University of Waterloo

Faculty of Engineering

Department of Electrical and Computer Engineering

ECE 356 Lab4 - Data Mining

Prepared by

Tianpeng Hong

UW Student ID Number: 20567046

UW User ID: t22hong @edu.uwaterloo.ca

Ziyan Liu

UW Student ID Number: 20608393

UW User ID: z363liu @edu.uwaterloo.ca

Zuju Sun

UW Student ID Number: xxxxxxxx

UW User ID: xxxxxxxx @edu.uwaterloo.ca

31 March 2019

# Analysis and results

## Feature Selection

Individuals in the relation HallOfFame fall under several categories, namely Player, Manager, Umpire and Pioneer/Execute. The category of Player takes up the largest portion of the record base being 96.99%. As a result of this finding, it is safe to assume that features should be selected to mainly reflect characteristics of players as they make up the vast majority of the records in the HallOfFame table.

The initial assumption is that the Batting and Pitching relations contain batting and pitching statistics of players, thus are considered as strong evidence indicating whether players are elected or nominated to be in the hall of fame. To check the integrity of such assumption, a SQL query is written to extract important features from these two tables. The combination of all columns from the Batting table, except playerID, yearID, stint, teamID and lgID, is considered to the best possible explanation of the batting performance of a given player. Similarly, all columns from Pitching table except for playerID, yearID, stint, teamID and lgID are considered as feature candidates.

For each pair of a given feature and a player, the average of such feature is calculated by taking into account every year up until the year that the player is elected/nominated. For example, the following sub-query calculates the average number of homeruns a player has from all records in Batting table with yearID less than the year they were actually nominated/elected:

(select avg(Batting.HR) from Batting where HallOfFame.playerID = Batting.playerID and

HallOfFame.yearID >= Batting.yearID) as HR

The complete query is constructed by calculating average of each feature as explained above and can be found within query.sql file.

## Impurity measures, Accuracies and Confusion Matrixes

Once the data has been extracted into the query\_result.csv file, two classification trees based on Gini and Entropy measurements are generated within each iteration. Take one iteration as an example, the accuracies of predications and confusion matrixes are shown as below:

Gini accuracy is 91.94711538461539%

Confusion matrix =

[[747 23]

[ 44 18]]

Entropy accuracy is 93.02884615384616%

Confusion matrix =

[[755 15]

[ 43 19]]

The accuracy is obtained by called the accuracy\_score function provided by Python Sklearn library, which essentially calculates the ratio of the number of correct predictions over the number of all predictions. The resultant accuracies over the five runs range from 91% to 94% and the complete set of accuracies is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1st Iteration | 2nd Iteration | 3rd Iteration | 4th Iteration | 5th Iteration |
| Gini Index | |  |  | | --- | --- | | |  | | --- | | 91.83% | | | |  | | --- | | 92.67% | | 91.47% | |  | | --- | | 92.19% | | |  | | --- | | 94.23% | |
| Entropy | |  |  | | --- | --- | | |  | | --- | | 91.83% | | | |  | | --- | | 92.79% | | |  | | --- | | 91.71% | | 92.43% | |  | | --- | | 93.51% | |

Since the parameter “random\_state” is not fixed in the model training part, the order of features, composition of training and validation sets are different for each iteration. Therefore, accuracy score may differ slightly for an individual iteration.

