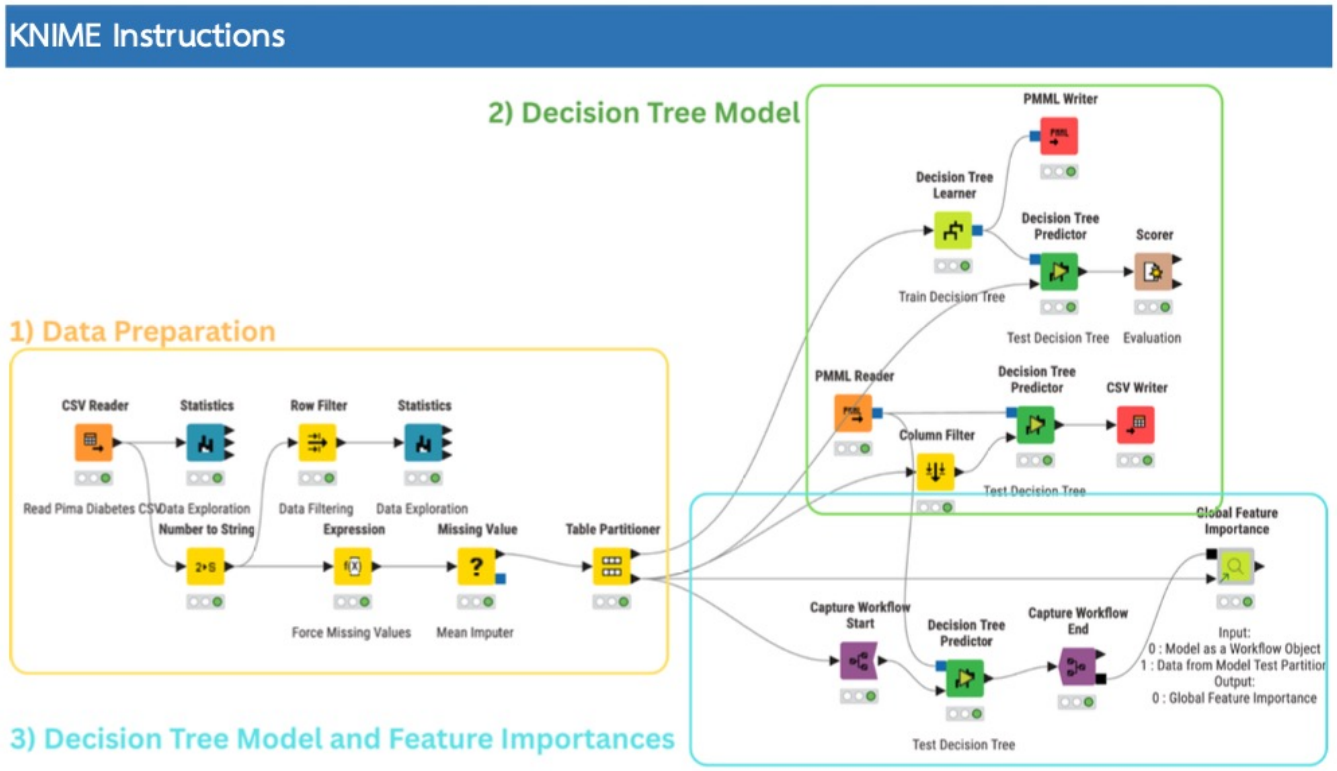
https://github.com/pvateekul/ieat2026

Classification with Tree-Based Models

Prof. Peerapon Vateekul, Ph.D.
Peerapon.v@chula.ac.th

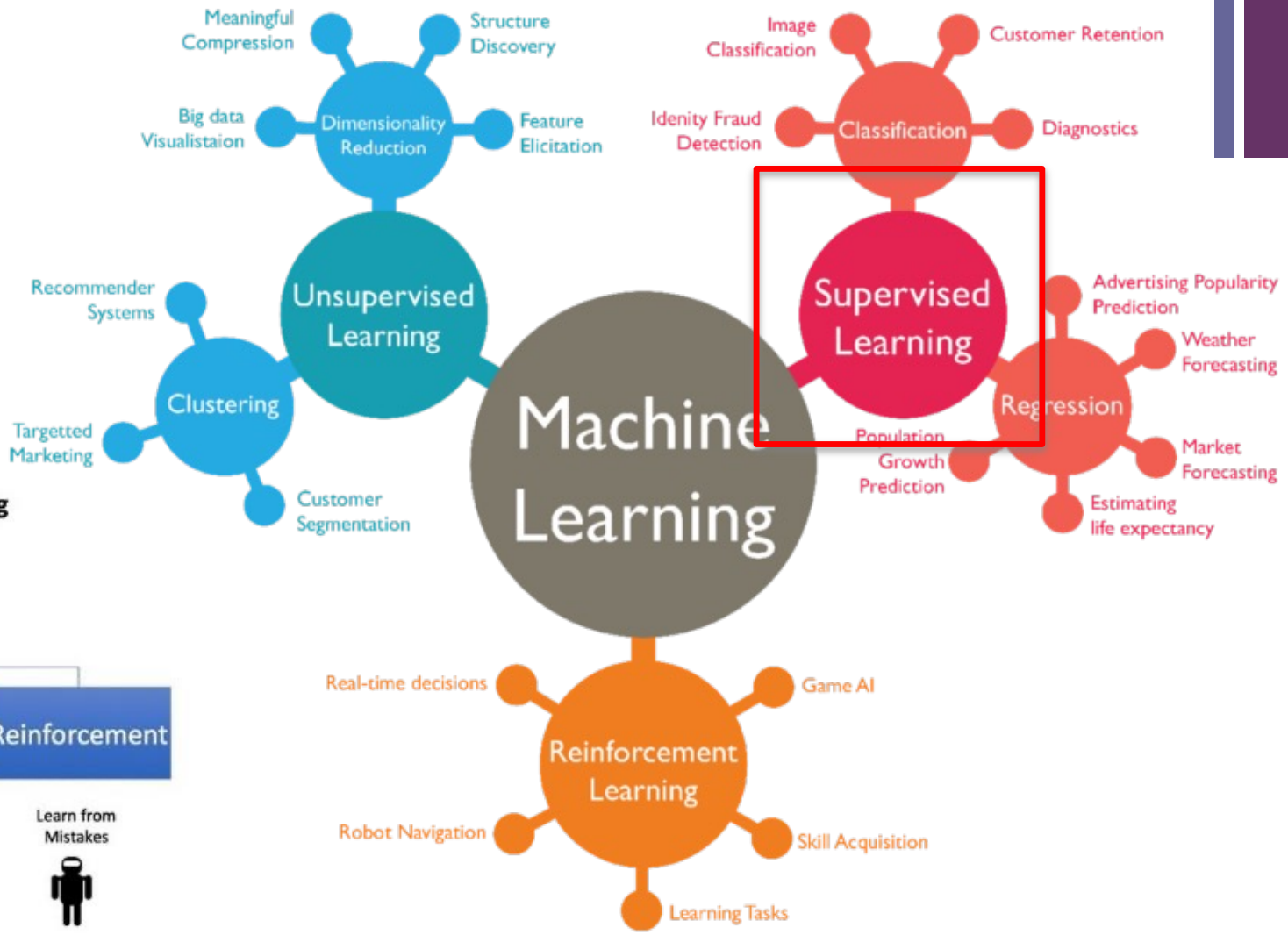
Outlines

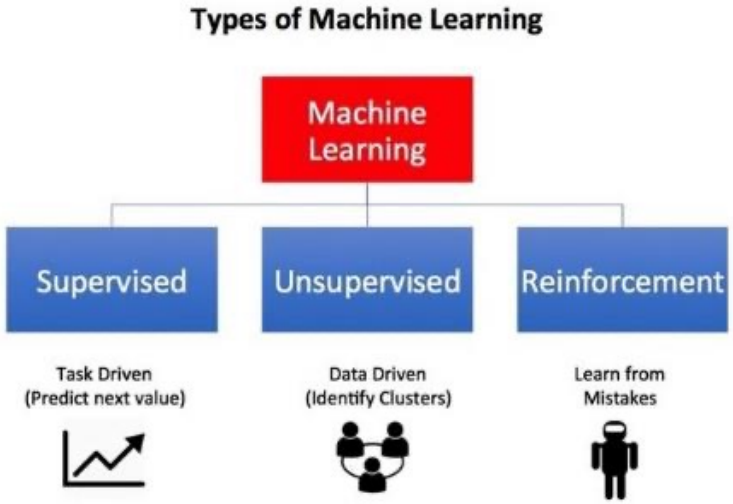
- Supervised Learning
- Decision Tree
- Classification Performance
- Knime
- Lab 1: Classification



Supervised Learning (Predictive Task)

Machine Learning



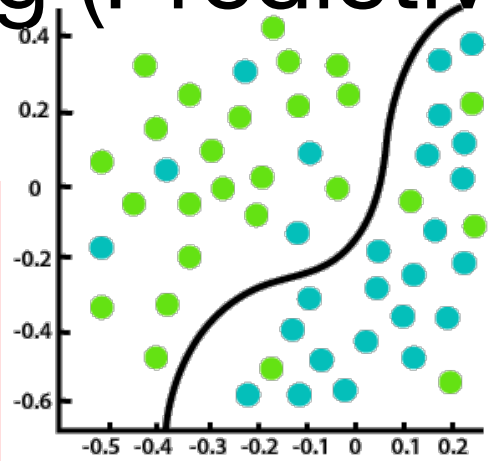


Supervised Learning (Predictive Task)

inputs | target

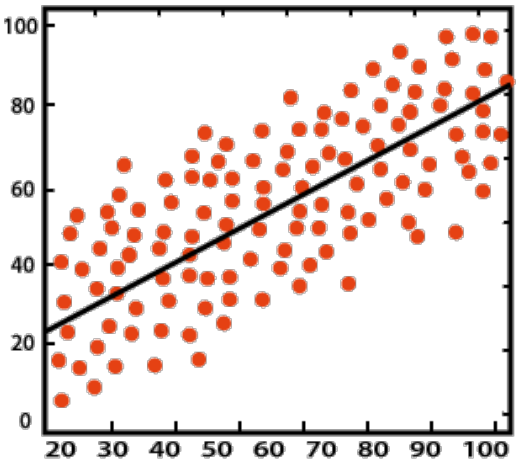
Age	Temp	Gender	Smell	Covid
25	39.0	Female	No	Yes
35	38.9	Female	No	Yes
32	36.5	Male	Yes	No

- Goal: To learn a prediction model mapping from inputs to output.
- Data without label (answer) is meaningless!
- Label should be provided by experts!



Classification

- Target is categorical variable.
- Example
- Covid diagnosis (yes/no)
- Disease diagnosis from gait information:
  - 1) Normal,
  - 2) Sick/Knee OA
  - 3) Sick/Parkinson



Regression

- Target is numeric variable.
- Example
- PD's state diagnosis from movement data.
- Glucose level prediction from breath particles.

