Abstract

Being a soccer fan myself, predicting the results of soccer matches poses a very interesting challenge to me and to most other soccer fans, pundits and bookmakers. However, just like stock-market prediction the problem is here that this is a non-deterministic problem. You cannot expect 100% accuracy in this problem as that would suggest that you can predict the future! During the development of my project, I tried out multiple machine learning techniques and decided on Adaboost as it gave the optimal results. The obtained results satisfactorily suggest that the 'form' of a team is a good indicator of the possible outcome of a match.

Introduction

Soccer is one of the most widely watched sports across the world and the English Premier League is one of the most competitive football leagues in the world with arguably the largest viewer-base worldwide. In this system I focus mainly on predictions and observations from this league.

Predicting soccer results is a 3-class problem since there are 3 possible outcomes for a match, Win, Loss or Draw. Thus, even if a random prediction is made for each match an accuracy of at least 33% is achievable. The problem can also be modeled as a 2-class problem by ignoring all matches with a 'Draw' result thus creating a 50% accuracy lower bound which is achievable by random guessing. Several papers that I went through approached this problem as a 2-class problem. I experimented with my system on both these types of problems and as expected it performed much better on the 2-class variation of the problem,

Dataset

A very exhaustive search was performed to look for a dataset that would be applicable to this problem. I was looking for one that would be freely available on the internet and found just one website that provided this;
http://www.football-data.co.uk/data.php
This website had datasets including fulltime and halftime results for up to 22 European league divisions from 19 seasons back to 1993/94. Also included for the major European football leagues were match statistics like shots on goal, corners, fouls, offsides, bookings, red card and referees. Betting odds for each match were also available from multiple betting websites.

Although I had data for 19 seasons, I only used the data for 5 seasons from the 2007/08 – 2011/12 as that provided a suitable and sufficient number of data-points to build the system upon.

The betting odds that are provided with each datapoint are used as 'Expert Prediction' labels so that I can evaluate my predictions with theirs. To convert the odds into predictions, I simply take the result with the lowest payoff and set that as the prediction.

Feature Selection

The crux of my time was spent on modeling features from my dataset that were suggestive of the result of a match. Most of the features that I tried were features that were based on the current form for that team. My definition of form was the performance of a team in the 3 most recent games. I chose the number 3 after experimenting with several other obvious numbers in the same range and figured out 3 to be giving the best results.

The features I experimented with were:

- Number of wins in last 3 home/away games
- Number of losses in last 3 home/away games
- Number of draws in last 3 home/away games
- Number of Wins/Losses/Draws in last 3 head-to-head encounters
- Number of Full Time goals in last 3 matches
- Number of Half Time goals in last 3 matches
- Number of Shots in last 3 matches
- Number of Shots on target in last 3 matches
- Number of Corners in last 3 matches
- Number of Fouls in last 3 matches
- Number of Yellow cards in last 3 matches
- Number of Red cards in last 3 matches
- Number of points current scored in the season

Additionally I also experimented with some out-of-the-box features like:

- Number of matches won/lost/drawn with the same referee
- Number of kilometers traveled by the away team

The motivation behind these features were as follows;
Number of matches won/lost/drawn with the same referee: It is a known fact to any English Premier League fan that the referees are usually biased to some teams (usually the big4) and by this feature I was hoping to capture this knowledge. However, this I found this feature to be too noisy as it is applicable to only some games and so I eventually ended up discarding it.

Number of kilometers traveled by the away team: This was an interesting feature that I read about in a technical paper and tried implementing it too. However, this feature is more applicable when using a dataset from a competition like the UEFA Champions league where teams travel to different countries with different weather conditions. The home team advantage is nicely incorporated into the system via this feature. However, since my dataset involves only teams in the same country, this is not as applicable as teams travel only across cities which aren't very different or far off from each other.

Structuring the Features and Datapoints

A major challenge at the beginning of this project was how to model this problem as a set of features and datapoints given the data that we have. The ideal method of prediction of a result between two teams playing any sport would be to look at the head-to-head performances of those two teams against each other in the past and base a prediction off of that. However in my project where we are trying the predict results in matches in the English Premier League (or any international league for that matter), the problem is that two teams face each other only twice in any given season. Even if we take 5 seasons of data, we would have only 10 such data points which is too sparse for any machine learning algorithm. Additionally, if we go too far back in the past to retrieve more of such head-to-head encounters. Following are a couple of approaches I took to solve this problem.

