Group Members:

- Stephen Chi (s3chi@uwaterloo.ca, 20843579)
- Nicholas Chung-Hun (nchunghu@uwaterloo.ca, 20837328)
- Ernest Ho (e44ho@uwaterloo.ca, 20844632)

Classification

Classification Categories

1. Inducted into the Baseball Hall of Fame (player appears in HallOfFame table and inducted = ‘Y’)
2. Nominated for the Baseball Hall of Fame but not inducted (player appears in HallOfFame table and inducted = ‘N’)
3. Not nominated for the Baseball Hall of Fame (player does not appear in HallOfFame table)

Given that we are classifying players using statistics, we have decided to exclude all players that do not have any games played or any games managed. Our justification is found in the data cleansing section.

Classification Task A

Since Task A is looking to classify a player as True if they belong to category 1 or 2, and False otherwise, category 1 and 2 can be generalized as whether the player appears in the HallOfFame table regardless of the value in the inducted column. A player can be nominated multiple times to the Baseball Hall of Fame, therefore we are only looking for unique entries of players that appear in the Hall of Fame.

SQL Query

```sql
-- Task A Classification Distribution
WITH c1 as (SELECT COUNT(DISTINCT h.playerID) as numTrueClass
            FROM HallOfFame h
            WHERE h.playerID IN (
                SELECT F.playerID FROM
                    (SELECT pe.playerID, SUM(m.G) as managerGames
                    FROM People pe
                    LEFT JOIN Managers m on pe.playerID = m.playerID
                    GROUP BY pe.playerID) AS F
                    JOIN
                    (SELECT pe.playerID, SUM(a.G_all) as totalGames
                    FROM People pe
                    LEFT JOIN Appearances a on pe.playerID = a.playerID
                    GROUP BY pe.playerID) AS G
                    ON F.playerID = G.playerID
                WHERE (F.managerGames > 0 OR G.totalGames > 0))),
    c2 as (SELECT COUNT(DISTINCT p.playerID) as totalPlayers
            FROM People p
            WHERE p.playerID IN (
                SELECT F.playerID FROM
                    (SELECT pe.playerID, SUM(m.G) as managerGames
                    FROM People pe
                    LEFT JOIN Managers m on pe.playerID = m.playerID
                    GROUP BY pe.playerID) AS F
                    LEFT JOIN
                    (SELECT pe.playerID, SUM(a.G_all) as totalGames
                    FROM People pe
                    LEFT JOIN Appearances a on pe.playerID = a.playerID
                    GROUP BY pe.playerID) AS G
                    ON F.playerID = G.playerID
                WHERE F.managerGames > 0 OR G.totalGames > 0))
SELECT c2.totalPlayers,
       c1.numTrueClass,
       c2.totalPlayers - c1.numTrueClass as numFalseClass,
       c1.numTrueClass / c2.totalPlayers as trueClassProportion,
       ((c2.totalPlayers - c1.numTrueClass) / c2.totalPlayers) as falseClassProportion
FROM c1, c2;
```

Output

totalPlayers	numTrueClass	numFalseClass	trueClassProportion	falseClassProportion
20305	1217	19088	0.0599	0.9401

- There are a total of 20,305 players in the database (Distinct players in the People Table)
    - True Classification - 1217 players belong to category 1 and 2 (Distinct players in HallOfFame Table) accounts for 5.99%
    - False Classification - 19088 players do not belong (Total players - True Classification) accounts for 94.01%

Classification Task B

Since Task B is looking to classify a player as True if they belong to category 1, and False if they belong to category 2, we will only be looking at players in the HallOfFame table and focusing on the inducted column. A player can be nominated multiple times to the Baseball Hall of Fame, therefore we are only looking for unique entries of players that appear in the Hall of Fame.