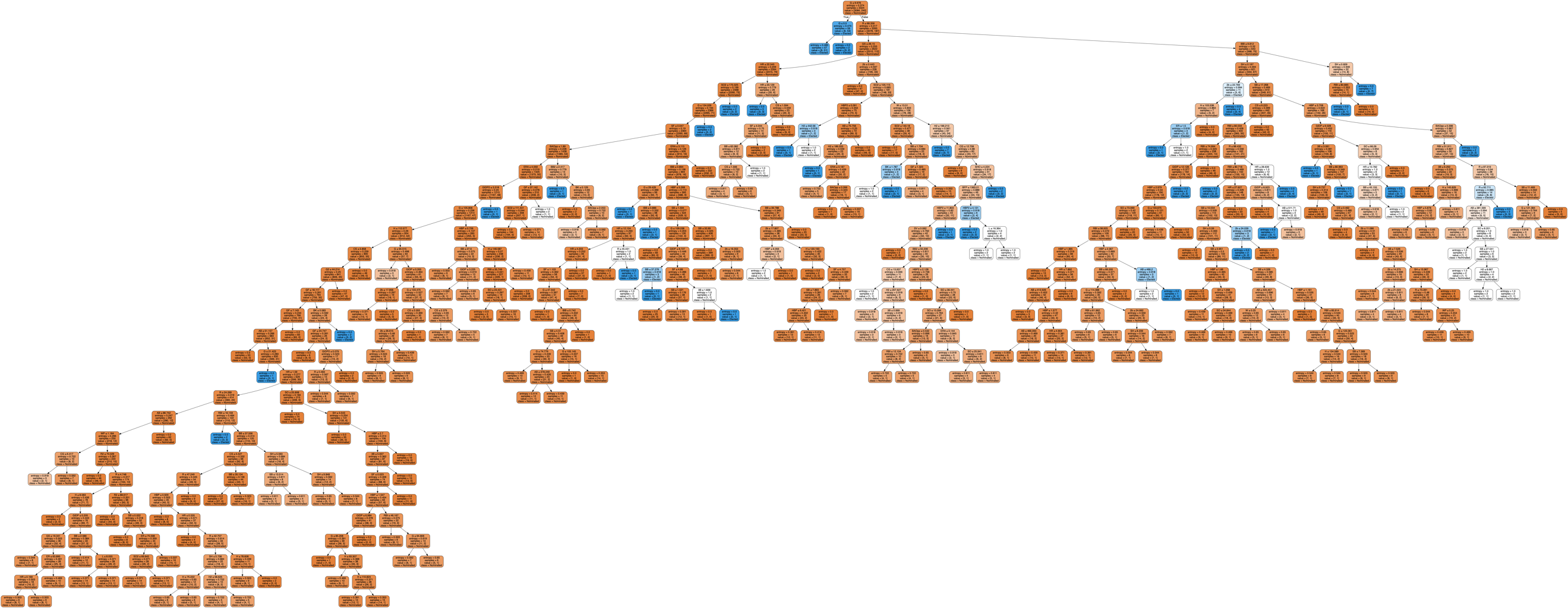
To justify the effectiveness of chosen features, one can look into the confusion matrix. For example, the confusion matrix of the Gini classification tree can be interpreted as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Actual Class 1 | Actual Class 2 | Total |
| Predicated Class 1 | 89.78% | 2.76% | 92.54% |
| Predicated Class 2 | 5.288% | 2.163% | 7.451% |
| Total | 95.068% | 4.923% | --- |

The True Positive rate is 89.78% and the True Negative rate is 2.163%, whereas the False Negative rate is 5.288% and the False Positive rate is 2.76%. By looking at the total percentage, it can be inferred that the Gini classification tree predicts Class 1 92.54% of the time and it actually occurs 95.068% of the time. There is an under-lifting of 2.528%. As for Class 2, the Gini classification tree predicts Class 1 7.451% of the time and it actually occurs 4.923% of the time. There is an over-lifting of 2.528%. In a nutshell, the values of over-lifting and under-lifting is rather small, thus we conclude that the list of selected features is a good choice in terms of determining whether a player has been elected or nominated into the hall of fame.

## Tree Snapshot

To show an example of generated classification tree, a snapshot of an entropy classification tree is shown as follows. Details of the tree can be examined by opening 0\_entropy\_snapshot.png and zooming in.



# Comparison

While Gini index and Entropy are both viable ways to build a classification tree, they use different mechanisms internally. Gini index is a measure of misclassification whereas entropy is a measure of information gain.

Despite their difference in terms of implementations, there isn’t a lot of differences in the accuracies of resultant trees as shown below. It’s obvious that the shapes of two curves almost follow each other, with the only noticeable different being the that Gini index tree has higher accuracy in the last run. Another difference, while not represented in this graph, is that building a Entropy classification tree usually takes longer than a Gini index tree as Entropy calculation involves logarithm whereas Gini index does not.