

Assignment 3: Practical Work

Stephen Barrett

MSc in Computing (Business Intelligence and Data Mining)

Institute of Technology Blanchardstown

Dublin 15 Ireland

B00037997@student.itb.ie

Table of Contents

Abstract	3
1.0 Introduction	3
2.0 Risk.csv - Rule Based Classifiers and a Decision Tree Algorithm.....	4
2.1 Dataset and its Meta Data	4
2.2 Investigating the data (Rule Based Classifiers & Decision Trees).....	5
2.2.1 Method (Decision Tree Algorithm).....	5
2.2.2 Method (Rule Based Classifiers).....	6
2.3 Results.....	7
2.3.1 Decision Tree.....	7
2.3.3 Rule Based Classifier	8
2.4 Conclusion.....	9
3.0 Clusterdataset.csv - Clustering.....	10
3.1 Dataset and Its Meta Data	10
3.2 Investigating the data (Hierarchical & K-Means)	11
3.2.1 Method (Hierarchical clustering)	11
3.2.2 Method (K-Means clustering)	13
3.3 Results.....	14
3.4 Subjective investigation	16
3.4.1 Method (Classifying Clustering Output).....	16
3.4.2 Results (Classifying Clustering Output)	17
3.5 Clustering Conclusion.....	18
4.0 SkeletalMeasurements.csv - Regression & Neural Networks.....	18
4.1 Dataset and its Meta Data	19
4.2 Investigating the data (Regression & Neural Networks).....	19
4.2.1 Method (Regression).....	19
4.2.2 Method (Neural Networks)	20
4.3 Results.....	21
4.4 Conclusion.....	24
5.0 HeartDisease - SVM Algorithms & Bayesian Classifiers	25
5.1 Dataset and its Meta Data	25
5.2 Investigating the data (Bayesian Classifiers & SVM Algorithms)	27
5.2.1 Method (Bayesian Classifiers)	27
5.2.2 Method (Support Vector Machine).....	27
5.3 Results.....	28
5.3.1 Bayesian Classifiers	28
5.3.2 Support Vector Machine	29
5.4 Conclusion.....	30

Abstract

The aim of this paper is to test eight algorithms on four sets of data of different properties applying a best fit algorithm for the data set at hand. The results will be looked at and modifications to each algorithm implemented to try and improve the accuracy of the models

1.0 Introduction

This paper must investigate 4 data sets. Three of those datasets will be of a classification problem and the final dataset will be of a clustering problem [Table 1.0]. The aim of the paper is to select an algorithm which will best suit the data mining problem and provide the highest accuracy. All experiments will be using the tool rapid miner

Table 1.0: Data sets and problem types

Data Set	Type of Data	Problem Type	Algorithm Chosen
<i>Risk.csv</i>	Attributes: <i>Nominal and Numeric attributes</i> Label: <i>Polynomial Label</i>	Classification problem	<i>Rule Based Classifiers and a Decision Tree Algorithm</i>
<i>HeartDisease.csv</i>	Attributes: <i>Numeric</i> Label: <i>Binomial Label</i>	Classification problem	Support Vector Machine Algorithms & Bayesian Classifiers
<i>SkeletalMeasurements.csv</i>	Attributes: <i>Numeric</i> Labels: <i>4 different labels</i>	Classification problem for two labels	Regression & Neural Networks
<i>Clusterdataset.csv</i>	Numeric numbers	Clustering	K-Means & Hierarchical clustering

2.0 Risk.csv - Rule Based Classifiers and a Decision Tree Algorithm

Decision Trees are supervised learners that categorize data by assigning predetermined class labels to a data set. The data labels or groups are known beforehand and the data is assigned to these classes based on their attributes. A decision tree consists of a root node, a decision node, a branch and a leaf node where each record/object is eventually assigned. Decision trees work well with nominal/numeric attributes that give rise to polynomial class labels.

Rule Based Classifiers are a series of “if a condition is true, then classify it as something” statements. The aim of Rule based classifiers is to move from specific instances of an object/record to a more generalized set of rules. Rule Based classifiers can easily be converted to decision trees and as a result they are good at dealing with the same types of data as decision trees.

For this experiment, due to its nominal and numeric attributes and its polynomial class label; we will be implementing *Rule Based Classifiers* and a *Decision Tree Algorithm* on the data set Risk.csv.

2.1 Dataset and its Meta Data

The data set consists of 4117 rows which consist of 12 columns, 10 of which are attributes of polynomial and integer type, one ID field and one class label (RISK) which will be the column that this experiment is trying to predict. It appears that the marital attribute is missing for 873 rows which may affect the performance of the algorithm. Table 2.1 provides more details on the data set such as statistics for the average and mode of the data within each column, the data type of each column and the range of values within each column. Table 2.1a describes the column and shows the role of each of the attributes in the data set

Table 2.1: Risk.csv Meta Data

Name	Data Type	Statistics	Range	Missing Values
ID	integer	avg = 102059 +/- 1188.620	[100001.000 ; 104117.000]	0
RISK	polynomial	mode = bad profit (2407), least = good risk (804)	good risk (804), bad loss (906), bad profit (2407)	0
AGE	integer	avg = 31.820 +/- 9.877	[18.000 ; 50.000]	0
INCOME	integer	avg = 25580.212 +/- 8766.867	[15005.000 ; 59944.000]	0
GENDER	binominal	mode = f (2077), least = m (2040)	m (2040), f (2077)	0
MARITAL	binominal	mode = married (2089), least = single (1155)	married (2089), single (1155)	873
NUMKIDS	integer	avg = 1.453 +/- 1.171	[0.000 ; 4.000]	0
NUMCARDS	integer	avg = 2.429 +/- 1.881	[0.000 ; 6.000]	0
HOWPAID	binominal	mode = weekly (2091), least = monthly (2026)	monthly (2026), weekly (2091)	0
MORTGAGE	binominal	mode = y (3200), least = n (917)	y (3200), n (917)	0
STORECAR	integer	avg = 2.516 +/- 1.353	[0.000 ; 5.000]	0
LOANS	integer	avg = 1.376 +/- 0.838	[0.000 ; 3.000]	0

Table 2.1a: Risk.csv Data Definition and Role

Name	Definition	Role
ID	ID of Record	id
RISK	What type of risk the customer is	label
AGE	Age Of Customer	regular
INCOME	How much the customer earns	regular
GENDER	Gender of the customer	regular
MARITAL	If the customer is married or not	regular
NUMKIDS	Number of Kids the customer has	regular
NUMCARDS	Number of Bank Cards the customer has	regular
HOWPAID	When the customer is paid...monthly weekly etc	regular
MORTGAGE	Does the customer have a mortgage	regular
STORECAR	How many cars do they own	regular
LOANS	How many loans does the customer currently have	regular

2.2 Investigating the data (Rule Based Classifiers & Decision Trees)

2.2.1 Method (Decision Tree Algorithm)

After importing in the dataset Risk.csv, A nominal building block of type *X-Validation* is added to the project and connected up to the dataset [Figure 2.2.1].

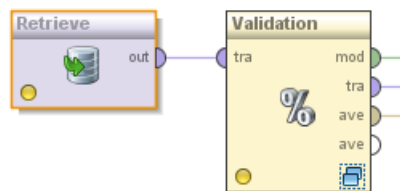


Figure 2.2.1: Risk.csv data set with nominal X-validation building block

Embedded in this building block is a *Decision Tree Operator* on the training side and *Apply Model* operator with a *Generic Performance Operator* on the testing side. The Generic Performance Operator is removed and replaced with a classification performance operator where the evaluators, accuracy¹ and classification error² are selected.

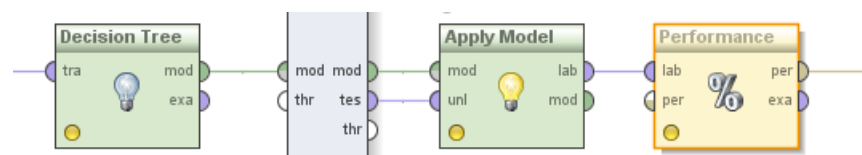


Figure 2.2.1a: Nested Operators within X-validation building block

The process is run and the results are recorded.

¹ **Accuracy** is the percentage of correct classification or records over the total number of records in the training set

² **Classification Error** is the percentage of misclassified records verses the total number of records in the training set

The following modifications to the process were then tried to see if it would improve the performance; Firstly the criteria for splitting a leaf in the Decision tree Operator were changed from gain ratio through to accuracy [Figure 2.2.1b] with the minimal split, minimal gain and confidence varied for each criteria.

