

Soccer Predictions - ISyE 7406 Data Mining Project

Group C9

April 23, 2017

Team

Parit Burintrathikul - 48129 - paritb@gatech.edu

Spencer Collins - 92822 - scollins46@gatech.edu

Johnny Humphrey - 15569 - johnny.humphrey@gatech.edu

Shoili Pal - 43987 - spal41@gatech.edu

Steven Yeh - 28005 - syeh35@gatech.edu

Abstract

The goal of this project is to predict soccer match outcomes. Using non-statistical heuristic models as baselines, we develop predictive models based on data mining techniques. We try to determine the factors having the greatest impact on outcomes. We compare our results to professional gambling websites' odds. Finally, we suggest ways that could possibly lead to improving our predictions further.

Introduction

Motivation and Description

Sports are a popular form of entertainment. A form of entertainment parallel to the playing or watching matches themselves is to predict the outcomes. Much time, effort, and money are spent trying to make accurate predictions, especially at a rate better than everyone else. With this project we enter that highly competitive parallel stream. The sport we chose to model was soccer, with some general statistics coming from European soccer and our predictive models built based on data from the English Premier League (EPL). We decided to concentrate on one league since we had more data for it and since different leagues tend to have different styles of play which could affect the accuracy of our models.

Challenges

The variability in sporting outcomes, and the fact that it is hard to quantify the variability given that athletes have good days and bad days, is what adds novelty to sporting events, and what helps make them so entertaining. However, it makes modeling sports with a high degree of accuracy much more difficult.

While we decided to model the outcome of a match instead of the scores, soccer has three possible outcomes, not just win and loss but also ties. In the English Premier League there are no tie-breakers. This makes the task of prediction more complex since the outcome is not binary. We tried several different model types and features to see which did the best job of accounting for the variation in outcomes, as measured by the accuracy of our predictions.

Data

Data Source and Description

The data we are using is a dataset obtainable at <https://www.kaggle.com/hugomathien/soccer> in the form of a SQL database. It was originally created by scraping several websites and crowdsourcing. It contains information about soccer matches in the top tier leagues from 11 different European countries. The following are the tables in the database and a description of the data that they hold.

- Country: Country name and ID for 11 European countries.
- League: League name, country, and ID for the top league in the 11 different countries.
- Team: Team name, league, IDs for this database, for the match information web site, and for the FIFA video game from Electronic Arts (EA). 299 records total, 34 in the EPL.
- Match: Season, match IDs, home and away team goals, home and away player starting position and IDs, match events(goal types, possession, shots on/off, penalty cards, fouls committed, corner kicks), betting odds from up to 10 providers. 25979 matches total, 3040 total for the EPL spread across 8 seasons.
- Player: ID, name, FIFA ID, birthday, height, and weight for 11060 total players.
- Player Attributes: Player ID and FIFA ID, date updated, 38 FIFA attributes (overall rating, potential, attacking and defensive work rate, crossing, finishing, reactions, shot power, etc.). Each row represents an updated FIFA profile for a given player, so multiple rows per player are possible.
- Team Attributes: Team ID and FIFA ID, date updated, 21 FIFA team attributes (build-up classes and values, chance creation classes and values, defense classes and values)

Data Pre-processing

In our initial exploration of the database, we quickly realized that many of the variables in the dataset had a lot of missing values, mostly in the betting data and players in each match variables. Many of the players' data were crowdsourced by individuals and thus were only filled in haphazardly. This may lead to being biased towards players who are popular and outstanding. Therefore, we decided to remove all player data and focused solely on the team data like overall style of play and match related data like goal differentials, specifically in the English Premier League as we had more data for that.

We joined all the separate tables except the players table into one aggregated dataset, cleaning out N/A values as well as removing all the duplicated identifiers. We built two separate datasets, one including and one excluding betting data. Our assumption was that since the betting data were modelled and calculated by betting companies, they would be highly correlated to the match outcomes and since our objective is to predict match outcomes, building our predictions using other prediction data would not be ideal. However, we also decided to include them in separate models to compare the performances of models with and without the betting data. The dataset ultimately contains each match as one row, with predictor variables pertaining to the teams and the response variable being win, loss, or draw for the home team.

While the Match database table had the home and away goal totals, it did not explicitly have the goal differential, which we were interested in. We created a custom table to hold the results of a team's home or away goal differential going into each match, with the time horizons for calculating the differential of 1, 2, 3, 5, and 10 games, as well as season-to-date.

Exploratory Data Analysis

Looking at each league across all of their seasons in the database, we can find the average percentage outcomes of a home win, an away win, or a draw for the league. We found that there was some variation. [Appendix Tables 1 & 2]

x	Home Win %	Away Win %	Draw %
High	48.8	33.8	28.3
Low	41.7	27.0	23.2
Average	45.9	28.7	25.4

Looking specifically at the EPL, and looking at per-season statistics, we notice that the average EPL season is a lot like the average across all European leagues, but there is still some fluctuation in the home-win/away-win/draw percentages across years. [Appendix Tables 3 & 4]

x	Home Win %	Away Win %	Draw %
High	50.8	32.4	29.2
Low	41.3	23.7	20.5
Average	45.7	28.5	25.8

Finally, as a reference for a good prediction accuracy, we note that the professional oddsmakers correctly predict home-win/away-win/draw for a match in the EPL 53% of the time. [Appendix Table 5]

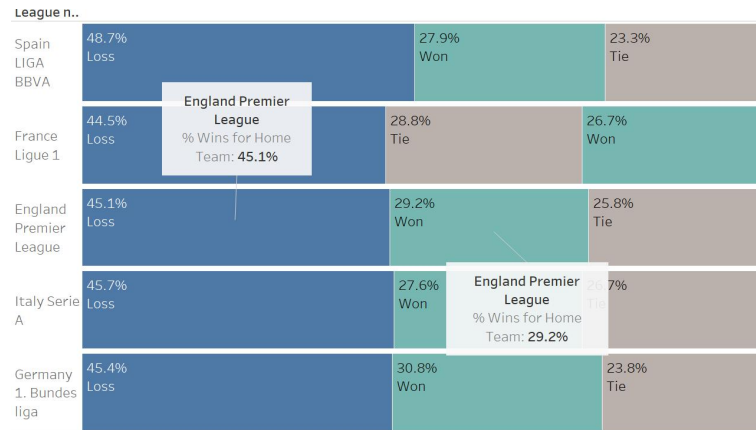
Methods

Baseline

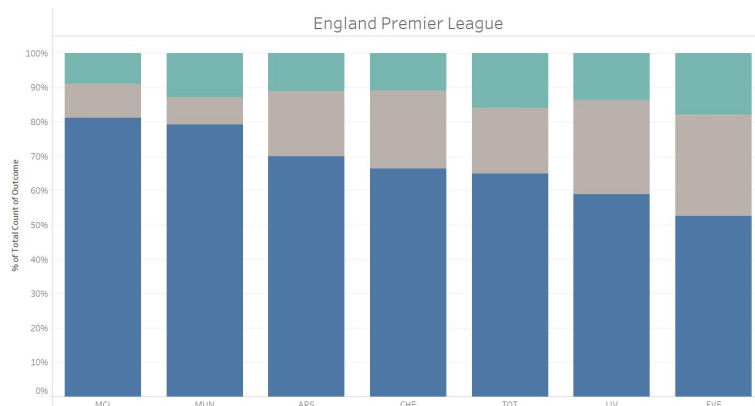
First we establish a baseline against which we will compare our models. We used a couple of prediction heuristics for this.

Home Wins

This heuristic for predicting winning outcomes is correct 46% of the time, as we see from the data distribution tables. That is better than random picks based on the 46% - 28% - 26% distribution of home-win/away-win/draw, which would be about 36%, but it is still significantly behind the oddsmakers' accuracy at 53%, so we hope to do better with a statistical model.



This heuristic works well for the English Premier League as well with home teams winning 45% of the time in our data. It is interesting to note that the home advantage is more pronounced for the top teams.



