USING SOCIAL NETWORKS TO PREDICT GROUP TRANSITION

A PREPRINT

Emily J. Evans
Department of Mathematics
Brigham Young University
Provo, UT 84602
ejevans@mathematics.byu.edu

Jason Kinghorn
Department of Mathematics
Brigham Young University
Provo, UT 84602
jason.m.kinghorn@gmail.com

Joseph Leung
Department of Mathematics
Brigham Young University
Provo, UT 84602
joeleung16@outlook.com

Benjamin Z. Webb
Department of Mathematics
Brigham Young University
Provo, UT 84602
bwebb@mathematics.byu.edu

August 22, 2019

ABSTRACT

This is the abstract

Keywords First keyword · Second keyword · More

1 Introduction

A classic problem that exists in Network Theory literature is that of the link prediction problem: trying to predict whether two nodes in a network will connected in some way give some previous information. A classic example of this nowadays is if two individuals in a social network such as Facebook will eventually "friend" each other if, say perhaps, they both have a friend in common. We expand on this question by observing a sort of group-link prediction; given information about some individual in a network, can we predict whether they will be a part of some group or organization? Applications of such a question can be seen with not too much effort; being able to answer this question could allow us to predict if an employee may eventually work for some particular company, judge if a leadership member in terrorist groups may play a role in a different one, or vice-versa, tell some organization what kinds of members they may anticipate having. In this study, we use baseball data as a proxy. As baseball has evolved to become a game of great interest to many statisticians, it follows that much data exists on the performance of each individual player going back several years. Thus, based on this data and each player's connection to their team, we designed a bipartite graph of sorts with each baseball player being connected to a team during a season. Ripe data and an institutional structure that we could represent graphically ultimately persuaded us to use baseball as empirical evidence for our purposes.

2 Methodology

2.1 Baseball performance dataset

The baseball performance dataset was obtained from `fangraphs.com` in late 2018 and covers the 2001 – 2018 baseball seasons. This website provides a number of baseball statistics for teams and individual players in easily downloadable `.csv` format. We collected the data for our data set by downloading the a team statistic `.csv` file for each major league baseball team and each season under consideration. We chose to focus on the years 2001–2018 so that the number of teams (30) would stay constant. Although the website contains a number of statistics, we chose to focus on a few

key variables. First, each player has a unique identifier. This identifier allowed us to track the careers of each player, specifically allowing us to know for each season: how long each player had played on their respective team, how long their professional careers have been during that year they played, and whether they left the team at the end of the season or not. In addition to this identifier we collected the following statistical information for each player: the weighted runs created plus (wRC+) score, earned runs average (ERA-) score, the dollar value of performance and their position on the team. The wRC+ score is an adjusted runs created measure that takes into account external factors (such as how hitter friendly a ballpark is) and adjusts the score so a wRC+ of 100 is league average and 50 would be 50 percent below league average. The ERA- works similarly in that it takes into account difference in ball parks, but is aimed towards pitchers. An ERA- of 100 is considered average, and an ERA- of 75 is considered to be 25 percentage points better than league average. The final performative statistic gathered for each player was the dollar value of perfomance which gives an estimate of the worth of the player in the free agent market measured in millions of dollars. While we acknowledge that there are many measures of performance, we concluded that these traits allowed a certain flexibility in judging the performative abilities yet minimized the number of possible variables for our learning algorithms.

We performed some necessary cleaning of the data from `FanGraphs.com` in order to manipulate it into a useful form. The cleaning steps were as follows:

- Merge the data into a large dataset as the data was downloaded by player position.
- Remove extra statistical features.
- Create a *leave* variable which specifies if a player is to leave after that season. This variable is critical as we will focus our efforts only on predicting next teams of those players that leave.
- Create a *target* variable which specifies what team a player plays for the next year, or if they do not return to play that season.
- Engineer other statistical measures such as "career length" which would be how long a player had played up until that year.

One might note an important factor in determining how long a player may play for one team or the other is the contract they have with that team, a detail we have omitted in this dataset. The focus of our work is predicting where a player will transition to and not whether a player will transition.