The criteria are algorithms that determine the best attributes to split the data by. Minimal split is defined as the minimum amount of records that have to be in a split for example if the minimal split was set to four, four records would have to be assigned within the new split for it to be considered doing otherwise no splitting if the data occurs. Minimal gain is the gain necessary for each split to occur for each criterion, for example setting the criterion to information gain and the minimal gain to .1 would mean that information gain was increase by 10% before a splitting of the data is considered

Pre pruning was removed and all pruning was removed to see if it had any tangible effect on accuracy

Decision Tree	
criterion	gain_ratio
minimal size for split	information_gain
minimal leaf size	accuracy
minimal gain	0.1
maximal depth	20
confidence	0.25
number of prepruning alternativ...	10
☐ no pre pruning	
☐ no pruning	

Figure 2.2.1b: Criterion for splitting/merging leaf nodes

Finally, as noticed on the cursory examination of the data, it appears there were 873 missing records for the field Marital. Adding a new operator to the project called *Select Attributes*, all attributes were selected apart from Marital to see if removing this field had any effect on the accuracy of the model.

2.2.2 Method (Rule Based Classifiers)

The setup is identical to that of the Decision tree algorithm (2.2.1) without the *Select Attributes* operator. Within the *X-Validation nominal building block* we removed the *decision tree operator* and replaced it with a *Rule Induction Operator* [Fig 2.2.2]. The process was then run

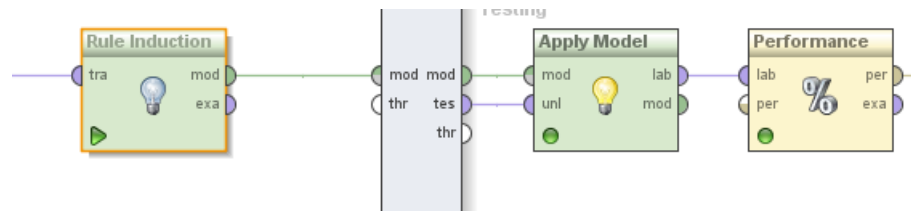


Figure 2.2.2: Nested Operators within X-validation building block

After the process was run, The Rule base operator was modified to try and improve performance by varying the criterion upon which the rules determined the split in the data. Criteria selected included accuracy and information gain. The minimal prune benefit was reduced from 25% to 10%. The operator *Select Attributes* was introduced to select all attributes except the marital attribute to see if this had any effect.

2.3 Results

2.3.1 Decision Tree

It appears to get the most accuracy from the model the best configuration was to base the splitting criterion on gain ratio and to enable pruning of the tree. Setting the minimal size of a node for a split to four and the minimal gain to at least 10% seemed to be the most optimal. Confidence is most optimal at a minimum at 25%.

The confusion matrix generated by the model predicted an accuracy of 67.68% of the true good risk classification records it did find, but only managed to find 66.42% of actual good risk records. For Bad loss it predicted a good 84.98% of the records that it managed to find but only managed to find 38% of those records. The model was good at finding bad profit as it managed to classify 76.15% of the records it did find correctly and found 92.44% of them.

The overall accuracy of the Decision Tree [Fig 2.3.1] was 75.39% +/- 2.28%

Table 2.3.1: Confusion Matrix for Risk.csv generated by decision tree

	true good risk	true bad loss	true bad profit	class precision
pred. good risk	534	118	137	67.68%
pred. bad loss	16	345	45	84.98%
pred. bad profit	254	443	2225	76.15%
class recall	66.42%	38.08%	92.44%	

As can be seen from Fig 2.3.1, it appears income is the attribute that splits the data the best using gain ratio. None of the leaves in this case give a clean leaf. The size of the columns in each of the leaves tells the researcher how many attributes have been assigned to a leaf. The colour coding indicates the distribution of the records. A single colour code in a leaf is optimal but as can be seen [Fig 2.3.1] all leaf nodes have multiple colours

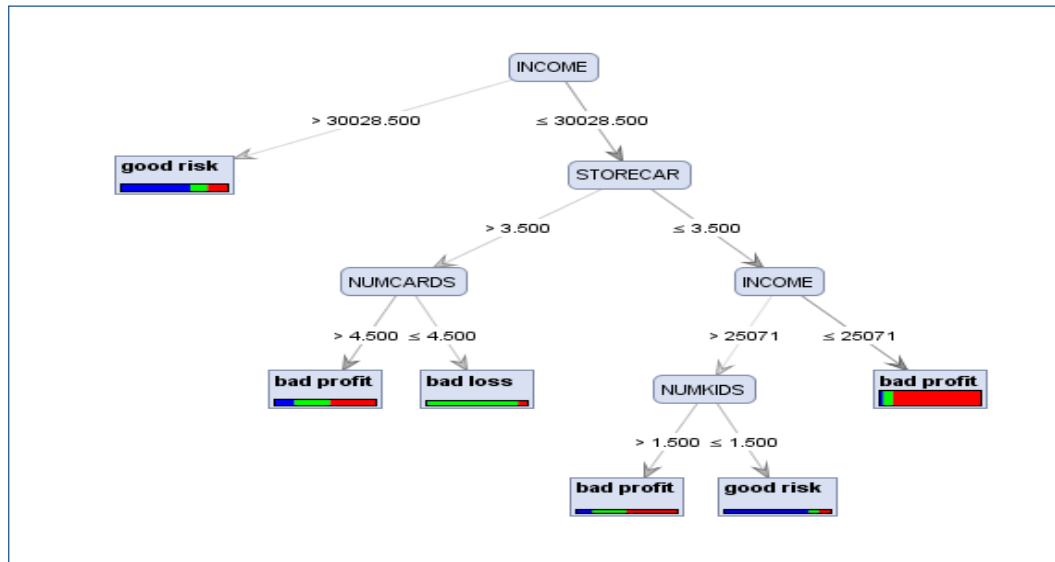


Figure 2.3.1: Generated Decision Tree for Risk.csv data

It appeared varying the criteria had little or no positive change in the accuracy. Changing the criteria on how to split the nodes resulted in gain ratio achieving the highest accuracy. Removing Pruning and pre-pruning had a negative impact on the tree. Removing the Marriage column had no impact on the results. The main change that was achievable was by changing the number of validations in the x-validation operator from 10 to 20 which raised the accuracy by almost 1 %.

Table 2.3.2: Changing Criterion split

Criterion	Accuracy
Gain Ratio	accuracy: 75.39% +/- 2.79% (mikro: 75.39%)
Information Gain	accuracy: 74.62% +/- 3.17% (mikro: 74.62%)
Gini_index	accuracy: 65.29% +/- 4.23% (mikro: 65.29%)
Accuracy	accuracy: 58.47% +/- 0.25% (mikro: 58.46%)

2.3.3 Rule Based Classifier

The confusion matrix generated [Table 2.3.3] for the Rule Based Classifier predicted an accuracy of 64.80% for the good risk classification records it did find. It only found 67.79% of the actual records. For true Bad loss it predicted roughly 69.09% of the records but missed 54.86% of the records it was supposed to find. Similar to the decision tree, the model was good at finding bad profit as it managed to classify 78.76% of the records it did find correctly while finding 87.83% of them.

With all of the modifications to improve accuracy, the overall accuracy of the model was 74.52% +/- 3.02%

Table 2.3.3: Confusion Matrix for Risk.csv generated by Rule Based Classifier

	true good risk	true bad loss	true bad profit	class precision
pred. good risk	545	133	163	64.80%
pred. bad loss	53	409	130	69.09%
pred. bad profit	206	364	2114	78.76%
class recall	67.79%	45.14%	87.83%	

It appears from the results that using information gain as opposed to accuracy as the criteria for splitting the data results provides a better accuracy for the model [Table 2.3.4]. Reducing the minimal prune benefit from 25% to 10% increased the accuracy by almost a percentile. Increasing the validations from 10 to 20 in the X-validation block also helped increase the accuracy. By removing the attribute marital, due to its missing values the models accuracy increased by almost a percentile.

Table 2.3.4: Accuracy comparison for criteria accuracy and Information Gain

Criterion	Accuracy
Accuracy	accuracy: 70.68% +/- 2.30% (mikro: 70.68%)
Information Gain	accuracy: 72.12% +/- 2.60% (mikro: 72.12%)

Table 2.3.4a: Additional Modifications and percentage increase

Action	Benefit
Reducing the Minimul Prune Benifet	accuracy: 73.23% +/- 2.55% (mikro: 73.23%)
Increased X-Validation from 10 to 20	accuracy: 73.94% +/- 3.16% (mikro: 73.94%)
Removed marital attribute	accuracy: 74.52% +/- 3.02% (mikro: 74.52%)

2.4 Conclusion

Both the Decision tree and the Rule based Classifier were excellent at finding the majority of the true bad profit and classifying them correctly. Both classifiers were good at finding the true good risk and classifying them correctly. Both classifiers however were very poor in correctly finding all the records that were actually belonged to true bad loss. It appears that the decision tree is still marginally better overall than the rule based classifier.

If presented with this model as a business user, I would disregard its classification for true bad loss. Further investigation would need to be undertaken to spot more true bad loss records, perhaps by using another algorithm such as neural networks or support vector machine where the three labels could be converted into 3 binary columns.

