# NBA Player Performance Prediction Based on XGBoost and Synergies

**Haochen Hu**
Department of Statistics
University of Washington
hhu05@uw.edu

**Gantcho Dimitrov**
Paul G. Allen School of Computer Science
University of Washington
gantcho@uw.edu

**David Menn**
Department of Aeronautics and Astronautics
University of Washington
dmenn@uw.edu

**Songhao Wu**
Department of Statistics
University of Washington
s246wu@uw.edu

## Abstract

Modern professional sports leagues keep track of a wide variety of detailed data regarding player performances to better gauge the development of players throughout the season. In this project we propose a decision tree-based model to predict a player's performance in a specific game based on information known from his earlier season performance. First, we carry out an extensive data analysis and design a clustering algorithm for a more comprehensive player feature selection process. Then, we consider how playing a game with certain clusters affects each players performance by calculating a synergy for each player with each cluster. Finally, we propose a gradient boosted decision tree model implemented through XBGoost equipped with the aforementioned synergies that outperforms all other baseline models and gives us more interpretable results, leaving room for improvements and interpretations for different purposes.

## 1 Introduction

The National Basketball Association (NBA) is the professional basketball league in America and is composed of 30 teams from across the country. Generally considered the most prestigious and skilled league in the world, the NBA has long attracted the attention of global viewers who support their team in their goal to a championship. With viewership growing by 1% from last year to an average of 1.6 million viewers per game, the NBA currently values its media rights at $25 billion and expects it to rise to $75 billion in the near future (6). As the league and teams themselves have grown more valuable, team owners and coaches have turned to advanced metrics to keep track of player performances in order to better evaluate the worth of various players across the league and their team's chances of success. With 30 teams of around 15 players each, playing in 82 games throughout the season, there is a plethora of data available regarding individual performances throughout the league.

The goal of this project is to utilize this existing database in order to generate a model that can predict a players performance in a specific game using only historic data. Specifically, we will present a model that accepts the average season performances of the players in a game as well as an estimate of the number of minutes that they will play and outputs a prediction for each players' game score.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Here a player's game score is defined as:

$$PTS + 0.4FG - 0.7FGA - 0.4(FTA - FT) + 0.7ORB + 0.3DRB + STL + 0.7AST + 0.7BLK - 0.4PF - TOV \quad (1)$$

In the equation above, PTS is the number of points scored by a player in the game, FG is the number of field goals scored, FGA is the number of field goals attempts, ORB is the number of offensive rebounds, DRB is the number of defensive rebounds, STL is the number of steals, AST is the number of assists, BLK is the number of blocks, PF is the number of personal fouls, and TOV is the number of turnovers. We chose to utilize game score as our desired predictive metric because it is a commonly used advanced statistic to track the holistic performance of a player within a specific game. Furthermore, this allows us to clearly define our evaluation metric which will be the root mean squared error between our predicted game score and the true values. In order to consider our model a success, we will evaluate the performance of our predictions against what we call the baseline prediction which outputs a game score for a player by assuming that they perform according to their season averages in every statistic. In this way, we can see that significant improvement on this estimate, will show that our model can predict a players performance deviation from their typical performance based on who he is playing with and who he is matched up against. Ultimately, a driving motivation behind this model is to be able to answer questions relating to how a new player might fit into a team, how well a team matches up against another, or how redistributing the minute shares in a team might improve that teams performance.

In the sections to follow we present the data set that we will be utilizing as well as the unique preprocessing required to extract the information needed to feed into the model. We further present interesting results from an initial data exploration surrounding clustering of player types and how we utilize this information to generate player features that incorporate their unique play style. In fact, this idea of generating a player profile based on their average impact becomes a driving factor in uncovering what we refer to as synergy in the sections to follow. Finally, we present experimental results from our model to show its performance and success. Specifically, with the correct feature selection, our novel model, which is based upon the gradient boosted decision tree framework implemented by XGBoost, can be seen to drastically improve upon the baseline prediction introduced above as well as the predictions produced from more standard multi-output regression models such as multi-layered perceptrons. This ultimately allows us to conclude that our model succeeds in uncovering a finer relationship between a players performance in an individual game and the other players on the court than simply predicting their season average performance.

