Ian Michel – Netid: ianjm2
CSE412 – Homework #4

Overall Accuracy

Type	balance.scale	nursery	led	synthetic social
Decision Tree	0.72	0.99	0.86	0.49
Random Forest	0.80	0.99	0.86	0.74

Classification Methods

- The Decision Tree implementation is a binary tree where the split is made on 1 attribute on one branch (the true branch), and the rest in another branch (the false branch). This is a one against the rest implementation. The Gini value is calculated at each true/false split, to determine which attribute/value has the greatest info-gain, which is then split to be the true side branch.
- Random Forest is implemented using the technique called *Forest-RI*, where an F value and number of trees was picked depending upon total run time (minutes) and the accuracy of the outcome. The F value is the number of attributes picked at random from the set of available to input to calculate the Gini value.
- For Random Forest, a majority vote is done by taking the prediction from each tree and taking the one which has the most counts as the winner.
- Performance enhancement: The Random Forest algorithm was implemented one thread per tree generation, to speed performance.

All Model Evaluation Measures

Test Data Sets

balance.scale	Accuracy	F-1 (Each Class – Tab Delimited)
Decision Tree	0.72	[0.00 0.82 0.78]
Random Forest	0.80	[0.00 0.84 0.83]

led	Accuracy	F-1 (Each Class – Tab Delimited)
Decision Tree	0.86	[0.77 0.90]
Random Forest	0.86	[0.77 0.90]

nursery	Accuracy	F-1 (Each Class – Tab Delimited)
Decision Tree	0.99	[0.99 0.98 0.99 1.00 0.00]
Random Forest	0.99	[0.99 1.00 0.99 1.00 0.00]

synthetic.social	Accuracy	F-1 (Each Class – Tab Delimited)
Decision Tree	0.49	[0.52 0.46 0.51 0.49]
Random Forest	0.74	[0.76 0.74 0.72 0.76]

Train Data Sets

balance.scale	Accuracy	F-1 (Each Class – Tab Delimited)
Decision Tree	1.00	[1.00 1.00 1.00]
Random Forest	0.92	[0.00 0.95 0.95]
led	**Accuracy**	**F-1 (Each Class – Tab Delimited)**
Decision Tree	0.86	[0.77 0.90]
Random Forest	0.86	[0.77 0.90]
nursery	**Accuracy**	**F-1 (Each Class – Tab Delimited)**
Decision Tree	1.00	[1.00. 1.00 1.00. 1.00 1.00]
Random Forest	1.00	[1.00. 1.00 1.00. 1.00 1.00]
synthetic.social	**Accuracy**	**F-1 (Each Class – Tab Delimited)**
Decision Tree	1.00	[1.00 1.00 1.00 1.00 1.00]
Random Forest	1.00	[1.00 1.00 1.00 1.00 1.00]

Parameters Chosen

There are 2 parameters for Random Forest implementation which are chosen: F-value and Number of Trees. There were no parameters for the Decision Tree because it built the tree automatically to the end nodes by itself.

F-value: This is the number of attributes chosen as input to calculate the Gini value within a single decision tree Gini calculation. An example might be an F value of 5. Then, 5 attributes of all N that are available at that node, are chosen at random. At first the number of attributes N is equal to the entire dataset. This number is pruned as the tree is generated, as data lines are isolated in the binary tree. Therefore, the set of attributes available at each node can possibly be different across nodes in the tree because sub trees do not include some data lines. The F-value was chosen primarily through experimentation. Please see the graphs below for the experiments done to pick the best F-values per dataset.

