Learning to Turn Fantasy Basketball Into Real Money
Introduction to Machine Learning

Abstract

Fantasy basketball is a rapidly growing multi-billion-dollar industry with colossal yet largely untapped potential for data mining. In fantasy basketball, fantasy owners aim to win money by picking the top statistically-performing NBA players on a weekly basis. In each game, every fantasy NBA player earns a certain number of points based on the actions of the corresponding real-life NBA player. With this in mind, we develop a machine learning (ML) model that predicts the statistical performance of NBA players in a given game. We perform feature selection from a wide array of both basic and derived statistics for individual players and opposing defenses. Linear regression, random forest regression, and support vector regression are compared and both feature and model parameters are searched to find the most accurate model for predicting future game fantasy scores. Additionally, we extend this model to predict the outcome of a given game by aggregating the individual fantasy scores of the players on each team in the game.

1. Introduction

In standard head-to-head fantasy basketball scoring, players earn fantasy points based on in-game actions and statistics. Table 1 shows a common fantasy scoring formula which is used for our study.

A fantasy players goal is to score the maximum number of points with his or her given roster every game from a selected subset of players typically drafted at the beginning of the season. Typically there are more players to choose from than there are available positions on the roster, so the ability to predict which players will perform the best in an upcoming game is extremely vital information. And to evaluate a players performance, in comparison to the rest of his peers and opponents, we use a metric of evaluation called Rank Difference Error, henceforth referred to as RDE. Rank Difference Error is the absolute difference between the actual rank and the predicted rank of a player in a game, playing for a specific team. Additionally, we leverage our fantasy score prediction model to present a novel method for game win-loss outcome prediction. We achieve this by aggregating the individual fantasy scores of the players on each team in the game and comparing the first teams total fantasy score to the second teams total fantasy score.

Table 1. Scoring settings based on Yahoo! Fantasy Basketball express settings

SCORING SETTINGS	COEFFICIENT
FIELD GOALS ATTEMPTED	-0.45 POINTS
FIELD GOALS MADE	1.0 POINTS
FREE THROWS ATTEMPTED	-0.75 POINTS
FREE THROWS MADE	1.0 POINTS
3-POINT SHOTS MADE	3.0 POINTS
POINTS SCORED	0.5 POINTS
TOTAL REBOUNDS	1.5 POINTS
ASSISTS	1.5 POINTS
STEALS	2.0 POINTS
BLOCKED SHOTS	3.0 POINTS
TURNOVERS	-2.0 POINTS

2. Related Work

Most previous literature on ML in the NBA focuses on predicting the outcomes of NBA games (i.e., which team wins). State-of-the-art linear and logistic regression models correctly predict the outcome of about 68% of games, while more complicated models such as artificial neural networks (ANNs) and deep belief networks (DBNs) have yielded similar success. However, there has been significantly less research related to predicting individual player stats, particularly as applied to fantasy basketball.

University of Pennsylvania, CIS 419/519 Course Project.

