

## **Classification Flow-A**

Shaillay Kumar Dogra  
Scientific Editor – QSAR World  
[editor@qsarworld.com](mailto: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 1. The following steps get executed in the script:

- 1) Top 400 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 400 descriptors gets created. (User can change the cutoff values in line #67 of the script.)
- 2) “Decision Forest” modeling algorithm is called on the descriptors in the “Ranked Subset” – a “Decision Forest” report is displayed in a table format. User can now select the given model(s) and choose to “Train Model” on that. (User can change various “Decision Forest” related parameters in line #113 of the script.)

Results for selected model(s) that were run are displayed under the “Classify” folder-node. This amongst other views contains the “Prediction Results”, “Confusion Matrix”, and the “Model Metrics”. User can now view and save the appropriate model(s).



Flow A  
The first shot quick  
classification flow with  
Decision Forests

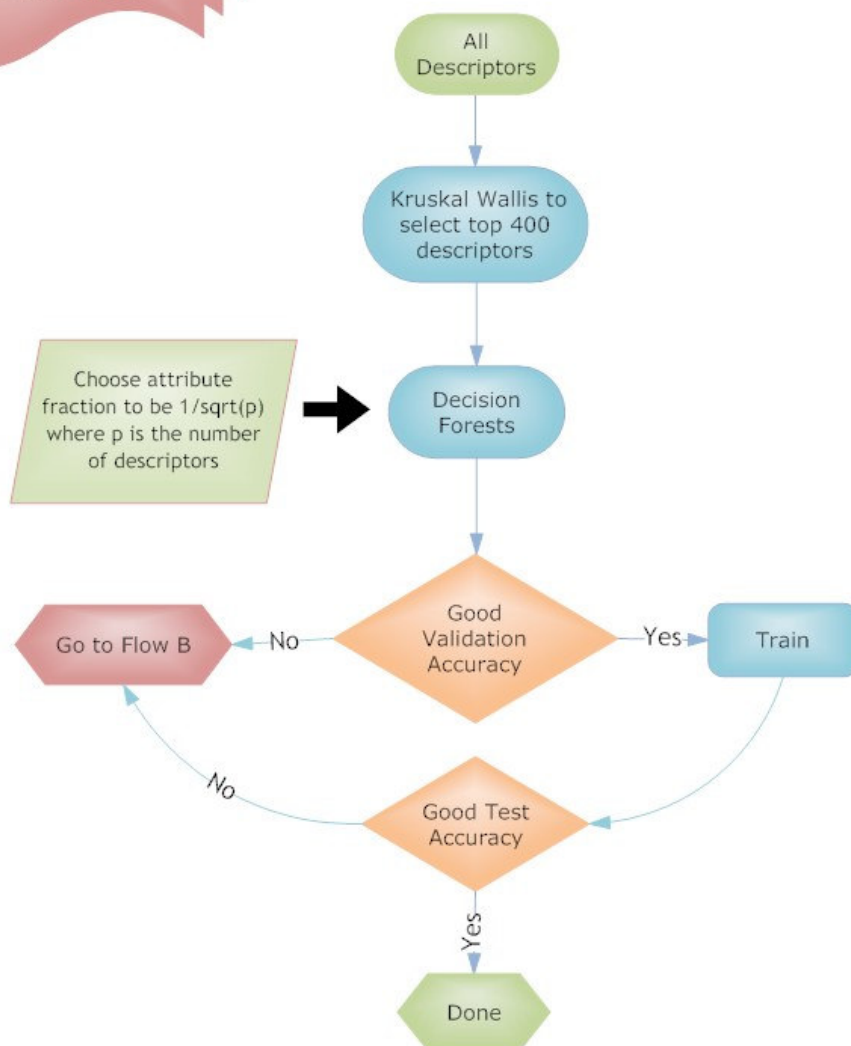


Figure 1

**Cite this as:**

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

---

```
##-----
##
##
## Sarchitect Designer 2.3 script
## implementing "Classification Flow-A"
##
##
##
## 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

##-----

dataset = script.project.getActiveDataset()

## Kruskal-Wallis Correlation against endpoint to select top N
descriptors
result =
script.algorithm.kwallisFeatureSelection(test="kwallis",select="Based
on rank", rank=400).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()

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

colcount = dataset.getColumnCount()
attributefraction = sqrt(float(colcount))/colcount

#attributefraction=0.4

algo =
script.algorithm.AxisParallelDTBaggingX(classLabelColumn=endpoint, columnIndices=collist, goodnessFunction="InformationGain", leafImpurity="0.001", leafImpurityType="Local", pruningMethod="PessimisticError", fraction="0.6", numModels="50", fractionAtEachNode=attributefraction)

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

##
## END
##
```

---

End of Document