Nets on Nets: Using Shot Tracking Data and Machine Learning to Predict NBA Game Outcomes

Keyur Mehta
keyur@stanford.edu

Simar Mangat
smangat@stanford.edu

Maneesh Apte
mapte05@stanford.edu

Abstract - This project focuses on using machine learning to both predict NBA game winners and to determine the margin of victory for the betting line. This project builds on existing systems by incorporating both macro and micro game features, the latter being less explored by prediction models in previous research. The '14-'15 season data constitutes the data set for which a variable number of games into the season are used for training. The remaining games in the season serve as a test set. Max classification accuracy was achieved using an SVM classifier with 72% accuracy with training on 67 games. Additionally, the betting model correctly won the 78% of its bets after training on 52 games.

I. Introduction

Over the last few years, there have been a number of systems created with the intention of predicting NBA games. These systems have been extremely diverse, ranging from using historical comparisons to predict player performance to an "Elo" predictor that uses the Chess-derived Elo ratings to predict games[1]. The majority of these systems have used macro-game indicators as their features, including statistics like Home Team Points Scored per Game or, in the case of Elo, performance against Elo-rated teams, to make predictions on how a specific game will turn out. In the 2014-2015 season, the NBA started using the SportsVu system which produces intricate and robust advanced statistics for every player in every game using computer vision. Opening access to this data has allowed the world to gain access to micro-game stats that incorporates court-location, time on the shot clock, and distance between players. These novel defensive and offensive insights form the basis for this project.

A. Task Definition

Our model seeks to leverage this new breed of statistics to better predict which NBA team will win and by how much. Micro-game indicators of each team are used as features with the hopes that understanding how teams stack up in part, as opposed to in whole, will better inform a prediction of how the game will go. For example, instead of statistics understanding how good a team's offense is compared to its opponent's defense, the micro-game indicators could tell us how well a team shot a specific type or shot or their success in running a type of play alongside how the other teams defends it. Our hope was that the micro-features would be an extremely rich feature set so that, unlike other models, we could only use games within a current season as the training data. Although most models use previous seasons to train their weights, we were motivated to stick to a single season because we hypothesized that the off-season movement between NBA teams would introduce significant noise into the model.

At a high level, the input for our system is the team's offensive and defensive metrics calculated as a running average over the season prior to the game we are predicting. The output, predicted from these features, would be either the win/loss classification or a score differential on how the game would go.

[1] https://projects.fivethirtyeight.com/2017-nba-predictions/

To evaluate the success of the model, we compared our predictions to one measure of ground truth as well as the one of the current best-in-class predictors. To accomplish our overarching goal of predicting the outcomes of the games themselves, we compared our Win/Loss classification to the true result of the game. Additionally, in order to see how we performed relative to other predictors, we used our predicted score differential to bet on the spread of NBA games. By betting on the side of the spread that our differential sat, we hoped to see our performance relative to the average better.

B. Literature Review

There have been a few projects in the past that have explored using machine learning techniques on predicting the outcomes of NBA games. For example, a report done at University of Wisconsin - Madison similarly tries to predict NBA games using machine learning algorithms[2]. Their approach is the starting point for our project; our model is complementary as it incorporates both the macro features that the paper outlines (win-loss percentage, points scored, etc) and micro features (3-pointers scored while guarded, etc.). Furthermore, the Madison report indicated the usage of multiple machine learning algorithms, including SVMs, Naive Bayes, and Multilayer Perceptron. Our project utilizes these algorithms as well. Another report that influenced our project was done at Stanford University as a CS229 project[3]. This report focused on both predicting raw win-loss as well as win-loss as it pertains to the betting line. This gave our project the impetus to similarly incorporate betting-line success as one of our evaluation metrics as well as to see how the micro-feature model performed relative to that used by the students in the CS229 project. Furthermore, we learned about nuances of algorithm techniques from their report, including the use of SVMs in 2 conjunction with a lasso regression and bootstrap aggregating. These reports served as inspiration for our project, and we sought to expand on them.

