

Classification Flow-C

Shaillay Kumar Dogra
Scientific Editor – QSAR World
editor@qsarworld.com

Notes:

1. This Jython script works in Sarchitect Designer version 2.3
2. Learn about Sarchitect Designer – <http://www.strandls.com/sarchitect/index.html>
3. Get Sarchitect – <http://www.strandls.com/sarchitect/freetrial.php>

The actual script follows this discussion. It is also accessible directly from the webpage in .py format.

Discussion:

The script provided here automates classification modeling. It is an implementation of the modeling workflow depicted in Figure 3. The following steps get executed in the script:

- 1) Top 100 descriptors are selected based on Kruskal-Wallis statistical test – a “Feature Ranking” report of all the descriptors is displayed as well as a “Ranked Subset” with top 100 descriptors gets created. (User can change the cutoff values in line #75 of the script.)
- 2) “Naïve Bayes” modeling algorithm is called on the descriptors in the “Ranked Subset” – an “Axis-Parallel Decision Tree” report is displayed in a table format. (User can change “Naïve Bayes” related parameters in line #118 of the script.)
- 3) Control goes back to the “Ranked Subset” and a “Forward Selection” wrapper with “Naïve Bayes” algorithm is used to select descriptors. The maximum number of descriptors that can get selected is set as 10. (User can change this in line #137 of the script). A node titled “Forward Selection” is created displaying various modeling parameters in a report and a scatter-plot titled “Feature vs. Accuracy”.

User can now view and select from the various model(s) and choose to “Train Model” on some appropriate one. Results for selected model(s) that were run are displayed under the “Classify” folder-node. This amongst other views contains the “Prediction Results” and “Confusion Matrix”. User can now view and finally save the appropriate model(s).

The loop-back, as shown in figure 3, can be made to start from some other point by declaring some node as “thisnode” (thisnode=node) in the script. For example, node at line# 63 can be declared as “thisnode” and “thisnode” at line#110 can be removed. This will have the effect of starting “Forward Selection” wrapper from the very first node containing all descriptors.



Flow C
Naive Bayes with the
Forward Selection
wrapper

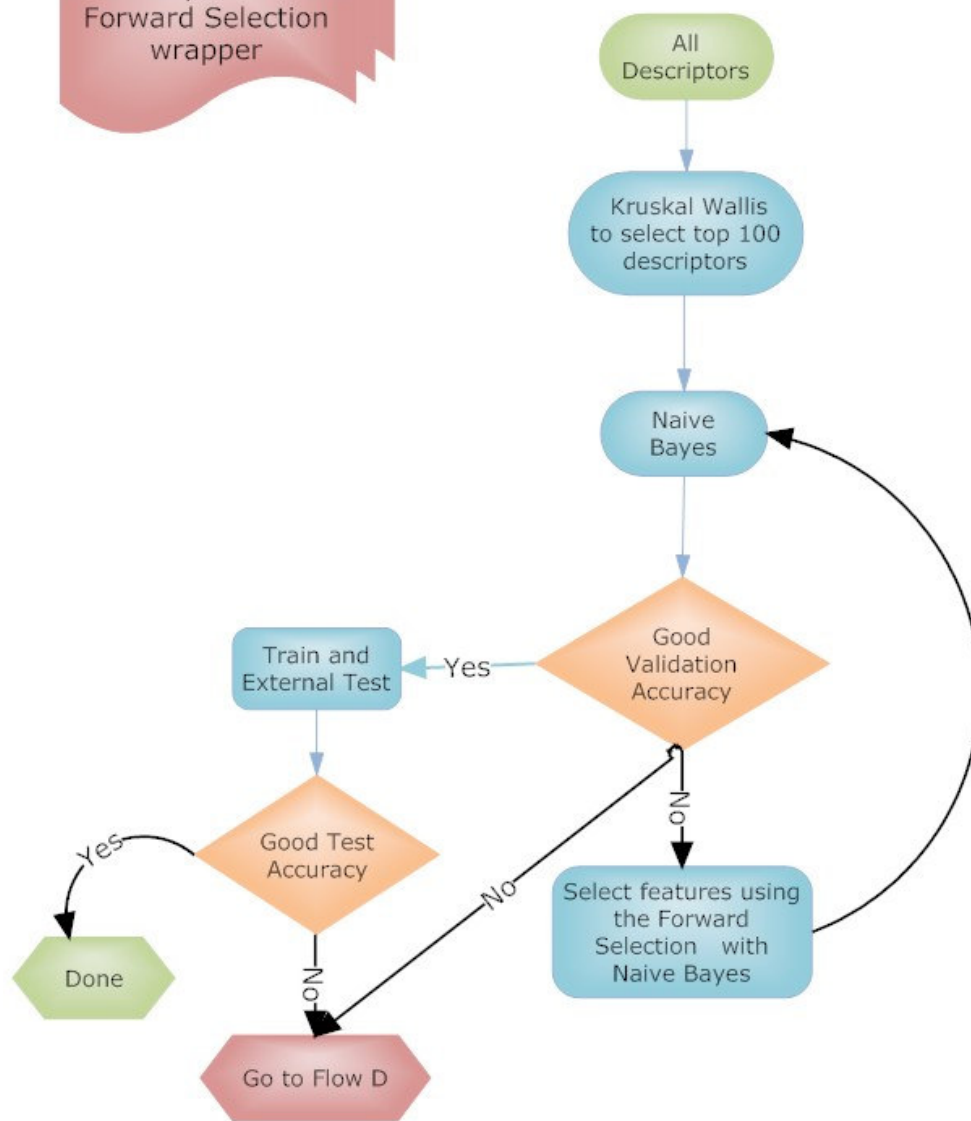


Figure 3

Cite this as:

Dogra, Shaillay K., "Script for automated Classification Flow-C" from QSARWorld – free online resource for QSAR modeling. <http://www.qsarworld.com/virtual-workshop.php>

```

##-----
##
##
## Sarchitect Designer 2.3 script
## implementing "Classification Flow-C"
##
##
##
## Shaillay Kumar Dogra
## editor@qsarworld.com
## August 07, 2007
##
##-----

import script
from script.dataset import *
from script.algorithm import *
from script.project import *
from script.view import *
from script.omega import createComponent, showDialog
from javax.swing import *
from com.strandgenomics.cube.dataset import *
import jarray
from math import *

##-----
## GET LIST OF CONTINUOUS, UNMARKED COLUMNS
def getColumnlist(dataset):
    ## Get columns, assumption: continuous and unmarked columns
    indices_continuous =
DatasetUtil.getContinuousColumnIndices(dataset)
    indices_nm_continuous =
script.project.removeMarkedColumns(dataset,indices_continuous)
    columnList = indices_nm_continuous
    #print columnList
    return columnList

##-----
##
def getIndexedIntArray(rowHeaderLabels,dataset):
    from com.strandgenomics.cube.framework.data import ArrayUtil,
DefaultIntArray
    size = rowHeaderLabels.getSize()
    array = DefaultIntArray(size)
    for i in range(size):
        colName = rowHeaderLabels.get(i)
        c = dataset.getColumn(colName)
        array.add(dataset.indexOf(c))
    return ArrayUtil.createIndexedIntArray(array)

##-----
##
def getStringArray(rowHeaderLabels):
    array = []
    from com.strandgenomics.cube.framework.data import DefaultIntArray

```

```

    size = rowHeaderLabels.getSize()
    for i in range(size):
        array.append(rowHeaderLabels.get(i))
    return array

##-----

node = script.project.getActiveDatasetNode()
dataset = script.project.getActiveDataset()

#
## This is what gets called from menu: Model->Select Model Descriptors-
>Kruskal Wallis
## Not using it in script as it requires user-intervention for input
#from script.chem.models.StatisticalCorrelation import createStatCorrUI
#createStatCorrUI()
#

## Kruskal-Wallis Correlation against endpoint to select top N
descriptors
result =
script.algorithm.kwallisFeatureSelection(test="kwallis",select="Based
on rank", rank=100).execute()

inputs = result.getInputs()
dataset = inputs["dataset"]
rankDataset = result["results"]
rowHeaderLabels = result["rowHeaderLabels"]

nameColumn = ColumnFactory.createStringColumn("Descriptor",
getStringArray(rowHeaderLabels))
columnList = []
columnList.append(nameColumn)
for i in range(rankDataset.getColumnCount()):
    columnList.append(rankDataset.getColumn(i))

columns = jarray.array(columnList, IColumn)
newDataset =
DatasetFactory.createDataset(rankDataset.getName(), columns)

node = script.project.getActiveDatasetNode()

from com.strandgenomics.cube.framework.selection import
MappedSelectionModel, DummySelectionModel
from com.strandgenomics.cube.framework.filter import DummyFilterModel
selModel =
MappedSelectionModel(node.getContext().getColumnSelectionModel(),
getIndexIntArray(rowHeaderLabels, dataset))

newnode = script.project.addFolderNode("Feature Ranking", node)
script.view.RankFeaturesView(node=newnode, dataset=newDataset,
title=newDataset.getName(), rowSelectionModel=selModel,
columnSelectionModel = DummySelectionModel.INSTANCE, filterModel =
DummyFilterModel(newDataset.getRowCount())).show()

```

```
pvalue = inputs['pvalue']
rank = inputs['rank']
select = inputs['select']
script.algorithm.SelectFeatures(node = node, featureselection =
newDataset, dataset = dataset, pvalue = pvalue, rank = rank, select =
select).execute(displayResult=1)

node = script.project.getActiveDatasetNode()
dataset = script.project.getActiveDataset()
thisnode = node

collist = getcolumnlist(dataset)
endpoint = DatasetUtil.getMarkedColumnIndices(dataset, "classlabel")
endpoint = ArrayUtil.createIndexedIntArray(endpoint)
endpoint = endpoint.get(0)

## Call Axis-Parallel Decision Tree
algo =
script.algorithm.NaiveBayesValidationX(classLabelColumn=endpoint, column
Indices=collist,
validationType="NFold", nFold=3, numRepeats=10)

algo.execute(interactive=0, displayResult=1, newThread=0, lockProject=0)

thisnode.setActive(1)
dataset = script.project.getActiveDataset()
collist = getcolumnlist(dataset)
endpoint = DatasetUtil.getMarkedColumnIndices(dataset, "classlabel")
endpoint = ArrayUtil.createIndexedIntArray(endpoint)
endpoint = endpoint.get(0)

#
## This is what gets called from menu: Model->Select Model Descriptors-
>Forward/Backward Selection
## Not using it in script as it requires user-intervention for input
#script.spring.featureselection.evaluator.runFSWiz()
#

target_size=10

fsAlgo = script.algorithm.ForwardSelection(dataset=dataset,
columnIndices=collist, classLabelColumn= endpoint,
targetSize=target_size, fitnessDef="Overall Accuracy",
targetAccuracy=100, accuracyType="Validation Accuracy",
fitnessEvaluationAlgorithm=script.algorithm.NaiveBayesValidationX(class
LabelColumn=endpoint, columnIndices=collist, validationType="NFold", nFold
=3, numRepeats=10))

result = fsAlgo.execute(interactive=0, displayResult=1, newThread=0)

##
```

END
##

End of Document