Implementation of a probabilistic In-Game Win Probability Model in Football

Morten Stehr[1]

HAW Hamburg, Germany
morten.stehr@haw-hamburg.de

Abstract. The Live-Prediction of a football game based on historical observations brings along different challenges that pre-game predictions don't need to tackle. To cope with these challenges, a Bayesian model is implemented to predict the live game based on these observations. The implemented model can outperform comparable models, but struggles when dealing with incomplete data.

Keywords: Bayesian Model · Probabilistic Programming · Football

1 Introduction

Predicting the result of a football match is an area of research since at least 1968 [11]. The results are not only of interest for the betting industry, but also for professional teams that try to maximize their chances of winning. Most of the available research has been done for pre-game prediction. But even small events in the game can result in an entirely different result than the pre-game prediction. Therefore, a model for predictions during a live game is needed.

The live-prediction of the result leads to new challenges. The most difficult challenge is how to represent the game state accurately. There are a lot of new data features that emerge in a live game that need to be used to represent the flow of the game. The challenge is to decide which features are important and therefore should be used. Generally, more features can represent the game state better, but more features also lead to a more complex model. Another challenge is the existence of injury time in football. As a consequence, the game time can vary from game to game. There can be a huge difference in prediction depending on the remaining game time and if the model assumes the game finished after 90 minutes or the model knows there will be 5 minutes injury time added on. Especially American sports don't have to deal with this problem because there is no injury time and every game finishes when the time is up. Another challenge is the rarity of goals. As a result, a goal has a much higher influence on the game and can lead to a changed game dynamic. Additionally, football has a high occurrence of draws, especially compared to the US sports where a draw is quite rare. Because of this, football can't be a binary win-loss prediction. This challenge is not specific to in game prediction, but for the result prediction in football generally.

Because of these challenges, existing models like a regression model for American football [10], models for baseball [7] and basketball [6] and a random forest model for American football [8] can't be applied or would perform badly. Therefore, this paper describes the implementation of a Bayesian model for in-game win probability. This model is originally described by Robberechts et al. in the paper A Bayesian Approach to In-Game Win Probability in Soccer [12].

In the following chapter, the model will be described, followed by a description of the data and the features in chapter 3. In chapter 4 the actual implementation will be covered in depth. The paper concludes with an evaluation of the model.

2 Model

In the following section the implemented Bayesian model will be described, before in section 4 the implementation details are presented.

The idea of the model is not to predict the outcome of the game directly, but the goals each team will score in the remaining game time. This is modelled as

$$P(y_{>t,team} = g | x_{t,team})$$

The model estimates the probability distribution that the team will score $g \in \mathbb{N}$ goals until the end of the game under the condition that the modelled game state x occurred. The goal probability distribution is calculated for both teams at time t. With the actual result at time t, the win/draw/loss probabilities can be derived. This modeling has multiple advantages. First, this removes the necessity to also predict draws because the win draw loss prediction happens implicitly by predicting the goals. And secondly, the distribution over the goal difference contains a prediction certainty.

The expected number of goals that a team will score are modelled as independent Poisson distributions.

$$y_{>t,team} \sim Pois((T-t) * \theta_{t,team})$$

Poisson's distributions are ideal when the events that should be simulated are countable, which is the given case when simulating goals scored. The λ parameter of the Poisson distribution is calculated from $(T-t) * \theta$, where θ models the scoring intensity at time t and $(T-t)$ is the remaining game time. The scoring intensity is estimated from the game state features x_t. Because the importance of those features varies over time, they must be weighted accordingly. Furthermore, it cannot be assumed that the development of importance is linear because of the game's final sprint. To estimate these importance parameters, a temporal stochastic process is implemented. This allows to share information and to perform coherent predictions between timeframes, which isn't possible using other approaches. Additionally, the stochastic process eliminates the risk of jumping predictions between timeframes. This problem is known in different result prediction models where the probability changes dramatically with only

the time advancing but now major event happening. A popular American football prediction model [1] suffers from this problem. The win probability for a tie game with 5:01 minutes left in the game is 50% whereas in the same situation but with only 5:00 minutes remaining the win probability jumps to 76.69%. This behavior is obviously unwanted and can be eliminated this way.