## 2 Related Work

Though we could not find any research projects that attempted to predict individual player performances in the context of basketball, we did rely on a set of existing literature on related topics. The existing literature that we utilized namely helped us to identify an effective model type for this task and introduced us to the notion of player synergy as a method to augment our player features at prediction time. Our initial investigation surrounding predictions of basketball performances led us to a project that claimed to predict the results of the extremely popular yearly collegiate basketball tournament known as March Madness with 90% test accuracy (5). Though this project focused on predictions at a team level, it did give us a sense of how to create features to represent a singular game as a vector. Furthermore, it introduced us to the xgboost framework as a popular implementation to the gradient boosted decision tree model and its effectiveness on tabular data.

### 2.1 Baseball Synergies

Ultimately, a major contribution towards the findings of this paper are in the discovery of the notion of player synergies and utilizing relationships between players to uncover deviations in player performances from their averages. The paper "Uncovering the sources of team synergy: Player complementarities in the production of wins" by Brave et al.(1) has given a lot of credibility to the idea of finding player synergies and using that to predict performance. The goal of this paper was to out-predict Wins Above Replacement (WAR), which is a baseball statistic that attempts to predict

how many wins a player contributes to a team over a season compared to a replacement level player, or average player. Just using the WARs of players on the team to construct a regression model to try to predict season standings led to a generally inaccurate model with large residual error. The paper argues that this is due to WAR being calculated without any context as to who is on the team and how they play with each other. They use the fact that positions in baseball are very regimented, and so each player has a defined position on defense and a place in the batting order for each game that can be used to classify a player's type. As a result they argue that a player's performance is heavily linked to the other players on the field and attempt to quantify this effect. This takes the form of a constructed synergy matrix for each player, which attempts to predict the residuals of the prediction on season standings just using WAR. Though the authors of this paper generate positional relationship coefficients that come from classical baseball statistics, we modify this approach to have our model learn these coefficients themselves. Ultimately, we leverage the ideas that lead to significant improvements in the performance of these authors' model to see a similar improvement on our player performance model in a basketball setting.

## 3 Dataset and Processing

### 3.1 Data

At the heart of this project is the selected data set. The dataset we used was found on Kaggle and can be found at `https://www.kaggle.com/datasets/nathanlauga/nba-games`. This dataset includes every game played in the NBA from October 2014 to December 2022. It has data on the outcome of each game and team-wide statistics, as well as statistics for each player during each game. In fact, for each game played, the data set has 20 game-specific features for each player including things like points scored, rebounds, assists, etc. We decided upon this data set because of the breadth of data it collected including statistics for 26,595 games, with 668,269 player performances recorded in those games for 7,229 distinct players.

### 3.2 Data Pre-processing

Our goal is to utilize a gradient boosted decision tree method such as the XGBoost framework which has shown promising results on tabular data similar to the format of our data. As a result, we need to pre-process our game data into a format that is recognizable for our model. To start, our raw data comes in the form of a per-player performance per-game. This means that we have data from the last decade on every single players performance in each of the games that they have played in. As a result, we require some pre-processing to get this into the form that we want. Firstly, we utilize spark to group the data by player and by season and then average the results to get a vector resulting in their average statistics per season. This means that a player like Lebron James that has played in many seasons will have a row in our table for their per season average statistics for each of the seasons that they have played.

For each data point we input to our model, we require all the data on for a game. This means that we have to utilize spark to group our data on a per-game then per-team level allowing us to get a sense of all of the players that played in every game and differentiate them by team. From this we utilize the per season average statistics of our players to develop a game vector which consists of a concatenated vector of all of the average statistics of players in the game. This is then used as a training data point in a regression setting against a vector of the players' PER which is passed into our model. One issue we faced in the data pre-processing step was that the number of players that play throughout the course of a game is not a constant. This means that certain games will have more players check in than others which complicates the requirement for a fixed input dimension to our model. To fix this issue, we noted that in every game played, both teams utilized at least 6 players for a significant number of minutes. Thus, to ensure that our per-game data is in a consistent format, we truncate our game vector to only include the 6 players per team that played the most number of minutes that game.