SQL Query

```sql
-- Task B Classification Distribution
WITH c1 as (SELECT COUNT(DISTINCT h.playerID) as numTrueClass
            FROM HallOfFame h
            WHERE inducted = 'Y'
            AND h.playerID IN (
                SELECT F.playerID FROM (
                    (SELECT pe.playerID, SUM(m.G) as managerGames
                    FROM People pe
                    LEFT JOIN Managers m on pe.playerID = m.playerID
                    GROUP BY pe.playerID) AS F
                    LEFT JOIN
                    (SELECT pe.playerID, SUM(a.G_all) as totalGames
                    FROM People pe
                    LEFT JOIN Appearances a on pe.playerID = a.playerID
                    GROUP BY pe.playerID) AS G
                    ON F.playerID = G.playerID)
                WHERE F.managerGames > 0 OR G.totalGames > 0)),
    c2 as (SELECT COUNT(DISTINCT p.playerID) as totalNominatedPlayers
            FROM HallOfFame p
            WHERE p.playerID IN (
                SELECT F.playerID FROM (
                    (SELECT pe.playerID, SUM(m.G) as managerGames
                    FROM People pe
                    LEFT JOIN Managers m on pe.playerID = m.playerID
                    GROUP BY pe.playerID) AS F
                    LEFT JOIN
                    (SELECT pe.playerID, SUM(a.G_all) as totalGames
                    FROM People pe
                    LEFT JOIN Appearances a on pe.playerID = a.playerID
                    GROUP BY pe.playerID) AS G
                    ON F.playerID = G.playerID)
                WHERE F.managerGames > 0 OR G.totalGames > 0))
SELECT c2.totalNominatedPlayers,
       c1.numTrueClass,
       c2.totalNominatedPlayers - c1.numTrueClass as numFalseClass,
       c1.numTrueClass / c2.totalNominatedPlayers as trueClassProportion,
       ((c2.totalNominatedPlayers - c1.numTrueClass) / c2.totalNominatedPlayers) as falseClassProportion
FROM c1, c2
```

Output

totalNominatedPlayers	numTrueClass	numFalseClass	trueClassProportion	falseClassProportion ▲ 1
1217	261	956	0.2145	0.7855

- There are a total of 1217 players nominated to the HallOfFame in the database (Distinct players in the HallOfFame Table)
    - True Classification - 261 players belong to category 1, and were inducted into the Hall of Fame (Distinct players in HallOfFame Table where inducted = ‘Y’) accounts for 21.45%
    - False Classification - 956 players do not belong, and were only nominated (Total Nominated Players - True Classification) accounts for 78.55%

Data Cleansing

Null/Missing Values

There were multiple players that had missing / null values in various tables in the dataset. This was related but not limited to the following reasons:

- The Major League Baseball (MLB) banned black players which did not end until 1947
- Not all players field, pitch and bat
- Not all players transition into to coaching / managing

We decided to remove all players that have not played a single game or have not coached a single game. This is because these values do not contribute meaningful data for our decision tree since we are basing both our prediction tasks on statistics of players. If the player has no statistics whatsoever, it skews the dataset with 0 values. For all other players that have played or managed but have missing / null values, we assigned a default value of 0 or the average over all of the players for those values.

Feature Selection

Task A

Our overall feature selection attempts to put equal weighting on exceptional statistics in all three categories that a baseball player could excel at: fielding, batting and pitching. Each category contains at least two statistics demonstrating their ability in that particular category.

Feature	**Description**	**Rationale**
gamesBatted	Sum total of # of games the player has batted	Longevity is a large determinant of Hall Of Fame nomination
totalRBI	Sum total of runs batted in (RBI)	Provides overall team impact that the player has had on scoring points on hits
totalHR	Sum total of home runs	Shows ability for a batter to hit home runs which is a difficult hit
gamesPitched	Sum total of # of games the player has pitched	Longevity is a large determinant of Hall Of Fame nomination
opponentBattingAvg	Average of opponent's batting	Shows impact of pitcher's ability to prevent hits
pitchingERA	Earned run average	Shows impact of pitcher's ability to prevent runs for the opposing team
gamesFielded	Sum total of # of games the player has fielded	Longevity is a large determinant of Hall Of Fame nomination
totalPutOut	Sum total of # of putouts	Shows defensive impact in preventing offensive players from reaching bases potentially leading to points
totalGames	Sum total of # of games played	Longevity is a large determinant of Hall Of Fame nomination
totalSeasons	Sum total of seasons	Only players with more than 10 seasons are allowed to be nominated to the Hall of Fame
managerWins	Sum total of wins as a manager	Managers are able to get nominated to the Hall of Fame therefore wins provide insight on their coaching ability
totalAwards	Sum total of any awards won over their career	End of season awards are only given to a handful of players that have had an exceptional season

allstarAppearances	Sum total of # of all star appearances	All-star selection provide insight on a player's performance during a given season
propotionalSal	Salary proportional to the league average salary	High performing players which are likely to be in the Hall of Fame are likely to be paid more than league average

Task B

For Task B, we also introduced the following features:

battingAverage	Average number of hits per time at bat	Shows the batter's average number of hits every time they bat
strikeOuts	Number of strikeouts a pitcher causes	Shows pitcher's ability to throw difficult to return pitches and cause strikeouts

Sampling

To split our data into two different subsets for training and testing, we use Python's sklearn train_test_split function to randomly divide the data into a 80-20 split. We perform this division of data into two subsets randomly to avoid introducing any bias in the split.

