Ontario Tech University | Seminar in SAS Machine Learning Tools

Framing, Part I: Learning Objectives and Purpose

- Expose participants to a broad range of machine learning tools
- Show how to produce advanced analytics with code and no-code options
- Provide useful follow-up resources

Framing, Part II: Brief Overview of SAS Academic Programs

- Our Mission
  - Helping to train the next generation of analytics talent
- How we support Academics
  - SAS Student Skill Builder
  - SAS Educator Portal
- Tools… +click here
  - Linking links
    - https://www.sas.com/en_us/learn/academic-programs.html
    - https://www.sas.com/en_us/software/viya-for-learners.html
    - https://www.sas.com/en_us/software/on-demand-for-academics.html
    - https://www.sas.com/en_us/learn/academic-programs/students.html
    - https://www.sas.com/en_us/learn/academic-programs/educators.html

Framing, Part III: Code versus No Code Options

- Broad software tour
  - OnDemand for Academics
    - SAS Studio
    - SAS Enterprise Miner (for a limited time only)
  - SAS Viya for Learners
    - SAS Studio
    - SAS Model Studio
    - SAS Visual Analytics
- What are the advantages of each approach?

Framing, Part IV: The Data

- We'll be using the classic ***Titanic*** data set in this analysis to explore survival rates after the sinking of the "unsinkable" ship in 1912
- Variables are:

Variable Name	Details
Age	Age
Cabin	Cabin
Embarked	Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)
Fare	Passenger Fare (British pound)
Name	Name
Parch	Number of Parents/Children Aboard
PassengerId	Passenger ID
Pclass	Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
Sex	Sex
SibSp	Number of Siblings/Spouses Aboard
Survived	Survival (0 = No; 1 = Yes)
Ticket	Ticket Number

  - More details about the Titanic data set can be found here.

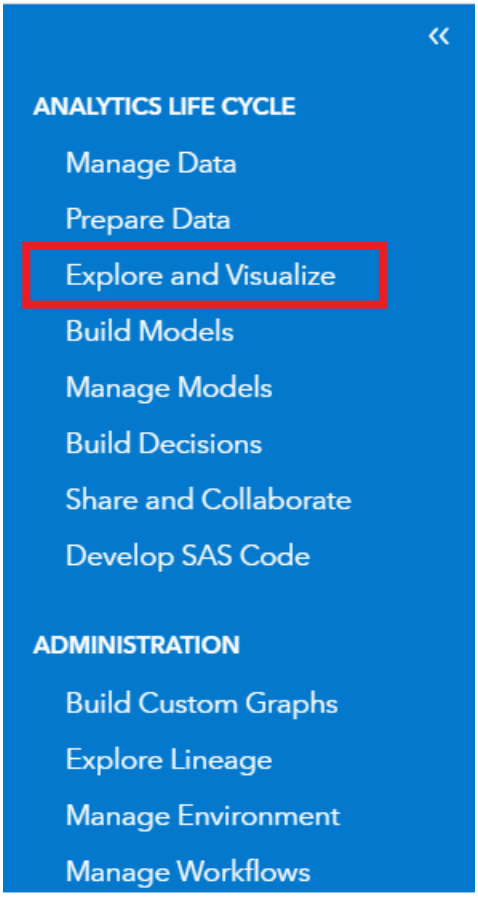
Framing, Part IV: Access Software

- Steps to accessing SAS Viya for Learners for the first time:
  - Navigate to the SAS Viya for Learners webpage, https://www.sas.com/en_us/software/viya-for-learners.html
  - Click the "Access for Educators" or "Access for Students" button based on your role.
  - 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.
  - Access the software by clicking on "Launch SAS Viya for Learners 3.5"
- 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.

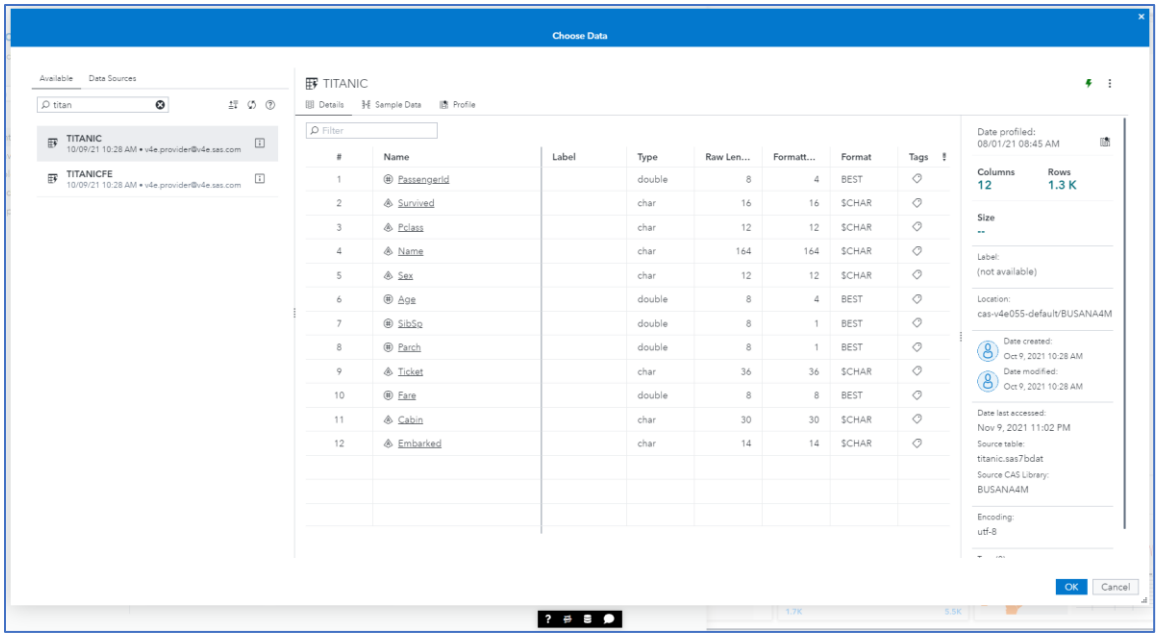
Part I: SAS Visual Analytics

- **Access SAS Visual Analytics**
  - How? Let the Hamburger become your friend!
    - From upper left, click

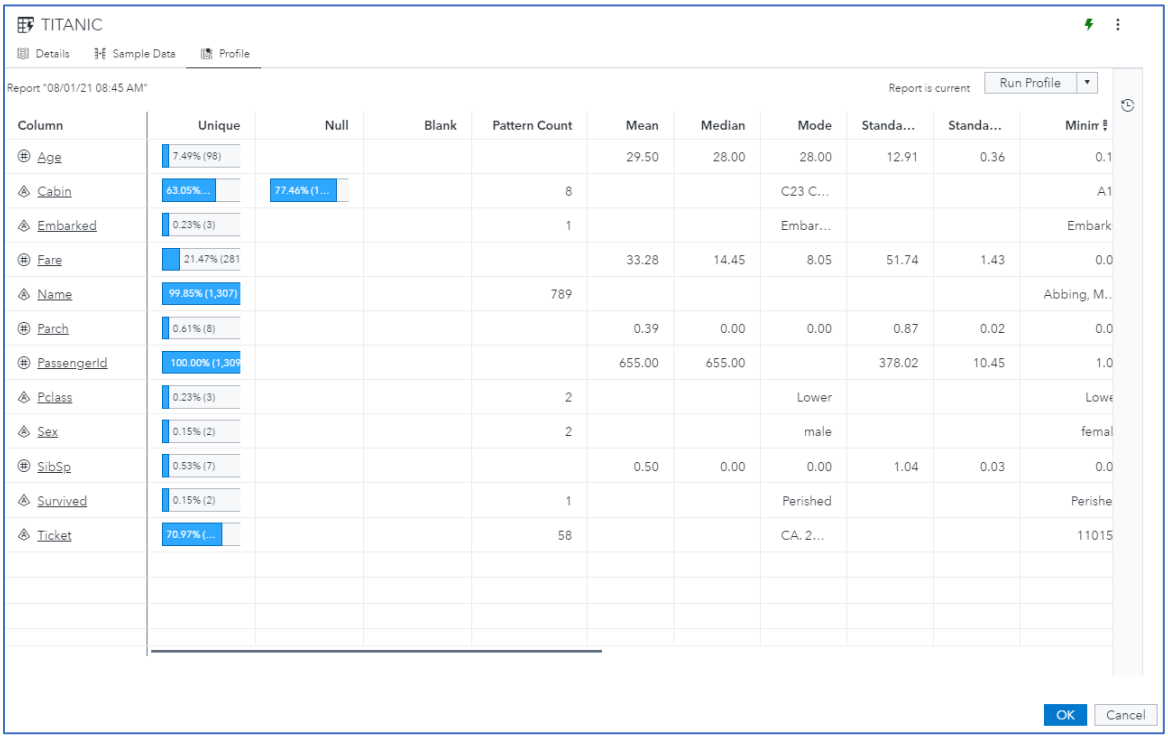
    - Then select Explore and Visualize from the options available:



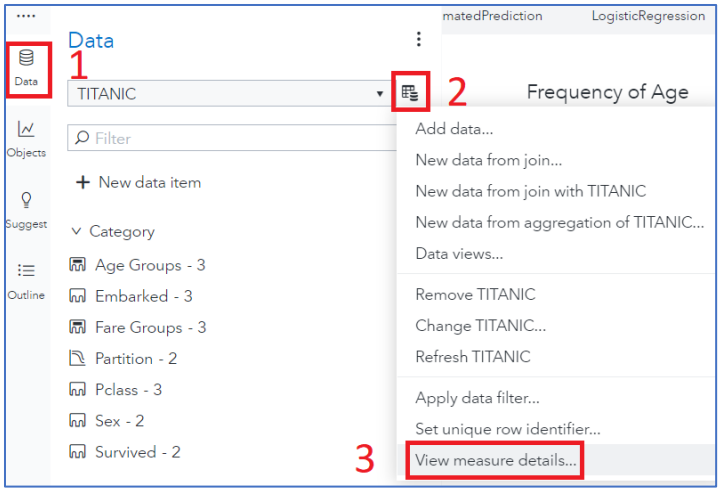
- **From here, *Start with Data***
  - Find **Titanic** in the available data sets in SAS Viya for Learners:



- In the **Choose Data** window, click on *TITANIC*. Next, examine the information about the data under the **Details**, **Sample Data**, and **Profile** tabs**.**
- Here is sample output from the **Profile** tab:



- **Let's further explore our data!**
  - We can do this via **View Measure Details**
    - Find the tool here:

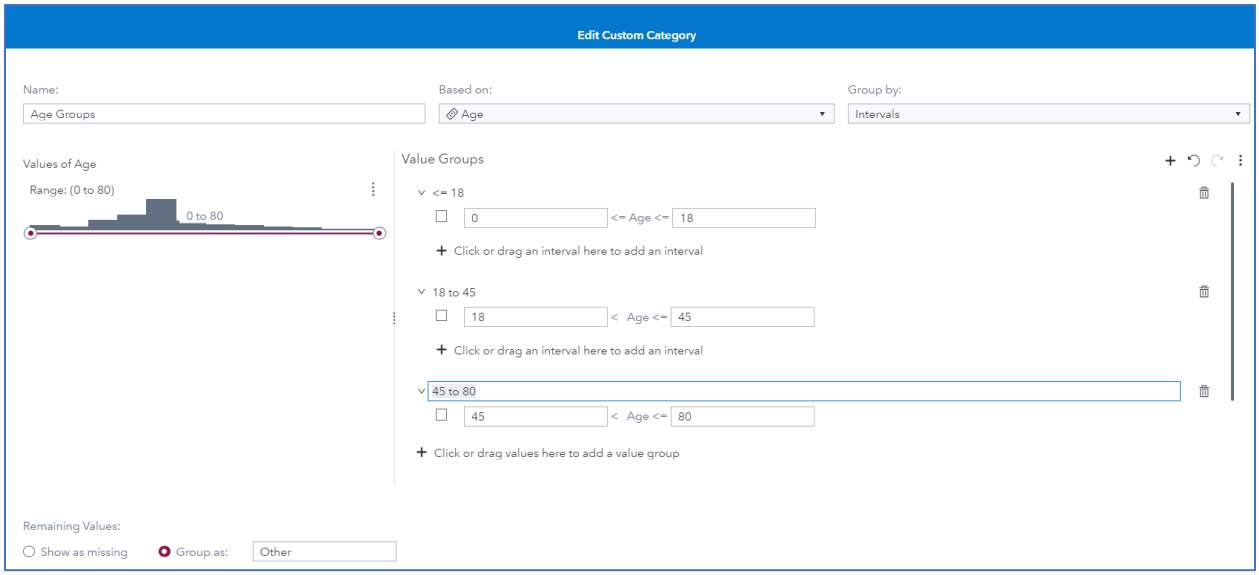


- Sample output:

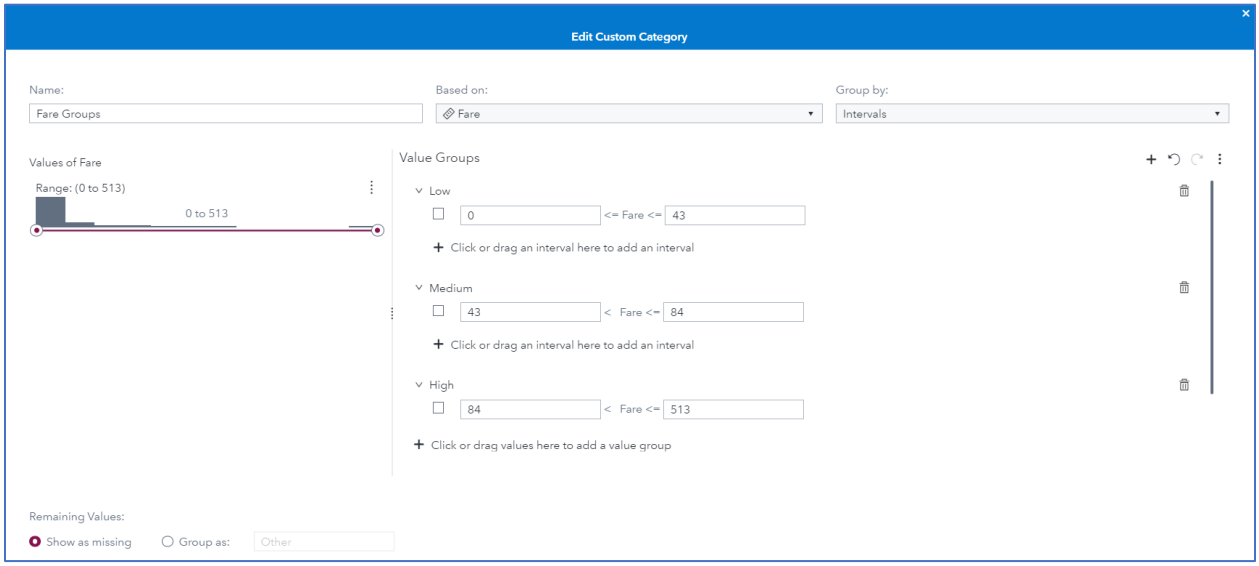
- Let's create a series of histograms in the default canvass to better understand the data
  - Label the page "DescriptiveStats" and replicate the following:

- **Helpful hints**
  - Frequency of Sex grouped by Survived
    - Grab sex and survived and then Auto Chart
    - Change **List table** to a **Bar chart** with the following settings:
      - Category = sex
      - Group = survived
      - Grouping style = Stacked
  - Frequency of Cabin
    - Change default Auto chart object to **Pie chart**

- **Now, let's create New Variables ➔ with + *New Data Item* ➔ *Custom Category***
  - Why do this?
    - Well, the relationship between Survived and variables such as Age or Ticket Fare is likely non-linear. By creating new custom categories, we can increase the accuracy and flexibility of our models.
  - Variable 1: Age Groups
    - Group 1 | 0 to 18
      - Label = "<= 18"
    - Group 2 | 18 to 45
      - Label = "18 to 45"
    - Group 3 | 45 to 80
      - Label = "45 to 80"
  - *Helpful hint before you get started:*
    - *Clicking on the end of the range line allows you to type in the exact value (rather than messing with the manual adjustment)*

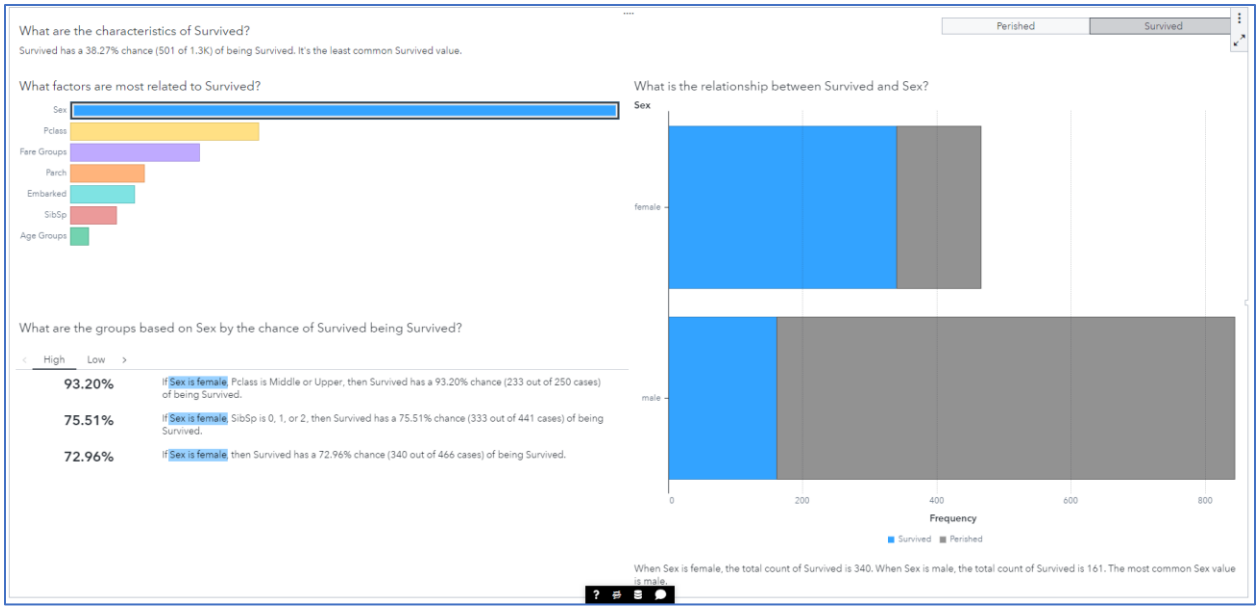


- Variable 2: Fare Groups
  - Group 1 | 0 to 43
    - Label = “Low”
  - Group 2 | 43 to 84
    - Label = “Medium”
  - Group 3 | 84 to 513
    - Label = “High”



