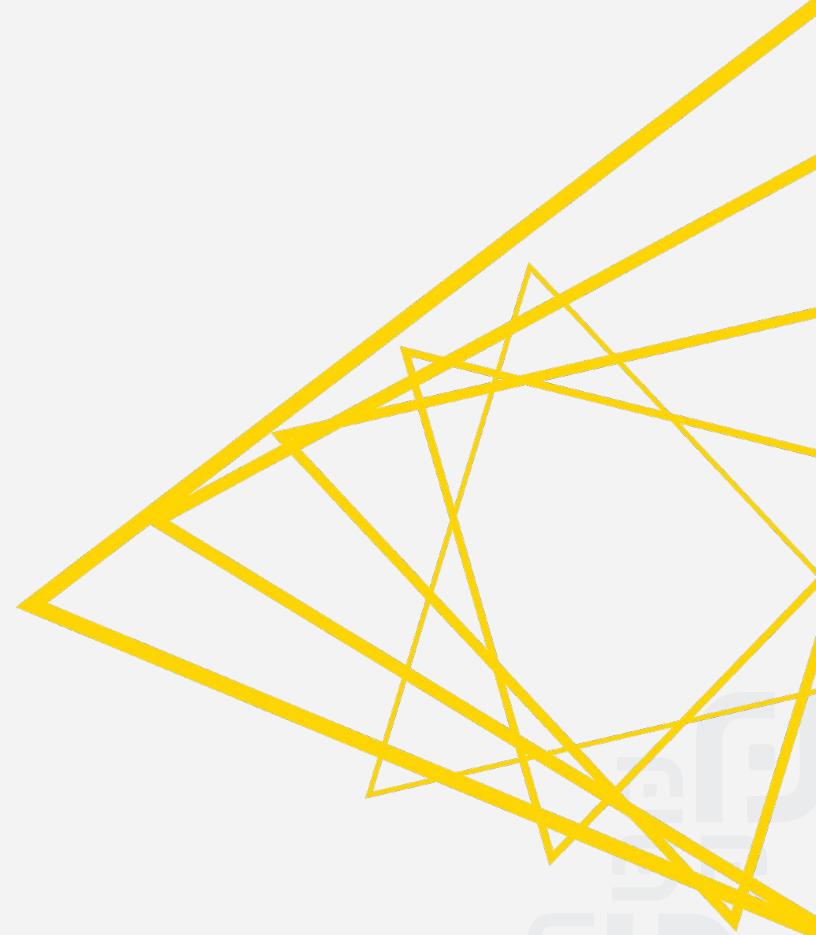


Intro to Machine Learning



Data Mining

Partition, Learn, Predict, Score

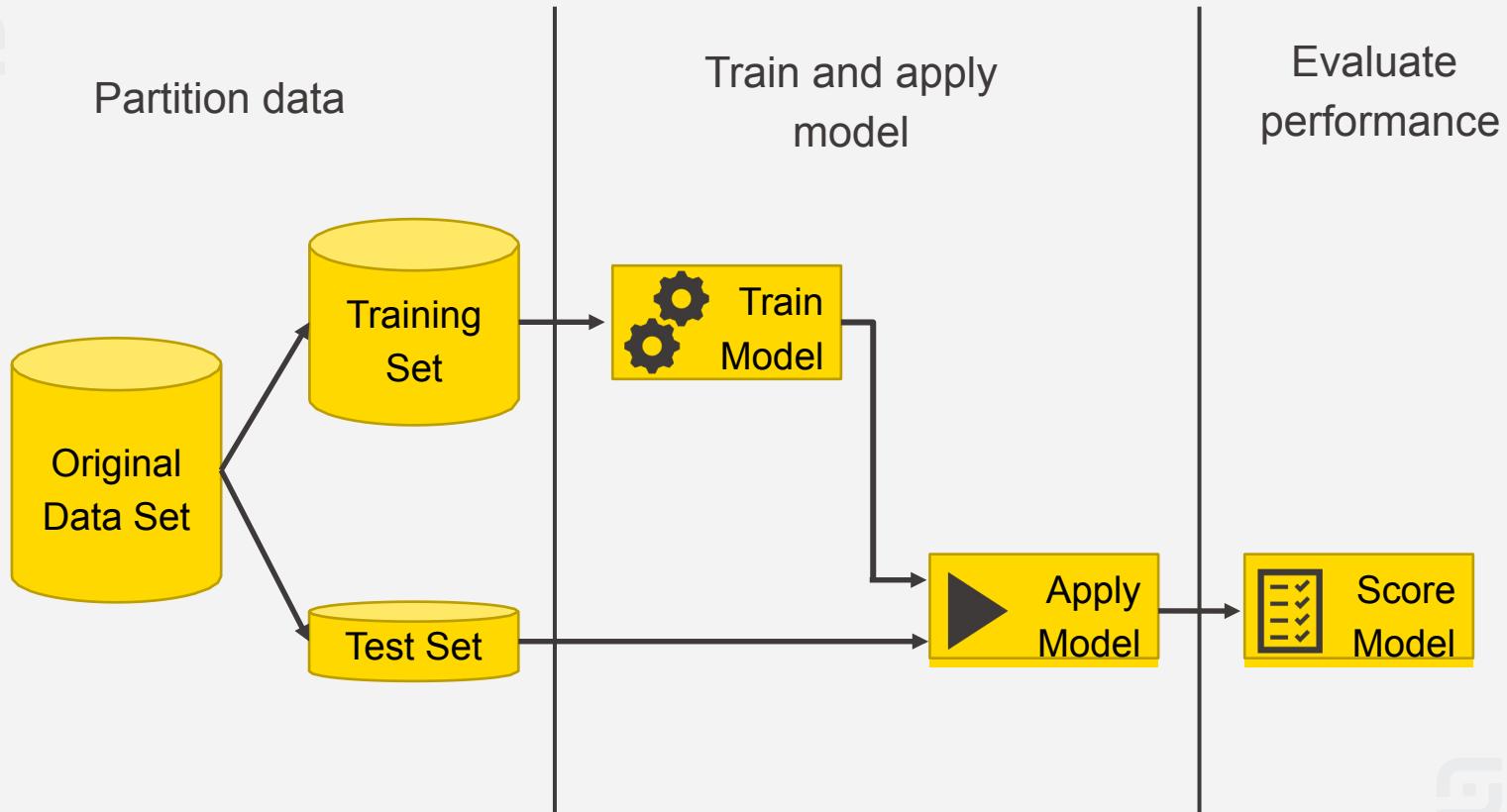


Data Mining Strategies

Example Applications:

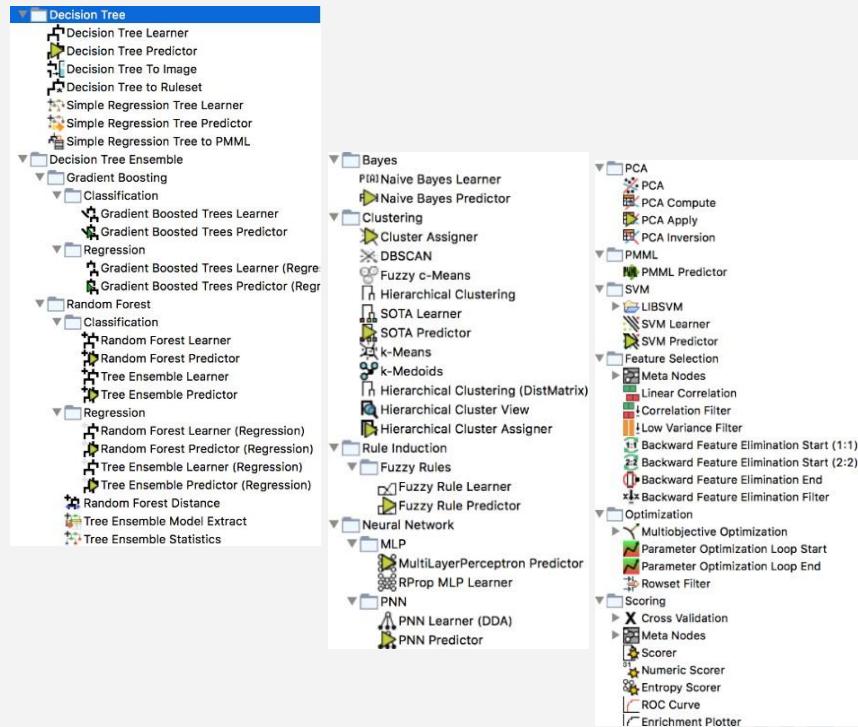
- Anomaly Detection (fraud, predictive maintenance)
- Association Rule Learning (market basket analysis)
- Clustering (market segmentation)
- Classification (next best offer, churn preventions)
- Regression (trend estimation)

Data Mining: Process Overview



Data Mining in KNIME

- KNIME has many modeling tools!
 - Decision tree, random forest, SVM, regression, neural networks, clustering, ...
 - and integrations with other libraries: R, Python, H2O, WEKA, libSVM, etc.
- And many model evaluation nodes
 - ROC, standard, numeric and entropy scorers
 - Feature elimination
 - Cross validation



Churn Prediction



CRM System

Data about your customer

- Demographics
- Behavior
- Revenues



Model

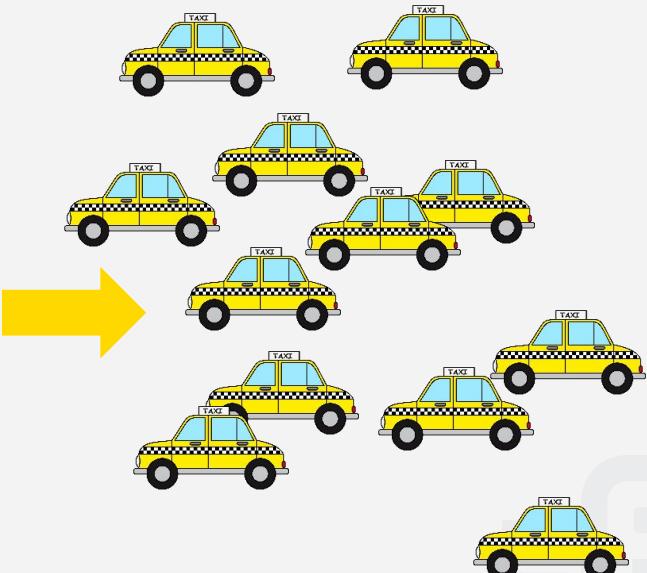


- Churn Prediction
- Upselling Likelihood
- Product Propensity /NBO
- Campaign Management
- Customer Segmentation
- ...

