CS3DS19 - Data Science Algorithms and Tools

Major Coursework

2021/2022

Student ID: 27020363

February 17, 2022

Task 0: Exploratory Data Analysis

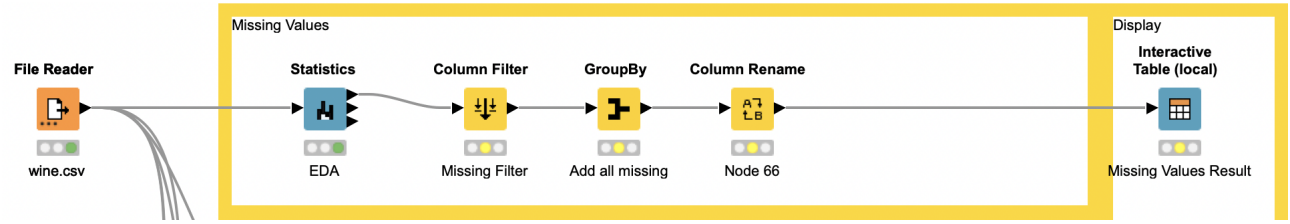
In this section, we will try to get familiarize ourselfs with the given dataset, with the ultimate goal of designing an appropriate pre-prossesing process, necessary for our future workflows. It is noteworthy that a normalisation proccess is absent from our EDA, this is because, normalisation/standardization will be explained and applied on Task2

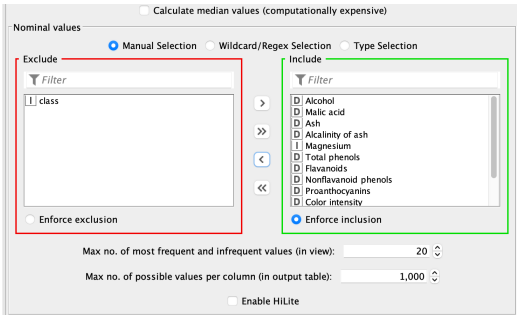
Missing values detection

The first part of EDA is the Missing values detection. We used the output of the statistics node, (output contains 'No of Missing' per variable) and we sum for all the variables, by using 'Group By' Node. The results along with the node configurations are given below. Based on the findings we can safely ignore a missing values pre-prepossessing step

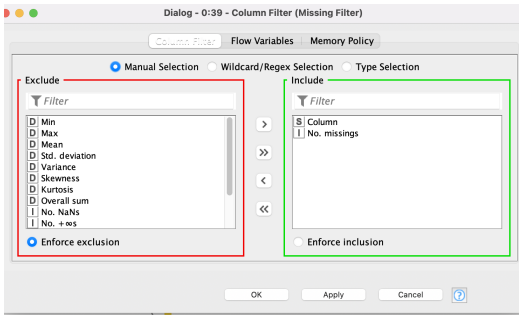
Row ID	No. missings	Variables
Row0	0	14

Workflow and node configurations for missing values detection

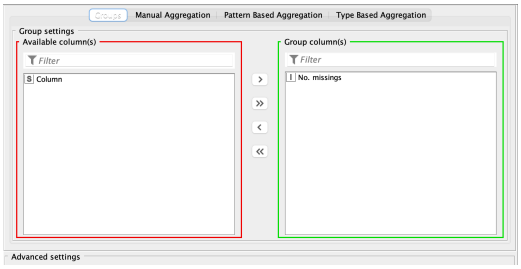




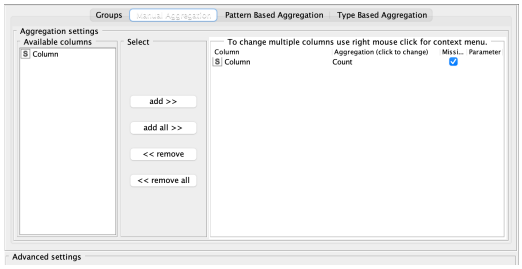
(a) Statistics, generate various statistics, Number of missing per variable included



(b) Column Filter : Keep only 'No of Missing' and Variable Name



(c) Reduce apply SUM function to 'No of Missing' per variable



(d) Reduce apply SUM function to 'No of Missing' per variable 2

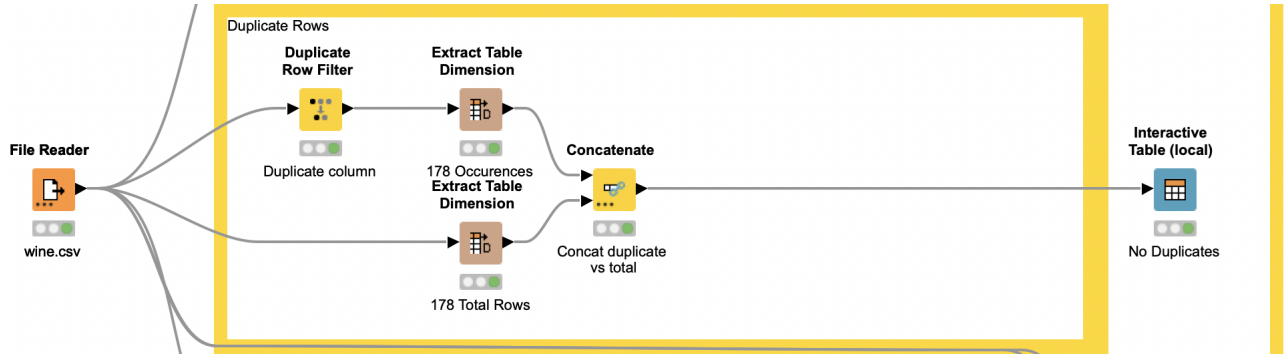
Duplicate detection

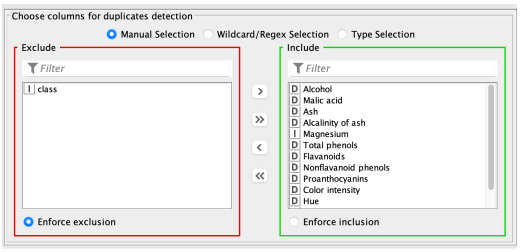
The second part of EDA is the duplicate values detection and removal. Our logic is rather simple, we applied a 'Duplicate Row Filter' into the dataset, and we counted the number of rows of this result, if the number of

rows remains unchanged, then we do not have duplicates. The results suggest that, indeed, there is no duplicate entries, so its safe to ignore a duplicate filter as a pre-processing step

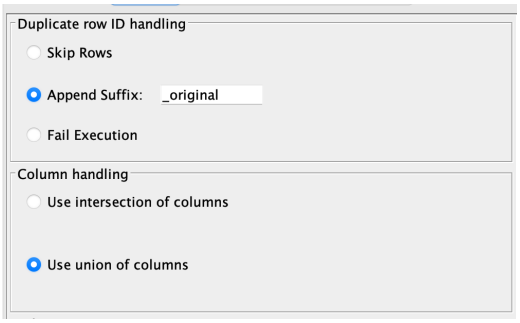
Row ID	Dimen...
Number Rows	178
Number Columns	14
Number Rows_original	178
Number Columns_original	14

Workflow and node configurations for duplicate detection





(a) Duplicate Row Filter: We exclude class for obvious reasons



(b) Concatenate: We perform union on the table dimensions before, and after the duplicates removal. If those numbers are equal, then the filter removed 0 rows, hence there is no duplicates

Outliers detection

Our third part of our EDA, contains the detection of outliers. We need to carefully thing about outliers, because one of the limitations of our chosen clustering algorithm of choice. One of K-Means limitations are the sensitivity to outliers [**?**].

