SAS Software Tour with iLink Mortgage, Inc.

SAS On-the-Job Activity

Purpose

This activity is a hands-on guide to using SAS Studio, Visual Analytics, and Model Studio within SAS Viya for Learners, while presenting learning in a real-world job situation. Students will modify and prepare data, create descriptive visualizations, estimate and compare machine learning models, and build and enhance visual reports.

SAS Software

We will use SAS Viya for Learners 4.0 (VFL4) software in this activity. Users can also use VFL3.5, but there may be slight differences in the user experience.

Industry Alignment

This On-the-Job activity fits within the mortgage industry of the financial sector. We explore data containing information about homeowner loan activity – and seek to predict which customers are most likely to default on their loans.

Table of Contents

Activity Notes and Requirements

Learning Objectives

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

- Creating visualizations for descriptive statistics
- Modifying data to be used across SAS Viya for Learners applications
- Building and comparing machine learning models

Estimated Completion Time

This activity contains three tasks. All together this activity will take students roughly 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 with SAS Viya for Learners.

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 the SAS Viya for Learners platform
- Learning to use Snippets in SAS Studio
- Running SAS Studio programs and examining results for key insights

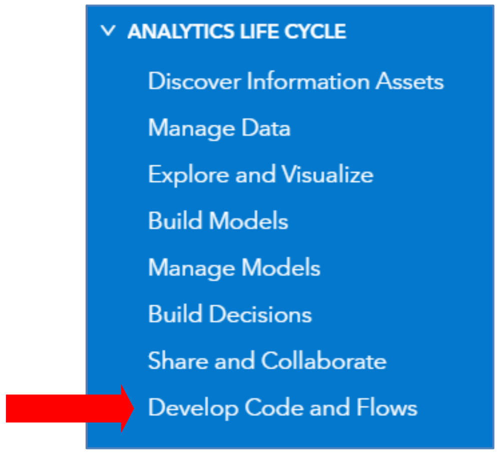
Estimated Time of Completion

This task is estimated to take about 30 minutes.

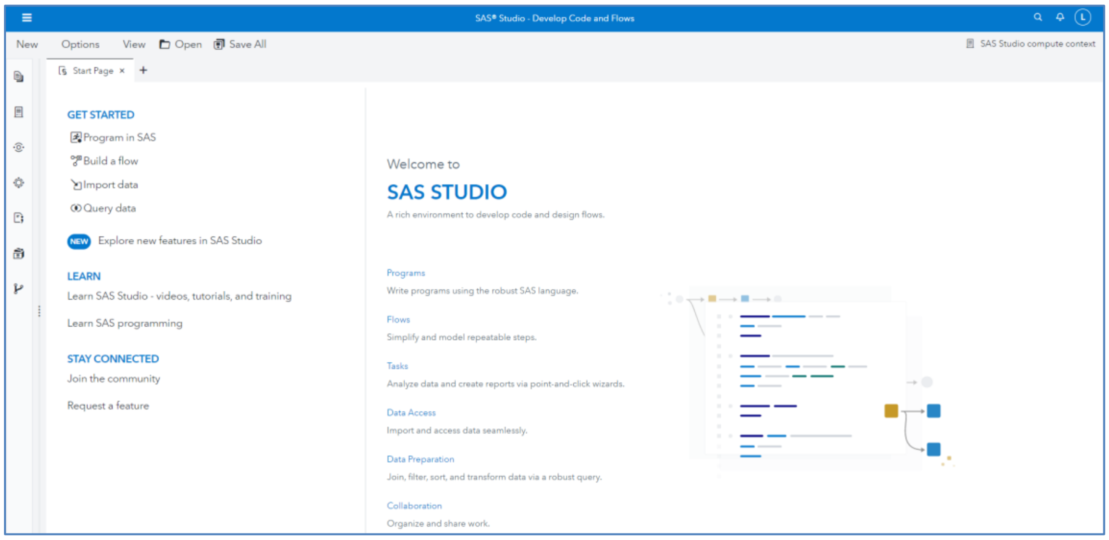
Task: Introduction to SAS Studio

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 into SAS Viya for Learners.
2. You will be brought to SAS Drive, which is your starting point in SAS Viya for Learners.
3. From the top left corner, select the **Applications** menu ≡ and click **Develop SAS Code and Flows** from the Analytics Life Cycle options.



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



Welcome to

SAS STUDIO

A rich environment to develop code and design flows.

Programs
Write programs using the robust SAS language.

Flows
Simplify and model repeatable steps.

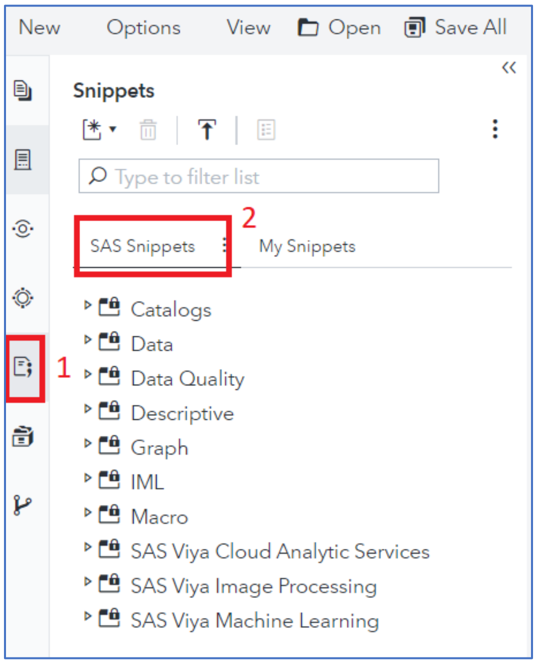
Tasks
Analyze data and create reports via point-and-click wizards.

Data Access
Import and access data seamlessly.

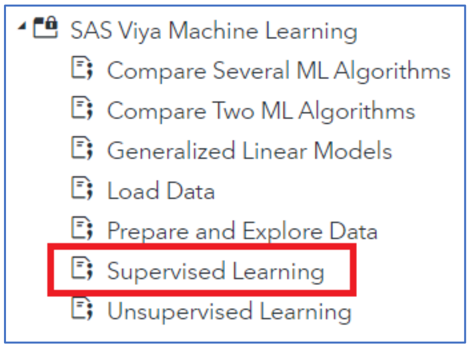
Data Preparation
Join, filter, sort, and transform data via a robust query.

Collaboration
Organize and share work.

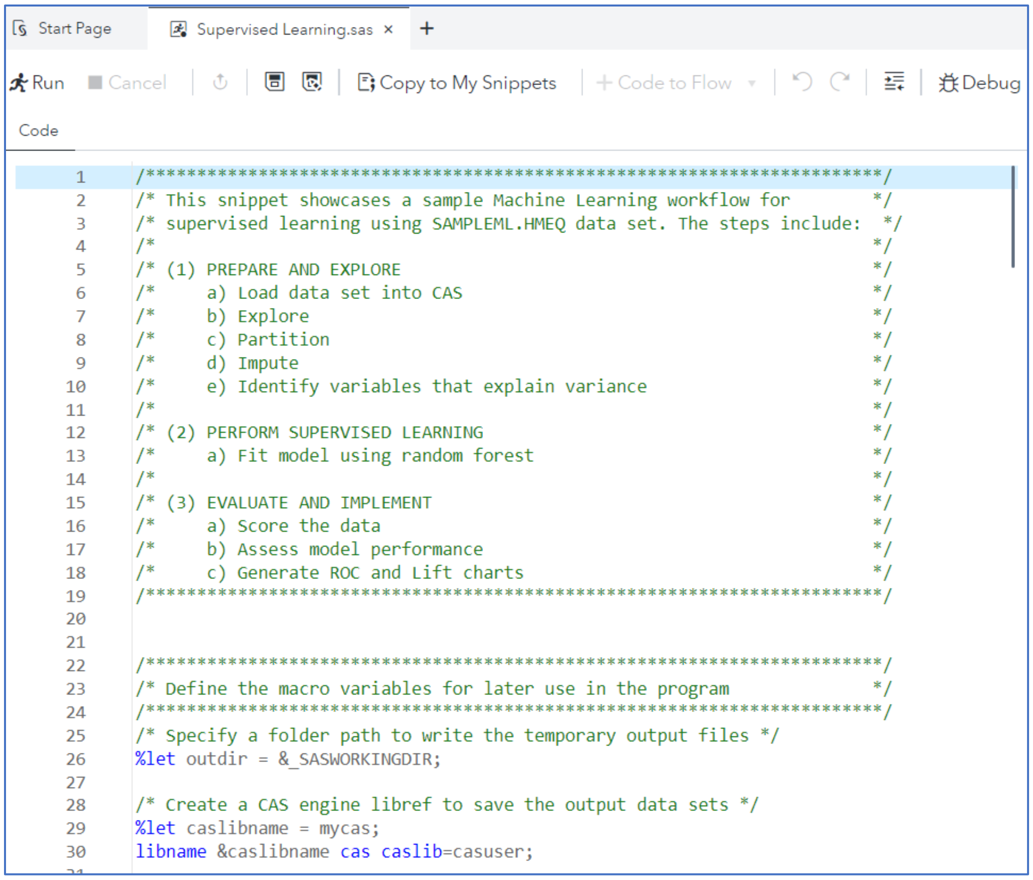
5. From the **Start Page**, you can begin your analytical adventure in a variety of ways, from programming in SAS Studio to building flows. You can even embark on a learning journey with videos, tutorials, and training.
6. For this analysis, we'll simply access a program Alex saved as a **Snippet** (i.e., 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 section **SAS Viya Machine Learning** and open the **Supervised Learning** snippet:


SAS Viya Machine Learning
Compare Several ML Algorithms
Compare Two ML Algorithms
Generalized Linear Models
Load Data
Prepare and Explore Data
Supervised Learning
Unsupervised Learning

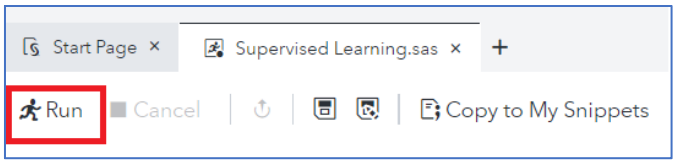
8. The snippet starts with the following preamble:



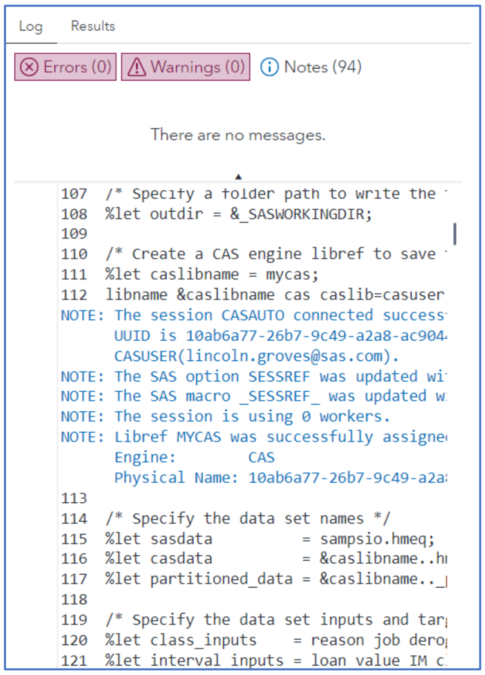
```
/***************************************************************************/
/* This snippet showcases a sample Machine Learning workflow for          */
/* supervised learning using SAMPLEML.HMEQ data set. The steps include:   */
/*                                                                         */
/* (1) PREPARE AND EXPLORE                                                 */
/*     a) Load data set into CAS                                           */
/*     b) Explore                                                          */
/*     c) Partition                                                        */
/*     d) Impute                                                           */
/*     e) Identify variables that explain variance                         */
/*                                                                         */
/* (2) PERFORM SUPERVISED LEARNING                                         */
/*     a) Fit model using random forest                                    */
/*                                                                         */
/* (3) EVALUATE AND IMPLEMENT                                              */
/*     a) Score the data                                                   */
/*     b) Assess model performance                                         */
/*     c) Generate ROC and Lift charts                                     */
/***************************************************************************/


/***************************************************************************/
/* Define the macro variables for later use in the program                */
/***************************************************************************/
/* Specify a folder path to write the temporary output files */
%let outdir = &_SASWORKINGDIR;

/* Create a CAS engine libref to save the output data sets */
%let caslibname = mycas;
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, wrangles the data, 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. Once better understanding our analytical tasks, let the program rip! In the program window for *Supervised Learning.sas*, click on the **Run code** icon:



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

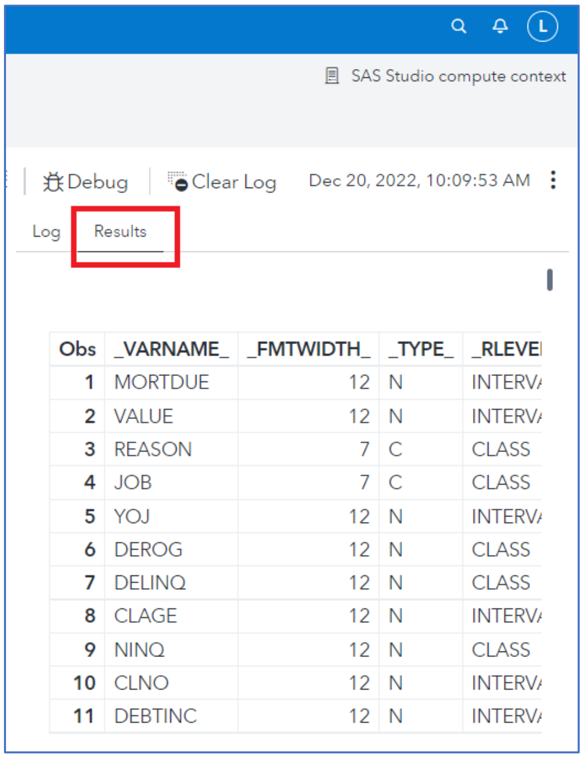


There are no messages.

```
/* Specify a folder path to write the
%let outdir = &_SASWORKINGDIR;

/* Create a CAS engine libref to save
%let caslibname = mycas;
libname &caslibname cas caslib=casuser
NOTE: The session CASAUTO connected success
      UUID is 10ab6a77-26b7-9c49-a2a8-ac904
      CASUSER(lincoln.groves@sas.com).
NOTE: The SAS option SESSREF was updated wi
NOTE: The SAS macro _SESSREF_ was updated w
NOTE: The session is using 0 workers.
NOTE: Libref MYCAS was successfully assigne
      Engine:        CAS
      Physical Name: 10ab6a77-26b7-9c49-a2a

/* Specify the data set names */
%let sasdata          = sampsio.hmeq;
%let casdata          = &caslibname..h
%let partitioned_data = &caslibname.._

/* Specify the data set inputs and tar
%let class_inputs    = reason job dero
%let interval_inputs = loan value IM c
```

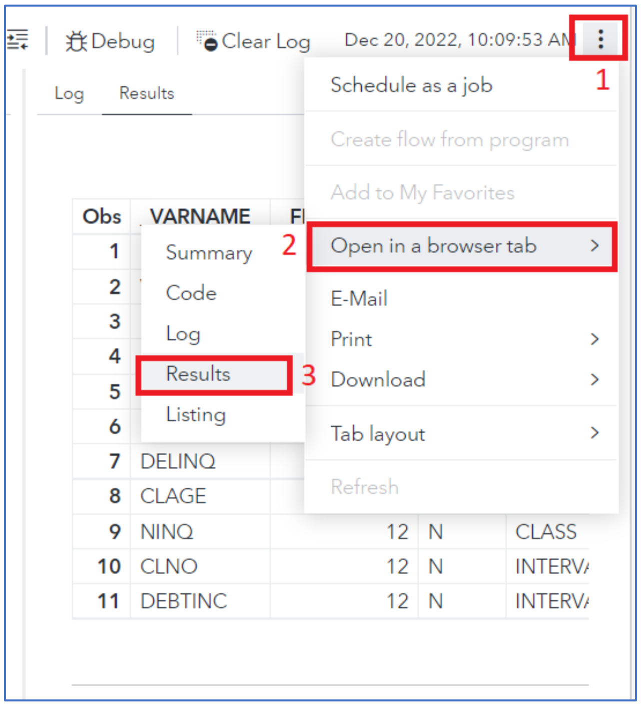
** Check that the log contains no Errors or Warnings.*

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

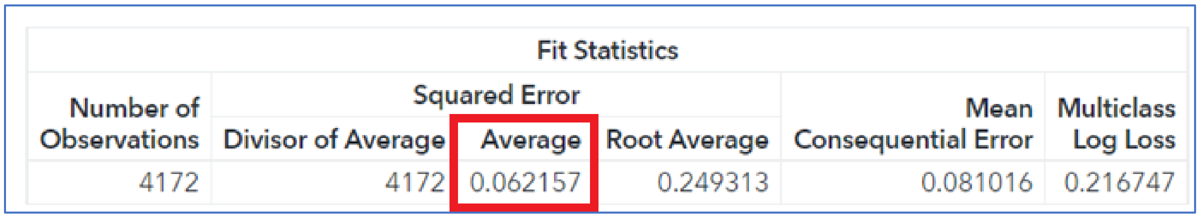


Obs	_VARNAME_	_FMTWIDTH_	_TYPE_	_RLEVE
1	MORTDUE	12	N	INTERV
2	VALUE	12	N	INTERV
3	REASON	7	C	CLASS
4	JOB	7	C	CLASS
5	YOJ	12	N	INTERV
6	DEROG	12	N	CLASS
7	DELINQ	12	N	CLASS
8	CLAGE	12	N	INTERV
9	NINQ	12	N	CLASS
10	CLNO	12	N	INTERV
11	DEBTINC	12	N	INTERV

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



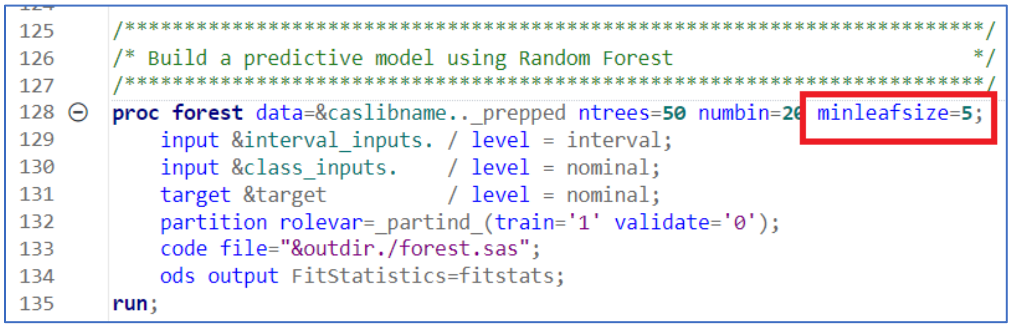
14. Examine the model output, including how the data are transformed and which plots are used to assess the model's performance. In other words, scroll, and scroll some more!
15. Find and record the model **Fit Statistics**:



Fit Statistics					
	Squared Error				
Number of Observations	Divisor of Average	Average	Root Average	Mean Consequential Error	Multiclass Log Loss
4172	4172	0.062157	0.249313	0.081016	0.216747

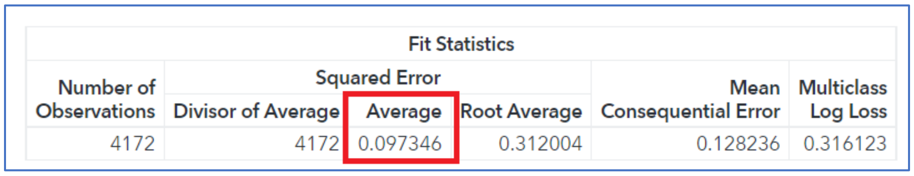
Note: your results may differ marginally, as we didn't set a seed to ensure that the results were perfectly replicable. No worries though! This also illustrates that different training/validation/testing samples will produce slightly different results.

16. Now let's tweak the model slightly and see if 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 refences 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;
```

