Astronomy & Astrophysics manuscript no. aa
December 7, 2025

Predicting NBA Team Performance: From Historical Standings to Player-level Forecasts

Weihao Li[1,⋆], Shuhao Gao[1], Shijie Chen[1], and Anbang Liu[1]

McCormick School of Engineering, Northwestern University
e-mail: {WeihaoLi2027, ShuhaoGao2027, ShijieChen2027, AnbangLiu2027}@u.northwestern.edu

Received December 8, 2025

ABSTRACT

In recent years, publicly available NBA play-by-play and box-score data have made it possible to build increasingly detailed models of team performance. At the same time, many forecasting systems still rely on simple team-level summaries such as past wins or point differential. We aim to compare a simple standings-based baseline with a player-centric forecasting pipeline for predicting future regular-season wins at the team level. We construct two models. The first directly regresses future team wins on historical win totals over the past two decades. The second learns generic player evolution patterns from a long-horizon panel of both retired and active players, uses these patterns to predict next-season box-score statistics for active players, and aggregates player-level projections into team-level features. Across held-out seasons, the player-based pipeline improves mean absolute error relative to the standings baseline, especially for teams undergoing large off-season roster changes. For more stable teams, historical standings are surprisingly competitive and sometimes match the player-based forecasts. Our results suggest that simple team-level baselines remain strong when rosters are stable, but player-level modeling adds value in high-turnover regimes and provides a more interpretable connection between roster moves and expected team performance.

Key words. basketball – sports analytics – machine learning – time series forecasting – player development

1. Introduction

In this project, we study how well different approaches can predict NBA team performance. Our goal is straightforward: given past data, can we forecast how many games each team will win in a future season? To explore this question, we compare two models that rely on fundamentally different sources of information.

The first model is a simple team-level baseline that uses only historical standings. For any pair of seasons within the past twenty years, we examine how a team's previous win total relates to its win total in another season. This provides a direct, standings-based method for predicting future performance without considering player-level factors.

The second model takes a player-centered perspective. We collect long-term statistics for both retired and still-active NBA players and train a model to learn how player performance tends to evolve over time. Using these learned patterns, we predict next-season box-score statistics for current active players and then aggregate the projected player output to estimate each team's future win total.

By comparing these two approaches, we aim to understand when a simple standings-based baseline is sufficient and when a more detailed, player-based forecasting pipeline provides an advantage, particularly in seasons affected by significant roster changes.

⋆ Corresponding author: wli@u.northwestern.edu

2. Prior Literature

2.1. Team-level rating and expectation models

A large body of work in sports analytics has focused on predicting team performance using aggregate team-level statistics. One influential family of models is the *Pythagorean expectation*, which estimates a team's theoretical winning percentage from points scored and points allowed via a power-law relationship. Originally proposed for baseball and later adapted to basketball, this idea underlies many simple baselines that relate scoring margins to wins and losses (see, e.g., Oliver 2004; Sarlis & Tjortjis 2020). These models provide strong, easy-to-interpret benchmarks but depend solely on aggregate scoring margins and do not incorporate information about individual players.

Another common approach is to model team strength with rating systems inspired by Elo. In basketball applications, Elo-style ratings are updated after each game based on the result, home-court advantage, and the margin of victory, and then used to forecast future game and series outcomes. Public-facing systems such as FiveThirtyEight's NBA model illustrate how dynamic ratings can track changes in team strength over a season and provide reasonably accurate probabilistic predictions for both games and playoff series (FiveThirtyEight 2015). Together, Pythagorean and Elo-style systems represent static or quasi-static team-level baselines that are closely related to our first model, which relies on historical team wins and standings. However, because these methods operate on aggregate team outcomes, they are inherently limited in their ability to anticipate abrupt changes in performance driven by roster turnover or player development.

2.2. Machine learning for NBA game and season prediction

Beyond analytic formulas and rating systems, many studies have applied machine learning to predict basketball results using team-level features. Early work by Loeffelholz et al. (2009) used neural networks to predict single-game NBA outcomes from box-score statistics and contextual variables, demonstrating that nonlinear models can capture interactions between basic team statistics. More recent surveys review a wide range of approaches, including logistic regression, support vector machines, tree-based ensembles, and deep neural networks, and typically find that machine-learning models outperform simpler statistical baselines when sufficient historical data are available (Sarlis & Tjortjis 2020).

