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.

We will investigate predictions based on both classification trees and random forests. Random forests, introduced by Leo Breiman of UC Berkeley in 2001, is an ensemble learning technique that can be used for both classification and regression and improves on decision trees by correcting for overfitting. A random forest for classification consists of a collection of decision trees and outputs a prediction representing the most commonly occuring prediction of the decision trees.

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. Batting statistics were supplied on a by-year basis, so a player's career statistics were computed by aggregating his yearly results. The data was then merged with awards, all star, and hall of fame voting results using the player's ID string. We will only be predicting whether or not players will be inducted by the BBWAA, since this is the current method of voting. All data preparation was done within R, the environment where we also build our learning models.

Attention is paid to 11 different statistics over the course of the player's career:

statistic	meaning
numSeasons	number of years in MLB
tAB	at-bats
tR	runs
tH	hits
tHR	homeruns
tBA	batting average
tRBI	runs batted in
tSB	stolen bases
mvp	number of mvp awards
gg	number of gold glove awards
Allstar	number of all star games played
inducted	predictor

Table 1: Hitting Statistics

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.

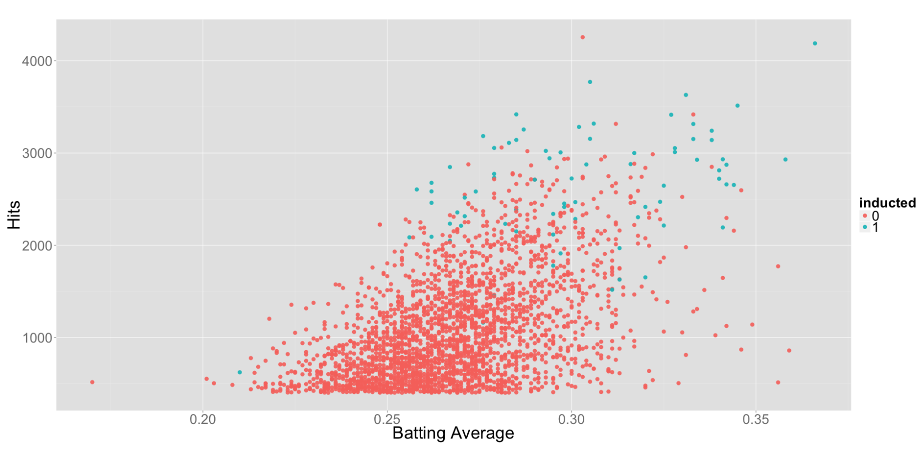

Figure 1: Batters with more hits and a higher batting average are inducted

We can see that more hits and a higher batting average seem to correlate with induction. This makes sense, as one often judges the quality of a batter by his batting average, the numbers of hits a player has per at-bat. Keep in mind that a lot of the players with many hits and a high batting average that weren't inducted are those players who's induction we seek to predict. Also, That player with over 4000 hits who is not in the Hall of Fames is Pete Rose, who received a lifetime ban from baseball activity for gambling against his own team.

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. We will also introduce the concept of bootstrap aggregating (bagging) which is a fundamental ingredient of 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