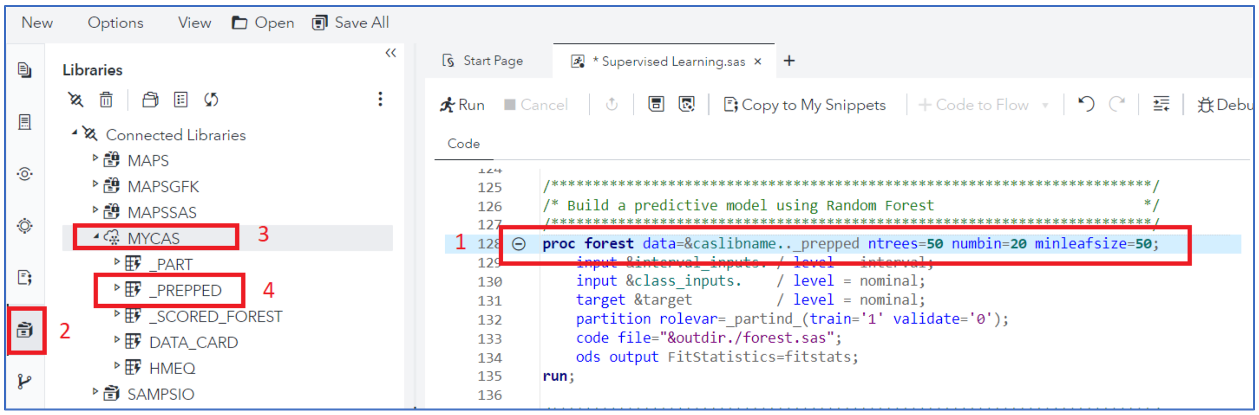
17. Let's change **minleafsize=50**. Then resubmit the code using Run .
18. Does this model fit the data better, as indicated by the average squared error? My findings:



Fit Statistics					
	Squared Error				
Number of Observations	Divisor of Average	Average	Root Average	Mean Consequential Error	Multiclass Log Loss
4172	4172	0.097346	0.312004	0.128236	0.316123

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**. Since we don't want to spend much time cleaning the data again, let's examine which datasets have been saved for us, by first looking at the dataset used in the forest modeling.
21. While I don't expect you to read SAS code already – particularly those crazy macros and the "&" notation – 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:



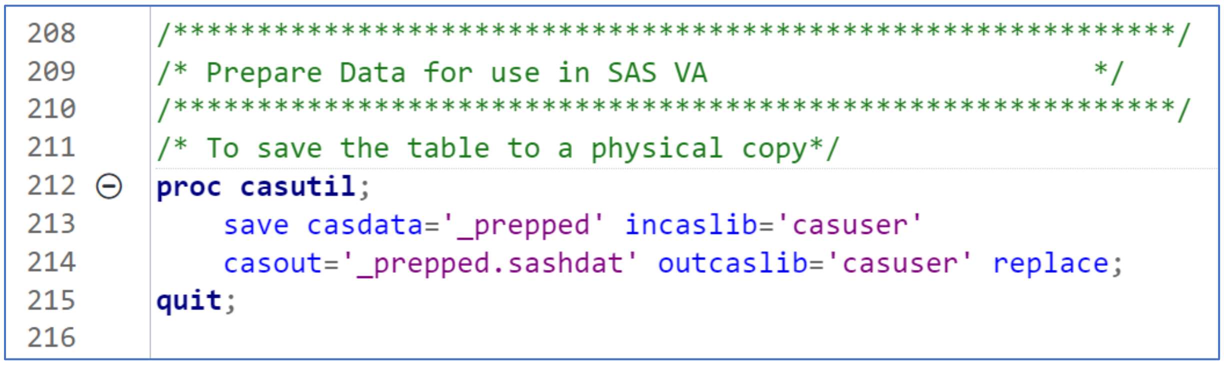
```
/*************************************************************************/
/* Build a predictive model using Random Forest                          */
/*************************************************************************/
proc forest data=&caslibname.._prepped ntrees=50 numbin=20 minleafsize=50;
    input &interval_inputs. / level   interval;
    input &class_inputs.    / level = nominal;
    target &target          / level = nominal;
    partition rolevar=_partind_(train='1' validate='0');
    code file="&outdir./forest.sas";
    ods output FitStatistics=fitstats;
run;
```

*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, so that it can be accessible in SAS Visual Analytics. Again, the goal here isn't to turn you into a full-fledged hacker on your first day, so we've written the following code for you to use:

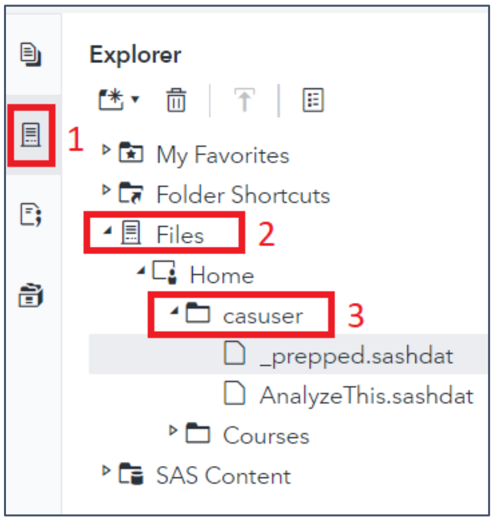
```
/*************************************************************/
/* Prepare Data for use in SAS VA                            */
/*************************************************************/
/* To save the table to a physical copy*/
proc casutil;
        save casdata='_prepped' incaslib='casuser'
        casout='_prepped.sashdat' outcaslib='casuser' replace;
quit;
```

23. Add the code above to the end of your program:



```
/*************************************************************/
/* Prepare Data for use in SAS VA                          */
/*************************************************************/
/* To save the table to a physical copy*/
proc casutil;
    save casdata='_prepped' incaslib='casuser'
    casout='_prepped.sashdat' outcaslib='casuser' replace;
quit;
```

24. Without any subsection of the code selected, resubmit the entire program via Run. And, voila, **_prepped** should now be accessible in SAS Visual Analytics!
25. We'll do a quick check to ensure that **_prepped** is in your CASUSER folder. Follow these clicks to investigate:



26. 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 for Learners platform
- Learning to use the Auto Chart feature
- Creating visualizations using different types of variables (categorical, numeric, geographic)
- Using predictive modeling functions within the Visual Analytics application

Estimated Time of Completion

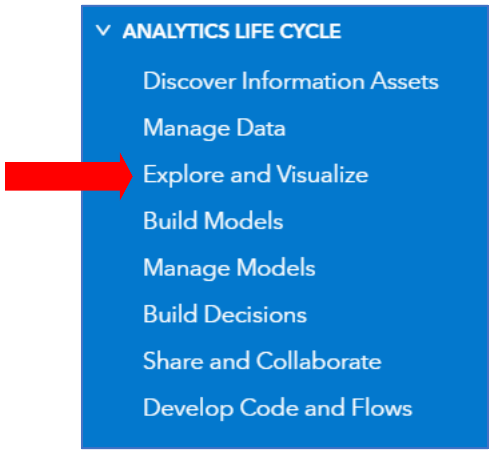
This task is estimated to take about 30 minutes.

Task: Introduction to SAS Visual Analytics

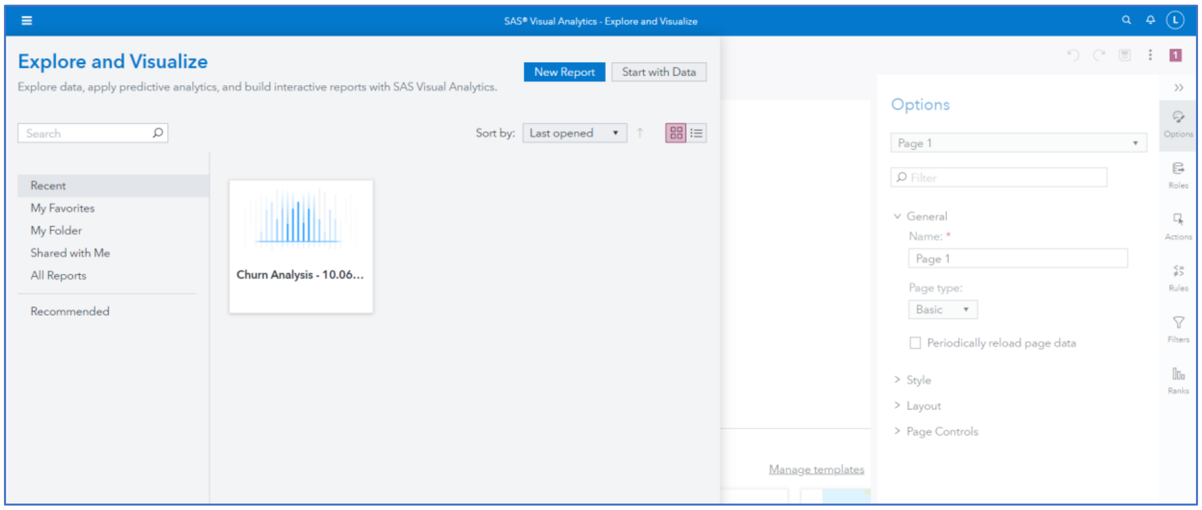
Coding is great. But as a new employee of iLink Mortgage, Inc. – and new user of SAS – you'd likely prefer to start your analytical journey with some no-code options. Fortunately, SAS Visual Analytics has several helpful tools to guide you along that path!

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

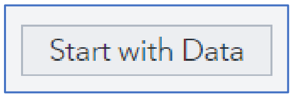
1. Let's move over into SAS Visual Analytics. Return to the **Applications** menu (☰) in the upper left corner and select **Explore and Visualize** under the *Analytics Life Cycle* options.



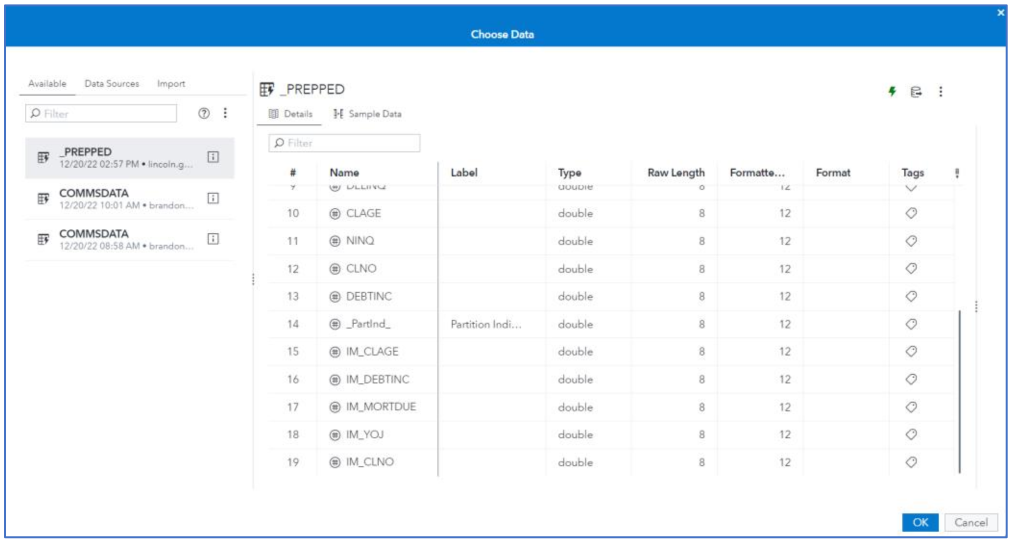
2. Welcome to **SAS Visual Analytics**! While your view will differ marginally, it should appear roughly akin to the following:



3. To continue our SAS VA adventure, choose **Start with Data** from the main **Explore and Visualize** window:



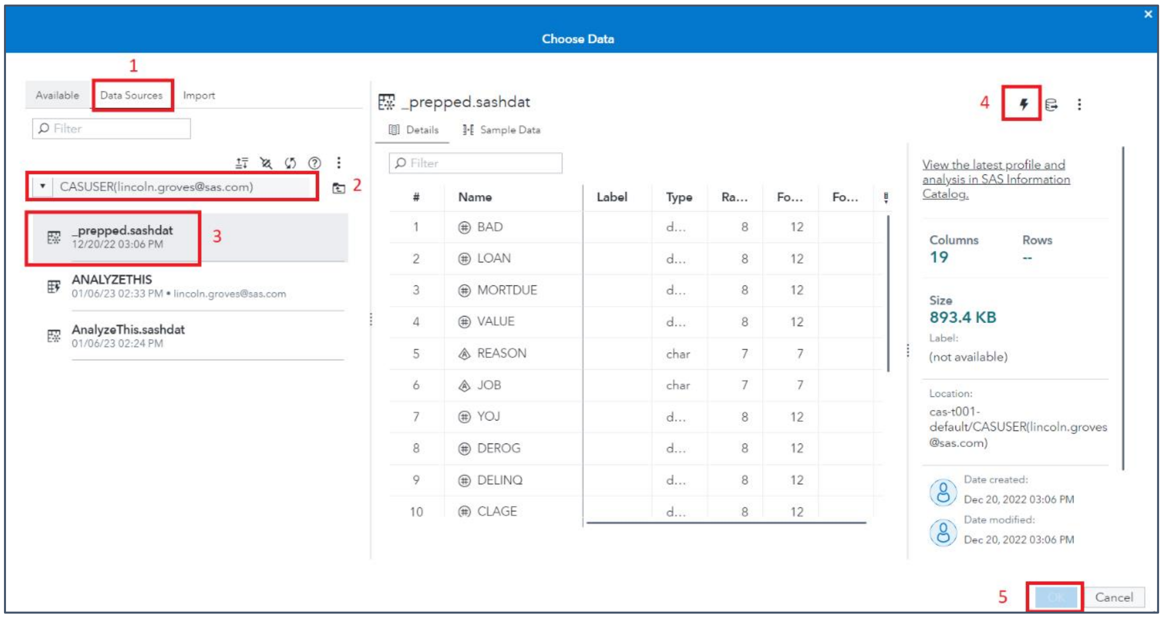
4. And… this is the part where you try to remember, and find, the dataset used for modeling in SAS Studio. No worries if you forgot that name – it's **_PREPPED**. And it could be one of the available in-memory tables:



#	Name	Label	Type	Raw Length	Formatte...	Format	Tags
10	CLAGE		double	8	12		
11	NINQ		double	8	12		
12	CLNO		double	8	12		
13	DEBTINC		double	8	12		
14	_PartInd_	Partition Indi...	double	8	12		
15	IM_CLAGE		double	8	12		
16	IM_DEBTINC		double	8	12		
17	IM_MORTDUE		double	8	12		
18	IM_YOJ		double	8	12		
19	IM_CLNO		double	8	12		

5. However, if this dataset is not already loaded into memory, then select the **Data Sources** tab. Then select the default folder and drill down to your CASUSER folder. **_PREPPED**.sashdat is there – but not yet loaded into memory (which is why you didn't

see it above). How do we get data into memory? Click the **Load into Memory** button – i.e., the lightning bolt – then click **OK**. Follow these steps:



CASUSER(lincoln.groves@sas.com)

_prepped.sashdat
12/20/22 03:06 PM

ANALYZETHIS
01/06/23 02:33 PM • lincoln.groves@sas.com

AnalyzeThis.sashdat
01/06/23 02:24 PM

_prepped.sashdat

#	Name	Label	Type	Ra...	Fo...	Fo...
1	BAD		d...	8	12	
2	LOAN		d...	8	12	
3	MORTDUE		d...	8	12	
4	VALUE		d...	8	12	
5	REASON		char	7	7	
6	JOB		char	7	7	
7	YOJ		d...	8	12	
8	DEROG		d...	8	12	
9	DELINQ		d...	8	12	
10	CLAGE		d...	8	12	

View the latest profile and analysis in SAS Information Catalog.

