Student number: 25015646
Student username: bx015646

<u>Task #1</u>

To start off, I will explain what I did in the KNIME workflow. I used a File Reader node to read in the wines.csv dataset.

I then used Color Manager to apply a color to the records according to the “class” attribute. I used the PCA node to perform PCA on the data. I used Column Resorter to move the important columns at the top since the Scatter Plot node plots the first two columns unless the user specifies otherwise. I used the Scatter Plot node to plot the two PCA dimensions.

I used the k-Means node to apply clustering to the data as required. I colored the data according to their cluster ID this time as also required, using Color Manager of course. After that, I applied the steps above: PCA, resort the columns and plot them on a scatter plot.

To visually verify the distribution of class labels in each cluster, I first used Color Manager on the “class” attribute again with the same colours as the first time. I have then used the Row Filter node 3 times to filter records by their cluster ID, each time specifying the different cluster number in the matching criteria. Then, again for presentation, I used Column Resorter and Scatter Plot.

I’ve used the Entropy Scorer node, to get clustering statistics and validity measures, which I will go into more detail below.

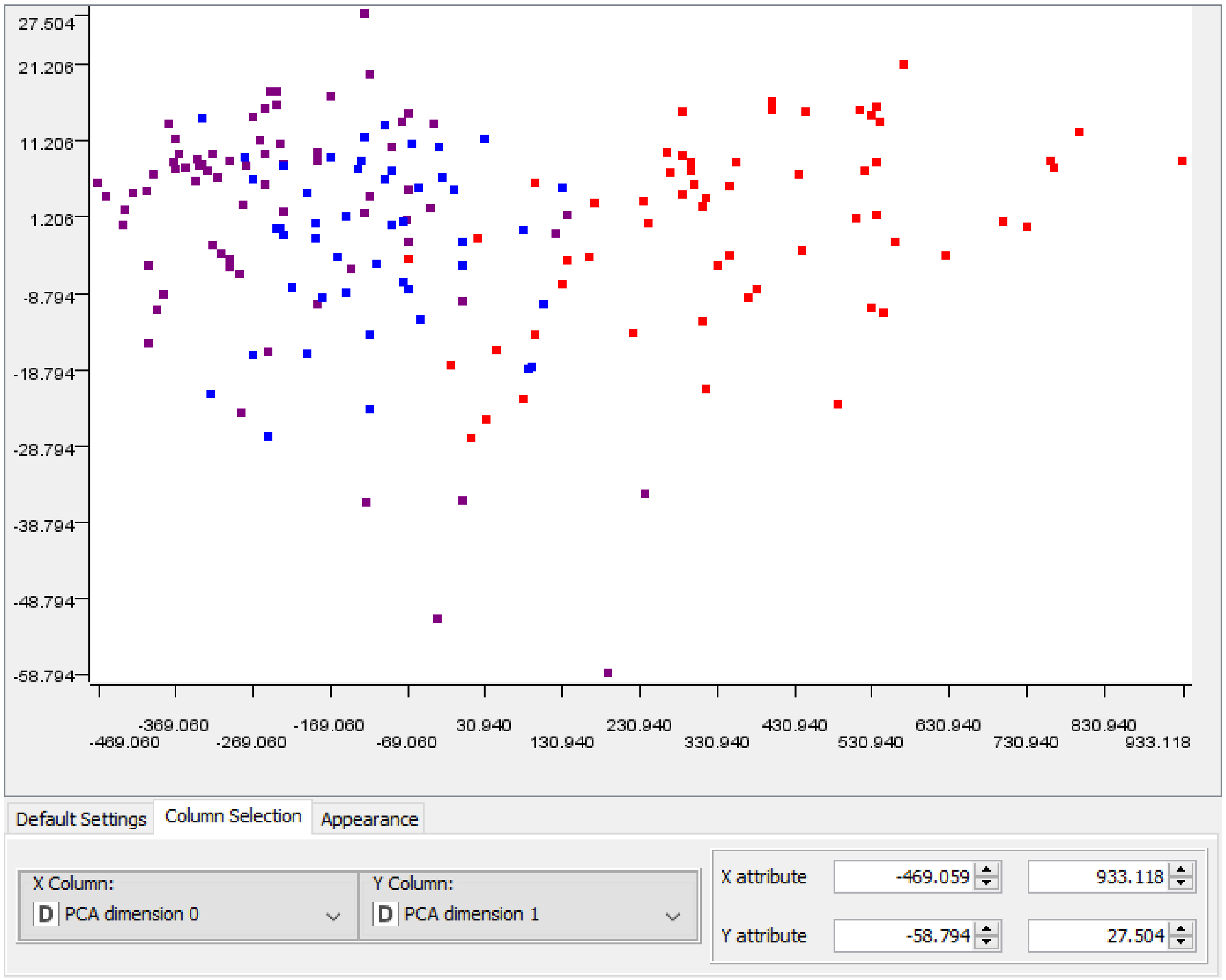
To normalize the dataset, I’ve used the Normalizer node, after reading the dataset with File Reader, which normalizes the values of all columns in the interval [0,1]. Afterwards, I’ve repeated all the aforementioned steps to the normalized data.

<u>Task 1.1</u>

Principal Component Analysis (PCA) is a dimension-reduction tool that can be used to reduce a large set of variables to a small set that still contains most of the information in the large set. PCA is a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components.

I have reduced the number of dimensions down to two so that I could generate two-dimensional coordinates and a 2D scatter plot of the records.

Student number: 25015646
Student username: bx015646



X Column: D PCA dimension 0

Y Column: D PCA dimension 1

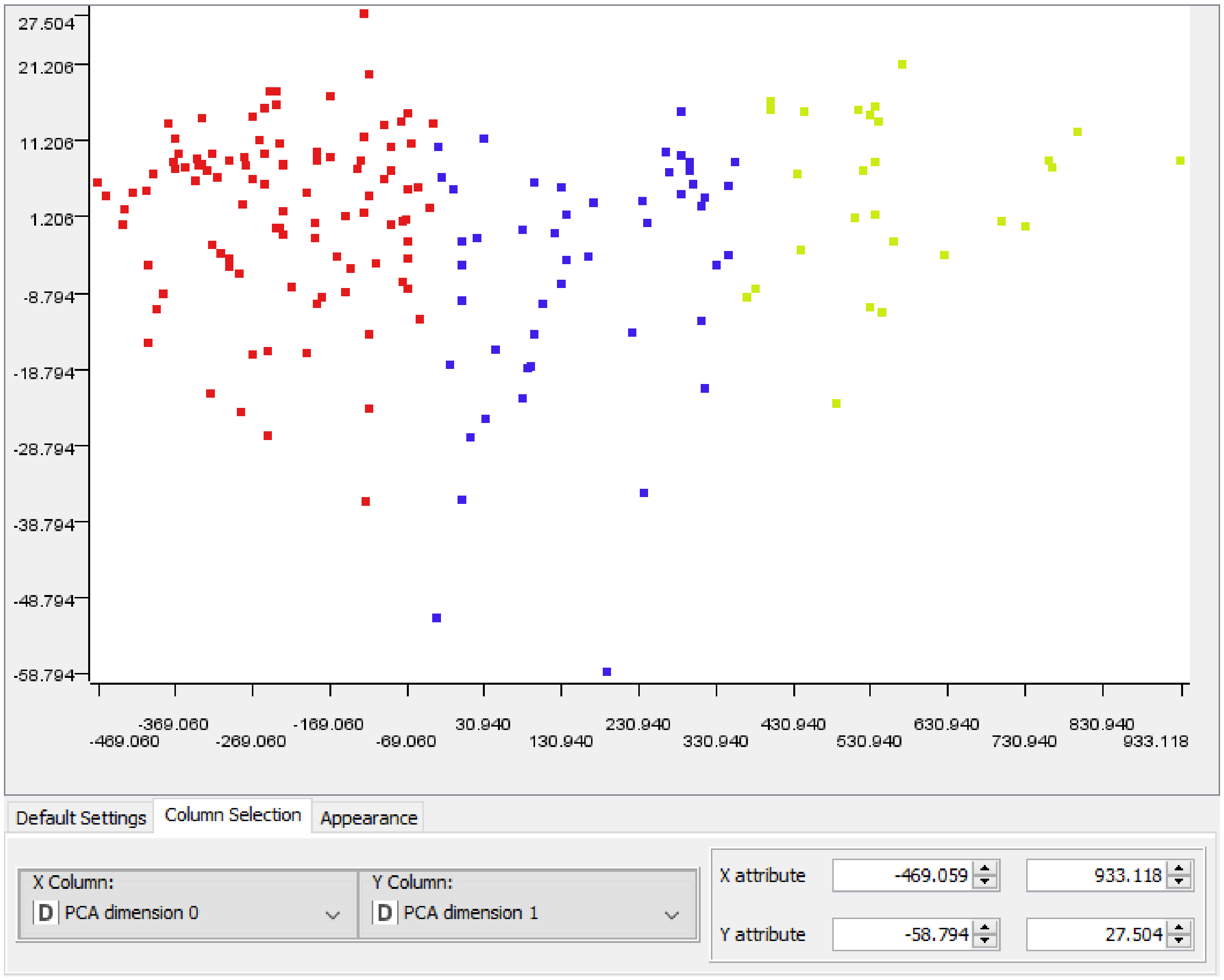
X attribute -469.059 933.118

Y attribute -58.794 27.504

The above is a scatter plot of the two PCA dimensions on x and y axis. As we can see, two of the classes are not distinctly separate from each other when applying PCA unlike the red class which is easier to spot.

