Data Science Algorithms & Tools

Student Number: 27016005

Data Science Algorithms & Tools: CS3DS19

Hours Spent on Assignment: 20

Date of Completion: 21/03/2021

Assignment evaluation:

1) This assignment was quite interesting and fun. It prompted more extensive research into this topic area, and was extremely insightful into a side of Computer Science i had not previously explored in depth.
2) More detailed guidance could have been provided - possibly in the form of an in-depth mark scheme / criteria or clearer goals.
3) The collision of so many deadlines at the same time made it hard to enjoy / focus.

Table of contents:

GitHub Repo: https://github.com/Dannybrush/CS3DS19

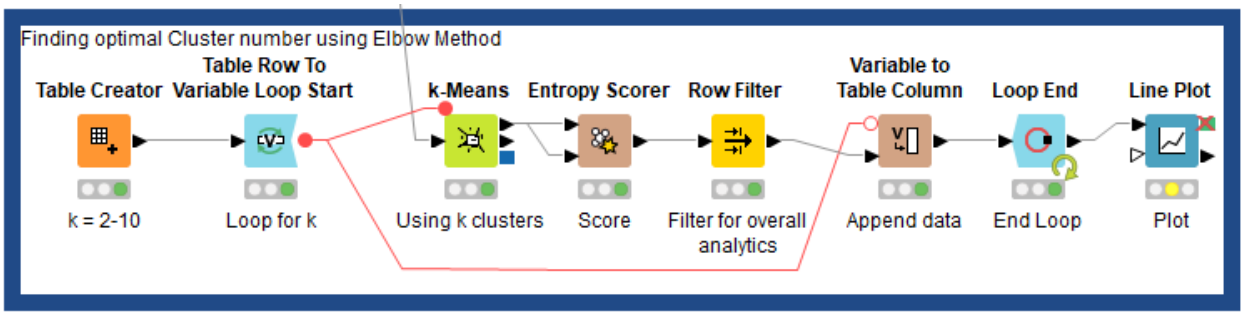
Task 1:

Description of Clustering algorithm, Cluster Validity Measure & Workflow:

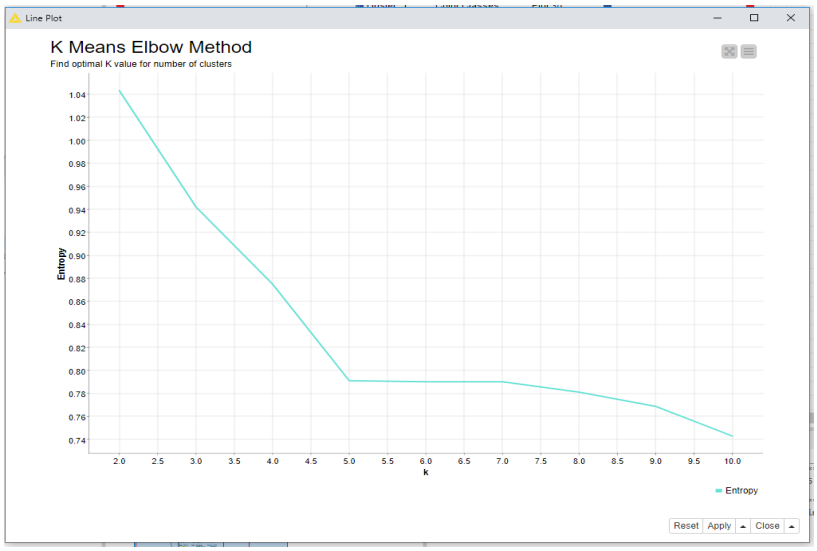
The Wine Dataset was imported, a table view node was added to show this in its entirety. A principal component analysis node was then applied to reduce the dataset to two dimensions, this was then outputted as a scatter plot.

The scatter graph shows three distinct groups of data, distinguished by their colours, with Yellow representing class 1, grey representing class 2 and blue representing class 3. Class 1 lies predominantly in the lower left quartile, as opposed to class 2 and three which are strongly mixed in the bottom right region. The main bulk of the data lies between -25 and +30 on the X-axis with all of the outliers (on both extremes) belonging to class 2.

The K-means method of clustering was chosen to be applied to the data. To find the optimal number of clusters to use, the "elbow method" was implemented.

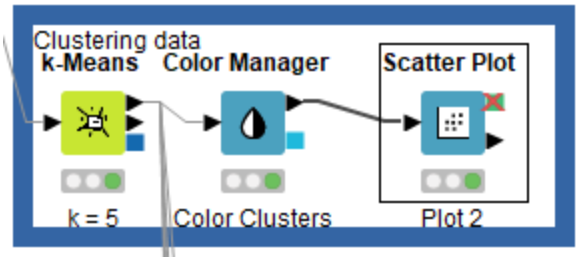


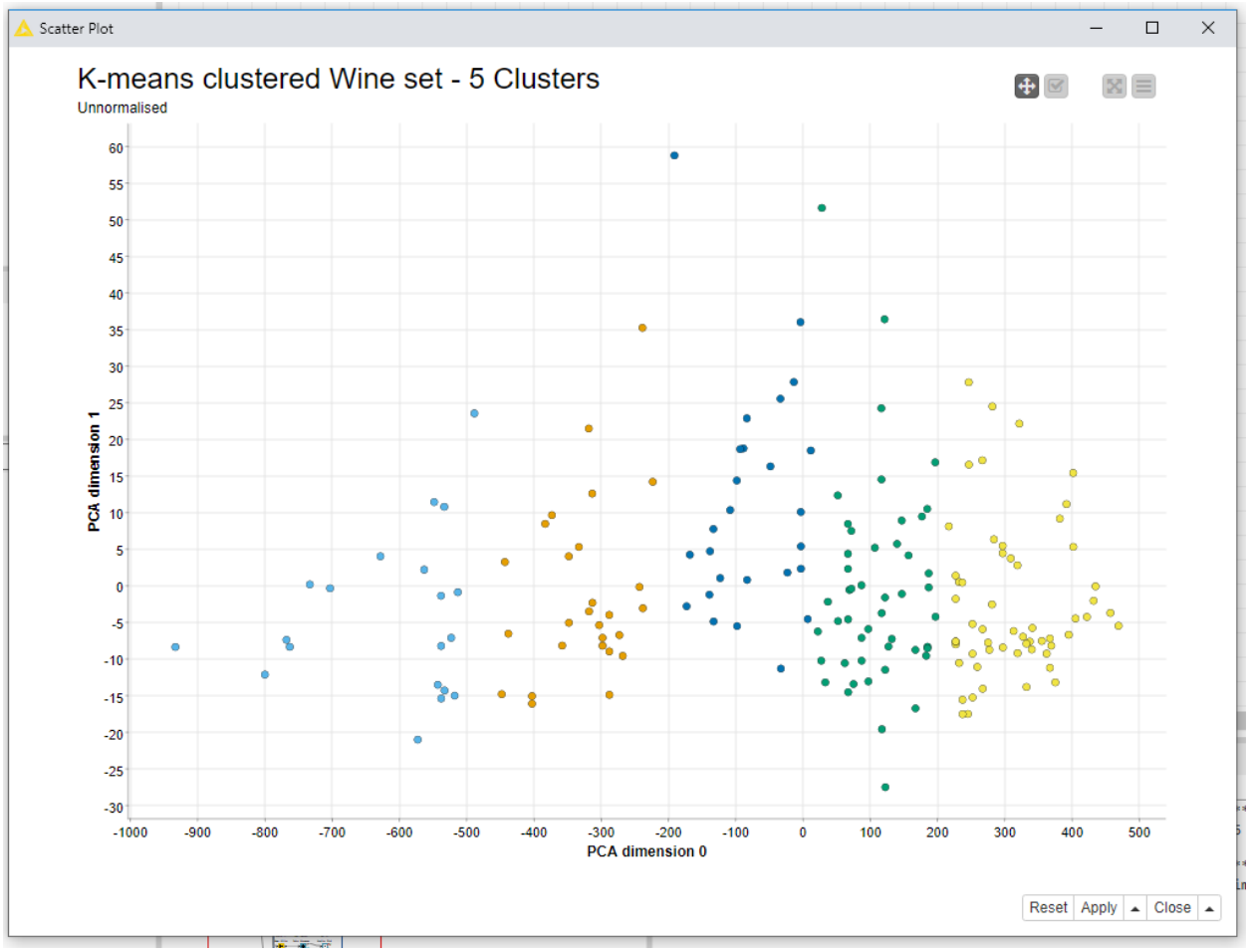
Producing the following graph:



