Interpretable Prediction of Goals in Soccer

Tom Decroos
KU Leuven, Department of Computer Science
tom.decroos@cs.kuleuven.be

Jesse Davis
KU Leuven, Department of Computer Science
jesse.davis@cs.kuleuven.be

Abstract

Valuing the actions a soccer player performs in a match is a crucial problem in soccer analytics. While many approaches have been proposed for this problem, a commonality among them is the need to build a model that can predict for a given game state the probability of a goal occurring in the near future. Often these works have two common shortcomings. First, the predictive models are often not thoroughly evaluated or may even be evaluated according to the wrong performance metric. Second, there is a tendency to sacrifice interpretability for performance. Hence, the models often yield no insight into why a given game state has a higher or lower probability of resulting in a goal. This paper analyzes VAEP, a recently proposed approach for valuing actions, and its model for estimating the probability of scoring in the near future. We discuss a number of design choices related to building this model and share insights on how to properly evaluate it. Finally, we replace VAEP's complicated non-interpretable gradient boosting tree model that uses 151 features with a simpler interpretable Generalized Additive Model (GAM) using only 10 features. We find that the GAM offers nearly identical performance to the more complicated gradient boost model while being interpretable and offering insights into what characteristics of a game state have an effect on the probability of scoring a goal in the near future.

Introduction

Objectively assessing how valuable an action in a soccer match is, is a crucial task in soccer analytics that has applications in scouting, player acquisition, and tactical analysis. Initial metrics such as expected goals (xG) (IJtsma 2015) and expected assists (xA) (Worville 2017) take a probabilistic approach to valuing the actions: xG estimates the probability that a shot will result in a goal and xA estimates the probability that a pass will become a goal assist. Unfortunately, these metrics focus on relatively rare actions within a match.

More recently, approaches like VAEP (Decroos et al. 2019), XG Chain (Lawrence 2018), xG Added (Mackay 2017), xG Threat (Singh 2019), and Attacking Contribution (Yam 2019) have attempted to assess a wider variety of on-the-ball actions in a match. These approaches give a more complete overview of a player's contribution by valuing frequent actions such as (non-assist) passes, dribbles, and tackles. While there are important differences among these techniques, at a high level they all value actions by exploiting the intuition that an action changes the game state. Then, they value each action by looking at the difference in a team's chance of generating a goal scoring opportunity before and after the action. Therefore, the key artificial intelligence task is to estimate a team's probability of scoring in the near future from a given game state.

Some techniques, such as expected threat (xT) (Singh 2019) derive this change in probability solely by valuing individual locations on the pitch. Ignoring relevant information such as the action used to progress the ball and the game context (e.g., score difference and time remaining) allows xT to give intuitive insights into why certain actions are useful. Simply stated, good actions move the ball to a location on the pitch where a team is more likely to generate a goal-scoring attempt. In contrast, other approaches such as VAEP use a rich feature set to describe the current game state and complex black-box models like ensembles of decision trees to estimate probabilities. A shortcoming is then that there is no intuitive explanation about why a given action altered a team's chance of scoring.

This paper examines VAEP in greater detail in hopes of gaining insights into how it assesses changes in the probability of a team scoring. The original paper used a gradient boosting model with over 150 features which is not interpretable. We replace the gradient boosting model with a Generalized Additive Model (GAM) using only 10 features. We find that the GAM offers nearly identical performance to the more complicated gradient boosting model while being interpretable and offering insights into what characteristics of an action alter a team's probability of scoring.

VAEP: Valuing actions by estimating probabilities

This paper analyzes the VAEP framework (Decroos et al. 2019). We briefly summarize the approach here. Like other approaches, VAEP exploits the fact that each action alters the game state. That is, an action a_i moves the match from game state S_{i-1} to game state S_i. The value of a_i for team t

is then the difference in value between S_i and S_{i-1}:

$$V(a_i, t) = V(S_i, t) - V(S_{i-1}, t)$$

where t is the team the value is computed for. When considering how valuable a game state is, VAEP leverages the intuition that a game state is valuable for team t if it offers (1) a high short-term probability of team t scoring; and/or (2) a low short-term probability of team t conceding. Therefore, VAEP values each game state according to the formula:

$$V(S, t) = P_{scores}(S, t) - P_{concedes}(S, t).$$

Hence a game state is positively valued if a team's short-term chance of scoring is higher than conceding and negatively valued if the opposite is true.