We also do not know how this data was collected. This data may have been collected in a non-random way. The data could have started sampling from a set of people all sharing some common trait, meaning that sample would not be representative of all baseball players. Randomly sampling the whole set to divide our data will negate this issue and give us appropriately representative subsets, which is essential to construct a strong decision tree.

We perform this split to have data to train our classifier, while also having data to evaluate whether our classifier has generalized the data to be able to classify new data. This gives us insight on whether we are overfitting or underfitting our decision tree classifier.

Validation Results

Task A

		Predicted condition	
	Total 256 + 3805= 4061	**Nominated** 261	**Not nominated** 3800
Actual condition	**Nominated** 256	187	69
	Not nominated 3805	74	3731

```
True Positive Rate:     0.73046875
False Negative Rate:    0.26953125
Accuracy:       0.9647869982762867
Recall:         0.73046875
Precision:      0.7164750957854407
F1 Score:       0.723404255319149
```

For Task A, we were able to obtain an accuracy of approximately 96.5%. At first glance, this seems to indicate that our decision tree classifier is performing well.

However, a decision tree classifier that would always predict “not nominated” would also have a high accuracy. This kind of classifier would have a true positive rate of 0, but a high true negative rate, since the percentage of players that are nominated for hall of fame is very low. This is why we obviously must also look at other performance metrics to measure correctness. The other three metrics we consider are recall, precision, and F1 score.

Recall is a measure of how many of the actual hall of fame nominees were correctly predicted as being hall of fame nominees.

Measuring precision allows us to see how well our classifier predicts people that are nominated by comparing the amount of true positives vs false positives. As we can see, even with a high accuracy, our precision is not nearly as high. This indicates that our classifier is incorrectly predicting many players as hall of fame nominees when they were in fact not.

Finally, F1 score is a metric that combines recall and precision, and is a good overall way to measure correctness for this classifier. This metric is much more useful than accuracy in this case, given the large amount of true negatives inflating the accuracy (since most players don’t get nominated for hall of fame).

Task B

		Predicted condition	
	Total 43 + 201 = 244	**Inducted** 34	**Not inducted** 210
Actual condition	**Inducted** 43	28	15
	Not inducted 201	6	195

```
True Positive Rate:     0.6511627906976745
False Negative Rate:    0.3488372093023256
Accuracy:       0.9139344262295082
Recall:         0.6511627906976745
Precision:      0.8235294117647058
F1 Score:       0.7272727272727273
```

For Task B, we were able to obtain an accuracy of approximately 91.4%. This indicates that our decision tree classifier for predicting inductees out of hall of fame nominees is quite accurate.

The number of people inducted vs not inducted (Task B) is much more even than the number of people nominated vs not nominated (Task A). Therefore, for Task B's classifier, accuracy is a much more useful measure of correctness than Task A. Even so, we still take a look at the other metrics that we looked at for Task A to provide more ideas of how well the classifier is performing.

The recall for Task B was around 65.1%, and precision was around 82.4%. Since F1 score is a balance of both recall and precision, F1 score ends up somewhere in the middle of them, at around 72.7%.

Hyperparameter Tuning

Hyperparameter tuning was done through a semi-manual inspection and GridSearchCV. We decided to focus on five (5) specific hyperparameters for both Task A and Task B. This is because of the imbalanced nature of both datasets and overfitted results that were found. The hyperparameters are as follows:

- CCP Alpha
- Class Weight
- Max Depth
- Minimum Leaf Samples
- Minimum Split Samples

Looking at our initial validation results without any hyperparameters, we could see that overfitting was present; our model is predicting the training set very well whereas not performing as well on the testing set. Below is a screenshot demonstrating the results of Task A before applying hyperparameters.

```
Training Dataset
True Negatives:         15288
False Positives:        0
False Negatives:        0
True Positives:         952
True Positive Rate:     1.0
False Negative Rate:    0.0


Testing Dataset
True Negatives:         3679
False Positives:        121
False Negatives:        121
True Positives:         139
True Positive Rate:     0.5346153846153846
False Negative Rate:    0.4653846153846154
```

Given that we are currently overfitting, we can apply pruning through these hyperparameters to reduce the size and change the complexity of the tree.

