Astronomy & Astrophysics manuscript no. aa
December 8, 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

December 8, 2025

ABSTRACT

Publicly available NBA box-score and standings data make it possible to build detailed models of team and player performance, yet many forecasting systems still rely on simple team-level summaries such as past wins or win percentage. In this project, we systematically compare team-centric and player-centric pipelines for predicting regular-season team performance and related player outcomes over roughly two decades of NBA data. First, we build a historical-standings baseline that regresses team win percentage on features constructed from past seasons (previous win rate, rolling 3-year and 5-year averages, and short-term trends). Second, we train player-level models on season-long box-score statistics to: (i) predict All-Star selection, (ii) predict whether a player receives All-NBA/MVP–related honors, and (iii) classify players into broad positional groups (Guard / Wing / Big). Third, we aggregate player statistics back to the team level to construct player-informed team features and use these to predict next-season team win percentage. Finally, we implement an automatic model search over feature groups, preprocessing schemes, and regression algorithms, selecting configurations that maximize ranking-based metrics on the two most recent seasons. Across held-out seasons, historical win–trend models already achieve high R^2 and correlation with true win percentage, while player-aggregated models and the auto-optimized pipeline further improve ranking accuracy for recent seasons. Our results suggest that simple team-level trends are strong baselines, but player-level modeling and automated pipeline search add value for capturing league-wide standings and connecting roster composition to expected team performance.

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

1. Introduction

In the modern NBA, decisions about roster construction, trades, and minutes allocation are increasingly driven by data. Teams, media, and fans all care about the same core question: given what we know about past seasons, how well can we forecast future team performance and key player outcomes? With two decades of publicly available box-score and standings data, it is now possible to go beyond simple narratives and rigorously test which modeling choices actually help for this kind of prediction.

A natural starting point is a team-centric view: treat each franchise as a single unit and summarize its history through wins, losses, and a small set of trend features. This mirrors many real-world forecasting systems that rely on past win percentage and simple trends to project future success. Such models are easy to interpret and compute, but they ignore who is actually on the roster and how individual players evolve over time.

Complementing this is a player-centric view, where we model players directly and then reason upward to teams. Season-level player box-score statistics capture scoring, playmaking, defense, and usage patterns that drive both individual recognition and team success. In this project, we use player data to tackle three prediction problems: (1) identifying All-Star–caliber seasons, (2) predicting whether a player will receive All-NBA or MVP–related honors, and (3) classifying players into broad positional groups (Guard / Wing / Big). These tasks test whether basic box-score data is sufficient to recover meaningful structure in how the league evaluates and uses players.

We then link the player and team perspectives by aggregating player statistics back to the team level and combining them with historical win–trend features. On top of this team-level table, we build and evaluate a variety of regression pipelines to predict team win percentage and resulting standings. Rather than hand-picking a single model, we run an automatic search over feature groups, preprocessing schemes, and algorithms and select configurations that perform best on ranking-based metrics for the final two seasons.

Together, these components allow us to ask a concrete question: when is a simple team-level win–trend baseline "good enough," and when do richer player-level features and automated model selection provide real gains in predicting how the NBA standings will shake out? This project aims to quantify that trade-off and clarify how much value is added by moving from coarse team summaries to more detailed, player-informed representations.

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 and 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 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 family of models, which relies on historical team wins and win-percentage trends. However, because these methods operate on aggregate team outcomes, they are limited in their ability to anticipate abrupt changes in performance driven by roster turnover or shifts in player roles, and they rarely make explicit use of the rich player-level data that are now widely available.

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 and 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, with a small number of hand-chosen algorithms and loss functions.

In contrast, our project situates season-level win-percentage prediction in a broader model-selection and evaluation framework. We combine simple historical win-percentage trends with richer team- and player-derived features, and systematically compare a large family of models (linear methods, tree ensembles, neural networks, kernel methods, k-NN, and simple ensembles) under a common ranking-based evaluation on held-out seasons. This lets us assess not only whether machine learning improves over simple standings-based baselines, but also which combinations of features, preprocessing, and algorithms are most effective for predicting the structure of the final league table.

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 and 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 concentrate on a single type of outcome (e.g., efficiency ratings or win shares), and most studies treat player-level prediction and team-level season forecasting as separate problems. There is comparatively little work that simultaneously (i) uses player box-score statistics to predict league recognition and role (e.g., awards or coarse positional groups), (ii) aggregates those same player-level statistics into team-level profiles, and (iii) compares these player-informed team models against strong, standings-based baselines under a unified evaluation protocol.

