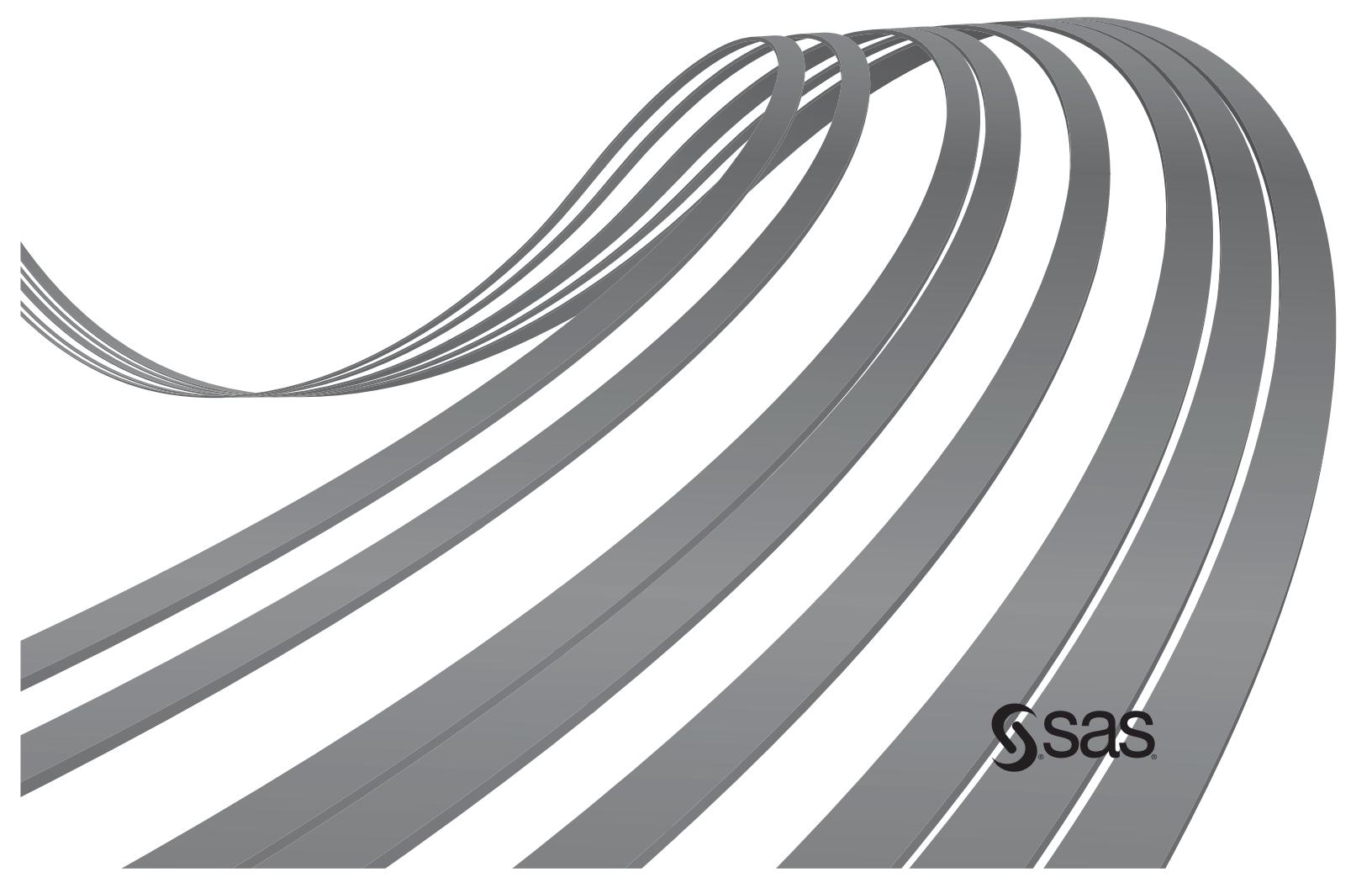


sas **innovate**

SAS® Viya® Tour: What Makes SAS® Viya® Special

Lincoln H. Groves
Senior Manager, Analytical Education



sas®



SAS Software Tour with iLink Mortgage, Inc.

SAS On-the-Job Activity

Purpose

This tour of SAS Viya comes from a teaching-and-learning activity produced by the Global Academic Programs team at SAS Institute Inc. In this *SAS On-the-Job* series, you'll get the opportunity to explore SAS Studio, SAS Visual Analytics, and SAS Model Studio, while assuming the role of Day 1 analyst at a fictitious mortgage company. As "students" – or "new employees" – you will modify and prepare data, create descriptive visualizations, estimate and compare machine learning models, and build and enhance visual reports.

SAS Software

We use SAS Viya 4 in this activity. In this tour, you'll explore SAS Studio, SAS Visual Analytics, and SAS Model Studio across a variety of tasks in the data analytics life cycle.

Industry Alignment

This On-the-Job activity fits mostly closely within the mortgage industry. We explore data containing information about homeowner loan activity – and seek to predict which customers are most likely to default on their loans. That stated, the general example can be applied to any industry where you are modeling a yes/no outcome.

Table of Contents

SAS Software Tour with iLink Mortgage, Inc.	3
<i>Purpose</i>	3
<i>SAS Software</i>	3
<i>Industry Alignment</i>	3
Activity Notes and Requirements	5
<i>Learning Objectives</i>	5
<i>Estimated Completion Time</i>	5
<i>Experience Level</i>	5
<i>Prerequisite Knowledge</i>	5
Software	5
Content Knowledge	5
<i>Additional Notes</i>	5
Required Setup	5
Task 1: Examining Alex's Work in SAS Studio	6
<i>Learning Objectives</i>	6
<i>Estimated Time of Completion</i>	6
<i>Task: Introduction to SAS Studio</i>	6
Task 2: SAS Visual Analytics – A Non-Coders Paradise	17
<i>Learning Objectives</i>	17
<i>Estimated Time of Completion</i>	17
<i>Task: Introduction to SAS Visual Analytics</i>	17
Task 3: Going Deeper with SAS Model Studio	42
<i>Learning Objectives</i>	42
<i>Estimated Time of Completion</i>	42
<i>Task: Learn to love SAS Model Studio</i>	42
Appendix	61
<i>Appendix A: Access Software</i>	61
<i>Appendix B: Helpful Documentation</i>	61
<i>Appendix C: Recommended Follow-up Learning</i>	62

Activity Notes and Requirements

Learning Objectives

This activity provides the opportunity for students to learn and practice skills such as

- creating visualizations for descriptive statistics
- modifying data used in SAS Viya
- building and comparing machine learning models.

Estimated Completion Time

This activity contains three tasks. All together this activity will take students approximately 90 to 120 minutes to complete.

Experience Level

This activity is designed for students new to SAS Viya. Students are encouraged to continue their SAS journey by visiting one of the multiple learning paths in Appendix C: Recommended Learning.

Prerequisite Knowledge

Software

No prior experience is needed in SAS Viya.

Content Knowledge

Students should have basic experience/knowledge of the following concepts:

- statistical graphs like bar charts, histograms, and pie charts
- statistical concepts like frequency, percentages, and mean (averages)

Additional Notes

Required Setup

Students need to complete each task sequentially to complete the activity as written.

Task 1: Examining Alex's Work in SAS Studio

Learning Objectives

This task provides the opportunity to learn and practice skills such as

- accessing and navigating SAS Viya
- learning to use Snippets in SAS Studio
- running SAS Studio programs and examining results for key insights.

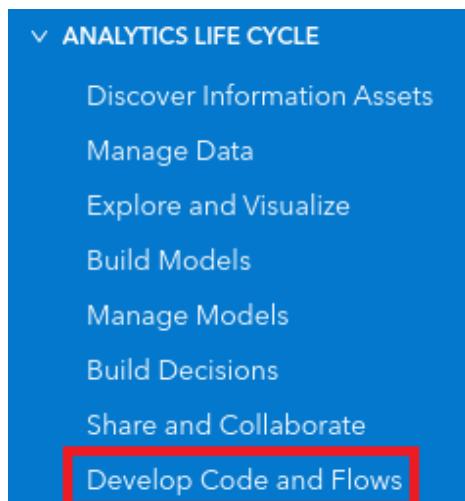
Estimated Time of Completion

This task is estimated to take about 30 minutes.

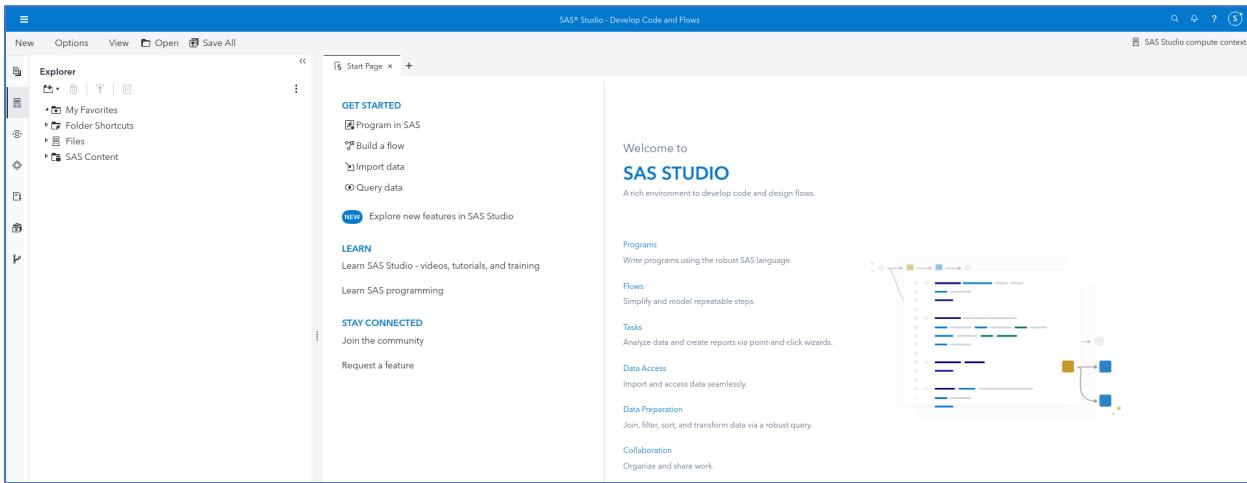
Task: Introduction to SAS Studio

Welcome to your first day at iLink Mortgage, Inc.! Your onboarding as a machine learning apprentice starts with an examination of your predecessor Alex's work. In particular, I'd like you to look at one of their SAS Studio projects, where they sought to predict which home loans were most likely to default. Defaulted loans cost our shareholders money, so we'll want any insights possible to determine which loans might be in trouble.

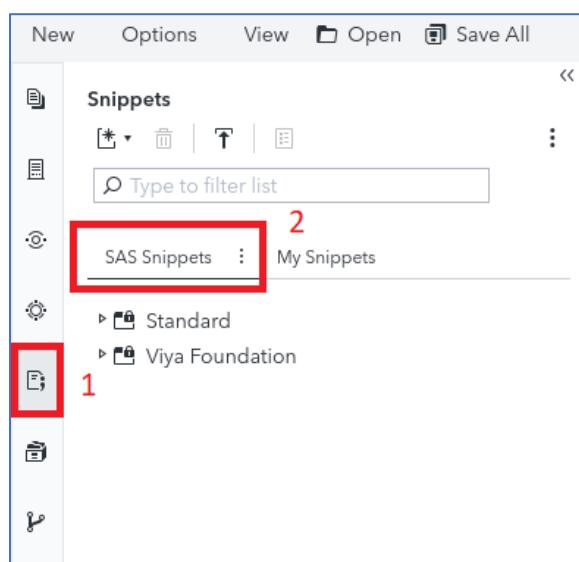
1. Log in to SAS Viya.
2. You are brought to SAS Drive, which is your starting point in SAS Viya.
3. From the top left corner, select the **Applications** menu  and click **Develop Code and Flows** from the Analytics Life Cycle options.



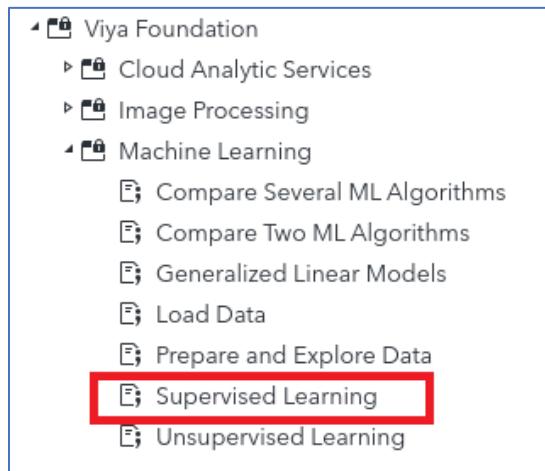
4. The Welcome to SAS Studio page should now be displayed.



5. So, welcome! From the **Start Page**, you can begin your analytical adventure in a variety of ways, from programming in SAS Studio to building flows. You can even embark on a learning journey with videos, tutorials, and training.
6. For this analysis, we'll simply access a program Alex saved as a snippet (that is, a chunk of reusable code). From the left pane, access the snippets saved in your SAS Studio environment. **SAS Snippets** should be selected by default.



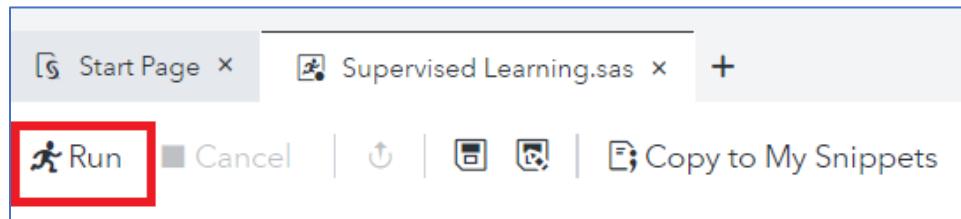
7. Expand the sections for **Viya Foundation** and then **Machine Learning**. Next find, and open, the **Supervised Learning** snippet.



8. The snippet starts with the following preamble:

```
Start Page Supervised Learning.sas +  
Run Cancel ⌂ ⌂ Copy to My Snippets + Code to Flow ⌂ ⌂ ⌂ Debug  
Code  
1  /**************************************************************************/  
2  /* This snippet showcases a sample Machine Learning workflow for */  
3  /* supervised learning using SAMPLEML.HMEQ data set. The steps include: */  
4  /* */  
5  /* (1) PREPARE AND EXPLORE */  
6  /* a) Load data set into CAS */  
7  /* b) Explore */  
8  /* c) Partition */  
9  /* d) Impute */  
10 /* e) Identify variables that explain variance */  
11 /* */  
12 /* (2) PERFORM SUPERVISED LEARNING */  
13 /* a) Fit model using random forest */  
14 /* */  
15 /* (3) EVALUATE AND IMPLEMENT */  
16 /* a) Score the data */  
17 /* b) Assess model performance */  
18 /* c) Generate ROC and Lift charts */  
19 /**************************************************************************/  
20  
21  
22 /**************************************************************************/  
23 /* Define the macro variables for later use in the program */  
24 /**************************************************************************/  
25 /* Specify a folder path to write the temporary output files */  
26 %let outdir = &_SASWORKINGDIR;  
27  
28 /* Create a CAS engine libref to save the output data sets */  
29 %let caslibname = mycas;  
30 libname &caslibname cas caslib=casuser;
```

9. Please take a moment to read Alex's notes. This program does several things, as it takes raw data and uploads it into the SAS Cloud, prepares the data, and then runs and analyzes a random forest model. The general analytical goal is to predict which loans will default, which occurs when the variable **BAD = 1**. BAD, in this case, is literally not good.
10. With a better understanding of our analytical tasks, let the program rip! In the program window for **Supervised Learning.sas**, click the **Run code** icon.



11. Examine the **Log** tab. Ensure that the program ran without any issues*.

The screenshot shows the SAS Log tab with the following interface elements:

- Top navigation bar: Log (selected), Results
- Status indicators: Errors (0), Warnings (0), Notes (94)
- Main area: "There are no messages."
- Log output area:

```
107 /* Specify a folder path to write the
108 %let outdir = &_SASWORKINGDIR;
109
110 /* Create a CAS engine libref to save
111 %let caslibname = mycas;
112 libname &caslibname cas caslib=casuser
NOTE: The session CASAUTO connected successfully.
      UUID is 10ab6a77-26b7-9c49-a2a8-ac904.
      CASUSER(lincoln.groves@sas.com).
NOTE: The SAS option SESSREF was updated successfully.
NOTE: The SAS macro _SESSREF_ was updated successfully.
NOTE: The session is using 0 workers.
NOTE: Libref MYCAS was successfully assigned.
      Engine:          CAS
      Physical Name:  10ab6a77-26b7-9c49-a2a8-ac904.
113
114 /* Specify the data set names */
115 %let sasdata           = sampsio.hmeq;
116 %let casdata           = &caslibname..hmeq;
117 %let partitioned_data = &caslibname..partitioned_hmeq;
118
119 /* Specify the data set inputs and target variables */
120 %let class_inputs      = reason job dero;
121 %let interval_inputs   = loan value IM c...
```

* Check that the log contains no Errors or Warnings.

12. Now let's examine the results. From the rightmost pane, select **Results**.

The screenshot shows the SAS software interface with the 'Results' tab selected in the top navigation bar. The main pane displays a table with 11 rows of variable metadata:

Obs	_VARNAME_	_FMTWIDTH_	_TYPE_	_RLEVEL_	_ORDER_	_MORE_	_CARDINALITY_	_NOB
1	MORTDUE	12	N	INTERVAL	ASC	Y		20 5€
2	VALUE	12	N	INTERVAL	ASC	Y		20 5€
3	REASON	7	C	CLASS	ASC	N		2 5€
4	JOB	7	C	CLASS	ASC	N		6 5€
5	YOJ	12	N	INTERVAL	ASC	Y		20 5€
6	DEROG	12	N	CLASS	ASC	N		11 5€
7	DELINQ	12	N	CLASS	ASC	N		14 5€
8	CLAGE	12	N	INTERVAL	ASC	Y		20 5€
9	NINQ	12	N	CLASS	ASC	N		16 5€
10	CLNO	12	N	INTERVAL	ASC	Y		20 5€
11	DEBTINC	12	N	INTERVAL	ASC	Y		20 5€

13. Those results feel a bit cramped, right? Let's open the **Results** in a new browser tab. Follow the clicks below:

The screenshot shows the SAS software interface with the 'Results' tab selected in the top navigation bar. A context menu is open over the results table, with the 'Results' option highlighted with a red box and a red number 2. Other options in the menu include 'Summary', 'Code', 'Log', 'E-Mail', 'Print', 'Download', 'Tab layout', and 'Refresh'. The top right corner of the interface shows the date and time: Feb 26, 2024, 6:41:07 PM.

14. Examine the model output, including how the data are transformed and which plots are used to assess the model's performance. In other words, scroll, and scroll some more!

