Perfect Form

A model for football (soccer) match result prediction using historical match result data

Patrick McGuinness
2016

Table of Contents

Abstract

Historical match result data from the English Premier League, 1998-99 season to the 2015-16 season, were explored and models created to estimate future match results. The exploration determined that a simple feature based on goal difference, had a high significance as a predictor. Five final models were tested comprising different algorithms using k-fold validation with the last season being used as a separate validation set. The models were evaluated using a metric related to maximum log-likelihood. An accuracy metric was also calculated. The algorithms logistic regression and a combination of a decision tree and logistic regression were found to perform reasonably. It is expected that refinements such as parameter tuning and bias correction could provide further improvements to models using goal difference as a feature.

Introduction

The market for gambling on the outcome of football matches is estimated to have a size of $700-1000 billion per annum. As such there is clear commercial value to football match prediction models. The aim of this project was to create models based on historical match result data. It was not anticipated that the models generated in the project would be commercially successful by themselves but that useful insights or features might be refined or added to other models to provide commercially valuble predictions.

Methods

Analysis was performed in R using the R-Studio environment.

Initial data collection

Match result data from the English Premier League, seasons 1998/99 – 2015/16 were sourced from www.football-data.co.uk and collated into a dataframe. `premiership`. Blank rows were removed and a new variable, `Season` was created to distinguish the different seasons.

Creation of current season 'snapshot' for each match

To provide a basis for feature analysis the league table information for each team at the time of each match was then calculated and added to a new version of the dataframe. This was accomplished by FOR looping through each season, and then using a mixture of methods to collate the table information. For each team, the Home record and Away record was compiled.

The cumsum() function from the dplyr package was used to compile the home record for the home team, and the away record for the away team. This summary function was felt to be an elegant method however it could not be used for the away record for the home team nor the home record for the away team. To collate the missing side of the league table further nested loopings were used. One FOR loop selected the match, and then a further nested FOR loop sought the last match in which the thus selected home team played away: the away record from this match was then pulled (with the match result added) to provide the away record for the home team. On completion of this, another FOR loop was used to repeat a similar process to collate the home

record for the away team.

This process produced a dataframe for each season which was then collated into a new dataframe `premiership2`.

This completed the collation of statistics from the current season for each match.

Creation of End-of-Season Tables

It was desired to prepare statistics from previous seasons also. Pertinently, when a season has had a few matches played, there is a scarcity of current season match results and so the data from the previous season was expected to be particularly valuable.

To create end-of-season tables, two dataframes, `final_tables_home` and `final_tables_away` were created using functions from the aforementioned dplyr package. This involved grouping the `premiership2` dataframe by `Season` and `HomeTeam` with group_by() and `Season` and `AwayTeam` respectively, and thus summarising the table information using summarise().

These two dataframes were then combined into one dataframe, `final_tables`.

Entries selected at random from this dataframe were were then checked against published league tables for integrity.

Creation of features

Home and Away Win / Draw / Loss ratios for each team were calculated as a starting point for analysis. Also dummy variables (`ResH`, `ResD`, `ResA`) for the match result (`FTR`) were created.

Initial explorations

Initial explorations were made using logistic regression on the dataset. This used the glm() function to predict `ResH` – i.e. to approximate the chance of a win for the home team. While this is not sufficient to predict all three match results it was assumed to sufficient to determine the significance of the predictive features. Models were evaluated using the p-values for the predictors and the AIC (Aikake Information

Criterion) metric.

model1 – model7: initial explorations using win/draw/loss ratios. To avoid volatility at the start of the season, the models were called on datasets filtered so that each team had played 15 matches.

After model7 two new variables, `T1GDr`, and `T2GDr`, reflecting the goal difference per game of Team1 and Team2 respectively (the Home and Away teams) was created.