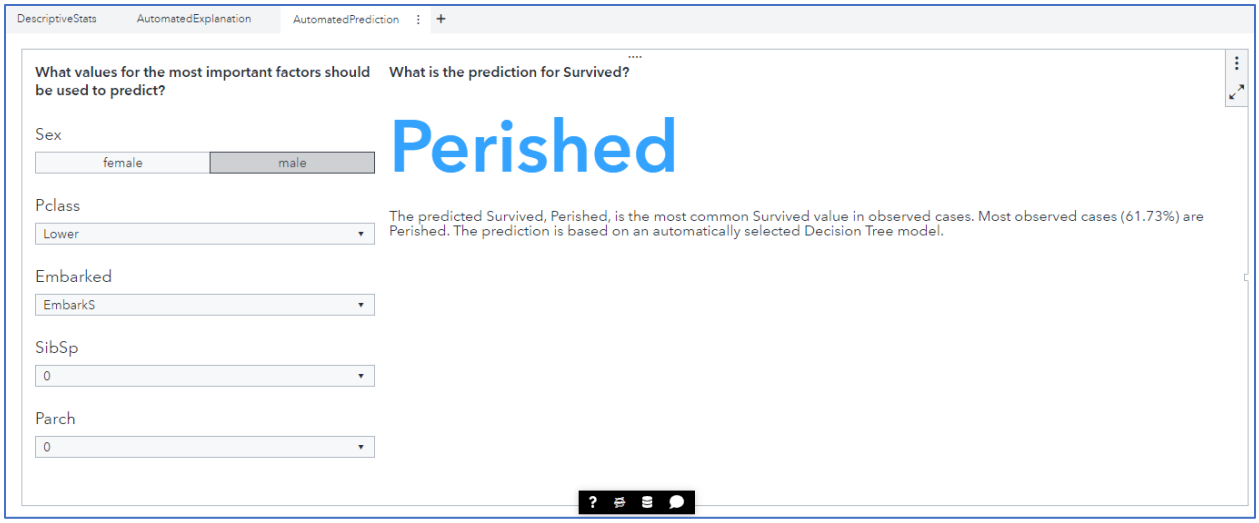
- Finally, add **Age Groups** and **Fare Groups** to ***DescriptiveStats***:
  - Sort variables as needed

- **Let's change the metadata**
  - What is metadata?
    - Simply data about the data
  - Why change it?
    - Changing the metadata helps SAS produce more useful output in our automated analyses
  - So, let's hide the following variables:
    - Cabin
    - Name
    - Ticket
    - Age
    - Fare
    - PassengerID
- **Let's see what insights we can gleam from Automated Explanation**
  - Create a new page to examine **Survived** versus **Perished**
    - Label this page "Automated Explanation"
  - Find *Automated Explanation* under *Analytics*. Drag the Object to the canvass.
    - Choose *Survived* as the *Response*.
  - Explore the output, including all the filters
    - Do you see any interesting relationships?
  - Sample output:



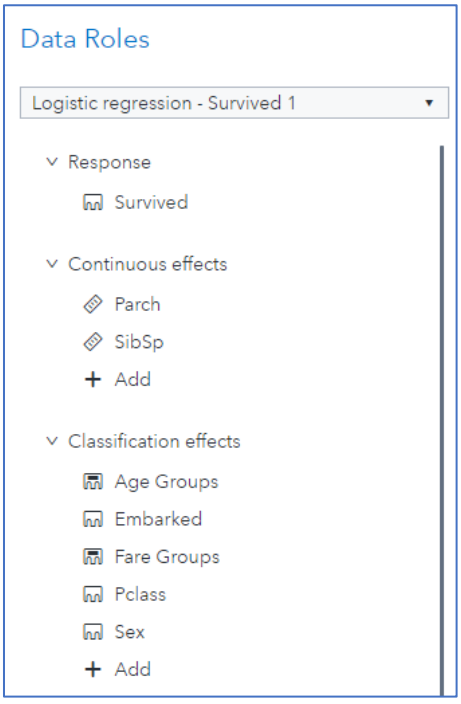
- Important Question:
  - What % of Titanic passengers survived?
    - Survival occurred 38.27% of the time
  - Why is this number important?
    - We'll use it shortly as our prediction cut-off – i.e., our probability decision threshold

- **Now on to Automated Prediction**
  - The backstory:
    - Rather than examining the bivariate relationships between Survived and individual variables – as with the Automated Explanation – let's look at all variables simultaneously via an A.I. tool called "Automated Prediction".
  - Create a new page called "Automated Prediction"
    - Find the *Automated Prediction* object under *Analytics* and add it to the canvass
      - Assign *Survived* as the **Response**
    - Try a sample prediction with the following settings:
      - Sex = male
      - Pclass = Lower
      - Embarked = EmbarkS
      - Parch = 0
      - SibSp = 0
    - What is the outcome?



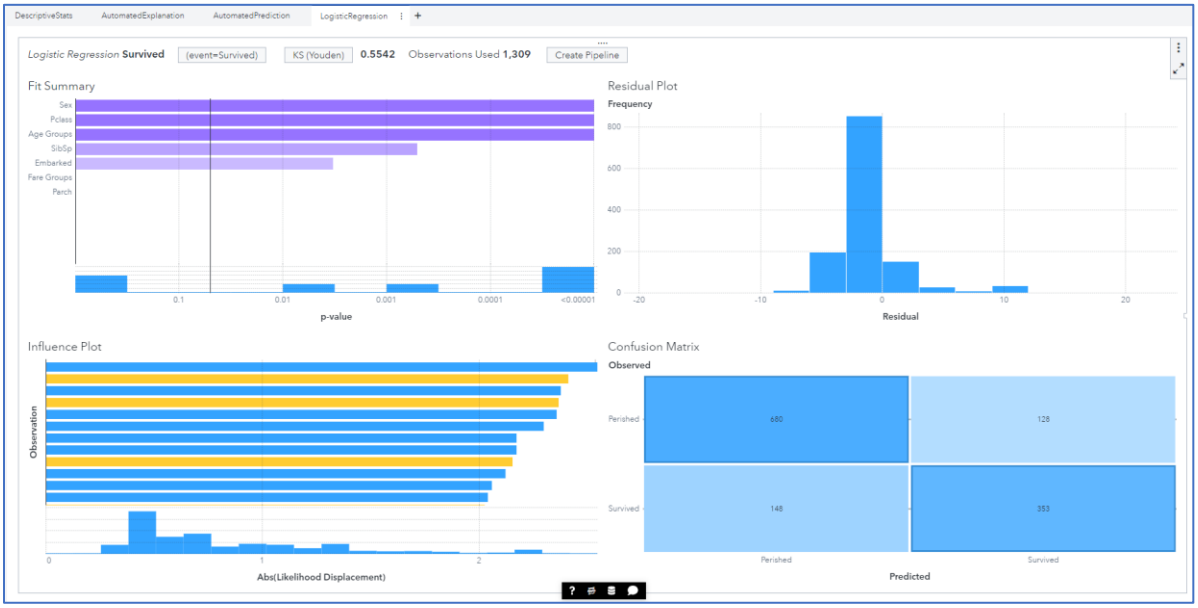
- Use the tool to examine the Survived variable
  - What interesting trends do you notice?

- **Now it is modeling time!**
  - Some modeling notes to get us started:
    - For all the models, create a new page and change the label to the name of the model.
    - The models are objects found under either SAS Visual Statistics or SAS Visual Data Mining and Machine Learning
- Start with a **Logistic Regression** model
  - Use the following Data Roles to set up your model:

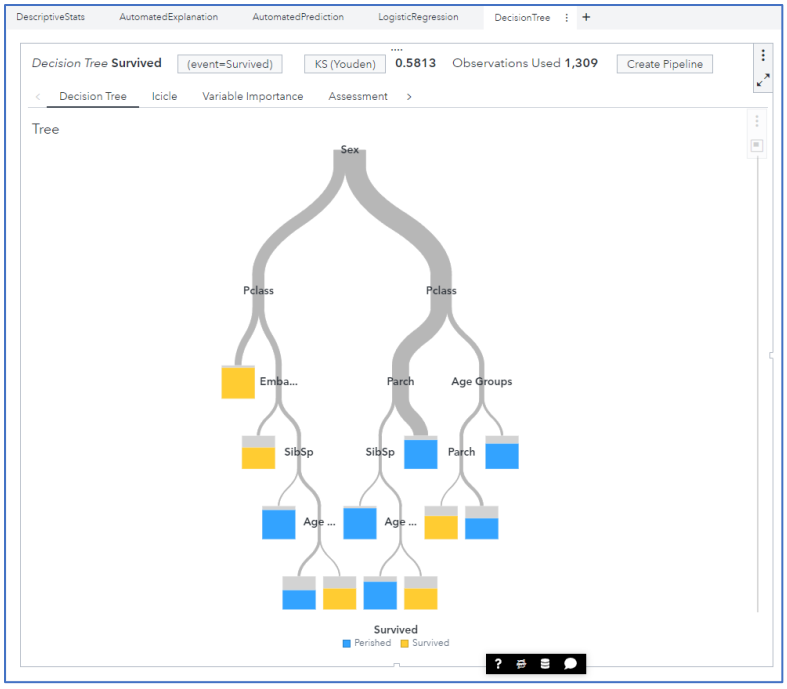


  - Let's change a few **Options**
    - For variable selection method, use fast backward

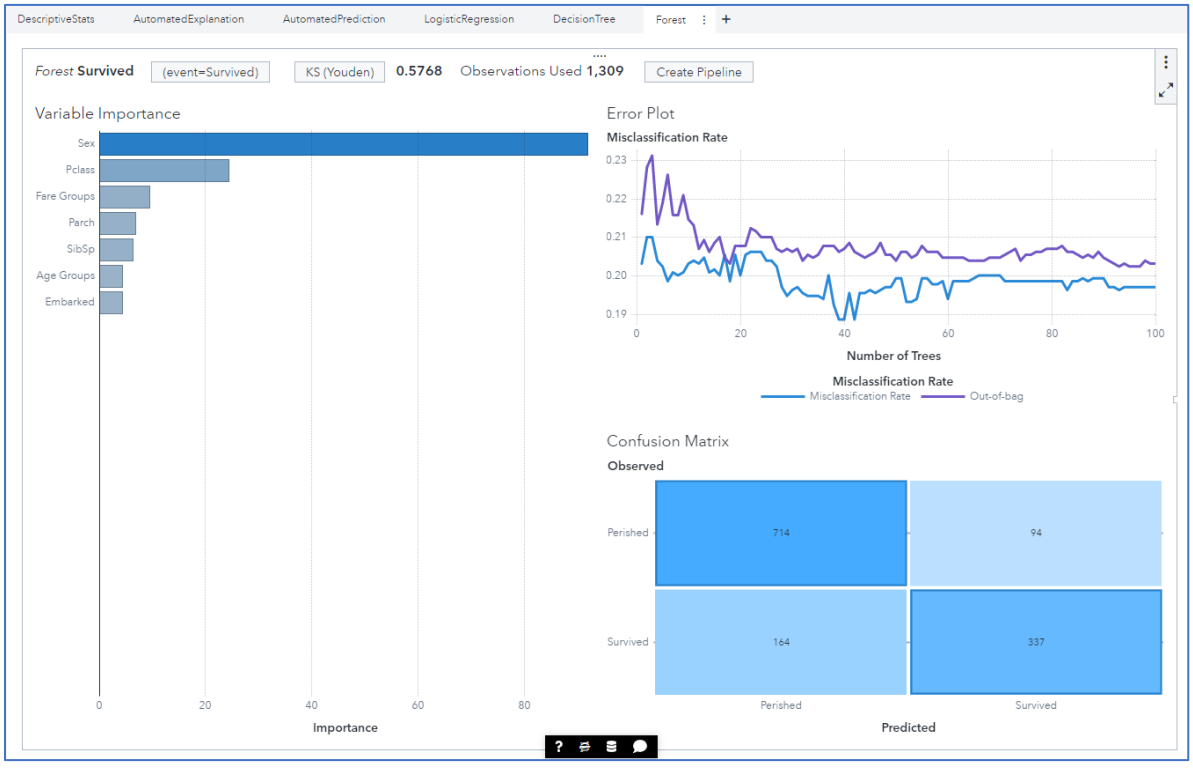
- For the residual plot ➔ use a histogram
- Under Influence plot/variable selection plot
    - Plot to show: influence plot



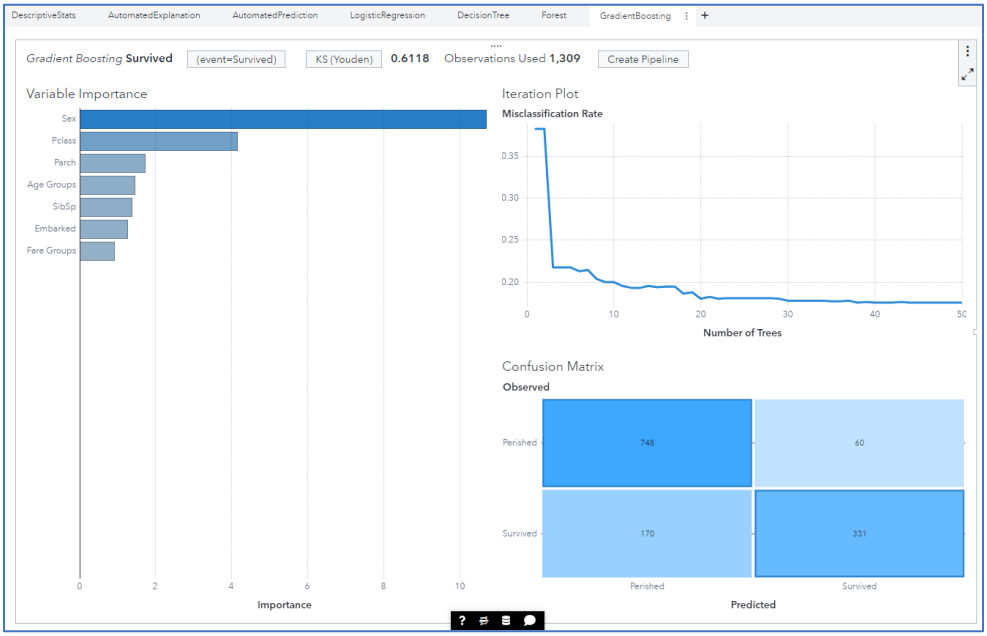
- Next up is a **Decision Tree**
    - To use your existing work, start with the Logistic Regression Object then:
        - Click alt + [duplicate on new page as… Decision Tree]
    - Let's explore some additional model options
        - Leaf size = 10
        - Model Display
            - Plot Layout: stack



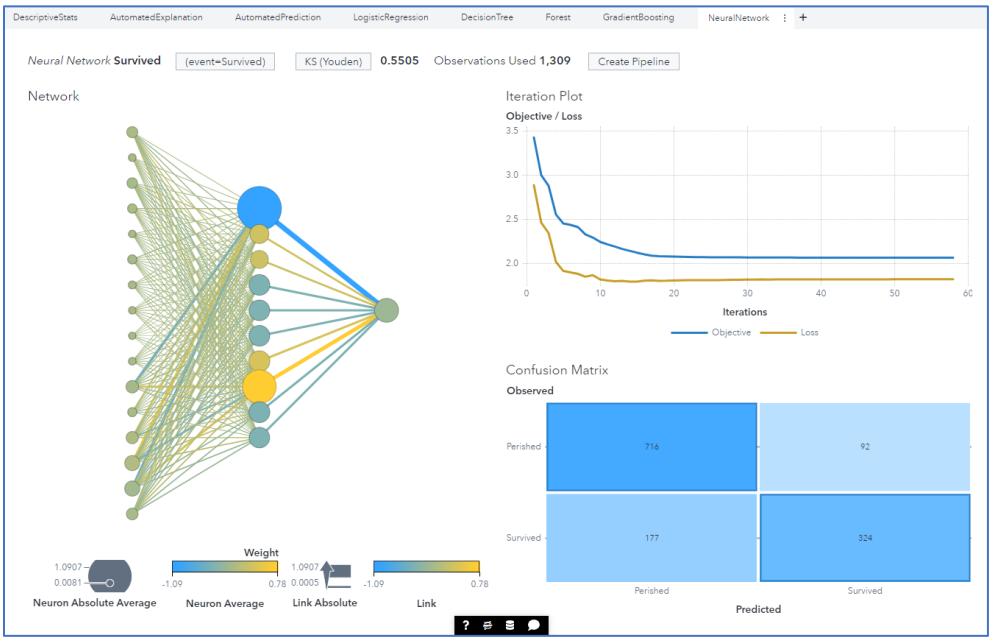
- Let's use a bunch of decision trees to create a **Forest**
  - Again, to use your previous work, start with the Decision Tree Object and then:
    - Press alt + [duplicate on new page as… Forest]
  - Additional options to explore
    - Leaf size = 10
  - Output:



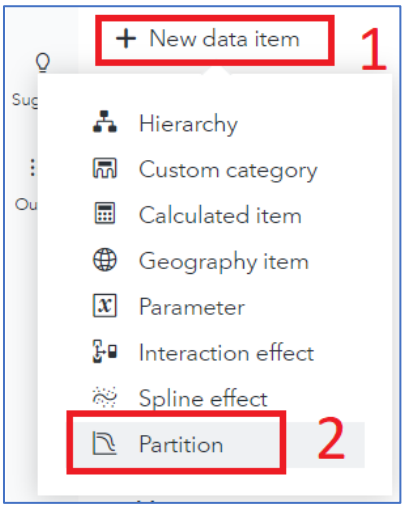
- **Gradient Boosting** is another tree-based model to explore
  - Similar approach as modeling the others
    - Forest Tree ➔ alt + [duplicate on new page as… Gradient Boosting]
  - Additional Setting
    - Leaf size = 10
  - Output:



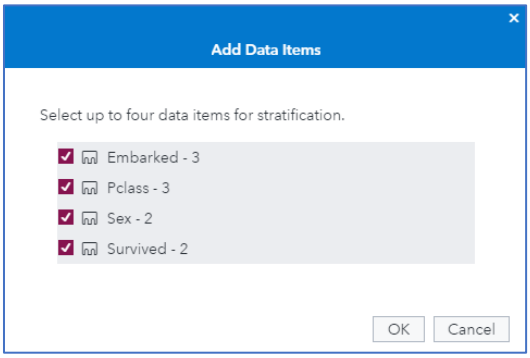
- Finally, let's end with a **Neural Network** model
  - One last time
    - Gradient boosting ➔ alt + [duplicate on new page as… Neural Network]
  - Accept defaults, which produces the following:



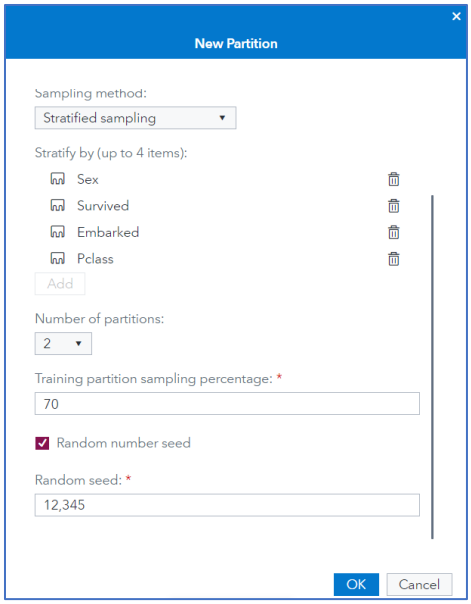
- It's time for **Model Comparison**!
  - We didn't do all that modeling for modeling's sake
    - Our goal is to find the best model, crown it champion, and do something with that model (oftentimes, predict new cases)
  - But, wait… should we have been performing honest assessment?
    - Yes! Oops!
    - Before comparing models, let's go back and fix the data!
      - +New data item ➔ partition



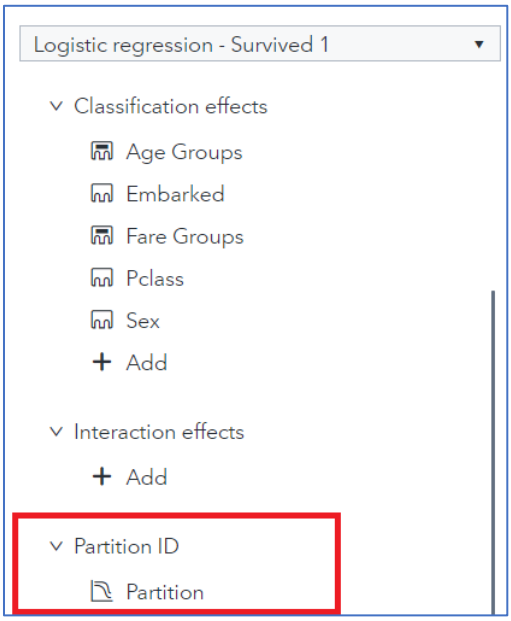
    - Use a **Stratified Sampling**, adding the following variables and settings:



      - Number of partitions = 2
      - Training partition = 70%
      - Random Seed = 12345
    - Your partition will appear as follows:

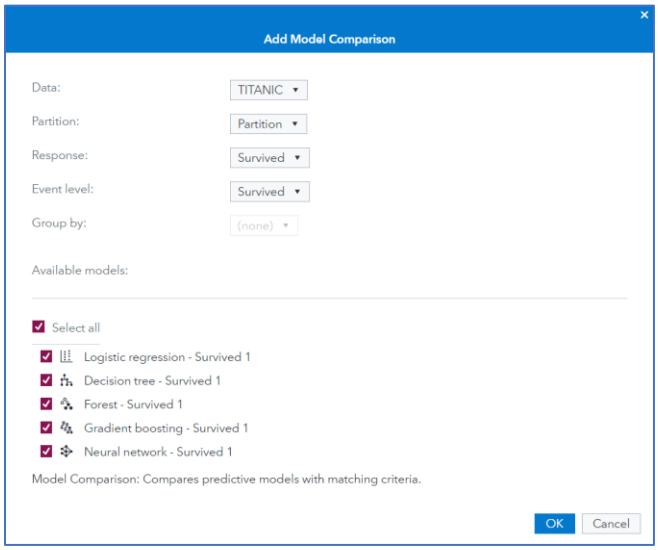


- Starting with the Logistic Regression, add the partition to the **Data Role**s in all our models and rerun!

