CSMDM16 Assignment

Jin Li (26801076)

2017/10/08

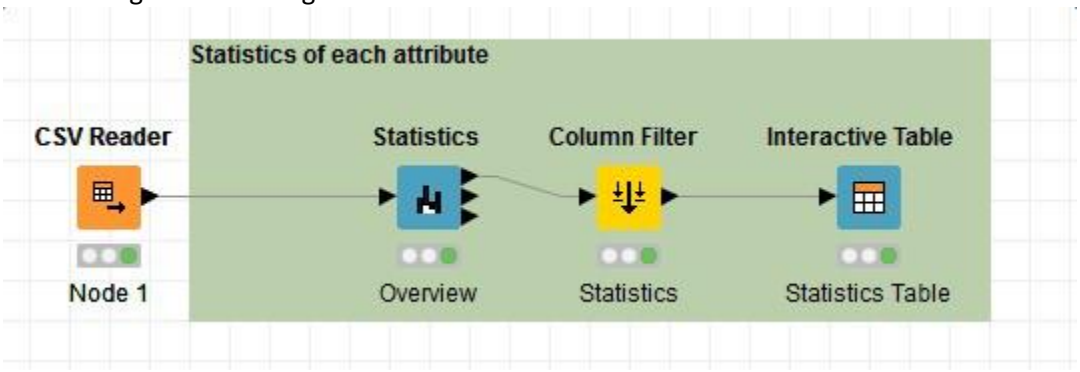
Problem#1-Hierarchical Clustering

You are required to extract and provide some statistics (task #1) about the dataset "teeth.csv" (available on Blackboard): these should include the number of records, the number of attributes, the range and mean value of each attribute, and the histogram for each attributes. You are required to apply hierarchical clustering to all the records in the dataset and to visualise the resulting dendrogram. The dendrogram should report the class labels of the data records at the leaf nodes. The generation of the dendrogram should be performed twice: with KNIME (task #2) and with R (task #3). The corresponding KNIME workflow and the R code should be reported and discussed.
In your coursework report, you should include:

1. a brief description of the adopted solutions (data workflows, KDD processes),
2. a brief description of the adopted data mining algorithms,
3. images of KNIME workflows and their presentation (incl. relevant node configurations),
4. R code and its presentation, and
5. the results and their critical analysis

Solution:

Task#1: Use Knime workflow to generate statistics of each attributes of dataset "teeth.csv"
Use R to generate histogram of each attributes of dataset "teeth.csv"



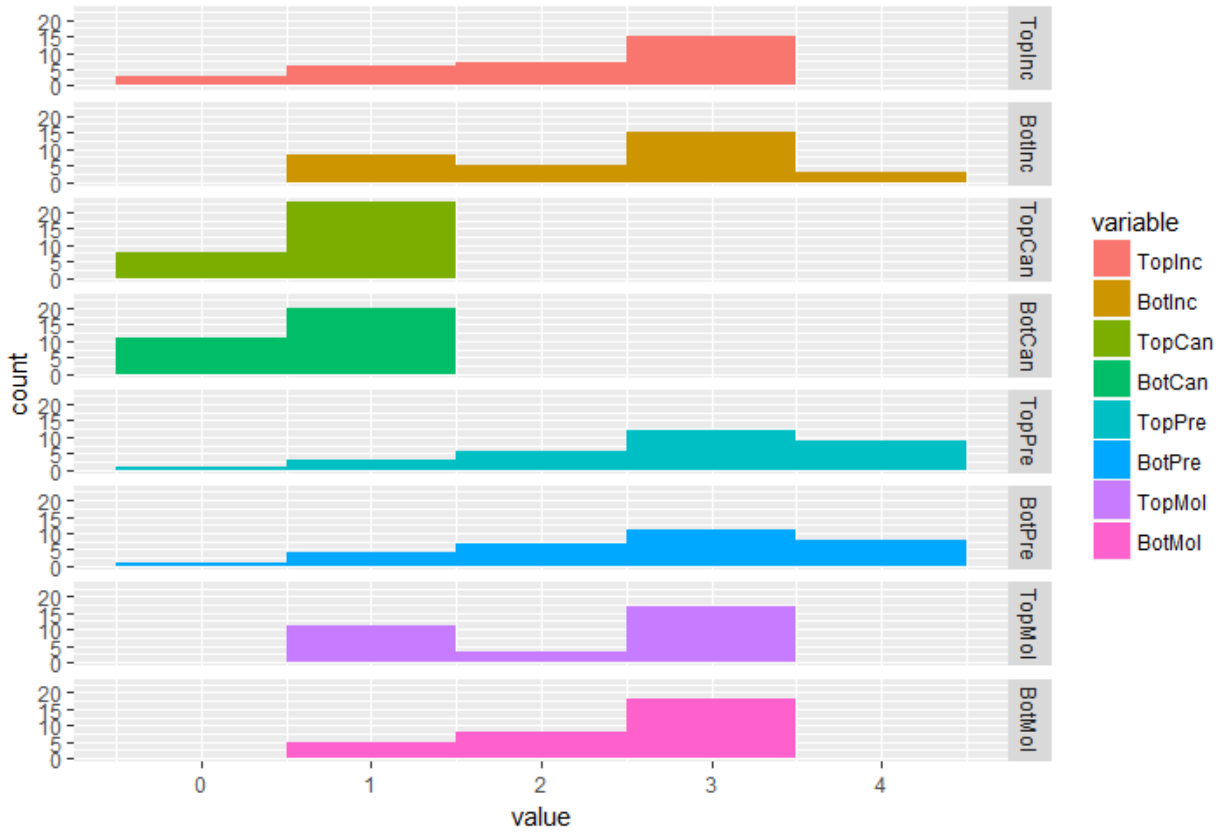
Summary of min,max,mean ,Stddev and number of each attributes

Row ID	S Column	D Min	D Max	D Mean	D Std. de...	I Row count
TopInc	TopInc	0	3	2.097	1.044	31
BotInc	BotInc	1	4	2.419	0.992	31
TopCan	TopCan	0	1	0.742	0.445	31
BotCan	BotCan	0	1	0.645	0.486	31
TopPre	TopPre	0	4	2.806	1.078	31
BotPre	BotPre	0	4	2.677	1.107	31
TopMol	TopMol	1	3	2.194	0.946	31
BotMol	BotMol	1	3	2.419	0.765	31

And here is a histogram of each attribute of the dataset

```
# Histogram for each attributes
library(gdata)
```

```
library(ggplot2)
library(reshape2)
mydata<-read.csv(file="C:/Users/Jerry/Desktop/Data Analystics and Mining/assignment/teeth(1).csv"
          ,header=TRUE, sep=",")
gg <- melt(mydata)
ggplot(gg, aes(x=value, fill=variable)) +geom_histogram(binwidth=1)
+facet_grid(variable~.)
```



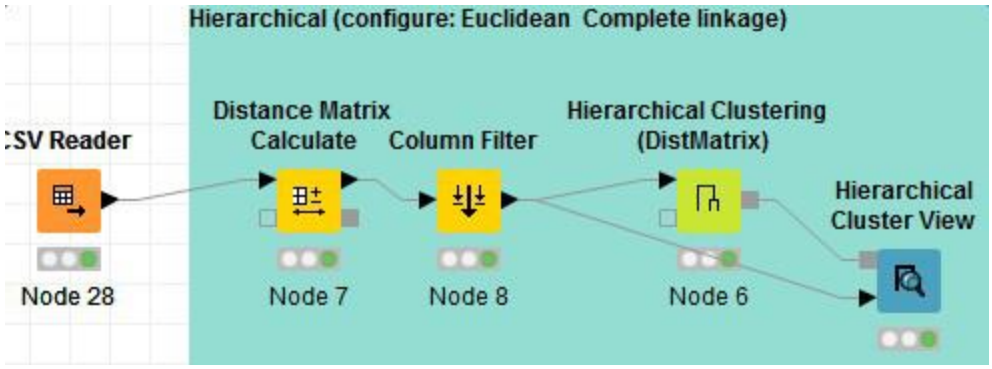
Task#2:

