Weka CA2

eNTERPRISE DATABASE TECHNOLOGIES

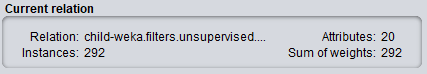
x00120179

2018

## Section 1.

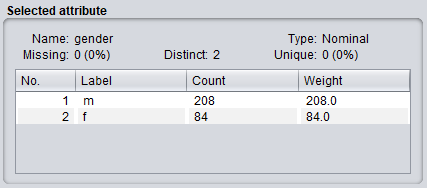
### **Review of data**

After loading the Autism\_screening\_child dataset into Weka, I established that the dataset was comprised of 20 columns and 292 rows of child autism data.

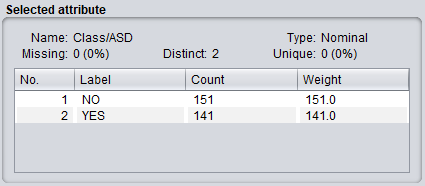


The first 10 columns are answers to 10 behavioural features in relation to autism in children and the last 10 rows are 10 individual characteristics that have proved to be effective in detecting autism cases from controls in behaviour science. I found that 3 of the 20 attribute columns are missing data (‘age’ missing 4 rows which is 1% of its entire data, ‘ethnicity’ missing 43 rows which is 15% of its entire data and ‘relation’ also missing 43 rows). 19 of the rows are of type nominal data, with age being the only numeric type of data in the dataset.

It is worth noting that 208 of the children in this dataset are male with the remaining 84 being female. That is a huge 71% of rows being in relation to males.



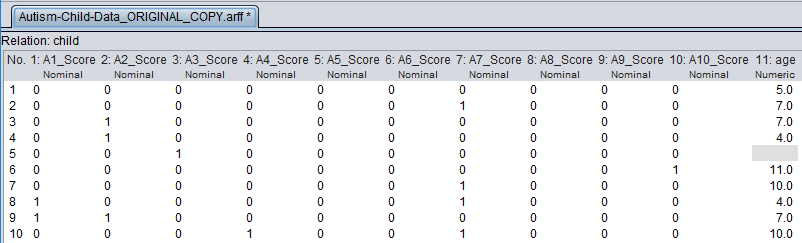
The target variable is the last attribute in the dataset ‘Class/ASD’. This attribute is also the target variable and it tells us whether the patient is diagnosed with ASD (Autistic Spectrum Disorder) or not.



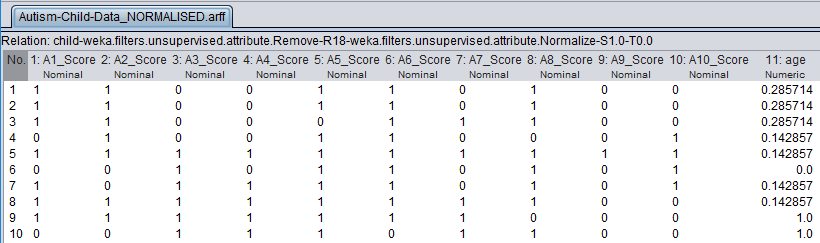
We can see that 141 of the 292 children (nearly half of the children involved in the dataset) are diagnosed with autism.

### **Preparing a number of views (formats) of the dataset.**

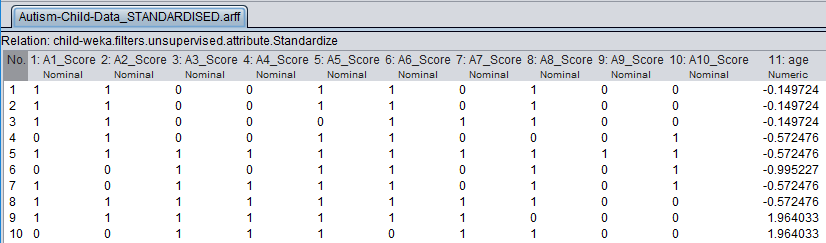
1. To prepare my data I loaded the original Autism-Child-Data.arff file into Weka and saved it as the original copy like so:



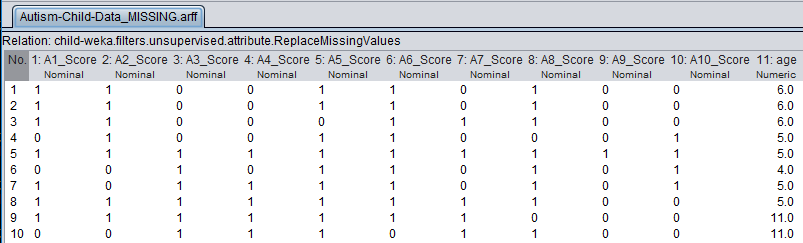
1. Normalising data is the process of rescaling the data attributes to a range of 0 to 1. The means the largest attribute is 1 and the smallest is 0. According to Jason Brownlee of machinelearningmastery.com “Normalisation is a good technique to use when you do not know the distribution of your data or when you know the data distribution is not a bell curve”. To Normalise my dataset I opened up the original file in the Weka explorer, clicked the “choose” button and applied the unsupervised.attribute.Normalise filter to the dataset. I then saved the dataset as Autism-Child-Data\_Normalised.arff as seen below.



1. Standardising data is the process of rescaling attributes so that they have an average value of 0 and a standard deviation of 1. Jason Brownlee from machinelearningmastery.com explains that Standardisation assumes that your data has a Gaussian (bell curve) distribution. While this does not strictly have to be true the technique is more effective if your distribution is Gaussian. To Standardise my data I opened the original file in the Weka explorer, clicked the “choose” button and applied the unsupervised.attribute.Standardise filter to the dataset. I then saved the dataset as Autism-Child-Data\_Standardised.arff as seen below.



1. For missing data I loaded the original dataset into the Weka explorer, clicked the “choose” button and applied the ReplaceMissingValues filter to the dataset and saved it as Autism-Child-Data\_MISSING.arff as seen below.



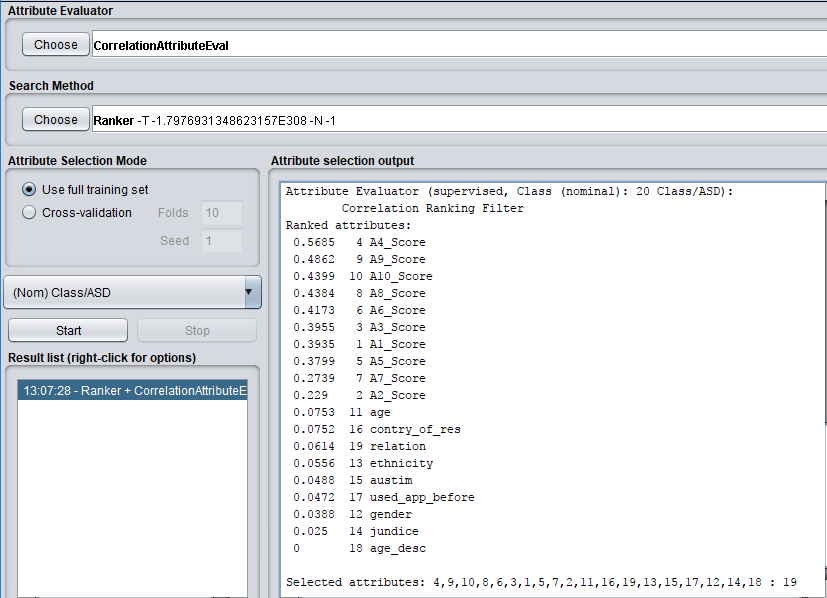
### **Attribute Selection (feature selection)**

Determining the optimal number of predictor variables to use in the modelling stage of the assignment, using three attribute selection methods to determine the attributes.

