In-game Prediction of Win Probability in NBA Games

Tom Bukić

accelerate

September 2025

Abstract

We estimate in-game home win probability in the NBA from game state and team form using four model classes (ridge, logistic, gradient-boosted trees, Bayesian neural net). Gradient boosting performs best on a held-out evaluation set (log loss 0.4739), followed closely by the neural network (0.4757). Linear and logistic regression exceed 0.5 log loss. Models are stable to random initialization and well calibrated. Limitations and possible extensions are discussed.

Keywords: basketball analytics; in-game prediction; calibration; CatBoost; Bayesian neural networks

1 Introduction

Basketball is modeled today both to analyze the contribution of individual players to team performance [2] and to predict outcomes [1, 3]. When predicting outcomes, one can predict before tipoff or during the game, conditional on game evolution. This paper estimates the home team's win probability as the game progresses, as a function of remaining time, score, and prior performance of the teams involved.

Four classes of machine learning algorithms are compared: linear and logistic regression, gradient boosted decision trees, and a Bayesian neural network. The best performer is the decision tree model with log loss 0.4739, closely followed by the neural network with log loss 0.4757. Both regressions have log loss greater than 0.5.

2 Background

2.1 Statistical modeling of basketball

In a recent literature review, Maddox traces the beginnings of in-game outcome prediction to plotting point differentials over time. Since then, match outcomes have been modeled with increasingly sophisticated methods, including game simulations via stochastic processes such as

Brownian motion, Markov chains, and Poisson processes. Maddox used a Bayesian estimator for in-game outcomes evaluated with the Brier score. The literature notes that bookmaker odds and betting markets are strong reference values that aggregate a wide spectrum of signals which are very hard to beat with purely statistical methods.

2.2 Project foundation

This work directly continues the NBA Probability project[1], a public GitHub project whose goal was to predict a team's win probability at a given in-game context defined by the score and time remaining. That work predicted the outcome from the current score, remaining time, and the teams involved. The model was trained on a Kaggle play-by-play dataset with an 80-20 train test split. It is unknown whether the modeling focused on recent data or used the entire dataset spanning more than 20 years of NBA games. Two logistic regression models were trained using: `AwayScore`, `HomeScore`, `TimeRemaining` in seconds, and `AwayName_encoded` and `HomeName_encoded` which are integer encodings from 0 to 29. One model output the home team's win probability and the other the away team's.

The prior work had major methodological issues: teams were represented in a linear model as integers from 0 to 29 via `LabelEncoder` from scikit-learn, which improperly turned a categorical variable into a ratio-scaled one. A second serious issue was data leakage from test into train: when splitting the data, events from a match appearing in the test set were not prevented from appearing in the training set, and there was no guarantee that matches in the training set chronologically preceded the matches in the test set. These problems invalidate subsequent analyses of the original model. Finally, the two separate models for home win and away win produced probabilities whose sum was neither constrained to 1 nor equal to 1, so they did not model a proper probability.

3 Data

The team list and historical results and play-by-play data were pulled from the official NBA site using the Python `nba_api` package that automates the process. Before feature construction, the tables were merged into a single table that, for each score change, contained the home and away team names, season and game date, final outcome, the time of the score change, and the number of points each team had scored at that moment.

Given how teams evolve throughout a season, we restricted to the two most recent seasons, 2023-24 and 2024-25. The final modeling dataset contains 2 029 games between the 30 NBA teams from October 2023 through January 2025, with a total of 30 435 score change events.

3.1 Feature engineering

Internet sourced data were transformed into features more suitable and informative for training. Instead of tracking each team's points separately, we used the home minus away point differential, and the original time variables (quarter, minute, second) were converted to seconds remaining.

[1] https://github.com/thfalan/nba-probability

For games with overtime, regulation time was set to run down to 0 seconds, then for each overtime the remaining time variable was set so that each overtime period also ended at 0 remaining seconds.