Columns 19
Rows --

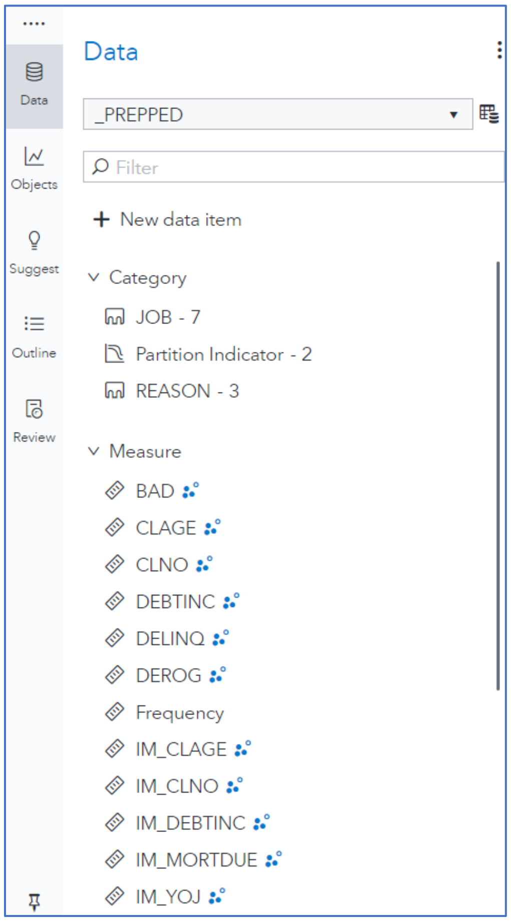
Size
893.4 KB
Label:
(not available)

Location:
cas-t001-default/CASUSER(lincoln.groves@sas.com)

Date created:
Dec 20, 2022 03:06 PM
Date modified:
Dec 20, 2022 03:06 PM

6. While a lot of the metadata definitions transferred from the SAS Studio environment, some of them did not. By cleaning up the metadata in SAS VA, 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…

   Click the **Data** button on the left-hand side pane and examine the current variables in your analysis:



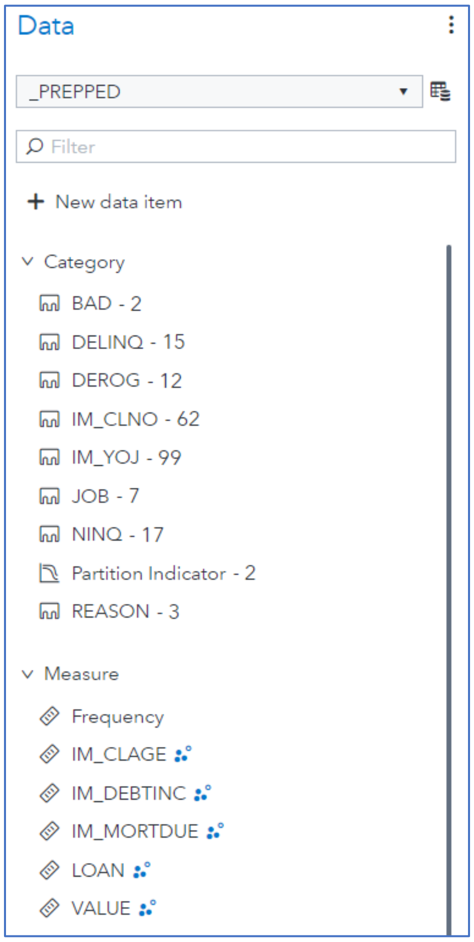
Category

- JOB - 7
- Partition Indicator - 2
- REASON - 3

Measure

- BAD
- CLAGE
- CLNO
- DEBTINC
- DELINQ
- DEROG
- Frequency
- IM_CLAGE
- IM_CLNO
- IM_DEBTINC
- IM_MORTDUE
- IM_YOJ

7. Whew! There are a lot of variables! For our upcoming modeling, we'll want to match the following metadata definitions:


Data

_PREPPED

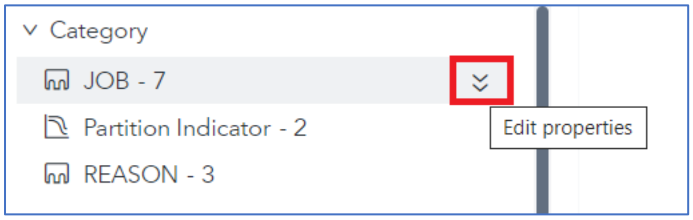
Filter

+ New data item

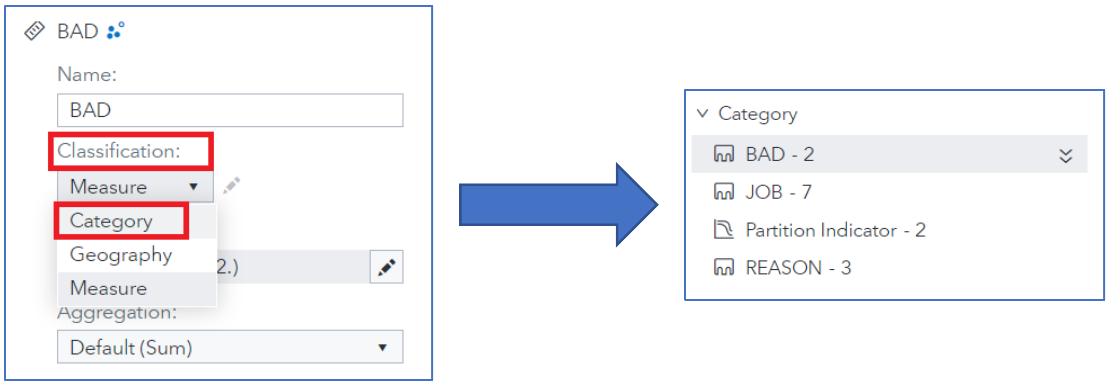
Category
- BAD - 2
- DELINQ - 15
- DEROG - 12
- IM_CLNO - 62
- IM_YOJ - 99
- JOB - 7
- NINQ - 17
- Partition Indicator - 2
- REASON - 3

Measure
- Frequency
- IM_CLAGE
- IM_DEBTINC
- IM_MORTDUE
- LOAN
- VALUE

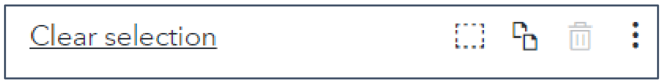
8. And how do we do that, you might ask? Each variable has an **Edit properties** button which allows you to change the metadata for that variable:


Category
- JOB - 7
- Partition Indicator - 2
- REASON - 3

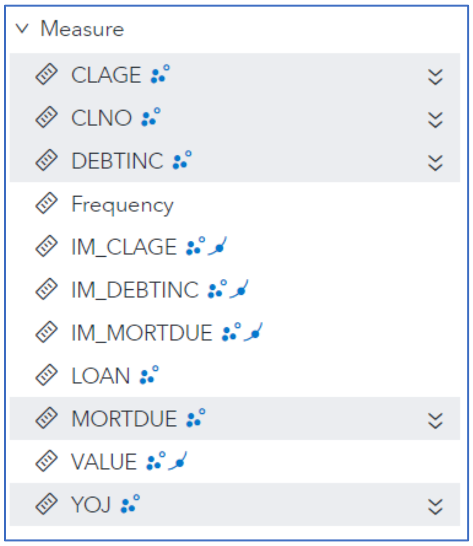
9. For our analysis, we want to ensure that BAD, DELINQ, DEROG, IM_CLNO, IM_YOJ, JOB, NINQ, and REASON are all classified as **Category**. Why? Because we'd like to treat them as Yes/No variables, instead of the usual 1/0 binary variable that is coded.
10. *BAD*, our outcome variable in the modeling, is possibly still classified as a **Measure**. If this is true, we can change this by clicking on 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:



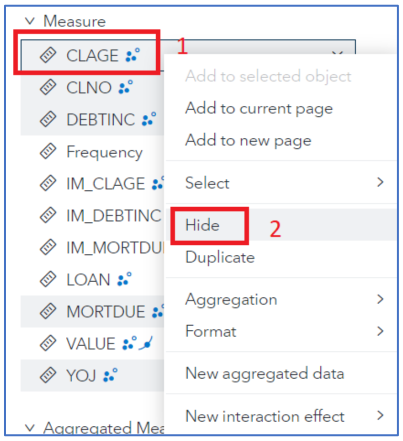
11. If required, make the change from **Measure** to **Category** for these other variables: DEROG, DELINQ, NINQ, IM_CLNO, and IM_YOJ
12. Finally, we'd like to exclude some of the Measures – as these variables will not be used in our modeling. Before getting started, ensure that no variables are selected. Why? Because those variables will be included in the action. If they are, click on the "Clear Selection" option at the bottom of the data pane:



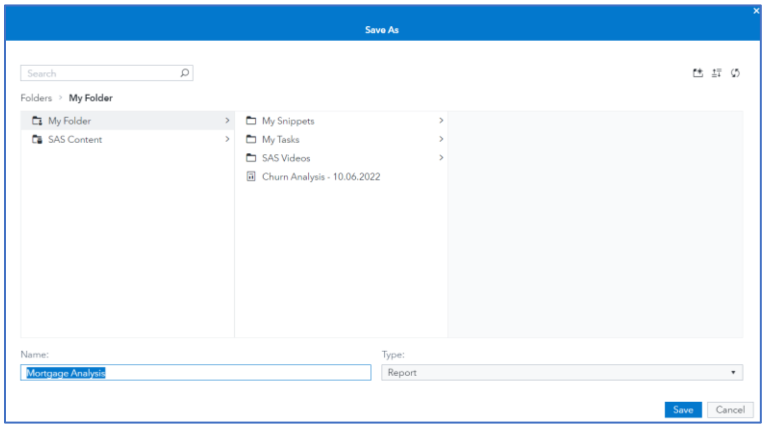
13. Then, select CLAGE, CLNO, DEBTINC, MORTDUE, and YOJ as shown:



14. Next, right-click on one of the selected variables, then select **Hide**:



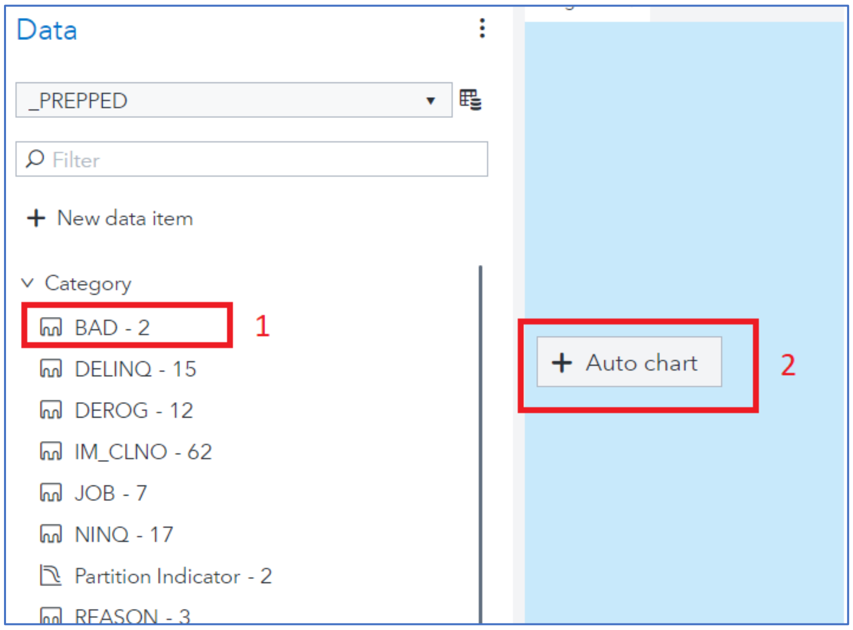
15. Presto! Variables be gone – and your metadata should now match the screenshot from above.
16. This is a great time to save your program. Find the classic **floppy-disk** icon in the upper-right hand corner. Name the analysis something creative like **Mortgage Analysis** and **Save** it in the default location:



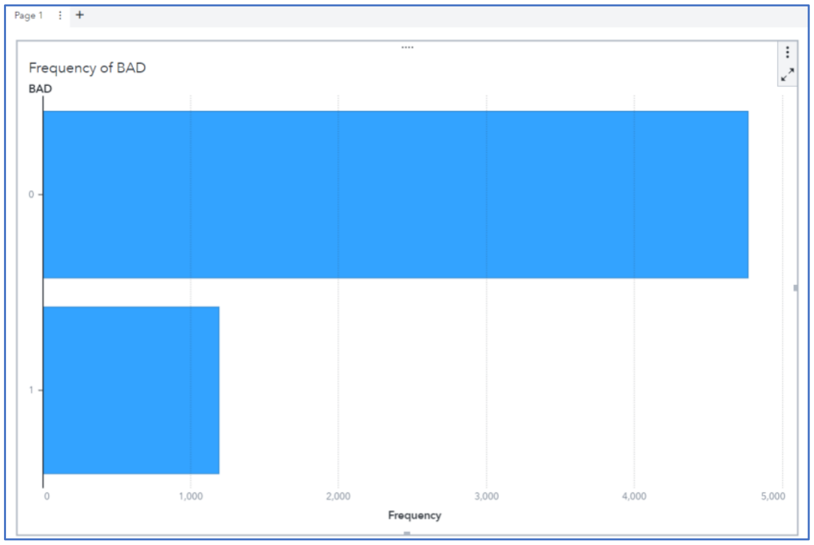
Name: Mortgage Analysis
Type: Report

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

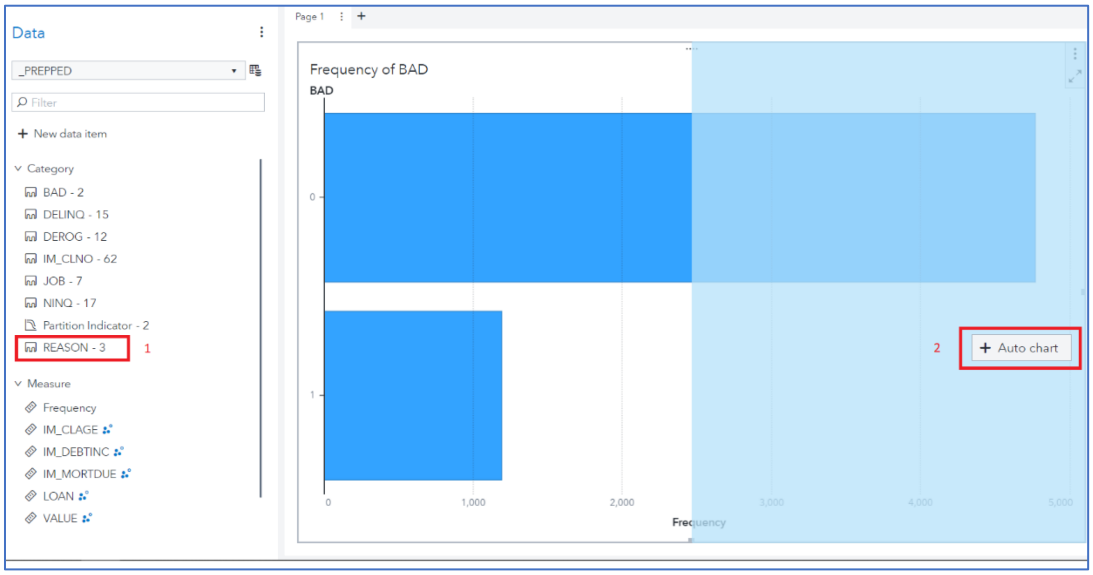
17. Now that the VA report is established and the data are where they need to be, let's better understand select variables by using **Auto charts**. With the **Data** icon selected, click on *BAD* and then drag and drop the variable to the canvas:



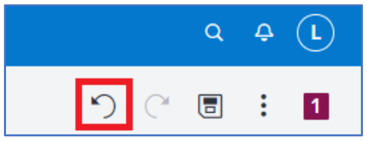
18. With **Auto chart**, Visual Analytics will automatically intuit what you'd like plotted. Here it will produce a bar chart with frequencies:



19. 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:

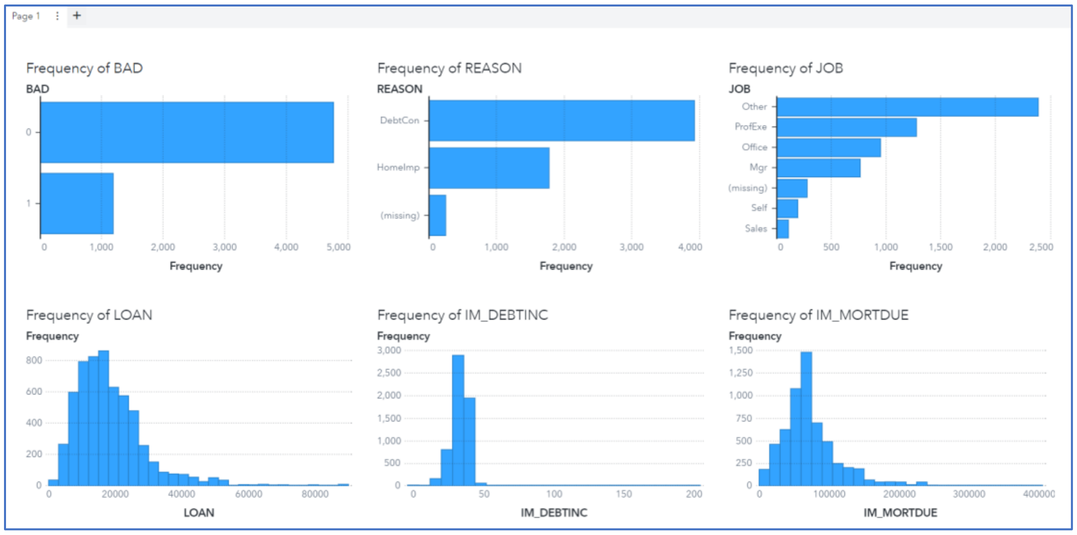


20. Variables can be placed above/below or left/right of existing objects. And if you make a mistake then no worries – there is an undo button in the upper-right hand corner:

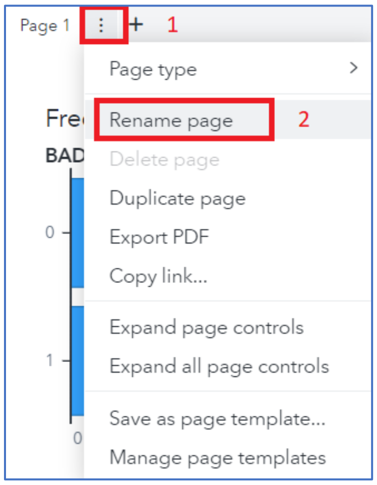


Use it. And use it often when needed.

