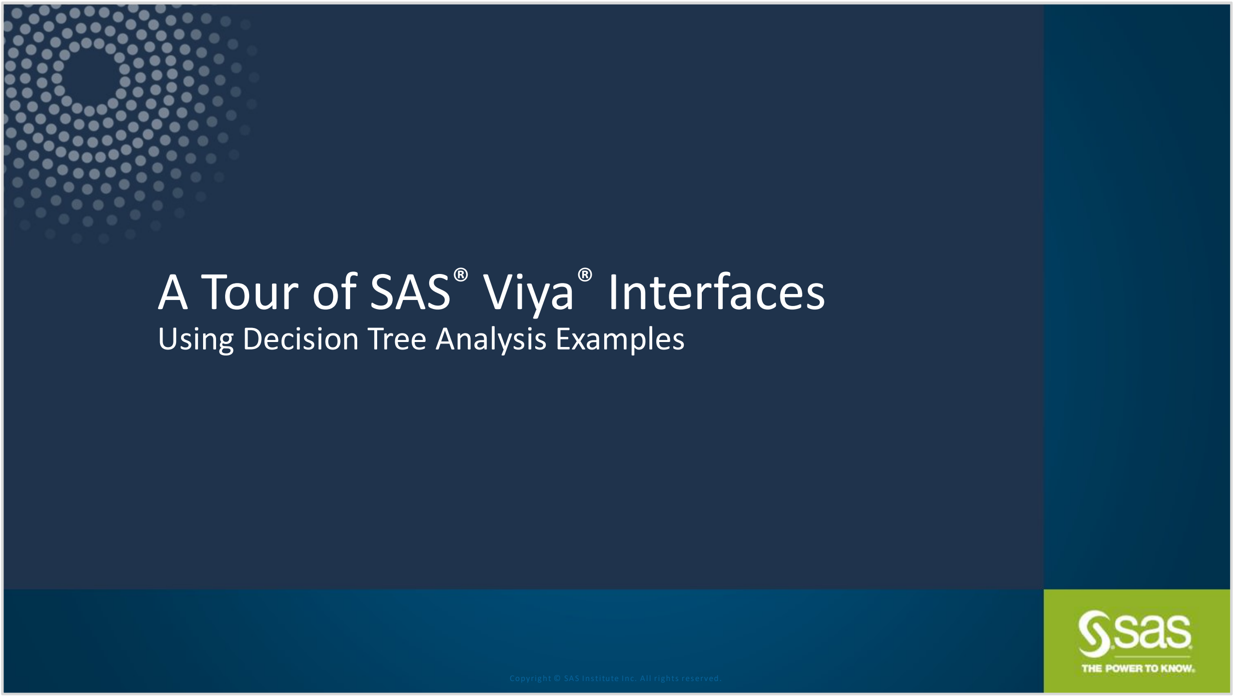
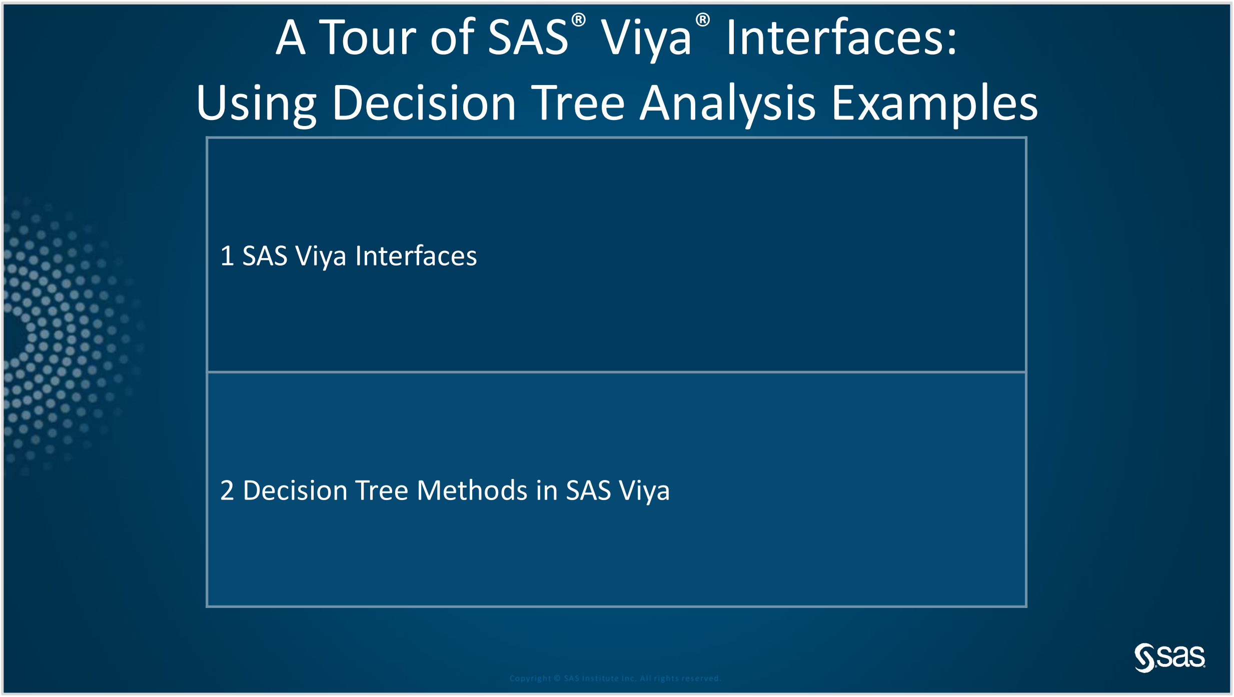


A Tour of SAS® Viya® Interfaces

Using Decision Tree Analysis Examples

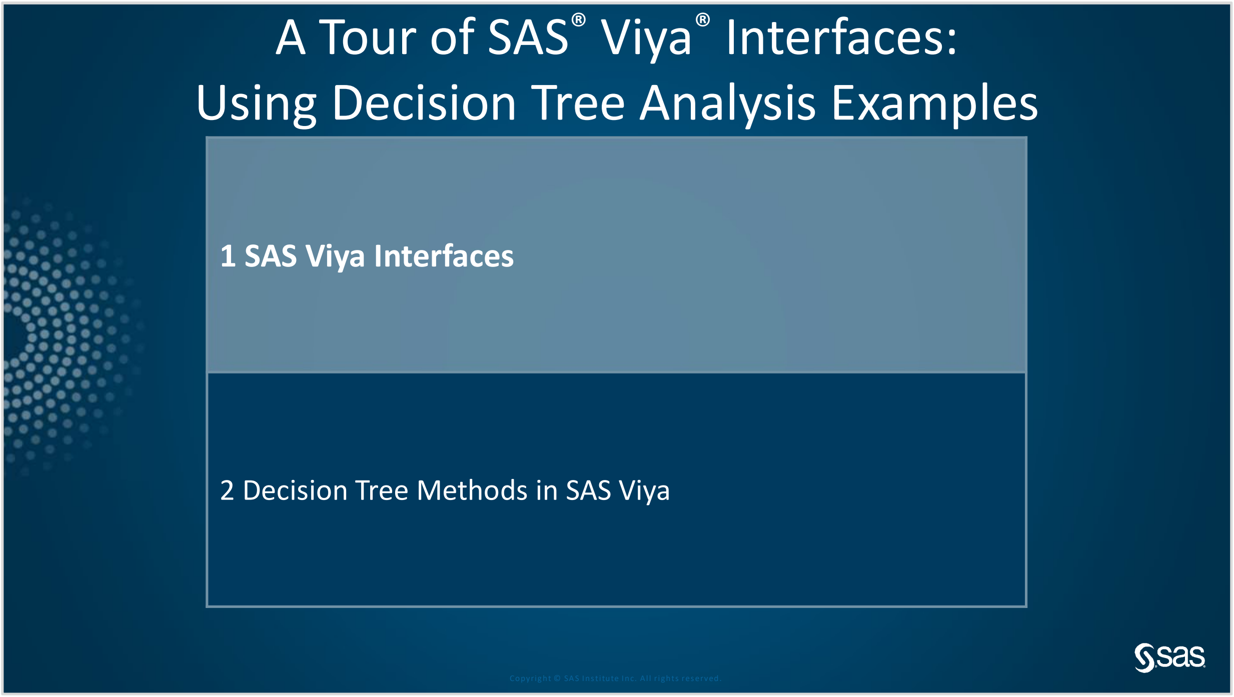


A Tour of SAS® Viya® Interfaces: Using Decision Tree Analysis Examples

1 SAS Viya Interfaces

2 Decision Tree Methods in SAS Viya

Hello everyone, and thanks for coming

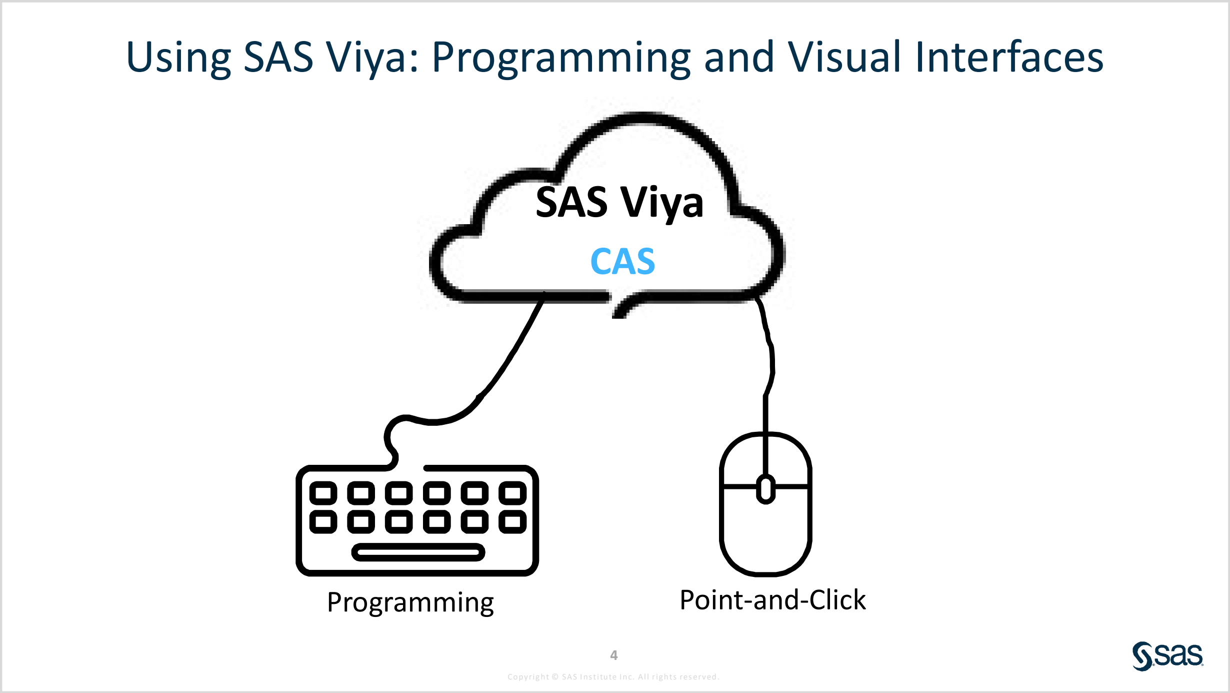


A Tour of SAS® Viya® Interfaces: Using Decision Tree Analysis Examples

1 SAS Viya Interfaces

2 Decision Tree Methods in SAS Viya

I'll begin by briefly overviewing a variety of SAS Viya application interfaces you can use. Then I'll demonstrate how to accomplish Decision Tree modeling using a number of these interfaces.