15. Find and record the model **Fit Statistics**.

Fit Statistics					
Number of Observations	Divisor of Average	Squared Error		Mean Consequential Error	Multiclass Log Loss
		Average	Root Average		
1788	1788	0.081757	0.285932	0.116890	0.270700

Note: Your results might differ marginally, as we didn't set a seed to ensure that the results were perfectly replicable. No worries though! This also illustrates that different training/validation/testing samples will produce slightly different results.

16. Now let's modify the model slightly and see whether we get a better random forest model. Return to your SAS program and find the PROC FOREST code that starts on **line 128**. Locate the portion of the code that references the minimum leaf size for the individual trees within the forest model.

```

125  *****/
126  /* Build a predictive model using Random Forest */
127  *****/
128  proc forest data=&caslibname.._prepped ntrees=50 numbin=20 minleafsize=5;
129    input &interval_inputs. / level = interval;
130    input &class_inputs. / level = nominal;
131    target &target / level = nominal;
132    partition rolevar=_partind_(train='1' validate='0');
133    code file="&outdir./forest.sas";
134    ods output FitStatistics=fitstats;
135  run;

```

17. Let's change **minleafsize=50**. Then resubmit the code using  Run .

18. Does this model fit the data better, as indicated by the average squared error? My findings:

Fit Statistics					
Number of Observations	Divisor of Average	Squared Error		Mean Consequential Error	Multiclass Log Loss
		Average	Root Average		
1788	1788	0.102914	0.320802	0.138702	0.334126

19. Turns out that increasing the minimum leaf size to 50 does not improve model fit in this scenario. Can't win them all.

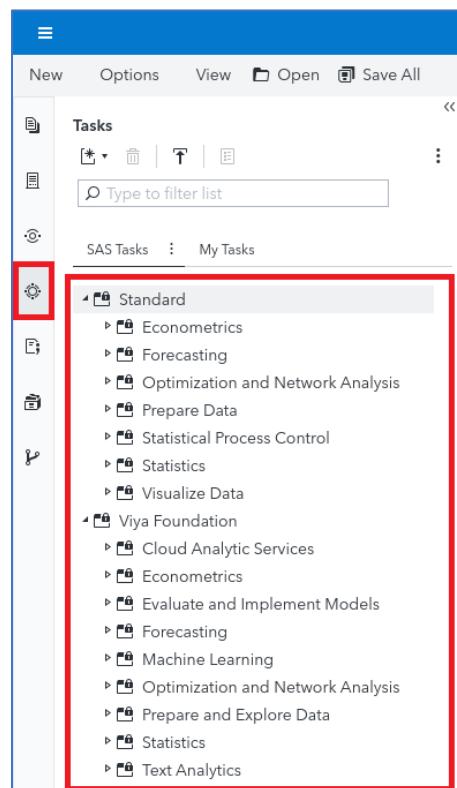
20. Finally, we would like to use the data created from this program in two other SAS applications: SAS Visual Analytics and SAS Model Studio. Because we don't want to spend much time cleaning the data again, let's examine which data sets have been saved for us, by first looking at the data set used in the forest modeling.
21. I don't expect you to read SAS code already – particularly SAS macros and that “&” notation – but we can see from the prompts in the screenshot below that the table name is **_prepped**. Moreover, the data reside in the SAS library **MYCAS**. The evidence, by numbers:

The screenshot shows the SAS Studio interface. On the left, the **Libraries** pane is open, displaying a tree view of connected libraries. A red box labeled '2' highlights the 'Libraries' icon. Another red box labeled '3' highlights the **MYCAS** library node. Inside the MYCAS node, a red box labeled '4' highlights the **_PREPPED** data set. In the center-right, the **Supervised Learning.sas** window is open, showing SAS code. A red box labeled '1' highlights the line of code:

```
proc forest data=&caslibname.._prepped ntrees=50 nmbin=20 minleafsize=50;
```

Note: Step 1 is in the programming window. Steps 2-4 are focused on the Libraries pane.

22. We'll need to save and promote **_PREPPED** to the shared CAS folder. The easiest way to do this is to use a SAS Studio task. "What's a SAS Studio task?", you're bound to ask. Well, it's a very useful tool that helps write the SAS code for you. And it is a particularly useful tool for those either (1) learning SAS code for the first time or (2) using a SAS procedure for the first time. You can find SAS Studio tasks in the left pane. I'll expand the **Standard** and **Viya Foundation** tasks so that you can better understand the breadth of support available.



23. Under **Viya Foundation**, expand the **Cloud Analytic Services** section. Open the CAS Operations task.

The screenshot shows the SAS Studio interface. On the left, the 'Tasks' sidebar is open, showing various categories like Standard, Viya Foundation, and Cloud Analytic Services. Under Cloud Analytic Services, 'CAS Operations' is selected and highlighted with a red box, labeled '1'. In the center, the 'Operations' tab of the CAS Operations task is active, with three sections: 'CAS Connection' (Session name: CASAUTO), 'CAS Object' (Specify the type of object to operate on: Session), and 'Operation' (Specify the operation: List information about session). This middle section is also highlighted with a red box, labeled '2'. On the right, the 'Log' tab is open, displaying generated SAS code. The code is as follows:

Code	Log		
1	/*		
2	*		
3	* Task code generated by SAS® Studio 6.0		
4	* Generated on '2/26/24, 6:58 PM'		
5	* Generated on SAS version 'V.04.00M0P121123'		
6	* Generated on browser 'Mozilla/5.0 (X11; Linux		
7	8	9	*/
10	cas CASAUTO list;		
11			

The 'Log' column is labeled '3'.

24. A couple of interesting notes. In (2) above, you'll find the setting that you can adjust for the task. As you adjust these settings, the code will automatically be generated in section (3). Yes, with the click of a few buttons, the SAS code will be generated for you!

25. Let's illustrate with an example. For our other two parts, we want to access a global CAS table, meaning that it's been promoted to be available everywhere. So, under **Specify the type of object to operate on**, select **Data (table, file, view)**. Next, under **Operation >> Specify the operation**, choose **Promote table to global**. Finally, the caslib will be **casuser** and – as noted – the file is **_prepped**. These appear in both the **From** and **To** sections, with the cumulative changes as follows:

▼ CAS Connection
Session name:

▼ CAS Object
Specify the type of object to operate on:

▼ Operation
Specify the operation: *

▼ From
caslib:
Table/File: *

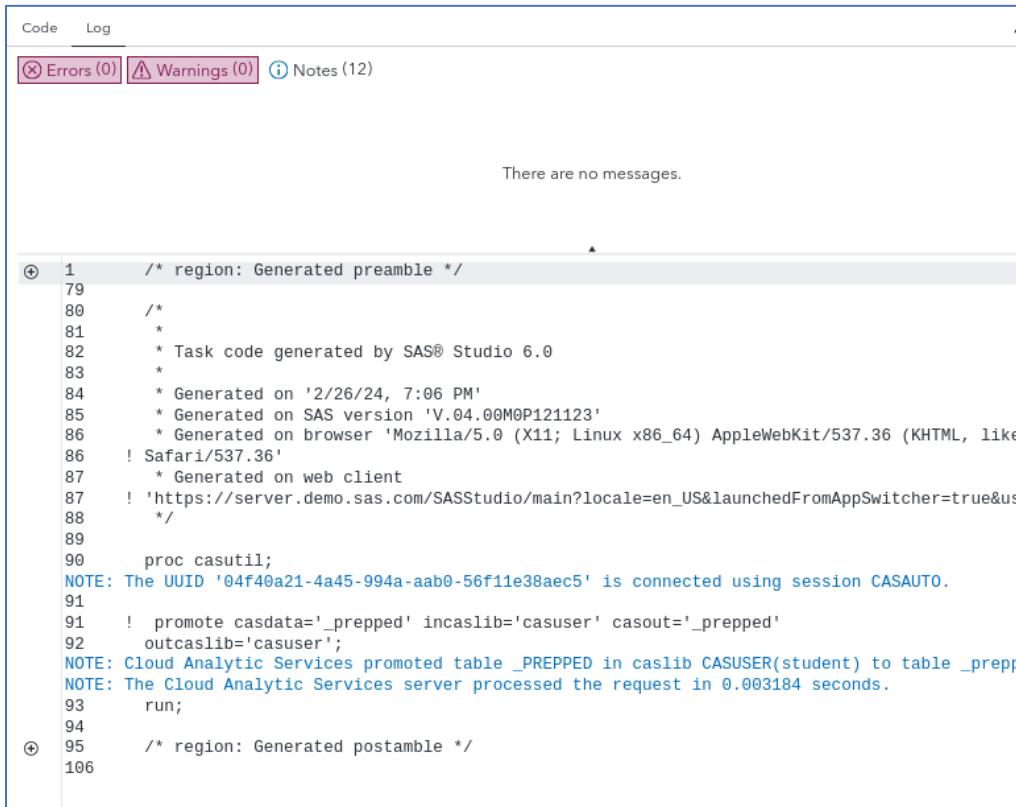
▼ To
caslib:
Table/File/View: *

26. With those settings provided, we've supplied the minimum information set, and SAS code is generated.

Code	Log
1 /*	
2 *	
3 * Task code generated by SAS® Studio 6.0	
4 *	
5 * Generated on '2/26/24, 7:06 PM'	
6 * Generated on SAS version 'V.04.00M0P121123'	
7 * Generated on browser 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,	
8 * Generated on web client 'https://server.demo.sas.com/SASStudio/main?locale=en_US&	
9 */	
10	
11 ⊕ proc casutil;	
12 promote casdata='prepended' incaslib='casuser' casout='prepended'	
13 outcaslib='casuser';	
14 run;	

27. Nice! Go ahead and submit that code via . And, voila, **_prepended** should now be accessible in SAS Visual Analytics as a global table in CAS.

28. But... before moving on too quickly, the last step in this section is to ensure that the code ran without error. You can confirm this in the log.



The screenshot shows a software interface with a blue border. At the top, there are two tabs: "Code" and "Log". The "Log" tab is selected. Below the tabs, there are three buttons: "Errors (0)" with a red background and white text, "Warnings (0)" with a yellow background and black text, and "Notes (12)" with a light blue background and black text. The main area of the window contains the following text:

```
+ 1 /* region: Generated preamble */
79 /*
80 *
81 * Task code generated by SAS® Studio 6.0
82 *
83 * Generated on '2/26/24, 7:06 PM'
84 * Generated on SAS version 'V.04.00M0P121123'
85 * Generated on browser 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
86 ! Safari/537.36'
87 * Generated on web client
88 ! 'https://server.demo.sas.com/SASStudio/main?locale=en_US&launchedFromAppSwitcher=true&us
89 */
90
91 proc casutil;
92 NOTE: The UUID '04f40a21-4a45-994a-aab0-56f11e38aec5' is connected using session CASAUTO.
93
94 ! promote casdata='_prepped' incaslib='casuser' casout='_prepped'
95 outcaslib='casuser';
96 NOTE: Cloud Analytic Services promoted table _PREPPED in caslib CASUSER(student) to table _prepp
97 NOTE: The Cloud Analytic Services server processed the request in 0.003184 seconds.
98 run;
99
100
101 /* region: Generated postamble */
102
103
104
105
106
```

The text is mostly in black, with some lines in blue indicating notes from the SAS server. The message "There are no messages." is displayed at the top of the log area.

29. All good on my end. So, that's it for this section. Congratulations on finishing the first part of our mortgage adventure!

Task 2: SAS Visual Analytics – A Non-Coders Paradise

Learning Objectives

This task provides the opportunity to learn and practice skills such as

- navigating Visual Analytics in the SAS Viya
- learning to use the Auto Chart feature
- creating visualizations using different types of variables (categorical, numeric, geographic)
- using predictive modeling functions within SAS Visual Analytics.

Estimated Time of Completion

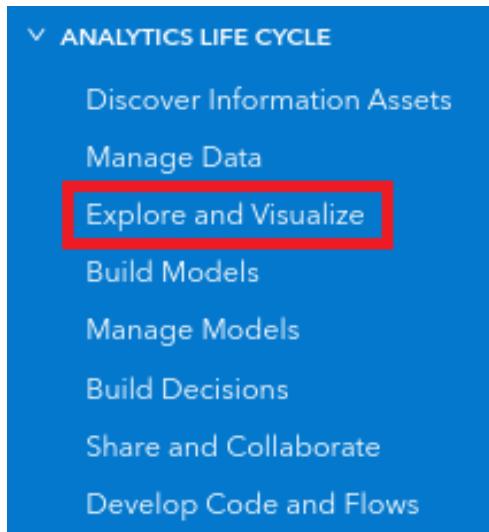
This task is estimated to take about 30 minutes.

Task: Introduction to SAS Visual Analytics

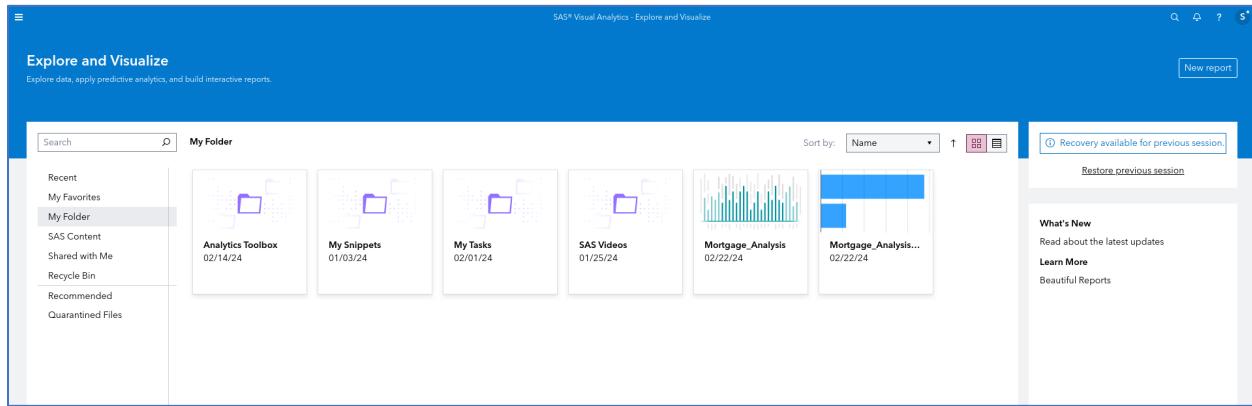
Coding is great. But as a new employee of iLink Mortgage, Inc. – and new user of SAS, you'd likely prefer to start your analytical journey with some no-code options. Fortunately, SAS Visual Analytics has several helpful tools to guide you along that path!

For this task, you'll create your first predictive models to examine whether the loan will default (that is, **BAD = 1**). Buckle up for the adventure!

1. Let's move over into SAS Visual Analytics. Return to the **Applications** menu (\equiv) in the upper left corner and select **Explore and Visualize** under the **Analytics Life Cycle**.



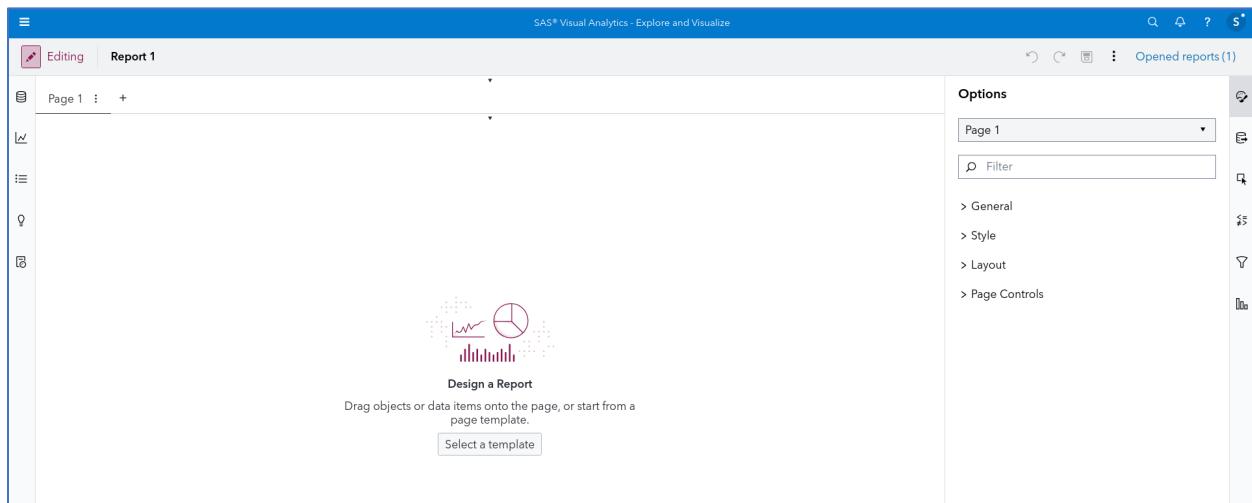
2. Welcome to SAS Visual Analytics! Your view will differ marginally, but it should appear like the following:



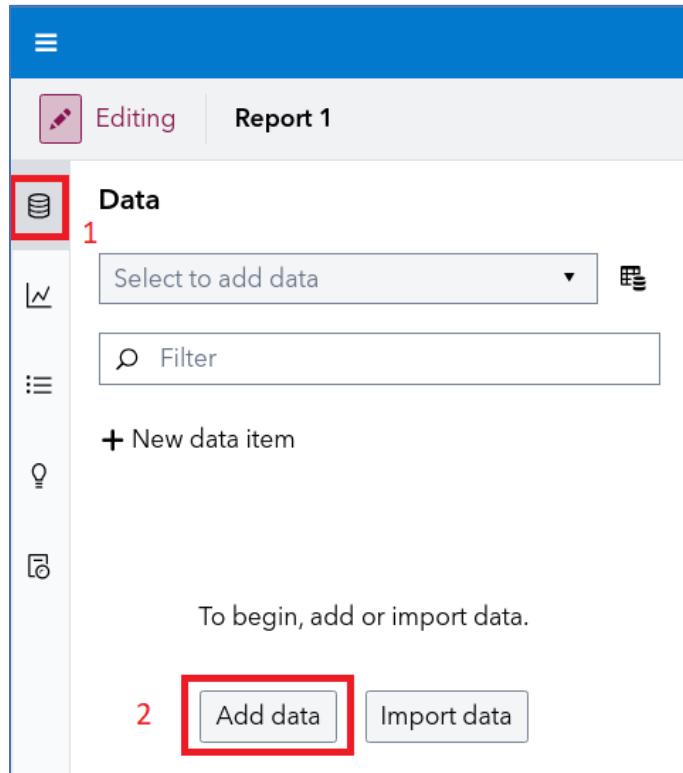
3. To continue our SAS Visual Analytics adventure, click **New report**.



4. The main SAS Visual Analytics screen will appear.



5. Because data is always a good thing, let's add some data to our report. Find – and click – the **Data** icon in the left pane. Then click **Add data**.