The scoring intensity is modeled as

$$\theta_{t,away} = invlogit(\alpha_t * x_{t,away} + \beta_t)$$

for the away team, and

$$\theta_{t,home} = invlogit(\alpha_t * x_{t,home} + \beta_t + Ha_t)$$

for the home team. α models the time varying importance of the feature x. Each feature has an own alpha to model their importance. Alpha is modelled as $\alpha_0 \sim N(0, 2)$ and $\alpha_0 \sim N(\alpha_{t-1}, 2)$. A Normal distribution with a zero-mean prior and a variance of two, which corresponds to weak information regarding the true values. The stochastic process is realized by setting the prior of the mean of $\alpha_{t>0}$ to the posterior value of α_{t-1}. The intercept β is also a Normal distribution with a zero-mean prior and a variance of two. Ha is the Home Advantage. It's modelled in the same way that beta is modelled, but only exists in the model for the home team.

In figure 1 an exemplary goal matrix can be seen. This matrix is created by combining the predictions of goals that still will be scored by the home and the away team. For this, samples are taken from the Poisson distribution of the home and the away team. In this case, the most likely goal distribution for the rest of the game will be one goal scored by the home team and none by the away team, with 19%. This is followed by two goals by the home team and none by the away team (16%) and none goal by either team (12%). By combining this matrix with the actual score at the time, the win-draw-loss prediction for the rest of the game is created. Assuming that the game is tied at this point of the prediction the probability of a home win is the sum of the probabilities over the diagonal (66%), a draw is the sum of the probabilities of the diagonal (23%) and the probability of an away win is the sum of the probabilities under the diagonal (9%).

3 Data

In the following section, the used data features are described. These features try to represent the game state as accurately as possible, restricted by the fact that too many features would lead to a model that is too complex. If possible, the used features are the same as in the Bayesian Model of Robberechts et al. [12]. Therefore, a comparability between the results is given.

The features can be categorized into three categories. The first category is pre-game features. These are features that are already available before the game started. This includes the league position, the current form or a chess like ELO

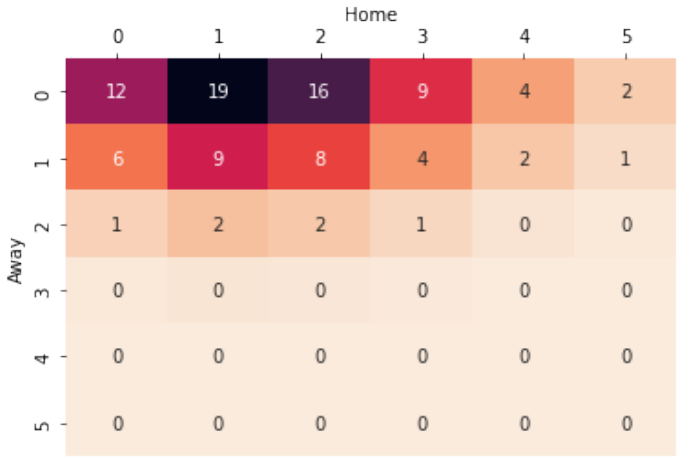


Fig. 1: Goal Distribution Matrix - The labels are the goals each team will score until the end of the game

rating. The second category consists of base features. Base features contain standard information about the game, such as the game time and the score differential of the teams. These features don't try to interpret the game, but present facts. Last are the contextual features. These are the features that try to represent and interpret the ongoing game. This includes a large variety of different aspects of the game such as possession, shots or successful tackles. An in-depth description of the used features can be seen in section 3.1.

The dataset used to acquire the features is the open-data dataset by StatsBomb [2]. The dataset contains event data for approximately 1000 games from different competitions. Ranging from the latest seasons of the English woman Premier League to selected world cups or Champions League campaigns. Unfortunately, the data didn't contain information about the pre-game features, and it wasn't possible to reconstruct those features for such a variety of different competitions. These features are therefore omitted from the model. The used features are covered in the next section.

3.1 Features

In the following, the nine features selected to represent the game state are described.

1. Game Time The game time is the percentage of the game that has already passed. This is equal to the time slot the data corresponds to. The remaining time can be an important feature because no matter how likely a team is to score, less time leads to a lower probability of success.

2. & 3. Score Differential and Team Goals These two features provide information about the scoreline at this time. The score differential describes if

the game is very one-sided, the team goals describe if the team already scored a lot and is therefore likely to score again.
4 & 5 Red and Yellow Card Differential The differential of received cards up to this point in the game. The received red and yellow cards are two separate features. Where red cards lead to a numerical disadvantage, yellow cards have the consequence that players are hampered while tackling.

