Stats 101C Final Project
Predicting NBA Wins

STATS 101C | Prof. Shirong Xu
Tessa Gervase (406229527), Cade Miller (006284059), Rory Freck (306145948), Max Chalekson (505696407), Vikas Sundar (205926235), Marc Walden (905979937)

Table of Contents

[1] Introduction

In this paper, we discuss our analysis of the NBA 2023-2024 Dataset, which is an extensive collection of game statistics that tries to predict the game outcome, (as a Win or Loss). The dataset has a total of 2,460 entries and 24 columns, and contains important features like points scored, assists, turnovers, rebounds, etc. To make sure we followed the project guidelines, we constructed features only based on historical game data that was available before each game. Some notable engineered features to keep in mind include point differentials, turnovers, cumulative weighted averages of prior game statistics, and binary indicators for home advantage.

To proceed with our task at hand, we used a wide range of classification machine learning models, like Logistic Regression, Random Forest, and Quadratic Discriminant Analysis (QDA), in addition to feature selection techniques. The use of recursive feature engineering allowed us to explore new dimensions, including differential defensive and shooting efficiencies, mapping gameplay dynamics with higher granularity.

Our most optimal model was our Logistic Regression one, which achieved a prediction accuracy of **70.2%**, highlighting its strength under different feature engineering scenarios. Other models had their own strengths and weaknesses, such as our Decision Tree model producing easily interpretable classifications, and our QDA model struggling to handle the complex interactions in the dataset, as evidenced by a lower prediction power. These results demonstrate the important role of model selection, especially when considering weighing recent games more heavily and including engineered features.

Our report summarizes the data preprocessing, feature construction, and experimental design processes, all while following the guideline of using only pre-game data. We further review the strengths and weaknesses of each model, investigate ways to further optimize our model of choice, suggest refinements to enhance predictive performance, and finally provide our conclusive findings.

[2] Data Preprocessing

Before constructing any models, our data needed to be processed. This included creating binary variables via one-hot encoding, feature engineering viable predictors, and handling sparse columns in our dataset.

Table 1: Descriptive Statistics of Selected Features

	W/L	MIN	PTS	FGM	FGA	FG%	3PM	3PA	3P%	FTM
Count	2460.000	2460.000	2460.000	2460.000	2460.000	2460.000	2460.000	2460.000	2460.000	2460.000
Mean	0.500	241.362	114.211	42.170	88.903	47.521	12.837	35.104	36.494	17.034
Std	0.500	6.351	12.846	5.343	7.013	5.498	3.837	6.542	8.341	5.890
Min	0.000	240.000	73.000	26.000	67.000	27.700	2.000	12.000	6.900	0.000
25%	0.000	240.000	105.000	38.000	84.000	43.800	10.000	30.000	31.000	13.000
50%	0.500	240.000	114.000	42.000	89.000	47.500	13.000	35.000	36.550	17.000
75%	1.000	240.000	123.000	46.000	93.000	51.200	15.000	39.000	41.700	21.000
Max	1.000	290.000	157.000	65.000	119.000	67.100	27.000	63.000	64.500	44.000

To provide a general idea of the structure of the data, provided above is a summary table of some of our variables prior to any manipulations. From this, we will process the data with the aims to improve the quality of our variables. Additionally, we will explore potential transformations or interactions between them to improve model performance. After these preprocessing steps, the data will be ready for feature selection and model training, ensuring that we can build the most robust classification model.

2.1 Creating the Binary Variable (Win/Loss)

The foundation of the entire project rests on our model's ability to accurately predict if a team is going to win or lose an individual game based on previous game statistics. To do this, we must first encode our dependent, or "predicted" variable, "W/L". Naturally, we chose to one-hot encode wins with a 1 and losses with a 0 for ease of interpretability.

