CS 760: Homework 2
Inducing Decision Trees - ID3

Sajika Gallege

Login: sgallege

3/16/2010

Part 1: Implementing ID3

In this homework I have Implement the algorithm in Mitchell's Table 3.1, augmenting it as explained below.

Continuous features in are handled as follows:

- Before starting the ID3 calc's for the root node, SORT each continuous feature in the current training set and make a list of the BOUNDARY VALUES (ie, those values halfway between two adjacent examples of different classes).
- In each cycle of the decision tree, score each of these boundary values (values used higher up in the path to the current node don't need to be rechecked nor do those between examples no longer in the current set of training examples, but this inefficiency is ok).

The ID3 can be called as

```
ID3Learner myID3_1 = new ID3Learner();
myID3_1.RUN_ID3(task.names, train_examples.data, splittingFuntion)
myID3_1.DUMP_TREE();
myID3_1.REPORT_TREE_SIZE();
myID3_1.CATEGORIZE(test_examples.data);
```

Part 1a: Handling Additional Splitting Functions

The Following Splitting features are implemented

Info_gain

random

Part 1b: Avoiding Overfitting

The following procedure is followed to avoid over fitting by post pruning, with settings Let *L*=100 and *K*=5

1. Given a training set, create a train' set and a *tuning* (or "pruning") set; place 20% of the examples in the tuning set.
2. Create a tree that fully fits the train' set; use info_gain as the scoring function for features. Call it *Tree*. Set *scoreOfBestTreeFound*to the score of *Tree* on the tuning set.
3. Number all the *interior* nodes from 1 to *N* (eg, using a "preorder" traversal; however the precise method for numbering nodes doesn't matter, as long as each interior node is counted once and only once).
4. *L* times do the following
   - Make a *copy* of *Tree*. Call it *CopiedTree*.
   - Uniformly pick a random number, *R*, between 1 and *K*.
   - *R* times, uniformly pick a random number, *D*, between 1 and *N*. Mark node *D* in *CopiedTree* as "to be deleted."

- Make a new copy of *CopiedTree*, this time pruning the tree whenever you encounter a "to be deleted" node. Replace the deleted node by the *majority* category for the subtree rooted at this node. *CopiedandPrunedTree* should point to this pruned tree.
- At this point, you will have a random subtree of *Tree*. Score it on the tuning set, and keep track of the best tree found.

5. Return the best tree found (on the *tune* set).

Part 1c: Making Decisions

Program output with tree printed 6 levels deep

```
HW2 task.names train_examples.data test_examples.data
```

Following is a sequence of training runs:

