Predicting Football Results With Statistical Modelling

Combining the world's most popular sport with everyone's favourite discrete probability distribution, this post predicts football matches using the Poisson distribution.

June 04, 2017

Football (or soccer to my American readers) is full of clichés: "It's a game of two halves", "taking it one game at a time" and "Liverpool have failed to win the Premier League". You're less likely to hear "Treating the number of goals scored by each team as independent Poisson processes, statistical modelling suggests that the home team have a 60% chance of winning today". But this is actually a bit of cliché too (it has been discussed here, here, here, here and particularly well here). As we'll discover, a simple Poisson model is, well, overly simplistic. But it's a good starting point and a nice intuitive way to learn about statistical modelling. So, if you came here looking to make money, I hear this guy makes £5000 per month without leaving the house.

Poisson Distribution

The model is founded on the number of goals scored/conceded by each team. Teams that have been higher scorers in the past have a greater likelihood of scoring goals in the future. We'll import all match results from the recently concluded Premier League (2016/17) season. There's various sources for this data out there (kaggle, football-data.co.uk, github, API). I built an R wrapper for that API, but I'll go the csv route this time around.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn
from scipy.stats import poisson,skellam

epl_1617 = pd.read_csv("http://www.football-data.co.uk/mmz4281/1617/E0
epl_1617 = epl_1617[['HomeTeam','AwayTeam','FTHG','FTAG']]
epl_1617 = epl_1617.rename(columns={'FTHG': 'HomeGoals', 'FTAG': 'Away
epl_1617.head()
```

	HomeTeam	AwayTeam	HomeGoals	AwayGoals
0	Burnley	Swansea	0	1
1	Crystal Palace	West Brom	0	1
2	Everton	Tottenham	1	1
3	Hull	Leicester	2	1
4	Man City	Sunderland	2	1

We imported a csv as a pandas dataframe, which contains various information for each of the 380 EPL games in the 2016-17 English Premier League season. We restricted the dataframe to the columns in which we're interested (specifically, team names and numer of goals scored by each team). I'll omit most of the code that produces the graphs in this post. But don't worry, you can find that code on my github page. Our task is to model the final round of fixtures in the season, so we must remove the last 10 rows (each gameweek consists of 10 matches).

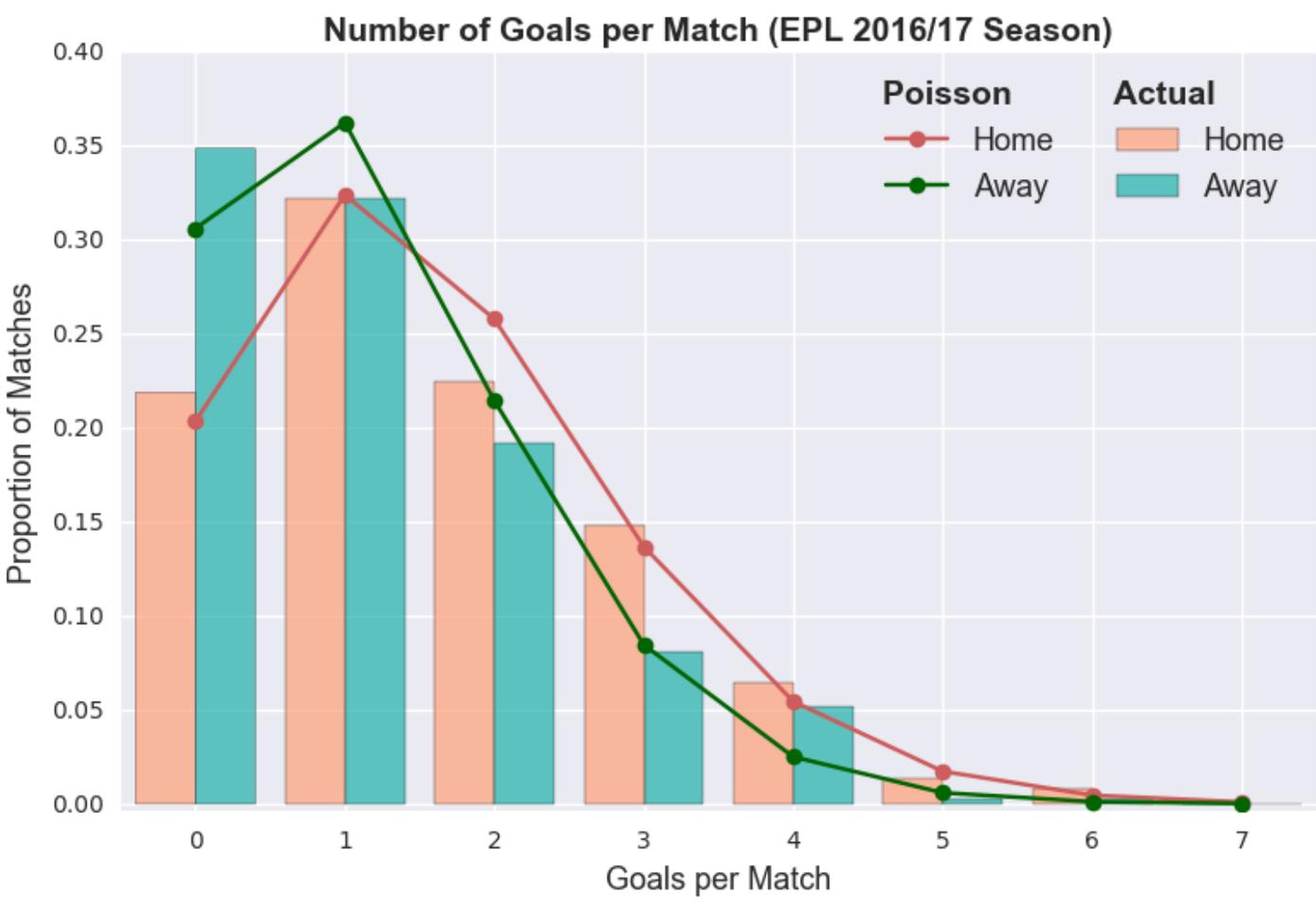
```
epl_1617 = epl_1617[:-10]
epl_1617.mean()
```

