DATA ANALYSIS REPORT OF CLASSIFIER ALGORITHMS USING WEKA

BY:

SHEETAL MORE

A20327606
smore2@hawk.iit.edu

I. TERMS AND DEFINITIONS

Confusion Matrix

It is a specific table layout that allows visualization of the performance of a classifier algorithm. It is of the form as shown here

	Predicted positive	Predicted negative
Actual positive	TP	FP
Actual negative	FN	TN

where TP = True Positive, TN = True Negative, FP = False Positive, FN = False negative. The diagonal values TP+TN correspond to the correctly predicted output instances count.

TP Rate

TRP or sensitivity is the rate of true positives (instances correctly classified as a given class). TPR=TP/(TP+FN.)

FP Rate

Rate of false positives (instances falsely classified as a given class). FPR = FP/(FP+TN)

Precision

Proportion of instances that are truly of a class divided by the total instances classified as that class

Recall

Proportion of instances classified as a given class divided by the actual total in that class (equivalent to TP rate)

F-Measure

A combined measure for precision and recall calculated as 2 * Precision * Recall / (Precision + Recall)

ROC Area

Receiver Operating Characteristic or ROC Curve is a curve created by plotting the true positive rate (TP Rate) against the false positive rate (FP Rate) at various threshold settings. An "optimal" classifier will have ROC area values approaching 1, with 0.5 being comparable to "random guessing" (similar to a Kappa statistic of 0).

Kappa Statistic

The Kappa statistic (or value) is a statistic measure that compares an observed accuracy with an expected accuracy or a random chance. It is used not only to evaluate a single classifier, but also to evaluate classifiers amongst themselves. Observed Accuracy is simply the number of instances that were classified correctly throughout the entire confusion matrix. Expected Accuracy is the accuracy that any random classifier would be expected to achieve based on the confusion matrix. Kappa statistic = (observed accuracy - expected accuracy)/(1 - expected accuracy). A kappa value > 0.75 is considered as excellent, 0.40-0.75 as fair to good, and < 0.40 as poor.

1 SE Rule

1-SE or One-Standard Error rule is known to favor simpler trees although possibly leading to lower predictive accuracy. [Used in useOneSE parameter of Simple CART classifier]

Pruning

Pruning is a process to reduce the size of decision trees by removing sections or nodes of the tree that provide little power to classify instances. Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting.

Fractional Instances

Whenever a split on an attribute with a missing value is considered, each incomplete instances is virtually replaced by several instances corresponding to all possible split outcomes with a fractional copy count. (Simple CART uses this method to handle missing values). The different "fractions" of the instance are weighted by the popularity of each branch.

Overfitting Model

A model that fits the training dataset too well can have a poorer generalization error than a model with a higher training error.

Underfitting Model

A model with a very small tree size but with large training and test error rates.

II. ALGORITHMS

1. SimpleCART

This algorithm implements minimal cost-complexity pruning.

Parameters:

Name	Description	Default Value
debug	If set to true, classifier may output additional info to the console. This does not have any impact on the classifier's performance.	FALSE
heuristic	If heuristic search is used for binary split for nominal attributes in multiclass problems.	TRUE
minNumObj	The minimal number of observations at the terminal nodes.	2.0
numFoldsPruning	The number of folds in the internal cross-validation.	5
seed	The random number seed to be used.	1
sizePer	The percentage of the training set size (0-1, 0 not included).	1.0
useOneSE	Use the 1SE (Standard Error) rule to choose a simple decision tree.	FALSE
usePrune	Use minimal cost-complexity pruning.	TRUE

2. Decision Stump

This algorithm does regression based on mean-squared error or classification based on entropy on only attribute of the dataset and models one level decision tree with three nodes (a root node and two terminal or leaf nodes) always. Missing values are treated as a separate value.

Parameters

Name	Description	Default Value
debug	If set to true, classifier may output additional info to the console. This does not have any impact on the classifier's performance.	FALSE

Notes:

- *All the **default** parameter values should be considered, unless explicitly mentioned in any of the forth-coming analysis in this report.*
- *Since **debug** only provides additional information, **seed** only changes dataset by re-sampling and **sizePer** only sets the training set's size, these parameters will be less helpful in the analysis of the classifier's performance. Thus, we will be proceeding our analysis by changing the remaining parameters to observe their impact.*
- *Decision Stump classifier does not have any significant parameters that could impact the classifier performance.*

III. DATASETS

1. Iris Dataset

Number of instance: 150
Number of attributes: 5 (4 Numerical + 1 Nominal)

No.	Attributes	Minimum Value	Maximum Value	Mean	Standard Deviation
1	Sepallength	4.3	7.9	5.843	0.828
2	Sepalwidth	2	4.4	3.054	0.434
3	Petallength	1	6.9	3.759	1.764
4	Petalwidth	0.1	2.5	1.199	0.763

Class Attribute:

No.	Attribute	Values	Count
5	Class	Iris-setosa	50
		Iris-versicolor	50
		Iris-virginica	50

Class Distribution: 33.3% for each of 3 classes.

2. Vote Dataset

Number of instances: 435
Number of attributes: 17 (17 Nominal)

No.	Attribute	Values	Count
1	handicapped-infants	n	236
		y	187
2	water-project-cost-sharing	n	192
		y	195
3	adoption-of-the-budget-resolution	n	171
		y	253
4	physician-fee-freeze	n	247
		y	177
5	el-salvador-aid	n	208
		y	212
6	religious-groups-in-schools	n	152
		y	272

7	anti-satellite-test-ban	n	182

		y	239
8	aid-to-nicaraguan-contras	n	178
		y	242
9	mx-missile	n	206
		y	207
10	immigration	n	212
		y	216
11	synfuels-corporation-cutback	n	264
		y	150
12	education-spending	n	233
		y	171
13	superfund-right-to-sue	n	201
		y	209
14	crime	n	170
		y	248
15	duty-free-exports	n	233
		y	174
16	export-administration-act-south-africa	n	62
		y	269

Class attribute:

No.	Attribute	Values	Count
17	class	democrat	267
		republican	168

3. Labor Dataset

Number of instances: 57
Number of attributes: 17 (7 Numerical + 10 Nominal)

No.	Nominal Attributes	Minimum Value	Maximum Value	Mean	Standard Deviation
1	duration	1	3	2.161	0.708
2	wage-increase-first-year	2	7	3.804	1.371
4	wage-increase-third-year	2	5.1	3.913	1.304
6	working-hours	27	40	38.039	2.506
8	standby-pay	2	14	7.444	5.028
9	shift-differential	0	25	4.871	4.544
11	statutory-holidays	9	15	11.094	1.26

No.	Numeric Attribute	Values	Count
5	cost-of-living-adjustment	none	22
		tcf	8
7	pension	none	11
		ret-allw	4
		empl-cntr	12
10	education-allowance	yes	10
		no	12
12	vacation	below_average	18
		average	17
		generous	16
13	long-term-disability-assistance	yes	20
		no	8
14	contribution-to-dental-plan	none	9
		half	15
		full	13
15	bereavement-assistance	yes	27
		no	3
16	contribution-to-health-plan	none	8
		half	9
		full	20

Class attribute:

No.	Attribute	Values	Count
17	class	bad	20
		good	37

4. Diabetes Dataset

Number of instances: 768
Number of Attributes: 9 (8 Numerical + 1 Nominal)

No.	Attributes	Minimum Value	Maximum Value	Mean	Standard Deviation
1	preg	0	17	3.845	3.37
2	plas	0	199	120.895	31.973
3	pres	0	122	69.105	19.365
4	skin	0	99	20.536	15.952
5	insu	0	846	79.799	115.244
6	mass	0	67.1	31.993	7.884
7	pedi	0.078	2.42	0.471	0.331
8	age	21	81	33.241	11.76

Class attribute:

No.	Attribute	Values	Count
9	class	tested_negative	500
		tested_positive	268

IV. ANALYSIS OF IRIS DATASET

Test Options

On Training Set

The classifiers built the model based on the input dataset for training. Since we evaluate the model with the same training set, we expect the accuracy to be high. Though, the results do not represent the accuracy of the classifier for an unseen independent dataset, we do this analysis to observe the classifier's behavior on the trained model.

Supplied Test Set

The Iris dataset was manually divided into two sets, training and test sets, in the ratio 2:1 of original Iris dataset so that there are no common instances between the two sets. The classifiers were built using this training set and then the test set whose instances are unseen for the classifiers was supplied and the performance of the classifier is observed. Also, the class distributions of Iris-setosa : Iris-versicolor : Iris-virginica are maintained in 1:1:1 ratio for the test dataset. Number of instances in training and test are 99 and 51 respectively.

Percentage Split

The Iris dataset is split into 2:1 ratio randomly based on this option as the percentage split that we have used here is 66. Though the input dataset has the class distribution of Iris-setosa : Iris-versicolor : Iris-virginica as 1:1:1, the random split could take any instances in training and test and thus there is a possibility for a skewed class distribution in the training or test sets that could not lead to a fair evaluation of the classifier.

10-fold Cross Validation

The original dataset is used for training the classifier using the 10-fold cross validation method. In this, the dataset is split into 10 sets and for each iteration 9 sets will be the training sets and the remaining one will be the test set. Since this is 10-fold, there will be 10 iterations run before the evaluation completes. This means, each of the 10 sets is used as a training set for 9 times and a test set for one time. And the classifier model generalizes the classifiers output based on the overall performance of the classifier. Moreover, this method splits the dataset into stratified subsamples and thus, it ensures a fair class distribution in available in each of the subsets similar to the original dataset.

Analysis of 10-fold Cross Validation

Below is an instance of subsets derived from the original Iris data set of 150 instances. It can be easily seen that datasets are divided into 10 equal subsets preserving a fair class distribution in each of the folds. Therefore, for the first iteration, folds 1-9 will be training set and fold 10 will be the test set. In the second iteration, folds 1-8 and 10 will be the training sets and fold 9 will be the test set. Similarly, 10 such iterations will be performed and the output will be shown for the overall performance. This enables smaller variance in the training and test dataset.

Filter = StratifiedRemoveFolds	Instances	Class Distribution		
Fold		Iris-setosa	Iris-versicolor	Iris-virginica
1	15	5	5	5
2	15	5	5	5
3	15	5	5	5
4	15	5	5	5
5	15	5	5	5
6	15	5	5	5
7	15	5	5	5
8	15	5	5	5
9	15	5	5	5
10	15	5	5	5

Performance Analysis of the Classifiers

A. Decision Stump

As the classifier chooses only one attribute with respect to the minimal entropy and yields only two branches leading to the two terminal nodes, it can classify the data into only two of the three cases. Since this dataset has three different possible class values, the model always ignores one of the three classes (iris-virginica was the third class ignored in above model). Even by repeating the experiment with different attributes of the dataset (by removing the attributes from the dataset), the classifier doesn't seem to perform any better to classify Iris-virginica. Since there are no other significant parameters for this filter, this filter is not suitable for the Iris dataset.

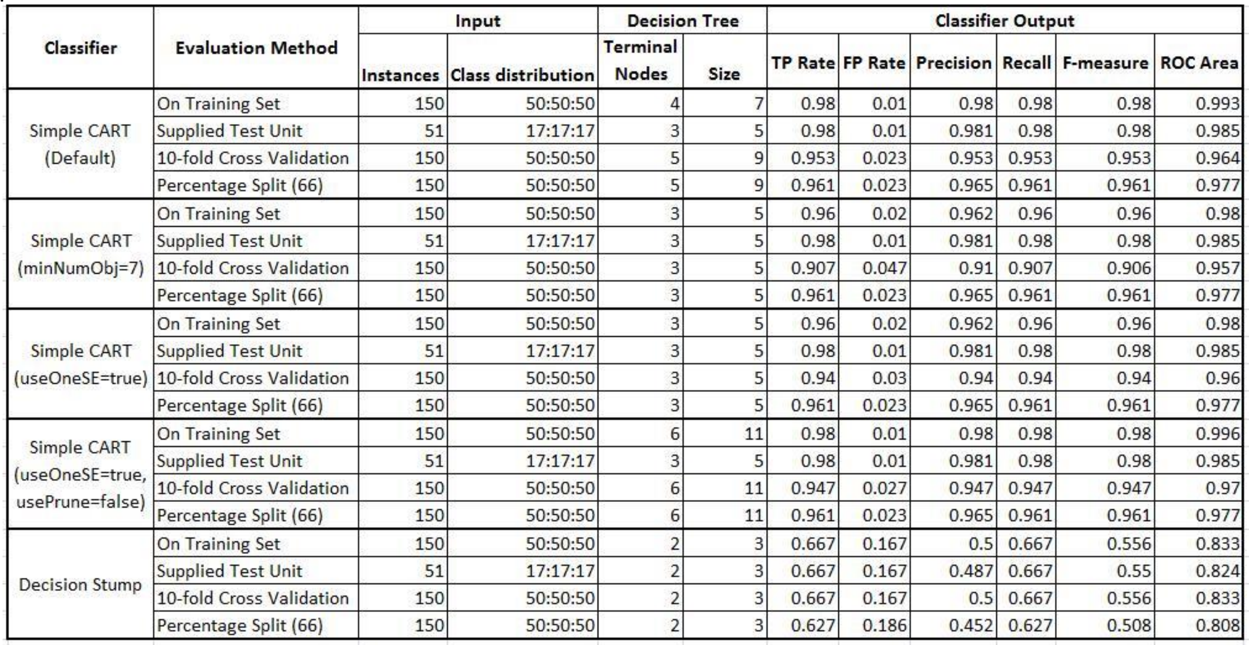
B. Simple CART

Unlike Decision Stump, the Simple CART algorithm clearly classifies the dataset into three different classes and with a better accuracy as expected. As the size of the tree and terminal nodes increase, we can observe that the decision tree is becoming more complex without a substantial increase in the performance. Consider, the sample run in the below table in which the *useOneSE = true* and *usePrune=false*. The size of the tree is 11 with 6 terminal nodes. The classifier output on training set result has TP rate of 0.98 and ROC Area 0.996 whereas using 10-fold cross validation method, the TP rate is 0.947 with ROC area as 0.97. This is an example of *overfitting* problem in which the classifier model has *'memorized'* the training model instead of *'learning'* from the training set. Moreover, by setting *UsePrune = false* leads to larger or more complex decision tree but better accuracy on the training set. This is because, the decision tree stops only when the terminal nodes are in their purest form without pruning enabled. This means, the model represents the training dataset better.

Also, we can observe that as *minNumObj* is increased from 2(default value) to 7, the accuracy and the size of the tree reduces. Consider the 10-fold cross validation results for default and *minNumObj*=7 sample runs. As the *minNumObj* value was increased, the terminal nodes of the tree reduced from 5 to 3 with decrease in tree size from 9 to 5. Also the TP rate and ROC Area decreased from 0.953 and 0.964 to 0.907 and 0.957. This is because, as the decision tree in *minNumObj*=7 was stopped branching when there were seven observations at the terminal nodes. In other words, it stopped before the terminal nodes were further recurred to end in purer observations, the accuracy of the decision tree is affected. Besides , by setting the parameter *useOneSE* = true, a simpler decision tree is chosen possibly leading to lower prediction accuracy as the standard error range is accepted for the decision tree.

Apart from the impact of the parameters. we can see that for the Supplied Test set cases, this algorithm is classifying the dataset much better.