Clustering

K-Means clustering is an unsupervised learning algorithm that finds a fixed number (k) of clusters in a set of data. A cluster is a group of data points that are grouped together due to similarities in their features. When using a K-Means algorithm, a cluster is defined by a centroid, which is a point (either imaginary or real) at the center of a cluster. Every point in a data set is part of the cluster whose centroid is most closely located. K-Means finds k number of centroids, and then assigns all data points to the closest cluster, with the aim of keeping the centroids small. K-means performs a crisp clustering that assigns a data vector to exactly one cluster. The algorithm terminates when the cluster assignments do not change anymore. The clustering algorithm uses the Euclidean distance on the selected attributes.



X Column: D PCA dimension 0

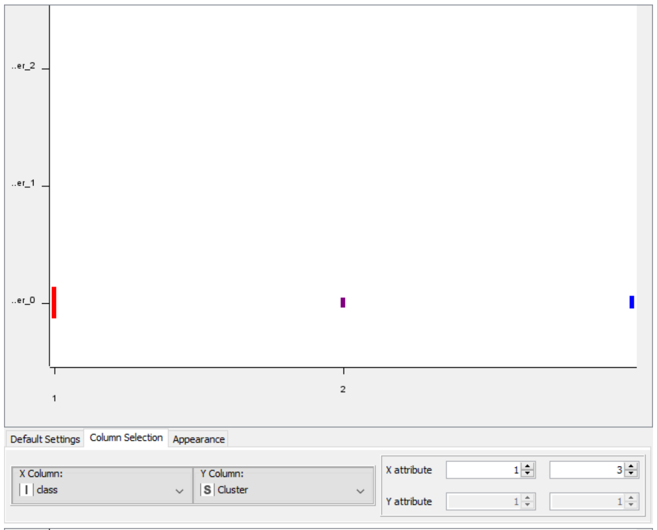
Y Column: D PCA dimension 1

X attribute -469.059 933.118

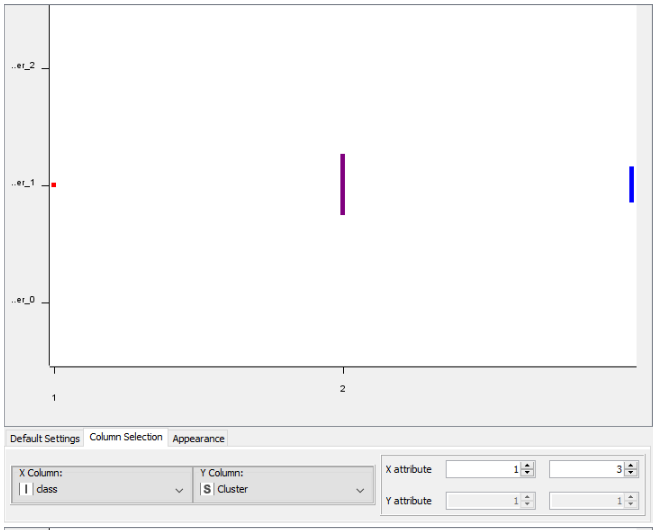
Y attribute -58.794 27.504

The above is a scatter plot of the two PCA dimensions, after the k-means algorithm is performed on the dataset. The plot shows that PCA is more efficient when k-means is applied to the dataset.

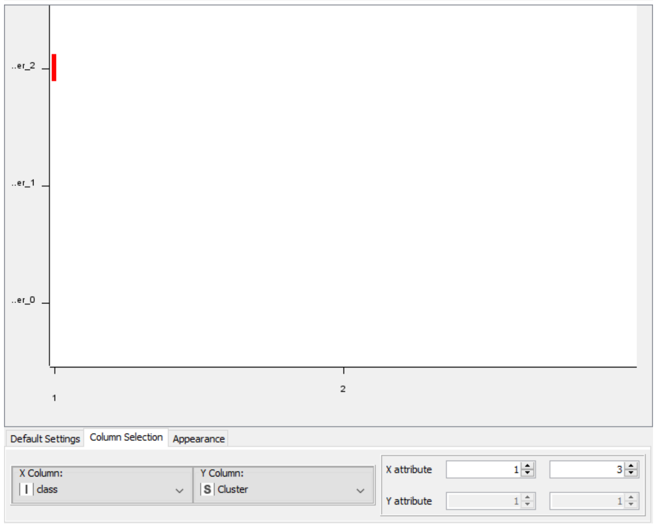
Student number: 25015646
Student username: bx015646



X Column: I | class
Y Column: S | Cluster
X attribute 1 3
Y attribute 1 1

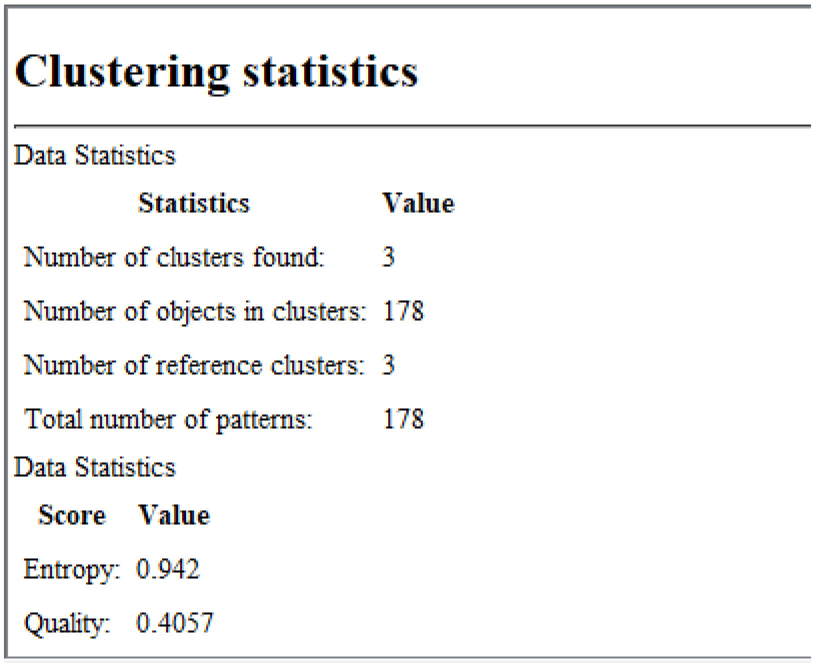


X Column: I | class
Y Column: S | Cluster
X attribute 1 3
Y attribute 1 1



X Column: I | class
Y Column: S | Cluster
X attribute 1 3
Y attribute 1 1

For the records associated to each cluster I generated a 2D plot with the colour associated to the class label so that I can visually verify the distribution of class labels in each cluster, with the x-axis being the class and the cluster on the y-axis. Only cluster 2 is assigned to a specific class, while the other clusters are intermixed with different classes.



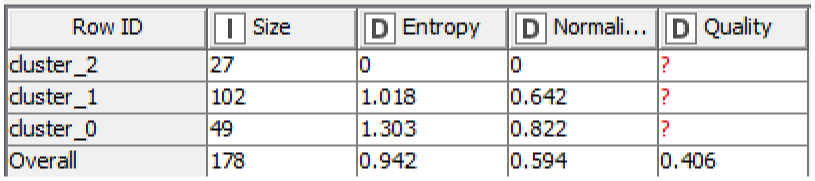
Clustering statistics

Data Statistics

Statistics	Value
Number of clusters found:	3
Number of objects in clusters:	178
Number of reference clusters:	3
Total number of patterns:	178

Data Statistics

Score	Value
Entropy:	0.942
Quality:	0.4057



Row ID	I Size	D Entropy	D Normali...	D Quality
cluster_2	27	0	0	?
cluster_1	102	1.018	0.642	?
cluster_0	49	1.303	0.822	?
Overall	178	0.942	0.594	0.406

