Getting Started: SAS® Visual Data Mining and Machine Learning

Overview
SAS® Visual Data Mining and Machine Learning combines data wrangling, exploration, feature engineering, and modern statistical, data mining and machine learning techniques in a single, scalable in memory processing environment.

As part of the Project Data Sphere Platform, authorized users can use SAS Visual Data Mining and Machine Learning to:

- **ACCESS** downloadable cancer clinical trial data that has been provided to the Project Data Sphere Platform
- **EXPLORE** data using SAS Studio and SAS Data Studio to learn more about the various datasets contributed with a study
- **VISUALIZE** data using SAS Visual Analytics to better share findings with individuals of all backgrounds
- **HARMONIZE** data using SAS Data Studio to compare or combine data from various studies to drive further research
- **DESIGN MODELS** using SAS Model Studio to discover new insights

Legal Disclaimer
Any sample code are provided as-is and SAS has no obligation to provide continued support for their use. Use of any part or all of this code is not guaranteed to fulfill any legal or other requirements of any specific data governance standard or law governing de-identification of data. Users assume all responsibility for ensuring that the final publication of any de-identified data meets all legal requirements for release of such data. Users assume all legal and ethical responsibility for determining the appropriateness of releasing data analyzed and/or modified using this code.

How to Access
Becoming an Authorized User of Project Data Sphere
To use SAS Visual Data Mining and Machine Learning or other SAS analytical tools, the user must first be an Authorized User of the Project Data Sphere website.

a. For additional information on becoming an Authorized User, see the separate “How To Guide: Becoming an Authorized User” document located in the Resources section of the Project Data Sphere website.

b. After receiving Authorized User login credentials, the user can click on the “Sign in” link in the upper right-hand corner of the Project Data Sphere website to access the SAS analytical tools.
Accessing SAS Visual Data Mining and Machine Learning

1. In the navigation bar, click Access Data.

2. Click the SAS Visual Data Mining and Machine Learning button (pictured below) either on the Access Data page or within the detailed information page of any contribution.

3. A dialog will be displayed with instructions on accessing the tool. After reading the instructions, click ‘GO TO TOOL’ and a new tab will be opened.
   a. Note: You must use your own user ID, not the one picture in the figure below.
4. Following the instructions from the dialog, log into SAS Visual Data Mining and Machine Learning using the specified username and password.
5. After logging in for the first time, you may be prompted to select a profile picture and a theme, but you can skip this setup if desired.

Quick Start
After logging in, the user will be directed to the SAS Drive. SAS Drive is the hub for the SAS Viya applications that enables you to easily view, organize, and share your content from one place. The availability of the features in SAS Drive depends on the applications that have been installed, and the features and permissions that have been specified by your administrator.
The application switcher is in the top left and can be used to access the different applications available.

These actions can be used to access the following tools:

- View Reports: SAS Report Viewer
- Develop SAS Code: SAS Studio
- Prepare Data: Data Studio
- Explore and Visualize Data: SAS Visual Analytics
- Build Models: Model Studio
- Explore Lineage: SAS Lineage Viewer
- Build Graphs: SAS Graph Builder
In addition, while not featured in our Getting Started guides, users also have access to SAS Report Viewer, SAS Lineage Viewer, and SAS Graph Builder which allow users to view reports, understand the relationships between objects in your SAS Viya applications, and to create custom graph objects respectively.

What’s Next?
To learn more about the different applications available with SAS Visual Data Mining and Machine Learning, check out the Quick Start guides created by the Project Data Sphere team.

The guides can be worked through independently, but all are related to a single example of the analytics lifecycle using Project Data Sphere’s DREAM Challenge data sets. In each guide, a user will navigate through a different application to prepare, explore, visualize, or model data.

- **Getting Started: Locating a contribution in the SAS Visual Data Mining and Machine Learning environment** – Learn how the Project Data Sphere contributions are organized in the SAS analytic tools.
- **Getting Started: SAS® Studio** – Learn how to load Project Data Sphere contributions into SAS Studio to start exploring and writing SAS programs.
- **Getting Started: SAS® Data Studio** – Learn how to transform data using Data Studio to prepare it for later analytics.
- **Getting Started: SAS® Visual Analytics** – Learn how to quickly create reports to visualize and explore data.
- **Getting Started: SAS® Model Studio** – Learn how to create analytic pipelines in Model Studio to perform machine learning.
- **Getting Started: API Access to SAS® Visual Data Mining and Machine Learning using Python** – Learn how to use SAS analytics through a Python environment.
- **Getting Started: API Access to SAS® Visual Data Mining and Machine Learning using R** – Learn how to use SAS analytics through an R environment.

Additional Videos
SAS® Viya™: The Big Picture
Getting Started with SAS Visual Data Mining and Machine Learning 8.1 on SAS Viya

SAS Visual Data Mining and Machine Learning on SAS Viya: Overview

Need Help?
Are you still not sure how to get started or need help? Get in touch with the Project Data Sphere team using the Contact Us feature.

You can also check the SAS Support Communities where you can ask questions and get help for a community of SAS experts!
Getting Started: Locating a contribution in the SAS Visual Data Mining and Machine Learning environment

The Unique Dataset ID

All downloadable contributions are automatically available to the SAS analytic tools on the Project Data Sphere platform. Contributions are organized into directories identified by their Unique Dataset ID. The Unique Dataset ID can be identified by finding a contribution on the Access Data page or a specific contribution details page.

Finding the CASLib

In SAS Visual Data Mining and Machine Learning, you can find a contribution either using SAS Studio or the Data Explorer dialog available in most applications (Please see the Getting Started guides for more
In the Data Explorer dialog, click the Data Sources tab and then click cas-shared-default. You should then be able to find the contribution by its Unique Dataset ID.

