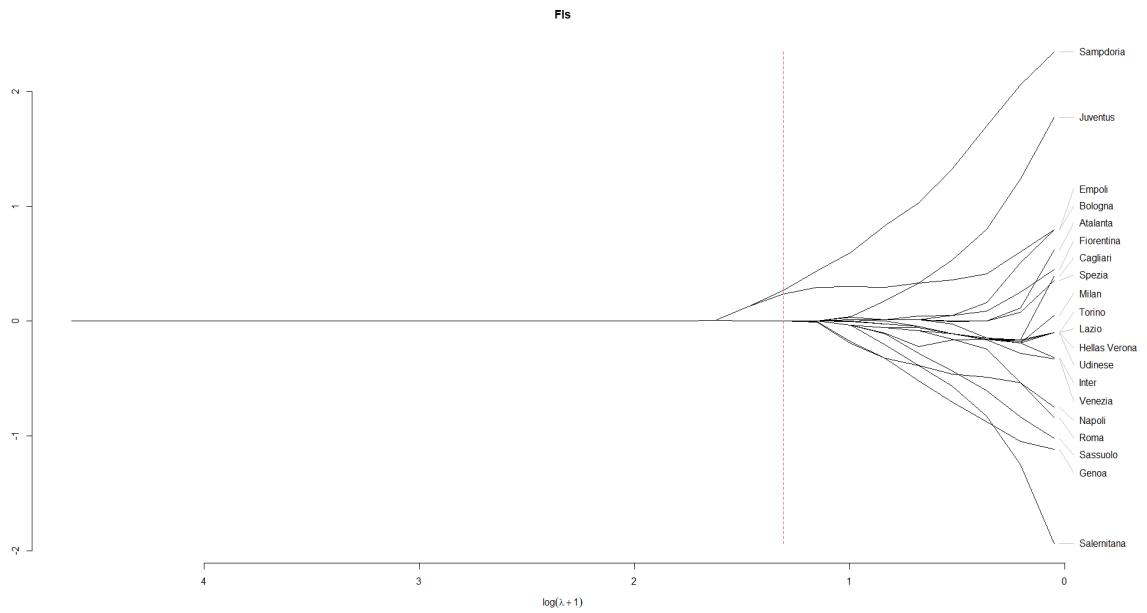


Figura 8.6: Grafico che riporta l'andamento stimato dal modello (4.12) con $Y = 5$ della stima del numero di falli subiti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

tutte le squadre. Subito sotto c'è il cluster che contiene Milan e Roma, mentre con un percorso leggermente più negativo, il cluster contenente il Napoli e, infine, con il percorso più negativo, il cluster che contiene l'Atalanta. Per quanto riguarda le variabili esplicative difensive, il numero di intercetti **Int** e il numero di contrasti vinti **TklWin** sono ancora associati ad un aumento della probabilità di vittoria. Viceversa, il numero di recuperi **Recov** si associa ad una diminuzione della probabilità di vittoria per tutte le squadre eccetto per Venezia e Genoa, per le quali non vi è alcuna associazione con l'esito della partita. Tale comportamento è mostrato nella Figura 8.8.

Ora che $Y = 5$ ci sono due soglie in più, per cui la stima delle soglie θ_1 , θ_2 , θ_3 e θ_4 vale, rispettivamente, -3.114, 3.114, -1.094 e 1.094.

Inoltre, i risultati ottenuti sono frutto di un parametro di tuning λ pari a 1.308 scelto attraverso la K-Fold Cross Validation. Nella Figura 8.9 è mostrato l'andamento delle prestazioni del modello su tutti i valori assunti dal parametro di tuning λ durante l'operazione di K-Fold Cross Validation. La K-Fold Cross Validation utilizza 10 gruppi ($k = 10$) e trenta valori diversi per *lambda*. Successivamente, i risultati ottenuti dal modello applicando i trenta diversi valori del parametro di tuning, sono stati confrontati in termini di RPS. Si nota che con il diminuire della penalizzazione il modello registra una RPS che migliora fino a quando la penalizzazione diventa troppo debole, causando un peggioramento delle prestazioni. Si è scelto perciò, il λ indicato dalla linea rossa tratteggiata.

Per riassumere, si analizzano i percorsi delle norme L2 che rappresentano l'importanza complessiva dei singoli effetti delle covariate. Tali percorsi sono visibili nella Figura 8.10. Gli andamenti sono molto simili a quelli visti nella Figura 5.18. Infatti **G/Sh**, **SoT** e **Sh** si confermano ancora fortemente associate all'esito della partita. Analogamente

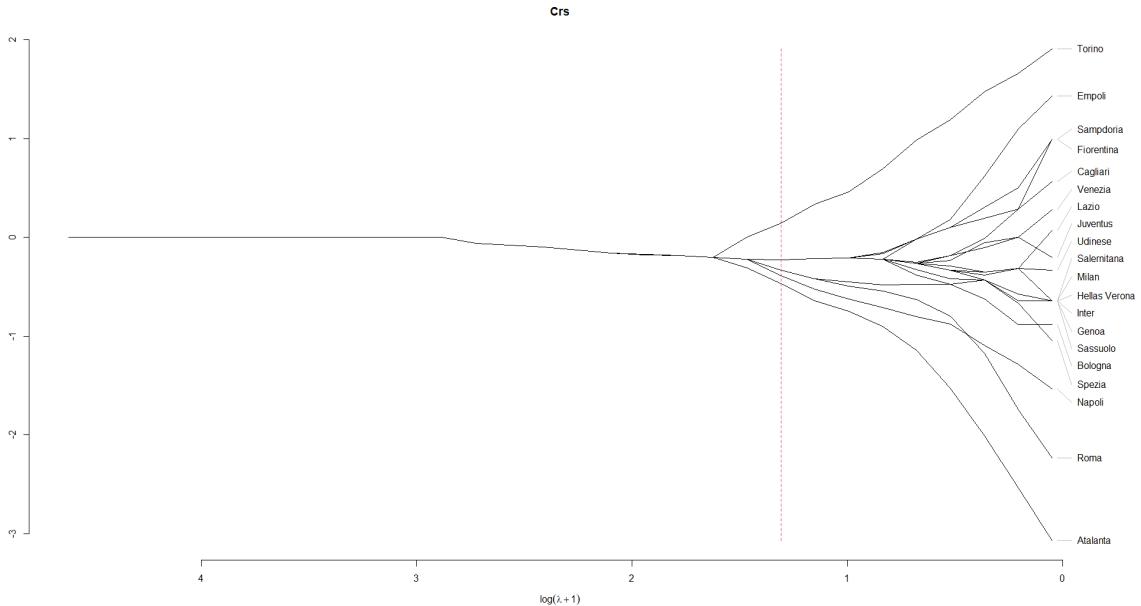


Figura 8.7: Grafico che riporta l'andamento stimato dal modello (4.12) con $Y = 5$ della stima del numero di cross fatti per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

