Predicting Outcomes of NBA Games Based On Historic Data

Mehrnaz Bastani, Ruchira Bhat, Ronit Bhatt, Shiun Choi, Jack Conner, Sujay Syal,

December 6, 2024

Contents

1 Introduction **2**

2 Feature Engineering and Selection **2**

2.1 Win/Loss Indicator Column 2

2.2 Home Game Column 2

2.3 Redundancy in Field and Three Point Goals and Free Throws 4

3 Model Development **7**

3.1 XGBoost Model 7

3.2 Generalized Least Squares Approach 8

4 Results and Analysis **9**

4.1 Model Evaluation 9

4.2 Model Strengths 10

4.3 Model Limitations 10

1 Introduction

In this paper, we aim to predict the outcomes of NBA games based on past games. Specifically, we were given data relating performance by NBA teams during the 202/2024 season. Our goal was to train a machine learning model that would give us at least 70% accuracy in predicting future games based on data from the 2023-2024 season. We were given the following features to work with during our analysis: team, match-up, game date, Win-loss indicator, minutes played, points scored, the following metrics for field goals, three point goals, and free throws: made, attempted, and percentage. We were also given offensive, defensive, and total rebounds, assists, blocks, steals, turnovers, personal fouls, and the overall +/- score at the end of the game.

We trained our machine learning model following feature selection and engineering. The model that we selected was initially XGBoost, which had resulted in a 55% testing accuracy. In order to improve that metric, we iteratively tested and created a custom model that predicts based on minimizing least squares, specifically weighted by the recency of the historic games. This model was able to generate a 73% test accuracy. We will now explain initial findings in the data based on feature selection, feature engineering, our model selection, and finally our results and analysis.

2 Feature Engineering and Selection

Before we could dive into our model, we did some exploratory analysis to determine the features that should be included in our final model. This included organizing existing columns to better allow for the extraction of important information from these columns. It also included removing columns that presented redundant information.

2.1 Win/Loss Indicator Column

Our first change was a very simple and easy fix. The W/L column in the original dataset was a string 'W' or 'L', meaning that it wasn't easy to decipher or use. Instead, we decided to code it as a binary variable, using 0 or 1.

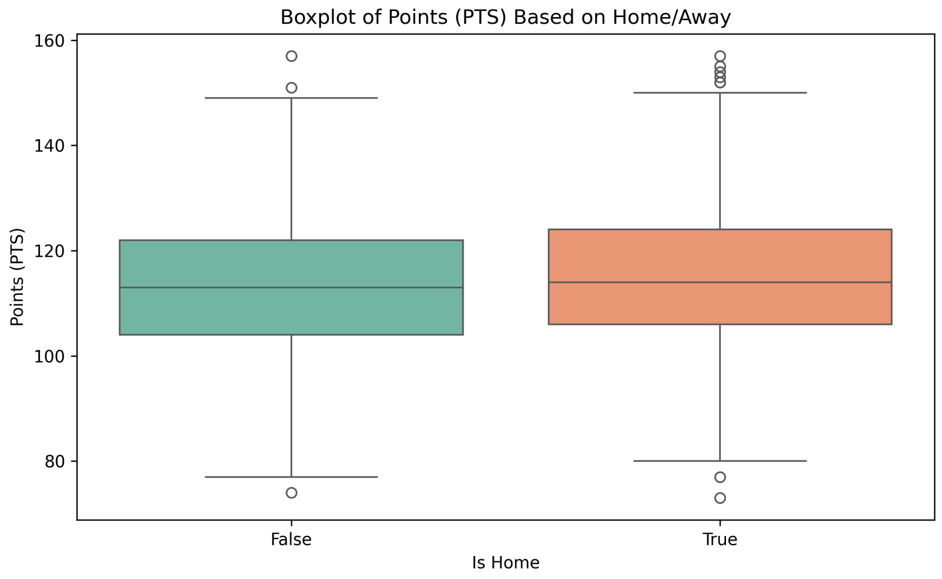
2.2 Home Game Column

During our analysis, one of the factors that we believed could be valuable was the binary variable of whether or not a certain game was a home or away game. However, in the current data set, we were not able to extract the information easily regarding whether or not the game was a home or away game. Instead, the only information provided was the Matchup column, where the symbol @ indicated that the second team was the home team (NYK @ BOS) and the vs. symbol indicated that the first team was the home team (NYK vs. BOS).