Sample output of each of the test experiment for each classifier with respect to the respective classifier parameters are mentioned in the below table:



Classifier	Evaluation Method	Input		Decision Tree		Classifier Output					
		Instances	Class distribution	Terminal Nodes	Size	TP Rate	FP Rate	Precision	Recall	F-measure	ROC Area
Simple CART (Default)	On Training Set	150	50:50:50	4	7	0.98	0.01	0.98	0.98	0.98	0.993
	Supplied Test Unit	51	17:17:17	3	5	0.98	0.01	0.981	0.98	0.98	0.985
	10-fold Cross Validation	150	50:50:50	5	9	0.953	0.023	0.953	0.953	0.953	0.964
	Percentage Split (66)	150	50:50:50	5	9	0.961	0.023	0.965	0.961	0.961	0.977
Simple CART (minNumObj=7)	On Training Set	150	50:50:50	3	5	0.96	0.02	0.962	0.96	0.96	0.98
	Supplied Test Unit	51	17:17:17	3	5	0.98	0.01	0.981	0.98	0.98	0.985
	10-fold Cross Validation	150	50:50:50	3	5	0.907	0.047	0.91	0.907	0.906	0.957
	Percentage Split (66)	150	50:50:50	3	5	0.961	0.023	0.965	0.961	0.961	0.977
Simple CART (useOneSE=true)	On Training Set	150	50:50:50	3	5	0.96	0.02	0.962	0.96	0.96	0.98
	Supplied Test Unit	51	17:17:17	3	5	0.98	0.01	0.981	0.98	0.98	0.985
	10-fold Cross Validation	150	50:50:50	3	5	0.94	0.03	0.94	0.94	0.94	0.96
	Percentage Split (66)	150	50:50:50	3	5	0.961	0.023	0.965	0.961	0.961	0.977
Simple CART (useOneSE=true, usePrune=false)	On Training Set	150	50:50:50	6	11	0.98	0.01	0.98	0.98	0.98	0.996
	Supplied Test Unit	51	17:17:17	3	5	0.98	0.01	0.981	0.98	0.98	0.985
	10-fold Cross Validation	150	50:50:50	6	11	0.947	0.027	0.947	0.947	0.947	0.97
	Percentage Split (66)	150	50:50:50	6	11	0.961	0.023	0.965	0.961	0.961	0.977
Decision Stump	On Training Set	150	50:50:50	2	3	0.667	0.167	0.5	0.667	0.556	0.833
	Supplied Test Unit	51	17:17:17	2	3	0.667	0.167	0.487	0.667	0.55	0.824
	10-fold Cross Validation	150	50:50:50	2	3	0.667	0.167	0.5	0.667	0.556	0.833
	Percentage Split (66)	150	50:50:50	2	3	0.627	0.186	0.452	0.627	0.508	0.808

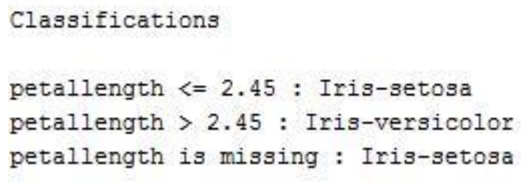
Impact of missing values

Missing values were introduced to the original Iris dataset in both stratified as well as skewed manner across classes and the outcomes were observed.

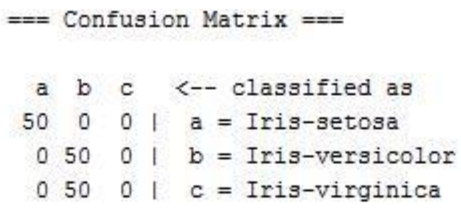
1. Decision Stump

Test Option: 10-fold Cross Validation   Input: 150 instances

a. Without Missing Values:

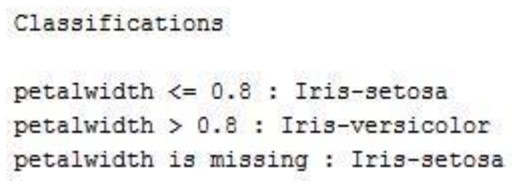


```
Classifications

petallength <= 2.45 : Iris-setosa
petallength > 2.45 : Iris-versicolor
petallength is missing : Iris-setosa
```

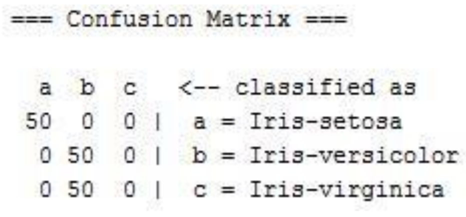


```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 50  0 |  b = Iris-versicolor
  0 50  0 |  c = Iris-virginica
```

b. To 10 % of instances, the missing values were introduced to petallength attribute across all classes:

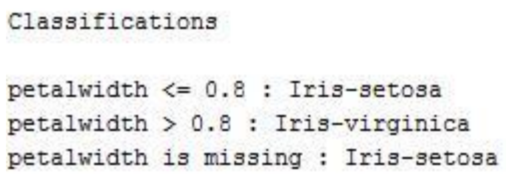


```
Classifications

petalwidth <= 0.8 : Iris-setosa
petalwidth > 0.8 : Iris-versicolor
petalwidth is missing : Iris-setosa
```

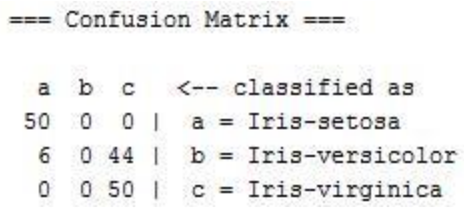


```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 50  0 |  b = Iris-versicolor
  0 50  0 |  c = Iris-virginica
```

b. To 10 % of instances, the missing values were introduced to petalwidth attribute across all classes (in addition to b):



```
Classifications

petalwidth <= 0.8 : Iris-setosa
petalwidth > 0.8 : Iris-virginica
petalwidth is missing : Iris-setosa
```



```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  6  0 44 |  b = Iris-versicolor
  0  0 50 |  c = Iris-virginica
```

Analysis:

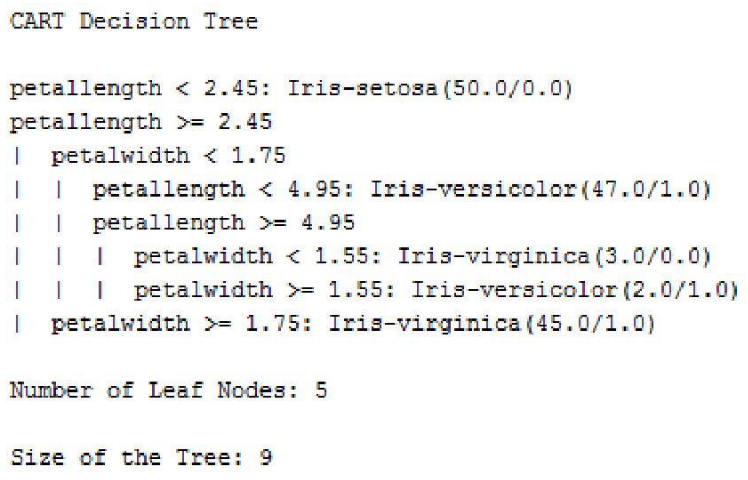
As more and more missing values were introduced to the dataset, the entropy value accordingly were impacted and the decision stump accordingly chose different attribute to classify the values better. Moreover, if there is any class with more missing values, then the error rate for that class was more. In addition to the algorithm's inability to classify the dataset into three different classes, these missing values still made this classifier unfit for further analysis.

2. Simple CART

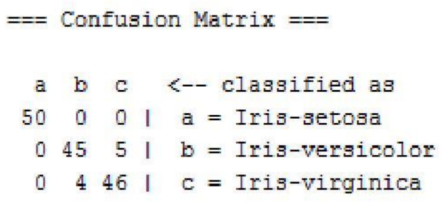
Test Option: 10-fold Cross Validation
minNumObj=3

Input: 150 instances

a. Without Missing Values:

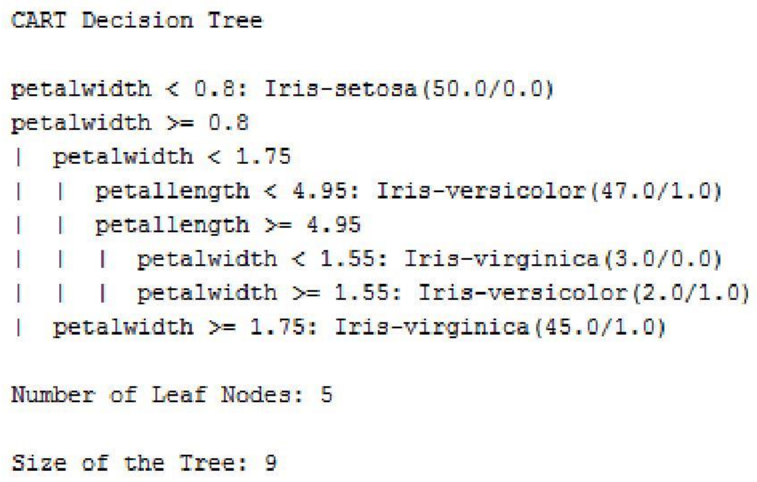


```
CART Decision Tree

petallength < 2.45: Iris-setosa(50.0/0.0)
petallength >= 2.45
|  petalwidth < 1.75
|  |  petallength < 4.95: Iris-versicolor(47.0/1.0)
|  |  petallength >= 4.95
|  |  |  petalwidth < 1.55: Iris-virginica(3.0/0.0)
|  |  |  petalwidth >= 1.55: Iris-versicolor(2.0/1.0)
|  petalwidth >= 1.75: Iris-virginica(45.0/1.0)

Number of Leaf Nodes: 5

Size of the Tree: 9
```

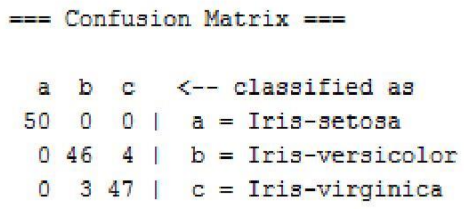


```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 45  5 |  b = Iris-versicolor
  0  4 46 |  c = Iris-virginica
```

b. To 10 % the instances, missing values were introduced to petallength attribute across all classes:

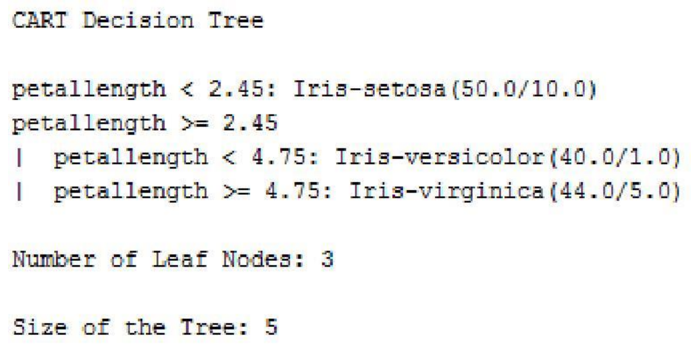


```
CART Decision Tree

petalwidth < 0.8: Iris-setosa(50.0/0.0)
petalwidth >= 0.8
|  petalwidth < 1.75
|  |  petallength < 4.95: Iris-versicolor(47.0/1.0)
|  |  petallength >= 4.95
|  |  |  petalwidth < 1.55: Iris-virginica(3.0/0.0)
|  |  |  petalwidth >= 1.55: Iris-versicolor(2.0/1.0)
|  petalwidth >= 1.75: Iris-virginica(45.0/1.0)

Number of Leaf Nodes: 5

Size of the Tree: 9
```

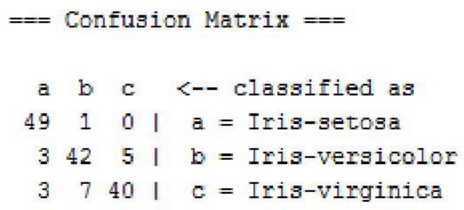


```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 46  4 |  b = Iris-versicolor
  0  3 47 |  c = Iris-virginica
```

c. In addition to b, 10% of instances, missing values introduced to petalwidth attribute across all classes:



```
CART Decision Tree

petallength < 2.45: Iris-setosa(50.0/10.0)
petallength >= 2.45
|  petallength < 4.75: Iris-versicolor(40.0/1.0)
|  petallength >= 4.75: Iris-virginica(44.0/5.0)

Number of Leaf Nodes: 3

Size of the Tree: 5
```



```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 49  1  0 |  a = Iris-setosa
  3 42  5 |  b = Iris-versicolor
  3  7 40 |  c = Iris-virginica
```

Analysis:

As more and more missing values were introduced to the dataset, the entropy value accordingly were impacted and the algorithm accordingly chose different attribute to classify the values better. Moreover, if there is any class with more missing values, then the error rate for that class was more. Also, as the missing values increased, the size of the tree decreased as it was not able to determine the attributes and conditions for the decision making step. More instances started to get pruned in the terminal nodes. However, the algorithm performed much better than the Decision Stump with a better accuracy.

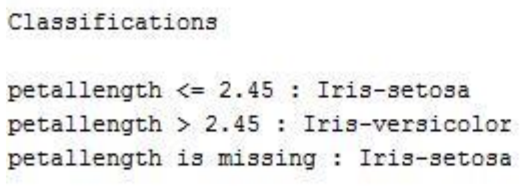
Impact of Noise

1. Decision Stump

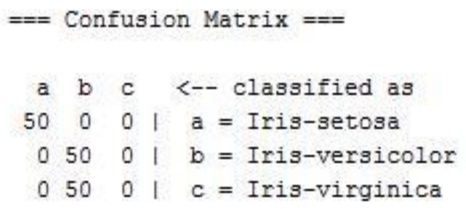
Test Option: 10-fold Cross Validation
useSERule=true

Input: 150 instances

a. Without Noise:

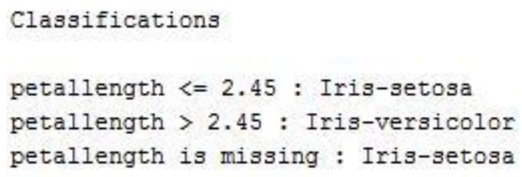


```
Classifications

petallength <= 2.45 : Iris-setosa
petallength > 2.45 : Iris-versicolor
petallength is missing : Iris-setosa
```

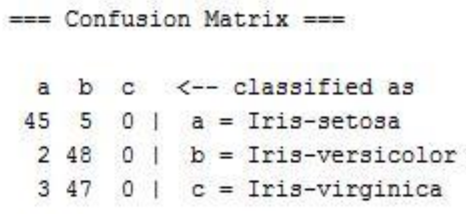


```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 50  0 |  b = Iris-versicolor
  0 50  0 |  c = Iris-virginica
```

b. To 10 % of the instances, noise introduced across all classes:

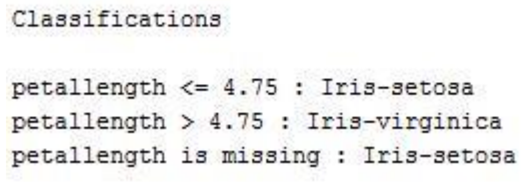


```
Classifications

petallength <= 2.45 : Iris-setosa
petallength > 2.45 : Iris-versicolor
petallength is missing : Iris-setosa
```

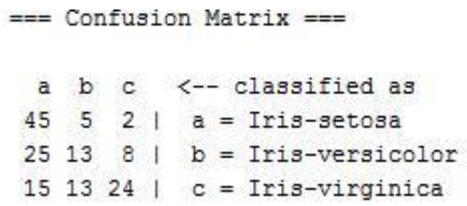


```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 45  5  0 |  a = Iris-setosa
  2 48  0 |  b = Iris-versicolor
  3 47  0 |  c = Iris-virginica
```

c. In addition to b, to 10 % of the instances, noise introduced across all classes:



```
Classifications

petallength <= 4.75 : Iris-setosa
petallength > 4.75 : Iris-virginica
petallength is missing : Iris-setosa
```



```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 45  5  2 |  a = Iris-setosa
 25 13  8 |  b = Iris-versicolor
 15 13 24 |  c = Iris-virginica
```

