From Code to No-Code
Alternative Paths for Data Analysts

Lincoln Groves
Manager, Analytical Education

#ExploreSAS

Machine Learning in SAS Model Studio with iLink Telecom, Inc.

SAS On-the-Job Activity

Purpose

This activity exposes students to the wonderful world of machine learning in SAS Model Studio. Assuming the role of Retention Specialist on their first day at iLink Telecom, Inc., students will learn how to create a SAS Model Studio project, alter metadata, execute pre-built modeling pipelines, and incorporate their favorite SAS®9 and open-source code into their modeling.

SAS Software

We will use SAS Viya for Learners 4.0 (VFL4.0) software in this activity. Students can also use VFL3.5 but will not be able to complete any of the tasks that require autotuning. Why? Because autotuning isn't a feature in VFL3.5.

Industry Alignment

This SAS On-the-Job aligns with the telecom industry – or those in an industry seeking to model churn (i.e., the probability that a customer chooses another provider). However, the materials are applicable to anyone seeking to unlock the power of SAS Model Studio – and looking for a high-level overview to get started.

Table of Contents

Activity Notes and Requirements

Learning Objectives

This activity exposes students to the wonderful world of machine learning in SAS Model Studio. More specifically, students will use SAS Model Studio and learn how to do the following:

- Create a project and load data
- Clean up variables and modify the data partition
- Build pipelines from SAS templates
- Run autotuned models
- Use the automatic pipeline generation option
- Incorporate both SAS®9 and open-source models
- Examine models within a pipeline
- Examine models across pipelines
- Use SAS insights to help with storytelling

Estimated Completion Time

This On-the-Job activity consists of nine tasks. Assuming that each task takes approximately 15 minutes to complete, students can expect to complete this SAS On-the-Job in about two to two and one-half hours.

Experience Level

This activity is designed for individuals new to SAS Viya for Learners. Students can continue their SAS journey by visiting one of the learning paths outlined in *Appendix C: Recommended Learning*.

Prerequisite Knowledge

Software

No prior experience is needed with SAS Viya for Learners. In fact, SAS On-the-Job activities are created for learners new to SAS Software.

Content Knowledge

Students should have some base-level familiarity with

- machine learning models
- autotuning.

Additional Notes

Autotuning is available only in VFL4.0. Thus, users must use VFL4.0 to complete the autotuning tasks in this SAS On-the-Job activity. For those still using VFL3.5, there is still a lot of good material in this activity – so all is not lost.

Required Setup

For an optimal experience, students should complete all the steps, in order, as outlined in this SAS On-the-Job.

Task 1: Create a SAS Model Studio Project and Load Data

Learning Objectives

- Understand the use case.
- Access SAS Viya for Learners.
- Create a new SAS Model Studio project.
- Load data.

Estimated Time of Completion

This task will take approximately 10-15 minutes to complete.

Task 1: Create a SAS Model Studio Project and Load Data

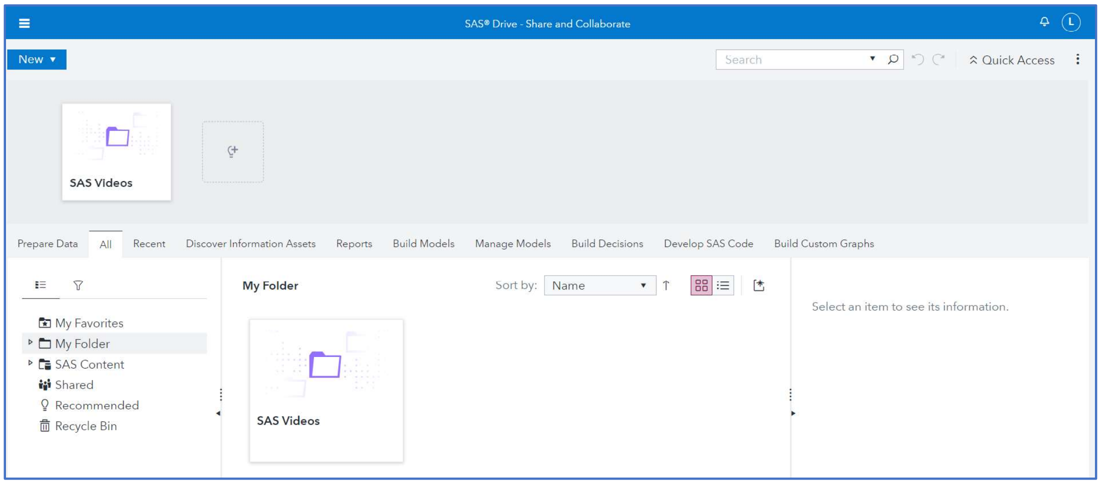
Welcome to your first day at iLink Telecom, Inc! As a Retention Specialist, you'll help us use machine learning models to retain our top wireless phone customers. Your onboarding will take several months, but we'd like to use your time today to expose you to the main tool that you'll use in your analyses: SAS Model Studio!

SAS Model Studio is a great no/low-code tool that we use to examine historical data on customer churn (i.e., the customer leaves us for another cell phone provider). By using tools readily available in SAS Model Studio, we can estimate a series of machine learning models and then choose the model that best estimates churn in our sample. With that model in hand, we can then predict churn for our set of current customers – where their outcome is unknown. Marketing will then help us decide which customers are worth retaining with special promotions – think new phones or special deals – or when we should do nothing.

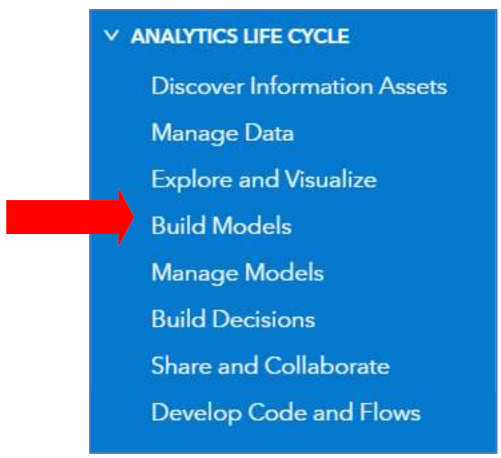
For our lessons today, I will walk you through several ways to run machine learning models in SAS Model Studio, which includes using SAS templates, autotuning, automatic pipeline generation, and even incorporating SAS®9 and open-source code. The goal here isn't to have you understand exactly what is going on in the modeling or to predict new cases. Instead, it's to expose you to the wide modeling universe – so that you can follow-up with more detailed studying on your own.

I hope that you're as excited as I am… so let's get started!

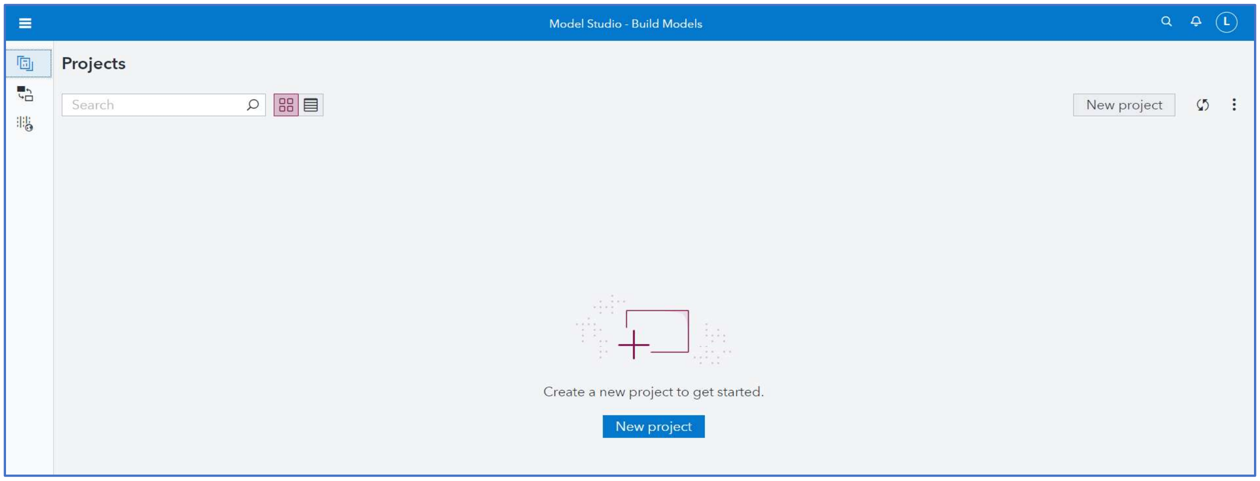
1. Log in to SAS Viya for Learners.
2. You will be brought to SAS Drive, which is your starting point in SAS Viya for Learners – and displays something like this:



3. Click the **Applications** menu (☰) on the upper left corner of the SAS Drive page. Select **Build Models** from the **ANALYTICS LIFE CYCLE** options.

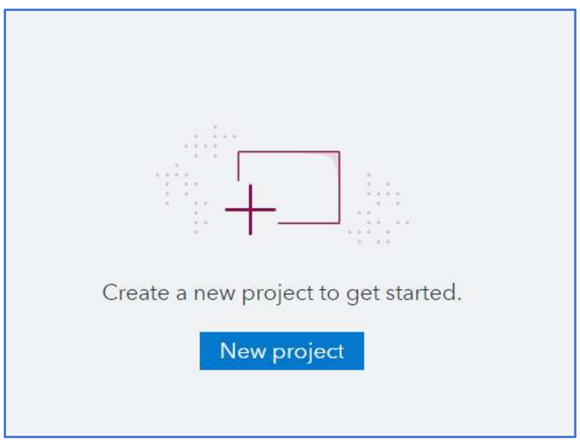


4. The **Model Studio Projects** page is now displayed.



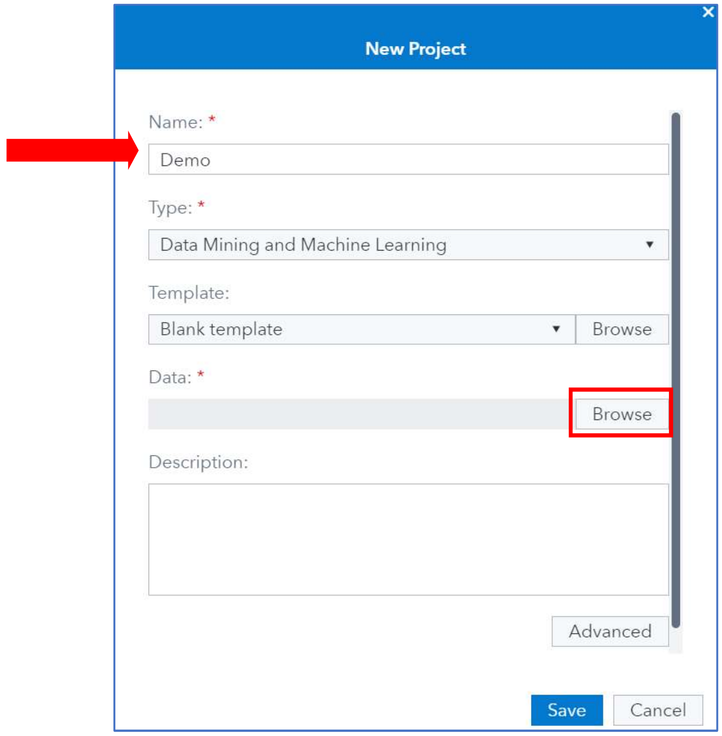
5. From the Model Studio Projects page, you can view existing projects, create new projects, and access the Exchange. Model Studio projects can be one of three types: *Forecasting projects*, *Data Mining and Machine Learning projects*, and *Text Analytics projects*.

6. Select **New Project** in the center of the Projects page – or the icon on the right, if you've been in SAS Model Studio before:

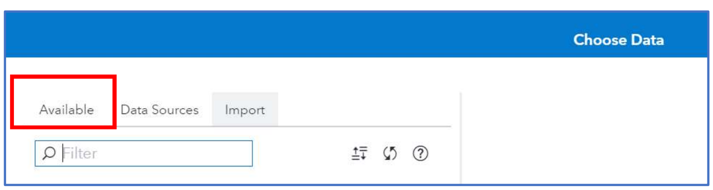


or

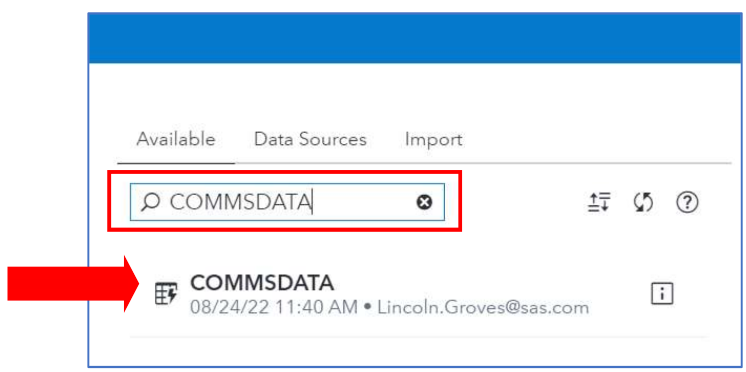
7. Enter **Demo** as the name in the **New Project** window. Leave the default type of **Data Mining and Machine Learning** and select **Browse** in the **Data Source** field.



8. Within the Choose Data window, ensure that the **Available** data sets tab is selected:



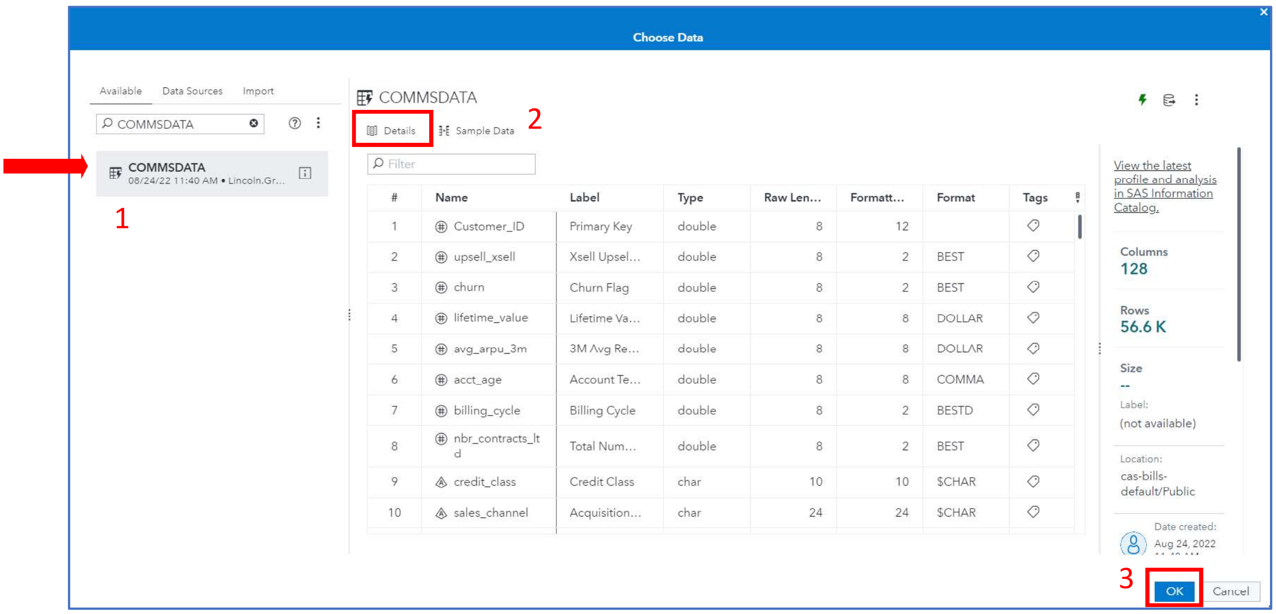
9. In the **Filter**, type **COMMSDATA**. You should see our data set under the **Available** options.



