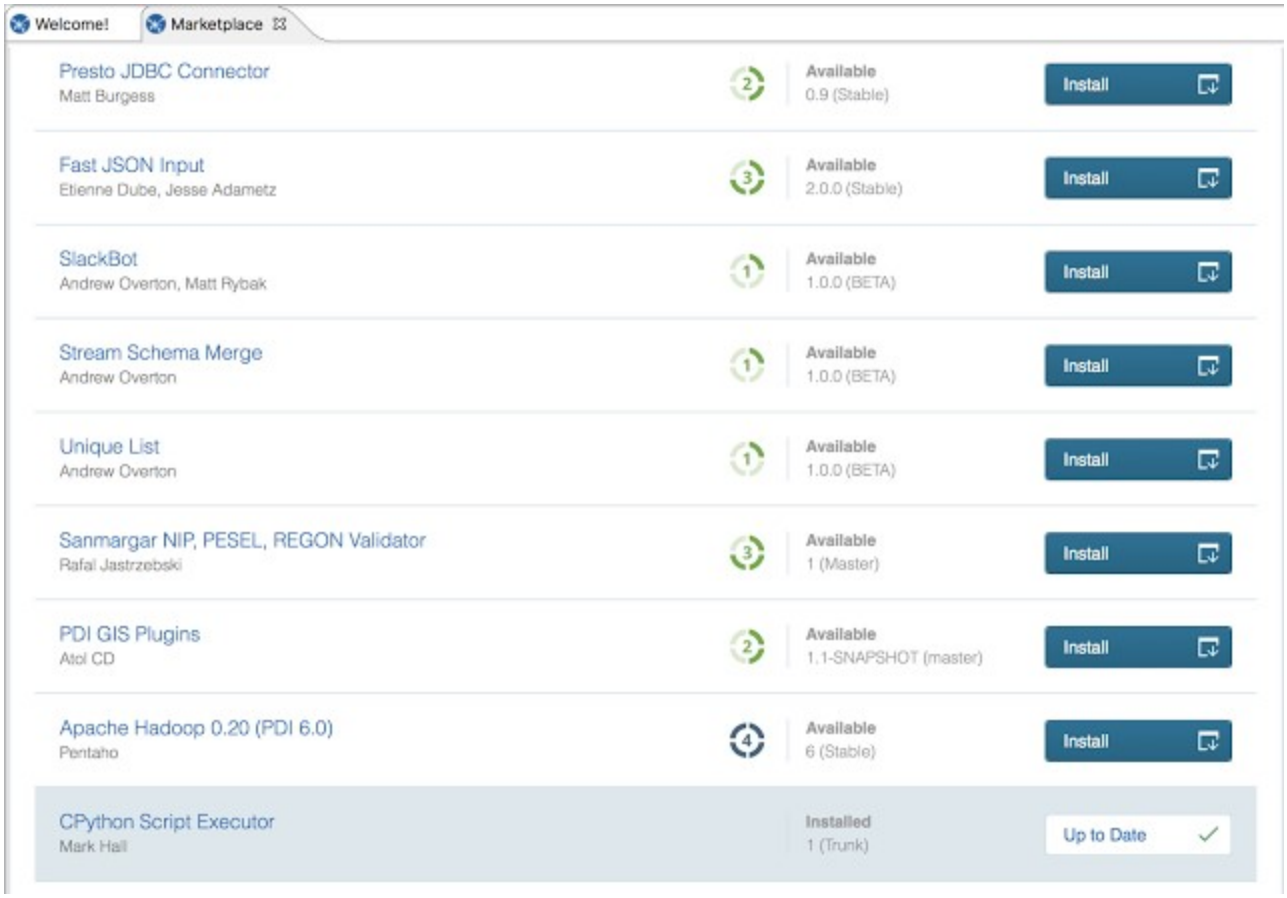


CPython Scripting in Pentaho Data Integration

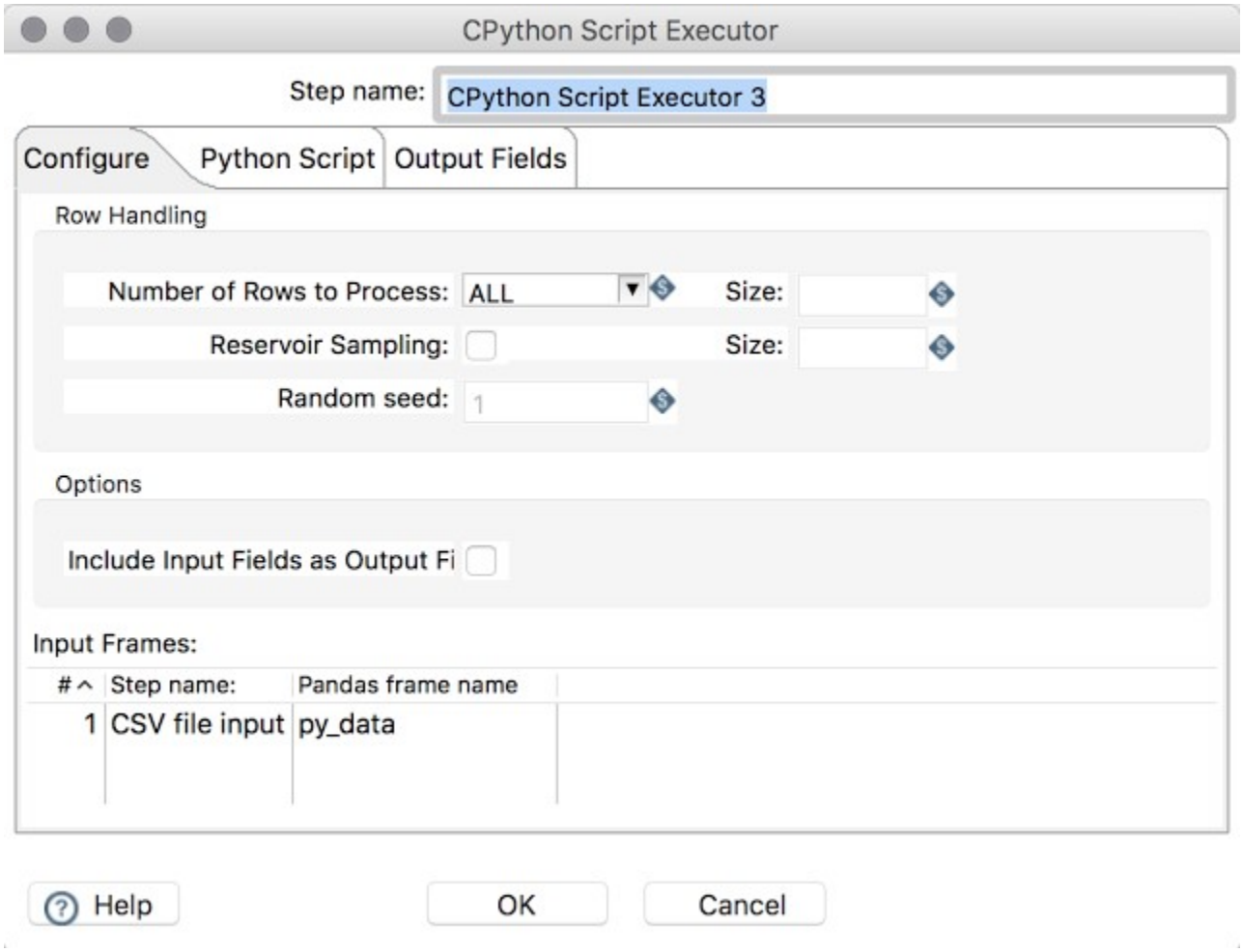
markahall.blogspot.com.br |
Using the approach developed for integrating Python into Weka, Pentaho Data Integration (PDI) now has a new step that can be used to leverage the Python programming language (and its extensive package-based support for scientific computing) as part of a data integration pipeline. The step has been released to the community from Pentaho Labs and can be installed directly from PDI via the marketplace.