Alternatively, perhaps another algorithm could be used to make the classification easier such as allowing a clustering algorithm to cluster the data prior to applying the decision tree or rule based classifier, this may improve performance.

It was also noted in this experiment that by outputting the model when using Rule based classifiers that the results were slow to compute, in several cases taking just under two hours

3.0 Clusterdataset.csv - Clustering

Clustering classifies data into groups based on the similarity of attributes within the dataset i.e. if you had a group of cars, it could cluster the cars based on colour or make. It is an unsupervised learner. Unsupervised learners are algorithms where the classification groups are not known in advance and the groups are generated based on the properties of the data that the algorithm is applied to. For the purposes of this experiment we will be implementing *K-Means* and *Hierarchical clustering* on the dataset Clusterdataset.csv.

K-means divides up a large set of data X (data) into a smaller number of clusters (K) with the amount of clusters (K) being specified by the user in advance. The algorithm randomly chooses the position of the centre of each of the clusters and assigns rows of data to each cluster based on how close the attributes of the row are to the attributes of the mean. Once all rows of data have been assigned to a cluster the mean is recalculated and the data redistributed again to the appropriate clusters. This process repeats itself continuously until the mean of the clusters no longer needs to be repositioned, at which point the algorithm terminates.

Hierarchical clustering takes data and divides it up into a tree like structure. There are two types of hierarchical clustering; top down and bottom up.

For *Bottom Up (agglomerative clustering)* every record is given its own cluster. Clusters are merged based on them having the smallest distance between them.

For *Top Down* all records are placed into a single cluster which is subdivided based on how far the distance is between neighbours. This process continues until either every record has its own unique cluster or until some sort of stopping condition has been met (cluster size, number of clusters etc.)

There are numerous splitting /merging algorithms to determine if a cluster should be merged or split, including *Single Link*, *Average Link* and *Complete link*. These will be looked at in more detail later in the paper.

K-Means works well with globular well defined data. Based on our cursory look at the data in section 3.1, it appears as if there are five such clusters. Hierarchical clustering has been shown to be superior to other models at creating generalized models and so will be used to confirm optimal cluster size.

3.1 Dataset and Its Meta Data

The one thousand row data set in Clusterdataset.csv is randomly generated with three attribute columns of the type real. The Meta data [Table 3.1] which shows the names of the columns, the type of data each of the columns has, statistical information on the average within each column, the range of values contained within the columns and if there are any missing values in any of the columns.

Table 3.1: Clusterdataset.csv Meta Data

Name	Type	Statistics	Range	Missing Values
att1	real	avg = -0.413 +/- 2.888	[-7.699 ; 8.729]	0
att2	real	avg = -0.301 +/- 2.721	[-6.196 ; 7.718]	0
att3	real	avg = 0.007 +/- 3.135	[-9.172 ; 7.286]	0

The first step is to plot the data in a scatter plot to see if there are any natural patterns that seem to be visible to the human eye. This was achieved by plotting the data in rapid miner using a scatter 3D plot where each of the attributes was applied to the x,y and z plane. From examining the diagram it appears that there are 5 natural globular clusters) [Fig 3.1], [Fig 3.1a]

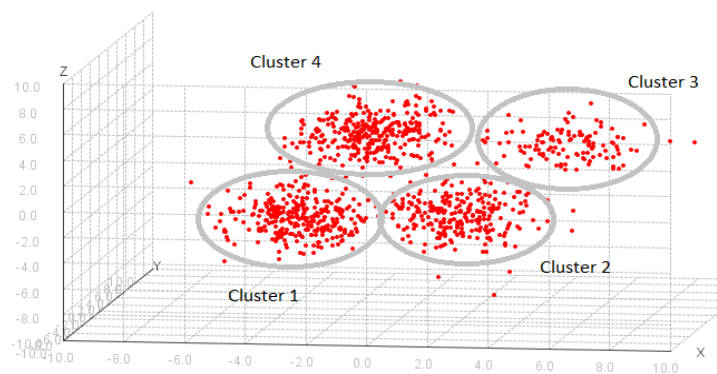


Fig 3.1: Clusterdataset.csv Mapped On 3D Scatter Plot (4 Globular clusters)

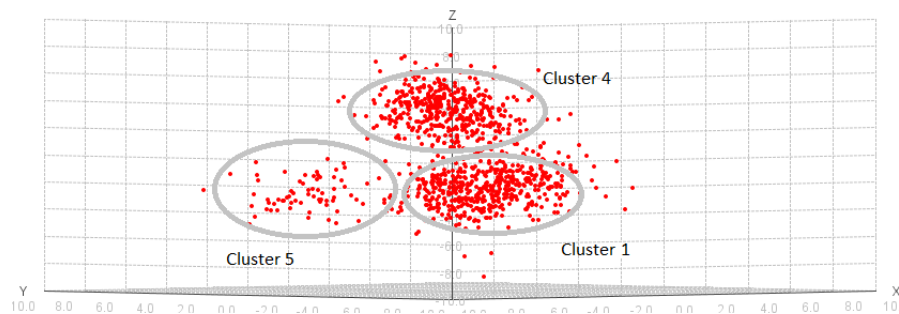


Fig 3.1a: Clusterdataset.csv Mapped On 3D Scatter Plot

3.2 Investigating the data (Hierarchical & K-Means)

3.2.1 Method (Hierarchical clustering)

After importing the data (Clusterdataset.csv) we apply an *agglomerative clustering* (bottom up) [Section 3.0] clustering algorithm to the dataset. The output is fed into a *Flatten Clustering operator* which allows the user to manually specify the number of clusters the data set should be segmented into until an optimal value is chosen [Fig 3.2.1].



Fig 3.2.1: Connecting agglomerative clustering with Flatten Clustering operators

In order to avoid having to manually change the value of the *Flatten Clustering* operator after each run until an optimal value is found, a *Loop Parameter Operator* is introduced and the *Clustering* and *Flatten Clustering* operators are nested in it. *Loop parameter* allows the user to loop through any parameters of any operators nested within it. [Fig 3.2.1a]

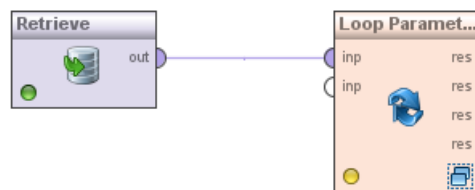


Fig 3.2.1a: Introducing Loop Parameters operator to the data set

Finally an evaluator of the model is necessary to check the accuracy of the clusters and for this we use two evaluators: *Performance Distribution*³ and *Performance Density*⁴. We connect up the evaluator operators as shown in 3.2.1b. The *Performance Density* requires a distance measure for an input which has to be manually selected by the user. In order to provide this number we need to add the operator *Data to Similarity*⁵ and connect it to *Flatten Cluster* and the *Performance Density* measure. A *log file* operator is added so we can compare the number of clusters with performance per run of the operator.

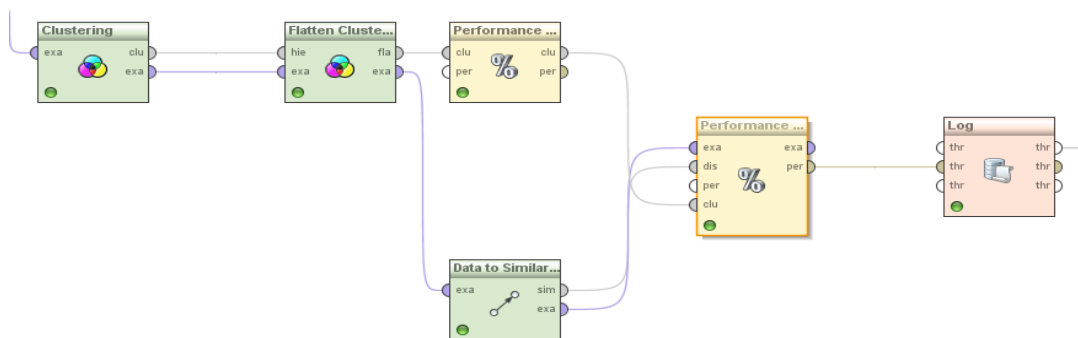


Fig 3.2.1b Nested Operators within Loop Parameters Operator

³ **Performance Distribution** looks at the distribution of objects within each cluster and reports on how evenly the distributions are. Aim is to get as close to zero as possible

⁴ **Performance Density** checks the density of objects within each cluster, if the density is similar among all clusters it will return a good score. The aim is to be as close to zero as possible

⁵ **Data to Similarity** calculates distance between every row of data and every other row of data

Additionally we tweaked the hierarchical algorithm by editing the *clustering operator* to run the process using merging/splitting algorithms, *single link*⁶, *complete link*⁷ and *average link*⁸.

