National College of Ireland
Project Submission Sheet – 2021/2022

Student Name:	Kieran O Hanlon
Student ID:	x21190020
Programme:	HDSDA_JAN22OL_Y1 **Year:** ...2022.....
Module:	Programming for Data Analytics
Lecturer:	Athanasios Staikopoulos
Submission Due Date:	8th of May 2022
Project Title:	How expected goals can help predict football results
Word Count:	2240

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature: Kieran O Hanlon

Date:

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**
5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Introduction

“Analytics, statistics and the use of data in many different forms have taken the sporting world by storm in recent decades and has arguably been one of the most significant revolutions within the industry across the globe” (Severini, Thomas 2012). Association football is extremely difficult to predict as it is a very low scoring sport with many variables difficult to comprehend, for example, a deflected goal, an inspired goal keeper, poor finishing and sometimes just plain old luck. However, an article written in 2012 by Sam Green "Assessing the performance of Premier League goalscorers" developed an analytical model based on goal scoring opportunities which is now known as expected goals (xG). The most widely used method to predict xG in association football is the Poisson distribution because we know how many times an event has occurred. Unfortunately, this does mean the Poisson method is without its limitations. For example, there is a tendency the model can be excessive in the estimating the number of draws as well as underestimating the amount of matches that end in both or one of the teams not scoring (Tiippana, 2020). However, from my own conclusions and from academic literature the Poisson models estimated the successful team very well.

What is expected goals (xG)

According to understat.com, 10 parameters is derived from a neural network prediction algorithm to produce the output. Some of the parameters include how the ball is struck, distance to goal, penalty, open play et cetera. For analysts it can help determine a better long range forecast. The number of chances a team or player should have scored is evaluated between 0 and 1 determines there xG output. The average number of goals scored per match from the football_data file is 2.69 from 4,181 observations. With such a low rate there is insufficient data to forecast a team or players output in the long term. Although, shots on target and shots on goal can help predict the number of goals a team scores, it can be a biased estimation. This is where xG can give a different forecast because of the different characteristics that can be used. In other words, it is an averaged probability of shots taken from historical data and then used to determine an implied probability of a goal being scored or not.

Various scenarios to analyse xG:

- Through historical data it can be used to predict a probability of an upcoming match
- Regardless of a players short term form, xG can determine the quality of chances being created or not being created.
- Players playing with underperforming teams can be easier identified
- The defensive performance of a team can be better understood on how effective they are at preventing goal scoring opportunities.

Exploratory Analysis

For the purpose of this analysis I will use three datasets which I have sourced from the following:

- https://www.kaggle.com/datasets/slehkyi/extended-football-stats-for-european-leagues-xg
- https://www.football-data.co.uk/data.php

The first two datasets from Kaggle was scrapped from understat.com which includes 29 and 24 variables respectively. The per_game dataset has 24,580 rows while the under_stat dataset has 684 rows with no missing values in either. Furthermore, both sets of data run from 2014 to 2019 and include several variables from the top European leagues. These include England (EPL), Spain (La liga), Italy (Serie A), Germany (Bundesliga), France (Ligue Un) and Russia (RFPL). Not all variables will be used, but some of the most common I will use are non-penalty expected goals (npxG). I will use this variable instead of expected goal (xG) as penalties are skewed and represent a 70% of being scored. I will also focus on passes per defensive action (PPDA), passes allowed within 20 yards of goal excluding crosses (deep), opponents passes per defensive action (OPPDA), opponents passes allowed within 20 yards of goal excluding crosses (deep allowed) as well as wins, points (Pts) and expected points (xpts). The other dataset I will include from football-data.co.uk will include some other variables I am missing, for example, away team goals (FTAG). By running a univariate analysis I was able to reasonably conclude goals scored followed a Poisson distribution which I will go into more detail later. From this analysis the average number of home team goals (FTHG) is 1.53 goals per match and FTAG is 1.16 which roughly equates to a quarter less goals for the away team.

After running a correlation between some of the variables, for example, between pts and npxG I got a figure of .45 but after running a pearson correlation I got a figure of .76 which was a lot closer to 1.0. This seems more intuitive as the higher the npxG are the higher a team's points should be. The figured I was surprised with the negative correlation between npxG and PPDA. I felt the Pearson method was a better fit as it evaluates the linear relationship between two continuous variable opposed to the Spearman which is based on ranked values.

The Variables in both per_game and under_stat are:

- xG - expected goals metric, it is a statistical measure of the quality of chances created and conceded.
- xG_diff - difference between actual goals scored and expected goals.
- npxG - expected goals without penalties and own goals.
- xGA - expected goals against.
- xGA_diff - difference between actual goals missed and expected goals against.
- npxGA - expected goals against without penalties and own goals.
- npxGD - difference between "for" and "against" expected goals without penalties and own goals.
- ppda_coef - passes allowed per defensive action in the opposition half (power of pressure)
- oppda_coef - opponent passes allowed per defensive action in the opposition half (power of opponent's pressure)
- deep - passes completed within an estimated 20 yards of goal (crosses excluded)
- deep_allowed - opponent passes completed within an estimated 20 yards of goal (crosses excluded)
- xpts - expected points
- xpts_diff - difference between actual and expected points.

The variables from football_data are with no missing values in either:

- FTHG - Full-time home goals
- FTAG – Full-time away goals

(Source, understat.com)

Objective 1 – Determining which variables correlate with xG

Not surprising deep and npxG are highly correlated with a positive linear relationship. By developing a linear regression, illustrated in figure 4 below. An R-squared of 70% further illustrates this correlation and for every 14 deep passes npxG increases by 13%. However, what seems counter intuitive is the relationship between PPDA and npxG. Illustrated in figure 5 it shows a negative correlation between the two variables with an R-square of 17%. Further analysis will have to be conducted but the negative relationship could have more to do with the quality of teams that press efficiently as well as being a difficult metric to quantify as most analysis is conducted on the ball as opposed to off-ball which pressing is. Another caveat is that PPDA takes into account defensive actions and passes rather than the actual outcome. By creating separate data frames for each league the analysis was in line with no league as an outlier. La liga had the lowest correlation on Pearson method at -30% indicating a low to moderate negative correlation in general. According to soccerment.com buildup distribution percentage (BDP) would be a better metric to quantify counter pressing.