How to detect outliers

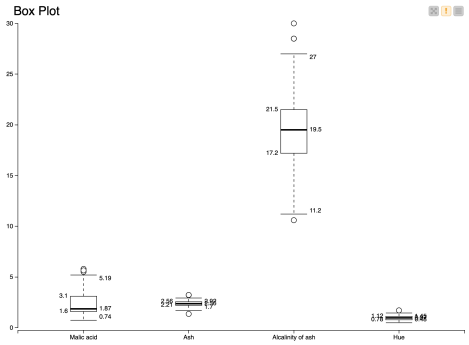
Assuming an underlying approximate normal distribution on every of our variables, we will use the famous 68-95-99 empirical rule[1] to detect outliers. Part of 68-95-99 rule states that, 99.7% of the data, lies within the range $\mu \pm 3 * \sigma$, where μ is the variable mean and σ is the standard deviation. It is rather simple to detect for outliers with the following method. We will use the output of the 'Statistics' node(columns 'max','min','mean','stddev'), and by using a simple mathematical formula, we can determine if, for a given variable, $\max > \mu + 3 \cdot \sigma$ or $\min < \mu - 3 \cdot \sigma$

Results

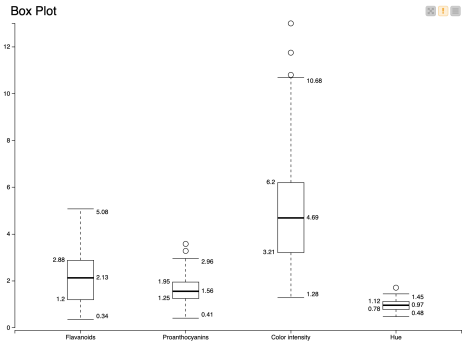
The next figure displays the variables that may contain outliers. We can see that 8 out of 14 variables contain outliers.

Row ID	max_outlier	min_outlier
Malic acid	1	0
Ash	1	1
Alcalinity of...	1	0
Magnesium	1	0
Flavanoids	1	0
Proanthocy...	1	0
Color inten...	1	0
Hue	1	0

We can visualize those suspected variables with box plots, to visually verify our findings

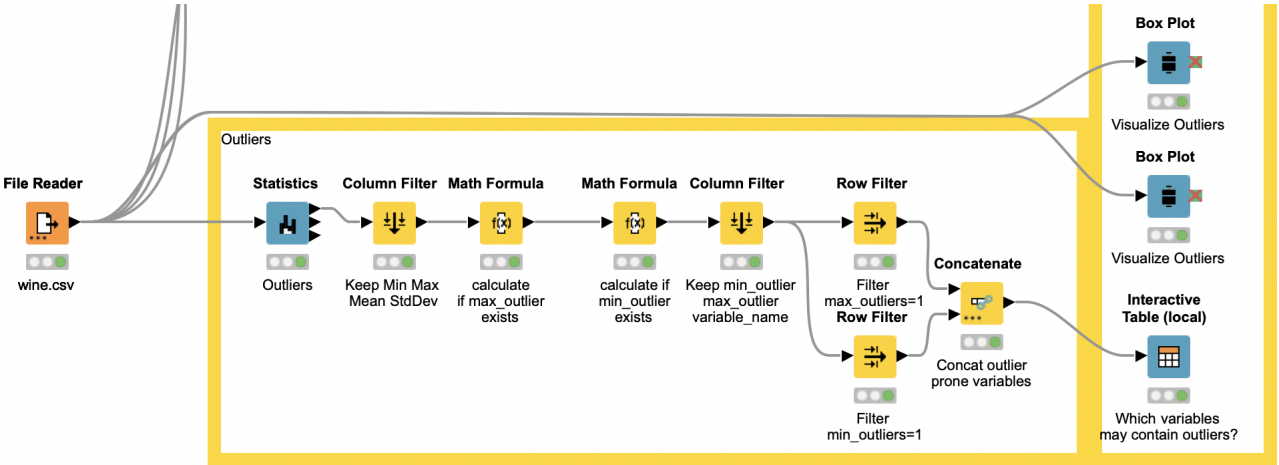


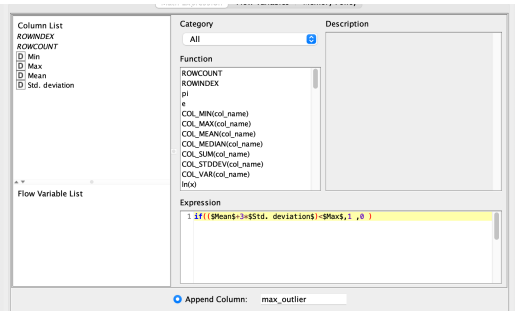
(a) Malic acid - Ash - Alkalinity of ash - Magnesium



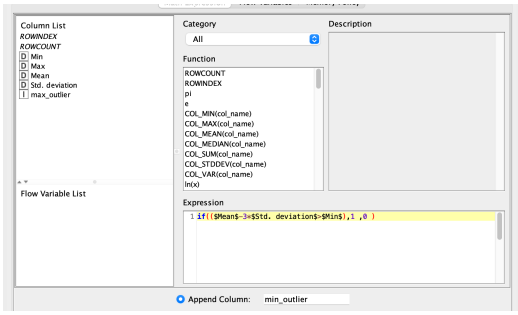
(b) Flavanoids - Proanthocyanins - Color intensity - Hue

Workflow and node configurations for outliers detection

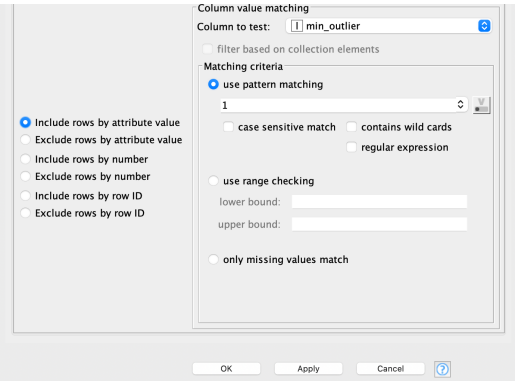




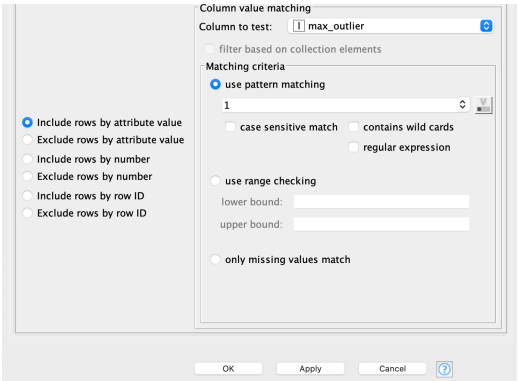
(a) We use the empirical rule, to calculate if every variable's max, exceeds $\mu + 3 \cdot \sigma$



(b) We use the empirical rule again, to calculate if every variable's min is smaller than $\mu - 3 \cdot \sigma$



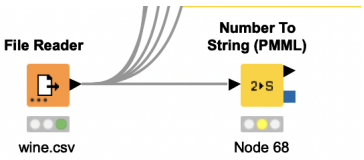
(c) Keep only the variables with min outliers(to be displayed later)



