I. Objective

Daily fantasy sport is a popular online game that allows participants to invest money in the performance of professional athletes, which is measured by the “fantasy score”. In this project, we predicted the fantasy score of athletes using machine learning and applied an optimization technique to determine the weights of a portfolio of athletes. Our machine learning model achieved a mean absolute error of approximately 7 points.

II. Introduction/Background

According to Fantasy Sports Trade Association, over 56 million people played daily fantasy sports (DFS) last year. This popular online game allows participants to invest money in a portfolio of real athletes. If these athletes perform well throughout the season, the participant can earn a return on investment. For the National Basketball Association (NBA) DFS, the performance of the athletes is indicated by the “fantasy score”, which is an aggregate of player performance attributes such as: number of steals, free throws, etc. In this project, the objective can be divided into two folds: the first is to predict fantasy score using machine learning and the second is to use optimization to determine the appropriate weights (amount of money) to invest in the team players. For predictive modelling, our team used boosting algorithms (extreme gradient boosting and gradient boosted machines) to predict fantasy score given information about the players, games, and performance indicators. As for determining the investment weights, we used linear programming optimization. Our team applied this technique to basketball data; however, this technique can be applied to any other type of fantasy sports.

Our motivations for using boosting algorithms are two folds. First, boosting models are frequently used in and have won many machine learning competitions such as Kaggle and the Netflix prize and “has achieved state-of-the art results on many standard classification benchmarks”[1]. Second, our assumption of the data is that the relationship between the predictors is nonlinear and highly complex, for this reason, boosting is an appropriate choice. Since boosting is a non-parametric model with little assumption about the data, this makes it a flexible model to work with and can potentially capture the complex relationships in our data well.

In this report, we present our work in the following manner: section III describes the dataset, feature engineering, and exploratory data analysis, section IV details our model and optimization technique, section V describes our results, and section VI summarizes our findings. Lastly, the appendix in section VII provides the code we used.

[1] https://arxiv.org/pdf/1603.02754.pdf

III. Data

Dataset Description

We retrieved the dataset from the NBA website[2] and NBA Stats API Wrapper in R:nbastatR[3]. The data spans four years from 2015 to 2018. The following table (Table 1) provides a brief description of the features in the dataset. The dimension of the data is 18163 x 52.

Feature Engineering

We further engineered lag variables (last 3 games and last 3 years' averages) to capture previous performances of the athletes. The features used to engineer lag variables are: rebounds, assists, blocks, steals, fouls, turnovers, points, 3-pointers, and results.

Table 1. Dataset Description

Feature Class	Feature	Description
Players	Position	Player's position: C, SF, PF, SG, or PG
	Team Ranking	The ranking of the player's team
	3 Game Lags	How the player played the last 3 games for each criterion
	3 Season Lags	How the player played the last 3 seasons for each criterion
Games	Month	Month of the game, from October to April
	Home/Away	Place of occurrence0=-0-
	Opponent Ranking	Ranking of the opposing team

Response Variable: Player's Fantasy Score in a Particular game

A player's fantasy score in a particular game is given by the following formula:
1**Points* + 0.5**Three-Pointers* + 1.25**Rebounds* + 1.5**Assists* + 2**Steals* + 2**Blocks* - 0.5**Turnovers* + 1.5**Double-Double* + 3**Triple-Double*

A double-double is when a player achieves double digits in two of the scoring criteria in the formula, and a triple-double is when a player achieves double digits in three of these. This calculation of fantasy score is based off of DraftKings' fantasy score calculation. DraftKings is a very popular platform for DFS.

[2] https://stats.nba.com
[3] http://asbcllc.com/nbastatR

Exploratory Data Analysis

In our exploratory data analysis, we began by examining the performance attributes of a single player, Lebron James. Figure 1 illustrates Lebron's performance attributes such as assist (AST), blocks (BLK), steals (STL), and others. As shown in this figure, there isn't a general pattern in his performance, suggesting that a nonparametric model might best capture these complex relationships. We further examined the performance of three teams over time (Figure 2) and saw that there are also no general trends in their performance which further warrant our previous claim of using a nonparametric model.

