Machine Learning in Basketball Scoring Predictions

An Analysis of Closest Defender Based Predictions

Darrick Oliver

Computer Engineering Senior Project

California Polytechnic State University, San Luis Obispo

June 8, 2022

1 Introduction

1.1 Background

The use of machine learning within a sports context has historically been focused on predicting the outcomes of matches. Just within the sport of basketball, there have been a number of research papers solely dedicated to determining whether or not a trained model can more accurately predict the winner of a professional basketball (NBA) game over other proposed models, and even over people considered experts in the field.

One of these articles in particular, lead by researcher Samuel Li, explored the possibility of using both standard and more complex statistics to determine relative strength of a team. Using a combination of statistics tracked by team, namely field goal attempts, offensive rebounds, turnovers, and free throw attempts , a statistic to describe the "chance for a team to score" was created [2]. From there, Li was able to quantify the offensive and defensive qualities for each team, which would eventually be used to develop the data set for the match outcome prediction engine. The model that was created was found to be fairly accurate, and exhibited 63 to 67 percent accuracy for each season that was examined [2]. While the models were more accurate than a coin flip, they were unable to reach the near 70 percent accuracy rate that the best experts can predict at. One point of failure of their model was the inability to account for player injuries, suspensions, surprise upsets, and new players with no recorded statistics to base predictions on. One of the recommended strategies for future research mentioned in the conclusion was to use a "sliding window." This means to take into account only the most recent results, considering the average statistics within a specified range of time, or "window."

Another method of projecting the outcome of professional basketball games is to use a statistical model, as demonstrated by Kai Song and Jian Shi in their study titled "A Gamma Process Based in-Play Prediction Model for National Basketball Association Games." Using what they called an "in-play" model, they were able to create a model that would update its prediction during a live match. They discovered that points scored by the home and away teams generally followed a linear trend over the time played, while the covariance of the home and away teams' scores over time more closely matched a quadratic polynomial trend [6]. This led them to the creation of a model using a pair of gamma processes, one representing the home team and the other representing the away team. A gamma process is "used for paired comparison and ranking" and since "the NBA scoring process is nonnegative, non-decreasing and nearly continuous," they proposed the gamma process as an appropriate model [6]. Their models for the home and away teams both required the use of an "ability parameter," estimated from the data set as a measurement of relative ability of the team as a whole. Their proposed model outperformed their baseline model towards the beginning of the game, but would approach similar accuracy near the end [6]. While they were able to outperform the baseline model, they were not able to reach the accuracy of a different model called the "Improved Brownian Motion" model.

Yet another use case of machine learning in basketball is to project the future careers of players entering the league from college, also called "rookies." Zafar Mahmood,

Ali Daud, and Rabeeh Ayaz Abbasi addressed this issue by presenting a machine learning model that attempts to determine if a young player will turn into a "star" based on their collegiate basketball statistics. What makes this study so interesting is that they attempt to factor in "co-players," players on the same or opposite team as the rising star, who played in the same game. Out of all the statistics that were taken into account, one in particular stood out as a feature of a future star: the amount of free throws they attempted per game.

1.2 Problem Statement

The measurement of success of a player offensively is hotly contested as more advanced statistics are developed, however the number of points an individual scores per game has consistently been a standard metric of success. Most sports betting websites will have the option of betting on whether a player will score over or under a specified number of points, a number determined by their own secret model or experts.

The task of this project is to predict the number of points a player will score, while minimizing the loss measured between the actual and predicted values. Using a "sliding window" to account for recency bias, a number of selected statistics will be used as inputs to train a machine learning model on what statistics should stand out the most in a high-scoring versus low-scoring player.

1.3 Proposed Method

The proposed methodology of this project was to account for the closest defender when determining the number of points a player will score. Defense in basketball is stressed as an integral part of the game, with a "Defensive Player of the Year" prize awarded at the end of the season to the player that media members consider the most valuable player on the defensive end. However, few models have attempted to account for head-to-head data when considering the number of points a player will score.

1.4 Metrics

One of the standard metrics for determining loss is Root Mean Squared Error, or RMSE for short. Selecting RMSE as the loss function will provide "an idea of how much error the system typically makes in its predictions, with a higher weight for large errors" [1]. RMSE is calculated using the equation shown below, with n representing the sample size, $\hat{y}_i$ representing the predicted value, and y_i representing the actual value of a single sample.

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

Another metric commonly used when figuring out the loss between the predicted and actual values is the Mean Average Error (MAE). The use case of MAE as opposed to RMSE, as described by Aurélien Géron, is that "RMSE is generally the preferred performance measure for regression tasks, [but] in some contexts you may prefer to use

another function. For example, suppose that there are many outlier districts. In that case, you may consider using the Mean Absolute Error," [1]. MAE is calculated using the following equation, where n represents the sample size, $\hat{y}_i$ represents the predicted value, and y_i represents the actual value of a single sample.

$$MAE = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n} \tag{2}$$

For this project, RMSE will be the primary metric to determine loss. This is because RMSE more heavily punishes large errors, penalizing models with significantly worse predictions. This is important for our purposes, as substantial underestimates and overestimates in points would be disadvantageous in getting close as possible to the player's actual points.