(d) Keep only the variables with max outliers(to be displayed later)

'Class' to string

The last step of our EDA, is to transform 'class' column, from an integer column, into a string column. This is essential to be handled as a categorical variable by the color manager, PCA, normalization and etc. This can be easily achieved with the following small workflow



and the node configuration is given below

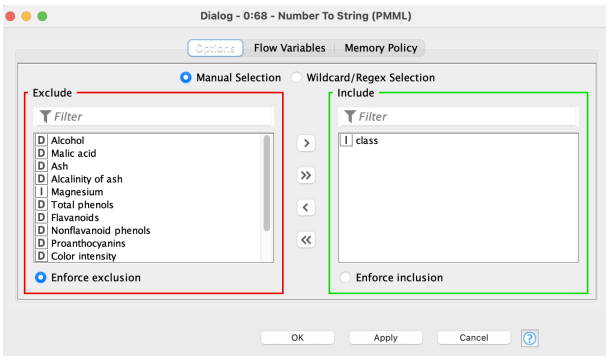
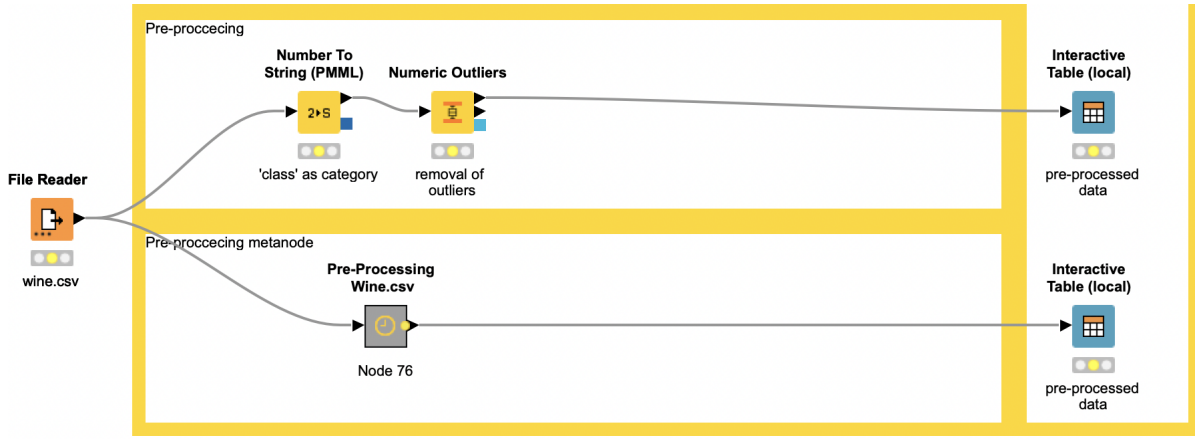


Figure 5: Number to String PPML Node: Keep and transform only 'class' column

Our Pre-processing strategy

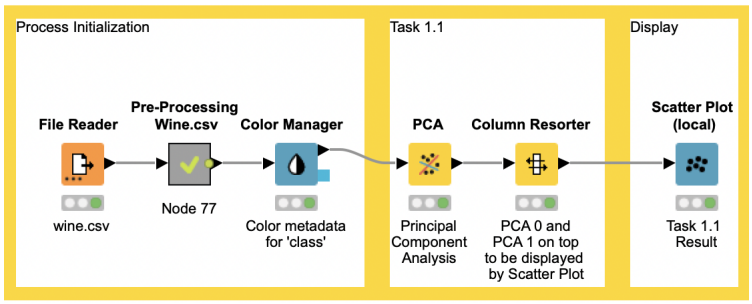
Given our EDA, we now have a clear strategy for the initial data cleansing, and processing. The following workflow will be transformed into a metanode, and will be included into every workflow from now on. A noteworthy mention, is the fact that the 'numeric outliers' strategy is to remove the outlier rows.



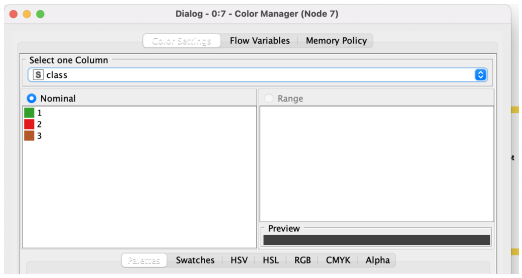
Task 1

Task 1.1 : Generate plot1

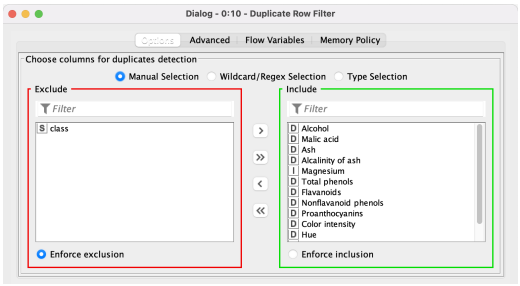
The workflow to generate plot1 is given below



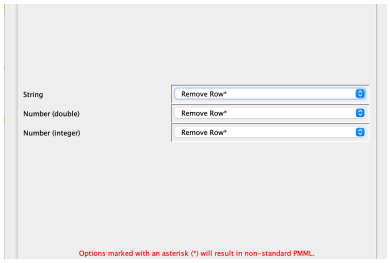
And the configurations of the nodes with descriptions are given below



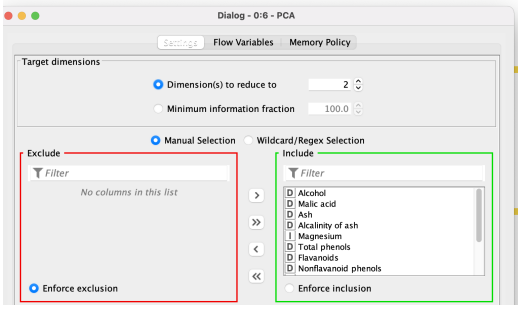
(a) Color Manager: Color Metadata for plot1 on class column



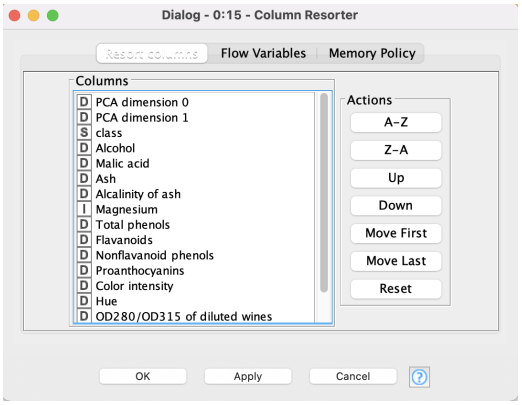
(b) Duplicate Row Filter: Standard data clearing, worth mentioning that we exclude the 'class' variable for obvious reasons



(c) Missing Value: Standard data clearing, removal of row in case some of the variables are empty(not observed)

