**EXPLORING WEKA SOFTWARE TOOL AND ANALYZING THE RESULTS PRODUCED BY DIFFERENT DATASETS WITH RESPECT TO DIFFERENT CLASSIFIERS.**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENTS** | **PAGE NO.** |
|  | Introduction to Weka tool and different types of parameters in Weka tool for analyzing the data. | 2 |
| 1. | Analyzing the IRIS datasets.   1. Number of attributes present. 2. Min/Max values, Standard Deviation for each attribute. 3. Class attribute of this Dataset. 4. SimpleCART and DecisionStump – Parameters for each of these decision tree algorithm. 5. Analyze the results obtained by running this dataset under both the decision tree algorithm. 6. Classification accuracy obtained on training and on test set. 7. Difference (if any) between the results from training set and the results from the test set. 8. Change the parameters for training and test set and record the difference. 9. Introducing some noise – Analyze the results. 10. 10 cross fold validation – Analyze the difference in accuracy obtained when using this dataset with both decision tree algorithms. 11. Size of training set and the test set in each cross-validation iteration. 12. Class distribution with respect to test/training set created in cross validation. | 3 |
| 2. | Analyzing the VOTE dataset.   1. Number of attributes present. 2. Min/Max values, Standard Deviation for each attribute. 3. Class attribute of this Dataset. 4. SimpleCART and DecisionStump – Parameters for each of these decision tree algorithm. 5. Analyze the results obtained by running this dataset under both the decision tree algorithm. 6. Classification accuracy obtained on training and on test set. 7. Difference (if any) between the results from training set and the results from the test set. 8. Change the parameters for training and test set and record the difference. 9. Introducing some noise – Analyze the results. 10. 10 cross fold validation – Analyze the difference in accuracy obtained when using this dataset with both decision tree algorithms. 11. Size of training set and the test set in each cross-validation iteration. 12. Class distribution with respect to test/training set created in cross validation. |  |
| 3. | Analyzing the LABOR dataset.   1. Number of attributes present. 2. Min/Max values, Standard Deviation for each attribute. 3. Class attribute of this Dataset. 4. SimpleCART and DecisionStump – Parameters for each of these decision tree algorithm. 5. Analyze the results obtained by running this dataset under both the decision tree algorithm. 6. Classification accuracy obtained on training and on test set. 7. Difference (if any) between the results from training set and the results from the test set. 8. Change the parameters for training and test set and record the difference. 9. Introducing some noise – Analyze the results. 10. 10 cross fold validation – Analyze the difference in accuracy obtained when using this dataset with both decision tree algorithms. 11. Size of training set and the test set in each cross-validation iteration. 12. Class distribution with respect to test/training set created in cross validation. |  |
| 4. | Analyzing the DIABETES datasets given as a part of this assignment.   1. Number of attributes present. 2. Min/Max values, Standard Deviation for each attribute. 3. Class attribute of this Dataset. 4. SimpleCART and DecisionStump – Parameters for each of these decision tree algorithm. 5. Analyze the results obtained by running this dataset under both the decision tree algorithm. 6. Classification accuracy obtained on training and on test set. 7. Difference (if any) between the results from training set and the results from the test set. 8. Change the parameters for training and test set and record the difference. 9. Introducing some noise – Analyze the results. 10. 10 cross fold validation – Analyze the difference in accuracy obtained when using this dataset with both decision tree algorithms. 11. Size of training set and the test set in each cross-validation iteration. 12. Class distribution with respect to test/training set created in cross validation. |  |

**Introduction to WEKA:**

* Weka is a software tool designed to analyze raw data from disparate sources.
* It was designed to visualize the data.

**Different tasks associated with WEKA tool for processing and analyzing the data.**

WEKA – WAIKATO ENVIRONMENT FOR KNOWLEDGE ANALYSIS

Visualize data

Select Attributes

Classifiers

Data Pre-processing

Associate

Clustering

**Decision tree Classifiers and their parameters:**

* An inverted tree like structure with root node at the top where it makes decision by adding some test on an attribute.

If x(i) >2

No

Yes

If x(i) = 3

The point or the value lies below x=2

No

Yes

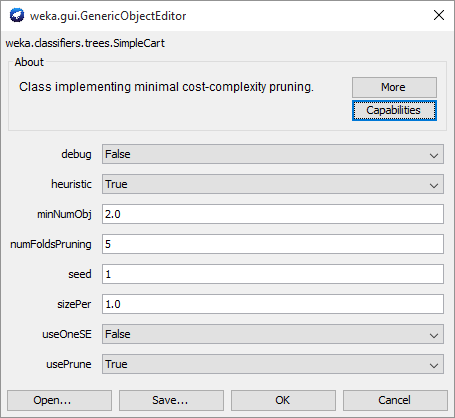
**And so on…. (Tree structure and visualizing the points)**

**SimpleCART decision tree algorithm:**

* It is based on CART analysis (Classification and Regression tree) that uses historical data.
* This decision tree is a technique which generates the binary decision tree.
* Since the output tree is in Binary form, this tree generates only 2 children.
* Missing data in the dataset is simply ignored by this algorithm and hence is suitable for Training data.
* This algorithm uses a large set of test samples and cross validation methods to identify the best tree from many sequence of trees in the pruning process.

**Parameters for SimpleCART decision tree algorithm:**

* **Debug**: If this is set to true, then the console (where the output is generated) will have more information for the user to analyze the data.
* **Heuristic**: This is used for binary split for nominal attributes in multi-class problems (default yes).
* **minNumObj:** The minimal number of observations at the terminal nodes (default value is 2).
* **numFoldsPruning:** The number of folds in an internal cross validation.
* **Seed:** The random number seed to be used.
* **sizePer:** The percentage of the training set size (0-1, 0 not included).
* **useOneSE:** Use the 1SE rule to make pruning decisoin.
* **usePrune:** Use minimal cost-complexity pruning (default yes).



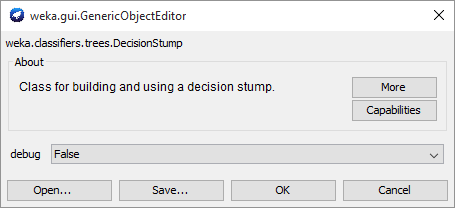
**Decision Stump tree algorithm:**

* This model is sometimes referred to as “One-level decision tree”.
* When this algorithm is used, it makes a prediction based on the value of 1 single input.
* Missing value will be treated as another category.
* It is also called weak learners (used for them majorily).

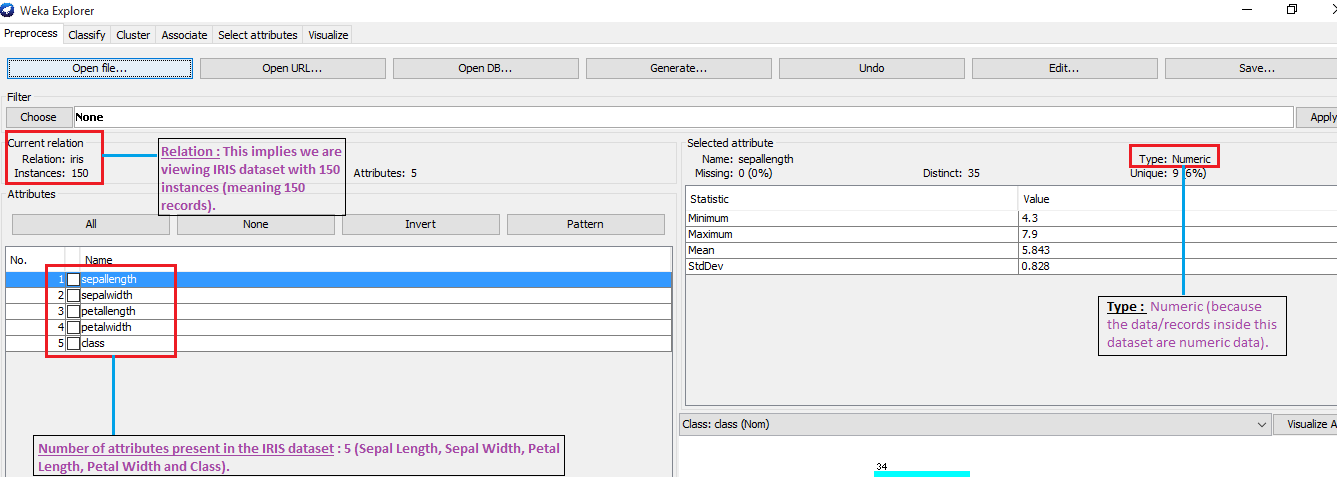
**Parameters for a Decision stump tree algorithm:**

* Since there is only one-level decision made by this algorithm, there is only one parameter – Debug.

**Debug:** It can be set to true when any additional information regarding the results are to be known in the console.



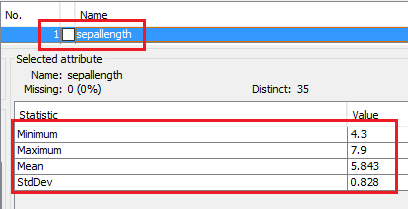
**1a), 1b), 1c)** - **Analyzing IRIS Dataset**

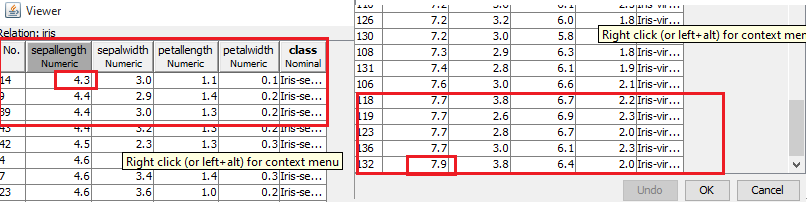


**Inference from the above screenshot:**

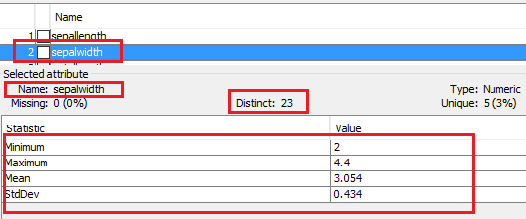
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset Name** | **Number of attributes present** | **Name of the attributes** | **Class attribute – Name of the instances** | **Number of attribute instances** |
|  |  |  |  |  |
| **IRIS dataset** | 5 | Sepal Length, Sepal Width, Petal Length, Petal width and Class. | Setosa, Virginica and Veriscolor | 150 (Setosa, Virginica, Veriscolor – each 50 instances) |

**1st Attribute - IRIS- Sepal Length:** Distinct: 35 (Number of different values that the data contains for this attribute).