![Data Explorer Dialog](image)

**Figure 10: Data Explorer Dialog**

Note: If a contribution contains sub directories then you must use SAS Studio to access them. See Getting Started: SAS® Studio for more information.
Getting Started: SAS® Studio

Overview
SAS Studio provides a web browser-based programming environment which lets you access your files and do all your SAS coding from anywhere. The environment will be familiar to SAS programmers who just want to write code, but SAS Studio also includes visual point-and-click tasks that generate code – so you don’t have to. In addition, users will have the ability to create and use existing code snippet libraries of often repeated tasks. Lastly, SQL queries can easily be generated allowing users to select, sort, filter, and join on tables to begin their data exploration.


Quick Start
In this Quick Start, we will learn how to access a Project Data Sphere contribution using SAS Studio.

2. In the application switcher click on Develop SAS Code to navigate to SAS Studio.

   ![Figure 11: Navigate to SAS Studio by clicking Develop SAS Code in the App Switcher](image)

3. Once in SAS Studio, we’re going to add a custom task created by the Project Data Sphere team to help with loading contributions. First, click the Explorer icon in the navigate pane and then navigate to SAS Content -> PDS -> Samples. Next, right click ‘Create Libref for PDS Contribution.ctm’ and select ‘Add to My Tasks’. In the dialog that is displayed, click OK.
4. To open the custom task, click the Tasks icon in the navigation pane and then double click the task displayed under ‘My Tasks’.

5. After the task has been opened in a new tab, the user can enter the desired name of their libname and select a contribution (Caslib) for which to load into memory. Remember, the name of the Caslib corresponds to the Unique Dataset ID.
   a. For this example, ‘dream’ was chosen for the libname because the datasets are from the DREAM challenge and the CASLib ‘Prostat_na_2006_149’ was selected.
b. In addition, if you would like to load and save the data from a contribution into your CASUSER library you can check the checkbox. For this example, we will leave the checkbox unchecked.

![LIBNAME](#)

Define the desired name of the Libref.

Libref:

```
dream
```

Load CASLIB into CASUSER library instead of a Libref.

![CONTRIBUTION](#)

Select the Unique Data Set ID of the PDS Contribution you would like to load into memory.

Contribution: *

```
Prostat_na_2006_149
```

*Figure 14: Task Data tab*

6. In most cases this should be all you have to do, but the DREAM challenge data does not match the default configuration expected by SAS Viya (the default is UTF-8). Click the OPTIONS tab and change the selected encoding to Latin1.

![ENCODING](#)

Select the encoding you would like to use when loading the contribution into memory. If the encoding of the files is not displayed below, please type in a custom encoding value.

Encoding:

```
Latin1
```

*Figure 15: Task Option tab*
7. Click the Submit button!
8. Lastly, navigate to the Libraries section of the navigation pane by clicking the Libraries icon. You will find all the data files in the DREAM challenge contribution have been loaded into the specified library.

![Figure 16: Data files loaded into Library](image)

You’re all done! Now you can begin exploring the individual data files and start programming with them in SAS Studio.

Documentation

Additional Videos
SAS® Viya™ CAS Libraries (Caslibs) Simplified
Understanding Caslibs and Loading Data in SAS® Viya™
Using the DATA Step in SAS® Viya™
Accessing SAS® Viya® Tasks in SAS® Studio or SAS® Enterprise Guide®
SAS® Viya™ Programming for SAS® 9 Programmers: Overview
Getting Started: Data Studio

Overview
SAS Data Studio provides an easy way to prepare data for analysis. With Data Studio users can create plans containing a variety of data transformations such as joining tables, appending data to tables, transposing columns, and creating calculated columns. Users can also view profiles of tables to view metrics such as pattern and frequency distributions. Data Studio is a self-service application, which users without data management programming skills can use to transform data to solve their analytic problems.

Quick Start
In this Quick Start, we will prepare data for visualization and model building. To do this, we will learn how to use Data Studio to transform a table from the DREAM challenge contribution into an analytics-ready state.

2. First, we must load data from the PDSSamples CASLib into our CASUSER CASLib.
   a. Note: If you have completed this step in another Getting Started guide, then you can proceed to step 3. The PDSSamples CASLib contains sample files for the Getting Started guides and is different from the files used in Getting Started: SAS® Studio.
   b. In the application switcher, click on Develop SAS Code to navigate to SAS Studio.
   c. Using the ‘Create Libref for PDS Contribution’ task under My Tasks load the data from the PDSSamples into your CASUSER library.

\[Figure 17: \text{Navigate to SAS Studio by clicking Develop SAS Code in the App Switcher}\]
3. Next, in the application switcher, click on Prepare Data to navigate to SAS Data Studio.

4. Once in Data Studio, click ‘New Plan’.
5. Next, we must select the table we’re going to transform. In the Choose Data dialog that opens, select Data Sources from the top three tabs and then select cas-shared-default, and then finally select CASUSER(YourUserId).

Figure 21: Choosing table to use in Data Studio

6. Next, double click on CASUSER.

7. To prepare data in Data Studio, the table we want to transform must first be loaded into memory. We want to work on the DC_CORE_TABLE.sashdat, but it may already be loaded into memory for us. The table is already loaded into memory if there is a row titled DC_CORE_TABLE and it has a lightning bolt icon to its left.

Figure 22: The DC_CORE_TABLE loaded into memory

8. If you do not see the in memory table, then select DC_CORE_TABLE.sashdat and then click the lightning bolt icon in the top right. Otherwise, just proceed to the next step.
a. From this view you can view details about the columns in the table, view a sample of the first 100 rows, and even profile the table to find more insights.

9. Select the in-memory table DC_CORE_TABLE (not DC_CORE_TABLE.sashdat) and then click OK.

![Figure 23: View information about a selected table](image)

![Figure 24: DC_CORE_TABLE loaded into memory](image)
10. Now that we have selected a table we can begin to transform it for our analytic goals. The first step we are going to take is to add a Custom Code transform to add labels to the columns in the table and transform the various flag columns in the table to store ‘Y’ and ‘N’. To do this, click the Code item in the left pane and then click ‘Add Transform’.

