# SAS Viya + SAS Viya Workbench Overview
# SAS Guided Demo

## Purpose

This SAS Guided Demo provides you with an overview of the wonderful, powerful world of SAS Viya. From no- and low-code environments to cutting-edge tools just for coders and developers, SAS Viya is a great place for anyone to start – or resume – an analytics journey. In this activity, we'll provide you with a broad overview of some important tools in the SAS Viya ecosystem – and then provide you some next steps to continue your learning.

## Overview

- This SAS Guided Demo provides a quick overview of two great SAS software platforms available to academics: (1) SAS Viya and (2) SAS Viya Workbench
- In SAS Viya, you'll see:
    - How quickly SAS Visual Analytics turns data into insights
    - How quickly we can get from data to completed models in SAS Model Studio
    - So, speed, speed, speed in SAS Viya… and once of the key is the embedded AI agents in the SAS Viya tools
- In SAS Viya Workbench, you'll see
    - How coding gives you the ultimate flexibility in your analysis
    - How WFL provides you the ultra-fast SAS platform to run both SAS + Open-Source code
- This guided demo hopefully gets you interested in one of *FOUR* learning paths, depending upon what questions you're still asking yourself after we finish:
    - *Still confused and want to learn more about the SAS Viya ecosystem?*
        - Take our SAS® Viya Overview course
    - *Like dashboarding and SAS Visual Analytics?*
        - Explore SAS Visual Analytics 1 for SAS® Viya: Basics
    - *Like predictive modeling and machine learning in a low/no-code environment?*
        - Machine Learning Using SAS® Viya® is a great option!
    - *Finally, love coding in a multi-language environment and want to explore SAS Viya Workbench more?*
        - Modern Data Science with SAS® Viya® Workbench and Python is perfect for you!

## Backstory for this Demo

- Let's play pretend.
    - You are a newly hired Data Scientist in the Customer Retention Department at Styled and Sophisticated Enterprises (SASe)
    - SASe provides personal stylist services, offering premium clothing boxes through a subscription model.
    - Your goal is to analyze customer churn to better understand which customers are on the brink of canceling their subscription service

## Software

This SAS Guided Demo uses SAS Viya for Learners 4, version 2024.09LTS. Other versions can be used, but some screenshots will differ.

## Prerequisites

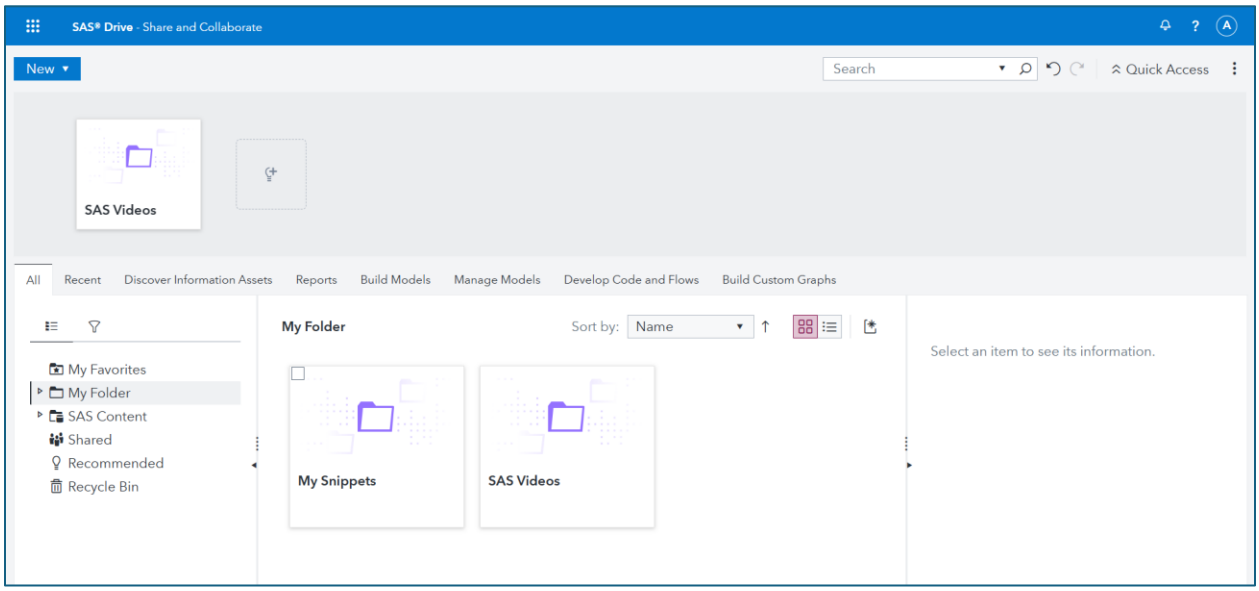
Some experience using the SAS Viya ecosystem is helpful but not required.

# Table of Contents

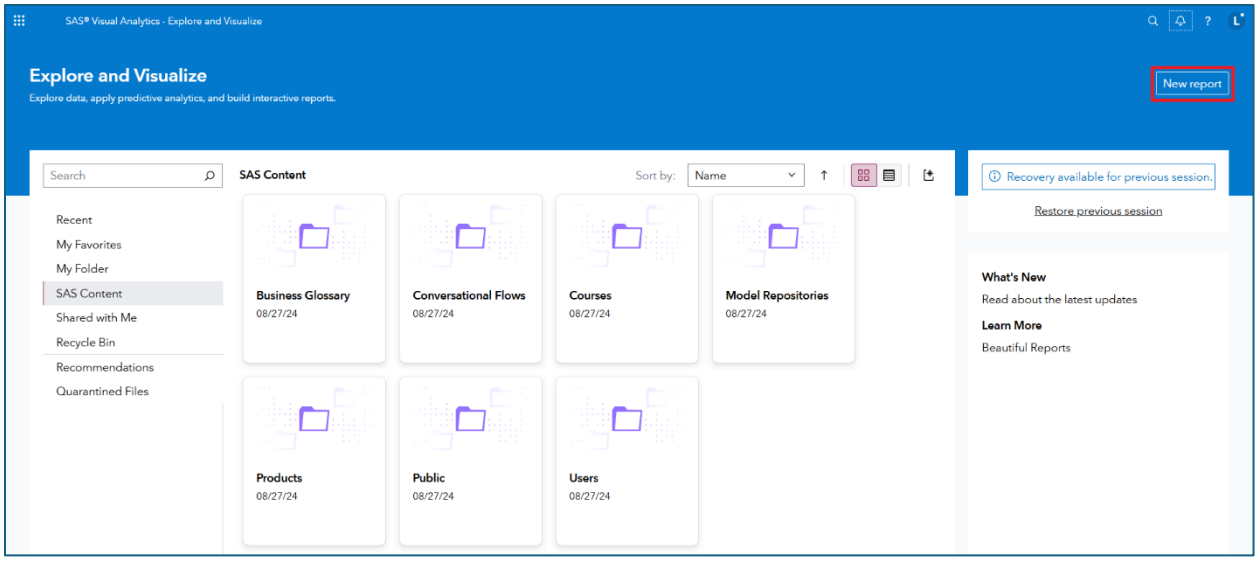
## Understanding Your Data in SAS Visual Analytics

A good data adventure often starts with software.

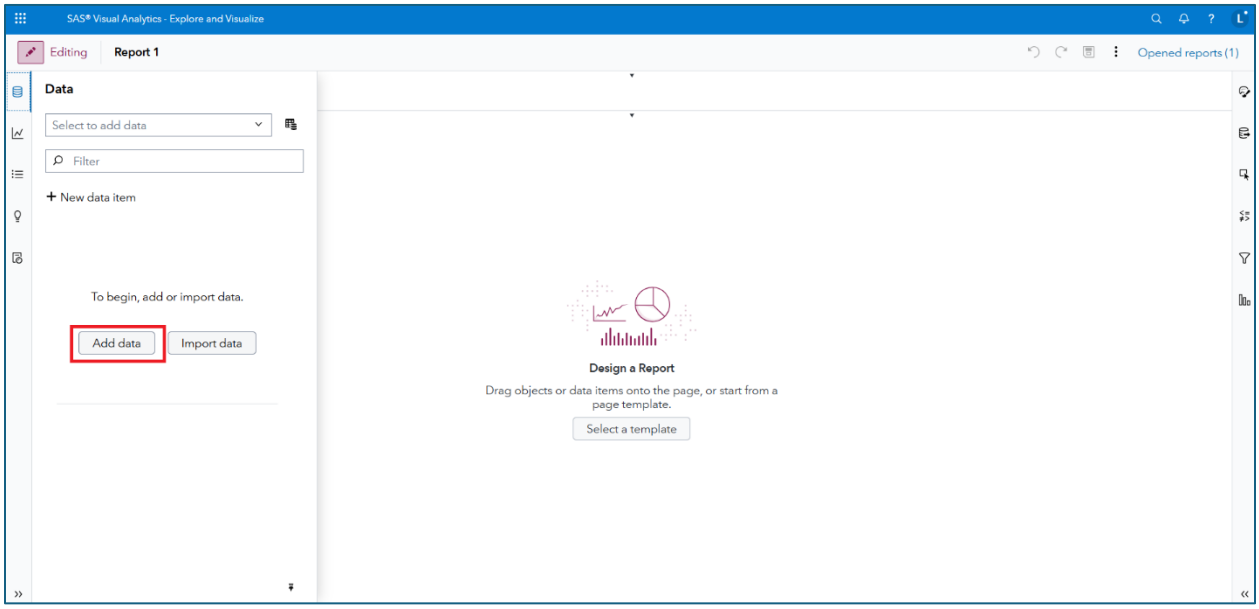
- As such, log in to **SAS Viya for Learners**.
- Once in, you'll land in **SAS Drive**, which is your starting point in SAS Viya for Learners. It will look something like the following:



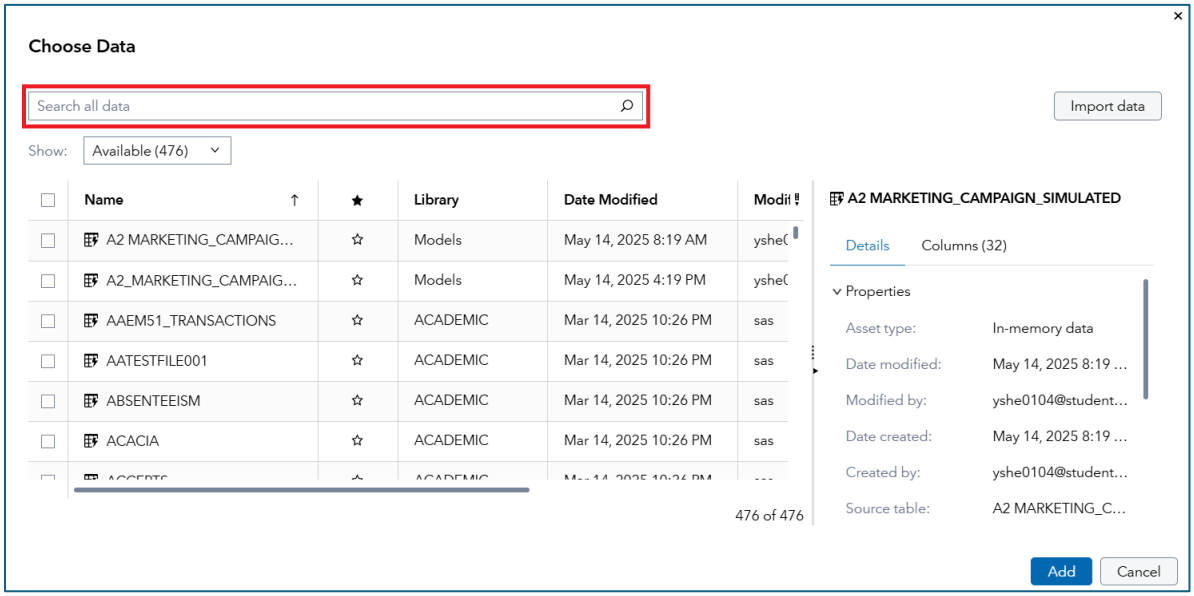

- But, we won't linger here long – as we want to get right into **SAS Visual Analytics**. From the top left corner, select the **Applications menu** and click **Explore and Visualize** from the **Analytics Life Cycle** options.
- And the **Explore and Visualize** landing page awaits. Click that **New report** button in the upper right corner:



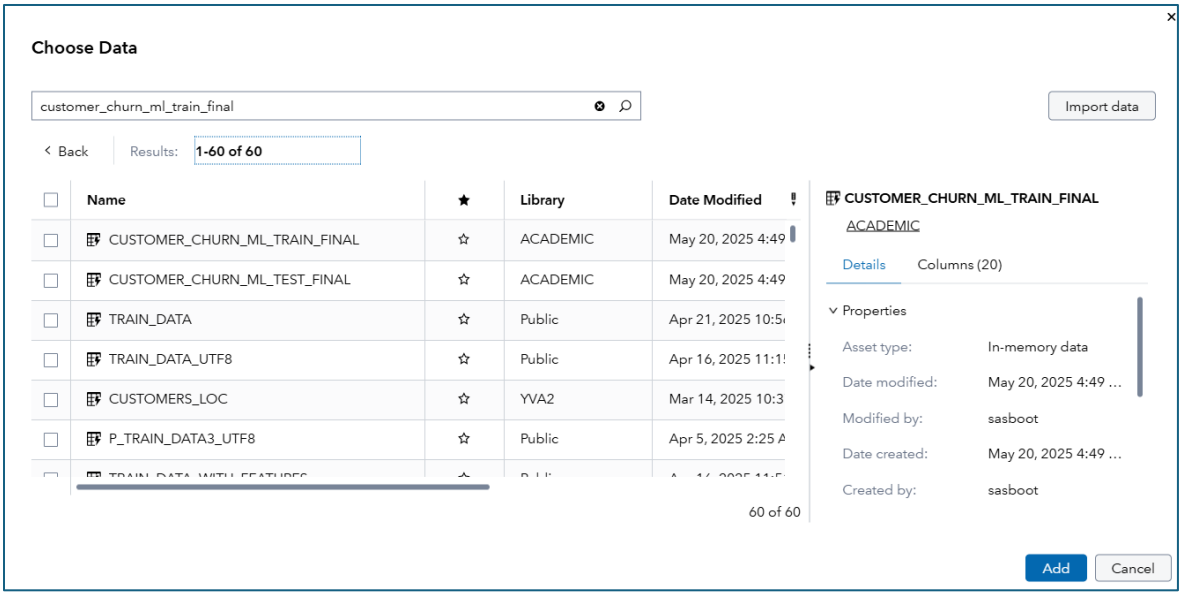
- Now you're really in SAS Visual Analytics. Look for the **Add data** button, as it's always great to start a data adventure with a data set:



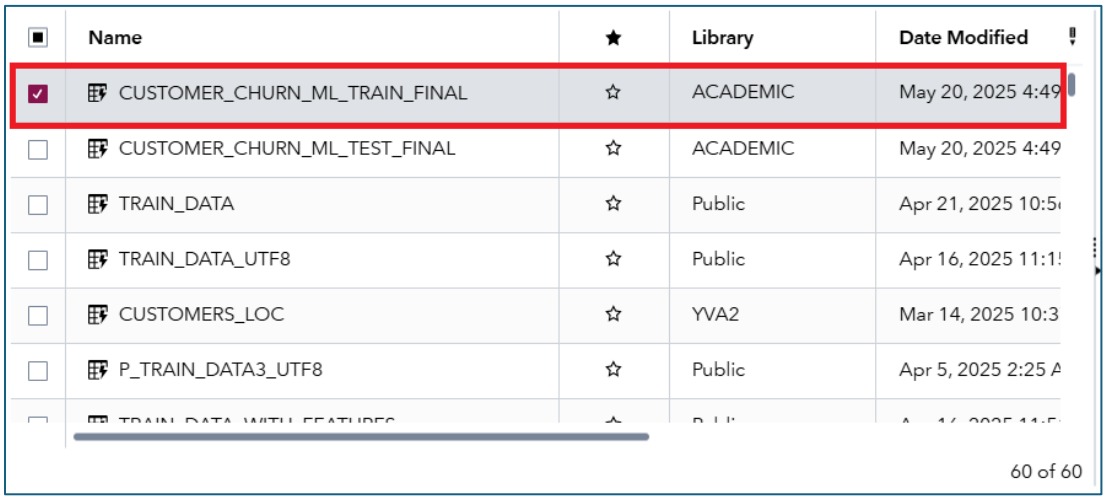

- Click **Add data**. And – in the **Choose Data** window – locate the **Search all data** bar:



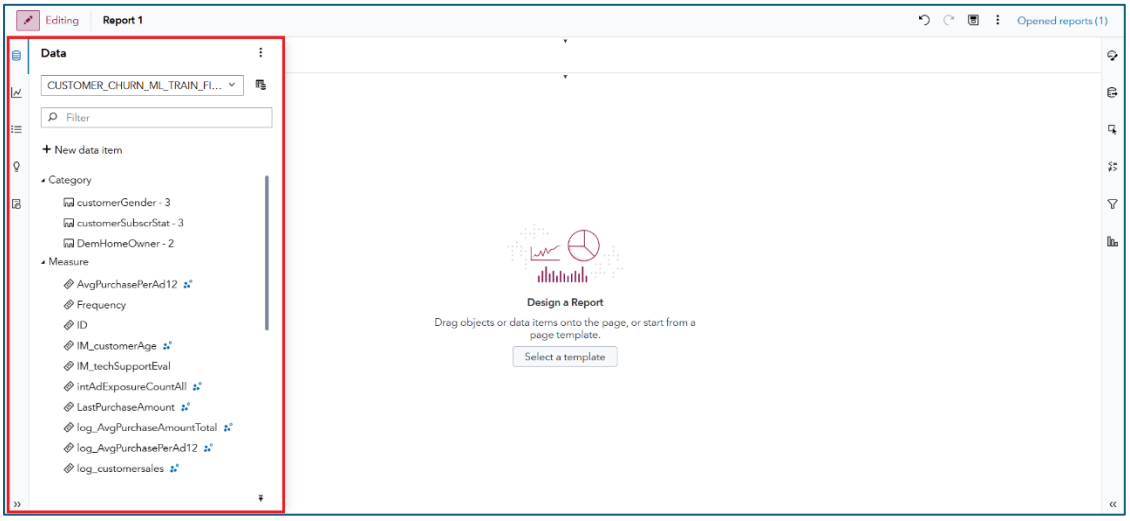

- Type (or paste) *customer_churn_ml_train_final* into the search bar. The press **Enter**. You should have a smaller number of datasets to process:




- And the first file in the ACADEMIC folder is perfect. Select that data set:

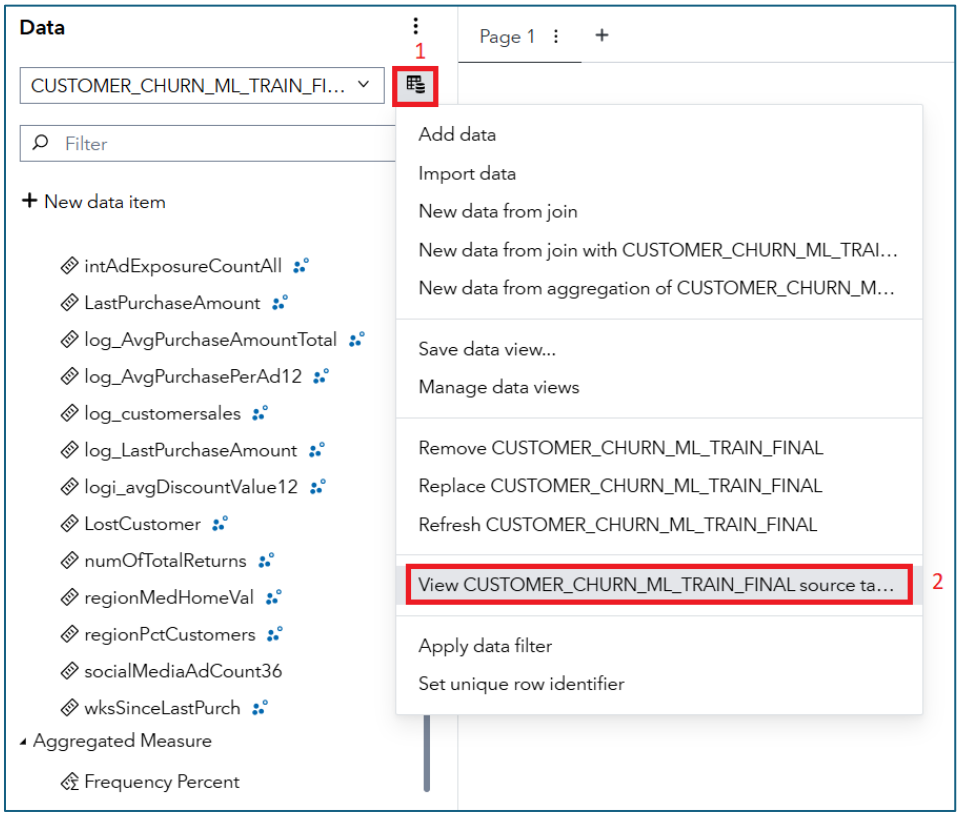



- Then click **Add**. You now have data in your VA report:



- Go ahead and click that **Pin pane** button, just to make navigation easier. It's found in the lower-right corner of the Data pane, and looks like this:

- Ready, set, dashboard! Are you ready to get to know your data a bit better with some dashboarding? Well, let's do it! A good place to start with a new data set is to look at the underlying values. So, from that **Data** tab, find the **Data source menu**. Then navigate to **View customer_churn_ml_train_final source table**, like so:




- A tabular table awaits!

View Source Table

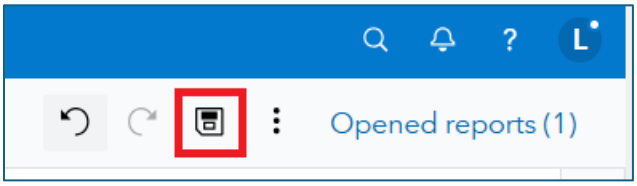
CUSTOMER_CHURN_ML_TRAIN_FINAL

20 of 20 columns | 3,501 of 3,501 rows | Find

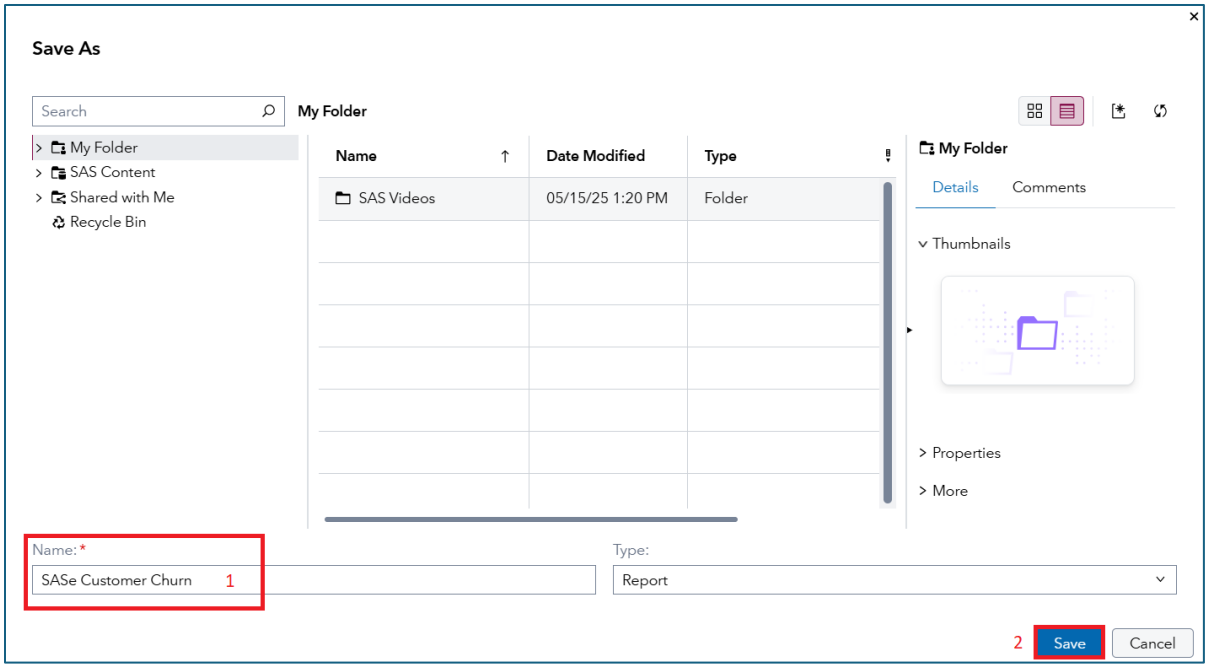
| # | LostCustomer | ID | regionPctCusto... | numOfTotalRet... | wksSinceLastPu... |
|---|---|---|---|---|---|
| 1 | 0 | 46197 | 3.0449805295 | -1.02740403 | 0.2365217516 |
| 2 | 0 | 72781 | -0.817002252 | -0.34396464 | -0.190204686 |
| 3 | 0 | 16300 | -0.64145758 | 0.3394747498 | 0.0231585327 |
| 4 | 0 | 102683 | -0.378140572 | -1.02740403 | 0.0231585327 |
| 5 | 0 | 184919 | 0.938444467 | 0.3394747498 | 0.876611408 |
| 6 | 0 | 1647 | -1.08031926 | 1.7063535296 | -2.750563312 |
| 7 | 0 | 9792 | 1.9917124982 | -1.02740403 | -2.110473655 |
| 8 | 0 | 47989 | -0.2025959 | -0.34396464 | 0.0231585327 |
| 9 | 0 | 187451 | -0.027051228 | 0.3394747498 | 0.6632481892 |
| 10 | 0 | 177149 | 1.2017614748 | 0.3394747498 | -0.403567905 |
| 11 | 0 | 10046 | -2.747993642 | -0.34396464 | -2.750563312 |
| 12 | 0 | 184328 | 0.5873551232 | 2.3897929196 | 0.2365217516 |
| 13 | 0 | 174129 | -1.431408603 | -0.34396464 | 0.4498849704 |

Close

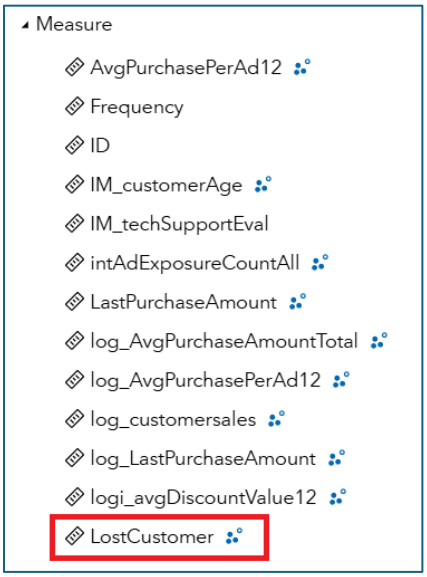
- And we can see a couple of important things quickly. To start, our data set is just 3,501 rows and has 20 columns. **LostCustomer** is our outcome of interest – as it is a binary variable which reveals whether the customer canceled their premium subscription box at SASe (1=yes and 0=no). After the unique customer ID, *ID*, we then have a number of variables which help explain the customer's behavior. Explore a bit to get to know the data a bit better – then click **Close** when you're satiated.
- Now let's create a couple of dashboards. But before getting there, we need to save – and save often in SAS Visual Analytics! How to save? Well, it's the old-school floppy disk found in the upper-right corner, here:



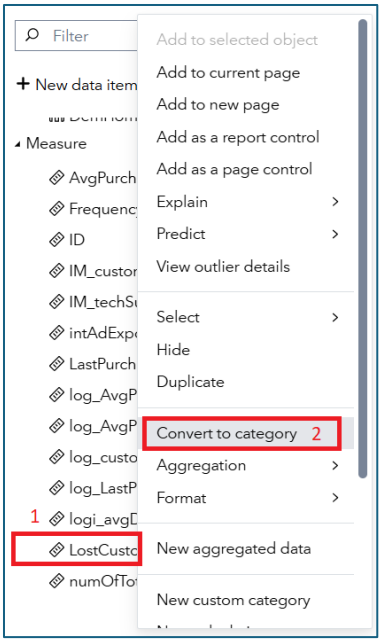

- The project **Name** can be something like *SASe Customer Churn*. Then just click **Save** to place the project in *My Folder*:



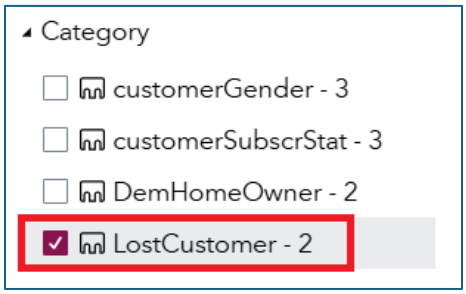

- The first page in our dashboard will be a bunch of descriptive plots to better understand our data. And we'll leverage our first AI agent – the **+ Auto chart** tool – to produce some quick charts. Ready? If yes, search for **LostCustomer**. Do you see it under **Measure**?