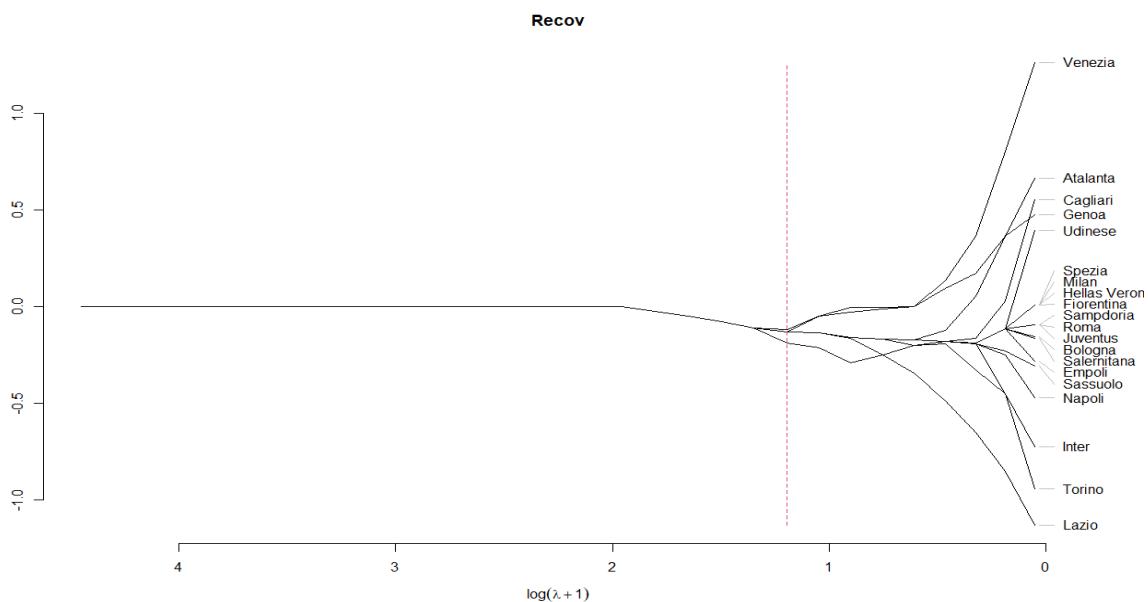


Figura 8.8: Grafico che riporta l'andamento stimato dal modello (4.12) con $Y = 5$ della stima del numero di recuperi per ogni squadra al variare del parametro di tuning λ . La linea rossa tratteggiata indica il parametro di tuning λ ottimo che è stato scelto per ottenere i risultati finali.

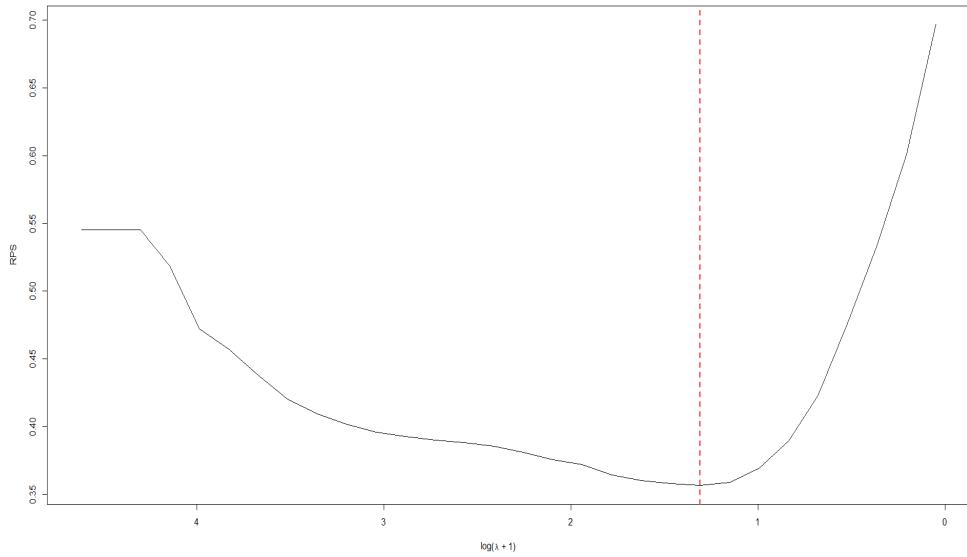


Figura 8.9: Grafico dell'andamento delle prestazioni del modello (4.12) con $Y = 5$, su tutti i trenta valori assunti dal parametro di tuning, indicato con il simbolo λ , durante l'applicazione di K-Fold Cross Validation. L'andamento viene valutato in termini di Ranked Probability Score (RPS). La linea rossa tratteggiata indica il parametro di tuning λ ottimo da utilizzare.

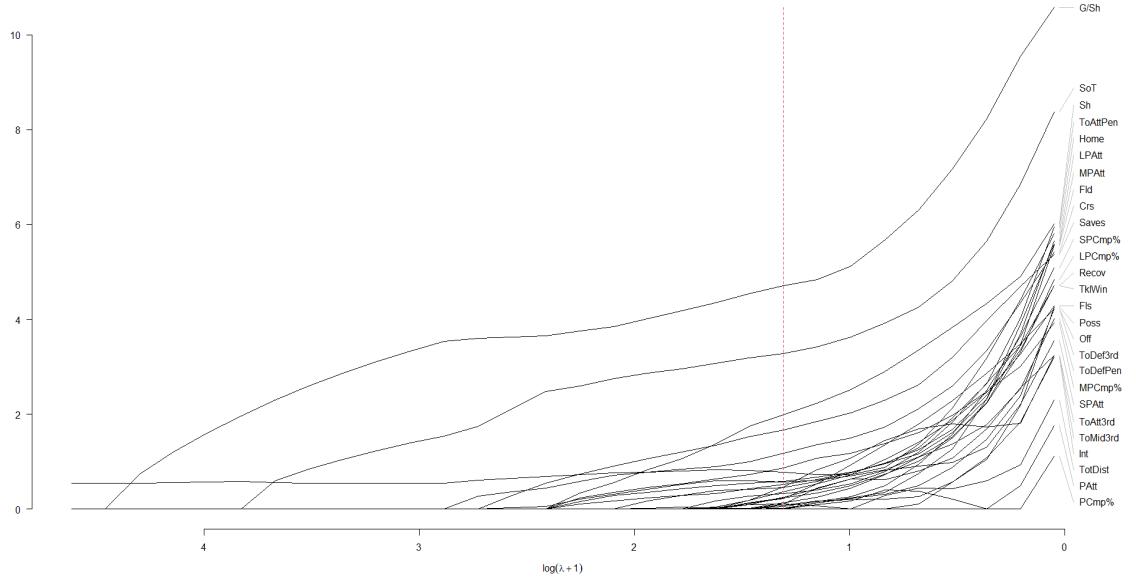


Figura 8.10: Grafico che riporta l'importanza delle covariate rispetto alle norme L2 al variare del parametro di tuning λ

per **Home**. Si segnala un aumento di importanza per le covariate **ToAttPen** e **Recov**, in termini di diminuzione della probabilità di vittoria. Viceversa, si ha un calo d'importanza per **Saves**, **Crs**, **TklWin** e **Off**. Si riconfermano ancora poco significative le variabili riguardanti i passaggi fatta, eccezione per **LPAtt**. Anche **Int** si conferma poco significativo, così come **Poss**. Pertanto, nel complesso, quanto ricavato del modello (4.12) con $Y = 3$ ora trova conferma anche con la variante $Y = 5$.

