Predicting NBA Game Outcomes Using Machine Learning

Kyle Beedon

May 2025

Contents

1 Abstract

The goal of this project is to predict the outcome of NBA games with an accuracy that exceeds that of picking the favored team. Using Python, I implemented several machine learning classification models, including Logistic Regression, Support Vector Machine, Random Forest, and XGBoost. These models were trained on a dataset that includes aggregates of box score statistics, game context (such as whether a team is playing at home or on a back-to-back), and engineered features from the 2021 through 2024 NBA seasons. The 2025 season was held out as a final test set to evaluate model performance. The best-performing model was the Support Vector Machine Classifier, which achieved an accuracy of 69.6% on the 2025 dataset. All code used in this project is available on my GitHub at https://github.com/KyleBeedon/Predicting-NBA-Game-Outcomes-Using-Machine-Learning

2 Introduction

The result of a National Basketball Association (NBA) game is simple: whichever team scores more points wins. The NBA is made up of 30 teams—15 in each conference—and each team plays an 82-game regular season. Every game has equal influence on a team's chances of making the playoffs. Only the top six teams in each conference are guaranteed a playoff spot, while teams ranked 7 through 10 compete in the play-in tournament for a chance to qualify. The remaining five teams are eliminated, so the outcome of each game can carry significant weight.

The NBA has a massive fan base and is a multi-billion-dollar business. When you consider TV deals, ticket sales, merchandise and other revenue streams, the league generates roughly $10 billion annually. Another major source of income is sports betting. For individual fans placing bets on games, having a reliable prediction of the outcome would be highly valuable. The goal of this project is not to give bettors an advantage, but to explore how the data behind teams and games influence winning.

The objective of this project is to predict the outcome of NBA games. In Section 3, we will begin with a review of the literature on similar projects, exploring the methods they used and the results they achieved. Section 4 will provide a detailed discussion of the data, including where it was sourced, how it was cleaned, what variables were used, and the reasoning behind

those choices. In Section 5, we will dive into the methodology, covering data processing, the modeling techniques applied, and other relevant details. Section 6 will present the results: highlighting the best-performing model, evaluating its accuracy on the 2025 NBA season, and analyzing predictions for individual teams. Finally, Section 7 will conclude with a reflection on what went well, what could be improved, and ideas for future work.

3 Literature Review

The problem of predicting NBA game outcomes has been explored before, particularly by college students working on senior theses or practicum projects, like myself. Among these efforts, there is some overlap in the modeling techniques used, and a few core approaches tend to remain consistent across different projects.

Josh Weiner, in a group project at the University of Pennsylvania titled, **"Predicting the Outcome of NBA Games with Machine Learning,"** set out to do exactly what I'm aiming to accomplish with this project. They scraped team and player box score data from 12 NBA seasons. For teams, they built an Elo variable that carried trends across seasons and updated after each game. They aggregated team and player stats over the previous 10 games, including player efficiency rating (PER). After compiling the data, they used train-test splits and various models. Their logistic regression model reached 66.3% accuracy, while their random forest model performed slightly better at 67%.

Junwen Wang's project, **"Predictive Analysis of NBA Game Outcomes through Machine Learning"**, compiled data from 16 years of NBA games. Unlike the previously mentioned project that focused on predicting future games, this one aimed to better understand what factors contribute to winning a game. The models used—Logistic Regression, SVM, DNN, and Random Forest—achieved notably higher accuracies than others in the literature, largely because they used in-game statistics to predict the outcome of that same game. It was a fascinating approach because it highlighted which stats—like shooting percentages and rebounding—had the greatest impact on winning. The Logistic Regression model alone reached an accuracy of 83.8%. This helped shape my thinking around feature selection for my own project.

Luke DiPerna's project, **"NBA Prediction Modeling"**, offered valu-

able insights into using aggregated stats from previous games. He collected data from 12,000 NBA games and built models using mean team statistics from the last 10, 20, and 30 games. He found that model performance improved slightly as more games were included. Luke tested several models—Logistic Regression, K-Nearest Neighbors, Random Forest, Gaussian Naive Bayes, SVC, and Neural Networks—with Gaussian Naive Bayes performing the best. He also developed his own Elo rating system, drawing inspiration from FiveThirtyEight's approach. One of his models simply picked the team with the higher Elo rating, achieving an accuracy of 67%.

Data science and sports intersect in so many exciting ways—whether it's boosting team performance, breaking down the game's dynamics, or finding new ways to drive revenue. The projects I explored were not only interesting reads but also shaped how I approached my own work. You'll find links to them in the appendix.

4 Discussion of Data

Accurately predicting the outcome of an NBA game requires a large amount of reliable data. While some previous projects incorporated individual player statistics, my approach focuses entirely on team stats and engineered features.

I compiled my dataset by scraping box score data from **Basketball-Reference.com** using a Python web scraping method, collecting team statistics for every season since 2020. Initially, I scraped teams in alphabetical order, but after noticing an imbalance in the models, I switched to randomly scraping teams to eliminate any potential bias. The raw data was then stored as a CSV, then imported into R, where I used the dplyr package for data manipulation.

All statistics were converted into rolling averages over the last 25 games. For example, to predict a team's 30th game, I used their averages from games 5–29 along with their opponent's. I tested aggregations of 10, 15, 20, 25, and 30 games, and 25 produced the highest model accuracy. Performance improved with more data but dropped slightly at 30, making 25 the optimal choice. A deeper analysis of model performance is included in the Anlysis of Results section.

As a college basketball player, I'm especially interested in understanding which statistics most influence winning and how they work together to

provide a complete picture of the game. To do this, I grouped metrics into key "buckets" that reflect core areas of basketball, categorized under either offense or defense.

