Data Mining COP259: F136413

Section 1: Pre-Processing Data

We pre-process data to check the quality and format before we perform any classifiers on our data. Our dataset shows us 4 groups of saliva samples. Those with chronic obstructive pulmonary disease (COPD), Asthma, Respiratory infections, and a healthy control sample. The dataset reveals the average permittivity of saliva samples across the dataset along with the age, smoking habits, gender, and ID of the entry. Initially the dataset couldn't open in Weka, I opened the dataset within excel and removed the white space and the key. Additionally, I removed the ID column to anonymise the data and preserve the privacy of the individuals taking part in the study. It's also not necessary nor needed for our research.

Missing Entries

As we observe the dataset, there are 598 absent accounts between the imaginary and real part average. Removing these instances would leave us with an inadequate sample size to run our machine learning methods, potentially hindering the effectiveness of our models. Instead of deleting the rows for these missing entries and excluding a large quantity of demographic information from my analysis, I will leave the entries blank treating the omitted values as missing. Enabling us to still retain a large sample size to attain reliable conclusions.

Duplicate entries

Duplicate entries of data will be removed from the dataset. It's imperative we remove duplicates as it can potentially cause biases and inaccurate predictions using the machine learning methods. We remove the duplicates in a similar way, on the pre-process tab under filter and instances click RemoveDuplicates and apply the filter methods. This lowers our instances from 399 to 338, removing 61 non-unique entries.

Outliers

Another important aspect to pre-processing the data is dealing with outliers. Outliers can potentially skew the data, causing biases and inaccuracies. As our dataset is dealing with medical data, it's significant that we don't remove important identifiers that could potentially lead to a vital correlation or trend. Firstly, I apply the filter function ReplaceMissingWithUserConstant and set the numeric replacement value to 0. I apply the InterquartileRange filter to correctly identify any outliers and extreme values within the total dataset.

Normalisation & Standardisation

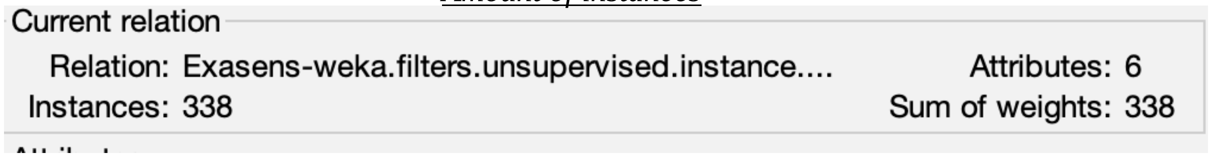
We normalise our data since our average values are distributed largely across a range of values, with a differing range of both nominal and numerical data. Normalizing will allow our data to be easily readable, comparable as the entries are scaled down to a range of 0-1. Like normalisations, standardisation is useful when there is a large range of entries which can cause issues that differ to each machine learning classifiers. Alternatively, standardising the data also changes the distribution to be gaussian, leading to a mean of zero and a standard deviation of 1 across the entire dataset. This is an important aspect to many machine learning algorithms that are optimised using normal distributions which standardizations.

If we don't standardise, interpreting the coefficients from methods like linear regression become obsolete. Furthermore, when undertaking clustering and k-nearest neighbours' models which are distance-based algorithms, ignoring standardization can cause larger entries within the dataset to dominate the metric. This can potentially skew the figures which leads to unreliable and unified data. We normalise and standardise the data by using the filter key within the pre-processing tab in Weka, clicking the unsupervised and attribute folder and applying both the filters.

Data types & setting class
It's also important our attributes that we are dealing with are the correct data types. To complete a linear regression, the attributes must all be numerical instead of a mixture between numerical and nominal. Therefore, the variables COPD is reclassified as 1, Asthma is changed to 2, respiratory infections is set to 3 and the healthy control sample is changed to 4. I made these changes using the find and replace feature within text editor. After this, I set the attribute of diagnostic to class to differentiate between the 4 different saliva types.