1. **CorrelationAttributeEval**

The CorrelationAttributeEval technique used in the next section can only be used with a Ranker Search Method, that evaluates each attribute and lists the results in a rank order.

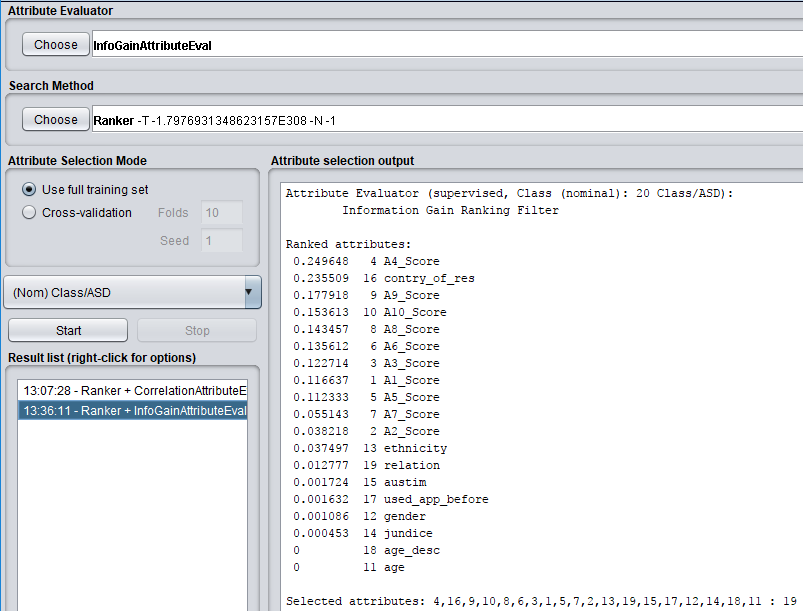
After running this evaluator the outcome suggested that the A4\_Score attribute was ranked 1st with a value of 0.5685 when it came to correlation between the class attribute “Class/ASD”. None of the individual characteristic attributes ranked ahead of the answer attributes which shows that the questions being asked to the children has a big impact of finding the outcome to whether a child has autism or not.



1. **InfoGainAttributeEval**

Weka tell us that this attribute evaluator evaluates the worth of an attribute by measuring the information gain with respect to the class.

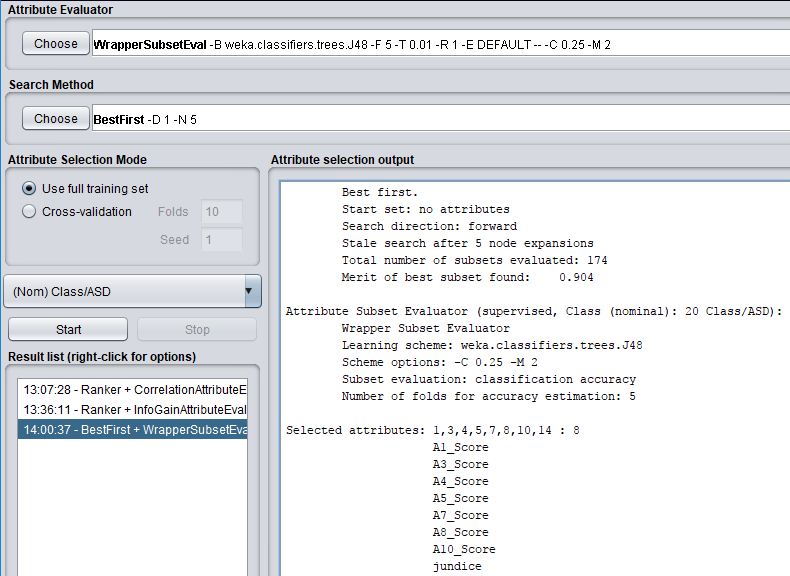
After running this evaluator the outcome once again tells us that the A4\_Score attribute is ranked the highest but this time in terms of information gained in relation to the class attribute “Class/ASD”. Also, there is an individual characteristic attribute “contry\_of\_res” ranked second where as in the correlation evaluator there was no individual characteristic attribute in the top 10 ranked attributes. The fact A4\_Score is the highest ranked again only strengthens the fact that it must be a huge factor in determining whether a child has autism.



1. **WrapperSubsetEval: Learner Based Feature Selection**

Weka tells us that this evaluation method evaluates attribute sets by using a learning scheme. Instead of using the “Ranked” search method this evaluation uses the “GreedyStepwise” or “BestFirst” search method. It is said that BestFirst is preferred so I will be using the BestFirst search method for this evaluation.

Running this feature selection technique on the Autism Screening Child dataset selects 8 of the 20 input variables: A1\_Score, A3\_Score, A4\_Score, A5\_Score, A7\_Score, A8\_Score, A1\_Score and Jaundice.



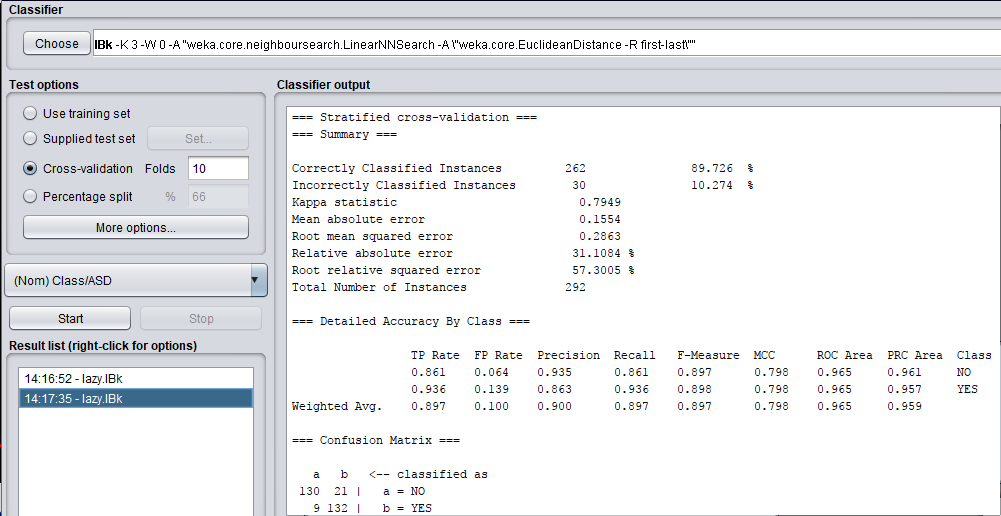
Looking back over the 3 techniques we can see that A4\_Score overlaps and is selected in all 3. Like mentioned previously this shows us that it has an impact in finding autism in children and that it should always be a factor in the autism behavioural science. Following the third method, I removed all other attributes and created a view of the Autism-Screening-Child dataset with just the learner based feature selection attributes for future evaluation if needed.