Figure 1:

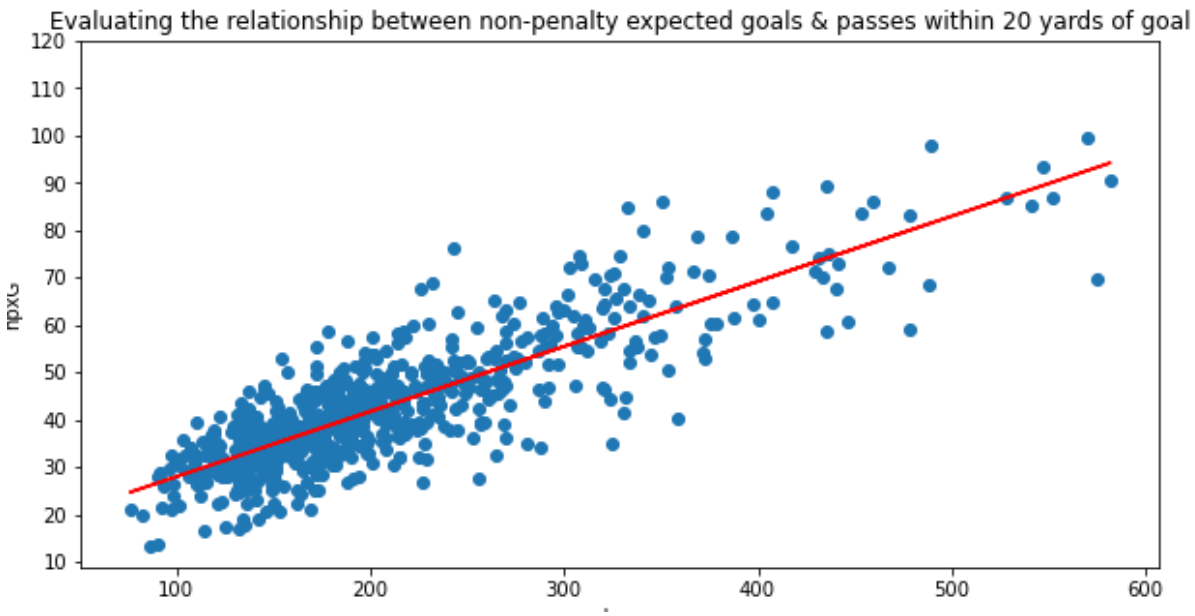
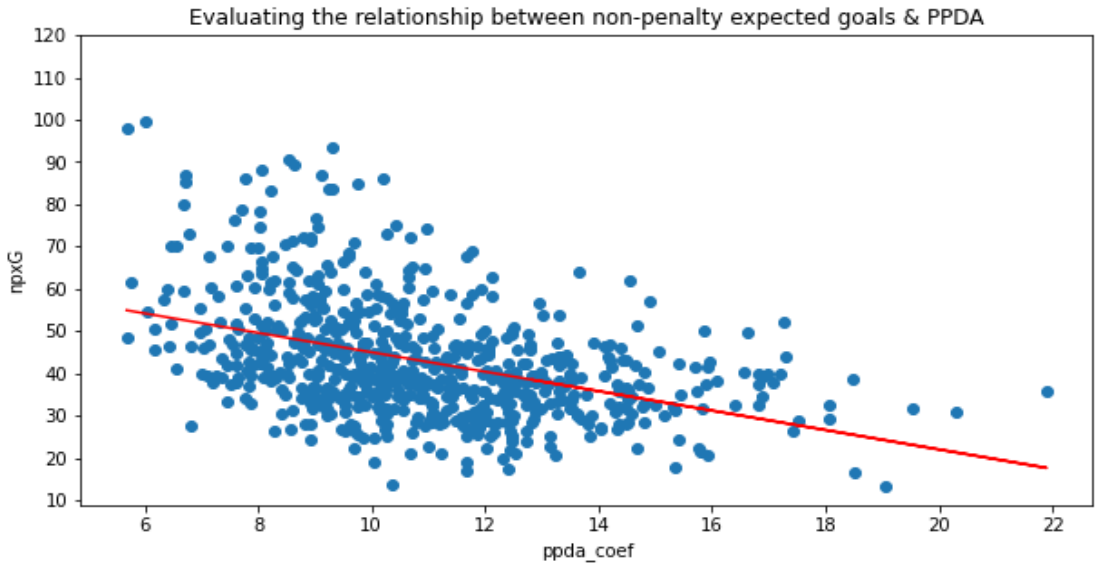


Figure 2:



Objective 2 – Comparing EPL Winners To There xG

Figure 3 below gives the us the EPL winners from 2014-2019. Leicester City being the huge exception who went off at 3000/1 to win the English Premier League (EPL). Figure 4 will compare their expected points (xpts) to the actual points they received.
As we can see from the above figure in 2018 Manchester City scored 95 goals with an xG of almost 94 whilst conceding 23 goals with an expected goals allowed (xGA) of almost 26. Although, Manchester City beat Liverpool by 1 point in 2018 their xpts was by 7 points better. This may just be down to luck or better finishing on Liverpool's half. However, the following year when Liverpool won the EPL they scored 85 goals with an xG of 75 whilst similarly again having an xGA of 6 goals higher than they conceded. This is illustrated in figure 3 below but in a season defining game when both teams played each other in the City of Manchester Stadium Man City Beat Liverpool 2-1 but Liverpool had an xG of .38 better and subsequently lost the league by one point.