At the season level and over the last 5 games for every team we computed form features: points scored, points allowed, and number of wins. For home and away teams these variables were available both cumulatively and as averages. Early in a season, last five game averages were divided by the appropriate smaller number of games. For the first game of a season, a team's values were initialized by copying end of previous season values.

3.2 Normalization and standardization

Variables were finally normalized and standardized. Time was linearly transformed to the interval $[-1, 1]$, while other integer variables were standardized around their mean. In the linear and logistic regression models, the team categorical label was converted to a 0-1 vector via one hot encoding. Bayesian neural network had team encoder which was learning a dense representation for every team.

4 Methods

We trained models from four classes: linear regression, logistic regression, gradient boosted decision trees, and a Bayesian neural network. Metrics used to evaluate performance were log loss, Brier score, accuracy, and expected calibration error (ECE). Hyperparameters were optimized with the Python library `optuna`. All evaluated hyperparameters are listed in Table 1.

4.1 Algorithms

4.1.1 Linear and logistic regression

We used ridge regression, linear regression with quadratic penalty α which controls weight regularization. Linear regression was computed with `Ridge` in scikit-learn [4]. Besides the weights, linear regression yields a unique estimate of predictive variance. Using this and the predicted point differential, and assuming normal residuals for the differential, we convert to the home win probability.

Logistic regression was computed with `LogisticRegression` in scikit-learn [4]. The output models the home win probability.

4.1.2 Decision tree

We used CatBoost with the `RMSEWithUncertainty` loss [5, 6]. This loss enables estimation of predictive uncertainty which includes both aleatoric and epistemic components. In practice we use it as an estimate of predictive variance and, under the normal residual assumption for the differential, convert it to the home win probability, analogously to linear regression. Key hyperparameters include `learning_rate`, `max_depth`, `iterations`, `l2_leaf_reg`, `colsample_bylevel`,

`subsample`, `max\_bin`. A further advantage of CatBoost is out of the box support for categorical variables.

4.1.3 Neural network

The neural network consisted of two symmetric subnets, team encoders, whose outputs are concatenated with the remaining time variable and then fed to the final `res_layers` layers of width `res_hidden_dim`. A team encoder learns vector representations of teams, one vector per team of dimension `embedding_dim`. If `embedding_dim=None`, this part is skipped. The team representation is concatenated with form variables and passed through `team_layers` layers of width `team_hidden_dim`. The final output has two components: μ and s. The first is the expected point differential. The second is transformed via $\text{softplus}(x) = \ln(1 + e^x)$ to an estimated variance. The loss is

$$L(y, \hat{\mu}, \hat{\sigma}^2) = \tfrac{1}{2}\ln(2\pi) + \ln\hat{\sigma}^2 + \frac{(y-\hat{\mu})^2}{\hat{\sigma}^2}, \quad \ln\hat{\sigma}^2 = \ln(1 + e^s + \epsilon),$$

where ϵ is a small constant for numerical stability.

Table 1: Searched hyperparameters by algorithm

Algorithm	**Parameter**	**Range or set**	**Scale**
Linear regression	α	$[10^{-4}, 100]$	logarithmic
Logistic regression	`C`	$[10^{-3}, 100]$	logarithmic
Decision trees	`learning_rate`	$[10^{-3}, 10^{-1}]$	logarithmic
	`max_depth`	[4, 9]	integer
	`iterations`	[500, 1500]	integer
	`l2_leaf_reg`	$[5 \cdot 10^{-2}, 10]$	logarithmic
	`colsample_bylevel`	[0.5, 1]	continuous
	`subsample`	[0.5, 1.0]	continuous
	`max_bin`	[32, 128]	integer
Neural network	`learning_rate`	$[5 \cdot 10^{-4}, 5 \cdot 10^{-2}]$	logarithmic
	`weight_decay`	$[10^{-6}, 10^{-1}]$	logarithmic
	`lr_gamma`	[0.8, 1.0]	continuous
	`team_hidden_dim`	[4, 32]	integer
	`team_layers`	[1, 3]	integer
	`res_hidden_dim`	[4, 64]	integer
	`res_layers`	[2, 6]	integer
	`embedding_dim`	{`None`, 1, 2, 4, 8, 16}	categorical