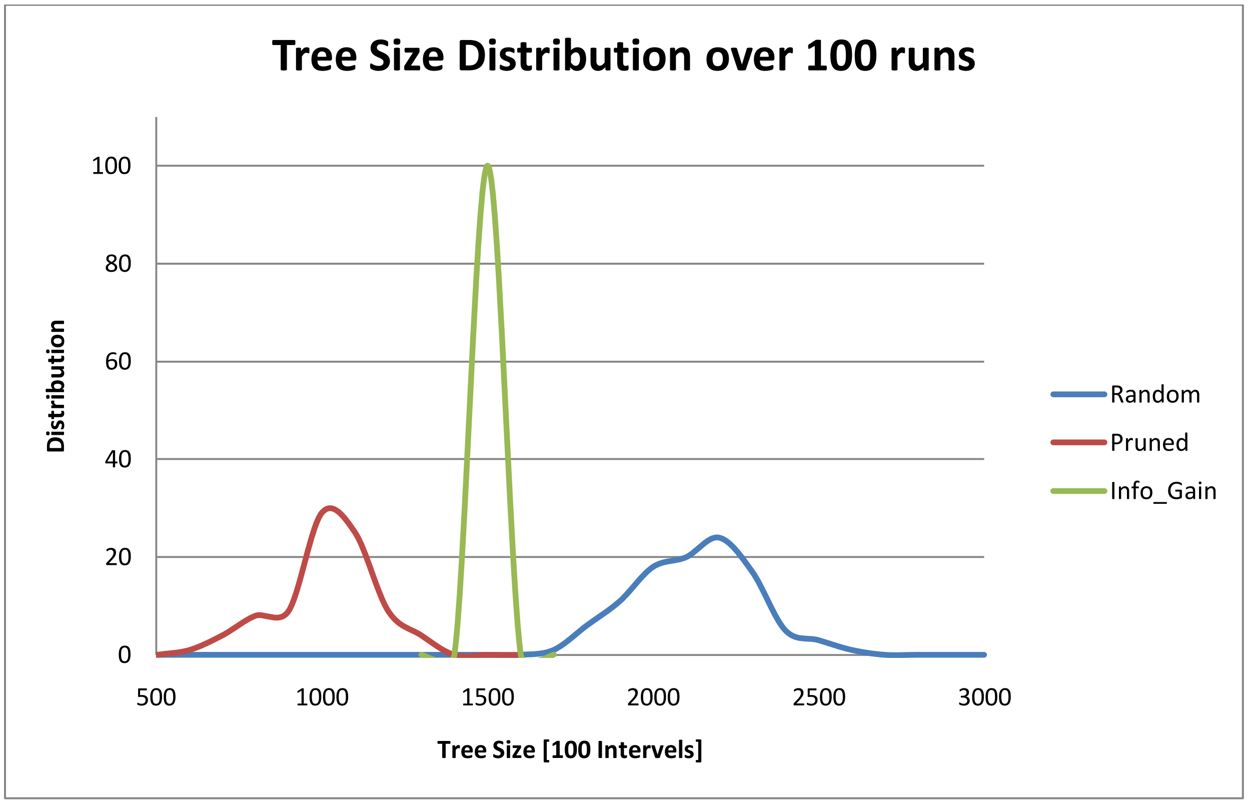
```
Info Gain

{Num_Gears<8.0?}
               N{Engine_Displacement<6.0?}
                       N(Y)
                       Y{Vehicle_Class?}
                               S_Truck(Y)
                               S_Car(Y)
                               Van(Y)
                               SUV(Y)
                               L_Truck(Y)
                               L_Car(Y)
                               M_Car{Drive_Type?}
                                       A(Y)
                                       4(Y)
                                       R{Manufacture_Country?} [Y]...
                                       F(Y)
                                       P(Y)
               Y{Manufacture_Country?}
                       France(Y)
                       USA{Air_Aspiration_Method?}
                               TC{Drive_Type?}
                                       A(N)
                                       4{Trans_Type?} [Y]...
                                       R(N)
                                       F(N)
                                       P(N)
                               NA{Engine_Displacement<6.0?}
                                       N{Trans_Type?} [Y]...
                                       Y{Trans_Type?} [N]...
                               SC(Y)
                       Germany{Air_Aspiration_Method?}
                               TC{Engine_Displacement<6.0?}
                                       N(Y)
                                       Y{Num_Cylynders<9.0?} [N]...
                               NA{Engine_Displacement<6.0?}
                                       N(Y)
                                       Y{Vehicle_Class?} [N]...
                               SC(N)
                       Italy(Y)
                       UK{Engine_Displacement<2.0?}
                               N(Y)
                               Y(N)
                       Korea{Air_Aspiration_Method?}
                               TC(N)
                               NA{Engine_Displacement<3.8499999999999996?}
                                       N(Y)
                                       Y{Variable_Valve_Timing?} [N]...
                               SC(N)
                       Japan{Num_Gears<7.0?}
                               N{Vehicle_Class?}
```

```
                                      S_Truck(Y)
                                      S_Car(N)
                                      Van(Y)
                                      SUV(Y)
                                      L_Truck(Y)
                                      L_Car(Y)
                                      M_Car(Y)
                              Y{Air_Aspiration_Method?}
                                      TC{Engine_Displacement<2.7?} [N]...
                                      NA{Variable_Valve_Timing?} [N]...
                                      SC(N)
                      Sweden{Engine_Displacement<3.25?}
                              N(Y)
                              Y{Vehicle_Class?}
                                      S_Truck(N)
                                      S_Car(N)
                                      Van(N)
                                      SUV{Drive_Type?} [Y]...
                                      L_Truck(N)
                                      L_Car(N)
                                      M_Car(N)

TREE_SIZE
Nodes: 581      Leaves: 817    Total: 1398

CATEGORIZE
Label   Predicted
Y       N
Y       N
Y       N
Y       N
Y       N
N       Y
N       Y
N       Y
Y       N
N       Y
Erros Rate=10.0/104= 0.09615384615384616

Pruned

Prune At [346 88 411 321 ]            Pruned Tree Nones=373 Score = 0.965287049399199
Prune At [28 377 211 232 238 ]                Pruned Tree Nones=339 Score = 0.965287049399199
Prune At [76 60 56 237 58 ]           Pruned Tree Nones=342 Score = 0.965287049399199
{Num_Gears<8.0?}
              Y{Manufacture_Country?}
                      France(Y)
                      USA{Air_Aspiration_Method?}
                              TC{Drive_Type?}
                                      A(N)
                                      4{Trans_Type?} [Y]...
                                      R(N)
                                      F(N)
                                      P(N)
                              NA{Engine_Displacement<6.0?}
                                      Y{Trans_Type?} [Y]...
                                      N{Trans_Type?} [Y]...
                              SC(Y)
                      Germany{Air_Aspiration_Method?}
                              TC{Num_Gears<7.0?}
                                      Y{Engine_Displacement<6.0?} [N]...
                                      N{Trans_Type?} [Y]...
                              NA{Engine_Displacement<6.0?}
                                      Y{Vehicle_Class?} [N]...
                                      N(Y)
                              SC(N)
                      Italy(Y)
                      UK{Engine_Displacement<2.0?}
                              Y(N)
                              N(Y)
                      Korea{Variable_Valve_Timing?}
                              Y{Air_Aspiration_Method?}
```

```
                                      TC(N)