6. Now this is the part where you try to remember, and find, the data set used for modeling in SAS Studio. No worries if you forgot that name – it's **_PREPPED**. And it should be one of the global, in-memory tables, because we pushed it there.

The screenshot shows the "Choose Data" dialog box. At the top, there is a search bar labeled "Search all data" and an "Import data" button. Below the search bar, a dropdown menu shows "Available (2)". The main table lists two datasets: "**_PREPPED**" and "SASVIYATYPES". The "**_PREPPED**" row is selected, indicated by a blue highlight. To the right of the table, a detailed view panel for "**_PREPPED**" is open, showing its properties. The properties listed are: Asset type: In-memory data; Date modified: Feb 26, 2024 7:08 PM; Modified by: student; Date created: Feb 26, 2024 6:49 PM; Created by: student; and Source table: --. There are "Details" and "Columns (19)" tabs in the panel. At the bottom of the dialog, there are "Add" and "Cancel" buttons.

7. Select **_PREPPED** and then click **Add**. The data should load in your SAS Visual Analytics report.

The screenshot shows the SAS Visual Analytics Data pane. At the top, there is a navigation bar with a pencil icon labeled "Editing" and the title "Report 1". Below this is a search bar with the placeholder "Data" and a dropdown menu showing "_PREPPED". A "Filter" search bar is also present. The main pane lists data items under two categories: "Category" and "Measure". Under "Category", there are three items: "JOB - 7", "Partition Indicator - 2", and "REASON - 3". Under "Measure", there are ten items: BAD, CLAGE, CLNO, DEBTINC, DELINQ, DEROG, Frequency, IM_CLAGE, IM_CLNO, and IM_DEBTINC. Each item has a small icon and a blue circular progress bar indicating its status.

8. Some of the metadata definitions transferred from the SAS Studio environment, but some of them did not. And, we'll actually want to make a few additional changes. By cleaning up the metadata in SAS Visual Analytics, we can explicitly state which variables are categorical variables, which are inputs on an interval, and which variables should be ignored all together. Let's get started...

9. Examine the current variables in your analysis. And know that – at the end of this subsection – we'll want to match the following:

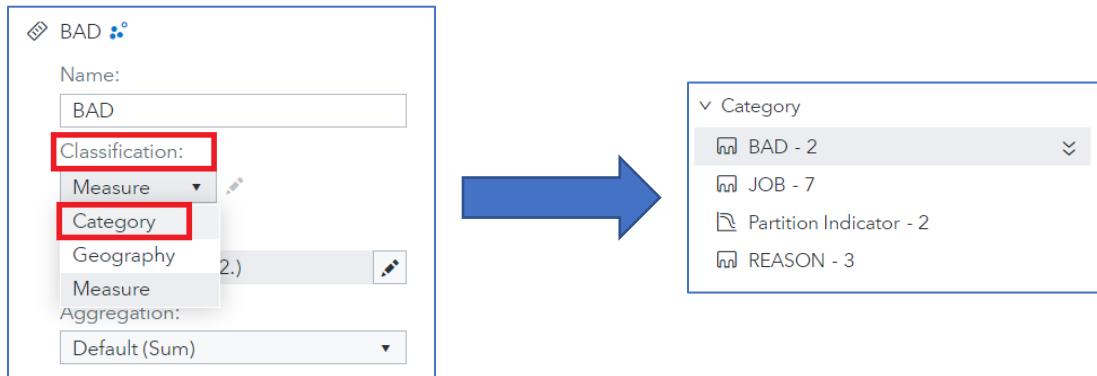
The screenshot shows the SAS Data window with the title 'Data'. At the top, there is a dropdown menu set to '_PREPPED' and a 'Filter' search bar. Below these are two main sections: 'Category' and 'Measure'. The 'Category' section contains variables: BAD - 2, DELINQ - 15, DEROG - 12, IM_CLNO - 62, IM_YOJ - 99, JOB - 7, and NINQ - 17. The 'Measure' section contains variables: Frequency, IM_CLAGE, IM_DEBTINC, IM_MORTDUE, LOAN, and VALUE. Each variable entry includes a small icon and a count in parentheses.

10. And how do we do that, you might ask? Well, each variable has an **Edit properties** button that enables you to change the metadata for that variable.

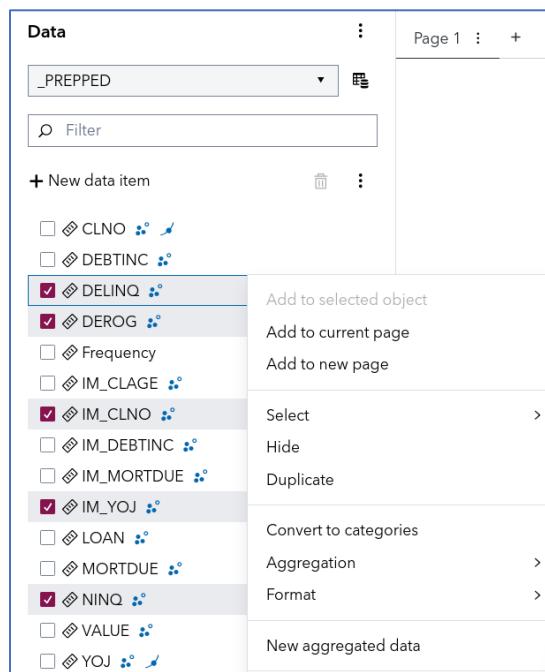
This screenshot shows a subset of the SAS Data window. It displays the 'Category' section with three variables: JOB - 7, Partition Indicator - 2, and REASON - 3. To the right of the 'JOB' variable, there is a small red-bordered square highlighting a downward-pointing arrow icon. Next to it is a rectangular button labeled 'Edit properties'.

11. For our analysis, we want to ensure that **BAD, DELINQ, DEROG, IM_CLNO, IM_YOJ, JOB, NINQ, and REASON** are all classified as **Category**. Why? Because we'd like to treat them as Yes/No variables, instead of the usual 1/0 binary variable that is coded.

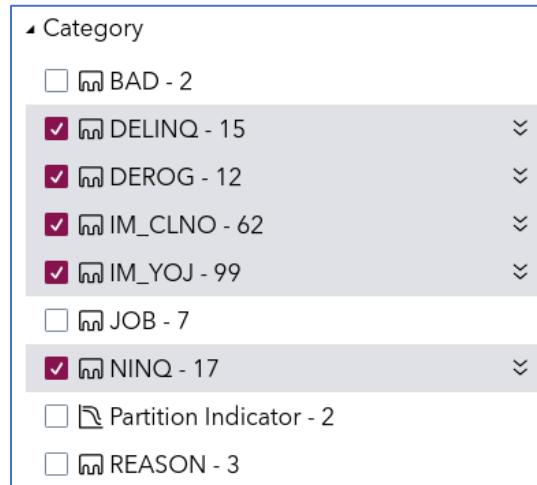
12. **BAD**, our outcome variable in the modeling, is possibly still classified as a **Measure**. If this is true, we can change this by clicking the **Edit properties** button and then changing the classification from **Measure** to **Category**. The variable will now appear under **Category** and will show the number of unique levels in the data.



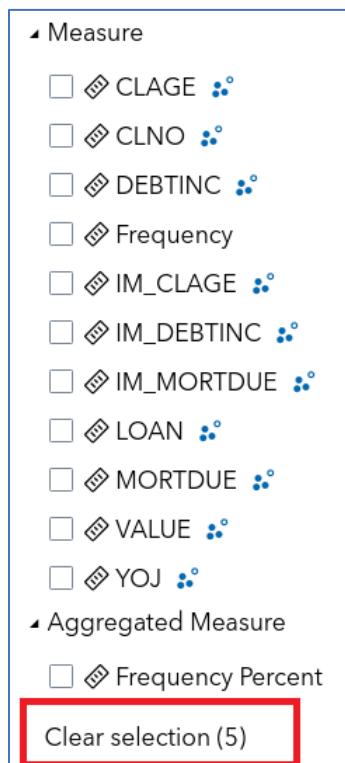
13. That's the longer way. But, we can also change several variables at once – while using a shortcut. Ensure that **BAD** is no longer selected. Then click **DELINQ**, **DEROG**, **IM_CLNO**, **IM_YOJ**, and **NINQ**. Then right-click one of the variables to get the following menu:



14. See the **Convert to categories** option? Select that option to change all five of those variables at once. Ensure that these variables are now found under the **Category** section.

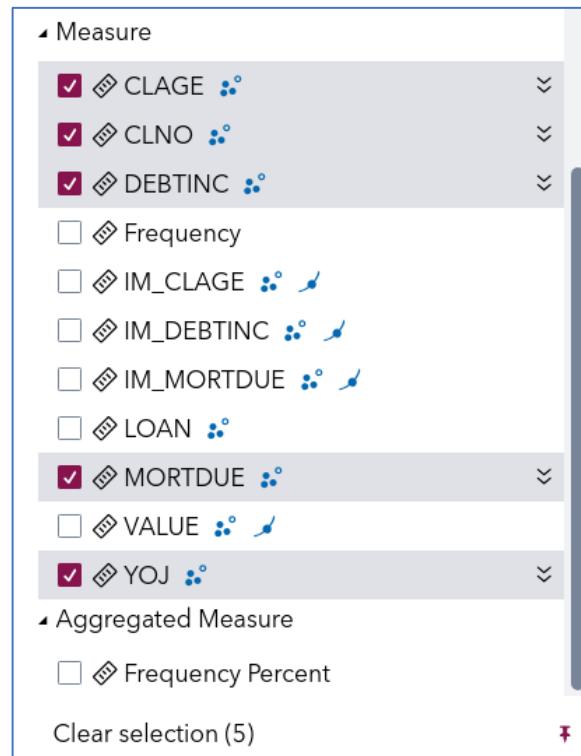


15. To ensure that further changes aren't applied to these variables, you can either deselect each of the five variables individually or – better yet – scroll to the end of the variables and select **Clear selection (5)**:

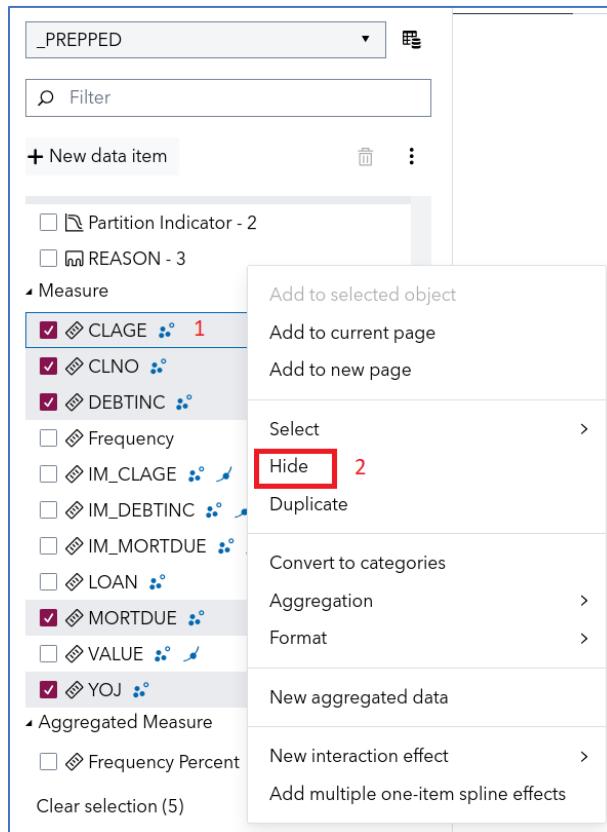


16. The (5) in the example above simply indicates how many variables are currently selected. That stated, seriously, clear the selections now.

17. Finally, we'd like to exclude some of the measures – as these variables will not be used in our modeling. Select **CLAGE**, **CLNO**, **DEBTINC**, **MORTDUE**, and **YOJ** as shown.

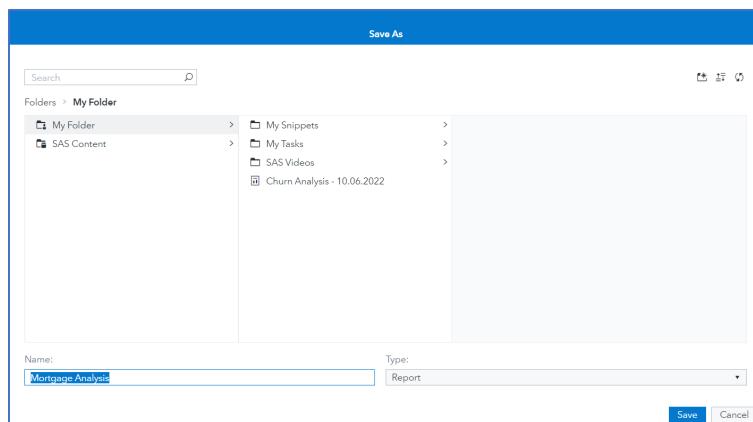


18. Next, right-click one of the selected variables, and then select **Hide**.



19. Presto! Variables be gone – and your metadata should now match the screenshot from above.

20. This is a great time to save your program. Find the classic **floppy disk** icon in the upper right corner. Name the analysis something creative like **Mortgage Analysis** and save it in the default location.

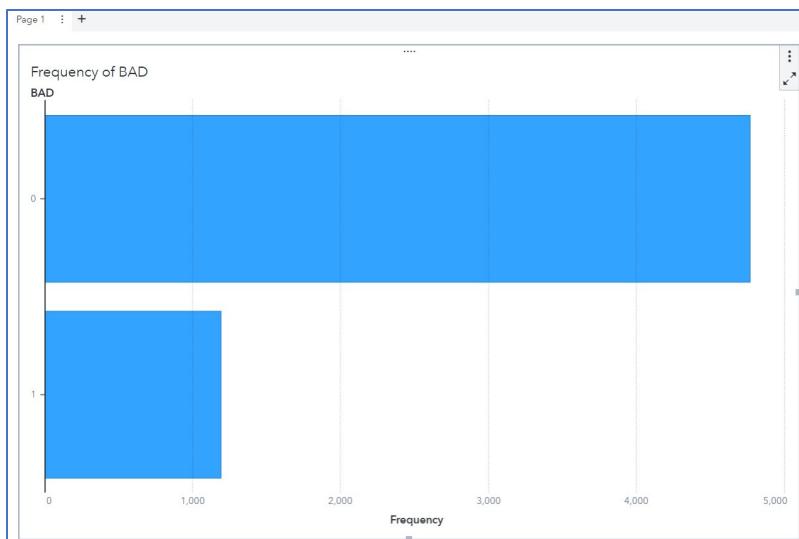


Please continually save your work as the analysis progresses. Lost work is kind of a bummer 😞 !

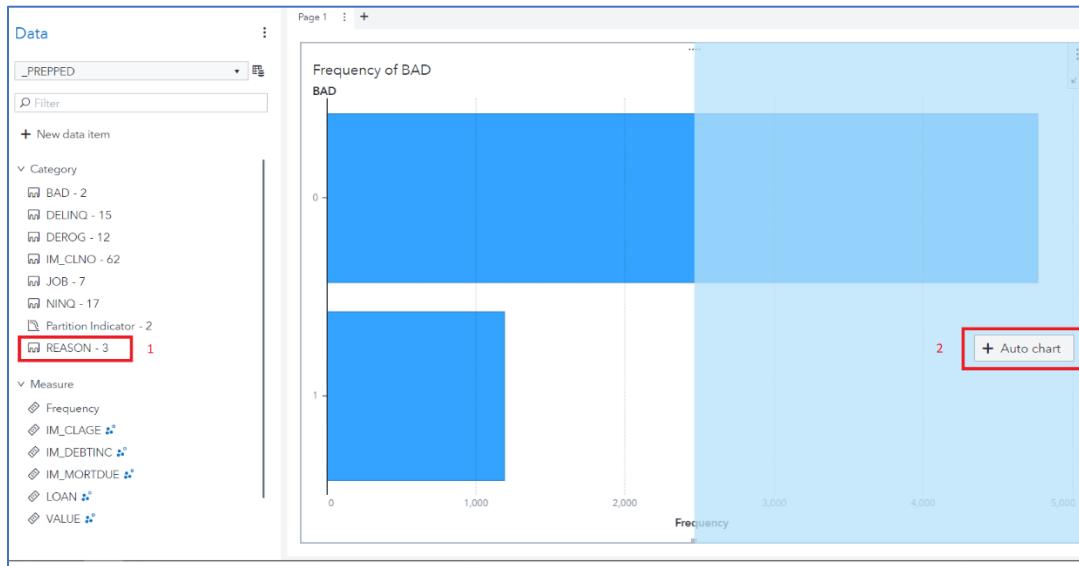
21. Now that the Visual Analytics report is established and the data are where they need to be, let's better understand select variables by using **Auto charts**. With the **Data** icon selected, click **BAD** and then drag and drop the variable to the canvas.

The screenshot shows the SAS Visual Analytics Data pane. On the left, there is a list of variables under a 'Category' section. The 'BAD - 2' item is highlighted with a red box and labeled '1'. To the right, there is a large blue area representing the canvas, with a '+ Auto chart' button highlighted with a red box and labeled '2'.

22. With **Auto chart**, Visual Analytics will automatically intuit what you'd like plotted. Here it will produce a bar chart with frequencies.



23. We'd also like to see distributions for five other variables: **REASON**, **JOB**, **LOAN**, **IM_DEBTINC**, and **IM_MORTDUE**. To add another variable to the canvas, move it from the Data tab to a unique space on the canvas. You'll see the **+ Auto chart** option when the space is available for a new chart.

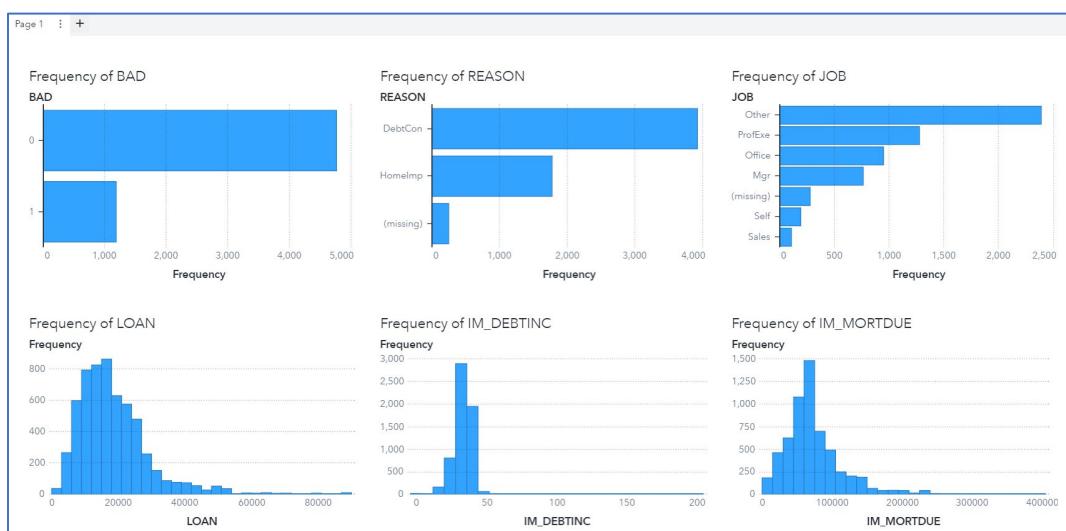


24. Variables can be placed above/below or left/right of existing objects. And if you make a mistake, then no worries – there is an undo button in the upper right corner.

