

SAS® Visual Data Mining and Machine Learning Trial

Table of Contents

SAS® Visual Data Mining and Machine Learning Trial.....	- 1 -
Script Format:	- 2 -
Analyzing Donations Data:	- 2 -
Log onto SAS Viya:	- 2 -
Preparing Data.....	- 3 -
Interactive Modeling	- 6 -
Automated Modeling with Pipelines	- 12 -
Adding Interactive Models into Model Studio Project	- 16 -
Scoring with Models built in Model Studio	- 22 -
Invoking score code directly	- 22 -
Calling an API to invoke a scoring service	- 25 -
Free Play	- 26 -
SAS Drive.....	- 26 -

Script Format:

This script contains instructions for navigating the demo image, background information, and notes on the information being displayed. The symbols below are used throughout the script.

- Navigation Instructions (note that mouse clicks are left clicks unless otherwise specified)



Background Information



Notes

Analyzing Donations Data:



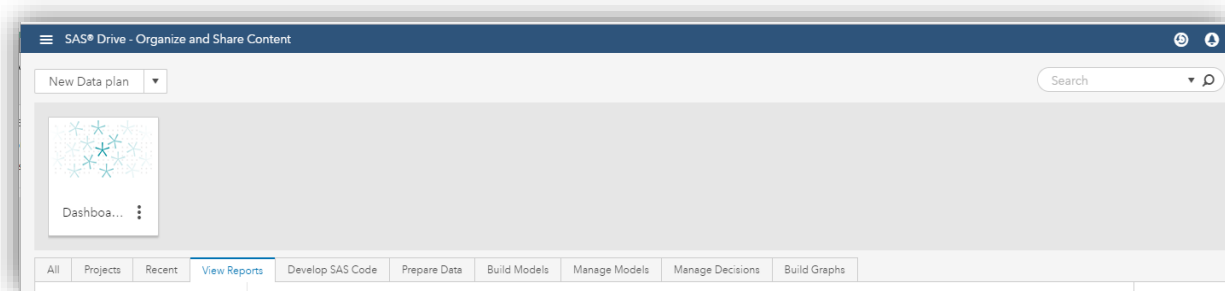
In this demo you will be analyzing donations data from DonorsChoose.org and building models to predict whether a given proposed project would be categorized as “exciting,” such that it will likely receive funding.



Discovery:

Log onto SAS Viya:

Using the logon provided in your welcome e-mail, log into SAS Viya.



Preparing Data



Data mining and machine learning applications start by selecting the data you want to work with—and transforming data in various ways to prepare for exploring and modeling. Here we will use the Data Studio application, which builds a “plan” of data preparation steps to transform a data set.

- Select “Prepare data” from the Actions (☰) menu in the upper left.
- Start a “New Plan”, and select the DONATIONS table as the data source.
 - Click on ‘Data Sources’
 - Expand the default folder and choose ‘Public’
 - Select “donations.sas7bdat” and click on the lightning bolt (⚡) on the top right.
 - Click the ‘OK’ button once it lights up.



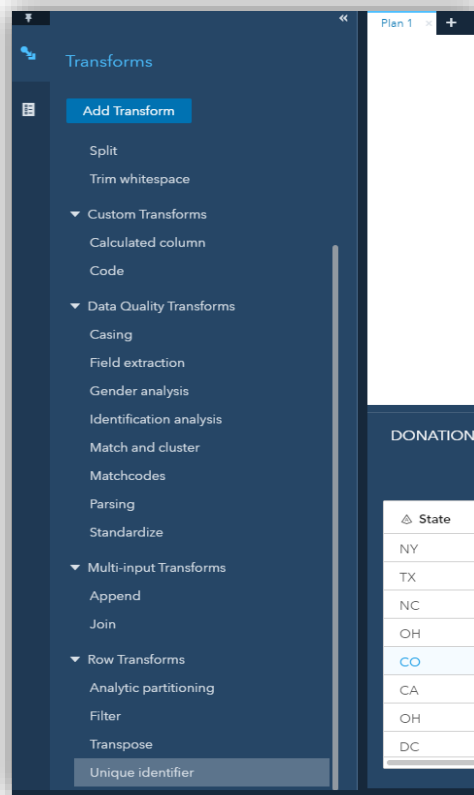
This data set has variables representing information about projects proposed by teachers soliciting donations for funding. It contains a binary variable “Exciting_project” that indicates whether this project was considered to be exciting (value of t=true and f=false) based on the school...the attributes of the project...and the donations. The goal is to build a predictive model using this data so that teachers can assess whether or not their project is likely to be funded.

- Take a quick look at a sample of the data presented.



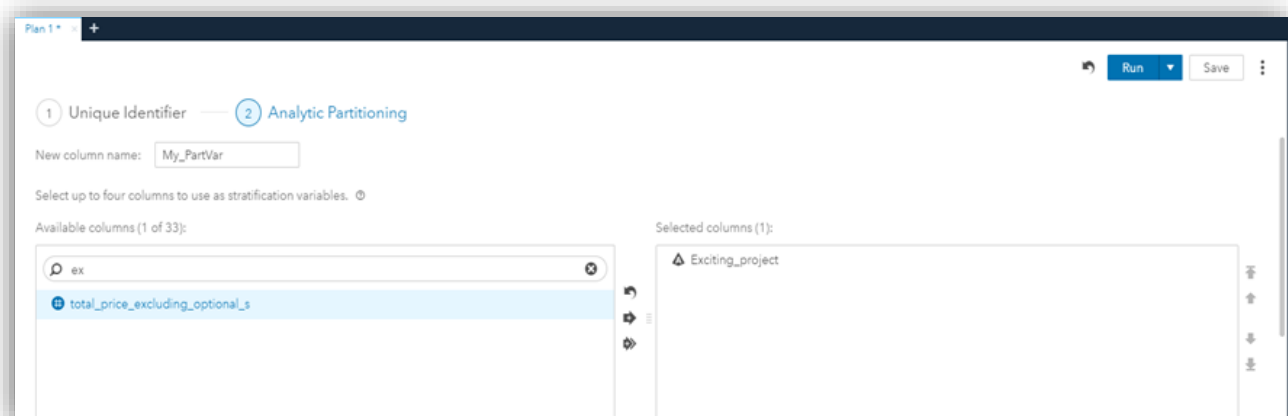
Data Studio provides numerous “Transforms” that you can apply to the data to prepare it for further exploration and modeling.

Use a Transform by double clicking the Unique Identifier and name it “ID” to your Plan. run it to see the new column appear in your table.

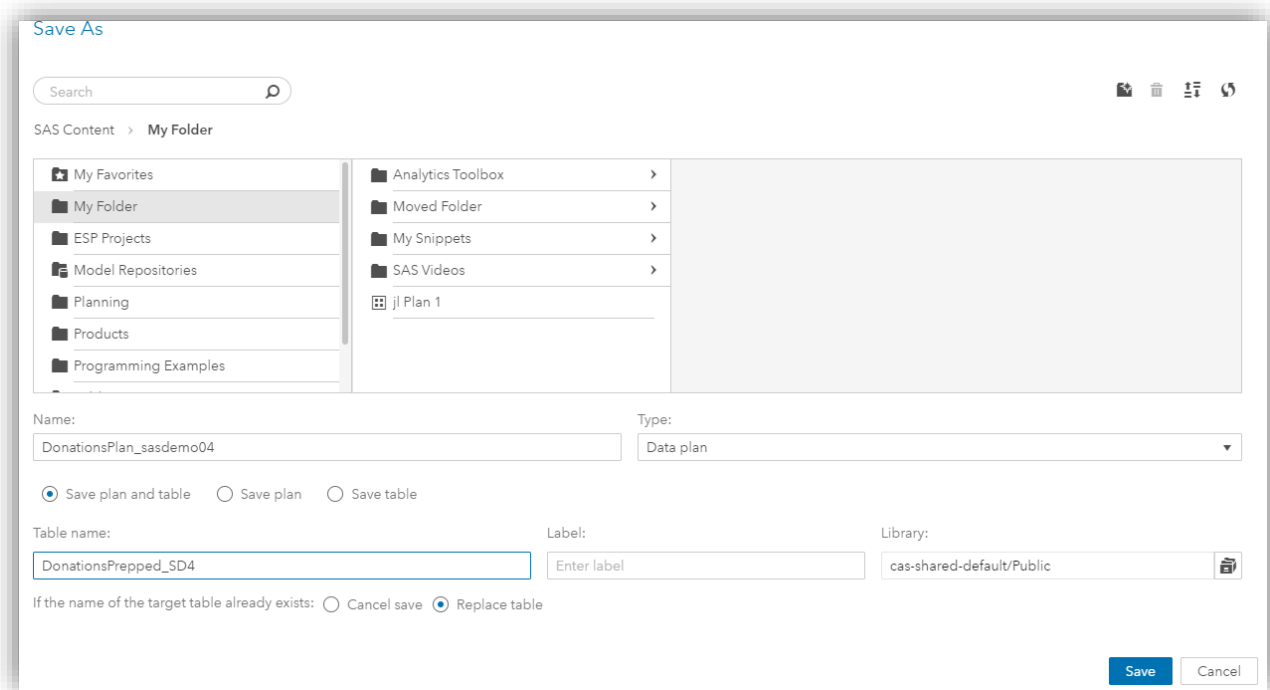


When you build predictive models, you need to assess them on data for validation that is not used to train the model—to avoid overfitting to the training data. Thus, you want to partition your data set into observations to use for training and observations to use for validation. We will do this by adding a partition column as an indicator.