Researchers have also moved from game-level prediction to season-level tasks, such as forecasting a team's final win total or playoff qualification. For example, Yang (2015) regress regular-season wins on team-level and aggregated player statistics to study which factors are most predictive of team success. These season-level models again treat each team as the unit of analysis and usually rely on summary statistics from the current or previous season.

2.3. Player-level prediction and its link to team performance

Complementary to team-level approaches, a growing literature studies player evaluation and performance prediction using detailed box-score and tracking data. Sarlis & Tjortjis (2020) review many of these methods, including regression-based models for player efficiency metrics, clustering techniques for grouping players with similar playing styles, and rating systems that quantify individual contribution to team success. In practice, coaches and analysts often combine such player-level models with domain knowledge to support decisions about rotations, matchups, and roster construction.

Some work, including Yang (2015), aggregates player statistics into team-level features to predict season outcomes, effectively creating a simple player-to-team pipeline. However, existing player-focused models typically train and evaluate on overlapping time periods for the same set of players, and they rarely combine long-run player evolution with explicit team-level baselines.

In contrast, our project is designed to learn generic player evolution patterns from a historical cohort that includes both retired and still-active players, and then apply these patterns to predict the next-season performance of currently active players. We subsequently aggregate predicted player statistics to the team level and compare the resulting win forecasts to those from a simple standings-based baseline over roughly two decades of NBA data. This setup allows us to evaluate how a player-based forecasting pipeline and a standings-based model perform side by side, especially in seasons where teams undergo substantial roster changes.

3. Tasks and evaluation

We study two related prediction tasks: (i) forecasting regular-season team rankings from team-level summary statistics, and (ii) predicting individual player statistics. Because the main results of this paper concern team-level accuracy, we focus on the first task here; the player-level task follows the same general protocol but uses different feature inputs.

3.1. Team-level ranking prediction

For each team t and season s, we assemble a feature vector $\mathbf{x}_{t,s}$ from season-level box-score summaries and simple trend indicators. The target is the realized winning percentage

$$\mathrm{WIN\_PCT}_{t,s} = \frac{\mathrm{WINS}_{t,s}}{\mathrm{WINS}_{t,s} + \mathrm{LOSSES}_{t,s}}.$$

A regression model f_θ produces a prediction $\hat{y}_{t,s} = f_\theta(\mathbf{x}_{t,s})$, and teams are ranked within each season by sorting $\hat{y}_{t,s}$ in descending order.

Data split. We use team–season summaries from the 2004–2005 through 2024–2025 seasons (630 observations in total). The two most recent seasons (2023–2024 and 2024–2025) serve as a held-out test set, and all earlier seasons form the training set. All feature engineering is performed on the full panel, but model fitting uses only the training seasons; performance is reported exclusively on the held-out seasons.

Evaluation metrics. Because the task is ranking rather than regression, we evaluate predictions using three rank-based metrics computed separately for each season:

$$\mathrm{ExactAcc}_s = \frac{1}{N_s}\sum_t \mathbb{1}[\mathrm{RANK}^{\mathrm{pred}}_{t,s} = \mathrm{RANK}^{\mathrm{true}}_{t,s}],$$

$$\mathrm{Within1}_s = \frac{1}{N_s}\sum_t \mathbb{1}[|\mathrm{RANK}^{\mathrm{pred}}_{t,s} - \mathrm{RANK}^{\mathrm{true}}_{t,s}| \le 1],$$

$$\mathrm{Within2}_s = \frac{1}{N_s}\sum_t \mathbb{1}[|\mathrm{RANK}^{\mathrm{pred}}_{t,s} - \mathrm{RANK}^{\mathrm{true}}_{t,s}| \le 2].$$

We report each metric for both held-out seasons and also average them for model comparison. Additional details on high-performing model configurations are provided in Appendix A. Figure 1 visualizes predicted versus actual rankings for the two held-out test seasons.