Our project is designed to fill this gap. We train player-level classifiers on season-long box-score features to predict All-Star selection, All-NBA/MVP–related honors, and coarse positional categories (Guard / Wing / Big), aggregate player statistics back to the team level to construct player-informed team features, and place these models alongside a purely standings-based baseline and an automatically tuned family of regression pipelines. Evaluating all models on the same task—predicting team win percentage and resulting standings over held-out seasons—allows us to quantify when player-informed models meaningfully improve upon historical win–trend baselines and to connect individual player profiles to expected team performance.

3. Tasks and evaluation

We organize our project around four related prediction tasks:

1. a *standings-only* baseline that forecasts team win percentage from historical wins and losses;
2. three *player-level* classification tasks (All-Star selection, major awards, and coarse position);
3. a *team-level* regression model that predicts team win percentage from richer team box-score and trend features; and
4. an *auto-tuned ranking model* that searches over feature groups, preprocessing strategies, and model families to maximize team-ranking accuracy.

The team-level tasks (1, 3, and 4) all operate at the season-by-team level but differ in how much information beyond past wins they use. The player-level tasks (2) operate on individual season-by-player rows and are evaluated independently. Below we describe the targets, data splits, and evaluation metrics used for each group of tasks.

3.1. Team-level win-percentage prediction from standings only

For the standings-only baseline, we work directly with season-level win–loss records. For each team t and season s, we define the realized winning percentage

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

and construct a feature vector $\mathbf{x}_{t,s}^{\text{stand}}$ from team-specific historical trends: previous-season win percentage, three- and five-year rolling averages, and a short-term trend in the rolling average. A linear regression model f_θ is trained to predict

$$\hat{y}_{t,s} = f_\theta(\mathbf{x}_{t,s}^{\text{stand}}),$$

interpreted as a forecast of WIN_PCT$_{t,s}$.

We use team–season summaries from the 2004–2005 through 2024–2025 seasons (630 team–season observations). The two most recent seasons (2023–2024 and 2024–2025) are held out for evaluation; all earlier seasons are used for training. Trend features are computed on the full panel, but model fitting uses only the training seasons. We evaluate using standard regression and correlation metrics computed separately for each held-out season: MAE, RMSE, R^2, and Pearson and Spearman correlation between predicted and realized win percentage.

3.2. Player-level prediction tasks

We consider three player-centric classification tasks, each defined on season-by-player rows with box-score features and temporally separated train/test splits.

All-Star selection. For each player p and season s, we build a feature vector $\mathbf{x}_{p,s}^{\text{AS}}$ from counting and rate statistics (minutes, shooting volume and efficiency, rebounding, playmaking, and defensive stats), together with one-hot encoded position and team indicators. The target label $y_{p,s}^{\text{AS}} \in \{0, 1\}$ indicates whether the player received an All-Star designation that season, obtained by parsing the awards string. We train on earlier seasons and evaluate on the last three.

All-NBA / MVP awards. The second player-level task predicts whether a player appears in major end-of-season awards. The feature vector $\mathbf{x}_{p,s}^{\text{MVP}}$ uses all numeric box-score features, and the binary label $y_{p,s}^{\text{MVP}}$ is set to 1 if the awards string contains any All-NBA or MVP-related marker and 0 otherwise. To handle extreme class imbalance, we learn an XGBoost classifier with a class-weighting term derived from the positive-to-negative ratio in the training data.

Coarse position classification. The third player-level task maps each player-season to one of three coarse position groups: Guard (PG/SG), Wing (SF), or Big (PF/C). We derive a label $y_{p,s}^{\text{pos}} \in \{\text{Guard}, \text{Wing}, \text{Big}\}$ by collapsing raw position strings and discarding ambiguous entries. The feature vector $\mathbf{x}_{p,s}^{\text{pos}}$ includes both raw box-score quantities and engineered features such as per-36-minute statistics, shooting rates, assist-to-turnover ratio, and a usage proxy. We again train on earlier seasons and test on the most recent seasons (starting in 2023).

For all three player-level tasks, we evaluate classifiers using overall accuracy on held-out seasons, per-class precision/recall/F1, and confusion matrices. For the multi-class position task, we also compare against a majority-class baseline.

3.3. Team-level ranking prediction with rich features

Beyond the standings-only baseline, we train models that use richer season-level team statistics to predict team strength. For each team t and season s, we assemble a feature vector $\mathbf{x}_{t,s}^{\text{team}}$ from box-score totals, efficiency metrics, and trend features (described below) and train a regression model g_ϕ to output a scalar score

$$\hat{z}_{t,s} = g_\phi(\mathbf{x}_{t,s}^{\text{team}}),$$

which we use as a proxy for team quality. Within each season, teams are ranked by sorting $\hat{z}_{t,s}$ in descending order. We reuse the same temporal split as in the standings-only baseline: 2023–2024 and 2024–2025 are held out, and all prior seasons are used for training. Feature engineering (rolling box-score averages and win-percentage trends) is performed on the full panel, but model fitting never accesses the held-out seasons.

