REPORT ON

THE CLASSIFICATION OF DATASETS USING DECISION TREE ALGORITHMS

Submitted by

Arun Rajagopalan

A20360689

For CS422 Data Mining

Illinois Institute of Technology

REPORT ON THE CLASSIFICATION OF DATASETS USING DECISION TREE ALGORITHMS

Introduction:

Classification, in data mining and machine learning, is the process of assigning one of the predetermined categories to an item or an instance that has not been encountered before, using the knowledge of categories of similar items that we have experience dealing with. This report is to document the outcomes of analysis and classification of different datasets using couple of decision tree algorithms in the environment of the machine learning tool Weka. The datasets used for this purpose are the Iris, Vote, Labor and the Diabetes datasets which come bundled with Weka.

Learn about the data:

The foremost step in any data mining activity is learning about the data. Only if we know the datasets that we are going to work on and its characteristics thoroughly well, we can design efficient schemes to classify the data and arrive at a model which has good accuracy. With the classification model, we can predict the category or the class label of any new instance that is posed before us, provided it has the same signature as the other instances in our dataset.

Iris dataset:

The Iris dataset has data about **150 instances** or samples of Iris plants, gathered across **4 attributes or traits,** with the **class attribute being the species** of the iris plant. This is one of the most popular datasets in the field of machine learning and pattern recognition.

Numeric Attributes:

Sl.No	*Attribute Name*	*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

Nominal Attribute(s):

Sl.No	*Attribute name*	*Value/Label*	*Count*
5	class	Iris-setosa	50
		Iris-versicolor	50
		Iris-virginica	50

Vote dataset:

This dataset has vote details for each of the Congressmen in the U.S House of representatives, **435 in total**, on 16 main votes, as identified by the Congressional Quarterly Almanac. Thus there are **16 attributes** alongside the **class attribute which indicates the party that the representative is associated with**. Noteworthy meta-data of the vote dataset is as follows.

Nominal Attributes:

Sl.No	*Attribute name*	*Value/label*	*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
17	Class	democrat	267
		republican	168

Labor dataset:

The labor dataset has **57 instances** where the combination of values for 16 different labor characteristics (like duration, pension etc.) in each instance is used to judge whether an agreement can be deemed good or not. The dataset, hence, has **16 attributes** with a **class attribute that determines if the contract is good or bad**.

Numeric attributes:

Sl.No	*Attribute Name*	*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
3	wage-increase-second-year	2	7	3.972	1.164
4	wage-increase-third-year	2	5.1	3.913	1.304
5	working-hours	27	40	38.039	2.506
6	standby-pay	2	14	7.444	5.028
7	shift-differential	0	25	4.871	4.544
8	statutory-holidays	9	15	11.094	1.26

Nominal attributes:

Sl.No	*Attribute name*	*Value/Label*	*Count*
9	cost-of-living-adjustment	none	22
		tcf	8
		tc	7
10	pension	none	11
		ret_allw	4
		empl_contr	12
11	education-allowance	yes	10
		no	12
12	vacation	below_average	18
		average	17
		generous	16
13	longterm-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
17	class	bad	20
		good	37

Diabetes dataset:

The diabetes dataset is a collection of data across **8 attributes** or diabetes risk factors representing **768 feminine instances** of Pima Indian origin. The class attribute is of binary nominal type indicating whether the subject has tested positive or not.

Numeric Attributes:

Sl.No	*Attribute Name*	*Minimum Value*	*Maximum Value*	*Mean*	*Standard Deviation*
1	preg (Number of times pregnant)	0	17	3.845	3.37
2	plas (Plasma Glucose concentration)	0	199	120.895	31.973
3	pres (Diastolic blood pressure)	0	122	69.105	19.356
4	skin (Triceps skin fold thickness)	0	99	20.536	15.952
5	insu (2 hour serum insulin)	0	846	79.799	115.244
6	mass (Body mass index)	0	67.1	31.993	7.884
7	pedi (Diabetes pedigree function)	0.078	2.42	0.472	0.331
8	age	21	81	33.241	11.76

Nominal Attribute(s):

Sl.No	*Attribute name*	*Value/Label*	*Count*
9	class	tested_negative	500
		tested_positive	268

Introduction to decision tree algorithms:

Decision tree, which is a classification technique, works by iteratively checking the attributes of the dataset for conditions at each level, based on which, the instances are grouped into several nodes, ideally upto a point at which each instance can be assigned a single definite label. There are usually several attributes in a dataset and to pick an attribute for checking a condition at each node or level is a challenge. While there are several ways to do this, the mostly used method is built upon the degree of class impurity levels of nodes. The common class impurity measures are Entropy, Gini and Classification error. Performance of a test condition at any node involving an attribute is evaluated by the difference between the class impurity level of parent node and the child node which is termed as gain. The attribute and the test condition that provides higher gain than its counterparts is chosen for that particular node. This process is recursed for all nodes in the tree, except the leaf nodes, until all branches of the tree end up in leaf nodes.

