-

Player Performance Prediction Automation for DraftKings

*Major Qualifying Project*

Advisors:

Donald Brown
Randy Paffenroth

Written By:

Kayleigh Campbell
Benjamin Huang
Gabriel Katz
Skylar O'Connell
Diego Villalobos Gonzalez


WPI

A Major Qualifying Project
Worcester Polytechnic Institute

Submitted to the Faculty of the Worcester Polytechnic Institute in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science, Electrical and Computer Engineering, and Mathematical Sciences. This report represents the work of one or more WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on the web without editorial or peer review.

August 2020 - May 2021

# Contents

# 1 Abstract

In this Major Qualifying Project, we worked with the Boston company DraftKings, an online fantasy sports and sportsbook provider, to build models that predict NBA players' performances in an individual game basis (number of points, assists, and rebounds scored in a single game). We also created models that predict individual game outcomes in the form of a sportsbook closed line. We built linear regression, Lasso regression, random forest, and neural network models which utilized data directly from DraftKings and reputable third-party sources to predict these performance metrics. Our most successful models reduced the prediction error by up to 11.46% from the baseline model predictions. When compared with current sportsbook information and newly developed DraftKings data science models, our results were found to be on par with or surpassing industry standards.

# 2 Acknowledgements

This Major Qualifying Project would not be possible without the contributions of certain individuals at Worcester Polytechnic Institute and DraftKings. We would like to thank our advisors, Professors Randy Paffenroth and Donald Brown, for the invaluable guidance and support provided throughout the project. We would also like to thank Brandon Ward and Ramin Ghelichi at DraftKings for the insight and resources they provided us in their advisory roles.

# 3 Executive Summary

This project aimed to develop machine learning models for DraftKings's online sportsbook player props betting feature. We were successful in developing models for player performance predictions in three main player statistics - points, rebounds, and assists. Our models were compared with current and developing industry models and performed on par or better in all three statistical categories.

For sportsbooks that look to minimize risk and maximize profits, it is important to set strong opening lines. The less a sportsbook's original line changes during a game and maintains at least 50% of the money bet above and below the line, then the sportsbook is less vulnerable to risk and it will probably generate revenue from the bets. Previously, DraftKings has outsourced their lines production to a third party, which is not only costly, but it can also open them up to potentially risky transactions by causing them to be slightly behind changes made to the lines. The goal of this project was to use historic data to create models that predict future player performance, and in turn help generate the best possible opening lines for DraftKings's online sportsbook.

From the very beginning of this project we had to adjust and re-format our data to best suit what we were looking to predict, the models we were trying to create, and to more effectively run our Python scripts. Through the use of data frames, cleaning tools, and scraped information from the web, we were able to effectively use the large amounts of data it takes to successfully predict player performance. We utilized several machine learning techniques, including linear and Lasso regression, random forests, and even neural networks, which allowed us to create several models to work with and compare.

Across all of our models and performance metrics, we discovered one of the most important steps when building these algorithms was how we defined our training and testing distributions in order to train and test the models. In our game line prediction models this was accomplished by re-training the models every three months of data and testing on the next month. Any less frequently meant there was not enough data to confidently predict game outcomes, and any more frequently meant the distributions became too different as postseason games began to alter the expected outcomes, plus this way we were able to account for teams that were on streaks or sudden performance changes within the teams. For our player prop models, we ensured strong distributions by dividing players into tiers based on skill level and number of games played. This allowed us to train on enough data to generate accurate predictions for players of similar skill levels. For example, rather than generating a singular model for LeBron James, another model for James Harden, a third for Jayson Tatum, and so on, we can predict each of these players performance just as accurately by lumping their data into what we called the A tier and only need one model to encapsulate players that fit the A tier criteria. Plus, this way we avoided overestimating performance for average players, or underestimating performance for great ones, since they are all been put together into groups of players with similar skill levels.

Another key step in building our machine learning models was selecting

which predictors we wanted to consider and then quantifying which were the most influential predictors on our models. For our player performance models we looked at several indicators over the player's most recent four games to get a trend instead of just a one game performance indicator, since players can have an off day and then set personal records the next night. For every model, from each of the previous four games, we considered the minutes played, steals, fouls, turnovers, blocks, and player position. Then depending on the statistic and model, we could add or remove predictors depending on what we are trying to accomplish. For example, some point-model-specific predictors include the previous four points scored, rebounds, and assists. For the assists and rebounds models we considered the base set of predictors as well as the previous four assists and rebounds respectively. For all our models the minutes played and the statistic in question tended to be the most indicative predictors for future performance.

To quantify our models' performance we looked at the root mean square error (RMSE: a measure of quality of fit), and standard deviation of our prediction error to give us a performance evaluation. The standard deviation of the actual score variation can then be interpreted as the error that would be incurred if DraftKings were to simply predict the average score each time. This served as our base case for comparison and if our prediction error had a lower standard deviation than the base it means we are improving the prediction. In the end we were able to significantly reduce the standard deviation of the error in all of our models. To name a few, our game line prediction error was reduced by 39.48%, A Tier point prediction error was reduced by 9.57%, A Tier assists by 7.86%, and A Tier rebound error was reduced by 12.8%.

# 4 Introduction

Since its founding in 2012, DraftKings has been widely known as the premier arena for daily fantasy sports and sportsbook operations. As part of its sportsbook operations, DraftKings offers a variety of lines, or point-spreads which set the odds for which team is favored in a game, as well as expected performance by players in key statistics, along with the expected pay out if the user bets on the correct odds. Prior to this project, DraftKings set many of its player prop lines from outside sources, a practice the company hopes to move away from. One of the main driving forces towards internalizing the line generation is the cost incurred from buying lines from so-called "market makers" - including the cost of employing people to constantly keep up to date on changes to the market and adjust lines as necessary. This form of line setting is also risky as the information and decisions that went into generating the lines being used remains with another company. Since DraftKings was receiving important information about line changes at a delay, they were at risk of losing a significant amount in losses. By controlling the lines they offer, DraftKings would be able to minimize the risk they face by splitting bets as evenly as possible between both sides of the line.

In this project, we used machine learning, which combines mathematics, statistics, and computer science to generate machine learning models that create predictions on future events based on data collected from past events. The goal of this project was to develop predictive models for DraftKings to implement when setting their player performance lines in hopes of providing the best starting line possible. Using these techniques, we were able to generate lines internally based on historic data, including past betting lines and previous player performance.

To do so we worked through several steps, beginning with getting familiarized with data science, computer science, and statistical learning concepts as well as with the world of sports-betting in general. We then implemented what we learned into building models to predict the closing line for NBA games to ensure we were able to understand and successfully generate predictions before moving onto the more complex player performance models. Along the way we experimented with various model types, training and testing data splits, and teaming schemes among other techniques. In the end, our player models for points, rebounds, and assists performed equally well or better than current industry standards and newly developed models from DraftKings' own data science team. Towards the end of this report, we lay out recommendations for future applications or expansions of the work we began.

# 5 Background

## 5.1 Sportsbooks

### 5.1.1 What Is A Sportsbook?

A sportsbook is a place where a gambler can wager on various sports competitions, including football, baseball, hockey, soccer, and most relevantly, basketball. As a sportsbook, DraftKings offers gamblers a variety of bets. In order for DraftKings to be profitable, DraftKings must find a way to place lines on bets such that approximately 50% of the money is on each side of the bet. This section will explain the different types of bets that DraftKings offers and how DraftKings arrives at their publicly offered lines.

### 5.1.2 Point Spread

One of the most common bets is a point spread. A point spread is "a bet on the margin of victory in a game" [5]. Depending on the perceived difference in quality between the two teams, the point spread will favor the better team by a certain number of points [5]. Bets that are listed negatively imply a favorite, while bets that are listed positively imply an underdog.

This is more easily illustrated in the context of an example. In the figure below, the Memphis Grizzlies are favored to beat the Washington Wizards by three points in a game between the two respective teams on March 10, 2021. There are two equivalent ways to describe this line; Wizards +3 and Grizzlies -3.

| **Bet** | **Odds** |
|---|---|
| Washington Wizards +3 | -109 |
| Memphis Grizzlies -3 | -112 |

Table 1: An example of game line bet. In this example, which was taken from DraftKings' website, we show the game line bets available for the March 10 game between the Washington Wizards and the Memphis Grizzlies.

### 5.1.3 Prop Bet

Another very common bet is a prop bet. A prop bet, also referred to as a player prop, is defined as any bet "pertaining to player statistics or outcomes" [2]. This is more easily discussed in the context of an example. In Figure 2, the prop bet shows an over/under line on how many points Bradley Beal will score in a game between the Washington Wizards and Memphis Grizzlies on March 10, 2021.

| Bet | Odds |
|---|---|
| Over 32.5 points | -115 |
| Under 32.5 points | -106 |

Table 2: An example of a prop bet. In this example, which was taken from DraftKings' website, we show the player prop lines available for how many points Bradley Beal would score in a March 10 game between the Washington Wizards and the Memphis Grizzlies.

#### 5.1.4 Converting Odds to Percentages

For the rest of this paper, it is crucial that the reader understands how to convert odds to percentages. While there are multiple ways to list odds (the main ways being British, European, and American), since most American sportsbooks list all odds in the American way, we will only be discussing odds listed in the American way for the remainder of this report [1].

The American way lists odds as a number that is either positive or negative and whose lowest possible value is 100 [1]. To convert bets to break-even percentages, we will apply

$$\text{Break-even percentage} = \text{Risk} \ / \ (\text{Risk} + \text{Win}) \ [1] \ ,$$

Bets that are listed as negative imply that a bettor on the favorite [1]. For example, suppose a bettor took a bet listed at -200. This means that the bettor would have to risk \$200 to win \$100. Our risk would then be equal to 200 and our win would then be equal to 100. When we apply the break-even percentage equation, we get (5.1.4).

$$\text{BE } \% = \text{Risk} \ / \ (\text{Risk} + \text{Win}) = 200/(200 + 100) = 66.7\% \ ,$$

Bets listed as positive imply that a bettor is betting on the underdog [1]. For example, suppose that the bettor took a bet listed at +200. This means that the bettor would have to risk \$100 to win \$200. Our risk would then be equal to 100 and our win would then be equal to 200. When we apply the break-even percentage equation, we get (5.1.4).

$$\text{BE } \% = \text{Risk}/(\text{Risk} + \text{Win}) = 100/(100 + 200) = 33.3\% \ ,$$

#### 5.1.5 Calculating Break-even Percentages

Another crucial concept is break-even percentages. A break-even percentage "is the percentage of time a bet must win for you to neither win nor lose money making the bet over time" [1]. Break-even percentages are based on how much the bet pays, not how often the bet wins [1].

In other words, if you were to bet on the result of a coin flip, your break-even percentage would be 50%. That is because you would need a bet to pay as much

as you risked (i.e. bet $10 to win $10) since the odds of a coin landing on heads (or tails) is exactly 50%.

If you were to bet on a coin flip at better than even odds (i.e. bet $10 to win more than $10), you would be taking a bet that wins at a higher percentage than the break-even percentage. Therefore, you would be taking a winning bet.

Similarly, if you were to bet on a coin flip at worse than even odds (i.e. bet $10 to win less than $10), you would be taking a bet that wins less often than the break-even percentage. Therefore, you would be taking a losing bet.

For the remainder of this paper, when evaluating or discussing bets, we will be converting odds to break-even percentages.

#### 5.1.6 Calculating the Hold

The way that sportsbooks make money is through the hold, which is "the spread between what [a sportsbook] will buy a bet for and what [a sportsbook will] sell it for" [1]. In the long run, sportsbooks want bets evenly distributed on both sides of the bet since that will allow the book to generate dependable revenue via the hold [1].

It is best to explain with an example. Let's return to the over/under line on how many points Bradley Beal will score in a game between the Washington Wizards and Memphis Grizzlies on March 10, 2021. Using the break-even percentage formula above, we deduce that the bettor's break-even percentage for this bet is 53.5% when betting that Beal will score over 32.5 points and 51.5% when betting that Beal will score under 32.5 points. From the sportsbook's perspective, that means that the break-even percentage is 46.5% on Beal scoring over 32.5 points and 48.5% on Beal scoring over 32.5 points. Essentially, the sportsbook is buying the over bet on Beal's points at 46.5% and selling the same bet at 53.5%. Likewise, the sportsbook is buying the under bet on Beal's points at 48.5% and selling the same bet at 51.5%. If gamblers were to spend equal amounts of money on each side of this prop bet, then this bet would be profitable for DraftKings because if sportsbooks "can manage to buy and sell exactly the same amount of this market, they will simply pocket the difference with zero risk" [1]. The hold percentage is the difference between the break-even percentage at which a book sells a bet and the break-even percentage at which a book buys a bet [1]. Since 53.5% - 46.5% = 7% and 51.5% - 48.5% = 3%, in the example used here, DraftKing's hold is 7% on the over and 3% on the under.

#### 5.1.7 Opening and Closing Lines

Using the example already mentioned, the Washington Wizards played the Memphis Grizzlies on March 10, 2021. Following the conclusion of Wizards and Grizzlies' previous games, the line for the Wizards-Grizzlies game is initially set at Grizzlies -2. As the day progresses, bettors evaluate the line and bet accordingly. By the night of March 11th, minutes before opening tip (when the line is no longer available), the line available is Grizzlies -3. In this example,

the opening line would be Grizzlies -2 and the closing line would be Patriots -3. In general, an opening line is the initial line that is made available for betting on a sports event and a closing line is "the final line available before a game actually begins" [3] [4].

### 5.1.8 Market Makers vs. Retail Books

There are two types of business models for books to follow: the market maker model and the retail book model [1]. It is worth noting that no book "will ever operate at either extreme described below," but will instead "fall on a spectrum between the extremes" [1]. Additionally, while a book can be a market maker for one sport, a book can act as a retail book for a different sport [1]. Finally, it is worth noting in the context of this report that when we use the term revenue, we mean "how much the business wins betting against their customers" [1].

A market maker is a book that relies on high volume to drive revenue [1]. In order to drive high volume, market makers place as few limits on gambling as possible, such as higher betting limits, lower hold percentages, and lines that are released as early as possible [1]. Although market makers do receive a higher volume of bets than retail books, market makers' margins tend to be lower than retail books [1].

A retail book operates on an entirely different business model than a market maker. While being a market maker offers much higher "boom" potential, it is also offers a much higher "bust" potential. Many sportsbooks would rather avoid this risk and instead "count on making a profit on each bet sold" [1]. Rather than focusing on high volume with low margins like market makers, retail books instead focus on maintaining high margins [1]. Instead of producing their own lines like market makers, retail books get their lines from a third party, either by copying the lines or by paying for access to a data feed that provides lines [1]. Since the retail book does not know how the lines were produced, "they're vulnerable to any bettor who may have more information about their markets than they do" [1]. This information is "market information like who bet what, when, and why into the market making sportsbook" rather than "inside information about players or coaches involved in the sporting event" [1]. To shield against this risk, retail books "curate their customer pool" and have low betting limits [1].

For the purposes of this report, the most important difference between market makers and retail books is that market makers adjust lines in response to customer action while retail books frequently "do not move their lines on action" [1]. Since DraftKings intends to be a market maker for basketball player prop bets, we are trying to build a machine learning model that will estimate opening lines that approximates closing lines as precisely as possible.

## 5.2 Key Basketball Statistics

For readers to interpret the results, it is important to be familiar with basic basketball statistics. In basketball, points are used to keep track of a game's

score and can be accumulated through two point field goals, three point field goals, and free throws. An assist is a pass from a player to a teammate who scores a two point or three point field goal, while a rebound is attributed to a player who retrieves the ball after a missed field goal or free throw attempt. For more information, a link to a full glossary of statistics can be found here [6].

While using basketball statistic and data science techniques to set player prop lines is not novel, due to the proprietary nature the gambling industry, there is not very much public information available about the best practices. Rather than hand-crafting an expert model, our approach was to use known data from past player and team performances as well as machine learning techniques to optimize prediction accuracy.

## 5.3 Basic Machine Learning Techniques

| Notation | Definition |
|---|---|
| $y$, $\hat{y}$ | Scalars are non-bold and non-italic |
| $\mathbf{z}$, $\mathbf{w}$ | Vectors are lowercase and **bold** |
| $\mathbf{X}$ | Matrices are uppercase and **bold** |
| $\mathbf{X^T}$ | T indicates matrix transpose |
| $\mathbf{X}_{ij}$ | Matrix elements written as scalars |
| $\mathbf{X}_{i,:}$ | Rows of a matrix |
| $\mathbf{X}_{:,j}$ | Columns of a matrix |
| $n$ | Number of examples |
| $p$ | Number of features |

### 5.3.1 What is Machine Learning?

Machine learning is a branch of artificial intelligence that aims to learn from sample data - called training data - and build algorithms, which are "sequence[s] of statistical processing step[s]" [10]. These algorithms "use statistics to find patterns in massive amounts of data" [9]. There are four crucial steps to building a machine learning model: selecting and preparing a training set, choosing an algorithm to run on the training set, training the algorithm to create a model, and using and improving the model [10].

The first step, selecting and preparing a training set, involves choosing "a data set representative of the data the machine learning model will ingest to solve the problem it's designed to solve" [10]. In choosing such a data set, it is important to clean the data prior to usage, which can include removing duplicates, fixing structural errors, filtering out unwanted outliers or unrepresentative data, and handling missing data. Finally, it is important to set aside some of the data to be testing data, where you can assess the accuracy of your model on data that the model has yet to see. In later sections, the report will detail specific steps we took to clean our data.

The second step, choosing an algorithm to run on the training set, consisted of the group choosing a variety of approaches to experiment with what algorithm would be most predictive. In later sections, we will explain linear regression, Lasso regression, random forests, and neural networks in greater detail.

The third step, training the algorithm to create a model, is an "iterative process" that "involves running variables through the algorithm, comparing the output with the results it should have produced, adjusting weights and biases within the algorithm that might yield a more accurate result, and running the variables again until the algorithm returns the correct result most of the time" [10].

The fourth step, using and improving the model, is where we evaluated the accuracy and efficacy of our algorithms by using the model on our testing data and making improvements and adjustments as needed.

In our specific project, utilizing supervised machine learning is beneficial as it allowed the computer to learn from the past to predict the future. For DraftKings, this means using the lines from past games to predict how future betting will look and using past player performance to predict future player performance.

### 5.3.2 Linear Regression

We began by trying to build a linear prediction model for closing lines (and later for the number of points, rebounds, and assists a player scores in a game). Linear regression is a useful but simple approach for predicting a quantitative variable under supervised conditions [7]. Linear regression is also a crucial concept to understand for more complex statistical methods which will be explained later in the background chapter. Given $p$ predictors, we represent multiple linear regression as (13).

$$\mathbf{y} = \beta_0 + \beta_1 \mathbf{X}_{:,1} + \beta_2 \mathbf{X}_{:,2} + ... + \beta_p \mathbf{X}_{:,p} + \epsilon = f(\mathbf{X}, \beta) + \epsilon, \tag{1}$$

where $\epsilon$ represents our model's error since the exact relationship between $\mathbf{X}$ and $\mathbf{Y}$ might not be identical in the training and testing data [7]. Consequently, we represent our estimate of $\mathbf{y}$, $\hat{\mathbf{y}}$, as (2).

$$\hat{\mathbf{y}} = \beta_0 + \beta_1 \mathbf{X}_{:,1} + \beta_2 \mathbf{X}_{:,2} + ... + \beta_p \mathbf{X}_{:,p} = f(\mathbf{X}, \beta), \tag{2}$$

While there are multiple ways to estimate the coefficients in a linear regression model, $\beta_{0,p}$ is most frequently found using ordinary least squares. Ordinary least squares aims to calculate $\beta$ of each variable such that the sum of the squared residuals is minimized [7]. Therefore, $\beta_0$, $\beta_1$, $\beta_2$, ... , $\beta_p$ can be expressed as (3) and (4).

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \mathbf{X}_{:,1} - \hat{\beta}_2 \mathbf{X}_{:,2} \ ... \ - \hat{\beta}_p \mathbf{X}_{:,p}, \tag{3}$$

$$\hat{\beta}_j = \frac{\sum_{i=1}^{n} (\mathbf{X}_{:,j} - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (\mathbf{X}_{:,j} - \bar{x})^2}, \ \textit{where } 1 \leq j \leq p, \tag{4}$$

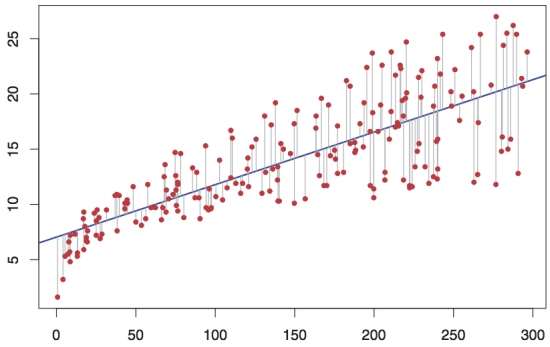


Figure 1: An image of the output of linear regression and the resulting residuals [8]

[7]

For our linear regression models, the value of the $\beta$ is found using the normal equation, where we let $\beta = \{\beta_0, \beta_1, ..., \beta_p\}$. Equation 2, the equation for linear regression, can be represented as (5).

$$\hat{\mathbf{y}} = \beta^T \mathbf{X} \tag{5}$$

Of course, we have yet to discuss how to find the value of the matrix $\beta$. Using the normal equation [17], we find that

$$\beta = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \tag{6}$$

### 5.3.3 Lasso Regression

Lasso regression is a form of linear regression that utilizes shrinkage, which is where data values are shrunk towards a central point [7]. Lasso regression reduces model complexity, prevents overfitting, and assists with variable selection while maintaining much of the predictability of a standard linear regression model.

The way that Lasso regression utilizes shrinkage is that Lasso regression assigns a penalty, $\lambda$, to each additional variable. Then, Lasso regression attempts to minimize

$$\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j \mathbf{X}_{i,j})^2 + \lambda \sum_{j=1}^{p} |\beta_j|, \tag{7}$$

As shown in the picture below, "the least squares solution is marked as $\hat{\beta}$, while the blue diamond... represent[s] the Lasso... regression constraints" [7]. The red ellipses "that are centered around $\hat{\beta}$ represent regions of constant residual sums squared" [7]. Simply put, every point on a given ellipse has the same residuals sum squared. The farther away an ellipse is from the least squares solution, $\hat{\beta}$, the larger residuals sum squared that an ellipse represents.

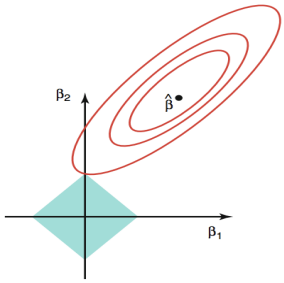


Figure 2: An image of Lasso regression's ability to eliminate variables [7]

Depending upon $\lambda$'s value, Lasso regression can actually eliminate variables from the regression by assigning them a coefficient of zero; this happens when the least squares solution and the constraint region land on a corner of the box representing the constraint region. For example, we would be able to assign $\beta_1$ a coefficient of zero in the picture shown in Figure 4.

After testing several values of $\lambda$ (called $\alpha$ in the python code) thorough trial and error as well as cross validation sets, we settled on the value of 0.1 across the board. Lower values tended to return nearly identical results to the linear model, likely because the number of features we consider was low enough to keep all $\beta$s above zero. Since we wanted to consider both model accuracy and ease of interpretation, we settled on $\lambda = 0.1$ so that we could take advantage of Lasso assigning unimportant features a coefficient of zero. For the remainder of this report, all Lasso models feature this value.

#### 5.3.4 Random Forests

A random forest is a method which builds upon a technique called decision trees. Decision trees, and in our case regression trees specifically, break down the space containing all predictors into smaller segments and make decisions on how testing data will perform based on quantifiers such as the mean or mode of the training data. A simple decision tree can be seen in Figure 3 on the right generated from the splits in the data seen on the right from "Introduction to Statistical Learning." This example tree splits a baseball hitters salary prediction space into three non-overlapping regions, $R_1, R_2, R_3$, called leaves. The internal decisions of the tree on each data point determine which terminal node (leaf) it will end up in.

Each data point in a given leaf receives the same prediction. In this example, the first split can be seen at the top of the tree and is based on whether or not the number of years the player has played is less than 4.5, and the second based on their hits. Where players fall in these regions determines what their predicted salary is as can be seen at the end of their corresponding branch. For example, we can write $R_1 = \{X | \text{ Years} < 4.5\}$, and see that a player who has been playing for under 4.5 years will have a predicted salary multiplier of 5.11 [7].

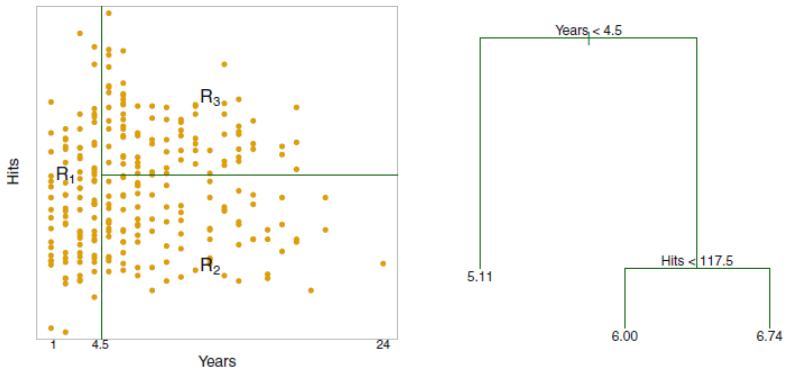


Figure 3: Left: A visual plot of the partition of Players in the data set [7], Right: The resulting tree from this partition [7]

The regions $R_1, R_2, ...R_j$, for a given problem can be determined in any way theoretically, but to obtain the best predictions they should be chosen to minimize the residual sum of squares, or RSS. The RSS sums the squared differences between the mean for region $R_j$, $\hat{y}_{R_i}$ and true value $y_{R_i}$ from all the data in that region, as seen in equation 8 below [7].

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_i})^2, \tag{8}$$

This helps to quantify the quality of fit because a small RSS indicates that predicted values are not far off from the actual result, where as a large RSS reveals large residuals and a poor model fit. Several splits/nodes can be created in this way and then "pruned" to prevent over-fitting.

There are several advantages to working with decision trees that contributed to our use of them in this project. To begin, methods utilizing trees are especially great for data sets which represent highly nonlinear and complicated relationships. Another notable key benefit is the ease of interpretation with this set up of model, even compared to simple linear regression. It is far easier to look at a decision tree and explain what a prediction will be or depend on than to look at a linear model with multiple features and $\beta$s. They also easily handle both quantitative variables and qualitative ones without having to feature engineer as much as in other classical modeling techniques.

Unfortunately, on its own a single decision tree is likely too simple to capture the full complexity of most problems, however prediction accuracy can be greatly improved with additional techniques and approaches to build upon the decision trees. Since an individual tree can vary greatly based on the training data, a technique called bagging, a form of bootstrapping for decision trees, is employed to reduce variance. Rather than taking a single sample with mean $\mu$ and variance

$\sigma^2$ several trees are generated from $n$, independent samples of the training data which reduces the variance to $\frac{\sigma^2}{n}$. These trees generate predictions from the bootstrapped samples and the mean of these predictions is then used to obtain a final result [7]. This can be expressed in equation 9 below,

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} f^b(x), \tag{9}$$

where $B$ is the number of bootstrapped samples, $f^b$ is the model generated from fitting sample $b$, and $f^b(x)$ is the prediction for test data point $x$ using the model generated from sample $b$ [7].

Random forests take this idea one step further. Not only are multiple trees generated from bootstrapped samples, but each tree only considers a subsection of the total predictors. This essentially prevents any one predictor from overpowering the other predictors in the data set. If a particularly strong predictor is considered in every tree in a set of bagged trees, then the variance will not actually be reduced as much as it should be. However, if the trees do not all contain that strong predictor, they can make connections based on some of the other predictors that might contain key information, therefore making the overall prediction more accurate and less varying [7].

### 5.3.5 Neural Networks

In addition to linear regression, Lasso regression, and random forests, we utilized feedforward neural networks when creating models to predict NBA player performance. Neural networks, named because of their representation as the composition of many different functions and their initial inspiration being neuroscience, are extremely important in modern machine learning [11]. Feedforward neural networks, also called multilayer perceptrons (MLPs), are designed so that "information flows through the function being evaluated from $\mathbf{X}$, through the intermediate computations used to define f, and finally to the output $\mathbf{Y}$" [11]. If we assume that the true mapping of $\mathbf{X}$ to $\mathbf{Y}$ is $\mathbf{y}_j = f^*(\mathbf{X}_{:,j})$, then the goal of a neural network is to find a function f with parameters $\theta$ such that $\mathbf{Y} = \mathrm{f}(\mathbf{X}_{:,j};\ \theta)$ [11].

In order to find $\mathrm{f}(\mathbf{X}_{:,j};\ \theta)$, we must revisit the representation of neural networks as the composition of many different functions. For example, say we have three functions $f^1$, $f^2$, and $f^3$ connected in a chain, to form $f(\mathbf{X}_{:,j}) = f^3((f^2(f^1(\mathbf{X}_{:,j}))))$. In this example, $f^1$ is the first layer of the network, $f^2$ is the second layer of the network, and $f^3$ is the third layer of the network. The final layer of the network, $f^3$ in our example, is called the output layer, while $f^1$ and $f^2$ would be the hidden layers, since the training data does not show the desired output for $f^1$ and $f^2$ [11]. The depth of the model is simply how many layers there are in the network.

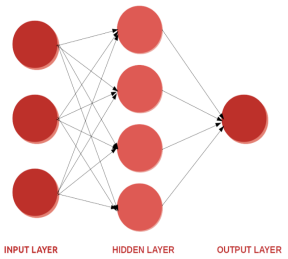


Figure 4: A visual representation of the role of a hidden layer [12]

It is worth noting that "every hidden layer of a model is typically vector-valued" [11]. We refer to the dimensionality of these hidden layers as the width of our model [11]. A helpful interpretation is to think of "each element of the vector ... as playing a role analogous to a neuron" [11]. Instead of thinking of each layer as representing a single vector-to-vector function, it is useful to consider each layer as "consisting of many units that act in parallel, each representing a vector-to-scalar function" where "each unit resembles a neuron in the sense that it receives input from many other units and computes its own activation value" [11]. While "the learning algorithm must decide how to use those layers to produce the desired output," "the training data does not say what each individual layer should do" [11].

Before proceeding any further, a useful way to understand neural networks is to "begin with linear models and consider how to overcome their limitations" [11]. Linear models, such as the linear regression and Lasso regression discussed earlier in the background chapter, are attractive because they are easily and efficiently fit and simple to interpret, but carry the obvious downside of being unable to model nonlinear functions well. This brings about the function of neural networks; "to extend linear models to represent nonlinear functions of $\mathbf{X}$, we can apply the linear model not to $\mathbf{X}_{:,j}$ itself but to a transformed input $\phi(\mathbf{X}_{:,j})$, where $\phi$ is a nonlinear transformation" [11]. The ultimate goal of deep learning is to learn $\phi$ and create a neural network represented by $\mathbf{y} = \mathrm{f}(\mathbf{X}_{:,j}; \theta, \mathrm{w}) = \phi(\mathbf{X}_{:,j}; \theta)^T \mathrm{w}$, where the parameters $\theta$ are used to learn $\phi$ and the parameters w map from $\phi$ $(\mathbf{X}_{i,:})$ to $\mathbf{Y}$.

Though neural networks are more complex and versatile algorithms than the linear models discussed earlier in the paper, building neural networks requires following the same four steps necessary to building any machine learning model: selecting and preparing a training set, choosing an algorithm to run on the training set, training the algorithm to create a model, and using and improving the model [10].

In our neural networks, we utilized the ADAM solver because it is widely used in deep learning and viewed as an effective method for optimizing real neural networks [14].

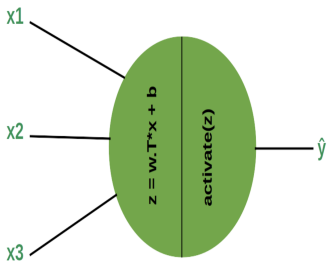


Figure 5: A visual representation of the role of an activation function [12]

Another crucial concept of neural networks is activation functions. An activation function is "a mathematical 'gate' in between the input feeding the current neuron and its output going to the next layer" [12]. Activation functions are useful because they allow a neural network to utilize nonlinear functions in order to better represent the relationship between the training data and the testing data, differentiating a neural network from the linear regression models the group used. The image below shows the role that an activation function plays in a neuron.

Our entire neural network can be represented as [11]

$$f(\mathbf{X}_{i,:}; \mathbf{W}, \mathbf{c}, \mathbf{w}, \mathbf{b}) = \mathbf{w}^T max\{\mathbf{0}, \mathbf{W}^T \mathbf{X}_{i,:} + \mathbf{c}\} + \mathbf{b}, \tag{10}$$

After seeing the results the neural network on the testing data, we quantify the neural network's error using a loss function. Note that what makes this function non-linear is $max\{\mathbf{0}, \mathbf{W}^T \mathbf{X}_{i,:} + \mathbf{c}\} + \mathbf{b}$, where W is the weights of a linear transformation and c the biases and the activation function is applied element-wise [11]. In the deep learning linear, such non-linearity is called an activation function. This particular activation is called ReLu [11].

