Lab 4 - Data Mining

Faculty of Engineering
Department of Electrical and Computer Engineering

December 07, 2020

Manthan Shah (205658832 | mskshah@uwaterloo.ca)
Stephanie Meler (20662056 | smeler@uwaterloo.ca)
Eric Song (20658437 | ejsong@uwaterloo.ca)

1 Instance count

For Task A, it was determined that an "arbitrary Player" was someone who had either debuted or was mentioned in the HallOfFame as a Player. Thus, someone who was a former Player turned Manager would be considered, whereas a Hall Of Fame inductee for Umpire or other non-Player role would not be counted. Similarly, for Task B, it was assumed that a Hall of Fame nominee would always have an entry in the HallOfFame table, with the Player category. The feature extraction queries for task A and B can be in the Appendix A and B respectively.

Using SQL, the number of instances of the total count of both task A and B, as well as the class counts were computed (Table 1).

Table 1. Total instance count and class count for Task A and B

	Task A	Task B
Total Instances (#)	18936	1208
Class 0 Instances (#)	17717	958
Class 1 Instances (#)	1219	250

1.1 Task A

Total Instances: 18936

```
SELECT count(distinct playerID) as totalInstances from HallOfFame where
category = 'Player';
```

Total class 0 (not nominated): 17717

```
SELECT count(distinct Master.playerID) as class0Instances FROM Master LEFT JOIN
HallOfFame on Master.playerID = HallOfFame.playerID WHERE (category = 'Player'
OR debut != '') AND IF(inducted = 'Y' or inducted = 'N', 1, 0)=0;
```

Total class 1 (nominated): 1219

```
SELECT count(distinct Master.playerID) as class1Instances FROM Master LEFT JOIN
HallOfFame on Master.playerID = HallOfFame.playerID WHERE (category = 'Player'
OR debut != '') AND IF(inducted = 'Y' or inducted = 'N', 1, 0)=1;
```

1.2 Task B

Total Instances: 1208

```
SELECT count(distinct playerID) as totalInstances FROM HallOfFame WHERE
category = 'Player';
```

Total class 0 (not inducted): 958

```
SELECT count(*) as class0Instances FROM (SELECT DISTINCT playerID, inducted
FROM HallOfFame  INNER JOIN  (SELECT playerID as inductedPlayerID,
max(inducted) as wasInducted FROM HallOfFame WHERE category = 'Player' GROUP BY
inductedPlayerID) AS InductedHOF ON HallOfFame.playerID =
InductedHOF.inductedPlayerID and HallOfFame.inducted = InductedHOF.wasInducted)
as HallOfFame WHERE IF(inducted = 'Y', 1, 0) = 0;
```

Total class 1 (inducted): 250

```
SELECT count(*) class Instances FROM (SELECT DISTINCT playerID, inducted FROM
HallOfFame  INNER JOIN  (SELECT playerID as inductedPlayerID, max(inducted) as
wasInducted FROM HallOfFame WHERE category = 'Player' GROUP BY
inductedPlayerID) AS InductedHOF ON HallOfFame.playerID =
InductedHOF.inductedPlayerID and HallOfFame.inducted = InductedHOF.wasInducted)
as HallOfFame WHERE IF(inducted = 'Y', 1, 0) = 1;
```

2 Data Cleansing

The first data cleansing issue was with players who were missing debut dates and final game dates. Although normally, a missing debut date would denote a non-player, some Hall of Fame inductees from the Negro league had a corresponding HallOfFame entry, but not a complete Master entry. Thus, these values were set to 0 for all null values. A mean or median would not have made sense in the context of distinct debut and final game dates. Furthermore, another placeholder value such as 1800 or 9999 that is fully outside the range of valid dates in the database would have the same effect as 0. It was assumed that the decision tree (DT) would be able to identify a debut/final game year of 0 and classify accordingly.

The next issue was with the Batting table statistics, such as Hits and RBI. These were important statistics that defined a player's performance over the year (since each entry denotes one year), and were missing from the Negro League Hall of Fame players, as well as players who were only pitchers. Consequently, a mean or median would have been invalid, since Hall of Fame nominees would have the same values compared to the rest of the player base (along with pitchers). Thus, a value of 0 was assigned instead of null, with the same assumption that the DT would be able to distinguish this from other valid data.

A similar approach was taken for the Pitching table statistics, where a large quantity of data was missing for players who were batters but not pitchers (which is the majority of players). These nulls were set to 0 for the same reason as the Batting table, since a mean/median/max would not be appropriate and could confuse the DT splitting algorithm.

Lastly, data cleansing was needed for the Awards table, where the feature dubbed "awardsCount" would be null if a player did not have any recorded awards. Although some affected players included the Negro League Hall of Famers, the majority was regular players who had never won awards. A value of 0 was the best replacement, since players who had never won any awards would accurately have 0 awards. Although a League specific alternative value was considered, it was ultimately decided that these players should also be assigned 0 since it was unknown whether these players won awards or not. For reference, there were many Hall of Fame nominees/inductees who also did not win any awards, making this a fair assumption.