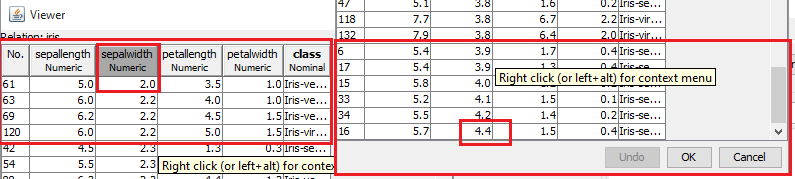


**View inside the Dataset with Minimum and Maximum values:**

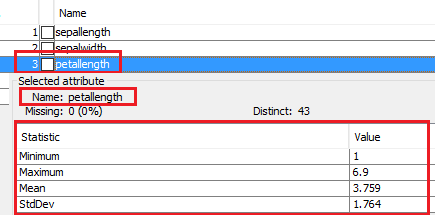
**2nd Attribute - IRIS- Sepal width:** - Distinct: 23



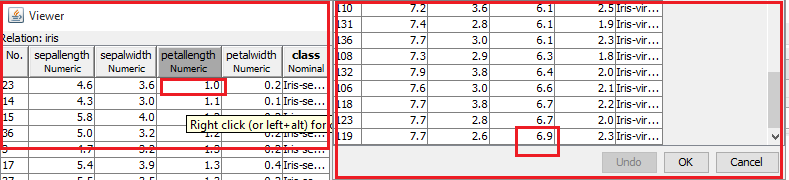
**View inside the Dataset for Minimum and maximum values:**

****

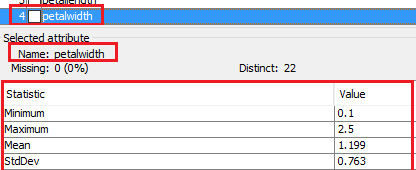
**3rd Attribute - IRIS- Petal length:** - Distinct: 43



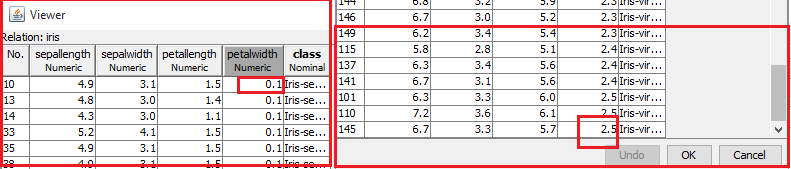
**View inside the Dataset for Minimum and maximum values:**



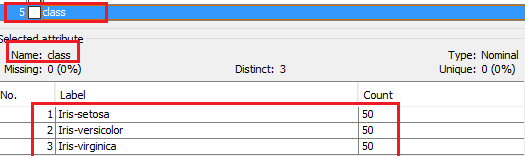
**4th Attribute – IRIS – Petal width: Distinct**



**View inside the dataset for Minimum and maximum values:**



**5th Attribute – Class (This is a Nominal type of attribute which is not numeric).**

****

**1a. 1b. 1c.) Inference from the above screenshots for IRIS Dataset:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No**  **(No.**  **Of**  **Attributes)** | **Name of the attribute** | **Type of the attribute** | **Minimum value** | **Maximum value** | **Mean – Average of all the values** | **Standard deviation – square root of variance.** | **If Class attribute – specify the name of the Class attributes.** |
|  |  |  |  |  |  |  |  |
| 1 | Sepal Length | Numeric | 4.3 | 7.9 | 5.843 | 0.828 | N/A |
|  |  |  |  |  |  |  |  |
| 2 | Sepal Width | Numeric | 2 | 4.4 | 3.054 | 0.434 | N/A |
|  |  |  |  |  |  |  |  |
| 3 | Petal Length | Numeric | 1 | 6.9 | 3.759 | 1.764 | N/A |
|  |  |  |  |  |  |  |  |
| 4 | Petal Width | Numeric | 0.1 | 2.5 | 1.199 | 0.763 | N/A |
|  |  |  |  |  |  |  |  |
| 5 | Class | Nominal | N/A | N/A | N/A | N/A | **IRIS** – Setosa, Virginica, Veriscolor. |