On offense, the first area I focused on was shooting. Instead of relying on basic 2P% or 3P%, I used effective field goal percentage **(eFG%)**, which accounts for the added value of three-pointers and provides a more accurate measure of shooting efficiency. I also included free throw percentage **(FT%)**, since it isn't captured by eFG%.

Next was ball security, measured by turnover percentage **(TOV%)**, which shows how often a team turns the ball over per possession—a critical factor in maximizing scoring opportunities.

Rebounding was another key category. I used offensive rebound percentage **(ORB%)** and defensive rebound percentage **(DRB%)**, which indicate how often a team secures rebounds on their own or their opponent's missed shots.

Defensively, I mirrored these metrics—eFG%, TOV%, ORB%, and DRB%, but from the opponent's perspective. These appear as variables with an **opp_ prefix** (e.g., opp_eFG%) and were calculated using opponents' averages over their last 25 games.

To capture overall team strength over the last 25 games, I used each team's **Net Rating**, which measures point differential per 100 possessions—a quick indicator of dominance. While many projects in the literature, such as Josh Weiner's, relied on Elo ratings, I created my own version of Elo but found it underperformed. As a result, I chose to rely solely on Net Rating for this project.

Along with the box score stats, I added two game context features: a **home variable** (1 if the team is at home, 0 if away) and a **back-to-back indicator** (1 if the team is playing on consecutive days, 0 otherwise). Both are strongly linked to winning. As in most sports, home-court advantage plays a major role—a point we'll revisit in the next section. Similarly, playing on a back-to-back often signals fatigue, making it a key factor in game outcomes.

After collecting and validating all features—by comparing aggregates to official Basketball-Reference stats—I joined the data on gameId, as each game appeared twice (once for each team). The first team in the dataset for a given game was labeled with the .x suffix, and the second with .y. An inner join on gameId combined the records into a single row per game. The target variable was result.x, meaning the model predicts the outcome from

the perspective of the .x team using data from both teams.

The final dataset was saved as a CSV and used for modeling in a Jupyter Notebook.

5 Methodology

After the dataset was properly cleaned and processed, I loaded the final CSV into a Jupyter Notebook file. I first checked for any null values (there weren't any), and then began exploring trends through a thorough data analysis.

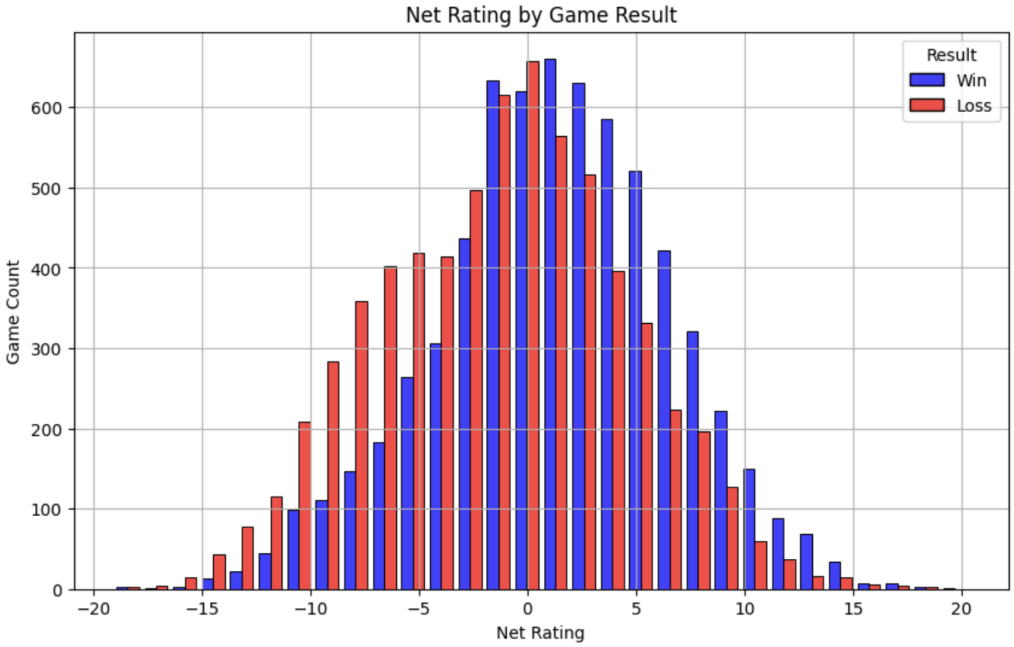
a) Exploratory Data Analysis

As I began exploring the dataset, I was particularly interested in which variables had the strongest relationship with winning. One metric that stood out was Net Rating, as it directly reflects point differential per 100 possessions. To visualize this, I created a histogram showing the distribution of Net Rating for wins versus losses, with the x-axis representing each team's Net Rating values averaged over the previous 25 games, and the y-axis representing the number of games. Wins and losses were color-coded, making the separation between the two distributions clear.

