EPL Football Matches Predictions

CS 302: Artificial Intelligence

Zaiyr Sharsheev

BISHKEK 2021

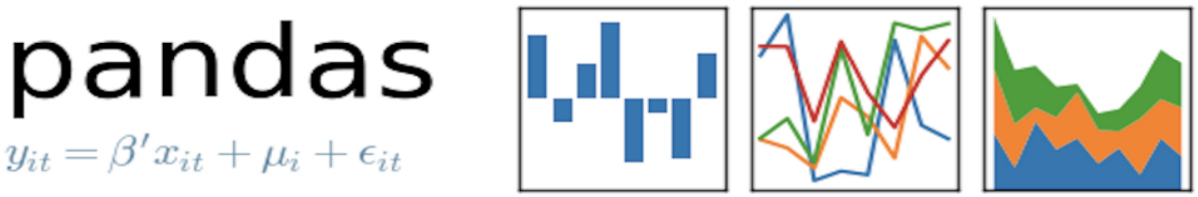
Table of Contents

ACKNOWLEDGEMENT

To begin with, I would like to say thanks to our lecturer **Mr. Selcuk Cankurt** for the knowledge that was gained during this year in the study of **Artificial Intelligence**. And also thanks to my classmates who helped to consolidate the knowledge gained and to even improve it somewhere.

INTRODUCTION

This project is based on developing an EPL (English Premier League) Football Matches Predictions using **Python Programming Language** and **Jupyter Notebook** (an open-source web application that allows to create and share documents that contain live code, equations, visualizations and narrative text). For that I used **Pandas** (an open source Python package that is most widely used for data science/data analysis and machine learning tasks), **Numpy** (a core library for scientific computing in Python), **SciPy** (a scientific library for Python for mathematics, science and engineering) and **Glob** (a module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell). You will see a more detailed explanation of each of them below.



EXPLANATIONS

In this report I have given explanations of what is the purpose of this project and how this EPL Football Matches Predictions works. I have explained here step by step so that it will surely help to become more understandable with it. Below are my explanations:

What is purpose?

The purpose of this project is to predict the score of the match based on the past games of these clubs since 1993. The time interval can be adjusted in the code, because since 1993 the balance of power of football clubs has changed a lot. I am a big fan of English football so I was very interested in this topic. Based on these results, you can roughly understand who is the favorite of the match. Although football is an unpredictable game, I think all fans of this sport would be interested to see the estimated score of the match and its percentage of the possible outcome.

Execution Procedure

Step №1

```
import pandas as pd
import numpy as np
from scipy.stats import poisson
from glob import glob
```

Figure 1. Import

I imported all the required libraries and modules that I described above. From scripy.stats I imported poisson. Poisson is a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time or space if these events occur with a known constant mean rate and independently of the time since the last event.

Step №2

```
files = glob('./statistics/EPL*.csv')

matches = pd.DataFrame()

for file in files:
    matches = pd.concat([
        matches,
        pd.read_csv(file, usecols=range(1,12), date_parser='pandas.to_datetime')])

matches['Date'] = pd.to_datetime(matches['Date'], dayfirst=True)
matches.dropna(how = 'all')
matches.to_csv('./all_matches.csv')
matches.info()
matches.head(10)
```

Figure 2. Data preparation

Here with this EPL*.csv pattern I got all the .csv files and with the DataFrame I converted the csv data directly into Dataframe objects. Then put all the available data into a new all_matches.csv file. EPL*.csv files contain all the game results of the EPL teams since 1993. These files I took from http://www.football-data.co.uk/englandm.php website.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11449 entries, 0 to 379
Data columns (total 18 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Date          10752 non-null  datetime64[ns]
 1   HomeTeam      10752 non-null  object
 2   AwayTeam      10752 non-null  object
 3   FTHG          10752 non-null  float64
 4   FTAG          10752 non-null  float64
 5   FTR           10752 non-null  object
 6   HTHG          9828 non-null   float64
 7   HTAG          9828 non-null   float64
 8   HTR           9828 non-null   object
 9   Unnamed: 10   0 non-null      float64
 10  Unnamed: 11   0 non-null      float64
 11  Referee       7928 non-null   object
 12  HS            6460 non-null   float64
 13  Unnamed: 7    0 non-null      float64
 14  Unnamed: 8    0 non-null      float64
 15  Unnamed: 9    0 non-null      float64
 16  Time          708 non-null    object
 17  Attendance    759 non-null    float64
dtypes: datetime64[ns](1), float64(11), object(6)
memory usage: 1.7+ MB
```

Figure 3. Output

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Unnamed: 10	Unnamed: 11	Referee	HS	Unnamed: 7	Unn
0	1996-08-17	Arsenal	West Ham	2.0	0.0	H	2.0	0.0	H	NaN	NaN	NaN	NaN	NaN	
1	1996-08-17	Blackburn	Tottenham	0.0	2.0	A	0.0	1.0	A	NaN	NaN	NaN	NaN	NaN	
2	1996-08-17	Coventry	Nott'm Forest	0.0	3.0	A	0.0	2.0	A	NaN	NaN	NaN	NaN	NaN	
3	1996-08-17	Derby	Leeds	3.0	3.0	D	0.0	1.0	A	NaN	NaN	NaN	NaN	NaN	
4	1996-08-17	Everton	Newcastle	2.0	0.0	H	2.0	0.0	H	NaN	NaN	NaN	NaN	NaN	
5	1996-08-17	Middlesbrough	Liverpool	3.0	3.0	D	2.0	2.0	D	NaN	NaN	NaN	NaN	NaN	
6	1996-08-17	Sheffield Weds	Aston Villa	2.0	1.0	H	0.0	0.0	D	NaN	NaN	NaN	NaN	NaN	
7	1996-08-17	Sunderland	Leicester	0.0	0.0	D	0.0	0.0	D	NaN	NaN	NaN	NaN	NaN	
8	1996-08-17	Wimbledon	Man United	0.0	3.0	A	0.0	1.0	A	NaN	NaN	NaN	NaN	NaN	
9	1996-08-18	Southampton	Chelsea	0.0	0.0	D	0.0	0.0	D	NaN	NaN	NaN	NaN	NaN	

Figure 4. Output

Step №3