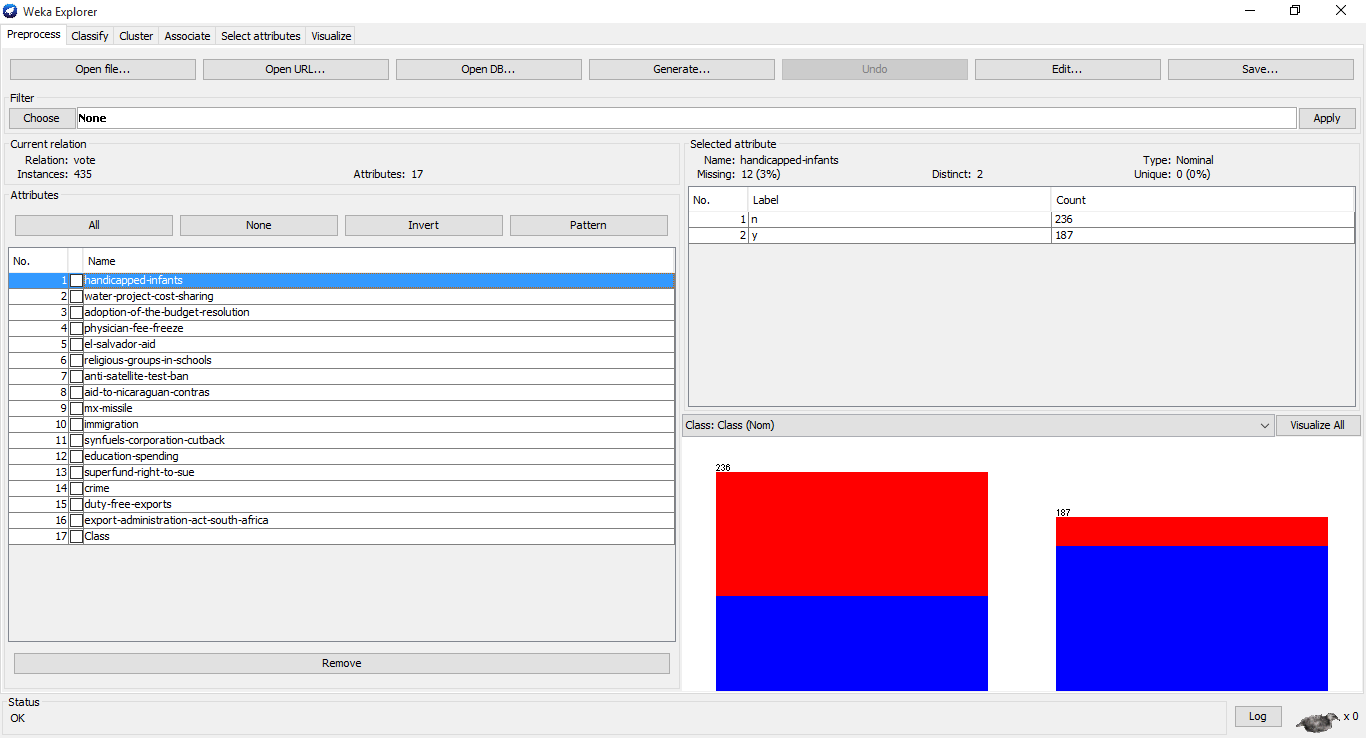
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function to be Tested:**  **Test IRIS dataset under DecisionStump Tree algorithm** and record the results for the below test cases without changing its parameters. | | | | |
| **Objectives/Conditions:**   1. Create a separate Dataset – IRIS.TRAINING to satisfy test case – TC 1d.) 2. Create 1 separate dataset in the data folder – IRIS.MISSING to satisfy test case – TC 1f.) | | | | **Test cases #**  **TC:**  **1a – 1i** |
| **Test data library:**  Weka-3-6\data-Iris.arff, Weka-3-6\data-Iris.Training.arff, Weka-3-6\data-Iris.Test.arff and Weka-3-6\data-Iris.Datastump.arff | | | |  |
| **TC-ID** | **Action/Input data** | **Results: Classification accuracy, Confusion matrix and Correct/incorrect instances.** | **Inference** | |
| TC 1a.) | 1. Set cross validation (by default) folds = 10. 2. Set the algorithm DecisionStump under tree classifier. | The above doc has the test result run for the input data specified. | Correct Instances for this run (against 150 instances) : 100 (as this is derived from only 2 class attributes)  Incorrect instances: 50 (as this is the 3rd attribute in the class) – as seen in the confusion matrix (attached in the test result doc).  In this case, petallength is taken as the input and a decision is made by this algorithm as below:  If petal length <= 2.5, then the class is determined as IRIS-SETOSA.  If petal length = 2.5 then the class is determined as IRIS-VERISCOLOR.  IRIS-VIRGINICA will not be considered here as the decision can be made only on 2 classes and those classes are derived based only on a single input (instance attribute).  **Thus leading to Classification accuracy: 66.667 %** (this is based on correct instances) which is a poor performer when it comes to IRIS dataset and not working fine.   * 150 instances = 100 % * Only 100 instances (in our case) = ?? * (100\*100)/150 = 66.667%   **33.33% based on the incorrect instances** (50 incorrect). | |
| TC 1b.) | 1. Change the testing option to “Training set” 2. Run the algorithm. | Testing on the same set that the classifier is trained on.    The above doc has the test run for the input specified. | **The result is same as the TC 1a.**   1. Training set is also an option which will do a manipulation on all the 150 instances. 2. The accuracy remains unchanged because the set is again tested on 100 correct instances and 50 incorrect instances (according to the decision tree).   **NO change in the result and is same as TC 1a.** | |
| TC 1c.) | 1. Change the testing option to “Percentage split”. Specify 66% so that the data would be considered as a training set for 66%. | The above doc is the test run for the specified input. | **The percentage split is done automatically** on a Dataset. This testing option determines the training records and the test records and separates them.  Then the percentage accuracy is calculated on the testing records set.  In our case, this testing option has split the 150 instances into 99 training records and 51 test records set. The accuracy is calculated on 51 records where there were 19 incorrect instances for IRIS-VERISCOLOR.  **The classification accuracy is even worse** as the algorithm had to calculate the regression (OR) the classification and hence percentage split (automatically) is not a good option to determine the accuracy of the DS. | |
| TC 1d.) | Test IRIS Dataset with explicitly supplied test set.   1. Create a dataset named IRIS.TRAINING   Actual IRIS dataset is split into 2 datasets.   * Training set – 99 records (66% of the actual file) * Test set – 21 records (33% of the actual file).  1. Change the testing option to “Supplied test set”. 2. Supply the IRIS.TRAINING. 3. Compare the results. | Below are the 2 input Datasets created as a part of this test case and supply both (one by one) in the “Supplied test set” option.  **Training data**: consists of 66% of the records from the original IRIS DS.    **Results:** | **Classification accuracy remains the same with 66.67 % (for number of correct instances) with respect to Test and training sets.**  33 instances are marked as correct instances with accuracy of 66.66% and 17 are marked with incorrect instances with 33.334% accuracy.  Still the IRIS dataset though split into 2 halves, training and test and when these are made to run one by one via Decision stump, the accuracy hasn’t changed. | |
| TC 1e.) | 1. Create a test file named IRIS.MISSING to execute. 2. Edit the dataset IRIS.MISSING and keep spaces/blank for Setosa class. 3. Repeat step b) in the next iteration for Virginica and versicolor. 4. Set cross validation (by default) folds = 10. | **Setosa with Missing information:**    Result of the above Dataset:    **Versicolor with Missing values:**    **Result of the above Dataset:**    **Virginica with missing values:**    **Result:** | Result Captured:    From the results, it is observed that, though missing values are introduced in Setosa, the accuracy does not differ because, the missing value is considered to be 0, and hence falls as a correct instance under Setosa.  While missing values in Versicolor run against DataStump algorithm, triggers a Classification between Setosa and Virginica and hence all the values in versicolor will be reported as Incorrect instances. This accuracy is not considered to be accurate.  Coming to Virginica, the classification changes back to Setosa and versicolor and hence the accuracy is based NOT on Virginica thus resulting in false accuracy.  If values of the attributes are predicted before the run and made changes accordingly so as to report the incorrect instances, the accuracy is still being poor as the classification is based on only 1 parameter. | |
| TC 1f.) | Test IRIS edited test Dataset (IRIS.MISSING) against testing option “Use training set”   1. Select “Use training set” option. | **Input:**    Result obtained having slightly more accuracy: | Missing value against a training set -🡪 Accuracy is slightly efficient than the accuracy obtained from the missing value on Cross fold validation.  Accuracy --------🡪 67.33 % (this is a better method of obtaining accuracy comparatively, but missing values result in a wrong conclusion sometimes when analyzing the data. | |
| TC 1g.) | Test IRIS edited test Dataset (IRIS.MISSING) against testing option “Percentage split”   1. Open IRIS.MISSING via pre-process. 2. Select “Percentage split” option and specify 66% (which will be taken as 66% of the data taken as training set). | Same i/p as TC 1f.) used here  Results obtained: | The accuracy obtained for the missing values using Percentage split option is even worse than any other accuracy.  Reason:  Only 34% of the data is able to make to the accuracy decision. In our case, 51 instances out of which some are with missing values will definitely not yield a good accuracy.  Percentage: 60.783% | |
| TC 1h.) | Test IRIS edited test Dataset (IRIS.MISSING) against testing option “Supplied test set”   1. Select “Supplied test set” option under test options. 2. Supply the test version of the dataset – IRIS.TRAINING. 3. Run the algorithm. | Same i/p as TC 1f.) used here  Results obtained: | Classification accuracy is same as obtained when tested with a training dataset.  Thus concluding IRIS dataset does not work good with DecisionStump algorithm as the accuracy remains between 66% - 67% and not above that even during the whole set tested on training data. | |
| TC 1i.) | Introduce some noise in the IRIS dataset by changing the class parameter from Setosa to Veriscolor. | Changed 2 IRIS\_setosa records to IRIS-veriscolor and IRIS-virginica.  **Results:** | It is seen that when a noise is introduced, the confusion matrix changes.  Iris-Veriscolor reports 1 incorrect instance and IRIS-virginica reports 1 incorrect instance.  Performance suffers because there are only 97 correct instances and dropped by 2 instances (which should have been correct).  Accuracy : 64% | |
|  |  |  |  | |
|  |  |  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function to be Tested:**  **Repeat the test cases for Decision stump algorithm (from TC 1a – 1i) by changing its parameters**.  **Parameter involved**: Only “Debug” option (Debug = T/F) since this is a single ruler algorithm. | | | | |
| **Objectives/Conditions:**  Same as TC 1a. – 1i | | | | **Test cases #**  **TC:**  **2a – 2i** |
| **Test data library:**  Weka-3-6\data | | | |
| **TC-ID** | **Action/Input data** | **Results : Classification accuracy** | **Inference** | |
| TC 2a- 2i.) | Repeat the test from TC 1a – 1i by changing the Debug’s parameter – from default False to True  Record the runs for both conditions. | This case the results obtained from 1a – 1i and from 2a – 2i are the same. | Accuracy and the results are the same even if the parameter is changed for Datastump.  This parameter is just about setting the mode Debug to “true” to show information on the console. | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function to be Tested:**  **Test IRIS dataset under SimpleCART Tree algorithm** and record the results for the below test cases without changing its parameters.  minNumObj = 2 (default), usePrune = True (default) | | | | |
| **Objectives/Conditions:**   1. Use the same test datasets that created from test case TC 1. | | | | **Test cases #**  **TC:**  **3a – 3i** |
| **Test data library:**  Weka-3-6\data | | | |
| **TC-ID** | **Action/Input data** | **Results : Classification accuracy** | **Inference** | |
| TC 3a.) | 1. Set cross validation (by default) folds = 10. 2. Set the algorithm SimpleCART under tree classifier. | Results obtained on running original IRIS dataset under SimpleCART | With default parameters set, IRIS dataset gives a higher accuracy rate as there are 143 correct instances and only 7 incorrect instances where instances 80, 92, 109 and 123 (for Virginica) and 15, 73, 119 are for (Veriscolor).  But then IRIS dataset holds good for SimpleCART algorithm which yields 95.33 percentage of accuracy. | |
| TC 3b.) | 1. Change the testing option to “Training set” |  | **Number of leave nodes: 5**  **Size of the tree : 9**  **Yields high accuracy: 98 %** (49 and 48 correct instances for versicolor and virginica respectively).  IRIS dataset holds good for SimpleCART algorithm | |
| TC 3c.) | 1. Change the testing option to “Percentage split”. Specify 66% so that the data would be considered as a training set for 66%. | This will not yield a good accuracy rate when compared with others using training set and test set. | **Accuracy rate : 96.0784 %** | |
| TC 3d.) | Actual IRIS dataset is split into 2 datasets.   * Training set – 99 records (66% of the actual file) * Test set – 21 records (33% of the actual file).  1. Change the testing option to “Supplied test set”. 2. Supply the IRIS.TRAINING dataset. | All 33 instances for each class attribute is tested against this option. | **Accuracy rate : 97%**  1 incorrect Veriscolor instance along with 2 incorrect Virginica instances.  Setosa – being reported perfectly fine as per the classification (petal length < 2.45) and all the values in the DS for petal length satisfies this condition. | |
| TC 3e.) | Test IRIS Dataset by editing the input file and by adding some missing values to the set.   1. Create a test file named IRIS.MISSING to execute. 2. Edit the dataset IRIS.MISSING and keep spaces/blank for some of the records. 3. Set cross validation (by default) folds = 10. | Use the same Missing values input file (TC 1f) for this run.  66.67% used on IRIS-setosa and versicolor and the remaining percentage is used on Iris-virginica. | **No.of leaf nodes : 6**  **Size of the tree : 11**  If petal length < 2.45 (then the records belong to sertosa class)  If not, then algorithm checks if Petal length < 4.75 and if yes, it checks for the Petal width and based on the results writes If It is either versicolor or virginica.  If Petal length > 4.75, checks for petal width < 1.75 and if yes, Petal length < 4.95 write either versicolor or virginica.  **Accuracy** : 91.33 % | |