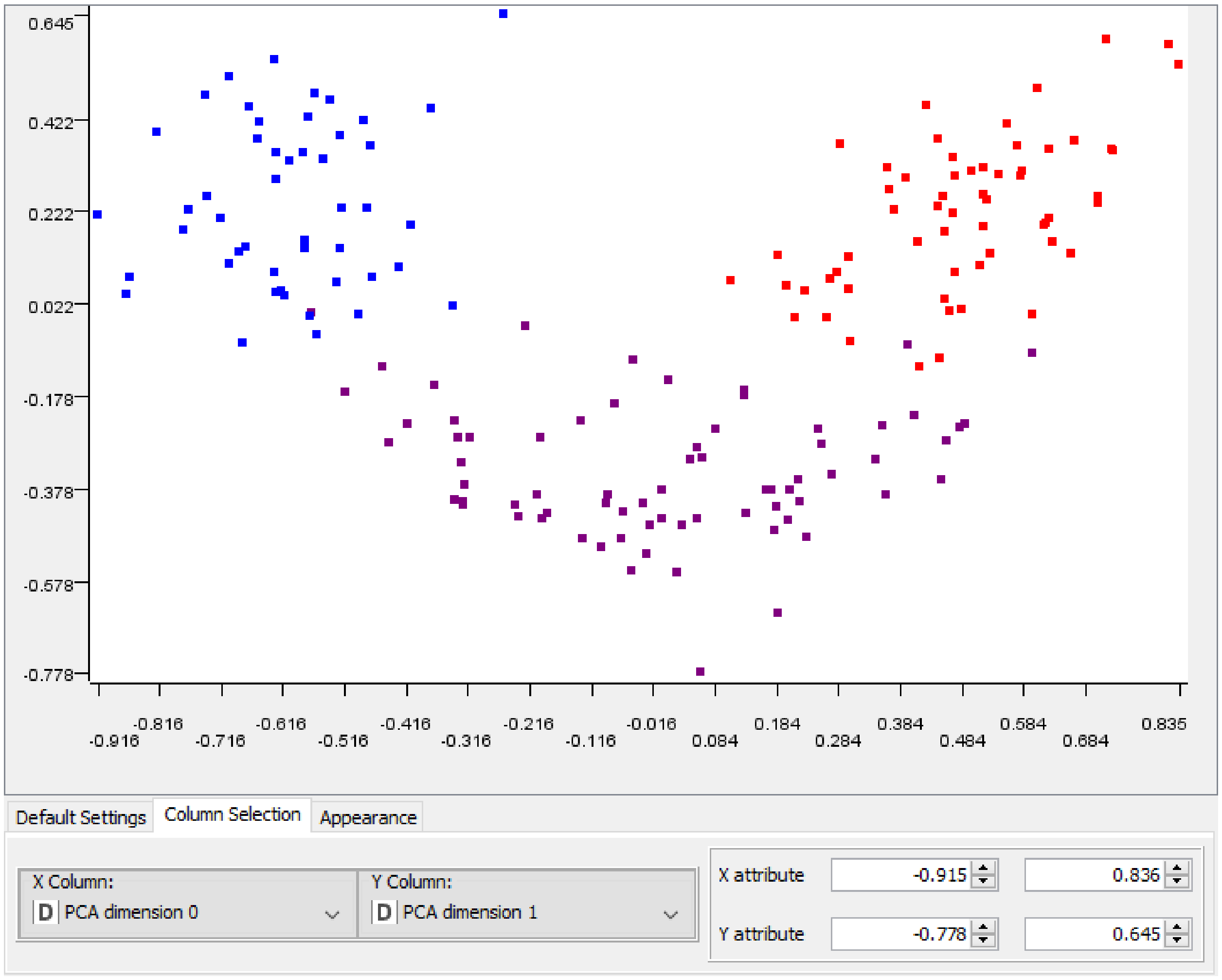
I've used an entropy scorer node which is a validity measure for clustering results given a reference clustering. It contains the following statistics, aside from the simple statistics:

- Entropy: The accumulated entropy of all identified clusters, weighted by the relative cluster size. The entropy is not normalized and may be greater than 1.
- Normalized Entropy: Entropy normalized in the domain of [0,1].
- Quality: The quality value according to the formula referenced above. It is the sum of the weighted qualities of the individual clusters, whereby the quality of a single cluster is calculated as (1 – normalized entropy). The domain of the quality value is [0,1].

It is clear that the results are quite bad, as the entropy is very high, and the quality is quite low. In addition to that, more than 50% of the data belongs to cluster 1, making the clustering unevenly distributed.

Task 1.2

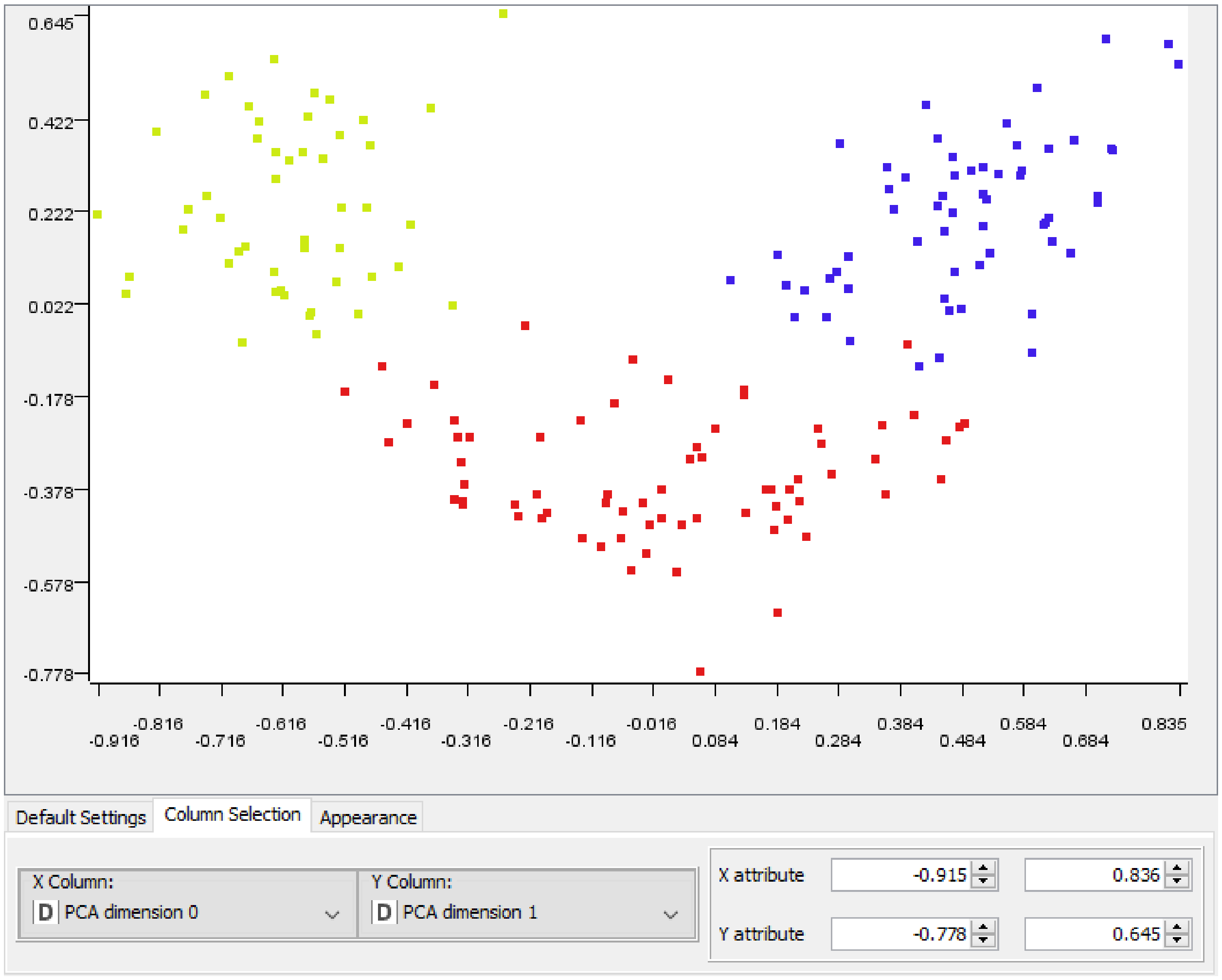
Student number: 25015646
Student username: bx015646



X Column: D PCA dimension 0
Y Column: D PCA dimension 1
X attribute -0.915 0.836
Y attribute -0.778 0.645

The above is a scatter plot of the two PCA dimensions on x and y axis that were created after performing PCA on the normalized dataset. The three class colours are more distinct with normalizing than without

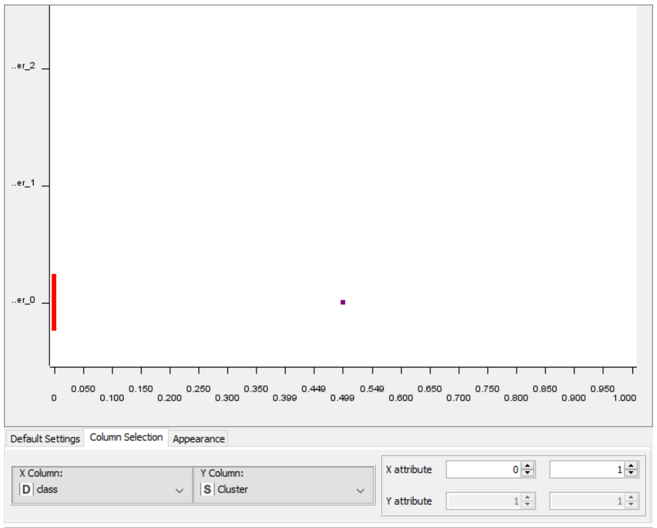
Student number: 25015646
Student username: bx015646



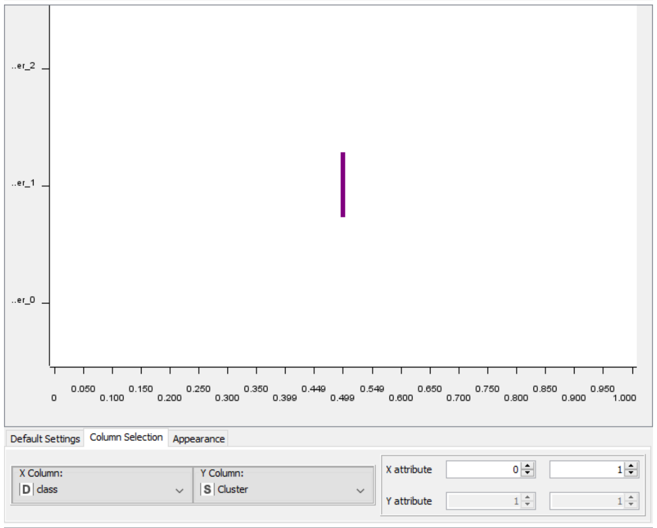
X Column: D PCA dimension 0
Y Column: D PCA dimension 1
X attribute -0.915 0.836
Y attribute -0.778 0.645

The above is a scatter plot of the two PCA dimensions, after the k-means algorithm is performed on the normalized dataset. The plot above (PCA on the normalized data) already split the classes well. This one is slightly improved, in terms of clarity, when PCA is applied to the k-means normalized dataset.