Up to this point, all features fall into the base feature category. The following features are contextual features that represent the flow of the game.
6. Attacking Passes The average of attacking passes per timeframe. Attacking passes are completed passes that are directed forward, in the direction of the opposition's goal, and where the receiver is located in the final third of the pitch. This feature is a variation of the possession stat that contains more valuable information.
7. Expected Threat (xT) Expected Threat is an extension of the expected Goals (xG) metric. xG gives more context about the quality of chances that a team accumulated over the curse of a game. A xG, depending on its complexity, takes factors like the position of the shot, positioning of the opposing players or if the shot was taken with the weak foot, into account and calculates how often shots with the same parameters lead to a goal. This probability is the xG for the given shot [4]. xT builds onto this idea, but also wants to credit passes that lead to dangerous situations. Every location on the pitch gets a pass/dribble and a shoot probability assigned, calculated from historic data. Additionally, a transition matrix is calculated for every position of the pitch, containing the probability to move from one part of the pitch to another. The xT value for a pitch position is recursively calculated by multiplying the xG value with the probability of a shot and adding the xT values of the transition matrix dependent on the transition probability. This modelling has similarities to a Markov model where each grid location is a state, and passing or dribbling leads to state transitions [5]. The xT of an action is the difference in xT between the starting and the end point of this action. xT rewards the creating of dangerous situations even if there is no shot. The rolling average over the last 12 time steps is used as the feature value.
8. Goal Scoring Opportunities Goal scoring opportunities are defined as shots, blocked shots and situations where a player was situated in a dangerous area, but there was no shot. A dangerous area is defined as a position where the xT value is high. The difference to the xT feature is that the xT tries to value these chances, whereas the goal scoring opportunities feature only counts them. The feature value is the average of goal scoring opportunities per timeframe.
9. Duel Strength Duel Strength consists of won duels, but also of won loose ball situations. It is very similar to the widely used won duel percentage, although in this case, a running average over the last frames is used. Therefore, containing more information about the short term and less about the whole game.

4 Implementation

In the following chapter, the implementation details of the model will be described. At first, the technical environment, as second the implementation of the model on Python code basis is described. This includes the definition and the fitting of the model, as well as predicting results using the model.

Environment In the following, the technical environment of the model is presented, including Software and Hardware aspects.

	Component	Specification
Hardware	CPU	Intel Core i7-8565U, 4.6 GHz
	RAM	16 GB
Software	Python	v.3.9.7
	pymc	v.3.11.2
	kloppy	v.3.5.0
	numpy	v.1.22.1
	pandas	v.1.4.3
	socceraction	v.1.2.3
	soccerdata	v.1.0.5

Table 1: Overview of software and hardware components used to fit and make predictions with the implemented model.

The proposed Bayesian model can't be solved analytically. Therefore, to perform inference, a probabilistic programming framework is necessary. PyMC [13], a probabilistic programming framework in Python, is used. With PyMC it is possible to define probabilistic models and perform inference on them using different methods.

The python packages kloppy, soccerdata and socceraction handle the data acquisition and import.

Model In the following section, the implementation of the model on a code level is presented. For this, Python snippets are used to further clarify the implementation.

```
import pymc3 as pm
...
with pm.Model() as away_model:
    alpha_1 = pm.Normal("alpha_1", alpha_1_t-1, 2)
    ...
    alpha_8 = pm.Normal("alpha_8", alpha_8_t-1, 2)
    beta = pm.Normal("beta", 0, 2)
    data = pm.Data("data", away_feature_data)
```

```
9     theta = pm.invlogit((alpha1 * data[0] + ... + alpha8 * data[7]) +
          beta)
10
11    like = pm.Poisson("like", (T-t) * theta, observed=result_data)
```

The Model is implemented as described in section 2. In this example, the model for the away side is presented. In line 3 a PyMC model is defined. The model acts as a container where distributions or other functionality can be added. The model is used to infer the parameters and perform predictions. From line 4 to line 6 the Normal distributions for alpha and beta are added. The distributions take a name as an identifier, and the prior values for the mean and the variance. The stochastic process is implemented by setting the mean prior of alpha to the posterior of the alpha in the previous time step, except for $t = 0$ where the prior is also 0. In line 9 theta is calculated. This is a deterministic calculation because the calculation only depends on the other distributions but has no individual stochastic property. In this operation, every feature is weighted according to its alpha. Adding Beta to the sum of these weighted features, theta can be calculated using the inverse logarithm. The Poisson distribution can be called just like the Normal distributions, by naming it and initializing the parameters. In this case, λ is $(T-t) * theta$. The new keyword is *observed*. With this keyword, the labels are passed to the model. The labels are in this case the goals that are still to be scored in the remainder of the game.