                                      NA{Engine_Displacement<3.8499999999999996?} [N]...
                                      SC(N)
                              N(N)
                      Japan{Num_Gears<7.0?}
                              Y{Variable_Valve_Timing?}
                                      Y{Air_Aspiration_Method?} [N]...
                                      N{Engine_Displacement<3.7?} [N]...
                              N{Vehicle_Class?}
                                      S_Car(N)
                                      S_Truck(Y)
                                      Van(Y)
                                      L_Car(Y)
                                      L_Truck(Y)
                                      SUV(Y)
                                      M_Car(Y)
                      Sweden{Engine_Displacement<3.25?}
                              Y{Air_Aspiration_Method?}
                                      TC(N)
                                      NA{Drive_Type?} [Y]...
                                      SC(N)
                              N(Y)
              N{Engine_Displacement<6.0?}
                      Y{Vehicle_Class?}
                              S_Car(Y)
                              S_Truck(Y)
                              Van(Y)
                              L_Car(Y)
                              L_Truck(Y)
                              SUV(Y)
                              M_Car{Drive_Type?}
                                      A(Y)
                                      4(Y)
                                      R(N)
                                      F(Y)
                                      P(Y)
                      N(Y)

TREE_SIZE
Nodes: 342      Leaves: 522    Total: 864

CATEGORIZE
Label  Predicted
Y      N
Y      N
Y      N
Y      N
Y      N
N      Y
N      Y
N      Y
N      Y
Y      N
N      Y
Erros Rate=11.0/104= 0.10576923076923077

Random 1

{Engine_Displacement<6.0?}
              N{Engine_Displacement<4.0?}
                      N{Engine_Displacement<3.8499999999999996?}
                              N{Num_Gears<4.0?}
                                      N(Y)
                                      Y(N)
                              Y(Y)
                      Y(Y)
              Y{Engine_Displacement<4.65?}
                      N(Y)
                      Y{Num_Cylynders<5.0?}
                              N{Engine_Displacement<4.6?}
                                      N{Engine_Displacement<2.8?} [Y]...
                                      Y{Num_Cylynders<2.5?} [N]...
                              Y{Num_Cylynders<2.5?}
                                     N{Num_Gears<4.0?} [N]...
                                     Y(Y)

TREE_SIZE
Nodes: 794      Leaves: 1132   Total: 1926

CATEGORIZE
Label  Predicted
Y      N
Y      N
Y      N
Y      N
N      Y
Y      N
Erros Rate=6.0/104= 0.057692307692307696

Random 2

{Engine_Displacement<2.3499999999999996?}
               N{Engine_Displacement<2.8?}
                      N{Vehicle_Class?}
                              S_Truck{Engine_Displacement<3.8?}
                                     N(Y)
                                     Y{Num_Cylynders<9.0?} [N]...
                              S_Car{Engine_Displacement<2.45?}
                                     N{Variable_Valve_Timing?} [N]...
                                     Y(N)
                              Van{Engine_Displacement<3.95?}
                                     N{Engine_Displacement<3.1?} [Y]...
                                     Y{Num_Gears<6.0?} [Y]...
                              SUV{Engine_Displacement<4.0?}
                                     N{Num_Cylynders<8.0?} [Y]...
                                     Y{Num_Cylynders<7.0?} [Y]...
                              L_Truck{Num_Gears<8.0?}
                                     N(Y)
                                     Y{Manufacture_Country?} [Y]...
                              L_Car{Num_Cylynders<5.0?}
                                     N{Num_Cylynders<4.0?} [Y]...
                                     Y(Y)
                              M_Car{Air_Aspiration_Method?}
                                     TC{Engine_Displacement<1.4?} [Y]...
                                     NA{Engine_Displacement<3.25?} [N]...
                                     SC{Engine_Displacement<4.65?} [Y]...
                      Y{Num_Cylynders<2.5?}
                              N{Engine_Displacement<4.65?}
                                     N(N)
                                     Y{Engine_Displacement<3.5?} [N]...
                              Y(N)
               Y{Engine_Displacement<3.2?}
                      N(N)
                      Y{Num_Gears<5.5?}
                              N{Engine_Displacement<2.0?}
                                     N{Num_Cylynders<9.0?} [N]...
                                     Y{Air_Aspiration_Method?} [N]...
                              Y{Engine_Displacement<4.0?}
                                     N(N)
                                     Y{Num_Cylynders<8.0?} [N]...

TREE_SIZE
Nodes: 987      Leaves: 1324   Total: 2311

CATEGORIZE
Label  Predicted
Y      N
Y      N
Y      N
Y      N
N      Y
N      Y
Y      N
Erros Rate=7.0/104= 0.0673076923076923
```