Figure 3:

Figure 4:

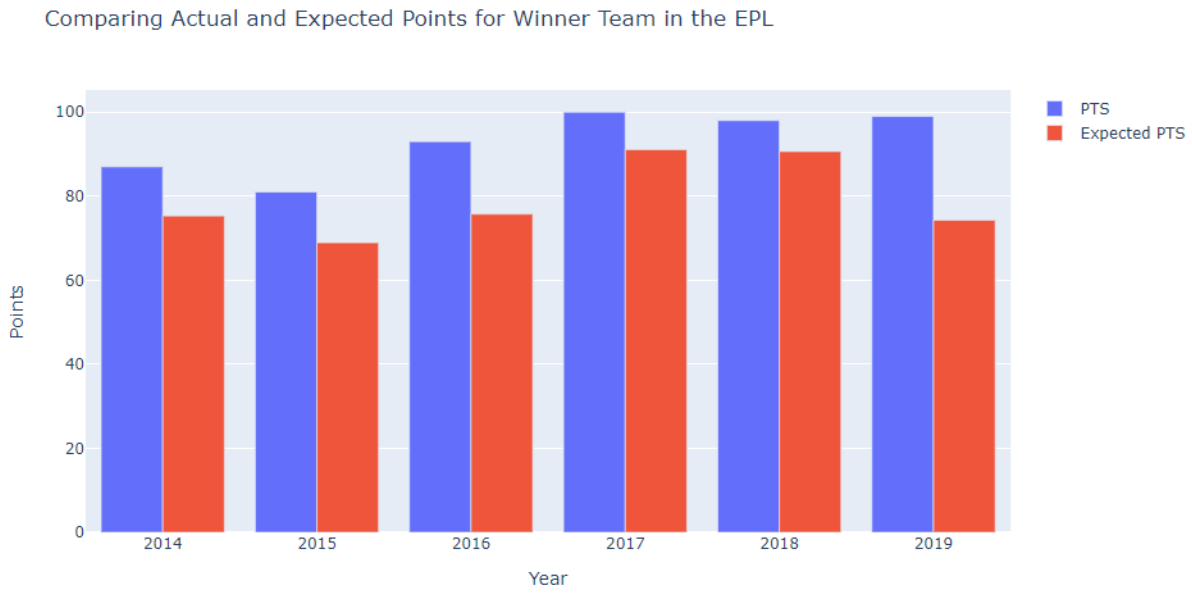


Figure 5a:

league	year	position	team	matches	wins	draws	loses	scored	conceded	pts	xG	xG_diff	npxG	xGA	xGA_diff
EPL	2019	2	Manchester City	38	26	3	9	102	35	81	102.21	0.21	93.53	37.00	2.00
EPL	2019	1	Liverpool	38	32	3	3	85	33	99	75.19	-9.81	71.39	39.57	6.57
EPL	2018	1	Manchester City	38	32	2	4	95	23	98	93.72	-1.28	90.68	25.73	2.73
EPL	2018	2	Liverpool	38	30	7	1	89	22	97	79.46	-9.54	74.13	29.15	7.15
EPL	2017	1	Manchester City	38	32	4	2	106	27	100	91.43	-14.57	85.12	24.51	-2.49
EPL	2017	2	Manchester United	38	25	6	7	68	28	81	59.04	-8.96	56.63	43.54	15.54
EPL	2016	1	Chelsea	38	30	3	5	85	33	93	61.80	-23.20	58.61	28.62	-4.38
EPL	2016	2	Tottenham	38	26	8	4	86	26	86	70.07	-15.93	63.22	33.78	7.78
EPL	2015	2	Arsenal	38	20	11	7	65	36	71	73.53	8.53	72.01	33.86	-2.14
EPL	2015	1	Leicester	38	23	12	3	68	36	81	68.42	0.42	58.52	45.02	9.02
EPL	2014	1	Chelsea	38	26	9	3	73	32	87	68.64	-4.36	64.74	31.52	-0.48
EPL	2014	2	Manchester City	38	24	7	7	83	38	79	75.82	-7.18	69.61	40.50	2.50

By plotting a z-score which is the number of standard deviations the mean is from a data point assuming >3 is an outlier we get these results. Liverpool had an xpts_diff of -24 and for 2018 and 2019 Manchester City's goals scored and conceded were almost identical to their respective expected goals scored and conceded. This goes some way to showing why the EPL is the strongest league in the world at the minute. Obviously huge amounts of money have been spent on players but by having arguably the two best managers in Pep Guardiola and Jurgen Klopp has only strengthened the league further. This is in contrast to Serie A which I is illustrated below.

Figure 6a:

league	year	position	team	matches	wins	draws	loses	scored	conceded	pts
Serie_A	2019	2	Inter	38	24	10	4	81	36	82
Serie_A	2019	1	Juventus	38	26	5	7	76	43	83
Serie_A	2018	2	Napoli	38	24	7	7	74	36	79
Serie_A	2018	1	Juventus	38	28	6	4	70	30	90
Serie_A	2017	2	Napoli	38	28	7	3	77	29	91
Serie_A	2017	1	Juventus	38	30	5	3	86	24	95
Serie_A	2016	1	Juventus	38	29	4	5	77	27	91
Serie_A	2016	2	Roma	38	28	3	7	90	38	87
Serie_A	2015	2	Napoli	38	25	7	6	80	32	82
Serie_A	2015	1	Juventus	38	29	4	5	75	20	91
Serie_A	2014	1	Juventus	38	26	9	3	72	24	87
Serie_A	2014	2	Roma	38	19	13	6	54	31	70