8.3 Predizioni

Come fatto nel Capitolo 5, in questa sezione si vuole valutare la prestazione del modello (4.12) con $Y = 5$ nella fase di predizione.

Nella Figura 8.11 viene mostrata la classificazione ottenuta perle 80 partite del *test set*. Le predizioni ottenute sono buone, con un'accuratezza registrata pari a 0.675.

		True					
		1	2	3	4	5	Sum
Predicted	1	10	0	0	0	0	10
	2	0	5	0	0	0	5
	3	6	8	23	6	6	49
	4	0	0	0	9	0	9
	5	0	0	0	0	7	7
	Sum	16	13	23	15	13	80

Figura 8.11: Tabella di confusione che indica le predizioni su 80 partite, fatte dal modello (4.12) con $Y = 5$. La classe 1 indica la vittoria della squadra in casa con due gol di scarto, la classe 2 indica la vittoria della squadra in casa con un gol di scarto, la classe 3 indica il pareggio tra le due squadre, la classe 4 indica la vittoria della squadra ospite con un gol di scarto e la classe 5 indica la vittoria della squadra ospite con due gol di scarto. Con **True** si indicano le classificazioni effettive mentre con **Predicted** le predizioni effettuate.

Nella Figura 8.12 vengono mostrate le misurazioni della precisione, della sensibilità e della specificità per ognuna delle cinque categorie registrate sulle predizioni fatte dal modello (4.12) con $Y = 5$. Le misurazioni ottenute sono buone. In molti casi si registra il miglior risultato possibile, ovvero 1. Nello specifico, analizzando la precisione, si nota che il modello non sbaglia nell'etichettare correttamente tutti le osservazioni, eccetto per il pareggio, per il quale il modello registra una precisione pari a 0.47: delle 49 osservazioni classificate con la classe pareggio solo 23 sono effettivamente della classe pareggio. Tuttavia, analizzando la sensibilità si scopre che, nonostante vengono etichettate erroneamente molte osservazioni con la classe pareggio, sono state identificate tutte le osservazioni di classe pareggio. Ovviamente questo va a scapito delle prestazioni della sensibilità sulle altre classi. Infatti, la sensibilità nella classe vittoria della squadra in casa con un gol di scarto è pari a 0.38: solo cinque su otto osservazioni vengono identificate correttamente appartenenti a questa classe. Analogamente anche per le altre classi si ha un calo nella sensibilità. Tuttavia, le misurazioni ottenute sono discrete. Per quanto riguarda la specificità, analogamente alla precisione, per tutte le classi si registra il miglior risultato possibili eccetto per la classe pareggio dove la specificità è pari a 0.54, poiché 26 osservazioni vengono erroneamente classificate con la classe pareggio.

precision	1	2	3	4	5
	1.0000000	1.0000000	0.4693878	1.0000000	1.0000000
recall	1	2	3	4	5
	0.6250000	0.3846154	1.0000000	0.6000000	0.5384615
specificity	1	2	3	4	5
	1.0000000	1.0000000	0.5438596	1.0000000	1.0000000

Figura 8.12: Grafico delle misurazioni riguardanti le predizioni fatte dal modello (4.12) con $Y = 5$ su 80 partite. La classe 1 indica la vittoria della squadra in casa con due gol di scarto, la classe 2 indica la vittoria della squadra in casa con un gol di scarto, la classe 3 indica il pareggio tra le due squadre, la classe 4 indica la vittoria della squadra ospite con un gol di scarto e la classe 5 indica la vittoria della squadra ospite con due gol di scarto. Con **precision** si indicano le misurazioni della precisione per ogni categoria. Con **recall** si indicano le misurazioni della sensibilità per ogni categoria. Con **specificity** si indicano le misurazioni della specificità per ogni categoria.

9 | DISCUSSIONE E COMPARAZIONE DEI RISULTATI

In questo capitolo si discuteranno i risultati ottenuti dai modelli di Bradley-Terry e dalle tecniche di machine learning. Si concluderà con una comparazione tra i risultati delle tecniche di data mining e di machine learning.

9.1 Discussione risultati dei modelli Bradley-Terry

Come illustrato nel Capitolo 5 sono state applicate quattro versioni del modello Bradley-Terry. Con la prima versione, ovvero il modello Bradley-Terry standard (4.9) (**bradley1952rank**) è stato possibile stimare l'abilità di ogni singola squadra e l'effetto di giocare in casa, costruendo così, una base da cui partire per ottenere risultati più esplicativi. Infatti, con l'introduzione di 26 covariate nelle comparazioni ovvero, con il modello (5.1), è stato possibile approfondire in che modo le variabili esplicative sono associate all'esito di una partita. Successivamente per abbassare la complessità del modello e per analizzare che stima viene associata ad ogni covariata per ogni squadra, nel modello (4.12) è stato introdotto e applicato il metodo di regolarizzazione LASSO. Per approfondire gli effetti delle singole variabili esplicative a seconda della squadra in esame è stato applicato il modello (5.2) in cui viene tolto l'effetto dell'intercetta. Si sottolinea inoltre che, come spiegato nel Capitolo 8, per utilizzare le variabili esplicative gol fatti GF e gol subiti GA è stato riapplicato il modello (4.9) ma con una variabile risposta Y con cinque categorie invece che tre. Tale modello a livello di stime ha prodotto risultati simili al modello (4.9) con $Y = 3$, con solo piccole differenze, permettendo così di raffinare l'analisi.