Part 2: Experimenting with the Different Splitting Functions

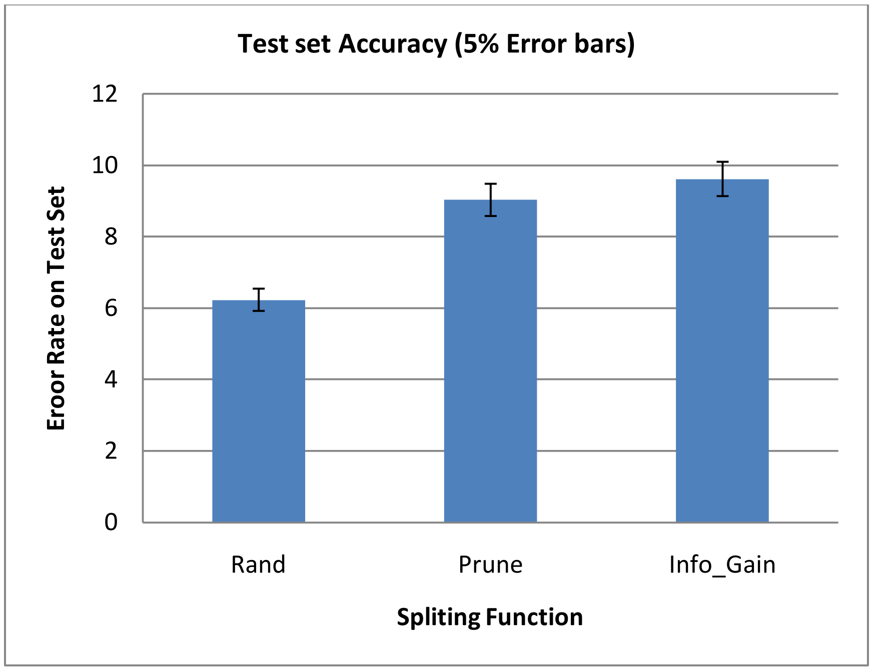
Part 2a: Using a Permutation Test to See if Using Information Gain is Better Than Randomly Choosing Features



The Pruned method clearly seems generate the smallest trees, size of the Info_gain trees fall in the middle while the random feature selected tree roughly twice the size of pruned trees. A random feature selected is highly likely to be larger in size than trees generated by prune or info gain.

According to the tests there the random splitting functions seems to perform better than the other two methods, although the difference small. This is somewhat counter intuitive. The learned trees are not printed due to length.

