Probability Estimation Within Soccer Matches

Isaac Schmidt ISCHMIDT20 @ BERKELEY.EDU
Department of Statistics

Abstract

In this paper, a public dataset containing events from over 1,800 matches across Europe is analyzed. Random forests are used to estimate the probabilities of each match ending in a win, draw, or loss for the home team at various points, using just a handful of features and following an extensive hyperparameter search. These probabilities are verified to match with common beliefs, and then used to measure each individual player's contribution to the outcome in terms of "expected points added." This paper ends with a discussion on future improvements upon this work.

1. Background

Over the past couple of decades, many sports have seen analytics revolutions, with teams and organizations beginning to leverage data to determine the optimal in-game strategy or roster construction. Such developments have also spurred new technologies, such as MLB's Statcast and NFL's NextGen Stats, to allow for the collection of data on every detail in the game. Enhanced statistics are also consumed by the fan—for example, win probability charts such as those displayed on FanGraphs or ESPN are fun ways to display how exciting a game was to witness.

Estimating win probabilities has been popular within baseball for some time, for a number of reasons:

- A manager can use them to determine whether or not to call for a certain action, such as a stolen base or a sacrifice bunt. If the expected win probability after attempting the action is greater than the current win probability, that is a reason to elect for that action.
- Sports betting agencies are interested in win probabilities to determine what odds to offer on prop bets, and bettors use their own internal estimates when deciding whether or not to place such a bet.
- A player's performance can be measured by how much he increased (or decreased) his team's win probability over the course of the game. For example, a player that consistently performs well in close games may be preferred to a player that "stat-pads" by only playing well when the game is out of hand.

It is the last item which will be the subject of this paper. Unfortunately, analytics has lagged behind in soccer, in part due to a lack of collection of detailed data. Unlike baseball, soccer is a very fluid sport, and actions that significantly influence the result only occur sporadically throughout each match. Measuring added win probability, or expected points,

over the course of a match can spotlight players that may have had a large influence on the outcome, even if these players did not record any goals or assists.

One hinderance to this analysis has been the lack of sufficient data that is publicly available. However, thanks to the efforts of Pappalardo et al. (2019), this is no longer the case.

2. The Dataset

A detailed description of the data, including how it was collected, is contained in Pappalardo et al. (2019), but it will be summarized here. The data of interest are events for every match in Europe's top 5 leagues (England, Spain, Germany, Italy, and France) for the 2017-18 season, in JSON format. This is a total of 1,826 matches, each with between 1,300 and 2,000 events, combining for a grand total of over 3 million data points. An "event" is any action during the course of a game, such as a pass, shot, foul, etc. Included for each event are the time at which it occurred, the position of the ball before and after the event, the player committing the event, and usually some tags describing the event, such as whether a pass was actually accurate, or whether a foul resulted in a yellow or red card. Also contained in the data download are files with information on each match and each player, along with tables describing each event code.

2.1 Cleaning

To transform the data into a format suitable for this analysis, a number of wrangling and cleaning operations were required. The code to reproduce these steps may be made available upon request.

1. Transform JSON format into a rectangular table with `pd.json_normalize`.
2. Join match information, such as the end result, to each event.
3. Pinpoint events resulting in goals or red cards, and use `np.cumsum` to determine the score and difference in players on the field at each event.
4. Transform the `seconds` column so that 0 refers to the start of the match, not the start of each half, and give all "stoppage time" events the same time, e.g. 2700 or 5400.
5. From the `subEventName` field, create a new "ball" variable taking on one of 5 possible values: Open Play, Free Kick, Throw-in, Corner Kick, and Penalty Kick.
6. Remove all events with an `eventName` of "Interruption."
7. Expand the `playerId` column into two columns, one containing the ID of the away player and one containing the ID of the home player, leaving one or the other blank, as necessary.
8. Some events have two distinct records corresponding to the same action, such as a "shot" followed by a "save attempt." For these events, collapse those rows so that there is only one record for each event, filling in the appropriate player fields. For

duels, mark the possession variable as -1, indicating that no team had possession at the start of the event.

The result of these cleaning operations was a table of about 2.5 million rows, and 25 columns.