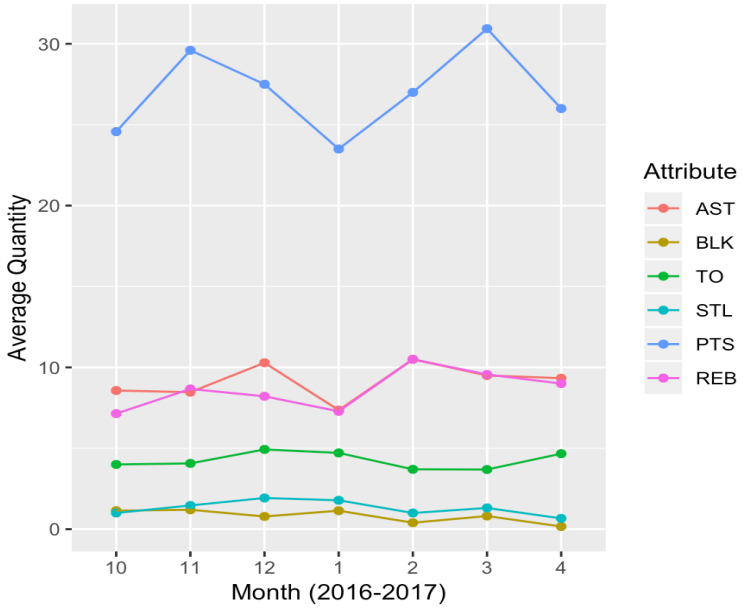

Fig 1. Performance Attributes of Lebron James

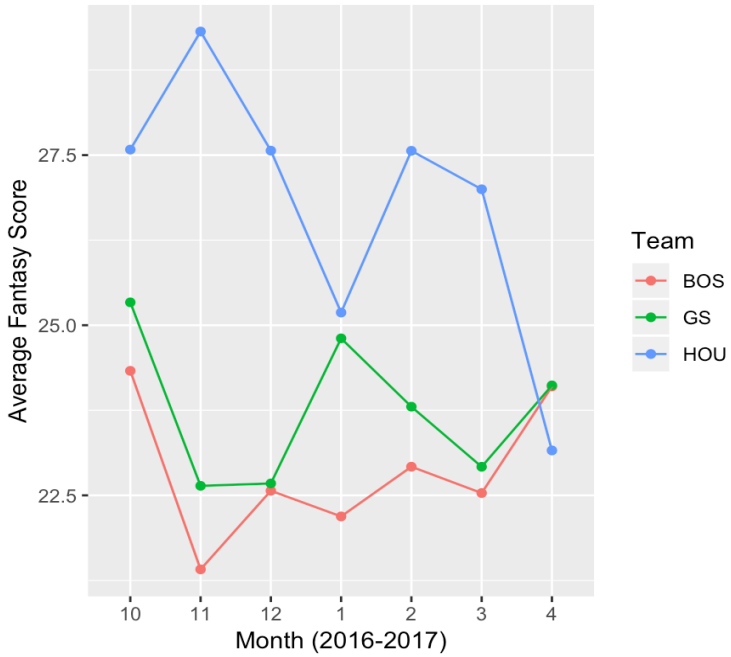

Fig 2. Average Fantasy Scores of Players from Top 3 Teams

We also visualized performance attributes of 5 randomly chosen players (Fig 3). As shown, there are different distributions for different players, which can make it difficult to predict their overall performance.