Because the alphas are one-dimensional, the model can only be trained with data from one time step. The consequence of this is that 100 of these models are necessary to represent a whole game. PyMC provides the possibility to create distributions in groups like

```
1  alphas = pm.Normal("alphas", 0, 2, shape=(8, 100))
```

This creates Normal distributions of the shape $(num\_features \times timesteps)$. With this modeling, only one model would be necessary. The problem is that the implementation of the stochastic process is not possible when using the shape feature. But the stochastic process is essential for the results. The consequences if the stochastic process is not implemented can be seen in chapter 6. Because of this, the shape feature can't be used, and the 100 models need to be trained in a loop.

In figure 2 the process of an exemplary feature can be seen. It's the feature red card differential. As seen in the figure, an early red card has a higher influence on the game than a red card received later in the game. This is indicated by the value of the feature being at almost -2 at the beginning of the game, but getting closer to 0 as the game progresses. This is reasonable, as an early red card has the consequence that the team is a man down for a longer time. A surprising point is that the influence, after being at around 0 from the 50 percent mark of the game to the 80 percent mark, dips again into the negative for the remaining 20% of the game. This means that a red card received in the final 20% of the game is worse than a red card received at half-time. When the opposition team received the

red card, the values are flipped because the differential is now negative. Making an opposition red card in the early stages of the game a big advantage for the team.

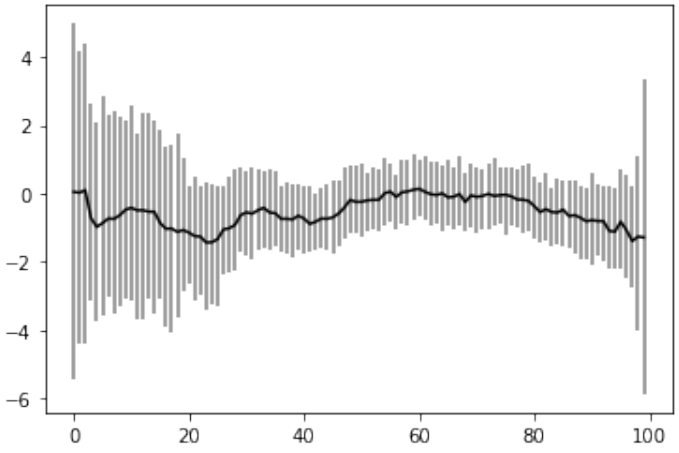


Fig. 2: Process of the importance of the red card differential feature.

Inference As already mentioned, the Bayesian model is too complex to be solved analytically. Instead, the model is inferred on the observed result_data. PyMC offers two different methods to perform the inference, Markov chain Monte Carlo (MCMC) and Automatic Differentiation Variational Inference (ADVI). MCMC allows sampling and inferring on the observed data. However, MCMC algorithms are computationally expensive. In comparison, ADVI algorithms aren't that computational expensive and scale better with large datasets [3]. For this reason, PyMC's ADVI implementation is used to perform inference.

```
1 with away_model:
2     approx = pm.fit(100000)
3     trace = az.from_pymc3(approx.sample(500))
```

The code snippet shows the code to run inference on the previously defined model. The first line of code defines the previously defined model as context, because of this the following lines of code correspond to the model without specifically addressing it. In line 2 the model is fitted using ADVI over 100000 iterations. The *fit* function uses ADVI implicitly. The 100000 iterations is the same number of iterations Robberechts et al. used. In the third line, 500 samples are drawn from the model, acquiring the parameter distributions.

In table 2 the runtime of an inference step is displayed. This is only one model, to train the models for the complete game all 100 models need this

Model	Runtime
home	∼45-50 sec per Model
away	∼40-45 sec per Model

Table 2: Runtime of inference per model.

training time. Noteworthy is that because of the Ha in the home model, the training takes approximately five seconds longer. This is also the reason why the feature number needed to be restricted, as every additional feature would lead to a more computationally intensive inference process.