2.2 Splitting the Data

A common approach when building a classification model is to split the original data set into a set which is used to train the model, and a test set to evaluate performance. However, in this instance, the goal is to estimate probabilities, not to maximize accuracy or any other function of the predicted classes. Therefore, instead of withholding a portion of the dataset for testing, 5-fold cross-validation is used instead, with the optimal model the one that minimizes the chosen objective function across all folds.

Instead of splitting at the individual data point level (event), the splitting was done at the match level, and was further stratified by league. Four of the leagues contain 380 matches, so 76 matches from each league were included in each fold. Germany's league only has 306 matches, so 61 matches were chosen for each of the first four folds, and the remaining 62 for the fifth.

3. Methods

3.1 Model Architecture and Exploratory Data Analysis

This dataset contains only a handful of relevant input features, to be discussed in greater detail in Section 3.3. The target variable is the result of the match, which is either a win for the home team, a loss, or a draw.

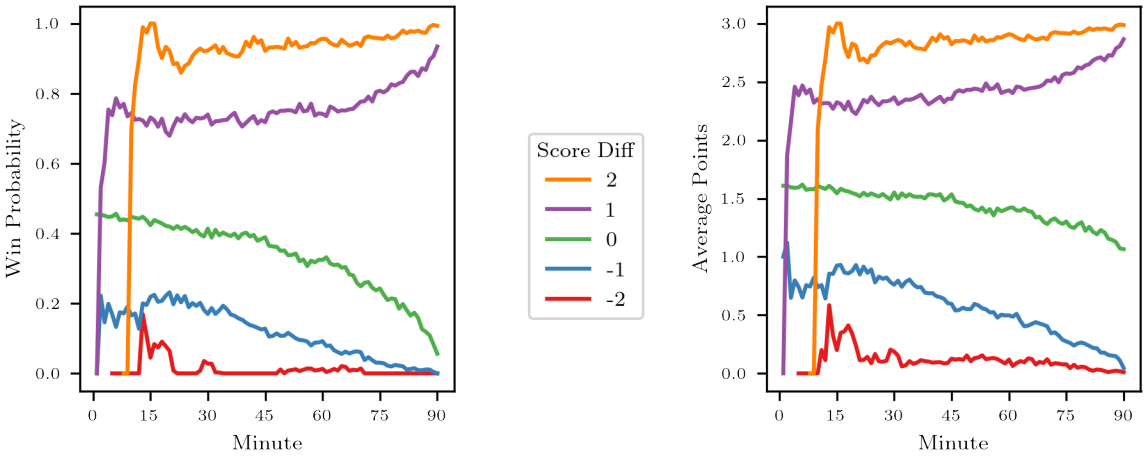

Figure 1: The effect of minute and score difference on match results.

With a small number of features, a common technique to estimate probabilities is logistic regression. However, logistic regression assumes that the log-odds of the outcome is a linear

function of the input variables, and this assumption does not hold for this dataset. Figure 1 shows that the effects of time on both home team win probability and average points (which is calculated as $3 \cdot \mathbb{P}(\text{Win}) + \mathbb{P}(\text{Draw})$) are different depending on the difference in score.

While logistic regression could work in theory, it would require complex feature engineering to determine the appropriate transformations, interactions, etc. required for an accurate model. Neural networks were also considered, due to their suitability for multi-class problems such as this one, but were discarded due to the small set of input features and for the lack of desire to determine the correct architecture. Ultimately, the decision was made to use random forests, for their ability to learn complex decision boundaries, and because they have a natural ability to output probabilities.

3.2 Objective Function

Common metrics to evaluate the performance of a classification model include accuracy, along with other functions of class predictions, such as precision or recall. However, the goal of this paper is not to accurately predict the class (i.e. the outcome of each match), but rather to accurately estimate the *probabilities* of each outcome given the input variables. Put another way, let's say there is some state in a match for which the model returns a probability of a home win as 60%, a draw as 25%, and a loss as 15%. If all such matches with that state resulted in a win for the home team, the model would have perfect accuracy. However, if this was the case, then the estimated 60-25-15 split amongst the outcomes is probably not accurate—the model should have predicted a higher probability of a win.