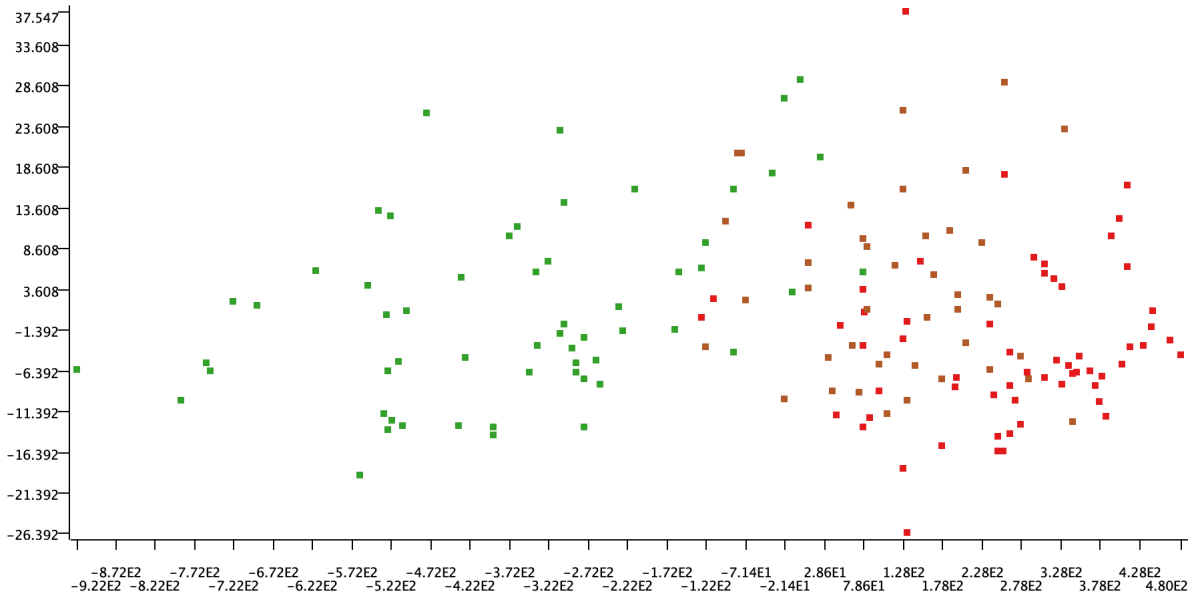


(d) PCA: The Principal Component Analysis node, we request to reduce the dimensionality to 2, worth mentioning that, as 'class' is a string column, is automatically excluded from PCA possible collumns



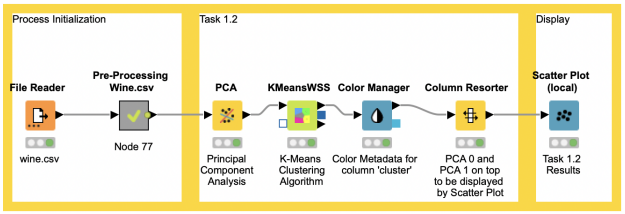
(e) Column Restorer: bring first and second principal components on top(first 2 columns), to be displayed by Scatter Plot Node

Finally, the plot1 is given below

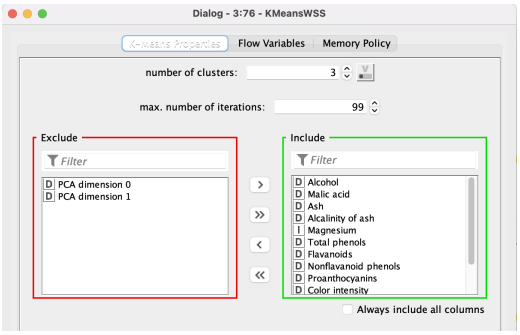


Task 1.2 : Generate plot2

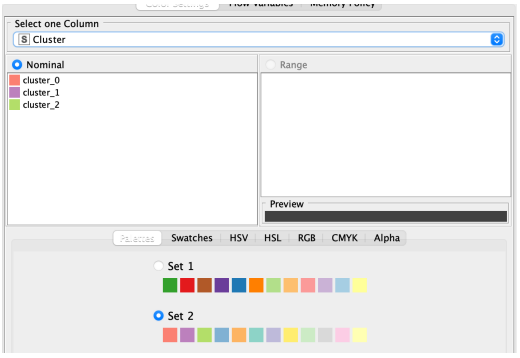
The workflow to generate plot2 is given below



Apart from the clustering algorithm adopted(K-Means), there is no significant changes with the worflow from the previous task, please see task1.1 for an overal review of the workflow. The Newrly introduced nodes configurations, as well as the descriptions are given below

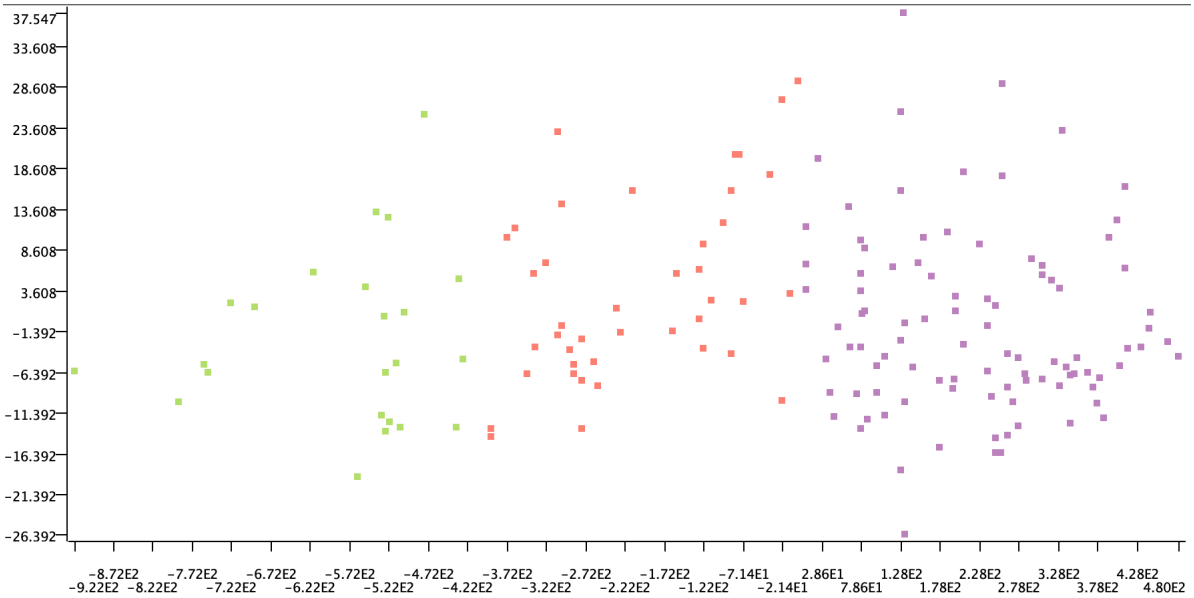


(a) KMeansWSS : this node implements our clustering algorithm of choice, the exact inner-workings to be explained on the next paragraph. A noteworthy configuration is the exclusion of PCA dimensions from the clustering process



(b) Color Manager: After KMeans finishes, we add colour metadata to our 'cluster' column, generated by the aforementioned KMeansWSS Node, to be displayed by the Scatter Plot node. The colour palette is different from the colour palette chosen on Task1.1

Finally, the plot2 is given below



Introduction to K-Means clustering algorithm

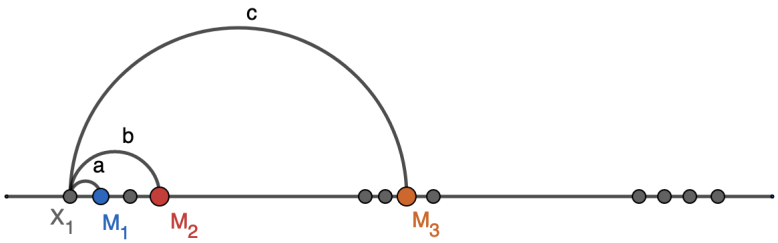
K-Means is a heuristic method for grouping similar items in a form of clusters. K is number of clusters and is pre-determined. By heuristic we mean that the algorithm does not produce absolute optimal solutions, but approximations with an given accuracy. By similarity ,on purely numeric data, we usually mean a geometric distance metric. 3 of the most widely accepted geometric metrics for determining similarity are Euclidean, Manhattan and Minkowski distances.