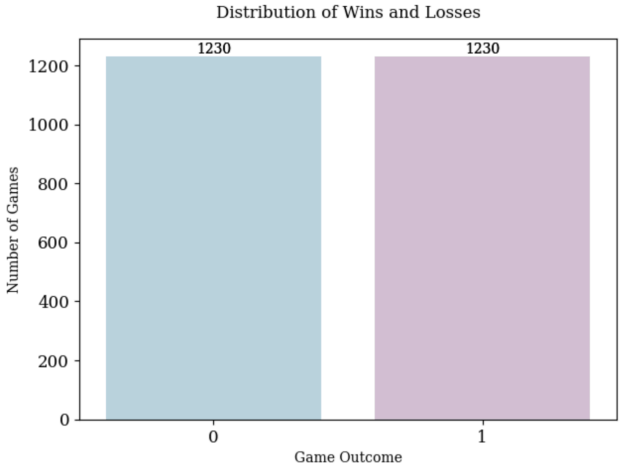

Figure 1: NBA Games Win/Loss Distribution

Another consideration when dealing with classification algorithms is class balance (or imbalance) within our dependent variable. Fortunately, by construction, our dataset has an equal number of wins and losses (as seen in the figure above). This allows us to disregard the possibility of misleading accuracy scores when comparing, choosing, and improving our models.

2.2 Creating Differential Dataframe

To prepare the data for our model to predict future game outcomes from past game data, we created a differential data frame from the dataset provided. First, we converted the 'Game Date' column to Pandas DateTime format for easy sorting. We then sort the dataset to be in chronological order. Using this newly sorted data frame, for each scheduled game, we then calculate the difference in the average of the two team's stats prior to that game date. The result was a data frame that summarizes a direct comparison of past performance for two teams, giving us the foundation we need to build predictive models to forecast future game outcomes.

To calculate the average game statistics of a team prior to an arbitrary game data, we used the following exponential decay formula:

$$Weight = e^{-\lambda(Days\ Before)}$$

This has a decay factor, λ. A higher λ gives more weight to recent games while older games have less influence. This is a natural step when processing our data because, in the NBA, a game from last week is more indicative of a current team's standing than a game result from weeks, or even months, ago. This could result from player morale, injured/benched players, if a team is on a "run", etc. We tested multiple λ's and found $\lambda = 0.05$ produced the best predictive performance.

2.3 Feature Engineering

The motivation behind feature engineering was to improve model accuracy by capturing a more holistic and better-weighted set of predictor variables. We based the features we engineered on their practicality. The most statistically significant ones were "Home Advantage", "Defensive Efficiency", and "Shooting Efficiency". We will now discuss the motivation behind the construction of these features.

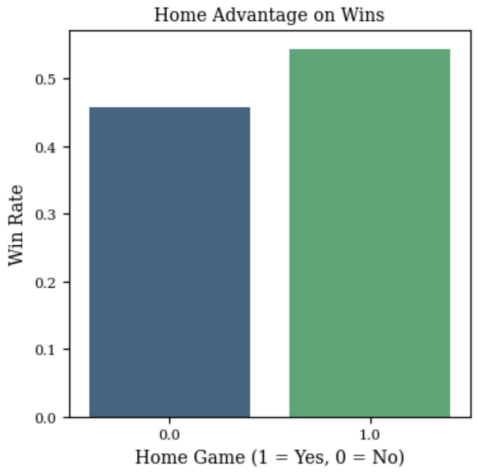

Figure 2: Proportion of Games Won at Home

"Home Advantage" is exactly what it sounds like, a home court advantage. This was the first feature we engineered because teams tend to perform better in familiar environments, benefiting from the support of a home crowd, reduced travel fatigue, and familiarity with the court's nuances. The most recent meta-analyses show that home teams win around 60% of games, making it significant in predicting game outcomes.