There are two main processes: Train/Test

1) Training Phase: Model Construction

Training Data

inputs

target

Age	Income	Gender	Province	Purchase
25	25,000	Female	Bangkok	Yes
35	50,000	Female	Nontaburi	Yes
32	35,000	Male	Bangkok	No

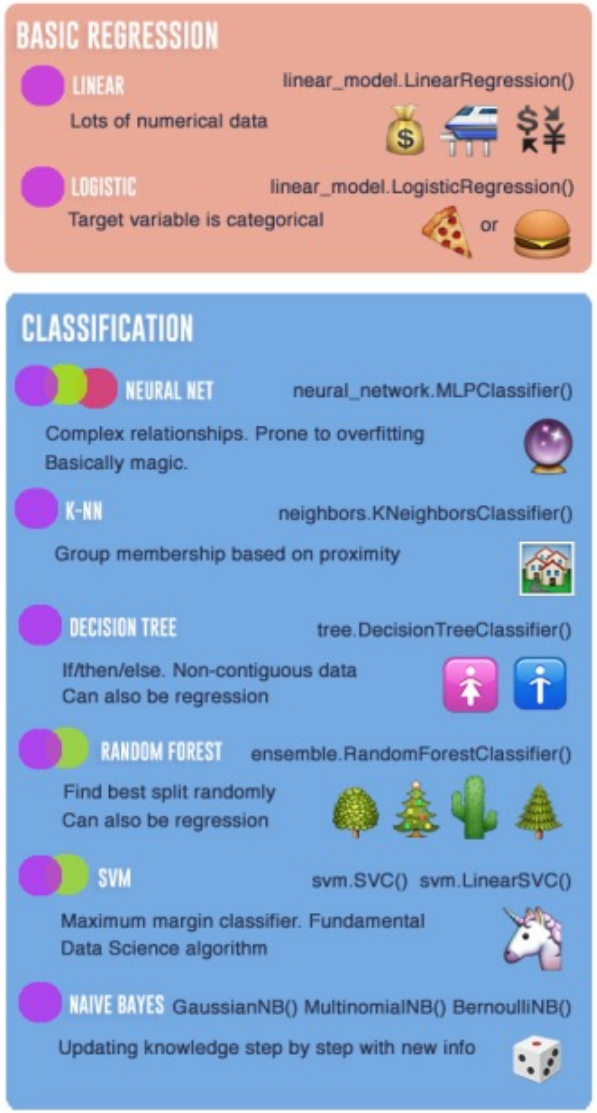
2) Testing Phase: Model Evaluation, Model Assessment
Also called "prediction, inference, scoring"

Testing Data

Age	Income	Gender	Province	Purchase
25	25,000	Female	Bangkok	?

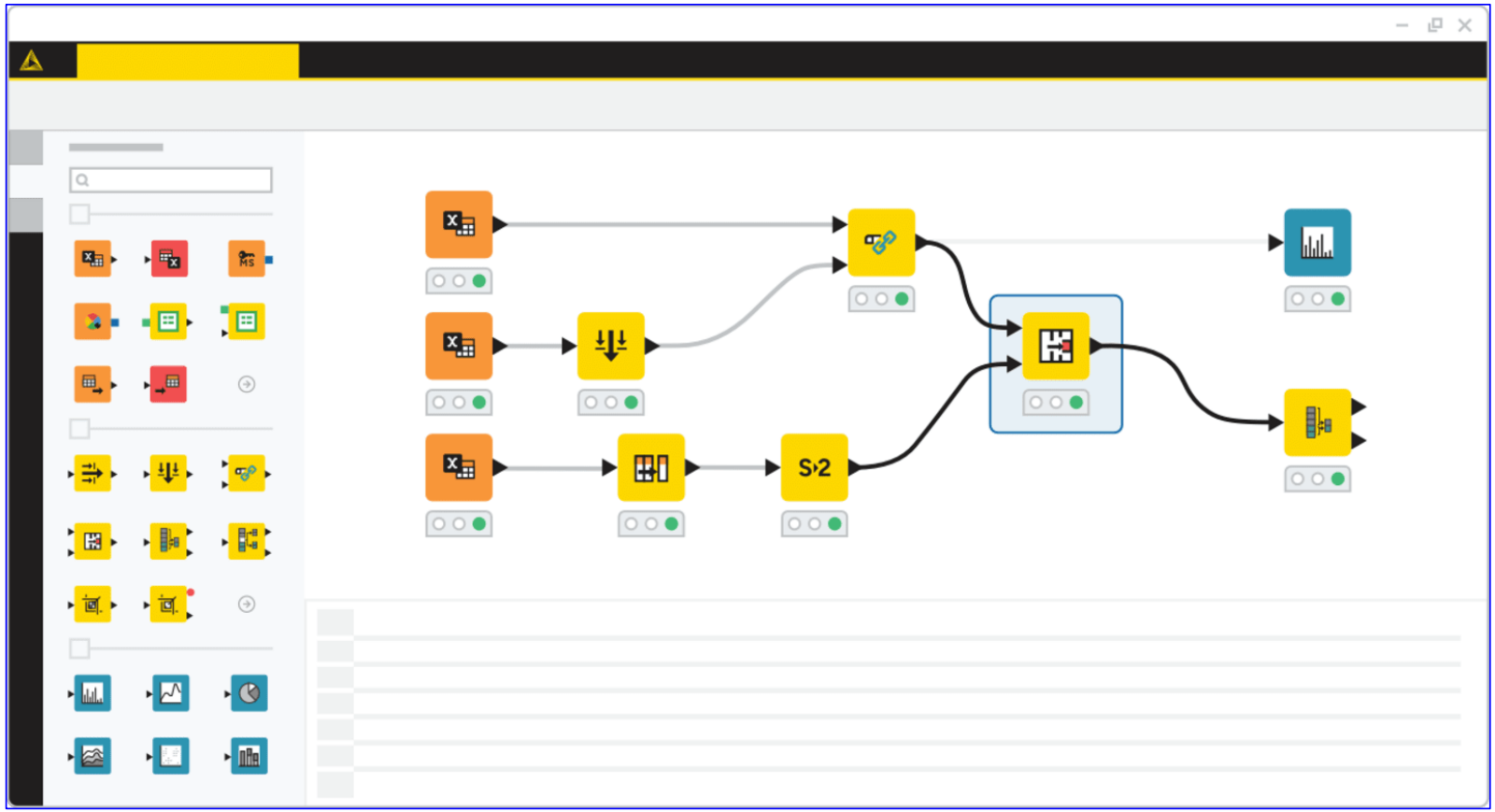
Prediction Algorithms

- Decision Tree
- (Logistic) Regression
- Neural Networks (NN)
- kNN
- Support Vector Machine
- Deep Learning