One approach that I experimented with was modeling the features based on the performances of the two teams against an arbitrary Team C. For example, if I had to predict the result of a match between Arsenal and Man United, I would model the features as follows:

Arsenal vs ManCity (W/L/D)	ManU vs ManCity (W/L/D)	Arsenal vs Chelsea (W/L/D)	ManU vs Chelsea (W/L/D)	Arsenal vs Liverpool (W/L/D)	ManU vs Liverpool (W/L/D)

In this manner I could model several features for each match. However I chose only a subset of the teams since choosing all teams meant creating 3*40=120 features for the 40 different teams and 3 possible match outcomes I had in my dataset. This would be way too many features and would slow down my Adaboost significantly. Therefore I did a couple of dry-runs and observed which decision stump was being selected most often and noticed it was usually either the top teams or bottom teams that were selected. My intuition is that performances against top or bottom teams are more deterministic because you can be more sure of a loss or win against a top or bottom team respectively. Due to this intuition I ended up choosing the 6 of the top flight teams in the EPL (viz. Arsenal, Chelsea, Man City, Man United, Tottenham, Everton).

However, after implementing this I realized that firstly the number of features were still much and the accuracy wasn't all the impressive either. I then went ahead and modeled the problem as follows by creating form-based features for each of the available stats for a match datapoint.

Feature Modeling

My initial approach to modeling the above features to was create a feature for each of the two teams for each type of feature for e.g. for a Chelsea vs Arsenal game;

W/L/D in last 3 for Chelsea	W/L/D in last 3 for Arsenal	Shots in last 3 for Chelsea	Shots in last 3 for Arsenal	Goals in last 3 for Chelsea	Goals in last 3 for Arsenal	Corners in last 3 for Chelsea	Corners in last 3 for Arsenal

However instead of taking a feature type for both teams, I instead decided to take the difference in values of that feature for each of the teams and use that delta value as a feature instead. Therefore my number of features was reduced by 2 and the above example was transformed to:

Δ(W/L/D in last 3)	Δ(Shots in last 3)	Δ(Goals in last 3)	Δ(Corners in last 3)

where for example

Δ(W/L/D in last 3) = (W/L/D in last 3 for Chelsea) – (W/L/D in last 3 for Arsenal)

Another optimization I added was to take the difference of squares of the two values. i.e. for a feature *x* for teams *a* and *b* I calculated the Δ value as