K-Means Algorithm

K-Means is an heuristic algorithm that repeats 2 distinct steps to converge into an acceptable solution. We will explain K-Means with a hands-on example. Let the following fictional 1D Data, $k = 3$

Let initialize $k = 3$ initial random estimated cluster centroids

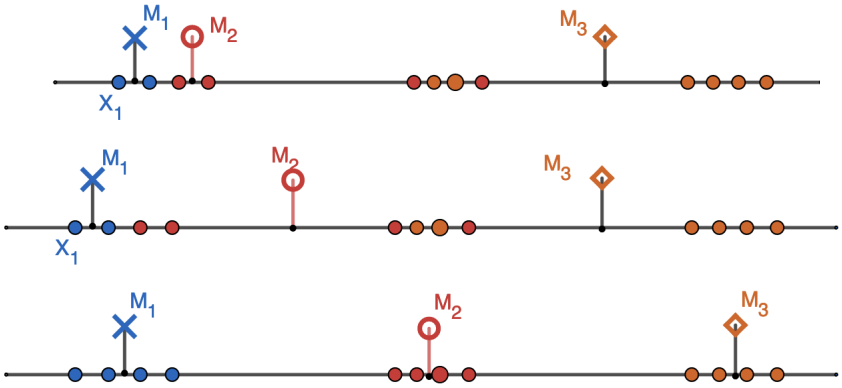
Now, the repetitive process, First, we should calculate the distances of any given point from each of the $k = 3$ estimated cluster centroids, and then assign the given datapoint into the cluster with the closest centroid. Here X_1 will be assigned on the blue cluster(M_1)



After the calculation is finished for all the datapoints, the clustering should look like the following figure

We now calculate the centroids of the clusters. the centroids do not need to be actual datapoints, they can lie on any point on the plane

We repeat steps 2,3,until we end up with an acceptable solution



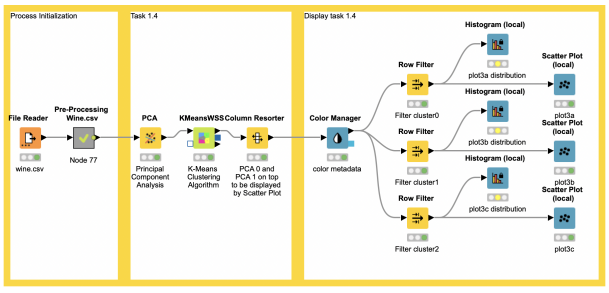
We can quantify the quality of a solution(i.e quantify if the given solution is acceptable), by using Sum of Squared errors statistic(SSE). K-Means terminates when SSE converges into a value. SEE is explained in detail on Task 1.5

Task 1.3 : Compare plot1 and plot2

The main point of interest in the figures above, is the fact that, the clustering algorithm(K-Means, k=3) failed to appropriately partition the data, and recognise the classes provided by the dataset, This phenomenon can be explained due to the absence of the standardization process, something that will be explained in the next Task. The exact scale of the problem cannot be appropriately assessed with only those two plots though, we will need to examine the distribution of the classes in each cluster, something that will be done on the next task(Task 1.4)

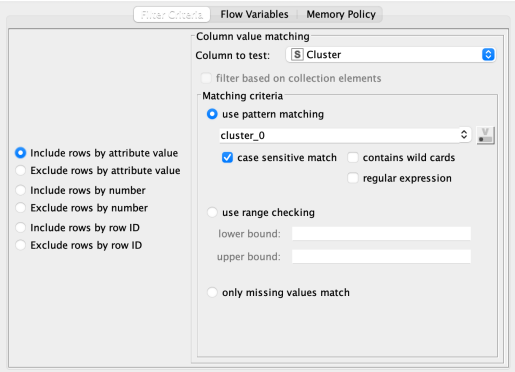
Task 1.4 : plot3a,plot3b,plot3c

The workflow to generate the requested plots is given below

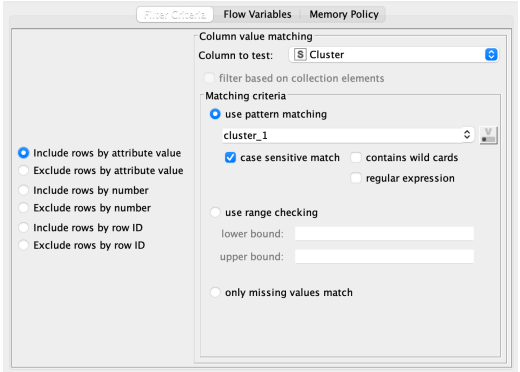


The majority of the workflow is identical to workflows of task1.1 and task1.2, so please refer to those tasks for a detailed explanations of those nodes. The major difference is on the display section. There, we use 3 row filters(one per cluster) to filter the datapoints that were assigned on each cluster. Hence in the first plot, we

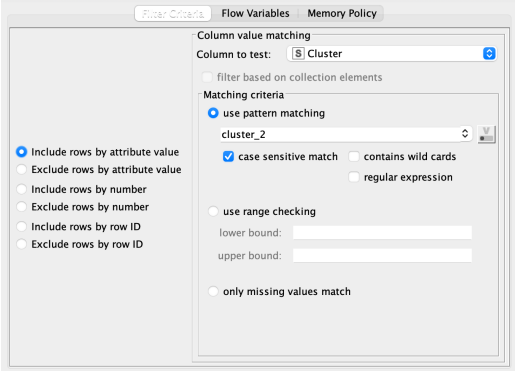
have all the datapoints assigned to cluster0 and vice versa. The node configurations with the nessesary filters are given below



(a) Row filter: cluster0 filter

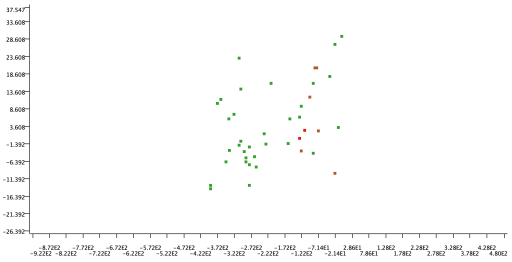


(b) Row filter: cluster1 filter

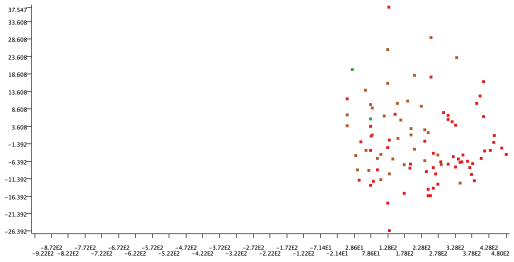


(c) Row filter: cluster2 filter

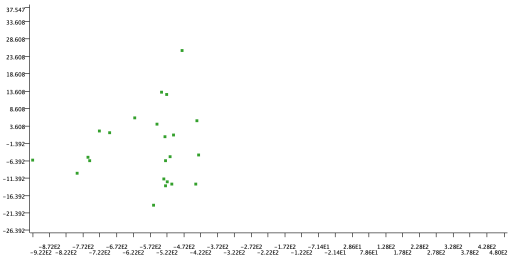
Finally, the generated plots are given in the next figure.