Thus, instead of accuracy, the metric used to determine the best model will be cross-entropy loss, traditionally used in logistic regression. Cross-entropy loss has the property that it is minimized when the predicted probabilities of each outcome equal the "true" probabilities, so minimizing the empirical cross-entropy loss means the predicted probabilities will be good estimates of the true probabilities.

$$L\left(f\left(X_i\right), y_i\right) = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i \left(- \sum_{r} \log\left(\mathbb{P}\left(r | X_i\right)\right) \mathbb{I}\left(y_i = r\right) \right)$$

$$w_i = \frac{1}{\sum_{j=1}^{n} \mathbb{I}\left(\text{matchId}(j) = \text{matchId}(i)\right)}$$

Here, X_i is a given match state, y_i is the result of the match, and r is one of three values: "win," "draw," or "loss." The w_i are weights assigned to each event, chosen such to ensure that each *match* is given the same weight in the training process, as opposed to each event. In other words, if a match has a high density of events, each individual event within that match will be given a lower weight, and vice versa for matches with very few events.

One final concern is that unlike in logistic regression, which wraps its output inside of the sigmoid function, the probabilities returned by random forests may equal 0 exactly (or 1), in which case the cross-entropy loss is undefined. In both cases, the probabilities are manually set to the next closest output of the model—e.g. a probability of 1 becomes .9987.

3.3 Feature Representation

The data contain the following features:

- **Position**: `x` and `y`, denoting the coordinates of the ball at the start of the event, with each element ranging from 0 to 100. A larger `x` means the ball is closer to the goal the home team is attacking, and a larger `y` means it is closer to the right side of the home team's attack.
- **Possession**: `home` is a variable taking on three values: 1 indicates the home team is in possession, 0 indicates the away team is in possession, and -1 indicates no team is in possession (e.g. before a duel).
- **Live/Dead Ball**: `ballNum` is a categorical variable describing the status of play, see Section 2.1.
- **Red Cards**: `menDiff` is a variable indicating how many more men on the field the home team has. Usually 0, but varies if one or more players have been shown a red card.

Unfortunately, `scikit-learn`, the software used to fit the models, is unable to directly handle categorical variables, so both the Live/Dead Ball feature and the Possession feature were one-hot encoded. There also remain a couple questions over the representations of the other features:

- **Time**: The cleaned dataset contains a `seconds` variable, taking on values from 0 to 5400, with precision down to the microsecond. Alternatively, time could be represented as a binned `minute`, taking on discrete integer values from 1 to 90.
- **Score**: There is a choice between feeding the random forest two columns—one containing the score of the away team, and the other the score of the home team—or a single column of the difference between the two scores.

Theoretically, these questions should not matter too much for a random forest, as if the simpler representation is sufficient, the algorithm should be able to learn the simpler decision boundary even if it is fed the more complicated features. However, with only a limited dataset, it is possible that using the more complicated features could lead to overfitting. Therefore, cross-validation is employed to choose which is appropriate, using the splitting method described in Section 2.2. For each fold, the model was trained on all events not in the fold. Then, predicted probabilities were calculated for all events not in the fold, which were then used to calculate the empirical loss, as described in Section 3.2.

3.4 Hyperparameter Tuning

In addition to feature representation, there are a number of hyperparameters that need to be tuned. Four hyperparameters were selected for adjustment:

- `max_samples`: The proportion of training data points to consider when building each individual decision tree.

- `n_estimators`: The number of decision trees from which to create the forest.
- `max_depth`: The maximum depth of each individual decision tree.
- `max_features`: The maximum number of features to consider when calculating the best split at each level of the tree.

All other hyperparameters were left to the default settings of `scikit-learn`. Additionally, as in the objective function, each data point was given weight w_i, to be used by the random forest algorithm, and a random state was set for each model.

To ensure efficient use of computing resources, multiple rounds of hyperparameter grid search were conducted, in parallel on a high-performance computing cluster. Each round built on top of the previous by zooming in on good candidate values, and increasing the number of estimators.