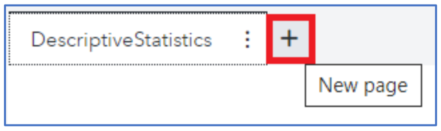
21. Now replicate the following:



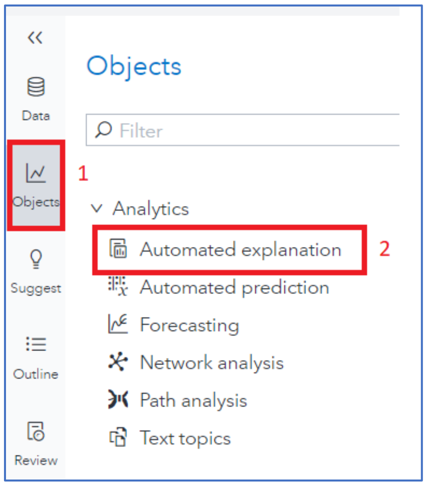
22. Change the name of “Page 1” to “DescriptiveStatistics”, by clicking on the **Page menu** and then selecting **Rename page** (or you can simply double click on the page tab):



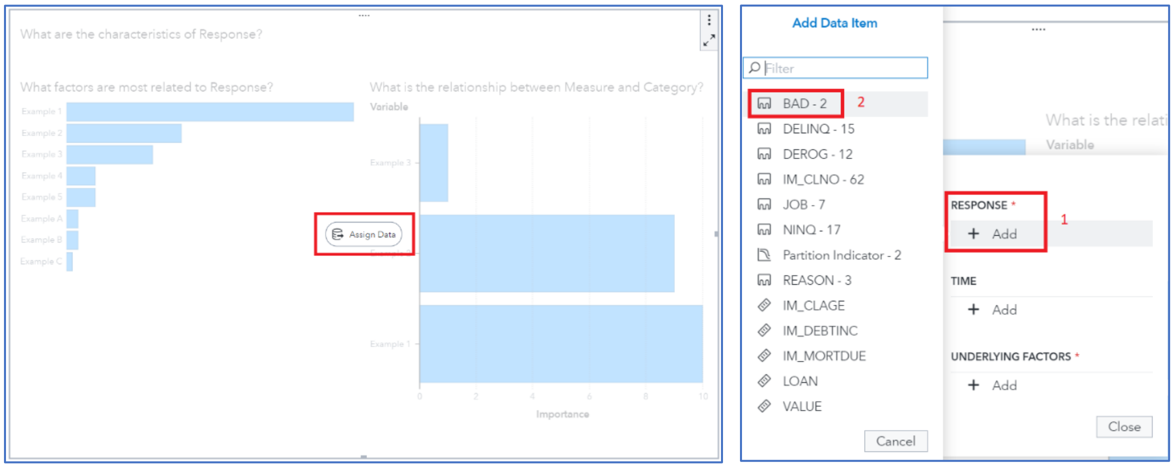
23. Create a **New page** by clicking the + button next to the “DescriptiveStatistics” tab. In this new canvas, rename “Page 2” to “AutomatedExplanation”.



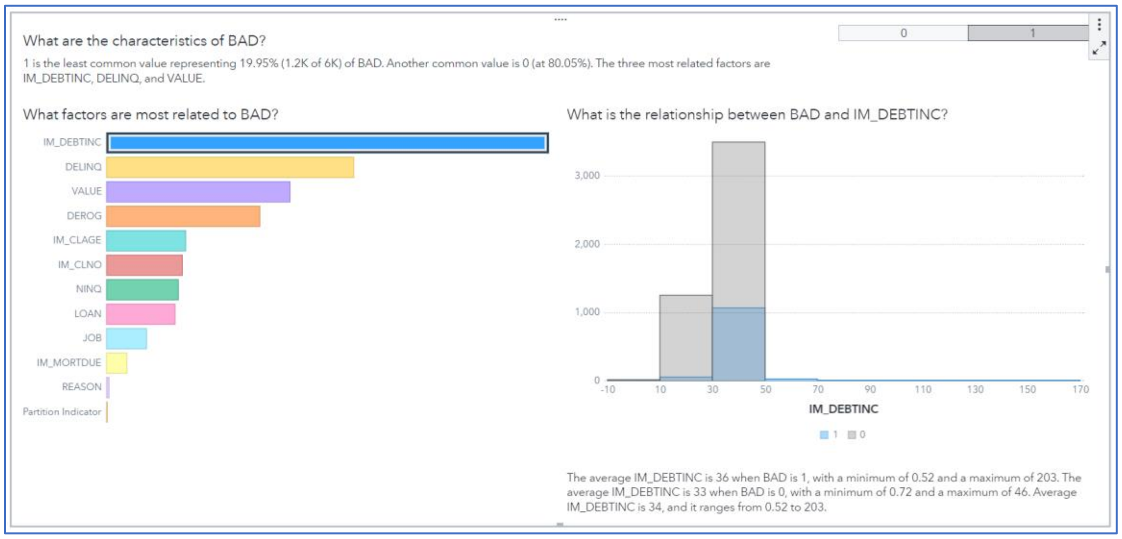
24. Let’s add an **Automated Explanation** object to this new canvas. It can be found under **Objects** under the left-hand pane and then **Analytics**:



25. Next, click on **Assign Data** in the **Automated Explanation** object and then choose *BAD* as the **Response**:



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


What are the characteristics of BAD?

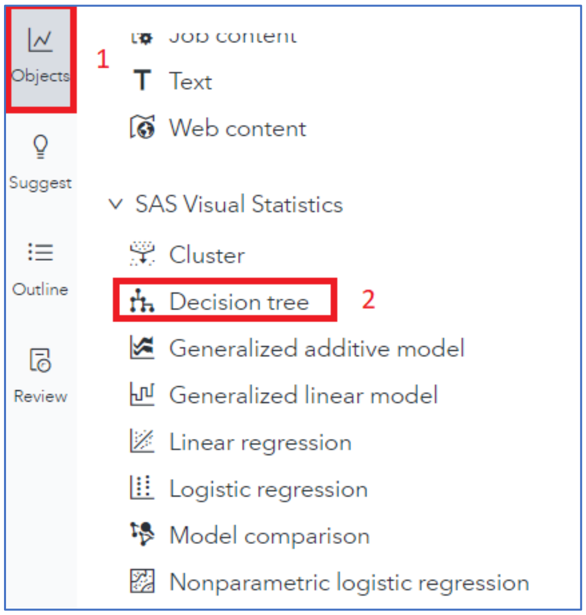
1 is the least common value representing 19.95% (1.2K of 6K) of BAD. Another common value is 0 (at 80.05%). The three most related factors are IM_DEBTINC, DELINQ, and VALUE.

The average IM_DEBTINC is 36 when BAD is 1, with a minimum of 0.52 and a maximum of 203. The average IM_DEBTINC is 33 when BAD is 0, with a minimum of 0.72 and a maximum of 46. Average IM_DEBTINC is 34, and it ranges from 0.52 to 203.

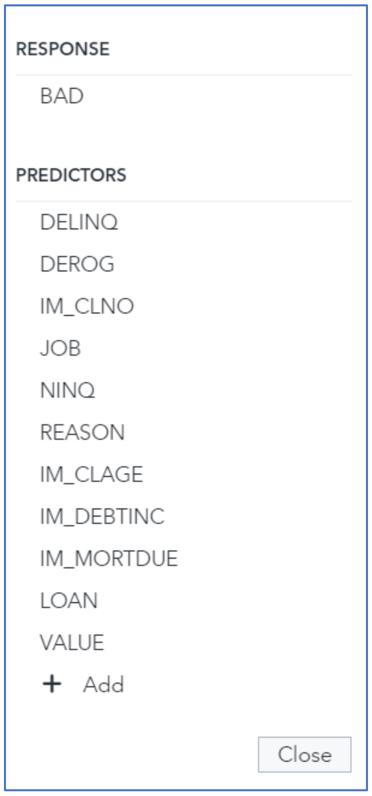
27. While **Automated Analysis** is a fast way to understand the bi-variate relationship between variables, we have more complex modeling tools. In other words, we have much cooler stuff. Let’s start with a **Decision Tree** and then build more sophisticated models from there.

28. Create a **New page** and change the name to “DecisionTree”.

29. From the **Objects** pane on the left, find the **Decision Tree** object under **SAS Visual Statistics** and drag-and-drop it on the DecisionTree canvas:



30. It's again time to **Assign Data**. Select **BAD** as the **Response** and the following variables as the **Predictors**. Then click **Close**:


RESPONSE

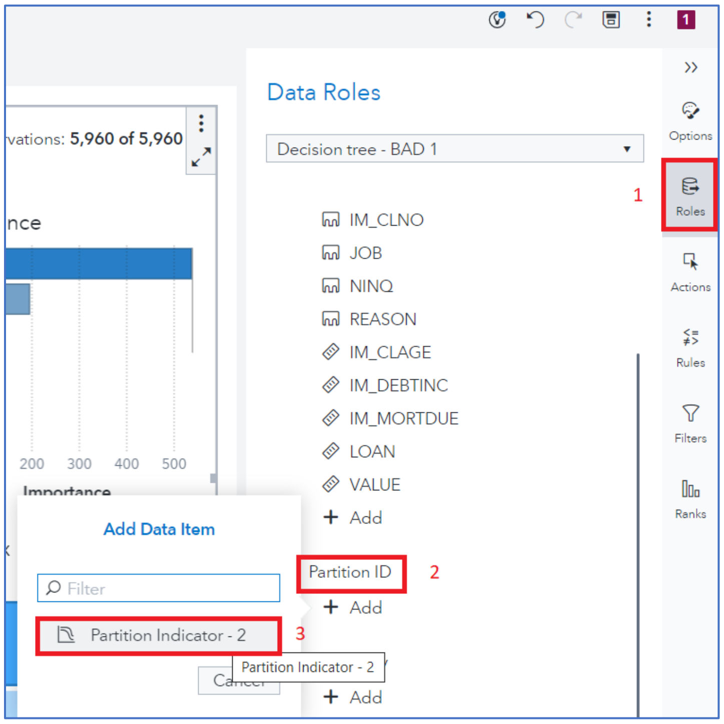
BAD

PREDICTORS

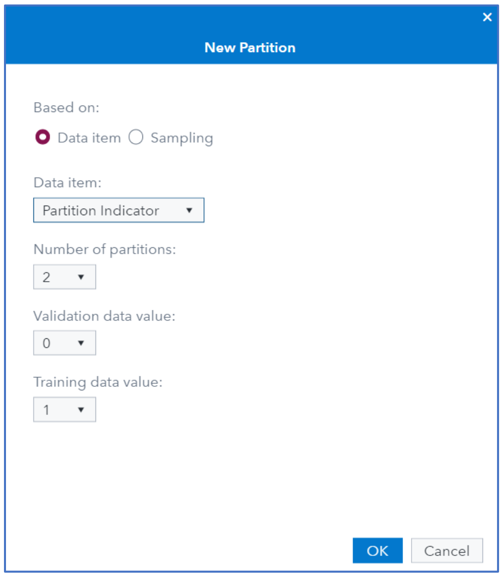
DELINQ
DEROG
IM_CLNO
JOB
NINQ
REASON
IM_CLAGE
IM_DEBTINC
IM_MORTDUE
LOAN
VALUE

31. Now – because we're aspiring data scientists, after all – let's add a data partition to allow for honest assessment. First, check to see if you have a partition already defined. From the

right-pane, select **Roles**. Then, scroll down to **Partition ID**. See if the **Partition Indicator** variable exists:



32. If yes, then choose *Partition Indicator* as your Partition ID – and skip the next two steps. If not, you'll need to create a partition. No worries, we've got you covered.
33. For the unlucky few out there, Partition Indicator was not imported as an actual partition. To fix this – by creating a new variable – click on **Data ➔ + New data item ➔ Partition**. Match the setting below in the pop-up window, and then click OK.



New Partition

Based on:
[✓ Data item] Sampling

Data item:
Partition Indicator

Number of partitions:
2

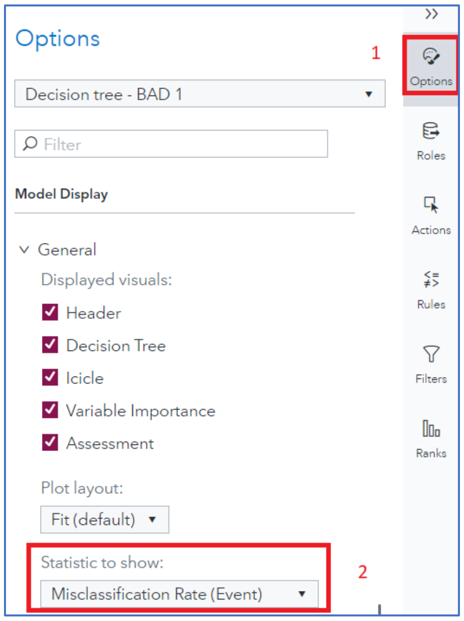
Validation data value:
0

Training data value:
1

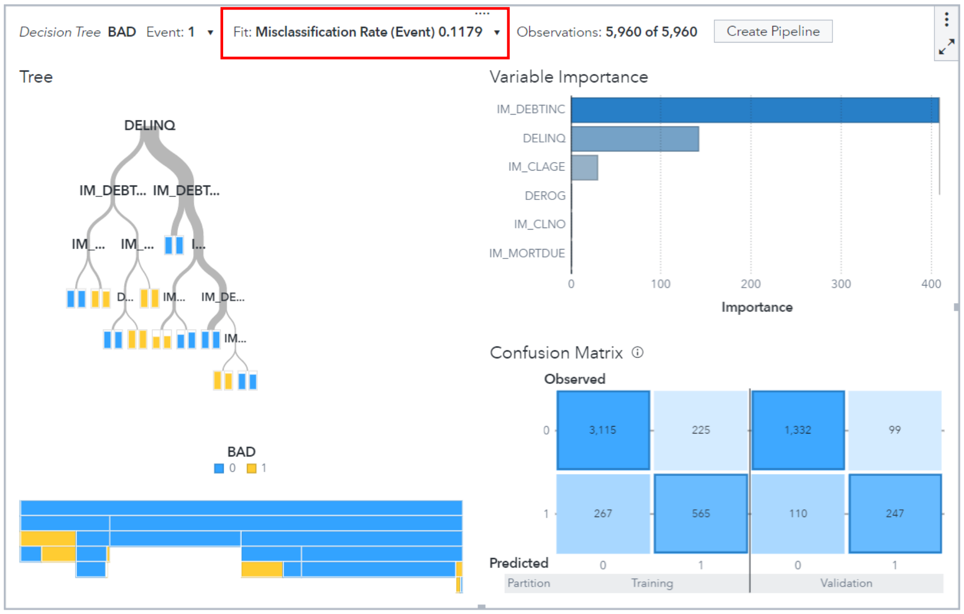
34. You can then use the newly created partition variable as, yes, your partition. And you're all caught up.

35. 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.

36. 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 **Misclassification Rate (Event)**:



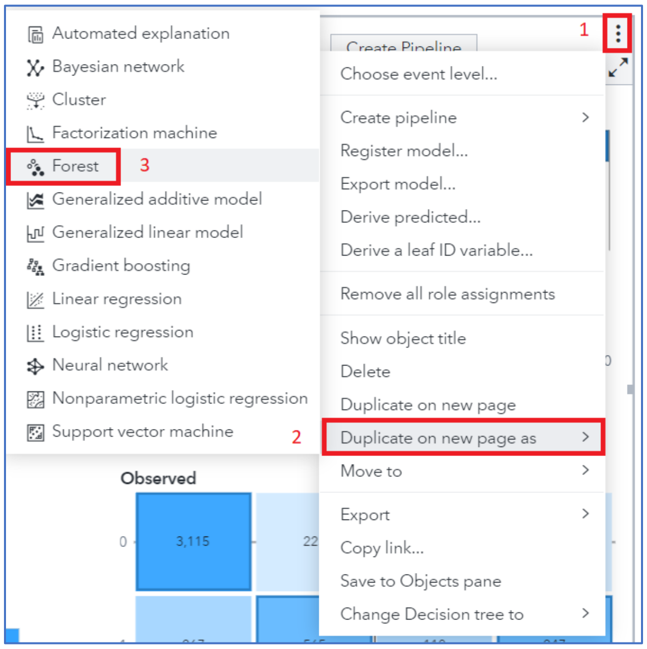
37. Observe and record the Misclassification Rate. Did you get a similar finding?



