ECE 356 Lab4 Report

Part 1: Analysis and results

Feature Extraction:

First, we need to select features for the classifier. We need to find features that reflect how a player performed. We choose 4 tables, which are *AwardsPlayers*, *AllstarFull*, *Batting* and *Pitching*. For *AwardsPlayers* table, we count how many times a player get the awards. We count three awards, which are "Most Valuable Player", "Gold Glove" and "Cy Young Award". For *AllstarFull* table, we count how many times a player played in all-star games. For *Batting* table, we calculate how many seasons the player played as a Batter and the last season the player played as a Batter. Then, we calculate the total number of G, AB, R, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, IBB, HBP, SH, SF, GIDP. Those statistics can represent how good a batter is. For *Pitching* table, we calculate how many seasons the player played as a Pitcher and the last season the player played as a Pitcher. Then, we calculate the total number of W, L, G, GS, CG, SHO, SV, IPOuts, H, ER, HR, BB, SO, BAOpp, ERA, IBB, WP, HBP, BK, BFP, GF, R, SH, SF, GIDP. Those statistics can represent how good a pitcher is.

Based on those features, we create, train and test the model and have the following testing result:

	Overall Accuracy	Class	Precision	Recall	F1-score
Train (80%)	1	Nominated	1	1	1
		Elected	1	1	1
Test (20%)	0.8305785123966942	Nominated	0.90	0.88	0.89
		Elected	0.60	0.66	0.63

From the result, it is clear that training set has a perfect accuracy and F1 score, but testing set has a lower accuracy and F1 score. Therefore, the model overfit the training data due to too many splits. To solve it, we should delete the correlated features. Let's remove to the next step feature selection.

Feature Selection

For the feature selection, we use "*feature_importances_*" property of the decision matrix to determine if the feature should be removed. Here are the feature importances:

```
('Pitching_LastSeason', 0.0)
('Pitching_SH', 0.0)
('Pitching_WP', 0.0)
('Pitching_IBB', 0.0)
('Pitching_BAOpp', 0.0)
('Pitching_Walks', 0.0)
('Pitching_Homeruns', 0.0)
('Pitching_Hits', 0.0)
('Pitching_Complete_Games', 0.0)
('Pitching_Games_Started', 0.0)
('Pitching_Losses', 0.0)
('Pitching_SF', 0.0)
('Pitching_Seasons_Number', 0.0)
('Batting_Sacrifice_flies', 0.0)
('Batting_Strikeouts', 0.0)
('Pitching_GIDP', 0.0)
('Pitching_GF', 0.003424777288792251)
('Pitching_Earned_Runs', 0.0042521503102086 86)
('Batting_Homeruns', 0.004783669098984772)
('Pitching_BFP', 0.005315187887760858)
('Batting_Doubles', 0.0056695337469449 13)
('Pitching_Shutouts', 0.005721333963911019)
('Batting_Sacrifice_hits', 0.006178905919521997)
('Batting_Grounded_into_double_plays', 0.0065111051625070 49)
('Batting_Intentional_walks', 0.006541769708013362)
('Batting_Caught_Stealing', 0.0069383362992784135)
('Pitching_Outs_Pitched', 0.007784431818828489)
('Pitching_HBP', 0.009111750664732897)
('Pitching_R', 0.009111750664732897)
('Pitching_Games', 0.009567338197969543)
('Batting_Hit_by_pitch', 0.0099659772895516 08)
('Pitching_BK', 0.010205160744500844)
('Gold_Glove', 0.010205160744500844)
('MVP', 0.010461863886151794)
('Batting_Base_on_Balls', 0.016006353202174858)
('Batting_Games', 0.01634487004494096)
('Cy_Young_Award', 0.01752919553276641)
('Batting_Seasons_Number', 0.017878884475026 42)
('Pitching_Saves', 0.018448383830457004)
('Pitching_ERA', 0.02047589848242212)
('Batting_Runs_Batted_In', 0.021055002342484 95)
('Pitching_Strikeouts', 0.02574609246376516)
('Batting_At_Bats', 0.047747934621289595)
('Batting_Stolen_Bases', 0.049629314472637 16)
('Batting_Runs', 0.05452278241364101)
('All_Star_Game', 0.05927912074411982)
('Batting_Triples', 0.07232330991782852)
('Pitching_Wins', 0.098190094004996)
('Batting_LastSeason', 0.15484822253571373)
('Batting_Hits', 0.1782243375188441)
```

Based on the feature importances, we removed the features with a **zero** importance, which are *Pitching_LastSeason, Pitching_SH, Pitching_WP, Pitching_IBB, Pitching_BAOpp, Pitching_Walks, Pitching_Homeruns, Pitching_Hits, Pitching_Complete_Games, Pitching_Games_Started, Pitching_Losses, Pitching_SF, Pitching_Seasons_Number, Batting_Sacrifice_flies, Batting_Strikeouts, Pitching_GIDP*.