Using SAS Viya: Programming and Visual Interfaces

SAS Viya

CAS

Programming

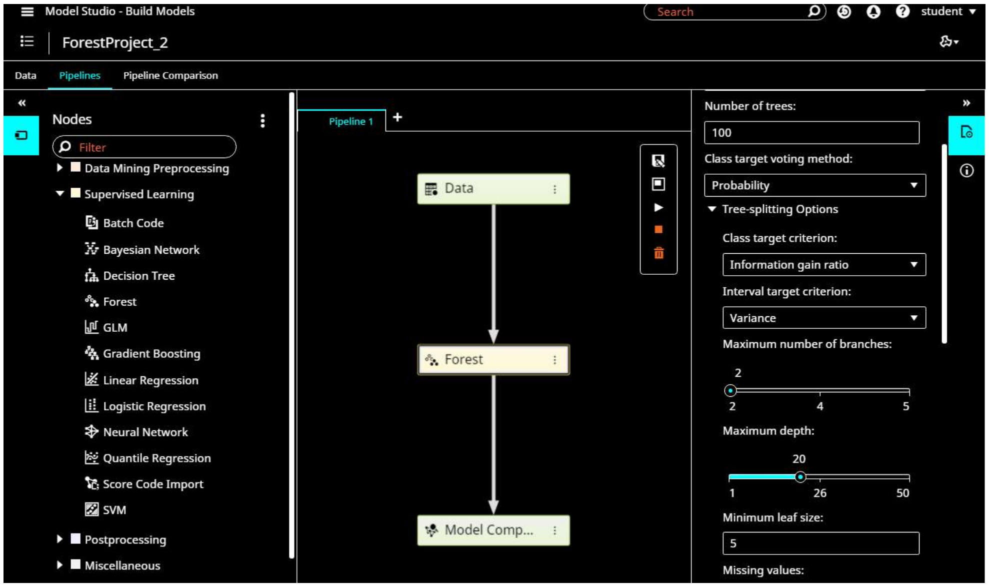
Point-and-Click

4

SAS Viya contains a server process called CAS – or Central Analytics Server. CAS is a distributed multi-machine system for parallel processing of distributed data. This architecture allows you to perform complex computations on large volumes of data quickly.

But to access and direct processing on the CAS server you need to use some kind of client application interface. Both programming and point-and-click application interfaces are available.

Point-and-Click Decision Trees with Model Studio



Number of trees: 100

Class target voting method: Probability

Tree-splitting Options

Class target criterion: Information gain ratio

Interval target criterion: Variance

Maximum number of branches: 2 (2, 4, 5)

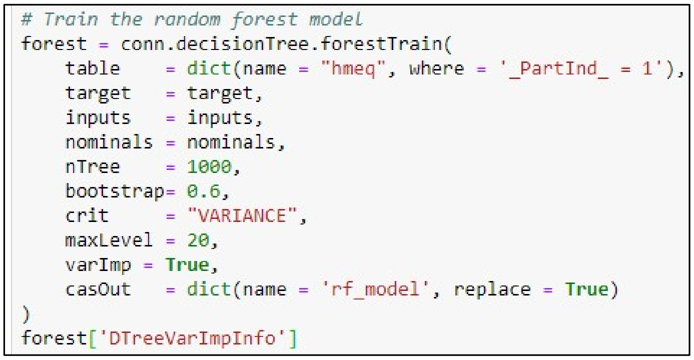
Maximum depth: 20 (1, 26, 50)

Minimum leaf size: 5

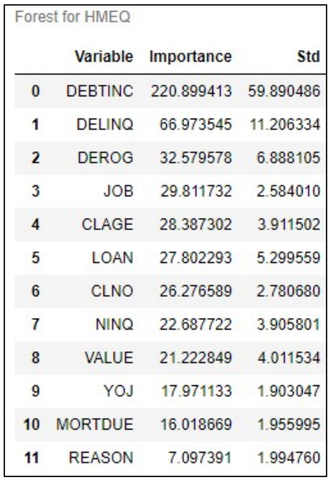
Missing values:

One of the point and click applications I'll demonstrate is the web application interface of Model Studio. I'll be showing you 2 other point-and-click interfaces as well.

Programming Decision Trees with Python and SAS SWAT[1]



```
# Train the random forest model
forest = conn.decisionTree.forestTrain(
    table     = dict(name = "hmeq", where = '_PartInd_ = 1'),
    target    = target,
    inputs    = inputs,
    nominals = nominals,
    nTree     = 1000,
    bootstrap= 0.6,
    crit      = "VARIANCE",
    maxLevel = 20,
    varImp = True,
    casOut    = dict(name = 'rf_model', replace = True)
)
forest['DTreeVarImpInfo']
```

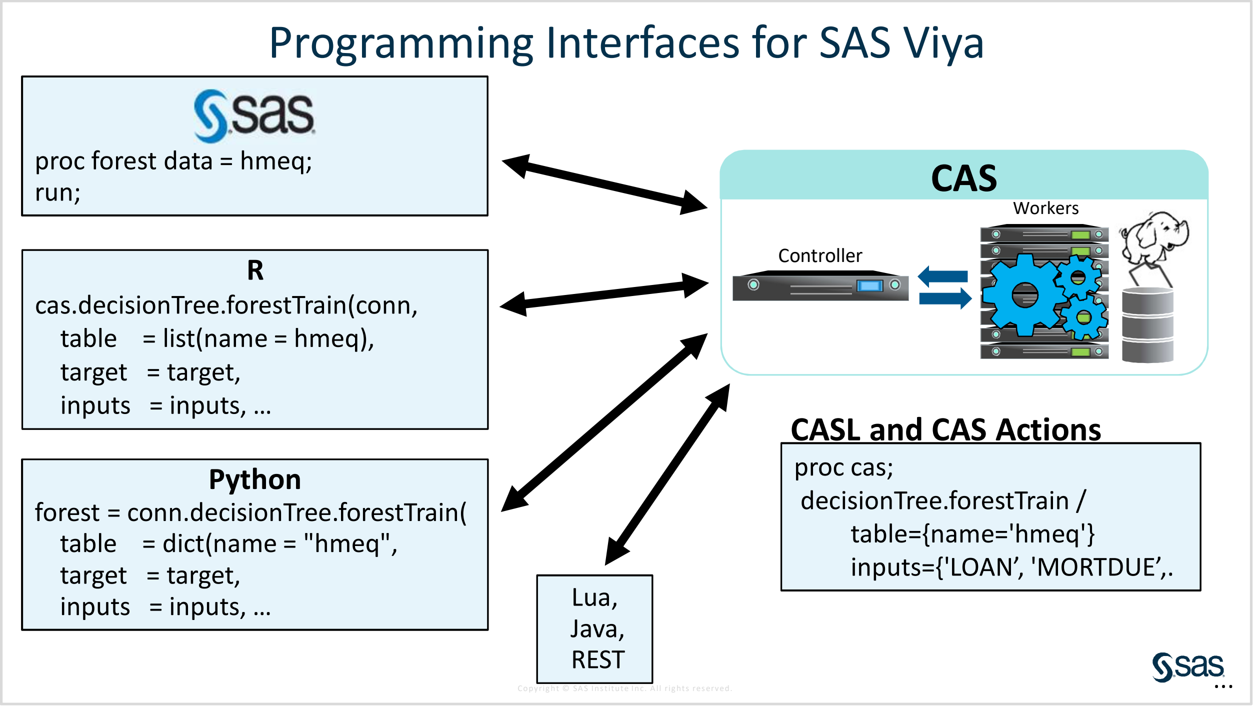


Forest for HMEQ

	Variable	Importance	Std
0	DEBTINC	220.899413	59.890486
1	DELINQ	66.973545	11.206334
2	DEROG	32.579578	6.888105
3	JOB	29.811732	2.584010
4	CLAGE	28.387302	3.911502
5	LOAN	27.802293	5.299559
6	CLNO	26.276589	2.780680
7	NINQ	22.687722	3.905801
8	VALUE	21.222849	4.011534
9	YOJ	17.971133	1.903047
10	MORTDUE	16.018669	1.955995
11	REASON	7.097391	1.994760

[1] SAS SWAT is the SAS Scripting Wrapper for Analytics Transfer

And here is a snippet of code from a Python program I'll demonstrate. SAS provides a downloadable set of python methods for CAS that you can use as part of your python code to connect to SAS Viya, submit processes that execute in Viya, and view and download results for further processing within your python client application.



Programming Interfaces for SAS Viya

SAS

```
proc forest data = hmeq;
run;
```

R

```
cas.decisionTree.forestTrain(conn,
    table    = list(name = hmeq),
    target   = target,
    inputs   = inputs, …
```

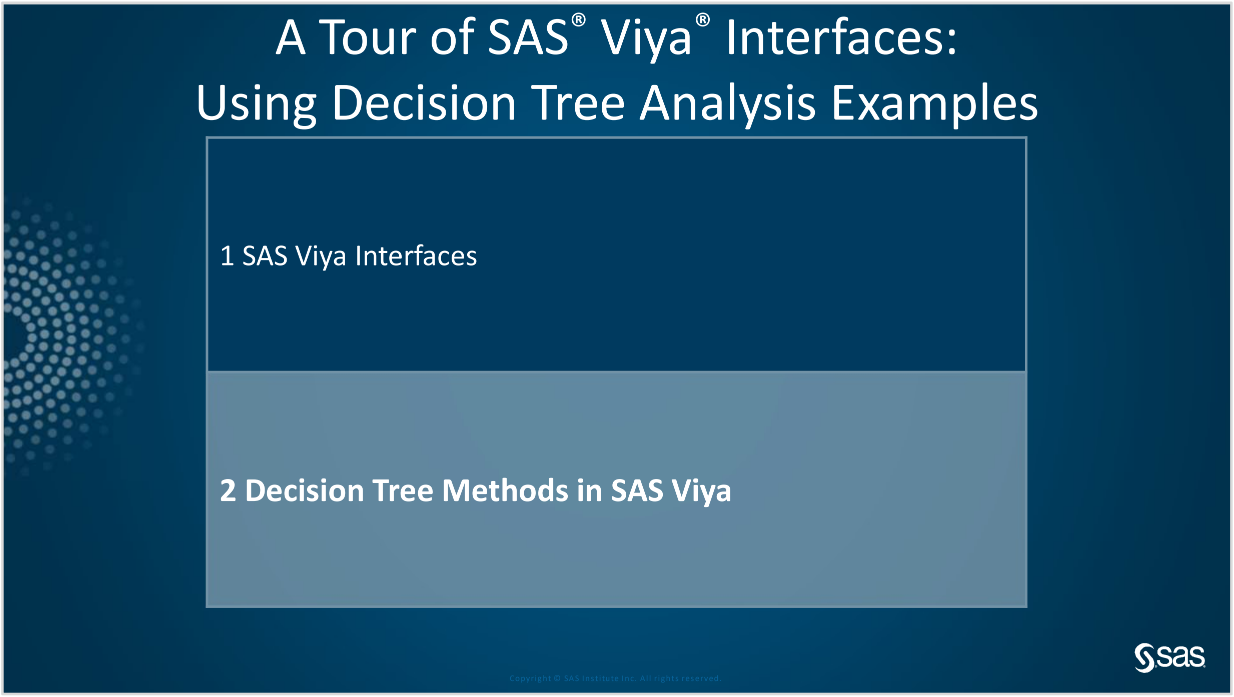
Python

```
forest = conn.decisionTree.forestTrain(
    table    = dict(name = "hmeq",
    target   = target,
    inputs   = inputs, …
```

CASL and CAS Actions

```
proc cas;
 decisionTree.forestTrain /
       table={name='hmeq'}
       inputs={'LOAN’, 'MORTDUE’,.
```

Programming interfaces for SAS Viya include SAS procedures (and the DATA step), Python, Lua, Java, and a REST interface that uses HTTP protocol. No matter which client interface you use, including the point-and-click ones, the requests you specify are converted into a new SAS language called CASL language which includes CAS actions. You can also use CASL directly. CASL has an object oriented syntax similar to python, R and other such languages.