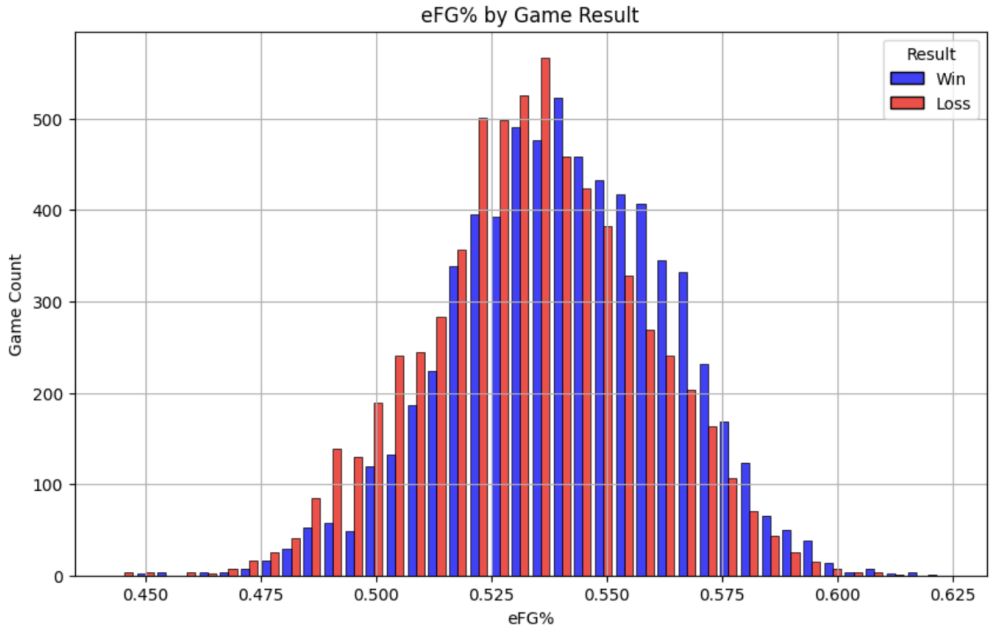
I performed a similar analysis for eFG%, given its importance in shooting efficiency. Again, the histogram plots eFG% on the x-axis and game count on the y-axis, with separate distributions for wins and losses. Both plots clearly, which can be seen in the following page, demonstrate that higher Net Ratings and higher eFG% are strongly associated with winning.

After observing the plots on the following page, it is clear that the distributions of Net Rating for both wins and losses are approximately normal, but the outcomes labeled as wins are shifted to the right. The eFG% plot shows a similar pattern: teams with lower eFG% tend to lose more games. While the distributions in this plot also appear roughly normal, they exhibit greater skewness than the Net Rating plot. In summary, these plots suggest that if a team has a higher point differential over the last 25 games and has been shooting well recently, they are more likely to be rewarded with a win.

I explored all of the other variables' distributions, using the same format, but they had less impressive results, the other plots can be found in the appendix.



(a) Distribution of Net Rating for wins vs. losses.



(b) Distribution of eFG% for wins vs. losses.

In addition to the box score statistics, I was also interested in examining the strength of two variables: home and back-to-back. I created two tables, shown below, that calculate the win rate for home teams versus away teams, and for teams playing on a back-to-back versus those that were not. The two tables showcase pretty obvious advantages.

In this dataset, the home team wins 55.29% of the time, which is a substantial advantage—just imagining that always picking the home team would give a better-than-even chance of winning. Similarly, teams playing on a back-to-back schedule lose 44.83% of their games. If a team is on a back-to-back and playing on the road, it would be reasonably safe to predict a loss. Of course, other factors, such as team quality, also play a role.

Overall, home teams win a much higher percentage of games, and teams on a back-to-back lose more frequently. Combined with what we observed from the eFG% and Net Rating plots, these four variables are likely to be key indicators in the modeling process.

Game Outcomes by Back-to-Back Status

	Losses	Wins	Total Games	Win Rate (%)
Not Back-to-Back	5318	5559	10877	51.11
Back-to-Back	1287	1046	2333	44.83
Total	6605	6605	13210	NaN

Game Outcomes by Home Status

	Losses	Wins	Total Games	Win Rate (%)
Not Home	3652	2953	6605	44.71
Home	2953	3652	6605	55.29
Total	6605	6605	13210	NaN

Figure 2: Win/Loss Tables for Home and Back-to-Back Games

A common issue in statistical modeling is highly correlated features. I generated a correlation matrix to check this, which shows the correlation between every pair of features. A feature is always perfectly correlated with itself, correlations of 0 indicate no relationship, and negative values indicate inverse relationships. In this dataset, all correlations were below 0.6, so overcorrelation was not a concern.

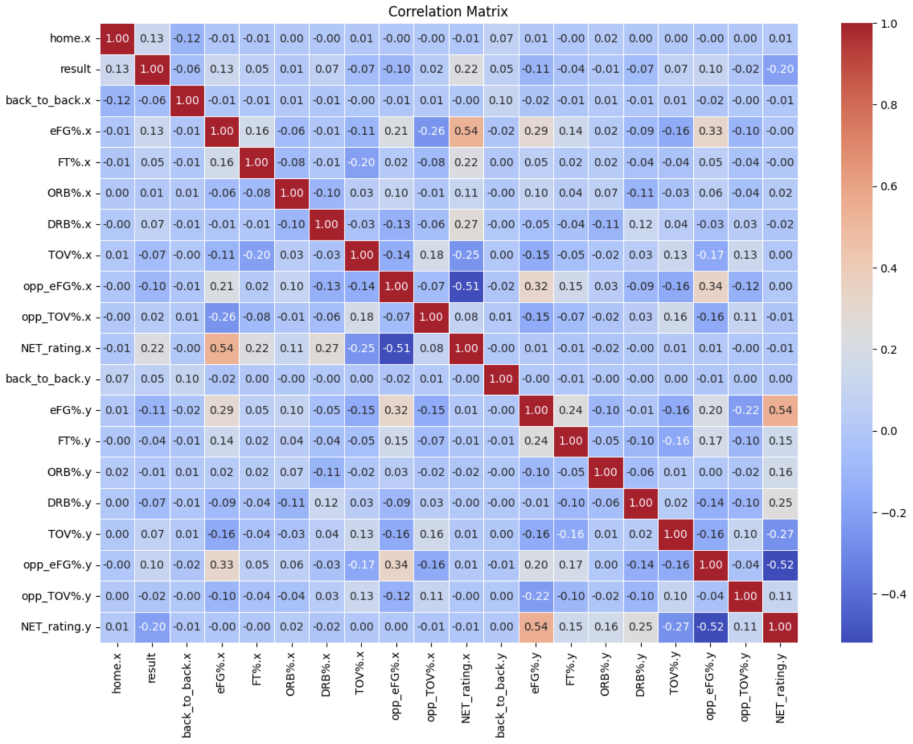


Figure 3: Correlation Matrix

With the data exploration complete, we can now move on to preparing the data for modeling.

b) Data Preparation

My first consideration in preparing the data was how to handle the start of each season, or more specifically, the lack of data available in the aggregations. I explored different approaches and identified two main options. The first was to use the previous season's total averages as placeholders for the

first game, then transition to season averages. The second was to truncate a few games at the start of each season to allow some data on the team's current performance to accumulate. I chose the second approach because NBA team performance can vary significantly from year to year due to factors such as new coaches, top draft picks, major free-agent signings, injuries, or a team's overall collapse.

