Beating the Daily Fantasy Basketball 50/50 Challenge

John Goettle JGOETTLE@SEAS.UPENN.EDU
Mark Bloom MCBLOOM@WHARTON.UPENN.EDU
Erica Winston EWINSTON@SEAS.UPENN.EDU

Abstract

In this paper, we explore the application of machine learning in daily fantasy sports (DFS). In particular, we are interested in beating the NBA 50/50 DFS challenge, where the top half of teams receive winnings proportional to their stake. In the 50/50 challenge, fantasy sports bettors compete against other fantasy sports bettors in the competition, contrasting traditional sports gambling where sports bettors compete against the sportsbook. While sportsbooks odds have proven difficult to beat, we propose that beating half of the sports gamblers in a competition pool is not quite as difficult. We exploited this by developing and validating a neural network with a quantile loss function that theoretically has the capability of generating a consistent profit in the NBA 50/50 Fanduel DFS challenge.

1. Introduction

In Fanduel DFS, sports bettors assemble virtual teams of athletes and compete against other bettors (Fanduel is a popular online fantasy sports platform). Each athlete has a salary, and all bettors are subject to the same salary cap - the sum of the salaries of the athletes on any given team may not exceed the salary cap. The number of fantasy points an athlete scores reflects his actual performance, calculated as his box score multiplied by a set of coefficients. An athletes box score includes points, rebounds, assists, steals, turnovers, and a few other stats achieved in a single game. For example, if an athlete X has five baskets, each worth four fantasy points, and three rebounds, each worth 2 points, X will score a total of $5 * 4 + 3 * 2 = 26$ fantasy points. Athletes projected to score a high number of fantasy points will have a higher salary; thus, there is a trade-off inherent to picking any athlete. Each athlete has a position, and each fantasy team must have a certain number of athletes at each position. A DFS competition runs for a single day, at the end of which winners are chosen based on some threshold. We are only concerned with the 50/50 challenge in which sports bettors scoring more than the median number of points win money relative to their initial bet.

If a Fanduel 50/50 sports bettor bets \$10 and scores in the top half of the sports bettors, he will win \$8 and walk away with \$18. On a \$10 bet, letting P be a random variable for profit and p the probability of a win, we can find the break-even win percentage by solving for $E[P] = 0$.

$$\begin{aligned} E[P] &= Pr(Win) * reward + Pr(loss) * bet \\ 0 &= p(\$8) + (1-p)(-\$10) \\ \$18p &= \$10 \\ p &= 0.555 \end{aligned}$$

Thus, we measure the success of our approach by comparing the frequency in which we output a lineup that beats 50% of the pool to the 55.5% winning percentage necessary to generate a consistent profit.

We designed a two-step approach to building a winning team:

1. Predict each athlete's fantasy points with a prediction interval.
2. Assemble a team based on predicted scores, prediction interval, and salary for each athlete to maximize the likelihood of scoring in the top half of bettors.

While our model trained on the error produced by miscalculating an athlete's fantasy points, we validated the success of our entire approach by comparing the output score from the second step to the points needed to be in the top 50% for the competition.

2. Background

In traditional sports gambling, sportsbooks create gambling odds and allow bettors to bet on positions, such as an athlete scoring over or under a given number of points. Several betting strategies have been proposed to beat bookmaker

University of Pennsylvania, CIS 419/519 Course Project.

odds, from prediction models and arbitrage strategies to odds bias exploitation. Still, very few (if any) have had consistent and sustainable success (Lisandro Kaunitz & Kreiner, 2017). The reason for this phenomenon is simple - sportsbooks are good at generating gambling odds. They employ teams of data scientists to build sophisticated forecast models for every game in every sport, virtually eliminating the potential that an outsider with access to much less data might be able to achieve an enduring information-driven edge. In daily fantasy sports, in contrast, sports bettors compete against other sports bettors rather than the sportsbook. We hypothesize that the average sports gambler is not quite as calculated as the sportsbook. Therefore, DFS may present a market suitable for the exploitation of an information-driven edge provided by machine learning.

We have chosen to model DFS data specifically for the NBA due to the increased relative role of skill in basketball gambling. Compared to other sports, fantasy basketball is much more driven by skill than luck because NBA players are more likely to have consistent performance. (Getty & Hosoi, 2018). We theorize that introducing machine learning into a environment with minimized chance will provide us with the sufficiently consistent predictive power needed to generate a profit.