Goal Differential

The Home Wins rule for picking the winner is a good heuristic only for predicting home team wins. For a simple way to pick between the three outcomes goal differential is a good model.

The correlation of goal differential with wins is common knowledge in the sport community. It is obvious that, when wins are determined by scoring more than your opponent, something that measures that ability should correlate well with wins. For this reason, web sites like ESPN list goal differentials along with the other statistics they provide in their standings. Once again, soccer has draws as an outcome, and they are not as good as a win but are better than a loss, the English Premier League weighs wins as 3 points and draws as 1 point when calculating total points to determine a team's position in the league table.

As an example, when we use that metric for points and compare a team's point total with their goal differential for the 2015 - 2016 EPL season, we find a correlation between points and goal differential of 0.961, which is very strong.

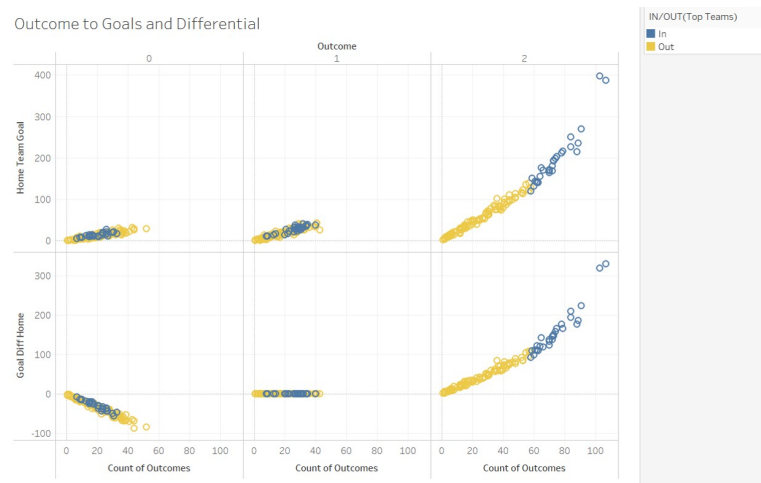
While we have seen a strong correlation between a season-long cumulative goal differential and total points, these are both aggregate measures, and so we still needed to determine how well goal differential helps predict individual match outcomes. To make those predictions we need to decide upon rules for mapping goal differential information into a match outcome. We use the following:

1. As we've seen in the outcome distributions, having the home field is an advantage, so we keep track of home goal differential and away goal differential for teams separately.
2. We treat seasons separately. It is possible that there could be some carry-over from season to season, but we don't try to determine this. For our calculations, goal differential starts back at 0 at the start of every season.
3. It's possible that goal differential is a stronger predictor over some number of the most recent games than season-to-date. We checked results using goal differentials calculated from the most recent single game, and the most recent 2, 3, 5, and 10 games.
4. We need to compare average per-game goal differentials even if the total is determined over some number of games. For example, if the home team has played 3 home games but the away team has only played 1 away game, it wouldn't be a fair comparison to use a 3-game home total against a 1-game away total.
5. The rule we use to determine a draw is if the home team's home goal differential average is within a half point of the away team's away goal differential average, then we predict a tie. If the difference is greater than a half point, then the team with the greatest goal differential is picked as the predicted winner.

We note that these rules are not a Data Mining or Machine Learning model. We aren't estimating any parameters to minimize error. For example, it is possible that we would have greater accuracy if we used some number higher or lower than 0.5 as the cutoff for draws. We did not seek to find out what that number

might be. We are only trying to see if we can improve on our baseline Home Wins rule. Also, even if goal differential isn't enough by itself for predictions, we may find that it is a useful feature to have in a model.

After calculating the goal differentials and then the predicted outcomes, we compared the actual outcomes and found that the accuracy in the EPL of the goal differential predictions was about 43%. [Appendix Table 7] This is actually slightly less than simply predicting that Home Wins.



Models

We tackle the problem in two ways -

- as a 3-class classification problem into wins, losses and draws for the home team.
- as a 2-class classification problem where the response is home win or not.

We use both parametric and non-parametric methods to model the problem. While the parametric models of logistic regression and SVM will give us a clear idea of the relation between the predictor variables and the response, sometimes non-parametric models like random forests can learn patterns that are not obvious to the human eye and give good results, especially in an area as unpredictable as sports. Hence we try both.

We use the same models for the binary classification and the 3-class classification.

Hundred fold validation is used on the regression and SVM models to establish the testing error.

Parametric

Logistic and Multinomial Regression

Logistic regression was used in the two-class prediction problem and Multinomial regression for the 3-class problem. Relying on the logit function as the link function between the calculated probability and the linear regression function, the model fits coefficients for each predictor variable. Logistic regression does not make any assumptions about common variance, and can be used as a robust estimator in this problem.

Support Vector Machines

Support Vector Machines were chosen as a model for predicting both the two-class (Win/Not) and three-class (Win/Loss/Draw) problems. SVMs are useful in high dimensional space and can be tuned using different kernel functions for the decision function and different decision boundaries 'gamma'. A grid search was performed to find the best value of gamma and several kernels were tried like the radial and sigmoid kernels.

Ensemble of Regression and SVM

Bradley Terry Model

Bradley Terry has been used to estimate sports outcome because of its ability to consider both teams attributes and records. It incorporates the Bayesian distributions of the home and away records of each team as well as attributes. For this model we trained the model including the records and goal differential.

Non-Parametric

Random Forest

Random Forest was chosen as one of the models to try as it is known to give good accuracy in many cases and is not too computationally intense. It was used for both Win/Not Win and Win/Draw/Loss prediction.

Results

The following table contains the mean accuracy and variance for each method when the calculations were run 100 times. The SVM model without betting data displayed the highest testing accuracy mean for the 2-class with ~64.5%. Random Forests with betting data had the highest accuracy for 3-class predictions, with respective accuracies of ~50.9%.

For Binary Classification

Name	Mean Error
Bradley Terry Model	0.536428
Logistic Regression (with Betting data)	0.6342500
Logistic Regression (without Betting data)	0.6235833
SVM (with Betting data)	0.5041667
SVM (without Betting data)	0.6447500
Random Forest (with Betting data)	0.5263779
Random Forest (without Betting data)	0.5325261

For 3-Class Classification

Name	Testing Error
Multinomial Regression (with Betting data)	0.5005000
Multinomial Regression (without Betting data)	0.4871667
SVM (with Betting data)	0.5065833
SVM (without Betting data)	0.5046667
Random Forest (with Betting data)	0.5277557
Random Forest (without Betting data)	0.5095616
Ensemble Method (with Betting data)	0.4471786
Ensemble Method (without Betting data)	0.4614107

Statistical Tests

We performed a statistical test (the t-test) to examine whether the mean accuracies are statistically significantly higher than the accuracies of other models.

For Binary Classification

Best Model: *SVM without Betting Data*

Compared Model	T-test P-value
Logistic Regression (with Betting data)	0.0504
Logistic Regression (without Betting data)	0.001147
SVM (with Betting data)	<2.2e-16
Random Forest (with Betting data)	<2.2e-16
Random Forest (without Betting data)	<2.2e-16

For 3-Class Classification

Best Model: *Random Forest with Betting Data*

Compared Model	T-test P-value
Multinomial Regression (with Betting data)	3.793e-07
Multinomial Regression (without Betting data)	1.148e-12
SVM (with Betting data)	9.504e-07
SVM (without Betting data)	4.868e-08
Random Forest (without Betting data)	<2.2e-16
Ensemble Method (with Betting data)	<2.2e-16
Ensemble Method (without Betting data)	<2.2e-16

For the 2-Class classification problem, the SVM model without the betting data had a statistically significantly better accuracy than the other models at the 5% significance level, except for the Logistic Regression model with betting data (p-value: 0.0504).

For the 3-Class classification problem, the Random Forest model with the betting data performed statistically significantly better than all the other models predicting outcome. All p-values were very small, with the largest p-value amongst the comparisons being 9.504e-07.