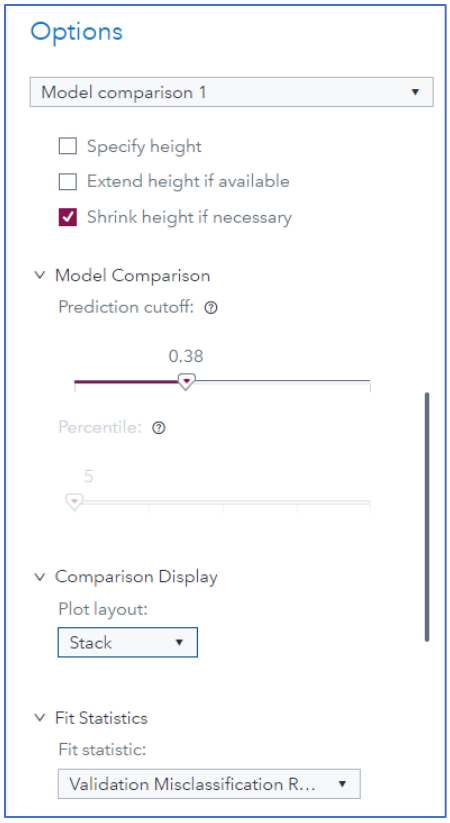


- **Which selection statistics should we choose?**
  - Discussion
    - What are you trying to optimize?
    - Which test statistic makes the most sense for your research question?
  - Settings
    - Let's choose Misclassification Rate (event) as the selection statistics
    - Use 38% as the *probability decision threshold*
- Now let's use the **Model Comparison Node** with the settings mentioned above

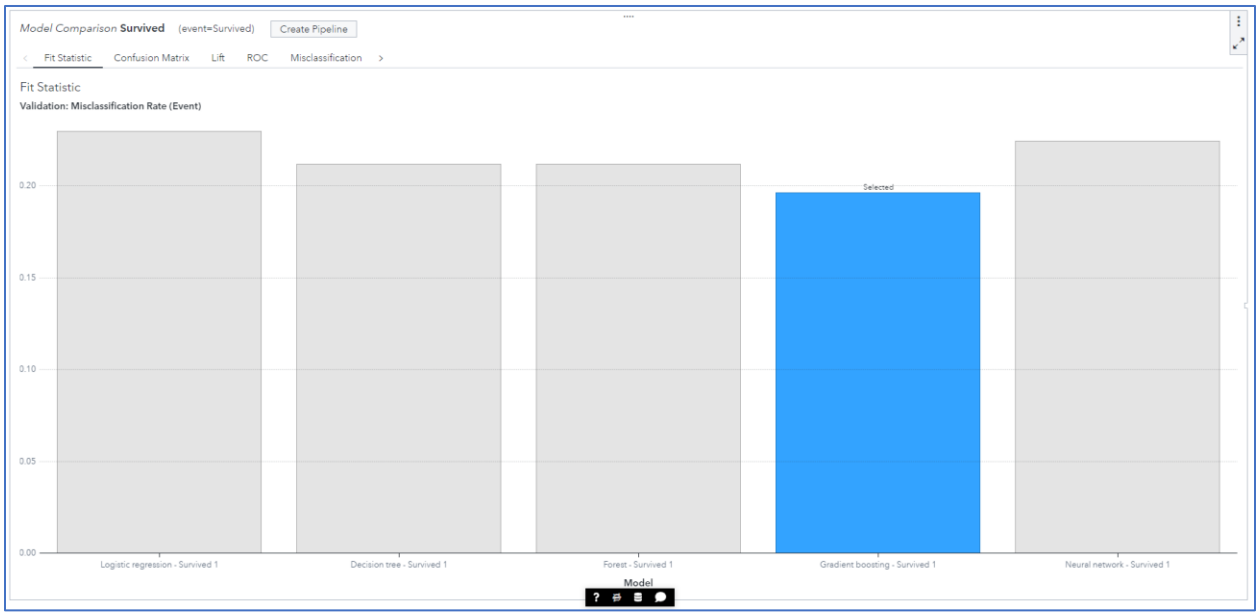
- From the options under SAS Visual Statistics, drag the *Model Comparison* object to a new page and select all available machine learning models:



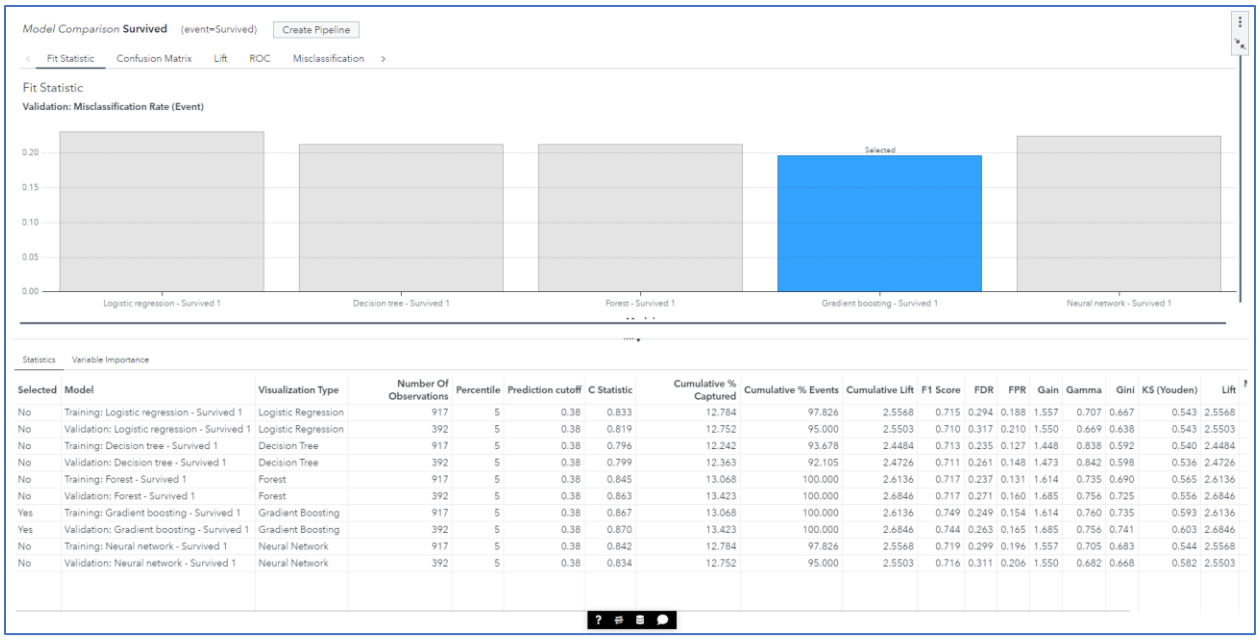
- Adjust a few of the **Options**:
  - Change the Prediction Cutoff to 0.38 (or 38%)
  - For the selection fit statistic, choose Validation Misclassification Rate
  - To make the output easier to digest, select:
    - Comparison Display ➔ Plot layout ➔ stack



- This yields the following:



- Which model was selected by the Model Comparison object?
  - The gradient boosting has the smallest misclassification rate!
- Examine the underlying data for more details:

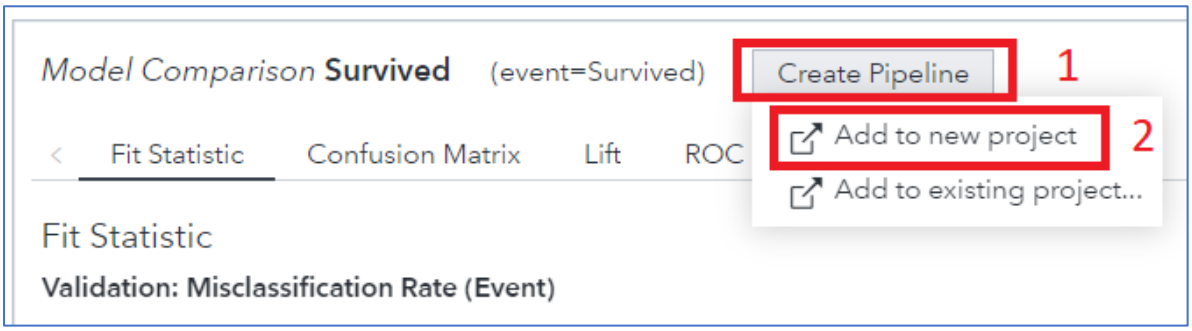


Selected	Model	Visualization Type	Number Of Observations	Percentile	Prediction cutoff	C Statistic	Cumulative % Captured	Cumulative % Events	Cumulative Lift	F1 Score	FDR	FPR	Gain	Gamma	Gini	KS (Youden)	Lift
No	Training: Logistic regression - Survived 1	Logistic Regression	917	5	0.38	0.833	12.784	97.826	2.5568	0.715	0.294	0.188	1.557	0.707	0.667	0.543	2.5568
No	Validation: Logistic regression - Survived 1	Logistic Regression	392	5	0.38	0.819	12.752	95.000	2.5503	0.710	0.317	0.210	1.550	0.669	0.638	0.543	2.5503
No	Training: Decision tree - Survived 1	Decision Tree	917	5	0.38	0.796	12.242	93.678	2.4484	0.713	0.235	0.127	1.448	0.838	0.592	0.540	2.4484
No	Validation: Decision tree - Survived 1	Decision Tree	392	5	0.38	0.799	12.363	92.105	2.4726	0.711	0.261	0.148	1.473	0.842	0.598	0.536	2.4726
No	Training: Forest - Survived 1	Forest	917	5	0.38	0.845	13.068	100.000	2.6136	0.717	0.237	0.131	1.614	0.735	0.690	0.565	2.6136
No	Validation: Forest - Survived 1	Forest	392	5	0.38	0.863	13.423	100.000	2.6846	0.717	0.271	0.160	1.685	0.756	0.725	0.556	2.6846
Yes	Training: Gradient boosting - Survived 1	Gradient Boosting	917	5	0.38	0.867	13.068	100.000	2.6136	0.749	0.249	0.154	1.614	0.760	0.735	0.593	2.6136
Yes	Validation: Gradient boosting - Survived 1	Gradient Boosting	392	5	0.38	0.870	13.423	100.000	2.6846	0.744	0.263	0.165	1.685	0.756	0.741	0.603	2.6846
No	Training: Neural network - Survived 1	Neural Network	917	5	0.38	0.842	12.784	97.826	2.5568	0.719	0.299	0.196	1.557	0.705	0.683	0.544	2.5568
No	Validation: Neural network - Survived 1	Neural Network	392	5	0.38	0.834	12.752	95.000	2.5503	0.716	0.311	0.206	1.550	0.682	0.668	0.582	2.5503

- Examine the Confusion Matrix, as well as the Lift, ROC, and Misclassification rates.
- Do you see any interesting trends?

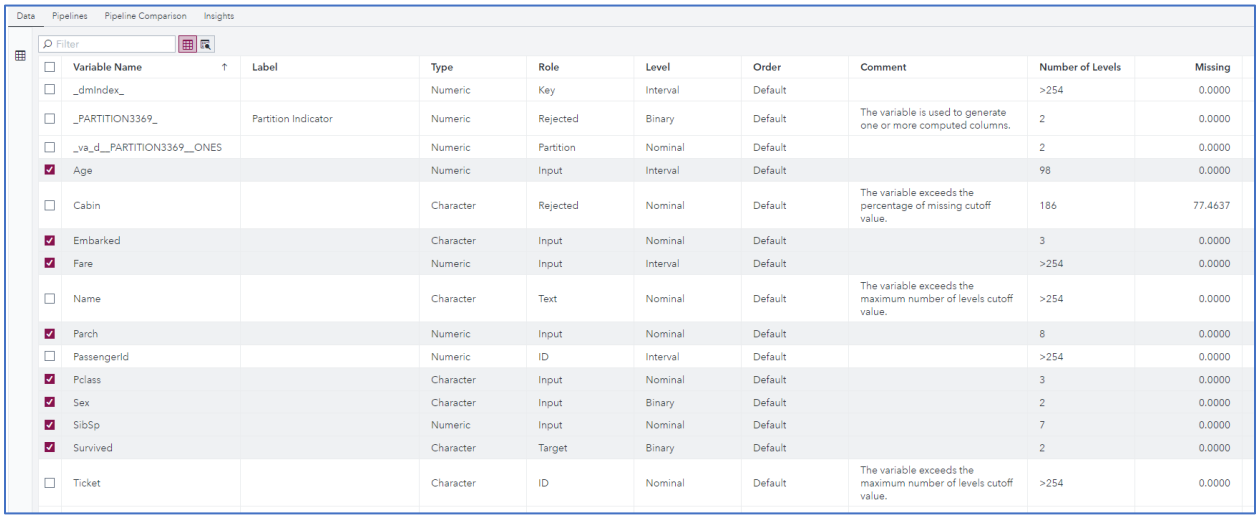
- Promote champion model to SAS Model Studio to get us started!
  - How do we do this?
  - Well, we are just two clicks away.

- First, select **Create Pipeline**
- Then **Add to new project**…



Part II: SAS Model Studio

- **Welcome to SAS Model Studio!**
  - Start with a tour.
- **Metadata**
  - It's still very important.
    - So, try not to skip over it – and simply plunge into the model. Instead, check it out under the **Data** tab.
  - Why is partition rejected?
    - SAS Model Studio automatically creates training/validation/testing samples in every pipeline. So, incoming partitions are, by default, rejected.
  - Here are the main variables in our analysis:

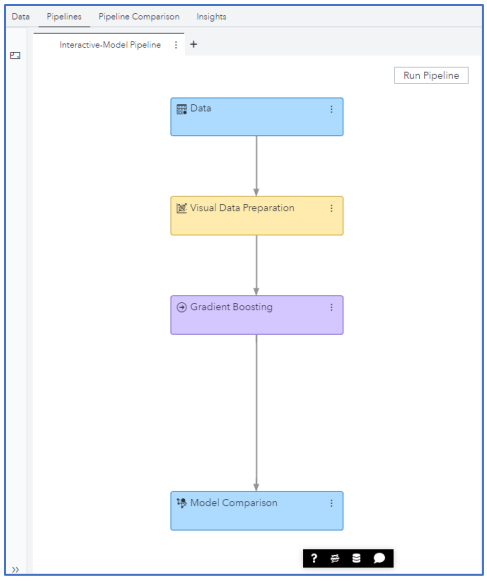


Data Pipelines Pipeline Comparison Insights

Filter

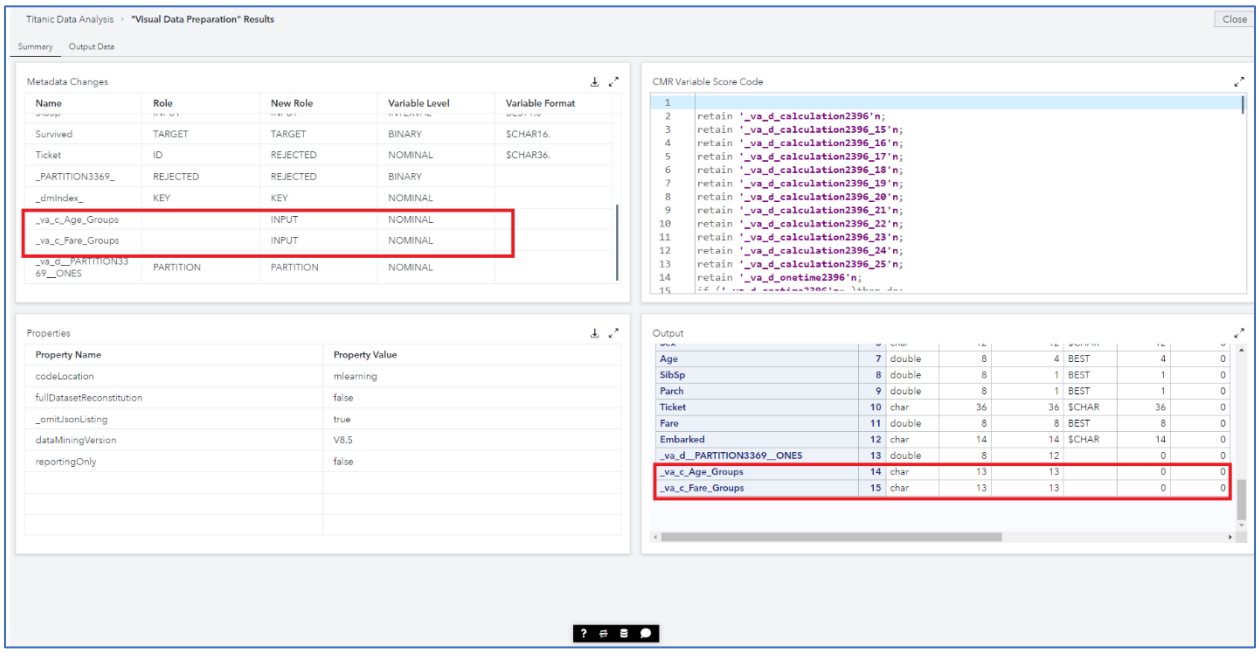
Variable Name	Label	Type	Role	Level	Order	Comment	Number of Levels	Missing
dmIndex		Numeric	Key	Interval	Default		>254	0.0000
PARTITION3369	Partition Indicator	Numeric	Rejected	Binary	Default	The variable is used to generate one or more computed columns.	2	0.0000
_va_d__PARTITION3369__ONES		Numeric	Partition	Nominal	Default		2	0.0000
Age		Numeric	Input	Interval	Default		98	0.0000
Cabin		Character	Rejected	Nominal	Default	The variable exceeds the percentage of missing cutoff value.	186	77.4637
Embarked		Character	Input	Nominal	Default		3	0.0000
Fare		Numeric	Input	Interval	Default		>254	0.0000
Name		Character	Text	Nominal	Default	The variable exceeds the maximum number of levels cutoff value.	>254	0.0000
Parch		Numeric	Input	Nominal	Default		8	0.0000
PassengerId		Numeric	ID	Interval	Default		>254	0.0000
Pclass		Character	Input	Nominal	Default		3	0.0000
Sex		Character	Input	Binary	Default		2	0.0000
SibSp		Numeric	Input	Nominal	Default		7	0.0000
Survived		Character	Target	Binary	Default		2	0.0000
Ticket		Character	ID	Nominal	Default	The variable exceeds the maximum number of levels cutoff value.	>254	0.0000

  - One Important note:
    - The variables we created were not transferred!
  - Where do they appear?
  - Where did they go?
    - Wait for it…
- **Interactive-Model Pipelines**
  - The variables we created will appear here!

