# NBA Player Stats and Team Win Prediction

**A Machine Learning Pipeline for CS4661**

**Team Members:** Ryan Dielhenn, Momoka Aung, Angel Trujillo, Jesus Villa, Harshil Patel
**Project Lead:** Ryan Dielhenn
Department of Computer Science
California State University Los Angeles
Course: CS4661 — Introduction to Data Science
**Date:** December 7, 2025

---

## Abstract

This paper presents a complete machine learning pipeline to predict NBA player performance and team results. We use player and team statistics from the 2024–2025 NBA season to predict: (1) individual player points (PTS), and (2) whether a team will win or lose a game. Our models include Linear Regression, Random Forest, and Gradient Boosting for regression tasks, and Logistic Regression, Random Forest, and Gradient Boosting for classification. For predicting player PTS, **Linear Regression** achieved the best performance with **CV RMSE = 2.188**, **CV MAE = 1.589**, and **CV R² = 0.939**. For predicting team wins, **Logistic Regression** achieved **CV Accuracy = 86.2%** and **CV ROC-AUC = 0.934**. We also review key features influencing results in predictions of baseline as well as tuned models. We add more advanced models such as XGBoost and LightGBM to determine if more complex models yield better results.

---

## 1. Introduction

Predicting basketball performance is a common data science problem that helps with scouting, coaching, and fan analysis. We focus on two goals using the same dataset:

1. Predicting player **points scored (PTS)** using regression models.
2. Predicting **team win/loss outcomes** using classification models.

### Contributions

Our project provides:

- A clear and reusable machine learning pipeline.
- Baseline models for regression and classification.
- Insights into which features matter most.
- A framework for future improvements using hyperparameter tuning and boosting techniques.

---

## 2. Data Overview

We use the **NBA player stats (2024–2025 season)** dataset with **16,512 rows** and **25 columns**. Each row represents a player's performance in one game. Features include shooting statistics (FG, 3P, FT), rebounds, assists, steals, blocks, and turnovers. Team, opponent, and result information are also included. There were no missing values after cleaning.

---

## 3. Methodology

### 3.1 Player Points (PTS) Prediction

- **Target:** Player PTS per game.
- **Features:** 16 numerical stats (MP, FGA, 3P, 3PA, FT, rebounds, assists, etc.).
- **Models:** Linear Regression, Random Forest Regressor, and Gradient Boosting Regressor.
- **Split:** 60% training, 40% testing (9,907 train / 6,605 test).

### 3.2 Team Win/Loss Prediction

- **Process:** Aggregate player data to team level (sum of stats, average minutes, and shooting percentages).
- **Target:** Win = 1, Loss = 0 (1,534 total team games).

- **Features:** 18 team stats including FG, FGA, 3P, 3PA, rebounds, assists, and shooting percentages.
- **Models:** Logistic Regression, Random Forest, Gradient Boosting.
- **Split:** Roughly 50/50 win/loss class balance.

### 3.3 Metrics

- **Regression:** RMSE, MAE, $R^2$.
- **Classification:** Accuracy, Precision, Recall, F1, ROC-AUC.

---

## 4. Exploratory Analysis and Feature Engineering

### 4.1 Player-Level Insights

- FGA (field goal attempts), 3P (made threes), and MP (minutes played) are the strongest predictors of PTS.
- Efficiency stats like 3P% and FT% add smaller but useful signals.

### 4.2 Team-Level Insights

- Team FG% and defensive stats (rebounds, steals) are linked to more wins.
- Turnovers and missed shots reduce win probability.

---

## 5. Data Visualization

To better understand the performance of our regression and classification models, we generated several visualizations for both the player-level regression task (PTS prediction) and the team-level classification task (win/loss prediction). These plots help illustrate how well the models fit the data, whether systematic errors exist, and which input features contribute most to the predictions.