### **kNearestNeighbour Classifier**

According to Michael Abernathy from www.IBM.com, kNearestNeighbour or Instance-based Learning is data mining is a technique that allows you to use your data instances, with known output values to predict unknown output value of a new data instance. He also tells us that KNN is extremely scalable for huge scaling databases.

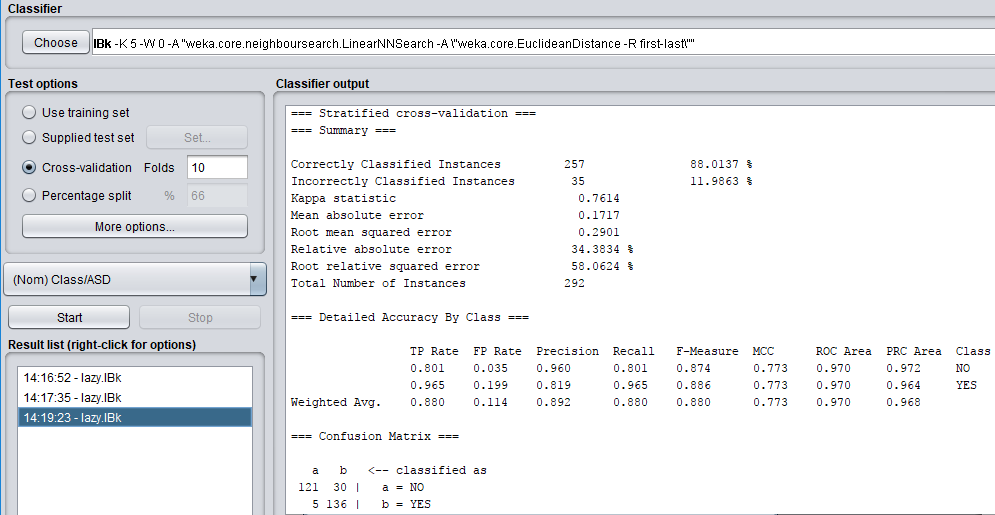
1. **K = 3.**

With K = 3 we get an accuracy of 89.726% and an inaccuracy of 10.274%. This tells us that over the whole dataset (292 rows), 262 were correctly classified. Michael Abernathy from www.IBM.com says that an accuracy of nearly 90% would be very acceptable. The result says K = 3 has 21 false positives (7%), and has 9 false negatives (3%).



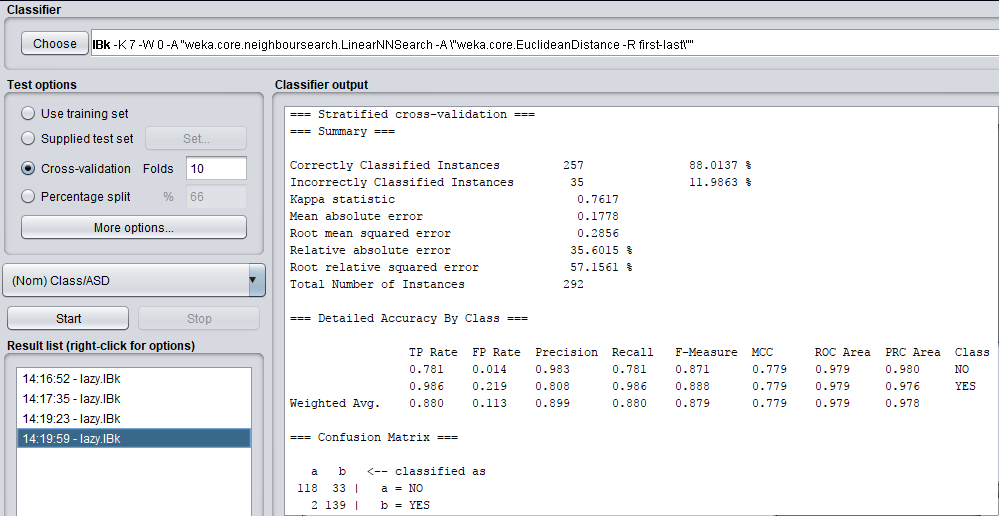
1. **K = 5.**

With K = 5 we get an accuracy of 88.0137% and an inaccuracy of 11.9863%. Where K = 5 is accurate and returns good data, it is slightly not as accurate as K = 3. The result says K = 5 has 30 false positives (10%), and has 5 false negatives (2%).



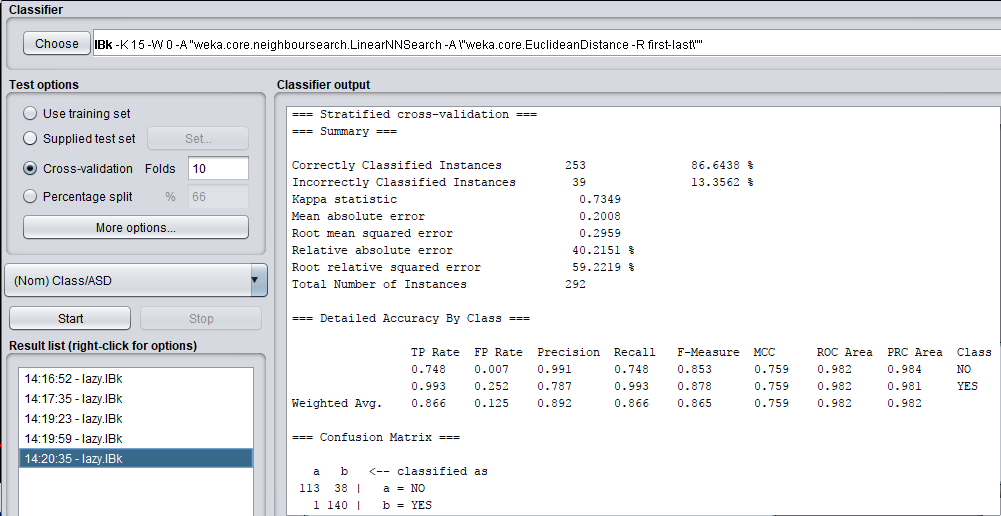
1. **K = 7.**

With K = 7 we get an accuracy of 88.0137%. Which is the same as when K = 5. We also get the same measure of inaccuracy at 11.9863%. While the accuracy and inaccuracy findings are the exact same as K = 5 the false positives and negatives of K = 7 are different. The result says K = 7 has 33 false positives (11%), and has 2 false negatives (0.3%).