3.2.2 Method (K-Means clustering)

For this experiment the dataset Clusterdataset.csv is imported into the process. The clustering algorithm K-Means is embedded with the performance measurement *Cluster Distance Performance* in the *Loop Parameters operator*. The loop parameter is used to loop through the values of K in the K-Means Operator [Fig 3.2.2]

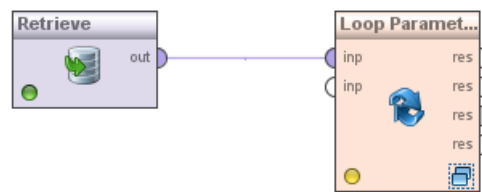


Fig 3.2.2 Introducing Loop Parameters operator to the data set

The *Cluster Distance Performance* measurement is set to *Davies Bouldin*⁹. A log operator is embedded in the loop operator to record the number of clusters and the *Davies Bouldin* metric.

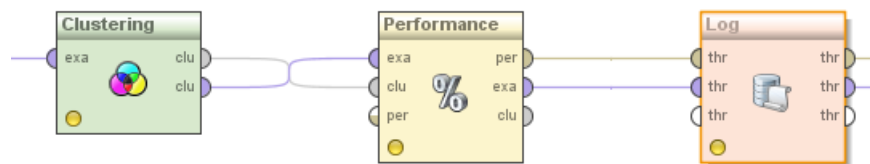


Fig 3.3.2b Nested Operators within Loop Parameters Operator

⁶ **Single Linkage** uses local decision making (doesn't take into account the whole cluster) as it uses the two closest points between data points in separate clusters. It doesn't handle noise well but is useful for irregular shaped data

⁷ **Complete Linkage** uses non local decision making (takes the whole cluster into consideration) as it uses the distance between the two furthest points in two separate clusters to determine split points. It is good with small globular clusters but can sometimes be susceptible to outliers

⁸ **Average Linkage** uses the average distance between pairs of data points between two clusters to determine split points. It is biased towards globular data but is not as susceptible to noise

⁹ **Davies Bouldin Index** is an evaluator that gives a ratio of the inter distance between data points versus the intra distance. It

requires a cluster centre which is why in this case we use it with K-Means. The formula is given as

$$DB = \frac{1}{n} \sum_{i=1}^n \max_{i \neq j} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)$$

The top line is the average distance of all points to the centre of their respective cluster. The bottom line determines the distance between the centres of different clusters. You run this for a single cluster versus all clusters in the data set which should give a range of ratios, the top ratio is taken. This is then repeated for all the clusters in the data set. All the ratios are added up and then divided by the number clusters

3.3 Results

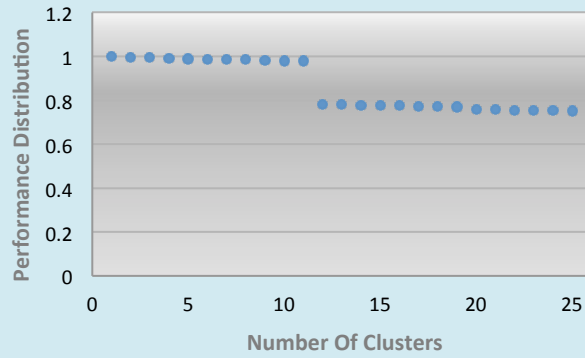
When running the algorithm with single linkage, it appeared the optimal number of clusters found by the algorithm appeared to be 12 [Table 3.3.2] for *both density and distribution performance* before the graph starts to converge i.e. the performance didn't improve dramatically. This differs somewhat when we implement complete linkage, where the optimal number of clusters appeared to be 8 [Table 3.3.2]. For the final test with average linkage the optimal value for the number of clusters was 5 [Table 3.3.2].

When applying the K means algorithm and measuring the performance using *Davies Bouldin Index* it can be seen that the optimal number of clusters is five i.e. lowest value of the *Davies Bouldin* metric occurs when the number of clusters is five [Fig 3.3.2].

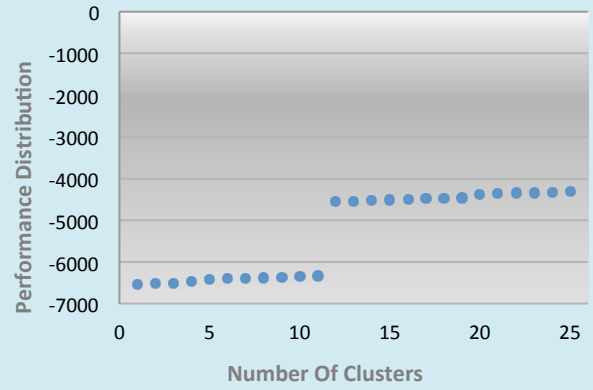
Table 3.3.2: Output and performance Hierarchical Clustering

Single Linkage

Performance Distribution Vs Number Of Clusters

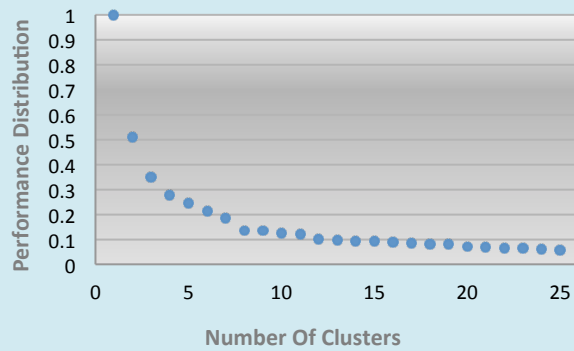


Density Distribution Vs Number Of Clusters

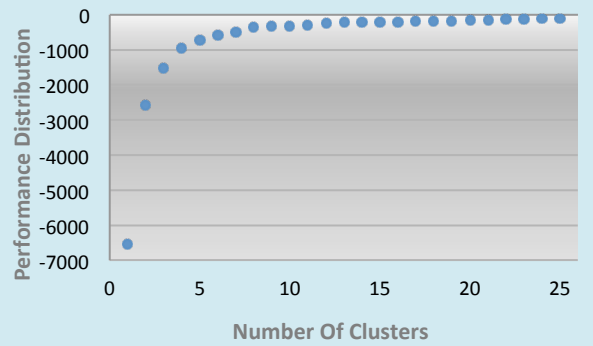


Complete Linkage

Performance Distribution Vs Number Of Clusters

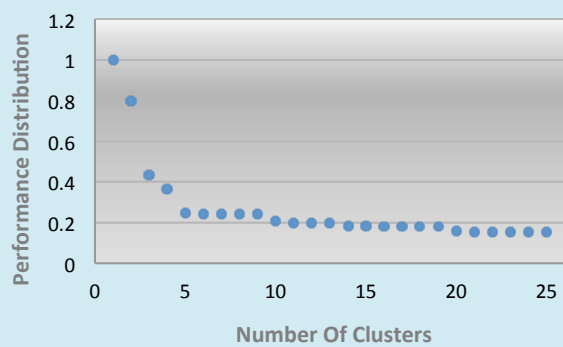


Density Distribution Vs Number Of Clusters

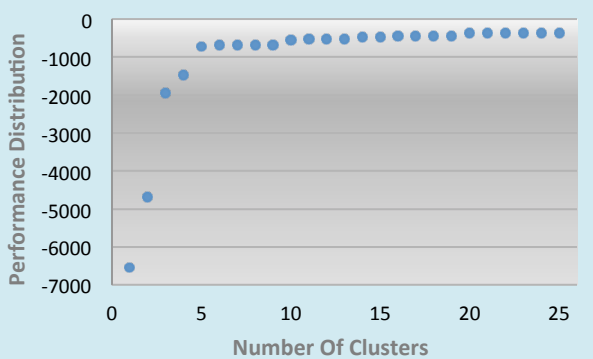


Average Linkage

Performance Distribution Vs Number Of Clusters



Density Distribution Vs Number Of Clusters



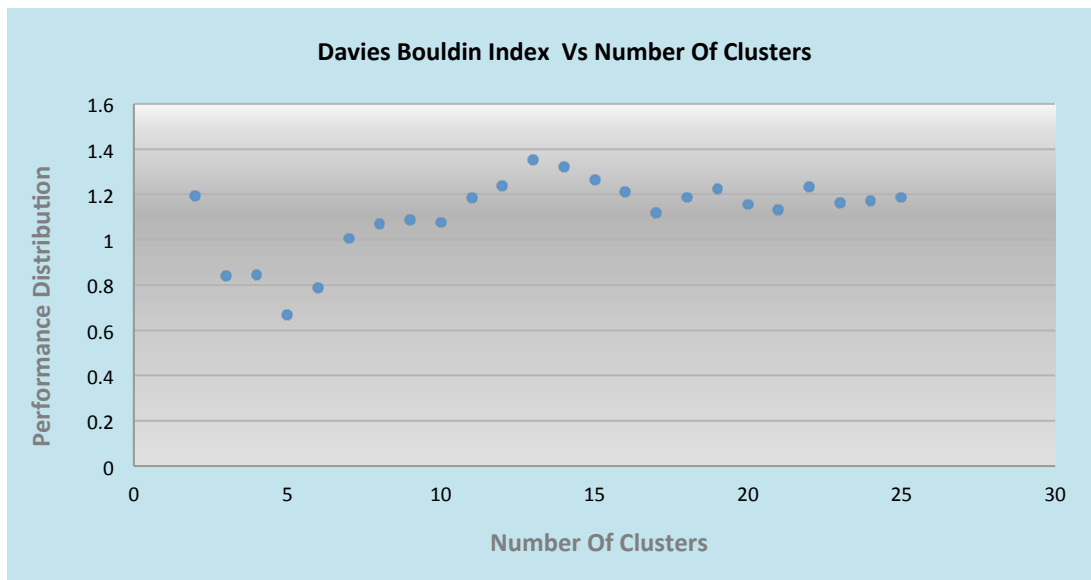


Fig 3.3.2: Output and performance K-Means Clustering

3.4 Subjective investigation

It is possible to use a number of tools to represent the data upon which a domain expert could then immediately confirm the accuracy of the clustering. Some of these tools include radar plots, circle cluster visualization tools and network diagrams. For the purposes of this paper we will be using a decision tree to help define the membership of each cluster. Clustering algorithms output cluster IDs in their results which for this experiment will be taken and set as a class label for the decision tree. The process will then be run and a decision tree diagram will be outputted.