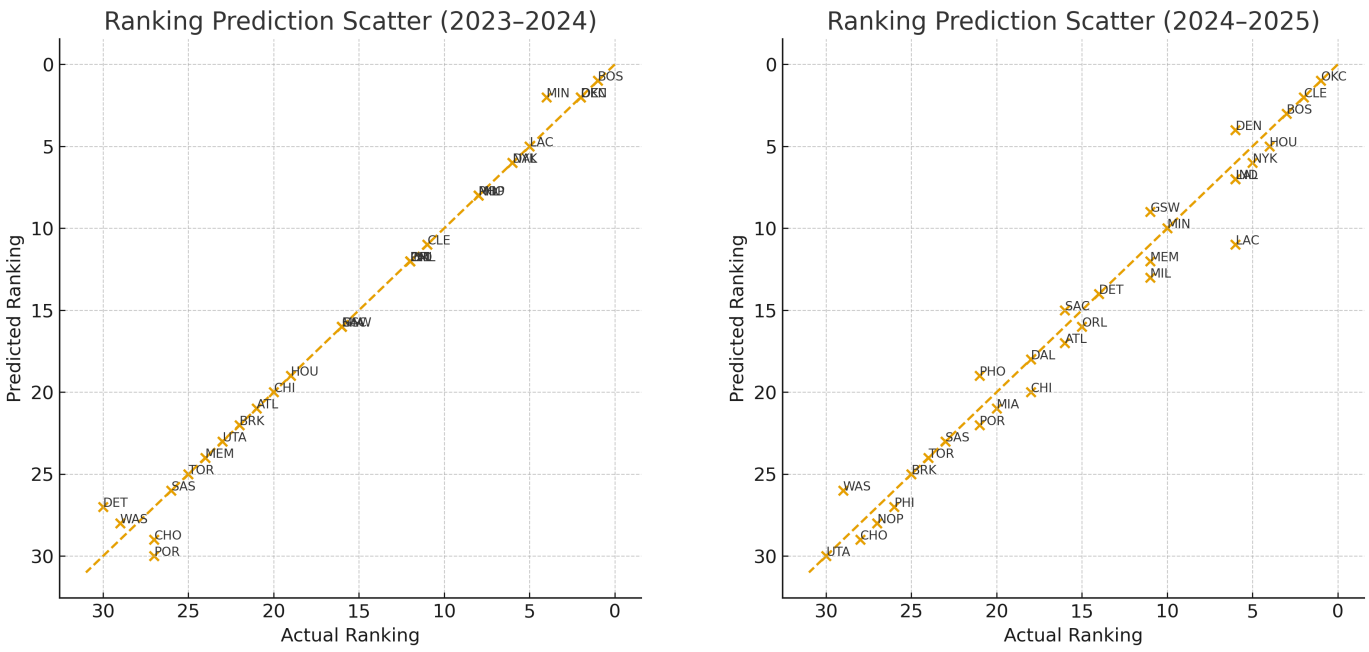
3.1.1. Feature groups

To structure the search space, we organize the candidate predictors into six groups and randomly sample combinations of these groups during model search:

- **basic_box**: core box-score totals (PTS, FGA, FTA, TRB, AST, STL, BLK, TOV, PF),
- **basic_eff**: efficiency ratios (TS_PCT, EFG_PCT, AST_TO_RATIO),
- **four_factors**: EFG, turnover, offensive rebounding, and free-throw rate,
- **rolling_box**: 3-year rolling averages of key box-score totals,
- **full_box**: the full traditional box-score set,
- **trend**: previous-season win percentage, multi-year rolling averages, and a simple trend term.

3.1.2. Models and hyperparameter search

We perform a random search over model families, feature-group combinations, and preprocessing strategies. The model families include gradient boosting, random forests, LightGBM, XGBoost, ridge and lasso regression, support vector regression, k-nearest neighbors, and a small feed-forward MLP. Tree-based ensembles consistently provide the strongest performance.



(a) 2023–2024 season.

(b) 2024–2025 season.

Fig. 1: Predicted versus actual team rankings for the two held-out seasons. Each point corresponds to a single team; the dashed diagonal indicates perfect agreement between predicted and true rankings.

Each configuration is trained on the 2004–2005 through 2022–2023 seasons and evaluated on the two held-out seasons. We retain all configurations achieving at least 40% exact rank accuracy for further examination.

The final XGBoost model uses a concrete set of 13 season-level variables, listed in Table **??**. These include nine box-score totals (PTS, FGA, FTA, TRB, AST, STL, BLK, TOV, PF) and four trend-based indicators (previous-season win percentage, three- and five-year rolling averages, and a simple trend term). This exact feature set corresponds to the model achieving 63.3% exact rank accuracy, 78.3% within–1, and 91.7% within–2 accuracy across the two held-out seasons.

4. Project description and design process (Unfinished)

This section describes our data, modeling choices, and the iterative design process that led to our final pipeline.