3 Selection of features

Upon investigation and research conducted regarding induction into the Hall of Fame, the criteria for nomination include that a player must have played at least ten seasons and the player must have been retired for at least five seasons before being nominated [1]. These election criteria were used to form an initial basis for the feature selection. This meant selecting the debut and finalGame as features from the Master table. Overall, it was found that besides the criteria listed above, there were no other defined criteria or features which would indicate whether a player would be nominated or inducted into the Hall of Fame. It is stated that voting is dependent on the player's record, playing ability, integrity, sportsmanship, character and contributions to the team(s) on which the player played, which leaves a lot of ambiguity into what other features may be used to inform nomination and induction [2].

To inform any other features that could be used for Task A and Task B, an investigation regarding the most important batting & pitching statistics was conducted. This was done to try and match how the voting committee may evaluate a player's record, playing ability and contributions to their team.

Initially, in terms of batting, we used all the columns from the batting table. This was found to be an inefficient strategy, as even though the accuracy was high (~93%), the precision and recall were relatively low (~50% each). We trimmed down the features from the Batting table to hits (H), homeruns (HR), and runs batted in (RBI). We decided to use these, because historically good batters that made it into the hall of fame, excelled in each of these categories (e.g. Ty Cobb, Babe Ruth, etc). The number of hits represents the number of times a player was able to get on base without getting out (strikeout, popout, etc) [3]. This is a valuable feature because a good batter will have a high batting average, and thus high number of hits throughout their career. Next, home runs occur when a batter hits a ball out of the baseball stadium, resulting in instant runs for their team [3]. A good batter will generally hit home runs often (e.g. 20+ per season). However, there are also utility batters that excel at getting on base, but do not hit home runs often. Lastly, the run batted in (RBI) represents how many runs a batter is able to drive home, when there are players on base [3]. The most valuable aspect for a team is the ability of a batter to get a hit with players on base (ie. batting well under pressure), as it results in runs for the team. So, hall of fame level batters have a large number of RBIs throughout their career, due to their inherent utility.

There are also various other columns in the Batting table that we did not use. This is because they likely did not provide any meaningful value for the decision tree. For example, columns such as the number of doubles (2B) and number of triples (3B) are just finer-grained statistics of the total number of hits of a player, so they aren't useful on their own as additional features [4]. Also, other columns such as stolen bases (SB), caught stealing (CS), walks (BB), hit by pitch (HBP), and many others provide insignificant measures [3]. This is because these are aspects of baseball that aren't critical to the success, and eventual nomination of a batter into the hall of fame.

In terms of pitching, we initially used all the columns from the pitching table as well. This was also found to be an inefficient strategy and yielded low accuracies (~80%) and overall lower precision and recall as well (~40%). This is because there are far fewer pitchers in the hall of fame, then there are batters based on the dataset. We trimmed down the features from the Pitching table to be hits (H), earned runs (ER), homeruns (HR), walks (BB), strikeouts (SO), and games started (GS). These were chosen because in baseball, these are the most meaningful measures for the performance of a pitcher [5]. The hits (H) refer to the number of times a pitcher lets a batter get on base without getting out (ie. a hit) [3]. Good pitchers will surrender fewer hits over their career, as hits eventually lead to runs which is bad for pitchers. Earned runs (ER) refer to how many runs a pitcher gives up. The better a pitcher, the fewer runs they give up, thus the more valuable they become. Home runs (HR) refer to how many home runs a pitcher gives up [3]. The better a pitcher, the fewer runs they give up. Walks (BB) refer to how many times a pitcher lets a batter get on base due to a lack of strikes [3]. The more walks a pitcher gives up, the more batters get on base, and eventually score. The strikeouts (SO) refer to how many times a pitcher gets a batter out from strikes (ie. not out due to a fielder) [3]. Games started (GS) refers to how many games a pitcher has started and pitched in [3]. The more games a pitcher pitches, the longer their service, and potential utility to teams. Based on these pitching features, an ideal hall of fame pitcher would have a low number of H, ER, HR, and BB, and high number of SO and GS.

The other features in the pitching column do not yield any additional, hall of fame level details about a pitcher. For example, wins (W) and losses (L) of a pitcher depend largely on their team's batters [3]. There are many instances of hall of fame level pitchers that have lots of losses (e.g. Felix Hernandez), even though they have extremely good core pitching statistics (features described above).

The last feature we used was the number of awards a player has received throughout their career. This is because good players (both batters and pitchers), will likely receive some form of recognition throughout their career. There are a large number of awards given out every season, so the likelihood of a strong player receiving one over their tenure is high [4].

