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

Simar Mangat, Maneesh Apte, and Keyur Mehta

Motivation

Problem

In the past few years, the NBA has added a number of player tracking systems that produce intricate and robust advanced statistics for every player in every game. This data is exploding but has not been meaningfully used to better predict the outcome of NBA games. We wanted to know 1.) how to best extract features from in-game data and 2.) how to combine them to better predict game outcome related to existing work.

Prior and Related Work

Project 1: Use previous game data to predict the outcome of the current game. This system used data about teams wins and losses, especially relative who they were against, with no usage of play-by-play data.

Project 2: Furthered ML algorithms by using SVMs in conjunction with a lasso regression and bootstrap aggregating but still did not use in-game shooting data.

A number of other systems have undoubtedly been used but, as these predictions are often used to in gambling, much of it is proprietary.

Goals

To extract rich enough data from each game such that sufficient accuracy can be achieved by only using games from the current season.

To better understand the dynamics that lead to score differential by extract "micro-game" features.

Approach

Data Source

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 via covers.com for the 2014-15 season.

Model

Games were modeled as classification/regression problems. The features of a given game would include the average of the teams statistics of all of the games before it this season. Our approach broke offense and defense into its important components:

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

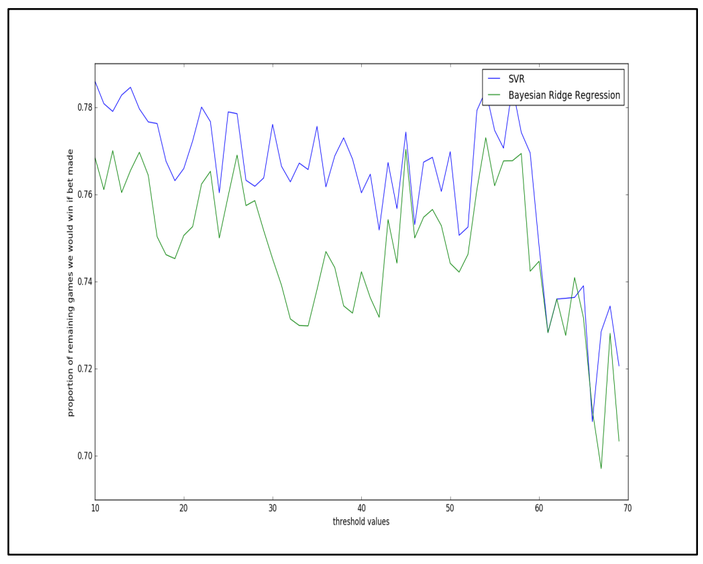
Input: For each game, the micro and macro features for each team were based on the running average for all metrics across every preceding game that season. The defensive side was the average of how other teams had performed offensively in the macro/micro metrics.

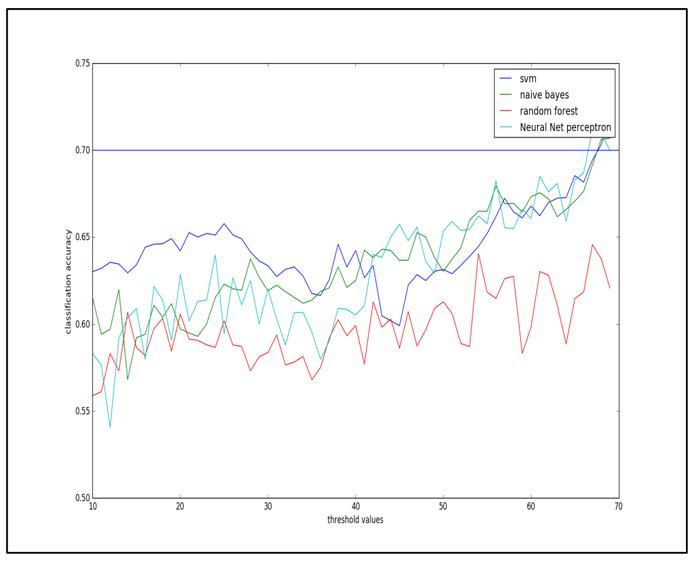
Label 1: The binary outcome of the game (Win vs. Loss)
Label 2: The score differential of the game (Team A won by X points)

