Predicting English Premier League Games using Machine Learning

Israel Owusu

September 13, 2017

1 Definition

Project Overview

Globally, soccer is the top ranked sport with enormous advertising revenues and the business of soccer is valued at $26 billion annually [6]. Aside from the monetary valuation, the sport has the largest global fan base with a Football confederation on each continent. The sport is popular because it is simple and does not need expensive equipment to play. Even in the United states, soccer viewership has steadily increased over the years, usually boosted by the national Women's and Men's teams performance in the World Cup. Despite, financial scandals in governing bodies and rampant match fixing in some leagues, fans remain loyal to the sport and new fans are joining the fold. With this much money involved and fan support invested, soccer betting and match prediction are a huge business but still a tough problem to solve. In European countries soccer leagues are much more developed and betting is allowed on most games. In the English Premier League (EPL), sports betting is woven into pregame discussions and betting companies advertise during games.

My interest predicting the outcomes of EPL matches is personal. I am an avid fan of Liverpool Football Club, a club with a extensive history of success in English and European soccer. Recent history for the club has been middling with the best performance in the EPL league being a second place finish in the 2013/14 season. An fairly accurate predictive model would provide a preview about Liverpool matches each weekend and make the 38 game season a bit more bearable. Unlike other European soccer leagues dominated by one or two teams, the EPL has a bit more parity between teams. It is not rare to have games where top ranked teams in the EPL lose to or draw with lower ranked teams. This makes predicting EPL match outcomes challenging and a interesting problem. In this work I discuss different ML approaches for predicting the outcomes for EPL matches (a win, loss or draw for the home team)

Each weekend, sports analysts make predictions for each EPL match. Mark Lawrenson, at the BBC, has been predicting games for a almost a decade [7]. The website, FiveThirtyEight.com, a website which reports news and sports through a statistical viewpoint, also has a detailed prediction model. Bayesian Nets and linear prediction approaches have also been used to predict matches for a specific team. The majority of research appears to rely on a combination of models and Monte Carlo simulation models to arrive at probabilistic outputs. A recent prediction analysis from the Financial Times newspaper is based on a model [10] which uses previous performances for individual teams to predict expected goals scored.

Problem Statement

In this project, we trained several machine learning models on matches from the English Premier league in order to predict outcomes (*win*, *lose*, *draw*) for a test set of matches. Multiple machine learning estimators were trained on historical match and team data obtained from a GitHub repository[1]. Given any match and team information, the goal was to create a machine model which is able to predict whether the home team would win, lose or draw the game.

The source dataset was generated via web-scraping prominent soccer websites tracking games and matches. The dataset contained features from both away and home teams in two database tables: the `Teams` and `Team_Attributes`. Each row in the `Team_Attributes` table contains over *20* features/columns with a corresponding index for matching the team with the season. The `Matches` table contains the specific data (e.g. the date, stage and home and away goals). Merging these tables provided a complete dataset for feature and model selection. Given this the number of columns in the merged dataset, the

problem was to use feature selection techniques to identify features that impact the home team match outcome. In this analysis, we also created additional features that could improve the performance and accuracy of the predicted match outcomes.

We assessed multiple machine learning classifiers and predictive algorithms. The selected models were then tuned on a training dataset and subsequently model used to predict the scores for test subset of matches; EPL seasons from 2010 to 2016 were included in this proect.

Model predictions for each estimator were compared with our benchmark predictions: the FiveThirtyEight model, and the prediction of British soccer pundit Mark Lawrenson.

Metrics

This project, primarily, used a weighted F1 score as the main model evaluation metric. This contrasts with the choice of an accuracy metric used in other soccer prediction analyses in literature. An F1 score is appropriate for this multi class problem as it provides a performance measure that accounts for both the precision and the recall of of the derived model performance on individual classes in the dataset; class labels are [*win*, *lose*, *draw*] defined in terms of the outcome for the home team. A class weighted F1 score helps to address the imbalance by weighting the score based on the proportion of each class in the dataset.

An initial exploration of the dataset shows that most of the premier league matches end in a win for the home team. With this imbalance, an accuracy metric alone is insufficient as it would give high scores to any estimator that just predicts the dominant class in the dataset. In this analysis, a Dummy classifier which predicts the dominant class for each match is also fit to the model and its performance is compared to the more rigorous estimators.

In addition, a confusion matrix will also be created for each algorithm and used in comparing the best performing method. The predictions of the benchmark models will also be assessed using the same metric.

2 Analysis

Data Exploration

The data for this analysis is the European Soccer Database[1] compiled by Hugo Mathien on Kaggle. The dataset contains an expansive list of soccer matches across leading leagues in Europe including England. A Github repository of the data is also available at Github-Football Data [2]. The data was compiled via web scraping of soccer matches, team and player statistics from a number of online sources including MX Data [3], Football Data [4] and SoFIFA [5].

The dataset is available as an SQLite database with the multiple tables including

- *Country/League* the Soccer league for the European nation of interest and country ID
- *Match* - information about individual matches including the date, teams at play, lineups where available and league ID. It also includes information about betting odds for different outcomes (home win, home draw, away win) from multiple betting houses. Descriptions for betting fields are obtained are available from the Football data website [11]
- *Player/Player attributes* - player names, physical attributes and FIFA ID and in-game performance attributes of each player
- *Team/Team attributes* - team names and FIFA IDs, team attributes capturing offensive and defensive rankings, style of play and the year

The dataset schema provided primary keys for joining and merging the various tables. For this analysis, the English Premier League subset of data is used to select and tune machine learning estimators and to create a prediction model.

Initial exploration of the data shows that the 2012/13 EPL season has a significant number of null entries in the `Team_Attributes` data. As such, this season was excluded entirely from the analysis. Each season consists of 20 teams playing each week for 38 weeks or stages totaling 380 games a season.