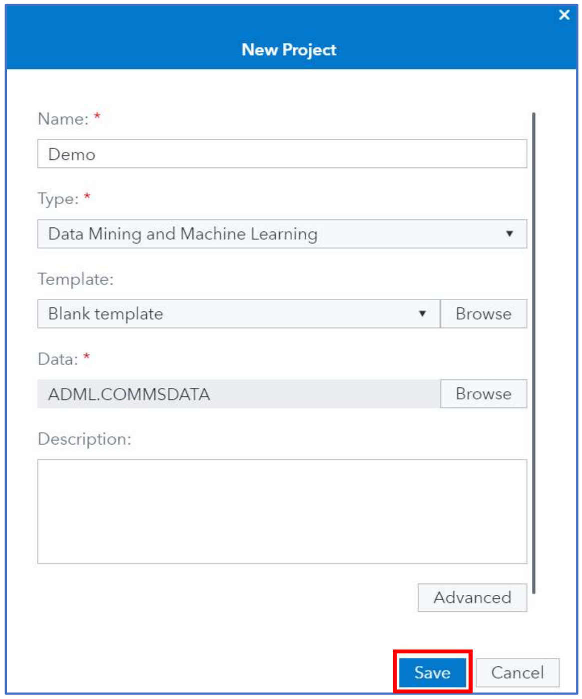
Note: *If there are multiple copies of COMMSDATA available, then just select the first one.*

10. Click **COMMSDATA**, which is our company's main database created for tracking customers. There are ***a lot of variables*** but – for today – we'll cover just a few of them. Again, the key for today is to become more comfortable with the SAS software – and the various machine learning tools in SAS. The other details can come later.

11. All that stated, some knowledge of the data can be useful. So, briefly explore the **Details** tab to become more familiar with the types of data available in **COMMSDATA** and then click **OK**:

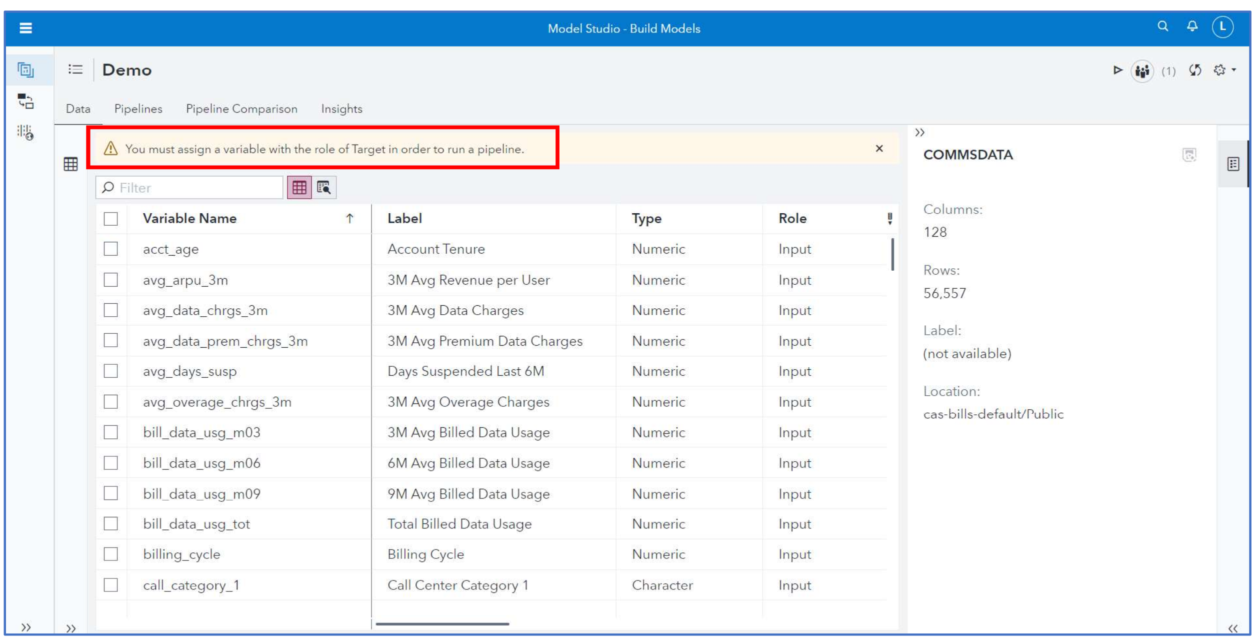


12. Return to the main **New Project** window and click **Save**.

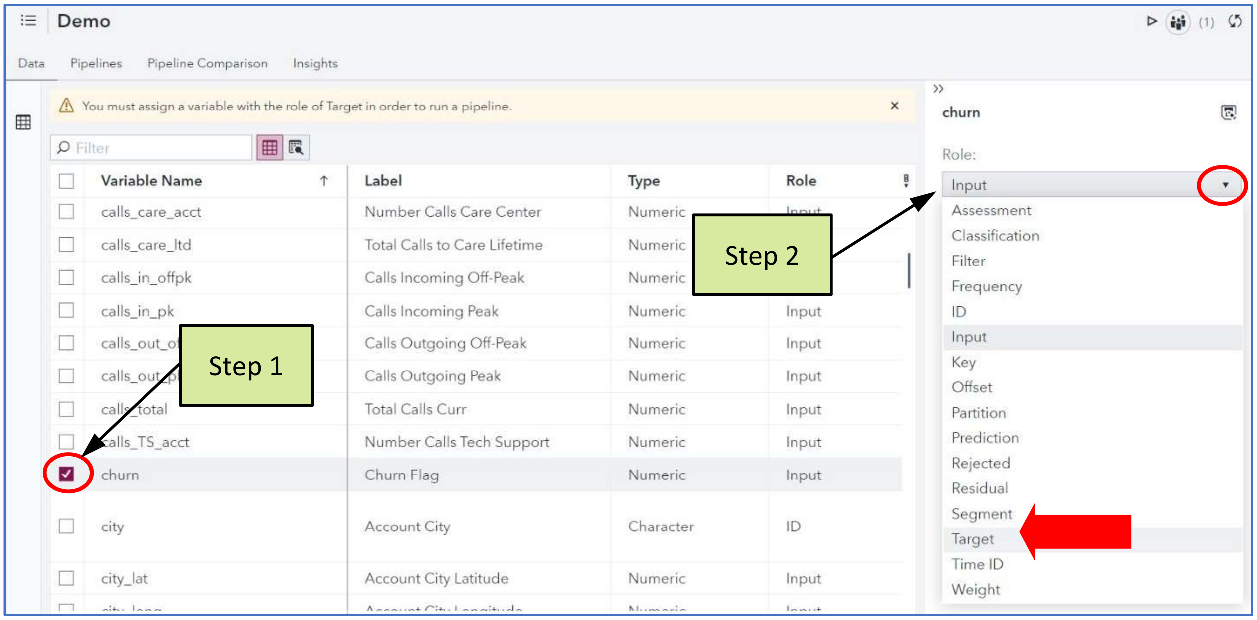


13. After you create your new project, SAS Model Studio takes you to the **Data** tab of your new project page. Here, you can adjust data source variable names, labels, type, role, and level assignments. The Data tab enables you to modify variable assignments and manage global metadata. And, yes, properly defined metadata is super important!

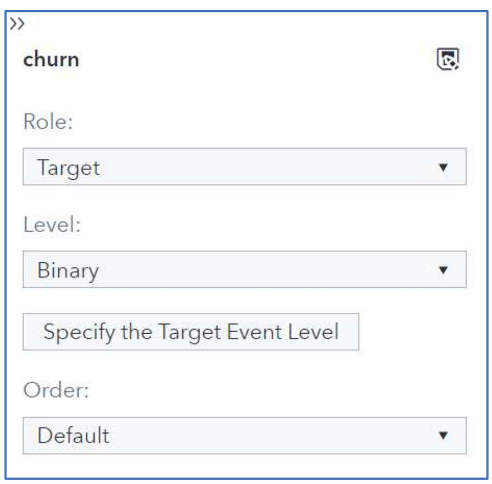
14. Fun fact: when a project is created, a target variable ***must be assigned*** to run a pipeline. Thus, expect to see the following message at the start of a new project:



15. In our analysis, **churn** is the target variable. Again, and put simply, churn indicates that the customer left iLink Telecom for another provider. Assign **churn** as the target variable by doing the following: in the variables window, select **churn** (Step 1). Then in the right pane, select **Target** under the **Role** property (Step 2).



16. That target variable, **churn**, will now have the following metadata:



17. With **churn** established as our target, we can now turn our attention to variable cleanup and data partitioning. Onward!

Task 2: Variable Cleanup and Modifying the Data Partition

Learning Objectives

- Reject irrelevant variables.
- Change the data partition.
- Run a pipeline.

Estimated Time of Completion

This task will take approximately 10-15 minutes to complete.

Task 2: Variable Clean-up and Modifying the Data Partition

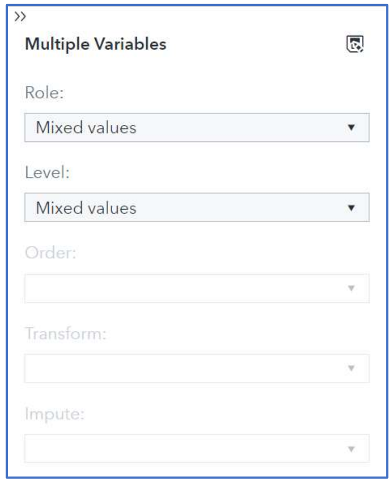
Our metadata needs a bit more work before we plunge into modeling. Moreover, we'll be running machine learning models, so we should ensure that our data partitions are the way we want them. Data wrangling is for the detail-oriented, so let's get it done!

1. To start, ensure that the **Demo** project is open, and that **churn** (our target variable) is not selected from the previous section. If **churn** is still selected, uncheck it:

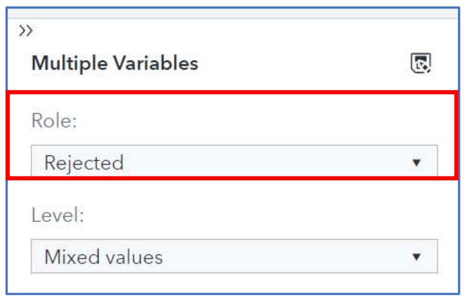
☐	Variable Name ↑	Label	Type	Role
☐	churn	Churn Flag	Numeric	Target

2. Before modifying the data partition, we'd like to exclude select variables from the analysis – because they are not relevant to our upcoming modeling. To start, select the following variables – by clicking the check mark before their names:
    - **city**
    - **city_lat**
    - **city_long**
    - **data_usage_amt**
    - **mou_onnet_6m_normal**
    - **mou_roam_6m_normal**
    - **region_lat**
    - **region_long**
    - **state_lat**
    - **state_long**
    - **tweedie_adjusted**

3. From the right pane, you can view the roles for the multiple variables – which appears as:

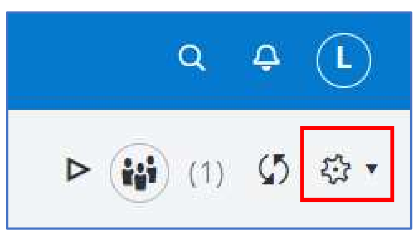


4. Change the role to **Rejected** for all variables listed above.

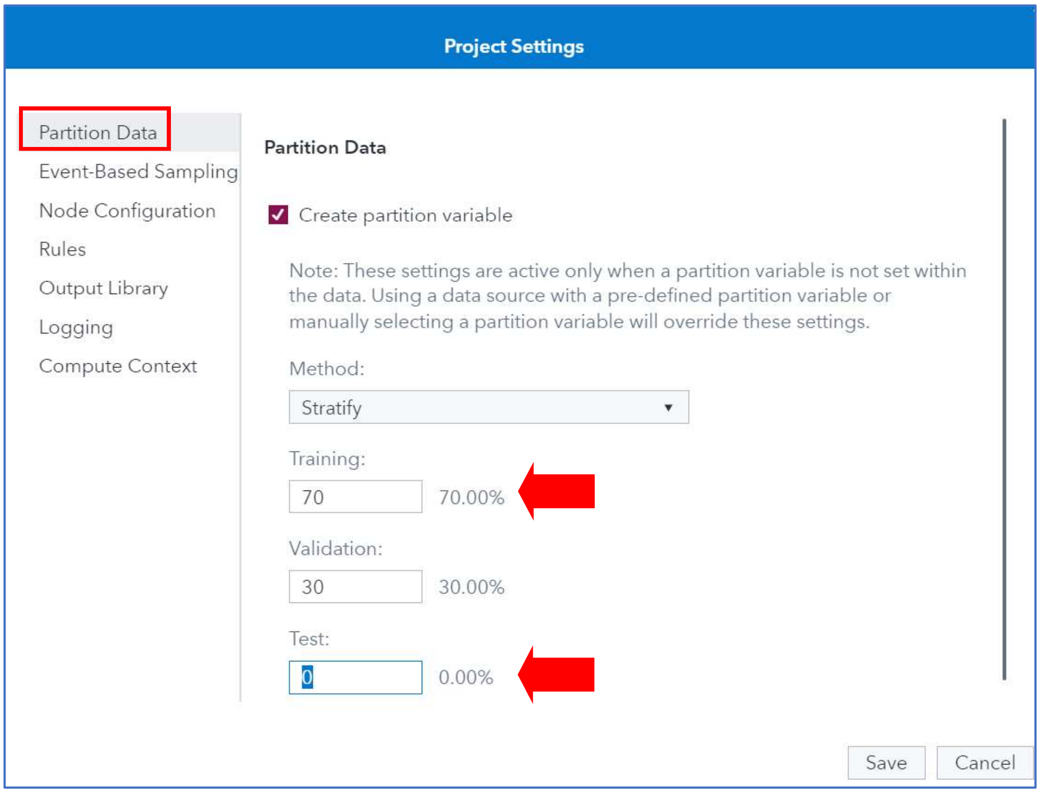


5. Data cleaned. Yay!

6. Now let's adjust the sampling. Start by clicking ⚙ (**Settings**) in the upper right corner of the window.

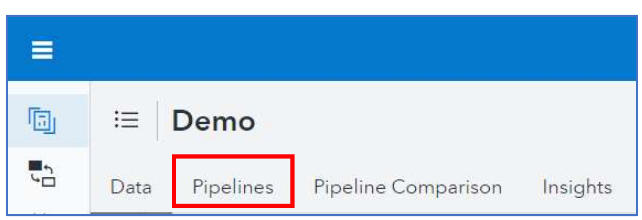


7. Model Studio uses partitioning by default. However, we can change the data partition percentage before running our first pipeline. The steps:
   a. Select **Project settings**.
   b. With **Partition Data** selected, change the **Training** percentage to **70**.
   c. Finally, set the **Test** percentage to **0**.



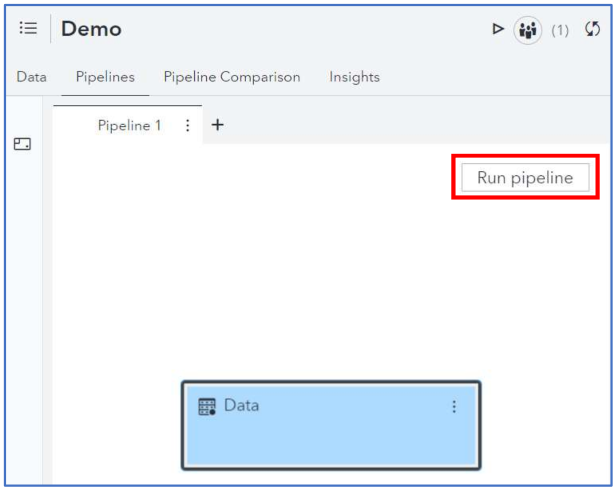
Note: These settings can be edited only if no pipelines in the project have been run. After the first pipeline is executed, the partition tables are created, and partition settings cannot be changed.