Figure 6b:

xG	xG_diff	npxG	xGA	xGA_diff	npxGA	npxGD	ppda_coef	oppda_coef	deep	deep_allowed	xpts	xpts_diff
73.95	-7.05	65.49	39.36	3.36	36.31	29.18	9.20	16.56	327	192	75.38	-6.62
73.77	-2.23	63.12	41.00	-2.00	31.64	31.48	9.04	15.61	270	174	71.24	-11.76
68.24	-5.76	65.19	36.50	0.50	33.46	31.74	10.72	18.07	344	181	74.45	-4.55
64.53	-5.47	58.44	35.03	5.03	31.99	26.45	10.16	13.96	290	158	70.93	-19.07
70.45	-6.55	64.18	25.27	-3.73	23.74	40.44	9.89	26.97	358	100	82.23	-8.77
59.23	-26.77	53.14	28.58	4.58	25.53	27.61	11.53	15.57	260	136	73.51	-21.49
68.74	-8.26	66.45	23.60	-3.40	22.84	43.62	9.50	13.00	302	117	82.86	-8.14
83.00	-7.00	72.34	41.08	3.08	38.03	34.31	8.92	13.63	354	201	77.39	-9.61
69.53	-10.47	63.44	26.69	-5.31	22.76	40.68	8.07	16.60	320	142	79.07	-2.93
62.99	-12.01	56.14	23.00	3.00	19.92	36.22	8.92	10.94	270	101	78.74	-12.26
59.08	-12.92	52.23	29.13	5.13	25.22	27.01	8.72	13.02	334	152	74.79	-12.21
50.85	-3.15	45.52	36.79	5.79	32.98	12.54	7.87	13.88	266	175	62.16	-7.84

Although Juventus having won Serie A 8 times in a row is an unbelievable achievement a lot of there success was down to the deficiencies of their opponents. As we can see from the past 3 years the runners up have had beaten the winners Juventus three times on xpts.

Objective 3 – Competitiveness in European leagues

One way to measure competitiveness in football is through the concentration ratio. Industrial economists who are interested in how firms dominate a particular industry use this method. To use this method in the top European leagues we will have to determine the top n sides within a k team league, where n<k. From year-to-year the dominance of the top three can alternate relative to the strength of the rest of the teams. However, in subsequent seasons the opposition may spend more, change managers as well as tactics which can bring an improvement from the challengers. Inevitably the top sides improve again which is creating a huge imbalance. This is

proving to be particularly true for the EPL as the TV money from Champions League keeps growing exponentially. Figure 8, illustrates the average movements for the top teams each year with Ligue Un being particularly volatile in terms of movement which would imply a competitive league with the Bundesliga being the least the volatile. However, this method has its limitations as both those league's have had predominantly only one winner over the past decade.

Figure 7:

Percentage point movements in each league

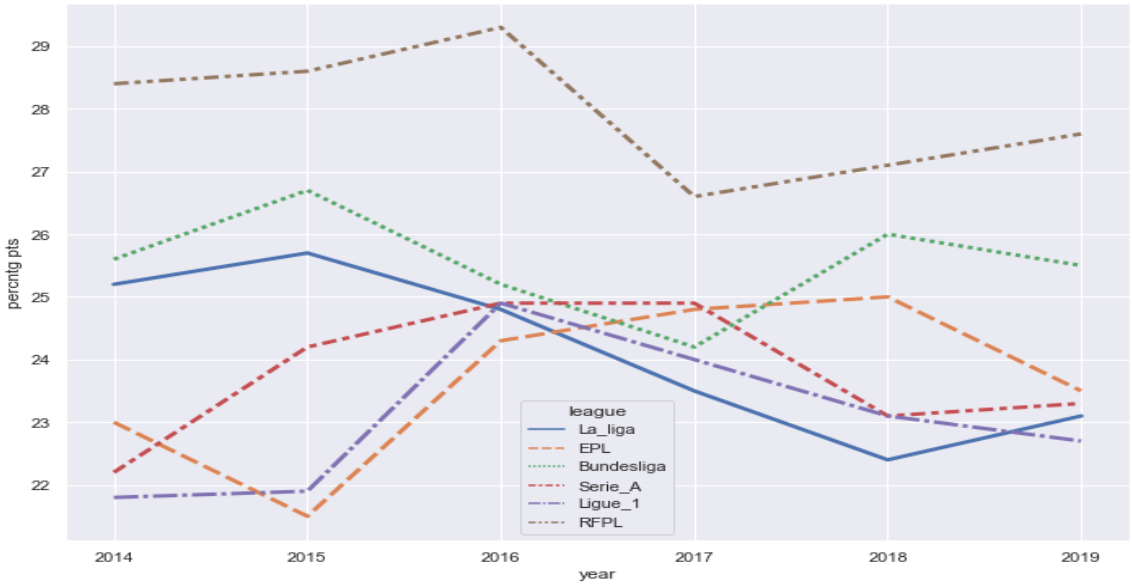
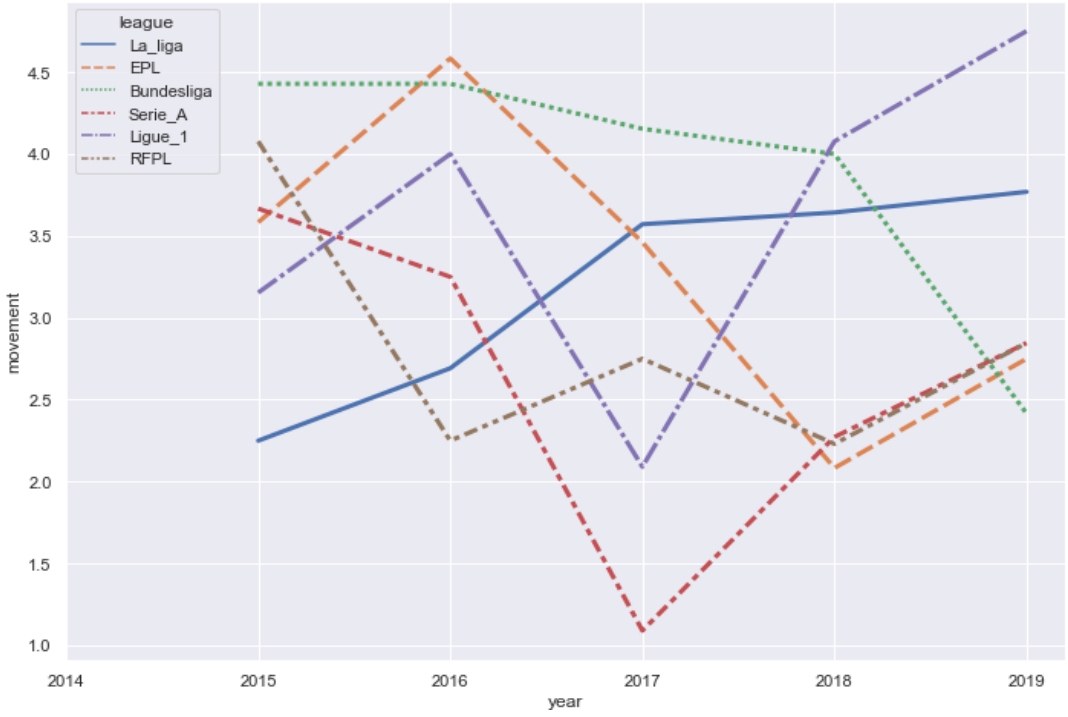


