# PROGETTO DI INTELLIGENZA ARTIFICIALE


Professori

**Endrit Xhina**

**Kosta Anxhela**

**Lili Ilma**

Studenti

**Longarini Lorenzo**

**Ramovini Loris**

INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE | AA 2022 / 2023

# Indice

# Introduzione

ll progetto consiste nell'utilizzare i risultati delle partite di calcio della Sserie A per poter prevedere alcuni risultati utili in un incontro tra due squadre.

Il knowledge base viene implementato attraverso i dati raccolti con le API gratuite messe a disposizione da football-data.

In particolare sarà possibile, utilizzare le seguenti regole:

- total_win_percent(): prevedere quale sarà la percentuale di vittoria di una determinata squadra contro la squadra avversaria;
- over_under(): prevedere quale sarà la media dei goal che verranno segnati in una partita;
- goal_or_not(): prevedere se entrambe le squadre segneranno almeno un goal in una partita;
- goal_odd_even(): se la somma dei goal della partita sarà pari oppure dispari.

Con BetTactics, inoltre, sarà possibile ottenere ulteriori informazioni specifiche, in particolare:

- il numero di vittorie, sconfitte e pareggi di una squadra;
- il numero di goal segnati e subiti da una squadra;
- i punti della classifica di una squadra;
- la differenza reti di una squadra (ossia la differenza tra i goal segnati e i goal subiti da una squadra);
- il numero di partite vinte, pareggiate e perse in casa di una squadra e la relativa percentuale rispetto al totale;
- il numero di partite vinte, pareggiate e perse fuori casa di una squadra e la relativa percentuale rispetto al totale;
- trovare il prossimo incontro di una squadra;
- trovare la differenza della posizione in classifica tra due squadre;
- trovare la forma di una squadra (la percentuale di vittorie delle ultime cinque partite).

## Dichiarazione liberatoria di non responsabilità

Con il seguente paragrafo si intende sottolineare che gli sviluppatori di BetTactics non si assumono alcuna responsabilità nel caso in cui quest'ultimo venga utilizzato per effettuare delle scommesse sportive, inoltre, l'affidabilità del programma non è garantita.

Consultare le probabilità di vincita nelle pagine dedicate.

Le scommesse sono vietate ai minori di 18 anni e possono causare dipendenza patologica.

Gioca responsabilmente.

# Come installare BetTactics

Per utilizzare BetTactics ci sono due possibili opzioni:

- con SWI-Prolog;
- con l'interfaccia grafica Ttk di Python.

In entrambi i casi la prima cosa che deve essere fatta è aprire il progetto all'interno del proprio editor (VS-Code consigliato), scaricandolo dalla repository di BetTactics.

A questo punto, se si vuole utilizzare SWI-Prolog, è sufficiente installarlo seguendo la documentazione. I file per lavorare con BetTactics si trovano all'interno della cartella nella quale è stato scaricato il progetto, per questo motivo dobbiamo cambiare la directory di lavoro all'interno di SWI-Prolog con il comando

```
working_directory(_, "YOUR\\PATH\\BetTactics\\swi_prolog").
```

Per poter verificare le regole e i fatti all'interno di SWI-Prolog si dovranno consultare i file base_functions.pl e BetTacticsScript.pl.

Se si desidera utilizzare Python, dopo aver effettuato il download del progetto, sarà necessario installare Python nel vostro pc.

Per poter utilizzare BetTactics sarà necessario scaricare la dipendenza Tkk di Python eseguendo il seguente comando:

```
pip3 install tk
```

A questo punto siete pronti per poter iniziare ad utilizzare l'applicazione.

Per poter lavorare, si rimanda alla sezione successiva.

## Utilizzo in SWI-Prolog

Utilizzando SWI -Prolog, se si decide di utilizzare come directory di lavoro la cartella swi-prolog nel progetto BetTactics in cui sono presenti i file .pl, è possibile eseguire direttamente le regole per la predizione dei risultati, altrimenti sarà necessario scaricare i knowledge base eseguendo delle regole specifiche.
BetTactics utilizza alcune regole per scaricare direttamente i knowledge base tramite API, per poi successivamente sfruttare le informazioni scaricate per poter predire i risultati desiderati o per visualizzare alcuni dati.

Per poter eseguire le regole che verranno mostrate, è necessario fare riferimento alla seguente tabella contenente i nomi e i codici delle squadre:

| **Squadra** | **Codice** |
|---|---|
| Napoli | 113 |
| Inter | 108 |
| Juventus | 109 |
| Roma | 100 |
| Lazio | 110 |
| Atalanta | 102 |
| Milan | 98 |
| Torino | 586 |
| Udinese | 115 |
| Bologna | 103 |
| Monza | 5911 |
| Empoli | 445 |
| Fiorentina | 99 |
| Lecce | 5890 |
| Sassuolo | 471 |
| Salernitana | 455 |

| Squadra | Codice |
|---|---|
| Spezia Calcio | 488 |
| Verona | 450 |
| Sampdoria | 584 |
| Cremonese | 457 |

La regola start_matches_SA prende in ingresso il nome di una squadra e il codice ad esso associata e genera un knowledge base con tutte le informazioni relative alle partite giocate e quelle ancora da giocare di quella squadra.
La seguente regola è limitata dalla stessa API che non ci consente di effettuare più di dieci chiamate al minuto, quindi, deve essere richiamata con un certo criterio per poter estrapolare tutti i dati nella maniera corretta:

```
start_matches_SA(Team,Code) :-
 atom_concat('database_',Team,X),
 atom_concat(X,'.pl',FileName),
      tell(FileName),
      write(':- module(\'database_'),
      write(Team),
      writeln('.pl\', [matchSA/5]).'),
      writeln(':- multifile(matchSA/5).'),
      list_json_array_matches_SA(ListMatchesSA,Code),
      take_matches_SA_list(ListMatchesSA.matches),
      fail.
start_matches_SA(_,_):-told.
```

