Introduction to SAS® and Open Source

SAS 9 and Viya

Melodie Rush
Global Customer Success Principal Data Scientist

Connect with me:
LinkedIn: https://www.linkedin.com/in/melodierush
Twitter: @Melodie_Rush

THE POWER TO KNOW.

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
  - Support for models created with R and Python delivered with SAS Viya 3.4 in 2018
- 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
- Base SAS - PROC LUA
  - SAS 9.4 Maintenance 3 (released July 2015)
- 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

Python Functions

- SAS 9.4M6 May 2019
- Using Python functions inside of SAS Programs

SAS Environment

```
*** Call Python;
 data _null_;
     python_pgm = "&WORK_DIR.\mppr_python.py";
     python_arg1 = "&WORK_DIR";
     python_exec_command = "&python_exec_command";
     python_call = cat('"', trim(python_exec_command), '" "', trim(python_pgm),
       '" "', trim(python_arg1), '"');
     put python_call= ;
     call system(python_call);
   run;
```

Base SAS DATA step

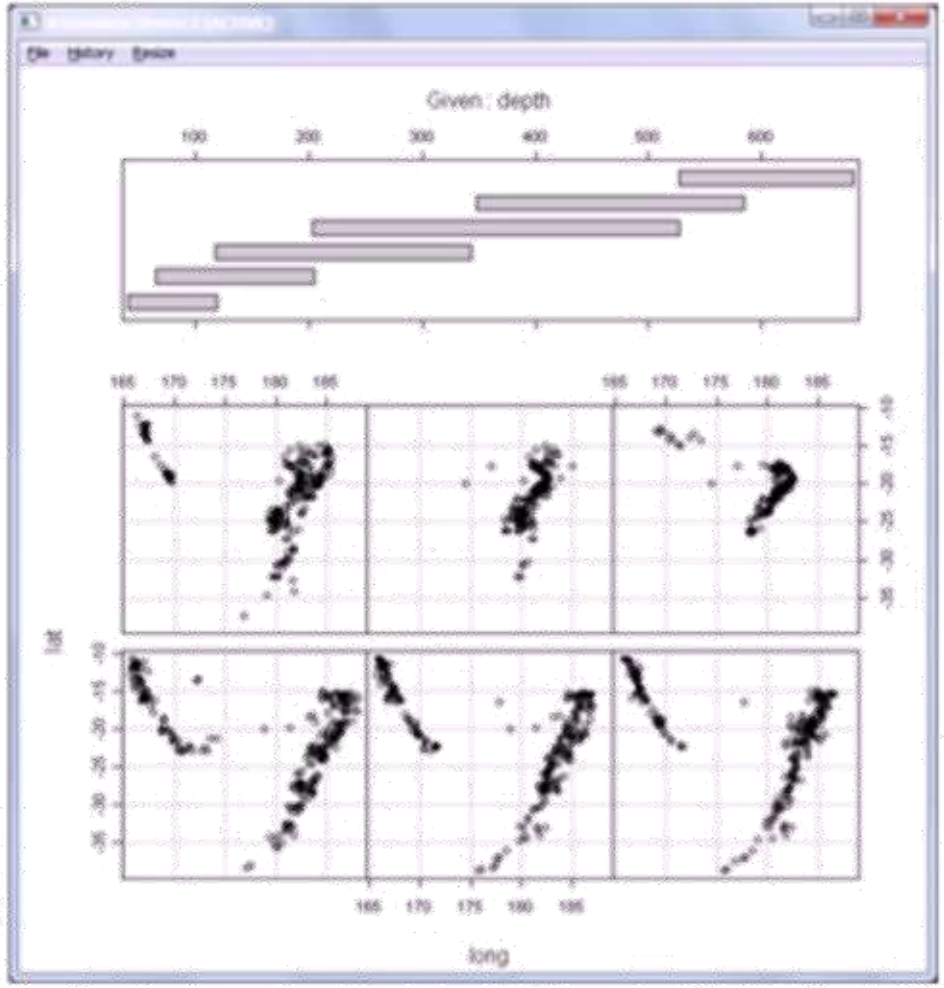
SAS and Open Source

SAS/IML

```
proc iml;
submit / R;
   coplot(lat ~ long | depth,
          data = quakes)
endsubmit;
```

Can run code in SAS Enterprise Guide, SAS Enterprise Miner, SAS Studio and SAS IML Studio (need IML licensed and installed)

Supported Since SAS 9.22



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;
```

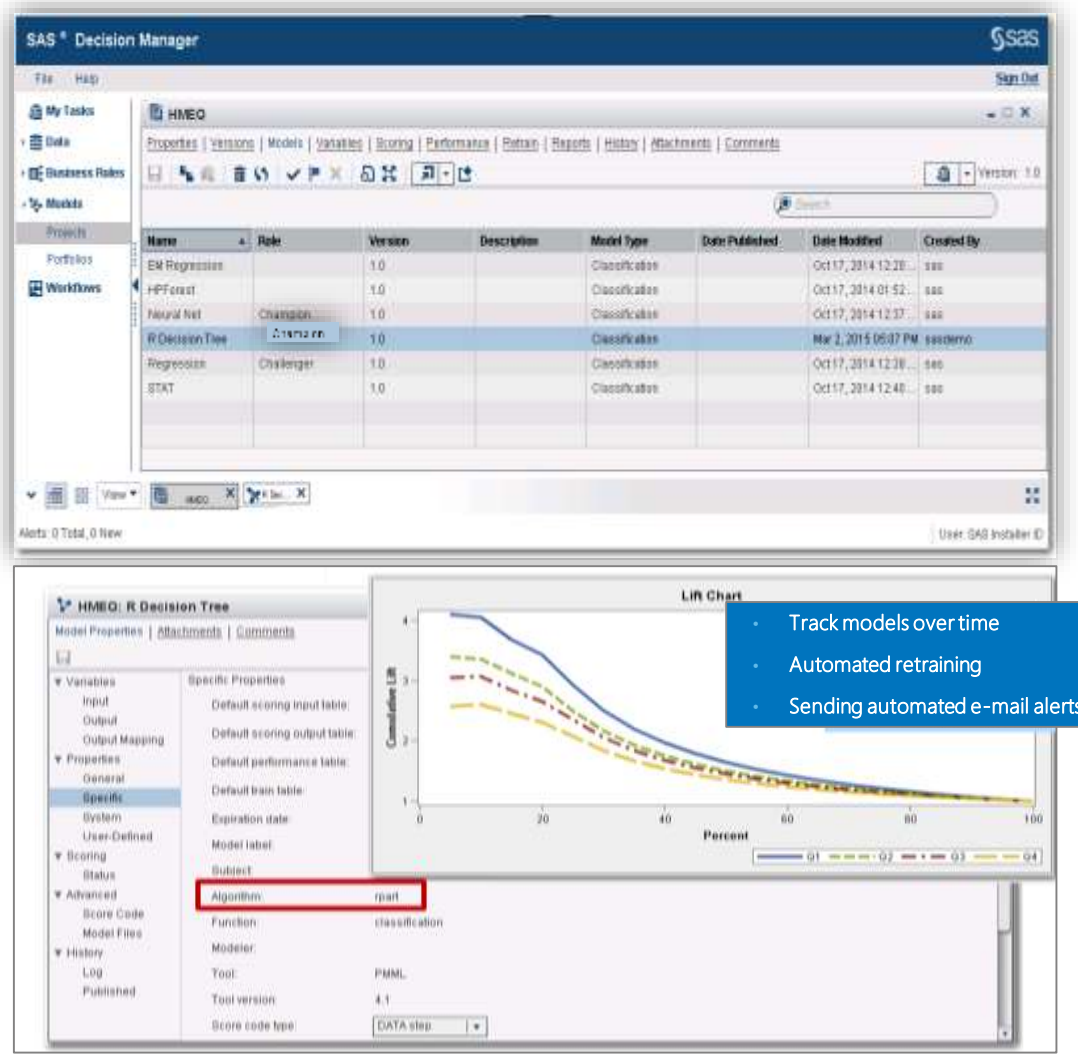
4. Transfer results into SAS/IML vectors

```
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;
```

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



SAS Enterprise Miner and R

R integration

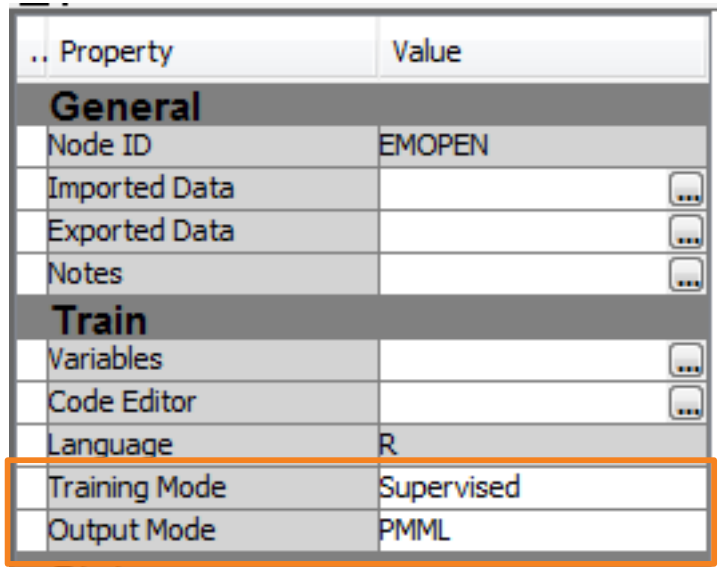
SAS® Enterprise Miner

- 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



Property	Value
General	
Node ID	EMOPEN
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Code Editor	
Language	R
Training Mode	Supervised
Output Mode	PMML

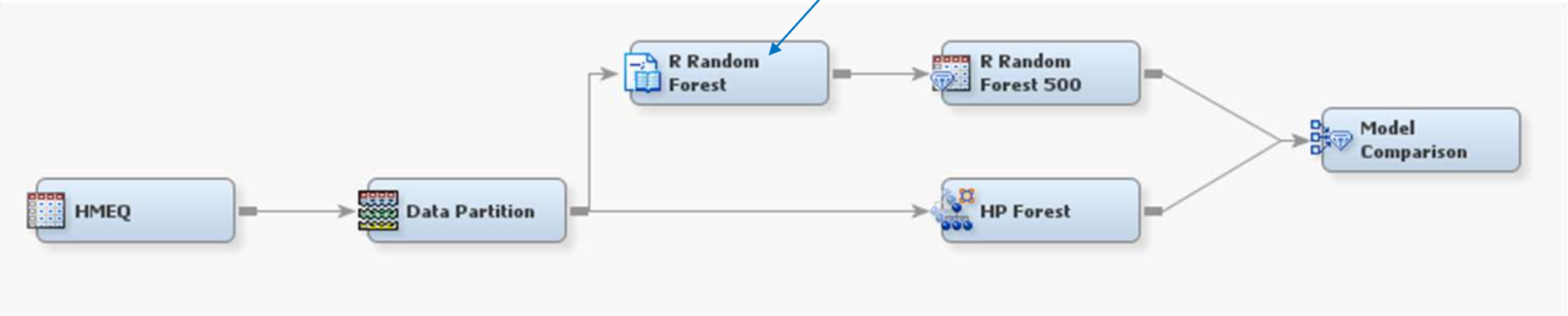
- Output Mode
  - PMML: Creates SAS Data step score code
  - Merge: Merge inputs with predictions
  - None: Troubleshooting R code, output graphs, simulations etc

SAS and Open Source

SAS Enterprise Miner

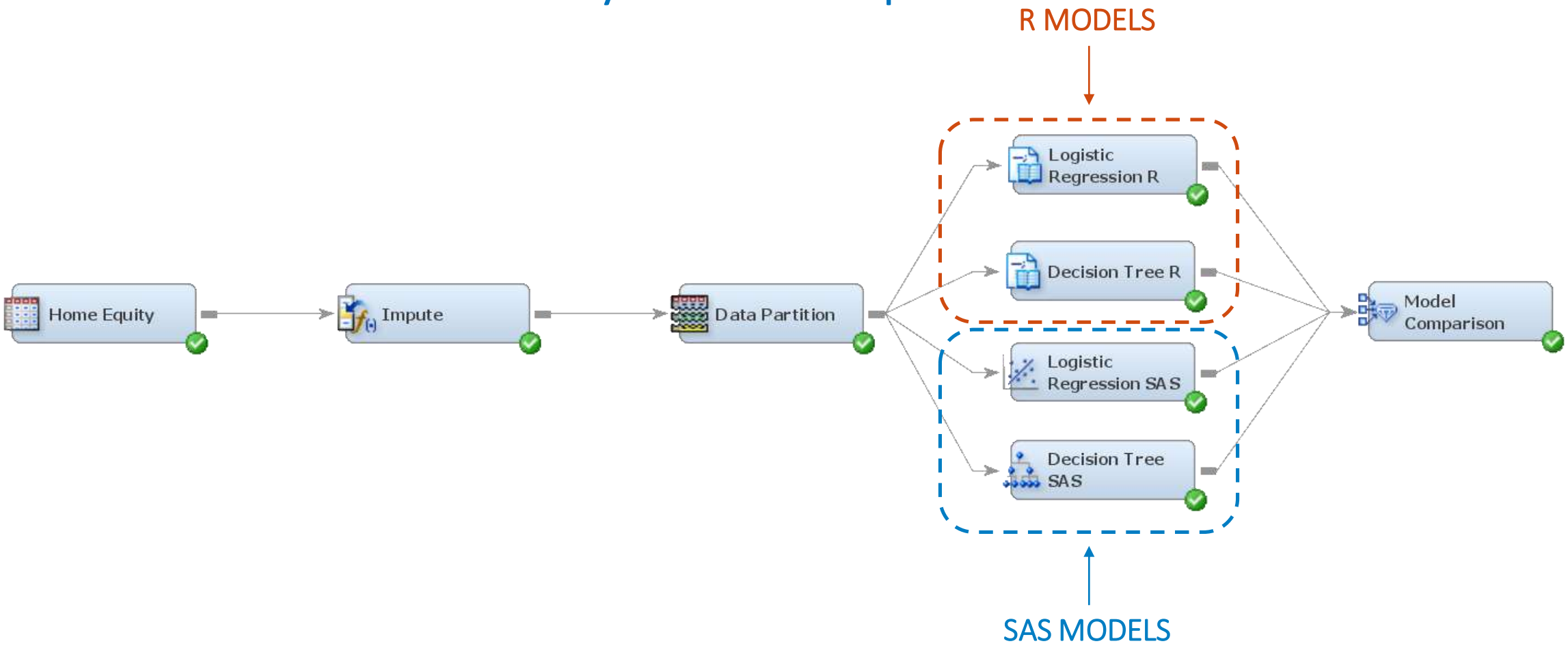
- Integrate R code inside Enterprise Miner
  - Includes R models in model assessment
  - SAS and R ensemble models