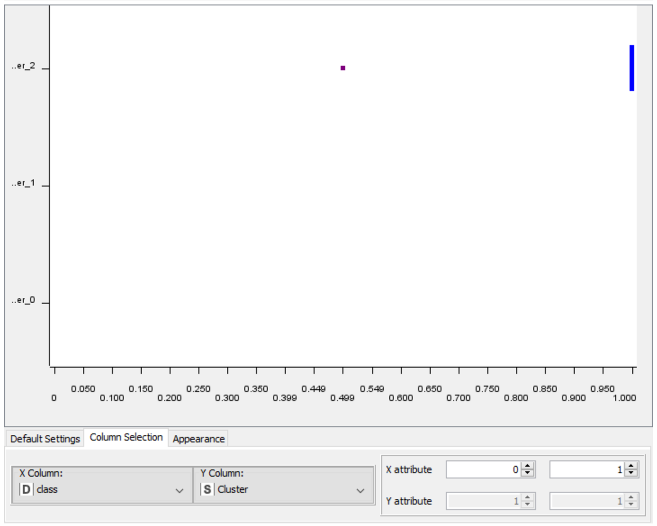
Student number: 25015646
Student username: bx015646



X Column: D | class
Y Column: S | Cluster
X attribute 0 1
Y attribute 1 1

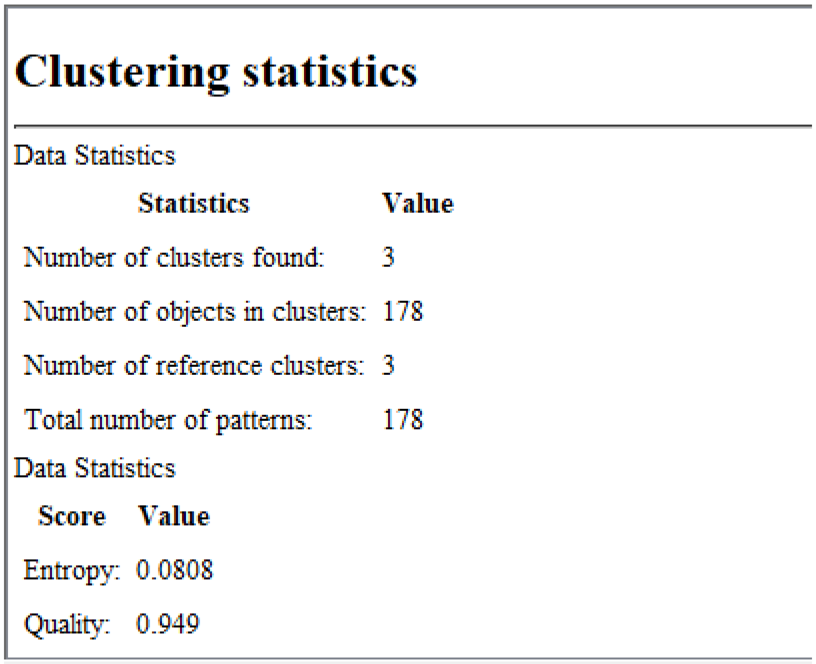


X Column: D | class
Y Column: S | Cluster
X attribute 0 1
Y attribute 1 1



X Column: D | class
Y Column: S | Cluster
X attribute 0 1
Y attribute 1 1

For the records associated to each cluster I generated a 2D plot with the colour associated to the class label so that I can visually verify the distribution of class labels in each cluster, with the x-axis being the class and the cluster on the y-axis. Cluster 0 and 2 are almost perfect while cluster 1 is completely perfect.



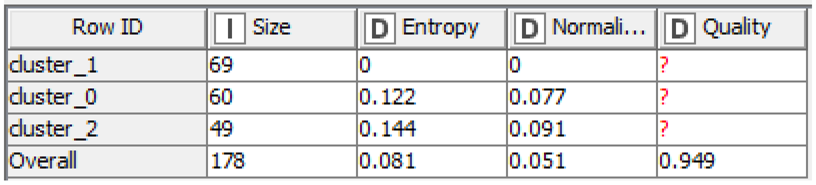
Clustering statistics

Data Statistics

Statistics	Value
Number of clusters found:	3
Number of objects in clusters:	178
Number of reference clusters:	3
Total number of patterns:	178

Data Statistics

Score	Value
Entropy:	0.0808
Quality:	0.949



Row ID	I Size	D Entropy	D Normali...	D Quality
cluster_1	69	0	0	?
cluster_0	60	0.122	0.077	?
cluster_2	49	0.144	0.091	?
Overall	178	0.081	0.051	0.949

The results are significantly better when the data is normalized. The entropy is very low, almost 0, and the quality is almost perfect at 0.95. Also, the data is more equally split among the clusters in comparison to the non-normalized dataset.

Task #2

I used a File Reader node to read in the wines.csv file. After that, I used the Number to String node to transform the class column to string from integer and make KNIME understand that this is a nominal column. The purpose of doing that we need a nominal attribute to use as the real class when doing classification. I've used the X-Partitioner node to split the data in the training and test set depending on the number of validations, 10 in this case. Then in each of the two models, there's a learner node that creates a model that will then be able to be connected to a predictor node to validate the data from the test set we just created. Then the X-Aggregator collects the results from the predictor node, compares predicted class and real class and outputs the predictions for all rows. Finally, I've used a Scorer node for each classification method to compare the real class to the predicted class, show the confusion matrix and more statistics, like F-measure, precision and recall, etc.

I've chosen to go with Decision Trees and Gradient Boosted Trees as the two classification models to use on the dataset.

Student number: 25015646
Student username: bx015646

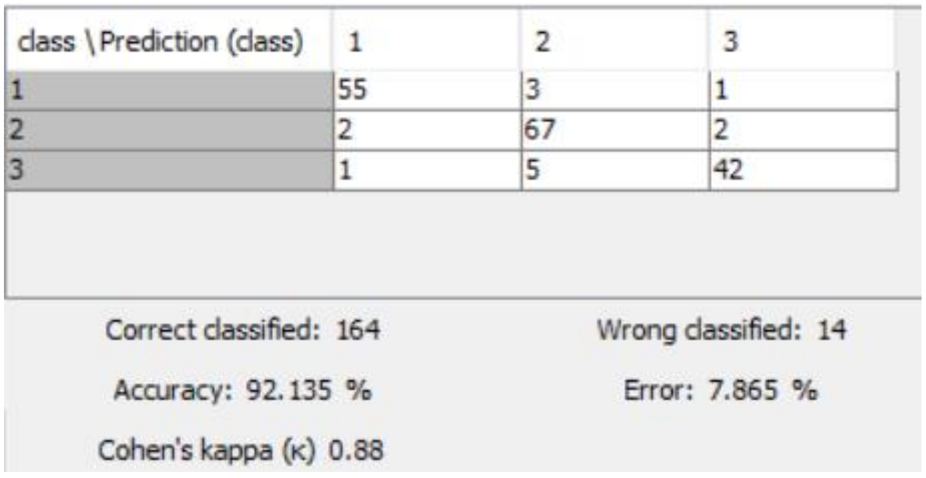
A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item to conclusions about the item's target value or class. Tree models where the target variable can take a discrete set of values are called classification trees.

Gradient boosting is a machine learning technique for regression and classification problems. It produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. The algorithm uses very shallow regression trees and a special form of boosting to build an ensemble of trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

Cross-validation is any of various similar model validation techniques for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of known data on which training is run (training dataset), and a dataset of unknown data (or first seen data) against which the model is tested (called the validation dataset or testing set). The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem).

In 10-fold cross-validation, the original sample is randomly partitioned into 10 equal sized subsamples. Of the 10 subsamples, a single subsample is retained as the validation data for testing the model, and the remaining 9 subsamples are used as training data. The cross-validation process is then repeated 10 times, with each of the 10 subsamples used exactly once as the validation data. The 10 results can then be averaged to produce a single estimation. The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation, and each observation is used for validation exactly once.

