REPORT

Wednesday, 15th November 2023

By

Codie Shannon

30013375

Assignment Two

In

COMP.7212

Artificial Intelligence Techniques

Table of Contents

Part 1 – Decision Tree Learning

A. Describe the Method

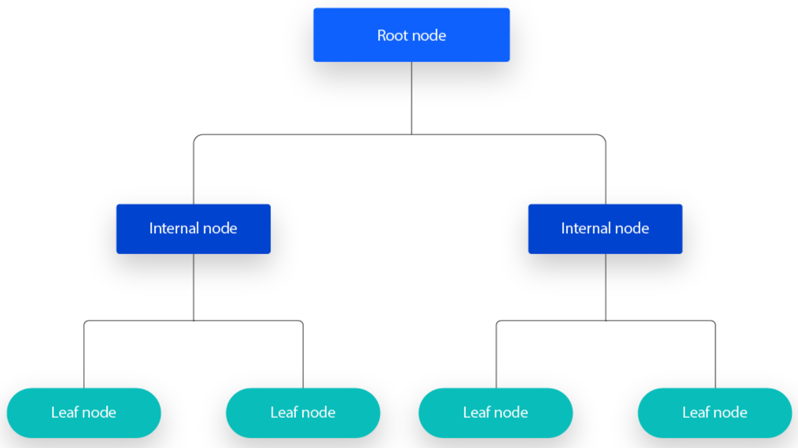
I. The Decision Tree Learning from a Technical Perspective

Describe what exactly the tree is trying to learn from a technical perspective.

A decision tree is a non-parametric supervised learning algorithm. The algorithm is utilized for both regression and classification tasks. Decision trees are comprised of a hierarchical, tree structure, consisting of a root node, branches, internal nodes, and leaf nodes (IBM, n.d.).

Figure 1

Hierarchical Tree Structure



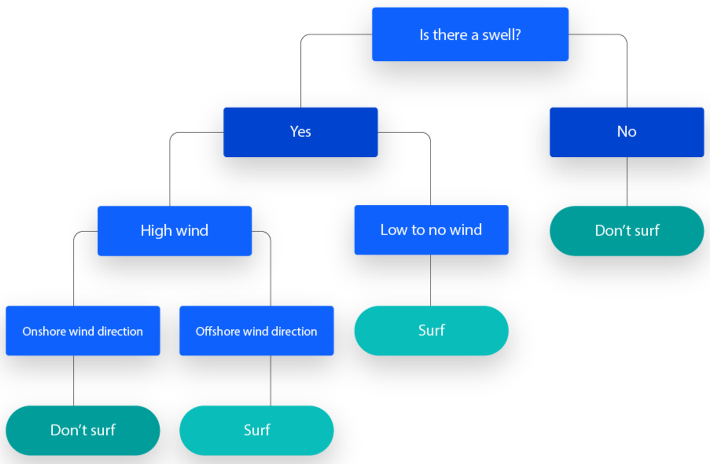
(IBM, n.d.)

As shown in Figure 1, a decision tree begins with a root node, which does not have any incoming branches. The output branches from the root node proceed to feed into the decision nodes, otherwise known as internal nodes. Based upon the accessible features, both nodes conduct an evaluation to formulate equivalent subsets, which are designated by terminal or leaf nodes. Leaf nodes constitute all possible outcomes within the dataset (IBM, n.d.).

As an example, imagine you were attempting to assess whether you should hit the waves, you may use the decision rules shown in Figure 2 to decide.

Figure 2

Surfing Decision Tree



(IBM, n.d.)

Decision tree learning enlists a divide and conquer strategy by performing a greedy search to recognize the optimal split points within a tree. This process of splitting is repeated in a recursive manner until all, or the greater some of records have been classified under specific class labels (IBM, n.d.).

II. How the Algorithm Generates the "Best" Answer

Describe how the tree goes about generating the 'best' answer.

Whilst there are numerous methods to select the best attribute at each node, a couple methods, information gain and Gini impurity, act as favourable splitting criterion for decision tree models. The methods aid to evaluate the quality of individual test conditions and how well it will be able to classify samples into a class (IBM, n.d.).

Unfortunately, it is quite difficult to explain information gain without first discussing entropy. Entropy is a notion that originated from information theory, which measures the impurity of the sample values (IBM, n.d.). Entropy is defined by the formula shown in Figure 3 below.

Figure 3

Entropy Formula

$$\text{Entropy}(S) = -\sum_{c \in C} p(c)\log_2 p(c)$$

(IBM, n.d.)

S - depicts the data set that entropy is calculated

c - portrays the classes in set, S

p(c) - describes the proportion of data points that belong to class c to the number of total data points in set, S

Entropy values may fall between 0 and 1. If all samples in data set, S, belong to one class, entropy will equal zero. However, if half of the samples are classified as one class and the other portion are in another class, entropy will be at its highest point at 1. In pursuance of the best feature to split on and locate the optimal decision tree, the attribute with the tiniest amount of entropy should be utilized (IBM, n.d.).

Information gain portrays the difference in entropy before and after a separation on a given attribute. The attribute with the greatest information gain will produce the foremost split as it's performing the best job at classifying the training data as specified by its target classification (IBM, n.d.). Information gain is primarily depicted with the formula found in Figure 4 below.

Figure 4

Information Gain Formula

$$\text{Information Gain}(S,a) = \text{Entropy}(S) - \sum_{v \in values(a)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

(IBM, n.d.)

a - portrays a specific attribute or class label

Entropy(S) - is the entropy of dataset, S

|Sv|/ |S| - depicts the proportion of the values in S to the number of values in dataset, S

Entropy(S) - is the entropy of dataset, S

Gini impurity is the expectation of inaccurately classifying a random data point in the dataset if it were labelled based upon the class distribution of the dataset. Indistinguishably from entropy, if set, S is pure i.e., it belongs to one class, in conjunction, impurity is set to zero (IBM, n.d.). This is denoted by the formula shown in Figure 5 below.

Figure 5

Gini Impurity Formula

$$\text{Gini Impurity} = 1 - \sum_i (p_i)^2$$

(IBM, n.d.)

III. How Overfitting the Data was Avoided

Describe how you avoided overfitting the data and how this method works. If you did not need to do this, explain why that was so.

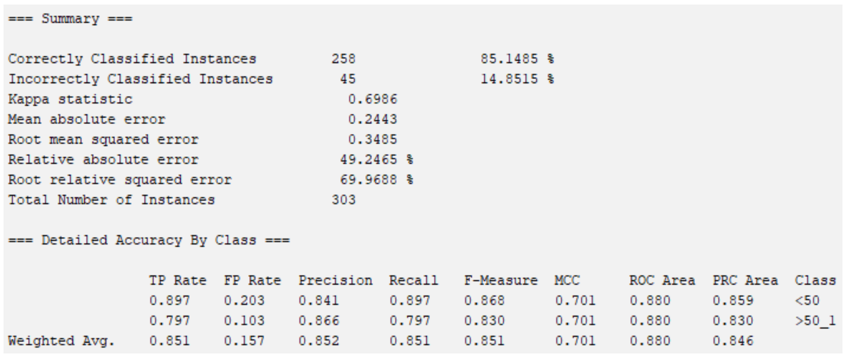
Overfitting emerges when a decision tree is extremely dependent on questionable features of the training data with the result of a reduction in predictive power for unseen instances. A singular approach stands out for avoiding overfitting amongst many. Pruning solidifies in many forms, however, Weka (the software utilized for this assignment) utilizes two, subtree replacement and subtree raising. Fortunately, they are both toggled on by default.

Subtree replacement operates by replacing nodes in a decision tree with a leaf, reducing the overall number of tests along a specific path. The process begins from the leaves of the fully formed tree and moves backwards towards the root (Ozgur, 2012).

In subtree raising, a node may be moved upwards towards the root of the tree, replacing other such nodes along the way. Unfortunately, subtree raising frequently has a negligible effect on decision tree models. Regrettably, there is frequently no clear method to predict the utility of the option (Ozgur, 2012).