### 5.1 Player Points (PTS) Prediction Visualizations

**Figure 1 — Linear Regression: Actual vs Predicted (PTS)**

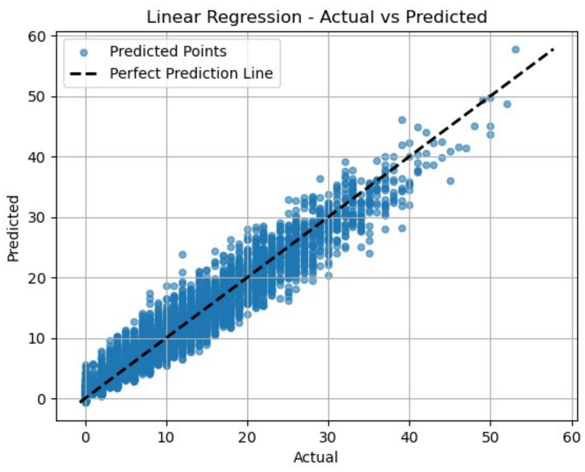

Figure 1: Linear Regression - Actual vs Predicted

This scatter plot compares the true PTS values with the model's predictions. The points fall closely to the diagonal "perfect prediction" line, which matches our evaluation metrics (low RMSE, high $R^2$), confirming that the model predicts player scoring very accurately. The linear structure indicates that player scoring is mostly driven by simple stats like FGA, 3P, and FTA, making it easy to predict.

**Figure 2 — Linear Regression: Residuals Plot (PTS)**

The residual plot shows the difference between predicted and actual values across the full scoring range. The residuals are centered evenly around zero with no obvious pattern, meaning the model is not systematically over- or under-predicting for high- or low-scoring players. This supports the suitability of a linear model for this task and validates the assumptions behind Linear Regression.

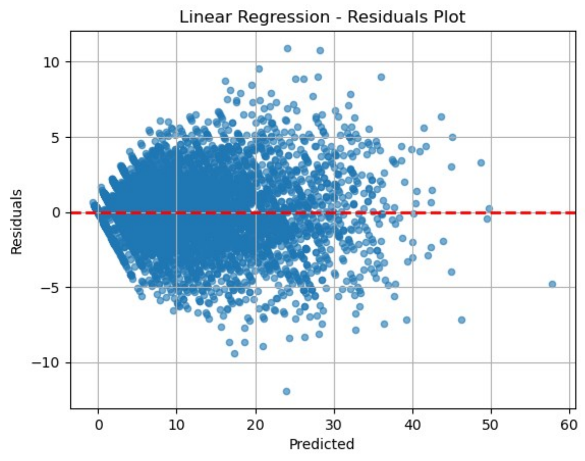

Figure 2: Linear Regression - Residuals Plot

### 5.2 Team Win/Loss Classification Visualizations

**Figure 3 — Logistic Regression: ROC Curve**

The ROC curve illustrates the trade-off between true positive rate and false positive rate. Logistic Regression achieves an AUC of 0.92, indicating excellent discriminatory power. The curve stays close to the top-left corner, showing that the model reliably distinguishes wins from losses across different thresholds. This aligns with the high accuracy, precision, recall, and F1-score observed during evaluation.

**Figure 4 — Logistic Regression: Confusion Matrix**

The confusion matrix summarizes the classification outcomes on the test set. The model correctly identifies a large number of wins and losses, with relatively few misclassifications. True positives and true negatives dominate both diagonals, confirming the model's stability and balanced performance across classes. The errors themselves also appear symmetric, indicating no bias toward predicting wins or losses.