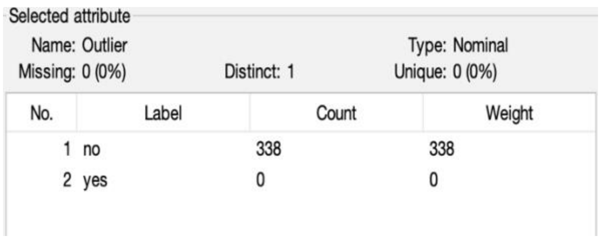
Experimental Results:

Amount of instances



```
Current relation
  Relation: Exasens-weka.filters.unsupervised.instance....        Attributes: 6
 Instances: 338                                          Sum of weights: 338
```

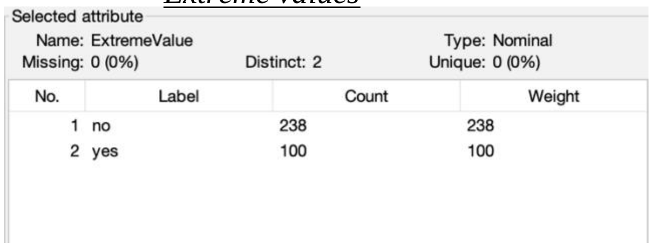
Outliers



```
Selected attribute
  Name: Outlier                         Type: Nominal
Missing: 0 (0%)        Distinct: 1      Unique: 0 (0%)
```

No.	Label	Count	Weight
1	no	338	338
2	yes	0	0

Extreme values



```
Selected attribute
  Name: ExtremeValue                    Type: Nominal
Missing: 0 (0%)        Distinct: 2      Unique: 0 (0%)
```

No.	Label	Count	Weight
1	no	238	238
2	yes	100	100

Interpretations of results
From these results we can clearly see the new level of instances has decreased to 338, from this amount we correctly found 0 outliers. This means our data will provide us with reliable and unbiased results. Moreover, we found 100 extremes although this can be disregarded

Attribute	Mean		Standard deviation	
	Before normalisation and standardisation	After normalisation and standardisation	Before normalisation and standardisation	After normalisation and standardisation
Diagnosis	2.734	2.734	1.178	1.178
Imaginary part average	-304.78	-0	25.834	1
Gender	0.417	0	0.484	1
Age	49.518	-0	18.534	1
Real part average	-458.702	-0	43.735	1

since these values come from setting the missing values to 0 to complete our interquartile range analysis. Our results have been successful normalised and standardised as seen from the table above.

Section 2: Linear regression & attribute evaluators

Linear regression is a useful statistical method that models a relationship between predictor and response variables. In our investigation, the predictor variables are age, smoking and the gender of the individual. With our response variable being the diagnosis of the salvia samples. Linear regression allows us to see the ranking of each attitude with its weighting coefficient value. The greater the value, the more influence it has on the response variable. In our response, we open go to the classify tab and choose the linear regression within the classifier selection. The test options we set the cross-validation to be 10 folds.

$$\text{Diagnosis} = -0.7305 * \text{Age} + -0.1156 * \text{Smoking} + 2.7337$$

The equation illustrates how to diagnose individuals based of the relevant formula. it takes the coefficient of -0.6199 multiplied the age, added to -0.0981 times the classification of what type of smoker the individual is. This number is the added to the error term, which in our equation totals to 2.7337. An interesting characteristic to this formula is that the gender, imaginary part, and real part weren't significant variables to be included in the prediction model of the diagnosis.

One Rule

To access the other attributes, the diagnosis variable needs to be changed back into a nominal variable. We do this by going back onto the pre-process tab, going to filter, unsupervised and attribute and selecting numeric to nominal. Once we select the indices to diagnosis, we apply the filter turning the class back to nominal. The first evaluator implemented is One Rule. This simple classification algorithm generates a single rule for every predicator within the table, then selects the lowest rule with the smallest total error as the rule. The algorithm then iterates through the attitudes, giving us an overall probability of how each of the variables will accurately predict the correct characteristic.

ReliefF

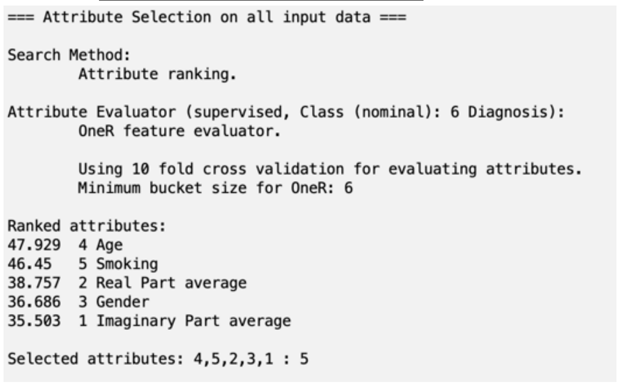
The ReliefF attribute selector works by assessing the gain of an attribute by repeatedly sampling an instance, considering the value of the given attribute for the nearest instance of the same and different class. Differently, the relief scoring method is based on identifying feature scoring pairs using the K-nearest neighbour method. The feature scores are then ranked enabling us to clearly to see the top scoring attributes.