For the remaining seasons, a combination of approaches are used to handle null or missing data. Since the data was obtained from web scraping and was likely to contain errors for non-null entries, standard approaches to filling in missing data like back-filling and forward-filling were not used. Missing data

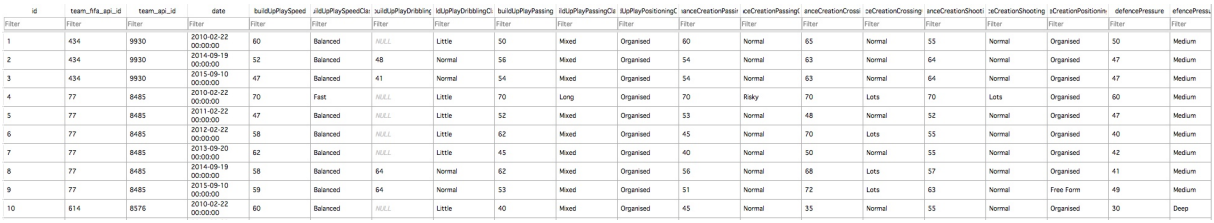
was handled by first dropping columns where a large fraction or majority percentage are null. Next, for columns with low incidence of null entries, for each null entry the corresponding row is dropped is entirely dropped from the dataset.

id	country_id	league_id	season	stage	date	match_api_id	home_team_api_id	away_team_api_id	home_team_goal	away_team_goal	home_player_X1	home_player_X2	home_player_X3	home_player_X4	home_player_X5	home_player_X6	home_player_X7	home_player_X8	home_play
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	2008/2009	1	2008-08-17 00:00:00	492473	9987	9993	1	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
2	1	1	2008/2009	1	2008-08-16 00:00:00	492474	10000	9994	0	0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3	1	1	2008/2009	1	2008-08-16 00:00:00	492475	9984	8635	0	3	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
4	1	1	2008/2009	1	2008-08-17 00:00:00	492476	9991	9998	5	0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
5	1	1	2008/2009	1	2008-08-16 00:00:00	492477	7947	9985	1	3	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
6	1	1	2008/2009	1	2008-09-24 00:00:00	492478	8203	8342	1	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 1: Sample Match rows

id	team_api_id	team_fifa_api_id	team_long_name	team_short_name
Filter	Filter	Filter	Filter	Filter
1	9987	673	KRC Genk	GEN
2	9993	675	Beerschot AC	BAC
3	10000	15005	SV Zulte-Waregem	ZUL
4	9994	2007	Sporting Lokeren	LOK
5	9984	1750	KSV Cercle Brugge	CEB

Figure 2: Sample Team rows



id	team_fifa_api_id	team_api_id	date	buildUpPlaySpeed	ildUpPlaySpeedCla	uildUpPlayDribbling	ildUpPlayDribblingCl	buildUpPlayPassing	ildUpPlayPassingCla	ildUpPlayPositioningC	anceCreationPassi	ceCreationPassingC	anceCreationCrossi	zeCreationCrossing	anceCreationShooti	zeCreationShooting	zCreationPositionin	defencePressure	efencePressu
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	434	9930	2010-02-22 00:00:00	60	Balanced	NULL	Little	50	Mixed	Organised	60	Normal	65	Normal	55	Normal	Organised	50	Medium
2	434	9930	2014-09-19 00:00:00	52	Balanced	48	Normal	56	Mixed	Organised	54	Normal	63	Normal	64	Normal	Organised	47	Medium
3	434	9930	2015-09-10 00:00:00	47	Balanced	41	Normal	54	Mixed	Organised	54	Normal	63	Normal	64	Normal	Organised	47	Medium
4	77	8485	2010-02-22 00:00:00	70	Fast	NULL	Little	70	Long	Organised	70	Risky	70	Lots	70	Lots	Organised	60	Medium
5	77	8485	2011-02-22 00:00:00	47	Balanced	NULL	Little	52	Mixed	Organised	53	Normal	48	Normal	52	Normal	Organised	47	Medium
6	77	8485	2012-02-22 00:00:00	58	Balanced	NULL	Little	62	Mixed	Organised	45	Normal	70	Lots	55	Normal	Organised	40	Medium
7	77	8485	2013-09-20 00:00:00	62	Balanced	NULL	Little	45	Mixed	Organised	40	Normal	50	Normal	55	Normal	Organised	42	Medium
8	77	8485	2014-09-19 00:00:00	58	Balanced	64	Normal	62	Mixed	Organised	56	Normal	68	Lots	57	Normal	Organised	41	Medium
9	77	8485	2015-09-10 00:00:00	59	Balanced	64	Normal	53	Mixed	Organised	51	Normal	72	Lots	63	Normal	Free Form	49	Medium
10	614	8576	2010-02-22 00:00:00	60	Balanced	NULL	Little	40	Mixed	Organised	45	Normal	35	Normal	55	Normal	Organised	30	Deep

Figure 3: Sample Team Attribute rows

Exploratory Visualization

Figures 4 and 5 show how win, draw and loss percentages for the home team varies with season stages[1] of a season and also with seasons respectively.

The overall statistics for the EPL seasons shows that the home team has a win percentage of 46%, loses 29% and draws 25% of home games. With these class percentages, we can compute the degree of randomness in the outcomes of home games via an entropy calculation. The entropy of the dataset is computed to be as

$$Entropy = -0.46 * log_2 0.46 - 0.29 * log_2 0.29 - 0.25 * log_2 0.25 = 1.53 \quad (1)$$

This value is about 96.7% of the maximum entropy of a 3 class system; the maximum entropy corresponds to equal split (33.3% per class) between all three classes. This measure of entropy indicates a high degree of randomness in the dataset. It is a measure of the information gain a given machine learning estimator would need to achieve in order to efficiently differentiate between classes in this dataset.

The plots also highlight the imbalance in the dataset between the three classes with the *win* class dominating the other two classes. Applying an classifier to separate the classes may result in a very high accuracy as the estimator gets tuned to overfitting for the dominant class. Given the imbalance, different performance metrics beyond a simple measure of accuracy are utilized for this project. These metrics include an F1-score, logarithmic log-loss, a confusion matrix and ROC curves adapted for this multi-class problem. These performance measures/indicators are able to quantify how well a chosen estimator correctly assigns examples to a class while wrongly assigning examples to other classes.