A loss function is "a measure of how wrong the model is in terms of its ability to estimate the relationship between X and y" that is "typically expressed as a difference or distance between the predicted value and the actual value" [13]. Therefore, the goal of our neural network is to minimize our loss function. We used two loss functions; mean absolute error and categorical cross-entropy. We can represent the mean absolute error loss function as [21]

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}, \tag{11}$$

We used the categorical cross-entropy to represent the error in our neural network when our $\mathbf{y}$ was a vector of probabilities for possibilities of events, where each vector is a bin that represents a possible range of outcomes, rather than a single numerical projection [11]. In order to do this, we used a concept called truth labels [22]. For each y, the bin containing the actual outcome was labeled

1, while every other bin was labeled 0. We could then compare the bins' truth labels to the neural network's prediction of the probabilities of y being in each bin. Mathematically, we can represent categorical cross-entropy as [22]

$$L_{CCE} = -\sum_{i=1}^{m} t_i log(p_i), \tag{12}$$

where m is the number of bins, $t_i$ is the truth label and $p_i$ is the Softmax probability for the $i^{th}$ class. For more information, interested readers can learn more about mean absolute error here and the categorical cross-entropy here.

# 6 Methods

## 6.1 Data Set

DraftKings provided us NBA data ranging between the 2012 and 2019 seasons. The data came in the form of multiple CSV files, each containing different types of data. The first file given to us that we used was called Player.csv, which contained all the names of the teams and players, as well as their position number. Next, we used the TeamSchedule.csv file, in which each game was listed with the ID, the Home team ID, Away team ID, the date played, and the score. The Team.csv gave us a compact list of all the teams with their abbreviation, city, and team ID. Another very important file was the Statistics.csv file which contained all of the statistics with each statistic's ID, abbreviation, and description. This file was very useful for us when we got into picking predictors that we used for predicting player props because we were able to see a comprehensive list of what was available to us. The last, and most important, file given to us was the PlayerBoxScore.csv file. This file contained all of the actual game information, statistics, scores, and more for every team and player. It tied together all of the other files that we used from DraftKings and made it so we could make predictions using this information.

When working on team game lines, we decided to focus on a singular season's data and settled on working with the 2018-2019 season's data. The data provided to us by DraftKings did not have data on the spreads for past games. To access data with the past game spreads, we went to Sportsbook Reviews Online and found a data set with data we needed.

When we began working with player data to create player prop models, the main file we used was the PlayerBoxScore.csv file. This contained all the data given to us from DraftKings starting from the 2015 season until the beginning of the 2020 season, including both player and team statistic scores for all the games listed in the TeamSchedule.csv. We also cross referenced the information in PlayerBoxScore.csv with information in Statistics.csv file in order to tell what each statistic ID represented.

## 6.2 Game Line Predictions

### 6.2.1 Game Data Cleaning

While going through the data provided to us by DraftKings, we realized there were pieces of it we did not need or parts that we needed to isolate. With this knowledge, we decided to create a few tools in Python that would go through the original files and filter out the data as we needed it.

Starting with TeamSchedule.csv given to us by DraftKings, we cleaned out and removed all games where data was completely missing and all the values for the games' statistics were zero. To access the data more easily in our code, we then converted the dates in the CSV from "month/day/year time" to "month—day". For example, a game played on February 10th, 2019 at 5:00 would go from "2/10/19 5:00" to "0210". The team then decided that we would

focus our attention on the 2018-2019 NBA season as it was the most recent fully completed season that was unaffected by COVID-19. Therefore, we removed all the data for games before and after the 2018-2019 season. Next, we decided to move forward with the 2018-2019 data from Sportsbook Reviews Online, which had past game lines in its data. We then took the "TeamScheduleId" for each game from the DraftKings data and assigned them to the corresponding game in the Sportsbook Reviews Online. This allowed us to connect the two data sets and access the game data from DraftKings while iterating through the Sportsbook Reviews Online data. Next, Sportsbook Reviews Online data recorded "pick'em" games, or games where neither team is favored, as "pk." This caused errors as our code expected numerical values instead of text. To fix this, we changed all instances of "pk" to be zero instead. Lastly, added a column into the Sportsbook Reviews Online data to note what game of the season it was for a team. Our model required data from the previous four game for both teams playing, so this added column would tell us if the teams had played enough games to run in the model. The cleaned data was saved into our NBAOddsCleanPk_GameNum.csv which was then broken up into different training and testing CSVs based on what splits we worked with.

### 6.2.2 Selecting Predictors

An important step of building any model is selecting predictors to use in the model. When building our initial model, we relied upon a mixture of data provided to us by DraftKings, data from an outside source containing historical closing line information, and data from social media websites about NBA teams' follower information. For linear regression, Lasso regression, and random forest, the final game lines model, the following predictors were used: average score differential for the home and away teams for the previous four games, the home team and away team's game line for the previous four games, the home and away status of both teams for both the previous four games and the upcoming game (whose game line we were trying to predict), and social media information on both the home and away team from Twitter, Facebook, and Instagram. The social media information was the ratio of both the home team and away team's number of Twitter followers to the Twitter followers of the most popular NBA team, the ratio of both the home team and away team's number of Facebook followers to the Facebook followers of the most popular NBA team, and the ratio of both the home team and away team's number of Instagram followers to the Instagram followers of the most popular NBA team.

We used the average score differential for the home and away teams for the previous four games as a predictor in an attempt to gauge how well the team was playing in recent games, with the idea that recent performance would be predictive of the future performance. We also used the game lines of from both teams from each of the previous four games as a predictor to gauge how previous game lines would affect future betting.

On a related note, we include the home and away status of the home and away teams for both the previous four games and the upcoming upcoming game

(whose game line we were trying to predict) because home court is usually a significant advantage in the NBA, with the home team winning between 56% and 58% of the time [15]. Since we were always trying to predict the game line of the home team, the home/away status of the upcoming game was always set to home.

Finally, we used social media information from Twitter, Facebook, and Instagram information to evaluate the relative popularity of NBA teams among NBA fans. Twitter, Facebook, and Instagram were chosen as those are platforms that are popular among NBA fans and cover different demographics, with Facebook generally having an older user base than Twitter or Instagram. We input the predictors as ratios between a team's number of followers and the number of followers from the most popular team on the respective platform in order to give observers a better view of the relative popularity of various NBA teams. The social media information was primarily added to test how important a team's popularity was to that team's game line.

### 6.2.3 Baseline

We began building out models by writing a function, which we called our baseline function, that would read in the csv file(s) containing the predictors we planned on using in our models and extracting/formatting the data we were actually interested in. For game line predictions, we wanted the baseline function to extract the home and away team, their last four game scores, and the last four point differentials (how much the team won or lost the game by) for each point of data (a game). We also pulled out our response variable, the actual closed line for that game, for each corresponding entry.

As we got results at this step, we wanted to consider predictors outside of the game in an attempt to better model the human side of sports-betting. To accomplish this, we also incorporated the closed lines of the past four games for each team to show betting trends, and a ratio of social media popularity. The idea behind these was that historic betting and overall popularity may be more predictive of future betting than how the team is actually performing.

### 6.2.4 Initial Predictions

In our initial attempt we decided to predict the spread of a game for the Miami Heat based on data from the previous (2018-2019) season. To do this, we fit a linear regression model in Python. The model took the form of the equation below

$$\mathbf{y} = \beta_0 + \beta_1 \mathbf{X}_{:,1} + \beta_2 \mathbf{X}_{:,2} + ... + \beta_p \mathbf{X}_{:,p} + \epsilon \tag{13}$$

where our predictors, $\mathbf{X} = \{\mathbf{X}_{:,1}, \mathbf{X}_{:,2}, ...\mathbf{X}_{:,p}\}$, were the Heat's scores of the final four games they played in 2019 and a binary variable to represent home versus away status [7]. Our response, $\mathbf{y}$, was the spread of the corresponding game at the close of betting.

Once our simple model was fit, we could input an expected score (the median of historic scores) as well as home versus away status to generate a prediction of the point spread for the Miami Heat.

This setup was able to run correctly in Python, but upon further consideration we realized the prediction was based on assumptions that we could not make, and so we decided to re-frame the inputs to our baseline model. We continued to use a linear regression with input data from both the home team and the opposing team, but adapted the data that was used to generate the model and prediction.

Before, we had used the number of points a team had scored in the previous four matches as a way to measure the team's strength and their opponent's strength. However, this was not an effective piece of input data for a few reasons. The first being that we could not use the points scored in a game to predict the spread simply because the game itself had not happened yet and this would only lead to extrapolation and poor predictions. Secondly, a team could score 90 points in one game, for example, and beat the opponent handily but score 110 points in another game which they lose. To adjust for this we decided to use a team's average point differential as our new baseline for strength.

To calculate this differential we created a function in Python. If we wanted to calculate the Boston Celtics' strength, for example, the function would subtract the Celtics' number of points scored in each of the four games from the number of points that the opposing team scored in each of four games, and then take the sum of those four numbers (point differentials) and find the average.

As mentioned earlier, the model also incorporated game lines from the home team and the away team. In order to retrieve this information, we wrote a function that retrieved game lines for both the home and opposing team's game lines for last four games of the 2018-2019 season.

At this point, the team introduced our rolling baseline model. The rolling baseline model simply modified our baseline model to allow for the pulling of data from the four most recent games, rather than from four set games. This allowed a game in April to have different predictors than a game in December.

After our initial predictions, we wanted to incorporate NBA teams' social media information to see if that would improve our model's performance. Our hypothesis was that a team's popularity on Facebook, Twitter, and Instagram would be predictive of a team's game line. In order to do this, we manually gathered data from Facebook.com, Twitter.com, and Instagram.com.

## 6.3 Player Prop Predictions

The main focus of this project was to generate predictions for player performance props. Once we felt comfortable with predicting game lines, moving into player predictions became much easier.

### 6.3.1 Player Data Cleaning

When we initially started working with the Player data, we began by doing some initial predictions using only a few predictors, such as minutes played and last game points scored. By doing this, we were able to see that some players had various zeros for their statistics that should not have been counted in for their predictions. For example, if someone did not play any minutes in the game, their statistic for minutes played would be 0 along with all their other statistics. After discussing this with DraftKings, we were told that if a player does not touch the court (plays zero minutes) all bets for that player are cancelled and returned. We decided to then clean out the PlayerBoxScore.csv file of all zero games. These zero games included the games where the minutes played equaled zero as well as games where the "Played" statistic equaled zero. After cleaning out all lines where these conditions were met, we continued making predictions with our newly cleaned data. We called the file we used to do this PlayerZeroGamesDataCleaner.py.

Since we were focusing on specific statistics for the Player Props, we created different cleaners for each of the statistics. We made one file for each statistic (Assists, Points, and Rebounds). For each of the statistics, the file would take the PlayerBoxScore.csv file and, for each player, would average the statistic being looked at. It would then rank the players based on their statistic's average. These files would then be used for our player tiers. The files are called AssistsCleaner.py for assists, PlayerTierCleaner.py for points, and ReboundsCleaner.py for rebounds.

### 6.3.2 Tier vs. Individual Models

Once we had working baselines and models, the main question that came up was how to train them. Should an individual model be generated for each player? Was this even possible - would all players have enough games under their belt to properly train a model? For the most elite players - take LeBron James for example - we were able to easily fit a model and train on two thirds of the games he played in and still have nearly 100 games to test on. However, this is not the case for all players.

Initially we were unsure of implementing a "tiered structure" to team comparable players for model training since it made sense that predictions for LeBron James' performance should be made based on LeBron James' previous performances. However, we decided to experiment with a tiered set up in order to allow access to more data for predictions on players that may not be the top of the top but would still warrant a point prop line from DraftKings. This also helps to account for Rookies, players who have been injured and missed several games, or other extenuating circumstances.

It's important to note that to limit the amount of data our baselines and models had to run through we only considered players that averaged around 10 points per game since it was determined by DraftKings that players who score too far below this would not get a prop line anyway. Using this idea of a cut-off

we devised the following tiers: the top 5% of point scorers on average were put into the "A Tier," the next 15% were put in "B Tier," and the next 15% were put in "C Tier." Any players not in any of these models were removed from the data set.

To generate a prediction for specific players, the model type desired was trained on all data for the tier in which that player fell. Going back to our LeBron James example, this means our model was fit with all the data from all A Tier players then predictions were generated based on the new data (LeBron's next game) we put in. This ended up giving us fairly consistent results across all models and tiers - even outperforming the individual model for LeBron. It also presented the advantage that we could begin considering predictors that had been made irrelevant in an individual model such as position - does a guard out score a center?

### 6.3.3 Selecting Predictors

As mentioned during the game line section, an important step of building any model is selecting predictors to use in the model. Due to data quality issues, we were limited by the data available in the DraftKings data set when selecting predictors. When building our initial model, we relied upon individual player statistics that were in the data provided to us by DraftKings. For the points, assist, and rebound models, these predictors included the following predictors for each of the previous four games: minutes played, assists, rebounds, steals, blocks, turnovers, and fouls. We also included points scored as a predictor in the player points model. Additionally, for all player prop models, we included the player's position as a predictor.

For all player prop models, minutes played is a good way to predict how much a player will play. This is important since a player will obviously be unable to accumulate points, assists, or rebounds if they are not playing.

Since a basketball team has five players on the court at the same time, we included assists in all models since assists are generally a good way to assess how involved a player is in his team's offense. For the assists model, in particular, we once again figured that the best descriptor of a player's ability to accumulate assists would be the number of assists a player had in previous games.

Rebounds were included as a predictor because we thought that rebounds were a good proxy for whether a player's game was more focused near the rim or around the perimeter (especially on defense, since most rebounds are defensive rebounds). This could help us have the model adjust for a player's role on the team. For the rebounds model, in particular, we once again figured that the best descriptor of a player's ability to accumulate rebounds would be the number of rebounds a player had in previous games.

While defensive statistics in basketball are not nearly as detailed as offensive statistics, blocks and steals are the best approximation we have of a player's defensive prowess. While this is most important for the rebound models (since a player can get a rebound on defense), we thought it would be best to include some defensive statistics in all of our models in order to find the correlation be-

tween a player's defensive skill (represented by blocks and steals) and a player's skill at scoring points or getting assists or rebounds.

Turnovers and fouls were included as an attempt to model previous negative play. The idea here was to attempt to have the model pick up on bad performance to anticipate a player's minutes possibly decreasing in the upcoming game. Admittedly, this was a flawed metric since players who play more minutes tend to accumulate more fouls and players who handle the ball more tend to accumulate more turnovers.

We included points scored as a predictor in the points model since we figured that the best descriptor of a player's scoring ability would be the number of points a player has scored in previous games.

Additionally, we used a player's position as a predictor to explore the relationship between a player's position and the number of points, assists, and rebounds a player accumulates. The table below explains how we labeled player positions, using a position numbering system that is very popular among basketball players, fans, and coaches.

| **Position** | **Number** |
|---|---|
| Point Guard | 1 |
| Point Guard/Shooting Guard | 1.5 |
| Shooting Guard | 2 |
| Shooting Guard/Small Forward | 2.5 |
| Small Forward | 3 |
| Small Forward/Power Forward | 3.5 |
| Power Forward | 4 |
| Power Forward/Center | 4.5 |
| Center | 5 |

Table 3: A table containing the values used in our models for player positions in Basketball. Some players play more than one position. In those cases, we took the average of the two positions. This results in the values above.

When selecting predictors, we initially incorporated the limited team data and more detailed player data from DraftKings. However, as we will discuss further in the Results chapter, we found that including more data actually caused our linear regression model to fail by creating linearly dependent columns without improving any of our other models.

#### 6.3.4 Baselines

To begin, we used the same model types as for game line predictions - linear regression, Lasso regression, and random forests - to generate predictions for player performance props. Specifically we wanted to predict props for points, rebounds and assists. For each of these statistics, a unique baseline function was written to organize the data going into training the models.

Using the predictors listed in the 6.3.3, we built similar functions to the ones in the game line model in order to retrieve the relevant information. Just like the rolling baseline function that we used for the game line model, the rolling baseline for the player prop models would read in the csv file(s) containing the predictors we planned on using in our models and extracting/formatting the data we were actually interested in, namely the relevant player data from the last four games.

## 6.4 Data Visualization & Analysis

### 6.4.1 Statistics Package

To quantify important aspects of the distributions of our data/predictions, we hand-coded several common statistics including mean, median, mode, and standard deviation. This step allowed us to be more confident in the results when performing quantitative analyses on our predictions and errors. These statistic functions were also called in the scatter-plot and histogram tools we created to visually analyze our data.

Perhaps the most well know statistical measure is the mean. We calculated the mean, also called the average, by giving a list of numbers as input, adding up the total value of all of them, and dividing by the number of entries in the list. This gives us one way to identify a "center" of the data. Another way to find a center of the data is the median. This can be found by arranging all numbers in a list in ascending order then selecting the middle value. If the list is comprised of an even number of entries, the middle two can be averaged to calculate the median.

Another important measurement is the standard deviation which lets us understand how spread out the data is. This is because the standard deviation represents the average amount that values in the list differ from the mean. To calculate the standard deviation, the mean is subtracted from each value in the list and the resulting difference is squared. Each of these terms is then added up and the sum is divided by the number of entries again. The square root of this values gives the final standard deviation.

Our last statistical quantifier is the mode which is simply the most commonly appearing value in the list.

### 6.4.2 Scatter-plots

To help determine what might be useful predictors in our models, we utilized scatter-plots to visually identify linear correlations. For example, one of the initial predictor we wanted to test was if a teams previous point differential was closely correlated with its upcoming closing lines. To look for this potential correlation, we created a scatter-plot where the x-coordinate of the points represent the closed spread of the game and the y-coordinate is the previous point differential. The resulting scatter-plot can be seen in Figure 6, surprisingly the correlation was not as strong as expected.

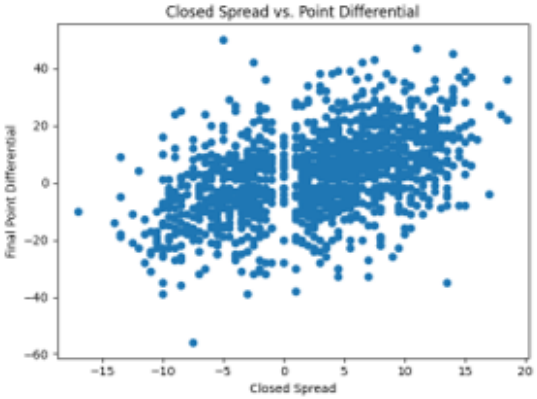


Figure 6: A scatter-plot showing the correlation between closed spread and point differential.

Despite this proving to be a less-than-useful predictor, it helped prompt us to investigate other predictors like previous closed lines - for example, are past betting trends more indicative of future trends than actual game results?

We also used this scatter-plot tool to help visualize how well our models ended up predicting player performance. An example of this can be seen in Figure 17 in the Results section as well.

### 6.4.3 Histograms

To look at the distributions of our predictions, the actual scores or lines, and our prediction error - which we defined to be the actual score/line subtracted from our prediction - we created a tool to generate various histograms. The height of each bar is determined by the number of data points that fall in that bin, allowing us to see how our data, predictions, and errors are distributed. We also incorporated the statistics functions we coded into our histograms by plotting vertical lines where the mean and median were, as well as a set of vertical lines that showed one standard deviation above and below the mean. The bin with the most data points was also highlighted green to further emphasize the mode.

Ideally we were looking for Gaussian distributions in our predictions with small standard deviations, and a mean/median close to zero for our errors.

## 6.5 Optimization

Because of the size of the files that we were working with and the amount of data we would have to comb through, optimizing run time and structured data was very important to us.

### 6.5.1 Pandas and Data Frames

One of the best ways we began trying to optimize our performance and code was by using the Pandas python package and its built in Data Frames[18]. We were able to read in our large .csv files in a single line and transform them into large tables that were easy to iterate through and get information from. Pandas has a built-in function that reads a csv file into a pandas.DataFrame. The Pandas Data Frames are like a table and are able to be iterated through by row or column. We used the Data Frames to iterate by row and get the information stored in each column for each row. This was more efficient because each row was like a different game or a different player, allowing us to quickly go through the games instead of using for-loops.

Pandas also has another built-in function that was very helpful for getting through large amounts of data quickly. This function is the .loc function. This function takes in a condition that is being looked for within the Data Frame and returns a new Data Frame containing only lines where the condition is true. This was especially useful for the Player Props because we were able to get only the lines in the PlayerBoxScore.csv file where the PlayerId was the one we were focusing on. This cut down run time since we no longer needed to use for-loops and check every line for a condition. Using methods like this allowed us to cut down the run time of our Tier Player Props from multiple hours to less than one hour.

### 6.5.2 Data Restructuring

Along with using Pandas, we also decided to restructure the data that DraftKings gave us. In the original DraftKings data given to us in the PlayerBoxScore.csv file, every 10-15 lines would be all the information for one player for one game. This is because each line would be for one statistic and that statistsics score for the game. Figure 7 shows an example of how the PlayerBoxScore was originally organized. Notice how all of the lines in the figure are all from the same TeamScheduleID, meaning from the same game. This caused our run time to be longer since we had to iterate through more lines to get information for the predictions.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | | SportId | PlayerId | TeamSch | StatisticId | Score |
| 240040 | 240038 | 4 | 3230 | 5479153 | 22 | 1 |
| 240041 | 240039 | 4 | 3230 | 5479153 | 23 | 7 |
| 240042 | 240040 | 4 | 3230 | 5479153 | 24 | 11 |
| 240043 | 240041 | 4 | 3230 | 5479153 | 25 | 1 |
| 240044 | 240042 | 4 | 3230 | 5479153 | 26 | 5 |
| 240045 | 240043 | 4 | 3230 | 5479153 | 27 | 2 |
| 240046 | 240044 | 4 | 3230 | 5479153 | 28 | 0 |
| 240047 | 240045 | 4 | 3230 | 5479153 | 29 | 1 |
| 240048 | 240046 | 4 | 3230 | 5479153 | 30 | 3 |
| 240049 | 240047 | 4 | 3230 | 5479153 | 31 | 0 |
| 240050 | 240048 | 4 | 3230 | 5479153 | 32 | 17 |
| 240051 | 240049 | 4 | 3230 | 5479153 | 33 | 34 |
| 240052 | 240050 | 4 | 3230 | 5479153 | 34 | 2 |
| 240053 | 240051 | 4 | 3230 | 5479153 | 35 | 2 |
| 240054 | 240052 | 4 | 3230 | 5479153 | 211 | 1 |
| 240055 | 240053 | 4 | 3230 | 5479153 | 212 | 0 |
| 240056 | 240054 | 4 | 3230 | 5479153 | 213 | 0 |
| 240057 | 240055 | 4 | 3230 | 5479153 | 214 | 2052 |
| 240058 | 240056 | 4 | 3230 | 5479153 | 955 | 0 |
| 240059 | 240057 | 4 | 3230 | 5479153 | 956 | -7 |

Figure 7: A screenshot of the original PlayerBoxScore.csv file for one of the players. It includes the PlayerId, TeamScheduleId, StatisticId, and Score where the Score field refers to amount of the statistic occurred for that game

We decided to rearrange the information in the PlayerBoxScore.csv file so that each game would only take up a single line and that all the statistics would be in columns. This allowed us to get all the data from one game for one player in one line instead of having to iterate through 10-15 lines per game. As seen in Figure 8, each game only requires us to look at one line to get all of the information. This cut down run time immensely. We were able to get multiple predictors at once using this method. For example, if we needed rebounds, points, and assists scored in the 3rd most recent game to the one we were looking at, we were able to just find the TeamScheduleId that made this true and get the scores for statistics 22, 32, and 27.

| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | PlayerId | TeamScheduleId | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 10194 | 10192 | 3230 | 5479153 | 1 | 7 | 11 | 1 | 5 | 2 | 0 | 1 | 3 |
| 10195 | 10193 | 3230 | 5479177 | 3 | 2 | 10 | 1 | 6 | 2 | 0 | 2 | 2 |

Figure 8: A screenshot of the new restructured data for one of the players. It includes the PlayerId, TeamScheduleId, and all the statistics that are available to us through the Statistics.csv that appear in the PlayerBoxScore.csv.

## 6.6 Training/Testing Splits

When doing any sort of machine learning, it is crucial to divide the data into sets of training and testing data. We use the training data to train the machine learning algorithms, generating a model that best fits the data points in the training data. Since we are using supervised learning, meaning we know the dependent variables in our training (and testing) data points, we allow the model to see these and learn from the dependent variable in the training data.

Once the model is fit, we used the testing data to make sure the model is able to accurately calculate predictions on another set of data that it has not seen. Since the model was not fit using the testing data, we can feed the independent variables of this set into the model and generate predictions for each entry. Then, using the actual results, we can use various statistical measurements to compare the predicted outcomes with the true outcomes and quantify the model's performance.

An important thing to note when splitting data into training and testing sets is the distributions of the resulting splits. If the distributions of the new data sets differs too greatly, the model fit on a set of training data will generate very poor predictions on the testing set. Over the course of this project, our team brainstormed and tried multiple data splits, which will be discussed in further detail in a later section.

Once a training/testing split is decided on, there comes the question of what type of model to fit. Our team looked at several types of models and was able to use the resulting predictions from our testing data to quantify how each performed. We began with a simple linear regression model to verify that the data was actually predictable to some degree, then moved on to Lasso linear regression models, random forests, and lastly a feedforward neural network.

# 7 Results

This section will cover our results both both our game line and our player props models. We begin by discussing different splits we experimented with for our training and testing data. Later on, we discuss our model's results, as well as what we learned from the linear and Lasso regressions' predictor coefficients and the random forest model's feature importance. Finally, we discuss other methods we tried, including normalization, the addition of team data in the player props models, breaking up the data by season, and our feedforward neural network that predicted A Tier points.

## 7.1 Error Metrics

In order to understand our results, it's important to recall some statistical measurements. First, the RMSE (root mean squared error) is a measure of how close the predicted values are from the true response variable values. The calculation for this is [20]:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{14}$$

adapted from "Introduction to Statistical Learning" by James, et. al. [7], where $n$ is the number of data points, $y_i$ is the actual value for point $i$, and $\hat{y}_i$ is our predicted value of point $i$. A smaller RMSE indicates a better because the predictions are closer to the true values.

Next are the mean and standard deviation, which were briefly explained in the Methods section, applied here to our prediction error. Our prediction error is how far off our prediction was from the actual value. The mean error was near zero for all models, meaning on average we predicted fairly close to the true value. The standard deviation of this error shows how much our predictions tended to differ when they were wrong. We then compared these standard deviations with the standard deviation from the true data.

The standard deviation of the prediction error is calculated as [23]

$$\text{Standard Deviation of Prediction Error} = \sqrt{\frac{1}{n-1}\sum_{1}^{n}(\hat{y}_i - y_i)^2}, \tag{15}$$

In some cases, our standard deviation of the prediction error is larger than the RMSE. Due to Bayes' Correction, we calculated the RMSE by dividing by $n$, as shown in (14), whereas we calculated the standard deviation of the prediction error by dividing by $n-1$, as shown in (15). For more information, interested readers can look at "Introduction to Statistics" by David M. Lane.

The last quantitative measurements we included were the percent reduction in the standard deviation as just mentioned, as well as the RMSE. We calculated this for each model as

$$\%\text{ Reduction} = \frac{\sigma_{base} - \sigma_{model}}{\sigma_{base}}, \tag{16}$$

where $\sigma_{base}$ is the base standard deviation and $\sigma_{model}$ is the standard deviation of the model being considered. The same calculation can be performed for RMSE reduction by replacing $\sigma$s in the formula with the base and model RMSEs respectively. This is an indicator of how much we were able to reduce the base error incurred from simply predicting the average each time.

In future sections it is important to note that our "best" performing models were selected based on standard deviation of the prediction error and of the RMSE reduction rather than having the mean closest to zero. This is because all our model means were sufficiently close to zero and differed between model types by less than 0.1 most often. This would not be enough of a change to drastically alter the actual predictions, therefore we chose to focus on more consistently accurate predictions which is measured by our standard deviation of the error.

For comparison, we created a "base" case for each statistic and tier. These cases were what would happen if one were to simply predict the mean for that data set every time they needed a new prediction. We could then use the base standard deviation to compare and quantify improvements in the amount of error from the "dumb" method. If our model has a lower standard deviation, that means it is more useful in predicting the game line or player performance than guessing the average. In reality, this method was actually smarter than it should have been since the mean for each tier/statistic combination was calculated over the entire data set, resulting in an instance of data snooping and an acausal relationship. This makes it even more impressive when our models were able to out perform the base cases because those predictions were technically "cheating" since they had some information from the future. This also accounts for why the RMSE and standard deviation for our base cases are the same, because the acausal relationship is unbiased.

## 7.2 Training and Testing Data

### 7.2.1 Game Line Data

A key aspect of machine learning is having comparable distributions on your training and testing data. If the distributions are drastically different, a model will make inaccurate predictions for the testing data since it has been trained on an entirely different distribution. For example, if your training data is mostly negative numbers but your testing data is mainly positive numbers, there will be a high prediction error because the model is trained on the negative data and will, therefore, predict a negative value. To resolve this, the data can be split so that predicting negative data, for example, is trained on mostly negative data.

When beginning to generate predictions, we split our full set of game data into training and testing data. The first 80% of the season we used to train the various models, then tested our work on the last 20% of the season. The initial predictions for this testing set were fairly inconsistent and not as accurate as we had been expecting. After trying to change predictors and model types, we tried plotted the training data and the testing data. Doing this visualized the real

problem behind our inaccurate and inconsistent results. By looking at Figure 10, we could tell that the training data had a much more even distribution while the testing data had a more negative distribution. As it was, our testing data took into account much of the NBA post-season which featured the best teams competing in the play-offs. Because of this, the models trained on early season performances were under-performing on the testing games.

To remedy this, we considered a few different approaches to splitting our training and testing data into more comparable sets. A first thought was to train on an entire season and use this to predict the next season. Ultimately, this was not the direction we went in as it would mean waiting an entire season to re-train models and predicting current play-off matches based on preseason data from over a year prior. The next split we discussed was a random 80-20 split or some other random percentage based split. The idea behind this choice was that a random split of the full season would more equally distributed early, mid and late season games. However, we again decided not to pursue this split because it would likely lead to training on future games. For example, if one of our testing games took place in January, we should not be able to train on data from February or March, which may occur with a random split of this nature.

Since we wanted our models to use the past to predict the future, we tried splitting the data by month, using three previous months of game data to predict the next month. For example, to predict closing lines in the month of April, models would be trained on data from January, February, and March. To predict lines in May, models would be re-trained with data from February, March, and the new April data. This not only keeps the models up to date, but it was found to yield much more comparable distributions than the original 80-20 split.

#### 7.2.2 Player Props Data

When we began looking at our player data and started making predictions, we had to first start with what training and testing splits we were going to use. Since we had worked with distributions for the game data, we already were aware of what the distributions needed to look like to get the best results. The main problem we faced with the player data was what data to use for each player. We decided to go with a two-thirds training and one-third testing split. As seen in Figure 9, it was a very even distribution between the two data sets. We found this to be the case across our different tiers and statistics. After working with game lines, we did not have to test out many splits for the player data since the distributions of the 2/3 training 1/3 testing split were so even.

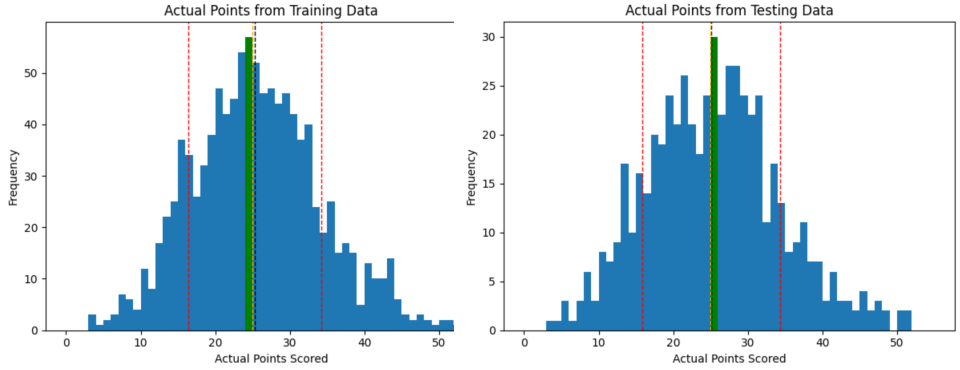


Figure 9: The training (left) and testing (right) distributions for the player props points used in our models for the A Tier. The red dotted lines represent the standard deviation of the data. The black line is the mean and the orange line is the mode.

