SAS & Open Source

How can we leverage both?

SAS and Open Source

What are the integration points?

- The Base SAS® Java Object
  - Call Open Sources from within SAS or Call SAS from within Open Source
  - Supported since 9.1.3
- SAS/IML Studio
  - Client-side integration with R was released in July 2009
  - Delivered with SAS/IML for SAS 9.2
- SAS/IML
  - Server-side integration with R delivered with IML 9.22 in November 2010
  - Extends support to Windows and Linux server environments
- Model Manager
  - Administrative tool for managing and monitoring predictive models
  - Support for models created with R was delivered with SAS 9.3 in July 2011
- Enterprise Miner (in SAS 9.4)
  - Open Source Node that enables users to submit R code as part of an EM process flow in December 2013
- SAS Viya
  - New architecture released October 2016
- Jupyter Notebook
  - SAS 9
    - Initial release on Linux September 2016
    - Kernel for Windows and Mainframe released March 2017
  - SAS Viya Oct 2016

Integration using base sas Java object

Base SAS java object

overview

- First introduced with SAS 9.1.3
- Provides a mechanism for instantiating Java objects and calling their methods within a DATA step
- Sample provides Java class (SASJavaExec) – class for executing any script with necessary (Python, R or any other) command-line arguments
- Data transfer using CSV files

Sasjavaexec class

- Available in SAS_Base_OpenSrcIntegration.zip
- Provides a universal mechanism for calling executables

- Constructor is overloaded with following arguments:
  - String execCommand, String script
  - String execCommand, String script, String inputString

- Has three public methods
  - executeProcess(): return int
  - executeProcess(double timeoutMin): return int
  - setWorkingDirectory: return void

Sasjavaexec class

- The executeProcess method creates a new Process using constructor arguments; redirects STDERR and STDOUT to SAS log, and begins executing the process
- When error occurs, process execution is halted with an exit code from the script/process
- If process completes without any errors, the return code is 0

python script example

Step 1: setup

- Pre-requisites:
  - SAS 9.1.3+
  - Python (2.7)
  - Oracle JDK (1.7.0_25)

- Download and unzip SAS_Base_OpenSrcIntegration.zip available at https://github.com/sassoftware/enlighten-integration
  - Training and test data sets: digitsdata_17_train.csv and digitsdata_17_test.csv
  - A Java class: SASJavaExec.java
  - A Python script and an R script: digitsdata_svm.py and digitsdata_svm.R
  - A SAS program: main_caller.sas

python script example

Step 2: COMPILE & specify Java classpath

- Compile provided Java classes
  - `javac src/dev/* -d bin`

- Set Java CLASSPATH to specify SAS the location of compiled Java classes
  - File C:\Program Files\SASHome\SASFoundation\9.4\nls\en\sasv9.cfg, add line
    - `-SET CLASSPATH "C:\SGF2015\OpenSrcIntegration\bin"`
  - Invoking SAS with appropriate arguments
    - `sas.exe -SET CLASSPATH "C:\SGF2015\OpenSrcIntegration\bin"`

python script example

Step 3: RUN SAS CODE (call using java object)

```
*** WORKING DIRECTORY      (----- USER UPDATE NEEDED -----);
%let WORK_DIR = C:\SGF2015\OpenSrcIntegration;
*** SYSTEM PYTHON LOCATION (----- USER UPDATE NEEDED -----);
%let PYTHON_EXEC_COMMAND = C:\Anaconda\python.exe;
/*** Part I: Python ***/
data _null_;
   length rtn_val 8;
   *** Python program takes working directory as first argument;
   python_pgm = "&WORK_DIR.\digitsdata_svm.py";
   python_arg1 = "&WORK_DIR";
   python_call = cat('"', trim(python_pgm), '" "', trim(python_arg1),'"');

   declare javaobj j("dev.SASJavaExec", "&PYTHON_EXEC_COMMAND",
                     python_call);
   j.callIntMethod("executeProcess", rtn_val);
run;
```

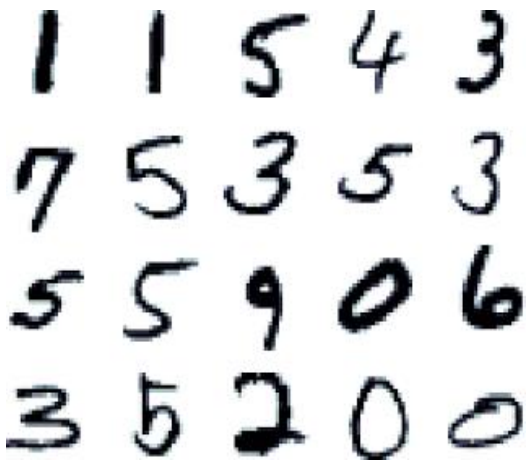
Python script example

Setup 4: check sas log

```
Apr 07, 2015 3:54:47 PM dev.SASJavaExec executeProcess
INFO: Executing [C:\Anaconda\python.exe,
C:\SGF2015\OpenSrcIntegration\digitsdata_svm.py, C:\SGF2015\OpenSrcIntegration] ...
Apr 07, 2015 3:54:47 PM dev.SASJavaExec executeProcess
INFO: Starting external process ...
Apr 07, 2015 3:54:47 PM dev.SASJavaExec executeProcess
INFO: External process exit value 0.
NOTE: DATA statement used (Total process time):
      real time           0.73 seconds
      cpu time            0.01 seconds
```

Output 1. Output from SAS Log after Executing the Python Script

example data



- Subset of MNIST database: labeled handwritten digit images of 1 and 7
- Has 784 predictor columns containing associated pixel intensity values on grey scale
- Includes one target column representing actual digit
- Train and test data are balanced with 200 observations (each) in CSV files

python script

Digitsdata_svm.py

```
import numpy as np
import sys, os
from sklearn import svm

# Load data
train = np.genfromtxt(os.path.join(sys.argv[1], "digitsdata_17_train.csv"),\
                      delimiter=",", skip_header=1)
test  = np.genfromtxt(os.path.join(sys.argv[1], "digitsdata_17_test.csv"),\
                      delimiter=",", skip_header=1)
X = train[:, 1:]
y = train[:, 0]
Xtest = test[:, 1:]

# Fit SVM
clf = svm.LinearSVC()
clf.fit(X, y)
pred = clf.predict(Xtest)
pred = pred.reshape(200, 1)

# Export output
np.savetxt(os.path.join(sys.argv[1], 'predict_test_python.csv'), pred, fmt='%f')
```

python script

Digitsdata_svm.py

```
import numpy as np
import sys, os
from sklearn import svm

# Load data
train = np.genfromtxt(os.path.join(sys.argv[1], "digitsdata_17_train.csv"),\
                      delimiter=",", skip_header=1)
test  = np.genfromtxt(os.path.join(sys.argv[1], "digitsdata_17_test.csv"),\
                      delimiter=",", skip_header=1)
X = train[:, 1:]
y = train[:, 0]
Xtest = test[:, 1:]

# Fit SVM
clf = svm.LinearSVC()
clf.fit(X, y)
pred = clf.predict(Xtest)
pred = pred.reshape(200, 1)

# Export output
np.savetxt(os.path.join(sys.argv[1], 'predict_test_python.csv'), pred, fmt='%f')
```

Read train and test data from CSV files

Execute python script

Digitsdata_svm.py

```
import numpy as np
import sys, os
from sklearn import svm

# Load data
train = np.genfromtxt(os.path.join(sys.argv[1], "digitsdata_17_train.csv"),\
                      delimiter=",", skip_header=1)
test  = np.genfromtxt(os.path.join(sys.argv[1], "digitsdata_17_test.csv"),\
                      delimiter=",", skip_header=1)
X = train[:, 1:]
y = train[:, 0]
Xtest = test[:, 1:]

# Fit SVM
clf = svm.LinearSVC()
clf.fit(X, y)
pred = clf.predict(Xtest)
pred = pred.reshape(200, 1)

# Export output
np.savetxt(os.path.join(sys.argv[1], 'predict_test_python.csv'), pred, fmt='%f')
```

Model train data using SVM and score test

python script

Digitsdata_svm.py

```
import numpy as np
import sys, os
from sklearn import svm

# Load data
train = np.genfromtxt(os.path.join(sys.argv[1], "digitsdata_17_train.csv"),\
                      delimiter=",", skip_header=1)
test  = np.genfromtxt(os.path.join(sys.argv[1], "digitsdata_17_test.csv"),\
                      delimiter=",", skip_header=1)
X = train[:, 1:]
y = train[:, 0]
Xtest = test[:, 1:]

# Fit SVM
clf = svm.LinearSVC()
clf.fit(X, y)
pred = clf.predict(Xtest)
pred = pred.reshape(200, 1)

# Export output
np.savetxt(os.path.join(sys.argv[1], 'predict_test_python.csv'), pred, fmt='%f')
```

Export predictions
to CSV file

Post modeling tasks

Run sas code

```
proc import out = predict_py
            datafile = "&WORK_DIR.\predict_test_python.csv"
            dbms = csv
            replace;
  getnames = no;
run;

data cmp_py;
  set digitsdata_17_test (keep=label);
  set predict_py (rename=(var1=pred_py));
run;

proc freq data = cmp_py;
  table label*pred_py / norow nocol;
run;
```

Table of label by pred_py

label pred_py

Frequency Percent	1	7	Total
1	100 50.00	0 0.00	100 50.00
7	2 1.00	98 49.00	100 50.00
Total	102 51.00	98 49.00	200 100.00

Integration Using SAS/IML® & IML Studio

SAS/IML®

WHAT IS IT?

A powerful, interactive matrix programming language for a wide range of applications – with integration to R

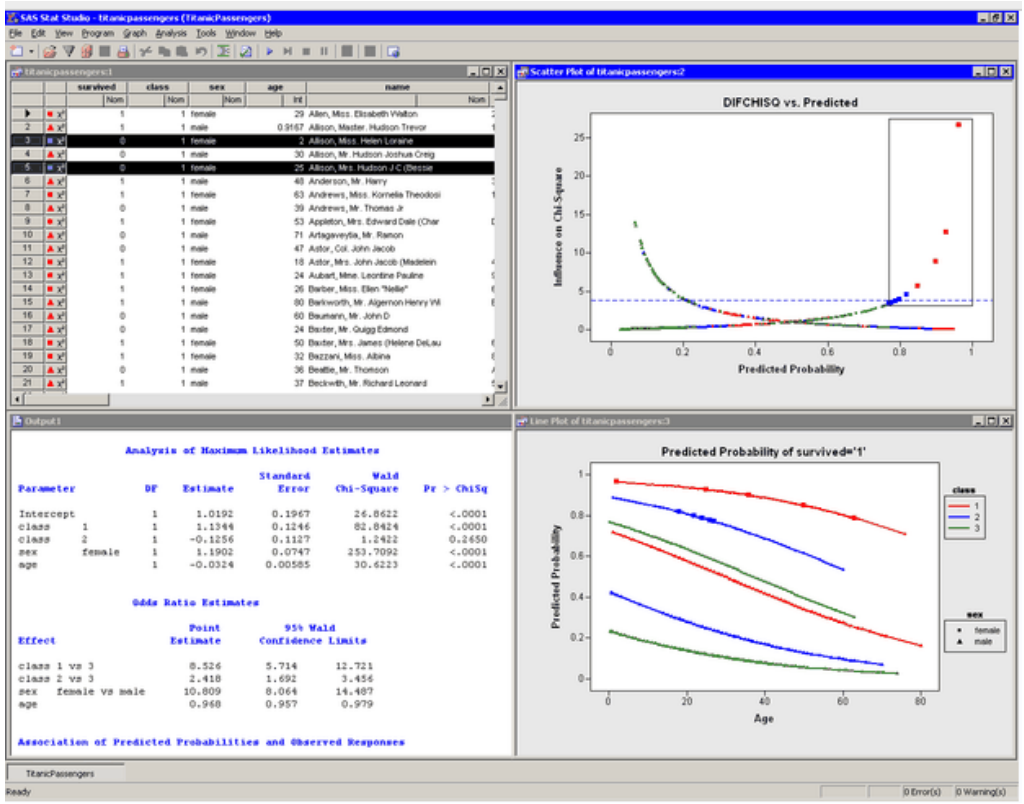
- Can be Used for a wide Range of Problems
- Extensive set of Operators

SAS/IML STUDIO

WHAT IS IT?

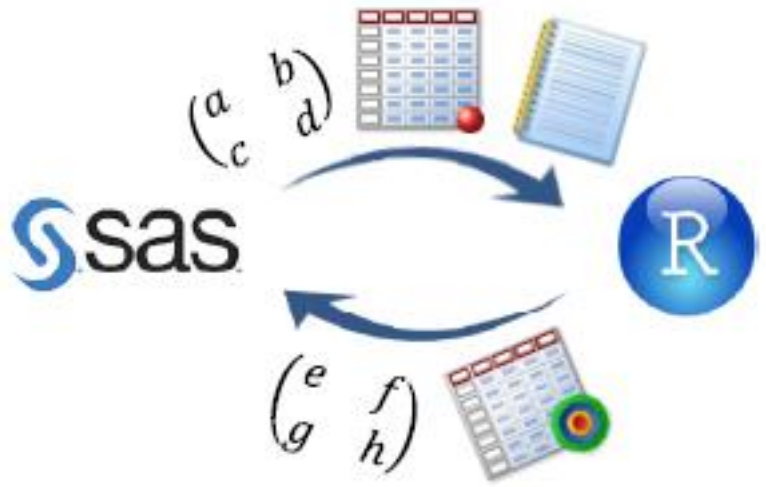
SAS/IML Studio is an interface that enables you to interactively debug and execute SAS/IML programs and adds dynamic graphics for exploratory data analysis

- explore data
- subset data
- analyze univariate distributions
- fit explanatory models
- investigate multivariate relationships
- Plus IDE to write, debug & execute programs
    - Flexibility of SAS/IML language
    - Analytics of SAS/STAT® Procedures
    - Data Manipulation of Base SAS®
    - Dynamically linked graphs
    - Functions and Packages of R



Using SAS/IML to Interact with R

- Send IML matrices and SAS data sets to R
- Submit R code in the IML script
- Return R results as IML matrices or SAS data sets



THREE MAIN INTEGRATION POINTS

- Transfer from a SAS source to an R target
- Transfer the results from an R source to a number of SAS data structures
- Submit R code and call R functions from PROC IML

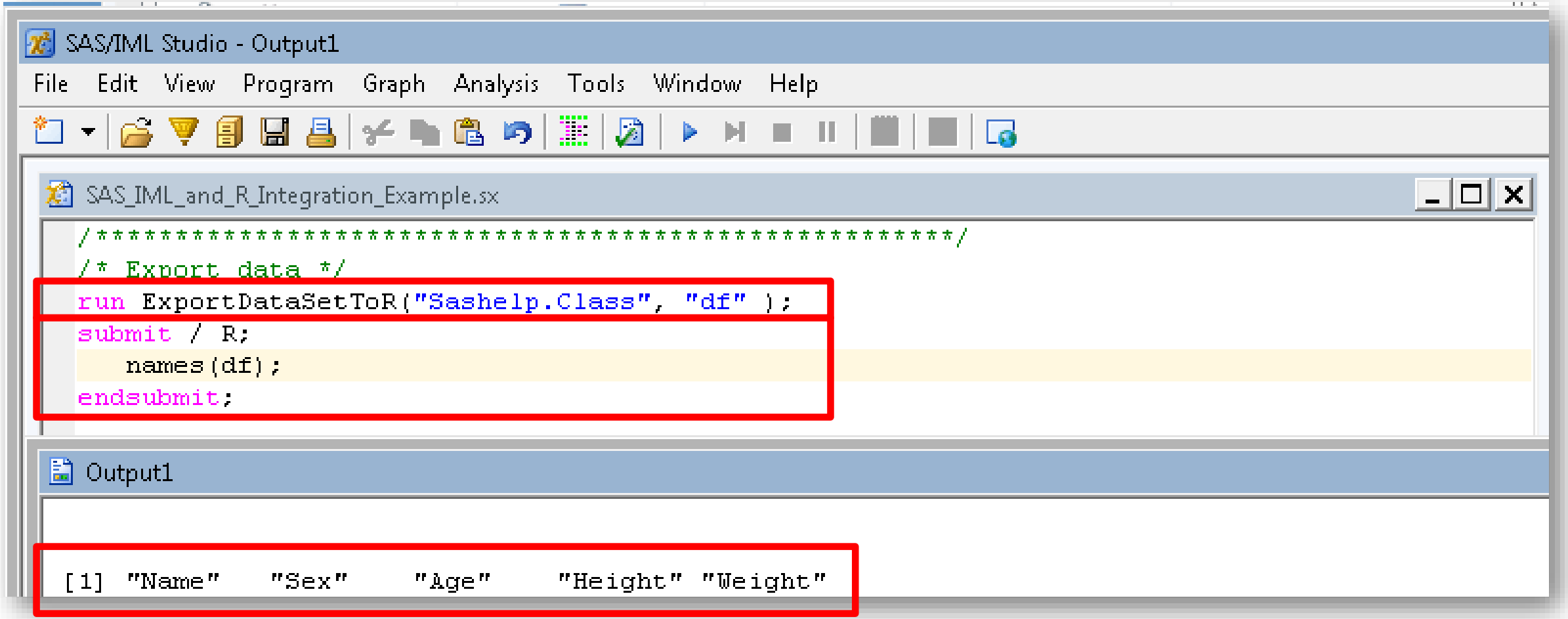
IML User's Guide:

http://support.sas.com/documentation/cdl/en/imlug/67502/PDF/default/imlug.pdf

TRANSFER FROM A SAS SOURCE TO AN R TARGET

METHOD or MODULE	SAS SOURCE	R TARGET
ExportDataSetToR	SAS data set	R data frame
ExportMatrixToR	SAS/IML Matrix	R matrix
DataObject.ExportToR	DataObject	R data

ExportDataSetToR Method:



ExportMatrixToR

```
/* Set-up for Calling R */
declare DataObject dobj;
dobj = DataObject.CreateFromFile("Hurricanes");    /* Create a data object from a SAS data set */
dobj.GetVarData( "wind_kts", w );                  /* Copy column "wind_kts" into IML vector   */
dobj.GetVarData( "min_pressure", p );              /* Copy column "min_pressure" into IML vector */

/* send matrices to R */
run ExportMatrixToR( w, "Wind" );                  /* Export IML vector w to R variable "Wind"   */
run ExportMatrixToR( p, "Pressure" );              /* Export IML vector p to R variable "Pressure" */
```