| TC 3f.) | Test IRIS edited test Dataset (IRIS.MISSING) against testing option “Use training set”   1. Select “Use training set” option. 2. Run the algorithm. | **Attributes used:**  Petal Length, Petal width for determining which classes they belong. | petallength < 2.45: Iris-setosa(50.0/0.0)  petallength >= 2.45  | petallength < 4.75  | | petalwidth < 1.65: Iris-versicolor(43.97/0.0)  | | petalwidth >= 1.65: Iris-virginica(1.0/0.02)  | petallength >= 4.75  | | petalwidth < 1.75  | | | petallength < 4.95: Iris-versicolor(3.0/0.0)  | | | petallength >= 4.95: Iris-virginica(4.33/2.0)  | | petalwidth >= 1.75: Iris-virginica(44.66/1.0)  All the real numbers are rounded off and all the correct instances are added up to obtain a particular class attribute.  **Accuracy obtained**: 98% with 147 correct instances all together. | |
| TC 3g.) | Cross fold validation – multiple iterations.  Unpruned = True (full trees) | **When cross fold = 10**  No of leaves : 6  Size of the tree :11  Numeric incorrect instances : 13    **When cross fold = 15**  Same number of leaves and tree size.  **When cross fold = 20**  Same number of leaves and tree size. | **Accuracy achieved : 91.33% for 10 cross**  This is not ideal as it is based on only 1 iteration where versicolor has 44 instances and remaining 6 instances will have to be incorrect. – This is a prediction.  **Accuracy achieved : 92.66% for 15 cross**  Number of correct instances are 2 up than the ones in 10 cross and hence the resulting accuracy **is slightly high.**  **Accuracy achieved : 92% for 20 cross**  Number of correct instances is 1 up than the ones in 10 cross and hence the resulting accuracy is slightly more. | |
| TC 3h.) | 1. Select “Supplied test set” option under test options. 2. Supply the test version of the dataset – IRIS.TRAINING. 3. Run the algorithm. | Since there are minimum number of data in the test set that is supplied, average accuracy rate will always be greater than the normal accuracy rate that was run on 10 cross fold. | **Accuracy achieved : 98%**  As there are 97 correct instances specified in a group of 99 instances together, the accuracy rate would be high. | |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function to be Tested:**  **Test IRIS dataset under SimpleCART Tree algorithm** and record the results for the below test cases by changing its parameters.  **Change the below parameters.**   * usePrune – Set to False (default – yes) * minNumObj – Number of leaves per instance (default – 2) : Set this to many iterations and observe.   Repeat TC 3a – 3i and record the results. | | | | |
| **Objectives/Conditions:**   1. Use the same test datasets that created from test case TC 1. | | | | **Test cases #**  **TC:**  **4a – 4i** |
| **Test data library:**  Weka-3-6\data | | | |
| **TC-ID** | **Action/Input data** | **Results : Classification accuracy** | **Inference** | |
| TC 4a.) | 1. Set cross validation (by default) folds = 10. 2. Set the algorithm SimpleCART under tree classifier.   All test cases are performed with usePrune = True | **When minNumObj = 5**  No. of leaf nodes : 4  Size of the tree: 7 (because they are pruned trees now).  **When minNumObj = 10**  No.of leaf nodes = 3  Size of tree = 5  **When minNumObj = 15**  No.of leaf nodes = 3  Size of tree = 5  **Results captured:** | **When observations at terminal node = 5**  This is because, decrease in leaf nodes results in decrease in size of the tree which in turn **decreases** the accuracy of the classification more in depth.  Accuracy obtained : 90.66%  143 correct instances and 7 are incorrect ones.  Not a good idea for IRIS dataset to be run on 10 cross fold under SimpleCART algos.  **When observations at terminal node = 10**  It seems to be like when the nodes are further reduced from 4 to 3, the accuracy is still maintained without any change.  Accuracy obtained : 90.667%  **When observations at terminal node = 15**  We could notice that the number of leaves did not differ. So from a certain terminal node, the number of leaf nodes and size of tree doesn’t differ. But number of correct instances increases (since 15 nodes) thus increasing the accuracy in total.  Accuracy obtained: 93.33%  Thus we can conclude that when number of observations at terminal node starts from 13, the size of the tree or the number of leaf nodes DO NOT differ and hence the accuracy maintained is constant.  Hence IRIS dataset with simple cart default value gives more accuracy than changing the parameters. | |
| TC 4b.) | Test IRIS Dataset on Training set option under SimpleCART.   1. Change the testing option to “Training set” 2. Run the algorithm.   All test cases are performed with usePrune = True | **When minNumObj = 3,4,5**  Num of Leaf nodes = 4  Size of the tree = 7  **When minNumObj = 6,7,10,15,20 etc**  Num of leaf nodes = 3  Size of the tree = 5 | **When observations at terminal node = 3,4,5**  Accuracy obtained = 97.33%  **When observations at terminal node = 6,7 etc..**  Accuracy obtained = 96% and its constant throughout from minNumObj 6.  Still the accuracy is not that efficient compared with the default parameters of SimpleCART from TC (3b).  But the accuracy is reduced when the number of observations goes down from 5 to 6 by one as the number of correct instances reduces though the tree structure is maintained the same. | |
| TC 4c.) | 1. Run the IRIS dataset under SimpleCART algorithm by changing the below parameters   minNumObj = 2 and usePrune = False  minNumObj = 5 and usePrune = False  minNumObj = 13 and usePrune = False  minNumObj = 15 and usePrune = False  minNumObj = 20 and usePrune = False | Tested with Cross validation 10.  **minNumObj = 2 and usePrune = False**  Num of leaf nodes = 6  Size of tree = 11  **Accuracy obtained = 94.66%**  **minNumObj = 5 and usePrune = False**  Num of leaf nodes = 4  Size of the tree = 7  **Accuracy obtained = 95.33%**  **minNumObj = 6 and usePrune = false**  Num of leaf nodes = 4  Size of the tree = 7  **Accuracy obtained = 94.66%**  **minNumObj = 7 and thereforth, usePrune = False**  Num of leaf nodes = 3  Size of the tree = 5  **Accuracy obtained = 94.66%** | Here, maximum accuracy is obtained when minNumObj = 5 and usePrune = False  Though the number of leaf nodes are reduced from 6 to 4 to 3, maximum efficiency is gained when number of leaf nodes = 4 (Correct instances are slightly 1 up)  **Results captured:** | |
| TC 4d.) | Actual IRIS dataset is split into 2 datasets.   * Training set – 99 records (66% of the actual file) * Test set – 21 records (33% of the actual file).  1. Change the testing option to “Supplied test set”. 2. Supply the IRIS.TRAINING dataset. 3. Run the algorithm. | **When minNumObj = 5 and Useprune = False**  No, of leaf nodes = 4  Size of tree = 7  Accuracy obtained = 97%  **When minNumObj = 6 and Useprune = False**  No, of leaf nodes = 4  Size of tree = 7  Accuracy obtained = 97%  **When minNumObj = 7 and Useprune = False**  No, of leaf nodes = 3  Size of tree = 5  Accuracy obtained = 96%  **When minNumObj = 8 and Useprune = False**  No, of leaf nodes = 3  Size of tree = 5  Accuracy obtained = 96%  **When minNumObj = 10 and Useprune = False**  No, of leaf nodes = 3  Size of tree = 5  Accuracy obtained = 96% | When IRIS.TRAINING dataset is run with minNumObj = 2,5,6 the resulting accuracy maintained is constant thereby resulting in high accuracy as the number of leaf nodes and the size of the tree is not changed when tested with a supplied test version.  When minNumObj = 7,8 and so on, the resulting accuracy is dropped by 1% thereby resulting in decrease in the number of leaf nodes and the size of the tree.  But the accuracy remains constant from minNumObj = 7 and thereforth.  **Results obtained:**    **When compared with training set option, this test case when minNumObj = 6 gives the MAX ACCURACY.** | |
| TC 4e.) | Test IRIS Dataset by editing the input file and by adding some missing values to the set.   1. Edit the dataset IRIS.MISSING and keep spaces/blank for some of the records. 2. Open the file IRIS.MISSING via pre-process. 3. Set cross validation (by default) folds = 10. 4. Run the algorithm. | Same input file is used from TC 1f.  Default parameters tested with 10 cross fold validation gives accuracy of 91.33%  **When minNumObj = 5, 6, usePrune = T/F**  No. of leaf nodes = 4  Size of tree = 7  Accuracy : 93.33%  **When minNumObj = 7,8,10, 15 etc Useprune = T/F**  No of leaf nodes = 4  Size of tree = 7  Accuracy = 92.66% | We could see that when minNumObj = 7,8, and thereforth – Accuracy suffers a lot though the number of leaf nodes do not differ.  IRIS dataset with missing values though they are pruned trees (which will reduce the size of the tree) Accuracy is badly hit.  **Results captured:** | |
| TC 4f.) | Test IRIS edited test Dataset (IRIS.MISSING) against testing option “Use training set”   1. Select “Use training set” option. 2. Run the algorithm. | Same i/p file is used from TC 1f.  Default parameters tested with TRAINING set option gives accuracy of 98%  **When minNumObj = 4,5, 6, usePrune = T/F**  No. of leaf nodes = 4  Size of tree = 7  Accuracy : 96%  **When minNumObj = 7,8,10, 15 etc Useprune = T/F**  No of leaf nodes = 4  Size of tree = 7  Accuracy = 96% | We could see that when minNumObj = 2,3 maximum accuracy is obtained and when the nodes are increased from 4, accuracy is reduced by two percentage.  IRIS dataset with missing values though they are pruned trees (which will reduce the size of the tree) Accuracy is still maintained to be good when it comes to missing values after changing the parameters. | |
| TC 4g.) | Test IRIS edited test Dataset (IRIS.MISSING) against testing option “Supplied test set”   1. Select “Supplied test set” option under test options. 2. Supply the test version of the dataset – IRIS.MISSING. 3. Run the algorithm. | IRIS.MISSING    Default parameters tested with TEST set option gives accuracy of 98% 6 NODES and 11 – size of the tree  **When minNumObj = 4,5, 6, usePrune = T/F**  No. of leaf nodes = 4  Size of tree = 7  Accuracy : 96%  **When minNumObj = 7,8,10, 15 etc Useprune = T/F**  No of leaf nodes = 4  Size of tree = 7  Accuracy = 96% | We could see that when minNumObj = 2,3 maximum accuracy is obtained and when the nodes are increased from 4, accuracy is reduced by two percentage.  We can conclude that when missing values in IRIS dataset are tested against TEST option or against TRAINING set option, both yields the same percentage when minNumObj is greater than 4.  During initial default parameters, there is a slight 0.7% difference in accuracy from training and test set when minNumObj = 2 or 3. | |
| TC 4h.) | Introducing noise in IRIS dataset by changing IRIS-SETOSA to IRIS-VERISCOLOR for the first 3 records. | When IRIS dataset run against simplecart without any noise, the percentage produced is 95.33% (max percent obtained)  When introduced to noise without changing parameters, the percent obtained is 91.33 (Low performance gained)  When introduced to noise by changing some parameters minNumObj = 5 and unpruned = False, better performance is obtained. | Without noise – default parameters – Highest performance accuracy is achieved  With noise – changed parameters for minNumObj = 5 and pruned trees, performance accuracy is better achieved.  With noise and default parameters – Poor performance is achieved.  **Results obtained:** | |
| TC 4i.) | Tested IRIS dataset with resample filter after changing the parameters of SimpleCART | **Resample filter – Default values of Simplecart achieves 98% on Cross validation and 99.33% on Training set**  Resample filter – after changing SimpleCART parameters with minNumObj = 5, Accuracy obtained – 98%  Resample filter – after changing SimpleCART parameters with pruned trees (Unpruned = false) Accuracy obtained –96%  Resample filter – after changing SimpleCART parameters with minNumObj = 5 and pruned trees, Accuracy obtained – 98% | By observing the results, we can conclude that using resample filter against training set with changing the parameter Unpruned = False -🡪 yields high percentage of 99.33% which the same that was obtained on testing this against the default parameters under Training set option.  **Results obtained:** | |
|  |  |  |  | |
|  |  |  |  | |