We considered other tables, but decided they did not yield better Decision Tree classifier performance. This was likely because other tables provided statistics that were not meaningful for hall of fame purposes. For example, the Fielding table is not very useful, because all batters are fielders, and batters are primarily judged on their batting abilities. It is beneficial for a batter to also be a good fielder, but it is not a defining characteristic of a hall of fame level player.

In our overall design, we used the debut and finalGame columns from the Master table, H, HR, and RBI from the Batting table, H, ER, HR, BB, SO, and GS from the Pitching table, and a count of the awards for each playerID from the AwardsPlayers table.

4 Sampling Training & Testing

The dataset obtained from the output file was split into two subsets: training and testing. The training subset was used in order to build the classification model and the testing subset was used for evaluating the performance of the developed model, given some unknown or new data to evaluate. The python library, Sklearn (or Scikit-learn) was used, and particularly the train_test_split function to split the output dataset into the two mentioned subsets, which splits the given dataset automatically. With this function, the size of the testing set could be specified and was chosen to be 0.2, which represented 20% of the total dataset that would be used to evaluate the model developed. By default, the function creates random partitions for the two subsets so each individually is a suitable representation of the dataset. Because the dataset was sufficiently large (over 18,000 rows for Task A and over 1200 rows for Task B), it is possible to cover all common and uncommon cases in the domain of the datasets. Using the train-test split, the model could be trained quickly and the 80-20 ratio used for training and testing subsets was selected to follow the general recommendations for classification.

5 Validation results

For task A, the debut, and finalGame from Master table, H, HR, and RBI from Batting table, H, ER, HR, BB, and SO from Pitching table, and count of awards per playerID from AwardsPlayers table were used. An overall accuracy of 96%, precision of 72%, and recall of 65% were achieved, as described in Table 2. The dataset for this task was much larger because all players were included (not nominated, nominated, and/or inducted).

For task B, the debut, and finalGame from Master table, H, HR, and RBI from Batting table, GS, H, BB, and SO from Pitching table, and count of awards per playerID from AwardsPlayers table were used. An overall accuracy of 88%, precision of 74%, and recall of 70% were achieved, as described in Table 3. The dataset for this task was much smaller because it only consisted of players that had been nominated and/or inducted.

It is important to consider that accuracy is not a meaningful measure on its own. This is because there are many more players in the dataset that have never been nominated to the hall of fame, and even fewer players (of those nominated), that were inducted into the hall of fame. Thus, the true positive rate and precision are important indicators of how good the classifier is at determining players' nominations and inductions into the hall of fame. Since there are much more negative samples in the dataset, the specificity is very high for both task A and task B. This makes sense as it is much more likely for a randomly sampled player, assigned to the negative case (ie. not nominated or inducted to hall of fame) to be correct. Ideally, the higher the precision and recall, the better the classifier. Thus, for this dataset, the precision and recall metrics are much more important because they demonstrate the performance of the classifier at classifying positive instances (ie. nominated or inducted into hall of fame) correctly, which occurs at a much smaller scale than negative instances.

Table 2 - Task A Confusion Matrix Results

TN	FP	FN	TP	Accuracy (%)	Precision (%)	Recall (%)	Specificity (%)
3450	68	94	176	96	72	65	98

Table 3 - Task B Confusion Matrix Results

TN	FP	FN	TP	Accuracy (%)	Precision (%)	Recall (%)	Specificity (%)
176	13	16	37	88	74	70	93

6 Hyperparameter Tuning

Firstly, the available hyperparameters for sklearn.tree.DecisionTreeClassifier were found on the Scikit-learn documentation [6]. Using this list, the efficacy of each hyperparameter was manually explored on a baseline CSV generated from section D. The default value was the starting point of iteration, and increased in a quadratic fashion until a "reasonable" end value was tried. For example, a reasonable end value for min_samples_leaf was 0.1, which would mean that each leaf must have 10% of the total sample population - thus causing a maximum of 10 leaves in the whole decision tree.

For each iteration, the delta change in test accuracy, positive value precision, and positive value recall was observed compared to the previous iteration. The trend was also considered, seeing if the next step caused a better, neutral, or worse outcome. With this manual method, it was determined that the only hyper parameters that produced a better outcome (from the default values) were max_depth and min_impurity_decrease. These results were sensible, since max_depth directly forced the DT to split "smarter" and reduced the chance of overfitting, whereas min_impurity_decrease could force the DT to use splitting criteria that produced a minimum change in sample impurity (i.e. balance between better splits and bigger splits).

Next, the effects of max_depth and min_impurity_decrease was explored using sklearn.model_selection.GridSearchCV. This object would exhaustively compare all combinations of the provided hyperparameters, and automatically select the best values and best classifier version. With this, max_depth was searched from [5,20] and min_impurity_decrease was searched from [0, 0.05] with a step size of 0.00002; these ranges were determined empirically from the previous manual step, to show the best improvement. Overall, the best (non-default) hyperparameters were tabulated in Table 4. Unfortunately, not all hyperparameters could be tested exhaustively via GridSearchCV, due to the exponential increase in computation time.

