Predicting English Premier League Winners using Machine Learning

Divyesh Harit , Rishi Mody
Univeristy of Massachusetts
dharit@cs.umass.edu ,rmody@cs.umass.edu

1 Introduction

Forecasting results in sports has always been something a fan has enjoyed, and they come up with expected lineups, team form and various other statistics before the game to back their predictions. Similarly, betting in sports in the United Kingdom is legal, and for soccer in particular, has been carried out with a great deal of interest by fans and regular betters alike. If there was a way to analyze various statistics of a game and help make predictions on the outcome of the game, it would be highly beneficial not only to teams which could easily identify their weaknesses and improve on them propelling them to their goal of performing well in the season, but also passionate fans and betters. This entire analysis would be easy to use and understandable for even a layman.

Being huge soccer fans ourselves, we take on this challenge of trying to predict results in a soccer league as unpredictable as English Premier League, where a single kick can flip the game on its head and throw all predictions out of the window. For some context, consider this: BBC soccer expert Mark Lawrenson[1] has a prediction success rate of 0.52, a target we hope to achieve. This shows that his prediction is pretty much as accurate as the flip of the coin, that's soccer for you!

What's unique is we ourselves extract 100+ features that may affect the outcome of a game in their own small ways from a soccer statistics website, season by season, and compare predictions when given an increasing number of seasons' data. For example, we train on the 1st season and predict on the 2nd, train on these two and predict on the 3rd, so on, and finally train on 11 seasons and predict on 12th season, and see how much the data of one single season affects the predictions.

We find that increasing the seasons' training data does not correlate to an increase in accuracy of predicting league position. Random Forest outperforms all other models, and we manage to achieve a peak of 0.60 accuracy, which is higher than BBC soccer expert Mark Lawrenson's predictions. SelectFromModel feature selection gives better results than PCA Dimensionality Reduction. We also learn that some key features play more role than others in deciding the league position.

2 Related Work

Soccer prediction is a popular area of interest mostly among young enthusiastic students like us, and therefore most of the related work has been in the form of course projects, like ours. Previous studies include ones such as [2], which took pre-formatted data in CSV format for the English Premier League from a website. This data was of individual games for a number of seasons, with a relatively small number of features. They managed to match the BBC accuracy of 0.52 for their One vs All SGD model, which is impressive.

[3] did a very extensive study on EPL, obtaining player performance data from OPTA[4] and expert ratings data from WhoScored[5]. They then divided their dataset into 4 parts: each for goalkeepers, defenders, midfielders and attackers, followed by analysis of performance metrics that influence the match outcome the most, and finally predicting how much the expert ratings for previous player performances affects the outcome of the match.

[6] do not mention the source of their data set for EPL, and considered just 3 features: goals scored, corners and shots on target. They managed to obtain accuracy of 0.52 and 0.48 for 2011-12 and 2012-2013 seasons, respectively, and their highest accuracy is 0.66, which we believe can't be generalized, as they trained on simply one season.

[7] computed 2 sets of features using their own formulae: static and dynamic. Static features included form and concentration of the team as well as motivation of the players for a particular game, depending on whether it's a derby, how far along in the season it is, etc. Dynamic features include goal difference, score difference and match history. Form, concentration and history data was extracted from ChampionAt[8], and scores and positions data from StatTo[9]. In all, a small number of features (9, to be exact) were considered and just 2 simple models: KNN and Random Forest, were used.

[10] is an older study that attempted to create statistical models that could predict results of the Greek soccer league matches. They used data for the 1997-98 season, and the factors considered were: Offensive parameters of the scoring team, defending parameters of the team which conceded these goals, and the home ground effect. Using these, they created a Poisson log-linear model that allowed them to observed the the difference in offensive and defensive performances of each team depending on whether they played an away game or a home game.