Dai risultati ottenuti è possibile concludere quanto segue. Nel campionato italiano, ai fini della vittoria o, in generale, dell'ottenimento di buoni risultati, è rilevante per una squadra adottare un comportamento tattico, in particolare la costruzione dal basso, e giocare prevalentemente nella propria metà campo. È importante che la squadra adotti un comportamento meno propenso a controllare il pallone per lungo tempo perché sia il possesso della palla Poss sia la distanza percorsa con la palla TotDist non risultano essere vantaggiosi. La squadra deve essere più propensa a giocare la palla nella propria area di difesa per evitare contropiedi perché le stime del numero di tocchi in area di rigore ToDefPen, il numero di tocchi nella trequarti difensiva ToDef3rd e il numero di tocchi a centrocampo ToMid3rd sono associati ad un aumento della probabilità di vittoria. Perciò, avere una buona difesa è fondamentale. La fase offensiva non deve essere troppo lunga in termini di possesso della palla. Infatti, il numero di tocchi fatti nella trequarti offensiva ToAtt3rd porta ad avere una diminuzione delle probabilità di vittoria. Se però, si fanno i giusti passaggi per entrare nell'area di rigore avversaria mantenendo sempre un possesso palla breve si aumentano le probabilità di vittoria. Considerando i casi di Inter e Atalanta, la prima si dimostra essere una delle squadre che più tira in generale (Sh) e in porta (SoT). Alti valori sono presenti anche per l'Atalanta.

Entrambe però mantengono troppo il controllo del pallone nell'area avversaria. Infatti, per entrambe le squadre si riscontrano notevoli diminuzioni della probabilità di vittoria a causa della stima del parametro di $ToAtt3rd$. Peggio ancora per l'Atalanta, che ha un gioco particolarmente offensivo (vedi **ataGioco**), che le fa ottenere una diminuzione della probabilità della vittoria dalla stima del parametro $ToAttPen$, diversamente dalle altre squadre dove la stima è leggermente positiva. Questo perché il prolungato controllo del pallone la porta a esporsi e a subire il contropiede. Si è parlato spesso di contropiedi nella nostra analisi. Quello che emerge sempre in tema di fase offensiva è che il numero di tiri è relativamente basso, fatto dimostrato dal notevole aumento della probabilità di vittoria portato della stima del rapporto gol/tiri G/Sh . Di conseguenza le squadre attaccano poco e, quando lo fanno, cercano di massimizzare la loro fase offensiva. Infatti, le partite nel campionato italiano, spesso finiscono con un massimo di due o tre gol segnati. Pertanto, l'efficacia di un'azione offensiva che porta al gol e la carenza di azioni offensive portano Sh , SoT , ma soprattutto G/Sh , ad assumere un elevato contributo nel determinare la vittoria.

Concludendo la trattazione sulla fase offensiva, si illustra quale sia il miglior modo di attaccare che emerge dai modelli. Si sa che il contropiede è efficace, ma allo stesso tempo difficile da attuare per via del comportamento delle squadre a non sbilanciarsi. Una valida alternativa che emerge è il lancio lungo che parte dall'area compressa tra l'area di rigore della squadra fino a centrocampo ed arriva nell'area avversaria. Infatti, la stima del parametro del numero di passaggi lunghi tentati $LPAtt$ è associata ad una crescita della probabilità di vittoria. L'utilizzo di passaggi filtrati $MPCmp\%$ non è una buona tattica dato che esso risulta associato a una diminuzione della probabilità di vittoria per tutte le squadre. Analogamente, i cross Crs sono associati a una diminuzione della probabilità di vittoria per tutte le squadre. Nello specifico, vengono leggermente più penalizzate Roma, Milan, Napoli mentre vengono particolarmente penalizzate Inter e l'Atalanta, squadra che con il suo gioco sfrutta molto le fasce (vedi **ataGioco**). Il fuorigioco risulta essere associato a una diminuzione della probabilità di vittoria per le squadre con l'abilità maggiore in particolare Milan, Napoli, Inter e Juventus. In conclusione, è importante sottolineare che un atteggiamento troppo speculativo o difensivo da parte della squadra non porta alla vittoria. Questo è il caso del Venezia, classificatosi come ultimo, e che ha ottenuto benefici dalle covariate $ToDefPen$ e $ToDef3rd$, ma non dalle variabili esplicative offensive. Perciò, dall'analisi emerge che per ottenere la vittoria una squadra debba mantenere un comportamento tattico e giocare prevalentemente nella propria metà campo.

Infine, ogni modello ha prodotto delle predizioni dei risultati di alcune partite. Si sono confrontate le prestazioni dei cinque modelli durante la fase di test insieme alle predizioni fatte dai *bookmakers*. Quello che si evince è che tutti i modelli Bradley-Terry hanno prestazioni migliori delle predizioni dei *bookmakers*, segno che le informazioni vengo utilizzate in modo opportuno. Purtroppo, non sono soddisfacenti in termini di predizioni. Infatti, si ottengono delle discrete prestazioni dal punto di vista dell'accuratezza, precisione, sensibilità e della specificità. Il modello Bradley-Terry che ha ottenuto le migliori prestazioni in fase di predizione è stato (4.12), ma con la variabile risposta Y a cinque categorie. Il modello (4.12) è il modello con le covariate specifiche del soggetto e dell'oggetto. Infatti, con questa modifica si va a migliorare le prestazioni del modello (4.12) che già risultava essere il più accurato con $Y = 3$.

9.2 Discussione risultati dei modelli di machine learning

Gli algoritmi di apprendimento automatico scelti sono stati utilizzati per predire l'esito di un insieme di partite, ovvero classificare le partite con una delle seguenti classi: vittoria della squadra in casa, pareggio o vittoria della squadra ospite.

Gli algoritmi utilizzati sono il K-Nearest-Neighbors (K-NN), la Support Vector Machine (SVM), il Decision Tree, la Random Forest e l'AdaBoost. L'algoritmo K-NN è stato scelto perché è semplice da implementare e rapido nell'esecuzione. L'algoritmo SVM è spesso impiegato grazie all'utilizzo del *kernel trick* che permette all'algoritmo di adattarsi bene in vari contesti. Per questa ragione si è deciso di applicare la SVM. L'algoritmo Decision Tree, oltre a essere semplice da implementare e molto usato, è stato scelto anche per individuare quali sono le *features* più decisive per la predizione. Infine, sono stati scelti gli algoritmi Random Forest e AdaBoost perché entrambi sono due tecniche di apprendimento *ensemble* ma con filosofie differenti. Inoltre, il Random Forest permette di individuare quali *features* danno maggior guadagno di informazioni. Grazie alla metrica AUC è possibile confrontare le performance degli algoritmi utilizzati. Nella Figura 9.1 vengono riportate le AUC che sono state misurate durante la fase di predizione di ogni algoritmo utilizzato. Dalle misurazioni ottenute si evince che