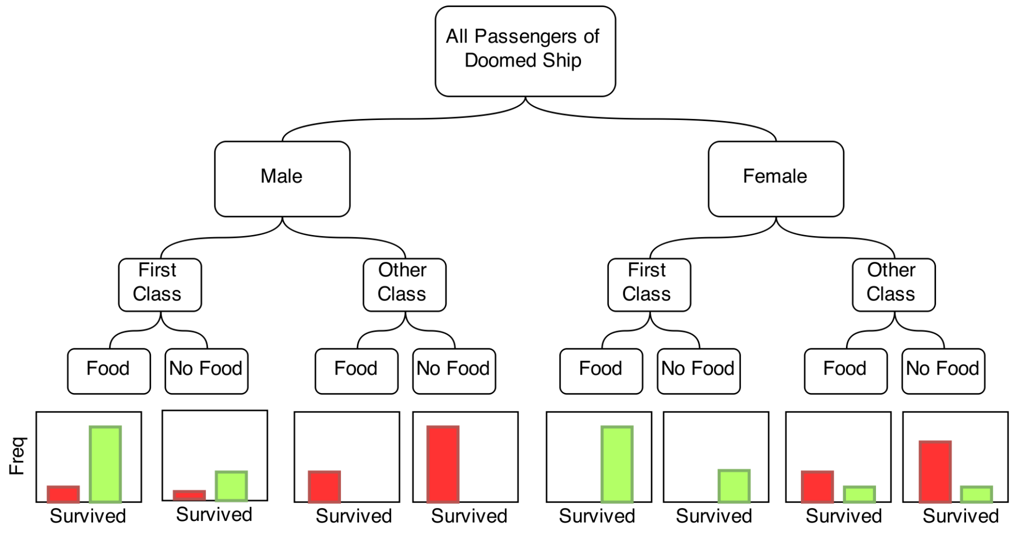

Figure 2: Decision Tree for Survival of Boat Disaster

to the most commonly occurring class of the training observations in that region.

When choosing splits in our tree, we are concerned with the *Gini Impurity* at each split. We want to choose the feature to split on based on which feature split will give us the most pure successive 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:

$$G = \sum_{i=1}^{m} p_i(1 - p_i) = 1 - \sum_{i=1}^{m} p_i^2$$

where i takes on values in $\{1, 2, \ldots, m\}$ and p_i is the fraction of items labled with value i in the node. So G can be seen as a measure of total variance across the m classes. Since this paper will only be discussing trees with binary splits, m will also be taken to be 2. Note that $G = 0$ when all observations in that node fall into one class. In splitting at nodes, we look to minimize this value.

Let us illustrate this with a very simple example of a decision tree created to predict the survivors of a boat disaster, show in Figure 2. The root node contains all passengers of the ship. We then split the full set of observations according to their sex, followed by a split on what type of ticket they had, followed lastly by a split on whether or not they chose to order a meal with their ticket. Below each terminal node is a distribution of those passengers who survived and those who didn't.

The tables below show the fates of all passengers based on their ticket type. Now, we can consider the Gini Index associated with the nodes occurring after the split on ticket type from the node labeled "male" in the decision tree.

Male	Survived	Died	total
First	122	19	141
Other	12	1113	1125
total	134	1132	1286

Female	Survived	Died	total
First	101	2	103
Other	69	809	878
total	170	811	981

Table 2: Aftermath of a Tragic Boat Disaster

If we split this dataset on ticket type, we can calculate Gini Impurity for each of the two possible responses as follows, with $i = 0$ indicating survival and $i = 1$ indicating death:

$$G(\text{male, first}) = 1 - \sum_{i=0}^{1} p_i^2 = 1 - ((122/141)^2 + (19/141)^2) \approx 0.233$$

$$G(\text{male, other}) = 1 - \sum_{i=0}^{1} p_i^2 = 1 - ((12/1125)^2 + (1113/1125)^2) \approx 0.021$$

This tells us that knowing that a male was holding a ticket other than first class leads us to a very pure node, that is, one in which we have a substantial amount of information to determine the fate of that passenger. Gini Impurity is related the related notion of entropy and can be seen as measures of how informative the answer to a "yes/no" question is at the split in a tree. Entropy can also be used in determining splits, and can calculated using the following equation, where p_i is the same quantity as defined above:

$$-\sum_{i=0}^{m} p_i log_2 p_i$$

Decision trees, while useful, have their limitations. Firstly, it is possible to grow overly complex trees that overfit the training data [1]. This can happen if we grow our tree in such a way to have only a single observation at each terminal node. The error rate in our training sample would be 0, but we would be very likely to encounter an error in our training set. In such cases, a method known as pruning is required to combat this problem. Cost complexity pruning is a method for considering subtrees of our initial tree model that minimize the *cost-complexity criterion*, which is defined as follows[2]:

Let T_0 be our unpruned decision tree, and define a subtree $T \subset T_0$ to be any tree that can be obtained by collapsing any number of internal nodes. Let $|T|$ denote the number of terminal nodes of our subtree , and let m be an index for them. Define the number of observations at each terminal node as:

$$N_m = |\{x_i \in R_m\}|$$

And the cost-complexity criterion:

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m G_m(T) + \alpha|T|$$

For each α, there is a unique smallest subtree $T_\alpha \subseteq T_0$ that minimizes $C_\alpha(T)$ An estimation $\hat{\alpha}$ is done by using tenfold cross-validation, and the final pruned tree is $T_{\hat{\alpha}}$. In tenfold cross validation, the original training set is paritioned into ten equal subsamples, nine of which are used to train are used to train the model, and one used for validation. This process is repeated 10 ten, each time selecting a different subsample as the test data. The 10 results from are then averaged to produce a single estimation. More details can be found in Breiman et al.[3]

Using our Boat example, we note that that our tree has 8 terminal nodes. If we take a look at the distributions below the nodes, we note that the split on food option is not that meaningful in the case of non-first class males and first class females. So let us consider the subtree T show in Figure 3 formed by removing these splits. We note that the Gini Coefficients of the nodes that precede the removed nodes are already equal to 1, indicating that we already have perfect information about the survival of these passengers. Splitting these nodes further by food option bring us no new information and are therefore useless in classification. Note that $C_\alpha(T) \leq C_\alpha(T_0)$.

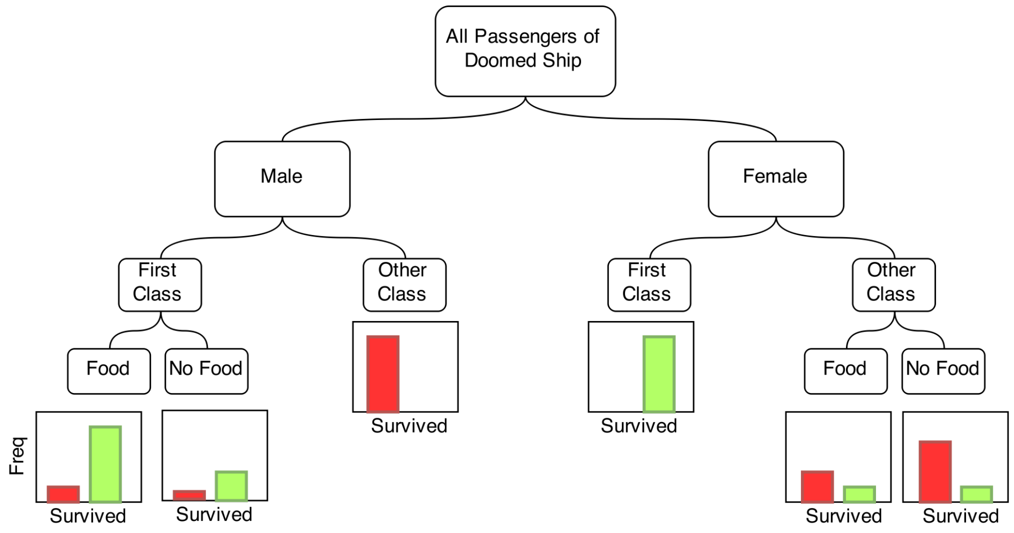

Figure 3: A less complex version of our tree

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. Tree Bagging

Decision trees as discussed in the previous section often suffer from high variance in the case of complex trees, 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, in the case of classification, the most commonly occurring class among the boostrapped predictions.

Given a set of n

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 variable 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.[1] 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 Coefficient. To do so, we take the sum of the amount that the Gini Coefficient has decreased by splitting on a given predictor and take the average over our B trees.