The total number of unique players in our data set was 3374. For the specific 2016 season we had:

- 826 unique players split between 30 teams.
- 403 players who were not on the same team in 2017. Of these 403 players
  - 189 players retired, or returned to the minor leagues
  - 214 players transition, i.e., returned the next year to play on a different team.

We note that these numbers are consistent with our data that approximately 50% of players leave their team each year in some manner.

2.2 Baseball social network dataset

Beyond performance data for each player, we also hoped to understand the role social connections play in the team transition in baseball. Acknowledging that it would be extremely difficult to create a ground truth social network for players, we created an approximation of a social network utilizing twitter data. We accomplished this by first obtaining all the known twitter handles of baseball players from `Baseballreference.com` and then creating a directed social network of players where player A has a connection to Player B if Player A followed Player B. Moreover for each baseball season we created a bipartite graph between teams and players. Using these two networks we created a team affinity score for each player as follows: If player A followed five different players on a different team, we would state that player A's social affinity score for that team was five, and so on for every team. See Figure 1 Of the 3374 unique players in our data set, we were able to collect twitter handles for 664 of them. These created a directed player social network with 6650 directed edges. Most players have a relatively small number of connections but a few players have a large number of connections (over 100). The distribution of connections is shown in Figure 2.

2.3 Team stratification dataset

Conjecturing that the prestige of a team may influence where a player may move we also collected three metrics to measure team prestige. First we conjectured that teams with more funds (typically resulting in higher pay and more

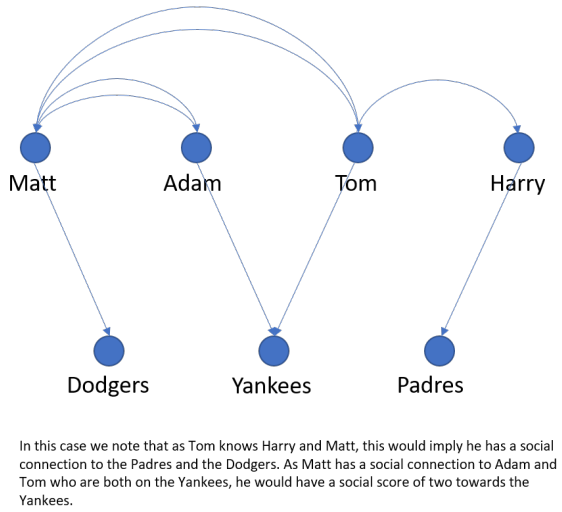


Figure 1: A visualization of the creation of the social affinity score.

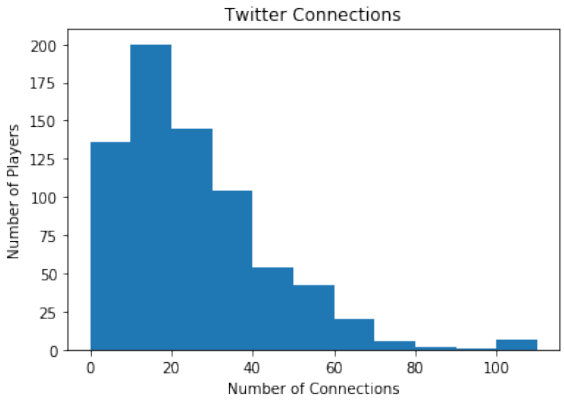


Figure 2: A histogram showing the distribution of twitter connections.

highly paid players) would be more desirable. We collected data on each team's valuation for each year in question through Forbes.com. This allowed us to rank the teams by total net worth. We also conjectured that players would like to be on teams with highly skilled players. To gauge the skill level of the team we looked at the WRC+ average of the team. Conjecturing that players tend to want to be on teams that frequently win, we collected the total number of wins to provide a winning rank.

2.4 Analysis

We utilized ensemble methods to predict which teams players would transition to. Ensemble methods combine several models (predictors) which operate independently and are typically very good for classification problems. The outputs of the models are then recombined into a single prediction that should be better than any of the initial models. We used four types of ensemble methods to make predictions that can be classified into two different categories: Randomized decision trees (Random Forests and Extremely Randomized Trees), and boosting algorithms (Adaptive boosting and Extreme Gradient Boosting).