The elbow method works by generation clusters for k values between a given range, scoring the entropy of each iteration and plotting it on a line chart, using visual inspection to find the most significant change in gradient, and thus the optimal number of clusters. As shown on the graph above, this occurs at k = 5.

K-means clustering was then applied using this value of k =5 representing 5 clusters. This was again represented as a scatter plot, with each predicted cluster classification being distinguished by their own colour.



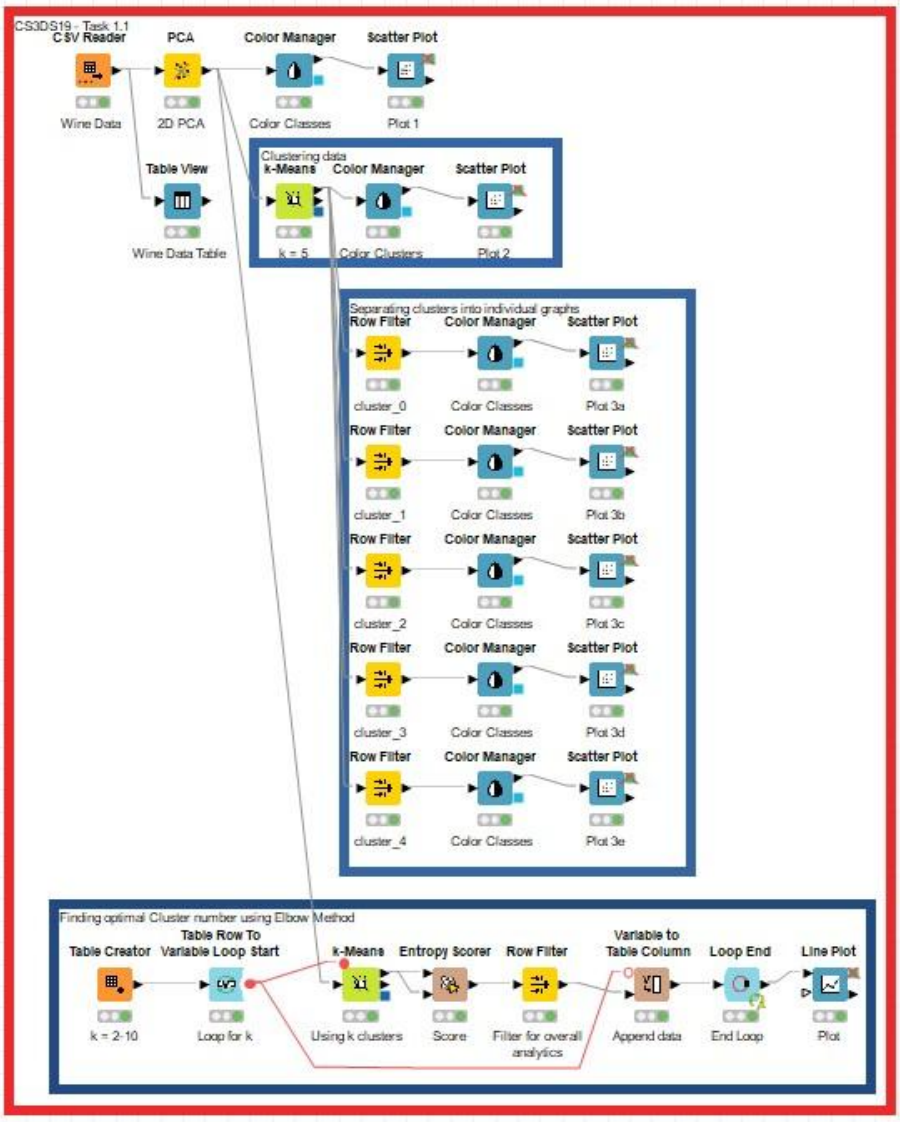


The graph clearly shows 5 distinct groups of clusters, bounded by regions on the X-axis

Where cluster 0 is assigned to values between -1000 and -475, cluster 1 is assigned -475 to -200, cluster 2 possessing any values between -200 and 20, cluster 3 being assigned values falling between the regions of 20 and 200 and cluster 4 being given any values surplus of 200.

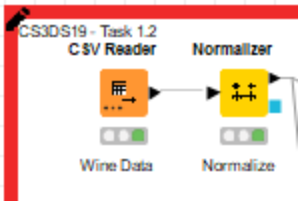
Each of these clusters are then separated into their individual groups and plotted on their own scatter graphs, coloured respective to their true class. The graphs for this have been added to the appendix.

As shown from these scatter graphs it can be inferred that the cluster 0 and 1 consist of solely class 1 data points, cluster 2 shows a strong mix of class 2 and 3 data points, including a single anomalous data point belonging to class 1. Cluster 3 shows predominantly data points belonging to class 2 with a significant inclusion of class 3 data points. Cluster 4 contains a mix of values from all 3 classes.