3.2.2. Random Forest

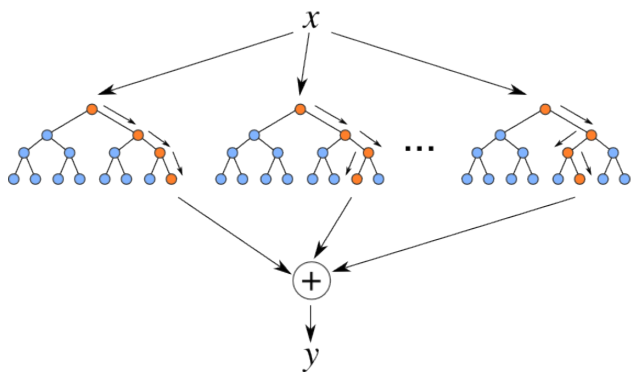

Figure 4: Random Forest. For classification, a prediction for $\mathbf{y}$ is made by majority vote.

The random forest technique is quite similar to taking bagged samples of 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[4].

In this paper, we will be utilizing 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 classification trees 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 model in order to see what parameters would be best to use.

After evaluating our decision trees, we will move onto random forest classification, to see how it improves our predictions.

4.1. Classification Tree

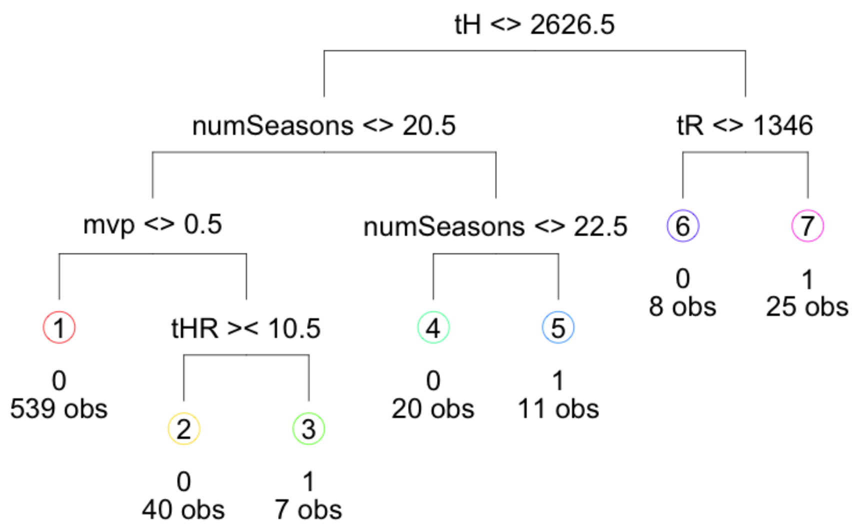

Figure 5: Batting

Using R's rpart package, we can fit a classification tree to our data. To do so, we will split the entire data set into a training and a test set and build our tree using the training data. As our training set, we will take a random sample of about 2/3 of our data. This amounts to ≈ 650 batters. The tree shown in figure 5 was obtained by predicting induction based on all of our features mentioned in section 2. Our model produced a tree with eight splits and 9 terminal nodes and we notice that only 5 variables were used in the

unpruned	actual	
prediction	0	1
0	356	23
1	23	27

pruned	actual	
prediction	0	1
0	369	35
1	10	15

Table 3: Confusion Matrices for unpruned and pruned decision trees

creation of the tree: mvp, tAB, tBA, tH, and R. The misclassification error of this tree on the training data was : $62/650 \approx 9.5\%$. Let's use this model on our test data to see how it performs.