Furthermore, though it may be noted that utilizing average season statistics to predict game performances of that same season is not practical because we would not have season averages in a test setting, initial data exploration revealed that average statistics do not change significantly following the first quarter of the season. Thus, we venture that our model should generalize well to games that occur in the latter half of the season. Though this pre-processing is sufficient for our initial version of the model we later incorporate player clusterings into our game features as well as the team synergies

which will be explained further in the coming sections to allow for more rich features and a more generalizable model. These additional steps required further data processing in order to generate training data from our raw data.

### 3.3 Clustering

Clustering is an important data processing step to allow us to calculate synergies in a future step (2). By assigning a cluster to each player, we group players into types. Then, when trying to develop a profile for a particular player A, we can look at that players history playing with and against other players based on those players' clusters. This way, when player A is in a game with a new teammate or an opponent they haven't played before our model has some idea of that player A's history against players of similar type.

To perform the clustering using k-means, we first standardized a players average statistics for a season using Z-scores based on means and deviations for each statistic for that season. We then used PCA to remove global correlations and retained 8 out of 16 stats with 88% variance explained. By utilizing PCA, not only did it greatly enhance the training speed, but also yielded a large improvement with regard to Silhouette scores as shown in figure 1. We find the optimal cluster by plotting figure 2. After k = 10, the sum of squared errors decreased in a linear fashion, which means 10 clusters is the optimal choice for clustering by the idea of elbow curve.

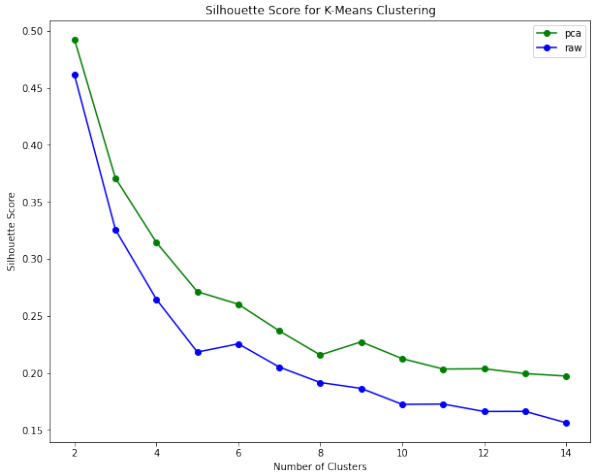


Figure 1: Silhouette Scores

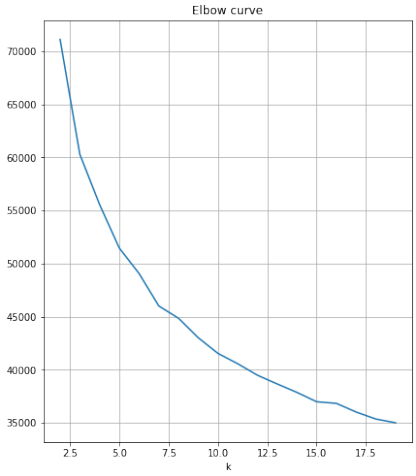


Figure 2: Elbow Curve

The clusters give us good interpretation for the data, where each cluster represents a type of player, e.g. cluster 2 represents pure shooter, cluster 4 represents low-volume shooters, cluster 9 represents small guards that consider passing first, etc. The interpretability of these clusters will hopefully translate to maintained interpretability through the calculation of synergies and the final decision tree model.

## 4 Model

### 4.1 Feature Selection

We used two main player profiles in our experiments. First, just using a players average statistics for a season was sufficient in providing enough information for a model to improve significantly over baseline predictions on game score. In this case, the training data given to the model was of size $x = 12 \times 18$. Each row was a player, grouped by team, and each column was a different statistical average that player posted that season.

The second model included all the data that the first model had, but also appended a vector of synergies to the profile. We calculate synergies for player A by trying to get the ideal fit of $\hat{\epsilon}^A = S^A x$, where $\tilde{\epsilon}^A = b^{A,avg} - b^{A,\text{in game}}$ would be one observation of the residual in box scores for a particular

game for player A, with respect to their average box scores. $b_i$ is a statistic of interest, like points or assists, on a per-minute basis. $b_i^{A,avg}$ is the average stats player A has put up in some past interval of games played before the game we are observing. This interval should be tuned to get the best results. For now, to ease computation, we will use the players average for an entire season (including games that are in the future of the game we are observing). This is of course not possible when trying to predict a future game, but since we will be fitting this data to past seasons we can use the players entire season average.