TRANSFER FROM AN R SOURCE TO A SAS TARGET

METHOD or MODULE	R SOURCE	SAS TARGET
DataObject.AddVarFromR	R expression	DataObject variable
DataObject.CreateFromR	R expression	DataObject
ImportDataSetFromR	R expression	SAS data set
ImportMatrixFromR	R expression	SAS/IML matrix

ImportMatrixFromR:

```
print "-----------  In SAS/IML  -------------";
run ImportMatrixFromR( pe, "ParamEst" );              /* Import R vector ParmEst to IML vector pe */
print pe[r={"Intercept" "min_pressure"}];             /* Print print parameter estimates computed R */

/* add variables to the DataObject */
dobj.AddVarFromR( "R_Pred", "Pred" );                 /*Add variable R_Pred from R vector Pred */
dobj.AddVarFromR( "R_Resid", "Resid" );               /*Add variable R_Resid from R vector Resid */
ScatterPlot.Create(dobj, "min_pressure", "R_Resid"); /* Plot R calculated residuals  */
```

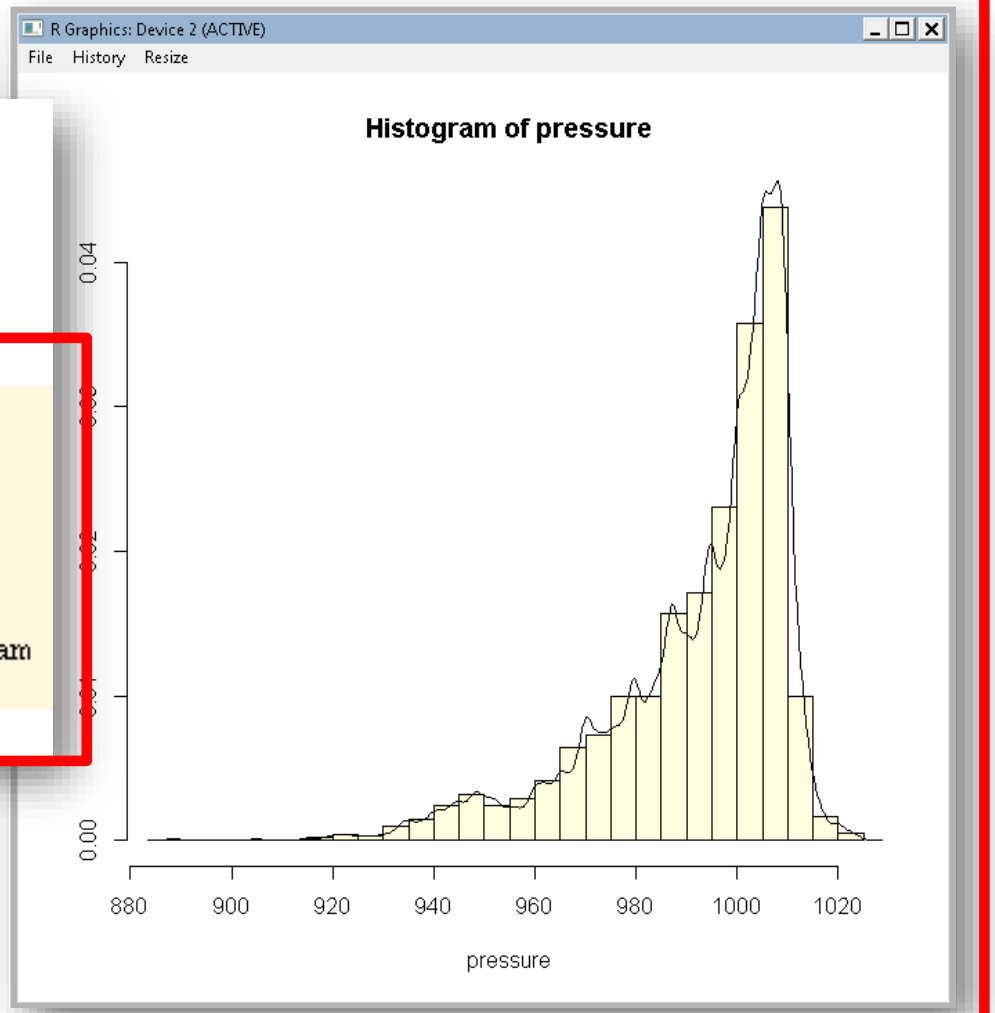
```
-----------  In SAS/IML  -------------

            pe

Intercept     1333.3549
min_pressure -1.291374
```

```
print "---------------   In R   ----------------";    /* Fit linear model in R */
submit / R;
  Model     <- lm(Wind~Pressure, na.action="na.exclude")
  ParamEst <- coef(Model)                                  # Create R vector ParamEst
  Pred <- fitted(Model)                                    # Creare R vector Pred
  Resid     <- residuals(Model)                            # Create R vector Resid
  print (ParamEst)
endsubmit;
```

```
---------------   In R   ----------------
(Intercept)     Pressure
1333.354893    -1.291374
```

SAS & R | CALLING R ANALYTICS & GRAPHICS PACKAGES

```
/* Calling packages and graphics */
declare DataObject dobj;
dobj = DataObject.CreateFromFile("Hurricanes");
dobj.GetVarData("min pressure", pressure);
run ExportMatrixToR( pressure, "Pressure" );

submit / R;
   library(KernSmooth)
   idx <-which(!is.na(Pressure))      # must exclude missing values (NA)
   pressure <- Pressure[idx]                  #    from KernSmooth functions
   h = dpik(p)                        # Sheather-Jones plug-in bandwidth
   est <- bkde(pressure, bandwidth=h)         # est has 2 columns

   hist(pressure, breaks="Scott", freq=FALSE, col="lightyellow") # histogram
   lines(est)                                                    # kde overlay
endsubmit;
```



SAS/IML®

R for data modeling

1. Read data into SAS/IML vectors

```
proc iml;
use Sashelp.Class;
read all var {Weight Height};
close Sashelp.Class;
```

2. Transfer data to R

```
/* send matrices to R */
call ExportMatrixToR(Weight, "w");
call ExportMatrixToR(Height, "h");
```

3. Call R functions for data analysis

```
submit / R;
Model <- lm(w ~ h,
na.action="na.exclude") # a
ParamEst <- coef(Model) # b
Pred <- fitted(Model)
Resid <- residuals(Model)
endsubmit;
```

4. Transfer results into SAS/IML vectors

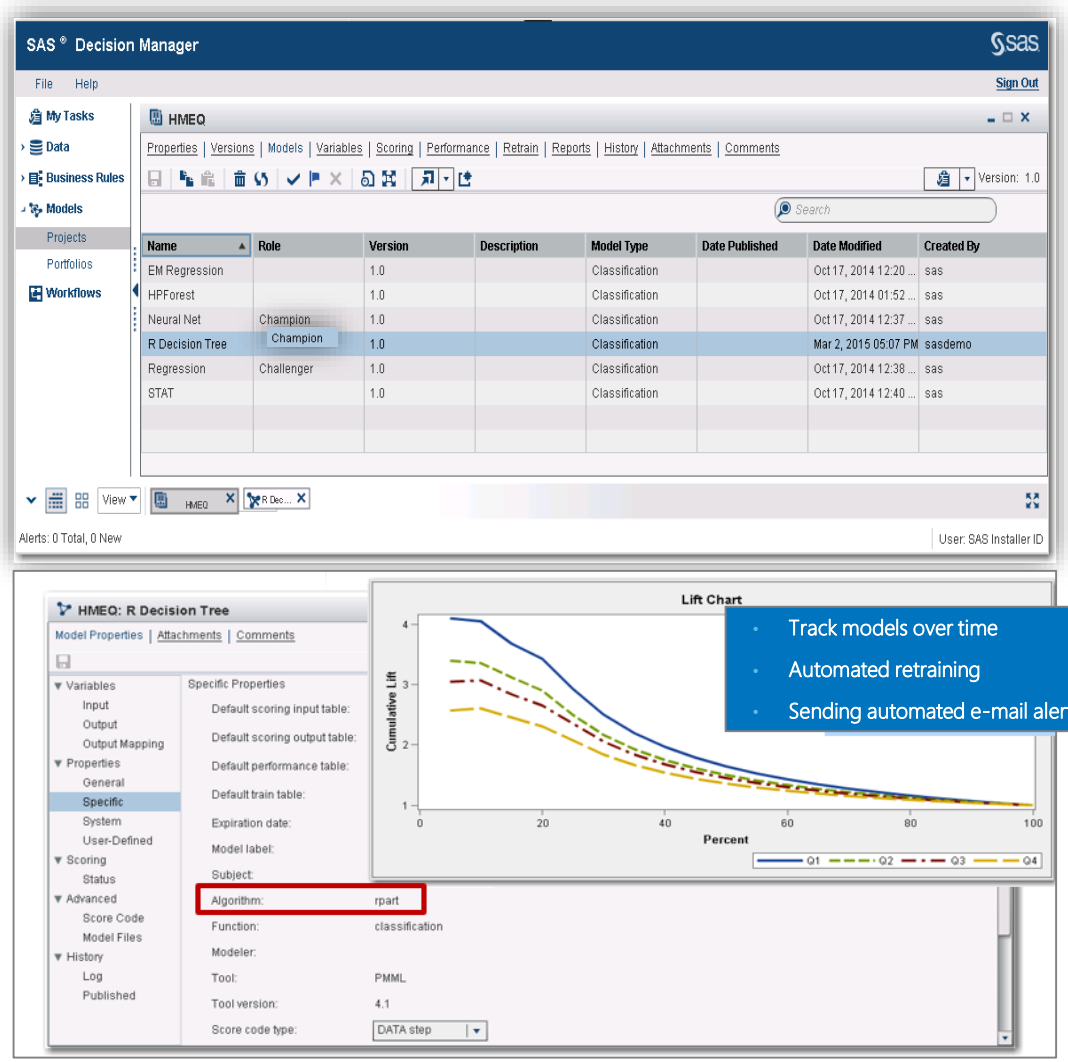
```
call ImportMatrixFromR(pe,
"ParamEst");
print pe[r={"Intercept" "Height"}];
ht = T( do(55, 70, 5) );
A = j(nrow(ht),1,1) || ht;
pred_wt = A * pe;
print ht pred_wt;
```

Integration with SAS Model Manager

Using SAS Model Manager to Organize R and Python

- SAS Model Manager streamlines the steps of creating, managing, deploying, monitoring, and operationalizing analytic models

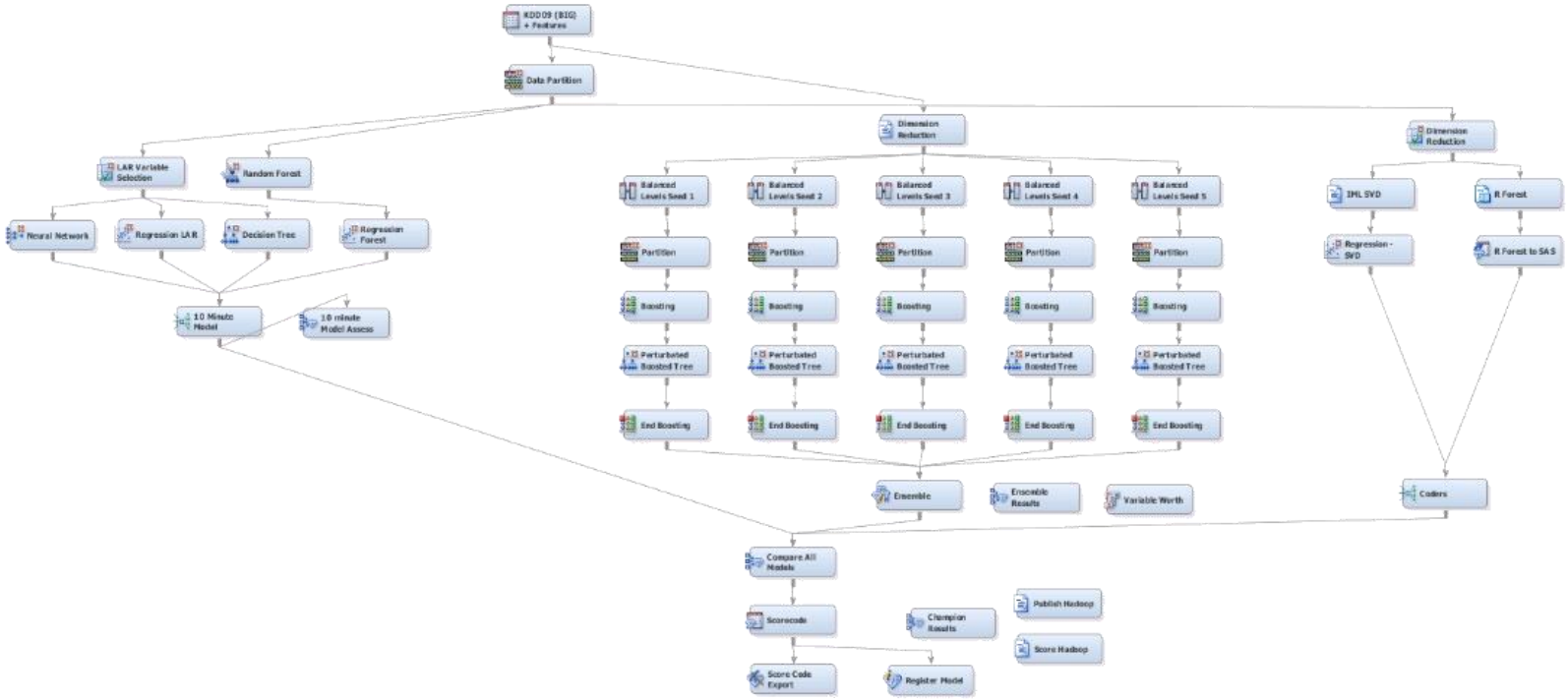
- Can accept R & Python models



Using R & Python in SAS Enterprise Miner

Using SAS Enterprise Miner with Python and R

- Leverage SAS advanced analytics in a easy-to-use GUI environment
- Leverage R & Python modeling packages from a SAS environment
- Compare & Ensemble R, Python, and SAS models in one interface



SAS® Enterprise Miner and R

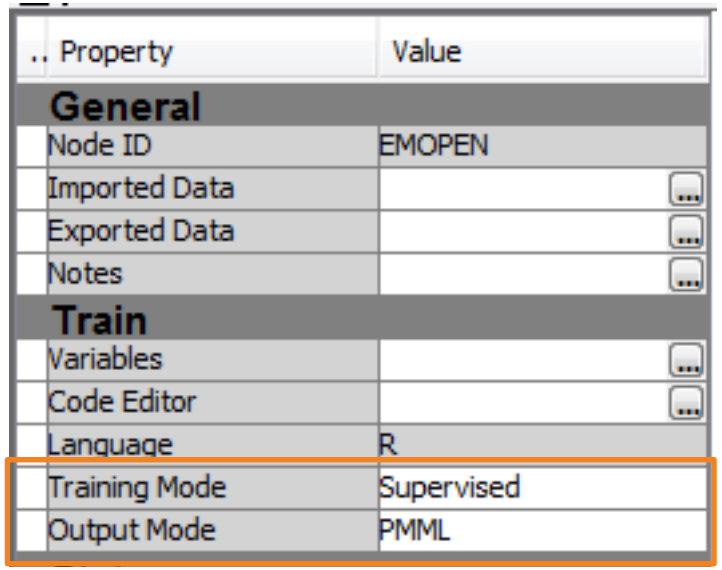
R integration
SAS® Enterprise Miner

- SAS Enterprise Miner Open Source Integration node
  - Enables the **execution of R code** within an Enterprise Miner flow
    - Facilitates **multitasking** in R
    - Generates **text and graphical output** from R
    - Integrates both **supervised and unsupervised** learning tasks
  - Transfers data, metadata, and results automatically between Enterprise Miner and R
  - Uses SAS/IML under the covers

Open source integration node

Modes of operation

- Training Mode
  - Supervised
  - Unsupervised
- Output Mode
  - PMML: Creates SAS Data step score code
  - Merge: Merge inputs with predictions
  - None: Troubleshooting R code, output graphs, simulations etc

