UNIVERSITA DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Fondamenti di Intelligenza Artificiale

Prof. Fabio Palomba

FootballPredictor

Introduzione

Il calcio è lo sport più popolare al mondo.

Come ogni altro sport è per sua natura imprevedibile, infatti alcune partite di calcio hanno avuto risultati completamente diversi da quelli attesi.

Fare predizione sul calcio è di interesse ai fan, agli allenatori, ai media , agli scommettitori facendo così crescere il business delle scommesse sul calcio. Al momento ci sono molti siti di predizioni sui risultati delle partite, come ad esempio Forebet che però per il campionato di Premier Legue 17/18 riuscì a predire correttamente solo 2 su 10 partite di ogni giornata, questo dimostra il rischio di affidarsi a queste predizioni.

Svilupperemo un modello in grado di fare predizioni con meno rischi, più accurato e preciso.

Specifica PEAS

L’acronimo PEAS sta per Perfomance, Enviroment, Actuators, Sensors e viene utilizzato per descrivere un ambiente.

Perfomance: è la misura di prestazione adottata per valutare l’operato di un agente. Nel caso di FootballPredictor è predire se una squadra vince, perde o pareggia una partita.

Enviroment: è la descrizione degli elementi che formano l’ambiente. Nel nostro caso saranno l’insieme di squadre di calcio ed i risultati ottenuti.

Actuators: servono all’agente per poter compiere azioni. Nel nostro caso sarà lo schermo per mostrare le predizioni.

Sensors: permettono all’agente di ricevere input dall’esterno. Nel caso di FootballPredictor saranno il dataset da cui prendere informazioni.

Proprietà dell’ambiente

Completamente osservabile: i sensori forniscono all’agente l’accesso allo stato completo dell’ambiente in qualsiasi istante.

Deterministico: lo stato successivo dell’ambiente è determinato interamente dall’azione eseguita dall’agente.

Episodico: Ogni previsione sarà indipendente da un’altra.

Statico: l’ambiente non cambia mentre l’agente elabora.

Discreto: l’ambiente ha un numero di stati discreto, chiaramente definiti.

Singolo Agente: Vi è un ‘unica entità che prende decisioni nell’ambiente.

Ingegneria del Machine Learning: CRISP-DM

Il CRISP-DM rappresenta il ciclo di vita di progetti di intelligenza artificiale e data science. Si può paragonare ad un modello di ciclo di vita waterfall con feedback ed è composto : Business understanding, Data understanding, Data Preparation, Data modelling, Evalutation e deployment.

Business Understanding

In questa fase ci focalizziamo sullo stilare gli obiettivi di business e di elencare le tecnologie e tool utilizzati.

Analisi del problema

Possiamo notare che l'agente dovrà essere in grado di predire una variabile categorica, quindi modelleremo un classificatore.

Tecnologie e Tool necessari

Useremo il linguaggio Python e diverse sue librerie.

Pandas: Libreria per la manipolazione e analisi dei dati.

Sklearn:Libreria per il machine learning composta di algoritimi, metriche.

3.Data Understanding

In questa fase ci siamo occupati di analizzare i datasets necessari al raggiungimento degli obiettivi

3.1. Descrizione del dataset

Il dataset è composto da due stagioni del campionato di Premier Legue 20/21 e 21/22. Come descritto nella documentazione, ci si aspetterebbero 38(giornate) * 20(squadre) * 2 (stagioni) = 1520 righe, invece ci sono solo 1389.Questo poiché lo scraping è stato fatto mentre la stagione 2022 era ancora in corso.

`date` anno/mese/giorno in cui si è tenuta la partita
`time` orario in cui si è tenuta la partita
`comp` competizione per cui si è tenuta la partita
`round` giornata della competizione
`day` nome del giorno in cui si è tenuta la partita
`venue` partita in casa o fuori casa
`result` L lose, W win, D draw
`gf` goal che la squadra ha eseguito
`ga` goal che la squadra ha subito
`opponent` nome della squadra opponente
`xg` #goal attesi che la squadra esegua
`xga` #goal attesi che la squadra subisca
`captain` capitano della squadra
`formation` formazione della squadra
`referee` nome dell'arbitro
`sh` tiri che la squadra ha eseguito
`sot` tiri che la squadra ha eseguito alla porta
`fk` calcio di punizione eseguiti
`pk` calci di rigore eseguiti
`penalty kicks attemps` tentativi di calci di rigore
`season` anno della competizione
`team` squadra di casa

4.Data Preparation

4.1 Feature selection

Ora interveniamo risistemando le colonne, individuando cioè informazioni ridondanti o non necessarie.

`date` ,`time` ,`comp` ,`round` ,`venue` ,`result` ,`gf` , `ga` ,`opponent` ,`dist` ,`sh` ,`sot` ,`team` ,`fk` ,`pk` , `penalty kicks attemps` .

4.2 Data cleaning

Per pulire il dataset abbiamo utilizzato una serie di tecniche per renderli più coerenti con le fasi successive, tra queste l'eliminazione di righe con valori Nan all'interno.

4.3 Feature scaling

Ora dividiamo il dataset in due parti:

- training set: dati di addestramento
- test set: dati per valutare l'accuratezza del modello sulle predizioni che eseguirà

La scelta è stata di dividere il dataset per data ovvero per il train set di tutti i dati che hanno data inferiore al primo gennaio 2022, viceversa per il test set.

In questo modo si hanno 1100 dati per train set e circa 300 per il test set.

5.Data Modelling

In questa fase ci occupiamo della vera e propria modellazione dell'algoritmo di machine learning ovvero un regressore.

5.1 Scelta dell'algoritmo da utilizzare

Ricordiamo che l'obiettivo è di predire il numero di Goal segnati da una squadra in un match per questo siamo di fronte ad un regressore.

Esistono centinaia di algoritmi di regressione ma siamo interessati solo ad alcuni di essi :

- Decision Tree
- Random Forest

5.2 Addestramento

Ecco l'algoritmo per addrestramento:

```
train = grouped_matchesLast3Perfomed[grouped_matchesLast3Perfomed["date"] < '2022-01-01']

test = grouped_matchesLast3Perfomed[grouped_matchesLast3Perfomed["date"] > '2022-01-01']

predictors += ["gf_last3Performed","ga_last3Performed","sh_last3Performed","sot_last3Performed","dist_last3Performed","fk_last3Performed","pk_last3Performed","pkatt_last3Performed"]

rf.fit(train[predictors], train["target"])

preds = rf.predict(test[predictors])

print(precision_score(test["target"],preds))

combined = pd.DataFrame(dict(actual = test["target"], prediction = preds), index = test.index)
```

6 Valutazione

6.1 RandomForestClassifier

ACCURANCY	PRECISION
0.64	0.56

6.2 DecisionTreeClassifier

ACCURANCY	PRECISION
0.58	0.45

6.3 Naive Bayes

ACCURANCY	PRECISION
0.62	0.5

Il RandomForestClassifier è stato Il migliore tra tutti i modelli testati.