In addition to these logical reasons, truncating the first five games of each season slightly improved overall model accuracy, which makes sense. By the fifth game, the dataset includes aggregations from the first four games—still a small sample, but more meaningful than relying on potentially misleading placeholder data.

To prepare for modeling, I created a separate series for the target variable (result) while retaining the full dataframe of features. I then removed all games from the 2025 season and stored them in a separate dataframe for final testing.

A key consideration was how to split the data into training and test sets. Standard practice in machine learning is to randomly assign 70% of the data for training and 30% for testing. I initially followed this approach but later decided to preserve chronological order, given the timing of NBA games. Season progression can influence performance—teams struggling midseason may lose intentionally for draft positioning. To capture this dynamic, I used the first 70% of games for training and the remaining 30% for testing, a common practice in time-series modeling.

c) Machine Learning Models

With the dataset ready, the next step was choosing models to predict the target variable: result (win = 1, loss = 0). Since this is a binary classification problem, classification algorithms were the obvious choice. I focused on four well-established models that strike a balance between interpretability, performance, and robustness.

Logistic Regression was the natural starting point. Its simplicity and transparency make it a common baseline for classification problems. Logistic Regression estimates the probability that an observation belongs to a given class—in this case, whether a team will win. If the probability exceeds a set threshold (usually 0.5), the model predicts a win (1); otherwise, a loss (0). Beyond being straightforward, it's useful for understanding how individual features influence the outcome.

Support Vector Machine (SVM) Classifier was included for its strength in finding optimal decision boundaries. SVM identifies the hyperplane that best separates the two classes while maximizing the margin between them. This approach performs well in high-dimensional spaces and can handle non-linear patterns through kernel functions.

Random Forest Classifier adds an ensemble approach by building multiple decision trees and combining their predictions. Each tree trains on a random subset of the data and features, reducing overfitting and improving generalization. Random Forests are robust, handle non-linear relationships effectively, and provide insights into feature importance.

XGBoost Classifier rounds out the selection. Renowned for its strong performance on structured data, XGBoost uses gradient boosting to iteratively improve weak learners (decision trees) by minimizing errors at each step. It's highly optimized, deals well with missing values, and consistently delivers top-tier accuracy.

With the models chosen, the next step was fine-tuning their hyperparameters to maximize performance.

d) Time Series Cross-Validation and Hyperparameter Tuning

After selecting the models, the next step was improving their performance through hyperparameter tuning. To ensure realistic evaluation and avoid data leakage, I used **time series cross-validation** instead of standard k-fold. Unlike traditional CV, which shuffles the data into random folds, time series CV respects the chronological order by training on past data and validating on future slices, mimicking real-world forecasting conditions.

Tuning was performed using parameter grids that define possible values for each hyperparameter. Depending on the model and grid size, I used two approaches: **GridSearchCV**, which exhaustively tests all combinations, and **RandomizedSearchCV**, which samples a fixed number of combinations for faster results.

For Logistic Regression, the grid was small enough for a full search. I tuned the penalty type—L1 (lasso), L2 (ridge), and ElasticNet—along with the C value, which controls the trade-off between bias and variance, and the solver used for optimization. Similarly, for the SVM classifier, I tested different values of C, various kernel types (linear, polynomial, RBF, sigmoid),

and gamma, which determines how much influence individual training points have on the decision boundary.

The parameter spaces for Random Forest and XGBoost were much larger, so RandomizedSearchCV was the practical choice. For Random Forest, I tuned parameters such as the number of estimators, maximum tree depth, and minimum samples required to split or remain at a node. For XGBoost, I focused on similar tree-based parameters but also included boosting-specific ones like learning rate and regularization terms (L1 and L2) to control complexity and reduce overfitting.

After selecting the best configurations through cross-validation, each model was retrained on the full training set and then evaluated on the test set and the 2025 season for final validation.

6 Analysis of Results

For all four models, the 2025 dataset achieved higher accuracy than the test set-which is a favorable outcome. While the difference between the accuracies of the two datasets was noticeable, it was not excessively large.

Model	Test Accuracy	2025 Accuracy
Logistic Regression	0.646	0.685
Support Vector Machine	0.637	0.696
Random Forest	0.617	0.648
XGBoost	0.635	0.660

Figure 4: Model Accuracies

The SVM model performed the best, achieving a very satisfying 69.6% accuracy on the 2025 dataset, while Logistic Regression also performed well with 68.5% accuracy. The other two models achieved acceptable results, especially considering that they were fine-tuned without a complete grid search. We will now take a closer look at the Logistic Regression and SVM models to better understand their results,

Before diving into the results, let's quickly review the key metrics used to evaluate classification models. In this context, each prediction can fall into one of four categories:

- **True Positive (TP)**: Correctly predicts a win.
- **True Negative (TN)**: Correctly predicts a loss.

- **False Positive (FP)**: Predicts a win, but the result was a loss.
- **False Negative (FN)**: Predicts a loss, but the result was a win.