**Analyzing VOTE Dataset**



**Attributes :** 17

**Instances for all the attributes (total) :** 435

**Class attribute : Instances :** Democrat (or) Public

**Type of the Dataset** : Nominal (Hence Mean, Median, SD and Variance do not apply for this Dataset).

Instances for each class attribute:

Democrat : 267 instances

Republican : 168 instances

**Class distribution** : 45.2% for democrat and 54.8 percent for republican.

**Count of each attributes:**

* handicapped-infants' { 'n - 236', 'y - 187'}
* water-project-cost-sharing' { 'n - 192', 'y - 195'}
* adoption-of-the-budget-resolution' { 'n – 171 ', 'y - 253'}
* physician-fee-freeze' { 'n - 247', 'y - 177'}
* el-salvador-aid' { 'n -208', 'y - 212'}
* religious-groups-in-schools' { 'n - 152', 'y - 272'}
* anti-satellite-test-ban' { 'n- 182', 'y - 239'}
* aid-to-nicaraguan-contras' { 'n 178', 'y - 242'}
* mx-missile' { 'n - 206', 'y -207'}
* immigration' { 'n- 212', 'y - 216'}
* synfuels-corporation-cutback' { 'n - 264', 'y - 150'}
* education-spending' { 'n – 233 ', 'y - 171'}
* superfund-right-to-sue' { 'n - 201', 'y - 209'}
* crime' { 'n – 170 ', 'y - 248'}
* duty-free-exports' { 'n - 233', 'y - 174'}
* export-administration-act-south-africa' { 'n - 62', 'y - 269'}
* Class' { 'democrat - 267', 'republican - 168'}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function to be Tested:**  **Test VOTE dataset under DecisionStump Tree algorithm** and record the results for the below test cases without changing its parameters.  DecisionStump with its default parameter – debug = False. | | | | |
| **Objectives/Conditions:**   1. Create a separate Dataset – VOTE.TRAINING which will have 66% of the original data. 2. Create 1 separate dataset in the data folder – VOTE.MISSING to satisfy test case with missing values. | | | | **Test cases #**  **TC:**  **5a – 5i** |
| **Test data library:**  Weka-3-6\data-Iris.arff, Weka-3-6\data-VOTE.Training.arff, and Weka-3-6\data-VOTE.missing.arff | | | |  |
| **TC-ID** | **Action/Input data** | **Results: Classification accuracy, Confusion matrix and Correct/incorrect instances.** | **Inference** | |
| TC 5a.) | 1. Set cross validation (by default) folds = 10. 2. Set the algorithm DecisionStump under tree classifier. 3. Set Training set option and cross verify the results obtained from Cross fold validation and training set option. | The above doc has the test result run for the input data specified. | In this case, the decision tree decides the class based on one attribute – Physician-fee-freeze.  If this attribute is ‘n’ then it belongs to democratic class else, republican.  When run against Cross validation:  Maximum accuracy is obtained: 95.33%  But surprisingly, when run the same dataset against Training set option, the same result is gained. (Percentage of accuracy : 95.33%)  **Possibilities:**  Since decision stump takes only 1 input and has 2 class, naturally the accuracy is maintained constant without any variations however the test options are changed. | |
| TC 5b.) | 1. Change the testing option to “Percentage split”. Specify 66% so that the data would be considered as a training set for 66%. | When Vote dataset without changing Decision stump parameters run against percentage split | Percentage of accuracy is 96.62% only for 148 instances (as this option tests the results based on the reminder of the training set)  Performance is poor when compared to that of the testing done against training set where (435 instances) accuracy obtained is 95.63%  **Conclusion:**  Percentage split is not a better option when analyzing data which are having many instances like this VOTE dataset. | |
| TC 5c.) | 1. Create a Dataset VOTE.TRAINING.   Actual VOTE dataset is split into 2 datasets.   * Training set – 99 records (66% of the actual file) * Test set – 21 records (33% of the actual file).   Test set will not be used (as this is the same result which will be produced by using percentage split)   1. Change the testing option to “Supplied test set”. 2. Supply VOTE.TRAINING in the TEST option. 3. Compare the results. | **Training data**: consists of 66% of the records from the original Vote DS.    **Results:**    **What happens when another training dataset(80% of the original data) made to run across Cross validation and training set and Supplied test set?**  **Input dataset – 80% of the data is manually clubbed)**    Maximum accuracy is obtained in all the 3 cases:  Accuracy : 96.8% ~ 97% | Accuracy obtained when running the TRAINING dataset under SUPPLY TEST option is slightly high when compared with test options.  Accuracy : 96.5% (for a total of 287 instances)  Hence for 66% of the original data when tested, gives only a slightly high percentage accuracy than the accuracy obtained with 100% data.  Accuracy obtained (original data) – 95.33%  **80% of the data tested:**  Maximum accuracy is obtained. Thus we can conclude that VOTE dataset under decision stump tree algorithm yields maximum accuracy when tested on 80% of the actual original DS.  **Results captured for the highest performance accuracy:** | |
| TC 5d.) | 1. VOTE dataset by default has Missing values. 2. When for first 2 records, under Physician-fee-freeze, missing values is generated manually apart from the existing missing values. 3. Set cross validation (by default) folds = 10. 4. Report the results for multiple iteration of cross validation. | Multiple iterations for cross validation = yields the same accuracy  Cross fold validation 10  Accuracy obtained : 95.1%  Cross fold validation 20  Accuracy obtained :  95.1%  Cross fold validation :11  Accuracy obtained :  95.1%  Tested under training set:  Accuracy obtained 95.6% | **Result Captured:**    From the results, it is observed that, though missing values are introduced in Democratic class, the percentage accuracy does not differ for multiple iterations of cross fold validation as there are only 2 conditions to be checked.  When tested against training set, the percentage of accuracy is changing but not much of a difference.  **Conclusion:**  Hence accuracy obtained on VOTE dataset doesn’t change for multiple iterations of cross fold and is the same when tested against Training set option or the supplied test option.  Max accuracy gained : 95.1 – 95.6% | |
| TC 5e.) | Test IRIS edited test Dataset (VOTE.MISSING) against testing option “Percentage split”   1. Select “Percentage split” option and specify 66% (which will be 34% of the data tested on reminder). | Introduce 2 missing values in the first 2 records for physician-fee-freeze and record the results. | Accuracy obtained : 96% (on 34% of the data) which is a poor performance as (95.33% is obtained on the original set of data 100% of the data)  **Result captured:**    **Conclusion:**  Percentage of accuracy cannot be predicted correctly by using the percentage split option for VOTE dataset. | |
| TC 5f.) | Introduce some noise in the VOTE dataset by changing the class parameter from Democrat to Republic for the first 3 records and note the differences if any. | Run against Cross fold 10 validation:  Accuracy : 95%  \*Mulitple iterations are going to give the same accuracy.  Training set : also gives 95 % accuracy. | **Conclusion:**  Even when noise is introduced on Datastump algorithm, the percentage of accuracy remains constant when tested using all test options. | |
|  |  |  |  | |
|  |  |  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function to be Tested:**  **Repeat the test cases for Decision stump algorithm (from TC 5a – 5f) by changing its parameters**.  **Parameter involved**: Only “Debug” option (Debug = T/F) since this is a single ruler algorithm. | | | | |
| **Objectives/Conditions:**  Same as TC 5a. – 5f | | | | **Test cases #**  **TC:**  **6a – 6f** |
| **Test data library:**  Weka-3-6\data\vote.arff | | | |
| **TC-ID** | **Action/Input data** | **Results : Classification accuracy** | **Inference** | |
| TC 6a-6f.) | Repeat the test from TC 5a – 5f by changing the Debug’s parameter – from default False to True  Record the runs for both conditions. | This case the results obtained from 5a – 5f and from 6a – 6f are the same. | Accuracy and the results are the same even if the parameter is changed for Datastump.  This parameter is just about setting the mode Debug to “true” to show information on the console. | |
| **Function to be Tested:**  **Test VOTE dataset under SimpleCART Tree algorithm** and record the results for the below test cases with and without changing its parameters.  minNumObj = 2 (default), usePrune = True (default) | | | | |
| **Objectives/Conditions:**   1. Use the same test datasets that created from test case TC 5. | | | | **Test cases #**  **TC:**  **7a – 7f** |
| **Test data library:**  Weka-3-6\data | | | |
| **TC-ID** | **Action/Input data** | **Results : Classification accuracy** | **Inference** | |
| TC 7a.) | Test VOTE dataset with the following:  With default parameters  By changing minNumObj and usePrune = False  Under cross fold validation  What happens under multiple cross fold validation. | **With default parameters under multiple cross fold iterations**:  Accuracy is 95.4 (initially) and boosted to 96.55% when cross folds = 35    **When parameters of Simple cart are changed:**  When cross fold = 10, minNumObj = 5, accuracy obtained = 95.63%  When cross fold = 30, minNumObj = 5 highest percentage of accuracy is obtained.- 96.02%  When cross fold = 10, minNumObj =5 and Unpruned = False, Accuracy is 95.62  When cross fold = 30, minNumObj = 5 and Unpruned = false highest percentage of accuracy is obtained.- 96.32% | When observed, from cross fold 10, the percentage of accuracy (95.4%) is increasing slowly to 96.32% and finally reaching 96.55% when cross fold = 35.  When cross fold = 36, the performance again starts to slowly decrease.  Hence to achieve a maximum percentage of accuracy when using cross fold testing option, we can assume that 35 folds will give me the max percentage of accuracy.  Vote dataset holds good when Cross fold = 35.  **Results captured for changed parameters:**    **Conclusion:**  Thus VOTE dataset for multiple cross fold validation though remains more or less the same, highest performance accuracy is achieved when number of folds is 5 and for reduced size of the tree.  Pruned trees – gives slightly a higher notch in determining the percentage of accuracy of the set | |
| TC 7b.) | Test VOTE dataset with the following:  With default parameters  By changing minNumObj and usePrune = False  By giving multiple iterations of minNumObj  Use training set option to test the data. | **Using default parameters and Useprune = False**  Accuracy obtained:98.39%  No.of leaves : 51 nodes  Size of the tree :101  **By changing parameters minNumObj = 5 and usePrune = false**  Accuracy obtained: 96.7%  No.of leaf nodes: 28  Size of the tree : 55 | Result captured:    From the above, we can conclude that higher percentage of accuracy is obtained when there are more number of leaf nodes and size of tree is huge)  **By having multiple iterations of minNumObj = 6,7,8,10 and usePrune = False**  Accuracy obtained: 96.7%  No.of leaf nodes: 23, 22 etc  Size of the tree: 45, 43 etc | |
| TC 7c.) | 1. Change the testing option to “Percentage split”. Specify 66% so that the data would be considered as a training set for 66%. | Yields a percentage of accuracy – 96.6%  Correct instances: 82 + 61 = 143  When minNumObj = 5 and usePrune = False,  Accuracy remains the same. | Accuracy rate: 96.62% is less than the percentage received on a training or cross validation 10 folds set.  But when run with minNumObj = 10 and usePrune = false, maximum percentage of accuracy is achieved. This means less incorrect instances reported.  **Results captured:** | |
| TC 7d.) | Test VOTE Dataset by editing the input file and by adding some missing values to the set.   1. Edit the dataset VOTE and keep spaces/blank for the first 2 records on physician-fee-freeze some of the records. 2. Set cross validation (by default) folds = 10. | Missing values are handled automatically by SimpleCart. | In pHysician-fee-freeze (which is set as an attribute as a part of condition based on which the algorithm decides the class), there are 11 instances where the fields are spaces (already in the original DS).  Accuracy obtained:  95.4%  After introducing 3 more fields to spaces,  Accuracy obtained :  95% (not much of a difference)  **Conclusion:**  Percentage of accuracy doesn’t varies much when missing records are introduced in the VOTE dataset and run across SimpleCART algorithm (testing option-10 Cross validation). | |
| TC 7e.) | Test VOTE edited test Dataset against testing option “Use training set” | **Default parameters:**  Accuracy obtained: 97.2%  No.Leaf nodes : 6  Size of the tree :11    **While using test option set;**  Accuracy is maintained and not changed even if the parameters of Simplecart are changed. | Even if the parameters are changed and tested, the accuracy gained is constant both in training set and in test set data.  **Conclusion:**  SimpleCART though having missing values in the dataset uses surrogate method to split the data and hence the missing records are taken care automatically. They are skipped and accuracy is got which is close to the one that is obtained on the original set.  Accuracy is constant in both training and test set (with or without changes). = 97.2% | |
| 7f.) | Test VOTE edited dataset and introduce some noise. | Accuracy obtained when using cross validation. 95.62%  Training set used:  Accuracy obtained – 96.72% | Even if noise is introduced, training set gives the maximum percentage of accuracy when SimpleCART algorithm is used. | |
|  |  |  |  | |