```
HomeGoals    1.591892
AwayGoals    1.183784
dtype: float64
```

You'll notice that, on average, the home team scores more goals than the away team. This is the so called 'home (field) advantage' (discussed here) and isn't specific to soccer. This is a convenient time to introduce the Poisson distribution. It's a discrete probability distribution that describes the probability of the number of events within a specific time period (e.g 90 mins) with a known average rate of occurrence. A key assumption is that the number of events is independent of time. In our context, this means that goals don't become more/less likely by the number of goals already scored in the match. Instead, the number of goals is expressed purely as function an average rate of goals. If that was unclear, maybe this mathematical formulation will make clearer:

$$P(x) = \frac{e^{-\lambda}\lambda^{x}}{x!}, \lambda > 0$$

λ
represents the average rate (e.g. average number of goals, average number of letters you receive, etc.). So, we can treat the number of goals scored by the home and away team as two independent Poisson distributions. The plot below shows the proportion of goals scored compared to the number of goals estimated by the corresponding Poisson distributions.



We can use this statistical model to estimate the probability of specfic events.

$$\begin{aligned} P(\geq 2|Home) &= P(2|Home) + P(3|Home) + \ldots \\ &= 0.258 + 0.137 + \ldots \\ &= 0.47 \end{aligned}$$

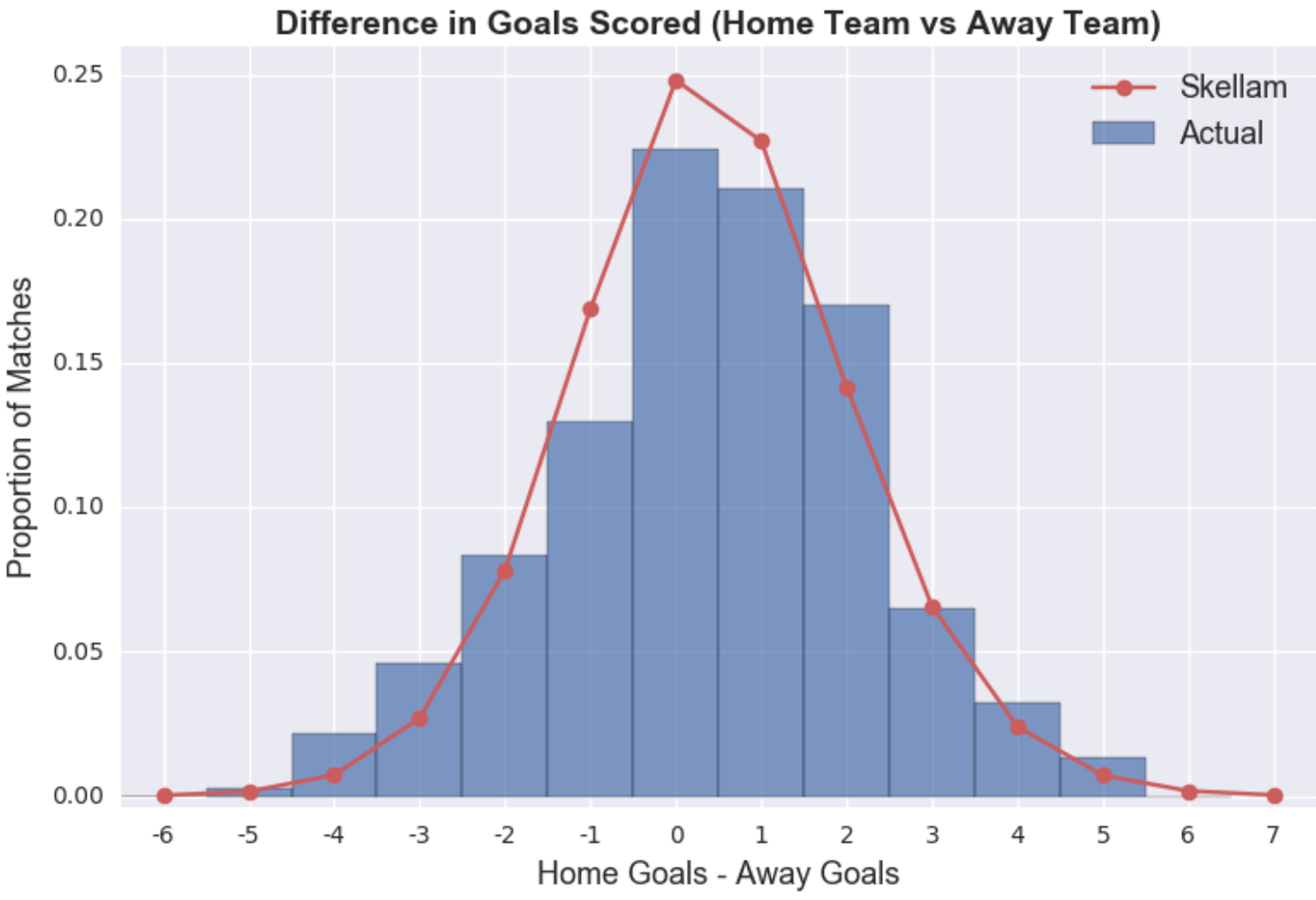
The probability of a draw is simply the sum of the events where the two teams score the same amount of goals.

$$\begin{aligned} P(Draw) &= P(0|Home) \times P(0|Away) + P(1|Home) \times P(1|Away) + \ldots \\ &= 0.203 \times 0.306 + 0.324 \times 0.362 + \ldots \\ &= 0.248 \end{aligned}$$

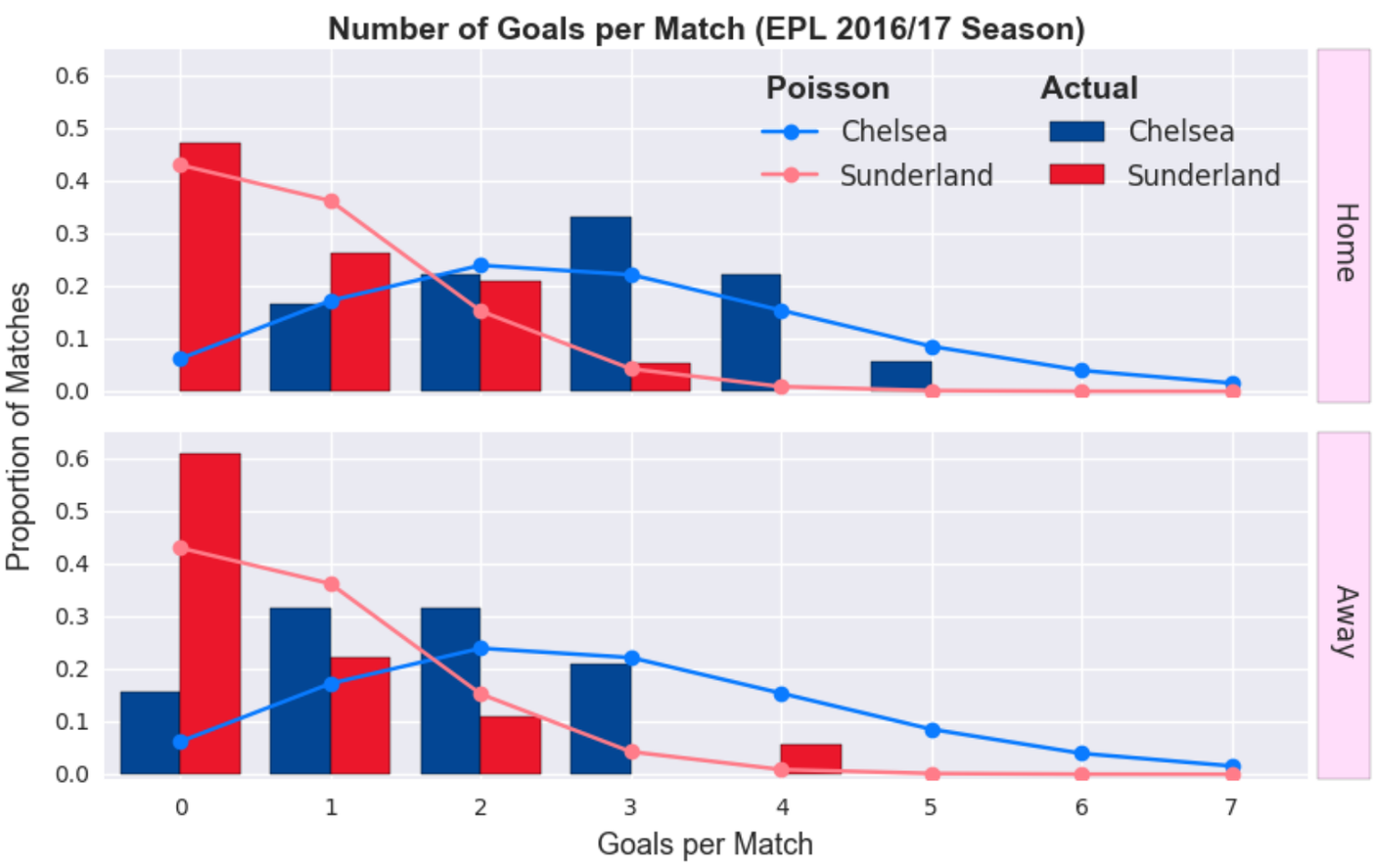
Note that we consider the number of goals scored by each team to be independent events (i.e. P(A n B) = P(A) P(B)). The difference of two Poisson distribution is actually called a Skellam distribution. So we can calculate the probability of a draw by inputting the mean goal values into this distribution.