Transforms

![Add Transform](image)

- Column Transforms
  - Change case
  - Convert column
  - Remove
  - Rename
  - Split
  - Trim whitespace

- Custom Transforms
  - Calculated column
    - Code

*Figure 25: Adding Custom Code Transform*

11. In the Code transform step that is created, copy the following code snippet and in the program editor paste it below line 5:

```sql
/* Add labels based on the data dictionary */
label DOMAIN="Domain" STUDYID="Study Identifier" RPT="Patient ID" LKADT_P="Last Known Alive Dt Period in days" DEATH="Patient Died Flag" DISCONT="Discontinue Flag" ENDTRS_C="Discontinue Trt Reason (category)" ENTRT_PC="Discontinue Trt Date Period in days" PER_REF="Reference Day used for ENTRT_PC" LKADT_REF="Reference Day used for LKADT_P" LKADT_PER="Period unit for period values, LKADT_P" GLEAS_DX="Gleason Score at Initial diagnosis" TSTAG_DX="Primary Tumor-stage Score at Initial diagnosis" AGEGRP="Age Group" AGEGRP2="Age Group (3 category)" RACE_C="Race" BMI="Baseline Body Mass Index (kg/m2)" WEIGHTBL="Baseline Weight in kg" HGTBLCAT="Baseline Height in cm category" TSTAG_DX="Primary Tumor-stage Score at Initial diagnosis" AGEGRP="Age Group" AGEGRP2="Age Group (3 category)" RACE_C="Race" BMI="Baseline Body Mass Index (kg/m2)" WEIGHTBL="Baseline Weight in kg" HGTBLCAT="Baseline Height in cm category" REGION_C="Region of the World" SMOKE="Ever Smoked" SMOKFREQ="Smoking Frequency" SMOKSTAT="Current Smoking Status" ECOG_C="Baseline Patient Performance Status" TRT1_ID="Treatment 1 product " TRT2_ID="Treatment 2 product (docetaxel)"
```
FRACTURES" SPINCOMP="MEDICAL HISTORY: SPINAL CORD COMPRESSION" COPD="MEDICAL
HISTORY: CHRONIC OBSTRUCTIVE PULMONARY DISEASE" MHBLOOD="MEDICAL HISTORY (Body
system): BLOOD & LYMPHATIC SYSTEM" MHCARD="MEDICAL HISTORY (Body system): CARDIAC
DISORDERS" MHCONGEN="MEDICAL HISTORY (Body system): CONGENITAL, FAMILIAL &
GENETIC" MHEAR="MEDICAL HISTORY (Body system): EAR & LABYRINTH" MHENDO="MEDICAL
HISTORY (Body system): ENDOCRINE DISORDERS" MHEYE="MEDICAL HISTORY (Body system):
EYE DISORDERS" MHEAR="MEDICAL HISTORY (Body system): GASTROINTESTINAL
DISORDERS" MHGEN="MEDICAL HISTORY (Body system): GEN DISORD & ADMIN SITE"
MHHEPATO="MEDICAL HISTORY (Body system): HEPATOBILIARY DISORDERS"
MHIMMU="MEDICAL HISTORY (Body system): IMMUNE SYSTEM DISORDERS"
MHINJURY="MEDICAL HISTORY (Body system): INJURY, POISON & PROCEDURAL"
MHINVEST="MEDICAL HISTORY (Body system): INVESTIGATIONS" MHMETAB="MEDICAL
HISTORY (Body system): METABOLISM & NUTRITION" MHMUSCLE="MEDICAL HISTORY (Body
system): MUSC/SKELETAL & CONNECT TISSUE" MHNEOPLA="MEDICAL HISTORY (Body system):
NEOPLASMS BENIGN, MALIG & UNSPEC" MHNERV="MEDICAL HISTORY (Body system):
NERVOUS SYSTEM DISORDERS" MHPSYCH="MEDICAL HISTORY (Body system): PSYCHIATRIC
DISORDERS" MHNRESP="MEDICAL HISTORY (Body system): RESP, THORACIC & MEDIASTINAL"
MHRENAL="MEDICAL HISTORY (Body system): SKIN & SUBCUTANEOUS TISSUE"
MHSOCIAL="MEDICAL HISTORY (Body system): SOCIAL CIRCUMSTANCES" MHSURG="MEDICAL
HISTORY (Body system): SURGICAL & MEDICAL PROCEDURES" MHVASC="MEDICAL HISTORY
(Body system): VASCULAR DISORDERS"