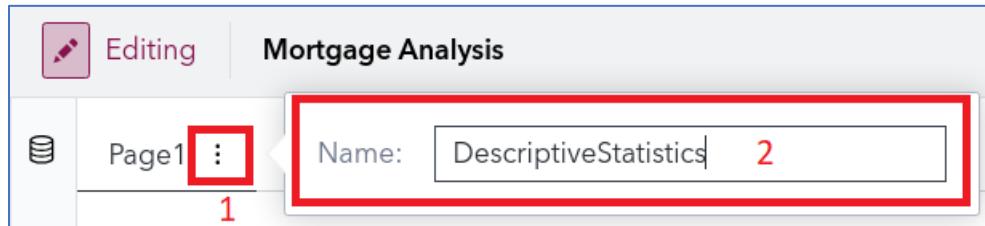


Note: Use it. And use it often when needed.

25. Now replicate the following:



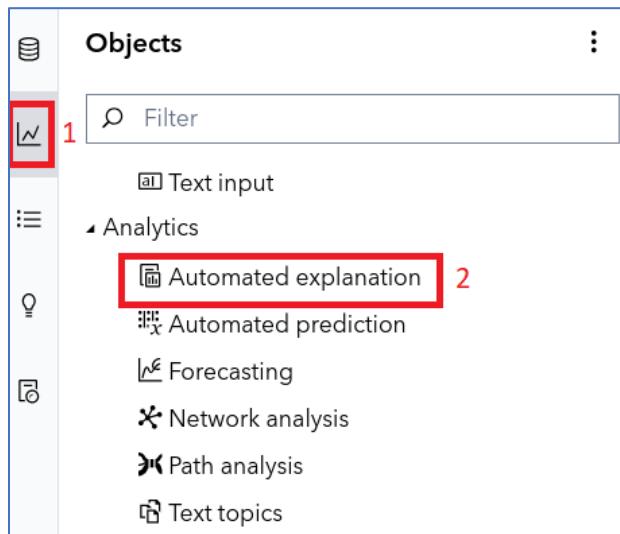
26. Change the name of Page 1 to **DescriptiveStatistics**, by clicking the **Page** menu and then selecting **Rename page** (or you can simply double-click the **Page** tab).



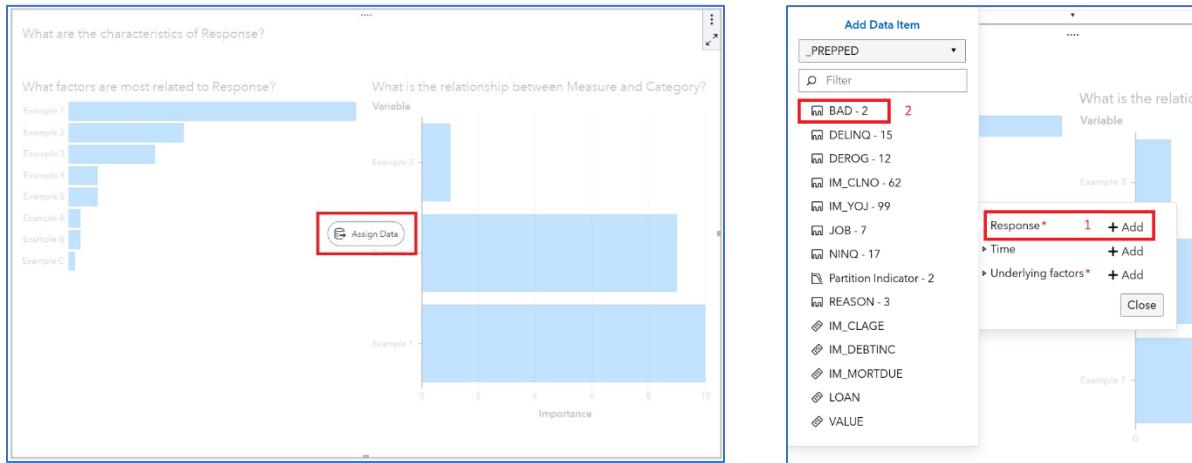
27. Click **Enter** when you're finished typing the new name. Then create a new page by clicking the **+** button next to **DescriptiveStatistics**. In this new canvas, rename Page 2 to **AutomatedExplanation**.



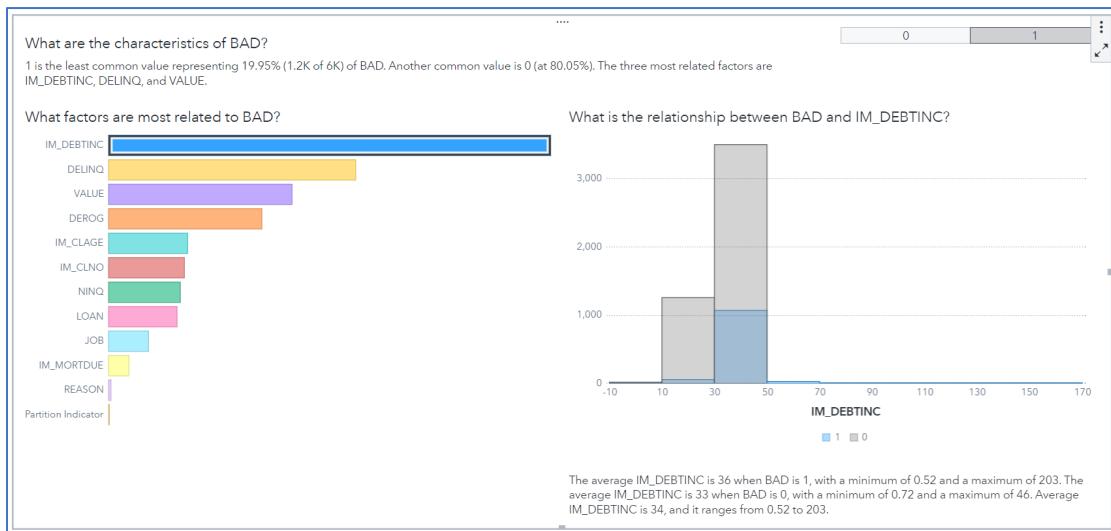
28. Let's add an Automated explanation object to this new canvas. It can be found under **Objects** in the left pane and then clicking **Analytics**.



29. Drag-and-drop the Automated explanation object onto your new page. Next, click **Assign Data** and then choose **BAD** as the **Response**.



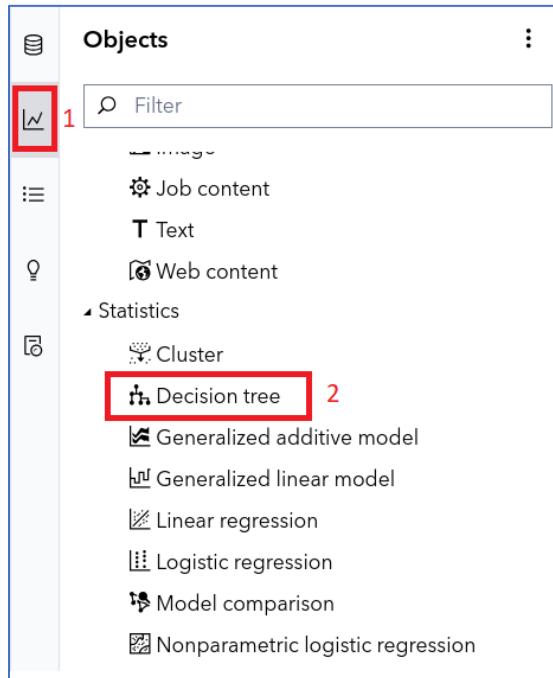
30. Examine the output from the **Automated Explanation** object. Start by clicking **BAD=1** in the upper right and then explore how individual factors are related to **BAD**, as follows:



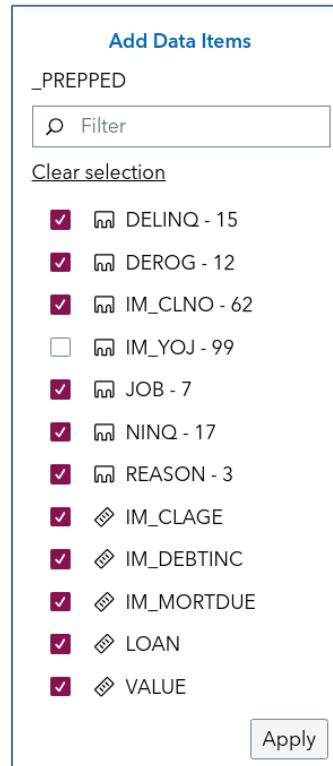
31. Automated analysis is a fast way to understand the bivariate relationship between variables, but we have more complex modeling tools. In other words, we have much cooler stuff. Let's start with a **decision tree** and then build more sophisticated models from there.

32. Create a new page and change the name to **DecisionTree**.

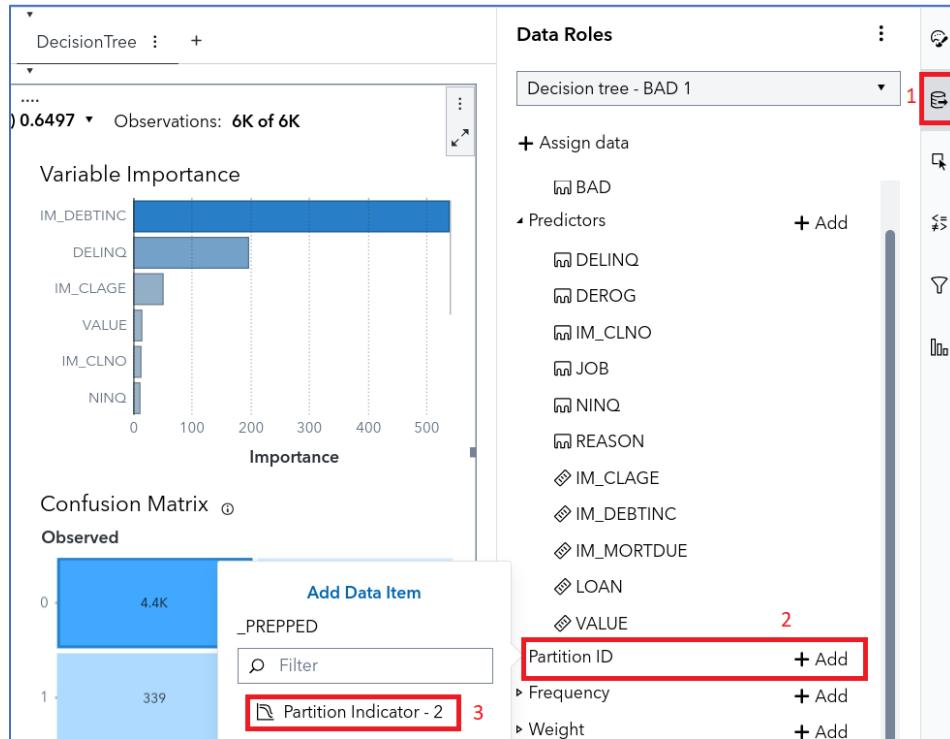
33. From the **Objects** pane on the left, find the **Decision tree** object under **Statistics** and drag-and-drop it on the DecisionTree canvas.



34. It's again time to assign data. Click the **Assign Data** button, select **BAD** as the **Response**, and select the following variables as the **Predictors** (hint: we're just excluding **IM_YOJ**). Then click **Apply**.

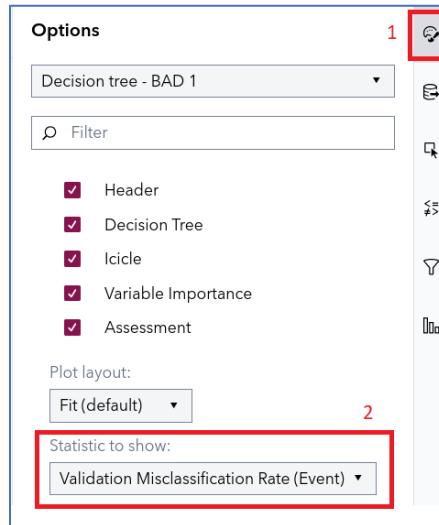


35. Now – because we’re aspiring data scientists, after all – let’s add a data partition to allow for honest assessment. Find the **Roles** button on the right pane. Expand the options and then scroll down to **Partition ID**. Click **+ Add** and then select **Partition Indicator**.

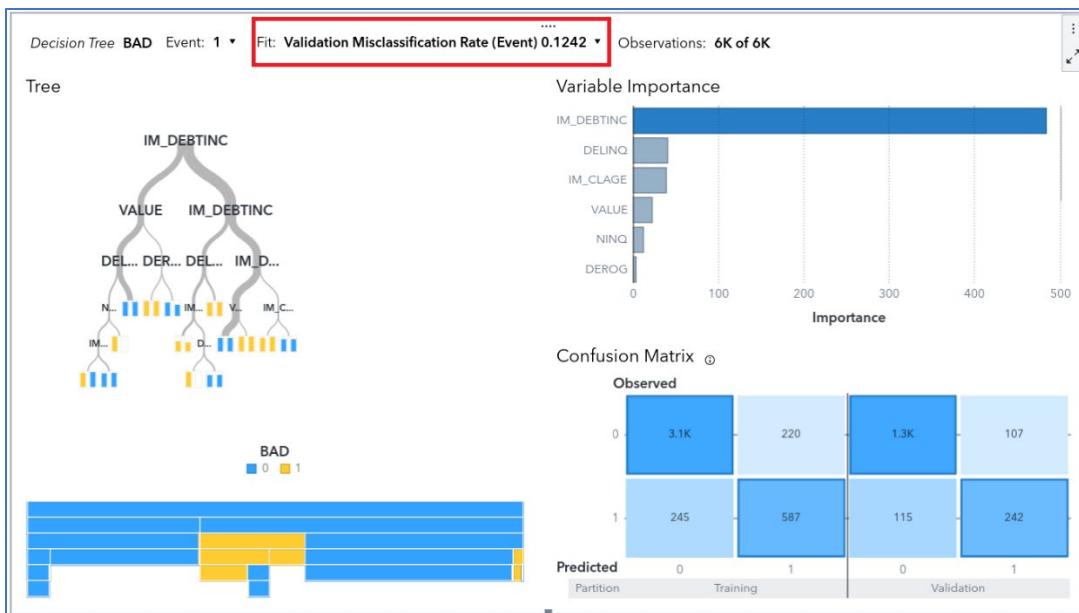


36. A partition is now added. Woot! One last item before we examine our output – let’s explore how to use other model goodness-of-fit statistics. *Misclassification rate* is the ratio of misclassified observations to total observations and is a common selection statistic used by many data scientists. Let’s explore that option here – with the general underlying point that we have many statistics to choose from when determining the “optimal” model.

37. From the right pane, select **Options** and then scroll down until you see the section for **Model Display**. Under **General**, change the **Statistic to show** to **Validation Misclassification Rate (Event)**.

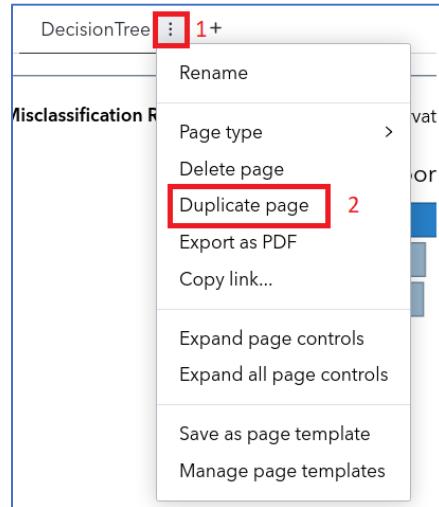


38. Observe and record the Validation Misclassification Rate (Event). Did you get a similar finding?

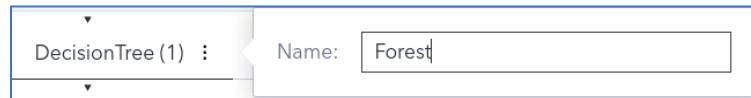


Don't sweat it if you didn't get the exact results reported above. It's likely because we're using different samples.

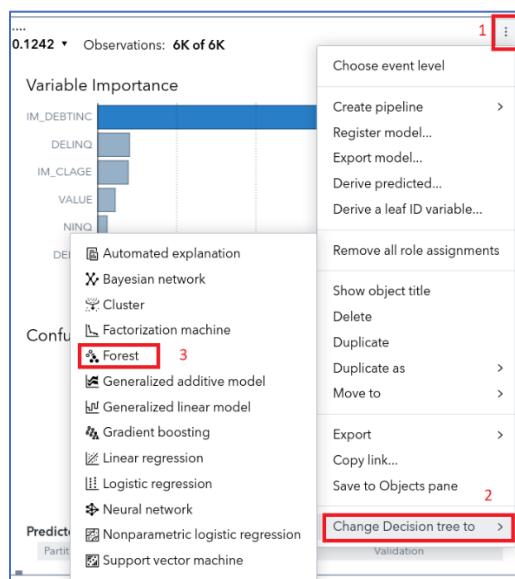
39. Now that was just a single model – based on a single decision tree. What is often better than a single tree? A forest! Let's use the information from the decision tree to quickly run a couple of other models. To start, find the options button next to the **DecisionTree** title and then navigate to **Duplicate page**.



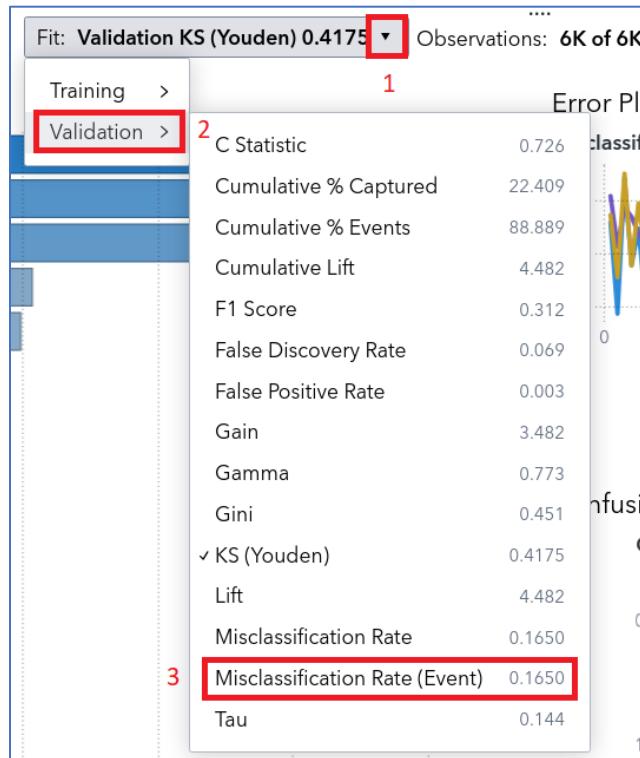
40. Change the name of the new page from DecisionTree(1) to **Forest**.