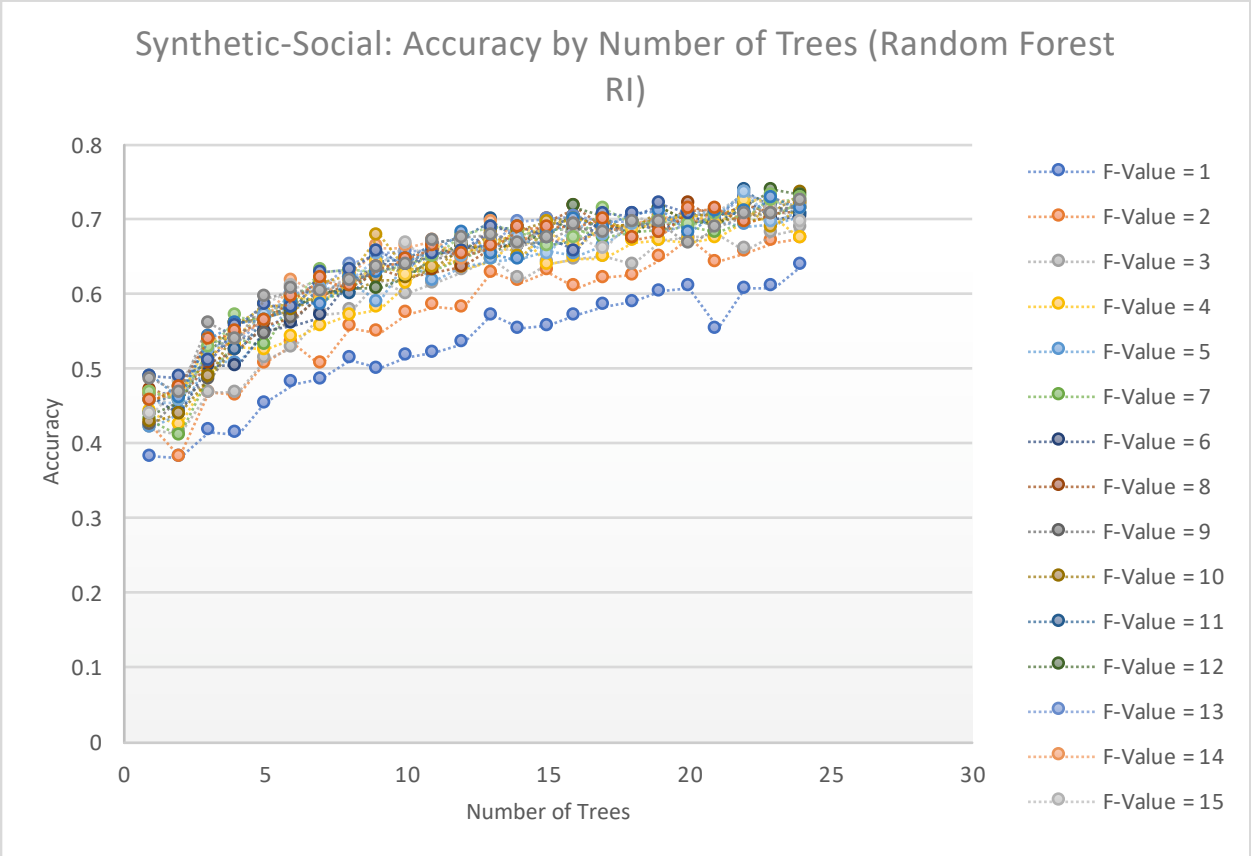
Number of Trees: The number of trees is the total number of classifiers to choose in this ensemble method. The output of each tree is input to the majority vote at the end for a majority prediction. This number per data set was chosen primarily based on experimentation. Please see the graphs below for the experiments done to pick the best Number of Trees per dataset.

The values that were found to consistently provide the highest accuracy are given below. The first number in the return is the F value, and the second the number of trees. For example, nursery has an F value of 8 and 2 trees were generated.

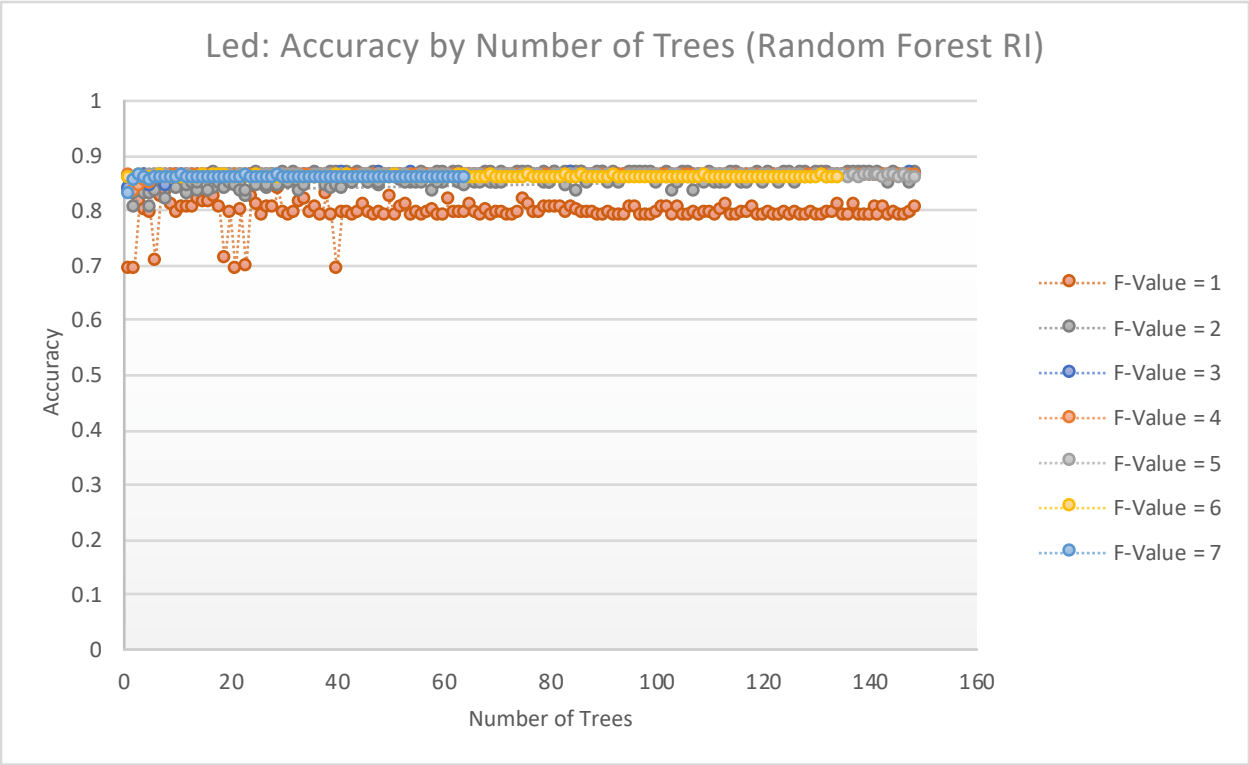
```
def getRunParameters (trainPath):
    if "nursery" in trainPath:
        return 7,25
```

```
    elif "synthetic.social" in trainPath:
        return 12,23
    elif "led" in trainPath:
        return 7,20
    elif "balance.scale" in trainPath:
        return 1,93
```