Low-Code/No-Code Software

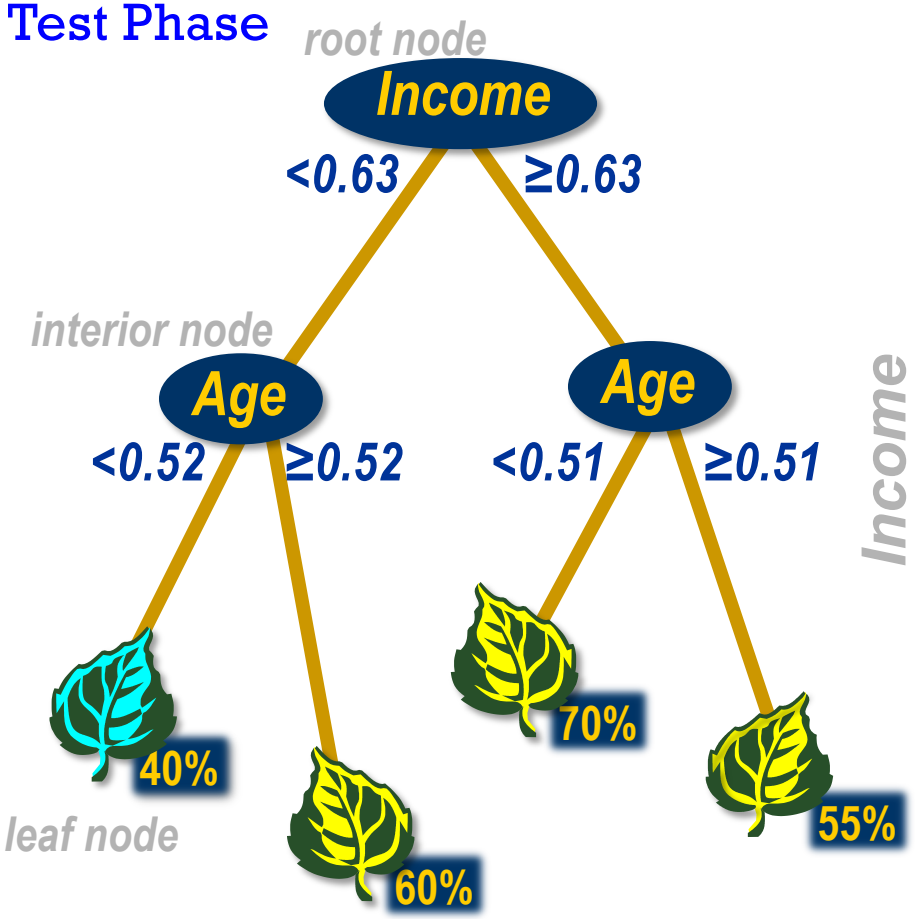
Open for Innovation
KNIME

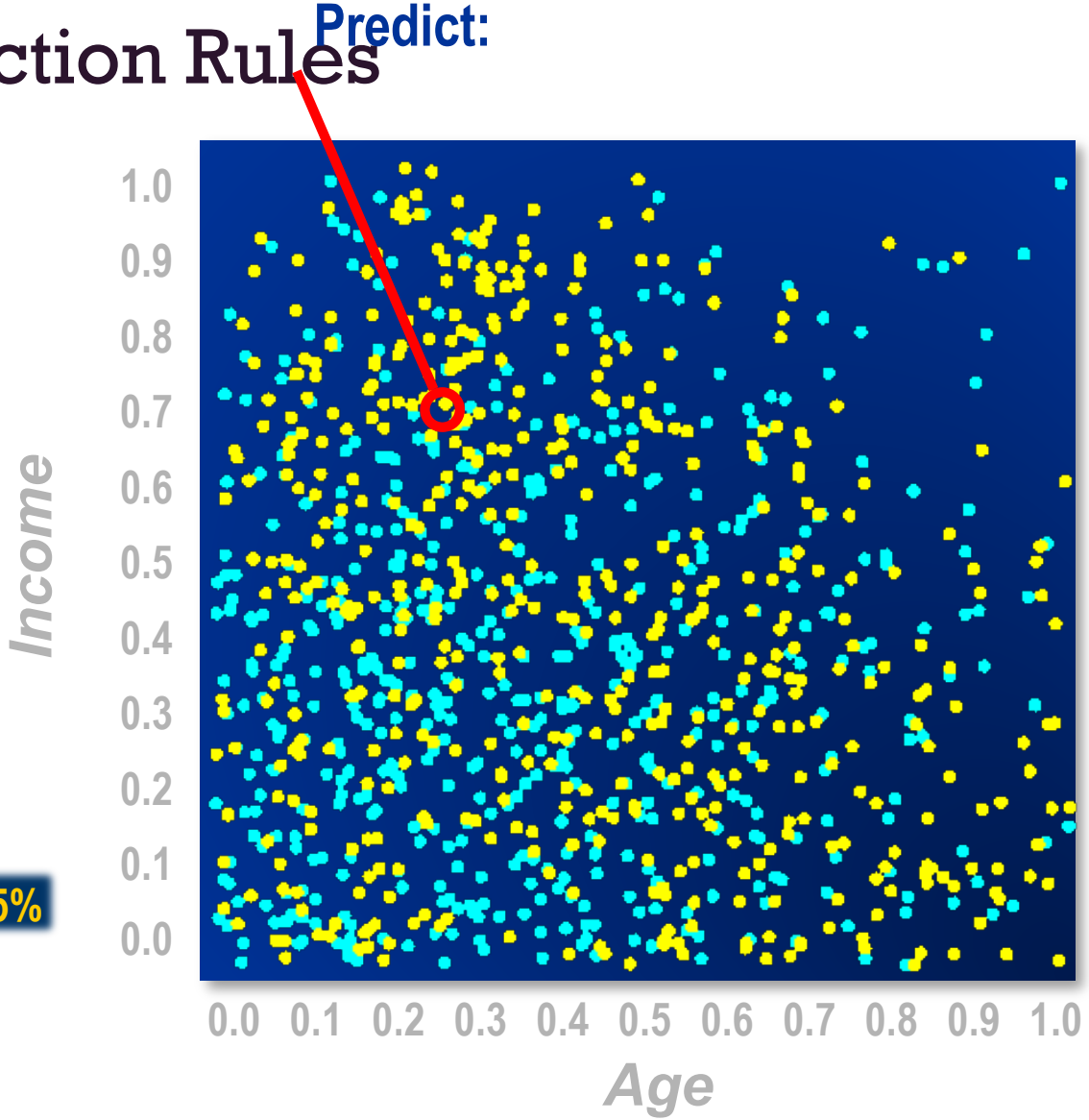


Decision Tree

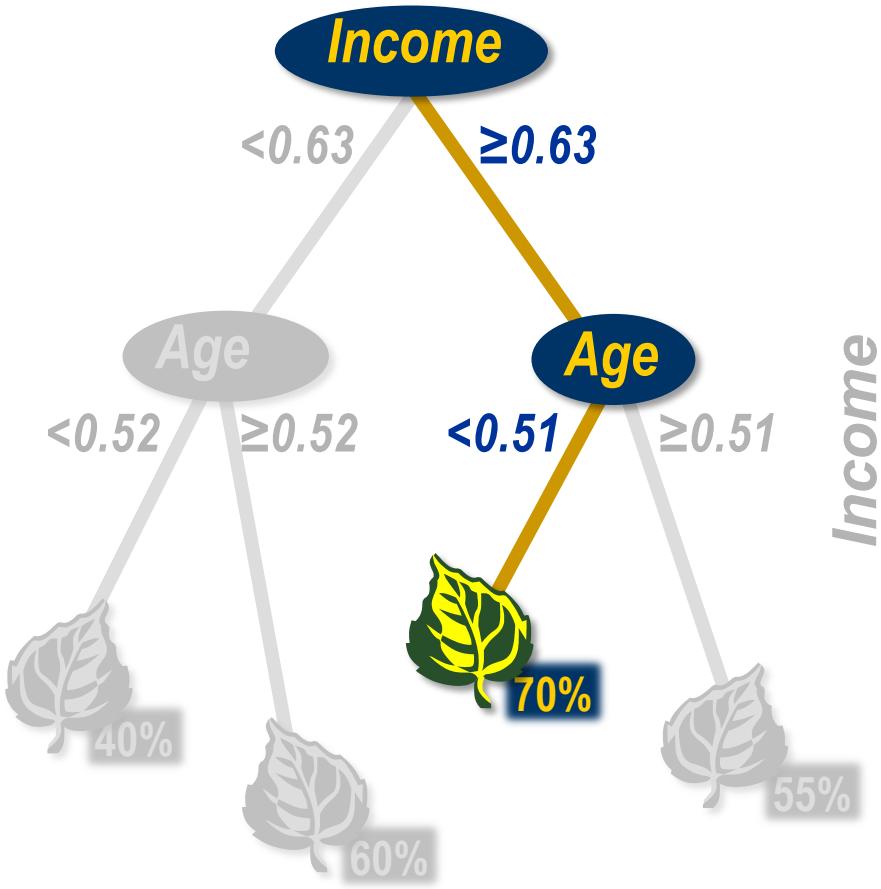
1) Decision Tree Prediction Rules

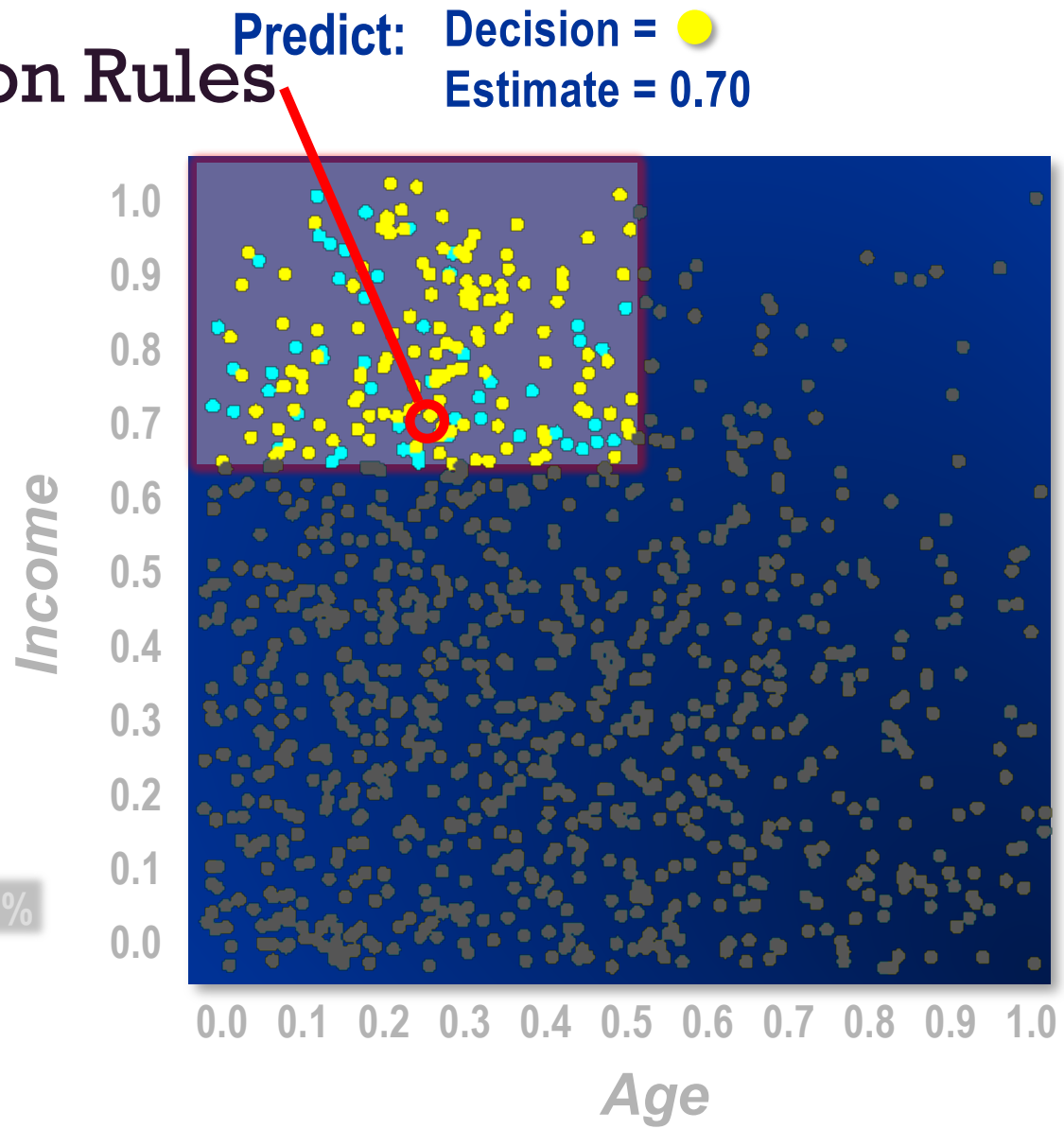
Test Phase





Decision Tree Prediction Rules

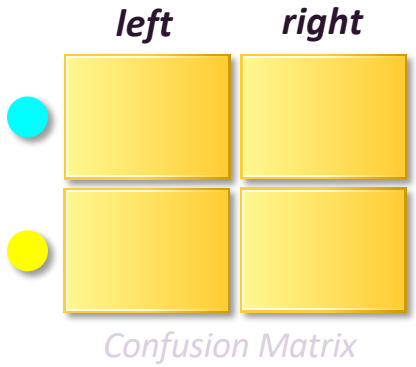




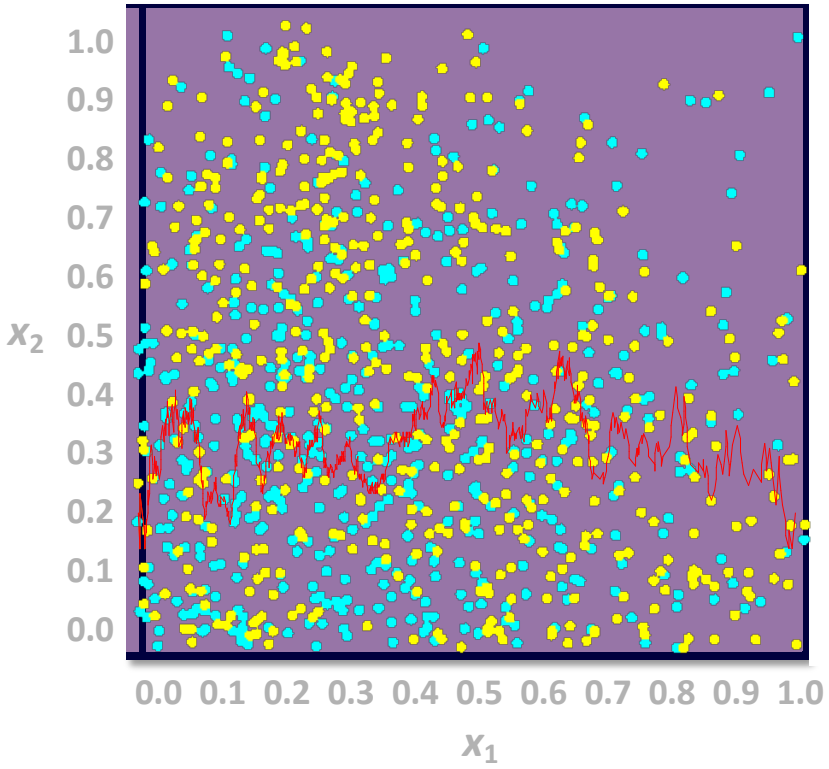
Model Essentials: Decision Trees

- Predict cases. — **Prediction rules**
- Select useful predictors. — **Split search**

Decision Tree Split Search

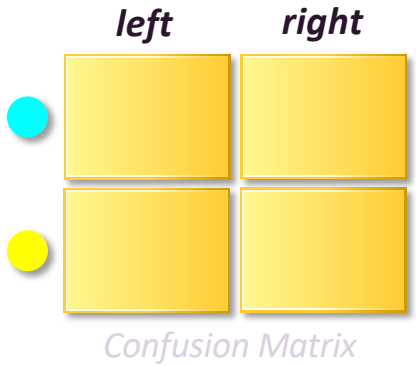


Calculate information gain on partitions on input x_1.

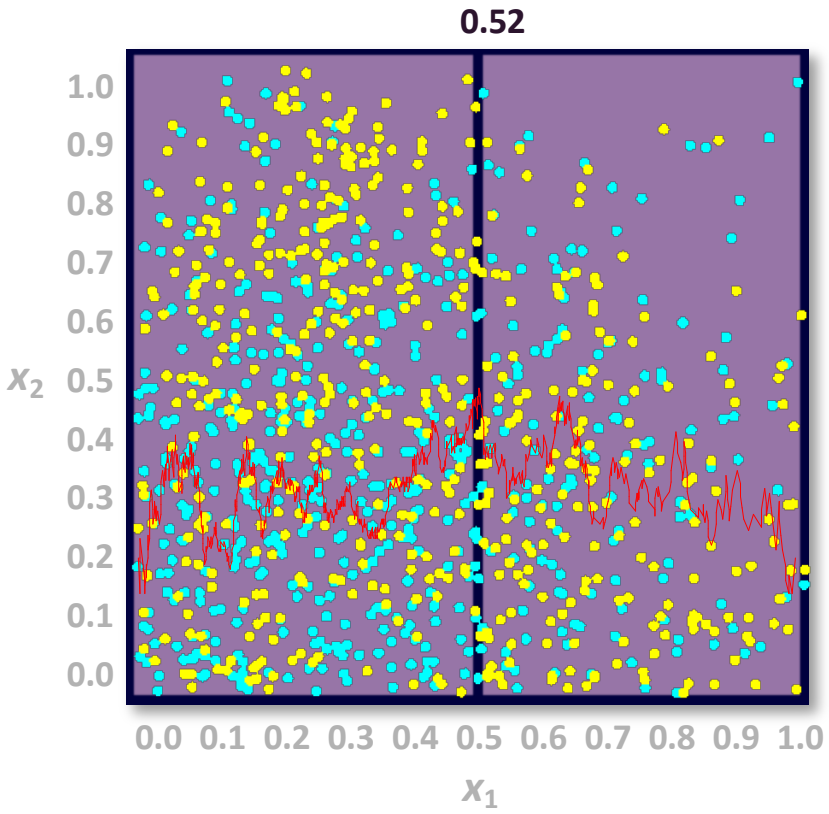


...

