Predicting Player Performance in the NBA

By Johnny McGregor

Project Goal

The NBA has embraced the use of data to evaluate players over the last decade, and the practice is become ever more popular. New advanced metrics beyond just the traditional statistics are used to evaluate players. While these advanced metrics have proven to be useful in predicting how valuable a player can be to a team over the course of a season, or a multi-year contract, I want to see if I can use the traditional and advanced statistics together to see how well a player might perform on a given night. Furthermore, I want to see if combining those advanced player statistics in a machine learning model with the advanced statistics available on his opponent will improve the predictions.

To quantify a player's performance, I am using a fantasy score determined by the daily fantasy sports site DraftKings. This score is compiled as follows:

- Each point scored is worth 1 fantasy point
- Each three-point basket made is worth an additional .5 fantasy points
- Each rebound is worth 1.25 fantasy points
- Each assist is worth 1.5 fantasy points
- Each steal is worth 2 fantasy points
- Each block is worth 2 fantasy points
- Each turnover deducts .5 fantasy points from their score
- If a player has a double-digit amount of two of the above stats (i.e. more than 10 points and more 10 rebounds) it is known as a double-double and worth an extra 1.5 fantasy points
- If a player has a double-digit amount in three above stats it is known as a triple-double and worth an extra 3 fantasy points.

There is a wrinkle in tackling this problem. When putting together a "team" of players on Draft Kings, you have to pick at least one player at each of the five different positions, plus an additional three players that can be of any position you choose. A player is assigned a dollar value between $3,000 and around

$10,500. You have a total of $50,000 dollars you can spend on a total of 8 players (an average of $6,250 a player). The task at hand isn't just trying to get an estimated fantasy output for a particular NBA player, but also finding which players may be a good value for a given game.

Root mean squared error will be the metric by which we evaluate the predictions of all regression models. If the model predicts relatively well, this could be a way to "game the system" of sorts to win money on Draft Kings.

Gathering Data

Basketball-Reference.com is a website that provides statistics on every player that has participated in at least one game. I used the python library Beautiful Soup to web scrape data from the current season as well as the five previous seasons dating back to the fall of 2014. This initial set of data totaled over 117,000 rows and had statistics on over 879 unique players. I also performed web scraping to gather the advanced statistics on every NBA team going back to the beginning of the 2104-2015 season. There are 30 teams in the NBA and each plays 82 games in a season, so each full season is a dataframe containing 2460 rows. The player and/or team features available to train a model include the statistics already mentioned above as well as the following:

- Field goals attempted (how many shots a player or team takes)
- Field goals (how many shots a player or team makes)
- Field goal percentage (% of field goals attempted that are made)
- 3-point field goals attempted
- 3-point field goal percentage
- Free throws attempted/made/percentage
- Minutes played
- Whether or not a player started a game (marked as 1 or 0)
- Whether a player or team is playing at home or away
- Plus-minus (the difference in score while a player is on the court)
- True shooting percentage (a measure of shooting efficiency that takes into account 2-point field goals, 3-point field goals, and free throws)

- Effective field goal percentage (adjusts for the fact that a 3-point field goal is worth 1 more point than a 2-point field goal)
- Total rebound percentage (an estimate of the percentage of available rebounds grabbed by a player or team)
- Total assist percentage
- Total steal percentage
- Total block percentage
- Total turnover percentage
- Usage percentage (an estimate of the percentage of teams plays that a player was used while he was on the floor)
- Offensive rating (points produced per 100 possessions
- Defensive rating (points allowed per 100 possessions)

Exploratory Data Analysis / Preprocessing Data

The advantage of web scraping from such a well-organized site is the absence of missing values. I wrote a line in the web scrape code not to include a game in the data if a player did not play. Therefore, there weren't any null values to search, nor were there any outliers due to incorrectly entered data.

There was a fair amount of cleaning to be done in other aspects, however. Each value from the web scrape was returned as an object. There were a handful of columns that had a duplicate labels and others that had "/" or "%" signs in them. I also needed to create and add columns to the data for double doubles and triple doubles, as well as the eventual target (fantasy points). In order to essentially automate the process of collecting the data from the hundreds of individual players' web pages I had to employ some rather lengthy functions to accomplish the following:

- Gather player data from the regular and advanced statistics pages and concatenate them together.

- Clean the concatenated player data, transform some column names, and add necessary columns mentioned above.
- Do the same thing for each team in the league's regular and advanced statistics
- Index all the data by date so that the row of data for a particular game could be joined with the team they were playing against that night.

To make predictions on future performances, the data that will be used as features will be some sort of aggregation of all the above stats up to the moment before the game. Since the data I initially collected was all single game data that could not be used to train a model, I had to create the features to be used by calculating the recursive averages. For example - if on November 1 Player X scored 8 points, had 10 rebounds, 2 assists, 2 turnovers for a total of 24.5 fantasy points, I cannot use those numbers to train the model. While predicting the fantasy point output for a future game we won't know how many points, rebounds, etc the player will get so instead we will train the model on what the player has averaged in all of those categories un until the day of the game in which they produced 24.5 fantasy points. Same goes for the opponent's statistics that may be used in the model training. Therefore, I wrote a function to make each row of data the average of all the features up to but not including that day's game. I also did this with a rolling five game average using the pandas rolling function, since each season is long and how a player performed earlier in the season probably doesn't have as big an impact on how he will perform three months later. Finally, a shift() had to be performed on the data as well so that each row was some sort of aggregation up to the game before the current one.

Modeling

During the course of modeling, it was difficult to get a good performance from any of the models I tried. These models include linear regression, random forest, ada boost, gradient boost, and neural networks with keras. The baseline root mean squared error (just using the average fantasy points for every player in every game) was 14.62. The three best performing models (take that description with a grain of salt) - random forest, gradient boosted regressor, and linear regression has root mean squared errors of 9.68, 9.61, and 9.58, respectively. When taking the mean of all three models' predictions, I got a root mean squared error of roughly 9.56. All of the R^2 scores on the test data was right around .57 for each model. The score on the training data was about the same for linear regression and gradient boost, with the random forest overfitting in a big way with a .939 R^2 score on the training data.

At this point, I decided I would see if classification models would fare any better. Instead of trying to predict a specific score, I assigned four levels to the possible fantasy score values. Guided somewhat by running summary statistics on the target variable (with a pandas describe( ) method), I decided to make any fantasy point total of 50 or more tier 1. Tier 2 is between 30 and 49 points. Tier 3 is 20-29 points and tier 4 is under 20.

The baseline accuracy score to test the models against was .475 (if the model predicted all tier 4, it would be 47.5% accurate). Using random forest classifier, gradient boosted classifier, and logistic regression – again no model jumped out ahead of the rest. Each was right around 60% accuracy on the test data. The random forest predicted the training data with 100% accuracy, while the other two models were at 64.8% (gradient) and 62.8% (logistic). For what it's worth, the random forest and gradient boosted models classified almost 100 more tier 1

performances correctly compared to logistic regression, but they also had a lot more falsely predicted tier 1 games.

Recommendations

For one to seriously attempt to use data to make money playing daily fantasy sports, they will have to do so using a combination of data analysis and human intuition. Incorporating the daily player salary information over time would be helpful too in order to be able to identify players who outperform their value more often. Natural language processing might be something to add to a complex tool to help predict when a bench player is starting due to an injury of one of their teammates. All in all, this is not an easy thing to predict. It will take a lot of work and diligence. If it were an easy thing to predict, everybody would be doing it!