Analysis:

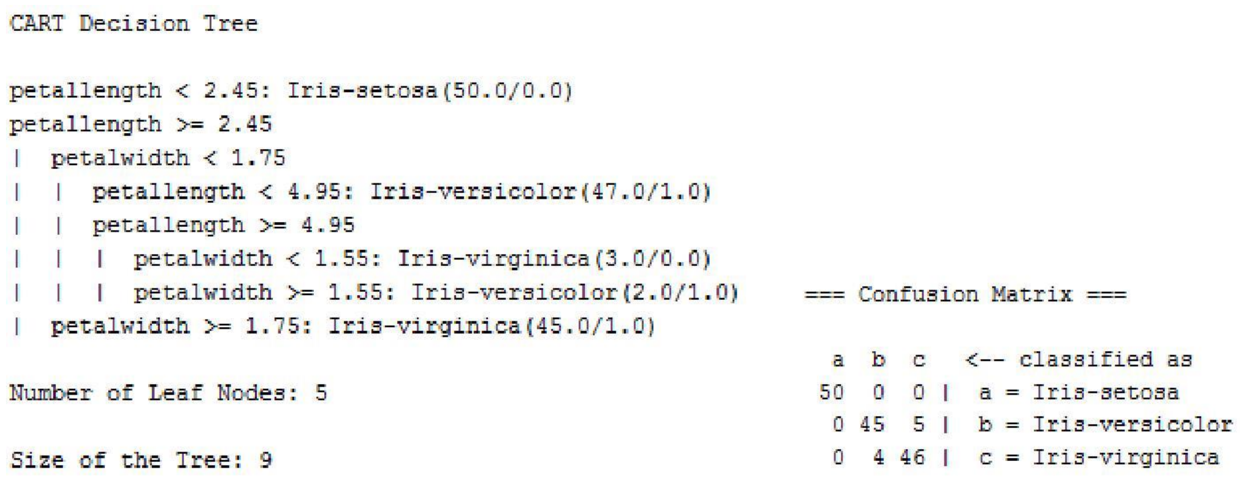
As more noise were introduced to the dataset, the classification accuracy was impacted badly but the decision stump decision tree did not change regardless of how much noise instances were introduced. Moreover, if there is any class with more noise values, then the error rate for that class was more. In addition to the algorithm's inability to classify the dataset into three different classes, these missing values still made this classifier unfit for further analysis. Since, the noise impacts the data quality of the training dataset, we use the classifier parameter useSERule=true in order to increase the tolerance range of the error values to get a better prediction and in turn it helps the classifier to classify the unseen test data better.

2. Simple CART

Test Option: 10-fold Cross Validation
minNumObj=3

Input: 150 instances

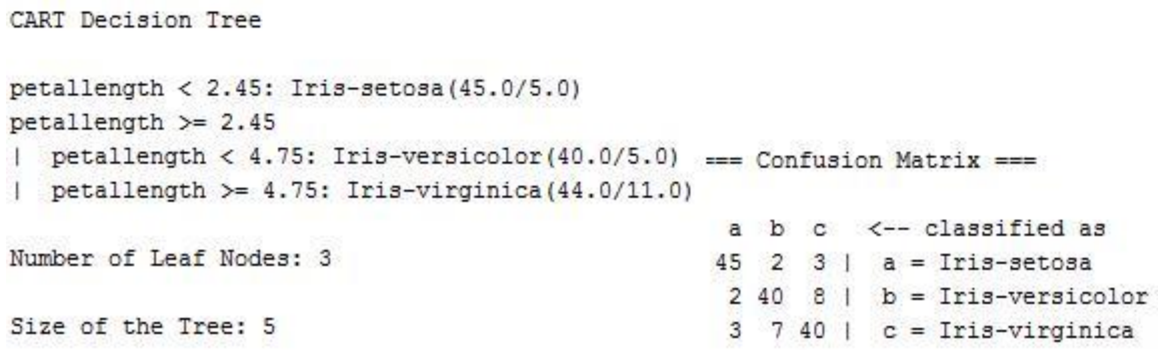
a. Without Noise:



```
CART Decision Tree

petallength < 2.45: Iris-setosa(50.0/0.0)
petallength >= 2.45
|  petalwidth < 1.75
|  |  petallength < 4.95: Iris-versicolor(47.0/1.0)
|  |  petallength >= 4.95
|  |  |  petalwidth < 1.55: Iris-virginica(3.0/0.0)
|  |  |  petalwidth >= 1.55: Iris-versicolor(2.0/1.0)
|  petalwidth >= 1.75: Iris-virginica(45.0/1.0)

Number of Leaf Nodes: 5

Size of the Tree: 9
```

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 45  5 |  b = Iris-versicolor
  0  4 46 |  c = Iris-virginica
```

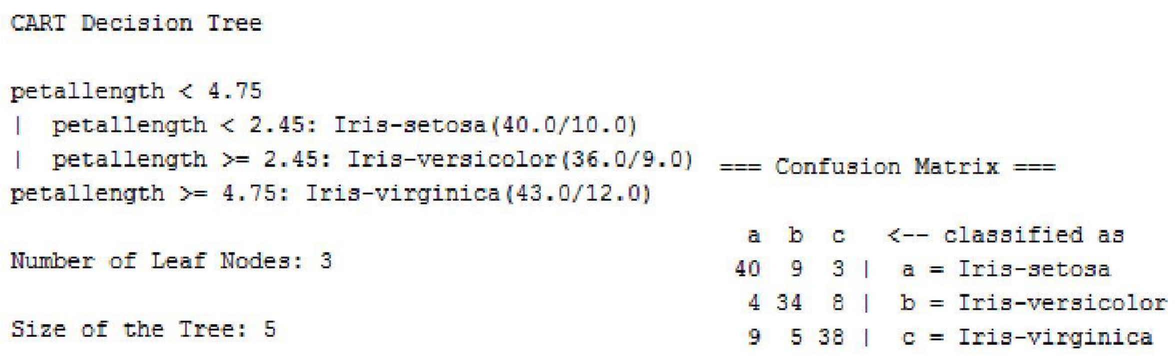
b. 10 % of Noise values introduced across all classes:



```
CART Decision Tree

petallength < 2.45: Iris-setosa(45.0/5.0)
petallength >= 2.45
|  petallength < 4.75: Iris-versicolor(40.0/5.0)
|  petallength >= 4.75: Iris-virginica(44.0/11.0)

Number of Leaf Nodes: 3

Size of the Tree: 5
```

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 45  2  3 |  a = Iris-setosa
  2 40  8 |  b = Iris-versicolor
  3  7 40 |  c = Iris-virginica
```

c. 10% of noise values introduced across all classes:



```
CART Decision Tree

petallength < 4.75
|  petallength < 2.45: Iris-setosa(40.0/10.0)
|  petallength >= 2.45: Iris-versicolor(36.0/9.0)
petallength >= 4.75: Iris-virginica(43.0/12.0)

Number of Leaf Nodes: 3

Size of the Tree: 5
```

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 40  9  3 |  a = Iris-setosa
  4 34  8 |  b = Iris-versicolor
  9  5 38 |  c = Iris-virginica
```

Analysis:

As more and noise values were introduced to the dataset, the condition on which the classifier was built, i.e. the decision tree got revised in order to classify the dataset better. Moreover, if there is any class with more noise values, then the misclassification for that class was more. However, the algorithm performed much better than the Decision Stump with a better accuracy.

Conclusion

As per the performance analysis results, Simple CART algorithm classifies the Iris dataset with much better accuracy as Decision Stump algorithm fails miserably in classifying a dataset with numerical attributes and three classes, as it cannot classify a third class. Since, using the usePrune=true and useOneSE=true parameters, the Simple CART model can handle noise and missing values with a decent classification accuracy as the class distributions are of equal in proportion for the training set.

V. ANALYSIS OF VOTE DATASET

Test Options

On Training Set

The classifiers built the model based on the input dataset for training. Since we evaluate the model with the same training set, we expect the accuracy to be high. Though, the results do not represent the accuracy of the classifier for an unseen independent dataset, we do this analysis to observe the classifier's behavior on the trained model.

Supplied Test Set

The dataset was manually divided into two sets, training and test sets, in the ratio 2:1 of original dataset so that there are no common instances between the two sets. The classifiers were built using this training set and then the test set whose instances are unseen for the classifiers was supplied and the performance of the classifier is observed. Also, the class distributions of the classes democrat and republican in the test set is maintained in 73:71 ratio. Number of instances in training and test are 287 and 148 respectively.

Percentage Split

The vote dataset is split into 2:1 ratio randomly based on this option as the percentage split that we have used here is 66. The random split could take any instances in training and test and thus there is a possibility for a skewed class distribution in the training or test sets that could not lead to a fair evaluation of the classifier depending on the random seed value.

10-fold Cross Validation

The details mentioned in Iris dataset analysis applies here as well.

Analysis of 10-fold Cross Validation

Below is an instance of subsets derived from the original data set of 435 instances. It can be easily seen that datasets are divided into 10 almost equal size of subsets preserving a fair class distribution in each of the folds as in the original set.

Filter = StratifiedRemoveFolds	Instances	Class Distribution	
Fold		democrat	republican
1	44	27	17
2	44	26	18
3	44	27	17
4	44	27	17
5	44	27	17
6	43	28	15
7	43	26	17
8	43	26	17
9	43	27	16
10	43	27	16

Performance Analysis of the Classifiers

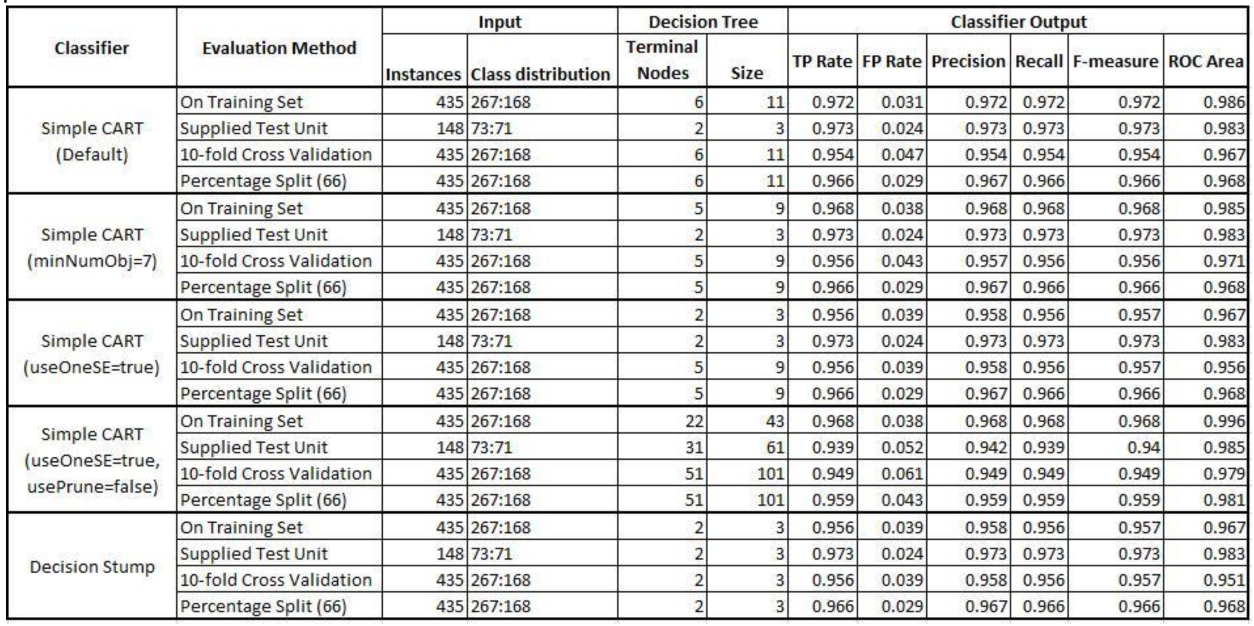
a. Decision Stump

This dataset has only two classes and thus the decision stump has performed well for this dataset unlike the Iris dataset. Also, as all the attributes are of nominal type, the classifier is able to easily classify the records into two classes based on the decision tree. As discussed earlier, the attribute with maximum information gain is chosen and the decision tree classifies using a simple condition. Here, we can compare the results of the Decision Stump algorithm with that of the Simple CART. Unlike the Iris dataset performance, this algorithm performs as good as the simple cart algorithm with a much simpler classification model of smaller tree and leaf nodes. Moreover, considering the supplied test set cases of Simple cart and this classifier, decision stump is able to achieve the accuracy of more complex simple cart model.

b. Simple CART

In the evaluation results given in the below table, we can clearly see the impact of usePrune=false on the classifier model. The model with tree size 11 performs the same way as that of the model with the usePrune=false in which tree size exploded to 101. Considering the missing values presence in the given vote dataset, the classifier has performed well with a greater area under ROC curve for supplied test unit case. This shows the performance of the classifier for an unseen test set. In order to get a better performance, it is good to consider the usePrune and minNumObj as the default values. It was also observed that increasing the minNumObj, increased the size of the tree initially with accuracy and then decreased gradually with both size and accuracy.

Sample output of each of the test experiment for each classifier with respect to the respective classifier parameters are mentioned in the below table:



Classifier	Evaluation Method	Input		Decision Tree		Classifier Output					
		Instances	Class distribution	Terminal Nodes	Size	TP Rate	FP Rate	Precision	Recall	F-measure	ROC Area
Simple CART (Default)	On Training Set	435	267:168	6	11	0.972	0.031	0.972	0.972	0.972	0.986
	Supplied Test Unit	148	73:71	2	3	0.973	0.024	0.973	0.973	0.973	0.983
	10-fold Cross Validation	435	267:168	6	11	0.954	0.047	0.954	0.954	0.954	0.967
	Percentage Split (66)	435	267:168	6	11	0.966	0.029	0.967	0.966	0.966	0.968
Simple CART (minNumObj=7)	On Training Set	435	267:168	5	9	0.968	0.038	0.968	0.968	0.968	0.985
	Supplied Test Unit	148	73:71	2	3	0.973	0.024	0.973	0.973	0.973	0.983
	10-fold Cross Validation	435	267:168	5	9	0.956	0.043	0.957	0.956	0.956	0.971
	Percentage Split (66)	435	267:168	5	9	0.966	0.029	0.967	0.966	0.966	0.968
Simple CART (useOneSE=true)	On Training Set	435	267:168	2	3	0.956	0.039	0.958	0.956	0.957	0.967
	Supplied Test Unit	148	73:71	2	3	0.973	0.024	0.973	0.973	0.973	0.983
	10-fold Cross Validation	435	267:168	5	9	0.956	0.039	0.958	0.956	0.957	0.956
	Percentage Split (66)	435	267:168	5	9	0.966	0.029	0.967	0.966	0.966	0.968
Simple CART (useOneSE=true, usePrune=false)	On Training Set	435	267:168	22	43	0.968	0.038	0.968	0.968	0.968	0.996
	Supplied Test Unit	148	73:71	31	61	0.939	0.052	0.942	0.939	0.94	0.985
	10-fold Cross Validation	435	267:168	51	101	0.949	0.061	0.949	0.949	0.949	0.979
	Percentage Split (66)	435	267:168	51	101	0.959	0.043	0.959	0.959	0.959	0.981
Decision Stump	On Training Set	435	267:168	2	3	0.956	0.039	0.958	0.956	0.957	0.967
	Supplied Test Unit	148	73:71	2	3	0.973	0.024	0.973	0.973	0.973	0.983
	10-fold Cross Validation	435	267:168	2	3	0.956	0.039	0.958	0.956	0.957	0.951
	Percentage Split (66)	435	267:168	2	3	0.966	0.029	0.967	0.966	0.966	0.968

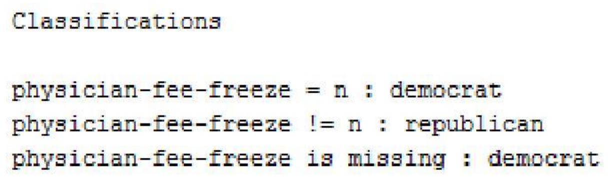
Impact of missing values

The given Vote dataset already had missing values and we could see the impact of the missing values in the classifier output. We consider 10-fold cross validation option as it gives a generalized value unlike choosing one random test set or percentage split.

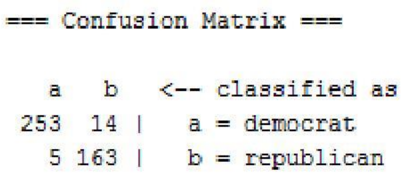
1. Decision Stump

Test Option: 10-fold Cross Validation Input: 435 instances

a. With default dataset

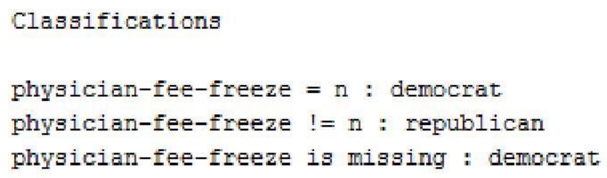


