DATA630 9042

Assignment 3: Decision Trees

Sina Alemi

[sina.alemi@gmail.com](mailto:sina.alemi@gmail.com)

Table of Contents

[Introduction](#intro)

[Data Exploration](#explore)

[Decision](#classification) Tree Model

Model [Based](#eval) on Trimmed Dataset

[Conclusion](#conclude)

[References](#references)

[Appendix](#appendix)

[R Script](#script)

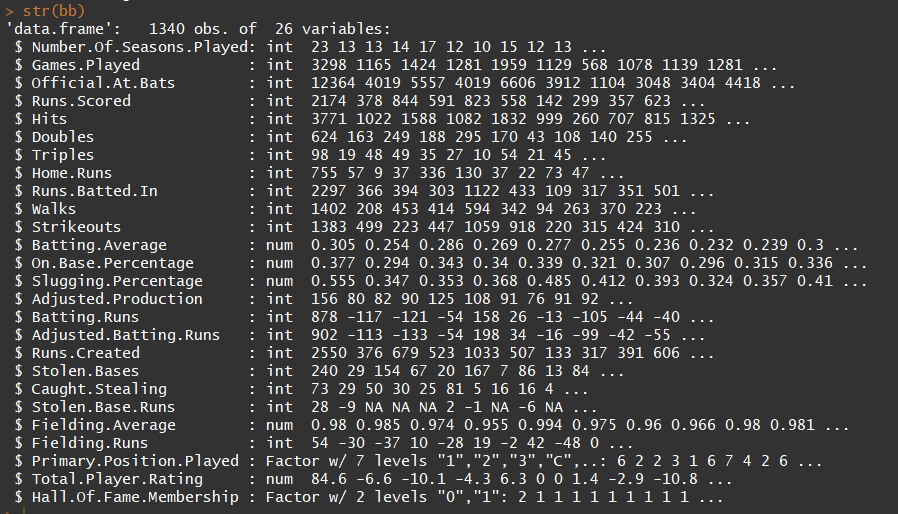
**Introduction**

A decision tree is a supervised analytical tool used to predict the class value of a dependent variable based on inputs of independent variables. Like all supervised analysis methods, the model is built on a “training” data set by attempting to model the relationships between independent and dependent variables. Once the model is built, it can be applied to new data points to predict class values of previously unseen instances. The method is called a decision “tree” due to the structure of the model. First, an input variable is selected to split the data into subsets. This splitting point is referred to as a node. The tree keeps splitting the data further on input variables until a stopping point is reached. The particular algorithm used for this analysis only splits data into two branches. Terminal nodes in the tree are called leaf nodes, and this is where the classes are determined. The most effective tree models have leaf nodes that have high purity. A leaf node has higher purity when the class distribution is more homogeneous. Leaf nodes with low purity are less useful in determining class values because there is a lower likelihood that a data point in that leaf belongs to a specific class. When choosing what variable to split on, decision tree algorithms try to maximize information gain (Tan et al., p. 158). In other words, the tree chooses the variable where splitting maximizes the purity of the lead nodes. Examples of such measures are the GINI index and entropy.

**Data Exploration**

A decision tree model will be built using the ctree package in R. The data set to be used contains statistics of 1340 former professional baseball players. The purpose of the analysis is to predict whether or not the player was inducted into the Baseball Hall of Fame (HoF). To be inducted into the HoF is reserved only for the greatest players in the history of baseball, with the official website of the HoF stating “… election to the Hall of Fame is the highest mark of achievement in the game.” There are currently 317 members (including only 220 players) in the HoF. According to Baseball Reference, there have been 18,853 MLB players since 1876, meaning that a ratio of 220/18853 or 1.17% of players have made it into the HoF. Of the 1340 players included in our dataset, only 124 (9%) of them made it into the HoF.