2.1. Related Work

While our approach in predicting DFS points for basketball is unique, machine learning in sports gambling is not a novel concept. (Brent Evans, 2018) provides a data-driven approach to quantifying skill in daily fantasy basketball and was one of the motivating studies for our paper. The researchers discussed techniques unique to skilled sports bettors that we considered in building our model. (Alexandre Bucquet, 2018) presents interesting conclusions on the viability of predicting the total number of points scored in an NBA game to gain an advantage. This approach is different than ours in that we are solely concerned with daily fantasy sports, but nonetheless, the conclusions are motivating. Other studies exploiting the sports-betting market for the NBA, including one which adopted modern portfolio theory to design a strategy for bet distribution according to the odds and model predictions (Hubáček & Železný., 2019). We may incorporate this approach into our methodology in the future.

3. Implementation

3.1. Data Collection

We collected data for every individual NBA athlete and team for every single game between the 2014 and 2019 NBA seasons. There was not a single online database that provided all of the information we needed, so we built web crawlers using Selenium [1], BeautifulSoup [2], and a chrome driver. Each data source had the primary key {Name, Team, Game Date } on which we were later able to join the tables.

Nba.com provided the best box score and advanced basketball statistics data. A simplified example of an athlete's box score may be the following:

{Name : 'Lebron James', Team : 'LAL', Match Up : 'LAL vs BKN', Game Date : 3/10/2020, W/L : 'L', Pts : 29, Reb: 12, Ast:9, Stl:1 ....}

Nba.com has 5 player statistics categories: traditional (ex. points, rebounds), advanced (ex. effective FG %, pace), misc (ex. points off a turnover), scoring (ex. % points 3-pointers), and usage (ex. % points). There are 89 unique statistics between the 5 categories. Our web crawler crawled over all 6 seasons and collected individual athlete data from every game for all 5 categories, forming a total of 30 datasets. We did the same for team data, which had 7 statistics categories, creating 42 more data sets.

rotoguru1 provided the fantasy salary and fantasy points scored for every athlete in every NBA game since 2014.

rotogrinders provided advanced betting data. The data includes an athlete's projected ownership percentage and points, and the betting line and over/under for the game.

fantasycruncher provided competition data for every DFS competition. The competition data includes the site hosting the competition (Fanduel), the entry cost, number of entries, prize pool, first-place score, and minimum score necessary to be "in the money". We used this data to build an optimal lineup using our the predicted fantasy points of our model.

3.2. Data Engineering

We first merged the 50+ datasets we scraped to create a single dataset per season. The data required extensive cleaning because we acquired each dataset through HTML scraping rather than an online sports database.

Trivially, we will not have access to game-stats for an athlete before a game begins, so we lagged the attributes that we would not have ahead of time, such as the athlete's box score and some of the advanced fantasy statistics. For each athlete x with a game on some date y, we calculated x's rolling average for each of these attributes over the past week, month, and entire season up until y. We did the same with team data and joined this onto our lagged athlete dataset so that each row had information about both the athlete's success and that of his team. Furthermore, we joined the lagged athlete data with the opponent's lagged data to

[1] https://www.selenium.dev/

[2] https://pypi.org/project/beautifulsoup4/

represent the impact of the opponent's strength on performance. We included these in our master dataset, along with attributes that did not require lagging.

We dropped any attribute that was missing more than 20% of its values - these were largely meta-statistics scraped from rotogrinders and were represented somewhere else in the dataset in some capacity. Hence, we weren't concerned that excluding them would negatively impact our model. We imputed the remaining missing values with the mean of their respective column because any column with missing values was numerical and inherently continuous.

We one-hot encoded the 'Position' attribute, which was the only categorical attribute on which we trained.

We engineered a few more attributes that we thought might represent some of the implicit information we know to impact an athlete's performance. For example, the importance of a game weighs heavily on an athlete's performance, but such a feature is difficult to quantify. We wrote an algorithm to represent game importance based on elements such as whether or not the game was a divisional game, the win count for each team, and the number of games left in the season. We also designed features to represent the impact of location (home or away) and the team's starting lineup on an athlete's performance.

