Making a Statistical Model for Predicting NBA Game Outcomes

Alex Clare
Maya Hirsch
Henry Johnson
Lior Levy Meruk
Mary McSweeney
Rishabh Sood

March 31, 2025

Abstract

This model attempts to predict the outcome (win or loss) of NBA games for the 2023-2024 season. Our dataset initially had 24 variables, and in order to correctly predict the game outcome we transformed the dataset only keeping 3 of the original variables and adding 26 new ones. Our model attempts to predict the 101st game and beyond using data from the previous games. After performing a 70-30 training testing split, we ran Logistic Regression, LDA, KNN, Random Forest, Gradient Boosting, and SVM with a variety of hyper-parameters to find KNN as our best model with a testing accuracy of 66%.

Table of Contents

1 Introduction

Predicting NBA game outcomes is a complex task that involves analyzing statistical data, historical performance, and various team dynamics. The dataset provided for this project includes a wide array of game statistics such as points, rebounds, assists, field goals made, and 3-point shooting metrics, alongside meta-information like home/away game indicators and cumulative team performance metrics.

This project aims to leverage significant predictors from the dataset to predict the outcomes of NBA games. Engineered features such as win percentages, weighted averages, and opponent-specific statistics were included to enhance the predictive model. Key contextual factors, such as home-court advantage and team stability, were also considered essential elements in the analysis.

The predictive model was constructed using a Random Forest classifier, which is particularly well-suited for capturing non-linear relationships and interactions among predictors. Feature selection was performed using Ridge and Lasso regression to identify significant predictors, ensuring the model focused on the most impactful variables. To evaluate model performance, K-fold cross-validation with K = 1 - 31 was employed to improve accuracy and minimize model variability. Optimal accuracy was found to be when K = 15. as well as test-train split of 70-30 with a variation or hyper-parameters.

After training and testing on data after the first 100 games, the model successfully predicted the outcomes of games scheduled for the season, offering valuable insights into the critical factors influencing NBA game outcomes.

2 Data Processing

For this project, we transformed the dataset to improve its usability for model building. New columns were constructed to ensure that each row contained only information from previous games. Numeric columns such as PTS (points), FGM (field goals made), and BLK (blocks) were recalculated into various statistical measures, including cumulative averages, weighted averages, total values, differences between opposing teams, and variances. These transformations helped capture more meaningful insights from the data, ensuring it was well-suited for the subsequent stages of analysis and model development.

Data Formatting and Initial Setup
In the initial stages of data preprocessing, several foundational transformations were applied to ensure the dataset was ready for analysis. The FT percent column, originally stored as a string, was converted into a numeric format to enable statistical computations. The Game Date column was standardized into a proper date format, allowing for time-based analyses and filtering. To better represent game-specific details, new columns were introduced. A Home Team column was created to indicate the team playing on their home court, while an Away Team column identified the visiting team. Additionally, an Opponent column was added to explicitly capture the oppos-

ing team for each game. These adjustments ensured the data was structured consistently and effectively for further feature engineering and modeling.

Creating New Features
To begin the preprocessing steps for model construction, a new data frame was created by excluding the first 100 rows of the original dataset. These rows were used to generate important features such as cumulative averages and differences, ensuring that each row represented the state of a team before the current game. This new data frame served as the foundation for the subsequent feature engineering process, providing a structure that could accommodate both team-specific and opponent-specific statistics.

Historical Team Performance Features needs to be proofed Several key features were created to capture the historical performance of each team. First, the total wins column was added to track the number of wins each team had accumulated before the current game. This was achieved by looping through the data and counting the number of victories for each team in the previous games. Similarly, the total games column was added to track the number of games played by each team up until that point. These columns were used to calculate additional features like win percentage, which represents the ratio of wins to total games played, offering an insight into a team's overall performance before each game.