4.2 Procedure

Data were split chronologically into training set (60 percent of games), test set (20 percent), and evaluation set (20 percent). This ensured an unbiased assessment, since the algorithm evaluated statistically was neither trained nor filtered on the evaluation data.

First we used linear regression and decision trees on the training set to compare feature importance and select variables for later stages. Then we performed hyperparameter optimization by training on the training set and comparing on the test set. For hyperparameter search, each algorithm was trained 50 times with different hyperparameters. We then chose the best hyperparameters per model and trained each with 30 different seed values to find the best seed and analyze sensitivity to randomness during optimization.

Finally, we evaluated one model from each class by training on the combined training and test sets and evaluating on the previously unseen evaluation set.

5 Results

5.1 Feature selection

Both models agree on the importance of the point differential and seconds remaining, while other variables are ranked differently. Figure 1 shows a per model feature importance comparison, and Figure 2 provides a combined view with partial dependence.

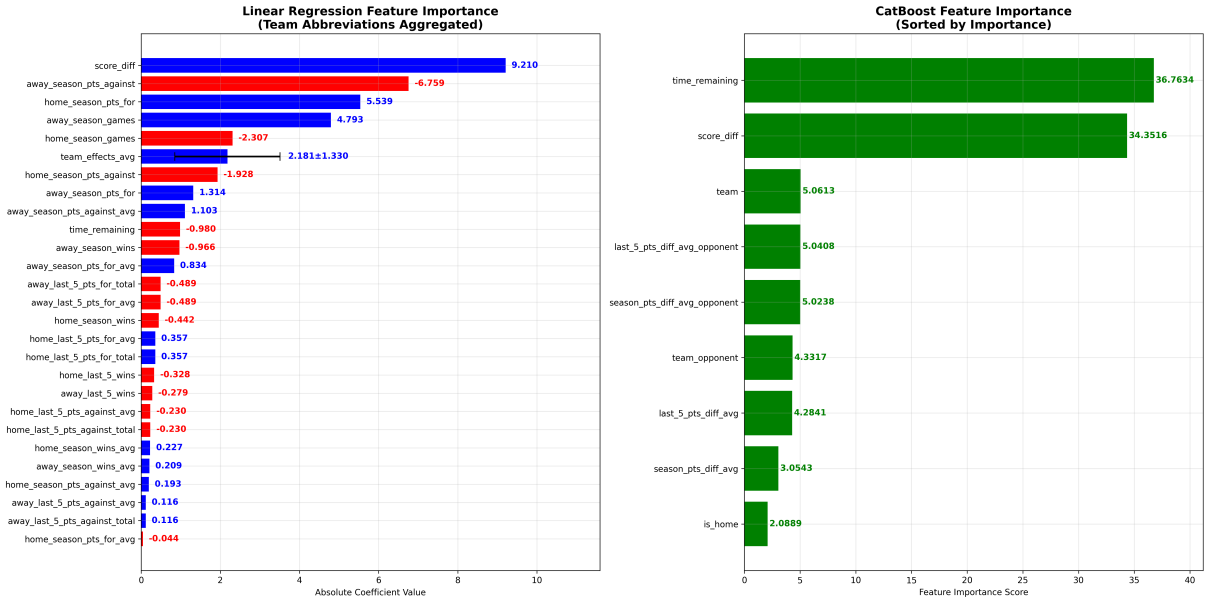

Figure 1: Model comparison: basic metrics.
Source: GitHub repository (image 1)

Both models identified the usefulness of all team form feature groups: offensive (points scored), defensive (points allowed), and overall team quality (wins). They also detected the importance of including the team identity itself.