Since the main objective is to recover the correct *ordering* of teams within a season, we evaluate team-level predictions using rank-based metrics. Let $\text{RANK}_{t,s}^{\text{true}}$ denote the rank of team t in season s when teams are sorted by realized winning percentage (1 = best), and let $\text{RANK}_{t,s}^{\text{pred}}$ be the rank when sorted by $\hat{z}_{t,s}$. For each season s with N_s teams, we compute exact rank accuracy and the fractions of teams whose predicted rank is within one or two places of the truth:

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

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

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

We report each metric for the two held-out seasons separately and also average them across seasons. Figure 1 visualizes predicted versus actual rankings for the held-out seasons under the best-performing configuration.

3.3.1. Feature groups

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

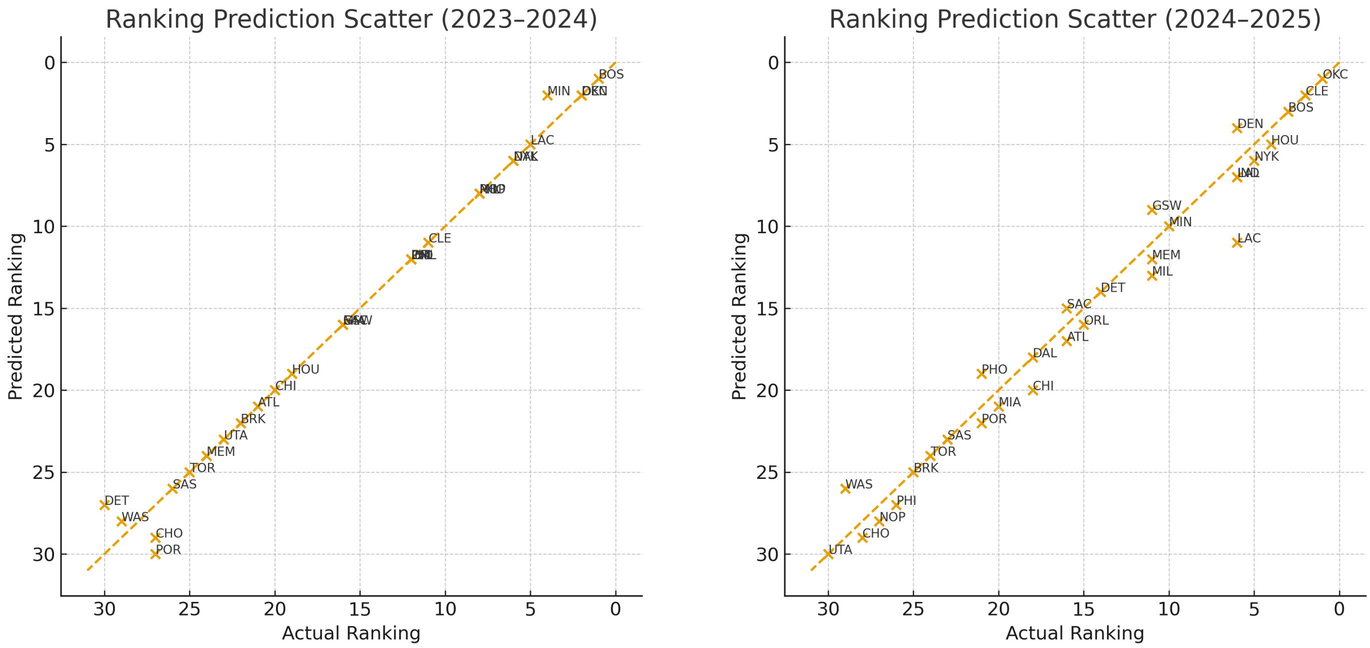
- **basic_box**: core box-score totals (PTS, FGA, FTA, TRB, AST, STL, BLK, TOV, PF),
- **basic_eff**: efficiency ratios (TS_PCT, EFG_PCT, AST_TO_RATIO) and previous-season win percentage (PREV_WIN_PCT),
- **four_factors**: the “Four Factors” metrics (effective field-goal percentage, turnover factor, offensive rebounding factor, and free-throw rate),
- **rolling_box**: 3-year rolling averages of key box-score totals,
- **full_box**: the full traditional box-score set, and
- **trend**: win-percentage trend features (three- and five-year rolling averages and a short-term trend).