8. Click **Save** to lock in the new partition settings.
9. Let's run our first pipeline. Click the **Pipelines** tab in our Demo project:



Note: On the Pipelines tab, you can create, modify, and run pipelines. Each pipeline has a unique name and optional description.

10. Click **Run Pipeline** to run the pipeline:



11. The green check mark in the node indicates that it ran without error:

12. And, in this example, this means that the data were loaded, and the partition was successfully created. Mission accomplished! That is, the mission of variable clean-up and data partition modification. But there are many other missions ahead!

Task 3: Building a Pipeline from a Basic Template

Learning Objectives

- Create new pipeline.
- Access SAS templates.
- Run a pipeline and review results.

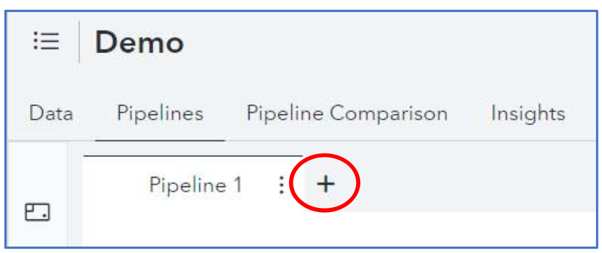
Estimated Time of Completion

This task will take approximately 10-15 minutes to complete.

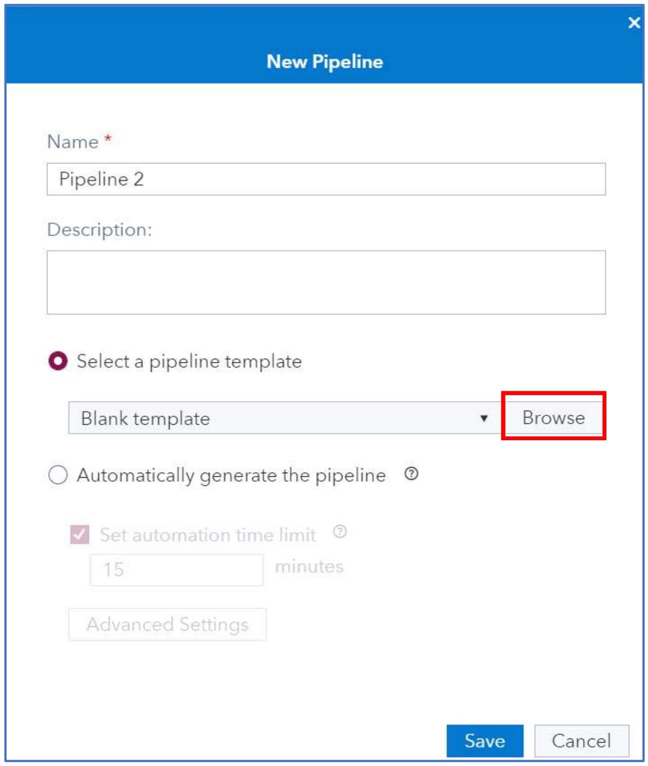
Task 3: Building a Pipeline from a Basic Template

Although it is nice to build your own pipelines from scratch – and you'll get there one day – it is often convenient to start from a template that represents best practices in building predictive models. SAS Model Studio comes with a nice set of default templates to create new pipelines. We'll start simple by showing you the default pipelines – with a new pipeline built from a basic template for class target.

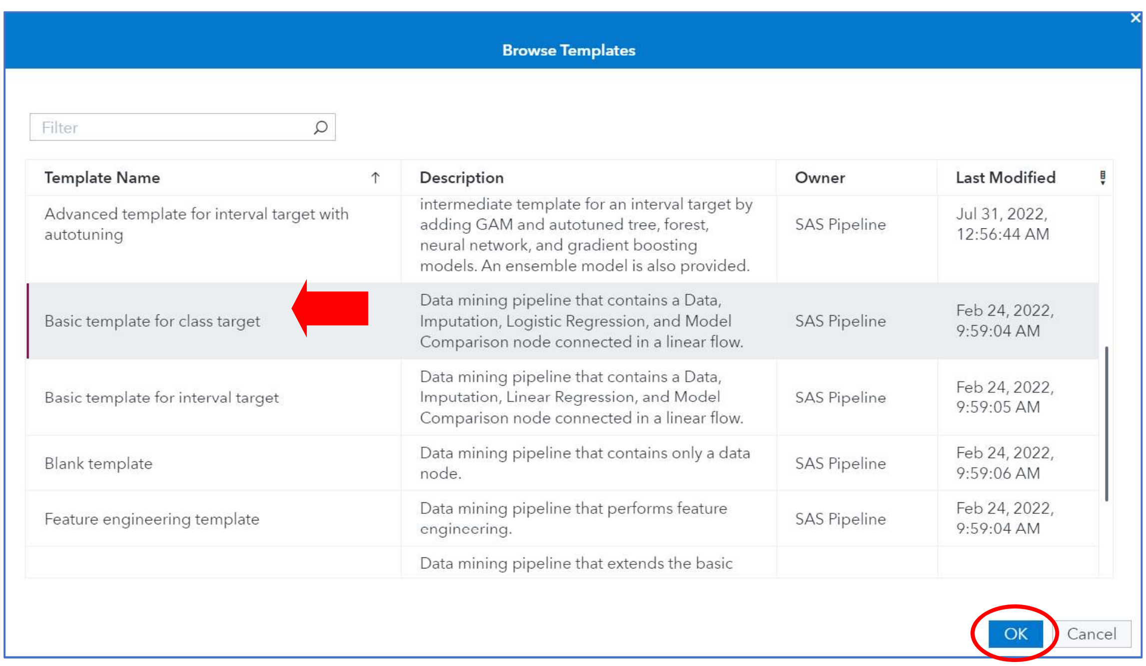
1. Click + next to the current pipeline tab in the upper left corner of the canvas.



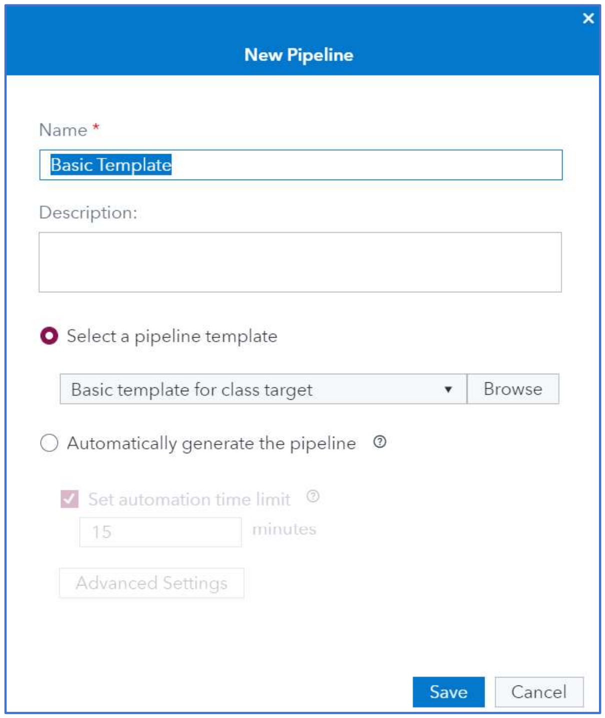
2. In the **New Pipeline** window, select **Browse** under **Select a pipeline template**.



3. In the **Browse Templates** window, select **Basic template for class target**. Click **OK**.

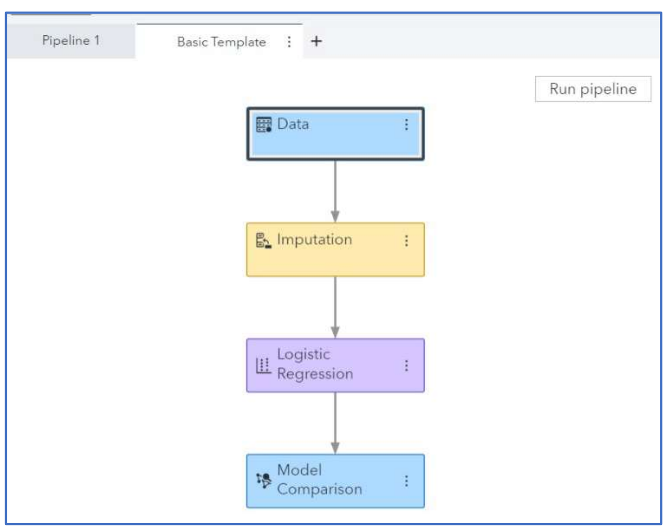


4. In the New Pipeline window, name the pipeline **Basic Template** – because, you know, it's pretty basic:



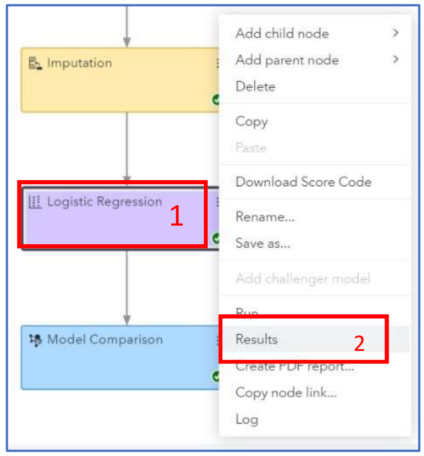
5. Click **Save**.

6. The *Basic template for class target* contains a simple linear flow with the following nodes: Data, Imputation, Logistic Regression, and Model Comparison. You can add additional nodes to this pipeline by right-clicking the existing nodes – or dragging and dropping from the Nodes pane. We'll get to that in a bit.

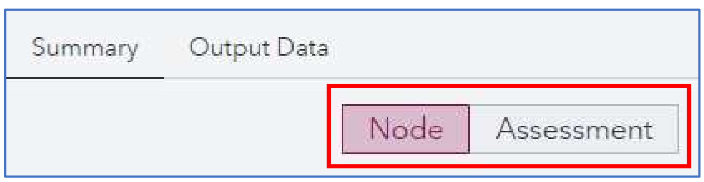


7. Before getting wild and running more models, let's simply click **Run pipeline** in the upper right corner of the pipeline.

8. After the pipeline has successfully run, right-click the **Logistic Regression** node and select **Results**.



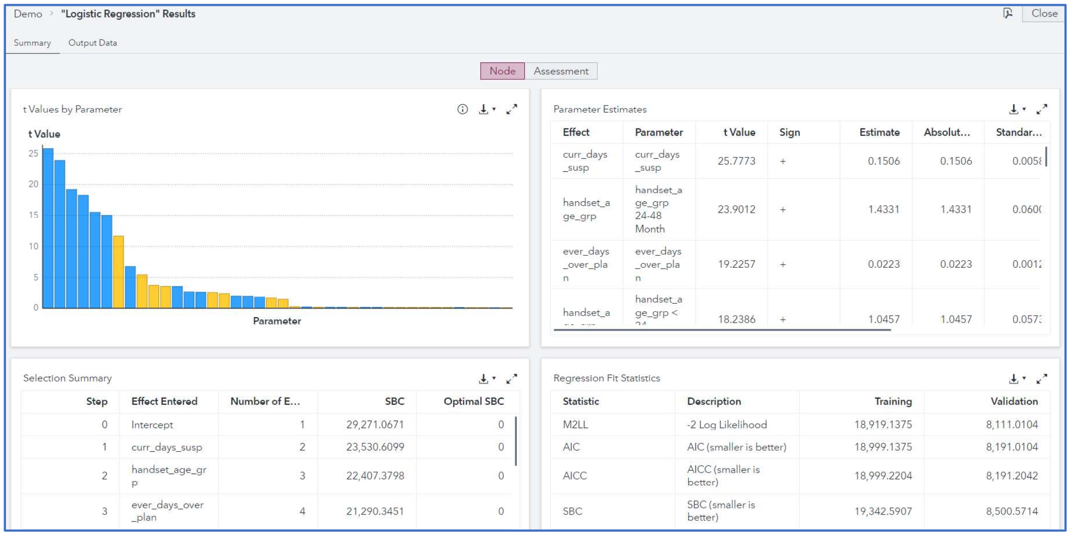
9. The Results window contains two important tabs at the top: one for **Node** results and one for **Assessment** results.



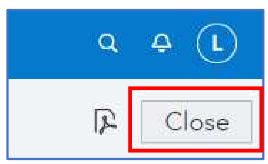
10. Here are some of the windows included under the **Nodes** tab:
    - t-values by Parameter table
    - Parameter Estimates table
    - Selection Summary table

11. And here are some of the windows under the **Assessment** tab:
    - Lift Reports plots
    - Fit Statistics table
    - Output

12. Explore the results as you see fit. Really, explore the space.

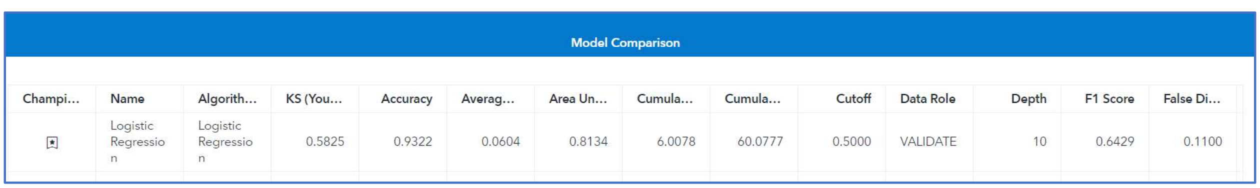


13. When you are satiated with statistics, close the Results window by clicking **Close** in the upper right corner of the window.



14. Right-click the **Model Comparison** node and select **Results**.

15. Click to expand the **Model Comparison** table. Unless otherwise specified, the Kolmogorov-Smirnov statistic (KS) selects the champion model for a class target.



Model Comparison

Champi...	Name	Algorith...	KS (You...	Accuracy	Averag...	Area Un...	Cumula...	Cumula...	Cutoff	Data Role	Depth	F1 Score	False Di...
	Logistic Regression	Logistic Regression	0.5825	0.9322	0.0604	0.8134	6.0078	60.0777	0.5000	VALIDATE	10	0.6429	0.1100

Note: *The Model Comparison node is always added by default when any model is contained in the pipeline. If the pipeline contains only a single model, the Model Comparison node summarizes performance of this one model.*

16. Close out of the window to exit the maximized view: .

17. Click **Close** to close the **Model Comparison Results** window. And we're now finished with this task – great work!

Task 4: Autotuning Using a Single Pipeline Node

Learning Objectives

- Add a node to a pipeline.
- Change node settings.
- Run an autotuned supervised learning model.

Estimated Time of Completion

This task will take approximately 10-15 minutes to complete. And just a reminder: autotuning is not available in SAS VFL3.5.

Task 4: Autotuning Using a Single Pipeline Node

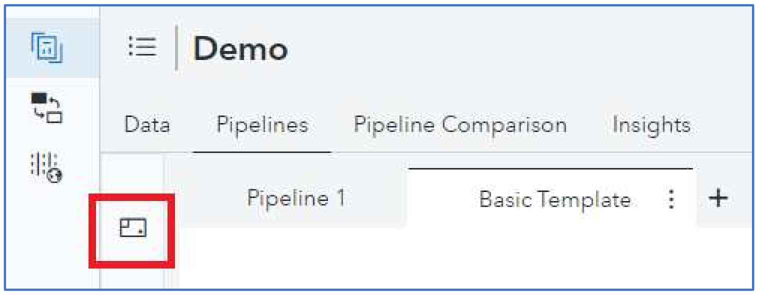
Thus far in our quest, we've cleaned the data, altered sampling proportions, and run a logistic regression model. But I promised some machine learning models, didn't I? Well, this is your section! And we'll start by adding a single autotuned model to Basic Template.

Wait a second, you might think: what is autotuning?