3.4.1 Method (Classifying Clustering Output)

A k-means algorithm is implemented on the data set where using the *Select Attributes* operator the relevant information (att1, att2, att3 and cluster) from the output of the clustering algorithm is selected and fed into the *Set Roles* which changes the role of the attribute cluster to be a label. This label is then used to build a decision tree within a *nominal X-Validation building block*.

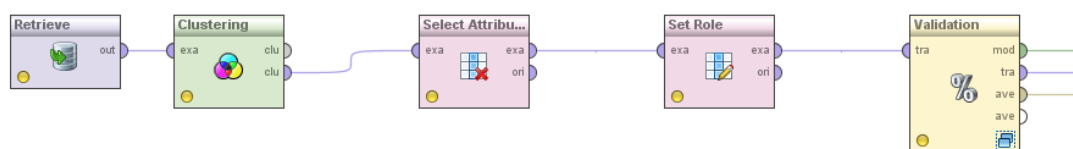


Fig 3.4.1: Classifying Clustering Output

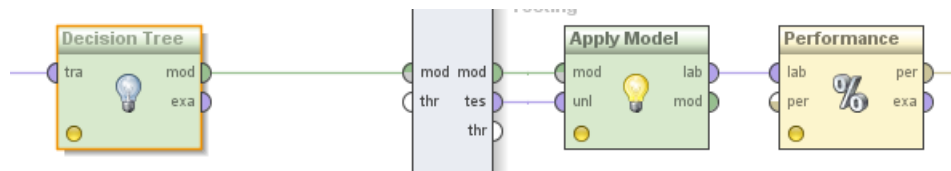


Fig 3.4.1a: Contents of X-Validation building block.

For the clustering algorithm, the number of clusters was set to 5, 8, 12 based on the previous experimental results to see which output would provide the most accurate output for the decision tree

3.4.2 Results (Classifying Clustering Output)

It appears that the best number of clusters to feed into the decision tree to get the most accurate classification is five [Table 3.4.2]. A decision tree with five clusters is generated. It is very difficult to provide subjective analysis of the tree due to the random generation of the data. There is no logical context so no conclusions can really be drawn only to say that all clusters appear to have been cleanly classified and that five clusters appear to be the optimal amount for this data.

Table 3.4.2: Performance of Classification with different variants of K

K	Accuracy
5	98.25
8	80.5
12	89.55

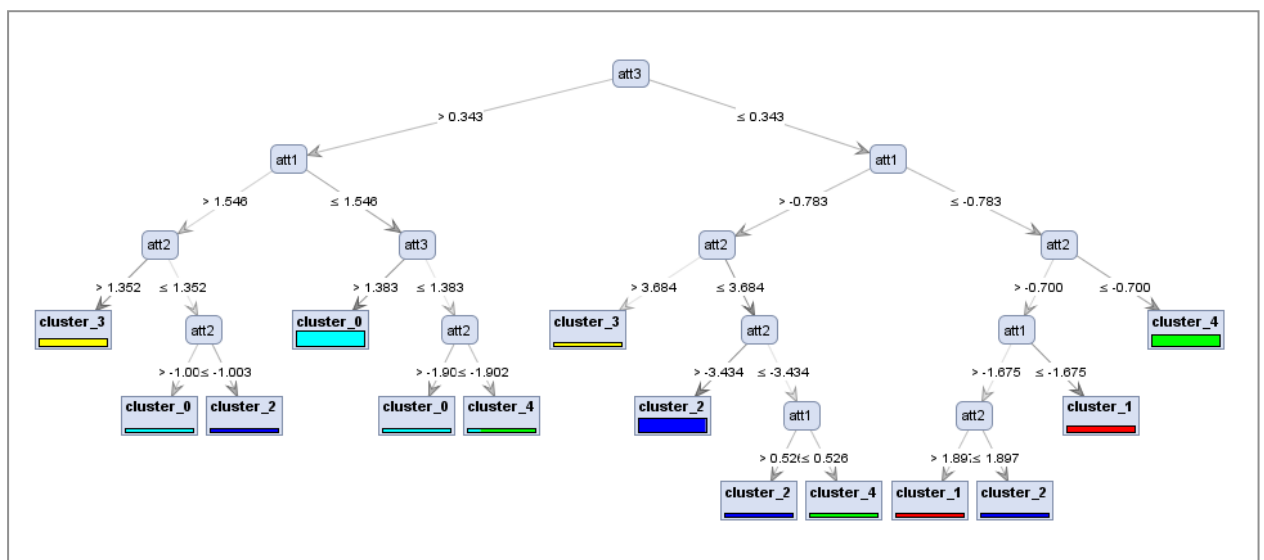


Fig 3.4.2: Generated Decision Tree where K is 5

3.5 Clustering Conclusion

Due to the lack of context of the information it is difficult to do subjective analysis to see if the clusters created are meaningful or not; however based on the experiments above, it appears that the optimal number of clusters for the dataset Clusterdataset.csv is five. Single linkage and complete linkage produced the answers twelve and eight respectively but this can be explained away by the fact that single linkage and complete linkage are susceptible to noise and outliers. Average linkage is more robust against noise and so is a better measurement in this case. Using k-means with Davies Bouldin Index appears to confirm the conclusion of 5 being the optimal number of clusters.

4.0 SkeletalMeasurements.csv - Regression & Neural Networks

Regression is used in statistics to predict numerous continuous variables. It is used to make a prediction (dependant variable) based on how the independent variables of the object change. The simplest form of regression is called linear regression which uses the formula of a straight line [Figure 4.0] to calculate the label variable based on the attributes of the dataset.

Table 4.0 Formula for Linear Regression

Linear Regression	Symbol Explanation
$Y_i = A + Bx_i + E$	Y_i = is the value you are trying to predict in row “i” of your data set A = is the starting point of the line (the intercept) B = The slope of the line (the co-efficient) E = the error/correction rate (other factors) that is used to correct

Neural Networks try to mimic how the neurons of the brain operate. It consists of input neurons, hidden neurons and output neurons. Each input is assigned a neuron and each output class is assigned a neuron. Neural networks receive an input and estimate an output. It compares its estimated output with that of the actual output of the data set and feeds this error rate back into the neural network using weights¹⁰ adjusting it for the error. This process is recursive until the error rate is at a low acceptable rate or is unchanging.

For the data set SkeletalMeasurements.csv we implement Regression & Neural Networks because they handle numeric attributes and numeric class labels well

¹⁰ **Weights** – Each input attribute has a weight known as an input weight and each neuron has a weight called a bias weight. The bias weight will be the same for all neurons in a single row of neurons

4.1 Dataset and its Meta Data

The data set consists of nine diameter measurements of skeletal parts of the human body. The measurements were taken from 247 men and 260 women the majority being in their late twenties or early thirties. Each column represents a skeletal area of the body. All columns are of numerical value either of real or integer type. There are no missing values Table 4.1 gives more details

Table 4.1: *SkeletalMeasurements.csv Meta Data*

Name	Type	Statistics	Range	Missing Values
biacromial	real	avg = 38.811 +/- 3.059	[32.400 ; 47.400]	0
pelvicBreath	real	avg = 27.830 +/- 2.206	[18.700 ; 34.700]	0
bitrochanteric	real	avg = 31.980 +/- 2.031	[24.700 ; 38.000]	0
chestDepth	real	avg = 19.226 +/- 2.516	[14.300 ; 27.500]	0
chestDiam	real	avg = 27.974 +/- 2.742	[22.200 ; 35.600]	0
elbowDiam	real	avg = 13.385 +/- 1.353	[9.900 ; 16.700]	0
wristDiam	real	avg = 10.543 +/- 0.944	[8.100 ; 13.300]	0
kneeDiam	real	avg = 18.811 +/- 1.348	[15.700 ; 24.300]	0
ankleDiam	real	avg = 13.863 +/- 1.247	[9.900 ; 17.200]	0
age	integer	avg = 30.181 +/- 9.608	[18.000 ; 67.000]	0
weight	real	avg = 69.148 +/- 13.346	[42.000 ; 116.400]	0
height	real	avg = 171.144 +/- 9.407	[147.200 ; 198.100]	0
gender	integer	avg = 0.487 +/- 0.500	[0.000 ; 1.000]	0