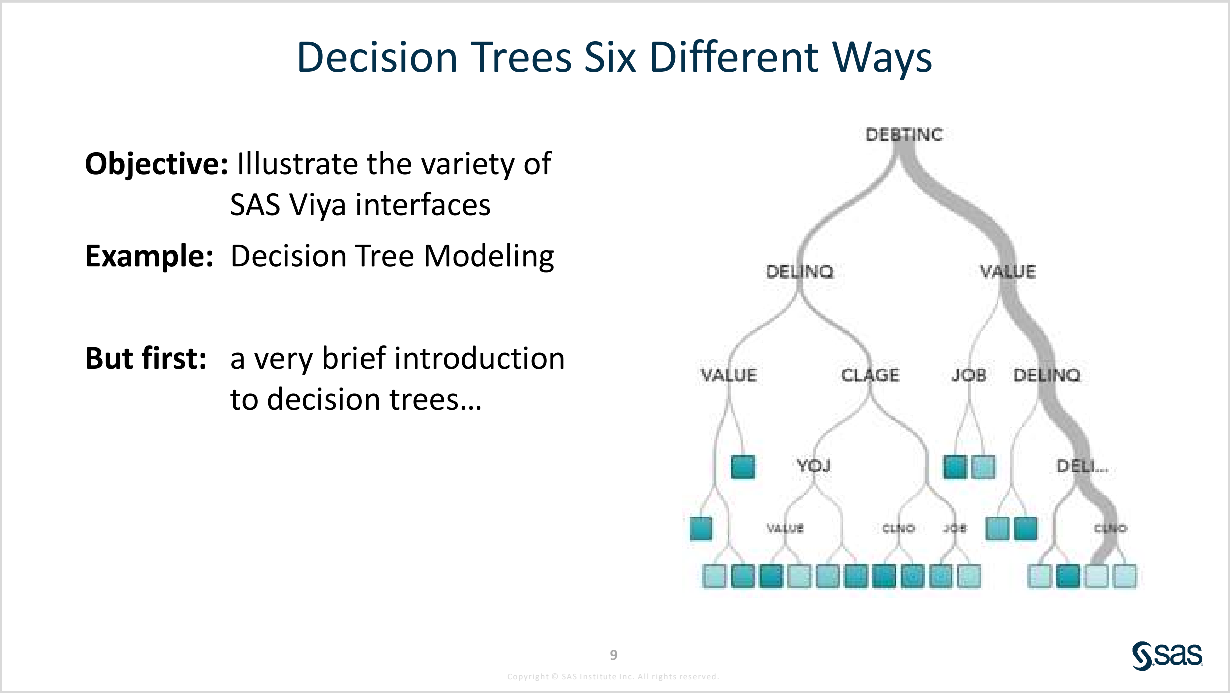


A Tour of SAS® Viya® Interfaces: Using Decision Tree Analysis Examples

1 SAS Viya Interfaces

2 Decision Tree Methods in SAS Viya

Next up I want to describe decision trees for a couple of minutes and then demonstrate decision tree modelling with a number of SAS Viya interfaces.



Decision Trees Six Different Ways

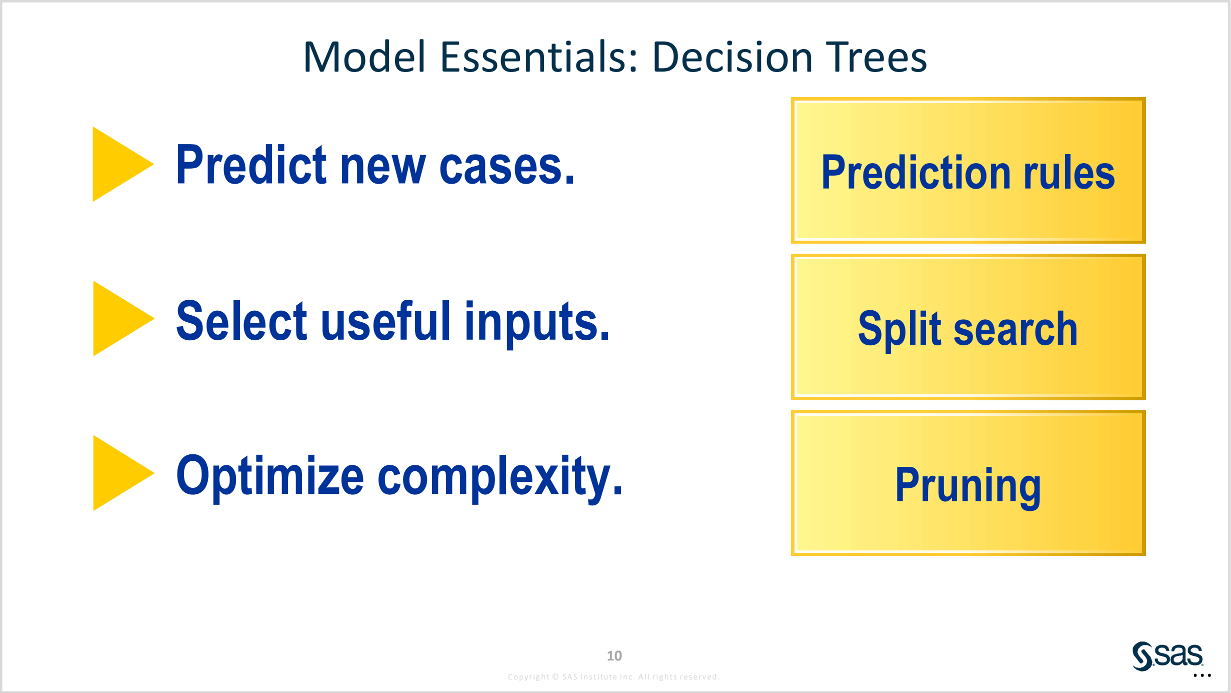
Objective: Illustrate the variety of SAS Viya interfaces

Example: Decision Tree Modeling

But first: a very brief introduction to decision trees…

9

Since I am using it as my example, I'll first give you a whilrwind overview of what decicion tree modeling is. If you are new to decision trees you are not likely to be able to take in the details, but please don't concern yourself with that. Just keep in mind that the main objective is to give you a chance to see the wide choice of application interfaces in SAS Viya and this is simply one example objective. SAS Viya has broad capabilities.



Model Essentials: Decision Trees

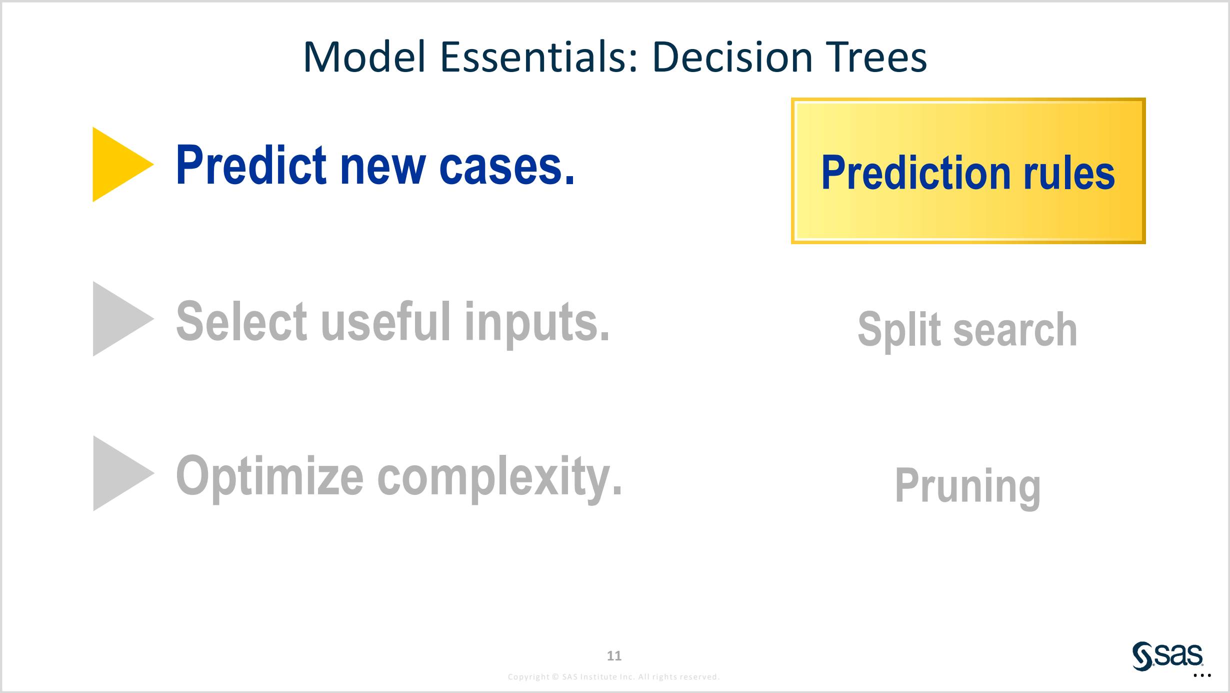
- Predict new cases.
- Select useful inputs.
- Optimize complexity.

Prediction rules

Split search

Pruning

With statistical modelling in general, the objective is to predict new cases or outcomes based on measurable inputs. Statistical models are created using past data with known outcomes. But if you create a complex model to maximally fit the past data, it usually does not predict new cases as well as a simpler model because the extra complexity is explaining random effects in the original data. So the model creation process usually includes techniques to remove some of the complexity so that the model generalizes better to new data.