Demand Prediction



How many taxis
do I need in NYC
on Wednesday at
12:00?



Recommendation Engines / Market Basket Analysis



Model

$$\text{Support} = \frac{\text{frq}(X, Y)}{N}$$

Rule: $X \Rightarrow Y$

$$\text{Confidence} = \frac{\text{frq}(X, Y)}{\text{frq}(X)}$$
$$\text{Lift} = \frac{\text{Support}}{\text{Supp}(X) \times \text{Supp}(Y)}$$

Recommendation



IF =>



Sentiment Analysis



Samsung

Samsung Galaxy S7 Edge G935A 32GB Unlocked - Gold Platinum

★★★★★ 125 customer reviews | 606 answered questions

★★★★★ Beautiful phone from a wonderful seller!

By A

May 29, 2017

Color: Gold | Verified Purchase

This practically new beautiful phone well exceeded my expectations!



★★★★★ One Star

By C

on August 3, 2016

Color: Black Onyx | Verified Purchase

Very bad experience



New Node: Partitioning

- Use to split data into training and evaluation sets
 - Partition by count (e.g. 10 rows) or fraction (e.g. 10%)
 - Sample by a variety of methods; random, linear, stratified

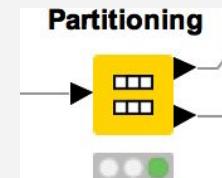
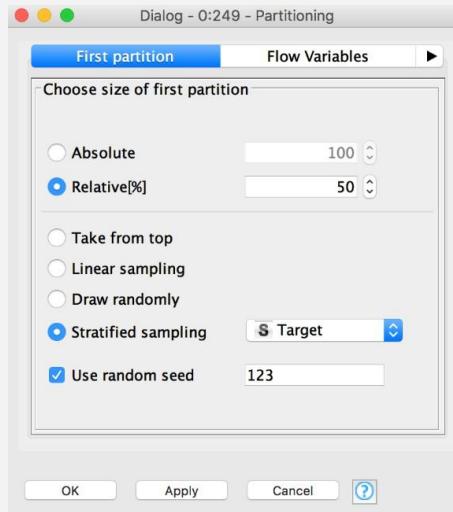
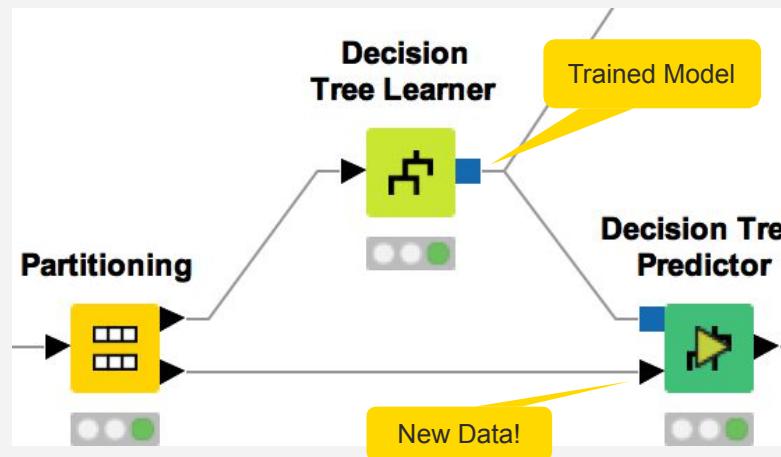


Table "default" - Rows: 5775 Spec - Columns: 13						
Row ID	\$ Marita...	\$ Gender	↓ Estim...	↓ Numb...	↓ Age	File
Row0	M	M	90000	0	44	
Row7	M	M	60000	1	46	
Row9	S	M	70000	1	46	
Row10	S	F	70000	1	46	
Row13	M	M	100000	3	42	
Row14	S	F	100000	3	42	
Row15	S	F	30000	1	31	
Row17	S	F	20000	2	66	
Row18	S	M	30000	2	66	
Row20	S	M	40000	2	32	
Row21	S	M	60000	1	44	

Table "default" - Rows: 5776 Spec - Columns: 13						
Row ID	\$ Marita...	\$ Gender	↓ Estim...	↓ Numb...	↓ Age	File
Row1	S	M	60000	1	45	
Row2	M	M	60000	1	45	
Row3	S	F	70000	1	42	
Row4	S	F	80000	4	42	
Row5	S	M	70000	1	45	
Row6	S	F	70000	1	44	
Row8	S	F	60000	3	46	
Row11	M	M	60000	4	46	
Row12	M	F	100000	2	42	
Row16	M	M	30000	1	31	
Row19	S	M	40000	2	32	
Row22	S	M	60000	1	44	

Learner-Predictor Motif

- Most data mining approaches in KNIME use a Learner-predictor motif.
- The Learner node trains the model with its input data.
- The Predictor node applies the model to a different subset of data.



Classification

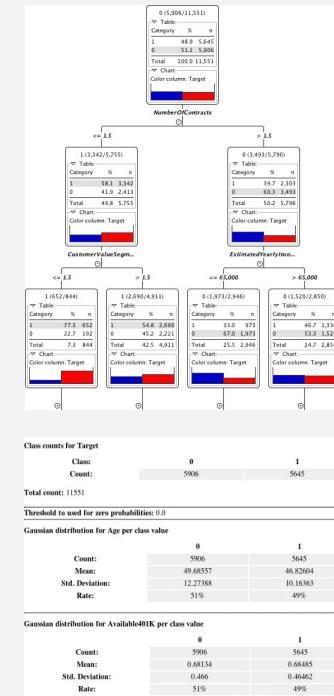
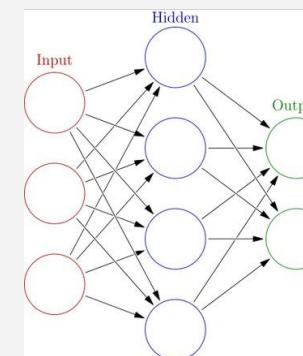
Predict *nominal* outcomes on existing data (supervised)

Applications

- Churn analysis (yes/no)
- Chemical activity (active/inactive)
- Spam detection (spam/not spam)
- Optical character recognition (A-Z)

Methods

- Decision Trees
- Neural Networks
- Naïve Bayes
- Logistic Regression



Target Column

- Target column contains values that are predicted by the classification model
- Binomial target values are often encoded to 1 and 0

Application	Target Column	Target Values
Churn analysis	Churn	Yes/No or 1/0
Chemical activity	Active	Yes/No or 1/0
Spam Detection	Spam	Yes/No or 1/0
Optical Character Recognition	Character	A-Z

Output data - 0:311 - Column Resorter

File Hilite Navigation View

Table "default" - Rows: 5776 Spec - Columns: 17 Properties Flow Variables

R...	CustomerKey	Marital...	Gender	Target	Prediction (Target)
...	11001	S	M	1	0
...	11002	M	M	1	0
...	11003	S	F	1	1
...	11004	S	F	1	0
...	11005	S	M	1	1
...	11006	S	F	1	1
...	11008	S	F	1	0
...	11011	M	M	1	0
...	11012	M	F	0	1
...	11016	M	M	1	1

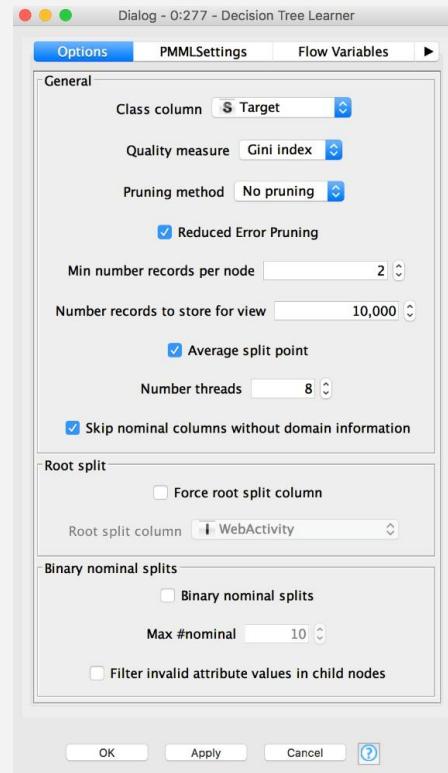
KNIME's Decision Tree

J.R. Quinlan, "C4.5 Programs for machine learning"

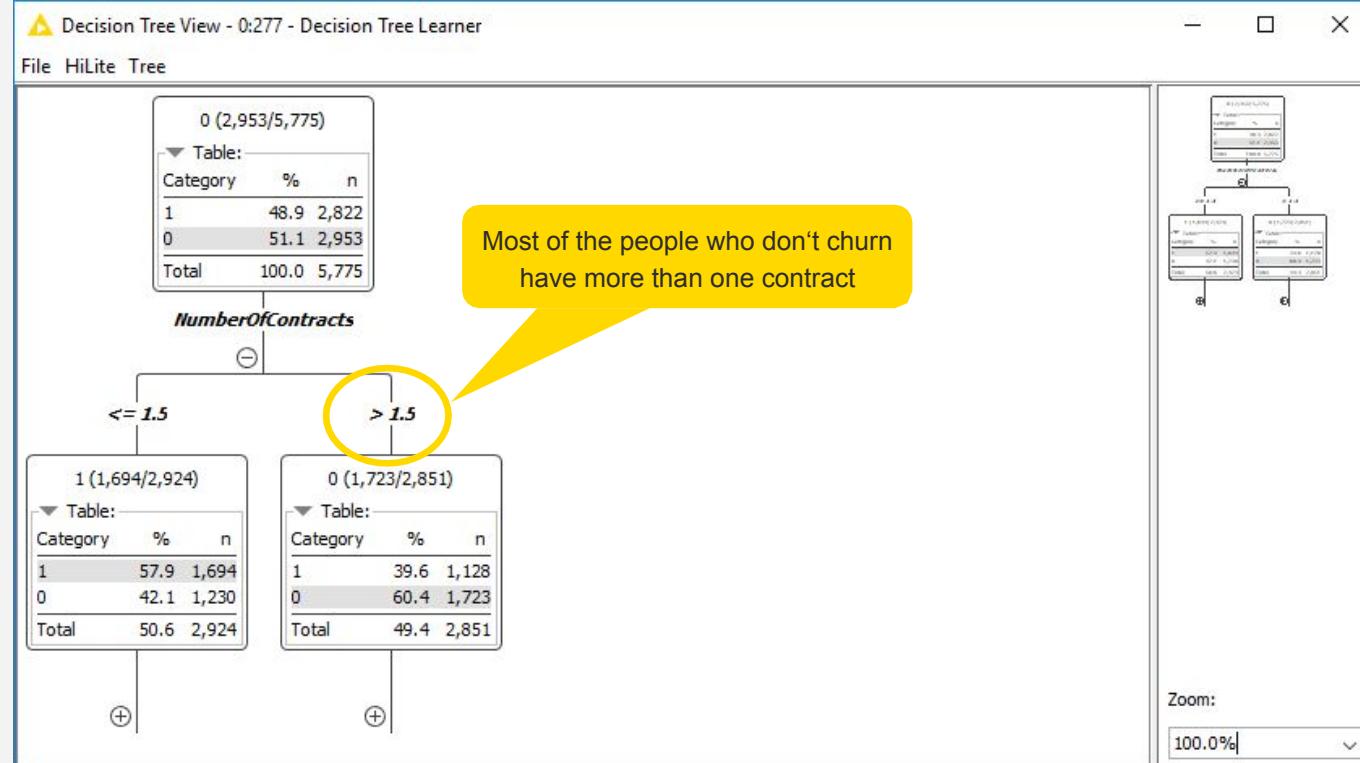
J. Shafer, R. Agrawal, M. Mehta, "SPRINT: A Scalable Parallel Classifier for Data Mining"