```
Classifications

physician-fee-freeze = n : democrat
physician-fee-freeze != n : republican
physician-fee-freeze is missing : democrat
```

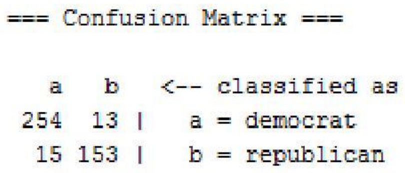


```
=== Confusion Matrix ===

   a   b   <-- classified as
 253  14 |   a = democrat
   5 163 |   b = republican
```

b. To 10 % of instances, missing values were introduced to physician-fee-freeze attribute across both the classes:

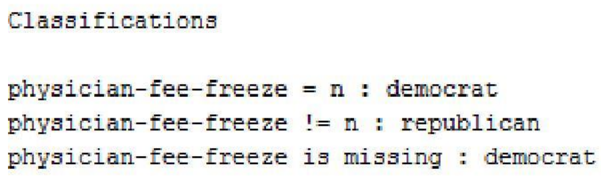


```
Classifications

physician-fee-freeze = n : democrat
physician-fee-freeze != n : republican
physician-fee-freeze is missing : democrat
```

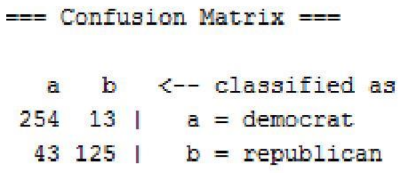


```
=== Confusion Matrix ===

   a   b   <-- classified as
 254  13 |   a = democrat
  15 153 |   b = republican
```

c. To 10 % of instances, missing values were introduced to physician-fee-freeze attribute across both the classes:



```
Classifications

physician-fee-freeze = n : democrat
physician-fee-freeze != n : republican
physician-fee-freeze is missing : democrat
```



```
=== Confusion Matrix ===

   a   b   <-- classified as
 254  13 |   a = democrat
  43 125 |   b = republican
```

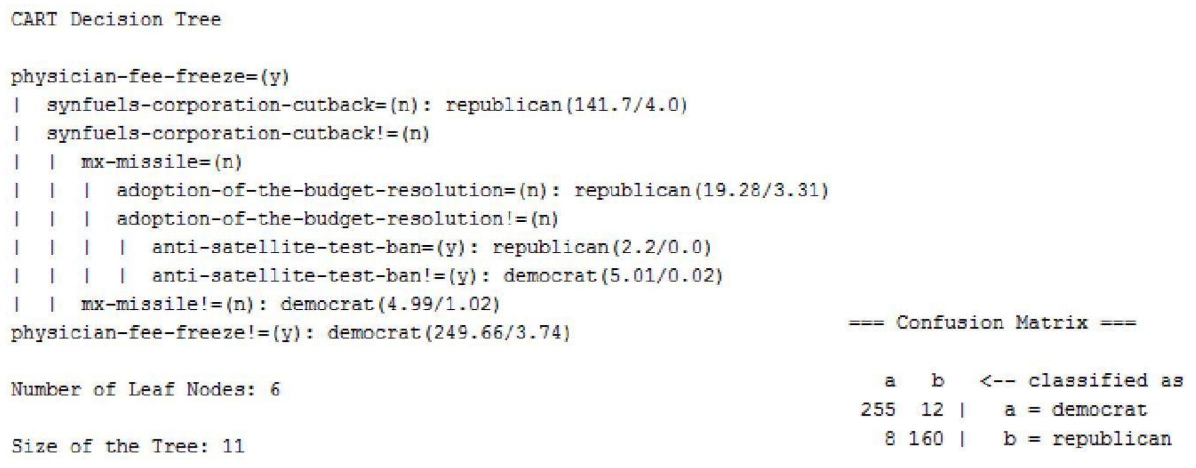
Analysis:

We could also observe that as the missing values were introduced to the attribute in the decision tree, the attribute was still in the decision tree in the next iterations. This dramatically impacted the performance of the Decision Stump as the classifications were done based on this attribute with more missing values. Though, the plus side is that even if other attributes miss many values, the decision stump decision will be unaffected. Also, as the missing values percentage was more for export-administration-act-south-africa attribute was more, the difference between and after the decision split entropies was less implying less information gain and thus that attribute was least preferred among other attributes.

2. Simple CART

Test Option: 10-fold Cross Validation Input: 435 instances

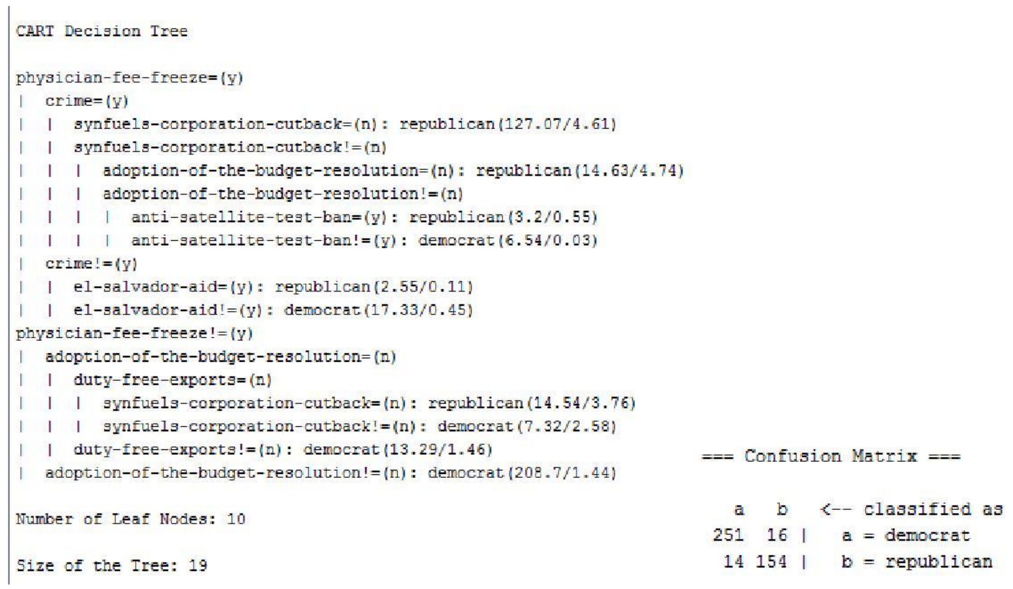
a. With default dataset:



```
CART Decision Tree

physician-fee-freeze=(y)
|  synfuels-corporation-cutback=(n): republican(141.7/4.0)
|  synfuels-corporation-cutback!=(n)
|  |  mx-missile=(n)
|  |  |  adoption-of-the-budget-resolution=(n): republican(19.28/3.31)
|  |  |  adoption-of-the-budget-resolution!=(n)
|  |  |  |  anti-satellite-test-ban=(y): republican(2.2/0.0)
|  |  |  |  anti-satellite-test-ban!=(y): democrat(5.01/0.02)
|  |  mx-missile!=(n): democrat(4.99/1.02)
physician-fee-freeze!=(y): democrat(249.66/3.74)

Number of Leaf Nodes: 6

Size of the Tree: 11
```

```
=== Confusion Matrix ===

   a   b   <-- classified as
 255  12 |   a = democrat
   8 160 |   b = republican
```

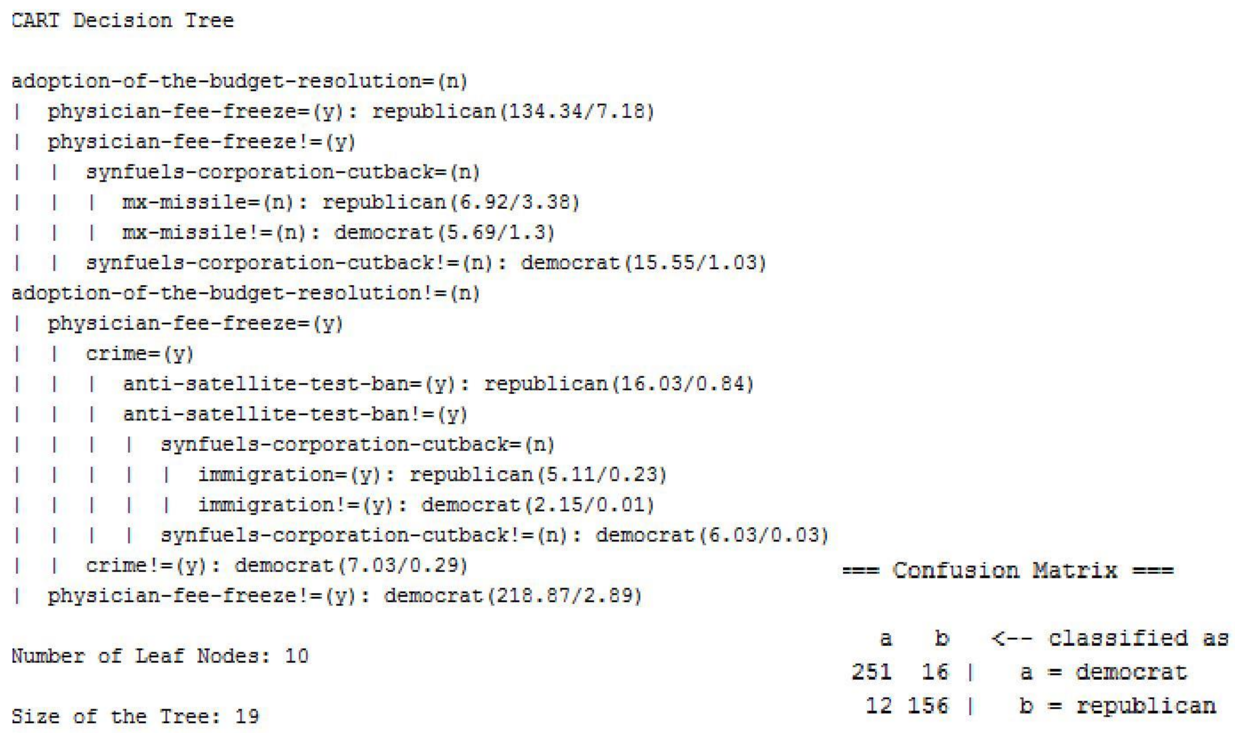
b. To 10 % of instances, missing values were introduced to physician-fee-freeze attribute across both the classes:



```
CART Decision Tree

physician-fee-freeze=(y)
|  crime=(y)
|  |  synfuels-corporation-cutback=(n): republican(127.07/4.61)
|  |  synfuels-corporation-cutback!=(n)
|  |  |  adoption-of-the-budget-resolution=(n): republican(14.63/4.74)
|  |  |  adoption-of-the-budget-resolution!=(n)
|  |  |  |  anti-satellite-test-ban=(y): republican(3.2/0.55)
|  |  |  |  anti-satellite-test-ban!=(y): democrat(6.54/0.03)
|  crime!=(y)
|  |  el-salvador-aid=(y): republican(2.55/0.11)
|  |  el-salvador-aid!=(y): democrat(17.33/0.45)
physician-fee-freeze!=(y)
|  adoption-of-the-budget-resolution=(n)
|  |  duty-free-exports=(n)
|  |  |  synfuels-corporation-cutback=(n): republican(14.54/3.76)
|  |  |  synfuels-corporation-cutback!=(n): democrat(7.32/2.58)
|  |  duty-free-exports!=(n): democrat(13.29/1.46)
|  adoption-of-the-budget-resolution!=(n): democrat(208.7/1.44)

Number of Leaf Nodes: 10

Size of the Tree: 19
```

```
=== Confusion Matrix ===

   a   b   <-- classified as
 251  16 |   a = democrat
  14 154 |   b = republican
```

c. To 10 % of instances, missing values were introduced to physician-fee-freeze attribute across both the classes:



```
CART Decision Tree

adoption-of-the-budget-resolution=(n)
|  physician-fee-freeze=(y): republican(134.34/7.18)
|  physician-fee-freeze!=(y)
|  |  synfuels-corporation-cutback=(n)
|  |  |  mx-missile=(n): republican(6.92/3.38)
|  |  |  mx-missile!=(n): democrat(5.69/1.3)
|  |  synfuels-corporation-cutback!=(n): democrat(15.55/1.03)
adoption-of-the-budget-resolution!=(n)
|  physician-fee-freeze=(y)
|  |  crime=(y)
|  |  |  anti-satellite-test-ban=(y): republican(16.03/0.84)
|  |  |  anti-satellite-test-ban!=(y)
|  |  |  |  synfuels-corporation-cutback=(n)
|  |  |  |  |  immigration=(y): republican(5.11/0.23)
|  |  |  |  |  immigration!=(y): democrat(2.15/0.01)
|  |  |  |  synfuels-corporation-cutback!=(n): democrat(6.03/0.03)
|  |  crime!=(y): democrat(7.03/0.29)
|  physician-fee-freeze!=(y): democrat(218.87/2.89)

Number of Leaf Nodes: 10

Size of the Tree: 19
```

```
=== Confusion Matrix ===

   a   b   <-- classified as
 251  16 |   a = democrat
  12 156 |   b = republican
```

Analysis:

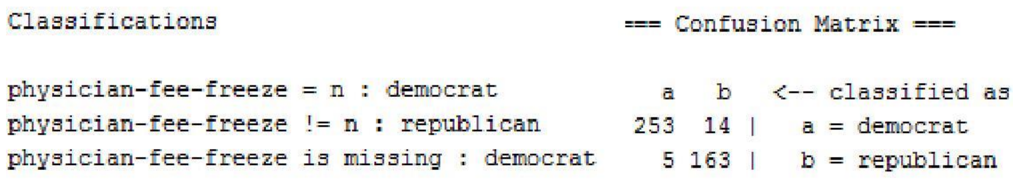
Here unlike the decision stump, we could see that the accuracy of the classifier is better. This is because, the decision tree is getting updated to another better attribute when the missing values are more for an attribute in order to maintain the accuracy. Though the complexity increases as more missing values are introduced, the overall performance is relatively more robust as the decision tree involves other attributes as well for the classification criteria.

Impact of Noise

1. Decision Stump

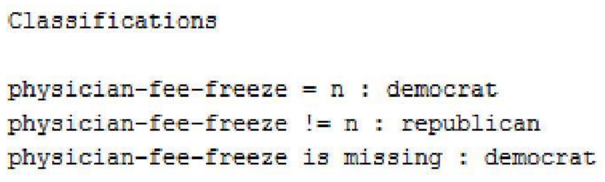
Test Option: 10-fold Cross Validation Input: 150 instances

a. Without Noise:

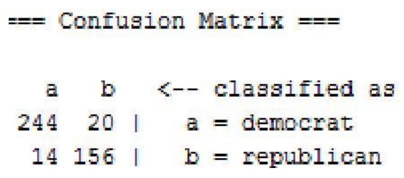