After removing those correlated features, we have the accuracy below:

	Overall Accuracy	Class	Precision	Recall	F1-score
Train (80%)	1	Nominated	1	1	1
		Elected	1	1	1
Test (20%)	0.8884297520661157	Nominated	0.90	0.88	0.89
		Elected	0.72	0.76	0.74

From the result, the accuracy and f1 score become higher after removing those features.
In conlucsion, these are the features we are going to use.

Tables	Features
AwardsPlayers	"MVP", "Gold_Glove", "Cy_Young_Award"
AllstarFull	"All_Star_Game"
Batting	"Batting_Seasons_Number", "Batting_LastSeason", "Batting_Games", "Batting_At_Bats", "Batting_Runs", "Batting_Hits", "Batting_Doubles", "Batting_Triples", "Batting_Homeruns", "Batting_Runs_Batted_In", "Batting_Stolen_Bases", "Batting_Caught_Steading","Batting_Base_on_Balls", "Batting_Intentional_walks", "Batting_Hit_by_pitch", "Batting_Sacrifice_hits", "Batting_Grounded_into_double_plays"
Pitching	"Pitching_Wins", "Pitching_Games", "Pitching_Shutouts", "Pitching_Saves", "Pitching_Outs_Pitched", "Pitching_Earned_Runs", "Pitching_Strikeouts", "Pitching_ERA", "Pitching_HBP", "Pitching_BK", "Pitching_BFP", "Pitching_GF", "Pitching_R"

Gini Measure Results:

Threashold on the first 20% testing dataset:

	Overall Accuracy	Class	Precision	Recall	F1-score
Test (20%)	0.8884297520661157	Nominated	0.90	0.88	0.89
		Elected	0.72	0.76	0.74

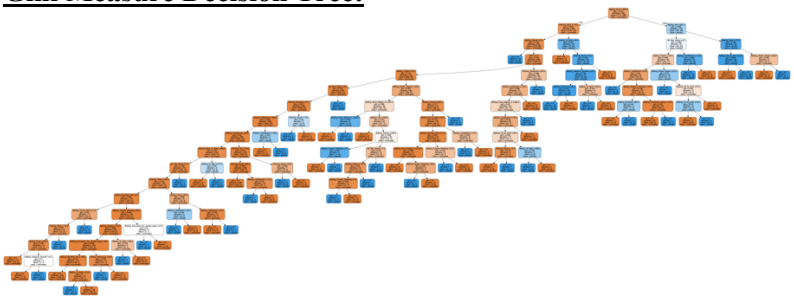
Result from 5 randomly selected testing dataset:

Dataset Number	Overall Accuracy	Class	Precision	Recall	F1-score
1	0.975206612	Nominated	0.99	0.97	0.98
		Elected	0.89	0.98	0.93
2	0.991735537	Nominated	1	0.99	0.99
		Elected	0.96	1	0.98
3	0.962809917	Nominated	0.97	0.98	0.98
		Elected	0.92	0.9	0.91
4	0.97107438	Nominated	0.98	0.98	0.98
		Elected	0.91	0.93	0.92
5	0.991735537	Nominated	0.98	1	0.99
		Elected	1	0.94	0.97
Average	0.9785124				

Based on the first 20% testing dataset, we can see that the accuracy is 0.8884297520661157. After using the 5 randomly selected testing dataset, it is clear that the accuracy increases a lot (to 0.98). This is because the randomly selected testing dataset contain some of the training set, which improves the accuracy and f1 score. For those 5 randomly selected testing dataset, the accurancy is

fairly stable from 0.962809917 to 0.991735537. From Gini measure, it shows that this model has a high accuracy and it is also very stable. The average accuracy is 0.9785124.