/* Make binary variables YES/NO instead of YES/missing */
if missing(DEATH) then death='NO'; else death='YES';
if missing(NON_TARGET) then NON_TARGET='NO'; else NON_TARGET='YES';
if missing(TARGET) then TARGET='NO'; else TARGET='YES';
if missing(BONE) then BONE='NO'; else BONE='YES';
if missing(RECTAL) then RECTAL='NO'; else RECTAL='YES';
if missing(LYMPH_NODES) then LYMPH_NODES='NO'; else LYMPH_NODES='YES';
if missing(KIDNEYS) then KIDNEYS='NO'; else KIDNEYS='YES';
if missing(LUNGS) then LUNGS='NO'; else LUNGS='YES';
if missing(LIVER) then LIVER='NO'; else LIVER='YES';
if missing(PLEURA) then PLEURA='NO'; else PLEURA='YES';
if missing(OTHER) then OTHER='NO'; else OTHER='YES';
if missing(PROSTATE) then PROSTATE='NO'; else PROSTATE='YES';
if missing(ADRENAL) then ADRENAL='NO'; else ADRENAL='YES';
if missing(BLADDER) then BLADDER='NO'; else BLADDER='YES';
if missing(PERITONEUM) then PERITONEUM='NO'; else PERITONEUM='YES';
if missing(COLON) then COLON='NO'; else COLON='YES';
if missing(HEAD_AND_NECK) then HEAD_AND_NECK='NO'; else HEAD_AND_NECK='YES';
if missing(SOFT_TISSUE) then SOFT_TISSUE='NO'; else SOFT_TISSUE='YES';
if missing(STOMACH) then STOMACH='NO'; else STOMACH='YES';
if missing(PANCREAS) then PANCREAS='NO'; else PANCREAS='YES';
if missing(THYROID) then THYROID='NO'; else THYROID='YES';
if missing(ABDOMINAL) then ABDOMINAL='NO'; else ABDOMINAL='YES';
if missing(ORCHIDECTOMY) then ORCHIDECTOMY='NO'; else ORCHIDECTOMY='YES';
if missing(PROSTATECTOMY) then PROSTATECTOMY='NO'; else PROSTATECTOMY='YES';
if missing(TURP) then TURP='NO'; else TURP='YES';
if missing(LYMPHADENECTOMY) then LYMPHADENECTOMY='NO'; else LYMPHADENECTOMY='YES';
if missing(SPINAL_CORD_SURGERY) then SPINAL_CORD_SURGERY='NO'; else SPINAL_CORD_SURGERY='YES';
if missing(BILATERAL_ORCHIDECTOMY) then BILATERAL_ORCHIDECTOMY='NO'; else BILATERAL_ORCHIDECTOMY='YES';
if missing(PRIOR_RADIOTHERAPY) then PRIOR_RADIOTHERAPY='NO'; else PRIOR_RADIOTHERAPY='YES';
if missing(ANALGESICS) then ANALGESICS='NO'; else ANALGESICS='YES';
if missing(ANTI_ANDROGENS) then ANTI_ANDROGENS='NO'; else ANTI_ANDROGENS='YES';
if missing(GLUCOCORTICOID) then GLUCOCORTICOID='NO'; else GLUCOCORTICOID='YES';
if missing(GONADOTROPIN) then GONADOTROPIN='NO'; else GONADOTROPIN='YES';
if missing(BISPHOSPHONATE) then BISPHOSPHONATE='NO'; else BISPHOSPHONATE='YES';
if missing(CORTICOSTEROID) then CORTICOSTEROID='NO'; else CORTICOSTEROID='YES';
if missing(IMIDAZOLE) then IMIDAZOLE='NO'; else IMIDAZOLE='YES';
if missing(ACE_INHIBITORS) then ACE_INHIBITORS='NO'; else ACE_INHIBITORS='YES';
if missing(HMG_COA_REDUCT) then HMG_COA_REDUCT='NO'; else HMG_COA_REDUCT='YES';
if missing(ESTROGENS) then ESTROGENS='NO'; else ESTROGENS='YES';
if missing(ANTI_ESTROGENS) then ANTI_ESTROGENS='NO'; else ANTI_ESTROGENS='YES';
if missing(ARTTHROM) then ARTTHROM='NO'; else ARTTHROM='YES';
if missing(CEREBACC) then CEREBACC='NO'; else CEREBACC='YES';
if missing(CHF) then CHF='NO'; else CHF='YES';
if missing(DVT) then DVT='NO'; else DVT='YES';
if missing(DIAB) then DIAB='NO'; else DIAB='YES';
if missing(GASTREFL) then GASTREFL='NO'; else GASTREFL='YES';
if missing(GIBLEED) then GIBLEED='NO'; else GIBLEED='YES';
if missing(MI) then MI='NO'; else MI='YES';
if missing(PUD) then PUD='NO'; else PUD='YES';
if missing(PULMEMB) then PULMEMB='NO'; else PULMEMB='YES';
if missing(PATHFRAC) then PATHFRAC='NO'; else PATHFRAC='YES';
if missing(SPINCOMP) then SPINCOMP='NO'; else SPINCOMP='YES';
if missing(COPD) then COPD='NO'; else COPD='YES';
if missing(MHBLOOD) then MHBLOOD='NO'; else MHBLOOD='YES';
if missing(MHCARD) then MHCARD='NO'; else MHCARD='YES';
if missing(MHCONGEN) then MHCONGEN='NO'; else MHCONGEN='YES';
if missing(MHEAR) then MHEAR='NO'; else MHEAR='YES';
if missing(MHENDO) then MHENDO='NO'; else MHENDO='YES';
if missing(MHEYE) then MHEYE='NO'; else MHEYE='YES';
if missing(MHGASTRO) then MHGASTRO='NO'; else MHGASTRO='YES';
if missing(MHGEN) then MHGEN='NO'; else MHGEN='YES';
if missing(MHEPATO) then MHHEPATO='NO'; else MHHEPATO='YES';
if missing(MHIMMUNE) then MHIMMUNE='NO'; else MHIMMUNE='YES';
if missing(MHINFE) then MHINFECT='NO'; else MHINFE='YES';
if missing(MHINJURY) then MHINJURY='NO'; else MHINJURY='YES';
if missing(MHINV) then MHINV='NO'; else MHINV='YES';
if missing(MHMETA) then MHMETAB='NO'; else MHMETAB='YES';
if missing(MHMUSCLE) then MHMUSCLE='NO'; else MHMUSCLE='YES';
if missing(MHNEOPLA) then MHNEOPLA='NO'; else MHNEOPLA='YES';
if missing(MHNERV) then MHNERV='NO'; else MHNERV='YES';
if missing(MHPSYCH) then MHPSYCH='NO'; else MHPSYCH='YES';
if missing(MHRENAL) then MHRENAL='NO'; else MHRENAL='YES';
if missing(MHRSP) then MHRSP='NO'; else MHRSP='YES';
if missing(MHSKIN) then MHSKIN='NO'; else MHSKIN='YES';
if missing(MHSOCIAL) then MHSOCIAL='NO'; else MHSOCIAL='YES';
if missing(MHSURG) then MHSURG='NO'; else MHSURG='YES';
if missing(MHVASC) then MVASC='NO'; else MVASC='YES';

Figure 26: Custom Code Transform in Data Studio

12. After updating the custom code transform, click Run. You should see the updates made to the table by the custom code in the table below.
13. Next, we’ll add a Remove transform to drop some columns from the source table that are not needed. In the transform pane, select the Remove item and then click Add Transform.