```
Classifications                                === Confusion Matrix ===

physician-fee-freeze = n : democrat              a   b   <-- classified as
physician-fee-freeze != n : republican         253  14 |   a = democrat
physician-fee-freeze is missing : democrat       5 163 |   b = republican
```

b. 5 % of Noise introduced across all classes:

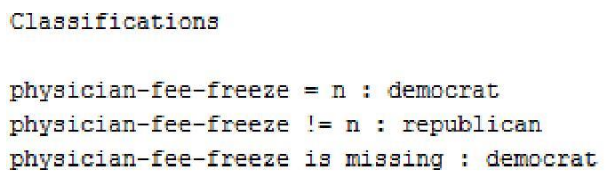


```
Classifications

physician-fee-freeze = n : democrat
physician-fee-freeze != n : republican
physician-fee-freeze is missing : democrat
```

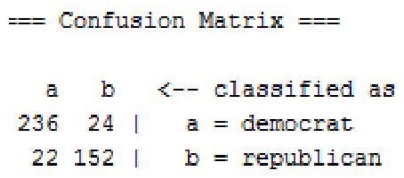


```
=== Confusion Matrix ===

   a   b   <-- classified as
 244  20 |   a = democrat
  14 156 |   b = republican
```

c. 5 % of Noise introduced across all classes:



```
Classifications

physician-fee-freeze = n : democrat
physician-fee-freeze != n : republican
physician-fee-freeze is missing : democrat
```



```
=== Confusion Matrix ===

   a   b   <-- classified as
 236  24 |   a = democrat
  22 152 |   b = republican
```

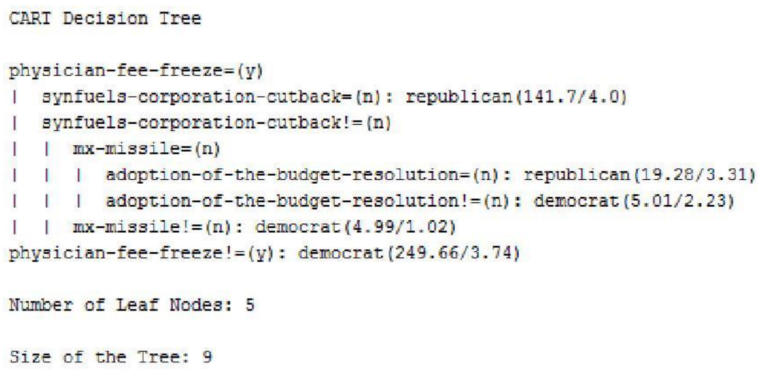
Analysis:

As more noise were introduced to the dataset, the classification accuracy was impacted badly but the decision stump decision tree did not change regardless of how much noise instances were introduced. Moreover, if there is any class with more noise values, then the misclassification error for that class was more. There are no option to tune this algorithm with parameters and so it is more likely that the algorithm will be more prone to errors on noise induction.

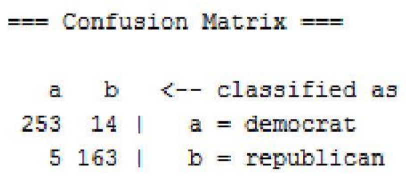
2. Simple CART

Test Option: 10-fold Cross Validation Input: 150 instances
useOneSE=true

a. Without Noise:

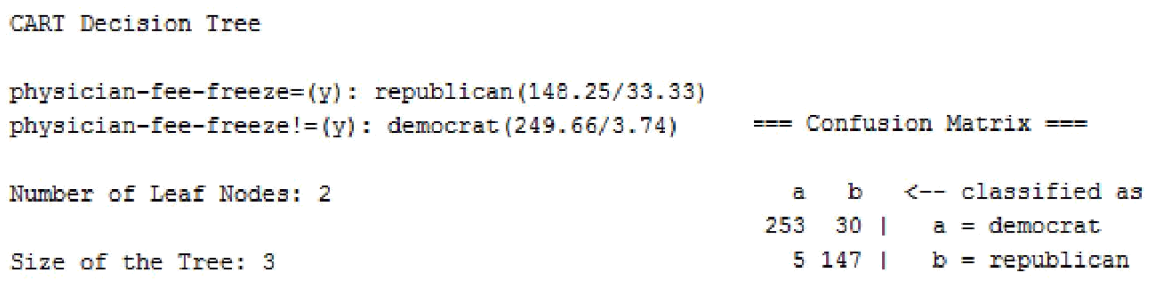


```
CART Decision Tree

physician-fee-freeze=(y)
|  synfuels-corporation-cutback=(n): republican(141.7/4.0)
|  synfuels-corporation-cutback!=(n)
|  |  mx-missile=(n)
|  |  |  adoption-of-the-budget-resolution=(n): republican(19.28/3.31)
|  |  |  adoption-of-the-budget-resolution!=(n): democrat(5.01/2.23)
|  |  mx-missile!=(n): democrat(4.99/1.02)
physician-fee-freeze!=(y): democrat(249.66/3.74)

Number of Leaf Nodes: 5

Size of the Tree: 9
```



```
=== Confusion Matrix ===

   a   b   <-- classified as
 253  14 |   a = democrat
   5 163 |   b = republican
```

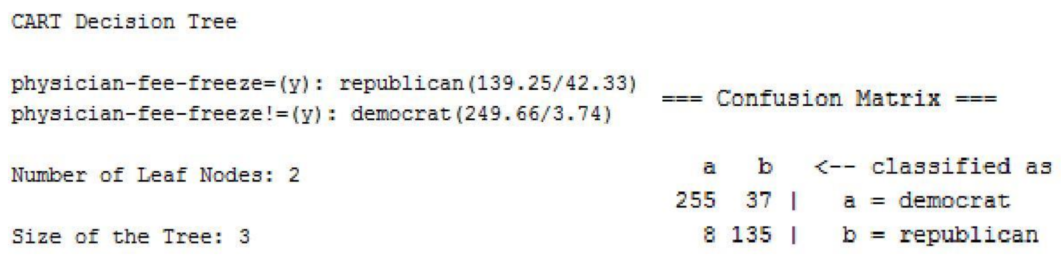
b. To 5 % of instances, Noise values were introduced on republican, i.e. democrats were classified as republicans. (Skewed noise):



```
CART Decision Tree

physician-fee-freeze=(y): republican(148.25/33.33)
physician-fee-freeze!=(y): democrat(249.66/3.74)

Number of Leaf Nodes: 2

Size of the Tree: 3
```

```
=== Confusion Matrix ===

   a   b   <-- classified as
 253  30 |   a = democrat
   5 147 |   b = republican
```

c. To 5 % of the instances, noise values were introduced for democrat class:



```
CART Decision Tree

physician-fee-freeze=(y): republican(139.25/42.33)
physician-fee-freeze!=(y): democrat(249.66/3.74)

Number of Leaf Nodes: 2

Size of the Tree: 3
```

```
=== Confusion Matrix ===

   a   b   <-- classified as
 255  37 |   a = democrat
   8 135 |   b = republican
```

Analysis:

We could see that the noise values reduced the tree size in the case observed here. Moreover, the class which was classified wrongly, was only getting impacted as we proceeded with the experiments. Between the first and second run shown here, skewed noises were introduced and only the republican class was impacted. When more noise were added to the democrats class, the impact was less in the output. The second and third run of the runs shown above in which more classes were wrongly classified as democrats but the output did not reflect much impact on the classification of the democrat as observe in the republican class. This is because, the class distribution shows that the overall number of instances for the 435 instances have democrats than the republican. Since republican class had lesser instances and more noise were introduced, the classifier was affected. However, when noises were added to democrats, the impact was less as it has more training instances in the class. This shows the impact of class distribution of the training set on the classifier vividly.

Conclusion

The Decision Stump algorithm performed equally well with respect to the Simple CART algorithm with a much simpler decision tree for the Vote dataset. If we expect a dataset with lot of missing values, we can opt for Simple CART as it uses fractional instances method unlike Decision Stump. If the main split includes an attribute with missing values, it could heavily impmact the classifier output when decision stump is used. Noise in the dataset can be handled in a much better manner using useOneSE=true parameter for Simple CART algorithm. Also, Decision Stump performs extremely well when the dataset has all the attributes of nominal type especially of binary type with only a binary decision to make.

VI. ANALYSIS OF LABOR DATASET

Test Options

On Training Set

These experiments were done on the training set to observe the behavior and not for evaluating the classifier performance.

Supplied Test Set

The dataset was manually divided into two sets, training and test sets, in the ratio 2:1 of original dataset so that there are no common instances between the two sets. The classifiers were built using this training set and then the test set whose instances are unseen for the classifiers was supplied and the performance of the classifier is observed. Also, the class distributions of the classes bad and good in the test set is maintained in 3:16 ratio. Number of instances in training and test are 38 and 19 respectively.

Percentage Split

The vote dataset is split into 2:1 ratio randomly based on this option as the percentage split that we have used here is 66. The random split could take any instances in training and test and thus there is a possibility for a skewed class distribution in the training or test sets that could not lead to a fair evaluation of the classifier depending on the random seed value.

10-fold Cross Validation

The details mentioned in Iris dataset analysis applies here as well.

Analysis of 10-fold Cross Validation

Below is an instance of subsets derived from the original data set of 57 instances. It can be easily seen that datasets are divided into 10 almost equal size of subsets preserving a fair class distribution in each of the folds as in the original dataset.

Filter = StratifiedRemoveFolds	Instances	Class Distribution	
Fold		bad	good
1	6	2	4
2	6	2	4
3	6	2	4
4	6	2	4
5	6	2	4
6	6	2	4
7	6	2	4
8	5	2	3
9	5	2	3
10	5	2	3

Performance Analysis of the Classifiers

a. Decision Stump

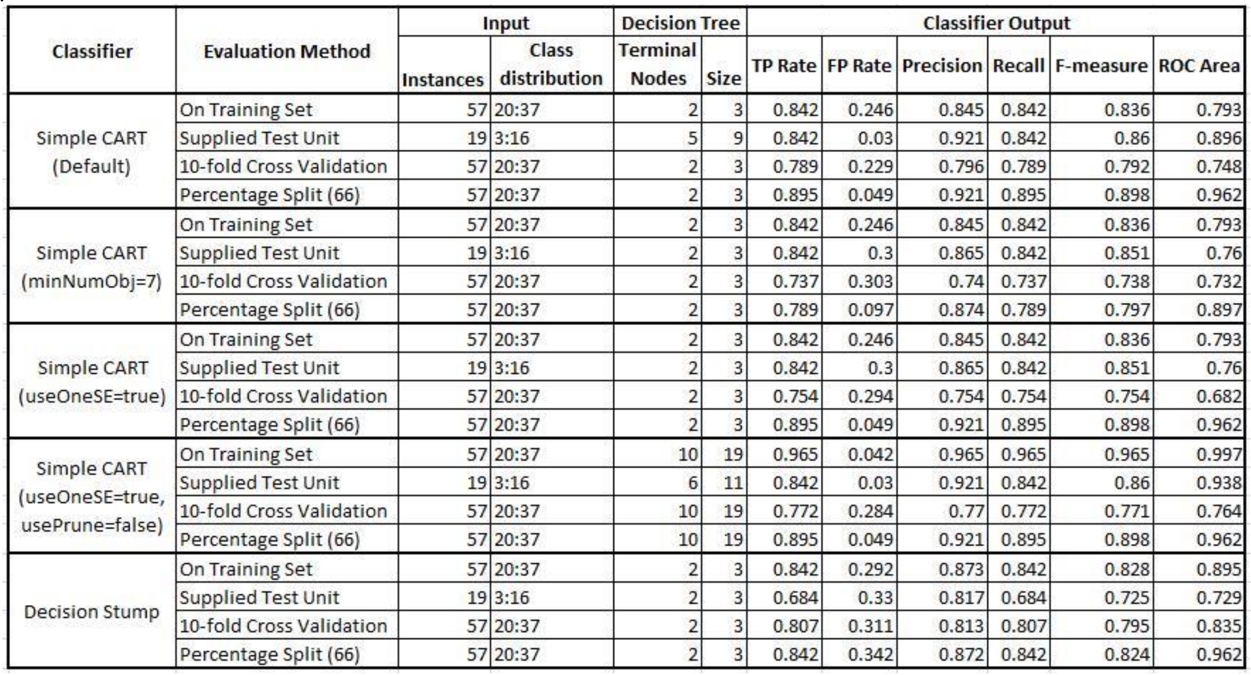
This dataset also has only two classes and thus the decision stump has performed well for this dataset unlike the Iris dataset. Also, it has both types of attributes - nominal as well as numeric. Another observation with respect to the dataset is that this has plenty of missing values than previous dataset we analyzed. When we can compare the results of the Decision Stump algorithm with that of the Simple CART, considering the supplied test set cases, decision stump is not performing as good as simple CART algorithm. This could be due to lot of missing values that the attribute chosen for the decision tree doesn't represent the best criteria for the classification.

b. Simple CART

It is observed that this algorithm performs better than decision stump overall. With a simpler model of decision tree, it is able to achieve more accuracy when compared to the decision stump's output. The tree size of 3, the simple cart is having a better performance. Also, consider the supplied test set cases of both the algorithms. The simple cart is able to predict for more number of instances, the correct classes unlike the decision stump for unseen test set.

We can also observe the impact of useOneSE = true cases. Since this dataset has numerous missing values for the attributes, the accuracy improves as we enable the Standard Error rule for the classifier. This can be compared for percentage split method. By setting useOneSE improves the performance of the classifier. The performance of the classifier doesn't improve much by changing other parameters due to the nature of the dataset- smaller dataset with missing values.

Sample output of each of the test experiment for each classifier with respect to the respective classifier parameters are mentioned in the below table:



Classifier	Evaluation Method	Input		Decision Tree		Classifier Output					
		Instances	Class distribution	Terminal Nodes	Size	TP Rate	FP Rate	Precision	Recall	F-measure	ROC Area
Simple CART (Default)	On Training Set	57	20:37	2	3	0.842	0.246	0.845	0.842	0.836	0.793
	Supplied Test Unit	19	3:16	5	9	0.842	0.03	0.921	0.842	0.86	0.896
	10-fold Cross Validation	57	20:37	2	3	0.789	0.229	0.796	0.789	0.792	0.748
	Percentage Split (66)	57	20:37	2	3	0.895	0.049	0.921	0.895	0.898	0.962
Simple CART (minNumObj=7)	On Training Set	57	20:37	2	3	0.842	0.246	0.845	0.842	0.836	0.793
	Supplied Test Unit	19	3:16	2	3	0.842	0.3	0.865	0.842	0.851	0.76
	10-fold Cross Validation	57	20:37	2	3	0.737	0.303	0.74	0.737	0.738	0.732
	Percentage Split (66)	57	20:37	2	3	0.789	0.097	0.874	0.789	0.797	0.897
Simple CART (useOneSE=true)	On Training Set	57	20:37	2	3	0.842	0.246	0.845	0.842	0.836	0.793
	Supplied Test Unit	19	3:16	2	3	0.842	0.3	0.865	0.842	0.851	0.76
	10-fold Cross Validation	57	20:37	2	3	0.754	0.294	0.754	0.754	0.754	0.682
	Percentage Split (66)	57	20:37	2	3	0.895	0.049	0.921	0.895	0.898	0.962
Simple CART (useOneSE=true, usePrune=false)	On Training Set	57	20:37	10	19	0.965	0.042	0.965	0.965	0.965	0.997
	Supplied Test Unit	19	3:16	6	11	0.842	0.03	0.921	0.842	0.86	0.938
	10-fold Cross Validation	57	20:37	10	19	0.772	0.284	0.77	0.772	0.771	0.764
	Percentage Split (66)	57	20:37	10	19	0.895	0.049	0.921	0.895	0.898	0.962
Decision Stump	On Training Set	57	20:37	2	3	0.842	0.292	0.873	0.842	0.828	0.895
	Supplied Test Unit	19	3:16	2	3	0.684	0.33	0.817	0.684	0.725	0.729
	10-fold Cross Validation	57	20:37	2	3	0.807	0.311	0.813	0.807	0.795	0.835
	Percentage Split (66)	57	20:37	2	3	0.842	0.342	0.872	0.842	0.824	0.962

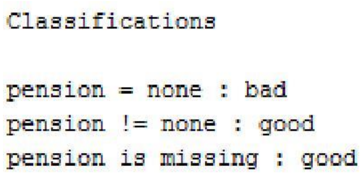
Impact of missing values

The given Labor dataset is small with lots of missing values and we could see that the performance of the classifiers are very much impacted overall based on the classifier outputs. We consider 10-fold cross validation option as it gives a generalized value unlike choosing one random test set or percentage split.

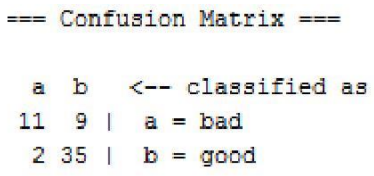
1. Decision Stump

Test Option: 10-fold Cross Validation Input: 57 instances

a. With default dataset:

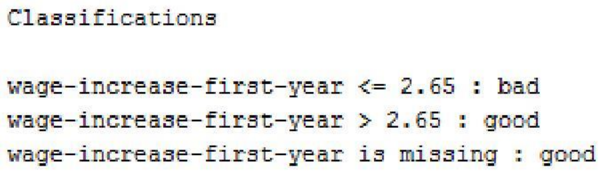