Conceptually, VAEP transforms the subjective task of valuing an action to the objective machine learning task of estimating the probabilities of future events (in this case scored and conceded goals). The machine learning task can then be summarized as follows:

Given: Game state S_i and team t.

Predict: $P_{scores}(S_i, t)$ and $P_{concedes}(S_i, t)$, the probabilities of team t scoring and conceding in the short-term future of game state S_i.

We can build a predictive model by training a probabilistic classifier on a set of features and labels. In the next section, we discuss the design choices that come up when building this model.

Building a predictive model for goals

To build a predictive model that can estimate the short-term probability of scoring and conceding from a game state, we require three ingredients: (a) features that describe the game state, (b) labels that capture the limited temporal influence that the current game state will have on the match's evolution, and (c) a probabilistic classifier that can learn the probabilistic mapping from the features to the labels.

Features

Constructing the right features is a crucial step in the process of valuing game states. By selecting and engineering specific features that accurately describe the pertinent aspects of the game state, we can increase the performance of the predictive model. It also accords the modeler the ability to decide which aspects of the game state to analyze and which aspects to ignore. For example, xT (Singh 2019) discretizes the pitch into zones and only uses the zone where the current action occurred to describe the game state. Ignoring the other aspects of the game state is a conscious choice that makes modeling simpler. Moreover, it makes the model much more understandable to humans. However, this choice may come at the expense of maximizing performance.

On the other hand, VAEP attempts to make the predictive model as performant as possible and therefore uses a rich set of features to describe a game state. More specifically, VAEP considers the following three types of features per game state.

Simple features VAEP describes a game state S_i by its three most recent actions $[a_i, a_{i-1}, a_{i-2}]$. To describe an action, Decroos et al. introduced the SPADL format for event stream data, which describes an action by the following eight attributes: *type*, *player*, *team*, *result*, *bodypart*, *time*, *start location*, and *end location*. All these attributes (except player and team) of the past three actions are used as features to describe the game state S_i.

Complex features The complex features combine information within an action and across consecutive actions. Within each action, these features include (1) the distance and angle to the goal for both the action's start and end locations, and (2) the distance covered during the action in both the x and y directions. Decroos et al. also compute the distance and elapsed time between consecutive actions and whether the ball changed possession. These features provide some intuition about the current speed of play.

Game context features The game context features are (1) the number of goals scored in the game by the team possessing the ball after action a_i, (2) the number of goals scored in the game by the defending team after action a_i, and (3) the goal difference after action a_i. Decroos et al. include these features because teams often adapt their playing style to the current scoreline (e.g., a team that is 1-0 ahead will play more defensively than a team that is 0-1 behind) (Robberechts, Van Haaren, and Davis 2019).

By considering all the described above features and one-hot encoding the categorical variables, Decroos et al. used 151 features in total to describe a game state (Decroos et al. 2019).

Labels

By assigning labels to game states, we answer the following question:

> How responsible is a game state for a goal scored or conceded in the near future?"

If a goal occurs in the subsequent game state, then obviously the current game state should receive a lot of credit. However, if the goal happens ten minutes after the current game state, then the current game state likely had no role in leading to the goal. The challenge is to assign a label for the game state that falls between these two more extreme cases. To estimate $P_{scores}(S_i, t)$, Decroos et al. assign game state S_i a positive label ($= 1$) if team t scored a goal in the subsequent 10 actions, and a negative label ($= 0$) in all other cases (Decroos et al. 2019). The same approach is used to assign labels to estimate $P_{concedes}(S_i, t)$. Choosing to look ahead 10 actions into the future to determine whether a game state affected the occurrence of a scored or conceded goal, is a parameter of the approach and can be altered depending on the preferences of the end user.

Probabilistic Classifiers

To estimate scoring probabilities, we need a probabilistic classifier that can predict the labels from the features. We

discuss two popular models, logistic regression and XGBoost, and one lesser known model, the generalized additive model.

Logistic regression Logistic regression is a statistical model that uses a logistic function to model a binary target variable that depends on the input features (Pedregosa, Varoquaux, and others 2011). Given an input feature vector $x = [x_1, x_2, \ldots, x_m]$ and a target variable y, logistic regression will learn the following function:

$$g(E[y]) = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_m x_m = \alpha_0 + \sum_i^m \alpha_i x_i$$

where g is the *logit* link function and the weights α_i are learned from the training data. This formula illustrates how logistic regression is a linear model. It is also interpretable, as the sign of α_i shows whether there is a positive or negative correlation between feature x_i and y and the magnitude of α_i hints how big its impact is.