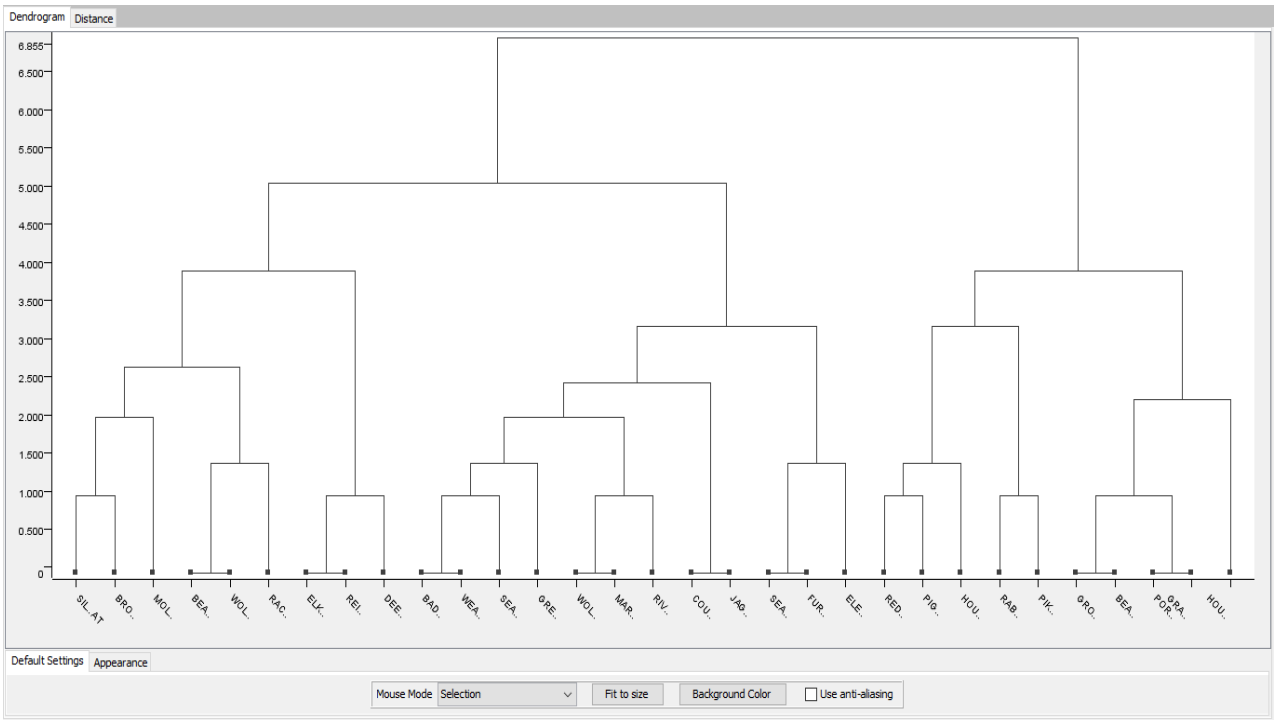
Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types: Agglomerative and Divisive[1]

Knime solution: It works agglomerative with Euclidean Distance Function and Complete Linkage

1. loading the CSV file using a CSVReader node where we provide the filename
2. Once the data file has been read, we can perform hierarchical clustering using the Hierarchical Clustering node simply or we can get the result step by step, calculating distance matrix first and clustering with it to display the Hierarchical Cluster view. The Euclidian distance function was chose in distance matrix and set linkage type to Complete to get similar results to the clustering performed in R. The image below shows the KNIME flow and the output of the hierarchical clustering.

[1] Rokach, Lior, and Oded Maimon. "Clustering methods." Data mining and knowledge discovery handbook. Springer US, 2005. 321-352



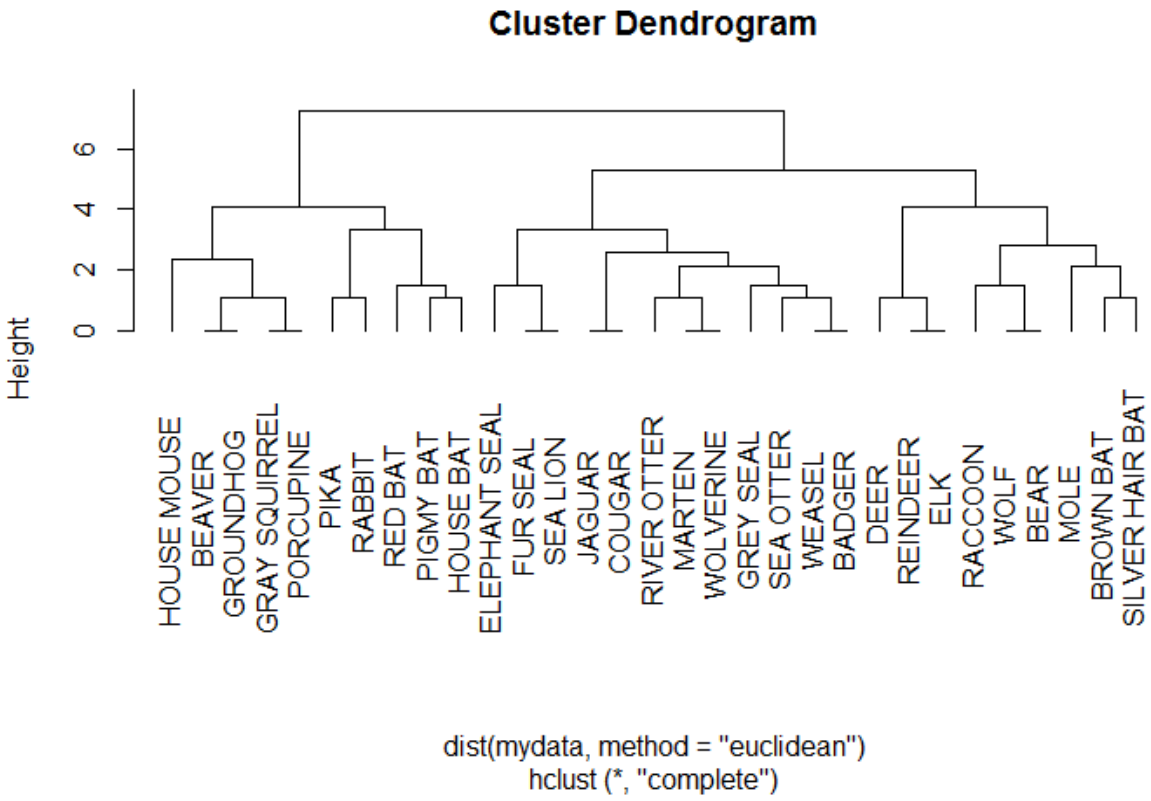


Task#3:

R solution: It works agglomerative with Euclidean Distance Function and Complete Linkage

We can set the number of cluster, but it will not affect the result, only can be display with different colours

```
# Hierarchical clustering
library(gdata)
mydata=read.csv(file="C:/Users/Jerry/Desktop/Data Analystics and Mining/assignment/teeth(1).csv",header=TRUE, sep=",")
hc<-hclust(dist(mydata,method = "euclidean"),method="complete")
plot(hc,hang=-1,labels=mydata$Animal)
```



In summary, it is almost the same if we use the same distance function and the method to compare two clusters.

Problem #2 – Classification

You are required to build and test a classification model for four datasets (task #4&5):

- the iris dataset
- https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data
- https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancerwisconsin/breast-cancer-wisconsin.data
- a dataset of your choice (excluding iris, wine, breast-cancer-wisconsin, teeth)

The description of the wine and breast-cancer-wisconsin datasets (including column headers) is provided in the corresponding files with extension ".names".