3 Data Set

We first tried to obtain our data from a data aggregator WhoScored.com[5]. We tried contacting them as even though their data is copyright protected they allow others to use it for the purpose of analysis, but did not receive a response. We then obtained all our data from StatBunker[11], a soccer statistics website. We tried a bunch of methods to scrape from StatBunker, such as using python library Scrapy. In the end, we decided to use python library BeautifulSoup to perform website scraping and extract the required data for each season, as we found it to be quite efficient and clean to use.

Our core dataset consists of 11 seasons of the English Premier League, starting from 2003-04 till 2013-14. We have also extracted additional data for 2014-15. The data for this additional season would be used as test input for the final leg of our predictions, in which we will train our models on all 11 seasons from 2003-04 till 2013-14.

The data for each season comprises of a total of 103 features. The number of samples increase as the range of seasons increases. For example, for our first experiment, we take the 2003-04 season data, so it contains a typical EPL season data with 20 teams each, and a class label (described in detail in methodology). For our 2^{2nd} experiment, it includes 40 teams, 20 from 2003-04 season and 20 from 2004-05 season, and so on, ending up with 220 samples for our final experiment. The features include key factors that may influence a game, such as total goals scored and conceded, goals scored and conceded per game, how many games won after losing/drawing/winning at half time, extensive data about penalties, distance from and the way in which the goals were scored, bookings for offences (yellow and red cards), etc. Most of them are integers, with some, such as goals scored and conceded per game, are floats.

Data extraction, cleaning and merging took a significant amount of time and we faced several challenges:

1) **Variety of features:** These features were spread out over around 20 different StatBunker web pages. We had to input different URL keywords for each web page and a different season ID to extract one page at a time. We did this by creating two lists, one containing these URL keywords and one holding the season IDs, and sending request for each combination in Scrape.py.

2) **Lack of common structure:** The pages for these features were not uniform throughout in their HTML code. For example, some feature rows had to be extracted through <tr> tag, some through <td> and some through <tbody> and then parsed for each team. So we had to identify the underlying DOM for each page before trying to extract the information.

3) **Conversion of empty values, missing data:** Some features were recorded with '-' when they were 0, and some were 0 by default. We had to change each '-' to 0 so that there was uniformity and they could be learned properly by the models. Some other missing data here and there was inputed manually by cross-checking with the official EPL website.

4) **Merging of Tables:** Some features such as goal scored distance, clean sheets etc. were statistics denoted to each player. Since we concentrate only on team statistics, we converted the above tables from player stats to team stats using Sum.py. Once this was done, we merged 16 tables to form statistics of one season using Merge.py which required the use of this script as the individual tables were not in the same sorted order.

4 Methodology

The points of each team at the end of the season has been divided into the following appropriate classes. Points 0-10, 10-20 and 20-30 belong classes 1,2 and 3 respectively. Points 35-40, 40-45, 45-50, 50-55, 55-60 belong to classes 4,5,6,7 and 8. Finally, points 60-63, 63-66..99-102 belong to classes in the range of 9-22. We design a mapping from team names to codes to facilitate in conversion of .csv to .npy - Team Arsenal is 501, and so on, with Cardiff mapping to 537. Thus ours is a classification problem, to try and predict the correct score class for the teams.

The pipeline of our application project is as follows:

4.1 Feature Selection

One of our key goals is to find out which features, i.e, aspects of a team's play, affect the game result the most. To do this, we tried the following approaches to select optimal features as well as do some additional playing around with the features:

i) **SelectFromModel:** We use SelectFromModel to select the best features that affect the result. It generated a slightly different combination of features for different seasons, showing that different features play a role depending on the season and number of seasons being considered. We were able to select some key features that appeared the most among these 11 seasons after being selected by SelectFromModel, and have been reported in Results.

ii) **PCA Dimensionality Reduction:** We were keen to do further experiments with this unknown data that we had on our hands. PCA seemed like a good bet as we had 100+ features, so reducing their dimensionality seemed like a logical approach. So every experiment that we did with 11 seasons and 6 models each in (i) above, we did with PCA as well. Results for both have been plotted together for juxtaposition.