Experimental results:

Attribute	Linear regression coefficients	One rule ranking	ReliefF ranking
Age	-0.7305	47.929	0.00926
Smoking	-0.1156	46.45	0.00796
Gender	-	36.686	0.29
Real part average	-	38.757	0.0215
Imaginary part average	-	35.503	0.0214

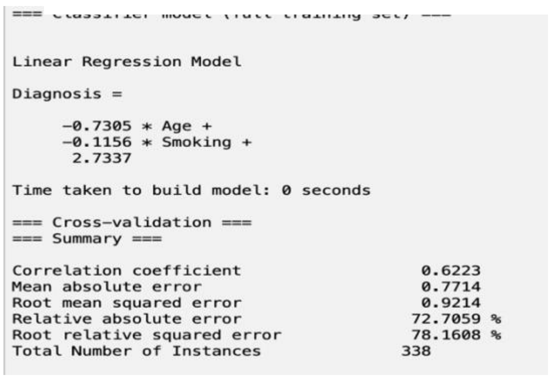
Error term	2.7337	-	-

One rule evaluator result



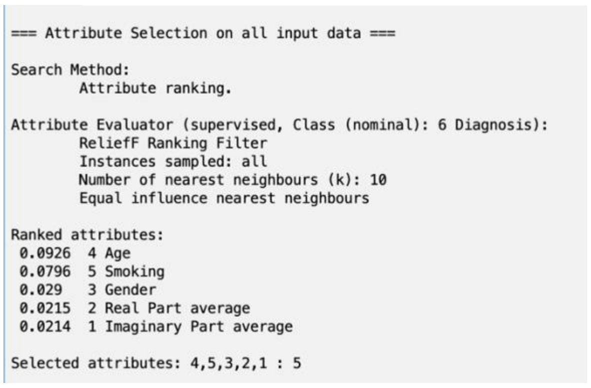
```
=== Attribute Selection on all input data ===

Search Method:
	Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 6 Diagnosis):
	OneR feature evaluator.

	Using 10 fold cross validation for evaluating attributes.
	Minimum bucket size for OneR: 6

Ranked attributes:
47.929  4 Age
46.45   5 Smoking
38.757  2 Real Part average
36.686  3 Gender
35.503  1 Imaginary Part average

Selected attributes: 4,5,2,3,1 : 5
```

Linear regression model results



```
=== Classifier model (full training set) ===


Linear Regression Model

Diagnosis =

     -0.7305 * Age +
     -0.1156 * Smoking +
      2.7337

Time taken to build model: 0 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.6223
Mean absolute error                      0.7714
Root mean squared error                  0.9214
Relative absolute error                 72.7059 %
Root relative squared error             78.1608 %
Total Number of Instances              338
```

ReliefF evaluator results



```
=== Attribute Selection on all input data ===

Search Method:
	Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 6 Diagnosis):
	ReliefF Ranking Filter
	Instances sampled: all
	Number of nearest neighbours (k): 10
	Equal influence nearest neighbours

Ranked attributes:
 0.0926  4 Age
 0.0796  5 Smoking
 0.029   3 Gender
 0.0215  2 Real Part average
 0.0214  1 Imaginary Part average

Selected attributes: 4,5,3,2,1 : 5
```

Interpretations of results

Evidently, we can conclude that the ReliefF attribute selector ranks both age and smoking significantly higher than the other attributes variables. Surprisingly, gender ranked higher than both the Imaginary and real part averages, differing to our results using One Rule where the real part average outperformed gender. Our main conclusion that we can draw is that both age and smoking are imperative attributes within our analysis however, variables such as gender, real and imaginary part average aren’t ranked highly therefore aren’t as significant. From this, I will be creating a reduced model that just removes the imaginary part average attribute. I do this by going back to the pre-process tab, clicking on imaginary part average, and selecting remove.