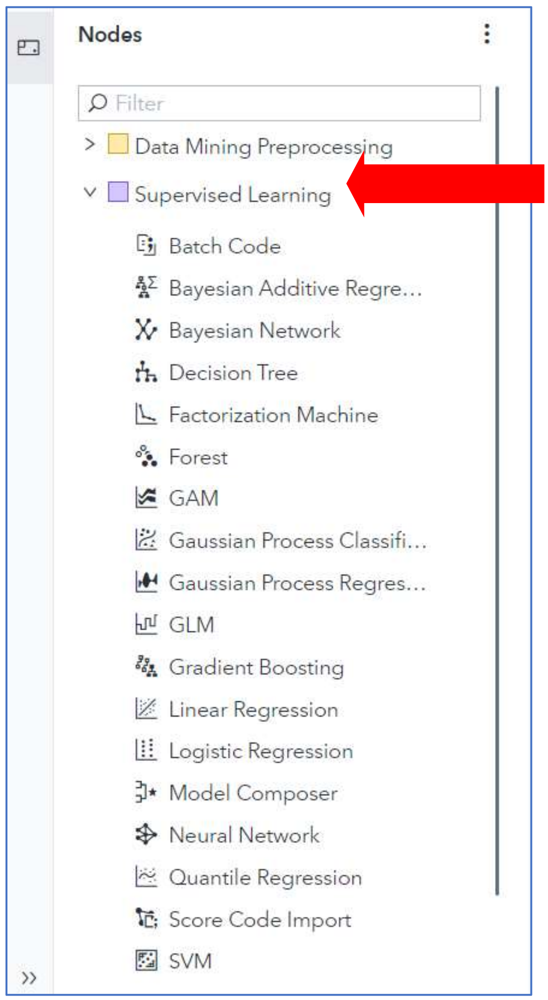
Well, autotuning is an advanced machine learning tool that automatically searches many competing predictive models, simultaneously. Autotuning – also known as hyperparameter tuning – can help automate the model selection process by identifying the optimal parameter setting for a wide range of machine learning models, including decision trees, random forests, gradient boosting, neural networks, support vector machines, and factorization machines. Put simply, autotuning really is machine learning – in that your machine will run many, many models for you and choose the best one based on your specifications (or the defaults). Good stuff!

Let's get started and run our first autotuned model.

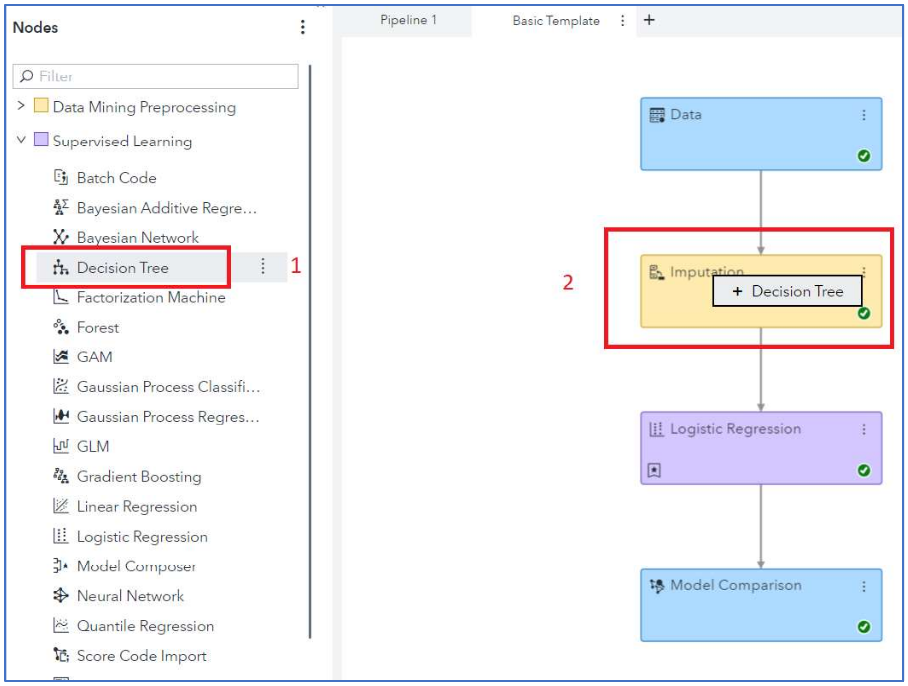
1. Return to the **Basic Template** pipeline.
2. Click the **Nodes** icon in the left pane:



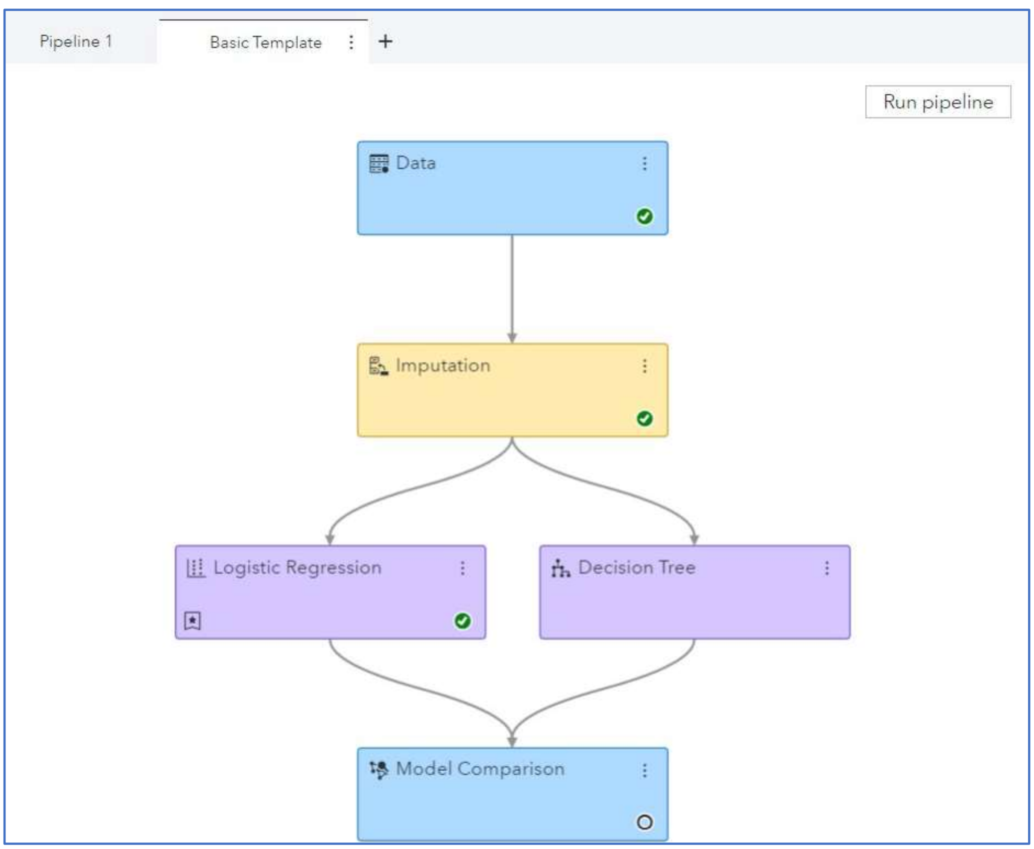
3. Then, expand the **Supervised Learning** options:



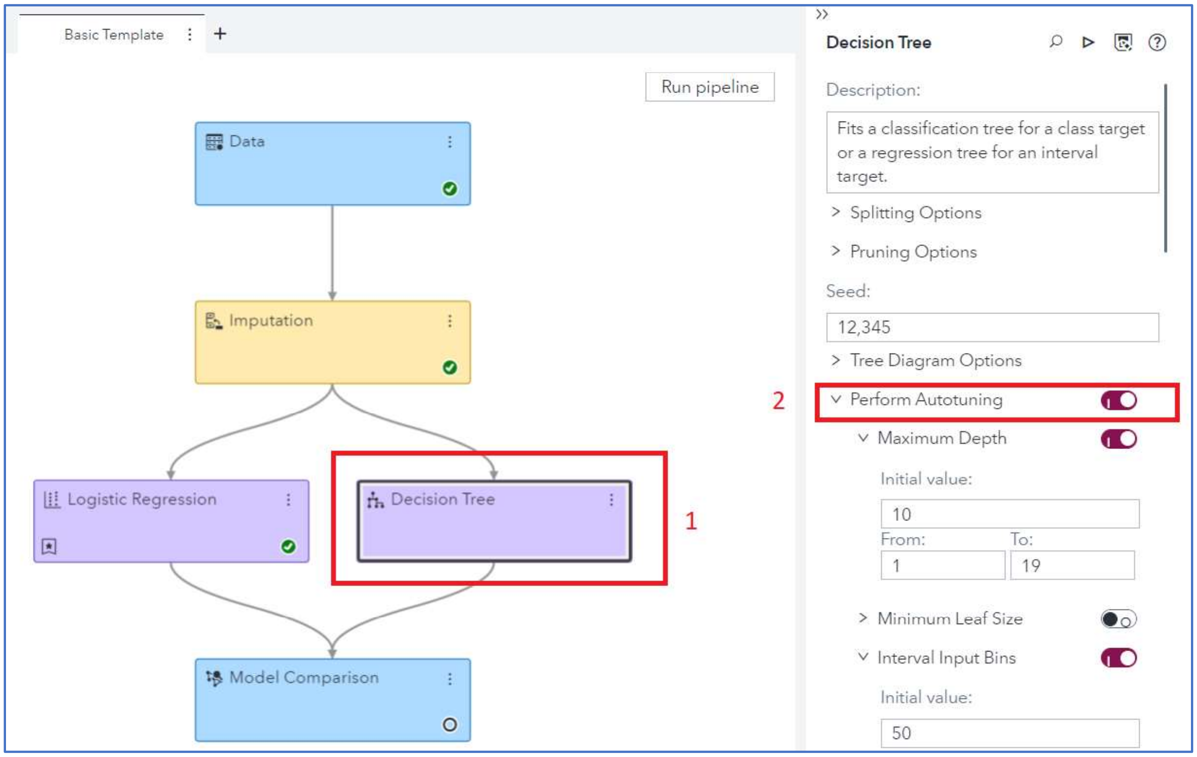
4. Select **Decision Tree** and drag-and-drop it on top of the **Imputation** node in the main pipeline:



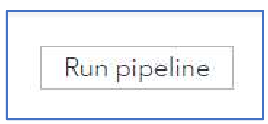
5. The **Basic Template** pipeline now appears as follows:



6. Now let's add the autotuning option to the Decision Tree. Select the **Decision Tree** node and then click the  symbol to examine the modeling options on the right pane. Select the **Perform Autotuning** option:

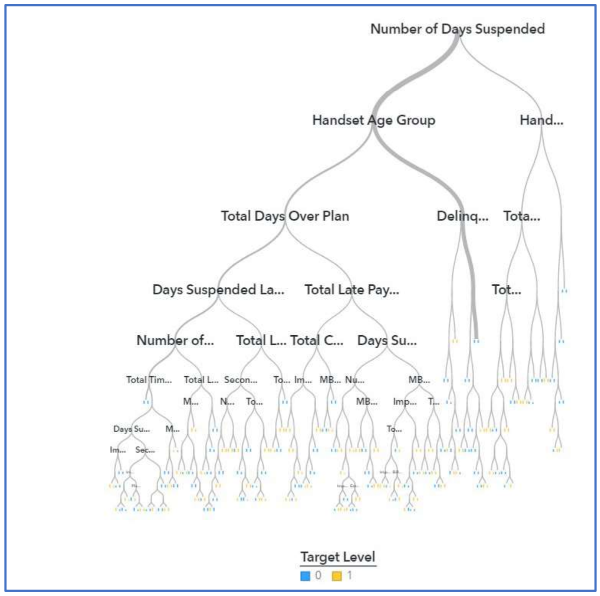


7. With autotuning now turned on for the Decision Tree – we now have a model that is truly "learning" in our pipeline. Click **Run pipeline**.



8. For a decision tree with autotuning, SAS Model Studio will automatically run series of decision trees with varying hyperparameter thresholds for modeling decisions such as Maximum Depth, Minimum Leaf Size, Interval Input Bins, Grow Criteria, and so on. After the pipeline is completed, let's examine the results for the "best" decision tree.

9. When the node is finished running, right-click the **Decision Tree** node and select **Results**. Select to expand the **Tree Diagram** under the **Node** tab, which appears akin to the following:



Note: *Your view can be different due to different underlying training, validation, and testing samples.*

10. Close out of the window to exit the maximized view.

11. Click **Close** to exit the **Decision Tree Results** window.

12. Finally, let's see whether our Autotuned Decision Tree is a better predictive model than our default Logistic Regression model. Return to **Basic Template** and examine the **Results** from the **Model Comparison** node. Again, results can be found by right-clicking **Model Comparison** and then selecting **Results**

13. As displayed by the **Model Comparison** table, the **Logistic Regression** model still does a better job of predicting churn in our analysis than our **Autotuned Decision Tree** model, as determined by the KS (Youden) statistic:

Model Comparison

Champi...	Name	Algorith...	KS (You...	Accuracy	Averag...	Area Un...	Cumula...
★	Logistic Regression	Logistic Regression	0.5825	0.9322	0.0604	0.8134	6.0078
	Decision Tree	Decision Tree	0.5704	0.9389	0.0564	0.8072	6.0009

Hmmm. Looks like the fancy stuff isn't always "better".

14. Click **Close** to close the **Model Comparison Results** window. You just ran your first autotuned model in SAS Viya – even if it wasn't a good one! Congrats!

Task 5: Advanced Pipeline with Autotuning

Learning Objectives

- Add another default pipeline to your SAS Model Studio project.
- Examine the results from seven autotuned models.

Estimated Time of Completion

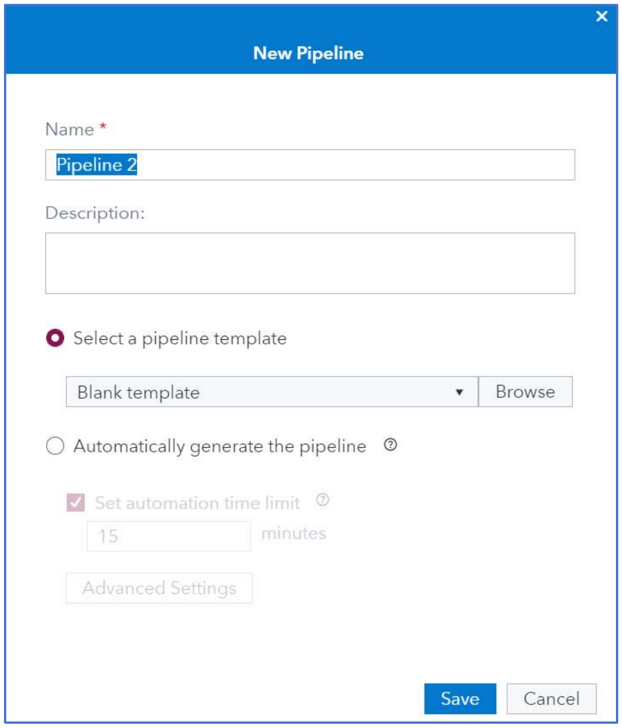
This task will take approximately 10-15 minutes to complete.

Task 5: Advanced Pipeline with Autotuning

In the last section, we added a single autotuned model to the pipeline. But, we can do so much more! Because what is better than running a single autotuned machine learning model? Using a pre-built SAS pipeline template to run a series of autotuned models! Let's make it happen!