These represent all the possible outcomes in a classification model, and from them, we derive the key performance metrics:

- **Accuracy**: (TP + TN) / (TP + TN + FP + FN), the proportion of correct predictions overall.
- **Precision**: TP / (TP + FP), how many predicted wins were actually wins.
- **Recall**: TP / (TP + FN), how many actual wins the model correctly identified.
- **F1-score**: 2 * (precision * recall) / (precision + recall), a balance between precision and recall.

These metrics are easy to generate in Python using a classification report. We'll look at them for both the Logistic Regression and SVM models.

Logistic Regression

As mentioned earlier, the Logistic Regression model achieved an accuracy of 68.5% on the 2025 NBA season. To put that in perspective, it's slightly better than simply betting on the favorite every game and far better than flipping a coin.

Looking at the classification report, we can dig deeper into how the model performed for both classes—wins (1) and losses (0):

```
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.70      0.70       608
           1       0.67      0.67      0.67       546

    accuracy                           0.68      1154
   macro avg       0.68      0.68      0.68      1154
weighted avg       0.68      0.68      0.68      1154
```

Figure 5: Logistic Regression Classification Report

- For losses (0), the precision, recall, and F1-score are all about 0.70, meaning the model correctly identified losses at a slightly higher rate than wins.

- For wins (1), precision and recall are both around 0.67, which is still solid but slightly lower than for losses.

The macro average across all metrics (precision, recall, F1) is about 0.68, which shows the model is reasonably balanced between predicting wins and losses. There's no extreme bias toward one class, and the performance differences between predicting wins and losses are minor.

Overall, the model doesn't show any glaring weaknesses—it's fairly consistent and performs about as expected for a baseline classification model.

Another important aspect to consider is **feature importance**, which helps explain why the model makes certain predictions. By looking at the coefficients from the Logistic Regression model, we can see which factors had the greatest influence on predicting wins and losses.

Best feature coefficients for the Logistic Regression Model

	Feature	Coefficient	Abs_Coefficient
18	NET_rating.y	-0.336418	0.336418
0	home.x	0.248862	0.248862
9	NET_rating.x	0.225007	0.225007
2	eFG%.x	0.212992	0.212992
1	back_to_back.x	-0.114964	0.114964
6	TOV%.x	-0.111321	0.111321
5	DRB%.x	0.102875	0.102875
15	TOV%.y	0.098938	0.098938
11	eFG%.y	-0.097269	0.097269
7	opp_eFG%.x	-0.076614	0.076614
8	opp_TOV%.x	0.074735	0.074735
14	DRB%.y	-0.067589	0.067589
10	back_to_back.y	0.060167	0.060167
13	ORB%.y	0.055897	0.055897
12	FT%.y	-0.052888	0.052888
16	opp_eFG%.y	0.042796	0.042796
17	opp_TOV%.y	-0.031225	0.031225
3	FT%.x	0.008601	0.008601
4	ORB%.x	0.000423	0.000423

Figure 6: Feature Importance of Logistic Regression Model

Unsurprisingly, Net Rating for both teams stands out as the most influential factor. A higher Net Rating for the target team increases the probability of winning, while a higher opponent Net Rating decreases it. The home-court advantage (home dummy variable) is the second most impactful feature, which aligns with the long-standing understanding that teams generally perform better at home.

Metrics like effective field goal percentage (eFG%) and turnover percentage (TOV%) also carry significant weight. These results make intuitive sense: teams that shoot efficiently and limit turnovers have a better chance of winning. The back-to-back game indicator negatively affects win probability, reinforcing the idea that fatigue matters.

When we interpret these coefficients, the story becomes clear: the fundamentals drive wins, but the margins matter. For instance, you can't just tell a team, "Make more shots" or "Have a better Net Rating"—those are obvious. But smaller factors, like reducing turnovers or grabbing more defensive rebounds, can make a real difference. In short, the game is often decided by possession battles and efficiency in key moments.

SVM Classifier

The SVM model achieved the highest overall accuracy among all models, with an impressive 69.6% accuracy. At the start of this project, I wasn't sure I would ever reach a number this high, so seeing this result was extremely satisfying.

Classification Report:

	precision	recall	f1-score	support
0	0.70	0.74	0.72	608
1	0.69	0.65	0.67	546
accuracy			0.70	1154
macro avg	0.70	0.69	0.69	1154
weighted avg	0.70	0.70	0.70	1154

Figure 7: SVM Classification Report

When examining the classification metrics, we observe a similar pattern to the logistic regression model: the SVM performs slightly better at predicting losses than wins. Specifically, precision for losses was marginally higher,

but the more notable difference came in recall, where the model achieved 0.74 for losses compared to 0.65 for wins. This suggests the model was more effective at correctly identifying losses.

Understanding which features influence the SVM model is crucial, but unlike linear models (such as logistic regression), non-linear SVMs do not produce straightforward coefficients because they operate in a transformed feature space. In this case, the hyperparameter tuning selected a sigmoid kernel, which is a function that helps the model capture complex, non-linear relationships between features. In simple terms, the sigmoid kernel allows the model to act a bit like a neural network, detecting patterns that a straight-line approach would miss.

Since we can't extract coefficients directly from a non-linear kernel, we used permutation importance to measure feature impact. This technique works by shuffling the values of each feature and observing the resulting drop in model performance—features that cause the largest decrease in accuracy when permuted are considered the most important.