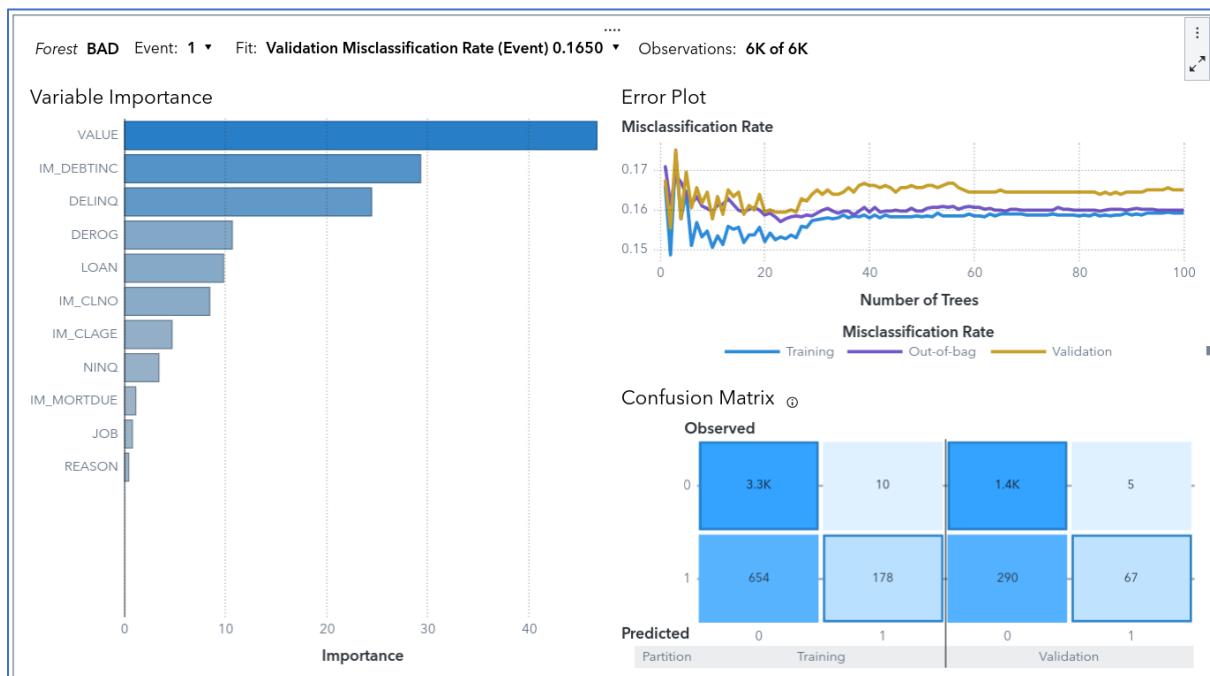
41. Click Enter. Next, find the **Object Menu** for your new object in the upper right corner. Navigate to **Change Decision tree to >** and then select **Forest**.



42. This great hack takes all the metadata definitions from the decision tree and puts them into the Forest model. And the model is run instantaneously! As in the previous example, we'll still need to change the **Fit** statistic to **Validation Misclassification Rate (Event)** – which we can also do from the drop-down menu here:



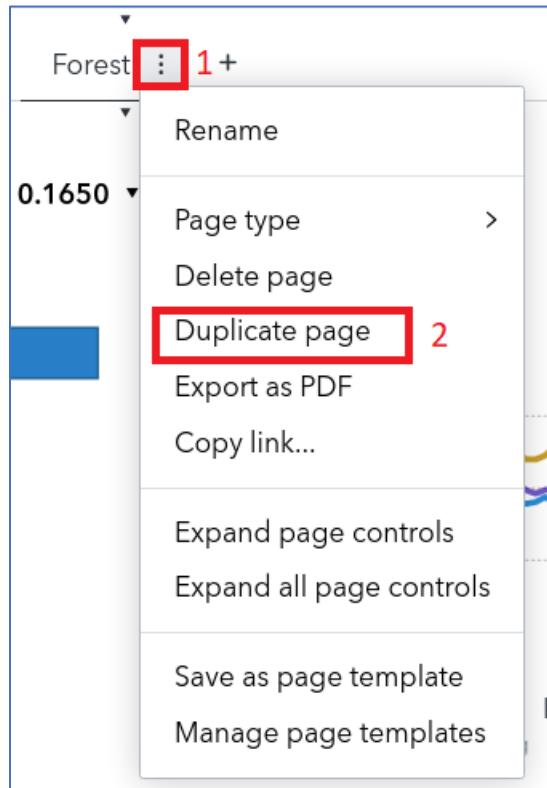
43. Let's then examine the output.



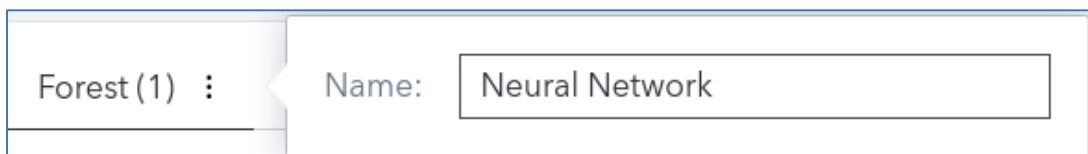
44. Based on **Misclassification Rate (Event)**, is this model better or worse than the decision tree?

...Yup, it's worse. Weird. But that just means that we probably need to tune the hyperparameters – which is a bedtime story for another time.

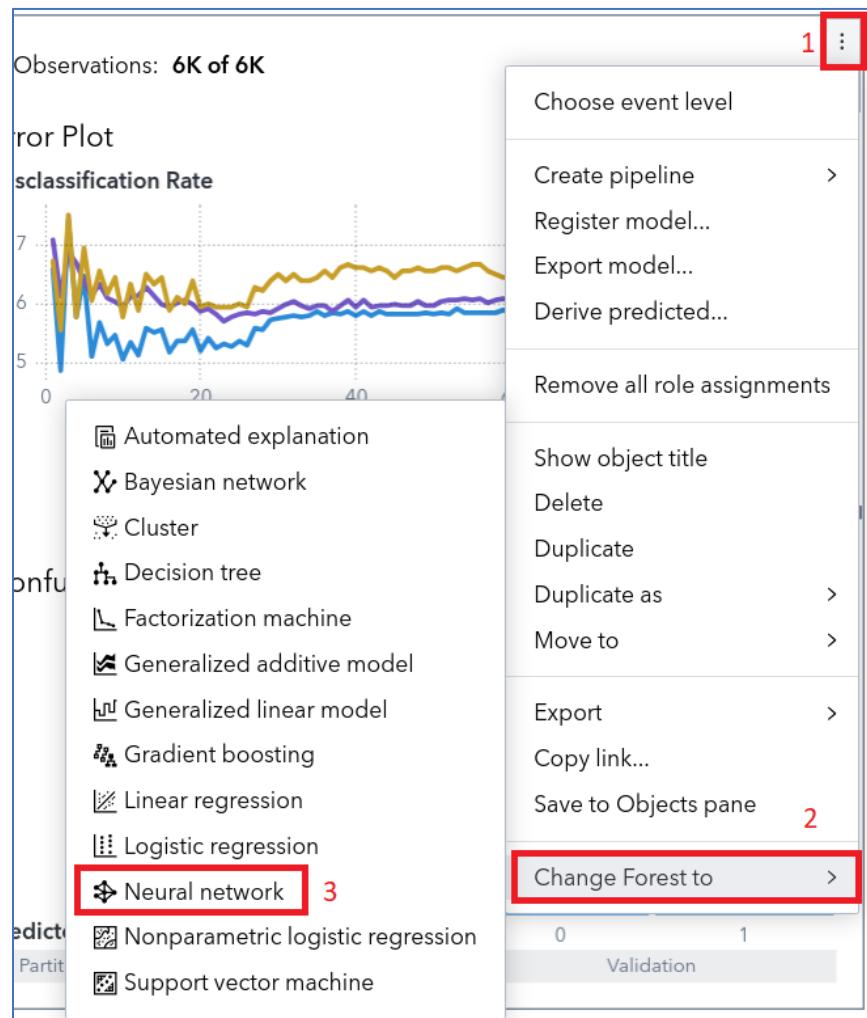
45. To complete our mini-analysis in this section, let's run one more model: a neural network! From **Options** on the Forest tab, again select **Duplicate page**.



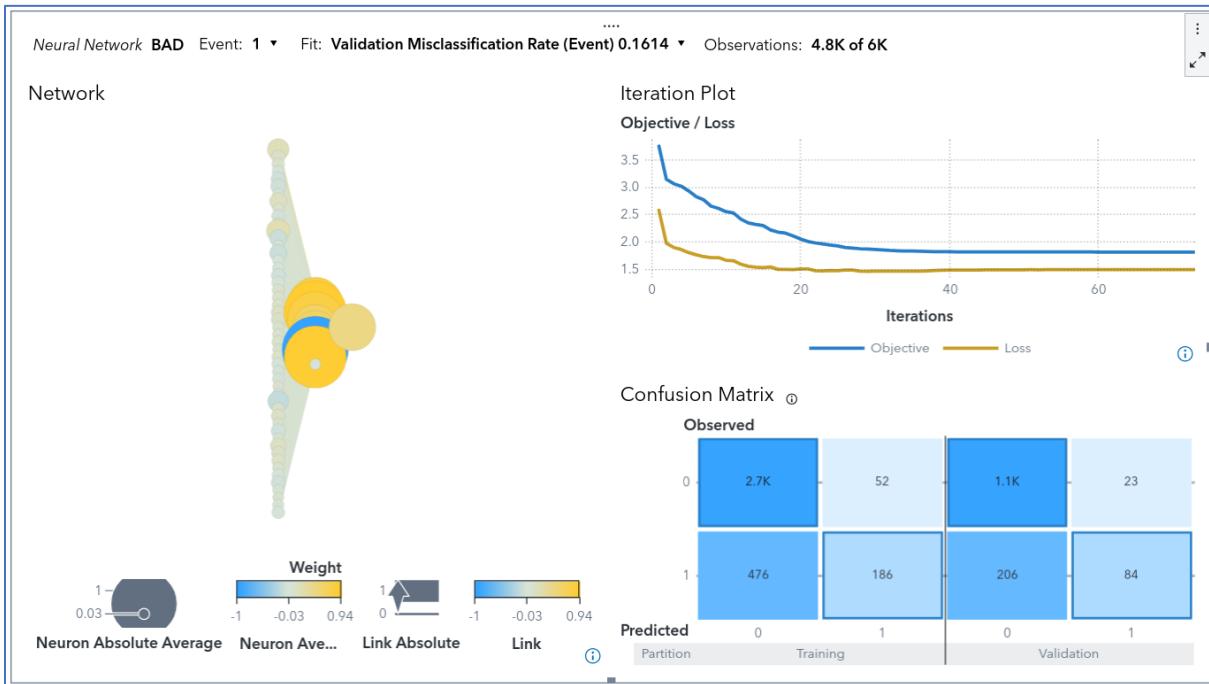
46. Rename the name of the tab from Forest(1) to **Neural Network**.



47. Like last time, find the **Object Menu** and select **Change Forest to > Neural network**.



48. To make the models comparable, ensure that the Fit statistic is **Validation Misclassification Rate (Event)**. Now let's examine the results.

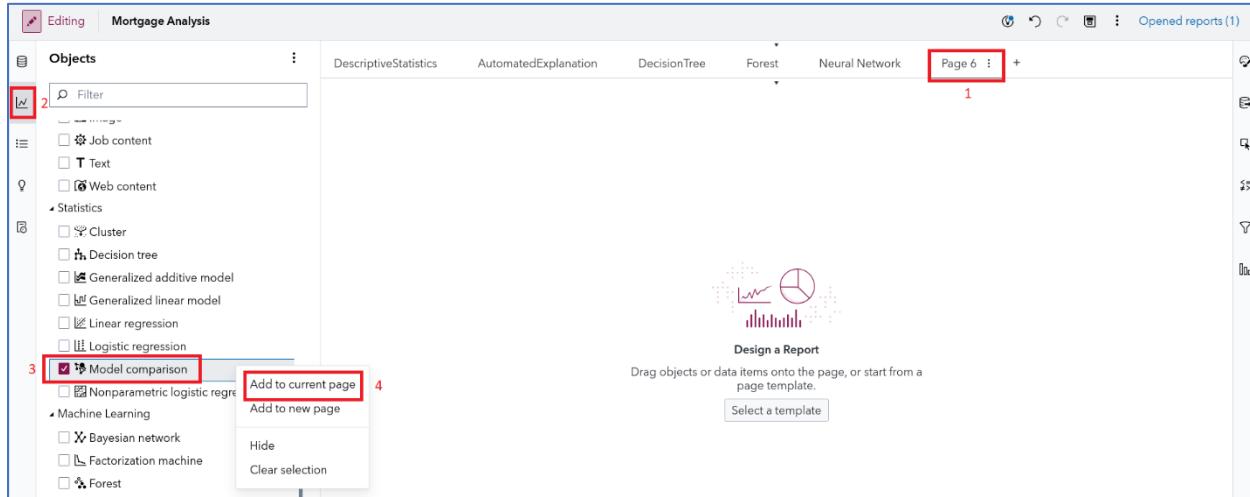


Note: It looks like we lost some observations – likely because neural networks use complete case analysis and we have missing data. But, this is a problem for another day.

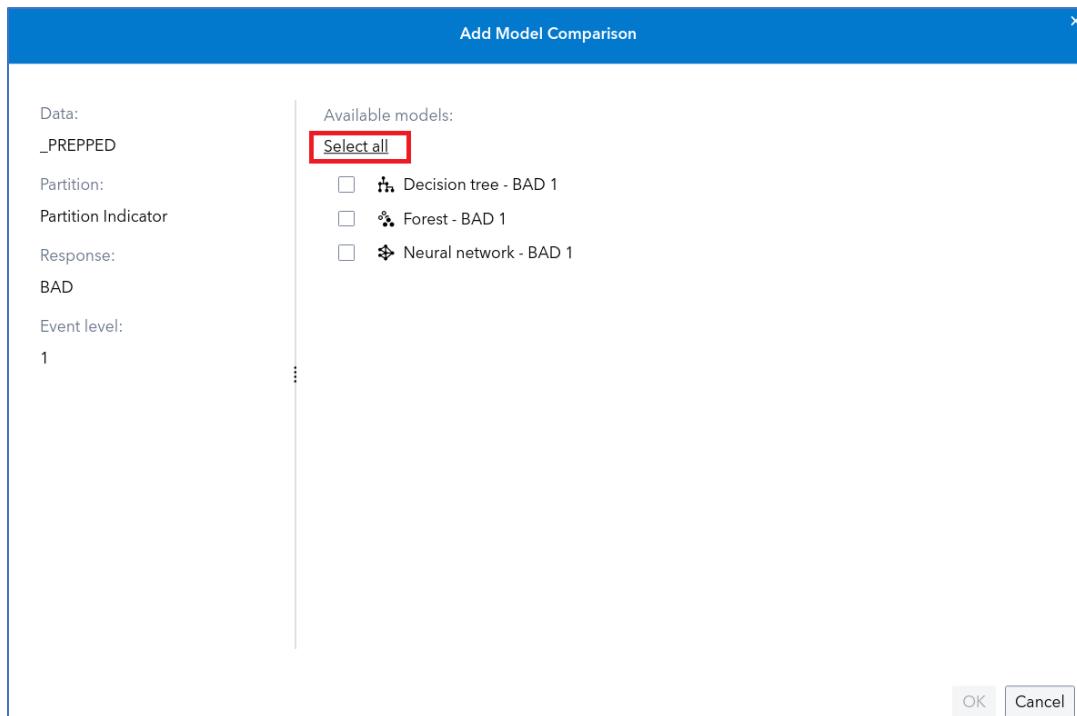
49. In general, how did the neural network do? Well, again, not great. The misclassification rate is 0.1614, which is – again – higher than the much simpler decision tree model.

50. The decision tree model appears to have the lowest Misclassification Rate for the current set of models. So, we can crown it as our initial “champion” model. But, before moving on, let's create one more page and use the **Model Comparison** object to nicely summarize the findings.

51. To do so, create a new page, find the **Model Comparison** object under **Statistics**, and select **Add to the current page**.



52. The Add Model Comparison window appears. Use the default settings and click **Select all** to select all available models.