Decision Tree Split Search



Calculate the gain of every partition on input x_1.

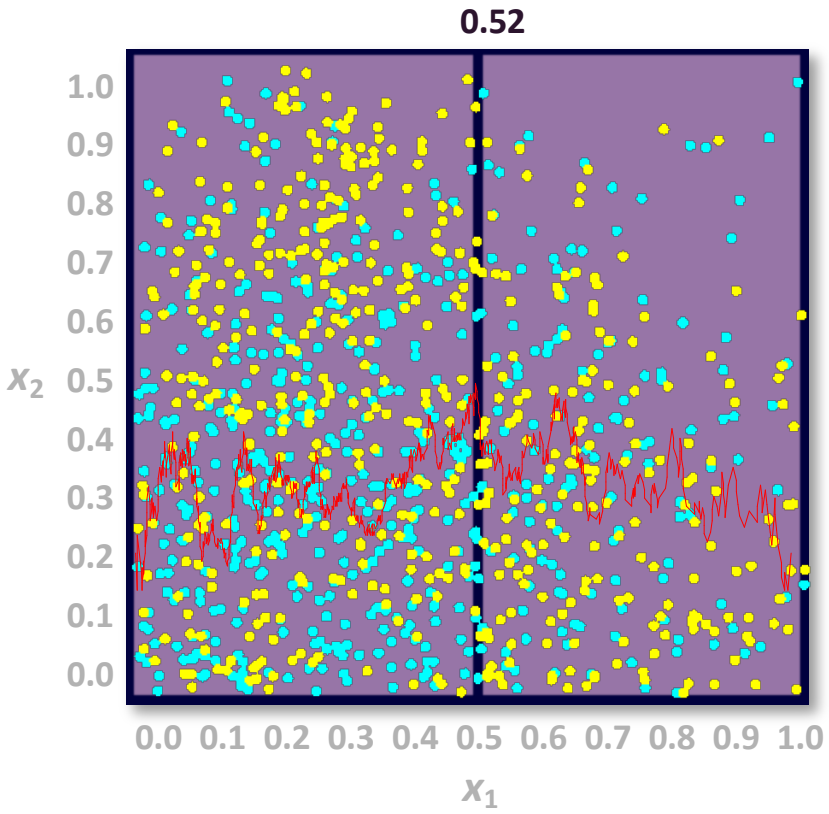


...

Decision Tree Split Search

	left	*right*
●	53%	42%
●	47%	58%

max gain(x_1)
0.0112



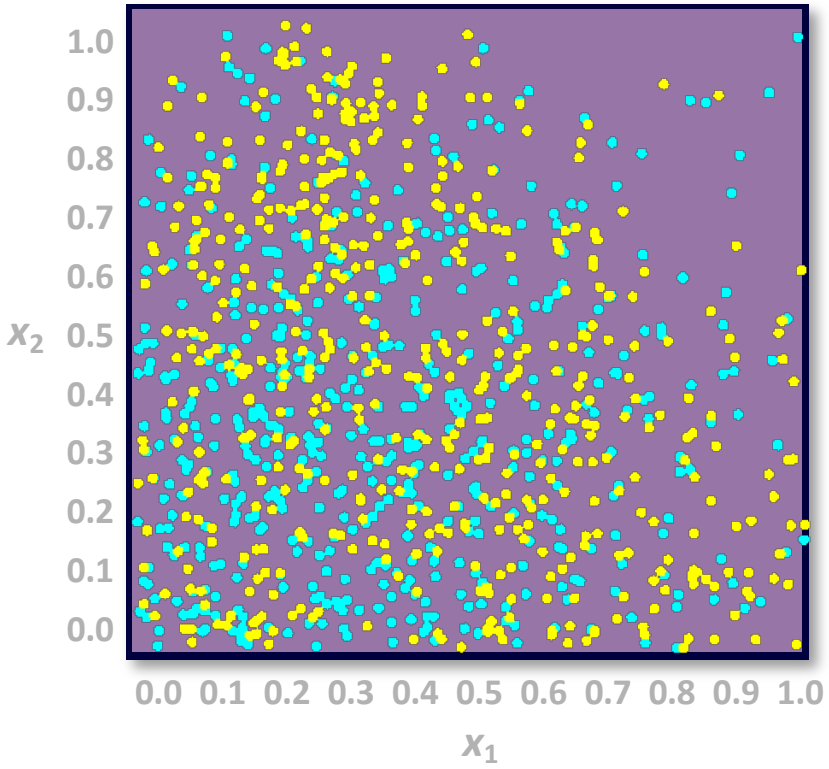
Select the partition with the maximum gain.

...

Decision Tree Split Search

	left	*right*
	53%	42%
	47%	58%

max Gain(x_1) 0.0112



Repeat for input x_2.

...

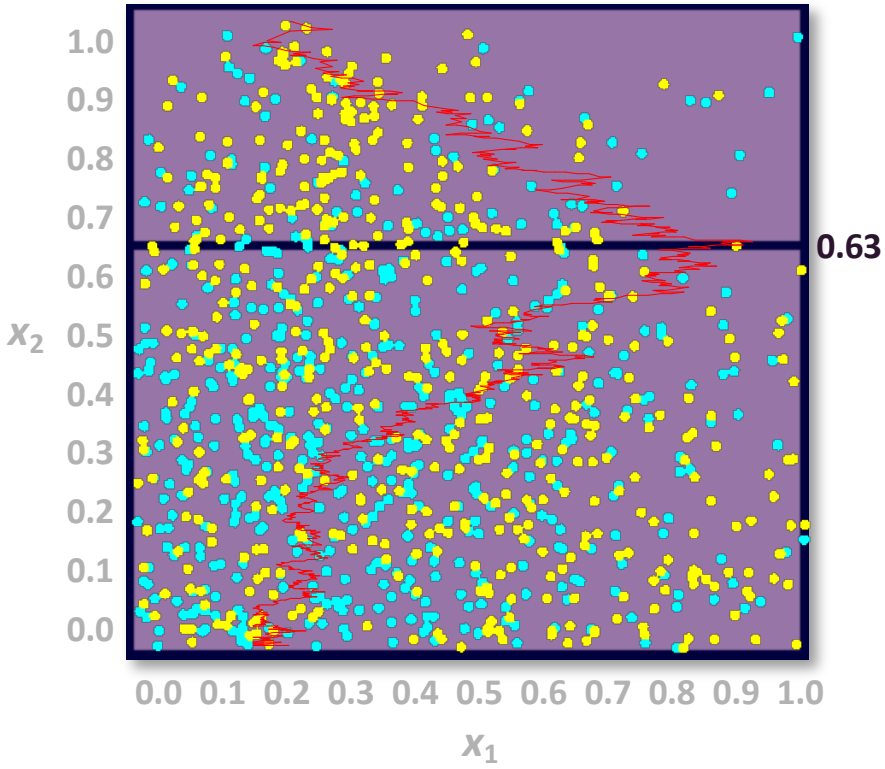
Decision Tree Split Search

	left	*right*
●	53%	42%
●	47%	58%

max gain(x_1) 0.0112

	bottom	*top*
●	54%	35%
●	46%	65%

max gain(x_2) 0.0273



...

Decision Tree Split Search

	left	*right*
●	53%	42%
●	47%	58%

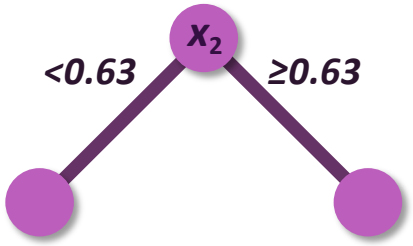
max gain(x_1)
0.0112

	bottom	*top*
●	54%	35%
●	46%	65%

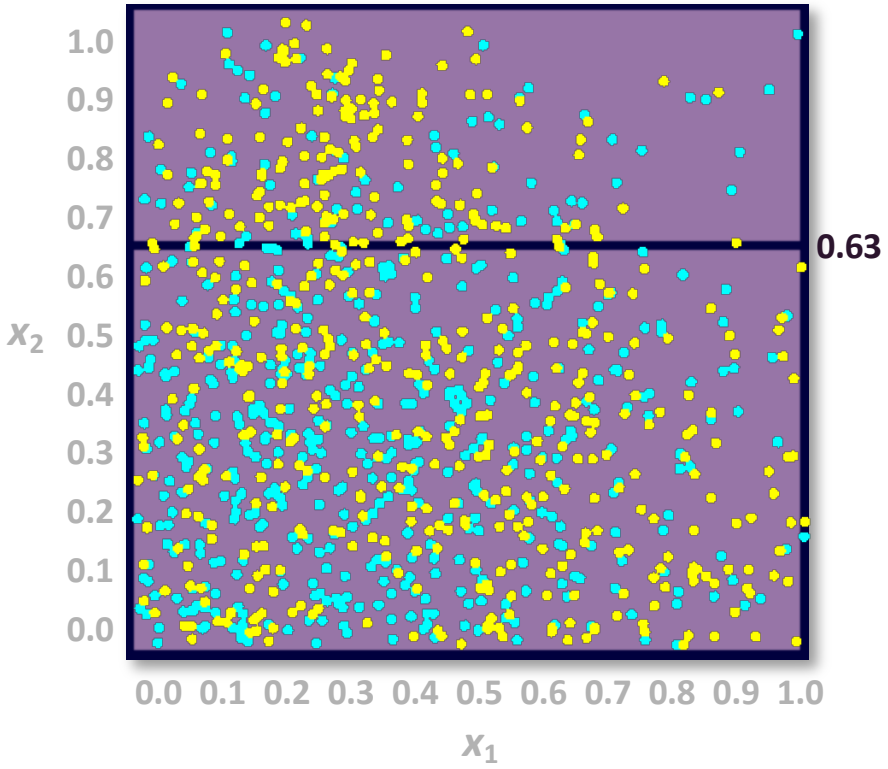
max gain(x_2)
0.0273

...

Decision Tree Split Search

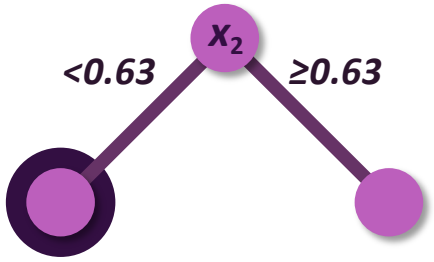


Create a partition rule from the best partition across all inputs.