XGBoost XGBoost is a popular gradient boosting decision trees model that solves many data science problems in a fast and accurate way (Chen and Guestrin 2016). Due to its excellent performance, it has become the de facto standard model of choice for many data science practitioners when classifying tabular data. One of the reasons XGBoost works so well out of the box is that by using decision trees and the boosting mechanism, it can learn complex non-linear decision boundaries. This is why a gradient boosting decision tree model was the chosen model in the original VAEP paper (Decroos et al. 2019). The downside is that these complex non-linear decision boundaries are often too complex for humans to grasp and thus XGBoost is in practice a black box model.

Generalized additive model Generalized additive models (GAMs) are statistical models that model the target variable as a sum of univariate functions (Hastie 2017). Standard GAMs have the form

$$g(E[y]) = f_1(x_1) + f_2(x_2) + \ldots + f_m(x_m) = \sum_i^m f_i(x_i)$$

where g is the *logit* link function and the functions f_i are learned from the training data. This model is interpretable in the sense that users can visualize the relationship between the univariate terms of the GAM and the dependent variable through a plot $f_i(x_i)$ vs. x_i. The formula illustrates how GAMs are a more powerful generalization of logistic regression models, replacing the linear functions $\alpha_i x_i$ with more complex non-linear functions $f_i(x_i)$.

One of the reasons why GAMs have been less popular up until now is lack of a mature and widely available implementation. However, Microsoft recently released InterpretML (Nori et al. 2019), an open-source Python package which exposes interpretable machine learning algorithms to practitioners and researchers. One of the contributions in InterpretML is the first implementation of the Explainable Boosting Machine (EBM). EBM is an implementation of a GAM that uses a boosting mechanism to learn the univariate functions and can also model the most important pairwise interactions among features. EBM is interpretable, yet can be nearly as accurate as many blackbox models such as XGBoost.

Evaluating a predictive model for goals

When building a predictive model, two underappreciated aspects in soccer analytics are (1) evaluating the performance of the predictive model, and (2) choosing the correct evaluation metric. In this section, we first identify shortcomings in recent works of various soccer analytics researchers. We discuss AUROC, Brier score, and logarithmic loss, the three most popular metrics for evaluating probabilistic classifiers. We offer some insights on when to use each metric and finally propose a small modification to the Brier score and logarithmic loss so that they become more interpretable.

Shortcomings in recent work

Often, articles do not provide a detailed description of both the methodology employed in the evaluation and the results themselves. Fernandez et al. introduced an Expected-Possession-Value-model for soccer that is composed of multiple smaller models that estimate the likelihood of future events such as goals or passes (Fernández, Bornn, and Cervone 2019). The paper mentions that they rigorously evaluated and tuned their models to obtain maximal performance. However, further details on how accurate the models were or which evaluation metric was used were omitted for the sake of brevity, which is detrimental to the reproducibility of their results. Similarly, Decroos et al. predict the likelihood of a goal occurring from a phase using a Dynamic Time Warping-based model, but do not report the performance of the model or the evaluation metric used to set its parameters (Decroos et al. 2017b). Many more soccer analytics articles exist where information on the evaluation approach is missing (IJtsma 2015; Singh 2019; Lawrence 2018; Mackay 2017).

Other articles do provide details on how the models were evaluated, but actually use the wrong evaluation metric. For example, Pappalardo et al. build a model that predicts the probability of a team winning a match based on some features describing the team and evaluate this model with AUROC (Pappalardo, Cintia, and others 2018). The predictive model is then used to assign weights to its features in a different use case, namely valuing players. Given this use case, it is important that the predictive model is well calibrated. However, AUROC is an evaluation metric that is agnostic to calibration. Another example is Decroos et al. who build a model to predict highlights in soccer matches (Decroos et al. 2017a). One of the components in this model is an expected goals model that predicts goals from shots. Decroos et al. used AUROC to evaluate their expected goals model. However, given how the output of their predictive model is used, they should have used logarithmic loss instead, as we will argue further in this section.

Finally, Lucey et al. also build models that predict goals from shots and use mean absolute error (MAE) to evaluate the performance of their predictive models (Lucey et al.

2014). However, mean absolute error is not a proper scoring rule. What this means is that a predictive model that is evaluated on mean absolute error can get better results by reporting probabilities of 100% or 0%, depending on which is closer to the real probability. The predictive model is incentivized to lie rather than report the true class distribution (Gneiting and Raftery 2007).

Luckily, there also exist articles where the evaluation approach is mentioned, motivated, and quantitative results are provided (Eastwood 2015; Decroos et al. 2019; Mackay 2017).