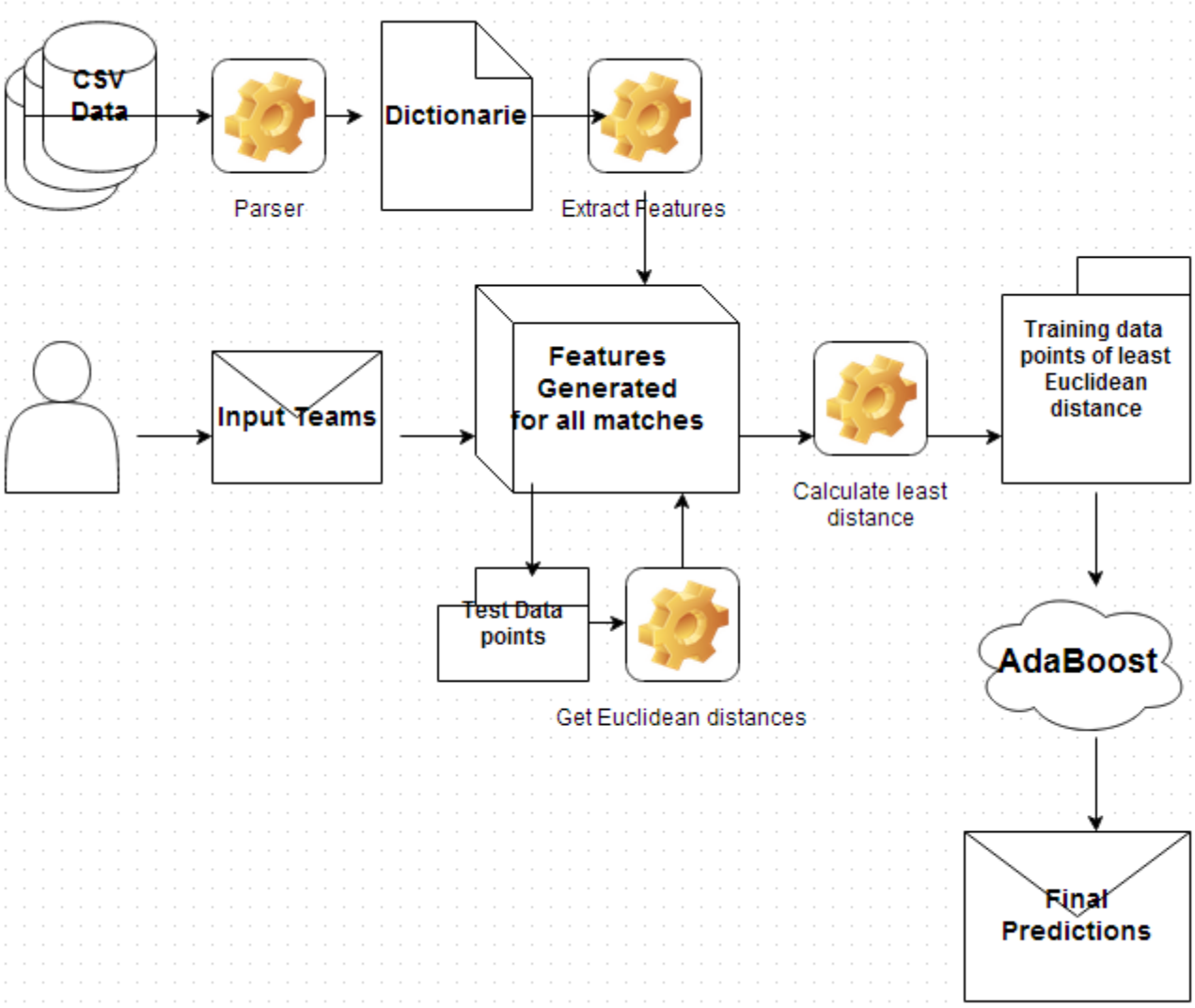
$$\Delta f^x = (f^x_a)^2 - (f^x_b)^2$$

I calculated a 5% boost in performance by trying this method. My intuition is that this happens because we are differentiating between closely matched values in the high range and the low range for a feature. For illustrating, consider the feature *Number of points currently scored in the season* in the following example;

Team A Points	Team B Points	$(f^x_a) - (f^x_b)$	$(f^x_a)^2 - (f^x_b)^2$
15	17	-2	-64
40	42	-2	-164

Thus this type of a delta helps in capturing the notion of a match being played by two top teams in the league versus two lower table teams. This is important because during the training phase of our algorithm, we choose only those matches with the similar feature values to train on. This sort of delta calculation helps in this situation.

System Design



Data Parsing

The dataset used was in csv format and so I parsed the dataset and organized each of the match statistics as hashes thus facilitating quick access. Also I made the match data available using the teams involved as keys thus facilitating quick access to all matches for a given team.

Calculate Feature Values

For every match in my dataset adhering to certain constraints the feature values were calculated and stored in serialized objects so that this process did not have to be repeated several times.

Following are some of the constraints I adhered to when calculating these feature values:

- Do not create datapoints for matches at the start of the season. This is done because, the form of a team would be then calculated based on their performance at the end of the previous season. This would be wrong because firstly, the matches that their form is being based off of occurred upto 3 – 5 months ago and matches that occur so long ago do not reflect form of a team. Also, these teams and their players may have changed significantly during this period of the transfer market.
- When calculating the form, only use the 3 most recent matches that the team played.
- There may be missing values contributed by teams that are newly promoted or relegated. I handled this by replacing this value by the mean value for that feature.

- When training the model, I do not consider points from the last season (i.e. 2007/08 season) as test points. It exists only as historical data to be used to calculate form-features for future matches.

Training the Model

Given 2 teams for which I want to make a prediction for their matches, I first go through all matches in all seasons except the last (2007/08) season and extract head-to-head matches between these two teams and use these datapoints as testing points.

Now, we need the training data points for these testing points. For this, we want to use only those matches that are most similar to the matches that we are predicting for. i.e. We want to train our model with pairs of teams that have similar form and other features to our test datapoints.

To implement this 'similarity' my first approach was to simply consider as training data only those matches in which the Home Team and Away Team from the test datapoint.
The system performed fairly well with setup however I tried the following method which gave much more accurate results.

To calculate the similarity between datapoints I instead used Euclidean Distance.