Some decision tree techniques may stop recursion, even before every branch gets to its leaf node, to favour performance and to reduce generalized error. But usually trees are fully developed and then pruned to optimize for performance. Speaking of the leaf nodes, there are several types of nodes in a decision tree, which we need to get accustomed to before delving deeper.

- ***Root node***: This is a node which holds one or many outgoing edges with no incoming edge, for it is the origin of any decision tree.
- ***Internal node***: This node possesses a single incoming edge and one or many outgoing edges. Internal node is used in propagating the tree from root node to the leaf nodes.
- ***Leaf node***: These nodes are the ending points of a decision tree with one or many incoming edges and no outgoing edge.

Two of the decision tree algorithms that are to be discussed and experimented with in this report are the Decision Stump and the Simple Cart algorithms. A brief description about these algorithms is as below.

Decision Stump algorithm:

This algorithm produces a Decision Stump, which is a variant of decision tree where there is only one level. This implies that the size of a decision tree (the number of nodes in a tree) is always 3 and the internal nodes are absent. There is a single root node, as with all other decision trees, where the attribute with maximal information gain is checked for a condition based on the result of which instances are split into two groups and different class labels are assigned to them. In case of an instance missing the value of the selected attribute, it is tagged the class label that is assigned to the instances that satisfy the test condition.

Parameters:

There are no performance inflicting parameters for this algorithm as the result is a one-level tree. The only parameter that Weka provides for Decision Stump is the 'Debug' parameter, which when set to True will output additional information to the console at some cases.

Simple Cart algorithm:

This algorithm results in multi-level classification or regression decision tress, based on the type of the dependent variable, whether it is nominal or numeric. The underlying technique is same as of that of the general decision trees. It uses the similar greedy approach, where the decision tree is developed by making a set of locally optimum decisions about which attribute is to be selected for data segregation, at every level and node except the leaf node. To pick the best attribute and condition for a node, Simple Cart algorithm uses Information gain which is derived from the entropy class impurity measure.

With multi-level trees, there arises a problem called overfitting, where our model performs well on training data but poorly on general test data. This is called the generalization error. To help reduce overfitting and minimize generalization error, size of the tree is reduced by removing nodes that don't contribute much to the model. We can perform and customize pruning for Simple Cart trees in Weka by appropriately changing some its parameters. The parameters that play an important role in the accuracy of Simple Cart in Weka follow.

Parameters:

minNumObj: This parameter allows to specify a minimum for number of observations at the leaf nodes.

numFoldsPruning: This parameter allows us to specify the number of folds that we wish Weka to use for internal cross-validation.

UseOneSE: When set to True, this parameter enables the use of 1 SE rule in pruning. Several literatures claim that 1SE rule is good at screening out the outliers while pruning.

usePrune: This parameter must be set to True, if pruning is needed to optimize and generalize the decision tree.

Application of decision tree algorithms:

Checking accuracy using the original and modified datasets:

Apart from the original datasets, two more versions of the same are used with the decision tree algorithms. One variant has missing values and the other has considerable noise (some instances are misclassified). Three testing options are used and the test results for all four datasets using both Decision Stump and Simple Cart algorithms are tabulated below.

Legends used in the tabulation and evaluation below:

Relative absolute error: A type of approximation error deduced by dividing the absolute error by the magnitude of exact value. Lower value is desired.

Root relative squared error: Another type of approximation error computed by dividing the original root mean squared error (which is a measure of average magnitude of error) by the same obtained by predicting the mean of target values. Lower value is desired.

Confusion matrix: A table layout which enables easy visualization of competence of an algorithm. The learning template is as follows.

	A* (Predicted)	B* (Predicted)
A (Actual)	True Positive	False Negative
B (Actual)	False Positive	True Negative

Using Iris datasets:

Summary:

Accuracy fields	**Decision Stump algorithm**			**Simple Cart algorithm**		
Testing option	Training set	Percentage split (90%)	Cross-validation (10 fold)	Training set	Percentage split (90%)	Cross validation (10 fold)
ORIGINAL Dataset: Iris						
Correctly classified	100	10	100	147	14	143
Incorrectly classified	50	5	50	3	1	7
Percent of correctly classified	66.6667%	66.6667%	66.6667%	98%	93.3333%	95.3333%
Relative absolute error	50%	50%	50%	5.2482%	14.2857%	9.8273%
Root relative squared error	70.7107%	70.7107%	70.7107%	22.9089%	41.8872%	37.1656%
MISSING Dataset: Iris (2% missing values in each attribute)						
Correctly classified	99	10	94	142	14	136
Incorrectly classified	51	5	56	8	1	14
Percent of correctly classified	66%	66.6667%	62.6667%	94.6667%	93.3333%	90.6667%
Relative absolute error	51%	50%	54.0225%	14.7463%	16.3282%	18.469%
Root relative squared error	71.4143%	70.7107%	75.5296%	35.8351%	41.7447%	47.6079%
NOISY Dataset: Iris (2% noise in each class)						
Correctly classified	96	9	94	135	12	133
Incorrectly classified	54	6	56	15	3	17

