ECE356 - Database Systems

Lab 4 - Data Mining

Group 3 report

Our general approach to this lab is summarized in the following steps:

1. Write the SQL to extract the dataset from the Lahman2016 database. The columns in the datasets following this order: playerID; feature1; feature2; …; featureN; classification.
2. Use the command "mysql -u <uid> -p -h marmoset04.shoshin.uwaterloo.ca db356_<uid> -B < lab4.sql | tr '\t' ',' > res.csv" and the corresponding database password to export the datasets(features) from the database to desired directory in .csv format. (in this case, it is res.csv)
3. Write the Python Decision Tree to train the dataset in the exported .csv file. The Python Decision Tree will output the report for the first trial for both gini and entropy and then the accuracy and predictions for the rest 5 iteration of trials.

All of our choice of datasets(features) during this lab will be tested using the above approach to determine the quality of the model.

Part 1. Analysis and Results:

The first set of features we chose is the playerId, all the columns that represents the players' stats in game and the votedBy, votes and classification (HallOfFame.inducted) which is listed below:

```
playerid, b_G, b_AB, b_R, b_H, b_2B, b_3B, b_HR, b_RBI, b_SB, b_CS, b_BB,
b_SO, b_IBB, b_HBP, b_SH, b_SF, b_GIDP,p_W, p_L, p_G, p_GS, p_CG, p_SHO,
p_SV,p_IPouts, p_H, p_ER, p_HR, p_BB, p_SO, p_BAOpp, p_ERA, p_IBB, p_WP,
p_HBP, p_BK, p_BFP, p_GF, p_R, p_SH, p_SF, p_GIDP, votedBy, votes and
classification
```

Where b and p means the column is from Batting or Pitching table respectively.

The reason we started with this set of features is because the data in the Batting and Pitching table record all the life time behaviors of players during their matchups. This are the most important stats that can be used to determine whether a player is good and will be likely to nominate or elected into the Hall of Fame. The features votedBy and votes is the fact that how a player is elected for Hall of Fame

With this set of features, the models trained from the Python Decision Tree has the following results:

Model	Accuracy	F1-score
Gini	0.9786776859504132	0.7994736842105263
Entropy	0.9669421487603306	0.7706024096385543

We use both Accuracy and F1-score to analyze the quality of our model because sometimes, accuracy is not a good parameter to determine whether the model is good or not. F1 Score is the weighted average of Precision and Recall. F1 is usually more

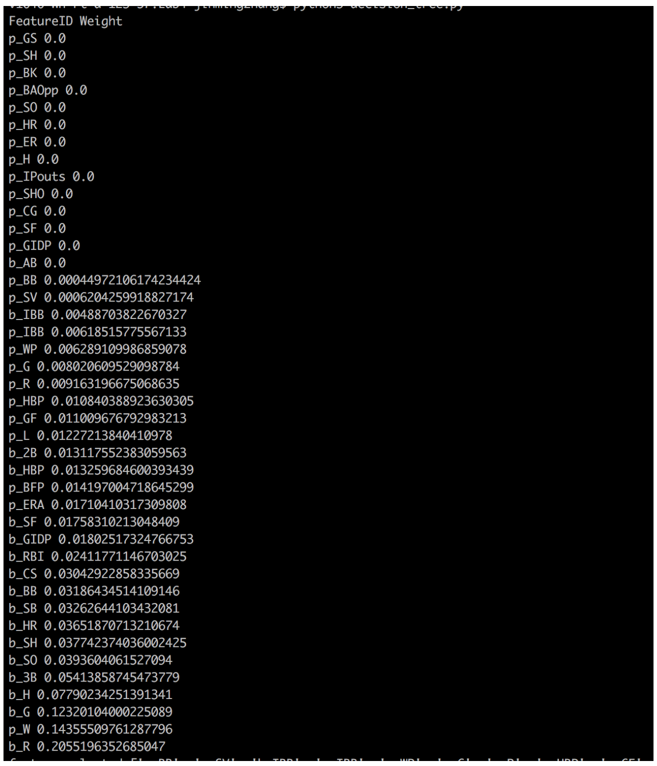
useful than accuracy, especially if the features are unevenly distributed. With the consideration of F1 score, we can have a better analysis. We can see here, both the Accuracy and the F1 score from this model is very high. This shows that this model is Overfitting. Overfitting usually means that a model has trained the data too well. During the learning, there are dominating/bias features acting as noise which will negatively impact the performance of the model on new data. In this case the dominating/bias features are votedBy and votes because even without other features, the model can still achieve a high Accuracy and the F1 score. This will make the model not able to predict new data and negatively impact the models quality. The votedBy and votes are simply the fact of why a player is elected. These two features are highly biased and will not contribute to the model's learning ability, so they should be removed.
Now we have:

```
playerid, b_G, b_AB, b_R, b_H, b_2B, b_3B, b_HR, b_RBI, b_SB, b_CS, b_BB,
b_SO, b_IBB, b_HBP, b_SH, b_SF, b_GIDP,p_W, p_L, p_G, p_GS, p_CG, p_SHO,
p_SV,p_IPouts, p_H, p_ER, p_HR, p_BB, p_SO, p_BAOpp, p_ERA, p_IBB, p_WP,
p_HBP, p_BK, p_BFP, p_GF, p_R, p_SH, p_SF, p_GIDP, and classification
```