- C4.5 builds a tree from a set of training data using the concept of information entropy.
- At each node of the tree, the attribute of the data with the highest **normalized information gain** (difference in entropy) is chosen to split the data.
- The C4.5 algorithm then recurses on the smaller sub lists.

New Node: Decision Tree Learner

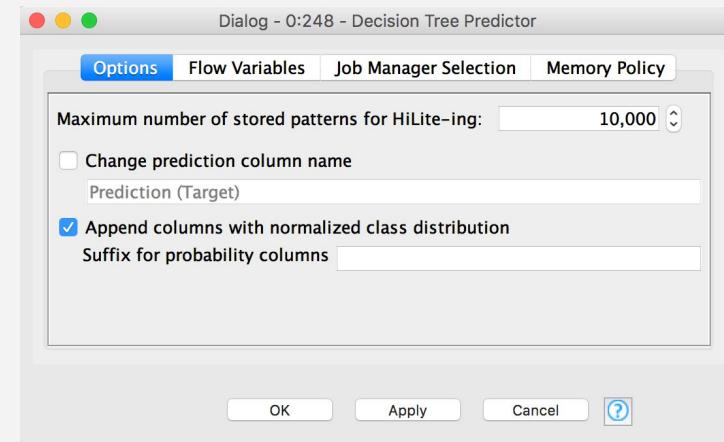


Decision Tree View



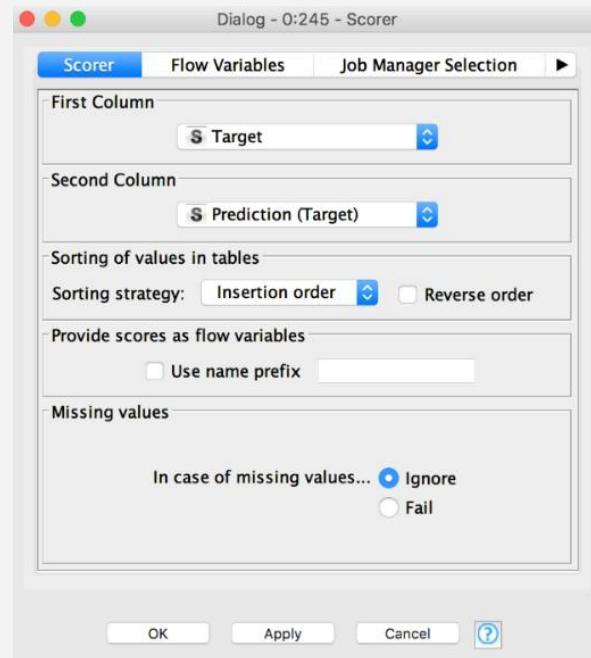
New Node: Decision Tree Predictor

- Takes a decision tree model & applies it to new data
- Check the box to append class probabilities



New Node: Scorer

Compare predicted results to known truth
in order to evaluate model quality



New Node: Scorer

- Confusion matrix shows the distribution of model errors
- An accuracy statistics table provides a detailed analysis of model quality

Confusion Matrix - 0:297 - Scorer		
File	Hilite	
Target \ Prediction (Target)	1	0
1	2073	750
0	759	2193

Correct classified: 4,266 Wrong classified: 1,509
Accuracy: 73.87 % Error: 26.13 %
Cohen's kappa (κ) 0.477

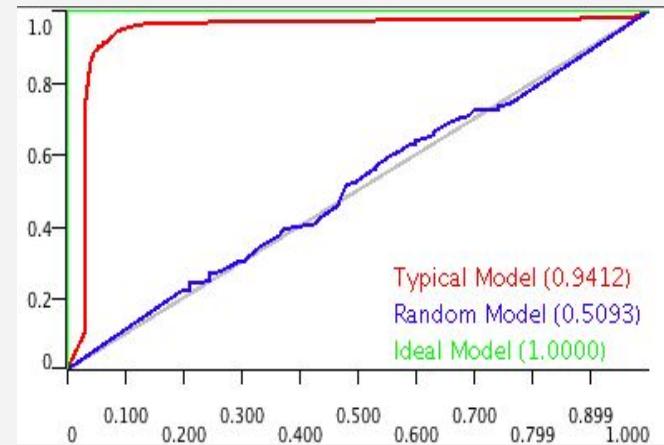
Accuracy statistics - 0:297 - Scorer												
File	Hilite	Navigation	View	Table "default" – Rows: 3			Spec – Columns: 11	Properties	Flow Variables			
Row ID	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa	
1	2073	759	2193	750	0.734	0.732	0.734	0.743	0.733	?	?	
0	2193	750	2073	759	0.743	0.745	0.743	0.734	0.744	?	?	
Overall	?	?	?	?	?	?	?	?	?	0.739	0.477	

Confusion Matrix

	Predicted class POSITIVE (churn)	Predicted class NEGATIVE (no churn)
Actual class POSITIVE (churn)	TRUE POSITIVE (TP) 2073	FALSE NEGATIVE (FN) 750
Actual class NEGATIVE (no churn)	FALSE POSITIVE (FP) 759	TRUE NEGATIVE (TN) 2193

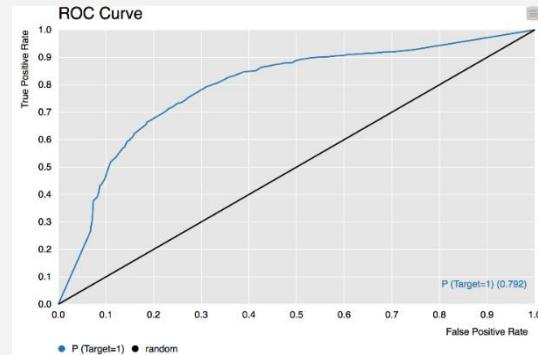
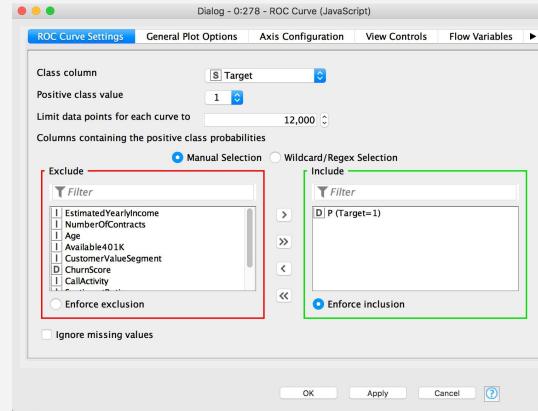
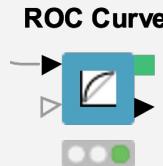
Receiver Operating Characteristics

- Sort by confidence in target class
- Plot true positive rate vs false positive rate
- Ideal models achieve 100% TPR with 0% FPR
- Area under the curve indicates model quality
 - (1=ideal model, 0.5 = random outcome)



New Node: ROC Curve

- Requires individual class probabilities from a preceding predictor
- User must define:
 - Original class column
 - Positive class value
 - Probability for the selected positive class value for one or multiple models



Data Mining Exercise, Activity I

Start with exercise: *Data Mining, Activity I*:

- Partition the fully joined data into a training and test set (50%, Stratified Sampling on Target)
- Train a decision tree on the training set to predict Target
- Use the trained model to predict Target in the test set
- What is the overall accuracy of your model?
- Optional: Evaluate the accuracy and robustness of the model with the ROC Curve node

Regression

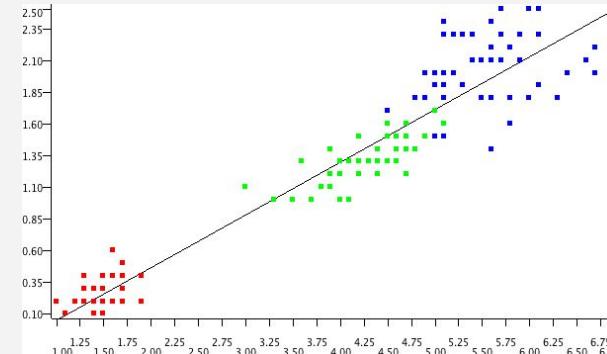
Predict *numeric* outcomes on existing data (supervised)

■ Applications

- Forecasting
- Quantitative Analysis

■ Methods

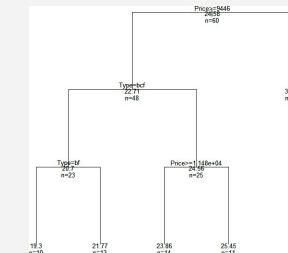
- Linear
- Polynomial
- Regression Trees
- Partial Least Squares



Statistics on Linear Regression

Variable	Coeff.	Std. Err.	t-value	P> t
Petal.Length	0.4158	0.0096	43.3872	0.0
Intercept	-0.3631	0.0398	-9.1312	4.44E-16