```
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



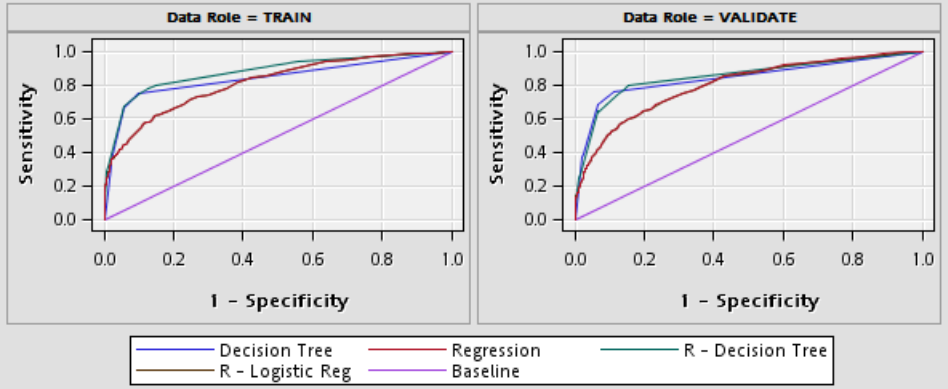
SAS and Open Source

SAS and R Model Comparison

Results - Node: Model Comparison Diagram: HMEQ - Open Source

File Edit View Window

ROC Chart : BAD

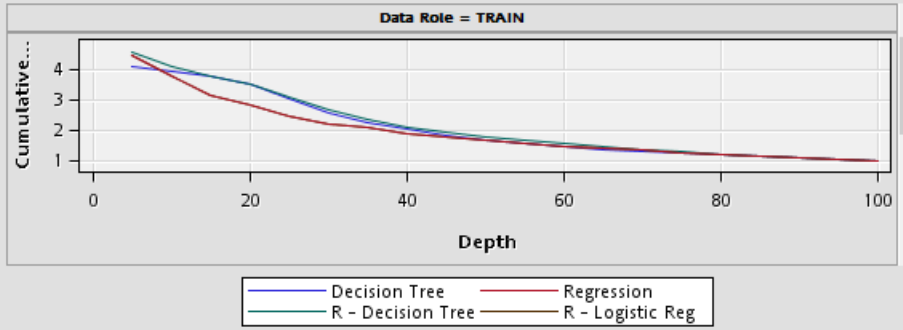


Fit Statistics

Selected Model	Predecessor Node	Model Node	Model Description	Target Variable	Target Label	Selection Criterion: Valid: Misclassification Rate	Train: Sum of Frequencies
Y	Tree	Tree	Decision ...	BAD		0.118189	357
	EMOPEN3	EMOPEN3	R - Decis...	BAD		0.127829	357
	EMOPEN	EMOPEN	R - Logis...	BAD		0.166387	357
	Reg	Reg	Regressi...	BAD		0.166387	357

Score Rankings Overlay: BAD

Cumulative Lift



Output

```
*-----------------------------------------------------------*
User:                    sasmwr
Date:                    February 06, 2017
Time:                    16:43:46
*-----------------------------------------------------------*
* Training Output
*-----------------------------------------------------------*




Variable Summary

          Measurement     Frequency
```

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

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

SAS Enterprise Miner metadata

```
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

SAS Enterprise Miner metadata

```
library(rpart)

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

R object translated to SAS DATA step code using PMML

SAS Enterprise Miner data source

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

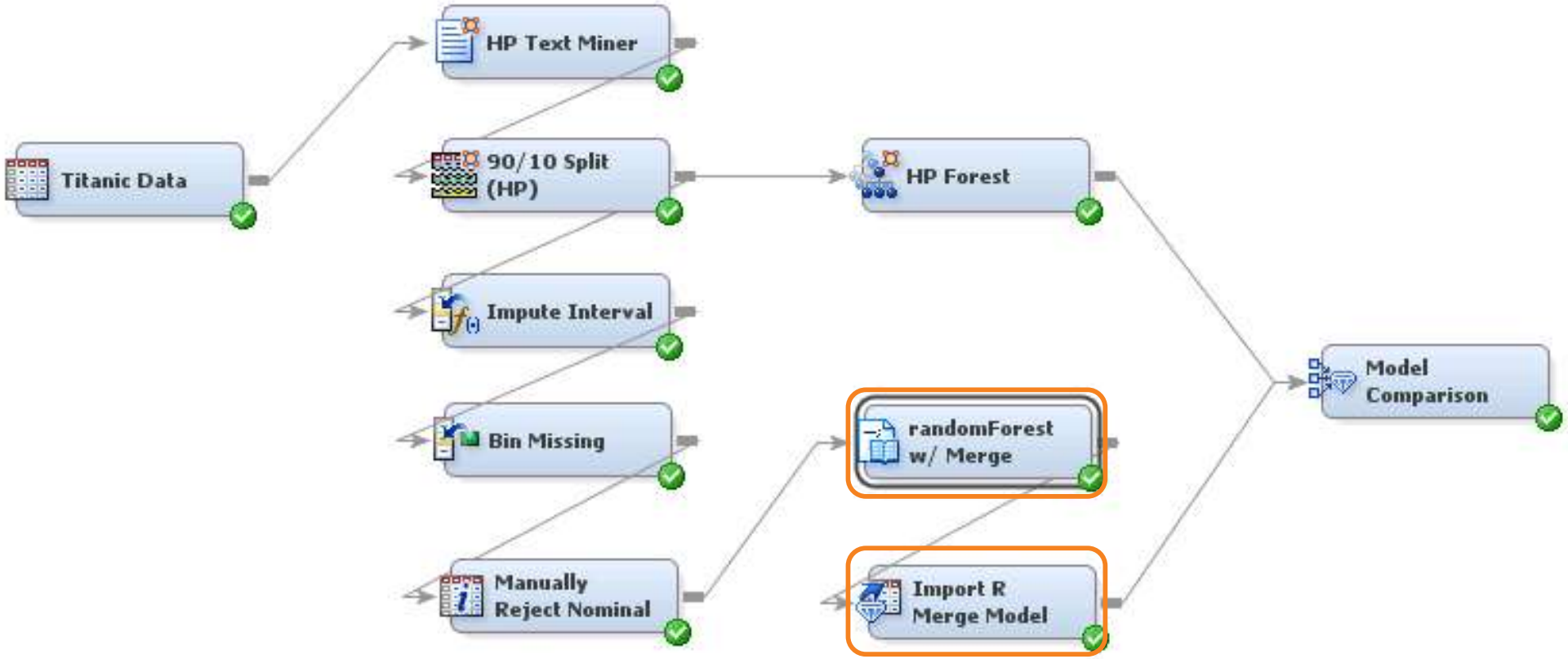
Variables created in R are merged with SAS Enterprise Miner data sources **by the user**

SAS DATA step code is not created

Property	Value
General	
Node ID	EMOPEN3
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
Code Editor	...
Language	R
Training Mode	Supervised
Output Mode	Merge

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

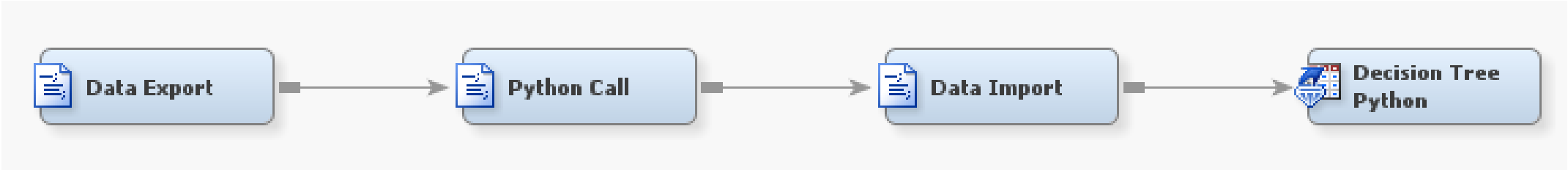
User must explicitly create exported variables to be merged with Enterprise Miner data sources

SAS Enterprise Miner and R

Demo

SAS Enterprise Miner and Python

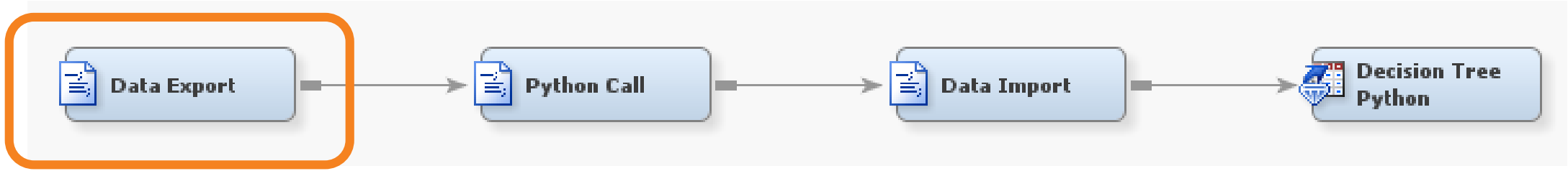
Python Example



- The code in SAS Code Nodes:
  - Export the data to .csv files
  - Execute a Python Program
  - Import the predictions and merge with training dataset

Python Example

First SAS Code Node

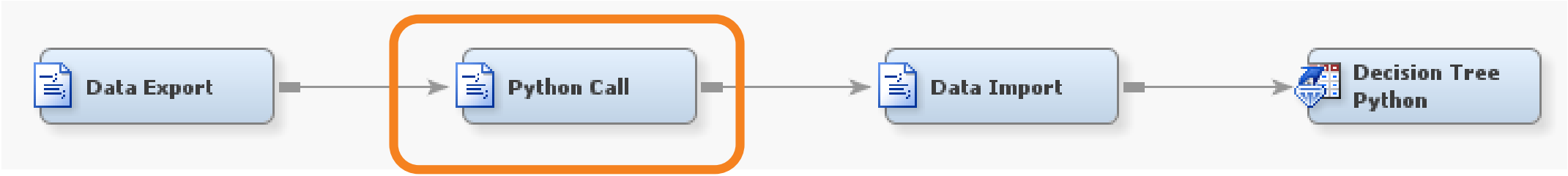


The code in Data Export SAS Code Node (repeat for the validation and test data sets):

```
data train;
 set &EM_IMPORT_DATA (keep = %EM_INPUT %EM_TARGET);
  format _all_;
run;
 proc export data=train
    outfile = "&WORK_DIR.\train.csv"
  dbms = csv
  replace;
run;
```

Python Example

Second Code Node



The code in Python Call SAS Code Node:

```
*** Call Python;
 data _null_;
    python_pgm = "&WORK_DIR.\hmeq_python_dt.py";
    python_arg1 = "&WORK_DIR";
    python_exec_command = "&python_exec_command";
    python_call = cat('"', trim(python_exec_command), '" "', trim(python_pgm), '" "', trim(python_arg1), '"');
    put python_call= ;
    call system(python_call);
  run;
```

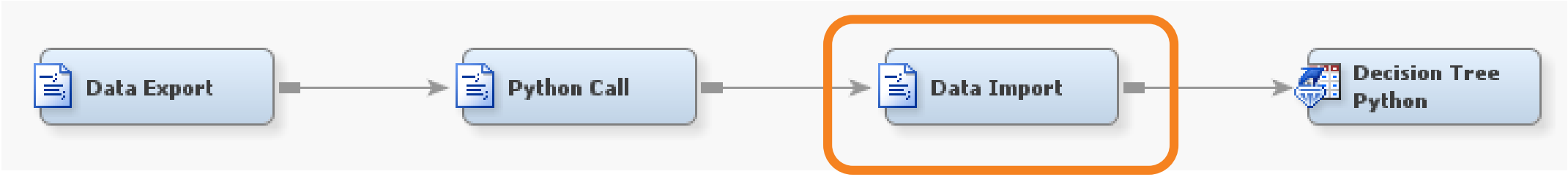
Python Example

hmeq_python_dt.py

```
import csv, os, sys
import pandas as pd
from sklearn.tree import DecisionTreeClassifier

#read data into panda data frames
train = pd.read_csv(os.path.join(sys.argv[1], "train.csv"))
validate = pd.read_csv(os.path.join(sys.argv[1], "validate.csv"))

#set up target and features list
train_target = train['BAD']
train_features = train.drop('BAD', axis=1)
validate_features = validate.drop('BAD', axis=1)

#run decision tree
dt = DecisionTreeClassifier(criterion='gini',
    max_depth=5, min_samples_leaf=5,
    random_state=31415)
dt.fit(train_features,train_target)

#score training and validation data
train_predictions = dt.predict_proba(train_features)
validate_predictions = dt.predict_proba(validate_features)

#export predictions to .csv files
train_predictions_file = open((os.path.join(sys.argv[1],
    "hmeq_python_train_predictions_dt.csv")), "w" , newline="")
open_file_object = csv.writer(train_predictions_file)
open_file_object.writerow(["p_BAD0","p_BAD1"])
open_file_object.writerows(train_predictions)
train_predictions_file.close()

validate_predictions_file = open((os.path.join(sys.argv[1],
    "hmeq_python_validate_predictions_dt.csv")), "w", newline="")
open_file_object = csv.writer(validate_predictions_file)
open_file_object.writerow(["p_BAD0","p_BAD1"])
open_file_object.writerows(validate_predictions)
validate_predictions_file.close()
```

Python Example

Third Code Node

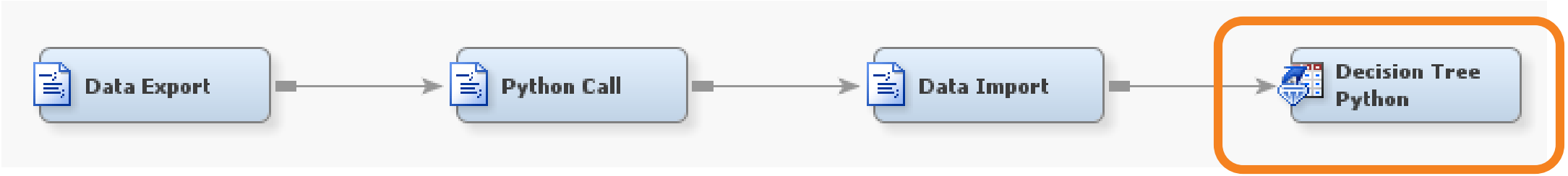


- The code in Data Import SAS Code Node (repeat for the validation and test data sets):

```
data predict_py_train;
   infile "&work_dir.\hmeq_python_train_predictions_dt.csv"
  dsd
  delimiter=','
  firstobs=2;
  input p_bad0 p_bad1;
run;

data &EM_EXPORT_TRAIN;
   merge &EM_IMPORT_DATA predict_py_train;
run;
```