Each sampled configuration selects a subset of these groups, and the corresponding union of columns is used as $\mathbf{x}_{t,s}^{\text{team}}$.

3.3.2. Models, preprocessing, and hyperparameter search

We perform a random search over model families, feature-group combinations, and preprocessing strategies. Model families include gradient boosting, random forests, LightGBM, XGBoost, ridge and lasso regression, support vector regression, k-nearest neighbors, and small multilayer perceptrons. For each model, we sample hyperparameters from predefined ranges.

Before model fitting, we also sample a preprocessing strategy from a small set of options: standardization, robust scaling, min–max scaling, or no scaling. For a given configuration, we:



(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.

1. build the selected feature matrix on all training seasons,
2. fit the chosen preprocessing transform (if any) and model on the training data, and
3. evaluate the resulting model on the held-out seasons using the rank-based metrics above.

We record all configurations and retain those that achieve strong ranking performance (e.g., high exact and Within1 accuracy) on the two held-out seasons for further analysis. Additional diagnostic summaries of the search space are provided in Appendix A.

4. Project description and design process

This section describes the concrete data we use and the models we implement for each task, along with the main design choices that led to the final pipeline.

4.1. Data sources and preprocessing

We work with season-level NBA team and player statistics spanning roughly the 2004–2005 through 2024–2025 regular seasons. All data come from publicly available box-score and standings tables and are stored in CSV files processed with Python.

Team-level data. At the team level we use two types of tables:

- **Win–loss summaries** (`Team_stats/WL`): for each team and season, we have wins, losses, and a team abbreviation. From these we compute the realized winning percentage WIN_PCT and trend features: previous-season win percentage, three- and five-year rolling averages, and a short-term trend for each franchise.
- **Expanded team summaries** (`Team_stats/all_seasons_team_summary.csv`): these tables contain traditional box-score totals, shooting percentages, and win–loss records for every team-season. We use these to derive efficiency metrics (true shooting percentage, effective field-goal percentage, assist-to-turnover ratio) and the "Four Factors," as well as three-year rolling averages of key stats.

We clean numeric columns by replacing infinite values with `NaN` and dropping rows with missing wins or losses. Seasons are indexed by both their string form (e.g., 2023–2024) and an integer end year (e.g., 2024) to support temporal splitting and rolling-window computations.

Player-level data. At the player level we use a single long panel (`all_stats/player_20years.csv`) containing one row per player-season with per-game box-score stats, minutes, age, team, raw position string, an awards string, and a season identifier.

For preprocessing we:

- filter out rows with zero games played,
- drop the aggregated "TOT" rows for players traded mid-season,
- derive season start and end years from the season string, and
- apply a minimum minutes-played threshold (e.g., 500 minutes) when learning player positions.

For the position classification task we engineer additional features: per-36-minute versions of core stats, per-36 shooting volume, and ratio-based descriptors such as three-point and free-throw rate, assist-to-turnover ratio, and a usage proxy based on scoring attempts.

Label construction. The three player-level tasks require different labels:

- **All-Star label**: parse the awards string for the substring `"AS"` and set a binary label accordingly.
- **All-NBA / MVP label**: define `LABEL_ALLNBA_MVP` by checking whether an awards string contains markers related to All-NBA teams or MVP voting.
- **Coarse position label**: map raw position strings into Guard (PG/SG), Wing (SF), and Big (PF/C), dropping ambiguous entries.

All temporal splits use these season identifiers: earlier seasons for training and the most recent ones as held-out test sets.

4.2. Model variants by task

We implement different model families tailored to each of the four main tasks introduced in Section 3.

Task 1: Standings-only team baseline. We use a linear regression model trained on team-level trend features derived from historical win–loss records (previous-season win percentage, three- and five-year rolling averages, and a short-term trend). Features are standardized before training. The model is trained on all but the last two seasons and evaluated on 2023–2024 and 2024–2025.

Task 2: Player-level classification. We train three separate classifiers on the long player panel.

- **All-Star prediction**: a `GradientBoostingClassifier` with 400 trees and depth 3. The input combines numeric box-score statistics with one-hot encoded team and position indicators via a column-wise preprocessing pipeline.
- **All-NBA / MVP prediction**: an `XGBClassifier` that ingests all numeric features except identifiers. To address class imbalance, we compute a `scale_pos_weight` from the positive rate in the training set and pass it to XGBoost.
- **Coarse position classification**: an XGBoost multi-class classifier on the enriched feature set with pre-tuned hyperparameters (e.g., 600 trees, depth 4, learning rate 0.1). We filter out low-minute seasons and standardize features before training.