- https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.names
- https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancerwisconsin/breast-cancer-wisconsin.names

You can use any classification algorithm. You are required to carry out a comparison of the accuracy (average and standard deviation over 20 trials) of these four evaluation methods for each of the four datasets using the same classification algorithm.

- resub: resubstitution error method
- hold-out-10%: hold-out method with 10% - 90% partition split
- xVal-10f: 10-fold cross-validation method
- LOOCV: leave-one-out cross-validation method

The KNIME workflows (task #4) and the R code (task #5) must be included and commented in the coursework report just for one of the datasets. The results should be presented (e.g., by means of one bar chart for each dataset to compare the four methods) and commented. Using either KNIME or R, and only for the breast-cancer-wisconsin dataset an analysis of the accuracy over different sample sizes should also be produced (task #6). This can be done by randomly selecting a sample (e.g., with size varying from 100 to n) from the original dataset D (|D|=n) and to estimate the accuracy with the four methods. The results should be presented with four charts of the accuracy (avg and stddev) versus the sample size, one for each evaluation method. Finally, an analysis of the running time (task #7) and some conclusions should be provided.

Solution description:

We use Decision tree learning as a method to build a classification model to predict the value of a target variable based on several input variables. A decision tree can be used to visually and explicitly represent decisions and decision making.

1) **Decision Tree in Knime**
   Most of the techniques used in decision tree implementation in Knime involved with SPRINT algorithms that removes all of the memory restrictions, and is fast and scalable.[i]
2) **Decision Tree in R**
   We use package rpart for recursive partitioning in Decision Tree implementation of R, which implement many of the ideas found in the Classification and Regression Tree (CART) [ii]

In order to evaluate the accuracy of model, we use four different evaluation methods.

1) **Resub:** It uses training dataset as testing dataset to predict the value of class. To be fair with other method, we use 90% of data as training dataset
2) **Hold-out-10%:** It is partitioned into 90% and 10%of dataset. 90% of dataset as training dataset ,while 10% as the testing dataset.
3) **xVal-10f:** The dataset is randomly partitioned into 10 equal sized subsamples. Of the 10 subsamples, a single subsample is data for testing the model, and the remaining 9 subsamples are used as training data.
4) **LOOCV:** It is a particular case of leave-p-out cross-validation with p = 1. For this comparison, we use 90% of dataset plus one data as sample dataset.

Task#4 : To compare the accuracy and Stddev of four evaluation methods with Knime

Setting: 20 trials

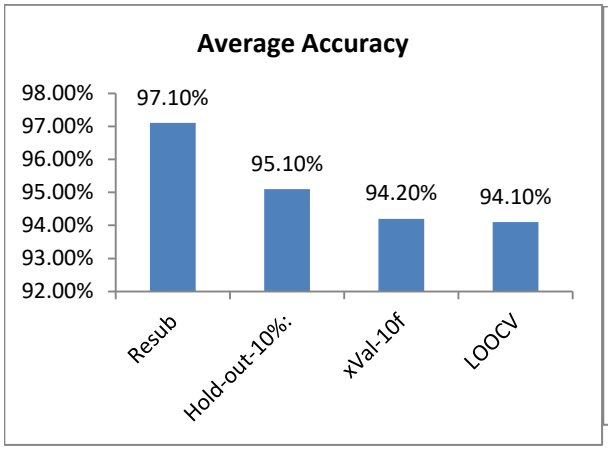
Decision Tree Learner : 5 records per node

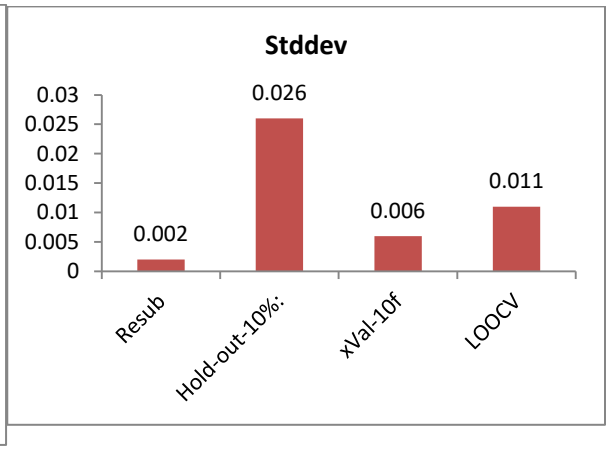
Knime solution-breast-cancer-wisconsin (The breast-cancer-wisconsin dataset has 10 attributes with 2 classes)

Evaluation method	Average Accuracy	Stddev
Resub	97.1%	0.002
Hold-out-10%:	95.1%	0.026
xVal-10f	94.2%	0.006
LOOCV[2]	94.1%	0.011

The accuracy of Resubstitution method still is highest. The hold out is not stable as to highest standard deviation. xVal-10f and LOOCV are similar with accuracy and standard deviation.
Please find the workflow after the summary.

[2] This is real LOOCV , running time about 6 hours





Knime solution-iris (The iris dataset has 4 attributes with 3 classes)

Evaluation method	Average Accuracy	Stddev
Resub	97.1%	0.006
Hold-out-10%:	96%	0.066
xVal-10f	95.5%	0.012
LOOCV[3]	95.2%	0.011

Knime solution-Wine (The wine dataset has 13 attributes with 3 classes)

Evaluation method	Average Accuracy	Stddev
Resub	96%	0.013
Hold-out-10%:	86.7%	0.094
xVal-10f	89.3%	0.014
LOOCV[4]	89.4%	0.038

Knime solution-Tae (The Tae dataset has 5 attributes with 3 Classes)

Evaluation method	Average Accuracy	Stddev
Resub	72.1%	0.022
Hold-out-10%:	51.6%	0.093
xVal-10f	50.8%	0.024
LOOCV[5]	50.4%	0.038

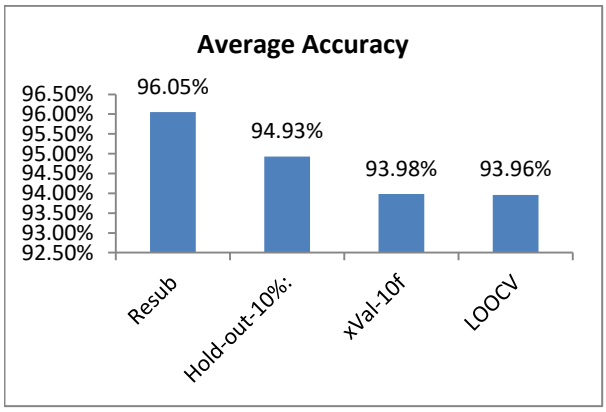
For data of wine and Tae, except Resub, the other three methods have similar result, which means the real accuracy of decision tree model is around 50%, not 72%. This is evidence that using training dataset to do the test is not correct.