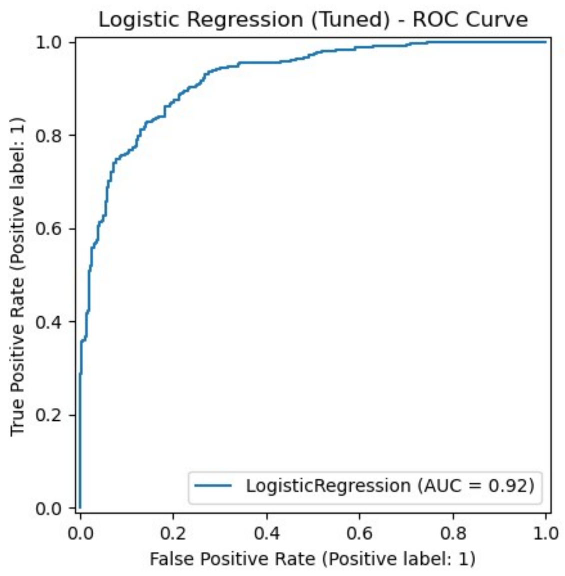

Figure 3: Logistic Regression - ROC Curve

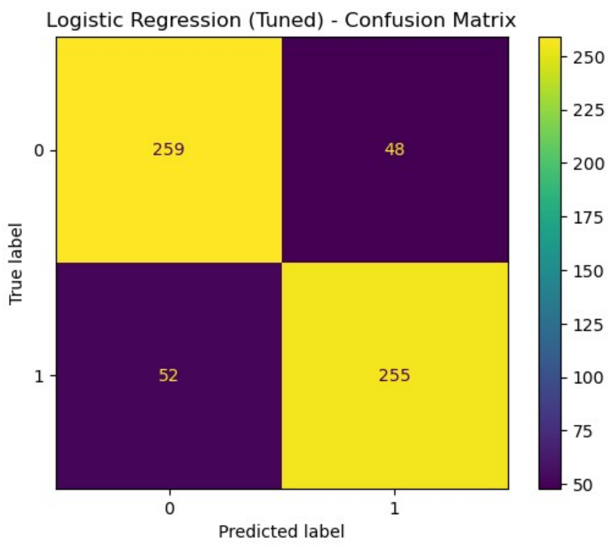

Figure 4: Logistic Regression - Confusion Matrix

## 6. Experiments

### 6.1 Player Points Prediction Results

| Model | RMSE | MAE | $R^2$ |
|---|---|---|---|
| Linear Regression | 2.174 | 1.588 | 0.939 |
| Random Forest | 2.387 | 1.717 | 0.927 |
| Gradient Boosting | 2.275 | 1.662 | 0.933 |

**Observation:** Linear Regression performed the best overall, which suggests that player scoring primarily depends on straightforward, additive relationships between basic statistics such as field-goal attempts, made threes, free throws, and minutes played. This indicates that most of the predictive power comes from volume-based metrics rather than complex interactions. Random Forest and Gradient Boosting also performed well but did not outperform Linear Regression, which implies that non-linear patterns or deeper interactions do exist but are not strong enough to significantly improve prediction accuracy. This also reinforces that basketball scoring outcomes—at the individual level—tend to follow consistent patterns tied to shot attempts and playing time more than subtle or highly complex statistical interactions.

### 6.2 Team Win/Loss Prediction Results

| Model | Accuracy | Precision | Recall | F1 | ROC-AUC |
|---|---|---|---|---|---|
| Logistic Regression | 0.844 | 0.850 | 0.834 | 0.842 | 0.921 |
| Random Forest | 0.790 | 0.776 | 0.814 | 0.795 | 0.871 |
| Gradient Boosting | 0.806 | 0.797 | 0.821 | 0.809 | 0.892 |

**Observation:** Logistic Regression achieved the best balance of accuracy and interpretability, meaning that the relationship between team statistics and win probability is largely linear and predictable. The model identifies clear patterns: teams with higher shooting efficiency, more rebounds, and fewer turnovers are significantly more likely to win games. Although Random Forest and Gradient Boosting can capture deeper interactions, they did not outperform Logistic Regression by a meaningful margin. This suggests that the key factors driving wins—such as efficiency, ball control, and defensive pressure—have strong and direct effects that do not require complex modeling to uncover. Logistic Regression's strong ROC-AUC score also indicates it consistently distinguishes winning from losing team performances across different game scenarios.

## 6.3 Hyperparameter Tuning and Cross-Validation

To ensure fair model comparison and assess whether tuning improves performance, we implemented 5-fold cross-validation for all models and applied hyperparameter optimization using RandomizedSearchCV with 10 iterations.

#### 6.3.1 Cross-Validation Methodology

**Regression Task (PTS Prediction):** - Used 5-fold KFold cross-validation on training data - Evaluated models using CV_RMSE, CV_MAE, and CV_$R^2$ metrics - Final test set evaluation provided independent performance estimates

**Classification Task (Win/Loss Prediction):** - Used 5-fold StratifiedKFold to maintain class balance across folds - Evaluated models using CV_Accuracy, CV_Precision, CV_Recall, CV_F1, and CV_ROC-AUC - Final test set evaluation assessed real-world generalization

#### 6.3.2 Hyperparameter Tuning Configuration

We tuned ensemble models using RandomizedSearchCV with the following parameter ranges:

**Regression Models:** - **Random Forest**: n_estimators (100-500), max_depth (10-50), min_samples_split (2-20), min_samples_leaf (1-10) - **Gradient Boosting**: n_estimators (100-500), learning_rate (0.05-0.30), max_depth (3-10), subsample (0.7-1.0) - **XGBoost**: n_estimators (100-500), learning_rate (0.05-0.30), max_depth (3-10), subsample (0.7-1.0), colsample_bytree (0.7-1.0) - **LightGBM**: n_estimators (100-500), learning_rate

(0.05-0.30), max_depth (3-12), subsample (0.7-1.0), colsample_bytree (0.7-1.0)

**Classification Models:** - **Logistic Regression**: C (0.01-10), penalty (L1/L2), solver (saga) - **Ensemble models**: Similar parameter ranges as regression

Linear Regression was not tuned as it has minimal hyperparameters and serves as a baseline.