La regola list_json_array_matches_SA() prende in ingresso il codice di una squadra, genera uno Stream che viene trasformato in un dict, il quale viene restituito dalla funzione stessa:

```
list_json_array_matches_SA(ListMatchesSA,X):-
  atom_concat('http://api.football-data.org/v4/teams/',X,Y),
  atom_concat(Y,'/matches?competitions=2019',Z),
  setup_call_cleanup(
      http_open(Z, In, [request_header('X-Auth-Token'='YOUR_TOKEN'),
      request_header('Accept'='application/json')]),
      json_read_dict(In,ListMatchesSA),
    close(In)
).
```

La regola take_matches_SA_list() prende in ingresso la lista generata dalla funzione precedente e scrive all'interno del file creato inizialmente tutte le gare di serie A di quella squadra:

```
take_matches_SA_list([]).
take_matches_SA_list([H|T]) :-
  write('matchSA('),
  writeq(H.homeTeam.name),  write(', '),
  writeq(H.awayTeam.name), write(', '),
  writeq(H.score.winner), write(', '),
  writeq(H.score.fullTime.home),write(', '),
  writeq(H.score.fullTime.away),writeln(').'),
  take_matches_SA_list(T).
```

La regola start_results crea un nuovo file che verrà popolato con le informazioni relative alla classifica della Serie A:

```
start_results :-
      tell('database_RESULTS_SA.pl'),
      list_json_array_results(List_res),
      maplist(get_dict(table), List_res.get(standings), R),
      select_first(X,R),
      member(Y,X),
      write('classifica('),
      write(Y.position), write(',"'),
      write(Y.team.shortName),write('", '),
      write(Y.team.id),write(', "'),
      write(Y.form),write('", '),
      write(Y.points),writeln(').'),
      fail.
start_results :- told.
```

La regola list_json_array_results() genera uno Stream che viene trasformato in un dict il quale viene restituito dalla funzione:

```
list_json_array_results(List_res):-
  setup_call_cleanup(
      http_open('http://api.football-data.org/v4/competitions/
                        SA/standings?2022',
      Stream,
      [request_header('X-Auth-Token'=YOUR_TOKEN'),
      request_header('Accept'='application/json')]),
      json_read_dict(Stream,List_res),
      close(Stream)
  ).
```

Dato che sono presenti tre table, la regola select_first() effettua la selezione della prima table:

```
select_first(X,R):-
  member(X,R),!.
```

Se ad esempio lanciassimo il seguente comando:

```
?- start_matches_SA("Atalanta BC", 102).
```

verrebbe generato il file database_Atalanta.pl contenente le seguenti informazioni:

```
:- module('database_Atalanta.pl', [matchSA/5]).
:- multifile(matchSA/5).
matchSA("Sampdoria", "Atalanta", "AWAY_TEAM", 0, 2).
matchSA("Atalanta", "Milan", "DRAW", 1, 1).
matchSA("Verona", "Atalanta", "AWAY_TEAM", 0, 1).
matchSA("Atalanta", "Torino", "HOME_TEAM", 3, 1).
matchSA("Monza", "Atalanta", "AWAY_TEAM", 0, 2).
matchSA("Atalanta", "Cremonese", "DRAW", 1, 1).
matchSA("Roma", "Atalanta", "AWAY_TEAM", 0, 1).
matchSA("Atalanta", "Fiorentina", "HOME_TEAM", 1, 0).
matchSA("Udinese", "Atalanta", "DRAW", 2, 2).
```

```
matchSA("Atalanta", "Sassuolo", "HOME_TEAM", 2, 1).
matchSA("Atalanta", "Lazio", "AWAY_TEAM", 0, 2).
matchSA("Empoli", "Atalanta", "AWAY_TEAM", 0, 2).
matchSA("Atalanta", "Napoli", "AWAY_TEAM", 1, 2).
matchSA("Lecce", "Atalanta", "HOME_TEAM", 2, 1).
matchSA("Atalanta", "Inter", "AWAY_TEAM", 2, 3).
matchSA("Spezia Calcio", "Atalanta", "DRAW", 2, 2).
matchSA("Bologna", "Atalanta", "AWAY_TEAM", 1, 2).
matchSA("Atalanta", "Salernitana", "HOME_TEAM", 8, 2).
matchSA("Juventus", "Atalanta", "DRAW", 3, 3).
matchSA("Atalanta", "Sampdoria", "HOME_TEAM", 2, 0).
matchSA("Sassuolo", "Atalanta", "HOME_TEAM", 1, 0).
matchSA("Lazio", "Atalanta", null, null, null).
matchSA("Atalanta", "Lecce", null, null, null).
matchSA("Milan", "Atalanta", null, null, null).
matchSA("Atalanta", "Udinese", null, null, null).
matchSA("Napoli", "Atalanta", null, null, null).
matchSA("Atalanta", "Empoli", null, null, null).
matchSA("Cremonese", "Atalanta", null, null, null).
matchSA("Atalanta", "Bologna", null, null, null).
matchSA("Fiorentina", "Atalanta", null, null, null).
matchSA("Atalanta", "Roma", null, null, null).
matchSA("Torino", "Atalanta", null, null, null).
matchSA("Atalanta", "Spezia Calcio", null, null, null).
matchSA("Atalanta", "Juventus", null, null, null).
matchSA("Salernitana", "Atalanta", null, null, null).
matchSA("Atalanta", "Verona", null, null, null).
matchSA("Inter", "Atalanta", null, null, null).
matchSA("Atalanta", "Monza", null, null, null).
```