The idea behind randomized decision trees is that a random subset of the data is sampled and a decision tree is created. This individual decision tree suffers from over-fitting and by itself is not useful. Random forests operate by in esence averaging many decision trees trained on different random parts of the training set. In fact random forests usually cerates thousands of these decsion trees. Of these thousand of trees, it is likely that a few of them are actually good models. To determine the "best" decision tree in the collection of trees

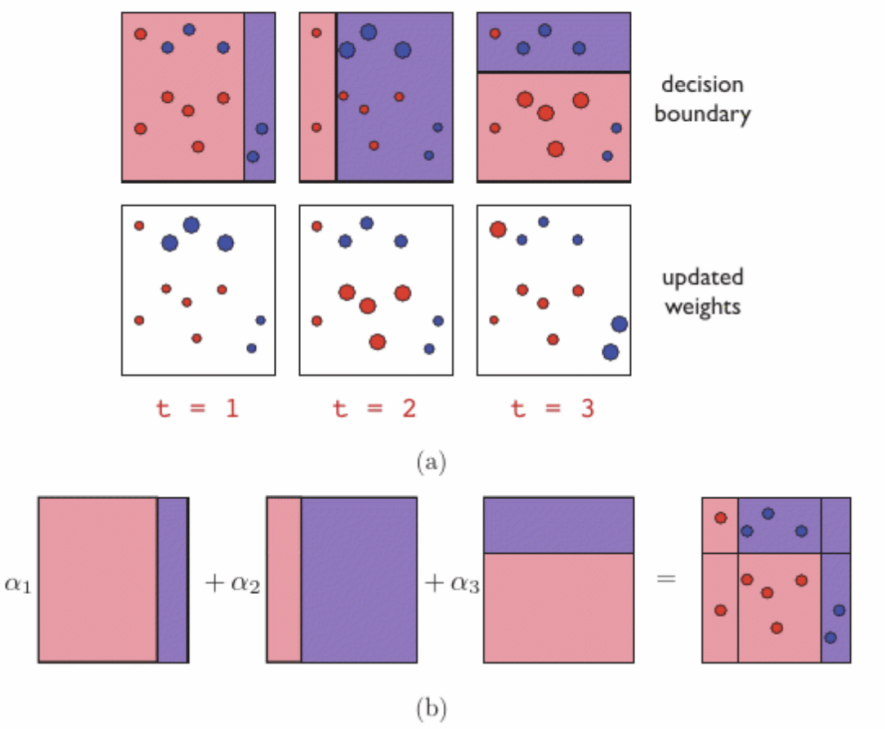


Figure 3: A figure depticting the AdaBoost algorithm in action. Observe that at the conclusion of each boosting round the weights of the incorrectly identified variables are increased and the weights of the correctly identified variables are decreased. The final learner is an ensemble of the weak learners. *Ben: If you like this figure we will have to recreate it. I stole it from a book but it should be easy to recreate*

In extremely randomized trees (see ExtraTreesClassifier and ExtraTreesRegressor classes), randomness goes one step further in the way splits are computed. As in random forests, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule. This usually allows to reduce the variance of the model a bit more, at the expense of a slightly greater increase in bias:

Boosting algorithms are a family of algorithms whose aim is to create a strong learner from a weak learner. Boosting algorithms work by applying the weak learner sequentially to weighted versions of the data where in each sequential application mis-classified data is given additional weight. The weak learner can be any classification or regression model, but the most frequently used learner is a decision tree [1]. Boosting seeks to find a function f such that

$$\min_f \sum_{i=1}^{N} L(y_i, f(x_i)) \tag{1}$$

where $L(\cdot, \cdot)$ is a loss function and f is the function that classifies the data, has a solution. In our work we consider two different boosting algorithms that use decision trees as learners: Adaptive Boosting (AdaBoost) and eXtreme Gradient Boosting (XGBoost). We briefly describe how these algorithms work.

In Adaboost the weak learners are decision trees with a single split (frequently called decision stumps).