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

I averaged the values of the features across all test points and created an average-test-point. Then using this average-test-point I calculated the Euclidean distance to every datapoint that we have. The top 400 datapoints with the lowest Euclidean distances are then selected as training datapoints. I chose the number 400 after trying out several numbers between 100-500 and figured out that we need at least 300 points to train on to get good results. I chose 400 and not more firstly because as we keep taking in more data points their Euclidean distances from the testpoints increase and secondly because it would cause Adaboost to take much longer to produce results.

Weka

After feature creation I initially tried out the different combinations of features with all the different types of Machine Learning options with Weka. Of all the options Adaboost worked best however it still performed only around 50% accurate for the 3 class problem. So I went ahead and made my own implementation of Adaboost which did much better than Weka's.

AdaBoost

Adaboost was implemented using a Binary Decision Stump as a weak learner. However, since ours is a 3-class problem, a Multi Class AdaBoost was implemented. Predictions were made for all 3 possible outcomes and then the outcome for a match with highest confidence was selected as the final outcome.

In my implementation:

- Each of the labels ['H'. 'A', 'D'] for 'Home', 'Away', 'Draw' are selected in a loop and predictions are made with each.
- Y is then set as -1 or +1 depending on whether it matches the label selected for that iteration.

- For each datapoint, D is then initialized to 1/(number of training points)
- Now, we apply our weak classifier and try to obtain the best split which gives the least error.
- Here error is the sum of D for datapoints that do not match label.
- We then update our distribution by updating alpha and z
  $\alpha = 0.5 (\log (1 - \text{error}) / \text{error})$
  $z = 2*\sqrt{(\text{error}*(1-\text{error}))}$
- Then we update our distribution D with

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{\sum_i D_t(i)\exp(-\alpha_t y_i h_t(x_i))},$$

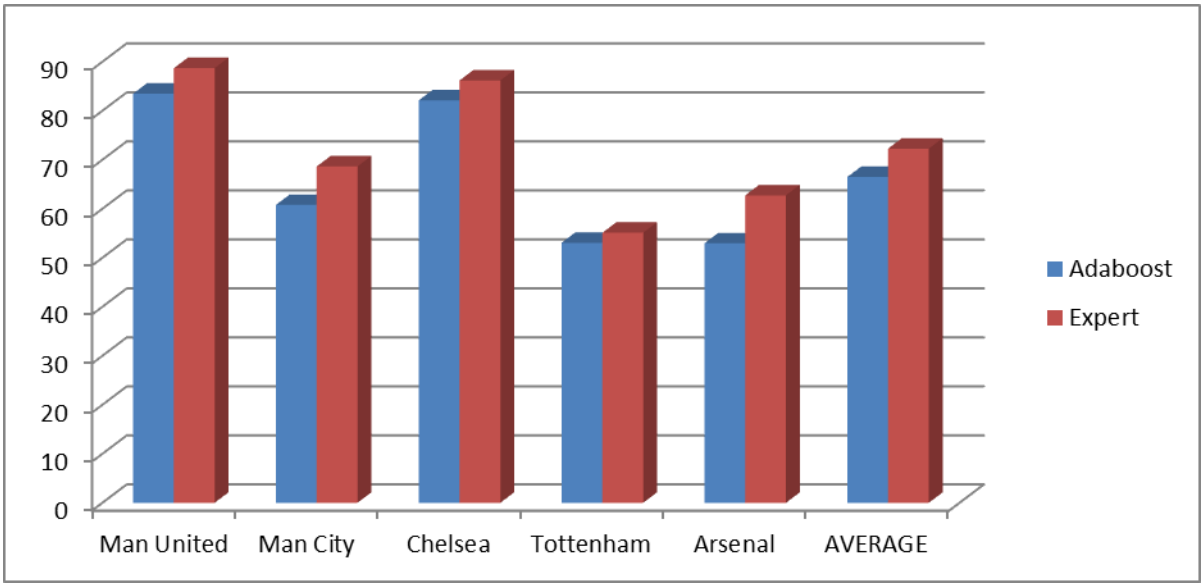
- This process is then repeated until we reach a stopping condition.

I used the following stopping conditions:

- Error reaches 0.5 (Converged)
- 50 iterations have occurred
- Change in error between current and previous iteration was 0.0000001

After getting predictions of all 3 labels we select label with the max(α) as our final predictions.