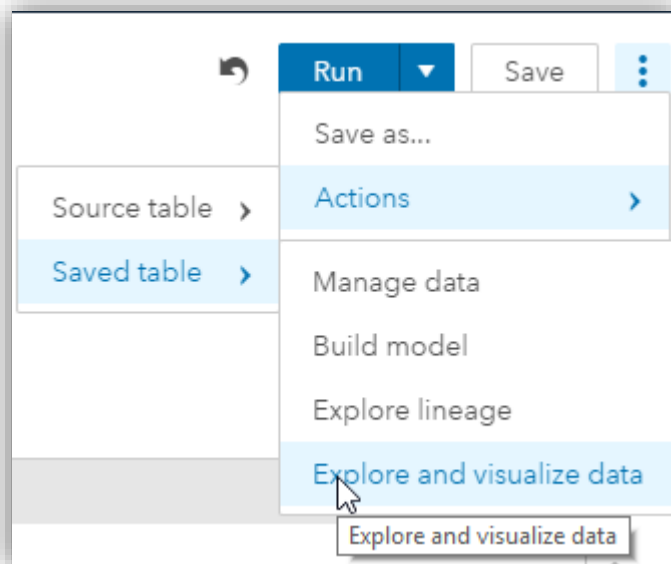
- Use the “Analytic Partitioning” transform to add a variable (i.e. new column) named myPartVar to contain the value of the partition for each observation. Stratify on “Exciting_project” by selecting the variable in the “Available columns” window (use the filter entry to find it easily) so that the true (t) and false (f) projects each get split into 60% training, 30% validation, and 10% test.
- Note: Be aware that some content in panels might not be visible, so you may need to scroll the panel.
 - Run this step in the data plan
 - Confirm by inspection that the new column contains 0, 1, and 2 values



- Save the plan and the output table:
 - Save the plan (with the name DonationsPlan_<your user ID>) into "My Folder", and
 - specify a name for the output table to be DonationsPrepped_<your initials>. This should be saved in /Public



- Find the Actions menu under the menu icon to see what you could do with your prepared data (the "saved table"). We want to get right into building some predictive models, so select "Explore and Visualize Data".

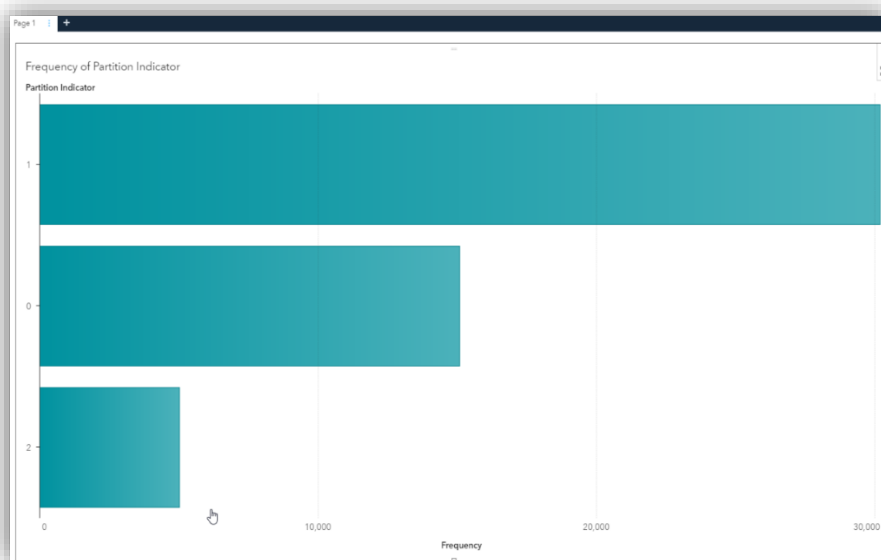


Interactive Modeling

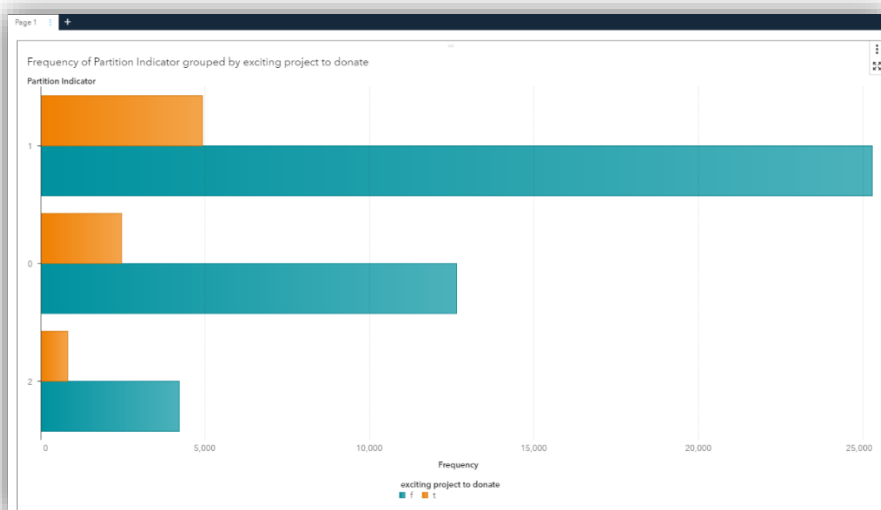


In this section, we continue to explore the Donation_Prepped data set to quickly build a decision tree model and a logistic regression model.

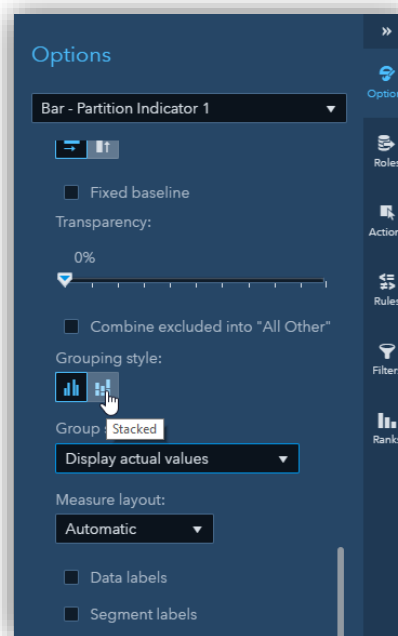
- First click the **Data** tab and take a look of the data set. You can see the two new columns (ID, myPartVar) are automatically recognized as an ID role and a Partition Indicator role.
- Drag and drop **Partition Indicator** to the central canvas. Now you can see we get the correct proportion of training (1), validation (0), and testing (2) samples.



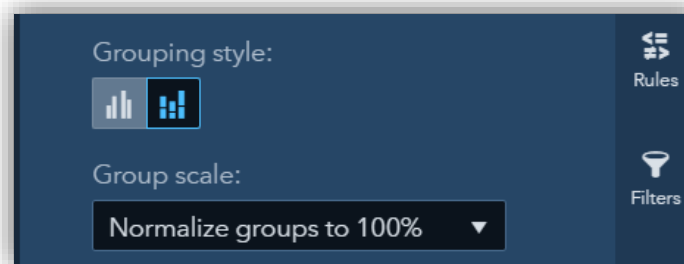
- Let us check the stratification as well. Drag and drop **Exciting Project to Donate** to the bar chart you just created. Assign it as a **Group** variable.



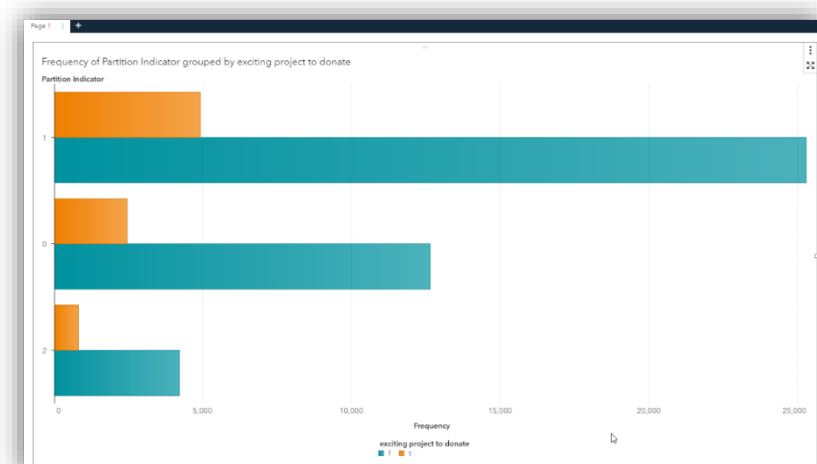
- In the **Options** tab, change **Grouping Style** to *Stacked*.



- Change **Group Scale** to *Normalize groups to 100%*.

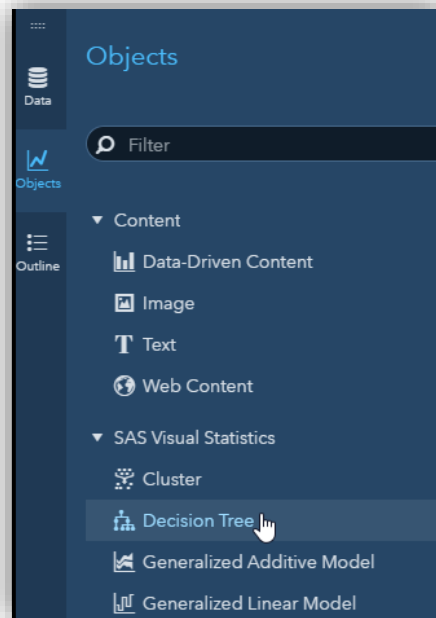


- As you can see, this is a valid stratified partition because each subset has the same proportion of events (Exciting Project to Donate=t).

