Machine Learning Applications in Fantasy Basketball

Eric Hermann and Adebia Ntoso*

Stanford University, Department of Computer Science
ehermann@stanford.edu, antoso@stanford.edu

Abstract

This paper is an attempt to apply machine learning to fantasy sports in order to gain an edge over the average player. Basketball players' fantasy scores were predicted using a linear regression algorithm and stochastic gradient descent as well as a naive bayes classifier with discretized state space. A team of eight players was then selected by framing the problem as a constraint satisfaction problem. Regression optimizations were complex, but an advantage of around 8 percent was gained over regular users of DraftKings, meaning with a large enough volume of teams, the algorithm will make money. Future investigation is necessary into factoring in riskiness of players, as well as purposely pursuing players who are less frequently selected in DraftKings contests.

I. Introduction

Fantasy sports have become increasingly popular over the last 10 years, and the last two to three have seen the rise of massive fantasy sports gambling websites like DraftKings and FanDuel. These sites allow users to draft a fantasy team in their sport of choice and pay to enter that team into a pool of other teams. If a team's total number of points is higher than that of a large percentage of it's competitors, the creator of that team receives a multiple of the buy-in as payout, and if the team is in the tiny top fraction of scorers in the pool, it stands to win much more.

This space seemed like an interesting one in which to apply machine learning principles, in particular because of the predictive nature of selecting a high-scoring team, the large volume of data that is publicly available from previous seasons, and the fun of using machine learning in a relatively new and unexplored domain.

II. Problem Framework

DraftKings was selected as the website to work with (though these techniques could be applied just as easily to FanDuel or any other fantasy gambling site), and basketball as the sport of choice due to the larger volume of games, which would give more data and thus more predictive power.

In DraftKings fantasy basketball, users draft a team each day they choose to play, selecting players from any team that is playing a game that particular night. A team is composed of eight players, which are constrained to a Point Guard, a Shooting Guard, a Small Forward, a Power Forward, a Center, a general Guard (point or shooting), a general Forward (small or power), and a Util (any of the five positions). Additionally, the players have to be from at least two different real NBA teams, and represent at least two NBA games being played on that day. Each player you draft has an associated cost, usually ranging from $3,000 to $11,000 in DraftKings "dollars," and each user has $50,000 to draft their team. Players score DraftKings points by doing things like making field goals, getting rebounds, blocking shots, and a couple of other simple categories that are easy to measure from a box score.

The problem was split into two parts. First was trying to predict the number of DraftKings points a player will score based on that player's previous performance. This was framed in two ways. The first was as a regression problem, starting with a linear regression model based on box score statistics from that player in previous games. The second was as a modified multinomial naive Bayes classifier, with discretized parameters to prevent an infinite event space.

Second was choosing a team based on the predicted points for each player who has a game in

*With help from Mikaela Grace

a particular night. This part was framed as a constraint satisfaction problem. The constraints for the CSP include the spending limit of $50,000 allocated to draft a team as well as the fact that every position must be filled by a unique player. The third constraint – that each team must consist of players from at least two different teams competing in at least two different real NBA games – was not included, as the event that this constraint is not satisfied is highly unlikely. After solving for the satisfactory assignments, the optimal assignment or the team with the highest predicted score will be selected and entered into the pool.

In terms of data, box score and team data from the 2014-2015 and 2015-2016 seasons were collected from ESPN.com using python web scraping libraries like BeautifulSoup. Salary and position data were downloaded daily from the DraftKings website.

III. Score Prediction

In general, if the generated algorithm to predict players' scores is more accurate than DraftKings' predicted scores for players, the algorithm will give an advantage over DraftKings users, assuming typical users use DraftKings predicted scores as their metric. The justification for this is that DraftKings prices have a close to linear relationship with DraftKings' predicted score for a player (see Figure 1).