1. Varied representations of both time and score. Five candidate values of `max_samples`, from .025 to .225. `N_estimators` of 10 and 100. Five candidate values of `max_depth`, from 10 to 90 (with 90 effectively corresponding to no depth limit). Five candidate values of `max_features`, from 2 to 14.
2. Fixed the representation of score to a single feature representing difference, and `n_estimators` to 100. Varied representation of time. Three candidate values of `max_features`, from 4 to 6. Four candidate values of `max_samples`, from .01 to .07. Three candidate values of `max_depth`, from 5 to 15.
3. Fixed `max_depth` to 10. Varied the representation of time. Three candidate values of `n_estimators`, from 500 to 1500. Five candidate values of `max_samples`, from .01 to .05. `Max_features` of 5 and 6.
4. Fixed the representation of score to a single feature representing difference, and `max_features` to 6. Five candidate values of `n_estimators`, from 2000 to 10000. Five candidate values of `max_samples`, from .001 to .009.

3.5 Chosen Model

The model with the best average empirical loss across all five folds was the random forest with `n_estimators` equal to 10000, `max_samples` equal to .003, a `max_depth` of 10, `max_features` equal to 6, and score represented by a single "difference" feature. The time representation did not affect model performance too much and was arbitrarily fixed to continuous seconds.

This model was then trained on the entire dataset (with `max_samples` $= .003 \cdot \frac{4}{5} = .0024$ to account for the larger dataset), and the fitted probabilities were returned. While it is likely the model could be further improved by increasing the number of trees, computing resources are limited so the decision was made to stop at 10000.

4. Results

4.1 Tracking Matches

Figure 2 shows the expected points earned by each team, throughout two selected matches. As a reminder, expected points is equal to:

$$\mathbb{E}\left(\text{Points}|X_i\right) = \sum_r \text{Points}\left(r\right) \cdot \mathbb{P}\left(r|X_i\right) = 3 \cdot \mathbb{P}\left(\text{win}|X_i\right) + 1 \cdot \mathbb{P}\left(\text{draw}|X_i\right) + 0 \cdot \mathbb{P}\left(\text{loss}|X_i\right)$$

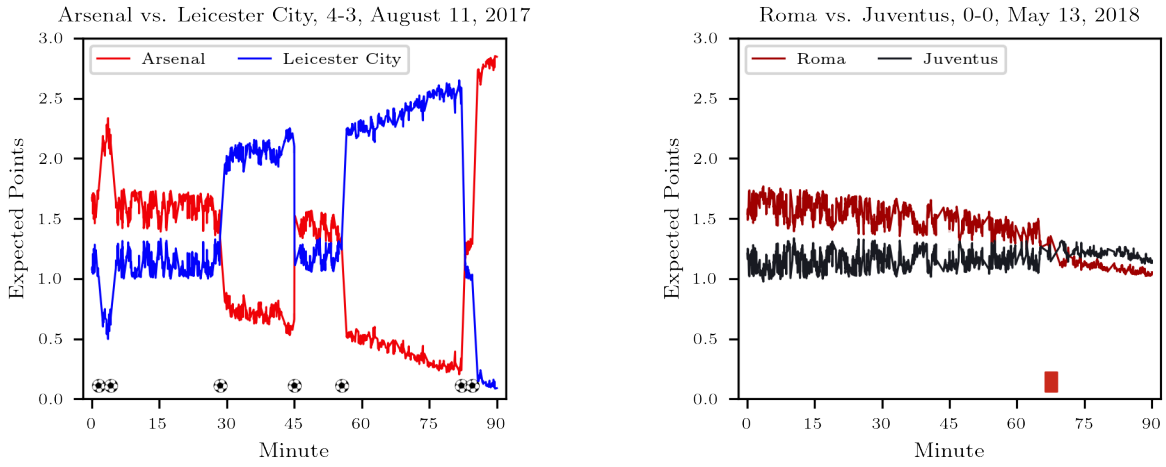

Figure 2: Expected point totals throughout two selected matches.

The first match was a topsy-turvy affair, with the home team Arsenal scoring in the second minute, significantly boosting its expected point total. However, the away team, Leicester City, equalized soon after, effectively returning the match to its original state. Note that the home team begins with a larger expected points total, due to the nature of home-field advantage. The two teams then trade three goals, with Leicester gaining the advantage, losing it, and then gaining it back again. Note that the gain in expected points for Leicester is greater for its third goal than its second, as a one-goal lead becomes more valuable later in the match. Arsenal then score two late goals in quick succession to flip the match in its favor—however, at 90 minutes, Arsenal's expected points total is not exactly 3. This is because all stoppage time events are truncated to the 90th minute, thus the model always assumes there is some amount of stoppage time left to play.