Column Name	Description
win_percentage	Proportion of wins to total games played before the current game.
home_adv	Indicates if the team is playing at home (1) or away (0).
<stat>_diff	Difference between the team's and opponent's average statistics for <stat>.
win_pct_vs_opponent	Win percentage of the team against the specific opponent.
win_pct_diff	Difference between team and opponent.
win_streak	Longest winning streak of the team up to the current game.
recent_win_pct	Win percentage of the team in their last 3-5 games.
weighted_avg_<stat>	Weighted average of a specific statistic (<stat>), with weights inversely proportional to the days since each game.
Win	Indicates whether the team won (1) or lost (0) the current game.

Table 1: Summary of New Columns Added

Historical Team Performance Features
Several key features were created to capture the historical performance of each team. First, the total wins column was added to track the number of wins each team had accumulated before the current game. This was achieved by looping through the data and counting the number of

victories for each team in the previous games. Similarly, the total games column was added to track the number of games played by each team up until that point. These columns were used to calculate additional features like win percentage, which represents the ratio of wins to total games played, offering an insight into a team's overall performance before each game.

Opponent-Specific and Contextual Features
To capture the dynamics of each game in relation to the opponent, several features were engineered. The opponent average columns were created to represent the average statistics for the opponent up until the current game. This allowed the model to consider not only a team's performance but also how their opponent performed historically. Additionally, the win percent vs opponent column was calculated to reflect the win percentage of a team specifically against their upcoming opponent, which could offer a more nuanced prediction of the game outcome based on historical head-to-head performance.

Advanced Statistical Features
In addition to basic performance metrics, more advanced features were created to capture variance and trends in performance. The total plus/minus column sums the plus/minus values from previous games, reflecting the overall performance impact of a team. Variance in-game statistics were also computed for each team, providing insight into how consistent a team's performance had been over the season. Additionally, recent win percent was calculated to measure a team's performance in their last few games, offering a glimpse into their current form. Finally, weighted averages of past performances were computed, with more recent games carrying greater importance, to give the model more recent context for predictions.

Outcome and Momentum Features
To help the model predict game outcomes, several features related to momentum and team performance were created. A win streak column was added to track the longest consecutive wins for each team, providing a measure of team momentum going into a game. The home adv column was also created to indicate whether a team was playing at home (1) or away (0), as home-court advantage can play a significant role in outcomes. These features, combined with all the previous performance data, helped provide a complete picture of each team's situation before the game.

3 Experiment Setup

After feature engineering, we looked into picking a subset of features to employ in building our predictive models. We began by assessing the correlations between our predictors, finding that most variables had low correlation values (0.1 and lower). Below is a correlation matrix demonstrating the predictions most strongly connected with. The predictors don't have a strong correlation with the outcome variable. There were no major indicators of multicollinearity between variables other than for +/- diff and win pct diff columns.

Due to high correlation between 2 features, we got rid of +/-_diff and win_pct_diff columns.

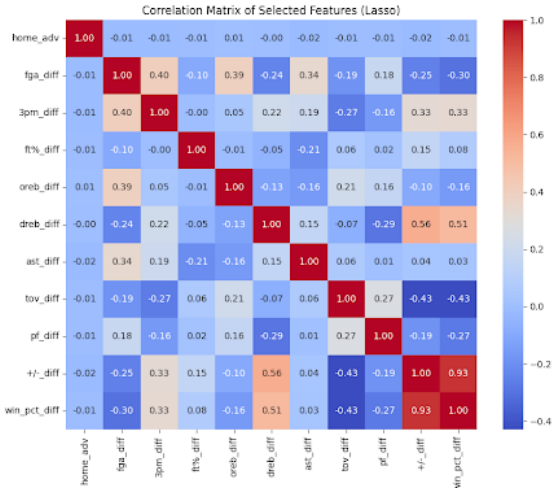

Figure 1: Correlation Matrix

This corroborates our VIF analysis.

