Enterprise Database Technologies

CA 2 – Ruairí Keogh

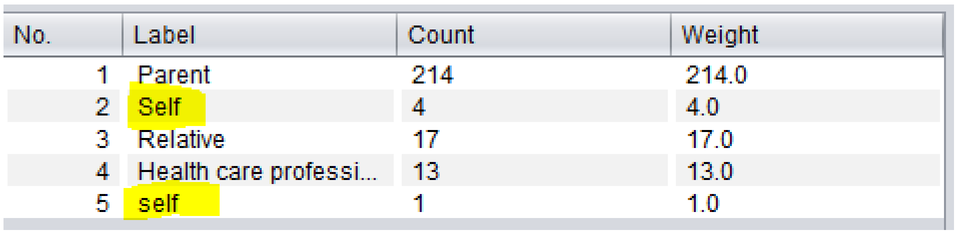
X00121581

Table of Contents

Load and Analyse the dataset 2
Preparing a number of views of the dataset 3
Attribute Selection 4
kNearestNeighbour (IBk) Classifier 7
Identify two suitable Machine Learning Algorithms 8
  Ensemble Method for Machine Learning Algorithms – Boosting 8
  Voted Perceptron 8
Evaluation of Machine Learning algorithms: 9
  ZeroR (baseline algorithm) 9
  JRip 9
  NaiveBayes 10
  SMO - Sequential Minimal Optimisation 10
  IBk - kNearestNeighbour (using k=3) 11
  J48 11
  Boosting 12
  Voted Perceptron 12
Performance of Machine Learning algorithms 13
  Results on using different data-sets 13
  Best performing algorithms 13
  Significant findings and supporting evidence 13
Final Version of the Model and Present Results 14
Section 2 15
Apriori Algorithm with minimum support 30% 15
References: 17
Appendices 18

Load and Analyse the dataset

Each of the attributes in this dataset for Autistic Spectrum Disorder Screening Data for Children were examined before the beginning of this assignment. From my investigation I was able to find that all attributes were nominal apart from "result" and "age", these are both numeric. As part of my investigation I looked into whether there was any missing values within these attributes. The result was 4 missing values in "age" and 43 missing values in both "relation" and "ethnicity". This indicates a direct correlation between the "relation" and "ethnicity" attributes as the missing values were in the same rows. The last significant finding I could see from a visual inspection was the value of "self" in the "relation" column. This was only entered once and is a misspelling of the appropriate value of "Self" with a capital "s". See figure 1 below.



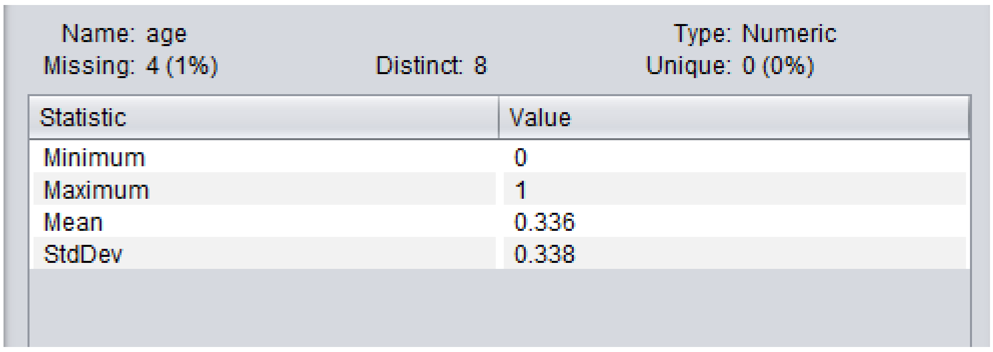
No.	Label	Count	Weight
1	Parent	214	214.0
2	Self	4	4.0
3	Relative	17	17.0
4	Health care professi...	13	13.0
5	self	1	1.0

Figure 1 – Image shows the misspelling of self, resulting in duplicated labels for relation attribute.

Preparing a number of views of the dataset

For this section I created 4 versions of this dataset saved as arff files. The first file I created was the original dataset file I obtained with the beginning of this assignment. The only change I made to this was removing the “result” attribute so that my calculations further into the assignment wouldn’t be affected.

The second file I created was a normalised version of the original. This consisted of me applying the normalise filter from weka. This created a range of 0-1 for each attribute. Zero being the minimum and one being the maximum. See figure 2.

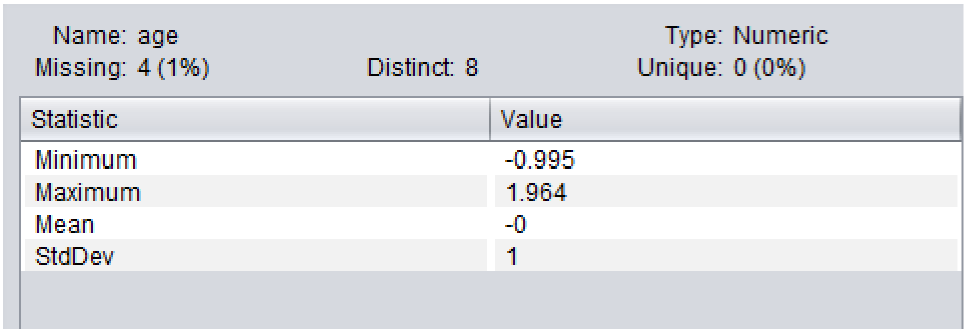


Name: age Type: Numeric
Missing: 4 (1%) Distinct: 8 Unique: 0 (0%)

Statistic	Value
Minimum	0
Maximum	1
Mean	0.336
StdDev	0.338

Figure 2 – Normalised attribute “age”.

The next file was similar to the normalised version, this was a standardised version of the original. This meant I added the standardise filter from weka and it changed the mean and standard deviation of the attributes so that the deviation was 1 and the mean was 0. See figure 3.



Name: age Type: Numeric
Missing: 4 (1%) Distinct: 8 Unique: 0 (0%)

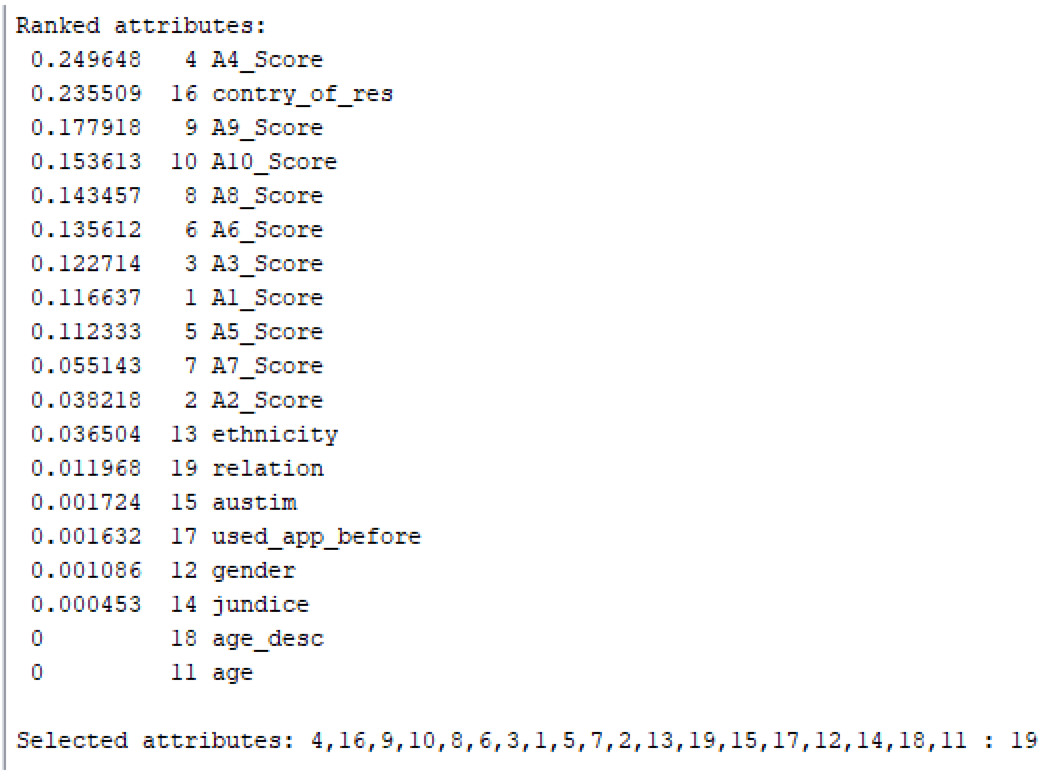
Statistic	Value
Minimum	-0.995
Maximum	1.964
Mean	-0
StdDev	1

Figure 3 – Standardised attribute “age”.

The final arff file I created was a missing data file which took the mode/average of each attribute and input it for any missing values. This meant that for “age” the value of 6 was entered, for “relation” the value of Parent was entered and for “ethnicity” white-Eurasian was entered. The final step I done for the missing file was to fix the misspelt “self” so that there was now 4 distinct labels for the “relation” attribute.