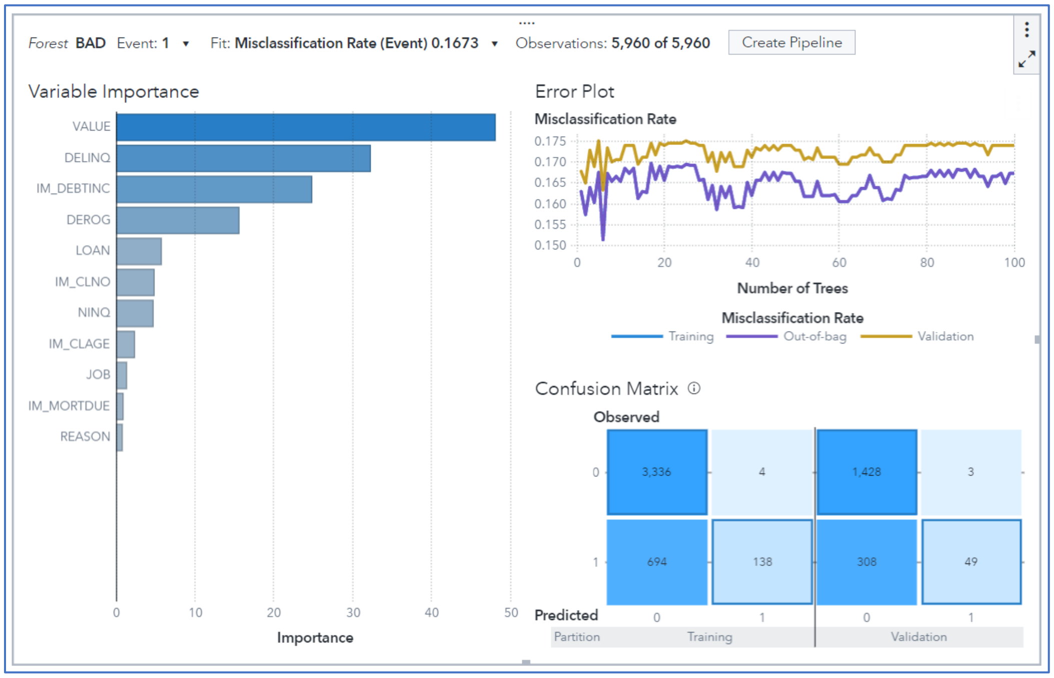
Don't sweat it if you didn't get the exact results reported above. It's likely because we're using different samples.

38. Now that was just a single model – based upon 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.

    From the **Object menu for the Decision tree**, hold down **ALT** and click **Duplicate on new Page as** > **Forest**. Follow through with these clicks:



39. 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, change the **Statistic to show** to **Misclassification Rate (Event)** and then examine the output:

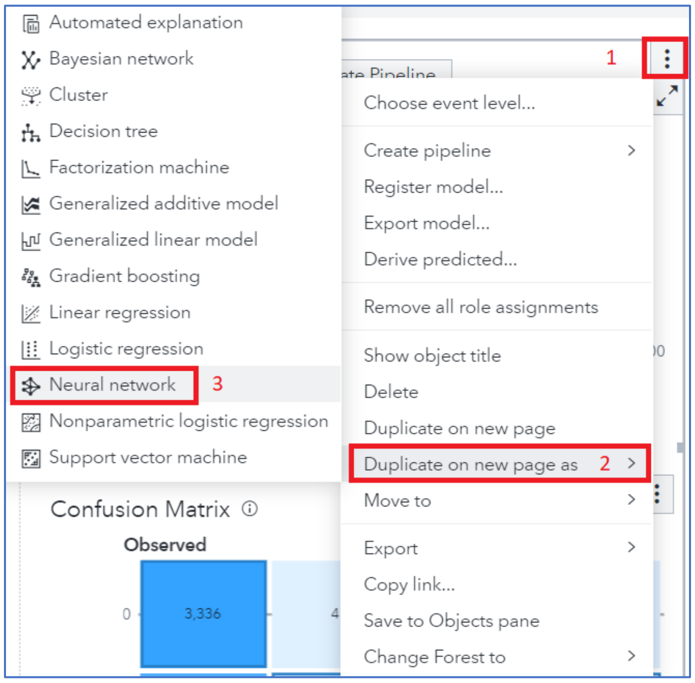

Forest BAD Event: 1 Fit: Misclassification Rate (Event) 0.1673 Observations: 5,960 of 5,960

Observed \ Predicted	Training 0	Training 1	Validation 0	Validation 1
0	3,336	4	1,428	3
1	694	138	308	49

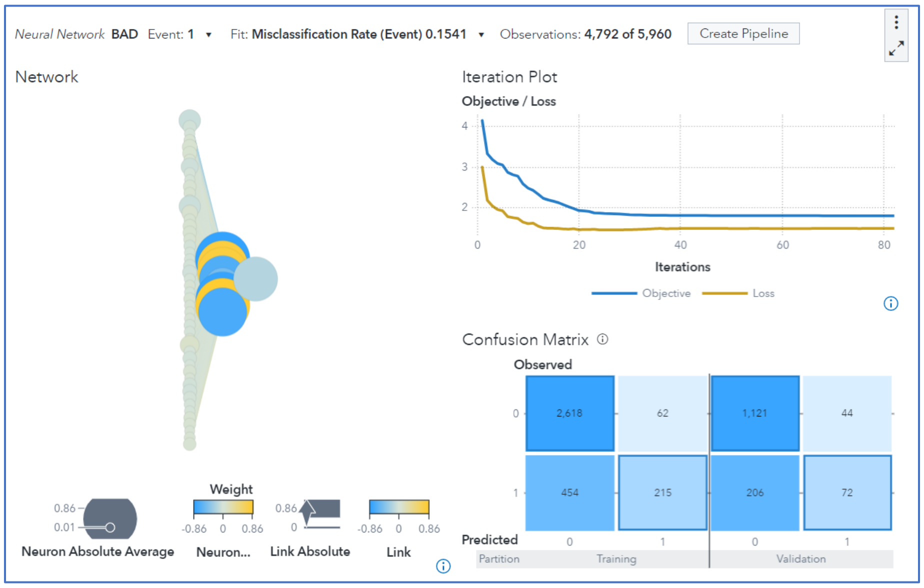
40. Based upon **Misclassification Rate (Event)** is this model better or worse than the decision tree?

…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.

41. Rename "Page 4" to "Forest" and let's run one more model – a Neural Network! From the **Object Menu for Forest**, again use **Alt + Duplicate on new page as > Neural Network** to create a third competitor model in a new page:

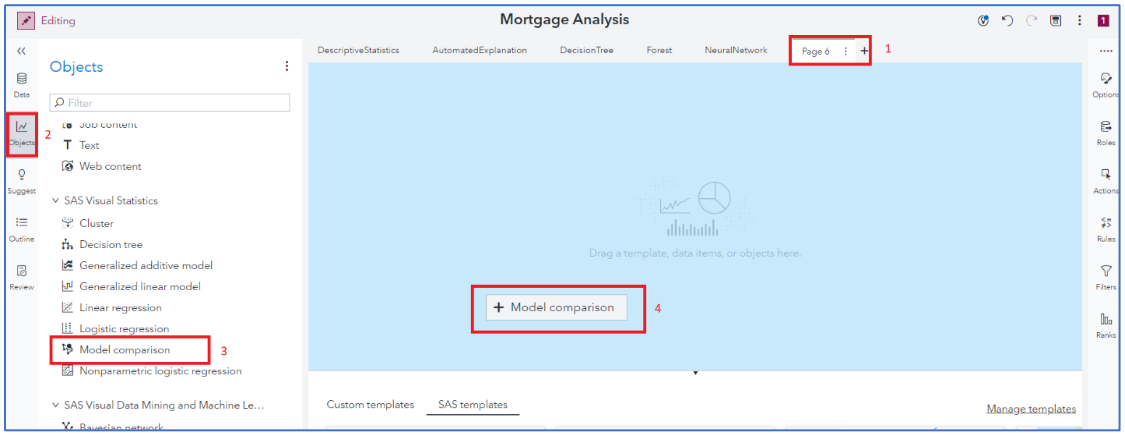


42. To make the models comparable, let's again change the **Statistic to show** to **Misclassification Rate (Event)**. Also, if you haven't already, let's change the name of the new page to "NeuralNetwork" and then examine the results:

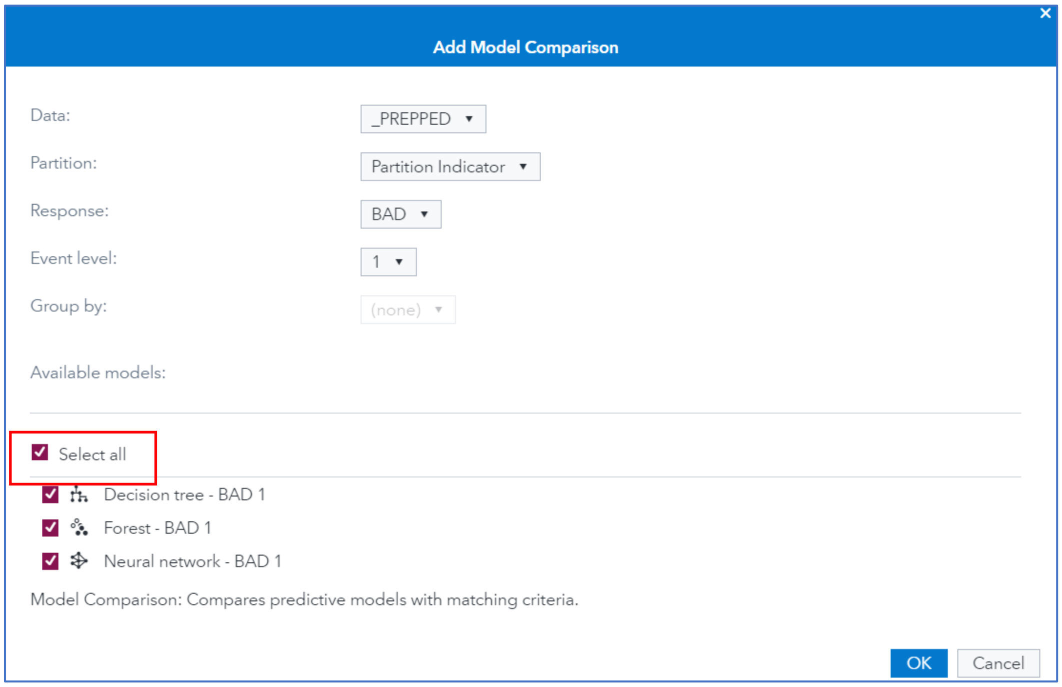


Note: it looks like we lost some observations – likely because neural networks use complete case analysis and we have missing data. But, this is a problem for another day.

43. In general, how did the neural network do? Well, again, not great. The misclassification rate is 0.1541, which is – again – higher than the simpler decision tree model.
44. 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.
45. To do so, create a new page, find the **Model Comparison** object and drag it to the new canvas:



46. The **Add Model Comparison** window appears. Use the default settings, **Select all** models, and then click **OK**:



Data: _PREPPED
Partition: Partition Indicator
Response: BAD
Event level: 1
Group by: (none)

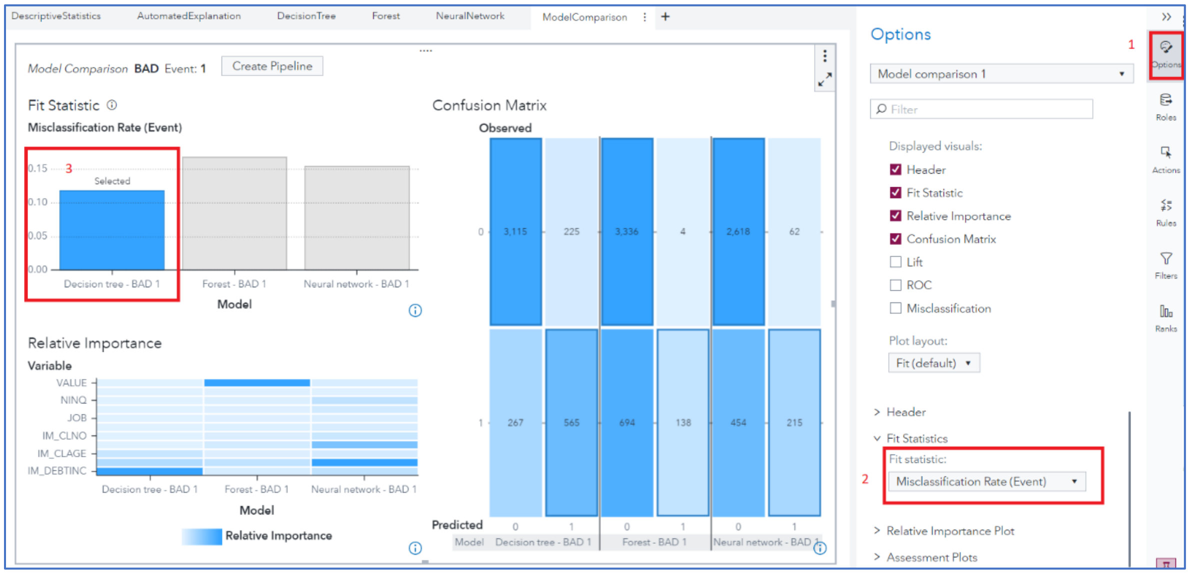
Available models:

☑ Select all
☑ Decision tree - BAD 1
☑ Forest - BAD 1
☑ Neural network - BAD 1

Model Comparison: Compares predictive models with matching criteria.

47. While data are being analyzed, rename “Page 6” to “ModelComparison”.

48. Finally – with the **Model Comparison** object selected – click **Options** and change the **Fit Statistic** to **Misclassification Rate (Event)**. Do you also get the Decision Tree as the champion?



Model Comparison BAD Event: 1

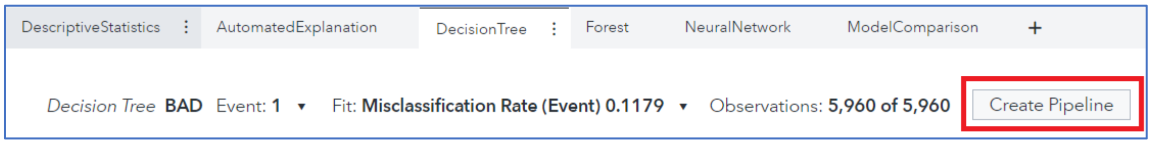
Fit Statistic
Misclassification Rate (Event)

Predicted	Decision tree - BAD 1 (0)	Decision tree - BAD 1 (1)	Forest - BAD 1 (0)	Forest - BAD 1 (1)	Neural network - BAD 1 (0)	Neural network - BAD 1 (1)
Observed 0	3,115	225	3,336	4	2,618	62
Observed 1	267	565	694	138	454	215

Options
Model comparison 1
Displayed visuals:
☑ Header
☑ Fit Statistic
☑ Relative Importance
☑ Confusion Matrix
☐ Lift
☐ ROC
☐ Misclassification
Plot layout: Fit (default)
> Header
⌄ Fit Statistics
Fit statistic: Misclassification Rate (Event)
> Relative Importance Plot
> Assessment Plots

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

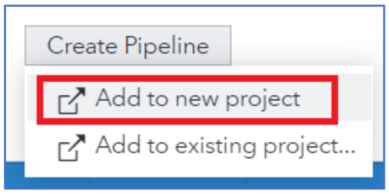
50. Our last Visual Analytics tasks is to export the champion model to **SAS Model Studio** – where we have many more no-code options to explore. To do so, return to the DecisionTree page. Find the following:



DescriptiveStatistics | AutomatedExplanation | DecisionTree | Forest | NeuralNetwork | ModelComparison | +

Decision Tree BAD Event: 1 ▾ Fit: Misclassification Rate (Event) 0.1179 ▾ Observations: 5,960 of 5,960 Create Pipeline

51. The **Create Pipeline** button has been hiding in plain sight all along! Pipelines are a fundamental tool in SAS Model Studio, which you will learn a lot about in the next section. Click that **Create Pipeline** button and then select **Add to new project**:



52. Your data and metadata definitions will then be transferred to SAS Model Studio, where the next adventure awaits!

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 ML models
- Incorporating open-source tools
- Comparing and assessing models

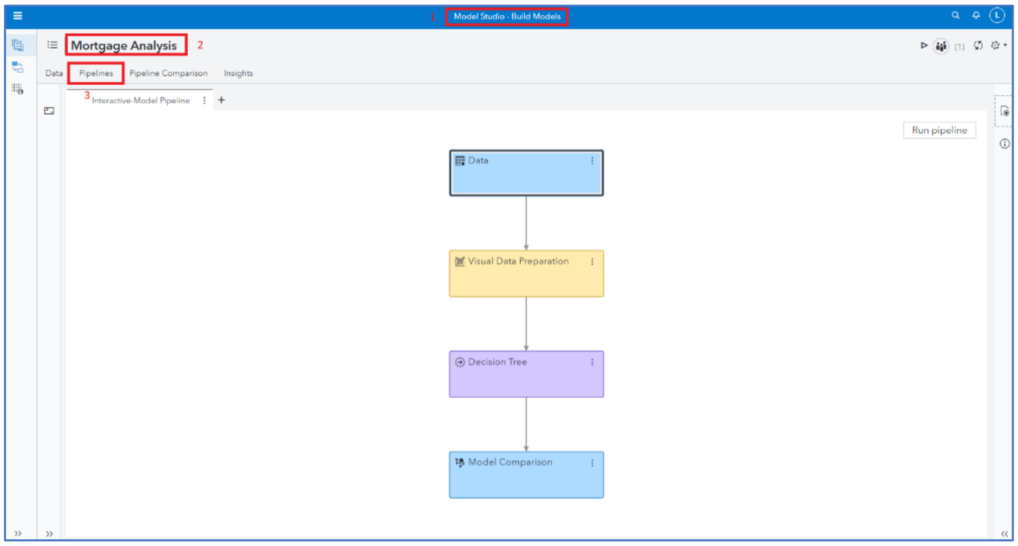
Estimated Time of Completion

This task is estimated to take about 45 minutes.