Each classifier is evaluated on a temporally separated test set and compared to simple baselines such as always predicting the majority class (for positions).

Task 3: Player-to-team aggregation model. To connect player performance back to team outcomes, we build a team-level panel by aggregating player statistics up to the team-season level and merging them with official team summaries.

We group player rows by season and team, summing counting stats and averaging age. From these we recompute team-level shooting percentages, true shooting percentage, effective field-goal percentage, and assist-to-turnover ratio. We then merge this player-derived table with the official team summary table and compute win-percentage trend features as before. On this merged panel we train a `RandomForestRegressor` on all seasons except the last two, targeting team win percentage and evaluating on 2023–2024 and 2024–2025.

Task 4: Auto-tuned team ranking model. Finally, we construct an automatic model-search pipeline on the expanded team summary table. For each random configuration, we:

1. sample one or more feature groups from the six groups in Section 3 (never using trend features alone);
2. build the corresponding feature matrix on all training seasons;
3. sample a preprocessing strategy from {`StandardScaler`, `RobustScaler`, `MinMaxScaler`, none};
4. sample a model family and its hyperparameters from a pool that includes LightGBM, XGBoost, random forests, gradient boosting, ridge and lasso regression, support vector regression, *k*-NN, simple multilayer perceptrons, and a few voting ensembles; and
5. fit the model on all non-held-out seasons and evaluate its ranking accuracy on the two held-out seasons using the metrics in Section 3.

We track the performance and configuration of each run and export the detailed predictions of the global best model for qualitative inspection.

4.3. Design process

We approached the project as an iterative design process that gradually increased model complexity while keeping temporally realistic evaluation.

Phase 1: Simple team baselines. We started with the most transparent team-level baselines: predicting future team win percentage from only past win–loss records. Early experiments showed that a linear model on a small set of historical win-percentage statistics already explains a large fraction of variance in future win percentage. This motivated treating a standings-only trend model as a central baseline and fixing a strict temporal split.

Phase 2: Player-level modeling. We then moved to player-level tasks to understand how much signal is available in individual statistics. We chose All-Star, major awards, and coarse position because they are tied to real basketball concepts and allow us to quantify how well gradient boosting and XGBoost models can recover expert decisions and positional roles from box-score data alone. Severe class imbalance for awards led us to add explicit class weighting and to monitor per-class metrics rather than only overall accuracy.

Phase 3: From players back to teams. Next, we asked whether aggregating player stats back to teams would help predict future team performance. Initial high-dimensional designs that concatenated player vectors proved fragile and hard to interpret, so we settled on a simpler aggregation that recomputes familiar team-level metrics and feeds them into a tree-based regressor. This provided a stable bridge between the player and team tasks.

Phase 4: Automated model and feature search. Finally, to avoid hand-picking feature sets and models at the team level, we built an auto-optimizer that samples feature-group combinations, preprocessing methods, and model hyperparameters and evaluates them against the same held-out seasons using rank-based metrics. This framework let us quantify variability across reasonable model choices, identify robustly strong configurations, and position our best model relative to the simpler standings-only baseline.

5. Evaluation, user testing, and iteration

Although our project does not have deployed "end users," we still treat evaluation as a user-centered process. The intended users are basketball fans and analysts who care about both accuracy and interpretability in basketball terms (wins, rankings, awards, positions).

5.1. Temporal train–test protocol

Across all four tasks we use temporally separated train and test sets to approximate realistic forecasting. Rather than randomly splitting within a season, we train on earlier seasons and evaluate on the most recent:

- **Team-level baselines (Tasks 1, 3, 4):** we train on all seasons up to 2022–2023 and treat 2023–2024 and 2024–2025 as held-out test seasons. Feature engineering (e.g., rolling averages) uses only past seasons for each team.
- **Player-level classification (Task 2):** for All-Star and MVP/All-NBA prediction we train on earlier seasons and hold out the last three. For position prediction we train on seasons up to 2022–2023 and test from 2023–2024 onward.

5.2. Metrics by task and iteration

We use different metrics for team-level and player-level tasks, as outlined in Section 3, and we track how these metrics change as we iterate on features and models.

For the standings-only baseline and the player-to-team aggregation model, we monitor MAE, RMSE, R^2, and correlations on the two most recent seasons. Once we observed that a small set of trend features already achieved low MAE and high R^2, we fixed this trend-based baseline as a strong reference point.

In the auto-tuned team model, we focus on ranking metrics (ExactAcc, Within1, Within2). We define a combined score that emphasizes Within1 and use the average of these metrics across the two held-out seasons as the optimizer's objective.