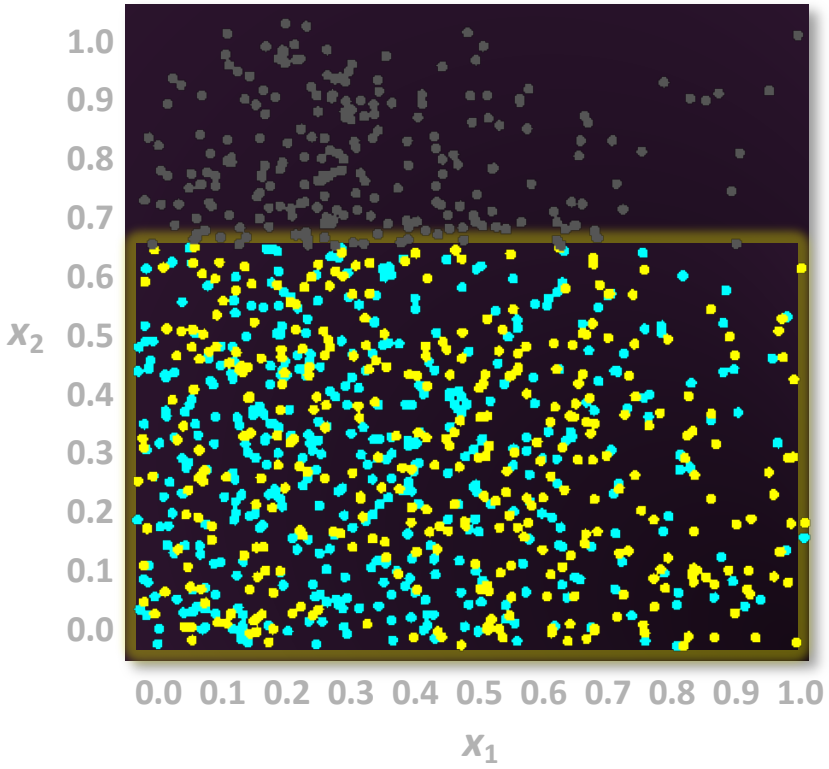


...

Decision Tree Split Search



Repeat the process in each subset.

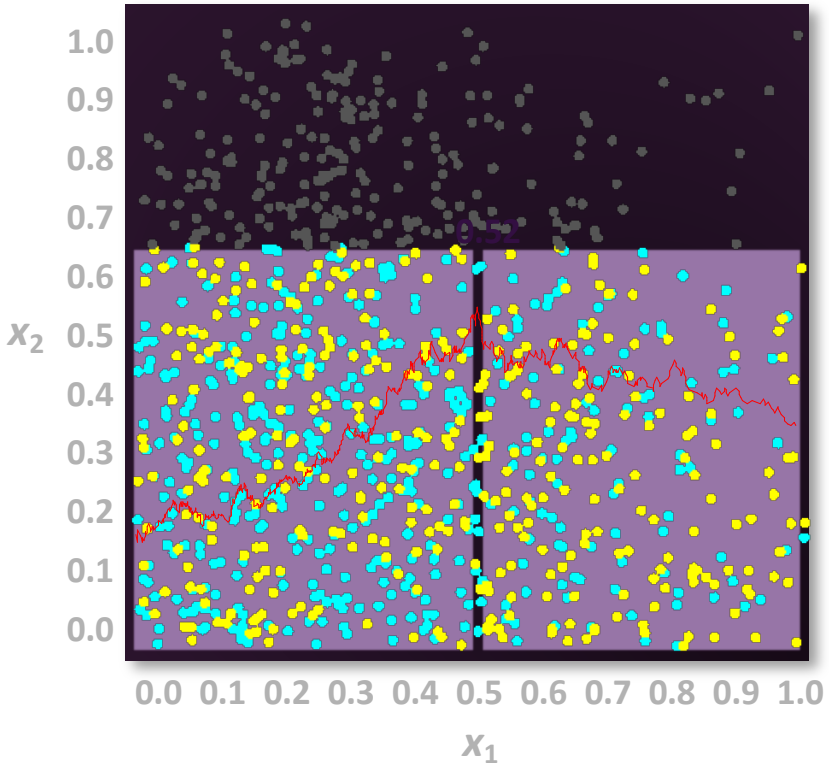


...

Decision Tree Split Search

	left	*right*
●	61%	55%
●	39%	45%

max gain(x_1)
0.0203



...

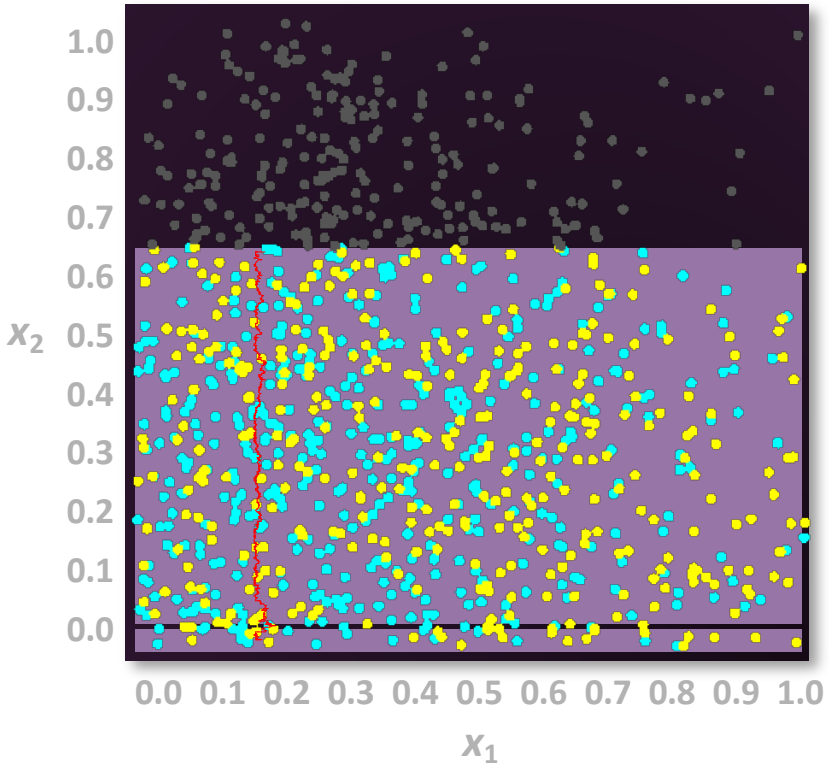
Decision Tree Split Search

	left	*right*
	61%	55%
	39%	45%

max gain(x_1) 0.0203

	bottom	*top*
	38%	55%
	62%	45%

max gain(x_2) 0.019



...

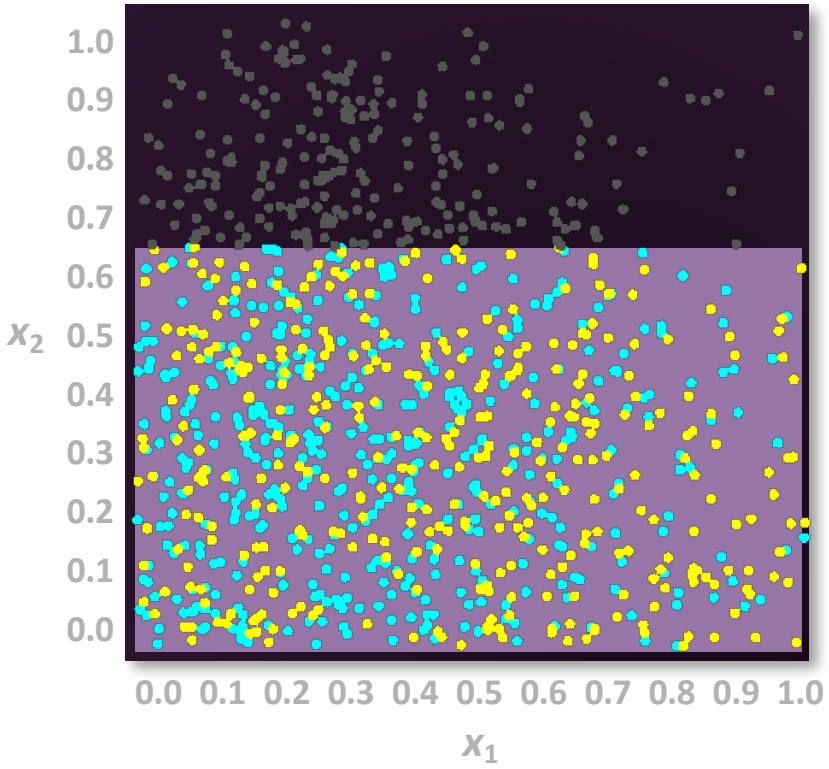
Decision Tree Split Search

	left	*right*
●	61%	55%
●	39%	45%

max gain(x_1)
0.0203

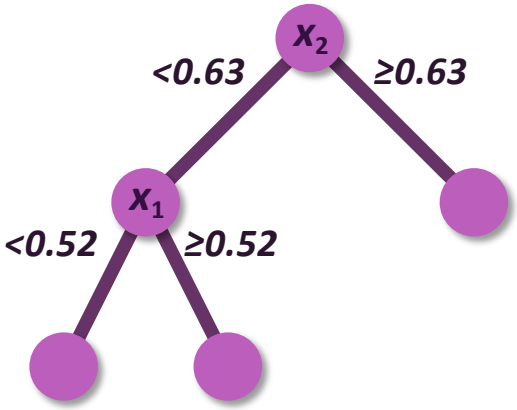
	bottom	*top*
●	38%	55%
●	62%	45%

max gain(x_2)
0.019

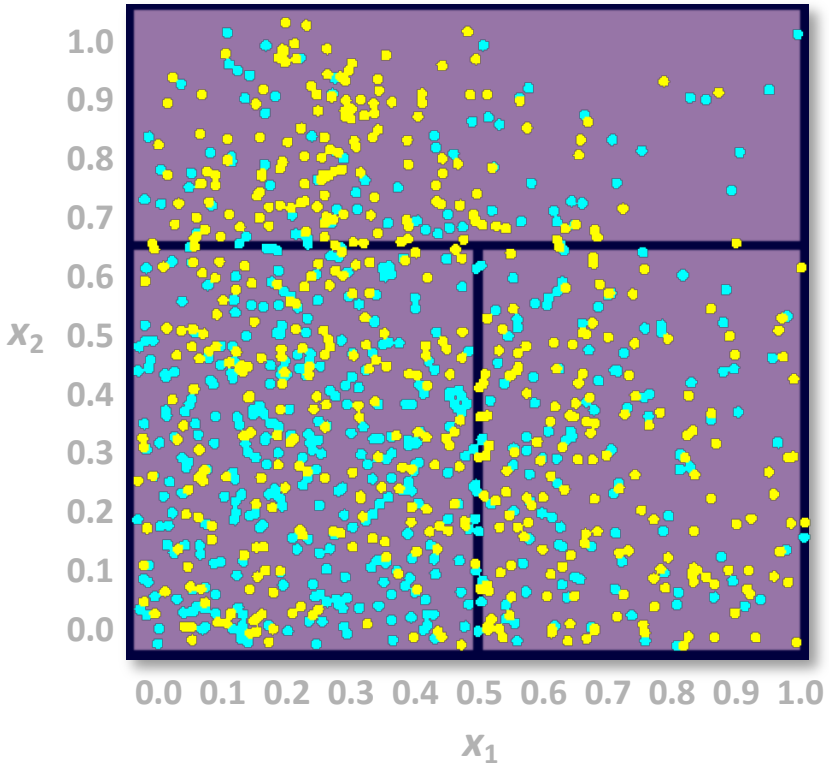


...