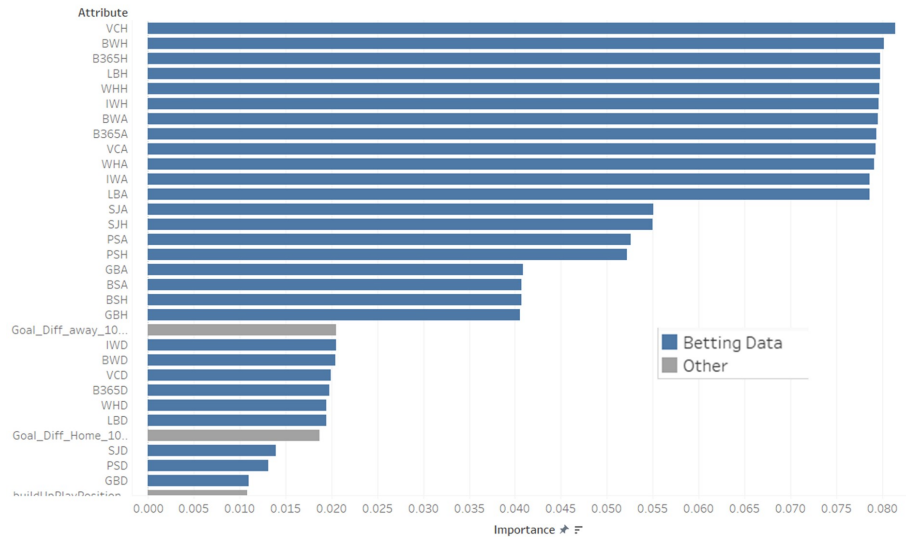
Discussion

The best models for both the binary and the 3-class classification were tuned using appropriate methods.

For the binary classification, the SVM model's decision boundary was tuned using grid search. Different kernels were tried. The radial kernel gave us the best results.

For the logistic regression model some interaction terms were created and feature selection done. It is interesting to note that goal differentials were consistently features that seemed to have a lot of signal in all the models. Style of play variables seemed to matter less which is interesting because chances created or attacking play are not as good predictors as one might expect, conversion of chances and not conceding goals being equally important in soccer.

The random forest prediction result on the binary classification was quite surprising as it did barely better than the home win baseline on the out of sample prediction while doing very well on the in sample error. Our initial thought was that since it is entirely possible that several of the variables are highly correlated amongst each other, and thus would make each decision tree less effective since they may split on similar information several times, i.e. splitting on correlated variables. In order to try and improve our prediction, we selected a subset of variables through looking at each variable's correlation against each other and its information gain ratio/entropy.



The variables with the highest information gain ratio were the betting odds data, as these were most likely fine-tuned by the betting companies. Amongst the non betting data, as shown in the figure above, the highest were actually the goal differential of the past 10 games. While this may sound logical, since the more goals you score the better chance you have of winning, one may expect certain variables such as shots on goal or shots creation would be more relevant.

However even with the trimming of the variables and the tuning of different random forests parameters e.g. number of trees or minimum leaf node size, we were only able to marginally improve the prediction by a few points.

For the 3-class classification however the random forest performed the best.

It is interesting to note that parametric models performed better on the binary classification but a non-parametric method did better on the multi-class problem.

Conclusions

Our model outperforms our baseline analysis by a good margin and provides insights into important team attributes. Our model does not beat the professional betting companies odds, but it is interesting to note that they underestimate the probability of a draw compared to real life data. Ours does better there. Our 3 class model also had results comparable to the top voted Kaggle project (using all leagues + betting data) that saw results of 0.5538.

Competitive sports are a notoriously difficult area to model. The day to day variation of a players performance cannot be taken into account. The models here also do not take into account player data which could make a difference to our accuracy. In soccer it is also interesting to see not just the line up but also the specific strategy teams use against other teams. The data we have here was the overall team style of play related since it was from gaming data. In reality, teams adjust to the other teams strength and differences and vary the number of forwards or defenders they play on a given day. Substitutions during the match as well as red cards, corners and penalties awarded make a big difference to the match outcome. These are all factors which can be taken into account to advance this project's effort further.

Appendices

Tables

Table 1: Win/Away/Draw All Leagues

League	Home	Away	Draw	Total	Home Fraction	Away Fraction	Draw Fraction
Belgium Jupiler League	810	493	425	1728	0.469	0.285	0.246
England Premier League	1390	867	783	3040	0.457	0.285	0.258
France Ligue 1	1359	822	859	3040	0.447	0.27	0.283
Germany 1. Bundesliga	1107	744	597	2448	0.452	0.304	0.244
Italy Serie A	1407	814	796	3017	0.466	0.27	0.264
Netherlands Eredivisie	1171	696	581	2448	0.478	0.284	0.237
Poland Ekstraklasa	870	525	525	1920	0.453	0.273	0.273
Portugal Liga ZON Sagres	908	611	533	2052	0.442	0.298	0.26
Scotland Premier League	760	617	447	1824	0.417	0.338	0.245
Spain LIGA BBVA	1485	851	704	3040	0.488	0.28	0.232
Switzerland Super League	650	426	346	1422	0.457	0.3	0.243

Table 2: Win/Away/Draw Stats All Leagues

Home Fraction	Away Fraction	Draw Fraction	
High	0.488	0.338	0.283
Low	0.417	0.27	0.232
Average	0.459	0.287	0.254

Table 3: Win/Away/Draw EPL All Seasons

Season	Home	Away	Draw	Total	Home Fraction	Away Fraction	Draw Fraction
2008/2009	173	110	97	380	0.455	0.289	0.255
2009/2010	193	91	96	380	0.508	0.239	0.253
2010/2011	179	90	111	380	0.471	0.237	0.292
2011/2012	171	116	93	380	0.45	0.305	0.245
2012/2013	166	106	108	380	0.437	0.279	0.284
2013/2014	179	123	78	380	0.471	0.324	0.205
2014/2015	172	115	93	380	0.453	0.303	0.245
2015/2016	157	116	107	380	0.413	0.305	0.282

Table 4: Win/Away/Draw Stats EPL All Seasons

Home Fraction	Away Fraction	Draw Fraction	
High	0.508	0.324	0.292
Low	0.413	0.237	0.205
Average	0.457	0.285	0.258

Table 5: Oddsmakers EPL Prediction Accuracy

Correct Predictions	Total Predictions	Accuracy
13676	25874	0.52856

Table 6: 2016 EPL Final League Table

Team	Wins	Losses	Draws	Goal Differential	Points
LEI	23	3	12	32	81
ARS	20	7	11	29	71
TOT	19	6	13	34	70
MCI	19	10	9	30	66
MUN	19	10	9	14	66
SOU	18	11	9	18	63
WHU	16	8	14	14	62
LIV	16	10	12	13	60
STK	14	15	9	-14	51
CHE	12	12	14	6	50
SWA	12	15	11	-10	47
WAT	12	17	9	-10	45
EVE	11	13	14	4	47
BOU	11	18	9	-22	42
CRY	11	18	9	-12	42
WBA	10	15	13	-14	43
SUN	9	17	12	-14	39
NEW	9	19	10	-21	37
NOR	9	22	7	-28	34
AVL	3	27	8	-49	17

Table 7: Goal Differential EPL Prediction Accuracy

Season	1 Game	2 Game	3 Game	5 Game	10 Game
2008/09	0.46	0.45	0.48	0.46	0.45
2009/10	0.46	0.44	0.45	0.49	0.48
2010/11	0.41	0.39	0.46	0.45	0.46
2011/12	0.4	0.43	0.43	0.41	0.44
2012/13	0.38	0.4	0.42	0.43	0.42
2013/14	0.43	0.43	0.43	0.43	0.42
2014/15	0.37	0.44	0.46	0.43	0.41
2015/16	0.36	0.39	0.39	0.41	0.42
Average	0.41	0.42	0.44	0.44	0.44