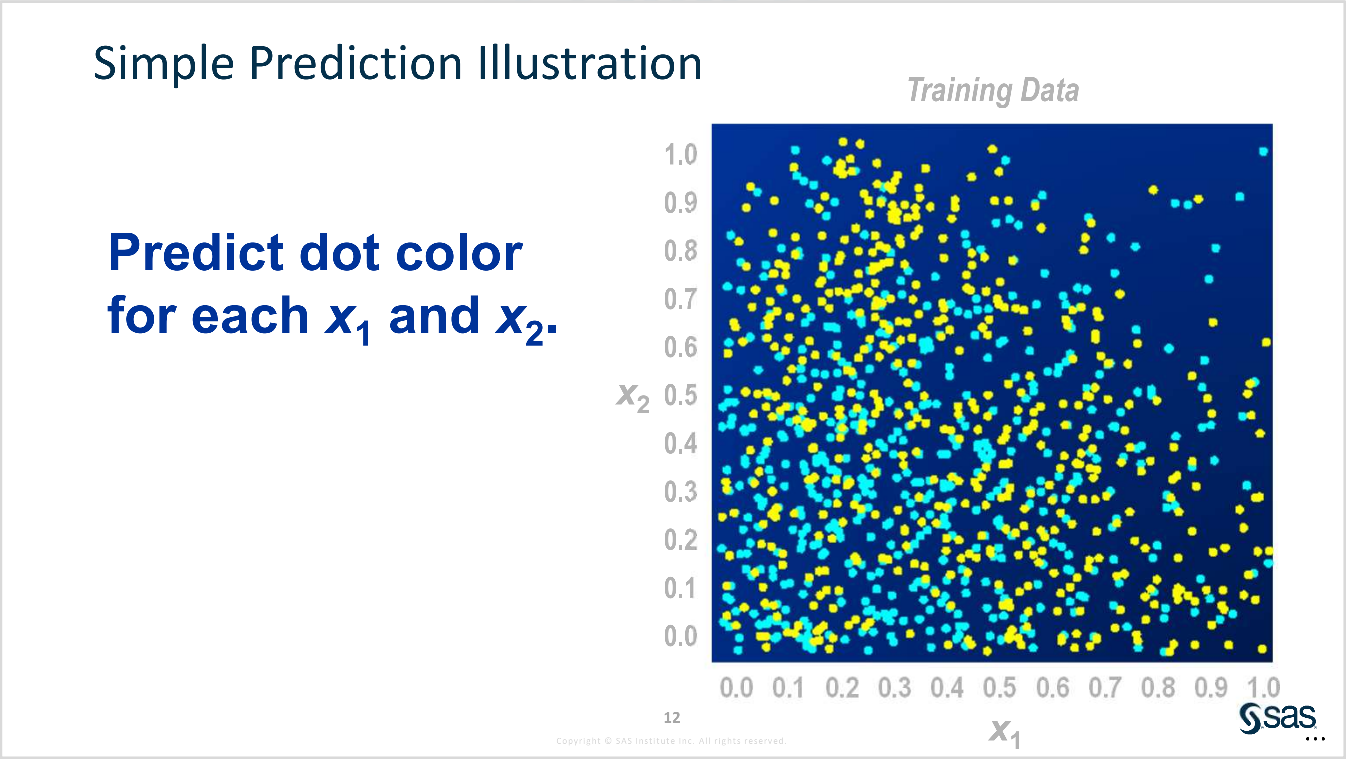


Model Essentials: Decision Trees

- Predict new cases. — Prediction rules
- Select useful inputs. — Split search
- Optimize complexity. — Pruning

11

First, lets look at how decision tree models, once they have been created, are used to predict cases.



Simple Prediction Illustration

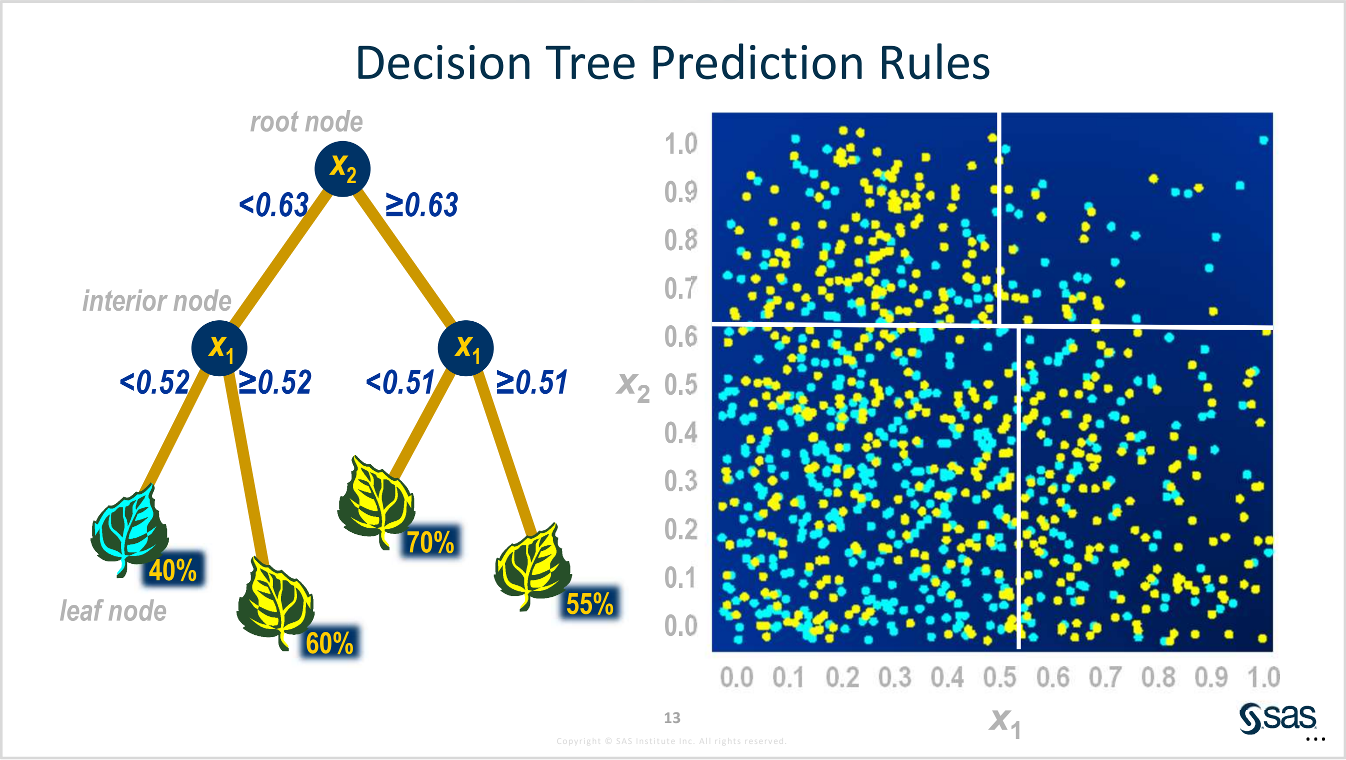
Predict dot color for each x_1 and x_2.

12

...

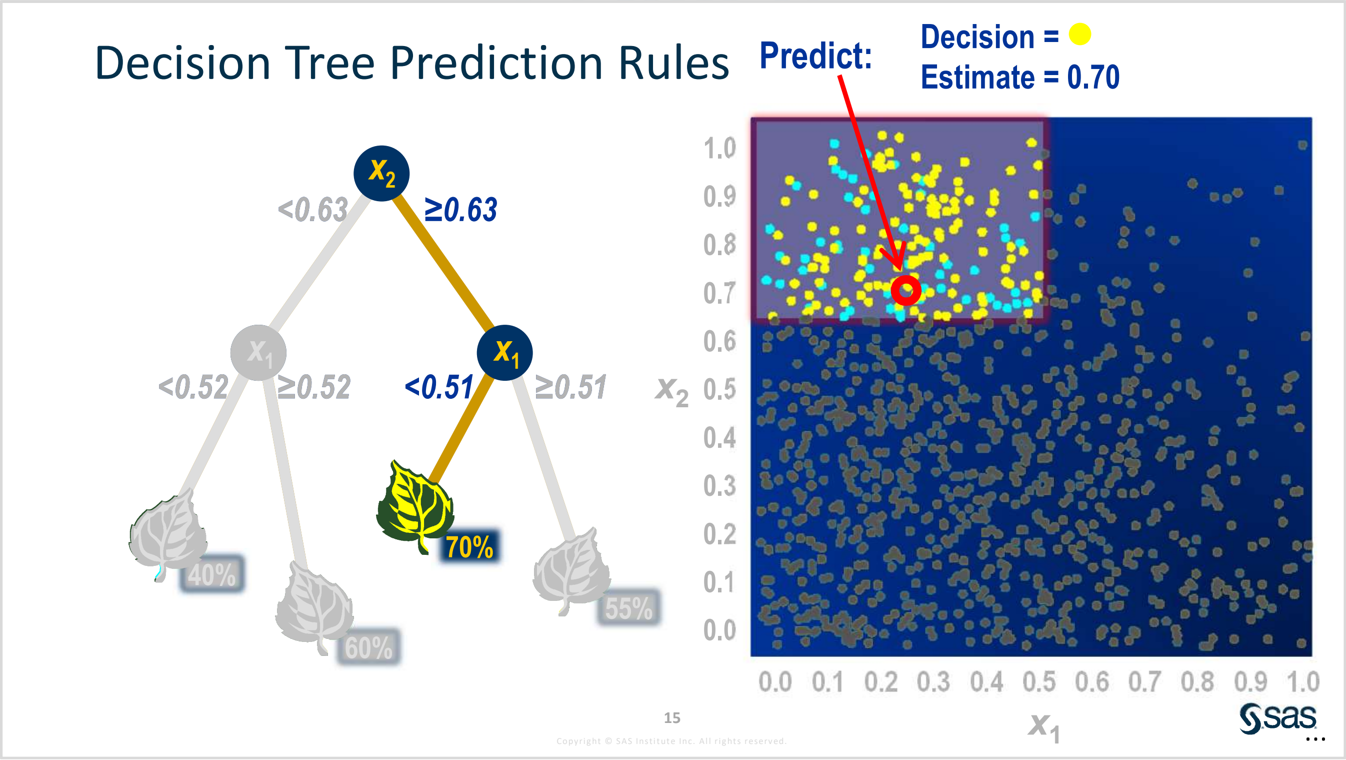
Consider a data set with two input variables and a binary outcome that we wish to predict based on the values of x1 and x2. The binary target outcome is represented by two colors; yellow and blue.

For a more concrete example consider each dot as a bank loan to a customer and 'yellow' indicated they defaulted on the loan and blue represents not defaulting on the loan. X1 might be the debt to income ratio of the person who took the loan and X2 might represent annual income. And we want to predict how debt to income ratio and annual income of a customer affect loan defaults. In real decision tree models you typically have a large number of predictor variables. Two is enough to illustrate how it works.



Decision Tree Prediction Rules

To predict outcome, decision tree models create rules that are applied in a stepwise order that can be represented like an upside down tree as shown here on the left. The rules create separate areas in the plot that maximally concentrate yellow outcomes in some areas and blue outcomes in other areas. Here the first rule splits our plot into 2 areas along the X2 axis (one area above x2=0.63 and one area below x2=0.63). A second rule is then applied to each of those two areas that splits those areas further based on the value of X1. This leaves us with four different areas in the grid, each with a different proportion of each outcome. The proportions of each outcome in each area become our prediction of the likelihood of each outcome based on the values of x1 and x2.