Decision tree 10-fold (randomly sampled) cross validation results:



class \ Prediction (class)	1	2	3
1	55	3	1
2	2	67	2
3	1	5	42

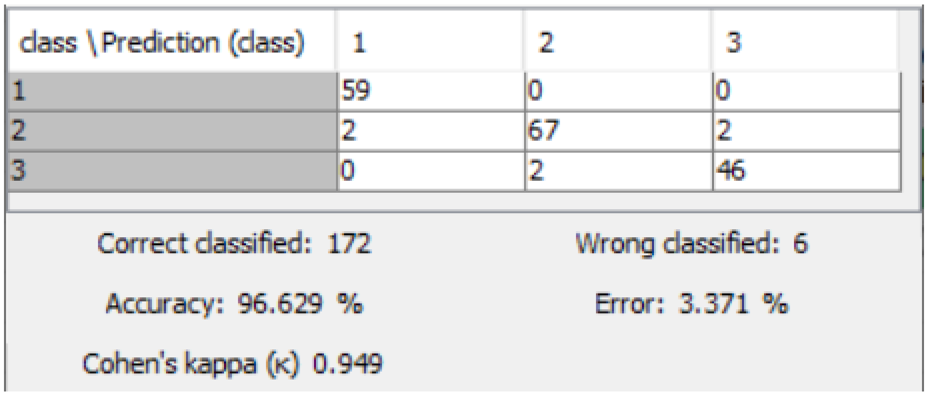
Correct classified: 164 Wrong classified: 14

Accuracy: 92.135 % Error: 7.865 %

Cohen's kappa (κ) 0.88

Student number: 25015646
Student username: bx015646

Gradient boosted trees 10-fold (randomly sampled) cross validation results:



class \ Prediction (class)	1	2	3
1	59	0	0
2	2	67	2
3	0	2	46

Correct classified: 172   Wrong classified: 6

Accuracy: 96.629 %   Error: 3.371 %

Cohen's kappa (κ) 0.949

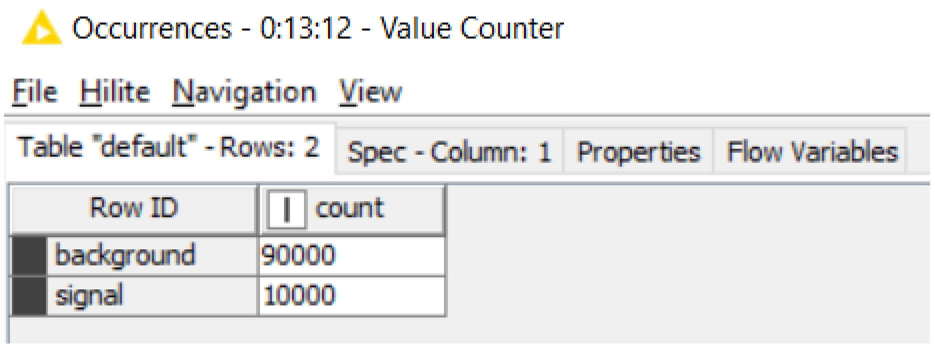
Just by looking at the accuracy as well the value of Cohen's kappa, the gradient boosted trees method looks more effective. Taking a look at more accuracy statistics such as F-measure, precision and recall, it confirms that gradient boosted trees are more effective than decision trees in this case.

Task #3

Subtask 1

For the "Rebalance classes by subsampling" metanode, I have a File Reader node going in, which will be the training set of 100 000 records, and an Equal Size Sampling node going out, which removes rows from the input data set such that the values in a categorical column are equally distributed. I included two Value Counter nodes to show the before and after effect of applying Equal Size Sampling.

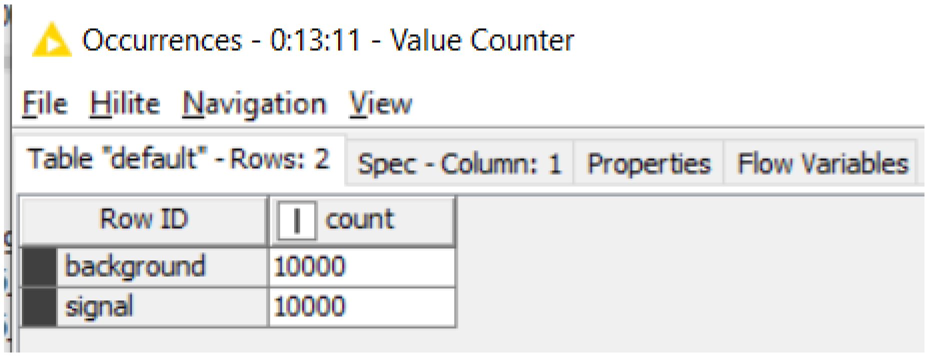
Here's the Value Counter of the original dataset with 100 000 records, 90 000 of which belong to the "background" class and the remaining 10 000 belong to the "signal" class.


Occurrences - 0:13:12 - Value Counter
File Hilite Navigation View
Table "default" - Rows: 2
Spec - Column: 1
Properties
Flow Variables

Row ID	count
background	90000
signal	10000

After using the Equal Size Sampling node, there are 10 000 records of each class:

Student number: 25015646
Student username: bx015646

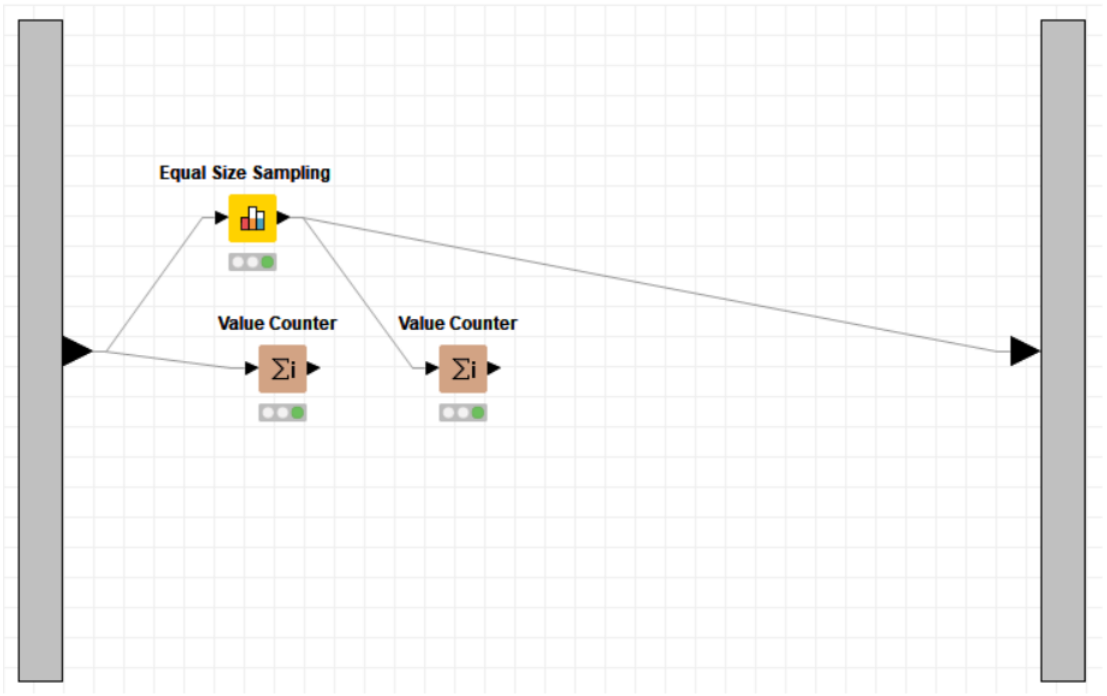

Occurrences - 0:13:11 - Value Counter

File Hilite Navigation View

Table "default" - Rows: 2 | Spec - Column: 1 | Properties | Flow Variables

Row ID	count
background	10000
signal	10000

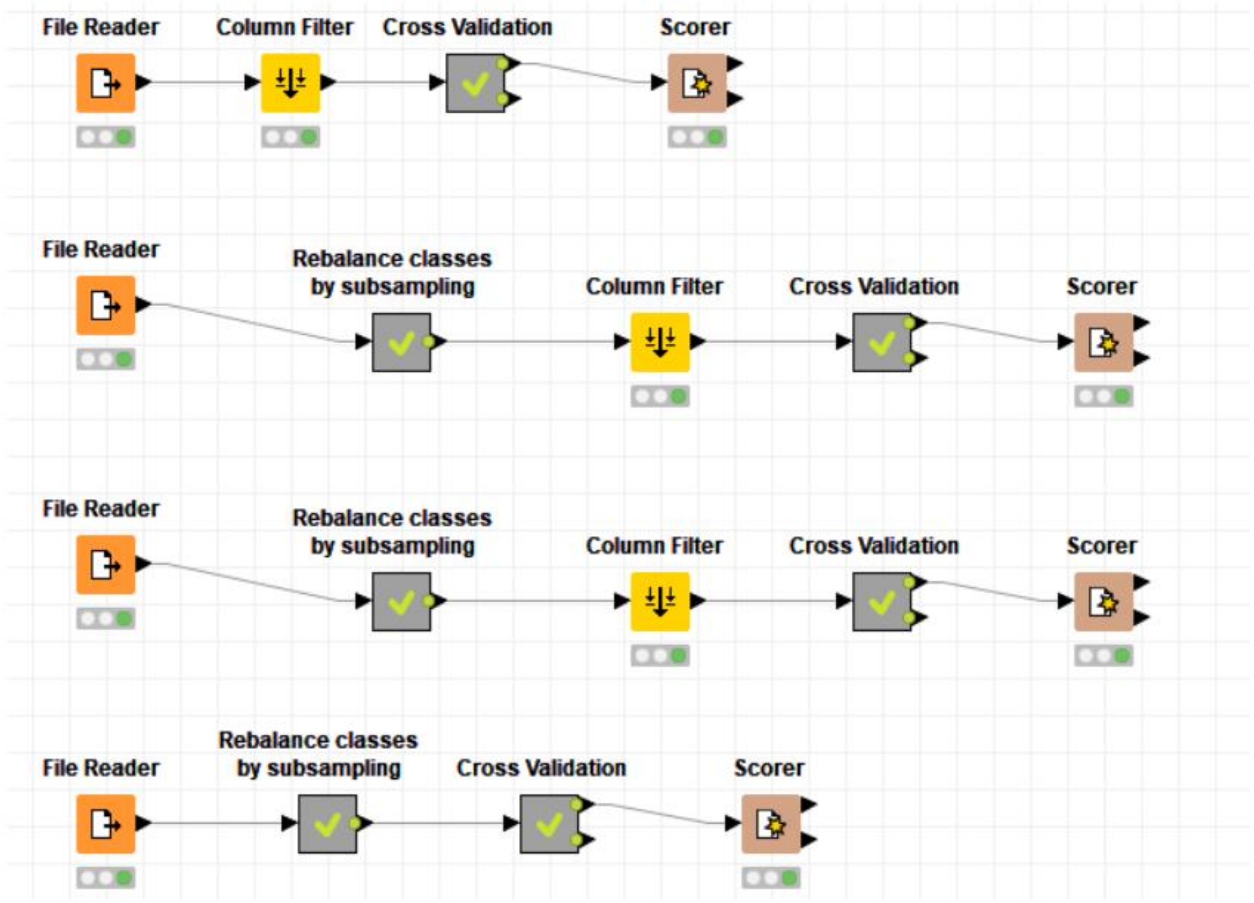
By having a balanced dataset, we avoid any classification bias. Here is the complete metanode:



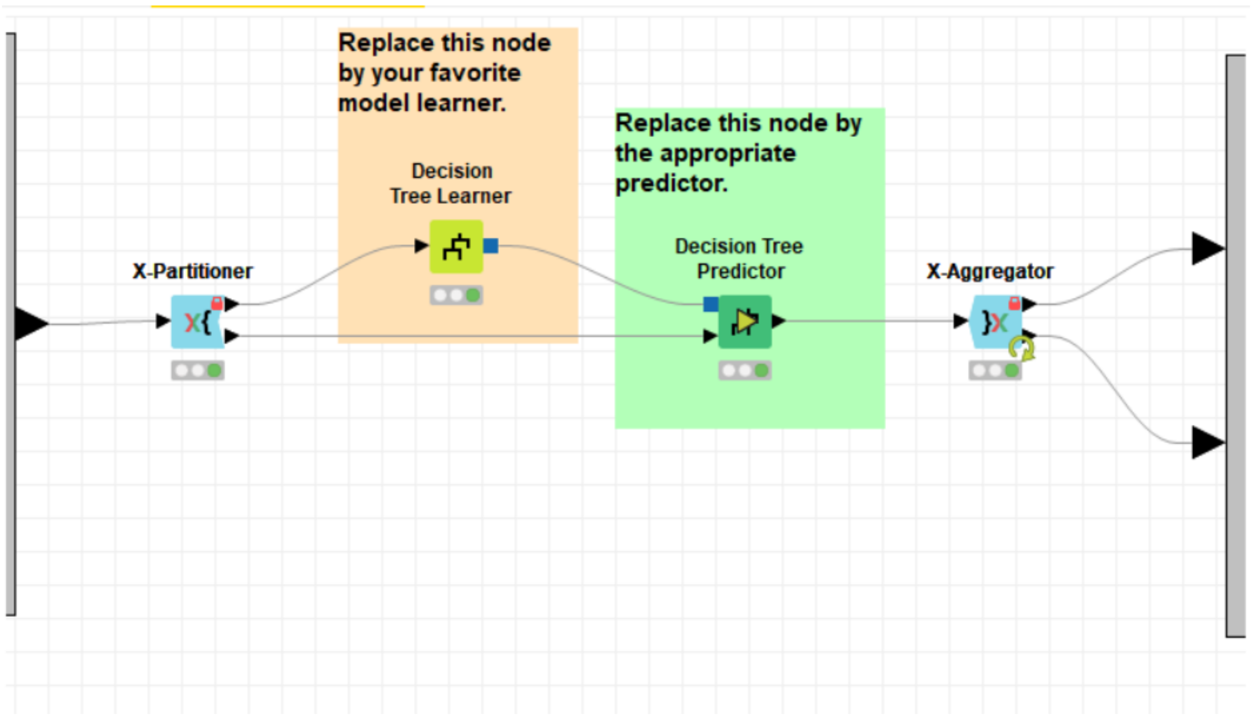
Subtask 2

I used decision trees and 10-fold (randomly sampled) cross validation to estimate and compare the classification performance in the four cases, using a Column Filter node to select the columns required and a Scorer node to evaluate the results, as shown below:

Student number: 25015646
Student username: bx015646



This is what is inside the Cross Validation metanode, which is provided by KNIME:


Replace this node by your favorite model learner.
Decision Tree Learner
Replace this node by the appropriate predictor.
Decision Tree Predictor
X-Partitioner
X-Aggregator

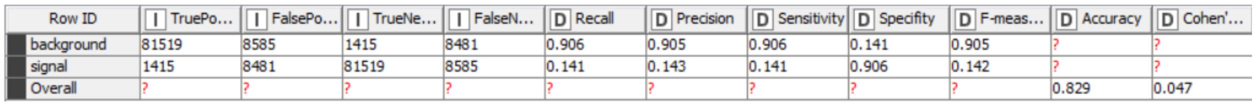
Student number: 25015646
Student username: bx015646

Performance:

In a classification task such as this, the precision for a class is the number of true positives divided by the total number of elements labeled as belonging to the positive class (the sum of true positives and false positives). Recall in this context is defined as the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives).

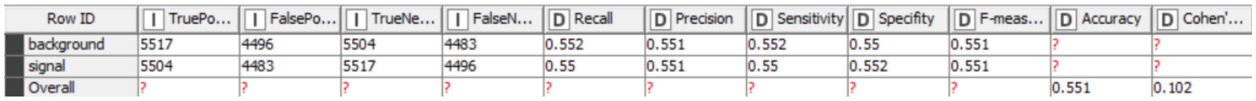
A measure that combines precision and recall is the harmonic mean of precision and recall, F-measure. F-measure reaches its best value at 1 (perfect precision and recall).

A)



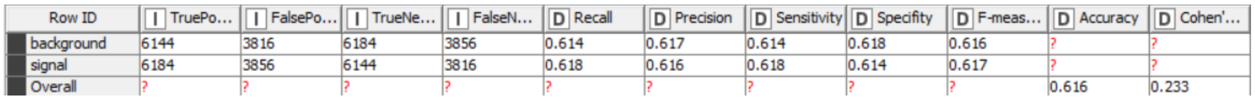
Row ID	I TruePo...	I FalsePo...	I TrueNe...	I FalseN...	D Recall	D Precision	D Sensitivity	D Specifity	D F-meas...	D Accuracy	D Cohen'...
background	81519	8585	1415	8481	0.906	0.905	0.906	0.141	0.905	?	?
signal	1415	8481	81519	8585	0.141	0.143	0.141	0.906	0.142	?	?
Overall	?	?	?	?	?	?	?	?	?	0.829	0.047

B)



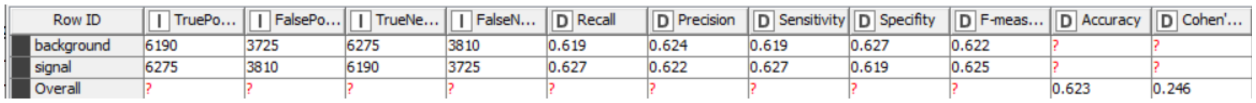
Row ID	I TruePo...	I FalsePo...	I TrueNe...	I FalseN...	D Recall	D Precision	D Sensitivity	D Specifity	D F-meas...	D Accuracy	D Cohen'...
background	5517	4496	5504	4483	0.552	0.551	0.552	0.55	0.551	?	?
signal	5504	4483	5517	4496	0.55	0.551	0.55	0.552	0.551	?	?
Overall	?	?	?	?	?	?	?	?	?	0.551	0.102

C)



Row ID	I TruePo...	I FalsePo...	I TrueNe...	I FalseN...	D Recall	D Precision	D Sensitivity	D Specifity	D F-meas...	D Accuracy	D Cohen'...
background	6144	3816	6184	3856	0.614	0.617	0.614	0.618	0.616	?	?
signal	6184	3856	6144	3816	0.618	0.616	0.618	0.614	0.617	?	?
Overall	?	?	?	?	?	?	?	?	?	0.616	0.233