4.2 Models:

We evaluate the following models and feed them with the above selected features:

i) **K-Nearest Neighbors:**

KNN is a non-parametric classifier that classifies a new instance by using the majority vote of its K neighbors, which are located at a minimum distance from this instance. For a distance $d : R^d \times R^d \rightarrow R$, x = new instance, $N_k(x) =$ K Nearest Neighbours, II is an identity function, thus KNN classification is expressed as:

$$f_{KNN}(x) = arg \quad max_{y \in Y} \sum_{i \in N_k(x)} II[y_i = y]$$

ii) **Random Forest:**

Random Forest is an ensemble classifier that combines multiple decision trees and thus overcomes the decision trees' limitation of overfitting. It involves bootstrap aggregating/bagging, is highly accurate and has a fairly quick prediction time.

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(\acute{x})$$

iii) **SVM:**

A Support Vector Machine takes labels in the set -1,1 and is a discriminative classifier. Its decision boundary is same as the Logistic Regression boundary. Its decision boundary is of the form:

$$f_{SVM}(x) = sign(w^T x + b)$$

iv) **Stochastic Gradient Descent:**

The standard gradient descent algorithm updates the parameters θ of the objective $J(\theta)$ as,

$$\theta = \theta - \alpha \bigtriangledown_\theta E[J(\theta)]$$

where the expectation in the above equation is approximated by evaluating the cost and gradient over the full training set.SGD on the other hand, only computes the gradient of the parameters using only a single or a few training examples ignoring the expectation in the update completely. The new update is given by, with a pair $(x^{(i)}, y^{(i)})$ from the training set.

$$\theta = \theta - \alpha \bigtriangledown_\theta J(\theta; x^{(i)}, y^{(i)})$$

v) **Logistic Regression:**

It is a probabilistic discriminative classifier. It models the decision boundary using a linear function in a binary case. If the case is multiclass the boundary is piecewise linear.

$$f_{LR}(x) = arg \quad max_{c \in y} P(Y = c|x)$$

vi) **Neural Network (Multi-Layer Perceptron):**

MLP is a network of neurons called perceptrons. MLP classification algorithm is trained using back-propagation that consists of two steps:

a) Predicted outputs are computed corresponding to given inputs in the forward pass using the following equation:

$$y = \varphi(\sum_{i=1}^{n} w^i x^i + b) = \varphi(w^T x + b)$$

b) Partial derivatives of the cost function are WRT different parameters are "back-propagated" through the network in the backward pass.

The output is computed from multiple real-valued inputs by forming a linear combination based on the input weights and then possibly putting the output through some nonlinear activation function. This activation is generally chosen to be a logistic sigmoid 1/(1+e^{-x}) or a hyperbolic tangent $tanh(x)$.

4.3 Hyperparameter optimization:

We use GridSearchCV to choose optimal hyperparameters for the above models. Hyperparameters of each model fed into the grid are:

i) **K-Nearest Neighbors:**

n_neighbors, weights [uniform, distance], algorithm [auto, ball_tree, kd_tree, brute], and metric[euclidean,manhattan,chebyshev,minkowski]

ii) **Random Forest:**

n_estimators [10-220], criterion [gini, entropy], max_features ['auto', 'sqrt', 'log2'].

iii) **SVM:**

kernel [linear, rbf], gamma [1e-1, 1, 1e1], C [0.05, 0.1, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5]

iv) **Stochastic Gradient Descent:**

loss [log, hinge, modified_huber], penalty [l1, l2], alpha [0.0001, 0.001, 0.01, 0.1, 1.0], n_jobs [-1, 1, 2, 3]

v) **Logistic Regression:**

penalty [l2], dual [false], C [0.1, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5], fit_intercept [True, False], solver [newton-cg, lbfgs, liblinear, sag]

vi) **Neural Network:**

activation [identity, logistic, tanh, relu], solver [lbfgs, sgd, adam], alpha [0.000001, 0.00001, 0.0001, 0.001, 0.05, 0.1, 0.25, 0.5, 0.75, 1.0], learning_rate [constant, invscaling, adaptive]