Nel knowledge-base i primi due elementi rappresentano le squadre che hanno giocato un incontro, l'ordine determina se la squadra ha giocato in casa oppure fuori casa. Il terzo elemento indica l'esito della gara (vitto della squadra di casa, vittoria della squadra ospite oppure pareggio). Infine, il quarto e il quinto parametro indicano i goal che sono stati segnati dalle due squadre, ossia il risultato dell'incontro. Lanciando, invece, il comando:

```
?- start_results
```

verrebbe generato il file database_RESULTS_SA.pl contenente le seguenti informazioni:

```
classifica(1,"Napoli", 113, "W,W,W,W,W", 56).
classifica(2,"Inter", 108, "W,W,L,W,D", 43).
classifica(3,"Juventus", 109, "W,L,D,L,W", 41).
classifica(4,"Milan", 98, "W,L,L,L,D", 41).
classifica(5,"Roma", 100, "W,L,W,W,D", 40).
classifica(6,"Lazio", 110, "D,D,W,W,D", 39).
classifica(7,"Atalanta", 102, "L,W,D,W,W", 38).
classifica(8,"Torino", 586, "L,W,D,W,L", 30).
classifica(9,"Udinese", 115, "L,D,W,L,L", 29).
classifica(10,"Bologna", 103, "W,W,D,W,L", 29).
```

```
classifica(11,"Empoli", 445, "D,L,D,W,W", 27).
classifica(12,"Monza", 5911, "D,W,D,W,D", 26).
classifica(13,"Fiorentina", 99, "L,D,L,L,W", 24).
classifica(14,"Lecce", 5890, "W,L,L,D,D", 23).
classifica(15,"Sassuolo", 471, "W,W,D,L,L", 23).
classifica(16,"Salernitana", 455, "L,W,L,L,D", 21).
classifica(17,"Spezia Calcio", 488, "D,L,L,L,W", 19).
classifica(18,"Verona", 450, "D,D,W,L,W", 14).
classifica(19,"Sampdoria", 584, "D,L,L,L,L", 10).
classifica(20,"Cremonese", 457, "L,L,D,L,L", 8).
```

Nel precedente knowledge-base il primo e il secondo parametro ci mostrano il nome di una squadra e la sua relativa posizione in classifica. Il terzo parametro rappresenta il codice della squadra (quello utile a generare il primo database). Infine, il quarto ed il quindi parametro rappresentano la forma di una squadra (ossia gli esiti delle ultime cinque partite) e i suoi punti in classifica.

A questo punto sarà possibile lanciare una serie di funzioni per manipolare questi dati ed effettuare delle previsioni riguardanti le partite.
Per prima cosa, è importante lanciare la funzione config, che consente di evitare che vengano generati i warning dovuti all'utilizzo di database differenti che definiscono gli stessi fatti.

```
config:-
    set_prolog_flag(warn_override_implicit_import,false).
```

Ora andiamo a mostrare alcune funzioni utili che abbiamo utilizzato:

```
% REGOLE UTILI

%questa funzione richiama start_matches_SA e ci consente
%di aggiornare il database di una squadra
start(TeamName, Cod, Result2):-
    consult('BetTacticsScript.pl'),
    start_matches_SA(TeamName, Cod),
    atom_string(TeamName, TeamName1),
    atom_concat('database_', TeamName1, Result1),
    atom_concat(Result1, '.pl', Result2),
    consult(Result2),
    use_module(Result2).

%questa funzione richiama start_results e
%ci consente di aggiornare la classifica
start_classifica:-
    consult('BetTacticsScript.pl'),
    start_results,
    consult('database_RESULTS_SA.pl').

%verifica se un numero è pari o dispari
pari(0).
pari(X):- X>0, X1 is X-1, dispari(X1).
dispari(X):- X>0, X1 is X-1, pari(X1).
```

```
%data una lista delle partite di una squadra,
%calcola quante volte il numero dei goal era pari e dispari
sum_list_oddeven([], 0, 0).
sum_list_oddeven([H|T], Even, Odd) :-
    sum_list_oddeven(T, Even1, Odd1),
    ((H == 'null') ->  Even = 0, Odd = 0, Even1 = 0, Odd1 = 0;
    (dispari(H)) -> Even is Even1 + 1, Odd = Odd1;
    Odd is Odd1 + 1, Even = Even1).

%permette di consultare il database di una squadra e leggere
%le sue regole
consult_team(TeamName, Result2):-
    atom_string(TeamName, TeamName1),
    atom_concat('database_',TeamName1,Result1),
    atom_concat(Result1,'.pl',Result2),
    consult(Result2),
    use_module(Result2).

%data una lista in ingresso ne somma gli elementi se non nulli
sum_list([], 0).
sum_list([H|T], Sum) :-
   sum_list(T, Rest),
   ((H == null) -> Sum is 0 + Rest;  Sum is H + Rest).

%funzione che ci permette di calcolare le occorrenze di
%una lista di una particolare stringa
count_occurrences([], _, 0).
count_occurrences([H|T], String, Count) :-
    (   H = String
    ->  count_occurrences(T, String, TailCount),
        Count is TailCount + 1;
        count_occurrences(T, String, Count)
    ).
```

Queste regole ci hanno permesso di alleggerire la scrittura di altre, dato che venivano richiamate svariate volte, oppure di semplificare notevolmente alcuni calcoli necessari all'estrapolazione dei dati.

La prima macro regola ci permette di prevedere con quale percentuale una squadra riuscirà a vincere la sua prossima partita. Ad esempio:

```
?- total_win_percent("Atalanta BC", 102).
```

Restituirà:

```
La prossima partita di Atalanta : Lazio - Atalanta
Atalanta ha una percentuale di vittoria pari al 65.48 %
true.
```