Figure 6

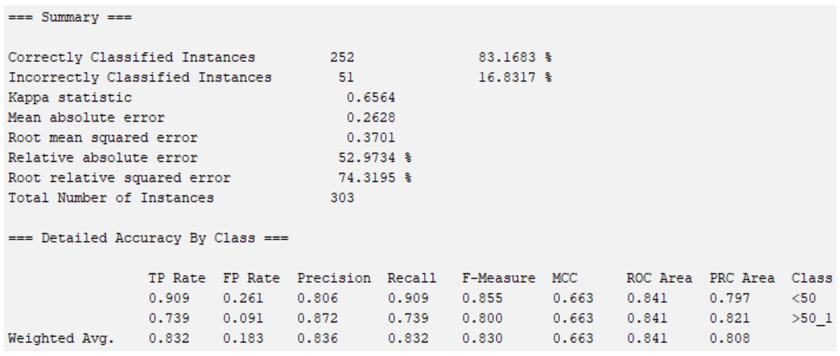
Decision Tree - Test Data



```
=== Summary ===

Correctly Classified Instances         258               85.1485 %
Incorrectly Classified Instances        45               14.8515 %
Kappa statistic                          0.6986
Mean absolute error                      0.2443
Root mean squared error                  0.3485
Relative absolute error                 49.2465 %
Root relative squared error             69.9688 %
Total Number of Instances              303

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.897    0.203    0.841      0.897    0.868      0.701    0.880     0.859     <50
                 0.797    0.103    0.866      0.797    0.830      0.701    0.880     0.830     >50_1
Weighted Avg.    0.851    0.157    0.852      0.851    0.851      0.701    0.880     0.846
```

Figure 7

Decision Tree - Data Set



```
=== Summary ===

Correctly Classified Instances         252               83.1683 %
Incorrectly Classified Instances        51               16.8317 %
Kappa statistic                          0.6564
Mean absolute error                      0.2628
Root mean squared error                  0.3701
Relative absolute error                 52.9734 %
Root relative squared error             74.3195 %
Total Number of Instances              303

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.909    0.261    0.806      0.909    0.855      0.663    0.841     0.797     <50
                 0.739    0.091    0.872      0.739    0.800      0.663    0.841     0.821     >50_1
Weighted Avg.    0.832    0.183    0.836      0.832    0.830      0.663    0.841     0.808
```

Taking a glance at the results shown in Figure 6 and 7 above, we can see that minimal overfitting has occurred. The test data has a success rate of eighty-five-point-one, whilst the data set has a success rate of eighty-three-point-one. This leaves a small overfitting separation of only two percent.

IV. Important J48-Specific Setting that was Changed and How it Helped.

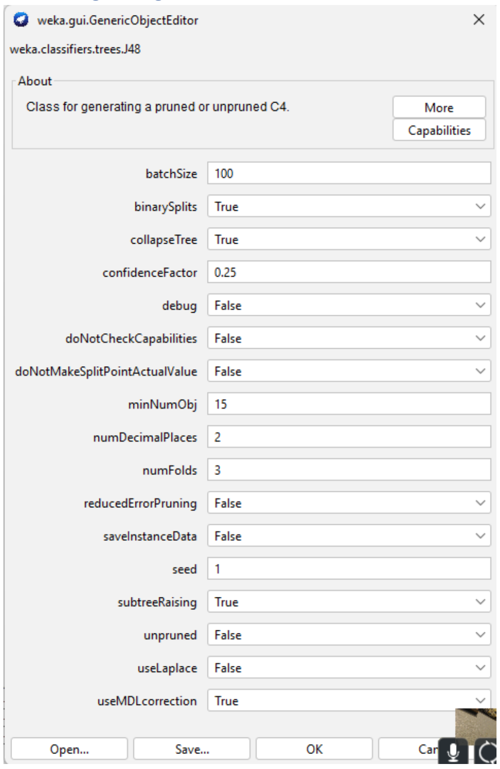
Describe an important J48-specific setting you changed that worked well and why you think it helped.

Before the J48 specific setting may be explained it is important to define nominal data. Nominal data is data that can be labelled or classified into mutually exclusive categories within a variable (Scribbr, n.d.). The primary J48 setting modified that saw the most improvement was binary splits. Binary splits is a J48 specific setting which specifies whether to utilize binary splits on nominal data. This is a process by which the tree is grown by considering a singular nominal value as opposed to all other nominal values. As a result, the tree only consists of two branches protruding from any given node (Schankacademy, n.d.).

Given specific circumstances, a decision learning algorithm utilizing binary split could see a positive impact due to a prime nominal value being selected. In this circumstance it is assumed that the nominal value being selected would have a good or great stature within the nominal data to overcome the lesser nominal data values being selected by an ordinary decision learning algorithm not utilizing binary split.

V. Settings Page and Test Options

i. Settings Page



weka.gui.GenericObjectEditor

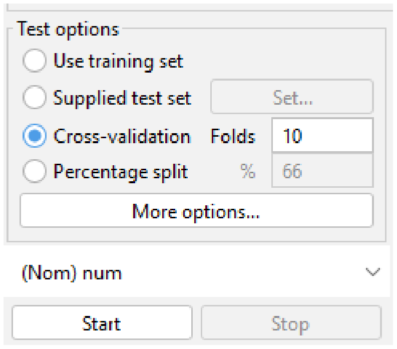
weka.classifiers.trees.J48

About

Class for generating a pruned or unpruned C4.

batchSize	100
binarySplits	True
collapseTree	True
confidenceFactor	0.25
debug	False
doNotCheckCapabilities	False
doNotMakeSplitPointActualValue	False
minNumObj	15
numDecimalPlaces	2
numFolds	3
reducedErrorPruning	False
saveInstanceData	False
seed	1
subtreeRaising	True
unpruned	False
useLaplace	False
useMDLcorrection	True

ii. Test Options



Test options

Use training set

Supplied test set

[✓ Cross-validation] Folds 10

Percentage split % 66

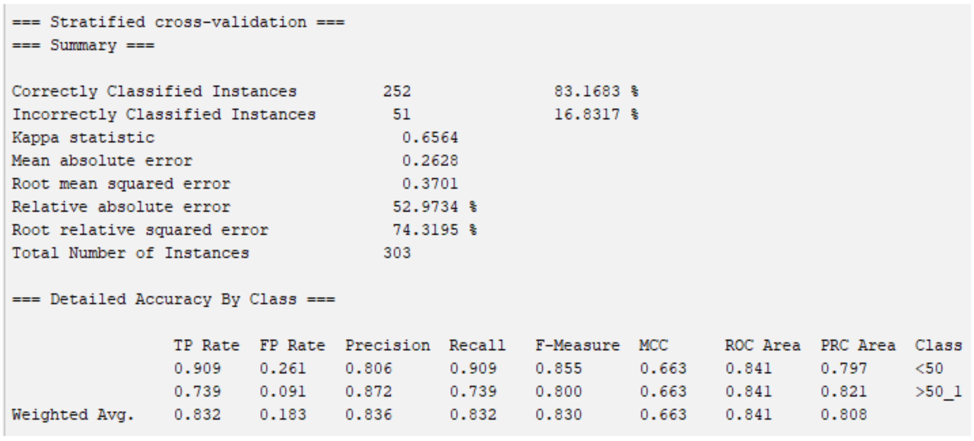
(Nom) num

B. Discuss the Results

I. Results and Confusion Matrix

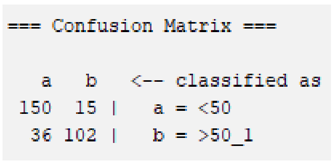
A screenshot or table of your results and confusion matrix.