[1] Each EPL season has 38 stages, a stage referring to a weekend of games when each team is in play

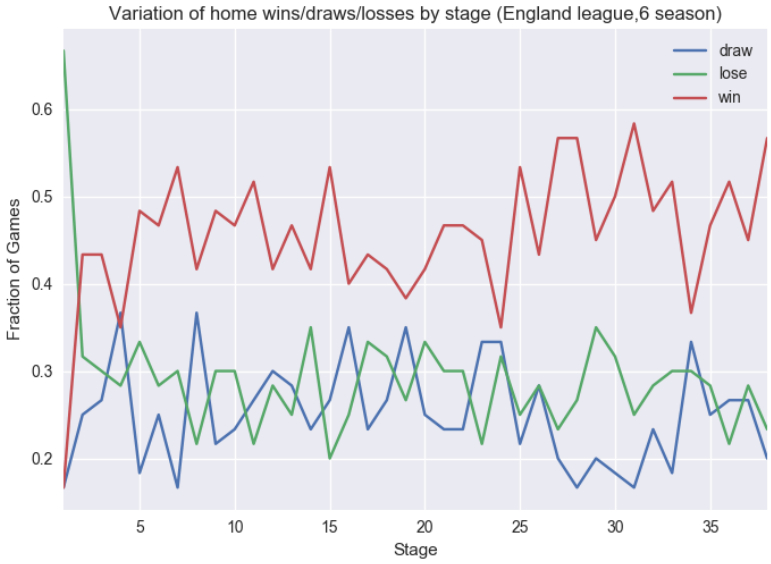


Figure 4: Variation of Home team outcomes by EPL season stage across 6 seasons

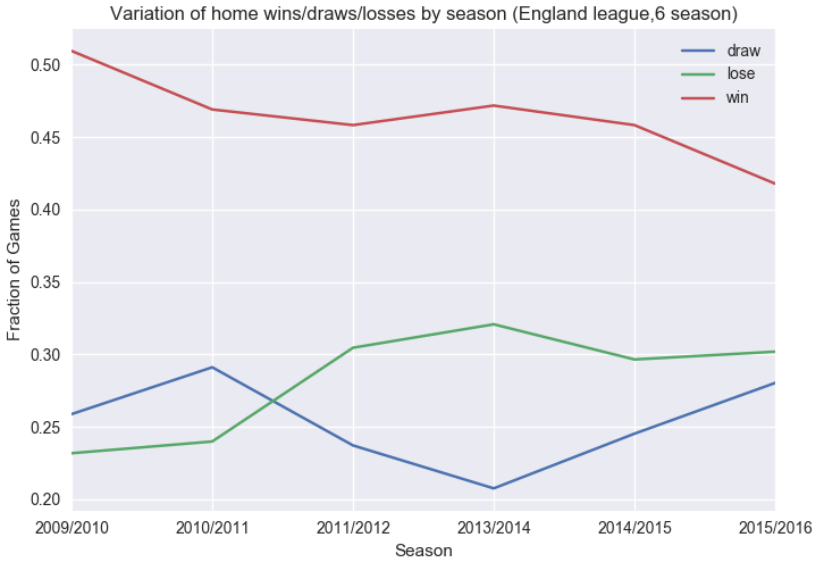


Figure 5: Variation of Home team outcomes with EPL season

Figure 5 shows the distribution of wins, draws and losses varies from season to season in the dataset. The home win percentage drops almost 20% between the 2009/10 to 2015/2016 seasons; the draw/loss percentages alternately increase and decrease with season. Training an estimator on a single season could lead to overfitting on a specific distribution of classes in the training set resulting in poor performance on the test set. For this project, season 2015/16 is set aside as the test dataset and estimators are trained on the preceding seasons.

Figure 4 shows significant variation in the class probabilities over the 38 stage season. The variation is more prominent in the early stages and class distributions stabilize after about 10 stages.

Algorithms, Estimators & Techniques

Multiple classification algorithms were trained on a subset of training matches and then their test error/accuracy were computed to select the best model. The Machine Learning estimators considered in this project included multiple Support Vector Machine (SVM) estimators with different kernels, Decision tree classifier, Random Forest, Sequential gradient descent, K-Nearest Neighbor and boosted classifiers (Adaboost in this case). A baseline Dummy classifier was also created and used to train the data as well.

2.0.1 Support Vector classifiers (SVC)

Linear and Radial Basis function (RBF) kernels for SVC are applied to this dataset. SVC classifiers are used since they are able to handle both linear and non-linear features given the right kernels. It is not immediately clear from the features available for our match prediction if there are non-linear correlations between features and the match outcomes; this is especially so for the categorical features. Although, SVCs are computationally expensive for large datasets as they require computing a distance metric, our dataset is small and computing cost is not a factor. Also, for multi-class problems a one-versus rest variant of SVC can be readily implemented. In literature, SVC classifiers have been shown to have the best performance for predicting soccer matches. SVC classifiers are initialized with most default parameters from the `sklearn.svm.svc` module. Given the unbalanced distribution of output classes in our three-class dataset we used a `class_weight = balanced` and a one-versus-rest `decision_function_shape` parameters. We also enabled the option to compute probabilities (`probability = True`); probability information for use in computing the log_loss metric and for generating Receiver-Operator curves (ROC).

2.0.2 Decision Tree Classifier

A Decicison tree classifier was applied to this dataset because it is able to handle multi-class problems and the large number of categorical features. In our implementation, default parameters in the `sklearn.tree.DecisionTreeClassifier` are used.

2.0.3 Random Forest and Boosting Classifiers

These ensemble classifier, like Decision trees, are chosen since they can handle multi-class problems. They are also able to reduce overfitting due to subsampling of the dataset. For this analysis an Adaboosted classifier is implemented as a `sklearn.multi-class.OneVsRestClassifier` with default module parameters. Default parameters for the `sklearn.ensemble.RandomForestClassifier` are chosen for the Random Forest classifier implementation.

2.0.4 k-Nearest Neighbor (kNN) Classifier

This is implemented as using the `sklearn.neighbors.KNeighborsClassifier` module with default parameters except for the weights parameter; a `weights = distance` setting is used to weigh predictions inversely to a distance metric. kNN applicable to multi-class problems like the match prediction problem. It can be computationally expensive due to computation of distance metrics and the choice of the distance metric is key. However, it is robust to noise in the training data which may be important for our web-scraped data.

