Machine Learning in Healthcare

Predicting Stent Failure

SAS Model Studio — Step-by-Step Instructions

Data Analytics & Data Mining

Gannon University — Dahlkemper School of Business

Spring 2026

Developed by: Dr. Benyawarath “Yaa” Nithithanatchinnapat

Adapted from the SAS Visual Analytics Healthcare Case Study
Original case study by Haidar Altaie, University of Kent

Overview

This guide walks you through the Healthcare Stent Failure case study using SAS Model Studio instead of SAS Visual Analytics. Model Studio gives you a pipeline-based workflow — a structured sequence of nodes for data preparation, modeling, and assessment — that mirrors how analytics teams build models in the real world.

Why Model Studio?

In SAS Visual Analytics, you built models one at a time by dragging objects onto a report canvas. **Model Studio** takes a different approach: you build a ***pipeline*** — a connected workflow that runs data preparation, multiple models, and model comparison all in one shot. This is how production analytics works in most enterprises.

Concept	SAS Visual Analytics	SAS Model Studio
Building models	Drag model objects onto report pages one at a time	Add model nodes to a pipeline; run all at once
Data preparation	Create calculated items manually in the Data pane	Use built-in Imputation and Transformation nodes
Partitioning	Create a partition variable in the Data pane	Set train/validate split in Project Settings
Model comparison	Add a Model Comparison object to a report page	Automatic — Model Comparison node is added when you add any supervised model
Variable roles	Assign in the Roles pane for each model object	Set once on the Data tab; applies to all pipeline models

What You Will Build

By the end of this activity, you will have:

- Created a Model Studio project with the Stent Failure dataset
- Configured variable roles (target, predictors, rejected variables)
- Built a pipeline with Imputation, Decision Tree, Logistic Regression, Forest, and Model Comparison nodes
- Evaluated models using misclassification rate, AUC, KS statistic, and false negatives
- Selected a champion model and interpreted what it means for patient safety

Estimated Completion Time

1.5 to 2 hours (individual) or 2 to 3 hours (team-based with discussion)

Section 1: Create the Project & Configure Data

In Model Studio, everything lives inside a project. A project connects your data source to one or more pipelines. Think of it as the container for your entire analysis.

Part 1.1: Launch Model Studio & Create a New Project

Step 1: Log in to SAS Viya for Learners and navigate to the SAS Landing Page.

Step 2: From the Applications menu (three horizontal lines in the upper left), find and click Model Studio under the Analytics Life Cycle section.

> **Tip:** *If you see "Develop Models" or "Build Models" — that also leads to Model Studio. The exact label varies by SAS Viya version.*

Step 3: Click New Project (or the + icon) to create a new project.

Step 4: Configure the new project with these settings:

- **a.** Project Name: "Healthcare Stent Failure — [Your Name]"
- **b.** Type: Data Mining and Machine Learning
- **c.** Template: Blank template (we will build our own pipeline from scratch)

Step 5: For the Data Source, locate the Stent Failure dataset.

- **a.** Navigate to: Cas-v4e***-default > ACADEMIC > Stent_Failure
- **b.** Select the dataset and click OK / Add.

> ⚠ **Note:** *If the dataset does not appear, it may need to be loaded into CAS memory first. Ask your instructor or check if you need to run a library assignment step.*

Step 6: Click Save to create the project. You should now see the project workspace with a Data tab and a Pipelines tab.

Part 1.2: Explore the Data Tab

Before building any pipeline, spend a few minutes reviewing the data. The Data tab in Model Studio shows you variable metadata — types, roles, distributions, and missing values — without building a single model.

Step 1: Click the Data tab at the top of your project.