4.2 Investigating the data (Regression & Neural Networks)

4.2.1 Method (Regression)

After importing the dataset *SkeletalMeasurements.csv* into the project the dataset is connected up to a *Select Attribute* Operator where the dataset is filtered to only include attributes and the dependant value that is trying to be predicted (weight). The filtered data is then fed into the *Set Roles* Operator where attribute weight is set as the label (dependant value). A *Numerical X-Validation* block is connected up to the *Set Roles* Operator which contains the linear regression algorithm that will be used for this data mining exercise

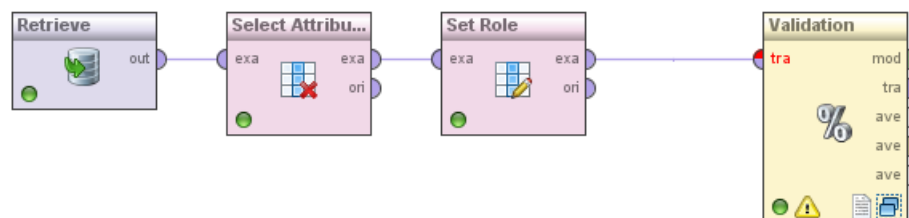


Fig 4.2.1: *Preparing the data for Linear Regression*

The *Numerical X-Validation* consists of a training side which has a linear regression model and a testing side which contains an *Apply Model* operator and a *Performance Measurement* Operator nested within it [Fig 4.2.1b].

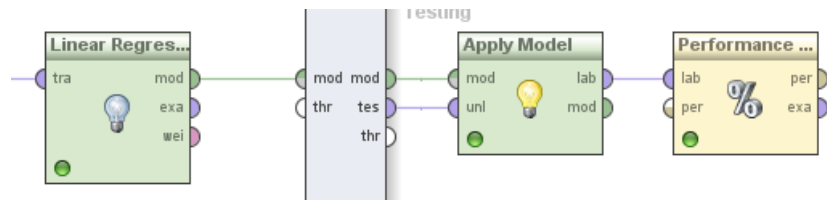


Fig 4.2.1b: Numerical X-Validation

A single modification of removing the generic *Performance Measurement Operator* and replacing it with a *Regression Performance Measurement Operator* is implemented. In the settings of *Performance Measurement Operator*, root mean squared error [Table 3.2.1] and root relative squared error are selected.

Table 4.2.1: Stages of calculating Root Mean Squared

Performance Measurement	Definition
Regression Residual	Actual Value of the prediction subtract the predicted value
Sum Of Squared Error	Sum(Regression Residual for each prediction) 2
Mean Squared Error	Average Of Sum Of Squared Error
Root Mean Squared Error	Square Root Of The Mean Squared Error



The size of Root Mean Squared Error depends on the range of the values that are being predicted. So if the Root Mean Squared Error is fifty, this says the prediction is accurate to within 50 units of the real value

Root Relative squared is very similar to root mean squared. Like Root mean squared, its aim is to measure how well the model explains the variance in the variable the model is trying to predict. It has a value between 0 and 1 but unlike root mean squared the closer you are to 0 to better your model handles the variance.

After the process finishes executing, the same implementation is configured again for the column gender by changing the *Select Attribute Operator* to remove weight column and replacing it with the gender column. Using the *Set Roles* Operator the column is converted to a label ID and process run again

4.2.2 Method (Neural Networks)

To set up the neural network to test on the data set, we set up our process in exactly the same fashion as the regression method in Section 3.2.1 except we modify the *Validation block* by removing the regression operator and replacing it with *Neural Networks operator* [Fig 4.2.2]

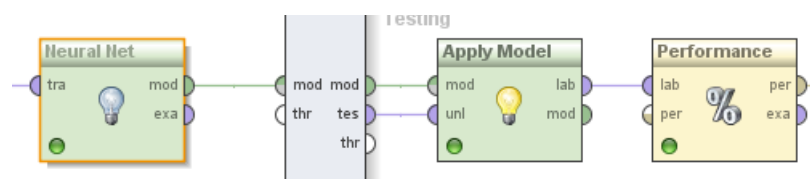


Fig 4.2.2b: Numerical X-Validation – Regression is removed and replaced with Neural Net Operator

We then run the process and record the results first for weight and then set up the *Select Attributes* and *Set Role* operators for the new label height. To modify the process in order to increase the accuracy of the model, the numbers of X-validations were increased from 10 to 20. Once that was complete 2 new hidden layers each containing 7 neurons were added to the project. The Learning Rate¹¹ was set at 10% and the Momentum¹² was set to 20%

4.3 Results

Regression - Weight

The overall algorithm performance predicts the value of weight within 4.639 +/- 0.668 (root mean squared error). The good accuracy of the model is confirmed by the low value of root relative squared error (0.352 +/- 0.048).

Table 4.3 provides a more detailed breakdown of the results. Table 4.3c provides a key to the results tables

Table 4.3: Results for predicting weight for regression

Attribute	Coefficient	Std. Error	Std Coefficient	Tolerance	t-Stat	p-Value
pelvicBreath	0.482	0.128072295	0.038192347	0.764940441	3.761559	0.0001980
bitrochanteric	0.811	0.155150099	0.051530069	0.565794028	5.23	0.0000003
chestDepth	1.713	0.117329281	0.224110317	0.498041843	14.59674	0.0000000
chestDiam	1.417	0.127826333	0.138830383	0.370899401	11.08162	0.0000000
elbowDiam	0.822	0.349475171	0.08305659	0.297922836	2.351339	0.0215610
wristDiam	1.428	0.435027764	0.127903328	0.377019526	3.282273	0.0011760
kneeDiam	1.66	0.257908318	0.118953365	0.423327963	6.438071	0.0000000
ankleDiam	-0.207	0.31271831	-0.018594678	0.391770194	-0.66087	0.5148380

Regression - Height

The algorithm predicts the value of height within 5.612 +/- 0.595 (root mean squared error). The variance in height isn't handled very well which is demonstrated by the very high root relative squared error (0.608 +/- 0.064). Table 4.3b provides a more detailed breakdown of the results

¹¹ **Learning Rate** is defined as how quickly the network learns and is set by the user in advance.

¹² **Momentum** determines how much adjustments made in the previous runs (epochs) influence weight updates in the current epoch of the neural network. It has a value between 0 and 1 where a value of 1 gives more importance to the weightings from the past epochs and a weighting of 0 gives more importance to the current epoch

Table 4.3b: Results for predicting height for regression

Attribute	Coefficient	Std. Error	Std Coefficient	Tolerance	t-Stat	p-Value
biacromial	1.378738204	0.137673667	0.108672649	0.382511245	10.01453825	0.0000000
pelvicBreath	0.560525556	0.121805063	0.044437402	0.879423938	4.601824777	0.0000055
chestDiam	-0.29529405	0.158364008	-0.028941143	0.344483047	-1.86465377	0.0749984
elbowDiam	1.809643548	0.420073207	0.182909168	0.258502517	4.307924235	0.0000206
wristDiam	0.718235308	0.52488958	0.064336425	0.333558591	1.368355051	0.2266317
kneeDiam	-0.48861985	0.297391611	-0.03500473	0.424348208	-1.64301827	0.1251552
ankleDiam	1.359436189	0.373372078	0.122315213	0.373498186	3.640969074	0.0003162

Table 4.3c: Results for predicting height for regression

Key	Description
Attribute	Column name of independent variables
Coefficient	Multiply the independent variable by the Coefficient to determine the impact on the dependant variable
Std. Error	Gives a value error on the coefficient. For example if the value is 100, then the coefficient could be wrong by + or – 100
Std. Coefficient	This is what the coefficient of the attribute would be if the attribute was scaled down to a variance of 1
Tolerance	Depicts the correlation of an attribute to other attributes. A high value (i.e. 1) signifies that the attribute is completely independent of other attributes
T-Stat	Coefficient divided by the standard error. Anything over 2 is considered a good baring on the data you are trying to predict
P-Value	Is referred to as a significance measure. It is the probability of observing the associated T Stat value when the coefficient is 0. It is calculated with the formula num of rows – num of cols .

Based on table 4.3b, the strongest attributes for predicting weight are Chest depth, Chest diameter, knee diameter and bitrochanteric in that order. These attributes all contain a high T-stat and a low P-value

It appears the strongest attributes for predicting height are the biacromial and pelvic breath columns which have a high tolerance and low P-Value. A good attribute for predicting a classifier should have a P-Value of less than 0.05 and a T-stat of greater than 2. Anything else can be considered a poor attribute for prediction of the class in the case of height wrist diameter; knee diameter and chest diameter appear to be poor variables for prediction.