Table 4. Optimal hyperparameter settings for each task

	Task A	Task B
max_depth	9	5
min_impurity_decrease	0.00026	0.00072

With the hyperparameters from Table 4, the overall increase in testing accuracy and other statistics were tabulated in Table 5. As seen in the table, there was a significant increase in the positive prediction ratios, with a negligible change in the negative prediction ratios. It also showed that Task B improved more drastically than Task A, which may be due to the smaller sample size.

Table 5. Change in test accuracy statistics with the hyperparameters from Table 4

		Task A	Task B
Positive	Precision	+0.19	+0.35
	Recall	+0.05	+0.16
Negative	Precision	0	+0.01
	Recall	+0.01	+0.07
Accuracy		+0.01	+0.07

As mentioned in section 3, our initial solution consisted of choosing all columns from both the Batting and Pitching table, which yield relatively low performance compared to later iterations. After the initial solution was attempted, various solutions were iterated upon to evaluate which subset of features would yield the most favourable results. An alternative solution that was attempted entailed only using four features : awardsCount, debut, death year and salary. This alternative solution was attempted because it was seen that because many players nominated and inducted into the hall of fame were not strictly batters or pitchers, but a combination of various players. Thus, choosing specific features could sway the model and result in lower performance. Thus, the four features included in this alternative solution were not dependent on whether the player was a batter or a pitcher, but entailed features that could encompass any player. This however still did not yield exceptional performance, but better than the initial solution attempted. Upon investigation, it was found that many players did not have salary information available, thus the feature was still not appropriate to be a representation of all players.

Thus, iterations were conducted, using only the most important features for batting and pitching performance (as discussed in section 3). The resulting solution for Task A consisted of the features: debut and finalGame from the Mater table, H, HR and RBI from the Batting table and finally H, ER, HR, BB and SO from the Pitching table, in addition to award count found in the AwardsPlayers table. For Task B, the debut and finalGame from the Master table, H, HR and RBI from the Batting table, GS, H, BB and SO from the Pitching table were used, in addition to the award count from the AwardsPlayer table.

Initially, when roughly 20 features (with ~8 each from the Batting and Pitching tables) were being fed into the decision tree, a very poor positive precision and recall were achieved at around 0.4. Furthermore, the tree depth was consistently >25. It was very evident that overfitting was occurring, since the training statistics showed positive precision/recall values of around 0.9, and prior machine learning experience saying that typical decision tree depths greater than 15 were typically overfitting. Furthermore, an initial test with the hyperparameter max_depth being set to 5 and 10 showed an immediate improvement of 0.2 points (for positive precision/recall) in both cases. We then trimmed the features and used the debut, finalGame, deathYear, and awardsCount. This solution significantly reduced the size of our DT, and increased the precision drastically, but reduced the recall. This was likely because these features were player type agnostic (did not skew towards batter or pitching statistics). Our final set of features were as described in section 5. These were optimal as they maximized the accuracy, precision, and recall, while also achieving a "balanced" DT relative to the aforementioned solutions.

This overfitting problem was solved primarily by limiting the max_depth to a maximum of 20. Although optimal values via GridSearchCV were usually from 5-10, the max of 20 gave a safe range for GridSearchCV to explore. Another method that was tested was to set max_leaf_nodes to roughly half the overfitted tree's leaves, which was roughly 30. Unfortunately, this had negligible results on the accuracy metrics, even at higher or lower values. It was ultimately discarded in favor of max_depth, since a small depth had a similar effect of dropping the leaf node number drastically.

Using the principle of Occam's Razor, "pruning" the tree to its most essential leaves/depth helped to prevent overfitting. An estimated optimal value was found by increasing/decreasing the values until the accuracy statistics started regressing. Using GridSearchCv (detailed previously in this section), the true optimal values were found, which balance between underfitting and overfitting.

7 Decision Tree Illustrations

The decision tree for task A, with the features identified in section 5, and optimal hyperparameters in section Y is shown in Figure 1. The decision tree has a depth of 5, 20 leaf nodes, and branching factor of 2.

The decision tree for task A, with the features identified in section 5, and optimal hyperparameters in section Y is shown in Figure 2. The decision tree has a depth of 9, 30 leaf nodes, and branching factor of 2.