Decision Tree Split Search

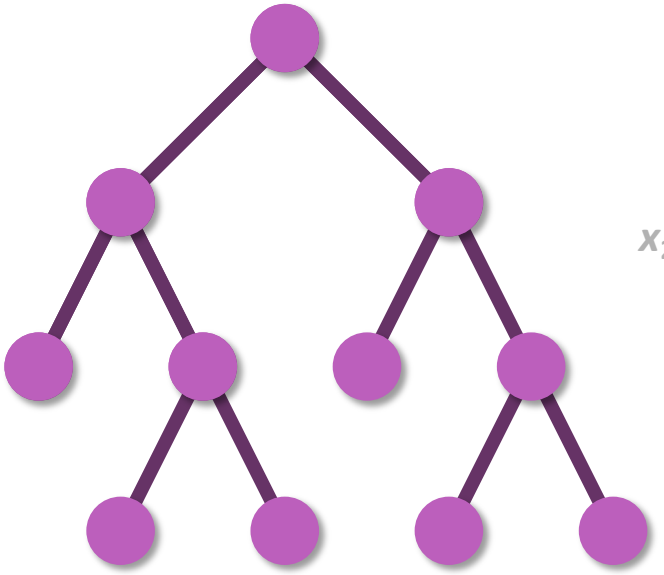


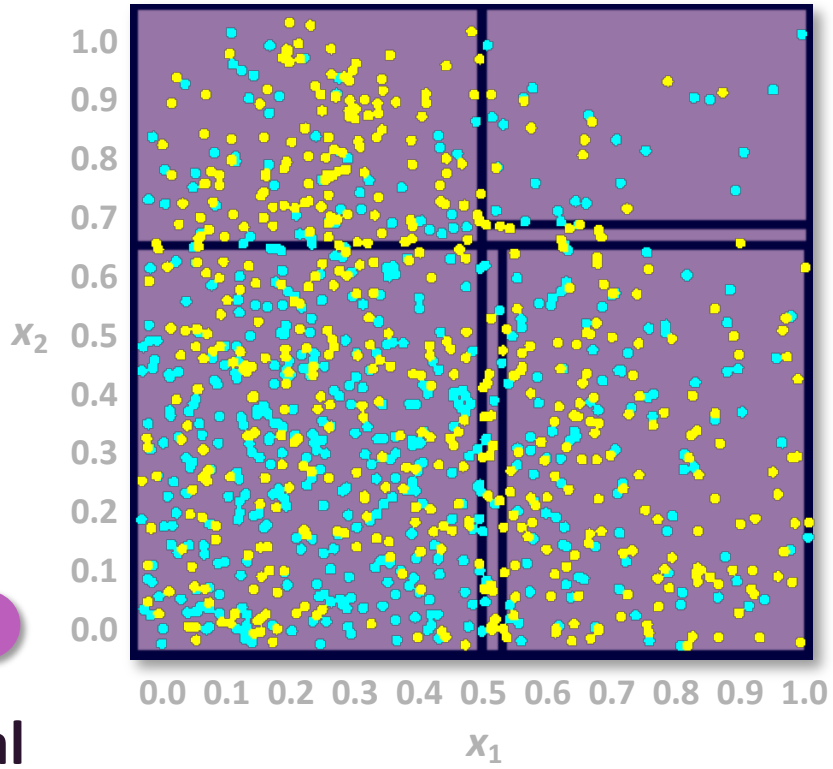
Create a second partition rule.



...

Decision Tree Split Search





Repeat to form a maximal tree.

1) Entropy (impurity)

- **Entropy** is a measure of disorder or uncertainty and the goal of machine learning models and general is to reduce uncertainty.

$$Entropy = \sum_{i=1}^{n} -p_i log_2 p_i$$

scipy.stats.entropy

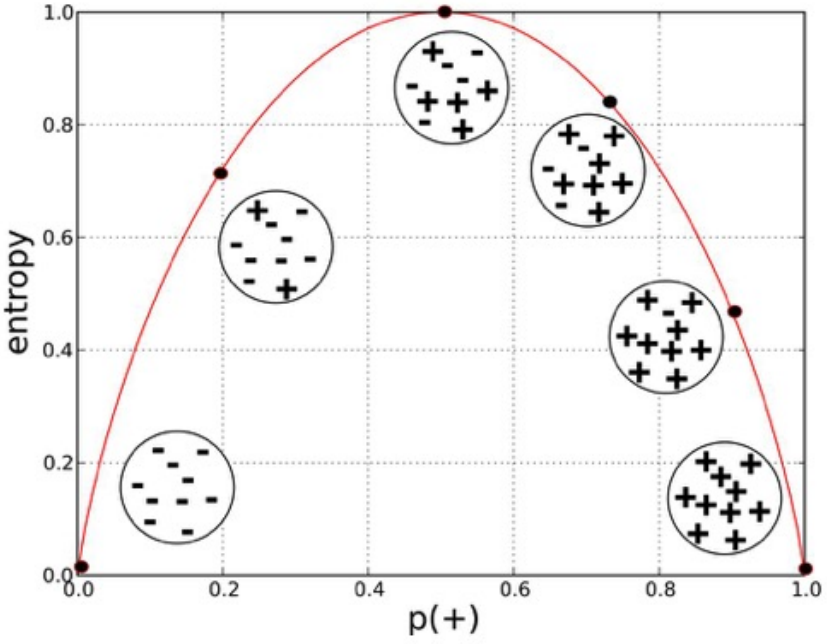
scipy.stats.**entropy**(*pk*, *qk=None*, *base=None*, *axis=0*)

Calculate the entropy of a distribution for given probability values.

If only probabilities *pk* are given, the entropy is calculated as `S = -sum(pk * log(pk), axis=axis)`.

If *qk* is not None, then compute the Kullback-Leibler divergence `S = sum(pk * log(pk / qk), axis=axis)`.

This routine will normalize *pk* and *qk* if they don't sum to 1.



Source: Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking

Information Gain
Before - After

- Which one is Better ?

Split w/ Age: 70
∑ Entropy: 0.350

Split w/ Age: 50
∑ Entropy: 0.348

- Information Gain: measure the reduction of this disorder in our target variable/class given additional information

$$InformationGain = Entropy(before) - \sum Entropy(after)$$

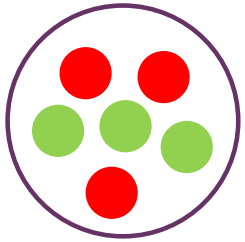
- "Before" = Entropy of Parent Node
  "After" = Entropy of Child Nodes

2) Gini Impurity
Gini Reduction = Before - After

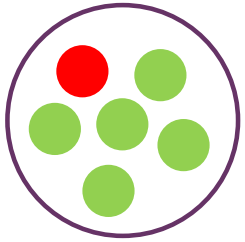
- Another way to measure how well a splitting feature.

$$Gini = 1 - \sum_{i=1}^{n} (P_i)^2$$

When P is the probability of class i in data-set.



$$Gini = 1 - ((3/6)^2 + (3/6)^2)$$
$$= 0.5$$



$$Gini = 1 - ((5/6)^2 + (1/6)^2)$$
$$= 0.28$$

- Easy to calculation, may take less time to build in large dataset.

Source: https://towardsdatascience.com/understanding-decision-tree-classification-with-scikit-learn-2ddf272731bd

Types of Decision Tree

Algorithm	Splitting Measure
ID3	Entropy
C4.5	Gain Ratio
CART	Gini index
CHAID	Chi-squared test

Important Parameters in Decision Tree

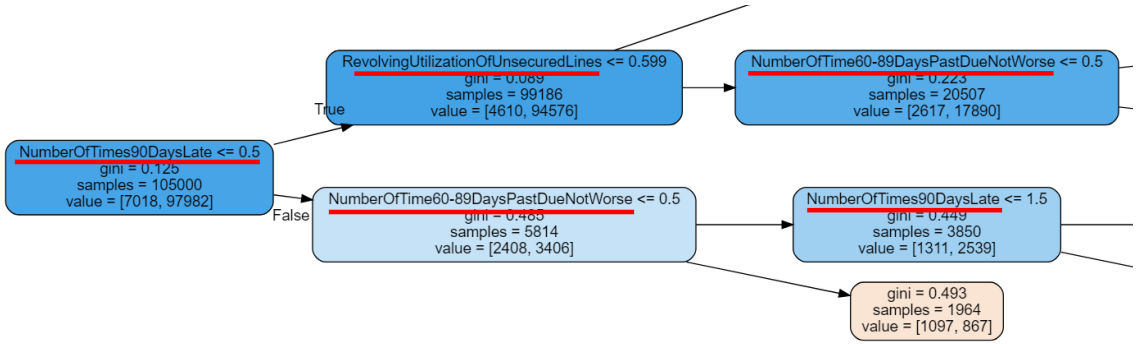
- Splitting measure (criterion) : gini / entropy
- Maximum depth : ~5-10 (depend on number of feature)
- Maximum leaf nodes : depend on number of class (target) and feature
- Minimum sample split : 5 – 20% (depend on number of data)

Which features are important ?

- Check how important by . feature_importances_
- The importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as the gini importance.

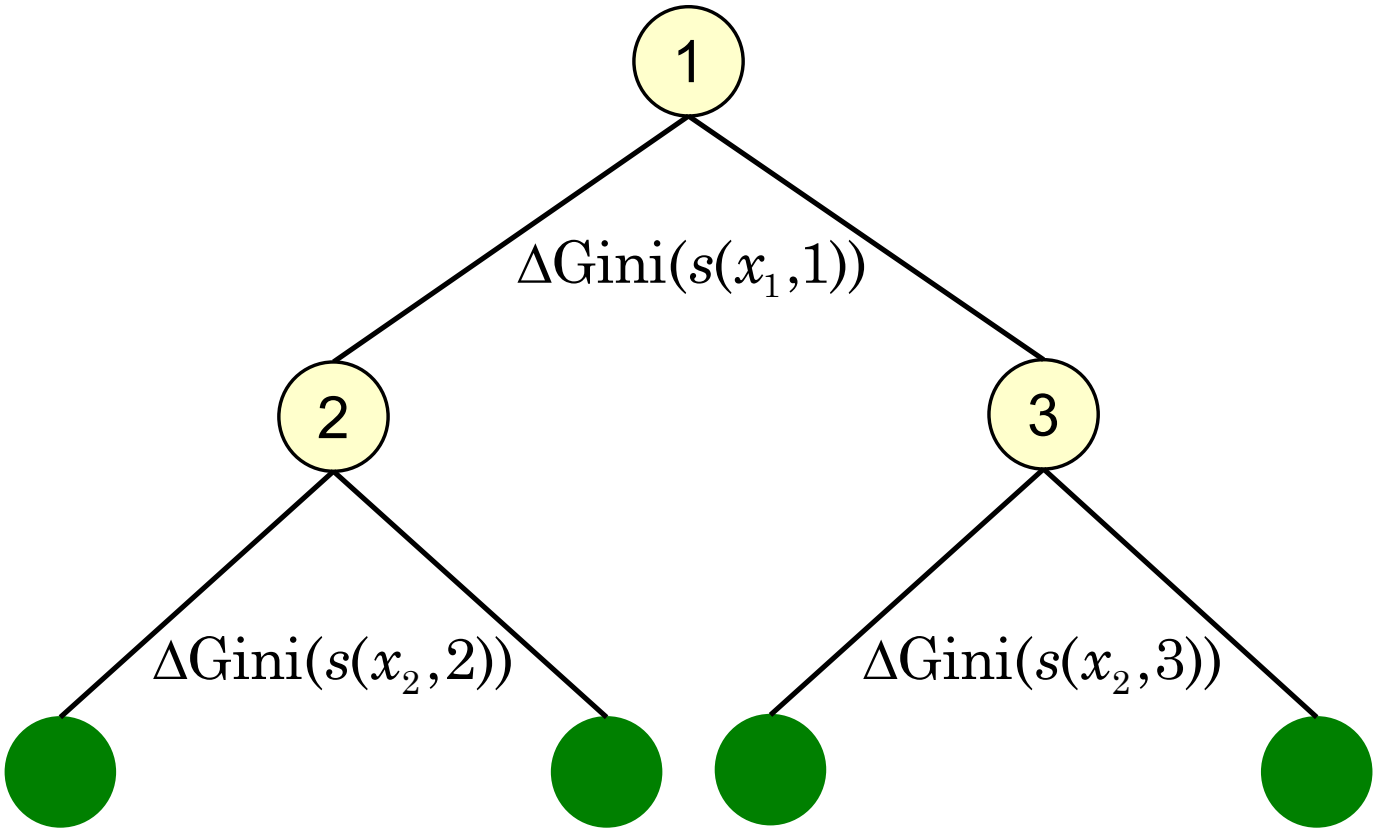
```
dtree.feature_importances_
```

```
array([0.15330291, 0.00371739, 0.07500437, 0.        , 0.        ,
       0.        , 0.61403332, 0.        , 0.15394201, 0.        ])
```




```
======= Features Important =======
0.614 : Numberoftimes90dayslate
0.154 : Numberoftime60-89dayspastduenotworse
0.153 : Revolvingutilizationofunsecuredlines
0.075 : Numberoftime30-59dayspastduenotworse
0.004 : Age
0.000 : Debtratio
0.000 : Monthlyincome
0.000 : Numberofopencreditlinesandloans
0.000 : Numberrealestateloansorlines
0.000 : Numberofdependents
```