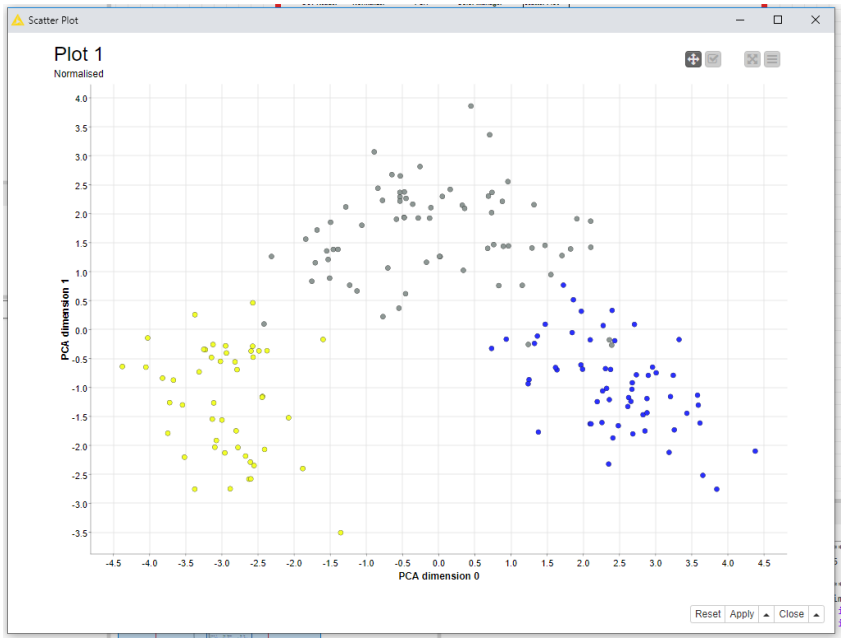


Task 1.2:

An option not previously performed, was the possibility of normalising the data prior to processing and classification. The entire workflow remains very similar for the classification process, however includes a Knime Normaliser node immediately after reading the data.

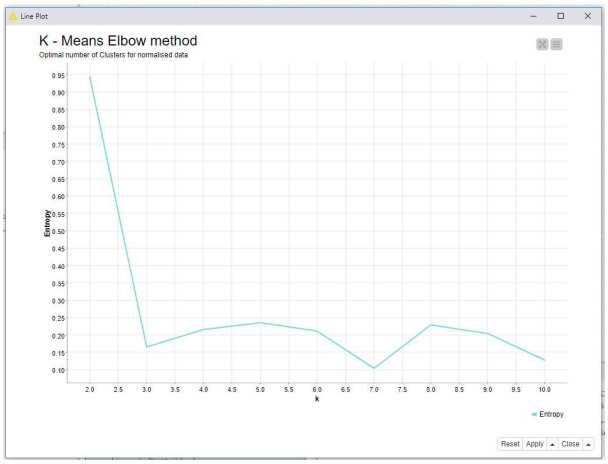


The Knime normaliser node uses a “Z-score” normalisation method to transform data to be within a certain number of standard deviations away from a mean value. The normalised data produces a significantly different plot, showing three clear groups of data, albeit not fully separable.

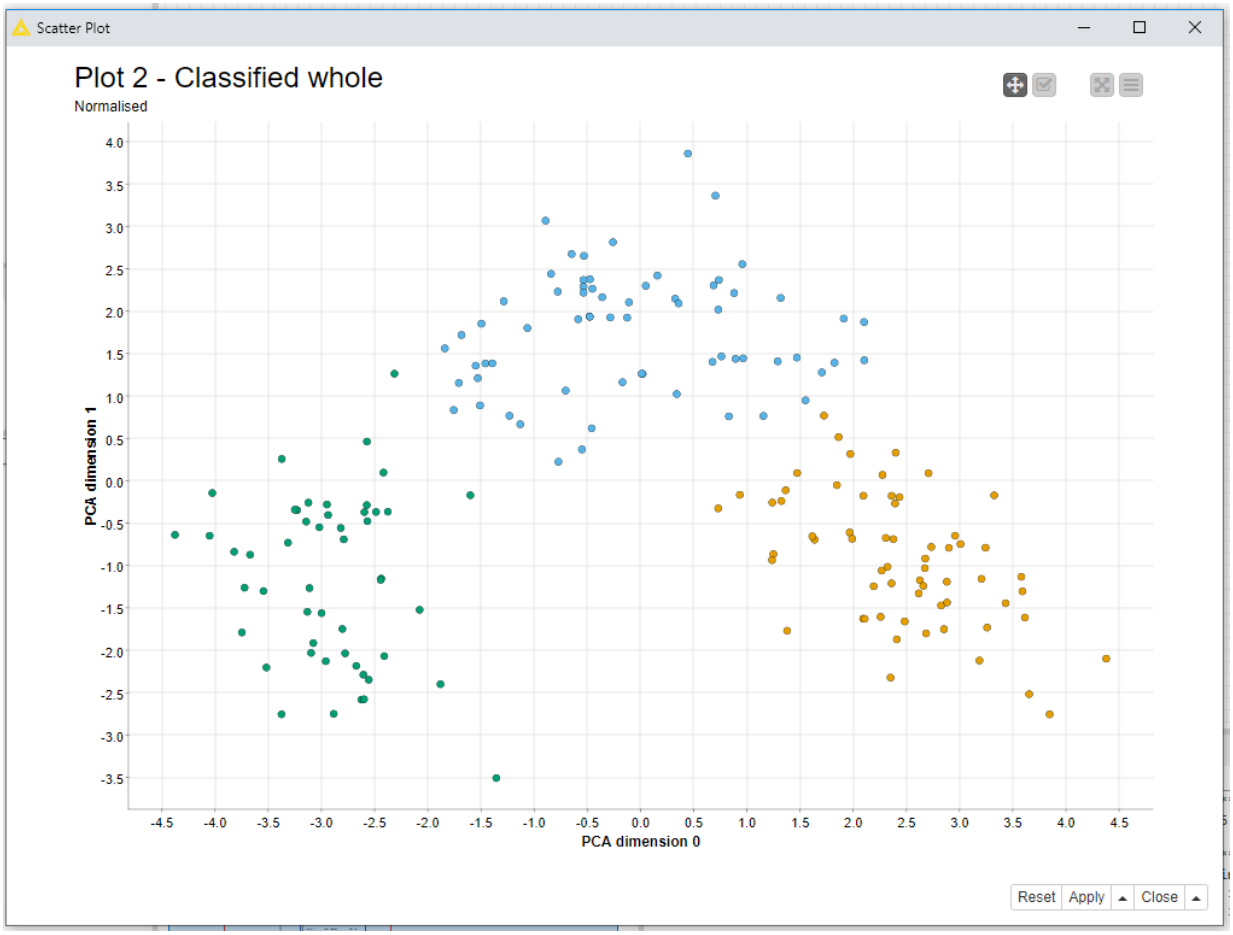


The data now shows class 1 occupying the bottom left quadrant, the class 3 occupies the bottom right quadrant and class 2 occupying the top centre region.

The normalised data was then run through the same “Elbow Method” workflow as before, however this time the resulting optimal cluster value of k = 3.



The normalised data is then again classified using a K-means method, this time using the optimal value of k = 3.