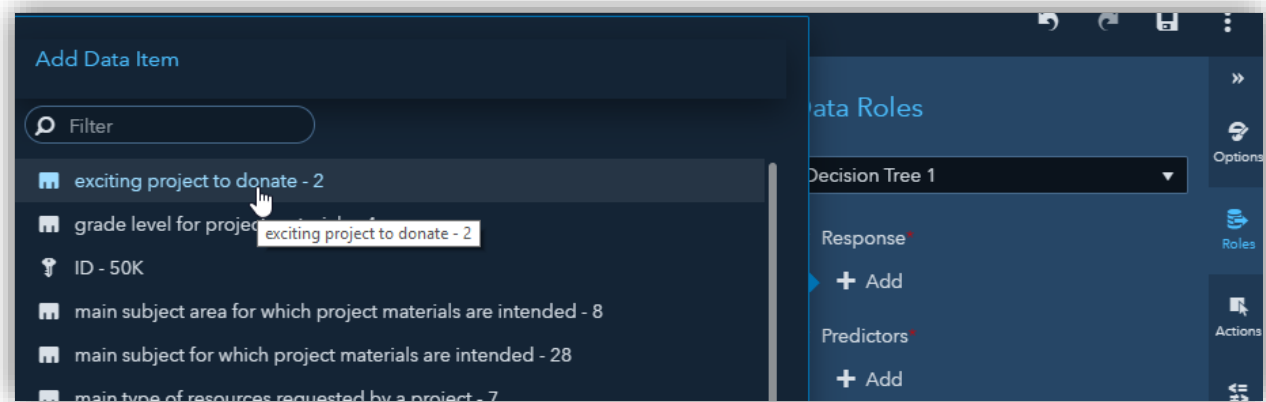


One important feature we have for SAS is the support of tree pruning based on validation data. Let us create a decision tree first.

- Add a new page 
- Drag a drop a [Decision Tree](#) object to page 2.



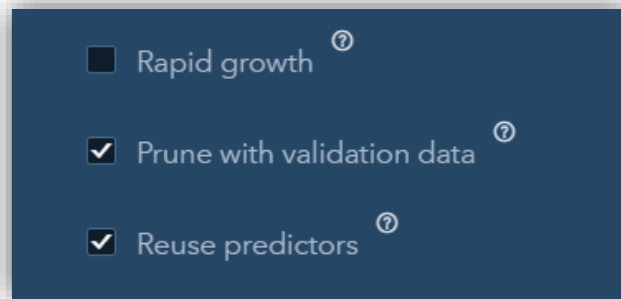
- Assign [Exciting Project to Donate](#) as the response variable of the decision tree model.




- Assign Partition Indicator as the partition ID of the decision tree model.
- Assign [all other columns except the ID](#) column as predictors. Note that you can use [SHIFT key](#) for batch selection.



The decision model by default prunes with validation data if partition ID is used. You can turn it off using the [Prune with validation data](#) option.




- Click the Maximize icon  to see the summary tables. You can see a table named [Cost-Complexity Pruning](#) which gives you an overview of the searching for the best complexity parameter. Actually, the model recommends complexity parameter = 0 in this case (**results will vary due to data distribution and sample partitioning**).

Node Statistics	Node Rules	Variable Importance	Cost-Complexity Pruning	Lift	ROC	Misclassification	Assessment Statistics
Best Tree	Complexity Parameter	Number of Leaves	Misclassification Rate	Validation Misclassification Rate			
Yes	0.0000	5	0.1615	0.1621			
	0.0003	1	0.1627	0.1627			

- Let us remove the most important predictor [cost of fulfillment](#) and see what happens. We generate a slightly larger tree and a slightly better error rate (from 16.21% to 16.14%).

Node Statistics	Node Rules	Variable Importance	Cost-Complexity Pruning	Lift	ROC	Misclassification	Assessment Statistics
Best Tree	Complexity Parameter	Number of Leaves	Misclassification Rate	Validation Misclassification Rate			
	0.0000	18	0.1598	0.1622			
	0.0000	16	0.1599	0.1616			
	0.0001	10	0.1605	0.1615			
Yes	0.0001	8	0.1607	0.1614			
	0.0001	6	0.1610	0.1615			
	0.0004	1	0.1627	0.1627			

Next, let us quickly build a logistic regression model with stepwise variable selection.

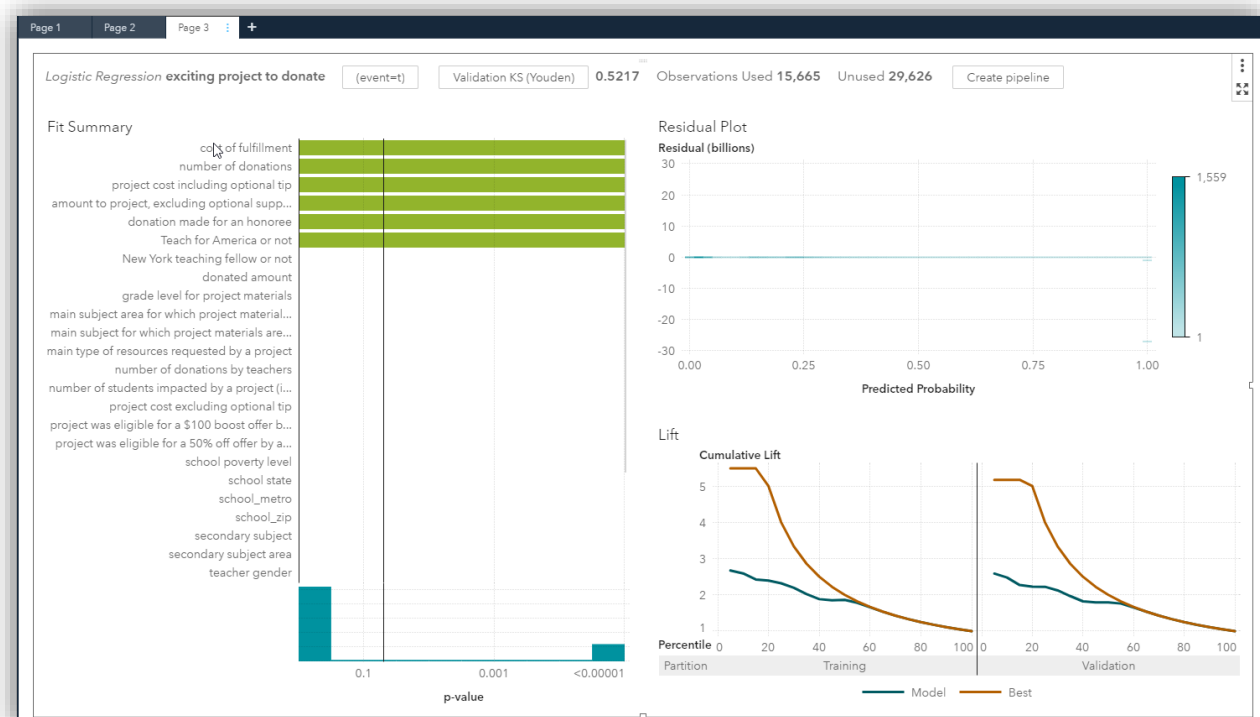
- Click the icon  to restore from maximized mode.
- Add a new page.
- Drag and drop a [Logistic regression](#) object to page 3.
- In the Options tab, turn on [Stepwise](#) variable selection. Like SAS/STAT, specifying Maximum Effects = 0 is equivalent to ignoring this option.

Variable selection method:
 Stepwise ▼

Maximum effects:
 0

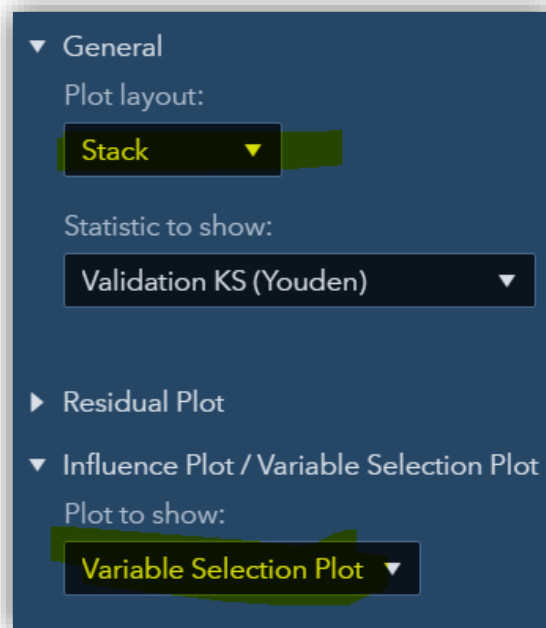
Selection criterion:
 SBC ▼

- Assign [Exciting Project to Donate](#) as the response variable of the logistic regression model.
- Assign [Partition Indicator](#) as the partition ID.
- Assign [all measure columns](#) as continuous effect. Note that you can use [SHIFT key](#) for batch selection.
- Assign [all categorical columns except the ID](#) column as classification effect.