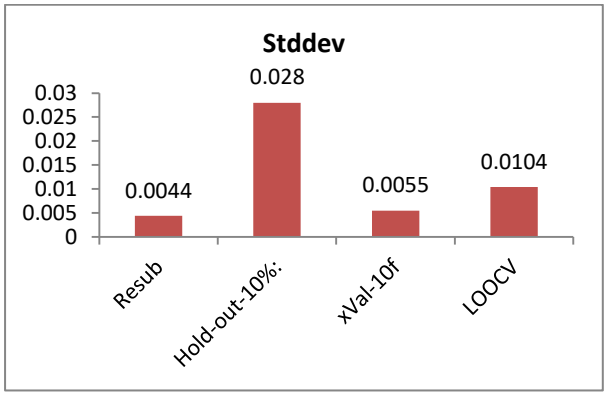
[3] Instead of LOOCV, I use K=30
[4] Instead of LOOCV, I use K=50
[5] Instead of LOOCV, I use K=50

Task#5: To compare the accuracy and Stddev of four evaluation methods with R

R solution-breast-cancer-wisconsin (The breast-cancer-wisconsin dataset has 10 attributes with 2 classes)

Evaluation method	Average Accuracy	Stddev
Resub	96.05%	0.0044
Hold-out-10%:	94.93%	0.0280
xVal-10f	93.98%	0.0055
LOOCV	93.96%	0.0104





In summary, the accuracy method of Resubstitution is the highest, because it uses training dataset as testing set, while standard deviation of the Hold-out-10% is the highest. The accuracy of xVal-10f and LOOCV are similar ,but the LOOCV has higher standard deviation, which means xVal-10f is the best

This is the script of R for Wine dataset analysis

```
# loading the data
library(gdata)

mydata < -read.csv(file="C:/Users/Jerry/Desktop/Data Analystics and Mining/assignment/wine.csv"
            , header=TRUE, sep=",")
# number to factor of class
mydata$Class < -  as.factor(mydata$Class)

# First :Resubstitution----------------------------------------------------------------------------------------------------
library("rpart")
acc_array < -array()
print(acc_array)
for (i in 1: 20) {
   sampleRate < -0.1
sampleSize < -round(nrow(mydata) * sampleRate)
testSampleIdx < -sample(nrow(mydata), size=sampleSize)
# testset
testSet < -mydata[testSampleIdx,]
# trainingSet
trainingSet < - mydata[-testSampleIdx,]
mydata.dt < -rpart(Class
~Alcohol + Malic.acid + Ash + Alcalinity.of.ash + Magnesium
+ Total.phenols + Flavanoids + Nonflavanoid.phenols
+ Proanthocyanins + Color.intensity + Hue
+ OD280.OD315.of.diluted.wines + Proline
, data = trainingSet, method = "class")
```

```
# Resubstitution method: acc/error on the training set
Prediction < -predict(mydata.dt, newdata=trainingSet, type="class")
cM < - table(trainingSet$Class, Prediction)
acc < - sum(diag(cM)) / sum(cM)
acc_array[i] < - acc
}

Acc_mean < -mean(acc_array)
print(Acc_mean)
Acc_sd < -sd(acc_array)
print (Acc_sd)

# Second -hold out-10%----------
library("rpart")
acc_array < -array()
print(acc_array)
for (i in 1: 20) {
   sampleRate < -0.1
sampleSize < -round(nrow(mydata) * sampleRate)
testSampleIdx < -sample(nrow(mydata), size=sampleSize)
# testset
testSet < -mydata[testSampleIdx,]
# trainingSet
trainingSet < - mydata[-testSampleIdx,]
mydata.dt < -rpart(Class
~Alcohol + Malic.acid + Ash + Alcalinity.of.ash + Magnesium
+ Total.phenols + Flavanoids + Nonflavanoid.phenols
+ Proanthocyanins + Color.intensity + Hue
+ OD280.OD315.of.diluted.wines + Proline
, data = trainingSet, method = "class")
Prediction < -predict(mydata.dt, newdata=testSet, type="class")
cM < - table(testSet$Class, Prediction)
acc < - sum(diag(cM)) / sum(cM)
acc_array[i] < - acc
}

Acc_mean < -mean(acc_array)
print(Acc_mean)
Acc_sd < -sd(acc_array)
print (Acc_sd)
##Third- 10-fold xVal----------

library("caret")
library("rpart")

acc_array < -array()
p < -10
for (i in 1: 20)
{
   correct_prediction < -0
require(caret)
folds < -createFolds(c(1:nrow(mydata)), p)
for (j in 1: p)
{
   testSet < -mydata[folds[[j]],]
trainingSet < -mydata[-folds[[j]],]
mydata.dt < -rpart(Class
~Alcohol + Malic.acid + Ash + Alcalinity.of.ash + Magnesium
```

```
+ Total.phenols + Flavanoids + Nonflavanoid.phenols
+ Proanthocyanins + Color.intensity + Hue
+ OD280.OD315.of.diluted.wines + Proline
, data = trainingSet, method = "class")
Prediction < -predict(mydata.dt, newdata=testSet, type="class")
cM < - table(testSet$Class, Prediction)
correct_prediction = correct_prediction + sum(diag(cM))
}
acc < -correct_prediction / nrow(mydata)
acc_array[i] < - acc
}
Acc_mean < -mean(acc_array)
print(Acc_mean)
Acc_sd < -sd(acc_array)
print (Acc_sd)
###Fourth-LOOCV method----------
library("caret")
library("rpart")

acc_array < - array()
for (i in 1: 20)
{
   correct_prediction < - 0

samplesize < - round(nrow(mydata) * 0.9) + 1
sampleindex < - sample(nrow(mydata), size=samplesize)
sampleset < - mydata[sampleindex,]
for (j in 1: length(sampleindex))
{
   trainingSet < - sampleset[-j,]
testSet < - sampleset[j,]
mydata.dt < -rpart(Class
~Alcohol + Malic.acid + Ash + Alcalinity.of.ash + Magnesium
+ Total.phenols + Flavanoids + Nonflavanoid.phenols
+ Proanthocyanins + Color.intensity + Hue
+ OD280.OD315.of.diluted.wines + Proline
, data = trainingSet, method = "class")
Prediction < -predict(mydata.dt, newdata=testSet, type="class")
cM < - table(testSet$Class, Prediction)
correct_prediction = correct_prediction + sum(diag(cM))
}
acc < - correct_prediction / length(sampleindex)
acc_array[i] < - acc
}
Acc_mean < - mean(acc_array)
print(Acc_mean)
Acc_sd < - sd(acc_array)
print (Acc_sd)
```

The analysis results of other three datasets are as below:

R solution-iris (The iris dataset has 4 attributes with 3 classes)

Evaluation method	Average Accuracy	Stddev
Resub	96.37%	0.0053
Hold-out-10%:	93.33%	0.0612
xVal-10f	93.43%	0.0095
LOOCV	93.97%	0.0202

R solution-Wine (The wine dataset has 13 attributes with 3 classes)

Evaluation method	Average Accuracy	Stddev
Resub	94.84%	0.0175
Hold-out-10%:	87.78%	0.0665
xVal-10f	88.29%	0.0121
LOOCV	88.82%	0.0291

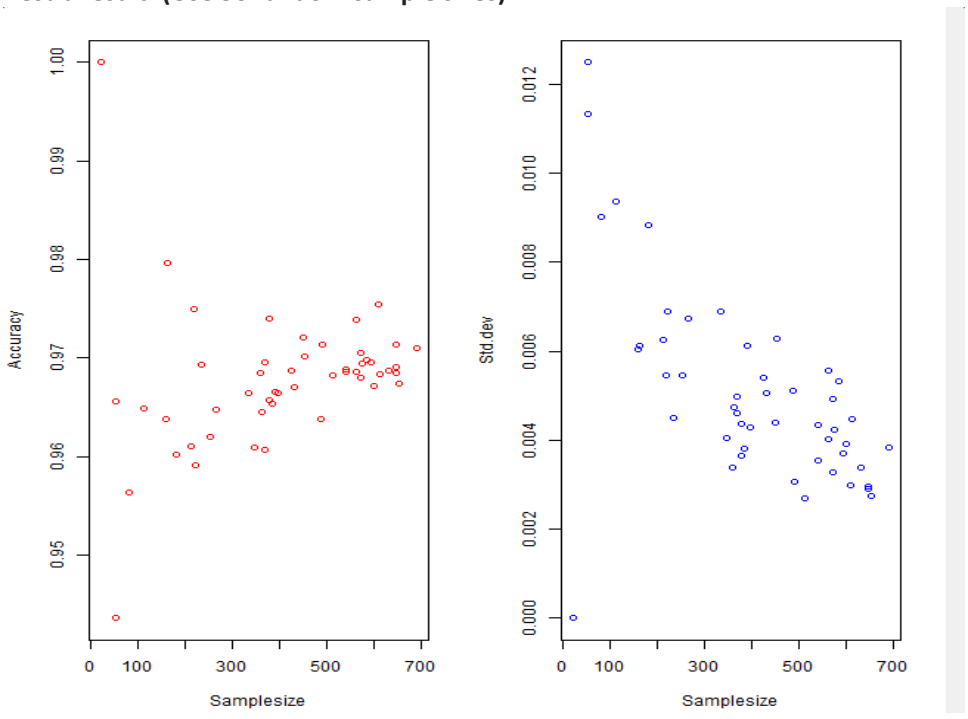
R solution-Tae (The Tae dataset has 5 attributes with 3 Classes)