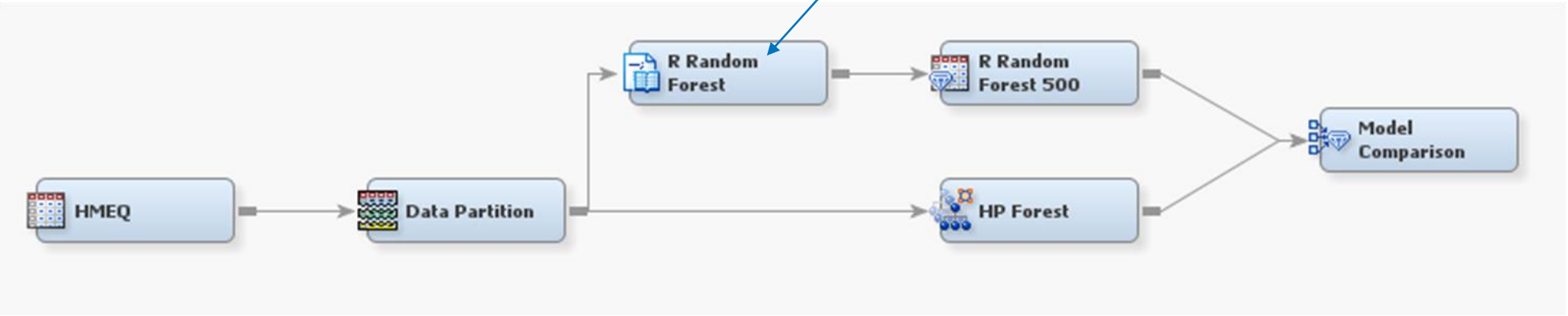


SAS and Open Source

SAS Enterprise Miner

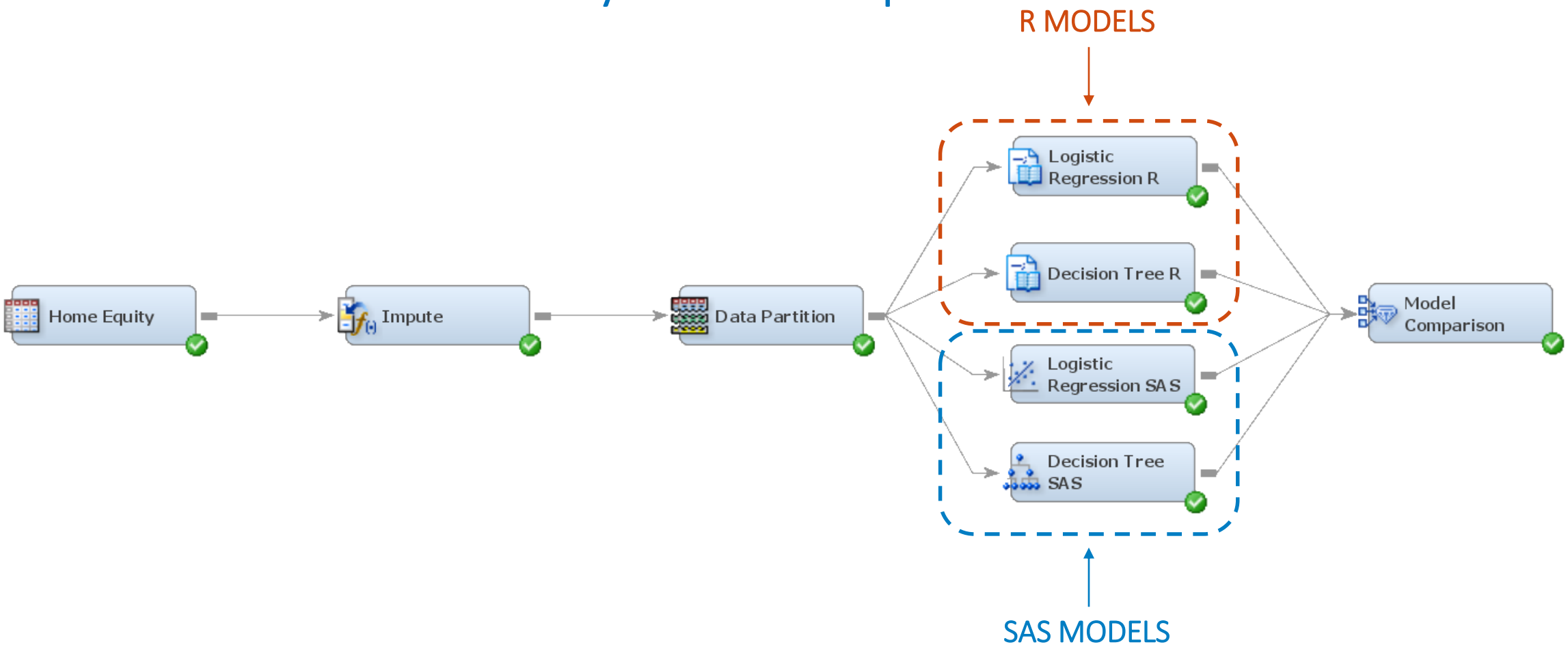
- Integrate R code inside Enterprise Miner
  - Includes R models in model assessment
  - SAS and R ensemble models

```
library(randomForest)

&EMR_MODEL <- randomForest(&EMR_CLASS_TARGET ~  &EMR_CLASS_INPUT + &EMR_NUM_INPUT, ntree= 500, mtry= 5, data= &EMR_IMPORT_DATA, importance= TRUE)

&EMR_EXPORT_TRAIN <- predict(&EMR_MODEL, &EMR_IMPORT_DATA, type="prob")


&EMR_EXPORT_VALIDATE <- predict(&EMR_MODEL, &EMR_IMPORT_VALIDATE, type="prob")
&EMR_EXPORT_TEST <- predict(&EMR_MODEL, &EMR_IMPORT_TEST, type="prob")

&EMR_EXPORT_TRAIN[1:10,]

png("EMR_forestMsePlot.png")
plot(&EMR_MODEL, main= 'randomForest MSE Plot')
dev.off()

write.table(round(importance(&EMR_MODEL),2), file= "EMR_forestImportance.csv", sep= ",", row.names= TRUE, col.names= TRUE)

print(&EMR_MODEL)
  round(importance(&EMR_MODEL),2)
```



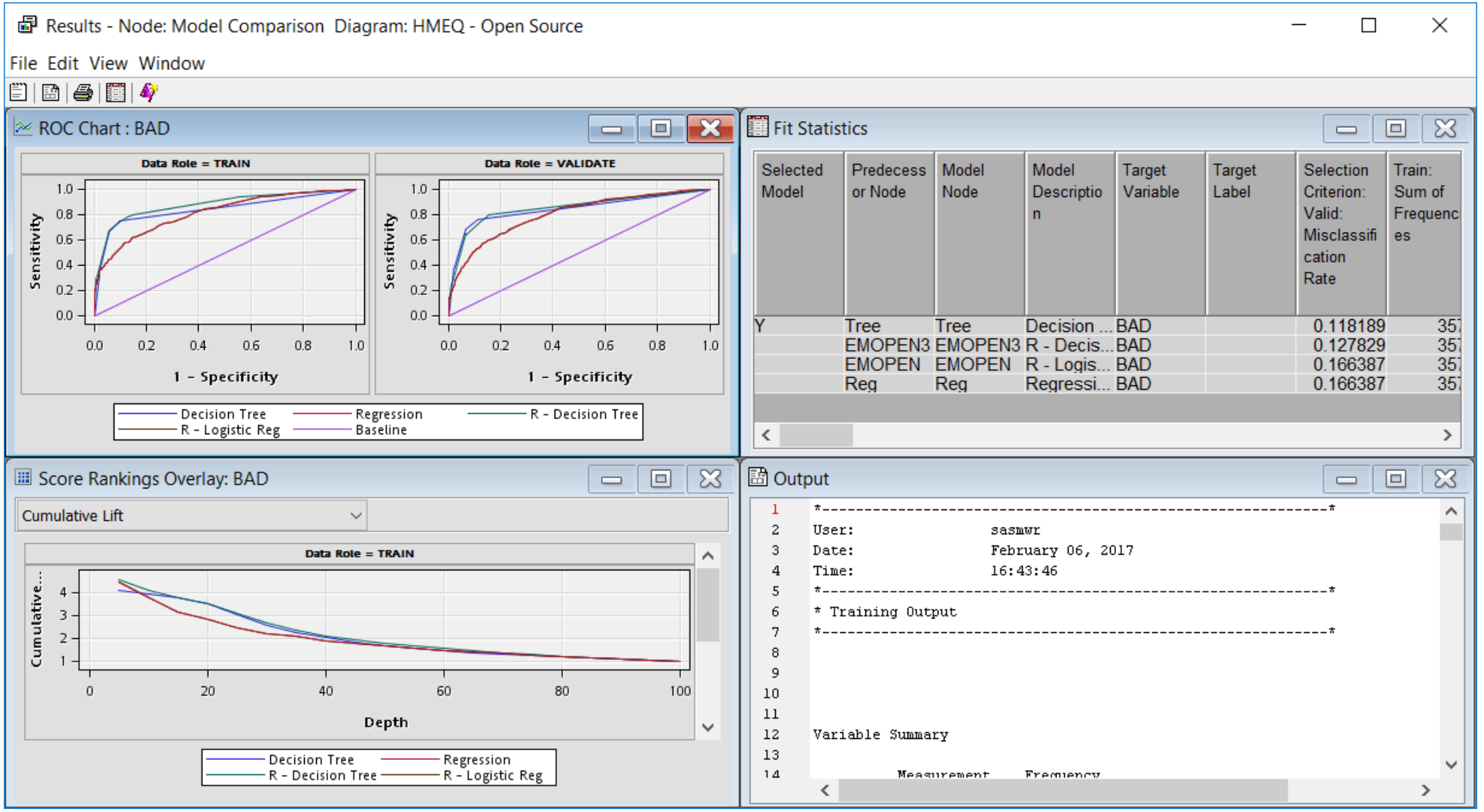
SAS and Open Source

Discovery – R in Enterprise Miner



SAS and Open Source

SAS and R Model Comparison



SAS and Open Source

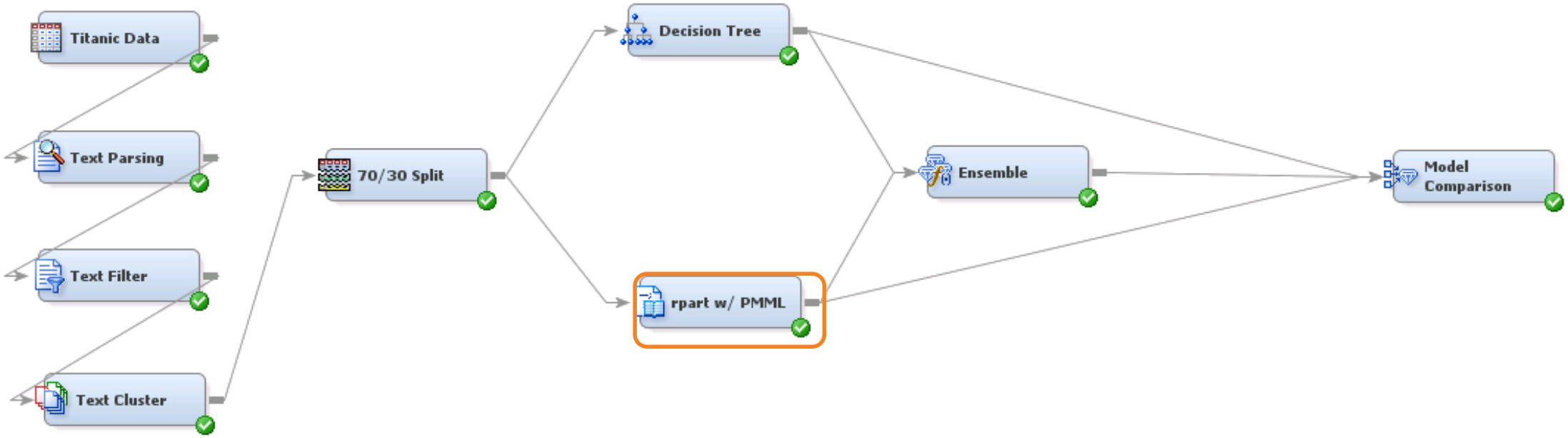
SAS and R Model Comparison

Fit Statistics

Selected Model	Predecessor Node	Model Node	Model Description	Target Variable	Target Label	Selection Criterion: Valid: Misclassification Rate	Train: Sum of Frequencies	Train: Misclassification Rate	Train: Maximum Absolute Error	Train: Sum of Squared Errors	Train: Average Squared Error	Train: Root Average Squared Error	Train: Divisor for ASE	Train: Total Degrees of Freedom	Valid: Sum of Frequencies
Y	Tree	Tree	Decision ...	BAD		0.118189	3574	0.114997	0.935823	655.6948	0.091731	0.302872	7148	3574	2386
	EMOPEN3	EMOPEN3	R - Decis...	BAD		0.127829	3574	0.113318	0.964939	632.8894	0.088541	0.297558	7148	.	2386
	EMOPEN	EMOPEN	R - Logis...	BAD		0.166387	3574	0.148573	0.999547	821.2193	0.114888	0.338951	7148	.	2385
	Reg	Reg	Regressi...	BAD		0.166387	3574	0.148293	0.999547	821.2158	0.114887	0.338951	7148	3574	2386

Using R in SAS Enterprise Miner

PMML output mode



What is PMML?

- A declarative, XML-based, open standard for representing predictive models
- Allows predictive models to be transferred between different languages and products
- What analytical tools can create PMML? R, SAS, Python, etc. etc. etc.
- What scoring tools can consume PMML? Teradata, Zementis, SAS, IBM

More info: http://www.dmg.org/products.html

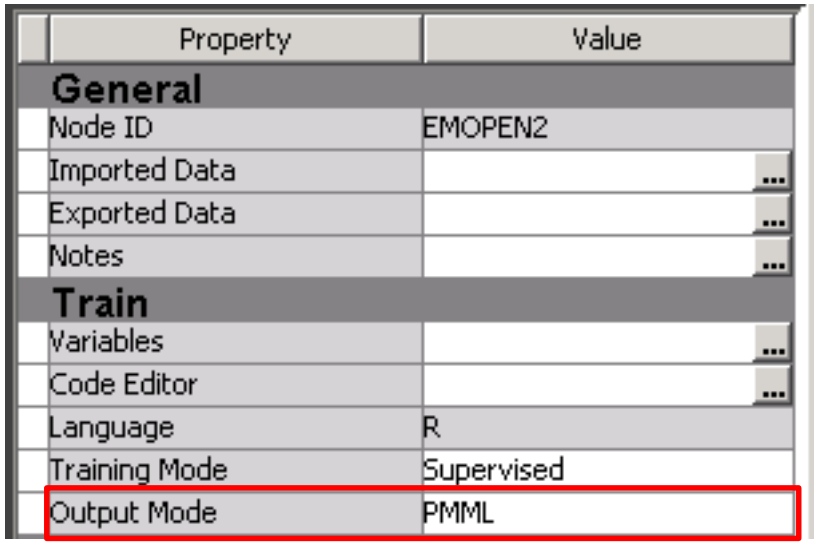
Using R in SAS Enterprise Miner

PMML output mode

Predictive modeling markup language (PMML) is an open standard enabling certain R models to be translated into SAS DATA step code

Currently supported R models include:

- Linear models (lm)
- Multinomial log-linear models (multinom {nnet})
- Generalized linear models (glm {stats})
- Decision trees (rpart)
- Neural networks (nnet)
- *K*-means clustering (kmeans {stats})



Open source integration node

data handles

- In EM, access to inputs and outputs of a node are through data handles
- Handles are case-sensitive !

	Inputs	Outputs
Training Data	&EMR_IMPORT_DATA	&EMR_EXPORT_TRAIN
Validation Data	&EMR_IMPORT_VALIDATE	&EMR_EXPORT_VALIDATE
Test Data	&EMR_IMPORT_TEST	&EMR_EXPORT_TEST
Score Data	&EMR_IMPORT_SCORE	&EMR_EXPORT_SCORE

Open source integration node

MODEL & Variable handles

- Model Handle &EMR_MODEL translates to R model
- Variable handles provide access to data set variables

Variable Handles	Description
&EMR_NUM_INPUT	List of + separated Interval level input variables
&EMR_CLASS_INPUT	List of + separated Binary, Nominal or Ordinal level input variables
&EMR_NUM_TARGET	Single Interval target variable
&EMR_CLASS_TARGET	Single Binary, Nominal or Ordinal target variable

Using R in SAS Enterprise Miner

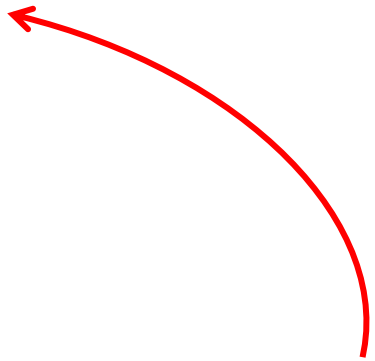
PMML output mode

```
library(rpart)

&EMR_MODEL <- rpart(&EMR_CLASS_TARGET ~ &EMR_CLASS_INPUT +
&EMR_NUM_INPUT, data= &EMR_IMPORT_DATA, method= "class")
```

Using R in SAS Enterprise Miner

PMML output mode

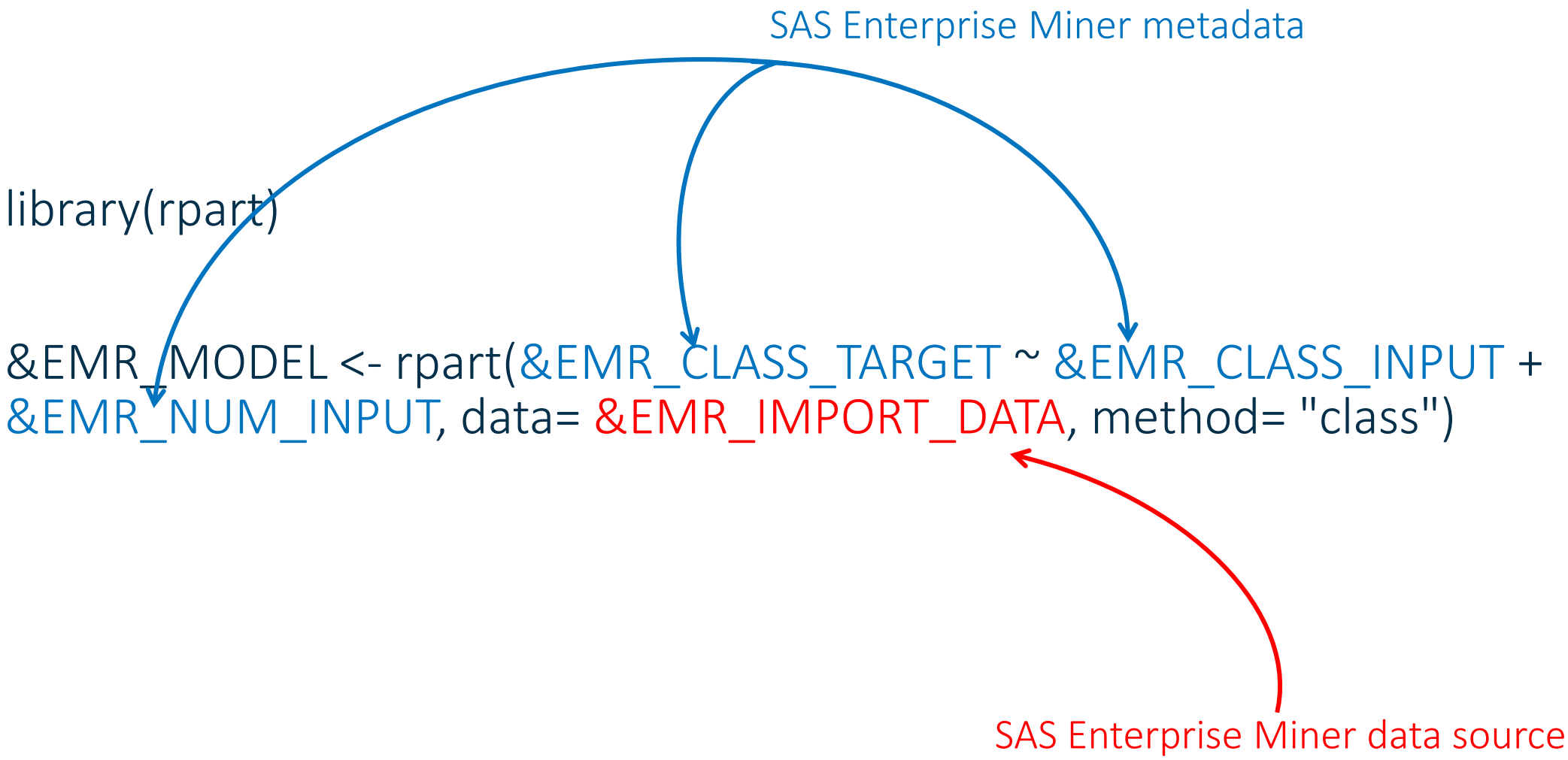
```
library(rpart)

&EMR_MODEL <- rpart(&EMR_CLASS_TARGET ~ &EMR_CLASS_INPUT +
&EMR_NUM_INPUT, data= &EMR_IMPORT_DATA, method= "class")
```