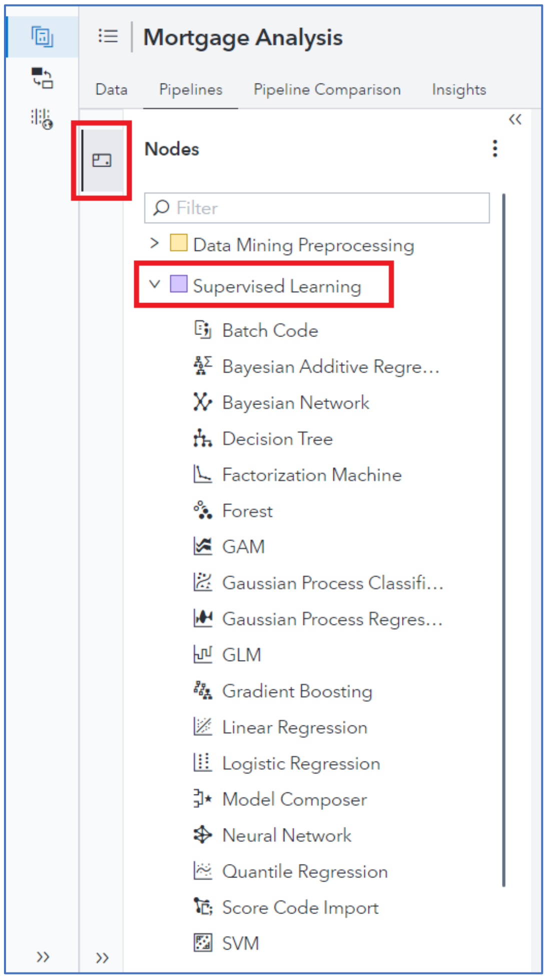
Task: Learn to love SAS Model Studio

SAS Visual Analytics is a great starting point for those new to SAS and analytics. But SAS VA simplifies many of the modeling options to make it a more user-friendly experience. In other words, there is a tradeoff 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.

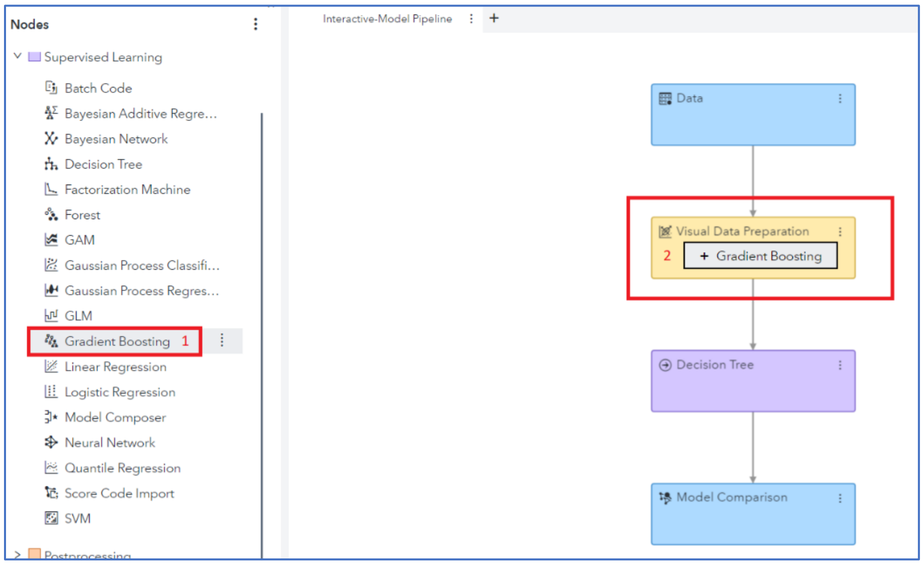
1. If you exported the pipeline properly in the last section, you should have a new SAS Model Studio Project named "Mortgage Analysis" – and the project should open to the **Interactive Model Pipeline**:



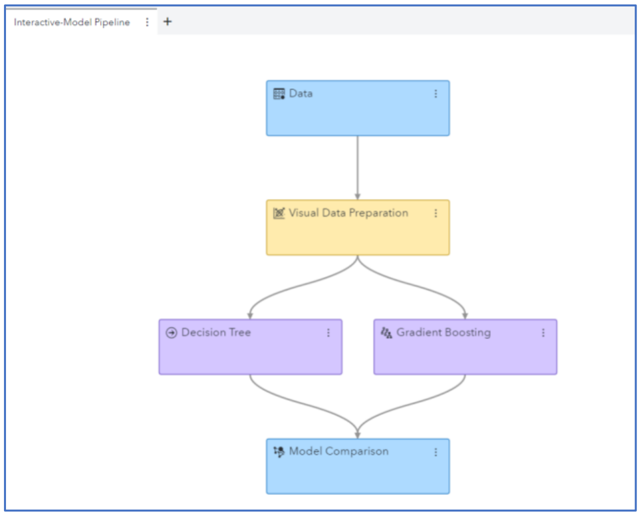
2. SAS Model Studio is pipeline centric and you can see our champion decision tree from our SAS Visual Analytics investigation has been converted to a pipeline.
3. Before running our first SAS Model Studio pipeline, let's explore how easy it is to add another model to the pipeline. Locate the **Nodes**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



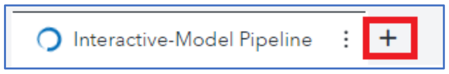
4. Now that's a lot of options! Let's see if the default **Gradient Boosting** model can defeat our Decision Tree. Grab the Gradient Boosting node and drop it on top of the **Visual Data Preparation** node:



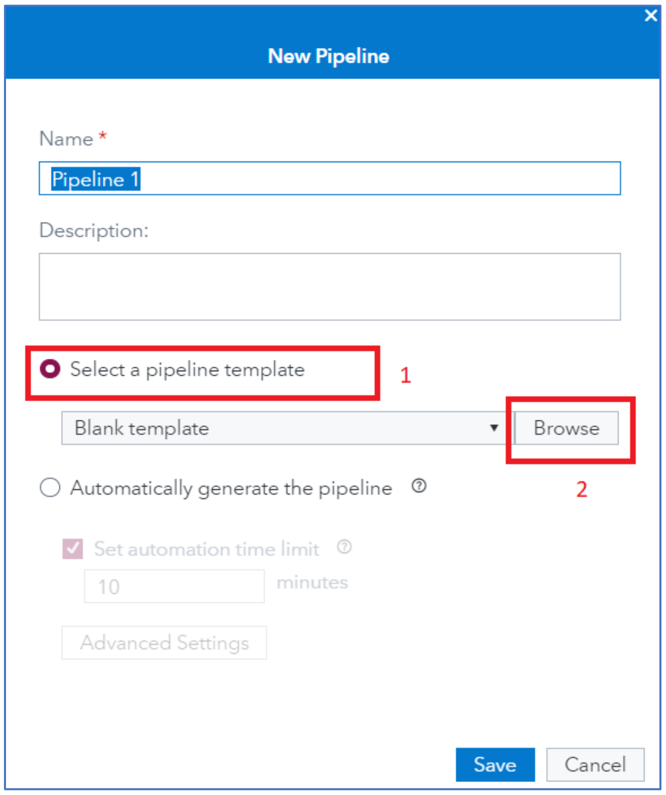
5. The new pipeline appears as:



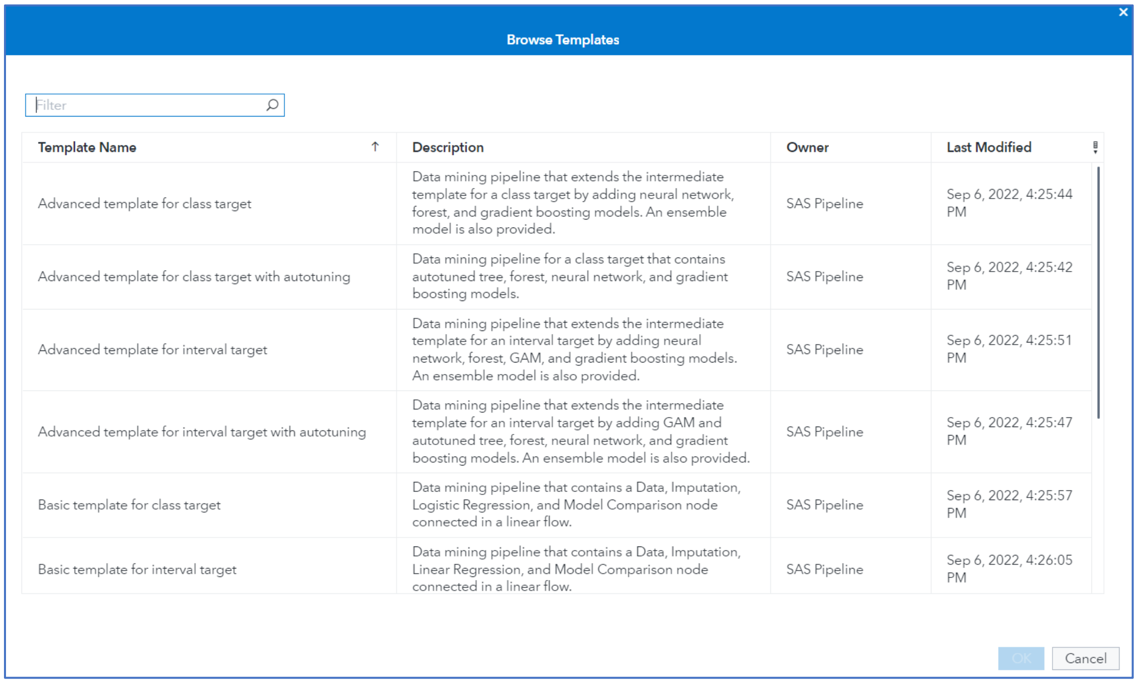
6. Submit the pipeline by selecting Run pipeline in the upper right-hand corner.
7. While this pipeline is running, I can show you one of my favorite features of SAS Model Studio. It's in the prebuilt pipelines, which makes running many models very easily. Create a new pipeline by clicking on the + next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



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

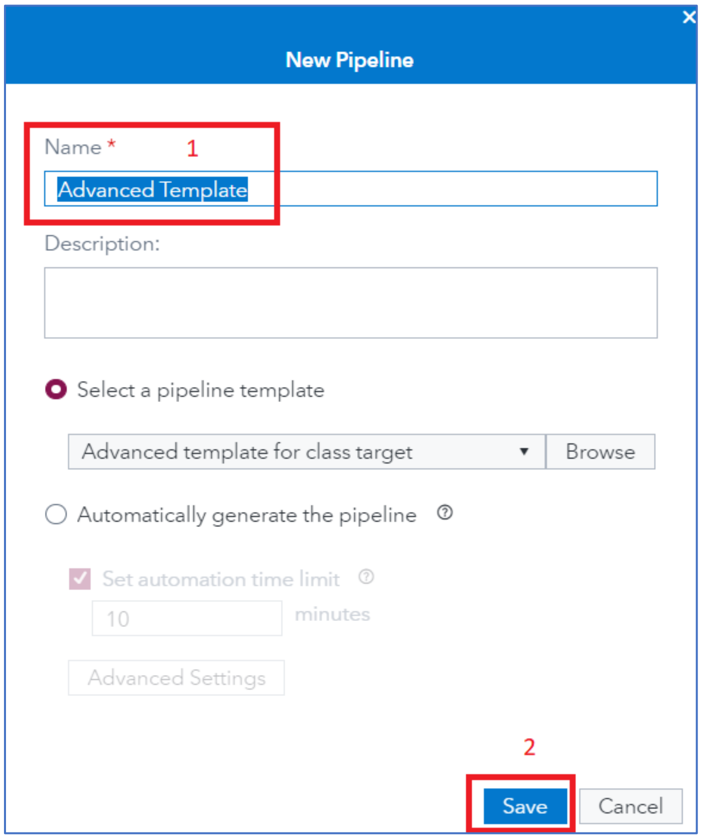


9. Check out all the pipeline templates available!



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	Sep 6, 2022, 4:25:44 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	Sep 6, 2022, 4:25:42 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	Sep 6, 2022, 4:25:51 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	Sep 6, 2022, 4:25:47 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	Sep 6, 2022, 4:25:57 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	Sep 6, 2022, 4:26:05 PM

10. Since we have a class target (i.e., “Yes” or “No” for a bad loan), let’s get crazy and select **Advanced template for class target** and 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 contains 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**:



New Pipeline

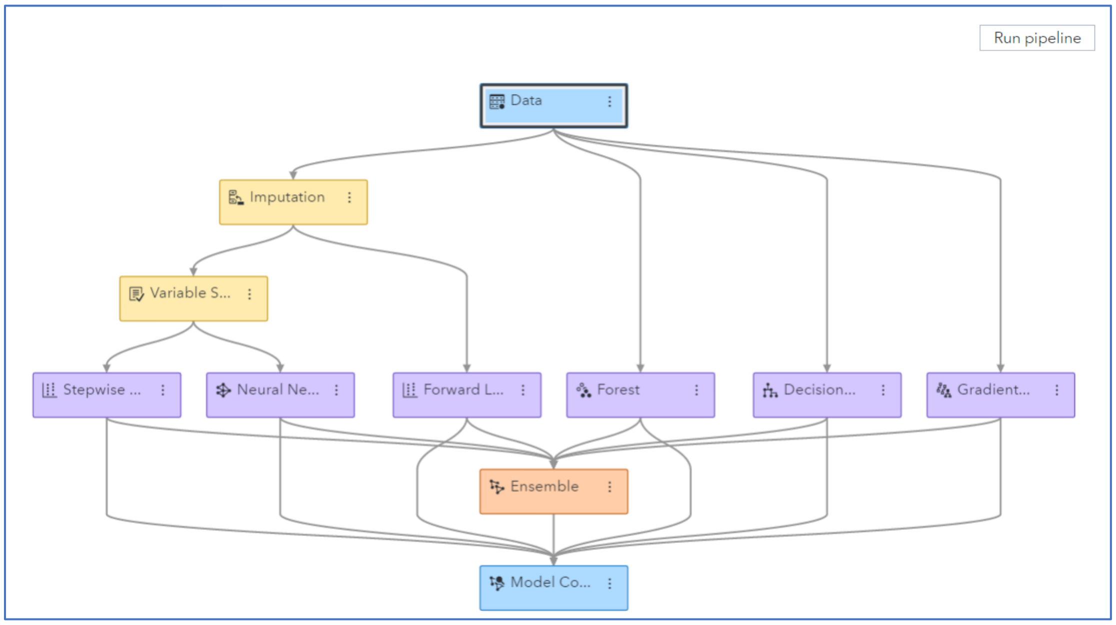
Name * Advanced Template

Description:

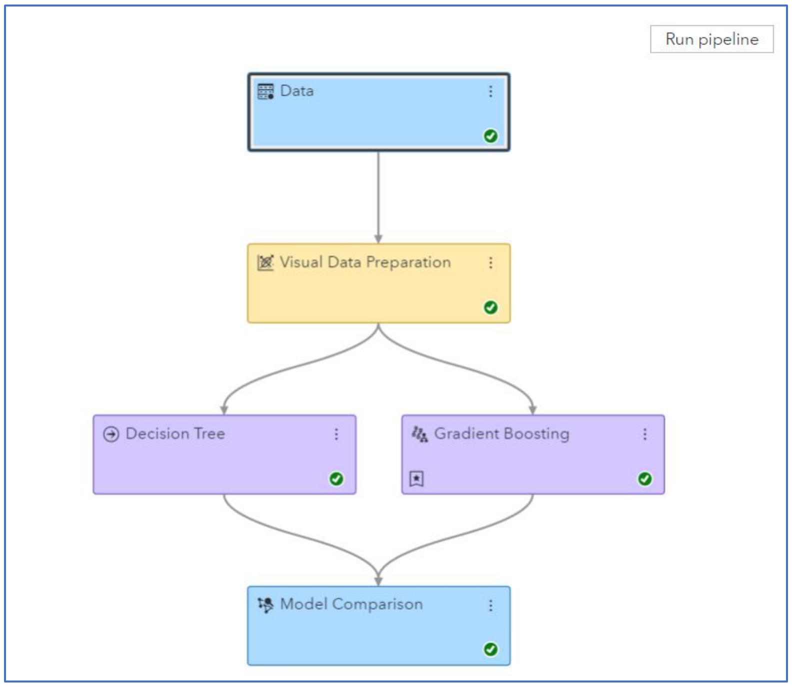
Select a pipeline template: Advanced template for class target

×
1
Browse
Automatically generate the pipeline
Set automation time limit
10
minutes
Advanced Settings
2
Save
Cancel

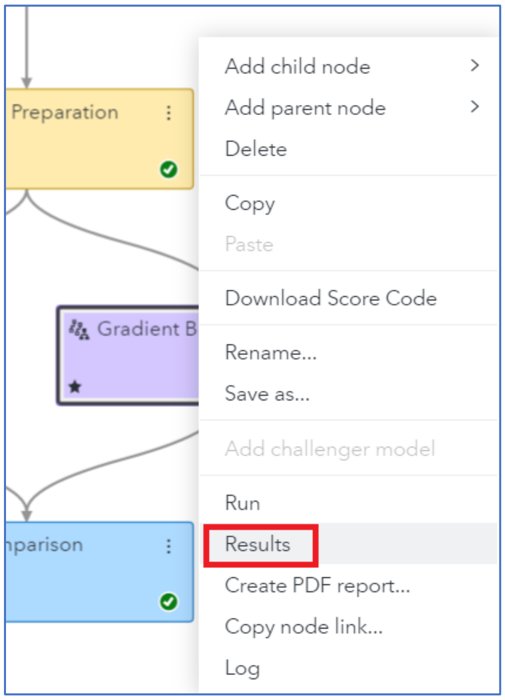
12. The new pipeline appears as follows:



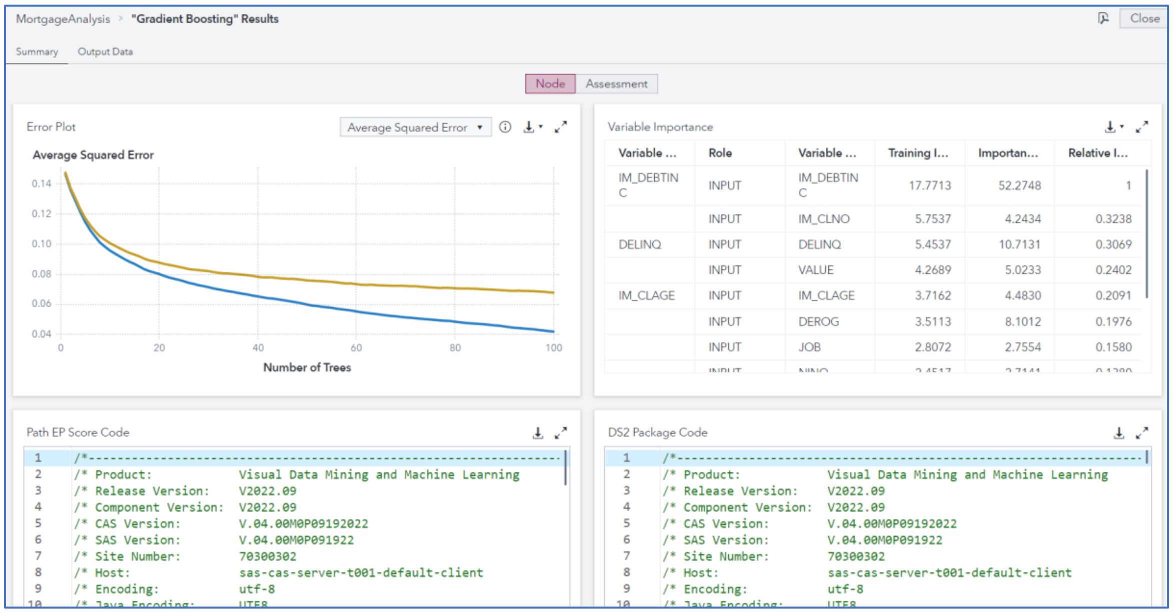
13. Again, look at all those modeling options! Yes, yes – that's seven for the price of one (click)! Since we're getting a bit pressed for time, let's simply run all the models – by clicking **Run pipeline** – and return to our original pipeline, **Interactive-Model Pipeline,** to examine those results.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



Variable Importance

Variable ...	Role	Variable ...	Training I...	Importan...	Relative I...
IM_DEBTINC	INPUT	IM_DEBTINC	17.7713	52.2748	1
	INPUT	IM_CLNO	5.7537	4.2434	0.3238
DELINQ	INPUT	DELINQ	5.4537	10.7131	0.3069
	INPUT	VALUE	4.2689	5.0233	0.2402
IM_CLAGE	INPUT	IM_CLAGE	3.7162	4.4830	0.2091
	INPUT	DEROG	3.5113	8.1012	0.1976
	INPUT	JOB	2.8072	2.7554	0.1580

Path EP Score Code

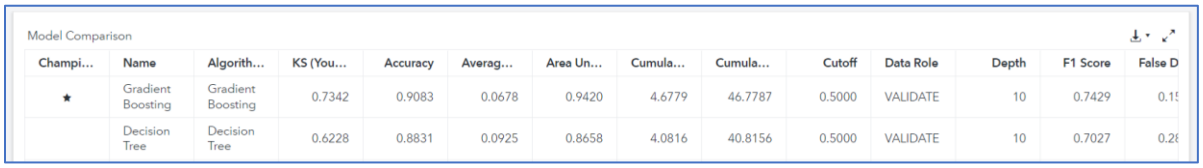
```
/*-----------------------------------------------------------------
/* Product:                Visual Data Mining and Machine Learning
/* Release Version:        V2022.09
/* Component Version:      V2022.09
/* CAS Version:            V.04.00M0P09192022
/* SAS Version:            V.04.00M0P091922
/* Site Number:            70300302
/* Host:                   sas-cas-server-t001-default-client
/* Encoding:               utf-8
/* Java Encoding:          UTF8
```

DS2 Package Code

```
/*-----------------------------------------------------------------
/* Product:                Visual Data Mining and Machine Learning
/* Release Version:        V2022.09
/* Component Version:      V2022.09
/* CAS Version:            V.04.00M0P09192022
/* SAS Version:            V.04.00M0P091922
/* Site Number:            70300302
/* Host:                   sas-cas-server-t001-default-client
/* Encoding:               utf-8
/* Java Encoding:          UTF8
```

17. Close **“GradientBoosting”Results** when you’ve had your fill.

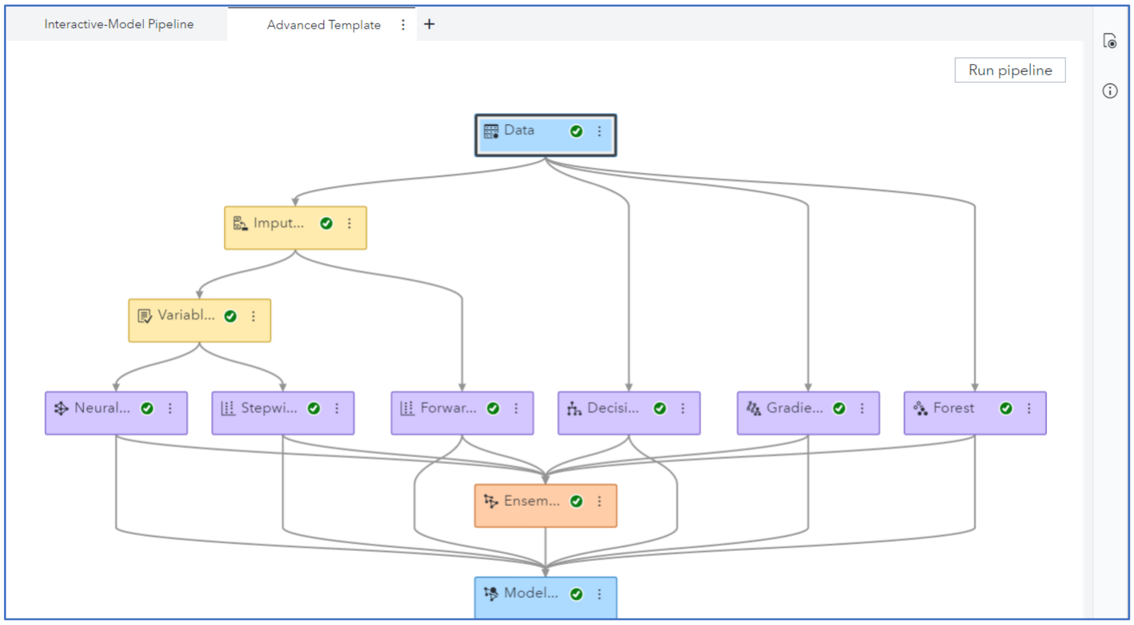
18. Next, select the Model Comparison node, right-click and select **Results**. Results will allow you to directly compare the two models in our pipeline:



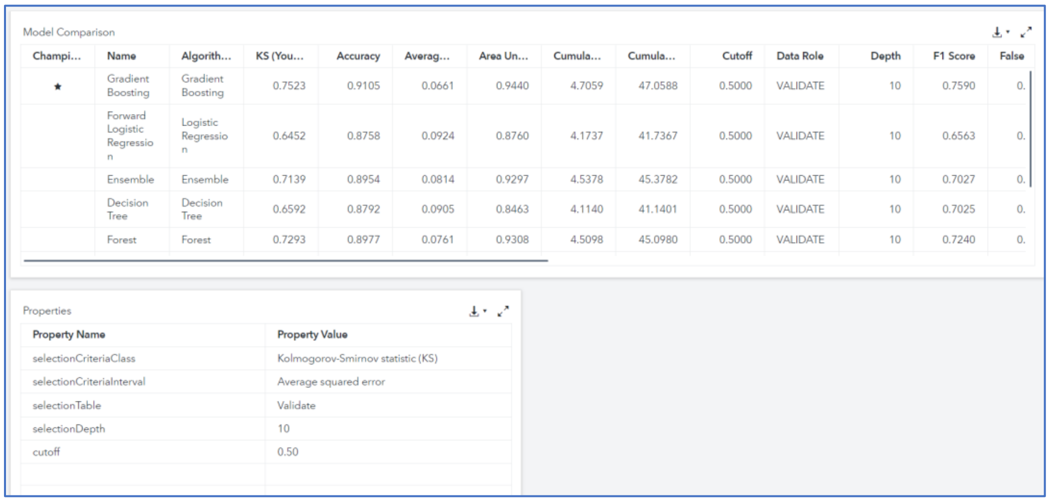
Model Comparison

Champi...	Name	Algorith...	KS (You...	Accuracy	Averag...	Area Un...	Cumula...	Cumula...	Cutoff	Data Role	Depth	F1 Score	False D
★	Gradient Boosting	Gradient Boosting	0.7342	0.9083	0.0678	0.9420	4.6779	46.7787	0.5000	VALIDATE	10	0.7429	0.1
	Decision Tree	Decision Tree	0.6228	0.8831	0.0925	0.8658	4.0816	40.8156	0.5000	VALIDATE	10	0.7027	0.2

19. We can see that the Gradient Boosting model is selected as champion – because it has the highest KS(Youden) statistic. If you scroll to the right, you can see the Gradient Boosting model has a lower misclassification rate. 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 VA and SAS Model Studio *and* the default gradient boosting model used in SAS VA. 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 checkmarks:



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



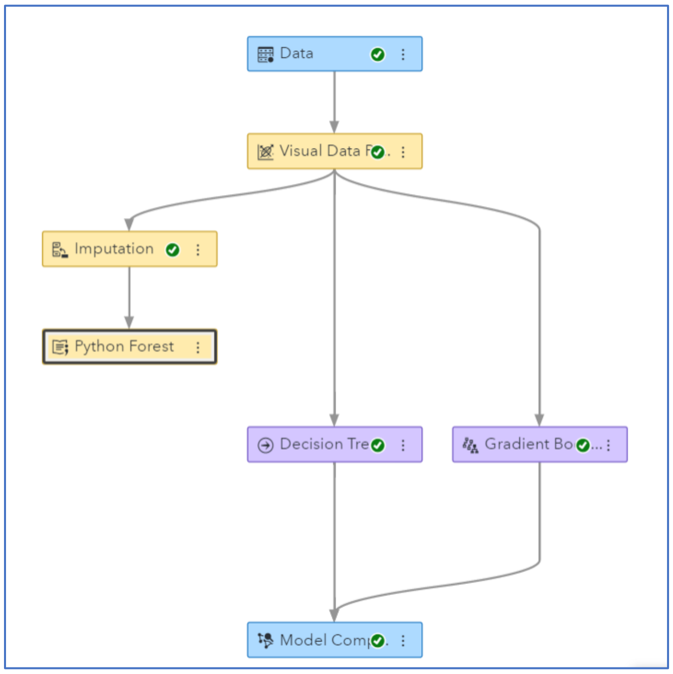
Model Comparison

Champi...	Name	Algorith...	KS (You...	Accuracy	Averag...	Area Un...	Cumula...	Cumula...	Cutoff	Data Role	Depth	F1 Score	False
★	Gradient Boosting	Gradient Boosting	0.7523	0.9105	0.0661	0.9440	4.7059	47.0588	0.5000	VALIDATE	10	0.7590	0.
	Forward Logistic Regressio n	Logistic Regressio n	0.6452	0.8758	0.0924	0.8760	4.1737	41.7367	0.5000	VALIDATE	10	0.6563	0.
	Ensemble	Ensemble	0.7139	0.8954	0.0814	0.9297	4.5378	45.3782	0.5000	VALIDATE	10	0.7027	0.
	Decision Tree	Decision Tree	0.6592	0.8792	0.0905	0.8463	4.1140	41.1401	0.5000	VALIDATE	10	0.7025	0.
	Forest	Forest	0.7293	0.8977	0.0761	0.9308	4.5098	45.0980	0.5000	VALIDATE	10	0.7240	0.

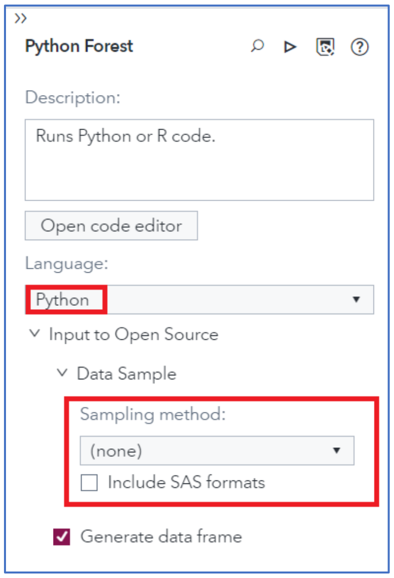
Properties

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? Gradient Boosting is still the best model! If you scroll to the right, find the Misclassification Rate (Event) column and sort from lowest to highest, you will again see that the Gradient Boosting Model is selected.
24. One lingering thought that you may 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.
26. Since the Open-Source nodes will handle missing values differently than our traditional SAS nodes, let's add an Imputation node to the flow. Right-click on the **Visual Data Preparation** node and select **Add child node** ⇨ **Data Mining Preprocessing** ⇨ **Imputation**.
27. Next, right-click on that new **Imputation** node and select **Add child node** ⇨ **Miscellaneous** ⇨ **Open Source Code**.
28. Right-click the **Open Source Code** node and rename the node to **Python Forest**. Your pipeline should appear as:



29. Under **Node options** for our **Python Forest**, verify that the language is set to **Python**. It should be Python by default…
30. Expand the **Data Sample** properties. Clear the check box for **Include SAS formats**.
    a. 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 DOLLAR*w.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)**. The cumulative changes:


Python Forest

Description:
Runs Python or R code.

Open code editor

Language: Python

Input to Open Source

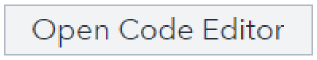
Data Sample

Sampling method: (none)

☐ Include SAS formats

☑ Generate data frame

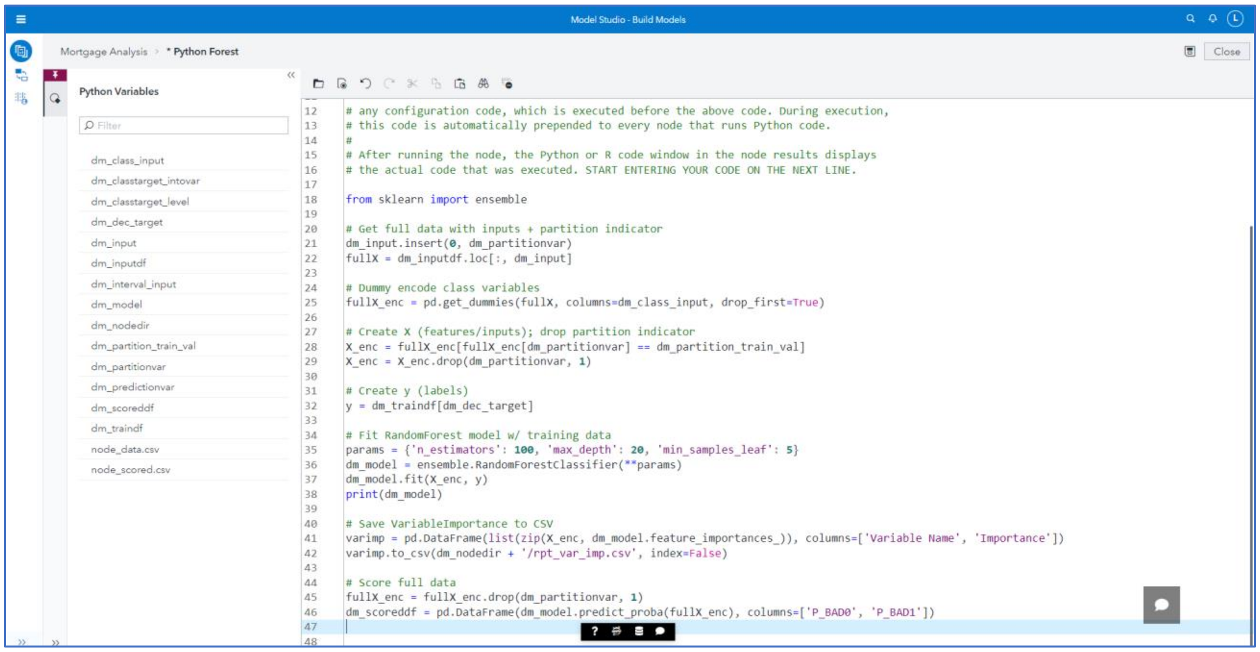
32. Now, click the **Open code editor** button in the Node Options pane:



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

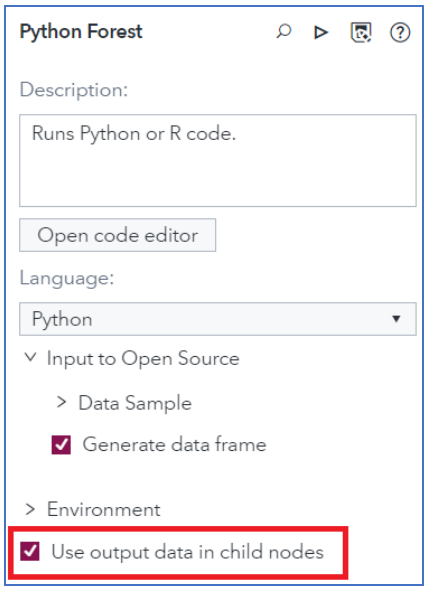
```
from sklearn import ensemble

# 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, 1)

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

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

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

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

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



```python
# 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_traindf[dm_dec_target]

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

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

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

35. A few notes about the Python code – while saving more of the gory details for another time:
    a. 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.
    b. The code that needs to be changed for different data sets is the last line – i.e., naming your predictions with the **P_ + "*target*"** naming convention.
    c. It's important to note that we are just modeling the data here. Currently, there is not a way to do 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**:



Python Forest

Description:

Runs Python or R code.