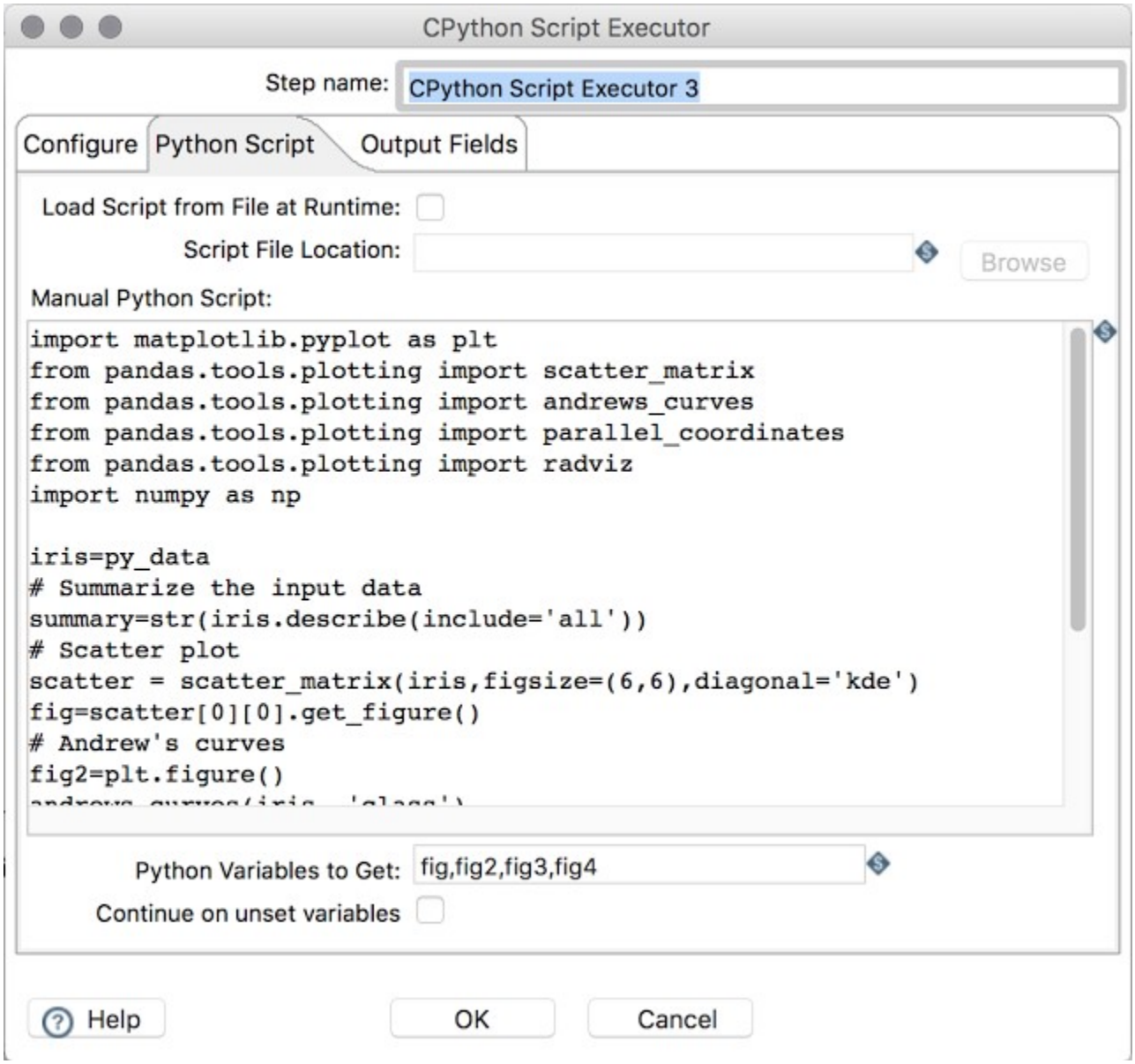
Python is becoming a serious contender to R when it comes to programming language choice for data scientists. In fact, many folks are leveraging the strengths of both languages when developing solutions. With that in mind, it is clear that data scientists and predictive application developers can boost productivity by leveraging the PDI + Python combo. As we all know, data preparation consumes the bulk of time in a typical predictive project. That data prep can typically be achieved more quickly in PDI, compared to developing code from scratch, thanks to its intuitive graphical development environment and extensive library of connectors and processing steps. Instead of having to write (and rewrite) code to connect to source systems (such as relational databases, NoSQL databases, Hadoop filesystems and so forth), and to join/filter/blend data etc., PDI allows the developer to focus their coding efforts on the cool data science-oriented algorithms.