```
# probability of draw between home and away team
skellam.pmf(0.0,  epl_1617.mean()[0],  epl_1617.mean()[1])

# probability of home team winning by one goal
skellam.pmf(1,  epl_1617.mean()[0],  epl_1617.mean()[1])
```



So, hopefully you can see how we can adapt this approach to model specific matches. We just need to know the average number of goals scored by each team and feed this data into a Poisson model. Let's have a look at the distribution of goals scored by Chelsea and Sunderland (teams who finished 1st and last, respectively).



Building A Model

You should now be convinced that the number of goals scored by each team can be approximated by a Poisson distribution. Due to a relatively sample size (each team plays at most 19 home/away games), the accuracy of this approximation can vary significantly (especially earlier in the season when teams have played fewer games). Similar to before, we could now calculate the probability of various events in this Chelsea Sunderland match. But rather than treat each match separately, we'll build a more general Poisson regression model (what is that?).

```
# importing the tools required for the Poisson regression model
import statsmodels.api as sm
import statsmodels.formula.api as smf

goal_model_data = pd.concat([epl_1617[['HomeTeam','AwayTeam','HomeGoal
                  columns={'HomeTeam':'team', 'AwayTeam':'opponent','HomeGoa
                 epl_1617[['AwayTeam','HomeTeam','AwayGoals']].assign(home=0
                  columns={'AwayTeam':'team', 'HomeTeam':'opponent','AwayGoa
```