Python Example

Model Import Node



- Map Predicted Variables

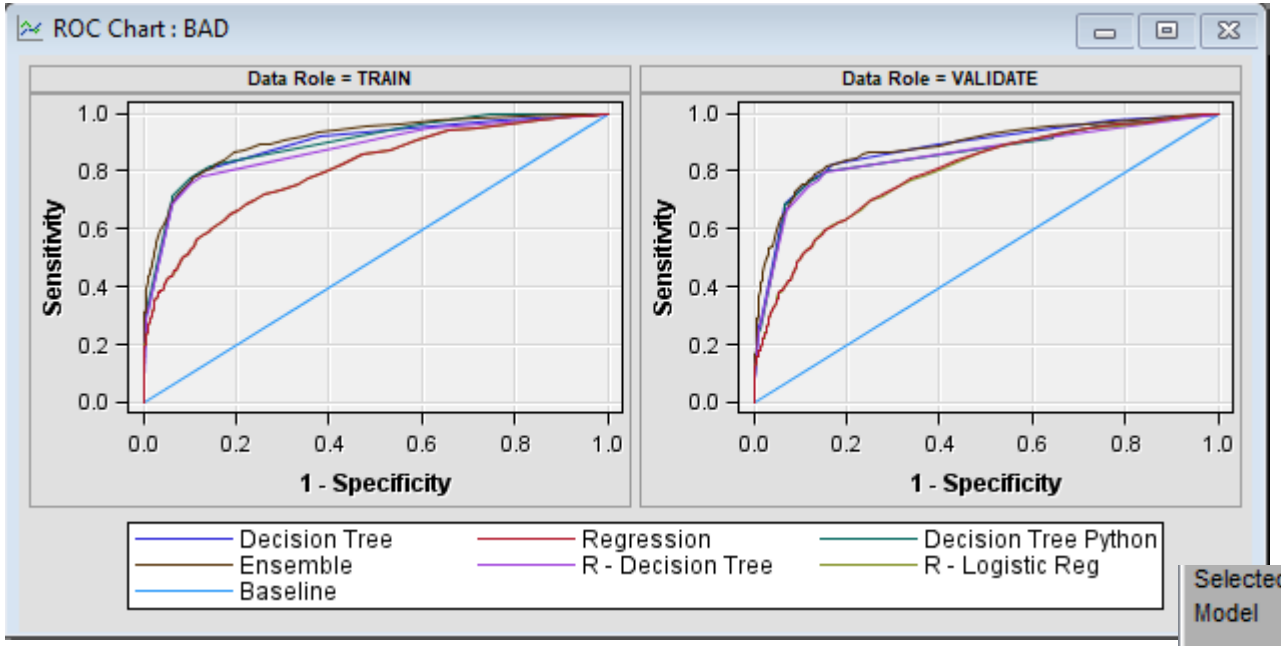
Mapping Editor-WORK.MAPPING

Level	Predicted Variable	Modeling Variable	Predicted Variable Label
0	P_BAD0	**P_BAD0**	Predicted: BAD=0
1	P_BAD1	**P_BAD1**	Predicted: BAD=1

- Model Import can be connected to Model Comparison node

Python Example

Model Comparison



Selected Model	Predecessor Node	Model Node	Model Description	Target Variable	Selection Criterion: Valid: Average Squared Error	Valid: Misclassification Rate
Y	Tree	Tree	Decision Tree	BAD	0.091844	0.11777
	MdlImp2	MdlImp2	Decision Tree Python	BAD	0.094479	0.117351
	Ensmbl	Ensmbl	Ensemble	BAD	0.094659	0.130763
	EMOPEN3	EMOPEN3	R - Decision Tree	BAD	0.095893	0.124476
	EMOPEN	EMOPEN	R - Logistic Reg	BAD	0.123401	0.168064
	Reg	Reg	Regression	BAD	0.123611	0.168064

SAS Enterprise Miner and Python

Demo

SASPy

SAS 9.4

Project headed by Jared Dean

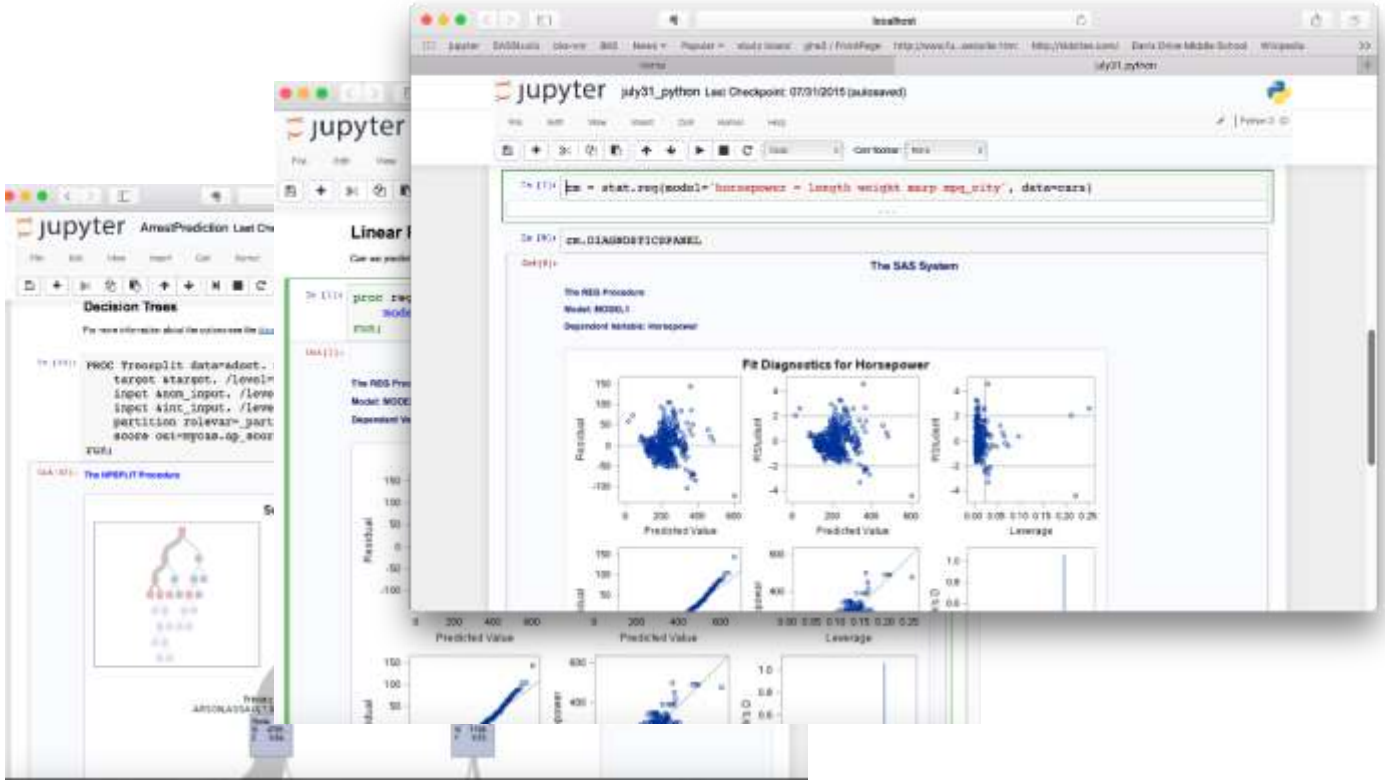
https://github.com/sassoftware

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 Notebooks integration being included in SAS University Edition
- More integration coming soon…

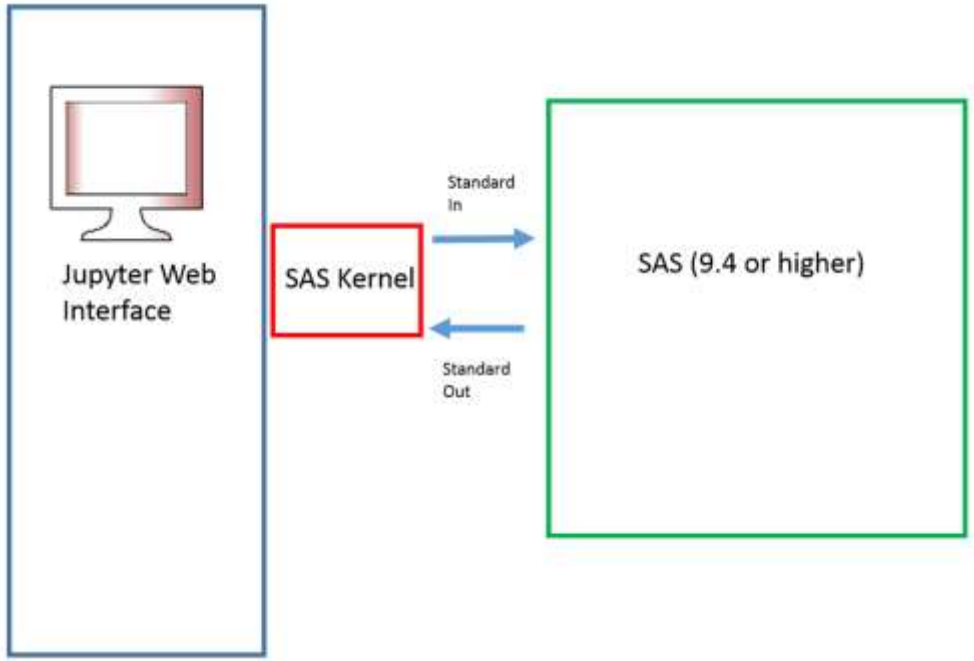


https://github.com/sassoftware/sas_kernel

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.

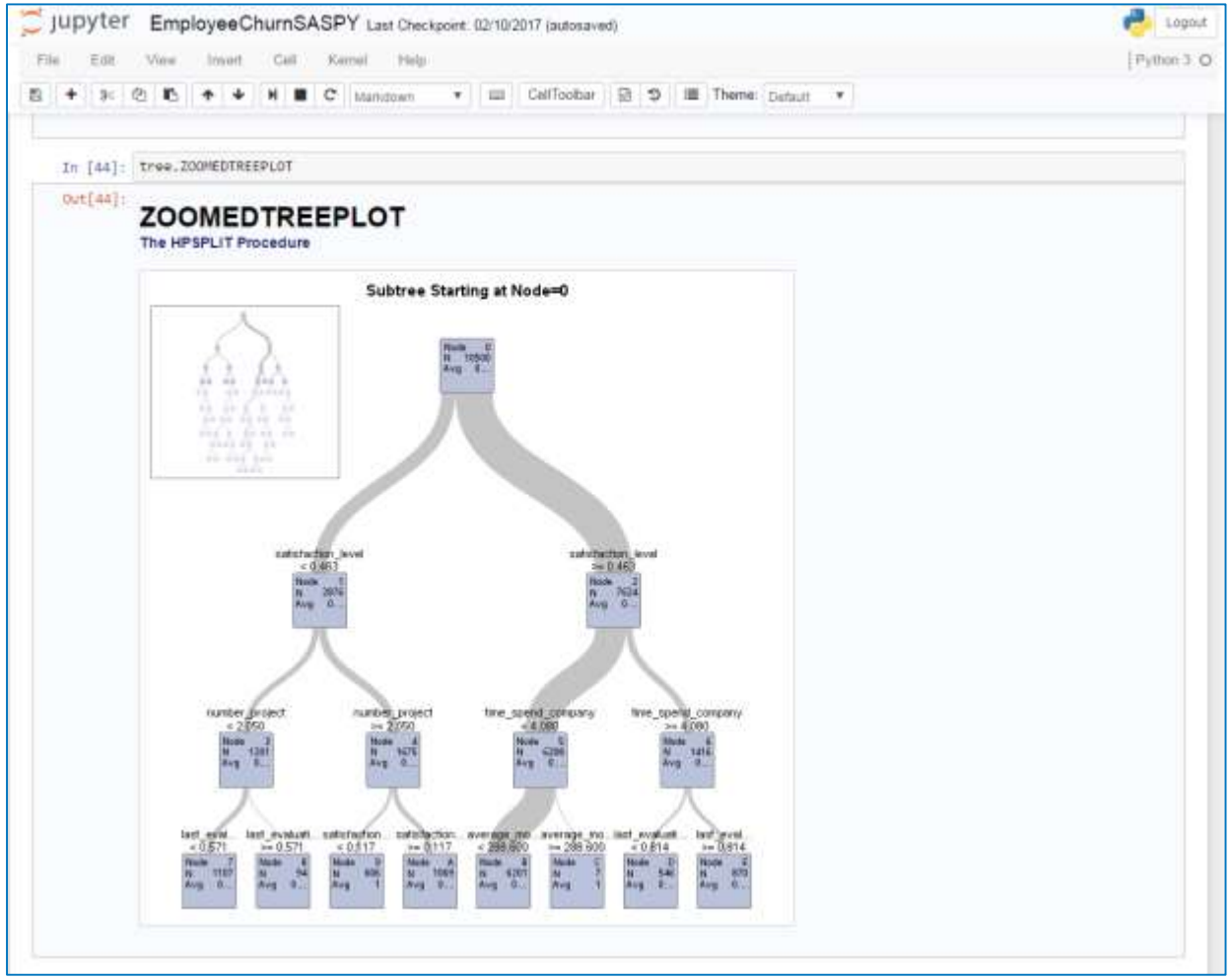
For SAS 9.4 and Python 3.X



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 [9]: import saspy

In [10]: #Assign SAS Data pointers
         prdsale = sas.sasdata('prdsale', libref='sashelp')