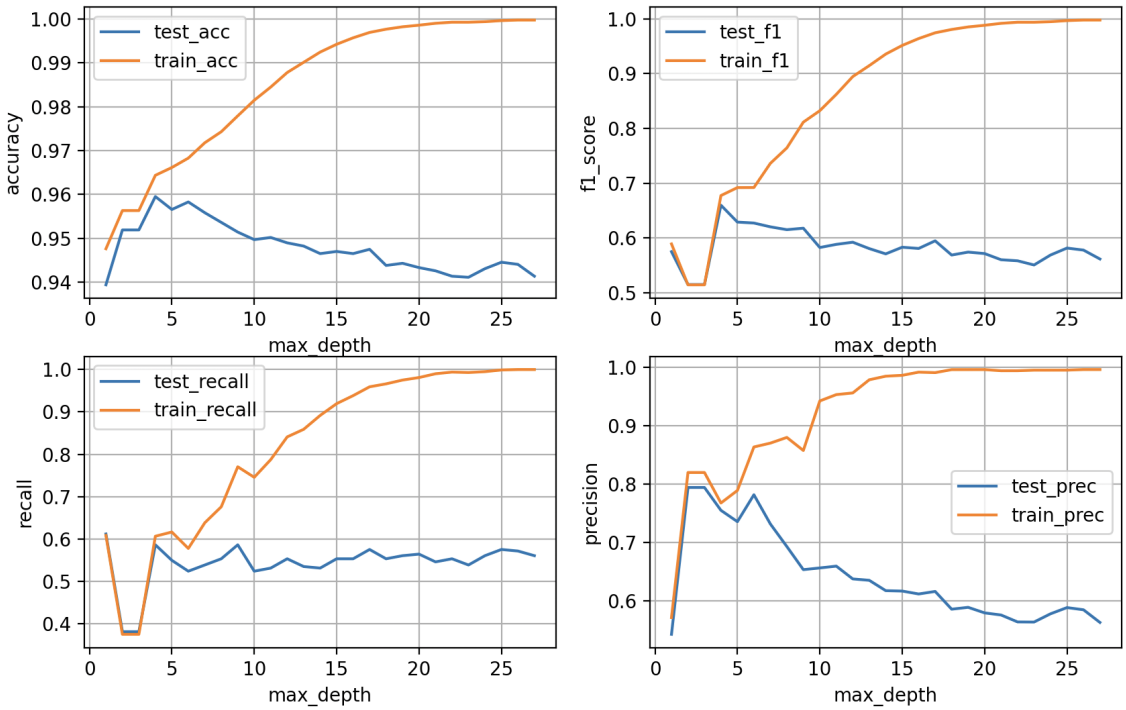
Class Weights

Given our highly imbalanced dataset as previously noted (94% False, 6% True) we can use the class weights to help shift the imbalance. We can use “balanced” as a class weight or manually apply weighting to both our classes. Using “balanced” would assign weights inversely proportional to their respective frequencies (i.e 0 (False) would receive a weight of 0.06 and 1 (True) would receive a weight of 0.94). The change in class weights did not change the results

in any significant way, therefore we will leave it as the default value, None. This was true in both task's cases.