1. **K = 15.**

With K = 15 we get an accuracy of 86.6428% and an inaccuracy of 13.3562%. This is the lowest accuracy and highest inaccuracy of all the values of K. It also has the highest amount of false positive with 38 (13%) and the lowest amount of false negatives at 1 (0.3%).



Looking over the results of K as the value of K has risen from 3 to 15 it is clear to see that the higher the value of K the lower the accuracy of the results. K = 3 was the most accurate while K = 15 was the lowest accurate. K = 5 and K = 7 both had identical accuracy and inaccuracy readings but they had differing Confusion Matrix’s. It’s worth noting that I didn’t round up accuracy percentages as the difference in readings is only small but never the less must be taken into consideration when evaluating the KNN. KNN definitely performs better than the previous models as it gives us much more information on the dataset and it also gives us this information as a percentage in relation to the data as a whole.

### **Two suitable Machine Learning (ML) Algorithms.**

1. **Random Forest: trees.RandomForest (Ensemble Method).**

The first machine learning algorithm I have chosen is the Random Forest algorithm which is also an Ensemble method. Necati Demir of [www.toptal.com](http://www.toptal.com) says that Ensemble methods are techniques that create multiple models and then combine them to produce improved results. Think of the idea of divide and conquer. The Random Forest algorithm creates a forest and makes it random. The “forest” is an ensemble of decision trees. Niklas Donges of [www.towardsdatascience.com](http://www.towardsdatascience.com) says “Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.” Donges also states that one big advantage of random forest is that it can be used for both classification and regression problems, which form the majority of current machine learning systems. From reading both Demir and Donges’ articles I have established that Random Trees is a very popular and suited algorithm for any type of machine learning and it is also extremely accurate so I think it would be a suitable algorithm for the evaluation. It would also be good to compare it to kNearestNeighbour and see the differences in their accuracy.

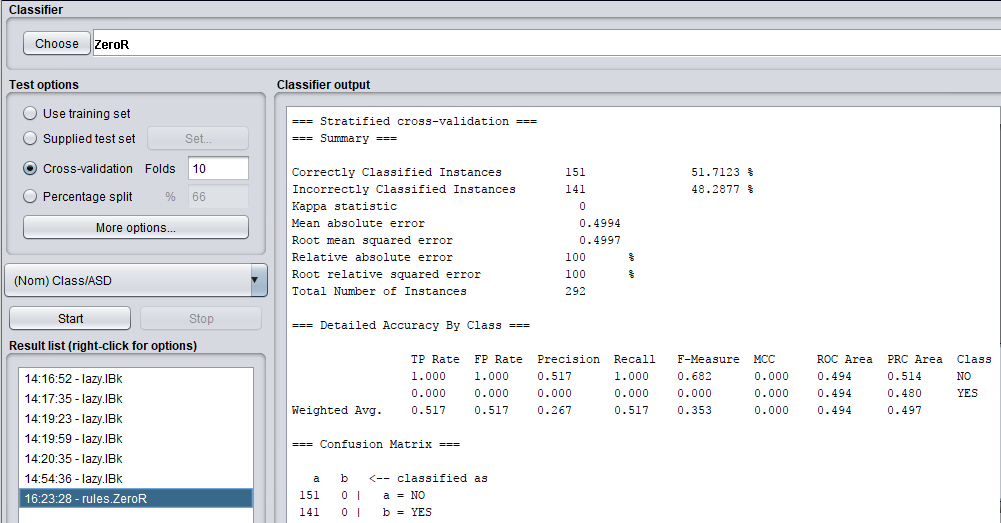
1. **K-Means Clustering Algorithm**

“K-means clustering is a type of unsupervised learning” says Andrea Trevino of [www.datascience.com](http://www.datascience.com). The algorithm is used to categorise unlabelled data, i.e. data without defined categories or groups. The K-means clustering algorithm works by locating groups within the data itself. The number of groups is then represented by variable K. It works iteratively to assign each data point to one of the K categories based on the data points features. Trevino says that rather than defining groups before looking at the data, clustering allows you to find and analyse the groups that have formed organically/naturally. From reading Trevino’s article on K-means clustering it seems that k-means clustering is the way forward when it comes to missing data in large datasets. There are some missing values in the Autism-Screening-Child dataset so this may be a suitable machine learning algorithm for filling in the gaps and correcting the missing data inside the dataset.

### **Evaluation of Machine Learning Algorithms.**

* **rules.ZeroR --baseline algorithm**

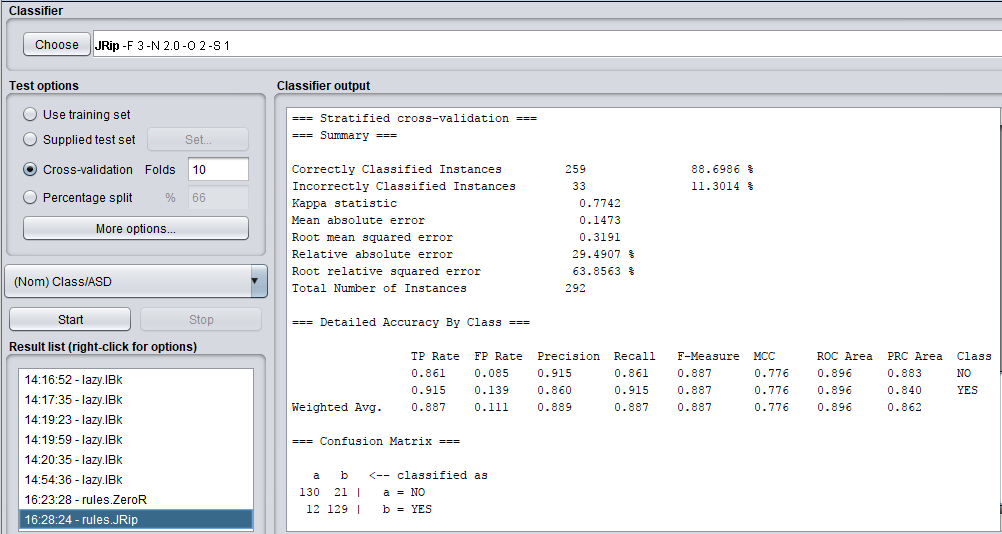
[www.chem-eng.utoronto.ca](http://www.chem-eng.utoronto.ca) says ZeroR is a simple classifier method which relies on the target variable (Class/ASD is this case) and ignores all predictor variables. The site goes on to say that the ZeroR classifier simply “predicts the majority category (class)” and that there is no predictability power in ZeroR. They finish off telling us that it is useful for determining a baseline performance benchmark for other classification methods.



When using ZeroR on the Autism-Screening-Child dataset we see that of the 292 instances 151 where correctly classified, which is an accuracy of 52%. this is low compared to the accuracy of kNearestNeighbour at around 88%. 141 instances of the 292 were also incorrectly classified at 48%. There is no B value in the confusion matrix which to me looks odd.