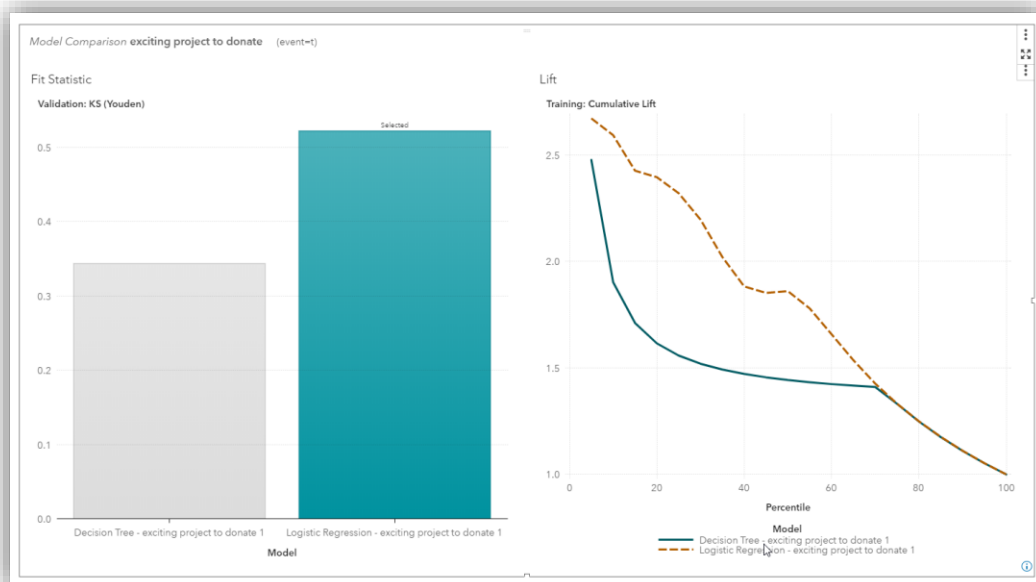
A logistic regression model is built. One new feature we added in this release is the iteration plot for variable selection.

- Go to Options tab and turn on [Variable Selection Plot](#). You can also maximize the chart by setting [Plot Layout](#) to Stack.



Now that we have created two models for Exciting Project to Donate, let us compare and select the best model using the Model Comparison object.

- Add a new page.
- Drag and drop a [Model Comparison](#) object to page 4. In the pop up window, click [Select All](#) and then [OK](#).
- Now that you can see Logistic Regression is selected based [Validation KS](#). If you change the Fit Statistics to [Validation Misclassification Rate](#), the decision tree model become the selected model.



- Once you select a best model, you can either pass the model to Model studio, register it with Model Manager, or download model score code.
- You can find these next steps by clicking the More icon

Automated Modeling with Pipelines



Model Studio is the interface that enables you to build pipelines to execute an automated sequence of steps to build models. Let's build a model from scratch.

- From the Actions (☰) menu on the top right, choose Build Models.
- Select New project. Select the **DonationsPrepped** table we created in Section 1.
- Name your Model Studio project "DonationsPipelines_<userID>
- Since we created our own partition variable during data preparation, myPartVar will automatically be used as the partition variable.
- Click Save to create the new project.

The data set has been analyzed and metadata associated with each variable is displayed.

- Review the data for this project in the Data tab
- You will see that it requires you to specify the target - select "Exciting_project" as the target

Variable Name	Label	Type	Role	Level	Order	Comment	Number of
<input type="checkbox"/> eligible_almost_home_match	project was eligible for a \$100 boost offer by a corporate partner	Character	Input	Binary	Default		2
<input type="checkbox"/> eligible_double_your_impact_matic	project was eligible for a 50% off offer by a corporate partner	Character	Input	Binary	Default		2
<input checked="" type="checkbox"/> Exciting_project	exciting project to donate	Character	Target	Binary	Default		2
<input type="checkbox"/> fulfillment_labor_materials	cost of fulfillment	Numeric	Input	Nominal	Default		4
<input type="checkbox"/> grade_level	grade level for project materials	Character	Input	Nominal	Default		4
<input type="checkbox"/> ID		Numeric	Key	Interval	Default		>254
<input type="checkbox"/> My_PartVar	Partition Indicator	Numeric	Partition	Nominal	Default		3
<input type="checkbox"/> n_corporate_match	whether a donation was matched 1-1 with corporate funds	Numeric	Input	Interval	Default		34
<input type="checkbox"/> n_donations	number of donations	Numeric	Input	Interval	Default		103
<input type="checkbox"/> n_honoree	donation made for an honoree	Numeric	Input	Nominal	Default		10
<input type="checkbox"/> n_teacher_donations	number of donations by teachers	Numeric	Input	Interval	Default		66

- Click on the target variable and note that, if desired, you can specify the target event level for a nominal target. This determines the value of the target that the model will predict a probability for (e.g., if the prediction is .8 and the event is exciting project = true, the model is stating there is an 80% chance that it is an exciting project)
- Find and select "myPartVar"
 - Click Map Partition Levels.
 - Ensure that the Training, Validation, and Test levels are as you defined them when you created the partition variable (1, 0, and 2 respectively)

Map Partition Levels

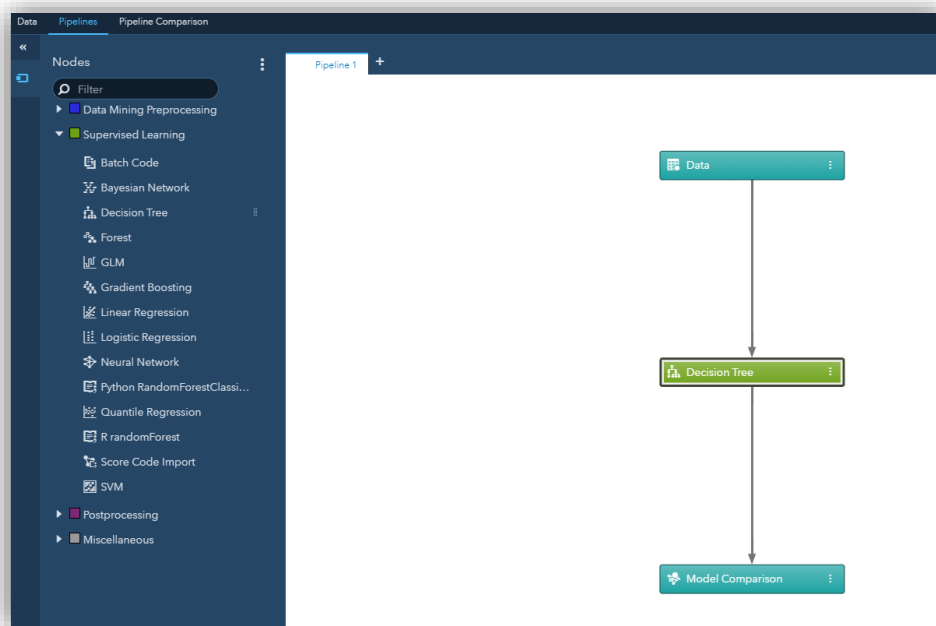
Training Level:

Validation Level:

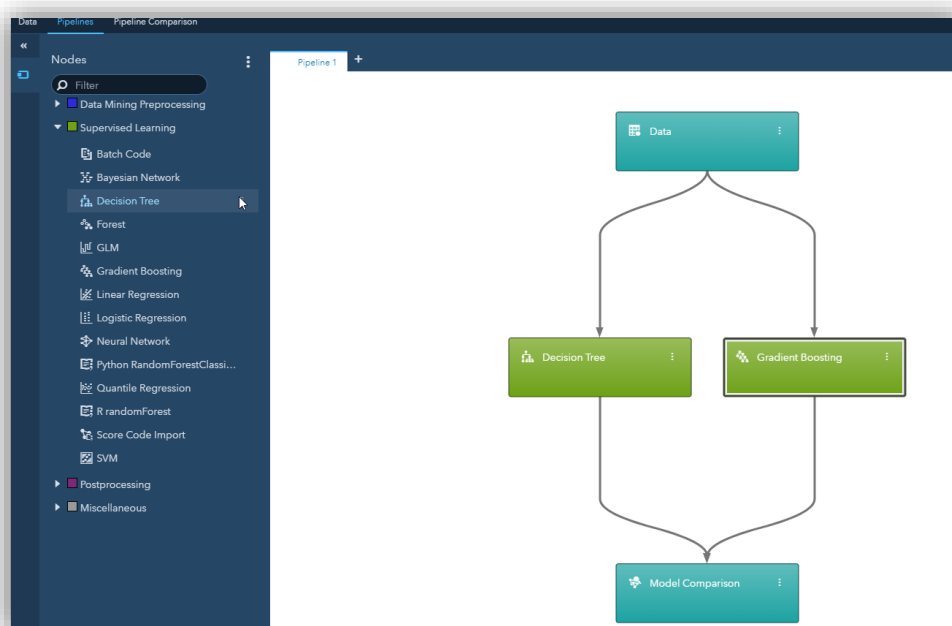
Test Level:

Now let's define a pipeline to build models in the Pipelines section.

- Let's start simple. Add a decision tree node to the pipeline such that it is right after the Data node that starts the flow (you can right click on nodes to see actions you can take (and don't forget the menu icon)...but dragging and dropping is also fun! Review the options for the nodes, but leave the options at default values.



- Review detailed information about the node using the information tab.
- Note that the decision tree node you just added is automatically connected to a Model Comparison node. at the end of the flow, allowing you to conveniently compare models very easily.
- Also add a Gradient Boosting model to the pipeline.