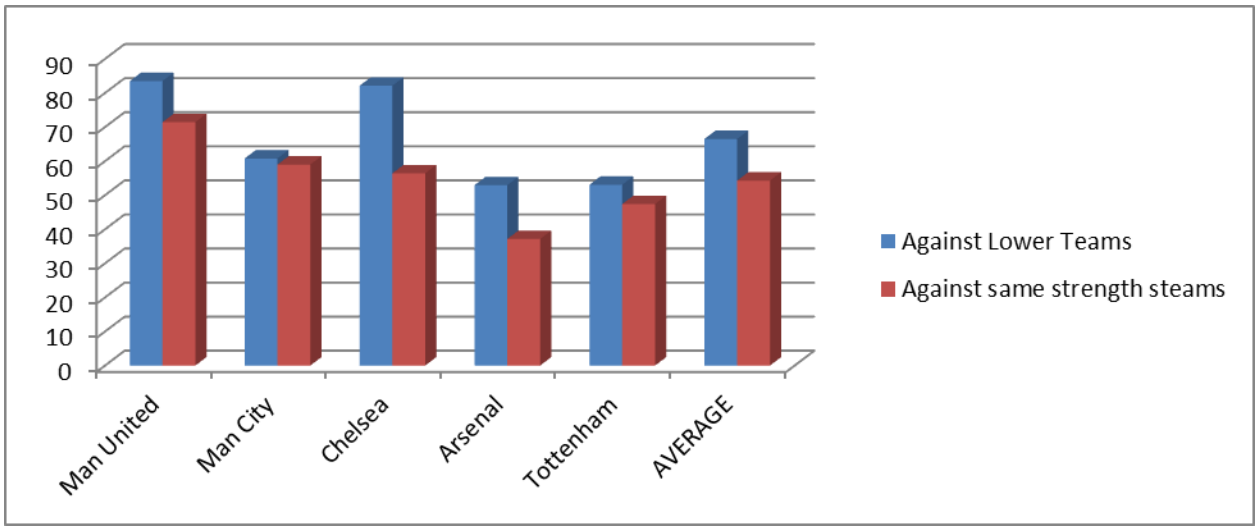
Results and Observations



(a) Prediction Accuracy for matches of strong teams against weak teams

From figure (a) we observe that the accuracy for predicting outcomes between strong teams against weak teams is fairly high with an average value of **66.47 %.**
My intuition is that these results are much more deterministic or predictable and as a result our system also does a much better job at predicting the results accurately. For example, if you look at the predictions for Manchester United, the accuracy is 83.47%. This makes sense since Manchester United (20-Time Champions) are far more consistent than other teams in the league.

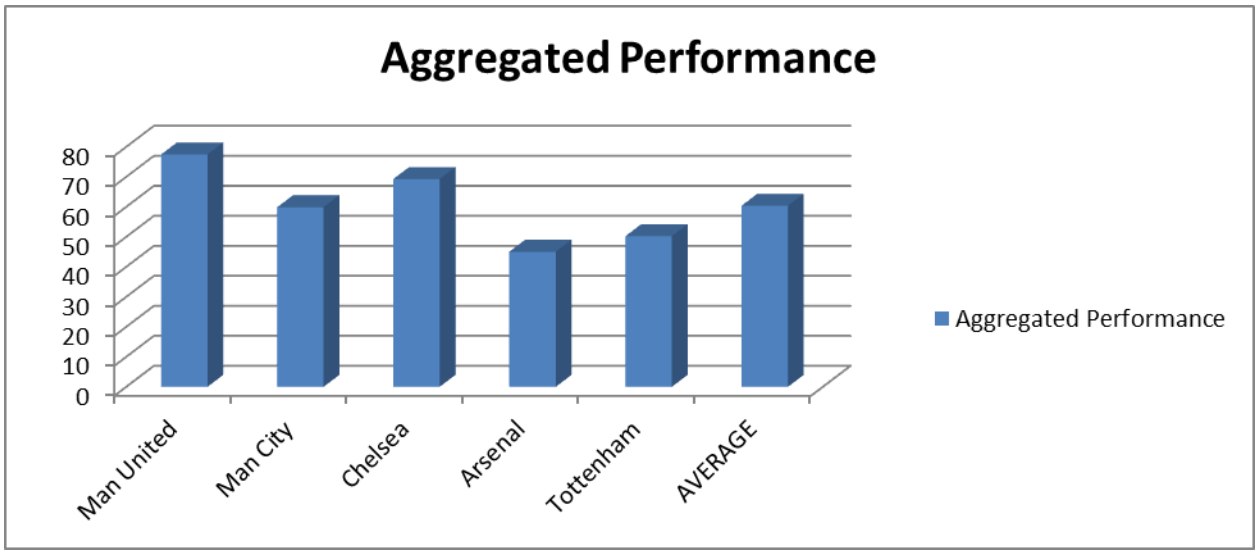
The red bars indicate the accuracy of predictions made by the Betting agencies and you can see that our system is almost on par with their predictions.