Attribute Selection

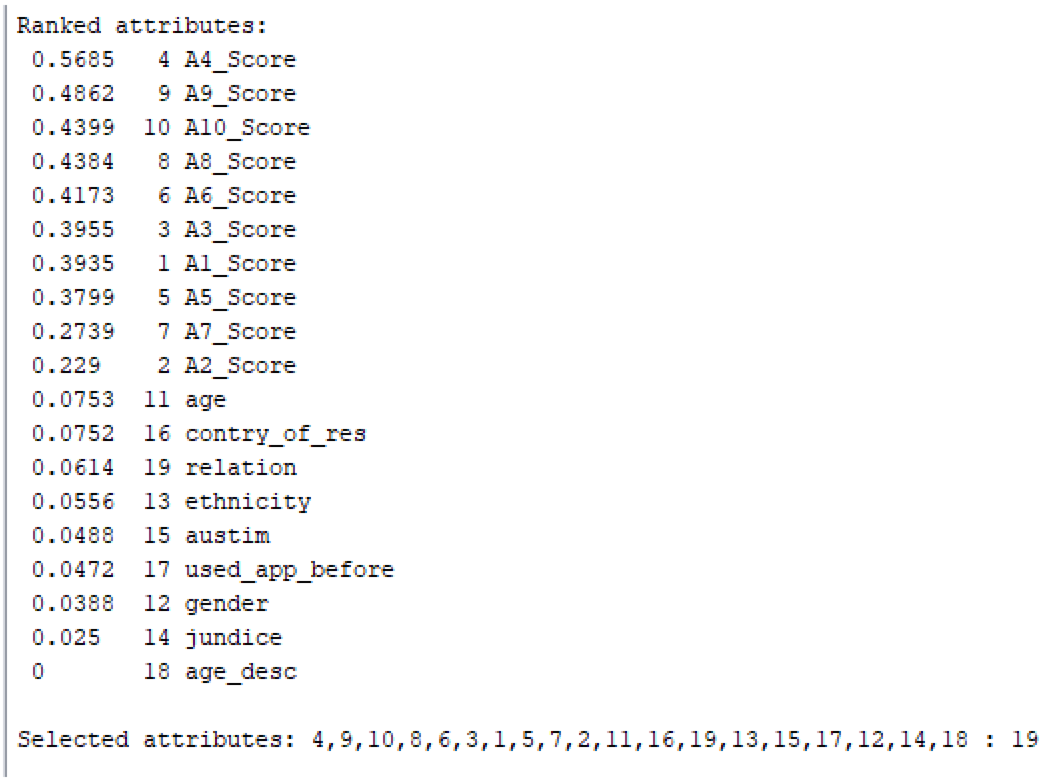
The first method of attribute selection I performed on this dataset was Information Gain Based. This calculates the value of the information provided by each attribute and ranks them. For each of the attributes ranked they receive a score based from 0-1 where 1 is maximum. To complete this method it must be done with a ranker search method. This will rank the attributes in order based on the scores they each receive. As seen from the image below (figure 4) the best attributes are "A4_Score" and "country_of_res". The worst attributes are "age" and "age_desc" which both received the minimum score of 0.



```
Ranked attributes:
 0.249648    4 A4_Score
 0.235509   16 contry_of_res
 0.177918    9 A9_Score
 0.153613   10 A10_Score
 0.143457    8 A8_Score
 0.135612    6 A6_Score
 0.122714    3 A3_Score
 0.116637    1 A1_Score
 0.112333    5 A5_Score
 0.055143    7 A7_Score
 0.038218    2 A2_Score
 0.036504   13 ethnicity
 0.011968   19 relation
 0.001724   15 austim
 0.001632   17 used_app_before
 0.001086   12 gender
 0.000453   14 jundice
 0          18 age_desc
 0          11 age

Selected attributes: 4,16,9,10,8,6,3,1,5,7,2,13,19,15,17,12,14,18,11 : 19
```

Figure 4 – Information Gain Based attribute selection method. Attributes ranked in descending order of importance.

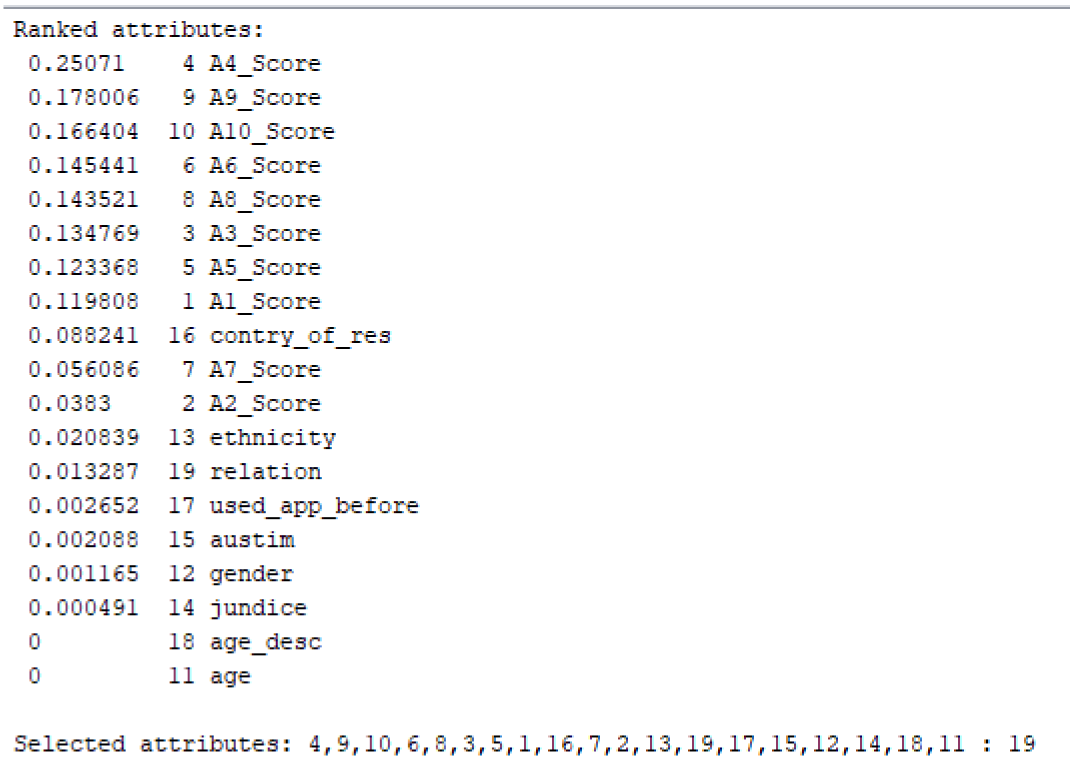
The next selection method I decided to use was a Correlation Based method. This method gave a value of correlation between each attribute and the output variable which is "Class/ASD". This must also be done using the ranker search method. The decision for the attribute selection should choose moderate to high positive or negative correlations e.g 1 or -1. Anything close to 0 should be ignored and removed from the dataset during this attribute selection. The strongest attributes were "A4_Score" and "A9_Score". The worst were "jundice" and once again "age_desc". The full list can be seen in figure 5 below.



```
Ranked attributes:
 0.5685    4 A4_Score
 0.4862    9 A9_Score
 0.4399   10 A10_Score
 0.4384    8 A8_Score
 0.4173    6 A6_Score
 0.3955    3 A3_Score
 0.3935    1 A1_Score
 0.3799    5 A5_Score
 0.2739    7 A7_Score
 0.229     2 A2_Score
 0.0753   11 age
 0.0752   16 contry_of_res
 0.0614   19 relation
 0.0556   13 ethnicity
 0.0488   15 austim
 0.0472   17 used_app_before
 0.0388   12 gender
 0.025    14 jundice
 0        18 age_desc

Selected attributes: 4,9,10,8,6,3,1,5,7,2,11,16,19,13,15,17,12,14,18 : 19
```

Figure 5 – The result of Correlation Based attribute selection method.

The final method I chose to use was the Symmetrical Uncertainty method. This measures the symmetrical uncertainty with respect to another set of attributes. Similarly to the previous two methods it was clear which attributes were best and worst. The worst included “age” and “age_desc”. See figure 6 below.



```
Ranked attributes:
 0.25071    4 A4_Score
 0.178006   9 A9_Score
 0.166404  10 A10_Score
 0.145441   6 A6_Score
 0.143521   8 A8_Score
 0.134769   3 A3_Score
 0.123368   5 A5_Score
 0.119808   1 A1_Score
 0.088241  16 contry_of_res
 0.056086   7 A7_Score
 0.0383     2 A2_Score
 0.020839  13 ethnicity
 0.013287  19 relation
 0.002652  17 used_app_before
 0.002088  15 austim
 0.001165  12 gender
 0.000491  14 jundice
 0          18 age_desc
 0          11 age

Selected attributes: 4,9,10,6,8,3,5,1,16,7,2,13,19,17,15,12,14,18,11 : 19
```

Figure 6 – The result of Symmetrical Uncertainty method.