```
poisson_model = smf.glm(formula="goals ~ home + team + opponent", data
                        family=sm.families.Poisson()).fit()
poisson_model.summary()
```

Generalized Linear Model Regression Results

Dep. Variable:	goals	**No. Observations:**	740
Model:	GLM	**Df Residuals:**	700
Model Family:	Poisson	**Df Model:**	39
Link Function:	log	**Scale:**	1.0
Method:	IRLS	**Log-Likelihood:**	-1042.4
Date:	Sat, 10 Jun 2017	**Deviance:**	776.11
Time:	11:17:38	**Pearson chi2:**	659.
No. Iterations:	8		

	coef	std err	z	P>|z|	[95 Co In
Intercept	0.3725	0.198	1.880	0.060	-0.(0.76
team[T.Bournemouth]	-0.2891	0.179	-1.612	0.107	-0.6 0.0(
team[T.Burnley]	-0.6458	0.200	-3.230	0.001	-1.0 -0.2
team[T.Chelsea]	0.0789	0.162	0.488	0.626	-0.2 0.3
team[T.Crystal Palace]	-0.3865	0.183	-2.107	0.035	-0.7 -0.(
team[T.Everton]	-0.2008	0.173	-1.161	0.246	-0.5 0.13
team[T.Hull]	-0.7006	0.204	-3.441	0.001	-1.1 -0.3
team[T.Leicester]	-0.4204	0.187	-2.249	0.025	-0.7 -0.(
team[T.Liverpool]	0.0162	0.164	0.099	0.921	-0.3 0.33

team[T.Man City]	0.0117	0.164	0.072	0.943	-0.3 0.3
team[T.Man United]	-0.3572	0.181	-1.971	0.049	-0.7 -0.(
team[T.Middlesbrough]	-1.0087	0.225	-4.481	0.000	-1.4 -0.5
team[T.Southampton]	-0.5804	0.195	-2.976	0.003	-0.9 -0.1
team[T.Stoke]	-0.6082	0.197	-3.094	0.002	-0.9 -0.2
team[T.Sunderland]	-0.9619	0.222	-4.329	0.000	-1.3 -0.5
team[T.Swansea]	-0.5136	0.192	-2.673	0.008	-0.8 -0.1
team[T.Tottenham]	0.0532	0.162	0.328	0.743	-0.2 0.3
team[T.Watford]	-0.5969	0.197	-3.035	0.002	-0.9 -0.2
team[T.West Brom]	-0.5567	0.194	-2.876	0.004	-0.9 -0.1
team[T.West Ham]	-0.4802	0.189	-2.535	0.011	-0.8 -0.1
opponent[T.Bournemouth]	0.4109	0.196	2.092	0.036	0.0 0.7
opponent[T.Burnley]	0.1657	0.206	0.806	0.420	-0.2 0.5(
opponent[T.Chelsea]	-0.3036	0.234	-1.298	0.194	-0.7 0.1
opponent[T.Crystal Palace]	0.3287	0.200	1.647	0.100	-0.(0.7
opponent[T.Everton]	-0.0442	0.218	-0.202	0.840	-0.4 0.3
opponent[T.Hull]	0.4979	0.193	2.585	0.010	0.1 0.8
opponent[T.Leicester]	0.3369	0.199	1.694	0.090	-0.(0.7

opponent[T.Liverpool]	-0.0374	0.217	-0.172	0.863	-0.4 0.3
opponent[T.Man City]	-0.0993	0.222	-0.448	0.654	-0.5 0.3
opponent[T.Man United]	-0.4220	0.241	-1.754	0.079	-0.8 0.0
opponent[T.Middlesbrough]	0.1196	0.208	0.574	0.566	-0.2 0.5
opponent[T.Southampton]	0.0458	0.211	0.217	0.828	-0.3 0.4
opponent[T.Stoke]	0.2266	0.203	1.115	0.265	-0.1 0.6
opponent[T.Sunderland]	0.3707	0.198	1.876	0.061	-0.0 0.7
opponent[T.Swansea]	0.4336	0.195	2.227	0.026	0.0 0.8
opponent[T.Tottenham]	-0.5431	0.252	-2.156	0.031	-1.0 -0.0
opponent[T.Watford]	0.3533	0.198	1.782	0.075	-0.0 0.7
opponent[T.West Brom]	0.0970	0.209	0.463	0.643	-0.3 0.5
opponent[T.West Ham]	0.3485	0.198	1.758	0.079	-0.0 0.7
home	0.2969	0.063	4.702	0.000	0.1 0.4

If you're curious about the `smf.glm(...)` part, you can find more information here (edit: earlier versions of this post had erroneously employed a Generalised Estimating Equation (GEE)- what's the difference?). I'm more interested in the values presented in the `coef` column in the model summary table, which are analogous to the slopes in linear regression. Similar to logistic regression, we take the exponent of the parameter values. A positive value implies more goals (

$e^x > 1 \forall x > 0$
), while values closer to zero represent more neutral effects (
$e^0 = 1$
). Towards the bottom of the table you might notice that `home` has a `coef` of 0.2969. This captures the fact that home teams generally score more goals than the away team (specifically,
$e^{0.2969}$
=1.35 times more likely). But not all teams are created equal. Chelsea has a `coef` of 0.0789, while the corresponding value for Sunderland is -0.9619 (sort of saying Chelsea (Sunderland) are better (much worse!) scorers than average). Finally, the `opponent*` values penalize/reward teams based on the quality of the opposition. This relfects the defensive strength of each team (Chelsea: -0.3036; Sunderland: 0.3707). In other words, you're less likely to score against Chelsea. Hopefully, that all makes both statistical and intuitive sense.

Let's start making some predictions for the upcoming matches. We simply pass our teams into `poisson_model` and it'll return the expected average number of goals for that team (we need to run it twice- we calculate the expected average number of goals for each team separately). So let's see how many goals we expect Chelsea and Sunderland to score.

```
poisson_model.predict(pd.DataFrame(data={'team': 'Chelsea', 'opponent'
                                         'home':1},index=[1]))
```

```
poisson_model.predict(pd.DataFrame(data={'team': 'Sunderland', 'oppone
                                         'home':0},index=[1]))
```

Just like before, we have two Poisson distributions. From this, we can calculate the probability of various events. I'll wrap this in a `simulate_match` function.