Step 2: You should see a list of all variables in the dataset. Review the following columns:

- Name — the variable name
- Role — how the variable will be used (Input, Target, Rejected, ID)
- Level — whether the variable is Nominal (categorical) or Interval (numeric)
- Missing Count / Missing % — how many observations are missing

Step 3: Review the Variables table and verify the variable types match the descriptions below:

Variable	Description	Type
Stent_Failure	Stent Failure (TARGET)	Character
Age	Patient Age	Numeric
Cell_Design	Stent Cell Design	Character
Cell_type	Stent Cell Type	Character

Coronary_Stream	Upstream/Downstream disease in coronary	Character
Device_age	Device age (days)	Numeric
Diabetic	Diabetic status	Character
Ethnic_group	Patient Ethnicity	Character
Gender	Patient Gender	Character
Geographic_Miss	Geographic Miss	Character
Hospital	Hospital	Character
Hours_Active	Hours active per week	Numeric
Multiple_Stent	Multiple Stent	Character
Patient_id	Patient ID	Numeric
Plaque_Prolapse	Plaque Prolapse through cells of the stent	Numeric
Smoker_flag	Smoker flag	Numeric
Stent_Failure_flag	Stent Failure flag (numeric version)	Numeric
Stent_Length	Stent Length (mm)	Numeric
Stent_Material	Stent Material	Character
Stent_Thickness	Stent Thickness (mm)	Numeric
Stent_Width	Stent Width (mm)	Numeric

Part 1.3: Set Variable Roles

Model Studio needs to know which variable is the target (what we are predicting), which are inputs (predictors), and which should be excluded. You set this once on the Data tab, and it applies to every model in your pipeline.

Step 1: Set the TARGET variable:

a. Find Stent_Failure in the variable list.

b. Click on its Role and change it to Target.

> ⚠ **Note:** *Use the character variable Stent_Failure (not Stent_Failure_flag) as the target. Model Studio will treat this as a classification problem.*

Step 2: REJECT the following variables (set Role = Rejected):

- Patient_id — This is an ID number, not a meaningful predictor.
- Stent_Failure_flag — This is a numeric duplicate of the target. Including it would cause data leakage.

Step 3: All remaining variables should have Role = Input. Verify that the following are set as Inputs:

- Diabetic, Ethnic_group, Gender, Geographic_Miss, Hospital, Multiple_Stent
- Cell_Design, Cell_type, Stent_Material, Coronary_Stream
- Age, Device_age, Hours_Active, Plaque_Prolapse, Smoker_flag
- Stent_Thickness, Stent_Width, Stent_Length

Step 4: Check the Level column to make sure SAS correctly classified each variable:

- Character variables should show as Nominal
- Numeric variables should show as Interval

- If Smoker_flag or Plaque_Prolapse show as Interval but should be treated as classes, you can change their Level to Nominal — but for this exercise, leave them as-is to match the SAS VA approach.

Section 2: Configure Project Settings

Before building the pipeline, configure the project-level settings that control partitioning and model comparison behavior.

Step 1: Click the Settings icon (gear icon) in the upper right of the project, or go to Project Settings via the menu.

Part 2.1: Partition Data

Step 2: Under the Partition Data section:

- **a.** Set the Partitioning method to Simple Random Sampling.
- **b.** Set the Training percentage to 70%.
- **c.** The Validation percentage should automatically populate to 30%.
- **d.** Check the box for Random number seed and enter: 772020

> **Tip:** *Using the same seed (772020) as the SAS Visual Analytics exercise means your train/validation split will be as comparable as possible. This makes cross-platform comparison meaningful.*

Part 2.2: Model Comparison Rules

Step 3: Under the Rules section, configure the Model Comparison settings:

- **a.** Class selection statistic: Choose Misclassification Rate (or leave as default — you can change this later).
- **b.** Selection partition: Default (the system will use Validation if available).
- **c.** Check the box to Override the default classification cutoff.
- **d.** Set the cutoff value to 0.3.

> ⚠ **Note:** *Why 0.3 instead of 0.5? In healthcare, missing a real stent failure (false negative) is far more dangerous than a false alarm. A lower cutoff means we flag more patients for follow-up — better safe than sorry. This matches the prediction cutoff used in the SAS VA exercise.*

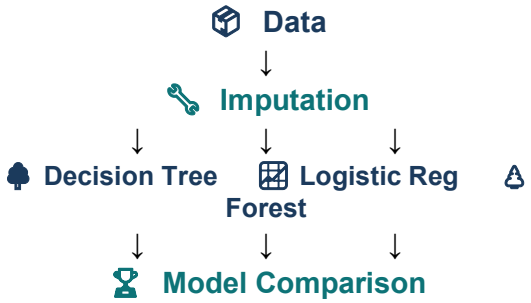
Step 4: Click Save to apply all settings.