Given the predictive value of all variable types, along with time remaining, point differential,

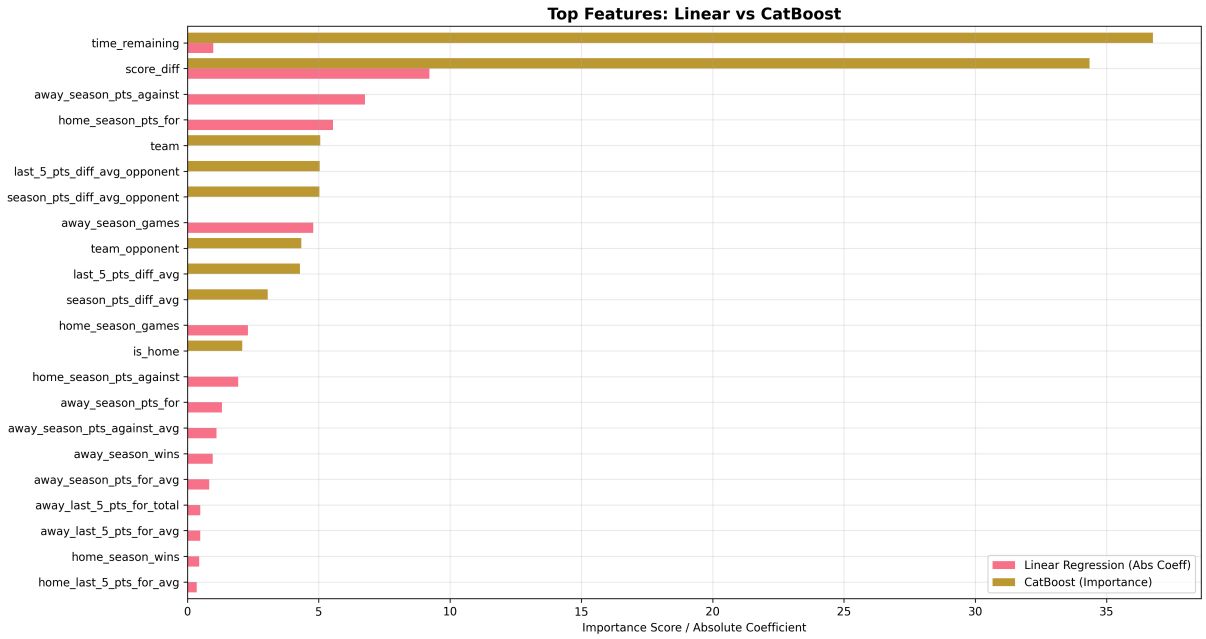

Figure 2: Feature importance and partial dependencies.
Source: GitHub repository (image 2)

and team identity, we selected both season form and last five games form variables for final modeling: total wins and average points allowed and scored. In total, 16 of the 24 candidate variables were chosen.

5.2 Hyperparameter selection

Optimal hyperparameters. For linear regression the optimal hyperparameter is

$$\texttt{alpha} = 99.94523496627072,$$

and for logistic regression

$$\texttt{C} = 0.0036383924346184548.$$

For gradient boosted trees we obtained

$$\begin{aligned}
\texttt{learning_rate} &= 0.09636223593149941,\\
\texttt{max_depth} &= 5,\\
\texttt{iterations} &= 1400,\\
\texttt{l2_leaf_reg} &= 0.4290344235973468,\\
\texttt{colsample_bylevel} &= 0.9433600798423123,\\
\texttt{subsample} &= 0.5369048376010586,\\
\texttt{max_bin} &= 91.
\end{aligned}$$

For the neural network the best are

$$\begin{aligned}
\texttt{learning_rate} &= 0.002178737122149179,\\
\texttt{weight_decay} &= 1.2171989575527562 \times 10^{-6},\\
\texttt{lr_gamma} &= 0.8354521167934316,\\
\texttt{team_hidden_dim} &= 4,\\
\texttt{team_layers} &= 3,\\
\texttt{res_hidden_dim} &= 22,\\
\texttt{res_layers} &= 2,\\
\texttt{embedding_dim} &= \texttt{None}.
\end{aligned}$$

Note: `embedding_dim=None` means team embeddings were not used in the final model.