"Defensive Efficiency", which we defined as the sum of steals and blocks, is a strong predictor of NBA wins as it reflects a team's ability to disrupt the opponent's offensive plays. Teams with high defensive efficiency can limit scoring opportunities and create fast-break chances, directly impacting game outcomes. After all, everyone knows defense wins championships

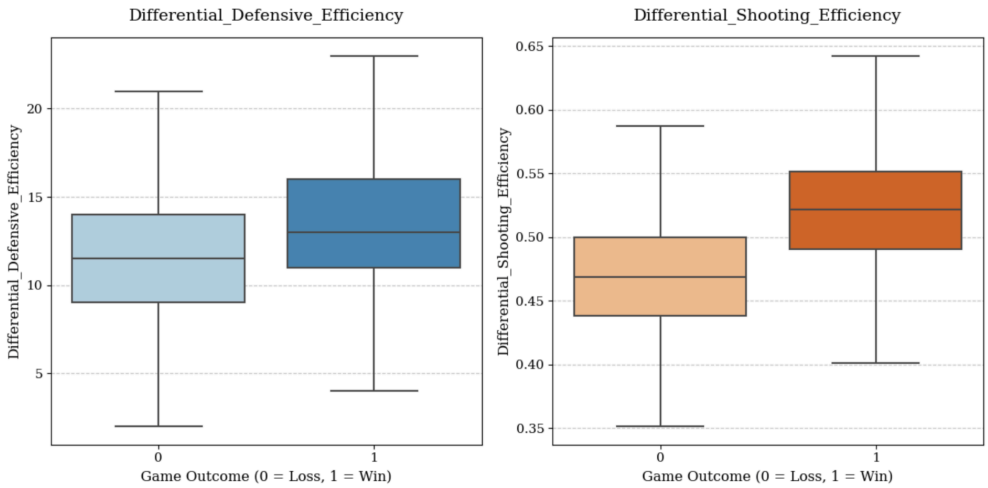

Figure 3: Defensive vs. Shoot Efficiency Distribution

"Shooting Efficiency", which we calculated as the ratio of made field goals, three-pointers, and free throws to total attempts, measures a team's scoring effectiveness per opportunity. Teams with higher shooting efficiency are more likely to capitalize on possessions, leading to more points scored and therefore a greater expectation of winning games. This is the most holistic quantification of offensive ability.

After all feature engineering, we ended with the following variables:

Table 2: Table of All Features

Original Column	**Preprocessing Action**	**Final Column**
Team	Removed	–
Match Up	Removed	–
Game Date	Removed	–
MIN	Difference in weighted averages	MIN
FGM	Difference in weighted averages	FGM
FGA	Difference in weighted averages	FGA
FTM	Difference in weighted averages	FTM
FTA	Difference in weighted averages	FTA
OREB	Difference in weighted averages	OREB
DREB	Difference in weighted averages	DREB
AST	Difference in weighted averages	AST
PF	Difference in weighted averages	PF
+/-	Difference in weighted averages	+/-
PTS	Difference in weighted averages	PTS
FG%	Difference in weighted averages	FG%
3PM	Difference in weighted averages	3PM
3PA	Difference in weighted averages	3PA
3P%	Difference in weighted averages	3P%
FT%	Difference in weighted averages	FT%
REB	Difference in weighted averages	REB
STL	Difference in weighted averages	STL
BLK	Difference in weighted averages	BLK
TOV	Difference in weighted averages	TOV
W/L	Engineered as binary response	W/L
home	Engineered as binary (1 for home games, 0 for away games)	home
Differential_Defensive_Efficiency	Calculated using differential statistics	Differential_Defensive_Efficiency
Differential_Shooting_Efficiency	Calculated using differential statistics	Differential_Shooting_Efficiency

2.4 Filling Missing Values

When working with the NBA dataset for the project, to predict the wins, we need to account for the missing values in the dataset. This was done to ensure the integrity

of the modeling and analyses done for this project. Through the different modeling methods that we've attempted: decision tree, logistic regression, random forest, and quadratic discriminant analysis (QDA) - numerical missing values in the feature matrix were replaced with 0.

When looking at the FT% column, it contained potential non-numeric entries, which were converted to `NaN` and filled with 0. This approach allowed us to avoid data loss and maintain consistency within our calculations, especially during the model training, and ensure the accuracy of each of the models.

We ran a quick count on the number of 'NaN' and numerical missing values in each of the columns, and there were so few that replacing them with 0s had virtually no effect on our model performance.