Section 3: Lazy IBk classifier & Naive Bayes

The lazy IBk classifier is a K-nearest neighbour supervised machine learning classification method. The method works by assuming that similar objects are near to each other, therefore classifying certain data points based on the fact on how the ‘neighbours’ are ordered. The number of neighbours used within the model is altered to find the largest correctly classified instances. I firstly click classify and choose IBk under the lazy classifier folder. Right clicking the classifier. As we increase the K value, the predictions become more stable causing accurate predictions. However, when we push the K value too far, we experience greater errors. To make the comparisons fair, I set the number of neighbours equal to 7.

Naive Bayes

Naive bayes is a supervised learning algorithm that uses bayes theorem to solve classification problems. Bayes is a conditional probability theorem that is displayed below. The algorithm takes the attributes and splits them into independent parts one for each attribute.

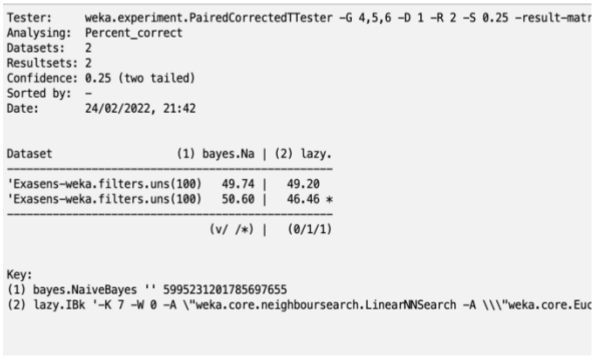
$$\mathrm{P}(\mathrm{A}|\mathrm{B}) = \frac{\mathrm{P}(\mathrm{B}|\mathrm{A})\mathrm{P}(\mathrm{A})}{\mathrm{P}(\mathrm{B})}$$

The A value describes the class, which in our investigation is the 4 classes of diagnoses. B illustrates our attributes in both our reduced and full model. We proceed with this method by navigating to the classifier tab, under classifiers choosing bayes, and clicking naive bayes.

T tests

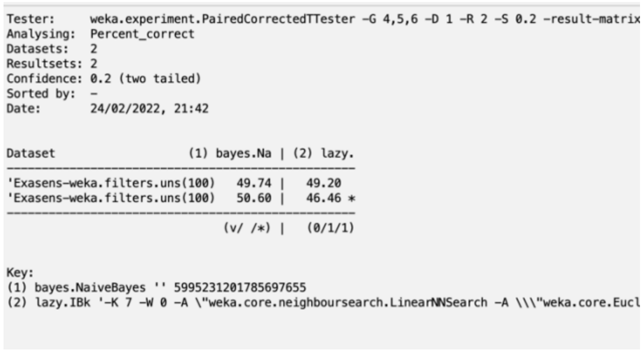
T testing is a statistical method of determining the level of significance of means within different groups. In our case, we will be comparing whether the performance classifiers implemented are statistically significant. A two tailed T test will be implemented enabling us to draw significant conclusions about the difference of the original and modified datasets, as well as our naive bayes and IBk classifiers. To access this statistical method, I go back to the Weka dashboard and open Weka explorer. Then I load both the original data and reduced datasets to run my tests. I add both my IBk and naive bayes classifiers before running and analysing my results. For this experiment I will be using a two-tailed T test at a 0.15, 0.2, 0.25 significance level. Statistical significance allows us to identify whether the attributes contribute towards the diagnosis class or if it is by chance, they're correlated.

Experimental results:



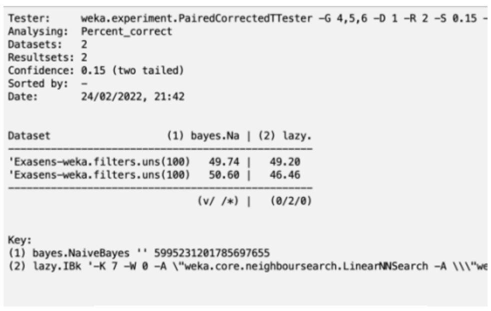
```
Tester:     weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.25 -result-matr
Analysing:  Percent_correct
Datasets:   2
Resultsets: 2
Confidence: 0.25 (two tailed)
Sorted by:  -
Date:       24/02/2022, 21:42


Dataset                   (1) bayes.Na | (2) lazy.
--------------------------------------------------
'Exasens-weka.filters.uns(100)   49.74 |   49.20
'Exasens-weka.filters.uns(100)   50.60 |   46.46 *
--------------------------------------------------
                               (v/ /*) |   (0/1/1)


Key:
(1) bayes.NaiveBayes '' 5995231201785697655
(2) lazy.IBk '-K 7 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.Euc
```

