Beating DFS: Application of ML for Top Player Performance Selection

Jeremy M. Saslaw
University of Pennsylvania, Master's in Robotics

Leandro Grigera-Monteagudo
University of Pennsylvania, Master's in Robotics

DFS (Daily Fantasy Sports) is a category of sports gambling where users predict the top performing lineups, and those within a certain percentile, based on their selected players' performances on a given night, win money. Each DFS user in a given contest is assigned the same amount of "cash" to select 9 NBA players, of which there are 4 Guards, 4 Forwards, and 1 Center, each priced at a variable amount each night based on how the website predicts they will perform. DFS users are tasked with creating the night's optimal lineup based on who they anticipate will perform best. The goal of this project was to develop a machine learning (ML) driven algorithm capable of providing DFS Basketball users with a significant advantage by identifying a pool of over performing players for any given night. In order to accomplish this, previous NBA statistical data and Fanduel DFS data was mined and acquired with plans to find a suitable machine learning algorithm to separate and fit the data, derive a proper and accurate hypothesis, and perform cross validation on past data to determine the most optimal prediction method. While there are many companies and services that currently work to predict sports gambling and DFS results, it is a relatively new strategy to include machine learning into this process, which is precisely what the goal of this project was. The process by which this was done, as well as the resulting model and results will be presented and analyzed.

I. Nomenclature

DFS	= Daily Fantasy Sports
ML	= Machine Learning
Players	= NBA players
Users	= DFS online players

II. Problem Definition

When playing DFS games, traditional DFS users are limited to drawing correlation between the conditions on a given night and a player's performance based on a fraction of the available data. "Lebron James plays well at home," might be a hypothesis that drives a user to invest significant cash on a blue-chip player, given that the specific NBA player is playing at home on a given night. The average DFS player, however, can't remember or isn't aware of the overwhelming majority of factors that actually correlate to Lebron James's performance on a given night. Furthermore, the ability to correlate stats with over performance for a player diminishes even more when it comes to NBA players that are less well known. Thus, a primary goal of this venture was to address the issues of a traditional DFS player, allowing him/her to leverage machine learning techniques to train a classifier on player data and performance to generate a set of over performers on a given night for him/her to choose from.

Model Structure

More formally, a machine learning algorithm will be trained based on data leading up to that given day. These inputs include NBA players' individual statistics, statistics from the team that they play on, statistics from the team that they're playing against on that given day, and game situational statistics based on their specific game that day. All of these statistics are kept as a running average yearly. The statistics themselves as well as the data mining techniques used to create these statistical features will be explained in the next section, followed by an algorithm section discussing the process by which the model works.

Data Mining

Mining the data for this model was an exhaustive process where multiple possibilities were pursued to obtain the maximal data readily available to the public. Two key tools were discovered and subsequently utilized to mine the data needed for the model - NBAPY Python API and BS4 (Beautiful Soup). NBAPY is a publicly available API that serves as a method for pulling data from *stats.nba.com*. Through an extremely time-intensive process, code was generated to automatically compile both excel spreadsheets and Pandas databases of all necessary data. Statistics included:

Table 1: Statistics included in features

Statistic Type	Statistics
Individual (per game)	Minutes, Points, Rebounds, Offensive Rebounds, Assists, Steals, Blocks, Turnovers, Fouls, Field Goals (made, attempted), 3-Point Field Goals (made, attempted), Free Throws (made, attempted), Usage %, PACE, PIE, POSS, Points Off Turnovers, 2nd Chance Points, Fastbreak Points, Points in Paint
Player's Team and Opposing Team (per game)	Points, Blocks, Steals, Assists, Field Goals Attempted, Turnovers, Free Throws Attempted, Offensive Rebounds, Usage %, PACE, PIE, POSS, Offensive Rebound %, Defensive Rebound %, Points Off Turnovers, 2nd Chance Points, Fastbreak Points, Points in Paint
Game Situational	Location (Home/Away), Days off prior to game

For each year in 2015, 2016, and 2017, each player that played a game in that given year was allocated his own data frame, where each row represents the day of the NBA calendar, and each column is a running average of each of his individual statistics, a running average of his team's statistics up until that day, and a running average of the opposing team's statistics up until that day as well. If a game is played on that day, then the game situational statistics listed above are included on that row as well. The second data acquisition tool, BS4, was then utilized to acquire Fanduel DFS data from the past 3 years, which included daily player price, position, and resulting score for that day. This data was appended to the end of the row for each given player in each given year.