i. Results



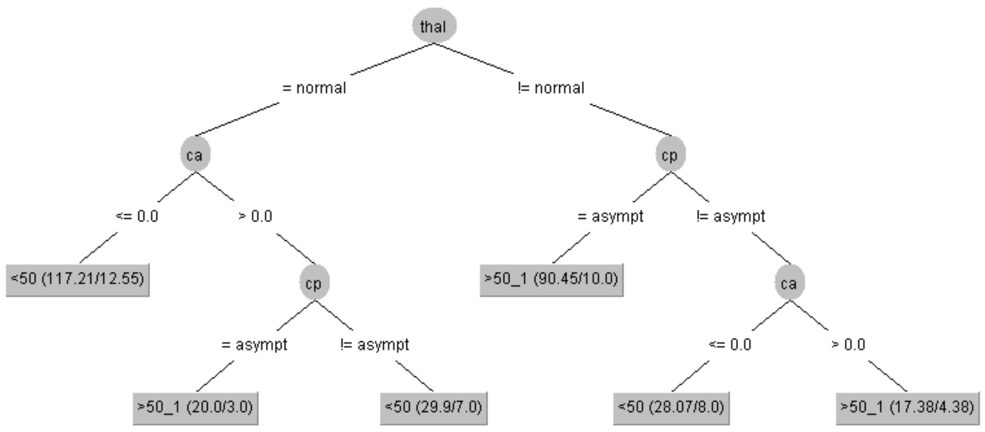
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         252               83.1683 %
Incorrectly Classified Instances        51               16.8317 %
Kappa statistic                          0.6564
Mean absolute error                      0.2628
Root mean squared error                  0.3701
Relative absolute error                 52.9734 %
Root relative squared error             74.3195 %
Total Number of Instances              303

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.909    0.261    0.806      0.909    0.855      0.663    0.841     0.797     <50
                 0.739    0.091    0.872      0.739    0.800      0.663    0.841     0.821     >50_1
Weighted Avg.    0.832    0.183    0.836      0.832    0.830      0.663    0.841     0.808
```

ii. Confusion Matrix



```
=== Confusion Matrix ===

   a   b   <-- classified as
 150  15 |   a = <50
  36 102 |   b = >50_1
```

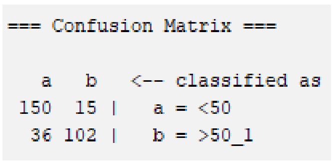
iii. Final Decision Tree



IV. Describe the Findings from the Results.

What do the results mean? What facts can you deduce from this result? E.g. from the confusion matrix.

For the purpose of comprehending the confusion matrix, we must first understand what a confusion matrix is. A confusion matrix is a table that is utilized to define the performance of a classification algorithm. The matrix visualizes and summarizes the performance of a classification algorithm (ScienceDirect, n.d.).