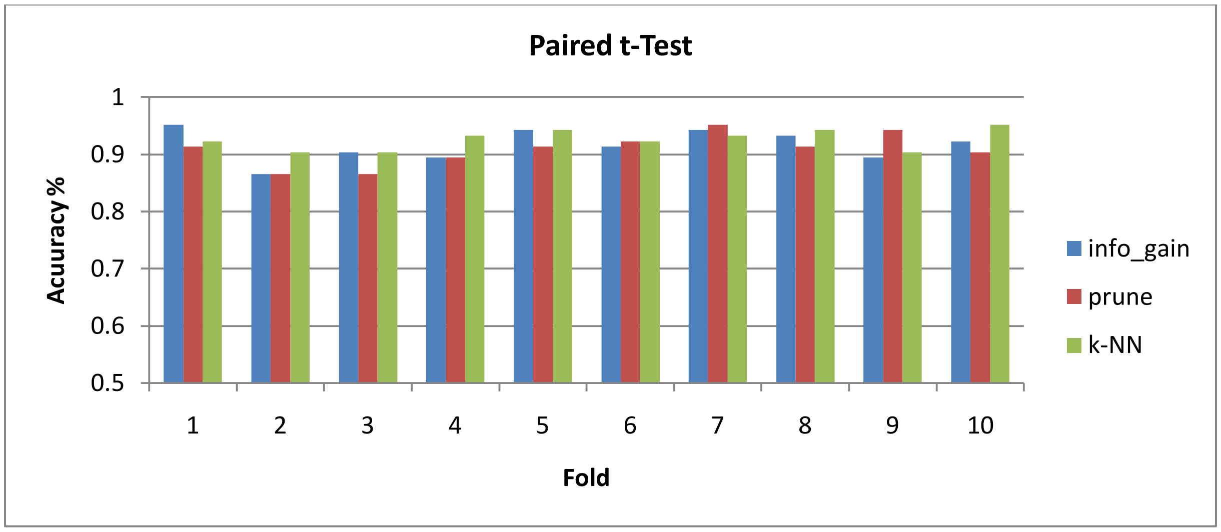
Part 2b: Measuring Generalization



The graph show the test set error rate with 95% confidence intervals as a function of the splitting technique used average over at least 100 runs. It is surprising to see that the random splitting has done better that the other two splitting technique. Although the difference small it is quite likely that a random tree gives better results on this test set.

Paired t-Test

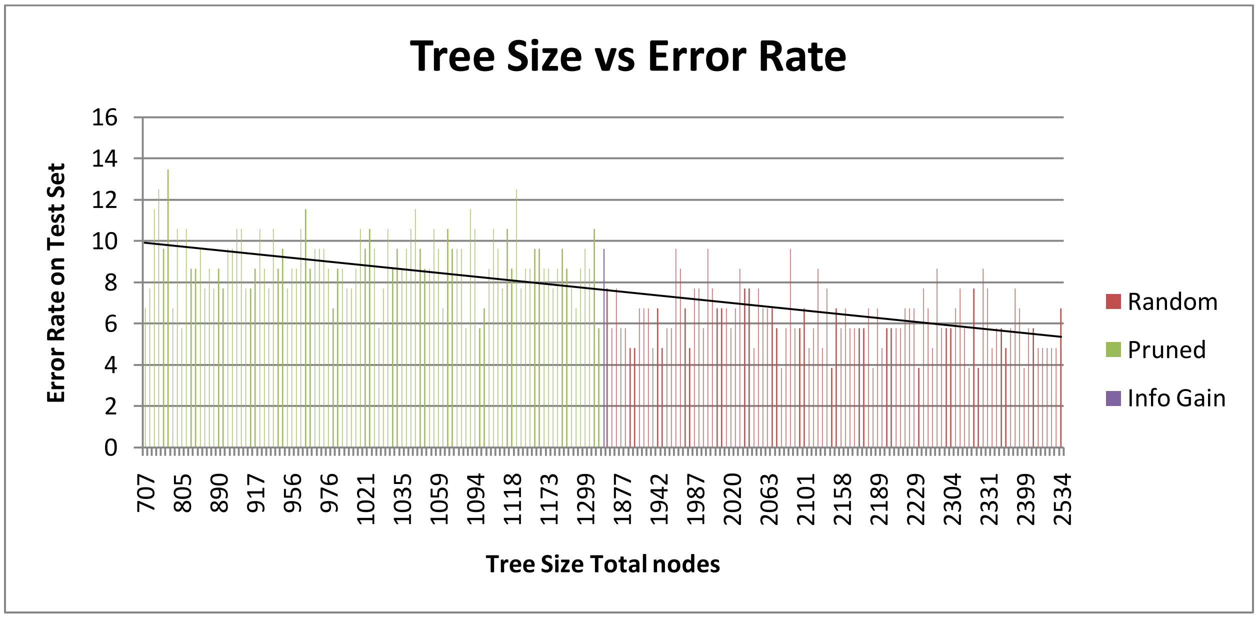
	IG	Pr	k-NN
1	0.951923	0.913462	0.923077
2	0.865385	0.865385	0.903846
3	0.903846	0.865385	0.903846
4	0.894231	0.894231	0.932692
5	0.942308	0.913462	0.942308
6	0.913462	0.923077	0.923077
7	0.942308	0.951923	0.932692
8	0.932692	0.913462	0.942308
9	0.894231	0.942308	0.903846
10	0.923077	0.903846	0.951923
Ave	0.916346	0.908654	0.925962
St Dev	0.027215	0.028398	0.017584



	Pr	k-NN	IG
IG	0		
Pr		0	
k-NN			0

None of the info_gain / pruned / k- NN are significantly different in paired t-Test