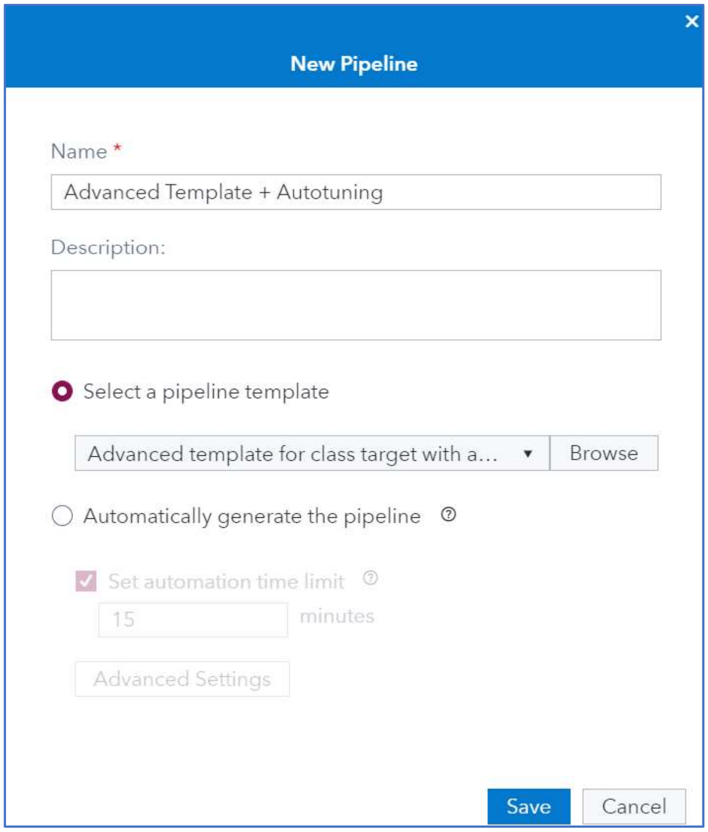
1. To get started, click the **Add new pipeline** button to the right of your **Basic Template** pipeline.

2. The **New Pipeline** window appears as follows:

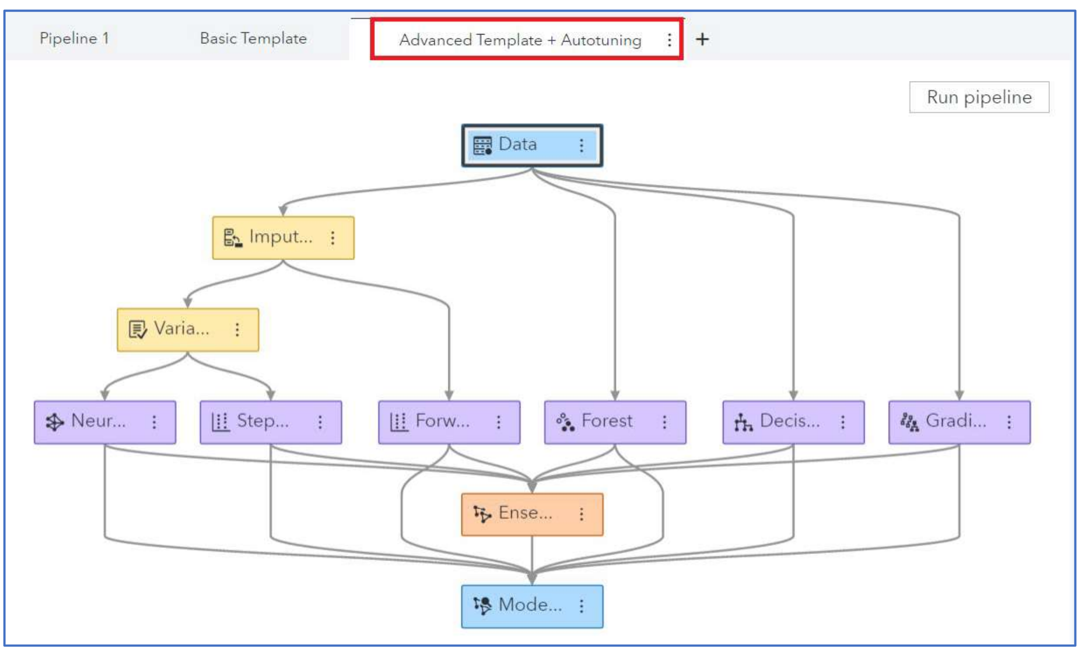


3. Let’s make a few changes. To start, change the pipeline name to **Advanced Template + Autotuning**.
4. Under **Select a pipeline template**, click **Browse** and then select **Advanced template for class target with autotuning**. Click **OK**.

5. After your new pipeline appears as follows, click **Save**:

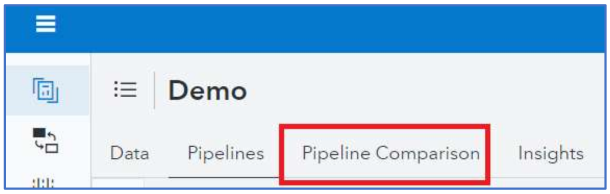


6. The new pipeline appears as follows:

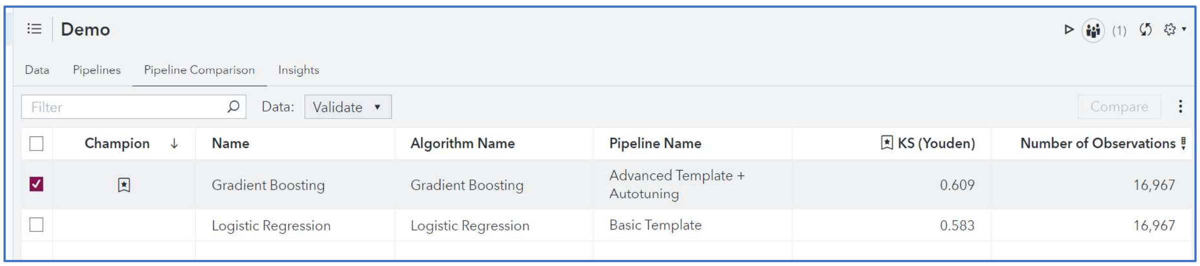


Dun, dun, dun… that's a lot of models!

7. Click **Run pipeline**. *Note: this pipeline could take some time to run (~10 minutes).*
8. After the pipeline is finished running, select **Pipeline Comparison**:



9. Did the autotuned models perform better than the models run in the Basic Template? Here are my findings:



10. Yes, it appears that the autotuned Gradient Boosting model outperforms the best model from the Basic Template, as determined by the KS (Youden) statistic. Progress!
11. But, if you thought we were done with the modeling options, you were wrong: we're only a little over halfway there!

Task 6: Automatic Pipeline Generation

Learning Objectives

- Let SAS produce a pipeline specifically generated for your data.
- Run an automatically generated pipeline.

Estimated Time of Completion

This task will take approximately 10-15 minutes to complete.

Task 6: Automatic Pipeline Generation

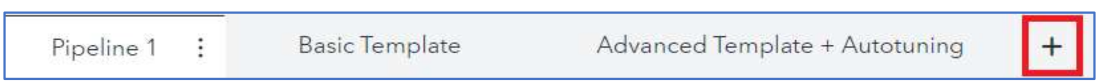
In our last task, we used a default pipeline provided by SAS Model Studio to run a whole bunch of machine learning models. But did you know that SAS Model Studio will examine your data and use AI technology to generate a pipeline specific to your data set?

No?

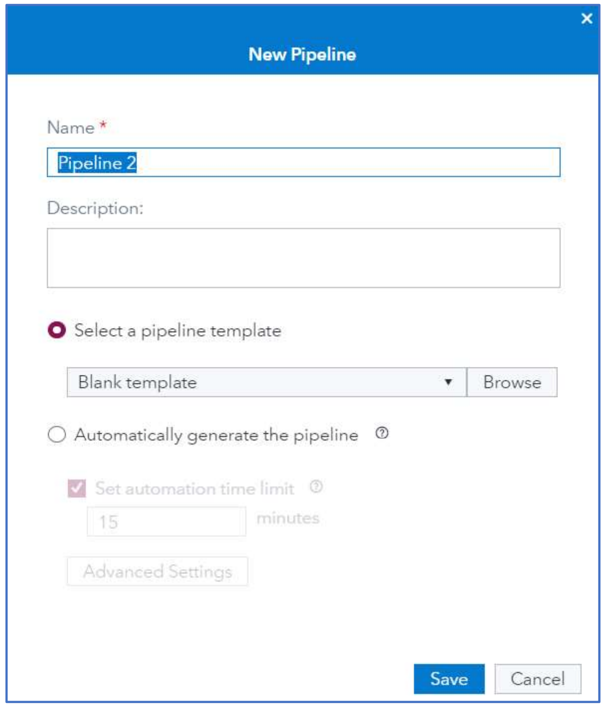
Consider yourself educated! And to further that education, we'll explore that exciting tool, called automatic pipeline generation, in this task.

1. Make sure that you are on the Pipelines tab:

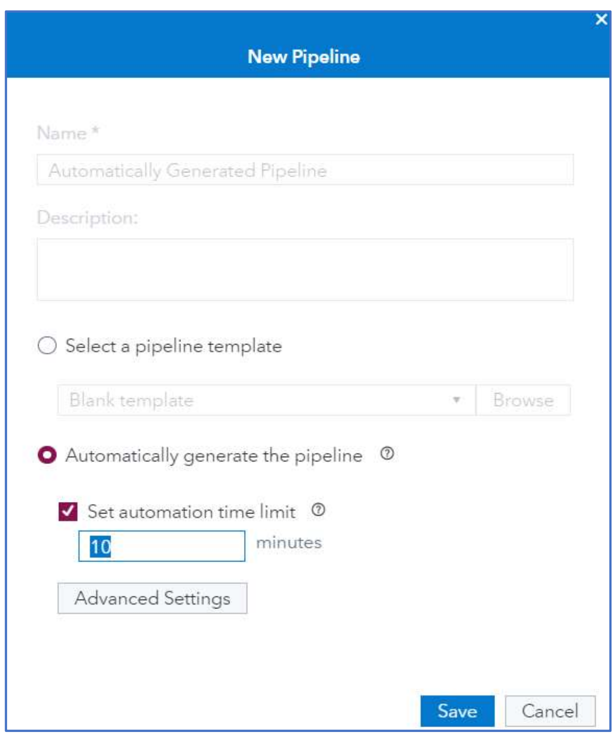
2. Next, click the **Add new pipeline** button to the right of your **Advanced Template + Autotuning** pipeline.



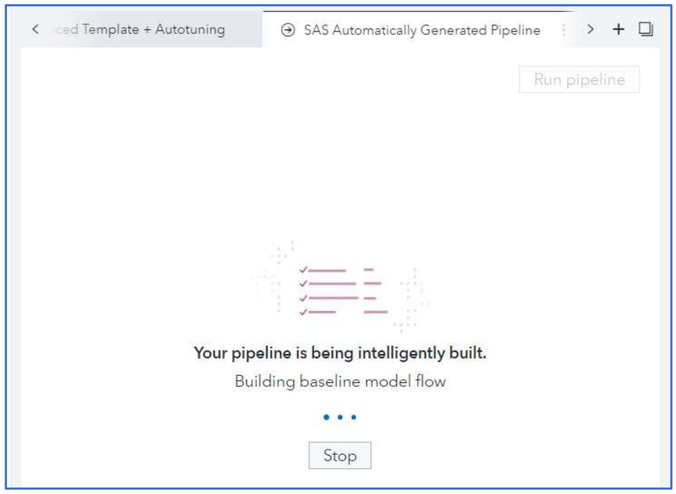
3. The **New Pipeline** window appears as follows:



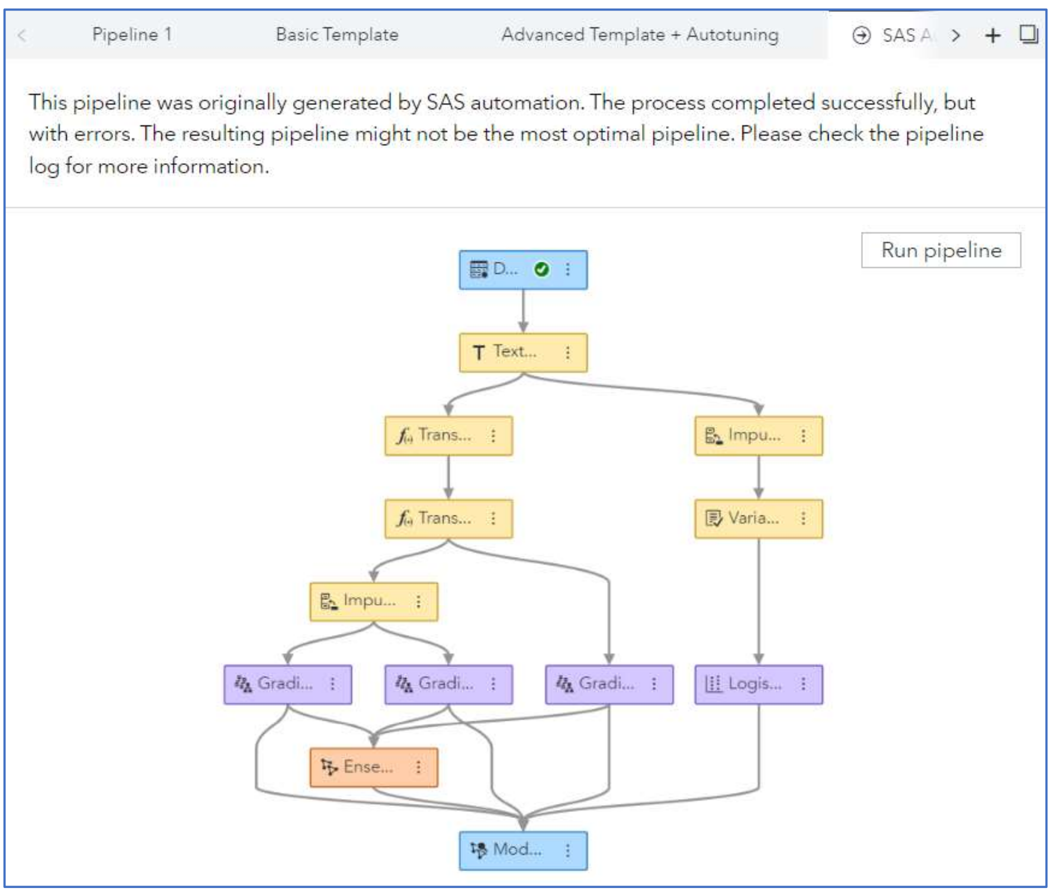
4. Let's start by renaming the pipeline to **Automatically Generated Pipeline**.
5. Select **Automatically generate the pipeline** and **Set automation time limit** to 10 minutes.
6. When the **New Pipeline** window appears with the new settings, click **Save**:



7. It will take up to 10 minutes for the pipeline to generate, as SAS analyzes your data, determines whether any data preprocessing is required, and then suggests a series of machine learning models to run.
8. A screenshot of your machine thinking:



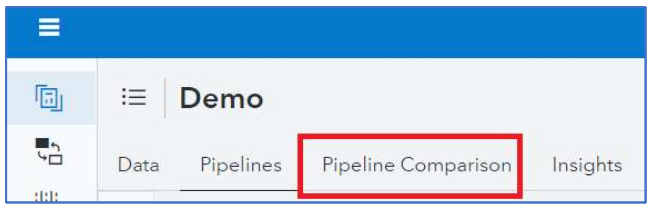
9. When completed, a new pipeline could appear as follows:



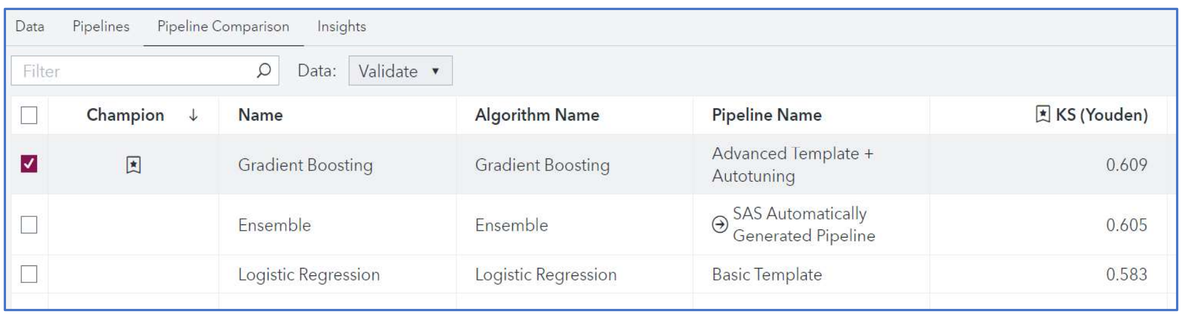
Note: Your pipeline might be different. Actually, it probably is. And that's okay. Also, examine the message above from SAS – i.e., that the "process was completed successfully, but with errors". Increasing the automation time limit would likely correct that issue, but we will continue the exposition of this tool.