5.3 Seed analysis

In the seed analysis, linear and logistic regression had no variation because the solvers are deterministic. The neural network and decision tree achieved very similar average and minimum log loss, as well as similar variability. We conclude that all analyzed algorithms are stable with respect to random initialization. All results are shown in Table 2.

Table 2: Seed analysis: log loss and best seed by algorithm

Algorithm	**Log loss**		**Seed (best)**
	mean ± std	**minimum**	
Decision tree	0.4145 ± 0.0057	0.4047	47
Neural network	0.4116 ± 0.0058	0.4017	5
Linear regression	0.4388 ± 0.0000	0.4388	2
Logistic regression	0.4489 ± 0.0000	0.4489	2

5.4 Final results

After final training on the train and test sets, the best evaluation set result is achieved by the decision tree with log loss 0.4739, closely followed by the neural network with 0.4756. The latter has slightly higher accuracy but almost twice the expected calibration error. A summary is in Table 3 and an illustrative calibration curve in Figure 3.

Figure 4 shows predictions of all algorithms for a tied score, Figure 5 for an away lead of 5 points, and Figure 6 for a home lead of 15 points.

All source code is publicly available on GitHub: [2].

[2] https://github.com/tbukic/NBA-Ingame-Win-Probability/

Table 3: Final results on the evaluation set

Algorithm	**log loss**	**Brier**	**ECE**	**Accuracy**
Decision tree	0.473 918 83	0.160 686 87	0.019 045 58	0.745 833 33
Neural network	0.475 668 85	0.160 846 61	0.039 562 05	0.750 961 53
Linear regression	0.502 442 79	0.170 622 48	0.038 592 56	0.738 782 05
Logistic regression	0.502 940 34	0.170 278 35	0.039 691 10	0.739 102 56