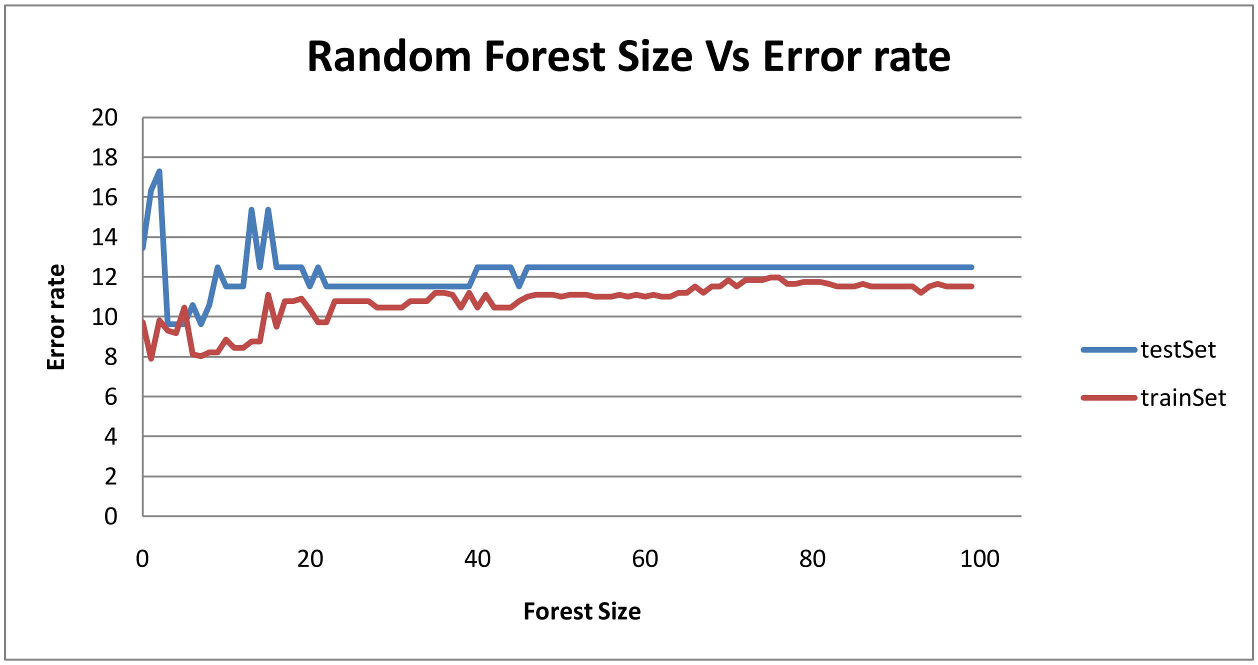
Part 2c: Investigating the Correlation between Tree Size and Tree Accuracy



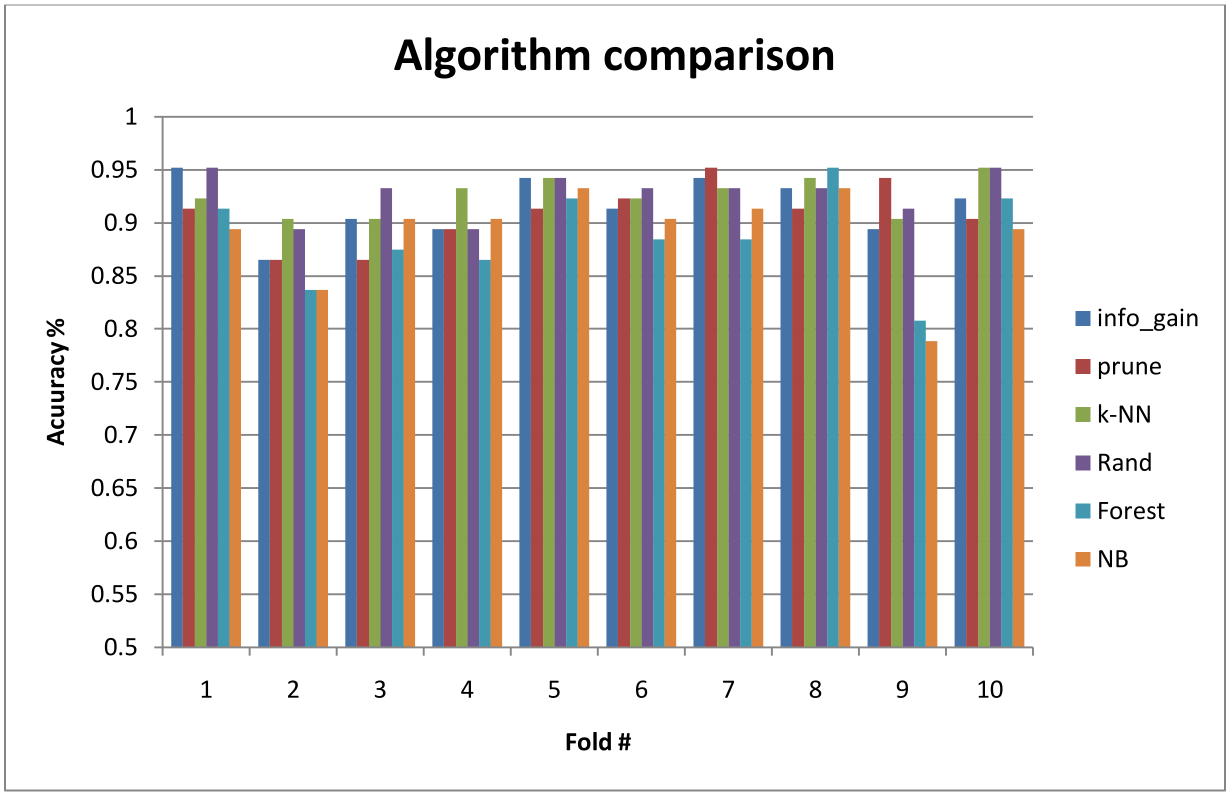
Larger trees seems to perform slightly better than the smaller, although the difference small. This is somewhat counter intuitive. So this data set dose not seem to support Occam's Razor hypothesis.