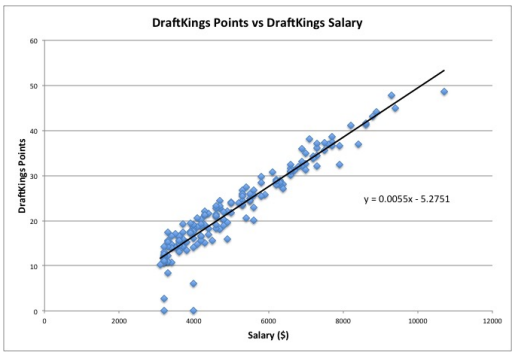

Figure 1: *DraftKings predicted points per game vs DraftKings salary for each player.*

Because of the nearly linear relationship between player price and their predicted player score, if the generated algorithm is better at predicting player scores than DraftKings is, then it will be able to find undervalued players according to their DraftKings' salaries. This is the justification for comparing our prediction algorithms outputs with the DraftKings predicted score to test actual predictive value of the algorithm.

The 2014-2015 season data was used for training and testing the score prediction algorithms, and predictions for DraftKings contests on a particular night use data from the 2015-2016 season. The means and variances for each parameter of the data from each season were nearly identical, so the theta values trained and tested in 2014-2015 are just as informative in 2015-2016.

I. Regression

The first attempt at predicting player scores from the data collected was using a regression algorithm, namely linear regression. The algorithm found the optimal theta values for the parameters using stochastic gradient descent, converging in general when the theta vector changed by less than 2.5% on a particular iteration through the data. Train and test error were calculated using the standard error estimate.

Initially, the linear regression algorithm's error on a prediction was around 150%. This obviously didn't provide much predictive value, so several specific changes were made to the algorithm to make it more accurate.

First, features were changed to be the average of that feature's values over the last five games, as opposed to just the value at the last game. Adding a feature for each of the last five games tended to decrease the accuracy, giving relatively poor predictors, likely because a feature from any game on its own is not necessarily informative about future performance. The averages over the last five games were more informative, decreasing the test error by several points (note that it's difficult to provide precise figures on error changes here because changes were not necessarily made exactly in order, and other changes to the algorithms were happening at the same time).

The data was also normalized to have zero mean and unit standard deviation, so the predictor could more accurately take into account differences in each feature.

Because the train and test error were so close together, more features were assumed to be helpful in predicting score, so the algorithm was changed to add more features. Features outside of the players' direct data turned out to be helpful, like the opponent's record as a percentage or whether the current game was at home, but beyond that adding new features didn't decrease the test error significantly.

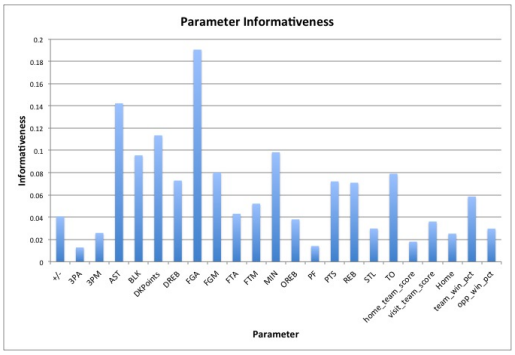

Figure 2: *Parameter informativeness for linear regression.*

Figure 2 shows the informativeness of a subset of the features we ended up using, as measured by the absolute theta values for each feature. Interestingly, parameters like field goals attempted and assists were more indicative of future performance than were features like previous points scored. Defining a quadratic feature set based on a subset of the features (58 total features), or even on a full set of the features (over 300 features), did not decrease the error significantly.

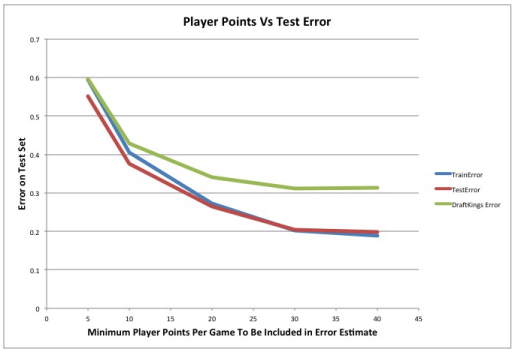

Figure 3: *Train and test error of the regression algorithm in comparison with DraftKings error, versus player points per game.*

The eventual optimal linear regression algorithm's train and test error are shown in Figure 3, in comparison with the error of DraftKings' predicted scores. When analyzing on all of the players, the regression algorithm is about as informative as is DraftKings. However, when looking only at players with a minimum average point total of 20 or higher, the regression algorithm's error is around 7.5% less than that of DraftKings, and the difference increases as the minimum average point total for players is increased (the x axis in the figure).