Visualizations

SQL

Table 1

```
SELECT league,
       sum(hm) AS home_total,
       sum(awy) AS away_total,
       sum(drw) AS draw_total
FROM (SELECT l.name AS league,
             CASE WHEN m.home_team_goal > m.away_team_goal THEN 1 ELSE 0 END AS hm,
             CASE WHEN m.home_team_goal < m.away_team_goal THEN 1 ELSE 0 END AS awy,
             CASE WHEN m.home_team_goal = m.away_team_goal THEN 1 ELSE 0 END AS drw
      FROM Match m,
           League l
      WHERE m.league_id = l.id)
GROUP BY league
ORDER BY league;
```

Table 3

```
SELECT season,
       sum(hm) AS home_total,
       sum(awy) AS away_total,
       sum(drw) AS draw_total
FROM (SELECT m.season,
             CASE WHEN m.home_team_goal > m.away_team_goal THEN 1 ELSE 0 END AS hm,
             CASE WHEN m.home_team_goal < m.away_team_goal THEN 1 ELSE 0 END AS awy,
             CASE WHEN m.home_team_goal = m.away_team_goal THEN 1 ELSE 0 END AS drw
      FROM Match m
      WHERE m.league_id = 1729)
GROUP BY season
ORDER BY season;
```

Table 5

```
SELECT sum(correct),
       count(*)
FROM (SELECT CASE WHEN ((B365H < B365D) AND (B365H < B365A) AND
                       (home_team_goal > away_team_goal)) OR
                     ((B365A < B365D) AND (B365A < B365H) AND
                       (away_team_goal > home_team_goal)) OR
                     ((B365D < B365A) AND (B365D < B365H) AND
                       (home_team_goal = away_team_goal))
          THEN 1 ELSE 0 END AS correct
      FROM Match
      WHERE league_id = 1729
          AND B365H IS NOT NULL AND B365A IS NOT NULL AND B365D IS NOT NULL
      UNION ALL
```

```

SELECT CASE WHEN ((BWH < BWD) AND (BWH < BWA) AND
    (home_team_goal > away_team_goal)) OR
    ((BWA < BWD) AND (BWA < BWH) AND
    (away_team_goal > home_team_goal)) OR
    ((BWD < BWA) AND (BWD < BWH) AND
    (home_team_goal = away_team_goal))
    THEN 1 ELSE 0 END AS correct
FROM Match
WHERE league_id = 1729
    AND BWH IS NOT NULL AND BWA IS NOT NULL AND BWD IS NOT NULL
UNION ALL
SELECT CASE WHEN ((IWH < IWD) AND (IWH < IWA) AND
    (home_team_goal > away_team_goal)) OR
    ((IWA < IWD) AND (IWA < IWH) AND
    (away_team_goal > home_team_goal)) OR
    ((IWD < IWA) AND (IWD < IWH) AND
    (home_team_goal = away_team_goal))
    THEN 1 ELSE 0 END AS correct
FROM Match
WHERE league_id = 1729
    AND IWH IS NOT NULL AND IWA IS NOT NULL AND IWD IS NOT NULL
UNION ALL
SELECT CASE WHEN ((LBH < LBD) AND (LBH < LBA) AND
    (home_team_goal > away_team_goal)) OR
    ((LBA < LBD) AND (LBA < LBH) AND
    (away_team_goal > home_team_goal)) OR
    ((LBD < LBA) AND (LBD < LBH) AND
    (home_team_goal = away_team_goal))
    THEN 1 ELSE 0 END AS correct
FROM Match
WHERE league_id = 1729
    AND LBH IS NOT NULL AND LBA IS NOT NULL AND LBD IS NOT NULL
UNION ALL
SELECT CASE WHEN ((PSH < PSD) AND (PSH < PSA) AND
    (home_team_goal > away_team_goal)) OR
    ((PSA < PSD) AND (PSA < PSH) AND
    (away_team_goal > home_team_goal)) OR
    ((PSD < PSA) AND (PSD < PSH) AND
    (home_team_goal = away_team_goal))
    THEN 1 ELSE 0 END AS correct
FROM Match
WHERE league_id = 1729
    AND PSH IS NOT NULL AND PSA IS NOT NULL AND PSD IS NOT NULL
UNION ALL
SELECT CASE WHEN ((WHH < WHD) AND (WHH < WHA) AND
    (home_team_goal > away_team_goal)) OR
    ((WHA < WHD) AND (WHA < WHH) AND
    (away_team_goal > home_team_goal)) OR
    ((WHD < WHA) AND (WHD < WHH) AND
    (home_team_goal = away_team_goal))
    THEN 1 ELSE 0 END AS correct
FROM Match
WHERE league_id = 1729

```

```

        AND WHH IS NOT NULL AND WHA IS NOT NULL AND WHD IS NOT NULL
    UNION ALL
    SELECT CASE WHEN ((SJH < SJD) AND (SJH < SJA) AND
        (home_team_goal > away_team_goal)) OR
        ((SJA < SJD) AND (SJA < SJH) AND
        (away_team_goal > home_team_goal)) OR
        ((SJD < SJA) AND (SJD < SJH) AND
        (home_team_goal = away_team_goal))
        THEN 1 ELSE 0 END AS correct
    FROM Match
    WHERE league_id = 1729
    AND SJH IS NOT NULL AND SJA IS NOT NULL AND SJD IS NOT NULL
    UNION ALL
    SELECT CASE WHEN ((VCH < VCD) AND (VCH < VCA) AND
        (home_team_goal > away_team_goal)) OR
        ((VCA < VCD) AND (VCA < VCH) AND
        (away_team_goal > home_team_goal)) OR
        ((VCD < VCA) AND (VCD < VCH) AND
        (home_team_goal = away_team_goal))
        THEN 1 ELSE 0 END AS correct
    FROM Match
    WHERE league_id = 1729
    AND VCH IS NOT NULL AND VCA IS NOT NULL AND VCD IS NOT NULL
    UNION ALL
    SELECT CASE WHEN ((GBH < GBD) AND (GBH < GBA) AND
        (home_team_goal > away_team_goal)) OR
        ((GBA < GBD) AND (GBA < GBH) AND
        (away_team_goal > home_team_goal)) OR
        ((GBD < GBA) AND (GBD < GBH) AND
        (home_team_goal = away_team_goal))
        THEN 1 ELSE 0 END AS correct
    FROM Match
    WHERE league_id = 1729
    AND GBH IS NOT NULL AND GBA IS NOT NULL AND GBD IS NOT NULL
    UNION ALL
    SELECT CASE WHEN ((BSH < BSD) AND (BSH < BSA) AND
        (home_team_goal > away_team_goal)) OR
        ((BSA < BSD) AND (BSA < BSH) AND
        (away_team_goal > home_team_goal)) OR
        ((BSD < BSA) AND (BSD < BSH) AND
        (home_team_goal = away_team_goal))
        THEN 1 ELSE 0 END AS correct
    FROM Match
    WHERE league_id = 1729
    AND BSH IS NOT NULL AND BSA IS NOT NULL AND BSD IS NOT NULL);

```

Table 6

```

SELECT t.team_short_name,
       sum(msq.win)        AS wins,
       sum(msq.loss)       AS losses,
       sum(msq.draw)       AS draws,