	Feature	Importance	Std
0	home.x	0.013142	0.005961
9	NET_rating.x	0.011491	0.007859
18	NET_rating.y	0.007885	0.006423
1	back_to_back.x	0.006683	0.004620
7	opp_eFG%.x	0.000346	0.003442
13	ORB%.y	0.000061	0.001896
17	opp_TOV%.y	-0.000204	0.003170
5	DRB%.x	-0.000244	0.004662
3	FT%.x	-0.000407	0.001954
2	eFG%.x	-0.000937	0.005877

Figure 8: Feature Importance of SVM Classification Model

The top three features were consistent with the previous model, though home.x claimed the top position this time. Back-to-back games remained an influential factor, and interestingly, opponent-based statistics appeared higher on the list compared to the logistic regression model. For instance, opp_eFG% (opponent effective field goal percentage) ranked as the fifth most

important feature, highlighting the importance of defensive performance. In other words, limiting your opponent's shooting efficiency strongly correlates with winning games.

I wanted to dive deeper into team-level performance, so I created a plot showing each team's individual accuracy. In the chart below, the y-axis represents the model's accuracy for a given team, while the x-axis represents the number of games that team won. To make the trend clearer, I added a reference line.

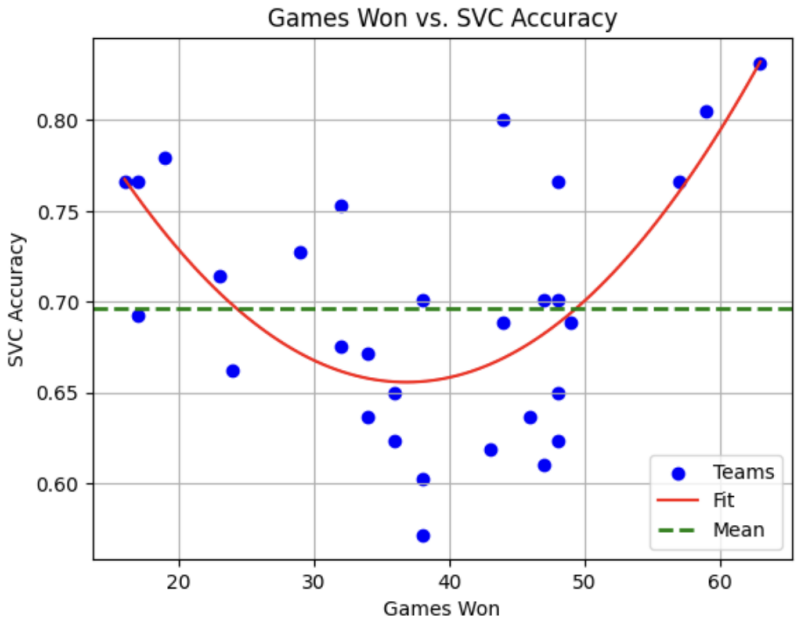


Figure 9: Games Won Vs. SVC Accuracy

As the plot shows, the model tends to achieve higher accuracy for teams that consistently win a lot or lose a lot. This suggests that the model relies heavily on overall team performance when predicting outcomes—which isn't surprising and indicates that it's functioning as expected to some degree.

On the other hand, the model is less accurate for teams in the middle of the standings. This makes sense because those teams are more unpredictable: on any given night, they could win or lose, making them harder to model accurately.

To gain deeper insights into model interpretability, I examined team-specific

precision and recall, which reveal how the model performs beyond overall accuracy. While the function I created can calculate these metrics for all teams, I decided to focus on four: the NBA champion Oklahoma City Thunder, the league's worst team, the Utah Jazz, the Orlando Magic, who finished exactly at 50%, and the Phoenix Suns, who were slightly below average and missed the playoffs.

```
Confusion Matrix for OKC:
                Predicted loss     Predicted win
Actual loss       |        1            13
Actual win        |        0            63

Precision (predicting wins):  0.8289
Recall    (predicting wins):  1.0000
Precision (predicting losses): 1.0000
Recall    (predicting losses): 0.0714


Confusion Matrix for UTA:
                Predicted loss     Predicted win
Actual loss       |       55             5
Actual win        |       13             4

Precision (predicting wins):  0.4444
Recall    (predicting wins):  0.2353
Precision (predicting losses): 0.8088
Recall    (predicting losses): 0.9167


Confusion Matrix for ORL:
                Predicted loss     Predicted win
Actual loss       |       29            10
Actual win        |       13            25

Precision (predicting wins):  0.7143
Recall    (predicting wins):  0.6579
Precision (predicting losses): 0.6905
Recall    (predicting losses): 0.7436


Confusion Matrix for PHO:
                Predicted loss     Predicted win
Actual loss       |       34            11
Actual win        |        8            24

Precision (predicting wins):  0.6857
Recall    (predicting wins):  0.7500
Precision (predicting losses): 0.8095
Recall    (predicting losses): 0.7556
```

Figure 10: Analysis of 4 Teams

The Thunder dominated the season, winning about 83% of their games. It's not necessarily a flaw that the model frequently predicted them to win—this aligns with reality. However, I would have liked the model to capture a few more of their rare losses.

For Utah, I was pleasantly surprised. Their recall for wins was just 0.253, meaning the model only correctly identified about a quarter of their wins. Still, the model predicted them to win nine times, five of which were correct—resulting in a precision of 0.444. While far from perfect, this is better than the model simply predicting them to lose every game.

For the two middle-tier teams, Phoenix and Orlando, the model performed remarkably well. Phoenix had one of the highest individual team accuracies overall, correctly predicting 58 of 77 games (about 75%). Orlando also showed strong results, indicating that the model handled these more balanced teams better than expected.

With these observations in mind, we can now move on to the conclusion to reflect on what worked well and where improvements are needed.

7 Conclusion

Overall, I'm really satisfied with how this project turned out. For the longest time, I struggled to push the accuracy past 64%, so seeing the final SVM classification model reach 69.6% accuracy felt like a huge accomplishment. The other models performed well too, with logistic regression coming in as a close second. The SVM model did an excellent job predicting games for teams that were either really good or really bad, although it did tend to overgeneralize a bit. The features that had the most impact were home-court advantage, net rating, the back-to-back indicator, effective field goal percentage (eFG%), and opponent turnover percentage.