**Transforms**

![Add Transform Button]

- Column Transforms
  - Change case
  - Convert column
  - Remove
  - Rename
  - Split
  - Trim whitespace

![Figure 27: Adding Remove Transform]

14. In the Remove transform step, drop the following columns: LKADT_P, LKADT_REF, LKADT_PER, DOMAIN, STUDYID, DISCONT, ENDTRS_C, ENTRT_PC, and PER_REF.

![Figure 28: The Remove Transform with columns selected for removal]

15. Click Run.
16. In our last step, we’re going to rename some columns using the Rename transform. In the transform pane, select the Rename item and then click Add Transform.

![Add Transform](image)

**Transforms**

- Column Transforms
  - Change case
  - Convert column
- Remove
  - Rename
  - Split
  - Trim whitespace

*Figure 29: Adding a 'Rename' transform to the data plan*

17. In the rename transform step, rename the NON_TARGET column to NON_TARGET_LESIONS and the TARGET column to TARGET_LESIONS. This rename will make it more easily known what information those columns store.

   a. Note, you may have to click ‘View all columns’ at the bottom of the source column dropdown to find the NON_TARGET and TARGET columns.

![Rename Transform](image)

*Figure 30: The Rename transform with columns selected to rename*

18. Click Run.

If you would like to save your plan and transform table. Following the below steps:

19. Click the SAVE button.
20. In the Save As dialog, you can save the data plan in your My Folder and the table in your CASUSER library.
   a. In the tree view, you may select the directory in which you save your data plan. We suggest saving it under My Folder, which should be selected by default.
   b. Under name, you may name the data plan anything you desire.
   c. Under table name, you may name the transformed DC_CORE_TABLE. We suggest naming it DC_CORE_TABLE_PR.
   d. Under library, you may select the CasLib to store the transformed table in. You must select your CASUSER library.

21. Lastly, click the Save button.

You’re all done! This Quick Start should have shown you the power available with Data Studio to quickly transform data to fit your research and analytic goals.

Documentation
SAS® Data Studio 2.2: User’s Guide

Additional Videos
Using SAS® Data Explorer to Load SAS® Data Sets
Using SAS® Data Explorer to Profile Data for Better Understanding
Using SAS® Data Studio to Prepare and Transform Data into the Desired Format
Getting Started: Model Studio

Overview

SAS Model Studio is an integrated visual environment that provides a suite of analytic data mining tools to facilitate end-to-end data mining analysis. In Model Studio, users can leverage the visual web interface to assemble, configure, build, and compare data mining models and pipelines for a wide range of analytic data mining tasks. Data miners of all levels can use Model Studio and most tasks can be used by individuals who have minimal SAS programming experience.

Quick Start

In this Quick Start, we will learn how to use Model Studio to create various machine learning models and pipelines with the goal of finding the pipeline that best classifies the data.

We will be using the DC_CORE_TABLE_PR table that we created in the Getting Started: Data Studio guide. If you have not completed that Getting Started guide or want to skip it, you can use the sample table located in the PDSSamples CASLib.

2. First, we must load data from the PDSSamples CASLib into our CASUSER CASLib.
   a. Note: If you have completed this step in another Getting Started guide, then you can proceed to step 3. The PDSSamples CASLib contains sample files for the Getting Started guides and is different from the files used in Getting Started: SAS® Studio.
   b. In the application switcher, click on Develop SAS Code to navigate to SAS Studio.

Figure 33: Navigate to SAS Studio by clicking Develop SAS Code in the App Switcher
c. Using the ‘Create Libref for PDS Contribution’ task under My Tasks load the data from the PDSSamples into your CASUSER library.

- **LIBNAME**
  - Define the desired name of the Libref.
  - Libref:
  - ![Check box] Load CASLIB into CASUSER library instead of a Libref.

- **CONTRIBUTION**
  - Select the Unique Data Set ID of the PDS Contribution you would like to load into memory.
  - Contribution: *
    - PDSSamples

*Figure 34 SAS Studio task to load data from PDSSamples to CASUSER library*

3. Next, in the application switcher, click on Build Models to navigate to Model Studio.

![Screen capture of the application switcher with Build Models highlighted]

*Figure 35: Navigate to SAS Model Studio by clicking Build Models in the App Switcher*

4. Once in Model Studio, click the New Project button.
5. In the New Project dialog, fill out the following:
   a. Name: Any name that you desire
   b. Data Source
      i. Click Browse.
      ii. In the Browse Data dialog, select the Data Sources tab and then click cas-shared-default.

   ![Browse Data](image)

   iii. Next, go to your CASUSER library and select the ‘DC_CORE_TABLE_PR’ in-memory table you created during the Data Studio Quick Start.
6. Click OK and then click Save.
7. Once the project opens, we first need to make a few changes in the Data tab. We want our target variable to be the Patient Died Flag (the variable named DEATH) and Model Studio may automatically set the target variable to something else.
   a. In the table find the target variable by looking at the Role column, if it is not the DEATH variable, then click on the row and then change the Role to Input in the properties pane on the right side.
   b. Next, in the table find the DEATH variable and change its role to Target.

8. Now we’re ready to start creating analytical pipelines. Click the Pipelines tab.
9. We are going to create two new pipelines called Logistic Regression and Gradient Boosting. Double-click the default pipeline (named ‘Pipeline 1’) to rename it. To create the other new pipeline click the plus icon. In the New Pipeline dialog assign a name and click Save.

10. First, let’s work on the Logistic Regression pipeline. Click on the Logistic Regression tab.
Figure 40: Select nodes from Nodes pane
a. Open the nodes pane on the left side, and then we’re going to add two different data mining preprocessing nodes: Variable Selection and Variable Clustering. Drag and drop these objects onto the Data node.