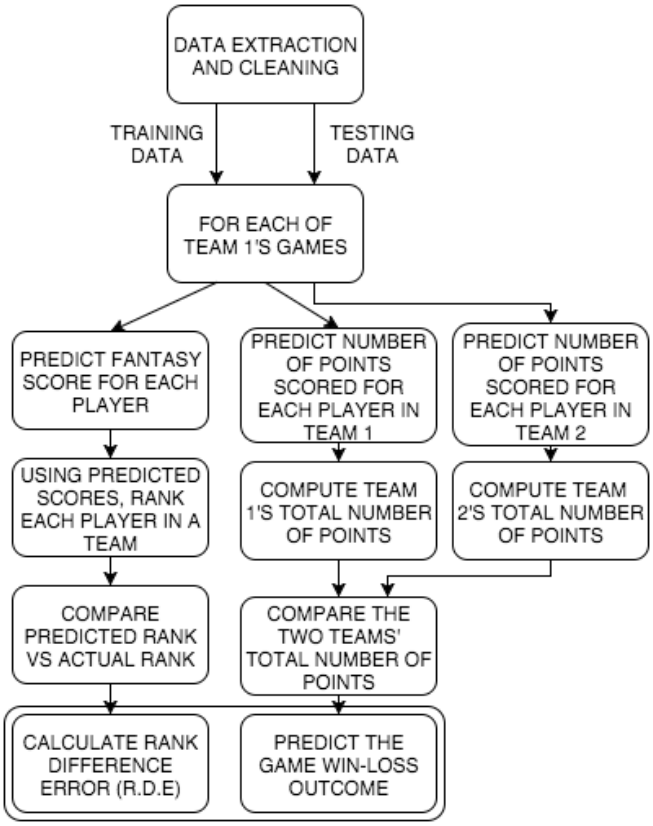

Figure 1. High Level System Design

3. Methodology

In this section, we provide an outline of the regression framework beginning with data collection and cleaning. Second, we outline the individual player model and parameter-tuning methodology used for aggregation of statistics at the team level. Third, we present the player-distance heuristic used to rank players in a start-or-sit fashion. Finally, we propose an aggregation model that leverages previously derived fantasy scores to predict individual game outcomes.

3.1. Data Extraction and Cleaning

The 2015 NBA season franchise consists of 6 divisions with 5 teams each, each of which play 82 games. Each team has anywhere from 15-25 players on its roster resulting in a significant number of game-player predictions necessary to make an exhaustive predictive model on the entire season. To limit the the computational complexity of the model, we have arbitrarily chosen the Memphis Grizzlies as the template team and focus on games 65-74 of the regular season. This limits our computation to the 11 teams (Memphis Grizzlies and 10 opponents) and 10 games shown in Table 2.

Table 2. Test set for games 65-74 of the 2014-2015 Memphis Grizzlies season

G	DATE		OPP	W/L
65	3/12/15	@	WAS	L
66	3/14/15		MIL	W
67	3/16/15		DEN	W
68	3/17/15	@	DET	L
69	3/20/15	@	DAL	W
70	3/21/15		POR	W
71	3/23/15	@	NYK	W
72	3/25/15		CLE	L
73	3/27/15		GSW	L
74	3/29/15	@	SAS	L

Although each team plays 82 games in a regular season, a player may play fewer than 82 games for a variety of reasons such as injury or not being put on the court by the coach. Sometimes, players do not play for the the entirety of the game and so no statistics are recorded for that game. In order to account for the probability of not playing a game, we consider two separate scenarios for game inactivity. (1) a player does not play a few game at various times throughout the season, typically to rest a player or for unique circumstances (2) a player does not play for a large continuous period of time, ranging for reasons such as being traded, injury, simply not putting up enough points to play as a starter most games. In the case of (1), we do not want the missed game to negatively affect the model so we remove the game from the dataset if the player has played at least 10 minutes on the court averaged over the past 5 games. Likewise, if a player does not meet this 10 minute, 5 game criteria, then we want to account for the low likelihood of playing an upcoming game by zeroing out the fields of that game statistics.

3.2. Individual Player Model and Parameter Tuning

The individual player model is predicated on the hypothesis that a players statistical performance against a particular team correlates strongly with: (1) the players statistical performance in recent games; (2) the statistical performance of the opposing teams defense in recent games. Under this assumption, our feature space is composed of features that represent: (1) the players statistical performance in the previous game; (2) the players average statistical performance across the past N games; (3) the opposing team defenses average statistical performance across the past N games; (4) the opposing team defenses average statistical performance across all of the games played in the season thus far. The value for N is a tuned parameter that varies from player to player

and was found by performing an exhaustive search over a range of moving-average values from 2 through 25. To quantify player performance in each game, our features consist of common traditional statistics and advanced statistics. In addition, we perform PCA in order to reduce the dimensionality of the data. The number of components chosen for a given player is found through an exhaustive search through PCA analysis of 1 to 92 dimensions.
For each combination of game and player, we produce an instance consisting of a 92-element feature vector and a real target value representing the total fantasy points earned by the player in the given game. We built a M-game training set corresponding to the number of remaining games after setting aside a 15-game test set. Games are then averaged based on the N parameter and potentially removed according to the methods described above.