And the models trained from the Python Decision Tree has the following results:

Model	Accuracy	F1-score
Gini	0.7829752066115702	0.3356626506024096
Entropy	0.7657446808510639	0.3510743801652892

We can see here, both the Accuracy and the F1 score from this model is very low. The model is not doing good for predictions. One of the reasons which might cost this problem is that some of features from the set have no weights during a prediction of elections meaning that those features are not crucial facts which determine whether a player is good or not. These features will also have a negative impact on the model's learning ability. This means that we need all the individual weight for each features.



```
FeatureID Weight
p_GS 0.0
p_SH 0.0
p_BK 0.0
p_BAOpp 0.0
p_SO 0.0
p_HR 0.0
p_ER 0.0
p_H 0.0
p_IPouts 0.0
p_SHO 0.0
p_CG 0.0
p_SF 0.0
p_GIDP 0.0
b_AB 0.0
p_BB 0.00044972106174234424
p_SV 0.0006204259918827174
b_IBB 0.00488703822670327
p_IBB 0.00618515775567133
p_WP 0.00628910998685907 8
p_G 0.008020609529098784
p_R 0.009163196675068635
p_HBP 0.010840388923630305
p_GF 0.011009676792983213
p_L 0.01227213840410978
b_2B 0.013117552383059563
b_HBP 0.013259684600393439
p_BFP 0.014197004718645299
p_ERA 0.01710410317309808
b_SF 0.01758310213048409
b_GIDP 0.018025173247667 53
b_RBI 0.02411771146703025
b_CS 0.03042922858335669
b_BB 0.03186434514109146
b_SB 0.03262644103432081
b_HR 0.03651870713210674
b_SH 0.03774237403600242 5
b_SO 0.0393604061527094
b_3B 0.05413858745473779
b_H 0.07790234251391341
b_G 0.12320104000225089
p_W 0.14355509761287796
b_R 0.2055196352685047
```

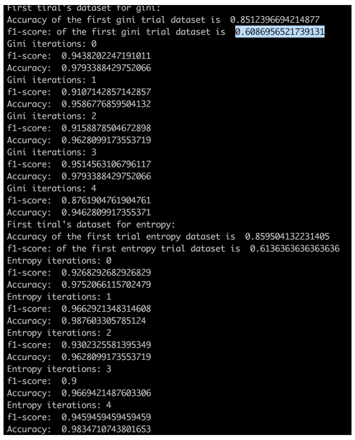
Above is the he individual weight for each features from python feature_importances. We can see there, the first a few features have the weight 0.0. This means those features are meaningless for determining whether a player is good or not and should be eliminated from the set. Once all of those zero-weight features get eliminated, re-run the program and there will be new zero-weight features appears. Do the elimination again and again and finally we have reached this set (final set we used):

```
playerid, p_HBP, p_SHO, b_AB, p_L, p_SV, p_GF, p_CG, p_R, b_HBP, b_GIDP,
b_BB, b_H, b_SH, b_IBB, b_2B, b_3B, b_SB, b_RBI, b_SO, b_G, p_W, b_R,
classification
```