3.3. Predicting DFS Points

We predict daily fantasy sports points in the first step of our approach. Given the continuous nature of our prediction variable, we opted for a regression technique. Modeling using a least-squares regression created non-normally distributed error terms, especially at the bottom of our predicted range, with non-constant variance in our residuals. An athlete cannot score less than 0 points, so the negative error term is bounded while the positive is not, and athletes who score a higher number of points are more susceptible to larger fluctuations, demonstrated in figure 1.

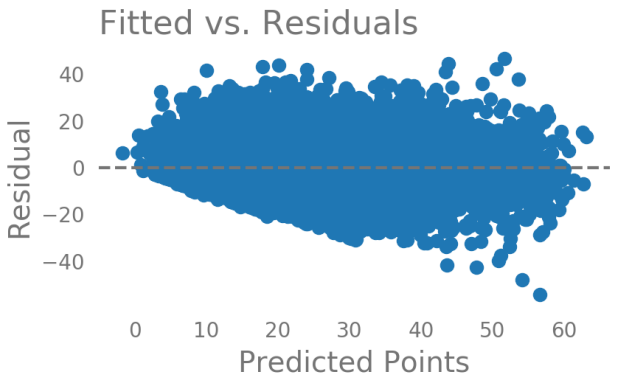

Figure 1. Linear Regression Heteroscedasticity

Furthermore, it's important to output not only a prediction for each athlete but also a prediction interval. Our ultimate goal is not to produce the highest-scoring lineup but rather a lineup with the highest probability to exceed the 50% threshold. In other words, in a pool of 100 sports bettors, having the top-scoring lineup yields the same profit as the 49th best. To reconcile our heteroscedasticity issue and provide some measurement of confidence in the output, we opted for a quantile loss function. For a given prediction $\hat{y}_i$ and outcome y_i, the loss for a quantile q can be defined as follows [3]:

$$L(\hat{y}_i, y_i) = max[q(\hat{y}_i, y_i), (q-1)(\hat{y}_i, y_i)]$$

For a set of predictions, the loss will be its average. By predicting over multiple quantiles, we can output a prediction interval along with a prediction for each athlete to our lineup builder.

We iteratively tuned, trained, and tested four ML models:

1. Linear quantile regression (statsmodels). [4]
2. Random Forest Quantile Regressor (scikit-garden). [5]
3. Gradient Boosting Regressor (scikit-ensemble)
4. Neural Network (PyTorch) [6]

For each model, we predicted 5 quantiles: $[0.16, 0.25, 0.5, 0.7, 0.86]$ which we chose in consideration of the distribution of our data. The first three libraries allowed quantile loss as a hyperparameter, but for the neural network, we wrote a custom quantile loss function. We measured each model's performance by performing a cross-validated grid search over a set of hyperparameters. This task was computationally heavy so we ported our code to Google Cloud and put our data in a Google storage bucket to distribute the computation and speed up our training and validation process. [7].

In our neural network, each output node corresponded to a quantile in $[0.16, 0.25, 0.5, 0.7, 0.86]$. We monitored its validation loss and incorporated early stopping to avoid over-fitting. We also included a drop-out rate of 0.2.

3.4. Optimizing the Lineup

To build the optimal team, we took in predictions and intervals from our model and created a linear optimizer using Pulp [8]. The linear optimizer took in the following parame-

[3] https://www.evergreeninnovations.co/blog-quantile-loss-function-for-machine-learning/
[4] https://www.statsmodels.org/
[5] https://www.scipy.org
[6] https://pytorch.org/
[7] https://cloud.google.com/ai-platform
[8] https://pypi.org/project/PuLP/

ters for each athlete: projected points, salary, position, and variance. The optimizer also took as input the salary cap of building a team, the target score, and the variance penalty.

The decision variables were binaries attached to each athlete playing that night, corresponding to either starting or not starting the athlete. The objective function of the optimizer was to maximize variance-adjusted total expected points for the team. We calculated this as the sum-product of the athletes' decision variables with their projected points from the model divided by the variance of this expected score. We raised the variance to a "variance penalty" to penalize predictions with a large prediction interval. We iterated on this to find the ideal variance penalty. The constraints of our optimizer mirrored the rules of the game:

1. The total number of athletes on the team $= 9 =$ the sum of the decision variables.
2. The sum-product of the athletes' decision variables and athletes' salaries is $\leq$ salary cap.
3. The sum-product of athletes' decision variables, with dummy values for each position, reflects a lineup with the correct number of athletes for each position.