Decision Tree Prediction Rules

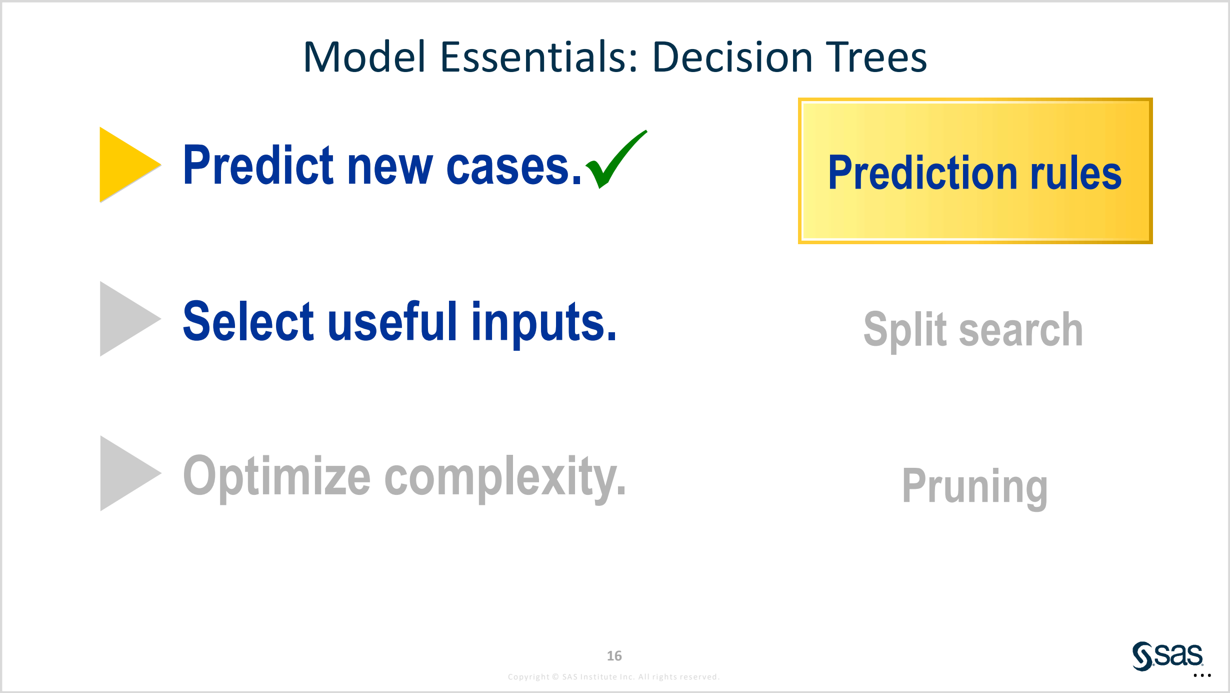
Predict:

Decision = ●

Estimate = 0.70

15

For example, for a new observation with values of x1 and x2 at the point indicated by the red circle in the plot, the decision tree model prodicts the color has a 70% chance of being yellow.

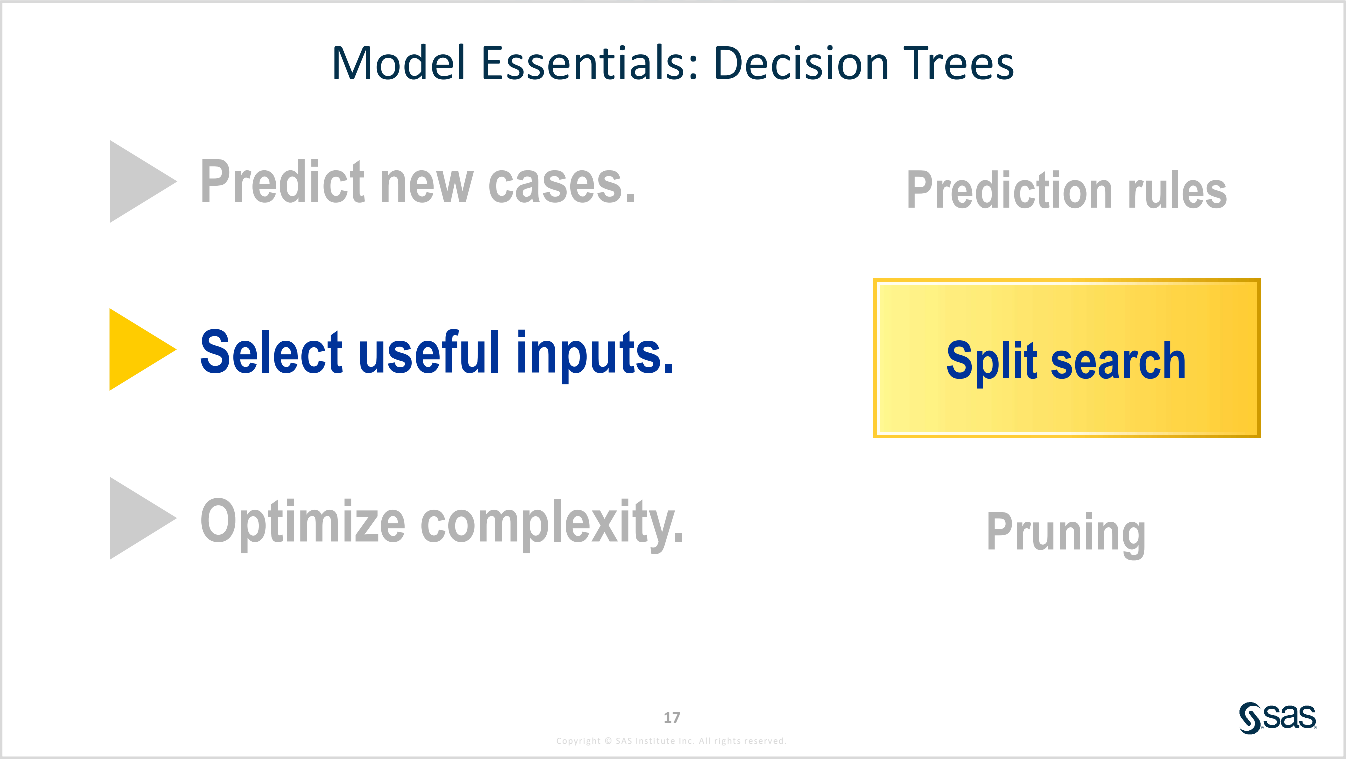


Model Essentials: Decision Trees

- Predict new cases. ✓ — Prediction rules
- Select useful inputs. — Split search
- Optimize complexity. — Pruning

16

So how are these prediction rules created for decision tree models?

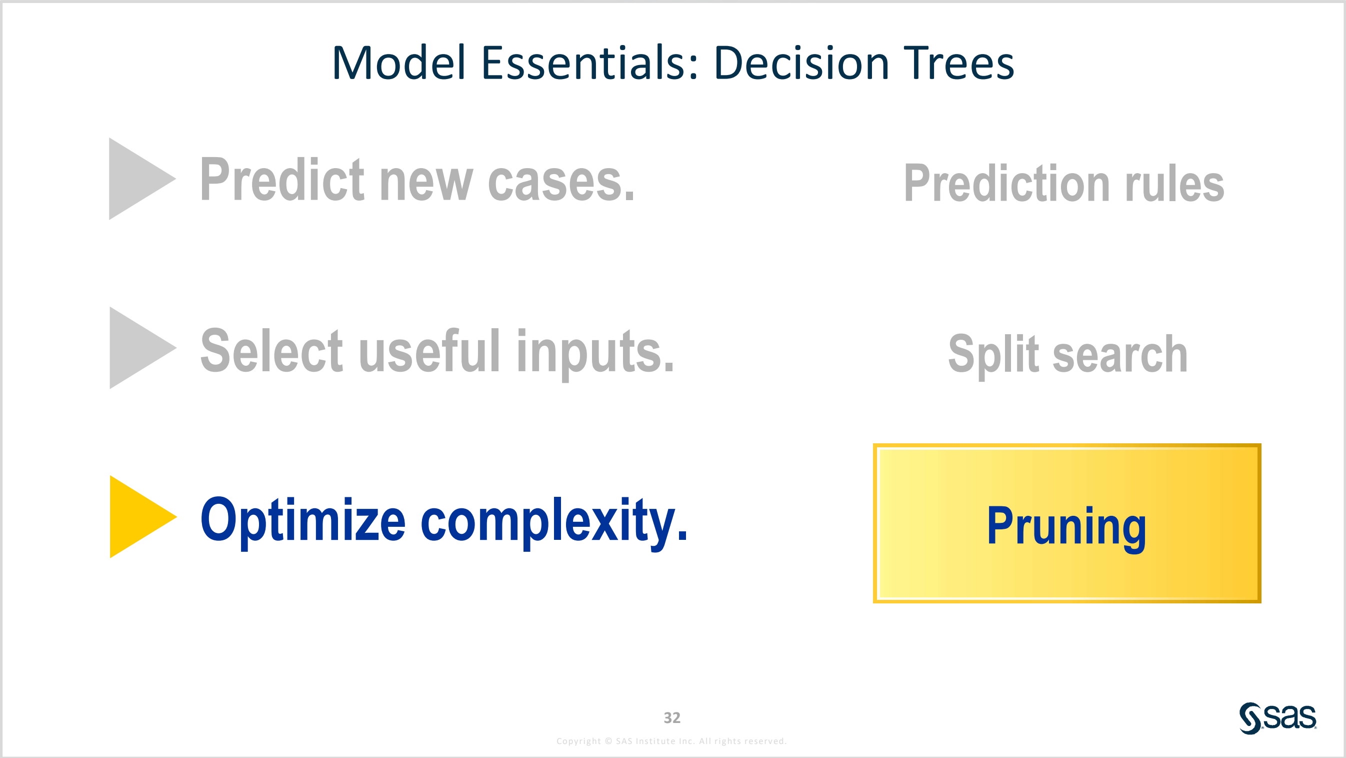


Model Essentials: Decision Trees

Predict new cases.	Prediction rules
Select useful inputs.	**Split search**
Optimize complexity.	Pruning

17

To select useful inputs, and come up with the best hierarchical set of rules, trees use a *split-search* algorithm. I don't have time today to overview how this works, so I have hidden the next set of about 15 slides. If you are interested, though, keep a look out for the SAS Communities posting when the recording for this talk is available. In that posting you will have access to this presentation material including these slides I am skipping and the slide notes.