3.2.1. Optimal Moving Window and PCA

Initially, we set out to search over all player models to find an optimum moving-average and number of principal components that would be a two sizes fit all, but to be true to our intuition that each players statistical performance in a given game is unique, searching over the entire space of moving-average possibilities and number of principal components to find the combination of the two parameters that gave best R2 scores over the training data was a better idea. And as you can see in Table 3, this decreased our RDE by a significant amount.

Four different regression models were tested. We confined ourselves to the readily available packages in the Sklearn Library, specifically testing Linear Regression, Ridge Regression, Random Forest Regression and Support Vector Regression.

We first narrowed down our list of algorithms by an initial test over our testing data. Our best performing algorithm straight off the shelf was Support Vector Regression because given the nature of SVRs design, we were able to get reasonably good values without needing to implement a prior step of PCA reduction. The second best algorithm was Random Forest Regression. The results for each of these algorithms is shown in Table 3.

3.2.2. Parameter Tuning and Final Results

The SVR model with a rbf kernel performed the best among the four models, with a C value of $1 * 10^6$ and a γ value of $1 * 10^{-5}$. We arrived at these optimal parameters for SVR using Sklearns GridsearchCV to test over a wide range of C and Gamma parameters. The final results and comparison are illustrated in Table 3.

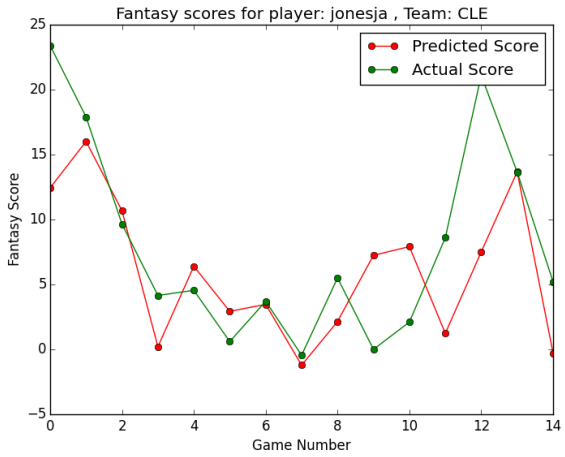

Figure 2. Fantasy scores for Cleveland Cavaliers player James Jones

3.3. Player-Distance Heuristic

The evaluation metric used here is RDE. We decided to use this metric rather than measuring the difference between the actual fantasy score and the predicted fantasy score because the absolute value of the predicted fantasy score was slightly noisy, but our model did however capture the players performance trends quite well. An example is shown in Figure 2.

As you can see, while the predicted scores may not be very similar, the overall trend has been predicted. We decided to use this intuition as a foundation to building the Rank Difference Error heuristic.

To predict each players performance on the games played by MEM, i.e each instance in the test set, we train our model on the players past performance, averaged using a specific moving window size, and in the case of Random Forest Regression and Linear Regression, a specific number of PCA components. These scores are then used to rank players in a team based on their fantasy score. Players with higher fantasy scores receive higher ranks and players with lower fantasy scores receive lower ranks. Additionally, a set of ranks is created using the actual score data. We then compute the RDE for each player in a given game by finding the absolute difference between the predicted score and the actual score. We then calculate the mean RDE over all player, in all teams through the set of 15 games. Our best value of RDE using Support Vector Regression was roughly 3.20, which essentially means that

Table 3. Algorithm Comparison with RDE score metric

ALGORITHM	RDE W/O TUNING SINGLE MODEL	RDE W/ TUNING SINGLE MODEL	RDE W/O TUNING MULTI-MODEL	RDE W/ TUNING MULTI-MODEL
SUPPORT VECTOR REGRESSION	4.04203	3.64597	3.87722	3.20426
LINEAR REGRESSION	4.14318		3.63612	
RANDOM FOREST REGRESSOR	3.77307	3.64302	3.56026	3.45550

our prediction of a players rank is approximate to 3.20 rank positions in a given game.