2 Data Collection and Cleaning

2.1 Collection Methods

In order to predict points based on the nearest defender, head-to-head statistics for every player that played in the NBA from 2018 to 2022 were collected through the official NBA website. This data set would then (after cleaning) be used as the ground truths, or desired outputs, of the proposed model.

The inputs to the model were selected as per-game statistics from both the offensive and defensive player over the entire season. These were also collected from the official NBA website, and saved to separate dataframes based on the year they were collected.

2.2 Cleaning the Output

To estimate the player's points per game against a given defender, their total points scored against the defender were first divided by the number of possessions they played against each other. This result would be the points per possession the offensive player scored against the given defender, as shown in the equation below.

$$pts/poss_{h2h} = \frac{pts_{tot}}{poss_{tot}} \tag{3}$$

By factoring in the offensive player's average possessions per game, their average points per game against the same defender could then be estimated by multiplying the value found above by their average possessions.

$$pts_{avg\_h2h} \approx pts/poss_{h2h} * poss_{avg} \tag{4}$$

Another important element to consider was that a small sample size could skew the results of a generally "average" player. If a player recorded statistics for only 1 possession against a defender, and managed to score three points, their estimated points per game against that defender would be unreasonably large considering the sample size. To combat this issue, a threshold of 30 minimum possessions was implemented. This specific number was found through experimentation to maximize the number of data points remaining, while also reducing the overall loss of a trained model to a noticeable degree.

With the remaining data points, a histogram was generated to display the distribution of estimated points per game. The following chart (figure 2.1) shows the point distribution by season, as indicated by the color key.

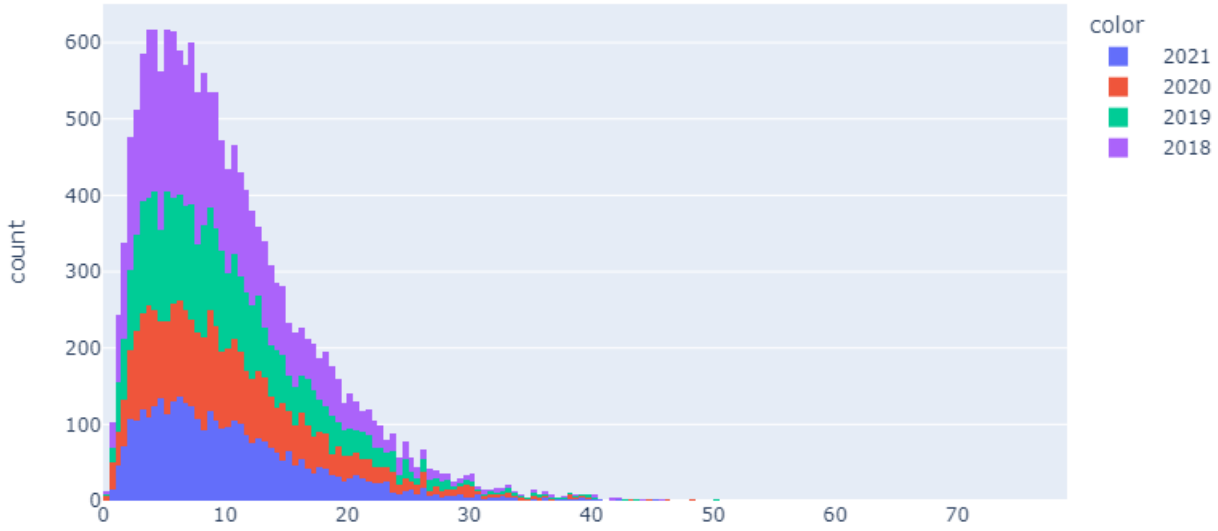

Figure 2.1: Estimated Points per Game by Season

2.3 Input Correlation Analysis

Cleaning the inputs to the model required removing any redundant parameters, ones that added no new information to the model. This was accomplished by examining the correlations between each pair of inputs in a matrix to ensure that the value wasn't over a specified threshold. The higher the correlation between inputs, the more often a large value in one parameter results in a large value in the other (and vice versa).

For example, it's expected that a high number of defensive rebounds results in higher total rebounds. Similarly, a higher field goal percentage (FG%) will typically result in higher effective field goal percentage (eFG%). This is because eFG% is calculated with a slight weight towards three point makes, but it makes a small difference from FG% for most players since the average NBA team will average around 12 three point makes per game between the every player on the team [4].

Shown in figures 2.2 and 2.3 are the input correlation matrices of selected offensive and defensive statistics respectively. Parameters with higher correlation have values closer to 1, while more unrelated parameters have correlations that approach 0. Neg-

ative values of correlation imply that the parameters have inverse correlation (higher values in one correlate to lower values in the other, and vice versa).