10. The next step is simply to accept the SAS defaults and to click **Run pipeline**.

11. After the automatically generated pipeline is finished, select **Pipeline Comparison**:



12. Did the autotuned models perform better than our other models? Here are our findings:



	Champion	Name	Algorithm Name	Pipeline Name	KS (Youden)
☑	★	Gradient Boosting	Gradient Boosting	Advanced Template + Autotuning	0.609
☐		Ensemble	Ensemble	SAS Automatically Generated Pipeline	0.605
☐		Logistic Regression	Logistic Regression	Basic Template	0.583

13. So, no, it appears that the autotuned Gradient Boosting model from our last task still outperforms our other options, based on the KS (Youden) statistic. But, hey, at least the results were better than the default logistic regression model!

Task 7: Incorporating Your Favorite SAS®9 Models

Learning Objectives

- Incorporate the SAS Code node.
- Submit SAS®9 regression code.
- Compare model results.

Estimated Time of Completion

This task will take approximately 10-15 minutes to complete.

Task 7: Incorporating Your Favorite SAS®9 Models

This fancy new SAS Viya thing might be new to many of you. And you might think, "I loved the classic SAS®9 models that I created using the SAS windowing environment – can't I use some of those?"

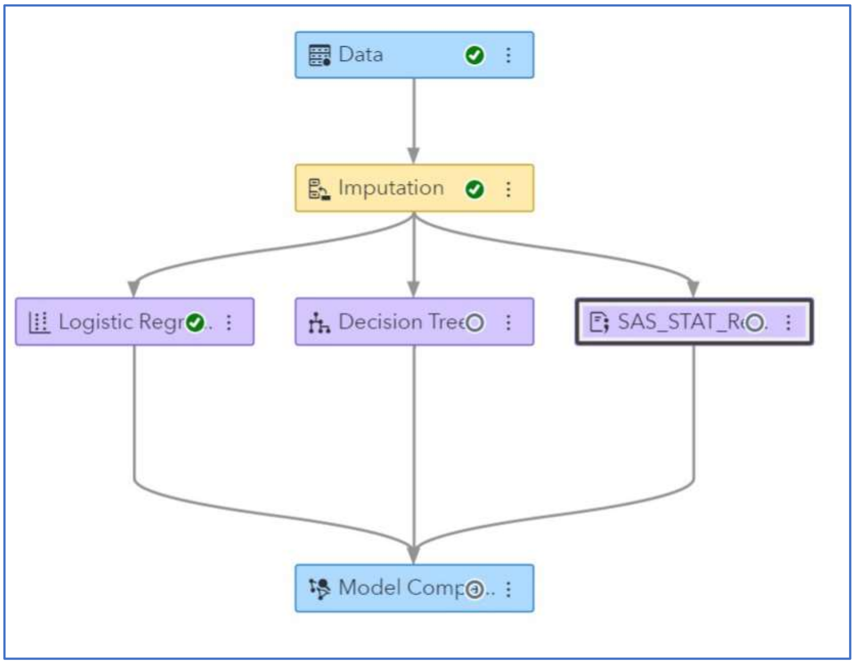
The good news? You can!

In this task, we'll examine another logistic regression model – this one created from SAS code that uses the stepwise variable selection procedure. You might not have noticed, but the default logistic regression model in our earlier SAS Model Studio pipeline uses the fast backward selection method as a default. So, in this lesson, we'll expose you to another machine learning tool within the classic logistic regression framework – all while showing you how to use the SAS Code node. That's a sneaky win-win in my book!

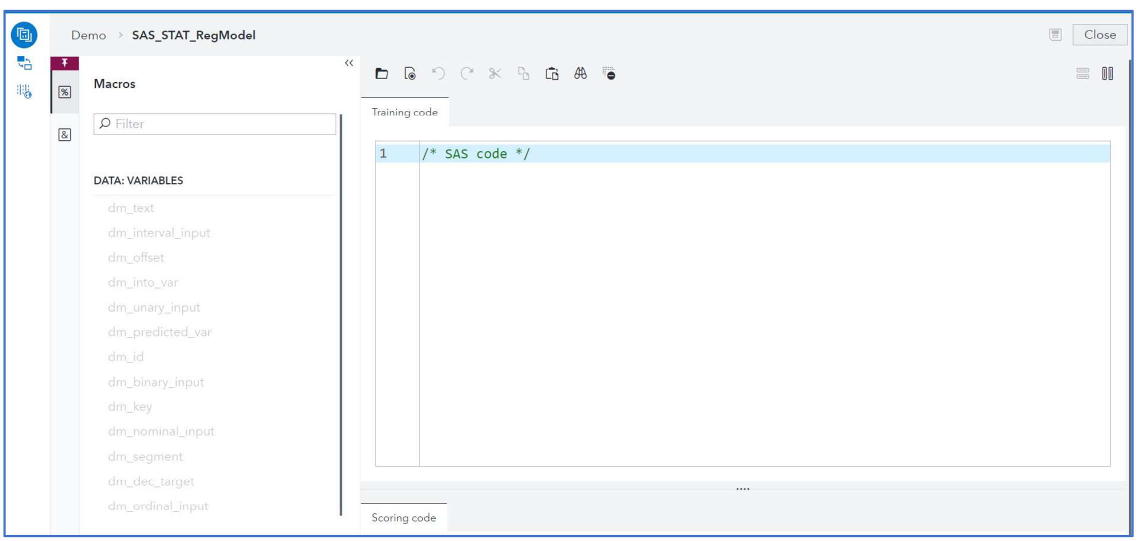
Let's get started.

1. Let's return to the **Basic Template** pipeline.
2. Add a SAS Code node to the Imputation node. How? Right-click the **Imputation** node and select **Add child node** > **Miscellaneous** > **SAS Code**.
3. Rename the SAS Code node. Right-click the **SAS Code** node and select **Rename**. Rename the node as **SAS_STAT_RegModel**.

4. Right-click the **SAS_STAT_RegModel** node and select **Move** > **Supervised Learning**. This moves the node to the Supervised Learning Lane so that it gets treated like other modeling nodes. Your pipeline should look like the one below.



5. Click the **Open Code Editor** button in the SAS Code node properties pane (or right-click the SAS Code Node and select Open). Your view should appear as follows:



6. Now let's get some code to train! Copy the code below and past it into the **Training code** section:

```
proc LOGISTIC data=&dm_data;
        class %dm_nominal_input %dm_binary_input %dm_dec_target;
        model %dm_dec_target(event="&dm_dec_event")= %dm_interval_input
                                                     %dm_binary_input
                                                     %dm_nominal_input /
                                                     selection=stepwise
                                                     slentry=0.3
                                                     slstay=0.35
                                                     details
                                                     lackfit ;
        where &dm_partitionvar=&dm_partition_train_val;
        ods output fitstatistics=&dm_data_outfit;
        code file="&dm_file_scorecode";
run;
```

7. The pretty formatted code in the training code editor should look like:

8. Let's do a deep dive into some of that code.

    a. Notice that we use macro variables that are used to identify the binary, nominal, and interval input variables along with the target variable. SAS Macros are indicated here by the "%" variables.

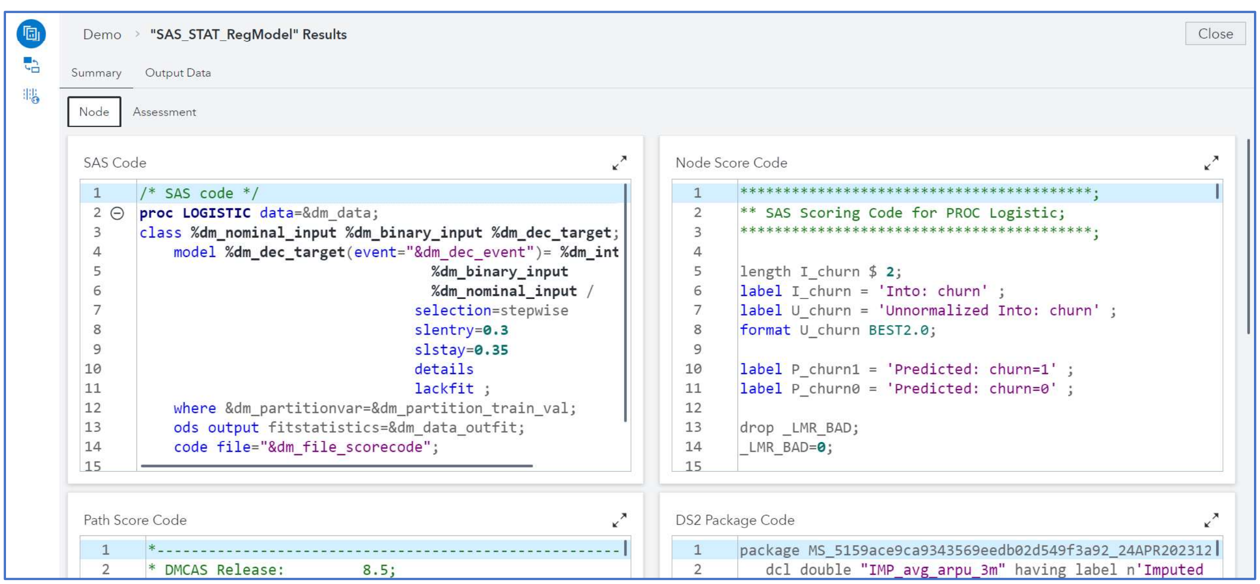
    b. Secondly, the **dm_partition_train_val** macro variable identifies the value of the partition variable that corresponds to the training observations. To carry forward the score code from a SAS Code node to the successor node (in this example, the Model Comparison node), you need to provide the score code in the **&dm_file_scorecode** macro variable. This macro variable is a file name in the node's working directory that contains the score code. When you run the SAS Code node with a DATA step score code, you see the assessment statistics that are provided to you in the results of the node: lift plots, ROC plots, and fit statistics for a class target.

    c. The model is also included in the results of the Model Comparison node for evaluating its performance against other Supervised Learning models. Good stuff!

9. Click **Close** and select **Save** to register the changes to the SAS Code node.

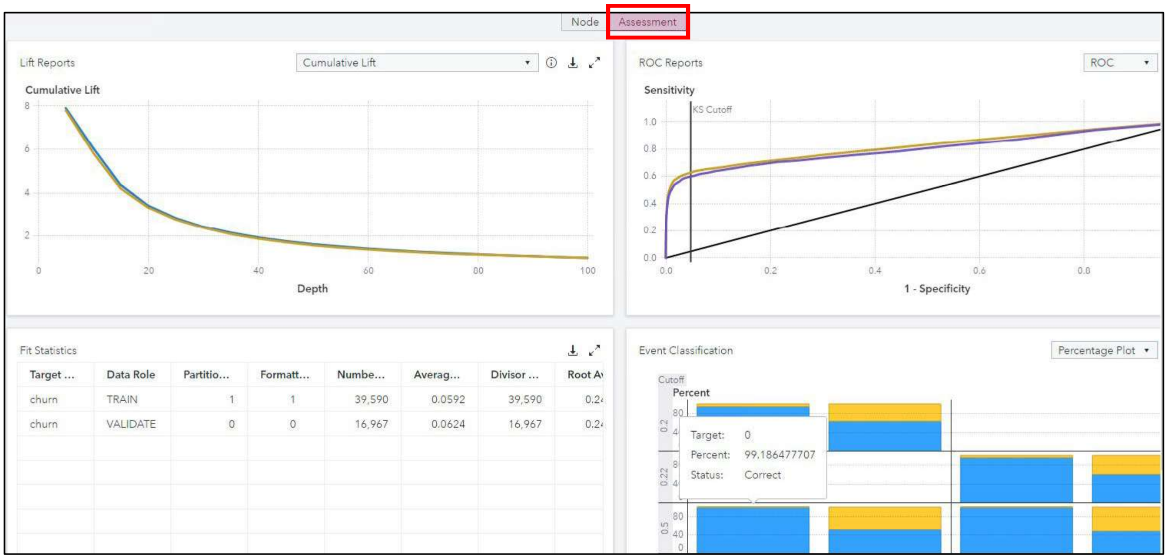
10. Right-click the **SAS_STAT_RegModel** node and select **Run**.

11. Open the **Results** of the **SAS_STAT_RegModel** node, which lands on the **Node** tab by default.



12. The **Node** tab of the results window contains all the reports included in supervised modeling nodes: Score Inputs and Score Outputs tables, Path Score Code, Node Score Code, Properties table, and the Output Delivery System (ODS) output that was generated by the LOGISTIC procedure.

13. Click the **Assessment** tab.



14. Explore the various assessment tools for a bit so that you can better understand the various ways to answer the question “Is this a good model?”

15. **Close** the results window.

16. To finish this task, let’s see how the SAS®9 model did against the two other models in the **Basic Template** pipeline. Run the **Model Comparison** node and view **Results**:

Model Comparison

Champi...	Name	Algorithm Name	KS (Youden)	Accuracy
★	SAS_STAT_RegModel	SAS Code	0.5836	0.9320
	Logistic Regression	Logistic Regression	0.5825	0.9322
	Decision Tree	Decision Tree	0.5707	0.9386

17. How did we do? Well, the classic SAS®9 model is the best algorithm yet – but not by much. :)

Task 8: Open Source Time!

Learning Objectives

- Incorporate the Open Source Code node into our pipeline.
- Run a Python forest.
- Run an R forest.
- Examine model results.

Estimated Time of Completion

This task will take approximately 10-15 minutes to complete.

Task 8: Open Source Time!

In the last task, we saw how to incorporate SAS®9 code directly into our **Basic Template** SAS Model Studio pipeline. But you might now have another new thought:

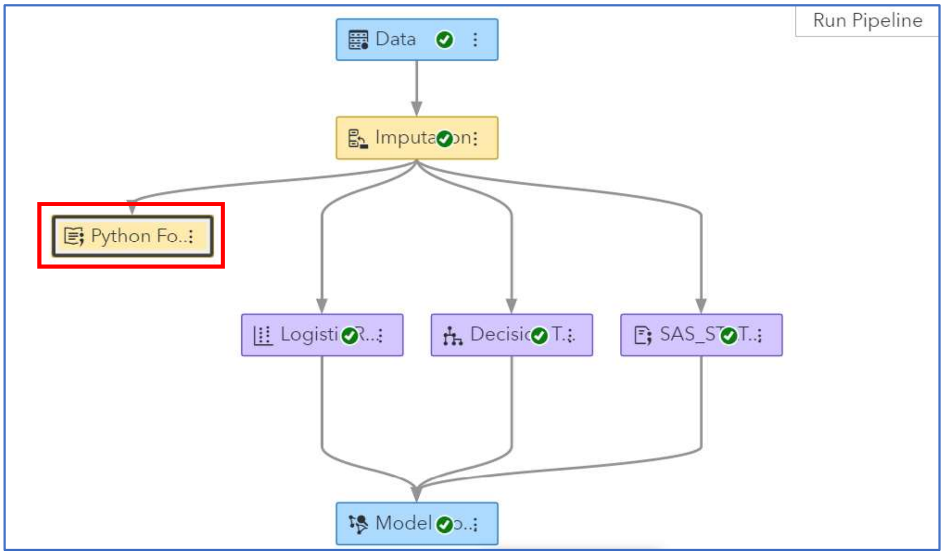
Wait, I'm a Python + R coder – and I'd like to incorporate some of those machine learning models into my analysis. Can I do that?