Prediction The next code snippet shows how to predict results using the fitted model.

```
1 with away_model:
2     pm.set_data({"data": X_test_data}, model=away_model)
3     ppc = pm.sample_posterior_predictive(trace, model=away_model,
          samples=1000)
```

To predict using the model, two steps need to be taken. The first can be seen in line two. With the function set_data the feature data is updated to the data that the model will use for the prediction. In the next line, the 1000 samples are drawn from the model using the provided posteriors in *trace*. *trace* is the result of the training step. Each of the 1000 samples contains a prediction that resembles the underlying distribution.

5 Improvement

Another aspect of the implementation of the model was to identify possible improvements of the model. For this, the equation

$$y_{>t,team} \sim Pois((T-t) * \theta_{t,team})$$

was seen as a possibility to improve the model. More precisely, multiplying the team strength factor θ by the linear process $T-t$. By removing the predetermined linear dependency, the model should be able to determine the factor for the θ at each time step. This is realized by changing $(T-t)$ to a Normal distribution.

$$y_{>t,team} \sim Pois(N(0,2) * \theta_{t,team})$$

The rest of the model remains the same. By implementing the model like this, the complexity is increased as more parameters need to be fitted.

Another option is to completely remove the factor of theta.

$$y_{>t,team} \sim Pois(\theta_{t,team})$$

Model	Runtime
t=Normal	∼50-55 sec per Model (away)
no t	∼35-40 sec per Model (away)

Table 3: Runtime of inference per improved away model.

In this case, there is no extra Normal distribution that corresponds to the importance of theta, but the Normal distributions corresponding to the feature strength need to adapt.

The influence on the runtime can be seen in table 3. The runtime with the additional distribution is considerably higher than the runtime without, and also higher than the runtimes from the original model. The influence these changes have on the results will be discussed in the next chapter.

6 Evaluation

In this section, the model will be evaluated on qualitative and quantitative aspects. The qualitative aspect is the test of the eye. It will be evaluated if the prediction is reasonable and if the other requirements of the model are satisfied. Afterwards, these results are quantified by using the expected calibration error.

6.1 Qualitative

In figure 3 is an example of a prediction visualized. The x-axis represents the game time, while the y-axis shows the result prediction. The green area indicates the probability for the home team winning, gray indicates a draw, and red represents the away team's predicted probability of winning the match. The vertical dotted lines are goal events by either team. The match shown is the 2019 Woman World Cup match Cameroon vs. the Netherlands. At this point, the Netherlands were reigning European champions and the clear favorites to win the game. Despite that, the model estimates the win probability for Cameroon at almost 50% at the start of the match. This is due to the missing pre-game features. Without any in game information at the start of the game, the model guesses a higher home win percentage, as statistically home teams win more often. With the Netherlands controlling the game, this initial prediction quickly turns upside down, up to the point where the Netherlands got an 50+% win probability at around the 20-minute mark. After that, the dominance of the Netherlands decreases, with a draw being the most likely result. From the live ticker can be concluded that in the 21-minute Cameroon got a huge chance to take the lead. As a result, the prediction got more balanced. In the 41st minute, the Netherlands take the lead. Because the actual result plays a big role in predicting the game outcome, the goal results in a surge of win probability for the Netherlands from around 25% to almost 75%. Only two minutes later, Cameroon can equalize, restoring not only the game state to a draw, but also the predicted outcome. After the equalizer, it can be observed that the Netherlands steadily