$S_{ij}^A$ is the impact that 1 minute of gametime from a player in cluster $j$ has on residual for that game of stat $b_i$. Note that $x \in \mathbb{R}^{2c}$, when $c$ is the number of clusters. This is because there are synergies for each cluster that plays on that players team, and synergies for each cluster that plays on the opposing team.

Now we have observation of $\tilde{\epsilon}^A$ and corresponding $\tilde{x}$ for many games. We just need to fit a matrix to this data. There are a few methods we might choose to make this matrix. We use Ordinary Least Squares linear regression. This will minimize the L2 norm loss on our data. The final step is to get the $S^A$ matrix into a vector of the synergies with respect to game score, and not each individual stat. To do this, we simply apply the formula for game scores in equation 1 to the synergies for the statistics in order to approximate the synergy for game score.

The motivation behind synergies is that an xgboost model that does not see a players synergies has no way of knowing anything about that particular player's playstyles. It can learn about trends for statistics of similar players, but it can never learn the actual play style of one particular player. The synergies allow us to give some information about one particular player's interactions with others in their history, thus giving the model some information about a particular player's identity beyond just their average stats.

### 4.2 Naive Model

Our naive model is one which ignores the other players on the court and instead utilizes only the specific player's season average statistics in order to calculate a prediction for their game score. Essentially, when predicting the performance of player A in any specific game, the naive model plugs player A's average performance into equation 1 to get an output for their predicted game score. This implies that the naive model will output the same prediction for every single game for an individual player. While this is obviously not a realistic depiction of a player's performance throughout an entire season, it does provide a valuable baseline in comparing the output of novel models. This serves as a best guess of a player's performance in a game and so any improvement upon this metric can be viewed as a success. After training on our data set and evaluating on a hold out test set of games, we see that this naive prediction results in a RMSE of 7.333. Thus, going forward we aim to develop a model which can achieve a RMSE lower than this.

### 4.3 MLP

We next attempted to uncover a more nuanced relationship between the players on the court and their individual performances utilizing supervised learned. To accomplish this, we decided to gain an understanding of how much of an improvement we can see over the baseline utilizing a simple multi-layered perceptron. Architecturally, our model includes an input layer that accepts data of dimension 216. This corresponds to the 18 season average statistics for the 6 players with the most minutes played in a game from both teams. Note we did not use the synergies to augment the MLP. From here we pass through a number of fully connected layers before a final output layer reduces our data to dimension 12, corresponding to the predicted game score for each player. Between each fully connected layer we introduce dropout at training time as well as ReLU non-linearity. While the purpose of this project is not to train as effective of an MLP as possible, since we are utilizing this as a comparison for simpler supervised learning techniques we still perform hyper-parameter tuning to uncover a reasonably effective model rather than blindly deciding on parameters for network depth and width.

Though not exhaustive, we perform a simple grid search over a set of network depths from 3 hidden layers up until 11, and width of hidden layers from 50 units until 300 units. The result from this allowed us to arrive at a final architecture which included 9 hidden layers with a width of 100 units.

We train for 50 epochs with a learning rate of 0.001 to arrive at our final validation RMSE of 6.453. While this does outperform our baseline prediction RMSE, it is not a very sizeable improvement, leaving room for more advanced techniques to outperform. Though there might be concerns over the amount of training time, figure 3 reveals that loss convergence over our training set is quite quick. The mean squared error seems to stabilize at just above 40 within 300 batch iterations which is just under 2 full epochs of training. This seems to suggest that the 50 epochs of training is more than enough as we do not see further improvement over the loss following the initial few epochs. This is likely due to a limit to the complexity of our model showing that a standard multi-layered perceptron is not powerful enough to uncover a much finer relationship than the baseline prediction. This however, is a promising step in the correct direction as we have shown that improvement over the naive prediction is possible and thus, a more powerful model might be better suited for this task.