```
=== Confusion Matrix ===

   a   b   <-- classified as
 150  15 |   a = <50
  36 102 |   b = >50_1
```

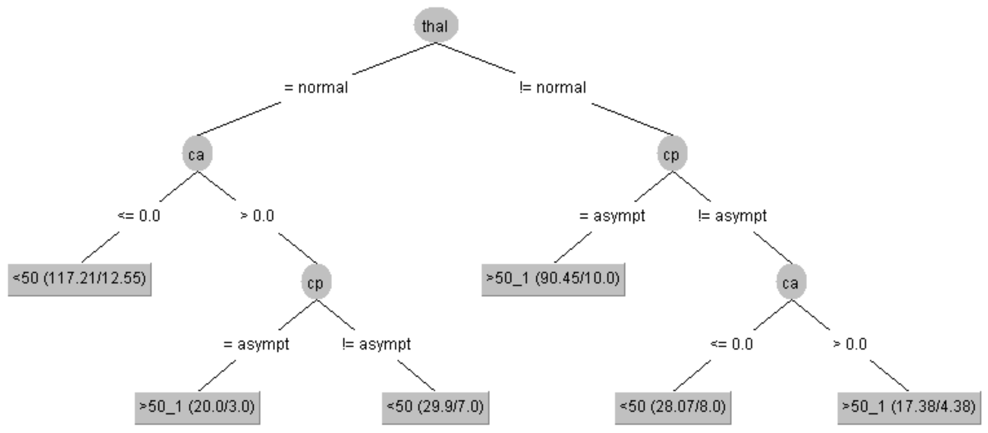
Inspecting the confusion matrix above, we may conceive that two representations exist a and b. a defines the classification group of patients with a diameter narrowing by fifty percent or less. Whilst b depicts the classification group of cases with a diameter narrowing by fifty percent or greater. A patient with a diameter narrowing by fifty percent or less is thought to have no heart disease. In conjunction with this, the opposite applies, a patient with a narrowing diameter by fifty percent or greater is thought to have heart disease.

From the confusion matrix we may deduce that one hundred and fifty cases were classified correctly for group a (no heart disease), whilst fifteen patients were misclassified from group b (heart disease) as a. In addition to this, we may conclude that one hundred and two sufferers were categorized precisely for group b (heart disease), while thirty-six patients were miss-categorised from group a (no heart disease) as b.

V. What Were the Most important Features / Attributes?

Which were the most important features/attributes in predicting the class variable for your final decision tree configuration? How did you decide this?

To understand what the most important features / attributes were, we must first understand feature importance which heavily involves the Gini index. To recap, Gini impurity (index) is utilized as an impurity measure. The calculation of feature importance involves two steps, calculating the importance of each node, and calculating each feature's importance utilizing node importance splitting on that feature. Without going into too much detail we may grasp the essential concept behind these operations. In simple terms, the process utilizes a predictive model to contrast and compare features against one another to locate the optimal feature for a specific node thus resulting in an optimal hierarchical decision tree. The execution of this process may sound tremendous, however in real-time, it takes but a few seconds.



Examining the decision tree above, we can conclude that the most important features for this specific use case are thal, ca, and cp.

C. Come to a Conclusion

I. Could the Results you Obtained be Practically Useful to a Hospital?

Could the results you obtained be practically useful to a hospital? Why or why not? (must be more than a yes or no answer)

Unfortunately, I do not believe the results obtained could be practically useful to a hospital. Whilst the a (no heart disease) classification group experienced a good success rate at ninety-point-nine percent. The b (heart disease) classification group did not observe similar success at only seventy-three-point-nine percent. Although seventy-three-point-nine percent is no small success rate, it cannot be overlooked when it comes to misdiagnose of heart disease patients who may suffer severely due to this particular algorithm being utilized practically within hospitals.

II. Practical Advice to the Hospital to Improve Detection of Heart Disease?

Based solely on these results, what would your practical advice to the hospital be to improve detection of heart disease? How did you make this decision?

I believe the practical advice given to the hospital should be to not utilize the decision tree learning algorithm in its present condition. Whilst the algorithm observed fair success rates at ninety-point-nine percent and seventy-three-point-nine percent for classification groups a (no heart disease) and b (heart disease). I imagine an acceptable success rate for heart disease should be no less than ninety percent or higher overall. In addition, I believe testing must be performed on other machine learning algorithms to locate the optimal solution for the heart disease problem.

If a hospital implemented the decision tree learning algorithm to detect heart disease, I would greatly recommend training multiple decision tree learning algorithms in the hope of performing cross validation to locate the optimal solution by finding the average result i.e., heart disease or no heart disease. The aforementioned process may involve increasing the test data size dramatically.

III. Anything Else Interesting Found or Learned During Your Experimentation.

Anything else interesting found or learned during your experimentation.

During experimentation, an abundance of knowledge was learned about the selection methods utilized to select the best attribute at each node in decision tree learning, information gain and Gini impurity.

To recap, entropy is a notion that originated from information theory, which measures the impurity of the sample values (IBM, n.d.). Alongside this, this unit of impurity is utilized heavily within information gain to calculate the optimal attribute at each node.

Information gain portrays the difference in entropy before and after a separation on a given attribute. The attribute with the greatest information gain will produce the foremost split as it’s performing the best job at classifying the training data as specified by its target classification (IBM, n.d.). Information gain is primarily depicted with the formula found in Figure 8 below.

Figure 8

Information Gain Formula

$$\text{Information Gain}(S,a) = \text{Entropy}(S) - \sum_{v \in values(a)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

(IBM, n.d.)

a - depicts a precise attribute or class label

Entropy(S) - is the entropy of dataset, S

|Sv|/ |S| - portrays the proportion of the values in S to the number of values in dataset, S

Entropy(S) - is the entropy of dataset, S

The Gini index (impurity) is the expectation of incorrectly classifying a random data point in the dataset if it were labelled based upon the class distribution of the dataset. Identically from entropy, if set, S is pure i.e., it belongs to one class, in conjunction, impurity is set to zero (IBM, n.d.). This is denoted by the formula shown in Figure 9 below.

Figure 9

Gini Impurity Formula

$$\text{Gini Impurity} = 1 - \sum_i (p_i)^2$$

(IBM, n.d.)

Part 2 – Clustering

A. Describe the Method

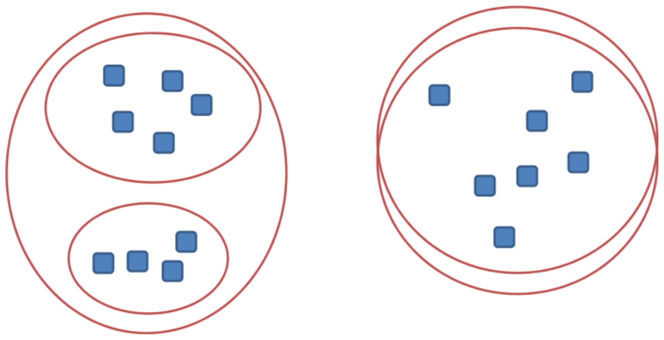
I. Clustering from a Technical Perspective

Describe what exactly the tree is trying to learn from a technical perspective.

Clustering is an unsupervised learning algorithm. It may be considered the most important unsupervised learning problem. Similar to other unsupervised learning algorithms, clustering deals with locating a structure in a collection of unlabelled data. A loose definition of clustering could be "the process of organizing objects into groups whose members are similar in some fashion". A cluster is therefore a collection of objects which are "similar" between themselves and are "dissimilar" to the objects belonging to other clusters (Mishra, 2017).

Figure 10

Clustering Example



(Mishra, 2017)

In Figure 10 shown above, how do we know what the best clustering solution is? In order to locate a particular clustering solution, we must define the similarity measures for the clusters.

For clustering, we must define a proximity measure for two data points. Proximity here signifies how similar/dissimilar the samples are with respect to each other (Mishra, 2017).

Similarity – measure S(xi, xk): large if xi, xk are similar.

Dissimilarity (or distance) – measure D(xi, xk): small if xi, xk are similar.

There are numerous similarity measures which could be utilized as shown in Figure 11, 12 and 13.

Figure 11

Vectors: Cosine Distance

$$s(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^t \mathbf{x}'}{\|\mathbf{x}\| \; \|\mathbf{x}'\|}$$

(Mishra, 2017)

Figure 12

Sets: Jaccard Distance

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

(If A and B are both empty, we define $J(A,B) = 1$.)

$$0 \leq J(A,B) \leq 1.$$

(Mishra, 2017)

Figure 13

Points: Euclidean Distance (q=2)

$$d(\mathbf{x}, \mathbf{x}') = \left(\sum_{k=1}^{d} |x_k - x'_k|^q \right)^{1/q},$$

(Mishra, 2017)

A “good” proximity measure is VERY application dependent. The clusters must be invariant under the transformations “natural” to the problem. In addition, whilst clustering it is not advised to perform normalization on data that are drawn from multiple distributions (Mishra, 2017).

II. How the Algorithm Generates the “Best” Answer

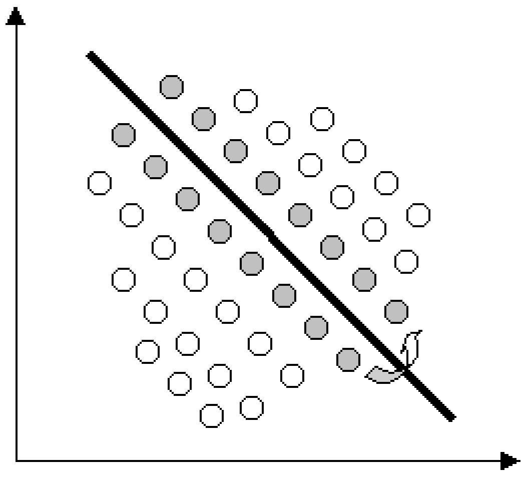
Describe how the tree goes about generating the ‘best’ answer.

The objective of clustering is to determine the internal grouping in a set of unlabelled data. How do you decide what constitutes a good clustering? Regrettably, it may be shown that there is no outright “best” criterion which would be independent of the final aim of the clustering. In consequence, it is the user who must supply this criterion, in such a fashion that the result of the clustering may suit their needs. Clustering algorithms may be classified under four different types, exclusive, overlapping, hierarchical, and probabilistic (Mishra, 2017).

In the first classification, objects (data) are grouped in an exclusive manner, so that if a definite data point belongs to a certain cluster, then it could not be incorporated into another cluster. A simple example of this can be found in Figure 14 below, where the separation of points is obtained by a straight line on a bi-dimensional plane (Mishra, 2017).

Figure 14

Exclusive Clustering



(Mishra, 2017)

The second type of clustering, overlapping, utilizes fuzzy sets to cluster data, so that each point can belong to multiple clusters with differentiating degrees of membership. In this case, data will be associated to an appropriate membership value.

Similarly, hierarchical clustering is based upon the union between the two nearest clusters. The starting condition is realized by setting every data point as a cluster. Following a few iterations, the algorithm reaches the final clusters wanted.

Finishing, the final type of clustering utilizes a completely probabilistic approach.

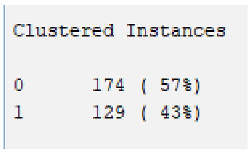
III. How Overfitting the Data was Avoided

Describe how you avoided overfitting the data and how this method works. If you did not need to do this, explain why that was so.

Unfortunately, modifying the SimpleKMeans settings did not result in any notable improvements, as such, little could be done regarding overfitting. However, in saying this, no problems arose for overfitting.

Figure 15

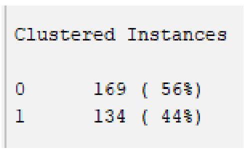
Clustering - Test Data