SAS Enterprise Miner data source

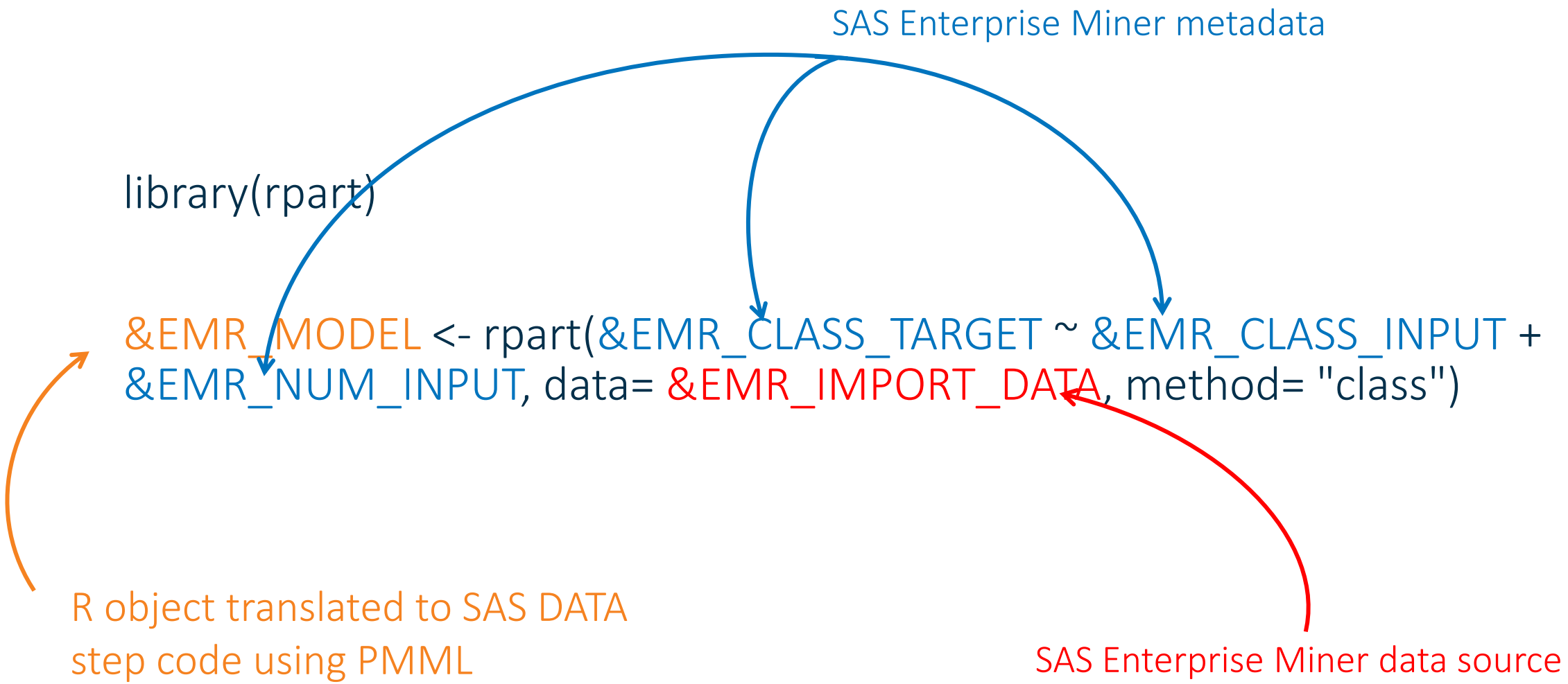
Using R in SAS Enterprise Miner

PMML output mode



Using R in SAS Enterprise Miner

PMML output mode



SAS Enterprise Miner Open Source Integration node

Environment

Train model

Generate PMML with R PMML package

Convert PMML to SAS code using PROC PSCORE

SAS/IML

Base SAS

Production Environment

Score new data using SAS tools or native process

SAS Environment

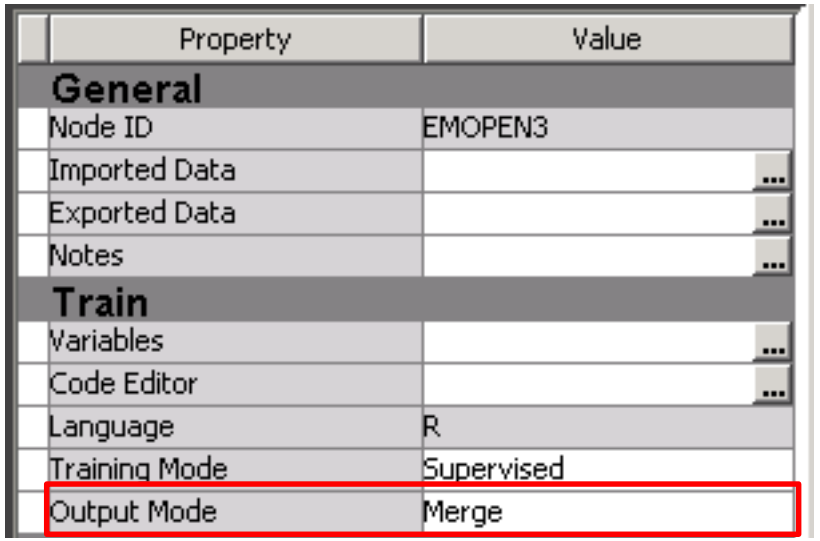
Using R in SAS Enterprise Miner

Merge output mode

Merge output mode enables integration with the thousands of R packages that are not supported in PMML output mode

Variables created in R are merged with SAS Enterprise Miner data sources by the user

SAS DATA step code is not created



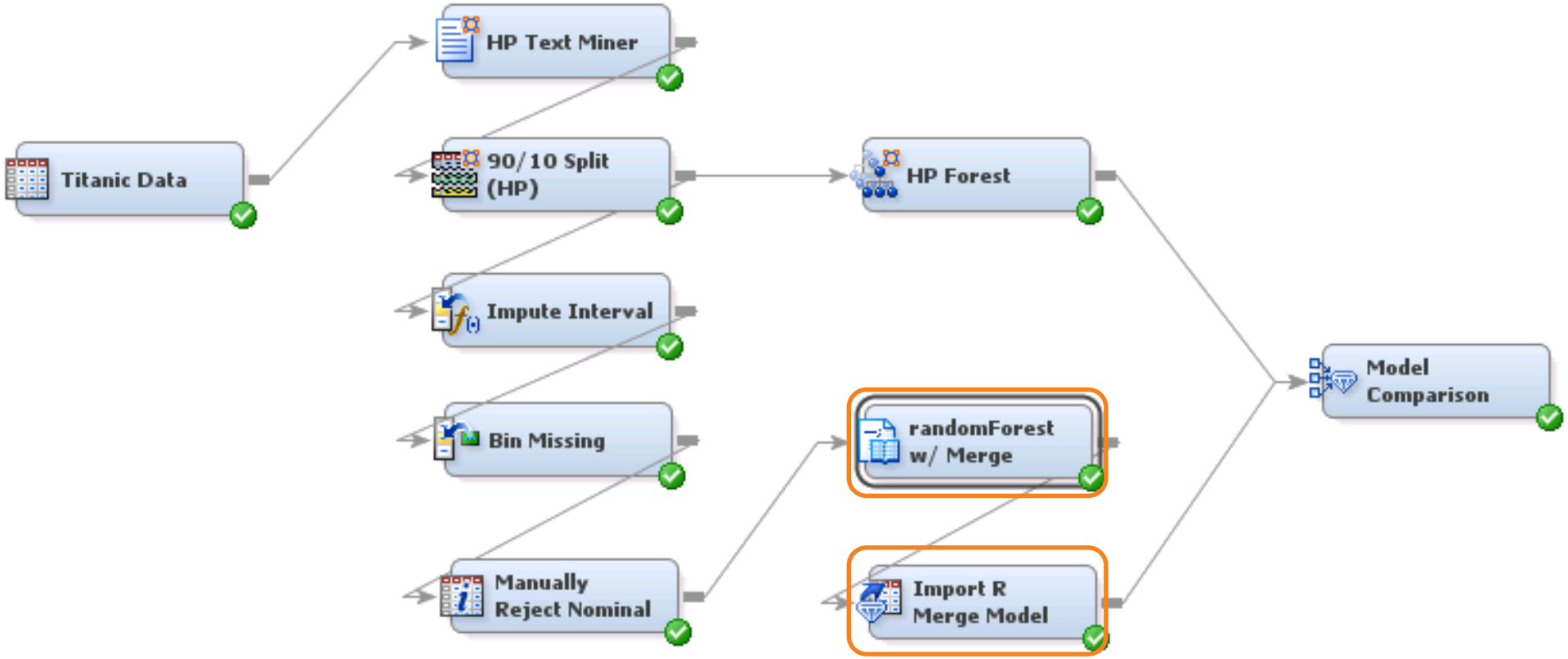
Open source integration node

Merge output mode

- Under the covers, a direct Data step Merge is attempted, hence the output should have same number of records as imported (or input) data
  - Missing values must be imputed in EM prior to the node or within R code in the node
  - Order of output data matches the order of input data
- Supervised models created must be imported and assessed with the Model Import node
- Unsupervised models can be assessed with Segment Profile node if segment variable is created

Using R in SAS Enterprise Miner

Merge output mode



Using R in SAS Enterprise Miner

Merge output mode

```
library(randomForest)


&EMR_MODEL <- randomForest(&EMR_CLASS_TARGET ~  &EMR_CLASS_INPUT + &EMR_NUM_INPUT, ntree=
250, mtry= 5, maxnodes= 50, data= &EMR_IMPORT_DATA,

importance= TRUE)


&EMR_EXPORT_TRAIN <- predict(&EMR_MODEL, &EMR_IMPORT_DATA, type="prob")

&EMR_EXPORT_VALIDATE <- predict(&EMR_MODEL, &EMR_IMPORT_VALIDATE, type="prob")


&EMR_EXPORT_TRAIN[1:10,]
```

Best practice

User must explicitly create exported variables to be merged with Enterprise Miner data sources

SAS® Enterprise Miner and Python

configuration

- Setup Java CLASSPATH

- Create new EM project and add the following start-up code into Project Start Code window

```
*** WORKING DIRECTORY      (----- USER UPDATE NEEDED -----);
%let WORK_DIR = C:\SGF2015\OpenSrcIntegration;
*** SYSTEM PYTHON LOCATION (----- USER UPDATE NEEDED -----);
%let PYTHON_EXEC_COMMAND = C:\Anaconda\python.exe;
*** JAVA LIBRARIES/CLASS FILES LOCATION;
%let JAVA_BIN_DIR = &WORK_DIR.\bin;
options linesize = MAX;
```

Simple diagram

- The code in SAS Code Node:
  - Executes Python script using Java Object in DATA step
  - Imports output CSV files from Python script into SAS data sets
  - Merge train data with predictions

Simple diagram

- Add metadata to the output data set

Name	Hidden	Hide	Role	New Role	Level	New Level	New Order	New Report
label	N	Default	Label	Target	Interval	Binary	Default	Default
p_label1	N	Default	Prediction	Default	Interval	Default	Default	Default
p_label7	N	Default	Prediction	Default	Interval	Default	Default	Default
pixel0	N	Default	Input	Default	Interval	Default	Default	Default

Simple diagram

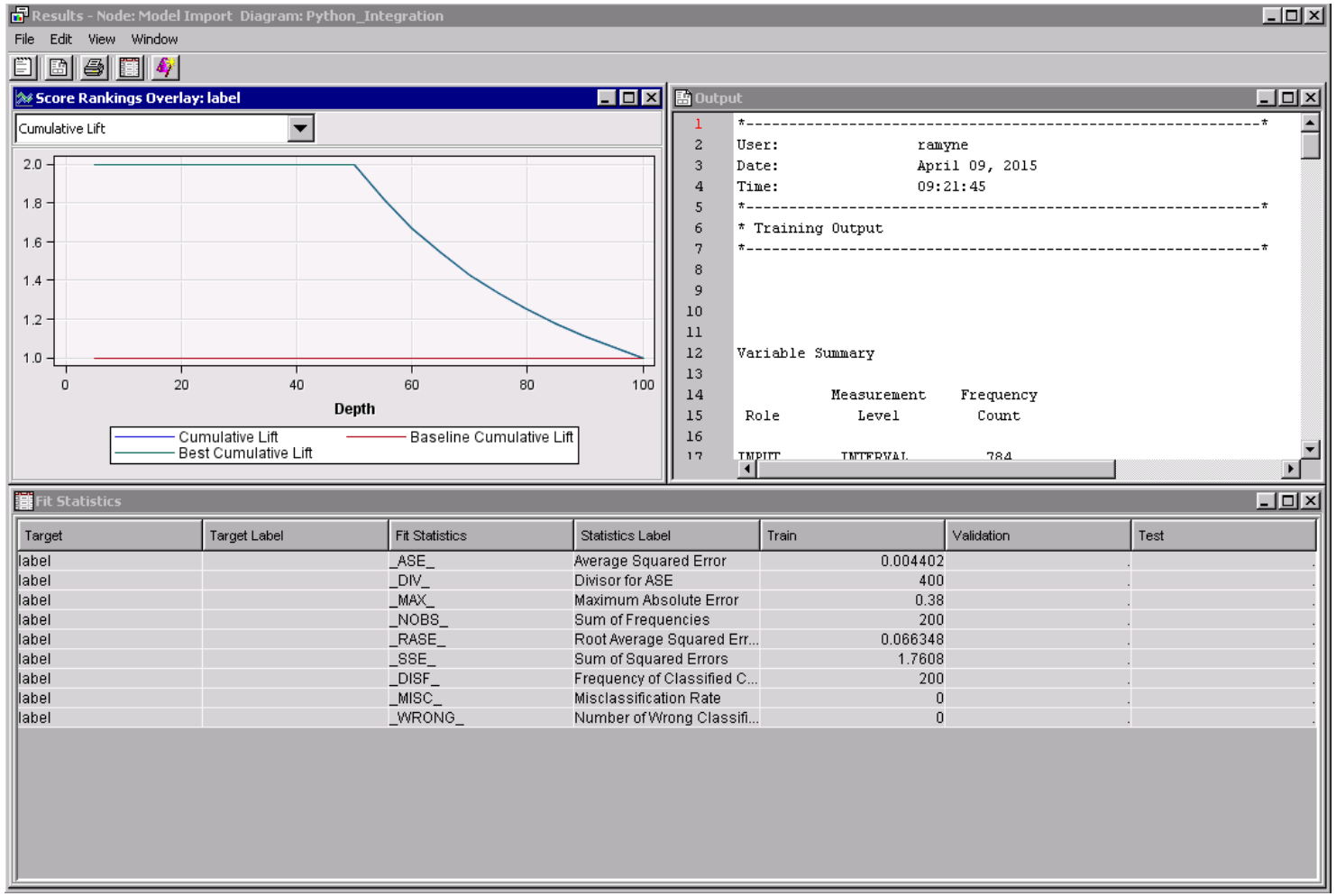
- Map Predicted Variables

Level	Predicted Variable	Modeling Variable	Predicted Variable Label
1	P_label1	P_label1	Predicted: label=1
7	P_label7	P_label7	Predicted: label=7

- Model Import can further be connected to Model Comparison node

results

Fit statistics



Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
label		_ASE_	Average Squared Error	0.004402	.	.
label		_DIV_	Divisor for ASE	400	.	.
label		_MAX_	Maximum Absolute Error	0.38	.	.
label		_NOBS_	Sum of Frequencies	200	.	.
label		_RASE_	Root Average Squared Err...	0.066348	.	.
label		_SSE_	Sum of Squared Errors	1.7608	.	.
label		_DISF_	Frequency of Classified C...	200	.	.
label		_MISC_	Misclassification Rate	0	.	.
label		_WRONG_	Number of Wrong Classifi...	0	.	.