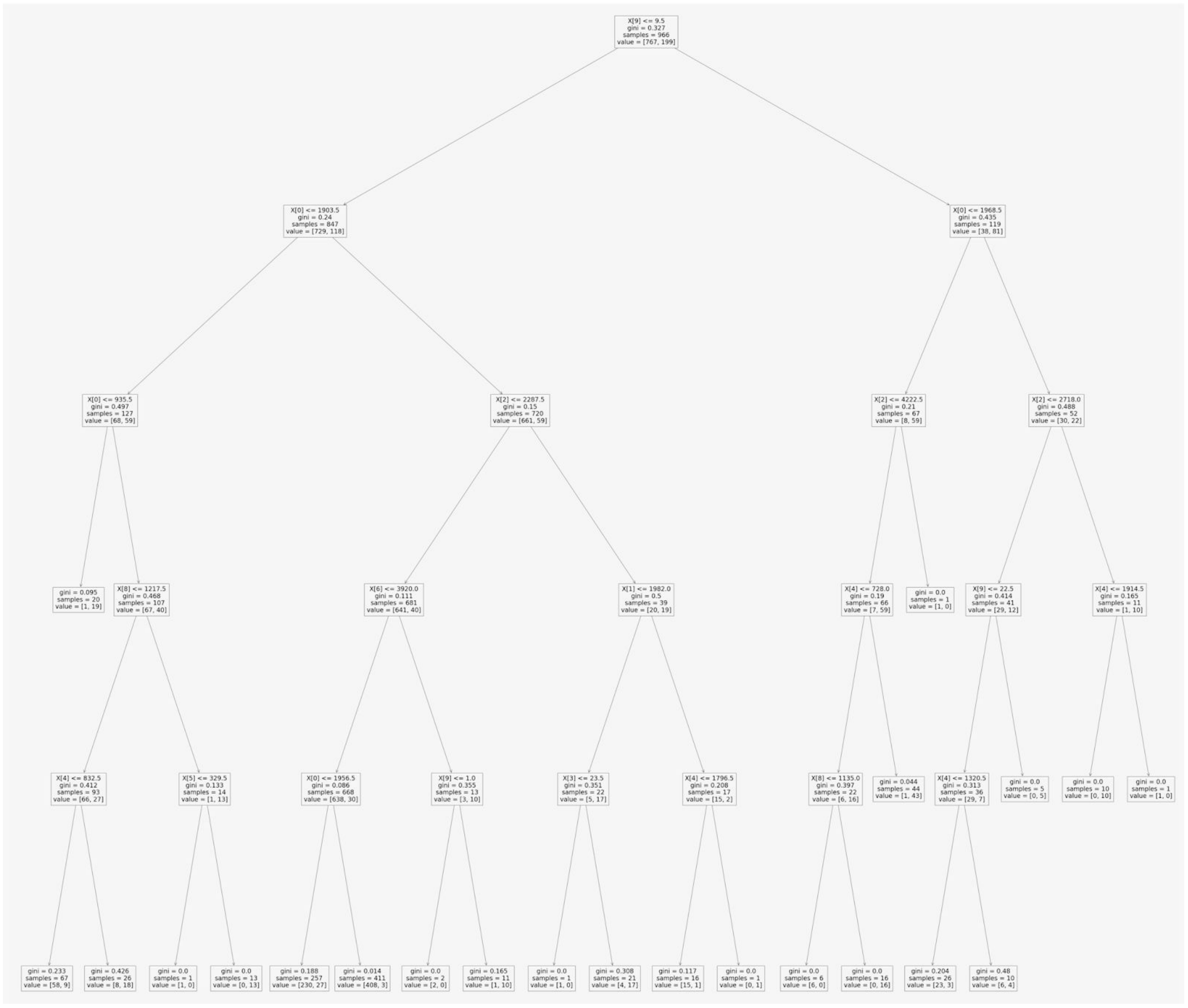
X[9] <= 9.5 gini = 0.327 samples = 966 value = [767, 199]
X[0] <= 1903.5 gini = 0.24 samples = 847 value = [729, 118]
X[0] <= 1968.5 gini = 0.435 samples = 119 value = [38, 81]
X[0] <= 935.5 gini = 0.497 samples = 127 value = [68, 59]
X[2] <= 2287.5 gini = 0.15 samples = 720 value = [661, 59]
X[2] <= 4222.5 gini = 0.21 samples = 67 value = [8, 59]
X[2] <= 2718.0 gini = 0.488 samples = 52 value = [30, 22]
gini = 0.095 samples = 20 value = [1, 19]
X[8] <= 1217.5 gini = 0.468 samples = 107 value = [67, 40]
X[6] <= 3920.0 gini = 0.111 samples = 681 value = [641, 40]
X[1] <= 1982.0 gini = 0.5 samples = 39 value = [20, 19]
X[4] <= 728.0 gini = 0.19 samples = 66 value = [7, 59]
gini = 0.0 samples = 1 value = [1, 0]
X[9] <= 22.5 gini = 0.414 samples = 41 value = [29, 12]
X[4] <= 1914.5 gini = 0.165 samples = 11 value = [1, 10]
X[4] <= 832.5 gini = 0.412 samples = 93 value = [66, 27]
X[5] <= 329.5 gini = 0.133 samples = 14 value = [1, 13]
X[0] <= 1956.5 gini = 0.086 samples = 668 value = [638, 30]
X[9] <= 1.0 gini = 0.355 samples = 13 value = [3, 10]
X[3] <= 23.5 gini = 0.351 samples = 22 value = [5, 17]
X[4] <= 1796.5 gini = 0.208 samples = 17 value = [15, 2]
X[8] <= 1135.0 gini = 0.397 samples = 22 value = [6, 16]
gini = 0.044 samples = 44 value = [1, 43]
X[4] <= 1320.5 gini = 0.313 samples = 36 value = [29, 7]
gini = 0.0 samples = 5 value = [0, 5]
gini = 0.0 samples = 10 value = [0, 10]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.233 samples = 67 value = [58, 9]
gini = 0.426 samples = 26 value = [8, 18]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 13 value = [0, 13]
gini = 0.188 samples = 257 value = [230, 27]
gini = 0.014 samples = 411 value = [408, 3]
gini = 0.0 samples = 2 value = [2, 0]
gini = 0.165 samples = 11 value = [1, 10]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.308 samples = 21 value = [4, 17]
gini = 0.117 samples = 16 value = [15, 1]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 6 value = [6, 0]
gini = 0.0 samples = 16 value = [0, 16]
gini = 0.204 samples = 26 value = [23, 3]
gini = 0.48 samples = 10 value = [6, 4]