0.25 significance



```
Tester:     weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.2 -result-matrix
Analysing:  Percent_correct
Datasets:   2
Resultsets: 2
Confidence: 0.2 (two tailed)
Sorted by:  -
Date:       24/02/2022, 21:42


Dataset                   (1) bayes.Na | (2) lazy.
--------------------------------------------------
'Exasens-weka.filters.uns(100)   49.74 |   49.20
'Exasens-weka.filters.uns(100)   50.60 |   46.46 *
--------------------------------------------------
                               (v/ /*) |   (0/1/1)


Key:
(1) bayes.NaiveBayes '' 5995231201785697655
(2) lazy.IBk '-K 7 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.Eucl
```

0.20 significance



```
Tester:     weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.15 -
Analysing:  Percent_correct
Datasets:   2
Resultsets: 2
Confidence: 0.15 (two tailed)
Sorted by:  -
Date:       24/02/2022, 21:42


Dataset                   (1) bayes.Na | (2) lazy.
--------------------------------------------------
'Exasens-weka.filters.uns(100)   49.74 |   49.20
'Exasens-weka.filters.uns(100)   50.60 |   46.46
--------------------------------------------------
                             (v/ /*) |   (0/2/0)


Key:
(1) bayes.NaiveBayes '' 5995231201785697655
(2) lazy.IBk '-K 7 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"we
```

0.15 significance

Original dataset		
Performance	Correctly identified instances	Incorrectly identified instances
Naive bayes	48.2249%	51.7751%
IBk	50.2959%	49.7041%

Original dataset		
Significance level	Naive bayes	IBk
0.25	49.74	49.20
0.20	49.74	49.20
0.15	49.74	49.20

Modified dataset		
Significance level	Naive bayes	IBk
0.25	50.60	46.46*
0.20	50.60	46.46*
0.15	50.60	46.46

Modified dataset		
Performance	Correctly identified instances	Incorrectly identified instances
Naive bayes	49.4083%	50.4083%
IBk	47.6331%	52.3669%

Interpretations of results

As we look at results for the original dataset, we can see that naive bayes performed better than the IBk classifier. Despite this, neither classifier results were statistically significant at any of the levels. Therefore, we’re unable to tell whether the results performance is above or below the baseline.

Comparatively, when we look at the modified data set, we see naive bayes also performed better in this instance too. The ‘*’ symbol means there was statistical significance at the 0.25 and 0.2 level, as you lower the significance level you require more evidence to reject the null hypothesis, as we move to a 0.15 level of significance both classifiers struggle to find enough evidence to prove they’re statistically significant.

The ‘*’ also is a sign of the lazy IBk classifier being meaningfully worse at that level of significance, whereas ‘v’ means the classifier is drastically better than the other. In our analysis, we can see that the lazy IBk method is greatly worse than naive bayes at a 0.25 and 0.2 significance level. From these compelling results, we can conclude that naive bayes is the better classifier in both the modified and original dataset.

Furthermore, now we look at how each classifier performs at correctly identifying instances across the original and modified dataset. Surprisingly, the lazy IBk classifier performed better than naive bayes in the original dataset. Inferring that IBk benefits greater including the ‘imaginary part average’ as an attribute within the dataset. Comparatively, we see that naive bayes benefitted greater in the modified dataset where this attribute is removed. Allowing the classifier to effectively identify correct instances. Furthermore, I conclude that naive bayes is the greater classifier emphasized by the t-test results and performance of classification of instances across both datasets.

Section 4: Binning width & optimal classifier

As discussed in the previous section, naive bayes was proven to be the better classifier and therefore we will be carrying on using this technique in the next section.

What is Equal binning width?