(a) plot3a

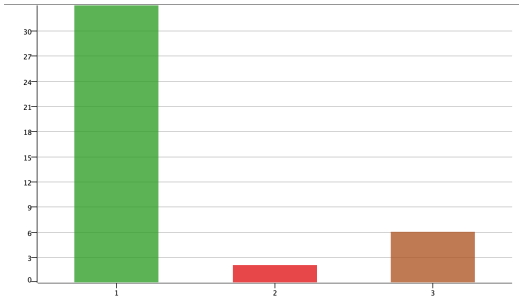


(b) plot3b

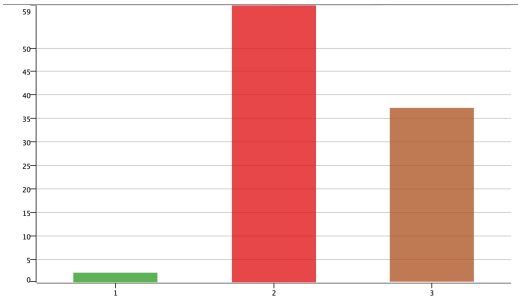


(c) plot3c

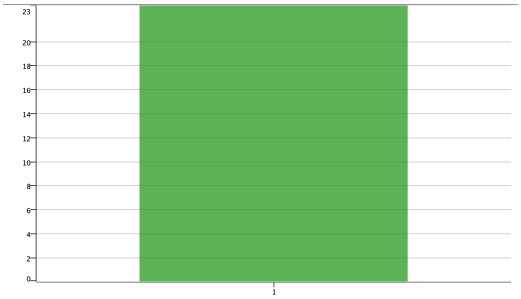
It becomes apparent that the clusters generated are composing an meaningless clustering solution. Under ideal circumstances, each cluster will had a vast majority of each of the classes, with some to none variation due to outliers or errors, Using histograms we can visualize the extend of the issue



(a) plot3a distribution



(b) plot3b distribution

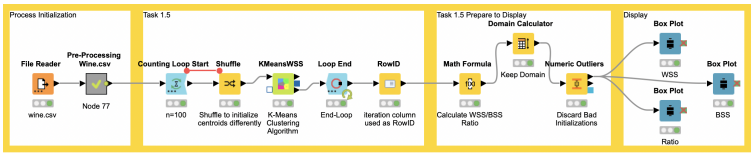


(c) plot3c distribution

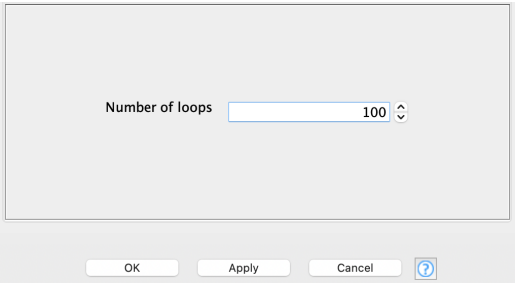
This is the result of the lack of a normalization process, something that we will explain on the next task.

Task 1.5 : Cluster Validity Measure

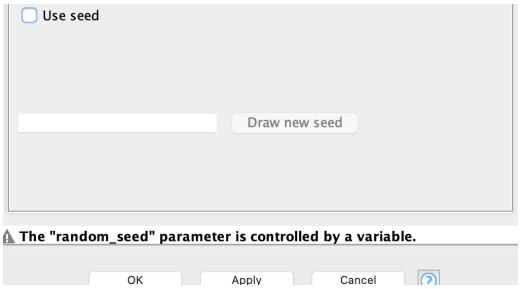
For our cluster validity measure, we will explain and intepret WSS (Within cluster sum of squares) and BSS (Between cluster sum of squares). The workflow for generating the cluster validity measures is given below



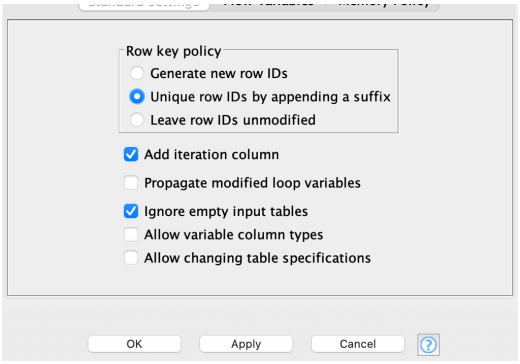
The majority of the workflow is identical to workflows of task1.1 to task1.4, so please refer to those tasks for a detailed explanations of those nodes. The Major Difference is on the repeat process and on the intepretation proccess. The newrly introduced nodes configurations are given below, along with a short explanation



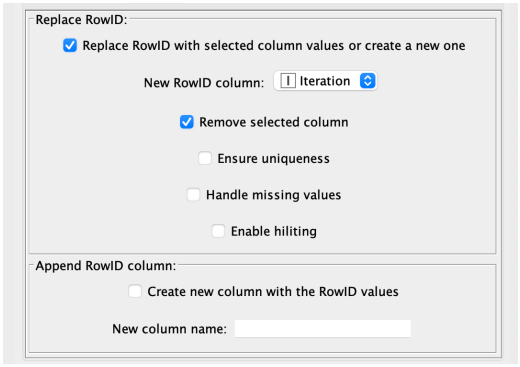
(a) Counting Loop Start: Because of K-Means is an heuristic algorithm, it makes sense to be analysed in a series of runs, we will run for $n = 100$. Thats necessary because of the fact that every heuristic algorithm is vulnerable to be stuck onto a local-minima and produce biased/wrong results [**?**].



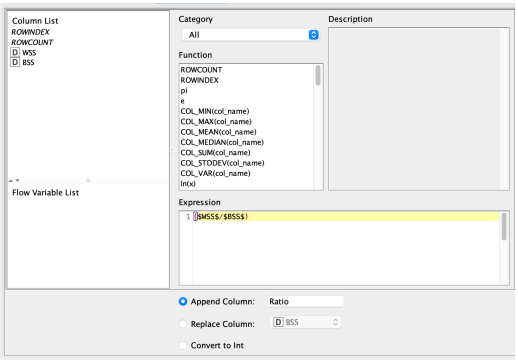
(b) Shuffle: This K-Means implementation, gets initialized by setting the first k elements as centroids. In order to evaluate the behaviour of the algorithm, we need to shuffle our dataset to enforce K-Means to initialized with different centroids [**?**]



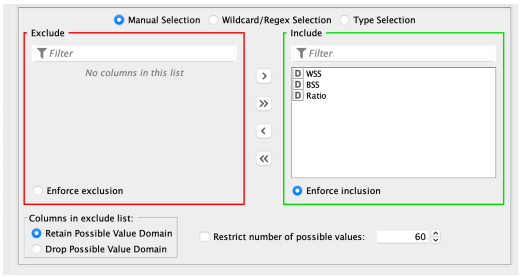
(c) Loop End: The only noteworthy mention here, is the fact that we append a new column, containing the number of iteration