In order to extract the home or away information, we split the strings in the Matchup Column. We then made new column titled 'Is Home', which encoded whether the 'Team' was playing at their home court.

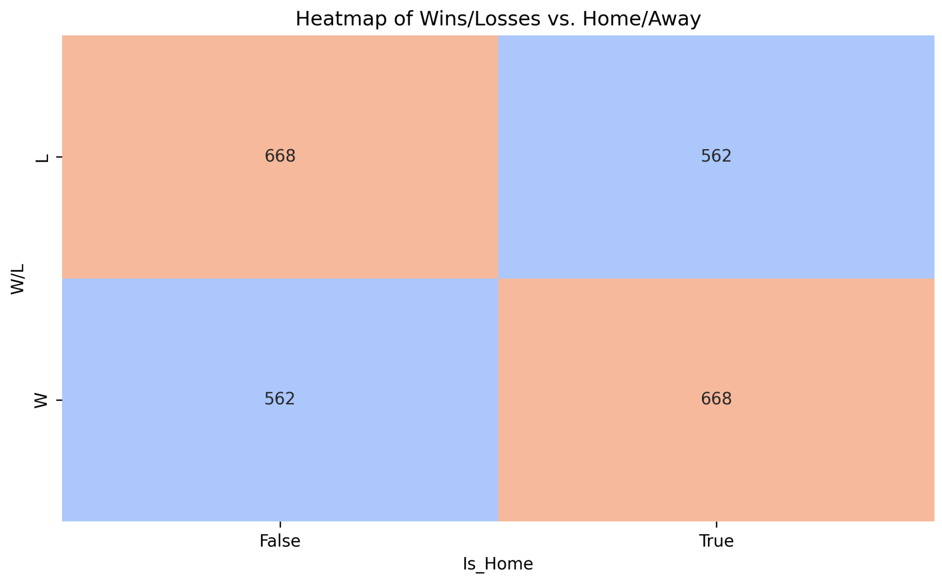
Is Home is a binary variable column.

Once we created this column we were able to see whether or not the number of points scored was different if a team was the home team or not.



From this boxplot, it is clear that there is a difference between number of points scored based on if the game was played in a home court or not. Although this difference may seem marginal, it is important to note that while predicting human behavior, even small differences might greatly improve predictability. Further, since our dataset spans all existing games in the season, we are able to conclude that in this season there was a difference in number of points a team was able to score based on if the game was in their home court or not