In the second match, on the right, Roma and Juventus play to a 0-0 draw. Roma starts with the advantage as it is the home team—however, this advantage diminishes over the course of the match. Around the 68th minute, a Roma player gets a red card, which means Juventus becomes the more likely team to win. Here, note that each team's line is not simply a mirror image of the other, as the likelihood of a draw increases as the game goes on, which is further exacerbated by the red card.

It is reassuring that these plots look the way they do. Obviously, goals should increase a team's expected points total, and red cards should decrease it. The effects of time on

the probabilities of each outcome also resemble the pattern shown way back in Figure 1. Curiously, there is a lot of variation from each event to the next, likely determined by the position of the ball on the field and which team has possession. Whether or not these features actually have the magnitude of effect shown here, is unclear—overfitting remains a possibility.

4.2 Importance of Ball Position

To follow up on the previous point, shown in Figure 3 are the expected point totals for the home team depending on where the ball is on the pitch and who has possession, in a tie game one minute into the second half.

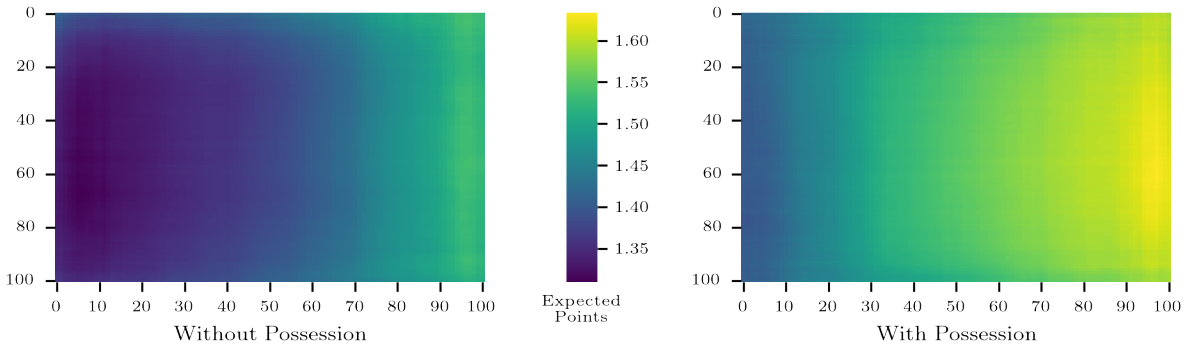

Figure 3: Expected points and ball position. The home team attacks to the right.

Again, this corresponds with common belief—the team with possession has a larger expected points total, and the closer the ball is to the away goal, the better off the home team is. Perhaps surprisingly, these heatmaps are fairly smooth, indicating that the model variance is low and that the concerns over overfitting expressed in the previous section might be misplaced.

4.3 Expected Points Added

With estimates of expected points for each match state, it is now possible to calculate expected points added (henceforth referred to as xPA) for each player. The calculation is simple—for each event, the player(s) involved get credited with the difference in their team's expected points total from before the event to after the event. The results are highly surprising. It turns out that goalkeepers and defenders are credited with large positive xPA totals, and forwards and midfielders receive large negative xPA.

Table 1 shows the players with the largest xPA and lowest xPA aggregated over the season, filtered to the English Premier League only:

Player	Position	xPA	Player	Position	xPA
Nick Pope	GK	30.00	Christian Eriksen	MF	-60.76
David de Gea	GK	27.99	Wilfred Ndidi	MF	-54.60
Ederson	GK	25.58	Kevin De Bruyne	MF	-53.33
Thibaut Courtois	GK	21.71	Aaron Mooy	MF	-52.17
Jordan Pickford	GK	19.63	Jack Cork	MF	-50.49

Table 1: Players with highest and lowest xPA, English Premier League, 2017-18

There appears to be a simple explanation for this phenomenon, and that is that forwards and midfielders play higher up the pitch, and lose the ball quite frequently through imperfect passes and misplaced shots. Figure 3 shows how important possession is, especially close to goal, so a shot from within the penalty box that gets saved and held by the goalkeeper has a big effect on expected points. Defenders and goalkeepers are the beneficiaries of these actions, which is why their xPA totals tend to be large.