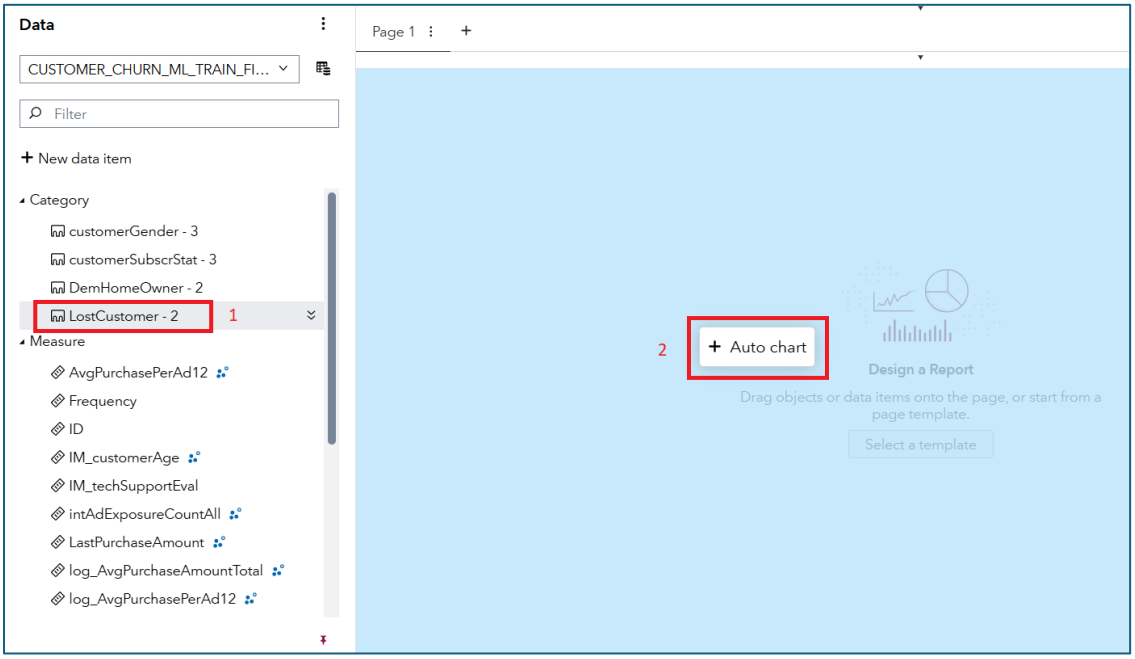

- Well, I do. But we actually want it as a yes/no variable in our model, rather than a continuous variable. But the fix is easy. Just right-click on **LostCustomer** and select **Convert to category**:



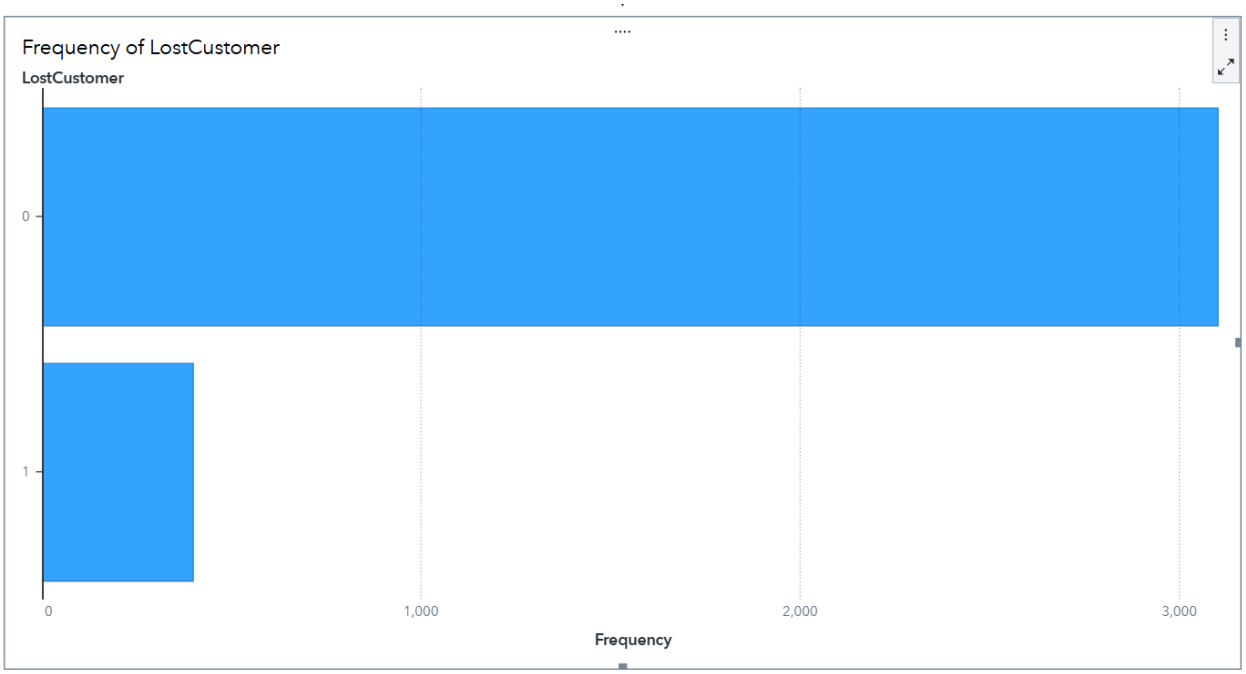

- **LostCustomer** will now show up under Category, with two values:



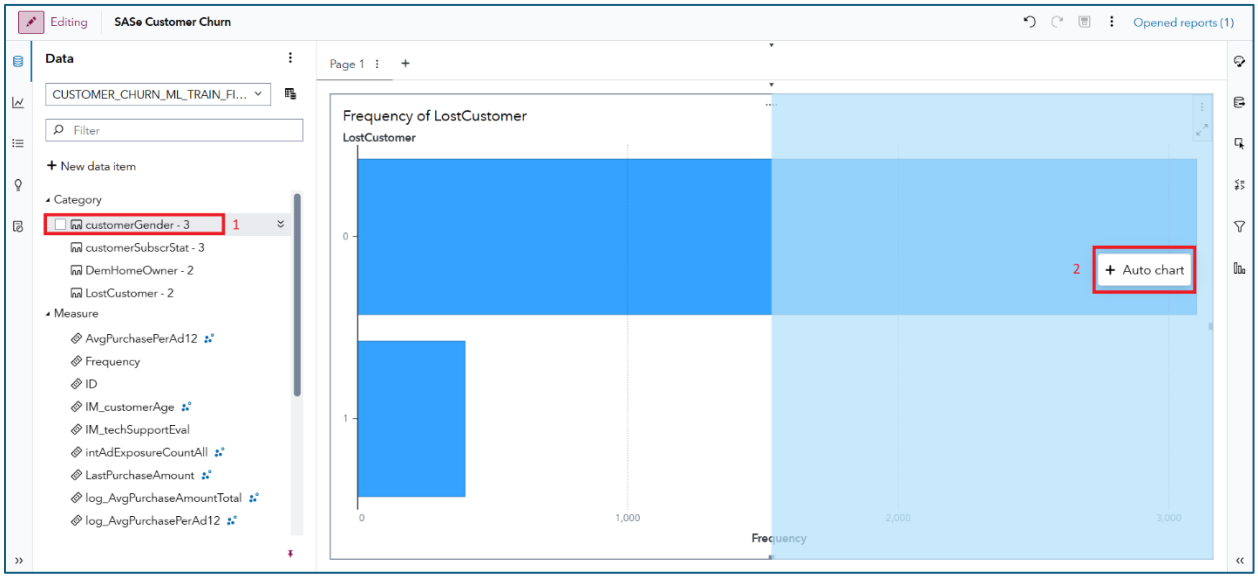

- Next, drag-and-drop **LostCustomer** onto the canvas, like so:



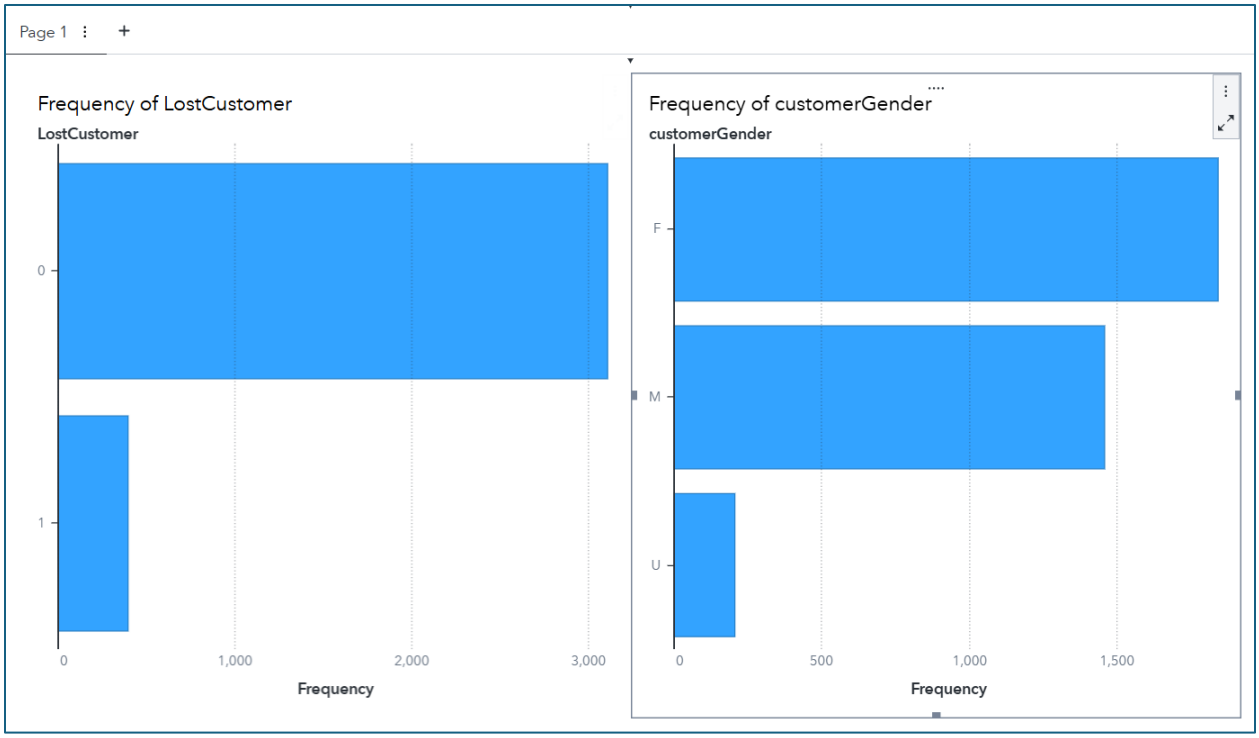
- And you've got a bar chart!



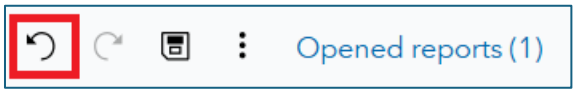

- And you can see – quickly – that while some customers churn (LostCustomer=1), we're doing a pretty solid job of keeping our clients at SASe. Now let's add a few more variables on the page to further explore the data. Find **customerGender**. Drag-and-drop it to the right of *LostCustomer*, like so:



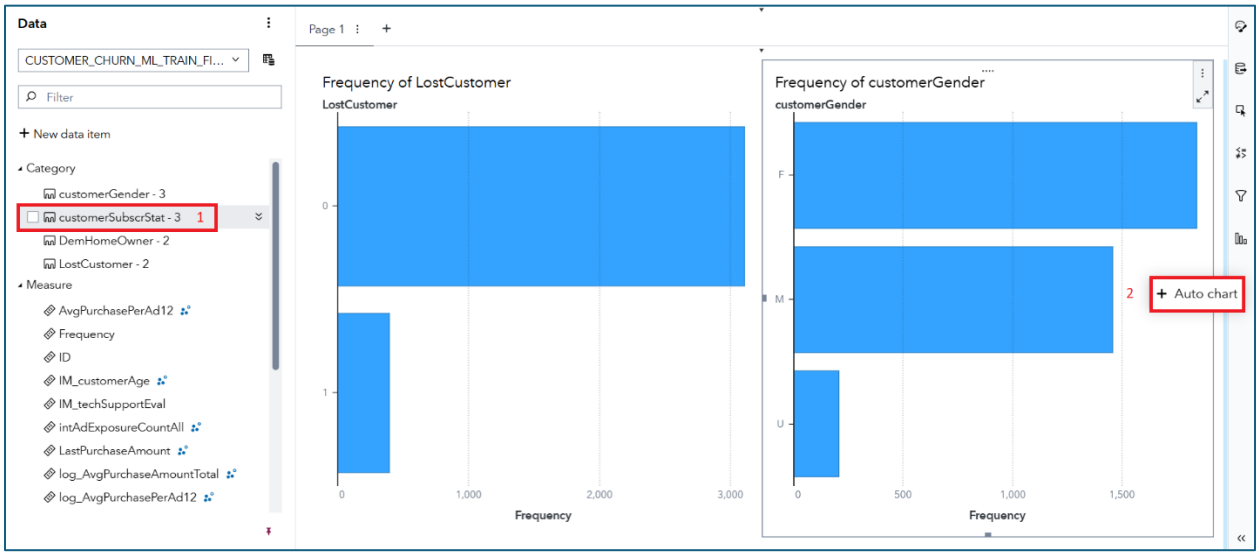

- Our dashboard is evolving!



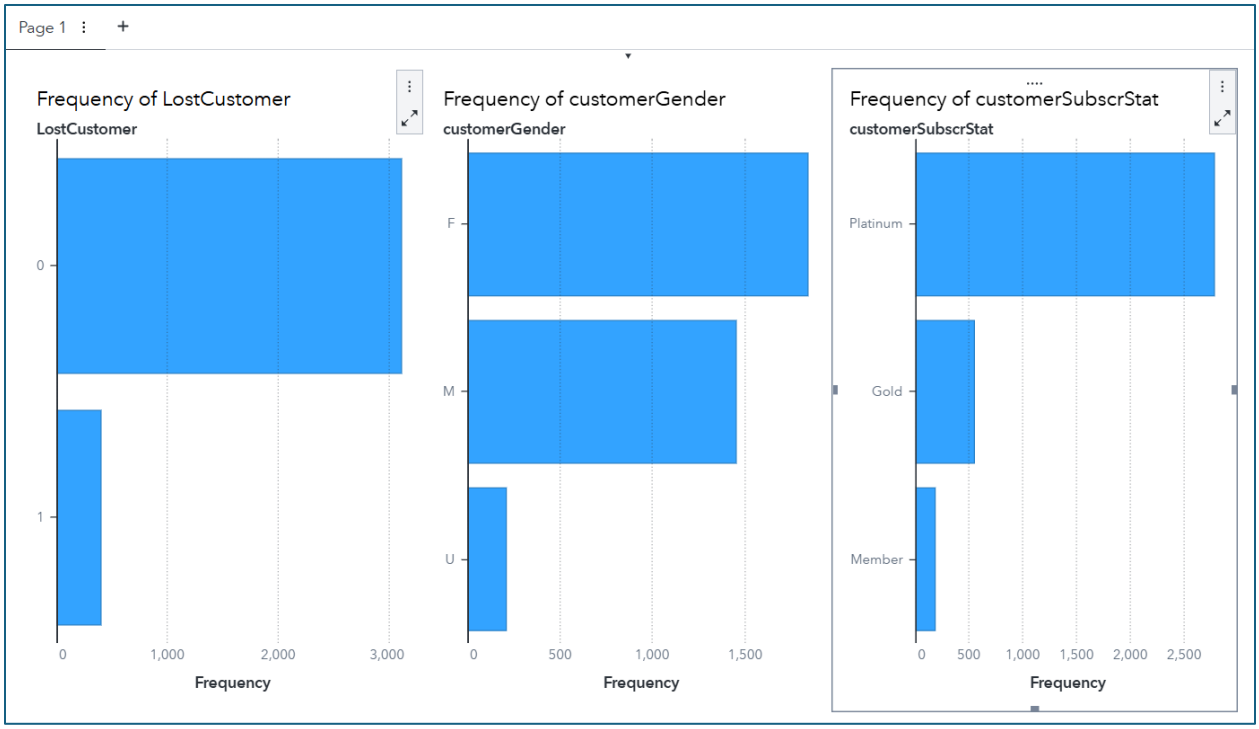

- And two notes before we add even more: (1) keep saving periodically. Like now. And, (2) drag-and-drop mistakes happen. Let the **Undo** button be your friend, when needed:




- Next, find **customerSubscrStat** and add it to the right of *customerGender*. Like this:

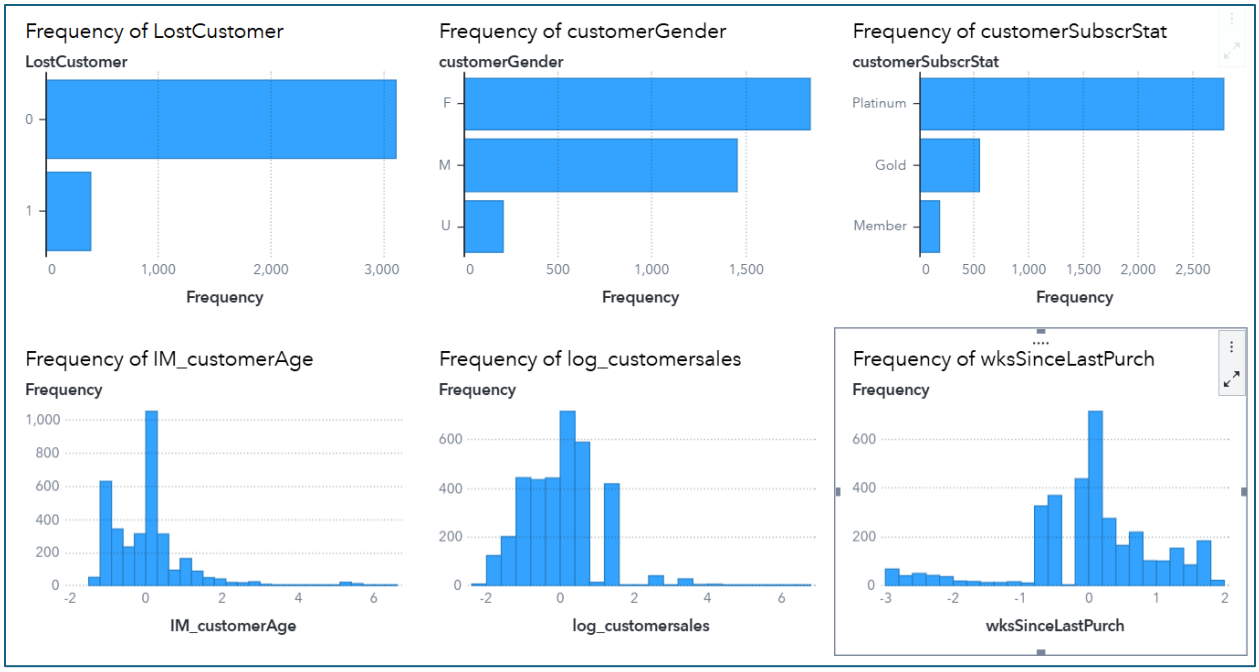



- More details are included on our page!

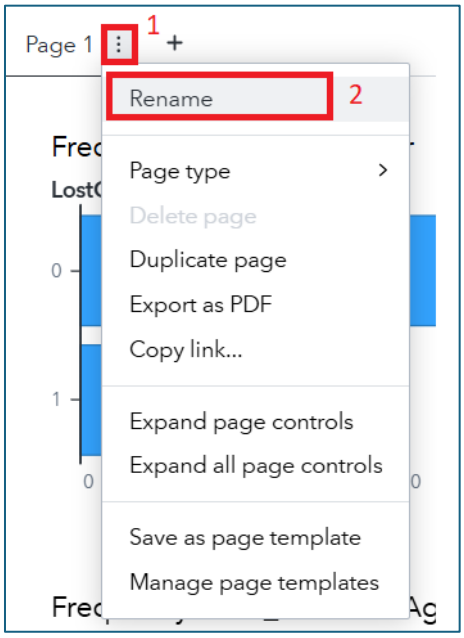



- Those are some helpful distributions on our categorical variables. Now, let's add three **Measures**, just for fun. Find **IM_customerAge** and put it below **LostCustomer**, here:

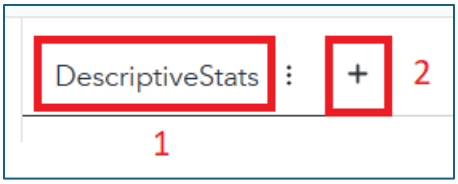

- Hmmm. Those are some strange values for age. They must be standardized somehow. Regardless, let's also add **log_customersales** and **wksSinceLastPurch** to the bottom row. In other words, replicate the following:



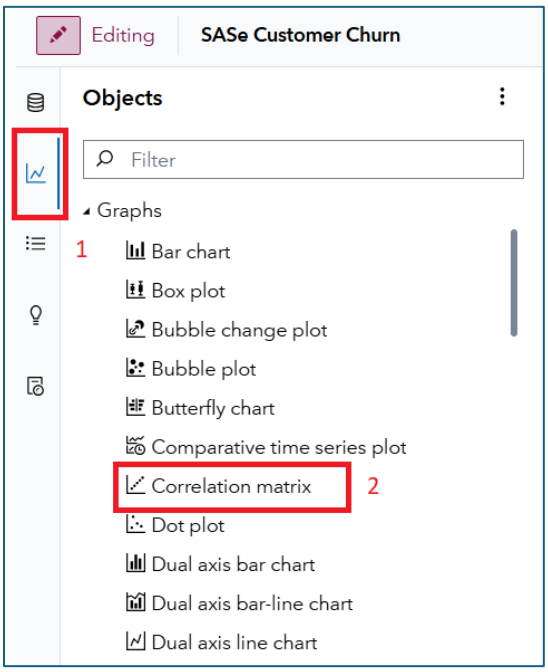

- Again, the measures appear to be standardized. But, the larger point is this: with just a couple of drag-and-drops, your dashboard is off to a great start! Save to keep it around for a bit. Then find the **Option** menu on the page tab. Next, select **Rename**, like so:



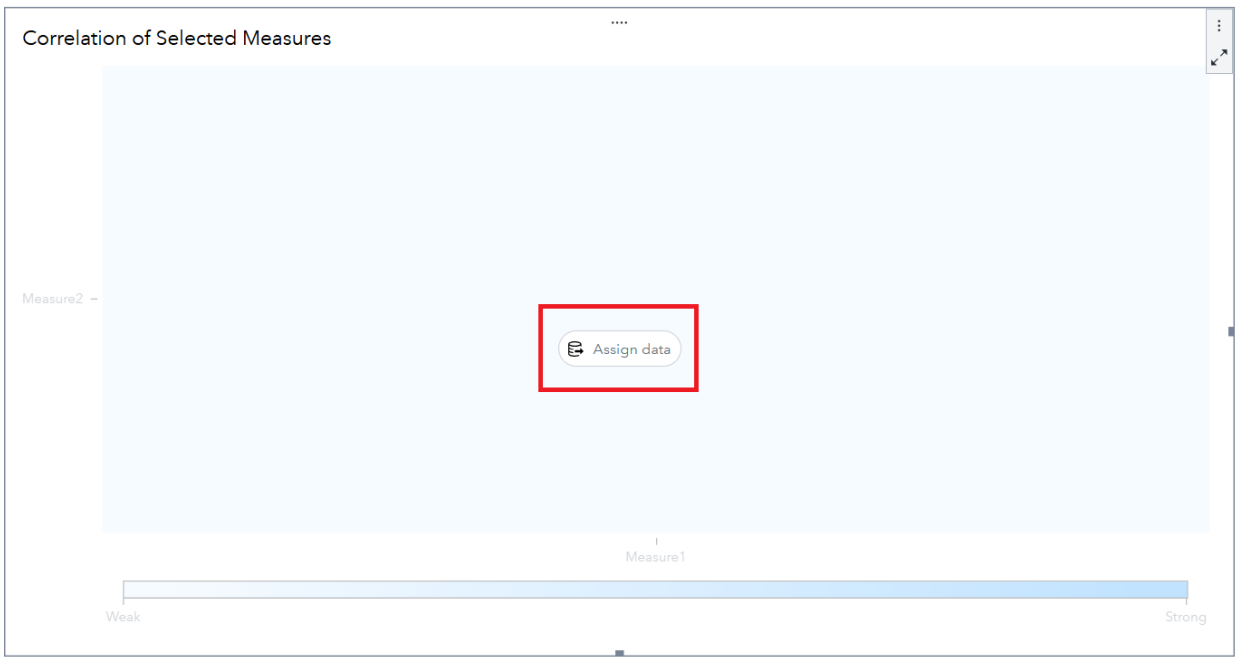

- Change Page 1 to something a bit more helpful like **DescriptiveStats**. Then click the **New Page** button, here:



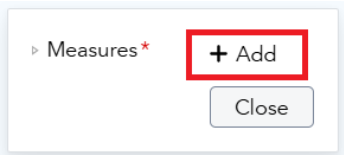

- We've got a new page to explore! And I'd like you to activate the **Objects** pane, then find **Correlation matrix** under **Graphs**. Like so:




- Go ahead and drag-and-drop **Correlation matrix** onto your page. You'll then be prompted to **Assign data**:

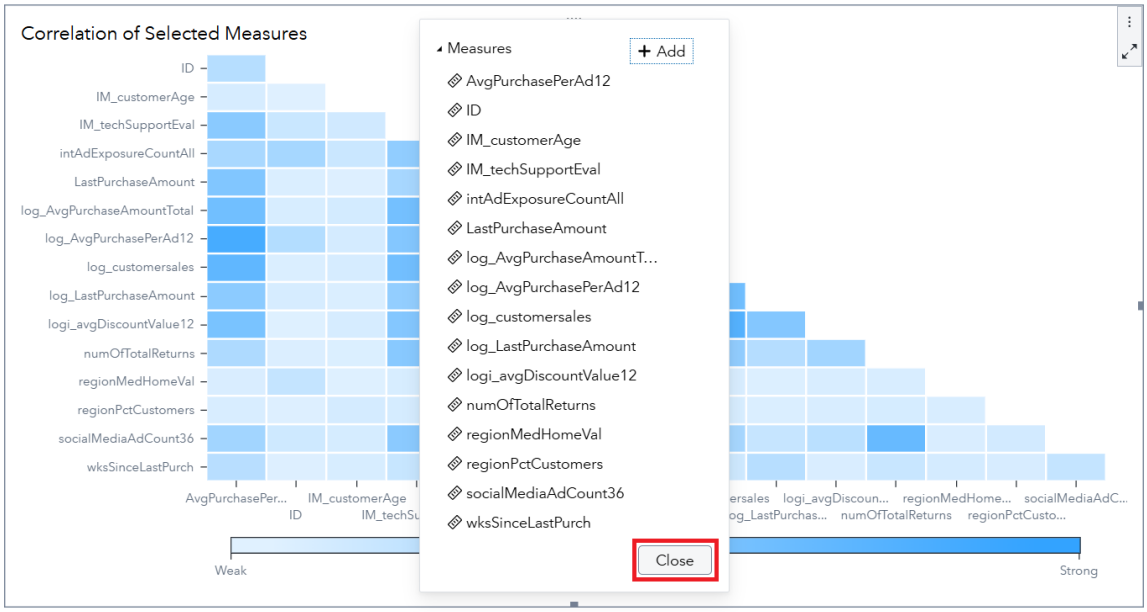


- So, let's assign away! For **Measures**, click **+ Add**:

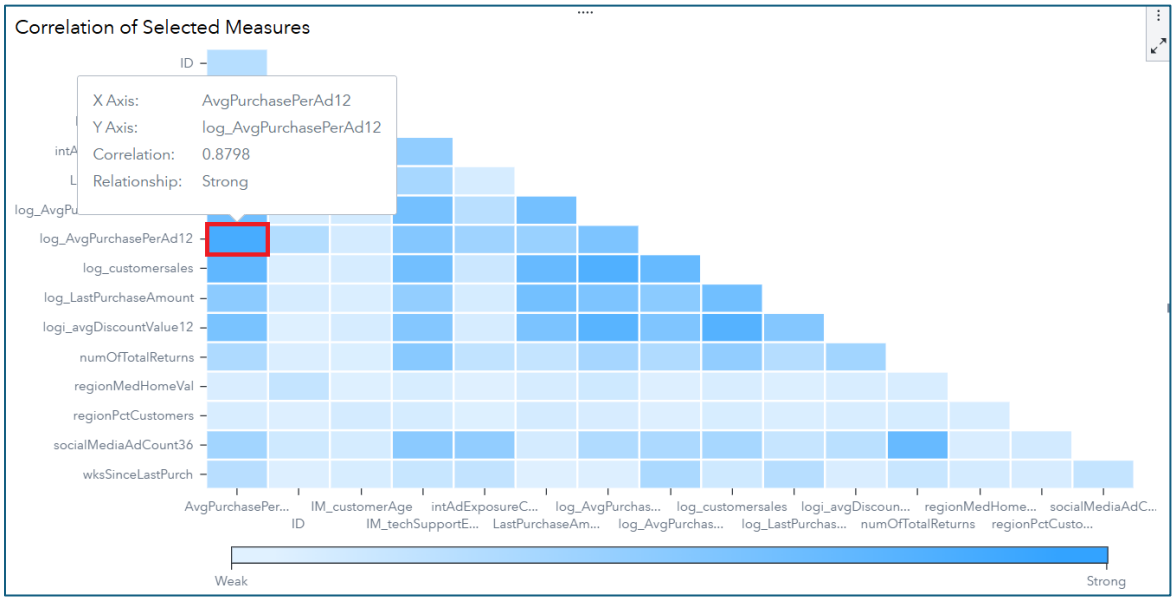



- And, just for fun, let's see it all. Click that **Select all** button

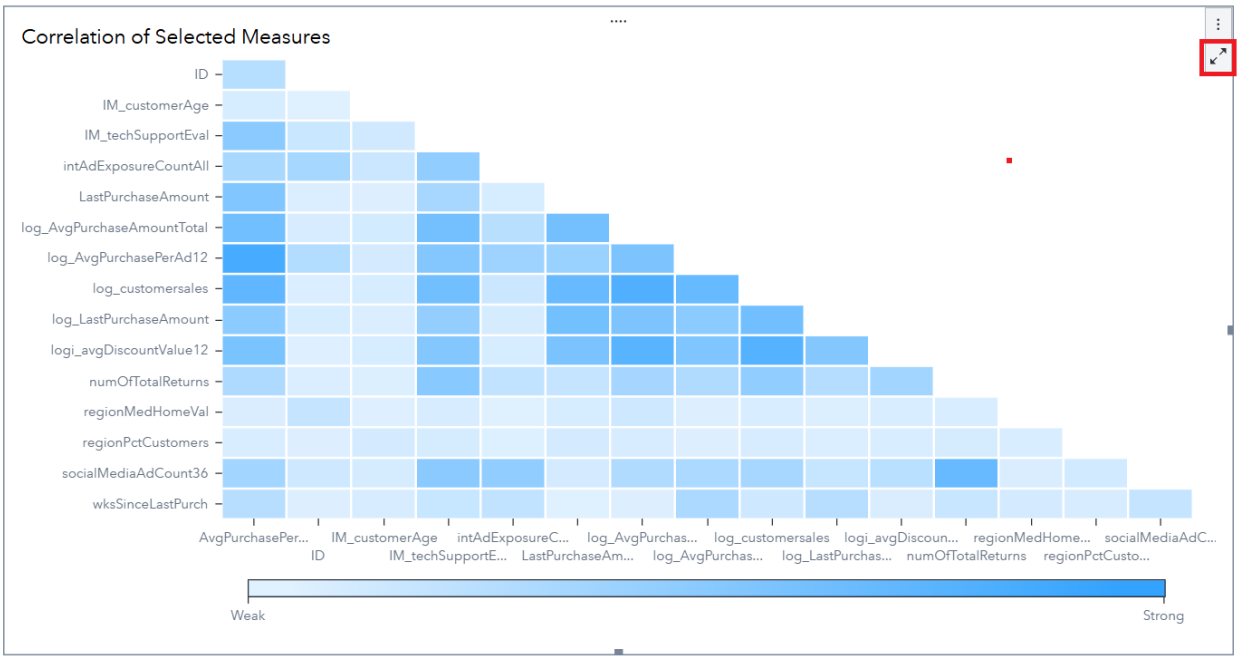

- Then select **Apply**. You can then click **Close** for the pretty graph:



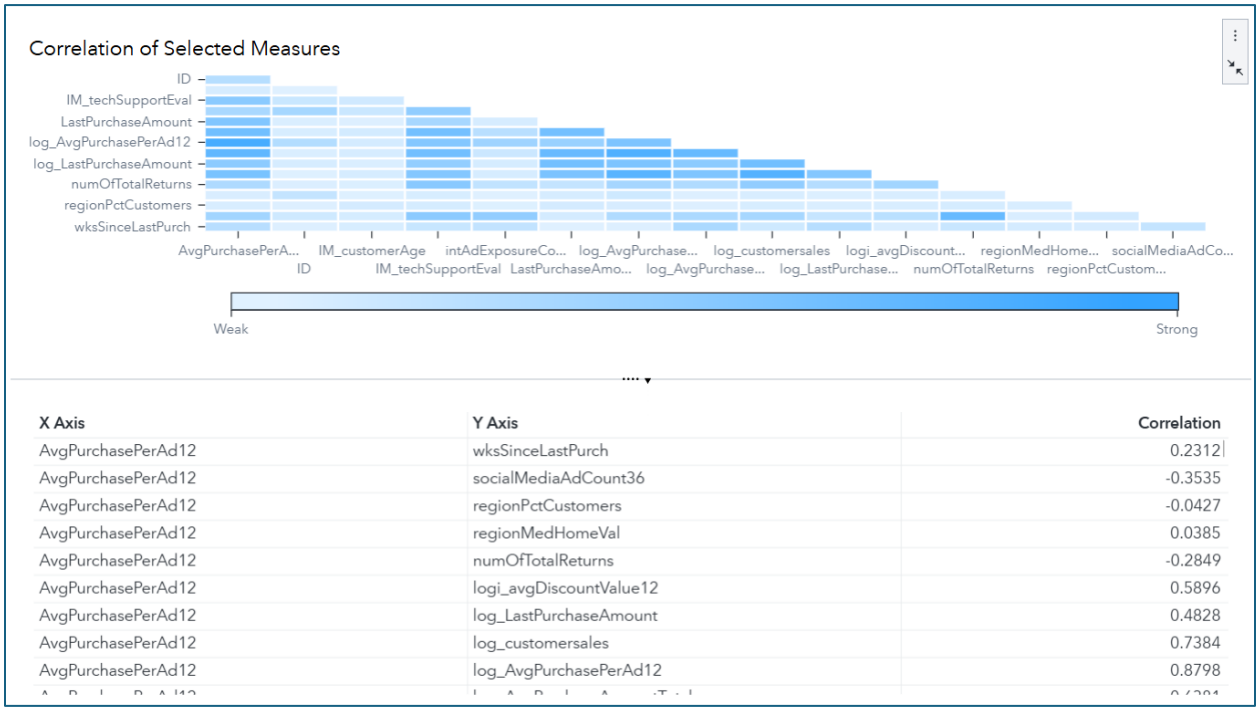

- Spend a bit of time processing the correlations. To learn more about a specific relationship, you can hover above a cell to get more information. For example:



- Want the individual values? Click the **Maximize** button in the object, here:

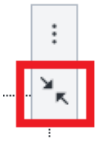



- You can now digest a whole bunch of bi-variate relationships at the bottom:

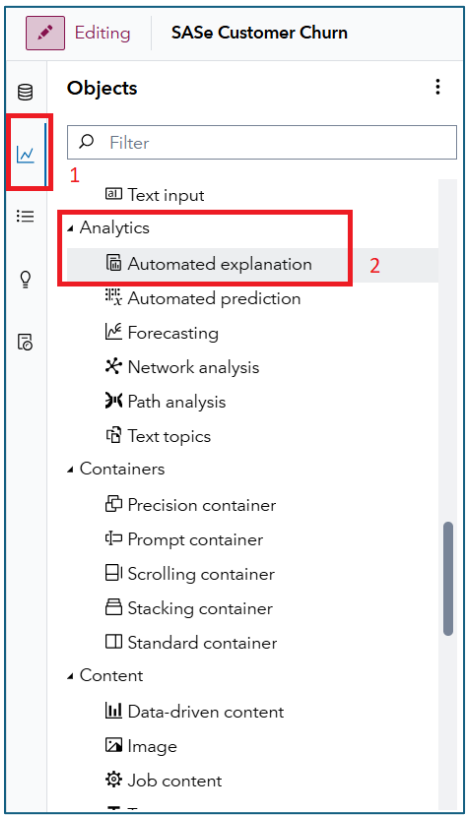



| X Axis | Y Axis | Correlation |
|---|---|---|
| AvgPurchasePerAd12 | wksSinceLastPurch | 0.2312 |
| AvgPurchasePerAd12 | socialMediaAdCount36 | -0.3535 |
| AvgPurchasePerAd12 | regionPctCustomers | -0.0427 |
| AvgPurchasePerAd12 | regionMedHomeVal | 0.0385 |
| AvgPurchasePerAd12 | numOfTotalReturns | -0.2849 |
| AvgPurchasePerAd12 | logi_avgDiscountValue12 | 0.5896 |
| AvgPurchasePerAd12 | log_LastPurchaseAmount | 0.4828 |
| AvgPurchasePerAd12 | log_customersales | 0.7384 |
| AvgPurchasePerAd12 | log_AvgPurchasePerAd12 | 0.8798 |

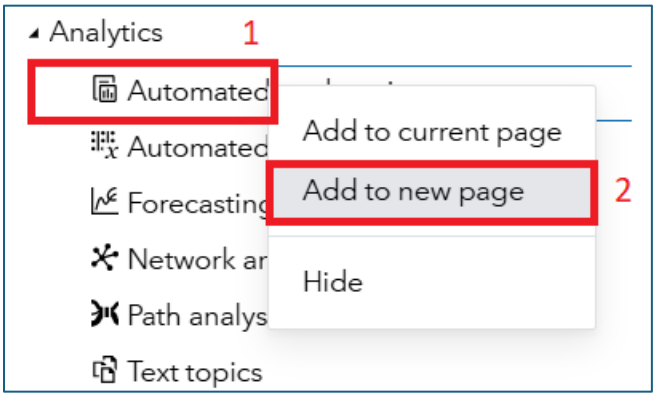
- Restore the view when you're done by clicking here:



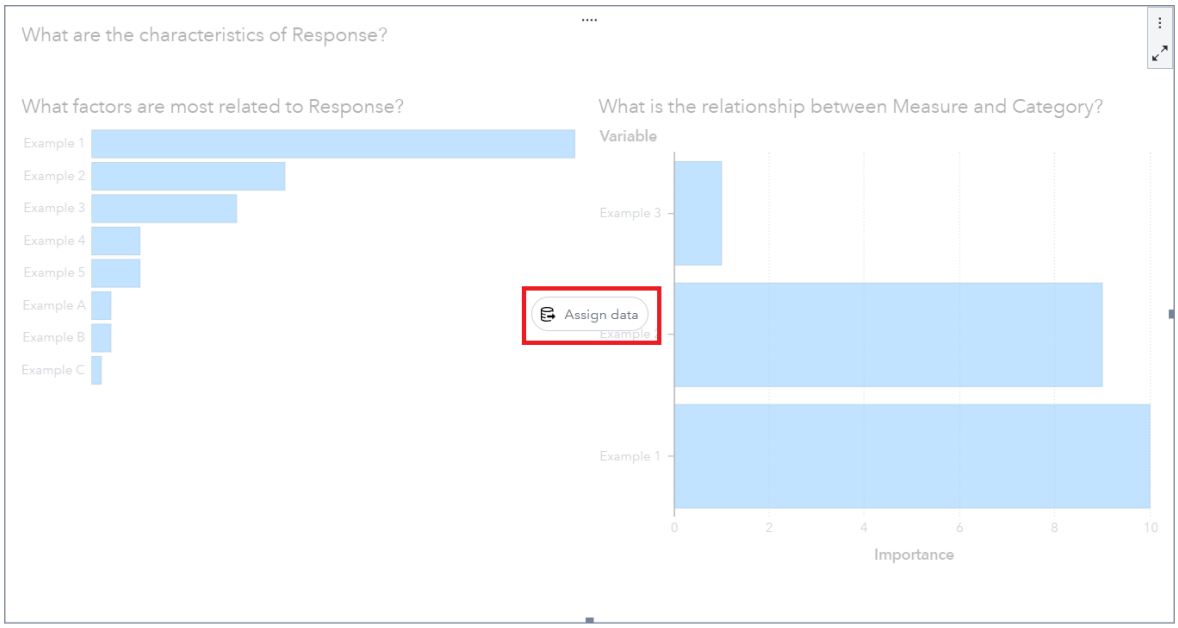
- And rename that page to “CorrelationAnalysis”. Then save. Please 😊
- For our next trick, I’ll show you how cool the **Automated Analysis** object is. To start, ensure the **Objects** pane is active. Then scroll down to **Analytics**, **Automated explanation**. Some guideposts to get you there:



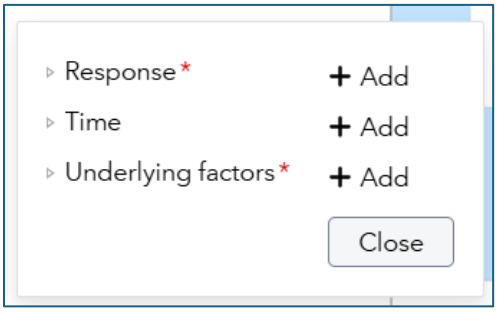

- And while you could drag-and-drop it onto the page, lemme show you another way. Right-click on **Automated explanation** and then select **Add to new page**:



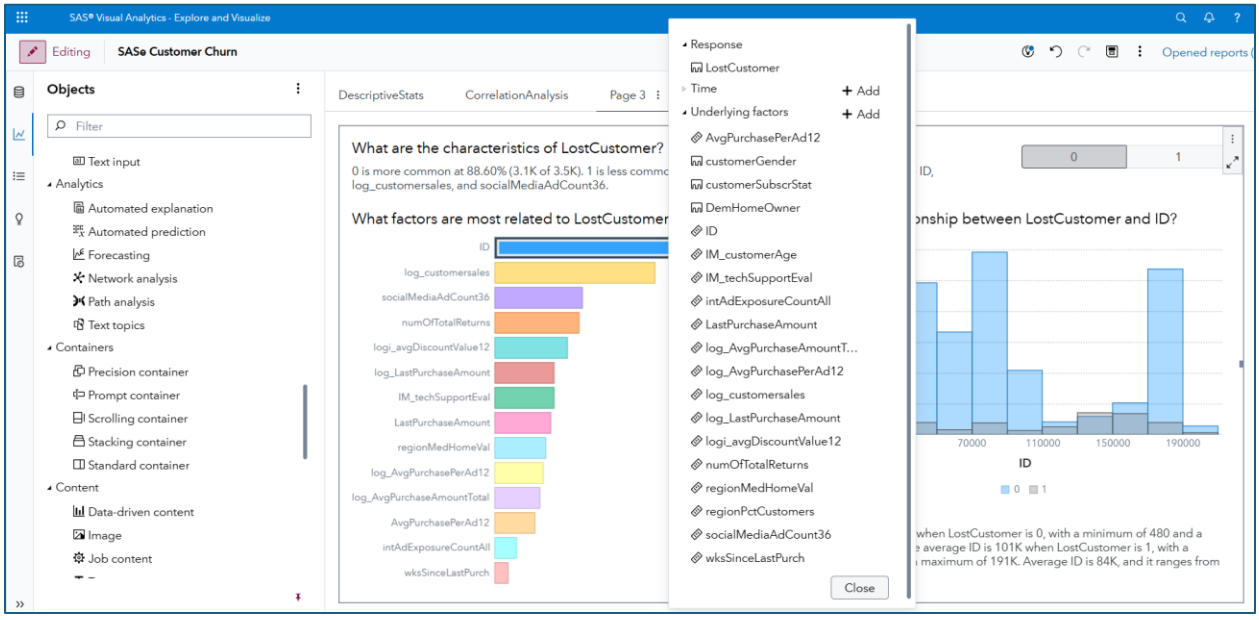

- Presto! A new view eagerly awaits. Click **Assign data**:



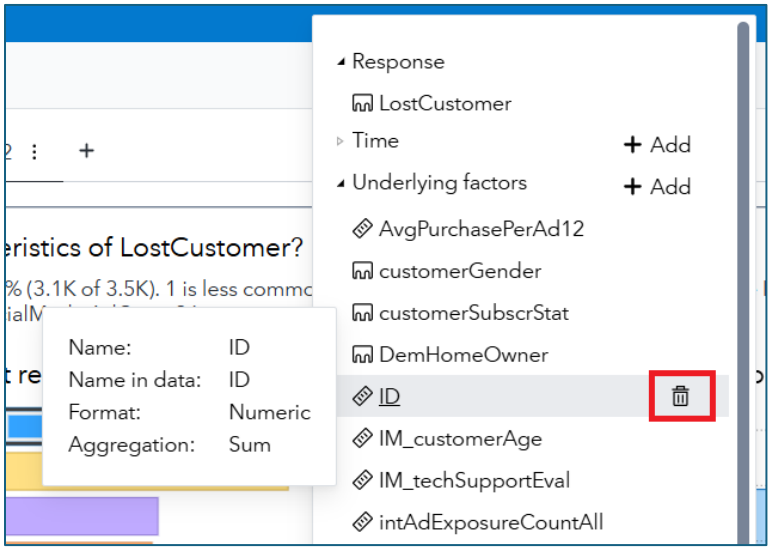

- You'll get the following box:



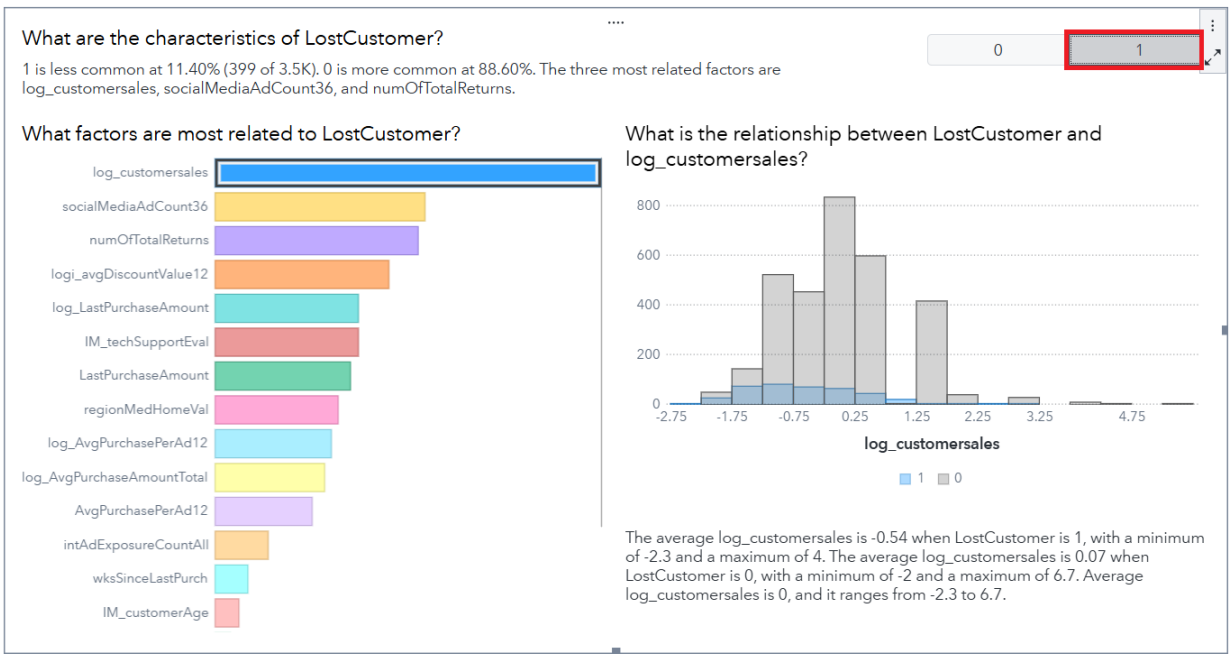

- Click **+ Add** for **Response**. Then select **LostCustomer**. And before you can blink the following appears:



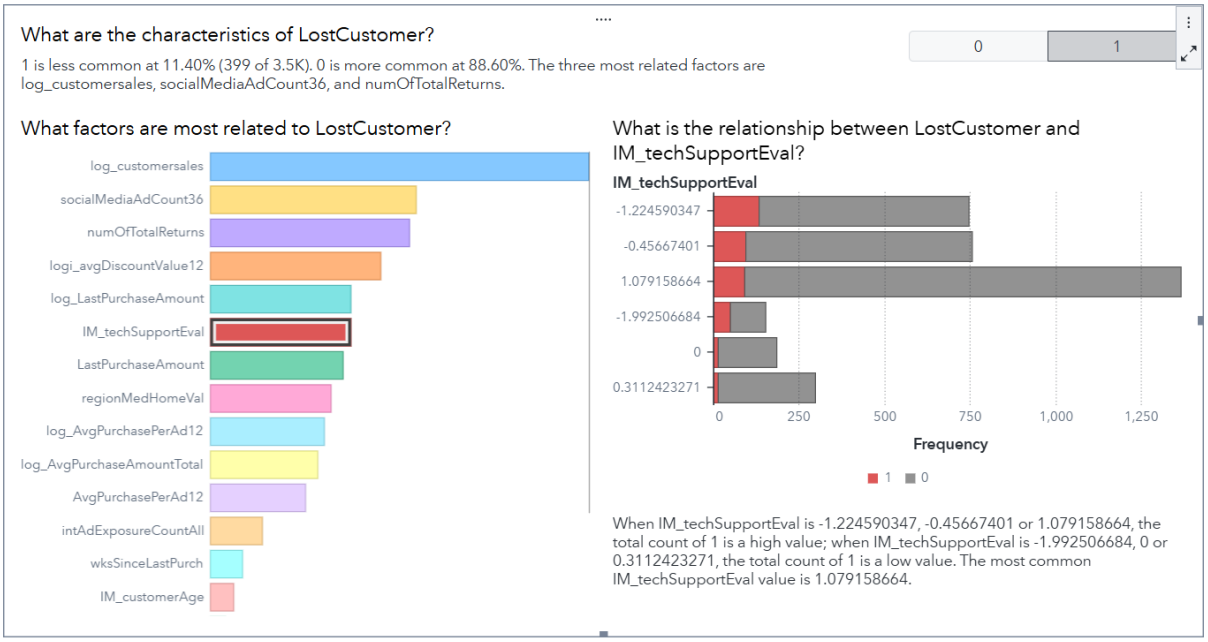
- All the response variables are included in the initial analysis. And know they're easily modified. For example, find **ID** and then click the Remove icon:



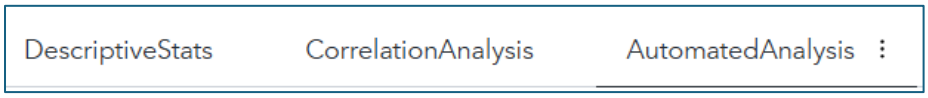

- Click **Close** and then we can digest some output. Actually one more thing – select 1 as the outcome we want to analyze and then start playing with your interactive dashboard:




- What do you see? I find that **customersales** is highly related to customer churn. **socialMediaAdCount** and **numOfTotalReturns** rounds out the top 3. Additionally, when I click on **IM_TechSupportEval**, I see that *LostCustomer* is more highly correlated with a low *TechSupportEval* score:

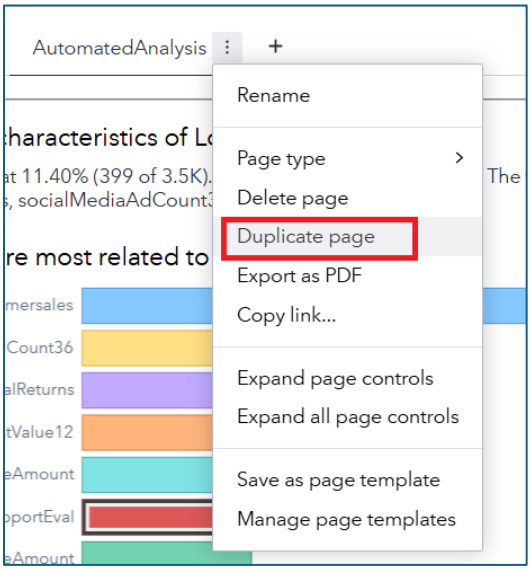



- Interesting. What do you see? Explore a bit more. Then **Save**. And we can also change the name of the page to **AutomatedAnalysis**, like so:

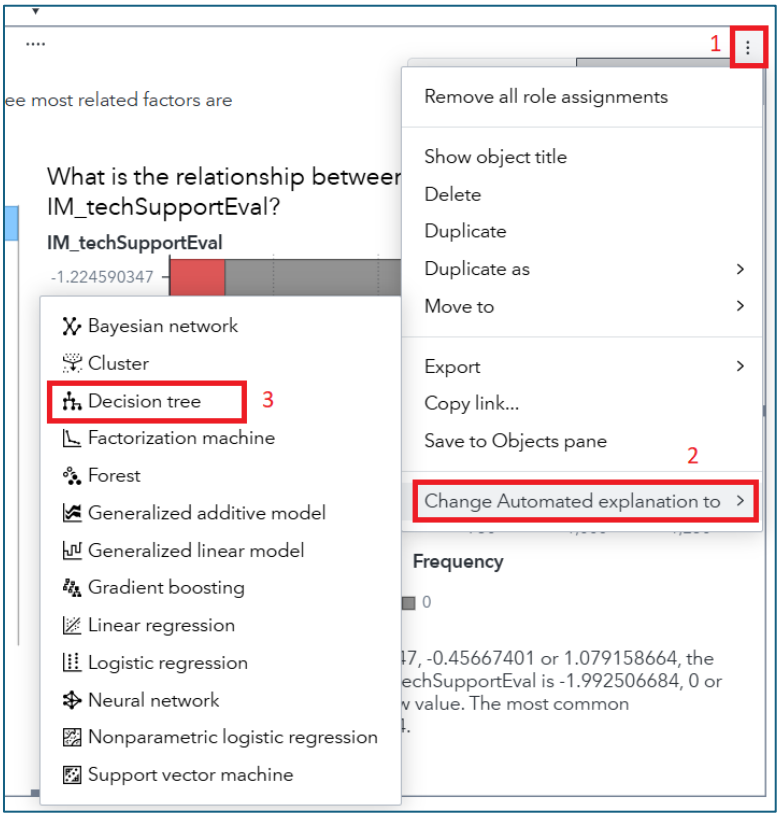



- Three pages down in our dashboard. Woot! While we have other tools for modeling, let's show you how easily we can incorporate SAS Visual Statistics into our dashboard. Find the **Options** button for the **AutomatedAnalysis** page. Like here:

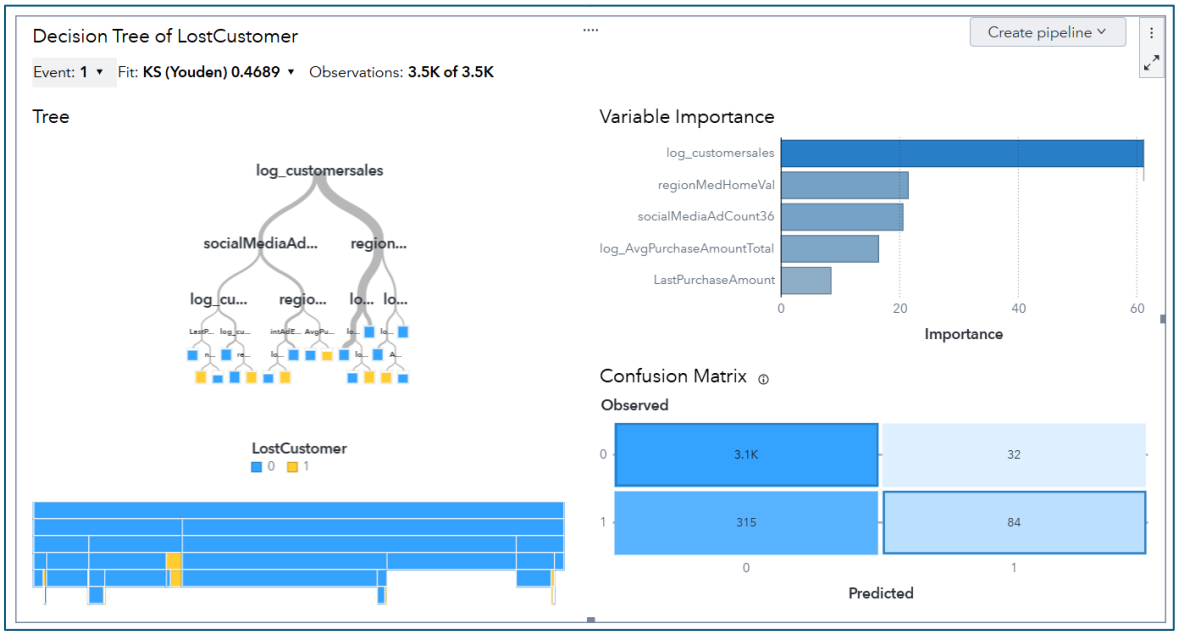

- Now select **Duplicate page**:



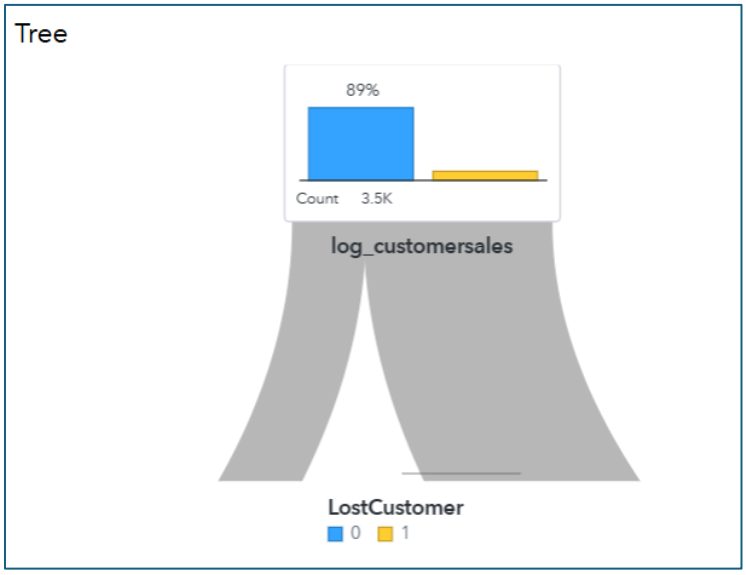
- What have we done? Well, we carried through the existing metadata to a new page. (What is metadata? Well… data about data. So meta!) With just a couple of clicks, we can then change the existing object into a model. Start by clicking the **Object Menu for Automated explanation**. Then select **Change Automated explanation to** > **Decision tree**. Like so:



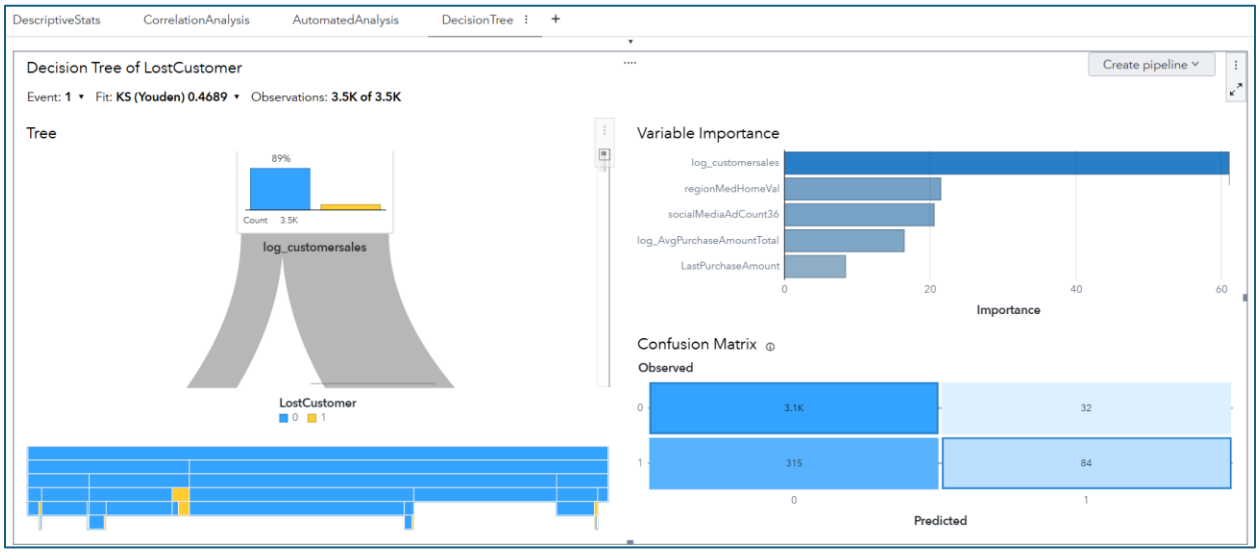

- And, instantly, a decision tree is created!



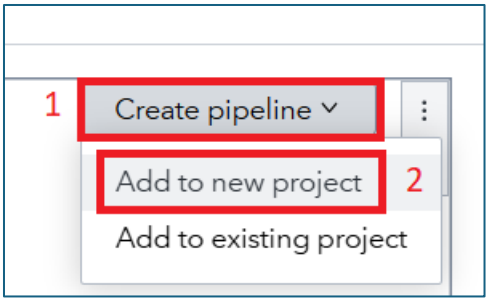
- Like our Automated explanation object, this one is interactive, too! So, spend a bit of time playing with the interface. For example, you can zoom into the decision tree to learn more about the breakpoints:




- After exploring, please **Save**. And then rename the page to **DecisionTree**. And, in just a couple of minutes, you're several pages into your dashboard:




- While we can do a LOT more in SAS Visual Analytics, we'll stop here in the interest of time. Why, because SAS Model Studio is a WAY more powerful modeling tool than SAS Visual Analytics, and I want to get you into that environment, ASAP. Find the **Create pipeline** button and then select **Add to new project**. Like so:

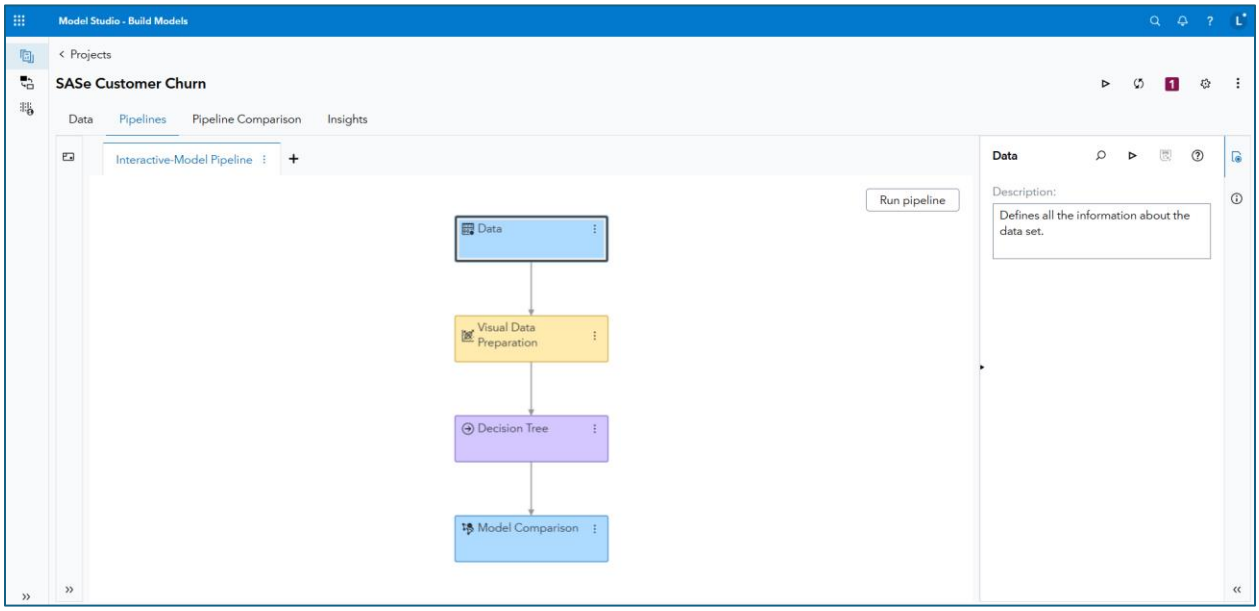



- Your project will be transferred over to SAS Model Studio… and I'll see you there!

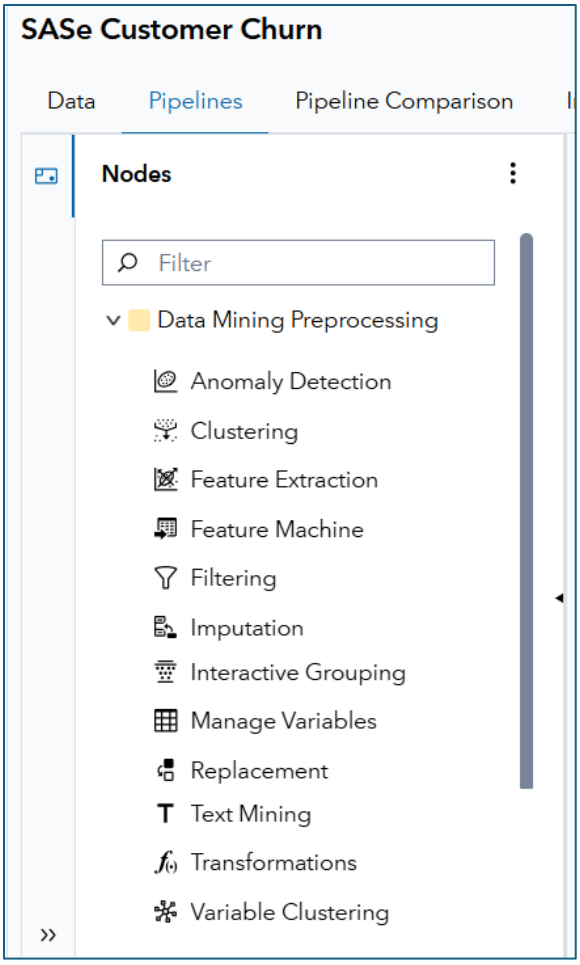
## Modeling in SAS Model Studio

Ready for the wonderful world of SAS Model Studio? If you're just getting started with predictive modeling and machine learning – or a veteran who is simply tired of all that coding – then SAS Model Studio is the place for you! Let's get right back to where we left off in our analysis of LostCustomers in the SASe Customer Churn project!

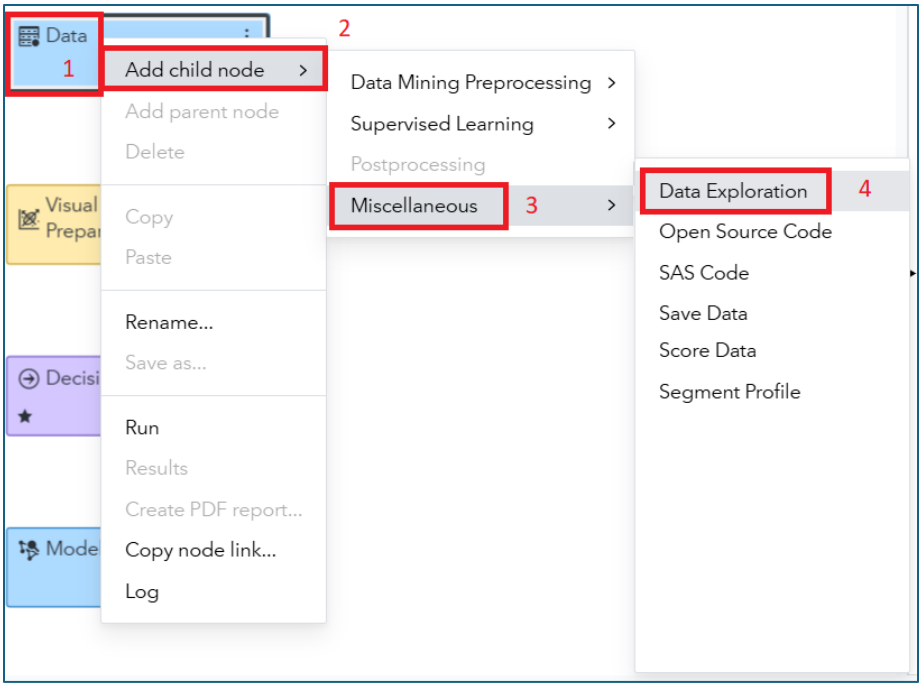
- If everything transferred properly from the Decision Tree in your SAS Visual Analytics report, you should land in the **Pipelines** tab in SAS Model Studio:



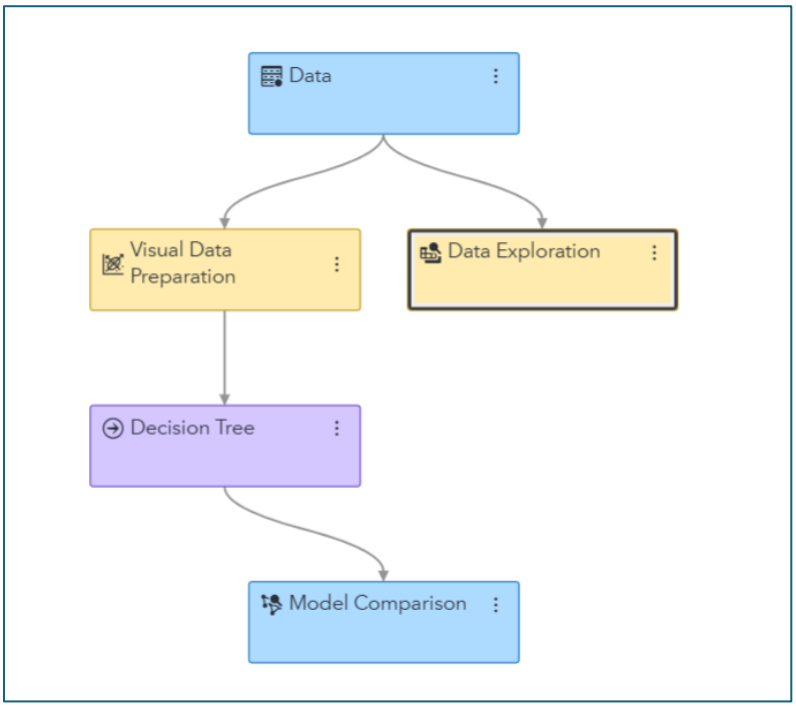

- Pipelines are analytical process flows that take us from data discovery and cleaning to modeling. SAS Model Studio is node-driven and you can see all the options available to you by clicking on the **Nodes** tab. For example, I'll expand the **Data Mining Preprocessing** tab just to show you some of the options available:



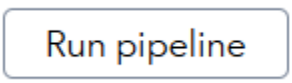

- Yup… that's a lot to explore! But, we're a bit pressed for time, so I'll have you just do one thing to this pipeline. Right-click on the **Data** node. Then select **Add child node** >> **Miscellaneous** >> **Data Exploration**. Like so:



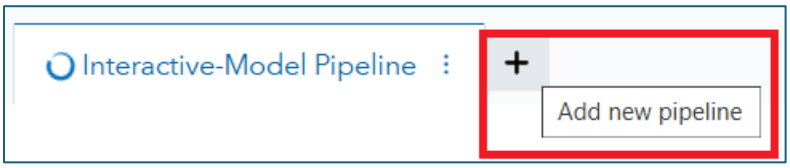

- Your pipeline should appear as:



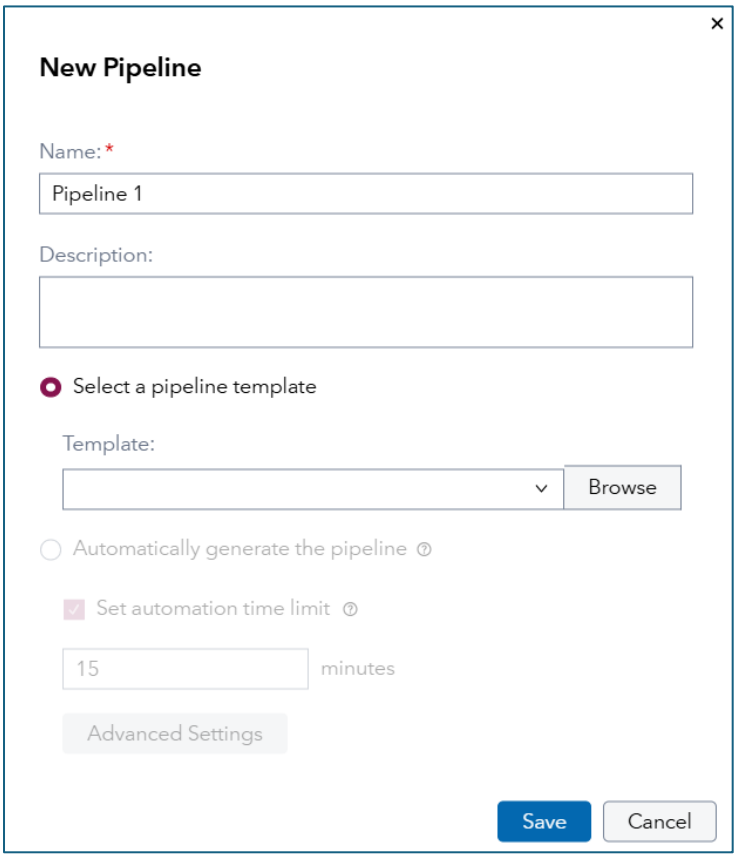

- Then just find and click that **Run pipeline** button:



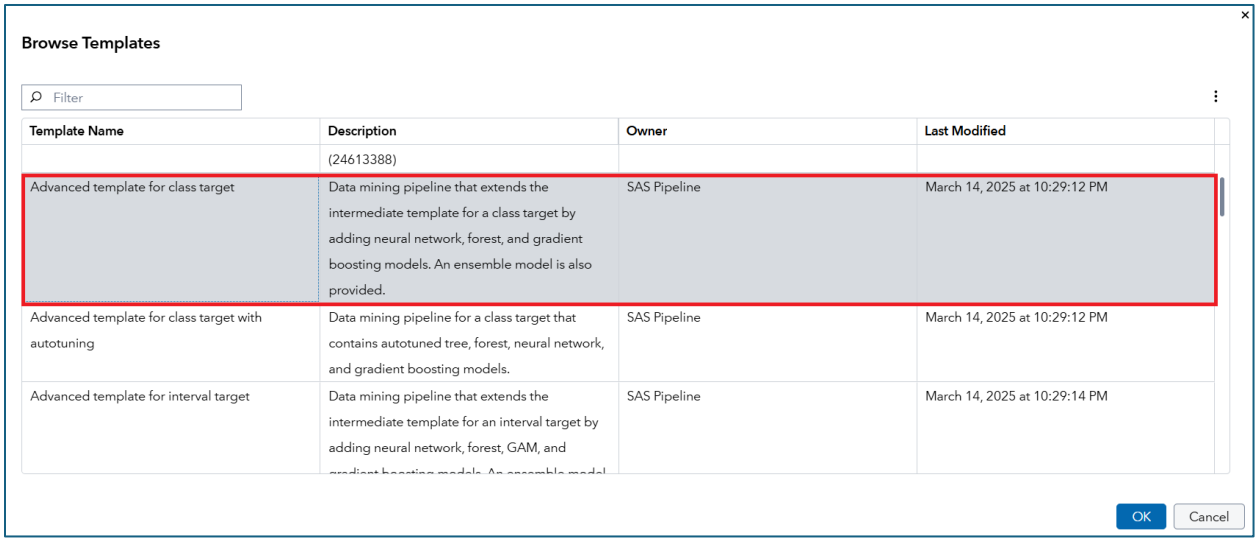
- While that pipeline is running, let's see if we can beat our model from SAS Visual Analytics with one of our pre-built pipelines in SAS Model Studio. Find and click the **Add new pipeline button**:



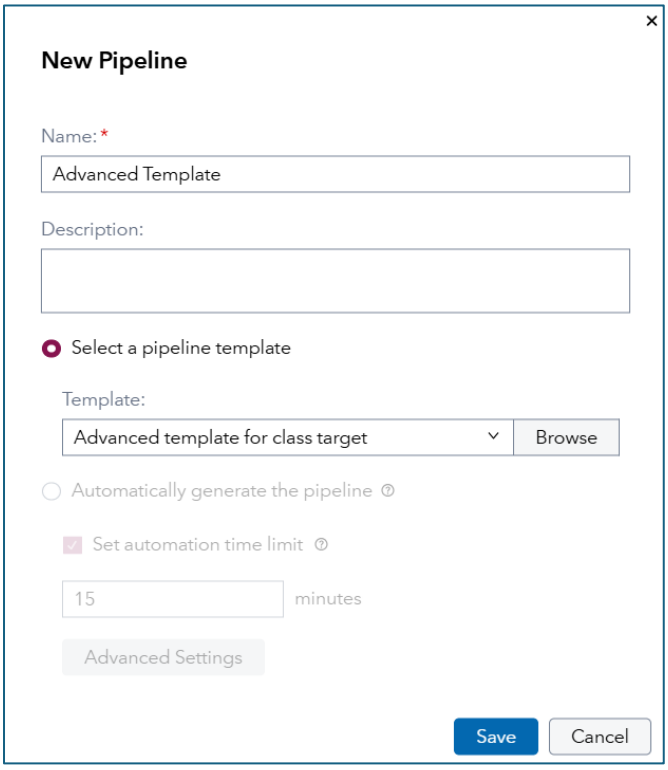

- A **New Pipeline** window appears:



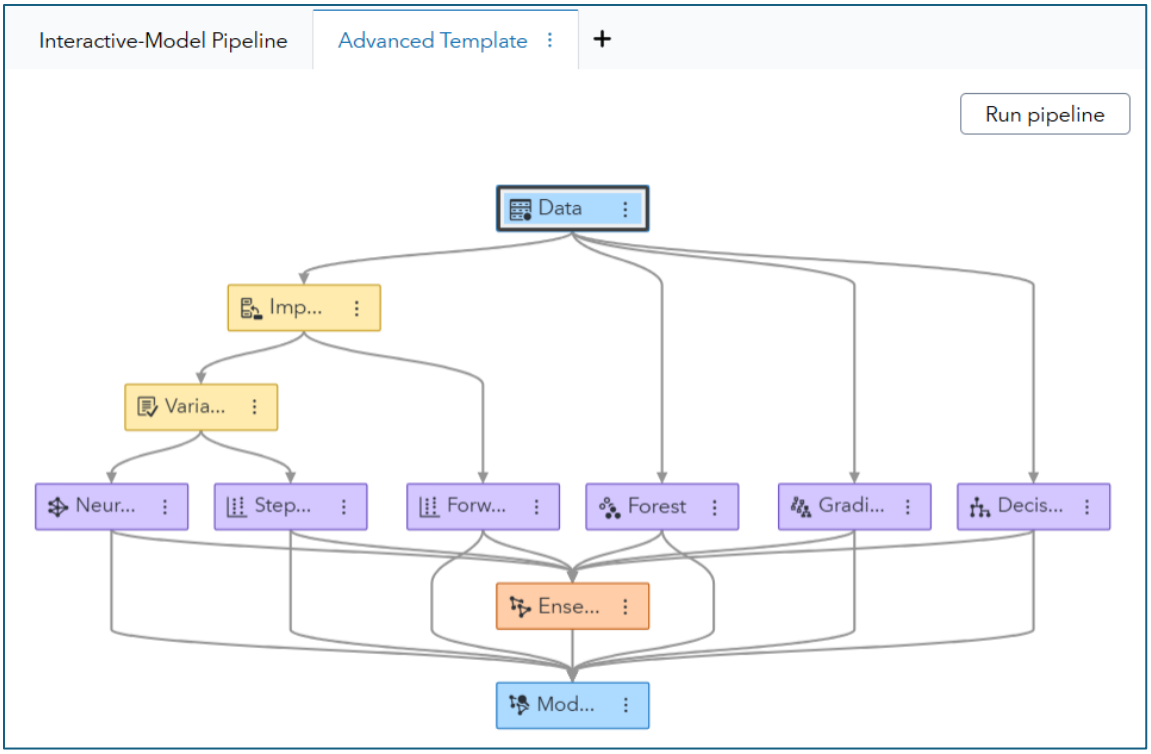

- Let's make a couple of updates to those settings. For **Name**, type **Advanced Pipeline**. Then click the **Browse** button under **Select a pipeline template**. In VFL there may be a LOT of other templates out there. But ignore them and simply find the one that says **Advanced template for class target**, with the Owner of **SAS Pipeline**... i.e., this one:



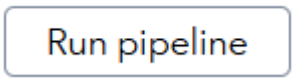

- Click **OK**… which then takes you back to the **New Pipeline** window. With the following cumulative settings:




- Click **Save** and be prepared to be impressed by your new pipeline:




- Do you see what I see? 7 advanced predictive models… for the price of just a couple of clicks. One more gets the pipeline to run:




- Run it! And while it's doing a TON of data preprocessing and then modeling, let's slide over into our coding adventure. Don't worry, we'll return to SAS Model Studio to digest the results!