[3] Experimental Setup

3.1) Feature Selection

After feature engineering, we made it our goal to find important predictors for our model. We used the random forest feature selection method and created a bar chart of each predictor's mean Gini decrease. Predictors with higher mean Gini decreases contribute more to the model's predictive power. Removing these predictors would cause a greater reduction in model performance compared to predictors with lower mean Gini decreases.

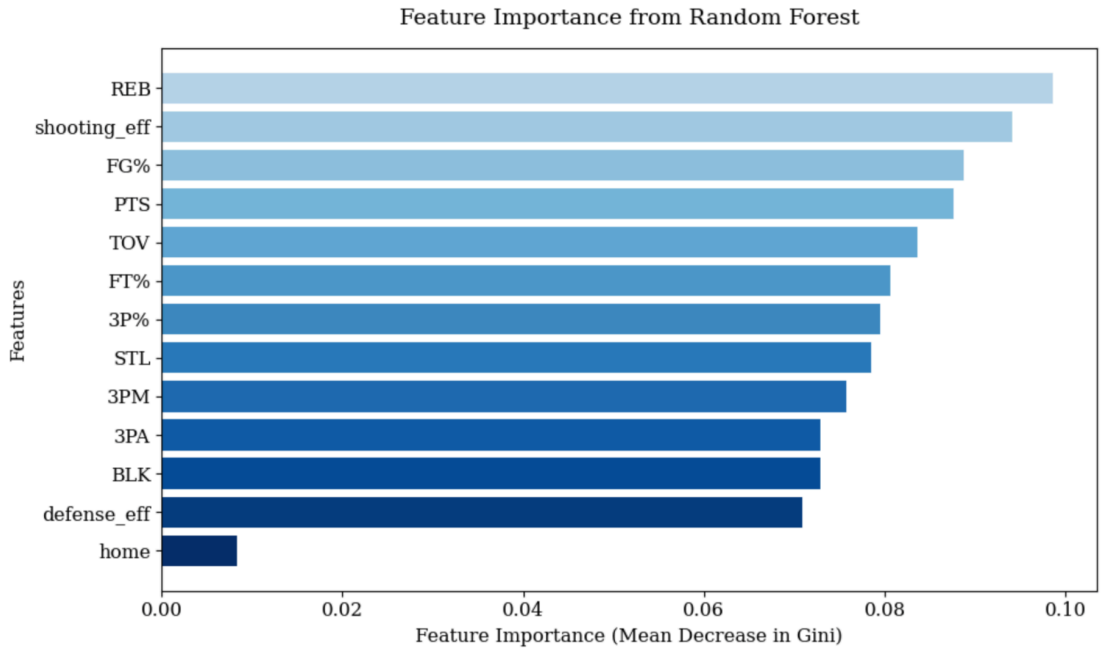

Figure 4: Using Random Forest on Features

The figure above displays the most significant features, by measure of Gini Importance, in addition to our engineered features. Home court advantage had a low mean Gini decrease, so we decided to remove it from our model.

3.2) Models of Interest

With 12 predictors finalized, we ran a variety of models to find which model had the best predictive performance. The models explored in this study are Logistic Regression, Quadratic Discriminant Analysis (QDA), Random Forest, Decision Tree, and Adaboost.

1. Logistic Regression: These models excel in classification problems when the data has linear relationships, and since we hypothesized that the data would possess linear trends, we predicted this model would perform well.

2. QDA: QDA assumes a unique joint normal distribution of each of the classes. We chose to examine this model because of its capability to produce highly non-linear decision boundaries, such as ellipses and hyperbolas, consequently yielding strong classification accuracies.

3. Decision Tree: A Decision Tree is an ensemble method that can detect nonlinear relationships in classification problems by learning implicit, hierarchical rules of the dataset. It is easily interpretable and is strong at handling categorical and numerical data at the same time.

4. Random Forest: Random Forest is an ensemble method that is built on decision trees. It uses the weights of predictors to classify outcomes in linear and nonlinear relationships. After we performed a grid search, we found a max depth of 5 to be the optimal hyperparameter.