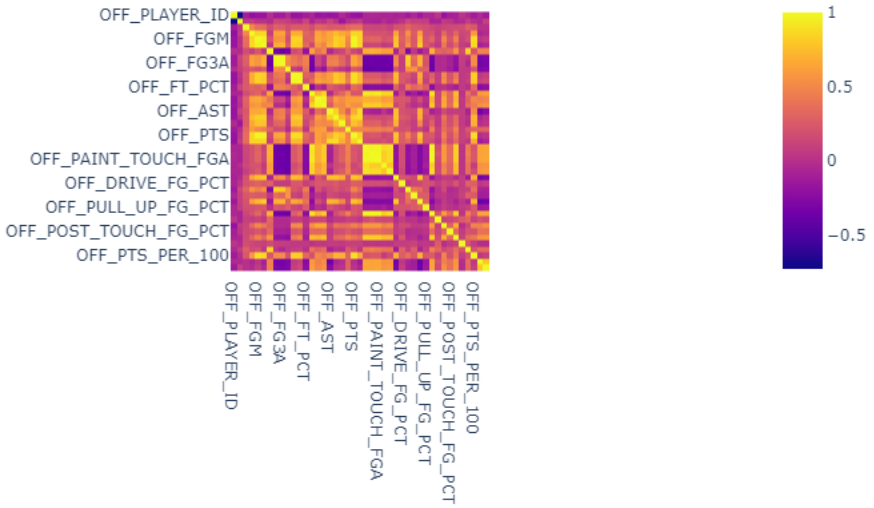

Figure 2.2: Input Correlation Matrix of Offensive Statistics Before Removal

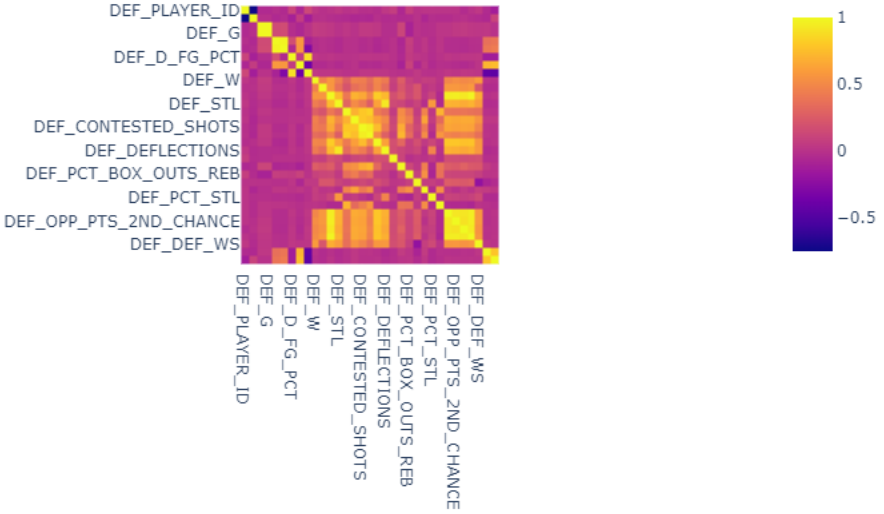

Figure 2.3: Input Correlation Matrix of Defensive Statistics Before Removal

When removing inputs from the dataframe, a threshold had to be decided to determine which inputs were too highly correlated. For this paper, the cutoff was selected as a correlation of over 0.9. Inputs past this threshold had to be manually reviewed to determine which to remove.

The offensive inputs that were removed due to correlations higher than 0.9 were field goal makes (FGM), effective field goal percent (eFG%), defensive rebounds (DREB), offensive rebounds (OREB), three point makes (FG3M), free throw makes (FTM), points per game (PTS), touches, and statistics taken from the paint such as field goal

attempts (PAINT_TOUCH_FGA) and field goal makes (PAINT_TOUCH_FGM). These statistics were mainly chosen for removal because they exhibited redundancy with inputs that already held percentages and attempts, points per 100 possessions, or other scoring statistics already present.

Defensive inputs that were selectively removed with high correlation were opponent field goal makes (D_FGM), contested two point shots, deflections, opposing player's points from fast breaks (OPP_PTS_FB), second chance points against, points off turnovers (PTS_OFF_TOV), games (G), and opposing player's points in the paint (OPP_PTS_PAINT). These statistics were chosen due to their redundancy with percentages and attempts, or simply were accounted for already in other categories like points against.

The updated matrices after removal of the aforementioned inputs are shown in the figures below.