Evaluation method	Average Accuracy	Stddev
Resub	65.29%	0.02258
Hold-out-10%:	53.33%	0.1451
xVal-10f	51.92%	0.03876
LOOCV	51.17%	0.06515

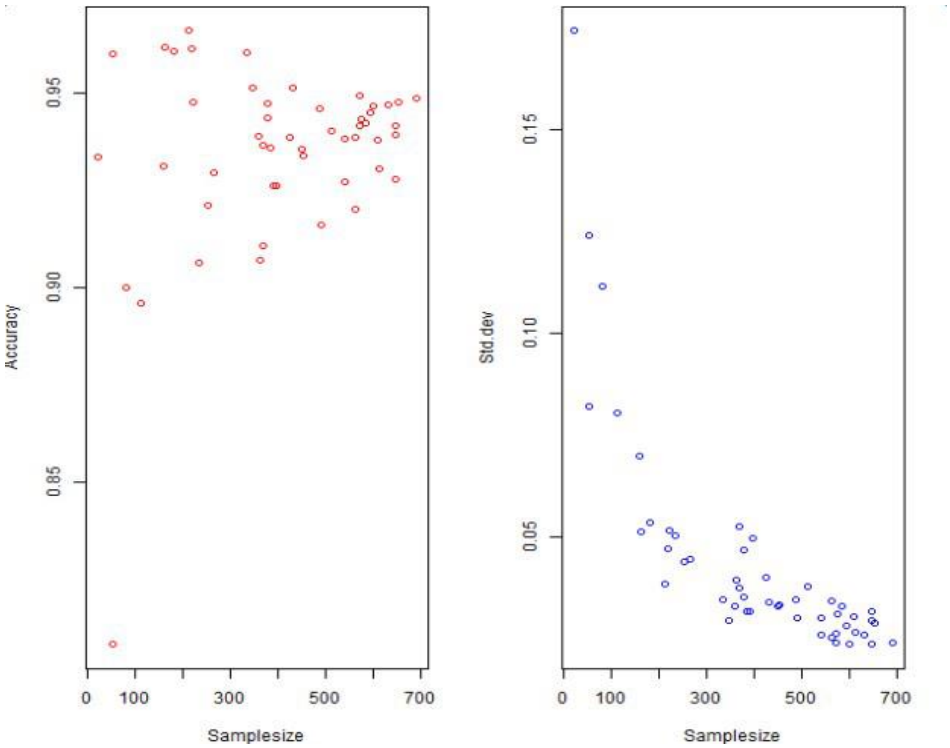
Task#6: To analyse the effect of the sample size to accuracy

Knime Solution : Test these three method **Resub/ Hold-out-10%/ xVal-10f**
Data: **breast-cancer-wisconsin**
Using Data Generator generate node gets 50 or 30 of random number between 0 and 1, which will create the size of sample. For each sample, it will iterate 20 times to generate average accuracy and standard deviation. Still setting Decision Tree Learner with 5 records per node.

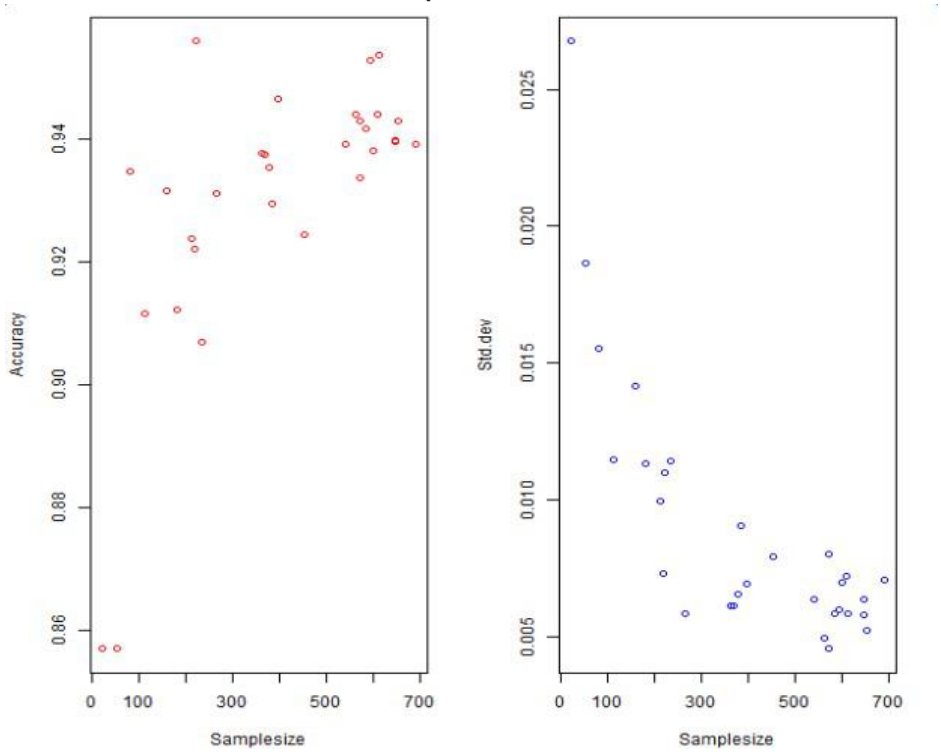
Resub result: (Use 50 random sample sizes)



Hold-out-10% result: (Use 50 random sample sizes)



xVal-10f result: Use 30 random sample sizes



As the size of sample increases, the standard deviation of accuracy will be low and the accuracy will be much closer to the mean of accuracy. The sample size analysis with LOOCV in Knime probably need over 24 hours running time.

Task#7: Analyse the running time for these four evaluation methods
R Solution : 20 trial to get average accuracy
Test1 : Different sample size with the same number of attributes
Data: **breast-cancer-wisconsin** (699 points)

Test1

Evaluation method	**Whole data (699points) Running time(seconds)**	**Half data(350 points) Running time(seconds)**
Resub	0.53	0.51
Hold-out-10%:	0.59	0.50
xVal-10f	3.58	2.64
LOOCV	172.84	74.12

Test2 : the same sample size with the different number of attributes
Data: **breast-cancer-wisconsin** (150 points and 10 attributes) VS **Iris**(150 points and 4 attributes)

Test2

Evaluation method	**breast-cancer-wisconsin (10 attributes) Running time(seconds)**	**Iris(4 attributes) Running time(seconds)**
Resub	0.33	0.34
Hold-out-10%:	0.36	0.33
xVal-10f	1.65	1.34
LOOCV	19.6	15.35

The running time of Resub and Hold-out10% is very short, there is no much difference with different sample size or different attributes, but the sample size and number of attributes affect the running time of xVal-10f and LOOCV.

Knime Solution : 20 trial to get average accuracy

Running time(seconds)

Evaluation method	**Effect factor**
Resub	sample size, min number record per node
Hold-out-10%:	sample size, min number record per node
xVal-10f	sample size, min number record per node
LOOCV	Sample size

The running time of LOOCV mainly depends on the sample size, because each iteration, it will loop N times.

Summary
These four evaluation methods have their own advantages and disadvantages. Resub has the highest accuracy but not reliable. Hold-out-10% is very time saving but with highest standard deviation. LOOCV is very time consuming, which is not useful with big sample size. Considering the accuracy , standard deviation and running time, xVal-10f should be the best choice.