5. AdaBoosting: This model learns from misclassified choices and corrects itself by assigning higher weights to difficult-to-classify instances. Its adaptive nature has the potential to produce high accuracies on our data's testing set.

3.3) Comparing Model Performances and Model Selection

The next step of the experimental process was to run the models on the training data set and measure their accuracy on the testing set. Due to the sequential nature of the data, we had to be cautious in maintaining its chronological integrity when splitting the training and testing data. Earlier we discussed the use of weights to increase the importance of recent game statistics relative to previously played games. This feature engineering technique makes the data order immutable, as the model uses previous games to predict future wins. With that said, we were unable to use a standard k-fold cross-validation procedure, because assessing model accuracy on randomly permuted training and testing data is a violation of the sequential nature of the data, as previously discussed

As a result, we needed to use a method that ensured the data's structure was maintained, so we used a train-test split that took the first 75% of the data to be used for training, and the latter 25% of the data to be used for testing. This straightforward method can be manually applied to ensure the data is not shuffled. We then calculated the accuracies of each model using our selected predictors and compared their results.

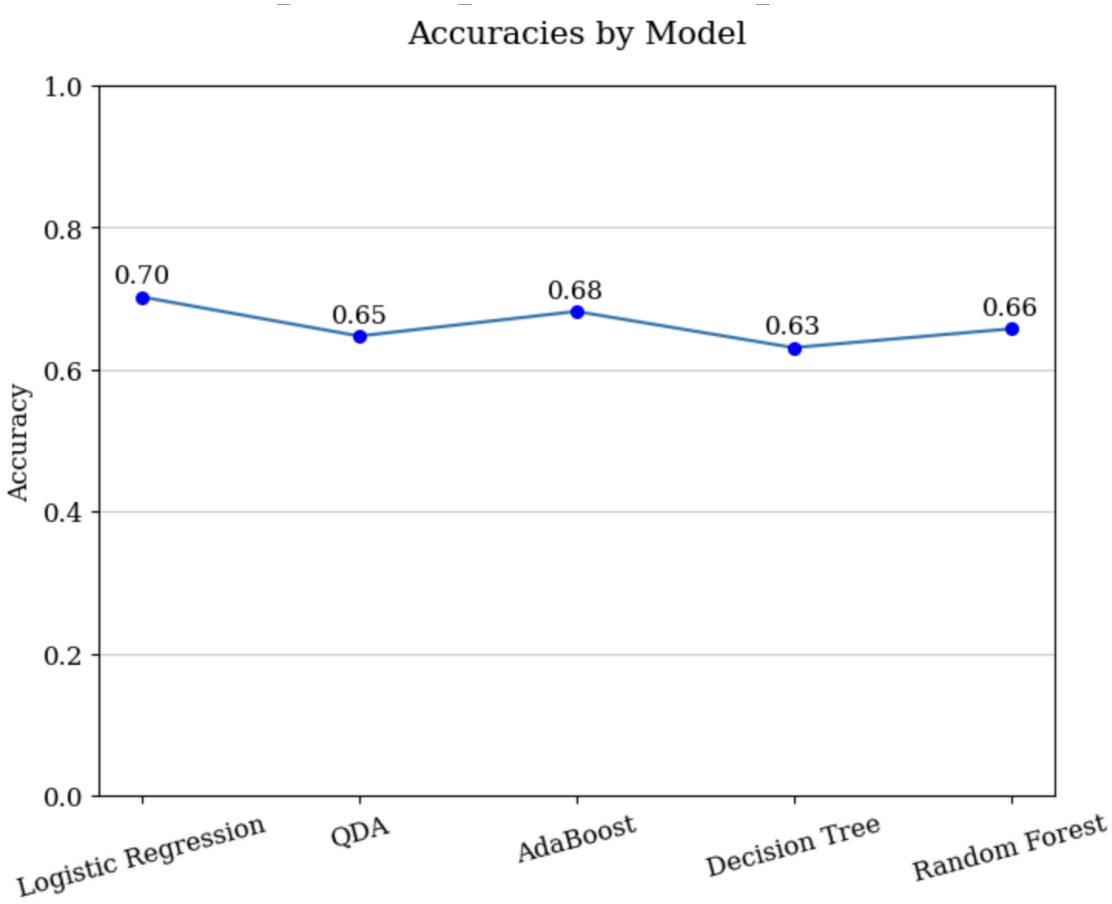

Figure 5: Comparing Accuracies of Models

Above are the testing accuracies for each of our models. Logistic regression was selected as our final model because it yielded the highest prediction accuracy over the testing set.