II. Dataset/Infrastructure

The macro game indicators (i.e. traditional statistics like points, assists, rebounds, turnovers per game) and micro game indicators (i.e. shot tracking data packaged into statistics) were both scraped from stats.nba.com for the 2014-2015 season. In total, 21 macro indicators and 28 micro categories with 13 metrics were pulled for all 1230 games in the season.

Betting line data was scraped from covers.com by first pinging each day of games during the season and then running appropriate regexes to extract the betting line information.

III. Approach

In the case of classification, our baseline involves always choosing the home team, often the favored team in professional sports, to win each game it played. If employed in the 2014-2015 season, this baseline would have 53.7% accuracy. The oracle, on the other hand, can be estimated with the Accuscore algorithm for NBA (win/loss) bets. We chose this algorithm as our oracle because Accuscore is a best-in-class betting prediction company with the strongest track record of success. For the 2014-2015 season, their model had an accuracy of 70.3%.

A. Initial Challenges

Regarding the scraping of betting line data, it was challenging to first find a website that delivers this information in a predictable way and furthermore was moderately challenging to parse this data. Betting sites intentionally make this information tough to acquire in an attempt to preempt projects like this.

---

[2] http://homepages.cae.wisc.edu/~ece539/fall13/project/AmorimTorres_rpt.pdf

[3] http://cs229.stanford.edu/proj2013/ChengDadeLipmannMills-PredictingTheBettingLineInNBAGames.pdf

Regarding the features and algorithms, in working with a dataset with such a huge range of features, it's hard to engineer features that will have high predictive power in the model. Thankfully, there is a suite of analyst research on basketball that we can use to inform important features like number of rebounds or drives down the court, but there will likely be additional information hiding in the data and also potential key features that are not in our dataset to begin with. Another key challenge will be generalizability. When training our model, we will likely face overfitting. An ML algorithm could foreseeably weight features from basketball game that are in truly irrelevant, but are learned nonetheless. Additionally, we are attempting a fairly difficult regression challenge by trying to predict the spread of a game since chance is especially prominent in sports games. To address the challenges in feature selection we could analyze expert opinions on important game characteristics and use quantitative selection techniques like dimensionality reduction or clustering.

B. Feature Selection

As mentioned previously, this model sought to go beyond traditional macro-game models by including micro-game features. To supplement these micro features, though, some typical macro-game features were retained to fill in areas that the SportsVu shot-tracking data could not. Those features included rebounds, turnovers, and free throws per game along with cross-game indicators like win percentage and number of days rest between games.

As stated, the core features used to make predictions were micro-game ones. So, the features of a given game X would be the running average of the team's statistics in all of the games prior to X in that season [Game 1 to Game (X-1)]. Our approach created features from shot-tracking data by breaking offense into the following components:

- Type of shot (location on the court)
- When the shot was taken (fast break vs. half court offense)
- How difficult the shot was (discretized range from "wide open" to "very tight")
- Type of play leading to shot (1 on 1 vs. passing to open shot)

Each of these categories both had a volume feature (number of shots made) as well as a accuracy feature (shots made/shots attempted) for both two and three point shots. To measure defensive prowess in these categories, we used the running averages of opponents that a team had played in their games prior to X in the given season. We chose these categories as ones to properly segment the offense and defense through consultation of literature on basketball strategy, including "Basketball on Paper" by Dean Oliver.

C. Models

Input: For each game, the micro and macro features for each team were based on the running average for all metrics across every preceding game that season. The defensive side was the average of how other teams had performed offensively in the macro/micro metrics.

Label 1: The binary outcome of the game (Win vs. Loss)
Label 2: The score differential of the game (Team A won by X points)

We used the following kinds of models in our project:

Classification*:*

Naive Bayes: We considered naive bayes as a simple probabilistic algorithm. However, Bayes assumes independence between the features which was certainly not the case in our dataset.

Random Forest: As a tree ensemble algorithm, one main advantage is that it does not expect linear features or even features that interact

linearly. As a result, because random forest is nothing more than a bunch of decision trees combined, they can handle our dataset and its features fairly well.