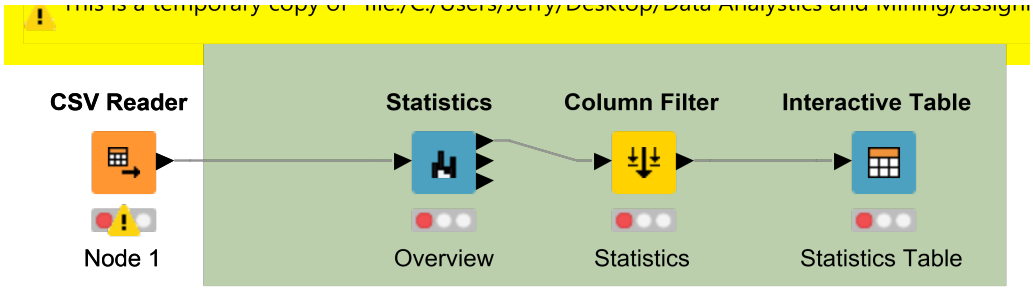
Discussion

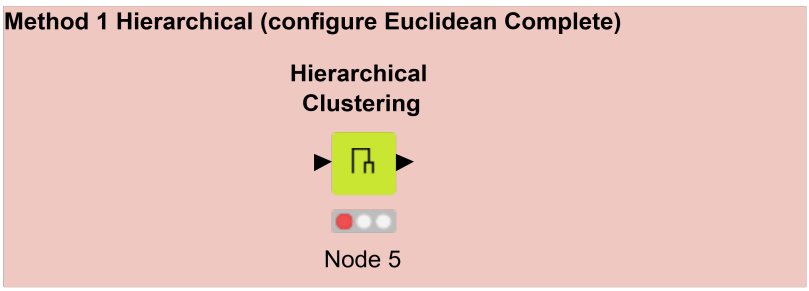
Solving problems with R and Knime in two environments can be integrated very well. Knime is more visualization, while R has more functions.

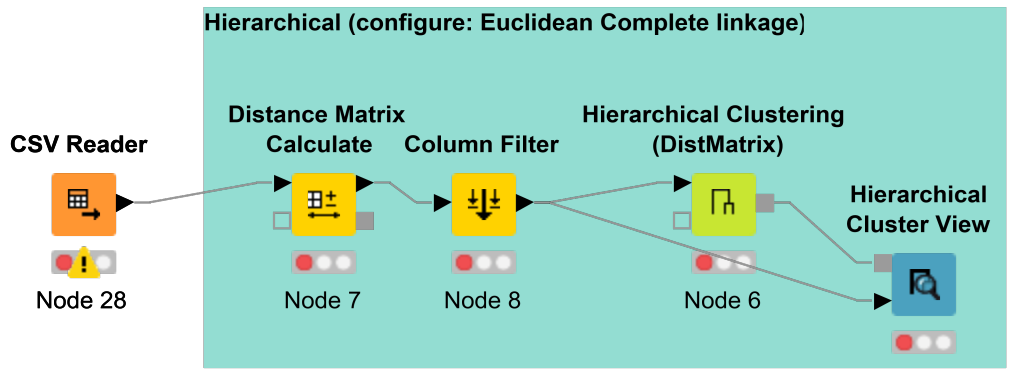
Problem one shows us that although we can cluster the sample and cut the tree, but we cannot decide which number is the best number of clusters.
Problem two shows us that we cannot decision which mini number of record per node is best choice when we build the decision tree model. In the section, we integrate R and Knime to compare the accuracy of four evaluation methods, which allows us to be familiar with different sampling and approach to solve the classification problem.