```
def simulate_match(foot_model, homeTeam, awayTeam, max_goals=10):
    home_goals_avg = foot_model.predict(pd.DataFrame(data={'team': hom
                                                           'opponent'
                                                  index=[1])).valu
    away_goals_avg = foot_model.predict(pd.DataFrame(data={'team': awa
                                                           'opponent'
                                                  index=[1])).valu
    team_pred = [[poisson.pmf(i, team_avg) for i in range(0, max_goals
    return(np.outer(np.array(team_pred[0]), np.array(team_pred[1])))
simulate_match(poisson_model, 'Chelsea', 'Sunderland', max_goals=3)
```

```
array([[ 0.03108485,  0.01272529,  0.00260469,  0.00035543],
       [ 0.0951713 ,  0.03896054,  0.00797469,  0.00108821],
       [ 0.14569118,  0.059642  ,  0.01220791,  0.00166586],
       [ 0.14868571,  0.06086788,  0.01245883,  0.0017001 ]])
```

This matrix simply shows the probability of Chelsea (rows of the matrix) and Sunderland (matrix columns) scoring a specific number of goals. For example, along the diagonal, both teams score the same the number of goals (e.g. P(0-0)=0.031). So, you can calculate the odds of draw by summing all the diagonal entries. Everything below the diagonal represents a Chelsea victory (e.g P(3-0)=0.149). If you prefer Over/Under markets, you can estimate P(Under 2.5 goals) by summing the entries where the sum of the column number and row number (both starting at zero) is less than 3 (i.e. the 6 values that form the upper left triangle). Luckily, we can use basic matrix manipulation functions to perform these calculations.

```
chel_sun = simulate_match(poisson_model, "Chelsea", "Sunderland", max_
# chelsea win
np.sum(np.tril(chel_sun, -1))
```

```
# draw
```

```
np.sum(np.diag(chel_sun))