(b) Prediction Accuracy for matches of same strength teams plotted against data from figure *(a)*

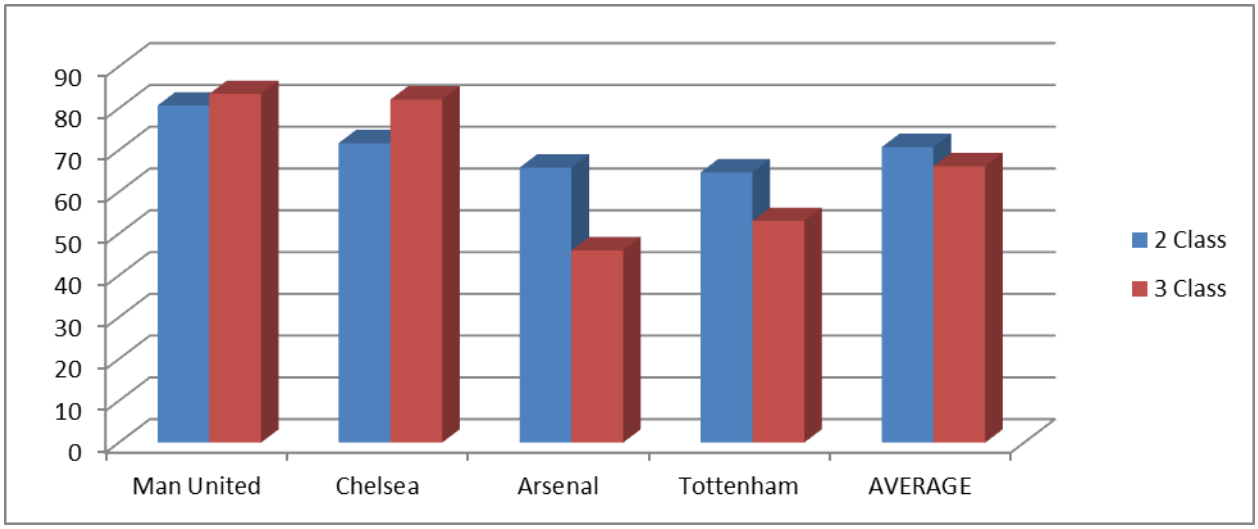
As expected when we make predictions for matches of same strength teams the accuracy drops to an average of 54.27%. This is because these matches are much more non-deterministic and unpredictable.

*Note: For the purpose of these observations, 'strong teams' are Manchester United, Manchester City, Chelsea, Arsenal and Tottenham. And 'weak teams' are Wigan, West Brom, West Ham, Bolton, Wolves, Sunderland, Stoke, Reading, Portsmouth, Blackburn, Hull, and Blackpool.



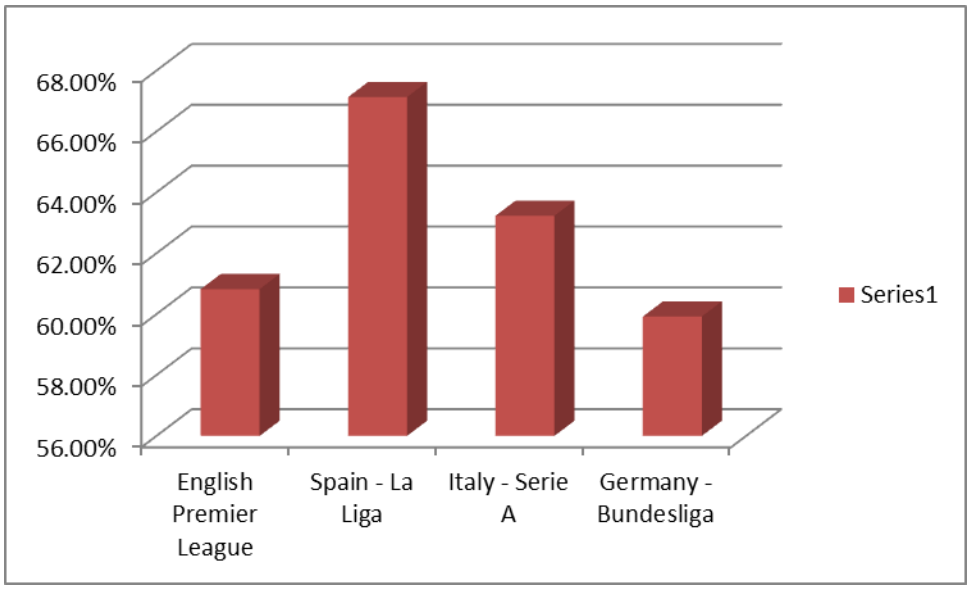
(c) Aggregated Prediction accuraces for each team and the whole system.

