

Progetto Machine Learning – Previsioni partite Serie A

Matteo Maraziti 534932

Federico Tocci 533449

Giacomo Scordino 533393

INTRODUZIONE

Il presente progetto si pone l'obiettivo di applicare le tecniche di machine learning per cercare di prevedere gli esiti delle partite di Serie A italiana. L'idea alla base di questo progetto è quella di utilizzare le statistiche delle squadre, raccolte in anni precedenti, per sviluppare un modello predittivo in grado di fornire stime attendibili riguardo agli esiti delle partite nella stagione attuale.

Nel corso di questa relazione, esamineremo in dettaglio i passaggi chiave del progetto, dalla raccolta e preparazione dei dati all'addestramento del modello, con un focus particolare sull'approccio metodologico adottato e sui risultati ottenuti.

RACCOLTA DATI

I dati relativi alle statistiche delle squadre sono stati estratti dalla fonte web fbref.com, un'autorevole risorsa che offre una vasta gamma di informazioni sul calcio e le prestazioni delle squadre. Questi dati coprono un periodo di tempo significativo, comprendendo le ultime quattro stagioni [Serie A 19-20, SerieA 20-21, SerieA 21-22, Serie A 22-23], fornendo così una panoramica completa delle performance delle squadre nel calcio durante un periodo di tempo significativo. Il dataset complessivo è quindi composto da un totale di 1520 righe di dati. Ogni riga contiene 23 colonne di informazioni relative alle squadre, alle loro statistiche e ai risultati delle partite. Di seguito è riportato il contenuto delle colonne:

1. **squadraa**: Il nome della squadra A.
2. **squadraa_xg**: Rappresenta gli Expected Goals (xG) per la squadra A. Gli xG sono una misura statistica che stima la probabilità di un tiro di una squadra di diventare un gol. Sono utilizzati per valutare quanto una squadra è stata efficace nel creare opportunità da gol.
3. **squadraa_xag**: Rappresenta gli Expected Goals Against (xG Against) per la squadra A. Questo valore indica quanto è probabile che la squadra A conceda un gol in base alle opportunità concesse agli avversari.
4. **squadraa_xgxag**: È la differenza tra gli xG e gli xG Against per la squadra A. Questo può essere utilizzato per valutare il bilancio complessivo delle opportunità da gol create e concesse dalla squadra A.
5. **squadraa_npzgxg**: Rappresenta gli Expected Goals non penalizzati (non-penalty xG) per la squadra A. Questi xG escludono i calci di rigore e sono utilizzati per valutare le opportunità da gol create dalla squadra escludendo i rigori.

6. **squadraa_npxgxag**: È la differenza tra gli xG non penalizzati e gli xG Against per la squadra A, escludendo i calci di rigore.
7. **avversarioa_xg**: Questa variabile rappresenta gli Expected Goals (xG) per la squadra avversaria (squadra B) nella partita contro la squadra A. Indica la probabilità stimata di segnare gol per la squadra avversaria sulla base delle opportunità create durante la partita.
8. **avversarioa_xag**: Gli Expected Goals Against (xG Against) per la squadra avversaria (squadra B) rappresentano la probabilità stimata di subire gol dalla squadra A in base alle opportunità create dalla squadra A durante la partita.
9. **avversarioa_xgxag**: Questo valore rappresenta la differenza tra gli Expected Goals (xG) della squadra avversaria e gli Expected Goals Against (xG Against) della squadra avversaria nella partita contro la squadra A. Può indicare quale delle due squadre ha avuto una migliore performance nel creare e difendere opportunità da gol.
10. **avversarioa_npzgx**: Questa variabile rappresenta gli Expected Goals per la squadra avversaria (squadra B) nella partita contro la squadra A. Esclude i calci di rigore e valuta le opportunità da gol create dalla squadra avversaria escludendo i rigori.
11. **avversarioa_npxgxag**: Questo valore rappresenta la differenza tra gli Expected Goals della squadra avversaria e gli Expected Goals Against della squadra avversaria nella partita contro la squadra A, escludendo i calci di rigore.

