Predicting Baseball Hall Of Fame Inductions

Michael Hirsch

ILLC, University of Amsterdam

michaelahirsch@gmail.com

Abstract

Every year, the Baseball Writers Association of America votes on a new Hall of Fame class. Each ballot consists of 10 votes, and players need to appear on 75% of ballots in order to be inducted into the hall. Players have 15 years to get inducted, and are no longer eligible if that time period has passed. Here we attempt to classify hall of fame batters based on their career statistics using decision trees and an ensemble method, random forests.

1. Introduction

Major League Baseball (MLB) has been keeping thorough records of batting, picthing, and fielding statistics since its innagural season in 1869. Recently, with the advent of sabermetrics by the Society for American Baseball Research (SABR), many new metrics building on traditional statistics were created. Such metrics caught the eye of stasticians, as these new metrics allowed for the creation of even more powerful predictive models.

In this paper, we are investigating what makes a Hall Of Fame (HOF) batter. There have been over 20,000 Major League baseball players in the history of the organization, and only 211 of them have been inducted into the HOF. There are several ways in which a player can be inducted, and we only concern ourselves with players inducted from Baseball Writers Association of America (BBWAA) ballots, as this process is regulated. Often, fans and players feel that a worthy candidate is unfairly denied entry into the HOF because of voter bias, stacked ballots, or negative associations with performance enhancing drugs. The model proposed in this paper should be able account for these cases.

2. Data

Data was collected from the Lahman Baseball Database, a freely available database that has been contiously updated since 1994 with the help of SABR and many individual researchers. There are many data tables available for download, but the ones that we are focusing on are *Batting.dat* and *HallOfFame.dat*. Batting statistics were supplied on a by-year basis, so a player's career statistics were computed by aggregating his yearly results. The batters' data was then merged with hall of fame voting results using the player's ID string. All data preparation was done within R, the environment where we also build our learning models.

Attention is paid to 18 different batting statistics over the course of the player's career:

Below is a list of features used

1. G: games played
2. AB: at-bats
3. R: runs
4. H: hits
5. X2B: doubles
6. X3B: triples
7. HR: homeruns
8. RBI: runs batted in
9. SB: stolen bases
10. CS: caught stealing
11. BB: walks
12. SO: strike outs
13. IBB: intentional walks
14. HBP: hit by pitches
15. SH: sacrifice hits
16. SF: sacrifice flys
17. GIDP: grounded into double plays
18. Inducted: classifier for HOF induction

We needed to further filter our data by not including pitchers in our set of observations. This was done by removing those observations that indicated that the player recorded less than 300 RBI in his career. This number was

not arbitrary, but rather chosen by cross referencing my data with HOF data from Baseball-Reference.

Our software of choice for this paper is R. We will be using several packages for our learning techniques, indicated in the sections below. Before training our model, it is important to have a quick look at our data.

3. Tree-Based Methods for Classification

In this section, we will give an account of two learning methods that we will use for the basis of prediction: classification trees, and an ensemble method, random forests. The information here is theoretical, and we will encounter applications of these ideas to our HOF data in the next section.

3.1. Classification Trees

Classification trees are a fairly effective predictive tool that are lauded for their high degree of interpretability. When constructing a classification tree, we begin with the full set of observations and begin dividing the predictor space into non-overlapping regions by means of binary splitting. In prediction, we assign each observation in a given region of the predictor space to the most commonly occuring class of the training observations in that region.