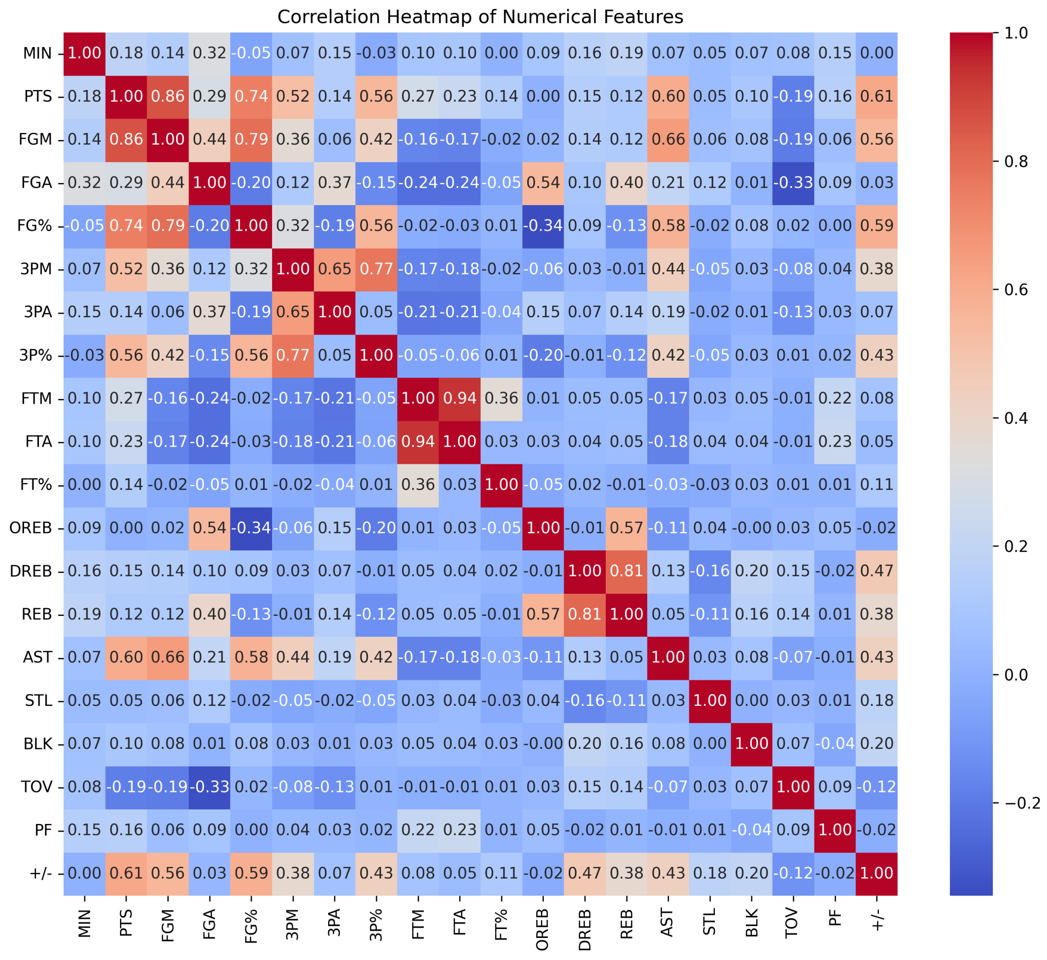
Let's see whether a similar relationship exists when looking at whether a team is playing in their home court and whether or not they win a game.



From this simple heat map, we can see that there is a higher percentage of teams that win games when they are on their home court. 668/1230 (54%) of teams won games on their home court while only 562/1230 (46%) of teams won games when they were not on their home court. Both of these visualizations provide justification for including the "Home Game Advantage" as a feature in our final model.

2.3 Redundancy in Field and Three Point Goals and Free Throws

While selecting our variables, we also wanted to make sure that our column information was not correlated with one another. In order to see correlation, we ran the folowing correlation matrix heatmap:

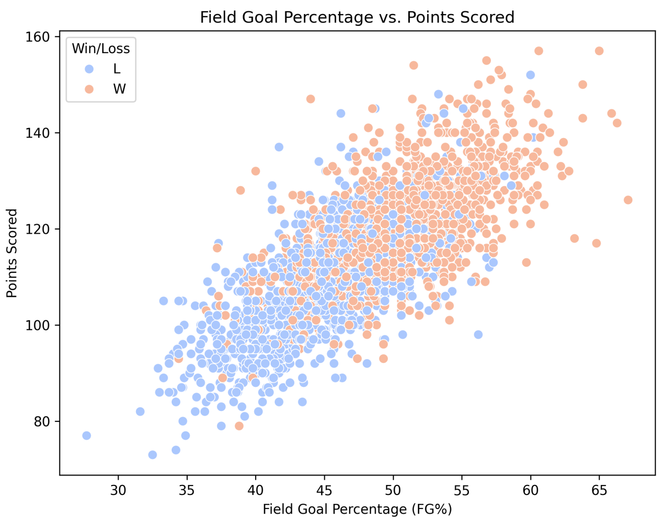


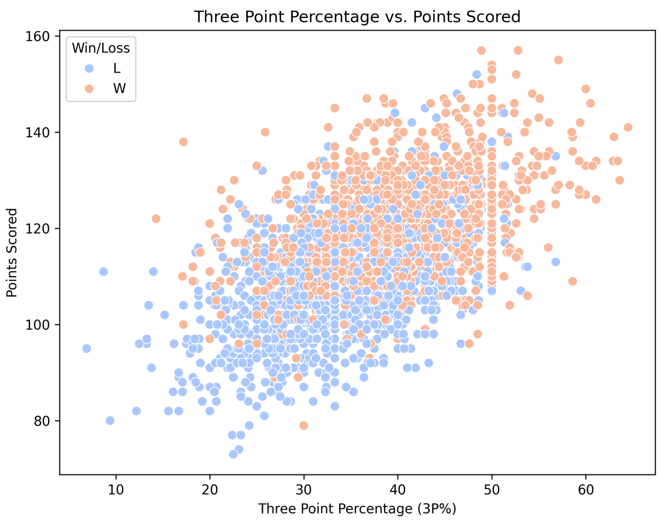
Correlation coefficients range from -1 to 1, where values closer to -1 or 1 indicate higher correlation

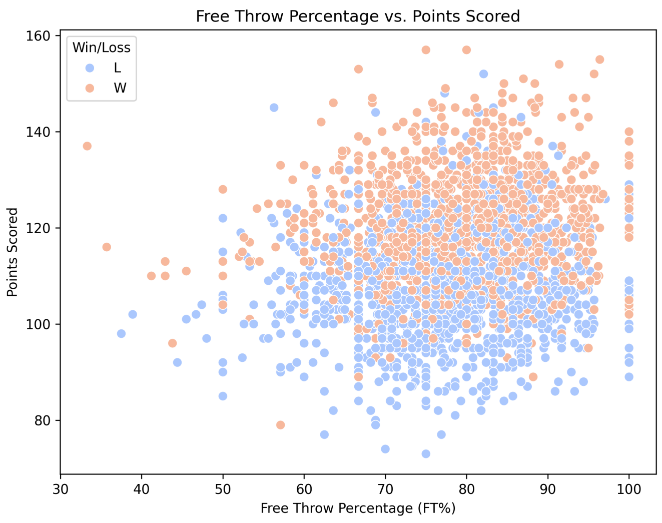
From this heatmap, it is clear that some of the features are strongly correlated with one another. On closer glance, it is clear that the columns FGA (Field Goal Attempted) And FG% (Field Goal %) are very correlated. This does make sense, since these metrics: the attempted, made, and percentages for field goals, three point goals, and free throws are overlapping in the information they give us.

In order to consolidate this information, we decided to instead only include the Field Goal Percentage, Three Point Goal Percentage, and Free Throw percentage instead of the trio of metrics for these values. We used percentage instead of attempted or made, since the percentage is simply made/attempted.

From here, we had to decide if these features added predictive power to determining the number of points a team scores in a given game. We investigated the following features: field goal percentage (FG%), three point percentage (3P%), and free throw percentage(FT%).