## 7.3 Game Line Predictions

The first thing our team predicted was closing lines for the home team in NBA games. To predict game lines in the 2018-2019 NBA season, we built linear regression, Lasso regression, and random forest models. In order to improve our models and utilize the best practices in machine learning, we normalized the data DraftKings that provided us. Using the normalized DraftKings data from the home team and opposing team's previous four games. The testing and training distribution we tried first was an 80%-20% split of the 2018-2019 season where the first 80% of games were the training data and the last 20% was the testing data. As seen in the table below, the results of all three models were virtually identical. The worst performing model, a random forest, reduced the standard deviation of the prediction error by 37.45% while the best performing model, Lasso regression, only slightly outperformed random forest by reducing the standard deviation of the prediction error by 39.48%.

| **Method** | **RMSE** | $\sigma$ | **Percent Reduction $\sigma$** | **Percent Reduction RMSE** |
|---|---|---|---|---|
| Linear Regression | 5.21 | 4.25 | 38.32% | 24.38% |
| Lasso Regression | 5.22 | **4.17** | **39.48%** | 24.23% |
| Random Forest | **5.20** | 4.31 | 37.45% | **24.53%** |
| Base | 6.89 | 6.89 | N/A | N/A |

Table 4: Displays the RMSE (Root Mean Square Error), the mean of the error, the standard deviation of the prediction error ($\sigma$), the percent reduction for standard deviation, and the percent reduction for RMSE for each of our models for the Game Line predictions. Base is the actual data given to us from DraftKings. The best performing fields are in bold.

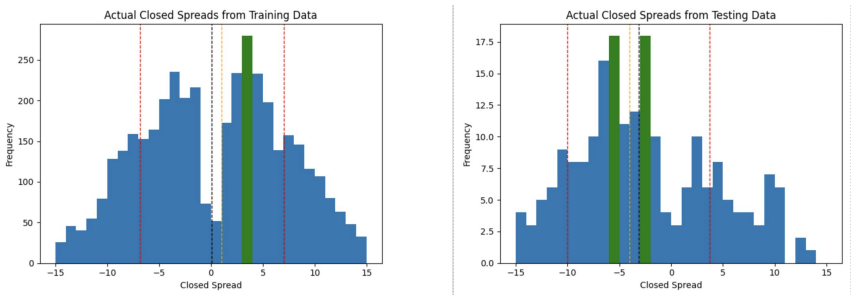


Figure 10: The training (left) and testing (right) distributions for the game lines used in our models. The red dotted lines represent the standard deviation of the data. The black line is the mean and the orange line is the mode.

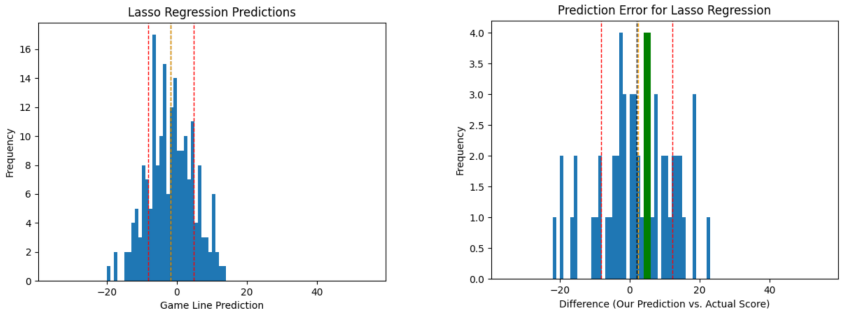


Figure 11: The predictions (left) and prediction error (right) for our Lasso Linear Regression game line model. The red dotted lines represent the standard deviation of the data. The black line is the mean and the orange line is the median. If the black line is not visible, the mean and median are equal or very close.

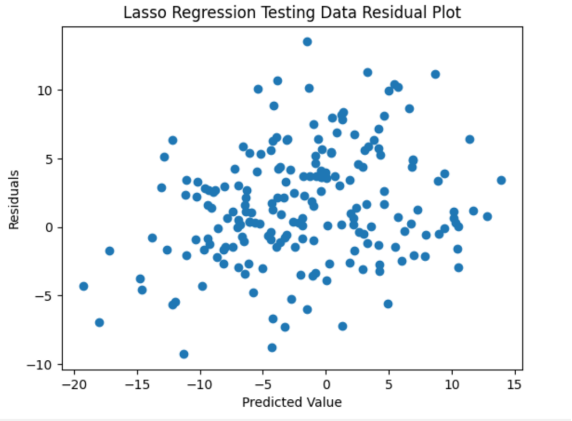


Figure 12: A scatter-plot showing the residuals of the Lasso Linear Regression model for Game Lines.

The team then examined our training and testing data closer and noticed that their closing spread distributions were shaped differently as seen in Figure 10. A possibility for the differently shaped distributions is the fact that the testing data contains the NBA playoffs which skews what teams are playing and how they behave and perform during the games. We then decided to alter our training/testing split to now be where the testing data were all the NBA games played in a particular month and the training data was all the NBA games played during the three months prior. For example, we would have all the games in January be the testing data and all the games in October, November, and December be the training data for the split. Below shows some examples of the training and testing distributions when split by the months.

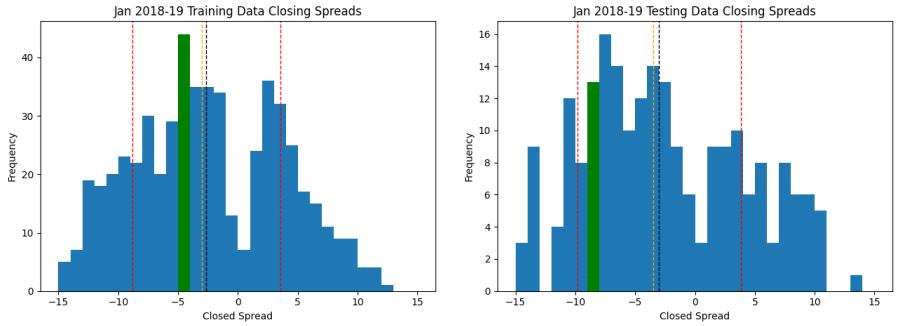


Figure 13: Histograms for training data (left) and testing data (right) game line distributions where testing is on games in January and training is on games in the prior three months (October, November, and December). The red dotted lines represent the standard deviation of the data. The black line is the mean and the orange line is the median.

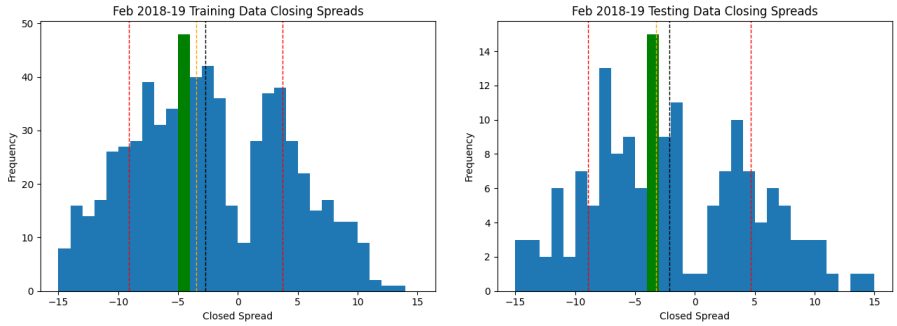


Figure 14: Histograms for training data (left) and testing data (right) game line distributions where testing is on games in February and training is on games in the prior three months (November, December, and January). The red dotted lines represent the standard deviation of the data. The black line is the mean and the orange line is the median.

After redefining the testing and training data, we ran the Linear Regression, Lasso Regression, and Random Forest models again on the new splits. The standard deviation of each month's game lines were around the same as the 80%-20% split (6.89) as they were 6.83, 6.84, 6.94, and 7.43 for January, February, March, and April respectively. The best performing model varied on the month as seen in Table 5. The best performing model on any given month was the Linear Regression model that reduced the standard deviation by 47.11%. The best performing model for January was the linear regression model that had

a 45.68% standard deviation reduction, February's best was both the Linear and Lasso Regression models that had a 29.68% reduction, March's best was the Linear Regression model that had a 47.11% reduction, and April's best was the Random Forest model that had a 43.88% reduction. The models all performed better in the new training and testing splits for January, March, and April except for the Random Forest model in January. The models however performed poorer for February which reduced the standard deviation by less than 30% for all three models. One notable factor that may have contributed to this is that the sample size for February's testing data was only 79 games while January, March, and April had 116, 153, and 60 games respectively.

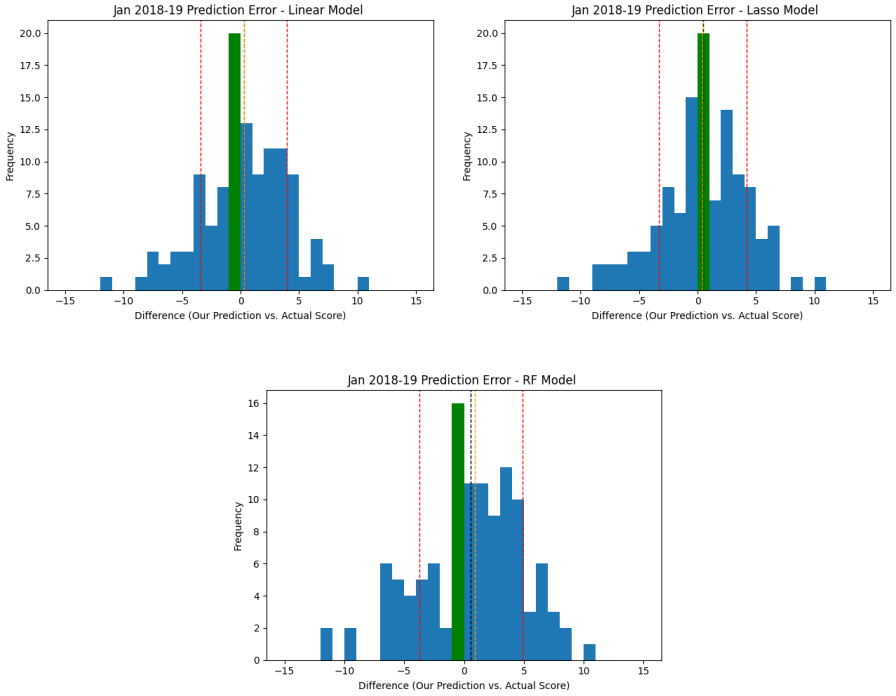


Figure 15: Histograms containing the prediction error for our Linear Regression, Lasso Regression, and Random Forest models using the January month training/testing splits. The red dotted lines represent the standard deviation of the data. The black line is the mean and the orange line is the median.

| Month | Method | RMSE | Mean of the Error | $\sigma$ | Percent Reduction $\sigma$ | Percent Reduction RMSE |
|---|---|---|---|---|---|---|
| January | Base | 6.83 | N/A | 6.83 | N/A | N/A |
| January | Linear Reg. | **3.71** | **0.28** | **3.71** | **45.68**% | **45.68**% |
| January | Lasso Reg. | 3.78 | 0.45 | 3.78 | 44.66% | 44.66% |
| January | Random Forest | 4.35 | 0.58 | 4.33 | 36.60% | 36.31% |
| February | Base | 6.84 | N/A | 6.84 | N/A | N/A |
| February | Linear Reg. | 5.05 | -1.62 | **4.81** | **29.68**% | 26.17% |
| February | Lasso Reg. | **4.96** | -1.36 | **4.81** | **29.68**% | **27.48**% |
| February | Random Forest | 5.15 | **-1.34** | 5.01 | 26.75% | 24.71% |
| March | Base | 6.94 | N/A | 6.94 | N/A | N/A |
| March | Linear Reg. | **3.67** | **0.30** | **3.67** | **47.12**% | **47.12**% |
| March | Lasso Reg. | 3.72 | 0.52 | 3.69 | 46.83% | 46.40% |
| March | Random Forest | 3.81 | 0.45 | 3.79 | 45.39% | 45.10% |
| April | Base | 7.43 | N/A | 7.43 | N/A | N/A |
| April | Linear Reg. | 4.89 | **1.94** | 4.52 | 39.17% | 34.19% |
| April | Lasso Reg. | 4.96 | 2.30 | 4.43 | 40.38% | 33.24% |
| April | Random Forest | **4.88** | 2.58 | **4.17** | **43.88**% | **34.32**% |

Table 5: Shows the RMSE, Standard Deviation of the Prediction Errors ($\sigma$), the Percent Reduction in Standard Deviation, and the Percent Reduction in RMSE for each month split. The bold numbers are the best performing model for each month and each category.

## 7.4 Player Models

Due to a lack of historical player prop line data, we built models that predicted the actual number of points, rebounds, and assists that players would have rather than the player prop lines themselves. DraftKings will be able to use that information to help create player prop lines in the future via an equation they have which can transform predicted performance to expected or estimated lines.

To predict player prop lines from January 2015 until October 2020, we built linear regression, Lasso regression, and random forest models, as well as a feed-forward neural network for our A Tier point scorers. The vast majority of player prop bets that DraftKings receives is on A Tier players, which would include headlining NBA stars such as LeBron James and James Harden. Accordingly, when creating models, we prioritized our A Tier results, and especially our results for A Tier points.

### 7.4.1 Points

**A Tier Points**

| Method | RMSE | Mean of the Error | $\sigma$ | Percent Reduction $\sigma$ | Percent Reduction RMSE |
|---|---|---|---|---|---|
| Linear Regression | 8.54 | 0.46 | 8.61 | 9.46% | 10.12% |
| Lasso Regression | 8.54 | 0.45 | 8.61 | 9.46% | 10.12% |
| Random Forest | **8.52** | 0.45 | **8.60** | **9.57%** | **10.41%** |
| Neural Network | 11.34 | -8.52 | 7.48 | N/A | N/A |
| Base | 9.51 | -0.31 | 9.51 | N/A | N/A |

Table 6: Displays the RMSE (Root Mean Square Error), the mean of the error, the standard deviation of the prediction error ($\sigma$), the percent reduction for standard deviation, and the percent reduction for RMSE for each of our models for A Tier Points. The bold numbers are the best performing model for each of the categories.

While the random forest model performed best by reducing the standard deviation of the prediction error by 9.57%, the results of linear regression, Lasso regression, and random forest models were virtually identical. In fact, all of the models had a RMSE within 0.02 points of each other and had a standard deviation of prediction error within 0.01 points of each other. Our random forest model has a RMSE percent reduction of 10.41% with the other two models not far behind.

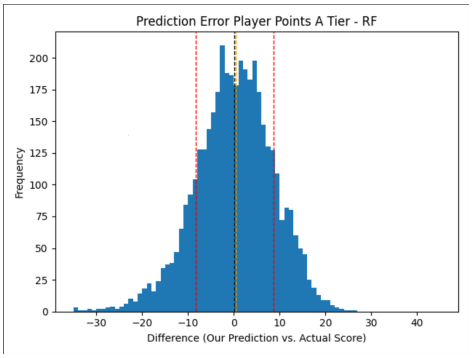


Figure 16: Histogram of the Prediction Error for A Tier Points using Random Forest. The red dotted lines represent the standard deviation of the data. The black line is the mean and the orange line is the median.

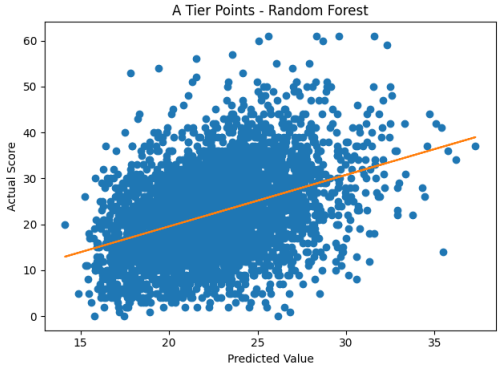


Figure 17: A scatter-plot of the Random Forest Predictions vs. the Actual Points Scored. The orange line in the image shows a slope of approximately 1.12.

Figure 17 shows a scatter-plot of the predictions from our best performing model (the random forest) compared with the actual points in each game. The line in the plot has a slope of 1.12 which indicates a strong, positive correlation between our predictions and the true points scored. Despite such an optimistic slope, as seen in the plot, the points are still a bit all over the place. This is because data points on either side of the line can balance each other out, and goes to show just how difficult these predictions are to make. A player could be set up perfectly going into a game and predicted to score 40 points only to have an off day and score 15. The opposite is also true - they could be coming off a multi-game slump and expect poor performance to instead achieve a new personal high score.

### B Tier Points

| Method | RMSE | Mean of the Error | $\sigma$ | Percent Reduction $\sigma$ | Percent Reduction RMSE |
|---|---|---|---|---|---|
| Linear Regression | **6.64** | **0.0** | **6.72** | **11.46%** | **12.52%** |
| Lasso Regression | 6.65 | 0.01 | 6.73 | 11.33% | 12.38% |
| Random Forest | 6.67 | 0.14 | 6.76 | 10.94% | 12.12% |
| Base | 7.59 | 0.32 | 7.59 | N/A | N/A |

Table 7: Displays the RMSE (Root Mean Square Error), the mean of the error, the standard deviation of the prediction error ($\sigma$), the percent reduction for standard deviation, and the percent reduction for RMSE for each of our models for the B Tier Points. The Base is the data given to us by DraftKings. The bold numbers are the best performing model for each of the categories. The best performing model was Linear Regression.

While the linear regression model performed best by reducing the standard deviation of the prediction error by 11.46%, the results of linear regression, Lasso

regression, and random forest models were very similar. In fact, all of the models had a RMSE within 0.03 points of each other and had a standard deviation of prediction error within 0.04 points of each other. Our linear regression model has a RMSE percent reduction of 12.52% with the other two models being very close.

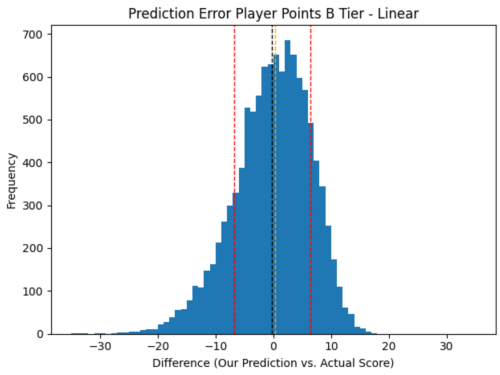


Figure 18: Histogram of the Prediction Error for B Tier Points using Linear Regression. The black line is the mean and the orange line is the median. We calculated our percent error by doing our prediction - the actual score.

### C Tier Points

| Method | RMSE | Mean of the Error | $\sigma$ | Percent Reduction $\sigma$ | Percent Reduction RMSE |
|---|---|---|---|---|---|
| Linear Regression | 5.60 | -0.15 | 5.78 | 8.54% | 11.39% |
| Lasso Regression | **5.59** | **-0.18** | **5.78** | **8.54%** | **11.55%** |
| Random Forest | 5.61 | 0.04 | 5.79 | 8.39% | 11.23% |
| Base | 6.32 | 3.69 | 6.32 | N/A | N/A |

Table 8: Displays the RMSE (Root Mean Square Error), the mean of the error, the standard deviation of the prediction error ($\sigma$), the percent reduction for standard deviation, and the percent reduction for RMSE for each of our models for the C Tier Points. The Base is the data given to us by DraftKings. The bold numbers are the best performing model for each of the categories. The best performing model here was Lasso Regression.

While the Lasso regression model performed best by reducing the standard deviation of the prediction error by 8.54%, the results of linear regression, Lasso regression, and random forest models were very similar. In fact, all of the models had a RMSE within 0.02 points of each other and had a standard deviation of prediction error within 0.01 points of each other. Our Lasso regression model had a RMSE percent reduction of 11.55% while the other models were very close to this.

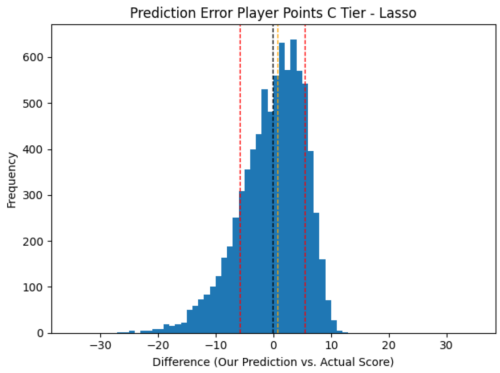


Figure 19: Histogram of the Prediction Error for C Tier Points using Lasso Linear Regression. The red dotted lines represent the standard deviation of the data. The black line is the mean and the orange line is the median.

### 7.4.2 Assists

**A Tier Assists**

| Method | RMSE | Mean of the Error | $\sigma$ | Percent Reduction $\sigma$ | Percent Reduction RMSE |
|---|---|---|---|---|---|
| Linear Regression | **2.91** | **0.14** | **2.93** | **7.86%** | **28.78%** |
| Lasso Regression | **2.91** | **0.13** | **2.93** | **7.86%** | **28.78%** |
| Random Forest | 2.95 | 0.19 | 2.97 | 6.60% | 7.23% |
| Base | 3.18 | 0.11 | 3.18 | N/A | N/A |

Table 9: Displays the RMSE (Root Mean Square Error), the mean of the error, the standard deviation of the prediction error ($\sigma$), the percent reduction for standard deviation, and the percent reduction for RMSE for each of our models for the A Tier Assists. The Base is the data given to us by DraftKings. The bold numbers are the best performing model for each of the categories. The best performing model was both Linear and Lasso Regression.

For A Tier assists, our linear regression and Lasso regression models performed identically, reducing the standard deviation of the prediction error by 7.86% and reducing the RMSE by 28.78%. Likewise, the results of our random forest model were very similar to the results of our linear regression and Lasso regression models. In fact, all of the models had a RMSE within 0.04 assists of each other and had a standard deviation of prediction error within 0.04 assists of each other.

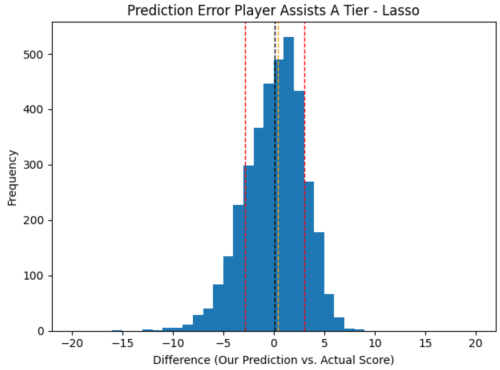


Figure 20: Histogram of the Prediction Error for A Tier Assists using Lasso Linear Regression. The red dotted lines represent the standard deviation of the data. The black line is the mean and the orange line is the median. We calculated our percent error by doing our prediction - the actual score.

## B Tier Assists

| Method | RMSE | Mean of the Error | $\sigma$ | Percent Reduction $\sigma$ | Percent Reduction RMSE |
|---|---|---|---|---|---|
| Linear Regression | **2.22** | **-0.16** | **2.24** | **7.05%** | **7.88%** |
| Lasso Regression | 2.22 | -0.17 | 2.25 | 6.64% | 7.88% |
| Random Forest | 2.23 | -0.09 | 2.27 | 5.80% | 7.47% |
| Base | 2.41 | 1.31 | 2.41 | N/A | N/A |

Table 10: Displays the RMSE (Root Mean Square Error), the mean of the error, the standard deviation of the prediction error ($\sigma$), the percent reduction for standard deviation, and the percent reduction for RMSE for each of our models for the B Tier Assists. The Base is the data given to us by DraftKings. The bold numbers are the best performing model for each of the categories. The best performing model was both Linear and Lasso Regression.

For B Tier assists, the actual standard deviation of the baseline prediction error was 2.41 assists. Linear regression was the best performing model, with a standard deviation of the prediction error of 2.24 assists and RMSE of 2.22 assists. The linear regression model reduced the standard deviation of the prediction error by 7.05% and reduced the RMSE by 7.88%. The results of the other models were very close.

## C Tier Assists

| Method | RMSE | Mean of the Error | $\sigma$ | Percent Reduction $\sigma$ | Percent Reduction RMSE |
|---|---|---|---|---|---|
| Linear Regression | **1.64** | **-0.2** | **1.72** | **6.01%** | **10.38%** |
| Lasso Regression | 1.65 | -0.21 | 1.74 | 4.92% | 9.83% |
| Random Forest | **1.64** | **-0.13** | **1.72** | **6.01%** | **10.38%** |
| Base | 1.83 | 1.60 | 1.83 | N/A | N/A |

Table 11: Displays the RMSE (Root Mean Square Error), the mean of the error, the standard deviation of the prediction error ($\sigma$), the percent reduction for standard deviation, and the percent reduction for RMSE for each of our models for the C Tier Assists. The Base is the data given to us by DraftKings. The bold numbers are the best performing model for each of the categories. The best performing model was both Linear Regression and Random Forest.

For C Tier assists, the actual standard deviation of the baseline prediction error was 1.83 assists. Linear regression and random forest were the best performing models, with a standard deviation of the prediction error of 1.72 assists and RMSE of 1.64 assists. The linear regression and random forest models reduced the standard deviation of the prediction error by 6.01% and reduced the RMSE by 10.38%.

### 7.4.3 Rebounds

**A Tier Rebounds**

| Method | RMSE | Mean of the Error | $\sigma$ | Percent Reduction $\sigma$ | Percent Reduction RMSE |
|---|---|---|---|---|---|
| Linear Regression | **3.90** | **0.19** | **3.95** | **12.80%** | **13.91%** |
| Lasso Regression | 3.91 | 0.19 | 3.97 | 12.36% | 13.69% |
| Random Forest | 3.95 | 0.3 | 4.02 | 11.26% | 12.80% |
| Base | 4.53 | -0.19 | 4.53 | N/A | N/A |

Table 12: Displays the RMSE (Root Mean Square Error), the mean of the error, the standard deviation of the prediction error ($\sigma$), the percent reduction for standard deviation, and the percent reduction for RMSE for each of our models for the A Tier Rebounds. The Base is the data given to us by DraftKings. The bold numbers are the best performing model for each of the categories. The best performing statistic here was Linear Regression.

For A Tier Rebounds, the linear regression model performed best by reducing the standard deviation of the prediction error by 12.80% and reducing the RMSE by 13.91%. Yet, the results of linear regression, Lasso regression, and random forest models were very similar. In fact, all of the models had a RMSE within 0.05 rebounds of each other and had a standard deviation of prediction error within 0.07 rebounds of each other. Also, the percent reductions of the lasso and linear regression models are nearly identical, meaning these two models perform the same for A Tier Rebounds.

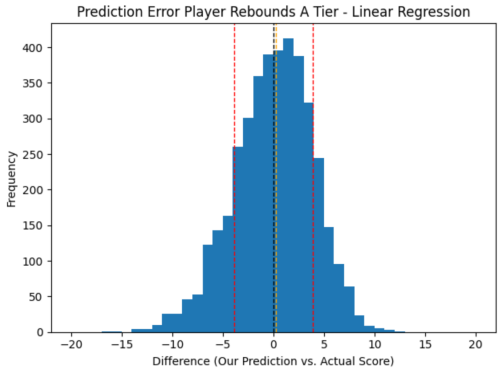


Figure 21: Histogram of the Prediction Error for A Tier Rebounds using Linear Regression. The red dotted lines represent the standard deviation of the data. The black line is the mean and the orange line is the median. We calculated our percent error by doing our prediction - the actual score

### B Tier Rebounds

| Method | RMSE | Mean of the Error | $\sigma$ | Percent Reduction $\sigma$ | Percent Reduction RMSE |
|---|---|---|---|---|---|
| Linear Regression | **2.43** | **-0.02** | **2.56** | **7.65%** | **25.69%** |
| Lasso Regression | 2.93 | -0.01 | 3.03 | 7.34% | 10.40% |
| Random Forest | 2.96 | 0.07 | 3.06 | 6.42% | 9.48% |
| Base | 3.27 | 0.55 | 3.27 | N/A | N/A |

Table 13: Displays the RMSE (Root Mean Square Error), the mean of the error, the standard deviation of the prediction error ($\sigma$), the percent reduction for standard deviation, and the percent reduction for RMSE for each of our models for the B Tier Rebounds. The Base is the data given to us by DraftKings. The bold numbers are the best performing model for each of the categories. The best performing statistic here was Linear Regression.

For B Tier rebounds, the actual standard deviation of the baseline prediction error was 3.27 rebounds. Linear regression was the best performing model, with a standard deviation of the prediction error of 2.56 rebounds and RMSE of 2.43 rebounds. The linear regression model reduced the standard deviation of the prediction error by 7.65% and reduced the RMSE by a stunning 25.69%.

### C Tier Rebounds

| Method | RMSE | Mean of the Error | $\sigma$ | Percent Reduction $\sigma$ | Percent Reduction RMSE |
|---|---|---|---|---|---|
| Linear Regression | **2.43** | **-0.23** | **2.56** | **4.12%** | **8.99%** |
| Lasso Regression | 2.43 | -0.25 | 2.57 | 3.75% | 8.99% |
| Random Forest | 2.45 | -0.16 | 2.58 | 3.37% | 8.24% |
| Base | 2.67 | 1.84 | 2.67 | N/A | N/A |

Table 14: Displays the RMSE (Root Mean Square Error), the mean of the error, the standard deviation of the prediction error ($\sigma$), the percent reduction for standard deviation, and the percent reduction for RMSE for each of our models for the C Tier Rebounds. The Base is the data given to us by DraftKings. The bold numbers are the best performing model for each of the categories. The best performing statistic here was Linear Regression.

For C Tier rebounds, the actual standard deviation of the baseline prediction error was 2.67 rebounds. Linear regression was the best performing model, with a standard deviation of the prediction error of 2.56 rebounds and RMSE of 2.43 rebounds. The linear regression model reduced the standard deviation of the prediction error by 4.12% and reduced the RMSE by 9%.The lasso regression model performed almost identically, reducing the standard deviation of the prediction error by 3.75% and the RMSE by also 9%.

| Category | Method | RMSE | Mean of the Error | $\sigma$ | Percent Reduction $\sigma$ | Percent Reduction RMSE |
|---|---|---|---|---|---|---|
| A Tier Points | Linear Regression | 8.54 | 0.46 | 8.61 | 9.46% | 10.12% |
| A Tier Points | Lasso Regression | 8.54 | 0.45 | 8.61 | 9.46% | 10.12% |
| A Tier Points | Random Forest | **8.52** | **0.45** | **8.60** | **9.57%** | **10.41%** |
| A Tier Points | Neural Network | 11.34 | -8.52 | 7.48 | N/A | N/A |
| A Tier Points | Base | 9.51 | -0.31 | 9.51 | N/A | N/A |
| B Tier Points | Linear Regression | **6.64** | **0.0** | **6.72** | **11.46%** | **12.52%** |
| B Tier Points | Lasso Regression | 6.65 | 0.01 | 6.73 | 11.33% | 12.38% |
| B Tier Points | Random Forest | 6.67 | 0.14 | 6.76 | 10.94% | 12.12% |
| B Tier Points | Base | 7.59 | 0.32 | 7.59 | N/A | N/A |
| C Tier Points | Linear Regression | 5.60 | -0.15 | 5.78 | 8.54% | 11.39% |
| C Tier Points | Lasso Regression | **5.59** | **-0.18** | **5.78** | **8.54%** | **11.55%** |
| C Tier Points | Random Forest | 5.61 | 0.04 | 5.79 | 8.39% | 11.23% |
| C Tier Points | Base | 6.32 | 3.69 | 6.32 | N/A | N/A |
| A Tier Assists | Linear Regression | **2.91** | **0.14** | **2.93** | **7.86%** | **28.78%** |
| A Tier Assists | Lasso Regression | **2.91** | **0.13** | **2.93** | **7.86%** | **28.78%** |
| A Tier Assists | Random Forest | 2.95 | 0.19 | 2.97 | 6.60% | 7.23% |
| A Tier Assists | Base | 3.18 | 0.11 | 3.18 | N/A | N/A |
| B Tier Assists | Linear Regression | **2.22** | **-0.16** | **2.24** | **7.05%** | **7.88%** |
| B Tier Assists | Lasso Regression | 2.22 | -0.17 | 2.25 | 6.64% | 7.88% |
| B Tier Assists | Random Forest | 2.23 | -0.09 | 2.27 | 5.80% | 7.47% |
| B Tier Assists | Base | 2.41 | 1.31 | 2.41 | N/A | N/A |
| C Tier Assists | Linear Regression | **1.64** | **-0.2** | **1.72** | **6.01%** | **10.38%** |
| C Tier Assists | Lasso Regression | 1.65 | -0.21 | 1.74 | 4.92% | 9.83% |
| C Tier Assists | Random Forest | **1.64** | **-0.13** | **1.72** | **6.01%** | **10.38%** |
| C Tier Assists | Base | 1.83 | 1.60 | 1.83 | N/A | N/A |
| A Tier Rebounds | Linear Regression | **3.90** | **0.19** | **3.95** | **12.80%** | **13.91%** |
| A Tier Rebounds | Lasso Regression | 3.91 | 0.19 | 3.97 | 12.36% | 13.69% |
| A Tier Rebounds | Random Forest | 3.95 | 0.3 | 4.02 | 11.26% | 12.80% |
| A Tier Rebounds | Base | 4.53 | -0.19 | 4.53 | N/A | N/A |
| B Tier Rebounds | Linear Regression | **2.43** | **-0.02** | **2.56** | 7.65% | **25.69%** |
| B Tier Rebounds | Lasso Regression | 2.93 | -0.01 | 3.03 | 7.34% | 10.40% |
| B Tier Rebounds | Random Forest | 2.96 | 0.07 | 3.06 | 6.42% | 9.48% |
| B Tier Rebounds | Base | 3.27 | 0.55 | 3.27 | N/A | N/A |
| C Tier Rebounds | Linear Regression | **2.43** | **-0.23** | **2.56** | **4.12%** | **8.99%** |
| C Tier Rebounds | Lasso Regression | 2.43 | -0.25 | 2.57 | 3.75% | 8.99% |
| C Tier Rebounds | Random Forest | 2.45 | -0.16 | 2.58 | 3.37% | 8.24% |
| C Tier Rebounds | Base | 2.67 | 1.84 | 2.67 | N/A | N/A |

Table 15: Displays the RSME, mean of the error, standard deviation of the prediction error ($\sigma$), the percent reduction of the standard deviation, and the percent reduction for RMSE for the models for all the notable categories. The best performing statistics for each is in bold

## 7.5 Other Methods Tried

Over the course of this project, our team experimented with implementing additional techniques to see how our results would change in certain circumstances and if any yielded improvements. We normalized our data before training our models, we incorporated team information (total team points scored, etc.) as new predictors, split our data by season, and even built a neural network. A brief description of each of these experiments and their results can be found in the following sections.