```
last_10_matches = (matches['Date'] > '2010-07') & (matches['Date'] < '2021-05')

matches[last_10_matches].groupby(['HomeTeam', 'AwayTeam']).mean()
```

Figure 5. Results of the last 10 matches

Here I pull out the results of the last 10 matches and based on these results make a prediction. After that I return average home away goals thanks to the groupby() method.

		FTHG	FTAG	HTHG	HTAG
HomeTeam	**AwayTeam**				
Arsenal	**Aston Villa**	2.375	1.375	0.750000	0.625000
	Birmingham	2.000	1.000	1.000000	1.000000
	Blackburn	3.500	0.500	1.500000	0.500000
	Blackpool	6.000	0.000	3.000000	0.000000
	Bolton	3.500	0.500	0.500000	0.500000
...	...	...	...	...	...
Wolves	**Tottenham**	1.400	2.200	0.400000	1.200000
	Watford	1.000	1.000	0.500000	1.000000
	West Brom	2.000	3.000	1.666667	0.666667
	West Ham	2.000	1.000	0.750000	0.750000
	Wigan	2.000	1.500	0.500000	1.500000

1073 rows × 11 columns

Figure 6. Output

Step №4

```
matches[last_10_matches]['HomeTeam'].value_counts()
```

Figure 7. Counting result table

Here I got a table with teams that have has the most success at home games for the last 10 matches using value_counts() method

```
Man City          207
Liverpool         207
Arsenal           207
Chelsea           206
Everton           206
Man United        206
Tottenham         206
West Ham          188
Newcastle         187
West Brom         168
Southampton       167
Stoke             152
Crystal Palace    149
Aston Villa       149
Swansea           133
Sunderland        133
Leicester         131
Fulham            112
Burnley           111
Bournemouth        95
Watford            95
Norwich            95
Wolves             93
Brighton           73
```

Figure 8. Output

Step №5

```
h2h = matches[last_10_matches].groupby(['HomeTeam', 'AwayTeam']).mean()
matches[last_10_matches].groupby(['HomeTeam', 'AwayTeam']).get_group(('Man United', 'Liverpool')).sort_values('Date')
```

Figure 9. Output

Here I pull out and display the results between two teams (Man United and Liverpool) for the last 10 matches thanks to groupby() and get_group() methods.

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Unnamed: 10	Unnamed: 11	Referee	HS	Unnamed: 7	Unnamed: 8	Unnamed: 9	Time	Attendance
48	2010-09-19	Man United	Liverpool	3.0	2.0	H	1.0	0.0	H	NaN	NaN	H Webb	16.0	NaN	NaN	NaN	NaN	NaN
244	2012-02-11	Man United	Liverpool	2.0	1.0	H	0.0	0.0	D	NaN	NaN	P Dowd	11.0	NaN	NaN	NaN	NaN	NaN
217	2013-01-13	Man United	Liverpool	2.0	1.0	H	1.0	0.0	H	NaN	NaN	H Webb	15.0	NaN	NaN	NaN	NaN	NaN
289	2014-03-16	Man United	Liverpool	0.0	3.0	A	0.0	1.0	A	NaN	NaN	M Clattenburg	13.0	NaN	NaN	NaN	NaN	NaN
157	2014-12-14	Man United	Liverpool	3.0	0.0	H	2.0	0.0	H	NaN	NaN	M Atkinson	11.0	NaN	NaN	NaN	NaN	NaN
43	2015-09-12	Man United	Liverpool	3.0	1.0	H	0.0	0.0	D	NaN	NaN	M Oliver	9.0	NaN	NaN	NaN	NaN	NaN
209	2017-01-15	Man United	Liverpool	1.0	1.0	D	0.0	1.0	A	NaN	NaN	M Oliver	9.0	NaN	NaN	NaN	NaN	NaN
293	2018-03-10	Man United	Liverpool	2.0	1.0	H	2.0	0.0	H	NaN	NaN	C Pawson	5.0	NaN	NaN	NaN	NaN	NaN
268	2019-02-24	Man United	Liverpool	0.0	0.0	D	0.0	0.0	D	NaN	NaN	M Oliver	6.0	NaN	NaN	NaN	NaN	NaN
88	2019-10-20	Man United	Liverpool	1.0	1.0	D	1.0	0.0	H	NaN	NaN	M Atkinson	NaN	NaN	NaN	NaN	16:30	NaN

Figure 10. Output

Step №6