Figure 1: DT for task A with optimal hyperparameters

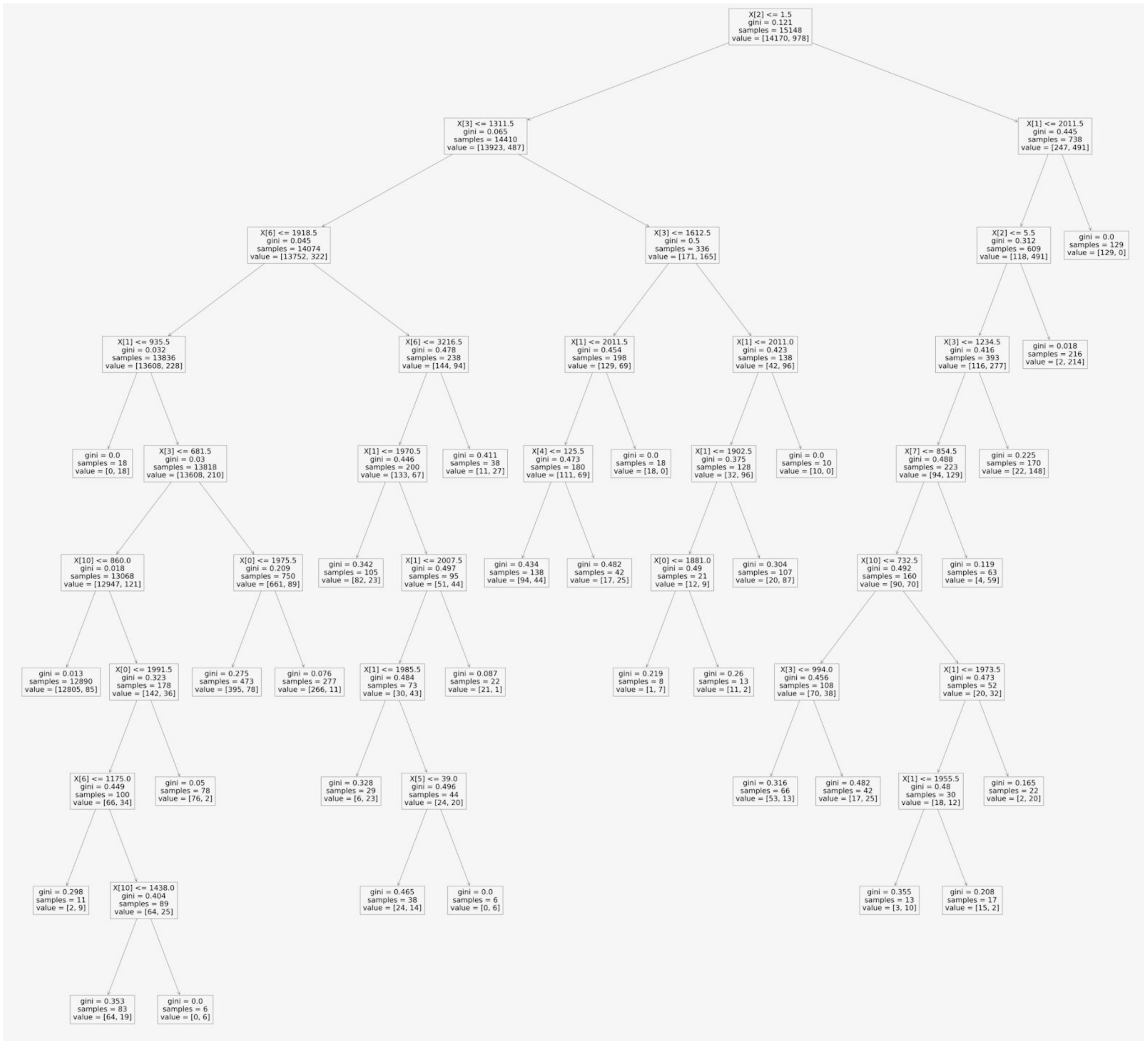
X[2] <= 1.5 gini = 0.121 samples = 15148 value = [14170, 978]
X[3] <= 1311.5 gini = 0.065 samples = 14410 value = [13923, 487]
X[1] <= 2011.5 gini = 0.445 samples = 738 value = [247, 491]
X[6] <= 1918.5 gini = 0.045 samples = 14074 value = [13752, 322]
X[3] <= 1612.5 gini = 0.5 samples = 336 value = [171, 165]
X[2] <= 5.5 gini = 0.312 samples = 609 value = [118, 491]
gini = 0.0 samples = 129 value = [129, 0]
X[1] <= 935.5 gini = 0.032 samples = 13836 value = [13608, 228]
X[6] <= 3216.5 gini = 0.478 samples = 238 value = [144, 94]
X[1] <= 2011.5 gini = 0.454 samples = 198 value = [129, 69]
X[1] <= 2011.0 gini = 0.423 samples = 138 value = [42, 96]
X[3] <= 1234.5 gini = 0.416 samples = 393 value = [116, 277]
gini = 0.018 samples = 216 value = [2, 214]
gini = 0.0 samples = 18 value = [0, 18]
X[3] <= 681.5 gini = 0.03 samples = 13818 value = [13608, 210]
X[1] <= 1970.5 gini = 0.446 samples = 200 value = [133, 67]
gini = 0.411 samples = 38 value = [11, 27]
X[4] <= 125.5 gini = 0.473 samples = 180 value = [111, 69]
gini = 0.0 samples = 18 value = [18, 0]
X[1] <= 1902.5 gini = 0.375 samples = 128 value = [32, 96]
gini = 0.0 samples = 10 value = [10, 0]
X[7] <= 854.5 gini = 0.488 samples = 223 value = [94, 129]
gini = 0.225 samples = 170 value = [22, 148]
X[10] <= 860.0 gini = 0.018 samples = 13068 value = [12947, 121]
X[0] <= 1975.5 gini = 0.209 samples = 750 value = [661, 89]
gini = 0.342 samples = 105 value = [82, 23]
X[1] <= 2007.5 gini = 0.497 samples = 95 value = [51, 44]
gini = 0.434 samples = 138 value = [94, 44]
gini = 0.482 samples = 42 value = [17, 25]