Model8

This model proved insightful. Created with

```
model8 <- glm(ResH ~ T1HWr + T1HDr + T1AWr + T1ADr +
          T2HLr + T2HDr + T2ALr + T2ADr +
          T1GDr + T2GDr,
          data = premiershipF15, family = "binomial")
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.55911    0.53473  -1.046  0.29576
T1HWr       -0.35717    0.41341  -0.864  0.38761
T1HDr        0.02405    0.32603   0.074  0.94120
T1AWr       -0.52009    0.41978  -1.239  0.21536
T1ADr       -0.24604    0.33702  -0.730  0.46536
T2HLr        0.36614    0.42557   0.860  0.38959
T2HDr        0.42787    0.35610   1.202  0.22954
T2ALr        0.88557    0.40859   2.167  0.03020 *
T2ADr        0.69756    0.32663   2.136  0.03271 *
T1GDr        1.05646    0.16770   6.300 2.98e-10 ***
T2GDr       -0.54144    0.16633  -3.255  0.00113 **
```

The high significance of `T1GDr` and `T2GDr` suggested that it was surpassing other variables in value as a predictor. To explore this further, in `model9` and `model10` the win/loss/draw ratio variables for `Team1` and then `Team2` were dropped out, resulting in a reduction of the AIC metric from 5191 (model8) to 5185 (model9) to 5183 (model10).

Subsequence models thus focussed on the use of goal difference features.

`model12` used a new variable, `GDrdiff`, which was defined as the difference between `T1GDr` and `T2GDr`. This model resulted in an AIC of 5181, again an improvement.

Other explorations (`model11` and `model13`) using co-dependence and variant goal

difference (log Goals For / Goals Against) resulted in higher AIC values and were not pursued.

With `GDrdiff` identified as a suitable predictor variable of choice for the current season, it was decided to explore the same variable for previous seasons.

To pull in the data from `final_tables`, a temporary dataframe `final_tables_lookup` was created which contained the goal difference per game for each team and raised the `Season` variable by one to match the next season. Then by way of two right_join() commands from dplyr, these variables were matched to `premiership3` and the desired new variable, `GDrdiffLS` was created. This was tested in `model12b` and `model13b`.
`model12b` (which included `GDrdiffLS`) had an AIC of 3201, while that of `model13b` had an AIC of 3244, confirming the value of last season's data.

Imputation of missing values

The inclusion of features for the previous season presented a problem in that promoted teams were missing values.

To impute values for the promoted teams, it was explored using the mean goal difference per game values for the correspondingly relegated teams, i.e. the teams whose places were being taken by the promoted teams. These relegated teams were selected from `final_tables` by attaching a logical vector to select the last three out of twenty repeatedly: this method was applicable as all the league tables had twenty teams and the tables were in order so the promoted teams were the bottom three, and then filtering to select this. The relegated teams were stored in the variable `relegations`. `RelegationsGD` was then also created to calculate the mean goal difference per game values, using group_by() and summarise() from dplyr.

The imputed values were then introduced into a new dataframe `premiership4` by means of right_join() from dplyr.

These imputed values were then tested by `model14`.

`model15` used another strategy for imputation, which was to take the mean of these values across all seasons. These were stored in a new dataframe `premiership5`. Results showed a stronger AIC metric for model15 (4825) compared to `model14` (4830) so this method was used for the imputation. This made intuitive sense in the that quality of the promoted teams would likely be more closely related to the average of the relegated teams over many seasons than those of the last year alone.

A weakness of this method of course is the equal treatment of the three promoted teams. A refinement would be to look at the goal differences from the lower league, and convert those so the promoted teams with better goal differences would be given a better imputed value than the lesser performing promoted team.

A variable `GDrdiffLS2` - i.e. a similar measure for the season before the previous season, was then calculated using similar procedures and the missing values imputed in the same way. This was then tested in `model16` which although not highly significant for the predictor (p value of 0.145747) and coefficient of 0.10605, it was concluded that it was making a useful contribution. This was with a filter of 15 matches for the current season, so it was expected to be more useful when less than 15 matches had been played in the current season.