Neural Networks – Height & Weight

Neural networks are a black box implementation so it's difficult to see the internal machinists of the algorithm. In Fig 4.3 for example each attribute is represented by an input neuron. The darker the lines, the more important they are for determining output. In the case of the diagram below [Fig 4.3] the biacromial (first neuron) appears to be the most important weighted attribute for determining height

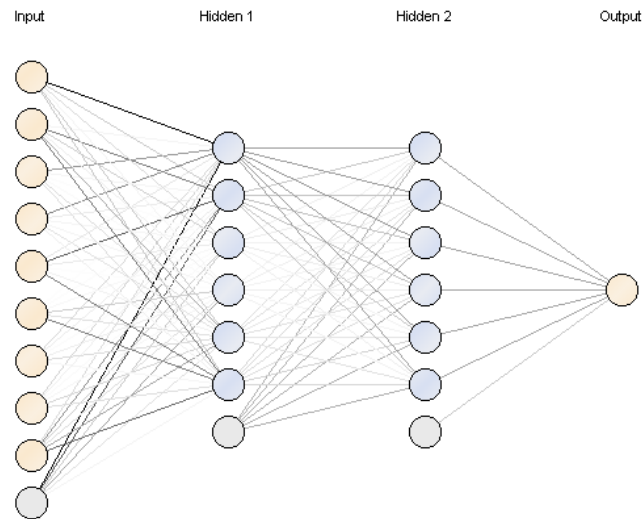


Fig 4.3: Output diagram for Neural Net operator for height. Circles represent neurons

For predicting the class height, it seems that increasing the number of x-validations from 10 to 20 improved the accuracy of the algorithm.

As expected increasing the number of neurons in the algorithm (2 banks of seven) [Fig 4.3] increased the accuracy as well as increasing the number of Epochs from the default 500 up to 750; any higher and the root mean squared error increased again [Table 4.3c].

Similar in trying to classify the height, the weight prediction can also be improved in the same fashion as the weight classification as outlined in Table 4.3d. It appears for determining weight, Chest Diameter is used to provide the highest weighting for the neurons

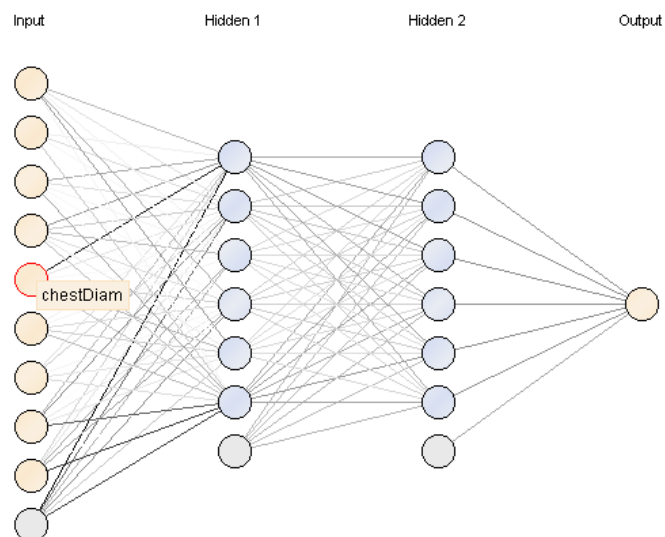


Fig 4.3: Output diagram for Neural Net operator for weight. Circles represent neurons

Table 4.3c: root mean squared error improved performance based on modifications to the process for height

Height	Performance
Basic Run	root_mean_squared_error: 6.702 +/- 0.839 (mikro: 6.753 +/- 0.000)
Increased the number of Epochs to 1000	root_mean_squared_error: 6.891 +/- 0.754 (mikro: 6.931 +/- 0.000)
Increased X-Validation from 10 to 20	root_mean_squared_error: 6.080 +/- 0.940 (mikro: 6.152 +/- 0.000)
Added 2 new layers	root_mean_squared_error: 6.508 +/- 1.310 (mikro: 6.640 +/- 0.000)
Set Learning Rate to .1 and set the Momentum to .2	root_mean_squared_error: 5.921 +/- 0.844 (mikro: 5.983 +/- 0.000)
Decreased the number of Epochs to 750	root_mean_squared_error: 5.911 +/- 0.889 (mikro: 5.980 +/- 0.000)



Table 4.3d: root mean squared error improved performance based on modifications to the process for weight

Weight	Performance
Basic Run	root_mean_squared_error: 5.126 +/- 1.106 (mikro: 5.240 +/- 0.000)
Increased the number of Epochs to 1000	root_mean_squared_error: 5.293 +/- 0.955 (mikro: 5.375 +/- 0.000)
Increased X-Validation from 10 to 20	root_mean_squared_error: 4.941 +/- 1.066 (mikro: 5.052 +/- 0.000)
Added 2 new layers	root_mean_squared_error: 4.903 +/- 0.852 (mikro: 4.972 +/- 0.000)
Set Learning Rate to .1 and set the Momentum to .2	root_mean_squared_error: 4.788 +/- 0.840 (mikro: 4.858 +/- 0.000)
Decreased the number of Epochs to 750	root_mean_squared_error: 4.715 +/- 0.840 (mikro: 4.785 +/- 0.000)



4.4 Conclusion

Based on the above algorithms, there appears to be not much to choose from them. Accuracy seems to be slightly higher for regression but where it excels above neural networks is the details it supplies on the importance of attributes to the prediction. For height it appears that biacromial and pelvic attributes are the most important and for weight, it appears the chest measurements play a key role in the prediction. For the better accuracy and the detail on how it based the prediction, I would recommend using regression above neural networks for this data set

5.0 HeartDisease - SVM Algorithms & Bayesian Classifiers

The aim of vector **machine algorithms** is to make a split in a data set (giving two classes). The vector/line splitting the data set tries to create the largest gap possible between the line (referred to as a decision boundary) and the outliers of the clusters, it is trying to separate.

Support vector machine algorithms are exceptionally good with numeric data and binary class labels which we will see from section 5.1 makes it ideal for the HeartDisease.csv

Bayesian Classifiers assigns an object or row of data to a class and then determines the probability of that row or object belonging to that class. It is based on Bayes Theorem and determines the probability of classifying an object based on the formula

(What is the probability of the attributes of an object occurring if the classification is X)
* (What is the probability of the classification of X)

(What is the probability of the attributes occurring?)

Bayesian Classifiers are good with numerical data and binomial class labels

5.1 Dataset and its Meta Data

The data set consists of a set of measurements of various health checks taken at four hospitals (Hungarian Institute of Cardiology, University Hospital, Zurich University Hospital, Basel, Switzerland, V.A. Medical Center, Long Beach and Cleveland Clinic Foundation) of which more detail can be found in Table 5.1a ([UCI Repository](#)) as well as the role each attribute will play in this data mining exercise. There are fourteen attributes with data types all of integer value. There appear to be no missing values in any of the columns. The Meta data for the average and range of the data sets can be found in table 5.1

Table 5.1: HeartDisease.csv Meta Data

Role	Name	Type	Statistics	Range	Missing Values
regular	age	integer	avg = 54.433 +/- 9.109	[29.000 ; 77.000]	0
regular	gender	integer	avg = 0.678 +/- 0.468	[0.000 ; 1.000]	0
regular	ChestPainType	integer	avg = 3.174 +/- 0.950	[1.000 ; 4.000]	0
regular	restingBloodPressure	integer	avg = 131.344 +/- 17.862	[94.000 ; 200.000]	0
regular	cholesterol	integer	avg = 249.659 +/- 51.686	[126.000 ; 564.000]	0
regular	bloodSugar	integer	avg = 0.148 +/- 0.356	[0.000 ; 1.000]	0
regular	electrocardiograph	integer	avg = 1.022 +/- 0.998	[0.000 ; 2.000]	0
regular	maxHeartRate	integer	avg = 149.678 +/- 23.166	[71.000 ; 202.000]	0
regular	angina	integer	avg = 0.330 +/- 0.471	[0.000 ; 1.000]	0
regular	oldpeak	real	avg = 1.050 +/- 1.145	[0.000 ; 6.200]	0
regular	slopeOfPeak	integer	avg = 1.585 +/- 0.614	[1.000 ; 3.000]	0
regular	flourosopy	integer	avg = 0.670 +/- 0.944	[0.000 ; 3.000]	0
regular	thal	integer	avg = 4.696 +/- 1.941	[3.000 ; 7.000]	0
regular	att14	integer	avg = 1.444 +/- 0.498	[1.000 ; 2.000]	0