Evaluation metrics

There exist three popular metrics to evaluate probabilistic classifiers: area under the ROC curve (AUROC), Brier score and logarithmic loss.

AUROC The area under the receiver operator curve (AUROC) evaluates how well a classifier can differentiate positive examples from negative examples. Intuitively, AUROC answers the following question: "Given a positive example and a negative example, how likely is it that our classifier will correctly rank the positive example ahead of the negative example". Note that even random guessing will achieve a AUROC of 50%. This presents a naive baseline any probabilistic classifier should always beat. One crucial aspect of AUROC that often goes ignored is that it is in essence a ranking metric. AUROC only considers the relative ranking of examples and ignores the actual predicted probabilities. This means that a classifier can be poorly calibrated, yet still achieve great AUROC.

Brier score The Brier score (BS) is a proper scoring rule that measures the accuracy of probabilistic predictions. A proper scoring rule is a metric that can only be minimized by reporting the true class distribution. It is essentially the mean squared error between the predictions and the labels and has the following formula:

$$BS = \frac{1}{N}\sum_{i}^{N}(p_i - y_i)^2$$

in which N is the number of examples, p_i is the probability that was predicted for example i and y_i is the label of example i.

Logarithmic loss The logarithmic loss (LL) is also a proper scoring rule that measures the accuracy of probabilistic predictions. The biggest difference with Brier score lies in the way that it weighs individual prediction errors. Logarithmic loss has strong foundations in information theory and its formula is:

$$LL = \frac{1}{N}\sum_{i}^{N} y_i \log p_i + (1 - y_i)\log(1 - p_i)$$

.

When to use which evaluation metric

As previously discussed, a common shortcoming in soccer analytics research is building predictive models by optimizing and evaluating on one of the above metrics without critical thought on why to use a specific evaluation metric (Pappalardo, Cintia, and others 2018; Decroos et al. 2017a; Lucey et al. 2014). Our key message is that the choice of evaluation metric should depend on the specific use case in which the predictive model will be used. In other words, once you have a predictive model that outputs probabilities, what will you do with these probabilities? We now discuss our insights on when each metric is applicable.

Choosing whether or not to use AUROC is the easiest choice. AUROC is the best metric for classification tasks or ranking examples based on how likely they are to be positive or negative. For example, when searching for the top-k game states that are most likely to result in a goal.

When we care about using the actual values of the probabilities, the choice is between the Brier score and logarithmic loss as AUROC is not suitable. Unfortunately, it is less clear when one should use the Brier score versus logarithimc loss. Brier score and logarithmic loss are similar in the sense that they are both proper scoring rules and can both only be minimized by reducing the individual prediction errors. However, they differ in how they aggregate the individual prediction errors.

To illustrate this difference and to more easily compare the two metrics, let $e_i = |p_i - y_i|$ be the prediction error for example i. Using this definition and the multiplication rule for logarithms, we can simplify the formulas for the Brier score and logarithmic loss to:

$$BS = \frac{1}{N}\sum_{i}^{N} e_i^2$$

and

$$LL = \frac{1}{N}\sum_{i}^{N}\log(1 - e_i) = \frac{1}{N}\log(\prod_{i}^{N} 1 - e_i).$$

This rewrite illustrates that the Brier score is simply the mean squared error. Moreover, the Brier score combines individual prediction errors by summing them while the logarithmic loss combines individual prediction errors by multiplying them.

This insight is the reason we recommend to use Brier score to build a predictive model if the resulting probabilities will be summed or subtracted. For example, (Decroos et al. 2019) construct player ratings by summing the deltas between game state probabilities. We recommend to use logarithmic loss if the resulting probabilities from the predictive model are more likely to be used in multiplications, such as in (Decroos et al. 2017a) and (Fernández, Bornn, and Cervone 2019), where the resulting probabilities are multiplied with the probabilities of predictive models of other tasks. Other use cases where probabilities are often used in multiplications are simulations, reinforcement learning, and recommender systems.

In summary, which evaluation metric to use depends on what the probabilities outputted by the predictive model will be used for. We recommend to use AUROC when *ranking* probabilities or *classifying* examples, Brier score when *summing* or *subtracting* probabilities, and log loss when *multiplying* or *dividing* probabilities.