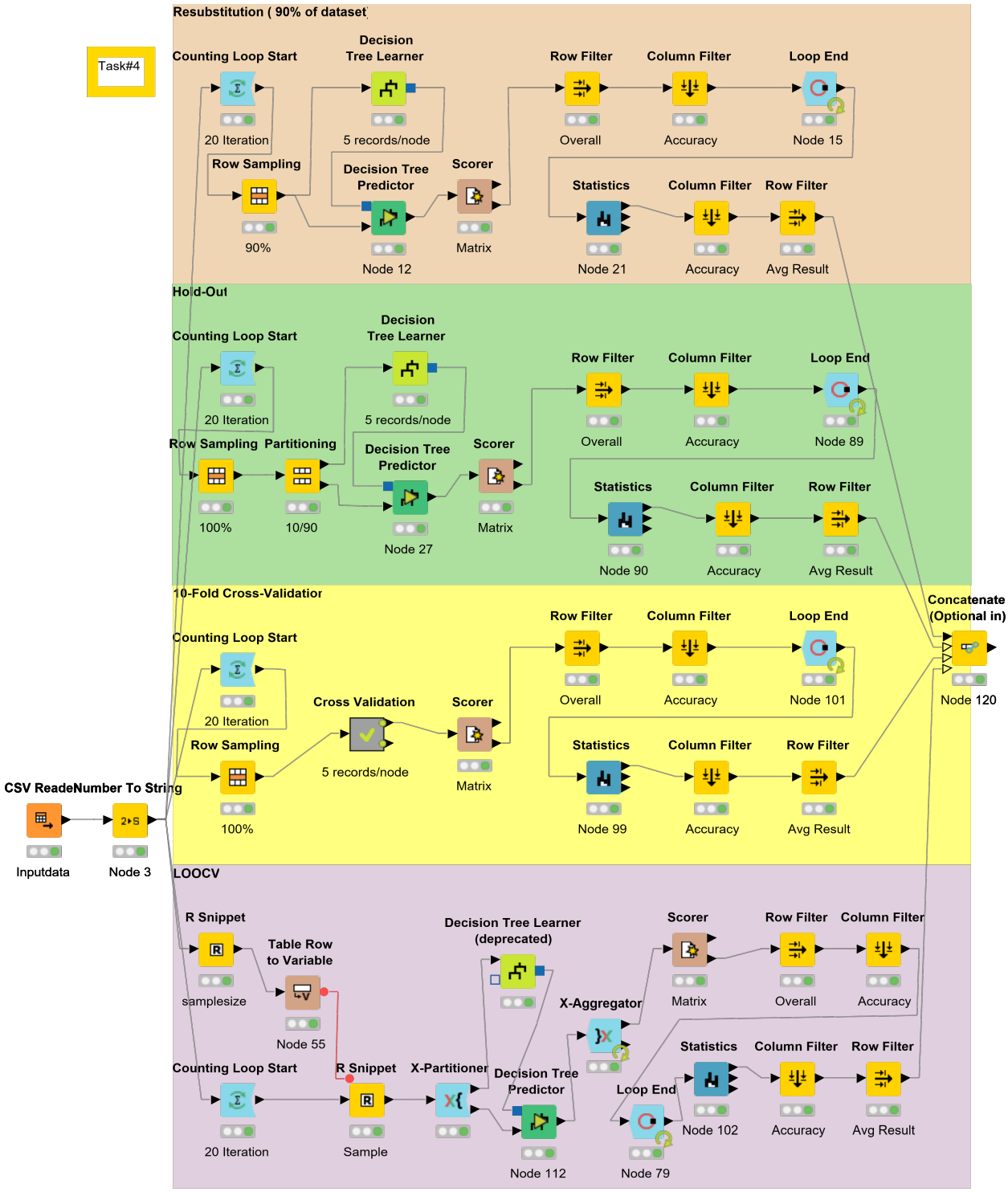
---

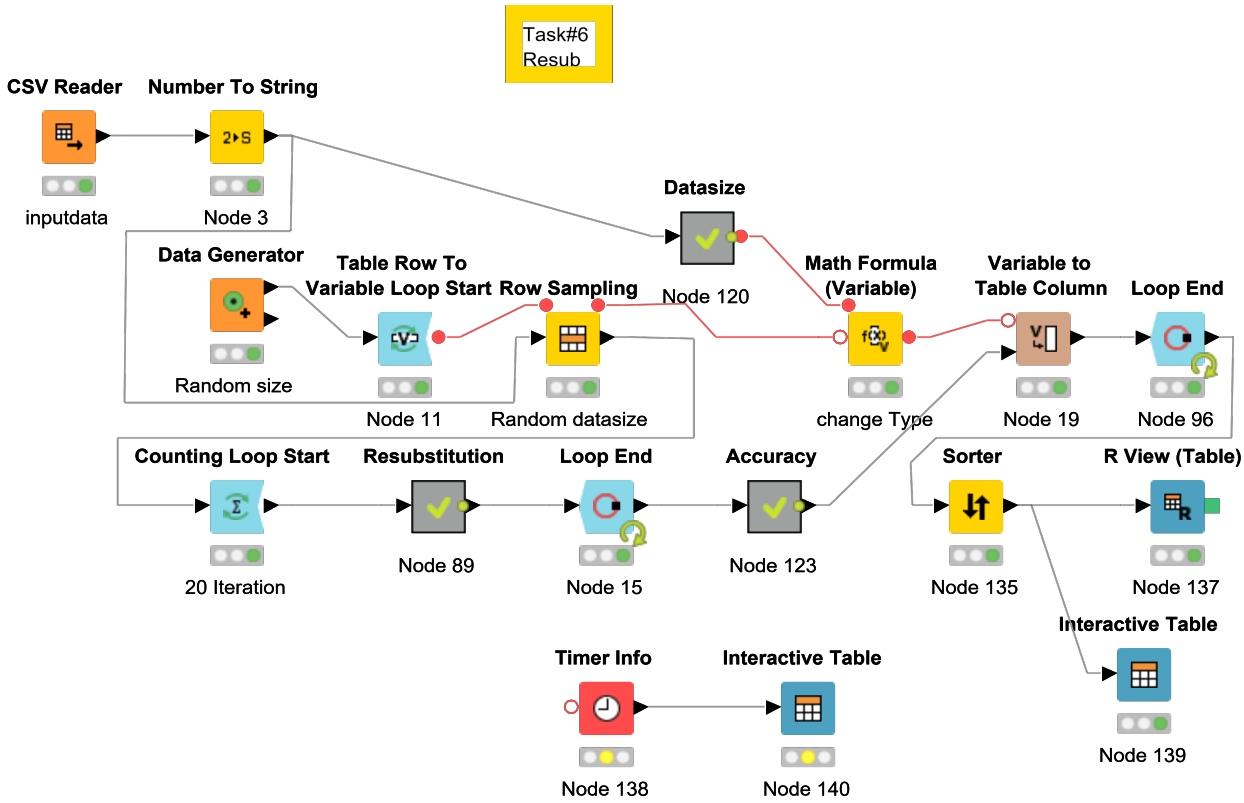
[i] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.152&rep=rep1&type=pdf
[ii] https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf

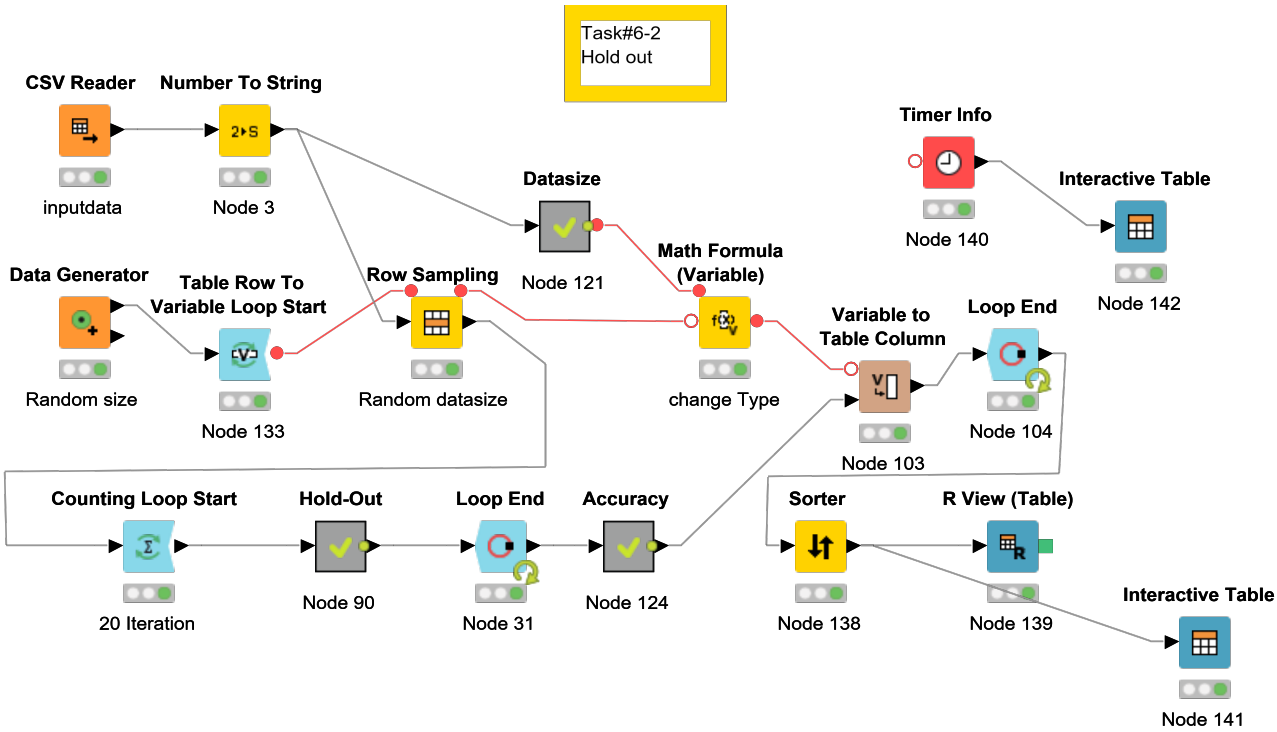

CSV Reader
Statistics
Column Filter
Interactive Table
Node 1
Overview
Statistics
Statistics Table

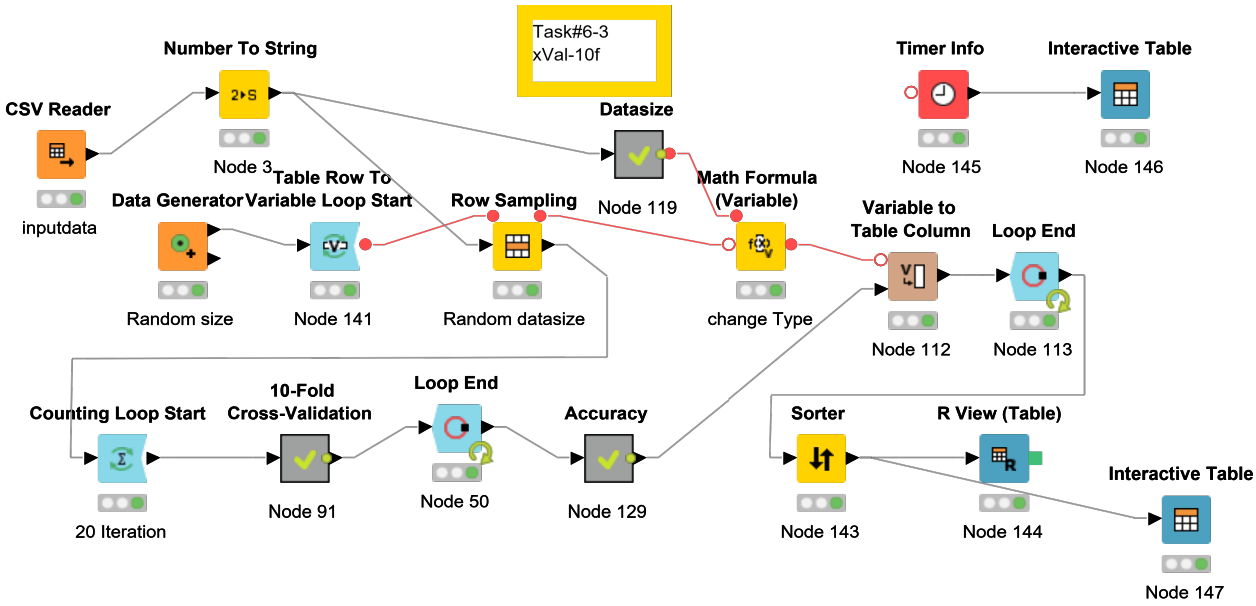

Method 1 Hierarchical (configure Euclidean Complete)
Hierarchical Clustering
Node 5