La percentuale viene ottenuta attraverso l'utilizzo di svariate regole che estrapolano i dati dai knowledge base che sono stati mostrati in precedenza.
In particolare:

```
%winner_home trova tutte le partite in cui la squadra
%ha vinto in casa
winner_home(Team,Result, Result2):-
    findall(Team,Result2:matchSA(Team,_,"HOME_TEAM",_,_),Result).

%sfrutta la lista creata precedentemente e ne conta le occorrenze
num_winner_home(Team,WinHome, Result2):-
    winner_home(Team,Result, Result2),
    length(Result,WinHome).

%winner_home trova tutte le partite in cui la squadra
%ha vinto fuori casa
winner_away(Team,Result,  Result2):-
    findall(Team,Result2:matchSA(_,Team,"AWAY_TEAM",_,_),Result).

%sfrutta la lista creata precedentemente e ne conta le occorrenze
num_winner_away(Team,WinAway, Result2):-
    winner_away(Team,Result,  Result2),
    length(Result,WinAway).

%somma tutte le vittorie di una squadra
total_win(Team, Result, Result2):-
    num_winner_home(Team,WinHome, Result2),
    num_winner_away(Team,WinAway, Result2),
    Result is WinHome + WinAway.

%lose_home trova tutte le partite in cui la squadra ha perso in casa
lose_home(Team,Result, Result2):-
    findall(Team,Result2:matchSA(Team,_,"AWAY_TEAM",_,_),Result).

%sfrutta la lista creata precedentemente e ne conta le occorrenze.
num_lose_home(Team,LoseHome,  Result2):-
    lose_home(Team,Result, Result2),
    length(Result,LoseHome).

%lose_home trova tutte le partite in cui la squadra
%ha perso fuori casa
lose_away(Team,Result, Result2):-
    findall(Team,Result2:matchSA(_,Team,"HOME_TEAM",_,_),Result).

%sfrutta la lista creata precedentemente e ne conta le occorrenze
num_lose_away(Team,LoseAway, Result2):-
    lose_away(Team,Result,  Result2),
    length(Result,LoseAway).

%somma tutte le sconfitte di una squadra
total_lose(Team, Result,Result2):-
    num_lose_home(Team,LoseHome, Result2),
    num_lose_away(Team,LoseAway, Result2),
    Result is LoseHome + LoseAway.
```

```
%draw_home trova tutte le partite in cui la squadra
%ha pareggiato in casa
draw_home(Team,Result, Result2):-
    findall(Team,Result2:matchSA(Team,_,"DRAW",_,_),Result).

%sfrutta la lista creata precedentemente e ne conta le occorrenze
num_draw_home(Team,DrawHome, Result2):-
    draw_home(Team,Result, Result2),
    length(Result,DrawHome).

%draw_home trova tutte le partite in cui la squadra
%ha pareggiato fuori casa
draw_away(Team,Result,  Result2):-
    findall(Team,Result2:matchSA(_,Team,"DRAW",_,_),Result).

%sfrutta la lista creata precedentemente e ne conta le occorrenze
num_draw_away(Team,DrawAway, Result2):-
    draw_away(Team,Result, Result2),
    length(Result,DrawAway).

%somma tutti i pareggi di una squadra
total_draws(Team, Result, Result2):-
    num_draw_home(Team,DrawHome, Result2),
    num_draw_away(Team,DrawAway, Result2),
    Result is DrawHome + DrawAway.

%trova il numero delle partite giocate da una squadra
%grazie alla somma di vittorie, pareggi e sconfitte
total_matches_played(Team, Result, Result4):-
    total_win(Team, Result1, Result4),
    total_lose(Team, Result2, Result4),
    total_draws(Team, Result3, Result4),
    Result is  Result1 + Result2 + Result3.

%trova la percentuale di vittoria di una squadra
percent_win(Team, Result, Cod):-
    start(Team, Cod, Result4),
    total_win(Team, Result1, Result4),
    total_matches_played(Team, Result2, Result4),
    Result3 is Result1 / Result2,
    Result is Result3 * 100.

%trova la percentuale di sconfitta di una squadra
percent_lose(Team, Result, Cod):-
    start(Team, Cod, Result4),
    total_lose(Team, Result1, Cod, Result4),
    total_matches_played(Team, Result2, Result4),
    Result3 is Result1 / Result2,
    Result is Result3 * 100.
```

```
%trova la percentuale di pareggio di una squadra
percent_draws(Team, Result, Cod):-
    start(Team, Cod, Result4),
    total_draws(Team, Result1, Cod, Result4),
    total_matches_played(Team, Result2,Result4),
    Result3 is Result1 / Result2,
    Result is Result3 * 100.

%trova le partite giocate in casa di una squadra
played_home(Team,Result, Result2):-
    findall(Team,Result2:matchSA(Team,_,_,_,_),Result).

%trova le partite che ancora non sono state giocate
not_played_home(Team,Result, Result2):-
    findall(Team,Result2: matchSA(Team,_,null,_,_),Result).

%trova il totale delle partite effettivamente giocate in casa
tot_played_home(Team,Play_home, Result2):-
    played_home(Team,Result, Result2),
    not_played_home(Team,Result1, Result2),
    length(Result,X),
    length(Result1,Y),
    Play_home is X - Y.

%trova la percentuale di vittorie sulle partite giocate in casa
percent_win_home(Team,Result, Cod):-
    start(Team, Cod, Result2),
    num_winner_home(Team,Ris1, Result2),
    tot_played_home(Team,Ris2, Result2),
    Ris3 is Ris1 / Ris2,
    Result is Ris3 * 100.

%trova le partite giocate fuori casa di una squadra
played_away(Team, Result, Result2):-
    findall(Team, Result2:matchSA(_,Team,_,_,_), Result).

%trova le partite che ancora non sono state giocate
not_played_away(Team,Result, Result2):-
    findall(Team, Result2:matchSA(_,Team,null,_,_), Result).

%trova il totale delle partite effettivamente giocate fuori casa
tot_played_away(Team,Play_away, Result2):-
    played_away(Team,Result, Result2),
    not_played_away(Team,Result1, Result2),
    length(Result,X),
    length(Result1,Y),
    Play_away is X - Y.

%trova la percentuale di vittorie sulle partite giocate fuori casa
```

```
percent_win_away(Team,Result, Cod):-
    start(Team, Cod, Result2),
    num_winner_away(Team,Ris1, Result2),
    tot_played_away(Team,Ris2, Result2),
    Ris3 is Ris1 / Ris2,
    Result is Ris3 * 100.

%trova la prossima partita di una squadra
next_match(X,Y, TeamName, Cod):-
    start(TeamName, Cod, Result2),
    findall([Home,Away],Result2:matchSA(Home,Away,null,_,_),Result),
    member([X,Y],Result),!.

%trova la differenza tra due posizioni in classifica
%e ne trova la percentuale
position_difference_percent(RisPercentuale,TeamName, Cod, Home, Away):-
    next_match(Home,Away,TeamName,Cod),
    start_results,
    consult('database_RESULTS_SA.pl'),
    findall(Casa,classifica(Casa,Home,_,_,_),Result),
    findall(Trasferta,classifica(Trasferta,Away,_,_,_),Result2),
    (( Result < Result2 ) -> Ris is Result2-Result;
        Ris is Result-Result2),
    ((TeamName==Home)-> RisPercentuale is Ris*5;
        RisPercentuale1 is Ris*5,
     RisPercentuale is 100 - RisPercentuale1).

%calcola la percentuale della forma di una squadra,
%grazie al numero di "W" delle ultime cinque partite
forma(Team, NumW):-
    consult('database_RESULTS_SA.pl'),
    classifica(_,Team,_,X,_),
    split_string(X,',',',',Y),
    count_occurrences(Y,"W",NumW1),
    NumW2 is NumW1 / 5,
    NumW is NumW2 * 100.
```

Il calcolo della percentuale finale, dopo aver trovato il match imminente della squadra selezionata, sfrutta quattro regole, quali forma(), position_difference_percent(), percent_win(), percent_win_home() oppure percent_win_away() (a seconda che la squadra selezionata giochi in casa oppure no).