Figure 8:

Average movements per top teams in each league

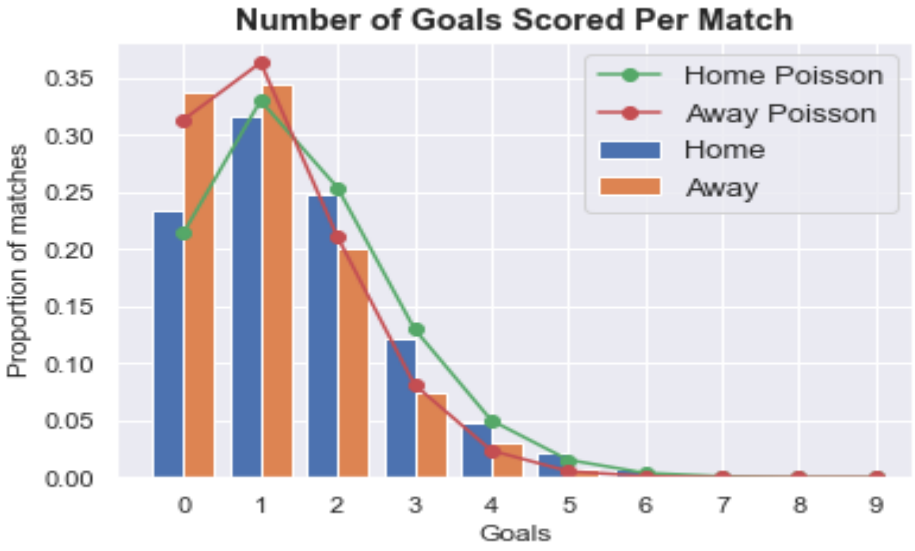


Objective 4 – Predicting Bookmaker Odds

After plotting goals scored (scored) I determined the distribution followed a Poisson distribution and was skewed to the right. This is illustrated in both figure eight and nine. A Poisson distribution will show over a specific period how likely an event is likely to occur. In other words it is a count method when events occur at a constant rate within a given interval of time (Investopedia).

Figure 9:

Figure 10:



I set the two data sets to a max of 10 goals as this would be a complete outlier in a football match and by getting home and away goals it gives a an accurate description that the home team generally scores more goals. Obviously it is not perfect but both sets are similar meaning we can approximate through a Poisson prediction method the number of goals scored by getting the average number of home and away goals. From the below Poisson analysis we can see Man City has an 80%(1/5) of chance of beating Man Utd. This looks to be an accurate interpretation considering the 4-1 and 2-0 score lines from earlier in the season and the goal parameters from figure . There is obvious limitations with this approach as many score lines might not be

independent of one another. For example, if the game is 0-0 with 10 or 15 minutes to go and team A might settle for this result. The other limitations is the over prediction of a draw at 13.5%

Figure 11:

```
goals = pois.predict("Man City", "Man United")
goals
```

```
: Module: Penaltyblog

  Class: FootballProbabilityGrid

  Home Goal Expectation: 2.635453519910054
  Away Goal Expectation: 0.6509818243552508

  Home Win: 0.7964127217430939
  Draw: 0.1348872668937208
  Away Win: 0.06869987678163515
```

Figure 12:

```
Module: Penaltyblog

Model: Poisson

Number of parameters: 41
Log Likelihood: -930.457
AIC: 1942.914

Team                 Attack               Defence
-------------------------------------------------------
Arsenal              1.202                -0.933
Aston Villa          0.988                -0.768
Brentford            0.953                -0.783
Brighton             0.637                -0.949
Burnley              0.615                -0.851
Chelsea              1.479                -1.292
Crystal Palace       1.018                -0.933
Everton              0.805                -0.593
Leeds                0.87                 -0.393
Leicester            1.155                -0.669
Liverpool            1.642                -1.5
Man City             1.594                -1.556
Man United           1.163                -0.719
Newcastle            0.872                -0.622
Norwich              0.341                -0.422
Southampton          0.892                -0.62
Tottenham            1.253                -0.972
Watford              0.696                -0.441
West Ham             1.142                -0.871
Wolves               0.683                -1.242
-------------------------------------------------------
Home Advantage: 0.147
```

Challenges

Not being able to source all data I would have liked, for example, BDP but the data I did get seems to be accurate in terms of the literature I have reviewed. Other analysis I would have liked to have conducted would have to been to split teams per year and make a comparison. Although, I was eventually able to split the data per league and per team when I did the same with year I was not able to analyse specific teams.

Conclusion

xG has become a widely accepted metric in association football and whilst it does draw a lot of correct forecasts, football in general is hard to predict with the low scoring nature of the game. Forecasting Man City to beat Man Utd at odds of 1/5 looks fair according to footballdata.co.uk Man City's starting price was 2/5 including the overround bookmakers include in their pricing. This is a 12% difference but the score line of 4-1 suggested this was reasonably accurate. However, the big discrepancy was the implied probability of 13% of the draw was an obvious limitation of the model as the starting price was over 20% at 15/4. It is suggested in the literature that the Dixon-Coles model can give a better estimate of future football matches. According to Tiippana 2020, the expected goals model is able to accurately predict 57% of results and 17% on correct with only the xG Poisson being a slight improvement on this. This may seem a poor forecast but football does have enormous complexities with many games ending with a low score due to the difficulty of scoring.