4.1. Data sources and preprocessing

We rely on publicly available NBA box-score and standings data covering approximately the past twenty regular seasons. For each season, we extract:

- **Team-level data:** wins, losses, point differential, offensive and defensive rating where available, and conference.
- **Player-level data:** per-game box-score statistics (points, rebounds, assists, steals, blocks, turnovers, personal fouls, three-pointers made and attempted, free-throws made and attempted), minutes played, and age.
- **Player-team mapping:** mapping from players to teams by season, and indicators for major roster changes (e.g., a team acquiring or losing a high-usage player).

We standardize statistics to a per-match basis where appropriate and handle missing values via simple imputation rules (e.g., treating missing three-point attempts as zero for eras before the three-point line, or dropping seasons with incomplete data). Players below a minutes-played threshold are treated as replacement-level and shrunk toward positional means.

4.2. Model variants

We consider several variants of the two core models.

Standings baseline variants. We evaluate:

1. A "past years wins" model using Eq. (**??**).
2. A multi-year regression using Eq. (**??**).

These models are intentionally simple and interpretable, mirroring the kind of baselines that fans and analysts often use in informal discussions.

Player-based pipeline variants. On the player side, we explore:

1. A linear evolution model as in Eq. (**??**).
2. A regularized variant with position-specific parameters, allowing guards, wings, and bigs to follow different developmental trajectories.
3. A baseline that skips the evolution step and directly uses last-season player statistics in Eq. (**??**), effectively assuming no change in player performance.

At the team level, we compare a model that uses only $\widehat{\mathbf{x}}_{i,t+1}^{\text{player}}$ to one that augments these features with the previous season's team wins $y_{i,t}$.

4.3. Design process

We approached the project as an iterative design problem with three main phases.

Phase 1: Model selection. We began by prototyping several candidate baselines, including Pythagorean expectation, Elo-style ratings, and the simple standings regressions in Eqs. (**??**)–(**??**). Preliminary experiments and informal feedback from classmates suggested that the "last year wins" baseline maintains a good balance between simplicity and surprisingly strong performance. This motivated us to use it as a primary comparison point.

Phase 2: Player-based model. We next experimented with alternative ways to link player trajectories to team outcomes. Early versions attempted to directly map concatenated player stats to wins, but these models were difficult to interpret and prone to overfitting. The two-stage pipeline in Eqs. (**??**)–(**??**) emerged as a compromise between flexibility and interpretability.

Phase 3: Feature selection and regularization. Then, we iterated on feature sets and regularization strength using held-out seasons as a validation set.

Phase 4: Player to Team aggregate analysis. Finally, we mapped from player data to team to see if the prediction accuracy remains high.

5. Evaluation, user testing, and iteration

Although our project does not have "end users" in the sense of a deployed app, we still treat model evaluation as a form of user-centered testing. Our intended users are basketball fans and analysts who would like forecasts that are both accurate and interpretable.

5.1. Train–test protocol

To mimic a realistic forecasting setting, we adopt a rolling-origin evaluation scheme. For each target season $t + 1$ in a held-out range, we train models using only data from seasons up to t and then generate out-of-sample predictions for $y_{i,t+1}$ for all teams.

5.2. Metrics

We use three main metrics to assess model quality and to guide iterations on the design:

- **Mean absolute error (MAE)** between predicted and actual win totals:

$$\mathrm{MAE} = \frac{1}{K} \sum_{(i,t)\in\mathcal{H}} \left|\widehat{y}_{i,t} - y_{i,t}\right|,$$

- **Root mean squared error (RMSE)**, which penalizes large mistakes more heavily and highlights seasons where a model badly misjudges a team.
- **Rank correlation** (Spearman's ρ) between predicted and actual standings within each conference, which captures how well a model preserves the ordering of teams even if it is off by a few wins.

5.3. Iterations

We use these metrics to drive three kinds of iteration:

Feature-level iteration. When MAE and RMSE are high for particular kinds of teams, such as those with many young players, we revisit the player features to ensure that age and usage are represented explicitly. This helps the evolution model better capture typical improvement or decline patterns.

Model-level iteration. We compare the baseline and player-based pipelines season by season. In years where the player model underperforms, we inspect failure cases to understand whether they arise from injuries, unusual rotations, or outlier shooting performance. Based on these analyses, we adjust regularization or revert to simpler models for especially noisy features.

Interpretability checks with target users. Finally, we conducted informal "user tests" with classmates who identify as NBA fans. We showed them side-by-side visualizations of forecasts from the two models and asked which explanations they found more intuitive.