Language: Python

Input to Open Source

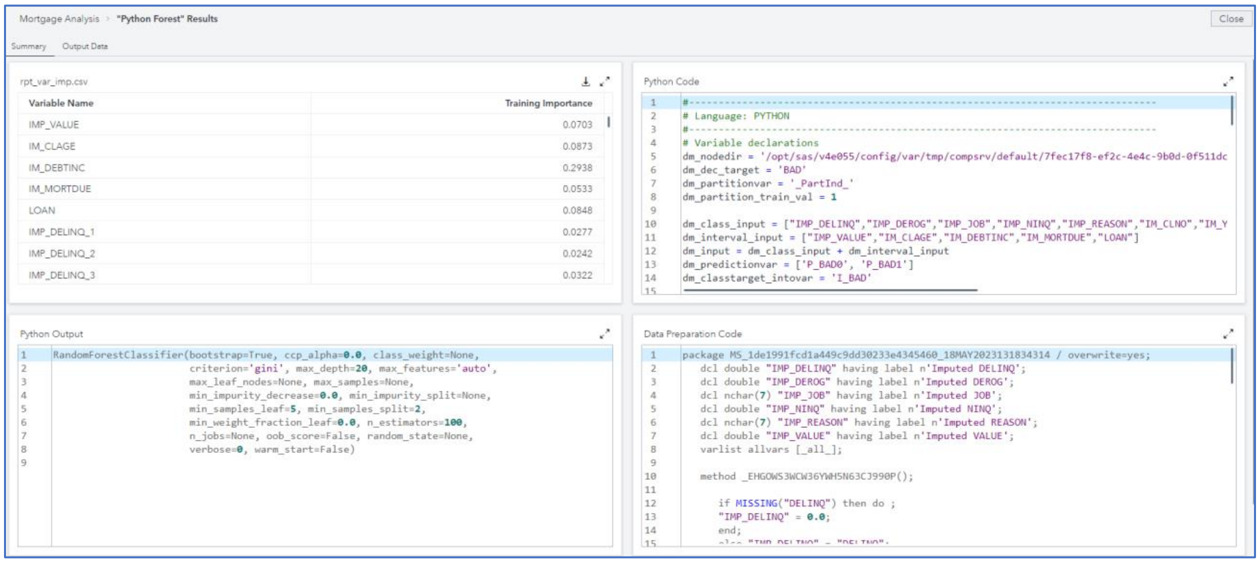
Data Sample

☑ Generate data frame

Environment

☑ Use output data in child nodes

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 on the node and select **Run**.
40. Open the **Results** of the Python Forest node to examine output.



rpt_var_imp.csv

Variable Name	Training Importance
IMP_VALUE	0.0703
IM_CLAGE	0.0873
IM_DEBTINC	0.2938
IM_MORTDUE	0.0533
LOAN	0.0848
IMP_DELINQ_1	0.0277
IMP_DELINQ_2	0.0242
IMP_DELINQ_3	0.0322

Python Code

```
#--------------------------------------------------------------------------------
# Language: PYTHON
#--------------------------------------------------------------------------------
# Variable declarations
dm_nodedir = '/opt/sas/v4e055/config/var/tmp/compsrv/default/7fec17f8-ef2c-4e4c-9b0d-0f511dc
dm_dec_target = 'BAD'
dm_partitionvar = '_PartInd_'
dm_partition_train_val = 1

dm_class_input = ["IMP_DELINQ","IMP_DEROG","IMP_JOB","IMP_NINQ","IMP_REASON","IM_CLNO","IM_Y
dm_interval_input = ["IMP_VALUE","IM_CLAGE","IM_DEBTINC","IM_MORTDUE","LOAN"]
dm_input = dm_class_input + dm_interval_input
dm_predictionvar = ['P_BAD0', 'P_BAD1']
dm_classtarget_intovar = 'I_BAD'
```

Python Output

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=20, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=5, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

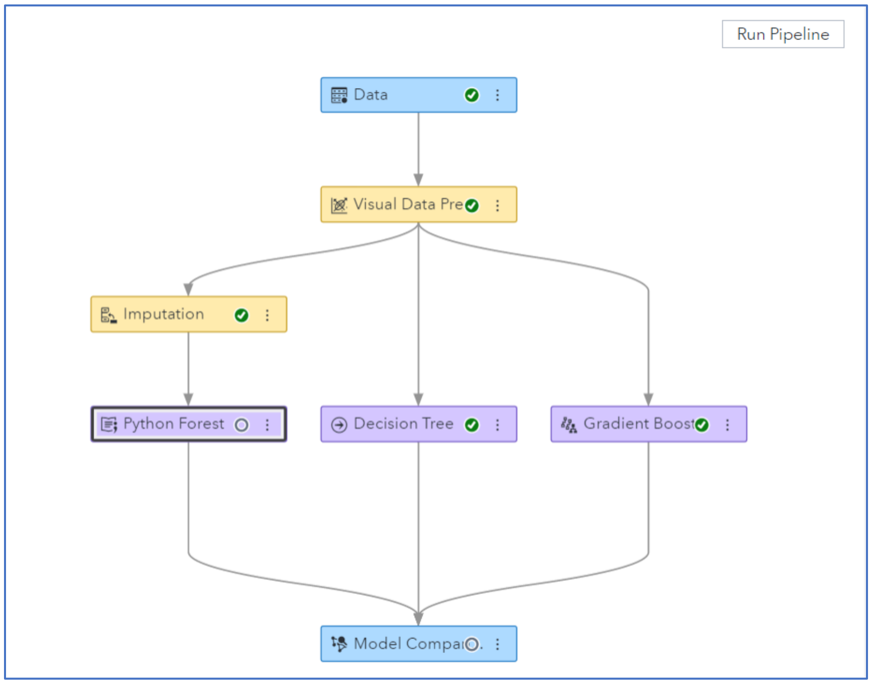
Data Preparation Code

```
package MS_1de1991fcd1a449c9dd30233e4345460_18MAY2023131834314 / overwrite=yes;
   dcl double "IMP_DELINQ" having label n'Imputed DELINQ';
   dcl double "IMP_DEROG" having label n'Imputed DEROG';
   dcl nchar(7) "IMP_JOB" having label n'Imputed JOB';
   dcl double "IMP_NINQ" having label n'Imputed NINQ';
   dcl nchar(7) "IMP_REASON" having label n'Imputed REASON';
   dcl double "IMP_VALUE" having label n'Imputed VALUE';
   varlist allvars [_all_];

   method _EHGOWS3WCW36YWH5N63CJ990P();

      if MISSING("DELINQ") then do ;
      "IMP_DELINQ" = 0.0;
      end;
```

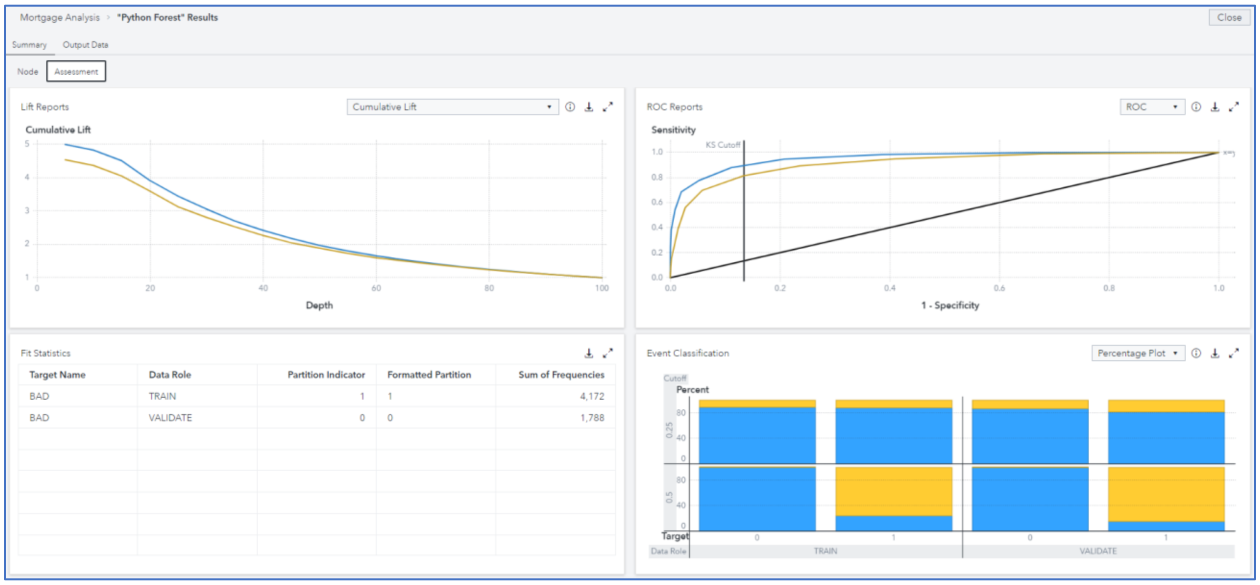
41. Examine the results for a bit, so you get a better understanding of model performance. Click **Close** when you're satiated.
42. And after you're done exploring, I have a thought-provoking question for you: *Why does the Open Source Code node lack assessment results even though it ran successfully?*

Well, for model assessment results, you need to move the nodes to the supervised learning lane.

43. To get our open-source node registered as a model, right-click the **Python Forest** node and select **Move** ⇨ **Supervised Learning**. Your new view should appear as follows:



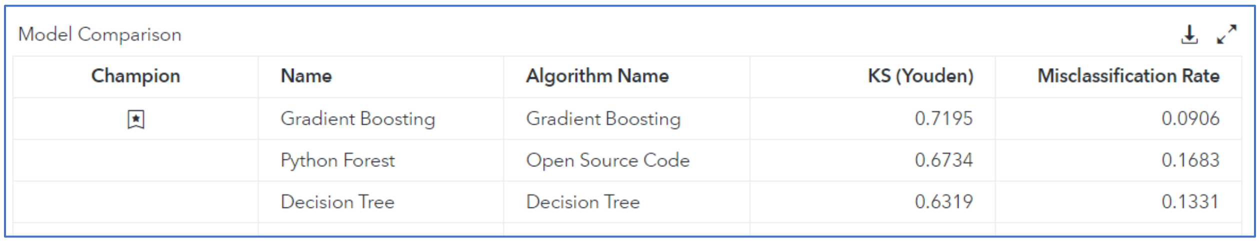
44. The color of *Python Forest* has changed to purple, showing that the node is now part of the Supervised Learning group. Notice also that the nodes need to be rerun, as the green check mark has disappeared.
45. Click the **Run Pipeline** button.
46. Open the **Results** of the Python Forest. Click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



Fit Statistics

Target Name	Data Role	Partition Indicator	Formatted Partition	Sum of Frequencies
BAD	TRAIN	1	1	4,172
BAD	VALIDATE	0	0	1,788

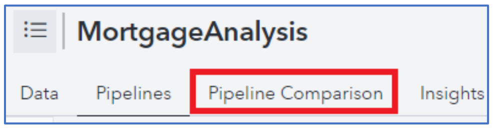
47. The usual assessment results are displayed. Explore until your heart is content and then close the results.
48. Open the **Results** of the **Model Comparison** node. Which model is our ultimate champion in the *Interactive-Model Pipeline*?



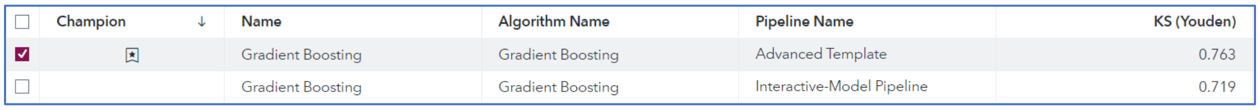
Model Comparison

Champion	Name	Algorithm Name	KS (Youden)	Misclassification Rate
★	Gradient Boosting	Gradient Boosting	0.7195	0.0906
	Python Forest	Open Source Code	0.6734	0.1683
	Decision Tree	Decision Tree	0.6319	0.1331

49. For our modeling, Gradient Boosting algorithm has the highest KS (Youden) statistic, followed closely by the Python Forest. So, it looks like both additions improve the fit over out default Decision Tree from SAS VA. Woot!
50. **Close** out of the Model Comparison Results window.
51. Let's do two more things before we wrap up. The first is to examine the Pipeline Comparison tab, found here:

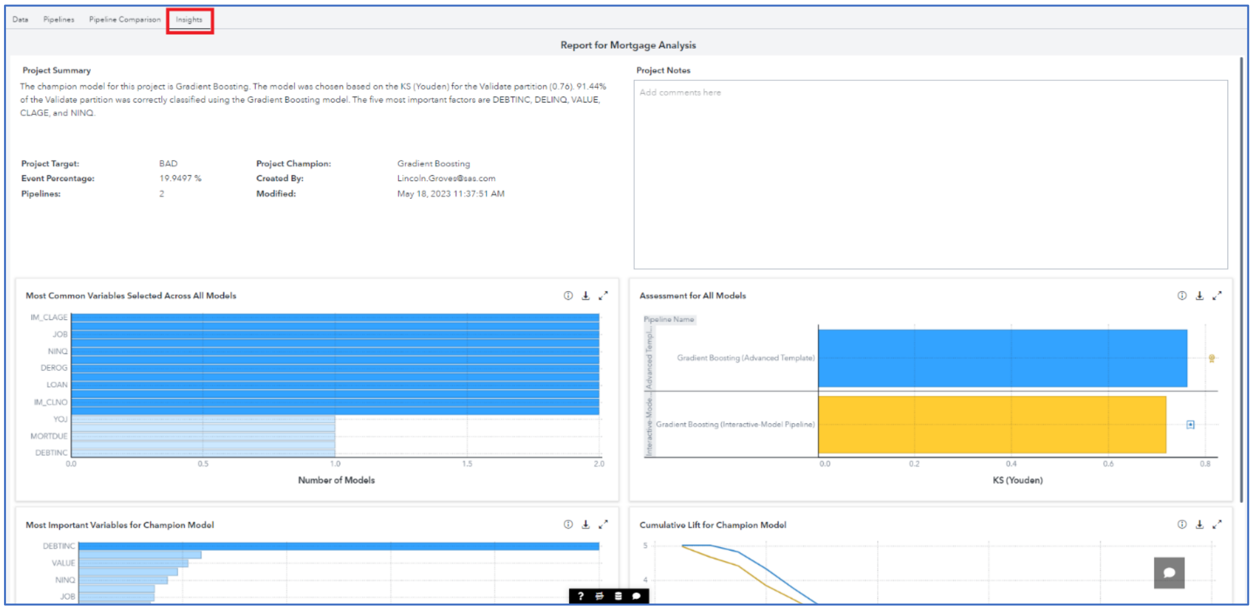


52. Rather than examining the champion of an individual pipeline, this will give us a champion of all the pipelines. We have only two potential models to choose from – but we could have many, many more! Select **Pipeline Comparison** tab – and examine the following:



	Champion	Name	Algorithm Name	Pipeline Name	KS (Youden)
✓	★	Gradient Boosting	Gradient Boosting	Advanced Template	0.763
☐		Gradient Boosting	Gradient Boosting	Interactive-Model Pipeline	0.719

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 may differ marginally due to different random samples being used.*
54. Finally, let's click on the **Insights** tab, found to the right of Pipeline Comparison:


Data
Pipelines
Pipeline Comparison
Insights
Most Common Variables Selected Across All Models
IM_CLAGE
JOB
NINQ
DEROG
LOAN
IM_CLNO
YOJ
MORTDUE
DEBTINC
Number of Models
Assessment for All Models
Pipeline Name
Gradient Boosting (Advanced Template)
Gradient Boosting (Interactive-Model Pipeline)
KS (Youden)
Most Important Variables for Champion Model
DEBTINC
VALUE
NINQ
JOB
Cumulative Lift for Champion Model

Report for Mortgage Analysis

Project Summary

The champion model for this project is Gradient Boosting. The model was chosen based on the KS (Youden) for the Validate partition (0.76). 91.44% of the Validate partition was correctly classified using the Gradient Boosting model. The five most important factors are DEBTINC, DELINQ, VALUE, CLAGE, and NINQ.

Project Target:	BAD	**Project Champion:**	Gradient Boosting
Event Percentage:	19.9497 %	**Created By:**	Lincoln.Groves@sas.com
Pipelines:	2	**Modified:**	May 18, 2023 11:37:51 AM

Project Notes

Add comments here

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

Steps to accessing SAS Viya for Learners for the first time:

1. Navigate to the SAS Viya for Learners webpage, https://www.sas.com/en_us/software/viya-for-learners.html
2. Click the "Access for Educators" or "Access for Students" button based on your role.
3. Log in with your SAS Profile that is linked to an academic affiliation. If you don't have a SAS Profile, click here to set one up.
4. Access the software by clicking on the **Launch SAS Viya for Learners** button.

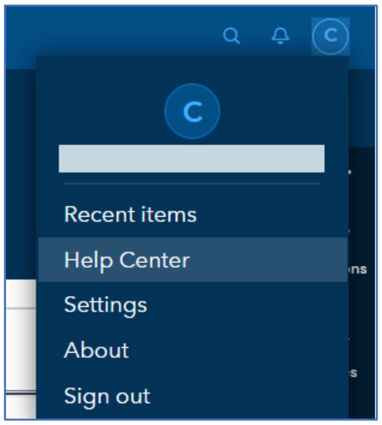
For future sessions with VFL, you can access VFL by visiting https://vle.sas.com/vfl.

Contact us at academic@sas.com if you have any questions about access.

Appendix B: Helpful Documentation

Here are two options for additional help and guidance:

1. **SAS Help Center** in SAS VFL
    a. It's been lurking in the upper-right-hand corner all long!



    b. Click the button to go to the **SAS Help Center** and the **SAS Documentation**.
2. **SAS Video Library**
    a. Prefer videos instead? We've got you covered here: https://video.sas.com/
    b. Check out the **How-to Tutorials** or simply **Search Videos**.

Appendix C: Recommended Learning

The SAS Global Academic Program offers free eLearning courses for students to learn SAS through the Student Skill Builder. The following eLearning courses and paths available are recommended to help with this activity:

- 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

Alternatively, the SAS Learning Subscription grants you access to an extensive library of SAS eLearning courses. Sign up for a free 7-day trial here!