2.0.5 Sequential Gradient Descent Classifier (SGDC)

The SGDC classifier was implemented using a `sklearn.multi-class.OneVsRestClassifier` module with the estimator set to an a `sklearn.linear_model.SGDClassifier` instance. Parameters for the SGDClassifier include using a log loss function and `n_iter = 100`.

2.0.6 Dummy Classifier (SGDC)

In the initial data exploration we observed that our labeled dataset has more samples with a *win* label than the *loss* and *draw* labels. This uneven distribution of class labels in the dataset may impact performance of some classifiers. To provide a baseline performance for this unbalanced dataset I also implemented a Dummy classifier using the `sklearn.dummy.DummyClassifier` This classifier is initialized with parameter `strategy = most_frequent` and predicts the most frequent label in the dataset.

For each selected classifier, where available, the option to compute prediction probabilities for each class is enabled. Based on performance on the training and test sets, selected classifiers were further tuned with the default parameters as a starting point to identify the best performing parameters for the classifier.

Other researchers have used SVM as the algorithm of choice based on predicted model errors; I will compare any selected algorithm against an Support Vector Machine (SVM) model.

Benchmark

For this project I will be using both the predictions of journalist Mark Lawrenson [9] and the match predictions from FiveThirtyEight[8] as the benchmark models for assessing my proposed solution. Each week during the EPL season, Mark Lawrenson's makes match predictions on his BBC sports blog. His predictions can be considered as subject matter expert opinion since they are based on his knowledge of the sport, the history and quality of teams (including players and managers) and recent performance. His predictions also include short explanations where he talks about recent team performances and head-to-head results.

The FiveThirtyEight EPL predictions are model based and use the Soccer Power Index (SPI) created by Nate Silver. The SPI index is a rating index with high rating for the best performing teams and low ratings for the worst performing teams. To predict the outcome of a given match, their model uses a Poisson model to predict the likely goals scored or conceded by each team. A team is predicted to win a given match if it scores the most goals and concedes the fewest. Draws are predicted when the likely goals scored and conceded for the two teams match. The outcomes of matches over a season are then predicted using a Monte Carlo Simulation. For each match the benchmark returns win, draw and loss probabilities for each team.

For each of these benchmark models, I computed the F1 scores and created the confusion matrices for comparison with the machine learning models I created.

3 Methodology

Data Preprocessing

The source dataset for this analysis was available as a SQLite database and contained match and player information from multiple leagues. An SQLite interface for panda was utilized to import the tables into a `pandas`Dataframe for additional preprocessing. To improve the performance of the data import Only data corresponding the league of interest, the EPL in this case, were filtered and imported into Dataframes.

The data schema for the three key database tables imported and merged to create our dataset are shown in Figure 6. It shows the subset of tables and columns from our source database which are used in this analysis.

Rules for merging the database tables are:

- `Team` and `Team Attribute` data are merged on the `team_api_id` during import.
- The `Match` table is merged with `Team Attribute` data by joining on the `team_api_id` with the `home_team_api_id` for the home team
- the away team information is merged by joining `away_team_api_id` of the `Match` table with the `team_api_id` of the Team and Team Attribute tables
- `Match` and `Team Attribute` tables are merged where the `Team Attribute date` falls within the start and end dates of the `Match season`

	cid	name	type
1	0	id	INTEGER
2	1	country_id	INTEGER
3	2	league_id	INTEGER
4	3	season	TEXT
5	4	stage	INTEGER
6	5	date	TEXT
7	6	match_api_id	INTEGER
8	7	home_team_api_id	INTEGER
9	8	away_team_api_id	INTEGER
10	9	home_team_goal	INTEGER
11	10	away_team_goal	INTEGER

(a) Match Table

	cid	name	type
1	0	id	INTEGER
2	1	team_api_id	INTEGER
3	2	team_fifa_api_id	INTEGER
4	3	team_long_name	TEXT
5	4	team_short_name	TEXT

(b) Team Table

	cid	name	type
1	0	id	INTEGER
2	1	team_fifa_api_id	INTEGER
3	2	team_api_id	INTEGER
4	3	date	TEXT
5	4	buildUpPlaySpeed	INTEGER
6	5	buildUpPlaySpeedClass	TEXT
7	6	buildUpPlayDribbling	INTEGER
8	7	buildUpPlayDribblingClass	TEXT
9	8	buildUpPlayPassing	INTEGER
10	9	buildUpPlayPassingClass	TEXT
11	10	buildUpPlayPositioningClass	TEXT
12	11	chanceCreationPassing	INTEGER
13	12	chanceCreationPassingClass	TEXT
14	13	chanceCreationCrossing	INTEGER
15	14	chanceCreationCrossingClass	TEXT
16	15	chanceCreationShooting	INTEGER
17	16	chanceCreationShootingClass	TEXT
18	17	chanceCreationPositioningClass	TEXT
19	18	defencePressure	INTEGER
20	19	defencePressureClass	TEXT
21	20	defenceAggression	INTEGER
22	21	defenceAggressionClass	TEXT
23	22	defenceTeamWidth	INTEGER
24	23	defenceTeamWidthClass	TEXT
25	24	defenceDefenderLineClass	TEXT

(c) Team Attributes

Figure 6: Key database table schema

After the data import, information from the team, team attributes, matches and match attributes tables were merged, without column duplication, into the initial dataframe for analysis. A number of match attribute columns were excluded from this analysis based on an initial exploration of the SQLite tables. Of the 115 columns in the Match table, only the first 11 columns were imported. The remaining 104 columns contain the team player lineups for the home and away sides. A large number of these columns contained null entries or the data was incomplete for a selected team. The excluded columns also included features that capture betting odds information for a given match from multiple sports betting sources. These were excluded from this supervised learning problem since this betting feature can be considered predictions from another model.

A number of the columns from the `Team Attribute` table contain categorical data which cannot be handled by the learning algorithms selected for this analysis. For some algorithms, like k-Nearest Neighbor and Support Vector machines, which use a distance estimation to model similarity between individual samples from the dataset, computing distance from categorical data is impossible. Each of the categorical columns were encoded into a set of binary columns using the Python `pandas.get_dummies` function with the number of encoded columns depending on the number of categories in each column. Once encoded these features can then be handled like other numerical features. The downside is the increase in the number of features and size of the model dataset.