We utilized Lasso Regression to improve our model. Lasso regression shrinks coefficients and selects relevant features in linear models using L1 regularization to reduce overfitting. It improves model performance and interpretability by eliminating irrelevant variables. According to Lasso, the features that should be kept include home adv, fga diff, 3pm diff, ft% diff, oreb diff, dreb diff, ast diff, tov diff, pf diff, +/- diff, win pct diff. We then broke up the data into a 70-30 train test split to see how our model performs on new unseen data. Finally we tested our data set under several different models (Logistic Regression, LDA, Random forest, Gradient Boosting, SVM, and KNN) to see how well they predict game outcomes.

4 Methods

Logistic Regression

Logistic regression is a widely used statistical method for binary classification tasks. It is a discriminative model that estimates the conditional probability of a sample belonging to class 1. In comparison to linear regression, which predicts continuous outcomes, logistic regression employs the logistic function to model the connection between input data and the likelihood of the target class. We used logistic regression with the following hyper-parameters:

- Penalty (["l2"]): Specifies the type of regularization to prevent overfitting, with L2 regularization penalizing large coefficients.
- Regularization Strength (C = [0.1, 1, 10]): Controls the balance between bias and variance, with smaller values increasing regularization and larger values reducing it.
- Solver (["lbfgs"]): Optimization algorithm used for training, chosen for its efficiency and compatibility with L2 regularization.

Support Vector Machine (SVM)
Support Vector Machine (SVM) is a supervised machine learning algorithm primarily used for classification and regression tasks. It is particularly effective in high-dimensional spaces and for problems with a clear margin of separation. We used SVM to train and test our model with the following hyper-parameters.

- Regularization (C = [0.1, 1, 10]): Balances the simplicity of the decision boundary with the fit to training data.
- Kernel Coefficient (gamma = ["scale", "auto"]): Adjusts the influence of individual data points in the kernel function.
- Kernel Type (kernel = ["rbf"]): Specifies the method for non-linear data transformation, with rbf chosen for its flexibility.

Random Forest
Random Forest is a supervised machine learning algorithm used for both classification and regression tasks. It is an ensemble learning method that combines multiple decision trees to create a robust, accurate, and versatile model. We used Random Forest with the following hyper-parameters:

- Number of Trees (n_estimators = [50, 100, 200]): Defines the number of decision trees, balancing accuracy and training efficiency.
- Tree Depth (max_depth = [None, 10, 20]): Limits the complexity of each tree, preventing overfitting.
- Minimum Samples to Split (min_samples_split = [2, 5]): Controls the minimum number of samples required to split a node, affecting tree depth and generalization.

K-Nearest Neighbors (KNN)
K-Nearest Neighbors (KNN) is simple and intuitive for classification and regression. It finds the K nearest data points (neighbors) to an input instance. The approach assigns the most common class label among these K neighbors, implementing a majority vote. KNN is a nonparametric model, which memorizes the training dataset and predicts during query time. The simplicity and effectiveness of KNN made it an appealing choice for us.

- Number of Neighbors (K = 1 through 31): Determines how many neighbors are considered for predictions, balancing sensitivity to local patterns and generalization.

Gradient Boosting
Gradient Boosting is a machine learning technique for building predictive models. It is an ensemble learning method that combines the outputs of multiple "weak learners," typically decision trees, to produce a robust and accurate prediction. It uses the principles of boosting, where models are trained sequentially, and each subsequent model tries to correct the errors of the previous ones.

- Number of Boosting Stages (n_estimators = [50, 100, 200]): Determines the number of iterations to improve accuracy and reduce residual errors.
- Learning Rate (learning_rate = [0.01, 0.1, 0.2]): Controls the contribution of each stage to the final model, balancing convergence speed and generalization.
- Maximum Tree Depth (max_depth = [3, 5, 10]): Limits the complexity of individual trees to prevent overfitting.