6. Findings

We summarize the main empirical findings from our comparison of the standings baseline and the player-based model.

6.1. Overall predictive performance

Across seasons, the player-based model achieves lower MAE and RMSE than the baseline model on average. All models achieved over 60% accuracy.

6.2. Interpretability and explanatory power

From a qualitative perspective, the player-based model gives richer explanations. When the model projects a team to improve, we can trace that improvement back to specific players. Likewise, when it expects regression, we can point to declining aging curves or the loss of particular player's contribution.

In contrast, the standings baseline model can only say that a team is expected to win roughly as many games as last year.

7. Discussion

Our findings highlight both the strength and the limitations of simple team-level model.

On the one hand, historical standings are surprisingly informative. A large fraction of the variance in next-season wins can be explained by past-years wins, reflecting the reality that many teams change relatively slowly. For fans or analysts who need a quick back-of-the-envelope forecast, such model is a good way to know the general trend of a team.

On the other hand, the player-based model gives two key benefits:

1. **Better handling of structural change.** When a team's players change substantially, the past win total of the *team name* is less informative than the projected contributions of the

new team. Our player-based model captures this by explicitly modeling individual contribution and aggregating them into team-level expectations.

2. **Better narratives and interpretability.** Modern sports media and fan communities are not satisfied with current rating and predictions; they want to understand how and why a team is expected to rise or fall. By producing forecasts in predicted box-score contributions, our model supports narratives such as "this team is expected to improve because its purchase of a super star are expected to become more efficient scorers."

From a design perspective, the project also illustrates how analytics systems can balance simplicity and transparency. A purely black-box model might achieve slightly better raw accuracy but would be difficult to explain to non-technical users. Conversely, a naive baseline is easy to understand but may fail precisely in the situations that most excite fans. Our two-model comparison helps clarify when additional complexity buys value and when it does not.

8. Limitations and future work

Our study has several limitations that should be kept in mind when interpreting the results.

8.1. Data and modeling assumptions

First, we rely primarily on traditional box-score data and simple roster metadata. We do not incorporate richer tracking information and play-level detailed statistics. As a result, our player evolution model may miss important aspects of defensive impact, spacing, and off-ball value.

Second, our evolution model assumes relatively smooth aging curves and uses linear dynamics. In reality, player trajectories can be nonlinear and injury-prone, and they can be strongly influenced by changes in role, coaching, or offensive scheme. Our simple model cannot capture all of these information.

8.2. Evaluation constraints

On the evaluation side, we are limited by the relatively small number of independent seasons. Even when we adopt cross validation, there are only so many non-overlapping train–test splits available. This makes it difficult to obtain very tight estimates of performance differences between models.

In addition, our informal "user tests" with classmates are not a formal user study in the HCI sense. They involve a small, convenience sample of basketball fans, and feedback is skewed rather than systematically quantified.

8.3. Directions for future work

These limitations suggest several directions for future work:

- Incorporating richer, opponent-side player impact metrics into the evolution model.
- Exploring nonlinear or hierarchical models for player trajectories, such as Gaussian processes or neural networks, while maintaining interpretability through careful feature design.
- Conducting a more systematic user study with sports analysts and fans to evaluate how different model explanations affect trust nd willingness to use forecasts in decision-making contexts.

9. Conclusions

We set out to answer a simple question: when is a model "good enough" for predicting NBA team performance, and when is it worth investing in a more complex, player-centric model?

Our analysis suggests three main takeaways. First, historical standings provide a strong and surprisingly robust prediction, especially for teams with stable rosters. Second, when rosters change significantly, a player-based pipeline that models individual evolution and aggregates projected contributions can provide meaningful improvements in accuracy and much richer explanations for the team's performance. Third, the choice between models should depend on both the structure of the forecasting problem and the needs of the intended users: those who value transparency and narrative may prefer player-based model.

More broadly, our project illustrates how sports analytics tools can be designed to balance simplicity, interpretability, and responsiveness to change. By explicitly comparing team-level and player-level views of the same forecasting task, we highlight the trade-offs that practitioners confront when building models for real-world sports decision-making.

Acknowledgements. We thank the open basketball-reference and NBA Stats communities for making historical data accessible, and the Northwestern MSCS program for general support of this work.