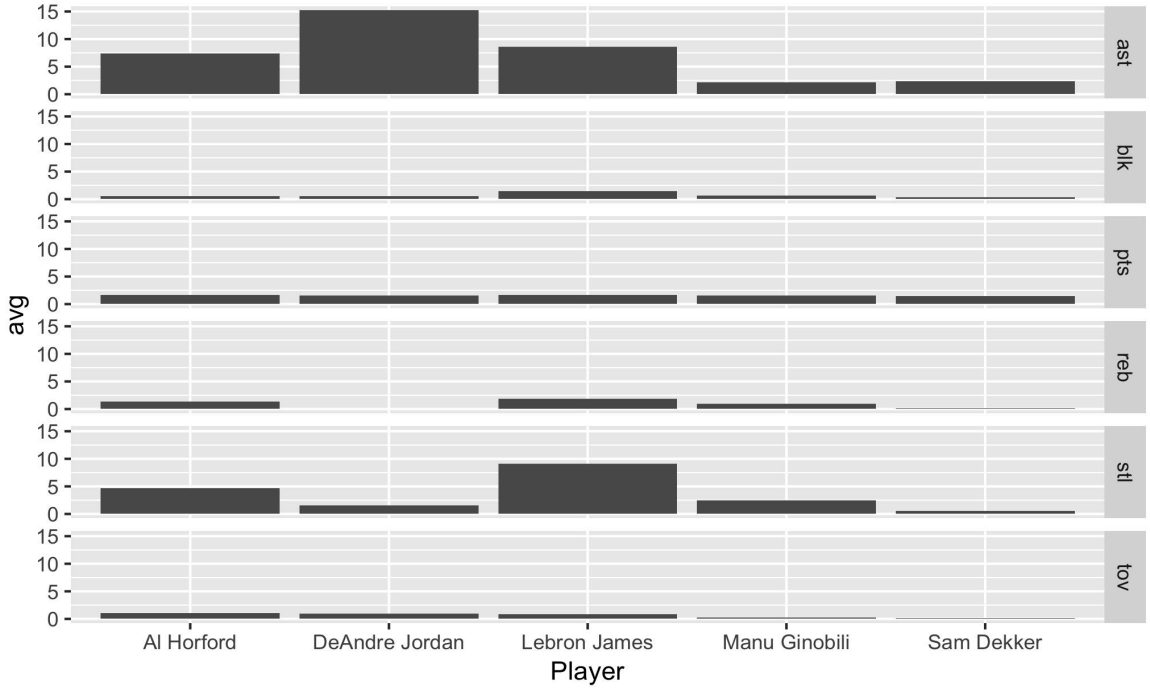

Fig 3. Performance Attributes Distribution for Five Random Players

In Figure 4, we observe the fantasy scores by month and this shows that in some months the players perform better than others, however there isn't a general trend throughout the year.