Preparation for new algorithms

For analysis with different algorithms it was necessary to eliminate NA values from the predictors.

Missing values of `GDrdiff` for the first match of the current season were imputed as zero.

For the first two seasons, there were missing values of `GDrdiffLS` and `GDrdiffLS2`. As `GDrdiff` was considered a more crucial predictor, the first season's data was discarded. For the second season, `GDrdiffLS2` was imputed to equal `GDrdiffLS`.

To facilitate readouts and analysis, the result variable `FTR` was tidied and reordered so that it was stored internally as 1 – H, 2 – D, 3 – A to match table convention, rather than the alphabetical default.

The variables `Team1P` and `Team2P` were rescaled to represent the fraction of the season played for the home and away team respectively.

These changes were applied to the new dataframe `premiership6`.

Two dataframes were then created by splitting `premiership6`, a training (train) and test dataset (test) with the ratio 80/20.

Exploration of alternative algorithms

Up to this point it was only logistic regression that had been used. This was felt to be useful to diagnose value in predictor variables but not known to be the best for a finished product.

For measuring the value of a model a likelihood metric was used to evaluate the test dataset. This was created as the function testLogL which took the exponential of the average log-likehood for the predictions (i.e. so this was a number between 0 and 1, which would return 0.3333 for the 'lobster' (a random selection) model H = 1/3, D=1/3, A=1/3, but increments much above 1/3 were expected to be hard to obtain. For instance, if a model was as good as possible with H=0.5, D=0.2, A=0.3, this metric would expect to return exp(0.5 * log 0.5 + 0.2 * log 0.2 + 0.3 * log 0.3) = 0.3571 in the long run if all the matches were the same. Whereas if all matches had 'true' probabilities of (0.6, 0.15, 0.25) such a model could theoretically obtain 0.3916 as a maximum in the long run.

The function `likelihood()` later in the model accomplished the same statistic but was generalised to cover any length of predicted values.

`model17` was a decision tree model. It produced a very simple but surprisingly effective result, evaluating around 0.37 on the likelihood metric, from simply placing a match into one of three categories based on `GDRdiffLS`. Attempts to improve this result with more 'leaves' were not successful.

`model18` was the 'lobster' model (i.e. emulations of a lobster randomly selecting one of three objects in a tank. No lobsters were actually used in the programming). The likelihood metric which resulted was 0.3333, which gave some confirmation of the integrity of the methods.

`model19` was a random forest model. This model produced a poor result compared to the decision tree (0.3534 at the instance of writing up this report). It was necessary to have many trees (2500) even to generate a meaningful result for this problem, as smaller values produced predictions of 0 for actual match outcomes, which pertinently resulted in a likelihood metric of 0. Pertinently, incorrect predictions of certainty are catastrophic in a real sports betting situation, so the metric was considered correct to have this property.

It appeared that for this problem the random forest approach was weaker than the decision tree.

`model20` was a logistic regression model, using a pair of logistic regression to

calculation p(H) and p(A) separately, and then p(D) – p(H) – p(A). At the instance of writing this report, it produced a likelihood of 0.3803.

`model21` and `model22` used calls from the nnet package (neural network models). `model21` used a `multinom()` call and at the instance of writing this report produced a likelihood of 0.3805. This model included predictors of `Team1P` and `Team2P` to incorporate the depth of the current season.

The result from `model21` – a `nnet()` model with a three node intermediate layer – was 0.3771

An issue was how to account for the incompleteness of the current season. Models23 - 30 explored ways of accounting for this.

`model23` was a neural net with the feature `sqrt(Team1P + Team2P) * Gdrdiff` to enhance `GDrdiff`. It produced a metric of 0.3771 at the instance of writing this report.

`model24` was an attempt to improve the decision tree model with greater complexity. That and undocumented attempts were unable to improve the resulting metric.

`model25` through to `model30` represented an attempt to model the result of the match by first modelling the supremacy (home team goals minus away team goals) and then producing a model from that prediction. This did not result in an improvement to the neural net model in `model30`, however the basis used for the supremacy model was considered to be a sound way of dealing with the incompleteness of the current season.