Predicting match outcomes from this dataset falls under classification problems in supervised learning. In this case, the categorical output is defined as a discrete set of labels containing a home *win*, *loss* or *draw*. The outcome of each match, which is the output column for each row of the dataset, is a single label from this set. Since the dataset itself does not contain this categorical output, it was computed using two features available in the Matches table of the original dataset: *home_team_goals* and *away_team_goals*. The output label for each row is assigned as follows

$$home\_team\_output = \begin{cases} \text{win}, & \text{if } home\_team\_goals > away\_team\_goals \\ \text{draw}, & \text{if } home\_team\_goals == away\_team\_goals \\ \text{loss}, & \text{otherwise} \end{cases} \tag{2}$$

After importing the dataset for the EPL, a simple splitting approach was used to create training and test datasets. EPL season 2015/2016, was split from the dataset to be used a test dataset; data for all preceding seasons are included in the training dataset. Seasonal splitting of the dataset, instead of the percentage or fractional split, is a valid approach because there is a time-dependency in a team's

performance tied to individual seasons. In soccer, as in other team sports, the players on a team may change significantly from season to season and some game rules may change as well. Also, in-season winning or losing streaks can impact the outcomes of subsequent matches; a team who wins a series of matches in a row may be more likely to win their next game (Cite researchs). However, the impact of such a winning or losing streak may not extend from the end of one season to another season.

Implementation

An initial evaluation of training and test errors for each classifier was performed using the preprocessed dataset. There were two sets of features in the dataset: one set for the home team and the other for the away team. A given feature for the home team had a complementary feature for the away team. Aside from removing null entries and encoding categorical data, our initial implementation of classifiers did not include any computed features in dataset.

For learning algorithms which use a distance or norm metric to determine classification, like kNN and SVC (both linear and RBF), it is necessary to standardize features. This helps prevent the classifier performance from being negatively impacted by large differences in scales between different features in the dataset. Without standardization, one or more features which are larger in magnitude might dominate the model leading to overfitting and poor classification. The final preprocessing step was to standardize all features using in our matches dataset using the `sklearn.preprocessing.StandardScaler` package with default parameters. The `StandardScaler` instance was fit to training data and then used to transform both the training and test datasets.

During training, each cassifier was initialized with parameters as discussed in Section 2. The classifier was then fit to the training data [2], setting the matches feature dataset to input and the calculated output labels (See Equation 2) as the true label. Using the sklearn the `Classifier.fit` method was called on each classifier after initialization as shown in the code-clock 3 below. Each classifier was then used to predict labels for samples in the training and test datasets. The predictions were then used to compute training and test errors for the dataset.

```
// initialize classifier with parameters
clf = Classifier(parameters)
// fit on training dataset
clf = clf.fit(training_match_features, training_match_labels)
// get classifier predictions for training season data
training_predictions = clf.predict(training_match_features)
// get classifier score for training data
training_scores = clf.score(training_predictions, training_match_labels)
// get classifier predictions for test season data
test_predictions = clf.predict(test_match_features)
// get classifier score for test data
test_scores = clf.score(test_predictions, test_match_labels)
```
(3)

Results for the classifiers we implemented in the model are presented in Table 1. F1 scores for training and test datasets are reported for each classifer. The results show a wide variation in F1 score across the different classifiers. For some classifiers a large difference in between the training and test F1 scores. This large gap in the performance metric is an indicator that the model is overfitting on the training data and is not easily extensible to new data. The Decision Tree (gap of 0.65756), Random Forest (0.61343) and kNN (gap of 0.58723) classifiers appear to overfit the training data. The SVC with the RBF kernel shows the best performance on the test data with an F1 score of 0.42151. Together with the SGD and Adaboosted classifiers show smaller gaps between training and test F1 scores indicating better robustness for their input parameters.

Similar trends are observed when the log-loss performance metric is used: the Decision Tree, Random Forest and kNN classifiers overfit to the training data and the SVC with RBF kernel has the lowest log-loss test value. Results are shown in Table 2.

[2]The training dataset consists of all EPL seasons except for the 2015/2016 season which is set aside as the test dataset

Classifier	Training Score	Test Score
Dummy Classifier	0.304 1	0.246 2
Linear SVC	0.469 0	0.370 5
RBF SVC	0.586 5	0.421 5
Decision Tree Classifier	0.998 9	0.341 4
Adaboost	0.517 1	0.326 7
Random Forest Classifier	0.985 4	0.372 0
SGD Classifier	0.501 9	0.376 1
kNN	0.999 5	0.412 2
One vs Rest SGDC	0.481 8	0.354 4

Table 1: Variation of F1 Score with Classifier for Training and Test datasets

Classifier	Training Score	Test Score
Dummy	18.191 0	20.108 8
Linear SVC	0.999 3	1.079 2
RBF SVC	0.940 0	1.064 7
Decision Tree	0.001 5	22.622 4
Adaboost	1.093 1	1.097 6
Random Forest	0.272 9	1.749 5
SGDC	1.015 1	1.122 2
kNN	0.000 7	4.947 3
One vs Rest SGDC	0.983 2	1.111 6

Table 2: Variation of Log-loss metric with Classifier for Training and Test datasets

To obtain better estimates of the model error for each of the classifiers we also used K-folds cross-validation. Given the relatively small size of samples in our matches dataset across the 6 EPL seasons, K-folds helps assess the predictive performance of each classifier. It can also provide some statistics on the variability in the predictive performance of each classifier. A k of 5 was used to generate the results in Table 3.

The cross-validation F1 scores confirm the best performing classifiers to be the RBF SVC, Adaboost and SGDC implementations. The RBF SVC model generates the lowest log-loss metric. The largest observed F1 score is 0.448, by the Adaboost classifier, and all classifiers consistently outperform than the Dummy classifier.

Classifier	F1 score	Log-loss
Dummy	0.304 1	18.191 0
Linear SVC	0.419 8	1.027 4
RBF SVC	0.441 0	1.024 1
Decision Tree	0.428 6	19.793 6
AdaBoost	0.448 0	1.095 3
Random Forest	0.430 4	3.168 1
SGDC	0.442 3	1.070 3
kNN	0.416 1	5.143 1
One vs. Rest SGDC	0.427 7	1.048 8

Table 3: Performance Metrics from k-Folds cross-validation runs

Refinement

The current set of features in our dataset capture attributes of the home and away teams for a given match. For the first pass implementation of classification algorithms, we used only features already available in the source Matches and Team tables. We then modified the dataset to include additional computed features described below.