On aggregating all these values we get the overall system performance of **60.37%.**



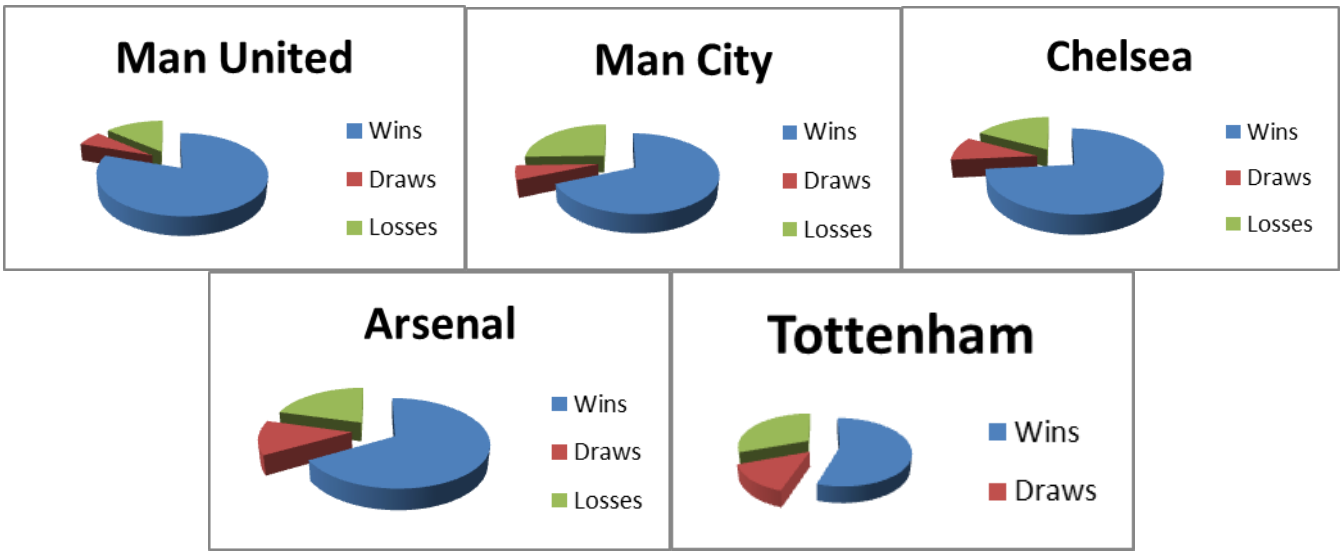
(d) 3-class vs 2-class results for top teams against weak teams

The above diagram highlights the fact that by reducing our problem to just 2 classes of Home Win or Away Win and by ignoring Drawn matches, we can achieve much higher accuracy as this is a much easier problem to solve. We go from 66.16% for the 3-class problem to 70.74% for the 2-class problem.