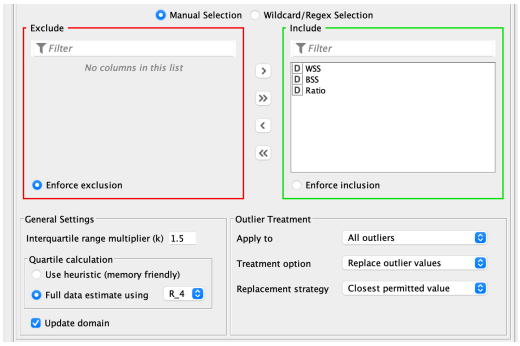
(d) Row ID: Use the number of iterations as RowID column



(e) Math Formula: A nice way to visualize the performance of K-Means, is to calculate the ratio of WSS/BSS, more on that on the 'interpretation' part



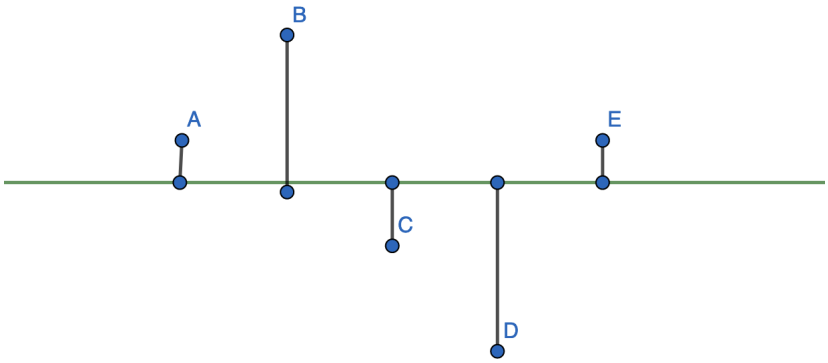
(f) Domain Calculator: As we involve very small numbers, we need to explicitly tell KNIME to keep the ranges (not to round) as much as possible.



(g) Numeric Outliers: In order to extract insightful box plots, we need to eliminate outliers(very bad initializations)

Prerequisites for explaining WSS ans BSS

Before our attempt of explaining WSS and BSS, it is essential to explain SSE (Sum of Square estimate of Errors). SSE is an evaluation metric with a plethora of applications in statistics and predictive analytics[**?**]. It is used to evaluate a model's performance against a training set[**?**]. The logic behind SSE is rather simple in nature. Let X an random variable and a model $y = \bar{X}$



We can evaluate our model's performance, by calculating the Sum of Errors, where 'error' in this case, is the distance of the observed variable from the responce of our model (i.e the mean $\bar{X}$), as follows

$$SE = \sum_{i=1}^{n} X_i - \bar{X}$$

Unfortunately, our evaluation metric has a fatal flaw, it allows for error's to 'cancel out' simply because they are placed beneath our model's line. It can be proven that, for the case of $y = \bar{X}$ SE is always 0.

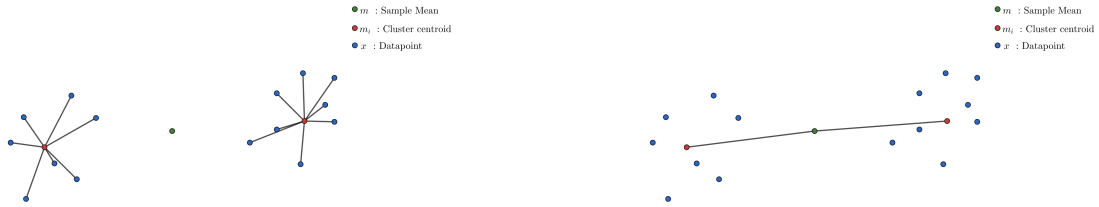
$$SE = \sum_{i=1}^{n} X_i - \bar{X}$$
$$SE = \sum_{i=1}^{n} X_i - \sum_{i=1} n\bar{X}$$
$$SE = \sum_{i=1}^{n} X_i - n\bar{X}$$
$$SE = \sum_{i=1}^{n} X_i - \sum_{i=1}^{n} X_i$$
$$SE = 0$$

For other models(in higher dimensions, where corellation of the involved variables is not 1), it may not be zero, but the fatal flaw remains, by cancelling errors we severely underestimate the errors involved. A more appropriate approach could be to square the distances, that would give us the Sum of Squared Errors (SSE)

$$SE = \sum_{i=1}^{n} (X_i - \bar{X})^2$$

WSS and BSS Inner-Workings

In the world of Clustering models performance evaluation, SSE is translated into two distinct but related ideas[**?**], WSS and BSS



(a) WSS

(b) BSS

WSS

WSS or Within Cluster Sum of Squares is a clustering model performance evaluation function. Our 'model' in this case is the cluster centroid and the error is the distance for each datapoint from the cluster centroid. WSS can be evaluated with the following formula[**?**]

$$WSS = \sum_{i=1}^{k} \sum_{x \in C_i} (x - m_i)^2$$

Where k, the number of clusters involved, C_i the individual cluster and m_i, the given cluster centroid.

BSS

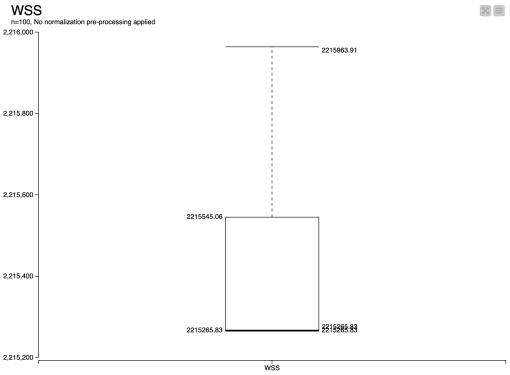
BSS or Between Cluster Sum of Squares is a clustering model performance evaluation function, just like WSS. Our 'model' in this case is the overall data mean and the error is the distance of each cluster's centroid to the overall mean. BSS can be evaluated with the following formula[**?**]

$$BSS = \sum_{i=1}^{k} |C_i|(m - m_i)^2$$

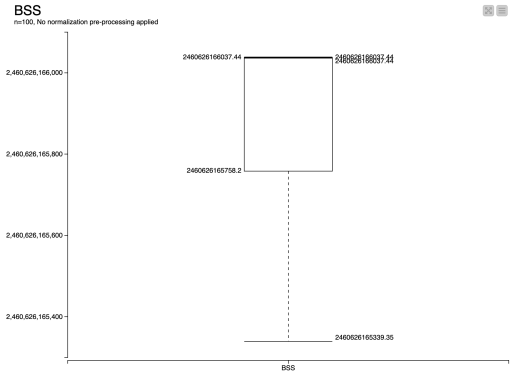
Where k, the number of clusters involved, C_i the individual cluster and m_i, the given cluster centroid.