3.0.1 Team Form Feature

For this analysis, we did not use any external team performance rankings like the SPI ratings used by FiveThirtyEight [8]. However, we created a *form* feature to capture each team's current performance

level at different stages as a season progresses. The form can be described as a dynamic measure of a teams performance. At any stage in a season, the form for a given team is captured in three new features: *team_win_average*, *team_draw_average* and *team_lose_average*. These features are added to our original feature dataset for both the home and away teams. In total, this added six (6) new features to our dataset. The win, draw and lose averages at each stage are computed using results from previous matches or stages within a given season; averages are not computed across seasons as teams may change personnel or playing approach.

The number of previous stages or matches used in calculating these new features, is termed the *window*, and is a new model parameter; *window* can range from 1 to maximum matches in a season per team (38). The choice of a window size then becomes an additional parameter to select via cross validation. For the first stage in the season, there is no previous results to compute form. To get a better measure of team performance we expect more than 1 previous game result is necessary to averages. This is also confirmed in Figure 4, the win, draw and loss percentages vary wildly in the early stages (up to about stage 5) of the season before stabilizing around a plateau.

With the additional form features added, we also implemented logic to drop the first N games in a season from the dataset, where N is the window of game used to compute the form averages. For example, selecting `window = 3` implies that the first 3 games of a given season for each team are used to compute the form features for game 4; the corresponding rows for the first 3 matches are then dropped from the dataset used in training the classifiers.

Results for K-fold cross-validation runs using different values for the *window* parameter and computing the corresponding team form are provided in Figures 7, 8. Figure 7 shows the F1 score increasing with the size of window, the number of previous games used to compute the form for a given team, up to about 14 games into a season when the One-vs-Rest SGD Classifier returns a score of 0.503191. As window size increases further the F1 metric then oscillates, then reaches another high of 0.506021 for the Adaboost Classifier at `window = 18`. The peak F1 scores observed by including form features and a parametrized form *window* are about 10% larger than the peak scores reported without including the form parameter. The results also imply that to obtain good predictions, I needed to use results from almost a third of matches in season to train and compute the form of each team. The best performing classifier are the SGD classifiers, Adaboost and RBF classifier in decreasing order of performance.

The Log-loss metric, in contrast, shows very little variation in the value of the metric as window size increases for each classifier. Figure 8 also shows a clearer differentiation between classifiers with a subset including the variants of SGD classifier, the RBF & Linear SVC, and Adaboost classifiers having low losses.

For both metrics, the Decision Tree and Dummy classifiers had the worst prediction performance.

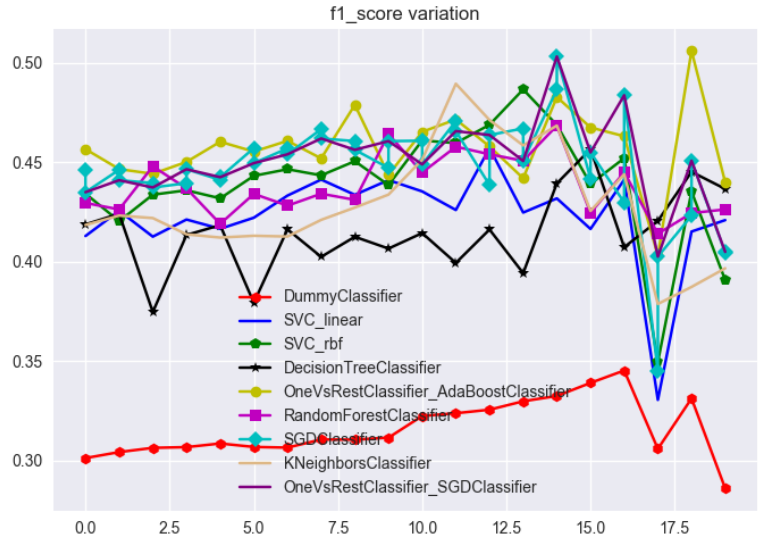


Figure 7: Variation of F1 score with size of Form window

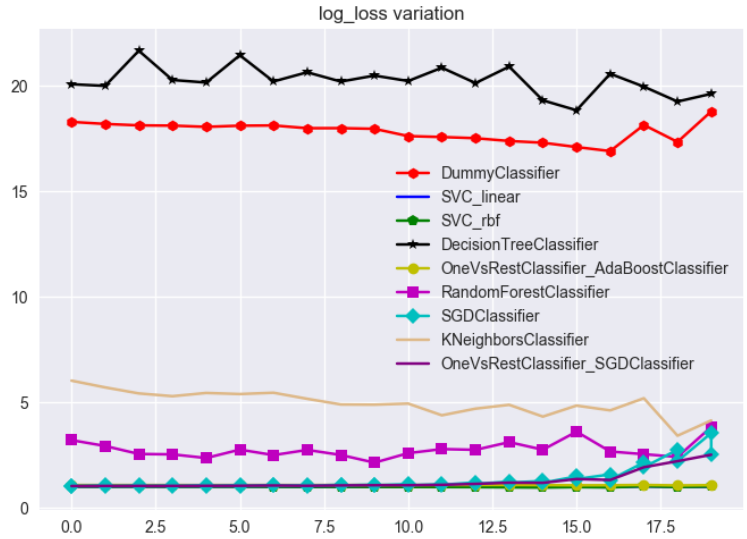


Figure 8: Variation of Log-loss score with size of Form window

3.0.2 Home Advantage Feature

Recent analysis of the EPL in literature indicates that there is a home team advantage that impacts the outcome of EPL matches [12, 13]. Reade and Koyama [13] show that, although historically home teams win more than 50% of the time, the advantage has decreased somewhat in recent years.They provide two alternate measures to capture the home team advantage for a given team a) ratio of point gained in home games to points gained in away games, and b) difference in mean goals scored at home and mean goals scored away from home

Each of these approaches uses the results of a team's previous matches to compute the home advantage feature for a current match. This feature is implemented by adding an additional column `advantage_home` to the matches dataset.