```
home_team = 'Man United'
away_team = 'Liverpool'

def get_score_results(home_team, away_team):
    home_mean = h2h.loc[(home_team, away_team)][0]
    away_mean = h2h.loc[(home_team, away_team)][1]

    home_score = poisson.rvs(home_mean, size=1)[0]
    away_score = poisson.rvs(away_mean, size=1)[0]

    return (home_score, away_score)

sims = {}
trials = 100
for i in range(trials):
    score = get_score_results(home_team, away_team)
    sims[score] = sims.get(score, 0) + 1

hist = []
for k, v in sims.items():
    p = v / trials
    hist.append((v, k, p))

hist.sort(reverse=True)
hist
```

Figure 11. Plot histogram of most probable results

Here I simulate score over 100 trials and plot histogram of most probable results using Poisson distribution in get_score_results() function.

```
[(14, (1, 1), 0.14),
 (13, (2, 0), 0.13),
 (12, (1, 0), 0.12),
 (10, (2, 2), 0.1),
 (10, (2, 1), 0.1),
 (6, (3, 0), 0.06),
 (6, (0, 0), 0.06),
 (4, (3, 2), 0.04),
 (4, (3, 1), 0.04),
 (4, (1, 2), 0.04),
 (3, (3, 3), 0.03),
 (2, (4, 1), 0.02),
 (2, (2, 3), 0.02),
 (2, (0, 2), 0.02),
 (2, (0, 1), 0.02),
 (1, (6, 0), 0.01),
 (1, (5, 2), 0.01),
 (1, (5, 1), 0.01),
 (1, (5, 0), 0.01),
 (1, (4, 3), 0.01),
 (1, (0, 4), 0.01)]
```

Figure 12. Output

Step №7

```
home_team = 'Man United'
away_team = 'Liverpool'
trials = 10000

def get_score_results(home_team, away_team):
    try:
        home_mean = h2h.loc[(home_team, away_team)][0]
        away_mean = h2h.loc[(home_team, away_team)][1]

        home_scores = poisson.rvs(home_mean, size=trials).astype(str)
        away_scores = poisson.rvs(away_mean, size=trials).astype(str)

        scores = pd.DataFrame(data={'home':home_scores, 'away':away_scores})
        scores['result'] = scores['home'] + '-' + scores['away']
        predictions = scores['result'].value_counts()
        probability = round(predictions / trials * 100, 1)

        return predictions.index[0], probability[0]

    except KeyError:
        return 'N/A', 'N/A'

get_score_results(home_team, away_team)
```

```
('1-1', 11.6)
```

Figure 13. Most probable result with output

Here I got the most probable result between two teams using Poisson distribution in get_score_results() function. If there are no records between two teams for the last 10 games, it returns N/A.

Step №8

```
home_teams = ['Arsenal', 'Aston Villa', 'West Ham', 'Wolves', 'Leicester', 'Liverpool', 'Leeds', 'Man City', 'Fulham', 'Sheffield United']
away_teams = ['Brighton', 'Chelsea', 'Southampton', 'Man United', 'Tottenham', 'Crystal Palace', 'West Brom', 'Everton', 'Newcastle', 'Burnl

weekday = pd.DataFrame(data={'Home':home_teams, 'Away':away_teams})
weekday['Predictions'] = weekday.apply(lambda x: get_score_results(x.Home, x.Away)[0], axis=1)
weekday['Probability of predictions%'] = weekday.apply(lambda x: get_score_results(x.Home, x.Away)[1], axis=1)
weekday
```

Figure 14. Simulates 2020/2021 last round of the season

Here I got the score predictions using the most probable result using DataFrame. It simulates 2020/2021 last round of the season

	Home	Away	Predictions	Probability of predictions%
0	Arsenal	Brighton	1-1	13.2
1	Aston Villa	Chelsea	0-1	11.8
2	West Ham	Southampton	2-1	9.2
3	Wolves	Man United	1-1	9.9
4	Leicester	Tottenham	1-2	10.6
5	Liverpool	Crystal Palace	2-1	8.8
6	Leeds	West Brom	N/A	N/A
7	Man City	Everton	1-0	14.7
8	Fulham	Newcastle	1-1	10.3
9	Sheffield United	Burnley	2-0	22.8

Figure 15. Output

REFERENCES

1. Results of all matches since 1993: http://www.football-data.co.uk/englandm.php
2. Abbreviations in tables: http://www.football-data.co.uk/notes.txt
3. Pandas library: https://pandas.pydata.org/
4. Numpy library: https://numpy.org/
5. Scipy library: https://scipy.org/scipylib/
6. Glob module https://docs.python.org/3/library/glob.html