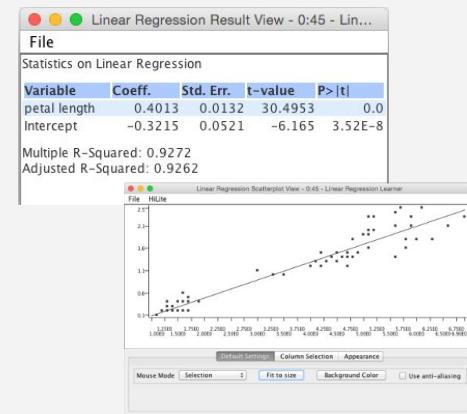
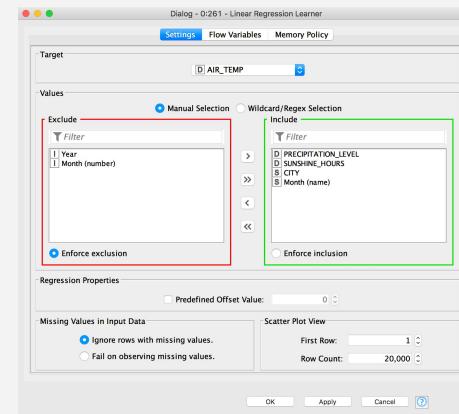
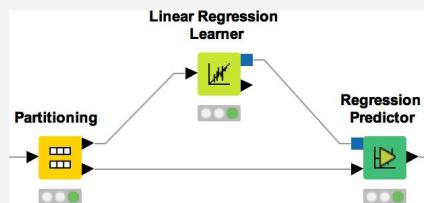
Multiple R-Squared: 0.9271
Adjusted R-Squared: 0.9266



New Nodes: Linear Regression Learner & Regression Predictor

A linear model relating a dependent variable to 1 or more independent variables

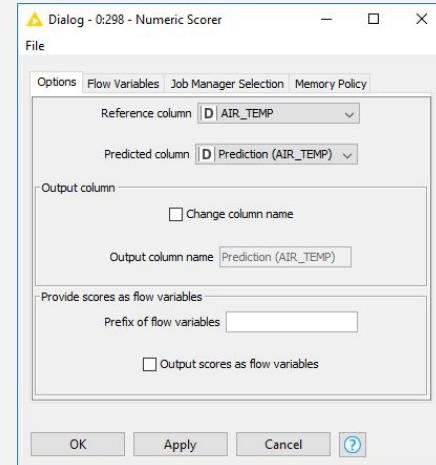
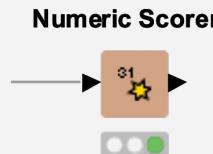
- Model coefficients provided in 2nd output port
- Also available: Polynomial and Tree Ensemble Regression nodes



New Node: Numeric Scorer

Similar to scorer node, but for nodes with *numeric* predictions (e.g. linear/polynomial regression)

- Compare dependent variable values to predicted values to evaluate goodness of fit.
- Report R², MAE, MSE, RMSE etc.



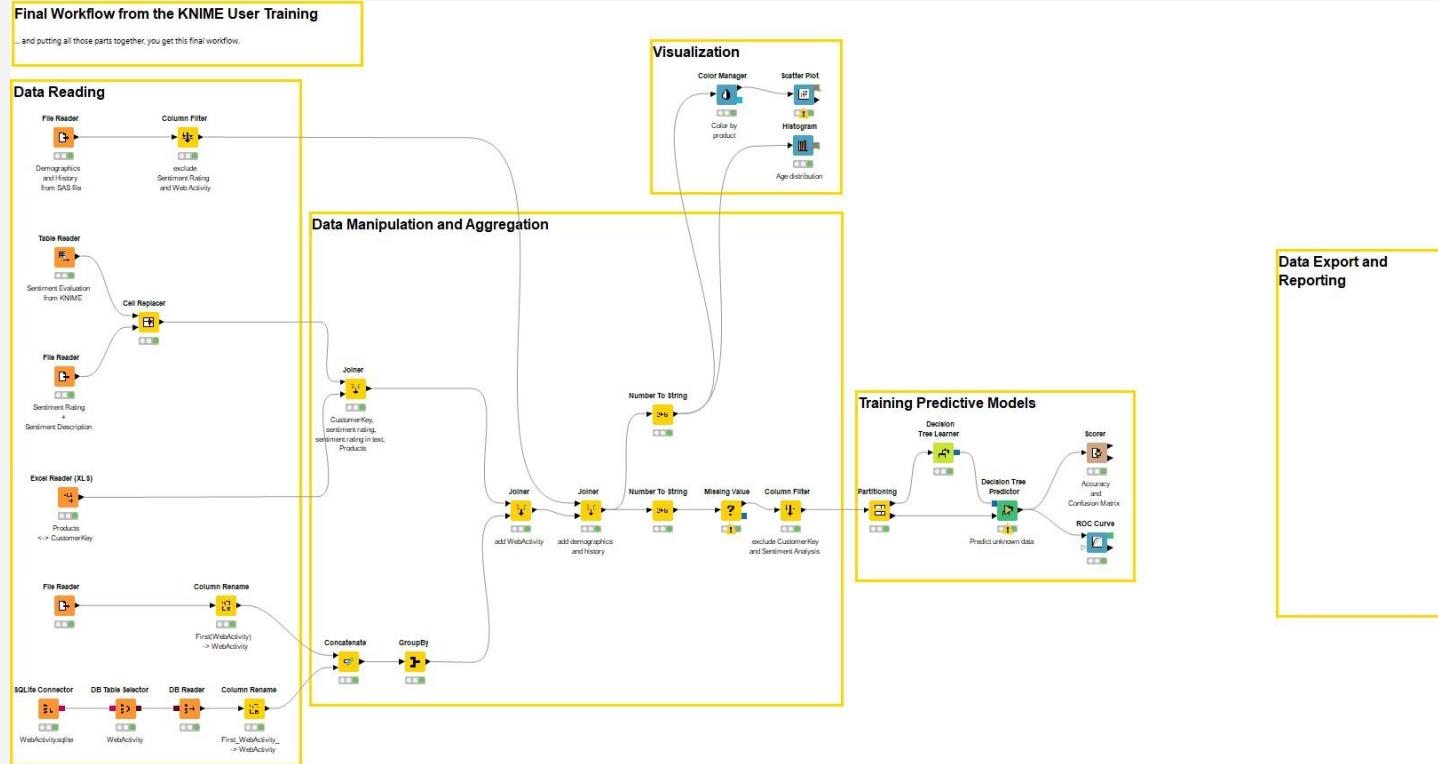
This screenshot displays the results table generated by the Numeric Scorer node. The table, titled "Scores", contains the following data:

Row ID	D Prediction (AIR_TEMP)
R^2	0.333
mean absolute error	3.574
mean squared error	21.329
root mean squared error	4.618
mean signed difference	1.048
mean absolute percentage error	NaN

Data Mining Exercise, Activity II

- Start with exercise: *Data Mining, Activity II*:
- Read *weather.table* data
- Split the data into rows up to 2016 (training set) and rows from 2017 on (test set)
- Train a linear regression model that predicts the AIR_TEMP as a function of all other features in the dataset
- Use the model to predict the temperature in 2017 and evaluate the model with the Numeric Scorer node
- Optional:
 - Calculate the mean temperature per month in the training data
 - Join the mean temperature per month to the test set
 - Use the Numeric Scorer to see if the average monthly temperature provides a better prediction than the Linear Regression model

Today's Example



Clustering

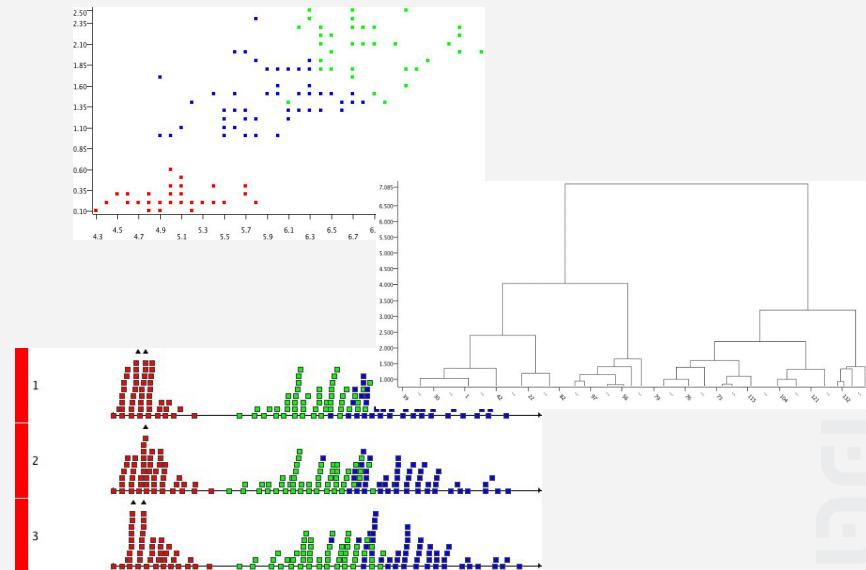
Discover hidden structure in **unlabeled** data (unsupervised)

■ Applications

- Market Segmentation
- Diversity picking

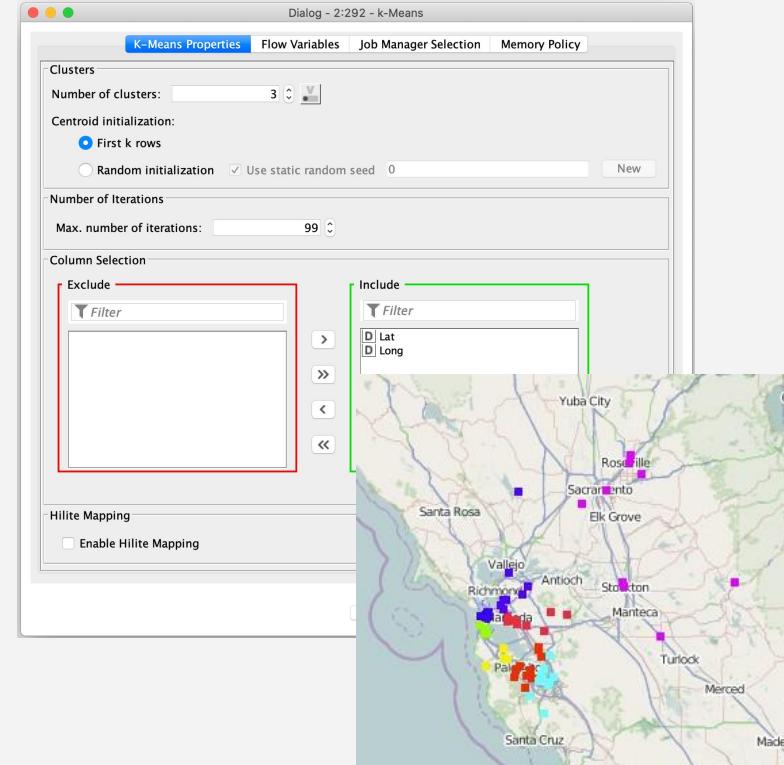
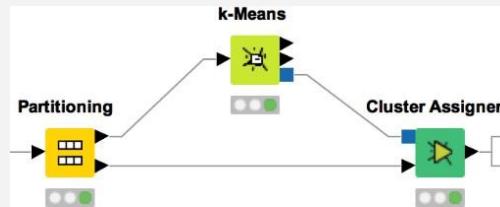
■ Methods

- K-means/medoids
- Hierarchical
- DBScan
- OPTICS
- Neighbourgrams



New Nodes: k-Means Clustering

- Looks at n observations to define the means for k clusters.
- Each observation is then assigned to its closest cluster center.
- You must provide k.