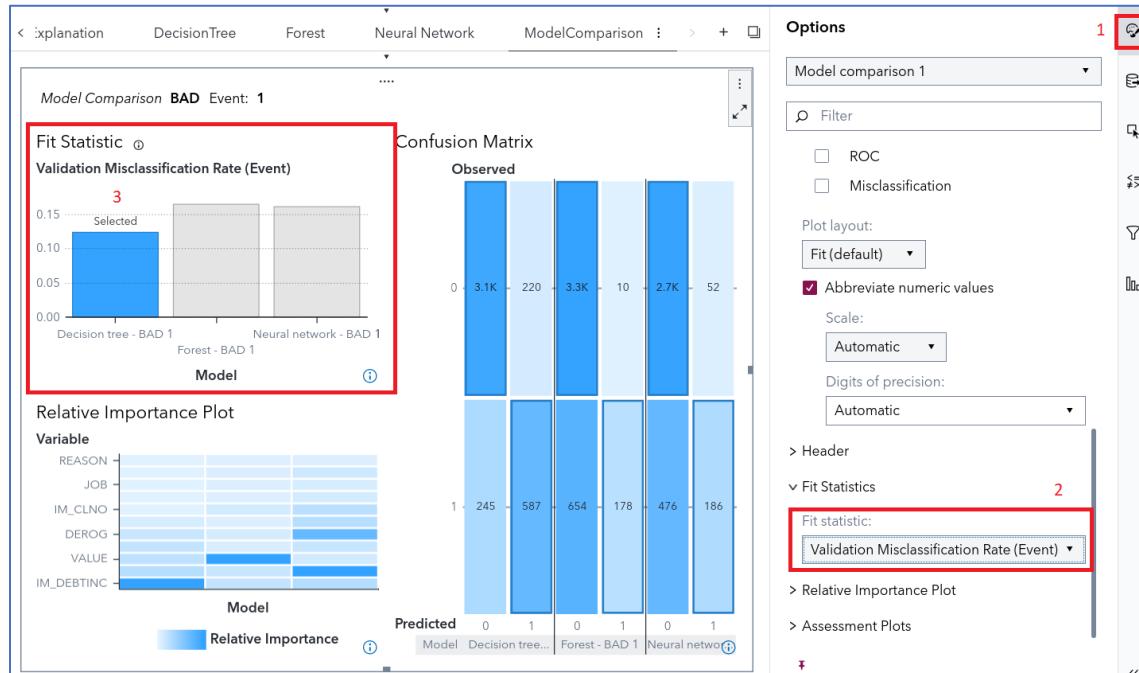


53. Click **OK**.



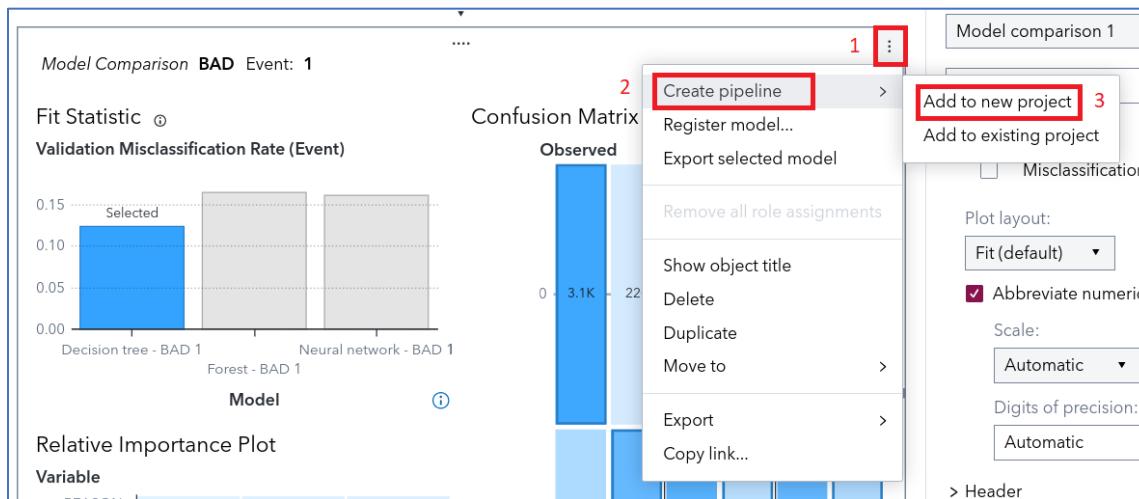
54. While data are being analyzed, rename the page to **ModelComparison**.

55. Finally, with the **Model Comparison** object selected, click **Options** and change the **Fit Statistic** to **Validation Misclassification Rate (Event)**. Do you also get the decision tree as the champion?



56. Hopefully there are no surprises there – as the Model Comparison object simply replicates and combines the findings from our three predictive models.

57. Our last Visual Analytics task is to export the champion model to SAS Model Studio – where we have many more no-code options to explore. From the Model Comparison object, return to the **Objects** menu. This time, find **Create pipeline** and then select **Add to new project**.



58. Now sit back and enjoy as your new SAS Model Studio project is constructed. Pipelines are a fundamental tool in SAS Model Studio, which you will learn a lot about in the next part of your adventure!

Task 3: Going Deeper with SAS Model Studio

Learning Objectives

This task provides the opportunity to learn and practice skills such as

- navigating the SAS Model Studio application
- creating and running pipelines for supervised machine learning models
- incorporating open-source tools
- comparing and assessing models.

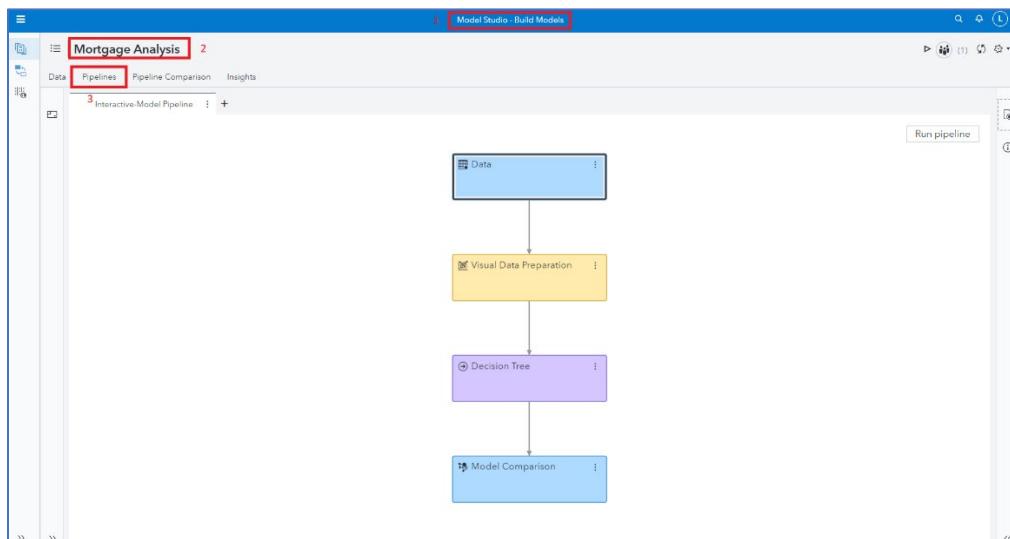
Estimated Time of Completion

This task is estimated to take about 45 minutes.

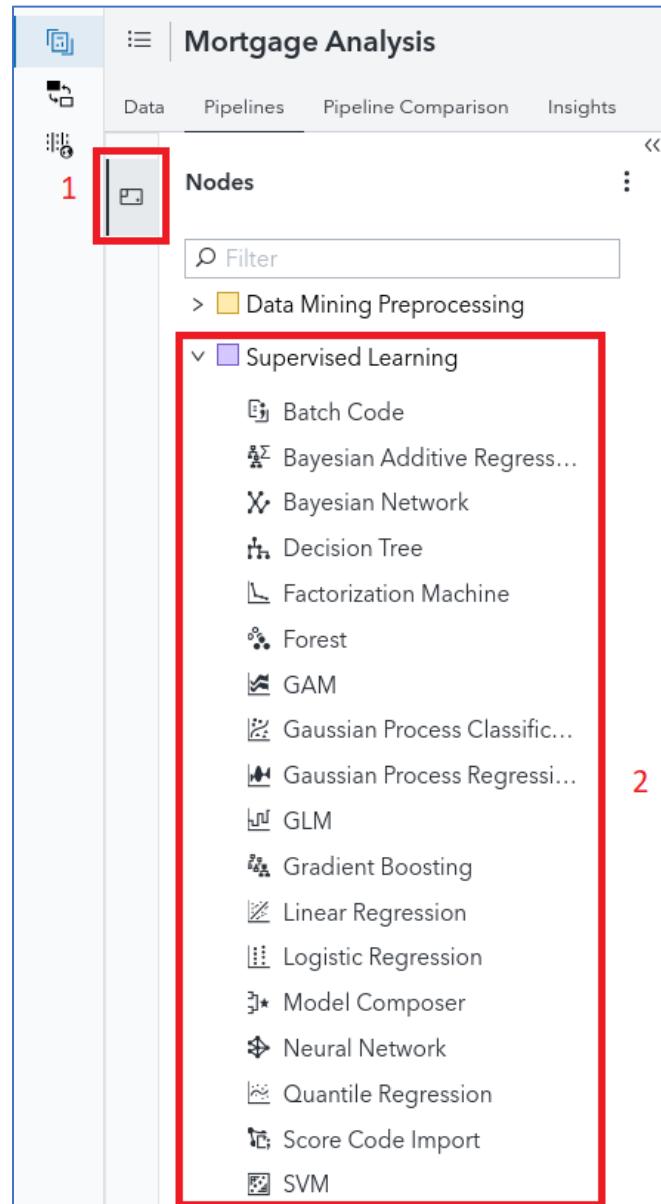
Task: Learn to Love SAS Model Studio

SAS Visual Analytics is a great starting point for those new to SAS and analytics. But SAS Visual Analytics simplifies many of the modeling options to make it a more user-friendly experience. In other words, there is a trade-off between ease-of-use and modeling options available. With SAS Model Studio, you get a much more robust no-code and (limited) code environment – but it takes a bit more technical know-how. So, let's take our analytical endeavors to the next level with this great tool.

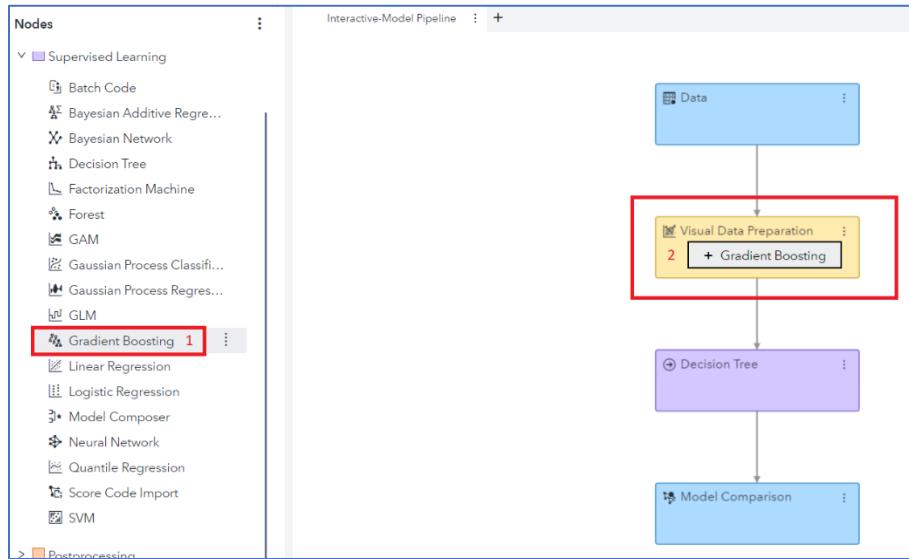
1. If you exported the pipeline properly in the last section, you should have a new SAS Model Studio Project named Mortgage Analysis – and the project should open to the **Interactive Model Pipeline**.



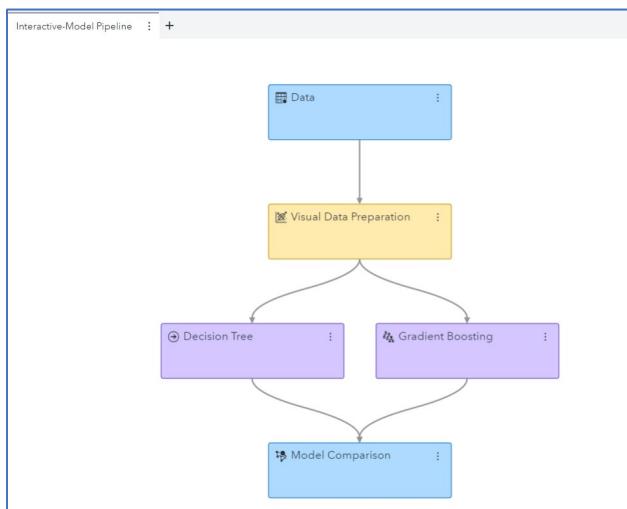
2. SAS Model Studio is pipeline centric and you can see that our champion decision tree from our SAS Visual Analytics investigation has been converted to a pipeline.
3. Before running our first SAS Model Studio pipeline, let's explore how easy it is to add another model to the pipeline. Locate the **Nodes** button in the left corner. Expand the **Supervised Learning** nodes and see what is available.



- Now that's a lot of options! Let's see whether the default **Gradient Boosting** model can defeat our beloved decision tree. Grab the **Gradient Boosting** node and drop it on top of the **Visual Data Preparation** node.



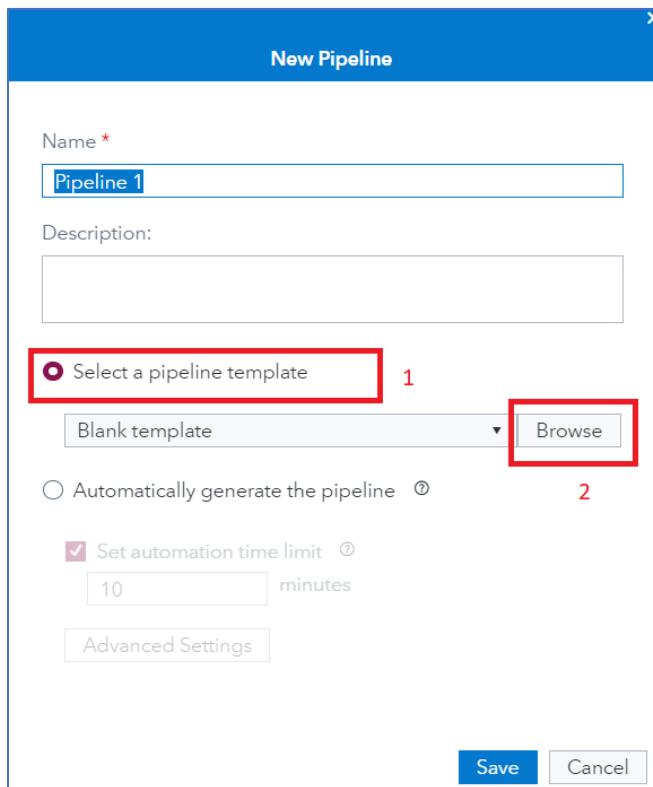
- The new pipeline appears.



- Submit the pipeline by selecting **Run pipeline** in the upper right corner.
- While this pipeline is running, I can show you one of my favorite features of SAS Model Studio. It's in the prebuilt pipelines, which makes running many models very easy. Create a new pipeline by clicking the + next to the Interactive-Model Pipeline tab.



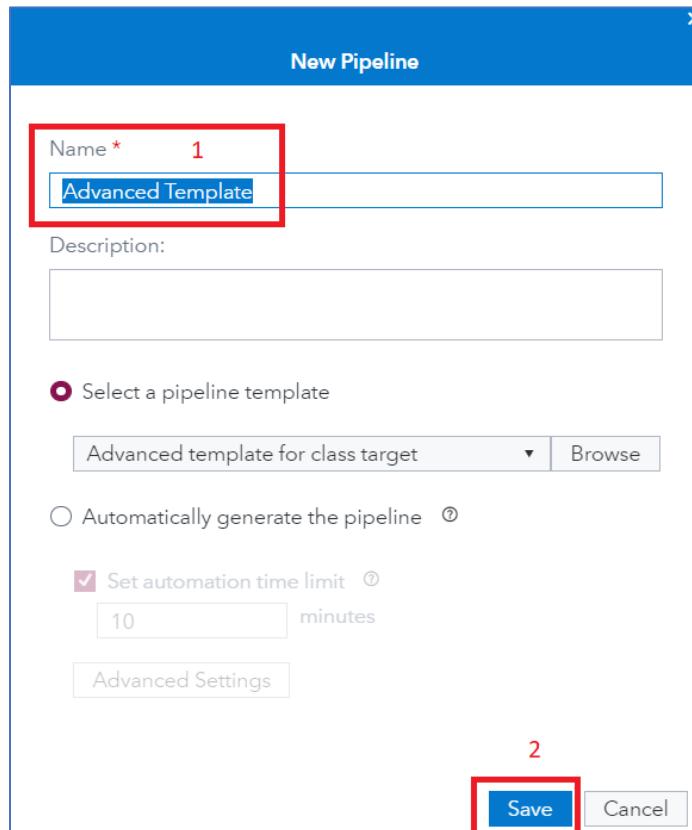
8. The New Pipeline window appears. Under **Select a pipeline template**, choose **Browse**.



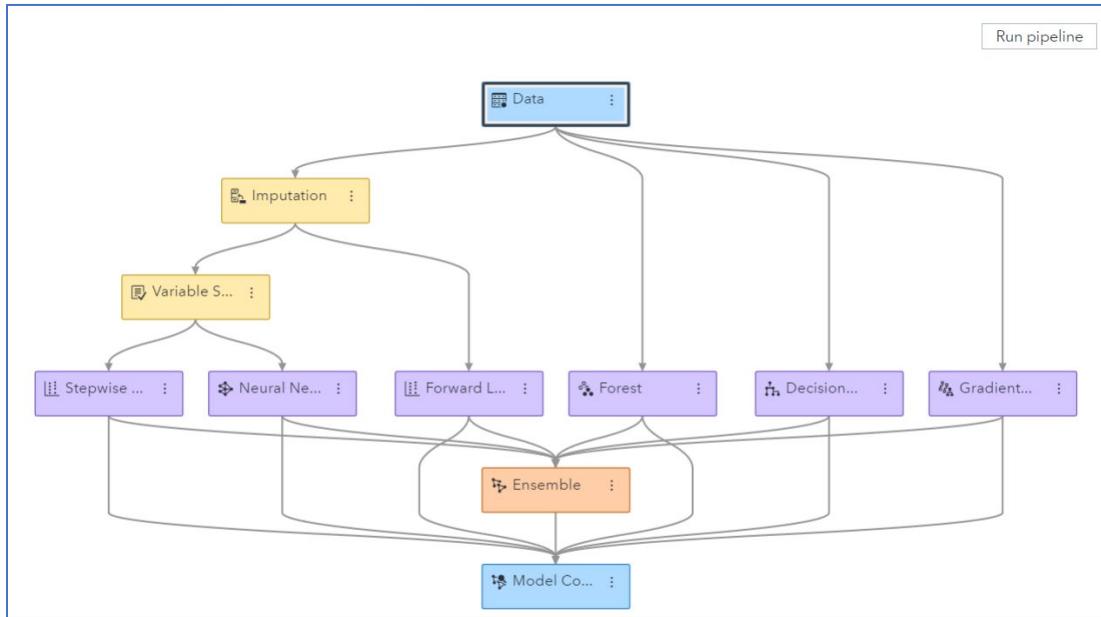
9. Check out all the pipeline templates available!