Making Brier score and logarithmic loss more interpretable

A downside to both Brier score (BS) and logarithmic loss (LL) is that their values are less interpretable than AUROC. Regardless of the class skew, an AUROC of 0.5 corresponds to random guessing. Moreover, the AUROC is the Wilcox-Mann-Whitney statistic, which is the probability that the model ranks a randomly selected positive example ahead of a randomly selected negative example. In contrast, how good or bad a specific BS or LL value is depends on the class distribution. A Brier score of 0.1 is impressive in a data set with a 50/50 class distribution, but terrible in a data set with a 99/1 class distribution.

To somewhat combat this lack of interpretability, we can compare the BS or LL of our predictive model to that of a simple baseline, namely always predicting the class distribution. For example, in a data set with only 1% positive examples, we always predict 0.01 as the chance of example i being positive. In our experience, this can be a surprisingly hard baseline to beat in some use cases.

To properly compare our model's BS/LL, we divide it by the BS/LL of our baseline. This number will be a value between 0 and infinity. A value of 0 means that this classifier offers perfect (and thus deterministic) predictions. A value between 0 and 1 means that this classifier offers better predictions than the naive baseline, and a value higher than 1 means that the model is worse than the naive baseline and practically useless, similar to a classifier with a AUROC of $< 50\%$. We call these metrics the normalized Brier score (NBS) and normalized logarithmic loss (NLL).

$$NBS = \frac{BS}{BS_{baseline}}, NLL = \frac{LL}{LL_{baseline}}$$

Experiments: Predicting the probability of scoring in the 2018/19 Premier League

The goal of the experiments is to understand the effect of the interplay between

1. the complexity of the feature set used to describe the game state; and
2. the selected probabilistic classifier used to estimate scoring probabilities for each game state.

Our hope is that we can approximate the performance of the original VAEP model (Decroos et al. 2019), which used 151 features and an uninterpretable gradient boosted tree model by applying a more interpretable model to a much smaller feature set.

Approaches considered

We consider the following three sets of features:

Location only This considers only the (x, y)-coordinates of the last action in game state S_i. Hence, each game state is described by two features.

VAEP This considers the set of 151 features used in the original VAEP model.

Top-10 This considers the 10 most important features from VAEP feature set. These features were selected using the built-in ordering of feature importance available in the implementations of the XGBoost and GAM models.

For each of the three feature sets, we train a logistic regression, XGBoost and GAM model.

Methodology

Our data set consists of event stream data of 760 matches from seasons 2017/18 and 2018/19 of the English Premier League. The data was provided to us by StatsBomb and then converted to the SPADL format using the freely available converter at https://github.com/ML-KULeuven/socceraction.

We trained all three classifiers on 747,813 game states in the 2017/18 Premier League season and evaluated them using the Normalized Brier Score (NBS) on 789,108 game states in the 2018/19 Premier League season.

For all three classifiers, we performed no tuning and set all parameters to the default values of their respective implementations.[1][2][3] Some examples of these default parameters are logistic regression using the L2 regularization penalty and L-BFGS as the optimization problem solver, XGBoost using 100 trees of maximum depth 6, and the GAM using 16 estimators to construct each univariate function. The only exception is that we allowed the GAM to learn three pairwise interaction terms rather than its default value of zero. This parameter was changed to leverage the capabilities of the underlying implementation of the GAM.

Results

Table 1 reports the normalized Brier scores for each classifier-feature set combination. From these results, we can infer the following conclusions:

Only considering location is insufficient. As can be seen in the first row of Table 1, logistic regression achieves an NBS of 98.4%, while both XGBoost and GAMs achieve a NBS of 96.4%. These scores only slightly improve upon the baseline. Furthermore, regardless of the learning method, only using the location does not come close to matching the performance of using a more expansive and expressive feature set.

Having a model that captures non-linearities helps. Regardless of the feature set, XGBoost and GAMs offer substantial improvements on predictive performance compared to using logistic regression. For the location only feature set, Figure 1 clearly demonstrates how GAMs are capable of capturing non-linear correlations between the features and the target variable, while Logistic Regression is not.