Algorithm

All of the statistics obtained above were converted into a ratio using the formula below, essentially normalizing the players with respect to their price for that night. This is significant because otherwise, top performing players would register as the best option and no competitive advantage would be gained.

$$\frac{Stat_i}{Price_d} \text{ for each stat i for that player's price on day d} \quad (1)$$

Furthermore, each ratio was discretized into j bins, by which binary features were obtained where:

$$feat_{i,d} = 1 \; if \; \frac{Stat_i}{Price_d} > limit_{bin_j} \mid else \; feat_{i,d} = 0 \quad (2)$$

Labels for each data point (any given player on any given day) were generated using the following formula:

$$label_{p,d} = 1 \; if \; \frac{Player's \; DFS \; Score}{Winning \; DFS \; Score} > \frac{Player's \; DFS \; price}{DFS \; budget \; (\text{"\$60,000"})} \mid else \; label_{p,d} = 0 \quad (3)$$

where *Player's DFS Score* represents that player's final score on that day, *Winning DFS Score* represents a tuned parameter for how many points it takes to win a DFS contest on a given night, *Player's DFS price* represents that player's cost on that day, and *DFS budget* represents the total cash that a user has to spend (namely $60,000). Thus, it is seen that a +/- label is assigned to a data point based on over/under performance of a player's contribution compared to their relative price for that given night.

With these features and labels obtained, the following step was to linearly separate the player pool into under/over performing players. It was determined that the SVM algorithm would be the best option, given that it provides the most robust linear separator. SGD was briefly investigated, although it was determined that SVM yielded a higher accuracy. The SVM classifier was tuned in order to find the best combination of penalty, loss function, parameter C, and tolerance. The following tuned parameters were determined:

Table 2: SVM Parameters

Penalty	L2
Loss	Squared Hinge
C Parameter	0.9
Tolerance	0.01

Expectations

By identifying a large quantity of recommended players who are predicted to overperform at a high accuracy, and filtering this set with common sense checks, the expectation was to effectively and consistently generate a DFS lineup. A common held belief derived from research is that a player's *minutes* and *points per game* relative to their price is a strong baseline method for predicting players performance. With this in mind, a baseline accuracy was determined to be around 50%, with a set size of about 15-17% of players recommended. By incrementally including more statistics, and analyzing accuracy, recommended set size, and true positives, the model was tuned to find the optimal combination of features to accurately predict labels.

III. Experimental Evaluation

Methodology

Equally important to choosing an algorithm and strategy is generating reasonable metrics for interpreting the effectiveness of the model's output. In order to quantify and understand how well the model is performing, several important metrics were generated that revealed a great deal about whether the model was tangibly improving its predictive capability through additional features, or simply stagnating due to non-useful information. Additionally, it was extremely important to know if the winning threshold established through tuning was too stringent. In other words, if the requirement was set so that the algorithm would only identify potential players that were exceedingly above average, it is possible that the set of returned players would be very small. If this were the case, no competitive advantage would be gained.

The first of the derived metrics was accuracy, which traditionally informs on what percentage of data points are being classified correctly. Accuracy allows for analysis of how effectively data is being separated, and is a great starting point for comparing different parameter configurations. Intuitively it makes sense to try to maximize this particular metric, not at the expense of player pool size.

Accuracy, however, provides no insight about the supposed set of over performing players, which ultimately are the output recommendations (solely those with label=1). In this regard, rather than knowing what percentage of data points are correctly classified as over perform/underperform, a more interesting and important metric is what percentage of predicted overperforming players are actually correct. This is the "true positive" metric, and it essentially determines the confidence with which you can choose from the output player set.