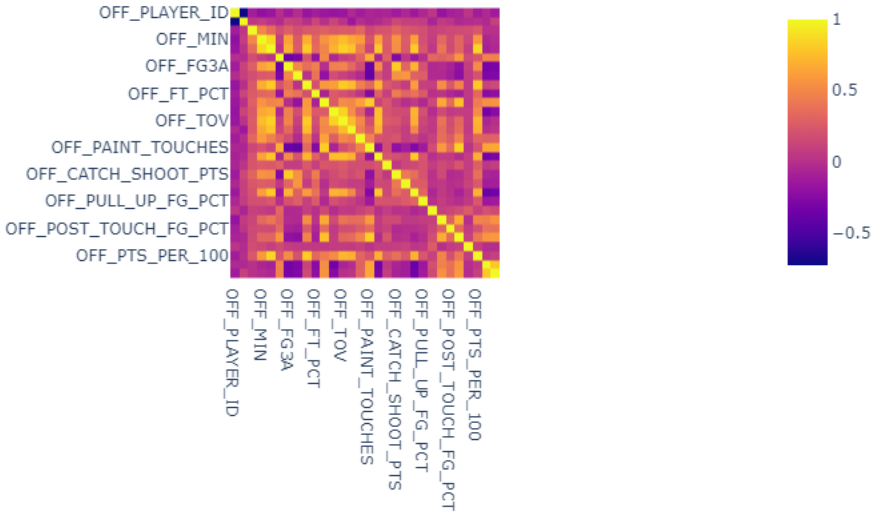

Figure 2.4: Input Correlation Matrix of Offensive Statistics After Removal

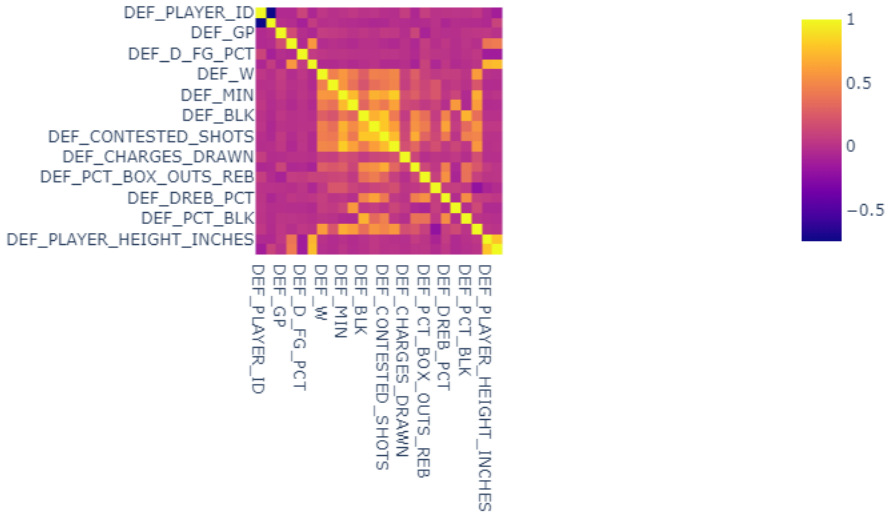

Figure 2.5: Input Correlation Matrix of Defensive Statistics After Removal

3 Models

3.1 Algorithms and Techniques

To have an adequate model to compare a proposed deep learning model to, I decided to use the Linear Regression model provided by the python library "scikit-learn." Linear Regression works by summing the input parameters with different weights, in addition to a constant called the "bias term" [1]. The equation for a Linear Regression prediction is:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \tag{5}$$

where n is the number of features, θ_0 is the bias term, θ_1 through θ_n are the feature weights, and x_1 through x_n are the feature values. Training the Linear Regression model on a data set will cause adjustments the feature weights to produce a linear prediction closer to the actual value.

The way that the Linear Regression model determines how to modify weights is based on the loss function used. Scikit-learn uses RMSE to determine weights, along with other additional mathematical tricks [5].

For the deep learning model, the keras Sequential model was selected to use in conjunction with a tensorflow backend. The python library "keras" makes use of a sequential neural network to create layers with one array of inputs and one array of outputs per layer. The size of these arrays can vary, and therefore the number of neurons between layers can vary as well. However, what makes it sequential is that it adheres to the rule that a single layer can only output to a single layer. An example of what such a neural network could look like is shown in the following figure.

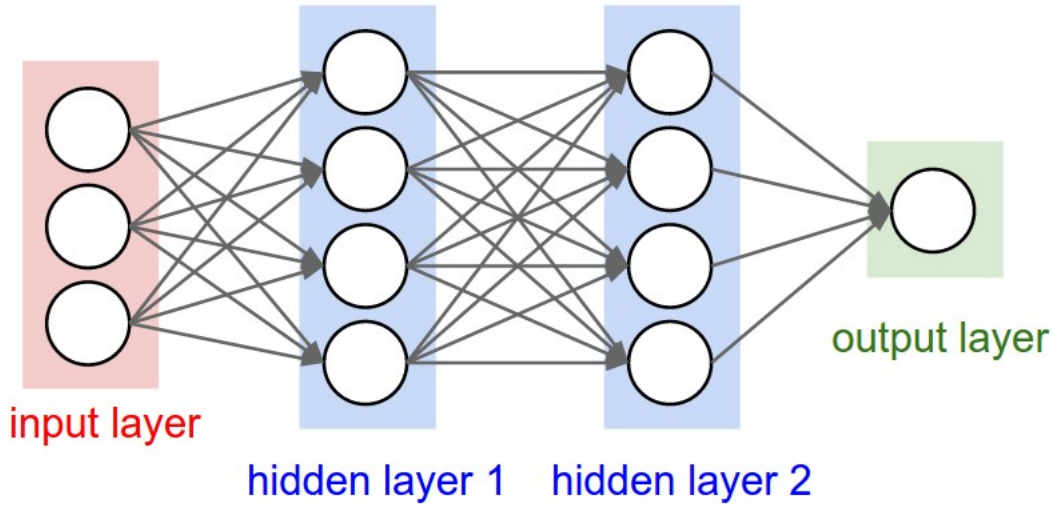

Figure 3.1: Sequential Neural Network (image credit)