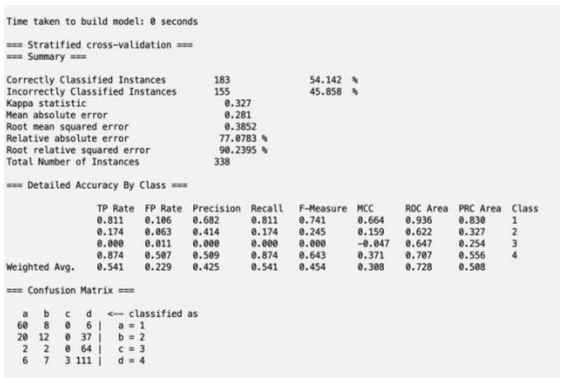
Equal binning is a method that divides the dataset within sections of ‘bins’ of equal proportions, the optimal level is between 5-20 bins. This is an unsupervised method of binning and is mainly implemented to reduce the number of potential errors within the dataset. It’s a discretization method that changes the values within the data set to a discrete form. We calculate the bin width by finding the range of our values and dividing by how many bins are necessary. Binning is useful as it increases the model response time, without suffering in model quality.

Method

We go back to the explorer section of Weka, opened our original dataset with all our prior pre-processed filters. We go to the pre-process tab, under filters we click unsupervised, attribute, then apply Discretize. Next, we right click on the classifier to open properties to find the optimal number of bins for our dataset. The bigger the dataset tends to lead to a greater number of bins. To find the optimal amount, I started with 5 bins, and increased in increments of 5 to see if it had any impact on the number of correctly identified instances.

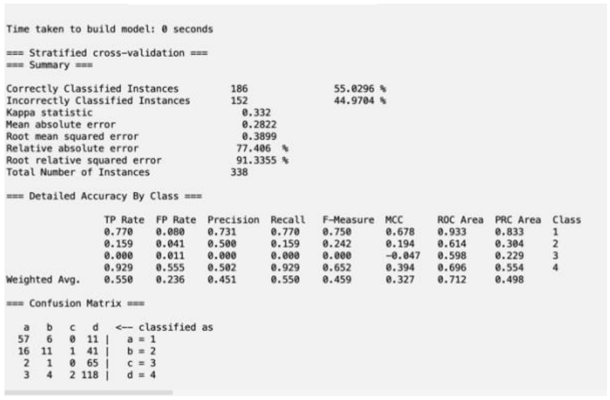
Experimental results:

10 bin results



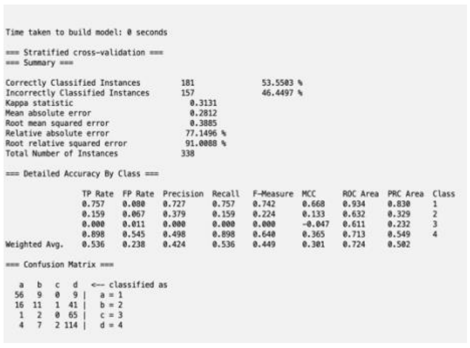
```
Time taken to build model: 0 seconds

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

Correctly Classified Instances         183               54.142  %
Incorrectly Classified Instances       155               45.858  %
Kappa statistic                          0.327
Mean absolute error                      0.281
Root mean squared error                  0.3852
Relative absolute error                 77.0783 %
Root relative squared error             90.2395 %
Total Number of Instances              338

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.811    0.106    0.682      0.811    0.741      0.664    0.936     0.830     1
                 0.174    0.063    0.414      0.174    0.245      0.159    0.622     0.327     2
                 0.000    0.011    0.000      0.000    0.000      -0.047   0.647     0.254     3
                 0.874    0.507    0.509      0.874    0.643      0.371    0.707     0.556     4
Weighted Avg.    0.541    0.229    0.425      0.541    0.454      0.308    0.728     0.508

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
  60   8   0   6 |   a = 1
  20  12   0  37 |   b = 2
   2   2   0  64 |   c = 3
   6   7   3 111 |   d = 4
```