Browse Templates				
Template Name	Description	Owner	Last Modified	
Advanced template for class target	Data mining pipeline that extends the intermediate template for a class target by adding neural network, forest, and gradient boosting models. An ensemble model is also provided.	SAS Pipeline	Jan 3, 2024, 2:57:59 PM	
Advanced template for class target with autotuning	Data mining pipeline for a class target that contains autotuned tree, forest, neural network, and gradient boosting models.	SAS Pipeline	Jan 3, 2024, 2:57:57 PM	
Advanced template for interval target	Data mining pipeline that extends the intermediate template for an interval target by adding neural network, forest, GAM, and gradient boosting models. An ensemble model is also provided.	SAS Pipeline	Jan 3, 2024, 2:58:03 PM	
Advanced template for interval target with autotuning	Data mining pipeline that extends the intermediate template for an interval target by adding GAM and autotuned tree, forest, neural network, and gradient boosting models. An ensemble model is also provided.	SAS Pipeline	Jan 3, 2024, 2:58:01 PM	
Basic template for class target	Data mining pipeline that contains a Data, Imputation, Logistic Regression, and Model Comparison node connected in a linear flow.	SAS Pipeline	Jan 3, 2024, 2:58:05 PM	
Basic template for interval target	Data mining pipeline that contains a Data, Imputation, Linear Regression, and Model Comparison node connected in a linear flow.	SAS Pipeline	Jan 3, 2024, 2:58:06 PM	
Blank template	Data mining pipeline that contains only a data node.	SAS Pipeline	Jan 3, 2024, 2:58:07 PM	
Feature engineering template	Data mining pipeline that performs feature engineering.	SAS Pipeline	Jan 3, 2024, 2:58:05 PM	
Intermediate template for class target	Data mining pipeline that extends the basic template for a class target by adding a stepwise logistic regression model and a decision tree.	SAS Pipeline	Jan 3, 2024, 2:58:08 PM	

10. Because we have a class target (that is, Yes or No for a bad loan), let's get (statistically) wild and select **Advanced template for class target**. Then click **OK**. The Advanced template contains a Neural Network, a Stepwise Logistic Regression, a Forward Logistic Regression, a Decision Tree, a Gradient Boosting model, and a Forest model. Finally, it also has an Ensemble estimate of all those models put together.
11. We can then change the name of the pipeline to **Advanced Template** and then click **Save**.

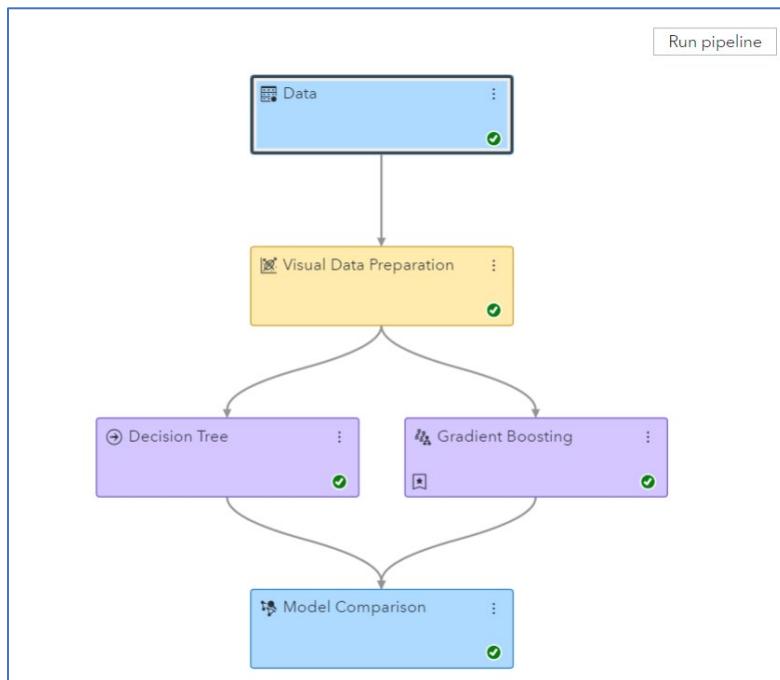


12. The new pipeline appears as follows:

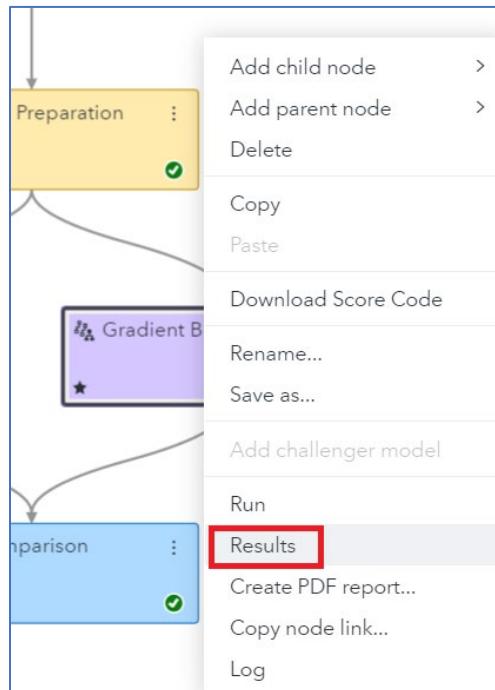


13. Again, look at all those modeling options! Yes, yes – that's seven for the price of one (click)! Because we're getting a bit pressed for time, let's simply run all the models – by clicking **Run pipeline** – and return to our original pipeline, **Interactive-Model Pipeline**, to examine those results.

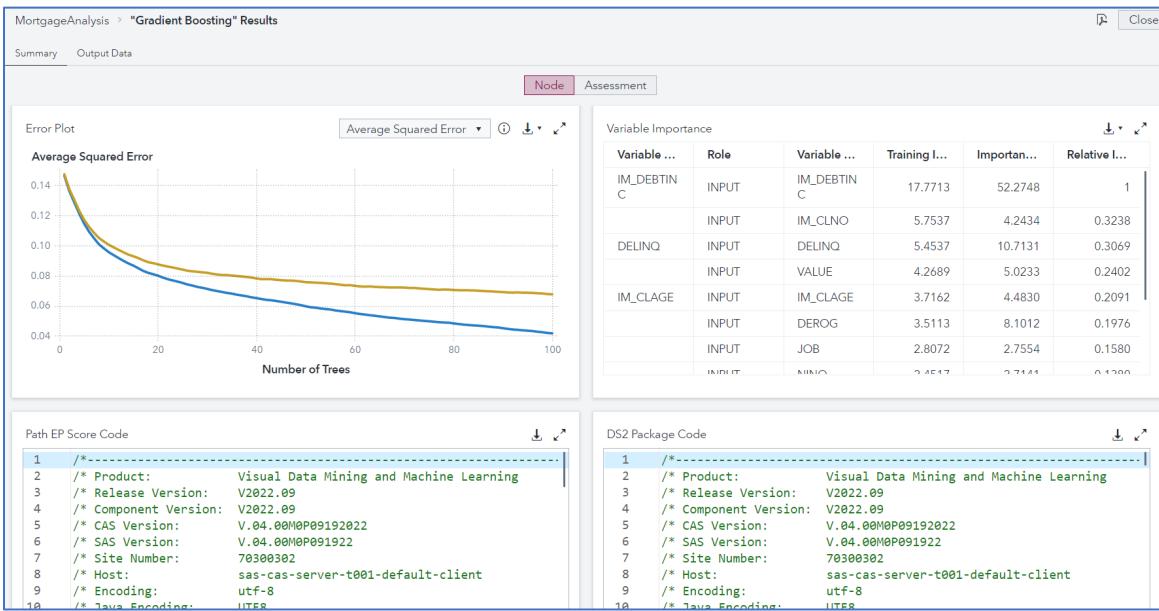
14. Green check marks in the pipeline indicate that the individual node has run. Thus, a completed **Interactive-Model Pipeline** appears as:



15. Let's examine our Gradient Boosting output. Right-click the node and select **Results**.



16. Explore both the **Node** and **Assessment** tabs so that you can better understand the results produced by SAS Model Studio.



17. Close **“Gradient Boosting” Results** when you've had your fill.

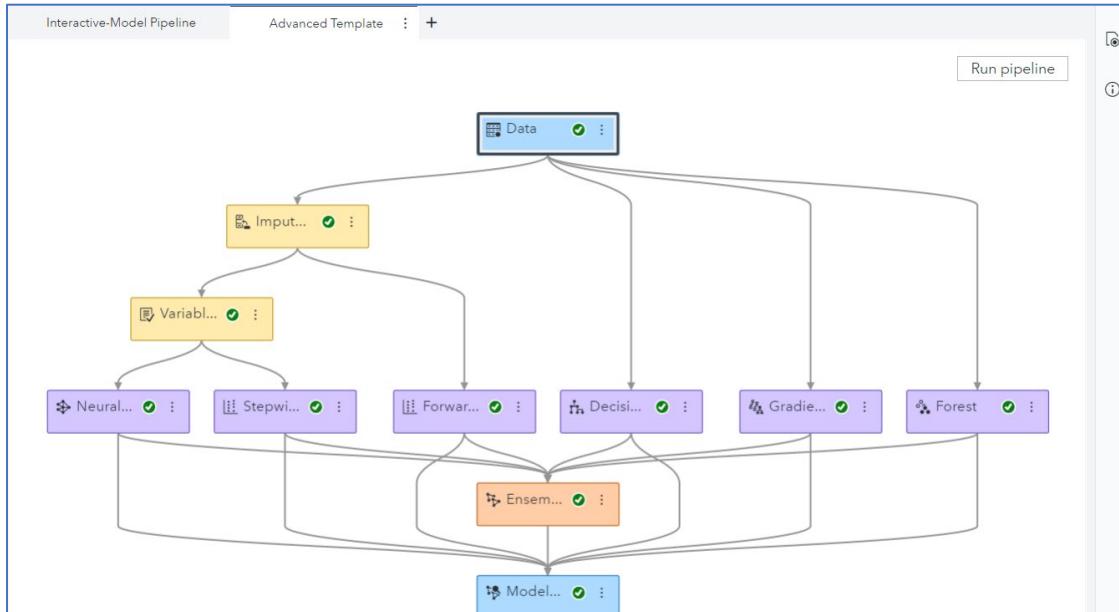
18. Next, select the **Model Comparison** node, right-click, and select **Results**. Results will enable you to directly compare the two models in our pipeline.

Model Comparison																
Champi...	Name	Algorith...	KS (Youn...	Accuracy	Averag...	Area Un...	Cumula...	Cumula...	Cutoff	Data Role	Depth	F1 Score	False Di...	False Po...	Gain	
*	Gradient Boosting	Gradient Boosting	0.7342	0.9038	0.0722	0.9350	4.6218	46.2185	0.5000	VALIDATE	10	0.7370	0.1886	0.0391	3.6218	
	Decision Tree	Decision Tree	0.6326	0.8758	0.0942	0.8582	4.1933	41.9331	0.5000	VALIDATE	10	0.6856	0.3066	0.0748	3.1933	

19. We can see that the Gradient Boosting model is selected as champion – because it has the highest KS (Younen) statistic. If you scroll to the right, you can see the Gradient Boosting model has a lower misclassification rate at 0.0962. Thus, the default Gradient Boosting model in SAS Model Studio – with all its hyper-tuned parameters – does a better job of modeling **BAD=1** than the decision tree used in both SAS Visual Analytics and SAS Model Studio **and** the default Gradient Boosting model used in SAS Visual Analytics. So, it looks like moving the modeling to SAS Model Studio improved our results. Good stuff!

20. After processing all the output from our first pipeline – we've nearly forgotten: we have a second pipeline to examine! Let's close the Model Comparison results window and return to the **Advanced Template**.

21. Ensure that the Advanced Template is finished running – which is indicated by a bunch of green check marks.



22. Check, check, check. Let's go straight to the Model Comparison and find which model is champion. Again, right-click the **Model Comparison** node, select **Results**, and then explore the following:

The screenshot shows two windows side-by-side. The left window is titled 'Model Comparison' and displays a table of model performance metrics. The right window is titled 'Properties' and shows a list of configuration properties.

Champi...	Name	Algorith...	KS (You... n)	Accuracy	Averag... ge	Area Un... der Curve	Cumula... tive KS	Cumula... tive Accuracy	Cutoff	Data Role	Depth	F1 Score	False Di... agnosis Rate	False Po... sitive Rate	Gain
★	Forest	Forest	0.7174	0.9038	0.0764	0.9320	4.5378	45.3782	0.5000	VALIDATE	10	0.7471	0.2136	0.0482	3.5378
	Gradient Boosting	Gradient Boosting	0.7104	0.9016	0.0748	0.9308	4.6499	46.4986	0.5000	VALIDATE	10	0.7389	0.2145	0.0475	3.6499
	Ensemble	Ensemble	0.7125	0.9021	0.0849	0.9262	4.4538	44.5378	0.5000	VALIDATE	10	0.7475	0.2292	0.0538	3.4538
	Stepwise Logistic Regression	Logistic Regression	0.6551	0.8770	0.0886	0.8878	4.3978	43.9776	0.5000	VALIDATE	10	0.6667	0.2739	0.0580	3.3978
	Neural Network	Neural Network	0.0259	0.8043	0.2497	0.5116	2.1008	21.0084	0.5000	VALIDATE	10	0.0541	0.2308	0.0021	1.1008

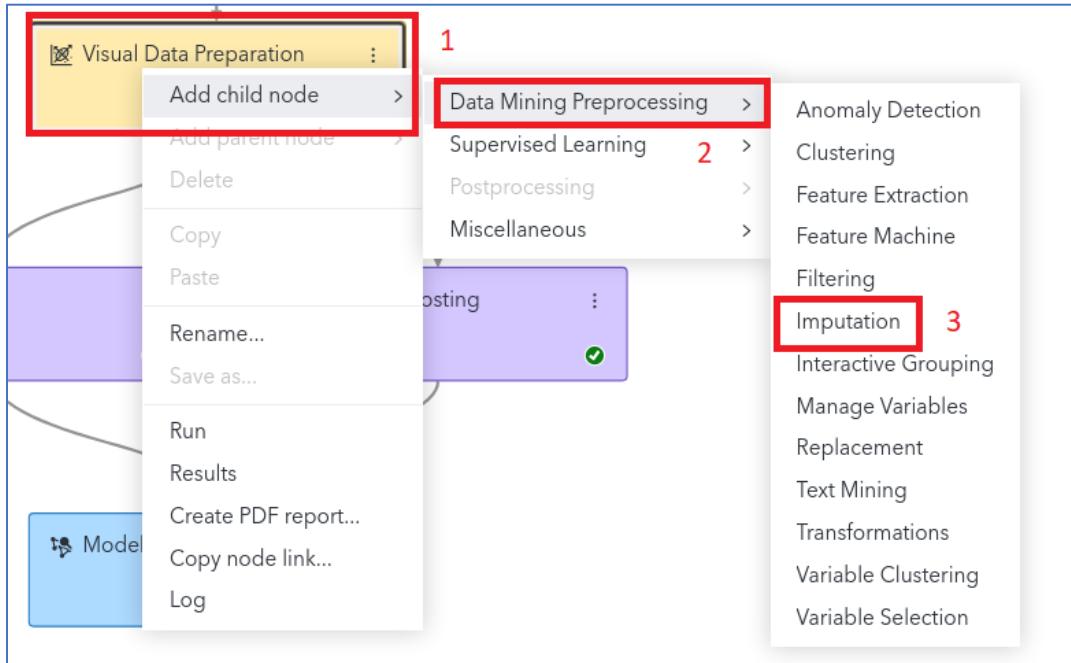
Properties	
Property Name	Property Value
selectionCriteriaClass	Kolmogorov-Smirnov statistic (KS)
selectionCriteriaInterval	Average squared error
selectionTable	Validate
selectionDepth	10
cutoff	0.50

23. What do we see? Forest appears to be the best model, based on the KS(Youden) statistics, with Gradient Boosting a very close second. The Forest also has the lowest misclassification rate in this modeling pipeline.

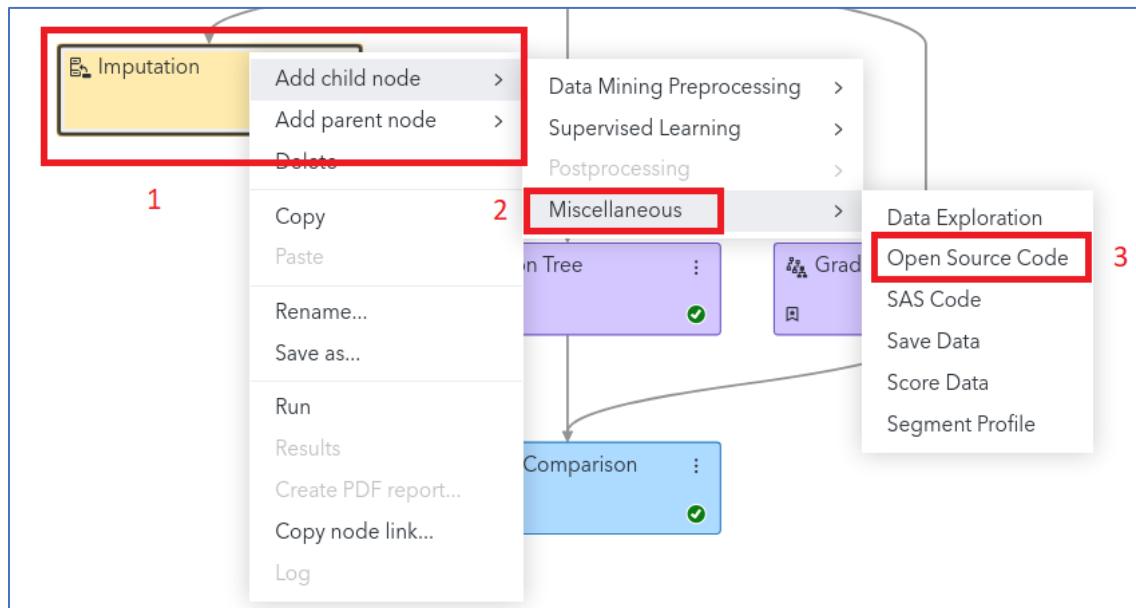
24. One lingering thought that you might have is “*Wait, I’m a Python + R coder – and I’d like to incorporate some of those machine learning models into my analysis. Can I do that?*” Of course you can!

25. Let’s return to our first pipeline – the Interactive-Model Pipeline – and show you how to incorporate some Python code. Don’t fret R fans: the process is very similar for R code... we just won’t have time to cover that today.

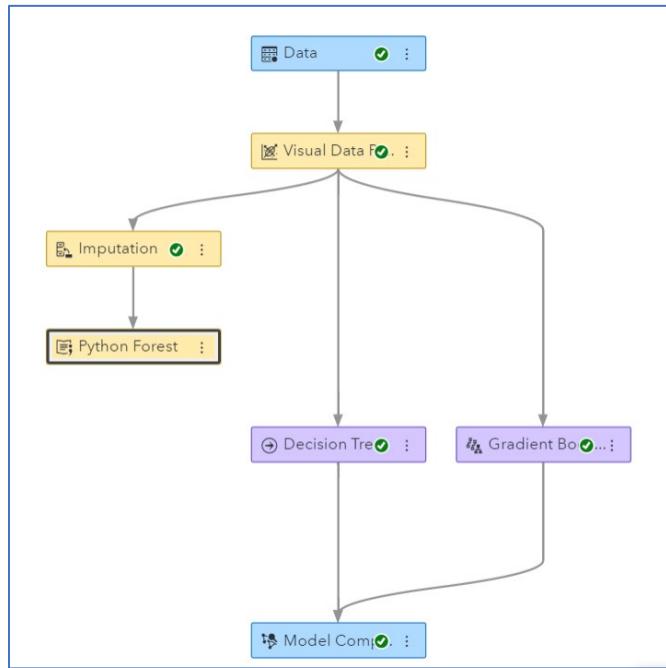
26. Because Open-Source nodes handle missing values differently than our traditional SAS nodes, we'll want to first add an n Imputation node to the flow. To do so, right-click the **Visual Data Preparation** node and select **Add child node** \Rightarrow **Data Mining Preprocessing** \Rightarrow **Imputation**.