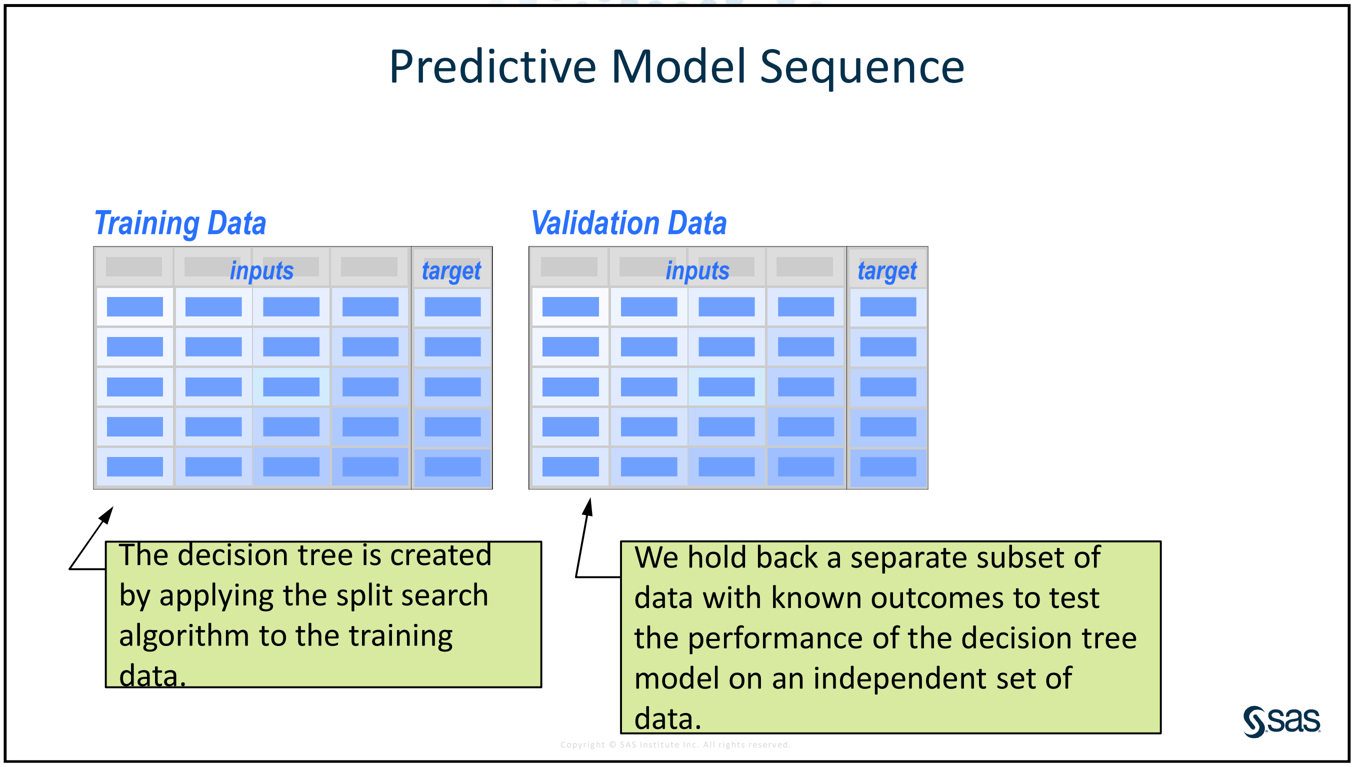


Model Essentials: Decision Trees

- Predict new cases. — Prediction rules
- Select useful inputs. — Split search
- **Optimize complexity.** — **Pruning**

32

When the split-search algorithm produces a model based on the training data, the result may have many branches that represents the most complicated model possible based on the chosen algorithm parameters. Usually this model is too specific due to random effects of the training data and does not generalize well on new observations. To avoid potential overfitting, many predictive modeling procedures offer some mechanism for adjusting model complexity to remove random effects of the training data. For decision trees, this process is known as *pruning*.



Predictive Model Sequence

Training Data

inputs *target*

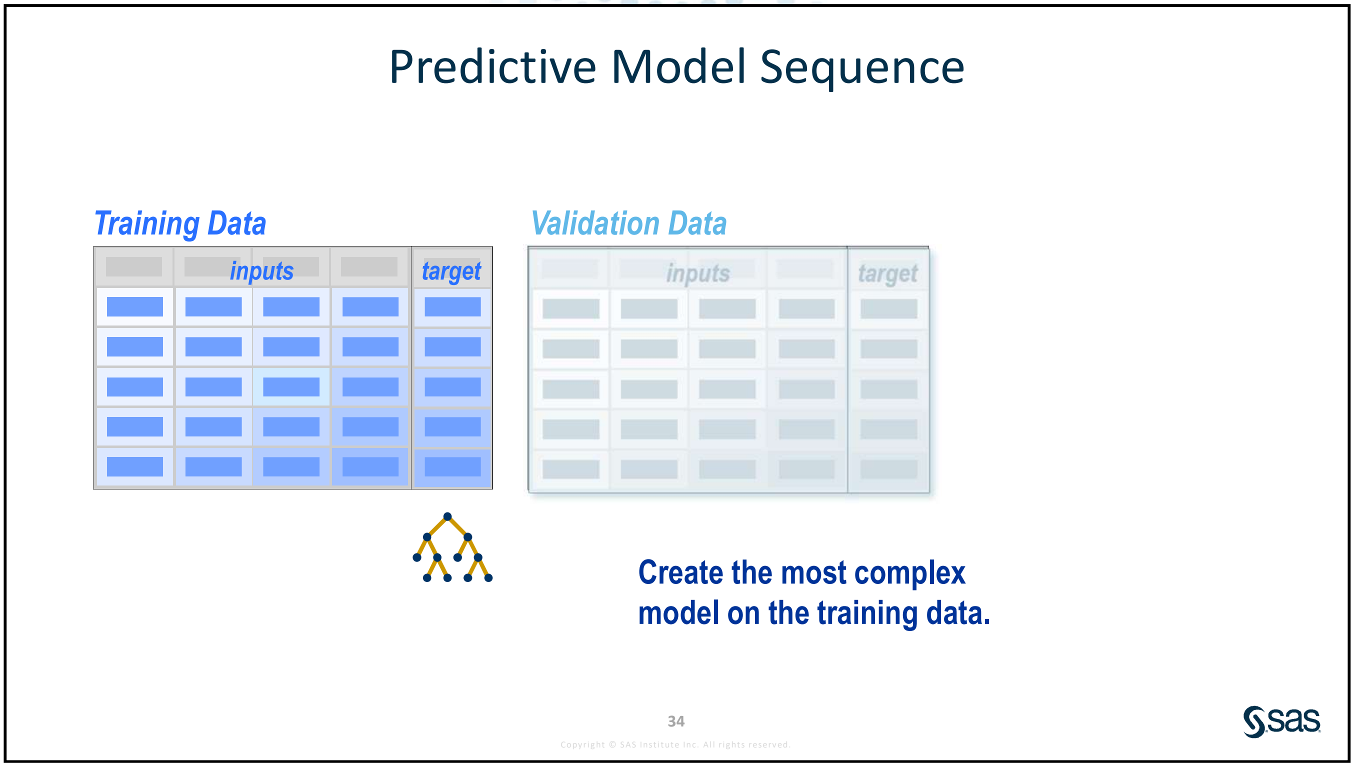
Validation Data

inputs *target*

The decision tree is created by applying the split search algorithm to the training data.

We hold back a separate subset of data with known outcomes to test the performance of the decision tree model on an independent set of data.

So, the decision tree is created by applying the split search algorithm to Training Data.
But, we hold back a separate subset of data with known outcomes to test the performance of the decision tree model on an independent set of data



Predictive Model Sequence

Training Data

inputs

target

Validation Data

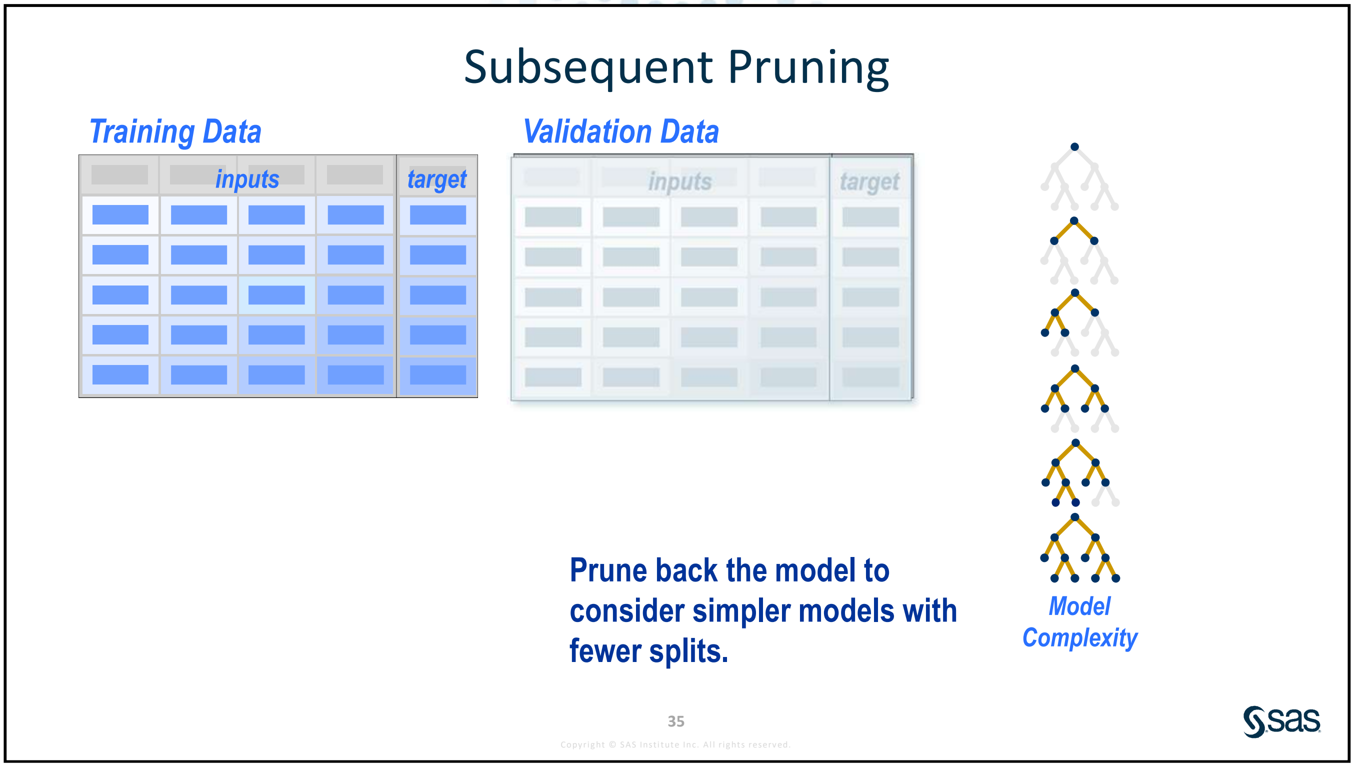
inputs

target

Create the most complex model on the training data.