25 bins results



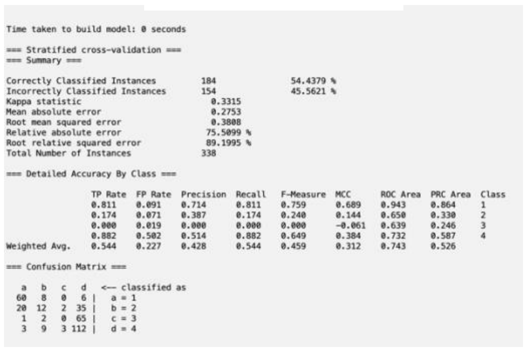
```
Time taken to build model: 0 seconds

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

Correctly Classified Instances         186               55.0296 %
Incorrectly Classified Instances       152               44.9704 %
Kappa statistic                          0.332
Mean absolute error                      0.2822
Root mean squared error                  0.3899
Relative absolute error                 77.406  %
Root relative squared error             91.3355 %
Total Number of Instances              338

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.770    0.080    0.731      0.770    0.750      0.678    0.933     0.833     1
                 0.159    0.041    0.500      0.159    0.242      0.194    0.614     0.304     2
                 0.000    0.011    0.000      0.000    0.000      -0.047   0.598     0.229     3
                 0.929    0.555    0.502      0.929    0.652      0.394    0.696     0.554     4
Weighted Avg.    0.550    0.236    0.451      0.550    0.459      0.327    0.712     0.498

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
  57   6   0  11 |   a = 1
  16  11   1  41 |   b = 2
   2   1   0  65 |   c = 3
   3   4   2 118 |   d = 4
```

20 bins results



```
Time taken to build model: 0 seconds

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

Correctly Classified Instances         181               53.5503 %
Incorrectly Classified Instances       157               46.4497 %
Kappa statistic                          0.3131
Mean absolute error                      0.2812
Root mean squared error                  0.3885
Relative absolute error                 77.1496 %
Root relative squared error             91.0088 %
Total Number of Instances              338

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.757    0.080    0.727      0.757    0.742      0.668    0.934     0.830     1
                 0.159    0.067    0.379      0.159    0.224      0.133    0.632     0.329     2
                 0.000    0.011    0.000      0.000    0.000      -0.047   0.611     0.232     3
                 0.898    0.545    0.498      0.898    0.640      0.365    0.713     0.549     4
Weighted Avg.    0.536    0.238    0.424      0.536    0.449      0.301    0.724     0.502

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
  56   9   0   9 |   a = 1
  16  11   1  41 |   b = 2
   1   2   0  65 |   c = 3
   4   7   2 114 |   d = 4
```

15 bin results



```
Time taken to build model: 0 seconds

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

Correctly Classified Instances         184               54.4379 %
Incorrectly Classified Instances       154               45.5621 %
Kappa statistic                          0.3315
Mean absolute error                      0.2753
Root mean squared error                  0.3808
Relative absolute error                 75.5099 %
Root relative squared error             89.1995 %
Total Number of Instances              338

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.811    0.091    0.714      0.811    0.759      0.689    0.943     0.864     1
                 0.174    0.071    0.387      0.174    0.240      0.144    0.650     0.330     2
                 0.000    0.019    0.000      0.000    0.000      -0.061   0.639     0.246     3
                 0.882    0.502    0.514      0.882    0.649      0.384    0.732     0.587     4
Weighted Avg.    0.544    0.227    0.428      0.544    0.459      0.312    0.743     0.526

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
  60   8   0   6 |   a = 1
  20  12   2  35 |   b = 2
   1   2   0  65 |   c = 3
   3   9   3 112 |   d = 4
```

Number of bins	Correctly identified instances	Incorrectly identified instances
5	54.4379 %	45.5621 %
10	54.142 %	45.858 %
15	54.4379 %	45.5621 %
20	53.5503 %	46.4497 %
25	55.0296 %	44.9704 %
30	53.8462 %	46.1538 %

Interpretations of results

Using naive bayes as our classifier, we surprisingly find that increasing the number of bins doesn’t always result in greater levels of correctly identified instances, instead the level of accuracy is entirely dependable on the dataset itself. As I increase the level of bins from 5-10, the level of correctly identifiable instances decreases by 0.2959%. The level of correct identifiable instances is then further decreased when we increase the level of bins from 15 to 20, we drop from 54.4379% to 54.5503%. Despite this, when the number of bins is increased to 25, we find our optimal level of bins. Scoring us a 55.0296% accuracy in correctly identifying instances within our dataset. However, as the optimal of binning is between 5-20, and the optimal is 25 we can see that there isn’t a linear relationship between increased efficiency and the number of bins.

Binning is clearly an easy way to efficiently increase the accuracy of correctly identifying instances. Without binning, our classifier correctly identified 48.2249% of instances, comparatively with discretization we can increase the quality of our classifier by up to

6.8047% with relatively no disadvantages to the method. Discretization also allows us to easily visualise the spread of large datasets relatively easy, enabling easy comparisons between multiple datasets.