In [11]: prdsale.head()
```

The SAS System

Obs	ACTUAL	PREDICT	COUNTRY	REGION	DIVISION	PRODTYPE	PRODUCT	QUARTER	YEAR	MONTH
1	$925.00	$850.00	CANADA	EAST	EDUCATION	FURNITURE	SOFA	1	1993	Jan
2	$999.00	$297.00	CANADA	EAST	EDUCATION	FURNITURE	SOFA	1	1993	Feb
3	$608.00	$846.00	CANADA	EAST	EDUCATION	FURNITURE	SOFA	1	1993	Mar
4	$642.00	$533.00	CANADA	EAST	EDUCATION	FURNITURE	SOFA	2	1993	Apr
5	$656.00	$646.00	CANADA	EAST	EDUCATION	FURNITURE	SOFA	2	1993	May

SASPy

SAS Procedure vs SASPy Method for Linear Regression

jupyter SAS Linear Example Last Checkpoint: 4 minutes ago (autosaved)

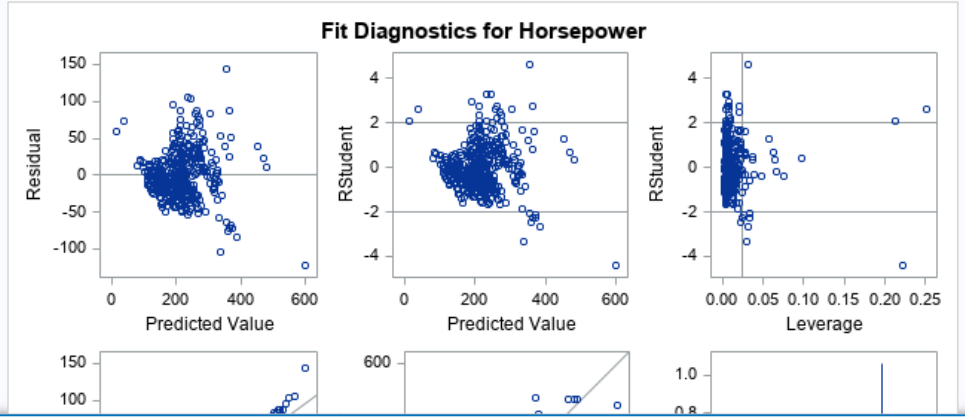
File Edit View Insert Cell Kernel Widgets Help

Run Code

Run SAS code in Jupyter Notebook using the SAS Kernel

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

Model: MODEL1
Dependent Variable: Horsepower



jupyter SASPy for slide example Last Checkpoint: 17 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [19]:
```
stat_results = sasstat.reg(model='horsepower = length weight msrp mpg_city', data=cars)
```

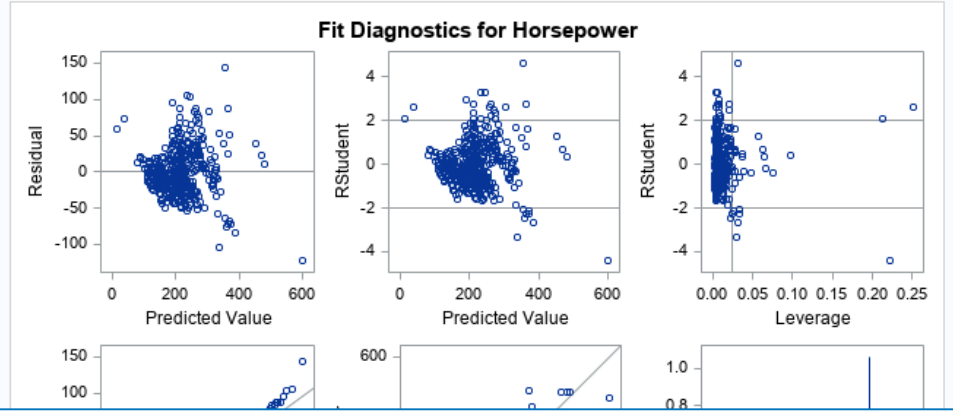
In [20]:
```
stat_results.DIAGNOSTICSPANEL
```

Out[20]:

DIAGNOSTICSPANEL

The SAS System

The REG Procedure
Model: MODEL1
Dependent Variable: Horsepower



Viewing the SAS Code Behind SASPy

```
In [43]: sas.teach_me_SAS(True)

In [44]: sales.series(y=['tot_sales','predicted_sales'], x='month', title='total vs. predicted sales')

proc sgplot data=WORK.sales;
        title "total vs. predicted sales";
        series x=month y=tot_sales;
        series x=month y=predicted_sales;
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

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

SASPy

Demo

PROC FCMP

Submit and execute functions written in Python from within a SAS session using the new Python object

PROC FCMP

What is it?

- SAS Function Compiler
- Enables you to create, test, and store SAS functions, CALL routines, and subroutines before you use them in other SAS procedures or DATA steps

PROC FCMP Documentation

Using PROC FCMP to the Fullest: Getting Started and Doing More

PROC FCMP

Python Objects

- Python objects enable you to embed and import Python functions into SAS programs
- The Python code is not converted to SAS code. Instead, the Python code runs in the Python interpreter of your choice and returns the results to SAS.
- With a small Python code modification, you can run your Python functions from SAS and easily program in both languages at the same time.
- Available with SAS 9.4 M6 (May 2019)

Using Python functions inside of SAS Programs

PROC FCMP

Using Python Functions in 5 Steps

Python Function Workflow

1. Declare a Python object & a dictionary object
2. Insert Python source code into SAS
3. Publish Python source code
4. Call the Python source code
5. Return results from the dictionary

```
proc fcmp;
declare object py(python);
submit into py;
def PyProduct(var1, var2):
   "Output: MyKey"
   newvar = var1 * var2
   return newvar,
endsubmit;
rc = py.publish();
rc = py.call("PyProduct", 5, 10);
MyResult =py.results["MyKey"];
put MyResult=;
run;
```

Results

```
MyResult=50
```

PROC FCMP

Demo

SAS Viya

R and Python Integration with Viya

^Note, SAS does NOT provide python or R kernel/packages

Python/R User Integration

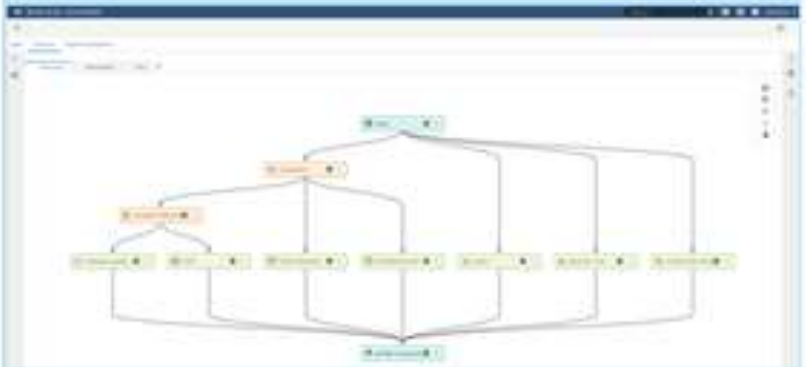
Adding SAS library (SWAT) to Python or R User Experience

```
# Impute missing values
castbl.dataPreprocess.impute(
    outVarsNamePrefix = 'IMP',
    methodContinuous  = 'MEDIAN',
    methodNominal     = 'MODE',
    inputs            = list(df_data_card['_VARNAME_'])[1:],
    copyAllVars       = True,
    casOut            = castbl
)

# Print the first five rows with imputations
castbl.head()
```

```
params = dict(
    table    = dict(name = indata, where = '_partind_ = 1'),
    target   = target,
    inputs   = all_inputs,
    nominals = class_vars,
)

s.decisionTree.gbtreeTrain(**params, seed = 1, casOut =
                           dict(name = 'gbt_model', replace = True))
```

Python/R Modeling

Run Python and R and SAS models, then compare results to find the best model



Python Deployment

Data Prep and Scoring Situations Viya can execute distributed and in parallel a custom python method against each row of data

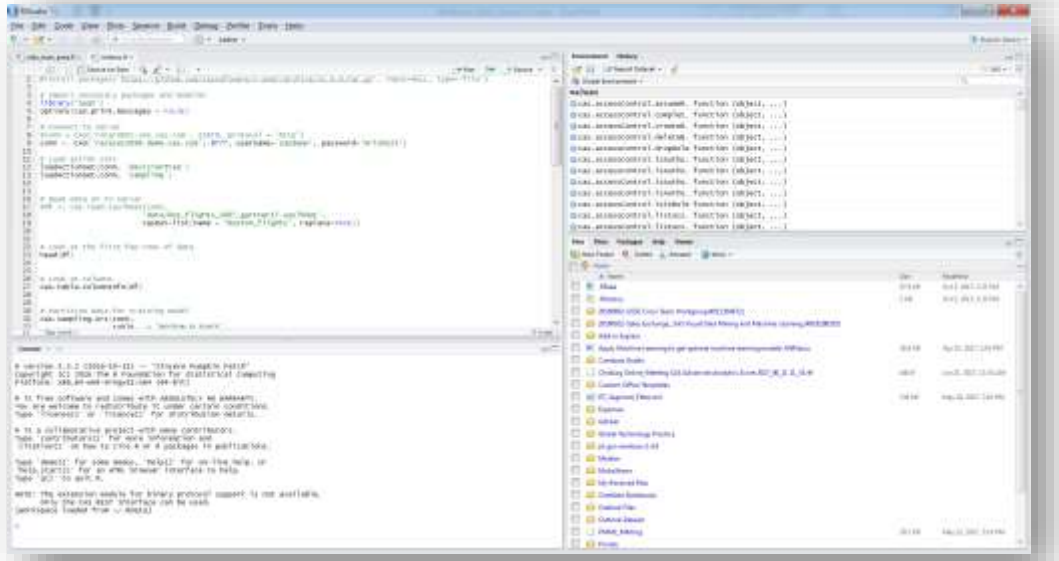
```
#Helper wrapper package
from capsule import main
p = main.from_pickle('model.pkl', 'predict', 'iris',
                     ['PetalLength','PetalWidth'])
```

```
s = swat.CAS(hostname, authinfo=authfile)
s.loadactionset('ds2')
#Run python score
r = s.ds2.runds2(repr(p))
```

*Not for Model Development

SAS® Scripting Wrapper for Analytics Transfer (SWAT)

- Access to SAS® Viya™from Python and R
- Integration of SAS® Analytics in Python and R code
- R Studio and Jupyter Notebook support
- Issue tracking and collaboration in development through GitHub project