5 Experiments and Results

5.1 Training and predicting:

We train the optimal models so obtained on an increasing number of seasons' data. More specifically, we perform an extensive study comprising of a total of 11 experiments on each of the 6 optimal models:

Since we have data from the years of 2003-04 season to the 2014-15 season, we start testing the pipeline from the 2004-05 season all the way to the 2014-15 season, each time the train data set comprising of entries from the 03-04 season upto the previous year of the season being tested on. For instance:

i) Train on 2003-04 season and predict on 2004-05 season.

ii) Train on 2003-04 and 2004-05 range of seasons and predict on 2005-06 season.

iii) Train on 2003-04, 2004-05 and 2005-2006 range of seasons and predict on 2006-07 season and so on upto the 2014-15 season.

Table 1: Details of experiments performed

		Accuracies after Feature Selection					
Trained on seasons(starting years)	Tested on seasons	KNN	RF	SVM	SGD	LR	Neural Network
2003	2004	0.25	0.25	0.1	0.25	0.15	0.3
2003-2004	2005	0.25	0.3	0.3	0.1	0.2	0.3
2003-2005	2006	0.45	0.45	0.4	0.2	0.1	0.35
2003-2006	2007	0.4	0.45	0.35	0.1	0.2	0.25
2003-2007	2008	0.4	0.55	0.5	0.35	0.35	0.5
2003-2008	2009	0.35	0.4	0.2	0.1	0.1	0.2
2003-2009	2010	0.3	0.4	0.35	0.15	0.25	0.35
2003-2010	2011	0.3	0.45	0.6	0.1	0.25	0.45
2003-2011	2012	0.35	0.5	0.55	0.35	0.2	0.4
2003-2012	2013	0.4	0.5	0.5	0.1	0.15	0.5
2003-2013	2014	0.2	0.5	0.35	0.1	0.1	0.5

The next page contains accuracy plots obtained by testing various models after using SelectFromModel and PCA