The attributes I would remove would be as follows, “age”, “age_desc”, “jundice”, “gender”, “used_app_before” and finally “autism”. These were consistently in the bottom of all the attribute selection methods results. They could be removed to help improve the dataset so that it consisted of an optimal number of predictor variables.

kNearestNeighbour (IBk) Classifier

For this section I will be investigating some different values of k for the kNearestNeighbour classifier. I will investigate each of the results from the different values and compare them to determine the optimal value of k to be used.

The first value I will evaluate is k=3. I will run the kNearest method using k=3 and use the resulting confusion matrix to perform some calculations. The calculations I will look at will be in a table (table 1) for comparison against the other values of k. The results for each value of k will be included in the appendixes from appendix 1-4.

K	**Sensitivity TP/(TP/FN)**	**Specificity TN/(TN+FP)**	**Accuracy TP+TN/(TP+TN+FN+FP)**	**Error Rate FP+FN/(TP+TN+FN+FP)**
3	130/130+21 = 0.86	132/132+9 = 0.94	130+132/130+21+9+132 = 0.90	9+21/292 = 0.10
5	121/121+30 = 0.80	136/136+5 = 0.96	121+136/121+30+5+136 = 0.88	5+30/292 = 0.12
7	118/118+33 = 0.78	139/139+2 = 0.99	118+139/118+33+2+139 = 0.88	33+2/292 = 0.12
15	113/113+38 = 0.74	140/140+1 = 0.99	113+140/113+38+1+140 = 0.87	9+21/292 = 0.13

Table 1 – Table shows calculations for accuracy of kNearestNeighbour based on resulting confusion matrices.

As the value of k increases while using the kNearestNeighbour Classifier on the dataset it is clear that the outcome percentages are decreased. For example the minimum value of k, 3, has the highest percentages in correctly classified instances compared to the maximum value of 15 where the difference is roughly 3% more. For a large dataset this would impact the model greatly. For this reason I would choose the value of k as 3 so that it maximises the chances of correctly predicting classifier instances.

Identify two suitable Machine Learning Algorithms

Ensemble Method for Machine Learning Algorithms – Boosting

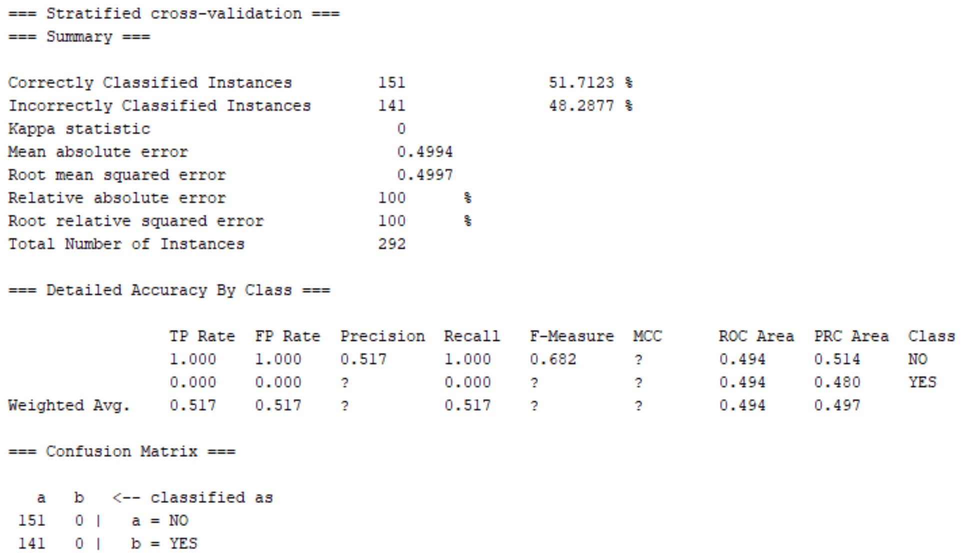
This is when you incrementally build an ensemble. This is done by training each model instance to emphasize the training instances that older models mis-classified. Boosting has been shown to have better results than bagging, which is also another ensemble method. The only downside of this is that boosting tends to over-fit the training data as a result. The most common implementation is Adaboost. Boosting is a supervised learning method that takes weak learners and "boosts" them to become strong learners. A weak learner is a learner that is only slightly correlated with the true classification. In contrast, a strong learner is a classifier that is arbitrarily well-correlated with the true classification. (Wikipedia, 2018)

Voted Perceptron

This is based on Rosenblatt and Frank's perceptron algorithm. This takes data that's linearly separable with large margins. This method can be used in high dimensional spaces using kernel functions. Assuming the instances are points, using the Euclidean length each value has a length of x and labels y. This is then used to calculate a prediction on a new instance. If the prediction is different than the current label y then it updates the prediction vector. If it is correct then the predictor vector remains the same. If this process is repeated the results of positive and negative examples will form a function of the gap. This information gained is then used to calculate such results as how long it survives until the next mistake is made, this is referred to as the weight. The weights are used to show which vectors are good as they would have survived a longer time without making a mistake. (Cohen Courses, 2018)

Evaluation of Machine Learning algorithms:

ZeroR (baseline algorithm)

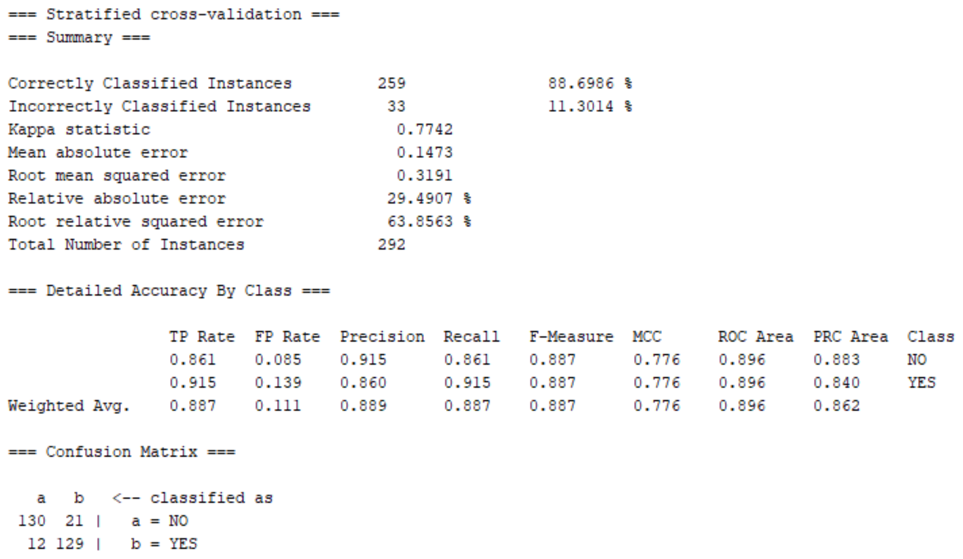


```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         151               51.7123 %
Incorrectly Classified Instances       141               48.2877 %
Kappa statistic                          0
Mean absolute error                      0.4994
Root mean squared error                  0.4997
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances              292

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    1.000    0.517      1.000    0.682      ?        0.494     0.514     NO
                 0.000    0.000    ?          0.000    ?          ?        0.494     0.480     YES
Weighted Avg.    0.517    0.517    ?          0.517    ?          ?        0.494     0.497

=== Confusion Matrix ===

   a   b   <-- classified as
 151   0 |   a = NO
 141   0 |   b = YES
```

Figure 7 – ZeroR baseline algorithm results.

Using the ZeroR as the baseline algorithm it is clear that every other algorithm must have an accuracy higher than the one achieved here. If an algorithm is to have strengths of classifying data then it should be greater than 52%.

JRip

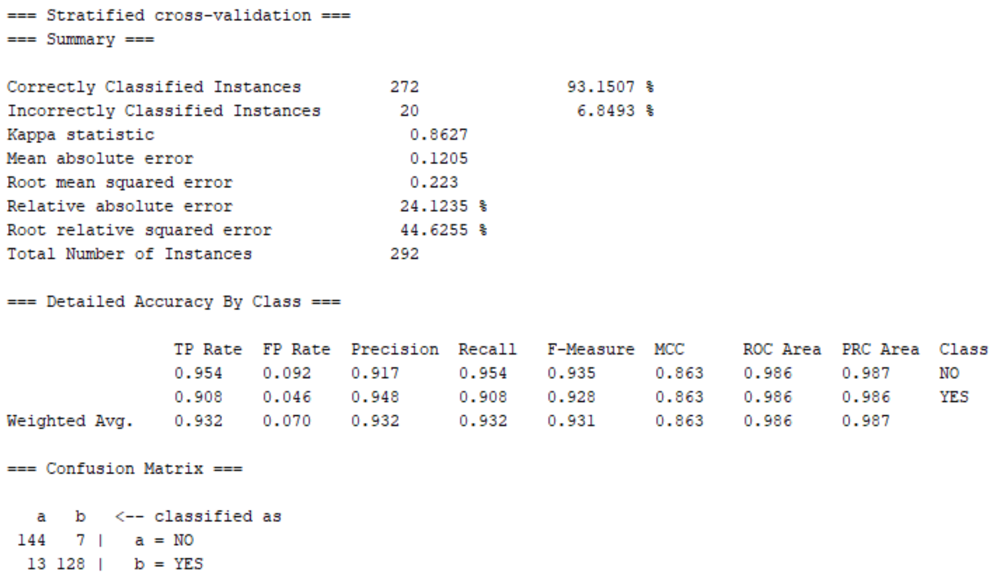


```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         259               88.6986 %
Incorrectly Classified Instances        33               11.3014 %
Kappa statistic                          0.7742
Mean absolute error                      0.1473
Root mean squared error                  0.3191
Relative absolute error                 29.4907 %
Root relative squared error             63.8563 %
Total Number of Instances              292

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.861    0.085    0.915      0.861    0.887      0.776    0.896     0.883     NO
                 0.915    0.139    0.860      0.915    0.887      0.776    0.896     0.840     YES
Weighted Avg.    0.887    0.111    0.889      0.887    0.887      0.776    0.896     0.862

=== Confusion Matrix ===

   a   b   <-- classified as
 130  21 |   a = NO
  12 129 |   b = YES
```

Figure 8 – JRip results.