Open to SAS®

APIs

sas

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

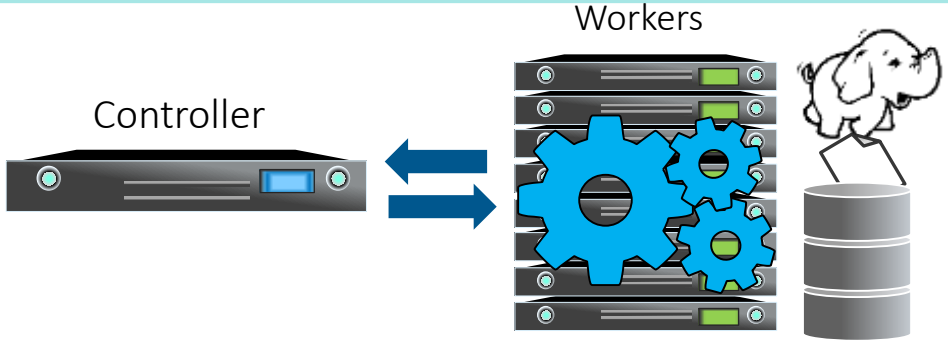
python

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

R

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

SAS Viya

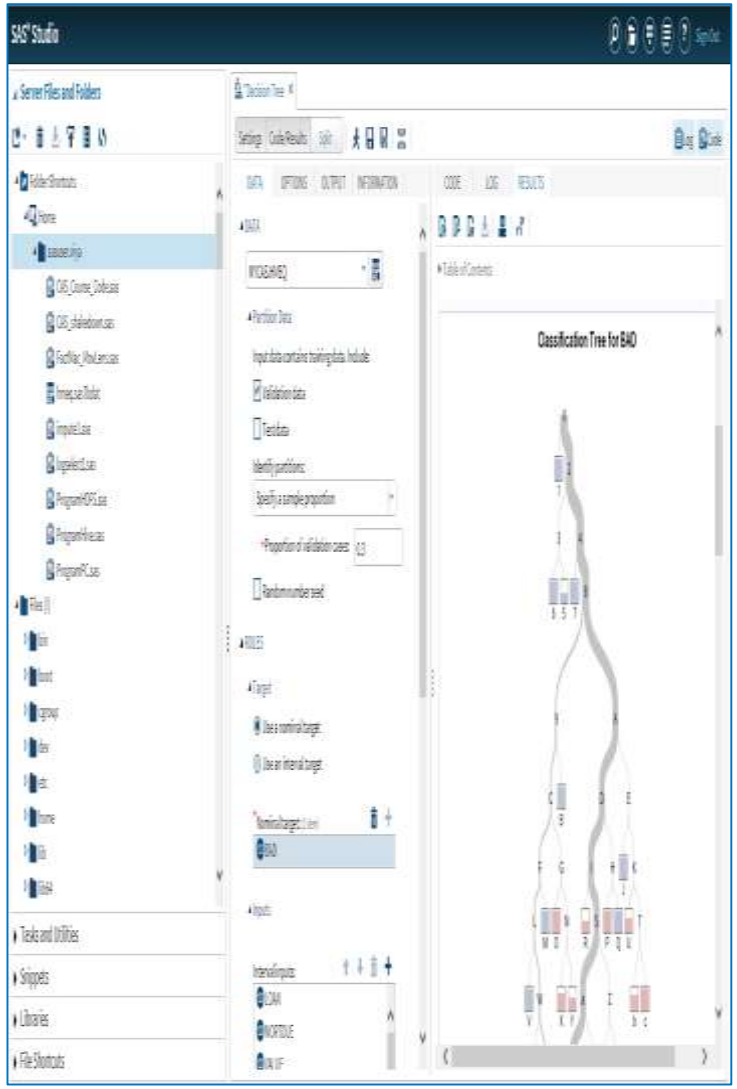


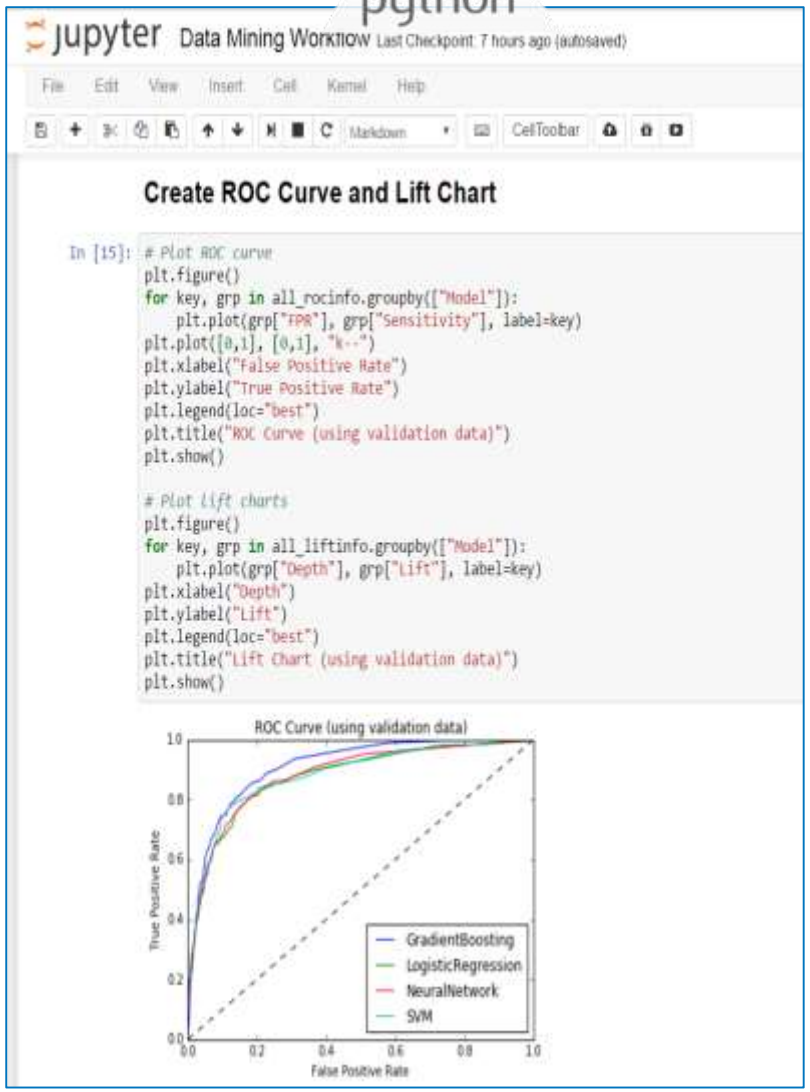
CAS Action

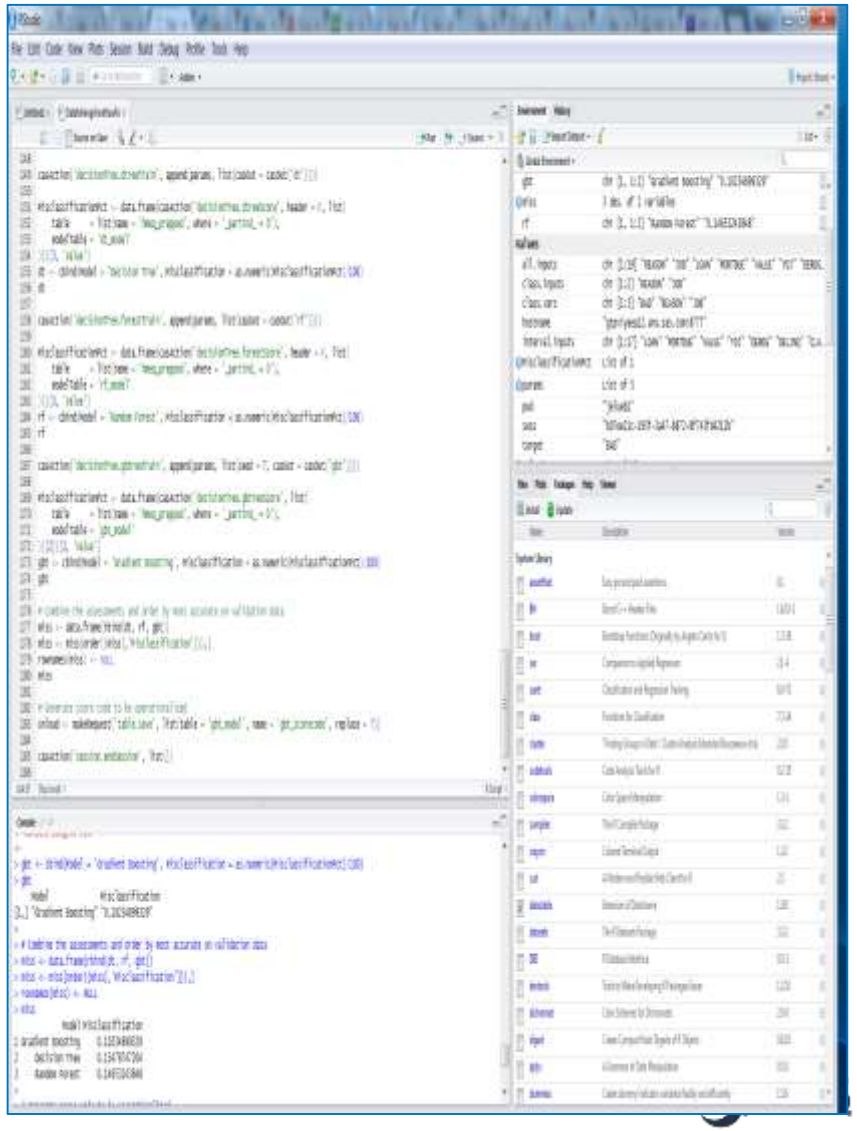
```
[table.fetch]
    table.name = "hmeq"
    from = 1 to = 10
```

sas

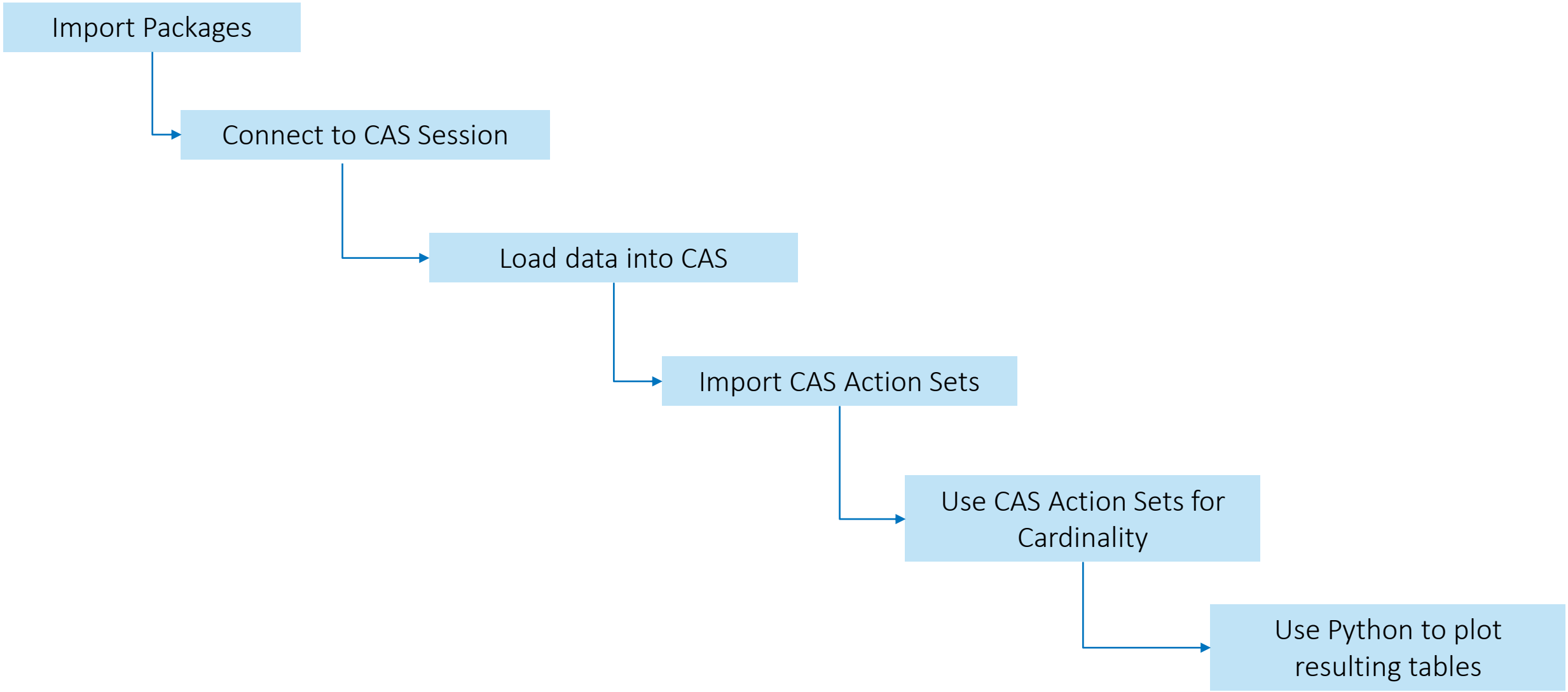
SAS Viya







An Example Flow for Python SWAT Interface to CAS



SWAT Python

Import packages and connect to CAS Session

Import needed packages

```
import swat
from swat.render import render_html
import pandas as pd
from matplotlib import pyplot as plt
```

Connect to SAS CAS Session

```
cashost='pdcesx06174.exnet.sas.com'
casport=5570

sas = swat.CAS(cashost, casport, 'sasdemo', 'Orion123')
```

SWAT Python

Load data into CAS

Load data

```
sas.read_csv("c:/public/financial/data/hmeq.csv", casout="hmeq")
```

```
NOTE: Cloud Analytic Services made the uploaded file available as table HMEQ in caslib CASUSER(sasdemo).
NOTE: The table HMEQ has been created in caslib CASUSER(sasdemo) from binary data uploaded to Cloud Analytic Services.
```

Take a look at the data

```
hmeq = sas.CASTable('hmeq', caslib="CASUser")
hmeq.head()
```

Selected Rows from Table HMEQ

	BAD	LOAN	MORTDUE	VALUE	REASON	JOB	YOJ	DEROG	DELINQ	CLAGE	NINQ	CLNO	DEBTINC
0	1.0	1100.0	25860.0	39025.0	HomeImp	Other	10.5	0.0	0.0	94.366667	1.0	9.0	NaN
1	1.0	1300.0	70053.0	68400.0	HomeImp	Other	7.0	0.0	2.0	121.833333	0.0	14.0	NaN
2	1.0	1500.0	13500.0	16700.0	HomeImp	Other	4.0	0.0	0.0	149.466667	1.0	10.0	NaN
3	1.0	1500.0	NaN	NaN			NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	0.0	1700.0	97800.0	112000.0	HomeImp	Office	3.0	0.0	0.0	93.333333	0.0	14.0	NaN

SWAT Python

Action Sets

```
sas.actionsetinfo(all=True)
```

§ setinfo

Action set information

	actionset	label	loaded	extension	version	product_name
0	access		0	tkacon	3.04.000	tkcas
1	accessControl	Access Controls	1	casmeta	3.04.000	tkcas
2	aggregation		0	tkcasagg	3.04.043	crsaggregate
3	astore		0	astore	3.04.043	crsastore
4	audio		0	audio	3.04.043	crsaudio
5	autotune		0	optminer	3.04.043	crsoptminer
6	bayesianNetClassifier		0	tkcasbnet	3.04.043	crsbayesian
7	bioMedImage		0	biomedimage	3.04.043	crsbiomedimg
8	boolRule		0	casblr	3.04.043	crsboolrule
9	builtins	Builtins	1	tkcasablt	3.04.000	tkcas
10	cardinality	Cardinality Analysis	1	cardinality	3.04.043	crscardinal
11	cdm		0	cdm	3.04.043	crsaglossmod
12	clustering		0	tkcaskclus	3.04.043	crskmeans
13	conditionalRandomFields		0	crf	3.04.043	crscrf

SWAT Python

Import CAS Action Sets

```
sas.loadactionset(actionset="dataStep")
sas.loadactionset(actionset="dataPreprocess")
sas.loadactionset(actionset="cardinality")
sas.loadactionset(actionset="sampling")
sas.loadactionset(actionset="regression")
sas.loadactionset(actionset="decisionTree")
```

```
NOTE: Added action set 'dataStep'.
NOTE: Added action set 'dataPreprocess'.
NOTE: Added action set 'cardinality'.
NOTE: Added action set 'sampling'.
NOTE: Added action set 'regression'.
NOTE: Added action set 'decisionTree'.
```

SWAT Python

Use SAS Action Sets for Cardinality

```
sas.cardinality.summarize(
    table=hmeq,
    cardinality={"name":"data_card", "replace":True}
)

tbl_data_card=sas.CASTable('data_card')

tbl_data_card.vars=['_VARNAME_','_NOBS_','_TYPE_','_NMISS_','_MEAN_','_MIN_','_MAX_','_STDDEV_']
df_data_card=tbl_data_card

df_data_card.head(15)
```

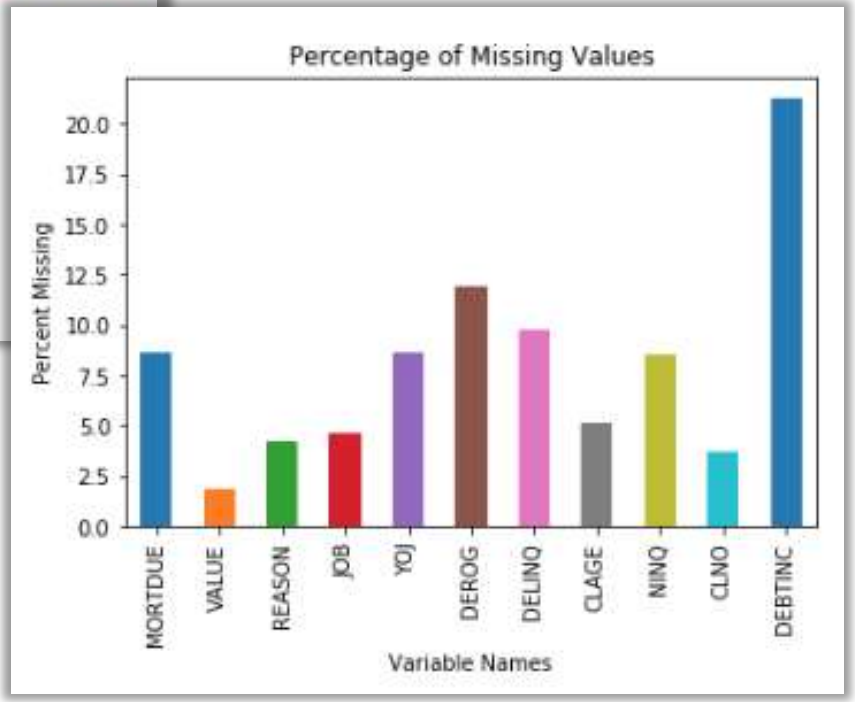
```
NOTE: Writing cardinality.
NOTE: status = 0.
NOTE: The Cloud Analytic Services server processed the request in 0.014589 seconds.
```

	VARNAME	_NOBS_	_TYPE_	_NMISS_	_MEAN_	_MIN_	_MAX_	_STDDEV_
0	MORTDUE	5960.0	N	518.0	73760.817200	2063.000000	399550.000000	44457.609458
1	VALUE	5960.0	N	112.0	101776.048741	8000.000000	855909.000000	57385.775334
2	REASON	5960.0	C	252.0	NaN	NaN	NaN	NaN
3	JOB	5960.0	C	279.0	NaN	NaN	NaN	NaN
4	YOJ	5960.0	N	515.0	8.922268	0.000000	41.000000	7.573982
5	DEROG	5960.0	N	708.0	0.254570	0.000000	10.000000	0.846047

SWAT Python

Use Python to plot resulting tables

```
tbl_data_card=sas.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');
```