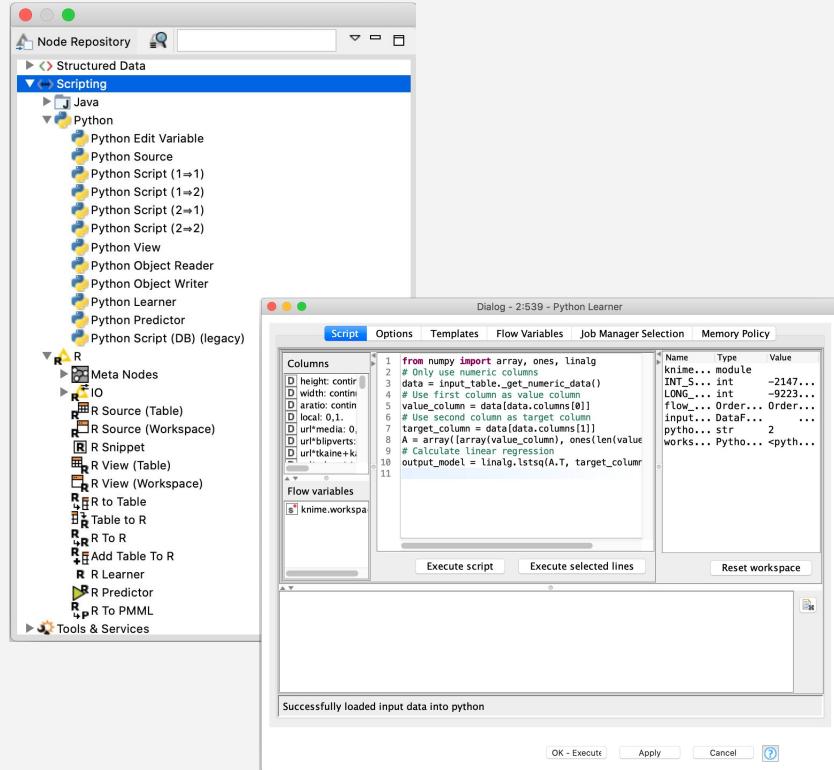
Data Mining Exercise, Activity III

Start with exercise: *Data Mining, Activity III*

- Read *location_data.table* data
- Filter the data to entries from California (region_code = CA)
- Perform k-means clustering with k=3. Use only latitude and longitude for clustering.
- Optional: plot latitude and longitude in a view (OSM Map or Scatter Plot) and use the view to visually optimize k

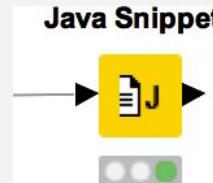
Scripting Integrations: R and Python

- Run R or Python code in KNIME Analytics Platform
- Works with existing Python and R installations
- Syntax highlighting support
- Different nodes for many tasks, e.g training a model using an algorithm available in Python



Java Snippet

- Fastest running scripting node in KNIME
- Syntax highlighting, auto completion, error checking
- Templates allow you to save scripts for later re-use
- Import custom libraries

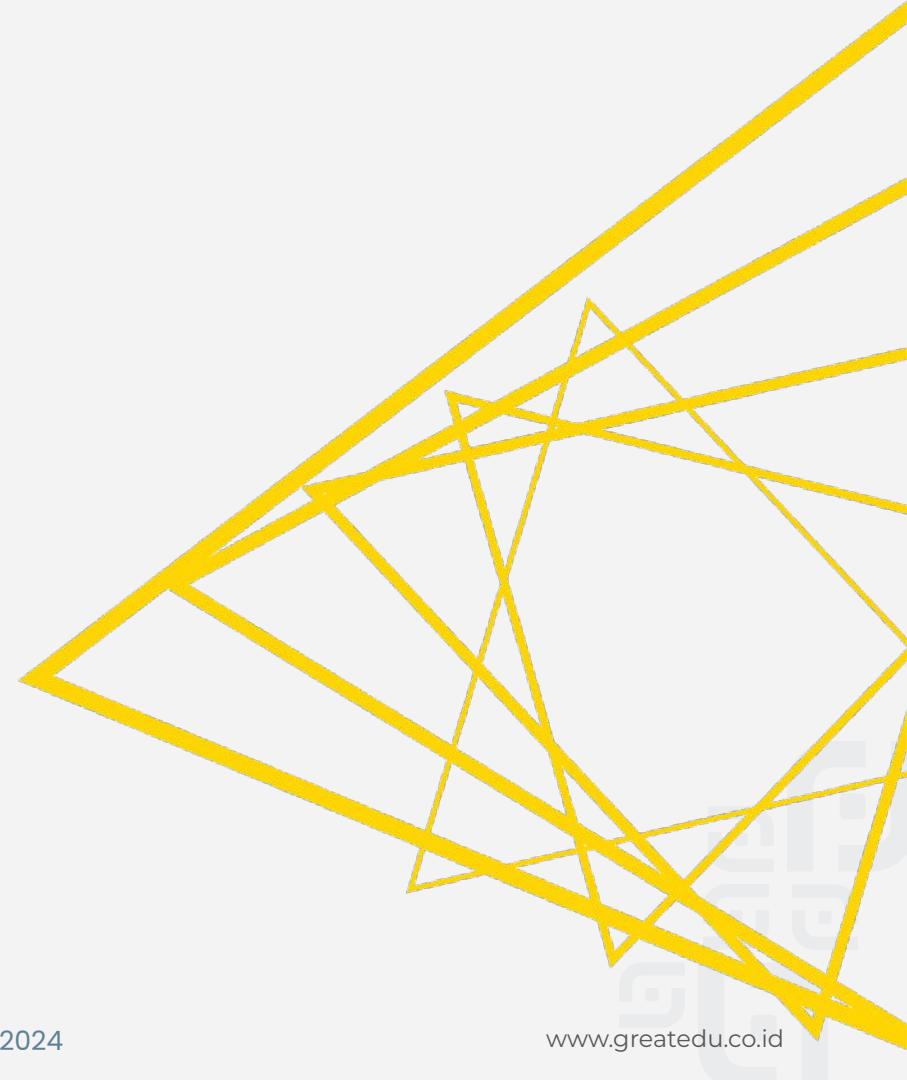


A screenshot of the 'Java Snippet' dialog box. The code editor contains the following Java code:

```
1// system imports
2
3// Your custom imports:
4
5// system variables
6// Your custom variables:
7
8// expression start
9
10// Enter your code here:
11
12try {
13    Thread.sleep(100);
14} catch (Exception e) {
15    // do nothing
16}
17
18
19
20
21// expression end
22
23
24
25
26
27
28
29
30
31
32
33
34
35
```

The dialog also shows sections for 'Column List', 'Flow Variable List', 'Input', and 'Output'.

Advanced Data Mining

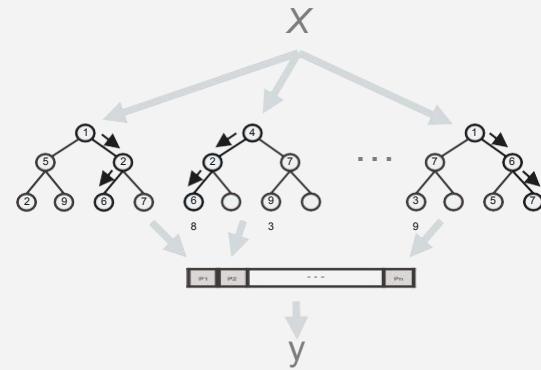


Overview

- Ensemble models
 - Random Forest / Tree Ensembles
 - Gradient Boosted Trees
- Parameter optimization
- Cross validation
- H2O and Keras integration in KNIME

KNIME's Tree Ensemble Models

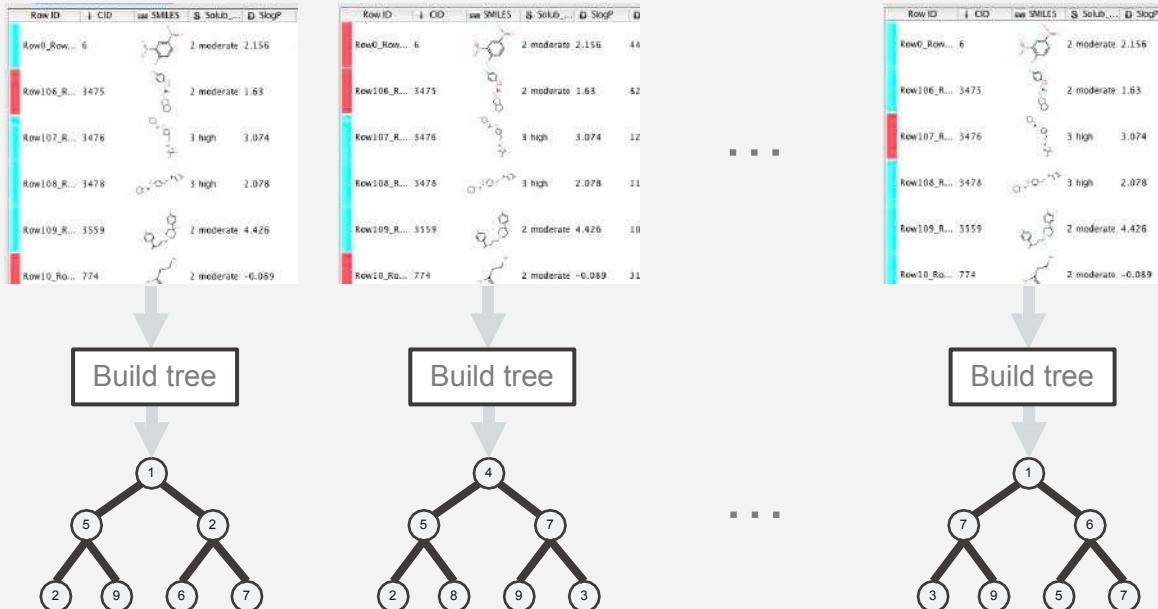
- The general idea is to take advantage of the “wisdom of the crowd”
- Ensemble models: Combining predictions from a large number of weak predictors, e.g. decision trees
- Leads to a more accurate and robust model
- This is called “bagging”