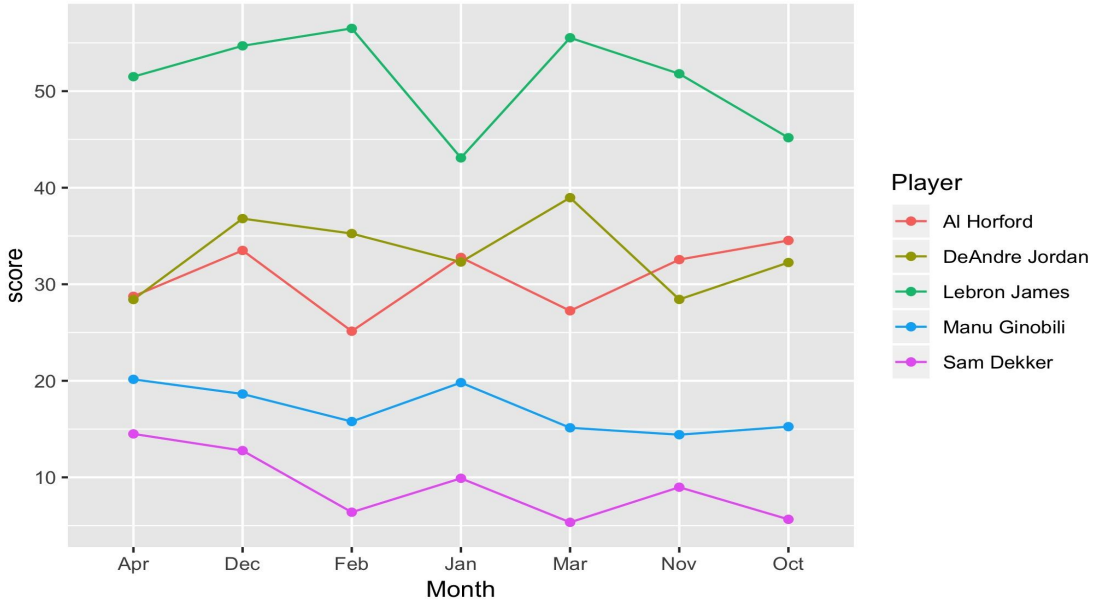

Fig 4. Fantasy Score of Five Random Players by Month

IV. Method

To predict the fantasy score, we begin with a simple parametric model and later tried nonparametric models to see if the performance can be improved. We began with a linear regression model, evaluated its performance and assumptions and subsequently moved to the other nonparametric models, specifically boosting algorithms. Finally, we used linear programming to find the optimal weights in our investment of the portfolio of players.

Baseline Model - Linear Regression

In order to be sure a nonlinear model would be appropriate for this data, we fitted a baseline linear regression model using the predictors mentioned above and fantasy score as the outcome variable. The adjusted R-squared of the linear model is 0.497. The qq-plot of the residuals is displayed in Figure 5. As shown, the residuals do not appear to be normal. We further tested this using Kolmogorov-Smirnov test for normality and obtained a p-value less than 2.2e-16, suggesting that indeed our residuals are not normally distributed. When we observed our residuals (Fig 6), they also show that there isn't homoscedasticity since the variance appears to be higher as the fitted values increase. In summary, when performing diagnostics of our model, it violates the normality and constant variance assumptions. This outcome further supports our decision to use a nonlinear machine learning method to predict fantasy score.