Swat Demo

Jupyter Notebooks

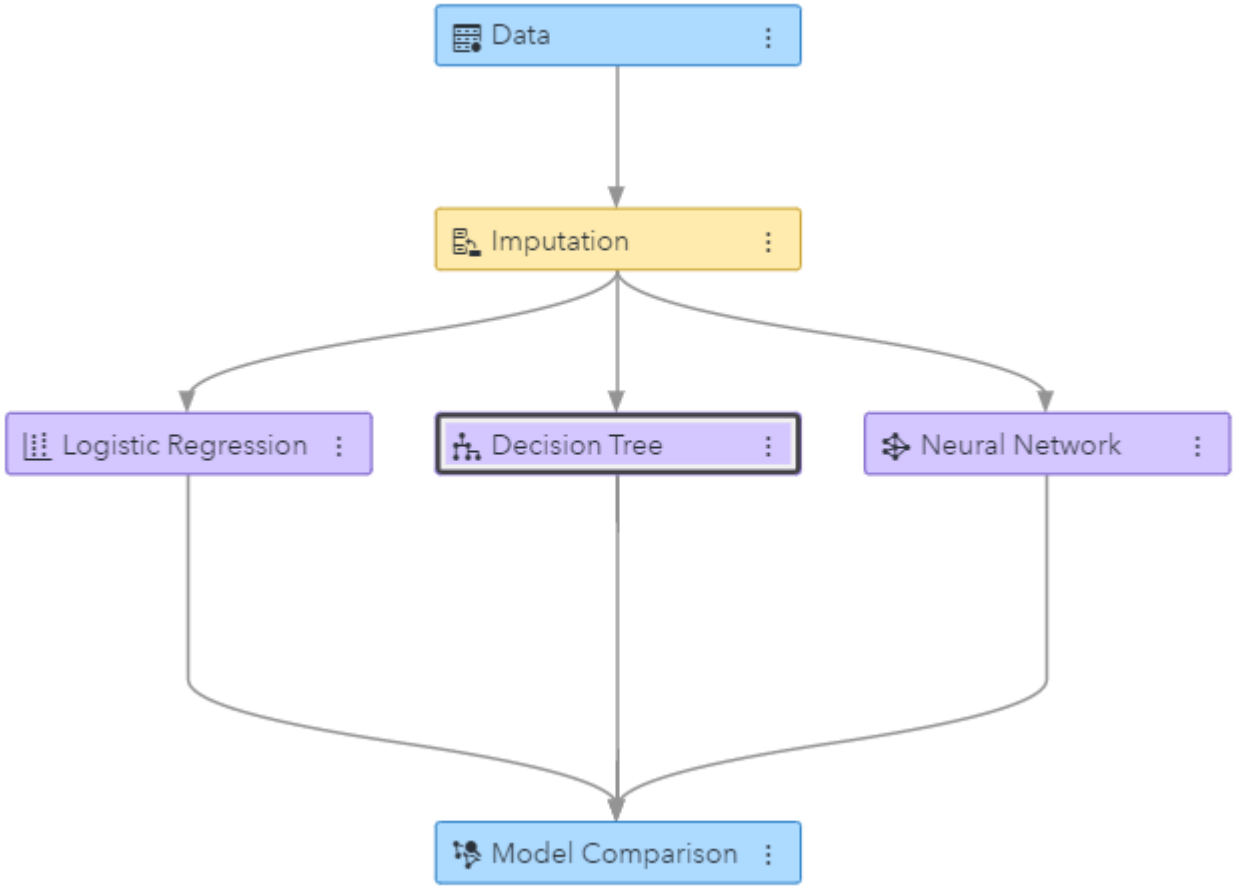
Visual Interface

Pipelines

SAS® Visual Data Mining and Machine Learning

Pipelines

- Drag-and-drop pipelines including preprocessing and machine learning techniques
- Customizable and portable nodes and SAS best practice pipelines (Toolbox)
- Support for SAS coding (macro, data step, procs, batch Enterprise Miner) within pipelines
- Collaboration through the use of the "Toolbox" – a collection of SAS Best Practice Pipelines, in addition to user-generated templates



Example Code for Pipeline

SAS® Visual Data Mining and Machine Learning

Pipelines

Data Mining Preprocessing

- Anomaly Detection
- Clustering
- Feature Extraction
- Feature Machine
- Filtering
- Imputation
- Interactive Grouping
- Manage Variables
- Reject Inference
- Replacement
- Text Mining
- Transformations
- Variable Clustering
- Variable Selection

Supervised Learning

- Batch Code
- Bayesian Network
- Decision Tree
- Forest
- GLM
- Gradient Boosting
- Linear Regression
- Logistic Regression
- Model Composer
- Neural Network
- Quantile Regression
- Score Code Import
- SVM

Postprocessing

- Ensemble

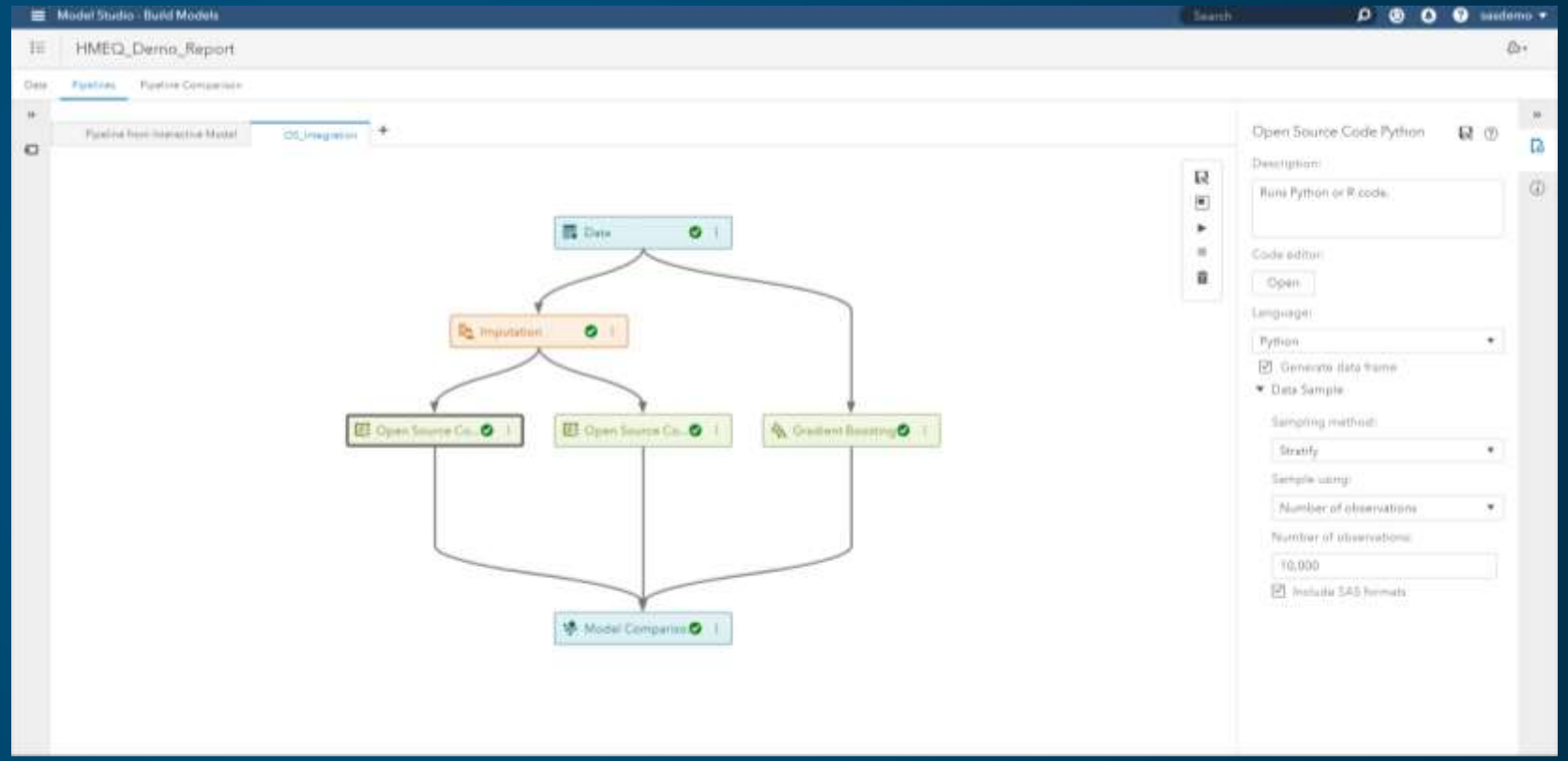
Miscellaneous

- Data Exploration
- Open Source Code
- SAS Code
- Save Data
- Score Data
- Scorecard
- Segment Profile

Pipelines

Open Source Code Node

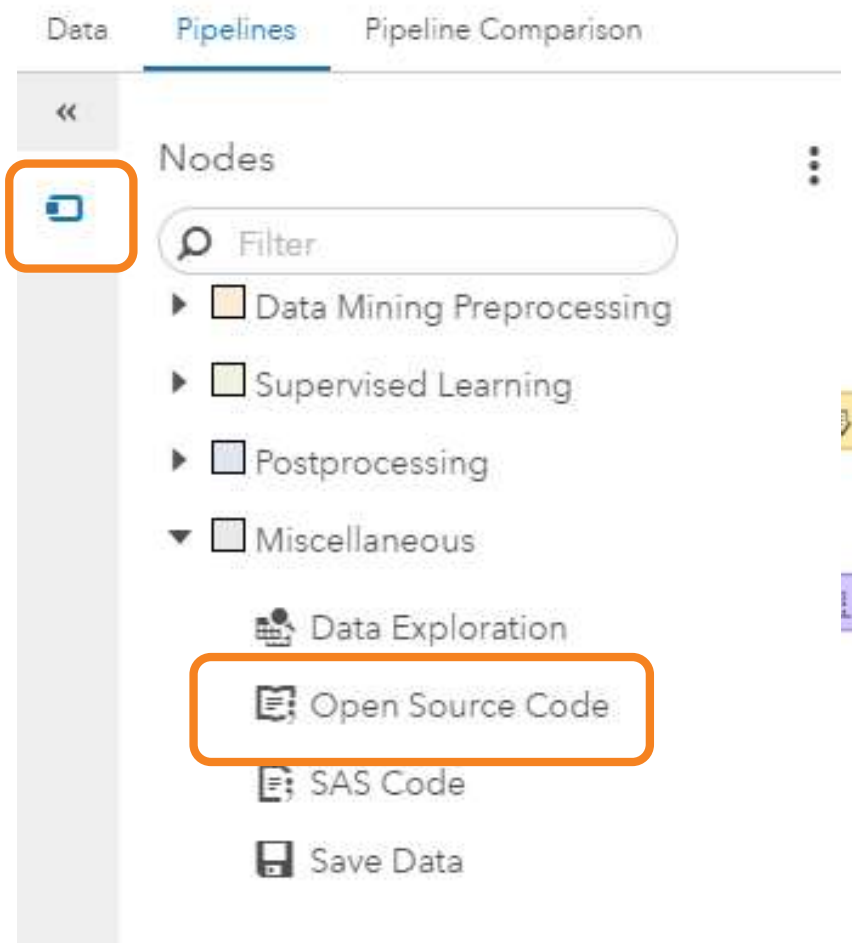
- Execute R and Python code and models from pipeline
- Downloads a sample of data to the CAS controller
- Runs the code natively in Python or R installed on the CAS controller.
- Produce model assessment and compare with other models (SAS and open source) to pick a champion.

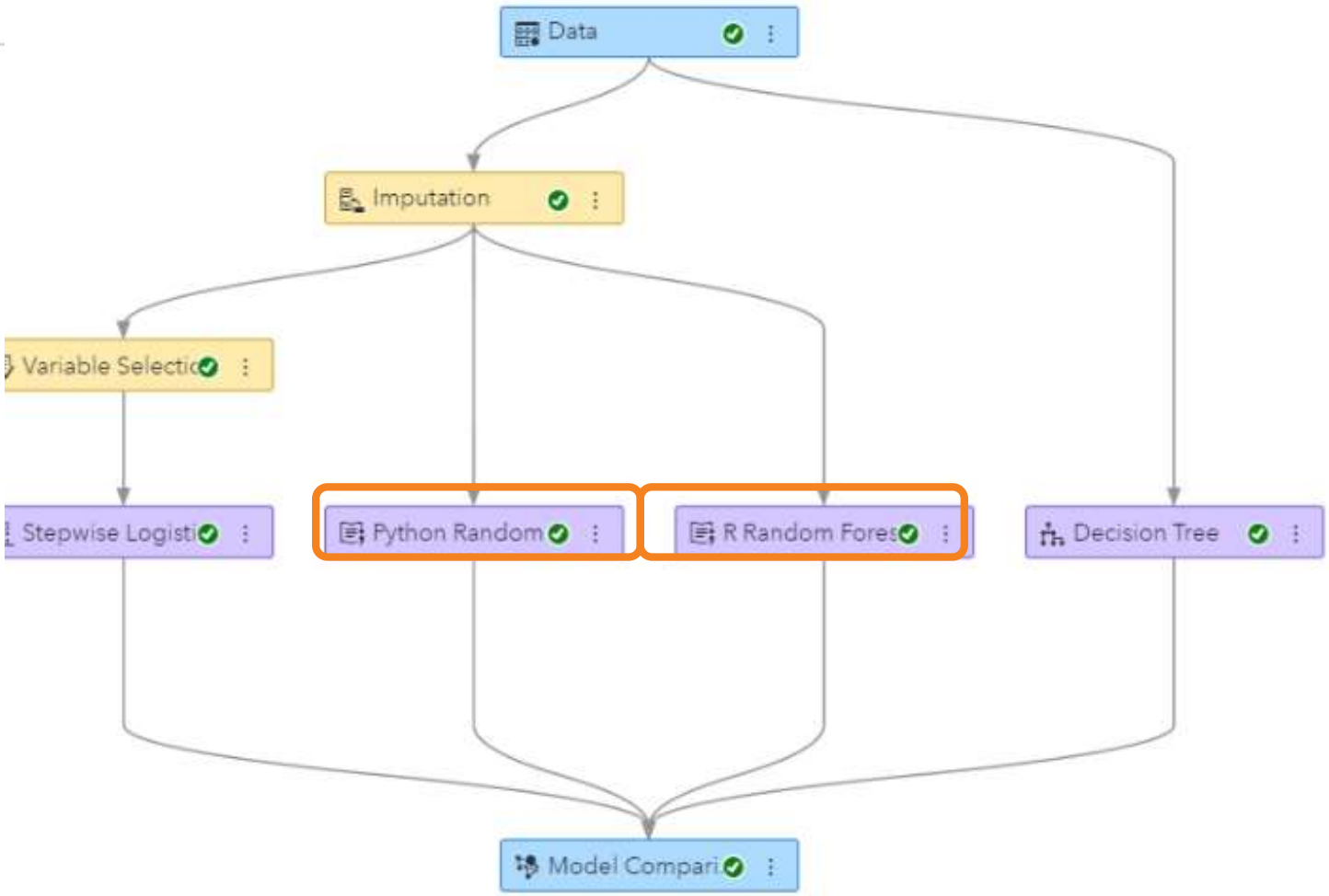


[Blog Examples]
[Sample Code on GitHub]