D)

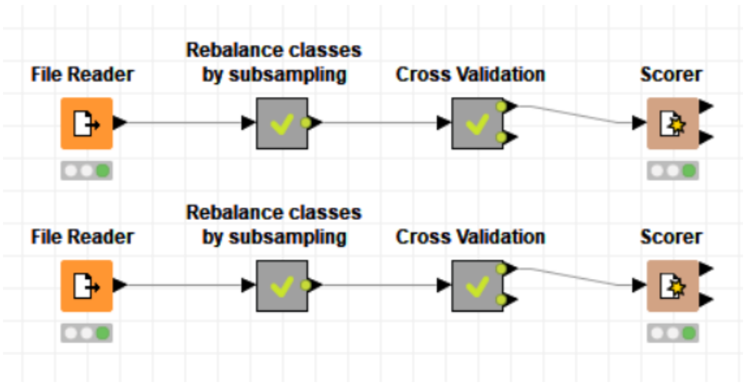


Row ID	I TruePo...	I FalsePo...	I TrueNe...	I FalseN...	D Recall	D Precision	D Sensitivity	D Specifity	D F-meas...	D Accuracy	D Cohen'...
background	6190	3725	6275	3810	0.619	0.624	0.619	0.627	0.622	?	?
signal	6275	3810	6190	3725	0.627	0.622	0.627	0.619	0.625	?	?
Overall	?	?	?	?	?	?	?	?	?	0.623	0.246

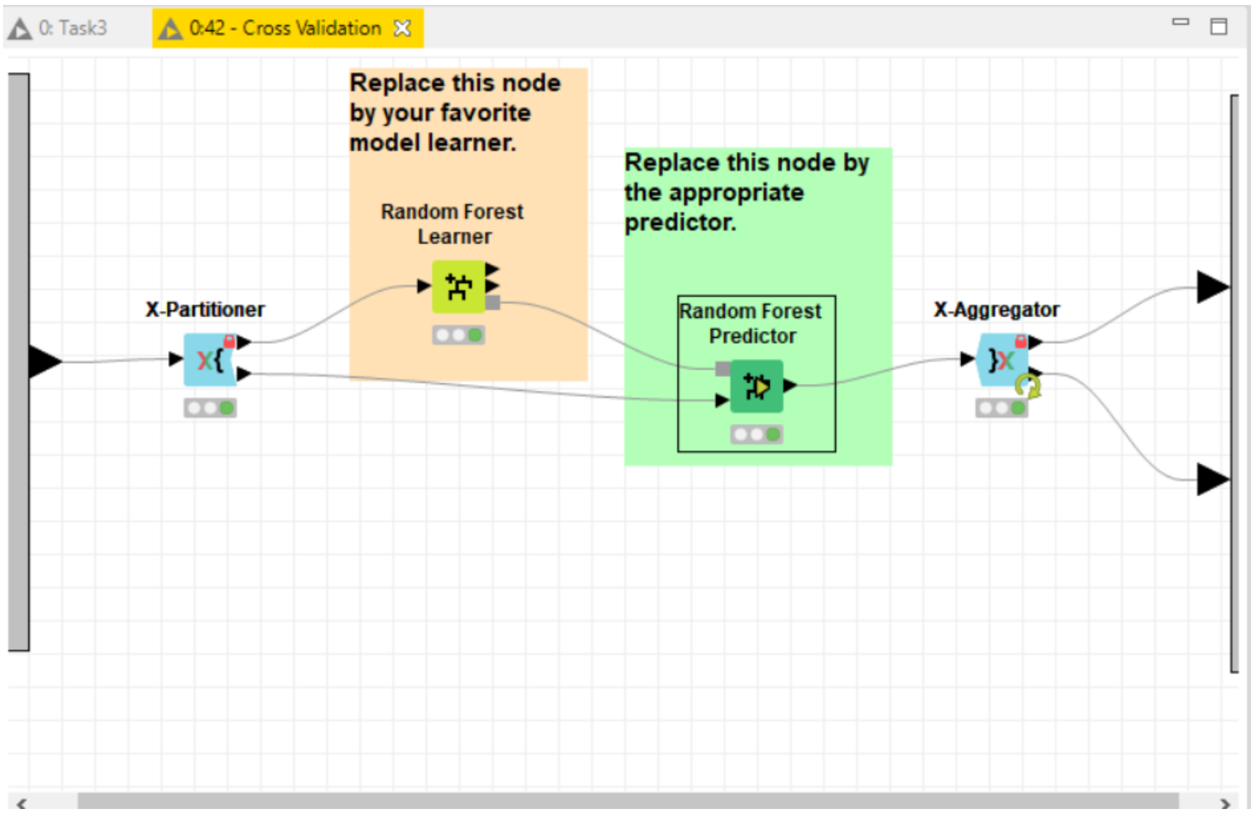
The best results are given in case D where the rebalanced training dataset with all 28 attributes is used.

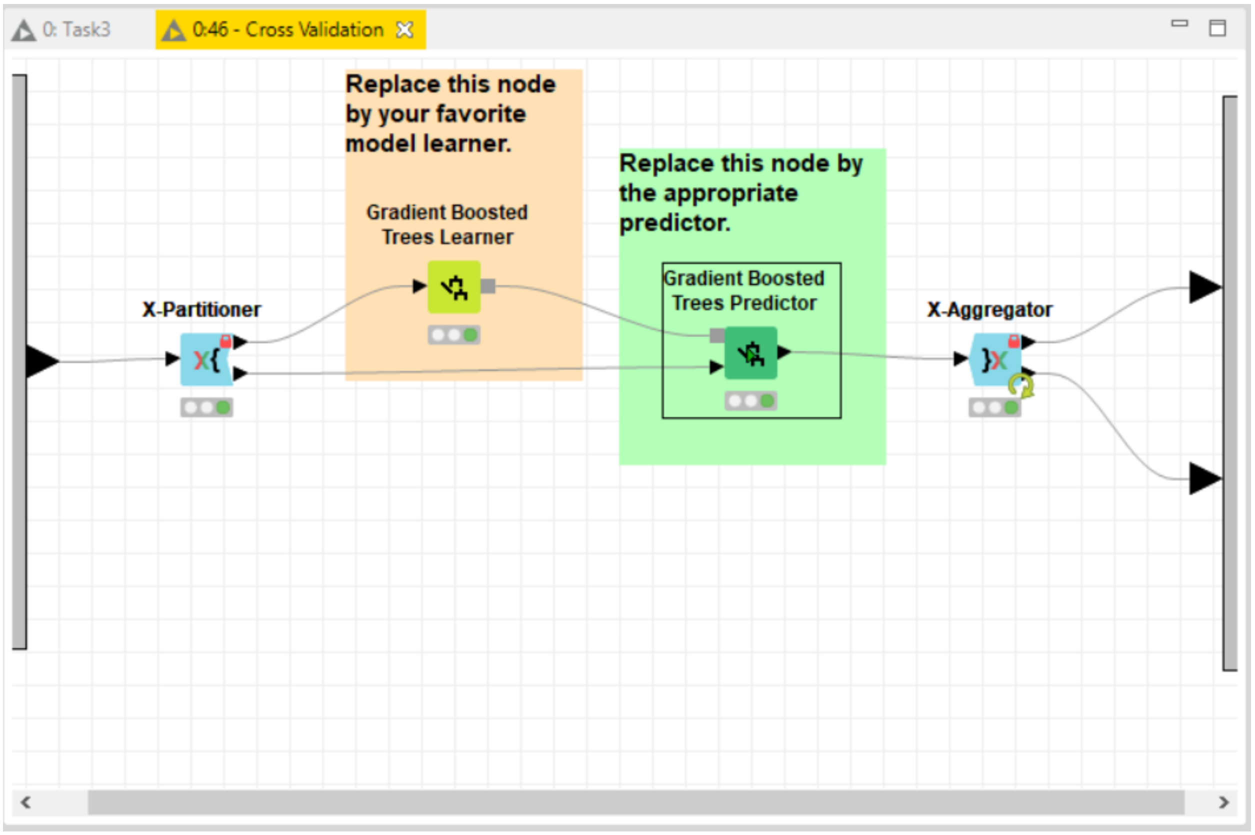
Subtask 3

Based on my research and results on Task #2, where I used two different classification algorithms, I decided to try different classification methods, in addition to the decision tree above, and compare them to find the most effective one to train this set on. Given my decision to select case D, I used the same nodes and only changed the inside of the Cross Validation metanode, like so:



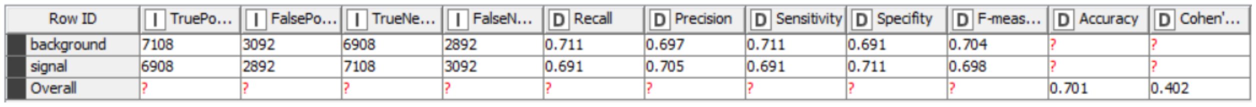
Student number: 25015646
Student username: bx015646