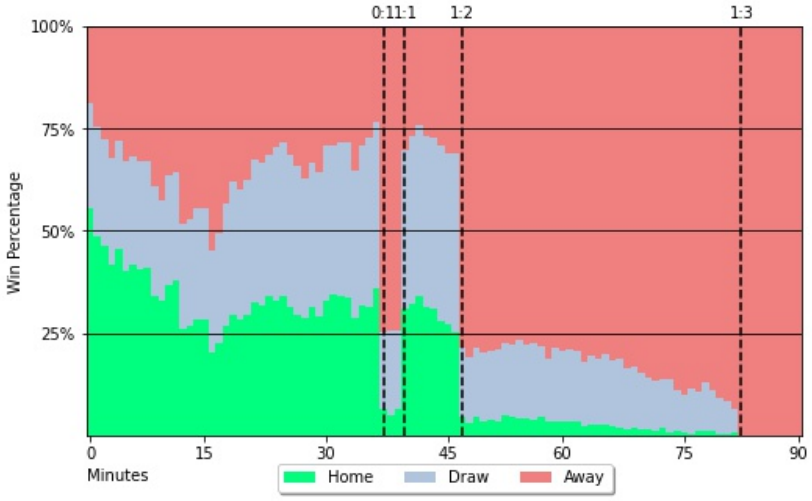
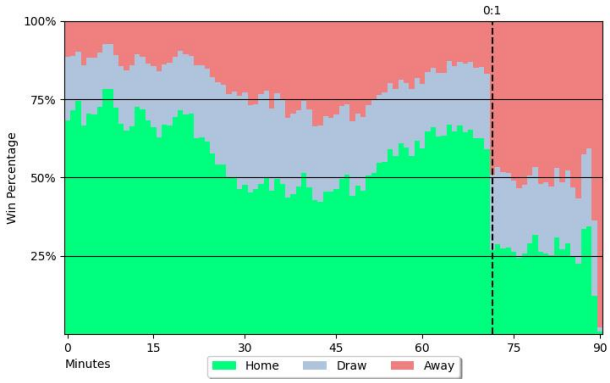


Fig. 3: Live win prediction of the world cup game Cameroon vs. the Netherlands. The Netherlands won the game 1:3.

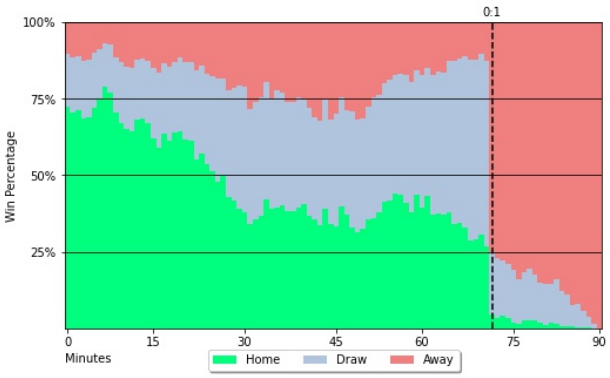
improve their win probability until they score again in the 48th minute. This again leads to a big surge in their win chances, up to 75+%. From this point on, the Netherlands dominate the game and with their odds of winning the game steadily increasing. This is not because they were getting better as the game went on, but because the time Cameroon had left was getting less. The Netherlands would score again in the 85 minute, deciding the game and bringing their win percentage up to 99+%.

This steadily increase in win probability of a leading team is an important property, and one that is only given because of the implementation using a stochastic process discussed in section 4. In figure 4 a bad and a good example can be seen. Figure (a) shows the predictions of the model without the stochastic process in place. Figure (b) is the prediction of the model that is trained with the stochastic process. Both predictions start the same way and the course is very similar until the 30 minute. After that, the odds for the home team in figure (a) never really fall under 50% and rising to around 70% just before the goal. Without knowing the actual run of the game, the probabilities in figure (b) seam more feasible. The draw gets more realistic the later the game gets without a goal being scored. The dramatic difference occurs after the goal. In figure (b) the win probability for the away team jumps to 75% and steadily rises until the end of the game. In figure (a) however, the win probability only rises to 50% and after that stays the same until the last timeframe where it jumps to 100%. According to this prediction, the home team had a 25% chance to win, although being 1-0 down with only one minute to play. Although it is possible that the home team can still win this match and it certainly happened before.

This prediction suggests it happens every four games, which is definitely not the case.



(a) Prediction without the stochastic process.



(b) Prediction with the stochastic process.

Fig. 4: Comparison of predictions of the same game. With and without the implementation of a stochastic process.

6.2 Quantitative

ECE[9] is short for expected calibration error. The calibration error is the difference between the fraction of correct predictions, the accuracy and the mean of the probabilities, the confidence. In the following table 4, the ECE for the in the paper described model and some reference models is shown.

The ECE score is calculated for the first and second half, the last 10% of the game and the whole game separately. The comparison models all originate from

	First Half	Second Half	Final 10%	Whole Game
LR	0.031	0.069	0.174	0.023
mLR	0.023	0.067	0.170	0.027
RF	0.051	0.013	0.101	0.024
Bayesian Model[12]	0.012	0.013	0.002	0.011
Paper Model	**0.049**	**0.020**	**0.009**	**0.031**
Paper Model with Improvement(t=Normal)	0.103	0.030	0.010	0.069
Paper Model with Improvement(no t)	0.119	0.031	0.010	0.073

Table 4: A table without vertical lines.

the paper from Robberechts et al. [12] and are therefore trained on the same data. These models are a linear regression model, a multiple linear regression model, a random forest model and the Bayesian model this paper's model is based on.