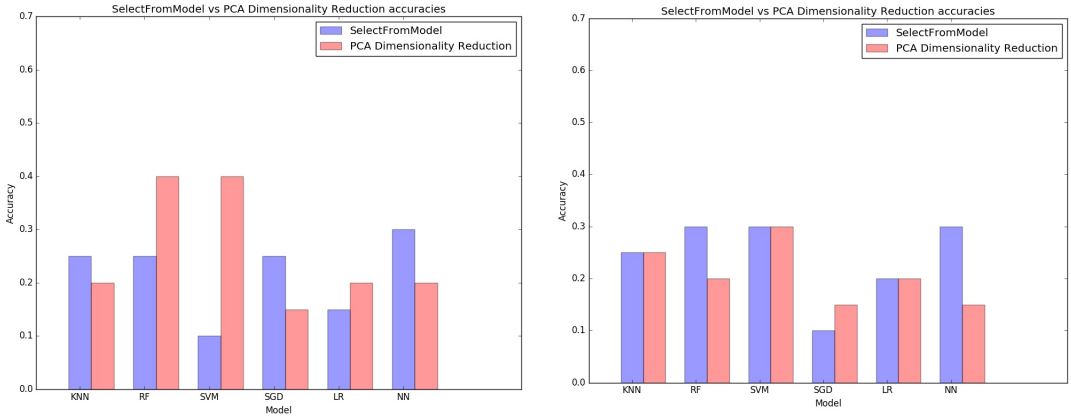

Figure 1: Seasons 2004-05 & 05-06

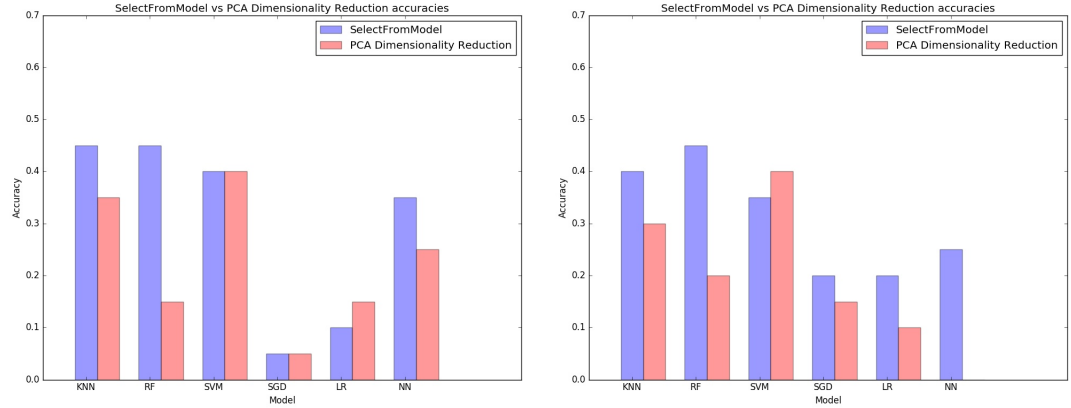

Figure 2: Seasons 2006-07 & 07-08

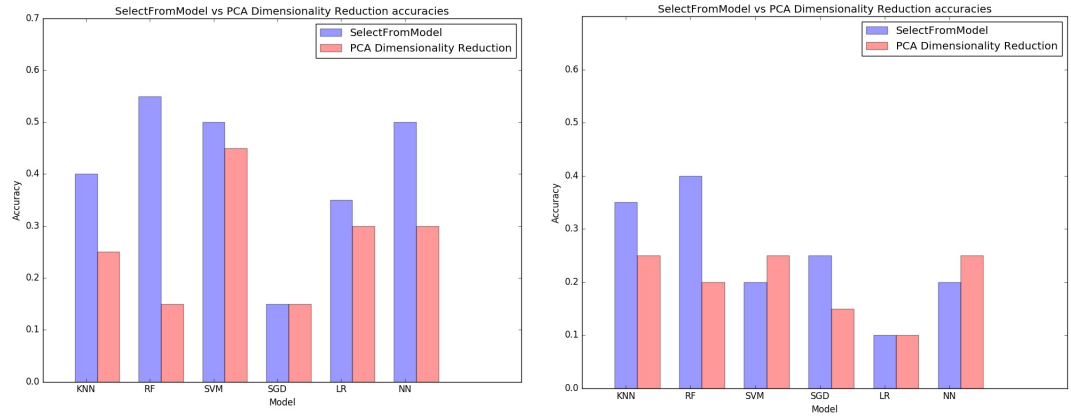

Figure 3: Seasons 2008-09 & 09-10

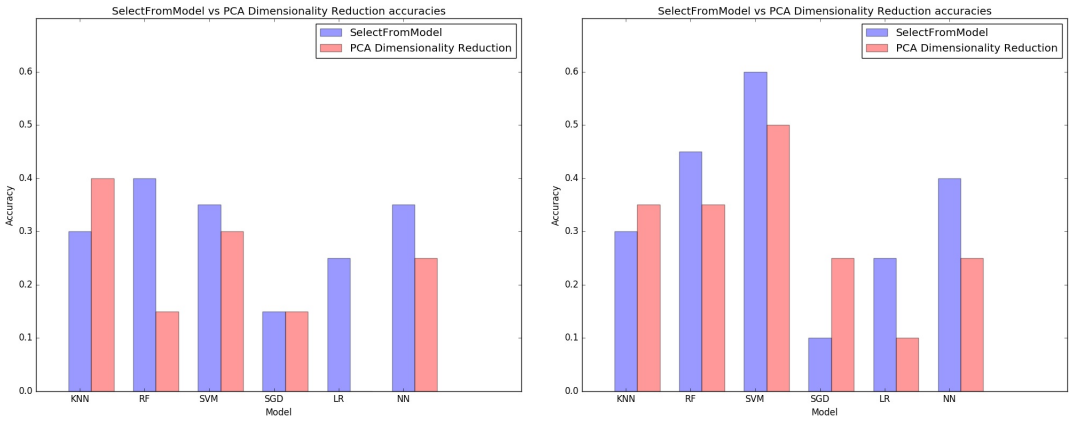

Figure 4: Seasons 2010-11 & 11-12

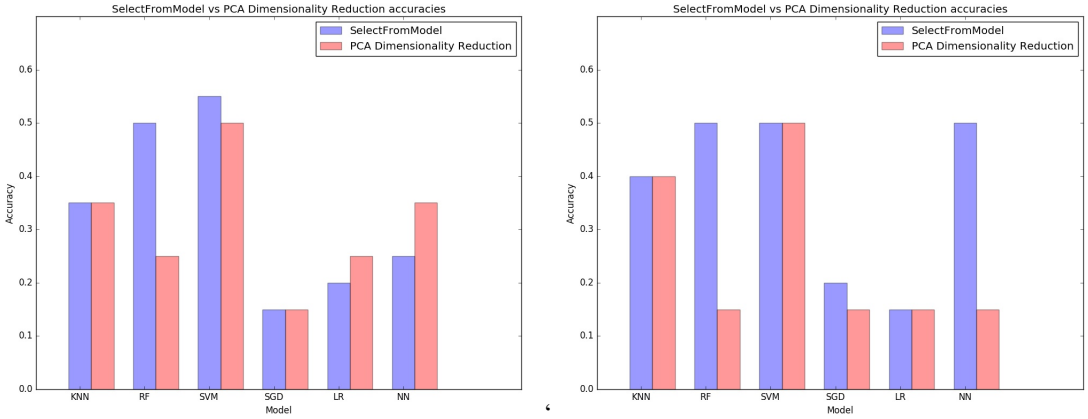

Figure 5: Seasons 2012-13 & 13-14

OBSERVATION: We noted the following key points regarding our results:

1) We (perhaps a bit naively) expected the accuracy to show a steady increase when we increase the number of seasons the models are trained on. This wasn't the case, and there seems to be no correlation between the amount of seasons the data is trained on. This can be attributed to:

a) Relegation of teams. Each season, 3 teams get relegated to a lower divison and do not feature in the next season, with 3 teams from a lower division being promoted and taking their place. So every season, there's 3 teams that the model is not trained on, unless the odd case where those 3 teams were present in one of the earlier seasons than the last season. This is a major cause for decreasing the accuracy.

b) Simply, the unpredictability of the English Premier League. EPL is the most competitive and unpredictable league of all soccer leagues in the world. There's been only a few occasions were a team has won the league for successive seasons. So, player transfers, managerial changes before a season's beginning, player morale etc are aspects that play a role but not accounted for by us.

2) PCA does not have a significant effect on increasing the accuracy. It does manage to outperform SelectFromModel for some seasons here and there for a few models, but generally, SelectFromModel's feature selection beats PCA's dimenstionality reduction.

3) Random Forest is the clear winner here, being a consistent high performer. We expected SVM and Neural Networks to show a similar performance, but they do have some fluctuations in performance; Random Forest dominates in almost all the seasons.