* **rules.JRip**

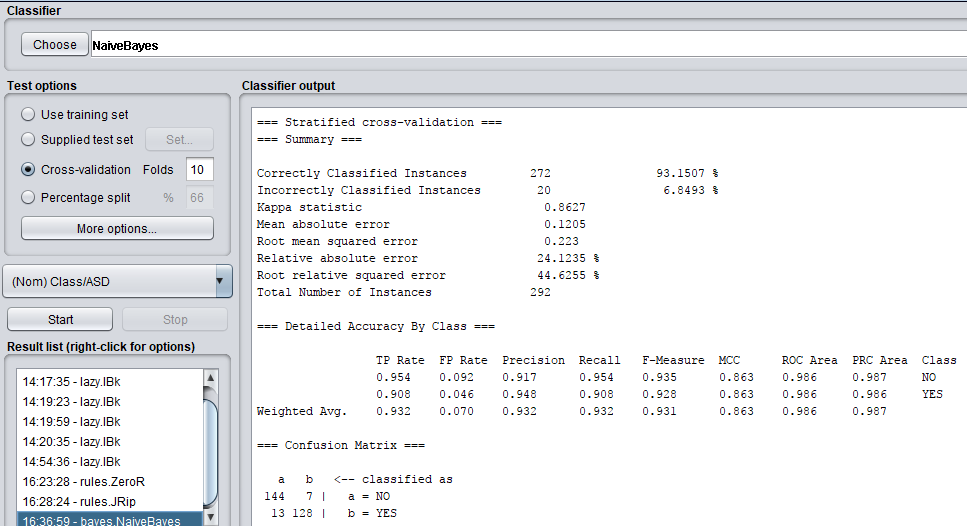
JRIP uses a propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER for short), which was originally proposed by William W. Cohen as an optimized version of IREP says [www.eecs.yorku.ca](http://www.eecs.yorku.ca). It consists of 4 stages/phases. The building stage, the grow phase, the prune phase, and the optimizing phase.



JRIP returned a much better accuracy than the ZeroR algorithm. 259 of the 292 instances where correctly classified at approximately 87% and it also only had an inaccuracy of 11%. The confusion matrix also looks better than ZeroR’s.

* **bayes.NaïveBayes**

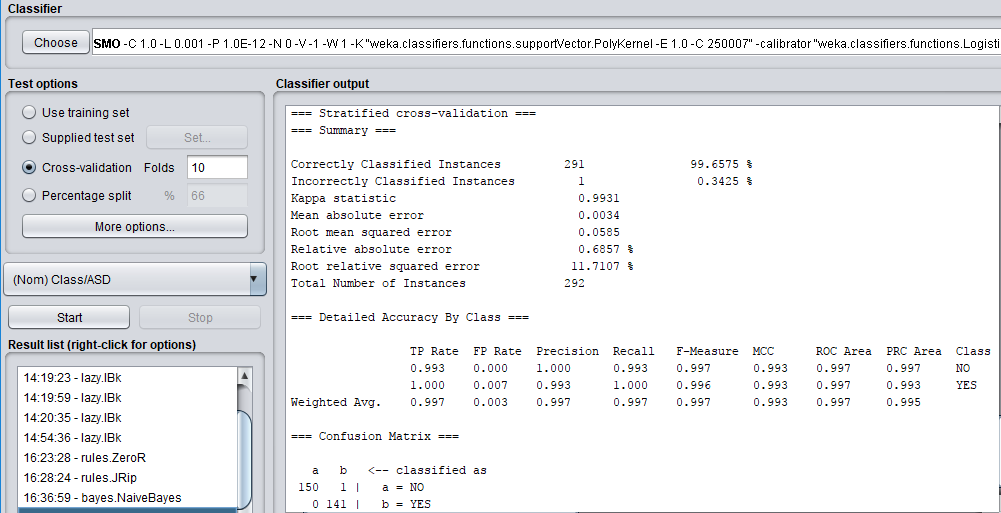
Mike Waldron of [www.blog.aylien.com](http://www.blog.aylien.com) says that Naïve Bayes is a collection of classification algorithms based on Bayes Theorem. He also goes on to say that it is not a single algorithm but a family of algorithms that all share a common principle.



To my surprise this returned the highest accuracy of all algorithms, with an accuracy of 93% and an inaccuracy of 7%. This may be down to the fact that it is a family of algorithms all working together in order to give a greater accurate reading of the data. It also has relatively low and balanced false positive and negative readings in the confusion matrix.

* **functions.SMO**

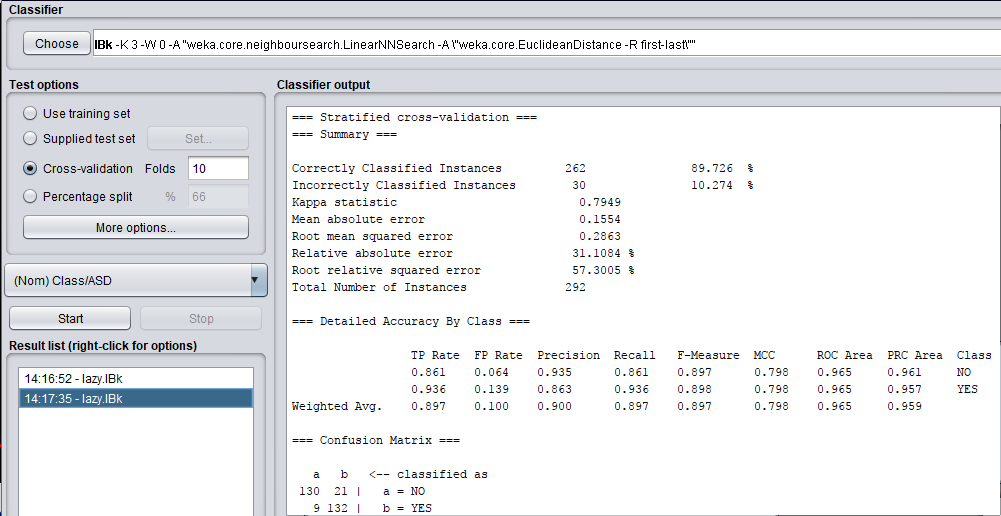
Sequential Minimal Optimisation. [www.weka.sourceforge.net](http://www.weka.sourceforge.net) says that SMO implements John Platt’s SMO algorithm for training a support vector classifier.



SMO returned an accuracy of 99% getting only 1 out of 292 instances incorrectly classified, which at first I thought was strange as there is a certain amount of missing data in the dataset but as I researched SMO as a machine learning algorithm I went on to find out by [www.weka.sourceforge.net](http://www.weka.sourceforge.net) that the Weka implementation of SMO globally replaces all missing values and transforms nominal attributes into binary ones which is the reason behind it having such a high accuracy. Its false positives and negative readings in its confusion matrix are nearly perfect.