In conclusion…

Open source integration node

Output

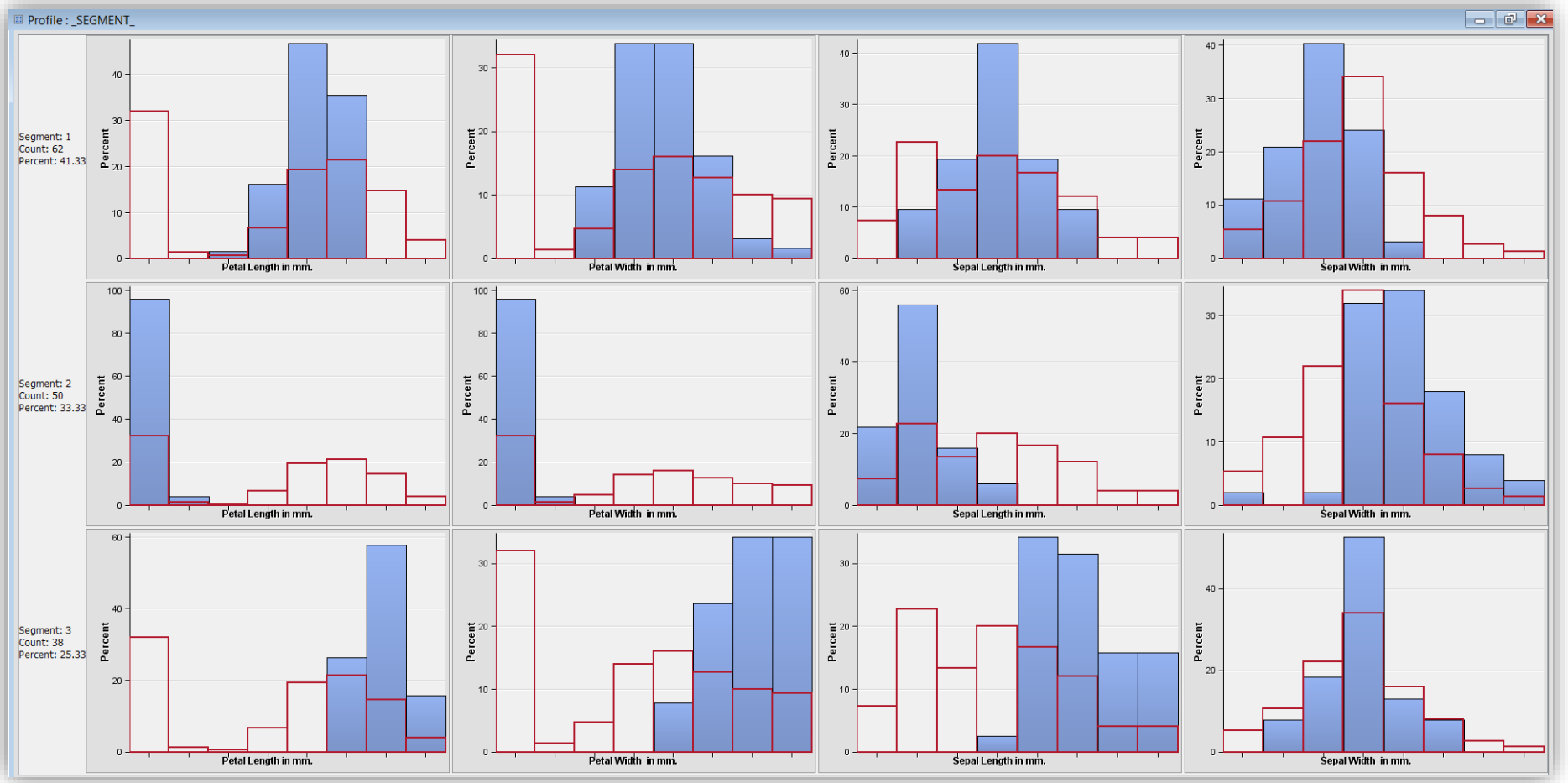
- SAS Data Step Score Code
  - Training Mode=Supervised, Unsupervised Output Mode=PMML
- Fit Statistics and Assessment Charts
  - Training Mode=Supervised, Output Mode=PMML
  - Training Mode=Supervised, Output Mode=Merge, followed by Model Import node
  - Training Mode=Unsupervised, Output Mode=PMML, Merge followed by Metadata and Segment Profile node
- Output, graphics {JPG, PNG, GIF}, text files {CSV, TXT}
  - All modes

Open source integration node
TIPS

- Errors in R & Python bring up a generic message in EM
  - Example: “After data transfers and R code submission, SYSCC=1012. Please see log for more details”.
  - Search for ERROR: R: string in SAS log for additional details.
- Variable names are case-sensitive in R but not in SAS
- Missing values in data might cause problems; use Imputation in Enterprise Miner or R to address this issue
- Be mindful when data (variables or variable levels) have SAS formats or special characters

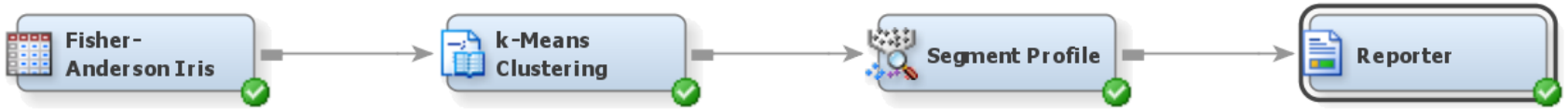
Open source integration

Discovery – visualization



Open source integration

Discovery – Self documenting processes



REPORT.pdf - Adobe Reader

File Edit View Window Help

Open 1 / 6 100% Tools Fill & Sign Comment

SAS Enterprise Miner Report

User = wephan
Date = 17:58:03 October 24
Project = EM_and_R
Diagram = document_clustering_r

Start Node = Report
Node label = Reporter
Nodes = PATH
Showall = N

Format = PDF
Style = LISTING

SAS Enterprise Miner Report
Process Flow Diagram

SAS Enterprise Miner Report

Node=Fisher-Anderson Iris
Summary

Node id = Ids
Node label = Fisher-Anderson Iris
Meta path = Ids
Notes =

Node=Fisher-Anderson Iris
Properties

Property	Value	Default	Property	Value	Default	Property	Value	Default
Component	DataSource		DropMapVariables	N	Y	NewTable	SAMPSIO.DMAIRIS	
ApplyIntervalLevelLowerLimit	Y		IdentifyEmptyColumns	Y		NewVariableRole	REJECT	
ApplyMaxClassLevels	Y		IntervalLowerLimit	20		OutputType	VIEW	
ApplyMaxPercentMissing	Y		Library	SAMPSIO		Role	TRAIN	
ComputeStatistics	N		MaxClassLevels	20		Sample	D	
DBPassThrough	Y		MaxPercentMissing	50		SampleSizeObs	10000	
Data	SAMPSIO.DMAIRIS		MetaAdvisor	BASIC		SampleSizePercent	20	
DataSelection	USERTABLE	DATASOURCE	NBytes	66560	.	SampleSizeType	PERCENT	
DataSource			NCols	5	.	Segment		
Description	All: Fisher-Anderson Iris Data		NObs	150	.	Table	DMAIRIS	

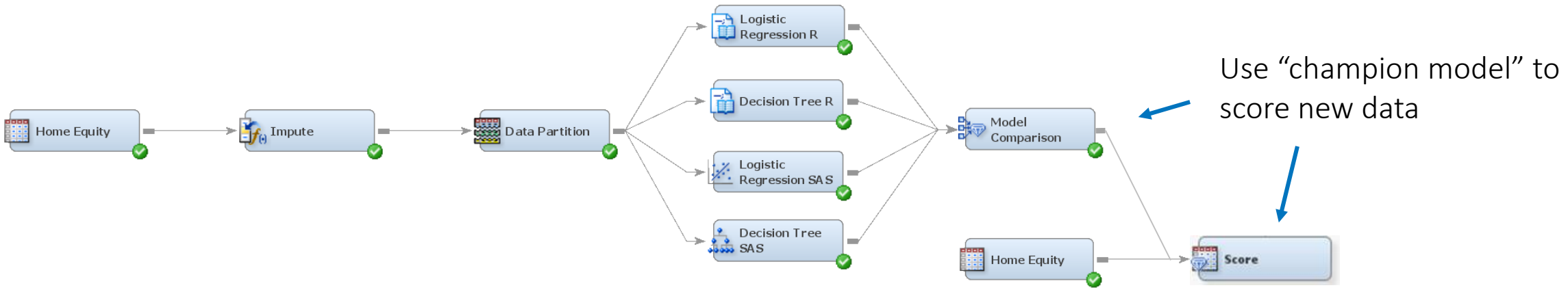
Node=Fisher-Anderson Iris
Data Attributes

Attribute	Value	Attribute	Value	Attribute	Value
Data Name	DMAIRIS	Date Created	20Jun2013:01:47:06	Data Size	66560
Data Type	DATA	Date Modified	20Jun2013:01:47:06	Role	TRAIN
Data Label	All: Fisher-Anderson Iris Data	Number Rows	150	Segment	
Engine	V9	Number Columns	5	Data Library	SAMPSIO

8.50 x 11.00 in

Open source integration

Score code generation to evaluate new data



Several deployment options:

- In modeling workbench to generate scoring table
- In-database with native scoring accelerators
- In batch in data integration processes
- Real-time with Decision services
- For Streaming Analytics

Open Source Integration

Other Benefits of Open Source in Enterprise miner

Model Building

- Use open source packages for model building

Multi-Threaded Processing of Workflows

- Use Open Source node in various flows simultaneously
- Enterprise Miner handles multi-threaded execution

Scoring

- Create supported models that can convert into scoring code for operational deployment (i.e. in-database)

Collaboration

- Many users can access the same EM diagram

General Use

- Use any packages within Enterprise Miner (EM), leveraging all EM functionality in prior nodes (i.e. data prep, prep-processing)

Integration with python & MORE …

SASPy and SAS Pipefitter

For Jupyter Notebooks

Project headed by Jared Dean

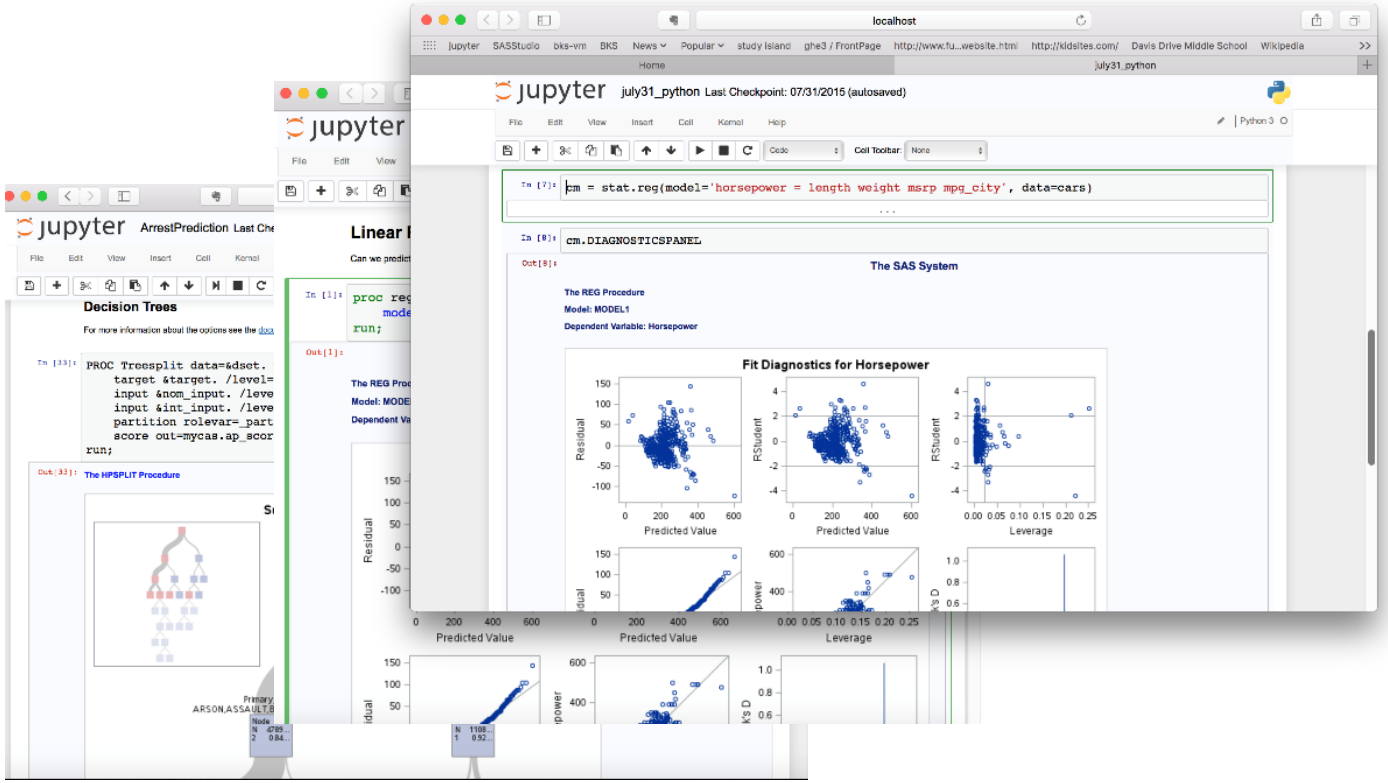
https://github.com/sassoftware

Open Source

Example: IDE Choice - Jupyter SAS Kernel

- The **Jupyter** Notebook is a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Currently supports about 40 languages.

- SAS Kernel released as Open source.
- Jupyter being included in SAS University Edition
- More integration coming soon…

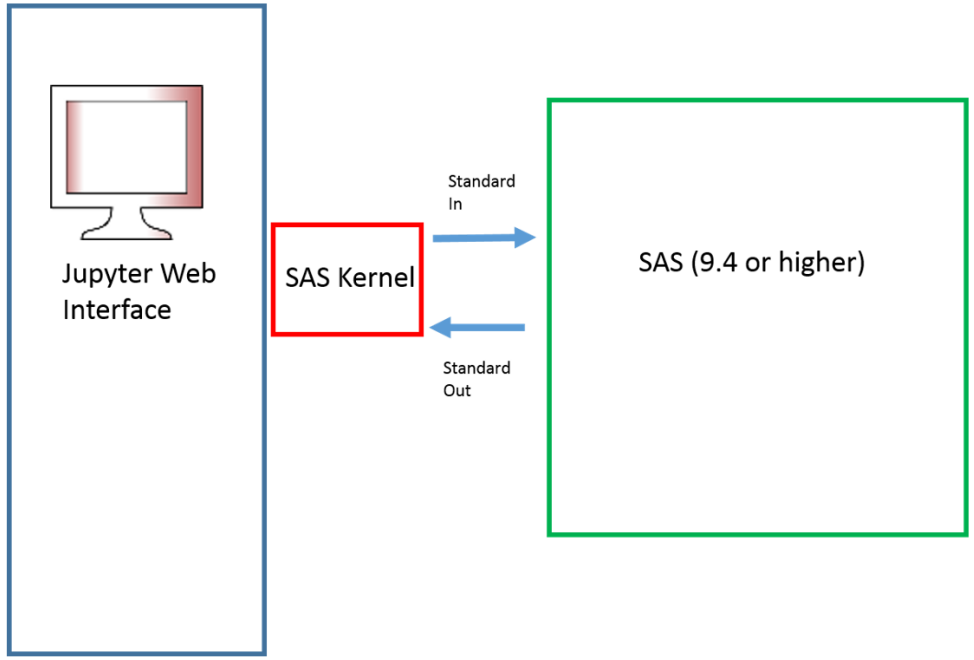


https://github.com/sassoftware/sas_kernel

What is the SAS Kernel?

- A Jupyter kernel for SAS. This opens up all the data manipulation and analytics capabilities of your SAS system within a notebook interface. Use the Jupyter Notebook interface to execute SAS code and view results inline.

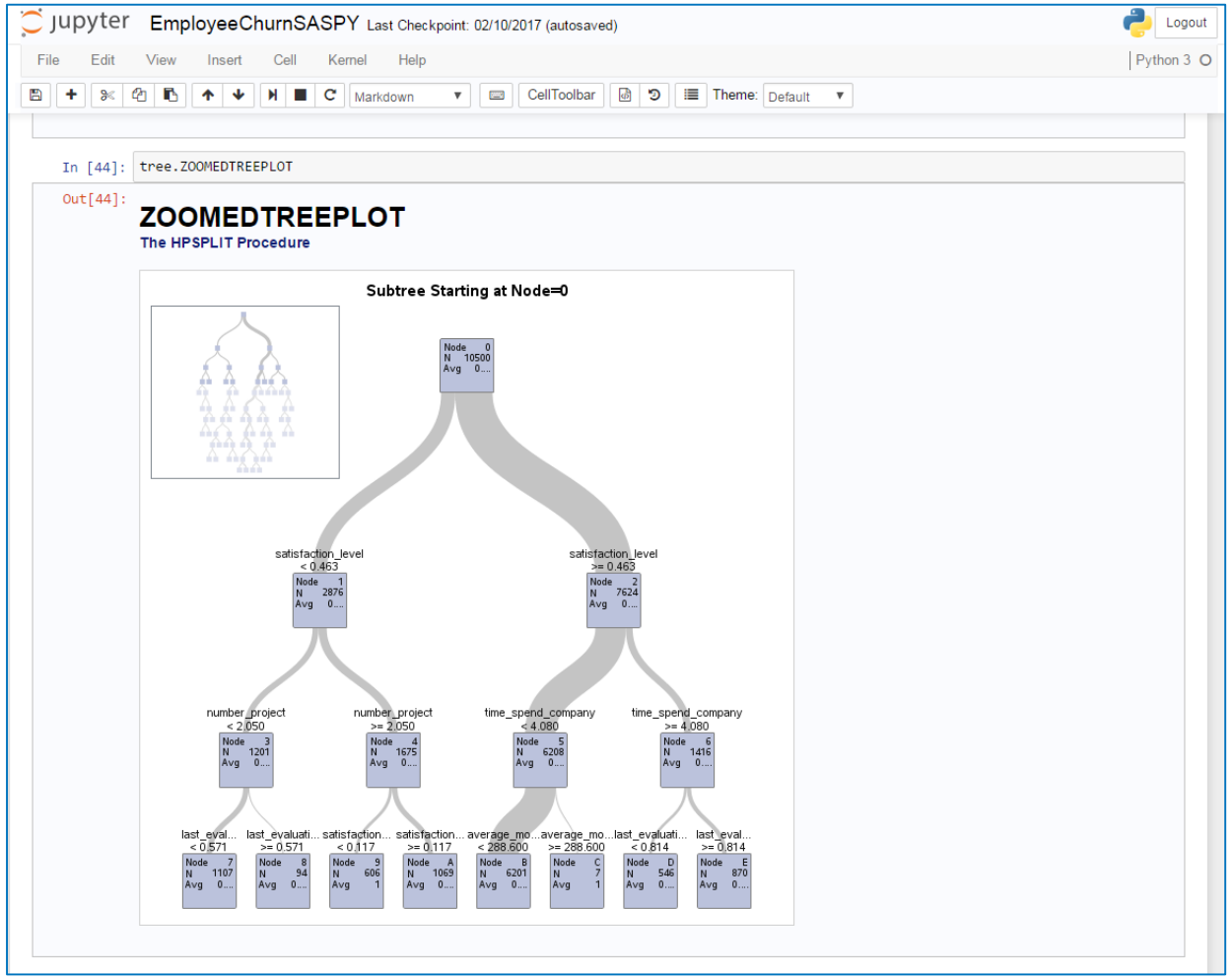
For SAS 9.4 and Python 3.X



https://github.com/sassoftware/sas_kernel

SASPy

- Provides Python APIs to the SAS system
- Run SAS code from Python
- Run analytics from Python using object-oriented methods
- Can transfer between SAS data sets and Pandas data frames



SASPy

```
In [13]: #import SAS package and start SAS Session
         from saspy import *
```

```
In [14]: #move the pandas dta to a SAS data object
         sas_dta=sas.dataframe2sasdata(dta, 'dta')
```

```
In [15]: sas_dta.head()
```

Out[15]:

SAS Output

The SAS System

The PRINT Procedure

Data Set WORK.DTA

Obs	rate_marriage	age	yrs_married	children	religious	educ	occupation	occupation_husb	affairs	affair
1	3	32	9.0	3	3	17	2	5	0.11111	1
2	3	27	13.0	3	1	14	3	4	3.23077	1
3	4	22	2.5	0	1	16	3	5	1.40000	1
4	4	37	16.5	4	3	16	5	5	0.72727	1
5	5	27	9.0	1	1	14	3	4	4.66667	1

SASPy

jupyter Untitled26 (unsaved changes)

File Edit View Insert Cell Kernel Help SAS

Code CellToolbar

Linear Regression

Can we predict the horsepower of a car based on several commonly available attributes

In [1]:
```
proc reg data=sashelp.cars plots=diagnosticpanel;
    model horsepower = length weight msrp mpg_city;
run;
```

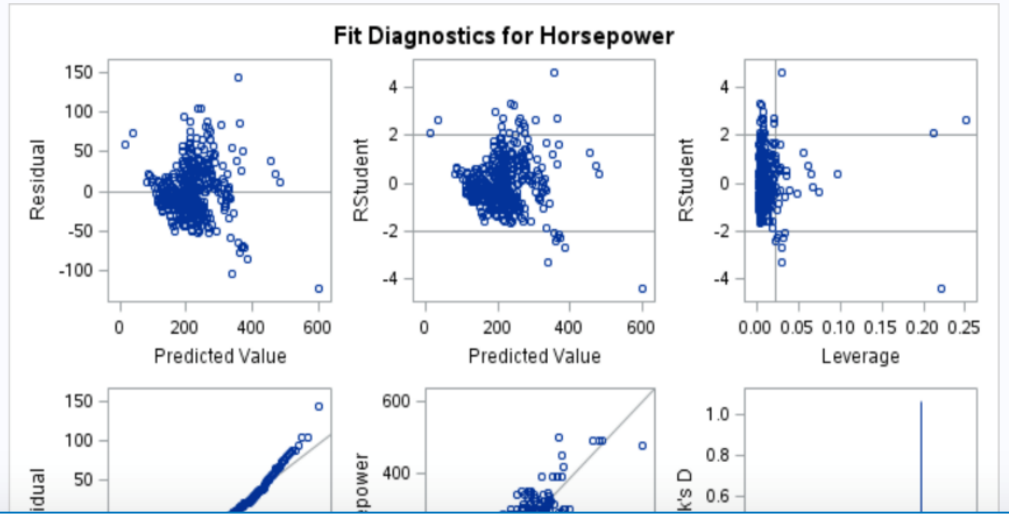
Out[1]:

The SAS System

The REG Procedure

Model: MODEL1

Dependent Variable: Horsepower



jupyter july31_python Last Checkpoint: 07/31/2015 (autosaved)

File Edit View Insert Cell Kernel Help Python 3

Code Cell Toolbar: None

In [7]:
```
cm = stat.reg(model='horsepower = length weight msrp mpg_city', data=cars)
```

...