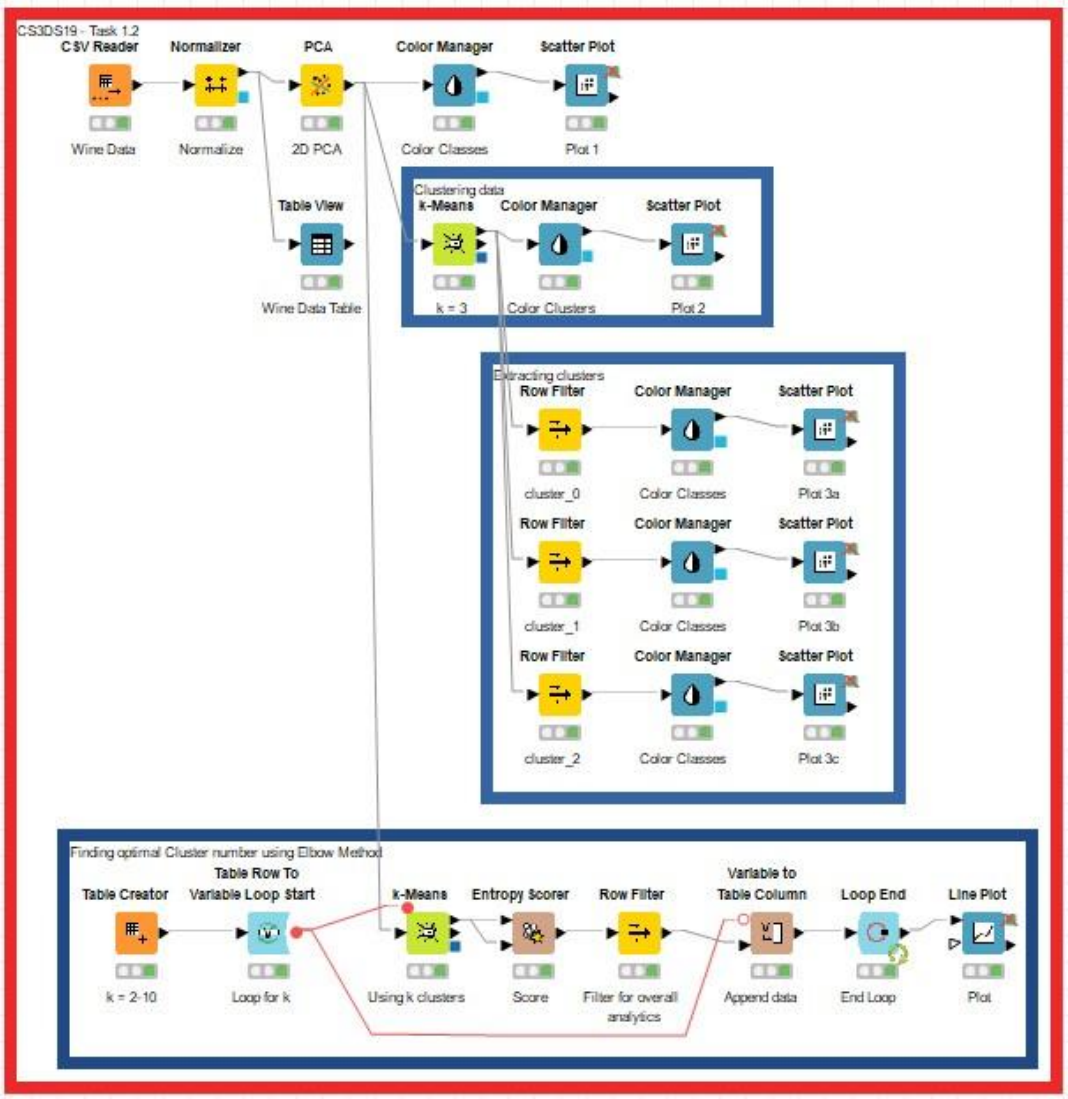
This Graph shows a much more clear divide between the three proposed classes, not too dissimilar to the true classes. This was again separated into the 3 individual proposed classifications, with the colours representative of the true classes.

Cluster 0 demonstrates an effective recognition of class 1 data points, encompassing 100% of the true class 1 values, however also including 3 data points belonging to class 2.

Cluster 1 effectively classifies class 2 with 100% of points in the cluster being true class 2. However not all true class 2 values are assigned to the Cluster 1.

Cluster 2 contains 100% of the true class 3 values, plus 2 values belonging to true class 2. Showing very strong classification results.

This shows a correct clustering accuracy of 97.2%.



Conclusion:

In conclusion, normalisation of data seemed to have a strongly positive effect on the overall outcome, reducing the amount of optimal clusters to the correct amount. The normalised workflow also produced a much more accurate result, correctly classifying 97.2% of the data into the correct clusters.

Task 2:

Description of classification algorithms adopted (2):

The two chosen classification algorithms adopted were the Naive Bayes Classifier, and the Logistic Regression Algorithm.

Naive Bayes Classification foundationally relies on the Bayes Theorem [Appendix2A] , a mathematical formula for determining conditional probability [7]. A Naive Bayes Classifier is a machine learning model which is mostly used in sentiment analysis, spam filtering and recommender systems. The largest drawback of Naive Bayes classification is the fact that it relies on the predictors to be independent, and in most real world applications this is not the case[8]. Dependent predictors can significantly affect the performance of this classifier.

In this implementation a Naive Bayes Classifier will individually calculate the probability that a wine belongs to a class for each isolated feature, assuming complete independence of variables. These are then aggregated together and the wine is associated to the predicted class with the highest correlation.

Logistic regression is an alternative method for classification, stemming from "supervised learning" methodologies [5]. The foundation of Logistic regression focusses around a sigmoidal function with the creating a "S" shaped curve with extremities tending toward negative infinity and positive infinity respectively. Naturally, the output of a logistic regression function is a binary output, usually class A or class B, however it can be adapted to iteratively produce the probability of Class X or NOT Class X, by performing this for each proposed class, it can estimate that the Class that the input belongs to, and which ones it doesn't, changing the weights where needed, allowing an effective classification.

Description of 10-fold Cross-Validation method:

K-fold cross validation is a statistical method commonly used to estimate the skill of a machine learning model. Benefits of using this method rely around the fact that it is easy to understand, easy to implement and often results in skill estimates that generally produce a lower bias than alternative methods [4].

K-fold validation often starts by first shuffling the dataset, then splitting the data into k subsets, for this instance the value of k = 10 was chosen, henceforth this will be referred to as 10-fold cross-validation.

This works by splitting the data into 10 subsets, using one subset for testing and the other nine for training, and repeating this process until all subsets have been tested. The benefit of using this method means that each data point is used for both training and testing, however the data point is never used for training and testing in the same iteration.

Data Workflow Description:

The dataflow has been split into 3 sub-flows, representing the Preprocessing, Training, and Prediction.