From the summary of `model27`

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.41535    0.02273  18.271  < 2e-16 ***
GDrdiffLS    0.85825    0.02586  33.193  < 2e-16 ***
b            0.17353    0.04894   3.546 0.000395 ***
a:c          0.64876    0.06777   9.574  < 2e-16 ***
```

where `b = GDrdiffLS2 – GDrdiffLS, a = Gdrdiff – GdrdiffLS, c = 0.5 * (Team1P + Team2P)`

Final modeling trials

Five algorithms were chosen for testing via a 10-fold cross-validation and also testing via a validation set. The cross validation set was created as the dataframe `premiership7` and the validation set as `premiership7V`. The validation set was the data pertaining to the last season (2015/16). The cross-validation sets were randomly selected from the remaining data from `premiership6`. To evaluate the validation set, all the cross-validation data was used to create the necessary model.

Results were determined with the likelihood metric and a new function, `likelihood()` was written to determine this from a vector of predicted probabilities.

Another function, `accuracy()` was written to calculation the accuracy as from a confusion table. In practice, this metric simply measured whether match was considered more likely to be won by the home team or the away team.

To help calculate the accuracy from model predictions it was necessary in some cases to convert an output of a vector of probabilities into a single prediction of the result as a factor variable. A function `matrix2class()` was written to accomplish this.

Decision Tree + Logistic Regression

Drawing on the surprising robustness of a simple decision tree, this method sought to improve on it by clustering according to the leaves of a decision tree model, and then apply separate pairs of logistic regressions for each of the leaves, to predict the probability of home wins and away wins separately. The decision tree model was taken from model31 and split according to `GDrdiffLS` as the variable `DTnode`; the three clusters were defined by

```
Dtnode = 2:                    GDrdiffLS    >= 0.1885965
Dtnode = 6:  -0.9736842 <=     GDrdiffLS    < 0.1885965
Dtnode = 7:                    GDrdiffLS    < - 0.9736842
```

i.e. so that the first cluster represented games with a strong advantage for the home team, the second more level affairs, and the third matches likely to be away wins.

The call for the logistic regressions included the predictors as a:c + GdrdiffLS + b

Predictions of match results were passed to the object `DecTreeLogRegProbMatrix`, which was a matrix of the predictions for (H, D, A) for the training set. For calculating accuracy a vector of match predictions was created as `DecTreeLogRegClassVector`. The predicted probabilities for the actual match outcomes was passed to `DecTreeLogRegPredictions`.

Similar variables were created for evaluating the validation set, these were

`DecTreeLogRegVProbMatrix`
`DecTreeLogRegVClassVector`
`DecTreeLogRegVPredictions`

Decision Tree (Alone)

Another evaluation was undertaken for the decision tree algorithm by itself. Similar variables were created for this model with titles `DecisionTree` as the prefix.

Random Forest

As above for the random forest algorithm. Output variables had the prefix `RandomForest`

Logistic Regression

As above for a logistic regression algorithm alone. This used a pair of logistic regression to compute the Home and Away win probabilities. It called using the predictor variables as `a:c + GdrdiffLS + b`

Neural Net

This used the nnet() function from the package of the same name. The model used a three-node intermediate layer with the call `FTR ~ a + b + c + GdrdiffLS`

Collection of results

A dataframe `summary` was created with the following variables,
`models` – vector of algorithm names
`k-fold likelihoods` - likelihood metric for the cross validations
`k-fold accuracy` – accuracy metric for the cross validations
`validation likelihoods` – likelihood metric for the validation set
`validation accuracy` – accuracy metric for the validation set.

Results and Discussion

Results of final modeling

At the instance of writing this report, the print of `summary` was