```
Classifications

pension = none : bad
pension != none : good
pension is missing : good
```

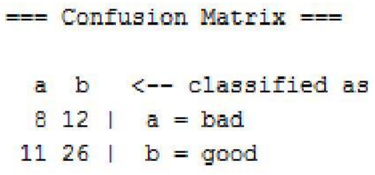


```
=== Confusion Matrix ===

  a  b   <-- classified as
 11  9 |  a = bad
  2 35 |  b = good
```

b. Missing values introduced to pension attribute for 10% of the instances



```
Classifications

wage-increase-first-year <= 2.65 : bad
wage-increase-first-year > 2.65 : good
wage-increase-first-year is missing : good
```



```
=== Confusion Matrix ===

  a  b   <-- classified as
  8 12 |  a = bad
 11 26 |  b = good
```

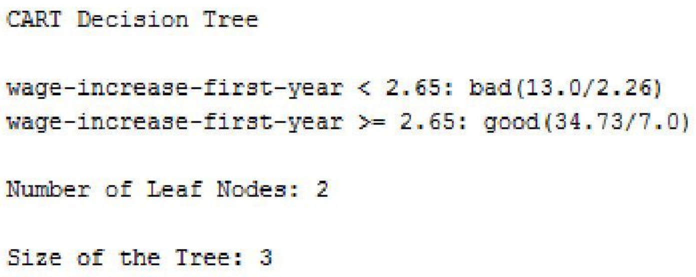
Analysis:

We can observe that the missing values are taking a toll on the classifier's performance. As more missing values were introduced to pension attribute, the decision tree chooses another attribute and the performance also gets worse.

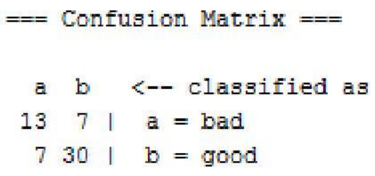
2. Simple CART

Test Option: 10-fold Cross Validation Input: 57 instances
useOneSE=true

a. With default dataset:

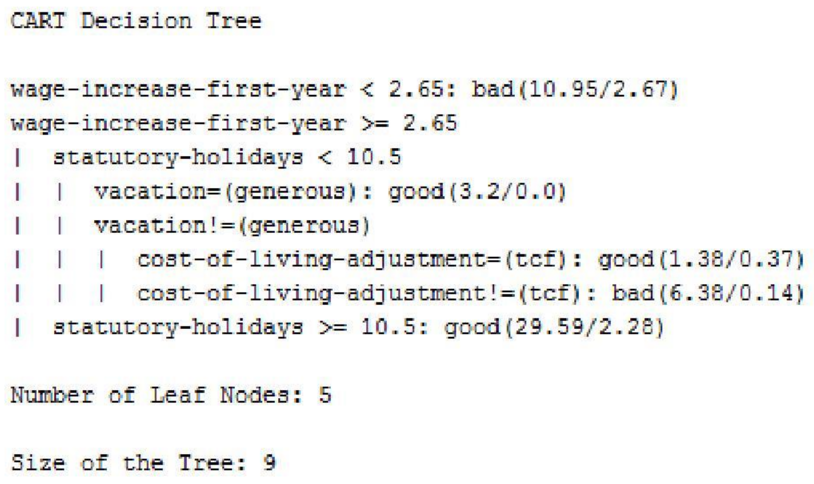


```
CART Decision Tree

wage-increase-first-year < 2.65: bad(13.0/2.26)
wage-increase-first-year >= 2.65: good(34.73/7.0)

Number of Leaf Nodes: 2

Size of the Tree: 3
```

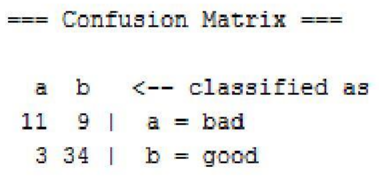


```
=== Confusion Matrix ===

  a  b   <-- classified as
 13  7 |  a = bad
  7 30 |  b = good
```

b. To 20 % of instances, missing values were introduced to wage-increase-first-year for both classes equally:



```
CART Decision Tree

wage-increase-first-year < 2.65: bad(10.95/2.67)
wage-increase-first-year >= 2.65
|  statutory-holidays < 10.5
|  |  vacation=(generous): good(3.2/0.0)
|  |  vacation!=(generous)
|  |  |  cost-of-living-adjustment=(tcf): good(1.38/0.37)
|  |  |  cost-of-living-adjustment!=(tcf): bad(6.38/0.14)
|  statutory-holidays >= 10.5: good(29.59/2.28)

Number of Leaf Nodes: 5

Size of the Tree: 9
```



```
=== Confusion Matrix ===

  a  b   <-- classified as
 11  9 |  a = bad
  3 34 |  b = good
```

Analysis:

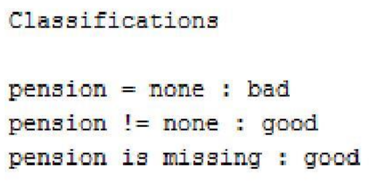
We can clearly see that the simple cart not only changes the decision tree to a more complex one, but also the performance is degraded.

Impact of Noise

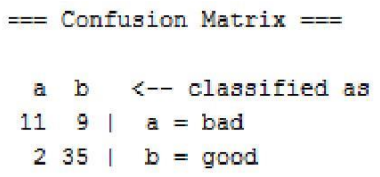
1. Decision Stump

Test Option: 10-fold Cross Validation Input: 57 instances
useOneSe=true

a. Without Noise:

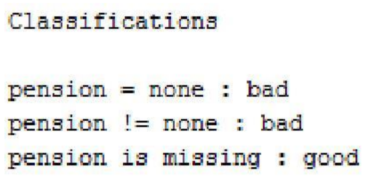


```
Classifications

pension = none : bad
pension != none : good
pension is missing : good
```

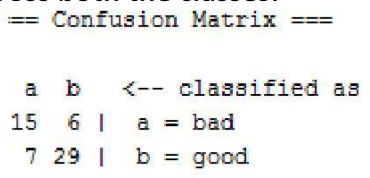


```
=== Confusion Matrix ===

  a  b   <-- classified as
 11  9 |  a = bad
  2 35 |  b = good
```

b. 5 % of Noise introduced across both the classes:

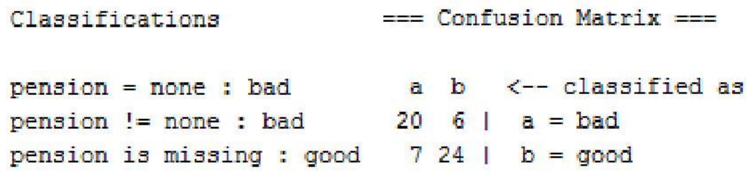


```
Classifications

pension = none : bad
pension != none : bad
pension is missing : good
```



```
== Confusion Matrix ===

  a  b   <-- classified as
 15  6 |  a = bad
  7 29 |  b = good
```

c. In addition 5 % of Noise introduced to class bad (misclassify good as bad):



```
Classifications            === Confusion Matrix ===

pension = none : bad         a  b   <-- classified as
pension != none : bad       20  6 |  a = bad
pension is missing : good    7 24 |  b = good
```

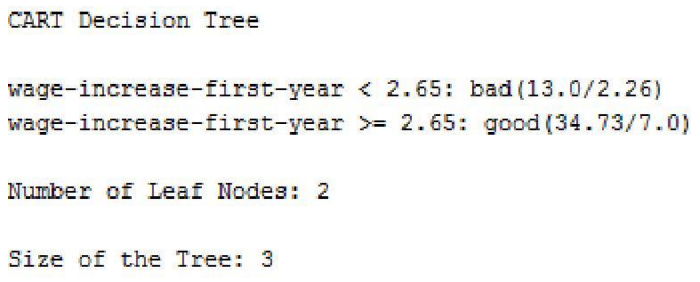
Analysis:

As more noise were introduced to the dataset, the classification accuracy was impacted but only for the class good. Surprisingly, instances with the class bad were getting classified correctly as the skewed noise was increased for that. This is because, the decision tree has got altered in such a way that only the missing pension cases will be classified as good which is absurd in this case.

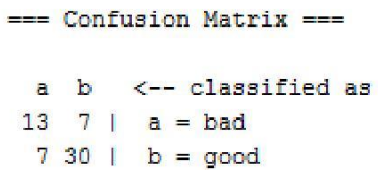
2. Simple CART

Test Option: 10-fold Cross Validation   Input: 57 instances
useOneSE=true

a. Without Noise:

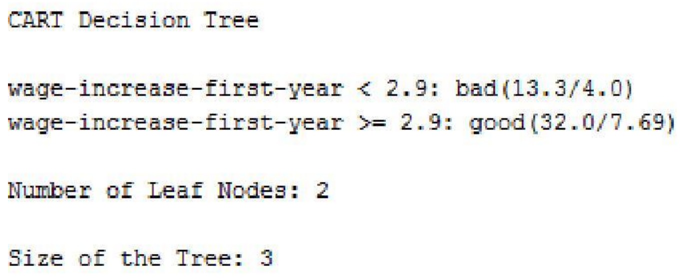


```
CART Decision Tree

wage-increase-first-year < 2.65: bad(13.0/2.26)
wage-increase-first-year >= 2.65: good(34.73/7.0)

Number of Leaf Nodes: 2

Size of the Tree: 3
```

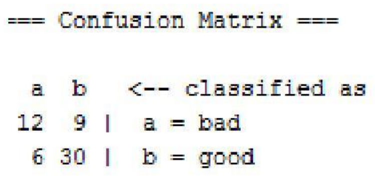


```
=== Confusion Matrix ===

  a  b   <-- classified as
 13  7 |  a = bad
  7 30 |  b = good
```

b. 5 % of Noise introduced across both the classes:

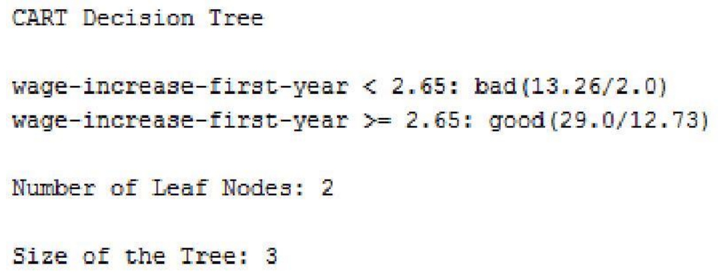


```
CART Decision Tree

wage-increase-first-year < 2.9: bad(13.3/4.0)
wage-increase-first-year >= 2.9: good(32.0/7.69)

Number of Leaf Nodes: 2

Size of the Tree: 3
```

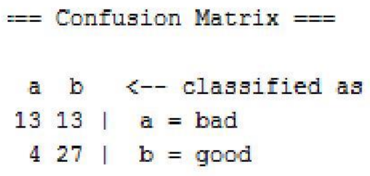


```
=== Confusion Matrix ===

  a  b   <-- classified as
 12  9 |  a = bad
  6 30 |  b = good
```

c. In addition 5 % of Noise introduced to class bad (misclassify good as bad):



```
CART Decision Tree

wage-increase-first-year < 2.65: bad(13.26/2.0)
wage-increase-first-year >= 2.65: good(29.0/12.73)