Interpretation

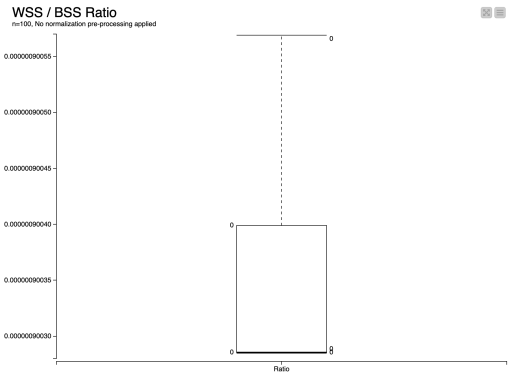
A good clustering solution, would involve well separated clusters, with low WSS to BSS ratio[**?**]. As K-Means is a heuristic based algorithm, we will not analyse WSS and BSS of a single K-Means run, but rather we will analyse their behaviour after a sequence of 100 runs.



(a) WSS



(b) BSS



(c) Ratio

It is rather obvious, that the WSS/BSS Ratio is quite small, something that indicates well separated clusters(i.e a good clustering solution). However, a single cluster validity measure on its own, is insufficient to provide enough evidence for a meaningful clustering. Taking into consideration our observations from Task 1.3 and Task 1.4, this clustering [**?**]is not meaningful, as it produces wrong clusters. This is something that can be explained due to absence of Normalisation/Standardization process, something that we will analyse on Task 2

Task 2

The importance of Normalization

Let the following figure with some of the statistics for Proline and Ash variables

Row ID	Max	Mean	Std. deviation	Variance
Ash	3.23	2.367	0.274	0.075
Proline	1,680	746.893	314.907	99,166.717

K-Means is a geometric algorithm, that uses a distance(proximity) metric to evaluate the simmilarity between datapoints. Having one or more variables with significantly greater range of values, will cause this variable to have greater influence on the clustering, producing biased results.

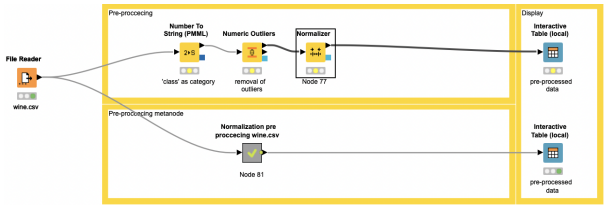
How Normalization solves the problem?

Normalization solves the aforementioned issue by transforming all the variables in question, within the range [0-1], without affecting the scales of the distances. Normalization is described with the following formula

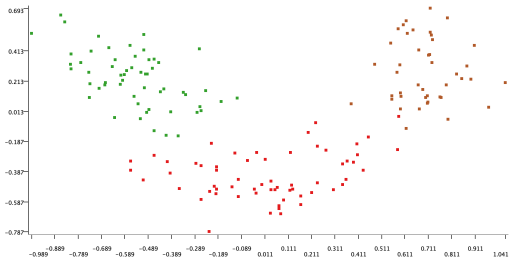
$$Norm(x) = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Changes in the workflows

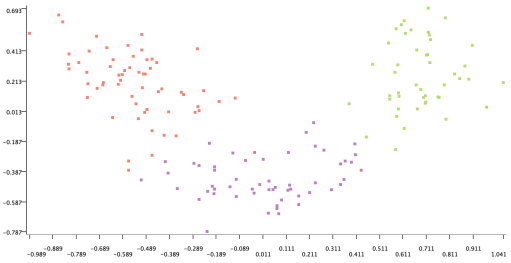
The only change to apply normalization, is on our pre-processing node. Everything else will be the same. Keep note that the 'normalization' node follows the 'min-max' normalization, with min=0 and max=1. The 'class' variable is automatically excluded.



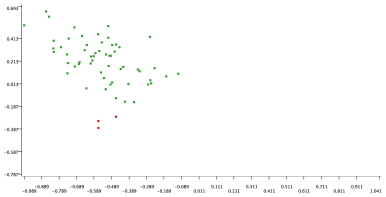
New Plots



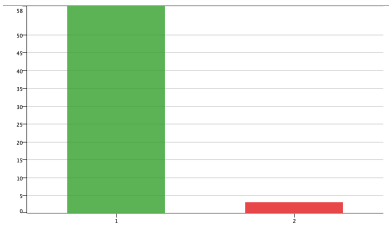
(a) normalized plot1



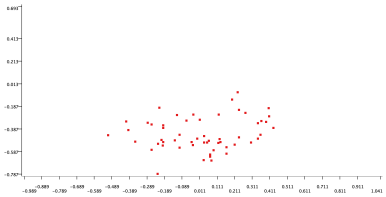
(b) normalized plot2



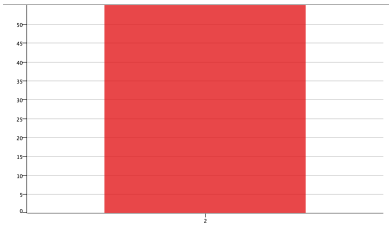
(c) normalized plot3a



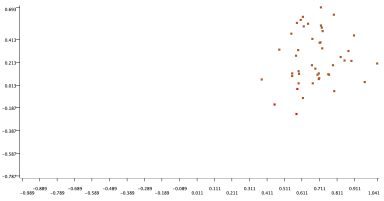
(d) normalized plot3a distribution



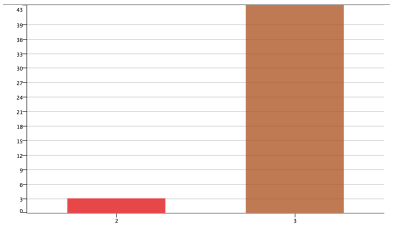
(e) normalized plot3b



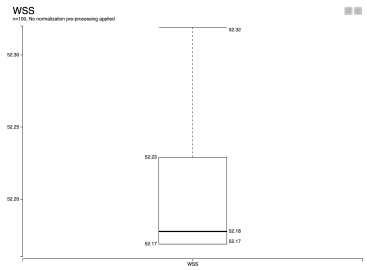
(f) normalized plot3b distribution



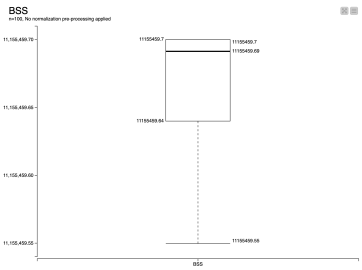
(g) normalized plot3c



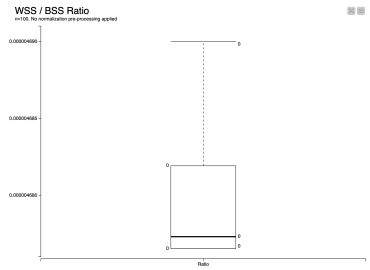
(h) normalized plot3c distribution



(i) WSS



(j) BSS



(k) Ratio

Compare and Contrast

As we can clearly see from the newrly generated plots, K-Means is now able to recognise the provided classes with little to no error. The classes per cluster distributions are now dominated from one class, meaning that every cluster contains the majority of one class, plus some errors(something expected, as our model is not perfect).Our clustering validity measure indicates again, a well separated solution, with a very low WSS and very high BSS. Our cluster validity measure, along with the new plots3a-c indicates a well separated, and meaningful clustering.

References

[1] *Grafarend, E. (2006). Linear and nonlinear models (p. 553). Berlin, New York, N.Y.: Walter de Gruyter*

[2] *Shannon, C., 1948. A Mathematical Theory of Communication. Bell System Technical Journal, 27(3), pp.379-423.*

[3] *Shanker M, Hu MY, Hung MS, Effect of Data Standardization on Neural Network Training* https://www.sciencedirect.com/science/article/pii/0305048396000102