CPython Script Executor

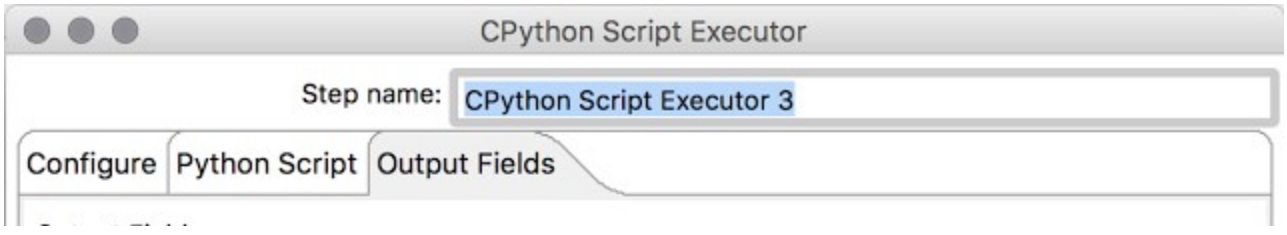
As the name suggests, the new step uses the C implementation of the Python programming language. While there are JVM-based solutions available - such as Jython - that allow a more tightly integrated experience when executing in the JVM, these do not facilitate the use of many high-powered Python libraries for scientific computing, due to the fact that such libraries include highly optimised components that are written in C or Fortran. In order to gain access to such libraries, the PDI step launches, and communicates with, a micro-service running in the C Python environment. Communication is done over plain sockets and messages are stored in JSON structures. Datasets are transmitted as CSV and the very fast routines for reading and writing CSV from the pandas Python package are leveraged.