While maximizing certainty is obviously ideal, doing so may lead to a quantitatively diminished player pool. If the model provides incredibly high certainty, but outputs a player pool with less than 9 recommended players, then one can't use the model to field a complete line-up. An even more stringent requirement arises from the fact that the user needs multiple players for each of the 5 basketball positions, so having an output of 9 total players may not even be sufficient. With this in mind, a "recommendation percentage" metric was developed, which describes what percentage of the total player pool is being output by the model.

Having established multiple metrics allowing for assessment of the model's output, testing the actual utility of including particular features now becomes viable. Given that there was little initial idea as to what feature set would be optimal, (i.e. which features should be included and excluded), a SVM algorithm with a single feature was ran and the result established as a benchmark. It was found that as features were added to the training set, the ability of the algorithm to effectively identify over performing players grew logarithmically**.** This allowed for an affective gauge of whether additional features made sense or not.

A summary of the above-described metrics can be seen below:

Table 3: Metrics used to analyze performance

Metric	Representation
Accuracy	$\frac{Correct\ labels}{All\ labels}$
True Positive %	$\frac{Correct\ positive\ labels}{All\ positive\ labels}$
Recommendation %	$\frac{\#\ of\ positive\ labels}{All\ labels}$

Results

Initially, testing was performed on a single set of features to establish a baseline. 5-fold cross validation was utilized to generate a strong indicator for testing data for each of the described metrics. Incrementally, features were then added in, and it was observed how the metrics defined previously reacted to the changes. This progression can be seen below in Figure 1:

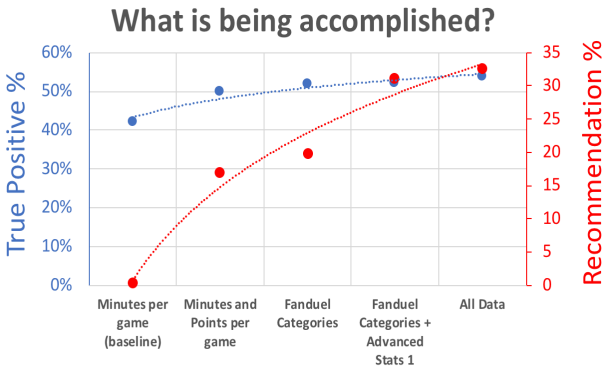

Figure 1 - Analyzing results of adding features.

As the amount of information shown to the algorithm increased, a significant improvement across metrics was observed, culminating in **62.45%** training accuracy, **61.53%** testing accuracy, a true positive value of **53.7%**, and a Recommendation percentage of **32.47%**. This is summarized below:

Table 4: Algorithm's Metrics

True Positive %	53.7%
Recommendation %	32.47%
Training Accuracy %	62.45%
Testing Accuracy %	61.53%

What is most interesting about these results is that the model was able to bolster predictive capability while increasing classification and player output percentages. This fundamentally proves that the implementation of the algorithm had meaningful impact. By comparing accuracy values between the testing and training accuracies, a negligible difference between the metrics was found, indicating that SVM was not significantly overfitting the data during training. This provided reassurance that the feature/output correlation was similar for both training and testing sets.

Discussion

As seen in Figure 1, as more of the obtained statistics and features were added to the SVM algorithm, there was a positive increasing correlation with both the true positive accuracy as well as the quantity of recommended players. Further, when adjusting the "Winning DFS Score" for label calculations, there was an evident inverse correlation between this score and the corresponding true positive percentage. Thus, this "Winning DFS Score" must be kept relatively modest to ensure a numerically significant player pool.

Further, compared to SGD, SVM linear classification provided more accurate fitting of the data. Overfitting was avoided, confirmed by the statistically insignificant difference between testing and training accuracy.