Section 3: Build the Pipeline

Now comes the main event. You will build a pipeline — a connected workflow of nodes — that handles imputation, runs three models, and compares them automatically.

What Your Final Pipeline Will Look Like

When complete, your pipeline should have this structure:



Part 3.1: The Data Node

Step 1: Click on the Pipelines tab in your project.

Step 2: If you selected a Blank template, you should see a single Data node. This node is automatically connected to the Stent Failure dataset you selected when creating the project.

Step 3: Click on the Data node to verify it shows the correct dataset. You should see a summary of the data, variable counts, and the target variable (Stent_Failure).

Part 3.2: Add the Imputation Node

In the SAS VA exercise, you manually created calculated items (Thickness_Imp, Width_Imp, Length_Imp) using IF-ELSE logic. In Model Studio, the Imputation node handles this automatically.

Step 1: In the pipeline canvas, click the + icon or right-click after the Data node to add a node.

Step 2: From the node menu, under Data Mining Preprocessing (or Miscellaneous, depending on your version), select Imputation and add it to the pipeline.

Step 3: Connect the Data node to the Imputation node by dragging a line from the Data node output to the Imputation node input (if not already connected).

Step 4: Click on the Imputation node to review its default settings:

- By default, the Imputation node replaces missing values in interval (numeric) variables with the mean.
- For class (categorical) variables, it treats missing values as a distinct level.

> **Tip:** *The SAS VA exercise replaced missing values with 0. Model Studio defaults to the mean, which is statistically more defensible. For this exercise, you can leave the default (mean imputation) — it will produce slightly different but valid results compared to your VA analysis.*

Step 5: After running the pipeline, you can click on the Imputation node results to see which variables were imputed and how.

Part 3.3: Add the Decision Tree Node

A Decision Tree splits the data into segments using a series of if-then rules. The first split is based on the variable that best separates stent failures from non-failures.

Step 1: Click the + icon after the Imputation node to add a new node.

Step 2: Under Supervised Learning, select Decision Tree.

Step 3: Connect the Imputation node output to the Decision Tree node input.

> ⚠ **Note:** *When you add your first supervised learning node, Model Studio automatically creates a Model Comparison node at the bottom of the pipeline. This is normal and expected.*

Step 4: Click on the Decision Tree node to view its properties panel. Review the default settings:

- The target should already be set to Stent_Failure (inherited from the project).
- All Input variables should be included as predictors.
- Default settings for maximum depth, leaf size, etc. are generally fine for an initial model.

Step 5: (Optional) If you want to customize the tree, you can adjust:

a. Maximum Depth — controls how deep the tree can grow (lower = simpler, less overfit)

b. Minimum Leaf Size — the smallest number of observations allowed in a leaf

> **Tip:** *For your first run, leave defaults. After reviewing the results, you can always re-run with different settings to see how they affect performance.*

Part 3.4: Add the Logistic Regression Node

Logistic Regression estimates the probability that a patient will experience stent failure, based on a weighted combination of all input variables. It produces interpretable coefficients — you can see exactly how much each factor contributes to the prediction.

Step 1: Click the + icon after the Imputation node to add another node.

Step 2: Under Supervised Learning, select Logistic Regression.

Step 3: Connect the Imputation node output to the Logistic Regression input.

> **Tip:** *Notice that the Logistic Regression and Decision Tree are both connected from the same Imputation node — they run in parallel, not in sequence. This means both models see the same cleaned data.*

Step 4: Click on the Logistic Regression node and configure:

a. Selection Method: Choose Stepwise (this lets the model automatically select the most important variables, similar to what you did in SAS VA).

b. Selection Criterion: Leave as default (SBC or AIC).

Step 5: Verify that the target and input variables are correctly inherited from the project settings.

Part 3.5: Add the Forest Node

A Forest (Random Forest) builds many decision trees — each trained on a random sample of data and a random subset of variables — then combines their predictions by majority vote. This typically gives more stable and accurate predictions than a single tree.

Step 1: Click the + icon after the Imputation node to add another node.

Step 2: Under Supervised Learning, select Forest.

Step 3: Connect the Imputation node output to the Forest node input.

