Final Project: NBA Fantasy Score Prediction

Connor Young (cey23), Andrew Koo (alk272), Saloni Gandhi (sg2452)
CS5785: Applied Machine Learning
Fall 2020
Cornell Tech
New York, NY 10044

Abstract

In order to succeed as a DFS player, one has to complete two difficult tasks: (1) accurately predict each player's value and (2) strategically select a lineup based on one's beliefs. The DFS world has advanced considerably in recent years with the widespread adoption of optimization methods and published research about selection strategies to solve the second problem. However, the first problem of prediction remains unsolved and highly subjective. We present methods to apply machine learning for prediction of player fantasy values using only free and open-sourced tools and data. Ultimately, we train XGBoost trees and achieve a performance 8.58% greater than the default projection native on the fantasy platform. Additionally, we reflect on the features driving this performance and identify opportunities for qualitative insights to further improve results.

1 Introduction

In this paper we explain the process of developing a machine learning model to predict performance of NBA players in regular-season games. These projections come in the form of a player's stat-line for the game (points, assists, etc.) and an associated fantasy score. We will cover the background necessary to understand this work, problem formulation and feature decisions, a comparison of the different model types that were tested, and key takeaways from the results. We will also touch on how the results fit with prior work in the area and inform future work.

2 Background

In order to understand the context of the results, a brief primer is needed on Daily Fantasy Sports (DFS) and the National Basketball Association (NBA). DraftKings, a dominant player in the DFS market, runs thousands of contests daily where users select a "lineup" of real-life professional athletes. These athletes receive a "fantasy score" based on their performance, or stat-line, in their game that day, and the contest is won by the entrant(s) with the highest combined fantasy score across all the players in their lineup. In the NBA specifically, players are scored based on their amount of points (PTS), rebounds (TRB), assists (AST), steals (STL), turnovers (TOV), 3-pointers made (3P), and blocks (BLK), with a different weight assigned to each. In order to prevent entrants from just selecting the top players in the NBA for their lineup, DraftKings enforces constraints on player positions and a salary cap of $50,000 on the draft, with better players costing more. Thus, the crux of the problem is in trying to determine and predict fantasy scores for all players so that the most efficient lineup can be constructed in terms of fantasy value per salary unit.

3 Method

Existing attempts at applying machine learning to fantasy value projection involve directly predicting a player's fantasy score from a set of player features. However, based on the observation that fantasy scores are a known linear combination of stat-lines and weights, we hypothesize that building custom models for each base statistic and using those predictions as inputs for the fantasy score would be a more precise model for fantasy predictions. In the DFS industry, there are three major players (DraftKings, FanDuel, and Yahoo!), each of which computes their fantasy score slightly differently. For the purposes of this project, we decided to focus on DraftKings predictions as it is the largest platform with the most external data and available research, but our findings should generalize to other fantasy platforms as well.

3.1 Data Collection and Preparation

We selected StatHead as our primary data source for player stat-lines. StatHead offers free, high-quality player stat-lines from 1946 - present, but does not currently support exporting data at scale. To work around this, we built a Python web scraper utilizing Selenium to collect data from the past 6 seasons (2014 - present, resulting in 163270 stat lines in total). We one-hot encoded non-ordinal data (such as Team and Opponent) and label-encoded ordinal and binary categorical columns (such as Home/Away). During our initial data cleaning, we decided to remove playoff games and players who played no minutes to simplify the game environment to model.

3.2 Feature Engineering

We structured time-series stat-line data into multiple rolling average features to represent short- and long-term trends. We also explicitly modeled aspects of sports commentary into our features so that we can measure their predictive power compared to purely historical, trends-based features.

The list of features that we created based on our research and sports experience include:
Team: The current team a player is on
Opponent: The opponent a player is facing
Home or Away: Whether the game is home or away
Position: What position the player plays
Rolling averages of each individual player stat over 3, 5, and 10 game windows (i.e. points scored over last 3 games, points scored over last 5 games, points scored over last 10 games, rebounds over last 3 games, rebounds over last 5 games, rebounds over last 10 games, etc.). There are a total of 24 individual player stats we are measuring, resulting in 72 additional features.
Odds: The probability of the player's team winning this game, according to Vegas betting odds

3.3 Model Selection

We considered various approaches for formulating the problem, including predicting the fantasy score directly, training models on solely a player's personal history versus the entire league's history, lineup position-based models, team-based models, and more. Ultimately, we trained one regression model per statistic using league-wide data and then computed fantasy score from the seven key stat predictions using the DraftKings fantasy score formula. In terms of the machine learning regression model selection, we prioritized models which perform well on high-level structured data (both discrete and continuous) and which could facilitate interpretation of feature importance. Based on these criteria, we test two models: random forest (sklearn) and regularized gradient boosted trees (XGBoost).