Section 4: K-means clustering

K-means clustering is a simple unsupervised machine learning algorithm. As the method is unsupervised, only patterns are considered, any labelled or known outcomes are disregarded. The algorithm is based on the level of ‘k’s which represent the number of centroids needed within our dataset. A centroid describes the location of the very centre of a cluster, our data is then assigned the nearest cluster based on certain similarities. The algorithm firstly picks random centroids and performs calculations to find out the best locations for the centroids to be. The calculations are complete once the locations have been stabilized or the number of iterative calculations has been completed.

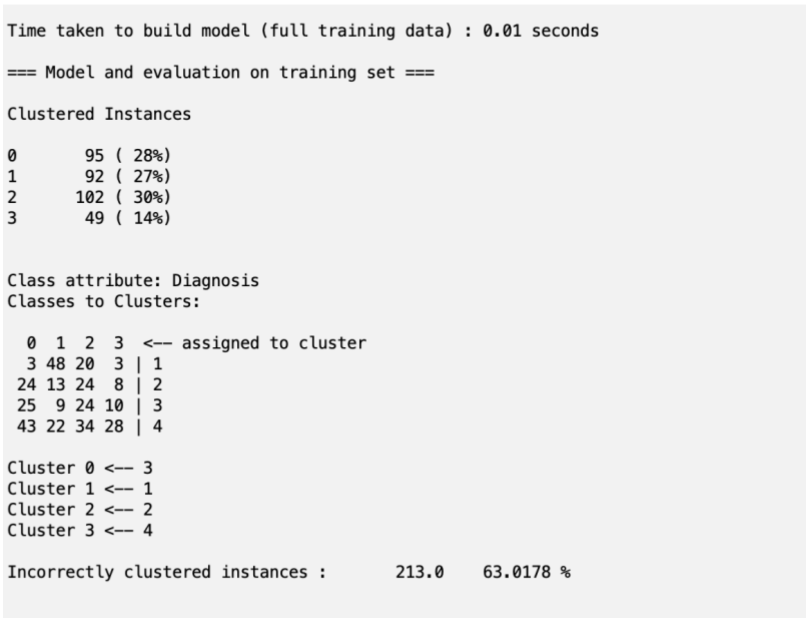
Method

Opening Weka, I open explorer, load my data, and proceed to the clustering tab. I then choose the SimpleKMeans method and select the number to the number of classes we know as 4.

Experimental results:

Cluster	Clustered Instances	Class assigned
0	95	3
1	92	1
2	102	2
3	49	4

Clustering results & ground truth



```
Time taken to build model (full training data) : 0.01 seconds

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

Clustered Instances

0       95 ( 28%)
1       92 ( 27%)
2      102 ( 30%)
3       49 ( 14%)


Class attribute: Diagnosis
Classes to Clusters:

  0  1  2  3  <-- assigned to cluster
  3 48 20  3 | 1
 24 13 24  8 | 2
 25  9 24 10 | 3
 43 22 34 28 | 4

Cluster 0 <-- 3
Cluster 1 <-- 1
Cluster 2 <-- 2
Cluster 3 <-- 4

Incorrectly clustered instances :       213.0    63.0178 %
```

Interpretations of results:

In our test options, I selected classes to clusters evaluation. This mode firstly ignores the class attributes and begins forming clusters, when the test phase begins the classes are resigned back to the clusters based on its most frequent class. The results illustrate how each cluster has been assigned to a class. Cluster 0 is assigned class 3, cluster 1 allocated class 1, cluster 3 is issued class 2 and cluster 3 is given class 4. The algorithm iterated 10 times before completing, giving a total sum of squared errors equal to 34.94 2.dp.

The confusion matrix allows us to identify the number of instances that are given to each cluster which are then attributed to the 4 classes. This gives us perspective of the ground truth, in cluster 0 identified 70 instances incorrectly, cluster 1 identified 44 instances wrongly, cluster 2 found 78 occurrences mistakenly identified and finally, cluster 3 found 21 incorrect data points. This totals 213 incorrectly clustered cases which illustrates the ineffectiveness of an unsupervised algorithm on our dataset. To gain better results specifically on our dataset, a better alternative could be the implementation of a supervised algorithm instead. Despite the lack of accurate results, clustering still is an easy and fairly simple way to evaluate how well certain datasets perform under unsupervised environments.