Field Goal Percentage vs. Points Scored
Win/Loss
L
W
Points Scored
Field Goal Percentage (FG%)
160
140
120
100
80
30
35
40
45
50
55
60
65

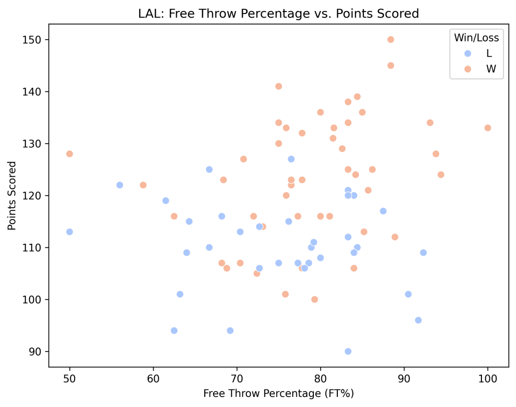

Three Point Percentage vs. Points Scored
Win/Loss
L
W
Points Scored
Three Point Percentage (3P%)
160
140
120
100
80
10
20
30
40
50
60


Free Throw Percentage vs. Points Scored
Win/Loss
L
W
Points Scored
Free Throw Percentage (FT%)
160
140
120
100
80
30
40
50
60
70
80
90
100

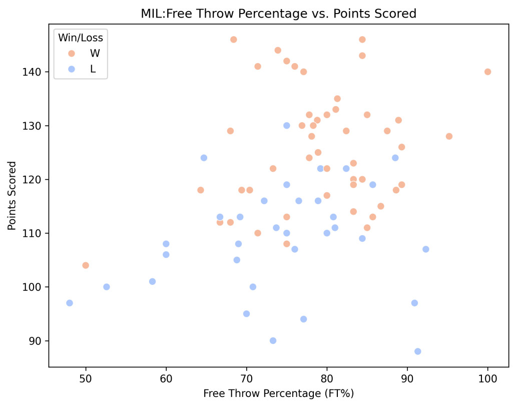
As we can see, there is a clear relationship overall between the percentage of field goals a team makes, and both the number of points, and whether or not they win or lose the game. As field goal percentage increases, the points scored by a team in a given game seems to increase as well. Furthermore, at a certain point threshold, teams are more likely to win or lose, indicating that field goal percentage is also correlated with the W/L of the team in a game.

Similarly to the Field Goal Percentages, there seems to be a positive correlation between percentage of Three Points a team makes and both their score and their W/L probability.

The graph between free throw percentages and scored points does not seem to show the same positive correlation. Instead, there seems to be no relationship between the two variables. However, before not including this feature in our model, we will ensure that there is no seeming correlation between these two variables in any individual team.



(a) Los Angeles Lakers



(b) Milwaukee Bucks

However, when we look at individual teams, there seems to be some correlation between free throw percentages and points scored, with teams such as the Los Angeles Lakers and Milwaukee Bucks reaching higher points only when their Free throw percentage is high. Thus, we will include the feature in the model.

