

Match Predictor


Un modello che prevede in base alle statistiche di una partita, l'eventuale risultato finale

Il dataset

European Soccer Data

Football match results and statistics from the 2023-2024 season

[Data Card](#) [Code \(2\)](#) [Discussion \(0\)](#) [Suggestions \(0\)](#)



About Dataset

This dataset includes 105 features for every single football match from the top European Leagues during the 2023-2024 season. It can be used for match analysis and prediction of future matches.

Please refer to notes.txt for the column descriptions.

Dataset collected from <https://www.football-data.co.uk/>.

Usability ⓘ
10.00

License
Other (specified in description)

Expected update frequency
Annually

Tags

[Sports](#) [Data Analytics](#)

il dataset preso in questione presenta i dati delle partite durante la stagione 2023-2024 dei maggiori campionati europei

Fase di importazione

fase di importazione delle varie librerie, tra cui sklearn per la standardizzazione dei dati e divisione del dataset (train test split)

Importazione librerie + Caricamento datasets

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam

df = pd.read_csv(r'/home/omori/Scrivania/its/deep_learning_venv/dataset_calcio/SerieA23.csv')
dfbundes = pd.read_csv(r'/home/omori/Scrivania/its/deep_learning_venv/dataset_calcio/Bundesliga23.csv')
dfliga = pd.read_csv(r'/home/omori/Scrivania/its/deep_learning_venv/dataset_calcio/LaLiga23.csv')
dfprem = pd.read_csv(r'/home/omori/Scrivania/its/deep_learning_venv/dataset_calcio/PL23.csv')

df_completo = pd.concat([df, dfbundes, dfliga, dfprem], ignore_index=True)

df.info()

df
df_completo
🔗 Open 'df_completo' in Data Wrangler
```

vengono inoltre importati anche i vari dataset, e caricati. Questi ultimi vengono poi caricati in un “dataset_completo” concatenandoli

Preprocessing dei dati

```
df_completo['FTR_num'] = df_completo['FTR'].map({'H':0, 'D':1, 'A':2}) #mappatura dei risultati utili

features = [ #seleziono solo le features che mi interessano per decidere una partita
    'HS', #Home shots
    'AS', #Away shots
    'HST', #Home shots on target
    'AST', #Away shots on target
    'AF', #Away fouls
    'HY', #Home YELLOW cards
    'AY', #Away YELLOW cards
    'HR', #Home RED cards
    'AR', #Away RED cards
    'HC', #Home corners
    'AC', #Away corners
    'HF', #Home fouls
]
```

```
X = df_completo[features]
y = df_completo['FTR_num'] #il target è il FULL TIME RESULT

scaler = StandardScaler() #standardizzo i dati
X_scaled = scaler.fit_transform(X)

#splitting dei dati
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Fase di preprocessing

1. creiamo una nuova colonna con “FTR_num” (full time result) mappandola con 0,1,2

2. Scegliamo la variabile target, in questo caso il “full time result” e poi dividiamo il set per permettere al modello di imparare al meglio

Creazione rete neurale

Rete Neurale

```
modello = Sequential([
    Dense(64, activation='relu', input_shape=X_train.shape[1:]), #creo un layer denso con 64 neuroni, attivazione relu
    Dropout(0.3), #questo parametro "azzeri i valori di alcuni neuroni" di volta in volta, per evitare overfitting
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(3, activation='softmax')
])

modello.compile(optimizer=Adam(learning_rate=0.0001), #ottimizzazione Adam, con learning rate bassissimo per permettere alla rete di non andare ad adattarsi troppo
                loss='sparse_categorical_crossentropy', #compatibilità molto buona con softmax
                metrics=['accuracy']) #per l'accuracy del modello

#addestramento
hs = modello.fit(X_train, y_train,
                epochs=500,
                batch_size=32, #in quanti "blocchi" vengono passati i dati
                validation_data=(X_test, y_test))
```

*fase di creazione del modello “Sequential”
con parametri “Dense” che rappresentano i layer densi
rispettivamente da 64 e 32 neuroni
e parametro “Dropout” che spegne i neuroni di volta in
volta per evitare l’overfitting*

il modello viene poi addestrato per 500 epoche con un
basso learning rate, sempre per evitare overfitting

Simulazione e predizione finale

dopo aver passato i dati di una partita in ordine come :

- Home Shots
- Away Shots
- Home shots on target
- Away shots on target
- Away fouls
- Home YELLOW cards
- Away YELLOW cards
- Home RED cards
- Away RED cards
- Home corners
- Away corners
- Home fouls

Simulazione di una partita

```
#dati di una partita in ingresso

match_example = np.array([[14 #Home shots
,8 #Away shots
,3 #Home shots on target
,13 #Away shots on target
,12 #Away fouls
,2 #Home YELLOW cards
,3 #Away YELLOW cards
,2 #Home RED cards
,4 #Away RED cards
,6 #Home corners
,4 #Away corners
,14 #Home fouls
]]) #passo secondo la posizione delle features i numeri delle statistiche di una partita

#features = [
#   'HS','AS','HST','AST',
#   'AF','HY','AY',
#   'HR','AR','HC','AC','HF'
#]
```

Predizione risultato finale

```
match_ex_scaled = scaler.transform(match_example) #normalizzo i dati del nuovo match
prediction = modello.predict(match_ex_scaled)

prob_vittoriaH = prediction[0][0]
prob_pareggio = prediction[0][1]
prob_vittoriaAW = prediction[0][2]

risultato = ['Vittoria Home','Pareggio','Vittoria Away'][np.argmax(prediction)]
print(f'Predizione risultato : {risultato}\ncon probabilità di Pareggio : {prob_pareggio:.1%},\nprobabilità di vittoria Casa : {prob_vittoriaH:.1%}\ne probabilità di vittoria Away : {prob_vittoriaAW:.1%}\n\n\n')
```

1/1 [=====] - 0s 42ms/step
Predizione risultato : Vittoria Away
con probabilità di Pareggio : 0.7%,
probabilità di vittoria Casa : 0.1%
e probabilità di vittoria Away : 99.3%

otteniamo una predizione sul risultato finale :

```
1/1 [=====] - 0s 42ms/step
Predizione risultato : Vittoria Away
con probabilità di Pareggio : 0.7%,
probabilità di vittoria Casa : 0.1%
e probabilità di vittoria Away : 99.3%
```