### 7.5.1 Normalization

One of the first experiments we performed was to see how to models performed when we normalized our data. Normalization helps to scale predictors so that they are more comparable. Specifically, we used the z-score of each predictor column to normalize the data. This quantifies how many standard deviations from the column mean each data point is and the formula for the z-score can

be seen in Equation 17

$$Z = \frac{x - \mu_X}{\sigma_X} \tag{17}$$

where the score, $Z$, is calculated from the column $X$ mean, $\mu_X$ subtracted from observed value, $x$, all over the column $X$ standard deviation $\sigma_X$. Ultimately, this test proved to have little to no impact on our predictions, however it was a crucial step in determining feature importance as it brought the information within each column to a scale where they could be compared fairly.

When applying this to our data, it originally didn't change the predictions. This is because the values in each column were very close to one another so the model was not thrown off by one being much larger than the others. The amount of rebounds, assists, and other predictors were close to one another. When we added in team data is when we ran into the most problems with not normalizing data. Since team data has much larger values, it threw off our predictions by causing the model to predict much higher numbers than it was suppose to. Once we normalized all of the columns which contained our predictors, it returned the predictors back to more normal numbers with good standard deviations, means, and medians.

### 7.5.2 Team Data

Another experiment we performed was to include statistics from team performance into our player models. The idea behind this was that players on high performing teams may score more points or get more assists due to the quality of the players around them or vice versa. Once again, however, this proved to be an insignificant change to our models. This could be because players on high performing teams still have to share court time and ball possession with their other, equally talented teammates therefore resulting in no net gain in player performance statistics.

### 7.5.3 Data by Season

Throughout this project our team worked with data spanning several years and seasons. Through data cleaning and elimination of unusable data, our final models were trained and tested on data from 2015 through 2019, however, this still was not a "perfect" data set. Our team wanted to see how the models would perform when handed a perfect season worth of information - ie. each player could be matched to a team for every game, no statistics were missing, all predictors were fully listed, etc. We were able to clean the 2018-2019 season data to fit this form and then trained and tested our models. We believe the models will perform most optimally with full sets of information, however more than one season of data may be necessary. The single season models did not out perform our multi-season models, likely because players can show growth or consistency over many years so one single season cannot fully and accurately capture their potential performance. We think it would be interesting to keep

as "perfect" of data as possible for the next few seasons and retry training on "perfect" data.

### 7.5.4 Neural Network for Points

An additional machine learning technique our team tried to improve upon the "base" or mean predictions was to build a neural network for the A Tier points. Initially we set up the network to return a singular prediction value for each piece of testing data as in our other models. Once this was running about on par with our linear, Lasso, and random forest models, we wanted to see if we could try a bit of a different approach. We altered our final layer to give us an output that was a probability distribution over a set of point "buckets." The idea behind this was that it would more accurately capture the range of possible points a player could score in a given game, and that that could be used to find the score at which the network estimated median occurs. Recall that the median is the point at which 50% of outcomes are above and 50% are below. Since DraftKings will maximize its profit and minimize its risk when betting is split evenly on a prop bet, finding the predicted median performance could prove to be an interesting line to set.

We began by finding the mean points scored by all A Tier players, which came out to approximately 22, then creating single point "buckets" above and below this mean. We then turned our training (and testing) Y vector into a probability distribution matrix, where a "1" occurred in the bucket of the true score - as we knew with 100% certainty that that was the amount of points scored - and "0" everywhere else in that row. Using our same predictors, we then trained the network and tested on the separate data. This successfully gave us a matrix where each row was a probability distribution for the amount of points that could be scored by the corresponding player in the corresponding game.

The final step we took to utilize the neural network was to find which "bucket," or at which point, the model estimated the median to occur. We did this by calculating the cumulative probability for each row beginning in the first column and adding the following one(s) until we reached a total probability of at least 50%. To determine if this method yielded useful results and to effectively compare it to our previous models we used the same error metrics as before for consistency. The summary of these numbers can be seen in in Table ?? **(broken link here)** alongside the other models. Despite an apparently low standard deviation of error and, the neural network ultimately overestimated the actual points scored by a significant amount - the mean of our error approximately -7, meaning it predicted 7 points more than the actual score on average.

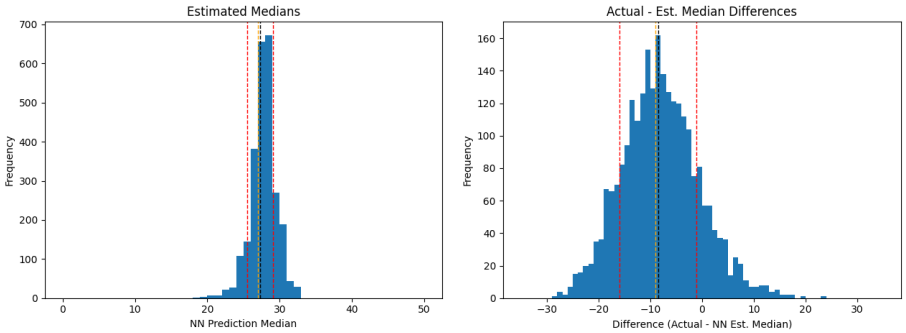


Figure 22: Histograms of the Predictions (left) and Prediction Error (right) for A Tier Points using a Neural Network to Estimate Median Points Scored. The red dotted lines represent the standard deviation of the data. The black line is the mean and the orange line is the median. We calculated our percent error by doing our prediction - the actual score. If the black line is not visible, that means the mean and median are approximately equal.

## 7.6 Feature Importance

Another important characteristic related to our models is that of the feature importance. Each of our models have their own specific features that were given as data in order to create our desired predictions. Feature importance refers to techniques that assign a value to this input features based on how useful they are at predicting a target value. It is important to identify feature importance in each model in order to see what features are actually making an impact on our predictions, and which features are just dead weight that could/should be removed from the model as a whole. Each model type (linear regression, random forest, neural network, etc.) has their own way of evaluating feature importance, since they are all different in nature. So, before presenting the feature importance for our models, we will quickly overview how feature importance works on each of our models.

### 7.6.1 Calculation Methods

Starting out with linear regression, feature importance in linear regression is represented as the linear regression coefficients. Coefficients have already been discussed, but as a quick reminder in a linear regression model can be see in equation 4 the coefficients are represented by the $\beta_p$ values. Each coefficient is given a weight by the model, and so when new data is inputted, each feature will be multiplied by their weight coefficient in order to get the corresponding prediction value.

Along with the coefficients, in linear regression there is another value attached to the coefficients called P-values. P-values for the coefficients show

whether the features have a statistically significant relationship. This means that the p-values evaluate whether the relationships one can observe in the sample data will exist in the larger population. The P-value for each feature tests if the variable has correlation with the dependent variable. If there is no correlation, it means the changes in the independent variable have no correlation to those in the dependent, therefore this feature is probably irrelevant. If the P-value for a feature is less than the significance level of the model, then one can assume the feature is statically significant and worth keeping in the model.

Lasso regression is also a linear model, so the coefficients can be generated in much the same way. What differentiates this method however is that a "budget" when calculating the coefficients, meaning the sum of the absolute values of the coefficients must be less than or equal to a set amount, as seen in equation 7. To account for this, Lasso regression will send the coefficients of the most ineffective coefficients to zero, allowing the more predictive features' coefficients to be greater. This helps make feature importance even more clear.

In a random forest model, feature importance is represented as a value which is appropriately called feature importance. Feature Importance values in random forest measure how much each of the features contributes to decreasing the variance. Using the Scikit-Learn feature-importances function, we get the average decrease in impurity/variance by each feature. [19]

The Sckikit-Learn simple feature importance is not the only indicator to feature importance for random forests. It is also a biased approach, as it has a tendency to inflate the importance of certain features. As a result, we also used a correlation matrix to show how our features correlate with each other and with our prediction value. That way we were able to efficiently judge each feature and choose the ones that made our model better.

### 7.6.2 Game Line Feature Importance

| Predictor | Feature Importance |
|---|---|
| Home or away status | 0.126387 |
| Game line for the home team's most recent game | 0.061352 |
| Game line for the home team's 2nd most recent game | 0.107570 |
| Game line for the home team's 3rd most recent game | 0.105891 |
| Game line for the home team's 4th most recent game | 0.066296 |
| Game line for the opposing team's most recent game | 0.054542 |
| Game line for the opposing team's 2nd most recent game | 0.082422 |
| Game line for the opposing team's 3rd most recent game | 0.124657 |
| Closing line for the opposing team's 4th most recent game | 0.067952 |
| Average score differential for the home team | 0.042554 |
| Average score differential for the opposing team | 0.056470 |
| Home team's Facebook popularity ratio | 0.024657 |
| Opposing team's Facebook popularity ratio | 0.013459 |
| Home team's Instagram popularity ratio | 0.014437 |
| Opposing team's Instagram popularity ratio | 0.026756 |
| Home team's Twitter popularity ratio | 0.011150 |
| Opposing team's Twitter popularity ratio | 0.013447 |

Table 16: A table showing the feature importance of Random Forest game lines model. This table includes each of our predictors and how effective each was in helping us make our predictions.

When evaluating the feature importance of our random forest model and the predictor coefficients of our linear regression and Lasso regression models, some interesting trends emerged. For all three models, the most important factor in predicting the closing line of a game was the home or away status of the team in question. Additionally, the game lines for the home and away team's and the average score differential of the home and away team over the four most recent games appears to possess much more predictive value than any information from social media. Finally, interestingly enough, Instagram appears to be about twice as predictive as Facebook or Twitter.

### 7.6.3 Points Feature Importance

Table 17 shows the feature importance for our A Tier Points random forest model as that was the best performing model (to see the feature importance and regression coefficients for all point models please see the appendix). The two strongest indicators for future points scored are the points scored in the last four games and the number of minutes played in those games as well. This trend of minutes played and previous performance in the statistic in question being key features continues across our assist and rebound models as well.

| **Predictor** | **Random Forest Feature Importance** |
|---|---|
| Points scored in the most recent game | 0.07349 |
| Points scored in the 2nd most recent game | 0.05380 |
| Points scored in the 3rd most recent game | 0.05357 |
| Points scored in the 4th most recent game | 0.05050 |
| Minutes played in the most recent game | 0.04022 |
| Minutes played in the 2nd most recent game | 0.03440 |
| Minutes played in the 3rd most recent game | 0.03100 |
| Minutes played in the 4th most recent game | 0.03693 |
| Rebounds in the most recent game | 0.03386 |
| Rebounds in the 2nd most recent game | 0.03213 |
| Rebounds in the 3rd most recent game | 0.03589 |
| Rebounds in the 4th most recent game | 0.03113 |
| Assists in the most recent game | 0.03893 |
| Assists in the 2nd most recent game | 0.03151 |
| Assists in the 3rd most recent game | 0.03218 |
| Assists in the 4th most recent game | 0.03091 |
| Steals in the most recent game | 0.01934 |
| Steals in the 2nd most recent game | 0.01791 |
| Steals in the 3rd most recent game | 0.02073 |
| Steals in the 4th most recent game | 0.02820 |
| Turnovers in the most recent game | 0.03188 |
| Turnovers in the 2nd most recent game | 0.02320 |
| Turnovers in the 3rd most recent game | 0.02730 |
| Turnovers in the 4th most recent game | 0.03266 |
| Player position | 0.01757 |
| Blocks in the most recent game | 0.01029 |
| Blocks in the 2nd most recent game | 0.01662 |
| Blocks in the 3rd most recent game | 0.01218 |
| Blocks in the 4th most recent game | 0.01396 |
| Fouls in the most recent game | 0.02094 |
| Fouls in the 2nd most recent game | 0.02034 |
| Fouls in the 3rd most recent game | 0.02541 |
| Fouls in the 4th most recent game | 0.02104 |

Table 17: A Table showing the feature importance of our Random Forest points model for A Tier players

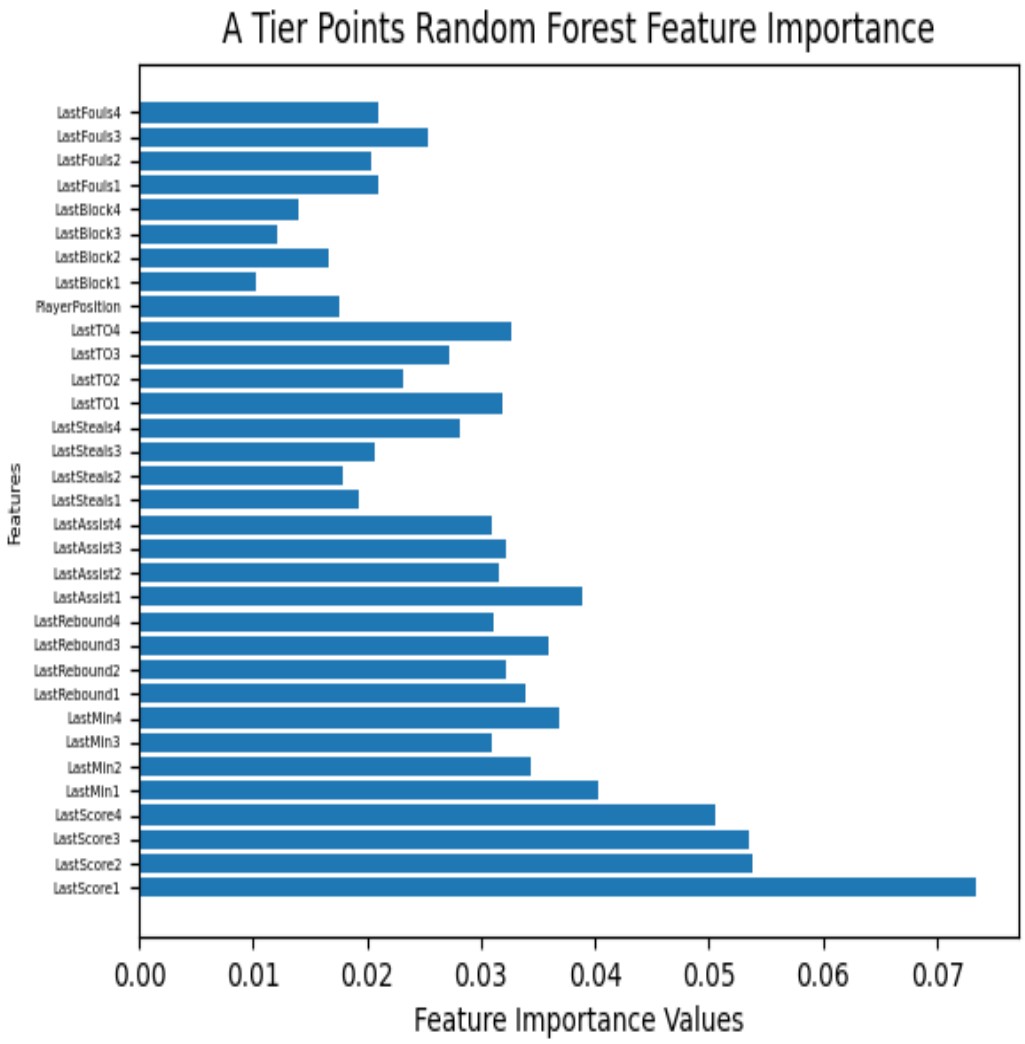


Figure 23: Our graph visualizing the feature importance for our A tier points random forest model

#### 7.6.4 Assists Feature Importance

Our best performing assist model was the Lasso regression model. Therefore, the feature importance is expressed through the coefficient values. Again we can see high levels of importance in previous assists scored, but we can also easily pick out the unimportant features. For example, steals any further back than

one game tend to be less predictive, so the Lasso model has set their coefficients to zero. An interesting thing to note is that for each of the sets of statistics (assists, steals, turnovers, etc.) over the last four games, the most recent game has the strongest coefficient, indicating that the most recent history is most predictive of how the next game will go.

| **Predictor** | **Predictor Coefficient** |
|---|---|
| Assists in the most recent game | 0.31151 |
| Assists in the 2nd most recent game | 0.06908 |
| Assists in the 3rd most recent game | 0.00000 |
| Assists in the 4th most recent game | 0.14248 |
| Minutes in the most recent game | 0.00000 |
| Minutes in the 2nd most recent game | 0.00000 |
| Minutes in the 3rd most recent game | 0.00000 |
| Minutes in the 4th most recent game | 0.05986 |
| Player position | -0.07096 |
| Steals in the most recent game | 0.07757 |
| Steals in the 2nd most recent game | 0.00000 |
| Steals in the 3rd most recent game | 0.00000 |
| Steals in the 4th most recent game | 0.00000 |
| Turnovers in the most recent game | 0.00623 |
| Turnovers in the 2nd most recent game | 0.18668 |
| Turnovers in the 3rd most recent game | 0.00000 |
| Turnovers in the 4th most recent game | 0.09110 |
| Rebounds in the most recent game | 0.08262 |
| Rebounds in the 2nd most recent game | 0.00000 |
| Rebounds in the 3rd most recent game | 0.07496 |
| Rebounds in the 4th most recent game | 0.03017 |
| Blocks in the most recent game | -0.05237 |
| Blocks in the 2nd most recent game | 0.00000 |
| Blocks in the 3rd most recent game | 0.00000 |
| Blocks in the 4th most recent game | 0.00000 |
| Fouls in the most recent game | 0.10390 |
| Fouls in the 2nd most recent game | 0.07641 |
| Fouls in the 3rd most recent game | 0.00000 |
| Fouls in the 4th most recent game | 0.00000 |

Table 18: A Table showing the predictor coefficients of our Lasso regression assists model for A Tier players

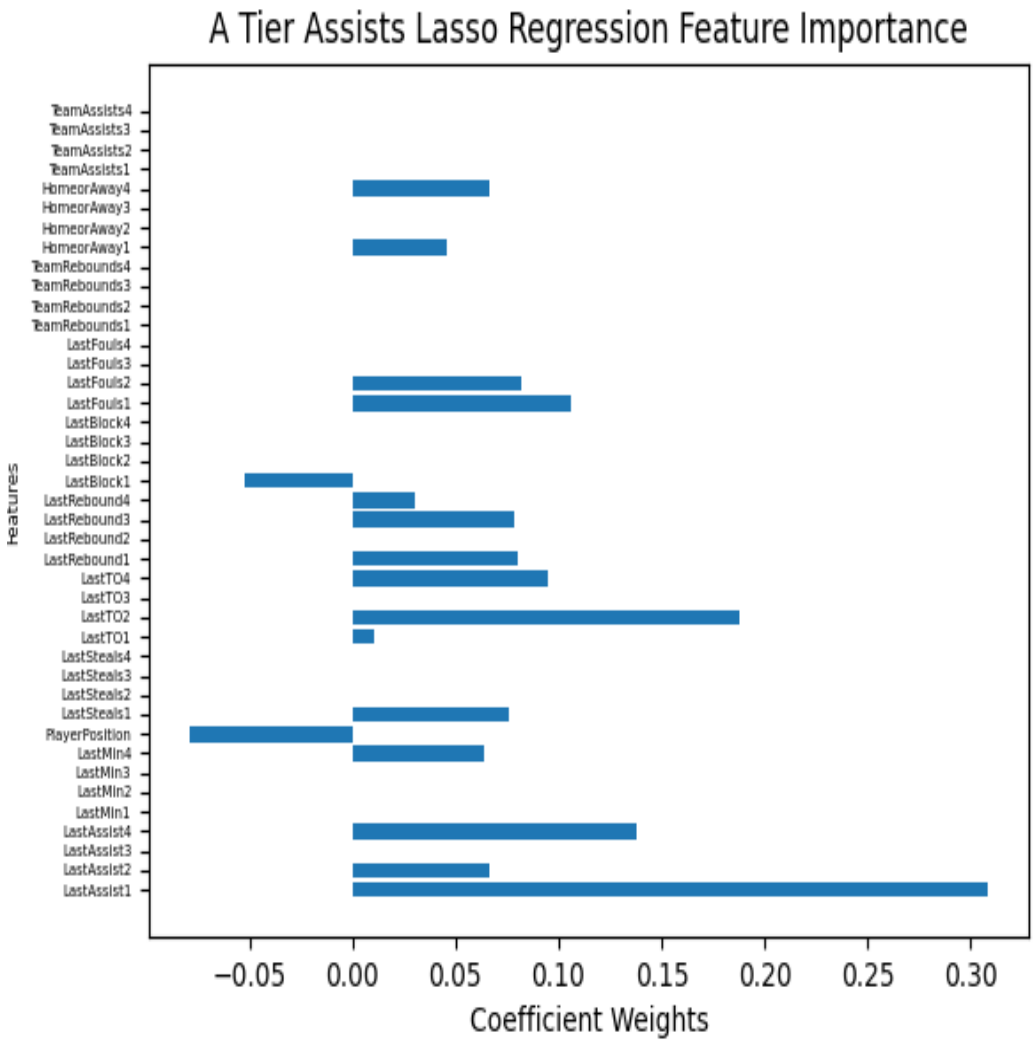


Figure 24: Our graph visualizing the feature importance for our A tier assists Lasso regression model

### 7.6.5 Rebounds Feature Importance

For rebounds, our linear model returned the best results with the coefficients seen in Table 19. Unlike Lasso models, no coefficients are set to zero, so we have to compare them all to determine which are important or unimportant.

| **Predictor** | **Predictor Coefficient** |
|---|---|
| Rebounds in the most recent game | -0.08641 |
| Rebounds in the 2nd most recent game | 0.75234 |
| Rebounds in the 3rd most recent game | 0.30431 |
| Rebounds in the 4th most recent game | 0.33001 |
| Assists in the most recent game | 0.19037 |
| Assists in the 2nd most recent game | 0.02863 |
| Assists in the 3rd most recent game | -0.00034 |
| Assists in the 4th most recent game | -0.11391 |
| Minutes in the most recent game | 0.37877 |
| Minutes in the 2nd most recent game | -0.33349 |
| Minutes in the 3rd most recent game | -0.07402 |
| Minutes in the 4th most recent game | -0.04604 |
| Player position | 0.59037 |
| Steals in the most recent game | 0.03816 |
| Steals in the 2nd most recent game | 0.32705 |
| Steals in the 3rd most recent game | 0.22775 |
| Steals in the 4th most recent game | 0.02841 |
| Turnovers in the most recent game | 0.13440 |
| Turnovers in the 2nd most recent game | 0.06633 |
| Turnovers in the 3rd most recent game | 0.20325 |
| Turnovers in the 4th most recent game | -0.06206 |
| Blocks in the most recent game | 0.06230 |
| Blocks in the 2nd most recent game | 0.30628 |
| Blocks in the 3rd most recent game | 0.20532 |
| Blocks in the 4th most recent game | 0.29893 |
| Fouls in the most recent game | -0.01391 |
| Fouls in the 2nd most recent game | -0.17368 |
| Fouls in the 3rd most recent game | 0.06482 |
| Fouls in the 4th most recent game | -0.02837 |

Table 19: A Table showing the predictor coefficients of our linear regression rebounds model for A Tier players

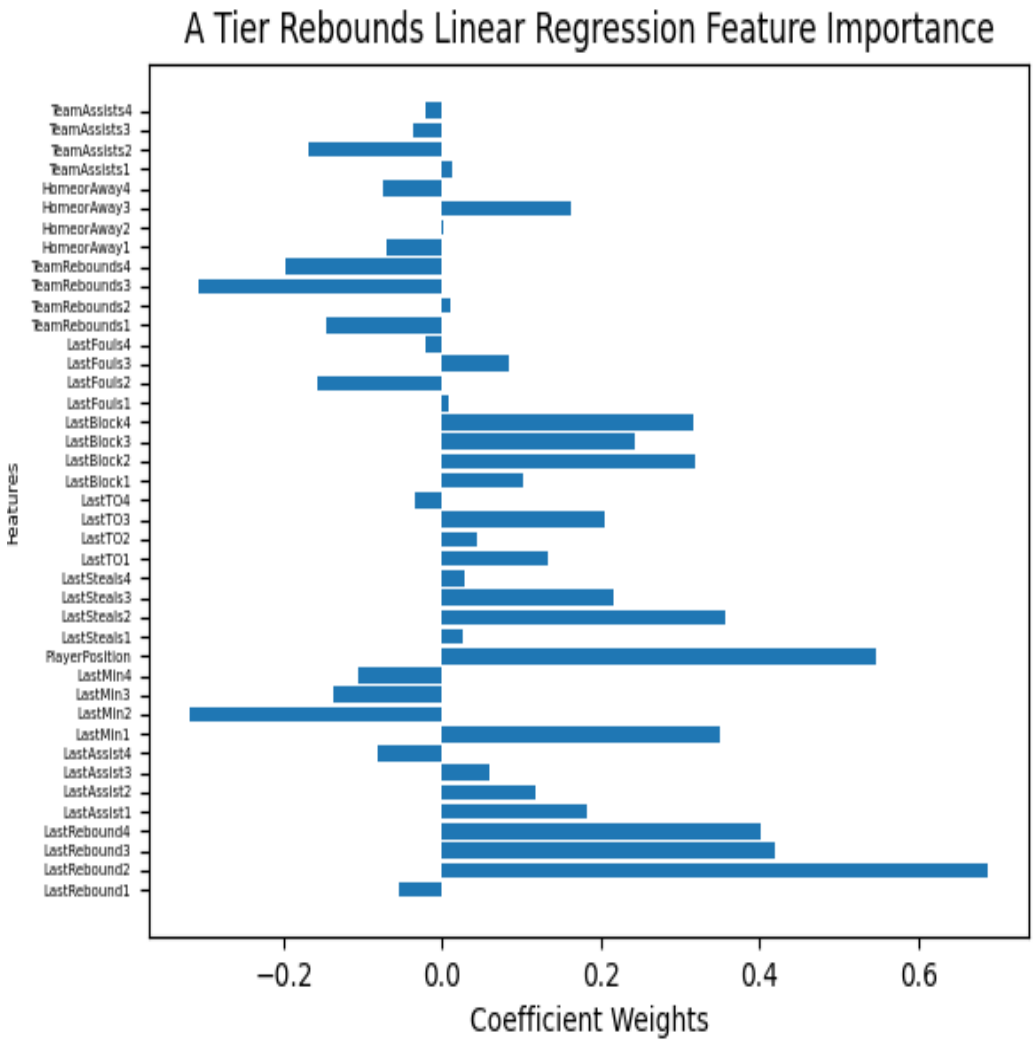


Figure 25: Our graph visualizing the feature importance for our A tier rebounds linear regression model

# 8 Recommendations & Future Work

When building our player performance models we wanted to be able to include predictors based on team and opponent information. For example, it is important for us to know if a player is playing on the Los Angeles Lakers alongside high scorers like LeBron James, since this could increase the number of assists they will score in a game. Originally however, we had a hard time accessing this sort of data. From the data given to us by DraftKings, we were able to verify that a player played in a specific game, however that set of data didn't tell us which team they played for. This prevented us from easily incorporating team statistics or opponent strength. Instead, we had to match up dates and adjust for time-zone information to match players to teams based on data from NBA.com in order to access these.

As a result, one of our major recommendations for generating player predictions is to include team data in the player models. We were able to do this in many of our models, however it required the creation of a fairly complex cleaner to tie together multiple data sets. In the future, we believe it would be very beneficial for DraftKings to include a team tag or statistic in their PlayerBoxScore.csv data to allow easy mapping of team and player information.

We also recommend restructuring the PlayerBoxScore.csv into a more table-like structure. This way each game for each person is one line instead of it being 10 or more lines for all of the statistics. Doing it this way will also allow there to be more statistics added easier, such as adding the team ID of the team each person played for.

Another recommendation we have is adding in Player Prop lines information into the models as predictors. When making models for the game line predictions, one of the best predictors was the past four game lines. Unfortunately, we did not have the Player Prop lines for our models. We highly recommend adding in these predictors for the models. We also encourage and recommend the models being applied to more statistics than only points, assists, and rebounds.

In the future we think DraftKings could further work on developing the neural network model we began. Some interesting things to try would include optimizing the number of epochs and batches to train on, investigating new loss functions, or experimenting with different layers than dense linear layers like we used. Due to time constraints, the team was unable to reach the performance level we had envisioned for this model, but remain hopeful that it can prove to be a strong prediction tool.

Another, more straightforward, application of this work would be to build models for other key basketball statistics that DraftKings hopes to offer lines for. From our work we can conclude that the three model types we tested are able to effectively reduce prediction error and have a better idea of what features are most crucial in generating useful predictions. This should be a good starting space for DraftKings to develop new models.

# 9 Appendix A - Game Line Results

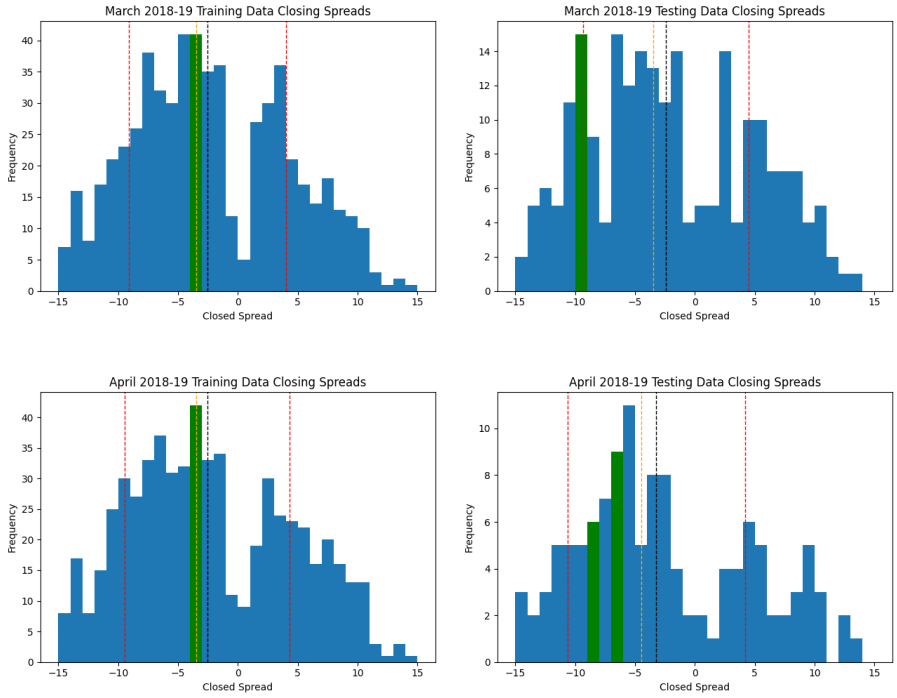


Figure 26: Histograms for training data and testing data game line distributions for March and April where Testing is games in the given month, and training data is the games from the three months prior.