Typically: for classification the individual models vote and the majority wins; for regression, the individual predictions are averaged

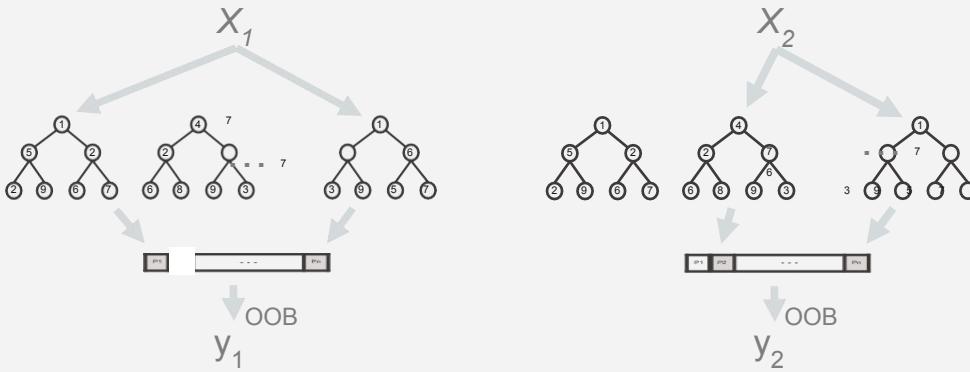
How Does Bagging Work?

- Pick a different random subset of the training data for each model in the ensemble (bag)



An Extra Benefit of Bagging: Out of Bag Estimation

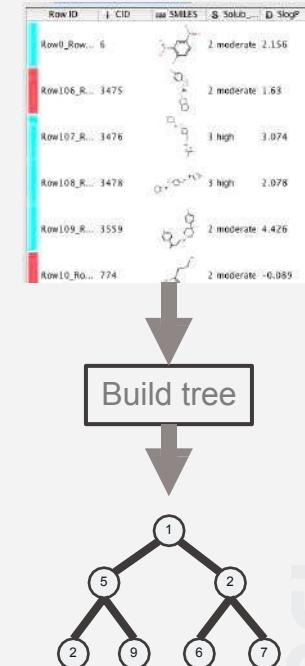
- Allows testing the model using the training data: when validating, each model should only vote on data points that were not used to train it



Row ID	S	State	P (Churn=0)	P (Churn=1)	Churn (Out-of-bag)	Churn (Out-of-bag)	model count
Row1_Row0	S	0.943	0.057	0	0.943	0.943	35
Row2_Row1	IH	1	0	0	1	1	33
Row3_Row2	U	1	0	0	1	1	37
Row4_Row3	IH	0.528	0.472	0	0.528	0.528	36
Row5_Row4	IK	0.976	0.024	0	0.976	0.976	41
Row6_Row5	L	0.848	0.152	0	0.848	0.848	33
Row7_Row6	IA	0.833	0.167	0	0.833	0.833	36
Row9_Row8	A	0.667	0.333	0	0.667	0.667	30
Row11_Row9	Y	0.138	0.862	1	0.862	0.862	29
Row13_Row10	Z	0.974	0.026	0	0.974	0.974	39
Row14_Row11	IT	0.917	0.083	0	0.917	0.917	36
Row15_Row12	I	0.387	0.613	1	0.613	0.613	31
Row18_Row13	T	0.974	0.026	0	0.974	0.974	39
Row19_Row14	A	1	0	0	1	1	38
Row21_Row15	L	0.971	0.029	0	0.971	0.971	34
Row22_Row16	O	0.03	0.97	1	0.97	0.97	33
Row23_Row17	Z	0.854	0.146	0	0.854	0.854	41
Row25_Row18	A	0.973	0.027	0	0.973	0.973	37
Row26_Row19	IE	0.886	0.114	0	0.886	0.886	35
Row27_Row20	VY	0.912	0.088	0	0.912	0.912	34
Row28_Row21	IT	0.976	0.024	0	0.976	0.976	42
Row29_Row22	IO	1	0	0	1	1	42
Row30_Row23	II	1	0	0	1	1	40
Row32_Row24	IH	0.914	0.086	0	0.914	0.914	35
Row33_Row25	A	0.875	0.125	0	0.875	0.875	32

Random Forest

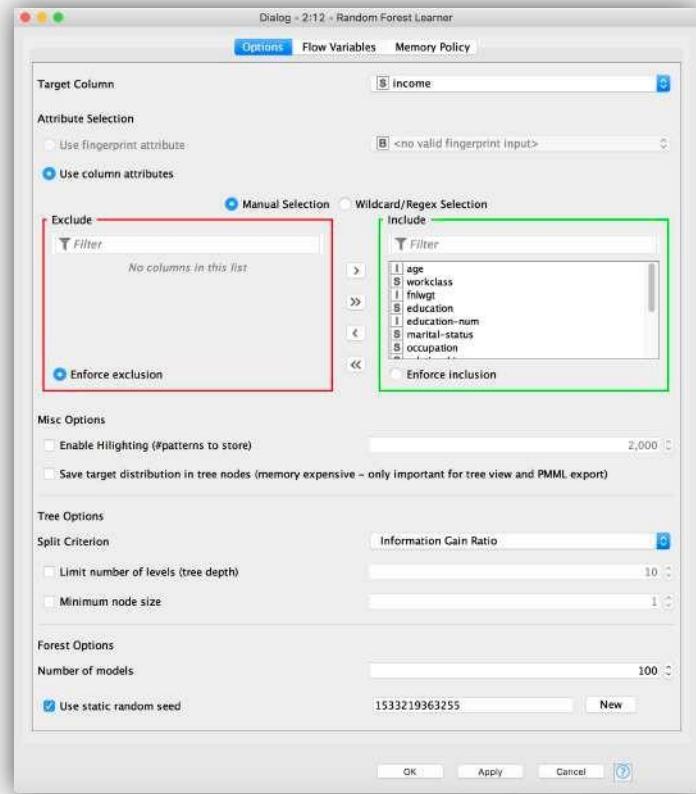
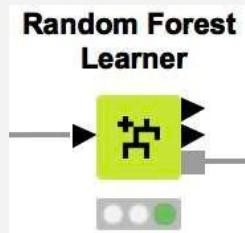
- Train a bag of decision trees
- For each tree / model a training set is generated by sampling uniformly with replacement from the standard training set
- An extra element of randomization is used when building the trees : **each node** in the decision tree only “sees” **a subset of the input columns**, typically N
- Random forests tend to be very robust w.r.t. overfitting (though the individual trees are almost certainly overfit)
- Extra benefit: training tends to be much faster



Random Forest

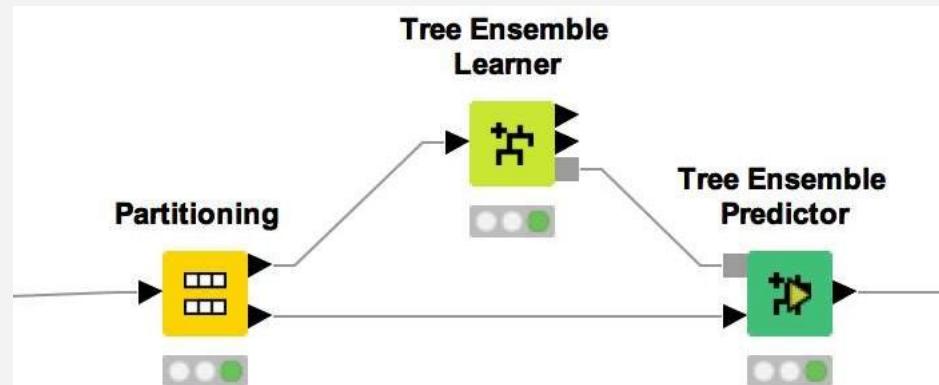
Learner

- The output model describes a random forest and is applied in the corresponding predictor node using a simple majority vote
- The statistics table on the attributes tells how often each attribute...
 - ... is used in the first three splits
 - ... was a possible candidate in the first three splits



Tree Ensembles

- Random Forest is a specific tree ensemble with predefined ensemble parameters
- The Tree Ensemble Learner node allows to train different tree ensembles
 - E.g. different row and column sampling options
- Optimization of a tree ensemble is complex due to a surplus of configuration options
 - Number of models
 - Number of columns
 - Number of rows
 - Tree depth
 - ...