Number of Leaf Nodes: 2

Size of the Tree: 3
```



```
=== Confusion Matrix ===

  a  b   <-- classified as
 13 13 |  a = bad
  4 27 |  b = good
```

Analysis:

We could see that only a slight variation with respect to the decision tree was observed in these cases. Also the performance degradation was as expected. The skewed noises added to bad class made the classifier to predict more instances as bad which can be clearly observed from the confusion matrix for the second column while comparing the second and third run of the noise induction steps shown here.

Conclusion

The Simple CART algorithm performed really well with the Labor dataset as there were numerous missing values due to which Decision Stump was prone to more classification errors inspite of a class attribute of binary type. Also there were not many instances in the dataset to enhance either of the classifiers. However, useOneSE=true parameter enhanced the Simple CART classifier's performance considerably even on an unseen test data. The class distribution in the dataset plays a critical role in training a classifier especially while handling missing values or noisy dataset. Overall, Simple CART is the better of the two for the Labor dataset.

VII. ANALYSIS OF DIABETES DATASET

Test Options

On Training Set

These experiments were done on the training set to observe the behavior and not for evaluating the classifier performance.

Supplied Test Set

The dataset was manually divided into two sets, training and test sets, in the ratio 2:1 of original dataset so that there are no common instances between the two sets. The classifiers were built using this training set and then the test set whose instances are unseen for the classifiers was supplied and the performance of the classifier is observed. Also, the class distributions of the classes tested_negative and tested_positive in the test set is maintained in 162:19 ratio. Number of instances in training and test are 507 and 261 respectively.

Percentage Split

The vote dataset is split into 2:1 ratio randomly based on this option as the percentage split that we have used here is 66. The random split could take any instances in training and test and thus there is a possibility for a skewed class distribution in the training or test sets that could not lead to a fair evaluation of the classifier depending on the random seed value.

10-fold Cross Validation

The details mentioned in Iris dataset analysis applies here as well.

Analysis of 10-fold Cross Validation

Below is an instance of subsets derived from the original data set of 768 instances. It can be easily seen that datasets are divided into 10 almost equal size of subsets preserving a fair class distribution in each of the folds as in the original dataset.

Filter = StratifiedRemoveFolds	Instances	Class Distribution	
Fold		tested_negative	tested_positive
1	77	50	27
2	77	50	27
3	77	50	27
4	77	50	27
5	77	50	27
6	77	50	27
7	77	50	27
8	77	50	27
9	76	50	26
10	76	50	26

Performance Analysis of the Classifiers

1. Decision Stump

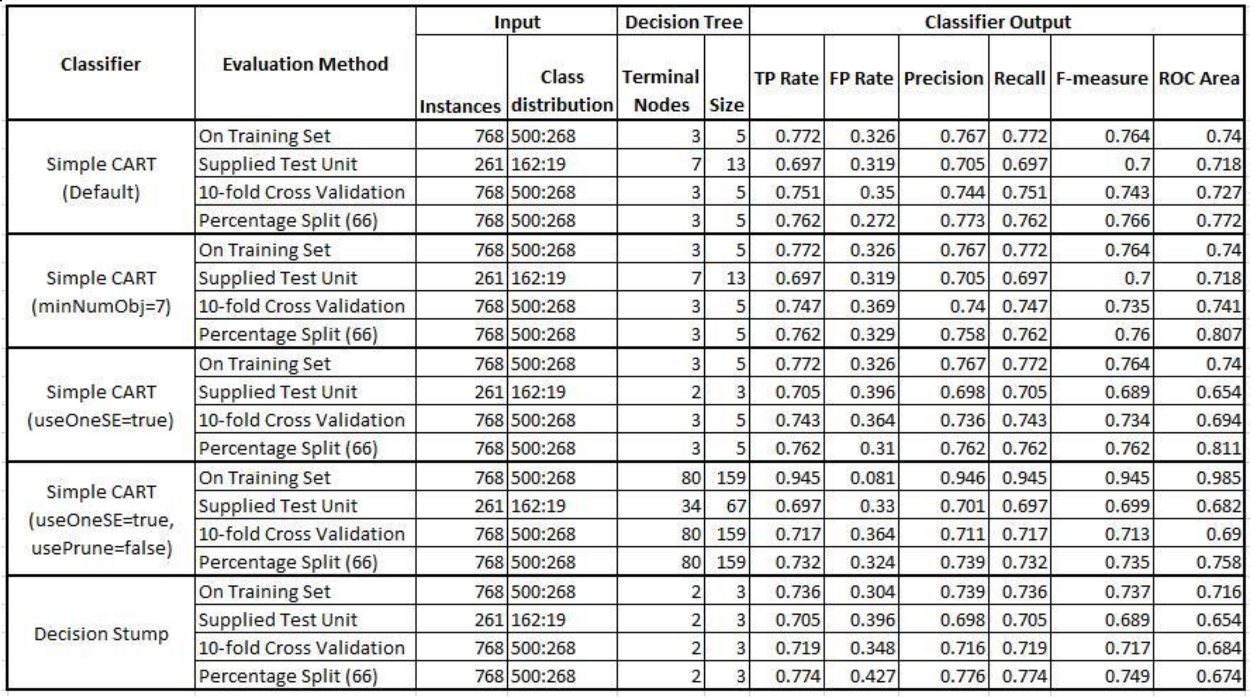
This dataset has only two classes and there are more number of instances are there to train the classifier. Also, it has only numeric type attributes except for the class attribute. Though the decision stump algorithm is able to classify the unseen test data (considering the supplied test case in the table) as good as the simple cart algorithm, the overall area under ROC curve is lesser than the simple cart algorithm. Considering more independent test data, the simple cart's prediction is better than the decision stump's predictions.

2. Simple CART

It is observed that this algorithm performs better than decision stump overall as mentioned in the above case. The algorithm is having a simpler model with better accuracy while compared to the decision stump. Moreover, we can clearly see the overfitting scenarion happening when the useOneSE=true and usePrune=false case for the simple cart in the below table. The classifier reached an accuracy with area under ROC curve as 0.985 on training set but at the same time, area under ROC curve for an unseen test set, was 0.682. Out of all the dataset and experiments, here we can see the impact of overfitting model with a tree size of 159 with 80 terminal nodes.

Moreover, useOneSE=true doesn't seem to improve the accuracy rather it increases the FP rate.

Sample output of each of the test experiment for each classifier with respect to the respective classifier parameters are mentioned in the below table:



Classifier	Evaluation Method	Input		Decision Tree		Classifier Output					
		Instances	Class distribution	Terminal Nodes	Size	TP Rate	FP Rate	Precision	Recall	F-measure	ROC Area
Simple CART (Default)	On Training Set	768	500:268	3	5	0.772	0.326	0.767	0.772	0.764	0.74
	Supplied Test Unit	261	162:19	7	13	0.697	0.319	0.705	0.697	0.7	0.718
	10-fold Cross Validation	768	500:268	3	5	0.751	0.35	0.744	0.751	0.743	0.727
	Percentage Split (66)	768	500:268	3	5	0.762	0.272	0.773	0.762	0.766	0.772
Simple CART (minNumObj=7)	On Training Set	768	500:268	3	5	0.772	0.326	0.767	0.772	0.764	0.74
	Supplied Test Unit	261	162:19	7	13	0.697	0.319	0.705	0.697	0.7	0.718
	10-fold Cross Validation	768	500:268	3	5	0.747	0.369	0.74	0.747	0.735	0.741
	Percentage Split (66)	768	500:268	3	5	0.762	0.329	0.758	0.762	0.76	0.807
Simple CART (useOneSE=true)	On Training Set	768	500:268	3	5	0.772	0.326	0.767	0.772	0.764	0.74
	Supplied Test Unit	261	162:19	2	3	0.705	0.396	0.698	0.705	0.689	0.654
	10-fold Cross Validation	768	500:268	3	5	0.743	0.364	0.736	0.743	0.734	0.694
	Percentage Split (66)	768	500:268	3	5	0.762	0.31	0.762	0.762	0.762	0.811
Simple CART (useOneSE=true, usePrune=false)	On Training Set	768	500:268	80	159	0.945	0.081	0.946	0.945	0.945	0.985
	Supplied Test Unit	261	162:19	34	67	0.697	0.33	0.701	0.697	0.699	0.682
	10-fold Cross Validation	768	500:268	80	159	0.717	0.364	0.711	0.717	0.713	0.69
	Percentage Split (66)	768	500:268	80	159	0.732	0.324	0.739	0.732	0.735	0.758
Decision Stump	On Training Set	768	500:268	2	3	0.736	0.304	0.739	0.736	0.737	0.716
	Supplied Test Unit	261	162:19	2	3	0.705	0.396	0.698	0.705	0.689	0.654
	10-fold Cross Validation	768	500:268	2	3	0.719	0.348	0.716	0.719	0.717	0.684
	Percentage Split (66)	768	500:268	2	3	0.774	0.427	0.776	0.774	0.749	0.674

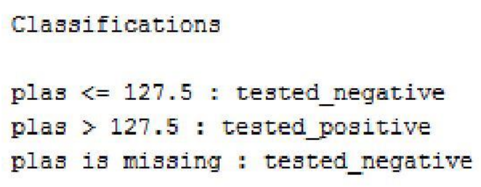
Impact of Missing Values

The given Diabetes dataset is small with lots of missing values and we could see that the performance of the classifiers are very much impacted overall based on the classifier outputs. We consider 10-fold cross validation option as it gives a generalized value unlike choosing one random test set or percentage split.

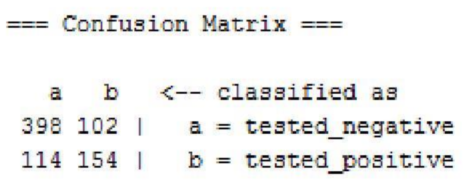
1. Decision Stump

Test Option: 10-fold Cross Validation   Input: 768 instances

a. With default dataset:

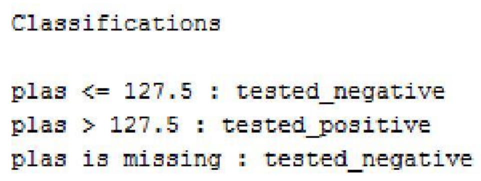


```
Classifications

plas <= 127.5 : tested_negative
plas > 127.5 : tested_positive
plas is missing : tested_negative
```

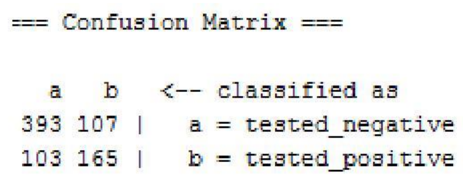


```
=== Confusion Matrix ===

   a   b   <-- classified as
 398 102 |   a = tested_negative
 114 154 |   b = tested_positive
```

b. Missing values introduced to plas attribute for 5% of the instances

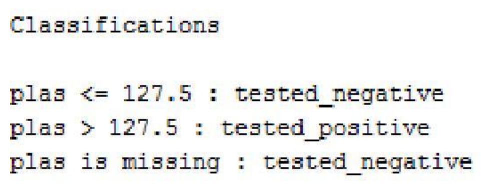


```
Classifications

plas <= 127.5 : tested_negative
plas > 127.5 : tested_positive
plas is missing : tested_negative
```

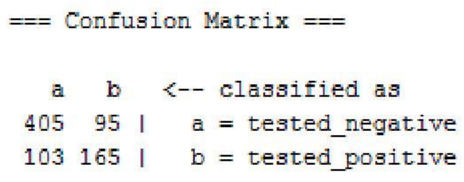


```
=== Confusion Matrix ===

   a   b   <-- classified as
 393 107 |   a = tested_negative
 103 165 |   b = tested_positive
```

c. Missing values introduced to plas attribute for 5% of the instances only for tested_negative class.



```
Classifications

plas <= 127.5 : tested_negative
plas > 127.5 : tested_positive
plas is missing : tested_negative
```



```
=== Confusion Matrix ===

   a   b   <-- classified as
 405  95 |   a = tested_negative
 103 165 |   b = tested_positive
```

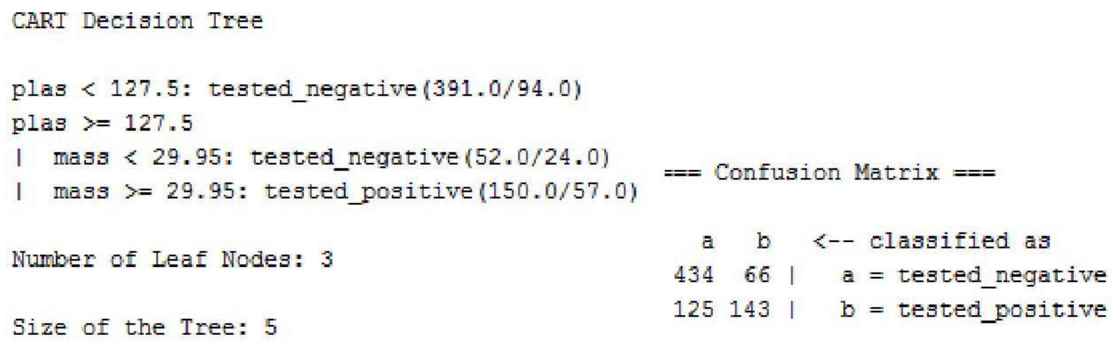
Analysis:

We can observe that with missing values in both the classes, the classifier was predicting more tested_positive cases correctly with a degradation in prediction for tested_negative cases. However when further missing values were added only to tested_negative cases, the classifier was predicting more tested_negative cases correctly with no change in tested_positive cases. The decision tree remained the same in all the cases. This is because the missing cases were classified as tested_negative by default. More missing values, more cases for tested_negative which may be either wrong(second run) or correct(third run - skewed missing value cases on tested_negative). The skewed missing value introduction to tested_negative class favored the output as that the default classification if plas attribute is missing.

2. Simple CART

Test Option: 10-fold Cross Validation Input: 768 instances
useOneSE=true

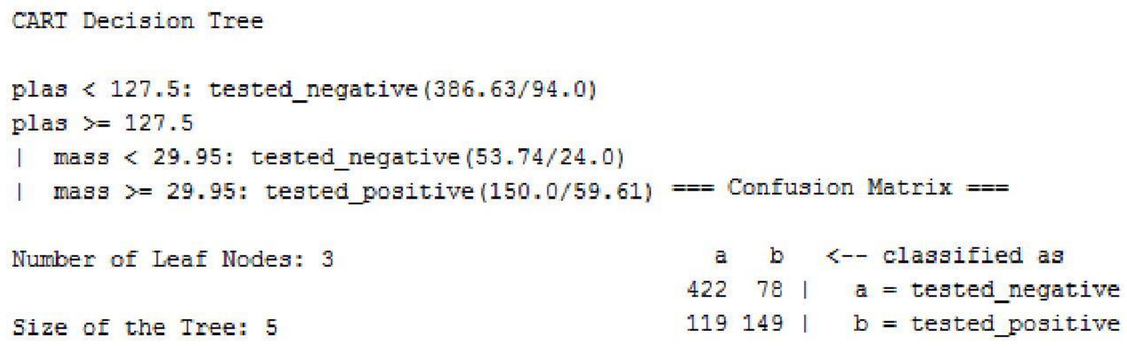
a. With default dataset:



```
CART Decision Tree

plas < 127.5: tested_negative(391.0/94.0)
plas >= 127.5
|  mass < 29.95: tested_negative(52.0/24.0)
|  mass >= 29.95: tested_positive(150.0/57.0)

Number of Leaf Nodes: 3

Size of the Tree: 5
```

```
=== Confusion Matrix ===

   a   b   <-- classified as
 434  66 |   a = tested_negative
 125 143 |   b = tested_positive
```

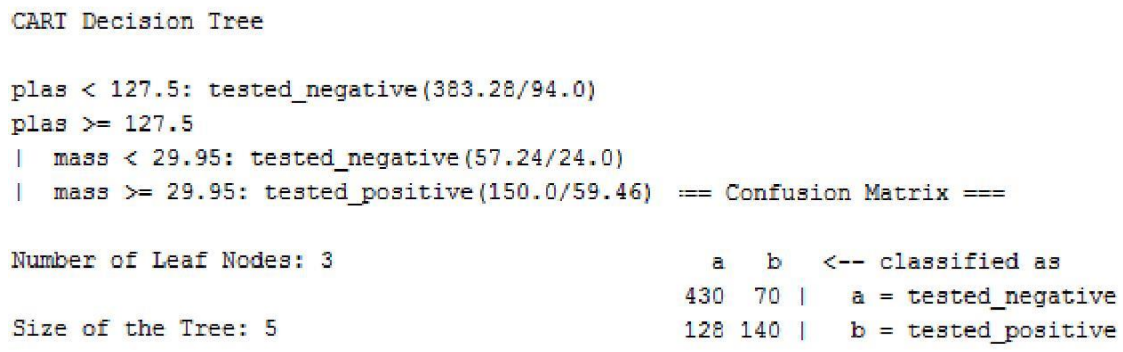
b. Missing values introduced to plas attribute for 5% of the instances



```
CART Decision Tree

plas < 127.5: tested_negative(386.63/94.0)
plas >= 127.5
|  mass < 29.95: tested_negative(53.74/24.0)
|  mass >= 29.95: tested_positive(150.0/59.61)

Number of Leaf Nodes: 3

Size of the Tree: 5
```

```
=== Confusion Matrix ===

   a   b   <-- classified as
 422  78 |   a = tested_negative
 119 149 |   b = tested_positive
```

c. Missing values introduced to plas attribute for 5% of the instances only for tested_negative class.