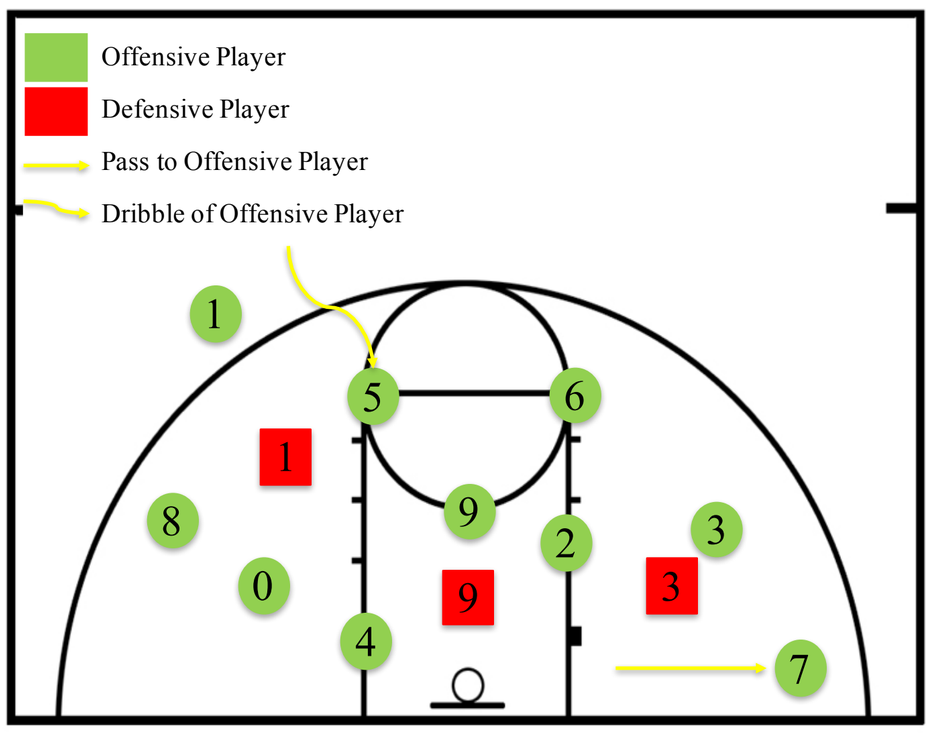
Algorithms

- SVM
- Naive bayes
- Random forest
- Multi-layer Perceptron (MLP)

Results







Top 10 Features:

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

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

0. DEF-22-18 Very Early (EFG_PCT)
1. DEF-6+ Feet - Wide Open (EFG_PCT)
2. DEF-15-7 Average (FG2A_FREQUENCY)
3. DEF-0-2 Feet - Very Tight (FG2_PCT))
4. DEF-22-18 Very Early (FG2_PCT)
5. DEF-Pullups (EFG_PCT)
6. DEF-15-7 Average (EFG_PCT)
7. DEF-Catch and Shoot (EFG_PCT)
8. DEF-Totals (FG2A_FREQUENCY)
9. DEF-4-6 Feet - Open (FG2_PCT)

Analysis

- 72% max accuracy prediction via SVM with 68 games taken as training
- 72% max accuracy prediction via multi layer perceptron with 68 games taken as training
- Accuracies for win-loss predictions were better as more in season games were taken as threshold
- Predictions for beating the betting line consistently were ~78% using SVM regression
- Past a threshold of 58 games the regression classifiers seem to poorly generalize on new games
- Defense seems to be an important indicator in the performance of a team

Challenges

- Original Kaggle dataset with micro features was incomplete and data was then manually scarped
- Finding historic betting line data is intentionally made challenging by betting companies
- Consistency across the storage of scraped data
- Size of dataset is relatively small
- The features may not be capturing the relationship between offense and defense as well as they could be

Next Steps

- Regularization of features
- Eliminating counterproductive features through techniques such as RFE
- Add domain knowledge about basketball as a prior in our machine learning algorithms
- Develop a function to bet money based on predicted scores and determine our net profit over a season
- Try different regression algorithms for the betting line calculations
- Feature engineering around relationship between offense and defense