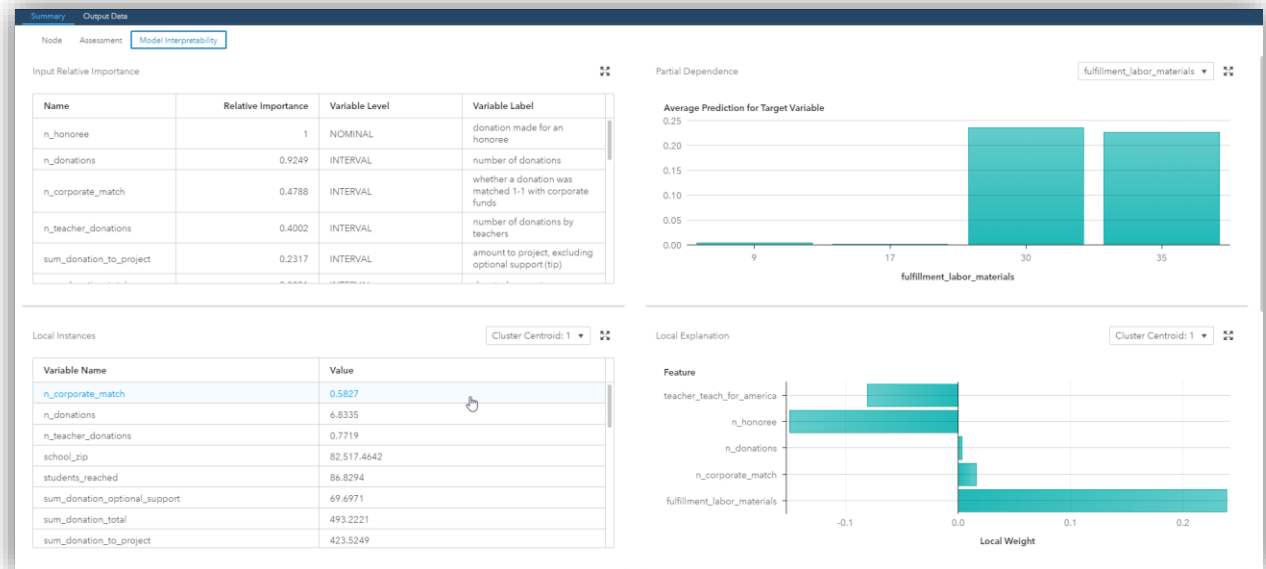


Interpretability of complex machine learning models is a major concern these days, especially in highly regulated industries and applications where discrimination and bias may be a concern. Unfortunately, the explanations are often buried within black boxes of mathematical and logical elements. There are techniques that can help shed some light on the behavior of these models, and in SAS these are provided as postprocessing steps within the modeling nodes.

- For the Gradient Boosting node, open the Model Interpretability options and select the 3 options under “Explain Model Inputs”; change the maximum number of variables to 10 for the plots.
 - Input relative importance table
 - Partial dependence plots
 - Cluster-based ICE plots
 - Also select ‘Explain Individual Predictions’ to perform LIME using cluster centroids.



On the information tab for this node (i) scroll to the last paragraph to read a brief explanation of these plots.



- Run the pipeline (it should take roughly a minute or so).
- Review the results of the Decision Tree using the menu icon on the node.
 - Expand the Tree Diagram using button and use the mouse wheel to zoom in to look at how it split the data at each node.
 - Look at the Variable Importance table to see which variables affected the target the most.
 - Look at the Train Code to see what SAS code was actually run to train the model.
- Check the logs for the Decision Tree. This is your standard SAS log...you'll just note that it's setting macro variables and processing many tables behind the scenes to actually manage all of the information in the pipeline.
- Review the results of the Gradient Boosting node.
 - Which variable is considered most important in predicting the target?
 - Take a look at the PD and ICE plots to get a sense of the relationships that the model is representing between input variables and the target.
- Review a comparison of the models in this pipeline using the Model Comparison node.
 - Which model is better based on misclassification rate?



-
- The screenshot shows the 'New Pipeline' dialog box in the Orange3 software. The dialog is titled 'New Pipeline' and has a close button (X) in the top right corner. It contains the following elements:
- Name:** A text field containing 'Pipeline 2'.
 - Description:** A text field with the placeholder text 'Enter description here'.
 - Template:** A dropdown menu that is currently open, displaying a list of template options:
 - Browse templates... (highlighted with a mouse cursor)
 - Blank Template
 - Advanced template for class target with autotuning
 - advanced_class_interpretability1_
 - Advanced template for class target
 - Intermediate template for class target
 - Browse templates... (highlighted)
- In the background, a workflow is visible with a 'Decision Tree' node (green) and a 'Model Comparison' node (dark green). Arrows indicate a flow from the 'Decision Tree' node to the 'Model Comparison' node, and a curved arrow points from the 'New Pipeline' dialog towards the 'Model Comparison' node.



- | Model Comparison | | | | |
|------------------|------------------------------|---------------------|-------------|------------------------|
| Champion | Name | Algorithm Name | KS (Youden) | Misclassification Rate |
| | Forward Logistic Regression | Logistic Regression | 0.3828 | 0.1624 |
| | Decision Tree | Decision Tree | 0.2858 | 0.1624 |
| | Stepwise Logistic Regression | Logistic Regression | 0 | 0.1627 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

- 15 -

Adding Interactive Models into Model Studio Project



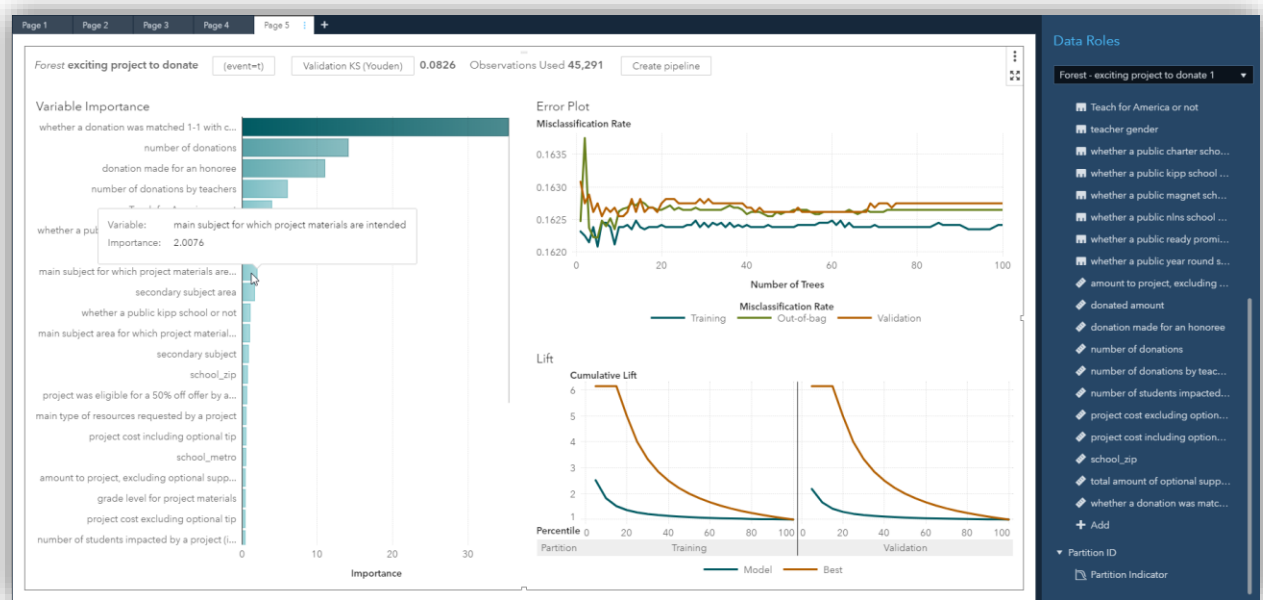
Building pipelines to train models in an automated fashion is great for productivity and repeatability. But you might also want to build models in a more interactive and experimental fashion.

- In the menu icon in the top left, select “Explore and Visualize Data”
- Select your “DonationsPrepped_<your user-ID>” table



You could explore the distributions of your variables using histograms and relationships among variables using scatter plots and other graphical objects, but we are going to focus on interactive modeling.

- From the “Objects” panel, add a Forest model to a new page.
 - NOTE: If you do not see this option on the right-click menu, you have hit a known bug. Sign out (using the menu in the upper right...select “Discard and Exit” when prompted) and sign back in again (select Yes to use the autosaved version of your work) and it should be fixed.
- From the Roles panel on the right side,
 - select the response (target) to be “exciting_project_to_donate”,
 - select the predictors (input variables) to be everything else **excluding ID and cost of fulfillment**.
 - Select Partition Indicator under the Partition ID role.
 - **Hint:** Select the first one (the text, not the checkbox) and type Ctrl+A to select all, then deselect individual variables.



Note: If the Roles panel says “Select an object to see its roles”, you need to click in the visuals on the page to activate them.



Note that once enough information is specified it will automatically begin the training process. Every change after that will retrain the model. Note at the top of the panel the accuracy of the model as determined by the KS statistic. Click on that and change it to Validation Misclassification Rate, something that is easier for most people to understand. Let’s try to improve it by tweaking the algorithm options, also known as tuning parameters or hyperparameters.

- In the Options for the Forest (make sure this object is selected in the page), try changing things like increasing the “number of trees” (try 150 or 200)
(Note: You can use the “Autotune” option to let it search for the best combination of option values for using a built-in optimization strategy...but don’t do that here, it takes a while since it is building many models to compare as it searches)



You could continue to interactively tweak your model here. But now let’s just pull this back into our automated modeling pipelines project so that we can compare it with the other models easily.