XGBoost

- Random Forest.
  - The Random Forest package from Python's Scikit Learn package is an algorithm which mainly utilizes decision trees to determine a classification. As we used it, the accuracy based on performance data was mildly better than accurate, often giving an accuracy of about 6%.
- Extra Trees.
  - Extra trees is similar to Random Forest, but a little more detailed, and thus resulted in better accuracy. It correctly placed a player in their next team approximately 10% of the time.

- XGBoost.
  - XGBoost was the most accurate of all the algorithms, often being 12% accurate with a standard dataset.
- Adaboost:
  - Performed similarly to Extra Trees

3 Results

Before delving in, it is worth noting that as a baseline we use what one might get if guessing the player's next team randomly, that being about 3% accuracy. A detail of much more interest than simply the algorithm used however was how the combination of features affected the accuracy regardless of the algorithm. At first glance, one might suppose that throwing in all of the performance data at once would result in the most accurate results. However, it turns out that carefully choosing what data to use is far more effective. For example, rather than looking at the combined effect of a player's position, how long they played, their performative wRC+ and ERA- scores, and whatever else, simply looking at the position a player had had a large advantage over putting all the details combined. When incorporating social data however, the accuracy would increase generally regardless of the combination of performative data. Consider the table below:

Features	Social Data Included	XGBoost Accuracy	AdaBoost Accuracy	Random Forest Accuracy	Extra Trees Accuracy	Logistic Regression Accuracy
Positions Only	N	23.6063%	23.2578%	23.6063%	23.6063%	23.1707%
	Y	20.9930%	24.2160%	20.8188%	21.1672%	23.2578
All data	N	22.5610%	19.7735%	17.6829%	16.8989%	22.6481%
	Y	22.2997%	19.7735%	18.9024%	16.8989%	22.6481%
Career Length	N	23.2578%	23.4321%	12.8049%	13.6836%	22.9965%
	Y	25.2613%	24.2160%	14.8084%	14.6341%	23.3449%
Social Only	Y	22.5609%	20.5575%	21.7770%	22.2125%	22.2125%
Team Only	N	22.5609%	18.6411%	13.4146%	16.8989%	23.9547%
Performance Only	N	21.4286%	21.5157%	11.4111%	9.6689%	22.8222%

It is fair to note, there may exist discrepancies in data. Different formats of data were tested on as when downloading from the fangraphs database it would often be updated with new statistics throughout the course of our research, as well as new players as the baseball seasons would transition from one to another. Every time a new batch of information would be downloaded to test a new particular statistic, we would also need to repeat the data cleaning process. It so follows that the data was not identical every time during our experiments.

Nonetheless, patterns exist in how different features were combined. As we see above, adding in social data always had a complementary effect to whatever features it was combined with, though no combination is particularly more effective than the other. Interestingly, without social data, sometimes less was more. Having all the details could detract from the effectiveness of just having just some data. We theorize that this may be because certain features may work against one another in some ways; for example, an individual that has played for several seasons on a particular team might be anticipated to be traded soon. However, if that same player has an astounding wRC+ score, the same team might be inclined to keep that player on the team as long as they can manage. Intuitively, this would be true for many star players such as Derek Jeter. Of course, this could work in the opposite way. A player with very high performative statistics may be more inclined to leave if they are currently on a mediocre team, and are thus sought after by other General Managers. What is interesting though is how this effect seems to appear as we add social data to our algorithms. There is no prejudice against the combination of performative statistics once we add performative data, and it only assists our machine learning algorithm's effectiveness. Why exactly this is so is still uncertain. Perhaps the way players are transferred between teams is inherently social in nature, and thus performative data simply helps classify these transfers a bit more.