### 6.3.3 Player Points Prediction: Tuning Results

**Cross-Validation Performance:**

| Model | CV RMSE | CV MAE | CV $R^2$ |
|---|---|---|---|
| **Linear Reg** | **2.188** | **1.589** | **0.939** |
| RF (Tuned) | 2.384 | 1.695 | 0.927 |
| RF (Base) | 2.406 | 1.712 | 0.926 |
| GB (Tuned) | 2.253 | 1.624 | 0.935 |
| GB (Base) | 2.282 | 1.651 | 0.933 |
| XGB (Tuned) | 2.290 | 1.640 | 0.933 |
| XGB (Base) | 2.446 | 1.731 | 0.923 |
| LGBM (Tuned) | 2.278 | 1.633 | 0.934 |
| LGBM (Base) | 2.304 | 1.649 | 0.932 |

**Test Set Performance:**

| Model | Test RMSE | Test MAE | Test $R^2$ |
|---|---|---|---|
| **Linear Reg** | **2.174** | **1.588** | **0.939** |
| RF (Tuned) | 2.364 | 1.698 | 0.928 |
| RF (Base) | 2.387 | 1.717 | 0.927 |
| GB (Tuned) | 2.239 | 1.630 | 0.935 |
| GB (Base) | 2.275 | 1.662 | 0.933 |
| XGB (Tuned) | 2.255 | 1.626 | 0.934 |
| XGB (Base) | 2.386 | 1.713 | 0.927 |
| LGBM (Tuned) | 2.250 | 1.625 | 0.935 |
| LGBM (Base) | 2.274 | 1.651 | 0.933 |

**Key Findings (Regression):**

**All tuned models improved over their baselines** when evaluated via cross-validation: - **XGBoost**: Largest improvement (CV RMSE: 2.446 $\rightarrow$ 2.290, **-0.156 reduction**) - **Gradient Boosting**: CV RMSE improved from 2.282 $\rightarrow$ 2.253 (**-0.029**) - **LightGBM**: CV RMSE improved from 2.304 $\rightarrow$ 2.278 (**-0.026**) - **Random Forest**: CV RMSE improved from 2.406 $\rightarrow$ 2.384 (**-0.022**)

**Test set performance aligned with CV results**, confirming models generalize well

**Linear Regression remained the best overall model** (CV RMSE: 2.188, CV $R^2$: 0.939), suggesting the relationship between features and points scored is predominantly linear

### 6.3.4 Team Win/Loss Prediction: Tuning Results

**Cross-Validation Performance:**

| Model | CV Acc | CV F1 | CV AUC |
|---|---|---|---|
| **LogReg (Base)** | **0.862** | **0.862** | **0.934** |
| LogReg (Tuned) | 0.862 | 0.862 | 0.934 |
| RF (Tuned) | 0.788 | 0.789 | 0.873 |
| RF (Base) | 0.782 | 0.779 | 0.875 |
| GB (Tuned) | 0.803 | 0.803 | 0.890 |
| GB (Base) | 0.802 | 0.803 | 0.891 |
| XGB (Tuned) | 0.805 | 0.805 | 0.894 |
| XGB (Base) | 0.805 | 0.808 | 0.889 |
| LGBM (Tuned) | 0.821 | 0.821 | 0.897 |
| LGBM (Base) | 0.810 | 0.810 | 0.895 |