It is clear that this has some strength over the baseline ZeroR. The accuracy of 89% is a massive increase from the baselines 52%. A more informative confusion matrix is also formed where further calculations can be made.

NaiveBayes

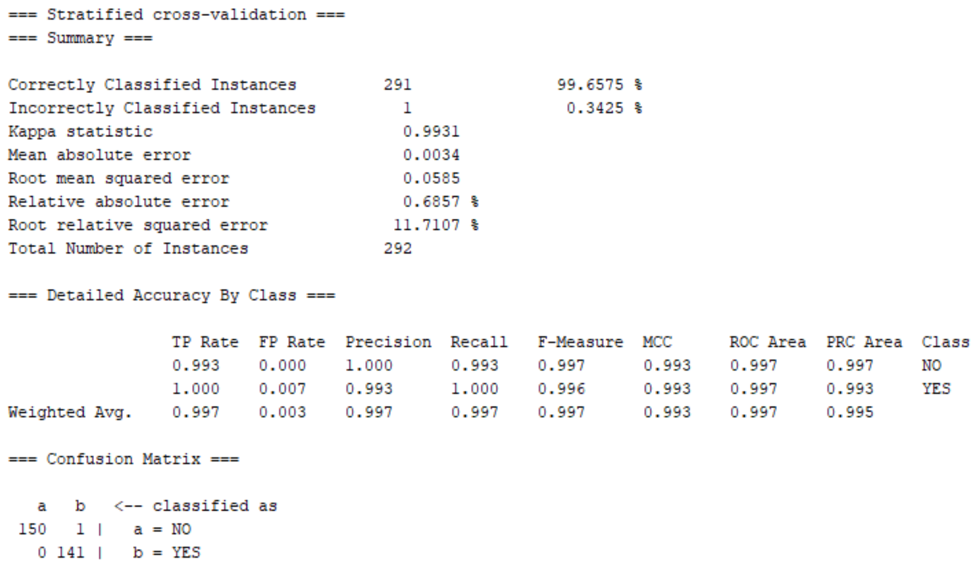


```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         272               93.1507 %
Incorrectly Classified Instances        20                6.8493 %
Kappa statistic                          0.8627
Mean absolute error                      0.1205
Root mean squared error                  0.223
Relative absolute error                 24.1235 %
Root relative squared error             44.6255 %
Total Number of Instances              292

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.954    0.092    0.917      0.954    0.935      0.863    0.986     0.987     NO
                 0.908    0.046    0.948      0.908    0.928      0.863    0.986     0.986     YES
Weighted Avg.    0.932    0.070    0.932      0.932    0.931      0.863    0.986     0.987

=== Confusion Matrix ===

   a   b   <-- classified as
 144   7 |   a = NO
  13 128 |   b = YES
```

Figure 9 – NaiveBayes results.

NaiveBayes is a choice for the optimal machine learning algorithm due to the correctly classified instances of over 90%. This is a popular algorithm as it only relies on a small about of learner data to produce this result.

SMO - Sequential Minimal Optimisation

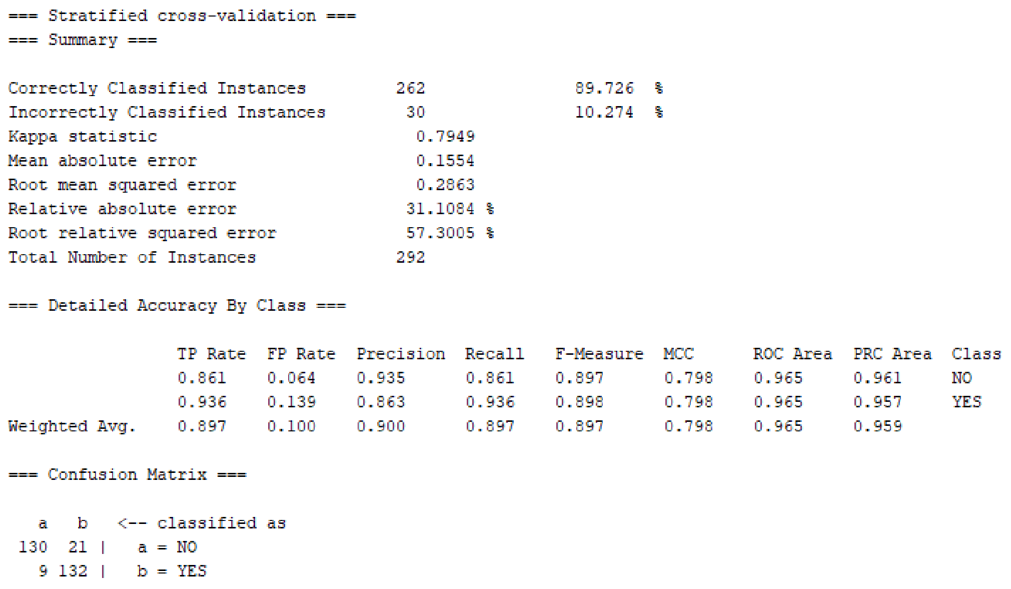


```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         291               99.6575 %
Incorrectly Classified Instances         1                0.3425 %
Kappa statistic                          0.9931
Mean absolute error                      0.0034
Root mean squared error                  0.0585
Relative absolute error                  0.6857 %
Root relative squared error             11.7107 %
Total Number of Instances              292

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.993    0.000    1.000      0.993    0.997      0.993    0.997     0.997     NO
                 1.000    0.007    0.993      1.000    0.996      0.993    0.997     0.993     YES
Weighted Avg.    0.997    0.003    0.997      0.997    0.997      0.993    0.997     0.995

=== Confusion Matrix ===

   a   b   <-- classified as
 150   1 |   a = NO
   0 141 |   b = YES
```

Figure 10 – Sequential Minimal Optimisation results.

The Sequential Minimal Optimisation algorithm has nearly achieved a perfect result for correctly classified instances. Having a 99.6% accuracy is a clear indication of how this might be used for the model. Any results higher than this would be a definite choice.

IBk - kNearestNeighbour (using k=3)

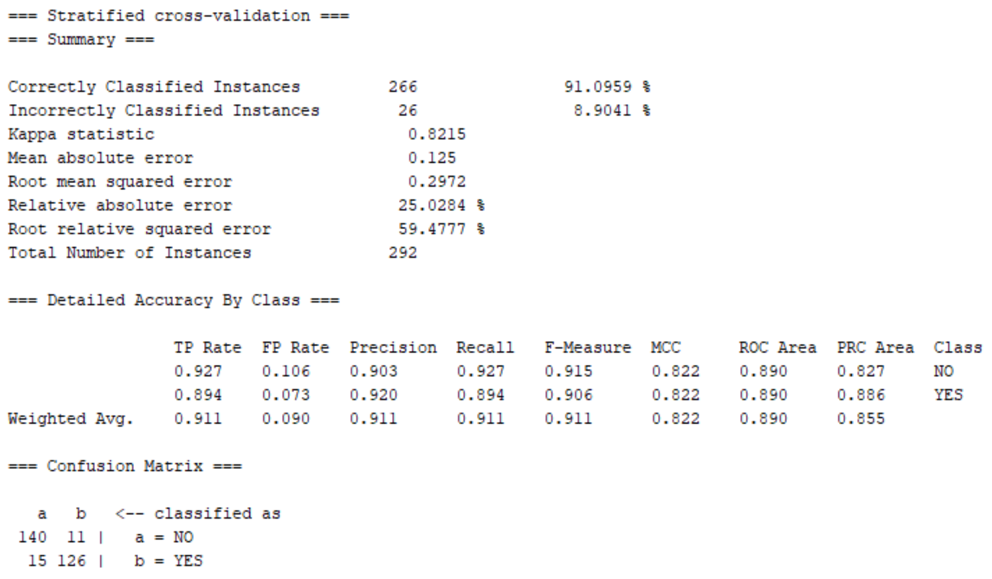


```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         262               89.726  %
Incorrectly Classified Instances        30               10.274  %
Kappa statistic                          0.7949
Mean absolute error                      0.1554
Root mean squared error                  0.2863
Relative absolute error                 31.1084 %
Root relative squared error             57.3005 %
Total Number of Instances              292

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.861    0.064    0.935      0.861    0.897      0.798    0.965     0.961     NO
                 0.936    0.139    0.863      0.936    0.898      0.798    0.965     0.957     YES
Weighted Avg.    0.897    0.100    0.900      0.897    0.897      0.798    0.965     0.959

=== Confusion Matrix ===

   a   b   <-- classified as
 130  21 |   a = NO
   9 132 |   b = YES
```

Figure 11 – kNearestNeighbour results using k=3.

kNearestNeighbour is a strong choice for the model. Although with the knowledge of results from the SMO and NaiveBayes it won't be used. If there was an unknown value for k which may result in a better correctly classified instances result.