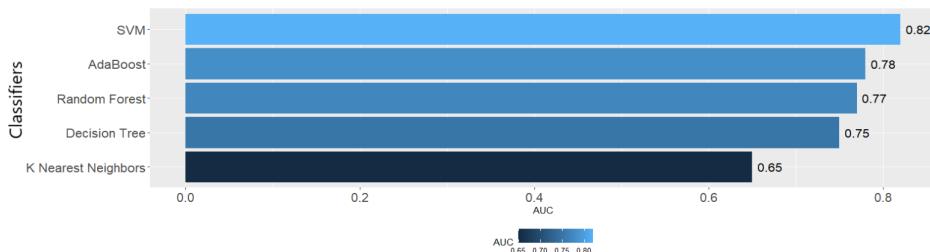


Figura 9.1: Grafico a barre in cui viene riportata la Area Under the Curve (AUC) registrata durante la fase di predizione dei algoritmi K-Nearest-Neighbors (K-NN), Support Vector Machine (SVM), Decision Tree, Random Forest e AdaBoost.

l'AUC più alta è stata registrata nell'algoritmo SVM con un valore pari a 0.82. Infatti, la SVM ha una buona accuratezza perché è l'algoritmo che riesce a identificare con maggior precisione la classe pareggio, la quale è risultata molto ostica da identificare per gli algoritmi trattati. Con una AUC leggermente inferiore, l'algoritmo AdaBoost si classifica come secondo con un valore pari a 0.78. Analogamente alla SVM, l'AdaBoost si distingue dagli algoritmi con prestazioni inferiori grazie alle discrete prestazioni nell'identificazione della classe pareggio. L'algoritmo con la terza AUC più alta è stato il Random Forest con una AUC pari a 0.77 e con prestazioni simili all'algoritmo AdaBoost. In quarta posizione si piazza l'algoritmo Decision Tree con una AUC pari a 0.75. Nonostante le buone prestazioni registrate nell'identificazione delle classi vittoria della squadra in casa e vittoria della squadra ospite, l'algoritmo Decision Tree paga una minor AUC a causa delle brutte prestazioni registrate nell'identificazione della classe pareggio. Infine, l'algoritmo con la più bassa AUC misurata è il K-NN con una AUC pari a 0.65. Infatti, l'algoritmo K-NN ha registrato delle pessime prestazioni nell'identificare le istanze di classe pareggio e delle discrete prestazioni nell'identificare le istanze delle altre due classi.

In conclusione, il problema presentato risulta essere troppo complesso per l'algoritmo K-NN: la strategia di classificare una nuova istanza con la classe di maggioranza dei k-vicini, seppur semplice, non è risultata efficace. Viceversa, la SVM, grazie all'utilizzo della funzione kernel, riesce a gestire la complessità dei dati ottenendo delle buone prestazioni in fase di predizione.

9.3 Confronto

Come riportato dai Capitoli 5, 7 e 8, le prestazioni migliori in fase di predizioni si ottengono con i metodi di *machine learning*. I modelli di *data mining* sono utili per interpretare le relazioni tra le variabili, meno in termini di predizioni. Ciononostante, grazie all'interpretabilità dei modelli è possibile individuare quali variabili siano associate all'esito della partita e in che misura. Viceversa, i modelli di *machine learning* in genere, si focalizzano su caratteristiche troppo complesse e difficili da interpretare. Grazie agli algoritmi Decision Tree e Random Forest è comunque possibile fare un confronto con i modelli Bradley-Terry riguardo a quali variabili siano associate all'esito della partita. Notiamo che le variabili legate ai tiri sono le più importanti per entrambi i modelli. Nei due modelli di *machine learning* viene data molta importanza sia ai passaggi lunghi tentati, come per i modelli BT, sia alla percentuale di passaggi lunghi completati in contrasto con i modelli BT. Sia i modelli BT e sia i modelli Decision Tree e Random Forest, la *feature* della percentuale dei passaggi medi completati ricopre un ruolo importante nell'esito della partita. Tuttavia, dai modelli BT sappiamo che essa è associata negativamente sulla probabilità di vittoria. Nonostante tante similitudini tra i risultati delle due tipologie di metodi, ci sono alcune differenze. Si hanno differenze di interpretazioni sull'importanza del numero di parate. Per il modello Bradley-Terry la variabile si associa fortemente all'esito della partita, mentre per i due modelli di *machine learning* no. Viceversa, la percentuale di passaggi completati è molto importante per i modelli di *machine learning*, mentre non lo è per i modelli BT. Anche per la distanza percorsa con la palla viene assegnato un rilevante guadagno di informazione, solo nei modelli di *machine learning*. Per entrambi i metodi viene ribadita la poca importanza riguardo al possesso della palla e delle variabili riguardanti il numero di contrasti vinti e il numero di intercetti. Anche i numeri di falli subiti e fatti non hanno una grande importanza per entrambi i metodi. Inoltre, per entrambi i metodi il numero di falli fatti è leggermente più rilevante rispetto al numero di falli subiti. Dal punto di vista dei tempi d'esecuzione si segnala una maggior velocità di esecuzione dei modelli Bradley-Terry rispetto ai modelli di *machine learning*. In particolare, l'algoritmo Random Forest impiega il maggior tempo d'esecuzione fra tutti gli algoritmi.

10 | CONCLUSIONI

In questa tesi, partendo dalla domanda posta inizialmente "Cosa influenza il successo o il fallimento delle singole squadre durante una partita di calcio?" si sono analizzate le possibili relazioni tra l'esito di una partita di calcio e le principali statistiche raccolte durante la partita. L'utilizzo di un modello di confronto a coppie, ovvero il modello Bradley-Terry è un naturale strumento di analisi che è stato applicato sui dati che si riferiscono agli incontri della Serie A italiana 2021/2022.

I risultati ricavati dai cinque modelli di Bradley-Terry e dai algoritmi Decision Tree e Random Forest, mettono in luce l'importanza dei tiri, l'effetto di giocare in casa, il mantenimento del controllo della palla nell'area della squadra in possesso della palla e dell'utilizzo di lanci lunghi al fine di una vittoria. Viceversa, viene individuato che i cross, i passaggi filtranti e un alto numero di tocchi del pallone nell'area dell'avversario sono associati negativamente all'esito della partita. Per le squadre con la maggior abilità stimata subire un fuorigioco incide negativamente sul loro successo sportivo. Infine, viene ricavato che il possesso della palla non viene associato all'esito della partita.