**Test Set Performance:**

| Model | Test Acc | Test F1 | Test AUC |
|---|---|---|---|
| **LogReg (Base)** | **0.844** | **0.842** | 0.921 |
| LogReg (Tuned) | 0.837 | 0.835 | **0.921** |
| RF (Tuned) | 0.783 | 0.785 | 0.866 |
| RF (Base) | 0.790 | 0.795 | 0.871 |
| GB (Tuned) | 0.805 | 0.805 | 0.885 |
| GB (Base) | 0.806 | 0.809 | 0.892 |
| XGB (Tuned) | 0.803 | 0.800 | 0.887 |
| XGB (Base) | 0.811 | 0.808 | 0.881 |
| LGBM (Tuned) | 0.814 | 0.811 | 0.888 |
| LGBM (Base) | 0.805 | 0.805 | 0.881 |

**Key Findings (Classification):**

**Hyperparameter tuning provided minimal benefit for classification**: - **Logistic Regression**: Identical CV scores for tuned vs baseline (0.862 accuracy), indicating default parameters were already optimal - **Random Forest**: Small improvement (CV Acc: 0.782 → 0.788, **+0.006**) - **LightGBM**: Modest improvement (CV Acc: 0.810 → 0.821, **+0.011**) - **Gradient Boosting**: Negligible change (CV Acc: 0.802 → 0.803, **+0.001**) - **XGBoost**: No change in CV Acc (0.805), slight improvement in CV ROC-AUC

**Test set performance sometimes favored baselines**, suggesting tuned models may have slightly overfit to CV folds: - Logistic Regression baseline achieved higher test accuracy (0.844 vs 0.837) - Gradient Boosting baseline had slightly better test metrics

*Abbreviations: RF = Random Forest, GB = Gradient Boosting, XGB = XGBoost, LGBM = LightGBM, LogReg = Logistic Regression, Acc = Accuracy, AUC = ROC-AUC, RMSE = Root Mean Squared Error, MAE = Mean Absolute Error, $R^2$ = Coefficient of Determination*

#### 6.3.5 Analysis and Interpretation

The contrasting results between regression and classification tuning reveal important insights about model selection and hyperparameter optimization:

**Why Tuning Helped Regression:** - Default hyperparameters for ensemble methods are often conservative and may underfit continuous prediction tasks - Point scoring involves a wide range of values (0-60+ points) requiring careful tuning of tree depth, learning rates, and ensemble size - XGBoost's dramatic improvement suggests its default parameters were particularly suboptimal for this task

**Why Tuning Had Minimal Impact on Classification:** - Scikit-learn's default hyperparameters are well-calibrated for binary classification problems - The win/loss decision boundary appears to be relatively linear and well-separated, reducing the need for complex model configurations - With balanced classes (~50/50 win/loss split) and clear separating features (shooting efficiency, rebounds, turnovers), simple models excel

**Overall Conclusion:**

Hyperparameter tuning is a standard best practice in machine learning, but our results demonstrate that **its value is highly task-dependent**. For regression tasks with continuous targets like player points, tuning yielded meaningful improvements across all ensemble methods. However, for well-separated binary classification problems like win/loss prediction, baseline models with default parameters performed nearly as well as extensively tuned alternatives, with Logistic Regression remaining the top performer regardless of tuning.

This finding has practical implications: practitioners should assess whether the time investment in hyperparameter tuning is justified based on the problem structure, baseline performance, and marginal gains achievable through optimization.

---

## 6.4 Model Interpretation

- **PTS Prediction:** FGA had the largest positive weight, followed by 3P and FT. TOV (turnovers) had a small negative impact.
- **Win Prediction:** Positive factors were FG%, rebounds, and steals. Turnovers negatively affected outcomes.

## 7. Model Evaluation and Improvements

- **Scaling:** Standard scaling improved linear models.
- **Regularization:** Will be tested in future versions to reduce overfitting.
- **Hyperparameter tuning:** Next step — use GridSearchCV or boosting frameworks (XGBoost, LightGBM).

---

## 8. Conclusion

We developed a comprehensive machine learning pipeline for NBA analytics with both baseline and hyperparameter-tuned models. Linear Regression achieved the best performance for player point prediction (CV RMSE: 2.188), while Logistic Regression excelled at team win prediction (CV Accuracy: 86.2%). Hyperparameter tuning provided substantial improvements for regression tasks but minimal gains for classification, demonstrating the task-dependent value of optimization. Key predictive features include field goal attempts, three-pointers, shooting efficiency, rebounds, and turnovers. This pipeline provides a robust foundation for NBA performance analysis and can be extended with additional features and models.

---

## Reproducibility Notes

Code modules:

- `data_utils`: data loading, cleaning, and aggregation.
- `training`: modeling and evaluation functions.