```
CART Decision Tree

plas < 127.5: tested_negative(383.28/94.0)
plas >= 127.5
|  mass < 29.95: tested_negative(57.24/24.0)
|  mass >= 29.95: tested_positive(150.0/59.46)

Number of Leaf Nodes: 3

Size of the Tree: 5
```

```
=== Confusion Matrix ===

   a   b   <-- classified as
 430  70 |   a = tested_negative
 128 140 |   b = tested_positive
```

Analysis:

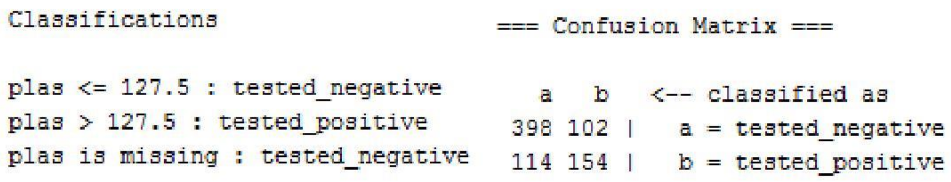
The simple cart algorithm handles the missing values well using fractional instances method, so that split attribute's missing values are replaced by all possible outcomes for that attribute with fractions based on the popularity. Hence the missing values added to plas were handled by the next level of the tree handled by mass attribute's condition which classified the instances much better.

Impact of Noise

1. Decision Stump

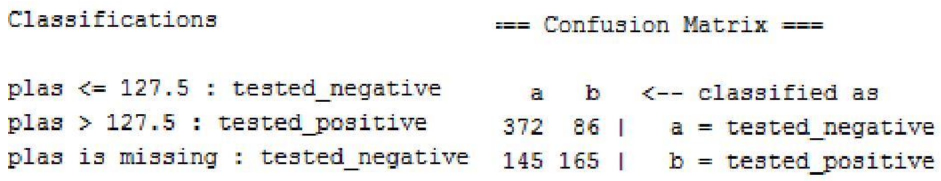
Test Option: 10-fold Cross Validation Input: 768 instances
useOneSe=true

Without Noise:



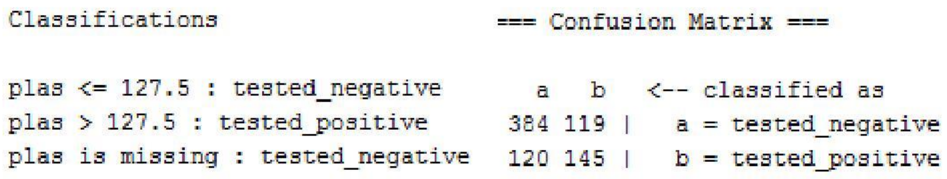
```
Classifications                   === Confusion Matrix ===

plas <= 127.5 : tested_negative     a   b   <-- classified as
plas > 127.5 : tested_positive    398 102 |   a = tested_negative
plas is missing : tested_negative 114 154 |   b = tested_positive
```

b. 5 % of Noise introduced only to tested_negative classes (misclassified testsed_negative to tested_positive):



```
Classifications                   === Confusion Matrix ===

plas <= 127.5 : tested_negative     a   b   <-- classified as
plas > 127.5 : tested_positive    372  86 |   a = tested_negative
plas is missing : tested_negative 145 165 |   b = tested_positive
```

c. In addition 5 % of Noise introduced to class bad (misclassify good as bad):



```
Classifications                   === Confusion Matrix ===

plas <= 127.5 : tested_negative     a   b   <-- classified as
plas > 127.5 : tested_positive    384 119 |   a = tested_negative
plas is missing : tested_negative 120 145 |   b = tested_positive
```

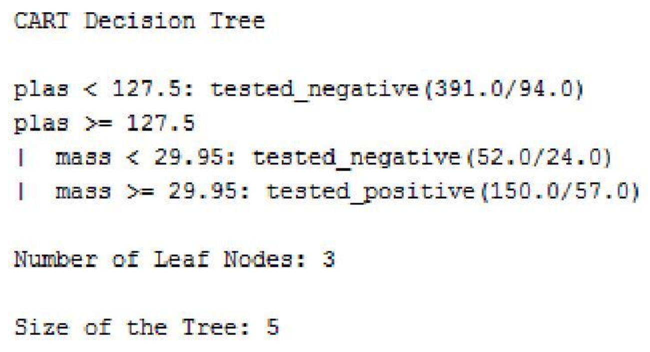
Analysis:

As more noise were introduced to the dataset, the classification accuracy was impacted but the decision tree was not at all changed. So, the classes that misclassified as tested_positive reduced in count (corresponding to first row second column) for the second run of skewed noise induction. However, the noises were induced equally, the balance was maintained and the accuracy was also reduced.

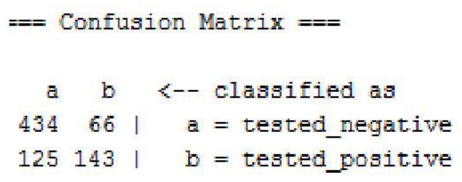
2. Simple CART

Test Option: 10-fold Cross Validation Input: 768 instances
useOneSE=true

a. Without Noise:



```
CART Decision Tree

plas < 127.5: tested_negative(391.0/94.0)
plas >= 127.5
|  mass < 29.95: tested_negative(52.0/24.0)
|  mass >= 29.95: tested_positive(150.0/57.0)

Number of Leaf Nodes: 3

Size of the Tree: 5
```



```
=== Confusion Matrix ===

   a   b   <-- classified as
 434  66 |   a = tested_negative
 125 143 |   b = tested_positive
```

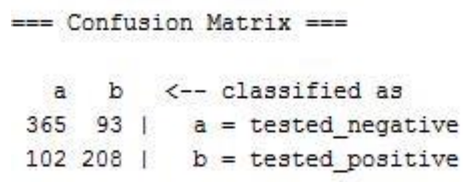
b. 5 % of Noise introduced across both the classes:

CART Decision Tree

```
plas < 127.5
| age < 28.5
| | mass < 45.4
| | | preg < 6.5
| | | | pedi < 0.485: tested_negative(159.0/13.0)
| | | | pedi >= 0.485
| | | | | pedi < 0.492: tested_positive(2.0/0.0)
| | | | | pedi >= 0.492
| | | | | | mass < 32.25: tested_negative(46.0/3.0)
| | | | | | mass >= 32.25
| | | | | | | mass < 34.3
| | | | | | | | skin < 30.5: tested_negative(5.0/2.0)
| | | | | | | | skin >= 30.5: tested_positive(4.0/0.0)
| | | | | | | mass >= 34.3: tested_negative(24.0/7.0)
| | | preg >= 6.5: tested_positive(2.0/0.0)
| | mass >= 45.4: tested_positive(3.0/1.0)
| age >= 28.5
| | plas < 96.5
| | | plas < 50.5: tested_positive(3.0/0.0)
| | | plas >= 50.5
| | | | insu < 153.0: tested_negative(48.0/6.0)
| | | | insu >= 153.0: tested_positive(2.0/0.0) |
| plas >= 96.5
| | | mass < 26.95
| | | | mass < 9.65: tested_positive(2.0/0.0)
| | | | mass >= 9.65: tested_negative(25.0/4.0)
| | | mass >= 26.95
| | | | pedi < 0.5055000000000001
| | | | | skin < 27.5
| | | | | | pres < 85.0
| | | | | | | age < 34.5
| | | | | | | | preg < 4.5: tested_negative(6.0/1.0)
| | | | | | | | preg >= 4.5: tested_positive(5.0/0.0)
| | | | | | | age >= 34.5: tested_positive(21.0/7.0)
| | | | | | pres >= 85.0
| | | | | | | mass < 33.45: tested_negative(5.0/0.0)
| | | | | | | mass >= 33.45
| | | | | | | | plas < 119.5: tested_positive(3.0/0.0)
| | | | | | | | plas >= 119.5: tested_negative(3.0/0.0)
| | | | | skin >= 27.5
| | | | | | mass < 43.099999999999994
| | | | | | | pres < 67.0
| | | | | | | | plas < 101.5: tested_negative(3.0/0.0)
| | | | | | | | plas >= 101.5: tested_positive(3.0/0.0)
| | | | | | | pres >= 67.0: tested_negative(21.0/2.0)
| | | | | | mass >= 43.099999999999994: tested_positive(3.0/1.0)
| | | | pedi >= 0.5055000000000001
| | | | | pres < 69.0
| | | | | | preg < 5.0
| | | | | | | plas < 121.0: tested_negative(5.0/0.0)
| | | | | | | plas >= 121.0: tested_positive(2.0/0.0)
| | | | | | preg >= 5.0: tested_positive(3.0/0.0)
| | | | | pres >= 69.0: tested_positive(27.0/3.0)
plas >= 127.5
| mass < 29.95
| | plas < 144.5
| | | skin < 33.5: tested_negative(31.0/6.0)
| | | skin >= 33.5: tested_positive(3.0/0.0)
| | plas >= 144.5
| | | age < 25.5: tested_negative(4.0/0.0)
| | | age >= 25.5
| | | | age < 61.0
| | | | | pres < 74.5: tested_positive(16.0/1.0)
| | | | | pres >= 74.5
| | | | | | preg < 5.5: tested_negative(4.0/1.0)
| | | | | | preg >= 5.5: tested_positive(5.0/1.0)
| | | | age >= 61.0: tested_negative(4.0/0.0) |
mass >= 29.95
| | plas < 157.5
| | | age < 30.5
| | | | pres < 61.0: tested_positive(10.0/0.0)
| | | | pres >= 61.0
| | | | | insu < 271.0
| | | | | | pres < 85.5
| | | | | | | pres < 73.0
| | | | | | | | pedi < 0.3185: tested_negative(4.0/1.0)
| | | | | | | | pedi >= 0.3185
| | | | | | | | | pres < 64.5: tested_negative(2.0/0.0)
| | | | | | | | | pres >= 64.5: tested_positive(7.0/0.0)
| | | | | | | pres >= 73.0: tested_negative(11.0/1.0)
| | | | | | pres >= 85.5: tested_positive(5.0/1.0)
| | | | | insu >= 271.0: tested_negative(8.0/0.0)
| | | age >= 30.5
| | | | pedi < 0.3965
| | | | | mass < 45.55
| | | | | | pres < 92.0
| | | | | | | pedi < 0.1365: tested_positive(2.0/0.0)
| | | | | | | pedi >= 0.1365: tested_negative(12.0/5.0)
| | | | | | pres >= 92.0: tested_positive(2.0/0.0)
| | | | | mass >= 45.55: tested_positive(5.0/0.0)
| | | | pedi >= 0.3965: tested_positive(37.0/2.0) |
| plas >= 157.5: tested_positive(81.0/11.0)
```

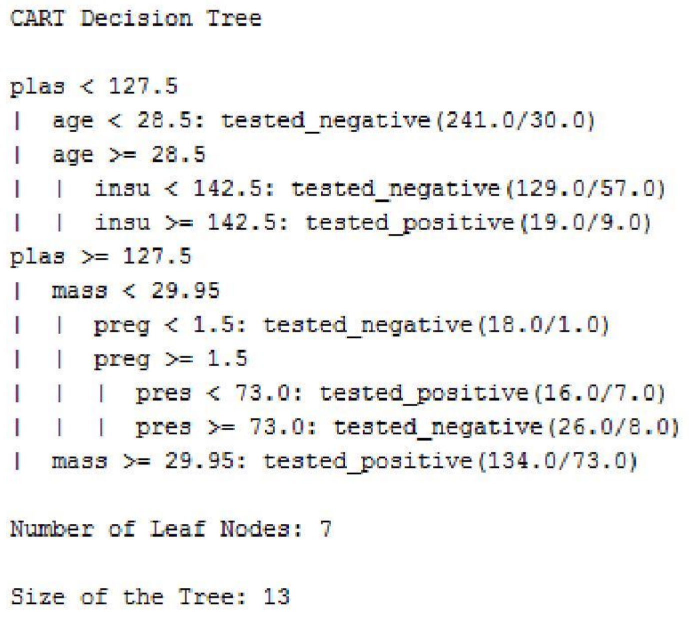
Number of Leaf Nodes: 47

Size of the Tree: 93

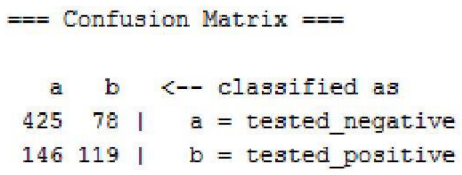


```
=== Confusion Matrix ===

   a   b   <-- classified as
 365  93 |   a = tested_negative
 102 208 |   b = tested_positive
```

c. In addition 5 % of Noise introduced to class bad (misclassify good as bad):



```
CART Decision Tree

plas < 127.5
|  age < 28.5: tested_negative(241.0/30.0)
|  age >= 28.5
|  |  insu < 142.5: tested_negative(129.0/57.0)
|  |  insu >= 142.5: tested_positive(19.0/9.0)
plas >= 127.5
|  mass < 29.95
|  |  preg < 1.5: tested_negative(18.0/1.0)
|  |  preg >= 1.5
|  |  |  pres < 73.0: tested_positive(16.0/7.0)
|  |  |  pres >= 73.0: tested_negative(26.0/8.0)
|  mass >= 29.95: tested_positive(134.0/73.0)

Number of Leaf Nodes: 7

Size of the Tree: 13
```



```
=== Confusion Matrix ===

   a   b   <-- classified as
 425  78 |   a = tested_negative
 146 119 |   b = tested_positive
```

Analysis:

Surprisingly, we were able to visualize the impact of skewed noise induction on simple cart algorithm. The decision tree of size 5 got exploded into a tree of size 93 with extremely complicated decision tree with a bad impact on the accuracy of the classifier. However when the noise induction was balanced, this extremely complicate tree got reduced to complex tree of size 13 with much worsening effect on performance of the classifier.

Conclusion

It is evident that Decision Stump is not as good as Simple CART in classifying the Diabetes dataset. The area under the ROC curve for Simple CART classifier was much better than the Decision Stump classifier. Impact of a skewed noise in the dataset (with respect to class distribution) was immense in Simple CART algorithm with not much substantial improvement in classification accuracy. However, on an Independent test data, Simple CART performed the classifications with good accuracy.

VIII. SUMMARY

Overall, the Decision Stump classifier is a simple powerful classfier whereas Simple CART is a sophisticated classifier which relatively performs well on most of the cases. Here are some of the pros and cons of each of the algorithms identified by the analysis.

A. Decision Stump Classifier:

Pros

- Performs extremely well in case the dataset has nominal attributes and class attribute of two classes.
- Immune to missing values or noise on attributes not part of the main split.
- Always forms a simple decision tree.
- No worries of overfitting.
- Performs well on a dataset with disproportional class distribution.

Cons

- Prone to form a underfit model.
- No parameters to configure or tune as per the dataset.
- Cannot classify complicated datasets successfully
- Class attribute should be having only two values at maximum.

B. Simple CART

Pros

- Good performance on most of the datasets.
- Parameters options let configure and tune the classifier as per the dataset to be classified.
- Ability to handle missing values in attributes.

Cons

- May lead to extremly complicated decision trees.
- May lead to decision trees of large tree size.
- Prone to chances of overfitting.
- Sensitive to the class disctribution of the training set.

In addition to these classifier algorithms, the evaluation method also plays an important role in identifying a suitable classifier. A 10-fold Cross validation method is better than the holdout method or percentage split method as the variance will be maintained even for a smaller dataset. Also it provides a overall generalized performance values for a classifier and maintains a fair representation of the class distribution in the training and test datasets by stratified approach.

Based on all the analysis presented, the Simple CART outperformed Decision Stump in most of the cases. However, Decision Stump was really powerful for a nominal type dataset due to the smaller effective tree size. So, as long as Simple CART has a simpler decision tree it can be used for all the four datasets even considering noise and missing values impact.