Results for both approaches to computing home advantage are presented in Tables 4 and 5. The maximum F1 and log-loss values obtained show that adding this additional feature does not improve the classifier performance over the initial results presented in Table 3. For the goals-based approach, the Adaboosted classifier has the maximum F1 score of 0.452154, a slight improvement over the 0.448 in Table 3. Using points to compute home advantage results in a maximum F1 score of 0.450356 for the Adaboosted classifier. These increases represent about a 1% increase in F1 score. As such, the home advantage feature is not included in subsequent model tuning using the matches dataset in this analysis.

Classifier	F1 score	Log-loss
Dummy	0.304 1	18.191 0
Linear SVC	0.418 7	1.026 4
RBF SVC	0.441 7	1.023 5
Decision Tree	0.425 1	19.901 8
Adaboost	0.452 2	1.095 4
Random Forest	0.433 7	3.224 3
SGD	0.438 8	1.072 2
kNN	0.417 0	5.516 8
One vs. Rest SGDC	0.428 3	1.058 0

Table 4: F1 scores with home advantage feature using home vs. away goal difference

Classifier	F1 score	Log-loss
Dummy	0.304 1	18.191 0
Linear SVC	0.416 5	1.027 8
RBF SVC	0.433 1	1.023 5
Decision Tree	0.426 4	19.833 7
Adaboost	0.450 4	1.095 4
Random Forest	0.436 0	3.044 2
SGD	0.427 2	1.079 3
kNN	0.411 8	5.418 4
One vs. Rest SGDC	0.427 5	1.054 6

Table 5: F1 scores with home advantage feature using home vs. away points ratio

Each of the three best performing classifiers - the one-vs-rest SGD, one-vs-rest AdaBoost and RBF SVC classifiers subsequently had their parameters tuned to obtain the best performance. We utilized a cross-validation approach using the GridSearchCV module in sklearn. The classifiers were tuned with a *form* `window = 14`, the peak window value from Figure 7. For the RBF SVC classifier the tuning parameters are C, to control the regularization, and γ, to control how points impact the selected kernel. The one-vs-rest classifier tuning parameters are α and *n_iter* (the number of iterations). For the Adaboost classifier we tuned *n_estimator* and `learning rate` parameters. The training and test results are shown in Table 6. The one-vs-rest Adaboost classifier performs best on the test with an F1 score of 0.471144. The one-vs-rest SGDC classifier is the next best classifier using the test score. In contrast, the RBF SVC had the worst F1 score (0.363685) on the test data although the training score was above 0.5 (=511753). The SVC classifier appears to be overfitting to the training set. In addition, the choice of a window size of 14 may also be impacting the performance. Re-running the parameter tuning with different values for *window* we obtain a different maximum test F1 score for the RBF SVC classifier: 0.446742 at `window = 10` with RBF SVC parameters of `C = 2.25025` and γ `= 0.001`.

With these results the best classifier identified for predicting match outcomes was the one-vs-rest Adaboost classifier.

Classifier	Training score	Test score
RBF SVC	0.511 8	0.363 7
One vs. Rest Adaboost	0.505 2	0.471 1
One vs. Rest SGDC	0.509 2	0.461 4

Table 6: F1 scores for parameter tuning classifiers

4 Results

Model Evaluation & Validation

The One-vs-rest implementation of the Adaboost classifier was tested with two additional leagues other than the EPL: the Spanish (*La Liga*) and German (*Bundesliga*) leagues. Outside of the EPL, these two leagues are two of the best soccer leagues in Europe. The goal is to assess how well the model generalizes to new data.

We implemented the One-vs-rest Adaboost classifier using a `window = 14` and the best tuned parameters for the classifier: α `= 0.0001` and `n_iter = 1800`. For both leagues, the One-vs-rest Adaboost classifier reports better F1 scores than for the EPL league. This is the case although the model has not been tuned for each of the two additional leagues to select either an optimal form *window* or Adaboost parameters, α and n_iter.

League	Training score	Test score
England (Premier League)	0.509 2	0.461 4
Spain (La Liga)	0.558 1	0.590 4
Germany (Bundesliga)	0.576 7	0.485 8

Table 7: One-vs-Rest Adaboost F1 scores for different European Leagues

Justification

The best reported training and test scores for the three (3) previously selected classifiers are reported in Table 6 For the 2015/16 EPL season, the benchmark predictions by *'Lawro'* Lawrenson BBC Sports reporter results in an F1 score of 0.46. This is less than the best training and test F1 scores obtained with the One-vs-rest Adaboost classifier. This is also true for the One-vs-rest SGD classifier as well. Our best performing classifier an expert with domain knowledge in this case. A look at the confusion matrix for Lawrenson's predictions in Table 8 shows he overpredicts wins for the home team probably due to his knowledge of the home team advantage discussed in Section 3.0.2. He underpredicts losses although for the season in question there are more home team losses than draws.

	Predict Draw	Predict Win	Predict Lose
Actual Draw	40	51	16
Actual Win	30	105	22
Actual Lose	35	47	34

Table 8: F1 scores for Mark Lawrenson 2015/16 EPL Predictions

For the 2015-16, there was no data available for the predictions from FiveThirtyEight, a sports and statistics news site. We used data for teh 2016/17 season. Although, this is not an exact match, it still provided insights into their use of a combined team ranking and probabilistic estimates of goals scored by home and away teams for match predictions. For the 2016/17 season, FiveThirtyEight had an F1 score score of 0.52. This was much larger than the best performing classifier and the Mark Lawrenson benchmark. However, a closer look at the confusion matrix for the FiveThirtyEight predictions in Table 9 shows their model never predicts a draw as the most likely outcome for any match. Their model generates probabilities for each of the three (3) labels for each match and we select the label with the maximum probability as the predicted outcome. This skews the results since their predictions are essentially for a two class system which may be easier to predict.