# sunderland win
np.sum(np.triu(chel_sun, 1))
```

Hmm, our model gives Sunderland a 2.7% chance of winning. But is that right? To assess the accuracy of the predictions, we'll compare the probabilities returned by our model against the odds offered by the Betfair exchange.

Sports Betting/Trading

Unlike traditional bookmakers, on betting exchanges (and Betfair isn't the only one- it's just the biggest), you bet against other people (with Betfair taking a commission on winnings). It acts as a sort of stock market for sports events. And, like a stock market, due to the efficient market hypothesis, the prices available at Betfair reflect the true price/odds of those events happening (in theory anyway). Below, I've posted a screenshot of the Betfair exchange on Sunday 21st May (a few hours before those matches started).

Sun 21 May			Multiples						
Coming up			1		X		2		
Today 15:00	Arsenal Everton	Matched: GBP 502,864	1.4 £2633	1.41 £584	5.7 £611	5.8 £132	8.6 £351	8.8 £440	i
Today 15:00	Burnley West Ham	Matched: GBP 127,063	2.38 £119	2.4 £84	3.6 £245	3.65 £2403	3.25 £1601	3.3 £473	i
Today 15:00	Chelsea Sunderland	Matched: GBP 312,295	1.13 £60703	1.14 £11594	11.5 £334	12 £221	29 £284	30 £23	i
Today 15:00	Hull Tottenham	Matched: GBP 274,460	9.2 £40	9.4 £784	5.8 £790	5.9 £642	1.39 £2224	1.4 £5137	i
Today 15:00	Leicester Bournemouth	Matched: GBP 216,678	1.87 £320	1.89 £170	4.1 £951	4.3 £1557	4.3 £279	4.4 £434	i
Today 15:00	Liverpool Middlesbrough	Matched: GBP 857,747	1.14 £176274	1.15 £56345	10.5 £2988	11 £2701	28 £1807	29 £1581	i
Today 15:00	Man Utd C Palace	Matched: GBP 684,142	2.4 £2992	2.42 £9765	3.45 £4522	3.5 £658	3.35 £696	3.4 £1223	i
Today 15:00	Southampton Stoke	Matched: GBP 100,032	1.75 £448	1.76 £2566	4.1 £1079	4.2 £49	5.2 £74	5.3 £217	i
Today 15:00	Swansea West Brom	Matched: GBP 118,771	2.32 £7	2.34 £825	3.5 £1942	3.55 £304	3.45 £612	3.5 £76	i
Today 15:00	Watford Man City	Matched: GBP 589,601	19.5 £27	20 £101	9.8 £203	10 £262	1.17 £32750	1.18 £3609	i

The numbers inside the boxes represent the best available prices and the amount available at those prices. The blue boxes signify back bets (i.e. betting that an event will happen- going long using stock market terminology), while the pink boxes represent lay bets (i.e. betting that something won't happen- i.e. shorting). For example, if we were to bet £100 on Chelsea to win, we would receive the original amount plus 100*1.13= £13 should they win (of course, we would lose our £100 if they didn't win). Now, how can we compare these prices to the probabilities returned by our model? Well, decimal odds can be converted to the probabilities quite easily: it's simply the inverse of the decimal odds. For example, the implied probability of Chelsea winning is 1/1.13 (=0.885- our model put the probability at 0.889). I'm focusing on decimal odds, but you might also be familiar with Moneyline (American) Odds (e.g. +200) and fractional odds (e.g. 2/1). The relationship between decimal odds, moneyline and probability is

illustrated in the table below. I'll stick with decimal odds because the alternatives are either unfamiliar to me (Moneyline) or just stupid (fractional odds).

Convert to Decimal Odds

Chance of Occurence (EPL Fixtures 21st May 2017)

Source: Betfair Exchange

Match	Home	Draw	Away
Arsenal v Everton	71.4 %	17.5 %	11.6 %
Burnley v West Ham	42 %	27.8 %	30.8 %
Chelsea v Sunderland	88.5 %	8.7 %	3.4 %
Hull v Tottenham	10.9 %	17.2 %	71.9 %
Leicester v Bournemouth	53.5 %	24.4 %	23.3 %
Liverpool v Middlesbrough	87.7 %	9.5 %	3.6 %
Man Utd v C Palace	41.7 %	29 %	29.9 %
Southampton v Stoke	57.1 %	24.4 %	19.2 %
Swansea v West Brom	43.1 %	28.6 %	29 %
Watford v Man City	5.1 %	10.2 %	85.5 %

So, we have our model probabilities and (if we trust the exchange) we know the true probabilities of each event happening. Ideally, our model would identify situations the market has underestimated the chances of an event occurring (or not occurring in the case of lay bets). For example, in a simple coin toss game, imagine if you were offered $2 for every $1 wagered (plus your stake), if you guessed correctly. The implied probability is 0.333, but any valid model would return a probability of 0.5. The odds returned by our model and the Betfair exchange are compared in the table below.

Match		Home	Draw	Away
Arsenal v Everton	Betfair	0.714	0.175	0.116
	Predicted	0.533	0.226	0.241