* **lazy.IBk - using the k value you have determined in 4.**

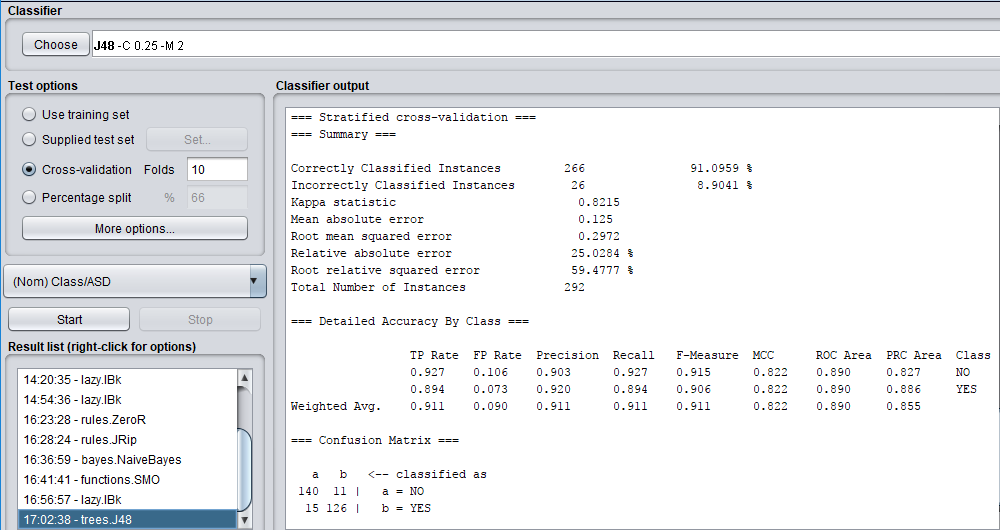
K-nearest neighbour classifier. [www.weka.sourceforge.net](http://www.weka.sourceforge.net) says with IBK you can select the appropriate value of K based on cross-validation or also do distance weighting.



Using K = 3 from before, IBK (kNearestNeighbour) returns a reasonably high accuracy in relation to the data and also based on its neighbours with 262 instances being correctly classified compared to the 30 instances (10%) who are incorrectly classified by the K = 3 kNearestNeighbour algorithm.

* **Trees.J48**

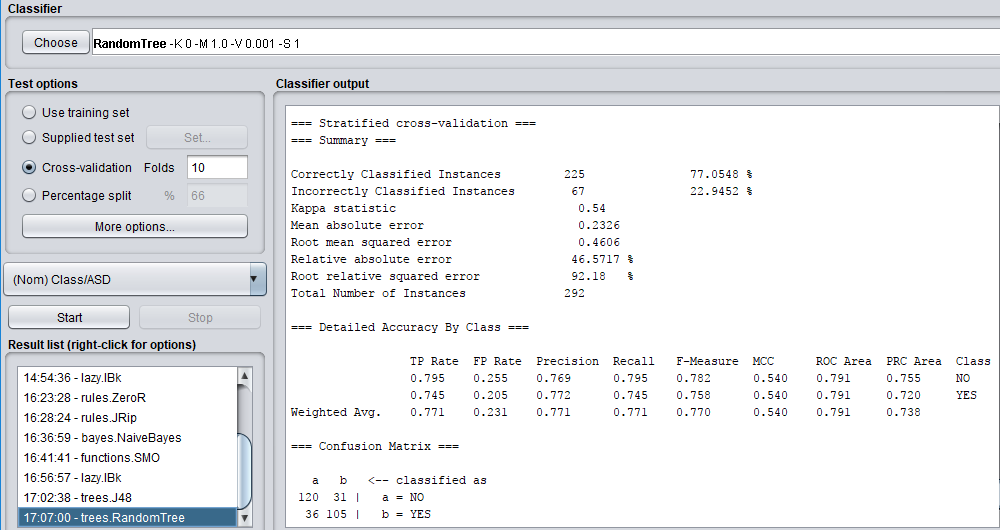
[www.weka.sourceforge.net](http://WWW.WEKA.SOURCEFORGE.NET) says that J48 is a class for generating pruned or unpruned C4.5 decision tree. C4.5 itself being an algorithm to generate a decision tree developed by Ross Quinlan.



We can see that the J48 algorithm returned 266 correctly classified instances out of 292 total instances, which was a solid 91% of the total data. It also only a had 9% inaccuracy of incorrectly classified instances which was 26 of the 292 instances.

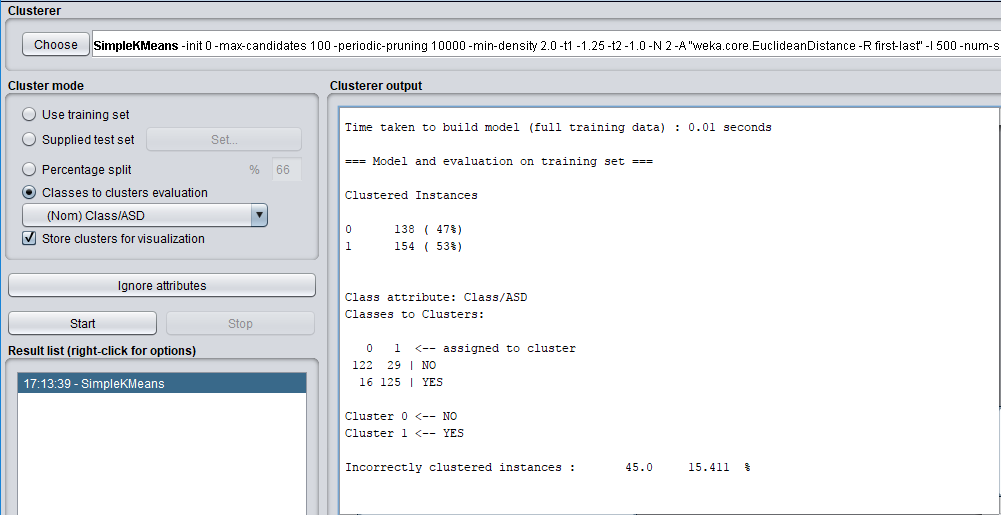
* **Random Forest Algorithm**

After running the algorithm on the Autism-Screening-Child dataset the Random Forest algorithm returned a value of 77% (225 of the 292 instances in the dataset) for correctly classified instances and also returned a large 23% of incorrectly classified data which is nearly a quarter of all the data in the dataset incorrectly classified. I expected a more correctly classified dataset from the Random Forest algorithm. It may be due to the missing data in the dataset that explains why the accuracy is so low compared to other algorithms. Also, after running this on my dataset with 292 and seeing the results I came to the realisation that maybe the Random Forest algorithm is best suited for datasets with a large number of instances instead of a relatively small number of instances.