27. Next, right-click that new **Imputation** node and select **Add child node** \Rightarrow **Miscellaneous** \Rightarrow **Open Source Code**.



28. Right-click the **Open Source Code** node and rename the node to **Python Forest**. Your pipeline should appear as follows:

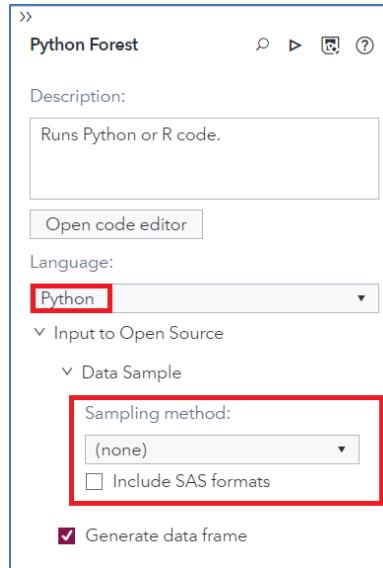


29. Under **Node options** for our **Python Forest**, verify that the language is set to **Python**. It should be Python by default.

30. Next, expand the **Data Sample** properties. Clear the check box for **Include SAS formats**.

A fun note: this property controls whether the downloaded data sent to Python or R should keep SAS formats. Why? Well, it is usually recommended that you keep SAS formats, and this should work in most cases. But... some numeric formats such as DOLLARw.d add a dollar sign and change the data type of the variable when exporting to CSV. In such cases, these formats must be removed.

31. Also change the **Sampling method** to **(none)**. Those last three changes:



32. Next, click the **Open Code Editor** button in the Node Options pane.

[Open Code Editor](#)

33. Now it's time to copy-and-paste some Python code. Grab the following:

```
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, axis=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, axis=1)
dm_scoreddf = pd.DataFrame(dm_model.predict_proba(fullX_enc),
columns=['P_BAD0', 'P_BAD1'])
```

34. Yes – that's a lot of code. Now paste it into the Training code section of the Python Forest window.

The screenshot shows the SAS Model Studio interface with the title "Model Studio - Build Models". The current project is "Mortgage Analysis > * Python Forest". In the center, there is a code editor titled "Python Variables" containing the following Python script:

```
# any configuration code, which is executed before the above code. During execution,
# this code is automatically prepended to every node that runs Python code.
#
# After running the node, the Python or R code window in the node results displays
# the actual code that was executed. START ENTERING YOUR CODE ON THE NEXT LINE.

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_trainddf[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)

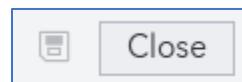
# Score full data
fullX_enc = fullX_enc.drop(dm_partitionvar, 1)
dm_scoreddf = pd.DataFrame(dm_model.predict_proba(fullX_enc), columns=['P_BAD0', 'P_BAD1'])


```

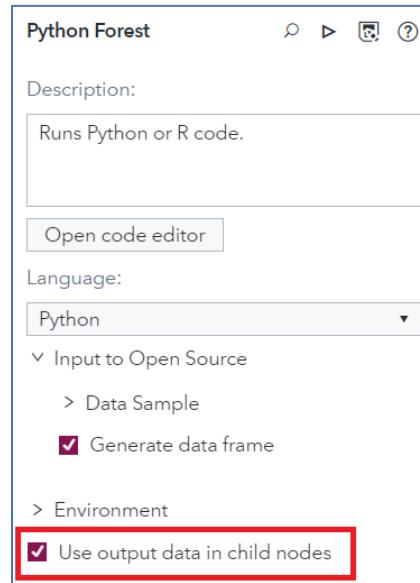
35. A few notes about the Python code – while saving more of the gory details for another time:

- This code fits a random forest classifier model in Python. The default values for the parameters that control the size of the trees (for example, **max_depth (default=None)**, **min_samples_leaf (default=1)**) lead to fully grown and unpruned trees, which can be very large data sets. To reduce memory consumption, the complexity and size of the trees are controlled by setting parameter values like the ones in the code above.
- The code that needs to be changed for different data sets is the last line – that is, naming your predictions with the **P_ + “target”** naming convention.
- It's important to note that we are just modeling the data here. Currently, we cannot perform data preparation within the Open Source Code node so that a subsequent node will recognize it. If this is necessary, either prepare data before Model Studio or perform both of the following: (1) open-source data preparation with the Open Source Code node (in the Preprocessing group) and (2) modeling with the Open Source Code node (in the Supervised Learning group).

36. In the upper right corner of the window, click the **Save** icon to save the Python code and then click the **Close** button to close the Code Editor window.



37. There is one more setting to change. Select the **Use output data in child nodes** property under **Environment**.



38. A little bit behind the why of the **Use output data in child nodes** button. Every time that this property is set, a copy of the output data is saved in the Model Studio project library, which can be used in later stages of the pipeline.

39. Now it is time to run the **Python Forest** node. Right-click the node and select **Run**.

40. Open the **Results** of the Python Forest node to examine output.

Variable Name	Training Importance
IMP_VALUE	0.0747
IM_CLAGE	0.1078
IM_DEBTINC	0.3004
IM_MORTDUE	0.0681
LOAN	0.0995
IMP_DELINQ_1	0.0206
IMP_DELINQ_2	0.0176
IMP_DELINQ_3	0.0247

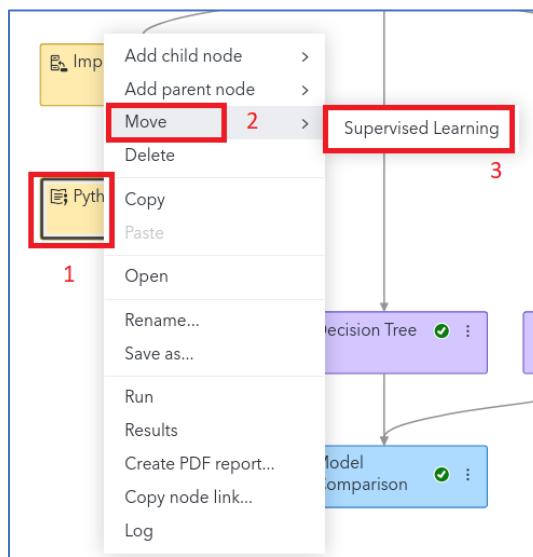
```

1 #-----
2 # Language: PYTHON
3 #-----
4 #Variable declarations
5 dm_nodedir = '/opt/kas/viya/config/var/tmp/compsrv/default/e61ce40a-d442-4942-b9b9-94f8f35c'
6 dm_dcc_target = 'BAD'
7 dm_partitionvar = '_PartInd_'
8 dm_partition_train_val = 1
9 dm_key = '_dmIndex'
10
11 dm_pkname = '_TSVCEQ50P8U0PHHFB46IDJDR_pk'
12 dm_pkpath = dm_nodedir + '/' + dm_pkname
13 dm_scoreinputs_path = dm_nodedir + '/dm_opensrcScoreInputs.csv'
14 dm_scoreoutputs_path = dm_nodedir + '/dm_opensrcScoreOutputs.csv'
15

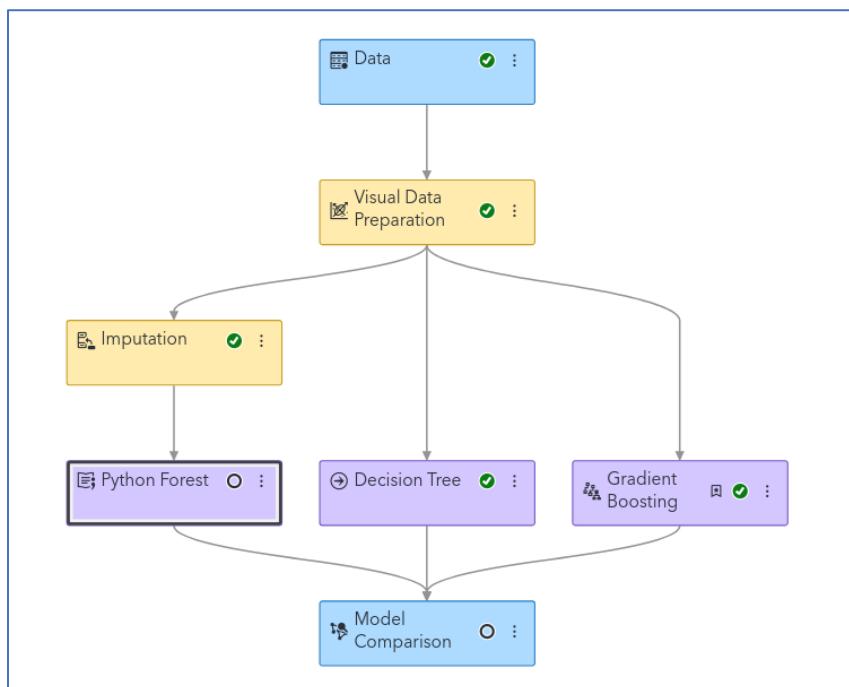
```

Property Name	Property Value
codeLocation	mlearning
dataMiningVersion	V2023.12
dframe	true
envVar	DM_PYPATH
exactPctLift	true
exportWithFormat	false
fullDatasetReconstitution	false
osCode_Language	py

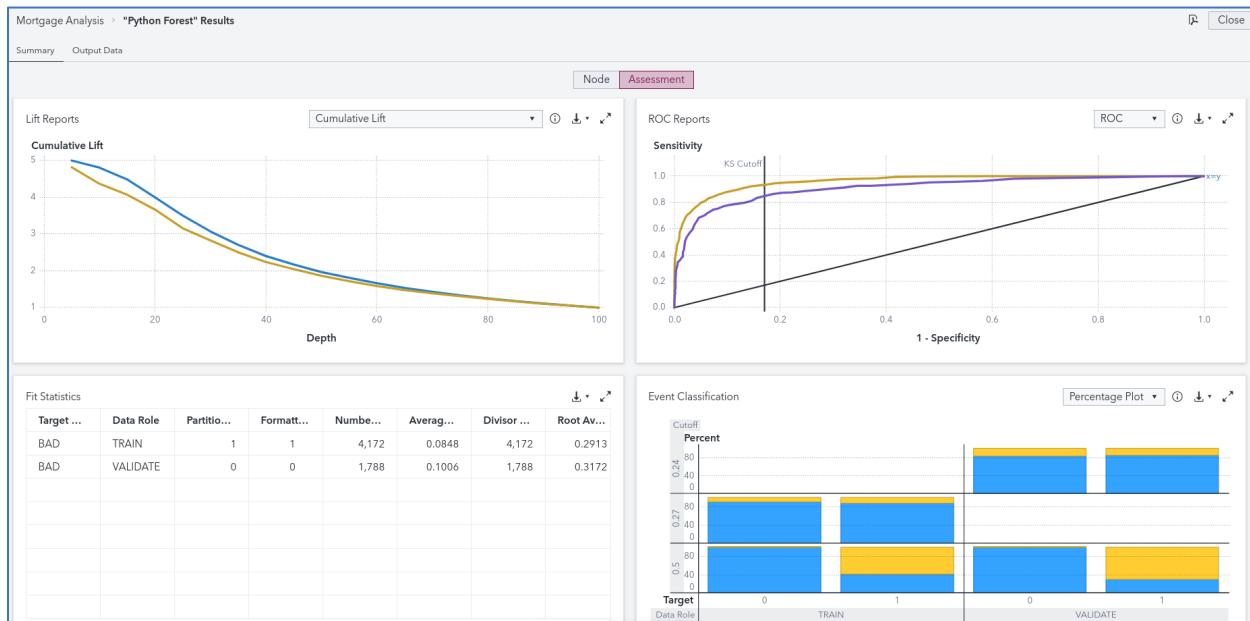
41. Examine the results for a bit, so that you get a better understanding of model performance. Click **Close** when you're satiated.
42. And after you're finished exploring, I have a thought-provoking question for you: *Where are the assessment results in the Open Source Code node?* Well, for model assessment results, you need to move the nodes to the supervised learning lane.
43. To get our Open Source Code node registered as a model, right-click the **Python Forest** node and select **Move** \Rightarrow **Supervised Learning**.



44. Your new view should appear as follows:



45. Python Forest has changed to purple, showing that the node is now part of the Supervised Learning group. Notice also that the nodes need to be rerun, as the green check mark has disappeared. So, click the **Run Pipeline** button to run the Python Forest and Model Comparison nodes.
46. Open the **Results** of the Python Forest. Click the **Assessment** tab to learn even more about model performance.

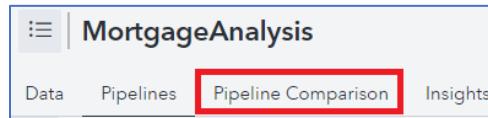


47. The usual assessment results are displayed. Explore until your heart is content and then close the results.
48. Open the **Results** of the Model Comparison node. Which model is our ultimate champion in the Interactive-Model Pipeline?

Model Comparison																
Champi...	Name	Algorith...	KS (You...)	Accuracy	Averag...	Area Un...	Cumula...	Cumula...	Cutoff	Data Role	Depth	F1 Score	False Di...	False Po...	Gain	
★	Gradient Boosting	Gradient Boosting	0.7342	0.9038	0.0722	0.9350	4.6218	46.2185	0.5000	VALIDATE	10	0.7370	0.1886	0.0391	3.6218	
	Decision Tree	Decision Tree	0.6326	0.8758	0.0942	0.8582	4.1933	41.9331	0.5000	VALIDATE	10	0.6856	0.3066	0.0748	3.1933	
	Python Forest	Open Source Code	0.6789	0.8574	0.1006	0.9119	4.3697	43.6975	0.5000	VALIDATE	10	0.4586	0.0526	0.0042	3.3697	

49. For our modeling, the Gradient Boosting algorithm still has the highest KS (Youden) statistic, followed closely by the Python Forest. So, it looks like both additions improve the fit over our default decision tree from SAS Visual Analytics. Woot!
50. Close the Model Comparison Results window.

51. Let's do two more things before we finish. The first is to examine the **Pipeline Comparison** tab, found here:



52. Rather than examining the champion of an individual pipeline, this will give us a champion of all the pipelines. We have only two potential models to choose from – but we could have many, many more! Click the **Pipeline Comparison** tab and examine the following:

Mortgage Analysis				
Data	Pipelines	Pipeline Comparison	Insights	
Filter	Data: Validate			
<input type="checkbox"/>	Champion ↓	Name	Algorithm Name	Pipeline Name
<input checked="" type="checkbox"/>		Gradient Boosting	Gradient Boosting	Interactive-Model Pipeline
<input type="checkbox"/>		Forest	Forest	Advanced Template
				KS (Youden)
				0.734
				0.717

53. No surprises here – a Gradient Boosting model is the overall champion. But it's the one configured in the **Advanced Template** that wins the crown, with a slightly higher KS (Youden) statistic. And, as stated earlier, your results might differ marginally due to different random samples being used.

54. Finally, let's click the **Insights** tab, found to the right of Pipeline Comparison.

A screenshot of the Mortgage Analysis interface showing the Insights tab selected. The top section displays a Project Summary table with information such as Project Target (BAD), Event Percentage (19.9497%), Pipelines (2), Project Champion (Gradient Boosting), Created By (Student), and Modified (February 27, 2024, 05:55:47 PM). To the right is a Project Notes section with a placeholder 'Add comments here'. Below this are two charts: 'Most Common Variables Selected Across All Models' (a horizontal bar chart showing variables like DEROG, IM_CLAGE, VALUE, etc.) and 'Assessment for All Models' (a stacked bar chart comparing Forest and Advanced Template models across various metrics).

55. Storytelling is an important part of the data analytics process, as you communicate your findings with others. The **Insights** tab contains several helpful resources – from overall project summaries to assessments of models and variables – that can help you better explain your analytic process and overall champion model.
56. Explore this output at your leisure. Then smile – you've made it through your first day at iLink Mortgage, Inc.!

Appendix

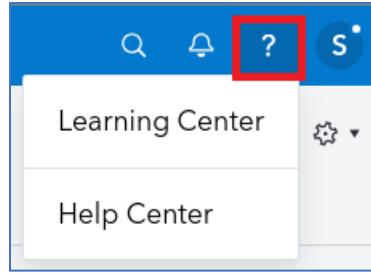
Appendix A: Access Software

There are a variety of ways to access SAS Viya. Let us count some ways.

1. Academics have access to the SAS Viya for Learners platform. Navigate to the SAS Viya for Learners web page, https://www.sas.com/en_us/software/viya-for-learners.html.
2. Non-Academics have a couple of different options for free SAS Viya trials:
 - a. Start here for a general overview:
https://www.sas.com/en_us/software/viya/try-or-buy.html
 - b. 14-day free trial in our SAS Trial Environment:
https://www.sas.com/en_us/trials/software/viya/viya-trial-form.html
 - c. 14-day free trial of SAS Viya on Azure Marketplace (Microsoft still charges for Azure infrastructure costs): <https://azuremarketplace.microsoft.com/en-us/marketplace/apps/sas-institute-560503.sas-viya-on-azure?tab=overview>

Appendix B: Helpful Documentation

Here are two options for further help and guidance.

1. **SAS Help Center** in SAS Viya for Learners
 - a. It's been lurking in the upper-right-hand corner all long!A screenshot of the SAS Viya for Learners interface. At the top right, there is a blue header bar with several icons: a magnifying glass for search, a bell for notifications, a question mark for help, and a user profile icon. The question mark icon is highlighted with a red square. A dropdown menu is open below these icons, showing "Learning Center" and "Help Center". To the right of the dropdown is a gear icon with a dropdown arrow.
 - b. The two available options provide more access to SAS **Documentation** and the **SAS Help Center**.
2. **SAS Videos**
 - a. Prefer videos instead?
 - b. Check out the SAS Software YouTube channel:
<https://www.youtube.com/user/sassoftware>

3. SAS Communities

- a. SAS Communities is a great, user support community that can answer many of your outstanding SAS questions.
- b. Go here - <https://communities.sas.com/> - to start a conversation today.

Appendix C: Recommended Follow-up Learning

There are numerous e-learning courses available to support your learning journey, which you can start today. The following courses would be most helpful in following up with the materials from this SAS On-the-Job:

- SAS Visual Analytics 1 for SAS Viya: Basics
- SAS Visual Analytics 2 for SAS Viya: Advanced
- SAS Visual Statistics in SAS Viya: Interactive Model Building
- Machine Learning Using SAS Viya
- Programming for SAS Viya

You can gain access to these self-paced courses through the [SAS Learning Subscription](#). Sign up for a free 7-day trial today!