The output of the optimizer was the "ideal" team, with 9 athletes' decision variables = 1, corresponding to those picked for the lineup, and the rest = 0.

4. Results

We compared the root mean squared quantile error for each of our ML techniques to decide which one worked best with our data, shown in Figure 2.

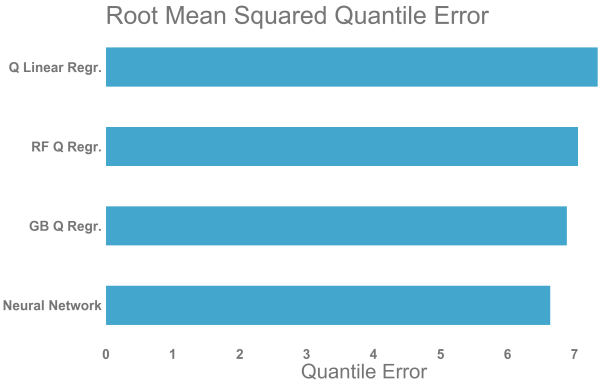

Figure 2. RMSQE of tested ML techniques

A neural network with a custom quantile loss function worked best with our data with RMSQE=6.64. We iteratively tested different neural network structures and ultimately found the following structure worked best: an input layer of size 640 (number of input features), 4 hidden layers of sizes 32, 32, 16, and 16, and an output layer of size 5, all with ReLU activation functions. Using 3 quantiles, $0.16, 0.5, 0.84$, consistently provided the lowest RMSQE through multiple cross-validated iterations of our network structure. These three quantile outputs provided a median prediction and 68% prediction interval to use as input to our lineup optimizer, shown in Figure 3

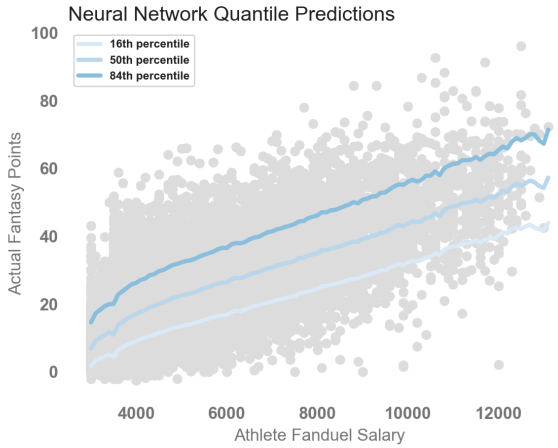

Figure 3. Neural Network Quantile Predictions

We were able to achieve a cross-validated win percentage of 64.3% using our neural network output as input to our linear optimizer. This win rate surpasses the 55.5% required for sustainable returns, giving us an expected return of \$1.57 for a \$10 bet. Our other ML models did not perform quite as well, with win rates ranging from 40% to 60%.

5. Conclusion

Our results suggest that there exists an opportunity for exploitation in introducing machine learning to daily fantasy sports. Achieving even a 1% edge over the sportsbook has proven to be a very challenging task in traditional sports gambling. We validate that the median sports gambler is not quite as informed, as our consistent win percentage of 64.3% far surpasses the required 55.5%.

Our next steps are to perform more data engineering to represent tricky latent relationships, such as the impact of a teammate's injury on an athlete's performance. We also will incorporate modern portfolio theory to influence our stake, depending on the confidence of the optimized lineup.

References

Alexandre Bucquet, Vishnu Sarukkai. The bank is open: Ai in sports gambling. Technical report, Stanford Uni-

versity, 2018.

Brent Evans, Joshua Pitts, Justin Roush. Evidence of skill and strategy in daily fantasy basketball. Technical report, Georgia College, 2018.

Getty, Daniel, Hao Li Masayuki Yano Charles Gao and Hosoi, A. E. Luck and the law: Quantifying chance in fantasy sports and other contests. Technical report, Massachusetts Institute of Technology, Cambridge, MA, 2018.

Hubáček, Ondřej, Gustav Šourek and Železný., Filip. Exploiting sports-betting market using machine learning. *International Journal of Forecasting*, 35(2):783–796, 2019.

Lisandro Kaunitz, Shenjun Zhong and Kreiner, Javier. Beating the bookies with their own numbers - and how the online sports betting market is rigged. Technical report, Research Center for Advanced Science and Technology, The University of Tokyo, Tokyo, Japan, 2017.