Le successive colonne seguono lo stesso schema, ma si riferiscono alla squadra B (squadrab) e al suo avversario (avversariob) anziché alla squadra A.

squadraa	squadraa_xg	squadraa_xag	squadraa_npxgxag	squadraa_npzgx	squadraa_npxgxag	avversarioa_xg	avversarioa_xag	avversarioa_xgxag	avversarioa_npzgx	avversarioa_npxgxag
Juventus	1,87	1,2	3,07	1,58	2,78	1,18	0,72	1,9	0,94	1,66

Figura 1: record di esempio delle statistiche della squadra a

squadrab	squadrab_xg	squadrab_xag	squadrab_npxgxag	squadrab_npzgx	squadrab_npxgxag	avversariob_xg	avversariob_xag	avversariob_xgxag	avversariob_npzgx
Inter	1,83	1,16	2,98	1,6	2,76	1,07	0,74	1,81	0,99

Figura 2 record di esempio delle statistiche della squadra b

informazioni squadra a	informazioni squadra b	risultato
------------------------	------------------------	-----------

Figura 3 struttura delle ennuple del dataset

SCELTE IMPLEMENTATIVE

L'obiettivo principale dell'analisi dei dati all'interno del dataset era assegnare a ciascuna partita una delle tre classi a disposizione: Vittoria, Pareggio o Sconfitta. Nella classe "Vittoria", viene identificato il caso in cui la squadra A ha vinto la partita contro la squadra B, nella classe "Pareggio" il caso in cui le due squadre hanno pareggiato e nella classe "Sconfitta" il caso in cui la squadra A ha perso la partita contro la squadra B. Con tale

premesse, sono state implementate diverse soluzioni basate su modelli di Machine Learning e nei prossimi paragrafi, vengono illustrati lo pseudocodice e le prestazioni di ciascuno di questi modelli.

MODELLO 1 – SVM no cross validation

Per il primo modello, è stata scelta l'implementazione di un Support Vector Machine (SVM) utilizzando la strategia "One Against One". Di seguito è spiegata la logica del codice python prodotto.

1. **Funzione evaluate_model:** Questa funzione è responsabile di creare un modello SVM con parametri specifici (C e kernel), addestrarlo sui dati di addestramento e valutarlo sui dati di convalida. L'accuratezza del modello sui dati di convalida viene calcolata e restituita come risultato.
2. **Inizializzazione delle variabili:** Vengono inizializzate alcune variabili, tra cui best_C (il miglior valore di C finora), best_kernel (il miglior tipo di kernel finora) e best_accuracy (l'accuratezza migliore finora).
3. **Ciclo di ottimizzazione:** il processo di ottimizzazione dei parametri avviene in modo iterativo. Per ogni iterazione, vengono generati nuovi valori casuali per i parametri C e kernel. Il modello SVM viene quindi addestrato con questi nuovi parametri utilizzando la funzione evaluate_model, e l'accuratezza sul set di dati di convalida viene calcolata. Se l'accuratezza ottenuta con i nuovi parametri è migliore dell'accuratezza migliore finora (best_accuracy), i nuovi parametri vengono considerati come i migliori.
4. **Modello finale:** Una volta completate tutte le iterazioni, i parametri best_C e best_kernel corrispondenti all'accuratezza migliore sono utilizzati per creare un modello SVM finale. Questo modello viene addestrato sui dati di addestramento e convalidato sui dati di validazione.
5. **Valutazione del modello finale:** Il modello SVM finale viene valutato sui dati di test per calcolare l'accuratezza sul set di dati di test.
6. **Stampa dei risultati:** Alla fine, vengono stampati i parametri del modello ottimizzato (best_C e best_kernel) insieme all'accuratezza sul set di dati di test.

MODELLO 2 – SVM con cross validation

Il secondo modello implementa un modello di Support Vector Machine (SVM) utilizzando la strategia "One Against One" e la tecnica di cross-validation. Di seguito è spiegata la logica del codice python prodotto.

1. **Funzione evaluate_model:** Questa funzione crea un modello SVM con i parametri dati, esegue il cross-validation; quindi, calcola e restituisce l'accuratezza media dei punteggi ottenuti durante la convalida incrociata.
2. **Inizializzazione delle variabili:** Vengono inizializzate alcune variabili, tra cui best_C e best_kernel, l'accuratezza migliore best_accuracy, e il numero massimo di iterazioni max_iterations.