A small feature set can yield excellent performance. Compared to seeing the full set of 151 features (second row in Table 1), using the top 10 features only slightly

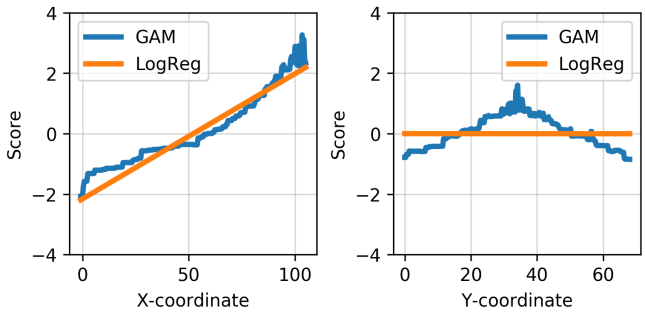
[1] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[2] https://github.com/dmlc/xgboost

[3] https://github.com/interpretml/interpret

Features	Classifier		
	LogReg	XGBoost	GAMs
Location only	98.6%	96.4%	96.4%
VAEP	89.5%	85.6%	85.8%
Top-10	91.2%	86.0%	86.1%

Table 1: Normalized Brier score (lower is better) of three different feature sets using three different probabilistic classifiers: logistic regression (LogReg), XGBoost, and generalized additive models (GAMs).



(a) (b)

Figure 1: A generalized additive model (GAM) consisting of two univariate models using (a) the x-coordinate and (b) the y-coordinate of a game state. The GAM predicts the probability of scoring a goal from a given x, y-location by summing the scores of the univariate models per feature and converting the resulting sum to a probability $P \in [0, 1]$ using the *logit* linking function. The straight orange line represents the weight given to the feature by a Logistic Regression model (LogReg) and illustrates the difference in predictive power.

decreases performance (third row in Table 1). However, using fewer features highly enhance the interpretability of these models. The performance of the GAM (86.1% NBS) is again similar to the performance of XGBoost (86.0% NBS).

In summary, regardless of the feature set GAMs achieve a performance that is better than Logistic Regression and similar to that of XGBoost, while remaining interpretable.

Discussion of top-10 features

Figure 2 details the ten univariate functions that make up the GAM that almost matches XGBoost's performance and provides some insights in what makes a game state (and an action) likely to result in a scored goal. The figure also includes the scores of an example game state to illustrate how a GAM can be used to understand the reasoning behind individual predictions.

Panels (a-c) capture location-based aspects of the game state. In Panel (a), we see that as the ball gets closer and closer to the center of the opponent's goal, the chances of scoring increase. This correlation increases dramatically when the ball is very close to the center of the goal. In Panel (b), we can see that being aligned to the center of the goal slightly increases in the chance of scoring whereas being at a tight angle decreases it. This makes sense, as a shooter likely has more places to aim when positioned in front of the goal. Finally, Panel (c) shows that when the ball enters the final third, there is a strong positive correlation with scoring. This increases as the ball gets closer to the endline behind the opponents goal, but not as dramatically as in Panel (a).

Panels (d-f) capture contextual aspects of the game state. Panel (d) shows how the probability of scoring is dramatically reduced if the last action was not successful. This makes sense, as in the SPADL representation an action not being successful means that the team has lost possession of the ball and therefore cannot attempt to score without first regaining possession.

Panel (e) shows an even stronger impact on goal scoring probability if the last action was a foul. The effect captured here is that while there might still be a slim chance that a team can quickly recover the ball following an unsuccessful action, this becomes virtually impossible if the team committed a foul. The reason is that after a foul the game temporarily suspends and is no longer in open play. This allows the players of the opposing team who are now in possession of the ball enough time to position themselves such that they obtain the maximum tactical advantage.

Panel (f) shows that the probability of scoring varies based on the score difference, with teams leading by ≥ 2 being more likely to score. Robberechts et al. showed that the probability of scoring a goal changes with the goalscore difference (Robberechts, Van Haaren, and Davis 2019). However, we currently are unsure of whether this is a causal effect (i.e., being three goals ahead or tree goals behind has an effect on the mental state of a player, making them perform better or worse actions (Bransen et al. 2019)) or whether this is simply a correlation (i.e., a team that is three goals ahead in a match is a good team with an above average finishing rate, therefore its games states are more valuable). Researching this further is an interesting direction for future work.

Panels (g-j) capture aspects about the speed of play. On Panels (g-i) as the values on the x-axis increase, it indicates that the ball is moving longer distances and hence getting closer and closer to your opponent's goal. In Panel (j), as the value on the x-axis decreases, this indicates that there is less time between consecutive actions. This may be a proxy for the ball moving more rapidly. Hence, in combination, these last four features hint towards the speed of play during the game state. This can be an important factor to decide goal-scoring probability, i.e., the odds of scoring are usually higher during a quick counter-attack than during slow build-up play.

For each of the three location-based features in Panels (a-c), the GAM also learns a pairwise interaction term where it combines each feature with the successfulness of the action in Panel (d). These interaction terms help fine-tune the performance of the GAM for specific examples, but are also more challenging to interpret than the simple univariate functions in Figure 2.