## SAS Viya Workbench

Where my coders at?

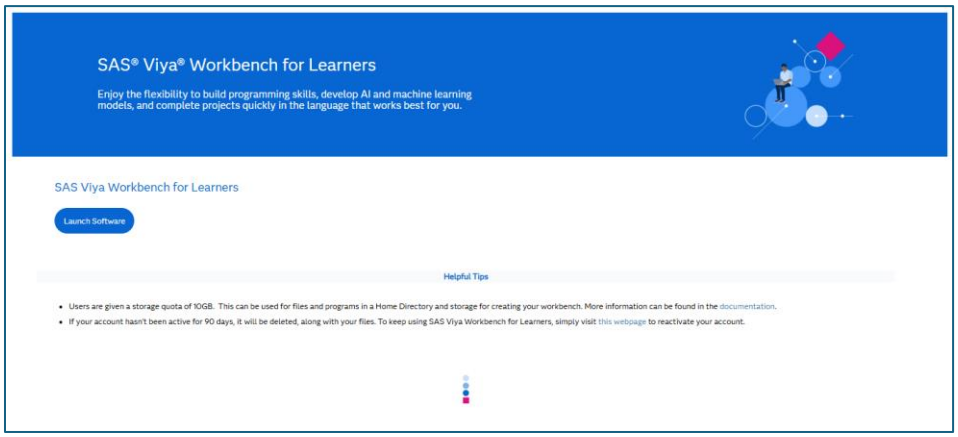
And by coders, I mean SAS, Python, and/or R coders!

Well, this next part of our SASe data exploration is just for you! And it's going to require us to slide into another software offering from SAS: SAS Viya Workbench for Learners (WFL). What is SAS Viya Workbench? Well, it's a lightweight, coder focused environment that provides easy access to SAS, Python, and R code… accessed via either Visual Studio or Jupyter.

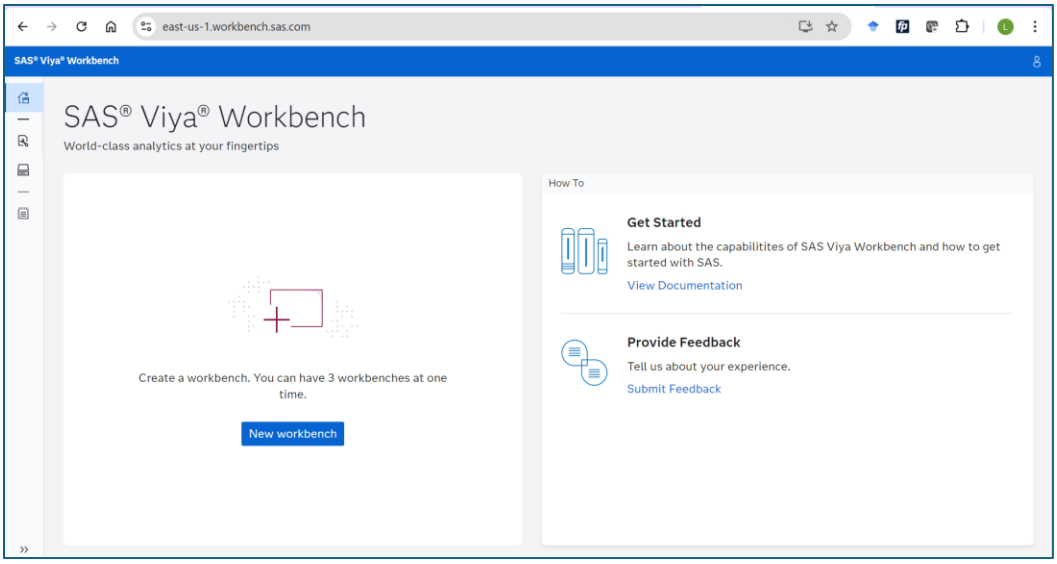
In this section, I'll show you how to access similar code – in both SAS and Python – code that's being automatically generated in SAS Model Studio with just the click of a few buttons. Why? Well, because there is still a HUGE market for coders out there, as coding unlocks the full potential of any coding language. And SAS Viya Workbench will help you do that seamlessly. Plus, I may be able to get some of you interested in taking the Modern Data Science with SAS® Viya® Workbench and Python course, which is nearly 4 riveting hours of content!

But… don't let me get ahead of myself. It's time to get into SAS Viya Workbench for Learners. Let's go!

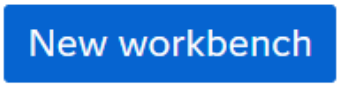
- For the uninitiated, a SAS Viya Workbench for Learners adventure begins here: www.sas.com/workbenchforlearners
- Once you have a SAS profile and accept the terms of use, you should be taken to the **SAS Viya Workbench for Learners** product page: Course: SAS Viya Workbench for Learners. It looks like this:



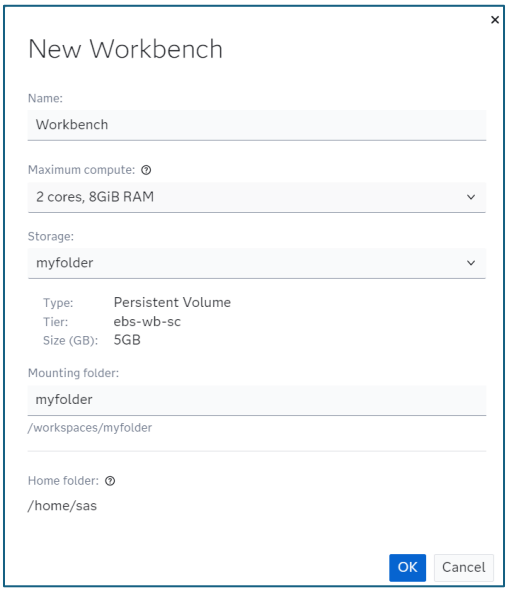

- Click the **Launch Software** button above to get started!
- If this is your first time in SAS Viya Workbench, you'll need to create a **New workbench**. And creation is just a few button-clicks away:



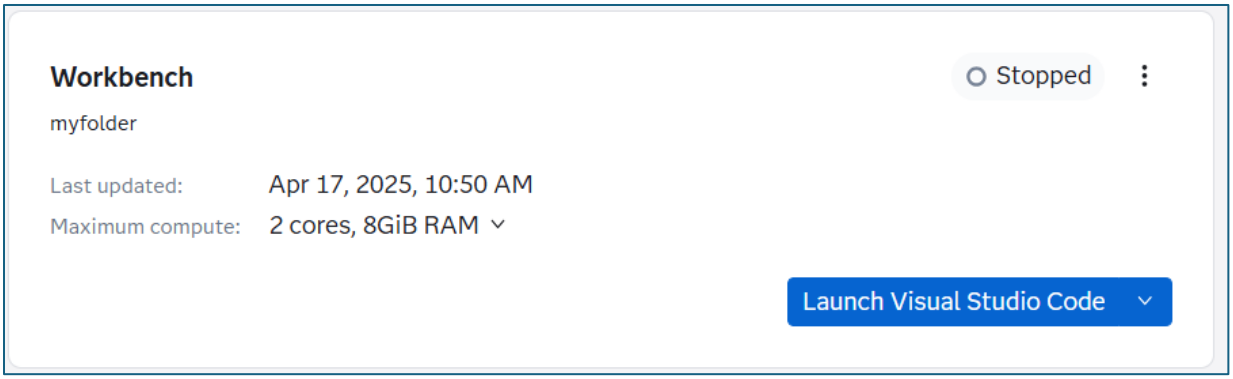

- But before getting right to it, spend a little time getting acquainted with the WFL landing page. For example, check out the **Get Started** resources, or those **Workbenches** and **Storage** icons on the left side. In other words, just become more and more comfortable with the resources available on the WFL landing page. And when you're satiated, click that **New workbench** button:



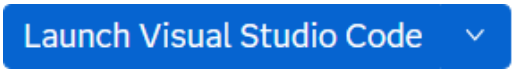

- The following dialog box should appear:



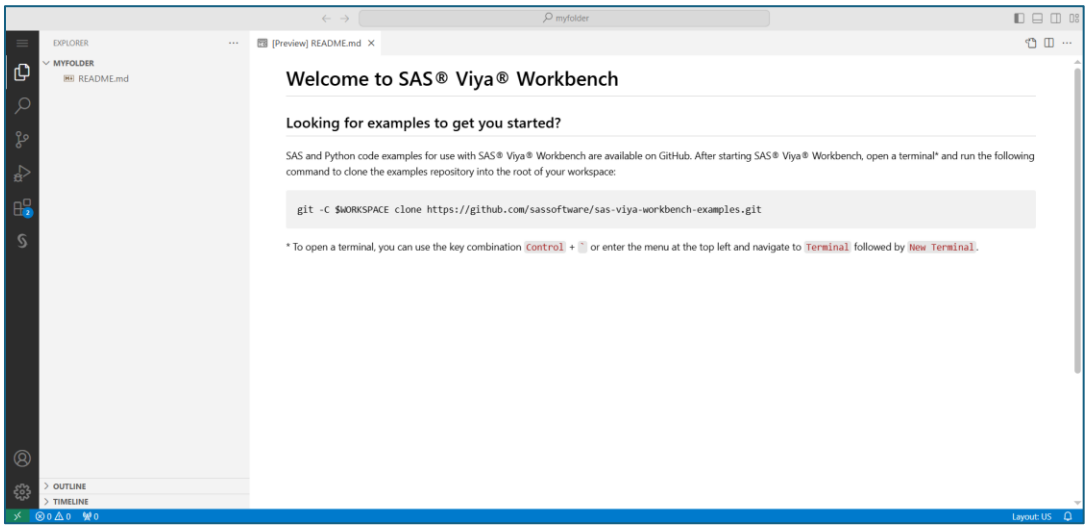

- For **Name**, let's just keep **Workbench** to simplify. And, in fact, just keep the other settings at their default and then click **OK**. Success looks like this:




- Next, in our new workbench, click the **Launch Visual Studio Code** button, here:

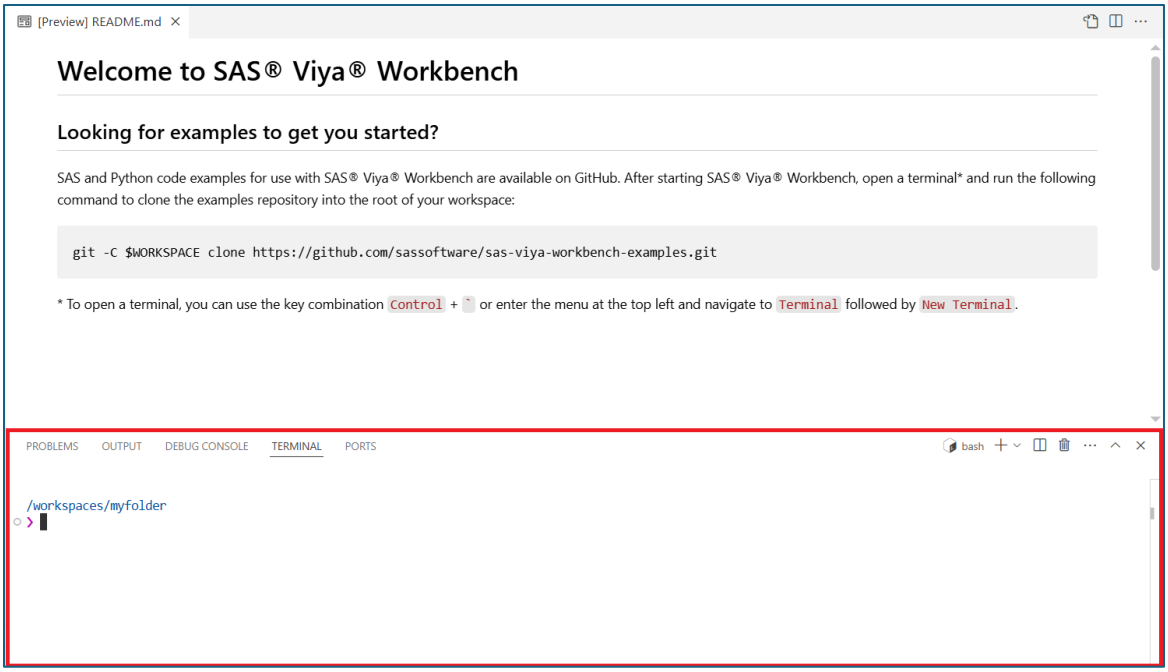



- Your workbench container loads! And welcome to **Visual Studio**!

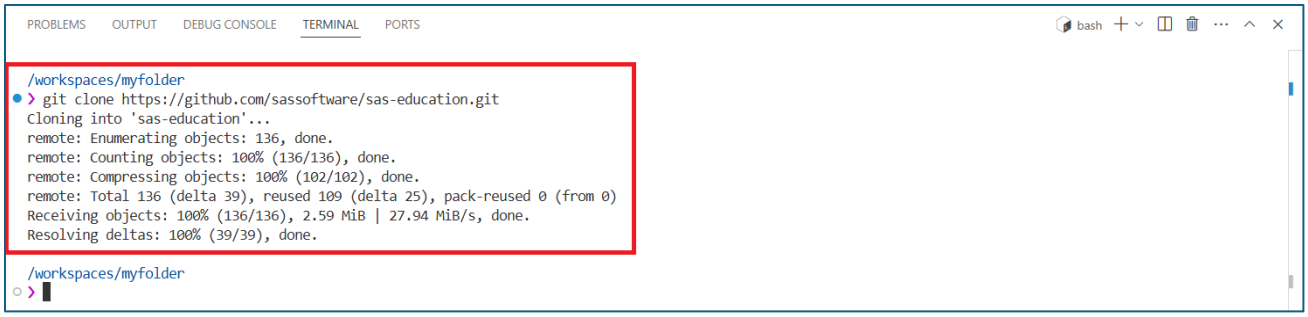



- If this is your first time in the environment, explore the Visual Studio environment for a bit. Because we're all about exploring and learning here.
- Once you're more comfortable with Visual Studio, the final setup task is to access the data for this SAS Guided Demo. You'll want to access the **Terminal** – which is likely hidden by default. No worries – it can be activated via the **Toggle Panel** if it's not already available. And that button is in the upper-right corner, here:

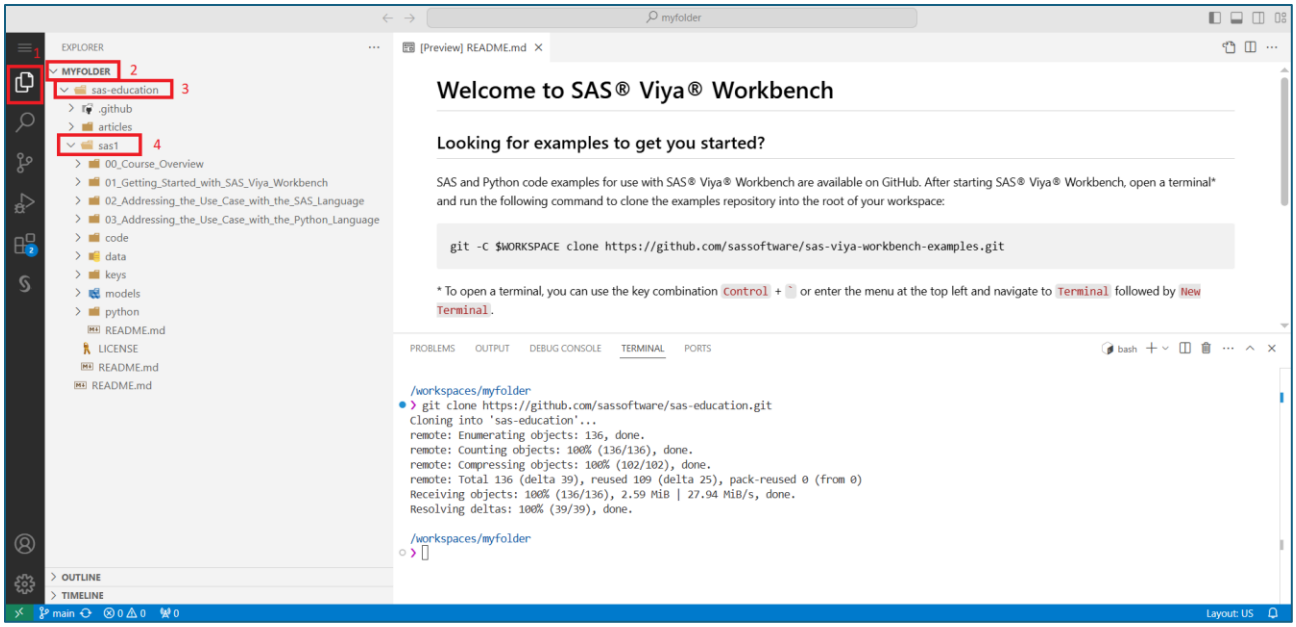
- And a happy terminal will look like the following:




- You're almost there! Simply type the following line of code into the Terminal:
  git clone https://github.com/sassoftware/sas-education.git
- Then hit **Enter**. Success looks like this:



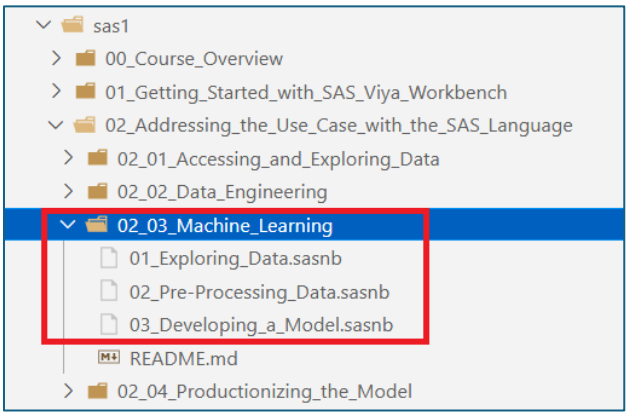

- The files that we'll need for our day 1 adventures at SASe can be found in the **Explorer**. The files are a couple of layers down into the folder, so follow this path: **MYFOLDER** >> **sas-education** >> **sas1**. And here are some click checkpoints to guide you:



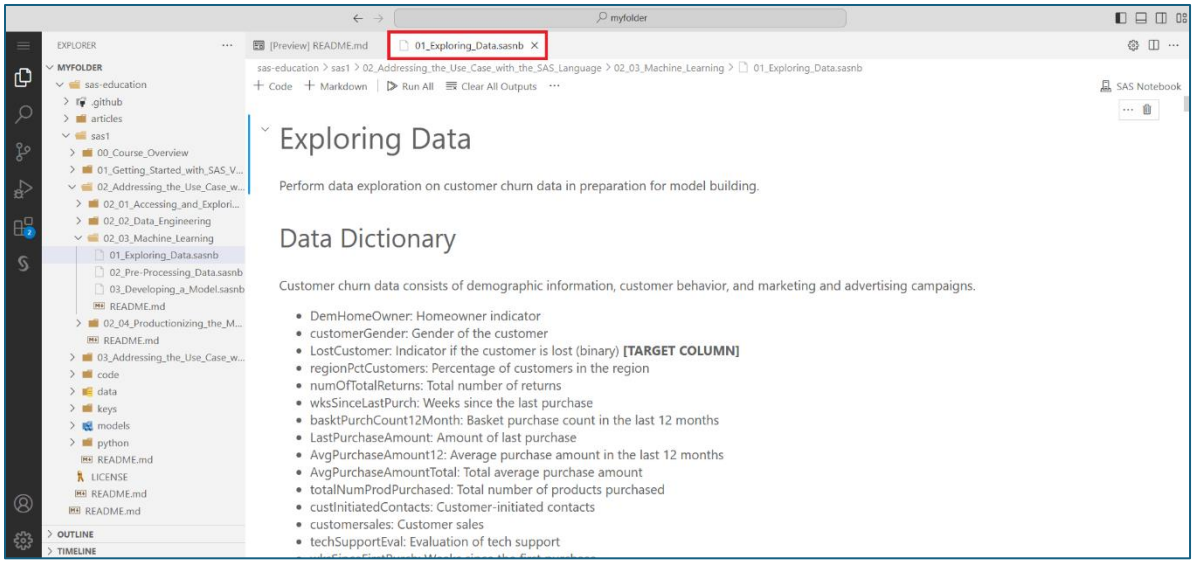

- Files are a great thing to have. And, believe it or not, you are mere clicks away from the awesomeness that is running SAS and Python code in Visual Studio. Without further ado, let's get at it… one by one!

## Modeling in SAS

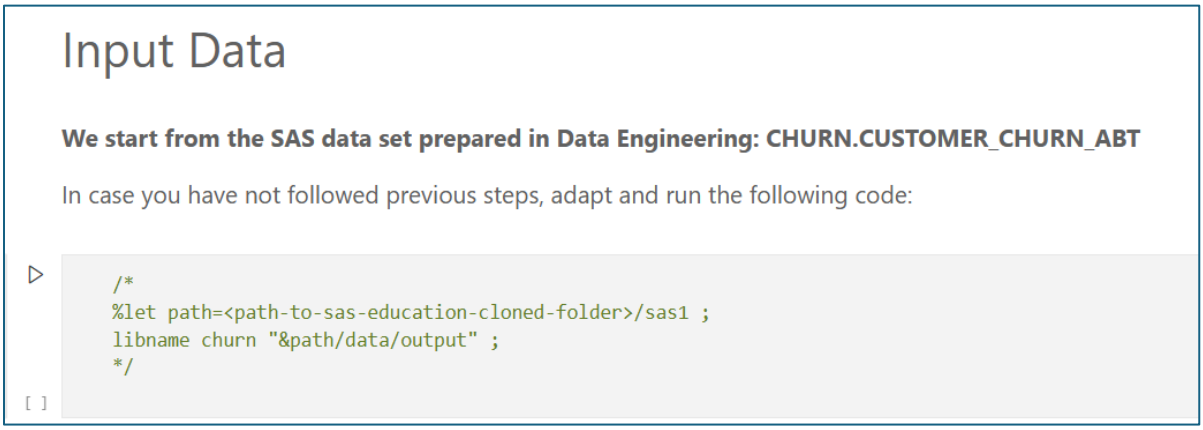
- Open the **02_Addressing_the_Use_Case_with_the_SAS_Language** folder under the **sas1** folder in SAS Viya Workbench. Then find – and expand – the **02_03_Machine_Learning folder**. Three notebooks await!