Percent of correctly classified	64%	60%	62.6667%	90%	80%	88.6667%
Relative absolute error	58.58%	59.1365%	58.6965%	27.2289%	38.7534%	26.6929%
Root relative squared error	76.5376%	77.3416%	76.8672%	52.1813%	72.3121%	55.6469%

Confusion matrices of 3 datasets when tested using cross validation: **(Decision Stump)**

Classified as		ORIGINAL dataset			MISSING dataset			NOISY dataset		
		a	b	c	a	b	c	a	b	c
setosa	a	50	0	0	48	2	0	47	3	0
versicolor	b	0	50	0	2	38	10	1	44	5
virginica	c	0	50	0	2	40	8	2	45	3

Confusion matrices of 3 datasets when tested using cross validation: **(Simple Cart)**

Classified as		ORIGINAL dataset			MISSING dataset			NOISY dataset		
		a	b	c	a	b	c	a	b	c
setosa	a	50	0	0	48	1	1	47	2	1
versicolor	b	0	47	3	1	42	7	1	44	5
virginica	c	0	4	46	1	3	46	2	6	42

Inferences: (using cross validation results)

- The accuracy of Simple Cart (~95%) is better than that of Decision Stump (~66%) in case of Iris dataset. Percentage of correctly classified instances is considerably higher and the error percentages are lower for Simple Cart.
- Tolerance of both algorithms on dataset with missing values look almost similar (~4% change), but the Decision Stump has smaller change in error estimates. Tolerance is observed by comparing the accuracy and error values on original and the missing dataset. Confusion matrices show the differences in predictions for both datasets, algorithm wise.
- For the given noisy dataset, tolerance is slightly better with Decision Stump than the other, this should probably be owed to simpler approach of the algorithm. Similar to previous case, we get tolerance by comparing the results of original and the noisy datasets while the confusion matrices show the prediction differences between the same.

Using vote datasets:

Summary:

Accuracy fields	**Decision Stump algorithm**			**Simple Cart algorithm**		
Testing option	Training set	Percentage split (90%)	Cross-validation (10 fold)	Training set	Percentage split (90%)	Cross validation (10 fold)
ORIGINAL Dataset: Vote						
Correctly classified	416	39	416	423	39	415
Incorrectly classified	19	4	19	12	4	20
Percent of correctly classified	95.6322%	90.6977%	95.6322%	97.2414%	90.6977%	95.4023%
Relative absolute error	16.5385%	27.1839%	16.693 %	10.9481%	29.3709%	17.2189%
Root relative squared error	40.6726%	64.6536%	41.2142%	30.9353%	64.8945%	41.1466%
MISSING Dataset: Vote (Each attribute is made to miss ten more of its values)						
Correctly classified	411	40	411	423	40	409
Incorrectly classified	24	3	24	12	3	26
Percent of correctly classified	94.4828%	93.0233%	94.4828%	97.2414%	93.0233%	94.023%
Relative absolute error	18.5493%	26.8038%	18.7527%	9.954%	25.2062%	20.331%
Root relative squared error	43.0743%	61.6835%	43.6924%	27.2951%	54.5815%	43.5545%
NOISY Dataset: Vote (Ten instances of each class are misclassified)						
Correctly classified	406	36	406	416	36	408
Incorrectly classified	29	7	29	19	7	27
Percent of correctly classified	93.3333%	83.7209%	93.3333%	95.6322%	83.7209%	93.7931%
Relative absolute error	25.1032%	43.7643%	25.3727%	17.3315%	45.5419%	23.4853%
Root relative squared error	50.1099%	83.9005%	50.7156%	40.348%	83.7324%	49.0516%

Confusion matrices of 3 datasets when tested using cross validation: **(Decision Stump)**

Classified as		ORIGINAL dataset		MISSING dataset		NOISY dataset	
		a	b	a	b	a	b
democrat	a	253	14	254	13	249	20
republican	b	5	163	11	157	9	157

Confusion matrices of 3 datasets when tested using cross validation: **(Simple Cart)**

Classified as		ORIGINAL dataset		MISSING dataset		NOISY dataset	
		a	b	a	b	a	b
democrat	a	255	12	254	13	253	16
republican	b	8	160	13	155	11	155

Inferences: (using cross validation results)

- Accuracy of Decision Stump (~95.6%) is marginally better than Simple Cart (~95.4%) for the vote dataset. Relative absolute error is also lower for Decision Stump.
- Tolerance of both algorithms towards missing values are comparable for the given missing dataset with Decision Stump performing better with minimal change in error values.
- Tolerance of Simple Cart is better than its counterpart in case of noisy dataset.