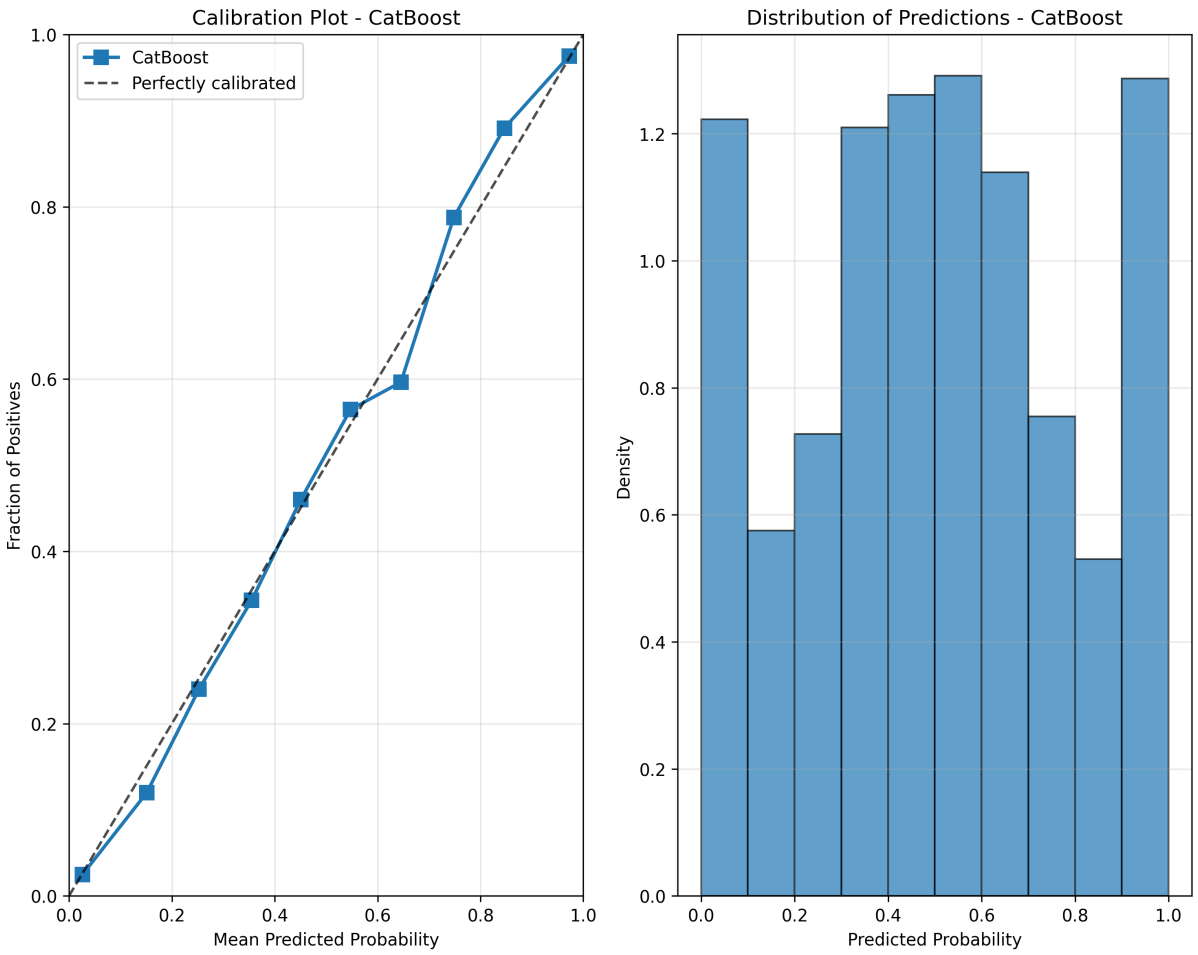

Figure 3: Calibration curve of the best algorithm and distribution of predicted probabilities.