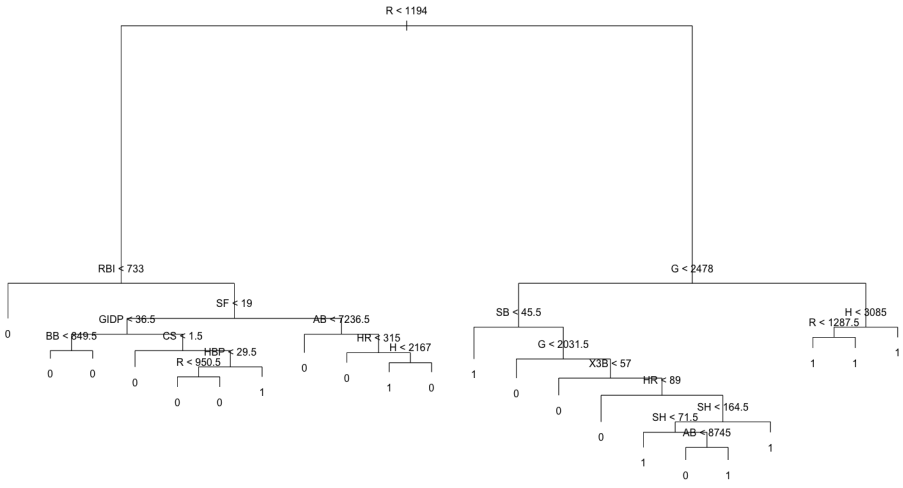

Figure 1: A very bushy classification tree

When choosing splits in our tree, we are concerned with the textitGini Impurity at each split, that is, we want to choose the feature to split on based on which feature split will give us the most pure sucessive node. Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the subset. Gini impurity can be computed by summing the probability of each item being chosen times the probability of a mistake in categorizing that item. It reaches its minimum (zero)when all cases in the node fall into a single target category. This defined as:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) = 1 - \sum_{k=1}^{K} \hat{p}_{mk}^2$$

where $\hat{p}_{mk}$ denotes the proportion of training observations in the mth region that are from the kth class. So G can be seen as a measure of total variance across the K classes. $G = 0$ when all observations in that region fall under on category. In splitting at nodes, we look to minimize this value.

There are other methods by which splits can be determined, another popular one being *information gain*, which is related to entropy. However, we concern ourselves only with *Gini Impurity* since this is the method by which trees a generated usin the *trees* package in R.

3.2. Ensemble Methods

Ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the individual learning algorithms composing the ensemble. Here, we look at two such methods, bagging and random forests.

3.2.1. Bagging

Decision trees as discussed in the previous section often suffer from high variance, meaning that if we fit a decision tree to two random samples from a set of training observations, we could end up with vastly different results. Bagging is a method for reducing the variance of any learning method by taking repeated samples from a single training set and taking the, in the case of classification, the most commonly occurring class among the boostrapped predictions.

On average, each of the B bootstrapped samples will take about two-thirds of the training observations. The remaining observations that were

not used in fitting a bagged tree are referred to as the *out-of-bag* (OOB) observations, and we can predict the target varialbe for the ith observation using each of the trees in which that observation was OOB. This will yield, on average $B/3$ predictions, from which we can predict the classification for the observation by taking majority vote. Using this method, we can get a single OOB prediction for each of our observations, obtaining an OOB error, which is a valid estimate of the test error for our bagged model.

Bagging also gives us insight into which of our predictors were the most important in creating our model by looking at the Gini Impurity. To do so, we take the sum of the amount that Gini Impurity has dereased by splitting on a given predictor and take the average over our B trees.

3.2.2. Random Forest

The random forest technique is quite similar to bagged trees, with the main difference being that at each of the splits only a random sample m of the set of all predictors p is considered. Typically, $m \approx \sqrt{p}$ predictors suffice. This technique helps to decorrelate the tree, as there may be one predictor that has a very high influence in classification. In the case of bagging, this predictor will most likely be used as the first split in a majority of the trees, causing most bagged trees to be quite similar and correlated. By forcing each split to consider on a subset of the predictors, on average $(p - m)/p$ of the splits will not consider this strong predictor.

In this paper, we will be utiliziing the *randomForest* package in R.

4. Prediction Models

In this section, we apply the techniques introduced in the last section to our HOF data. We will start by creating a single decision tree based on some training data, and then predicting the induction classifier of our training data based on our model. We will also look at some metrics that will allow us to test the effectiveness of our classification models in order to see what parameters would be best to use.

4.1. Decision Tree

Using R's tree package, we can

```
text(tree.hof,label="yval", all = FALSE)
```

4.2. Random Forest