In an attempt to better understand how the performative features were used in the classification process, we used an inherent module within XGboost. In order to determine what features would be most effective at successfully predicting where a player would go or not, various measures were considered, mainly considering the cover, gain, and strength. To understand the different between the three:

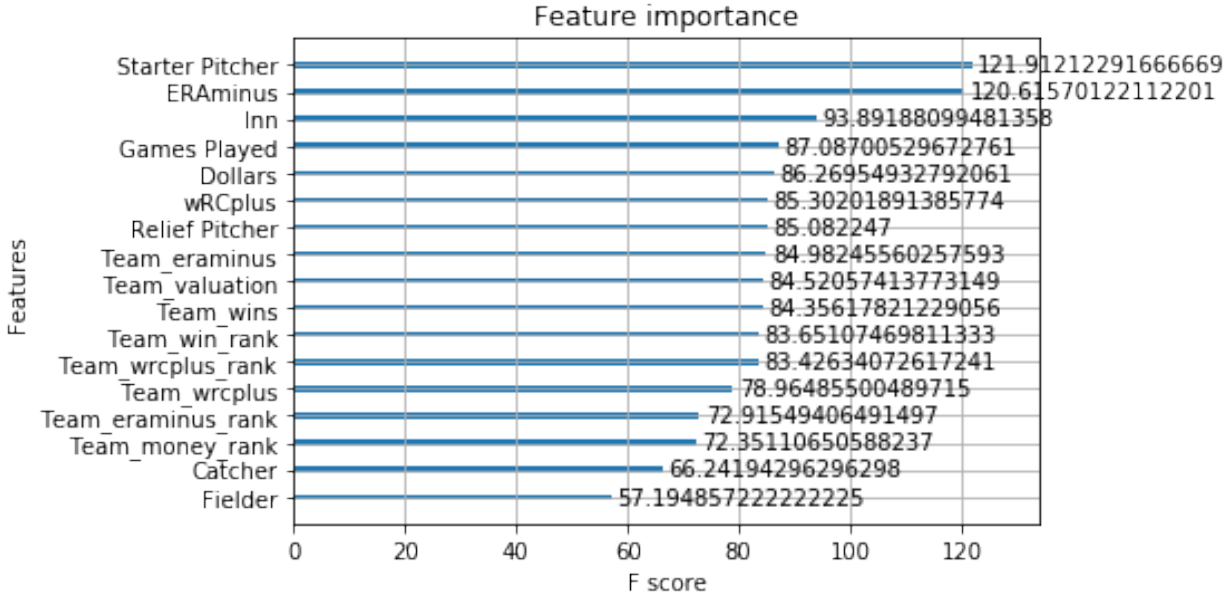
1. **Weight**. The number of times a feature is used to split the data across all trees.

2. **Cover**. The number of times a feature is used to split the data across all trees weighted by the number of training data points that go through those splits.

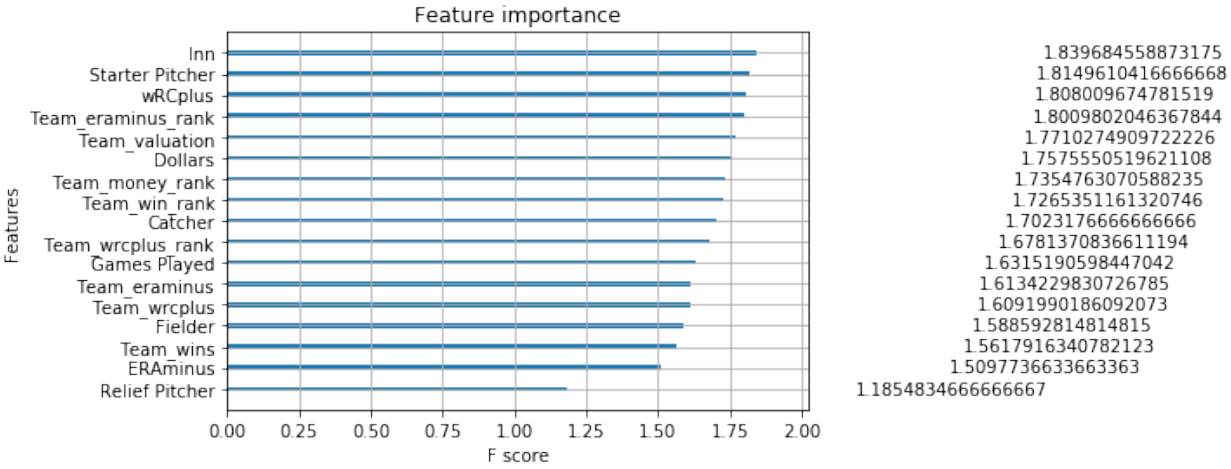
3. **Gain**. The average training loss reduction gained when using a feature for splitting.