34

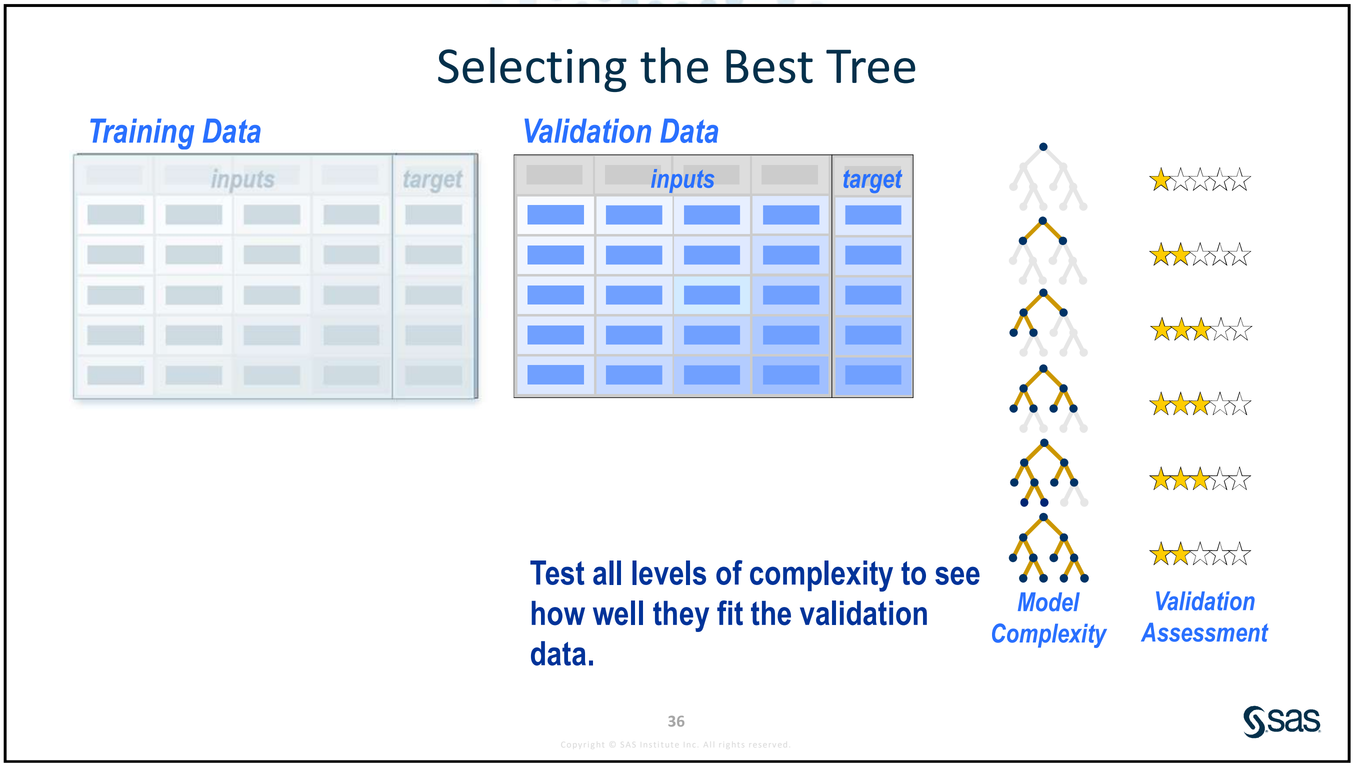
So we create the most complex model on the training data.



Subsequent Pruning

Prune back the model to consider simpler models with fewer splits.

But we prune it back in several stages to create simpler models with less splits.

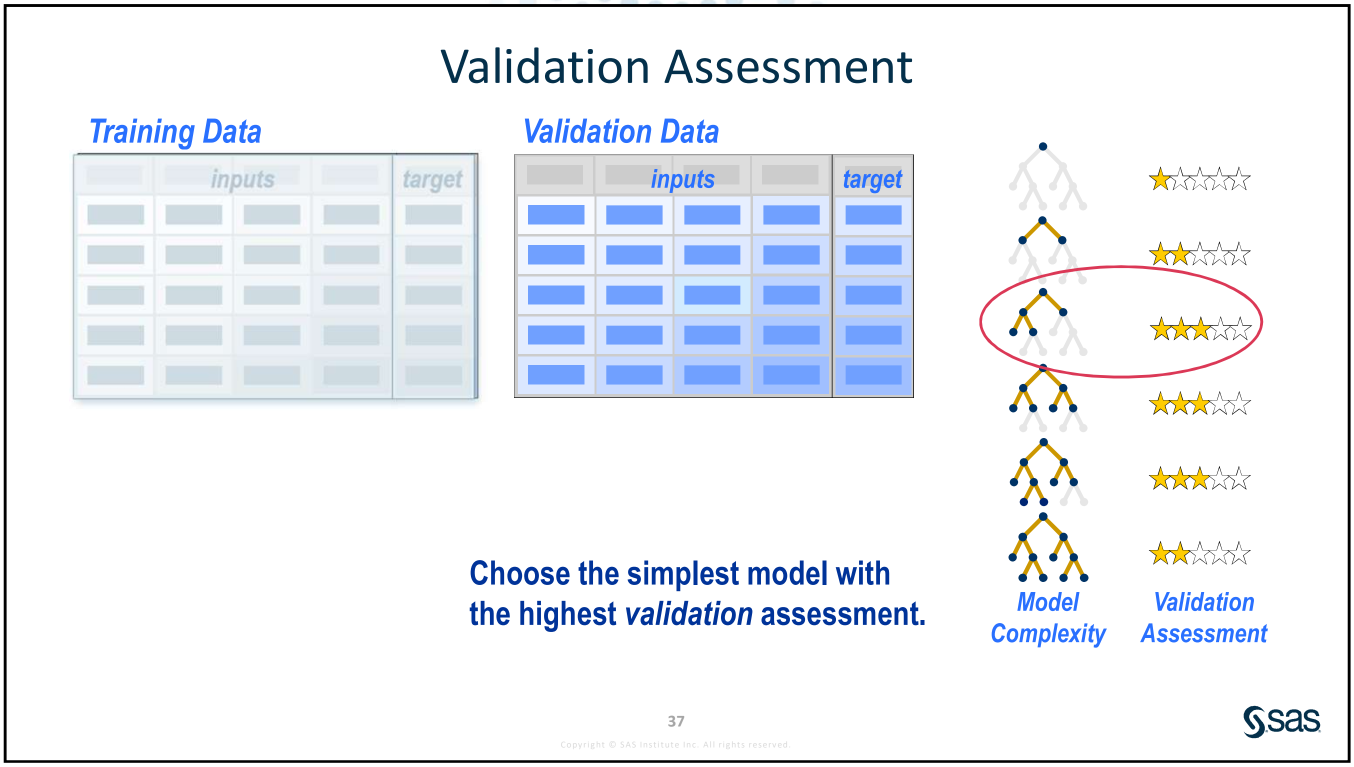


Selecting the Best Tree

Test all levels of complexity to see how well they fit the validation data.

36

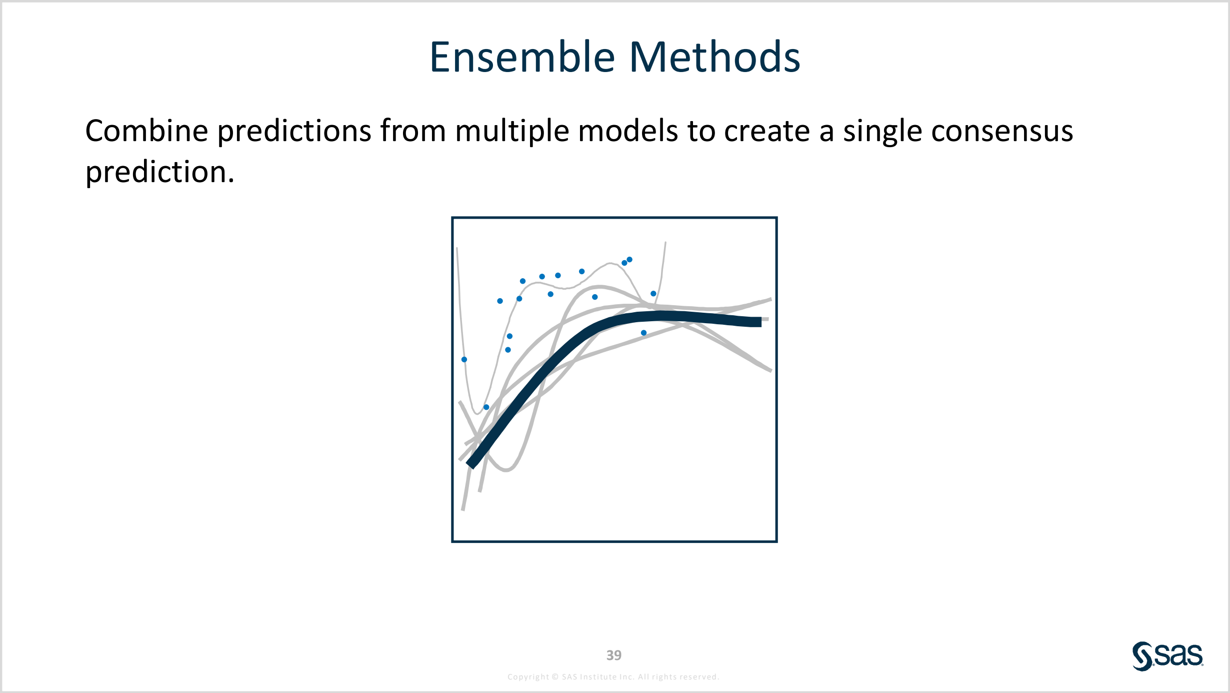
Then we compare how well each level of model complexity fits the validation data that we held back.



Validation Assessment

Choose the simplest model with the highest *validation* assessment.

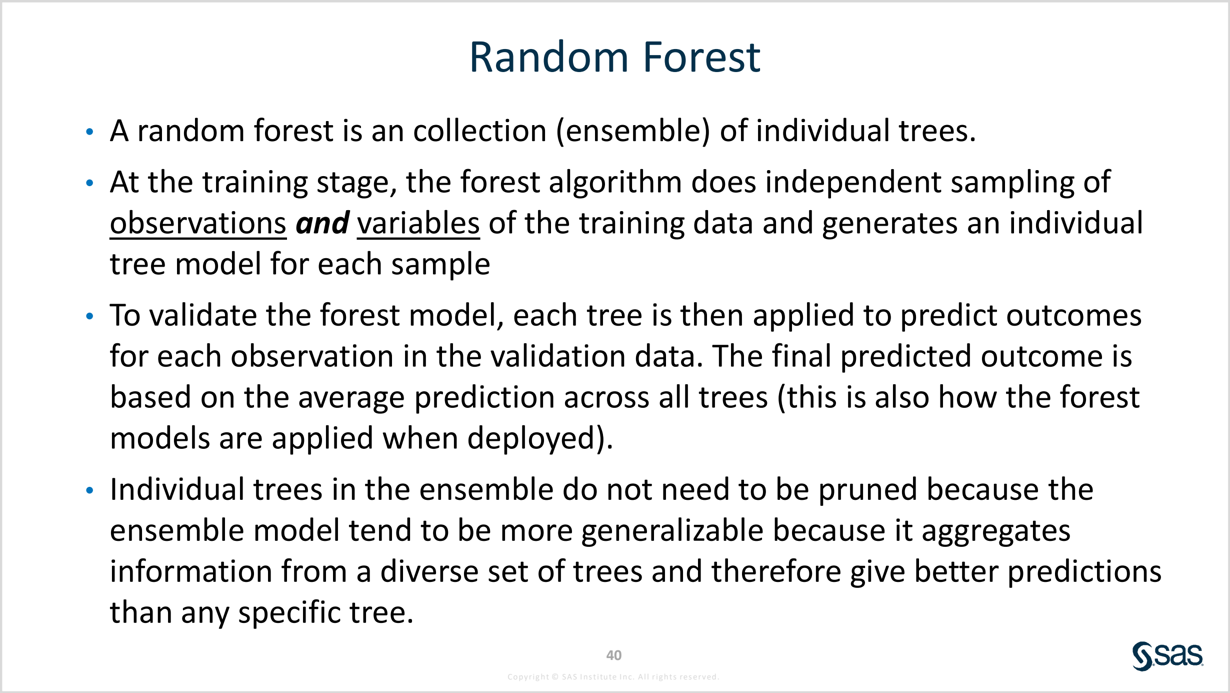
The simplest model with the best fit to the validation data is selected.