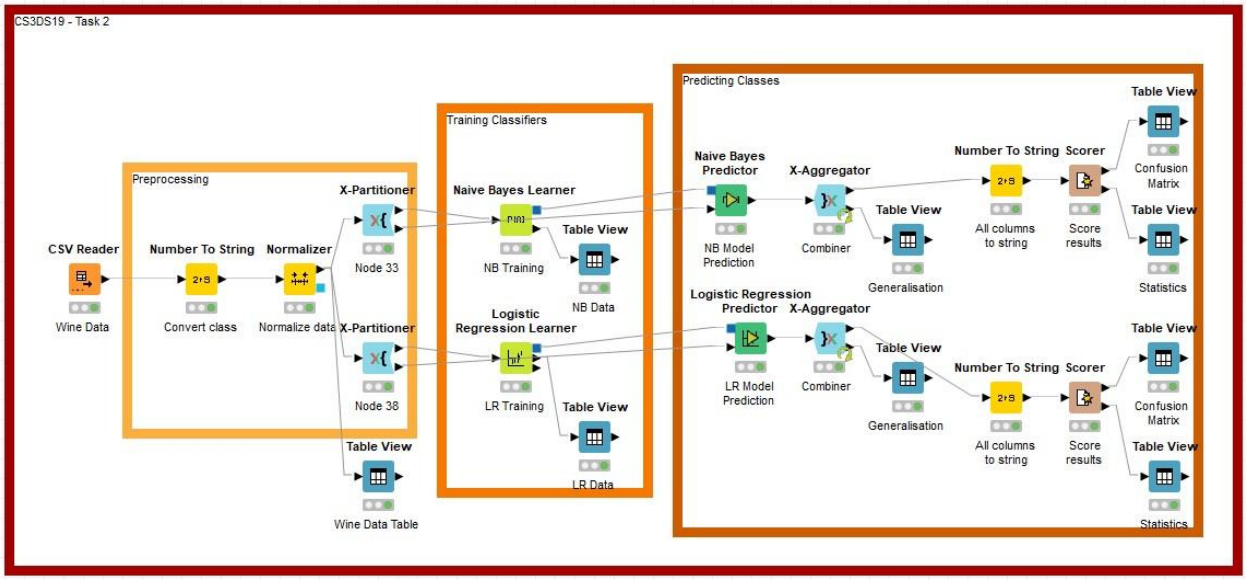
The data is first loaded in using the CSV Reader Node, then passed to the preprocessing section, where the "Class" attribute is first converted to a string using the Knime module "Number to String", then the data is normalised using Z-Score normalisation. After which the data is split into two identical branches. One branch will be passed into a Naive Bayes Learner and Predictor, whereas the other will be passed into a logistic regression learn and predictor nodes. Each branch is then fed into a "X-Partitioner" node, which instantiates the loop for the 10 fold Cross validation. The node creates 10 subsets of data causing stratified sampling, this node has two outputs, one feeding into the learning node, and one feeding into the predictor node.

The learning node generates a trained model which feeds back into the predictor node which then outputs a table, adding a field for the predicted classification. The "X-Aggregator" node is then used to collate the results from the Xfold validation loop. There are two output branches for each "X-Aggregator" one of which leads to the output for each individual loop and then one which leads to the prediction table. The prediction table is then passed through a "Scorer" node to produce a confusion matrix and statistic table.

Experimental Results (Comparative Performance Analysis):

The Naive Bayes model proved to be very effective, with 50% of the tests having only a single error, in an sample size of 18 (17 for two tests) giving a maximum error rate of 5.55% in a test, and an average error rate of 2.76% over the 10 iterations. Half of the iterations also produced 100% accuracy. Upon investigation of the confusion matrix [APPENDIX 2B] it is apparent that the Naive Bayes model produced a 97.2% accurate solution, with 5 errors in the total 178 values, with 3 values being incorrectly identified as class 2, and 2 values being incorrectly identified as class 1.

The Logistic Regression managed to surpass the Naive Bayes model producing only 3 errors in the 178 values, across 3 separate iterations, resulting in a similar maximum error rate per iteration of 5.55% with an overall average error of 1.667% and 70% of the iterations having no errors. Investigation of the confusion matrix [APPENDIX 2C]revealed that all 3 incorrectly classified results, were false positives for the second class, with one being from true class 1 and two being from true class 3. This means the logistic regression model had a 98.3% accuracy.



Conclusion:

To summarise what has been demonstrated, the Logistic Regression model performs marginally better overall with a 98.3% accuracy when compared to the 97.2% accuracy achieved by the Naive Bayes model. Both models seemed to produce the most errors with incorrectly classifying true class 3 items into predicted class 2.

Both of these models make use of probability methods in order to create the model, and therefore more significant differences may have been obtained if a different style of classifier was chosen, such as an MLP.

Both of these models seem to produce sufficiently effective solutions.

Task 3:

Description of Data Mining Algorithm, the solution (data workflow) adopted, and predicted performance indices.

One complication that arose while attempting task 3 extends from the nature of the data. For a problem of this kind, the dataset is extremely unbalanced. With the "Signal" class making up approximately 0.0000003% of overall cases, and "Background" class making up the rest. The Training data consists of 100,000 records, in which 90,000 make up the "Background" class and 10,000 (10%) make up the "Signal" class. This inherent imbalance of the dataset creates a significant skew to the learning of the classifier.[1] Under first impression, a 90% accuracy is perceived from training and validating the model, however upon investigation, the classifier fails to correctly identify any of the signal class, which is the primary purpose for this application. The seemingly high accuracy stems from the fact that the background class makes up so much of the dataset meaning that by purely classifying the background class values correctly, the classifier can achieve an ostensibly high score, without being able to achieve the functionality it is intended for.

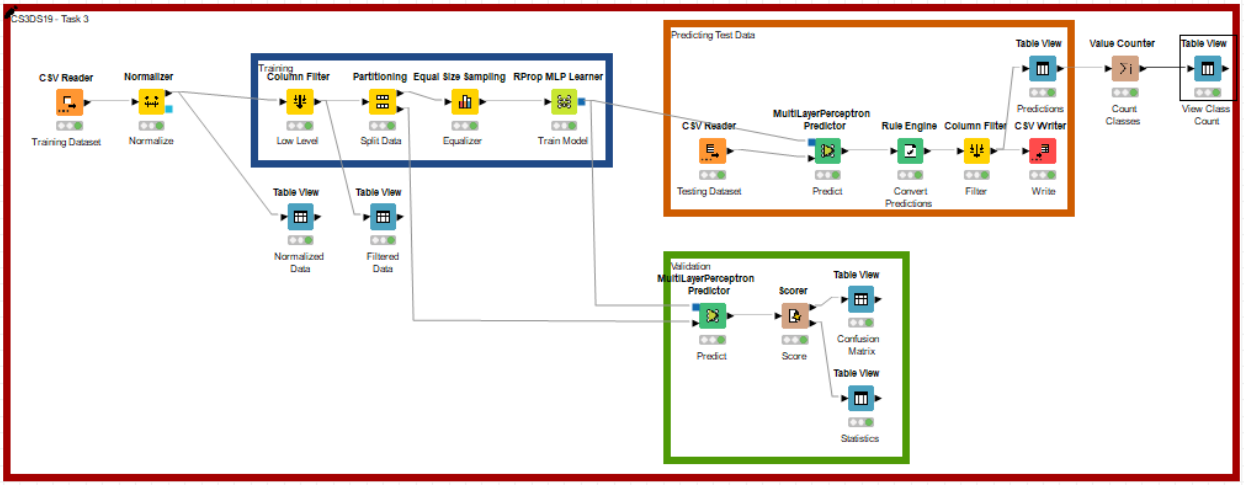
There are two main methods used for countering imbalanced data, Oversampling,[3] and undersampling,[2] each has distinctive pros and cons. Oversampling functions by randomly generating samples in order to "inflate" the minor class, the most common method of this is to use **SMOTE** (**S**imple **M**inority **O**versampling **TE**chnique)[1] which addresses the imbalance by creating new samples through interpolation of existing samples and their neighbours, while this creates a small problem due to the overlapping classes it significantly decreases the chance of overfitting the model to the data, which is the likely outcome with another method such as duplication of existing records.