```
               models    k_fold_likelihoods      k_fold_accuracy
1             DT + LR             0.3759587            0.5281250
2       Decision Tree             0.3689336            0.5279605
3       Random Forest             0.3370782            0.4710526
4 Logistic Regression             0.3763816            0.5291118
5      Neural Network             0.3744540            0.5282895

                     validation_likelihoods validation_accuracy
1                                 0.3542421           0.4631579
2                                 0.3444160           0.4552632
3                                 0.3134958           0.4210526
4                                 0.3518680           0.4500000
5                                 0.3483187           0.4473684
```

The results support the use of goal difference metrics over the current season, the previous season and the season before the previous season for use as predictors for future match results.

The explorations suggest that other metrics based on simply the match results (e.g. win percentages, points scored) are far less useful. It is not known whether they are entirely made redundant when a suitable predictor based on goals scored and conceded for each of the historical matches is available. While writing the report, it was noticed by the author that variables reflecting the away form of the away team `T2ALr` and `T2ADr` were not considered except with those reflecting the home form of the away team, and inclusion of just these particular variables resulted in a slight apparent improvement to `model10`. It is a point of interest that the away form of the away team seems pertinent. Goal difference metrics were not split into home and away, and the author believes it might result in a modest improvement to the model were these considered separately.

A winning model

The algorithms Decision Tree + Logistic Regression, Logistic Regression and Neural Network all performed creditably with likelihood metrics for the cross-validation over 37. It is difficult to select one model as the best, however based on this

likelihood metric for the cross validation set the best performing model was simply Logistic Regression.

This logistic regression model used for the validation set was:

p(Home Win) = logistic(α) = exp(α) / ((exp(α) + 1)

where α = `-0.1396 + 0.9157*GDrdiffLS + 0.6622 a*c + 0.1683*b`

p(Away Win) = logistic(α)

where α = `-1.1175 + 0.9751*GDrdiffLS – 0.6799 a*c – 0.1419*b`

with

P(Draw) = P(Home Win) – p(Away Win)

Other points for further research: this model considers the goal differences for seasons separately. There also is the issue with bias for the current season, as at various points a team may have played a strong or weak selection of teams, and this could be particularly important near the start or perhaps the end of a season. A more refined model could investigate methods of accounting for this bias, and perhaps disregarding seasons as a method of collating the goal differences for each game.

Recommendations

Although the analyses performed are based on traditional match result information, model results appear quite respectable with accuracies over 52% (reported accuracies for bookmaker's predictions are 53% [source https://www.kaggle.com/hugomathien/soccer].

With refinements, such as improved handling of promoted teams, data from other leagues, or an adjustment of the goal difference statistic, it is possible that a commercially successful betting model might result, or a feature that could be part of such a model, and these avenues appear worth pursuing in the opinion of the author.

It appears that the use of logistic regression is suitable both for analysing features and final model building for this purpose.

Appendix

Dataframe Structures

Premiership- Dataframes

Variables included in the dataframes premiership1, premiership2, premiership3, premiership4, premiership5, premiership6, premiership7, premiership7V

(listed in alphabetical order)

a	GDrdiff - GDrdiffLS
AwayTeam	Away Team
b	GDrdiffLS2 - GDrdiffLS
c	Fraction of current season played [0,1]
Date	Date of match
Div	Division