La seconda macro regola ci consente di prevedere quale sarà la percentuale di goal nella partita successiva di una squadra. Ad esempio:

```
?- over_under("Atalanta BC", 102)
```

Restituirà:

```
La prossima partita di Atalanta BC : SS Lazio - Atalanta BC
La media dei goal: 2.73
true.
```

La media dei goal viene ottenuta attraverso l'utilizzo di alcune funzioni che estrapolano i dati dai knowledge base che sono stati mostrati in precedenza.
In particolare:

```
%calcola i goal segnati dalla squadrea di casa
goal_team_home_scored(TeamName, Final):-
    consult_team(TeamName, Result2),
    findall(Goal, Result2:matchSA(TeamName, _, _, Goal,_), Result),
    tot_played_home(TeamName, PlayedSum, Result2),
    sum_list(Result, Sum),
    Final is Sum / PlayedSum.

%calcola i goal segnati dalla squadra fuori casa
goal_team_away_scored(TeamName, Final):-
    consult_team(TeamName, Result2),
    findall(Goal, Result2:matchSA(_, TeamName, _, _,Goal), Result),
    tot_played_away(TeamName, PlayedSum, Result2),
    sum_list(Result, Sum),
    Final is Sum / PlayedSum.

%calcola i goal subiti dalla squadra di casa
goal_team_home_sub(TeamName, Final):-
    consult_team(TeamName, Result2),
    findall(Goal, Result2:matchSA(TeamName, _, _, _,Goal), Result),
    tot_played_home(TeamName, PlayedSum, Result2),
    sum_list(Result, Sum),
    Final is Sum / PlayedSum.

%calcola i goal subiti dalla squadra ospite
goal_team_away_sub(TeamName, Final):-
    consult_team(TeamName, Result2),
    findall(Goal, Result2:matchSA(_, TeamName, _ ,Goal,_), Result),
    tot_played_away(TeamName, PlayedSum, Result2),
    sum_list(Result, Sum),
    Final is Sum / PlayedSum.
```

La terza macro regola ci permette di prevedere se nella gara successiva di una squadra entrambe le squadre segneranno oppure no .
Ad esempio:

```
?- goal_or_not("Atalanta BC", 102).
```

Restituirà:

```
Entrambe le squadre segneranno un goal
true.

Una delle due squadre non segnerà
true.
```

La percentuale viene ottenuta attraverso l'utilizzo di alcune regole che estrapolano i dati dai knowledge base che sono stati mostrati in precedenza.

Per un maggiore dettaglio, fare riferimento a quelle viste per la regola over_under().

La quarta macro regola ci consente di prevedere con quale percentuale una squadra riuscirà a vincere la prossima partita.
Ad esempio:

```
?- even_or_odd("Atalanta BC", 102).

Inserire risultato pari dispari.
true.
```

La percentuale viene ottenuta attraverso l'utilizzo di alcune funzioni che estrapolano i dati dai knowledge base che sono stati mostrati in precedenza.
Per un maggiore dettaglio, fare riferimento a quelle viste per la regola over_under().

## Utilizzo di Python

Se si sceglie di utilizzare Python sarà sufficiente eseguire nel terminale:

```
python main.py
```

Il main deve necessariamente essere lanciato prima di poter utilizzare l'interfaccia grafica, poichè ci permette di creare e aggiornare i database relativi ai match delle squadre.
Una volta lanciato il comando precedente, è possibile osservare che nella stessa cartella dove viene eseguito, verranno creati tanti file .pl quante sono le squadre di Serie A.
Il codice relativo al main è il seguente:

```python
import os
from pathlib import Path
from time import sleep
from pyswip import Prolog

prolog = Prolog()


def download_databases():
    prolog.consult("BetTacticsScript.pl")
    dbs_info = []  # lista contenente tuple con id, nome squadra
    file = open("database_RESULTS_SA.pl", "r")
    # estrazione id e nome da file database_RESULTS_SA
    for s in file.read().split("\n"):
        if "classifica" in s:
            s = s[s.index("(")+1:s.index(")")+1].split(",")
            id = s[2].lstrip()
            name = s[1].lstrip().replace("\"", "")
            dbs_info.append((id, name))
    # rimozione di tutti i file corrispondenti a dbs_info
    for id, name in dbs_info:
        fn = f"database_{name}.pl"
        if Path(fn).exists():
```

```
            os.remove(fn)
    # download files corrispondenti a dbs_info
    for id, name in dbs_info:
        query = f'start_matches_SA("{name}", {id}).'
        res = prolog.query(query)
        list(res)
        print(query)
        sleep(10)


download_databases()
```

A questo punto sarà possibile lanciare il file contente l'interfaccia grafica con il seguente comando:

```
python gui.py
```

Questo genererà una finestra contente l'interfaccia generata tramite ttk:

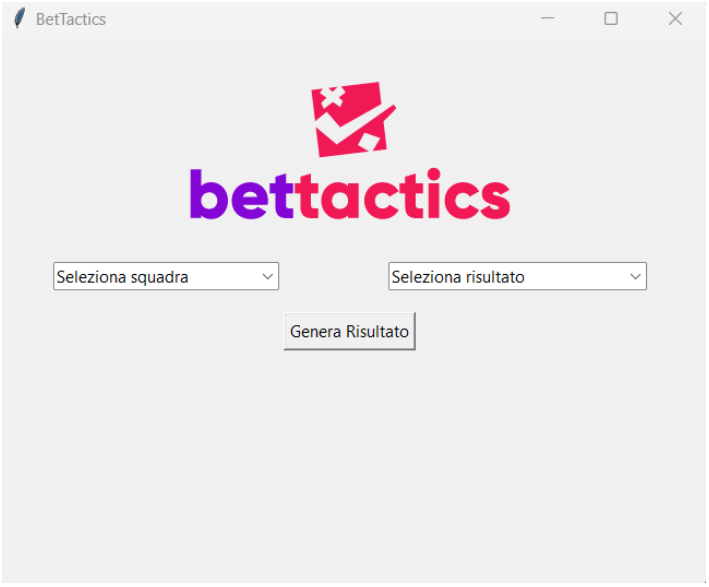



**Figura 1:** Schermata di base all'apertura dell'interfaccia grafica

A seconda della regole che si decide di selezionare, verrà mostrato un risultato diverso, come mostrato nelle seguenti figure:


**Figura 2:** Schermata che mostra i possibili goal dell'incontro


**Figura 3:** Schermata che mostra la potenziale percentuale di vittoria della squadra selezionata


**Figura 4:** Schermata che mostra se entrambe le squadre segneranno un goal oppure no


**Figura 5:** Schermata che mostra se la somma totale dei goal del match sarà pari o dispari

Oltre alle quattro macro regole appena mostrate, portiamo a titolo di esempio una regola relativa alle statistiche che possono essere calcolate:


**Figura 6:** Schermata che mostra il numero di vittorie nelle ultime cinque partite

Il codice per la gestione dell'interfaccia Python è il seguente:

```
import tkinter as tk
from tkinter import ttk
from pyswip import Prolog
from time import sleep
import codecs


prolog = Prolog()
prolog.consult("base_functions.pl")

window = tk.Tk()
window.title('BetTactics')
window.geometry('650x500')

result_label = ttk.Label()


class MyCombobox(ttk.Combobox):

    # le seguenti funzioni sfruttano la dipendenza di Prolog
    #        per effettuare delle query che sono presenti
    #nel file base_functions

    def total_win(self, event):
        key = self.get_key()
        query = f'total_win("{key}",X, \'{key}\').'
        results = list(prolog.query(query))
        for res in results:
            result_label = ttk.Label(
                window, text="Il numero di vittorie di " +
```

```
                key + " è: " + str(res["X"]))
            result_label.grid(column=0, row=6, columnspan=3)
            window.after(5000, lambda: result_label.destroy())

    def total_lose(self, event):
        key = self.get_key()
        query = f'total_lose("{key}",X, \'{key}\').'
        results = list(prolog.query(query))
        for res in results:
            result_label = ttk.Label(
                window, text="Il numero di sconfitte di " +
                key + " è: " + str(res["X"]))
            result_label.grid(column=0, row=6, columnspan=3)
            window.after(5000, lambda: result_label.destroy())

    def total_draws(self, event):
        key = self.get_key()
        query = f'total_draws("{key}",X, \'{key}\').'
        results = list(prolog.query(query))
        for res in results:
            result_label = ttk.Label(
                window, text="Il numero di pareggi di " +
                key + " è: " + str(res["X"]))
            result_label.grid(column=0, row=6, columnspan=3)
            window.after(5000, lambda: result_label.destroy())

    def total_goals_done(self, event):
        key = self.get_key()
        query = f'total_goal_do_team("{key}",X).'
        results = list(prolog.query(query))
        for res in results:
            result_label = ttk.Label(
                window, text="Il numero di goal fatti da " +
                key + " è: " + str(res["X"]))
            result_label.grid(column=0, row=6, columnspan=3)
            window.after(5000, lambda: result_label.destroy())

    def total_goals_conceded(self, event):
        key = self.get_key()
        query = f'total_goal_sub_team("{key}",X).'
        results = list(prolog.query(query))
        for res in results:
            result_label = ttk.Label(
                window, text="Il numero di goal subiti da " +
                key + " è: " + str(res["X"]))
            result_label.grid(column=0, row=6, columnspan=3)
            window.after(5000, lambda: result_label.destroy())

    def points(self, event):
        key = self.get_key()
```

```
        query = f'points("{key}",X).'
        results = list(prolog.query(query))
        for res in results:
            result_label = ttk.Label(
                window, text="Il numero di punti fatti da " +
                key + " è: " + str(res["X"]))
            result_label.grid(column=0, row=6, columnspan=3)
            window.after(5000, lambda: result_label.destroy())

    def goal_difference_team(self, event):
        key = self.get_key()
        query = f'goal_difference_team("{key}",X).'
        results = list(prolog.query(query))
        for res in results:
            result_label = ttk.Label(
                window, text="La differenza reti di " +
                key + " è: " + str(res["X"]))
            result_label.grid(column=0, row=6, columnspan=3)
            window.after(5000, lambda: result_label.destroy())

    def next_match(self, event):
        key = self.get_key()
        query = f'next_match(X,Y, "{key}").'
        results = list(prolog.query(query))
        for res in results:
            result_label = ttk.Label(
                window, text="La prossima partita di " +
                key + " è: " + str(res["X"].decode(encoding="UTF-8"))
                          + " - "
                + str(res["Y"].decode(encoding="UTF-8")))
            result_label.grid(column=0, row=6, columnspan=3)
            window.after(5000, lambda: result_label.destroy())

    def position_difference_percent(self, event):
        key = self.get_key()
        query = f'position_difference_percent(X, "{key}", W, Y, Z).'
        results = list(prolog.query(query))
        for res in results:
            result_label = ttk.Label(
                window, text="La prossima partita di " +
                key + " è: " + str(res["W"].decode(encoding="UTF-8"))
                          + " - "
                + str(res["Y"].decode(encoding="UTF-8")) +
                        "\n la differenza tra le posizioni in classifica è:
                str(res["Z"]))
            result_label.grid(column=0, row=6, columnspan=3)
            window.after(5000, lambda: result_label.destroy())

    def forma(self, event):
        key = self.get_key()
```