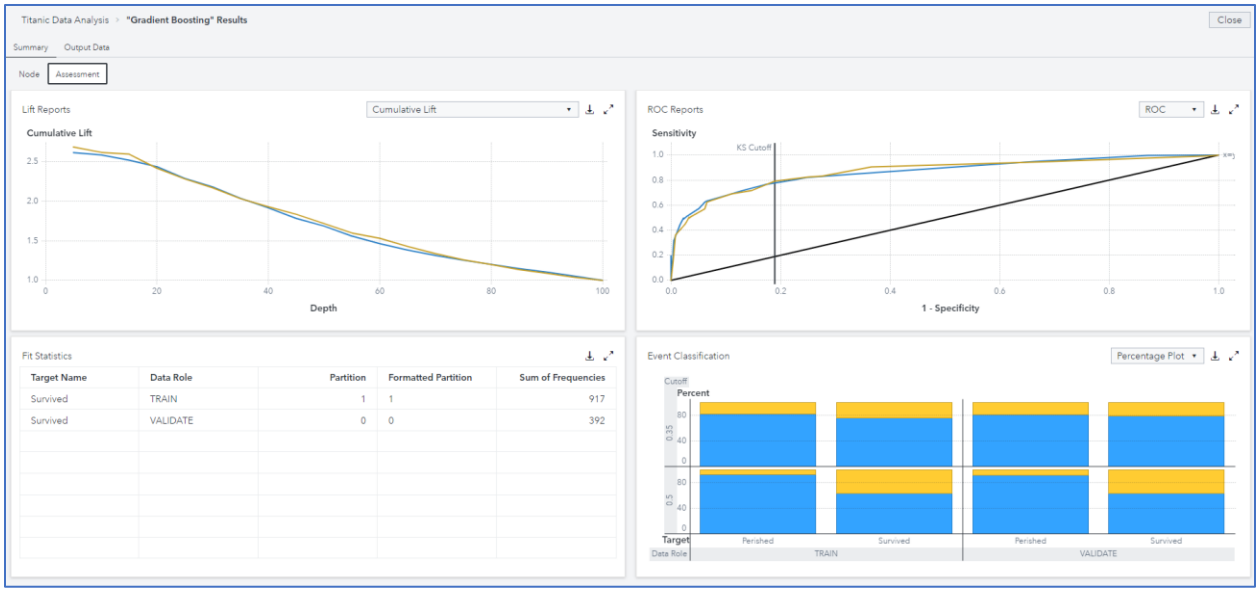


- To see the variables, start by running the pipeline by clicking here:

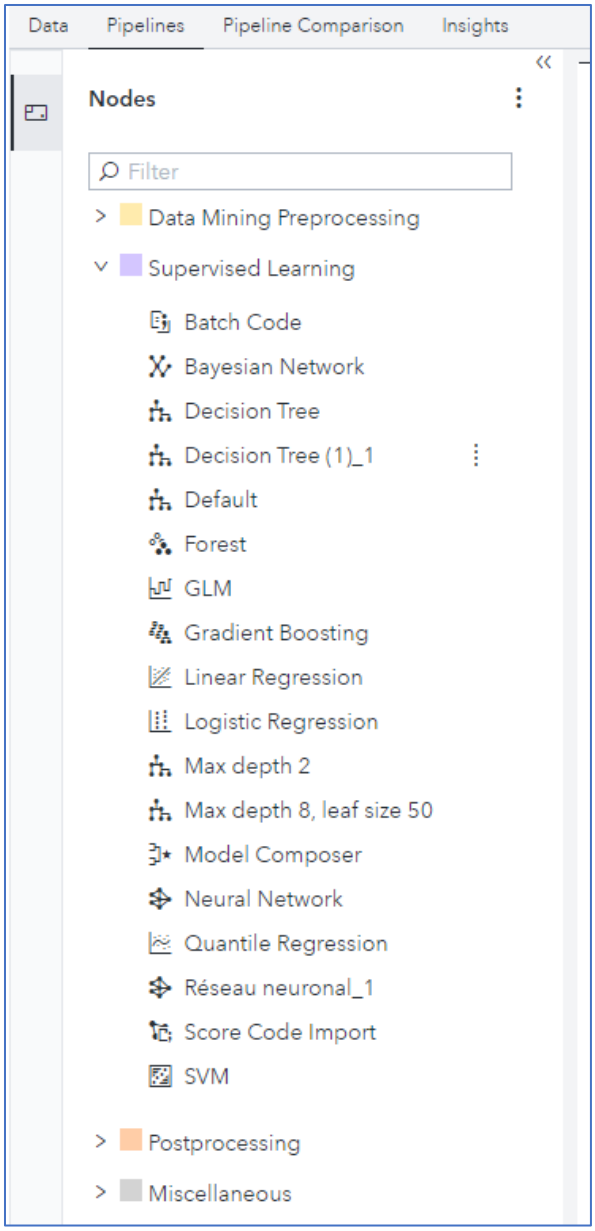
- Examine the results of the **Visual Data Preparation** node
  - Were the variables created?
    - Yes!
    - And here is the proof:



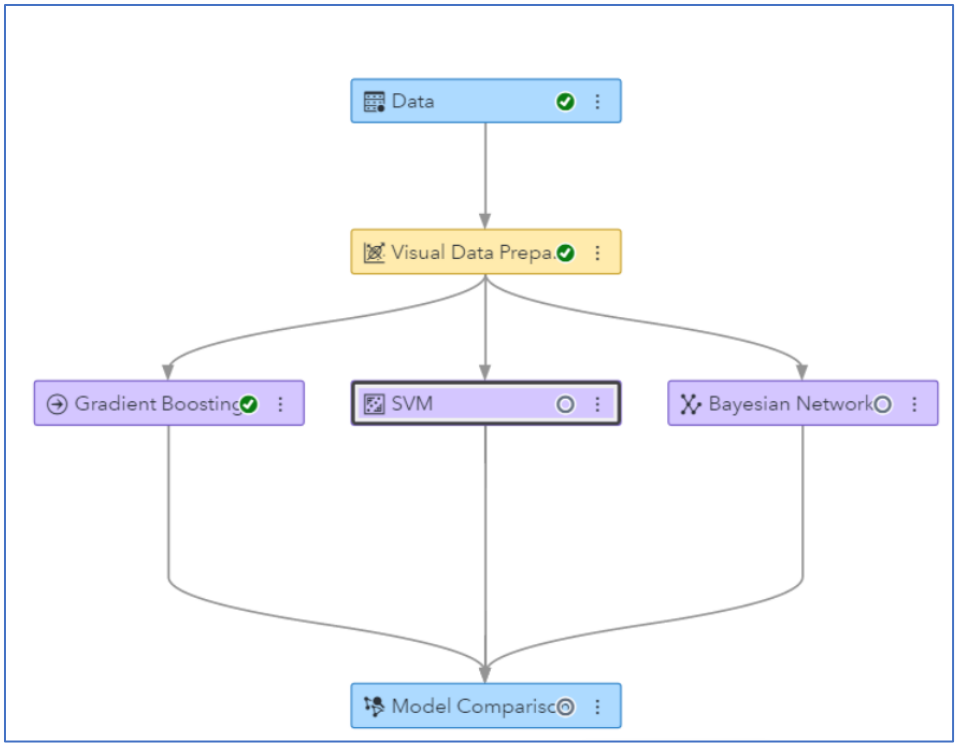
- Now, let's examine the Gradient Boosting Model



- Why stop here? Let's add more models to this pipeline!
  - General goal:
    - Can we do better than the default models of SAS Visual Analytics?
    - I think we can!
- Start by examining the types of models under the **Supervised Learning Nodes**
  - Note: your view of supervised learning models will differ slightly from mine – simply because we are (likely) on difference instances of SAS Viya for Learners, and individual users can save custom nodes to the shared spaces.
  - For examples, "Max depth 2" is a user created node – whereas "Bayesian Network" is a node created by SAS. Use the SAS nodes to start.



- To keep it simple – yet still expose you to new tools – let's add a SVM Model and a Bayesian network:
  - There are two general options to add nodes to an existing pipeline
    - You can drag-and-drop the node on top of Visual Data Preparation, or
    - Right-click on Visual Data Preparation and add the tool below.
  - After adding both notes, simply accept the defaults and your pipeline will appear as follows:



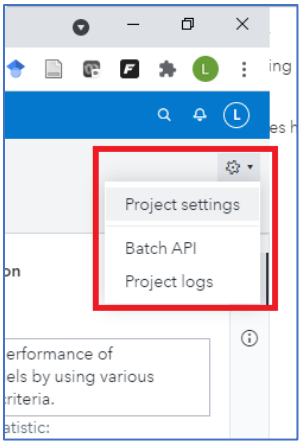
- Open the Model Comparison node and examine the models:

Model Comparison

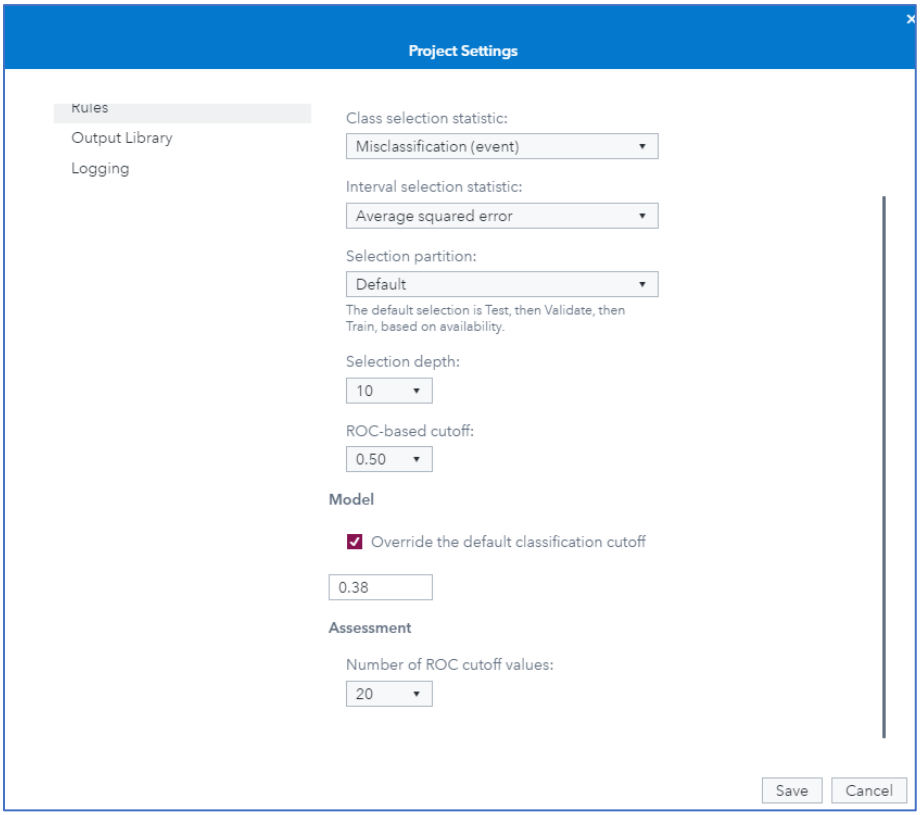
Champion	Name	Algorithm Name	KS (Youden)	Misclassification Rate
★	Gradient Boosting	Gradient Boosting	0.6087	0.1786
	Bayesian Network	Bayesian Network	0.5323	0.2270
	SVM	SVM	0.5241	0.2194

- Did we beat the best model from SAS Visual Analytics?
  - No. No we did not.
  - On the plus side, it looks like the default model from SAS VA was a decent one!
- Moreover, note that we have many selection statistics to choose from.
  - We show just the KS(Youden) and Misclassification Rate in the screenshot above – but there are many more options available!

- **Let's assume that we want to choose the misclassification rate.**
  - We need to now adjust the global project setting to reflect this modeling choice.
  - To start, open project settings



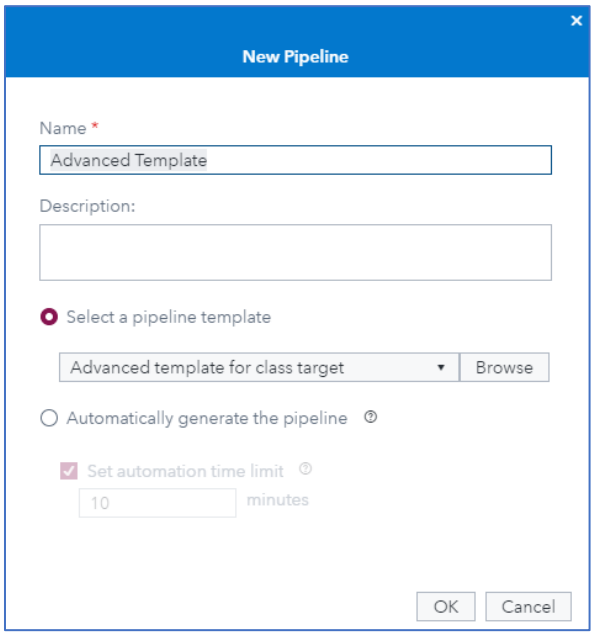
- Under Rules, select the following:
  - Class selection statistic: Misclassification (event)
  - Override the default classification cutoff = 0.38
- The final setting with appear as follows:



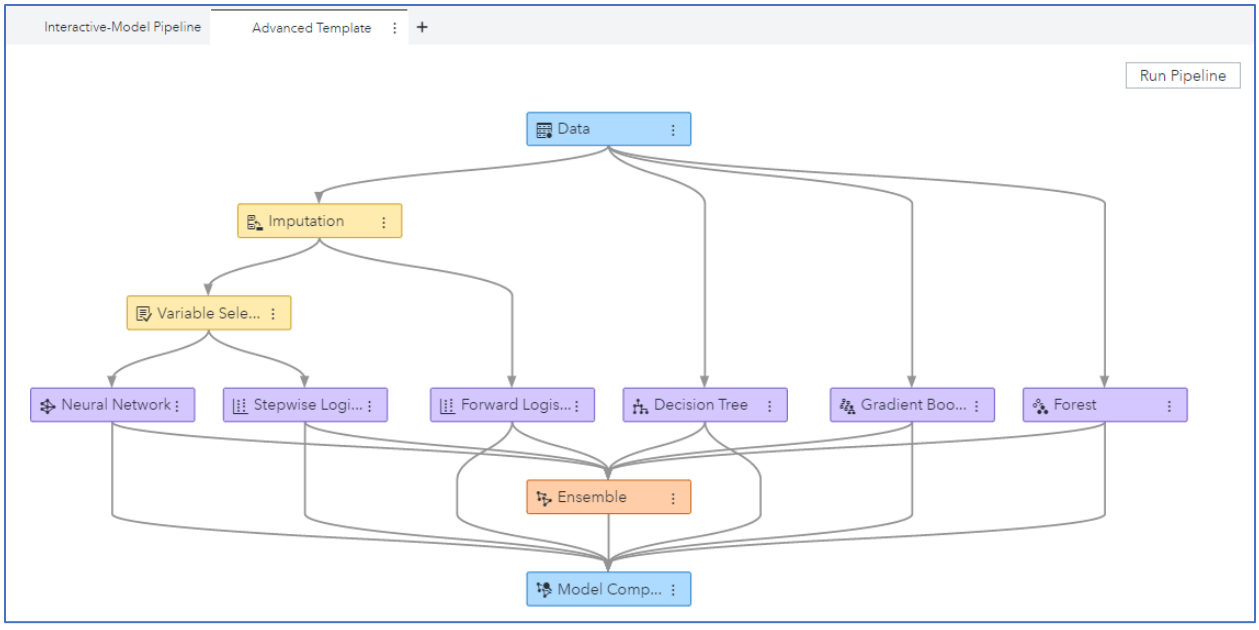
  - Make sure you save before you exit this window
- Rerun your Interactive Model Pipeline (if needed)

- **Now let's quickly add more pipelines – because this is the true value of SAS Model Studio!**
  - We can add two pre-built pipeline templates to our analysis – and see if we can beat our SAS VA gradient boosting model
  - Pre-built templates are a great tool to speed up the modeling process

- **Template 1 – Advanced template**
  - Use the following settings:



    - Hint: Click on the **Browse** button to find the *Advanced template for class target*
  - The new pipeline will appear as follows:

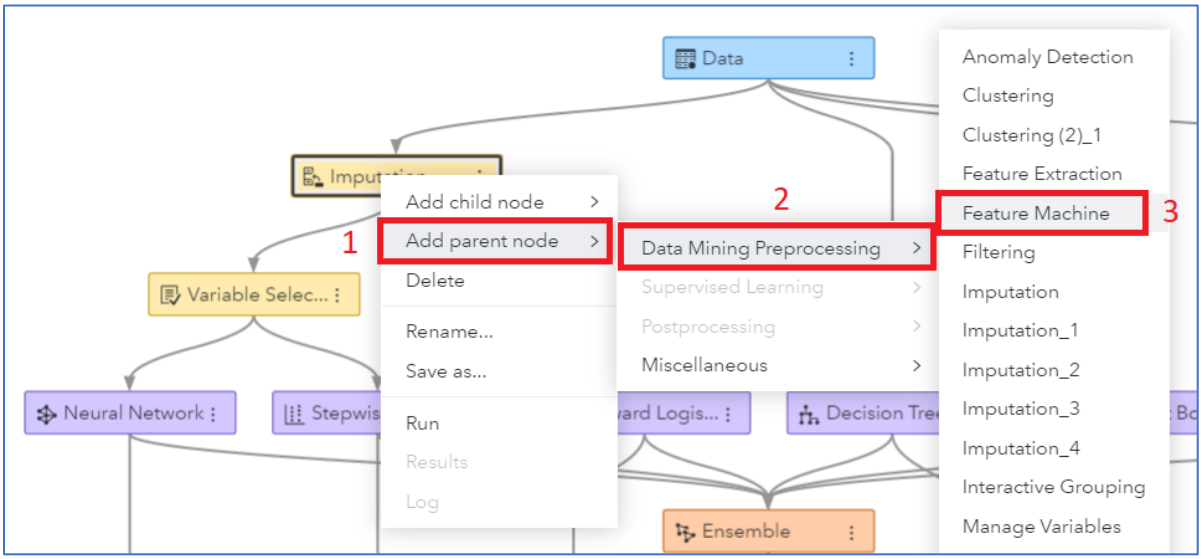


  - This template is a great start…
    - But, it will not necessarily account for the fare and age variables we created.
      - Note: implicitly, tree-based models will, because split points can occur at any point on the interval. So, this is actually a significant upgrade over the variables we created.