X[0] <= 1881.0 gini = 0.49 samples = 21 value = [12, 9]
gini = 0.304 samples = 107 value = [20, 87]
X[10] <= 732.5 gini = 0.492 samples = 160 value = [90, 70]
gini = 0.119 samples = 63 value = [4, 59]
gini = 0.013 samples = 12890 value = [12805, 85]
X[0] <= 1991.5 gini = 0.323 samples = 178 value = [142, 36]
gini = 0.275 samples = 473 value = [395, 78]
gini = 0.076 samples = 277 value = [266, 11]
X[1] <= 1985.5 gini = 0.484 samples = 73 value = [30, 43]
gini = 0.087 samples = 22 value = [21, 1]
gini = 0.219 samples = 8 value = [1, 7]
gini = 0.26 samples = 13 value = [11, 2]
X[3] <= 994.0 gini = 0.456 samples = 108 value = [70, 38]
X[1] <= 1973.5 gini = 0.473 samples = 52 value = [20, 32]
X[6] <= 1175.0 gini = 0.449 samples = 100 value = [66, 34]
gini = 0.05 samples = 78 value = [76, 2]
gini = 0.328 samples = 29 value = [6, 23]
X[5] <= 39.0 gini = 0.496 samples = 44 value = [24, 20]
gini = 0.316 samples = 66 value = [53, 13]
gini = 0.482 samples = 42 value = [17, 25]
X[1] <= 1955.5 gini = 0.48 samples = 30 value = [18, 12]
gini = 0.165 samples = 22 value = [2, 20]
gini = 0.298 samples = 11 value = [2, 9]
X[10] <= 1438.0 gini = 0.404 samples = 89 value = [64, 25]
gini = 0.465 samples = 38 value = [24, 14]
gini = 0.0 samples = 6 value = [0, 6]
gini = 0.355 samples = 13 value = [3, 10]
gini = 0.208 samples = 17 value = [15, 2]
gini = 0.353 samples = 83 value = [64, 19]
gini = 0.0 samples = 6 value = [0, 6]

Figure 2: DT for task B with optimal hyperparameters

References

[1] Baseball Reference, “Hall of Fame,” *Hall of Fame - BR Bullpen*. [Online]. Available: https://www.baseball-reference.com/bullpen/Hall_of_Fame. [Accessed: 05-Dec-2020].