Chiaramente quanto ricavato vale per la stagione 2021/2022 del campionato italiano di Serie A. È naturale quindi che l'analisi possa essere estesa su un'altra stagione della Serie A oppure ad gli altri campionati europei quali, la Premier League, la Liga spagnola, la Ligue 1 e la Bundesliga. Occorre fare attenzione al fatto che sia nella Ligue 1 e nella Bundesliga negli ultimi anni Paris Saint Germain e Bayern München hanno monopolizzato la vittoria del campionato. Quindi c'è da aspettarsi ampi dislivelli di abilità tra queste squadre e le altre dei rispettivi campionati. Per quanto riguarda la Premier League è possibile aspettarsi una stima delle abilità più bilanciata, ma soprattutto una stima fortemente positiva riguardo al possesso della palla grazie allo stile di gioco proposto dall'allenatore del Manchester City, Pep Guardiola (vedi **futbol**), vincitore di quattro delle ultime cinque edizioni del campionato inglese.

Sicuramente d'interesse potrebbe essere l'applicazione di un modello Bradley-Terry dinamico (**cattelan2013dynamic**), ovvero che vada a valutare la variazione dell'abilità di ogni singola squadra durante la stagione sportiva, in modo da individuare possibili fenomeni che possano ripercuotersi positivamente o negativamente sulle prestazioni delle squadre.

Naturalmente, oltre a valutare l'abilità di una squadra durante la stagione un'ulteriore estensione dell'analisi è considerare più di una stagione (**tsokos2019modeling**), sebbene ci sia un atteso aumento di complessità computazionale.

Occorre sottolineare che nelle analisi svolte, le covariate vengono incorporate in modo lineare nelle abilità delle squadre nelle specifiche partite. L'aggiunta di effetti non lineari attraverso l'utilizzo di metodi di *machine learning* non supervisionati come ad esempio, il Clustering (**dunn1974well**), può essere una possibile estensione del lavoro. Infatti come riportato nel lavoro svolto da **shin2014novel**, il Clustering è utilizzato per individuare caratteristiche non lineari comuni tra squadre di calcio come ad esempio, le strategie di gioco. Infatti, gli algoritmi di apprendimento non supervisionato hanno l'obiettivo di raggruppare e interpretare i dati basandosi solo sull'input ricevuto, attra-

verso l'individuazione di *features* non lineari. Questi algoritmi permettono di migliorare l'analisi ma con lo svantaggio di un maggior costo computazionale da sostenere. Certamente anche l'aggiunta di altre covariate come, ad esempio, la distanza percorsa dai giocatori o il numero di calci d'angolo battuti, potrebbe permettere di individuare nuove statistiche chiave che determinano l'esito della partita.

11 | CODICE IN R

11.1 extractRowsAway

Codice della funzione per la raccolta delle partite giocate fuori casa dalla squadra indicata nella variabile Team.

```
1 extractRowsAway <-function () {
2   k <- 1
3   z <- 1
4   for(i in 1:nrow(AdjserieA)){
5     if(AdjserieA$AtHome[i] == TRUE){
6       for(j in 1:nrow(AdjserieA)){
7         if((AdjserieA$Team[j] == AdjserieA$Vs[i]) && (AdjserieA$Team[i]
8 == AdjserieA$Vs[j]) && (AdjserieA$AtHome[j] == FALSE)){
9           PossVs[k] <-> AdjserieA$Poss[j]
10          ShVs[k] <-> AdjserieA$Sh[j]
11          ShTVs[k] <-> AdjserieA$SoT[j]
12          G.ShVs[k] <-> AdjserieA$'G/Sh'[j]
13          SavesVs[k] <-> AdjserieA$Saves[j]
14          PAttVs[k] <-> AdjserieA$PAtt[j]
15          PCmp.Vs[k] <-> AdjserieA$'PCmp%'[j]
16          SPAttVs[k] <-> AdjserieA$SPAtt[j]
17          SPCmp.Vs[k] <-> AdjserieA$'SPCmp%'[j]
18          MPAttVs[k] <-> AdjserieA$MPAtt[j]
19          MPCmp.Vs[k] <-> AdjserieA$'MPCmp%'[j]
20          LPAttVs[k] <-> AdjserieA$LPAtt[j]
21          LPCmp.Vs[k] <-> AdjserieA$'LPCmp%'[j]
22          ToDefPenVs[k] <-> AdjserieA$ToDefPen[j]
23          ToDef3rdVs[k] <-> AdjserieA$ToDef3rd[j]
24          ToMid3rdVs[k] <-> AdjserieA$ToMid3rd[j]
25          ToAtt3rdVs[k] <-> AdjserieA$ToAtt3rd[j]
26          ToAttPenVs[k] <-> AdjserieA$ToAttPen[j]
27          ToDistVs[k] <-> AdjserieA$TotDist[j]
28          FlsVs[k] <-> AdjserieA$Fls[j]
29          FldVs[k] <-> AdjserieA$Fld[j]
30          OffVs[k] <-> AdjserieA$Off[j]
31          CrsVs[k] <-> AdjserieA$Crs[j]
32          IntVs[k] <-> AdjserieA$Int[j]
33          TklWinVs[k] <-> AdjserieA$TklWin[j]
34          RecovVs[k] <-> AdjserieA$Recov[j]
35          GFVs[k] <-> AdjserieA$GF[j]
36          GAVs[k] <-> AdjserieA$GA[j]
37          k <- k + 1
38        }
39      }
40    } else{
41      del[z] <- i
42      z <- z + 1
43    }
44 }
```

Listing 11.1: Codice di adattamento dataset per il trasferimento dati.

11.2 createYFull

Codice della funzione per la creazione della variabile Y3.

```

1  createYFull <- function(SerieA){
2    for(i in 1:38){
3      for(y in 1:10){
4        for(z in 1:nrow(SerieA)){
5          if((paste(str1, i) == SerieA$Round[z]) && (SerieA$AtHome[z] ==
6            TRUE ) && (first.time[z] == TRUE)){
7            if(SerieA$Res[z] == -1)
8            {
9              response[w] <- 3
10            }
11            else{
12              if(SerieA$Res[z] == 0){
13                response[w] <- 2
14              }
15              else{
16                response[w] <- SerieA$Res[z]
17              }
18            }
19            switch(
20              SerieA$Team[z],
21              "Atalanta" = first.object[w] <- 1,
22              "Bologna" = first.object[w] <- 2,
23              "Cagliari" = first.object[w] <- 3,
24              "Empoli" = first.object[w] <- 4,
25              "Fiorentina" = first.object[w] <- 5,
26              "Genoa" = first.object[w] <- 6,
27              "Hellas Verona" = first.object[w] <- 7,
28              "Inter" = first.object[w] <- 8,
29              "Juventus" = first.object[w] <- 9,
30              "Lazio" = first.object[w] <- 10,
31              "Milan" = first.object[w] <- 11,
32              "Napoli" = first.object[w] <- 12,
33              "Roma" = first.object[w] <- 13,
34              "Salernitana" = first.object[w] <- 14,
35              "Sampdoria" = first.object[w] <- 15,
36              "Sassuolo" = first.object[w] <- 16,
37              "Spezia" = first.object[w] <- 17,
38              "Torino" = first.object[w] <- 18,
39              "Udinese" = first.object[w] <- 19,
40              "Venezia" = first.object[w] <- 20,
41            )
42            switch(
43              SerieA$Vs[z],
44              "Atalanta" = second.object[w] <- 1,
45              "Bologna" = second.object[w] <- 2,
46              "Cagliari" = second.object[w] <- 3,
47              "Empoli" = second.object[w] <- 4,
48              "Fiorentina" = second.object[w] <- 5,
49              "Genoa" = second.object[w] <- 6,
50              "Hellas Verona" = second.object[w] <- 7,
51              "Inter" = second.object[w] <- 8,
52              "Juventus" = second.object[w] <- 9,
53              "Lazio" = second.object[w] <- 10,
54              "Milan" = second.object[w] <- 11,
55              "Napoli" = second.object[w] <- 12,
56              "Roma" = second.object[w] <- 13,
57              "Salernitana" = second.object[w] <- 14,
58              "Sampdoria" = second.object[w] <- 15,
```

```

59         "Sassuolo" = second.object[w] <- 16,
60         "Spezia" = second.object[w] <- 17,
61         "Torino" = second.object[w] <- 18,
62         "Udinese" = second.object[w] <- 19,
63         "Venezia" = second.object[w] <- 20,
64     )
65
66     subject[w] <- paste(str1, i)
67     with.order[w] <- TRUE
68     first.time[z] <- FALSE
69     w <- w + 1
70     break
71   }
72 }
73 }
74 }
75 }
```