Ensemble Methods

Combine predictions from multiple models to create a single consensus prediction.

An alternative approach to creating a single tree and then pruning it is to use an ensemble methods. Ensemble methods create a new model by combining the predictions from multiple models. The commonly observed advantage of ensemble models is that the combined model is better than the individual models that compose it.



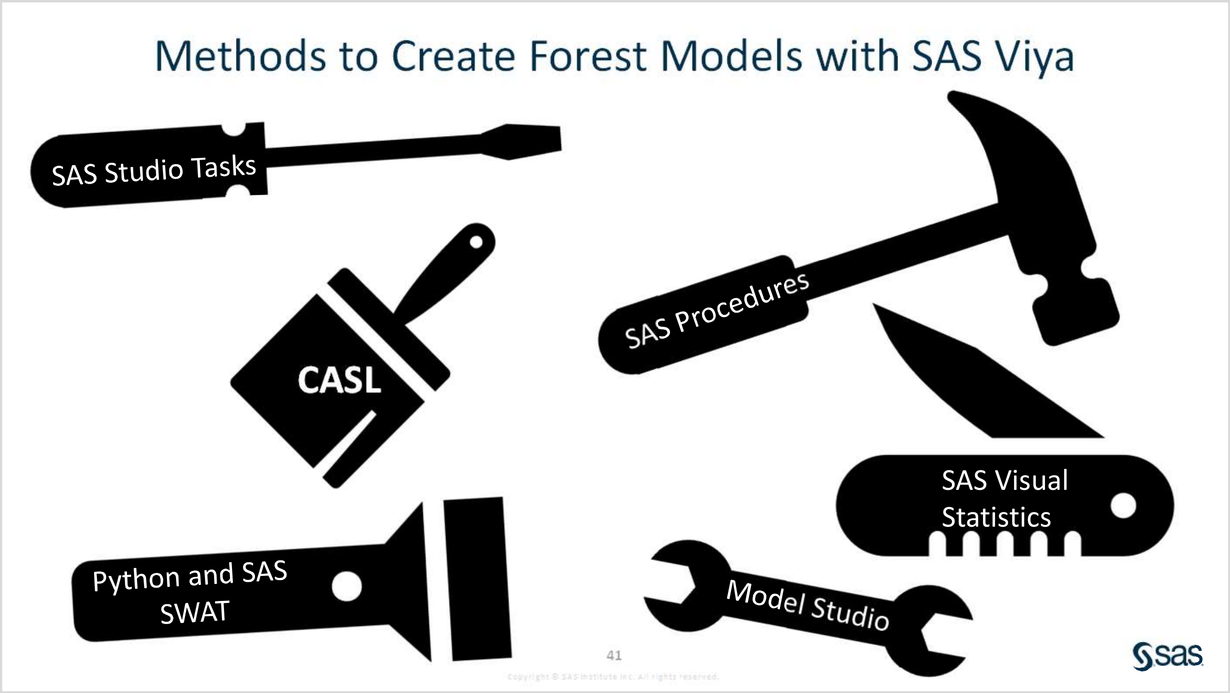
Random Forest

- A random forest is an collection (ensemble) of individual trees.
- At the training stage, the forest algorithm does independent sampling of observations ***and*** variables of the training data and generates an individual tree model for each sample
- To validate the forest model, each tree is then applied to predict outcomes for each observation in the validation data. The final predicted outcome is based on the average prediction across all trees (this is also how the forest models are applied when deployed).
- Individual trees in the ensemble do not need to be pruned because the ensemble model tend to be more generalizable because it aggregates information from a diverse set of trees and therefore give better predictions than any specific tree.

40

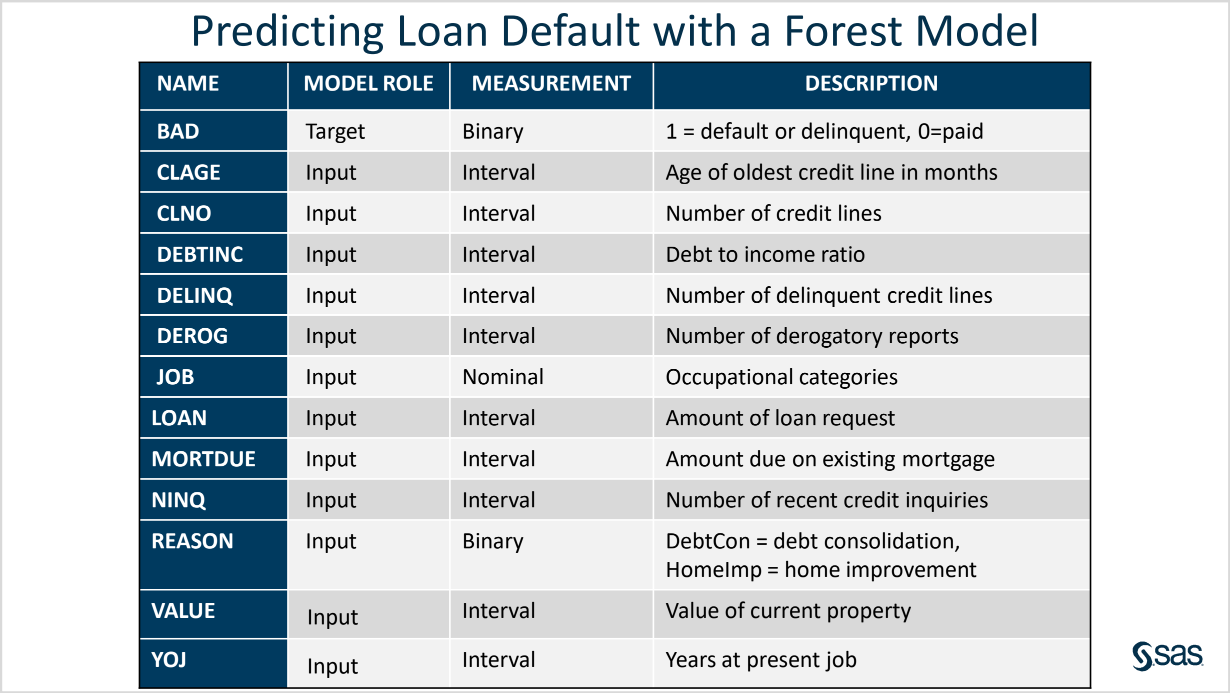
Ensemble models for decision trees are referred to as Random Forests.

- A random forest model is an collection of individual trees.
- Each tree in the collection is created at the training stage from an independent random sample of a subset of observations ***and*** variables from the training data
- To validate the forest model, each tree is then applied to predict outcomes for each observation in the validation data. Tthe final predicted outcome is based on the average prediction across all trees (this is also how the forest models are applied when deployed).
- Individual Individual trees in the ensemble do not need to be pruned because the ensemble model tend to be more generalizable because it aggregates information from a diverse set of trees and therefore give better predictions than any specific tree.



Methods to Create Forest Models with SAS Viya

So I will actually demonstrate how to build ensemble forest models using each of the application interfaces listed here.



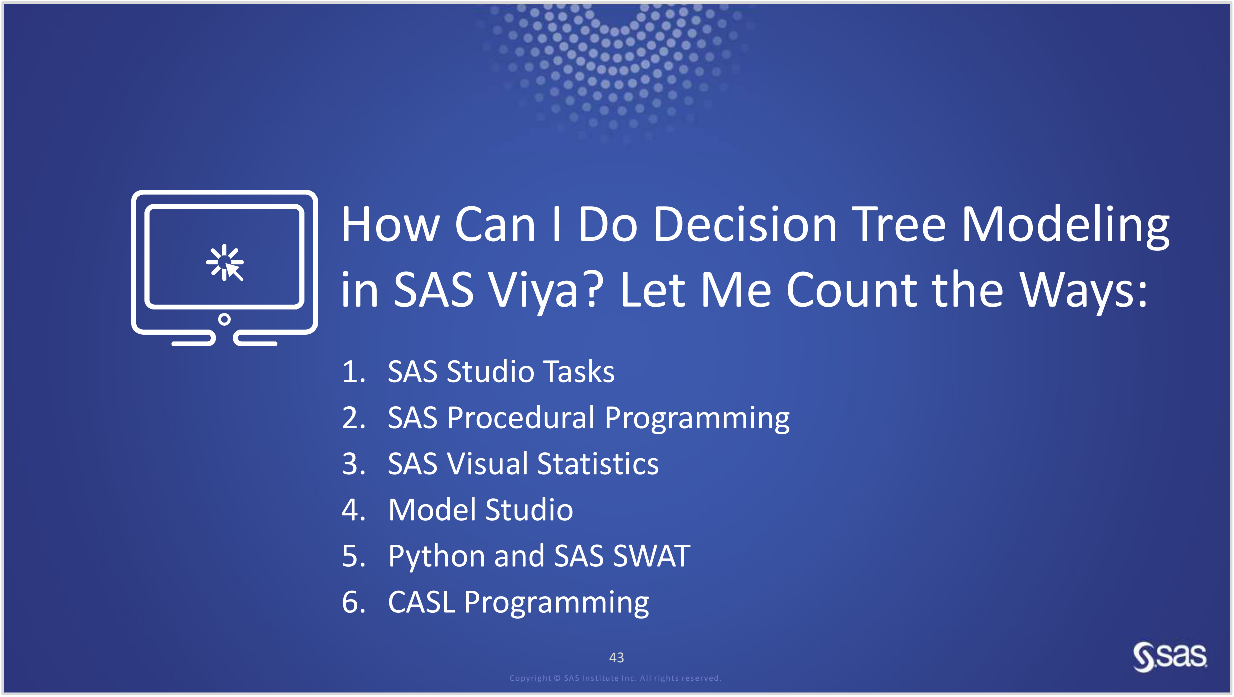
Predicting Loan Default with a Forest Model

NAME	MODEL ROLE	MEASUREMENT	DESCRIPTION
BAD	Target	Binary	1 = default or delinquent, 0=paid
CLAGE	Input	Interval	Age of oldest credit line in months
CLNO	Input	Interval	Number of credit lines
DEBTINC	Input	Interval	Debt to income ratio
DELINQ	Input	Interval	Number of delinquent credit lines
DEROG	Input	Interval	Number of derogatory reports
JOB	Input	Nominal	Occupational categories
LOAN	Input	Interval	Amount of loan request
MORTDUE	Input	Interval	Amount due on existing mortgage
NINQ	Input	Interval	Number of recent credit inquiries

REASON	Input	Binary	DebtCon = debt consolidation,
			HomeImp = home improvement

VALUE	Input	Interval	Value of current property
YOJ	Input	Interval	Years at present job

This is a description of the variables in the dataset I'll use to build the forest model. This is a Home Equity Line of Credit dataset, named HMEQ for short. The table contains one row per loan. For each loan, BAD = 1 for loans that are in default or delinquent, and BAD = 0 if the loan is in good status. The value of BAD will be the binary target variable we will model with Forest Models, using the rest of the variables as input variables.



How Can I Do Decision Tree Modeling in SAS Viya? Let Me Count the Ways:

1. SAS Studio Tasks
2. SAS Procedural Programming
3. SAS Visual Statistics
4. Model Studio
5. Python and SAS SWAT
6. CASL Programming