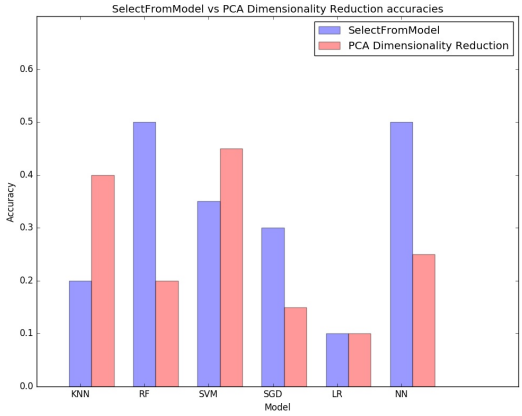

Figure 6: Final experiment, Season 2014

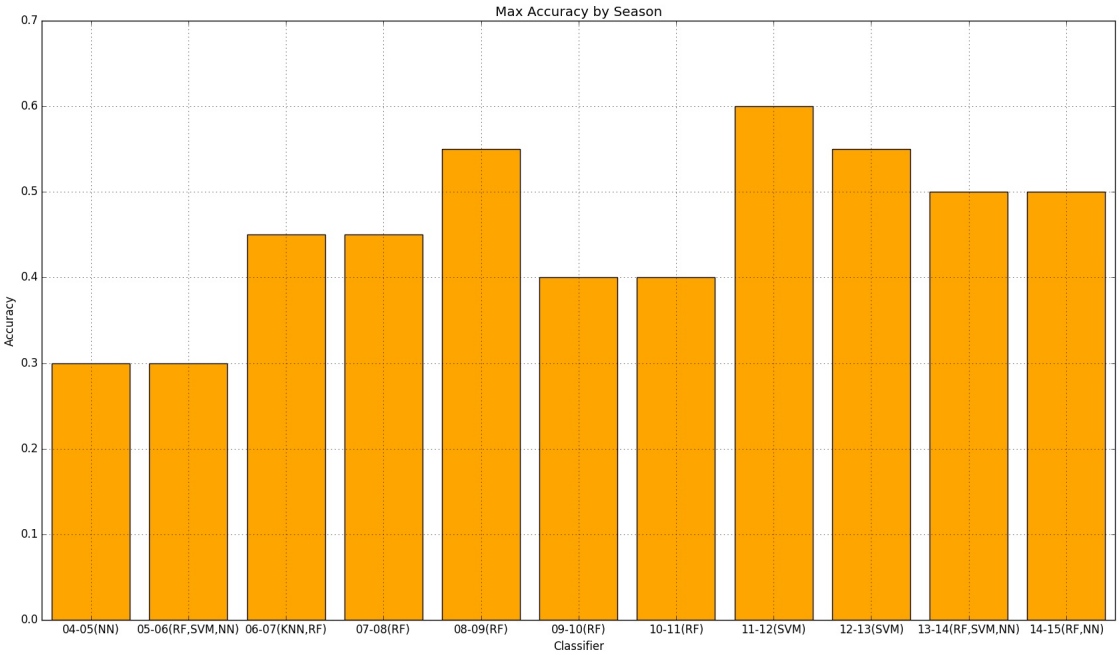

Figure 7: Max Accuracies by Season

4) Most important features seem to be ones such as proportion of games won after leading at Half Time, goals scored, bookings (yellow/red cards) in each of the first and second halves, clean sheets, etc. Interestingly, name of the team is also a key feature. We believe this is because there were consistent performances by some teams between 2004-2010, and this leads to weightage being given to the team name itself, which is very much how an average joe predicts the results - simply by knowing which teams are playing!

6 Discussion and Conclusion

Our best performing results seem to be on par with related work that had used multiple seasons' data, such as [2]. We exceed their best performing model, which managed to have an accuracy of 0.52. Ours reached a peak of 0.6 with SVM once and 0.55 multiple times with Random Forest, SVM and MLP Perceptron. Others such as [6] reached 0.66 once, which outperformed our best performing model.