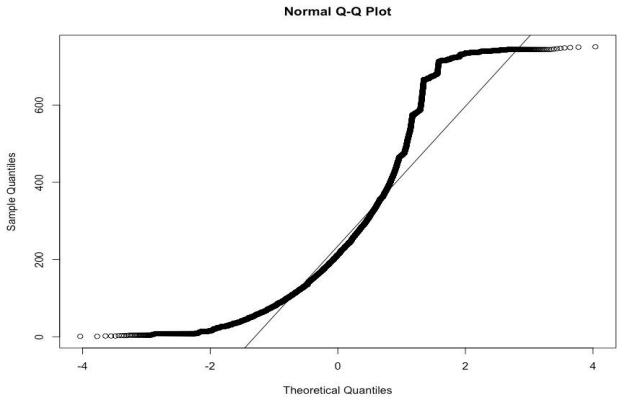

Fig 5. Actual vs. Theoretical Residual Quantiles for Linear Regression Model

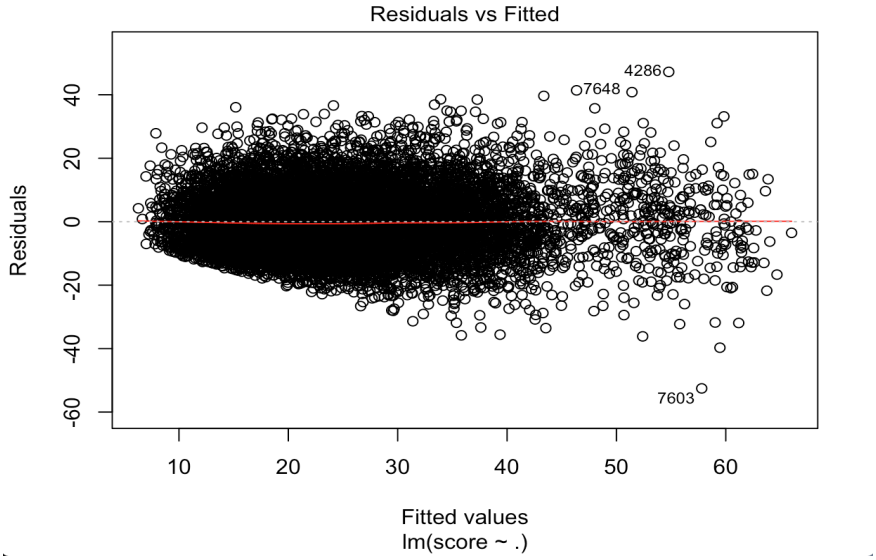

Fig 6. Residual vs. Fitted Values of Linear Regression

Predictive Modelling

Once we finished our EDA and baseline model, we proceeded to fit a machine learning model to predict a player's estimated fantasy score in a particular game. As described previously, the response variable would be the fantasy score for that game. The goal of this machine learning model is to use information we know about an upcoming game (team's ranking, opponent's ranking, 3 game lags for this player, 3 season lags for this player, home/away, and month of game) to predict a player's fantasy score for this game. We would use the predicted fantasy scores for each player to put together our fantasy team for that day.

We selected a gradient boosting model (GBM) and XGBoost for comparison. The full data is first randomly divided into training and test datasets with 75/25 consensus. We tried fitting these models on both the full training dataset, and a different model for each position (C, SF, SG, PF, and PG). To tune our model, we created a tuning algorithm, which is presented in the appendix. The algorithm takes a range of values or a fixed value for each parameter:

t: number of trees, which is equivalent to number of iterations of the algorithm. We set the initial number as high as 2000.
i: interaction depth, which is related to interaction terms and how "deep" each level of the tree goes. It is tuned from 1 to 10.
s: shrinkage, also known as step size or learning rate. Usually a decimal less than one. Start with a broad tuning, like seq(from=0.1, to=1, by=0.1), and then do a finer tuning, rounding to the nearest hundredth.
n: n.minobsinnode, which is the minimum observations in terminal nodes. Usually an integer from 1 to 15.
b: bag fraction. At each iteration, only a percentage of predictors are randomly predicted to use for each decision tree, which keeps it random and helps to avoid overfitting. This parameter is first tuned between 0.30 to 0.90 (which means 30% to 90%) and finer tuned to the nearest hundredth.
k: number of folds in k-fold cross validation. It is set as 5 in the model.

The algorithm would output the cross-validation error for each parameter, and the optimal iteration number(t). For example, if we wanted to fix shrinkage(s) at 0.02, bag fraction(b) at 0.44, and number of folds(k) at 5, and tune interaction depth(i) in the range [7,9] and number of terminal nodes(n) in the range [6,7] with 2000 trees, we would get the following table in R:

Table 2. Parameter Tuning Table (From R)

trees	interdepth	shrinkage	n.minobosinnode	bag.fraction	k	error
1147	7	0.02	6	0.44	5	7.5925
943	7	0.02	7	0.44	5	7.5013
1897	8	0.02	6	0.44	5	7.5029
1201	8	0.02	7	0.44	5	7.5031
1003	9	0.02	6	0.44	5	7.4963
1143	9	0.02	7	0.44	5	7.5000

We would use this output to determine further tuning and find the parameter set achieving the minimum cross validation error. After fitting all the models, we found that they all performed very well, but **GBM subsetted by position** was the model that performed the best. Following is the set of parameters of the best performing model:

Table 3. Parameters for Best Performing Model -- GBM subsetted by position

Position	trees	interdepth	shrinkage	n.minobsinnode	bag.fraction	k
C	309	7	0.02	7	0.7	5
SG	765	8	0.01	20	0.35	5
PG	61	8	0.08	20	0.6	5
SF	733	9	0.01	6	0.46	5
PF	40	10	0.01	14	0.7	5

The results for evaluation metrics of best performing model are listed below:

Table 4. Evaluation Metrics for Best Performing Model -- GBM subsetted by position

Position	Cross Val Mean Abs Error	Median Abs Test Error	Mean Abs Test Error	Median Percent Test Error	R Squared (Test)	R Squared (Train)
C	7.73	6.21	7.56	26.99%	48.54%	58.71%
SG	7.41	5.84	7.21	29.40%	49.50%	58.70%
PG	7.82	6.38	7.89	25.90%	45.10%	56.60%
SF	7.09	3.84	5.48	19.20%	54.40%	58.40%
PF	7.29	6.39	7.69	29.90%	54.80%	65.30%

As we can see from the evaluation results, the median absolute test error in each sub-model is around 6 points and the mean absolute test error is around 7, which is relatively small. Using this prediction model, we are able to forecast the fantasy score for each player from different positions.