That being said, basketball is about more than just statistics. While I chose advanced team stats for this project—because using basic counting stats would have oversimplified things—there were definitely areas where the model could have been improved. One of the biggest missing components was individual player statistics. Including advanced player-level metrics like Player Efficiency Rating (PER) or similar aggregates would have likely improved performance. The NBA is a star-driven league, and accounting for individual player impact would have been an important addition. I didn't include player stats because I wanted to keep the focus on team-level metrics, but this is definitely something I would explore in a future iteration.

Another major piece that was missing was player health. Ideally, I would have liked to add a feature to represent team health before each game, maybe as a score from 0.0 to 1.0, where 1.0 means the entire roster is healthy and

active. Subtractions could be made for inactive players based on their average contribution to team success, so losing a star player would have a much bigger impact than losing a bench player. The challenge was finding reliable pre-game injury data, which I couldn't get from any existing datasets. Without that, I couldn't include health as a variable, but it would have been an incredibly valuable addition.

Two other factors I thought about were culture and coaching. Both of these play a huge role in team success, but they're nearly impossible to measure in a meaningful way. Great coaches can struggle on bad teams—look at Head Coahc Will Hardy and the Utah Jazz—while average coaches can win titles with stacked rosters. Culture is also extremely influential but very difficult to quantify. As much as I would have liked to include them, it just didn't make sense to force in features without reliable data behind them.

In the end, there's always room for improvement, and this model could definitely be taken further, but I'm proud of what I was able to accomplish. Not only did it perform well in terms of accuracy, but the process of building it and analyzing the results was a tremendous learning experience. This project was challenging, fun, and gave me a lot of insight into how complex predictive modeling can be when applied to something as unpredictable as basketball.

8 Appendix

Weiner, Josh (2021, January 5). Predicting the Outcome of NBA Games with Machine Learning. Towards Data Science. Retrieved from https://towardsdatascience.com/predicti the-outcome-of-nba-games-with-machine-learning-a810bb768f20/

Junwen Wang. 2023. Predictive Analysis of NBA Game Outcomes through Machine Learning. In The 6th International Conference on Machine Learning and Machine Intelligence (MLMI 2023), October 27-29, 2023, Chongqing, China. ACM, New York, NY, USA, 14 Pages. https://doi.org/10.1145/3635638.3635646

DiPerna, Luke (2023, August). NBA Prediction Modeling. Github. Retrieved from https://github.com/luke-lite/NBA-Prediction-Modeling