![Diagram showing nodes](image)

Figure 41: Added preprocessing nodes for Logistic Regression pipeline

b. After that, we’re going to add three logistic regression nodes from the supervised learning section. Three should be attached to each of the preprocessing nodes and one to the data node.

i. For each of the logistic regression nodes change the name, so it’s easier to discern the different nodes and also change the configuration options for each node as noted below:

1. Variable Selection
   a. Name: Logistic Regression (VarSel)
      i. Note: To change the name of a node, right-click the node and select rename.
   b. Options
      i. Selection Method: None
      ii. Note: To change the options, first select the options panel in the right pane.

2. Variable Clustering
   a. Name: Logistic Regression (VarClus)
   b. Options
      i. Selection Method: None

3. Logistic Regression
   a. Name: Logistic Regression (Backward selection)
   b. Options
i. Selection Method: Fast backward

Figure 42: Completed Logistic Regression pipeline

c. Click Run to execute the pipeline.
   i. Note: You can continue work while a pipeline is executing.

Figure 43: Executing pipeline

11. Next, let’s work on the Gradient Boosting pipeline. Click on the Gradient Boosting tab.
Figure 44: Selecting nodes from Nodes pane
a. Open the nodes pane on the left side, and add two different data mining preprocessing nodes: Variable Selection and Feature Extraction. Drag and drop these objects onto the Data node.

b. Next, we’re going to add three Gradient Boosting nodes from the supervised learning section. Three should be attached to each of the preprocessing nodes and one to the Data node.
   i. For each of the gradient boosting nodes change the name, so it’s easier to discern the different nodes as noted below:
      1. Variable Selection
         a. Name: Gradient Boosting (VarSel)
      2. Feature Extraction
         a. Name: Gradient Boosting (Ext)
      3. Gradient Boosting
         a. Name: Gradient Boosting

c. Click Run to execute the pipeline.

12. At this point, we’ve finished setting up all our pipelines and we can go to the Pipeline Comparison tab to see which of our pipelines performs the best. Click on the Pipeline Comparison tab.
13. In the Pipeline Comparison tab, the pipelines should be ranked by the default model performance statistic. Since this is a classification problem, the default is the KS (Youden) statistic.
   a. Note: You may receive different results from what we have received.

<table>
<thead>
<tr>
<th>Champion</th>
<th>Name</th>
<th>Algorithm Name</th>
<th>Pipeline Name</th>
<th>KS (Youden)</th>
</tr>
</thead>
<tbody>
<tr>
<td>✅</td>
<td>Gradient Boosting</td>
<td>Gradient Boosting</td>
<td>Gradient Boosting</td>
<td>0.684</td>
</tr>
<tr>
<td>❌</td>
<td>Logistic Regression</td>
<td>Logistic Regression</td>
<td>Logistic Regression</td>
<td>0.121</td>
</tr>
</tbody>
</table>

*Figure 46: Comparing pipeline accuracy.*

You’re all done! With this Quick Start you’ve learned how to navigate Model Studio and seen how easy it is to set up analytical pipelines.

**Documentation**

*SAS® Visual Data Mining and Machine Learning 8.3: User’s Guide*

**Additional Videos**

*Getting Started with Data Mining and Machine Learning Pipelines on SAS® Viya®*

*Managing Variables in Model Studio*

*The Variable Selection Node in Model Studio*

*Training Gradient Boosting Models in Model Studio*
Getting Started: SAS® Visual Analytics

Overview

SAS Visual Analytics provides a modern, integrated environment for discovery and exploration. Users, including those without advanced analytical skills, can examine and understand patterns, trends, and relationships in data. Reports and dashboards can easily be created and shared to display analyses and visualizations to help everyone get insights from data to better solve complex problems.

SAS® Visual Analytics 8.3 Video

Quick Start

In this Quick Start, we will learn how to use SAS Visual Analytics to create data visualizations in a report to present information and find new insights.

We will be using the DC_CORE_TABLE_PR table that we created in the Getting Started: Data Studio guide. If you have not completed that Getting Started guide or want to skip it, you can use the sample table located in the PDSSamples CASLib.

2. First, we must load data from the PDSSamples CASLib into our CASUSER CASLib.
   a. Note: If you have completed this step in another Getting Started guide, then you can proceed to step 3. The PDSSamples CASLib contains sample files for the Getting Started guides and is different from the files used in Getting Started: SAS® Studio.
   b. In the application switcher, click on Develop SAS Code to navigate to SAS Studio.

Figure 47: Navigate to SAS Studio by clicking Develop SAS Code in the App Switcher
c. Using the ‘Create Libref for PDS Contribution’ task under My Tasks load the data from the PDSSamples into your CASUSER library.

![LIBNAME](image)

- **LIBNAME**
  - Define the desired name of the Libref.
  - **Libref:**
  - **Load CASLIB into CASUSER library instead of a Libref.**

![CONTRIBUTION](image)

- **CONTRIBUTION**
  - Select the Unique Data Set ID of the PDS Contribution you would like to load into memory.
  - **Contribution:**
  - PDSSamples

*Figure 48 SAS Studio task to load data from PDSSamples to CASUSER library*

3. In the application switcher, click on ‘Explore and Visualize Data’ to navigate to SAS Visual Analytics.

![Explore and Visualize Data](image)

*Figure 49: Navigate to SAS Visual Analytics by clicking Explore and Visualize Data in the App Switcher*
4. To get started creating a SAS Visual Analytics report, we will click the 'Data' button.

Welcome to SAS Visual Analytics

Select an option to get started:

- Data
- New
- Open

☐ Make this selection the default

Figure 50: Creating a new SAS Visual Analytics report

5. In the Open Data Source dialog that is displayed, select the Data Sources tab and then select cas-shared-default. Next, use the ‘DC_CORE_TABLE_PR’ in-memory table stored in your CASUSER directory. After selecting the table, click OK.