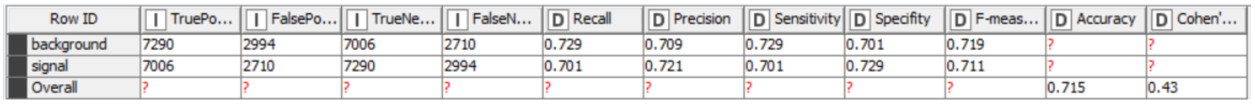

0: Task3
0:46 - Cross Validation
Replace this node by your favorite model learner.
Gradient Boosted Trees Learner
Replace this node by the appropriate predictor.
Gradient Boosted Trees Predictor
X-Partitioner
X-Aggregator

Student number: 25015646
Student username: bx015646

Both algorithms outperform the decision tree algorithm, with the results shown below:



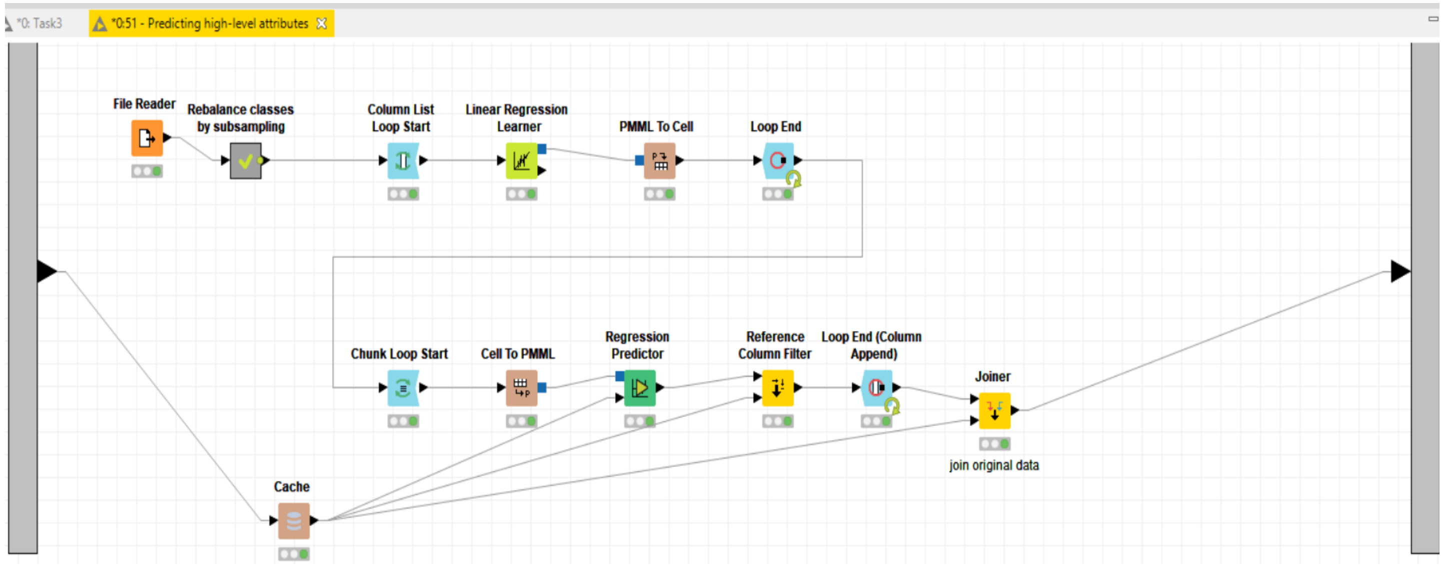
Row ID	TruePo...	FalsePo...	TrueNe...	FalseN...	Recall	Precision	Sensitivity	Specifity	F-meas...	Accuracy	Cohen'...
background	7108	3092	6908	2892	0.711	0.697	0.711	0.691	0.704	?	?
signal	6908	2892	7108	3092	0.691	0.705	0.691	0.711	0.698	?	?
Overall	?	?	?	?	?	?	?	?	?	0.701	0.402



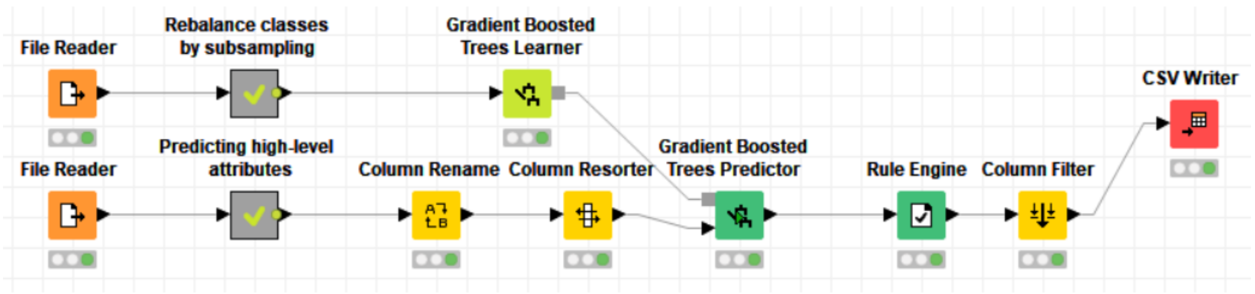
Row ID	TruePo...	FalsePo...	TrueNe...	FalseN...	Recall	Precision	Sensitivity	Specifity	F-meas...	Accuracy	Cohen'...
background	7290	2994	7006	2710	0.729	0.709	0.729	0.701	0.719	?	?
signal	7006	2710	7290	2994	0.701	0.721	0.701	0.729	0.711	?	?
Overall	?	?	?	?	?	?	?	?	?	0.715	0.43

The gradient boosted trees algorithm performs the best out of the three, therefore I have decided to use that to predict the classes of the records in the test set.

Before predicting the classes for the records in the test set, we need to predict the 7 high level attributes that are not present in the test set. To predict the high-level attributes, I used a multiple target prediction technique by using loops and regression. The first loop selects one target column per loop. We need to set the target column in the learner to the flow variable of the corresponding column name. I have trained the learner on the rebalanced training set of the 20 000 records. The second loop applies the PMML to unseen data. Lastly, it combines all the predictions in one table, as shown below:



To finally classify the test set, I have done the following:

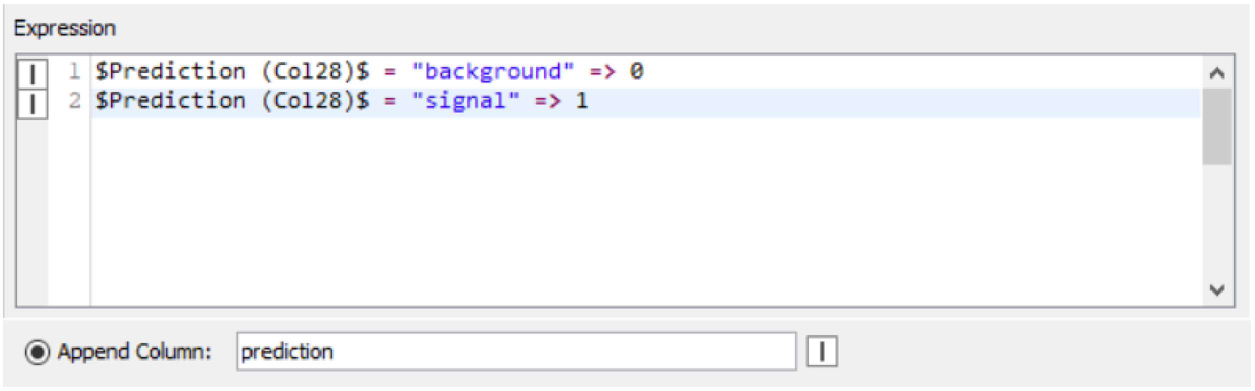


File Reader
Rebalance classes by subsampling
Gradient Boosted Trees Learner
CSV Writer
Predicting high-level attributes
File Reader
Column Rename
Column Resorter
Gradient Boosted Trees Predictor
Rule Engine
Column Filter

Student number: 25015646
Student username: bx015646

I used a gradient boosted trees learner node to train the model based on the down-sampled training set of 20 000 records. For the test set, I predicted the 7 high-level attributes and, after leaving the metanode, I renamed and resorted the columns to make everything neat and organized. Afterwards, I used a gradient boosted trees predictor, to predict the class of the test set, given the trained model and the reorganized test set.

After the classification is done, I used the Rule Engine node to create a column called "prediction" that will turn any record belonging to class "background" to 0 and any record that belongs to class "signal" to 1, like so:



```
$Prediction (Col28)$ = "background" => 0
$Prediction (Col28)$ = "signal" => 1
```

Append Column: prediction

Finally, I used Column Filter to only keep the prediction column and CSV Writer to save the table as a csv file on my disk, with the option to "Write row ID" checked.

References

ftp://statgen.ncsu.edu/pub/thorne/molevoclass/AtchleyOct19.pdf

https://blog.easysol.net/machine-learning-algorithms-3/

https://en.wikipedia.org/wiki/Decision_tree

https://en.wikipedia.org/wiki/Decision_tree_learning

https://en.wikipedia.org/wiki/Gradient_boosting

https://en.wikipedia.org/wiki/Precision_and_recall

https://en.wikipedia.org/wiki/F1_score