```
Clustered Instances

0       174 ( 57%)
1       129 ( 43%)
```

Figure 16

Clustering - Data Set



```
Clustered Instances

0       169 ( 56%)
1       134 ( 44%)
```

Taking a glimpse at the results shown in Figure 15 and 16 above, we can perceive that negligible overfitting has occurred. The test data has a success rate of fifty-seven and forty-three percent for cluster 0 (no heart disease) and cluster 1 (heart disease) respectively. Whilst the data set has a success rate of fifty-six and forty-four percent for cluster 0 (no heart disease) and cluster 1 (heart disease) correspondingly. This leaves a tiny overfitting separation of only one percent.

IV. Important SimpleKMeans-Specific Setting that was Changed and How it Helped.

Describe an important SimpleKMeans-specific setting you changed that worked well and why you think it helped.

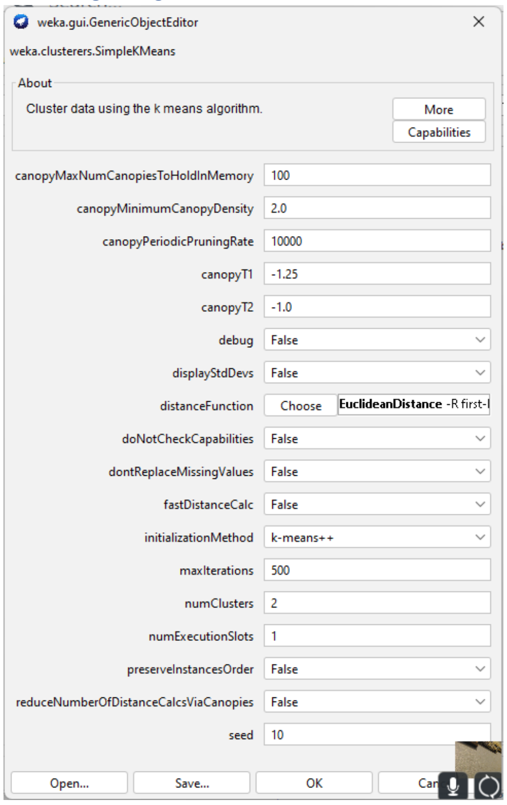
Unfortunately, modifying the SimpleKMeans settings did not prove any use as a noticeable improvement was not realised. However, an improvement was made over the base settings. In the first instance, the initialization method was set to random. Random, as the name suggests, selects a random initialization method for individual executions of the algorithm.

However, the primary factor that affects the performance of K-means clustering is how the initial cluster centres are selected. If they are too close to one another, or too far away from the actual data clusters, the algorithm can take longer to converge, or even get stuck in a local optimum (Callahan, 2023).

Appropriately, it is best to select an initialization method outright. The chosen method was k-means++. K-means++ is a smart centroid initialization method for the K-means algorithm. The goal is to spread out the starting centroid by assigning the initial centroid randomly, then selecting the rest of the centroids based upon the maximum squared distance. The concept is to push the centroids as far as possible from each other (Sharma, 2023).

V. Settings Page and Test Options

i. Settings Page



weka.gui.GenericObjectEditor

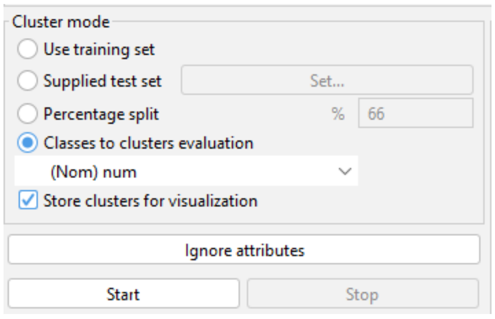
weka.clusterers.SimpleKMeans

About

Cluster data using the k means algorithm.

canopyMaxNumCanopiesToHoldInMemory	100
canopyMinimumCanopyDensity	2.0
canopyPeriodicPruningRate	10000
canopyT1	-1.25
canopyT2	-1.0
debug	False
displayStdDevs	False
distanceFunction	Choose **EuclideanDistance** -R first-l
doNotCheckCapabilities	False
dontReplaceMissingValues	False
fastDistanceCalc	False
initializationMethod	k-means++
maxIterations	500
numClusters	2
numExecutionSlots	1
preserveInstancesOrder	False
reduceNumberOfDistanceCalcsViaCanopies	False
seed	10

ii. Test Options

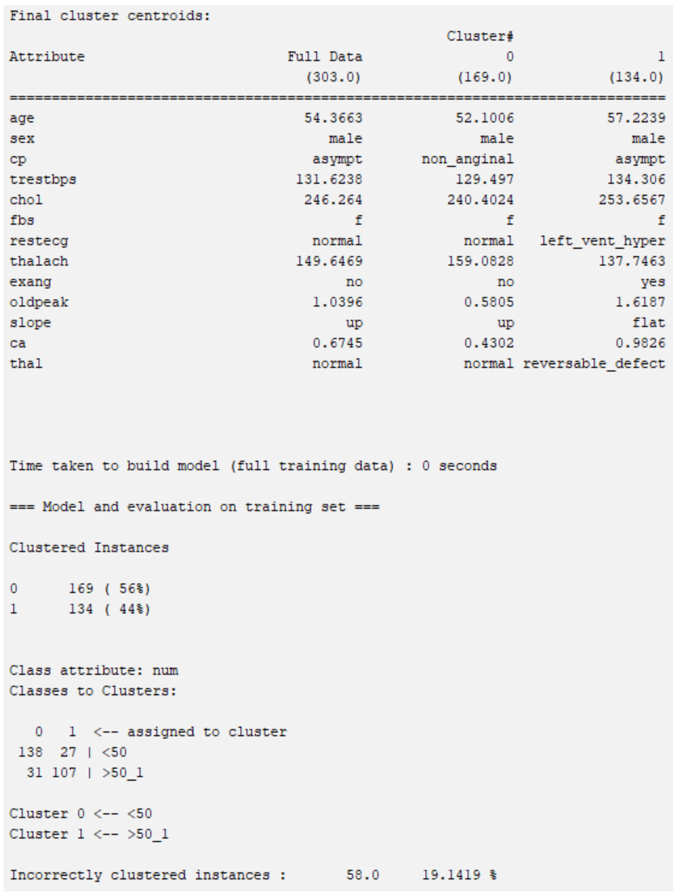


B. Discuss the Results

I. Results and Confusion Matrix

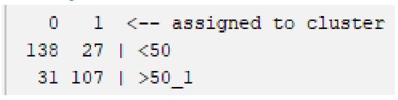
A screenshot or table of your results and confusion matrix.

i. Results



```
Final cluster centroids:
                                         Cluster#
Attribute                Full Data              0                1
                           (303.0)        (169.0)          (134.0)
==================================================================
age                        54.3663        52.1006          57.2239
sex                           male           male             male
cp                          asympt    non_anginal           asympt
trestbps                  131.6238        129.497          134.306
chol                       246.264       240.4024         253.6567
fbs                              f              f                f
restecg                     normal         normal  left_vent_hyper
thalach                   149.6469       159.0828         137.7463
exang                           no             no              yes
oldpeak                     1.0396         0.5805           1.6187
slope                           up             up             flat
ca                          0.6745         0.4302           0.9826
thal                        normal         normal reversable_defect




Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       169 ( 56%)
1       134 ( 44%)


Class attribute: num
Classes to Clusters:

   0   1  <-- assigned to cluster
 138  27 | <50
  31 107 | >50_1

Cluster 0 <-- <50
Cluster 1 <-- >50_1

Incorrectly clustered instances :       58.0     19.1419 %
```

ii. Confusion Matrix

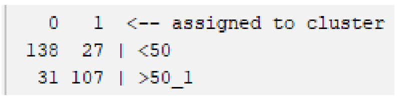


```
   0   1  <-- assigned to cluster
 138  27 | <50
  31 107 | >50_1
```

IV. Describe the Findings from the Results.

What do the results mean? What facts can you deduce from this result? E.g., from the confusion matrix.

For the purpose of comprehending the confusion matrix, we must first understand what a confusion matrix is. A confusion matrix is a table that is utilized to define the performance of a classification algorithm. The matrix visualizes and summarizes the performance of a classification algorithm (ScienceDirect, n.d.).