- And, yes, those numbers denote the order you'll run them in. So, start with *01_Exploring_Data.sasnb*… as in double-click it. You'll see your very first **SAS Notebook** in Visual Studio:



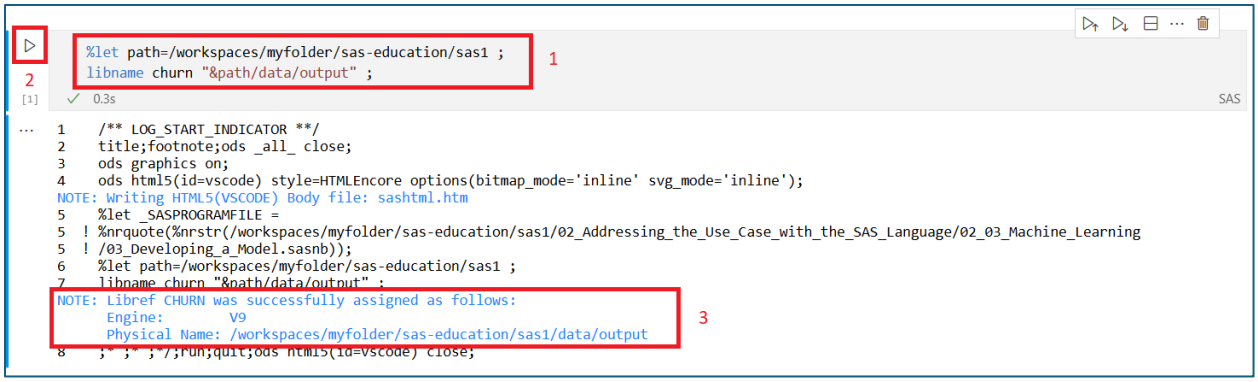

- Nice! Before we can run any of the code, we need to do one more thing. And that's to adjust the location of the **Input Data**, found in this cell:



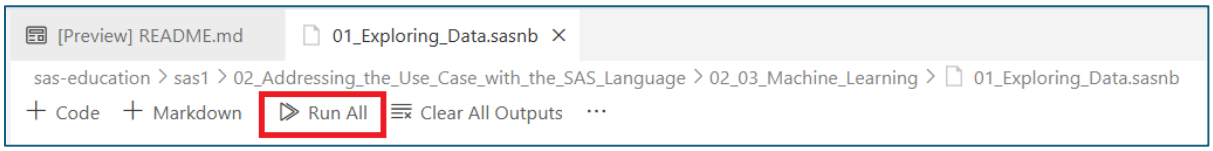

- Click on that cell. Then simply copy and paste the code below to replace:

```
%let path=/workspaces/myfolder/sas-education/sas1 ;
libname churn "&path/data/output" ;
```

- Then click the **Execute Cell** button, which yields – hopefully – a happy log… assuming you set up the Workbench as described above:




- Now, there are a couple of ways to continue this "course preview" party. You can simply click **Run All**... and run all the code, scroll to absorb some random content, and then just move on to the next program. That magical **Run All** button is found here:




- Or... you can actually walk through the program as a "true" learner, which means running the program cell-by cell. This requires navigating to a cell, finding the little **Play** button, and mashing each and every one of those:


- For those not in the know, "mashing" is a southern verb meaning "to push". Local dialects aside, the goal here is to show you an end-to-end data science project – done with two approaches – once in SAS and once in Python. And, as noted, we're leading with SAS.
- Once you're happy with exploring the data, know that we've got 5 more programs to explore: 2 SAS notebooks and 3 corresponding ones for Python. Because we're going to do (roughly) the same thing in both SAS and Python.
- But... I'm not going to simply "punt" and say "have at it". At the very least I can show you the code in each program that replicates something we did earlier in this SAS Guided Demo. So, in *01_Exploring_Data.sasnb*, find the code on Histograms. It appears like so:

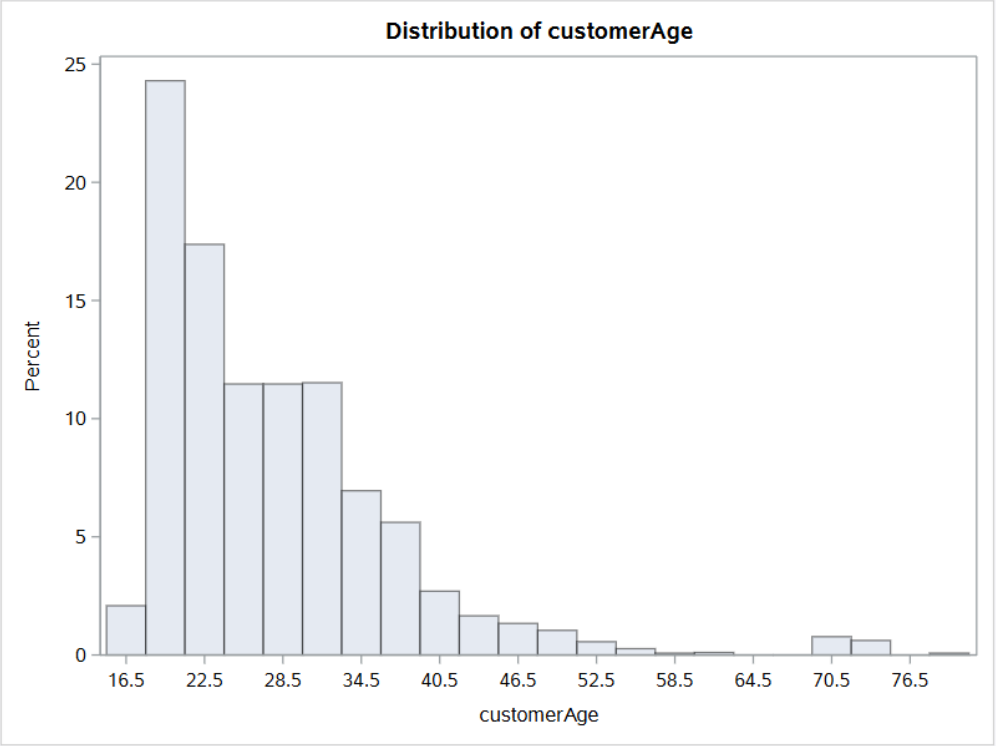
Histograms

Use the Univariate procedure to generate histograms to explore distributions. Use ODS to capture variables with missing values from the Univariate output. This table of missing values is shown later.

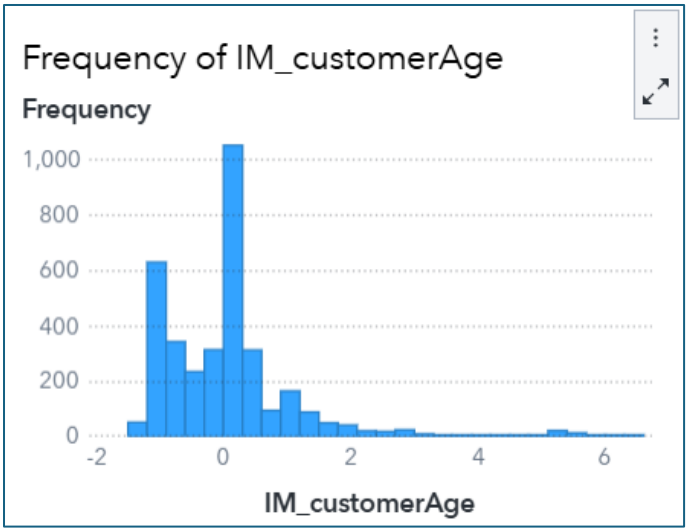
```
/*---------------------------------------------------
 * Use PROC UNIVARIATE to generate the histograms.
 */

TITLE;
TITLE1 "Summary Statistics";
TITLE2 "Histograms";

ods output Moments=Moments MissingValues=MissingValues;
PROC UNIVARIATE DATA=customer_churn;
  VAR &num_vars;
  HISTOGRAM ;
RUN; QUIT;
ods output close;
```

[29] ✓ 5.3s SAS

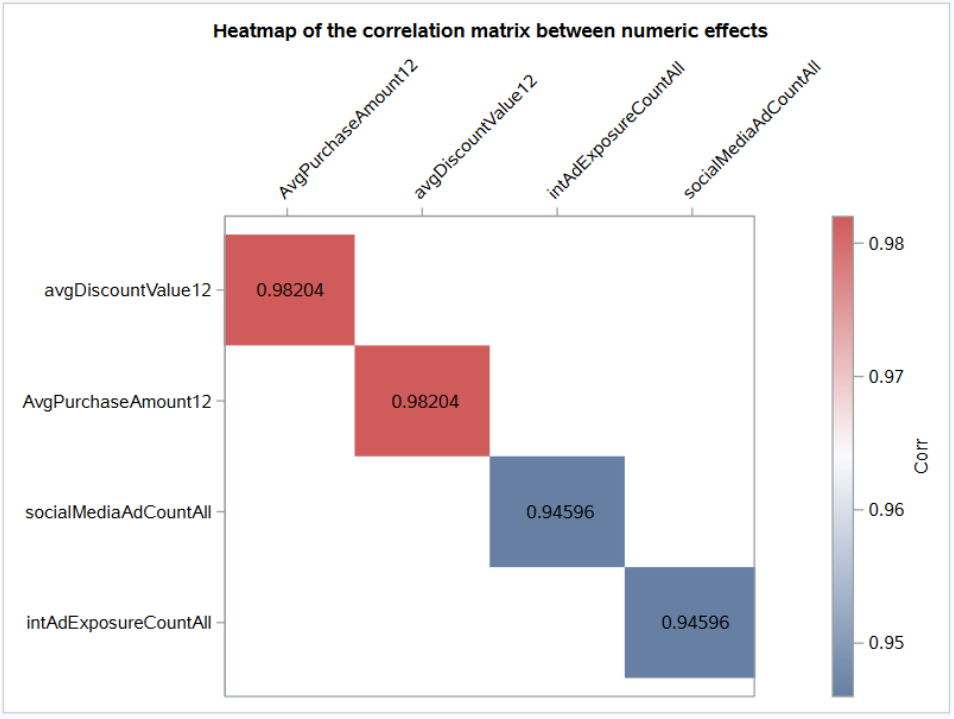
- This code replicates the bar charts created earlier in SAS Visual Analytics. And a WHOLE lot of other things. Get those scrolling fingers ready and scroll down the input to **Distribution of customerAge**. I see the following:




- What in the what? That's a bit different than our findings in the VA report:




- But… fret not! The age variable still needs to be processed in the SAS notebook – meaning we need to (1) impute missing values and (2) normalize the variable for modeling. And we'll get there in *02_Pre-Processing_Data.sasnb*. So wait for it 😊
- Explore the data a bit more. For example, here is a heatmap of the correlation between select variables:




- That's kind of fun! When you're done exploring, move over to **02_Pre_Processing_Data.sasnb**. In this notebook, we're going to transform the data to get it closer to the data we examined at the outset in SAS Viya, which includes imputation and addressing outlier observations. You'll also see that we

partition the data and reduce the number of variables available for modeling via variable reduction. Then, we create that final dataset for modeling.

- Looking for fun output in **02_Pre_Processing_Data.sasnb**? Well, there's not a ton of options. But how about this table?

**Supervised variable Selection/Reduction**

**The VARREDUCE Procedure**

| | |
|---|---|
| Number of Observations Read | 3501 |
| Number of Observations Used | 3501 |

**Selection Summary**

| Iteration | Parameter | Proportion of Variance Explained | SSE | MSE | AIC | AICC | BIC |
|---|---|---|---|---|---|---|---|
| 1 | numOfTotalReturns | 0.045503 | 0.954497 | 0.00027271 | -0.043715 | 1.956289 | -0.044240 |
| 2 | regionMedHomeVal | 0.064943 | 0.935057 | 0.00026724 | -0.063149 | 1.936857 | -0.062486 |
| 3 | log_customersales | 0.079222 | 0.920778 | 0.00026323 | -0.077395 | 1.922614 | -0.075543 |
| 4 | customerSubscrStat Gold | 0.084354 | 0.915646 | 0.00026184 | -0.081841 | 1.918172 | -0.078801 |
| 5 | AvgPurchasePerAd12 | 0.087136 | 0.912864 | 0.00026112 | -0.083742 | 1.916275 | -0.079514 |
| 6 | IM_techSupportEval | 0.090435 | 0.909565 | 0.00026025 | -0.086219 | 1.913803 | -0.080803 |
| 7 | IM_customerAge | 0.092414 | 0.907586 | 0.00025976 | -0.087255 | 1.912773 | -0.080650 |
| 8 | log_AvgPurchaseAmountTotal | 0.094142 | 0.905858 | 0.00025934 | -0.088018 | 1.912016 | -0.080224 |
| 9 | regionPctCustomers | 0.095813 | 0.904187 | 0.00025893 | -0.088722 | 1.911319 | -0.079740 |

- Finally, open **03_Developing_a_Model.sasnb**. Here we're going to run A LOT of models to best determine which customers will churn at SASe. For example, here is the code for a gradient boosting model:

## Create Gradient Boosting Model

Save the analytic store for the model and use it for scoring.

```
title2 'GRADBOOST on Churn Data';
proc gradboost data=work.customer_churn_ml_train_final ntrees=100 maxdepth=5 assignmissing=useinsearch;
    input &char_vars / level=nominal;
    input &num_vars / level=interval;
    target &target / level=nominal;
    savestate rstore=gbstore;  /* creates ASTORE binary for scoring  */
    *code file='gbscore.sas';  /* creates 70,000 line SAS scoring program  */
run;

title2 'ASTORE describe and scoring';
proc astore;
    describe rstore=gbstore;
    score data=work.customer_churn_ml_test_final rstore=gbstore
          out=grad_scored copyvars=(&target);
run;
```

[54] ✓ 1.6s SAS

- And there is a whole lot of output to digest, including a useful variable importance table:

| Variable Importance | | | |
|---|---|---|---|
| Variable | Importance | Std Dev Importance | Relative Importance |
| log_customersales | 10.0373 | 14.2437 | 1.0000 |
| regionMedHomeVal | 3.7701 | 3.1782 | 0.3756 |
| intAdExposureCountAll | 3.0205 | 2.7501 | 0.3009 |
| socialMediaAdCount36 | 2.6756 | 4.5852 | 0.2666 |
| AvgPurchasePerAd12 | 2.6396 | 2.8070 | 0.2630 |
| log_AvgPurchaseAmountTotal | 2.5607 | 2.8658 | 0.2551 |
| regionPctCustomers | 2.4552 | 2.3313 | 0.2446 |
| IM_customerAge | 2.3066 | 2.0132 | 0.2298 |
| wksSinceLastPurch | 1.8216 | 2.0439 | 0.1815 |
| LastPurchaseAmount | 1.6652 | 2.1211 | 0.1659 |
| logi_avgDiscountValue12 | 1.3916 | 1.5490 | 0.1386 |
| numOfTotalReturns | 1.2637 | 1.8423 | 0.1259 |
| IM_techSupportEval | 0.7433 | 1.6033 | 0.0741 |
| customerSubscrStat | 0.6176 | 1.7181 | 0.0615 |
| customerGender | 0.3801 | 0.6091 | 0.0379 |
| demHomeOwner | 0.2731 | 0.6044 | 0.0272 |

- Finally, a big part of predictive modeling and machine learning is comparing models and then selecting a champion model. Scroll down to the bottom to see the **Model Assessment** code. It's a (coding) mouthful:

## Additional Model Assessment

Create macro program Modelstats that calculates accuracy, sensitivity, and specificity for all three models including ensemble.

```
%macro modelstats(mtype);

/* Run PROC FREQ to generate the frequency table */
proc freq data=ensemble_predictions noprint;
  tables &target*I_&target._&mtype / out=freq_out;
run;

/* Extract the counts for TP, TN, FP, FN and save to a new dataset */
data confusion_matrix;
    length TP TN FP FN 8;
    set freq_out;

    /* Initialize counts */
    retain TP TN FP FN 0;

    /* True Positive: actual=1 and predicted=1 */
    if &target = 1 and I_&target._&mtype = "1" then TP = count;

    /* True Negative: actual=0 and predicted=0 */
    if &target = 0 and I_&target._&mtype = "0" then TN = count;

    /* False Positive: actual=0 and predicted=1 */
    if &target = 0 and I_&target._&mtype = "1" then FP = count;

    /* False Negative: actual=1 and predicted=0 */
    if &target = 1 and I_&target._&mtype = "0" then FN = count;

    /* Keep only one row with the final values */
    if _N_ = 4 then output;
    keep TP TN FP FN;
run;

data metrics;
    set confusion_matrix;
    Accuracy = (TP + TN) / (TP + TN + FP + FN);
    Sensitivity = TP / (TP + FN);
    Specificity = TN / (TN + FP);
run;

Title2 "Overall Model Perfomance Statistics for &mtype";
proc print data=metrics;
run;
%mend modelstats;

%modelstats(log)
%modelstats(grad)
%modelstats(ensemble)

/* Clean up */
title;
footnote;
```

[57] ✓ 0.3s

- And the output:

**Overall Model Perfomance Statistics for log**

| Obs | TP | TN | FP | FN | Accuracy | Sensitivity | Specificity |
|---|---|---|---|---|---|---|---|
| 1 | 102 | 987 | 342 | 68 | 0.72648 | 0.6 | 0.74266 |

**Overall Model Perfomance Statistics for grad**

| Obs | TP | TN | FP | FN | Accuracy | Sensitivity | Specificity |
|---|---|---|---|---|---|---|---|
| 1 | 114 | 1135 | 194 | 56 | 0.83322 | 0.67059 | 0.85403 |

**Overall Model Perfomance Statistics for ensemble**

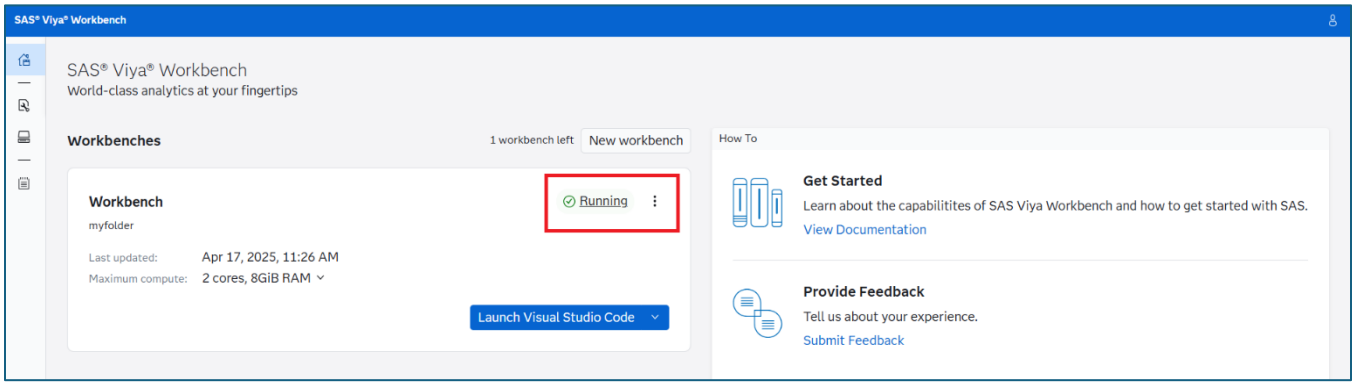
| Obs | TP | TN | FP | FN | Accuracy | Sensitivity | Specificity |
|---|---|---|---|---|---|---|---|
| 1 | 116 | 1080 | 249 | 54 | 0.79787 | 0.68235 | 0.81264 |

- Why would I put you through all that? Just to show you that the gradient boosting model would be selected if Accuracy was the selection statistic of choice? Well… kind of. I also want to show you the full code so that you can see the beauty and power of the code… and how flexible your program can be. But… it comes at the expense of time and expertise… as it takes a long time to master this code. That stated, you can do it – and the learning paths at the end will help! But, for now, let's slide over to the Python code

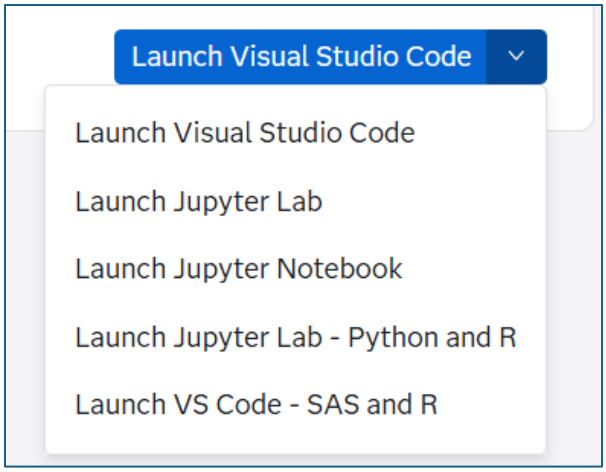
## Modeling in Python

You just saw how to explore data, pre-process data, and then model in SAS using SAS notebooks in Visual Analytics in SAS Viya Workbench. Yes, that was a mouthful. But, word-jumbo aside, let's show you how to do it all again in Python.

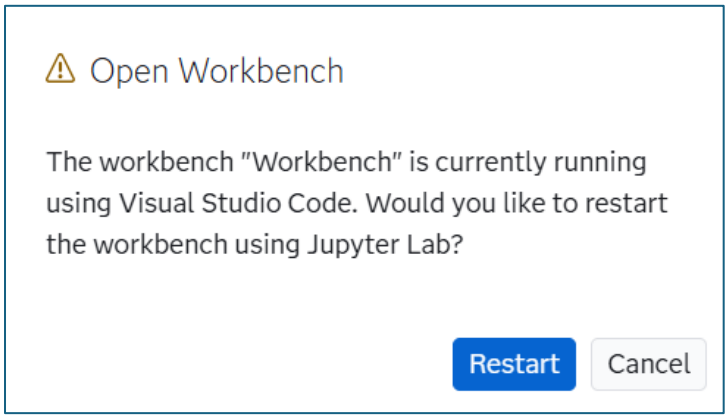
- Since I'm a teacher at heart, I'm going to expose you to another Integrated Development Environment – aka IDE. **Close** the **Visual Studio** browser window and go back to the **SAS Viya Workbench** landing page. And confirm that your session is still **Running**… because running is a good thing. It means the session is still active:



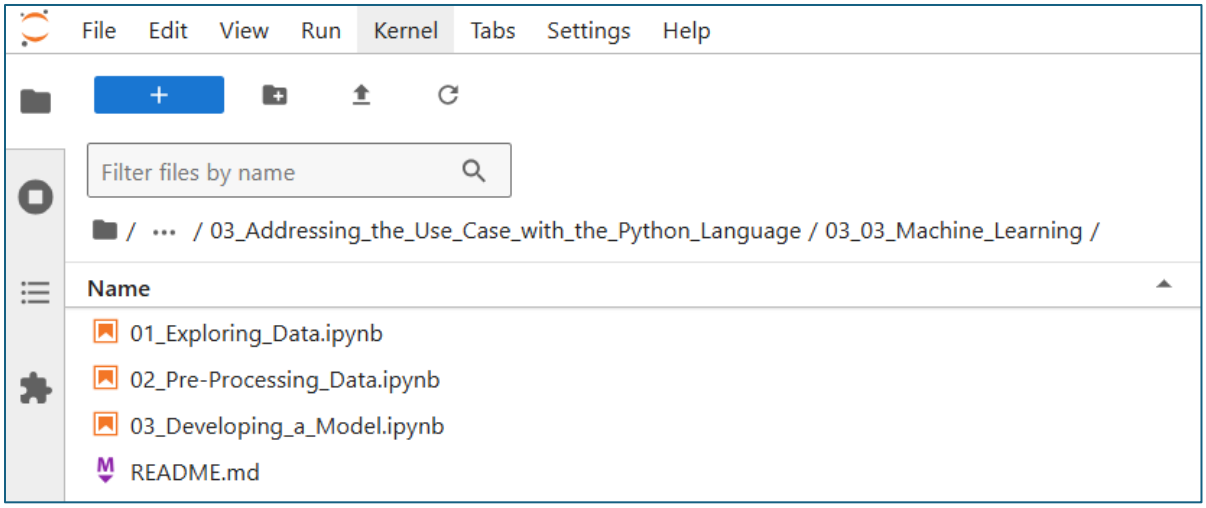

- Now examine the **Editor** options next to the **Launch Visual Studio Code** button:



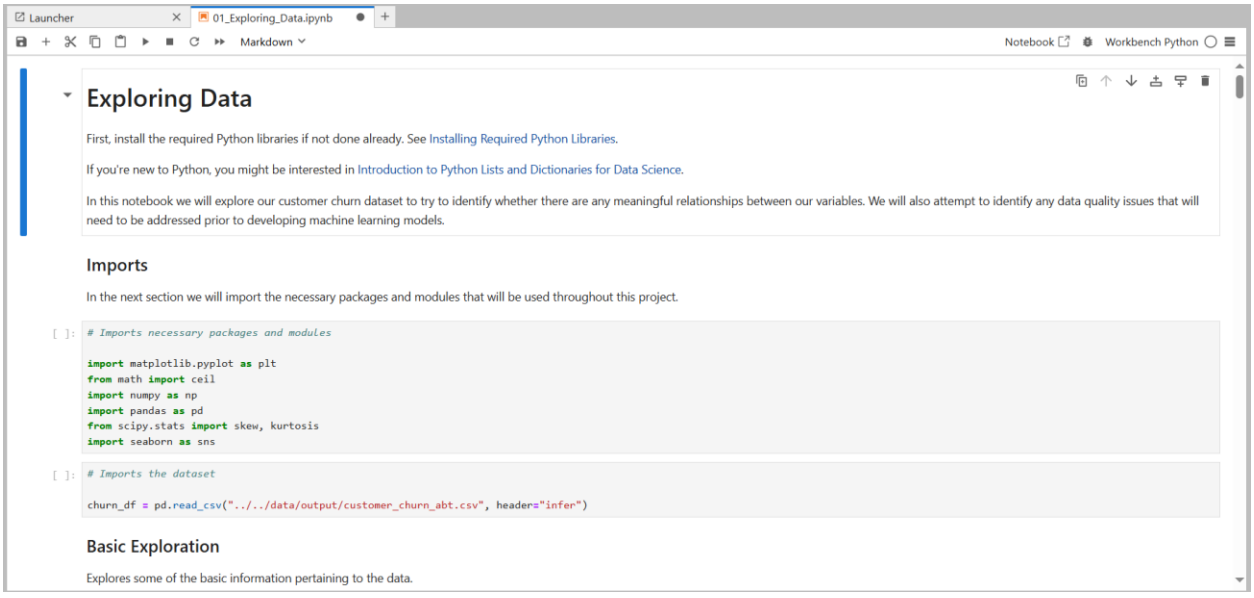

- Lookie what we have there! For those looking for a Jupyter experience closer to the one provided by default in SAS Viya, I recommend using **Jupyter Lab**.
    - And for those of you interested, my good friend ChatGPT has this to say about the difference between Jupyter Notebook and Jupyter Lab:
    - "Jupyter Notebook offers a simple, single-document interface where you work on one notebook at a time, making it ideal for focused tasks. JupyterLab provides a more flexible, IDE-like interface that allows for working on multiple documents and tools simultaneously, with customizable layouts for enhanced productivity."
- So, open **Jupyter Lab** for today, because we've got multiple programs to look at. And you'll likely get this message:



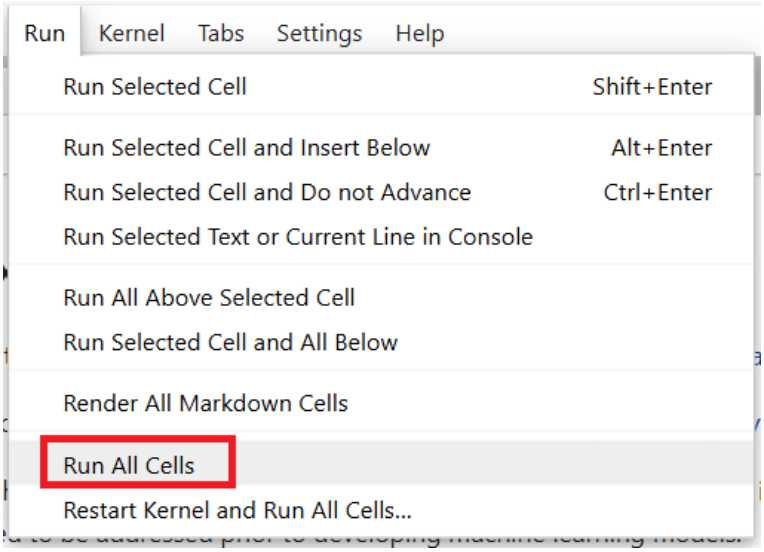

- All good, let's **Restart**! And then Welcome to Jupyter Lab!
- Now we already did the hard work of setting up the data and notebooks in Visual Studio. So, to find the Python programs using the **File Browser**, drill into **sas-education** >> **sas1** >> **3_Addressing_the_Use_Case_with_the_Python_Language >> 03_03_Machine_Learning**. Like so:




- And the corresponding notebooks await. Nice!
- Even though we're in a second IDE, the actions are the same:
    - Open the notebooks in order.
    - Run them… either as individual cells or as entire programs.
    - Digest as much of this Machine Learning section as you can handle in one sitting.
    - And, finally, more high fives – you're done.
- But, again, I won't just leave you at that. Let's explore some of the same items we examined in the **Modeling in SAS** section. So, open **01_Exploring_Data.ipynb**, which reveals:



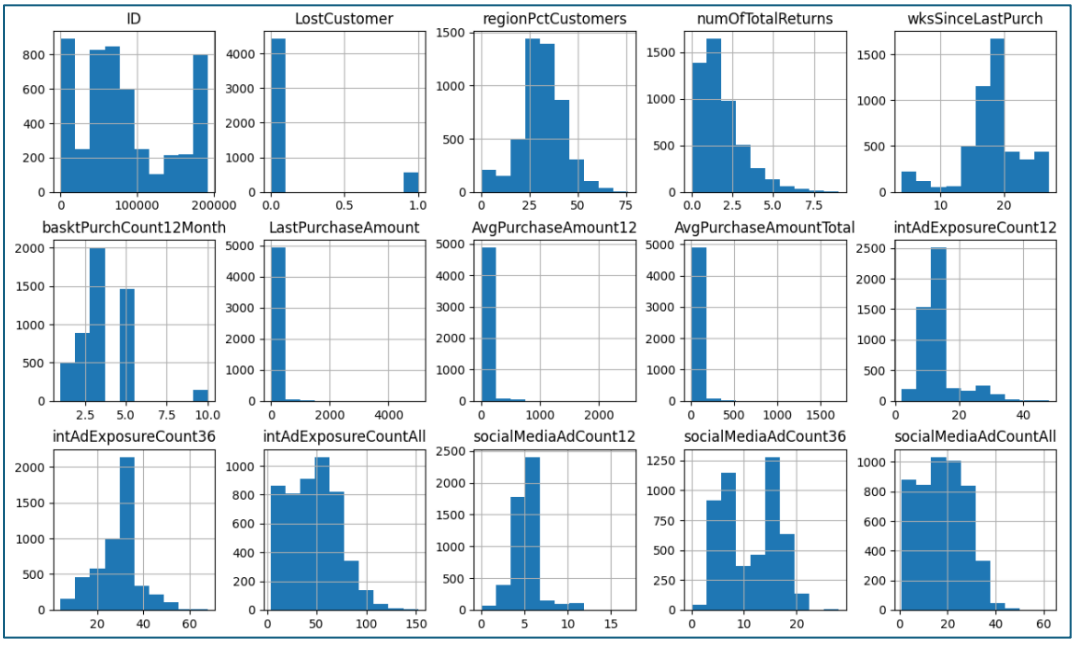
- In Jupyter, you can either select **Run** >> **Run All Cells** or select an individual cell and then submit with a **Shift+Enter**. I'm going to Run All Cells just to simplify my storytelling:



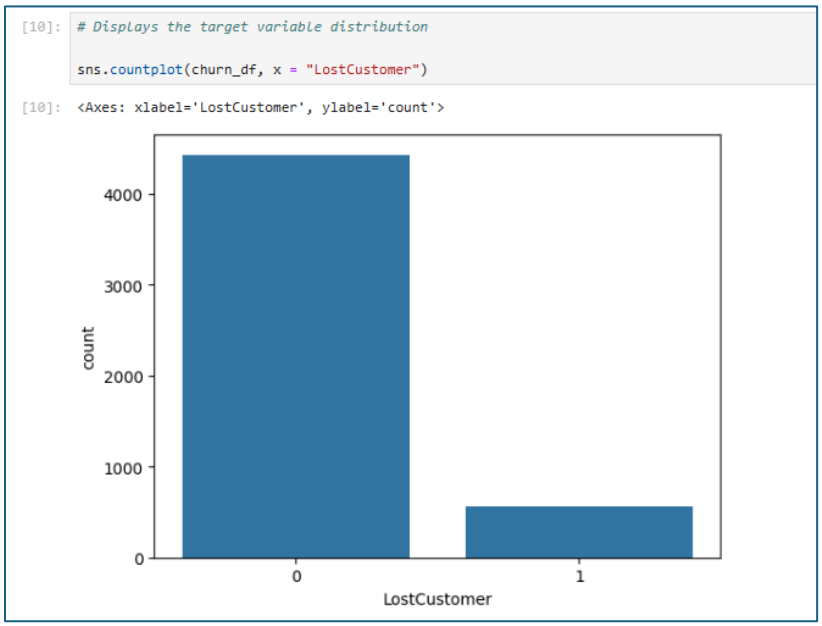

- Scroll until you find the code for the Histograms:

```
# Displays a histogram of all of the float and int columns

churn_df.hist(figsize = (15,15))
```

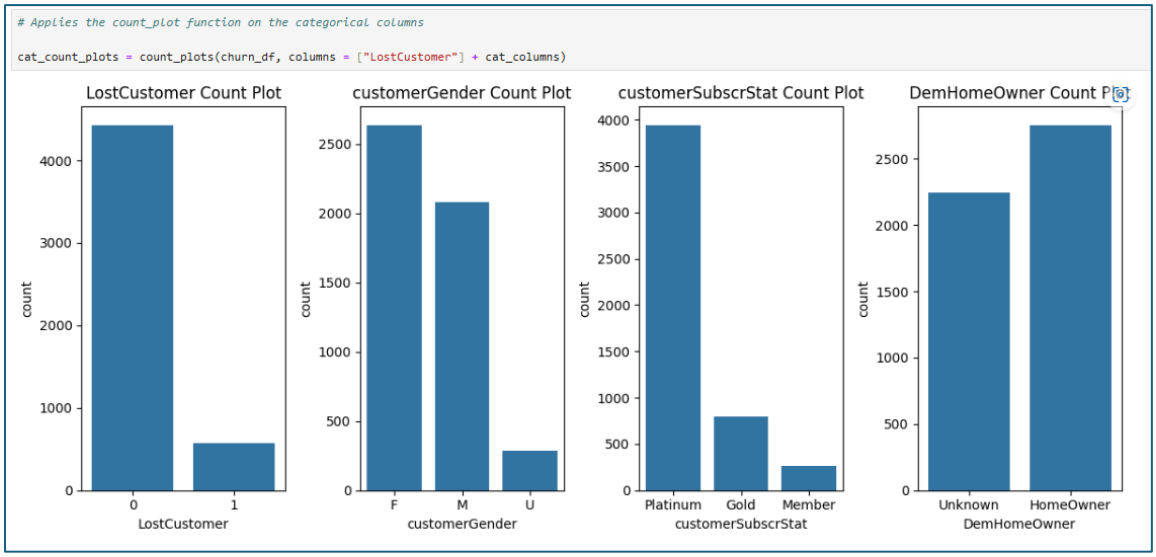
- Pretty succinct code, eh? And check out the output table:



- That's a great way to digest the information! Points for Python! Moreover, here is the code and output for the *LostCustomer* variable analysis:

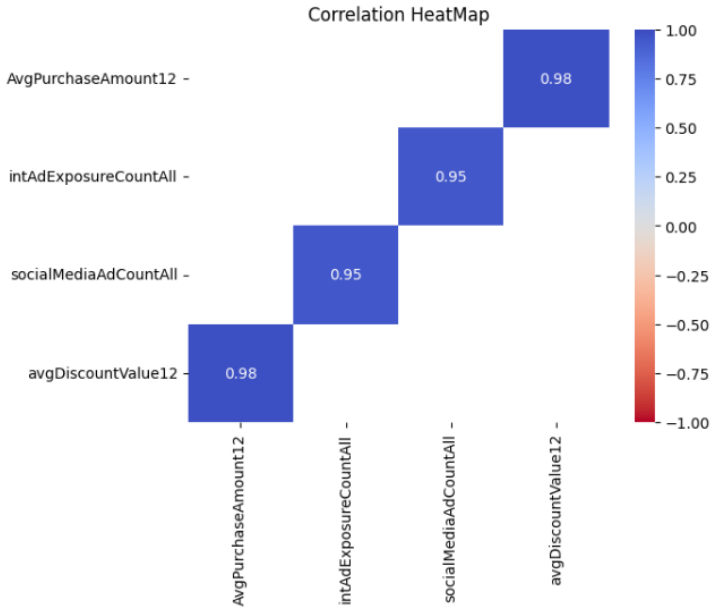



- Python is a lower-level language than SAS – meaning that you need to do a lot more coding to get output. But, that does have advantages when you just want something simple. For example, here is another easily digestible table:




- My point: every language does some things very well – so learn as many as you can so that you can use the right tool for the job! And after absorbing those words of wisdom, check out the HeatMap, with some useful correlations:

```
[26]: # Generates a heatmap using the correlation values

      cmap = plt.get_cmap("coolwarm").reversed()
      corr_plot = sns.heatmap(high_corr, vmin = -1, vmax = 1, annot=True, cmap=cmap)
      corr_plot.set_title("Correlation HeatMap")

[26]: Text(0.5, 1.0, 'Correlation HeatMap')
```

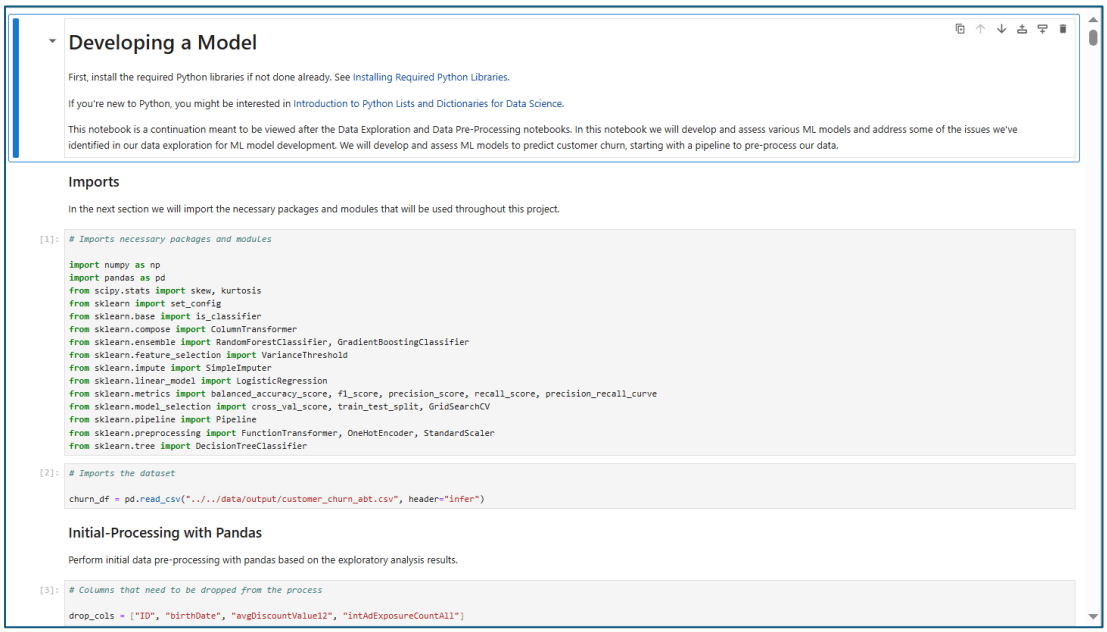



- Alright, that was fun. Open **02_Pre-Processing_Data.ipynb** to keep the adventure rolling. Submit and explore that code, which will partition the data, impute missing values, address outliers, and reduce the number of input variables. Note that the last part can take a bit of time because it uses a compute-intensive backwards elimination tool!
- And here is a summary of the variables selected:

```
[29]: # Selects the features the tree used in both partitions

      selected_features = [col for col in inputs if (col in variance_features) or (col in rfe_features) or (col in tree_features)]
      selected_features

[29]: ['regionPctCustomers',
       'numOfTotalReturns',
       'wksSinceLastPurch',
       'basktPurchCount12Month',
       'intAdExposureCount12',
       'intAdExposureCount36',
       'socialMediaAdCount12',
       'socialMediaAdCount36',
       'socialMediaAdCountAll',
       'totalNumProdPurchased',
       'custInitiatedContacts',
       'wksSinceFirstPurch',
       'EstimatedIncome',
       'regionMedHomeVal',
       'techSupportEval',
       'customerAge',
       'LOGLastPurchaseAmount',
       'LOGAvgPurchaseAmount12',
       'LOGAvgPurchaseAmountTotal',
       'LOGcustomersales',
       'LOGAvgPurchasePerAd12',
       'customerGender_F',
       'customerGender_M',
       'customerGender_U',
       'customerSubscrStat_Gold',
       'customerSubscrStat_Platinum',
       'DemHomeOwner_Unknown']
```

- Mheh. I've seen better summary tables. But that works.

- Alright, what do we have: (1) data read in. Check. (2) Data cleaned. Check. Now let's open **03_Developing_a_Model.ipynb** and explore some models! That notebook:



## Developing a Model

First, install the required Python libraries if not done already. See Installing Required Python Libraries.

If you're new to Python, you might be interested in Introduction to Python Lists and Dictionaries for Data Science.

This notebook is a continuation meant to be viewed after the Data Exploration and Data Pre-Processing notebooks. In this notebook we will develop and assess various ML models and address some of the issues we've identified in our data exploration for ML model development. We will develop and assess ML models to predict customer churn, starting with a pipeline to pre-process our data.

### Imports

In the next section we will import the necessary packages and modules that will be used throughout this project.

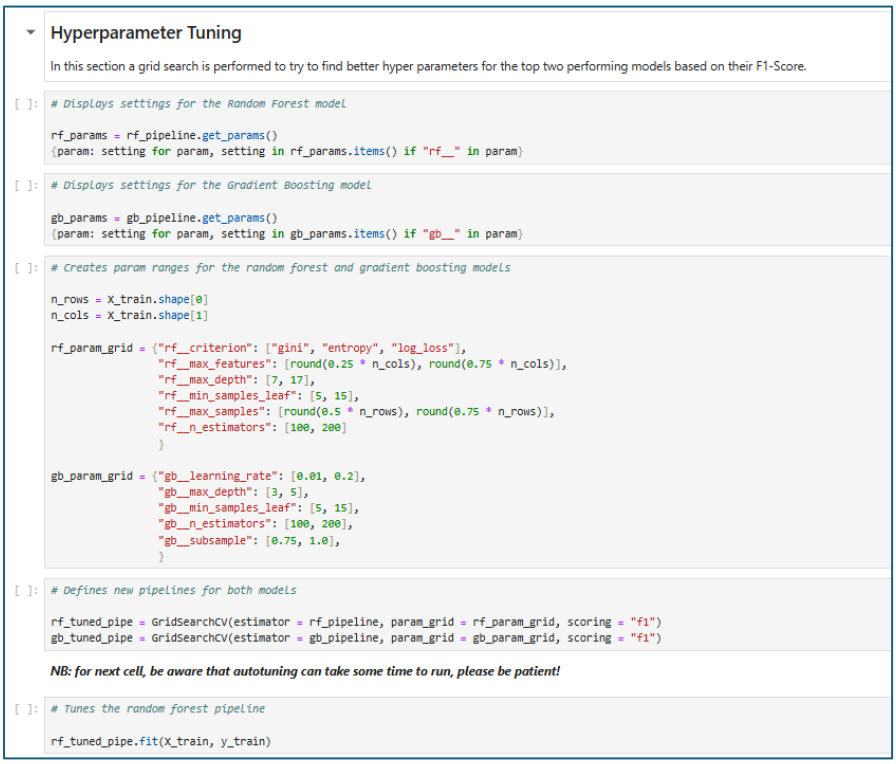
```
[1]: # Imports necessary packages and modules

import numpy as np
import pandas as pd
from scipy.stats import skew, kurtosis
from sklearn import set_config
from sklearn.base import is_classifier
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.feature_selection import VarianceThreshold
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import balanced_accuracy_score, f1_score, precision_score, recall_score, precision_recall_curve
from sklearn.model_selection import cross_val_score, train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import FunctionTransformer, OneHotEncoder, StandardScaler
from sklearn.tree import DecisionTreeClassifier

[2]: # Imports the dataset

churn_df = pd.read_csv("../../data/output/customer_churn_abt.csv", header="infer")
```

### Initial-Processing with Pandas

Perform initial data pre-processing with pandas based on the exploratory analysis results.

```
[3]: # Columns that need to be dropped from the process

drop_cols = ["ID", "birthDate", "avgDiscountValue12", "intAdExposureCountAll"]
```

- As noted in the preamble, we're going to use the sklearn packages to run a series of predictive models. Moreover, one special feature in this notebook is the **Hyperparameter Tuning** code, found here:



## Hyperparameter Tuning

In this section a grid search is performed to try to find better hyper parameters for the top two performing models based on their F1-Score.

```
[ ]: # Displays settings for the Random Forest model

rf_params = rf_pipeline.get_params()
{param: setting for param, setting in rf_params.items() if "rf__" in param}

[ ]: # Displays settings for the Gradient Boosting model

gb_params = gb_pipeline.get_params()
{param: setting for param, setting in gb_params.items() if "gb__" in param}

[ ]: # Creates param ranges for the random forest and gradient boosting models

n_rows = X_train.shape[0]
n_cols = X_train.shape[1]

rf_param_grid = {"rf__criterion": ["gini", "entropy", "log_loss"],
                 "rf__max_features": [round(0.25 * n_cols), round(0.75 * n_cols)],
                 "rf__max_depth": [7, 17],
                 "rf__min_samples_leaf": [5, 15],
                 "rf__max_samples": [round(0.5 * n_rows), round(0.75 * n_rows)],
                 "rf__n_estimators": [100, 200]
                 }

gb_param_grid = {"gb__learning_rate": [0.01, 0.2],
                 "gb__max_depth": [3, 5],
                 "gb__min_samples_leaf": [5, 15],
                 "gb__n_estimators": [100, 200],
                 "gb__subsample": [0.75, 1.0],
                 }

[ ]: # Defines new pipelines for both models

rf_tuned_pipe = GridSearchCV(estimator = rf_pipeline, param_grid = rf_param_grid, scoring = "f1")
gb_tuned_pipe = GridSearchCV(estimator = gb_pipeline, param_grid = gb_param_grid, scoring = "f1")
```

***NB: for next cell, be aware that autotuning can take some time to run, please be patient!***

```
[ ]: # Tunes the random forest pipeline

rf_tuned_pipe.fit(X_train, y_train)
```

- What is hyperparameter tuning… aka autotuning? Well, it is machine learning in the truest sense: it means the machine will run thousands of different models, based upon combinations of settings (or hyperparameters) in the models. In our case, the machine will iterate on the random forest and gradient boosting models. As a more concrete example for tree-based models, hyperparameter tuning will adjust the number of branches available, the depth of the trees, the decay rates, etc. all automatically. Whoa. And if you're not sure what that all means – don't worry – there is a SAS course for that!

- **03_Developing_a_Model.ipynb** will take a long time to run because of the autotuning, a bonus feature in that notebook not currently available in the SAS code (but wait for it!). Once finished, you'll see that the assessment tables are a bit buried. But the selected model has the following performance measures:

```
[50]: # Runs the assessment function on the un-pickled object

      viya_assess, models_info = model_assessor([loaded_gb_pipe], X_test, y_test)
      viya_assess
```

[50]:

| | Model | F1-Score | Balanced Accuracy | Precision | Recall |
|---|---|---|---|---|---|
| 0 | gb | 0.735849 | 0.830819 | 0.795918 | 0.684211 |

- Digest all that output for a bit. Then exhale. Our coding adventure is over. Now let's return to SAS Model Studio in SAS Viya for Learners and see how we'd do a similar analysis in a low/no-code environment.

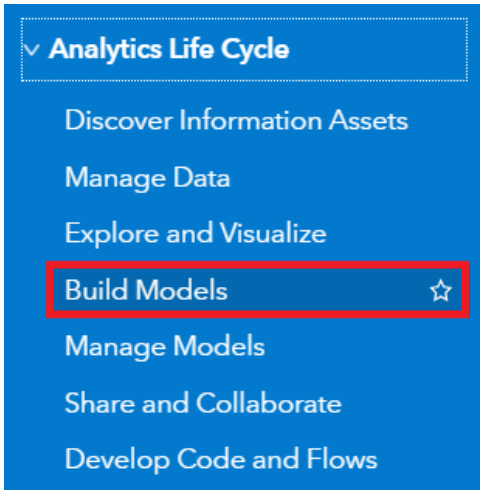
## SAS Model Studio, Revisited + Expanded

Let's face it. We've done a LOT already. And, yes, I'm going to plunge full steam ahead and show you how SAS Model Studio does pretty much everything we did… with just a few clicks. SAS Model Studio is a prime example of how SAS is embedding AI agents throughout the analytics lifecycle – which helps speed up the analysis time. In the past data scientist could expect to spend roughly 80% of their time preparing data and just 20% doing the good stuff – i.e., modeling and deploying models. However, with no + low-code tools, we can spend a LOT more time running models, working on the storytelling, and then asking whether we should be running those models – aka ethical data analysis. It's a brave new world indeed with AI!

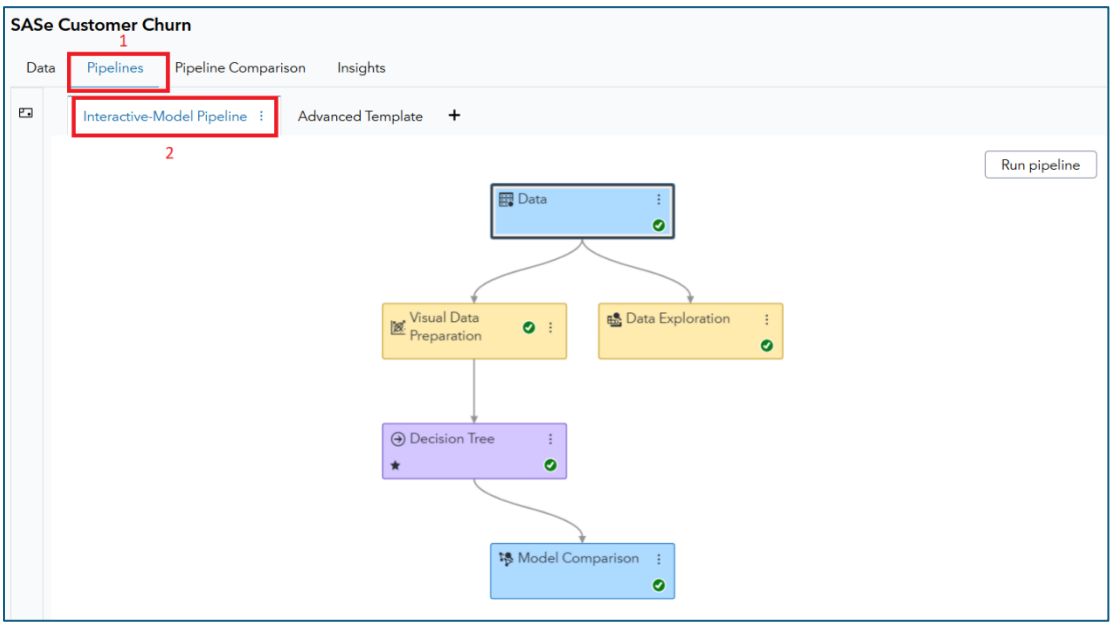
Let's get back into SAS Model Studio and examine our pipelines.