[4] Model Analysis

4.1) Model Enhancement

After choosing logistic regression as our final model, we sought to improve it. The objective of our study was to maximize predictive power, so we addressed the potential shortcomings of our models. One such factor that was salient in our model was multicollinearity in the predictors.

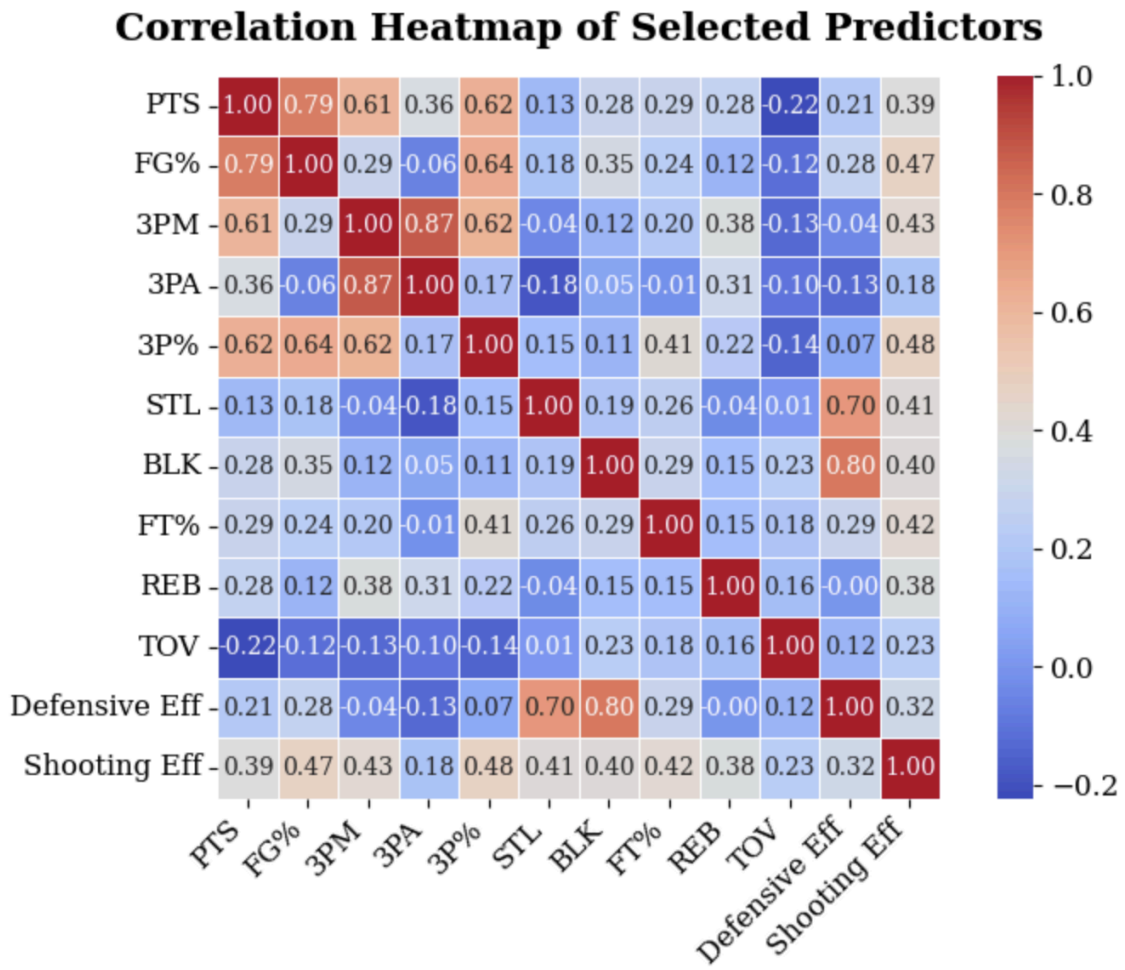

Figure 6: Correlation Matrix of Selected Predictors

We noticed a few predictors that were highly correlated with one another, so we decided to enforce an L2 penalty in the model to reduce the multicollinearity. We used GridSearch to find the optimal hyperparameter, which resulted in a moderate L2 penalty (C = 1) being the best medium between predictive power and multicollinear handling. The logistic regression model with the L2 penalty achieved a marginally higher accuracy while penalizing highly correlated predictors by shrinking them slightly.

Table 3: Comparison of L2 and Regular Logistic Regression Coefficients

Feature	L2 Coefficient	Regular Coefficient	Difference
Differential_Shooting_Efficiency	0.5048	0.5081	Minimal
FG%	0.2926	0.2924	Minimal
TOV	-0.2030	-0.2029	Minimal
STL	0.1441	0.1440	Minimal
BLK	-0.0477	-0.0476	Minimal
FT%	0.0194	0.0194	Minimal

The shrinkage in the coefficients is minimal, but we forfeited predictive power when we increased the regularization any further. Next, we shifted our focus to other metrics of this model.

4.2) Model Results

Table 4: Confusion Matrix - Logistic Regression

True Label	Predicted: Loss	Predicted: Win
Loss	173	73
Win	73	171

The confusion matrix shows how well the model predicts "Loss" and "Win" outcomes. Out of 490 samples in the testing set, it correctly predicted 173 "Loss" cases and 171 "Win" cases, achieving an overall accuracy of 70.2%. However, it misclassified 73 instances for both categories. While the model performs well, nearly 30% of predictions are incorrect.

Table 5: Classification Report

Class	Precision	Recall	F1-Score	Support
Class 0	0.70	0.70	0.70	246
Class 1	0.70	0.70	0.70	244
Accuracy		0.70		490
Macro Avg	0.70	0.70	0.70	490
Weighted Avg	0.70	0.70	0.70	490