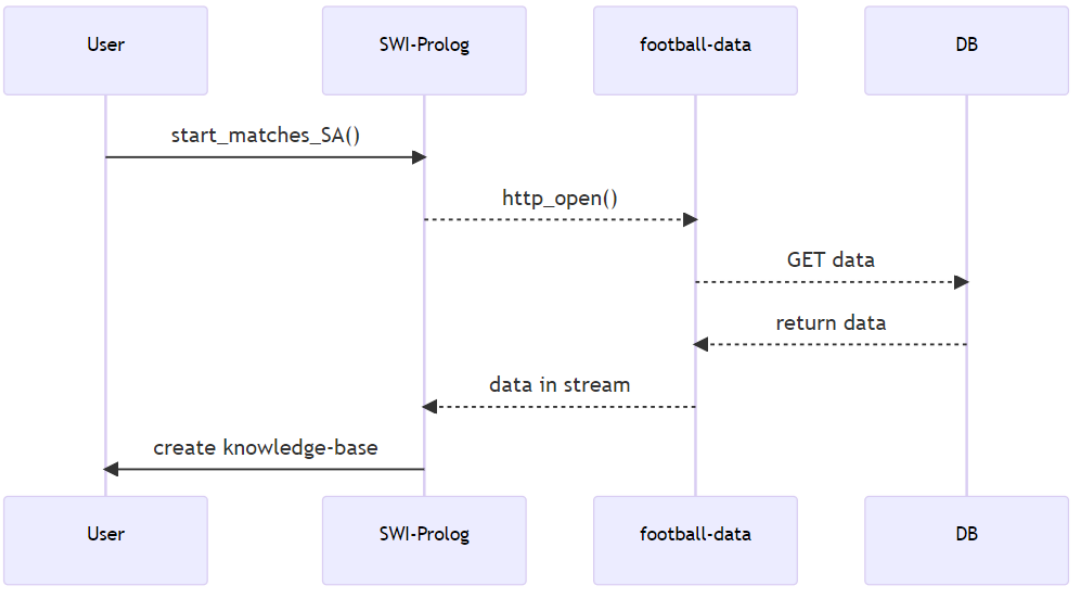
```
        query = f'forma("{key}",X, Z).'
        results = list(prolog.query(query))
        for res in results:
            result_label = ttk.Label(
                window, text="La forma di " +
                key + " è: " + str(res["X"]) + " %"
                "\n" + key + " ha vinto " + str(res["Z"]) +
                        " delle ultime 5 partite")
            result_label.grid(column=0, row=6, columnspan=3)
            window.after(5000, lambda: result_label.destroy())

    def over_under(self, event):
        key = self.get_key()
        query = 'over_under("{}", X, Y, Z).'.format(key)
        results = list(prolog.query(query))
        for res in results:
            XObject = res["X"]
            YObject = res["Y"].decode(encoding="UTF-8")
            ZObject = res["Z"].decode(encoding="UTF-8")
            result_label = ttk.Label(
                window, text="La prossima partita di " +
                key + " è: " + str(YObject) + " - "
                + str(ZObject) + "\nLa media dei goal è: " + str(XObject))
            result_label.grid(column=0, row=6, columnspan=3)
            window.after(5000, lambda: result_label.destroy())

    def total_win_percent(self, event):
        key = self.get_key()
        query = f'total_win_percent("{key}", X, Y, Z, \'{key}\').'
        results = list(prolog.query(query))
        for res in results:
            YObject = res["Y"].decode(encoding="UTF-8")
            ZObject = res["Z"].decode(encoding="UTF-8")
            result_label = ttk.Label(
                window, text="La prossima partita di "
                + key + " è: " + str(YObject) + " - " + str(ZObject)
                + "\nLa percentuale di vittoria di " + key + " è: "
                        + str(res["X"]) + " %")
            result_label.grid(column=0, row=6, columnspan=3)
            window.after(5000, lambda: result_label.destroy())

    def goal_nogoal(self, event):
        key = self.get_key()
        query = 'goal_or_not("{}", X, Y, W, Z).'.format(key)
        results = list(prolog.query(query))
        for res in results:
            XObject = res["X"].decode(encoding="UTF-8")
            YObject = res["Y"].decode(encoding="UTF-8")
            if res["W"] > res["Z"]:
                result_label = ttk.Label(
```

```
                    window, text="La prossima partita di "
                    + key + " è: " + str(XObject) + " - " + str(YObject)
                    + "\nEntrambe le squadre segneranno almeno un goal")
                result_label.grid(column=0, row=6, columnspan=3)
                window.after(5000, lambda: result_label.destroy())
            else:
                result_label = ttk.Label(
                    window, text="La prossima partita di "
                    + key + " è: " + str(XObject) + " - " + str(YObject)
                    + "\nUna delle due squadre non segnerà")
                result_label.grid(column=0, row=6, columnspan=3)
                window.after(5000, lambda: result_label.destroy())

    def goal_odd_even(self, event):
        key = self.get_key()
        value = self.get_value()
        query = 'goal_odd_even("{}", X, Y, W, Z).'.format(key)
        results = list(prolog.query(query))
        for res in results:
            XObject = res["X"].decode(encoding="UTF-8")
            YObject = res["Y"].decode(encoding="UTF-8")
            if res["W"] > res["Z"]:
                result_label = ttk.Label(
                    window, text="La prossima partita di "
                    + key + " è: " + str(XObject) + " - " + str(YObject)
                    + "\nLa somma totale dei goal è PARI")
                result_label.grid(column=0, row=6, columnspan=3)
                window.after(5000, lambda: result_label.destroy())
            else:
                result_label = ttk.Label(
                    window, text="La prossima partita di "
                    + key + " è: " + str(XObject) + " - " + str(YObject)
                    + "\nLa somma totale dei goal è DISPARI")
                result_label.grid(column=0, row=6, columnspan=3)
                window.after(5000, lambda: result_label.destroy())

    def __init__(self, master=None, cnf={}, **options):

        self.dict = None

        # prende il dictionary dalle options e fa una lista di key
        if 'values' in options:
            if isinstance(options.get('values'), dict):
                self.dict = options.get('values')
                options['values'] = sorted(self.dict.keys())

        # costruttore
        ttk.Combobox.__init__(self, **options)

        # assegna le funzioni
```