In [8]:
```
cm.DIAGNOSTICSPANEL
```

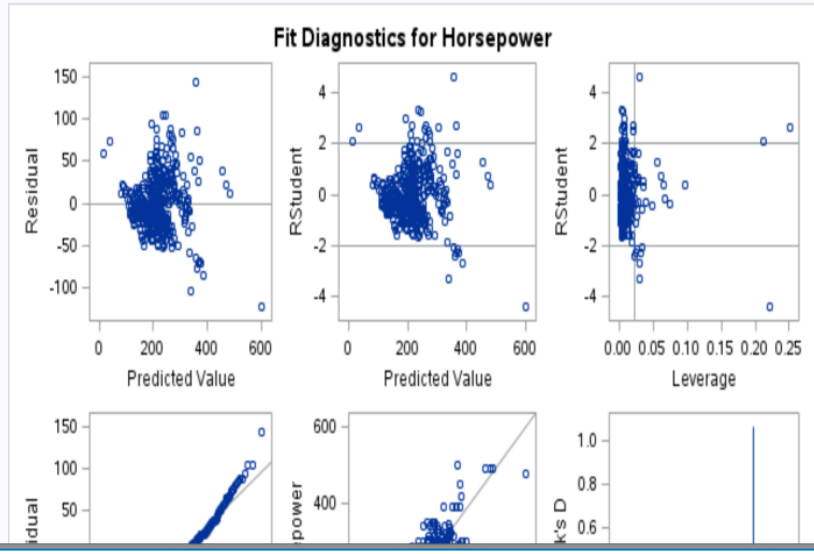
Out[8]:

The SAS System

The REG Procedure

Model: MODEL1

Dependent Variable: Horsepower



Viewing the SAS Code Behind SASPy

In [12]:

```
sas.teach_me_SAS(True)
sas_dta.hist('educ',title='Histogram of Education', label='Education Level')
sas.teach_me_SAS(False)
```

Lastly executed on Fri, Oct 30 2015 at 11:35 AM in 0 s

```
proc sgplot data=work.dta;
        histogram educ / scale=count LegendLABEL='Education Level';
        title "Histogram of Education";
        density educ;
run;
title "";
```

SASPy

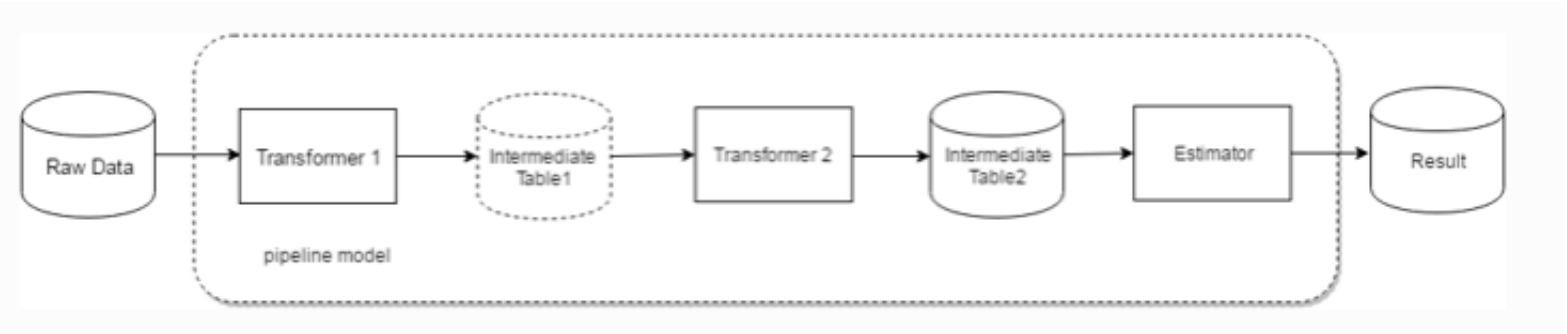
Technical Information

- The SASPy project provides Python APIs to the SAS system. You can start a SAS session and run analytics from Python through a combination of object-oriented methods and Python magics.
- **For SAS 9.4 and Python 3.x**
- Works using SAS Kernel on
  - LINUX
  - WINDOWS
  - UNIX

https://github.com/sassoftware/saspy

What is SAS Pipefitter

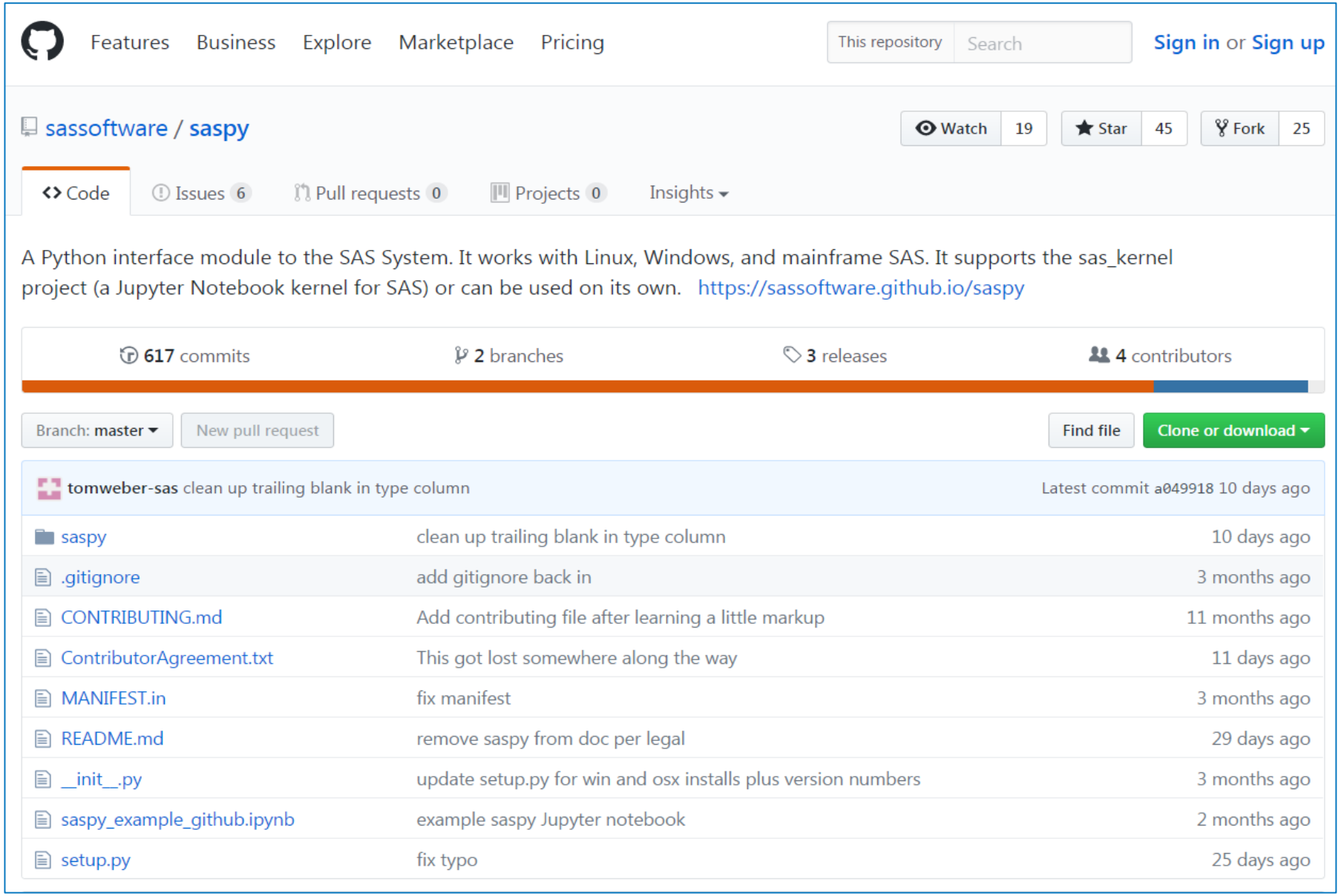
- The SAS pipefitter package provides a Python API for developing pipelines for data transformation and model fitting as stages of a repeatable workflow.



- Works with both SAS Viya (SWAT) and SAS 9.4 (SASPy)

https://sassoftware.github.io/python-pipefitter/overview.html

SASPy and Pipefitter Available on GitHub



Features Business Explore Marketplace Pricing

This repository Search

Sign in or Sign up

sassoftware / saspy

Watch 19 Star 45 Fork 25

Code Issues 6 Pull requests 0 Projects 0 Insights

A Python interface module to the SAS System. It works with Linux, Windows, and mainframe SAS. It supports the sas_kernel project (a Jupyter Notebook kernel for SAS) or can be used on its own. https://sassoftware.github.io/saspy

617 commits 2 branches 3 releases 4 contributors

Branch: master New pull request Find file Clone or download

tomweber-sas clean up trailing blank in type column Latest commit a049918 10 days ago

saspy	clean up trailing blank in type column	10 days ago
.gitignore	add gitignore back in	3 months ago
CONTRIBUTING.md	Add contributing file after learning a little markup	11 months ago
ContributorAgreement.txt	This got lost somewhere along the way	11 days ago
MANIFEST.in	fix manifest	3 months ago
README.md	remove saspy from doc per legal	29 days ago
__init__.py	update setup.py for win and osx installs plus version numbers	3 months ago
saspy_example_github.ipynb	example saspy Jupyter notebook	2 months ago
setup.py	fix typo	25 days ago

SAS Viya

SAS® Viya™

MULTIPLE INTERFACES, SINGLE CODE BASE

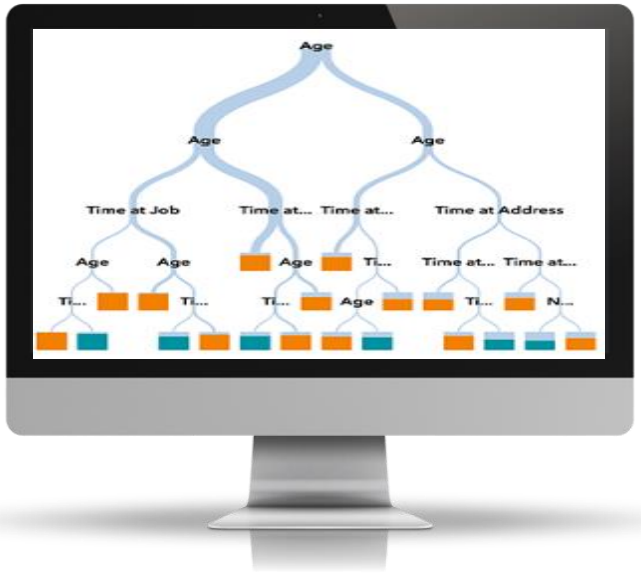
Visual Interfaces

Programming Interfaces

python™ sas Java R

API Interfaces

REST API



sas

SAS® Viya™

SAS Viya Products

- SAS Viya is an underlying foundation for additional products that will take advantage of this cloud-enabled, open platform. Most offerings include both a coding interface as well a visual interface.
  - SAS Visual Statistics
  - SAS Visual Data Mining and Machine Learning
  - SAS Visual Forecasting
  - SAS Optimization
  - SAS Econometrics
  - SAS Visual Investigator

SAS® Visual Data Mining and Machine Learning 8.1

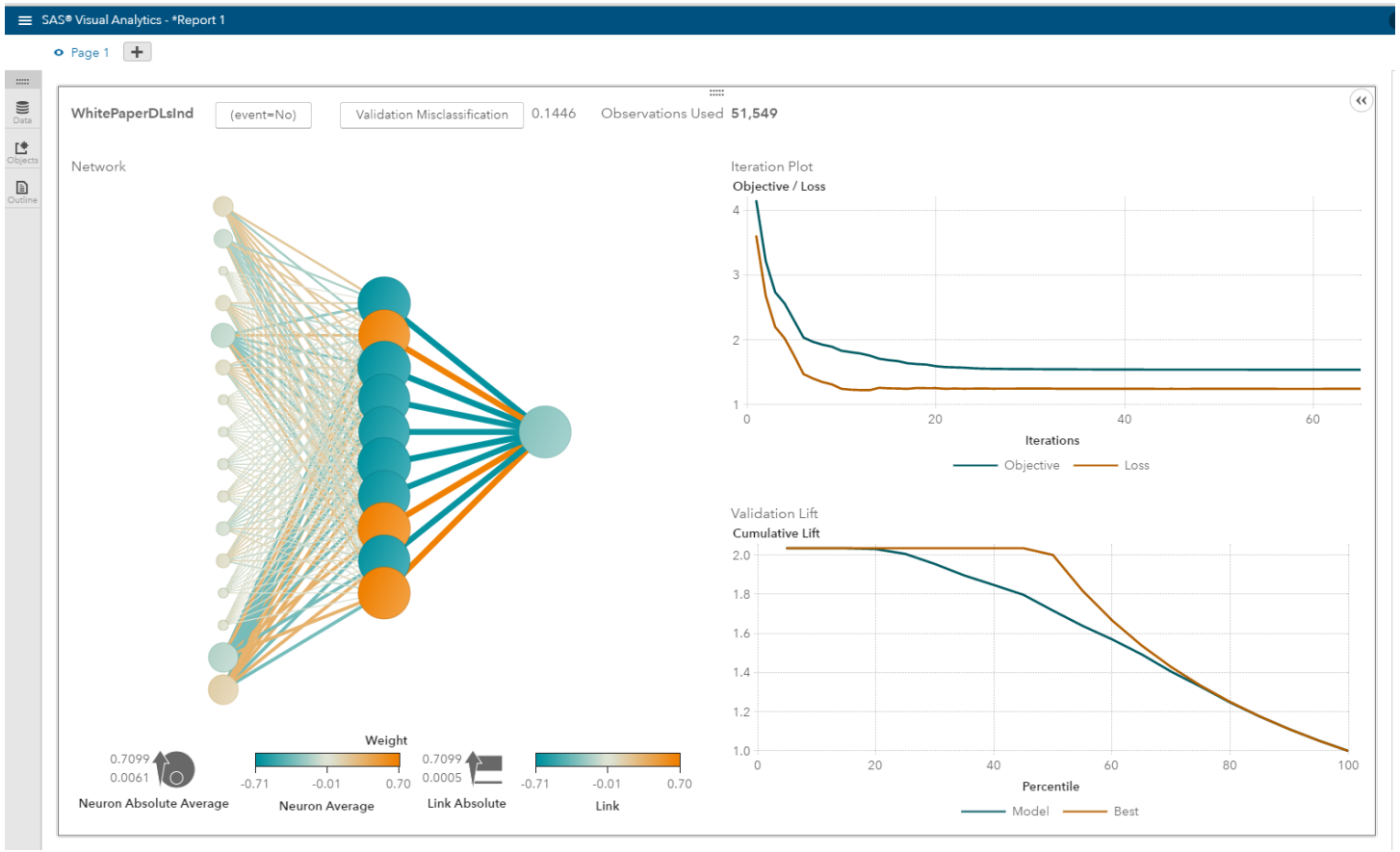
Visual Interface

Machine Learning Techniques

- Forest
- Factorization Machine
- Gradient Boosting
- Neural Network
- Support Vector Machine