The classification report provides deeper insight into the model's performance. Precision, recall, and F1-score are all 0.70 for both "Loss" and "Win." Precision reflects how many of the predicted "Loss" or "Win" cases were actually correct, so 70% of predictions for each class were accurate. Recall measures how well the model identified all actual "Loss" or "Win" cases. The report shows that recall correctly captured 70% of the true outcomes in each category. The F1 score is the balance between precision and recall, combining the two into a single metric.

[5] Conclusion

In this project, we aimed to predict the outcome of NBA games using a variety of machine learning models and pre-game statistical data. Through extensive data preprocessing, feature engineering, and model evaluation, we identified that Logistic Regression provided the most accurate and reliable predictions, with an overall accuracy of 70.2%. This model was able to predict both wins and losses with a true positive and true negative rate of 70%. Our approach, which included analyzing engineered features like defensive efficiency, shooting efficiency, and home court advantage, proved effective in capturing important game dynamics. Given its balanced performance across both classes and its good interpretability, we conclude that Logistic Regression is the best choice of model. That said, we also tried other models, such as QDA, Decision Trees, Random Forest, and AdaBoost, which yielded similar, but slightly worse results.

While additional hyperparameter tuning and even other model types may improve performance, this study successfully demonstrates that NBA game outcomes can be predicted with reasonable accuracy using historical game statistics. The findings contribute to a deeper understanding of the factors influencing game outcomes and set the stage for future research to enhance prediction accuracy in sports analytics.

Possible extensions of this project include incorporating additional features such as player-level statistics, injury reports, and changes in player lineups. Deep learning techniques like neural networks could be used for more complex pattern recognition. The model could potentially use in-game data to update in real-time, and predict other important statistics.