**ANALYZE LABOR DS:**

**Attributes :** 17

**Instances for all the attributes (total) :** 57

**Class attribute : Instances :** bad (or) good

**Type of the Dataset** : Nominal (Hence Mean, Median, SD and Variance do not apply for this Dataset) and Numeric.

**Instances for each class attribute:**

Bad : 20

Good :37

Mean, Median, Min, max and Std Deviation are shown in the below doc for each attribute including class attribute

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function to be Tested:**  **Test LABOR dataset under Decision Stump Tree algorithm** and record the results for the below test cases with and without changing its parameters.  Debug = False | | | | |
| **Objectives/Conditions:**   1. Create a new DS with 66% of data from the original set. | | | | **Test cases #**  **TC:**  **8a – 8f** |
| **Test data library:**  Weka-3-6\data\labor | | | |  |
| **TC-ID** | **Action/Input data** | **Results : Classification accuracy** | **Inference** | |
| 8a.) | 1. Set cross validation (by default) folds = 10. 2. Set the algorithm DecisionStump under tree classifier. 3. Set Training set option and cross verify the results obtained from Cross fold validation and training set option. | Percentage of accuracy (80.70%) and the performance suffers badly during 10/20/30 cross fold validation.  Percentage of accuracy is achieved at a max rate when Decision stump is made to run on Training set option.  **Results captured:** | In this case, the decision tree decides the class based on one attribute – Pension  If this attribute is = ‘none’ then it belongs to ‘bad’ class else, ‘good’ class.  **When run against Cross validation:**  Maximum accuracy is obtained: 80.72% (performance suffers and remains constant for all folds of validation – 20,30,40 etc)  But surprisingly, when run the same dataset against Training set option , high performance is gained and reached max % of accuracy.  Accuracy obtained : 84.2% (which is still poor performance but a better one when tested with cross validation.  **Possibilities:**  Decision stump on LABOR DS doesn’t have any incorrect instances for ‘bad’ class and hence the performance seems better thus increasing percentage of accuracy. | |
| 8b. | When LABOR DS is made to run against percentage split under Decision tree algorithm. | When percentage split is 66%, 34% of the data is tested.  Accuracy obtained – 84.2% | What we could observe is, percentage of accuracy got on complete training set and the test set are equal and no difference as there is the same percentage of number of incorrect instances.  But when the percentage split is 80 and 20, accuracy suffers. | |
| 8c. | When Labor DS is edited and introduced any value which will not satisfy the condition, then the incorrect instances will be increasing accordingly  How Cross validation reacts  How training set reacts on **Missing values**  Multiple iterations of cross validation | 10/20 cross fold validation gives an accuracy of 80.72%  But 30, 40 cross fold validation gives an accuracy of 84.02%  **Training set:**  Missing values on training set yields a high percentage of 84.21% | Percentage is increased in the above screenshot, because missing value introduced on an incorrect instance leads to decrease in the number of incorrect instances thus increasing the percentage of accuracy.  Cross fold validation – more number of folds increases the percentage of accuracy and reduces the number of instances.  **Training set:**  Labor DS – training set is equal to 30/40 folds of cross validation. | |
| 8d. | When noise is introduced in Labor dataset. | If a noise is introduced,  For a correct instance, percentage is decreased.  For an incorrect instance already marked, percentage is increased. | Thus increase in percentage of accuracy is achieved in noise (if an incorrect instance is marked/tagged to a correct class attribute).  The percentage remains constant when run on Cross validation and on training set option though noise is introduced. | |
|  |  |  |  | |
|  |  |  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function to be Tested:**  **Test LABOR dataset under Decision Stump Tree algorithm** and record the results for the below test cases with changing parameters.  Debug = True | | | | |
| **Objectives/Conditions:**  Repeat all the test cases from 8a to 8d | | | | **Test cases #**  **TC:**  **9a – 9d** |
| **Test data library:**  Weka-3-6\data\labor | | | |
| **TC-ID** | **Action/Input data** | **Results : Classification accuracy** | **Inference** | |
| 9a. to 9d) | Repeat all 8a to 8e. | Percentage of accuracy is same as obtained from 8a to 8e respectively. | Percentage is same because there is no parameter which will affect the performance as this is a DEBUG parameter. | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function to be Tested:**  **Test LABOR dataset under SimpleCART Tree algorithm** and record the results for the below test cases with and without changing its parameters.  minNumObj = 2 (default), usePrune = True (default) | | | | |
| **Objectives/Conditions:**  Run the Labor DS with default parameters and changed parameters. | | | | **Test cases #**  **TC:**  **10a – 10f** |
| **Test data library:**  Weka-3-6\data\labor | | | |
| **TC-ID** | **Action/Input data** | **Results : Classification accuracy** | **Inference** | |
| TC 10a.) | Test LABOR dataset with the following:  With default parameters  By changing minNumObj and usePrune = False  Under cross fold validation  What happens under multiple cross fold validation. | **With default parameters under multiple cross fold iterations**:  % of accuracy : 78.9% (suffers)  **20 cross fold :**  Percentage : 82.45  And constant from there on.  **Training set:**  Achieves max percentage of accuracy : 84.21%  **Percentage split:**  Though leading to inaccuracies – percentage is 89.4%  **When parameters of Simple cart are changed:**  When minNumObj = 5 with 20 cross fold, it gives the same result as what it gave for the default parameters.  When unpruned = False and cross validation = 10, % of accuracy suffers a lot | When observed, from cross fold 10, the percentage of accuracy (78.9%) is increasing slowly to 82.45% and finally reaching 82.45% when cross fold = 20/30/40 and remains constant.  Hence to achieve a maximum percentage of accuracy when using cross fold testing option, we can assume that 20 folds will give me the max percentage of accuracy.  LABOR DS holds good when cross fold = 20 and suffers a lot when cross fold = 10  **Results captured for changed parameters:**    **Conclusion:**  Pruned trees with cross fold 10 – also gives a lower percentage of accuracy | |
| TC 10b.) | Test VOTE dataset with the following:  With default parameters  By changing minNumObj and usePrune = False  By giving multiple iterations of minNumObj  Use training set option to test the data. | **Using default parameters and Useprune = False**  Accuracy : 84.21 (better than the one got on 10 cross validation)  **By changing parameters minNumObj = 5 and usePrune = false**  Accuracy obtained: 94.7%  No.of leaf nodes: 5  Size of the tree : 9 | **Result captured:**    From the above, we can conclude that Simplecart provides better accuracy in training set than in cross validation report. | |
| TC 10c.) | Test LABOR Dataset by editing the input file and by adding some missing values to the set.   1. Edit the dataset LABOR and keep spaces/blank for the first 2 records on WAGE-INCREASE-FIRST-YEAR some of the records. 2. Set cross validation (by default) folds = 10. | Missing values are handled automatically by SimpleCart. | In wage-increase-first-year (which is set as an attribute as a part of condition based on which the algorithm decides the class), there is only 1 instance where the field is blank. Introduced 5 instances to be blank.  Accuracy obtained:  94.7%(achieved on a training set)  After introducing 3 more fields to spaces,  Accuracy obtained :  94.72% (same)  **Conclusion:**  Percentage of accuracy doesn’t varies much when missing records are introduced in the VOTE dataset and run across SimpleCART algorithm (testing option-10 Cross validation). | |
| 10 d.) | Test LABOR edited dataset and introduce some noise. | Accuracy obtained when using TRAINING SET 94.7%  After introducing noise, by changing the class parameter,  Percentage of accuracy falls down as the incorrect instances have risen up. | Even if noise is introduced, training set gives a standard output but will definitely depend on the incorrect instances caused.  minNumObj = 5, pruned tree  No. of leaves : 5  Size of the tree : 9  Total instances : 57  Percentage of accuracy : 93% (which is less than what we got when the original DS was run on Training set earlier). | |
|  |  |  |  | |

**ANALYZING DIABETES DS:**

**Attributes :** 9

Instances : 768

Type : Numeric

Class attribute : tested\_negtive (count – 500) and tested\_positive(count - 268)