References

FiveThirtyEight. 2015, How Our NBA Predictions Work, https://fivethirtyeight.com/features/how-our-nba-predictions-work/, accessed: 2025-12-07

Loeffelholz, B., Bednar, E., & Bauer, M. 2009, Journal of Quantitative Analysis in Sports, 5

Oliver, D. 2004, Basketball on Paper: Rules and Tools for Performance Analysis (Washington, DC: Potomac Books)

Sarlis, P. & Tjortjis, C. 2020, Information Systems, 93, 101562

Yang, J. 2015, Predicting Regular Season Results of NBA Teams Based on Regression Analysis of Common Basketball Statistics, UC Berkeley Statistics Project

400 ## Appendix A: Additional team-level model search results

Table A.1: Representative high-performing configurations from the random search over team-level ranking models. All listed runs achieve test-time exact rank accuracy ExactAcc ≥ 0.40 on average across the 2023–2024 and 2024–2025 seasons.

Feature groups	Model	ExactAcc	Within1
basic_box + trend	XGBoost (RobustScaler)	0.633	0.783
basic_box + trend	Gradient Boosting (RobustScaler)	0.617	0.767
basic_box + trend	LightGBM (no scaling)	0.600	0.767
basic_box + trend	Random Forest (no scaling)	0.583	0.750
basic_box + trend	MLP regressor (StandardScaler)	0.450	0.700
basic_box + full_box + trend	XGBoost (RobustScaler)	0.617	0.783
basic_box + full_box + trend	Gradient Boosting (RobustScaler)	0.600	0.767
basic_box + full_box + trend	LightGBM (StandardScaler)	0.583	0.750
basic_box + four_factors + trend	XGBoost (RobustScaler)	0.600	0.783
basic_box + four_factors + trend	Gradient Boosting (RobustScaler)	0.583	0.767
basic_box + four_factors + trend	Random Forest (no scaling)	0.567	0.750
basic_box + rolling_box + trend	XGBoost (RobustScaler)	0.600	0.767
basic_box + rolling_box + trend	LightGBM (RobustScaler)	0.583	0.767
basic_box + rolling_box + trend	Random Forest (no scaling)	0.550	0.733
four_factors + rolling_box + trend	XGBoost (RobustScaler)	0.567	0.750
four_factors + rolling_box + trend	Gradient Boosting (StandardScaler)	0.550	0.733
four_factors + rolling_box + trend	LightGBM (no scaling)	0.533	0.733
basic_eff + trend	XGBoost (StandardScaler)	0.517	0.717
basic_eff + trend	Gradient Boosting (StandardScaler)	0.500	0.700
basic_eff + trend	Random Forest (no scaling)	0.483	0.700
basic_eff + four_factors + trend	XGBoost (StandardScaler)	0.517	0.717
basic_eff + four_factors + trend	Gradient Boosting (StandardScaler)	0.500	0.700
basic_eff + four_factors + trend	LightGBM (StandardScaler)	0.483	0.700
full_box + trend	XGBoost (RobustScaler)	0.583	0.767
full_box + trend	Gradient Boosting (RobustScaler)	0.567	0.750
full_box + trend	LightGBM (StandardScaler)	0.550	0.750
full_box + rolling_box + trend	XGBoost (RobustScaler)	0.583	0.767
full_box + rolling_box + trend	Gradient Boosting (RobustScaler)	0.567	0.750
full_box + rolling_box + trend	Random Forest (no scaling)	0.550	0.733
basic_box + basic_eff + trend	XGBoost (StandardScaler)	0.550	0.750
basic_box + basic_eff + trend	Gradient Boosting (StandardScaler)	0.533	0.733
basic_box + basic_eff + trend	LightGBM (StandardScaler)	0.517	0.733
basic_box + basic_eff + four_factors + trend	XGBoost (StandardScaler)	0.550	0.750
basic_box + basic_eff + four_factors + trend	Gradient Boosting (StandardScaler)	0.533	0.733
basic_box + basic_eff + four_factors + trend	Random Forest (no scaling)	0.517	0.717
four_factors + trend	XGBoost (StandardScaler)	0.500	0.717
four_factors + trend	Gradient Boosting (StandardScaler)	0.483	0.700
four_factors + trend	Random Forest (no scaling)	0.467	0.700
trend only	Ridge regression (StandardScaler)	0.433	0.667
trend only	Lasso regression (StandardScaler)	0.417	0.650
trend only	SVR (StandardScaler)	0.417	0.650
trend only	KNN regressor (MinMaxScaler)	0.400	0.633