	Predict Draw	Predict Win	Predict Lose
Actual Draw	0	51	16
Actual Win	0	105	22
Actual Lose	0	47	34

Table 9: F1 scores for Mark Lawrenson 2015/16 EPL Predictions

5 Conclusion

Free-Form Visualization

multi-class ROC curves for the one-vs-rest Adaboost classifier applied to both the EPL and Spanish Leagues are plotted in Figures 9 and 10 respectively. TThe area under the curve (AUC) for each of the match outcome classes are low (approximately 0.6). This implies that, for this problem, the selected classifier has difficulty in discriminating between the classes. This difficulty is more pronounced for the *draw* class. The ROC curve for the *draw* class in the EPL league actually drops below the diagonal of the plot for a large range of the False positive rate. This difficulty in predicting draws was also observed when we examined the confusion matrices for the benchmark solutions. This impacts the overall performance of the classifiers as any misclassification of the draw class reduces the F1 score. The larger AUC for each of the classes in Figure 10 also validates the higher F1 performance scores I obtained for the Spanish league compared to the EPL.

Reflection

In this project, we selected a classifier for predicting English Premier league games for multiple seasons. The model generation required implementing multiple classifiers and evaluating performance by training on training and test data. The performance metrics chosen for the analysis included an F1 score and log-loss metric. A subset of the classifiers were also compared to benchmark solutions from Mark Lawrenson, a sports pundit, and a statistical model from FiveThirtyEight. The best performing classifier, a one-vs-rest Adaboost model, performed better than the predictions for the BBC sports pundit, Mark Lawrenson.

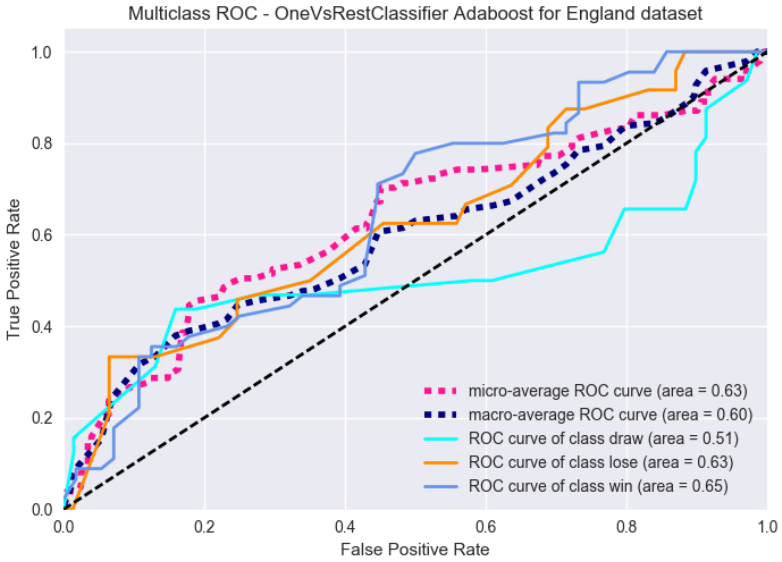


Figure 9: multi-class ROC for Adaboost classifier for EPL

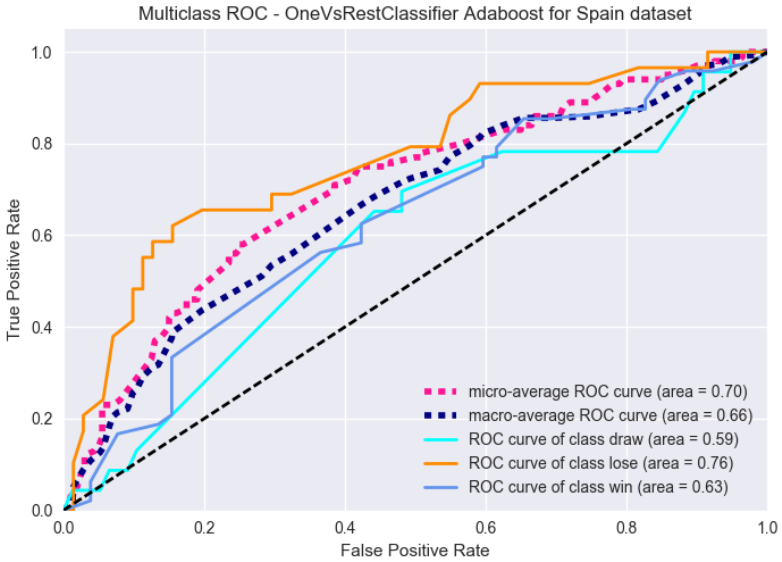


Figure 10: multi-class ROC for Adaboost classifier for Spanish La Liga

Improvement

The learning model performed well compared to benchmarks and was easily extended to additional soccer leagues in Europe. However, a number of the classifiers implemented, including the SVC classifier which has been used extensively in literature for this problem, created models which overfit the training data. Possible improvements to these models include performing extensive tuning of parameters over a wider parameter space. Other additional improvements would be to refine calculation of the team *form* without having to drop a large fraction of dataset rows. This might require computing team rankings that span multiple seasons and allow using all games in a season to train and test a classifier; this would be similar to the SPI ratings used in the FiveThirtyEight models.

Even without these improvements the Adaboost classifier did outperform one of the benchmarks. This helps validate the approach utilized in this analysis.

References

[1] European Soccer Database - https://www.kaggle.com/hugomathien/soccer

[2] Github Soccer Data - https://github.com/hugomathien/football-data-collection

[3] MX Data - http://football-data.mx-api.enetscores.com/

[4] Football data - http://www.football-data.co.uk/

[5] SoFIFA - http://sofifa.com/

[6] Deloitte Annual Review of Football Finance 2016 - https://www2.deloitte.com/uk/en/pages/sports-business-group/articles/annual-review-of-football-finance.html

[7] Martin Lawrenson BBC - http://www.bbc.com/sport/football/39795853

[8] FiveThirtyEight EPL predictions - https://projects.fivethirtyeight.com/soccer-predictions/premier-league/

[9] Lawros Predictions

[10] Financial Times EPL 2016/17 Predictions

[11] Betting Site descriptions

[12] Nevill, Alan and L Holder, R - Home advantage in sport: an overview of studies on the advantage of playing at home (1999)

[13] Reade, J James and Koyama, Mark - Playing Like the Home Team: An Economic Investigation into Home Advantage in Football (2009)