Tree Ensemble Learner

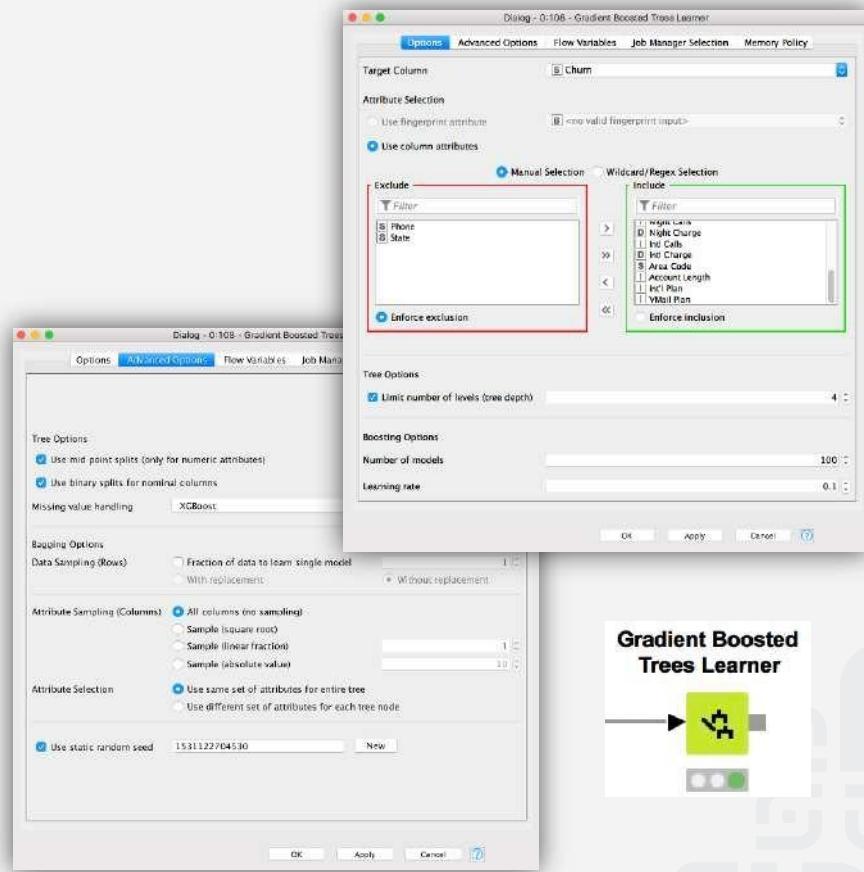
- Choose which columns to include
- Configure a prototype tree (depth, split criteria etc.)
- Setup ensemble parameters (model count, row/column subsampling)

The image displays three side-by-side screenshots of the "Dialog - 0.40 - Tree Ensemble Learner (deprecated)" interface.

- Attribute Selection Tab:** Shows the "Target Column" section with "Use column attributes" selected. The "Include" dropdown is set to "Information Gain Ratio". The "Exclude" section has a red border around it, indicating it is active. The "Include" section shows a list of attributes: MaritalStatus, Gender, EstimatedYearlyIncome, NumberofContracts, Age, AvailableCredit, CustomerValueSegment, and ChurnScore. A green border highlights the "Enforce inclusion" button at the bottom of the list.
- Tree Options Tab:** Shows the "Split Criterion" dropdown set to "Information Gain Ratio". Other options include "Use mid point splits (only for numeric attributes)" and "Use binary splits for nominal columns".
- Ensemble Configuration Tab:** Shows the "Number of models" input field set to 100. The "Data Sampling (Rows)" section includes "Fraction of data to learn single model" (set to 1), "Data Sampling Mode" (set to "With replacement"), and "Random" (radio button selected). The "Attribute Sampling (Columns)" section includes "All columns (no sampling)" and "Sample (square root)" (set to 1). The "Attribute Selection" section includes "Use same set of attributes for entire tree" and "Use different set of attributes for each tree node" (radio button selected).

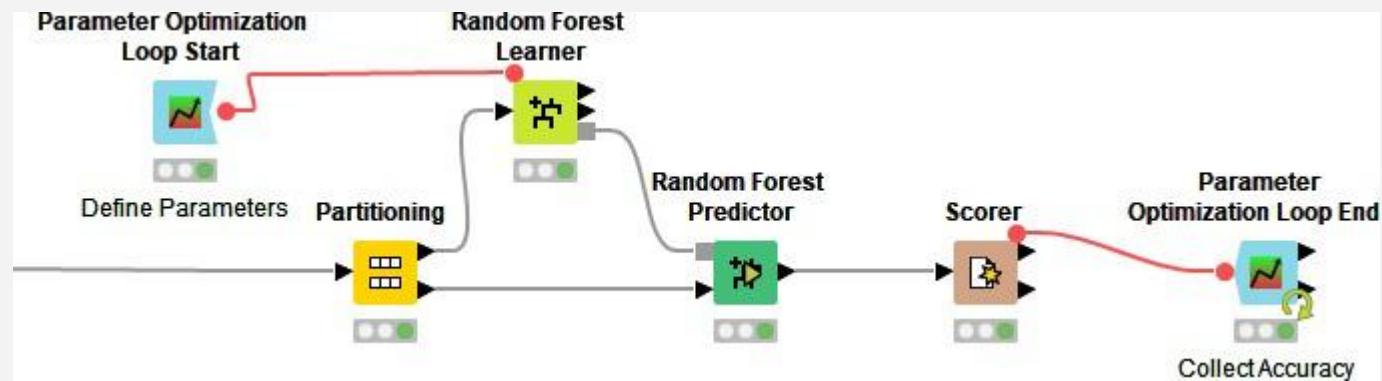
Gradient Boosted Trees Learner

- Another algorithm for creating ensembles of decision trees
- Starts with a shallow tree
- Builds additional trees to fit the residual errors
- Can introduce randomness in choice of data subsets (“stochastic gradient boosting”) and in variable choice (Advanced Options)



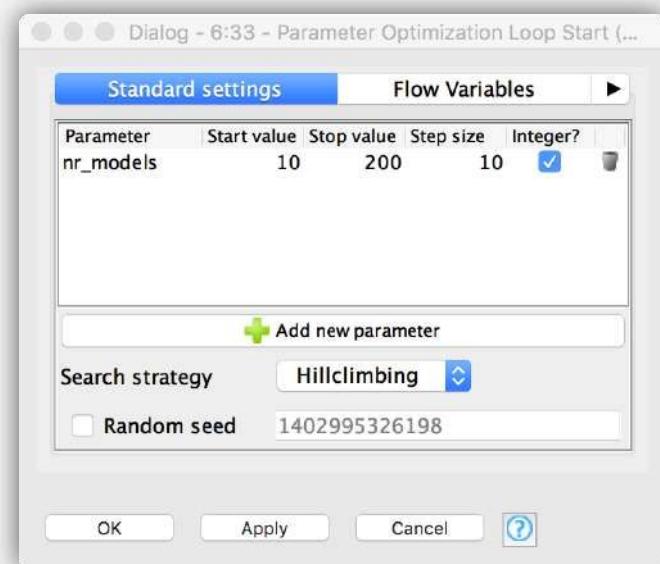
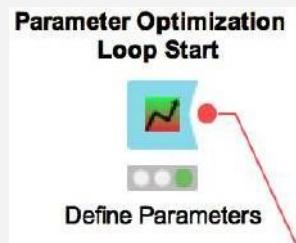
Parameter Optimization

- Some modeling approaches are very sensitive to their configuration.
- Calculating optimum settings is not always possible.
- Parameter Optimization loops may help find a good configuration.



Parameter Optimization Loop Start

- Define parameters to optimize
- Set upper/lower bounds and step sizes (and flag integers)
- Choose an optimization method
 - Brute force for maximum accuracy, but slower computation
 - Hillclimbing for better faster runtimes, but may get stuck in local optimum settings
 - Random search to randomly search for parameter values within a given range
 - Bayesian Optimization (TPE)



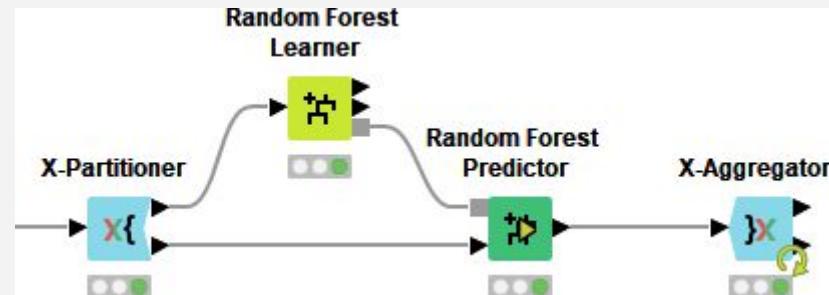
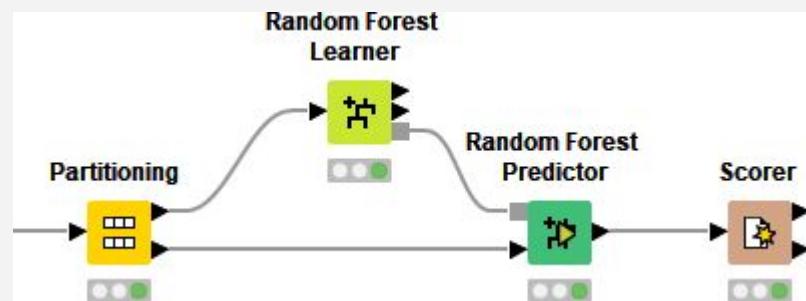
Parameter Optimization Loop End

- Collects a value to optimize as Flow Variable.
- Value may be maximized (accuracy) or minimized (error)



Cross Validation

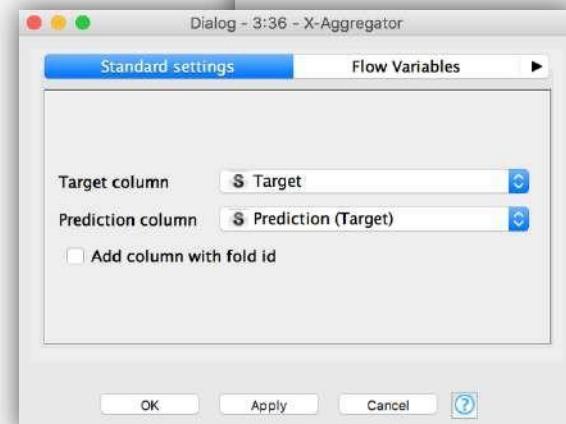
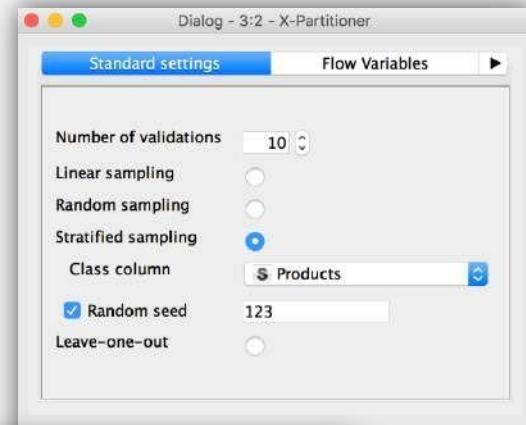
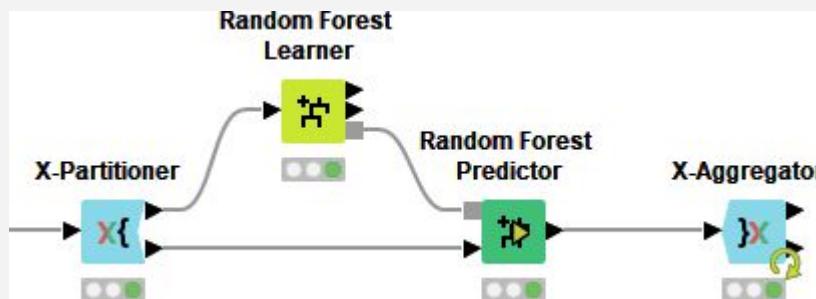
- Used to evaluate model stability
- Re-execute the modeling process many times using different data partitions
- Collect aggregated statistics on model accuracy