- Log back into SAS Viya for Learners, if you timed-out. Timeouts happen. And if you're no longer in SAS Model Studio, you can get there via the **Applications menu**:

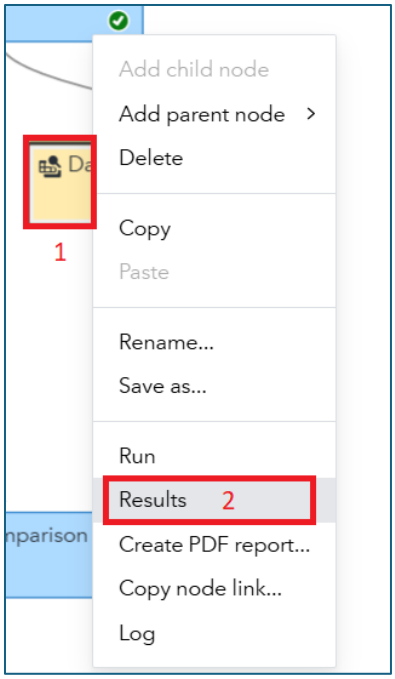
- Click that button and then select **Build Models**:



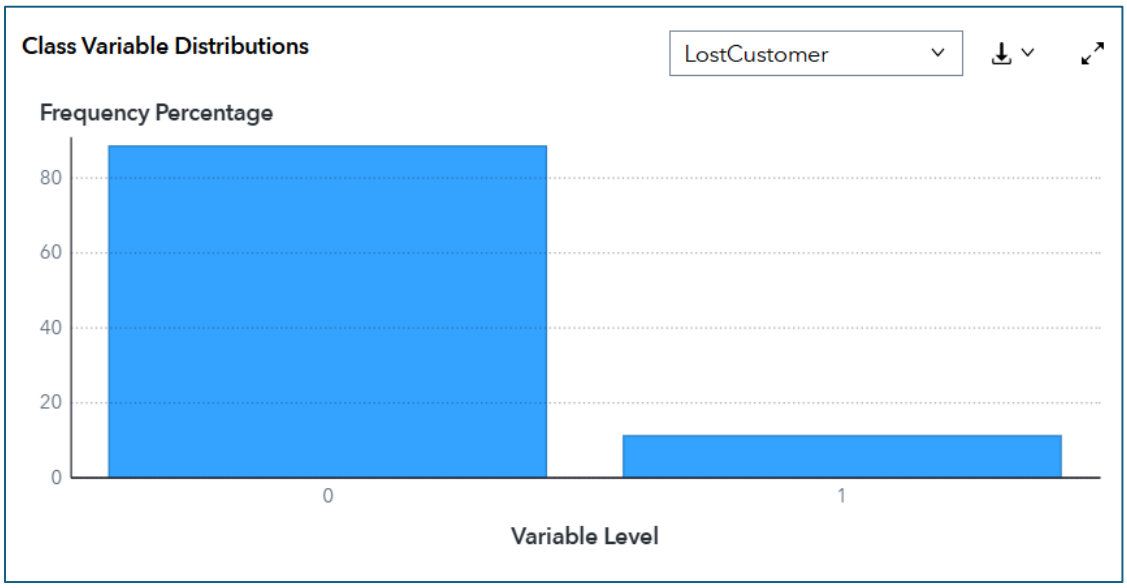

- And if you really need to, open your **SASe Customer Churn** project. I'm hoping that it's still active. Additionally, return to the **Interactive-Model Pipeline**, here:



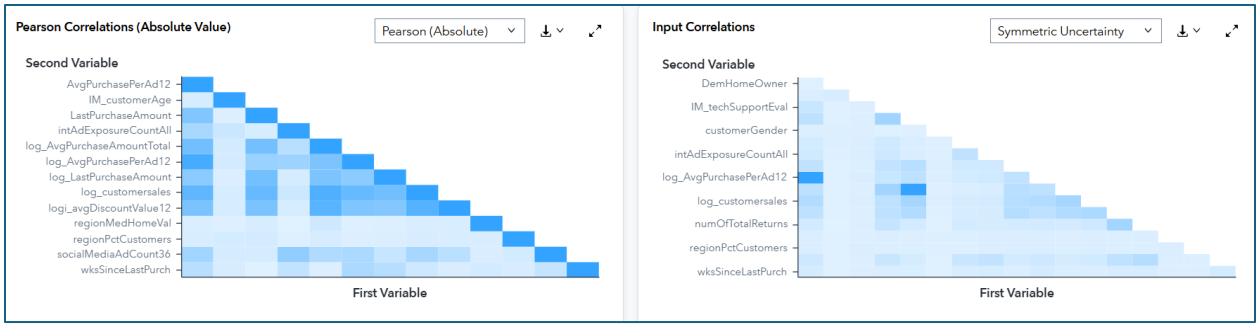

- Green checkmarks mean that modeling life is good! We'll start our SAS Model Studio digest in the **Data Exploration** node. Right-click on the node and then select **Results**, like so:



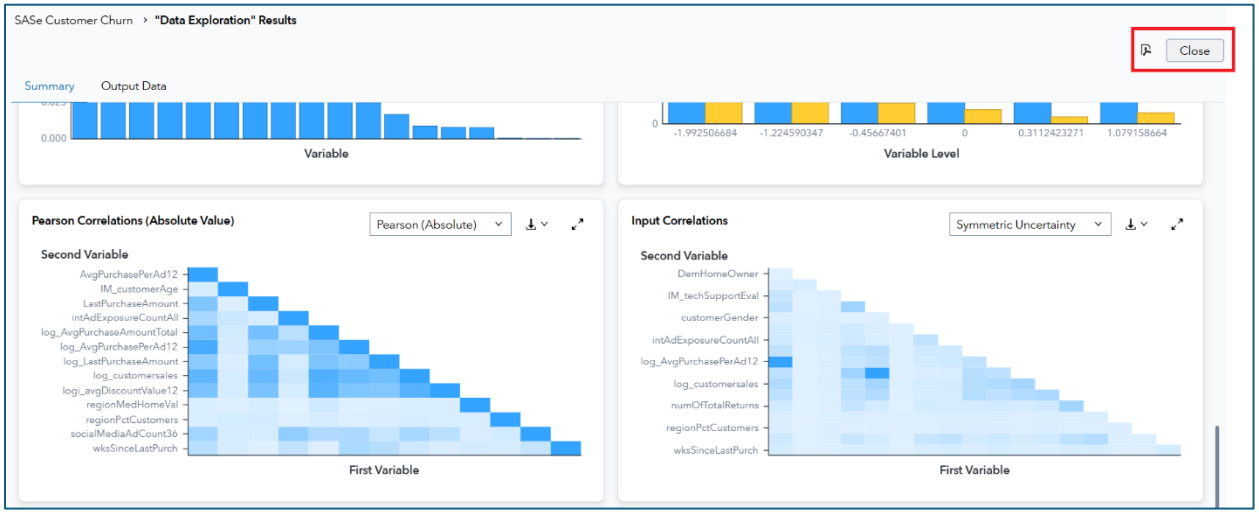

- What do we have here? Well, well – a LOT of useful information on our data! Remember we had to either code or create this in a dashboard in our previous work. But with one node, I get helpful information like **Class Variable Distributions**:



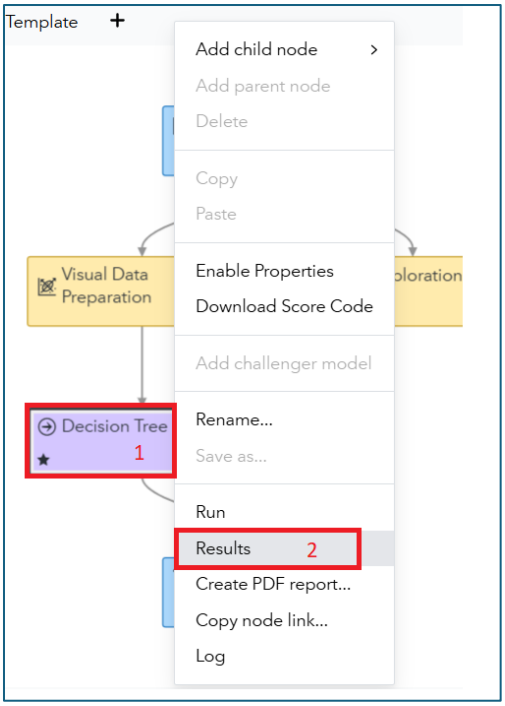

- And, wait there's more! Check out the correlations with fun phrases like "Symmetric Uncertainty":



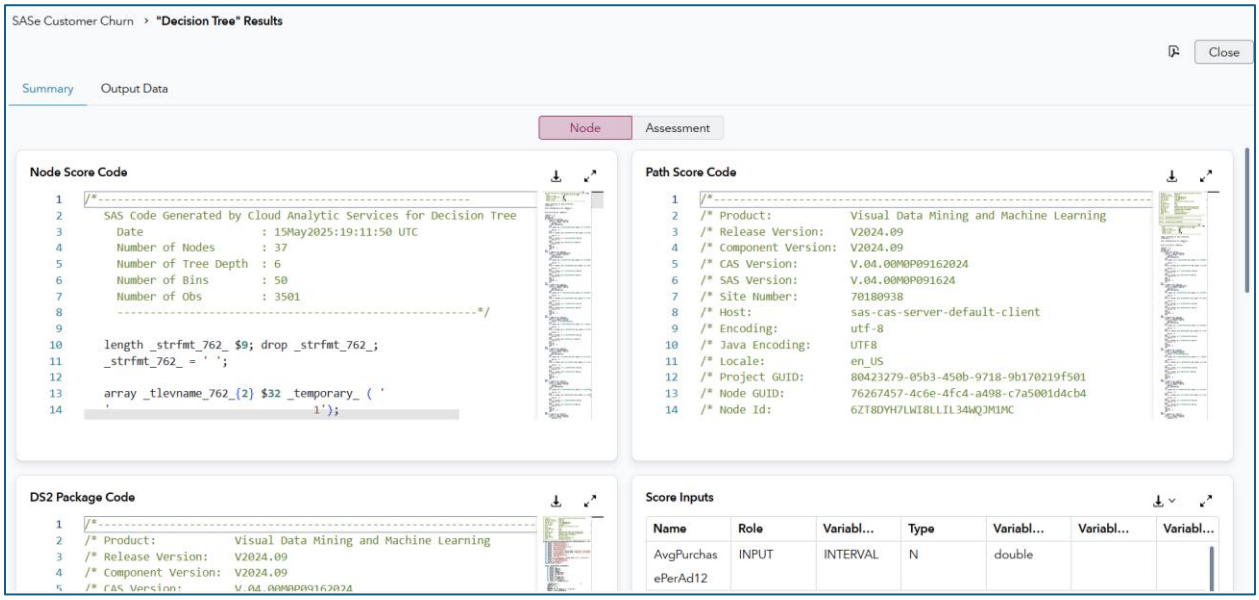

- Anyway, there is a lot to digest here. Digest away and click **Close** when you're done. Bonus points for downloading a PDF and sharing with your family. Those two buttons are found here:



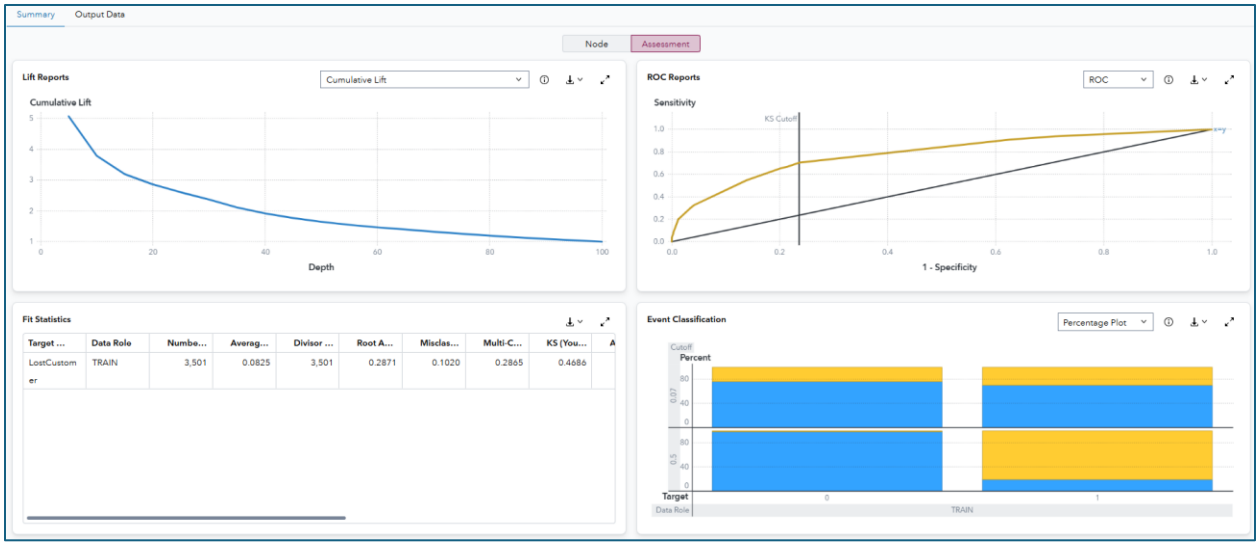

- Now let's examine the decision tree that we imported from SAS Visual Analytics. Right-click **Decision Tree** in this pipeline and then select **Results**. With these two steps:



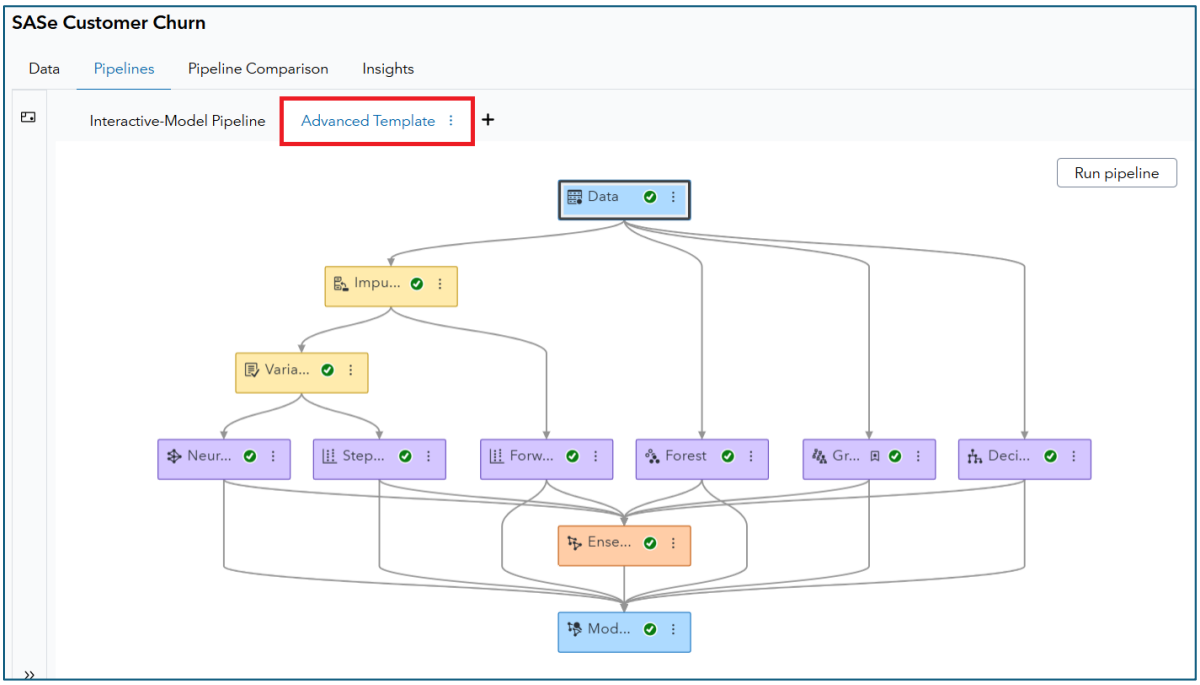

- Another interactive dashboard thing awaits! And you land on the **Node** tab:



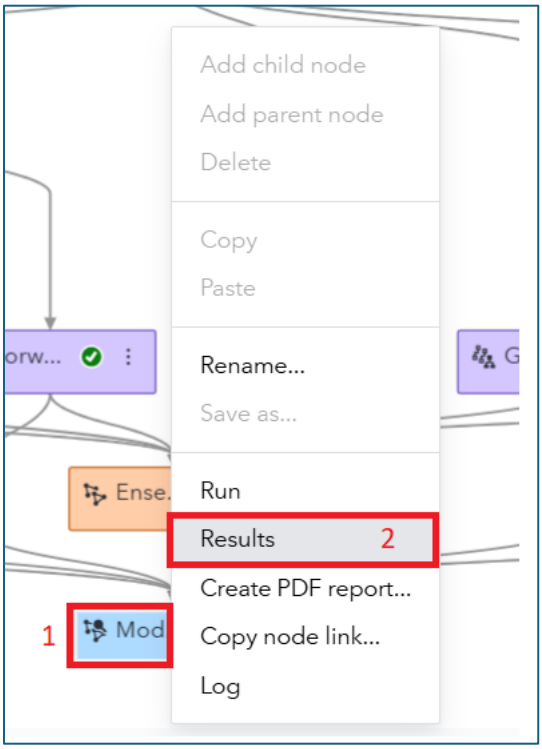

- Click **Assessment** so we can see how the model performed:




- Here you'll find a lot of assessment tools to see how well your models fit the various data samples. Explore for a bit and then pause to think: *if you were to program all this, how long would it take you?* For me… a little bit 😊
- Click **Close** when you've processed the **Results** for the **Decision Tree**. And the same bonus points apply for creating a PDF and sharing with family.
- Alright – that's one pipeline… but what about the **Advanced Template**. Wasn't that a big deal? Yes, yes – it was. Activate that pipeline and ensure that you've got a bunch of green checks:



- Remember that there are 7 models in this pipeline, including the Ensemble which is based on the estimates from the previous 6 contenders. And while you could explore the models individually, you're bound to think: *which model is the best*? And there's a node for that! Right-click on **Model Comparison** and then select **Results**, as follows:



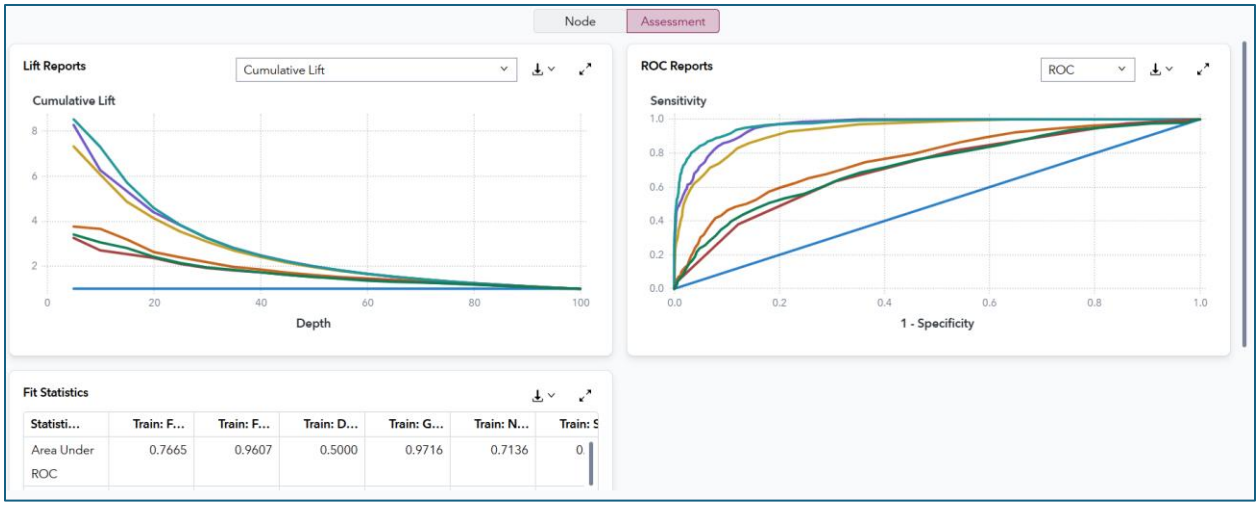

- Look how nicely those results are displayed! And we can quickly see that the Gradient Boosting model is chosen, with the default selection statistic of the KS(Youden) statistic:

Node | Assessment

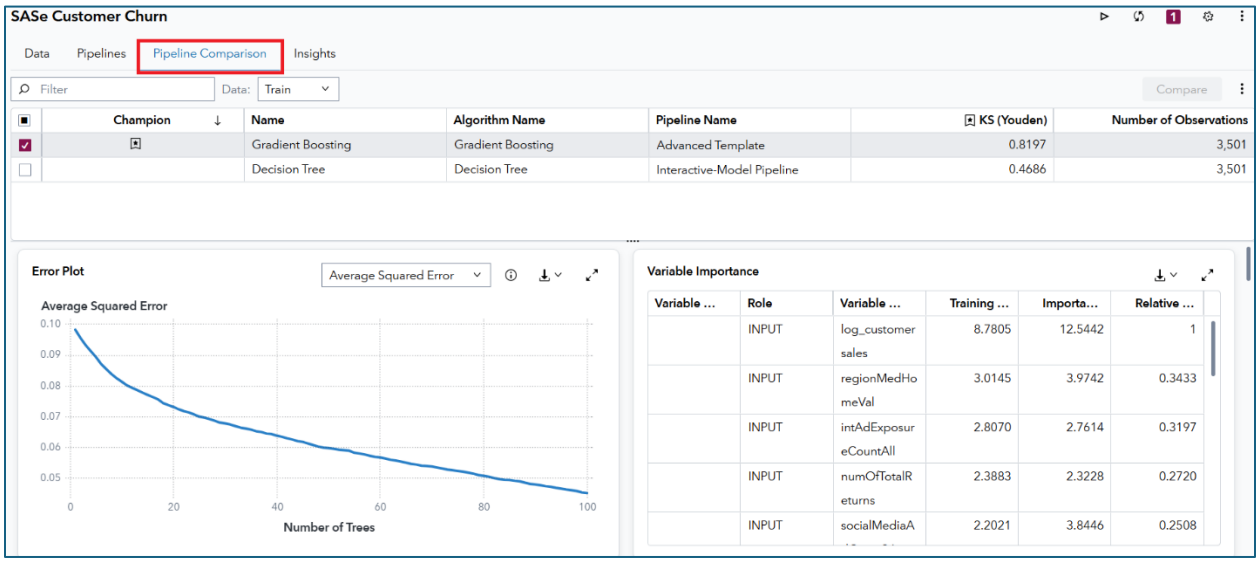
Model Comparison

| Cha... | Name | Algorithm Name | ↓ KS (Youden) | Accuracy | Averag... | Area U... | Cumula... | Cumula... | Cutoff | Data Role |
|---|---|---|---|---|---|---|---|---|---|---|
| ★ | Gradient Boosting | Gradient Boosting | 0.8197 | 0.9380 | 0.0452 | 0.9716 | 7.2932 | 72.9323 | 0.5000 | TRAIN |
| | Forest | Forest | 0.7946 | 0.9117 | 0.0594 | 0.9607 | 6.2657 | 62.6566 | 0.5000 | TRAIN |
| | Ensemble | Ensemble | 0.7172 | 0.8866 | 0.0761 | 0.9348 | 6.0652 | 60.6516 | 0.5000 | TRAIN |
| | Forward Logistic Regression | Logistic Regression | 0.3966 | 0.8855 | 0.0893 | 0.7665 | 3.6591 | 36.5915 | 0.5000 | TRAIN |
| | Stepwise Logistic Regression | Logistic Regression | 0.3322 | 0.8849 | 0.0929 | 0.7229 | 3.0576 | 30.5764 | 0.5000 | TRAIN |
| | Neural Network | Neural Network | 0.3275 | 0.8860 | 0.0991 | 0.7136 | 2.7068 | 27.0677 | 0.5000 | TRAIN |
| | Decision Tree | Decision Tree | 0 | 0.8860 | 0.1010 | 0.5000 | 1.0054 | 10.0543 | 0.5000 | TRAIN |

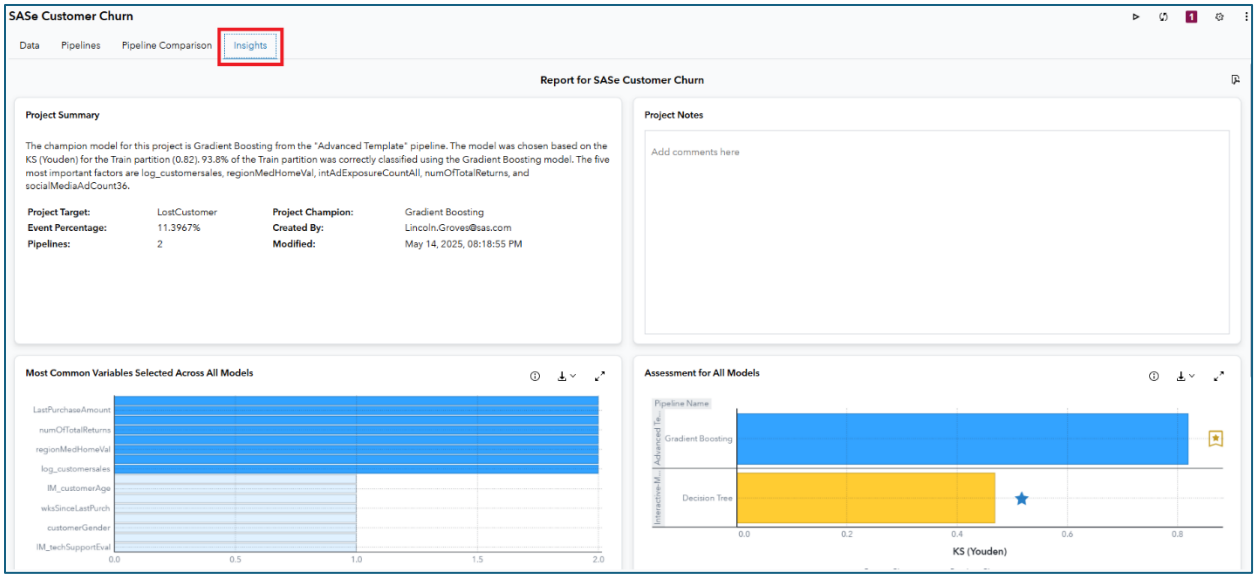
- Additionally, the **Assessment** tab allows you to compare the model based upon other statistics, such as Lift Reports and ROC Reports:




- And I'll ask the question again. How quickly could you code those model comparisons. Me? Well, that would take a LONG time! Explore the **Results** a bit and then click **Close**.
- Two more things to show you before we wrap up. We have two pipelines… but we haven't crowned an overall project champion. And that information is waiting for you under the **Pipeline Comparison** tab. Check it out:




- You see what I see? Yes, your eyes are not deceiving you. The Gradient Boosting model crushes that Decision Tree from VA. Next step would be to put that model into production – which means identifying the current set of customers who could be on the cusp of canceling. Then we can pass that list on to marketing and let them custom-tailor promotions to keep them around. But that is a story for another day.
- The last thing I want you to explore is the **Insight** tab. It's found here:




- **Insights** are a series of AI generated summaries which help you explain your data to a broader audience. Because modeling is often just half the battle. You then need to explain your results to other people. And that is another task for another day. But just explore for a bit and I'll see you in the wrap-up section.

## Wrap-Up and What's Next.

Thank you for joining me on this Day 1 SASe adventure. Hopefully you got some good insights into the life of a newly hired Data Scientist in the Customer Retention Department at Styled and Sophisticated Enterprises. *LostCustomer* identification was our endgame and there were lots of twists and turns along that learning path – all while gaining exposure to the SAS Viya ecosystem!

I hope you're thinking – *that was a lot… but I really want to learn more!* And, if yes, then I'll return us back to the 4 recommendations that I have for you based upon questions you may be asking:

- *Still confused and want to learn more about the SAS Viya ecosystem?*
  - Take our SAS® Viya Overview course
- *Like dashboarding and SAS Visual Analytics?*
  - Explore SAS Visual Analytics 1 for SAS® Viya: Basics
- *Like predictive modeling and machine learning in a low/no-code environment?*
  - Machine Learning Using SAS® Viya® is a great option!
- *Finally, love coding in a multi-language environment and want to explore SAS Viya Workbench more?*

- Modern Data Science with SAS® Viya® Workbench and Python is perfect for you!

For students, all of these courses are in the SAS Skill Builder for Students. And professors can get access via the SAS Educator Portal. And all academics get these courses for free.

Good luck – and happy learning!