For the All-Star and MVP/All-NBA tasks, early prototypes that optimized only accuracy performed poorly on the rare positive class. Inspecting classification reports and confusion matrices revealed that the model often predicted "no award" for almost everyone. In response, we introduced class weighting in the MVP model, monitored positive-class recall and F1 as primary metrics, and adjusted thresholds to balance precision and recall.

For the position classifier, comparing against the "always Guard" baseline and inspecting confusion matrices showed that Guards and Bigs were learned relatively well, while Wings were harder to distinguish. This guided feature engineering toward adding per-36 rebounding, shooting volume, and usage-related features to better separate wings from guard-sized and big-sized players.

For the player-to-team aggregation pipeline, we compared MAE, RMSE, and correlations to the standings-only baseline on the same held-out seasons. High-dimensional variants did not clearly improve these metrics and were difficult to interpret, so we kept the simpler aggregation that sums player stats and recomputes standard team-level efficiencies.

5.3. Informal user feedback

We gathered informal feedback from classmates who follow the NBA by showing:

- side-by-side tables of predicted vs. actual rankings,
- scatter plots of predicted vs. actual rankings, and
- examples of All-Star and MVP predictions with underlying box-score features.

They found the "within one rank" metric and ranking scatter plots more intuitive than raw error tables, and preferred models whose mistakes were small shifts in rank rather than large misorderings. This feedback reinforced our choice to emphasize ranking metrics in the team-level models and to present classification performance with per-class precision and recall, not just overall accuracy.

6. Findings

We summarize the main empirical findings from our four tasks: (i) team-level forecasting from standings history, (ii) team-level forecasting from rich box-score features, (iii) player-level prediction tasks, and (iv) automated model and feature search.

6.1. Team-level forecasting from W/L history

Using only team wins, losses, and simple trend features (previous-season win percentage and multi-year rolling averages), the baseline linear model already predicts win percentage very well on the two most recent seasons. For the 2023–2024 hold-out season, the model attains MAE $\approx$ 0.032, RMSE $\approx$ 0.043, $R^2 \approx 0.93$, with Pearson and Spearman correlations above 0.92. For 2024–2025, performance remains strong (MAE $\approx$ 0.040, RMSE $\approx$ 0.044, $R^2 \approx 0.92$). A "last-year plus trend" baseline already captures much of the signal in team outcomes.

6.2. Player-level prediction tasks

On the player side, we obtain three complementary findings.

All-Star prediction. A gradient boosting classifier using per-player box-score features, minutes, age, and one-hot team/position indicators achieves 98.2% accuracy on held-out seasons. Despite strong class imbalance (about 3% of players are All-Stars), the model reaches recall 0.88 and F1-score 0.75 for the positive class, recovering most true All-Stars with relatively few false positives.

All-NBA / MVP prediction. An XGBoost classifier trained to detect whether a player earns any All-NBA or MVP-related award achieves 99.1% test accuracy. For the small positive class (about 1.5% of players), the model attains recall 0.90 and F1-score 0.76, suggesting that simple season-long box-score profiles are highly informative about end-of-season award outcomes.

Position classification. For three-way position labels (Guard / Wing / Big), an XGBoost multi-class model with engineered per-36 and rate statistics achieves 76.6% test accuracy, substantially above the majority baseline (44.4%, always predicting "Guard"). Performance is strongest for Guards and Bigs (F1 $\approx$ 0.82–0.84), and weakest for Wings (F1 $\approx$ 0.45), reflecting that Wings overlap statistically with both backcourt and frontcourt roles.

6.3. From players back to teams

Aggregating player-level statistics up to team-season level and then predicting team win percentage yields performance comparable to the trend-only baseline. Using a Random Forest on trend features derived from the player-aggregated data, we obtain, for 2023–2024, MAE ≈ 0.041, RMSE ≈ 0.054, $R^2 \approx 0.89$, and for 2024–2025, MAE ≈ 0.046, RMSE ≈ 0.056, $R^2 \approx 0.88$, with Pearson and Spearman correlations above 0.93. Aggregating player trajectories back to the team level produces realistic forecasts, but does not easily outperform a carefully tuned standings-based baseline on this dataset.

6.4. Best team-level ranking model from automated search

The automated search identifies a gradient boosting regressor using four feature groups: `basic_box`, `basic_eff`, `rolling_box`, and `trend`. Evaluated as a ranking model on the two held-out seasons, this configuration achieves:

- 2023–2024: 83.3% exact rank accuracy, 83.3% within 1, and 96.7% within 2;
- 2024–2025: 40.0% exact rank accuracy, 76.7% within 1, and 93.3% within 2.

Averaged across the two seasons, the model reaches 63.3% exact rank accuracy, 78.3% within 1, and 91.7% within 2, substantially improving on naive baselines that rely only on last year's standings.