- Click the “Create pipeline” button at the top of the page. You can create a new project, but we want to add it to our existing project, so select that option and select the project when the list is provided.

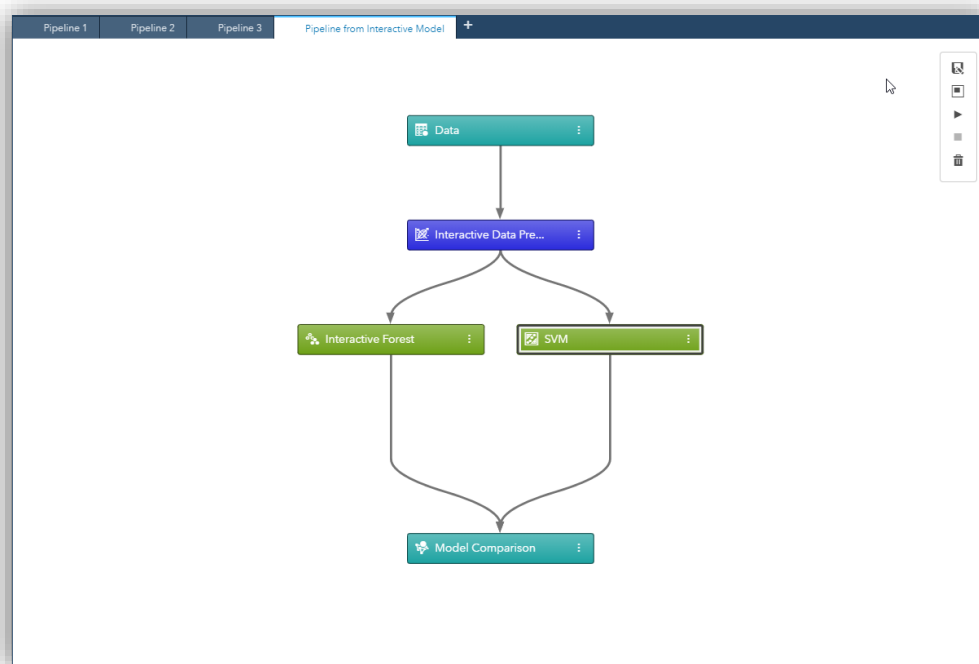


Note that the pipeline that is automatically created has nodes for (1) loading the data, (2) any data preparation that was done, and (3) the model that was trained interactively. The interactive data preparation and modeling nodes are not editable (as indicated in the description for them), but you can extend this pipeline as desired.



Note the properties when clicking on Interactive Forest. They match the properties run in Visual Analytics.

- Add a SVM modeling (supervised learning) node after the Interactive Data Preparation node so that you now have 2 branches for modeling – 1 being built when the pipeline executes (SVM) and one that was trained in your interactive modeling step previously.

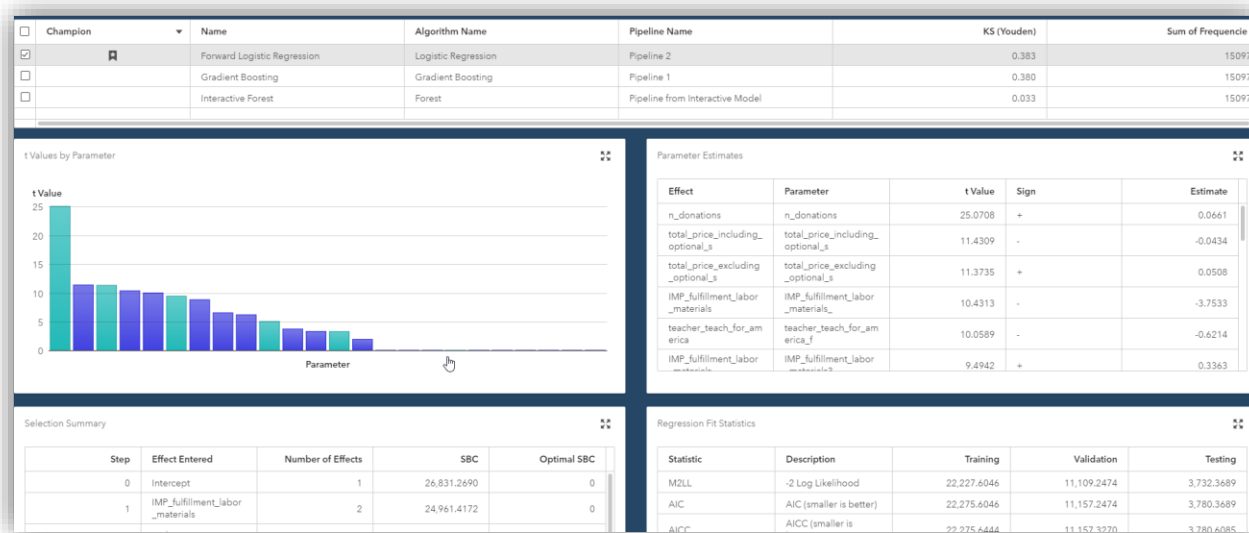


- Run this pipeline.



We have three different pipelines that build multiple predictive models and automatically determine which is the best (champion) from that pipeline. We can also compare models throughout the entire project (i.e., across pipelines).

- Select the Pipeline Comparison tab and review how it automatically compares the champion models from different pipelines.



Continue comparing these models.

- Change what columns are displayed in the table at the top to see other assessment statistics.
 - Misclassification Rate, Lift, and Area Under ROC are commonly used
- Select individual models to see the results particular to the selected model in the charts/tables below.
- Select multiple models (using the checkboxes) and click the Compare button to compare them.

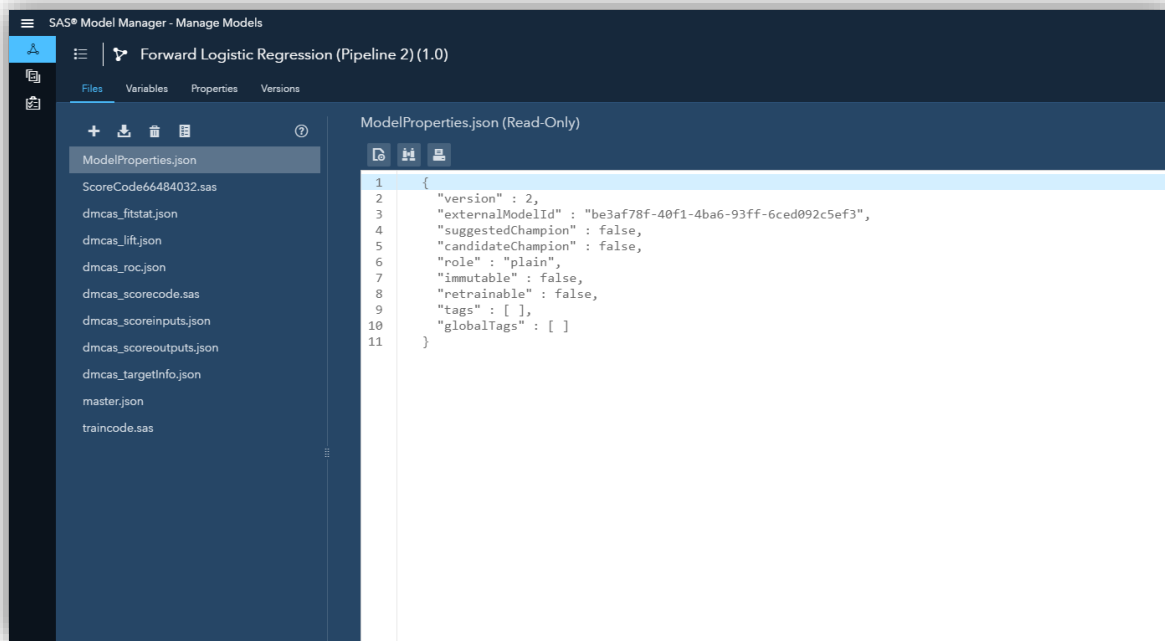


Note how, for any selected model, you can download the score code or a score API in various languages, allowing you to embed the scoring of models in your own applications. We will come back to this in a dedicated section later.



Ultimately, when you have a good model you will want to deploy it into your production environment. Registering your model in a common model repository is the first step toward that.

- Select one of your models and register it.
- Once it is done registering, select the option to Manage Models to view the registered model in SAS Model Manager.



Feel free to take a quick look at the various files and other metadata registered for the model in SAS Model Manager, but don't spend much time there (this is not a Model Manager enablement session).

- Use the applications menu in the upper left to return to Model Studio ("Build Models").