Error Analysis

While relatively high success in identifying over performers was achieved, there are multiple non-statistical considerations which could not at this time be accounted for which would have significantly improved the model. These factors include, namely, injuries and their associated effects on the rest of the team, non-disqualifying injuries, particular circumstances, team conditions, and more. For example, the model was unable to take into account when a player was injured. The reason why this is difficult to include is due to the fact that NBA teams are secretive about this data and there is a lack of accurate statistics to pull from the web other than a few minutes before a game starts. Not only does this present a (relatively common) scenario when the model predicts a player will overperform when he is not actually playing that night, but it also has residual effects. This includes the injured player's backup, who may generally not be an

optimal performer, but is suddenly allocated significantly more minutes and opportunities than he would normally. This example illustrates a particular case where a player would be an ideal play in DFS, thus creating a significant hindrance to the algorithm, which could be remedied in a later version.

The model's proposed player pool, when taking into account moderate human judgement regarding injuries and unforeseen non-statistical circumstances, is a resounding success. By providing the user with a nearly ~33% segment of the player pool, of which nearly ~54% of those players will overperform, the user can create an optimal lineup using his/her judgement.

Thus, currently, the best way to utilize the model is to analyze the output pool and post process with known user deduction. Assuming a DFS user applies outside information known about the player to navigate the theoretical output, it is possible to get better results from the algorithm.

IV. Related Work

An initial literature review returned a significant MIT Sloan paper detailing the ins and outs of DFS and all that comes with it, including the various types of games users play, the payout structures, the types of users that exist, and their opinion on how to succeed.[1]

Further research was then conducted on the sport of basketball as it relates to DFS, where an important Harvard article presented at the MIT Sloan Sports Analytics Conference detailed the importance of interior defense analytics as a source of predictive measure for sports, which was taken into account in this paper's model.[2]

Additionally, NBA Optical Tracking Data is a common resource of predicting points and valuing decisions in the NBA, which was described in another Harvard paper presented at the MIT Sports Analytics Conference. This tracking was used by these authors to track EPV (Expected Possession Value), which could be extrapolated to help predict a DFS player's success on a given night.[3]

Other related work involved investigating the use of SVM modeling with respect to sports gambling. A group at Stanford had explored this pairing with predicting the betting line in NBA games. This group found SVM to be a useful tool in their modeling.[4]

V. Future Work

In its current state, the model developed is capable of aiding DFS players by providing them with an effective tool for sourcing potential players on a given night. However, there are several areas in which significant improvements can be made to the model. The first of these involves finding a data-centric way to integrate injury management to automatically filter out potentially over performing players who are playing at a disadvantage due to injury, or are completely eliminated from playing at all on a given night. This would reflect a significant boost across all of the metrics (accuracy, true positive, and recommendation).

Another important improvement would be to explore a non-binary data input. In the development of the model, significant work was done tuning the binary case of SVM output, but more work could be done in the future on various SVM libraries that can handle non-binary labeling with weights.

Lastly, the model did not effectively deal with the many linear constraints of a lineup – primarily the fact that on a given night a player needs to field certain players at specific positions. The use of linear programming would aid in this regard, allowing for the output of the model to require significantly less post processing and manual sorting. Since the goal in the game of DFS is to field the ideal

lineup, if the model could identify the singular best or secondary option for each position, it would minimize the work necessary for the average user.

VI. Conclusion

In conclusion, it is determined that the model built provides interesting and useful predictions for a DFS Basketball user to obtain a significant advantage over the everyday player. By providing a pool of overperforming players at a high enough quantity, the user can sort through this data and make educated decisions on how to build the optimal lineup.

VII. References

1. Haugh, Martin B, and Raghab Singal. "How to Play Strategically in Fantasy Sports (and Win)." *MIT Sloan Sports Analytics Conference*, 2018.
2. Goldsberry, Kirk, et al. "The Dwight Effect: A New Ensemble of Interior Defense Analytics for the NBA." *MIT Sloan Sports Analytics Conference*, 2013.
3. Cervone, Dan, et al. "POINTWISE: Predicting Points and Valuing Decisions in Real Time with NBA Optical Tracking Data." *MIT Sloan Sports Analytics Conference*, 2014.
4. Cheng, Bryan, et al. "Predicting the Betting Line in NBA Games." *Stanford*, 2013.