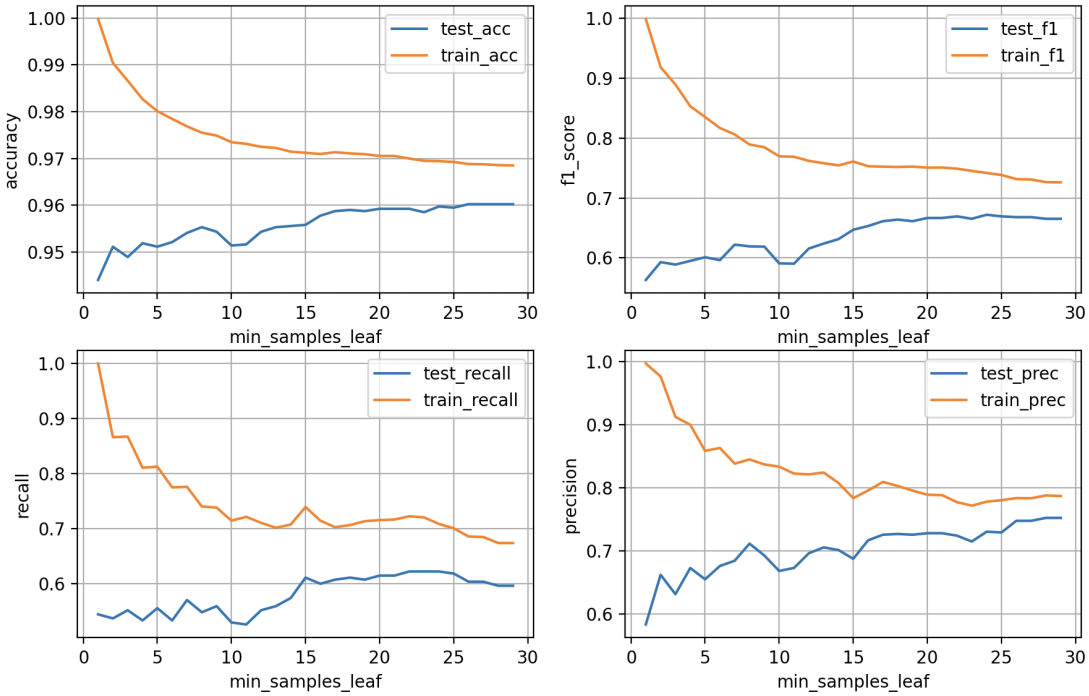
Max Depth

Max depth restricts the depth of the tree. On average, the tree's depth when not restricted hits a maximum of 28, therefore we will input the range of 1 to 28 into GridSearchCV to find the optimal depth. By restricting the depth of the tree, we can see that a lower tree depth has a positive effect on all metrics with most reaching a maximum around a depth of 4. An extremely low max depth of 1 to 3 has the best effect on precision but recall and f1 score suffers.



Minimum Leaf Samples

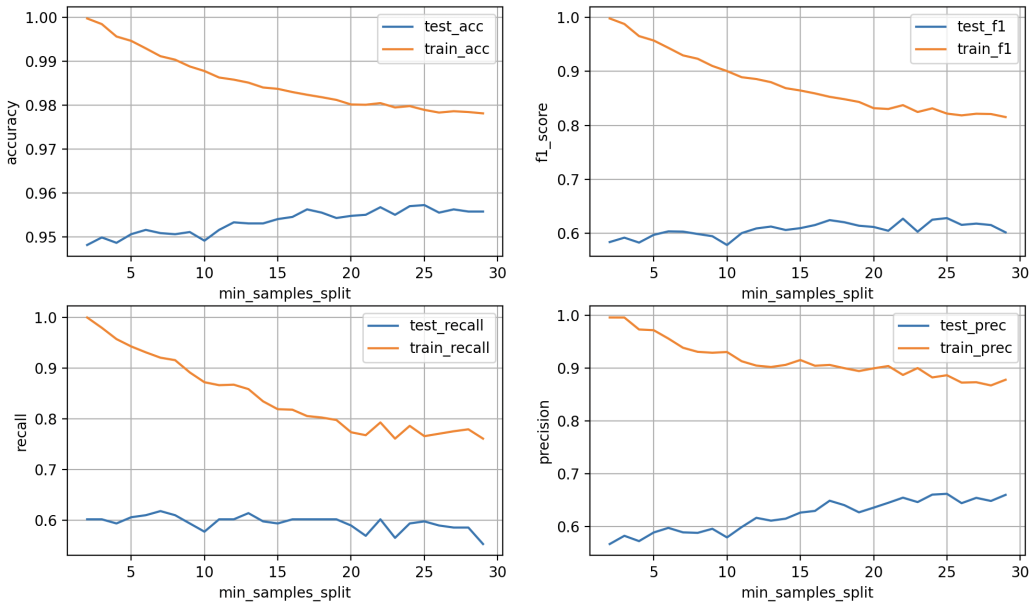
Minimum leaf samples specify the minimum number of samples that are required at a leaf node. By default, the tree requires only 1 sample therefore we will search from a range of 1 to 30.



All metrics improve with a higher sample per leaf on our test sets. This increases the complexity in the sense that it restricts the tree from creating single sample branches which leads to overfitting the data.