Linear Discriminant Analysis (LDA)
Linear Discriminant Analysis (LDA) is a supervised machine learning technique used primarily for:

- Dimensionality Reduction: Reducing the number of features while preserving as much class discrimination information as possible.
- Classification: Creating a linear decision boundary to separate different classes.

LDA works by projecting high-dimensional data onto a lower-dimensional space, maximizing the separation between multiple classes by finding a linear combination of features that best separates the classes. Linear Discriminant Analysis (LDA) typically does not require hyperparameter tuning because of its inherent simplicity and underlying assumptions.

5 Results and Analysis

The mean accuracy scores from the different models are given below:

Model	**Best Parameters**	**Test Accuracy**
Logistic Regression	C: 0.1, Penalty: l2, solver: lbfgs	64.69%
LDA	N/A	64.97%
Random Forest	Max Depth: 20, min_samples_split: 2, Estimators: 50	62.43%
Gradient Boosting	Learning Rate: 0.1, Max Depth: 3, Estimators: 50	65.54%
SVM	C: 0.1, Gamma: scale, Kernel: rbf	65.25%
KNN	K: 15	66.3%

Table 2: Hyperparameter Tuning Results and Test Accuracy for Models

The results in Table 2 show the performance of different machine learning models after hyperparameter tuning to predict NBA game outcomes (win or loss). Among the models, K-Nearest Neighbors (KNN) achieved the highest test accuracy of 66.3% with K = 15, making it the top-performing model for this task. This suggests that KNN, which bases predictions on the outcomes of similar past games, effectively captures patterns in the data to determine the likelihood of a win or loss. Gradient Boosting closely followed with an accuracy of 65.54%, leveraging a learning rate of 0.1, a maximum tree depth of 3, and 50 estimators. Gradient Boosting's ability to iteratively

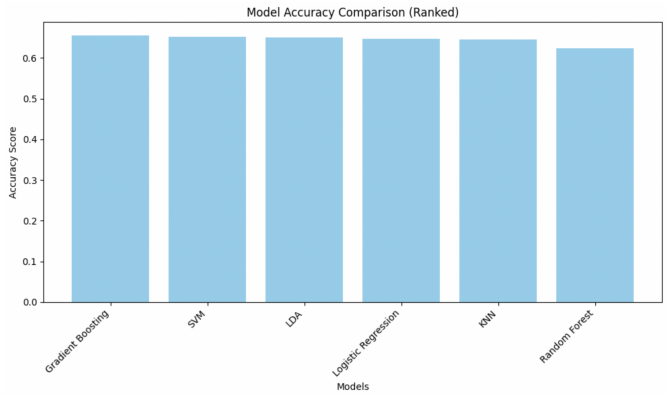

Figure 2: Model Accuracy Comparison

refine predictions might be particularly useful in identifying subtle game dynamics. The Support Vector Machine (SVM) model also performed well, achieving an accuracy of 65.25% with the rbf kernel, which maps game features into higher dimensions to better separate winning and losing patterns. Logistic Regression (C = 0.1, L2 penalty) and Linear Discriminant Analysis (LDA) delivered similar results, with accuracies of 64.69% and 64.97%, respectively, providing simple yet reliable predictions based on a linear understanding of game features.

On the other hand, the Random Forest model had the lowest accuracy of 62.43%, even with optimized parameters such as a maximum tree depth of 20, 50 estimators, and a minimum sample split of 2. This lower performance suggests that Random Forest might struggle with this specific dataset, potentially due to overfitting or difficulty capturing relationships between game features. These results emphasize the value of selecting and tuning models thoughtfully for predicting NBA game outcomes. Simpler models like KNN and Gradient Boosting, when well-optimized, can effectively analyze patterns in historical game data to provide actionable predictions, such as assessing a team's likelihood of winning based on performance metrics. This insight can be valuable for teams, analysts, or enthusiasts aiming to evaluate game outcomes or refine strategies based on predictive analysis.