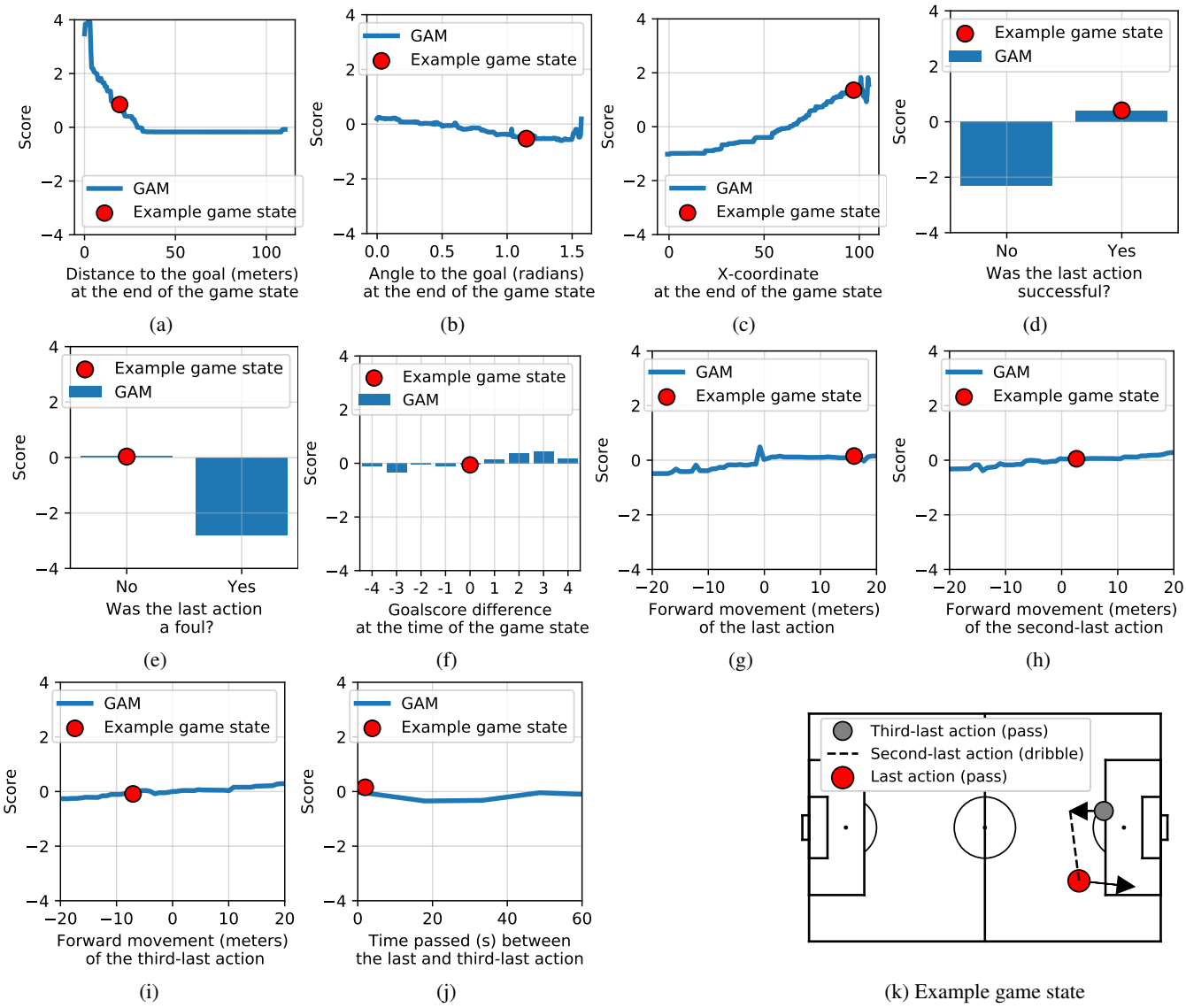


Figure 2: Univariate functions of the ten features (a-j) that make up the GAM that predicts the short-term goal-scoring probability from a given game state. The red dots denote the scores of an example game state (k) with a short-term goal-scoring probability of 4.9%. This goal-scoring probability is computed using the formula $g^{-1}(\sum_i^m f_i(x_i))$ where $f_i(x_i)$ are the scores in Panels (a-j) and g^{-1} is the inverse logit function. The two features that have the biggest positive impact on the goal-scoring probability of the example game state are (a) the small distance to the goal and (c) the high X-coordinate. The feature with the biggest negative impact is (b) the bad angle to the goal.