SVM (Support Vector Machine): SVMs use a hinge loss function and is particularly useful when the data is not linearly separable. We tried different kernels, but rbf worked the best. SVMs are also effective for looking at high dimensional data such as ours with a number of offensives/defensive features.

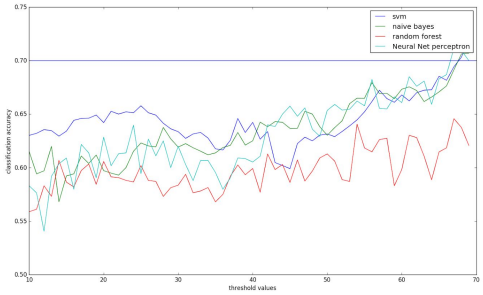
Multilayer Perceptron: The multilayer perceptron was a fourier into neural nets to see how a classifier that had hidden layers would perform.

Regression:

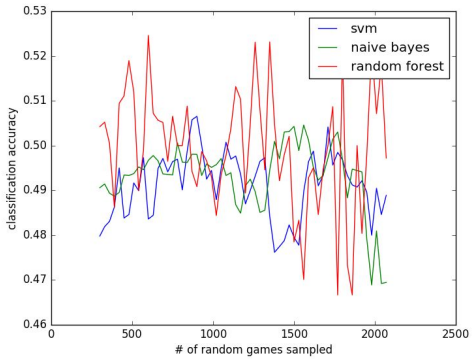
SVR (Support Vector Regression): The SVR uses an epsilon insensitive loss function and performs linear regression in the high-dimension feature space. The underlying fundamentals are similar to SVM.

Bayesian Ridge Regression: Uses Bayesian statistical analysis to perform a regression on the dataset.

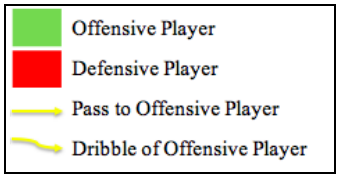
IV. Results and Error Analysis

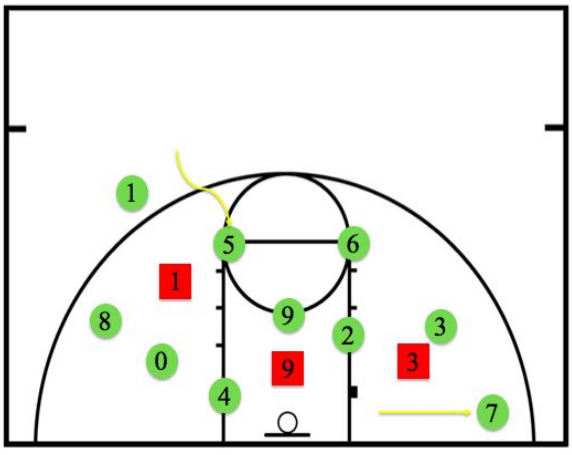


Beginning with classification, we witnessed temporal improvement throughout a season for all classifiers with peak accuracy at around 72% classification at games 69 and 67 for the SVM and Multilayer perceptron respectively. This accuracy, though late in a season, surpasses the Oracle of 70% which is modeled with insider information and additional in-game features that we didn't have access too. The training errors on this classification were .672 and .681 respectively as well, suggesting that overfitting was not a significant error. In addition, we compare our accuracies here using sequential data in season to a randomized classifier below that trained and tested on random games that hovered around 50%. This further indicates that using the sequential game training approach was important to the success of our model though it limited the net amount of data we could use.