The other main method of dealing with imbalanced data is Undersampling, which conversely takes the opposite approach and removes data linked to the major class until there is a level of balance between the two classes. This does result in a significant loss of data, introducing the possibility of a loss of potentially useful data.

Undersampling has been chosen to be applied in this situation. This is because although it significantly decreases the training data set, it still keeps it to a reasonably large size, and increases the speed at which the process can be completed drastically.

A multiLayerPerceptron classifier was chosen to be implemented, via the “RProp MLP Learner” and “MultiLayer Perceptron Predictor” nodes available in Knime. MLP classifiers use neural networks to perform classifications opposed to probability methods discussed in Task 2. The model learns by adjusting weights connecting the hidden layers of the neural network, using a feedback loop to compare the predicted values to the true values for the testing set, then once the model has learnt for the training set, it can be applied to the testing / validation sets.

For the completion of this task, The workflow is as follows:



First the training data is read using a CSV Reader module, the Data is then Normalised using a Min-Max methodology, after this the last 7 columns which contain the “high-level attributes” are removed, this is because the test data set does not include these 7 values, making it unnecessary to train the MLP with these attributes. From here the training data is partitioned using the Knime “partitioning” node, the purpose of this is to create a “training” portion and a “Validation” portion. The training portion is then passed into a Knime “Equal Size Sampling” node, which performs undersampling, similar to what is mentioned above. From here the undersampled training set can be fed into the MLP learner. The model produced by this learning is then used by two branches simultaneously one leading to an MLP predictor processing the Validation subset, and one leading to an MLP predictor to analyse the unseen Testing data.

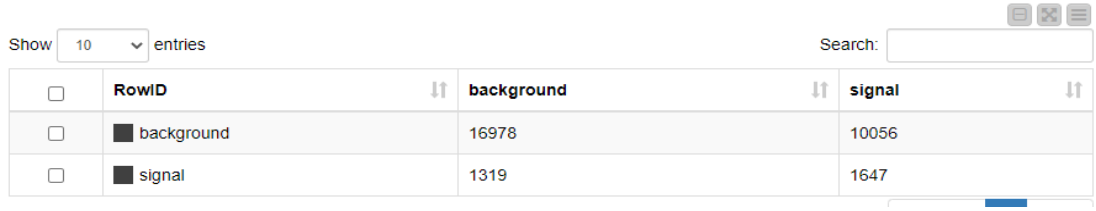
The model processing the validation data is then run through a “scorer” node, and used to produce a confusion matrix and a statistics table.

The model processing the unseen data produces a prediction table which is then passed into the “Rule Engine” node converting the “Background” values into 0 values, and “Signal” values into 1 values. This output is then passed through a “Column Filter” node to extract only the record ID and predicted classification columns, and then written to a CSV file using the “CSV Writer” node. Further data analysis has been included on a parallel branch which

includes a table view to allow viewing of the individual classifications, and a value counter which combines the results to display the overall count of each classification.

Prediction Results (overall accuracy & 3 indices for the class "signal": precision, recall, F-measure).

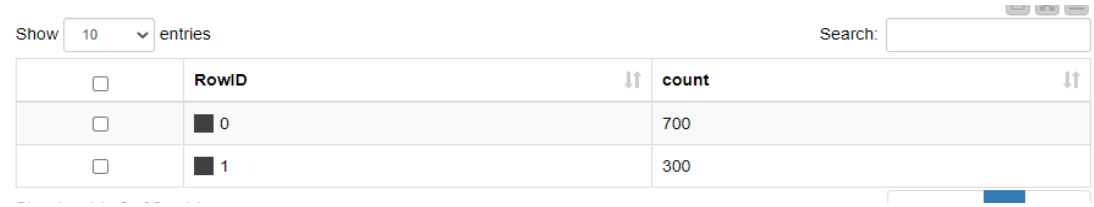
The Confusion matrix produced shows a significantly high false classifications when compared to the validation set.


Show 10 entries Search:

	RowID	background	signal
☐	background	16978	10056
☐	signal	1319	1647

This produces an accuracy of around 62%.

For the testing set, the predicted results are as followed:


Show 10 entries Search:

	RowID	count
☐	0	700
☐	1	300

This predicts for the 1000 samples, 700 of them are "Background" and 300 are "Signal" values.

Conclusion:

An imbalanced dataset proves to be extremely dangerous, appearing to give fairly accurate results while not effectively solving the issue which the system is designed for, and on some occasions, actively producing results which oppose the design of the system while still appearing accurate. This has been addressed through the use of preprocessing and undersampling. In the future, more research could have gone into the possibility of oversampling and various other techniques in order to increase the accuracy, however the time and efficiency payoff did not seem justifiable in this case.

The proposed solution produces an accuracy rate of around 62%, including identifying "signal" class values and while this is significantly better than the prior solution, it is undoubtedly improvable through more intensive research, testing and tweaking.

References

[1]M. Altini, "Dealing with imbalanced data: undersampling, oversampling and proper cross-validation", *Marco Altini*, 2015. [Online]. Available: https://www.marcoaltini.com/blog/dealing-with-imbalanced-data-undersampling-oversampling-and-proper-cross-validation. [Accessed: 23- Mar- 2021].

[2]K. Pykes, "Oversampling and Undersampling", *Towards Data Science*, 2020. [Online]. Available: https://towardsdatascience.com/oversampling-and-undersampling-5e2bbaf56dcf. [Accessed: 23- Mar- 2021].

[3]S. Glen, "Undersampling and Oversampling in Data Analysis - Statistics How To", *Statistics How To*, 2019. [Online]. Available: https://www.statisticshowto.com/undersampling/. [Accessed: 23- Mar- 2021].

[4]J. Brownlee, "A Gentle Introduction to k-fold Cross-Validation", *Machine Learning Mastery*, 2020. [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation/. [Accessed: 23- Mar- 2021].

[5]C. Subasi, "LOGISTIC REGRESSION CLASSIFIER", *Towards Data Science*, 2021. [Online]. Available: https://towardsdatascience.com/logistic-regression-classifier-8583e0c3cf9. [Accessed: 23- Mar- 2021].