+ Variable Importance



Classification Performance

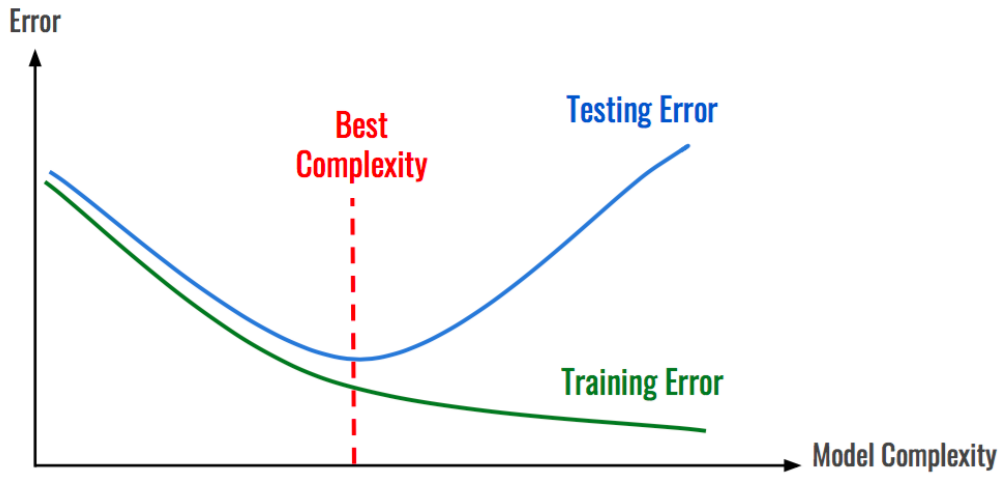
Evaluation (Train/Test Split)

Training Data

Age	Income	Purchase
25	25,000	Yes
35	50,000	Yes
32	35,000	No

Testing Data

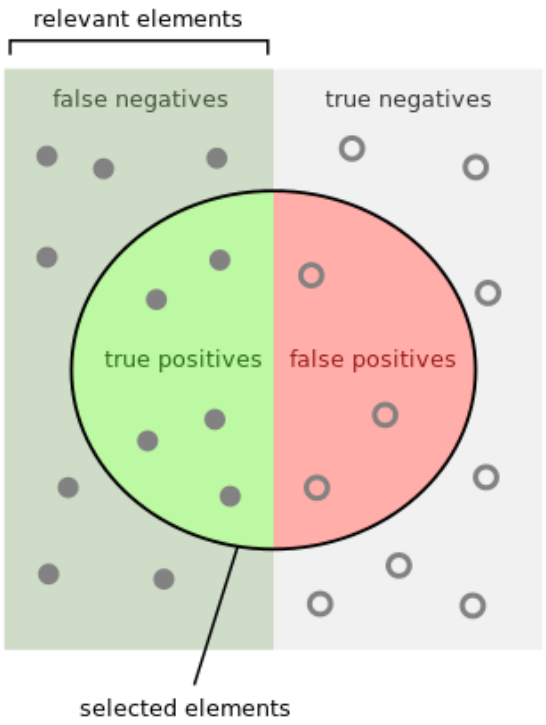
Age	Income	Purchase
27	35,000	Yes
23	20,000	No
45	34,000	No

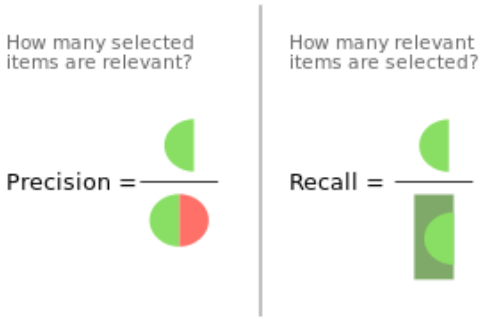


Confusion Matrix Precision, Recall, F1

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

- Precision = correctly predict = TP / (TP + FP)
- Recall = coverage = TP / (TP + FN)
- F1 = (2*pre*rec) / (pre + rec)





Knime

What is KNIME

KNIME (Konstanz Information Miner) is a visual, drag-and-drop platform for data analytics and machine learning that lets you build workflows without heavy coding.

- **Visual workflow tool:** connect “nodes” to create a data pipeline (like a flowchart).
- **End-to-end analytics:** import data → clean/transform → explore → train models → evaluate → export results.
- **Low-code / no-code friendly:** great for beginners, but can also integrate Python/R/SQL when needed.
- **Reproducible workflows:** your steps are saved as a workflow, so you can rerun and share the same process.
- **Extensible:** lots of built-in nodes + community extensions for ML, text mining, and more.

How to use KNIME?

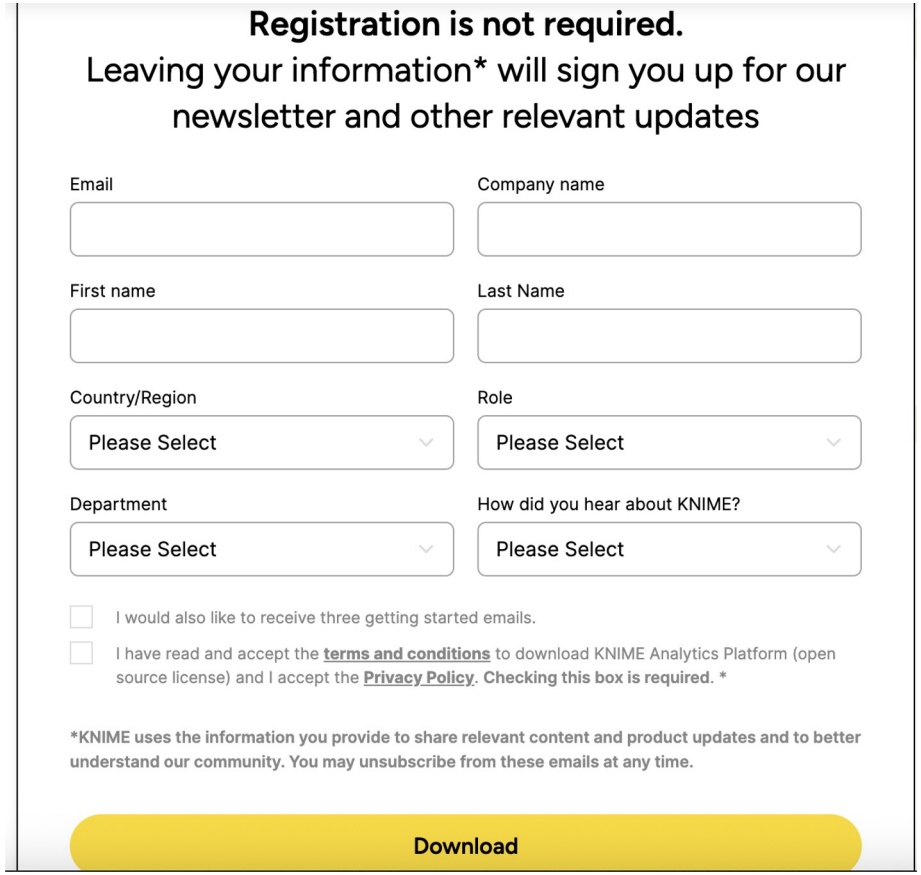
Local (Free)

- **KNIME Analytics Platform (Desktop)** = runs on your own computer
- Free and open-source (use it without paying).

Cloud / Collaboration (Paid)

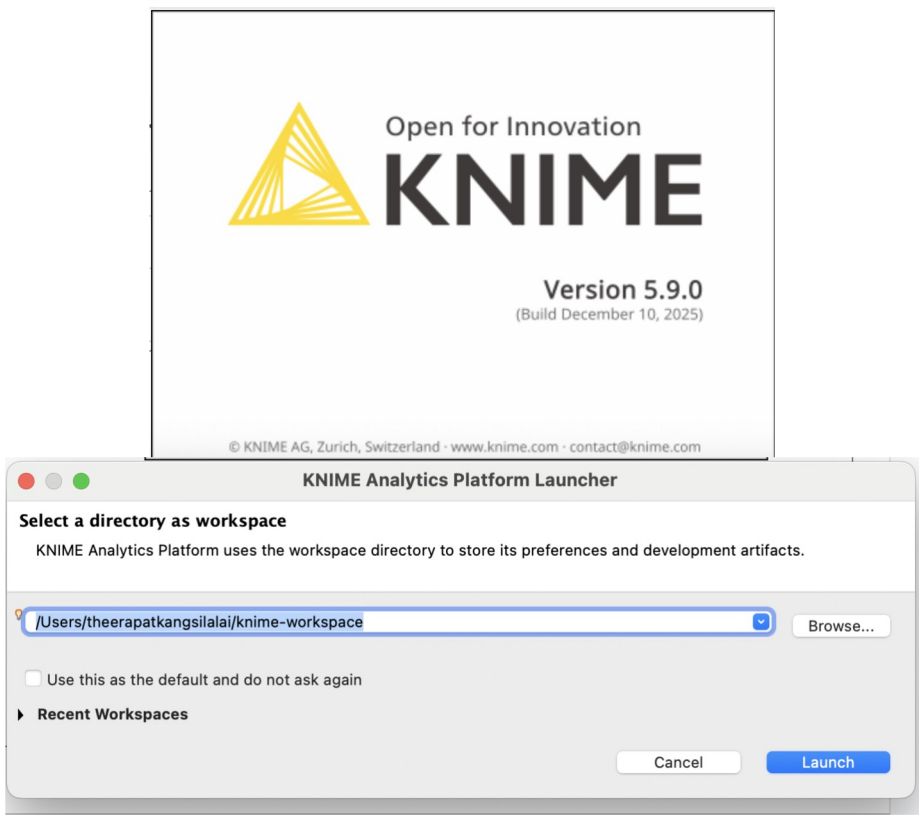
- KNIME Team plans add private collaboration and managed execution features.
- KNIME Business Hub: enterprise offering (SaaS or self-hosted); pricing is on request.

Download KNIME (LINK)



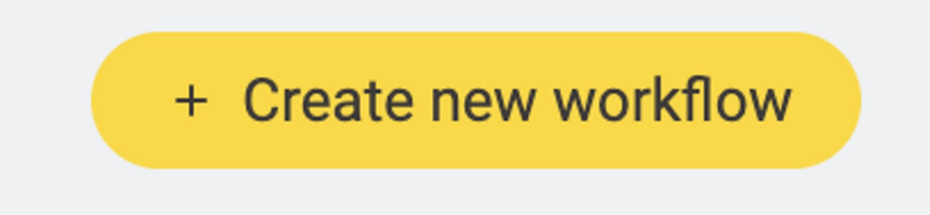
- Download KNIME Analytics Platform (Desktop) from the official KNIME website.
- **Install and launch the application.**
- Registration is not required to use KNIME Desktop for most labs (you can build and run workflows locally).
- In this course, **you will need to register/sign in later** to access a community extension node.