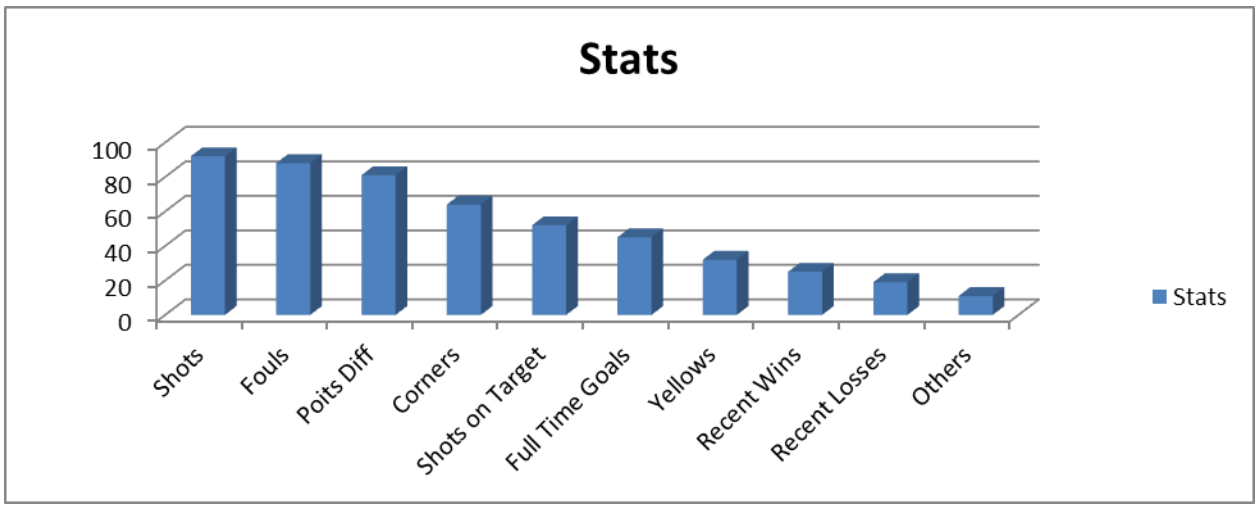
(e) Comparison of aggregate accuracy in different international leagues

On comparing the aggregate accuracy in different international leagues you notice that the La Liga does far better than the EPL or Bundesliga. My intuition is that this is because the EPL league is much more unpredictable and balanced than it's Spanish counterpart. As we noticed from fig. (a), oft times the top teams are defeated by lower table teams. However in the Spanish league, Real Madrid and Barcelona have consistently been topping the tables for several years, staying for ahead of the rest of the pack. As a result, match results are much more predictable.



(f) Pie Char of number of wins of the top English teams

The reason behind this figure is to illustrate the fact of how the top teams consistently keep producing the same results whereas mid-table teams like Tottenham aren't as consistent with their results. The larger the difference in size of the slices, the greater the consistency in results. For example, Man United and Chelsea have large slices of Wins and tiny slices of Draws and losses however Arsenal and Tottenham have Losses/Draws slices that are comparatively larger. Thus by comparing with fig. (a) you can see that the predictability of a team is directly proportional to how often it can keep producing the same result.



(g) Selection of features as decision stumps in AdaBoost

This indicates that 'Shots', 'Fouls' and 'Difference in points' were the best classifiers for the problem and this makes sense intuitively.Features like Recent Red cards, Draws and Half Time goals were less efficient classifiers.

Possible Improvements

Following are some improvements that can be made to this system that I believe would significantly boost the prediction accuracy

Player Information – Possessing individual player information would be very valuable in predicting the outcome of a match. Just by looking at the starting line-ups of two teams a sound prediction of the result can be made based on which regular-starters are missing, players absent due to injuries, yellow cards etc. Additionally information like the form of key-players could be instrumental in tipping the scale towards a more accurate prediction.

Social Event Data – Scouring Tweets or Facebook statuses for data about a particular match could provide useful insight into the possible outcome of that match. These tweets can be processed to extract the sentiment and the team involved and accordingly our machine learning algorithm could process this information and make predictions based on sentiment analysis of tweets before match-day.

References

[1] A. Joseph and N.E. Fenton and M. Neil, “Predicting football results using Bayesian nets and other machine learning techniques ” Jelsevier B.V., 2006

[2] Josip Hucaljuk and Alen Rakipovic, “Predicting football scores using machine learning techniques” MIPRO 2911

[3] Jack David Blundell, “Numerical Algorithms for Predicting Sports Results”

[4] Jim Warner, “Predicting Margin of Victory in NFL Games”

[5] Matthew Tucker, “Football Match Result Predictor Website”

[6] Kou-Yuan Huang, “A Neural Network Method for Prediction of 2006 World Cup Football Game”

[7] H. Rue and O. Salvesen, “Prediction and retrospective analysis of soccer matches in a league,” 1997

[8] Wikipedia – AdaBoost - http://en.wikipedia.org/wiki/AdaBoost

[9] Weka Docs - http://www.cs.waikato.ac.nz/ml/weka/documentation.html