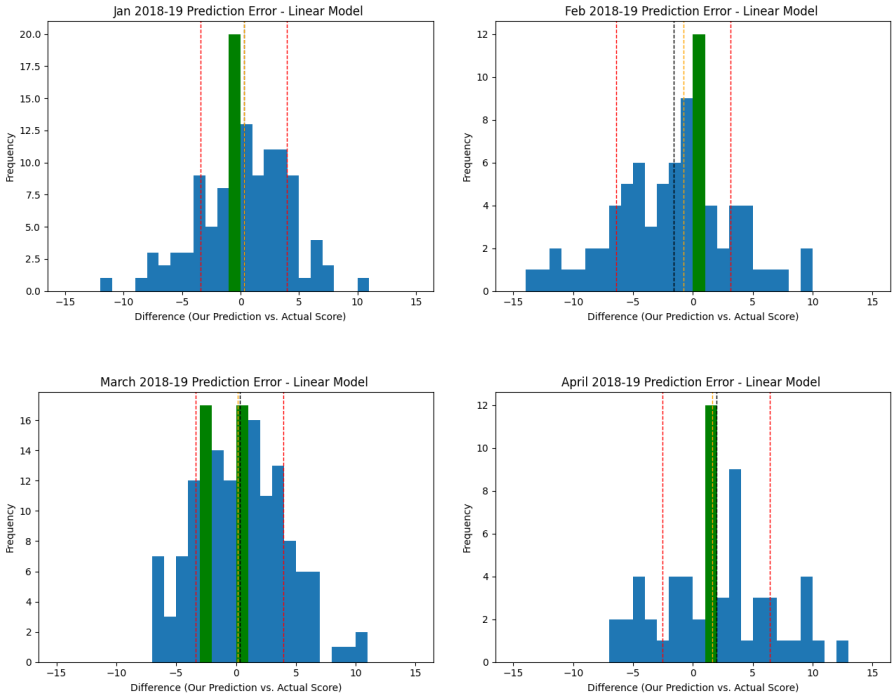


Figure 27: Histograms for prediction error for our game line Linear Regression model when testing data is games in a given month and training data is games from the three months prior

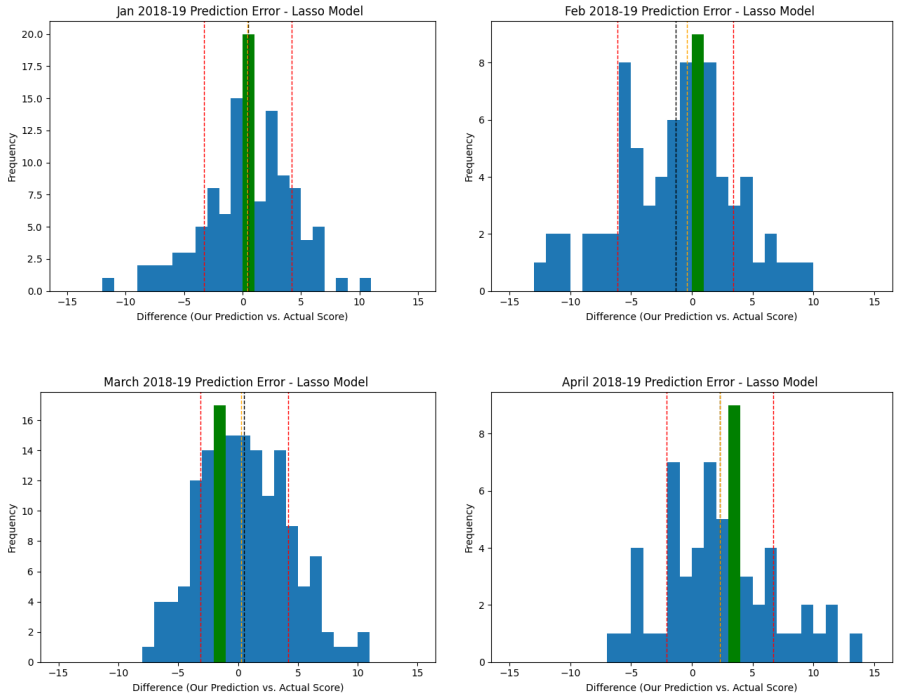


Figure 28: Histograms for prediction error for our game line Lasso model when testing data is games in a given month and training data is games from the three months prior

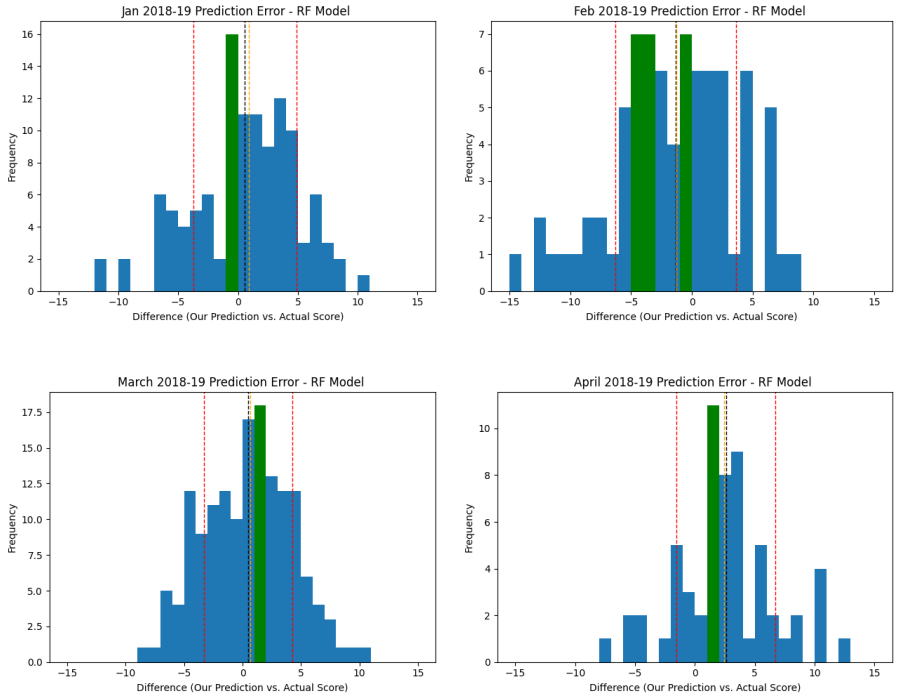


Figure 29: Histograms for prediction error for our game line Random Forest model when testing data is games in a given month and training data is games from the three months prior

| **Predictor** | **Predictor Coefficient** |
|---|---|
| Home or away status | heightGame line for the home team's most recent game |
| Game line for the home team's 2nd most recent game | |
| Game line for the home team's 3rd most recent game | |
| Game line for the home team's 4th most recent game | |
| Game line for the opposing team's most recent game | |
| Game line for the opposing team's 2nd most recent game | |
| Game line for the opposing team's 3rd most recent game | |
| Closing line for the opposing team's 4th most recent game | |
| Average score differential for the home team | |
| Average score differential for the opposing team | |
| Home team's Facebook popularity ratio | |
| Opposing team's Facebook popularity ratio | |
| Home team's Instagram popularity ratio | |
| Opposing team's Instagram popularity ratio | |
| Home team's Twitter popularity ratio | |
| Opposing team's Twitter popularity ratio | |

Table 20: A table showing the predictor coefficients of Lasso regression game lines model

# 10 Appendix B - Additional Points Results

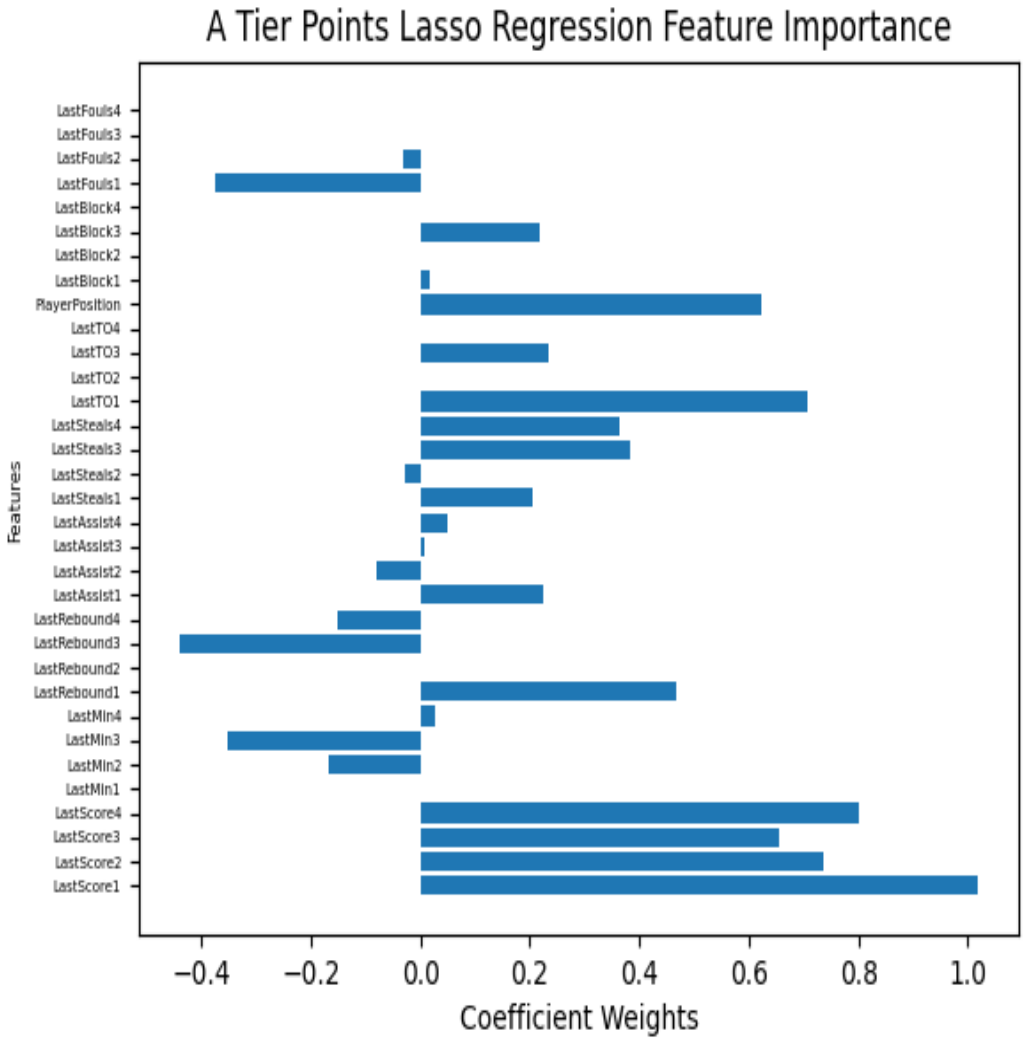


Figure 30: Our graph visualizing the feature importance for our A tier points Lasso regression model

| **Predictor** | **Predictor Coefficient** |
|---|---|
| Points scored in the most recent game | 1.04742 |
| Points scored in the 2nd most recent game | 0.84790 |
| Points scored in the 3rd most recent game | 0.84049 |
| Points scored in the 4th most recent game | 0.82153 |
| Minutes played in the most recent game | 0.00130 |
| Minutes played in the 2nd most recent game | -0.31377 |
| Minutes played in the 3rd most recent game | -0.60174 |
| Minutes played in the 4th most recent game | 0.17962 |
| Rebounds in the most recent game | 0.57344 |
| Rebounds in the 2nd most recent game | 0.00110 |
| Rebounds in the 3rd most recent game | -0.71007 |
| Rebounds in the 4th most recent game | -0.44729 |
| Assists in the most recent game | 0.35330 |
| Assists in the 2nd most recent game | -0.24812 |
| Assists in the 3rd most recent game | 0.20906 |
| Assists in the 4th most recent game | 0.17798 |
| Steals in the most recent game | 0.28758 |
| Steals in the 2nd most recent game | -0.08473 |
| Steals in the 3rd most recent game | 0.52785 |
| Steals in the 4th most recent game | 0.45636 |
| Turnovers in the most recent game | 0.77631 |
| Turnovers in the 2nd most recent game | 0.05851 |
| Turnovers in the 3rd most recent game | 0.34730 |
| Turnovers in the 4th most recent game | -0.12114 |
| Player position | 1.06001 |
| Blocks in the most recent game | 0.07763 |
| Blocks in the 2nd most recent game | -0.04507 |
| Blocks in the 3rd most recent game | 0.32894 |
| Blocks in the 4th most recent game | 0.06144 |
| Fouls in the most recent game | -0.50250 |
| Fouls in the 2nd most recent game | -0.13145 |
| Fouls in the 3rd most recent game | -0.05575 |
| Fouls in the 4th most recent game | -0.08759 |

Table 21: A Table showing the predictor coefficient for our linear regression points model for A Tier players

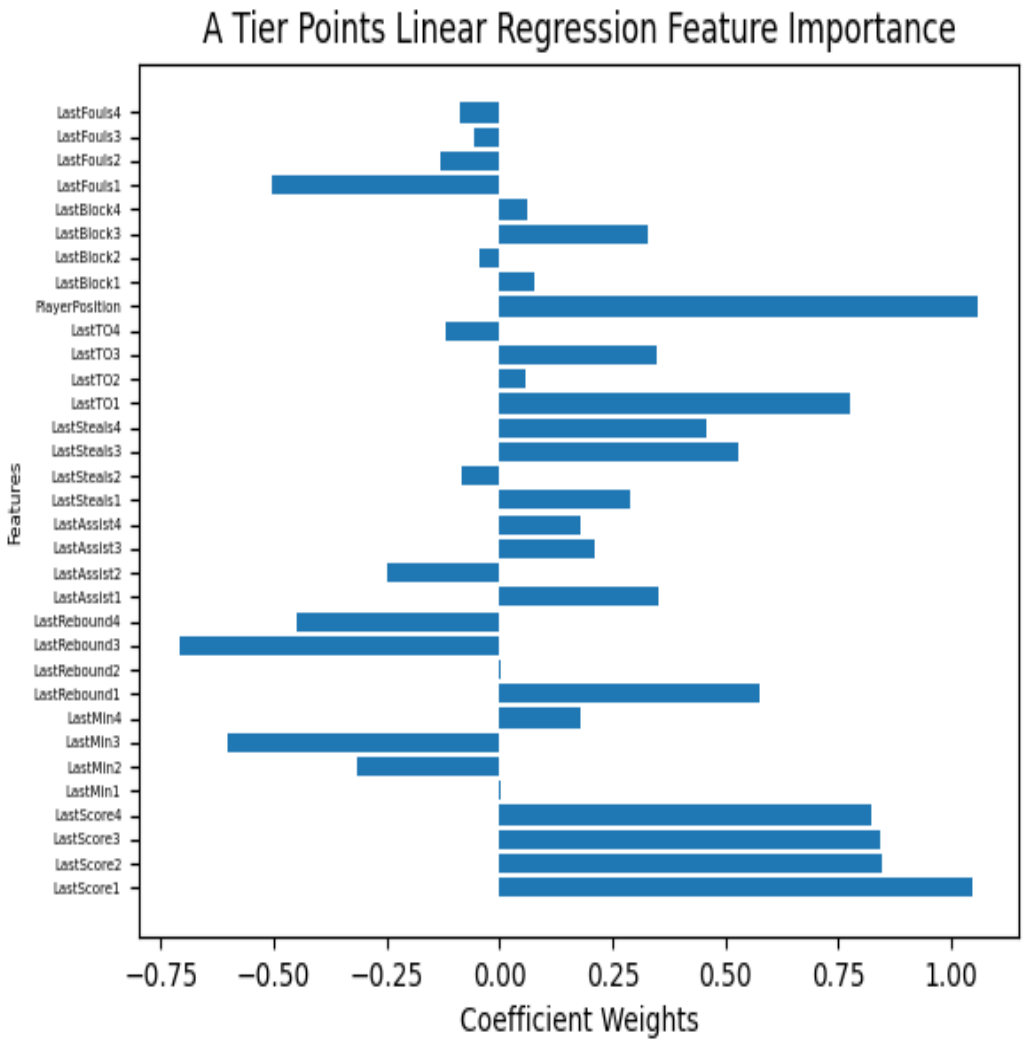


Figure 31: Our graph visualizing the feature importance for our A tier points linear regression model

| **Predictor** | **Predictor Coefficient** |
|---|---|
| Points scored in the most recent game | 1.01945 |
| Points scored in the 2nd most recent game | 0.73523 |
| Points scored in the 3rd most recent game | 0.65425 |
| Points scored in the 4th most recent game | 0.79994 |
| Minutes played in the most recent game | 0.00000 |
| Minutes played in the 2nd most recent game | -0.16636 |
| Minutes played in the 3rd most recent game | -0.35232 |
| Minutes played in the 4th most recent game | 0.02646 |
| Rebounds in the most recent game | 0.46608 |
| Rebounds in the 2nd most recent game | 0.00000 |
| Rebounds in the 3rd most recent game | -0.44050 |
| Rebounds in the 4th most recent game | -0.15026 |
| Assists in the most recent game | 0.22363 |
| Assists in the 2nd most recent game | -0.07894 |
| Assists in the 3rd most recent game | 0.00666 |
| Assists in the 4th most recent game | 0.04923 |
| Steals in the most recent game | 0.20456 |
| Steals in the 2nd most recent game | -0.02883 |
| Steals in the 3rd most recent game | 0.38395 |
| Steals in the 4th most recent game | 0.36447 |
| Turnovers in the most recent game | 0.70629 |
| Turnovers in the 2nd most recent game | 0.00000 |
| Turnovers in the 3rd most recent game | 0.23395 |
| Turnovers in the 4th most recent game | 0.00000 |
| Player position | 0.62234 |
| Blocks in the most recent game | 0.01788 |
| Blocks in the 2nd most recent game | 0.00000 |
| Blocks in the 3rd most recent game | 0.21759 |
| Blocks in the 4th most recent game | 0.00000 |
| Fouls in the most recent game | -0.37392 |
| Fouls in the 2nd most recent game | -0.03157 |
| Fouls in the 3rd most recent game | 0.00000 |
| Fouls in the 4th most recent game | 0.00000 |

Table 22: A Table showing the predictor coefficient for our Lasso regression points model for A Tier players

| Predictor | Predictor Coefficient |
|---|---|
| Points scored in the most recent game | 0.56006 |
| Points scored in the 2nd most recent game | 0.35022 |
| Points scored in the 3rd most recent game | 0.16182 |
| Points scored in the 4th most recent game | 0.20800 |
| Minutes played in the most recent game | -0.35836 |
| Minutes played in the 2nd most recent game | 0.34293 |
| Minutes played in the 3rd most recent game | 0.13869 |
| Minutes played in the 4th most recent game | 0.31101 |
| Rebounds in the most recent game | 0.62994 |
| Rebounds in the 2nd most recent game | 0.02173 |
| Rebounds in the 3rd most recent game | -0.24730 |
| Rebounds in the 4th most recent game | -0.35935 |
| Assists in the most recent game | 0.29791 |
| Assists in the 2nd most recent game | -0.17404 |
| Assists in the 3rd most recent game | 0.34504 |
| Assists in the 4th most recent game | 0.34910 |
| Steals in the most recent game | -0.14787 |
| Steals in the 2nd most recent game | 0.11762 |
| Steals in the 3rd most recent game | 0.24214 |
| Steals in the 4th most recent game | -0.09396 |
| Turnovers in the most recent game | 0.05803 |
| Turnovers in the 2nd most recent game | -0.11229 |
| Turnovers in the 3rd most recent game | -0.02994 |
| Turnovers in the 4th most recent game | 0.02096 |
| Player position | 0.43634 |
| Blocks in the most recent game | -0.12714 |
| Blocks in the 2nd most recent game | -0.15216 |
| Blocks in the 3rd most recent game | -0.06962 |
| Blocks in the 4th most recent game | 0.02279 |
| Fouls in the most recent game | -0.27141 |
| Fouls in the 2nd most recent game | -0.16265 |
| Fouls in the 3rd most recent game | -0.07094 |
| Fouls in the 4th most recent game | -0.20333 |

Table 23: A Table showing the predictor coefficients of our linear regression points model for B Tier players

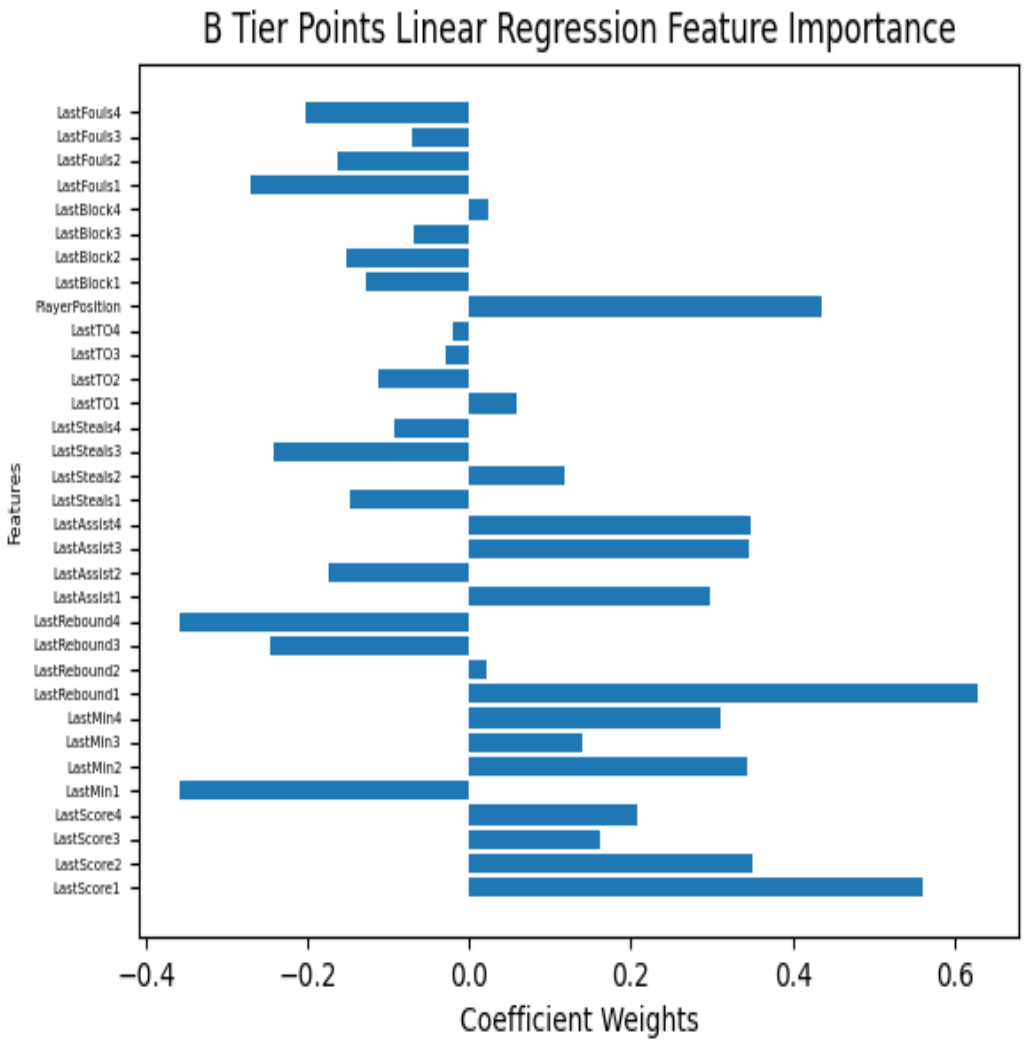


Figure 32: Our graph visualizing the feature importance for our B tier points linear regression model

| Predictor | Predictor Coefficient |
|---|---|
| Points scored in the most recent game | 0.39960 |
| Points scored in the 2nd most recent game | 0.32070 |
| Points scored in the 3rd most recent game | 0.11579 |
| Points scored in the 4th most recent game | 0.17050 |
| Minutes played in the most recent game | 0.00000 |
| Minutes played in the 2nd most recent game | 0.20432 |
| Minutes played in the 3rd most recent game | 0.00000 |
| Minutes played in the 4th most recent game | 0.14697 |
| Rebounds in the most recent game | 0.31862 |
| Rebounds in the 2nd most recent game | 0.00000 |
| Rebounds in the 3rd most recent game | 0.00000 |
| Rebounds in the 4th most recent game | -0.06500 |
| Assists in the most recent game | 0.08232 |
| Assists in the 2nd most recent game | 0.00000 |
| Assists in the 3rd most recent game | 0.19401 |
| Assists in the 4th most recent game | 0.21176 |
| Steals in the most recent game | -0.07849 |
| Steals in the 2nd most recent game | 0.00000 |
| Steals in the 3rd most recent game | -0.12734 |
| Steals in the 4th most recent game | 0.00000 |
| Turnovers in the most recent game | 0.00000 |
| Turnovers in the 2nd most recent game | 0.00000 |
| Turnovers in the 3rd most recent game | 0.00000 |
| Turnovers in the 4th most recent game | 0.00000 |
| Player position | 0.00000 |
| Blocks in the most recent game | -0.04203 |
| Blocks in the 2nd most recent game | -0.06180 |
| Blocks in the 3rd most recent game | 0.00000 |
| Blocks in the 4th most recent game | 0.00000 |
| Fouls in the most recent game | -0.20917 |
| Fouls in the 2nd most recent game | -0.08733 |
| Fouls in the 3rd most recent game | 0.00000 |
| Fouls in the 4th most recent game | -0.11367 |

Table 24: A Table showing the predictor coefficients of our Lasso Regression points model for B Tier players

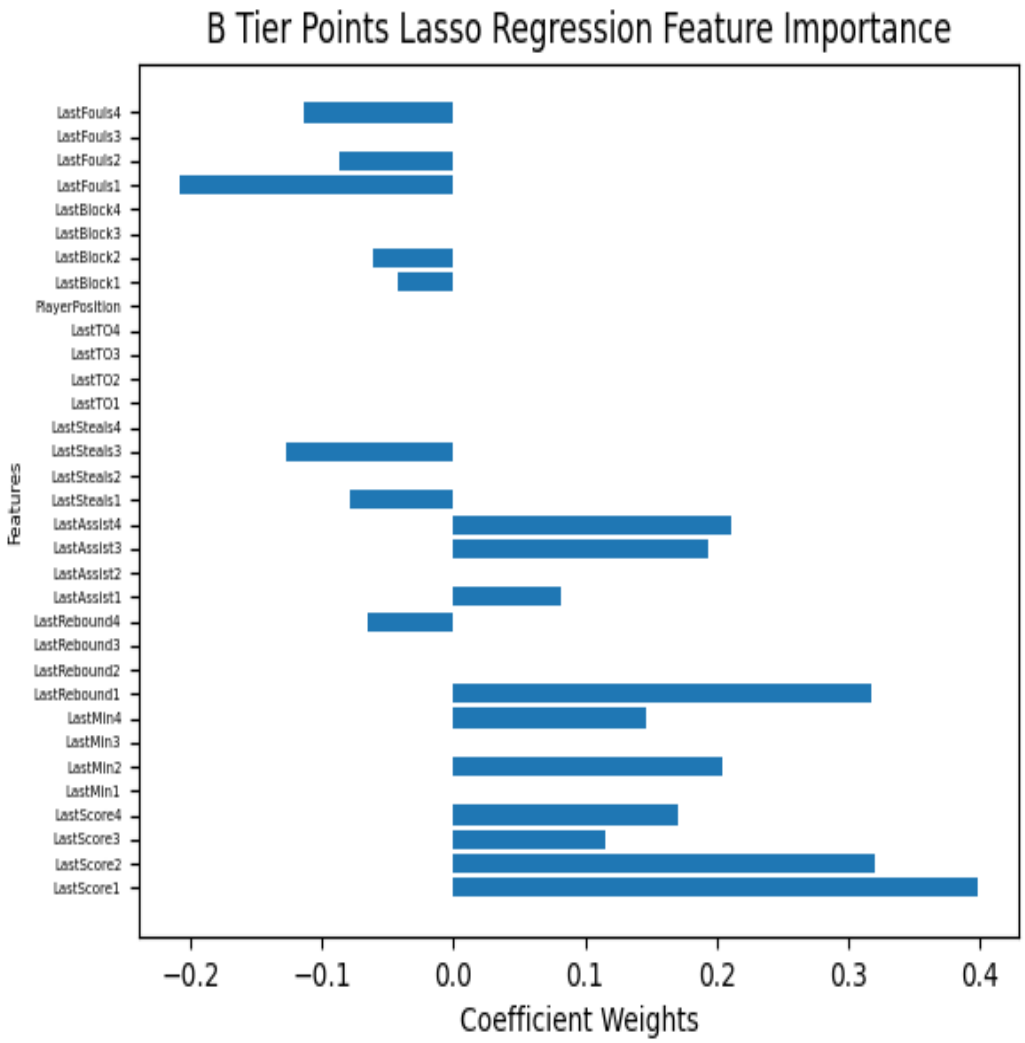


Figure 33: Our graph visualizing the feature importance for our B tier points Lasso regression model

| Predictor | Random Forest Feature Importance |
|---|---|
| Points scored in the most recent game | 0.05950 |
| Points scored in the 2nd most recent game | 0.05668 |
| Points scored in the 3rd most recent game | 0.05013 |
| Points scored in the 4th most recent game | 0.05041 |
| Minutes played in the most recent game | 0.04483 |
| Minutes played in the 2nd most recent game | 0.04821 |
| Minutes played in the 3rd most recent game | 0.04792 |
| Minutes played in the 4th most recent game | 0.04774 |
| Rebounds in the most recent game | 0.03690 |
| Rebounds in the 2nd most recent game | 0.03404 |
| Rebounds in the 3rd most recent game | 0.03280 |
| Rebounds in the 4th most recent game | 0.03665 |
| Assists in the most recent game | 0.03320 |
| Assists in the 2nd most recent game | 0.03108 |
| Assists in the 3rd most recent game | 0.03276 |
| Assists in the 4th most recent game | 0.03402 |
| Steals in the most recent game | 0.01671 |
| Steals in the 2nd most recent game | 0.02090 |
| Steals in the 3rd most recent game | 0.01689 |
| Steals in the 4th most recent game | 0.01966 |
| Turnovers in the most recent game | 0.02277 |
| Turnovers in the 2nd most recent game | 0.02427 |
| Turnovers in the 3rd most recent game | 0.02340 |
| Turnovers in the 4th most recent game | 0.02272 |
| Player position | 0.01309 |
| Blocks in the most recent game | 0.01084 |
| Blocks in the 2nd most recent game | 0.01361 |
| Blocks in the 3rd most recent game | 0.01023 |
| Blocks in the 4th most recent game | 0.01084 |
| Fouls in the most recent game | 0.02602 |
| Fouls in the 2nd most recent game | 0.02278 |
| Fouls in the 3rd most recent game | 0.02513 |
| Fouls in the 4th most recent game | 0.02327 |

Table 25: A Table showing the feature importance of our Random Forest points model for B Tier players

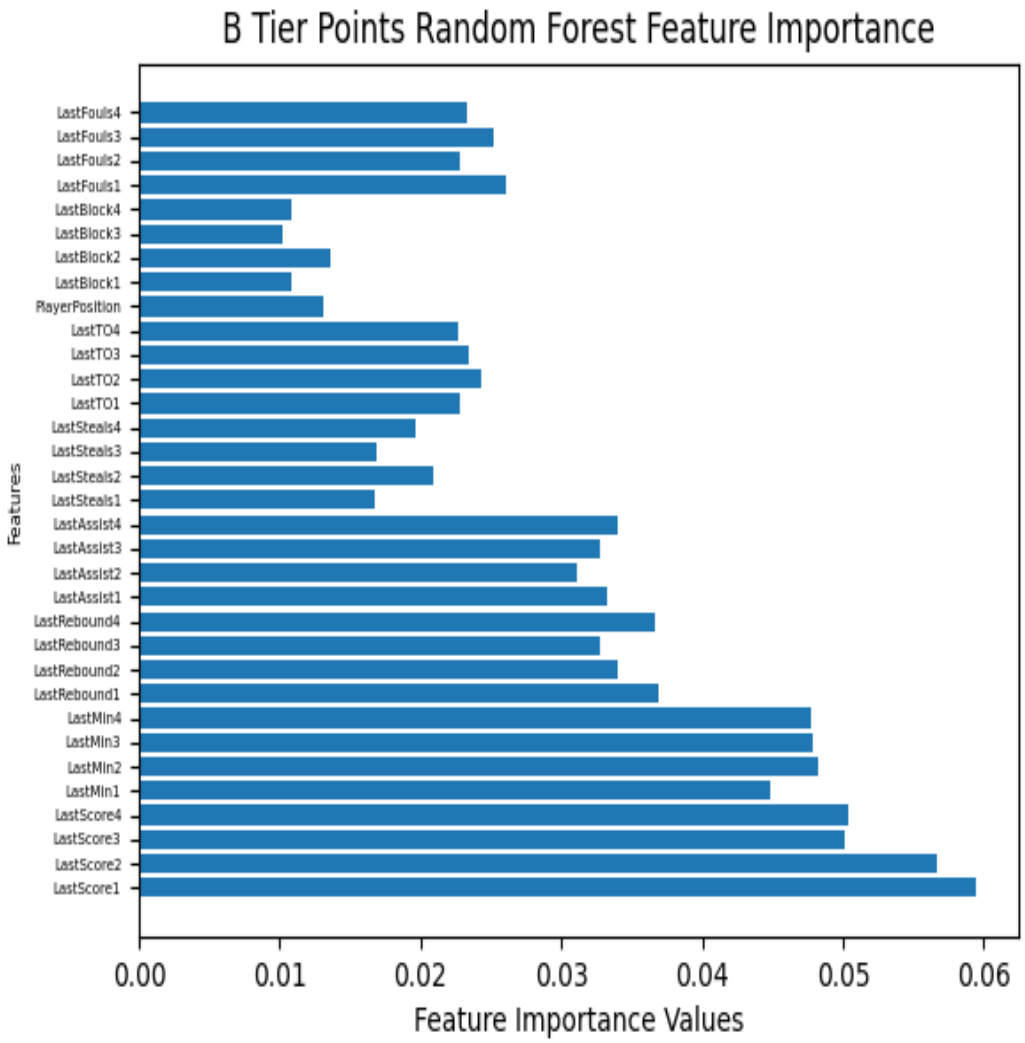


Figure 34: Our graph visualizing the feature importance for our B tier points random forest model

| Predictor | Predictor Coefficient |
|---|---|
| Points scored in the most recent game | 0.30554 |
| Points scored in the 2nd most recent game | 0.00799 |
| Points scored in the 3rd most recent game | 0.16732 |
| Points scored in the 4th most recent game | 0.03963 |
| Minutes played in the most recent game | 0.32409 |
| Minutes played in the 2nd most recent game | 0.19435 |
| Minutes played in the 3rd most recent game | 0.28346 |
| Minutes played in the 4th most recent game | 0.02614 |
| Rebounds in the most recent game | 0.13442 |
| Rebounds in the 2nd most recent game | 0.04288 |
| Rebounds in the 3rd most recent game | -0.06954 |
| Rebounds in the 4th most recent game | 0.06662 |
| Assists in the most recent game | 0.01253 |
| Assists in the 2nd most recent game | -0.19230 |
| Assists in the 3rd most recent game | -0.06702 |
| Assists in the 4th most recent game | 0.08346 |
| Steals in the most recent game | -0.08867 |
| Steals in the 2nd most recent game | 0.16854 |
| Steals in the 3rd most recent game | -0.10141 |
| Steals in the 4th most recent game | 0.05002 |
| Turnovers in the most recent game | 0.03268 |
| Turnovers in the 2nd most recent game | 0.03446 |
| Turnovers in the 3rd most recent game | 0.12702 |
| Turnovers in the 4th most recent game | 0.09863 |
| Player position | 0.06724 |
| Blocks in the most recent game | 0.01962 |
| Blocks in the 2nd most recent game | -0.15816 |
| Blocks in the 3rd most recent game | 0.05156 |
| Blocks in the 4th most recent game | 0.21868 |
| Fouls in the most recent game | -0.04422 |
| Fouls in the 2nd most recent game | -0.14803 |
| Fouls in the 3rd most recent game | 0.13546 |
| Fouls in the 4th most recent game | -0.06797 |