J48

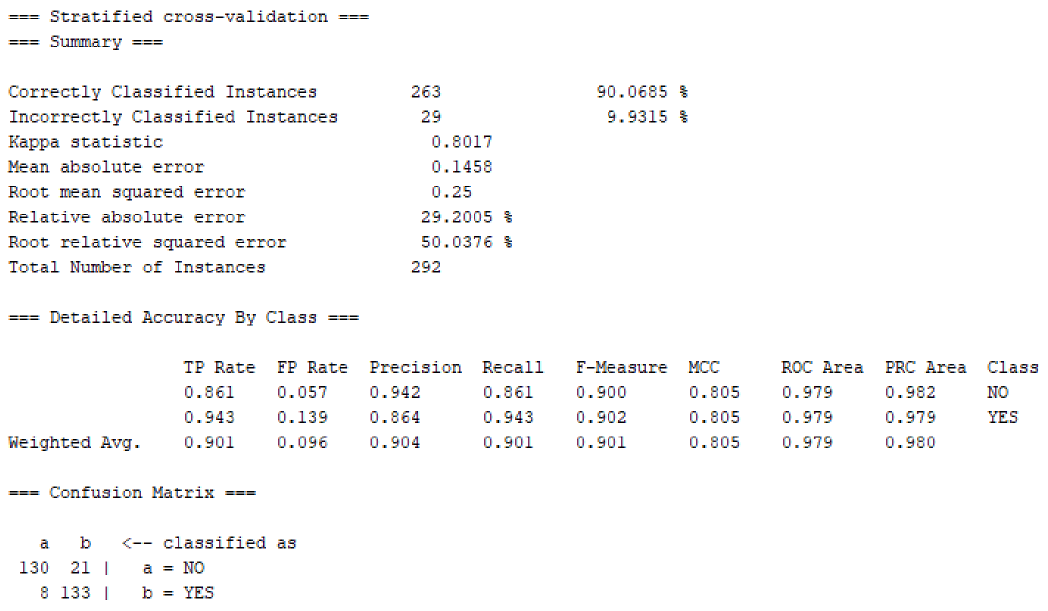


```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         266               91.0959 %
Incorrectly Classified Instances        26                8.9041 %
Kappa statistic                          0.8215
Mean absolute error                      0.125
Root mean squared error                  0.2972
Relative absolute error                 25.0284 %
Root relative squared error             59.4777 %
Total Number of Instances              292

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.927    0.106    0.903      0.927    0.915      0.822    0.890     0.827     NO
                 0.894    0.073    0.920      0.894    0.906      0.822    0.890     0.886     YES
Weighted Avg.    0.911    0.090    0.911      0.911    0.911      0.822    0.890     0.855

=== Confusion Matrix ===

   a   b   <-- classified as
 140  11 |   a = NO
  15 126 |   b = YES
```

Figure 12 – J48 results.

J48 achieves above the baseline algorithm but still falls short compared to others such as SMO. It could be considered for the model as it is above 90% which would normally be considered high but due to SMO being significantly higher again it won't be used.

Boosting

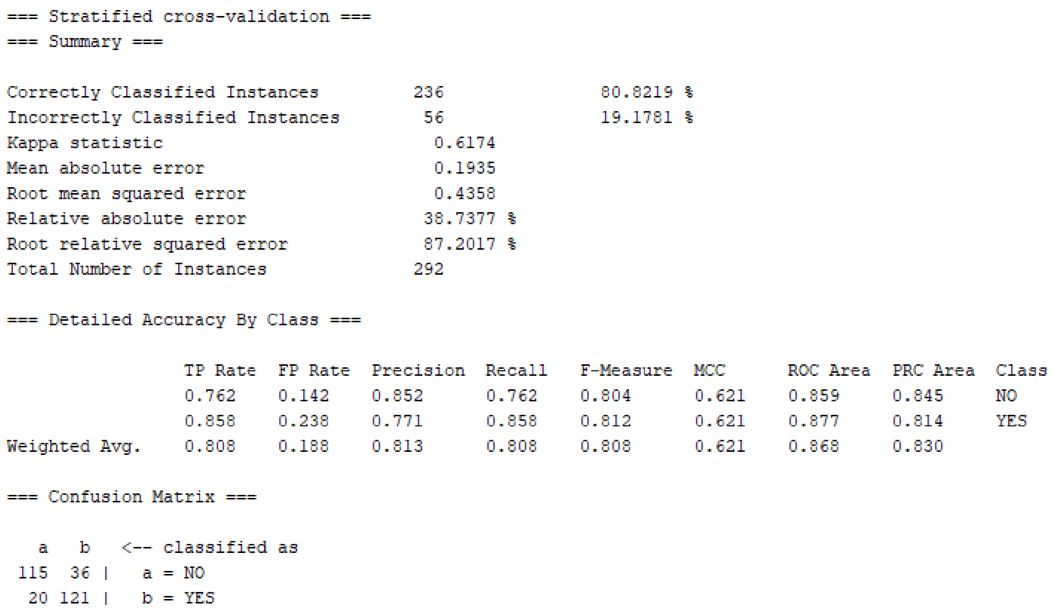


```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         263               90.0685 %
Incorrectly Classified Instances        29                9.9315 %
Kappa statistic                          0.8017
Mean absolute error                      0.1458
Root mean squared error                  0.25
Relative absolute error                 29.2005 %
Root relative squared error             50.0376 %
Total Number of Instances              292

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.861    0.057    0.942      0.861    0.900      0.805    0.979     0.982     NO
                 0.943    0.139    0.864      0.943    0.902      0.805    0.979     0.979     YES
Weighted Avg.    0.901    0.096    0.904      0.901    0.901      0.805    0.979     0.980

=== Confusion Matrix ===

   a   b   <-- classified as
 130  21 |   a = NO
   8 133 |   b = YES
```

Figure 13 – Boosting results.

The ensemble algorithm proves its worth with achieving over 90% accuracy but is still behind other options for machine learning algorithms. It could be an option if an ensemble algorithm is a must for the model but as it is not for this assignment it won't be important.

Voted Perceptron



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

Correctly Classified Instances         236               80.8219 %
Incorrectly Classified Instances        56               19.1781 %
Kappa statistic                          0.6174
Mean absolute error                      0.1935
Root mean squared error                  0.4358
Relative absolute error                 38.7377 %
Root relative squared error             87.2017 %
Total Number of Instances              292

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.762    0.142    0.852      0.762    0.804      0.621    0.859     0.845     NO
                 0.858    0.238    0.771      0.858    0.812      0.621    0.877     0.814     YES
Weighted Avg.    0.808    0.188    0.813      0.808    0.808      0.621    0.868     0.830