Table 5.1a: HeartDisease.csv Data Definition and Role

Name	Definition	Role
age	The age of the person	regular
gender	Gender of the person (1 Male or 0 Female)	regular
ChestPainType	Type of chest pain the person is feeling Values: 1: typical angina 2: atypical angina 3: non-anginal pain 4: asymptomatic	regular
restingBloodPressure	Resting blood pressure in mm	regular
cholesterol	Cholesterol (serum) measurement in mg/dl	regular
bloodSugar	Blood sugar level after fasting (> 120 mg/dl) (1 or 0)	regular
electrocardiograph	Electro cardiograph measurement at rest Values: 0: Normal 1: ST-T wave abnormality T wave inversions and/or ST elevation or depression of > 0.05 mV) 2: Showing probable or definite left ventricular hypertrophy by Estes' criteria	regular
maxHeartRate	Max heart rate after exercise	regular
angina	Was Angina caused by exercise (1 = true; 0 = false)	regular
oldpeak	depression caused by exercise relative to rest	regular
slopeOfPeak	slope: the slope of the peak exercise ST segment slope and the peak exercise ST segment Values: 1: upsloping 2: flat 3: downsloping	regular
flourosopy	number of major vessels brought up by fluoroscopy values 1-3	regular
thal	Values: 3 = normal; 6 = fixed defect; 7 = 26eversible defect	regular
att14	Heart Disease Diagnosis Values: 1 - No <50% diameter narrowing 2 - Yes >50% diameter narrowing	Label

The aim of this experiment is to try and predict att14, i.e. if someone has heart disease or not

Note: In the UCI Repository website it claims the values of attribute 4 are 0 (No) and 1(Yes) but this isn't reflected in the data. So for the purposes of data mining, I chose 1 as No and 2 as Yes

5.2 Investigating the data (Bayesian Classifiers & SVM Algorithms)

5.2.1 Method (Bayesian Classifiers)

First we connected up the data set to the X-Validation nominal building block [Fig 5.2.1]. Then the Decision Tree Operator removed and replaced with the NaiveBayes Operator. The generic performance operator was removed and replaced with a Classification Performance operator which was set to record accuracy and classification error. The process was run and the results recorded

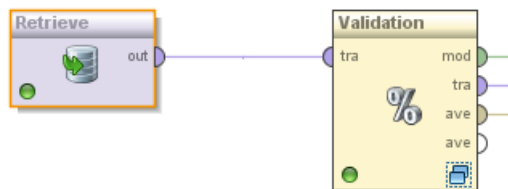


Fig 5.2.1: HeartDisease dataset with nominal X-validation building block

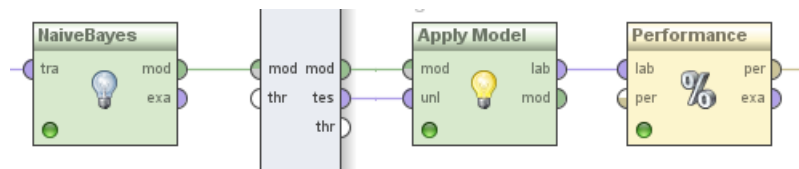


Figure 5.2.1: Nested Operators within X-validation building block

Modifications were made to the X-Validation building block varying the number of validations and sampling strategies to try and bolster accuracy.

5.2.2 Method (Support Vector Machine)

The setup for *Support Vector Machine* is exactly the same as in 5.2.1 except the *NaiveBayes* Operator is removed and replaced with the SVM operator within the X-Validation Building block [Fig 5.2.2]

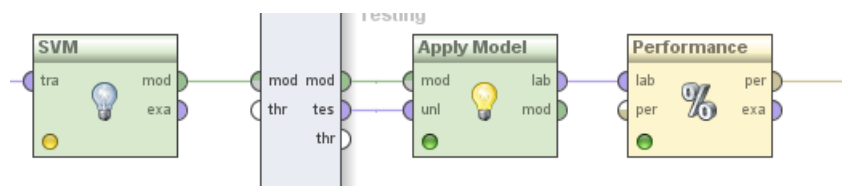


Figure 5.2.2: Nested Operators within X-validation building block

Modifications to try and improve the accuracy included: modifying the X-Validation folds and modifying the SVM operator. The two most important variables to modify in an SVM operator are the SVM type, Kernel type and the C Value.

The SVM type has only two options that can be used for classification in rapid miner C-SVC and NU-SVC.

The kernel determines if you wish to train the model using a linear classifier or a nonlinear classifier. It defaults to linear in rapid miner

The C value determines if you want the model to be a generic model or be more specific by allowing us to determine how much to allow the process to be influenced by noise. The higher the C value the more specific the model which runs the risk of being over fitted. By leaving the C value at 0, the C value is determined by heuristic methods

For this experiment we varied the SVM type and the Kernel and let C stay at zero. The results were recorded

5.3 Results

5.3.1 Bayesian Classifiers

The prediction model was excellent for classifying att14 as can be seen from the confusion matrix generated by the algorithm. It managed to classify 99 records correctly which is an 82.5% (class precision) success rate and was successfully able to find 82.5% of the records (class recall) only missing out on 21 [Fig 5.3.1].

	true 2	true 1	class precision
pred. 2	99	21	82.50%
pred. 1	21	129	86.00%
class recall	82.50%	86.00%	

Figure 5.3.1: Nested Operators within X-validation building block

The modifications to the process to varying the sampling Type and the Number of folds in X-validation seemed to be optimal at 10 validations using a sampling type of stratified giving a total accuracy for the model of 84.44% [Fig 5.3.1a].

Number Of X Validations	Sampling Type	Accuracy Performance
10	Shuffled	accuracy: 83.70% +/- 6.24% (mikro: 83.70%)
10	Stratified	accuracy: 84.44% +/- 5.19% (mikro: 84.44%)
20	Shuffled	accuracy: 82.99% +/- 8.85% (mikro: 82.96%)
20	Stratified	Accuracy: 84.18% +/- 9.63% (mikro: 84.07%)

Figure 5.3.1a: Varying X-Validation

5.3.2 Support Vector Machine

The simplest form of SVM is a linear model. The results in Fig 5.3.2 take each of the attributes and multiply them by the weight for example Age * 45.327 and then place them into a class determined by if the number is above or below a certain threshold.

Total number of Support Vectors: 141
Bias (offset): 1.941
w[age] = 45.327
w[gender] = 0.622
w[ChestPainType] = 2.718
w[restingBloodPressure] = 103.148
w[cholesterol] = 195.918
w[bloodSugar] = 0.175
w[electrocardiograph] = 0.909
w[maxHeartRate] = 118.390
w[angina] = 0.256
w[oldpeak] = 0.733
w[slopeOfPeak] = 1.326
w[flourosopy] = 0.500
w[thal] = 4.052
number of classes: 2
number of support vectors for class 2: 70
number of support vectors for class 1: 71

Figure 5.3.2: Output Model from linear Support Vector Machine

After modifications to the X-validation it was determined that the optimal setup was achieved by setting the number of folds to being 10 and the type of sampling to shuffled when leaving the SVM on default settings (linear) [Table 5.3.2]

Table 5.3.2: Results of X-Validation manipulations

Number Of X Validations	Sampling Type	Accuracy Performance	The
10	Shuffled	accuracy: 82.59% +/- 7.60% (mikro: 82.59%)	
10	Stratified	accuracy: 81.85% +/- 6.30% (mikro: 81.85%)	
20	Shuffled	accuracy: 82.14% +/- 10.76% (mikro: 82.22%)	
20	Stratified	accuracy: 81.65% +/- 11.88% (mikro: 81.48%)	

results of varying the kernel type to both different SVM types can be seen in Tables 5.3.2a and 5.3.2b respectively.

Table 5.3.2a: Results of C-SVC manipulations

C-SVC	
Kernel Type	Accuracy
Linear	accuracy: 60.74% +/- 9.83% (mikro: 60.74%)
Poly	accuracy: 66.67% +/- 10.21% (mikro: 66.67%)
RBF	accuracy: 62.22% +/- 6.79% (mikro: 62.22%)
SIGMOID	accuracy: 55.56% +/- 9.94% (mikro: 55.56%)

Table 5.3.2b: Results of NU-SVC manipulations

NU-SVC	
Kernel Type	Accuracy
Linear	accuracy: 82.59% +/- 7.60% (mikro: 82.59%)
Poly	accuracy: 81.48% +/- 9.94% (mikro: 81.48%)
RBF	accuracy: 62.96% +/- 6.83% (mikro: 62.96%)
SIGMOID	accuracy: 55.19% +/- 10.14% (mikro: 55.19%)

It appears the best set up for this algorithm is to use the NU-SVC algorithm with a Linear Kernel type and X-validation folds set to ten with sampling being set to shuffling. To provide an accuracy of 82.59%

5.4 Conclusion

It appears that both Support Vector Machine Algorithms & Bayesian Classifiers are excellent for handling numerical data attributes while trying to predict binomial class labels with both algorithms having an accuracy of over 80%. There doesn't seem to be too much to choose between the two algorithms although Bayesian Classification seems to be more accurate in its classification of the data set