In this figure, the circles represent neurons, or feature values. The arrows connecting between neurons each have linear weights applied to them, modifying the values between each layer.

3.2 Proposed Model Framework

The hidden layers in the Sequential model created for this project are shown in figure 3.2.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 100)               5100

 dropout (Dropout)           (None, 100)               0

 dense_1 (Dense)             (None, 64)                6464

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 11,629
Trainable params: 11,629
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.2: Sequential Neural Network Framework

The first layer shown is a dense layer of 100 neurons, followed by a dropout layer.

A dense layer is a layer in which each neuron connects to every neuron of the previous layer, similar to the hidden layers shown in figure 3.1. The way the inputs connect to this model is all 50 inputs are normalized and connected to all 100 neurons individually in the first dense layer.

A dropout layer will randomly set its neurons to zero to prevent a model from overfitting, which means to fit the model too closely to the training data. Overfitting has the potential to negatively impact performance, as random noise within the training data could get picked up as features of the entire data set. The rate at which the dropout layer drops neurons is determined by the percentage it's given as a parameter.

For this model, the first dropout layer was set to drop neuron values at a rate of 30%. This first dropout layer then leads to the second dense layer, which holds 64 neurons. The second dense layer's neurons are then fed into the second dropout layer, which drops neuron values at a rate of 60%. Finally, the output of this dropout layer is sent to a single neuron output, which is the final output, the predicted points the offensive player will score on the defensive player.

This model was chosen experimentally through many model iterations. The number of neurons per layer and dropout layer percentages were values found through automated parameter experimentation, where a python script tested a variety of different parameters in combination with each other. The final model that was selected provided the lowest RMSE given the data set.

4 Results

4.1 Model Comparisons

The table below contains the final models that were selected, with the loss values of both RMSE and MAE recorded.

Model	**Description**	**RMSE**	**MAE**
Baseline	Baseline result of average points per game	13.09	10.21
Average	Simple average of all players	11.59	9.10
Linear Regression	Regression trained on 10 primary inputs	11.56	9.02
Linear Regression	Regression trained on all 50 inputs	11.33	8.83
Sequential Neural Network	5 Layered NN trained on all 50 inputs	11.01	8.51

Table 4.1: Model Comparisons

The first model listed is the baseline, which took the offensive player's average points per game and assumed that for every defender, the offensive player will score their average points per game. This model proved to be the most inaccurate, as its RMSE was found to be fairly high.

The second model listed is a simple average, taking the average of all players and assuming that every individual player will score exactly the overall average against every defender. While there was some improvement over the baseline, there still was a lot of issues with this method. Although the loss is lower, it would never be a good idea to distribute this model as it only makes the "safe" prediction that no one will ever score more or less than the average.

The Linear Regression models that were tested contained both a model trained on only 10 primary inputs, and a model that trained on all 50 inputs. The 10 primary inputs that were selected were the defender's contested shots, defender's opposing field goal percentage (DFG%), defender's block percentage (DBLK%), defender's steal percentage (DSTL%), offensive player's assists (OAST), offensive player's blocks against (OBLKA), offensive player's field goal percentage (OFG%), offensive player's three point percentage (O3P%), offensive player's free throw percentage (OFT%), and the offensive player's turnovers (OTOV).

The purpose of training with only 10 inputs as opposed to the entire 50 was to show the impact that more inputs can have on the model's loss. This will be discussed in more detail in section 5.2, future recommendations.

And finally, the Sequential model had the lowest RMSE compared to all other models. This proved to a degree that the inputs are too complex to be analyzed linearly, and potentially could be used to justify the use of a Neural Network over a Linear Regression. This will be discussed further in section 5.1.

4.2 Feature Value

To figure out which features had the most impact on the final output in the sequential neural network, the python library shap was utilized. This library was used to train an explainer, which would take the fully trained model and the training data set in order to determine which inputs had the highest influence on the output when predicting. The graph produced by shap is shown in the figure below.