6.5. Interpretability and explanatory power

Qualitatively, the different components of the pipeline offer different kinds of explanations:

- The W/L-trend baseline is extremely simple and transparent: it encodes the belief that teams tend to regress slowly toward their historical average.
- The player-level award and position models expose which statistical profiles are associated with "star" seasons and specific roles, making it easy to reason about why a given player is flagged as All-Star, All-NBA/MVP, or a particular position type.
- The player-to-team aggregation closes the loop by showing how the collection of individual contributions translates into expected team success, even if it does not always beat the standings-only baseline numerically.
- The final gradient-boosting ranking model balances accuracy and interpretability at the team level: its feature groups correspond to familiar concepts (box-score totals, efficiency, long-term trends), which can be inspected when explaining why a team is projected higher or lower than its previous record.

Overall, simple standings-based models are competitive, but carefully engineered season-level features and player-centric analyses narrow the remaining performance gap and provide richer, more intuitive explanations for forecasted outcomes.

7. Discussion

Our project set out to build a forecasting pipeline that is both reasonably accurate and understandable for basketball fans and analysts. The findings support three main contributions: (i) quantifying how far simple standings-based models can go, (ii) showing where richer team-level modeling and automated search add value, and (iii) demonstrating how player-level models provide complementary, interpretable views of team strength.

7.1. What simple baselines already capture

A first takeaway is how strong the historical-standings baseline is. Using only past win percentage and a few trend features, the baseline model achieves MAE ≈ 0.03–0.04 and $R^2 \approx 0.92$–0.93 on the two most recent seasons, with correlations above 0.92. Much of next season's variance can be explained by recent team performance alone: rosters, coaching staffs, and organizational strategies tend to change gradually, so win totals show a high degree of inertia. For users who just want a quick, back-of-the-envelope forecast, this type of model already provides a solid answer.

7.2. Where richer team modeling helps

The automated model and feature search demonstrates that carefully chosen team-level features can further improve ranking quality. The gradient boosting model built on `basic_box`, `basic_eff`, `rolling_box`, and `trend` features attains 63.3% exact rank accuracy and more than 90% within–2 across the two held-out seasons. This is a clear gain over relying only on last year's record: the model exploits information about scoring volume, efficiency, rebounding, playmaking, and long-term trends to sort teams more precisely.

At the same time, the gains are not unbounded. In some seasons the optimized model substantially improves exact rank accuracy; in others it mainly refines the ordering within tiers. This suggests a design principle for fan-facing tools: adding complexity beyond simple standings is most useful when grounded in interpretable basketball quantities rather than opaque feature engineering.

7.3. What player-level models add

The player-level tasks highlight a different dimension. The All-Star and All-NBA/MVP classifiers reach high accuracy and strong recall on heavily imbalanced labels, indicating that basic box-score profiles contain enough information to recover most "star seasons." The position classifier shows that Guards and Bigs occupy relatively distinct statistical regions, while Wings are more ambiguous, sitting between backcourt and frontcourt profiles. These models do not directly improve team-level R^2, but they clarify how the system "thinks" about individual roles and star power.

Aggregating player statistics back to the team level yields performance comparable to the standings baseline and clarifies where that baseline may break: when a roster changes dramatically or when a team's star power collapses or spikes. In those settings, being able to trace a forecast back to specific players, aging curves, and usage patterns gives analysts and fans a more satisfying narrative than team-only regressions can provide.

7.4. Implications for fan-facing analytics

From a design perspective, the comparison between models illustrates how to balance simplicity, accuracy, and transparency:

- The standings baseline is easy to explain and serves as a robust default.
- The feature-rich team model offers a measurable boost in ranking accuracy while still using familiar statistics.
- The player-level models and player-to-team aggregation provide narratives about *who* is driving a forecast, even when the overall win prediction is numerically similar to the baseline.

For fan- and analyst-facing systems, the most useful designs are not necessarily the most complex models, but those that combine: (i) a strong, simple baseline, (ii) modest accuracy gains from richer features, and (iii) player-centric explanations that align with how people already talk about the game.

8. Limitations and future work

Our study has several limitations that are important to keep in mind when interpreting the results and thinking about generalization beyond our specific setting.

8.1. Data and modeling assumptions

First, we work entirely within the NBA and rely primarily on traditional box-score data plus simple roster metadata. We do not incorporate richer tracking data, play-by-play information, lineup context, or opponent-adjusted metrics. As a result, both the team-level and player-level models may miss important aspects of defensive impact, spacing, on/off effects, and scheme-specific roles that are not well captured by box-score statistics alone.