SAS Viya

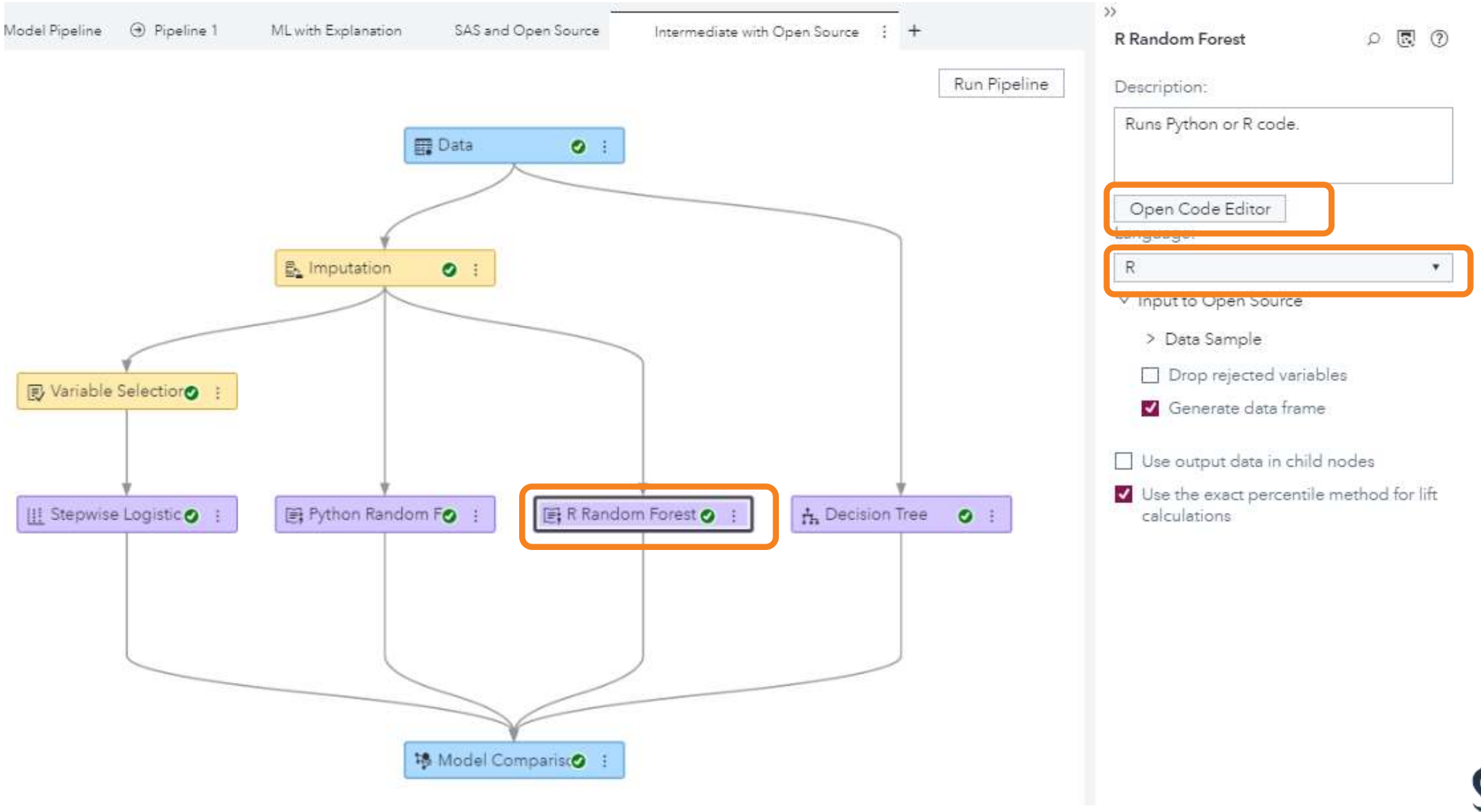
Pipelines





SAS Viya

Pipelines – Supervised Learning – R



SAS Viya

Pipelines – Supervised Learning – R

Model Studio - Build Models

Home Loan Default Demo > R Random Forest

R Variables

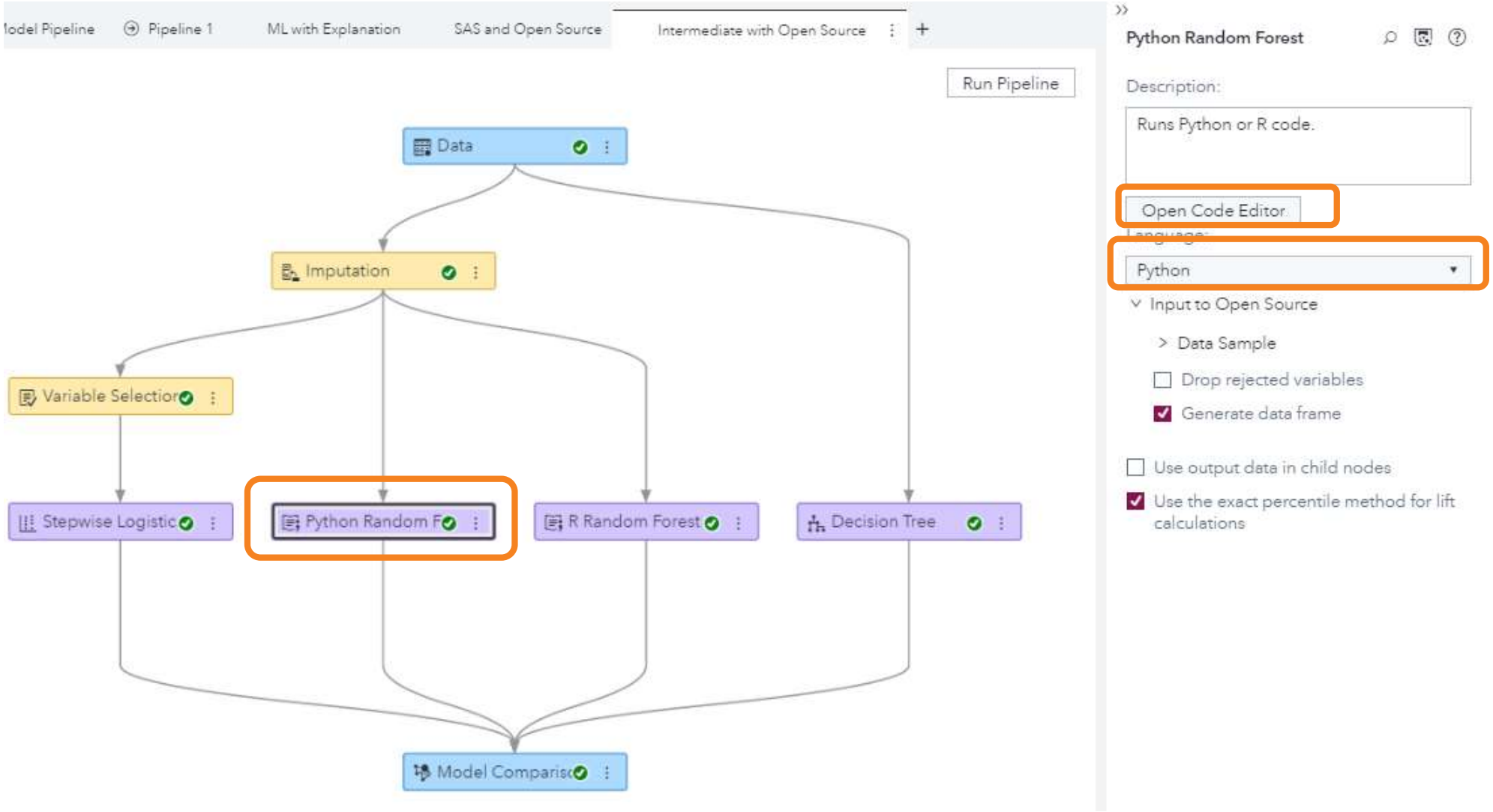
Filter

- dm_class_input
- dm_classtarget_intovar
- dm_classtarget_level
- dm_dec_target
- dm_input
- dm_inputdf
- dm_interval_input
- dm_model
- dm_model_formula
- dm_nodedir
- dm_partition_train_val
- dm_partitionvar
- dm_predictionvar
- dm_scoreddf
- dm_traindf
- node_data.csv
- node_scored.csv

```r
library(randomForest)

# RandomForest
dm_model <- randomForest(dm_model_formula, ntree=100, mtry=5, data=dm_traindf, importance=TRUE)

# Score
pred <- predict(dm_model, dm_inputdf, type="prob")
dm_scoreddf <- data.frame(pred)
colnames(dm_scoreddf) <- c(dm_predictionvar)

# Print/plot model output
png("rpt_forestMsePlot.png")
plot(dm_model, main='randomForest MSE Plot')
dev.off()

write.csv(importance(dm_model), file="rpt_forestIMP.csv", row.names=TRUE)
```

SAS Viya

Pipelines - Supervised Learning – Python



SAS Viya

Pipelines – Supervised Learning - Python Code

Model Studio - Build Models

Home Loan Default Demo > Python Random Forest

Python Variables

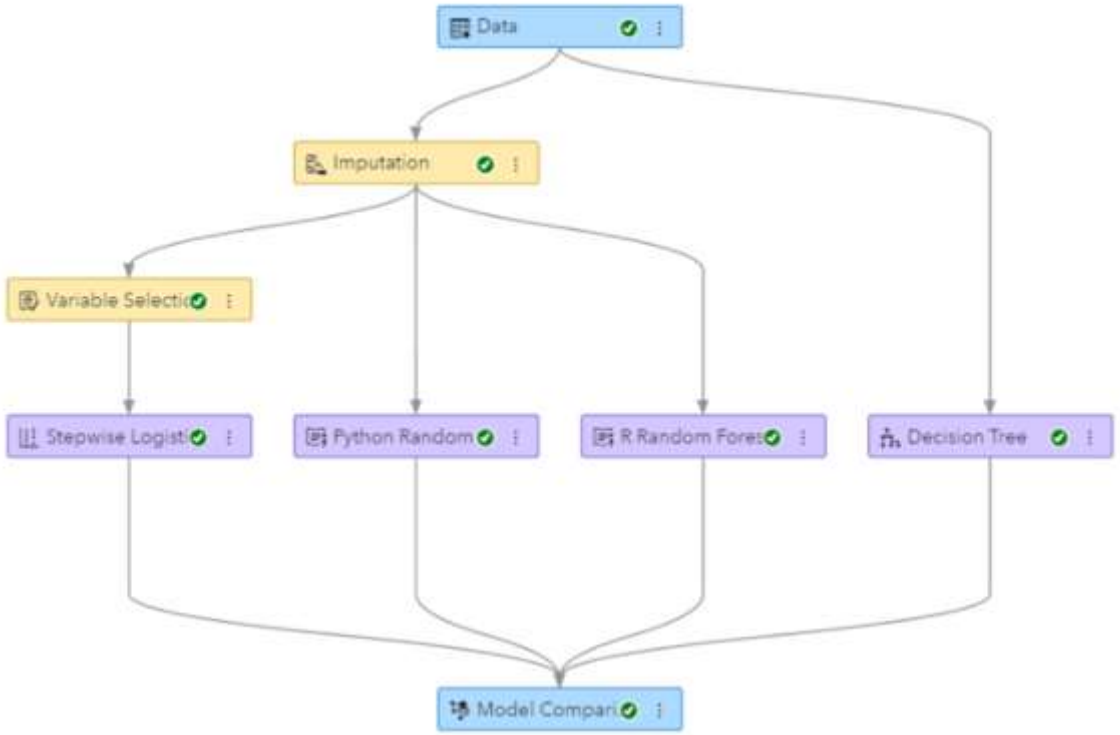
Filter

- dm_class_input
- dm_classtarget_intovar
- dm_classtarget_level
- dm_dec_target
- dm_input
- dm_inputdf
- dm_interval_input
- dm_model
- dm_nodedir
- dm_partition_train_val
- dm_partitionvar
- dm_predictionvar
- dm_scoreddf
- dm_traindf
- node_data.csv
- node_scored.csv

```python
from sklearn import ensemble

# Get full data with inputs + partition indicator
dm_input.insert(0, dm_partitionvar)
fullX = dm_inputdf.loc[:, dm_input]

# Dummy encode class variables
fullX_enc = pd.get_dummies(fullX, columns=dm_class_input, drop_first=True)

# Create X (features/inputs); drop partition indicator
X_enc = fullX_enc[fullX_enc[dm_partitionvar] == dm_partition_train_val]
X_enc = X_enc.drop(dm_partitionvar, 1)

# Create y (labels)
y = dm_traindf[dm_dec_target]

# Fit RandomForest model w/ training data
params = {'n_estimators': 100, 'max_depth': 20, 'min_samples_leaf': 5}
dm_model = ensemble.RandomForestClassifier(**params)
dm_model.fit(X_enc, y)
print(dm_model)

# Save VariableImportance to CSV
varimp = pd.DataFrame(list(zip(X_enc, dm_model.feature_importances_)), columns=['Variable Name', 'Importance'])
varimp.to_csv(dm_nodedir + '/rpt_var_imp.csv', index=False)

# Score full data
fullX_enc = fullX_enc.drop(dm_partitionvar, 1)
dm_scoreddf = pd.DataFrame(dm_model.predict_proba(fullX_enc), columns=[dm_predictionvar])
```