=== Confusion Matrix ===

   a   b   <-- classified as
 115  36 |   a = NO
  20 121 |   b = YES
```

Figure 14 – Voted Perceptron results.

Lowest result compared to the baseline algorithm. Should not be considered when progressing with the data model. Although this is above the baseline algorithm it is very low compared to the other machine learning algorithms it is being compared to for this segment of the assignment.

Performance of Machine Learning algorithms

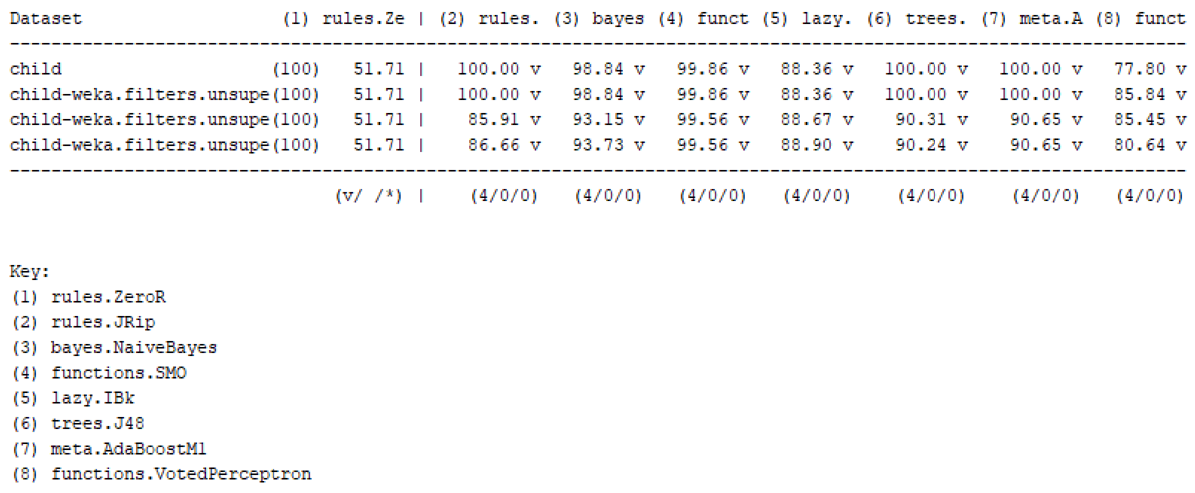
All of the algorithms resulted in values that were statistically larger. This is shown with the symbol (v). The datasets were as follows in order, original dataset, normalised dataset, standardised dataset and missing dataset.

Results on using different data-sets

For every algorithm bar the last, VotedPerceptron, there was the same accuracy for the original dataset and the normalised dataset. Similarly both the SMO and boosting algorithms produced the same accuracy for the standardised and missing datasets. This was two of the highest results on these datasets.

Best performing algorithms

The best performing algorithms I can conclude from evaluating these results are the Sequential Minimal Optimisation (SMO) and NaiveBayes. These both did not have any 100% results which would make you assume they weren't good. What they did have was consistent results over 90% which meant they had the highest averages. This can be seen in figure 15 below.



```
Dataset                   (1) rules.Ze | (2) rules. (3) bayes (4) funct (5) lazy. (6) trees. (7) meta.A (8) funct
-------------------------------------------------------------------------------------------------------------------
child                    (100)   51.71 |  100.00 v   98.84 v   99.86 v   88.36 v  100.00 v  100.00 v   77.80 v
child-weka.filters.unsupe(100)   51.71 |  100.00 v   98.84 v   99.86 v   88.36 v  100.00 v  100.00 v   85.84 v
child-weka.filters.unsupe(100)   51.71 |   85.91 v   93.15 v   99.56 v   88.67 v   90.31 v   90.65 v   85.45 v
child-weka.filters.unsupe(100)   51.71 |   86.66 v   93.73 v   99.56 v   88.90 v   90.24 v   90.65 v   80.64 v
-------------------------------------------------------------------------------------------------------------------
                              (v/ /*) |    (4/0/0)   (4/0/0)   (4/0/0)   (4/0/0)    (4/0/0)    (4/0/0)   (4/0/0)


Key:
(1) rules.ZeroR
(2) rules.JRip
(3) bayes.NaiveBayes
(4) functions.SMO
(5) lazy.IBk
(6) trees.J48
(7) meta.AdaBoostM1
(8) functions.VotedPerceptron
```

Figure 15 – A comparison of the performance algorithms.

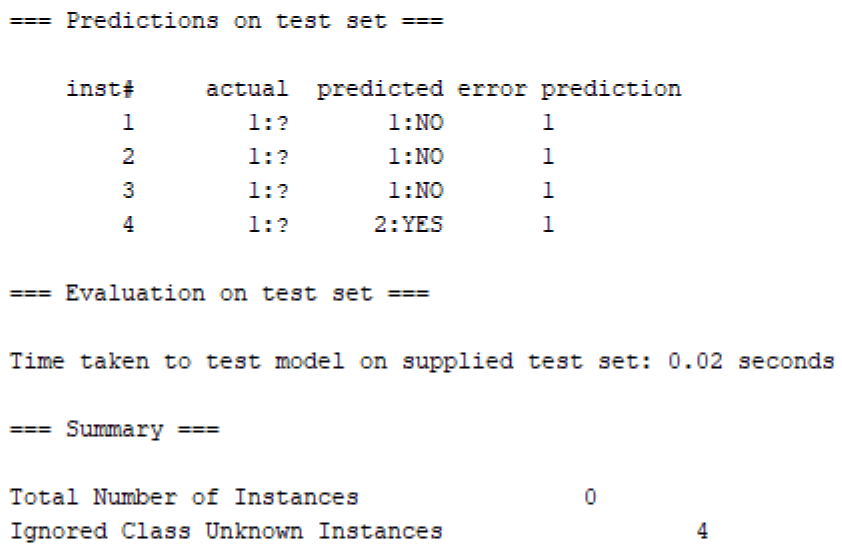
Significant findings and supporting evidence

The standardised and missing datasets had the lowest scores consistently except for the last algorithm VotedPerceptron. This algorithm had two highest results for normalised and standardised.

Final Version of the Model and Present Results

I started by saving my model of SMO based on the full original dataset. Once I has this model I was able to use the unseen dataset provided and assign the supplied test set to this unseen dataset. The unseen dataset consisted of 4 unseen cases which will be used to try predict the outcome of whether or not the child is on the autism spectrum.

When I ran the SMO model against this the results were as follows in the figure 16 below. The first three instances were predicted no and the final was predicted yes. This shows a prediction accuracy of 1 for each case. This means there was a 100% accuracy for these cases. Accompanied with the empty error fields we can conclude that our machine learning model has correctly predicted the results for our 4 unseen cases.



```
=== Predictions on test set ===

    inst#     actual  predicted error prediction
        1        1:?       1:NO       1
        2        1:?       1:NO       1
        3        1:?       1:NO       1
        4        1:?      2:YES       1