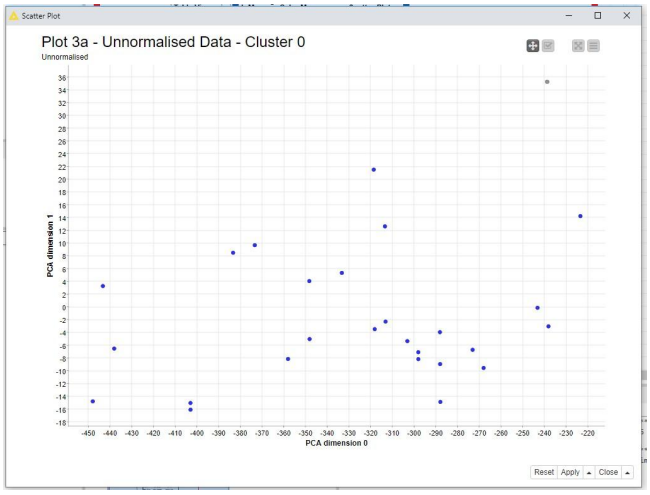
[6]A. Hayes, "Bayes' Theorem", *Investopedia*, 2020. [Online]. Available: https://www.investopedia.com/terms/b/bayes-theorem.asp. [Accessed: 23- Mar- 2021].

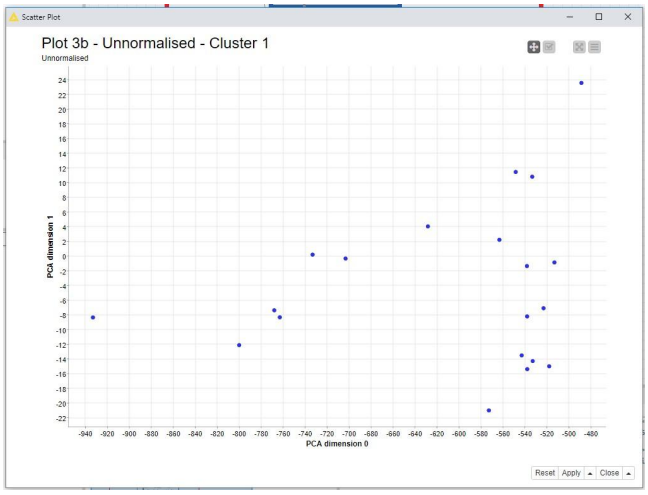
[7]E. S. Yudkowsky, "An Intuitive Explanation of Bayes' Theorem – Eliezer S. Yudkowsky", *Yudkowsky.net*, 2006. [Online]. Available: https://www.yudkowsky.net/rational/bayes. [Accessed: 23- Mar- 2021].

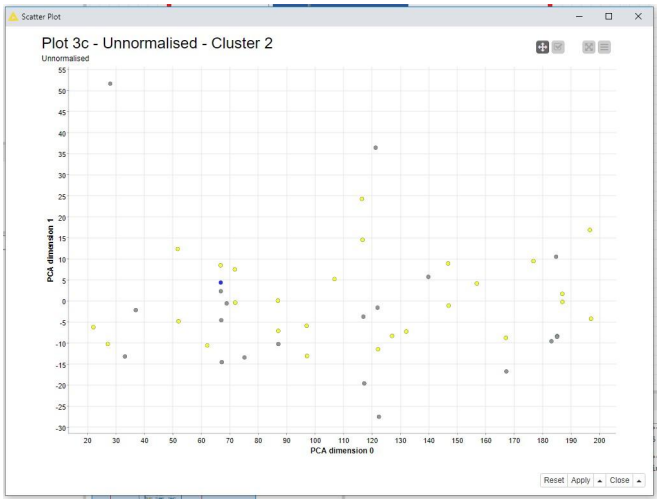
[8] R. Gandhi, "Naive Bayes Classifier", *Towards Data Science*, 2021. [Online]. Available: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c. [Accessed: 23- Mar- 2021].

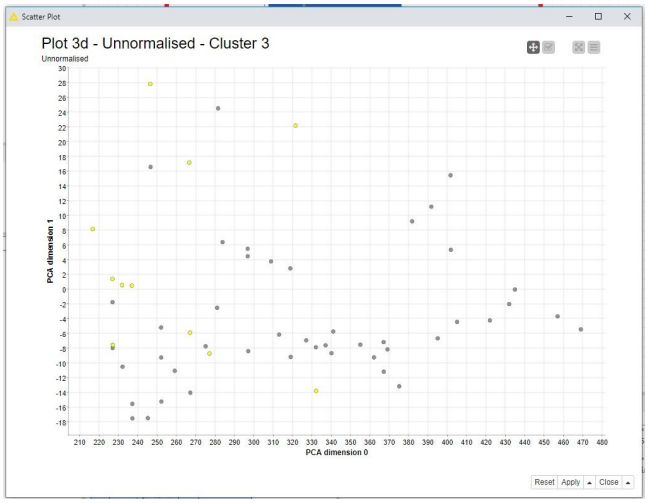
[9] Kotsiantis, Sotiris & Kanellopoulos, D. & Pintelas, P.. (2005). Handling imbalanced datasets: A review. GESTS International Transactions on Computer Science and Engineering. 30. 25-36.