3. **Ciclo di ottimizzazione:** il processo di ottimizzazione dei parametri avviene in modo iterativo. Per ogni iterazione nel range `max_iterations`, vengono generate casualmente due nuove variabili: `new_` e `new_kernel`. Questi parametri casuali vengono quindi utilizzati per addestrare un modello SVM e valutarlo utilizzando la funzione `evaluate_model`. Se l'accuratezza ottenuta con questi parametri casuali è maggiore dell'accuratezza migliore finora registrata (`best_accuracy`), i nuovi parametri diventano i migliori finora trovati.
4. **Modello finale:** Una volta completate tutte le iterazioni, i parametri `best_C` e `best_kernel` corrispondenti all'accuratezza migliore sono utilizzati per creare un modello SVM finale. Questo modello viene addestrato sui dati di addestramento e convalidato sui dati di validazione.
5. **Valutazione del modello finale:** Il modello SVM finale viene valutato sui dati di test per calcolare l'accuratezza sul set di dati di test.
6. **Stampa dei risultati:** Alla fine, vengono stampati i parametri del modello ottimizzato (`best_C` e `best_kernel`) insieme all'accuratezza sul set di dati di test.

MODELLO 3 – ALBERO DECISIONALE `random forest`

Per il terzo modello, è stata scelta l'implementazione di un albero decisionale `random forest`. Di seguito è spiegata la logica del codice python prodotto.

1. **Funzione `evaluate_model`:** viene definita una funzione per valutare modelli Random Forest con diversi set di iperparametri.
2. **Inizializzazione delle variabili:** vengono inizializzati alcuni valori di partenza, tra cui il numero massimo di iterazioni e i migliori iperparametri trovati fino a quel momento.
3. **Ciclo di ottimizzazione:** Inizia un ciclo che cerca di migliorare i risultati del modello:
 - Genera casualmente nuovi iperparametri.
 - Valuta un modello con questi nuovi iperparametri utilizzando dati di validazione.
 - Se il modello è più accurato dei precedenti, aggiorna i migliori iperparametri trovati.
4. **Modello finale:** crea un modello finale con i migliori iperparametri e lo addestra con i dati di addestramento.
5. **Valutazione del modello finale:** Utilizza il modello finale per fare previsioni sui dati di test e calcola l'accuratezza sui dati di test.
6. **Stampa dei risultati:** Stampa a schermo i migliori iperparametri trovati e l'accuratezza sui dati di test.

CONFRONTO DELLE PRESTAZIONI

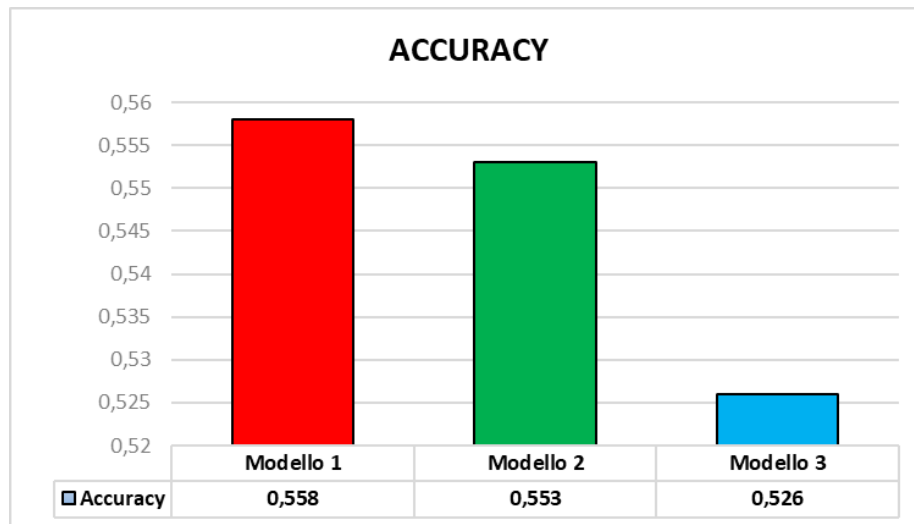


Figura 4 confronto delle accuratie dei modelli utilizzati

Modello 1

Migliori iperparametri: $C = 7.15$, Kernel = rbf

Modello 2

Migliori iperparametri: $C = 1.16$, Kernel = linear

Modello 3

Migliori iperparametri: $n_estimators = 52$, $max_depth = 80$