Of course you can. Of course you can…

In this task, we'll use the **Open Source Code** node to create forest models in R and Python. Let's finish this last modeling section strong!

1. Remain in the **Basic Template** pipeline. If you went down a different rabbit hole, no worries: please return to that pipeline in SAS Model Studio.
2. Right-click the **Imputation** node and select **Add child node → Miscellaneous → Open Source Code**.

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

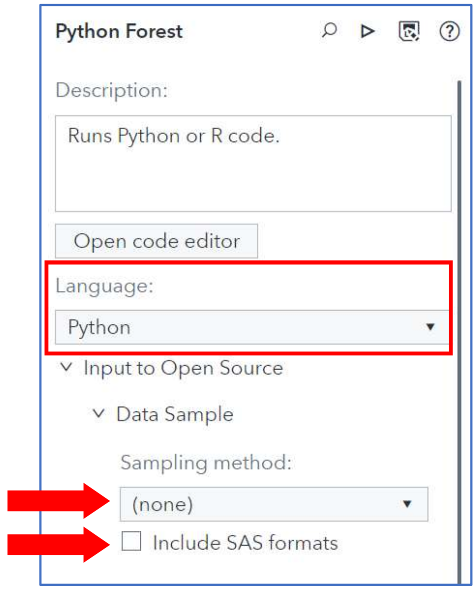


4. In the properties panel of the **Open Source Code** node (renamed **Python Forest**), verify that the language is set to **Python**. It should be Python by default.

5. Expand the **Data Sample** properties. Clear the check box for **Include SAS formats**.

   A fun note: this property controls whether the downloaded data sent to Python or R should keep SAS formats. Why? Well, it is usually recommended that you keep SAS formats, and this should work in most cases. But… some numeric formats such as 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.

6. Also change the **Sampling method** to **(none)**. The cumulative changes:

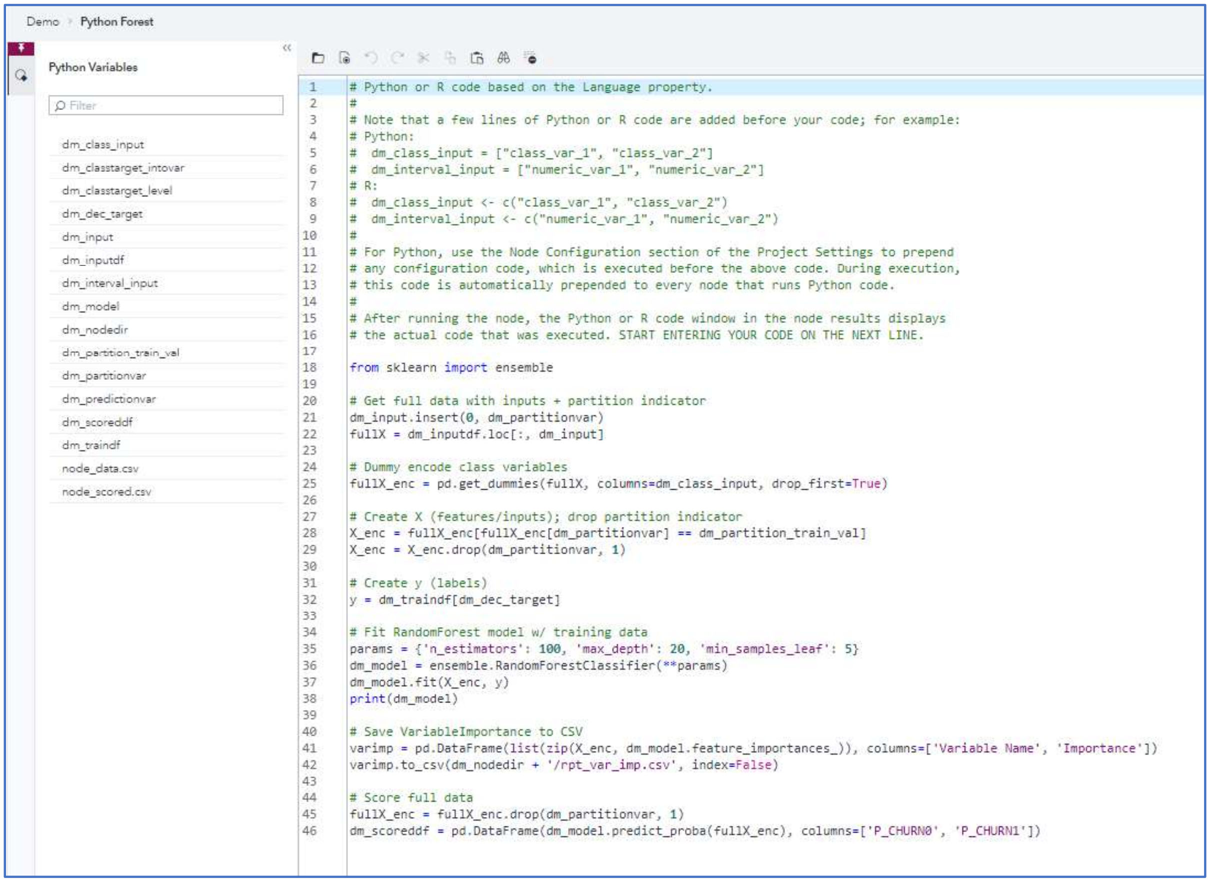


7. Click the **Open Code Editor** button to invoke the SAS Code Editor.

8. 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_CHURN0', 'P_CHURN1'])
```

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



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

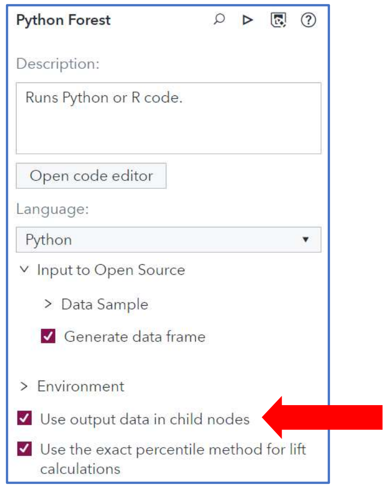
   a. This code fits a random forest classifier model in Python. The default values for the parameters that control the size of the trees (for example, **max_depth (default=none)**, **min_samples_leaf (default=1)**) lead to fully grown and unpruned trees, which can be very large data sets. To reduce memory 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).

11. 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.

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

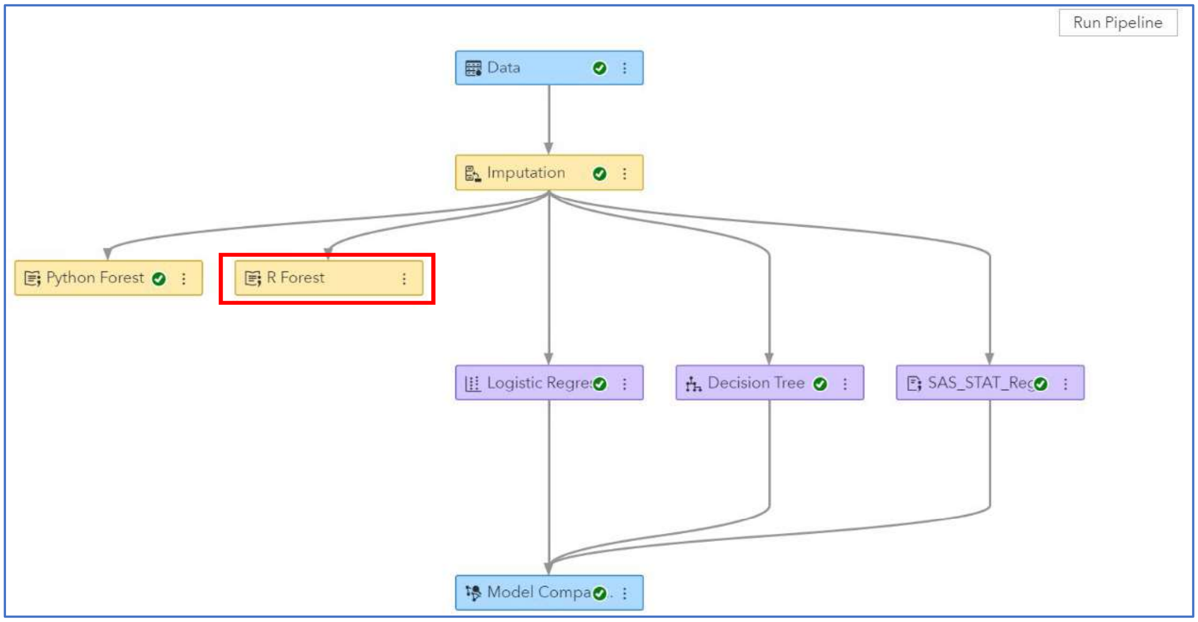


13. A little bit behind the why of the *Use output data in child nodes* button. Every time that this property is set, a copy of the output data is saved in the Model Studio project library, which can be used in later stages of the pipeline.

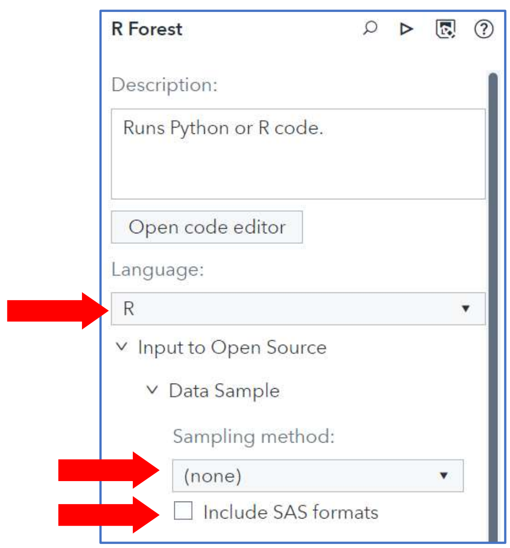
14. Now it is time to run the **Python Forest** node.

15. Repeat the previous steps for fitting a forest model in R. To start, right-click the **Imputation** node and select **Add child node ➔ Miscellaneous ➔ Open Source Code**.

16. Then right-click the **Open Source Code** node and rename it to **R Forest**. Your pipeline should look like the one below.



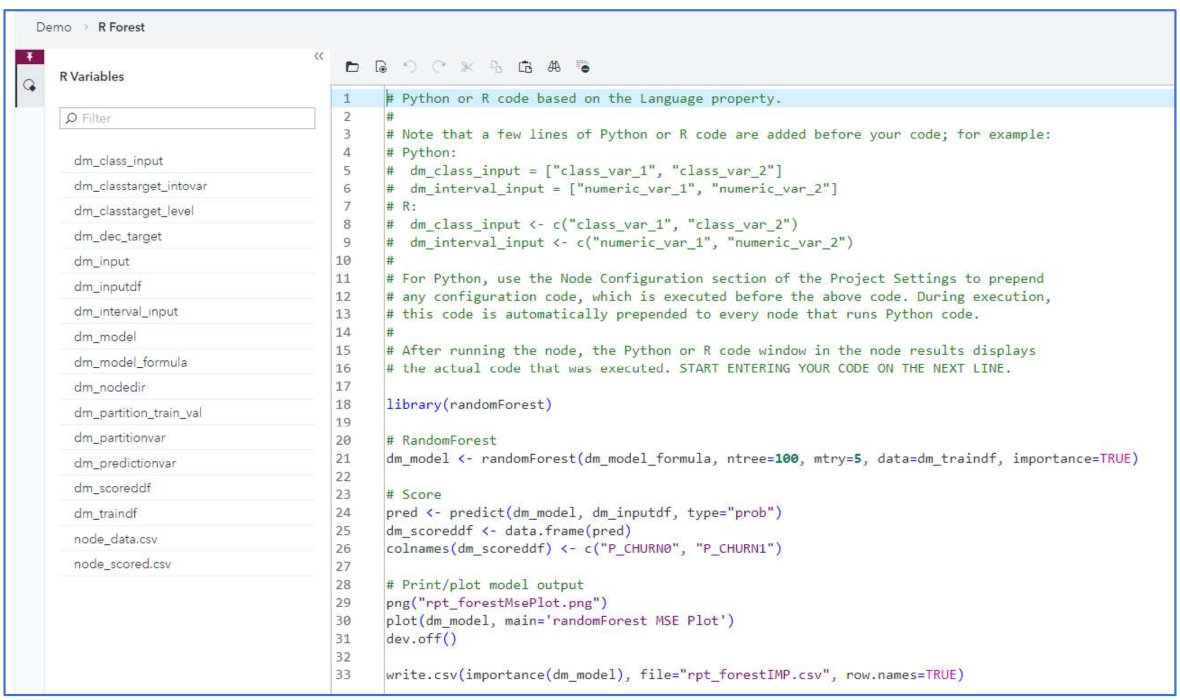
17. In the properties panel of the Open Source Code node (renamed **R Forest**), set the language to **R**, change the **Sampling method** to **(none)**, and clear the check box for **Include SAS formats** under **Data Sample**:



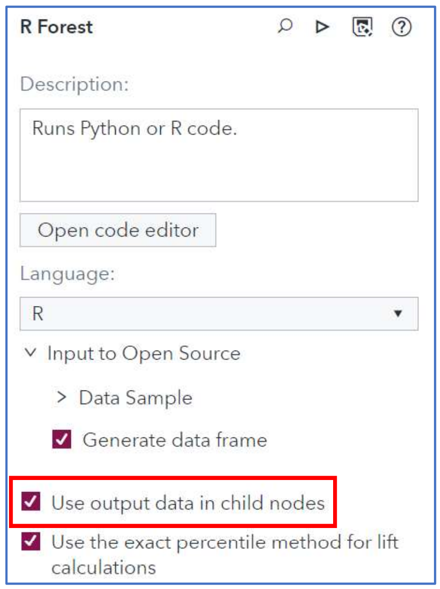
18. Click the **Open Code Editor** button to invoke the SAS Code Editor. It's copy-and-paste time for one more chunk of code! So, grab the following:

```
library(randomForest)

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

# Score
pred <- predict(dm_model, dm_inputdf, type="prob")
dm_scoreddf <- data.frame(pred)
colnames(dm_scoreddf) <- c("P_CHURN0", "P_CHURN1")

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

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

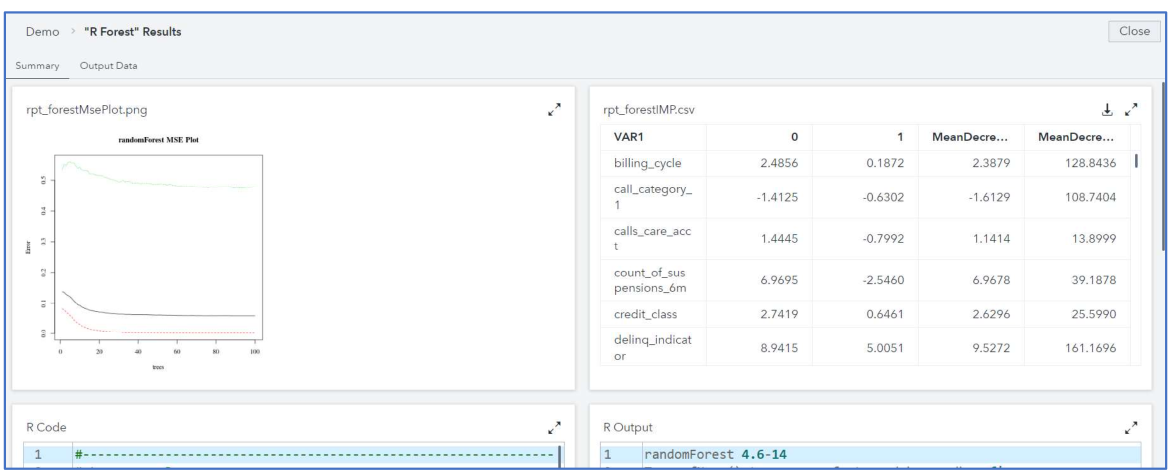
19. And one more time showing the R script in the pretty code editor:



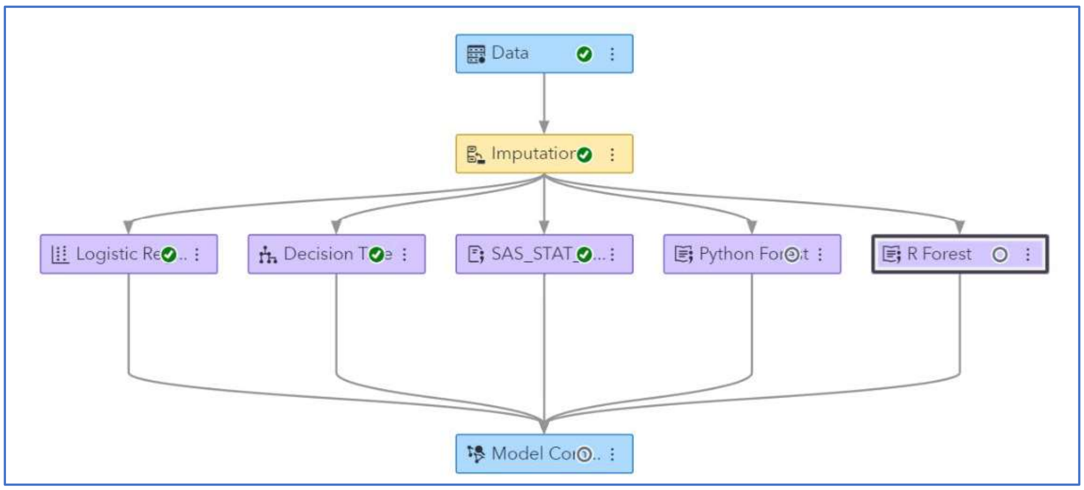
20. Now, some fun modeling notes:
    a. This fits Breiman and Cutler's random forest classifier model in R.
    b. The code that needs to be changed for different data sets is the last line – i.e., naming your predictions with the **P_ + “*target*”** naming convention.
21. In the upper right corner of the window, click the **Save** icon to save the R code and then click the **Close** button to close the Code Editor window.
22. Finally, let’s make one more change before running the R node. Like last time, select the **Use output data in child nodes** property:



23. Run the R Forest node.
24. Open the results of either the R Forest node or the Python Forest node to examine output.

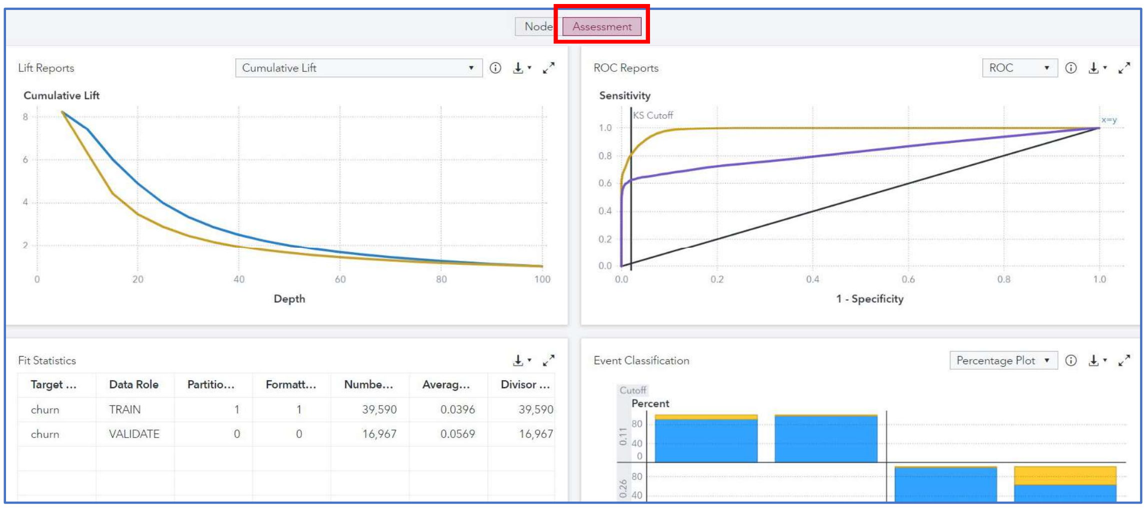


25. Examine the results for a bit, so you get a better understanding of model performance.
26. And while you're doing that, I have a thought-provoking question for you: *Why does the Open Source Code node not have assessment results even though it was successfully executed?* Well, for model assessment, you need to move the nodes to the supervised learning lane. And recall that we did this with the SAS®9 regression model.
27. To get the two open-source nodes registered as models, right-click the **R Forest** node and select **Move → Supervised Learning**.
28. Repeat the same for the Python Forest node – and the pipeline should appear as follows:



29. The color of both the nodes has changed to purple, showing that these nodes have changed to the group of Supervised Learning nodes. Notice also that the nodes need to be rerun, as the green check mark has disappeared.
30. Click the **Run Pipeline** button.

31. Open the **Results** of the Python Forest or the R Forest node (or both). Click the **Assessment** tab to learn even more about model performance.



32. The usual assessment results are displayed. Explore until your heart is content and then close the results.

33. Open the **Results** of the **Model Comparison** node. Which model is our ultimate champion in the *Basic Template* pipeline?

Model Comparison

Champi...	Name	Algorithm Name	KS (Youd... ↓	Accuracy
★	Python Forest	Open Source Code	0.6061	0.9401
	R Forest	Open Source Code	0.6009	0.9410
	SAS_STAT_RegModel	SAS Code	0.5836	0.9320
	Logistic Regression	Logistic Regression	0.5825	0.9322
	Decision Tree	Decision Tree	0.5707	0.9386

34. For our modeling, the Python Forest has the highest KS (Youden) statistic, followed closely by the R Forest. So, it looks like the open source modeling was a nice addition. Woot!

35. Close out of the **Model Comparison Results** window – if, and when, you've seen all the statistics in this section that you can handle.

Task 9: Wrapping Up

Learning Objectives

- Crown a champion of champions across all pipelines using pipeline comparison.
- Use Insights to make your analysis more digestible for your (likely non-technical) audience.

Estimated Time of Completion

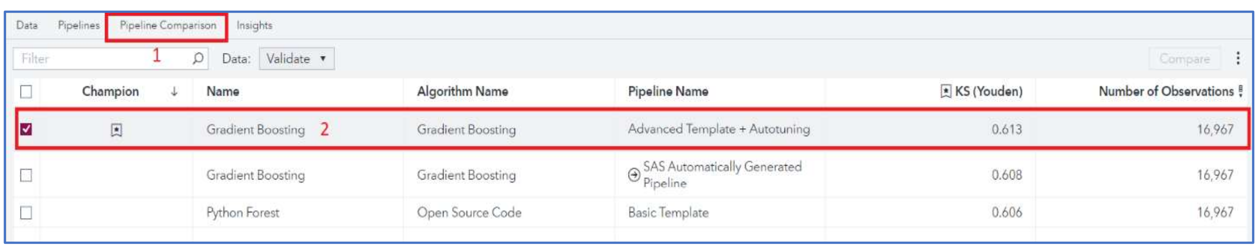
This task will take approximately 10-15 minutes to complete.

Task 9: Wrapping Up

Congrats on nearly making it through your first full day at iLink Telecom, Inc! Think about all modeling goodness that you've accomplished thus far: at least 15 machine learning models across three pipelines. But, the goal isn't simply modeling for modeling's sake. We actually want to choose a "best" model – and then share these results with others.

So let's get on it!

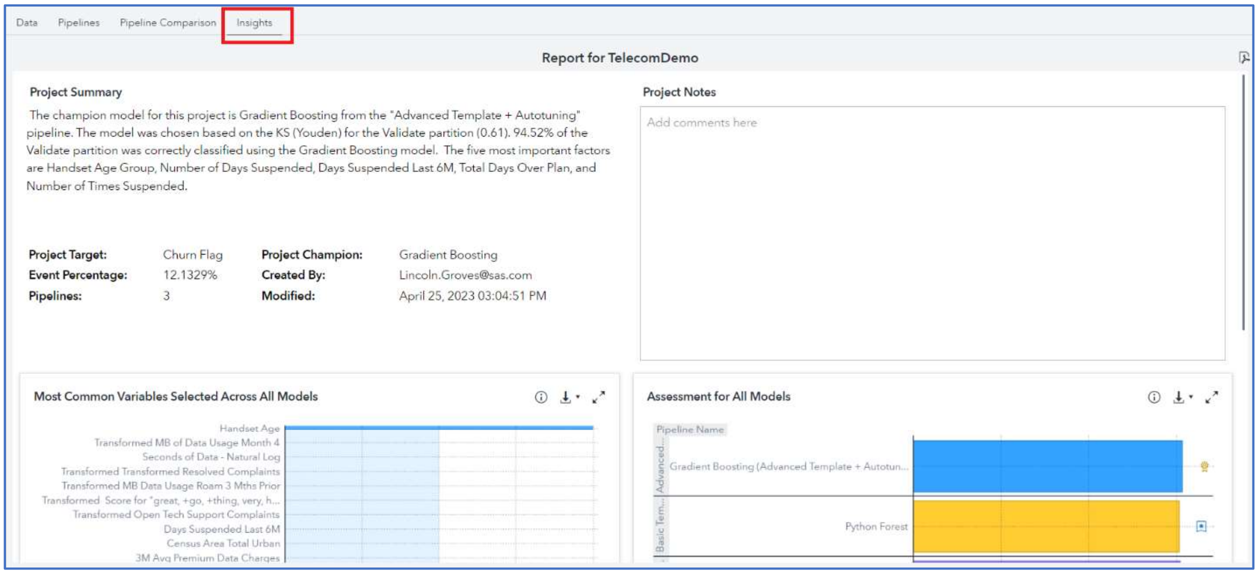
1. We've already seen the **Pipeline Comparison** tool in SAS Model Studio a couple of times. Let's return to it one more time to see which model is crowned the ultimate modeling champion across our three pipelines.



2. It appears that the Gradient Boosting model from the *Advanced Template + Autotuning* pipeline has the highest KS (Youden) statistic. So, that's something!

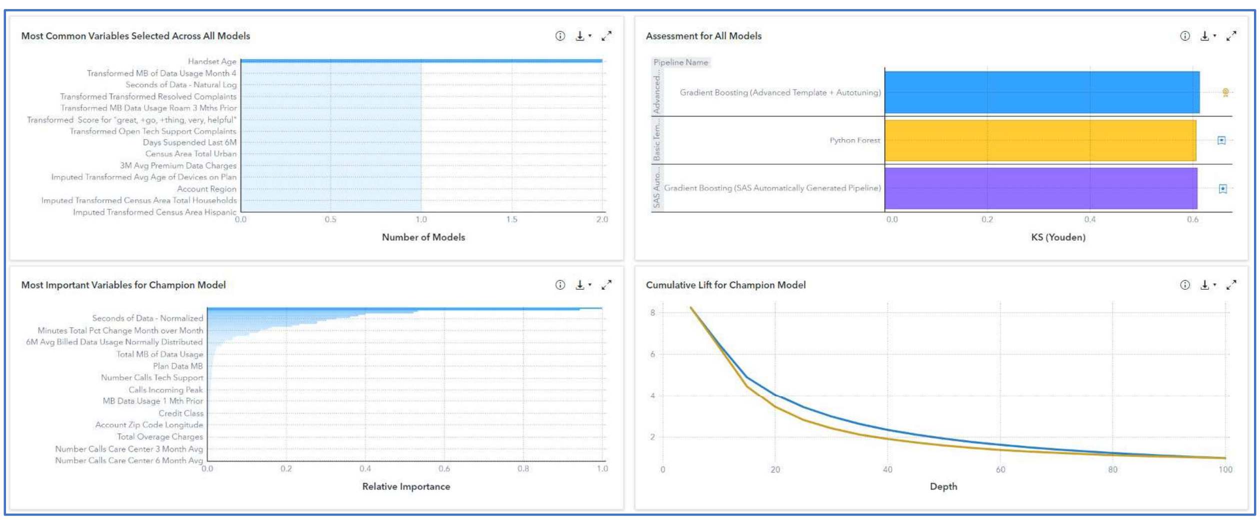
3. Traditionally, we would then register this model and then apply the model to new cases where the "churn" decision hasn't yet occurred. In other words, we'd predict new cases and then share with marketing. However, we'll stop here for today and focus now on tools for storytelling.

4. Our last tool for investigation will be found under the **Insights** tab. Storytelling is such a critical part of data analytics and modeling – and SAS Model Studio has a tool that will help you convey your findings to the appropriate audience. To get started, click the **Insights** tab:



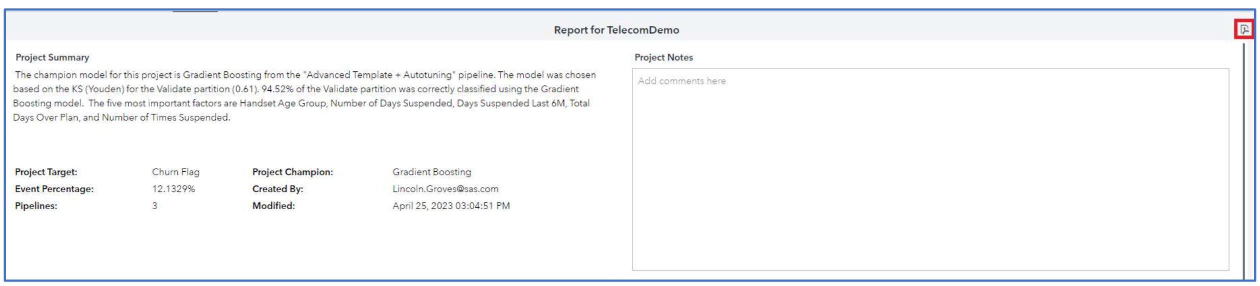
5. On the **Insights** tab, you'll find a host of useful infographics, such as a **Project Summary** and the **Most Common Variables Selected Across All Models**.

6. Scroll down a bit, and you'll also see the **Assessment for All Models**, the **Most Important Variables for Champion Models**, and the **Cumulative Lift for Champion Model**:



7. For many of the infographics, you can click the ⓘ icon for an explanation of the graphic. This tool can help you better convey this information to your audience. An example:

8. Finally, would you like to export your results to a PDF? Well, SAS Model Studio has got you covered with the press of one little button:



9. [insert applause here]

10. We're finished! Congratulations on completing your first day at iLink Telecom, Inc! We look forward to many, many more analytical adventures in the future!

Appendix

Appendix A: Access Software

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

1. Navigate to the SAS Viya for Learners web page: 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 institution. If you don't have a SAS Profile, click here to set one up.
4. Access the software by clicking the **Launch SAS Viya for Learners** button.

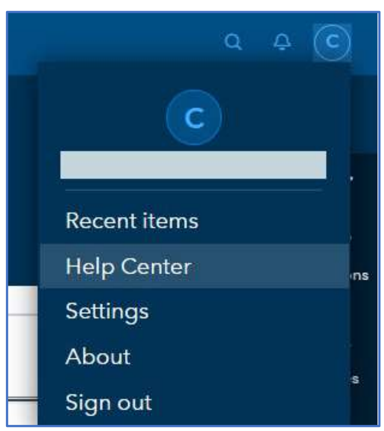
For future sessions with VFL, you can access VFL by visiting https://vle.sas.com/vfl.

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 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 Academic Hub offers free e-learning courses for students to learn SAS. The following e-learning courses and paths available in the Academic Hub are recommended to help with this activity:

- Machine Learning Using SAS Viya

Students can access these courses by going here:

- The SAS Skill Builder for Students.
- Or, the SAS Learning Subscription grants you access to an extensive library of SAS e-learning courses. Sign up for a free seven-day trial.