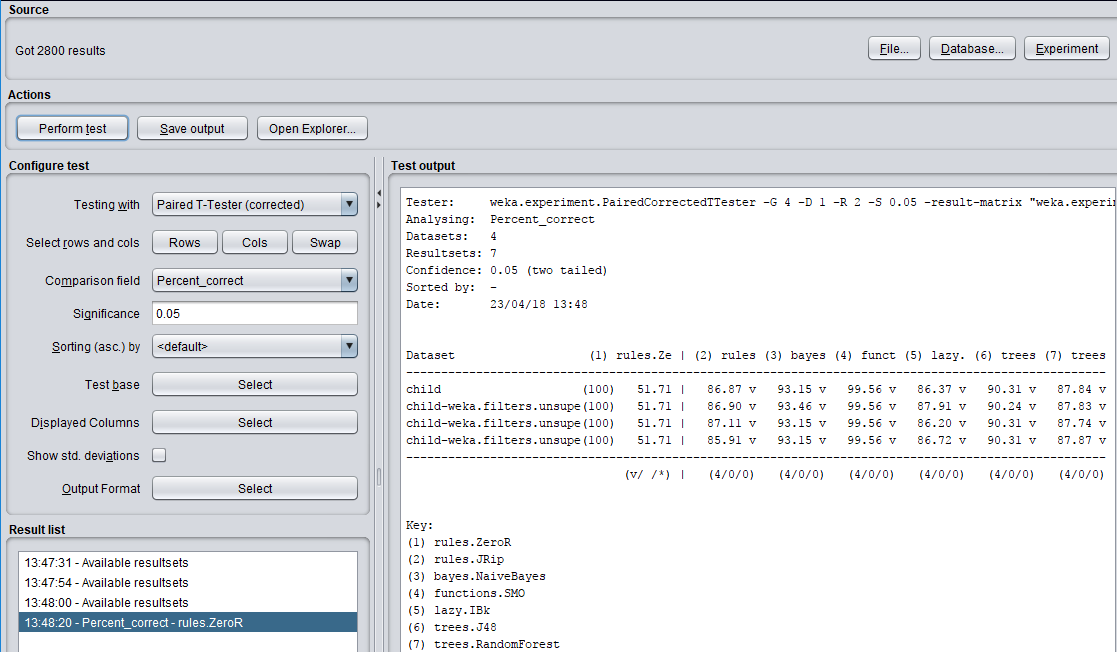
* **K-means Clustering Algorithm**

The K-means clustering algorithm returned 45 incorrectly clustered instances where as the K-nearest neighbour returned 30 incorrectly classified instances. Thus proving for this dataset that K-nearest neighbour is a more suitable algorithm in terms of reading the data and providing accurate readings based on the data. The K-means clustering algorithms confusion matrix is also slightly not as good as the K-nearest neighbours but still within good reason compared to other algorithms confusion matrix’s (i.e. where K = 15 on the K-nearest neighbour algorithm, or the ZeroR algorithm.)



After carrying out my initial evaluation on the machine learning algorithms I have come to the conclusion that the best performing algorithm I have studied is the functions Sequential Minimal Optimization algorithm as it gave me a value of 99.7% for correctly classified instances which was all the instances bar one. This was the best out of all the algorithms I had tested. K-nearest Neighbour was definitely the second best. Its ability to allow you to change the value of K in accordance to what dataset you have, how much data is in the dataset and what you are trying to achieve with the data helps give you a better picture of the completeness of the data, all depending on how many neighbours you choose the algorithm to run with. I found out that with my data of 292 rows the higher the value of K was the worse the accuracy and confusion matrix reading I got back. The lower the value of K the more pinpointed accuracy I got back from the algorithm. This is completely my opinion based on the dataset I am using which consists of 292 rows and this evaluation on these algorithms may have returned completely different answers had it been on any other dataset with many more rows.

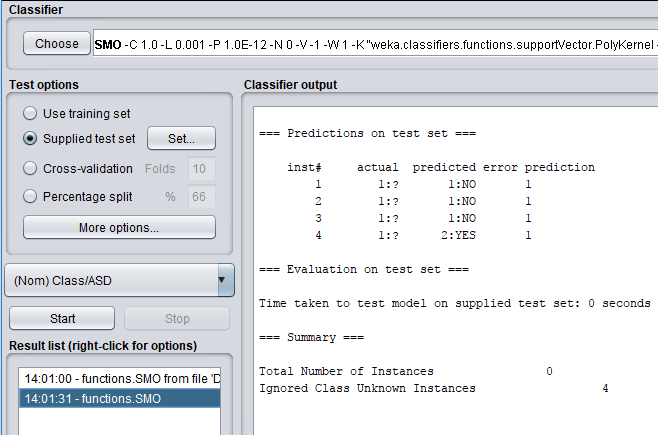
I then loaded the four edited arff files in the Weka Experimenter. I also loaded each the algorithms in the experimenter and ran a test on all of the algorithms on all four of the arff files. Once again after looking at and analysing the results It was clear to me that the functions algorithm ‘SMO’ was the best performing algorithm across the board. It had a 99% correctness for all four datasets. Naïve bayes algorithm was second best here as it had 95% correctness for the datasets. The worst performing algorithm on the four datasets was rules algorithm ‘Zero’. It maxed out at 51.71% on all four files. This means that only half of the data was correctly classified for this algorithm. The results can be seen below.



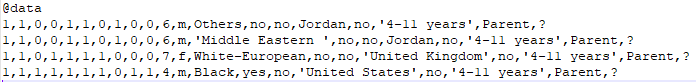
### **Final Version of the Model and Present Results**

I ran the SMO algorithm along with the unseen dataset as I found out previously that this had the best percentage of correctness throughout all four of the datasets (original, normalised, standardised and missing). This meant that I would get the most accurate and precise reading of whether the unseen to patients had autism or not.

The unseen dataset consists of 4 patients who have not yet been diagnosed so for this I decided to run the best algorithm I had found to establish a predictor of whether the unseen patients had autism or not. I found that instance/patient 4 had been diagnosed with autism (as per SMO algorithm).



This patient was a 4 year old black male from the United States. Looking in the actual unseen dataset you can see that this patient gave an answer of 1 for 9 out of the 10 questions asked. Much more than any other patient in the unseen dataset.



## Section 2.

**\*ALL SCREENSHOTS FROM THIS SECTION CAN BE FOUND IN THE APRIORI\_ALGORITHM\_ITEMSET\_BREAKDOWN.xlsx EXCEL SPREADSHEET.**

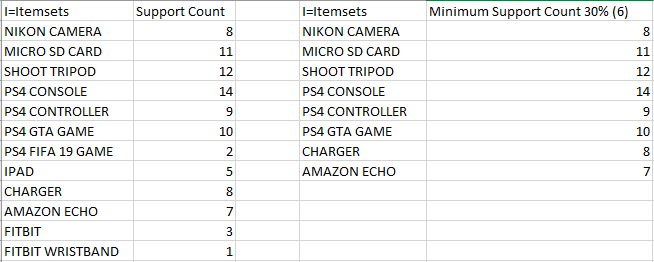
Local supermarket with 12 items available for purchase in set I{}. Each customer purchases different subsets of I. Each customer transaction tracked, showing which items purchased.