Since the algorithm has the end goal of choosing the best overall team, the predictions need to be accurate for players that would often be chosen for a DraftKings team. In practice, teams usually consist of players who have been averaging more than 20 DraftKings points per game, meaning that the ideal algorithm would be much more predictive for players with higher average DraftKings points. Since linear regression has much more predictive value than the DraftKings predictions at higher average points per game for players, it should work well in providing an edge when competing with other fantasy basketball players in actual competitions.

II. Naive Bayes

The naive Bayes model used is actually a modification of the typical multinomial naive Bayes algorithm to fit the data given here. Each parameter space is discretized into k equal sized groups, where k is a parameter varied to maximize the model. Then, each example in the training set is converted into values from 0 to k-1 for each parameter. All examples with the same set of values have their actual DraftKings score averaged to provide a predicted output for that set.

The test set is then converted in the same way, and each test example takes on the predicted output for its set of values. This model ignored test examples whose converted parameter values did not have a match in the train examples.

By intelligently varying k, the number of quantiles, and the number of variables, the naive Bayes model's accuracy can be improved. Table 1 shows a table of values for test errors given different k and variable numbers. After 5 variables and 5 quantiles, decreasing either variable further doesn't sig-

nificantly improve the error.

Num. Variables	Quantiles	Test Error
25	10	1.0
25	5	1.0
18	3	0.495
18	2	0.470
11	3	0.428
5	10	0.419
5	5	0.403

Table 1: *Naive Bayes accuracy as the number of discretized states and variables are varied.*

This technique gets relatively close to the error rates seen in linear regression. However, looking at the error as the player set is limited to players who score more points on average shows that linear regression has a natural advantage with better players (the ones that the algorithm needs to accurately predict). Figure 4 shows the comparison. In general, Naive Bayes tends to perform about as well as the DraftKings predictions, while linear regression has a significant advantage for better players.

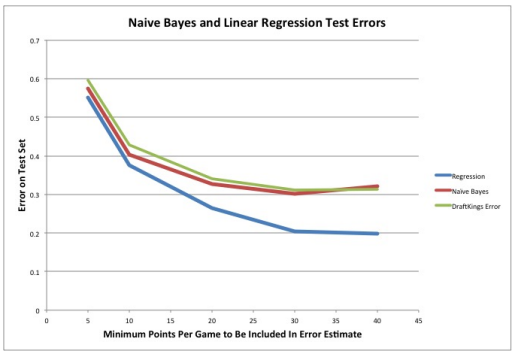

Figure 4: *Naive Bayes and regression error rates in comparison with that of DraftKings, versus average player score.*

IV. Constraint Satisfaction Problem

After constructing an algorithm to predict the score for each player, the constraint satisfaction problem solved for the team that would produce the highest total score. The CSP consists of a set of variables and two sets of factors to account for each constraint. The variables are each of the eight positions to which a player must be assigned. The first set of factors is a set of binary factors between each permutation of position pairs to assure that every position has been assigned a unique player. The second variable is an n-ary variable which limits the total sum of the player costs to be at most \$50,000. Figure 5 shows the detailed factor graph of the CSP.

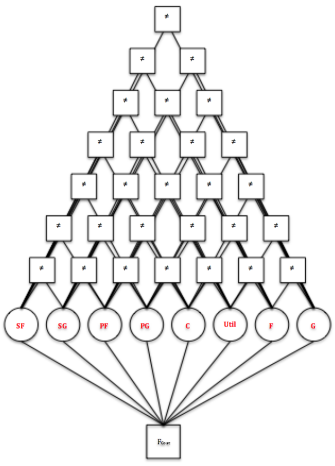

Figure 5: *Factor graph of the CSP*

V. Backtracking and Beam Search

Even when assigning most constrained variables first and enforcing arc consistency, the backtracking search algorithm took several minutes to find the optimal assignment from a pool of 40 players. The backtracking search algorithm finds every possible satisfactory assignment for the positions and thus the runtime increases on the order of $O(n^8)$. Given that there are more that 240 players to choose from each night, this algorithm would be too long to find the optimal assignment.