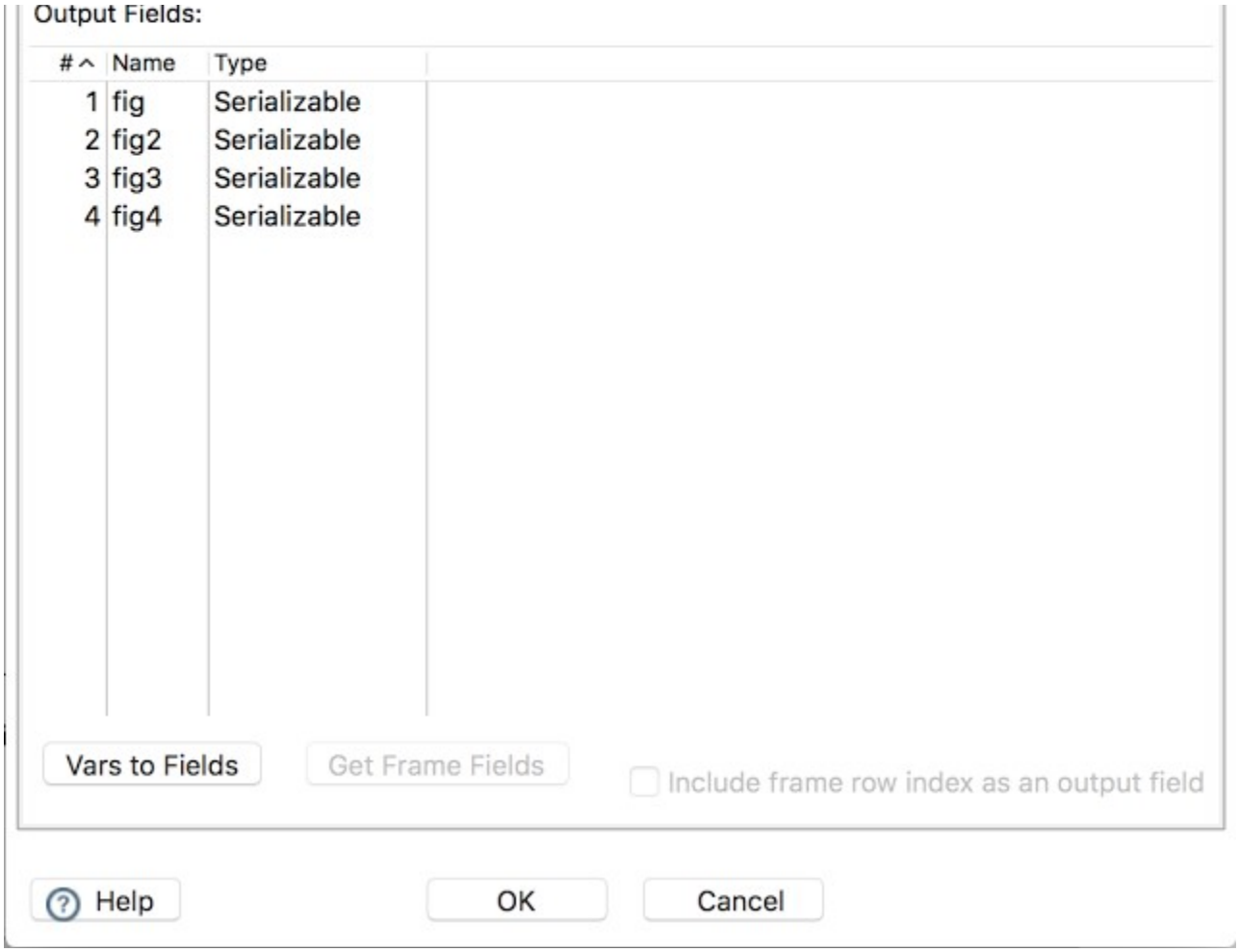
The step itself offers maximum flexibility when it comes to dealing with data. It can act as a start point/data source in PDI (thus allowing the developer the freedom to source data directly via their Python code if so desired), or it can accept data from an upstream step and push it into the Python environment. In the latter case, the user can opt to send all incoming rows to Python in one hit, send fixed sized batches of rows, or send rows one-at-a-time. In any of these cases the data sent is considered a dataset, gets stored in a user-specified variable in Python, and the user's Python script is invoked. In the "all data" case, there is also the option to apply reservoir sampling to down-sample to a fixed size before sending the data to Python. The pandas DataFrame is used as the data structure for datasets transferred into Python.





A python script can be specified via the built-in editor, or loaded from a file dynamically at runtime. There are two scenarios for getting output from the Python environment to pass on to downstream PDI steps for further processing. The first (primary) scenario is when there is a single variable to retrieve from Python and it is a pandas DataFrame. In this case, the columns of the data frame become output fields from the step. In the second scenario, one or more non-data frame variables may be specified. In this case, their values are assumed to be textual (or can be represented as text) or contain image data (in which case they are retrieved from Python as binary PNG data). Each variable is output in a separate PDI field.