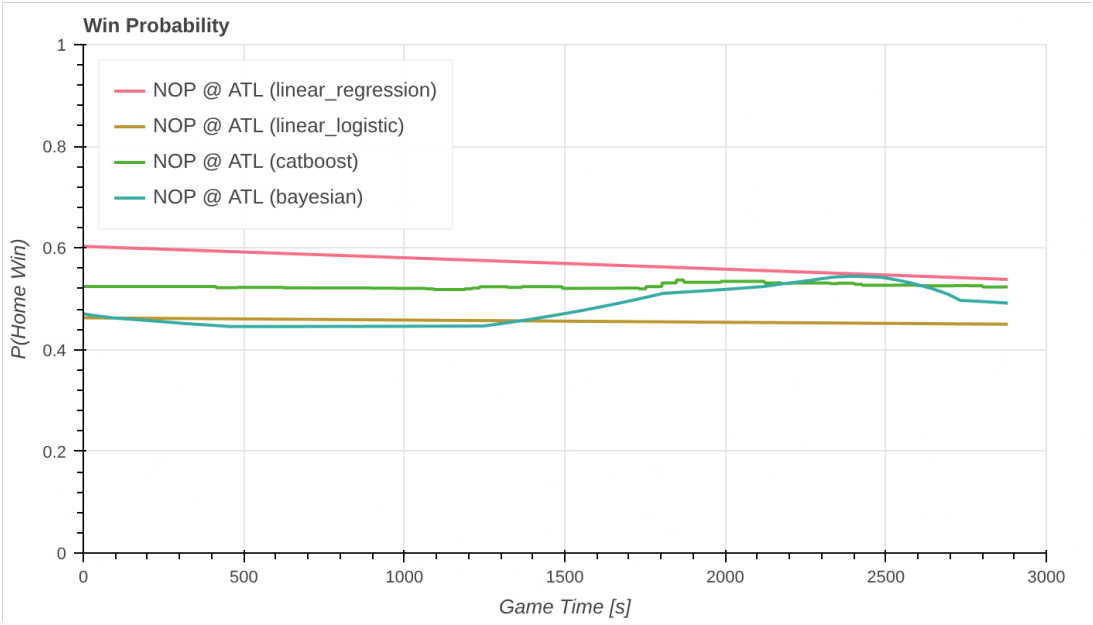

Figure 4: Predictions of all algorithms for a tied score.

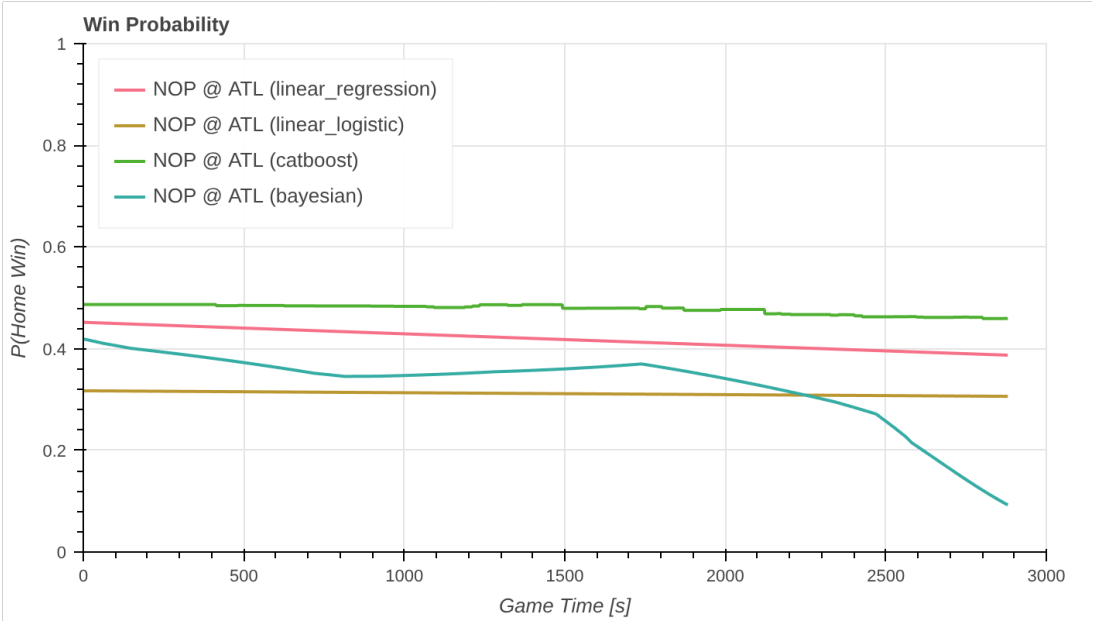

Figure 5: Predictions of all algorithms for an away lead of 5 points.