Using labor datasets:

Summary:

Accuracy fields	**Decision Stump algorithm**			**Simple Cart algorithm**		
Testing option	Training set	Percentage split (90%)	Cross-validation (10 fold)	Training set	Percentage split (90%)	Cross validation (10 fold)
ORIGINAL Dataset: Labor						
Correctly classified	48	5	46	48	5	45
Incorrectly classified	9	1	11	9	1	12
Percent of correctly classified	84.2105%	83.3333%	80.7018%	84.2105%	83.3333%	78.9474%
Relative absolute error	44.5603%	67.0972%	45.9597%	59.5515%	53.4813%	59.2308%
Root relative squared error	66.8601%	81.6116%	70.3345%	76.8348%	89.8097%	89.8961%
MISSING Dataset: Labor (Each attribute is made to miss two more of its values)						
Correctly classified	47	4	43	47	5	43
Incorrectly classified	10	2	14	10	1	14
Percent of correctly classified	82.4561%	66.6667%	75.4386%	82.4561%	83.3333%	75.4386%
Relative absolute error	49.5345%	72.1416%	52.603%	64.573%	55.6827%	65.7207%
Root relative squared error	70.4932%	84.3333%	77.1274%	79.3261%	88.1842%	93.8191%
NOISY Dataset: Labor (Two instances of each class are misclassified)						
Correctly classified	46	4	34	46	4	43

Incorrectly classified	11	2	23	11	2	14
Percent of correctly classified	80.7018%	66.6667%	59.6491%	80.7018%	66.6667%	75.4386%
Relative absolute error	67.5787%	87.5375%	89.5206%	69.0869%	87.2645%	72.6833%
Root relative squared error	82.3376%	106.0718%	105.5946%	82.9274%	105.3445%	93.056%

Confusion matrices of 3 datasets when tested using cross validation: **(Decision Stump)**

Classified as		ORIGINAL dataset		MISSING dataset		NOISY dataset	
		a	b	a	b	a	b
bad	a	11	9	11	9	9	11
good	b	2	35	5	32	12	25

Confusion matrices of 3 datasets when tested using cross validation: **(Simple Cart)**

Classified as		ORIGINAL dataset		MISSING dataset		NOISY dataset	
		a	b	a	b	a	b
bad	a	15	5	13	7	12	8
good	b	7	30	7	30	6	31

Inferences: (using cross validation results)

- In the case of Labor dataset as well, accuracies of the algorithms are comparable with Decision Stump (~81%) performing better than the other (~79%)
- Tolerance of both algorithms on missing dataset is comparable but the noise tolerance of Simple Cart is significantly high in this case.

Using diabetes datasets:

Summary:

Accuracy fields	**Decision Stump algorithm**			**Simple Cart algorithm**		
Testing option	Training set	Percentage split (90%)	Cross-validation (10 fold)	Training set	Percentage split (90%)	Cross validation (10 fold)
ORIGINAL Dataset: Diabetes						
Correctly classified	565	59	552	593	59	577
Incorrectly classified	203	18	216	175	18	191
Percent of correctly classified	73.5677%	76.6234%	71.875%	77.2135%	76.6234%	75.1302%
Relative absolute error	81.8217%	79.0404%	83.6467%	76.4974%	62.2643%	75.2162%

Root relative squared error	90.4671%	86.121%	92.6909%	87.4742%	89.6902%	90.7881%
MISSING Dataset: Diabetes (Each attribute is made to miss fifteen of its values)						
Correctly classified	562	59	541	591	61	571
Incorrectly classified	206	18	227	177	16	197
Percent of correctly classified	73.1771%	76.6234	70.4427%	76.9531%	79.2208%	74.349%
Relative absolute error	82.2275%	82.6427%	84.5279%	77.1324%	66.595%	77.7964%
Root relative squared error	90.6912%	90.7055%	93.4758%	87.6275%	86.3833%	90.8625%
NOISY Dataset: (Fifteen instances of each class are misclassified)						
Correctly classified	553	57	544	614	59	566
Incorrectly classified	215	20	224	154	18	202
Percent of correctly classified	72.0052%	74.026%	70.8333%	79.9479%	76.6234%	73.6979%
Relative absolute error	84.6012%	85.3951%	86.259%	66.1261%	71.8723%	75.2356%
Root relative squared error	91.9909%	92.9417%	94.1549%	81.3285%	86.6637%	91.4657%

Confusion matrices of 3 datasets when tested using cross validation: **(Decision Stump)**