Part 3: Experimenting with Random Forests

Random Forest algorithm presented in class. Let i=5 , k=100

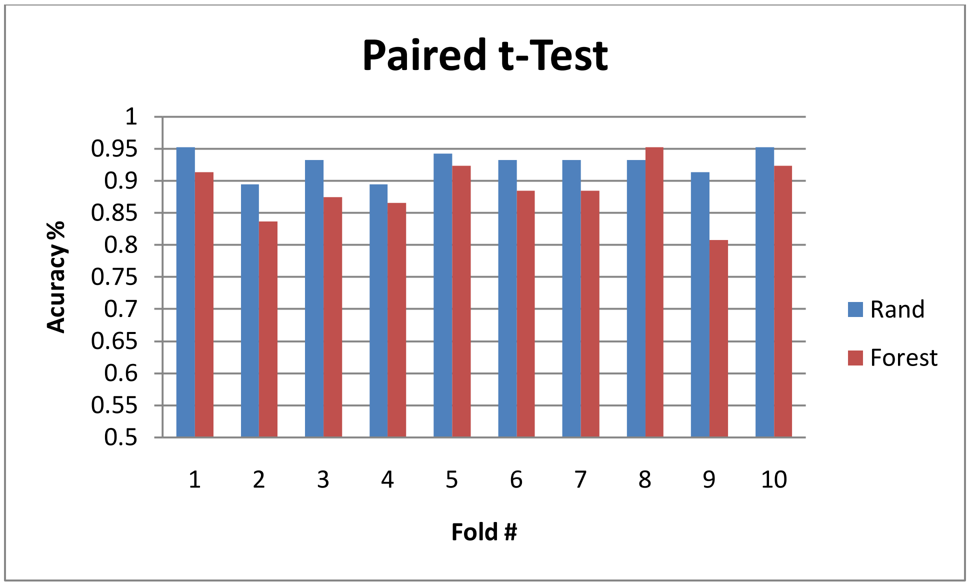


On the Test set error rate seems to converge at forest size 50 for an error rate of 12.5%. The Train set seems to still change a bit but averages around 11.2%



	info_gain	prune	k-NN	Rand	Forest	NB
1	0.951923	0.913462	0.923077	0.951923	0.913462	0.894231
2	0.865385	0.865385	0.903846	0.894231	0.836538	0.836538
3	0.903846	0.865385	0.903846	0.932692	0.875	0.903846
4	0.894231	0.894231	0.932692	0.894231	0.865385	0.903846
5	0.942308	0.913462	0.942308	0.942308	0.923077	0.932692
6	0.913462	0.923077	0.923077	0.932692	0.884615	0.903846
7	0.942308	0.951923	0.932692	0.932692	0.884615	0.913462
8	0.932692	0.913462	0.942308	0.932692	0.951923	0.932692
9	0.894231	0.942308	0.903846	0.913462	0.807692	0.788462
10	0.923077	0.903846	0.951923	0.951923	0.923077	0.894231
Ave	0.916346	0.908654	0.925962	0.927885	0.886538	0.890385
St Dev	0.027215	0.028398	0.017584	0.020895	0.043429	0.044688