```

```

        sum(msq.goal_diff) AS goal_diff_total
FROM (SELECT m.home_team_api_id AS team_id,
        CASE WHEN (m.home_team_goal > m.away_team_goal)
            THEN 1 ELSE 0 END AS win,
        CASE WHEN (m.home_team_goal < m.away_team_goal)
            THEN 1 ELSE 0 END AS loss,
        CASE WHEN (m.home_team_goal = m.away_team_goal)
            THEN 1 ELSE 0 END AS draw,
        m.home_team_goal - m.away_team_goal AS goal_diff
FROM Match m
WHERE m.league_id = 1729
AND m.season = '2015/2016'
UNION ALL
SELECT m.away_team_api_id AS team_id,
        CASE WHEN (m.away_team_goal > m.home_team_goal)
            THEN 1 ELSE 0 END AS win,
        CASE WHEN (m.away_team_goal < m.home_team_goal)
            THEN 1 ELSE 0 END AS loss,
        CASE WHEN (m.away_team_goal = m.home_team_goal)
            THEN 1 ELSE 0 END AS draw,
        m.away_team_goal - m.home_team_goal AS goal_diff
FROM Match m
WHERE m.league_id = 1729
AND m.season = '2015/2016') msq,
Team t
WHERE msq.team_id = t.team_api_id
GROUP BY t.team_short_name
ORDER BY 2 DESC, 4 DESC;

```

Goal Differential

```

-- Create a table like the Match table, but having
-- calculated information about wins, losses,
-- and goal differential at various time horizons.
--

```

```

CREATE TABLE Match_Goal_Differential (
    league_id      INTEGER,
    season         TEXT,
    stage          INTEGER,
    timeline_id    INTEGER,
    match_api_id   INTEGER,
    home_team_api_id  INTEGER,
    home_team_short_name TEXT,
    away_team_api_id  INTEGER,
    away_team_short_name TEXT,
    home_team_goal  INTEGER,
    away_team_goal  INTEGER,
    home_win        INTEGER,
    home_loss       INTEGER,
    home_draw       INTEGER,
    away_win        INTEGER,
    away_loss       INTEGER,

```

```

away_draw          INTEGER,
home_goal_diff_0   INTEGER,
away_goal_diff_0   INTEGER,
home_goal_diff_1   INTEGER,
away_goal_diff_1   INTEGER,
home_goal_diff_2   INTEGER,
away_goal_diff_2   INTEGER,
home_goal_diff_3   INTEGER,
away_goal_diff_3   INTEGER,
home_goal_diff_5   INTEGER,
away_goal_diff_5   INTEGER,
home_goal_diff_10  INTEGER,
away_goal_diff_10  INTEGER,
home_goal_diff_total INTEGER,
away_goal_diff_total INTEGER
);

--
-- Create our data.
--
INSERT INTO Match_Goal_Differential
    (league_id,
     season,
     stage,
     timeline_id,
     match_api_id,
     home_team_api_id,
     home_team_short_name,
     away_team_api_id,
     away_team_short_name,
     home_team_goal,
     away_team_goal,
     home_win,
     home_loss,
     home_draw,
     away_win,
     away_loss,
     away_draw,
     home_goal_diff_0,
     away_goal_diff_0,
     home_goal_diff_1,
     away_goal_diff_1,
     home_goal_diff_2,
     away_goal_diff_2,
     home_goal_diff_3,
     away_goal_diff_3,
     home_goal_diff_5,
     away_goal_diff_5,
     home_goal_diff_10,
     away_goal_diff_10,
     home_goal_diff_total,
     away_goal_diff_total)
SELECT m.league_id,

```



```

away_team_api_id,
away_team_short_name,
home_team_goal,
away_team_goal,
home_win,
home_loss,
home_draw,
away_win,
away_loss,
away_draw,
home_goal_diff_0,
away_goal_diff_0,
home_goal_diff_1,
away_goal_diff_1,
home_goal_diff_2,
away_goal_diff_2,
home_goal_diff_3,
away_goal_diff_3,
home_goal_diff_5,
away_goal_diff_5,
home_goal_diff_10,
away_goal_diff_10,
home_goal_diff_total,
away_goal_diff_total
FROM Match_Goal_Differential
WHERE league_id = 1729
ORDER BY timeline_id, match_api_id;

```

R Code

Goal Differential

```

# This code uses as input the goal differential file output from the SQLite database.
goal_diff <- read.table(file="match_goal_differential.csv",sep = ",",header=TRUE);

seasons <- goal_diff$season
s08 <- which( seasons == "2008/2009" )
s09 <- which( seasons == "2009/2010" )
s10 <- which( seasons == "2010/2011" )
s11 <- which( seasons == "2011/2012" )
s12 <- which( seasons == "2012/2013" )
s13 <- which( seasons == "2013/2014" )
s14 <- which( seasons == "2014/2015" )
s15 <- which( seasons == "2015/2016" )

gd08 <- goal_diff[s08,]
gd09 <- goal_diff[s09,]
gd10 <- goal_diff[s10,]
gd11 <- goal_diff[s11,]
gd12 <- goal_diff[s12,]
gd13 <- goal_diff[s13,]
gd14 <- goal_diff[s14,]

```

```

gd15 <- goal_diff[s15,]

#
# Information from the database
# Teams by season
#
t08 <- c("ARS","AVL","BLB","BOL","CHE","EVE","FUL","HUL","LIV","MCI","MID","MUN","NEW","POR",
"STK","SUN","TOT","WBA","WHU","WIG")
t09 <- c("ARS","AVL","BIR","BLB","BOL","BUR","CHE","EVE","FUL","HUL","LIV","MCI","MUN","POR",
"STK","SUN","TOT","WHU","WIG","WOL")
t10 <- c("ARS","AVL","BIR","BLA","BLB","BOL","CHE","EVE","FUL","LIV","MCI","MUN","NEW","STK",
"SUN","TOT","WBA","WHU","WIG","WOL")
t11 <- c("ARS","AVL","BLB","BOL","CHE","EVE","FUL","LIV","MCI","MUN","NEW","NOR","QPR","STK",
"SUN","SWA","TOT","WBA","WIG","WOL")
t12 <- c("ARS","AVL","CHE","EVE","FUL","LIV","MCI","MUN","NEW","NOR","QPR","REA","SOU","STK",
"SUN","SWA","TOT","WBA","WHU","WIG")
t13 <- c("ARS","AVL","CAR","CHE","CRY","EVE","FUL","HUL","LIV","MCI","MUN","NEW","NOR","SOU",
"STK","SUN","SWA","TOT","WBA","WHU")
t14 <- c("ARS","AVL","BUR","CHE","CRY","EVE","HUL","LEI","LIV","MCI","MUN","NEW","QPR","SOU",
"STK","SUN","SWA","TOT","WBA","WHU")
t15 <- c("ARS","AVL","BOU","CHE","CRY","EVE","LEI","LIV","MCI","MUN","NEW","NOR","SOU","STK",
"SUN","SWA","TOT","WAT","WBA","WHU")

# Since we are doing cumulative counts within a season
# it is easier to process the seasons individually.

#
# 2008/2009 season
#
for ( t in t08 ) {

  #
  # Home games for this team
  #
  htsn <- gd08$home_team_short_name
  gmh <- which( htsn == t )

  #
  # Single-game home-team goal differential
  #
  hgd0 <- gd08[gmh,]$home_goal_diff_0

  #
  # Goal diff for last previous game only
  #
  gd08[gmh[2:19],]$home_goal_diff_1 <- hgd0[1:18]

  #
  # Goal diff for last 2 previous games
  #
  gd08[gmh[2],]$home_goal_diff_2 <- hgd0[1]
  for ( i in 1:17 ) {
    gd08[gmh[i+2],]$home_goal_diff_2 <- sum( hgd0[i:(i+1)] )
  }
}