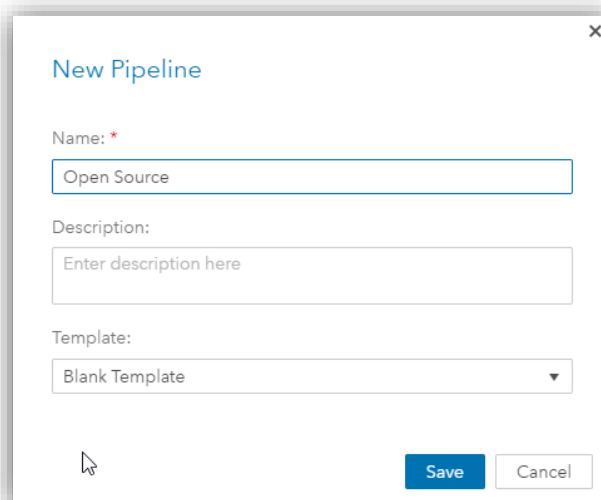
Adding Open Source Models into Model Studio Project



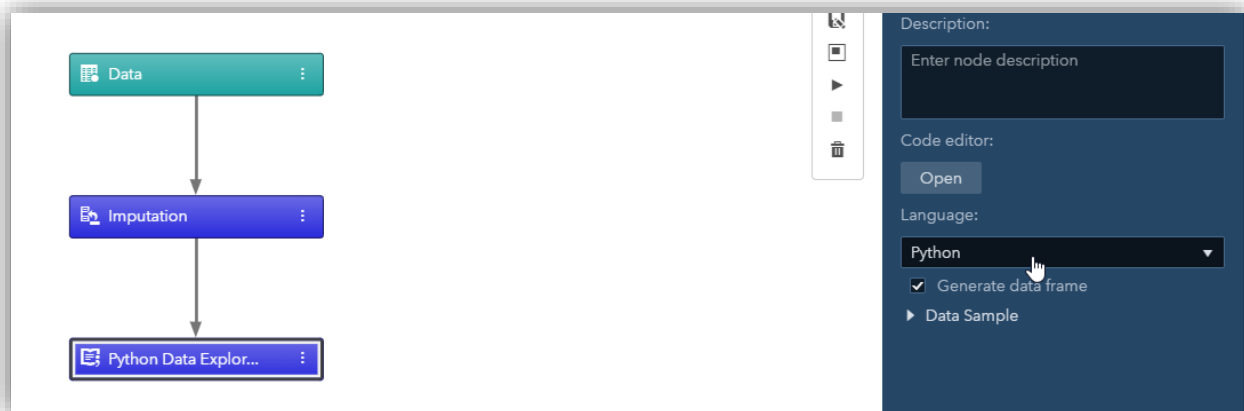
There may be times when it is beneficial to add components of open source technologies into Model Studio. We now make this easily available for Python and R scripts that either help you explore data or build models within a pipeline.

First let's start with some data exploration. We will be leveraging nodes previously saved to the exchange to alleviate copying and pasting.

- Add a new pipeline named Open Source to your existing donations project using a Blank Template.



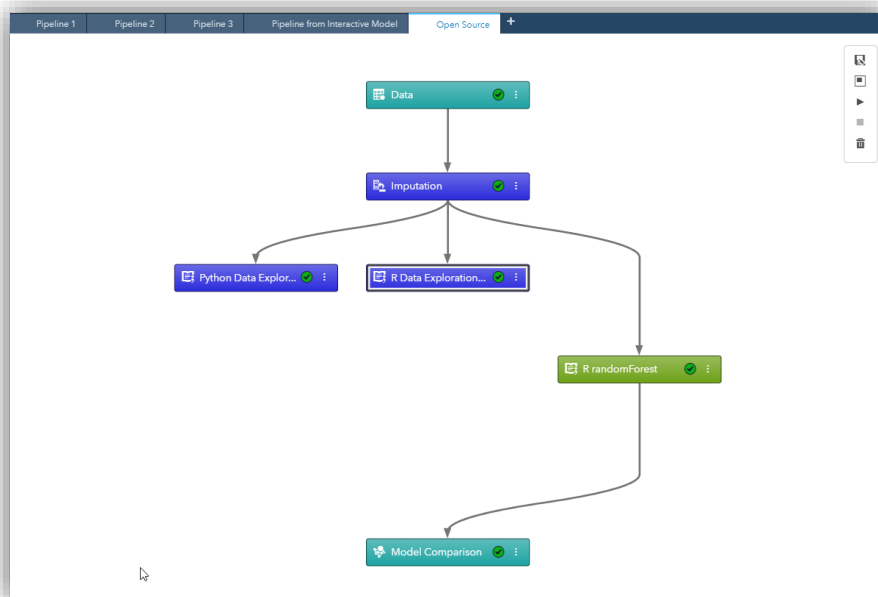
- Add an imputation node from Data Mining Preprocessing. Many open source packages do not like missing values.
- Insert “Python Data Exploration: Clustering” open source code node from Miscellaneous below the imputation node. Verify the language is set to Python. Note: we are sampling to use 10,000 observations, but have the option to use all the data. Open the node, and notice how your visualization needs to be saved as a file in the node directory (line 23).



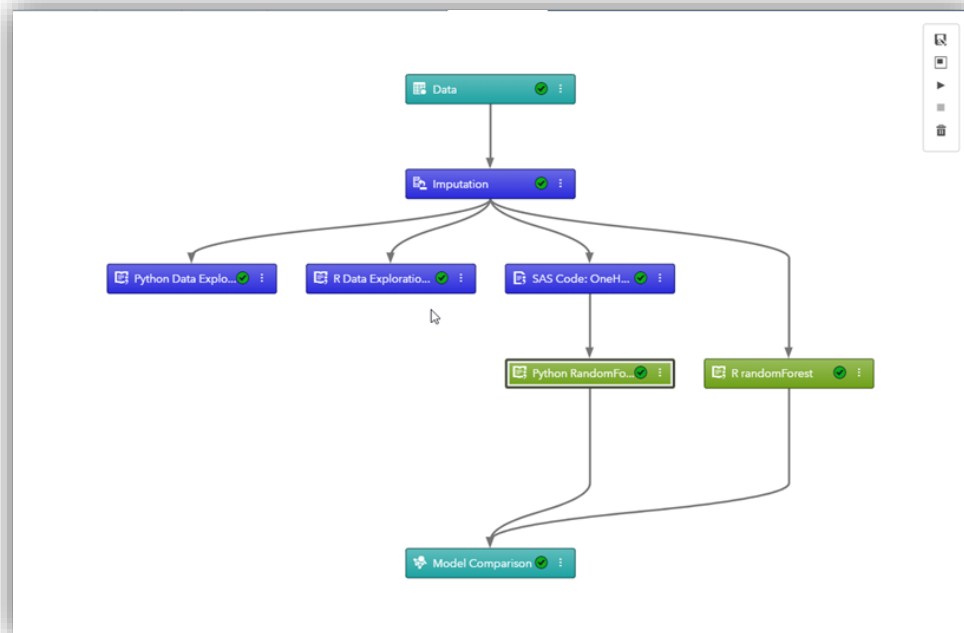
- Insert “R Data Exploration: SVD” open source code node from Miscellaneous below the imputation node. Make sure the language is set to R (the open source code node defaults to Python). Copy and paste R code to visualize the first two singular values of the SVD.
- Run your pipeline and check out the visuals in the results of the open source nodes by right clicking and the individual nodes and selecting Results.

For the sake of this example we want to try other versions of a random forest.

- Insert “R Supervised Learning: randomForest” open source code node from Supervised Learning below the imputation node. Again, make sure the language is set to R. Examine the code. Note: The piece of code that needs to be changed for different datasets is line 18 (how your predictions are named using the P_ + “target”... naming convention).



- Insert “SAS Code: OneHot Encoding” from Miscellaneous below the imputation node. Note: many Python modelling packages do not like class variables. Note: the code is also located here: https://raw.githubusercontent.com/sassoftware/sas-viya-dmml-pipelines/master/sas_code_node/class_level_indicators/classLevelIndicators.sas.
- Insert Python RandomForestClassifier open source code node below from Supervised learning below the “SAS Code: OneHot Encoding” node. Verify the language is set to Python. Examine the code. Again, the piece that needs to be changed for different datasets is line 18 (how your predictions are named using the P_ + “target”... naming convention).



Run the pipeline to compare models. Which is the champion?

Model Comparison				
Champion	Name	Algorithm Name	KS (Youden)	Misclassification Rate
	Python RandomForestClassifier	Python RandomForestClassifier	0.4739	0.1568
	R RandomForest	R RandomForest	0.4503	0.1520

Properties	
Property name	Property value
selectionCriteriaClass	Kolmogorov-Smirnov statistic (KS)
selectionCriteriaInterval	Average squared error
selectionTable	Validate
selectionDepth	10
cutoff	0.5000

- Bonus: Try using an open source code node and build a Gradient Boosting Classifier in Python. Make sure you change the node to Supervised Learning by right clicking on the node. Does adding gradient boosting improve performance?

Scoring with Models built in Model Studio




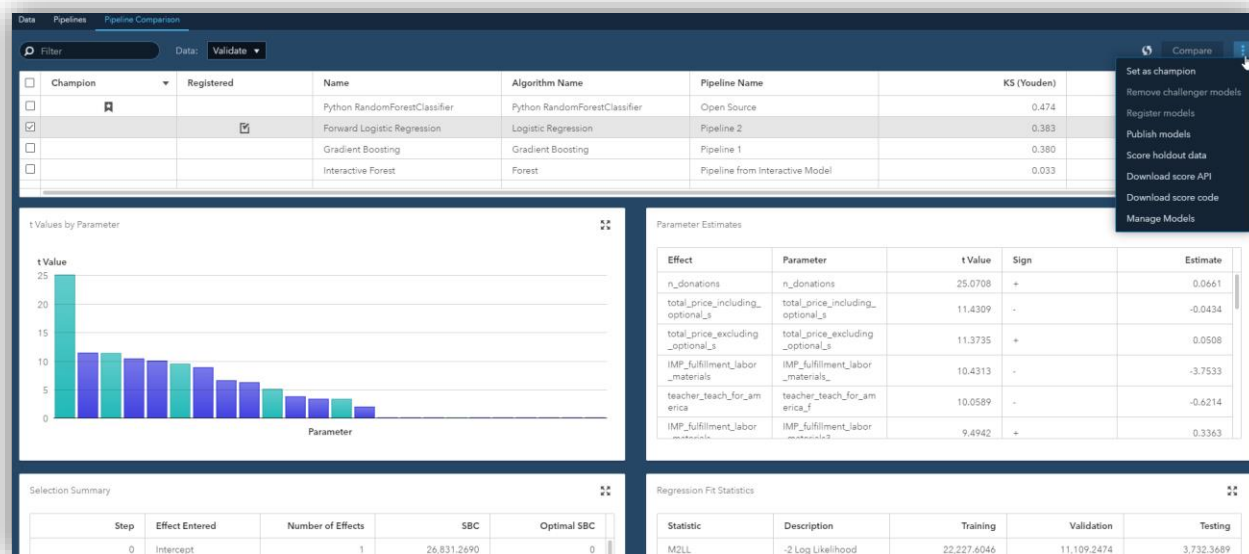
The purpose of building predictive models is to score new observations for which you do not yet know the target value. SAS® Visual Data Mining and Machine Learning provides a few different ways to use your models to score new data.