Table 26: A Table showing the predictor coefficients of our linear regression points model for C Tier players

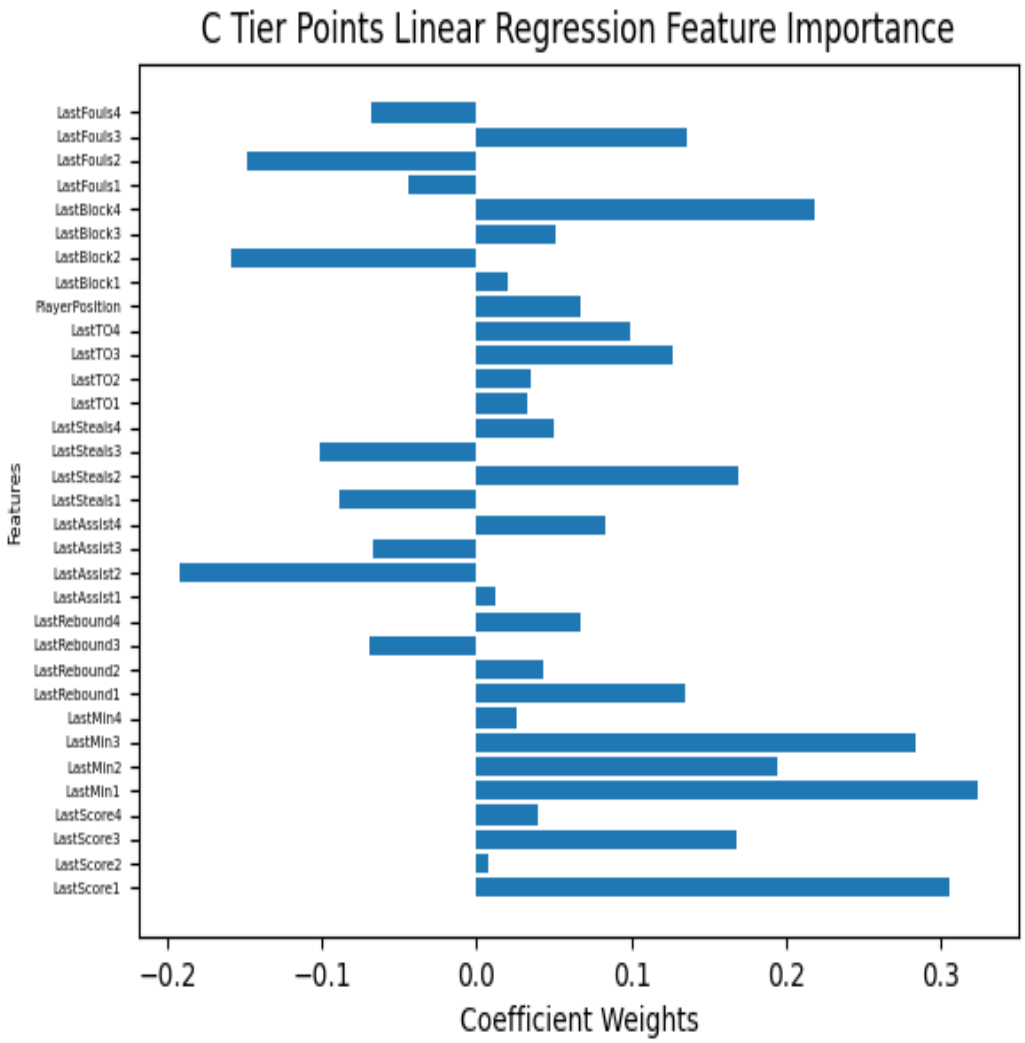


Figure 35: Our graph visualizing the feature importance for our C tier points linear regression model

| **Predictor** | **Predictor Coefficient** |
|---|---|
| Points scored in the most recent game | 0.23678 |
| Points scored in the 2nd most recent game | 0.00000 |
| Points scored in the 3rd most recent game | 0.09306 |
| Points scored in the 4th most recent game | 0.00000 |
| Minutes played in the most recent game | 0.28838 |
| Minutes played in the 2nd most recent game | 0.08206 |
| Minutes played in the 3rd most recent game | 0.22396 |
| Minutes played in the 4th most recent game | 0.05390 |
| Rebounds in the most recent game | 0.11491 |
| Rebounds in the 2nd most recent game | 0.00000 |
| Rebounds in the 3rd most recent game | 0.00000 |
| Rebounds in the 4th most recent game | 0.04033 |
| Assists in the most recent game | 0.00000 |
| Assists in the 2nd most recent game | 0.00000 |
| Assists in the 3rd most recent game | 0.00000 |
| Assists in the 4th most recent game | 0.00000 |
| Steals in the most recent game | 0.00000 |
| Steals in the 2nd most recent game | 0.06623 |
| Steals in the 3rd most recent game | 0.00000 |
| Steals in the 4th most recent game | 0.00000 |
| Turnovers in the most recent game | 0.00000 |
| Turnovers in the 2nd most recent game | 0.00000 |
| Turnovers in the 3rd most recent game | 0.03293 |
| Turnovers in the 4th most recent game | 0.00557 |
| Player position | 0.00000 |
| Blocks in the most recent game | 0.00000 |
| Blocks in the 2nd most recent game | 0.00000 |
| Blocks in the 3rd most recent game | 0.00000 |
| Blocks in the 4th most recent game | 0.14990 |
| Fouls in the most recent game | 0.00000 |
| Fouls in the 2nd most recent game | 0.00000 |
| Fouls in the 3rd most recent game | 0.07664 |
| Fouls in the 4th most recent game | 0.00000 |

Table 27: A Table showing the predictor coefficients of our Lasso regression points model for C Tier players

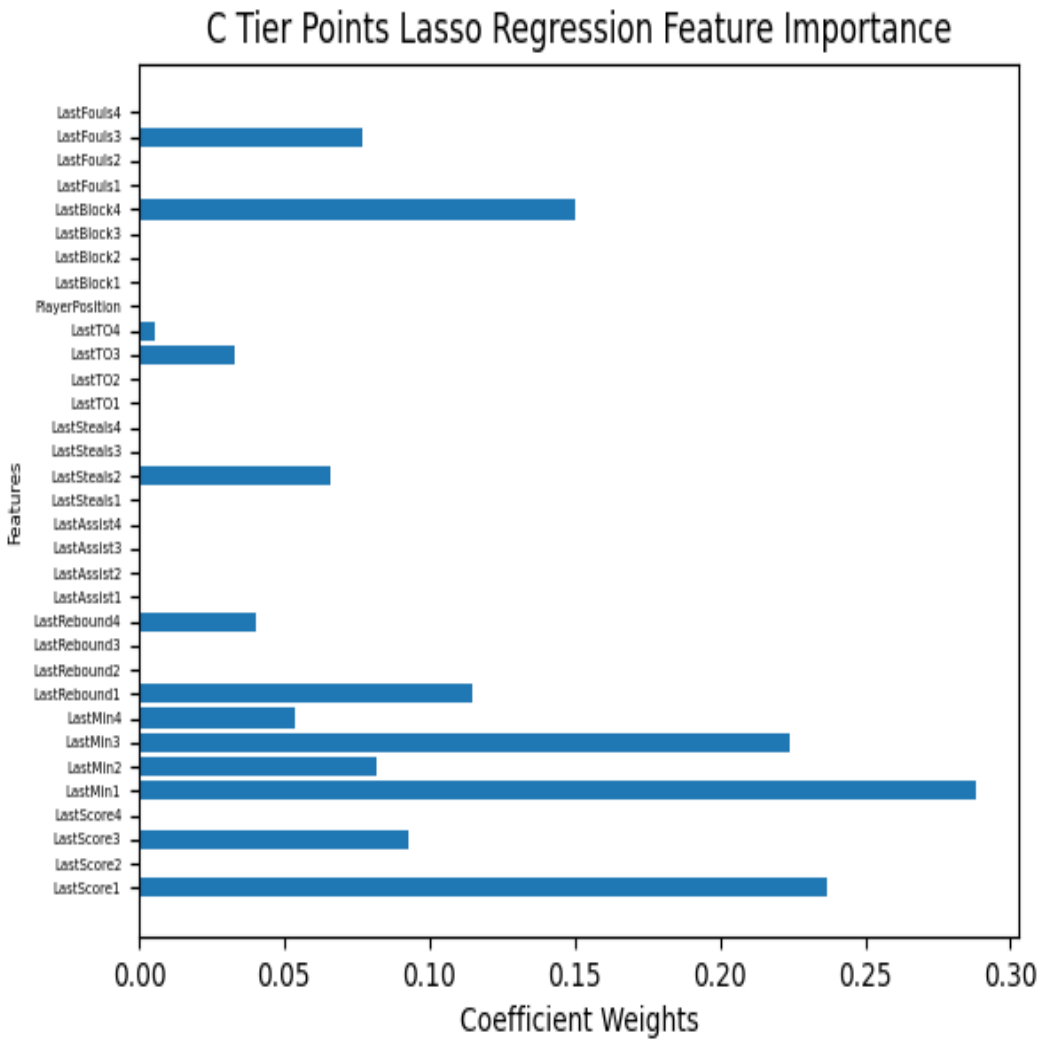


Figure 36: Our graph visualizing the feature importance for our C tier points Lasso regression model

| Predictor | Random Forest Feature Importance |
|---|---|
| Points scored in the most recent game | 0.05111 |
| Points scored in the 2nd most recent game | 0.05288 |
| Points scored in the 3rd most recent game | 0.05396 |
| Points scored in the 4th most recent game | 0.05119 |
| Minutes played in the most recent game | 0.05822 |
| Minutes played in the 2nd most recent game | 0.05468 |
| Minutes played in the 3rd most recent game | 0.05723 |
| Minutes played in the 4th most recent game | 0.05155 |
| Rebounds in the most recent game | 0.03917 |
| Rebounds in the 2nd most recent game | 0.03318 |
| Rebounds in the 3rd most recent game | 0.03286 |
| Rebounds in the 4th most recent game | 0.03531 |
| Assists in the most recent game | 0.02737 |
| Assists in the 2nd most recent game | 0.02767 |
| Assists in the 3rd most recent game | 0.02658 |
| Assists in the 4th most recent game | 0.02699 |
| Steals in the most recent game | 0.01782 |
| Steals in the 2nd most recent game | 0.01836 |
| Steals in the 3rd most recent game | 0.01656 |
| Steals in the 4th most recent game | 0.01628 |
| Turnovers in the most recent game | 0.01956 |
| Turnovers in the 2nd most recent game | 0.02118 |
| Turnovers in the 3rd most recent game | 0.02205 |
| Turnovers in the 4th most recent game | 0.02235 |
| Player position | 0.01433 |
| Blocks in the most recent game | 0.01281 |
| Blocks in the 2nd most recent game | 0.01269 |
| Blocks in the 3rd most recent game | 0.01323 |
| Blocks in the 4th most recent game | 0.01353 |
| Fouls in the most recent game | 0.02514 |
| Fouls in the 2nd most recent game | 0.02501 |
| Fouls in the 3rd most recent game | 0.02493 |
| Fouls in the 4th most recent game | 0.02422 |

Table 28: A Table showing the feature importance of our Random Forest points model for C Tier players

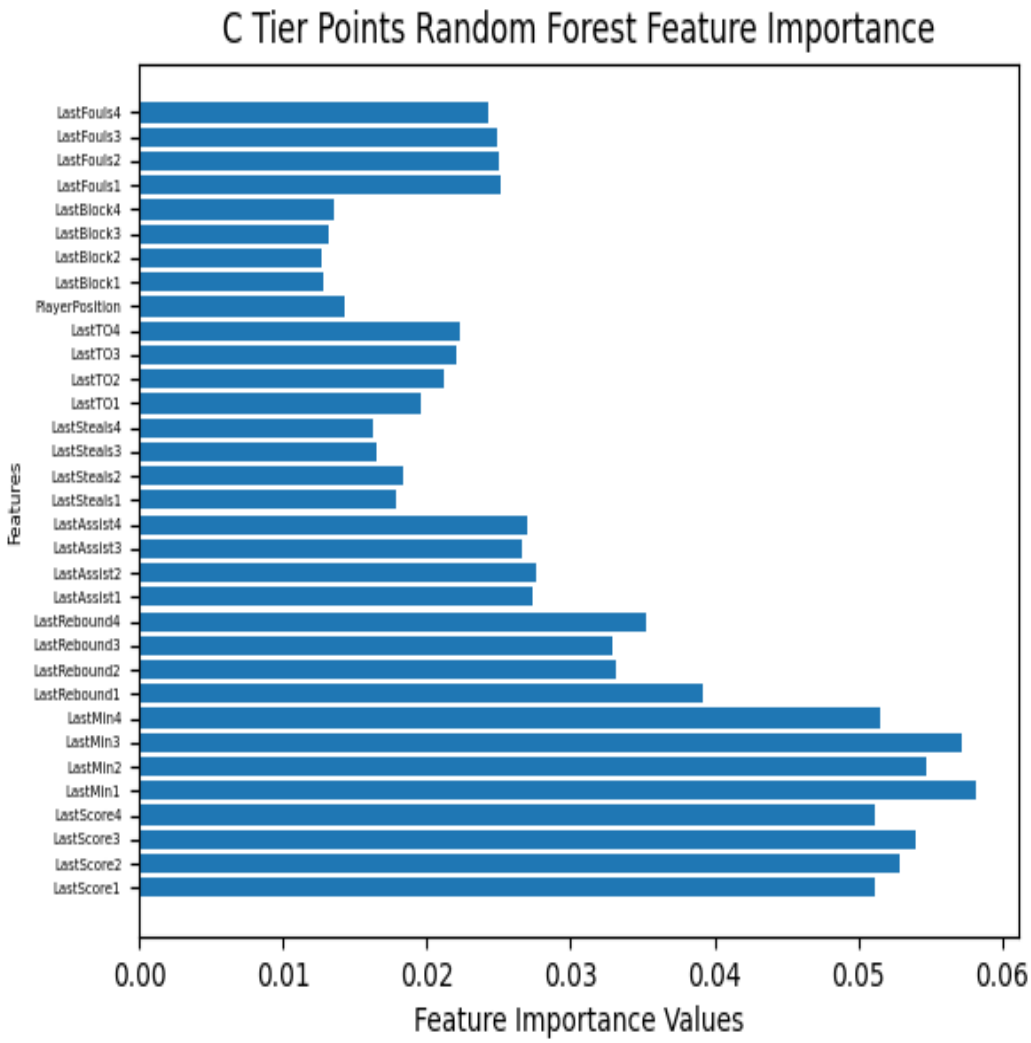


Figure 37: Our graph visualizing the feature importance for our C tier points random forest model

# 11 Appendix C - Additional Assists Results

| Predictor | Predictor Coefficient |
|---|---|
| Assists in the most recent game | 0.37821 |
| Assists in the 2nd most recent game | 0.13468 |
| Assists in the 3rd most recent game | 0.00379 |
| Assists in the 4th most recent game | 0.17664 |
| Minutes in the most recent game | -0.06565 |
| Minutes in the 2nd most recent game | -0.06315 |
| Minutes in the 3rd most recent game | -0.08694 |
| Minutes in the 4th most recent game | 0.15535 |
| Player position | -0.26937 |
| Steals in the most recent game | 0.16463 |
| Steals in the 2nd most recent game | -0.08715 |
| Steals in the 3rd most recent game | 0.01755 |
| Steals in the 4th most recent game | -0.08046 |
| Turnovers in the most recent game | 0.03802 |
| Turnovers in the 2nd most recent game | 0.22454 |
| Turnovers in the 3rd most recent game | 0.01771 |
| Turnovers in the 4th most recent game | 0.14036 |
| Rebounds in the most recent game | 0.15504 |
| Rebounds in the 2nd most recent game | 0.02748 |
| Rebounds in the 3rd most recent game | 0.18596 |
| Rebounds in the 4th most recent game | 0.08652 |
| Blocks in the most recent game | -0.15135 |
| Blocks in the 2nd most recent game | -0.00582 |
| Blocks in the 3rd most recent game | 0.06423 |
| Blocks in the 4th most recent game | -0.05674 |
| Fouls in the most recent game | 0.18649 |
| Fouls in the 2nd most recent game | 0.17285 |
| Fouls in the 3rd most recent game | -0.10585 |
| Fouls in the 4th most recent game | -0.02778 |

Table 29: A Table showing the predictor coefficients of our linear regression assists model for A Tier players

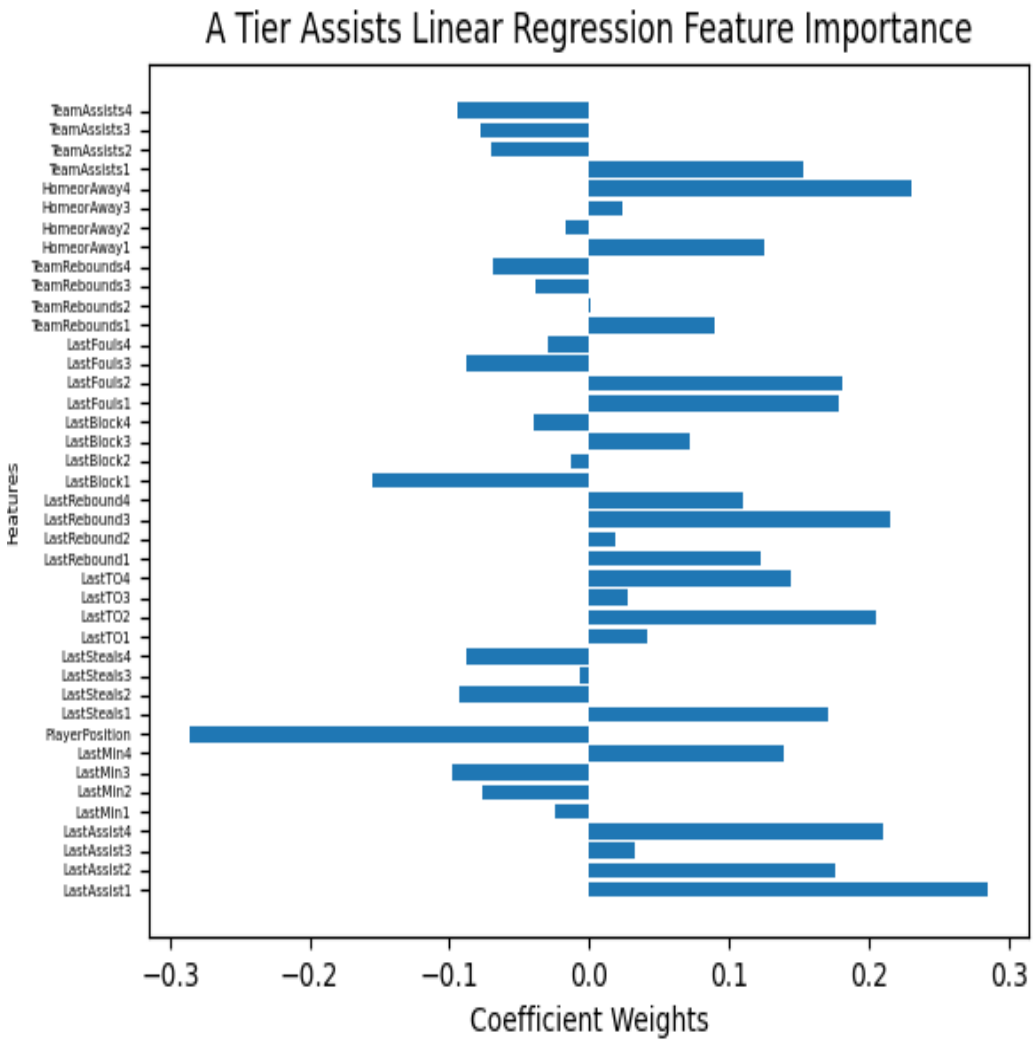


Figure 38: Our graph visualizing the feature importance for our A tier assists linear regression model

| Predictor | Random Forest Feature Importance |
|---|---|
| Assists in the most recent game | 0.05340 |
| Assists in the 2nd most recent game | 0.04267 |
| Assists in the 3rd most recent game | 0.03851 |
| Assists in the 4th most recent game | 0.04638 |
| Minutes in the most recent game | 0.05410 |
| Minutes in the 2nd most recent game | 0.05854 |
| Minutes in the 3rd most recent game | 0.05458 |
| Minutes in the 4th most recent game | 0.05953 |
| Player position | 0.01174 |
| Steals in the most recent game | 0.02450 |
| Steals in the 2nd most recent game | 0.02330 |
| Steals in the 3rd most recent game | 0.02171 |
| Steals in the 4th most recent game | 0.02326 |
| Turnovers in the most recent game | 0.03592 |
| Turnovers in the 2nd most recent game | 0.04504 |
| Turnovers in the 3rd most recent game | 0.04369 |
| Turnovers in the 4th most recent game | 0.03170 |
| Rebounds in the most recent game | 0.04420 |
| Rebounds in the 2nd most recent game | 0.03745 |
| Rebounds in the 3rd most recent game | 0.04871 |
| Rebounds in the 4th most recent game | 0.04002 |
| Blocks in the most recent game | 0.01014 |
| Blocks in the 2nd most recent game | 0.01315 |
| Blocks in the 3rd most recent game | 0.01010 |
| Blocks in the 4th most recent game | 0.01066 |
| Fouls in the most recent game | 0.02767 |
| Fouls in the 2nd most recent game | 0.03653 |
| Fouls in the 3rd most recent game | 0.02613 |
| Fouls in the 4th most recent game | 0.02668 |

Table 30: A Table showing the feature importance of our Random Forest assists model for A Tier players

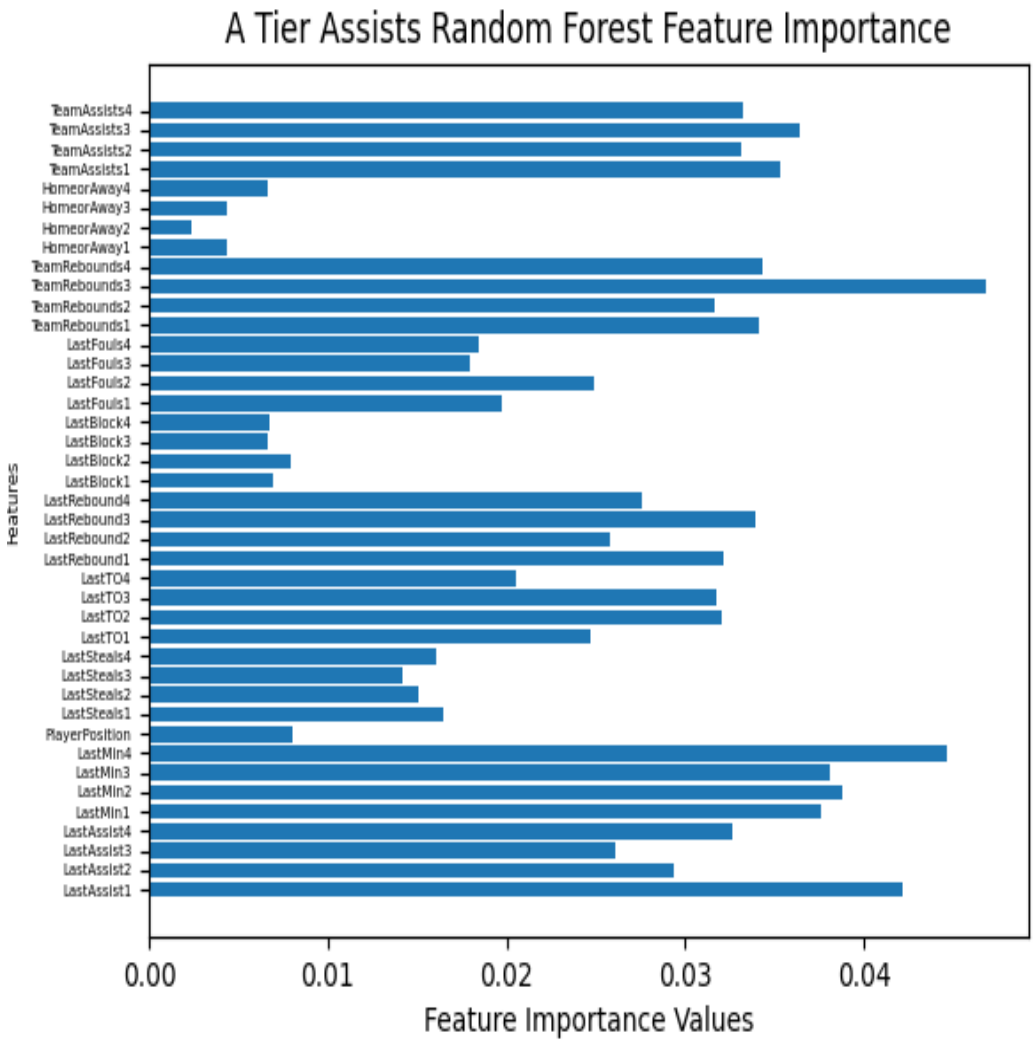


Figure 39: Our graph visualizing the feature importance for our A tier assists random forest model

| **Predictor** | **Predictor Coefficient** |
|---|---|
| Assists in the most recent game | 0.23883 |
| Assists in the 2nd most recent game | 0.06479 |
| Assists in the 3rd most recent game | 0.11329 |
| Assists in the 4th most recent game | 0.18067 |
| Minutes in the most recent game | 0.10313 |
| Minutes in the 2nd most recent game | 0.07116 |
| Minutes in the 3rd most recent game | 0.16353 |
| Minutes in the 4th most recent game | 0.01151 |
| Player position | -0.14307 |
| Steals in the most recent game | -0.00382 |
| Steals in the 2nd most recent game | -0.02274 |
| Steals in the 3rd most recent game | -0.07795 |
| Steals in the 4th most recent game | -0.00980 |
| Turnovers in the most recent game | 0.10037 |
| Turnovers in the 2nd most recent game | 0.24249 |
| Turnovers in the 3rd most recent game | -0.01543 |
| Turnovers in the 4th most recent game | 0.02509 |
| Rebounds in the most recent game | 0.04204 |
| Rebounds in the 2nd most recent game | -0.08675 |
| Rebounds in the 3rd most recent game | -0.08987 |
| Rebounds in the 4th most recent game | 0.04354 |
| Blocks in the most recent game | 0.07494 |
| Blocks in the 2nd most recent game | -0.05281 |
| Blocks in the 3rd most recent game | 0.04765 |
| Blocks in the 4th most recent game | 0.00102 |
| Fouls in the most recent game | -0.07359 |
| Fouls in the 2nd most recent game | -0.01128 |
| Fouls in the 3rd most recent game | -0.04212 |
| Fouls in the 4th most recent game | -0.04331 |

Table 31: A Table showing the predictor coefficients of our linear regression assists model for B Tier players

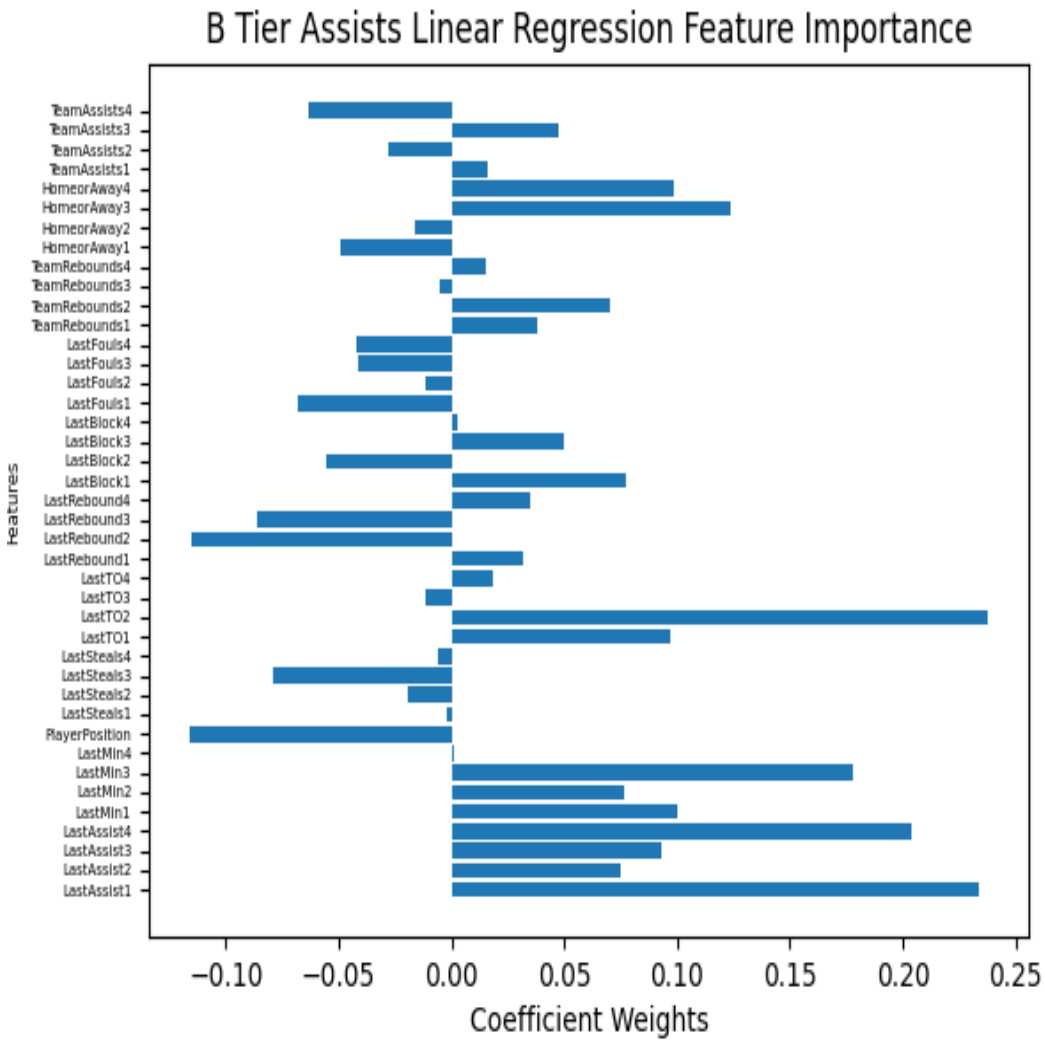


Figure 40: Our graph visualizing the feature importance for our B tier assists linear regression model

| Predictor | Predictor Coefficient |
|---|---|
| Assists in the most recent game | 0.20085 |
| Assists in the 2nd most recent game | 0.02653 |
| Assists in the 3rd most recent game | 0.07486 |
| Assists in the 4th most recent game | 0.13816 |
| Minutes in the most recent game | 0.08008 |
| Minutes in the 2nd most recent game | 0.00198 |
| Minutes in the 3rd most recent game | 0.06604 |
| Minutes in the 4th most recent game | 0.00000 |
| Player position | -0.02385 |
| Steals in the most recent game | 0.00000 |
| Steals in the 2nd most recent game | 0.00000 |
| Steals in the 3rd most recent game | 0.00000 |
| Steals in the 4th most recent game | 0.00000 |
| Turnovers in the most recent game | 0.01837 |
| Turnovers in the 2nd most recent game | 0.16006 |
| Turnovers in the 3rd most recent game | 0.00000 |
| Turnovers in the 4th most recent game | 0.00000 |
| Rebounds in the most recent game | 0.00000 |
| Rebounds in the 2nd most recent game | 0.00000 |
| Rebounds in the 3rd most recent game | 0.00000 |
| Rebounds in the 4th most recent game | 0.00000 |
| Blocks in the most recent game | 0.00000 |
| Blocks in the 2nd most recent game | 0.00000 |
| Blocks in the 3rd most recent game | 0.00000 |
| Blocks in the 4th most recent game | 0.00000 |
| Fouls in the most recent game | 0.00000 |
| Fouls in the 2nd most recent game | 0.00000 |
| Fouls in the 3rd most recent game | 0.00000 |
| Fouls in the 4th most recent game | 0.00000 |