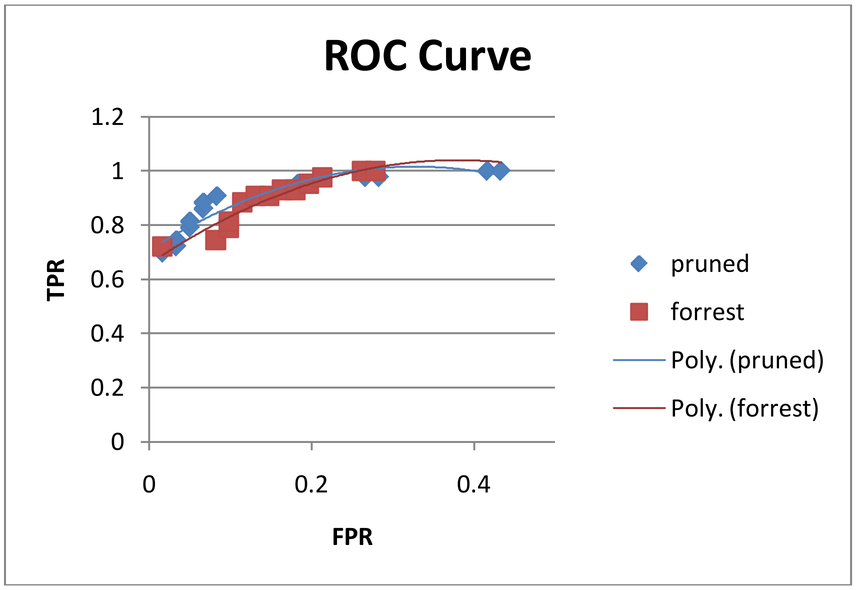
A comparison of all algorithms on the 10 folds Random split decision trees show the best accuracy, Compared to the other algorithms Random forest seems to perform poorly on this data set.



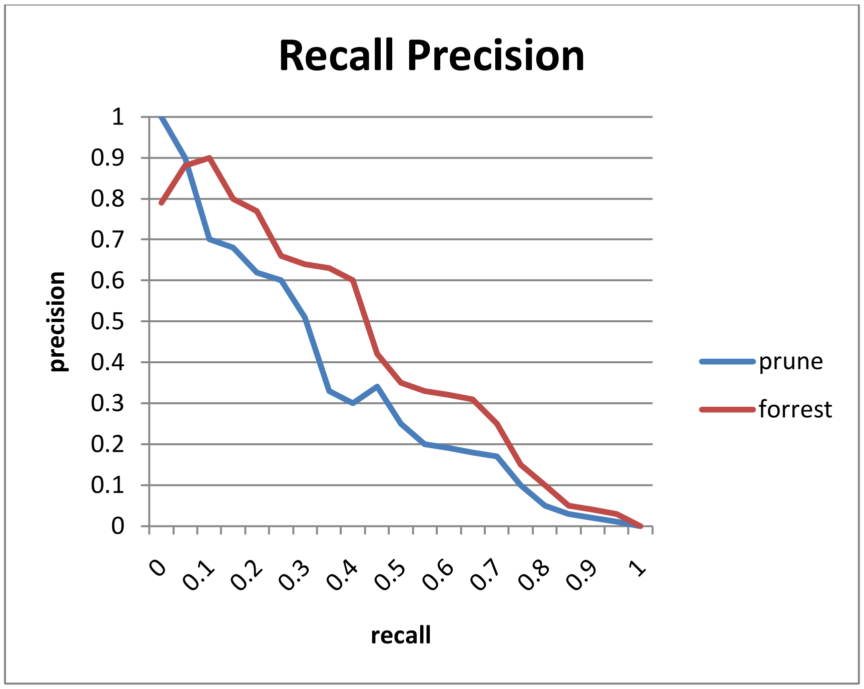
	Rand	Forest
1	0.951923	0.913462
2	0.894231	0.836538
3	0.932692	0.875
4	0.894231	0.865385
5	0.942308	0.923077
6	0.932692	0.884615
7	0.932692	0.884615
8	0.932692	0.951923
9	0.913462	0.807692
10	0.951923	0.923077
Ave	0.927885	0.886538
St Dev	0.020895	0.043429
p-Value		0.002769

The accuracy difference between Random feature selection and Random Forrest is statistically significant.

Part 4: ROC and Recall-Precision Curves



Both algorithms seem to be similar in the ROC Curve Pruned performs better at the start but Forest gets better later.



Forest shows better performance on the PR curve over Prune