3.4 Training and Tuning

We split the full dataset into three partitions: the train, dev, and test sets. The train set contains our first four seasons of data, the dev set contains the next season, and our test set contains the last season. By splitting the data based on seasons (rather than via randomized split) we can potentially observe whether the model generalizes across time. We trained our models on the train set and tuned model hyperparameters using grid search, evaluating periodically by comparing the generalization of the train set's mean squared error to the dev set's mean squared error.

3.5 Evaluation and Benchmarking

In practice, DFS participants are given each eligible player's salary, positions, and fantasy points per game (FPPG), where FPPG is a simple long-term average of that player's fantasy performance. We recomputed a proxy for this given metric as a 20-game moving average of fantasy score and benchmarked our algorithm's mean squared error against the 20-game moving average's mean squared error. In addition to measuring the model's quantitative performance on the test set, we examined the feature importances in our final model to gain qualitative understanding of relative feature impact on the final result.

4 Experimental Analysis

We split our experiment analysis into two main sections: (1) comparing a single model which predicted a fantasy score to seven dedicated models, each of which predicted a single player statistic; and (2) comparing the performance of different machine learning models, specifically seven random forest regressors to seven XGBoost tree regressors. The goal in the first section is to test our hypothesis that decomposing the fantasy score prediction problem into stat line subproblems will improve model accuracy. The goal in second section is to optimize the XGBoost machine learning ensemble via hyperparameter tuning to outperform the out-of-box random forest ensemble.

Table 1: Final R^2 Values of Various Model Types on Data Sets

	Single Random Forest			Seven Random Forests			XGBoost		
Statistic	**Train**	**Dev**	**Test**	**Train**	**Dev**	**Test**	**Train**	**Dev**	**Test**
Points				0.931	0.522	0.527	0.511	0.527	0.531
Rebounds				0.927	0.488	0.472	0.484	0.498	0.486
Assists				0.935	0.512	0.522	0.541	0.525	0.535
Steals	0.909	0.335	0.330	0.877	0.104	0.079	0.144	0.143	0.125
Turnovers				0.900	0.301	0.310	0.300	0.325	0.331
3-Pointers				0.908	0.331	0.326	0.354	0.352	0.341
Blocks				0.890	0.199	0.184	0.225	0.234	0.229

Table 2: Evaluation of Models vs. Benchmark

	Single Random Forest		Seven Random Forests		XGBoost	
Metric	**Dev**	**Test**	**Dev**	**Test**	**Dev**	**Test**
Model MSE	93.156	94.605	91.354	93.183	90.407	92.269
Benchmark MSE	94.939	100.931	94.939	100.931	94.939	100.931
Improvement	1.783	6.326	3.585	7.748	4.533	8.662

4.1 One Random Forest vs. Seven Random Forests

In our original discussion, we hypothesized that seven random forests should outperform one random forest by simplifying the training process for each forest. Preliminary results validate our hypothesis, as the one random forest model slightly under-performs the seven dedicated random forest models.

This result is consistent with our intuition, as rather than requiring one model to learn the idiosyncrasies of each stat line prediction and also the relationship between each stat line, we instead task seven models to each learn just one stat line's behavior. Despite the improvement, the marginal gain was relatively small compared to the total improvement over baseline. After examining the most important features driving our best models (Table 3), this outcome makes sense as we see that only one or two features dominate each prediction model. In other words, we believe that each

dedicated model was sufficiently simple with the given set of features that the single random forest was able to capture a majority of the same insight.

4.2 Seven Random Forests vs. Seven XGBoost Models

Following the experiment in section 4.1, we observed that while seven random forests performed well on the dev set compared to baseline, the default forests had high variance resulting in overfitting, as indicated by the large decrease in R^2 scores from train to dev.

Due to this, we attempted to improve our prediction accuracy by utilizing a more complex and difficult-to-tune model: XGBoost's gradient boosting algorithm. This model does not have as consistently good out-of-box performance as random forest, but has the potential to perform much better if hyperparameters are tuned correctly. Gradient boosting machines (GBM) differ from random forests in how they build their trees. In random forests, each tree is built independently and results are combined at the end to develop an average model. In GBMs, trees are built sequentially, combining results along the way so that the following tree improves upon the errors of the previous.