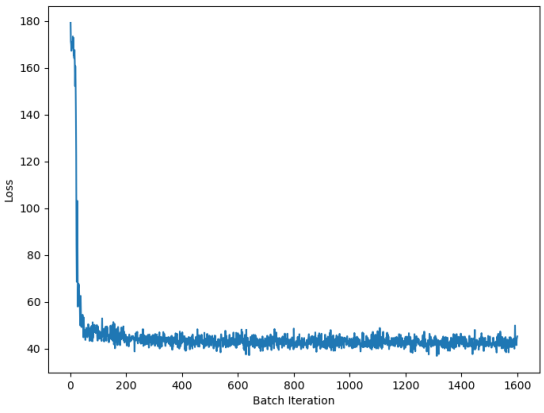


Figure 3: Training loss plot for our finalized MLP over the course of the first 50 epochs of training

### 4.4 XGBoost

XGBoost is a gradient boosting method that works by creating new models that predict the residuals of the prior models then compound them to make a prediction (3) (4). So far, XGBoost seems extremely effective in tasks similar to ours (5), so we will focus primarily on that model. The input will be a stacked vector of each player's player profiles, grouped by team. The output will be the first player's game score. We expand the data by having each game appear twelve times, with a different player as the first player in the vector. This way each game gives us twelve different game scores to predict, allowing us to make full use of our data.

### 4.5 Training XGBoost

XGBoost was succesful at improving over baseline for the prediction of game score. While the naive prediction for game score would give you a root mean squared error of 7.33, our best model could predict game score with a root mean squared error of 3.127.

The first comparison of interest is the different player profiles that were proposed. The two main ones that proved viable were

1. Player profile of just the average per-minute statistics a player posted for the season and that player's cluster

2. Player profile of the average per-minute statistics, along with that player's cluster and their synergy vector

Figure 4 shows a comparison of how the two profiles compared when trained on a standard XGBoost model with no parallel trees, a max depth of 6, and a learning rate of .3. This model configuration

was used to test a variety of different profile and label configurations, as it was quick to train and showed strong results. We do see a marginal benefit with the addition of synergies, with a consistent improvement in predictions of 7.2% for this simpler model.

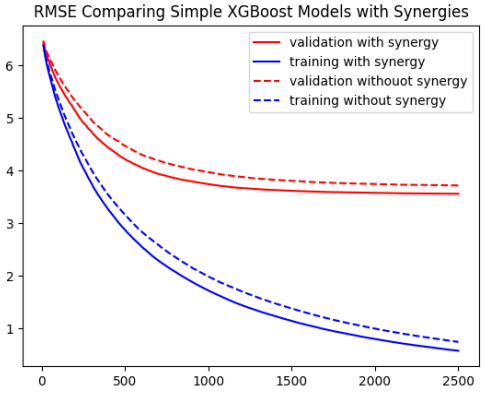


Figure 4: Final Validation RMSE without synergies was 3.716, Final Validation RMSE with synergies was 3.557

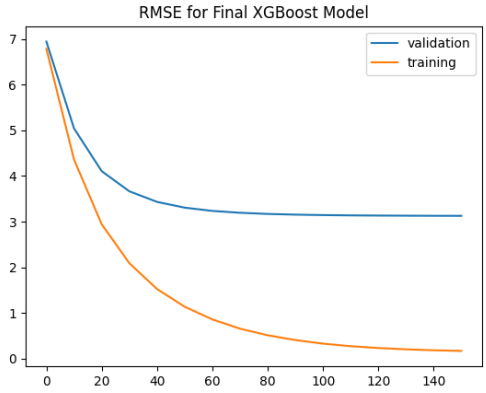


Figure 5: Final Validation RMSE of full XGBoost model was 3.127

The final model we selected had a max depth of 10 and 50 parallel trees, with an 80% sub-sample per tree. A hyper-parameter we spent time tuning was the trade-off between increasing the maximum depth and the number of parallel trees. We found that after around a depth of 10, the benefits from increasing the depth were minimal and did not help combat over fitting. However, adding more parallel trees did help decrease the validation loss by preventing over fitting, allowing us to increase model complexity without having an increased risk of over fitting. Figure 5 shows the loss curves from training of this model. Compared to the simple models, which took on the order of minutes to train, this model took in excess of an hour.