Open KNIME and select you folder



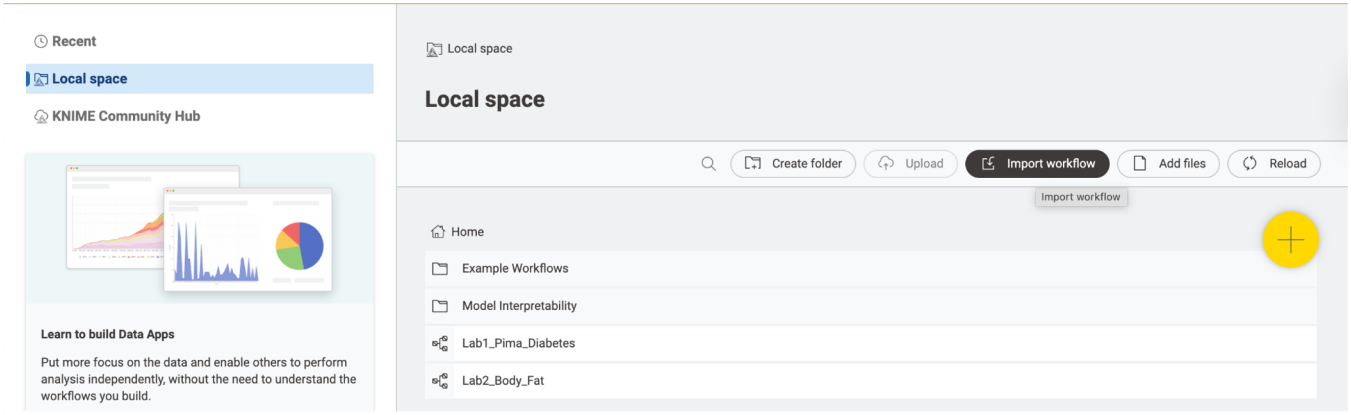
Then launch

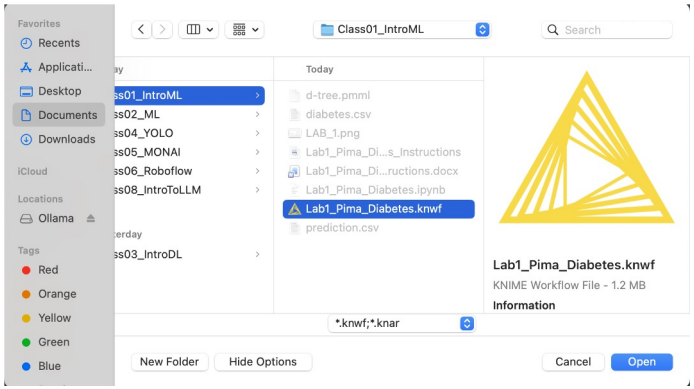
Create workflow



You can create your workflow **by clicking this button**.

Or import the existed workflow into your workspace in `**Local space**` tab.

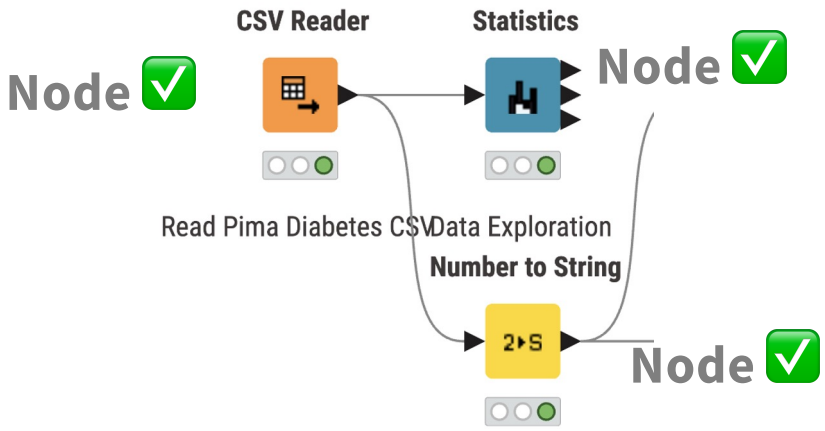




KNIME Basics: Nodes, Colors, Ports, and Status

What is a “Node”?

- A node is one step in your workflow (e.g., read data, clean data, train a model, evaluate results).
- Nodes are shown as colored boxes with input/output ports and a status indicator.



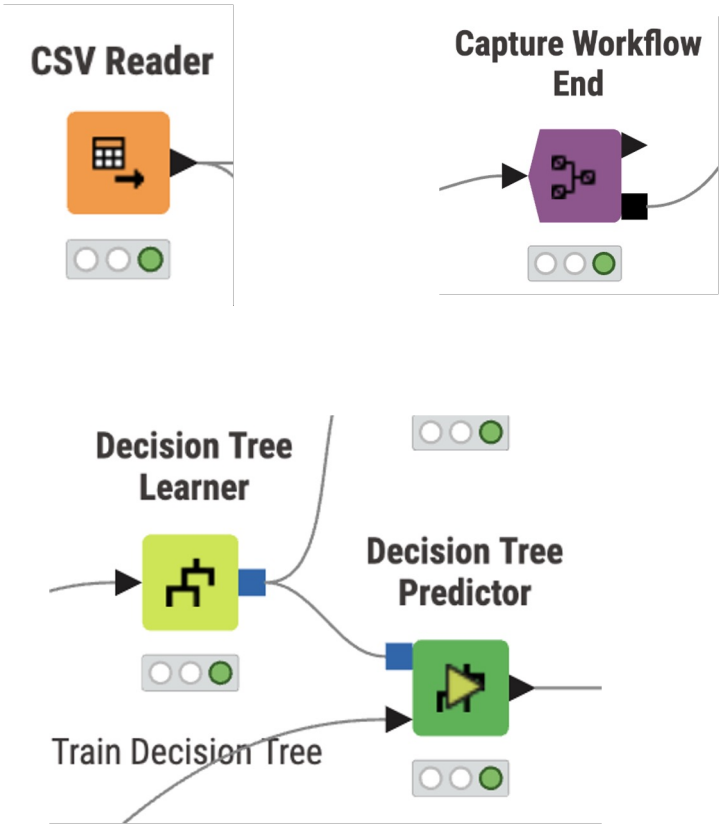
KNIME Basics: Nodes, Colors, Ports, and Status

KNIME uses color as a quick “category hint” (useful for scanning a workflow; not a strict rule):

- **Orange:** Readers (import data)
- **Red:** Writers (export/save results)
- **Yellow:** Data manipulation / transformation
- **Light Green:** Model learners (train models)
- **Dark Green:** Predictors (apply models)
- **Blue:** Visualization
- **Light Blue:** Loop / flow control
- **Brown:** Utility
- **Gray:** Components (a packaged mini-workflow)

KNIME Basics: Nodes, Colors, Ports, and Status

- **Triangle ports (Data Table):** pass tabular data (rows/columns) between nodes.
- **Square ports (Model / Special objects):** pass objects like **models (PMML)** or other non-table objects.
- **Blue square** commonly indicates a **model/PMML** connection between learner → predictor.



KNIME Basics: Nodes, Colors, Ports, and Status

The “Three Circles” under each node (Node Status / Traffic Light)

- **Red** = not configured (new node / missing settings)
- **Yellow** = configured (settings done, not executed yet)
- **Green** = executed successfully

Lab 1: Classification

LAB 1: Classification

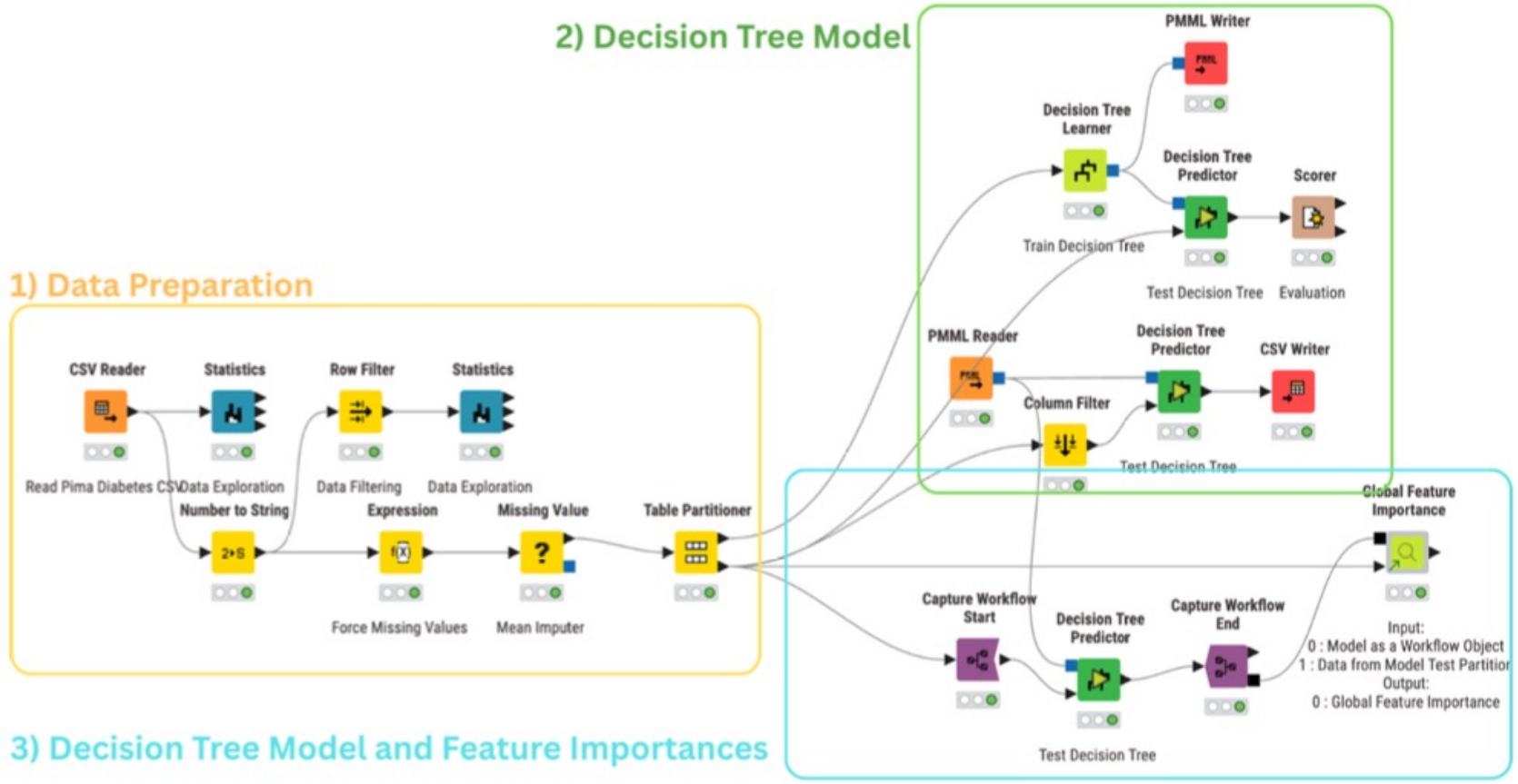
Data Description

- **Dataset:** Pima Indians Diabetes dataset
- **Records:** 768 patients (female, Pima Indian heritage, age ≥ 21)
- **Features (inputs):** Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age
- **Target (label): Outcome** (binary)
    - 1 = diabetes, 0 = not diabetes

What We Will Do in KNIME

- **Load + inspect** the dataset (basic statistics, spot suspicious values like zeros)
- **Prepare the target** (convert Outcome to a nominal class column)
- **Handle "0" medical values** (demonstrate dropping rows vs converting to missing + imputing)
- **Split train/test** (e.g., 70/30, stratified, fixed seed for reproducibility)
- **Train a Decision Tree** classifier and **evaluate** with standard metrics (confusion matrix/ROC via Scorer)
- **Export and reuse the model** (PMML), generate **predictions.csv**, and explore **Global Feature Importance**

LAB 1: Classification



Thank you & any questions