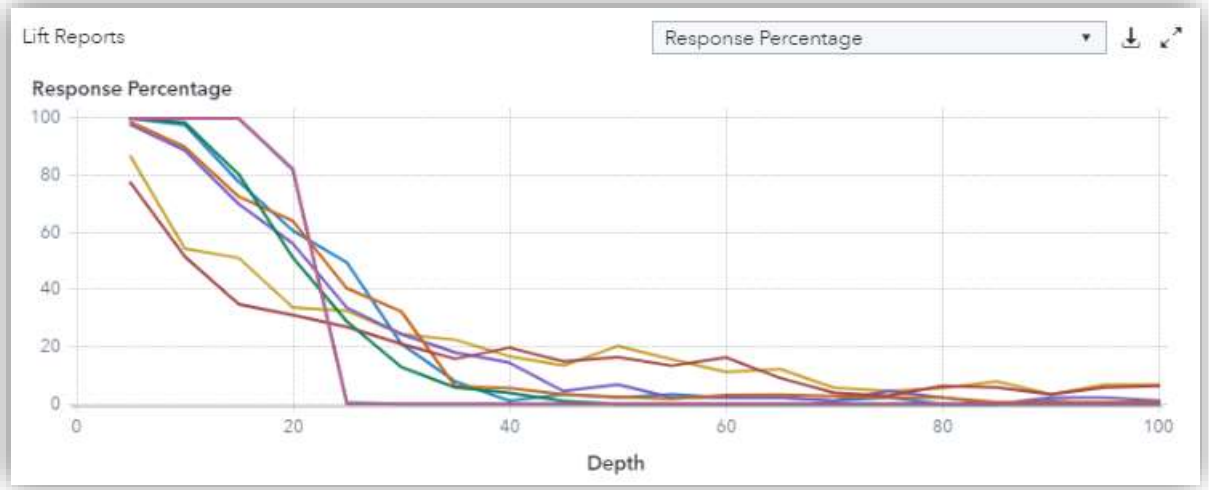
Performing model comparison

When executing in Supervised Learning



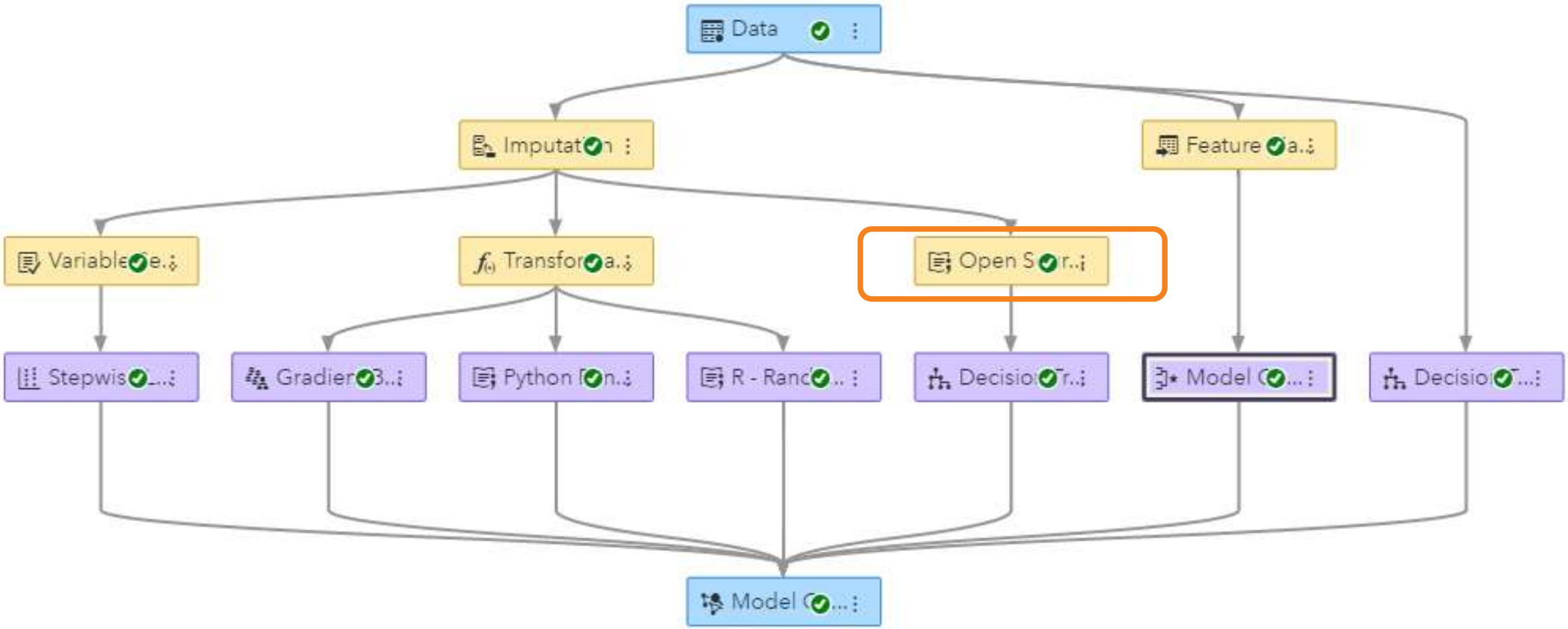
Model Comparison

Champion	Name	Algorithm Name	Misclassification Rate (Event)
	Gradient Boosting	Gradient Boosting	0.0811
	R Random Forest	Open Source Code	0.0940
	Python Random Forest	Open Source Code	0.1247
	Stepwise Logistic Regression	Logistic Regression	0.1779



SAS Viya

Pipelines - Data Mining Preprocessing – Python or R



SAS Viya

Pipelines – Data Mining Preprocessing – Python Code

Model Studio - Build Models

Home Loan Default Demo > Open Source Code

Python Variables

Filter

- dm_class_input
- dm_classtarget_intovar
- dm_classtarget_level
- dm_dec_target
- dm_input
- dm_inputdf
- dm_interval_input
- dm_model
- dm_nodedir
- dm_partition_train_val
- dm_partitionvar
- dm_predictionvar
- dm_scoreddf
- dm_traindf
- node_data.csv
- node_scored.csv

```python
import numpy as np
print('input column names:')
print(dm_inputdf.columns)

#Log transform IMP_VALUE variable dm_scoreddf should contain output data
dm_scoreddf=dm_inputdf
dm_scoreddf['LOG_IMP_VALUE']=np.log(dm_inputdf['IMP_VALUE'])
dm_scoreddf.drop('IMP_VALUE',axis=1, inplace=True)

print('output column names:')
print(dm_scoreddf.columns)
```

Visual Interface Demo

Pipelines

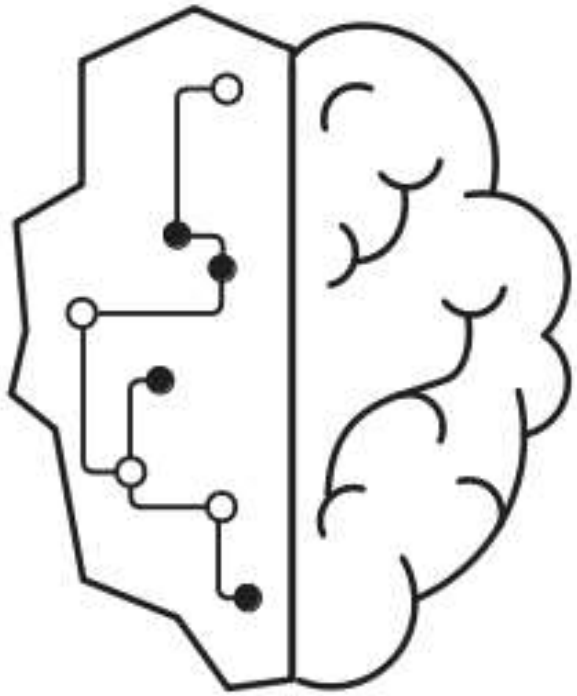
Deep Learning

DLPY

DLPy

Python package for SAS Deep Learning

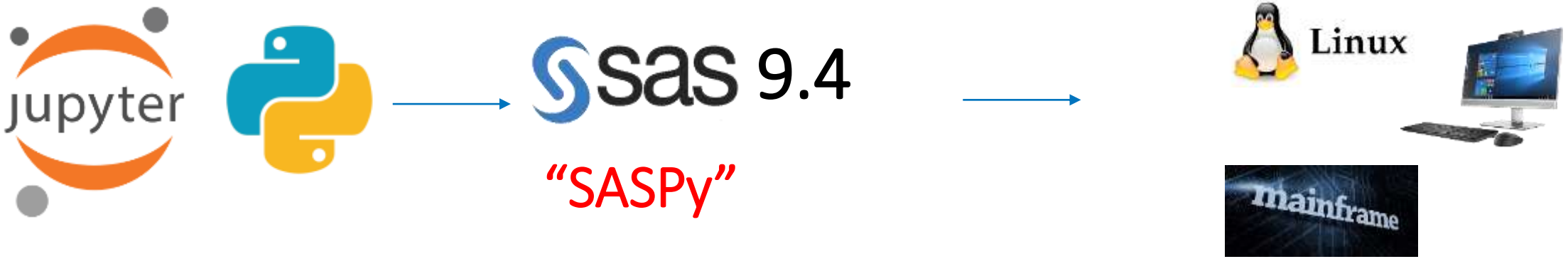
- DLPy provides the high-level Python APIs to deep learning methods in SAS Visual Data Mining and Machine Learning.
  - Saves the user from calling multiple, low level CAS actions to build / train neural networks
  - Utility functions to work with image tables, image processing functions in CAS
- DLPy is SAS version of “Keras”
- Supports several types of neural networks
  - Deep Neural Network (DNN)
  - Recurrent Neural Network (RNN)
  - Convolutional Neural Network (CNN)

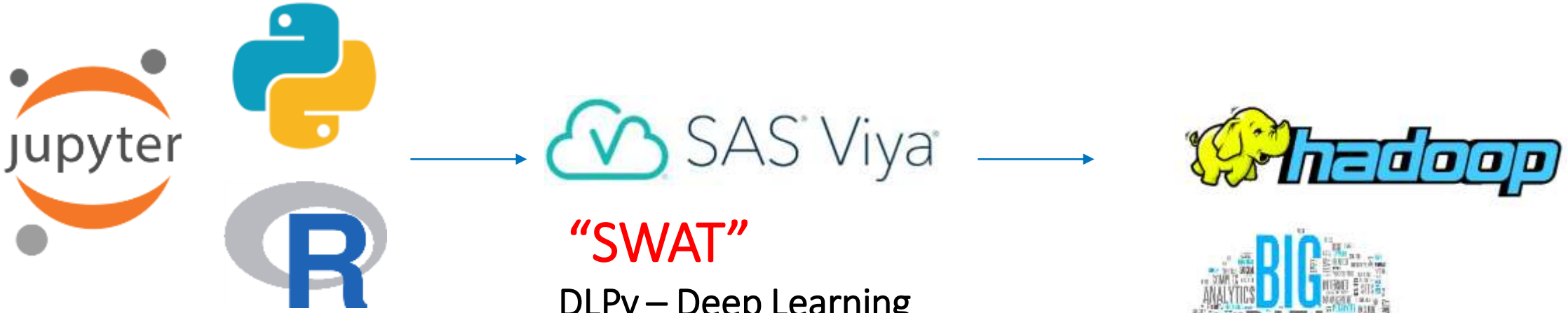


Resources

SAS and Open Source

Open APIs to Access SAS – All Accessible on Github





DLPy – Deep Learning
SASOPTPy – Optimization
ESPPy – Streaming Analytics

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
SAS 9.4

- **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

SAS and Open Source Resources

Empowering the Base SAS user

- SAS or Python? Why not use both? Using Python functions inside SAS
  https://blogs.sas.com/content/sgf/2019/06/04/using-python-functions-inside-sas-programs/
  https://proceedings.wuss.org/2019/73_Final_Paper_PDF.pdf

- Using Python to run jobs in your SAS Grid NEW 9/19/2019
  https://blogs.sas.com/content/sgf/2019/09/19/using-python-to-run-jobs-in-your-sas-grid

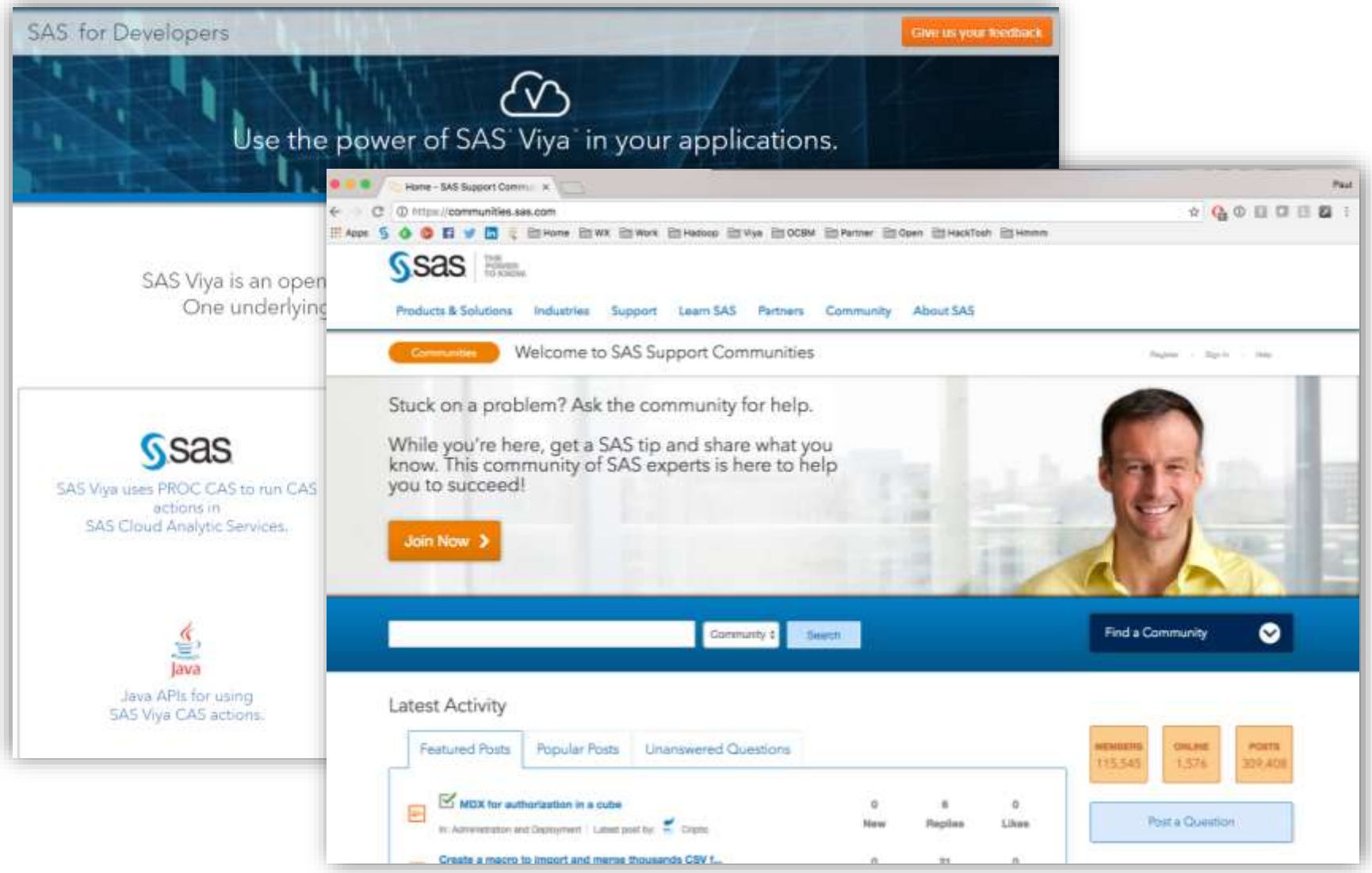
SAS and Open Source

Viya

- CAS actions documentation
- SAS Github page for SWAT-Python
- SAS Github page for SWAT-R
- More example scripts for using SWAT-R & SWAT-Python

Useful Websites

Developer.sas.com, Communities.sas.com



Questions?

Thank you for your time and attention!

Connect with me:
LinkedIn: https://www.linkedin.com/in/melodierush
Twitter: @Melodie_Rush

sas.com