And the models trained from the Python Decision Tree has the following results:

Model	Accuracy	F1-score
Gini	0.8512396694214877	0.6086956521739131
Entropy	0.859504132231405	0.6136363636363636

We can see there, the Accuracy and the F1 score from this model improved a lot. The results are acceptable. Some sample output for all 5 iterations:

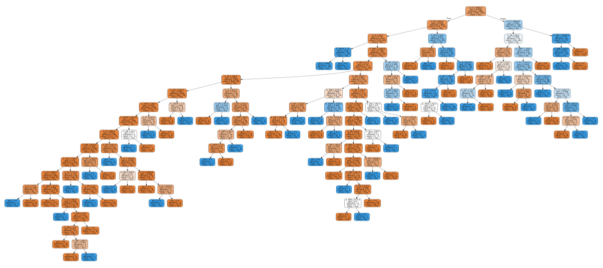


```
First tiral's dataset for gini:
Accuracy of the first gini trial dataset is  0.8512396694214877
f1-score: of the first gini trial dataset is  0.6086956521739131
Gini iterations: 0
f1-score:  0.9438202247191011
Accuracy:  0.9793388429752066
Gini iterations: 1
f1-score:  0.9107142857142857
Accuracy:  0.9586776859504132
Gini iterations: 2
f1-score:  0.9158878504672898
Accuracy:  0.9628099173553719
Gini iterations: 3
f1-score:  0.9514563106796117
Accuracy:  0.9793388429752066
Gini iterations: 4
f1-score:  0.8761904761904761
Accuracy:  0.9462809917355371
First tiral's dataset for entropy:
Accuracy of the first trial entropy dataset is  0.859504132231405
f1-score: of the first entropy trial dataset is  0.6136363636363636
Entropy iterations: 0
f1-score:  0.9268292682926829
Accuracy:  0.9752066115702479
Entropy iterations: 1
f1-score:  0.9662921348314608
Accuracy:  0.987603305785124
Entropy iterations: 2
f1-score:  0.9302325581395349
Accuracy:  0.9628099173553719
Entropy iterations: 3
f1-score:  0.9
Accuracy:  0.9669421487603306
Entropy iterations: 4
f1-score:  0.9459459459459459
Accuracy:  0.9834710743801653
```

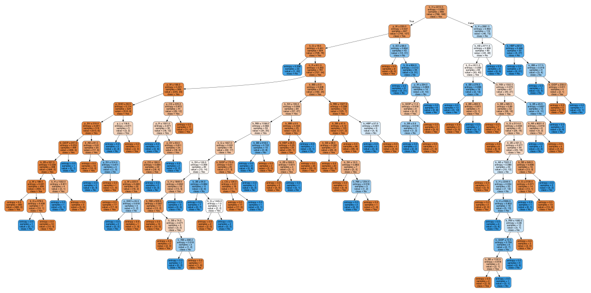
The Accuracy and the F1 score after iterations becomes high. This is because the 20% of the test data is randomly selected for the whole set. After the very first trial, the Accuracy and the F1 score is expected to increase because some of the test data are likely coming from the training data. This also show that the quality of our model are acceptable and can be used for prediction.

The sample of decision trees are shown below:

For Gini:



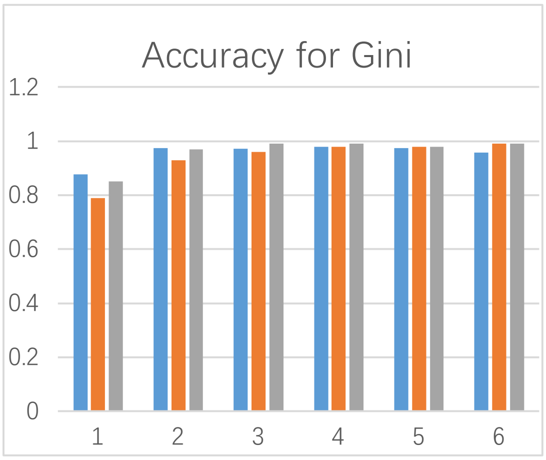
For Entropy:

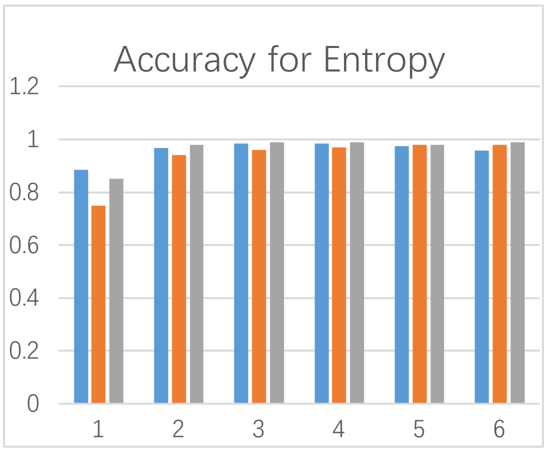


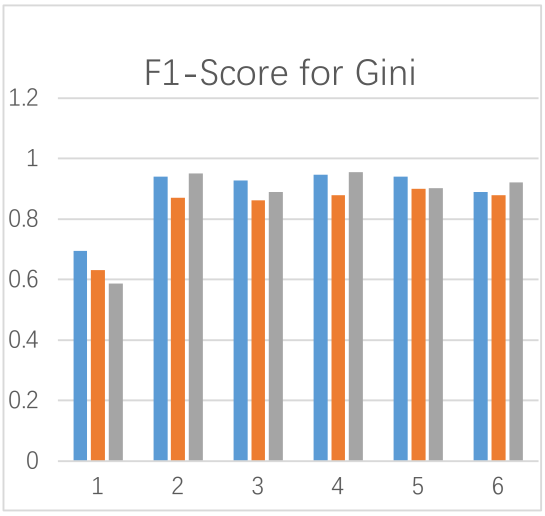
Blue indicates inducted = 0 and Red indicates inducted = 1

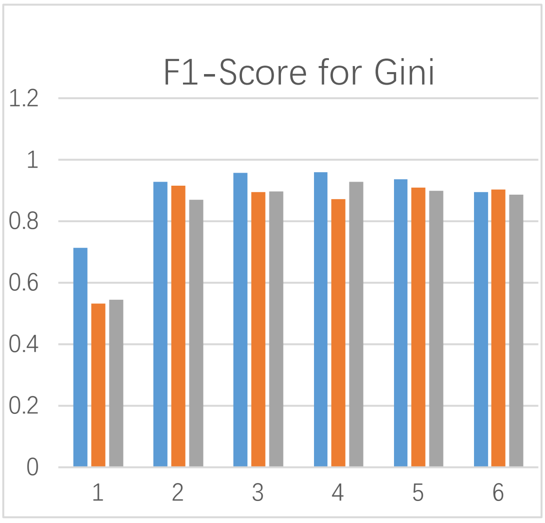
Part 2. Comparison:

For comparisons, the easiest way is to visualize the similarities and differences between all 3 choices of feature sets and their resulting models on their Accuracy and the F1 score in bar charts.









For all the plots above, blue bars are the values for the first choice (include votedby and votes), red bars are the values for the second choice (exclude votedby and votes) and grey bars are the values for the final choice (based on feature importance).

As we can see here, for gini accuracy, on the very first iteration, the second choice and the final choice starts low and first choice starts high. For later iteration, all three converged to a high value around 99%. However the first choice is lower compare to the rest because of the model's quality.

For gini F1 score, it follows a similar trend as the gini accuracy.

For entropy accuracy, on the very first iteration, the second choice and the final choice starts low and first choice starts high. For later iteration, all three converged to a high value around 95%. However the first choice are the final choice usually higher than the second choice. This might due to the fact that the second choice has meaningless features.

For entropy F1 score, it follows a similar trend as the entropy accuracy. However, on the 6th iteration, all three choices have a similar f1 score around 94%. This might be costed by the uncertain percentage of changes during each trial.