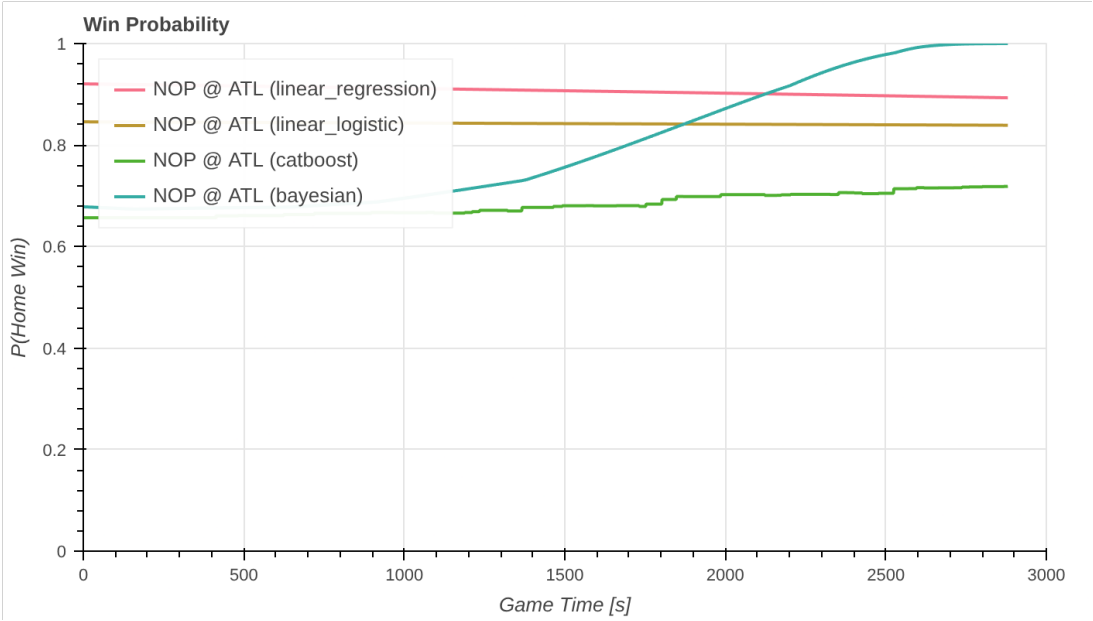

Figure 6: Predictions of all algorithms for a home lead of 15 points.

6 Conclusion

We show that in-game win probability in basketball can be reliably estimated using a combination of game-state variables (time remaining, point differential) and team-form summaries. The best algorithm is gradient-boosted trees (log loss 0.4739), with a Bayesian neural network very close behind (0.4757), while linear and logistic regression lag behind (greater than 0.5). Models are stable with respect to random initialization, and the best model's calibration is very good with a low ECE. Relative to prior work, methodological problems were removed, such as treating teams as ordinal variables and splitting train and test without preventing leakage. Key limitations are the normality assumption used when converting predicted point-differential distributions into win probabilities and dependence among within-game events (non-i.i.d.), which can make naive standard errors/intervals optimistic; robust reporting should use game-level macro-averages and game-clustered uncertainty. Further gains may come from incorporating more detail such as head-to-head history, lineups, possession context (e.g., possession indicator, bonus/penalty status, timeouts remaining, and whether free throws are pending), and injuries.

References

[1] J. T. Maddox, R. Sides, and J. L. Harvill, *Bayesian estimation of in-game home team win probability for National Basketball Association games*, arXiv:2207.05114, 2022. doi: 10.48550/arXiv.2207.05114.

[2] H. Gong and S. Chen, *Estimating positional plus-minus in the NBA*, Journal of Quantitative Analysis in Sports, 20(3), 193–217, 2024. doi: 10.1515/jqas-2022-0120.

[3] P. Vračar, E. Štrumbelj, and I. Kononenko, *Modeling basketball play-by-play data*, Expert Systems with Applications, 44, 58–66, 2016. doi: 10.1016/j.eswa.2015.09.004.

[4] F. Pedregosa et al., *Scikit-learn: Machine Learning in Python*, JMLR 12, 2825–2830, 2011.

[5] A. Malinin, L. Prokhorenkova, and A. Ustimenko, *Uncertainty in Gradient Boosting via Ensembles*, arXiv:2006.10562, 2021. doi: 10.48550/arXiv.2006.10562.

[6] T. Duan et al., *NGBoost: Natural Gradient Boosting for Probabilistic Prediction*, arXiv:1910.03225, 2019. doi: 10.48550/arXiv.1910.03225.