Invoking score code directly

Your models can be used to score new observations by using SAS code to invoke the models. This code can be in 2 forms: code that can be run directly in a **data step**, and code that is in a binary format known as an **"astore"** (analytical store), which was designed and developed to accommodate larger, more complex models. In either case, the entire flow of nodes in the pipeline up to the node for the selected model is accounted for in the score code.

Note: In the environment for this session, access to files on the server is not allowed. Invoking an astore model requires you to point to an *epscorecode* (embedded process score code) file that you download to the client machine. Since this server cannot access that file, we will limit this lesson to scoring non-astore models. Be sure to select a non-astore model such as a decision tree or a regression model for the following steps; if you do not have one listed in the table of models on the "Pipeline Comparison" tab, find one in a pipeline, right-click on it, and select "Add challenger model".

- Select "Download score code" from the  menu in the upper right, which will download it to your local machine. This is provided as a zip file which you must extract. Once extracted, open the file and inspect it to see the concatenated score code from all nodes leading up to and including the modeling node.



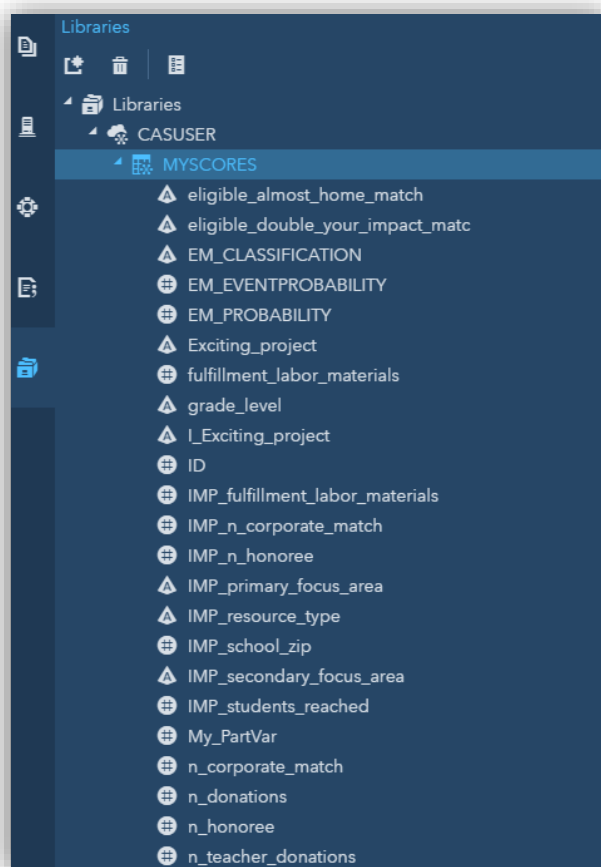
- In the Applications menu in the upper left, select “Develop SAS Code”
- Create a new program (New→Program) and start a CAS session and issue a command that will show you all the libraries (listed on the left)
- In the program editor, type the following

```
cas;  
caslib _all_ assign;  
data casuser.myscores;  
    set public.donations_to_predict;  
    <copy and paste your score code here>  
run;
```



Note: Typically you would not copy and paste all the score code but instead would %include <path to your code on disk>; but as previously mentioned, this server cannot access your local machine.

- Submit this code to run, refresh the list of tables in the Libraries on the left, and review the scored data set that was generated by double-clicking it to open it. In this case, you can sort the “I_Exciting” column (right-click on the column header) to see the projects that were predicted to be exciting.





As mentioned, this environment does not allow access to files on your machine, so we cannot reference the epscorecode downloaded for any astore models. In a real environment with some sort of file system access provided, you would simply invoke your astore models as follows:


```
proc astore;
  score data=casuser.data2score
    rstore=<location of astore binary model>
    epcode="<location of epscorecode>"
  out=casuser.scoreddata;
quit;
```




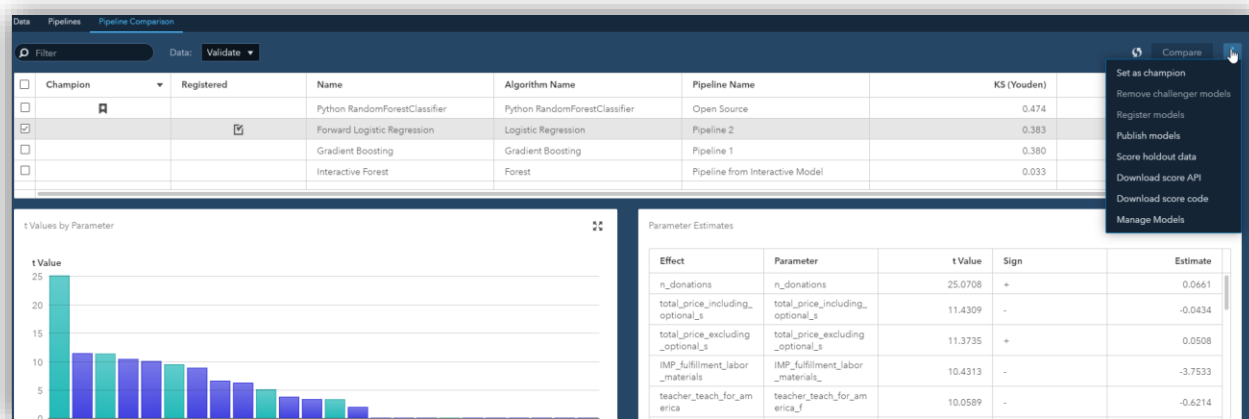
The rstore is a reference to the “remote” store model that is a table created in CAS in the “Models” caslib when you download the score code; it contains the binary form of the model and must be used in conjunction with the epscorecode. Feel free to download the score code for an astore model and see this in the Models caslib and look at the epscorecode file...you just won’t be able to run the proc astore code above.

Calling an API to invoke a scoring service



Your models can be accessed for scoring new observations via a web service call, giving you the ability to score new observations from any application that can issue a REST call (which most modern languages provide). Model Studio gives you the option to download code for the raw REST call itself, or the call wrapped in SAS or Python. Here we will just use the SAS code to call the REST API using proc http.

- Select “Download score API” from the  menu in the upper right



- Take a look at the Python and REST versions, but select SAS and click Download
- Open the downloaded file and copy the text from it
- In Applications menu in the upper left, select “Develop SAS Code”
- If you haven’t already done so, create a new program (New→Program) and start a CAS session and issue a command that will show you all the libraries

```
cas;
caslib _all_ assign;
```

- Paste the API code into the program editor, and find the following lines and set the values as shown here

```
%let hostname = <your server>.exnet.sas.com;
```

```
%let port = 80;  
  
%let outputCasLib = casuser;  
%let outputTableName = myscores;
```

- In the Libraries on the left, refresh the list and see if you have a new scored data table. Open it and view the predictions.

Free Play



Now that you have gone through the entire application from end-to-end...data preparation→visual exploration and interactive modeling→automated modeling→model deployment...we'd like you to focus on data mining and machine learning pipelines and take the rest of the time to play around with whatever part of the pipeline application interests you.

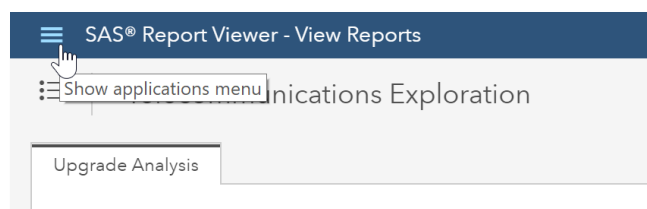
SAS Drive



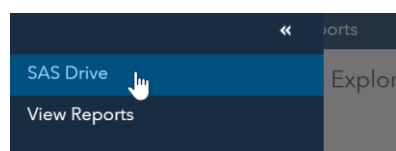
SAS Drive is the landing page that allows users to customize access to information and create shortcuts to SAS components. From there, depending on their roles, users can go to different components for different tasks, which varies from managing the system and tables to analyzing data and creating reports and dashboards, viewing reports and accessing SAS Studio code.

With SAS Drive users see, organize and collaborate on their work. Users can favorite, share, preview and tag their content from one place. They can create projects that share data, content and other resources with project members. A search engine makes it easier to find assets that other users created and shared.

- Click the three stacked lines (**Show applications menu**) at the top left of the screen to go to SAS Drive.



- Click on **SAS Drive**.



Thank you for taking the time to walk through this demo. We hope you found it helpful. Check out the other options available for the report and SAS Drive! Have fun exploring SAS Visual Data Mining and Machine Learning!