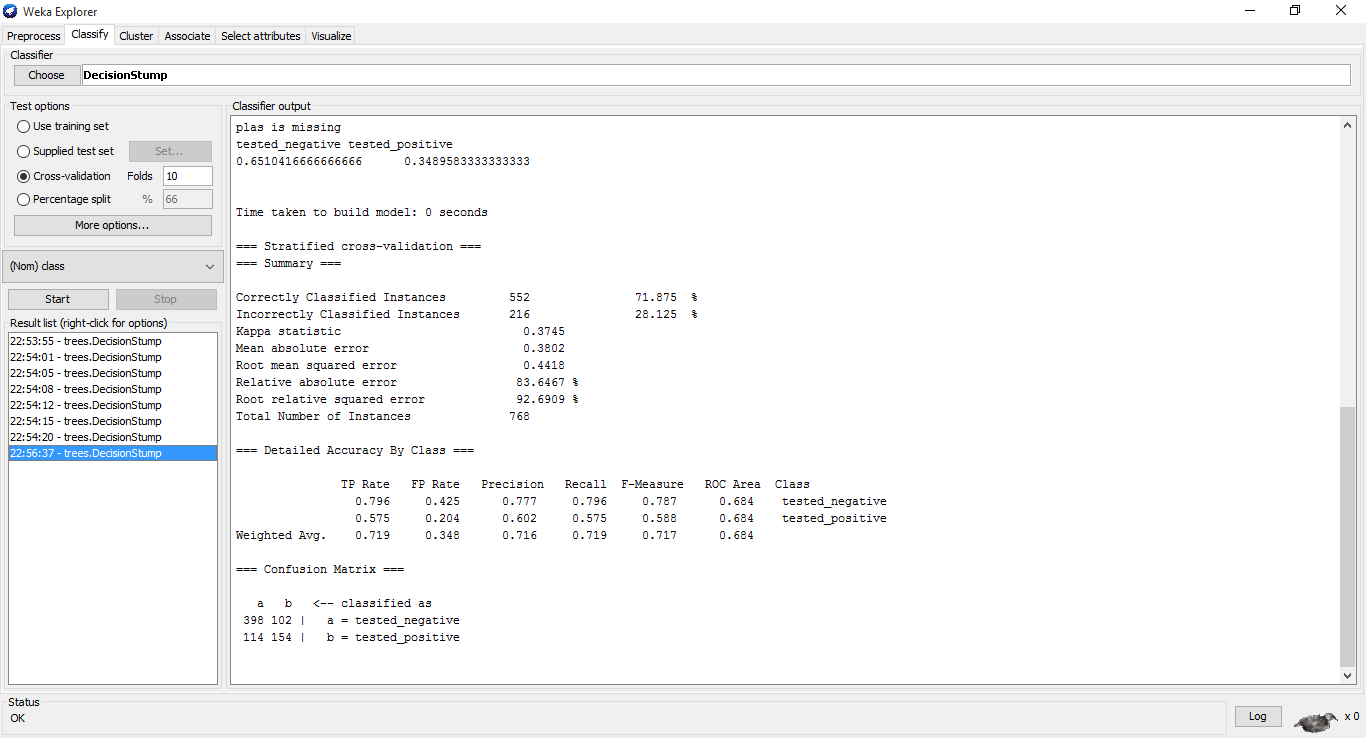
Min, Max, SD of all attributes:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name of attribute** | **Type** | **Min value** | **Max value** | **Mean** | **SD** |
|  |  |  |  |  |  |
| Preg | Numeric | 0 | 17 | 3.845 | 3.37 |
|  |  |  |  |  |  |
| Plas | Numeric | 0 | 199 | 120.895 | 31.973 |
|  |  |  |  |  |  |
| Pres | Numeric | 0 | 122 | 69.105 | 19.356 |
|  |  |  |  |  |  |
| Skin | Numeric | 0 | 99 | 20.536 | 15.952 |
|  |  |  |  |  |  |
| Insu | Numeric | 0 | 846 | 79.799 | 115.244 |
|  |  |  |  |  |  |
| Mass | Numeric | 0 | 67.1 | 31.993 | 7.884 |
|  |  |  |  |  |  |
| Pedi | Numeric | 0.078 | 2.42 | 0.472 | 0.331 |
|  |  |  |  |  |  |
| Age | Numeric | 21 | 81 | 33.241 | 11.76 |
|  |  |  |  |  |  |
| class | Nominal | N.A | N.a | N/A | N/A |
|  |  |  |  |  |  |

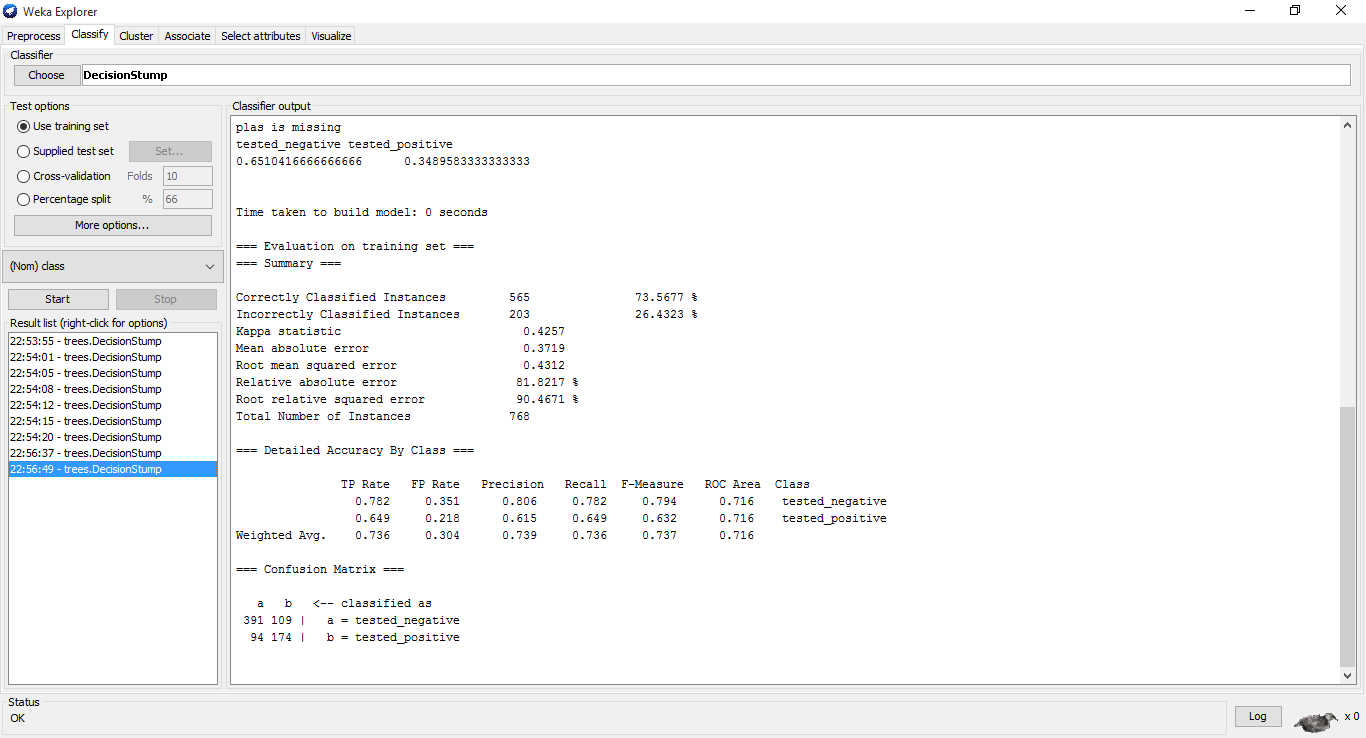
**Analyzing the data manually with previous experience of 3 datasets when run with DATASTUMP algorithm,**

**Decision stump run with 10 cross validation** – should yield a low percentage than all because when number of instances increases, chances of incorrect instances being increased thus resulting in Low % of accuracy when compared with the other 3 datasets.

**The same result should show up when done with multiple iterations of cross fold validations.**



**Tested on Training set:**



* Same accuracy is arrived as there are more number of incorrect instances which leads to poor % of accuracy.
* Splitting the data with the help of percentage split will give a high notch on the % of accuracy but since the data is taken random and only 34% of the data we cannot rely on the accuracy.



**Decision is made on the below algorithm:**

plas <= 127.5 : tested\_negative

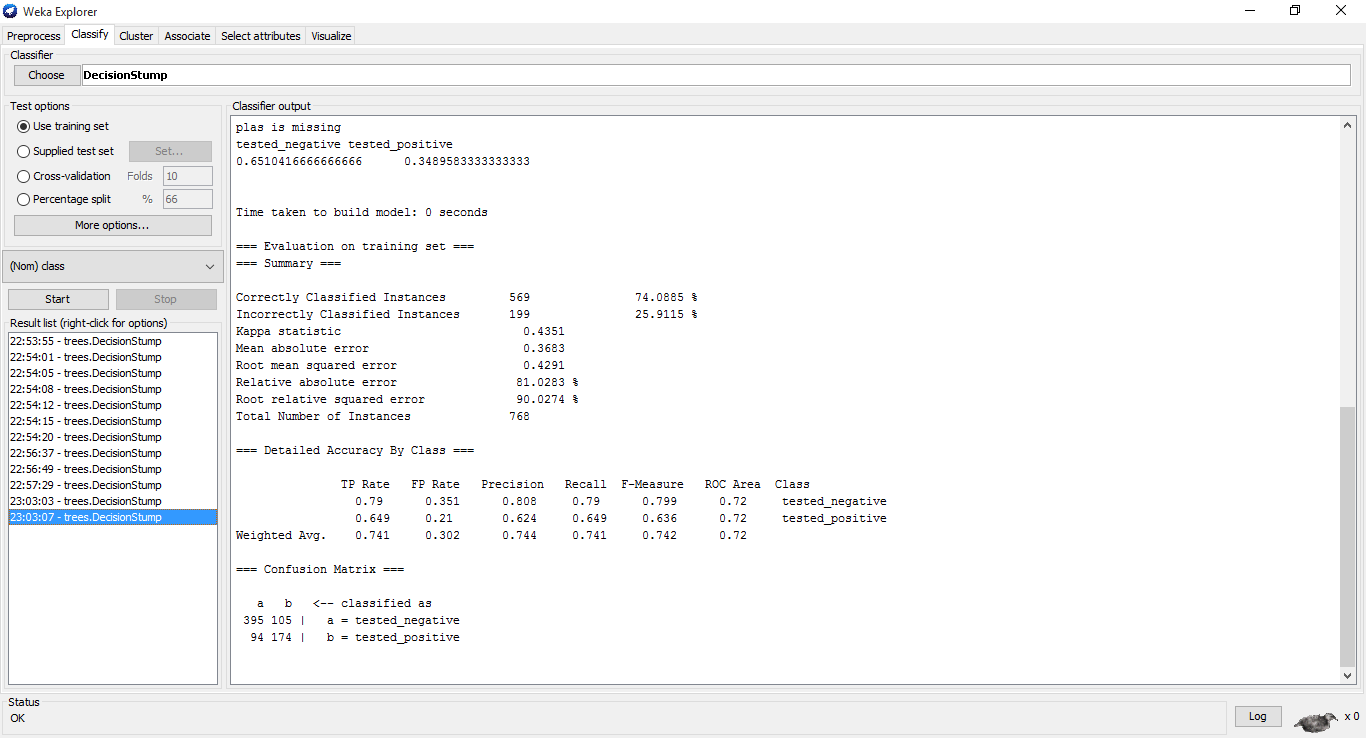
plas > 127.5 : tested\_positive

plas is missing : tested\_negative

**Made the below changes to DS:**

* Changed the value of 4 incorrect instances reported tested\_negative (which has to be ideally tested\_postive) edited these values to fit it with tested\_negative class as per the above algorithm and the number of incorrect instances reduced increasing slightly on the % of accuracy.