Minimum Split Samples

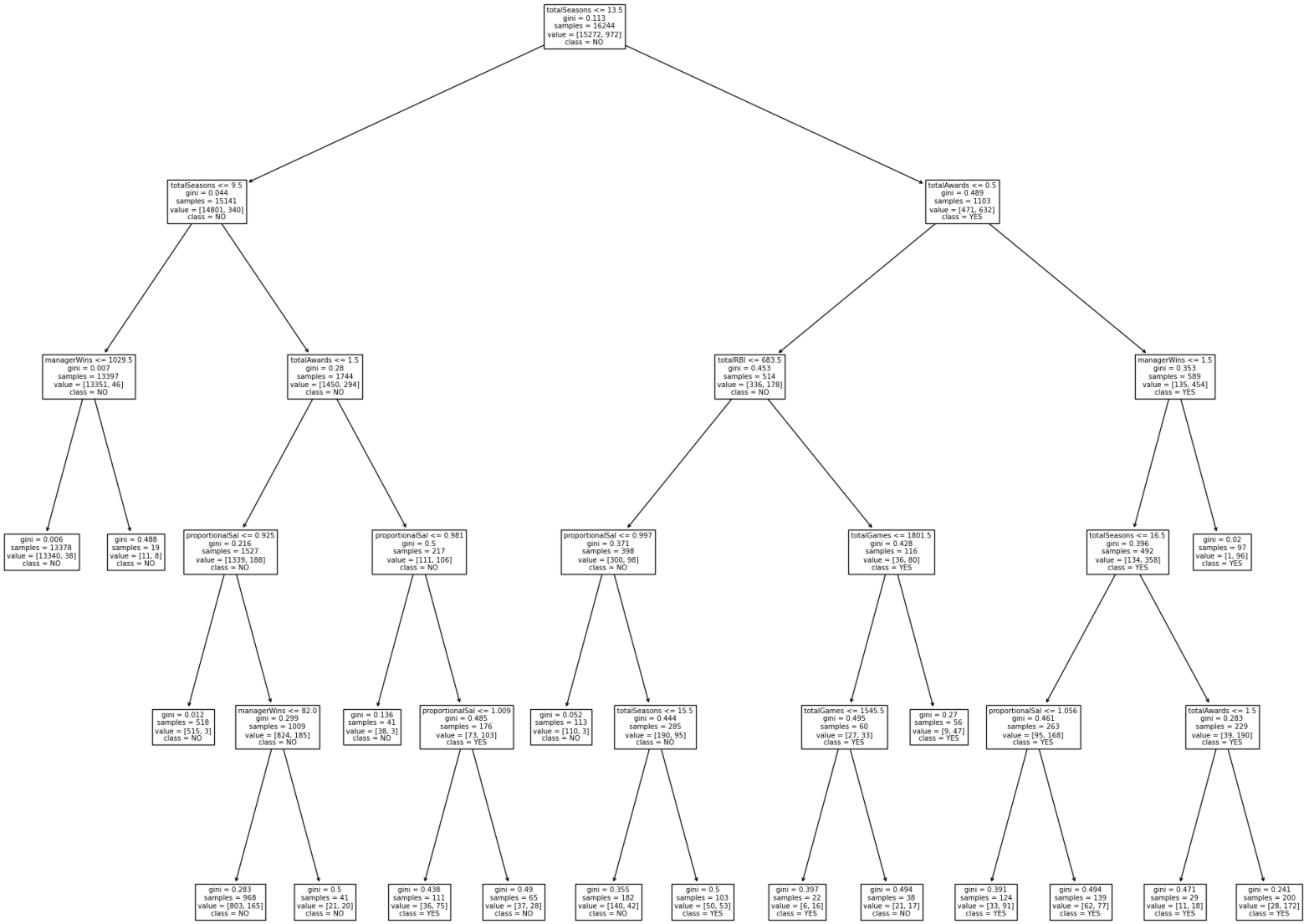
Minimum split samples specify the minimum number of samples needed to split a node into left and right subtrees. By default, the tree requires only 2 samples therefore we will search from a range of 2 to 30.