Classified as		ORIGINAL dataset		MISSING dataset		NOISY dataset	
		a	b	a	b	a	b
tested_negative	a	398	102	390	110	418	82
tested_positive	b	114	154	117	151	120	148

Confusion matrices of 3 datasets when tested using cross validation: **(Simple Cart)**

Classified as		ORIGINAL dataset		MISSING dataset		NOISY dataset	
		a	b	a	b	a	b
tested_negative	a	434	66	441	59	418	82
tested_positive	b	125	143	138	130	125	143

Inferences: (using cross validation results)

- For the diabetes dataset, Simple cart algorithm (~75%) scores over the Decision Stump results (~72%) in accuracy. Expectedly error estimates are also lower for Simple Cart.
- Decision Stump has comparatively smaller change in error values for missing dataset and Simple Cart has the same trait for noisy dataset.

Final Inferences:

- Decision Stump performs well in datasets where there is considerable or significant number of nominal attributes because binary splits are more efficient in nominal attributes. Simple Cart performs well in a predominantly numerical attribute based datasets.
- On an average, Decision Stump has better missing value tolerance, as it takes the missing values into account and assigns them a class label and Simple Cart has better noise tolerance, as it splits the noisy items to separate nodes which are later pruned off to get an optimized tree.

Checking accuracy using different parameters: (Applicable only for Simple Cart)

Some of the parameters of Simple Cart are changed and their effect on the accuracy of all the four original datasets is observed.

Using Iris datasets: *Summary:*

Values	Leaf count	Tree size	Training set			Percentage split (90%)			Cross validation (10 fold)		
			Percent of correctly classified (%)	**Relative absolute error (%)**	**Root relative squared error (%)**	**Percent of correctly classified (%)**	**Relative absolute error (%)**	**Root relative squared error (%)**	**Percent of correctly classified (%)**	**Relative absolute error (%)**	**Root relative squared error (%)**
Parameter: minNumObj											
2	5	9	98	5.2482	22.9089	93.3333	14.2857	41.8872	95.3333	9.8273	37.1656
4	4	7	97.3333	6.5815	25.6545	93.3333	14.2857	41.8872	93.3333	13.6051	42.8938
8	3	5	96	11.0306	33.2123	93.3333	14.2857	41.8872	90.6667	18.0655	46.5819
Parameter: numFoldsPruning											
2	4	7	97.3333	6.5815	25.6545	93.3333	14.2857	41.8872	94	12.3927	41.121
4	5	9	98	5.2482	22.9089	93.3333	12.6915	43.05	95.3333	10.1527	36.7274
8	5	9	98	5.2482	22.9089	93.3333	12.6915	43.05	94.6667	10.6381	39.255
Parameter: useOneSE with usePrune True											
False	5	9	98	5.2482	22.9089	93.3333	14.2857	41.8872	95.3333	9.8273	37.1656
True	3	5	96	11.0306	33.2123	93.3333	14.2857	41.8872	94	12.8716	41.4959
Parameter: usePrune (Default True)											
False	6	11	98	4.625	21.5058	93.3333	6.6667	29.8142	94.6667	7.7991	34.848

Using vote datasets: *Summary:*

Values	Leaf count	Tree size	Training set			Percentage split (90%)			Cross validation (10 fold)		
			Percent of correctly classified (%)	**Relative absolute error (%)**	**Root relative squared error (%)**	**Percent of correctly classified (%)**	**Relative absolute error (%)**	**Root relative squared error (%)**	**Percent of correctly classified (%)**	**Relative absolute error (%)**	**Root relative squared error (%)**
Parameter: minNumObj											
2	6	11	97.2414	10.9481	30.9353	90.6977	29.3709	64.8945	95.4023	17.2189	41.1466

4	7	13	97.0115	11.5419	31.7441	90.6977	29.3709	64.8945	95.8621	15.404	38.6672
8	5	9	96.7816	12.4231	33.1078	90.6977	29.3709	64.8945	95.4023	17.5186	40.7472
Parameter: numFoldsPruning											
2	2	3	95.6322	18.7857	40.9106	90.6977	29.3709	64.8945	95.6322	18.842	41.107
4	2	3	95.6322	18.7857	40.9106	90.6977	29.3709	64.8945	95.4023	15.2492	40.6029
8	6	11	97.2414	10.9481	30.9353	90.6977	29.3709	64.8945	95.4023	14.7309	40.2428
Parameter: useOneSE with usePrune True											
False	6	11	97.2414	10.9481	30.9353	90.6977	29.3709	64.8945	95.4023	17.2189	41.1466
True	5	9	96.7816	12.4231	33.1078	90.6977	29.3709	64.8945	95.6322	18.2864	40.8964
Parameter: usePrune (Default True)											
False	51	101	98.3908	5.2582	19.9519	95.3488	15.0616	48.849	94.9425	12.1088	41.9524