I = {NIKON CAMERA, MICRO SD CARD, SHOOT TRIPOD, PS4 CONSOLE, PS4 CONTROLLER, PS4 GTA GAME, PS4 FIFA 19 GAME, IPAD, CHARGER, AMAZON ECHO, FITBIT, FITBIT WRISTBAND}

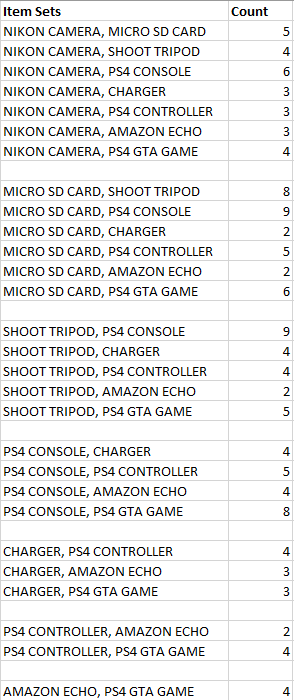
Let D = set of transactions {T1, T2, … T20} Each T represents a set of items contained in I.

### **Applying Apriori Algorithm.**

First I counted how many times the items appeared in all 20 transactions. I then discarded all items outside of the minimum support of 30% (appearing less than 6 times).

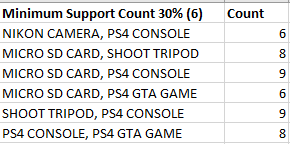


I then matched each item with every other item that was above the minimum support level.



This game me a unique grouping of all items in sets of 2 that had over 6 appearances in the transaction list on their own. The count to the right of the above list is how many times the set of the two items appears on the transaction list. E.g. Nikon Camera appears with Micro SD Card in a single transaction 5 times, therefore they have been bought together 5 times.

The same thing was then done again on the sets of two. All sets with less than 6 appearances were dropped from the list which then gave me the following list:



I then had to make up sets of 3 out of these sets of 2. If the two sets shared a common item and had over 6 transactions (above the minimum support) then they could combine to make an item set of 3 which I ended up getting the following:

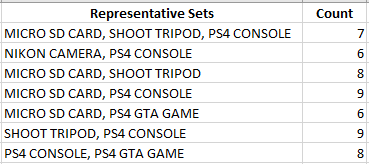


Applying the minimum support to this list you end up with only one grouping which is the following:

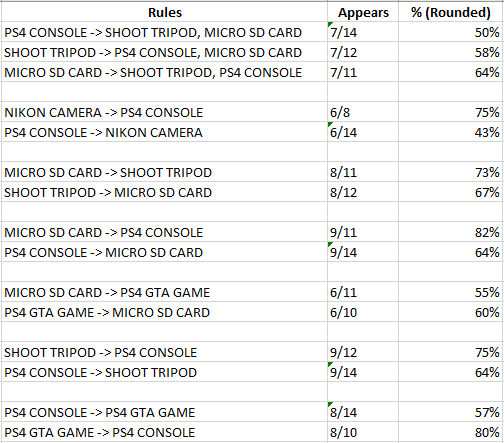


This is the most common grouping in the transaction list. All three of these items appear together 7 times.

With all this done, this then gave me my representative sets.

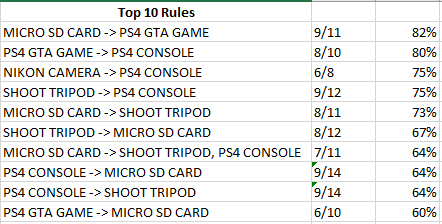


With the representative sets I then got the association rules.



### **Generate and identify the top 10 association rules that maximise support and confidence of the rule.**

After getting the rules from the data I then got the top 10 rules out of the total amount of rules.



### **Discuss and interpret the significant rules you have identified.**

The top rule is as follows, out of the 11 times a Micro SD Card was bought, a PS4 GTA Game was bought 9 out of them times which gives me a percentage of 82%. Thus meaning, 82% of the time a Micro SD Card is bought, the PS4 GTA Game is also bought.

The second most popular rule is when 10 PS4 consoles were bought, 8 PS4 GTA Games were also bought along with the console. Thus meaning, 80% of the time a console is bought the game GTA is also bought with it. This rule makes sense as you would need to play a game on the console so customers must be choosing GTA at a high rate along with their PS4 consoles.

3 of the 10 rules have the same percentage at 64%.

One of the rules involves 3 items. When a Micro SD Card was bought, 64% of the time the Shoot Tripod and a PS4 Console were also bought with it. This is the only rule that includes 3 items, all the rest have only 2 items involved.

The last rule appears at a rate of only 60% compared to the top rule at 82%. When the PS4 GTA Game was bought 10 times the Micro SD Card was also bought along with it 6 of them 10 times.

## Bibliography

**Machine Learning Mastery.** 2018. *How to Normalize and Standardize Your Machine Learning Data in Weka*. Author: Jason Brownlee. [ONLINE] Available at: <https://machinelearningmastery.com/normalize-standardize-machine-learning-data-weka/>. [Accessed 10 April 2018].

**Data mining with WEKA, Part 3: Nearest Neighbor and server-side library. 2018.** *Data mining with WEKA, Part 3: Nearest Neighbor and server-side library.* Author: Michael Abernathy. [ONLINE] Available at: <https://www.ibm.com/developerworks/library/os-weka3/>. [Accessed 13 April 2018].

**Toptal Engineering Blog. 2018. Ensemble Methods in Machine Learning | Toptal.** Author: Necati Demir. [ONLINE] Available at: <https://www.toptal.com/machine-learning/ensemble-methods-machine-learning>. [Accessed 14 April 2018].

**Towards Data Science. 2018. The Random Forest Algorithm – Towards Data Science.** Author: Niklas Donges. [ONLINE] Available at: <https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd>. [Accessed 14 April 2018].

**Introduction to K-means Clustering. 2018*.****Introduction to K-means Clustering.* Author: Andrea Trevino. [ONLINE] Available at: <https://www.datascience.com/blog/k-means-clustering>. [Accessed 15 April 2018].

**ZeroR.** 2018. *ZeroR*. [ONLINE] Available at: <http://chem-eng.utoronto.ca/~datamining/dmc/zeror.htm>. [Accessed 17 April 2018].

**JRip.** 2018. *JRip.* [ONLINE] Available at: <https://www.eecs.yorku.ca/tdb/_doc.php/userg/sw/weka/doc/weka/classifiers/rules/JRip.html>. [Accessed 17 April 2018].

**AYLIEN.** 2018. *Naive Bayes for Dummies; A Simple Explanation - AYLIEN.* Author: Mike Waldron. [ONLINE] Available at: <http://blog.aylien.com/naive-bayes-for-dummies-a-simple-explanation/>. [Accessed 17 April 2018].

**SMO.** 2018. *SMO*. [ONLINE] Available at: <http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/SMO.html>. [Accessed 17 April 2018].

**IBk.** 2018. IBk. [ONLINE] Available at: <http://weka.sourceforge.net/doc.dev/weka/classifiers/lazy/IBk.html>. [Accessed 17 April 2018].

**J48**. 2018. J48. [ONLINE] Available at: <http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html>. [Accessed 17 April 2018].