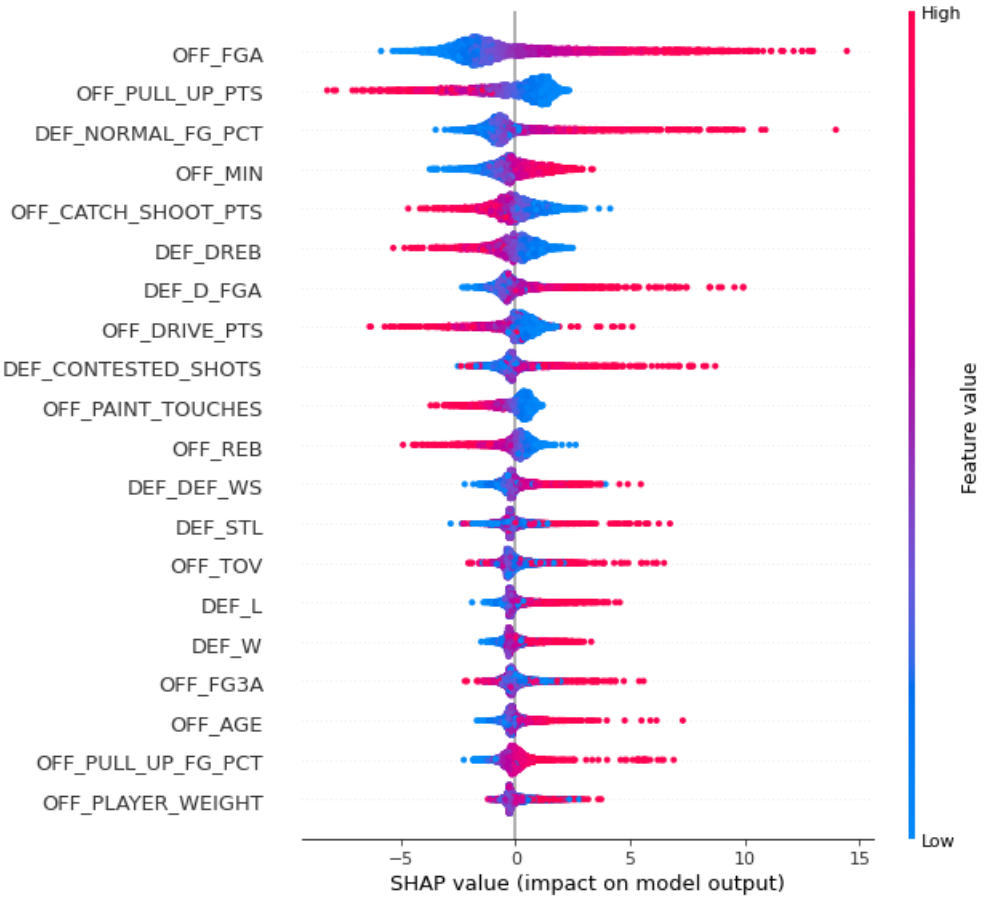

Figure 4.1: Feature Value

To interpret the graph's results, it's key to first understand that the vertical axis represents the inputs that the model found were the most important when determining a final output. The higher the feature on the vertical axis, the more important it was. The second key point to understand is that the horizontal axis represents a quantitative indication of positive or negative impact on the model's output, while the color of each dot represents the value of the feature itself. For example, a more red point to the right of the vertical axis indicates that the specific input had both a high value, and a positive impact on what the final model outputted. However, as seen with the pull-up points (OFF_PULL_UP_PTS) and catch and shoot points (OFF_CATCH_SHOOT_PTS), higher input values can also lead to more negative point predictions depending on how the neural network valued them.

4.3 Predictions

In addition to training the neural network model, predictions were made in the future to see if the model had any value over Linear Regression. To do so, statistics from the last 10 games were pulled from the NBA's website and used as inputs to the model. Four key players from each playoff game were selected, and were tested on both the Linear Regression and Sequential model.

For each offensive player selected, the percentage of playing time against a specific defender is calculated by dividing the total minutes against that defender with the sum

of the minutes played against all defenders of the same team. This is assumed to be the percentage of the time that the offensive player will be against that defender in the upcoming game. From there, predictions for each defender are made, weighted by the estimated percentage calculated before, and summed to produce a final result. This result is the estimation of points the offensive player will score overall in the match.

The first playoff game that was selected was the Golden State Warriors against the Memphis Grizzlies on May 11th, 2022. The four key players selected were Steph Curry and Klay Thompson of the Golden State Warriors, and Jaren Jackson Jr and Kyle Anderson of the Memphis Grizzlies. The predicted scores of each player are shown in the following figure.