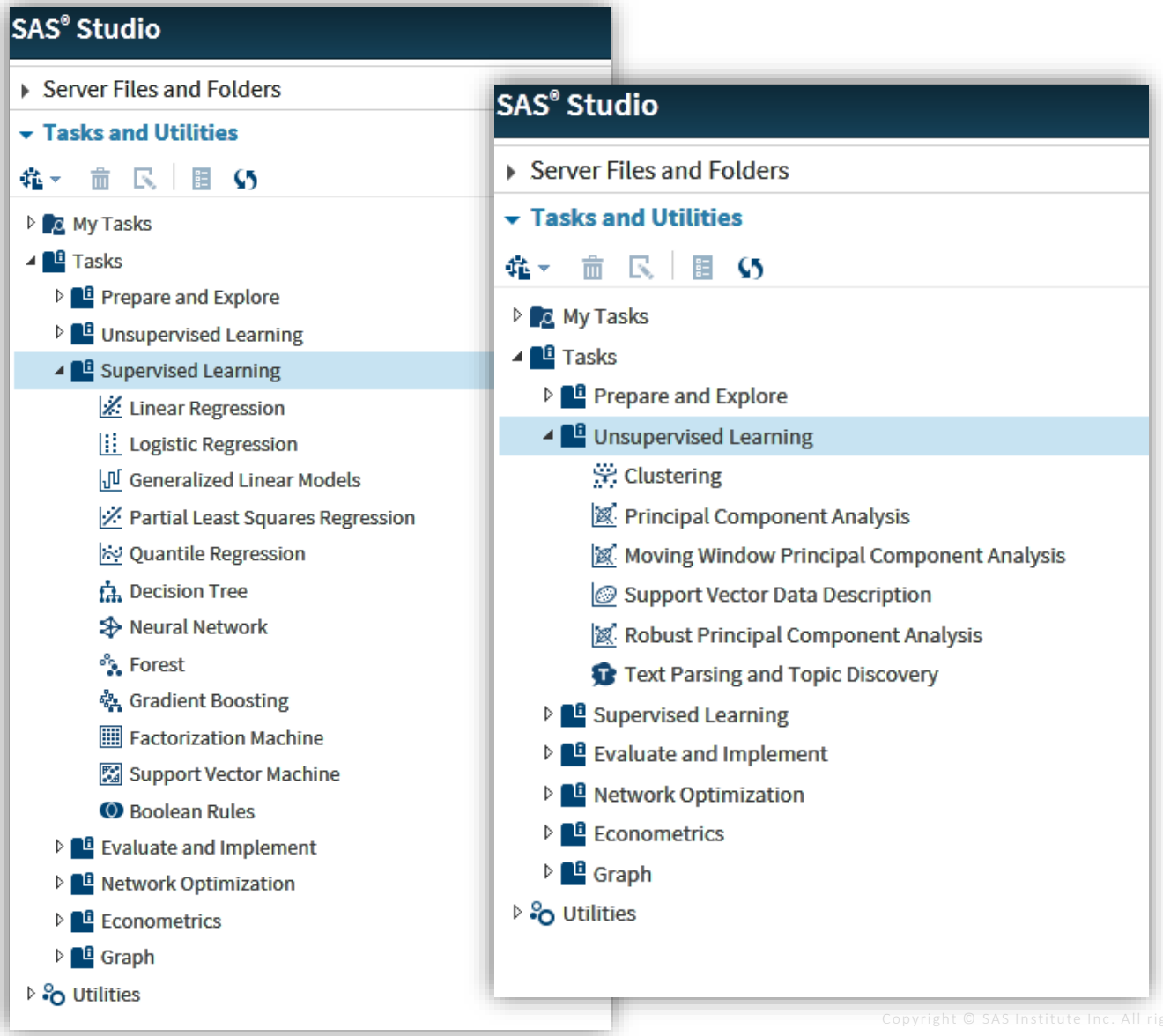
Common Features

- Training-Validation
- Model Assessment
- Model Comparison
- Score Code or Astore Table
- Ability to export model statistics into Excel



SAS Visual Data Mining and Machine Learning 8.1

Tasks in SAS Studio



Includes algorithms in the visual interface plus

- Unsupervised Learning
  - Moving Window PCA
  - Robust PCA
  - Support Vector Data Description
  - Text Parsing and Topic Discovery
- Supervised Learning
  - Boolean Rules

SAS® Visual Data Mining and Machine Learning

Programming with Open Source

jupyter Tree_RF_GBM_NN_SVM Example Last Checkpoint: 2 minutes ago (autosaved)

File Edit View Insert Cell Kernel Help Python 2

Markdown CellToolbar

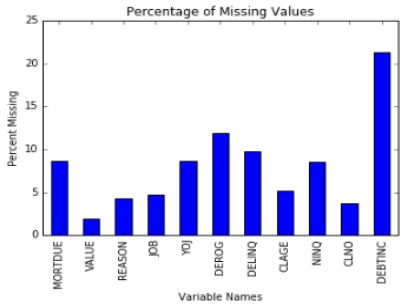
Explore and Impute missing values

Explore data and plot missing values

```
In [ ]:  .cardinality.summarize(
           table={"name":indata},
           cardinality={"name":"data_card", "replace":True}
         )

         tbl_data_card=sess.CASTable('data_card')
         tbl_data_card.where='_NMISS_>0'
         print("Data Summary".center(80, '-')) # print title
         tbl_data_card.fetch() # print obs

         tbl_data_card.vars=['_VARNAME_', '_NMISS_', '_NOBS_']
         allRows=20000  # Assuming max rows in data_card table is <= 20,000
         df_data_card=tbl_data_card.fetch(to=allRows)['Fetch']
         df_data_card['PERCENT_MISSING']=(df_data_card['_NMISS_']/df_data_card['_NOBS_'])*100

         tbl_forplot=pd.Series(list(df_data_card['PERCENT_MISSING']), index=list(df_data_card['_VARNAME_']))
         ax=tbl_forplot.plot(
           kind='bar',
           title='Percentage of Missing Values'
         )
         ax.set_ylabel('Percent Missing')
         ax.set_xlabel('Variable Names')
```

```
E: Writing cardinality.
OTE: status = 0.
NOTE: The Cloud Analytic Services server processed the request in 0.000874288 seconds.
      ----------------------------Data Summary----------------------------------
```

Out[4]: <matplotlib.text.Text at 0xe97f908>



SAS Visual Data Mining and Machine Learning with Python Demo
https://youtu.be/LXoikPWQJ3o

Visual Data Mining and Machine Learning

What do you get?

Visualizations

- Forest
- Gradient Boosting
- Neural Networks
- Support Vector Machines
- Factorization Machines

VDMML PROCS

- FOREST
- GRADBOOST
- NNET
- SVMACHINE
- FACTMAC
- TEXTMINE
- TMSCORE
- BOOLRULE
- ASTORE
- CAS
- NETWORK

VDMML CAS action sets

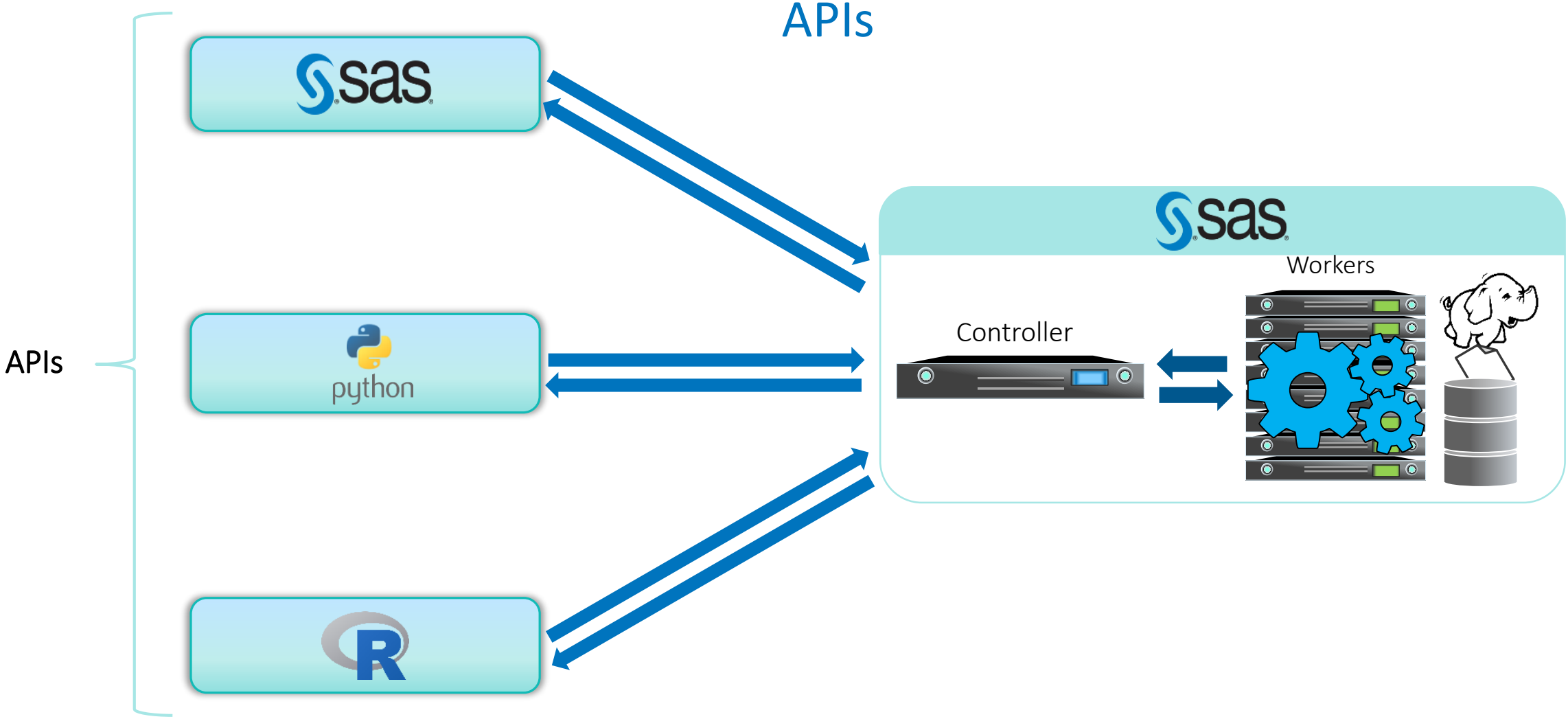
- MLEARNING
- TEXTMINE
- DMMLVISSET
- CRSBOOLRULE
- CRSNEURALNET
- CRSSVM
- CRSTKFACTMAC
- TKCAS
- CRSNETSOC
- CSRNETCOMMON
- CRSASTORE
- CRSCMPTRVSN
- CRSDTREEADVN
- CRSTXTMINADV

Visual Data Mining & Machine Learning PROCs and Action Sets

VDMML	Action Set	Action
ASTORE	astore	astore
BOOLRULE	boolRule	train, score
FACTMAC	factmac	factmac
FOREST	decisionTree optminer table sampling	foresttrain, forestscore, forestcode tuneRandomForest drop srsact
GRADBOOST	decisionTree table sampling	gbtreetrain, gbtreescore drop srsact
NETWORK	networkCommon networkSocial	shortestPath centrality, community
NNET	neuralNet optminer	anntrain, annscore, anncode tuneNeuralNet
SVMACHINE	svm	svmtrain
TEXTMINE	textparse textMining	Parse, accumulate textMine, parse, accumulate, svd
TMSCORE	textMining	tmScore, parse

SAS Viya

APIs



SAS Viya

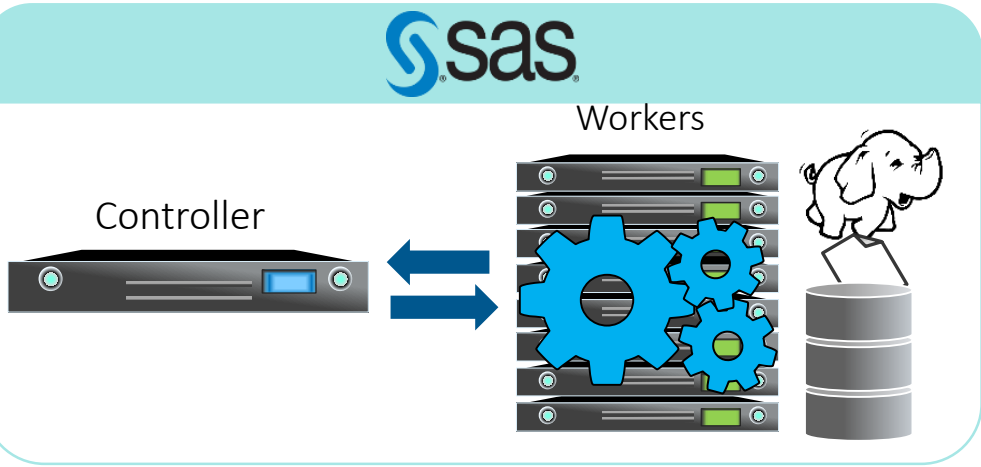
An Example

APIs

```
proc print data = hmeq (obs = 10);
run;
```

```
df = s.CASTable('hmeq')
df.head(10)
```

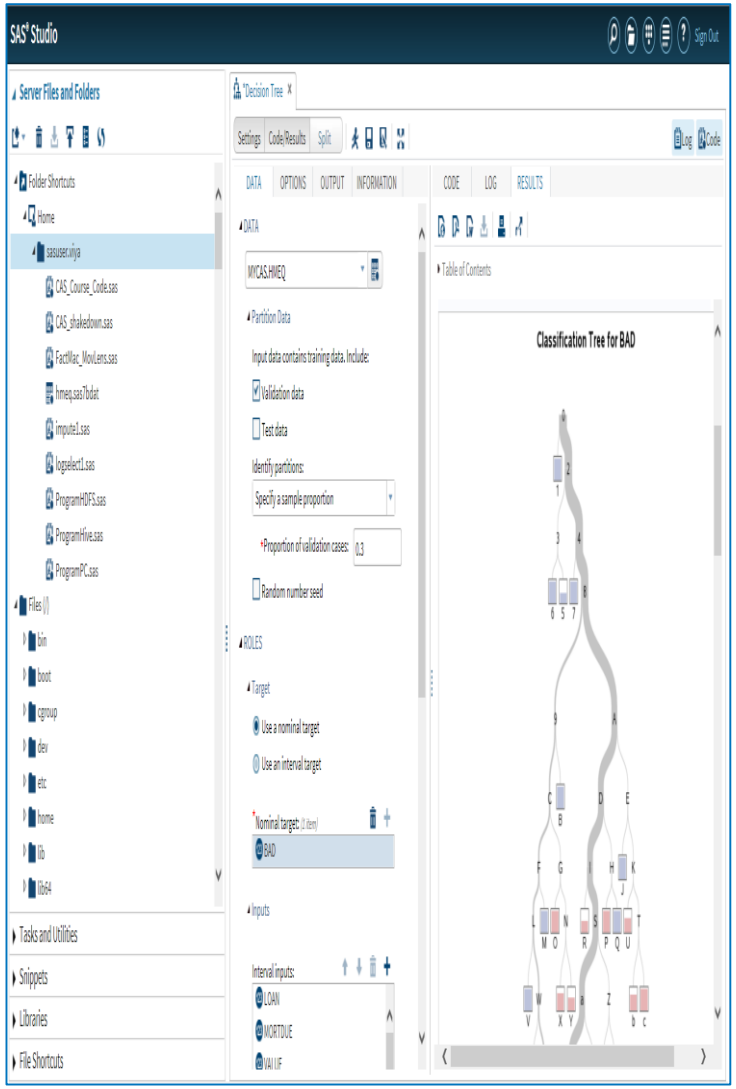
```
df <- defCasTable(s, 'hmeq')
head(df, 10)
```

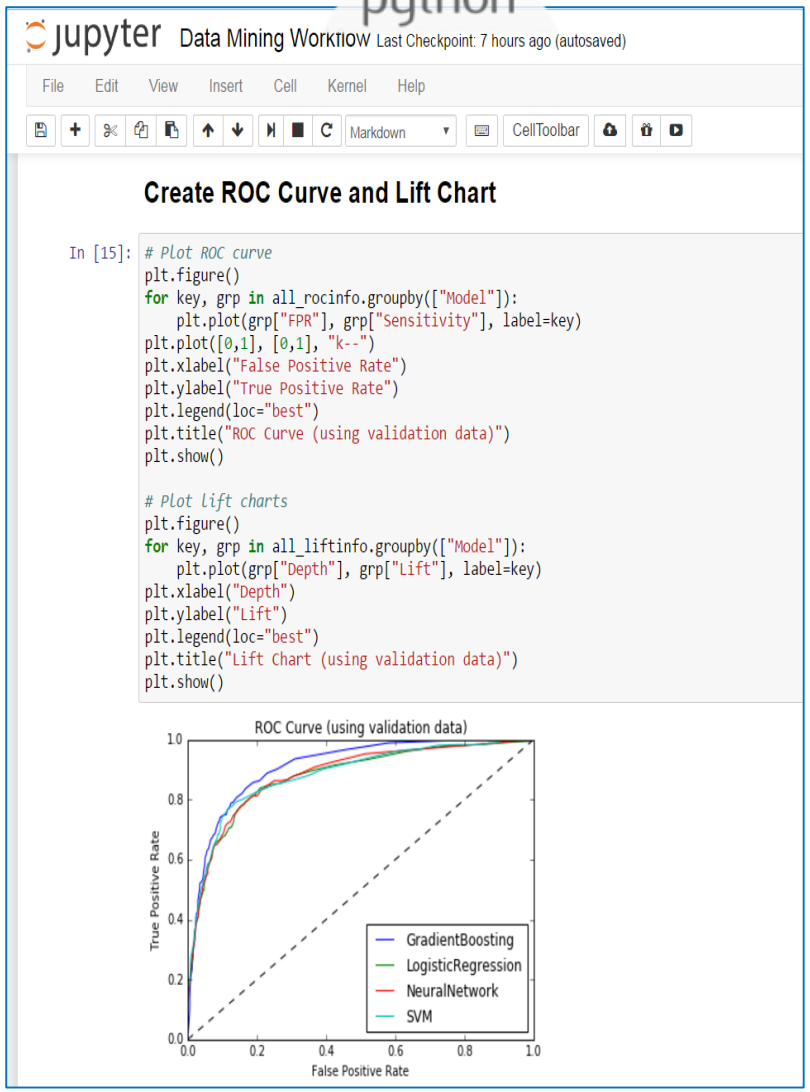


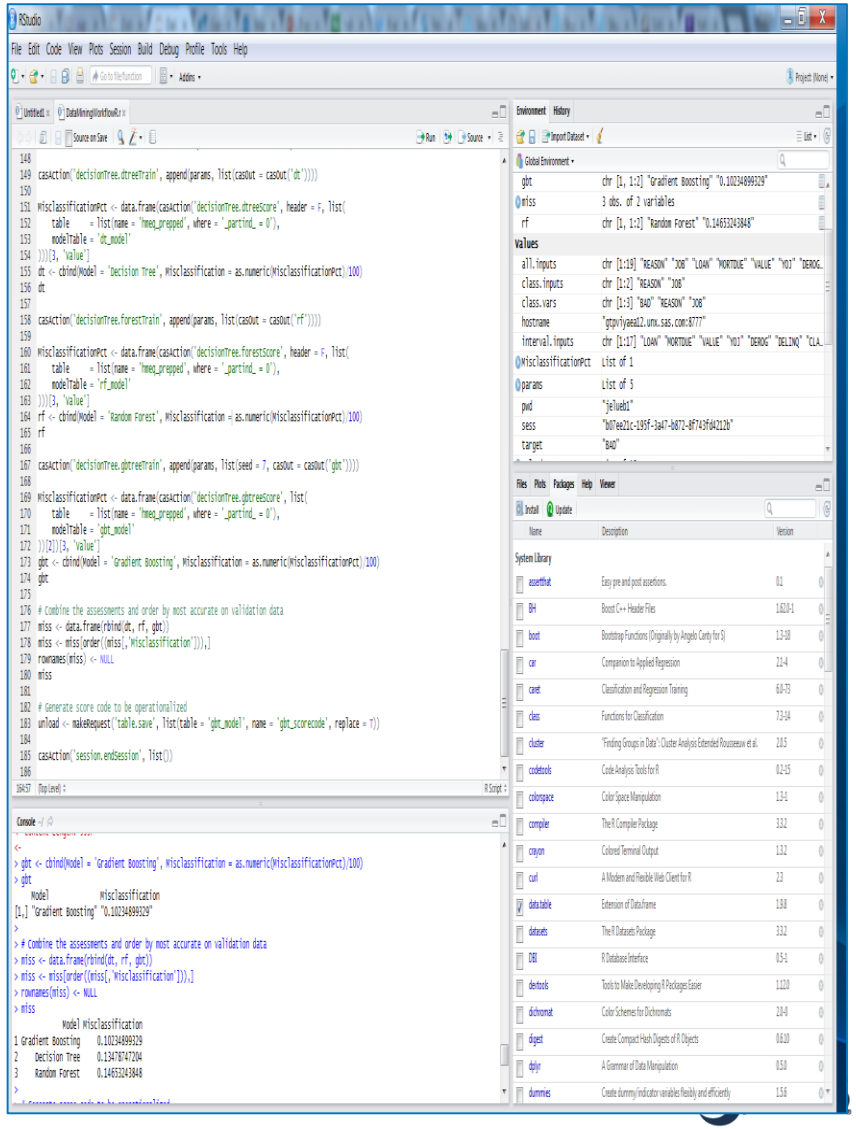
Translated Command

```
[table.fetch]
    table.name = "hmeq"
    from = 1 to = 10
```