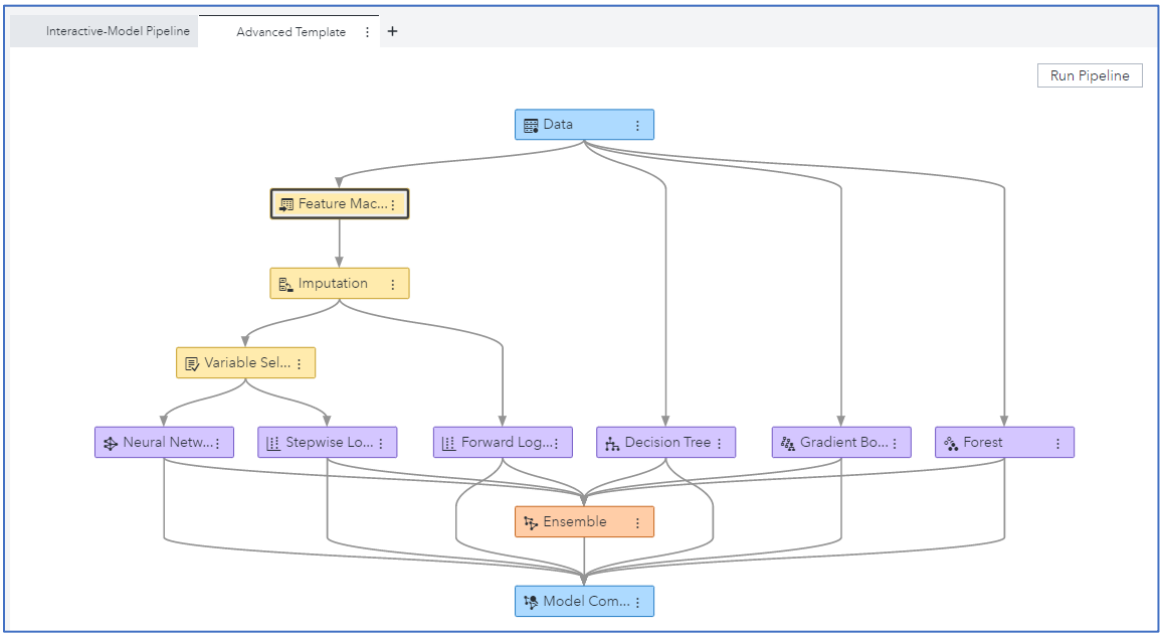
- But those are just half our models.

- For the Neural Network and the Logistic Models, we'll need to create new variables. Never fear, we can do this automatically with a **Feature Machine**!
  - Learn more about this AI tool here: https://go.documentation.sas.com/doc/en/vdmmlcdc/8.5/vdmmlref/p180c9r7trn944n1ei7hilzu4cst.htm

- You'll need to add the Feature Machine in a very precise location to have it affect our non-tree based models.
  - Start by selecting the **Imputation** node.
  - Right click, select Add parent node ➔ Data Mining Preprocessing ➔ Feature Machine:

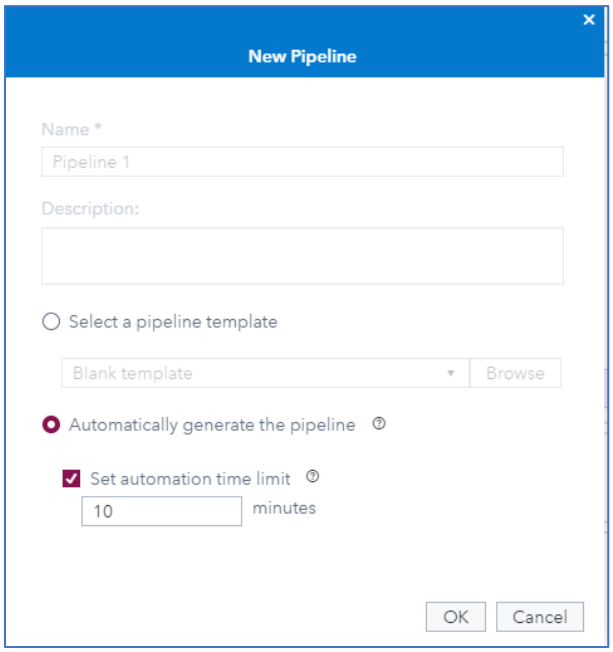


- Your new pipeline now appears as:

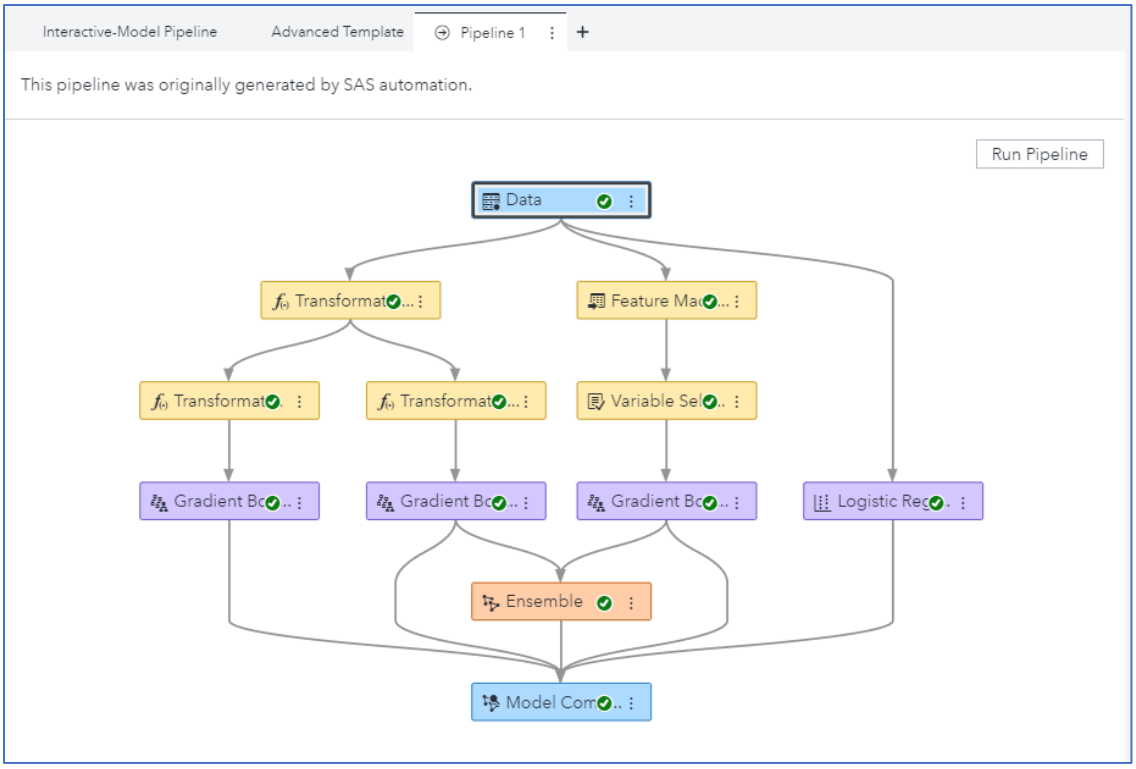


- Let it rip by clicking on *Run Pipeline*

- While that pipeline is running, create a third pipeline – this time **automatically generating the pipeline**
  - Create this **New Pipeline** by using the following settings:



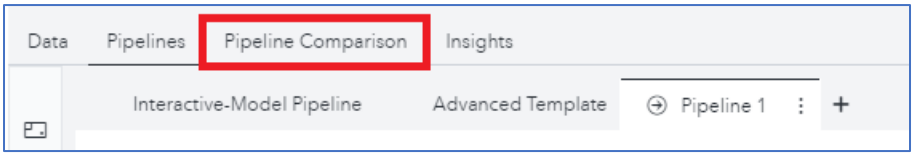
  - Keep the default automation time limit of 10 minutes.
  - Click OK and generate the pipeline
    - Here is what mine looks like:



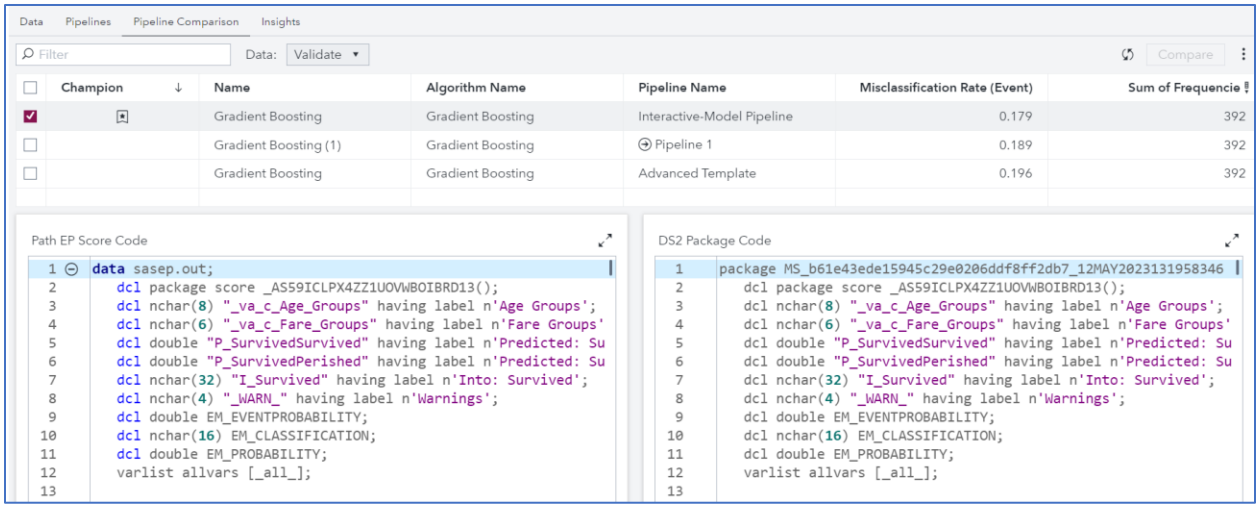
- Note: yours can/will look different!

  - Run the pipeline once it is created

- **It's coffee break time!**
  - Take a break while the pipelines are running.
  - Don't worry, they're doing all the hard work for you.
- **Now we have pipelines to compare!**
  - While you can certainly examine the individual pipeline, most machine learning guru's only care about the predictive model performance – and want to get right to the champion models. We'll do the same.
  - Click on **Pipeline Comparison:**

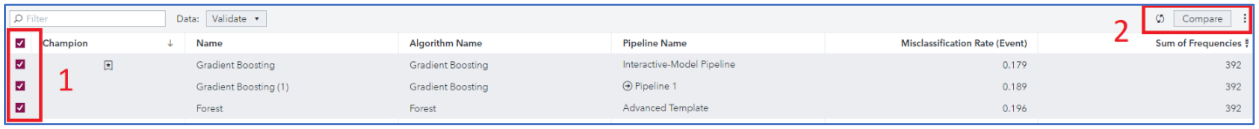


  - Which model has been crowned the champion? Our results:

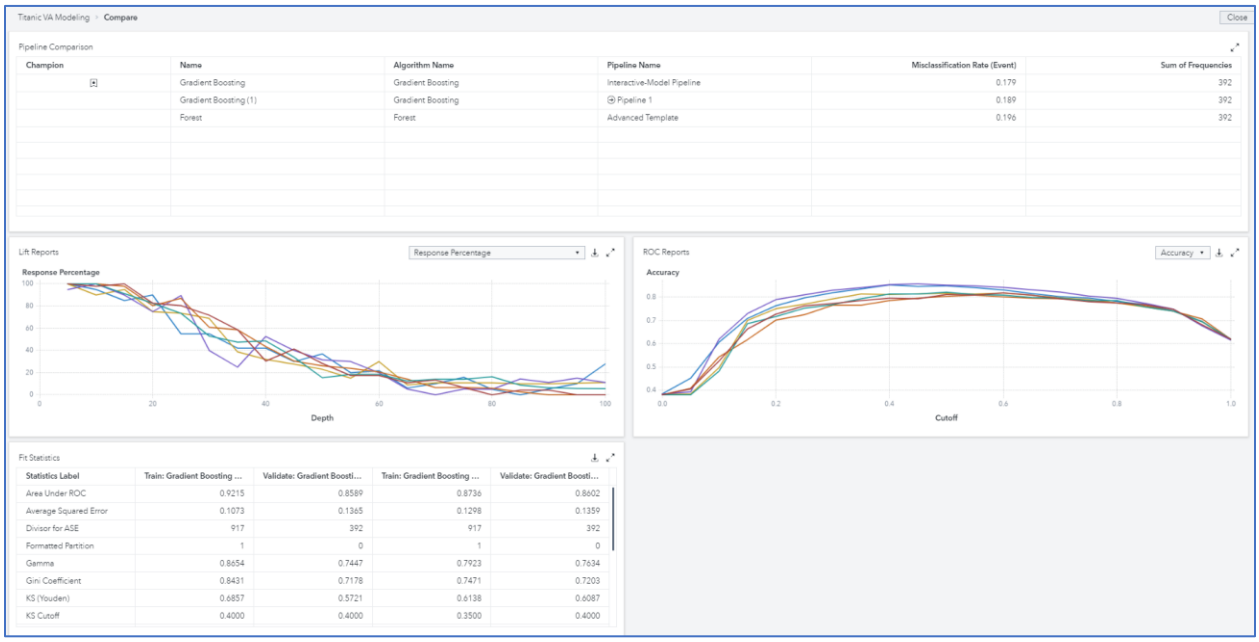


- Our gradient boosting model retains its crown

- But, let's compare the **diagnostic statistics** for all three models:



- The output from this comparison:

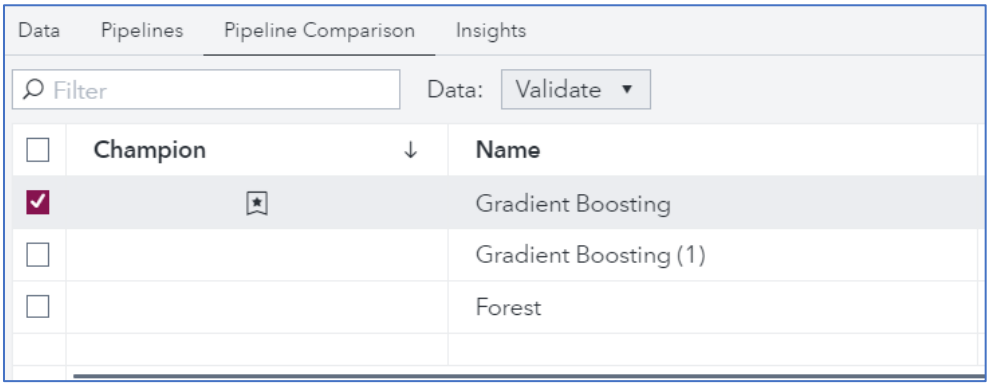


- Anything interesting in this output?
- Select **Close** when you're done comparing the models

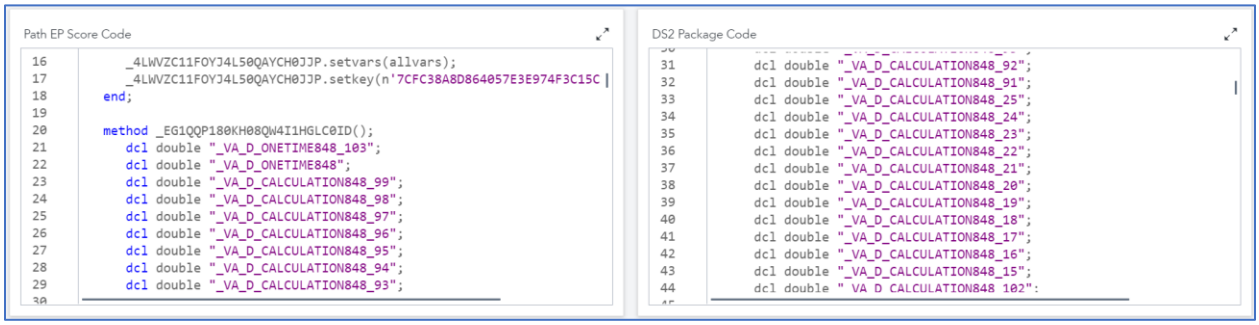
- Finally, let's find the **Score Code** for the champion model.
  - But, wait – what does it mean to "Score"?
    - Scoring simply means that we'll apply the parameters estimated in the champion model to a new set of data.

- In other words, we'll use the old data to predict new cases – often with new data – and, typically, want to predict an event that hasn't occurred yet.

- **Step 1**: select the single model



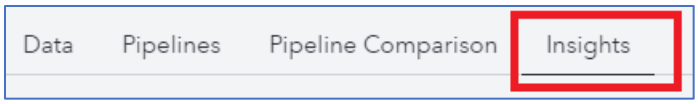
- **Step 2:** scroll and find the **SAS Score Code**



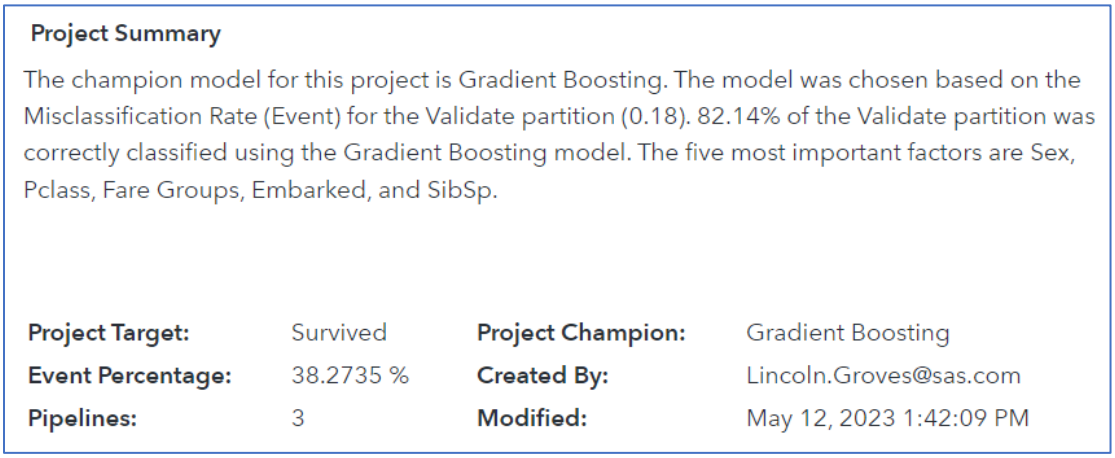
- This is the SAS code that you could use to apply the entire analysis you've conducted – in SAS Model Studio – to a new data set. Two notes:
  - In SAS Viya, there is a point-and-click tool that allows direct application of the champion model on new data, and it's called **SAS Model Manager**.
    - But that's a story – and tool – for another time.
  - Secondly, I'm not sure the Titanic case is the best predictive modeling example.
    - Why?
    - Well, it seems a little inappropriate/morbid to model the survival rates of passengers on a ship that hasn't sunken – yet – using data from 1912.

- Finally, one thing we haven't spent a lot of time analyzing is **which explanatory variables matter most** in predicting *Survived* on the Titanic.
  - Why?
  - Well, again, the truth is that machine learning specialists don't really care about the RHS variables – they just care that the model predicts well, relative to all other options.
    - Therefore, we fixate on competitor models – and our goodness-of-fit statistics.
  - All that stated, this doesn't mean that others won't still be interested in the inner details of your model.
    - In other words, many in your audience will want to know that your models make sense, even if predictive accuracy is really all they should care about.