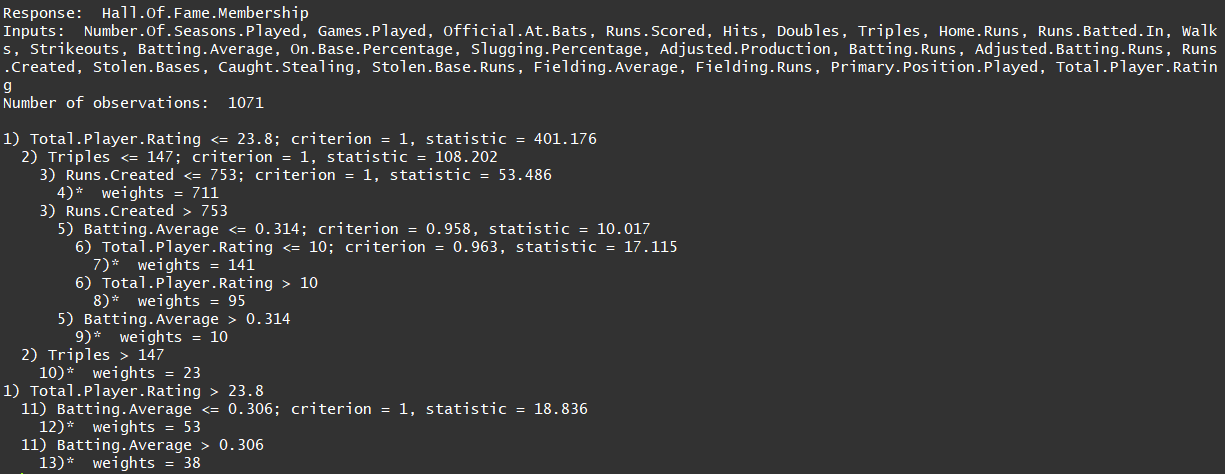
Each tuple in the dataset contains the player’s name along with stats counting various career totals, such as homeruns and hits, along with stats representing ratios of these totals (e.g. ‘batting average’ = ‘hits’ divided by ‘official at bats’), and composite states derived from the aforementioned stats. The player names will be removed as it does not make sense to include in classification. The class variable has a range of 0-2. Values of 1 and 2 both represent a player who made it into the HoF albeit under a different set of rules. To simplify the analysis, values of 2 will be replaced by values of 1. We are only interested in whether the player joined the Hall, not necessarily under which rule set. Finally, a few categorical variables, such as position played and HoF membership, are factorized. The final structure of the dataset can be seen below:

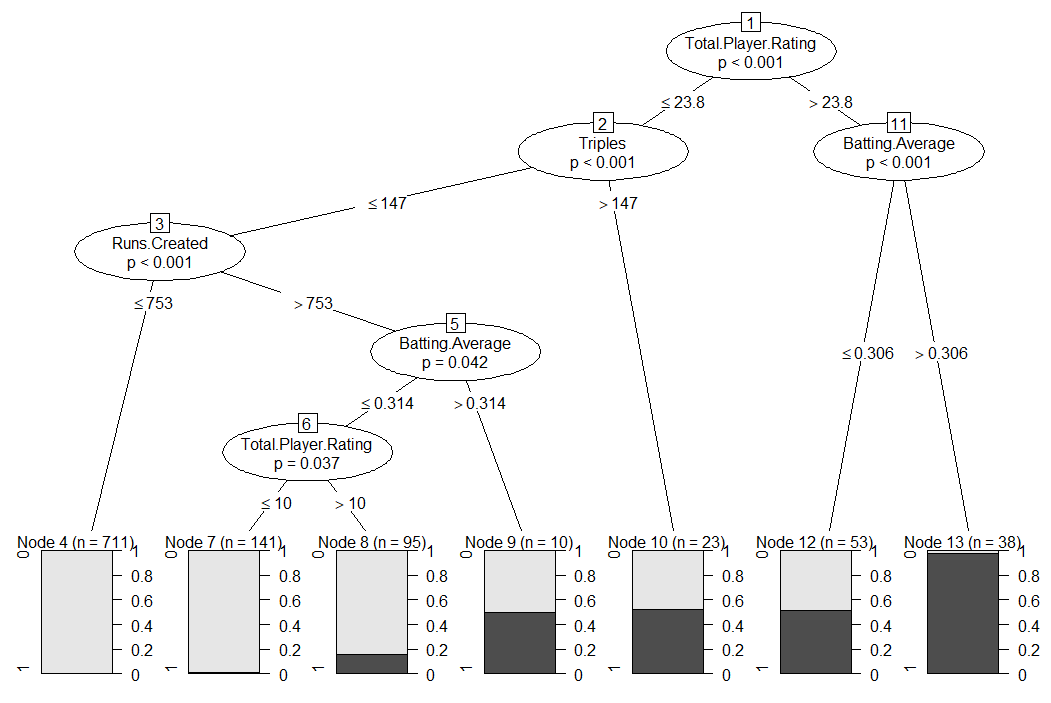


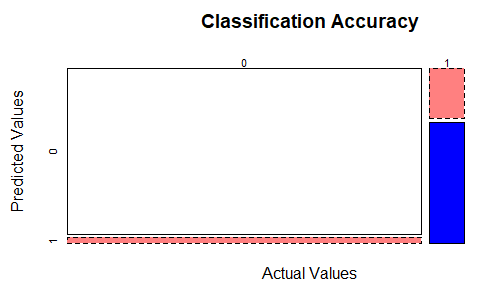
A few of the numeric variables contain missing values. The decision tree algorithm is able to handle these missing values by simply omitting them when evaluating a split criterion. Variables with missing values are likely to have less of an effect on classification.

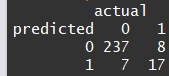
**Decision Tree Model**

Before building any models, the dataset is split into training and test sets with a 70/30 split ratio. The initial model is built on training data using HoF membership as the designated response variable and all remaining variables as inputs. The model’s output is below:



 The model’s plot is easier to follow visually. Although the dataset contains 25 input variables, only 4 are considered as split criteria: total player rating, triples, batting average, and runs created. The first split point (or root node) is total player rating, which is a composite score that aims to measure the player’s overall ability. This intuitively makes sense as a starting point, as the stat’s purpose is to encompass all other measures of the player’s ability. The path branches when the value is 23.8. If the player’s value is greater than 23.8, the path continues to node 11 where batting average is considered. At this point, if the player’s batting average is greater than 0.306, the path proceeds to node 13, which is a terminal leaf node containing 38 observations. There is a near 100% chance that the players in this bucket made it into the HoF. If the player at node 11 has less than a 0.306 batting average, he is placed into node 12 where only 50% of players made the HoF. This is the quickest path to the bottom of the tree ending in a node with very high purity. If the player’s total player rating is less than 23.8, instead of proceeding to node 11 the player path proceeds to node 2 where triples are evaluated. If triples are greater than 147, the tree ends at node 10 where nearly 50% of players are in the HoF. If triples are fewer than 147, runs created are considered as the next split point in node 3. The tree also considers batting average and player rating again at nodes 5 and 6 respectively.

 Players classified into leaf nodes 4 and 7 are almost guaranteed not to be in the HoF. There are 852 players in these two nodes alone, accounting for more than half of the observations. Node 8 has 20% of players in the HoF, and in nodes 9, 10, and 12 have players have a 50% chance. The only way to be relatively certain that a player made the HoF is if that player is classified into node 13. The model’s classification accuracy is measured by creating a confusion matrix of the test data set. The predicted values from the model are compared to the actual values. The matrix is visualized as a mosaic plot.