References

Arastey, G.M. (2018). What are expected goals [Online] https://www.sportperformanceanalysis.com/article/what-are-expected-goals-xg [Accessed 29th of April 2022]

Green, S (2012). Assessing the Performance of Premier League goalscorers/ [Online] https://www.optasportspro.com/news analysis/assessing-the-performance-of-premier-league-goalscorers/ [Accessed 29th of April 2022]

Soccerment Research (2022). Measuring pressing success. [Online]https://soccerment.com/measuring-pressing-success-buildup-disruption-percentage-bdp/ [Accessed 29th of April 2022]

The Open Source Dataset utilized for the purpose of this present report
lehkyi, S. (2020).Football Data: Expected Goals And Other Metrics. [online] https://www.kaggle.com/datasets/slehkyi/extended-football-stats-for-european-leagues-xg/code/ [Accessed 29th of April 2022]

Severini, Thomas, A. 2012. Analytic Methods in Geomechanics Analytic Methods in Sports. CRC Press

Football-data.co.uk. 2022. *Football Results, Statistics & Soccer Betting Odds Data*. [online] https://www.football-data.co.uk/data.php [Accessed 29 April 2022].

Eastwood, M., 2022. *pena.lt/y*. [online] http://www.pena.lt/y/2021/06/18/predicting-football-results-using-the-poisson-distribution/ [Accessed 29 April 2022]

iippana, T., 2022. [online] https://aaltodoc.aalto.fi/bitstream/handle/123456789/99805/bachelor_Tiippana_Tuomas2020.pdf [Accessed 29 April 2022]

Appendix

```
# Import Pandas Library, used for data manipulation
import pandas as pd

# Import matplotlib, used to plot our data
import matplotlib.pyplot as plt

# Import numpy for mathemtical operations
import numpy as np

# Import visualisations
import seaborn as sns

# Import Poisson
from scipy.stats import poisson

# Contains a function for rank probability in football
import penaltyblog as pb

# Import linear regression
from sklearn.linear_model import LinearRegression

# import plotly a scientific graphing library
import plotly
import plotly.figure_factory as ff
import plotly.graph_objs as go
import plotly.offline as py
from plotly.offline import iplot, init_notebook_mode
import plotly.tools as tls

# Importing the data

under_stat =
pd.read_csv(r'C:\Users\hanlo\Desktop\Programming\assignment\under_stat.csv')
under_stat = under_stat.rename(index=int, columns={'Unnamed: 0': 'league',
'Unnamed: 1': 'year'})
under_stat.head()

football_data =
pd.read_csv(r'C:\Users\hanlo\Desktop\Programming\assignment\football_data.csv')
per_game =
pd.read_csv(r'C:\Users\hanlo\Desktop\Programming\assignment\per_game.csv')
```

EDA

```
# check for missing values
under_stat.isnull().sum()

# check for missing values
```

```
per_game.isnull().sum()

football_data.isnull().sum()

# Barchart
plt.subplot(221)

per_game['scored'].value_counts().plot(kind='bar', title='Goals Scored',
figsize=(16,9))

plt.xticks(rotation=0)

per_game.boxplot('scored', 'missed')

xG = under_stat.pivot_table(index = "year", columns = "team", values = "xG",)
xG[["Liverpool", "Manchester City"]].replace(0, np.nan).fillna(method = "bfill", axis =
0).plot(xlabel= "Year", ylabel = "Expected Goals", title = "Expected Goals For Top
Teams By Year", figsize = (10,7))

# Creating separate DataFrames per top teams
liverpool = per_game[per_game['team'] == 'Liverpool']
liverpool.reset_index(inplace=True)

manutd = per_game[per_game['team'] == 'Manchester United']
manutd.reset_index(inplace=True)

bayern = per_game[per_game['team'] == 'Bayern Munich']
bayern.reset_index(inplace=True)

burnley = per_game[per_game['team'] == 'Burnley']
burnley.reset_index(inplace=True)

barca = per_game[per_game['team'] == 'Barcelona']
barca.reset_index(inplace=True)

atleti = per_game[per_game['team'] == 'Atletico Madrid']
atleti.reset_index(inplace=True)

torino = per_game[per_game['team'] == 'Torino']
torino.reset_index(inplace=True)

juve = per_game[per_game['team'] == 'Juventus']
juve.reset_index(inplace=True)

inter = per_game[per_game['team'] == 'Inter']
inter.reset_index(inplace=True)

napoli = per_game[per_game['team'] == 'Napoli']
napoli.reset_index(inplace=True)
```

```
psg = per_game[per_game['team'] == 'Paris Saint Germain']
psg.reset_index(inplace=True)

mancity = per_game[per_game['team'] == 'Manchester City']
mancity.reset_index(inplace=True)

# Set x to our passes completed within an estimated 20 yards of goal (crosses
excluded)
x = under_stat['scored']
# Set y to our non-penalty expected goals (nxpxG)
y = under_stat['npxG']

# Correlate passes within 20 yards of goal against npxG
plt.scatter(x, y)
# Add labels to the graph
plt.xlabel('scored')
plt.ylabel('npxG')

# To find the correlation among
# the columns using pearson method
#summarize the linear relationship between two variables
corr1 = under_stat.corr(method ='pearson')
# Pearson's r

corr1

corr1['npxG']['deep']

#changing the style
corr1.style.background_gradient(cmap="Blues").format(precision=3)

#visualise correlation with a heatmap
sns.heatmap(corr1, annot=True)
plt.show()

# Creating x & y variables
y = epl['npxG'] # dependent variable
x = epl['ppda_coef'] # independent var

# Summarise the linear relationship between two variables
corr2 = epl.corr(method ='pearson')
corr3 = seriea.corr(method ='pearson')
corr4 = bundesliga.corr(method ='pearson')
corr5 = rfpl.corr(method ='pearson')
corr6 = ligue1.corr(method ='pearson')

# EPL npxG V ppda_coef
corr1['npxG']['ppda_coef']
# Serie A npxG V ppda_coef
corr2['npxG']['ppda_coef']
```

```
# La Liga npxG V ppda_coef
corr3['npxG']['ppda_coef']
# Bundesliga npxG V ppda_coef
corr4['npxG']['ppda_coef']
# RFPL npxG V ppda_coef
corr5['npxG']['ppda_coef']
# Ligue 1 npxG V ppda_coef
corr6['npxG']['ppda_coef']

corr7 = per_game.corr(method ='pearson')

corr7
```