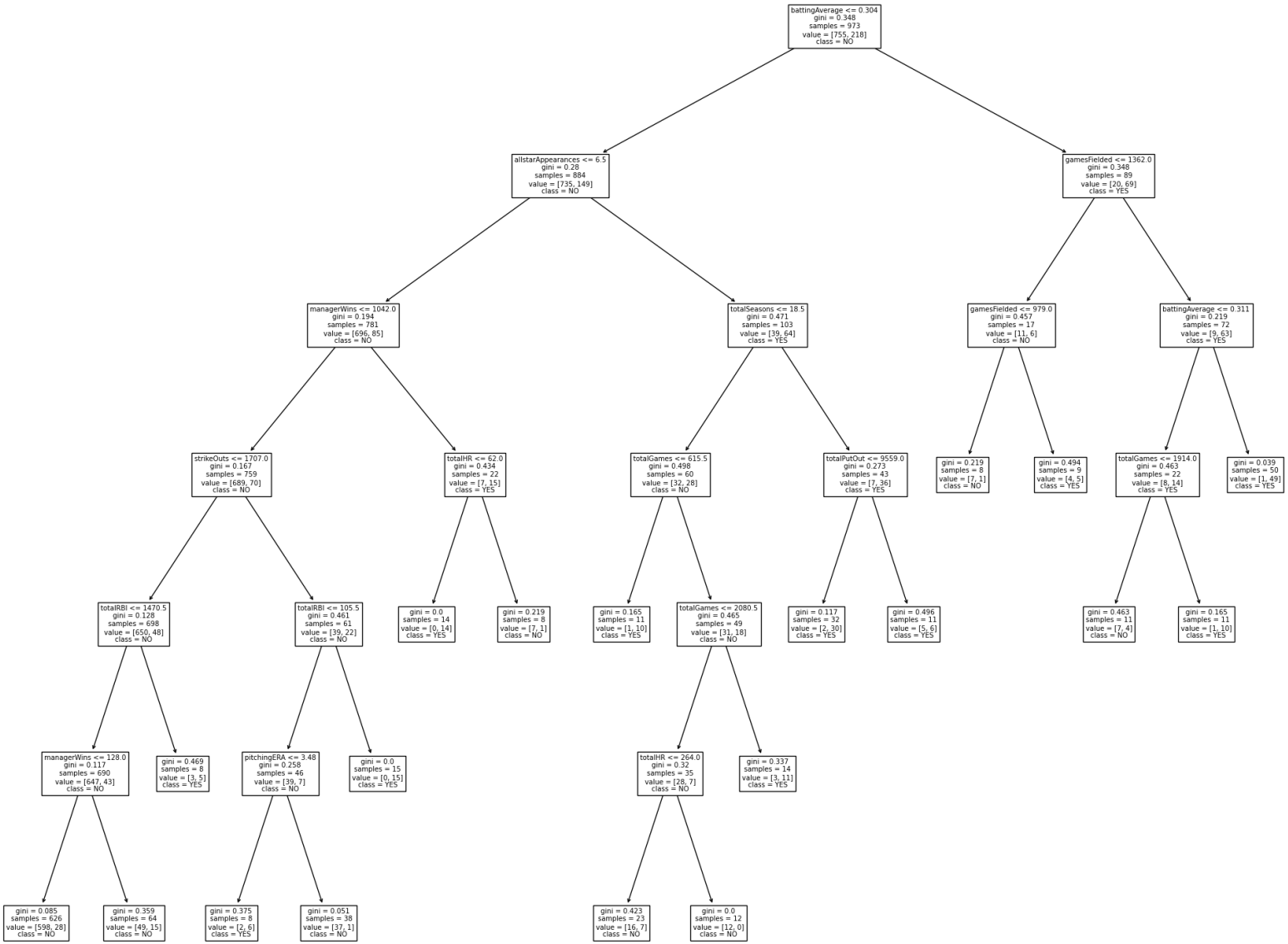


With all this information, we use GridSearchCV and found the optimal parameters as used in the final decision tree classifier.

CCP Alpha

The CCP alpha value controls post-pruning of the tree which essentially removes the “weakest link” or in other words, less useful information. CCP alphas were selected through the cost_complexity_pruning_path function. Pruning reduces the size and complexity of the tree which reduces the overfitting that occurred. When the size and complexity decreased, the overall accuracy, precision and recall all improved.


totalSeasons <= 13.5
gini = 0.113
samples = 16244
value = [15272, 972]
class = NO
totalSeasons <= 9.5
gini = 0.044
samples = 15141
value = [14801, 340]
class = NO
totalAwards <= 0.5
gini = 0.489
samples = 1103
value = [471, 632]
class = YES
managerWins <= 1029.5
gini = 0.007
samples = 13397
value = [13351, 46]
class = NO
totalAwards <= 1.5
gini = 0.28
samples = 1744
value = [1450, 294]
class = NO
totalRBI <= 683.5
gini = 0.453
samples = 514
value = [336, 178]
class = NO
managerWins <= 1.5
gini = 0.353
samples = 589
value = [135, 454]
class = YES
gini = 0.006
samples = 13378
value = [13340, 38]
class = NO
gini = 0.488
samples = 19
value = [11, 8]
class = NO
proportionalSal <= 0.925
gini = 0.216
samples = 1527
value = [1339, 188]
class = NO
proportionalSal <= 0.981
gini = 0.5
samples = 217
value = [111, 106]
class = NO
proportionalSal <= 0.997
gini = 0.371
samples = 398
value = [300, 98]
class = NO
totalGames <= 1801.5
gini = 0.428
samples = 116
value = [36, 80]
class = YES
totalSeasons <= 16.5
gini = 0.396
samples = 492
value = [134, 358]
class = YES
gini = 0.02
samples = 97
value = [1, 96]
class = YES
gini = 0.012
samples = 518
value = [515, 3]
class = NO
managerWins <= 82.0
gini = 0.299
samples = 1009
value = [824, 185]
class = NO
gini = 0.136
samples = 41
value = [38, 3]
class = NO
proportionalSal <= 1.009
gini = 0.485
samples = 176
value = [73, 103]
class = YES
gini = 0.052
samples = 113
value = [110, 3]
class = NO
totalSeasons <= 15.5
gini = 0.444
samples = 285
value = [190, 95]
class = NO
totalGames <= 1545.5
gini = 0.495
samples = 60
value = [27, 33]
class = YES
gini = 0.27
samples = 56
value = [9, 47]
class = YES
proportionalSal <= 1.056
gini = 0.461
samples = 263
value = [95, 168]
class = YES
totalAwards <= 1.5
gini = 0.283
samples = 229
value = [39, 190]
class = YES
gini = 0.283
samples = 968
value = [803, 165]
class = NO
gini = 0.5
samples = 41
value = [21, 20]
class = NO
gini = 0.438
samples = 111
value = [36, 75]
class = YES
gini = 0.49
samples = 65
value = [37, 28]
class = NO
gini = 0.355
samples = 182
value = [140, 42]
class = NO
gini = 0.5
samples = 103
value = [50, 53]
class = YES
gini = 0.397
samples = 22
value = [6, 16]
class = YES
gini = 0.494
samples = 38
value = [21, 17]
class = NO
gini = 0.391
samples = 124
value = [33, 91]
class = YES
gini = 0.494
samples = 139
value = [62, 77]
class = YES
gini = 0.471
samples = 29
value = [11, 18]
class = YES
gini = 0.241
samples = 200
value = [28, 172]
class = YES