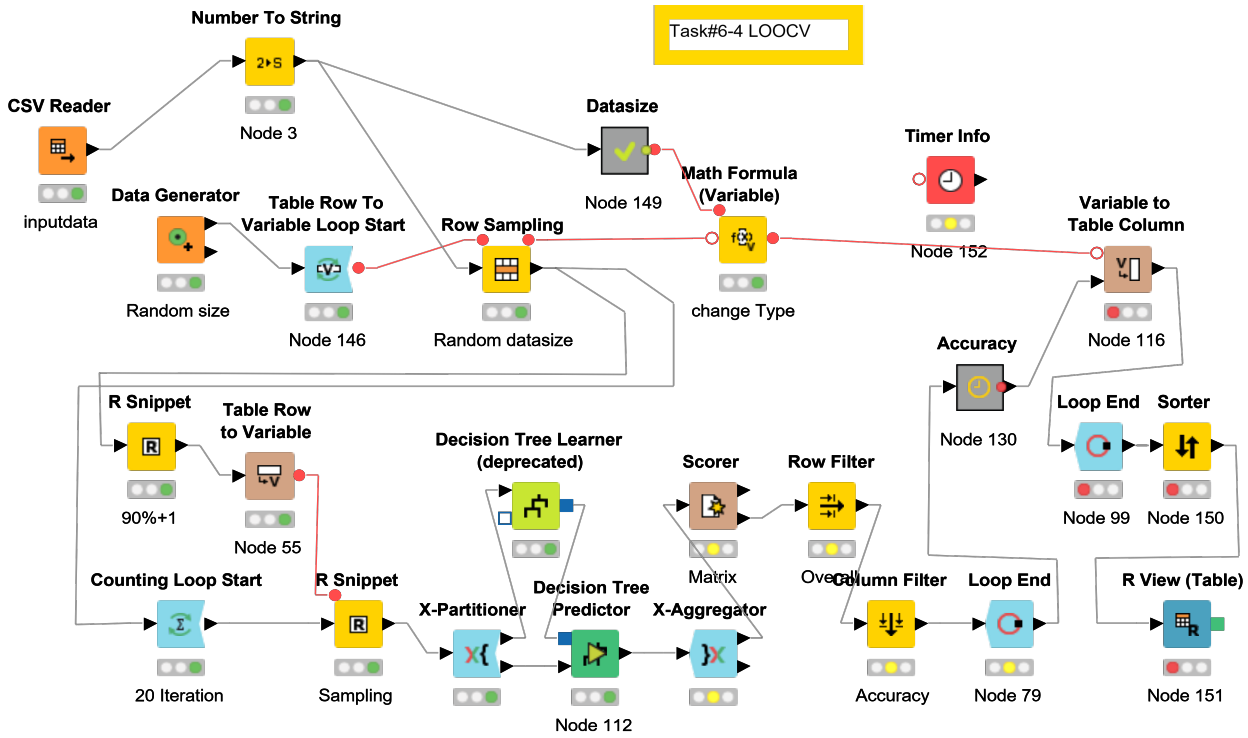

Hierarchical (configure: Euclidean Complete linkage)
CSV Reader
Distance Matrix Calculate
Column Filter
Hierarchical Clustering (DistMatrix)
Hierarchical Cluster View
Node 28
Node 7
Node 8
Node 6


Resubstitution (90% of dataset
Task#4
Counting Loop Start
20 Iteration
Row Sampling
90%
Decision Tree Learner
5 records/node
Decision Tree Predictor
Node 12
Scorer
Matrix
Row Filter
Overall
Column Filter
Accuracy
Loop End
Node 15
Statistics
Node 21
Column Filter
Accuracy
Row Filter
Avg Result
Hold-Out
Counting Loop Start
20 Iteration
Row Sampling
100%
Partitioning
10/90
Decision Tree Learner
5 records/node
Decision Tree Predictor
Node 27
Scorer
Matrix
Row Filter
Overall
Column Filter
Accuracy
Loop End
Node 89
Statistics
Node 90
Column Filter
Accuracy
Row Filter
Avg Result
10-Fold Cross-Validation
Counting Loop Start
20 Iteration
Row Sampling
100%
Cross Validation
5 records/node
Scorer
Matrix
Row Filter
Overall
Column Filter
Accuracy
Loop End
Node 101
Statistics
Node 99
Column Filter
Accuracy
Row Filter
Avg Result
Concatenate (Optional in)
Node 120
CSV ReadeNumber To String
Inputdata
Node 3
LOOCV
R Snippet
samplesize
Table Row to Variable
Node 55
Counting Loop Start
20 Iteration
R Snippet
Sample
X-Partitioner
Decision Tree Learner (deprecated)
Decision Tree Predictor
Node 112
X-Aggregator
Loop End
Node 79
Scorer
Matrix
Row Filter
Overall
Column Filter
Accuracy
Statistics
Node 102
Column Filter
Accuracy
Row Filter
Avg Result


Task#6
Resub
CSV Reader
Number To String
inputdata
Node 3
Datasize
Node 120
Data Generator
Table Row To
Variable Loop Start
Row Sampling
Math Formula
(Variable)
Variable to
Table Column
Loop End
Random size
Node 11
Random datasize
change Type
Node 19
Node 96
Counting Loop Start
Resubstitution
Loop End
Accuracy
Sorter
R View (Table)
20 Iteration
Node 89
Node 15
Node 123
Node 135
Node 137
Interactive Table
Timer Info
Interactive Table
Node 139
Node 138
Node 140


Task#6-2
Hold out
CSV Reader
inputdata
Number To String
Node 3
Datasize
Node 121
Timer Info
Node 140
Interactive Table
Node 142
Data Generator
Random size
Table Row To
Variable Loop Start
Node 133
Row Sampling
Random datasize
Math Formula
(Variable)
change Type
Variable to
Table Column
Node 103
Loop End
Node 104
Counting Loop Start
20 Iteration
Hold-Out
Node 90
Loop End
Node 31
Accuracy
Node 124
Sorter
Node 138
R View (Table)
Node 139
Interactive Table
Node 141


Task#6-3
xVal-10f
Number To String
CSV Reader
Node 3
inputdata
Datasize
Node 119
Timer Info
Interactive Table
Node 145
Node 146
Table Row To
Data Generator
Variable Loop Start
Row Sampling
Math Formula
(Variable)
Variable to
Table Column
Loop End
Random size
Node 141
Random datasize
change Type
Node 112
Node 113
Counting Loop Start
10-Fold
Cross-Validation
Loop End
Accuracy
Sorter
R View (Table)
Interactive Table
20 Iteration
Node 91
Node 50
Node 129
Node 143
Node 144
Node 147


Task#6-4 LOOCV
Number To String
Node 3
CSV Reader
inputdata
Datasize
Node 149
Math Formula (Variable)
change Type
Timer Info
Node 152
Variable to Table Column
Node 116
Data Generator
Random size
Table Row To Variable Loop Start
Node 146
Row Sampling
Random datasize
Accuracy
Node 130
Loop End
Node 99
Sorter
Node 150
R Snippet
90%+1
Table Row to Variable
Node 55
Decision Tree Learner (deprecated)
Scorer
Matrix
Row Filter
Counting Loop Start
20 Iteration
R Snippet
Sampling
X-Partitioner
Decision Tree Predictor
Node 112
X-Aggregator
Column Filter
Accuracy
Loop End
Node 79
R View (Table)
Node 151