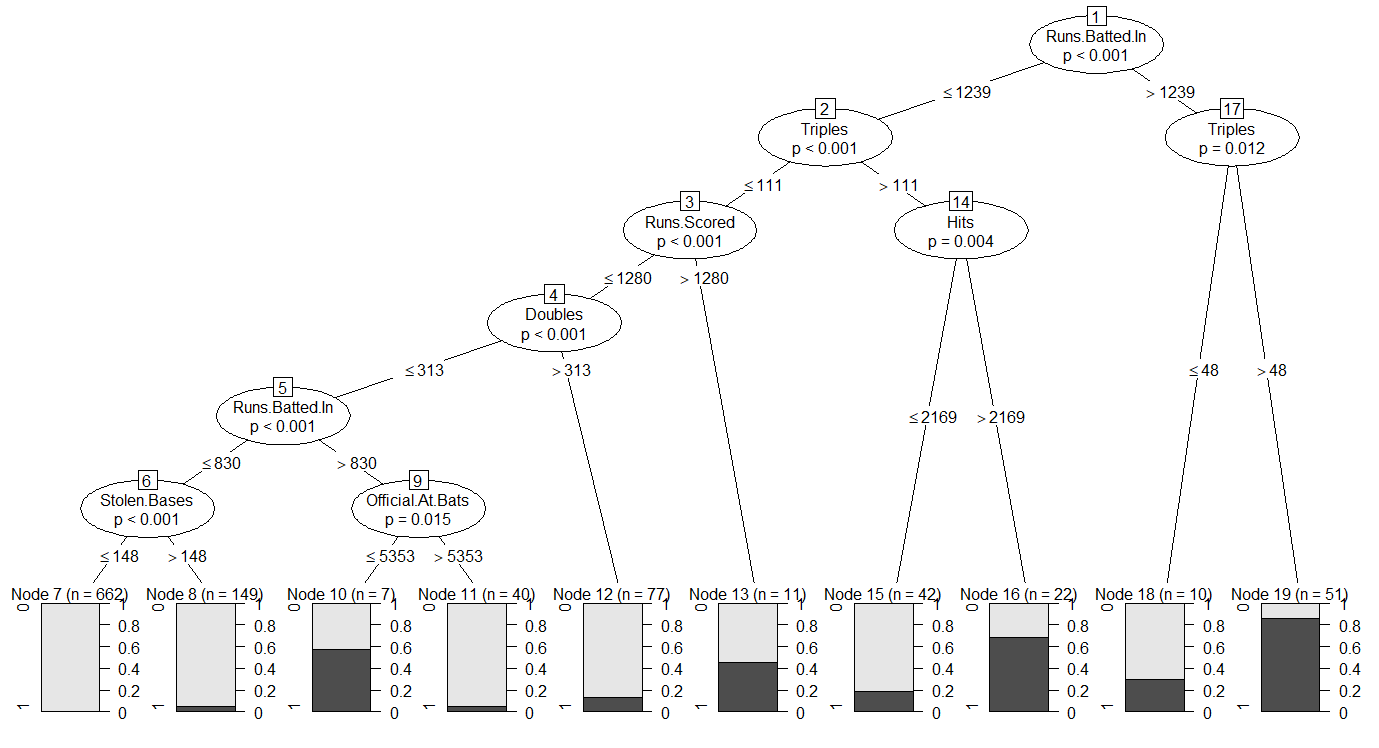


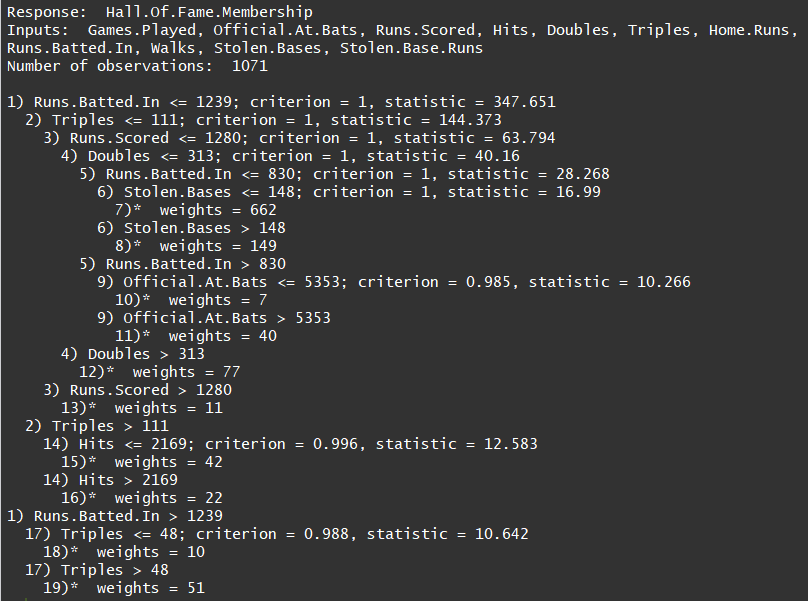
The classification accuracy is (237+17)/(237+17+7+8)= 94.4%. Overall the model did a good job predicting the class values. It mis-classified 7 as being in the HoF and 8 actual HoFers as not having made it. The sensitivity of the model is TP/(TP+FN), in this case 17/(17+8) = 68%. The specificity is TN/(TN+FP), in this case 237/(237+7) = 97%. The model has high specificity but poor sensitivity. This is a sign that the classifier can and should be improved.