The graphs are seen thus:

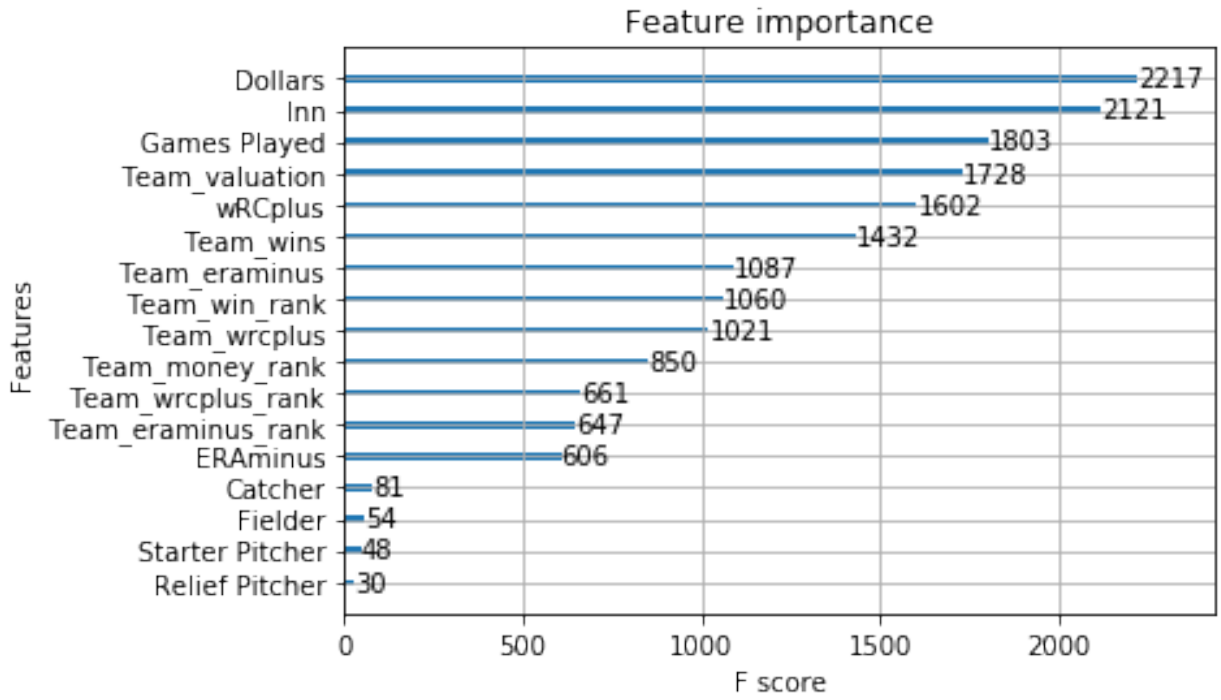
Cover



Gain



Weight



We can note quickly that these graphs are rather conflicting. We don't gain much information by looking at the roles of all the features when they're put together. Thus, it seems unlikely that there is just one feature that is most effective. In adding social data, the results of our algorithm increase fairly significantly. Admittedly, with the data, the general base level increased substantially as well, to about 10-12% accuracy for some reason. It is possible that adding in 2018 data made some difference, and also some extra cleaning was necessary to incorporate the social data. Regardless, adding the social data increased the accuracy to approximately 20%, regardless of what performative data was contributed. Alone, the social data performed reasonably well – about 12%.

4 General Result

It seems that we can conclude that there is sufficient information in facets of performative data and social data, and especially the interaction of the two, that allows us to predict what team some individual may transition to at the end of the baseball season. Rather than a mass of information, a particular kind of interaction of features seems to be most contributive to the effectiveness of the prediction, such as looking merely at the position the baseball player may have. Otherwise, too much data may contribute to an issue of overfitting, and ultimately decrease the accuracy of the algorithm.

References

[1] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.