**Tested on Training set:**

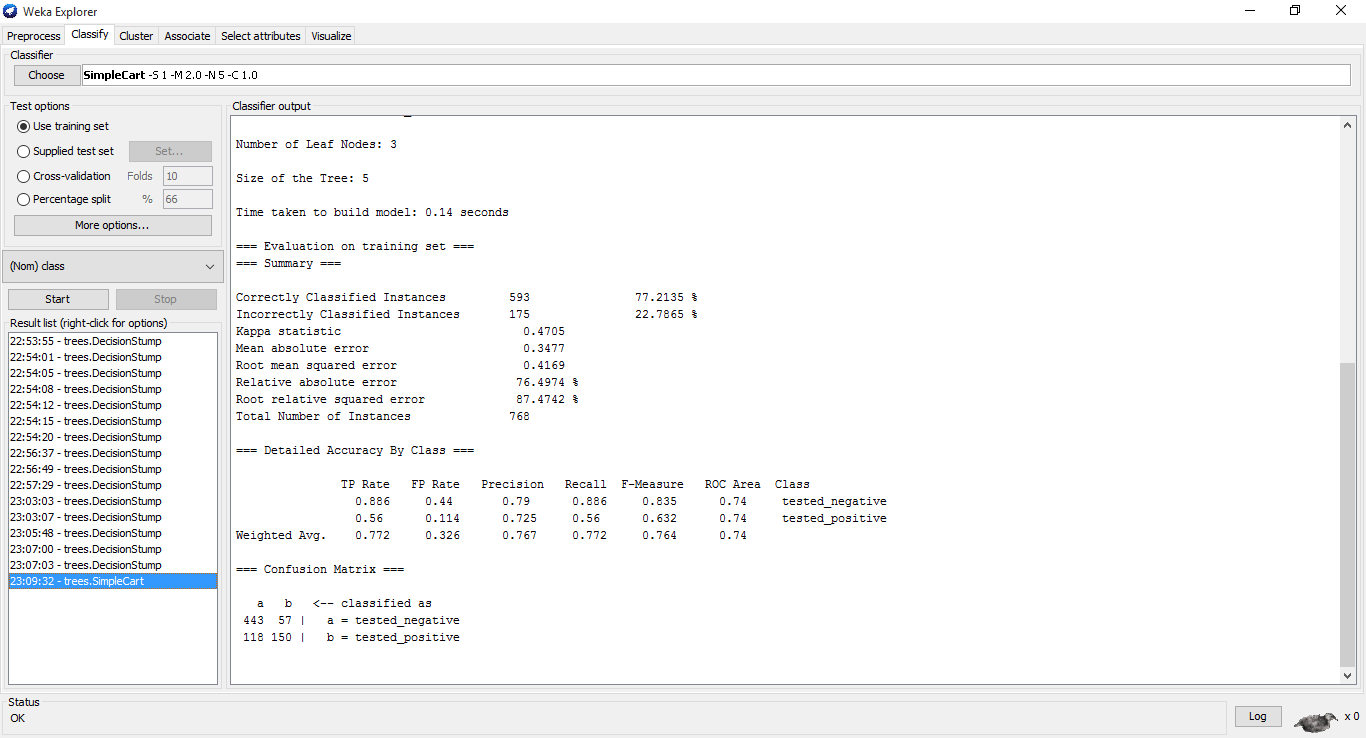


Introducing missing values also rely on Incorrect instances. Since there are many instances reported for this Dataset, and many instances are incorrectly specified, hence the % of accuracy will lower.

When introducing noise, to the Dataset, depends on how many instances are incorrectly reported for each of the class and then % of accuracy is determined. In our case, if the noise introduced increases the number of incorrect instances results in low efficiency.

**ANLAYZING DIABETES DS ON SIMPLECART**

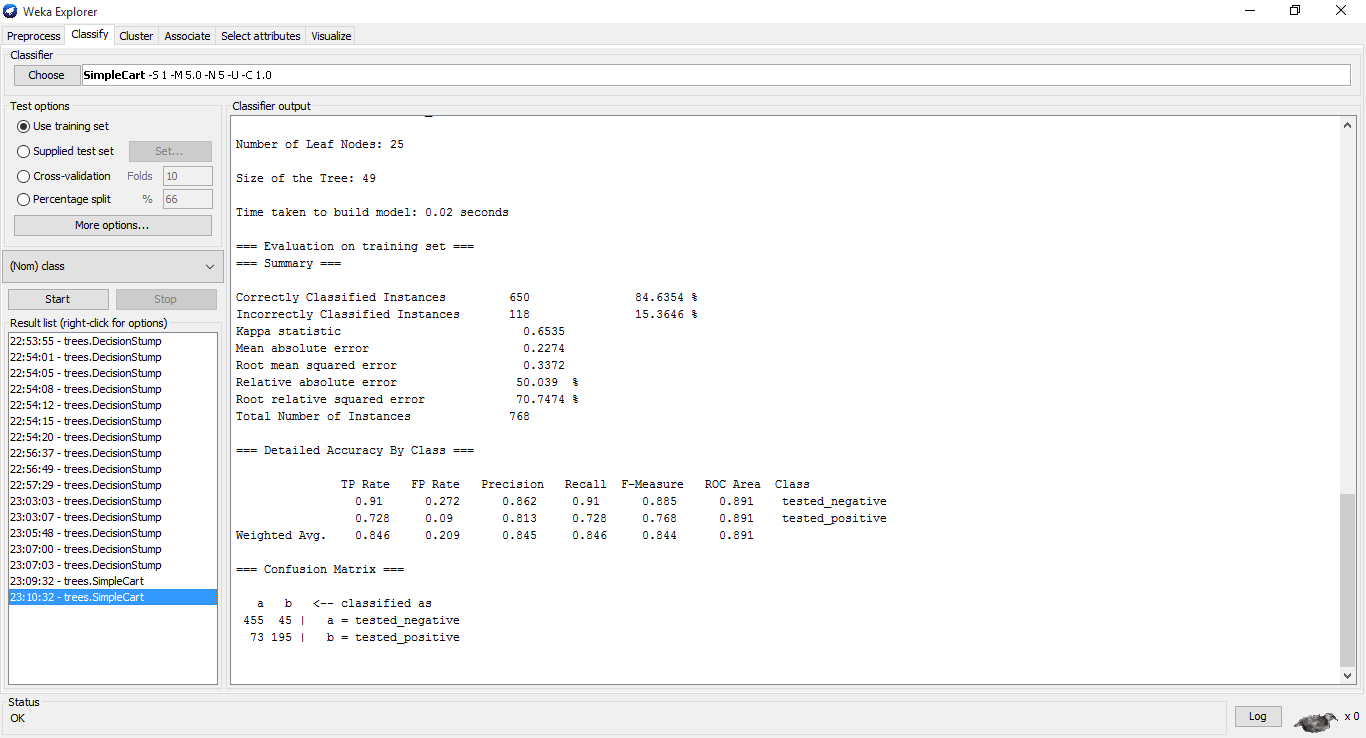
**SimpleCART** will give a high efficiency when tested on training set.



Still the performance when tested against training set will not meet its expectations than the ones obtained from the other sets.

**Changing parameters of SimpleCART and executing:**

If the instances can be correctly classified when Pruned trees are introduced, then % of accuracy is achieved at the max. – 84%



**CONCLUSION:**

* When size of the tree or number of leaf nodes reduces, Percentage of accuracy also reduces.
* While introducing the missing values, both training and test set will yield the same percentage of accuracy when minNumObj > 4
* SimpleCART without noise (default parameters) gives a high % of accuracy.
* SimpleCART without noise (when parameters are changed) gives a lower % of accuracy.
* Resample filter with default values yield a higher % of accuracy when used in 10 cross fold validation.
* Whereas, Resample filter yields a better and max % of accuracy when tested under “Training set”.
* Resample filter when used under SimpleCART (after changing parameters) yields highest percentage of accuracy (less compared to that of the training set and same when compared to that of Cross validation) when minNumObj = 5 and unpruned = false.
* On a training set, Pruned trees achieve max % of accuracy. (99.3%)
* On VOTE DS, decision stump with CV/training/test yields the same percentage of accuracy as i/p is based on only physician-fee-freeze.
* More number of instances leads to low % of accuracy as high chances of incorrect instances.
* A decision stump algorithm for only 2 class with less number of instances yields the max percentage of accuracy (100% sometime)
* A decision stump algorithm for 3 or more classes with less number of instances yields better percentage of accuracy (84% sometimes)
* A decision stump algorithm for 3 or more classes with more number of instances yields poor % of accuracy (on 10 cross validation) – 74%
* SimpleCART – yields the maximum % of accuracy on training set as the number of incorrect instances are reduced.
* More you have the percentage split, leads to less accuracy.
* Noise introduced to an incorrect instance, increases the incorrect instances and makes the % of accuracy low.
* Noise introduced to the correct instance, (say noise introduced to an instance which was tagged wrongly resulting in incorrect instance already) and when this is corrected to be tagged to the proper instance based on the algorithm, then number of incorrect instances reduces thus increasing the % of accuracy.
* LABOR DS – cross field = 20 yields max % of accuracy.
* Cross field = 10 for Labor DS any time (with or without changing parameters) in SimpleCART yields a worst % of accuracy.
* Labor DS for pruned trees with 10 fold cross yields poor %.
* Labor DS under SimpleCART with changing minNumObj = 5 and Pruned trees, max % of accuracy is obtained.
* When minNumObj = 10 and cross validation is also 10, it yields a poor accuracy.
* When internal cross validation >= 20, % of accuracy suffers a lot.
* VOTE – Cross validation multiple iterations do not vary in accuracy.
* VOTE – 80% of the data tested gives max percentage of accuracy.
* Percentage split is not a good option for all these DS. And mainly on a huge DS with many instances as chances of incorrect instances (amongst reminder test data) may be high.
* 80% of the data tested on training set > 66% of the data tested on the same training set(with supply test option).
* 34% of test data tested on a test set > 66% of data tested on the training set – VOTE percentage split – 96% whereas, training set – 95.3%
* Missing values are simple ignored and taken care by SimpleCART by using surrogate method and hence % of accuracy is maintained.
* Missing value is considered as a value in Decision stump.
* Decision stump – Minimum time to execute numeric large data but less accuracy.
* CV in simple cart tells you when to stop the pruning.
* SimpleCART is best for training set.
* Decision stump training set – LABOR DS is based on incorrect instances of that set.
* Percentage split given high leads to inaccuracy.
* Cross validation does not hold good for IRIS Dataset to determine the % of accuracy as the number of incorrect instances is high.