Classification Feature Results





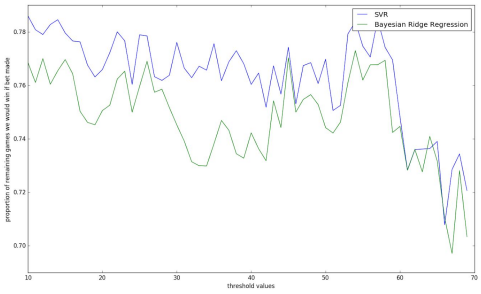
0. DEF-22-18 Very Early (EFG%)
1. DEF-6+ Feet - Wide Open (EFG%)
2. DEF-15-7 Average (FG2 Freq. %)
3. DEF-0-2 Feet - Very Tight (FG2%)
4. DEF-22-18 Very Early FG2%)
5. DEF-Pullups (EFG%)
6. DEF-15-7 Average (EFG%)
7. DEF-Catch and Shoot (EFG%)
8. DEF-Totals (FG2 Freq. %)
9. DEF-4-6 Feet - Open (FG2%)

After doing Mutual Information (MI) tests, the top 10 most predictive features were revealed. Interestingly enough, they were all defensive features, mostly focused on the following:

- Defense against the fast break (#0, #4)
- Defense against usual HC plays (#2, #5, #6, #7, #8)
- Defense when near offensive player (#4)

Our hypothesis on why defensive features seemed to be the most predictive relates to consistency. Although the "Hot Hand Fallacy" has been disproven within a game, we believed that offense had the potential to be more inconsistent (variation in game-to-game shooting prowess) than defense. Additionally, we postulated that defensive success was more system-based than offensive success (i.e. there is a greater impact on offense if a certain player is subbed out than on defense). Because our model does not capture which players are in the lineup before each game, variation in which players did not play would then cause more variation in the offensive metrics. Therefore, this could lead to the more stable defensive metrics being more consistently predictive. If these hypotheses prove to be true, they lend credence to the model as also being a tool help humans to better understand the in-game dynamics that lead to score differentials.

Betting Line and Regression



For the betting line regression, we assumed that a player would bet on every game. If the actual score of the game was on the same side of the betting line as the actual score, the bet was won. There was roughly an equal proportion (939 to 868) of games under the betting line to over the betting line that our model predicted correctly. As shown in the graph, the % of bets won hovered around 76% until game 60 at which it dropped to around 72%. Monetarily, if the player were to adopt a strategy where he/she bets half his/her money on every game, the winnings would be close to $10.8 million with a starting pot of $1000 and betting from game 70 onward. This is remarkably profitable.

We postulate that the games after 60 present confounding in our model's ability to correctly predict the score of a game. In particular, playoff seats tend to be decided late in the season so there's an advantage to season abnormal behaviors such as giving key players rest. It is also important to note that the Vegas betting lines did not have the micro features available to us that were made public after the 14-15 season. As a result, our regression model may have an inherent advantage by being privy to features past models did not have access too.

V. Future Work

Moving forward, the most important step would be to validate our results with additional seasons of information. The micro-game data is available for the 2015-2016 seasons as well as the 25 games that have been played so far in the current, 2016-2017 season, and could be used to

confirm the model's success. On a related note, to better understand the betting success of our model, instead of betting a similar amount per game, we could use the predicted differential relative to the spread to derive a "confidence score" that would then proportionally determine the amount of money bet per game. Then, instead of betting accuracy as a metric for success, we could also evaluate the model by true betting standards: amount of money earned.

Additionally, our features could be improved if we modulated the averages that depict quality of offense and defense by the quality of the opposition. For example, if the Rockets held the Warriors fast break offense to a low accuracy and frequency, a component that was particularly good relative to the rest of the league, then the Rockets' defense should get a greater reward than if they held a worse team to the same totals. We could also place more emphasis on the recency of results by having recent games contribute more to statistics than games near the beginning of the season. Other macro features could also be added to capture a team's momentum (number of games won or lost in a row) as well as the quality of coaching (measure the coach's success historically in the NBA against certain types of teams.)

Finally, additional Machine Learning techniques could be implemented to improve the predictive power off of the features currently used. Particularly, we could try advanced eliminating counterproductive features through advanced feature selection techniques such as RFE that would enable dimensionality reduction. An area of future experimentation is also deep learning, through which we could perhaps gain deeper insights into the features that are relevant in basketball - we would ideally have a larger training set to do deep learning, however.