Determining which variables correlate with xG

```
under_stat.describe()

corr1['npxG']['deep']

corr1['npxG']['ppda_coef']

# Creating variables for non-penalty expected goals and passes within 20 metres of
goal(crosses excluded)
y = under_stat['npxG'] # dependent variable
x = under_stat['deep'] # independent var

# split data

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.15,
random_state=42)

from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

# Set x to our passes completed within an estimated 20 yards of goal (crosses
excluded)
x = under_stat['deep']
# Set y to our non-penalty expected goals (nxpxG)
y = under_stat['npxG']

# Correlate passes within 20 yards of goal against npxG
plt.scatter(x, y)
# Add labels to the graph
plt.xlabel('deep')
plt.ylabel('npxG')
```

```
from sklearn.linear_model import LinearRegression

x = under_stat['deep'].values.reshape(-1,1)
y = under_stat['npxG']

lr_model = LinearRegression()
lr_model.fit(x, y)
y_pred = lr_model.predict(x)

plt.scatter(x, y)
plt.xlabel('deep')
plt.ylabel('npxG')

plt.plot(x, y_pred)

# Intercept and Beta

theta_0 = lr_model.intercept_
theta_1 = lr_model.coef_
theta_0, theta_1

# R-Squared

model = LinearRegression().fit(x,y)

r_sq = model.score(x,y)
intercept = model.intercept_
slope = model.coef_

print(r_sq)

# Developing a linear relationship
y_pred = intercept + slope*x

fig, ax = plt.subplots(figsize = (10,5))

plt.scatter(x,y)

plt.plot(x,y_pred,c = 'red')

plt.yticks([10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120])

plt.xlabel('deep')
plt.ylabel('npxG')

plt.title('Evaluating the relationship between non-penalty expected goals & passes
within 20 yards of goal'

# Creating variables for non-penalty expected goals and power of pressure
y = under_stat['npxG'] # dependent variable
```

```
x = under_stat['ppda_coef'] # independent var

# Set x to ppda_coef, power of pressure
x = under_stat['ppda_coef']
# Set y to our non-penalty expected goals (nxpxG)
y = under_stat['npxG']

# Correlate power of pressure
plt.scatter(x, y)
# Add labels to the graph
plt.xlabel('ppda_coef')
plt.ylabel('npxG')

x = under_stat['ppda_coef'].values.reshape(-1,1)
y = under_stat['npxG']

lr_model = LinearRegression()
lr_model.fit(x, y)
y_pred = lr_model.predict(x)

plt.scatter(x, y)
plt.xlabel('ppda_coef')
plt.ylabel('npxG')

plt.plot(x, y_pred,c = 'red')

# Intercept and Beta

theta_0 = lr_model.intercept_
theta_1 = lr_model.coef_
theta_0, theta_1

# R-Squared

model = LinearRegression().fit(x,y)

r_sq = model.score(x,y)
intercept = model.intercept_
slope = model.coef_

print(r_sq)

y_pred = intercept + slope*x

fig, ax = plt.subplots(figsize = (10,5))

plt.scatter(x,y)

plt.plot(x,y_pred,c = 'red')
```

```
plt.yticks([10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120])

plt.xlabel('ppda_coef')
plt.ylabel('npxG')

plt.title('Evaluating the relationship between non-penalty expected goals & PPDA')
```

Differences Between Leagues

```
# Getting first place positions
first_place = under_stat[under_stat['position'] == 1]

# Get list of leagues
leagues = under_stat['league'].drop_duplicates()
leagues = leagues.tolist()

# Get list of years
years = under_stat['year'].drop_duplicates()
years = years.tolist()

first_place[first_place['league'] == 'EPL']

pts = go.Bar(x = years, y = first_place['pts'][first_place['league'] == 'EPL'], name =
'PTS')
xpts = go.Bar(x = years, y = first_place['xpts'][first_place['league'] == 'EPL'], name =
'Expected PTS')

data = [pts, xpts]

layout = go.Layout(
   barmode='group',
   title="Comparing Actual and Expected Points for Winner Team in the EPL",
   xaxis={'title': 'Year'},
   yaxis={'title': "Points",
   }
)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig)

# comparing with runner-ups
under_stat[(under_stat['position'] <= 2) & (under_stat['league'] ==
'EPL')].sort_values(by=['year','xpts'], ascending=False)

# Creating separate DataFrames per each league
laliga = under_stat[under_stat['league'] == 'La_liga']
laliga.reset_index(inplace=True)

epl = under_stat[under_stat['league'] == 'EPL']
epl.reset_index(inplace=True)
```

```
bundesliga = under_stat[under_stat['league'] == 'Bundesliga']
bundesliga.reset_index(inplace=True)

seriea = under_stat[under_stat['league'] == 'Serie_A']
seriea.reset_index(inplace=True)

ligue1 = under_stat[under_stat['league'] == 'Ligue_1']
ligue1.reset_index(inplace=True)

rfpl = under_stat[under_stat['league'] == 'RFPL']
rfpl.reset_index(inplace=True)

epl.describe()

# Outliers for xPTS
# EPL[(np.abs(zscore(EPL[['xpts_diff']])) > 2.0).all(axis=1)]
epl[(np.abs(zscore(epl[['xpts_diff']])) > 3.0).all(axis=1)]

# Outliers for xPTS
# EPL[(np.abs(zscore(EPL[['xpts_diff']])) > 2.0).all(axis=1)]
epl[(np.abs(zscore(epl[['xG']])) > 3.0).all(axis=1)]
```