XGBoost's implementation of gradient boosting uses a more regularized model formalization than other GBMs, which helped with the overfitting problem realized in the random forest approach. Evidence of this can be seen in comparing the R^2 scores for XGBoost on the train versus dev set, where the difference is much smaller than the random forest implementations. On some statistics, the XGBoost implementation is actually more accurate on the dev set than the train.

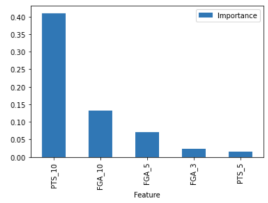
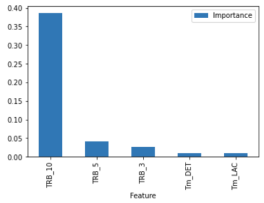
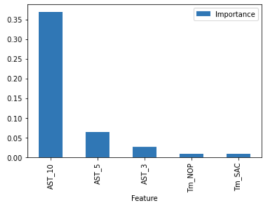
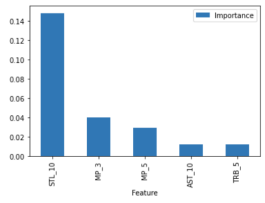
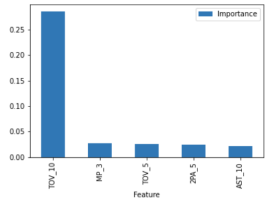
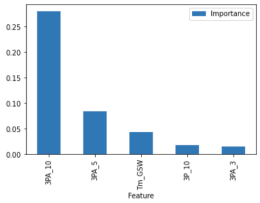
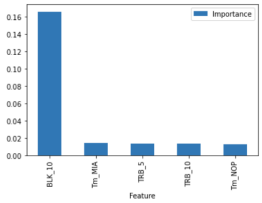
To tune our hyperparameters, we conducted a grid search on the following fields:

colsample_bytree, learning_rate, max_depth, min_child_weight, n_estimators, objective, subsample

Through careful tuning of the seven XGBoost models, the ensemble outperformed the default random forests and generalized well to the dev set. Finally, we tested all models on the test set and confirmed that the XGBoost model generalized the best to the test set as expected.

4.3 Analysis of XGBoost Feature Importances

Table 3: Feature Importance for Prediction of Various Statistics

Points	Rebounds	Assists	Steals
			
Turnovers	**3-Pointers**	**Blocks**	
			

In addition to creating a highly performant model, another motivation for this project was to compare the influence of quantitative player trends versus the qualitative traits that drive sports narratives, such as matchups. By visualizing the top 5 most important features in our tuned XGBoost models (Table 3), we see that the quantitative rolling average features across 10 games were the strongest for their respective statistic. For example, the strongest predictor for blocks by large margin was simply the rolling 10-game average of blocks by that player.

Player statistics that have inherently higher volume, such as points, assists, and total rebounds were generally the most predictable (as given by the R^2 scores in Table 1). Consequently, this also enabled shorter rolling timelines to be useful in their predictions. Conversely, metrics with smaller volumes such as steals were too inconsistent to be measured on shorter timelines, and all the purely quantitative features struggled to reliably predict these metrics. Interestingly, for higher variance metrics the models attributed importance to qualitative, discrete features. For example, the one-hot encoded team feature for the Golden State Warriors, a team known for its long-range shooting, rose to the top-5 features of the 3-point prediction model. This may indicate an opportunity to include customized qualitative features for these models to better determine low-volume statistics in the future.

5 Discussion, Prior Work, and Future Work

DFS is a large and rapidly growing industry, evidenced by market leader DraftKings reporting 42% year-over-year growth with revenues of $133 million in its most recent quarterly earnings (DraftKings 2020). Increased interest has pushed participants to develop advanced techniques in this highly competitive environment, especially with regards to optimizing lineup selection strategies. One research lab achieved success by formulating lineup selections as a type of covering problem (Hunter 2016).

On the prediction side of DFS, a host of sites such as Daily Fantasy Nerd have evolved to offer paid services, however projecting player values remains largely opinionated. Previous attempts at employing machine learning models to directly solve this problem have been met with mixed results (Harner 2020). In this project, we decomposed the prediction problem into seven subproblems which improved our model's accuracy. Via this decomposition, we identified that the biggest errors in fantasy score predictions stems from low volume statistics. This presents an opportunity to augment those models with idiosyncratic qualitative features in the future, such as opponent matchups or defensive rank. As a tangential extension of error analysis, we may also use our model to measure marginal impacts of irregular events such as playoff games or the shift to the NBA "Bubble" on fantasy score expectations.