```
   0   1  <-- assigned to cluster
 138  27 | <50
  31 107 | >50_1
```

Examining the confusion matrix above, we may conclude that two clusters exist 0 and 1. 0 depicts the classification cluster of cases with a diameter narrowing by fifty percent or less. Whilst 1 defines the classification cluster of patients with a diameter narrowing by fifty percent or greater. A patient with a diameter narrowing by fifty percent or less is believed to have no heart disease. In conjunction with this, the opposite applies, a patient with a narrowing diameter by fifty percent or greater is believed to have heart disease.

From the confusion matrix we may gather that one hundred and thirty-eight patients were categorized correctly for cluster 0 (no heart disease), while twenty-seven cases were miscategorized from cluster 1 (heart disease) as 0. In conjunction with this, we may conceive that one hundred and seven sufferers were classified accurately for cluster 1 (heart disease), whilst thirty-one patients were misclassified from cluster 0 (no heart disease) as 1.

V. What Were the Most important Features / Attributes?

Do these results show you which the most important features/attributes were? Why or why not?

Unfortunately, dissimilar to decision tree learning, clustering is an unsupervised machine learning algorithm which analyses objects (data) to determine which cluster an object should belong to. A cluster is therefore a collection of objects which are "similar" between themselves and are "dissimilar" to the objects belonging to other clusters (Mishra, 2017). Put simply, clustering has no such thing akin to "most important" features.

C. Come to a Conclusion

I. Could the Results you Obtained be Practically Useful to a Hospital?

Could the results you obtained be practically useful to a hospital? Why or why not? (must be more than a yes or no answer)

Regrettably, I could not imagine the results acquired being practically useful to a hospital. Each classification cluster, 0 (no heart disease) and 1 (heart disease) observed poor success rates. The success rates observed were each below ninety percent at eighty-three-point-six and seventy-seven-point-five percent respectively. These success rates are atrocious at best and should not be taken under consideration by any hospital. Misdiagnose of heart disease in patients could lead to severe suffering and/or death.

II. Practical Advice to the Hospital to Improve Detection of Heart Disease?

Based solely on these results, what would your practical advice to the hospital be to improve detection of heart disease? How did you make this decision?

I postulate that the practical advice supplied to the hospital should be to not utilize the clustering algorithm under any circumstance. The algorithm noticed horrendous success rates at eighty-three-point-six and seventy-seven-point-five percent for classification clusters 0 (no heart disease) and 1 (heart disease). I believe a justifiable success rate for heart disease should be no lower than ninety percent or higher overall. In conjunction, I surmise testing should be executed on other machine learning algorithms to find the optimal solution for the heart disease problem.

III. Anything Else Interesting Found or Learned During Your Experimentation.

Anything else interesting found or learned during your experimentation.

Throughout the time of experimentation, an ample amount of knowledge was formed about the similarity measures utilized within the clustering algorithm to group similar objects together in clusters, Cosine distance, Jaccard distance, Euclidean distance.

To recap, A loose definition of clustering could be “the process of organizing objects into groups whose members are similar in some fashion”. A cluster is therefore a collection of objects which are “similar” between themselves and are “dissimilar” to the objects belonging to other clusters (Mishra, 2017). To identify the similarities between objects we must define the similarity measures for the clusters. For clustering, we must define a proximity (similarity) measure for two data points. Proximity here signifies how similar/dissimilar the samples are with relation to one another (Mishra, 2017).

Similarity – measure S(xi, xk): large if xi, xk are similar.

Dissimilarity (or distance) – measure D(xi, xk): small if xi, xk are similar.

There are numerous similarity measures which could be utilized as shown in Figure 17, 18 and 19.

Figure 17

Vectors: Cosine Distance

$$s(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^t \mathbf{x}'}{\|\mathbf{x}\| \ \|\mathbf{x}'\|}$$

(Mishra, 2017)

Figure 18

Sets: Jaccard Distance

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

(If A and B are both empty, we define $J(A,B) = 1$.)

$$0 \le J(A,B) \le 1.$$

(Mishra, 2017)

Figure 19

Points: Euclidean Distance (q=2)

$$d(\mathbf{x}, \mathbf{x}') = \left(\sum_{k=1}^{d} |x_k - x'_k|^q \right)^{1/q},$$

(Mishra, 2017)

Part 3 – Multi-Layer Perceptron

A. Describe the Method

I. Multilayer Perceptron from a Technical Perspective

Describe what exactly the tree is trying to learn from a technical perspective and how the tree goes about generating the 'best' answer.

Similar to Decision Tree, MLP (Multi-layer Perceptron) is a non-parametric supervised learning algorithm. The algorithm is utilized to simplistically perform binary classification i.e., it predicts whether input belongs to a specific category of interest or not (ex: fraud / not-fraud) (Leonel, 2018).

The perceptron is a linear classifier, which is an algorithm that classifies input by separating two categories with a straight line. Input is typically a feature vector x multiplied by weights w and added to a bias b: y = w * x + b (Leonel, 2018).

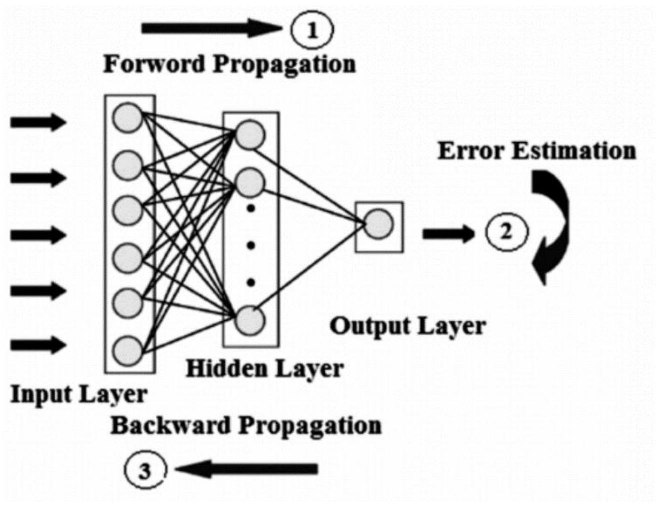
An MLP may be thought of as a deep artificial neural network. Multilayer perceptron's are composed of more than one perceptron. They are comprised of an input layer to receive the signal, an output layer that makes a prediction or decision about the input, and in between these two layers, are an arbitrary number of hidden layers that are the true computational engine of the MLP (Leonel, 2018).

Multilayer perceptrons train on a set of input-output twins and learn to model the dependencies (or correlation) between those inputs and outputs. Training revolves around adjusting the parameters or the biases and weights of the model in order to minimize possible errors. Backpropagation is utilized make those bias and weigh adjustments relative to the error, and the error itself may be measured in countless ways, including by root mean squared error (RMSE) (Leonel, 2018).

Feedforward networks like MLPs, have a remarkable resemblance to ping pong, in that they are primarily involved in two motions, a constant back and forth (forward and backward passes) (as shown in Figure 20 below) (Leonel, 2018).

Figure 20

Multilayer Perceptron Diagram



(Leonel, 2018)

Forward Pass - the signal flow navigates from the input layer through the hidden layers to the output layer, and the decision of the output layer is measured against the ground truth labels (Leonel, 2018).

Backward Pass - utilizing backpropagation and the chain rule of calculus, partial derivatives of the error function regarding the numerous weights and biases are back-propagated through the multilayer perceptron. That act of differentiation gives us a gradient, or a landscape of error, along which the parameters can be adjusted as they proceed one step closer to the error minimum (Leonel, 2018).

Backpropagation - a procedure to repeatedly adjust the weights so as to minimize the difference between actual output and desired output (Leonel, 2018).

Hidden Layers – which are neuron nodes stacked in between inputs and outputs, permitting neural networks to learn more complex features (such as XOR logic, the XOR operator triggers when input exhibits either one trait or another, but not both, it stands for "exclusive OR") (Leonel, 2018).

Perceptron's construct a single output based upon several real-valued inputs by forming a linear combination utilizing input weights (and occasionally sending the output through a non-linear activation function). The math terminology for this may be seen in Figure 21 below (Leonel, 2018).

Figure 21

Perceptron Math Function

$$y = \varphi(\sum_{i=1}^{n} w_i x_i + b) = \varphi(\mathbf{w}^T \mathbf{x} + b)$$

(Leonel, 2018)

where w denotes the vector of weights, x is the vector of inputs, b is the bias and ϕ is the non-linear activation function (Leonel, 2018).

The bias may be thought of as how much flexibility the perceptron has. Similar to the constant b of a linear function y = ax + b, it allows use to move the line down and up to fit the prediction with the data more accurately. Without b (the bias) the line will always go through the origin (0, 0) and a poorer fit may be returned as a result (Leonel, 2018).

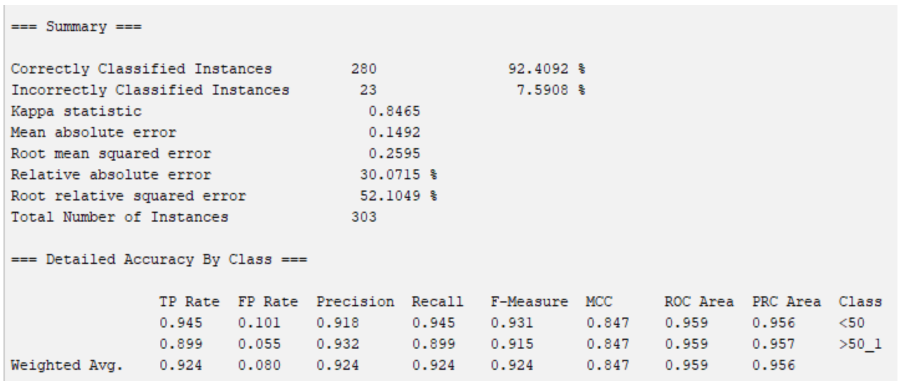
II. How Overfitting the Data was Avoided

Describe how you avoided overfitting the data and how this method works. If you did not need to do this, explain why that was so.

One method to minimize overfitting when it comes to an artificial neural network which is prone to it such as Multilayer Perceptron is to reduce the network's capacity by removing layers or reducing the number of elements within the hidden layers. Fortunately, Weka performs this action by default, 'a' is the default value for the number of hidden layers. 'a' adds the number of attributes and classes together before dividing the total by two.

Figure 22

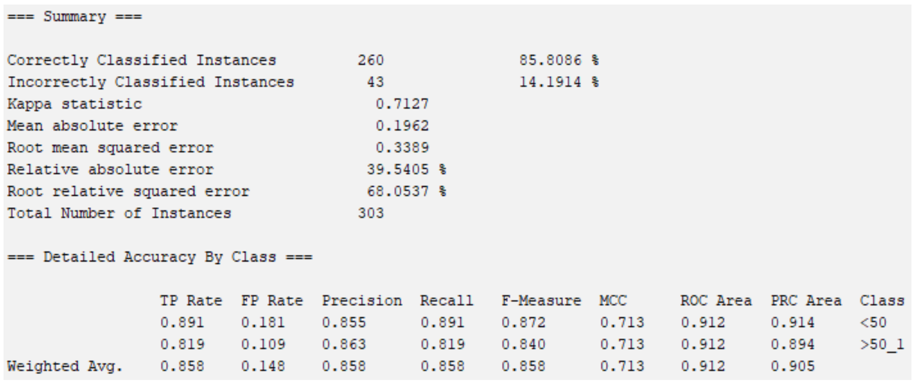
Multilayer Perceptron - Test Data