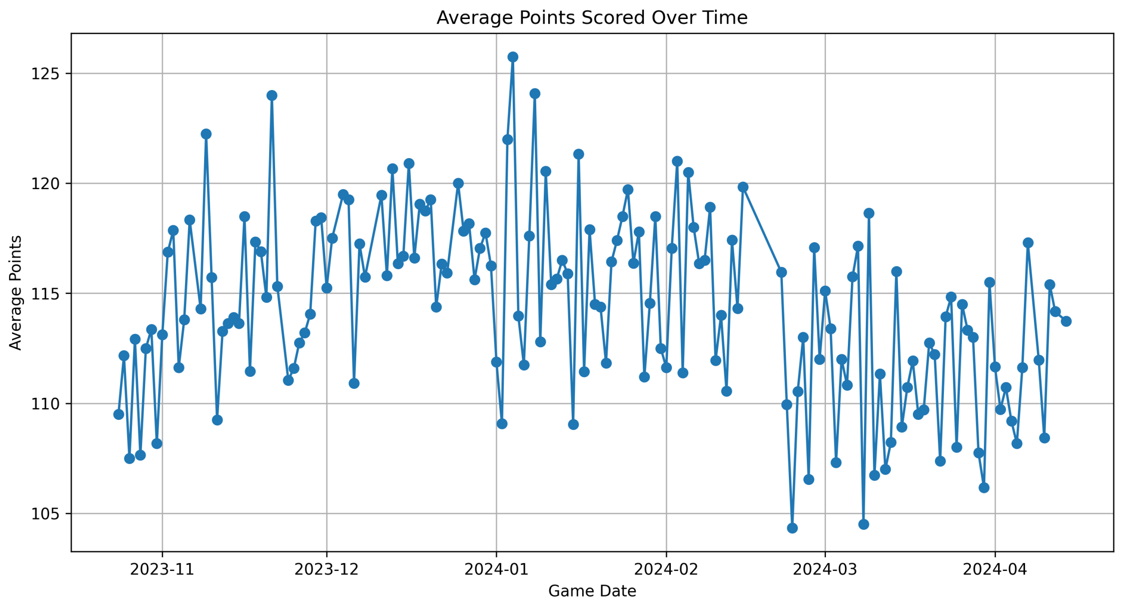
3 Model Development

3.1 XGBoost Model

After creating our self-designed features, we formed a new dataset out of the standardized, averaged statistics of previous games for each team. In this dataset, each row corresponds to the average of the standardized statistics for that team from the previous five games. Thus, it was a slightly smaller dataset, but one which contained data that could actually be used in training our model. This is because each row only contained two non-standardized variables: the binary win/loss, indicating whether that team won or lost the game on that date, and the binary Is Home variable, indicating whether or not the team played that game in their home stadium. From here, we fit a gradient boosting tree model using XG

Boost, shuffling our standardized, workable data and splitting it into a training-test split with 70% of the data in the training set and 30% in the testing set. After fitting our model on the trained data, we evaluated it on the test data to determine its predictive strength. To our surprise, the boosting tree did little better than random chance (accuracy = 55%), and was ultimately not effective in predicting a win or a loss for a game given past statistical averages for the teams involved. We then investigated which features were most useful in the prediction, and found that FG%, STL, and AST were the most useful, while Is Home was the least useful by a large margin. One possible reason this model failed is that it doesn't consider the most recent games as being weighted more.

In order to perform a sanity check to determine if time does in fact affect the points scored, we created a scatterplot of the two variables:



From this figure, it is clear that the average points do vary over time, meaning that we might have better predictability if we allow for more importance for recent games. This led us to creating our final model, which leverages a least squares approach to optimize time-based weights in predicting how many points a team will score based on past game statistics.

3.2 Generalized Least Squares Approach

Since the XGBoost model accuracy was not high enough to be used as a tool for prediction, we took a step back and decided to approach the problem in a different way.

Our main goal with this analysis is to find the outcome of some current NBA game based on previous game information. Another way to view this problem is to determine which of the two teams will score more points in the current game based on their past games in the season. Thus, we can utilize a least-squares approach to predict game-day point totals for each team by minimizing the least-squares term $||y - w^T X\beta||_2^2$, where X is a third-order tensor of dimension n x k x m representing the previous game data for the previous k games, wherein each previous game data contains the n games prior to that game with m features, w is a vector in R^n with n components, providing a time-based weight to be applied to the n most recent games, β is a vector in R^m with m components representing regression coefficients for each of the time-weighted features of X, and y is a vector in R^k representing the actual score in the kth game.