Gini Measure Decision Tree:



Entropy Measure Results:

Threashold statistics on the first 20% testing dataset:

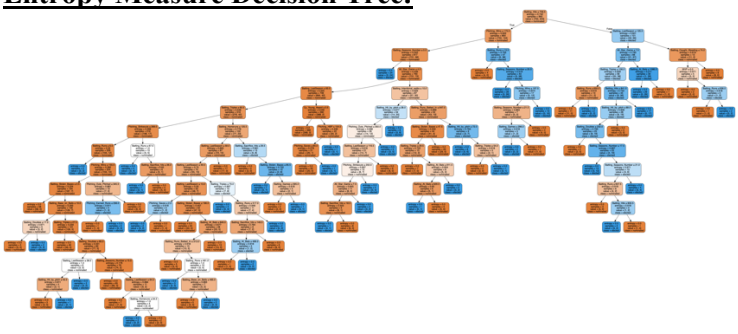
	Overall Accuracy	Class	Precision	Recall	F1-score
Test (20%)	0.8636363636363636	Nominated	0.93	0.90	0.92
		Elected	0.62	0.69	0.65

Result from 5 randomly selected testing dataset:

Dataset Number	Overall Accuracy	Class	Precision	Recall	F1-score
1	0.9752066115702479	Nominated	0.98	0.98	0.98
		Elected	0.95	0.95	0.95
2	0.9669421487603306	Nominated	0.98	0.98	0.98
		Elected	0.93	0.93	0.93
3	0.9669421487603306	Nominated	0.97	0.98	0.98
		Elected	0.94	0.91	0.92
4	0.9545454545454546	Nominated	0.98	0.96	0.97
		Elected	0.85	0.91	0.88
5	0.9793388429752066	Nominated	0.98	0.99	0.99
		Elected	0.96	0.94	0.95
Average Value	0.96859504				

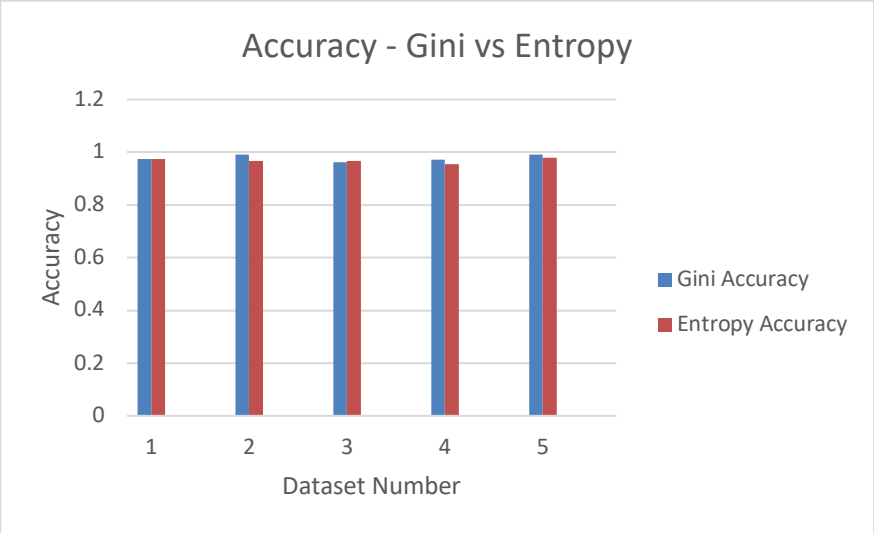
Based on the first 20% testing dataset, we can see that the accuracy is 0.8636363636363636. After using the 5 randomly selected testing dataset, for each test dataset, the accuracy increases to around 0.96. This is because the randomly selected testing dataset contain some of the training set, which improves the accuracy and f1 score. For those 5 randomly selected testing datasets, the accurancy is fairly stable from 0.9545454545454546 to 0.9793388429752066. From Entropy measure, it shows that this model has a high accuracy and it is also very stable. The average accuracy is 0.96859504

Entropy Measure Decision Tree:



Part 2: Comparison

From accuracy bar chart below, it shows that Gini measure and Entropy measure has similar accuracy, which is around 0.95 to 1. If we look a little bit closer, Gini measure has slightly higher accuracy on dataset 1, 2, 4, 5. Since Gini measure has an average accuracy of 0.9785124 and Entropy measure has an average accuracy of 0.96859504, Gini measure has an better accuracy on average.



From f1-score bar char below, it shows that Gini measure and Entropy measure has similar f1 scores. If we look a little bit closer, Gini measure has a slightly higher f1 score overall. This is because Gini measure has a higher precision and recall. After calculation, Gini measure has an average f1 score of 0.984 on Nominated, and an average f1 score of 0.942 on Elected. Entropy has an average f1 score of 0.98 on Nominated, and average f1 score of 0.926 on Elected. Therefore, Gini measure has higher f1 scores for predicting Nominated and Elected. Gini measure has an better f1 score overall.