DTnode	Cluster for Decision Tree + Logistic Regression model
FTAG	Goals scored by away team
FTHG	Goals scored by home team
FTR	Result
GDrdiff	T1GDr – T2GDr
GdrdiffLS	T1GDrLS – T2GDrLS
GDrdiffLS2	T1GDrLS2 - T2GDrLS2
HomeTeam	Home Team
K	Cluster for K-fold validation. [1,10]
meanGDr	mean goal difference per game for relegated teams, all seasons
ResA	Dummy variable for FTR. 1 = Away Win, 0 = otherwise
ResD	Dummy variable for FTR. 1 = Draw, 0 = otherwise
ResH	Dummy variable for FTR. 1 = Home Win, 0 = otherwise
Season	Season. [1, 18] 1= 1998/99, 18 = 2015/16
Supr	Supremacy. FTHG - FTAG
T1ADr	Home Team Away Draw ratio, current season
T1ALr	Home Team Away Loss ratio, current season
T1AWr	Home Team Away Win ratio, current season
T1GDr	Home Team Goal Difference per game, current season
T1GDrLS	Home Team Goal Difference per game, previous season
T1GDrLS2	Home Team Goal Difference per game, season before previous
T1Glr	Home Team log (Goals For / Goals Against), current season
T1HDr	Home Team Home Draw ratio, current season
T1HLr	Home Team Home Loss ratio, current season

T1HWr	Home Team Home Win ratio, current season
T2ADr	Away Team Away Draw ratio, current season
T2ALr	Away Team Away Loss ratio, current season
T2AWr	Away Team Away Win ratio, current season
T2GDr	Away Team Goal Difference per game, current season
T2GDrLS	Away Team Goal Difference per game, previous season
T2GDrLS2	Away Team Goal Difference per game, season before previous
T2Glr	Away Team log (Goals For / Goals Against), current season
T2HDr	Away Team Home Draw ratio, current season
T2HLr	Away Team Home Loss ratio, current season
T2HWr	Away Team Home Win ratio
Team1AwayD	Home Team Away Draws, current season
Team1AwayGA	Home Team Away Goals Against, current season
Team1AwayGF	Home Team Away Goals For, current season
Team1AwayL	Home Team Away Losses, current season
Team1AwayP	Home Team Away Matches Played, current season
Team1AwayW	Home Team Away Wins, current season
Team1GA	Home Team Goals Against, current season
Team1GD	Home Team Goal Difference, current season
Team1GF	Home Team Goals For, current season
Team1HomeD	Home Team Home Draws, current season
Team1HomeGA	Home Team Goals Against, current season
Team1HomeGF	Home Team Goals For, current season
Team1HomeL	Home Team Home Losses, current season
Team1HomeP	Home Team Home matches played, current season
Team1HomeW	Home Team Home Wins, current season
Team1P	Home Team matches played, current season
Team1Pscaled	Home Team fraction of current season played [0,1]
Team1Pts	Home Team total points, current season
Team2AwayD	Away Team Away Draws, current season
Team2AwayGA	Away Team Away Goals Against, current season
Team2AwayGF	Away Team Away Goals For, current season
Team2GF	Away Team Goals For, current season
Team2HomeD	Away Team Home Draws, current season
Team2HomeGA	Away Team Home Goals Against, current season
Team2HomeGF	Away Team Home Goals For, current season
Team2GD	Away Team Goal Difference, current season
Team2GF	Away Team Goals For, current season
Team2HomeD	Away Team Home Draws, current season
Team2HomeGA	Away Team Home Goals Against, current season
Team2HomeGF	Away Team Home Goals For, current season
Team2HomeL	Away Team Home Losses, current season
Team2HomeP	Away Team Home Matches Played, current season
Team2HomeW	Away Team Home Wins, current season
Team2P	Away Team Matches Played, current season

Team2Pscaled	Away Team fraction of current season played [0,1]
Team2Pts	Away Team total points, current season

final_tables Dataframe

(variables listed in column order)

Div	Division
Season	Season [1,18]
Rank	Finishing position [1,20] 1 = League Winner, [17,20] - relegated
Team	Name of Team
HomePlayed	Home Matches Played
HomeW	Home Wins
HomeD	Home Draws
HomeL	Home Losses
HomeGF	Home Total Goals For
HomeGA	Home Total Goals Against
AwayPlayed	Away Matches Played
AwayW	Away Wins
AwayD	Away Draws
AwayL	Away Losses
AwayGF	Away Total Goals For
AwayGA	Away Total Goals Against
Played	Total Matches Played
GF	Total Goals For
GA	Total Goals Against
GD	Goal Difference
Pts	Total Points

Contact details

Comments about this research are welcomed to squirrelpatrick@protonmail.com