SAS Viya



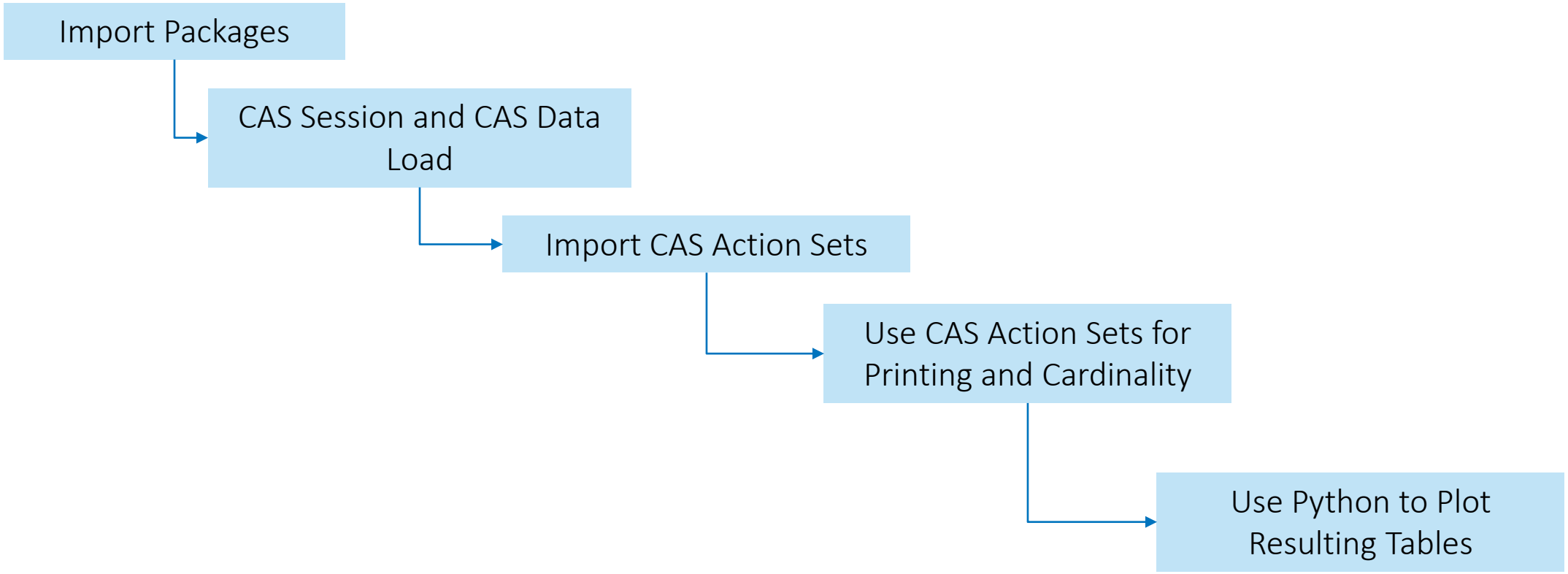




SAS Scripting Wrapper for Analytics Transfer (SWAT)

- SWAT packages are available for Python, R and Lua free on GitHub or developer.sas.com.
- Download and install SWAT, connect to a CAS server, then write code to drive CAS actions.
- The SWAT package mimics much of the APIs of the native packages making it an easy addition for programmers familiar these languages.

An Example Flow for Python SWAT Interface to CAS



Open architecture in Action

Example Python Code using SWAT

SAS Wrapper for Analytics Transfer (SWAT)

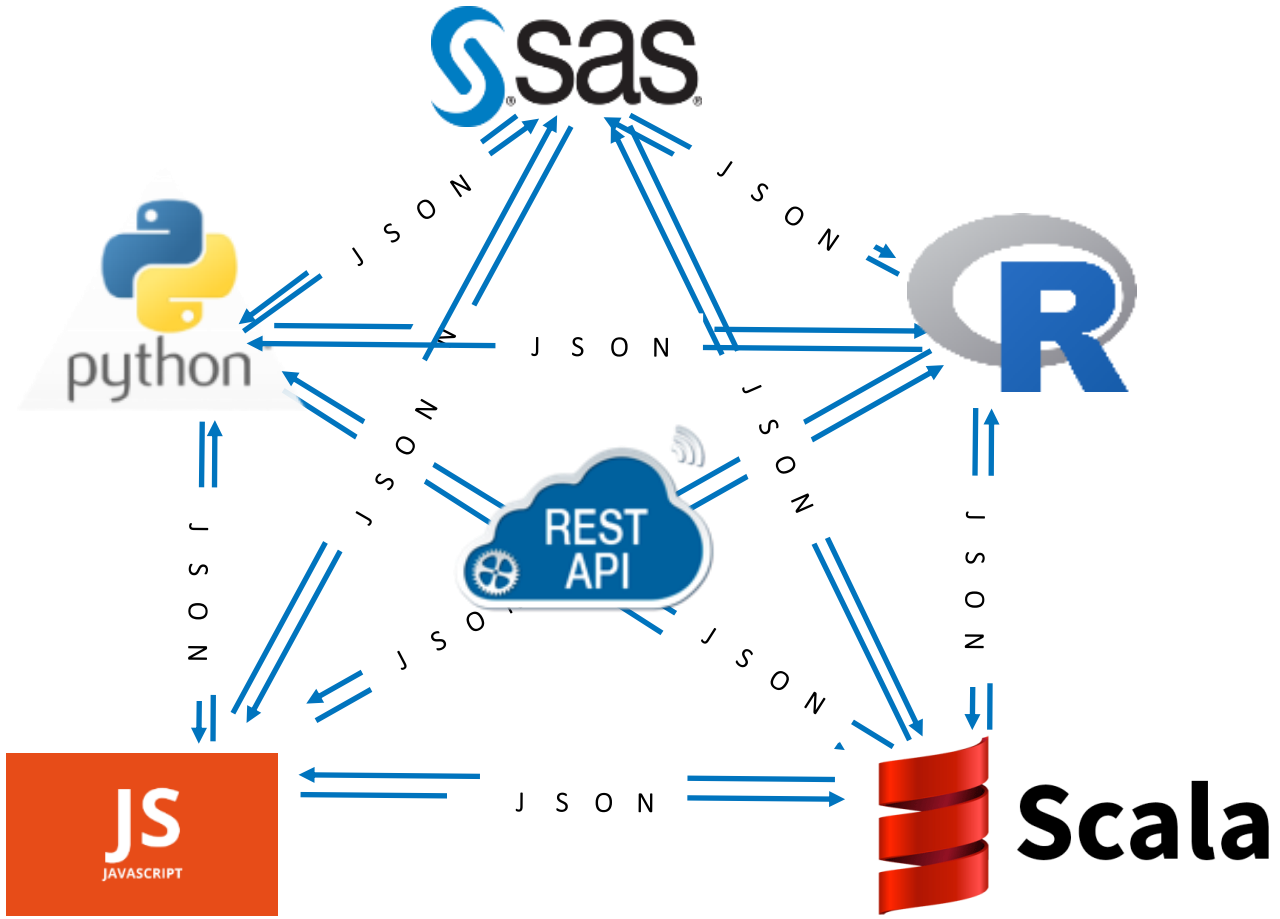
```
In [1]: import swat
In [2]: conn = swat.CAS(host, port, userid, password)
In [3]: out = conn.serverstatus()

In [10]: conn.help(actionset='simple');
In [11]: tbl = conn.read_csv('https://raw.githubusercontent.com/‘
    ....: 'sassoftware/sas-viya-programming/master/data/cars.csv')
    ....:
In [12]: out = conn.summary(table=tbl)
In [14]: out = tbl.summary()
In [16]: summ = out.Summary
In [17]: summ = summ.set_index('Column')
In [18]: summ.loc['Cylinders', 'Max']
```

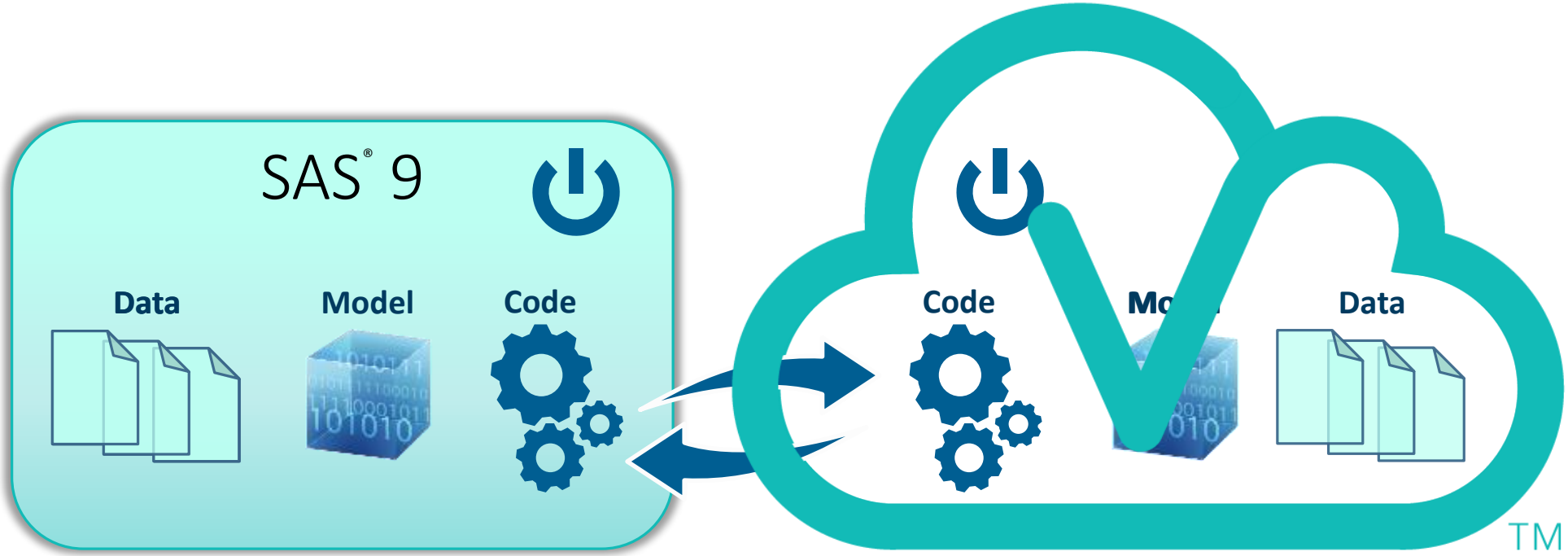
Special SAS Python package

Look and feel is designed to mimic pandas.read_csv

Native language bindings also used here

REST APIs



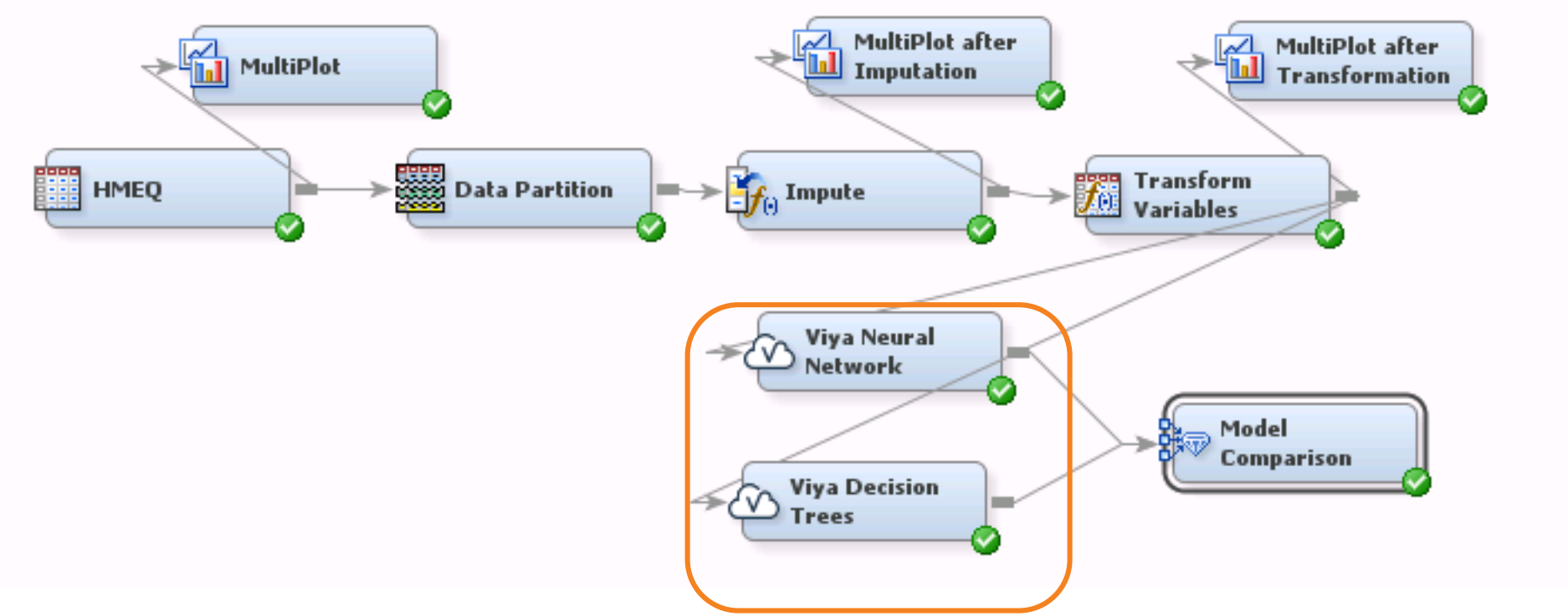


Bridging SAS® 9 and SAS® Viya™

- SAS® 9 data accessible within SAS® Viya™
- SAS® Viya™ models publishable to SAS® 9
- Remotely execute code from / to both
- More bridges planned

Integrating Viya and SAS Enterprise Miner

Viya Code Node



Resources

SAS and Open Source Resources
Empowering the SAS Enterprise Miner user

Video: *Using R in SAS Enterprise Miner*
https://www.youtube.com/watch?v=TbXo0xQCqDw

Blogs: *Spectral Clustering in SAS® Enterprise Miner™ Using Open Source Integration Node*
https://communities.sas.com/docs/DOC-8011

Blogs: *How to execute a Python script in SAS® Enterprise Miner™*
https://communities.sas.com/docs/DOC-10832

Article: *The Open Source Integration node installation cheat sheet*
https://communities.sas.com/docs/DOC-9988

Usage Notes:
http://support.sas.com/dsearch?Find=Search&ct=&qt=open+source&col=suppprd&nh=25&qp=&qc=suppsas&ws=1&qm=1&st=1&lk=1&rf=0&oq=&rq=0

SAS and Open Source Resources
Empowering the SAS/IML user

Documentation: *IML User's Guide - Calling Functions in the R Language*
http://support.sas.com/documentation/cdl/en/imlug/66845/HTML/default/viewer.htm#imlug_r_toc.htm

Video: *Calling R Procedures from SAS/IML® Software*
https://www.youtube.com/watch?v=rUaTTre24kI

Video: *SAS/IML and R: Using Them Together*
https://www.youtube.com/watch?v=nmRQ3MtkG6A

Blogs: *The DO Loop – R tags*
http://blogs.sas.com/content/iml/tag/r/

Paper (p 14-17): *Rediscovering SAS/IML® Software: Modern Data Analysis for the Practicing Statistician*
http://support.sas.com/resources/papers/proceedings10/329-2010.pdf

Article: *Versions of R that are supported by SAS/IML*
http://blogs.sas.com/content/iml/2013/09/16/what-versions-of-r-are-supported-by-sas.html

SAS and Open Source Resources
Empowering the Base SAS user

- **Blogs:** *Open Source Integration Using the Base SAS Java Object* *https://communities.sas.com/docs/DOC-10746*

- **Github Page:**
  - *SAS_Base_OpenSrcIntegration https://github.com/sassoftware/enlighten-integration/blob/master/SAS_Base_OpenSrcIntegration/main_caller.sas*
  - *SASPy https://github.com/sassoftware/saspy*
  - *Python-pipefitter https://github.com/sassoftware/python-pipefitter*

- **SAS Community tips (Cheat sheet for version numbers – R/PMML/Linux)** https://communities.sas.com/t5/SAS-Communities-Library/The-Open-Source-Integration-node-installation-cheat-sheet/ta-p/223470

Online. Everyday.

"I always learn something new when I post in this forum. Just what I needed…"

SAS Online Community

Communities.sas.com/data-mining

Questions?

Thank you for your time and attention!

sas.com