Table 32: A Table showing the predictor coefficient of our Lasso regression assists model for B Tier players

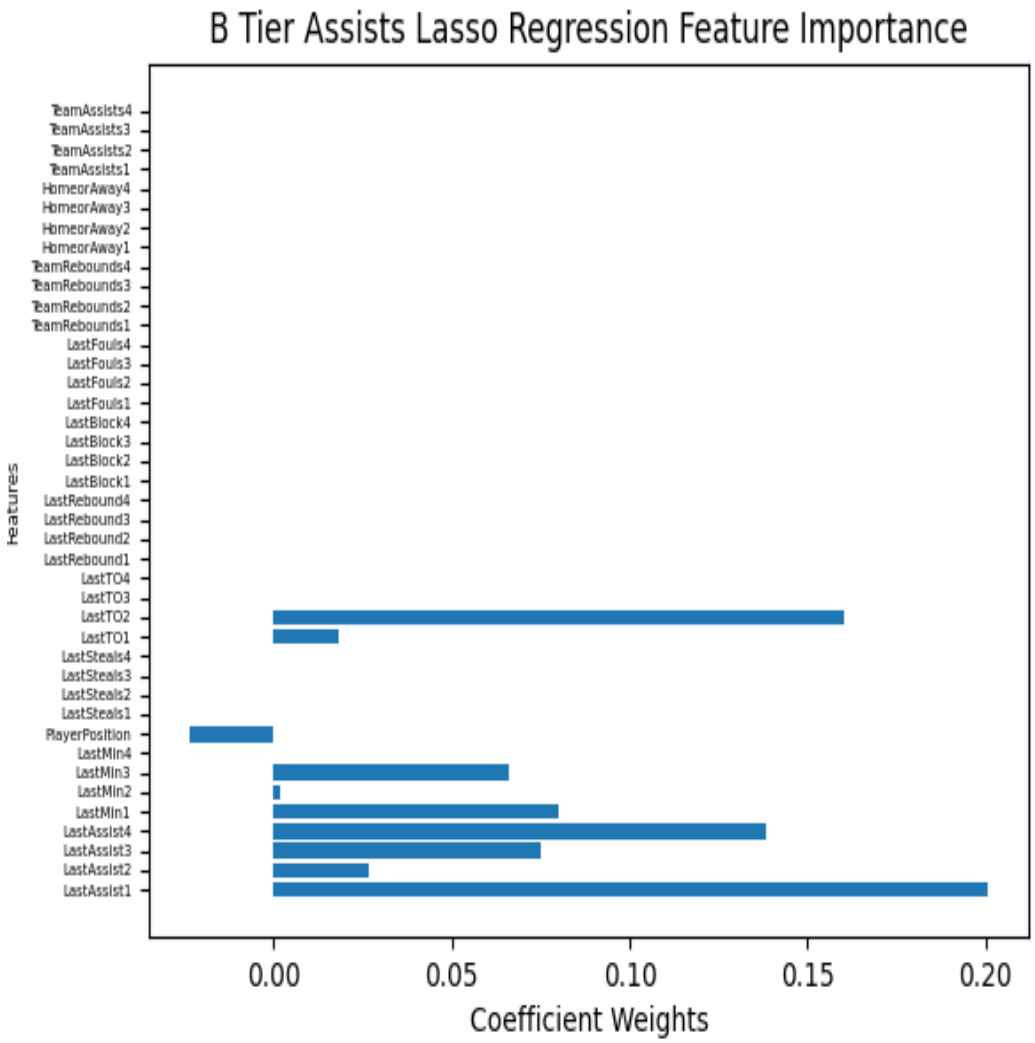


Figure 41: Our graph visualizing the feature importance for our B tier assists Lasso regression model

| Predictor | Random Forest Feature Importance |
|---|---|
| Assists in the most recent game | 0.05062 |
| Assists in the 2nd most recent game | 0.04284 |
| Assists in the 3rd most recent game | 0.04023 |
| Assists in the 4th most recent game | 0.04692 |
| Minutes in the most recent game | 0.06957 |
| Minutes in the 2nd most recent game | 0.06175 |
| Minutes in the 3rd most recent game | 0.06582 |
| Minutes in the 4th most recent game | 0.06935 |
| Player position | 0.01676 |
| Steals in the most recent game | 0.02103 |
| Steals in the 2nd most recent game | 0.02170 |
| Steals in the 3rd most recent game | 0.01970 |
| Steals in the 4th most recent game | 0.02111 |
| Turnovers in the most recent game | 0.02993 |
| Turnovers in the 2nd most recent game | 0.03876 |
| Turnovers in the 3rd most recent game | 0.02925 |
| Turnovers in the 4th most recent game | 0.02874 |
| Rebounds in the most recent game | 0.03743 |
| Rebounds in the 2nd most recent game | 0.03893 |
| Rebounds in the 3rd most recent game | 0.04479 |
| Rebounds in the 4th most recent game | 0.03847 |
| Blocks in the most recent game | 0.01463 |
| Blocks in the 2nd most recent game | 0.01238 |
| Blocks in the 3rd most recent game | 0.01185 |
| Blocks in the 4th most recent game | 0.01110 |
| Fouls in the most recent game | 0.03094 |
| Fouls in the 2nd most recent game | 0.02684 |
| Fouls in the 3rd most recent game | 0.02902 |
| Fouls in the 4th most recent game | 0.02957 |

Table 33: A Table showing the feature importance of our Random Forest assists model for B Tier players

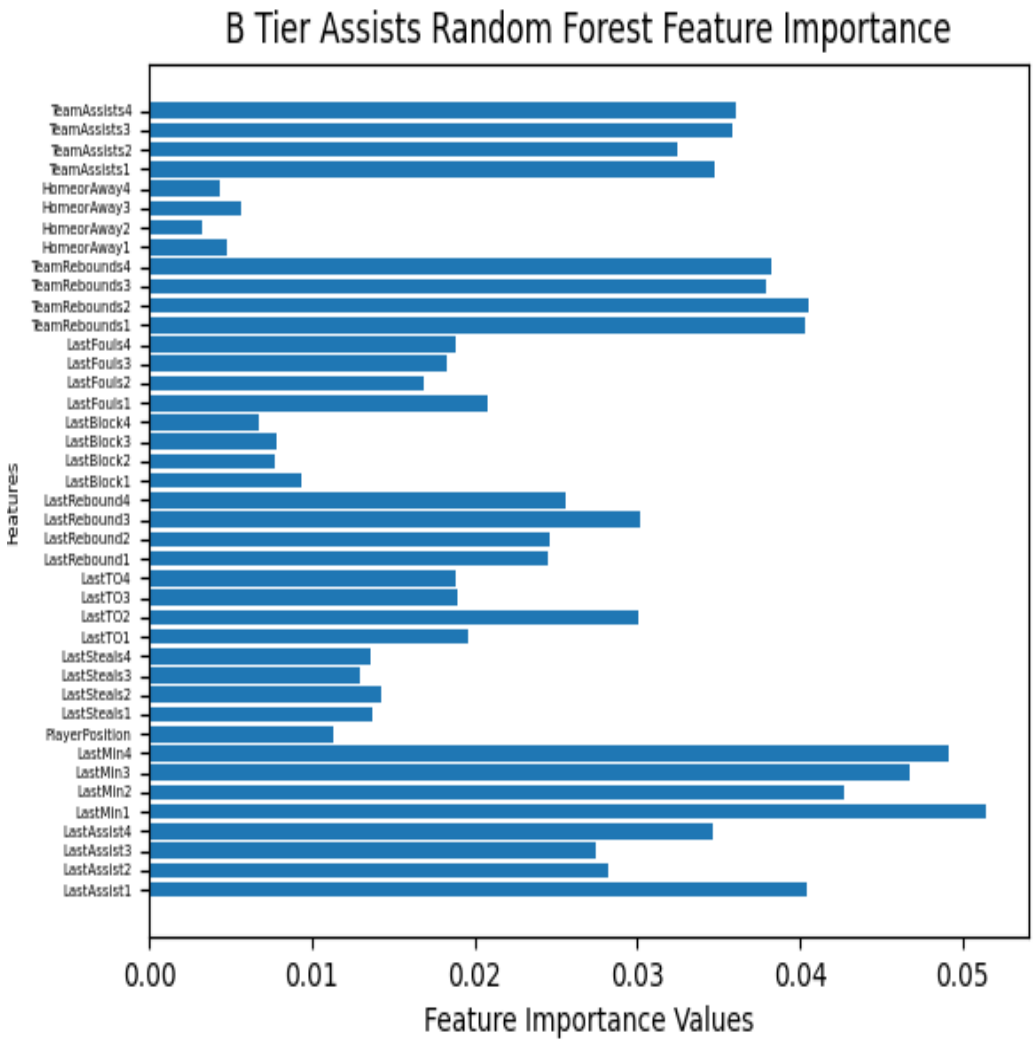


Figure 42: Our graph visualizing the feature importance for our B tier assists random forest model

| **Predictor** | **Predictor Coefficient** |
|---|---|
| Assists in the most recent game | 0.07408 |
| Assists in the 2nd most recent game | 0.10642 |
| Assists in the 3rd most recent game | 0.05396 |
| Assists in the 4th most recent game | 0.02993 |
| Minutes in the most recent game | 0.04827 |
| Minutes in the 2nd most recent game | 0.05739 |
| Minutes in the 3rd most recent game | 0.00632 |
| Minutes in the 4th most recent game | -0.10974 |
| Player position | -0.15045 |
| Steals in the most recent game | 0.02928 |
| Steals in the 2nd most recent game | 0.01654 |
| Steals in the 3rd most recent game | -0.03014 |
| Steals in the 4th most recent game | 0.03460 |
| Turnovers in the most recent game | 0.02931 |
| Turnovers in the 2nd most recent game | 0.00418 |
| Turnovers in the 3rd most recent game | -0.01137 |
| Turnovers in the 4th most recent game | -0.04312 |
| Rebounds in the most recent game | -0.04192 |
| Rebounds in the 2nd most recent game | -0.02033 |
| Rebounds in the 3rd most recent game | 0.09836 |
| Rebounds in the 4th most recent game | 0.11210 |
| Blocks in the most recent game | 0.03990 |
| Blocks in the 2nd most recent game | -0.00704 |
| Blocks in the 3rd most recent game | -0.01589 |
| Blocks in the 4th most recent game | -0.05205 |
| Fouls in the most recent game | 0.06787 |
| Fouls in the 2nd most recent game | -0.02251 |
| Fouls in the 3rd most recent game | 0.01571 |
| Fouls in the 4th most recent game | -0.02205 |

Table 34: A Table showing the predictor coefficients of our linear regression assists model for C Tier players

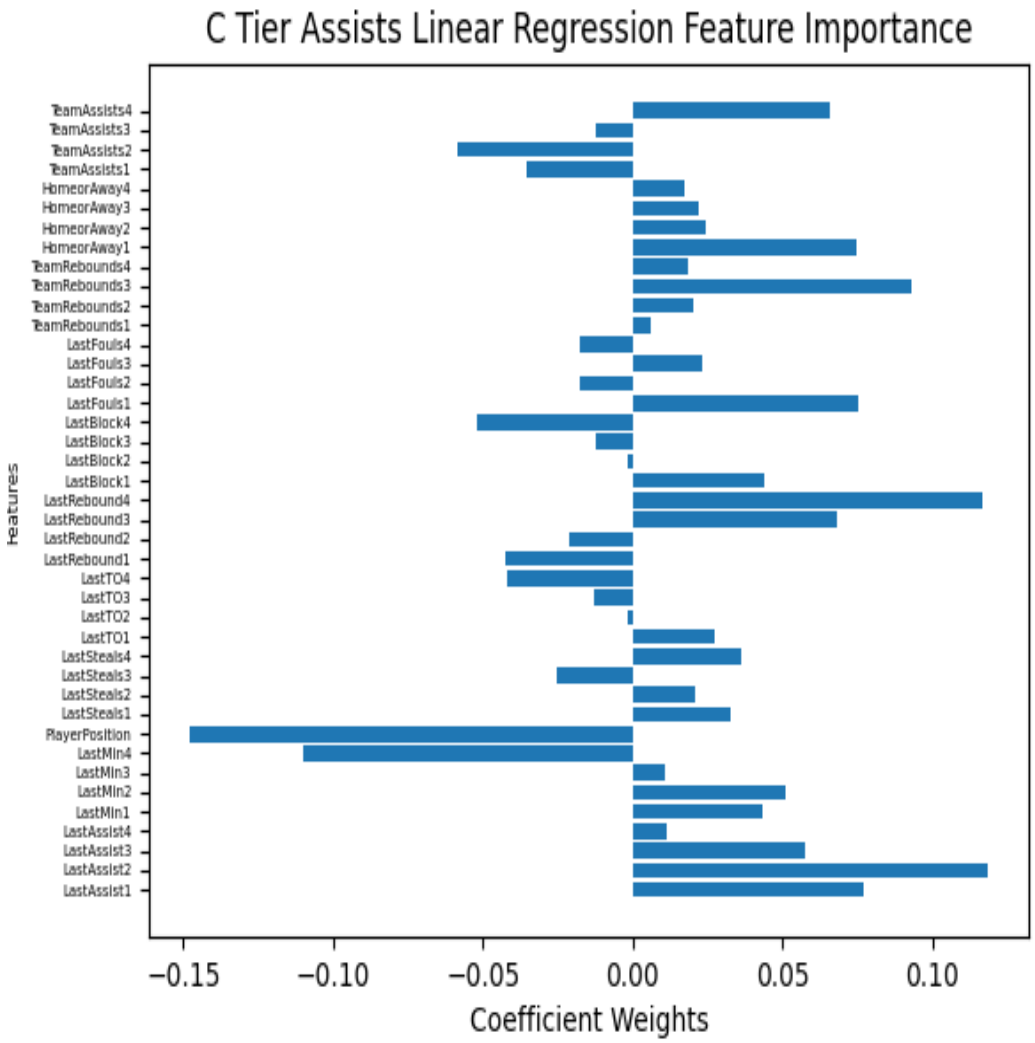


Figure 43: Our graph visualizing the feature importance for our C tier assists linear regression model

| Predictor | Predictor Coefficient |
|---|---|
| Assists in the most recent game | 0.01289 |
| Assists in the 2nd most recent game | 0.03437 |
| Assists in the 3rd most recent game | 0.00000 |
| Assists in the 4th most recent game | 0.00000 |
| Minutes in the most recent game | 0.00000 |
| Minutes in the 2nd most recent game | 0.00000 |
| Minutes in the 3rd most recent game | 0.00000 |
| Minutes in the 4th most recent game | 0.00000 |
| Player position | 0.00000 |
| Steals in the most recent game | 0.00000 |
| Steals in the 2nd most recent game | 0.00000 |
| Steals in the 3rd most recent game | 0.00000 |
| Steals in the 4th most recent game | 0.00000 |
| Turnovers in the most recent game | 0.00000 |
| Turnovers in the 2nd most recent game | 0.00000 |
| Turnovers in the 3rd most recent game | 0.00000 |
| Turnovers in the 4th most recent game | 0.00000 |
| Rebounds in the most recent game | 0.00000 |
| Rebounds in the 2nd most recent game | 0.00000 |
| Rebounds in the 3rd most recent game | 0.00000 |
| Rebounds in the 4th most recent game | 0.00000 |
| Blocks in the most recent game | 0.00000 |
| Blocks in the 2nd most recent game | 0.00000 |
| Blocks in the 3rd most recent game | 0.00000 |
| Blocks in the 4th most recent game | 0.00000 |
| Fouls in the most recent game | 0.00000 |
| Fouls in the 2nd most recent game | 0.00000 |
| Fouls in the 3rd most recent game | 0.00000 |
| Fouls in the 4th most recent game | 0.00000 |

Table 35: A Table showing the predictor coefficient of our Lasso regression assists model for C Tier players

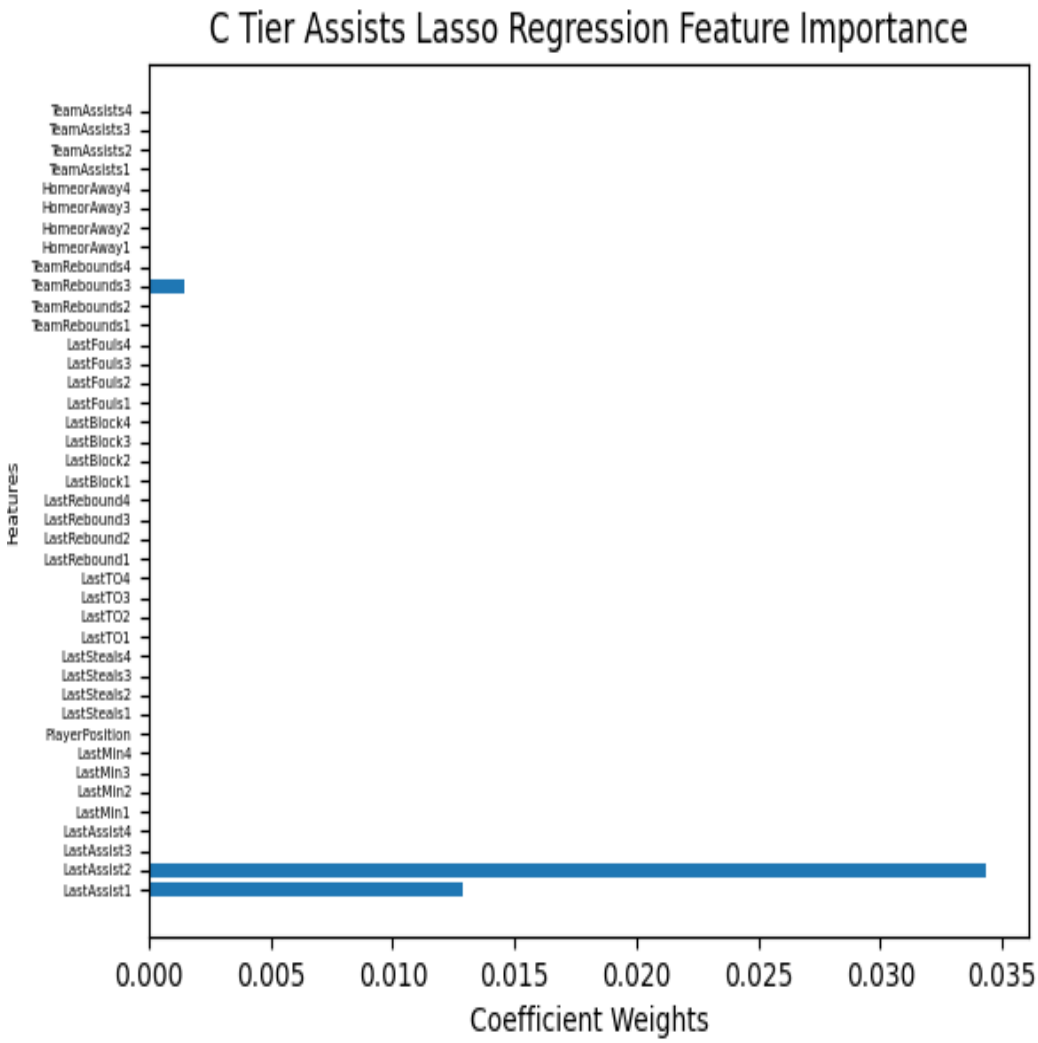


Figure 44: Our graph visualizing the feature importance for our C tier assists Lasso regression model

| Predictor | Random Forest Feature Importance |
|---|---|
| Assists in the most recent game | 0.04150 |
| Assists in the 2nd most recent game | 0.03973 |
| Assists in the 3rd most recent game | 0.03497 |
| Assists in the 4th most recent game | 0.03558 |
| Minutes in the most recent game | 0.07083 |
| Minutes in the 2nd most recent game | 0.06562 |
| Minutes in the 3rd most recent game | 0.06521 |
| Minutes in the 4th most recent game | 0.07063 |
| Player position | 0.01942 |
| Steals in the most recent game | 0.02197 |
| Steals in the 2nd most recent game | 0.02550 |
| Steals in the 3rd most recent game | 0.01938 |
| Steals in the 4th most recent game | 0.02162 |
| Turnovers in the most recent game | 0.03241 |
| Turnovers in the 2nd most recent game | 0.02765 |
| Turnovers in the 3rd most recent game | 0.02742 |
| Turnovers in the 4th most recent game | 0.02995 |
| Rebounds in the most recent game | 0.04046 |
| Rebounds in the 2nd most recent game | 0.04670 |
| Rebounds in the 3rd most recent game | 0.04228 |
| Rebounds in the 4th most recent game | 0.04194 |
| Blocks in the most recent game | 0.01188 |
| Blocks in the 2nd most recent game | 0.01187 |
| Blocks in the 3rd most recent game | 0.01261 |
| Blocks in the 4th most recent game | 0.01322 |
| Fouls in the most recent game | 0.03230 |
| Fouls in the 2nd most recent game | 0.03185 |
| Fouls in the 3rd most recent game | 0.03406 |
| Fouls in the 4th most recent game | 0.03144 |

Table 36: A Table showing the feature importance of our Random Forest assists model for C Tier players

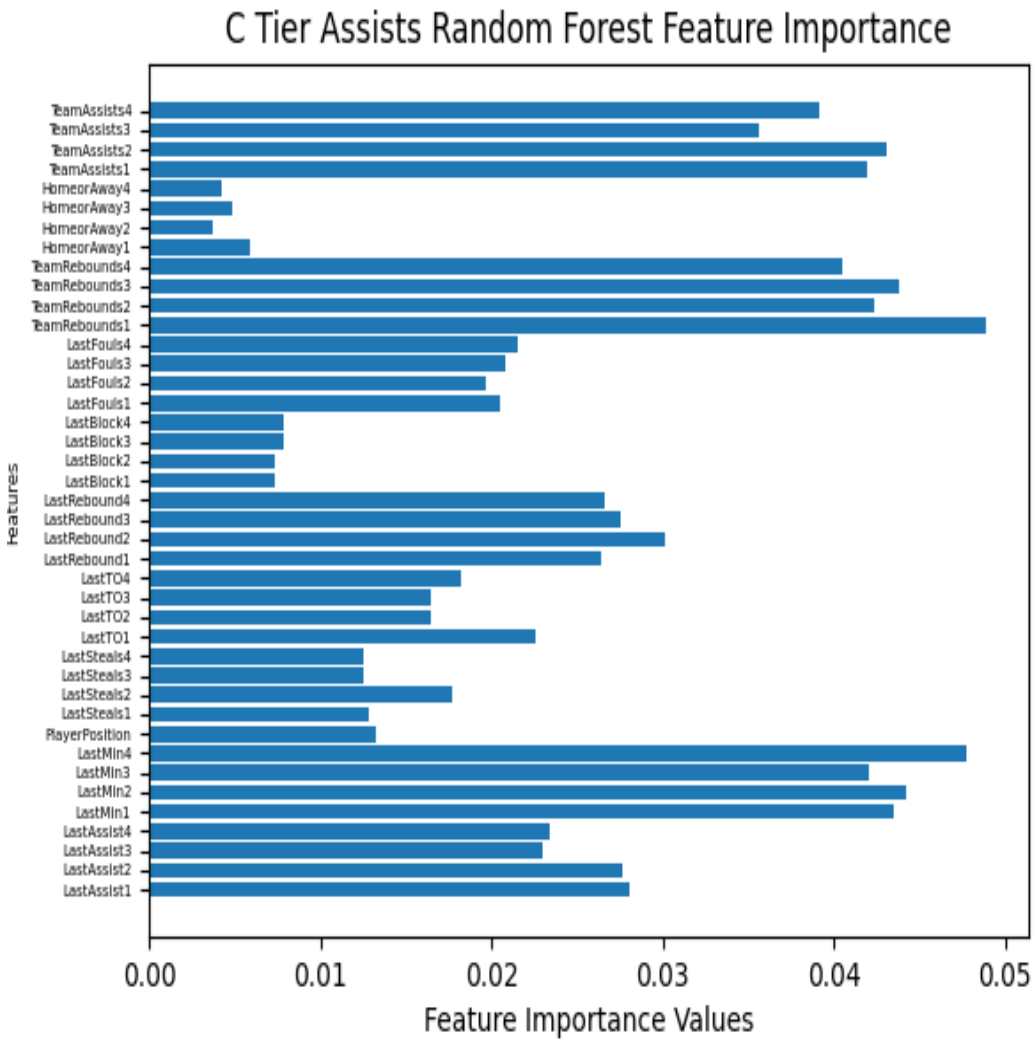


Figure 45: Our graph visualizing the feature importance for our C tier assists random forest model

# 12 Appendix D - Additional Rebounds Results

| Predictor | Predictor Coefficient |
|---|---|
| Rebounds in the most recent game | 0.00000 |
| Rebounds in the 2nd most recent game | 0.61850 |
| Rebounds in the 3rd most recent game | 0.24132 |
| Rebounds in the 4th most recent game | 0.26618 |
| Assists in the most recent game | 0.00108 |
| Assists in the 2nd most recent game | 0.00000 |
| Assists in the 3rd most recent game | 0.00000 |
| Assists in the 4th most recent game | 0.00000 |
| Minutes in the most recent game | 0.24569 |
| Minutes in the 2nd most recent game | -0.11180 |
| Minutes in the 3rd most recent game | 0.00000 |
| Minutes in the 4th most recent game | 0.00000 |
| Player position | 0.41255 |
| Steals in the most recent game | 0.00000 |
| Steals in the 2nd most recent game | 0.22606 |
| Steals in the 3rd most recent game | 0.13299 |
| Steals in the 4th most recent game | 0.00000 |
| Turnovers in the most recent game | 0.02829 |
| Turnovers in the 2nd most recent game | 0.00000 |
| Turnovers in the 3rd most recent game | 0.11850 |
| Turnovers in the 4th most recent game | 0.00000 |
| Blocks in the most recent game | 0.03469 |
| Blocks in the 2nd most recent game | 0.25681 |
| Blocks in the 3rd most recent game | 0.15132 |
| Blocks in the 4th most recent game | 0.25885 |
| Fouls in the most recent game | 0.00000 |
| Fouls in the 2nd most recent game | -0.04516 |
| Fouls in the 3rd most recent game | 0.00000 |
| Fouls in the 4th most recent game | 0.00000 |

Table 37: A Table showing the predictor coefficients of our Lasso regression rebounds model for A Tier players

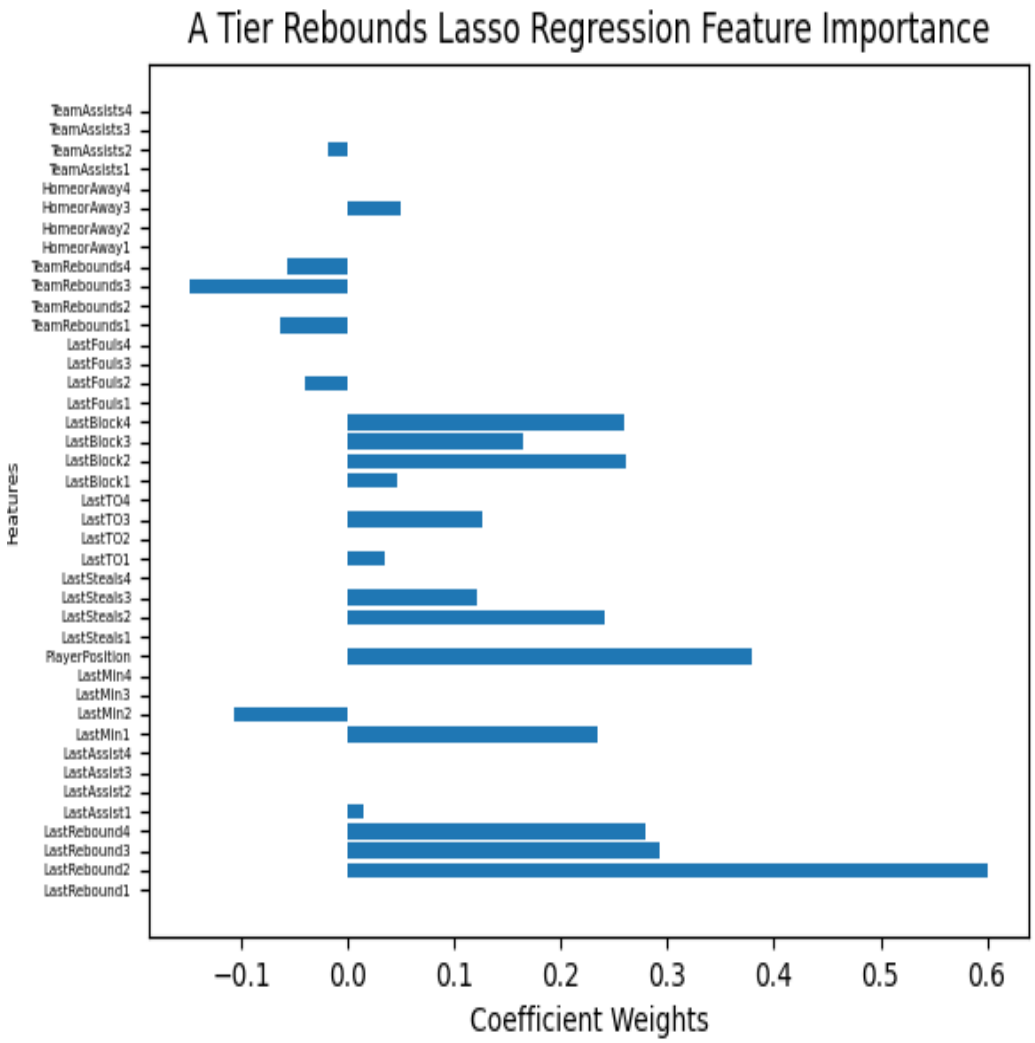


Figure 46: Our graph visualizing the feature importance for our A tier rebounds Lasso regression model

| Predictor | Predictor Coefficient |
|---|---|
| Rebounds in the most recent game | 0.32979 |
| Rebounds in the 2nd most recent game | 0.13222 |
| Rebounds in the 3rd most recent game | 0.05348 |
| Rebounds in the 4th most recent game | 0.15538 |
| Assists in the most recent game | 0.00000 |
| Assists in the 2nd most recent game | 0.00000 |
| Assists in the 3rd most recent game | 0.00000 |
| Assists in the 4th most recent game | 0.01645 |
| Minutes in the most recent game | 0.00000 |
| Minutes in the 2nd most recent game | 0.00000 |
| Minutes in the 3rd most recent game | 0.11017 |
| Minutes in the 4th most recent game | 0.06485 |
| Player position | 0.13367 |
| Steals in the most recent game | 0.00000 |
| Steals in the 2nd most recent game | 0.00000 |
| Steals in the 3rd most recent game | 0.00000 |
| Steals in the 4th most recent game | 0.00000 |
| Turnovers in the most recent game | 0.05173 |
| Turnovers in the 2nd most recent game | 0.00000 |
| Turnovers in the 3rd most recent game | 0.00000 |
| Turnovers in the 4th most recent game | 0.00000 |
| Blocks in the most recent game | 0.00000 |
| Blocks in the 2nd most recent game | 0.00000 |
| Blocks in the 3rd most recent game | 0.00000 |
| Blocks in the 4th most recent game | 0.00000 |
| Fouls in the most recent game | 0.01915 |
| Fouls in the 2nd most recent game | 0.00000 |
| Fouls in the 3rd most recent game | 0.00000 |
| Fouls in the 4th most recent game | 0.00000 |

Table 38: A Table showing the predictor coefficient of our Lasso regression rebounds model for B Tier players