[2] National Baseball Hall Of Fame, “BBWAA Election Rules,” *Baseball Hall of Fame*. [Online]. Available: https://baseballhall.org/hall-of-famers/rules/bbwaa-rules-for-election. [Accessed: 05-Dec-2020].

[3] D. McConnell, "How Baseball Works (a guide to the game of Baseball)", Howbaseballworks.com, 2020. [Online]. Available: http://www.howbaseballworks.com/Statistics.htm. [Accessed: 05- Dec- 2020].

[4] "MLB Awards", MLB.com, 2020. [Online]. Available: https://www.mlb.com/awards. [Accessed: 05- Dec- 2020].

[5] Baseball Reference, “Baseball statistics,” *Baseball statistics - BR Bullpen*. [Online]. Available: https://www.baseball-reference.com/bullpen/Baseball_statistics. [Accessed: 05-Dec-2020].

[6] "sklearn.tree.DecisionTreeClassifier — scikit-learn 0.23.2 documentation", Scikit-learn.org, 2020. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. [Accessed: 05- Dec- 2020].

Appendix

A. SQL query for task A feature extraction

```
SELECT Master.playerID playerID, debut feature1, finalGame feature2,
awardsCount feature3, B_H feature4, B_HR feature5, B_RBI feature6, P_H
feature7, P_ER feature8, P_HR feature9, P_BB feature10, P_SO feature11,
IF(inducted = 'Y' OR inducted = 'N', 1, 0) as class FROM (SELECT playerID,
YEAR(debut) as debut, YEAR(finalGame) as finalGame FROM Master) as Master
LEFT JOIN (SELECT playerID, sum(H) as B_H, sum(HR) as B_HR, sum(RBI) as B_RBI
FROM Batting GROUP BY playerID) as Batting on Master.playerID =
Batting.playerID LEFT JOIN (SELECT playerID, sum(H) as P_H, sum(ER) as P_ER,
sum(HR) as P_HR, sum(BB) as P_BB, sum(SO) as P_SO FROM Pitching GROUP BY
playerID) as Pitching on Master.playerID = Pitching.playerID
LEFT JOIN (SELECT playerID, count(*) as awardsCount FROM AwardsPlayers GROUP BY
playerID) as AwardsPlayers on Master.playerID = AwardsPlayers.playerID
LEFT JOIN (SELECT DISTINCT playerID, inducted, category FROM HallOfFame
INNER JOIN (SELECT playerID as inductedPlayerID, max(inducted) as wasInducted
FROM HallOfFame GROUP BY inductedPlayerID) AS InductedHOF ON
HallOfFame.playerID = InductedHOF.inductedPlayerID and HallOfFame.inducted =
InductedHOF.wasInducted) as HallOfFame on Master.playerID = HallOfFame.playerID
WHERE category = 'Player' OR debut != '' ORDER BY Master.playerID;
```

B. SQL query for task B feature extraction

```
SELECT HOF.playerID playerID, debut feature1, finalGame feature2, B_H feature3,
B_HR feature4, B_RBI feature5, P_GS feature6, P_H feature7, P_BB feature8, P_SO
feature9, awardsCount feature10, IF(inducted = 'Y', 1, 0) as class FROM
(SELECT DISTINCT playerID, inducted, category FROM HallOfFame WHERE category =
'Player') as HOF INNER JOIN (SELECT DISTINCT playerID as inductedPlayerID,
max(inducted) as wasInducted FROM HallOfFame WHERE category = 'Player' GROUP BY
inductedPlayerID) AS InductedHOF ON HOF.playerID = InductedHOF.inductedPlayerID
and HOF.inducted = InductedHOF.wasInducted LEFT JOIN (SELECT playerID,
YEAR(debut) as debut, YEAR(finalGame) as finalGame FROM Master) as Master on
HOF.playerID = Master.playerID LEFT JOIN (SELECT playerID, sum(H) as B_H,
sum(HR) as B_HR, sum(RBI) as B_RBI FROM Batting GROUP BY playerID) as Batting
on HOF.playerID = Batting.playerID LEFT JOIN (SELECT playerID, sum(GS) as P_GS,
sum(H) as P_H, sum(BB) as P_BB, sum(SO) as P_SO FROM Pitching GROUP BY
playerID) as Pitching on HOF.playerID = Pitching.playerID LEFT JOIN (SELECT
playerID, count(*) as awardsCount FROM AwardsPlayers GROUP BY playerID) as
AwardsPlayers on Master.playerID = AwardsPlayers.playerID ORDER BY
HOF.playerID;
```