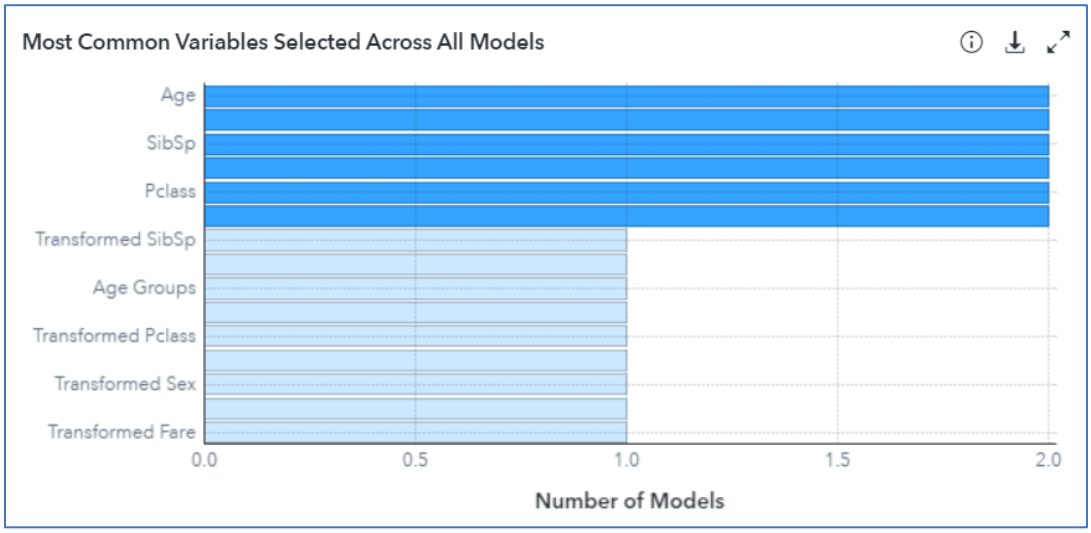
- Have no fear – there's a SAS tool for that:



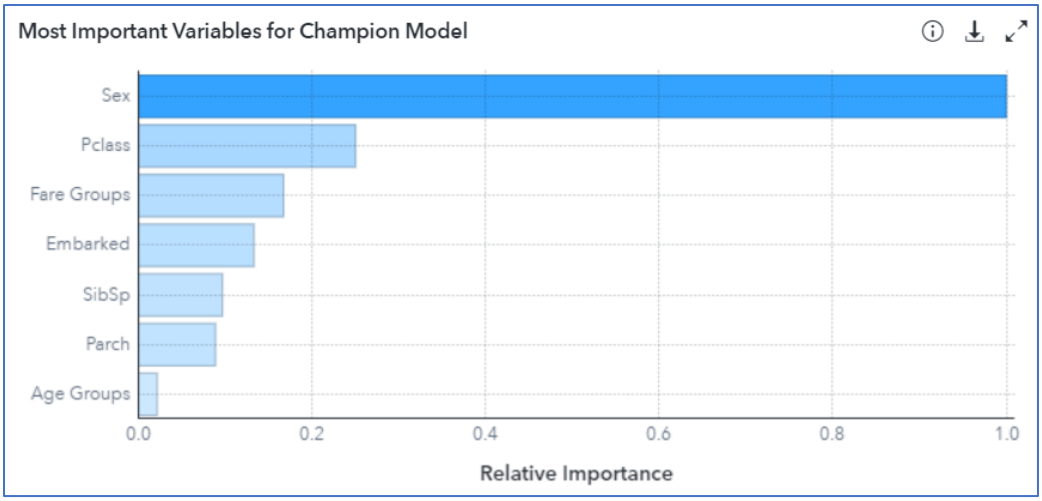
  - Under the Insights tab, you'll find a whole host of information that will help with your storytelling. Some fun highlights:
    - Project Summary:



    - Most common variables used across all modeling options:



    - Most important variables in the champion model:

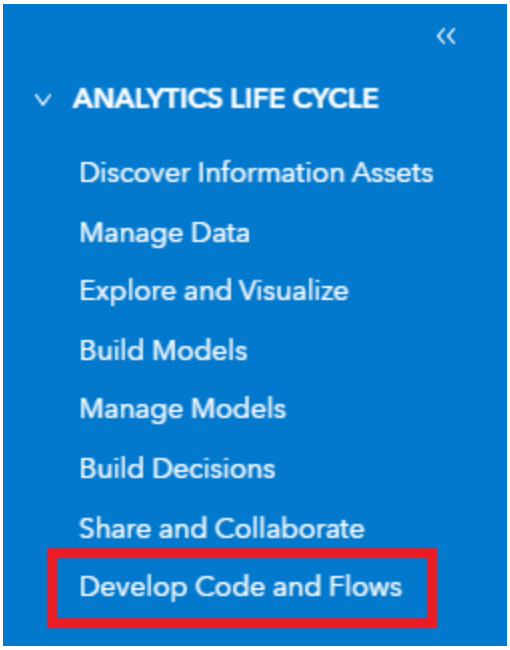


- For details on relative importance and it's calculation in an example, learn more here.

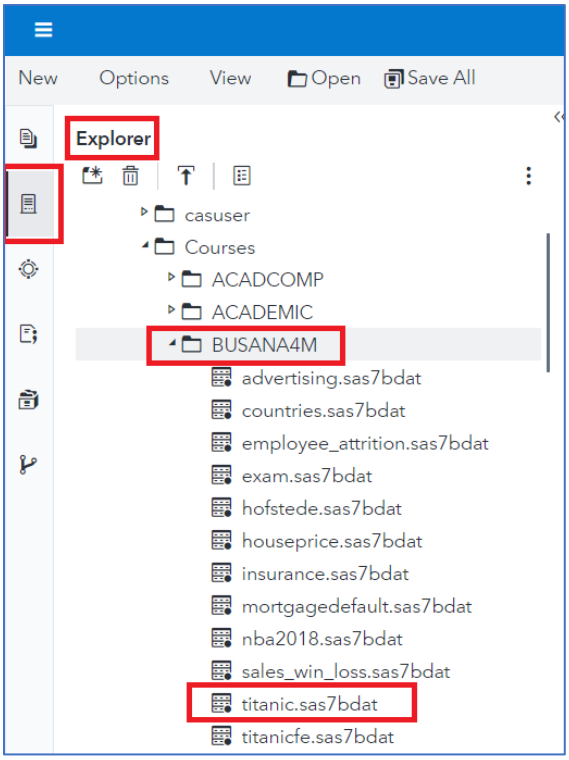
Part III: Programming in SAS Studio

- **Overview**
  - This section is all about the SAS Studio Tasks
  - We will explore the following data lifecycle tools:
    - Prepare Data
      - Examine Data
    - SAS Viya Prepare and Explore Data
      - Transform Data
    - SAS Viya Machine Learning
      - Supervised Learning
- **How to find SAS Studio?**
  - Return to the Hamburger in the upper left-hand side!

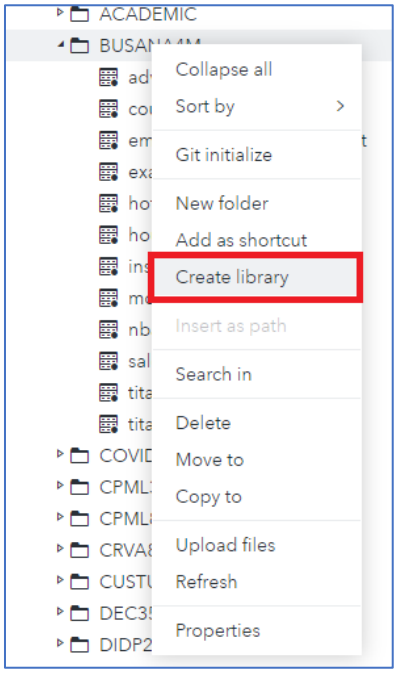
  - Select *Develop Code and Flows*



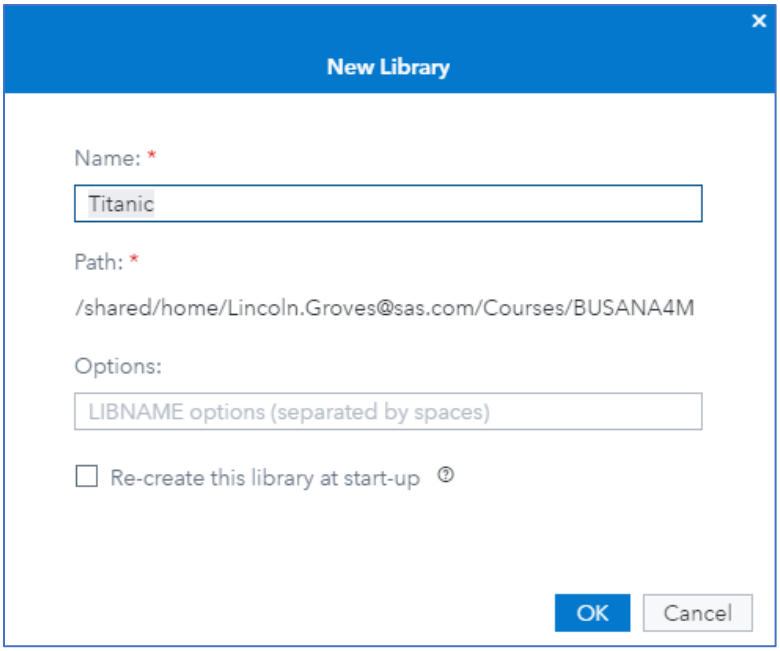
- **We need to first prepare the data**
  - **Step 1** – find the Titanic Data in SAS Studio



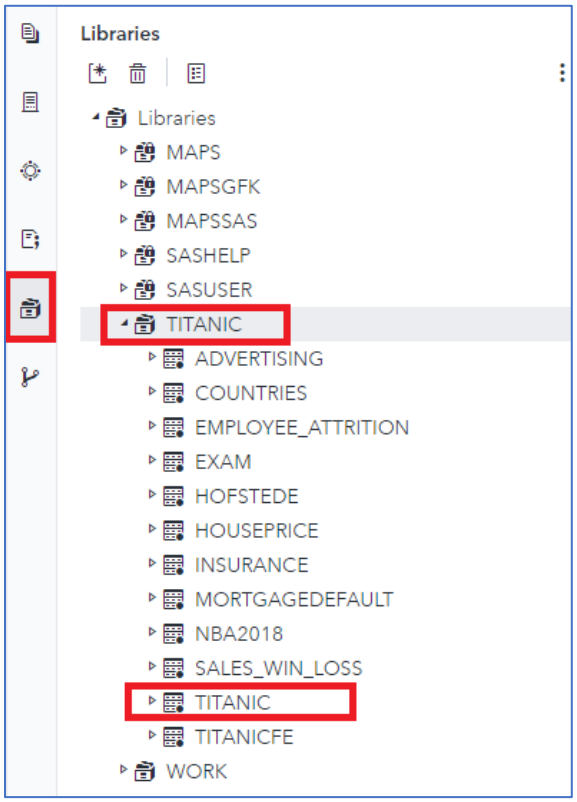
- Step 2 – create a SAS Library
  - Wait, what is a SAS Library?
    - Put simply, it tells the software where your data sets are stored
  - To create a SAS library, right-click on the BUSANA4M folder and select "Create library":



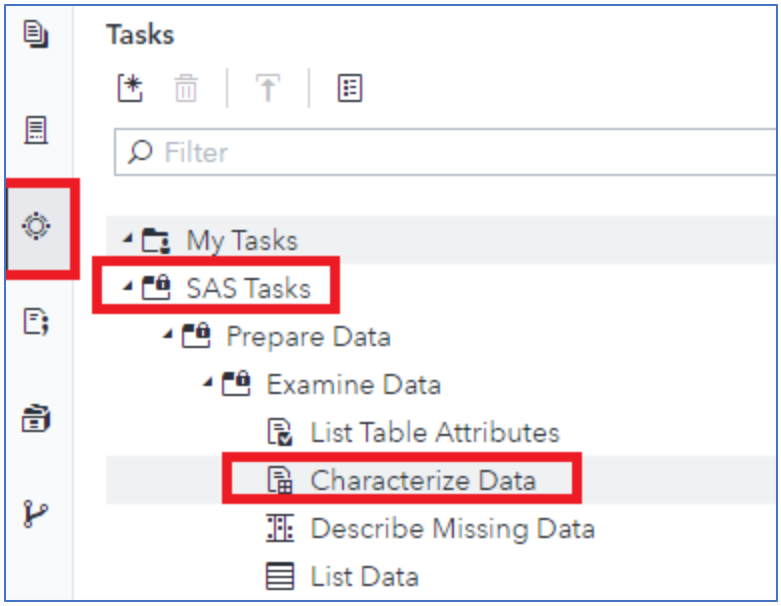
  - Use the following general settings in your new library:



- Ensure that your library has been initialized by checking that your library can be accessed:

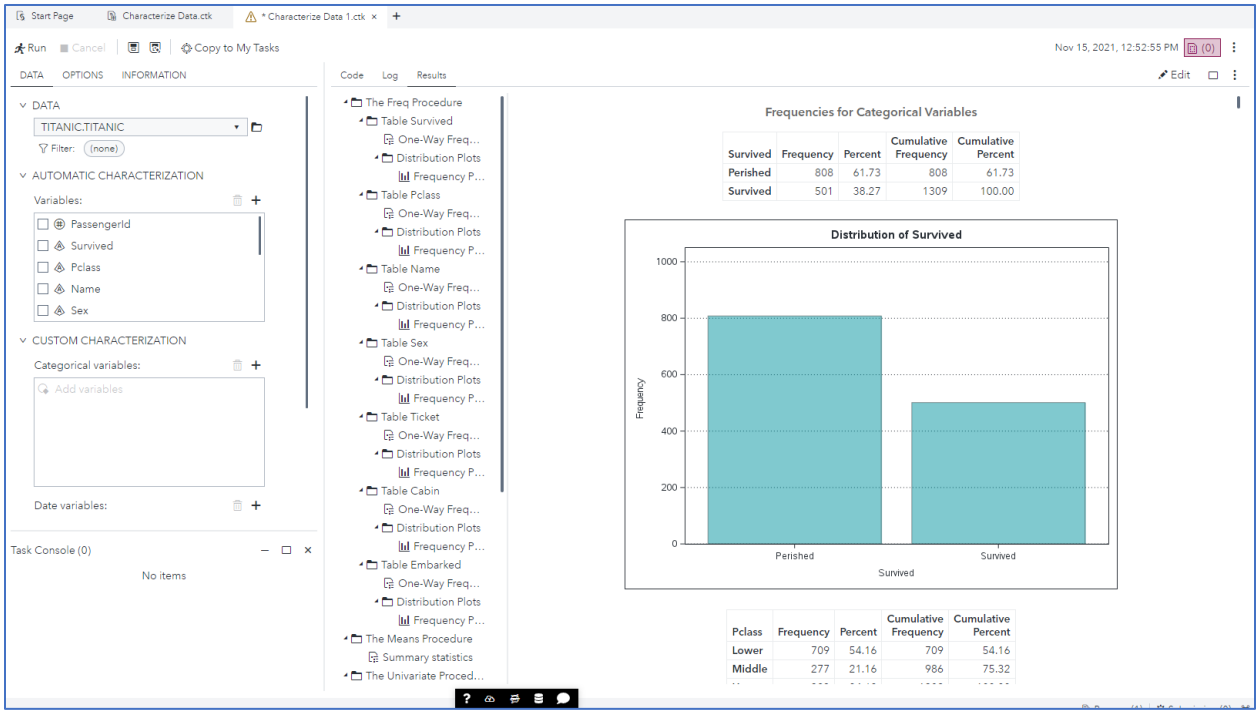


- **Better understand your data with the Characterize Data Tasks**
  - Find it here:

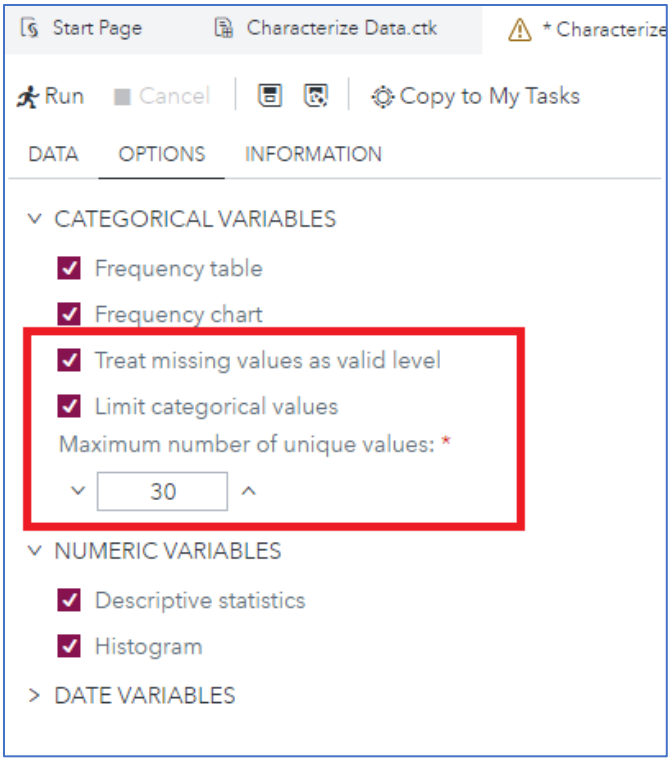


- In the Characterize Data Task, do the following:
  - Select **titanic.titanic** as the Data
  - Select all variables for Automatic Characterization
- See the settings and SAS Code below:

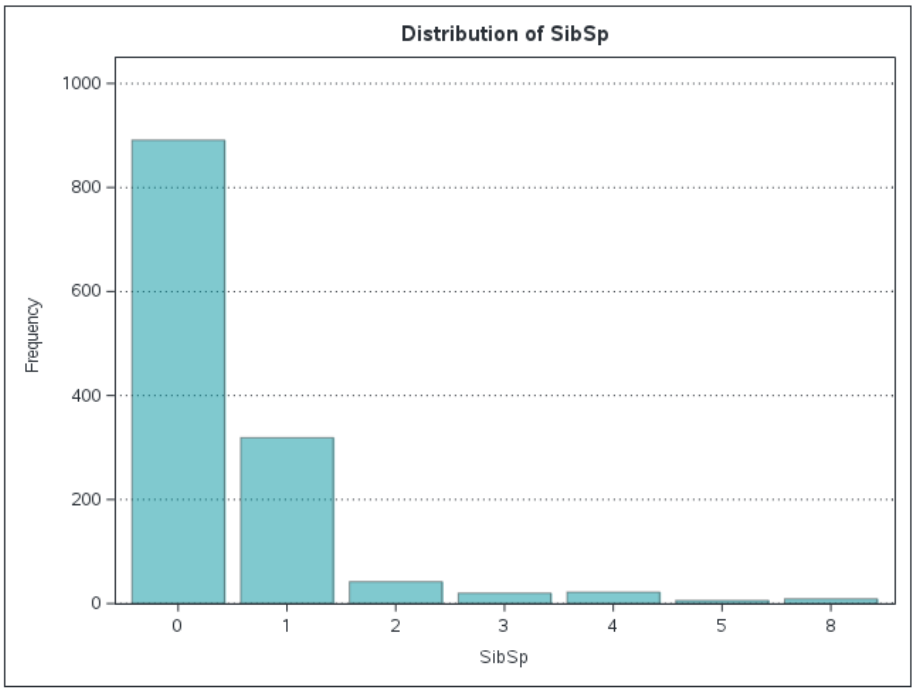
- Run the code by clicking Run and examine the output as follows:



- TMI? Yeah, I think so! Let's make the following changes to the SAS Studio task:
  - Drop *PassengerID, Name*, *Ticket* from **Automatic Characterization**
  - Change *SibSp* and *parch* to **Categorical Variables**
    - They will display better as discrete variables
  - Click on the **Options** tab. Then under **Categorical Variables** options, select:
    - Treat missing values as valid level
    - Limit categorical values to a max of 30 variables
      - This is not required but useful to know it exists

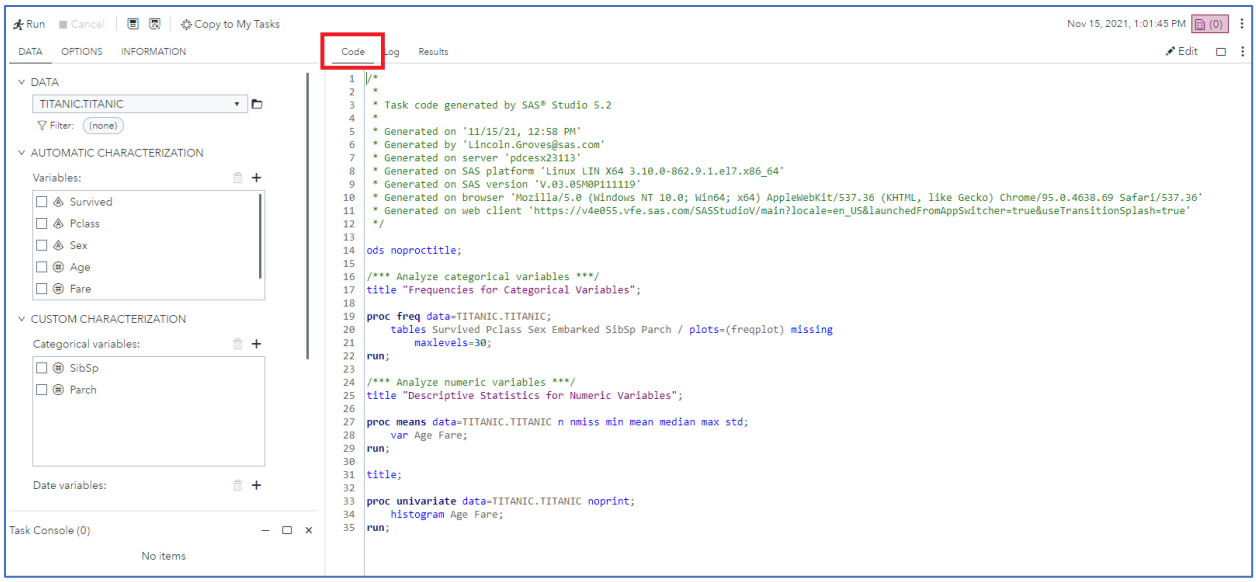


- Rerun the SAS Studio Task and examine the output
  - In particular, examine the distribution of Sibling/Spouse:

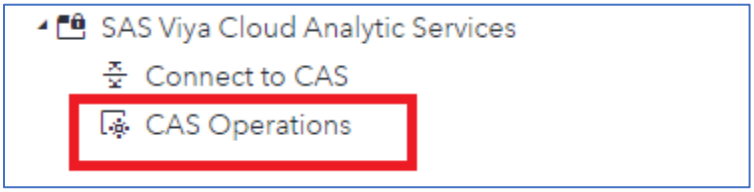


  - That's a bit easier to read…
- As a last step, examine the SAS Code. We'll return to the SAS Code eventually.