Another semi-related cause is that the effects of individual actions tend to be more negative than positive. Out of more than 2.8 million player actions (with actions related to two players counted twice), more than 55% resulted in a negative xPA. This likely results from the setup of xPA—if a player attempts a pass which is intercepted by the opposition, the player making the pass will lose xPA, but the opposing player will not gain anything.

5. Future Considerations

5.1 Probability Estimation

There are a number of improvements that could be made to the probability estimation model, most of which having to do with introducing additional features.

1. The teams playing the match. For example, a match between the team in first place and the team in last place should start with win probabilities skewed towards the team at the top of the table.

2. The player(s) involved in each action. Lionel Messi with the ball in the opposition penalty box is probably better for the attacking team than if another player had the ball in the same position.

3. The positions of all players on the field, not just the player with the ball. Such information would allow the model to distinguish between scenarios where a team is attacking against a set, low-block defense, and those where a team has the ball in the same position but is counter-attacking against a stretched defense.

4. Extend the scope to include tournament matches and games that go to extra time.

However, it is important to recognize that the inclusion of these additional features would require vastly more data than what is currently publicly available, in order for the model to generalize well. Even with over 1,800 games, the model would overfit when presented with the score as a two-dimensional feature rather than the single "difference" feature. The data used here may seem quite large, but there are plenty of reasonable match states that rarely

occur in the dataset, leading to high model variance. One example is that the only match in the data for which the home team was up two goals in under 9 minutes actually resulted in a loss for the home team, which is why it is imperative that there be a large enough sample of the many different states in order to accurately estimate win probabilities.

5.2 xPA Calculation

Unfortunately, attempting to use expected points added as a metric to evaluate player performance was unsuccessful. The tallies are almost nonsensical—Wilfed Ndidi accrued -54 xPA over the 2017-18 season, but his team, Leicester City, only earned 47 points. The calculation as designed gives all the credit to the player who initiated the action, yet passes, tackles, and the like usually involve two players, not one. Perhaps it is not fair to put all of the weight of an intercepted pass on the passer himself, when the intended recipient was easily outmuscled by his opponent.

There is also a lingering question over whether or not trying to determine the importance of a single pass or run is even a task worth undertaking in the first place. Even if the xPA totals were "accurate," they could still be misleading without context. A player who constantly turned the ball over for 89 minutes, but scored a tap-in for a late stoppage-time winner may well end up with a positive score, even though anyone watching the game would likely say he had a poor performance.

Measuring xPA also implies that maximizing expected points is always the best course of action, but this is not the case. In the 0-0 draw between Roma and Juventus from Figure 2, one point was all Juventus needed to clinch the title, and one point was all Roma needed to qualify for the Champions League. In this case, both teams were primarily interested in simply not losing, and were unwilling to take risks that may have increased the possibility of doing so.

However, such calculations are not entirely useless. xPA allows for weighting the relative importance of big match events, like goals and red cards. In Roma-Juventus, Radja Nainggolan's red card decreased Roma's expected points total, but the consequences were far less than if Roma had conceded a goal instead. In the Arsenal-Leicester City matchup, Olivier Giroud's late winner was worth far more to his team than Alex Lacazette's opener. These analyses might just be regurgitating what soccer fans already know, but it allows the fan to *quantify* such moments.

6. Conclusion

This paper presented a model to estimate the probability of a soccer match ending in each of three possible results, using a random forest trained on only a small set of features. Upon inspection, these probabilities matched up well with expectations, and further analysis could incorporate additional features to determine even better estimates. These probabilities were then used to create the metric of expected points for each team, which could potentially be extended to estimating the probability of season outcomes, such as a team winning the league, or avoiding relegation. Another motivation behind determining such probabilities was to create a metric of "expected points added" to measure individual performance, yet this ended up being biased towards defensive positions and unfairly penalizing attacking players.

References

Luca Pappalardo, Paolo Cintia, Alessio Rossi, Emanuele Massucco, Paolo Ferragina, Dino Pedreschi, and Fosca Giannotti. A public data set of spatio-temporal match events in soccer competitions. *Scientific Data*, 6(1):236, October 2019. ISSN 2052-4463. doi: 10.1038/s41597-019-0247-7. URL `https://doi.org/10.1038/s41597-019-0247-7`.