To make this explanation more clear, imagine we want to predict the outcome of the game where the LA Lakers play the New Orleans Pelicans on April 14, 2024. Say we wish to use information from the previous 3 games to predict the outcome. In other words, our input into the model will be the tensor of dimension 3 x 1 x m, where m is the number of features we consider. Since we are only looking to predict one game, we end up with $k = 1$. Let us then state that this game is the Lakers' ith game. Then, for training our model, we will consider $k = i - 4$; essentially, we will calculate the optimal weights and regression coefficients trained on all games from the 3rd game (since we consider only the previous 3 games) to the $i - 1$th game (i.e. the game before the one we want to predict). This takes full advantage of all previously gathered information, thus making our model more effective at producing the correct output. To continue with our example, then, we have our training set of data given by X in $R^{3xk-4xm}$, and y in R^{k-4}. These represent the outcomes of the jth game, where j is a game in the set of $k - 4$ games considered. Essentially, for each game we have in our training set, we have the actual score of the game, which will be y_j, and the scores of the previous 3 games before the jth game. In other words, each slice 3 x j x m gives us the game statistics of the 3 games before the jth game. We then utilize Python's scipy.optimize package to minimize the least squares problem $||y - w^T XB||_2^2$ after initializing some random weights in R^3 and some random regression coefficients β in R^m. We also introduce the constraints that all w_i in $w >= 0$, and that sum $\sum w_i = 1$, to ensure we accurately depict a time-weighted average. After fitting this model, we extract some w_{opt} and β_{opt}, the optimal w and β that minimize the least squares problem on our training dataset, and use them to predict the score of the current Lakers' game based on the previous 3 games' statistics. This will give us a prediction y, which we then compare to the y we get when we run the same process for New Orleans.

4 Results and Analysis

4.1 Model Evaluation

When we tested our model, we didn't necessarily have a training or testing dataset, as the model trained on each team's previous game information for each prediction calculated. Instead, we used the proportion of correct predictions over total predictions for the last 1500 games to evaluate our model's performance. This was done to ensure adequate sample size in calculating the predicted scores for each team, as the model needs at least five previous games to function properly, and performs better with more data to train on. We ran our model on these last 1500 games, utilizing the five most recent games for each previous game gathered in the training step. We then used all previous subsets of the five most recent games to predict the score for each game based on the teams' five previous games, giving us a final accuracy of roughly 0.73, or 73%.

This process is similar to Leave One Out Cross Validation, as we are defining testing accuracy based on the accuracy of the model with all of the prior data except the current data that we are using to test. However, it is a little different, since we are also leaving out the data that comes after the test case, since data after the current game cannot be used in prediction However, this still offers us a good estimate into the accuracy of our model because we are testing with data points that are not included in the creation of our model.

4.2 Model Strengths

The best aspect of our model is that it makes full use of all previously available data for each team. By training the model on all of this past data in each iteration, our model is better able to predict a single point than a model trained on some mass training dataset. In other words, because our model is individually trained at each game matchup, its prediction is more likely to be correct for that game than it would be if it were trained on some x random games from the dataset.

Additionally, because our model minimizes our time-based weights and regression coefficients using a least-squares approach, we are able to mathematically find the true optimal values of w and B at each time point, resulting in a model that doesn't require supervision in order to return an accurate prediction. The model also provides a decent amount of flexibility in its hyperparameters; in particular, we can set both n and k to be however many games we choose. If we only want to consider the weights and coefficients from the past 3 games based on the past 5 games, we would be looking at n = 5 and k = 3. In our model, we used $n = 5$ and $k = j - 6$, where we were predicting the jth game, but, if one wished to reduce the time required to run the model, it could easily be changed to $n = 3$ and $k = 3$, for example, to reduce the number of computations required to find the optimal weights and coefficients.

4.3 Model Limitations

While our model does well in estimating accurate final scores for each game, it could certainly be improved by considering the effects of the opposing team on the score. For example, if one team is doing well offensively, but the other team has a strong defense, then it should be expected that the offensively strong team performs worse than expected in the game due to the effect of the other team's defense. This could be added as another term in our least squares problem, but would require more calculation and a different format of data input to implement within our code.

The biggest area for improvement in our model is in its computational complexity. Since each prediction for the model involves training the model on 5 x k games, the computation required to train each new game increases as the number of previous games (i.e. k) does. While this method makes the most use of the data available, it comes at a steep cost in terms of the time required to run the model. While our model achieved accuracy slightly higher than the test case presented of Boston versus the Lakers, it took a great deal longer to do so, taking approximately an hour to run fully over the final 1500 games.