Requirements

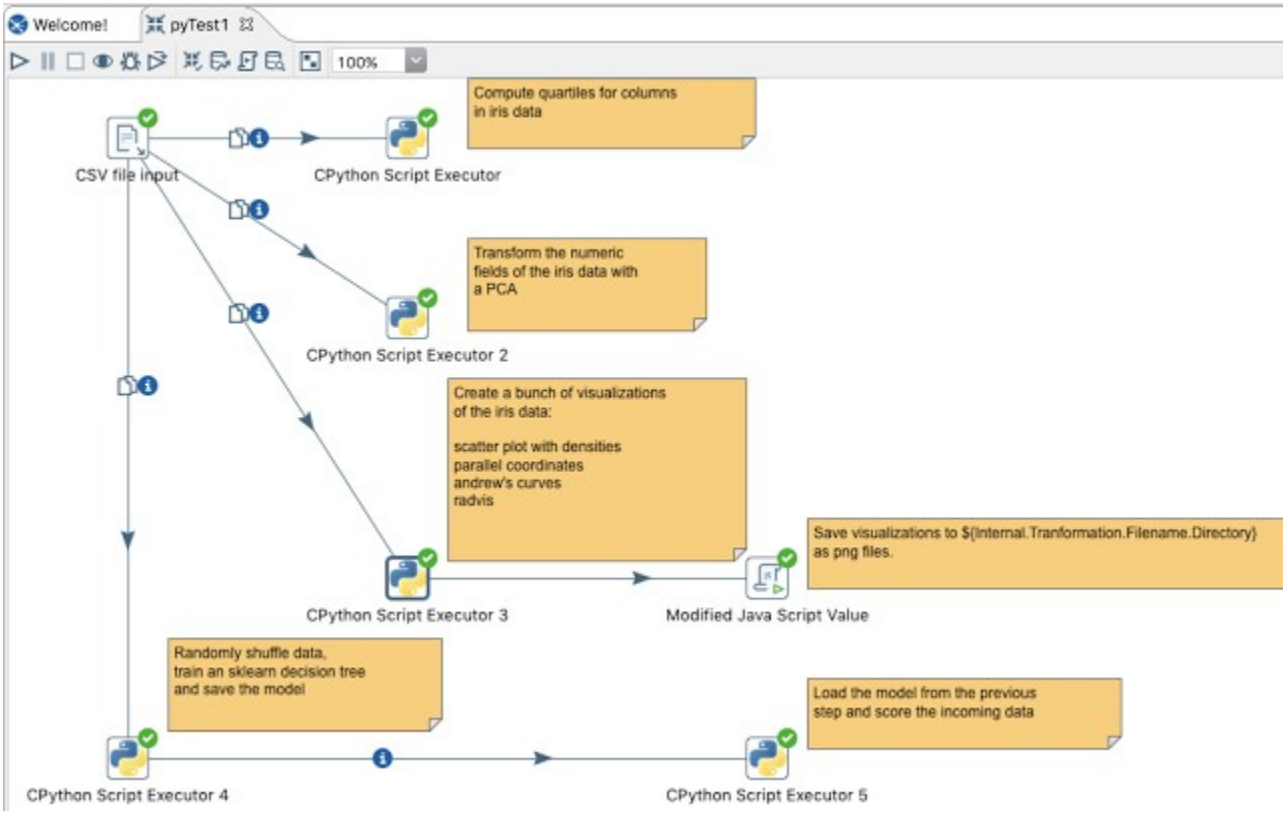
The CPython Script Executor step will work with PDI >= 5.0. Of course, it requires Python to be installed and the python executable to be in your PATH environment variable. The step has been tested with Python 2.7 and 3.x and, at a minimum, needs the pandas, matplotlib and numpy libraries to be installed. For Windows users in particular, I'd recommend installing the excellent Anaconda python distribution. This includes the entire SciPy stack (including pandas and scikit-learn) along with lots of other libraries.

Example

The example transformation shown in the following screenshot can be obtained from here.

The example uses Fisher's classic iris data. The first python step (at the top) simply computes some quartiles for the numeric columns in the iris data. This is output from the step as a pandas DataFrame, where each row corresponds to one of the quartiles computed (25th, 50th and 75th), and each column holds the value for one of the numeric fields in the iris data. The second python step from the top uses the scikit-learn decomposition routine to compute a principal components analysis on the iris data and then transforms the iris data into the PCA space, which is then the output of the step. The third python

step from the top uses the matplotlib library and plotting routines from the pandas library to compute some visualisations of the iris data (scatter plot matrix, Andrew's curves, parallel coordinates and radviz). These are then extracted as binary PNG data from the python environment and saved to files in the same directory as the transformation was loaded from. The two python steps at the bottom of the transformation learn a decision tree model and then use that model to score the iris data respectively. The model is saved (from the python environment) to the directory that the transformation was loaded from.



Conclusion

The new PDI CPython Script Executor step opens up the power of Python to the PDI developer and data scientist. It joins the R Script Executor and Weka machine learning steps in PDI as part of an expanding array of advanced statistical and predictive tools that can be leveraged within data integration processes.