Comparing this paper's model with the other models on the first half, this model's predictions calculated ECE is on a level with random forest, but almost twice as bad as the linear regression models and over four times as bad as the Bayesian model. The biggest and probably most influential reason for this result is the state of the available data. As already described, it wasn't possible to acquire pre-game data that contained information over the quality of the teams. The other models had access to an ELO like rating. Especially at the beginning of the game, this will have a big impact on the initial prediction. Without this information, the model starts with uninformed predictions that, at the beginning of the game, can be easily skewed by single events.

This assumption is confirmed by looking at the ECE for the second half. The error is still almost twice as high as the Bayesian model and this time also the random forest model, but the error overall has gone down by almost 60%. At this stage, the error of the models predictions is only a third of the regression models error.

The same trend can be observed in the final stages of the game. The model performs significantly better than all the other non-Bayesian models, confirming that Bayesian models can handle the final sprint of the game better than other models. Although the model still performs worse than the Bayesian model, the ECE error of both is small.

The results of the possible improvements of the model are disappointing. At all stages of the game, these models perform considerably worse than the unchanged paper model. Although still following the same pattern, that the results in the last 10% are better than the results calculated for the complete second half and the significantly better than the results for the first half. It follows that the Normal distributions alone cannot map the timely progress of the game. The simple multiplication on the other hand with the remaining game time can.

7 Conclusion

It was possible to implement a Bayesian model following the model of Robberechts et al. This model was able to predict the outcome of games based on a live game state. These predictions passed the test of the eye and the model outperformed different comparison models. Further review needs the bad results in early stages of the game. It needs to be evaluated if the data situation really is the biggest concern, or if the model has further problems. It was also not successful to improve the model, as the results got worse with both approaches.

References

1. Pro Football Reference - win prediction model. https://www.pro-football-reference.com/boxscores/win_prob.cgi, accessed: 2022-09-11
2. StatsBomb - open-data. https://github.com/statsbomb/open-data, accessed: 2022-09-14
3. Blei, D.M., Kucukelbir, A., McAuliffe, J.D.: Variational inference: A review for statisticians. Journal of the American Statistical Association **112**(518), 859–877 (apr 2017). https://doi.org/10.1080/01621459.2017.1285773, https://doi.org/10.1080%2F01621459.2017.1285773
4. Eggels, H.H.: Expected goals in soccer:explaining match results using predictive analytics (2016)
5. Gagniuc, P.: Markov Chains: From Theory to Implementation and Experimentation (05 2017). https://doi.org/10.1002/9781119387596
6. Ganguly, S., Frank, N.: The problem with win probability
7. Lindsey, G.R.: The progress of the score during a baseball game. Journal of the American Statistical Association **56**(295), 703–728 (1961). https://doi.org/10.1080/01621459.1961.10480656, https://www.tandfonline.com/doi/abs/10.1080/01621459.1961.10480656
8. Lock, D., Nettleton, D.: Using random forests to estimate win probability before each play of an nfl game. Journal of Quantitative Analysis in Sports **10**(2), 197–205 (2014)
9. Nixon, J., Dusenberry, M.W., Zhang, L., Jerfel, G., Tran, D.: Measuring calibration in deep learning. In: CVPR Workshops. vol. 2 (2019)
10. Pelechrinis, K.: iwinrnfl: A simple, interpretable & well-calibrated in-game win probability model for nfl. arXiv preprint arXiv:1704.00197 (2017)
11. Reep, C., Benjamin, B.: Skill and chance in association football. Journal of the Royal Statistical Society. Series A (General) **131**(4), 581–585 (1968), http://www.jstor.org/stable/2343726
12. Robberechts, P., Van Haaren, J., Davis, J.: A bayesian approach to in-game win probability in soccer. pp. 3512–3521 (08 2021). https://doi.org/10.1145/3447548.3467194
13. Salvatier, J., Wiecki, T.V., Fonnesbeck, C.: Probabilistic programming in python using PyMC3. PeerJ Computer Science **2**, e55 (apr 2016). https://doi.org/10.7717/peerj-cs.55, https://doi.org/10.7717/peerj-cs.55