Step 4: Click on the Forest node to review its properties:

- **a.** Number of Trees: The default is typically 100. Leave as-is.
- **b.** Maximum Depth: Leave as default.
- **c.** All other settings can remain at defaults for the initial run.

Part 3.6: Verify the Model Comparison Node

The Model Comparison node was automatically added when you created your first supervised learning node. It compares all models in the pipeline using the metrics and cutoff you specified in Project Settings.

Step 1: Click on the Model Comparison node to verify:

- **a.** All three models (Decision Tree, Logistic Regression, Forest) are listed.
- **b.** The prediction cutoff is 0.3 (inherited from Project Settings).
- **c.** The selection statistic matches what you configured.

Step 2: If any model is not connected to the Model Comparison node, drag a connection line from the model node to the Model Comparison node.

Section 4: Run the Pipeline & Evaluate Results

Part 4.1: Run the Pipeline

Step 1: Click the Run Pipeline button in the upper right corner of the pipeline view.

Step 2: Wait for the pipeline to complete. You will see a green checkmark on each node as it finishes. The entire pipeline typically takes 2-5 minutes depending on server load.

> ⚠ **Note:** *If a node shows a red X, click on it to view the error log. Common issues include: incorrect variable roles, missing target variable, or data not loaded in CAS memory.*

Step 3: Once complete, each node will display a green checkmark. You can now click on any node to explore its results.

Part 4.2: Examine the Imputation Node Results

Step 1: Click on the completed Imputation node.

Step 2: Review the Input Variable Statistics table. For each variable, verify:

- Which variables had missing values (Number Missing > 0)
- Which imputation method was applied (Mean for interval variables)

Q1: *Which variables had missing values? How does this compare to what you found in SAS VA?*

Q2: *What imputation method was applied to each?*

Part 4.3: Examine the Decision Tree Results

Step 1: Click on the completed Decision Tree node.

Step 2: Explore the results tabs:

a. Tree Diagram — Shows the tree structure with splits and leaf predictions. Click on nodes to see the splitting rules.

b. Variable Importance — Bar chart showing which variables contributed most to the tree. Compare this to the SAS VA results.

c. Fit Statistics — Contains misclassification rates, AUC, and KS statistic for both training and validation.

d. ROC Chart — The ROC curve for training and validation partitions.

e. Lift Chart — Shows how much better the model performs compared to random chance.

f. Misclassification Chart (Assessment) — The confusion matrix showing true/false positives and negatives.

Decision Tree Questions

Q1: *Which variables are being used by the model? (Check Variable Importance)*

Q2: *Which variable is the first split? On which value?*

Q3: *How many leaves does the tree have?*

Q4: *What is the Validation AUC (from the ROC chart)?*

Q5: *What is the KS Statistic (from the Fit Statistics)?*

Q6: *How many false negatives are there in Validation (from the Misclassification/Assessment chart)?*

Q7: *What are the Misclassification Rates on Training and Validation?*

Part 4.4: Examine the Logistic Regression Results

Step 1: Click on the completed Logistic Regression node.

Step 2: Explore the results tabs:

a. Variable Importance / Effects Plot — Shows the relative importance of each variable, ranked by significance.

b. Iteration Plot (if stepwise selection was used) — Shows which variables entered the model at each step.

c. Fit Statistics — Misclassification rates, AUC, KS statistic.

d. ROC Chart — Compare the curve shape to the Decision Tree.

e. Assessment (Misclassification Chart) — Confusion matrix for training and validation.

Logistic Regression Questions

Q1: *Which variables were selected by the model? (If using Stepwise selection)*

Q2: *What is the Validation AUC?*

Q3: *What is the KS Statistic?*

Q4: *How many false negatives in Validation?*

Q5: *What are the Misclassification Rates on Training and Validation?*

Q6: *Do you see any difference in which variables the Logistic Regression selected compared to the Decision Tree? Why might they differ?*

Part 4.5: Examine the Forest Results

Step 1: Click on the completed Forest node.

Step 2: Explore the results tabs:

a. Variable Importance — Compare the ranking of important variables to the Decision Tree and Logistic Regression.

b. Fit Statistics — AUC, KS, misclassification rate.

c. ROC and Lift Charts — Compare the Forest curves to the other models.