3.4. Game Outcome Prediction

A natural extension of predicting individual player performance in a particular game is predicting the outcome of that game. Given two teams, Team 1 and Team 2, we use the procedure described above to predict the number of fantasy points scored by each of the active players on Team 1 and by each of the active players on Team 2. For the learning component, we use a moving average of seven games and support vector regression with radial basis function kernel, $C = 10^6$, and γ = 1E-05. Next, we take the sum of the predicted fantasy points scored by Team 1s players and compare it to the sum of the predicted fantasy points scored by Team 2s players. Our game outcome prediction model then predicts that the team with the higher predicted fantasy points total will win the game matchup.

In this application, the threshold for precision in predicting each teams total points is not as high as for the player ranking application described in the previous sections. Due to the lower complexity of this binary classification problem and the fact that prediction error for a given player is smoothed out when we aggregate the total prediction error for the corresponding team, we are allowed more room for error in our predictions.

As in the previous sections, we evaluated our model on games 65 through 74 for the Memphis Grizzlies during the 2014-2015 regular season. Team 1 denotes the Memphis Grizzlies (MEM), while Team 2 denotes MEMs opponent. The results of this experiment are shown in Table 4.

As shown in Table 4, our model using points as the target value predicted the results of games 65 to 74 with 70% accuracy, performing slightly better than our model using fantasy points as the target value.

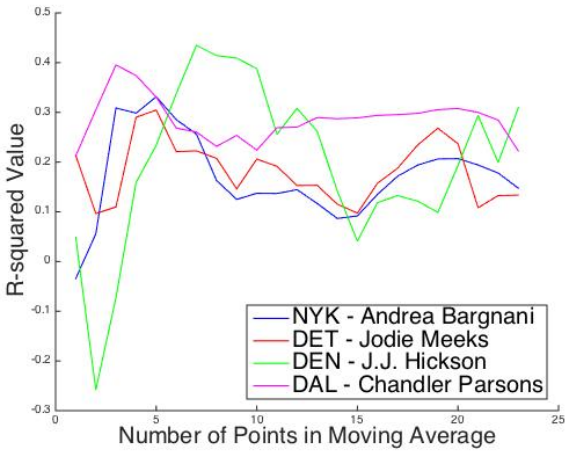

Figure 3. Sampling of player R2 values across varying moving average parameter

4. Performance Evaluation

Figure 4 shows the results of sweeping the moving average value for the SVR-rbf kernel from 2 through 25. Although a unique value is derived for each player to best fit the model, figure 1 shows that best R-squared performance and thus most correlated games tends to lie in the range of the past 3-7 games.

Similarly, Figure 5 shows the PCA analysis that was performed for each player to best fit the number of components to transform the data. As with the moving average, a unique value was optimized for each player but values tend to range from 5-15.