	Difference	0.181	0.051	0.125

Burnley v West Ham	Betfair	0.42	0.278	0.308
	Predicted	0.461	0.263	0.276
	Difference	0.041	0.015	0.032
Chelsea v Sunderland	Betfair	0.885	0.087	0.034
	Predicted	0.889	0.084	0.027
	Difference	0.004	0.003	0.007
Hull v Tottenham	Betfair	0.109	0.172	0.719
	Predicted	0.063	0.138	0.799
	Difference	0.046	0.034	0.08
Leicester v Bournemouth	Betfair	0.535	0.244	0.233
	Predicted	0.475	0.22	0.306
	Difference	0.06	0.024	0.073
Liverpool v Middlesbrough	Betfair	0.877	0.095	0.036
	Predicted	0.77	0.161	0.069
	Difference	0.107	0.066	0.033
Man Utd v C Palace	Betfair	0.417	0.29	0.299
	Predicted	0.672	0.209	0.119
	Difference	0.255	0.081	0.18
Southampton v Stoke	Betfair	0.571	0.244	0.192
	Predicted	0.496	0.277	0.226
	Difference	0.075	0.033	0.034
Swansea v West Brom	Betfair	0.431	0.286	0.29
	Predicted	0.368	0.266	0.366
	Difference	0.063	0.02	0.076
Watford v Man City	Betfair	0.051	0.102	0.855
	Predicted	0.167	0.203	0.631
	Difference	0.116	0.101	0.224

Green cells illustrate opportunities to make profitable bets, according to our model (the opacity of the cell is determined by the implied difference). I've highlighted the difference between the

model and Betfair in absolute terms (the relative difference may be more relevant for any trading strategy). Transparent cells indicate situations where the exchange and our model are in broad agreement. Strong colours imply that either our model is wrong or the exchange is wrong. Given the simplicity of our model, I'd lean towards the latter.

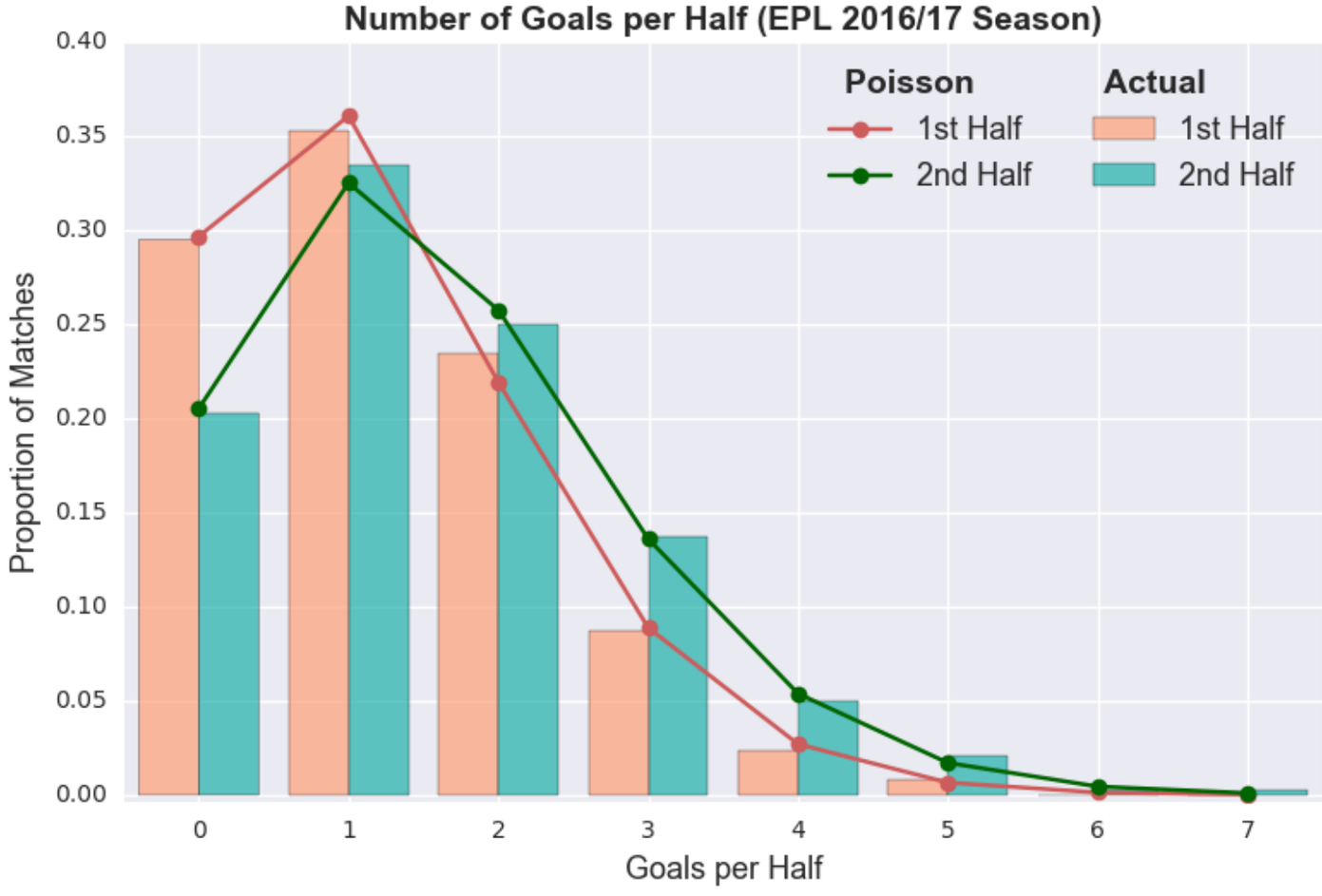
Something's Poissony

So should we bet the house on Manchester United? Probably not (though they did win!). There's some non-statistical reasons to resist backing them. Keen football fans would notice that these matches represent the final gameweek of the season. Most teams have very little to play for, meaning that the matches are less predictable (especially when they involve unmotivated 'bigger' teams). Compounding that, Man United were set to play Ajax in the Europa Final three days later. Man United manager, Jose Mourinho, had even confirmed that he would rest the first team, saving them for the much more important final. In a similar fashion, injuries/suspensions to key players, managerial sackings would render our model inaccurate. Never underestimate the importance of domain knowledge in statistical modelling/machine learning! We could also think of improvements to the model that would incorporate time when considering previous matches (i.e. more recent matches should be weighted more strongly).

Statistically speaking, is a Poisson distribution even appropriate? Our model was founded on the belief that the number goals can be accurately expressed as a Poisson distribution. If that assumption is misguided, then the model outputs will be unreliable. Given a Poisson distribution with mean
λ
, then the number of events in half that time period follows a Poisson distribution with mean
λ

/2. In football terms, according to our Poisson model, there should be an equal number of goals in the first and second halves. Unfortunately, that doesn't appear to hold true.

```
epl_1617_halves = pd.read_csv("http://www.football-data.co.uk/mmz4281/
epl_1617_halves = epl_1617_halves[['FTHG', 'FTAG', 'HTHG', 'HTAG']]
epl_1617_halves['FHgoals'] = epl_1617_halves['HTHG'] + epl_1617_halves
epl_1617_halves['SHgoals'] = epl_1617_halves['FTHG'] + epl_1617_halves
                                epl_1617_halves['FHgoals']
epl_1617_halves = epl_1617_halves[['FHgoals', 'SHgoals']]
```



We have irrefutable evidence that violates a fundamental assumption of our model, rendering this whole post as pointless as Sunderland!!! Or we can build on our crude first attempt. Rather than a simple univariate Poisson model, we might have more success with a bivariate Poisson distriubtion. The Weibull distribution has also been proposed as a viable alternative. These might be topics for future blog posts.

Summary

We built a simple Poisson model to predict the results of English Premier League matches. Despite its inherent flaws, it recreates several features that would be a necessity for any predictive football model (home advantage, varying offensive strengths and opposition quality). In conclusion, don’t wager the rent money, but it’s a good starting point for more sophisticated realistic models. Thanks for reading!