Implementing beam search instead limited the search space to assignments with only the k highest scores. This k parameter was varied to determine how large the search space must be in order to find a high scoring assignment. For the CSP, this parameter was limited to 32 since higher values produced the same optimal assignment. Although beam search isn't guaranteed to find the optimal solution, it finds a high scoring assignment relatively quickly as the runtime increases on the order or $O(n(kb)\log(kb))$

where b is the branching factor. Figure 6 shows a list of the top three assignments for the games played on December 8^{th}, 2015 after running beam search.

Players	Richard Jefferson Marc Gasol Draymond Green Randy Foye Rodney Stuckey Gordon Hayward Victor Oladipo LeBron James	Victor Oladipo Marc Gasol Draymond Green Randy Foye Ben McLemore Joe Johnson Alec Burks LeBron James	Trevor Booker Marc Gasol Draymond Green Randy Foye Rodney Stuckey Omri Casspi Victor Oladipo LeBron James
Score	280.486	272.153	271.040
Cost	$50,000	$49,400	$49,400

Figure 6: *Optimal teams for the games played on Dec. 8^{th}*

VI. Future Work

The following techniques have not yet been implemented, mostly because the data to concretely test them is not publicly available so it would be hard to gauge their effectiveness without investing a large sum of money into the algorithm to see how they actually play out in DraftKings contests.

Drafting the best players will generally result in a good team, but the real money in DraftKings is paid to a tiny fraction of the players in each contest. In a typical contest, about half of the prize money goes to the top ten performing teams, and a contest will have around 100,000 entrants on average. To win, the team chosen needs to be outstanding in comparison to others.

To that end, picking players with higher "risk," where risk is defined as the ratio of a players' standard deviation to their mean, is a strategy more likely to win the largest prizes. This is because riskier players are more likely to do exceedingly well on a particular night (and also to do exceedingly poorly, but the idea would be to draft a huge number of teams and make money back when only one or two of them are successful). Figure 7 shows a graph of riskiness ratio, a players' riskiness divided by the average riskiness, versus that players' average number of points per game.

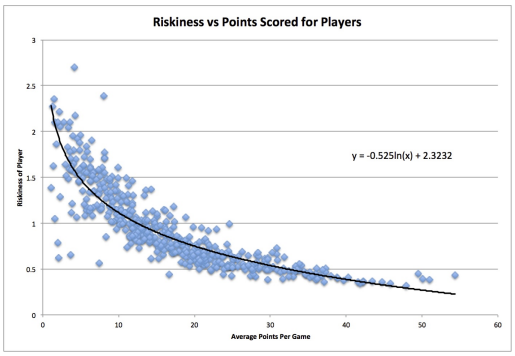

Figure 7: *Player riskiness as compared with average points per game.*

This ratio is clearly skewed to players with lower average points per game, and would need to be adjusted so all values are closer to 1. Incorporating riskiness is a necessary and important future step to optimizing a DraftKings prediction algorithm.

Additionally, winning at DraftKings is not necessarily just about picking the best predicted team on a given night. If the goal is to beat competitors, it's not advantageous to draft players that just about all of the competitors have also chosen, because that player provides the user's team with no comparative advantage. Choosing less frequently chosen players is a better strategy because the uniqueness of the team goes up, and a more unique team that does well is more likely to be separate from its competitors and to score in the top 10. Data for percentage of teams that drafted a particular player is not publicly available, but future work could explore proxies that represent that data well.

References

[ESPN] "ESPN." Fantasy Basketball. Web. 27 Oct. 2015. `http://games.espn.go.com/frontpage/basketball`.

[DraftKings] "DraftKings | Daily Fantasy Sports For Cash." DraftKings. Web. 27 Oct - 09 Dec. 2015. `https://www.draftkings.com/`.

[CS 221] Liang, Perry. "Lecture 11: CSPs II." Artificial Intelligence. Stanford University. NVIDIA Auditorium, Stanford. 02 Nov. 2015. Lecture.