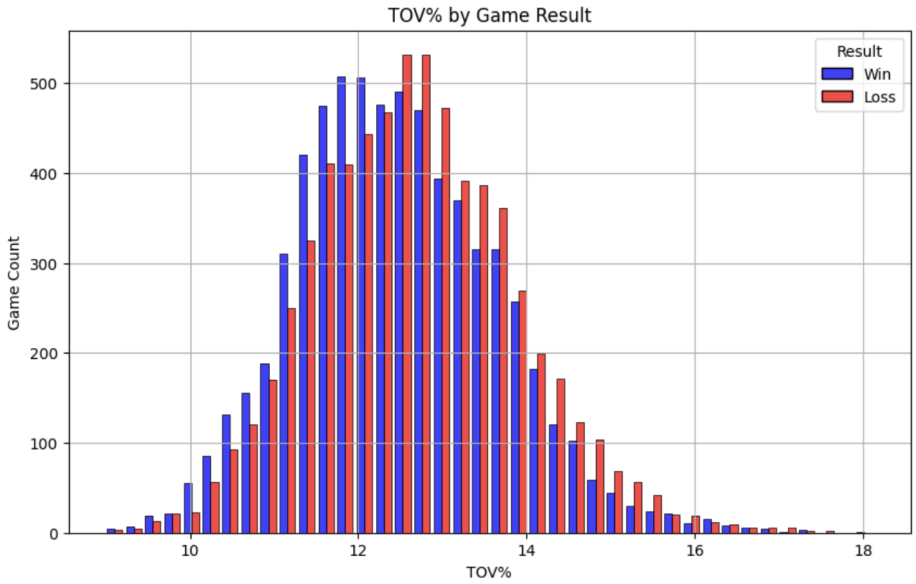


Figure 11: TOV% by Game Result

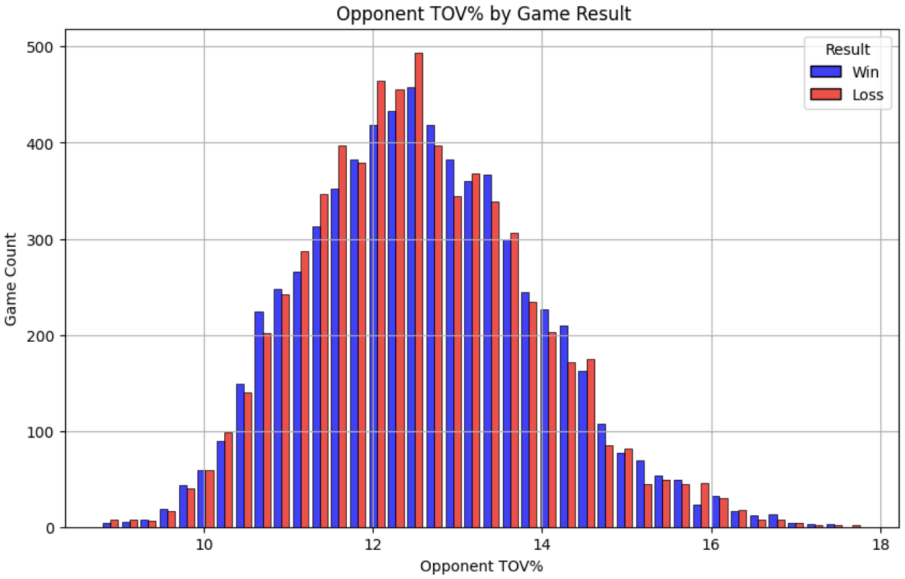


Figure 12: Opponent TOV% by Game Result

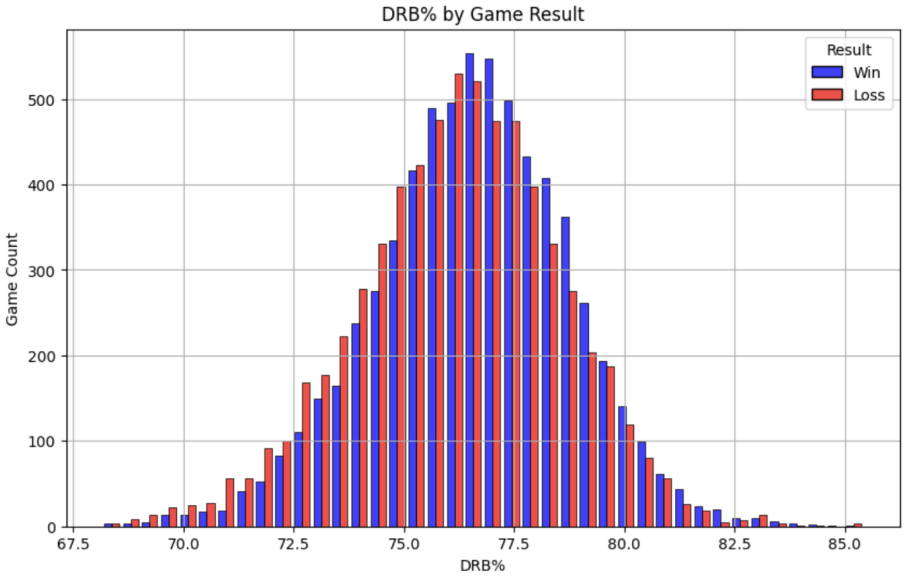


Figure 13: DRB% by Game Result

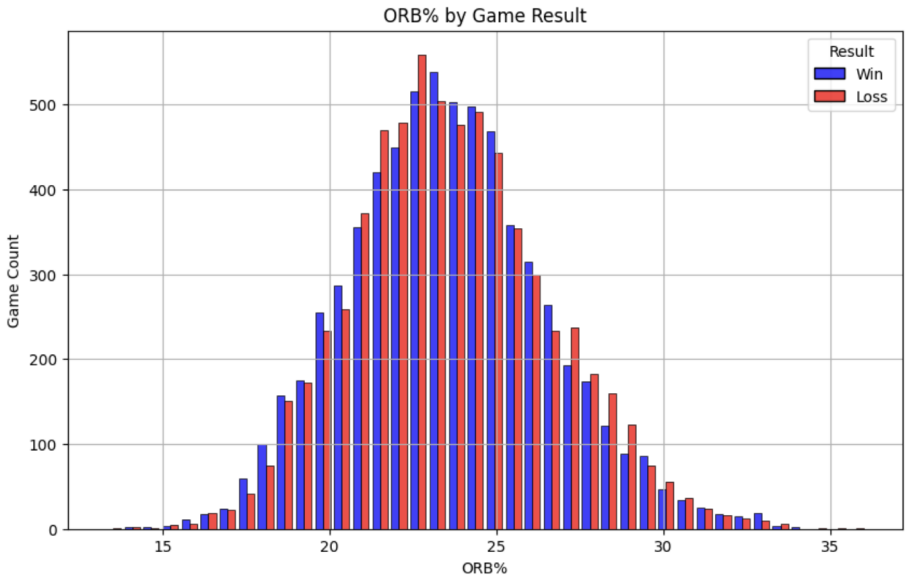


Figure 14: ORB% by Game Result

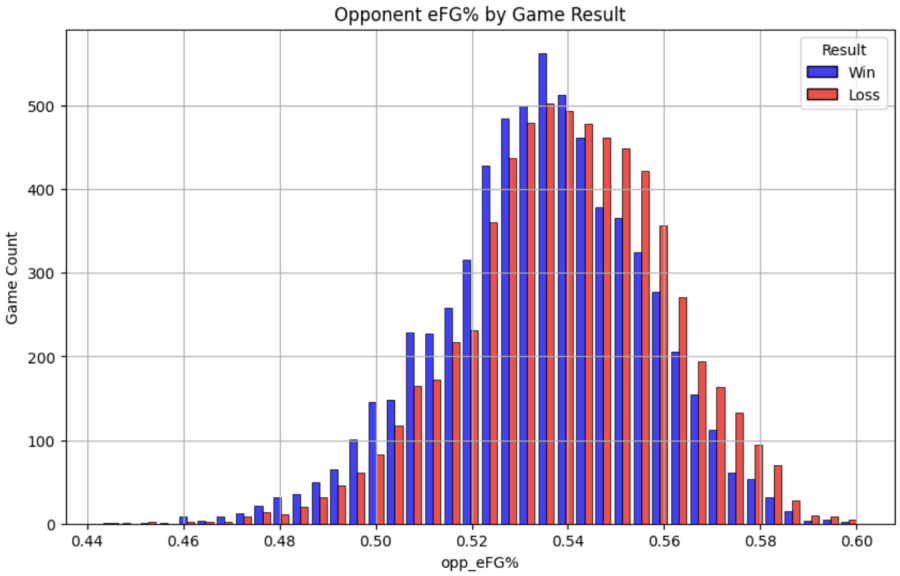


Figure 15: Opponent eFG% by Game Result