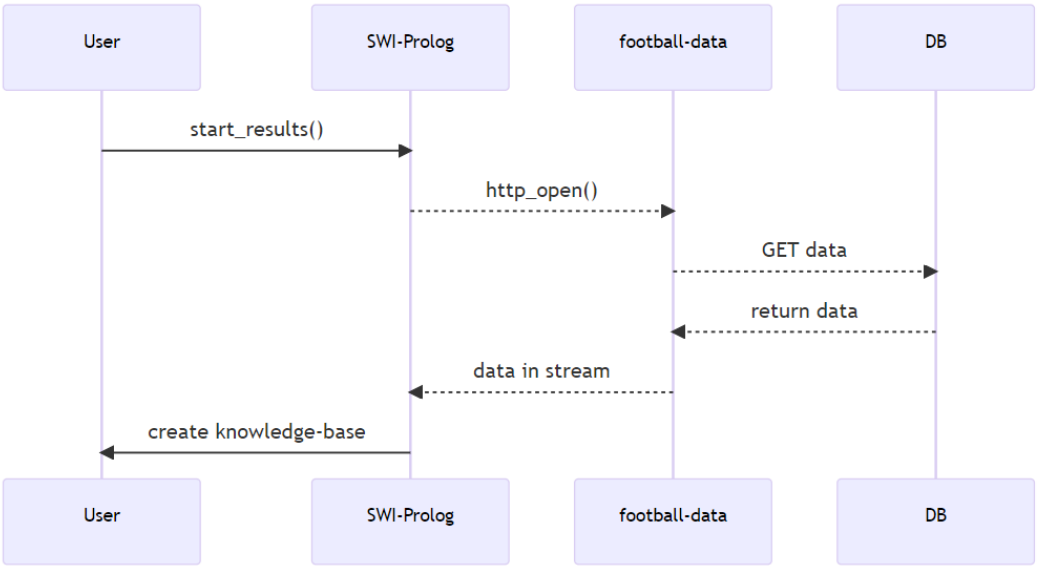
```
        self.bind('<<ComboboxSelected>>', self.on_select)

    def on_select(self, event):
        print(self.get(), self.get_key(), self.get_value())

    # sovrascrive `get()` per permetterci di ritornare
    #`value` invece di `key`
    def get(self):
        if self.dict:
            return self.dict[ttk.Combobox.get(self)]
        else:
            return ttk.Combobox.get(self)

    def get_key(self):
        return ttk.Combobox.get(self)

    def get_value(self):
        return self.get()


# select relativo alle funzioni
function_selector = ttk.Combobox(
    window, values=['over_under', 'total_win_percent', 'goal_nogoal',
                            'goal_odd_even','total_win', 'total_draws',
                            'total_lose', 'total_goals_done',
                            'total_goals_conceded', 'points',
                            'goal_difference_team', 'next_match',
                            'position_difference_percent', 'forma'])
function_selector.grid(column=1, row=1, ipadx=25, padx=50)
function_selector.set('Seleziona risultato')


def launch_function(event):
    # prende il nome della funzione selezionata
    function_name = function_selector.get()
    # prende la funzione selezionata
    selected_function = getattr(cb, function_name)
    # chiama la funzione selezionata
    selected_function(event)


# bottone che lancia le funzioni che vengono selezionate
launch_button = tk.Button(window, text='Genera Risultato')
launch_button.grid(column=0, row=3, columnspan=3, pady=20)
launch_button.bind('<Button-1>', launch_function)


# logo dell'applicazione
logo = tk.PhotoImage(file='logo.png')
labelLogo = tk.Label(window, image=logo)
```

```
labelLogo.grid(column=0, columnspan=4, row=0, padx=20)


# db utile alla creazione della select nell'interfaccia
dbs = {'Napoli': '113', 'Inter': '108', 'Atalanta': '102',
           'Roma': '100', 'Milan': '98', 'Lazio': '110',
           'Torino': '586', 'Udinese': '115', 'Juventus': '109',
           'Monza': '5911', 'Bologna': '103', 'Empoli': '445',
           'Lecce': '5890', 'Fiorentina': '99', 'Sassuolo': '471',
           'Salernitana': '455', 'Spezia Calcio': '488',
           'Verona': '450', 'Sampdoria': '584', 'Cremonese': '457'}
# select delle squadre
cb = MyCombobox(window, state='readonly', values=dbs)
cb.grid(column=0, row=1, ipadx=10, padx=50)
cb.set('Seleziona squadra')


window.mainloop()
```

Il meccanismo di funzionamento di BetTactics può essere mostrato con alcuni diagrammi delle sequenze.
In particolar modo andremo ora ad analizzare i diagrammi relativi alle chiamate API.
Il primo diagramma mostra l'interazione necessaria alla creazione del file dei risultati di Serie A di una squadra, inserendo il nome di una squadra e il relativo codice




Il secondo diagramma mostra un'altra interazione che avviene quando l'utente vuole scaricare la classifica della Serie A.


User
SWI-Prolog
football-data
DB
start_results()
http_open()
GET data
return data
data in stream
create knowledge-base
User
SWI-Prolog
football-data
DB

# Conclusioni

Giunti al termine del progetto, abbiamo deciso di analizzarne i punti di forza come la particolare accuratezza della funzione over_under() la quale ci consente di calcolare la media dei goal che potrebbero essere segnati in un particolare incontro (a seconda della squadra che viene selezionata dall'utente) anche in riferimento ad altri applicativi software di statistica calcistica.

L'applicazione presenta, però, dei limiti che sono dovuti principalmente all'utilizzo dell'API, la quale essendo gratuita non ci consente di effettuare un numero sufficientemente elevato di richieste che garantirebbero una maggiore fluidità del servizio ed, inoltre, la mole di dati che viene raccolta non ci permette di sfruttare i dati nella loro totalità per poter ottenere una maggior precisione.

Nonostante le limitazioni, possiamo ritenerci soddisfatti del lavoro compiuto e ci auguriamo in un futuro di poter estendere le previsioni anche ad altri campionati internazionali e altre potenziali funzioni di interesse.

Grazie per l'attenzione