Competitiveness Between Leagues

https://www.kaggle.com/code/adisri26/measure-competitiveness-in-football-leagues

```
# Percentage of points over x years
def comp(under_stat,n):
    a=[]
    for i in under_stat.league.unique():
        for j in under_stat.year.unique():
            d=under_stat[(under_stat["year"]==j)&(under_stat["league"]==i)]
            a.append([i,j,((d["pts"].nlargest()[:n].sum()*100)/(d["pts"].sum())).round(1)])
    sns.set(rc={'figure.figsize':(11.7,8.27)})
    sns.lineplot(data=pd.DataFrame(a,columns=["league","year","percntg pts"]),
x="year", y="percntg pts", hue="league", style="league",linewidth = 3)


comp(under_stat,3)

# Ranking movement over x years
def move(under_stat,k):
    arr=[]
    for i in under_stat["league"].unique():
        for j in under_stat["year"].unique():
            if(j-1 not in under_stat["year"].unique()):
                continue
```

```
d=under_stat[(under_stat["league"]==i)&((under_stat["year"]==j)|(under_stat["year
"]==j-1))]
        d=d[["team","year","position","league"]]
        a=d[d["position"]<=k]["team"].unique()
        d=d[d["team"].isin(a)]
        b=list(d.groupby("team")["position"].diff().fillna(0))
        result = sum(abs(number) for number in b)
        arr.append([i,j,result/len(a)])
   return pd.DataFrame(arr,columns=["league","year","movement"])

under_stat["year"]=pd.to_numeric(under_stat["year"],errors='coerce')

sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.lineplot(data=move(under_stat,10), x="year", y="movement", hue="league",
style="league",linewidth = 3)
plt.gca().set_xticks(under_stat["year"].unique())

# Standard Deviation over x years
stat="pts"

def SD_stats(league,year,stat):
   d=under_stat[(under_stat["league"]==league) & (under_stat["year"]==year)]
   std=d[stat].std()
   return std

df_stat=[]
for i in under_stat.league.unique():
   for j in under_stat.year.unique():
      df_stat.append([i,j,SD_stats(i,j,stat)])
df_stat=pd.DataFrame(df_stat,columns=["league","year",stat+"_SD"])
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.lineplot(data=df_stat, x="year", y=stat+"_SD", hue="league",
style="league",linewidth = 3)
```

Poisson

http://www.pena.lt/y/2021/06/18/predicting-football-results-using-the-poisson-distribution/

```
"""
File path & first 5 rows of EPL
"""


football_data1 = pd.read_csv("https://www.football-
data.co.uk/mmz4281/2122/E0.csv")
football_data1[["Date", "HomeTeam", "AwayTeam", "FTHG", "FTAG"]].head()

"""
# Average No. of goals for home & away teams
"""
```

```
football_data1[["FTHG", "FTAG"]].mean()

"""
# Distribution of goals scored by the home and away teams
"""
sns.set()

max_goals = 10
plt.hist(
   football_data[["FTHG", "FTAG"]].values, range(max_goals), label=["Home",
"Away"], density=True
)
plt.xticks([i - 0.5 for i in range(1, max_goals + 1)], [i for i in range(max_goals)])
plt.xlabel("Goals")
plt.ylabel("Proportion of matches")
plt.legend(loc="upper right", fontsize=13)
plt.title("Number of Goals Scored Per Match", size=14, fontweight="bold")

"""
# Overlay the number of goals we expect from a Poisson distribution
"""

home_poisson = poisson.pmf(range(10), football_data1["FTHG"].mean())
away_poisson = poisson.pmf(range(10), football_data1["FTAG"].mean())

max_goals = 10
plt.hist(
   football_data1[["FTHG", "FTAG"]].values, range(max_goals), label=["Home",
"Away"], density=True
)

plt.plot(
   [i - 0.5 for i in range(1, max_goals + 1)],
   home_poisson,
   linestyle="-",
   marker="o",
   label="Home Poisson",
)

plt.plot(
   [i - 0.5 for i in range(1, max_goals + 1)],
   away_poisson,
   linestyle="-",
   marker="o",
   label="Away Poisson",
)


plt.xticks([i - 0.5 for i in range(1, max_goals + 1)], [i for i in range(max_goals)])
plt.xlabel("Goals")
```

```
plt.ylabel("Proportion of matches")
plt.legend(loc="upper right", fontsize=13)
plt.title("Number of Goals Scored Per Match", size=14, fontweight="bold")

"""
# Defence and attack parameters
"""

def log_likelihood(
   goals_home_observed,
   goals_away_observed,
   home_attack,
   home_defence,
   away_attack,
   away_defence,
   home_advantage,
):
   goal_expectation_home = np.exp(home_attack + away_defence +
home_advantage)
   goal_expectation_away = np.exp(away_attack + home_defence)

   if goal_expectation_home < 0 or goal_expectation_away < 0:
      return 10000

   home_llk = poisson.pmf(goals_home_observed, goal_expectation_home)
   away_llk = poisson.pmf(goals_away_observed, goal_expectation_away)

   log_llk = np.log(home_llk) + np.log(away_llk)

   return -log_llk

"""
# Defence and attack parameters
"""

def log_likelihood(
   goals_home_observed,
   goals_away_observed,
   home_attack,
   home_defence,
   away_attack,
   away_defence,
   home_advantage,
):
   goal_expectation_home = np.exp(home_attack + away_defence +
home_advantage)
   goal_expectation_away = np.exp(away_attack + home_defence)

   if goal_expectation_home < 0 or goal_expectation_away < 0:
      return 10000
```

```
    home_llk = poisson.pmf(goals_home_observed, goal_expectation_home)
    away_llk = poisson.pmf(goals_away_observed, goal_expectation_away)

    log_llk = np.log(home_llk) + np.log(away_llk)

    return -log_llk


pois = pb.poisson.PoissonGoalsModel(football_data1["FTHG"],
football_data1["FTAG"], football_data1["HomeTeam"],
football_data1["AwayTeam"])
pois.fit()

pois

"""
# Predicting probabilities of team X to win
"""

probs = pois.predict("Man City", "Man United")
print(probs.home_draw_away)
```