Forest Questions

Q1: *Which variables are the most important in the Forest model?*

Q2: *What is the Validation AUC?*

Q3: *What is the KS Statistic?*

Q4: *How many false negatives in Validation?*

Q5: *What are the Misclassification Rates on Training and Validation?*

Section 5: Model Comparison & Champion Selection

The Model Comparison node pulls together performance metrics from all three models so you can compare them side by side — just like the Model Comparison object you used in SAS VA, but built into the pipeline workflow.

Part 5.1: Review the Model Comparison Results

Step 1: Click on the Model Comparison node.

Step 2: Review the Fit Statistics comparison table. This shows all key metrics for every model, for both training and validation partitions.

Step 3: Examine the following in the Model Comparison results:

a. Champion Model — Model Studio automatically highlights the "champion" based on the selection statistic you configured. Which model was selected?

b. ROC Overlay — The overlaid ROC curves for all models. Which model has the highest AUC on validation?

c. Lift Overlay — Which model captures the most responders (failures) in the top percentiles?

d. Misclassification/Assessment — At the 0.3 cutoff, compare the confusion matrices.

Model Comparison Questions

Q1: *Which model was selected as the champion (at prediction cutoff 0.3)?*

Q2: *Which model has the best (lowest) Validation False Positive Rate?*

Q3: *Which model has the best Validation KS Statistic?*

Q4: *Which model has the fewest false negatives on Validation? Why does this matter in a healthcare context?*

Q5: *Is there a difference between the model with the best AUC and the model with the fewest false negatives? If so, which would you prioritize for deployment in a hospital, and why?*

Part 5.2: Pipeline Comparison (Optional)

If you created multiple pipelines (for example, one with default settings and another with tuned hyperparameters), you can compare champion models across pipelines using the Pipeline Comparison tab at the top of the project.

Step 1: Click the Pipeline Comparison tab at the top of the project.

Step 2: By default, it compares the champion model from each pipeline.

Step 3: Review the cross-pipeline comparison table and charts.

Section 6: Reflection & Cross-Platform Comparison

Now that you have completed the analysis in both SAS Visual Analytics and SAS Model Studio, take a step back and think about the bigger picture.

Discussion Questions

Q1: *Compare your SAS Visual Analytics results to your Model Studio results. Were the same variables important? Were the performance metrics similar? If not, what might explain the differences?*

Q2: *What are the advantages of Model Studio's pipeline approach over building models one at a time in Visual Analytics? In what situations might Visual Analytics be preferred?*

Q3: *Model Studio automatically handled imputation and partitioning for you. In Visual Analytics, you had to create calculated items and a partition variable manually. Which approach do you think is better for learning? Which is better for production work?*

Q4: *You used a prediction cutoff of 0.3 instead of the default 0.5. How did this affect the number of false negatives versus false positives? In a healthcare setting, is this tradeoff worth it?*

Q5: *If a cardiologist asked you to explain why the model flagged a specific patient for follow-up, which model would be easiest to explain — the Decision Tree, Logistic Regression, or Forest? Why?*

Q6: *What are the ethical considerations of deploying a predictive model for stent failure? What happens when the model is wrong? Who should be responsible?*

Summary: SAS VA vs. Model Studio vs. Python

Aspect	SAS Visual Analytics	SAS Model Studio	Python (scikit-learn)
Approach	Interactive drag-and-drop on report canvas	Pipeline-based workflow with connected nodes	Code-based scripting in Jupyter notebooks
Strengths	Great for exploration and dashboarding	Production-ready pipelines, automated comparison	Full flexibility, free and open source
Best for	Data exploration, stakeholder reporting	Building and comparing multiple models efficiently	Custom analysis, integration with other tools
Learning curve	Low — visual and intuitive	Medium — need to understand pipeline concepts	Higher — requires coding skills
Reproducibility	Moderate — settings embedded in report	High — pipeline captures full workflow	High — code documents every step

The key takeaway: The tool is just the vehicle. Whether you use SAS Visual Analytics, SAS Model Studio, Python, R, or any other platform, the analytical thinking is the same — understand the business problem, prepare the data, build and compare models, and communicate results to stakeholders. Master the thinking, and you can pick up any tool.

— End of Instructions —