While the model is relatively accurate in its classification accuracy, it is reliant on very few input variables. Moreover, the most important input it uses is player total rating, which is a convoluted stat derived from other stats. The average fan does not know how this metric is calculated. The model is not very helpful for a fan who wants to see whether his/her favorite player has a chance at making the HoF. Most fans are more familiar with career totals such as home runs and runs-batted-in (RBIs). Because many of the inputs are related to each other or derived from each other, the algorithm is not finding value in evaluating them. A second model will be built using a trimmed dataset containing only career totals, omitting any ratio and composite statistics such as player total rating.

**Model based on Trimmed Dataset**

The original training data set is trimmed down to only include the following stats: games played, at bats, runs scored, hits, doubles, triples, homes runs, RBIs, walks, stolen bases, and stolen base runs. All other statistics, as well as number of seasons played and position played, are excluded. The same operations will be conducted on the test set to obtain the corresponding trimmed test set. The model based on the smaller training set can be seen below:

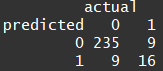
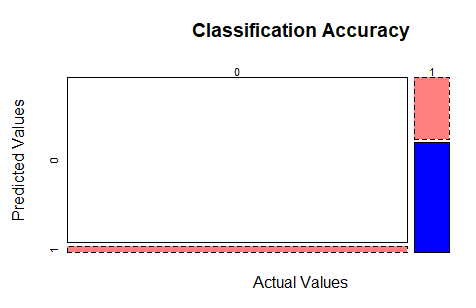


This model uses more splitting criteria and contains more leaf nodes. The first splitting criterion is now RBIs. A run is batted in when a player gets a hit or home run that results in his teammate coming home to score a run. As such, RBIs are generally a good indicator of how well a player comes through in high pressure situations on offense. Triples remain an important criterion, being used as the splitting criterion for both child nodes of node 1.

Like the first model, the majority of instances are classified into one node, node 7. The overall purity of the nodes seems to be higher than the first model. Of the 10 available leaf nodes, only 2 of them (10 and 13) have a roughly 50/50 distribution. The rest are at least 70/30 or better. Node 19 has the highest ratio of players who made the HoF, with over 80% of players succeeding. Based on this decision tree, a fan can simply observe a player’s RBI and triple totals and have a decent idea of whether or not that player will join the HoF. Leaf nodes 18 and 19 only depend on RBIs and triples. Adding hits to the mix allows us to evaluate node 14. A player at node 2 with fewer than 112 triples is not very likely to make the HoF.

Node 10 appears to be an interesting anomaly, albeit containing only 7 instances. Looking at node 3, we see that if a player has fewer than 1280 runs scored, he will be classified into leaf node 7, 8, 10, 11, or 12. Of these leaf nodes, all have a distribution of less than 20% of players making it into the HoF. The lone exception is node 10 with a roughly 50/50 split. It’s interesting to note that if a player at node 9 has more than 5353 at bats, he will almost certainly not make the HoF. Having fewer than 5353 at bats gives him a 50% chance. In general, a player’s number of at bats increases steadily the longer he plays. It is not a performance measuring statistic like home runs; it simply counts the number of times a player has gone up to the plate to hit. A likely explanation for this is that players at leaf node 11 had longer careers as those in node 10, but amassed a similar amount of statistics. In other words, players in node 10 had the same success as those in node 11 in a smaller time window.

The model’s classification is evaluated on the smaller test set. The measures are similar to the original dataset’s, with an accuracy of 93%, sensitivity of 64% and specificity of 96%. This model was not any more accurate than the first, but it is based on more readily available statistics and thus should be more useful for the average fan.



**Conclusion**

To summarize, a decision tree model was built using aggregate, ratio, and composite statistics and evaluated for accuracy, specificity, and sensitivity. The model achieved high accuracy and specificity, but poor sensitivity. The model also took many variables as input but only used a small number of them in the model. I suspected I could build a model just as effective with fewer inputs. The second model was built using only the aggregate statistics. This new model had more paths and nodes, but was built from a smaller dataset. The accuracy, specificity, and sensitivity levels remained roughly the same as the first model. Although the new model was not necessarily a more effective classifier, it is usable with less readily available input statistics.