APPENDIX: 1.1





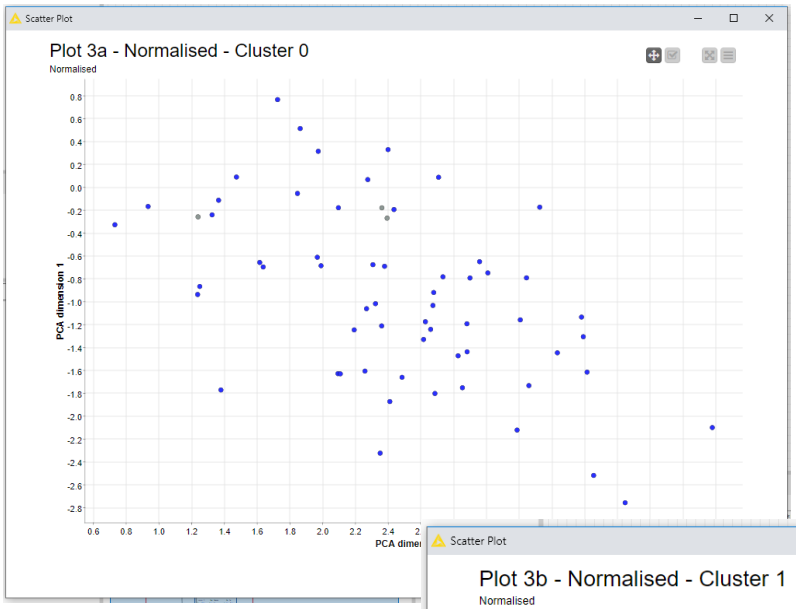


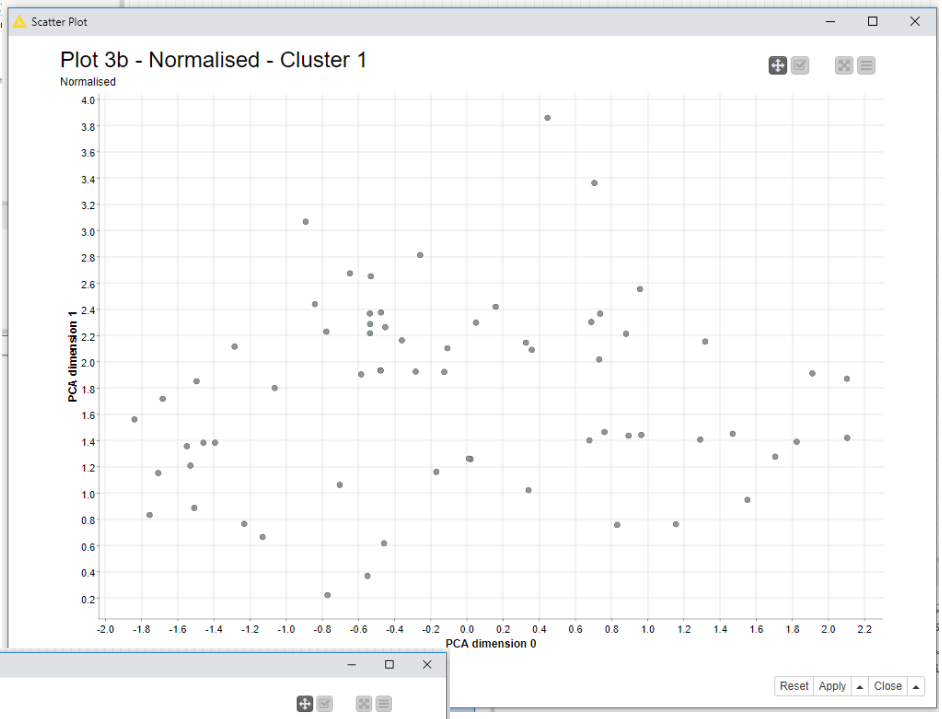


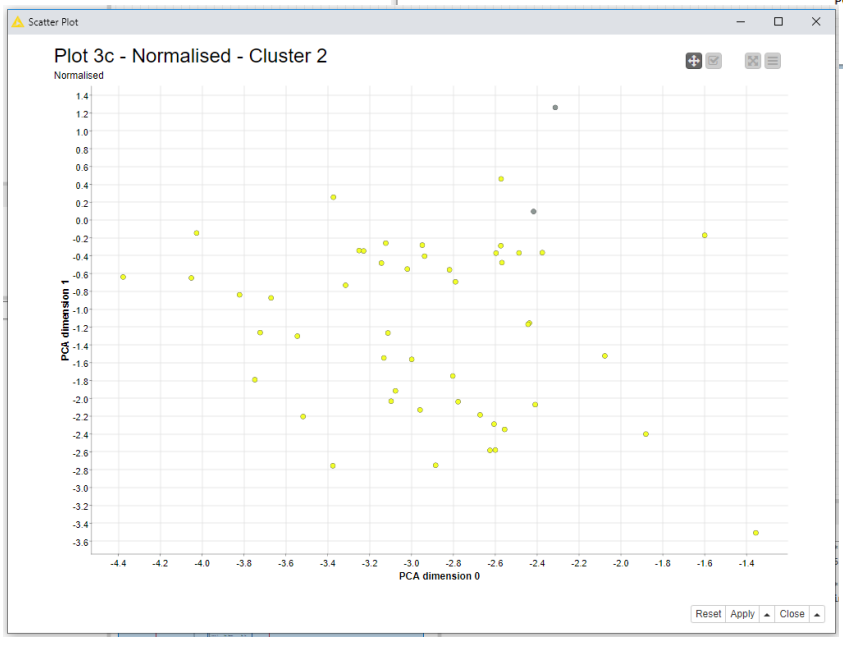
Individual Clusters for unnormalised data: Task 1.1 -> plots 3a - 3e

Appendix 1.2:

Individual Clusters for normalised data:
Task 1.2 -> plots 3a - 3c







Appendix 2A:

Formula for Bayes Theorem, taken from [6]

Formula For Bayes' Theorem

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

where:

$P(A) =$ The probability of A occurring

$P(B) =$ The probability of B occurring

$P(A|B) =$ The probability of A given B

$P(B|A) =$ The probability of B given A

$P\left(A \bigcap B\right)) =$ The probability of both A and B occu ing

Appendix 2B: Naive Bayes Classification table and Confusion matrix

JavaScript Table View

Show 10 entries Search:

	RowID	Error in %	Size of Test Set	Error Count
	fold 0	5.555555555555555	18	1
	fold 1	5.555555555555555	18	1
	fold 2	5.555555555555555	18	1
	fold 3	0	18	0
	fold 4	5.555555555555555	18	1
	fold 5	5.555555555555555	18	1
	fold 6	0	18	0
	fold 7	0	18	0
	fold 8	0	17	0
	fold 9	0	17	0

Showing 1 to 10 of 10 entries Previous 1 Next

Reset Apply Close

JavaScript Table View

Show 10 entries Search:

	RowID	1	2	3
	1	57	2	0
	2	1	68	2
	3	0	0	48

Showing 1 to 3 of 3 entries Previous 1 Next

Appendix 2C: Logistic Regression Classification table and Confusion Matrix

JavaScript Table View

Show 10 entries Search:

RowID	Error in %	Size of Test Set	Error Count
fold 0	0	18	0
fold 1	5.555555555555555	18	1
fold 2	5.555555555555555	18	1
fold 3	5.555555555555555	18	1
fold 4	0	18	0
fold 5	0	18	0
fold 6	0	18	0
fold 7	0	18	0
fold 8	0	17	0
fold 9	0	17	0

Showing 1 to 10 of 10 entries Previous 1 Next

Reset Apply Close

JavaScript Table View

Show 10 entries Search:

RowID	1	2	3
1	59	0	0
2	1	68	2
3	0	0	48

Showing 1 to 3 of 3 entries Previous 1 Next

MarkScheme - For Student's benefit:

CS3DS19 – Data Science Algorithms and Tools
Major Coursework - Assessment and Feedback Form

		comments and feedback	range for marking	Lecturer's evaluation
1.	Completeness of the submission and quality of the report: overall quality of the document (readability, completeness, presentation quality, etc.)		0-20	
2.	Task #1: description of the Clustering algorithm and the cluster validity measure		0-5	
3.	Task #1: description of the data workflow		0-5	
4.	Task #1: results (10 charts and measures)		0-10	
5.	Task #1: Conclusions and References		0-5	
6.	Task #2: description of the two Classification algorithms adopted		0-10	
7.	Task #2: description of the 10-fold cross-validation method		0-5	
8.	Task #2: experimental results (comparative performance analysis)		0-10	
9.	Task #2: Conclusions and References		0-5	
10.	Task #3: description of the data mining algorithm, the solution (data workflow) adopted and the predicted performance indices.		0-10	
11.	Task #3: prediction results (overall accuracy and three indices for the class "signal": precision, recall and F-measure). These indices will be computed by the lecturer using the submitted file "Task3-predictions.csv".		0-10	
12.	Task #3: Conclusions and References		0-5	
		Total	**0-100**	

Classification Range	Typically the work should meet these requirements
First Class (>= 70%)	Outstanding/excellent work with correct results, a good presentation of the workflows, code and results, and a critical analysis of the results. An outstanding work will present fully automated solutions based on advanced techniques.