Data Visualization and Analysis

In order to use the datasets collected, we first needed to clean and reconfigure the data available. We extracted only the English Premier League data from the European Soccer Database and joined the various tables together using the id values present in each. Finally, we removed the corrupted data and added in the relevant data from the Datahub dataset. This step was not particularly straightforward, as there was no match_id link between datasets. Thus, we were forced to generate a new column in each dataset by concatenating the date of a match and the home & away teams involved. This then allowed us to combine the two datasets.

Another challenge we faced during the data processing stage was the FIFA attributes. We found out that the FIFA attributes for both players and teams changed with time, as updates were added to the game. As such, it was difficult to obtain the relevant attributes for each match across years 2009 - 2016. We resorted to using Python and the Pandas library to find the closest attribute set based on the date of each match. This gave us an estimate for the relative strength of the teams facing each other.

After cleaning the data, we began to think about how we would be able to validate our simulation. Thus, we decided to create an analytical predictive model to estimate the probability of each team winning a match. This would be used as a point of comparison with our simulation output. Using Python, we modelled each game as a Poisson process using data from 2009 to 2016, with goals arriving based on an exponential distribution. We also discovered some useful facts. The first is that historically, home teams score 0.4 more goals per match than away teams. The second is that there is approximately 45.8% chance of a home team winning a game. Thus, the prediction generated from both our analysis and simulation should have better performance than this base case.

Preliminary Analysis

```
In [4]: epl['home_team_goal'].mean()
        epl['away_team_goal'].mean()

Out[4]: 1.5725563909774436

Out[4]: 1.1710526315789473
```

On average, it seems that home teams score more goals than away teams!

```
In [11]: len(epl[epl['result'] == 'H'])
         len(epl[epl['result'] == 'A'])
         len(epl[epl['result'] == 'D'])
         len(epl[epl['result'] == 'H'])+len(epl[epl['result'] == 'A'])+len(epl[epl['result'] == 'D'])

Out[11]: 1217

Out[11]: 757

Out[11]: 686

Out[11]: 2660
```

There is about 45.8% chance of a home team winning a game! (Our prediction needs to be better than this base case)

The analytical model takes into account results from historical meetings between teams, as well as their relative strength. After running the model, we arrive at the model shown below:

Generalized Linear Model Regression Results

Dep. Variable:	goals	**No. Observations:**	5320
Model:	GLM	**Df Residuals:**	5254
Model Family:	Poisson	**Df Model:**	65
Link Function:	log	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-7686.2
Date:	Sat, 17 Nov 2018	**Deviance:**	5964.1
Time:	15:54:46	**Pearson chi2:**	5.18e+03
No. Iterations:	5	**Covariance Type:**	nonrobust

Using the model, we could then write a simple function to calculate scores and results of a match between any two teams.

```
def simulate_match(foot_model, homeTeam, awayTeam, max_goals=10):
    home_goals_avg = foot_model.predict(pd.DataFrame(data={'team': homeTeam,
                                                           'opponent': awayTeam,'home':1},
                                                     index=[1]))#.values[0]
    away_goals_avg = foot_model.predict(pd.DataFrame(data={'team': awayTeam,
                                                           'opponent': homeTeam,'home':0},
                                                     index=[1]))#.values[0]
    team_pred = [[poisson.pmf(i, team_avg) for i in range(0, max_goals+1)] for team_avg in [home_goals_avg, away_goals_avg]]
    results = np.outer(np.array(team_pred[0]), np.array(team_pred[1]))
    home_win = np.sum(np.tril(results, -1))
    away_win = np.sum(np.triu(results, 1))
    draw = np.sum(np.diag(results))
    return([home_win, draw, away_win])
```

```
simulate_match(poisson_model, "Man United", "Sunderland", max_goals=10)
```

```
[0.62987839936933954, 0.22841515265014634, 0.14170402366279922]
```

In the example above, we calculate the probability of either a Home Win, Draw or Away Win and return the values. It is also possible to obtain a matrix of score distributions by modifying the simulate_match function. In the simulation above, Man United has a 63% chance of winning the match versus Sunderland, which makes sense because Man United is considered a stronger team than Sunderland. The most likely score we obtain is 1-0, with 2-0 a close second.

We then tested the model on data from 2017.

```
In [30]: len(test[test['FTR']==test['predicted']])
         len(test[test['FTR']!=test['predicted']])
         len(test)

Out[30]: 180

Out[30]: 161

Out[30]: 341
```

There are 180 correct predictions and 161 incorrect predictions, which comes up to a 52.8% prediction success rate! Interestingly, many popular bookies also have approximately a 53% prediction success rate.

With an analytical model prepared and data in an easily interpretable and clean format, we realized that our simulation required much more granular information about a game (this is explained further in the Performance Measurement file). Thus, we turned to existing research and mathematical models to simulate the various states of the ball during a match.

All python code can be found in our Github repository:
https://github.com/ryanndelion/soccer-simulator.git