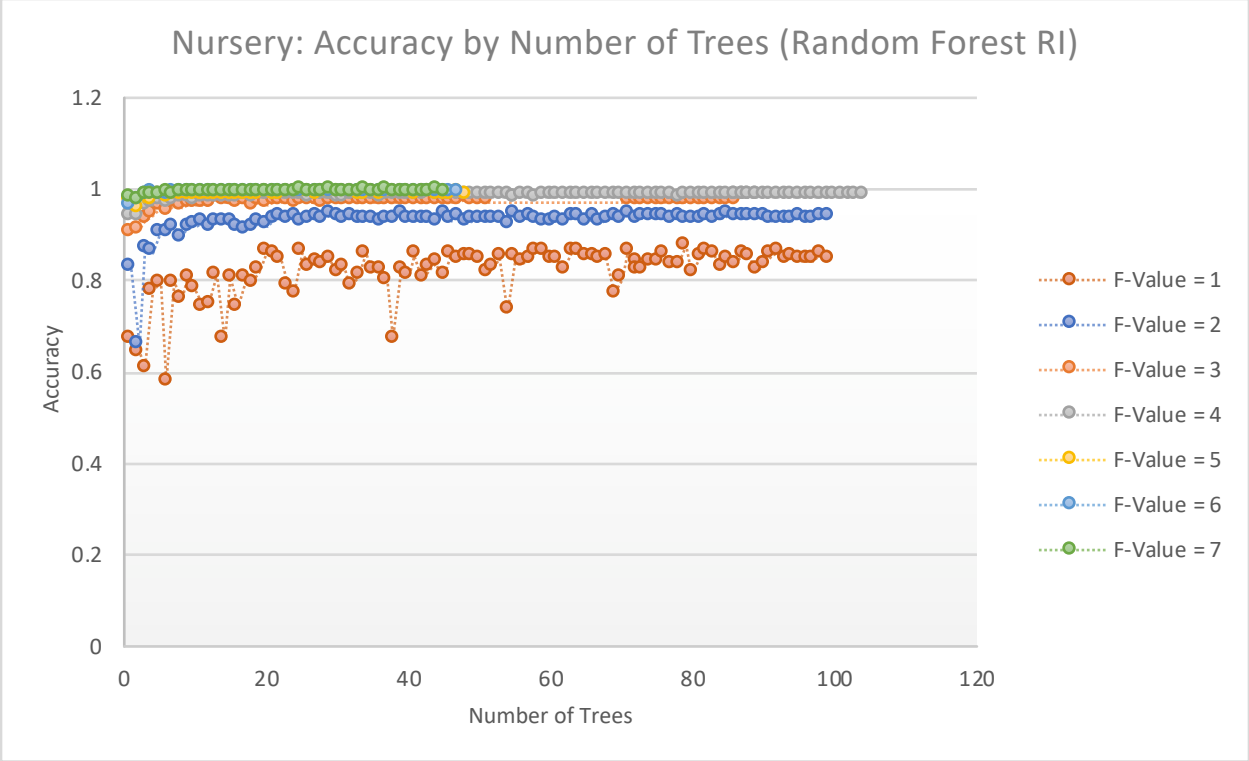
Synthetic-Social F-Value and Number of Trees Analysis