Conclusion

Assessing how valuable an action in a soccer match is, is a crucial task in soccer analytics. VAEP is a framework that addresses this task by solving a probabilistic classification problem: given a game state, predict the probability of a goal being scored or conceded in the near future. We have discussed a number of design choices related to this classification problem such as the choice of features, labels, and classifier. When building a model to predict goals, earlier work often has two shortcomings. The first shortcoming is that the performance of the predictive model is often not thoroughly or wrongly evaluated. We discussed the occurrence of this shortcoming in recent related work and showed how to combat this by sharing insights on when to use which evaluation metric.

The second shortcoming is that the predictive models use a complicated classifier such as XGBoost that offers no intuitive explanations on why a given game state produced a higher or lower chance of scoring. To address this shortcoming, we replaced the complicated non-interpretable gradient boosting tree model using 151 features from the original VAEP paper (Decroos et al. 2019) with a Generalized Additive Model (GAM) using only 10 features. We illustrated how the GAM can get close to the performance of XGBoost while remaining interpretable. Given how crucial the interpretability of models can be in soccer analytics, GAMs may be a better choice than XGBoost to build predictive models with, even if their performance is slightly worse.

Acknowledgements

Tom Deroos is supported by the Research Foundation-Flanders (FWO-Vlaanderen). Jesse Davis is partially supported by the EU Interreg VA project Nano4Sports, the KU Leuven Research Fund (C14/17/07) and the Research Foundation-Flanders under EOS No. 30992574. Thanks to StatsBomb for providing the data used in this paper.

References

Bransen, L.; Robberechts, P.; Van Haaren, J.; and Davis, J. 2019. Choke or Shine? Quantifying Soccer Players' Abilities to Perform Under Mental Pressure. In *MIT Sloan Sports Analytics Conference*.

Chen, T., and Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. ACM.

Decroos, T.; Dzyuba, V.; Van Haaren, J.; and Davis, J. 2017a. Predicting Soccer Highlights from Spatio-Temporal Match Event Streams. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 1302–1308.

Decroos, T.; Van Haaren, J.; Dzyuba, V.; and Davis, J. 2017b. STARSS: A Spatio-temporal Action Rating System for Soccer. In *ECML/PKDD 2017 Workshop on Machine Learning and Data Mining for Sports Analytics*.

Decroos, T.; Bransen, L.; Van Haaren, J.; and Davis, J. 2019. Actions speak louder than goals: Valuing player actions in soccer. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery; Data Mining*, KDD '19, 1851–1861. New York, NY, USA: ACM.

Eastwood, M. 2015. Expected Goals And Support Vector Machines. pena.lt/y.

Fernández, J.; Bornn, L.; and Cervone, D. 2019. Decomposing the Immeasurable Sport: A Deep Learning Expected Possession Value Framework for Soccer. In *MIT Sloan Sports Analytics Conference*.

Gneiting, T., and Raftery, A. E. 2007. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association* 102(477):359–378.

Hastie, T. J. 2017. Generalized additive models. In *Statistical models in S*. Routledge. 249–307.

IJtsma, S. 2015. A Close Look at My New Expected Goals Model. 11tegen11.

Lawrence, T. 2018. Introducing xGChain and xGBuildup. StatsBomb IQ Services.

Lucey, P.; Bialkowski, A.; Monfort, M.; Carr, P.; and Matthews, I. 2014. Quality vs. Quantity: Improved Shot Prediction in Soccer Using Strategic Features from Spatiotemporal Data. In *MIT Sloan Sports Analytics Conference*.

Mackay, N. 2017. Predicting Goal Probabilities for Possessions in Football. Master's thesis, Vrije Universiteit Amsterdam.

Nori, H.; Jenkins, S.; Koch, P.; and Caruana, R. 2019. Interpretml: A unified framework for machine learning interpretability.

Pappalardo, L.; Cintia, P.; et al. 2018. Playerank: data-driven performance evaluation and player ranking in soccer via a machine learning approach. *arXiv preprint arXiv:1802.04987*.

Pedregosa, F.; Varoquaux, G.; et al. 2011. scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12(Oct):2825–2830.

Robberechts, P.; Van Haaren, J.; and Davis, J. 2019. Who will win it? an in-game win probability model for football. *arXiv preprint arXiv:1906.05029*.

Singh, K. 2019. Introducing Expected Threat (xT).

Worville, T. 2017. Expected assists in context.

Yam, D. 2019. Attacking Contributions: Markov Models for Football.