5. Conclusion

Overall, our learning models performance results in player fantasy score prediction and game outcome prediction are very encouraging. With an overall RDE score of approximately 3.2, our player fantasy score prediction/ranking system could certainly benefit fantasy basketball owners who need help setting their lineup for a certain game. Looking at our performance results, the model might not be strong enough that an owner could blindly rely

Table 4. Game outcome prediction results using points as the target value

GAME	65	66	67	68	69
TEAM 2	WAS	MIL	DEN	DET	DAL
TEAM 1 TOTAL POINTS	52.29244	148.04757	144.25861	154.25470	164.84836
TEAM 2 TOTAL POINTS	149.82268	194.17713	143.84321	243.45083	252.68286
PREDICTED TEAM 1 OUTCOME	L	L	W	L	L
ACTUAL TEAM 1 OUTCOME	L	W	W	L	W
GAME	70	71	72	73	74
TEAM 2	POR	NYK	CLE	GSW	SAS
TEAM 1 TOTAL POINTS	167.19978	120.18366	148.22923	150.33212	141.24276
TEAM 2 TOTAL POINTS	169.37400	116.49133	217.98367	157.18418	147.08796
PREDICTED TEAM 1 OUTCOME	L	W	L	L	L
ACTUAL TEAM 1 OUTCOME	W	W	L	L	L

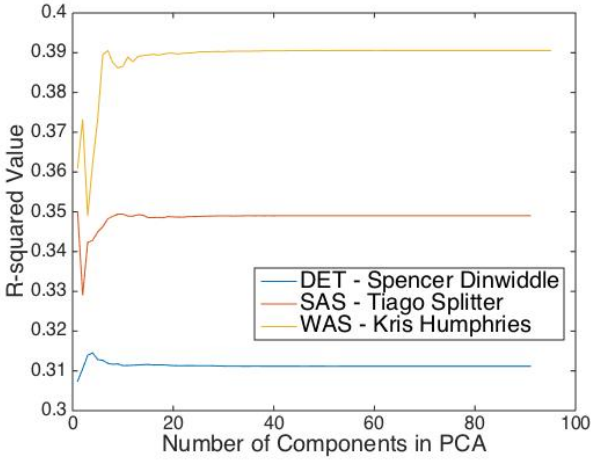

Figure 4. Sampling of player R2 values across varying components in PCA

on it in setting his or her lineup, but it provides some non-trivial insight about how certain players are likely to perform against a given opponent in the context of a certain game/season situation. Particularly, in situations where the fantasy owner is having trouble deciding which of two similarly-performing players to start, our fantasy score prediction/ranking system would be quite helpful. Furthermore, for those who bet on the outcomes of NBA games, our game outcome prediction system could be a useful tool. In our limited experiment, our model achieved 70% accuracy. This is on par with the state-of-the-art NBA game outcome prediction models, which predict game outcomes with 68% accuracy (note that the state-of-the-art models have been tested on more games).

However, there are a number of possible improvements to our learning model. First, our performance might improve with an increase in the size of our training set. Our model uses 15 training games for each prediction. The model might have benefited from using more training games per prediction, but, because it is better to train the model using only games from the current season, there is a tradeoff between number of training games and number of games available to make predictions on. Further experimentation with different numbers of training games could give us more insight about how to optimize our performance with respect to this tradeoff. Second, our model has a rich feature space (92 basic and advanced metrics), but may include too many features that are not strongly correlated with players fantasy points. This could be remedied by using more sophisticated feature selection techniques (e.g., greedy hill climbing, targeted projection pursuit) and by introducing new and potentially more expressive features (e.g., team chemistry statistics, shooting tendencies statistics, SportVU tracking statistics). Third, to further validate our model, we need to acquire more player game log data (both from the current season and past seasons) so that we can test our model on more than just ten games. Data acquisition was a significant obstacle for us because, for each game, we need to scrape and clean basic and advanced stats data for all of the players on both teams in the game, and, even using an automated process, doing so for each player took a considerable amount of time. With more data, we could perform k-fold cross validation, which would in turn help us determine the best learning parameters and feature set.

References

[1] Web. 7 Dec. 2015. ⟨*http://www.basketball-reference.com*⟩

[2] Sklearn Supervised Learning Documentation Web. 7 Dec. 2015 ⟨*http://scikit-learn.org/stable/supervised_learning.html*⟩

[3] B. Ulmer and M. Fernandez *Predicting Soccer Match Results in the English Premier League*