6 Conclusion

With our best model (XGBoost) we were able to improve upon DraftKings published projections by 8.58%. An example of a selected contest lineup using our model is shown in Table 4 below, and one can see how our projection compares to DraftKing's projections and to actual performance. This lineup was created by plugging our score projections into a linear programming tool that maximizes fantasy score within the $50,000 salary cap and positional constraints of the contest. The salary and performance data comes from a random, real contest from November 2019. Although our model still deviates from true performance, it correctly predicts the increase in relation to the default projections. In addition, it picks a roster that achieves an overall actual score of 337, which is more than enough to place in the money for an average DraftKings contest.

Table 4: Example of an Optimal Lineup Constructed Using XGBoost Model Output

Player Name	**Position Slot**	**DK Salary**	**DK Projection**	**Model Projection**	**Actual Score**
Jordan Clarkson	PG	$3,800	22.375	24.378	33.50
Luka Doncic	SG	$9,700	61.825	61.553	73.75
Bruce Brown	SF	$3,100	20.738	22.468	9.25
Giannis Antetokounmpo	PF	$10,900	62.488	63.058	58.00
Andre Drummond	C	$9,500	48.838	54.813	67.00
Marcus Smart	G	$4,900	26.688	28.208	21.75
Jordan Poole	F	$3,200	17.675	24.083	23.00
Marcus Morris	Util	$4,800	31.313	31.985	50.75
Total	-	$49,900	291.938	310.543	337.00

References

[1] Basketball Reference, 2020, basketball-reference.com.

[2] "DraftKings NBA Classic Rules." DraftKings, 2020, www.draftkings.com/help/rules/nba.

[3] Barry, Christopher, et al. Winning the Draft: Maximizing Value in Daily Fantasy Sports. 2016, web.stanford.edu/class/stats50/projects16/BarryCanovaCapiz-slide.pdf.

[4] Harner, Nate. "Creating a Fully Automated Daily Fantasy Sports Strategy." Medium, Towards Data Science, 12 May 2020, towardsdatascience.com/creating-a-fully-automated-daily-fantasy-sports-strategy-6842d2e1ccb6.

[5] Du, Alan. "Daily Fantasy Basketball - DraftKings NBA." Kaggle, 29 Dec. 2017, www.kaggle.com/alandu20/daily-fantasy-basketball-draftkings.

[6] Goldstein, Omri. "NBA Players Stats since 1950." Kaggle, 27 Apr. 2018, www.kaggle.com/drgilermo/nba-players-stats?select=player_data.csv.

[7] "Sportsbook Reviews." Historical NBA Scores and Odds Archives, 2020, www.sportsbookreviewsonline.com/scoresoddsarchives/nba/nbaoddsarchives.htm.

[8] "Daily Fantasy Projections." Sports Analytics Group Berkeley, 2020, sportsanalytics.berkeley.edu/fantasyprojections.html.

[9] "Historical Basketball Statistics." Stathead.com, 2020, stathead.com/basketball/.

[10] "Historical DraftKings Contest Data." Daily Basketball Points, 2020, rotoguru1.com/cgi-bin/hyday.pl?game=dk.

[11] Hunter, David, et al. Picking Winners Using Integer Programming. 2016, mitsloan.mit.edu/shared/ods/documents/?DocumentID=2858.

[12] Don Cazentre. "Daily Fantasy Sports Contests Are Illegal in NY, Court Rules." Newyorkupstate, 6 Feb. 2020, www.newyorkupstate.com/sports/2020/02/daily-fantasy-sports-contests-are-illegal-in-new-york-court-rules.html.

[13] DraftKings Inc. "DraftKings Reports Third Quarter Results and Raises 2020 Revenue Guidance." GlobeNewswire News Room, "GlobeNewswire", 13 Nov. 2020, www.globenewswire.com/news-release/2020/11/13/2126388/0/en/DraftKings-Reports-Third-Quarter-Results-and-Raises-2020-Revenue-Guidance.html.

[14] Płoński, Piotr. "Does Random Forest Overfit?" MLJAR, Piotr Płoński, 5 Apr. 2019, mljar.com/blog/random-forest-overfitting/.

[15] Glen, Stephanie. "Decision Tree vs Random Forest vs Gradient Boosting Machines: Explained Simply." Data Science Central, 28 July 2019, www.datasciencecentral.com/profiles/blogs/decision-tree-vs-random-forest-vs-boosted-trees-explained.

[16] Jain, Aarshay. "XGBoost Parameters: XGBoost Parameter Tuning." Analytics Vidhya, 1 Mar. 2016, www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/.