Using labor datasets: *Summary:*

Values	Leaf count	Tree size	Training set			Percentage split (90%)			Cross validation (10 fold)		
			Percent of correctly classified (%)	Relative absolute error (%)	Root relative squared error (%)	Percent of correctly classified (%)	Relative absolute error (%)	Root relative squared error (%)	Percent of correctly classified (%)	Relative absolute error (%)	Root relative squared error (%)
Parameter: minNumObj											
2	2	3	84.2105	59.5515	76.8348	83.3333	53.4813	89.8097	78.9474	59.2308	89.8961
4	2	3	84.2105	59.5515	76.8348	83.3333	53.4813	89.8097	73.6842	66.3023	97.1088
8	2	3	84.2105	59.5515	76.8348	66.6667	68.5795	104.828	73.6842	67.6599	94.7975
Parameter: numFoldsPruning											
2	10	19	96.4912	13.6345	31.2347	83.3333	53.4813	89.8097	77.193	60.729	92.4699
4	2	3	84.2105	59.5515	76.8348	83.3333	53.4813	89.8097	77.193	62.2163	96.0821
8	4	7	92.9825	31.673	53.6192	83.3333	53.4813	89.8097	75.4386	63.5449	97.5116
Parameter: useOneSE with usePrune True											
False	2	3	84.2105	59.5515	76.8348	83.3333	53.4813	89.8097	78.9474	59.2308	89.8961
True	2	3	84.2105	59.5515	76.8348	83.3333	53.4813	89.8097	75.4386	69.4087	96.5096
Parameter: usePrune (Default True)											
False	10	19	96.4912	13.6345	31.2347	66.6667	71.7565	107.304	77.193	53.1838	92.0538

Using diabetes datasets: *Summary:*

Values	Leaf count	Tree size	Training set			Percentage split (90%)			Cross validation (10 fold)		
			Percent of correctly classified (%)	Relative absolute error (%)	Root relative squared error (%)	Percent of correctly classified (%)	Relative absolute error (%)	Root relative squared error (%)	Percent of correctly classified (%)	Relative absolute error (%)	Root relative squared error (%)
Parameter: minNumObj											
2	3	5	77.2135	76.4974	87.4742	76.6234	62.2643	89.6902	75.1302	75.2162	90.7881

4	13	25	82.8125	58.3989	76.4291	75.3247	65.0623	91.0598	73.9583	77.5566	92.4326
8	3	5	77.2135	76.4974	87.4742	80.5195	68.8238	86.1926	75.3906	77.1449	90.6128
Parameter: numFoldsPruning											
2	2	3	73.5677	81.8217	90.4671	79.2208	79.4374	86.883	75	79.693	90.8391
4	6	11	79.0365	71.2952	84.4474	80.5195	70.0471	85.6059	74.0885	72.6755	91.3513
8	17	33	84.375	54.5859	73.8919	79.2208	67.308	85.5979	74.7396	74.0473	91.4778
Parameter: useOneSE with usePrune True											
False	3	5	77.2135	76.4974	87.4742	76.6234	62.2643	89.6902	75.1302	75.2162	90.7881
True	3	5	77.2135	76.4974	87.4742	80.5195	70.0471	85.6059	74.349	80.6808	91.2569
Parameter: usePrune (Default True)											
False	80	159	94.5313	18.6775	43.2231	75.3247	57.7378	98.0956	71.7448	66.7987	104.798

Final Inferences:

By analysing all the results of parameter change on the four datasets, the following conclusions can be safely made and they hold good for most cases.

- As the minNumObj parameter increases, the size of the tree and the number of leaf nodes decreases since the increased minimal limit on the number of leaf node observations limits splitting and allows instances to gather at leaf nodes. Since splitting is interrupted, the model accuracy decreases as well.
- Maximizing numFoldsPruning, increases the number of iterations of internal cross-validation. At the end of every iteration, a tree is produced and more trees mean more options from which an optimal tree can be selected. Moderate increase in the parameter value increases the size of the tree and leaves but improves accuracy also. Increasing too much overfits the model to the training set.
- Enabling use1SE usually reduces the accuracy but it generalizes the model for unseen data as 1 SE is known to omit noisy values.
- Enabling Pruning, as the previous parameter, reduces accuracy, but decouples the model a bit more from the training set. Unpruned trees will perform well with training set but it is not generally the case with new data. Pruning is the solution the reduce the overfitting of models to training data.

Effect of class distribution on the experiments:

To study the effect of class distribution on accuracy, a review of distribution of the 4 datasets may be in order.