Investment Optimization

After we got the predicted score of each player in each game, the best team should be chosen within the $50,000 budget. Due to the overwhelming large size of the data set, we developed an algorithm that can guarantee optimization within an acceptable running time.

Step1:
Remove the players who are marked as either O (out) or Q (questionable) from the player pool, which indicates that they are less likely to actually play the game on a specific date.

Step 2:
Subset the player pool by predicted fantasy scores, leaving the top 20% players. This subsetting ensures we choose among the best performing players in each position.

Step 3:
Define the following notations for player i:

1) Predicted fantasy score S_i
2) Salary C_i
3) Position $P_i = \{P_{i1}, \ldots, P_{i8}\}$ Where

$$j \in \{PG, SG, SF, PF, C, F, G, U\}$$

$$P_{ij} = 1 \; if \; player \; i \; players \; the \; position \; j \in \{PG, SG, SF, PF, C, F, G, U\}$$

This problem is a discrete optimization problem with target function as

$$max \; \Sigma_i S_i$$

And with the following constraints:

$\Sigma_i C_i \leq 50{,}000$	The cost must be less or equal than $50,000
$\Sigma_j P_{ij} = 1 \; for \; all \; j$	Each player can only play one position
$\Sigma_i P_{ij} = 1 \; for \; all \; i$	Each position can only be assigned one player

After we formed all the possible combinations of the players in the new player pool we got from step 2, we imposed restrictions as explained in step 3. This chooses the team with the highest total score within the $50,000 budget constraint. We also considered the fact that each player can be assigned to many different positions, which ensures that we assigned players to the optimal position. In this way, we got the best team with the highest total predicted score under the budget limit.

Step 4:
After the games are over, we calculated the real scores of our team based on the game log and checked how our prediction was close to the players' real performance. This measures the accuracy of our model in predicting the score of a team. For the most recent two games, we participated in the real fantasy basketball games to gauge the profitability of our model.

V. Model Application Results

We applied our algorithm for the past 3 days, December 6th to 8th, to check the accuracy of our model when compared to the real scores. Apart from the pure simulation and checking only the prediction accuracy, we actually participated in the most recent real fantasy game to see how we rank among the players participating in the fantasy game and the profitability of our model.

Fantasy Score Prediction Results

First, the GBM model we got in the previous section is used to predict fantasy score for individual players on daily basis from 12/06 to 12/08. Following are the evaluation results:

Table 5. Prediction Evaluations from 12/06 to 12/08

Date	Median Absolute Prediction Error	Median Percent Prediction Error	Mean Absolute Prediction Error
12/06/18	6.03	37.59%	8.72
12/07/18	7.29	33.06%	9.18
12/08/18	7.84	35.37%	9.21
12/08/18 (Night time)	5.91	33.33%	8.63

We calculated error rates by comparing the real fantasy scores with our predicted ones. The results are similar to the one from modelling. The median absolute test error is around 7 and the mean absolute test error is around 8.5, which is slightly worse than the modelling result.

Fantasy Basketball Team Optimization Results and Evaluation

Table 6. Results on 12/06/18

	name	salary	score_predicted	score_real
0	Al Horford	6200.0	22.309097	47.00
1	Kevin Knox	4500.0	27.754042	19.00
2	Mario Hezonja	3200.0	10.350022	8.75
3	James Harden	10800.0	56.149145	17.75
4	Kyrie Irving	9000.0	37.937631	41.50
5	Enes Kanter	6800.0	29.545050	32.25
6	Eric Gordon	5000.0	31.821960	8.25
7	Jaylen Brown	4500.0	32.294358	26.75
Total	NaN	50000.0	248.161305	201.25

On 12/06, our model overestimated the real score of the team by 46.9 points.