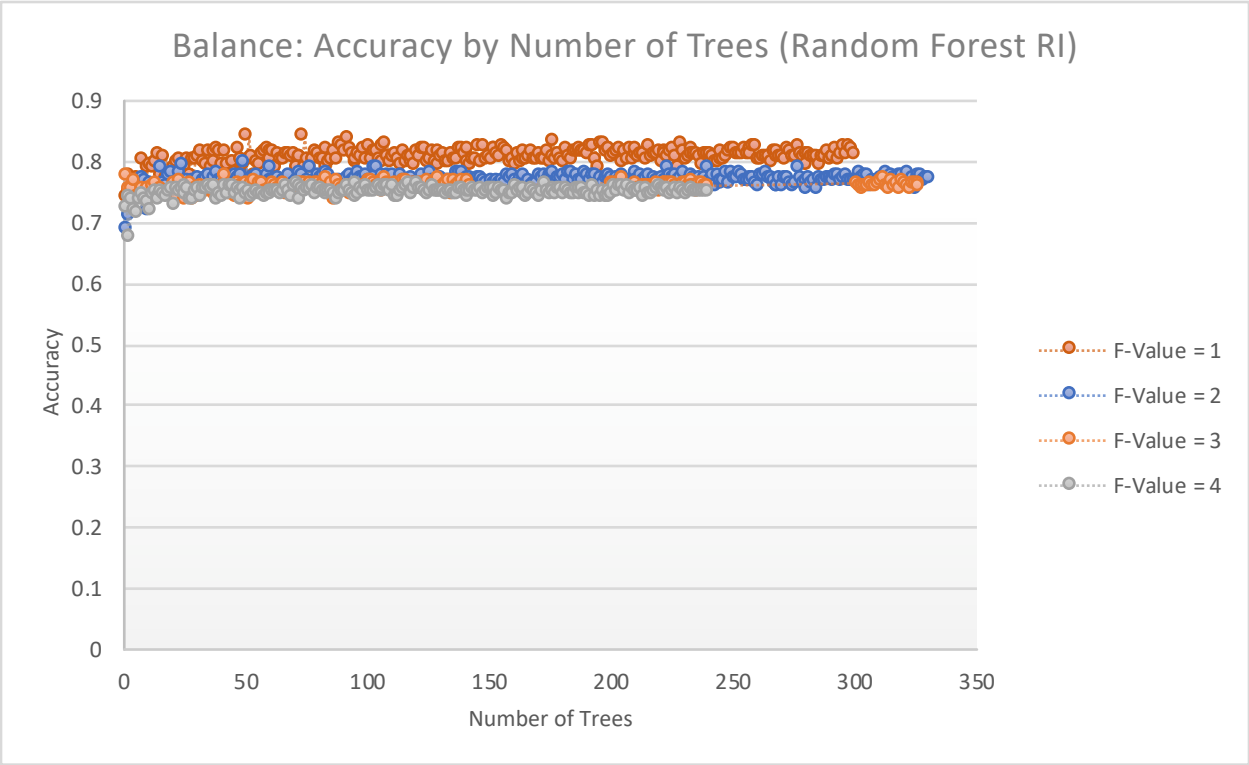
LED F-Value and Number of Trees Analysis



Nursery F-Value and Number of Trees Analysis



Balance F-Value and Number of Trees Analysis



Comments on F-Value and Number of Trees Analysis

A few striking observation were found that are important to note. The first is that synthetic social accuracy improved by a factor of nearly 2 fold as compared to the decision tree accuracy as the F-value increased and the Number of Trees increased. The only thing that stopped the algorithm from proceeding with greater accuracy than 74% was the time limit of 3 minutes. More research could be done to improve the performance of the algorithm to in the future provide greater accuracy here.

The second major observation here has to do with the balance data set. It turns out that an F-value of 1 is more accurate than an F-value of 4. Although the number of trees seemed not to matter for this data set. This means that random forest was more accurate when each Gini decision took into account 1 random attribute.

Conclusion

- The random forest in general improved the accuracy of prediction of each model against the test data set by these below percents, when fitting the runs into a 3-minute window:
    - Balance: 11%
    - Synthetic Social: 51%
- Random Forest implementation did not improve the other two datasets. In fact for nursery it only matched the normal decision tree implementation when F=1 was picked. Any other F-value actually decreased the accuracy (See the graph above):
    - Led: 0%
    - Nursery: 0%
- Random Forest outperforms the decision tree classifier overall in 50% of the data sets tested. Therefore, it would make sense to choose Random Forest as an appropriate

algorithm in most future cases. One important note is that Random Forest takes more time to run than Decision Tree implementation, because of the number of trees that it generates. The generation of trees increases the memory footprint and CPU usage. If performance is a high priority then Random Forest should not be used.