```
=== Summary ===

Correctly Classified Instances         280               92.4092 %
Incorrectly Classified Instances        23                7.5908 %
Kappa statistic                          0.8465
Mean absolute error                      0.1492
Root mean squared error                  0.2595
Relative absolute error                 30.0715 %
Root relative squared error             52.1049 %
Total Number of Instances              303

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.945    0.101    0.918      0.945    0.931      0.847    0.959     0.956     <50
                 0.899    0.055    0.932      0.899    0.915      0.847    0.959     0.957     >50_1
Weighted Avg.    0.924    0.080    0.924      0.924    0.924      0.847    0.959     0.956
```

Figure 23

Multilayer Perceptron - Data Set



```
=== Summary ===

Correctly Classified Instances         260               85.8086 %
Incorrectly Classified Instances        43               14.1914 %
Kappa statistic                          0.7127
Mean absolute error                      0.1962
Root mean squared error                  0.3389
Relative absolute error                 39.5405 %
Root relative squared error             68.0537 %
Total Number of Instances              303

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.891    0.181    0.855      0.891    0.872      0.713    0.912     0.914     <50
                 0.819    0.109    0.863      0.819    0.840      0.713    0.912     0.894     >50_1
Weighted Avg.    0.858    0.148    0.858      0.858    0.858      0.713    0.912     0.905
```

Unfortunately, examining the results shown in Figure 22 and 23 above, we can perceive that substantial overfitting has occurred. The test data has a success rate of ninety-two-point-four percent, whilst the data set has a success rate of eighty-five-point-eight percent. This leaves a mass overfitting separation of six-point-six percent. Through further optimization it could be possible to bridge the gap.

III. Important Perceptron-Specific Setting that was Changed and How it Helped.

Describe an important Perceptron-specific setting you changed that worked well and why you think it helped.

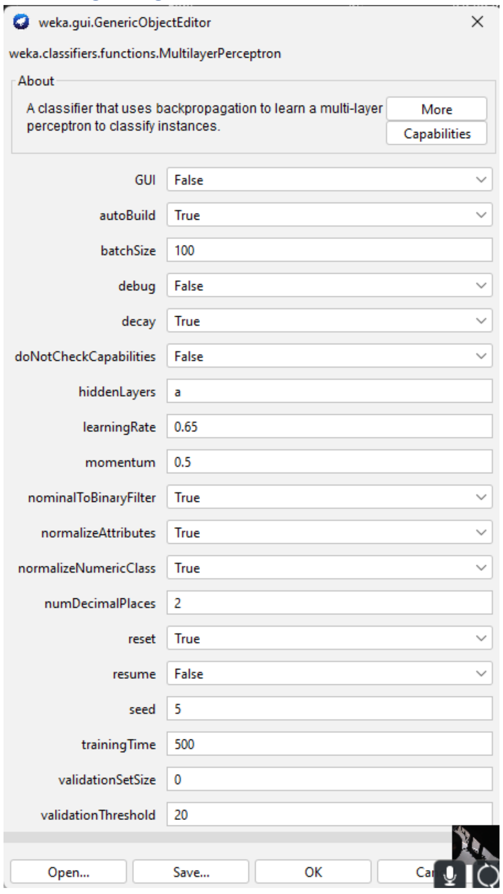
The backpropagation learning rate (known as learningRate within Weka), is a hyper-parameter utilized to control the pace at which the multilayer perceptron learns or updates the values of a parameter estimate while looping backwards. Rephrased, the learning rate manages the weights of the neural network concerning the loss gradient. The learning rate specifies how frequently the neural network should refresh the notions or knowledge it has learned whilst backpropagating (Deepchecks, n.d.).

As expected, the parameter governs how rapidly the algorithm moves towards the optimal weights. If the learning rate is too large the optimal solution may be skipped. If the learning rate is too small the algorithm may require countless iterations to converge to the best values. So, utilizing a good learning rate is crucial for the multilayer perceptron (Rakhecha, 2019).

The original backpropagation learning rate was improved upon by upscaling the value from 0.3 to 0.65. This greatly improved the accuracy for the algorithm by 0.6601. However, if it went any higher it could risk being too large and result in the optimal solution being skipped.

V. Settings Page and Test Options

i. Settings Page



weka.gui.GenericObjectEditor

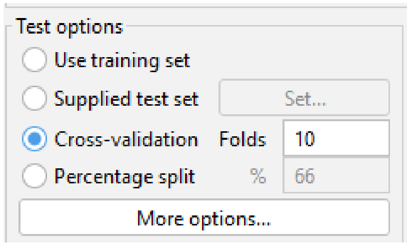
weka.classifiers.functions.MultilayerPerceptron

About

A classifier that uses backpropagation to learn a multi-layer perceptron to classify instances.

GUI	False
autoBuild	True
batchSize	100
debug	False
decay	True
doNotCheckCapabilities	False
hiddenLayers	a
learningRate	0.65
momentum	0.5
nominalToBinaryFilter	True
normalizeAttributes	True
normalizeNumericClass	True
numDecimalPlaces	2
reset	True
resume	False
seed	5
trainingTime	500
validationSetSize	0
validationThreshold	20

ii. Test Options



Test options

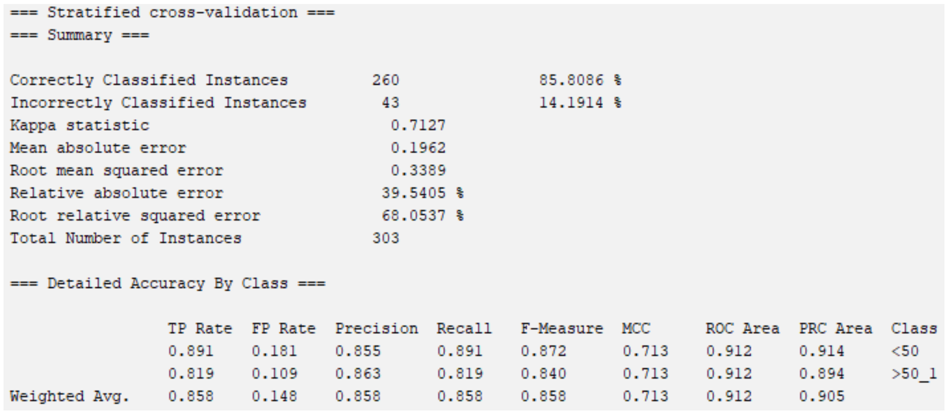
- Use training set
- Supplied test set
- [✓ Cross-validation] Folds 10
- Percentage split % 66

B. Discuss the Results

I. Results and Confusion Matrix

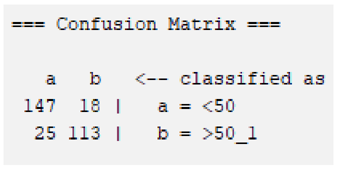
A screenshot or table of your results and confusion matrix.

i. Results



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         260               85.8086 %
Incorrectly Classified Instances        43               14.1914 %
Kappa statistic                          0.7127
Mean absolute error                      0.1962
Root mean squared error                  0.3389
Relative absolute error                 39.5405 %
Root relative squared error             68.0537 %
Total Number of Instances              303

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.891    0.181    0.855      0.891    0.872      0.713    0.912     0.914     <50
                 0.819    0.109    0.863      0.819    0.840      0.713    0.912     0.894     >50_1
Weighted Avg.    0.858    0.148    0.858      0.858    0.858      0.713    0.912     0.905
```