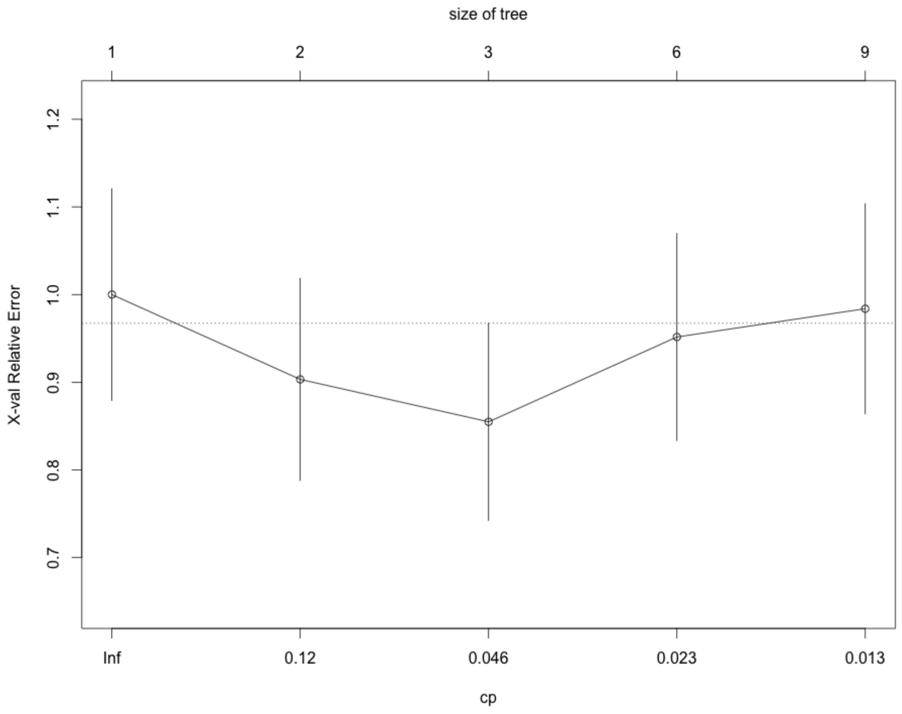

Figure 6: Complexity Parameter of Various Trees

Our model predicted the induction of our test players with a misclassification rate of $1 - 383/429 \approx 10.7\%$. We can perform cross validation on the tree to see if there is a more ideal pruned tree size. Having a look at figure 6, we notice that the cross validation error is minimized around cost complexity parameter (cp) 0.046. We can use this value to prune our original tree in hopes of obtaining a more stable model of size 3. Generating a new tree model having only 3 terminal nodes actual performs slightly better than our original tree, with a misclassification rate of $1 - 384/429 \approx 10.4\%$ on the test data. This data is summarized in table 3.

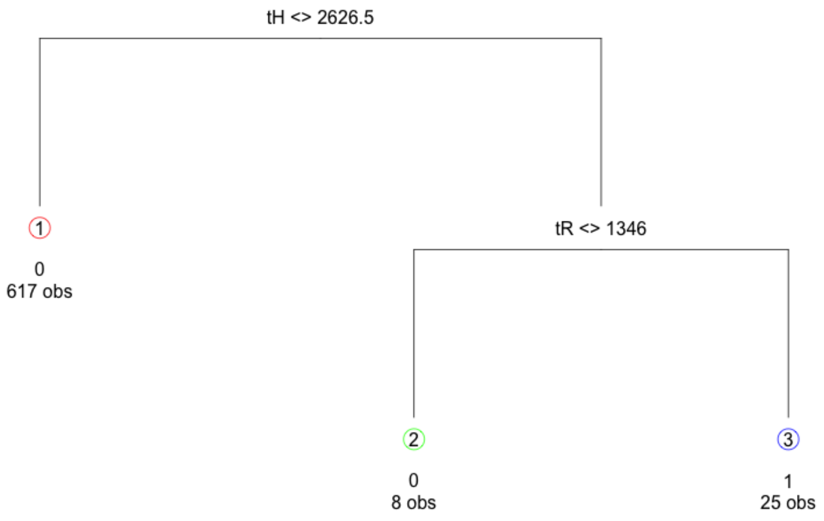

Figure 7: Our Pruned Tree

4.2. Random Forest

5. Conclusions

6. References

[1] Hastie, James, Tibshirani, Witten, An Introduction to Statistical Learning, Springer, 1st edition, 2014.

[2] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer Series in Statistics, Springer New York Inc., New York, NY, USA, 2001.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, Classification and Regression Trees, Wadsworth International Group, Belmont, CA, 1984.

[4] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.