Figure 51: Selecting table to use for report
6. In this report, we’re going to create two tabs: Variable Explanation and Demographics & Baseline Labs. Create the first tab by right-clicking ‘Page 1’ and renaming it to Variable Explanation. Add the other tab by clicking the plus icon and then renaming it to Demographics & Baseline Labs.

   a. Note: The left pane may be covering up the ‘Page 1’ tab. You can pin the left pane by clicking the push tab icon.
7. First, we’ll start with the Variable Explanation tab. Select the objects icon in the left pane and then drag and drop the ‘Automated Analysis’ item into the report.

a. After adding the ‘Automated Analysis’ object to the report, you’ll see a message ‘The required roles have not been assigned a data item’. To define the roles for the object, click the roles icon in the right pane. Then under ‘Response’, select ‘Patient Died Flag’.
   i. Note: The underlying factors will be automatically selected for you.
b. That’s it! The Variable Explanation tab is now complete and you should see a report like the image below.

*Figure 55: Selecting role for Automated Analysis object*

*Figure 56: The completed Variable Explanation report tab*
Next, we’ll work on the Demographics & Baseline Labs tab. In this tab, we’re going to create some basic bar and box charts to examine the distribution of some demographics and baseline lab variables with respect to the Patient Died Flag. First, we’ll add a bar chart and box chart object to the report.

Objects

Filter

- Tables
- Graphs
  - Bar Chart
  - Box Plot
  - Bubble Change Plot
  - Bubble Plot
  - Butterfly Chart

*Figure 57: Adding Bar Chart objects to report*
Figure 58: Adding a bar and box chart to report

a. The charts have been added to the tab, but now we need to set up each of them. We’ll start with the left-side bar chart we want to display the frequency of the age group grouped by the patient died flag. Select the bar chart and then open the roles pane.

i. In the roles pane, under category select ‘Age Group (3 category)’. Under group, select ‘Patient Died Flag’.
ii. Lastly, for this bar chart let’s change the orientation of the bars from being horizontal to vertical. We can do that by going to the options pane and selecting the vertical icon for the direction.
iii. All done! The left-side bar chart is now complete.
Next, let’s work on the right-side box chart. In this box chart, we want to display 'BASELINE LAB VALUE: LACTATE DEHYDROGENASE U/L' baseline lab grouped by the patient died flag.

i. Select the right-side box chart and then in the roles pane, under category select the ‘Patient Died Flag’ column and under measure select the ‘BASELINE LAB VALUE: LACTATE DEHYDROGENASE U/L’ column.
Lastly, let’s change the appearance of the box chart in a few ways. Select the right Options pane and change the following:

1. Box direction: horizontal
2. Outliers: Ignore Outliers
3. Averages: True
9. That it’s for the Demographics & Baseline Labs tab.
If you would like to save your report, follow the instructions below:

10. To save the report, click the save icon in the top right.

11. In the Save As dialog, you must specify where you want to save the report and the name of the report. We suggest saving it in your My Folder and you can name the report whatever you desire.
12. Lastly, click the Save button.

You’re all done! You should now know how to quickly create a report in SAS Visual Analytics.

**Documentation**

[SAS® Visual Analytics 8.3: Overview](#)

[SAS® Visual Analytics 8.3: Getting Started with Reports](#)

**Additional Videos**

[Creating a Basic Report in SAS® Visual Analytics on SAS® Viya®](#)

[Exploring Data in SAS Visual Analytics 8.1 on SAS Viya](#)

[Viewing a Report in SAS Visual Analytics 8.1 on SAS Viya](#)
Getting Started: API access to SAS® Visual Data Mining and Machine Learning using Python

Quick Start
Requirements
To use Python with SAS Cloud Analytic Services (CAS), the client machine that runs Python must meet the following requirements:

- Use 64-bit Linux or 64-bit Windows.
- Use a 64-bit version of Python, such as the Anaconda platform from Continuum Analytics.
- Use Python version 2.7.x or 3.4+. The Python package from SAS, SWAT, is compatible with both versions.
- The SAS Scripting Wrapper for Analytics Transfer (SWAT) package is available for download from https://github.com/sassoftware/python-swat. Information for installing is available from a README that is included in the download.
- Create an authinfo file with your Project Data Sphere credentials. See Create an Authinfo File in Client Authentication Using an Authinfo File.

Connect and Start a Session
To enable a Python program to work with SAS Cloud Analytic Services, you must establish a connection with the server.

To run the following code sample in your Python client, do the following:

- Download pdsviyacert.pem.
- Replace /path/to/pdsviyacert.pem with the absolute path to the downloaded file, pdsviyacert.pem.

```python
import os, swat

# Set the SSL/TLS certificate
os.environ["CAS_CLIENT_SSL_CA_LIST"] = '/path/to/pdsviyacert.pem'

# Connect to CAS
s = swat.CAS("https://mpmprodvdmm1.ondemand.sas.com/cas-shared-default-http/", 443)

# Print the server status to make sure we’re connected
print(s.serverstatus())
```

The s variable now holds the connection to CAS, which can be used to load data and run CAS actions.

Load a Project Data Sphere Data Set
The Project Data Sphere data sets are automatically available to the platform’s SAS tools, including connections to CAS through Python. First, identify the Unique Dataset ID for the data set you are interested in using. You can find this ID on the Access Data page for the data set. This example will use
the Prostate Cancer DREAM Challenge data sets, which has a Unique Dataset ID of Prostat_na_2006_149.