- But it is nice to know that SAS is doing a lot of work for you behind the scenes.

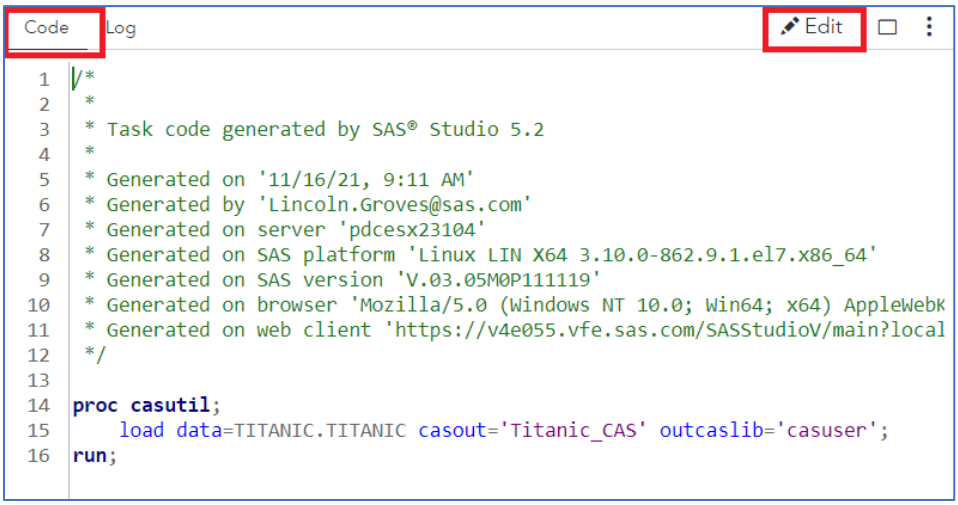


- Our next step in the data prep process is to use **SAS Viya Prepare and Explore Data**
  - To start, we need to push data up into CAS
    - Wait, **what is CAS?**
      - CAS stands for Cloud Analytics Services. It's a multinet cloud environment founded upon microservices that can run models on a distributed system.
      - The shorter answer is that data run faster in the cloud. And CAS is SAS' tool to run models optimally on the cloud.
    - Why push it?
      - By pushing data up to CAS, we can use a suite of tools specifically designed to leverage machine learning in the cloud.
  - The process:
    - Start with a **SAS Studio Tasks** and navigate to the following:



    - Select the following settings for the **CAS Operations Task** in SAS Studio:

- But wait!
  - The code is not yet ready to be submitted, because we haven't assigned defined a CAS engine libref for our CAS in-memory data tables.
  - We can do this by adding one line to the code to the CAS Operations task.
- To start the coding process, select the **Edit button** on the code generated by SAS:



- This creates a new SAS Program. To get that statement to execute, add the following line of code before the proc casutil statement:

  libname mycaslib cas caslib=casuser;

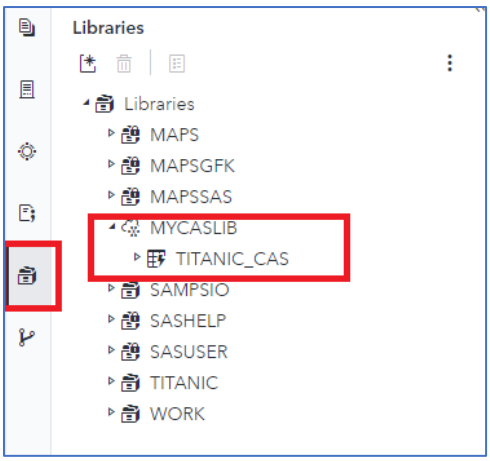
- The program now appears as follows:

Run | Cancel | Copy to My Snippets | Debug

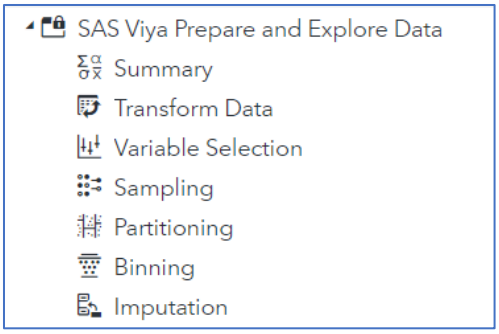
Code

```
/*
 *
 * Task code generated by SAS® Studio 5.2
 *
 * Generated on '11/16/21, 9:11 AM'
 * Generated by 'Lincoln.Groves@sas.com'
 * Generated on server 'pdcesx23104'
 * Generated on SAS platform 'Linux LIN X64 3.10.0-862.9.1.el7.x86_64'
 * Generated on SAS version 'V.03.05M0P111119'
 * Generated on browser 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTM
 * Generated on web client 'https://v4e055.vfe.sas.com/SASStudioV/main?locale=en_US&launche
 */

libname mycaslib cas caslib=casuser;

proc casutil;
    load data=TITANIC.TITANIC casout='Titanic_CAS' outcaslib='casuser';
run;
```

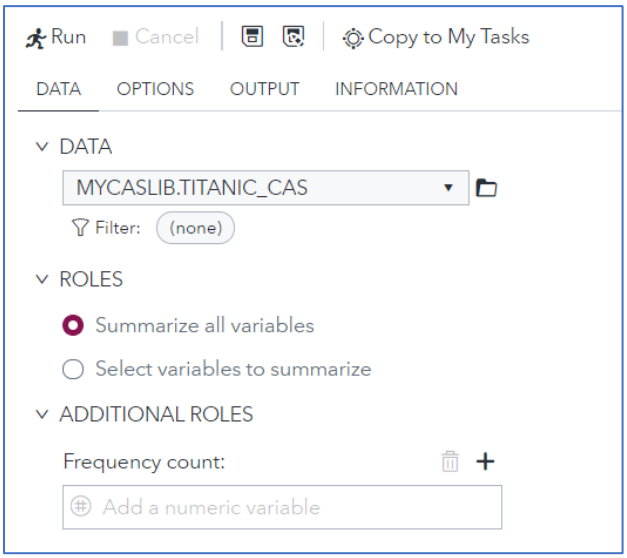
- Submit the code by clicking Run
- Again, let's ensure that the file has loaded by checking the library, which can be found under MYCASLIB:



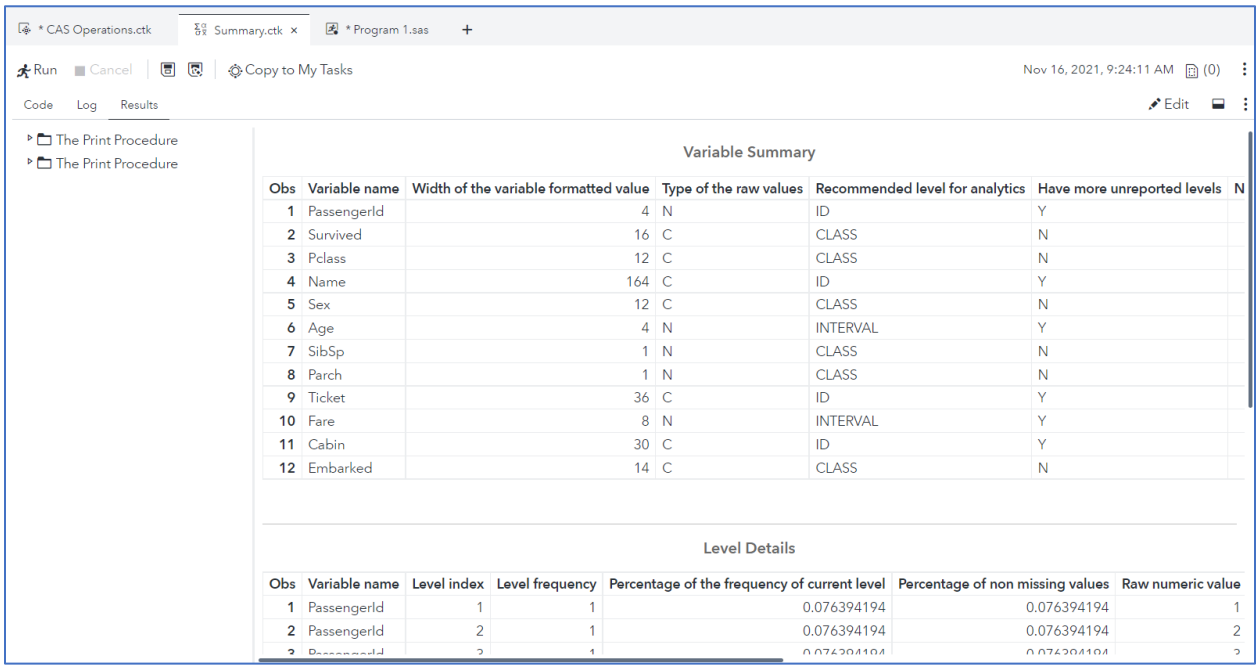
- Summary, One More Time
  - For completeness, let's examine output from the SAS Viya Prepare and Explore Data Tasks once more.
  - Start with the **SAS Viya Prepare and Explore Data** tasks:



- In particular, select **Summary**, with the following basic settings:

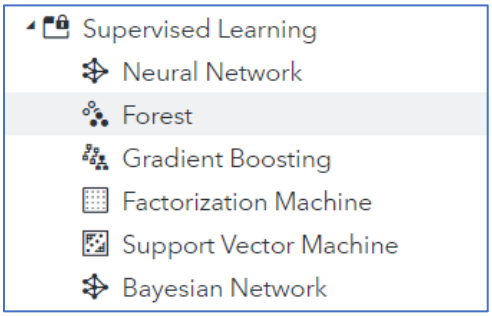


- Here is some of the summary output:

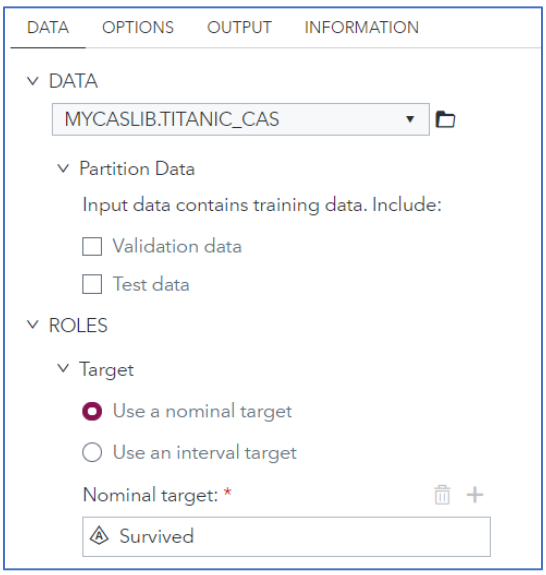


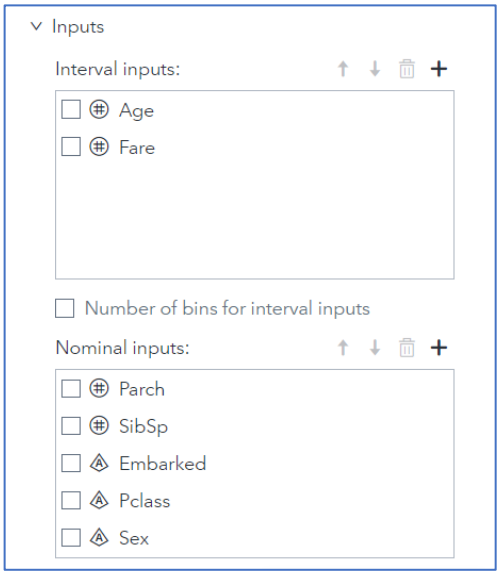
- What, if any, new information did you learn from this analysis?

- Alright, now on to **SAS Viya Machine Learning**
  - Here comes the good part!
    - One note before we get started: with coding, notice how many more options you have… but how much more time consuming the process can be!
  - The supervised learning tools available:



  - Let's do a deeper dive into **Supervised Learning using SAS Studio Tasks**
    - Model 1: **Gradient Boosting Model**
      - Use the following settings:





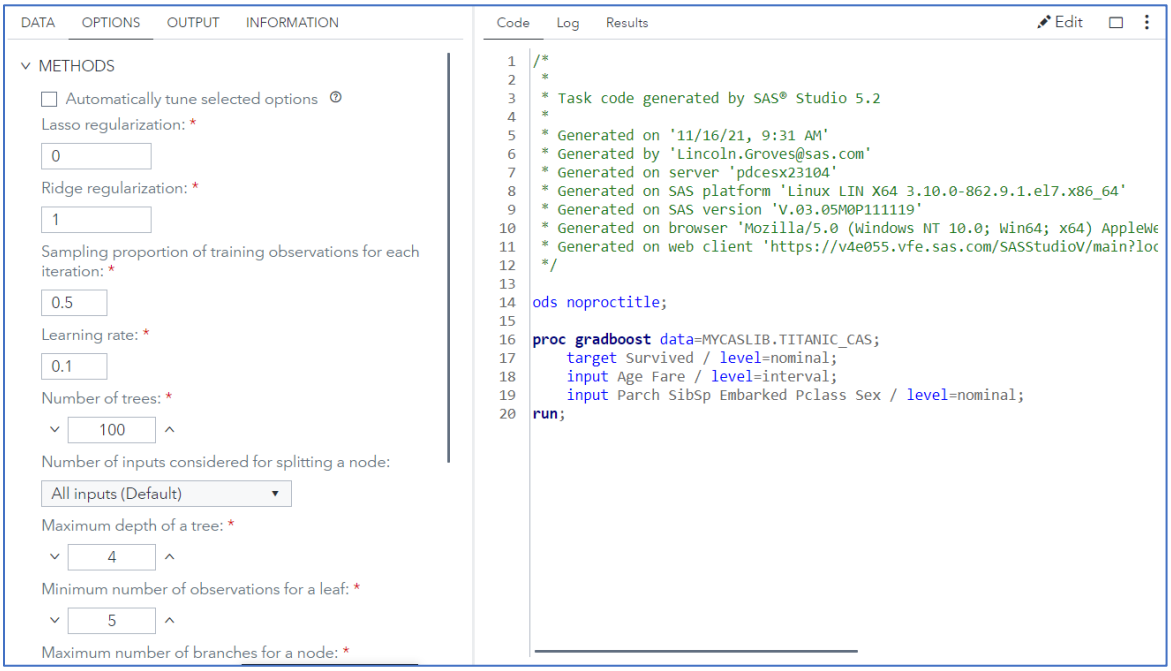
      - Submit the code and examine the **Output**:

Model Information	
Number of Trees	100
Learning Rate	0.1
Subsampling Rate	0.5
Number of Variables Per Split	7
Number of Bins	50
Number of Input Variables	7
Maximum Number of Tree Nodes	31
Minimum Number of Tree Nodes	15
Maximum Number of Branches	2
Minimum Number of Branches	2
Maximum Depth	4
Minimum Depth	4
Maximum Number of Leaves	16
Minimum Number of Leaves	8
Maximum Leaf Size	520
Minimum Leaf Size	5
Seed	106187529
Lasso (L1) penalty	0
Ridge (L2) penalty	1
Actual Number of Trees	100
Average Number of Leaves	13.39

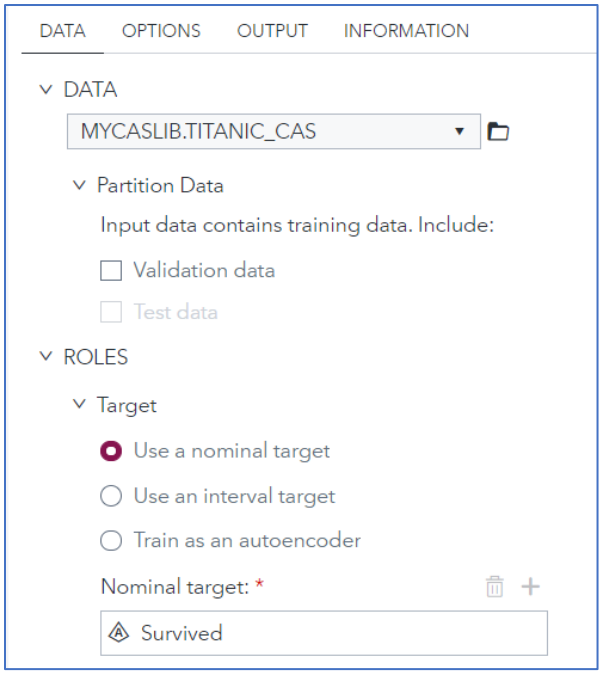
	Training
Number of Observations Read	1309
Number of Observations Used	1309

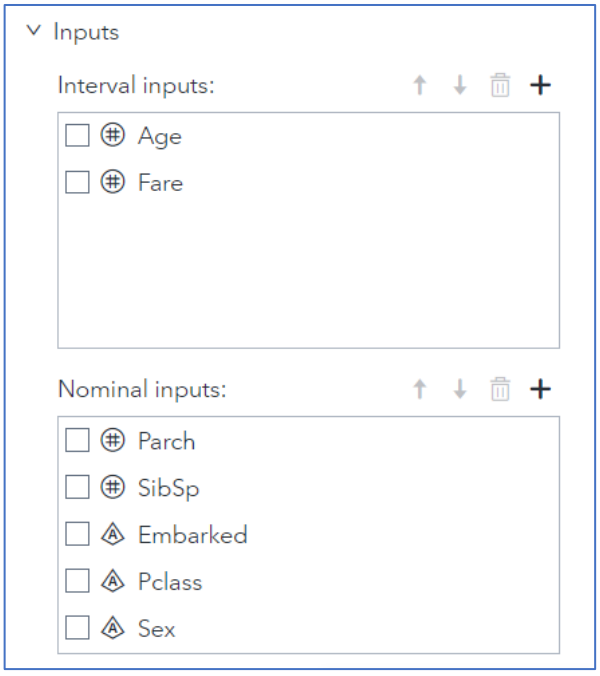
Variable Importance			
Variable	Importance	Std Dev Importance	Relative Importance
Sex	5.6400	17.6343	1.0000
Fare	4.4698	0.8795	0.7925
Age	3.7470	0.9282	0.6644
Pclass	2.2488	3.3836	0.3987
SibSp	1.2356	0.9545	0.2191
Embarked	0.8325	0.8117	0.1476
Parch	0.7256	0.7450	0.1287

- Is this a good model?
  - Examine which options you can change to refine the results.