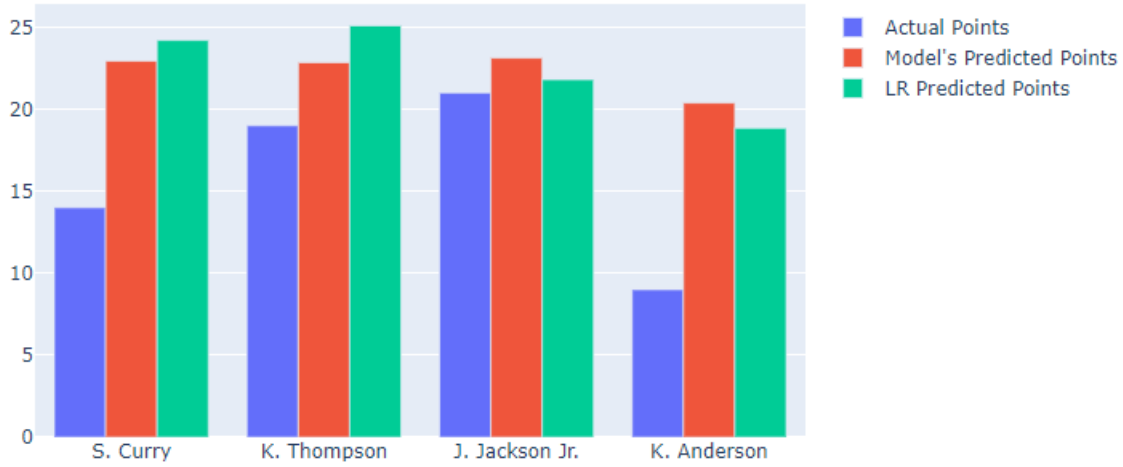

Figure 4.2: Warriors Versus Grizzlies Predictions, May 11 2022

The second game that was chosen was the Miami Heat against the Philadelphia 76ers, which took place on May 12th, 2022. The four key players selected were Jimmy Butler and Tyler Herro of the Miami Heat, and Joel Embiid and Tobias Harris of the Philadelphia 76ers. Scores predicted by each model are shown below.

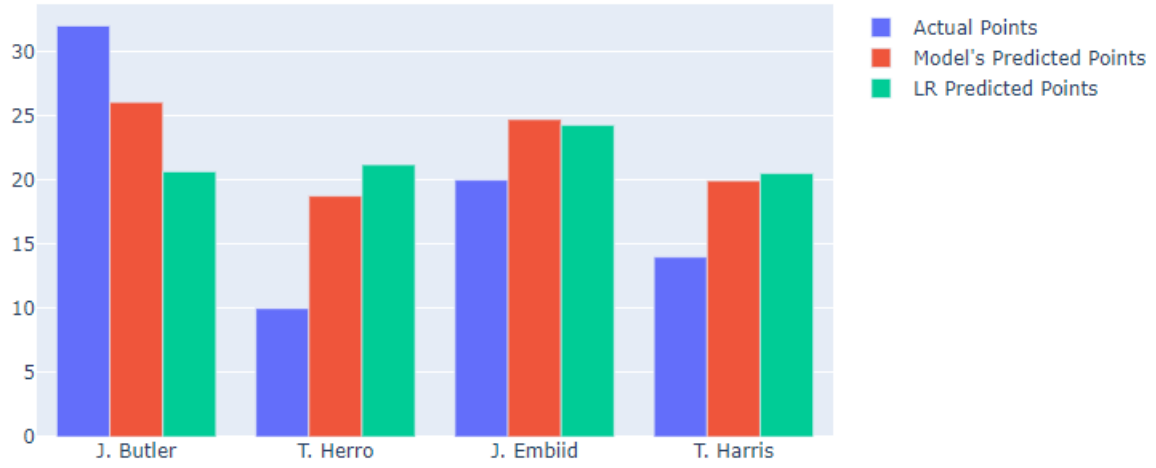

Figure 4.2: Heat Versus 76ers, May 12 2022

And finally, the last playoff game selected was the Phoenix Suns against the Dallas Mavericks, which also took place on May 12th, 2022. The four players chosen were Devin Booker and Chris Paul of the Phoenix Suns, and Luka Doncic and Jalen Brunson of the Dallas Mavericks. Predictions by each model are shown in the following figure.