ii. Confusion Matrix

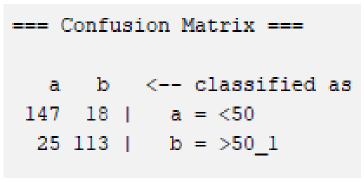


```
=== Confusion Matrix ===

   a   b   <-- classified as
 147  18 |   a = <50
  25 113 |   b = >50_1
```

IV. Describe the Findings from the Results.

What do the results mean? What facts can you deduce from this result? E.g. from the confusion matrix.

For the purpose of comprehending the confusion matrix, we must first understand what a confusion matrix is. A confusion matrix is a table that is utilized to define the performance of a classification algorithm. The matrix visualizes and summarizes the performance of a classification algorithm (ScienceDirect, n.d.).



```
=== Confusion Matrix ===

   a   b   <-- classified as
 147  18 |   a = <50
  25 113 |   b = >50_1
```

Inspecting the confusion matrix above, we may conclude that two representations exist a and b. a portrays the classification group of sufferers with a diameter narrowing by fifty percent or less. Whilst b depicts the classification group of patients with a diameter narrowing by fifty percent or greater. A patient with a diameter narrowing by fifty percent or less is thought to have no heart disease. In conjunction with this, the opposite applies, a patient with a narrowing diameter by fifty percent or greater is believed to have heart disease.

From the confusion matrix we may deduce that one hundred and forty-seven cases were classified accurately for group a (no heart disease), whilst eighteen cases were misclassified from group b (heart disease) as a. In addition to this, we may conclude that one hundred and thirteen sufferers were categorized correctly for group b (heart disease), while twenty-five patients were miscategorized from group a (no heart disease) as b.

V. What Were the Most important Features / Attributes?

Do these results show you which the most important features/attributes were? Why or why not?

Unfortunately, you cannot derive feature importance from multilayer perceptrons. Multilayer perceptrons utilize backpropagation to repeatedly adjust the weights so as to minimize the difference between actual output and desired output to locate an optimal solution (Leonel, 2018).

C. Come to a Conclusion

I. Could the Results you Obtained be Practically Useful to a Hospital?

Could the results you obtained be practically useful to a hospital? Why or why not? (must be more than a yes or no answer)

Sadly, I do not see the possibility of the obtained results being practically useful to a hospital. Each classification group, a (no heart disease) and b (heart disease) noticed mediocre success rates. The success rates observed were each below the substandard of ninety percent at eighty-nine and eighty-one-point-eight percent correspondingly. These success rates are not awful however, they should not be taken under consideration by any hospital. Misdiagnose of heart disease could be dangerous for patients and could lead to severe suffering either financially or medically.

II. Practical Advice to the Hospital to Improve Detection of Heart Disease?

Based solely on these results, what would your practical advice to the hospital be to improve detection of heart disease? How did you make this decision?

My belief is that the practical advice submitted to the hospital should be to not utilize the multilayer perceptron algorithm in its current condition. Whilst the algorithm observed fairly good success rates at eighty-nine and eighty-one-point-eight percent for classification groups a (no heart disease) and b (heart disease). I conceive a maintainable success rate for heart disease should be no less than ninety percent or higher summed. In addition, I feel testing should be executed on other machine learning algorithms to locate the optimal solution for the heart disease problem.

If a hospital applied the multilayer perceptron algorithm to detect heart disease, I would highly suggest optimizing the multilayer perceptron by, the addition of more data, utilizing multiple algorithms to find an optimal result, and tuning the algorithm to locate an optimal solution for the heart disease problem.

III. Anything Else Interesting Found or Learned During Your Experimentation.

Anything else interesting found or learned during your experimentation.

At the time of experimentation, an interesting setting was found in regard to multilayer perceptrons within Weka, learningRate. learningRate within Weka is the label given to the backpropagation learning rate of multilayer perceptrons within Weka. The backpropagation learning rate is a hyper-parameter utilized to control the speed at which the multilayer perceptron learns or updates the values of a parameter estimate while looping backwards. Expressed differently, the learning rate manages the weights of the neural network concerning the loss gradient. The learning rate specifies how regularly the neural network should revitalize the notions or knowledge it has learned whilst backpropagating (Deepchecks, n.d.).

References

Callahan, E. (2023, March 9). *How do you Improve the Speed and Efficiency of K-Means Clustering?* LinkedIn. https://www.linkedin.com/advice/1/how-do-you-improve-speed-efficiency-k-means-clustering#:~:text=them%20in%20Python.-,1%20Choose%20a%20smart%20initialization,stuck%20in%20a%20local%20optimum.

Deepchecks. (n.d.). *Learning Rate in Machine Learning*. https://towardsdatascience.com/https-medium-com-dashingaditya-rakhecha-understanding-learning-rate-dd5da26bb6de#:~:text=This%20parameter%20determines%20how%20fast,good%20learning%20rate%20is%20crucial.

IBM. (n.d.). *What is a Decision Tree?* https://www.ibm.com/topics/decision-trees#:~:text=Decision%20tree%20learning%20employs%20a,classified%20under%20specific%20class%20labels.

Leonel, J. (2018, October 30). Multilayer Perceptron. *Medium*. https://medium.com/@jorgesleonel/multilayer-perceptron-6c5db6a8dfa3

Mishra, S. (2017, May 20). Unsupervised Learning and Data Clustering. *Towards Data Science*. https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a

Ozgur, A. (2012, July 31). What is Pruned and Unpruned Tree in Weka. *Stack Overflow*. https://stackoverflow.com/a/11739494

Rakhecha, A. (2019, June 29). Understanding Learning Rate. *Towards Data Science*. https://towardsdatascience.com/https-medium-com-dashingaditya-rakhecha-understanding-learning-rate-dd5da26bb6de#:~:text=This%20parameter%20determines%20how%20fast,good%20learning%20rate%20is%20crucial.

Schankacademy. (n.d.). *J48 Classifier Parameters*. https://www.schankacademy.com/demos/data-analytics/xt/lib/docs/0/j48_parameters.pdf

Science Direct. (n.d.). *Confusion Matrix*. https://www.sciencedirect.com/topics/engineering/confusion-matrix#:~:text=A%20confusion%20matrix%20is%20a,performance%20of%20a%20classification%20algorithm.

Scribbr. (n.d.). *What is Nominal Data?* https://www.scribbr.com/frequently-asked-questions/what-is-nominal-data/#:~:text=Nominal%20data%20is%20data%20that,%2C%20train%2C%20tram%20or%20bicycle.

Sharma, N. (2023, August 8). K-Means Clustering Explained. *Neptune*. https://neptune.ai/blog/k-means-clustering#:~:text=K%2Dmeans%2B%2B%20is%20a,as%20possible%20from%20one%20another.