**Issues in using PIMS for scikit-learn**

This document summarizes the issues detected while using PIMS for importing and using scikit-learn library in SCILAB.

The issues considered in this document occured while trying [kernel ridge regression vs SVR](http://scikit-learn.org/stable/auto_examples/plot_kernel_ridge_regression.html#sphx-glr-auto-examples-plot-kernel-ridge-regression-py) from the scikit-learn documentation using PIMS.

1. Importing libraries  
   When importing python libraries such as numpy or scikit-learn, use of pyImport is justified. But once a library has been imported every time the entire path/structure of the library needs to be specified to reach a particular module.

Eg. python -- >> *import matplotlib.pyplot*

PIMS -- >> *pyImport matplotlib.pyplot*

!--error 999

pyImport: An error occured: Cannot load the module matplotlib.pyplot

Python interpreter threw an exception:

Traceback:

File C:\Python27\lib\site-packages\matplotlib\pyplot.py, line 115, in <module>

File C:\Python27\lib\site-packages\matplotlib\backends\\_\_init\_\_.py, line 32, in pylab\_setup

File C:\Python27\lib\site-packages\matplotlib\backends\backend\_tkagg.py, line 6, in <module>

File C:\Python27\lib\site-packages\six.py, line 203, in load\_module

File C:\Python27\lib\site-packages\six.py, line 115, in \_resolve

File C:\Python27\lib\site-packages\six.py, line 82, in \_import\_module

File C:\Python27\Lib\lib-tk\Tkinter.py, line 38, in <module>

File C:\Python27\Lib\lib-tk\FixTk.py, line 65, in <module>

exceptions.ImportError: DLL load failed: The specified module could not be found.

One possibility is that matplotlib has a different structure in terms of calling modules, but lets user access them through standard python import syntax *import library.module\_name*

Or that the user needs to specify the entire path to reach the actual module through PIMS.

Need to specify full name of library, if not named as a separate object.   
Eg.

*pyImport numpy;*

*numpy.array([1 2 3 4 5 5]);*

**Alternative:**

*pyImport numpy;*

*np = numpy;*

*np.array([1 2 3 4 5 5]);*

1. Calling functions from libraries  
   Once a library has been imported, methods cannot be imported for direct usage using standard python syntax  
   *method\_name(parameters)*Instead number of steps increased to  
   *pyImport library.module\_name  
   xyz = library.module\_name;  
   xyz.method\_name(parameters);*For every function/method call from an imported library, same syntax needs to be used which is an overhead.

1. Python object created in SCILAB using PIMS cannot be visualized  
   Any python object is created using PIMS it is visible in the variable explorer but only as a Mlist type, which cannot be visualized by the user for inspection.  
     
   When solving any machine learning problem, visualization of datasets once pre-processed, or for data-manipulation/editing is very necessary. But this is currently not possible with using PIMS.
2. Help documentation or python library methods  
   Only if a user has access to working internet and the library documentation (eg scikit-learn), he/she can work with PIMS in Scilab, since currently the user is expected to know all required parameters and contexts in which a method can be used.  
     
   Furthermore on using pyGetMethods(library\_name) in Scilab we don’t always get a list of all available methods in the library, just a list of those which are available immediately inside the first level in which library is organized.  
     
   Eg.   
   pyImport sklearn  
   pyGetMethods(sklearn)  
     
   Output :

!clone !

! !

!setup\_module !  
  
If train\_test\_split method needs to be called, we need to specify as below

*xyz = sklearn.setup\_module.cross\_validation.train\_test\_split(parameters\_required)*

1. Using builtin functions   
   like append and ravel on an expression using expression.method\_name( )  
   Since when using python in scilab, we are importing the builtin module as py, it is required to pass the expression by prefixing py to every builtin function.   
   py.ravel(expression)

Usage of [ ] for specifying list of values to be taken   
invokes error since scilab also uses [ ] for creating and manipulating matrices.   
  
Eg.  
*svr = ms.GridSearchCV(svm.SVR(kernel='rbf', gamma=0.1),cv=5,param\_grid={"C": [1e0, 1e1, 1e2, 1e3], "gamma": np.logspace(-2, 2, 5)});*!--error 43

Not implemented in scilab...

at line 7 of function %c\_b\_s called by :

, cv=5,param\_grid={"C":[1e0, 1e1, 1e2, 1e3], "gamma": np.logspace(-2, 2, 5)})

at line 30 of exec file called by :

exec('F:\Internship\SCILAB ML\scikit-learn scilab\kernel\_ridge\_svr.sce', -1)

[*http://scikit-learn.org/stable/auto\_examples/plot\_kernel\_ridge\_regression.html#sphx-glr-auto-examples-plot-kernel-ridge-regression-py*](http://scikit-learn.org/stable/auto_examples/plot_kernel_ridge_regression.html#sphx-glr-auto-examples-plot-kernel-ridge-regression-py)

*Error logs for* ***y[::5] += 3 \* (0.5 - rng.rand(X.shape[0]/5))*** *:*

*sys:1: VisibleDeprecationWarning: using a non-integer number instead of an integer will result in an error in the future*

*y[::5] += 3 \* (0.5 - rng.rand(X.shape[0]/5));*

*!--error 276*

*Missing operator, comma, or semicolon.*

*at line 25 of exec file called by :*

*exec('F:\Internship\SCILAB ML\scikit-learn scilab\kernel\_ridge\_svr.sce', -1)*

*y(::5) += 3 \* (0.5 - rng.rand(X.shape[0]/5));*

*!--error 2*

*Invalid factor.*

*at line 25 of exec file called by :*

*exec('F:\Internship\SCILAB ML\scikit-learn scilab\kernel\_ridge\_svr.sce', -1)*

Notable points after detailed discussion with Simon Marchetto on 14th June, 2017 :

1. Few of the issues mentioned above can be solved right now, but it doesn’t promise a fully working machine learning implementation using scikit-learn
2. Issues related to use of default python syntaxes like those for creating lists and dictionaries, can be reduced. However many other default syntaxes and builtin methods will take time to be resolved.
3. Requirement of help documentation for machine learning functions, through the common scilab help interface possible in a later release
4. Visualization of numpy arrays, or python lists and other data types is essential for any user trying to use PIMS in Scilab for machine learning. We can start by numpy arrays and gradually include all other data types over time.
5. Need to discuss possible approaches for building the machine learning toolbox in Scilab, in a joint exchange with Philippe Saade, Simon Marchetto, Yann Debray, Mandar Deshpande and project mentors