- **Model 2: Neural Network**
  - Use the following settings:





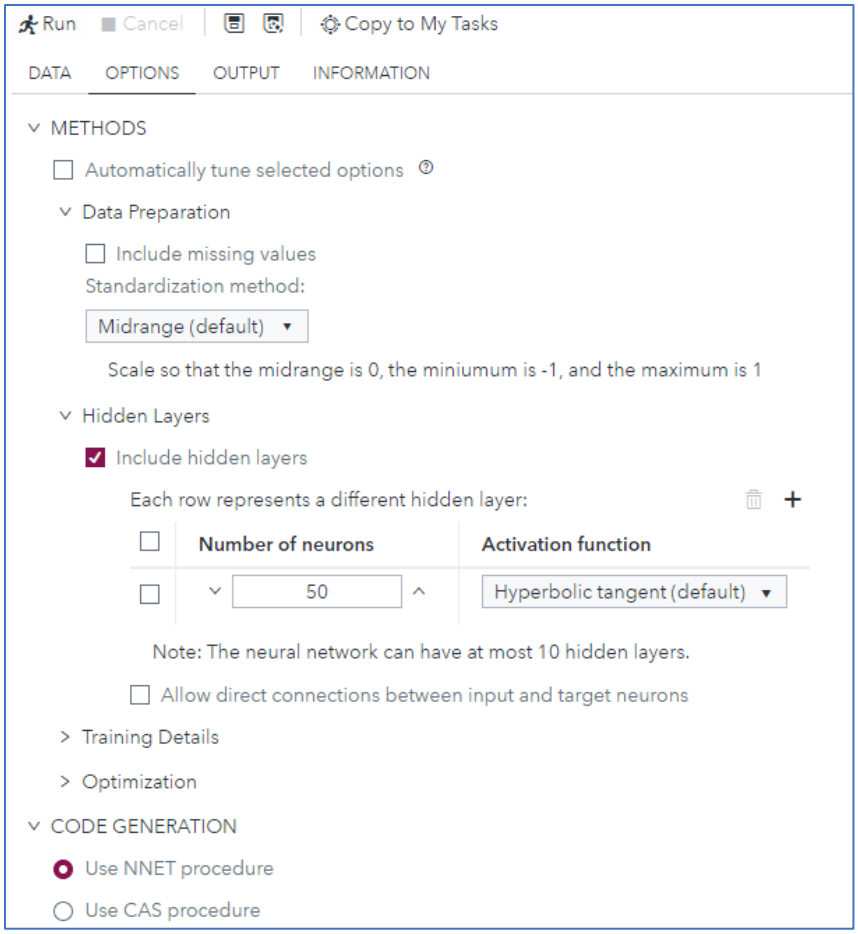
- Submit the code and examine the Output

Model Information	
Model	Neural Net
Number of Observations Used	1309
Number of Observations Read	1309
Target/Response Variable	Survived
Number of Nodes	77
Number of Input Nodes	25
Number of Output Nodes	2
Number of Hidden Nodes	50
Number of Hidden Layers	1
Number of Weight Parameters	1300
Number of Bias Parameters	52
Architecture	MLP
Seed for Initial Weight	1605445419
Optimization Technique	LBFGS
Number of Neural Nets	1
Objective Value	2.0416870809

Iteration History

Iteration Number	Objective Function	Norm of Gradient	Loss	Step Size	Norm			Fit Error
					L1	L2	Maximum	
1	2.857615	0.522868	2.774181	0	26.05427	0.834338	0.040000	0.627960
2	2.807446	0.575029	2.730891	1.875958	22.58570	0.765552	0.109539	0.382735
3	2.679234	0.329565	2.605401	1	22.06933	0.738328	0.096917	0.382735
4	2.463666	1.792478	2.280740	5	32.40929	1.829263	0.320387	0.262032
5	2.146350	0.475897	1.922871	1	38.89754	2.234781	0.394969	0.222307
6	2.118857	0.283720	1.918210	0.321574	35.20444	2.006473	0.356874	0.220779
7	2.104127	0.179213	1.896175	1	36.42326	2.079522	0.376426	0.218487
8	2.095734	0.186704	1.887008	1	36.64942	2.087262	0.381073	0.219251
9	2.071317	0.156038	1.859759	1	39.05841	2.115577	0.392531	0.210084
10	2.061105	0.132983	1.848242	1	40.33493	2.128621	0.391608	0.216960
11	2.055460	0.118111	1.835147	1	43.12972	2.203133	0.399375	0.214668
12	2.054750	0.103660	1.833985	1	43.24292	2.207648	0.402831	0.212376
13	2.054034	0.103225	1.831781	1	43.71058	2.222534	0.409630	0.212376
14	2.053518	0.090536	1.830741	1	43.92131	2.227765	0.416256	0.211612
15	2.053091	0.095639	1.829022	1	44.60839	2.240692	0.421766	0.207028
16	2.052352	0.114326	1.828766	1	45.11380	2.235861	0.431612	0.207792
17	2.052128	0.104081	1.828078	1	45.26978	2.240504	0.431587	0.207792

- Is this a good model?
  - Examine which options you can change to refine the results.

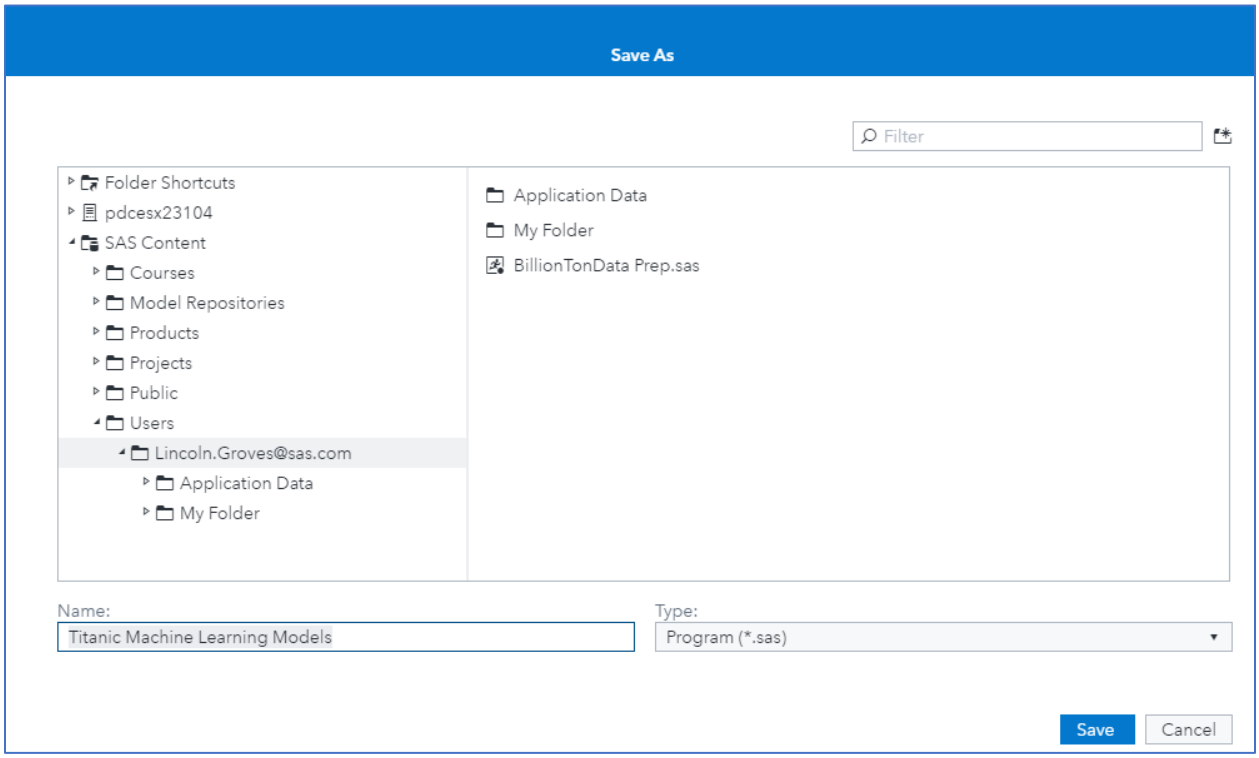


- In particular, is it better than the gradient boosting model?

- Now let's **Put all the SAS Studio Task Pieces Together**
  - Let's pause and think about everything we did with our no-code options.
  - Going back to the beginning, we:
    - Imported data
    - Checked underlying data to make sure it made sense
    - Created new variables
    - Dropped variables from the analysis
    - Created a data partition
    - Performed descriptive analysis
    - Estimated models
    - Compared models
    - Choose a champion model
  - With coding, you have complete flexibility, but you must code everything!
    - So, we really just got started with our coding work – a complete analysis would take a LOT more time before we would be ready to share results
  - But, let's start now by moving over to SAS Programming
  - One nice element of SAS Studio Tasks is that you can use SAS to write the code for you – and then you can save the code in a single file to be executed at a later date.

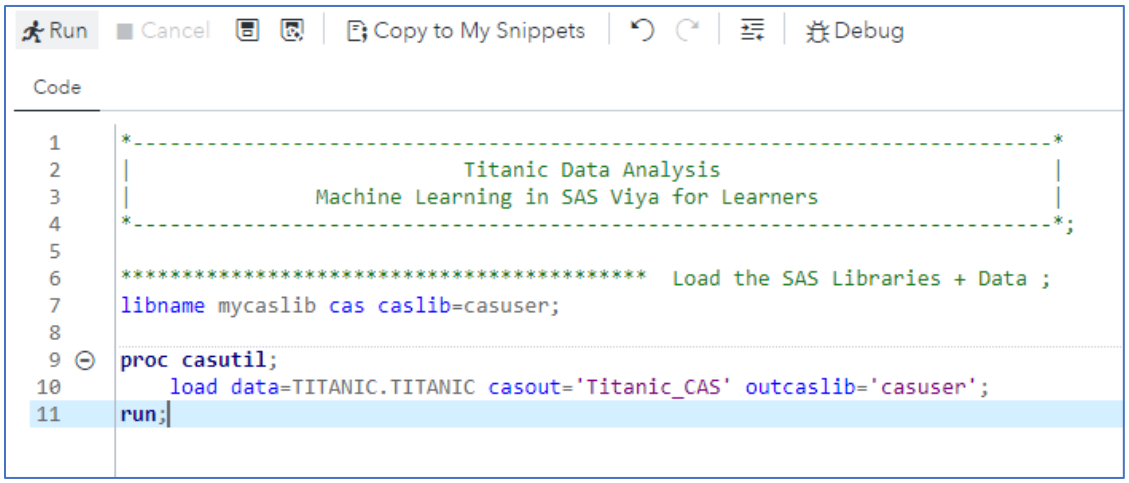
- Yes, we've seen this code being generated all along

- Assigning SAS libraries is one of the first thing we do.
  - So, let's use the library assignment program that we created earlier as the starting point for a larger program.
  - Save it to your user-space in the cloud as follows:



  - Note: your view will be different

- Add some comments to the top of the program so that you can more easily remember what you were trying to do… months from now:



- The next step in the analysis would be to examine the underlying descriptive statistics. Let's grab the code from the SAS Studio Summary Task and add some comments:

Run | Cancel | Copy to My Snippets | Debug

Code

```
*------------------------------------------------------------------------*
|                          Titanic Data Analysis                         |
|              Machine Learning in SAS Viya for Learners                 |
*------------------------------------------------------------------------*;

****************************************  Load the SAS Libraries + Data ;
libname mycaslib cas caslib=casuser;

proc casutil;
    load data=TITANIC.TITANIC casout='Titanic_CAS' outcaslib='casuser';
run;


*------------------------------------------------------------------------*
|                       Examine Descriptive Statistics                   |
*------------------------------------------------------------------------*;
ods noproctitle;
libname _tmpcas_ cas caslib="CASUSER";

proc cardinality data=MYCASLIB.TITANIC_CAS outcard=_tmpcas_.varSummaryTemp
        out=_tmpcas_.levelDetailTemp;
run;

proc print data=_tmpcas_.varSummaryTemp label;
    var _varname_ _fmtwidth_ _type_ _rlevel_ _more_ _cardinality_ _nmiss_ _min_
        _max_ _mean_ _stddev_;
    title 'Variable Summary';
run;

proc print data=_tmpcas_.levelDetailTemp (obs=20) label;
    title 'Level Details';
run;

proc delete data=_tmpcas_.varSummaryTemp _tmpcas_.levelDetailTemp;
run;

libname _tmpcas_;
```

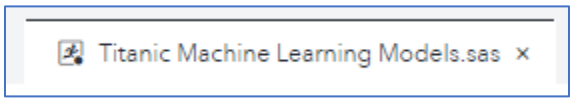
- Continuing on – lets add the Gradient boosting code now – even though we'd likely spend a LOT of time creating new variables, partitions, and, in general, refining the data before modeling:

```
proc delete data=_tmpcas_.varSummaryTemp _tmpcas_.levelDetailTemp;
run;

libname _tmpcas_;


*------------------------------------------------------------------------*
|                          Gradient Boosting Model                       |
*------------------------------------------------------------------------*;
ods noproctitle;

proc gradboost data=MYCASLIB.TITANIC_CAS;
    target Survived / level=nominal;
    input Age Fare / level=interval;
    input Parch SibSp Embarked Pclass Sex / level=nominal;
run;
```

- Finally, let's also stick in that Neural Network model:

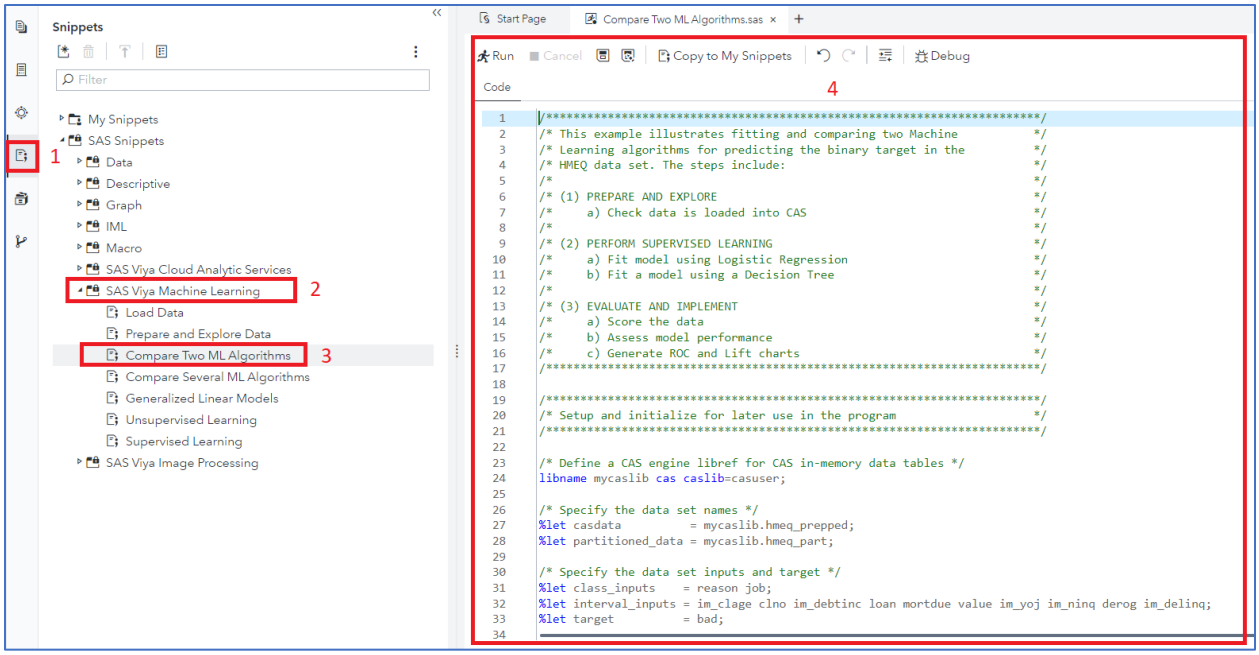
```
50
51
52  *------------------------------------------------------------------------*
53  |                          Neural Network Modeling                       |
54  *------------------------------------------------------------------------*;
55  ods noproctitle;
56  libname _tmpcas_ cas caslib="CASUSER";
57
58  proc nnet data=MYCASLIB.TITANIC_CAS;
59      target Survived / level=nominal;
60      input Age Fare / level=interval;
61      input Parch SibSp Embarked Pclass Sex / level=nominal;
62      hidden 50;
63      train outmodel=_tmpcas_._Nnet_model_;
64      optimization regL2=0.1;
65  run;
66
67  proc delete data=_tmpcas_._Nnet_model_;
68  run;
69
70  libname _tmpcas_;
71
```

- Then, of course, let's save the SAS Program – because we don't want to lose all our hard work!
    - A saved file does not have an asterisk before the file name:



- Finally, let's take it from the top and submit the whole program at once.
    - If this is your first time, welcome to your first full-fledged SAS Program!

- Final thoughts on the code-based approach
    - Notice how much longer it takes to produce results with code? And, technically, we didn't come even close to replicating what we created in SAS VA and SAS Model Studio.
    - One big item we omitted was comparing the results of our two ML algorithm. And, to be honest, this was intentional, because it's a LOT of code to process.
    - But for those interested coders, I leave you with one final resource, which is the **Compare Two ML Alogorithm** snippet found here:



- You can modify these 182 lines of coding bliss to fit our Titanic data analysis – or simply run the code, as is, and see the output for another data set and research question.
- Regardless, know that SAS has the tools to support you on nearly all your analytical adventures!

Part IV: Learning, continued

- We made it!
- Here are some free SAS resources to continue your learning
    - Free eLearning
        - https://www.sas.com/en_us/learn/academic-programs/resources/free-sas-e-learning.html
    - LiveWeb Courses
        - https://www.sas.com/professor-workshops
    - Certification series
        - www.sas.com/certguidedprep
- Find these resources here in one location here: https://www.sas.com/en_us/learn/academic-programs.html
    - SAS Educator Portal
        - https://www.sas.com/en_us/learn/academic-programs/educators.html
    - SAS Student Skill Builder
        - https://www.sas.com/en_us/learn/academic-programs/students.html
- And check out the free software options here:
    - https://www.sas.com/en_us/software/viya-for-learners.html
    - https://www.sas.com/en_us/software/on-demand-for-academics.html