## 5 Evaluation and Interpretation of Results

### 5.1 Interpreting a Single Game

Below is a table of predictions from a single game (id :52100211):

| Team | Player | Game Score | Prediction | Baseline |
|---|---|---|---|---|
| New Orleans Pelicans | Brandon Ingram | 23.3 | 21.52 | 20.60 |
| New Orleans Pelicans | Jonas Valanciunas | 4.6 | 11.15 | 12.43 |
| New Orleans Pelicans | Herbert Jones | 4.5 | 8.76 | 9.26 |
| New Orleans Pelicans | CJ McCollum | 5.2 | 17.90 | 18.83 |
| New Orleans Pelicans | Larry Nance Jr. | 20.5 | 9.07 | 8.05 |
| New Orleans Pelicans | Trey Murphy III | 7.1 | 7.28 | 9.39 |
| LA Clippers | Nicolas Batum | 10.9 | 10.41 | 10.01 |
| LA Clippers | Marcus Morris Sr. | 21.8 | 15.96 | 15.11 |
| LA Clippers | Terance Mann | 7.3 | 12.13 | 11.72 |
| LA Clippers | Reggie Jackson | 22 | 19.15 | 15.08 |
| LA Clippers | Norman Powell | 14 | 7.29 | 7.31 |
| LA Clippers | Robert Covington | 1.6 | 8.22 | 9.91 |

Note that our prediction usually get closer to the true game score than the baseline method. However, in general, the model tends to stay close to the baseline game score. The tendency is expected as the model is trained to obtain good mean error, so it will avoid predict extreme values while a fair NBA game is of high variance due to unforeseen affects, even luck. Our model still gives us valuable

results as it predicts the tendency of if a player will be performing above or below its average level solely based on its previous game data.

### 5.2 Visualizing the Importance of Features

We train a simple Synergy-XGBoost model with depth of five to see the effects and importance of different features.

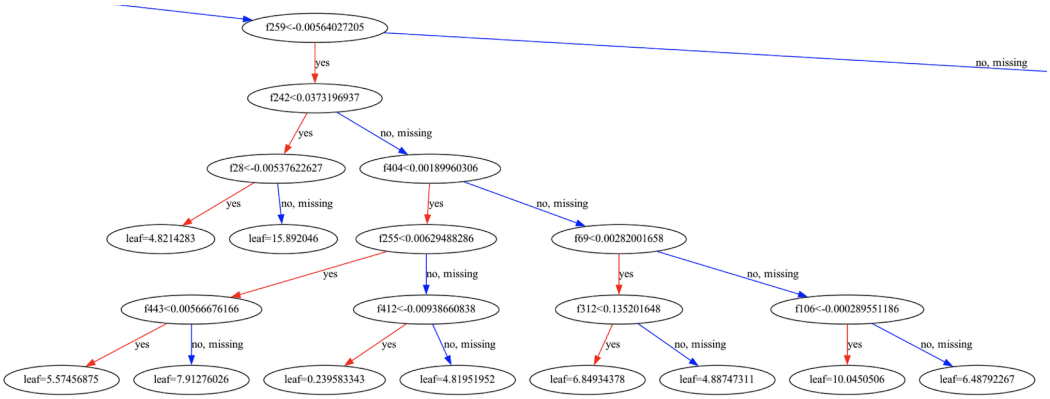


Figure 6: Simple Subtree

Define

$$
\begin{aligned}
d &:= \text{feature in tree node} \mod \text{len(player\_profile)} \\
\mathcal{F} &:= \{\text{All features considered in a subtree}\} \\
\mathcal{D}_+ &:= \{f \in \mathcal{F} : f \text{ is a synergy feature}\}.
\end{aligned}
$$

Recall that the player profile data that we feed into the model is each player's profile data concatenated, namely each 38 features belong to a player, and for the 38 features, top 18 are game score data and the last 20 are synergies playing with or against players from different clusters. Look at the subtree shown below, there are 10 tree splitting point,

$$
\begin{aligned}
\mathcal{F} &= \{259, 242, 28, 404, 255, 69, 443, 412, 312, 106\} \\
D_+ &= \{f \in \mathcal{F} : d_{38}(f) = f \mod 38 > 18\} = \{259, 28, 404, 255, 69, 443, 312, 106\}
\end{aligned}
$$

Note that $|D_+| = 8$ out of 10 splitting points come from synergy features in the example subtree. And by observation, most trees have over $80\%$ of the splitting points coming from synergies. We have shown empirically that the synergies features are important and preferred in a decision tree-based prediction model.