Feature Name	Number of times Selected	Feature Name	Number of times Selected	Feature Name	Number of times Selected
Team	6	Scored zero	1	Scored from 18 yds +	3
Won	**11**	Scored 1	4	**Goals scored**	**8**
Lost	**11**	Scored 2	5	Cross	4
Goal Difference	4	Scored 3	4	Corner	3
Lost at home	1	Conceded 0	4	Yellow to red	1
Lost away	3	Conceded 1	6	**First Half booking**	**8**
Goals for away	3	Conceded 5+	3	**Second Half Booking**	**10**
Goal diff away	7	Away Goals scored	4	Home Booking	4
Full time win(after leading at half time)	3	First half goals scored	1	Away Booking	5
Win %(after leading at Half Time)	**8**	Second half goals scored	4	Penalties won	1
Half time losing	**9**	First 15mins goals scored	1	Away penalties won	1
Full time lose	1	First Half goals conceded	1	Penalty conceded	4
Win %	**11**	Second Half goals conceded	5	**Clean Sheets**	**10**
Half time draw	1	Last 10mins goal conceded	1		
Win %	**11**	Scored from 18 yds	1		

Figure 8: Most Important Features

Taking multiple seasons' data helped us see whether there was any correlation to increasing the training data by an absolute number of seasons to an increase in accuracy - there wasn't, due to results stated above under Results. Choosing Random Forest worked very well for us, as it consistently performed high results, so did SVM. Logistic Regression did not work as well as we had hoped, being a very poor performer on all the seasons. PCA dimensionality reduction did not outperform SelectFromModel as we had hoped, but it was worthwhile to compare their performances, and a good learning experience nonetheless.

Data extraction, cleaning, dealing with missing data, merging various seasons' data according to various teams took a lot of our time and was definitely the most challenging and cumbersome aspect of the project.

Given more time, we would have liked to try more powerful techniques, such as some deep learning ones, to see whether they outperform our best performing Random Forest model. We would have also liked to calculate expected goals per game using stats per game from previous matches such as various positions from which the goal was scored, technique used, whether it was a crucial goal or not, etc., and see whether it has any effect on increasing the accuracy of predicting league positions.

References

[1] ”Mark Lawrenson vs. Pinnacle Sports”. Web. Sept. 12, 2013. [http://www.pinnaclesports.com/en/betting-articles/soccer/mark-lawrenson-vs-pinnacle-sports]

[2] "Predicting Soccer Match Results in the English Premier League", Ben Ulmer and Matthew Fernandez, Stanford University. Dec. 11, 2014. [http://cs229.stanford.edu/proj2014/Ben

[3] "Machine Learning for Soccer Analytics", Gunjan Kumar, MS Thesis, KU Leuven, 2012-13. [https://www.researchgate.net/publication/257048220_Machine_Learning_for_Soccer_Analytics]

[4] Opta Sports. Web. [http://www.optasports.com/us]

[5] Whoscored - Football Statistics and Football Live Scores. Web. [https://www.whoscored.com]

[6]"Game ON! Predicting English Premier League Match Outcomes", Aditya Srinivas Timmaraju, Aditya Palnitkar and Vikesh Khanna, Stanford University. 2014. [http://cs229.stanford.edu/proj2013/TimmarajuPalnitkarKhanna-GameON!PredictionOfEPLMatchOutcomes.pdf]

[7] "Predicting outcome of soccer matches using machine learning", Albina Yezus, Term Paper, Saint-Petersburg State University. 2014. [http://www.math.spbu.ru/SD_AIS/documents/2014-12-341/2014-12-tw-15.pdf]

[8] ChampionAt. Web. [https://www.championat.com/]

[9] StatTo. Web. [http://www.statto.com/]

[10] "Statistical Modelling For Soccer Games: The Greek League", Dimitris Karlis and Ioannis Ntzoufras, Athens University of Economics and Business, September 1998. [http://homepages.cae.wisc.edu/ dwilson/rsfc/rate/papers/statistical-modelling-for-socce.pdf]

[11] StatBunker - Football Stats, Team Stats, Player Stats, Soccer Stats. Web. [https://www.statbunker.com/]

[12] English Premier League Officail Website - [https://www.premierleague.com/]