battingAverage <= 0.304
gini = 0.348
samples = 973
value = [755, 218]
class = NO
allstarAppearances <= 6.5
gini = 0.28
samples = 884
value = [735, 149]
class = NO
gamesFielded <= 1362.0
gini = 0.348
samples = 89
value = [20, 69]
class = YES
managerWins <= 1042.0
gini = 0.194
samples = 781
value = [696, 85]
class = NO
totalSeasons <= 18.5
gini = 0.471
samples = 103
value = [39, 64]
class = YES
gamesFielded <= 979.0
gini = 0.457
samples = 17
value = [11, 6]
class = NO
battingAverage <= 0.311
gini = 0.219
samples = 72
value = [9, 63]
class = YES
strikeOuts <= 1707.0
gini = 0.167
samples = 759
value = [689, 70]
class = NO
totalHR <= 62.0
gini = 0.434
samples = 22
value = [7, 15]
class = YES
totalGames <= 615.5
gini = 0.498
samples = 60
value = [32, 28]
class = NO
totalPutOut <= 9559.0
gini = 0.273
samples = 43
value = [7, 36]
class = YES
gini = 0.219
samples = 8
value = [7, 1]
class = NO
gini = 0.494
samples = 9
value = [4, 5]
class = YES
totalGames <= 1914.0
gini = 0.463
samples = 22
value = [8, 14]
class = YES
gini = 0.039
samples = 50
value = [1, 49]
class = YES
totalRBI <= 1470.5
gini = 0.128
samples = 698
value = [650, 48]
class = NO
totalRBI <= 105.5
gini = 0.461
samples = 61
value = [39, 22]
class = NO
gini = 0.0
samples = 14
value = [0, 14]
class = YES
gini = 0.219
samples = 8
value = [7, 1]
class = NO
gini = 0.165
samples = 11
value = [1, 10]
class = YES
totalGames <= 2080.5
gini = 0.465
samples = 49
value = [31, 18]
class = NO
gini = 0.117
samples = 32
value = [2, 30]
class = YES
gini = 0.496
samples = 11
value = [5, 6]
class = YES
gini = 0.463
samples = 11
value = [7, 4]
class = NO
gini = 0.165
samples = 11
value = [1, 10]
class = YES
managerWins <= 128.0
gini = 0.117
samples = 690
value = [647, 43]
class = NO
gini = 0.469
samples = 8
value = [3, 5]
class = YES
pitchingERA <= 3.48
gini = 0.258
samples = 46
value = [39, 7]
class = NO
gini = 0.0
samples = 15
value = [0, 15]
class = YES
totalHR <= 264.0
gini = 0.32
samples = 35
value = [28, 7]
class = NO
gini = 0.337
samples = 14
value = [3, 11]
class = YES
gini = 0.085
samples = 626
value = [598, 28]
class = NO
gini = 0.359
samples = 64
value = [49, 15]
class = NO
gini = 0.375
samples = 8
value = [2, 6]
class = YES
gini = 0.051
samples = 38
value = [37, 1]
class = NO
gini = 0.423
samples = 23
value = [16, 7]
class = NO
gini = 0.0
samples = 12
value = [12, 0]
class = NO