## 6 Future Work and Challenges

In Section 4.1, we discuss our method for determining synergies. We used the simplest fit possible in order to speed up computation and allow us to get results. The main issue with OLS is that there are 10 clusters, and thus the vector $x \in \mathbb{R} * 20$. If there are less than 20 games that a player played in a season, then we will get a "perfect" fit to the data. This means that this method is extremely sensitive to a few outliers in a players performance. In general, since we have at most 82 data points for a player for a season, it is extremely challenging to justify making this profile with just that players games for a season. It would improve the robustness and generality of the data if we factored in other similar player's performances, like using a players cluster as a baseline for the synergy matrix, or using a method like a recommender system to fill in data for players that didn't play games or never played with certain clusters. Improving on the method for calculation of synergy is likely the area where we can improve the model the most without the addition of more data.

In section 5.1, we notice that our model is always scared to predict more extreme game scores resulting in less powerful prediction. In the future, we can measure the variance of a player's history game score and assign each player a new feature called 'stability', measuring whether a player's

performance is consistent. Thus for more inconsistent players, the model are trained to fit more extreme game scores compare to those players that are more consistent.

In section 5.2, we conclude that the synergy vector is the one most frequent and important feature that our tree-based model find. We can develop an algorithmic analysis on the feature importance ranking. Then, based on the ranking, we can readjust the player game score and reweight the model into favoring the better features, further increase the convergence rate and reduce the computation time.

## 7 Conclusion

Our investigation revealed that gradient boosted decision tree-based models can be effective at predicting a player's performance in a specific NBA game based on information known from his earlier career. In fact, when compared to the naive prediction which assumed a player would always perform according to their season wide averages, our novel model improved root mean squared error from 7.333 down to 3.127. This is an improvement of over 50%, which further supports these types of models for tabular data like our own. Our XGBoost based model was also able to outperform other supervised learning techniques such as a simple multi-layered perceptron trained in a regression setting which only achieved a root mean squared error of 6.453. Furthermore, we define a concept of synergy, building upon one presented in the context of baseball score predictions, which allowed us to augment our data with player performances based on the type of other players in the game around them. These new features proved to be quite effective, with them leading to lowered training and validation loss, and them being a significant portion of the causes for splitting within our final trained decision tree model. This ultimately, leads us to conclude that XGBoost was an effective model for the task at hand and the inclusion of synergies was a valuable addition to the data itself. Though there is certainly work to build upon that which is presented in this report, we believe that this is a promising start to a more analytical approach to evaluating player performances based not only on their historic performance, but also the impact of the other players in the game.

## References


[1] R. A. . R. K. A. Brave, Scott A.a; * | Butters. Uncovering the sources of team synergy: Player complementarities in the production of wins. *Journal of Sports Analytics*, 5:247–279, 2019.

[2] F. Dobrykh, S. Muravyov, and O. Ilyasova. Ensemble clustering algorithm development for tabular data by a given partition quality measure. *Procedia Computer Science*, 193:415–421, 2021. 10th International Young Scientists Conference in Computational Science, YSC2021, 28 June – 2 July, 2021.

[3] J. Feng, Y. Yu, and Z.-H. Zhou. Multi-layered gradient boosting decision trees. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[4] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko. Revisiting deep learning models for tabular data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18932–18943. Curran Associates, Inc., 2021.

[5] H. Hussien. Xgballing: Hacking basketball game prediction with ml. *Towards Data Science*, 2020.

[6] B. Shea. Nba viewership grows despite rsn troubles as league banks on future deals. *The Athletic*, 2023.