```

```

}

#
# Goal diff for last 3 previous games
#
gd08[gmh[2],]$home_goal_diff_3 <- hgd0[1]
gd08[gmh[3],]$home_goal_diff_3 <- sum( hgd0[1:2] )
for ( i in 1:16 ) {
  gd08[gmh[i+3],]$home_goal_diff_3 <- sum( hgd0[i:(i+2)] )
}

#
# Goal diff for last 5 previous games
#
gd08[gmh[2],]$home_goal_diff_5 <- hgd0[1]
gd08[gmh[3],]$home_goal_diff_5 <- sum( hgd0[1:2] )
gd08[gmh[4],]$home_goal_diff_5 <- sum( hgd0[1:3] )
gd08[gmh[5],]$home_goal_diff_5 <- sum( hgd0[1:4] )
for ( i in 1:14 ) {
  gd08[gmh[i+5],]$home_goal_diff_5 <- sum( hgd0[i:(i+4)] )
}

#
# Goal diff for last 10 previous games
#
gd08[gmh[2],]$home_goal_diff_10 <- hgd0[1]
for ( i in 2:9 ) {
  gd08[gmh[i+1],]$home_goal_diff_10 <- sum( hgd0[1:i] )
}
for ( i in 1:9 ) {
  gd08[gmh[i+10],]$home_goal_diff_10 <- sum( hgd0[i:(i+9)] )
}

#
# Cumulative season goal diff
#
for ( i in 1:18 ) {
  gd08[gmh[i+1],]$home_goal_diff_total <- sum( hgd0[1:i] )
}

#
# Same for this team when away
#
atsn <- gd08$away_team_short_name
gma <- which( atsn == t )
agd0 <- gd08[gma,]$away_goal_diff_0

#
# Goal diff for last previous game only
#
gd08[gma[2:19],]$away_goal_diff_1 <- agd0[1:18]

```

```

#
# Goal diff for last 2 previous games
#
gd08[gma[2],]$away_goal_diff_2 <- agd0[1]
for ( i in 1:17 ) {
  gd08[gma[i+2],]$away_goal_diff_2 <- sum( agd0[i:(i+1)] )
}

#
# Goal diff for last 3 previous games
#
gd08[gma[2],]$away_goal_diff_3 <- agd0[1]
gd08[gma[3],]$away_goal_diff_3 <- sum( agd0[1:2] )
for ( i in 1:16 ) {
  gd08[gma[i+3],]$away_goal_diff_3 <- sum( agd0[i:(i+2)] )
}

#
# Goal diff for last 5 previous games
#
gd08[gma[2],]$away_goal_diff_5 <- agd0[1]
gd08[gma[3],]$away_goal_diff_5 <- sum( agd0[1:2] )
gd08[gma[4],]$away_goal_diff_5 <- sum( agd0[1:3] )
gd08[gma[5],]$away_goal_diff_5 <- sum( agd0[1:4] )
for ( i in 1:14 ) {
  gd08[gma[i+5],]$away_goal_diff_5 <- sum( agd0[i:(i+4)] )
}

#
# Goal diff for last 10 previous games
#
gd08[gma[2],]$away_goal_diff_10 <- agd0[1]
for ( i in 2:9 ) {
  gd08[gma[i+1],]$away_goal_diff_10 <- sum( agd0[1:i] )
}
for ( i in 1:9 ) {
  gd08[gma[i+10],]$away_goal_diff_10 <- sum( agd0[i:(i+9)] )
}

#
# Cumulative season goal diff
#
for ( i in 1:18 ) {
  gd08[gma[i+1],]$away_goal_diff_total <- sum( agd0[1:i] )
}
}

#
# A similar code was used to process the remaining seasons. There were only 2
# changes/substitutions:
#   1) The main loop variable would represent the teams for a different season.
#   For example, instead of using the teams in t08 for the 2008/2009 season,
#   we would use the teams in t12 for the 2012/2013 season.

```

```

# 2) The goal differential season information changes. This is held in the
# data frame variables named gdXX. So instead of using gd08 as shown in
# the R code above for the 2008/2009 season, we would use gd12 for the
# calculations for the 2012/2013 season.
#

# Save calculations
write.table( gd08, "goal_differential_info.csv", append = FALSE, sep = ",",
             row.names=FALSE, col.names=TRUE );
write.table( gd09, "goal_differential_info.csv", append = TRUE, sep = ",",
             row.names=FALSE, col.names=FALSE );
write.table( gd10, "goal_differential_info.csv", append = TRUE, sep = ",",
             row.names=FALSE, col.names=FALSE );
write.table( gd11, "goal_differential_info.csv", append = TRUE, sep = ",",
             row.names=FALSE, col.names=FALSE );
write.table( gd12, "goal_differential_info.csv", append = TRUE, sep = ",",
             row.names=FALSE, col.names=FALSE );
write.table( gd13, "goal_differential_info.csv", append = TRUE, sep = ",",
             row.names=FALSE, col.names=FALSE );
write.table( gd14, "goal_differential_info.csv", append = TRUE, sep = ",",
             row.names=FALSE, col.names=FALSE );
write.table( gd15, "goal_differential_info.csv", append = TRUE, sep = ",",
             row.names=FALSE, col.names=FALSE );

```

SVM and Logistic/Multinomial Regression

```

setwd('/Users/Stevenstuff/Downloads')
#setwd("/Users/Stevenstuff/soccerdatamining")
soccer <- read.csv('Soccerdata.csv')

# fill in blanks with NA's
soccer[soccer==""] <- NA

# betting data as numeric instead of factor
for(i in rep(20:49)) {
  soccer[,i] <- as.numeric(soccer[,i])
}

# remove "diff" columns: time since last update
soccer <- subset(soccer, select=-c(rank_Update_diff1, rank_Update_diff11,
                                   Update_diff1, Update_diff11))

# buildUpPlayDribbling_home and buildUpPlayDribbling1_away has 19088 NA's
soccer <- subset(soccer, select=-c(buildUpPlayDribbling_home, buildUpPlayDribbling1_away))

# remove duplicate columns
soccer <- subset(soccer, select=-c(country_id1, league_id, team_api_id,
                                   team_api_id1, team_api_id1_away_away,
                                   id3_home, id4, team_fifa_api_id2_home,
                                   team_api_id2_home, team_fifa_api_id3_away,
                                   team_api_id3_away))

```

```

# multinomial regression: 1 if home wins, 2 if home loses, 3 if tie
soccer$outcome <- ifelse(soccer$home_team_goal-soccer$away_team_goal > 0, 1,
                        ifelse(soccer$home_team_goal-soccer$away_team_goal<0, 2, 3))
soccer$outcome <- as.factor(soccer$outcome)
soccer$win <- as.factor(ifelse(soccer$home_team_goal-soccer$away_team_goal > 0, 1, 0))
soccer$lose <- as.factor(ifelse(soccer$home_team_goal-soccer$away_team_goal < 0, 1, 0))
soccer$tie <- as.factor(ifelse(soccer$home_team_goal-soccer$away_team_goal == 0, 1, 0))

# remove useless columns
soccer <- subset(soccer, select=-c(id, date, match_api_id, home_team_goal,
                                   away_team_goal, Goal_Diff_Home,
                                   Goal_Diff_away, id1_home_home, team_fifa_api_id_home_home,
                                   team_long_name_home_home, team_short_name_home_home,
                                   id2_away_away,team_fifa_api_id1_away_away,team_long_name1_away_away,
                                   team_short_name1_away_away, Country_Name_home,League_name,id5,date1_h
                                   id6_away,date2_away))

# remove columns with too many factors (messes up classification)
soccer <- subset(soccer, select=-c(home_team_api_id, away_team_api_id))

# factorize columns
names<-c("country_id", "season", "stage")
soccer[,names] <- lapply(soccer[,names], factor)

# dataset with betting data removed
library(dplyr)
nan_columns = colnames(soccer)[colSums(is.na(soccer)) > 0]
soccer_nobet <- soccer[, !colnames(soccer) %in% nan_columns]
write.csv(soccer_nobet, 'soccer_nobet.csv', row.names=FALSE)

# dataset with NA rows removed
soccer_nona <- na.omit(soccer)
write.csv(soccer_nona, 'soccer_nona.csv', row.names=FALSE)

write.csv(soccer, 'soccer_clean.csv', row.names=FALSE)

library(nnet)
# doesn't work
test <- multinom(outcome ~ country_id+season+stage+Goal_Diff_Home_10.Games+Goal_Diff_away_10.Games+
                  B365H+B365D+B365A+BWH+BWD+BWA+IWH+IWD+IWA+LBH+LBD+LBA+PSH+PSD+PSA+
                  WHH+WHD+WHA+SJH+SJD+SJA+VCH+VCD+VCA+GBH+GBD+GBA+BSH+BSD+BSA+
                  buildUpPlaySpeed_home+buildUpPlaySpeedClass_home+
                  buildUpPlayDribblingClass_home+buildUpPlayPassing_home+buildUpPlayPassingClass_home+
                  buildUpPlayPositioningClass_home+chanceCreationPassing_home+chanceCreationPassingClass_home+
                  chanceCreationCrossing_home+chanceCreationCrossingClass_home+chanceCreationShooting_1
                  chanceCreationShootingClass_home+chanceCreationPositioningClass_home+defencePressure
                  defencePressureClass_home+defenceAggression_home+defenceAggressionClass_home+
                  defenceTeamWidth_home+defenceTeamWidthClass_home+defenceDefenderLineClass_home+
                  buildUpPlaySpeed1_away+buildUpPlaySpeedClass1_away+
                  buildUpPlayDribblingClass1_away+buildUpPlayPassing1_away+buildUpPlayPassingClass1_away+
                  buildUpPlayPositioningClass1_away+chanceCreationPassing1_away+chanceCreationPassingClass1_away+
                  chanceCreationCrossing1_away+chanceCreationCrossingClass1_away+chanceCreationShooting