There are a few downsides to using the decision tree method. Because our dataset is mostly dominated by one class - those players not in the HoF - the tree may be a bit biased towards that class value. Indeed, the low sensitivity metrics of both models suggest that they are not good at classifying true positives. In order to achieve a better sensitivity measure, we may use a dataset containing similar numbers of tuples in each class value. Because there are only 220 players in the HoF, this dataset would be limited in size. Having a smaller dataset would mean it is less representative of the overall baseball player population, and thus it made reduce the classification accuracy.

There is a limitation to this analysis that is outside of any statistical method used and it has to do with the data itself. Baseball players do not automatically join the HoF when their statistics cross a certain threshold. Players are voted in by a select group of people, each with their own feelings, preferences, and biases. As a result, their methodology for choosing players is not entirely consistent over time. Considering that the first players entered the HoF in 1936, the qualities that made a player worthy of entering the HoF may have changed since that time.

**References**

Hall of Famers. (n.d.). Retrieved October 26, 2017, from <https://baseballhall.org/hall-of-famers>

Major League Baseball & MLB Encyclopedia. (n.d.). Retrieved October 26, 2017, from <https://www.baseball-reference.com/leagues/index.shtml>

Tan, P., Steinbach, M., Karpatne, A., & Kumar, V. (n.d.). *Introduction to data mining*. New York, NY: Pearson Education.

**A****ppendix**

R Script

#DATA630 9042

#Assignment 3

#By: Sina Alemi

library(party)

#dataframe creation

setwd("D:/school/DATA630/module 4/ass 3")

bb<-read.csv(file='MLBHOF.csv',head=TRUE, sep=",")

str(bb)

summary(bb)

#data pre-processing

#removing unncessary variables

bb$Name<-NULL

#replacing values of '2' with '1' in Hall of Fame Membership

bb$Hall.Of.Fame.Membership<-replace(bb$Hall.Of.Fame.Membership, bb$Hall.Of.Fame.Membership==2,1)

#factorizing some variables that should be treated as categorical

bb$Hall.Of.Fame.Membership<-factor(bb$Hall.Of.Fame.Membership)

bb$Primary.Position.Played<-factor(bb$Primary.Position.Played)

#splitting data into training and test sets

set.seed(1234)

ind <- sample(2, nrow(cars), replace = TRUE, prob = c(0.7, 0.3))

train.data <- bb[ind == 1, ]

test.data <- bb[ind == 2, ]

#MODEL 1

#----------

#Decision tree model

model<-ctree(Hall.Of.Fame.Membership~., data=train.data)

print(model)

#Decision tree plot

plot(model)

#Classification Accuary

#A confusion matrix is built to compare values of predicted class vs actual class

#test data evaluation

table(predict(model, test.data), test.data$Hall.Of.Fame.Membership, dnn=c("predicted", "actual"))

#mosaic plot of test data evaluation

mosaicplot(table(predict(model, test.data), test.data$Hall.Of.Fame.Membership),shade=TRUE, xlab="Actual Values", ylab= "Predicted Values", main='Classification Accuracy')

#MODEL 2

#-------

#creating smaller training/test datasets from the original sets

smalltrain<-train.data[,2:10]

smalltrain$Stolen.Bases<-train.data$Stolen.Bases

smalltrain$Stolen.Base.Runs<-train.data$Stolen.Base.Runs

smalltrain$Hall.Of.Fame.Membership <- train.data$Hall.Of.Fame.Membership

smalltest<-test.data[,2:10]

smalltest$Stolen.Bases<-test.data$Stolen.Bases

smalltest$Stolen.Base.Runs<-test.data$Stolen.Base.Runs

smalltest$Hall.Of.Fame.Membership <- test.data$Hall.Of.Fame.Membership

#model for smaller set

smallmodel<-ctree(Hall.Of.Fame.Membership~., data=smalltrain)

print(smallmodel)

plot(smallmodel)

#evaluating the small model

#test data evaluation

table(predict(smallmodel, smalltest), smalltest$Hall.Of.Fame.Membership, dnn=c("predicted", "actual"))

mosaicplot(table(predict(smallmodel, smalltest), smalltest$Hall.Of.Fame.Membership),shade=TRUE, xlab="Actual Values", ylab= "Predicted Values", main='Classification Accuracy')