Sl.No	Dataset	Class values	Distribution (%)
1	Iris	Iris-setosa	33.33%
		Iris-versicolor	33.33%
		Iris-virginica	33.33%

2	Vote	democrat	61.38%
		republican	38.62%
3	Labor	bad	35.09%
		good	64.91%
4	Diabetes	tested_negative	65.105%
		tested_positive	34.895%

Now we link these distribution details with our results to arrive at a new set of inferences on the characteristics of the decision trees.

- Accuracy of general recursive machine learning algorithms on datasets with uniform class distribution is usually high. Since the algorithm has equal count of instances to learn about each class, its predictions will have more discernment. Eg. Simple Cart on Iris data has 95.3% accuracy (on cross-validation).
- While on the other hand, if class distribution is skewed greatly in the favour of a single class, our model will know much about that single class and relatively less about the other classes. Only the favoured class will have desired values in the confusion matrix and all other classes will get hit.
- Also, not all algorithms are benefited by equal class spread. 1-rule decision trees like Decision Stump split on a single attribute, so a uniform distribution may adversely affect their performance. Eg. Decision Stump has 66.6% accuracy on cross-validation with the same Iris dataset.
- Another example where class skew affects accuracy is the percentage split testing option. That option is not stratified by default, hence every split may have different skewed combination of classes ending up with varied results on every run. Eg. Multitude of previous percentage split tests.
- Class skew may also affect cross-validation when the number of folds is too high producing very small training and testing sets where class usually gets dispersed.

Inferences on training and testing methods with cross-validation in focus:

Based on the outcomes of using several training and testing methods available in Weka on the target datasets all this long, the conclusions that follow are obvious.

- Using training set for testing always gives favourable outcomes because it is what the model is built with. The option can be used to deduce the best accuracy that can be achieved in a model, but cannot be used to predict the performance of model on a new data.
- Using percentage split, as discussed before, is not reliable owing to the disproportionate class spread in the sets used for training and testing. To improve accuracy, the process may be done multiple times with different seeds and the accuracy may be aggregated. But variance would be high here which is not desired.

For eg., consider original Iris dataset employing 75% split on Simple Cart with varied seeds.

Seed 1 :	Accuracy – 96.6667%	Seed 2 :	Accuracy – 93.3333%
Seed 3 :	Accuracy – 93.3333%	Seed 4 :	Accuracy – 100%
Seed 5 :	Accuracy – 93.3333%		

The aggregate of these results, which should be the actual accuracy is 95.3332%, but as we can see the individual results vary greatly ranging from 93.3333% to 100% .

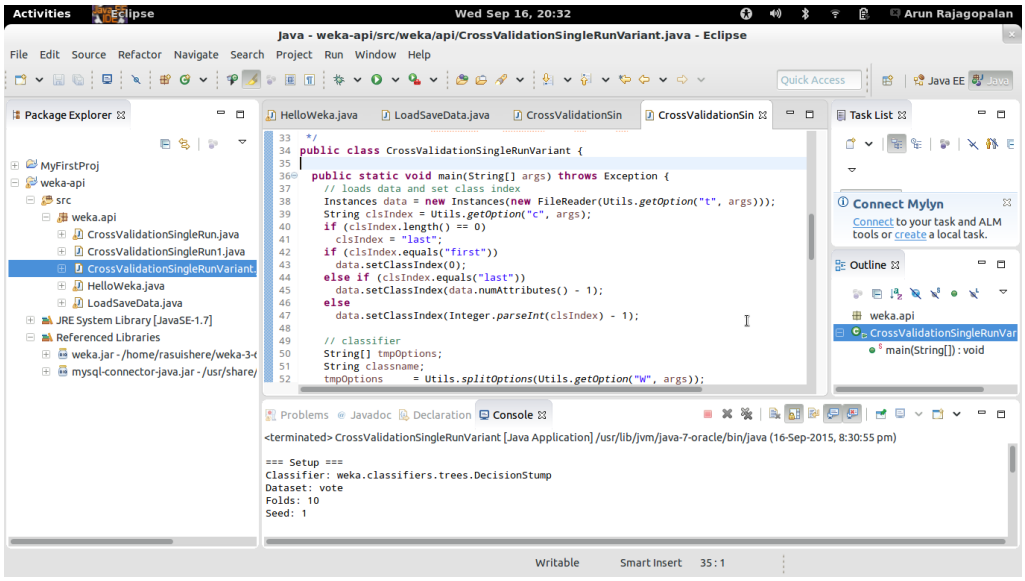
Cross-validation revealed using Java API:

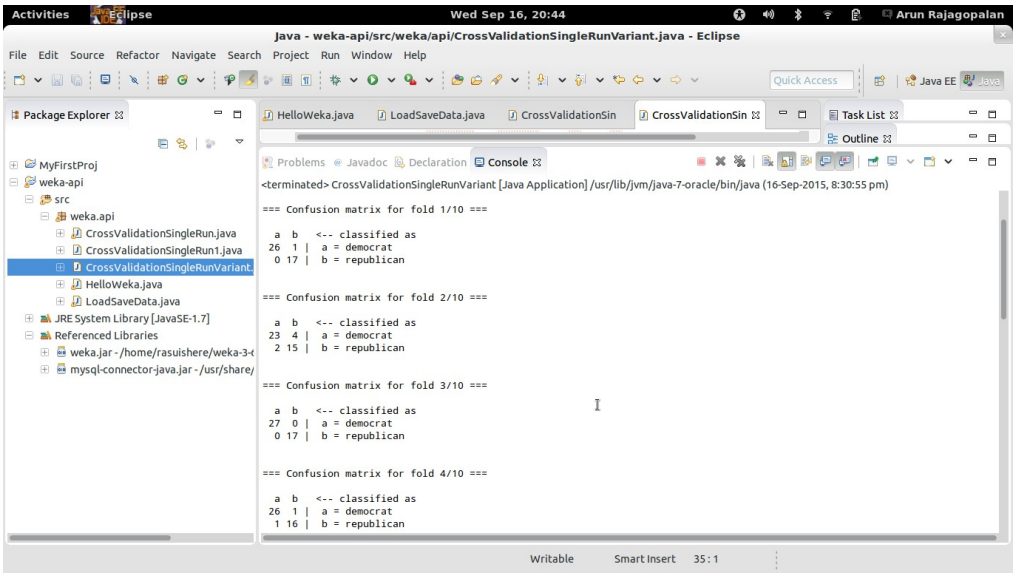
Instead of repeated percentage split, cross-validation can be used to get reliable, generalized accuracies that will hold true for the newer datasets also. The main factor for this credibility in cross-validation is stratification which means the class distribution of the original dataset is preserved in the partitions. A brief description about the working fashion of cross-validation follows.

- The entire dataset is divided into the specified number (fold count) of partitions of equal or nearly equal sizes with original class distribution intact amongst them.

  For instance, 10 fold cross-validation on Iris, divides the entire dataset into 10 subsets with 15 instances each. Every subset has 5 setosa, 5 virginica and 5 versicolor instances or 33.33% of each class, which is the class distribution of the original dataset.
- Now, 9 partitions are used for training and building a model while the last partition is used for testing. For the next iteration, another partition is used for testing and while the other 9 is used for training. This process repeats for n iterations where n is the number of fold.
- In this case, for 10 fold validation, total number of iterations is 10 where each data instance is used 9 times for training and once for testing.
- Aggregate of accuracies at each iteration is shown as the final accuracy.
- It is important to note that after running 10 iterations, Weka runs one more iteration to build the final classifier model using the entire dataset. So effectively, for n-fold validation, Weka runs n+1 iterations of training and testing.
- Additionally, to exactly know the size of each fold that will be used in cross-validation, the filter 'stratifiedremovefolds' in Weka can be used.
- Also gauging the performance of an algorithm at each fold of cross-validation can also be done, not by GUI though. Weka has a Java API harnessing which per-iteration stats can be found out.

For this project, to get a deeper understanding of cross-validation, API was used with an example program available in Weka site publicly called 'CrossValidationSingleRunVariant.java'. Some changes in the code were made for customization and to enable compilation. Using the program, the confusion matrices for vote dataset when used with Decision Stump algorithm was found out. Snapshots of the same are below.





Average of accuracies achieved at each fold was 95.6322% which is same as that displayed in Weka while using the GUI tool. Thus by using the API, it is made clear that our understanding about the working of cross-validation is correct.

Summary:

- Firstly, before starting the experiments, we perused the datasets for better understanding. Then after a brief introduction on decision tree algorithms, we went on with the application of two of the decision tree algorithms, Decision Stump and Simple Cart, on the four target sets of data.
- The performance of the algorithms on the datasets, before and after the introduction of missing values and noise was analysed.
- Broadly, it was seen that Decision Stump has better tolerance with missing values and Simple Cart has better noise tolerance.
- Also Decision Stump seemed to perform better on binary splitting problems or problems involving nominal attributes while Simple Cart seemed to deal better with numerical attributes.
- Impact of parameters of Simple Cart on its performance was looked into.
- Class distribution does have an impact on the accuracy. Datasets with greatly skewed class distributions do not give favourable results with any algorithm.
- Of all the training and testing methods, cross-validation was chosen as the best owing to its generalized results. The working methodology of cross-validation in Weka was studied at depth using the Weka Java API.

Conclusion:

Thus the aforementioned experiments using decision tree algorithms on the four chosen datasets in Weka, have brought forth much better understanding of classification techniques in data mining and on how to master them. Couple of algorithms and several testing approaches were studied in great detail. The study would prove worthy and valuable for any data analytics student.