# List all files in the caslib
res = s.table.fileInfo(
    allFiles=True, # show all files
    caslib="Prostat_na_2006_149") # caslib name is the Unique Dataset ID
print(res)

After listing the files, let’s look for the training data files:

# List all files in the dataFiles_859 subdirectory
res = s.table.fileInfo(
    allFiles=True, # show all files
    caslib="Prostat_na_2006_149", # caslib name is the Unique Dataset ID
    path="dataFiles_859") # the subdirectory with the training data
print(res)

Let’s load one of the tables into CAS:

    # Create an output table to load the file into
core_train = s.CASTable('core_train', replace=True, caslib='CASUSER')
    # Load the table into CAS
s.table.loadTable(
    sourceCaslib="Prostat_na_2006_149",
    casOut=core_train,
    path="dataFiles_859/CoreTable_training.csv")

    # View the first 5 rows
print(core_train.head())

Notice we were able to run the head() function on the CASTable object. The SWAT package mimics much of the API of the Pandas package so that using CAS should feel familiar to current Pandas users.
Run a CAS Action to Perform Data Analysis

Let’s run a frequency analysis on the STUDYID and DEATH columns in this table:

```python
# Load the freqTab action set
s.loadactionset('freqTab')
# Create a two-way cross tabulation table
res = s.freqTab.freqTab(
    table=core_train,
    includeMissing=True,  # Include missing values
    tabulate= [{'vars': {'DEATH', 'STUDYID'}}])
print(res)
```

In this example we used the “freqTab” action set. Many other data mining and machine learning action sets are available on this platform through SAS Viya. A comprehensive list can be found [here](#).

Save a CAS table to CASUSER

Changes to CAS tables are performed in-memory and will not persist once the user’s session ends. You can save CAS tables to a SAS data set on the filesystem:

```python
# Save the CAS table to a .sashdat file in the CASUSER caslib
core_train.table.save(caslib="CASUSER", name="core_train.sashdat", replace=True)
```

Alternatively, you can save your changes to a CSV file:

```python
# Save the CAS table to a CSV file in the CASUSER caslib
core_train.table.save(caslib="CASUSER", name="core_train.csv", replace=True)
```

Finishing Up

Once you are done working with a CAS table and have saved any changes you want to keep, you should drop it from CAS memory to save system resources:

```python
# Drop the table from memory
core_train.table.dropTable()
```

Documentation

[Python Programming for SAS](#)

[Getting Started with SAS® Viya® 3.4 for Python](#)

Additional Videos

[Getting Started with the Python Interface of SAS Viya](#)
Getting Started: API access to SAS® Visual Data Mining and Machine Learning using R

Quick Start

Requirements
To use R with SAS Cloud Analytic Services (CAS), the client machine that runs R must meet the following requirements:

- Use 64-bit Linux or 64-bit Windows.
- Use a 64-bit version of R.
- Use R 3.1.0 or later.
- Install the dplyr, httr, jsonlite, and bindrcpp packages. These packages have additional dependencies that are automatically installed from CRAN when you run install.packages().
- The SAS Scripting Wrapper for Analytics Transfer (SWAT) package is available for download from https://github.com/sassoftware/R-swat/releases. Information for installing is available from that page.
- Create an authinfo file with your Project Data Sphere credentials. See Create an Authinfo File in Client Authentication Using an Authinfo File.

Connect and Start a Session
To enable an R program to work with SAS Cloud Analytic Services, you must establish a connection with the server.

Run the following code sample in your R client:

```r
library(swat)
# Connect to CAS
s <- CAS("https://mpmprodvdml.ondemand.sas.com/cas-shared-default-http/", 443, protocol='https')
# Print the server status to make sure we're connected
print(cas.builtins.serverStatus(s))
```

The `s` variable now holds the connection to CAS, which can be used to load data and run CAS actions.

Load a Project Data Sphere Data Set
The Project Data Sphere data sets are automatically available to the platform’s SAS tools, including connections to CAS through Python. First, identify the Unique Dataset ID for the data set you are interested in using. You can find this ID on the Access Data page for the data set. This example will use the Prostate Cancer DREAM Challenge data sets, which has a Unique Dataset ID of Prostat_na_2006_149.
# List all files in the caslib
res = cas.table.fileInfo(
    s,
    allFiles=TRUE,  # show all files
caslib="Prostat_na_2006_149")  # caslib name is the Unique Dataset ID
print(res)

After listing the files, let’s look for the training data files:

# List all files in the dataFiles_859 subdirectory
res = cas.table.fileInfo(
    s,
    allFiles=TRUE,  # show all files
caslib="Prostat_na_2006_149",  # caslib name is the Unique Dataset ID
    path="dataFiles_859")  # the subdirectory with the training data
print(res)

Let’s load one of the tables into CAS:

# Load the table into CAS
res = cas.table.loadTable(
    s,
    sourceCaslib="Prostat_na_2006_149",
casOut=list(name='core_train', replace=TRUE, caslib='CASUSER'),
    path="dataFiles_859/CoreTable_training.csv")
print(res)

# Create a CASTable to reference the table we just loaded into CAS
core_train = defCasTable(s, 'core_train', caslib='CASUSER')
# View the first 5 rows
head(core_train)
Run a CAS Action to Perform Data Analysis

Let’s run a frequency analysis on the STUDYID and DEATH columns in this table:

```r
# Load the freqTab action set
cas.builtins.loadActionSet(s, 'freqTab')
# Create a two-way cross tabulation table
res = cas.freqTab.freqTab(
    s,
    table=core_train,
    includeMissing=TRUE, # Include missing values
    tabulate=c(vars=c('DEATH', 'STUDYID'))
)
print(res)
```

In this example we used the “freqTab” action set. Many other data mining and machine learning action sets are available on this platform through SAS Viya. A comprehensive list can be found [here](#).

**Save a CAS table to CASUSER**

Changes to CAS tables are performed in-memory and will not persist once the user’s session ends. You can save CAS tables to a SAS data set on the filesystem:

```r
# Save the CAS table to a .sashdat file in the CASUSER caslib
cas.table.save(s, table=core_train, caslib="CASUSER",
name="core_train.sashdat",
replace=TRUE)

Alternatively, you can save your changes to a CSV file:

# Save the CAS table to a CSV file in the CASUSER caslib
cas.table.save(s, table=core_train, caslib="CASUSER",
name="core_train.csv",
replace=TRUE)
```

**Finishing Up**

Once you are done working with a CAS table and have saved any changes you want to keep, you should drop it from CAS memory to save system resources:

```r
# Drop the table from memory
cas.table.dropTable(s, name='core_train', caslib='CASUSER')
```

**Documentation**

[R Programming for SAS Viya](#)
Getting Started with SAS® Viya® 3.4 for Python