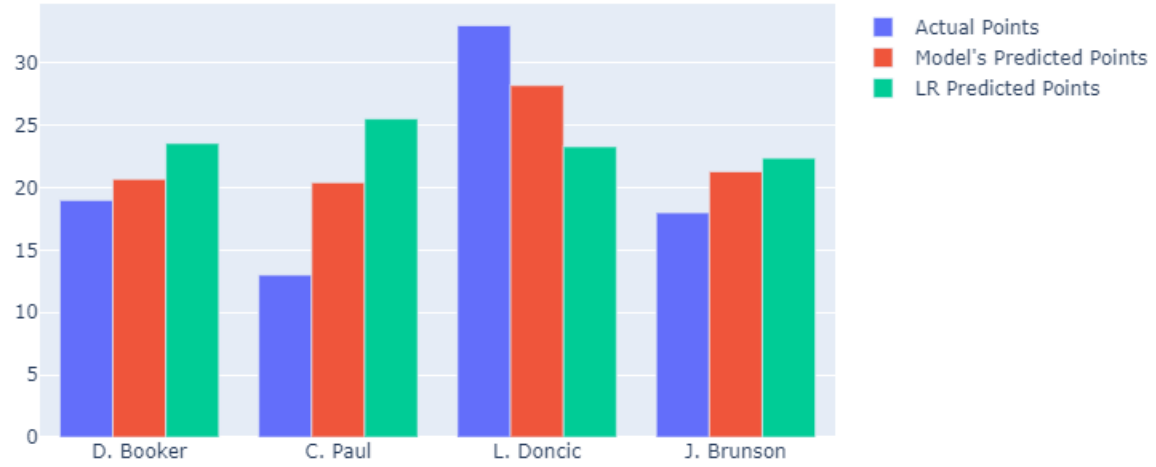

Figure 4.2: Suns Versus Mavericks, May 12 2022

5 Conclusions

5.1 Interpreting Results

Comparing the different models tested showed that the sequential neural network outperformed the baseline model by 15.9%, the average model by 5.0%, and the Linear Regression model trained on all 50 inputs by 2.8% based on RMSE. The neural network showed huge improvement over each of these models, especially over the baseline and average models. This improvement was made even more evident in generating new predictions, where it was able to outperform the Linear Regression model in a majority of

the cases. While no model will be perfect, as there will always be games where players underperform or overperform, the main takeaway from the results of testing was that using a Neural Network for predictions provides a clear advantage over using Linear Regression due to the complicated nature of sports and statistics.

In terms of the feature values generated by shap (seen in figure 4.1), the feature that stood out most to me was normal field goal percent (DEF_NORMAL_FG_PCT). This statistic is a way of telling what an offensive player will normally shoot against everyone EXCEPT the given defender. This input had an increasingly more positive impact on the output when the value was higher, leading me to believe it was perhaps being used to estimate the defender's quality of opposition. If the defender plays more against high quality players, their normal field goal percentage will be much higher than, say, a rotational defender that plays against lower quality opposition. This could then lead to the assumption that if the offensive player is playing against a high quality defender, the offensive player is also most likely considered high quality. It's important to note, however, that this could be an erroneous way of thinking. This is primarily because if a prediction of a bench player against a starting defender is made, it could lead the model to assuming the bench player has more impact on the game offensively than they actually do.

Many of the predictions made in section 4.3 showed that both models had some issues predicting players' points during a lower scoring game. I believe this is likely due to the low sample size of playoff games that were selected for the inputs, and could potentially be changed by extending the sliding window size. In the case of the playoffs, the sliding window has a shorter cap that cannot be passed, as the playoffs contain far fewer games than the regular season. Another reason that this could be happening is the quality of outputs being used. Because the output is merely an estimation of points scored per game based on a minimum of 30 possessions, it likely won't be resulting in an accurate representation of the player's ability. 30 possessions typically happens in under a single half of basketball. However, there simply are not enough recorded possessions in a single game between two players to justify increasing that threshold. Potential improvements to this are mentioned in the following section.

5.2 Future Recommendations

As evident by the training of both Linear Regression models on 10 and 50 inputs, the more inputs that are taken into account, the more accurate a prediction can be, and by definition the lower the loss will be. In future research, it may be ideal to investigate even more statistics to be used as inputs to the neural network. Because life itself is so complex, it's difficult to predict the final points a player will have in a game with only a select few statistics. There is a limit to how much this will improve the RMSE, however, as there are only so many statistics that the NBA keeps track of.

Another improvement that could be made is to have the model determine a player's points per possession rather than estimate their points per game. With the points per possession being generated by the model, the average number of possessions a player has had over the range of the sliding window can then be used to have a more accurate estimation of their points in a given game. This would eliminate the use of an overall average possessions per game, and would be better suited for the sliding window strat-

egy.

The final suggested improvement for future continuation of this study would be to remove certain features that may confuse the model. The one mentioned in the above section, DEF_NORMAL_FG_PCT, may be misleading the model in its assumptions. While this will not show up in the overall loss, it will show up when attempting to generate new predictions, especially in the example provided previously. Further research into other parameters that behave similarly would also be highly advised, to ensure they are not negatively effecting future predictions.

6 Bibliography

[1] Géron, Aurélien, and Rebecca Demarest. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly, 2019.

[2] Li, Samuel. "Revisiting the Correlation of Basketball Stats and Match Outcome Prediction." *Proceedings of the 2020 12th International Conference on Machine Learning and Computing*, 2020, https://doi.org/10.1145/3383972.3383980.

[3] Mahmood, Zafar, et al. "Using Machine Learning Techniques for Rising Star Prediction in Basketball." *Knowledge-Based Systems*, vol. 211, 2021, p. 106506., https://doi.org/10.1016/j.knosys.2020.106506.

[4] *NBA league averages - per game.* Basketball. (n.d.). Retrieved June 7, 2022, from https://www.basketball-reference.com/leagues/NBA_stats_per_game.html

[5] Prasad, D. (2020, September 14). *Sklearn: Linear Regression Under the Hood.* Medium. Retrieved June 8, 2022, from https://medium.com/analytics-vidhya/sklearn-linear-regression-under-the-hood-31ee71aec00

[6] Song, Kai, and Jian Shi. "A Gamma Process Based in-Play Prediction Model for National Basketball Association Games." *European Journal of Operational Research*, vol. 283, no. 2, 2020, pp. 706–713., https://doi.org/10.1016/j.ejor.2019.11.012.