Table 7. Results on 12/07/18

	name	salary	score_predicted	score_real
0	Jimmy Butler	7400.0	43.058766	63.00
1	JJ Redick	5400.0	28.361545	30.00
2	Tyreke Evans	5000.0	32.583880	20.25
3	Jamal Murray	7100.0	46.223538	40.25
4	Dennis Schroder	5500.0	26.638946	29.75
5	LaMarcus Aldridge	7600.0	32.630041	27.50
6	Paul Millsap	6400.0	34.426374	26.75
7	Trevor Ariza	5600.0	30.783922	28.25
Total	NaN	50000.0	274.707012	265.75

On 12/07, our model overestimated the real score of the team by 8.96 points.

Table 8. Results on 12/08/18

	name	salary	score_predicted	score_real
0	Tyreke Evans	4900.0	22.581577	27.00
1	Kent Bazemore	4800.0	24.410348	35.00
2	Damyean Dotson	4500.0	20.053716	19.25
3	Zach LaVine	7600.0	40.118543	16.75
4	Lou Williams	5300.0	33.395521	28.75
5	Damian Lillard	9400.0	51.040071	49.50
6	Jusuf Nurkic	7500.0	35.275914	54.25
7	Marvin Bagley III	5300.0	18.198245	9.00
Total	NaN	49300.0	245.073935	239.50

On 12/08, our model overestimated the real score of the team by 5.58 points.

Table 9. Results on 12/08/18 10pm (Night)

	name	salary	score_predicted	score_real
0	Andrew Wiggins	5700.0	29.069173	33.50
1	Dwyane Wade	5400.0	27.437624	46.50
2	Lou Williams	5300.0	33.853894	28.75
3	Damian Lillard	9400.0	50.953480	49.50
4	Jusuf Nurkic	7500.0	34.949511	54.25
5	Kelly Olynyk	5100.0	31.588585	39.50
6	Al-Farouq Aminu	4600.0	28.209019	32.50
7	Danilo Gallinari	6900.0	26.457360	31.00
Total	NaN	49900.0	262.518645	315.50

On 12/08 real night game, our model underestimated the real score of the team by 52.9 points, performing relatively poorly in predicting the real scores as compared to our other four games. However, within the context of the real competition, we ranked 27th among 1585 participants (top 1.7%), resulting in a 50% return (we invested $10 and won $15).

Although our model prediction often differed from the true team sum of fantasy points by as much as 20%, our optimization process shows promise as a successful method of drafting a fantasy team with high performance. Furthermore, since there are random components to every professional sports game, our values for test error do not seem inappropriate, and perhaps indicate that our model accounts for some of these random components, Since the goal of our project was to draft the fantasy team within a certain budget that would win the most money in DFS, and not necessarily to predict a team's fantasy point total with the highest accuracy, the random components of our model may ultimately allow us to earn more return over time. Given more time and resources, we could further test our model's accuracy and overall return by competing in more DFS games.

Table 10. Summary of Results

Date	Total Predicted Score	Total Real Score	Percent Difference
12/06/18	248.16	201.25	18.90%
12/07/18	274.71	265.75	3.26%
12/08/18	245.07	239.50	2.27%
12/08/18 (Night time)	262.52	315.50	-20.18%

VI. Discussion

We tested our model by using it to predict the actual game results in the real fantasy game. As shown in Table 10, our model performs very well with the absolute percent differences between the total predicted and actual scores to be less than 25% for any given day between December 6th to December 8th of 2018. These small error rates suggest that boosting algorithm can capture the relationship between the predictors and the fantasy score well. For future work, our team can try more flexible models such as deep learning to see if we can improve the performance. Additionally, we would like to automate the modelling process so that the model takes in the most recent data to ensure the latest information. We hypothesize that both approaches can improve our results.

VII. Appendix

A. Key R Code (see attached)
B. Python Optimization Code (see attached)