Listing 11.2: Codice per la creazione della variabile Y3.

11.3 extractData

Codice della funzione per estrarre le informazioni di una variabile.

```

1 extractData <-function(cov){
2   for(i in 1:nrow(SerieA)){
3     switch(
4       SerieA$Team[i],
5       "Atalanta" = {ata[it[1]] <- SerieA[[cov]][i]
6         it[1] <- it[1] + 1
7       },
8       "Bologna" = {bol[it[2]] <- SerieA[[cov]][i]
9         it[2] <- it[2] + 1
10      },
11       "Cagliari" = {cag[it[3]] <- SerieA[[cov]][i]
12         it[3] <- it[3] + 1
13      },
14       "Empoli" = {emp[it[4]] <- SerieA[[cov]][i]
15         it[4] <- it[4] + 1
16      },
17       "Fiorentina" = {fio[it[5]] <- SerieA[[cov]][i]
18         it[5] <- it[5] + 1
19      },
20       "Genoa" = {gen[it[6]] <- SerieA[[cov]][i]
21         it[6] <- it[6] + 1
22      },
23       "Hellas Verona" = {ver[it[7]] <- SerieA[[cov]][i]
24         it[7] <- it[7] + 1
25      },
26       "Inter" = {int[it[8]] <- SerieA[[cov]][i]
27         it[8] <- it[8] + 1
28      },
29       "Juventus" = {juv[it[9]] <- SerieA[[cov]][i]
30         it[9] <- it[9] + 1
31      },
32       "Lazio" = {laz[it[10]] <- SerieA[[cov]][i]
33         it[10] <- it[10] + 1
34      },
35       "Milan" = {mil[it[11]] <- SerieA[[cov]][i]
36         it[11] <- it[11] + 1
37     },
```

```

38     "Napoli" = {nap[it[12]] <- SerieA[[cov]][i]
39         it[12] <- it[12] + 1
40     },
41     "Roma" = {rom[it[13]] <- SerieA[[cov]][i]
42         it[13] <- it[13] + 1
43     },
44     "Salernitana" = {sal[it[14]] <- SerieA[[cov]][i]
45         it[14] <- it[14] + 1
46     },
47     "Sampdoria" = {sam[it[15]] <- SerieA[[cov]][i]
48         it[15] <- it[15] + 1
49     },
50     "Sassuolo" = {sas[it[16]] <- SerieA[[cov]][i]
51         it[16] <- it[16] + 1
52     },
53     "Spezia" = {spe[it[17]] <- SerieA[[cov]][i]
54         it[17] <- it[17] + 1
55     },
56     "Torino" = {tor[it[18]] <- SerieA[[cov]][i]
57         it[18] <- it[18] + 1
58     },
59     "Udinese" = {udi[it[19]] <- SerieA[[cov]][i]
60         it[19] <- it[19] + 1
61     },
62     "Venezia" = {ven[it[20]] <- SerieA[[cov]][i]
63         it[20] <- it[20] + 1
64     },
65   )
66 }
67 }
```

Listing 11.3: Codice per estrarre le informazioni di una variabile.

11.4 extractAll

Codice della funzione per estrarre le informazioni di tutte le variabili.

```

1 extractAll <- function(covs, row, teams, n = 20, m = 38){
2   clear()
3   num <- 20
4   isFirst = TRUE
5   tab <-c()
6   for(z in 1:length(covs)){
7     extractData(covs[z])
8     if(isFirst){
9       tab <- matrix(cbind(ata,bol,cag,emp,fio,gen,ver,int,juv,laz,mil,
10                   nap,rom,sal,sam,sas,spe,tor,udi,ven), nrow = m, ncol = num)
11     isFirst = FALSE
12   }
13   else{
14     tab <- matrix(cbind(tab, ata,bol,cag,emp,fio,gen,ver,int,juv,laz,
15                   mil,nap,rom,sal,sam,sas,spe,tor,udi,ven), nrow = m, ncol = num)
16   }
17   refresh()
18   num <- num + n
19 }
20 col <- colLabel(teams, covs)
21 num2 <- num - n
22 dimnames(tab) <- list(row[1:m], col[1:num2])
23 return(tab)
```

```
22 }
```

Listing 11.4: Codice per estrarre le informazioni di tutte le variabili.

11.5 colLabel

Codice della funzione per la creazione delle etichette delle colonne.

```
1 colLabel <- function(teams, cov){
2   nameCov <- c()
3   z <- 1
4   for(i in 1:length(cov)){
5     for(y in 1:length(teams)){
6       nameCov[z] <- paste(cov[i], teams[y], sep="..")
7       z <- z + 1
8     }
9   }
10  return(nameCov)
11 }
```

Listing 11.5: Codice per la creazione delle etichette delle colonne.

11.6 rowLabel

Codice della funzione per la creazione delle etichette delle righe.

```
1 rowLabel <- function(str, n){
2   nameRows <- c()
3   for(i in 1:n){
4     nameRows[i] <- paste(str, i)
5   }
6   return(nameRows)
7 }
```

Listing 11.6: Codice per la creazione delle etichette delle righe.