```

```

        chanceCreationShootingClass1_away+chanceCreationPositioningClass1_away+defencePressureClass1_away+defenceAggressionClass1_away+
        defenceTeamWidth1_away+defenceTeamWidthClass1_away+defenceDefenderLineClass1_away, data=soccer_nobet)
# multinomial model
model <- multinom(outcome ~ country_id+season+stage+Goal_Diff_Home_10.Games+Goal_Diff_away_10.Games+
    buildUpPlaySpeed_home+buildUpPlaySpeedClass_home+
    buildUpPlayDribblingClass_home+buildUpPlayPassing_home+buildUpPlayPassingClass_home+
    buildUpPlayPositioningClass_home+chanceCreationPassing_home+chanceCreationPassingClass_home+
    chanceCreationCrossing_home+chanceCreationCrossingClass_home+chanceCreationShooting_home+
    chanceCreationShootingClass_home+chanceCreationPositioningClass_home+defencePressure_home+
    defencePressureClass_home+defenceAggression_home+defenceAggressionClass_home+
    defenceTeamWidth_home+defenceTeamWidthClass_home+defenceDefenderLineClass_home+
    buildUpPlaySpeed1_away+buildUpPlaySpeedClass1_away+
    buildUpPlayDribblingClass1_away+buildUpPlayPassing1_away+buildUpPlayPassingClass1_away+
    buildUpPlayPositioningClass1_away+chanceCreationPassing1_away+chanceCreationPassingClass1_away+
    chanceCreationCrossing1_away+chanceCreationCrossingClass1_away+chanceCreationShooting1_away+
    chanceCreationShootingClass1_away+chanceCreationPositioningClass1_away+defencePressure1_away+
    defencePressureClass1_away+defenceAggression1_away+defenceAggressionClass1_away+
    defenceTeamWidth1_away+defenceTeamWidthClass1_away+defenceDefenderLineClass1_away, data=soccer_nobet)

### save the TE values for all models in all $B=100$ loops
B= 100;          ### number of loops
TEALL = NULL;    ### Final TE values
n = dim(soccer_nobet)[1]
n1 = round(n/10)

for (b in 1:B){
    ### randomly select 25 observations as testing data in each loop
    flag <- sort(sample(1:n, n1));
    train <- soccer_nobet[-flag,];
    test  <- soccer_nobet[flag,];
    wins <- glm(win ~ country_id+season+stage+Goal_Diff_Home_10.Games+Goal_Diff_away_10.Games+
        buildUpPlaySpeed_home+buildUpPlaySpeedClass_home+
        buildUpPlayDribblingClass_home+buildUpPlayPassing_home+buildUpPlayPassingClass_home+
        buildUpPlayPositioningClass_home+chanceCreationPassing_home+chanceCreationPassingClass_home+
        chanceCreationCrossing_home+chanceCreationCrossingClass_home+chanceCreationShooting_home+
        chanceCreationShootingClass_home+chanceCreationPositioningClass_home+defencePressure_home+
        defencePressureClass_home+defenceAggression_home+defenceAggressionClass_home+
        defenceTeamWidth_home+defenceTeamWidthClass_home+defenceDefenderLineClass_home+
        buildUpPlaySpeed1_away+buildUpPlaySpeedClass1_away+
        buildUpPlayDribblingClass1_away+buildUpPlayPassing1_away+buildUpPlayPassingClass1_away+
        buildUpPlayPositioningClass1_away+chanceCreationPassing1_away+chanceCreationPassingClass1_away+
        chanceCreationCrossing1_away+chanceCreationCrossingClass1_away+chanceCreationShooting1_away+
        chanceCreationShootingClass1_away+chanceCreationPositioningClass1_away+defencePressure1_away+
        defencePressureClass1_away+defenceAggression1_away+defenceAggressionClass1_away+
        defenceTeamWidth1_away+defenceTeamWidthClass1_away+defenceDefenderLineClass1_away,
        family = binomial(link="logit"))
    loss <- glm(lose ~ country_id+season+stage+Goal_Diff_Home_10.Games+Goal_Diff_away_10.Games+
        buildUpPlaySpeed_home+buildUpPlaySpeedClass_home+
        buildUpPlayDribblingClass_home+buildUpPlayPassing_home+buildUpPlayPassingClass_home+
        buildUpPlayPositioningClass_home+chanceCreationPassing_home+chanceCreationPassingClass_home+
        chanceCreationCrossing_home+chanceCreationCrossingClass_home+chanceCreationShooting_home+
        chanceCreationShootingClass_home+chanceCreationPositioningClass_home+defencePressure_home+
        defencePressureClass_home+defenceAggression_home+defenceAggressionClass_home+
        defenceTeamWidth_home+defenceTeamWidthClass_home+defenceDefenderLineClass_home+
        buildUpPlaySpeed1_away+buildUpPlaySpeedClass1_away+
        buildUpPlayDribblingClass1_away+buildUpPlayPassing1_away+buildUpPlayPassingClass1_away+
        buildUpPlayPositioningClass1_away+chanceCreationPassing1_away+chanceCreationPassingClass1_away+
        chanceCreationCrossing1_away+chanceCreationCrossingClass1_away+chanceCreationShooting1_away+
        chanceCreationShootingClass1_away+chanceCreationPositioningClass1_away+defencePressure1_away+
        defencePressureClass1_away+defenceAggression1_away+defenceAggressionClass1_away+
        defenceTeamWidth1_away+defenceTeamWidthClass1_away+defenceDefenderLineClass1_away,
        family = binomial(link="logit"))
}