Second, the player evolution components assume relatively smooth aging curves and largely linear dynamics in standard features. In reality, player trajectories can be highly nonlinear and injury-prone, with abrupt jumps due to role changes, coaching shifts, or major trades. Our player-to-team aggregation further assumes that team strength is approximately the sum of individual contributions after basic filtering (e.g., minutes thresholds), and does not explicitly model interaction effects such as lineup fit or coaching strategy.

Third, our label choices and thresholds introduce their own biases. All-Star and All-NBA/MVP indicators are influenced by media narratives, fan voting, and award rules, and the extreme class imbalance means that small absolute errors can translate into large swings in precision and recall. Similarly, position labels are collapsed into three broad groups, obscuring finer-grained role distinctions that matter in modern positionless basketball.

8.2. Evaluation constraints

On the evaluation side, we are constrained by the relatively small number of independent seasons. Even with rolling train–test splits, there are only so many non-overlapping temporal splits available. This makes it difficult to obtain very tight confidence intervals on performance differences and raises the risk that some conclusions are specific to the recent era of NBA rules and play style.

For the team-ranking task we evaluate on the two most recent seasons, which provides a realistic forecasting scenario but also means that our best-performing configuration is tuned to a small number of out-of-sample years. Likewise, our informal "user tests" involve a small, convenience sample of classmates; they provide useful qualitative feedback but not a controlled HCI study.

Moreover, for the auto-tuned team-ranking model we implicitly use the same two held-out seasons both to select the best-performing configuration and to report its final performance. Although the test seasons are never used for training the underlying models, this reuse of the evaluation set as a model-selection objective introduces a mild form of leakage: our reported metrics for Task 4 are likely to be somewhat optimistic, and a truly unbiased estimate of generalization would require fixing a configuration first and then evaluating it on additional, unseen future seasons.

8.3. Directions for future work

These limitations suggest several directions for improvement:

- **Richer and more granular data.** Incorporating tracking data, play-by-play logs, opponent-adjusted impact metrics, and explicit injury or availability information could help the models better capture defense, spacing, and lineup context.
- **More flexible but structured models.** Exploring nonlinear or hierarchical models for player trajectories and team strength, while constraining them with interpretable feature sets, could relax smooth-aging and additivity assumptions without sacrificing transparency.
- **Richer evaluation and user studies.** Extending the evaluation to additional seasons, other leagues, or other sports would clarify how robust our design choices are. A more systematic user study with analysts and different types of fans could measure how model explanations affect trust and perceived usefulness.
- **Interactive, user-facing designs.** Building an interactive interface (e.g., allowing users to adjust hypothetical trades or injuries and see updated forecasts) would test how well the pipeline supports real analytical workflows and what additional explanations are needed in practice.

Overall, our work should be viewed as a first step toward combining simple standings-based baselines, richer team-level features, and player-level reasoning in a way that is both quantitatively reasonable and accessible to basketball audiences.

9. Conclusions

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

Our results point to three main takeaways.

First, historical standings by themselves are a remarkably strong baseline. A large share of the variance in future win percentage can be explained by past wins and simple trend features, especially for teams with relatively stable rosters and coaching situations. For many quick-turnaround use cases, this kind of standings-based model offers a lightweight, easy-to-communicate forecast.

Second, player-based pipelines matter most when teams undergo structural change. When rosters shift substantially, a model that explicitly projects individual player trajectories and aggregates their contributions to the team level can improve ranking accuracy and, just as importantly, offer richer explanations of *why* a team is expected to rise or fall. Being able to trace a forecast back to specific players, aging curves, and usage patterns gives analysts and fans a more satisfying narrative than team-only regressions can provide.

Third, the choice between these models should be driven by both the structure of the forecasting problem and the needs of the intended users. In settings where data are limited, rosters are stable, or users mainly need a rough directional signal, a simple standings-based approach may be sufficient. In settings where stakeholders care about transparency, counterfactual scenarios (e.g., trades, injuries), or storytelling around players, a player-centric design offers clear added value even if raw accuracy improves only modestly.

More broadly, our project illustrates how sports analytics tools can be designed to balance simplicity, interpretability, and responsiveness to change. By putting a team-level baseline and a player-level pipeline side by side on the same forecasting task, we highlight the trade-offs that practitioners face when choosing model complexity, and show how user-centered considerations can guide that choice rather than accuracy alone.

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., and Bauer, M. (2009). Predicting NBA games using neural networks. *Journal of Quantitative Analysis in Sports*, 5(3).

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

Sarlis, P. and Tjortjis, C. (2020). Sports analytics: A systematic review of basketball player and team evaluation. *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.

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