11.7 Librerie

- * **BradleyTerry2 (bt2)** è una libreria per il linguaggio R che implementa il modello di Bradley-Terry per la classificazione del risultato delle competizioni e la stima dei parametri del modello. Inoltre fornisce metriche per la valutazione delle stime calcolate.
- * **BTLLasso (btl)** è una libreria per il linguaggio R che implementa diverse varianti del modello Bradley-Terry. Inoltre, fornisce il metodo di regolarizzazione LASSO per raggruppare determinati effetti e per ridurre la complessità dei modelli.
- * **EffectStars2 (EffectStars2)** è una libreria per il linguaggio R che implementa il grafico a stella per la visualizzazione delle stime dei parametri corrispondenti a diversi gruppi, ad esempio nei modelli logit multinomiali.
- * **ggplot2 (ggplot2)** è una libreria per il linguaggio R che implementa grafici e figure per la visualizzazione dei dati.
- * **mltest (mltest)** è una libreria per il linguaggio R che fornisce metodi per il calcolo delle metriche di valutazione per classificazione multi classe.

12 | CODICE IN PYTHON

12.1 createTable

Codice per la creazione del dataset utilizzabile con i metodi di machine learning.

```
1  def createTable(df, dfM):
2      for i in range(0, df.shape[0], 1):
3          if df.AtHome[i] == True:
4              for j in range(0, df.shape[0], 1):
5                  if df.Team[j] == df.Vs[i] and df.Team[i] == df.Vs[j] and df.
6                      AtHome[j] == False:
7                          dfM2 = pd.DataFrame({'Date': [df.Date[i]],
8                                              'Round': [df.Round[i]],
9                                              'AtHome': [df.AtHome[i]],
10                                             'Result': [df.Res[i]],
11                                             'G_Home': [df.GF[i]],
12                                             'G_Away': [df.GA[i]],
13                                             'Home_Team': [df.Team[i]],
14                                             'Away_Team': [df.Vs[i]],
15                                             'Home_Poss': [df.Poss[i]],
16                                             'Home_Sh': [df.Sh[i]],
17                                             'Home_SoT': [df.SoT[i]],
18                                             'Home_G/Sh': [df['G/Sh'][i]],
19                                             'Home_Saves': [df.Saves[i]],
20                                             'Home_PAtt': [df.PAtt[i]],
21                                             'Home_PCmp%': [df['PCmp%'][i]],
22                                             'Home_SPAtt': [df.SPAtt[i]],
23                                             'Home_SPCmp%': [df['SPCmp%'][i]],
24                                             'Home_MPAtt': [df.MPAtt[i]],
25                                             'Home_MPCmp%': [df['MPCmp%'][i]],
26                                             'Home_LPAtt': [df.LPAtt[i]],
27                                             'Home_LPCmp%': [df['LPCmp%'][i]],
28                                             'Home_ToDefPen': [df.ToDefPen[i]],
29                                             'Home_ToDef3rd': [df.ToDef3rd[i]],
30                                             'Home_ToMid3rd': [df.ToMid3rd[i]],
31                                             'Home_ToAtt3rd': [df.ToAtt3rd[i]],
32                                             'Home_ToAttPen': [df.ToAttPen[i]],
33                                             'Home_TotDist': [df.TotDist[i]],
34                                             'Home_Fls': [df.Fls[j]],
35                                             'Home_Fld': [df.Fld[i]],
36                                             'Home_Off': [df.Off[i]],
37                                             'Home_Crs': [df.Crs[j]],
38                                             'Home_Int': [df.Int[j]],
39                                             'Home_TklWin': [df.TklWin[i]],
40                                             'Home_Recov': [df.Recov[i]],
41                                             'Away_Poss': [df.Poss[j]],
42                                             'Away_Sh': [df.Sh[j]],
43                                             'Away_SoT': [df.SoT[j]],
44                                             'Away_G/Sh': [df['G/Sh'][j]],
45                                             'Away_Saves': [df.Saves[j]],
46                                             'Away_PAtt': [df.PAtt[j]],
47                                             'Away_PCmp%': [df['PCmp%'][j]],
48                                             'Away_SPAtt': [df.SPAtt[j]],
49                                             'Away_SPCmp%': [df['SPCmp%'][j]],
```

```

49     'Away_MPAtt': [df.MPAtt[j]],
50     'Away_MPCmp%': [df['MPCmp%'][j]],
51     'Away_LPAtt': [df.LPAtt[j]],
52     'Away_LPCmp%': [df['LPCmp%'][j]],
53     'Away_ToDefPen': [df.ToDefPen[j]],
54     'Away_ToDef3rd': [df.ToDef3rd[j]],
55     'Away_ToMid3rd': [df.ToMid3rd[j]],
56     'Away_ToAtt3rd': [df.ToAtt3rd[j]],
57     'Away_ToAttPen': [df.ToAttPen[j]],
58     'Away_TotDist': [df.TotDist[j]],
59     'Away_Fls': [df.Fls[j]],
60     'Away_Fld': [df.Fld[j]],
61     'Away_Off': [df.Off[j]],
62     'Away_Crs': [df.Crs[j]],
63     'Away_Int': [df.Int[j]],
64     'Away_TklWin': [df.TklWin[j]],
65     'Away_Recov': [df.Recov[j]]})
66 dfM = dfM.append(dfM2, ignore_index=True)
67 return dfM
68

```

Listing 12.1: Codice per la creazione del dataset utilizzabile con i metodi di machine learning.

12.2 Librerie

- * **Sklearn (sklearn)** è una libreria *open source* di machine learning per il linguaggio di programmazione Python. La libreria Sklearn fornisce una serie di algoritmi di machine learning per l'analisi dei dati, ovvero modelli di classificazione, regressione, clustering ma anche algoritmi di model selection, di dimensionality reduction e funzionalità di preprocessing dei dati. È progettato per lavorare con le librerie NumPy, pandas e matplotlib.
- * **Matplotlib (matplotlib)** è una libreria *open source* per la visualizzazione dei dati per il linguaggio di programmazione Python. La libreria Matplotlib fornisce una vasta gamma di opzioni per la creazione di grafici e figure. È particolarmente utile per la visualizzazione di dati multidimensionali e per la creazione di grafici di grandi dimensioni e complessi.
- * **NumPy (numPy)** è una libreria *open source* per il linguaggio di programmazione Python. La libreria NumPy fornisce metodi per l'elaborazione di vettori multidimensionali e matrici. Offre una serie di funzionalità avanzate per l'elaborazione di dati, come ad esempio il calcolo di statistiche, l'algebra lineare e la generazione di numeri casuali.
- * **Pandas (pandas)** è una libreria *open source* per il linguaggio di programmazione Python. La libreria Pandas fornisce funzionalità di manipolazione e analisi di dati di diverse forme e fonti.

BIBLIOGRAFIA

SITOGRAFIA