```

```

chanceCreationShootingClass_home+chanceCreationPositioningClass_home+defencePressure_home+
defencePressureClass_home+defenceAggression_home+defenceAggressionClass_home+
defenceTeamWidth_home+defenceTeamWidthClass_home+defenceDefenderLineClass_home+
buildUpPlaySpeed1_away+buildUpPlaySpeedClass1_away+
buildUpPlayDribblingClass1_away+buildUpPlayPassing1_away+buildUpPlayPassingClass1_away+
buildUpPlayPositioningClass1_away+chanceCreationPassing1_away+chanceCreationPassingClass1_away+
chanceCreationCrossing1_away+chanceCreationCrossingClass1_away+chanceCreationShooting1_away+
chanceCreationShootingClass1_away+chanceCreationPositioningClass1_away+defencePressure1_away+
defencePressureClass1_away+defenceAggression1_away+defenceAggressionClass1_away+
defenceTeamWidth1_away+defenceTeamWidthClass1_away+defenceDefenderLineClass1_away, data=
family = binomial(link="logit"))
tie <- glm(tie ~ country_id+season+stage+Goal_Diff_Home_10.Games+Goal_Diff_away_10.Games+
buildUpPlaySpeed_home+buildUpPlaySpeedClass_home+
buildUpPlayDribblingClass_home+buildUpPlayPassing_home+buildUpPlayPassingClass_home+
buildUpPlayPositioningClass_home+chanceCreationPassing_home+chanceCreationPassingClass1_away+
chanceCreationCrossing_home+chanceCreationCrossingClass_home+chanceCreationShooting_home+
chanceCreationShootingClass_home+chanceCreationPositioningClass_home+defencePressure_home+
defencePressureClass_home+defenceAggression_home+defenceAggressionClass_home+
defenceTeamWidth_home+defenceTeamWidthClass_home+defenceDefenderLineClass_home+
buildUpPlaySpeed1_away+buildUpPlaySpeedClass1_away+
buildUpPlayDribblingClass1_away+buildUpPlayPassing1_away+buildUpPlayPassingClass1_away+
buildUpPlayPositioningClass1_away+chanceCreationPassing1_away+chanceCreationPassingClass1_away+
chanceCreationCrossing1_away+chanceCreationCrossingClass1_away+chanceCreationShooting1_away+
chanceCreationShootingClass1_away+chanceCreationPositioningClass1_away+defencePressure1_away+
defencePressureClass1_away+defenceAggression1_away+defenceAggressionClass1_away+
defenceTeamWidth1_away+defenceTeamWidthClass1_away+defenceDefenderLineClass1_away, data=
family = binomial(link="logit"))
#Testmod <- mean(predict(model,test[,1:8]) != test$outcome)
predict_wins <- predict(wins,test[,1:45], type="response")
predict_wins <- ifelse(predict_wins > win_thresh,1,0)
Testwins <- mean(predict_wins == test$outcome)
# losses
predict_losses <- predict(loss,test[,1:45], type="response")
predict_losses <- ifelse(predict_losses > loss_thresh,1,0)
Testlosses_bench <- mean(predict_losses == test$outcome)
# ties
predict_ties <- predict(tie,test[,1:45], type="response")
predict_ties <- ifelse(predict_ties > tie_thresh,1,0)
Test_tie <- mean(predict_ties == test$outcome)
TEALL = rbind(TEALL, cbind(Testmod))
}
# decision boundary for wins
summary(soccer_nobet$win)
perc = 8835/(10515+8835) #~0.2899225
win_thresh <- quantile(predict(wins,test[,1:45], type="response"), perc)

# determining decision boundary for loss
summary(soccer_nobet$lose)
perc = 5610/(5610+13740) #~0.2899225
loss_thresh <- quantile(predict(loss,test[,1:45], type="response"), perc)

# decision boundary for ties
summary(soccer_nobet$tie)

```



```
perc = 4905/(4905+14445) #~0.253
tie_thresh <- quantile(predict(tie,test[,1:45], type="response"), perc)
```

Random Forest

```
setwd("C:\\Users\\Parit\\Desktop\\Classwork\\Georgia Tech\\Spring 2017 work\\ISyE 7406 - Data Mining and Analytics")
data <- read.csv('soccer_4_14.csv',header=TRUE)
y <- data$home_team_goal - data$away_team_goal
y[y>0] <- 1
y[y<0] <- -1
data1<-data
data[,c('home_team_goal','away_team_goal','Goal_Diff_away','Goal_Diff_Home','outcome',
        'id','country_id','match_api_id','home_team_api_id')<-NULL]
data[,c('home_team_goal','away_team_goal','Goal_Diff_away','Goal_Diff_Home','outcome',
        'id','country_id','match_api_id','home_team_api_id','away_team_api_id','id1_home_home',
        'team_fifa_api_id_home_home','id2_away_away','team_api_id1_away_away',
        'team_fifa_api_id1_away_away','id5','date','team_short_name_home_home','team_short_name1_away_away',
        'id6_away','team_api_id3_away','date2_away','date1_home','outcome',
        'team_long_name1_away_away','team_long_name_home_home')<-NULL]
datanobet <- data
datanobet <- datanobet[""]
filternull<-data[, apply(data, 2, function(x) !any(is.na(x)))]
is.fact <- sapply(filternull, is.factor)
cat<- filternull[is.fact]
# filternull$Goal_Diff_Home_10.Games <- as.numeric(filternull$Goal_Diff_Home_10.Games)
# filternull$shoton <- as.numeric(filternull$shoton)
# filternull$shotoff <- as.numeric(filternull$shotoff)
# filternull$foulcommit <- as.numeric(filternull$foulcommit)
# filternull$card <- as.numeric(filternull$card)
# filternull$possession <- as.numeric(filternull$possession)
# filternull$corner <- as.numeric(filternull$corner)
# filternull$cross <- as.numeric(filternull$cross)

filternull$stage <- as.factor(filternull$stage)
filternull$country_id <- as.factor(filternull$country_id)

y<-as.factor(y)
filternull['y']<-as.factor(y)
data['y']<-as.factor(y)
epl <-filternull[filternull$League_name == "England Premier League",]
yepl <-y[filternull$League_name == "England Premier League"]

### Train test partitioning
n = dim(data)[1]; ### total number of observations
n1 = round(n/5); ### number of observations randomly selected for testing data### set the random seed
flag = sort(sample(1:n, n1));
train = data[-flag,]; test = data[flag,];
ytr = y[-flag]; yt = y[flag];
trainf = filternull[-flag,]; testf = filternull[flag,];
```

```

ne = dim(epl)[1]; ### total number of observations
ne1 = round(ne/5); ### number of observations randomly selected for testing data### set the random seed
flage = sort(sample(1:ne, ne1));
traine = epl[-flage,]; teste = epl[flage,];
ytre = yepl[-flage]; yte = yepl[flage];

library(randomForest)

fit <- randomForest( y~ .,
                    data=traine,
                    importance=TRUE,
                    ntree=100)
predtrain <- mean(predict(fit,traine) == ytre)
predtest <- mean(predict(fit,teste) == yte)

fit <- randomForest( y ~ Goal_Diff_away_10.Games + Goal_Diff_Home_10.Games + buildUpPlayPositioningClass_away +
                    buildUpPlayPositioningClass1_away + chanceCreationPositioningClass_home +
                    chanceCreationPositioningClass1_away + defenceAggression_home +
                    chanceCreationShooting_home + chanceCreationShootingClass_home +
                    chanceCreationShooting1_away + chanceCreationShootingClass1_away +
                    buildUpPlayPassing_home + chanceCreationPassing_home + chanceCreationPassing1_away +
                    stage + defenceDefenderLineClass_home + buildUpPlayPassingClass_home +
                    chanceCreationCrossing1_away + defenceAggression1_away +
                    Country_Name_home,
                    data=trainf,
                    importance=TRUE,
                    ntree=500)
predtrainf <- mean(predict(fit,trainf) == ytr)
predtestf <- mean(predict(fit,testf) == yt)

library(party)
fitp <- cforest(y~.,
               data = train,
               controls=cforest_unbiased(ntree=100, mtry=3))
Prediction <- mean(predict(fitp, test, OOB=TRUE, type = "response") ==y)

#entropy
library(FSelector)

weights <- information.gain(y~., data)
print(weights)
subset <- cutoff.k(weights, 20)
f <- as.simple.formula(subset, "y")
print(f)

weights <- information.gain(y~., filternull)
print(weights)
subset <- cutoff.k(weights, 20)
f <- as.simple.formula(subset, "y")
print(f)

```

```
write.csv(weights, "dataingfogainnobets.csv")
```

There is no code for our Bradley Terry Model since we used XLSTAT in Microsoft Excel for that analysis. Some of our visualizations were also produced using Tableau, hence we submit no code for it.

Bibliography and Credits

- European Soccer Database

<https://www.kaggle.com/hugomathien/soccer>

- SQLite

<https://www.sqlite.org/index.html>

- R

<https://www.r-project.org/>

<https://cran.r-project.org/doc/manuals/r-patched/R-intro.pdf>

- Tableau

<https://www.tableau.com/>