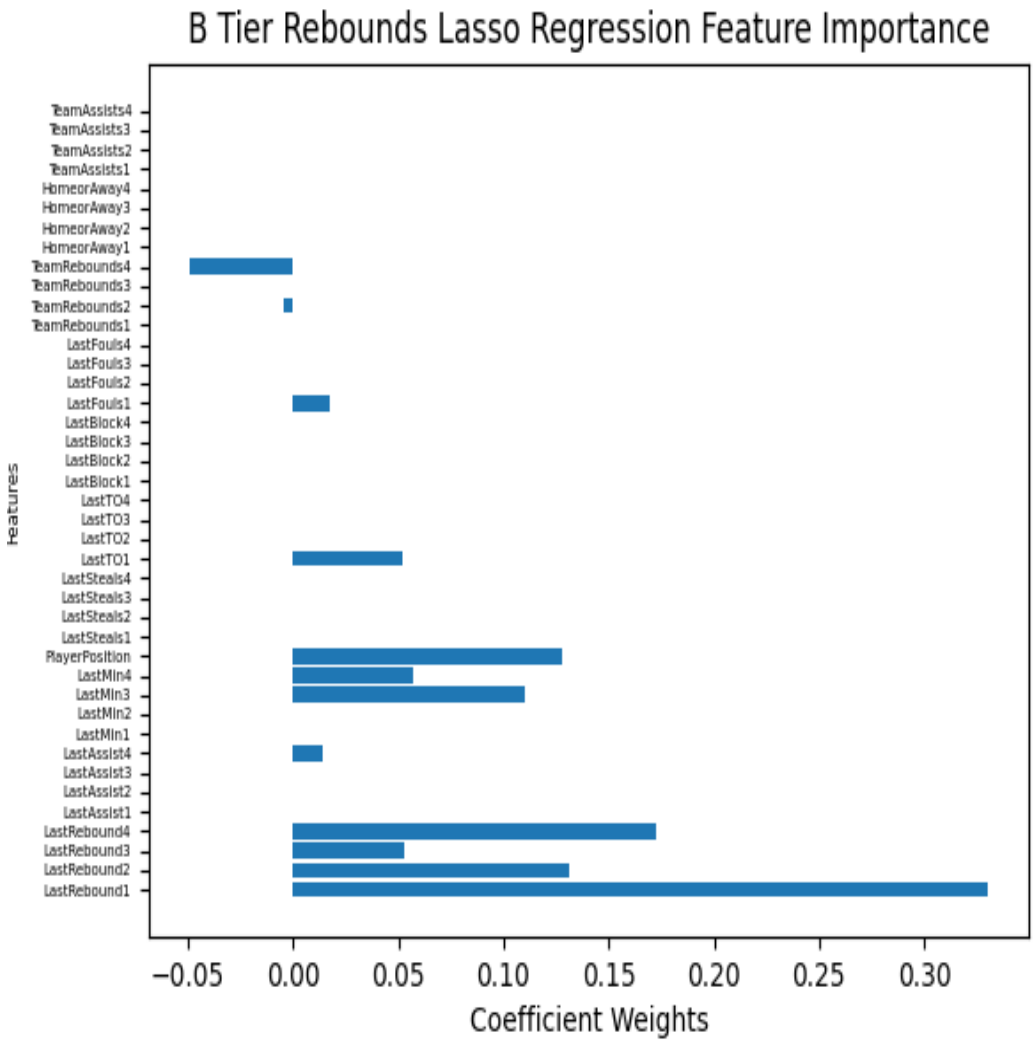


Figure 47: Our graph visualizing the feature importance for our B tier rebounds Lasso regression model

| Predictor | Random Forest Feature Importance |
|---|---|
| Rebounds in the most recent game | 0.05793 |
| Rebounds in the 2nd most recent game | 0.05145 |
| Rebounds in the 3rd most recent game | 0.04905 |
| Rebounds in the 4th most recent game | 0.05029 |
| Assists in the most recent game | 0.03039 |
| Assists in the 2nd most recent game | 0.03185 |
| Assists in the 3rd most recent game | 0.03119 |
| Assists in the 4th most recent game | 0.02940 |
| Minutes in the most recent game | 0.05853 |
| Minutes in the 2nd most recent game | 0.06551 |
| Minutes in the 3rd most recent game | 0.06351 |
| Minutes in the 4th most recent game | 0.06500 |
| Player position | 0.01918 |
| Steals in the most recent game | 0.01830 |
| Steals in the 2nd most recent game | 0.02070 |
| Steals in the 3rd most recent game | 0.02226 |
| Steals in the 4th most recent game | 0.02075 |
| Turnovers in the most recent game | 0.02607 |
| Turnovers in the 2nd most recent game | 0.02946 |
| Turnovers in the 3rd most recent game | 0.02458 |
| Turnovers in the 4th most recent game | 0.02854 |
| Blocks in the most recent game | 0.01854 |
| Blocks in the 2nd most recent game | 0.02052 |
| Blocks in the 3rd most recent game | 0.02237 |
| Blocks in the 4th most recent game | 0.02042 |
| Fouls in the most recent game | 0.03509 |
| Fouls in the 2nd most recent game | 0.03181 |
| Fouls in the 3rd most recent game | 0.02907 |
| Fouls in the 4th most recent game | 0.02823 |

Table 39: A Table showing the feature importance of our Random Forest rebounds model for B Tier players

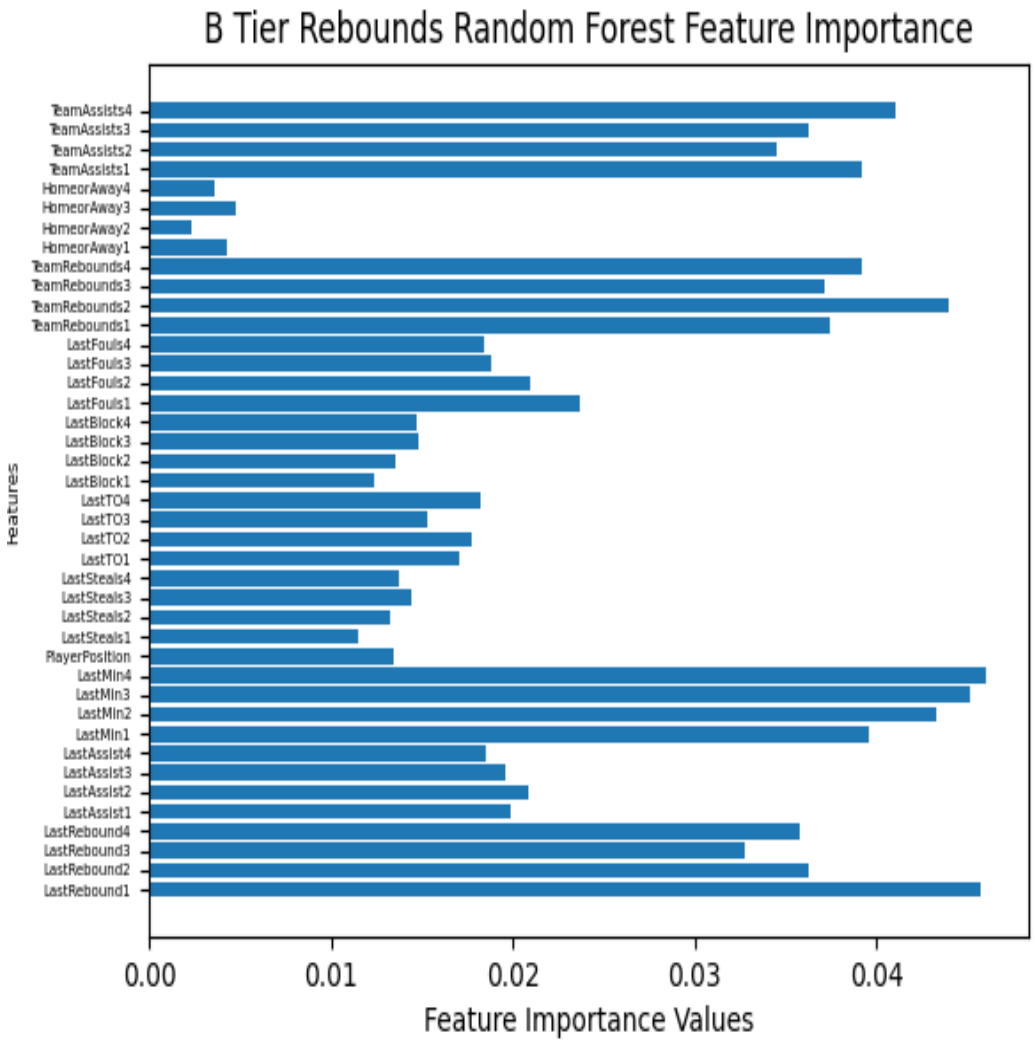


Figure 48: Our graph visualizing the feature importance for our B tier rebounds random forest model

| **Predictor** | **Predictor Coefficient** |
|---|---|
| Rebounds in the most recent game | 0.10304 |
| Rebounds in the 2nd most recent game | 0.00000 |
| Rebounds in the 3rd most recent game | 0.00000 |
| Rebounds in the 4th most recent game | 0.00427 |
| Assists in the most recent game | 0.00000 |
| Assists in the 2nd most recent game | 0.00000 |
| Assists in the 3rd most recent game | 0.00000 |
| Assists in the 4th most recent game | 0.00000 |
| Minutes in the most recent game | 0.00000 |
| Minutes in the 2nd most recent game | 0.00000 |
| Minutes in the 3rd most recent game | 0.01235 |
| Minutes in the 4th most recent game | 0.00000 |
| Player position | 0.00000 |
| Steals in the most recent game | 0.00000 |
| Steals in the 2nd most recent game | 0.00000 |
| Steals in the 3rd most recent game | 0.00000 |
| Steals in the 4th most recent game | 0.00000 |
| Turnovers in the most recent game | 0.00000 |
| Turnovers in the 2nd most recent game | 0.00000 |
| Turnovers in the 3rd most recent game | 0.00000 |
| Turnovers in the 4th most recent game | 0.00000 |
| Blocks in the most recent game | 0.02894 |
| Blocks in the 2nd most recent game | 0.00000 |
| Blocks in the 3rd most recent game | 0.00000 |
| Blocks in the 4th most recent game | 0.00000 |
| Fouls in the most recent game | 0.04667 |
| Fouls in the 2nd most recent game | 0.00000 |
| Fouls in the 3rd most recent game | 0.00000 |
| Fouls in the 4th most recent game | 0.00000 |

Table 40: A Table showing the predictor coefficient of our Lasso regression rebounds model for C Tier players

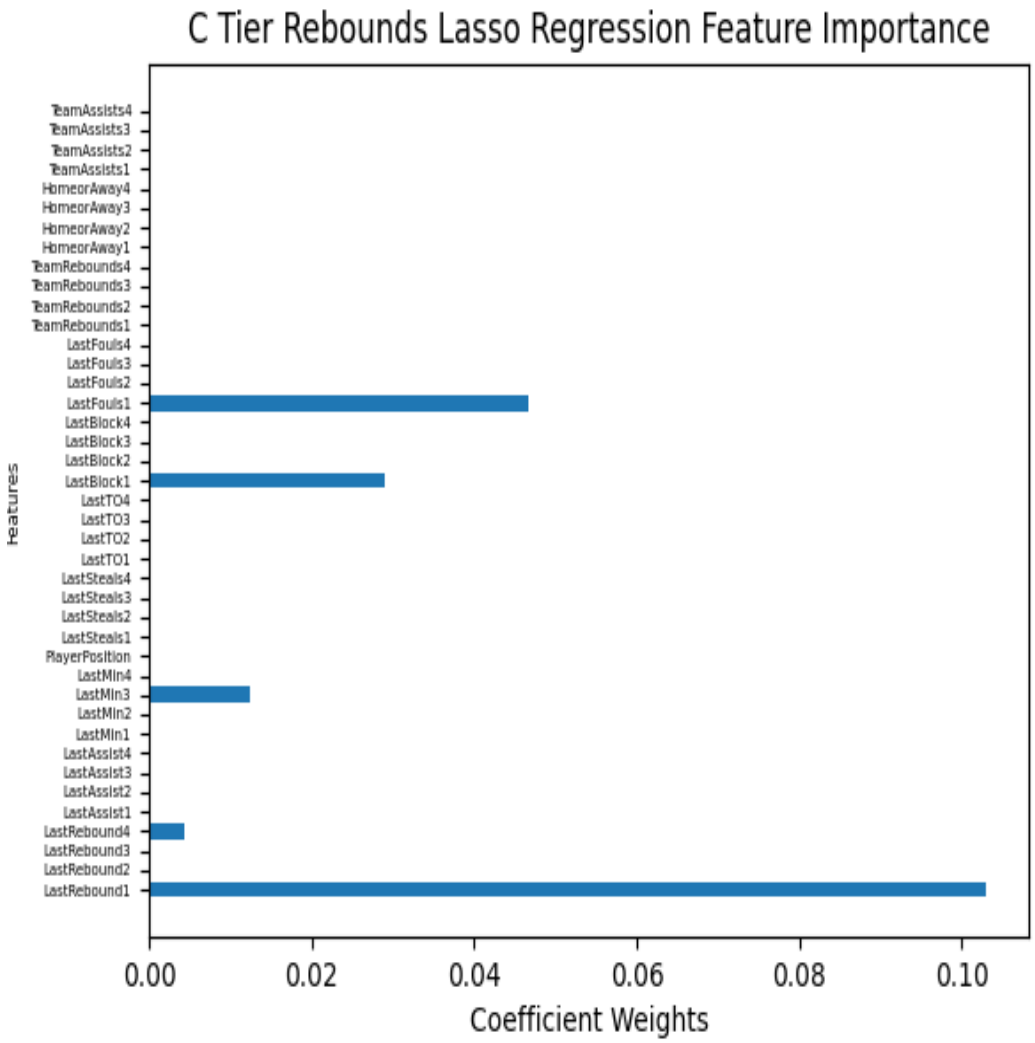


Figure 49: Our graph visualizing the feature importance for our C tier rebounds Lasso regression model

| Predictor | Random Forest Feature Importance |
|---|---|
| Rebounds in the most recent game | 0.04688 |
| Rebounds in the 2nd most recent game | 0.04851 |
| Rebounds in the 3rd most recent game | 0.04345 |
| Rebounds in the 4th most recent game | 0.04519 |
| Assists in the most recent game | 0.03308 |
| Assists in the 2nd most recent game | 0.02976 |
| Assists in the 3rd most recent game | 0.02895 |
| Assists in the 4th most recent game | 0.03037 |
| Minutes in the most recent game | 0.06802 |
| Minutes in the 2nd most recent game | 0.07147 |
| Minutes in the 3rd most recent game | 0.07444 |
| Minutes in the 4th most recent game | 0.06924 |
| Player position | 0.01946 |
| Steals in the most recent game | 0.02118 |
| Steals in the 2nd most recent game | 0.01887 |
| Steals in the 3rd most recent game | 0.01887 |
| Steals in the 4th most recent game | 0.01939 |
| Turnovers in the most recent game | 0.02696 |
| Turnovers in the 2nd most recent game | 0.02374 |
| Turnovers in the 3rd most recent game | 0.02455 |
| Turnovers in the 4th most recent game | 0.02751 |
| Blocks in the most recent game | 0.02064 |
| Blocks in the 2nd most recent game | 0.01868 |
| Blocks in the 3rd most recent game | 0.01769 |
| Blocks in the 4th most recent game | 0.01943 |
| Fouls in the most recent game | 0.03203 |
| Fouls in the 2nd most recent game | 0.03597 |
| Fouls in the 3rd most recent game | 0.03123 |
| Fouls in the 4th most recent game | 0.03442 |

Table 41: A Table showing the feature importance of our Random Forest rebounds model for C Tier players

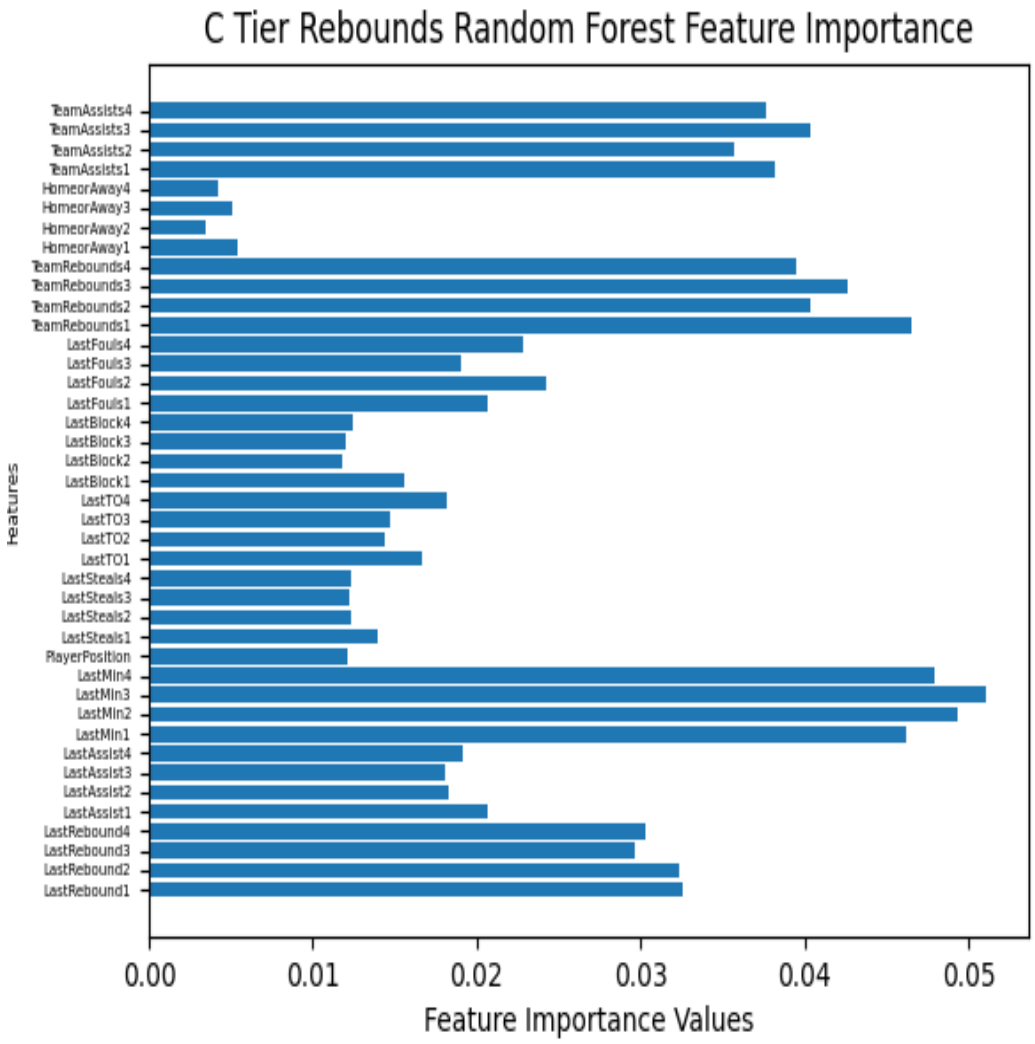


Figure 50: Our graph visualizing the feature importance for our C tier rebounds random forest model

| Predictor | Predictor Coefficient |
|---|---|
| Rebounds in the most recent game | 0.38480 |
| Rebounds in the 2nd most recent game | 0.20278 |
| Rebounds in the 3rd most recent game | 0.08093 |
| Rebounds in the 4th most recent game | 0.18450 |
| Assists in the most recent game | 0.06880 |
| Assists in the 2nd most recent game | -0.01284 |
| Assists in the 3rd most recent game | -0.02140 |
| Assists in the 4th most recent game | 0.09709 |
| Minutes in the most recent game | -0.02782 |
| Minutes in the 2nd most recent game | -0.01537 |
| Minutes in the 3rd most recent game | 0.20899 |
| Minutes in the 4th most recent game | 0.12108 |
| Player position | 0.38205 |
| Steals in the most recent game | 0.01140 |
| Steals in the 2nd most recent game | 0.00797 |
| Steals in the 3rd most recent game | 0.08319 |
| Steals in the 4th most recent game | 0.05837 |
| Turnovers in the most recent game | 0.12948 |
| Turnovers in the 2nd most recent game | 0.01404 |
| Turnovers in the 3rd most recent game | -0.07945 |
| Turnovers in the 4th most recent game | 0.03365 |
| Blocks in the most recent game | -0.06450 |
| Blocks in the 2nd most recent game | 0.00747 |
| Blocks in the 3rd most recent game | -0.00924 |
| Blocks in the 4th most recent game | 0.05148 |
| Fouls in the most recent game | 0.10176 |
| Fouls in the 2nd most recent game | -0.12620 |
| Fouls in the 3rd most recent game | -0.03522 |
| Fouls in the 4th most recent game | -0.05495 |

Table 42: A Table showing the predictor coefficients of our linear regression rebounds model for B Tier players

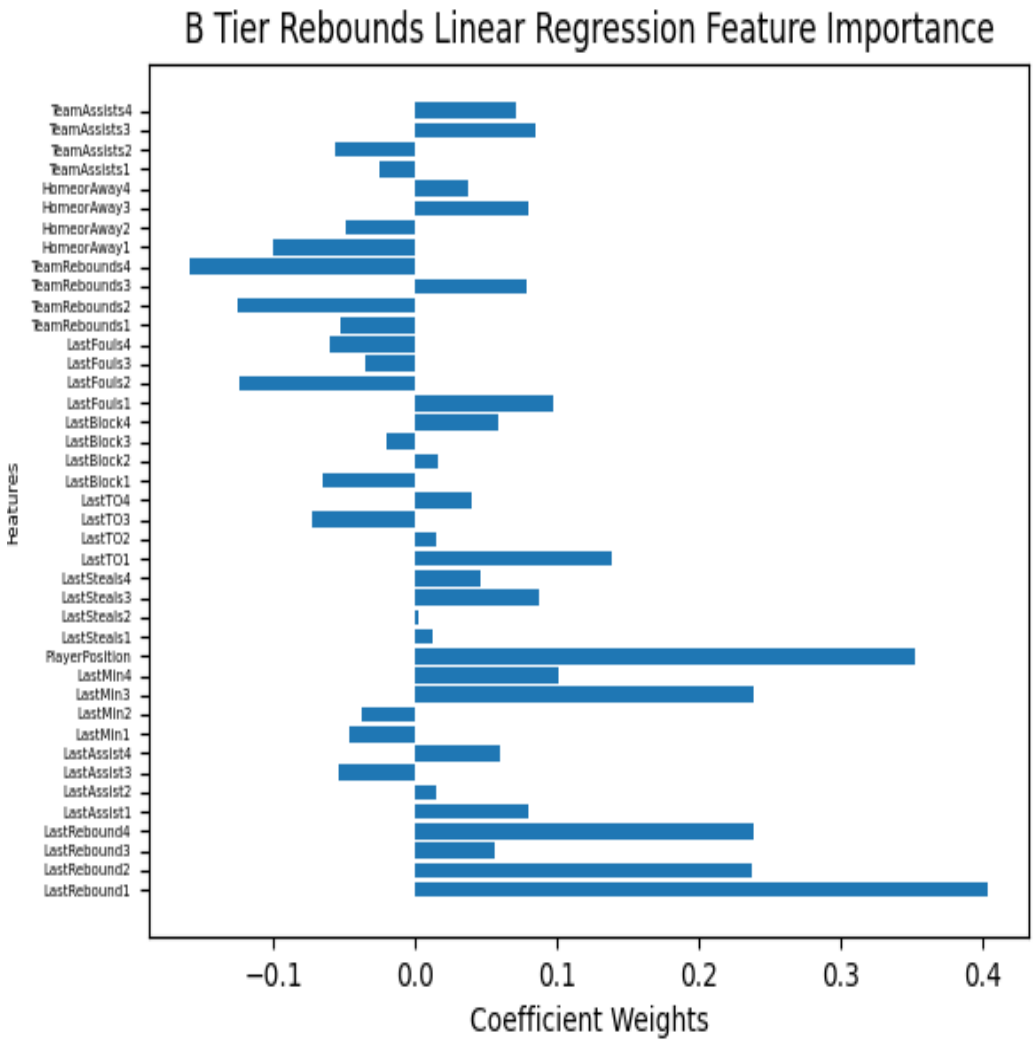


Figure 51: Our graph visualizing the feature importance for our B tier rebounds linear regression model

| Predictor | Predictor Coefficient |
|---|---|
| Rebounds in the most recent game | 0.19177 |
| Rebounds in the 2nd most recent game | -0.02924 |
| Rebounds in the 3rd most recent game | 0.00790 |
| Rebounds in the 4th most recent game | 0.09814 |
| Assists in the most recent game | 0.12077 |
| Assists in the 2nd most recent game | -0.02103 |
| Assists in the 3rd most recent game | -0.09686 |
| Assists in the 4th most recent game | 0.11934 |
| Minutes in the most recent game | -0.04913 |
| Minutes in the 2nd most recent game | 0.02369 |
| Minutes in the 3rd most recent game | 0.21553 |
| Minutes in the 4th most recent game | -0.00255 |
| Player position | 0.10534 |
| Steals in the most recent game | -0.11122 |
| Steals in the 2nd most recent game | 0.08539 |
| Steals in the 3rd most recent game | -0.02025 |
| Steals in the 4th most recent game | -0.11680 |
| Turnovers in the most recent game | 0.00931 |
| Turnovers in the 2nd most recent game | -0.06552 |
| Turnovers in the 3rd most recent game | -0.00874 |
| Turnovers in the 4th most recent game | -0.03581 |
| Blocks in the most recent game | 0.11142 |
| Blocks in the 2nd most recent game | 0.02872 |
| Blocks in the 3rd most recent game | -0.03642 |
| Blocks in the 4th most recent game | -0.02214 |
| Fouls in the most recent game | 0.12657 |
| Fouls in the 2nd most recent game | 0.04343 |
| Fouls in the 3rd most recent game | -0.04942 |
| Fouls in the 4th most recent game | -0.04659 |

Table 43: A Table showing the predictor coefficients of our linear regression rebounds model for C Tier players

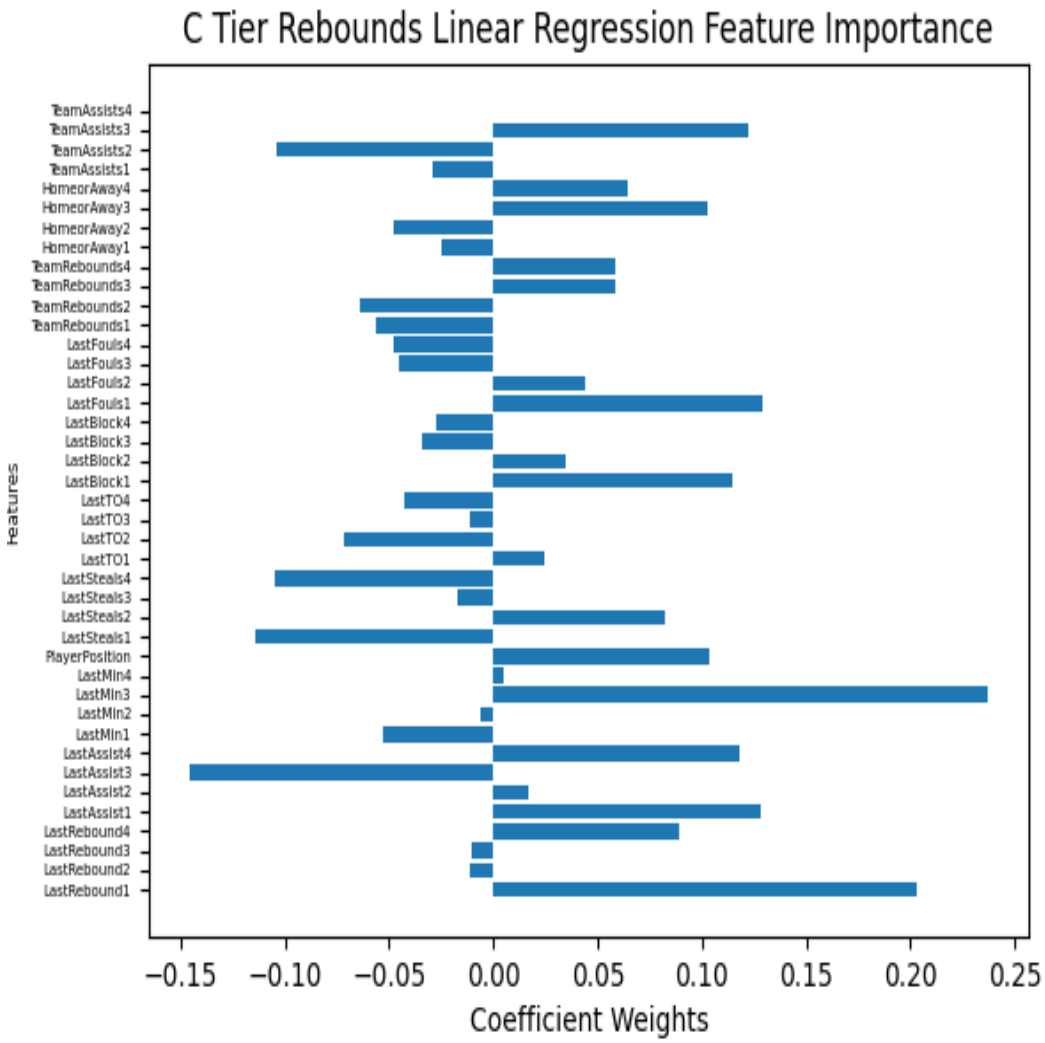


Figure 52: Our graph visualizing the feature importance for our C tier Rebounds linear regression model

# References


[1] Miller, E., & Davidow, M. (2019). The logic of sports betting. Henderson, Nevada: Ed Miller.

[2] Petrella, S. (2020, September 15). *Prop Bet in Sports Betting: Definition, Examples, More.* Action Network. https://www.actionnetwork.com/education/prop-bet

[3] Sports-King. (n.d.). *Definition of Opening Line.* https://www.sports-king.com/dictionary.php?q=opening-line#:~:text=Definition%20of%20Opening%20Line&text=The%20%22opening%20line%22%20is%20the,as%20the%20%22opening%20line%22.

[4] Donahue, D. (2020, March 3). *The Importance of Closing Line Value in Sports Betting.* Action Network. https://www.actionnetwork.com/how-to-bet-on-sports/general/closing-line-value-definition-importance-sports-betting

[5] Petrella, S. (2020, June 30). *Point Spread in Sports Betting: Definition, Examples, How to Make a Spread Bet.* Action Network. https://www.actionnetwork.com/education/point-spread

[6] Basketball-Reference. (n.d.). *Glossary.* https://www.basketball-reference.com/about/glossary.html

[7] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An introduction to statistical learning with applications in R.* New York, New York: Springer.

[8] DraftKings. (2021, March 11). Retrieved March 11, 2021, from https://www.draftkings.com/

[9] Hao, K. (2020, April 02). What is machine learning? Retrieved February 14, 2021, from https://www.technologyreview.com/2018/11/17/103781/what-is-machine-learning-we-drew-you-another-flowchart/

[10] IBM Cloud Education. (2020, July 15). What is machine learning? Retrieved February 15, 2021, from https://www.ibm.com/cloud/learn/machine-learning

[11] Goodfellow, Ian, et al. Deep Learning. MIT Press, 2016, http://www.deeplearningbook.org.

[12] Gharat, Snehal. "What, Why and Which?? Activation Functions." Medium, Medium, 14 Apr. 2019, medium.com/@snaily16/what-why-and-which-activation-functions-b2bf748c0441.

[13] M., Conor. "Machine Learning Fundamentals (I): Cost Functions and Gradient Descent." Medium, Towards Data Science, 3 Apr. 2021, towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220.

[14] Brownlee, Jason. “Gentle Introduction to the Adam Optimization Algorithm for Deep Learning.” Machine Learning Mastery, 12 Jan. 2021, machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

[15] Beck, Howard. “The Truth About NBA Home-Court Advantage.” Bleacher Report, Bleacher Report, 18 Aug. 2020, bleacherreport.com/articles/2905080-the-truth-about-nba-home-court-advantage.

[16] “Sportsbook Reviews.” Historical NBA Scores and Odds Archives, www.sportsbookreviewsonline.com/scoresoddsarchives/nba/nbaoddsarchives.htm.

[17] Kwiatkowski, R. (2020, December 4). Performing Linear Regression Using the Normal Equation. Medium. https://towardsdatascience.com/performing-linear-regression-using-the-normal-equation-6372ed3c57.

[18] McKinney, W., others. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56).

[19] Pedregosa, F., Varoquaux, Gael, Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.

[20] RMS Error, statweb.stanford.edu/ susan/courses/s60/split/node60.html.

[21] Parmar, R. (2018, September 2). Common Loss functions in machine learning. Medium. https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23.

[22] Koech, K. E. (2021, February 25). Cross-Entropy Loss Function. Medium. https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e.

[23] Watkins, J. (n.d.). An Introduction to the Science of Statistics: From Theory to Implementation. https://www.math.arizona.edu/ jwatkins/statbook.pdf. University of Arizona.

[24] Lane, D. M., Scott, D., Hebl, M., Guerra, R., Osheron, D., amp; Zimmer, H. (n.d.). Introduction to Statistics. Rice University; University of Houston, Downtown Campus.