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.02 seconds

=== Summary ===

Total Number of Instances                0
Ignored Class Unknown Instances                  4
```

Figure 16 – Results of predictions on the unseen dataset using SMO model.

Section 2

Apriori Algorithm with minimum support 30%.

For this segment I charted the individual products included in the transaction list provided on the assignment.

The products included from this local supermarket are as follows with the corresponding frequencies that they were bought. See table 2 below.

Name:	**Frequency:**
Nikon Camera	8
Micro SD Card	11
Shoot Tripod	12
PS4 Console	14
PS4 GTA Game	10
PS4 FIFA 19 Game	2
Charger	8
PS4 Controller	9
iPad	5
Amazon Echo	7
Fitbit	3
Fitbit Wrist Bands	1

Table 2 – Name of products from supermarket transactions and their corresponding frequencies.

From this table we can remove any products with frequencies are below φ (phi) = 6. This leaves us with 8 products remaining. We can then proceed to create pairs from these remaining products. For each of the pairs we can calculate the frequencies of when both products were purchased together and once again remove any pairs that are below 6.

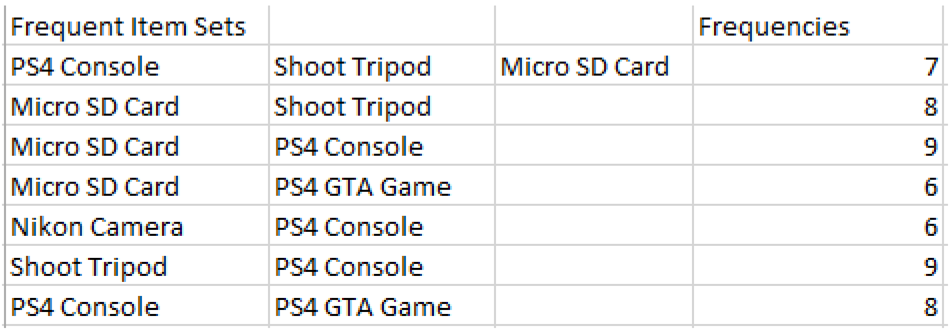
The pairs we are left with then are as follows in table 3.

Pair:	**Frequency:**
Nikon Camera, PS4 Console	6
Micro SD Card, Shoot Tripod	8
Micro SD Card, PS4 Console	9
Micro SD Card, PS4 GTA Game	6
Shoot Tripod, PS4 Console	9
PS4 Console, PS4 GTA Game	8

Table 3 – Pair of products from supermarket transactions and their corresponding frequencies.

From this then we can try create triplets of products that were purchased together in the transactions. It is important to make sure all subsets of the triplets are also greater than or equal to the value of 6. If the subset has a value less than 6 it cannot be a triplet. With this in mind there are only 2 sets of triplets. They are PS4 Console, Shoot Tripod, Micro SD Card which has a frequency of 7 and PS4 Console, PS4 GTA Game, Micro SD Card which had a frequency of below 6 so we don't include it for the next part.

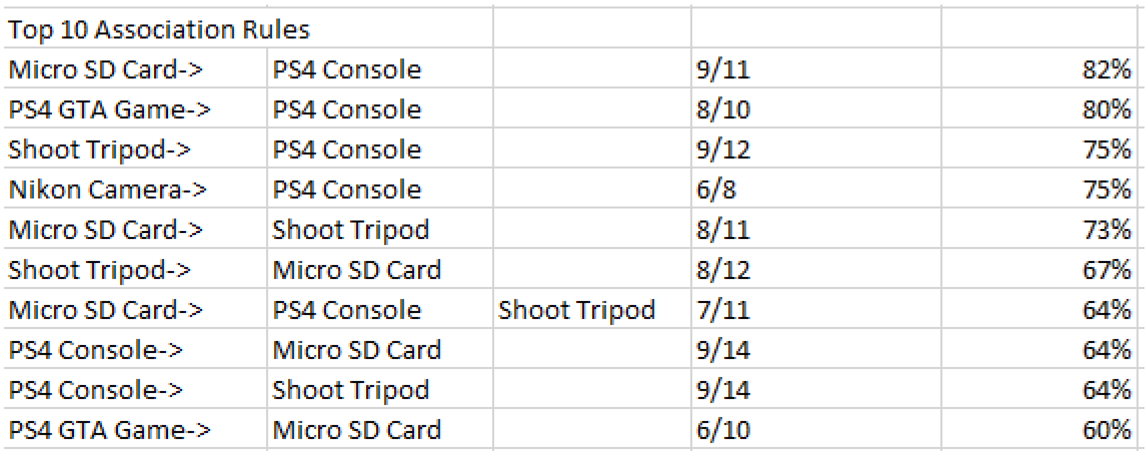
We now have 7 frequent item sets which can be seen in the figure 17 below.



Frequent Item Sets			Frequencies
PS4 Console	Shoot Tripod	Micro SD Card	7
Micro SD Card	Shoot Tripod		8
Micro SD Card	PS4 Console		9
Micro SD Card	PS4 GTA Game		6
Nikon Camera	PS4 Console		6
Shoot Tripod	PS4 Console		9
PS4 Console	PS4 GTA Game		8

Figure 17 – End result of frequent item sets.

From these I have calculated the top 10 association rules that maximise support and confidence of the rule. They can be seen in figure 18.



Top 10 Association Rules				
Micro SD Card->	PS4 Console		9/11	82%
PS4 GTA Game->	PS4 Console		8/10	80%
Shoot Tripod->	PS4 Console		9/12	75%
Nikon Camera->	PS4 Console		6/8	75%
Micro SD Card->	Shoot Tripod		8/11	73%
Shoot Tripod->	Micro SD Card		8/12	67%
Micro SD Card->	PS4 Console	Shoot Tripod	7/11	64%
PS4 Console->	Micro SD Card		9/14	64%
PS4 Console->	Shoot Tripod		9/14	64%
PS4 GTA Game->	Micro SD Card		6/10	60%

Figure 18 – Top 10 Association Rules for the item sets chosen in previous section.

From these results I can say that the top association rule for Micro SD Card and PS4 Console that there was an 82% chance that when the Micro SD Card was bought that the PS4 Console was also purchased within the same transaction. The value of 82% is the highest value and can prove that there is a high certainty of the items being purchased together but there is still some margin of error. The remaining 18% can indicate that we should not guarantee based on these results.

It is interesting that only one rule from the set of three products has made it to the top 10 association rules. This proves that the chances of products being bought together are much more likely to occur when the products are in a pair as opposed to a set of three products.

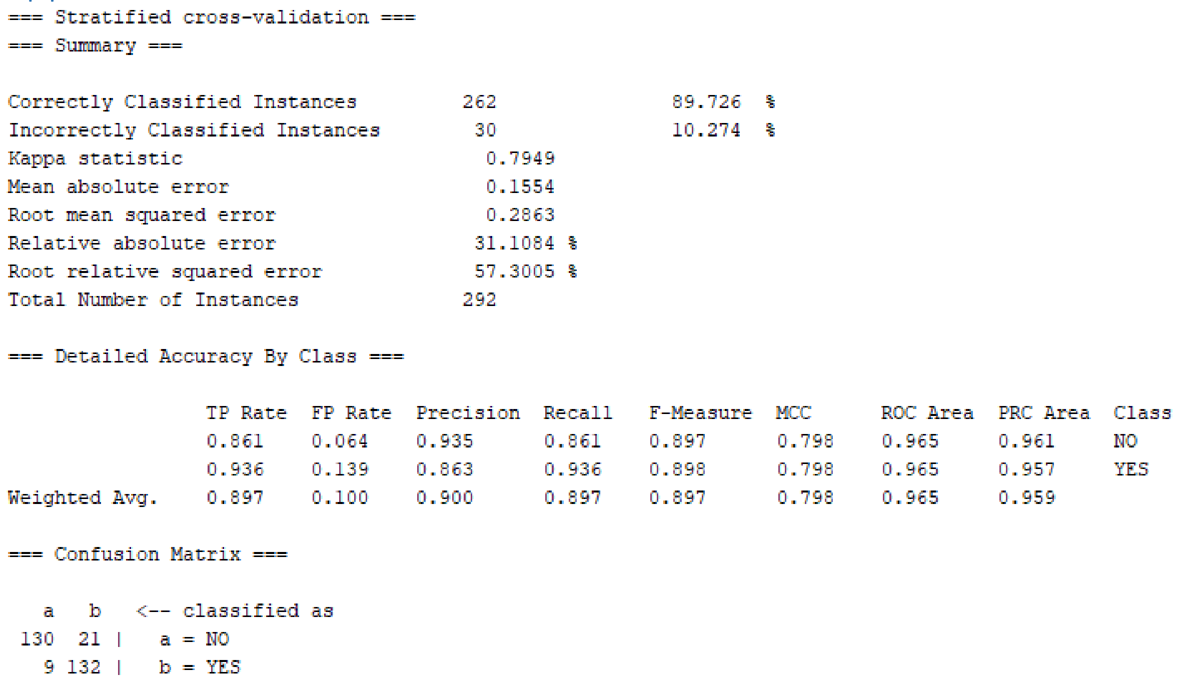
Another rule that interested me was the Nikon Camera, PS4 Console. This was due to the small frequencies of 6/8. From a small sample size this appears in the top 10 rules as it reaches 75% confidence. If the transaction history was larger it would be interesting to see if the freque ncy for this pair would remain low compared to the pairs that it is compared to in this table of top 10 rules.

References:

Boosting (machine learning) - Wikipedia. 2018. *Boosting (machine learning) - Wikipedia*. [ONLINE] Available at: https://en.wikipedia.org/wiki/Boosting_(machine_learning). [Accessed 23 April 2018].

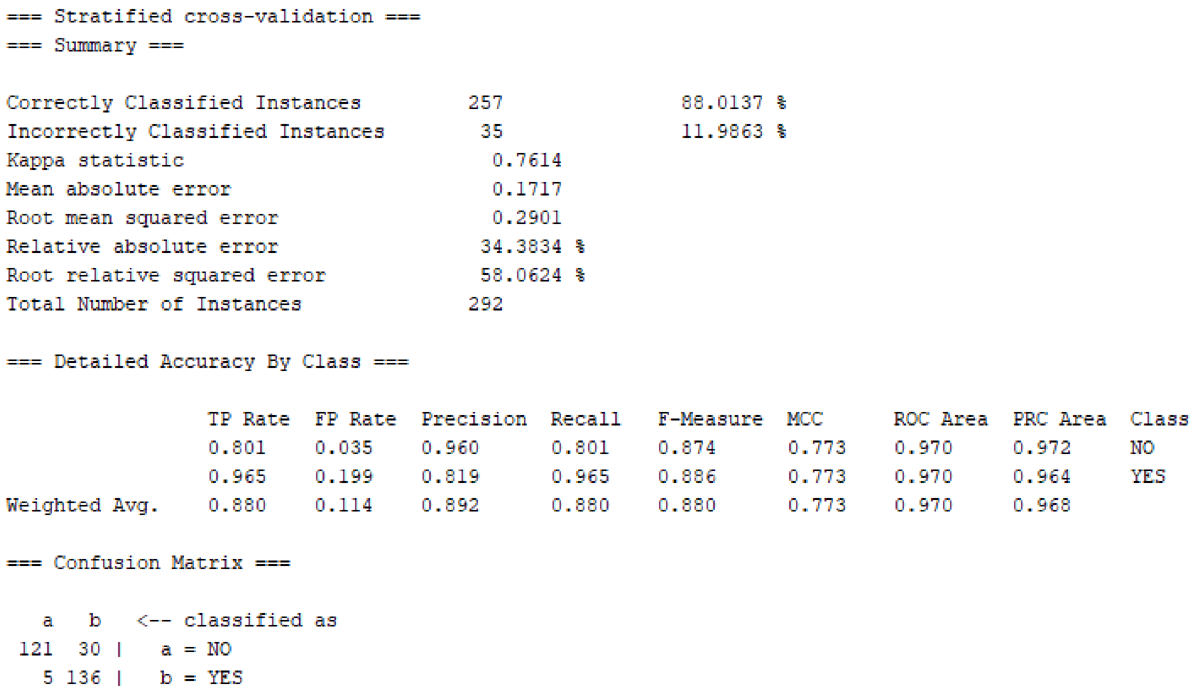
Voted Perceptron - Cohen Courses. 2018. *Voted Perceptron - Cohen Courses*. [ONLINE] Available at: http://curtis.ml.cmu.edu/w/courses/index.php/Voted_Perceptron. [Accessed 23 April 2018].

Appendices



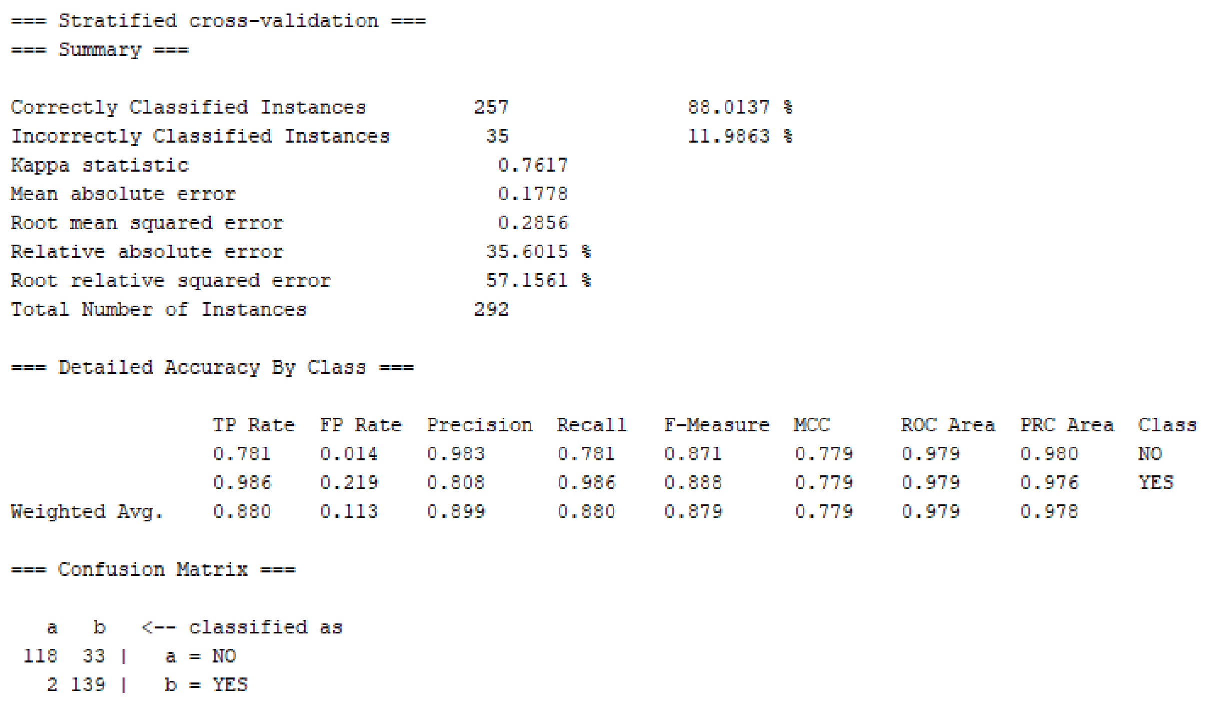
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         262               89.726  %
Incorrectly Classified Instances        30               10.274  %
Kappa statistic                          0.7949
Mean absolute error                      0.1554
Root mean squared error                  0.2863
Relative absolute error                 31.1084 %
Root relative squared error             57.3005 %
Total Number of Instances              292

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.861    0.064    0.935      0.861    0.897      0.798    0.965     0.961     NO
                 0.936    0.139    0.863      0.936    0.898      0.798    0.965     0.957     YES
Weighted Avg.    0.897    0.100    0.900      0.897    0.897      0.798    0.965     0.959

=== Confusion Matrix ===

   a   b   <-- classified as
 130  21 |   a = NO
   9 132 |   b = YES
```

Appendix 1 – Result of kNearestNeighbour with value k=3



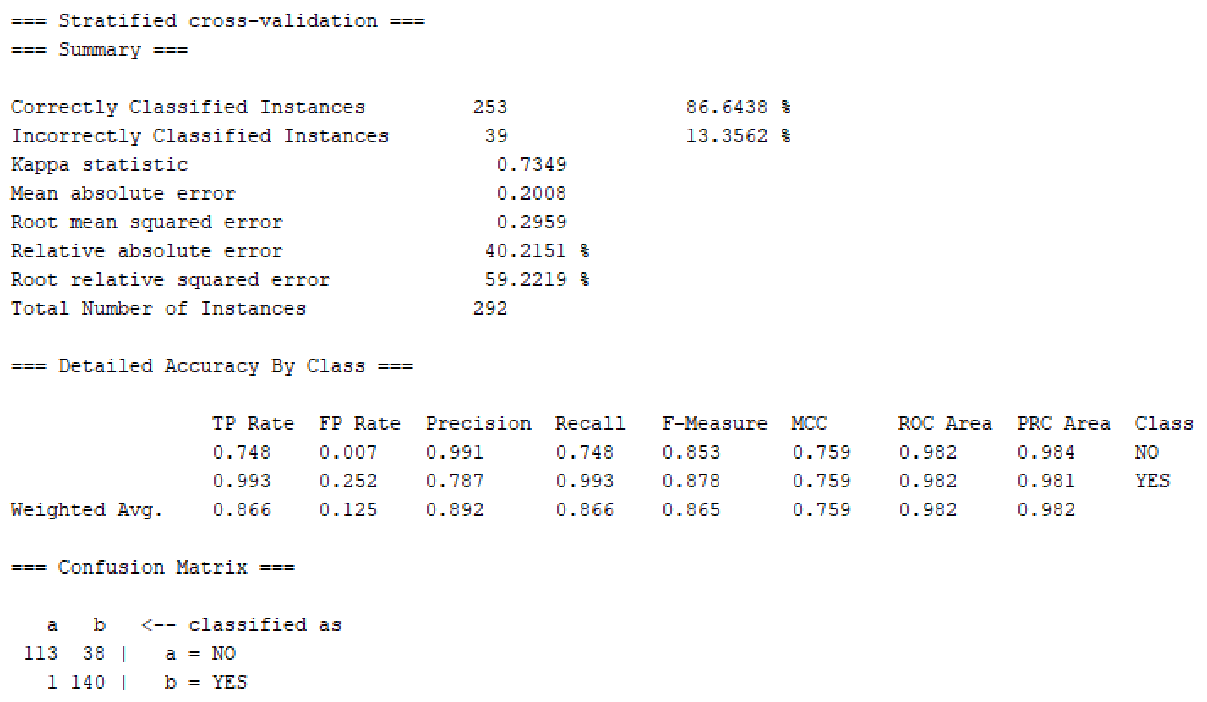
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         257               88.0137 %
Incorrectly Classified Instances        35               11.9863 %
Kappa statistic                          0.7614
Mean absolute error                      0.1717
Root mean squared error                  0.2901
Relative absolute error                 34.3834 %
Root relative squared error             58.0624 %
Total Number of Instances              292

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.801    0.035    0.960      0.801    0.874      0.773    0.970     0.972     NO
                 0.965    0.199    0.819      0.965    0.886      0.773    0.970     0.964     YES
Weighted Avg.    0.880    0.114    0.892      0.880    0.880      0.773    0.970     0.968

=== Confusion Matrix ===

   a   b   <-- classified as
 121  30 |   a = NO
   5 136 |   b = YES
```

Appendix 2 – Result of kNearestNeighbour with value k=5



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

Correctly Classified Instances         257               88.0137 %
Incorrectly Classified Instances        35               11.9863 %
Kappa statistic                          0.7617
Mean absolute error                      0.1778
Root mean squared error                  0.2856
Relative absolute error                 35.6015 %
Root relative squared error             57.1561 %
Total Number of Instances              292

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.781    0.014    0.983      0.781    0.871      0.779    0.979     0.980     NO
                 0.986    0.219    0.808      0.986    0.888      0.779    0.979     0.976     YES
Weighted Avg.    0.880    0.113    0.899      0.880    0.879      0.779    0.979     0.978

=== Confusion Matrix ===

   a   b   <-- classified as
 118  33 |   a = NO
   2 139 |   b = YES
```

Appendix 3– Result of kNearestNeighbour with value k=7



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

Correctly Classified Instances         253               86.6438 %
Incorrectly Classified Instances        39               13.3562 %
Kappa statistic                          0.7349
Mean absolute error                      0.2008
Root mean squared error                  0.2959
Relative absolute error                 40.2151 %
Root relative squared error             59.2219 %
Total Number of Instances              292

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.748    0.007    0.991      0.748    0.853      0.759    0.982     0.984     NO
                 0.993    0.252    0.787      0.993    0.878      0.759    0.982     0.981     YES
Weighted Avg.    0.866    0.125    0.892      0.866    0.865      0.759    0.982     0.982

=== Confusion Matrix ===

   a   b   <-- classified as
 113  38 |   a = NO
   1 140 |   b = YES
```

Appendix 4– Result of kNearestNeighbour with value k=15