Example: Cross Validation

X-Partitioner □ X-Aggregator

- X-Partitioner replaces Partition
- X-Aggregator replaces Scorer
- Can be used with any learner/predictor

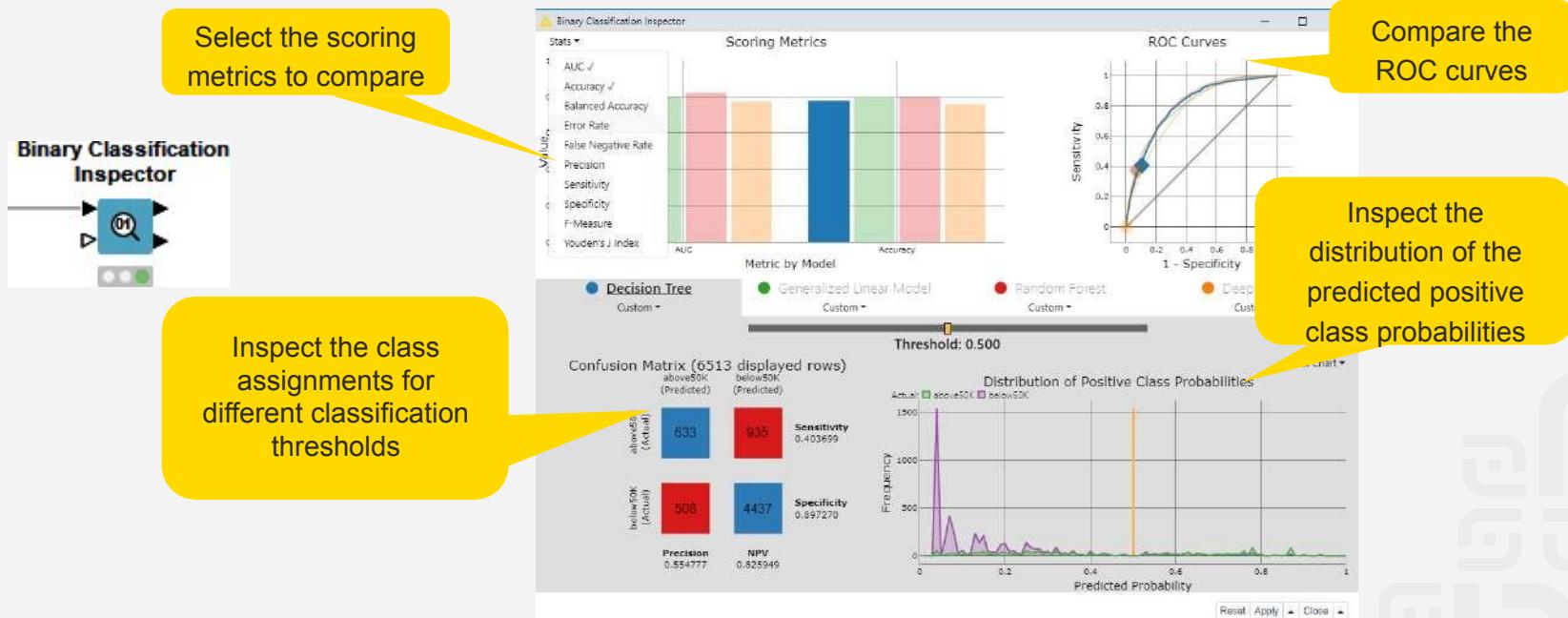


Binary Classification

Inspector

Inspect and compare the performances of classification models

- Adjust the classification threshold according to the goal of your model



Advanced Data Mining Exercise

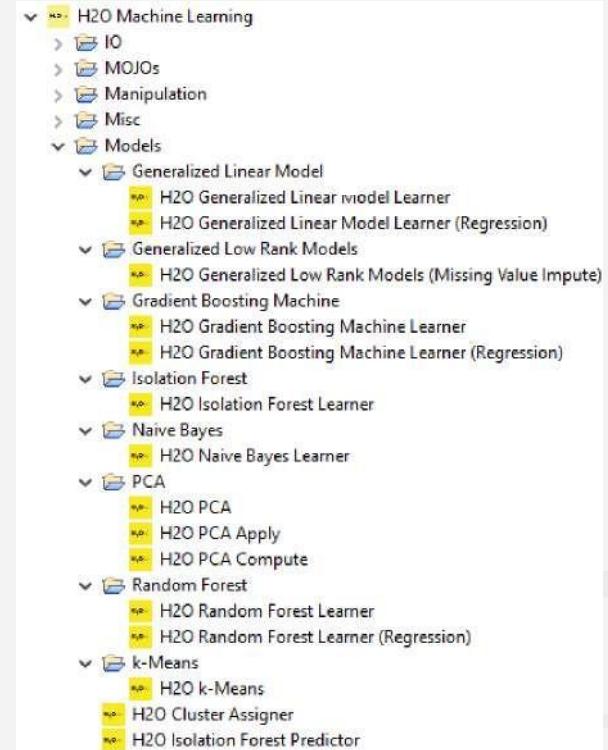
Start with exercise: *Advanced Data Mining*

Goal: Train a decision tree and a random forest model and compare their performance

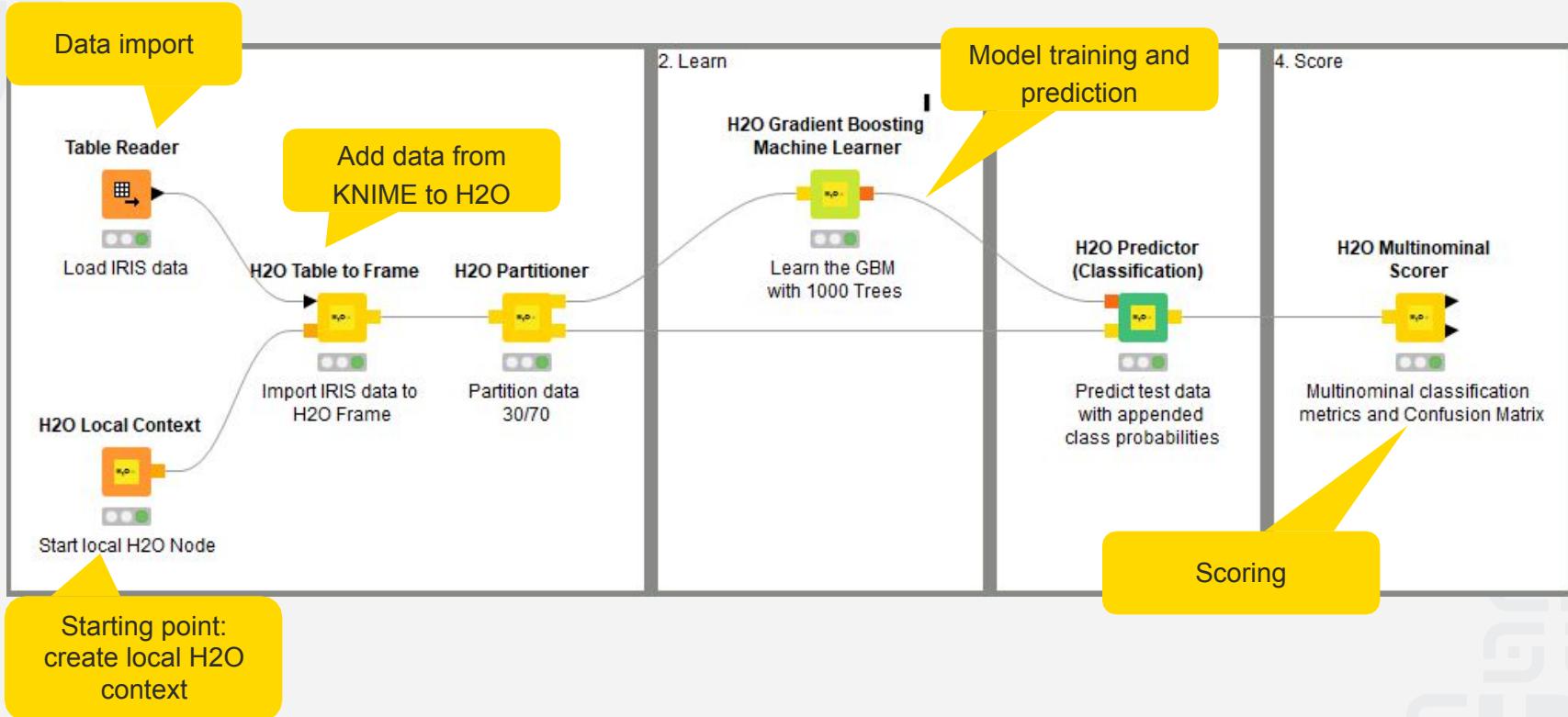
- Partition the data 50/50 using stratified sampling on the "Target" column
- Train and apply a Random Forest model to predict the "Target" column
- Train and apply a Decision Tree model to predict the "Target" column
- Combine the performances of both models (Column Appender node)
- Evaluate the performances of the models (Binary Classification Inspector node)
Which model performs better?

H2O Integration

- KNIME integrates the H2O machine learning library
- H2O: Open source, focus on scalability and performance
- Supports many different models
 - Generalized Linear Model
 - Gradient Boosting Machine
 - Random Forest
 - k-Means, PCA, Naive Bayes, Isolation Forest, etc. and more to come!
- Includes support for MOJO model objects for deployment



H2O Integration - Example

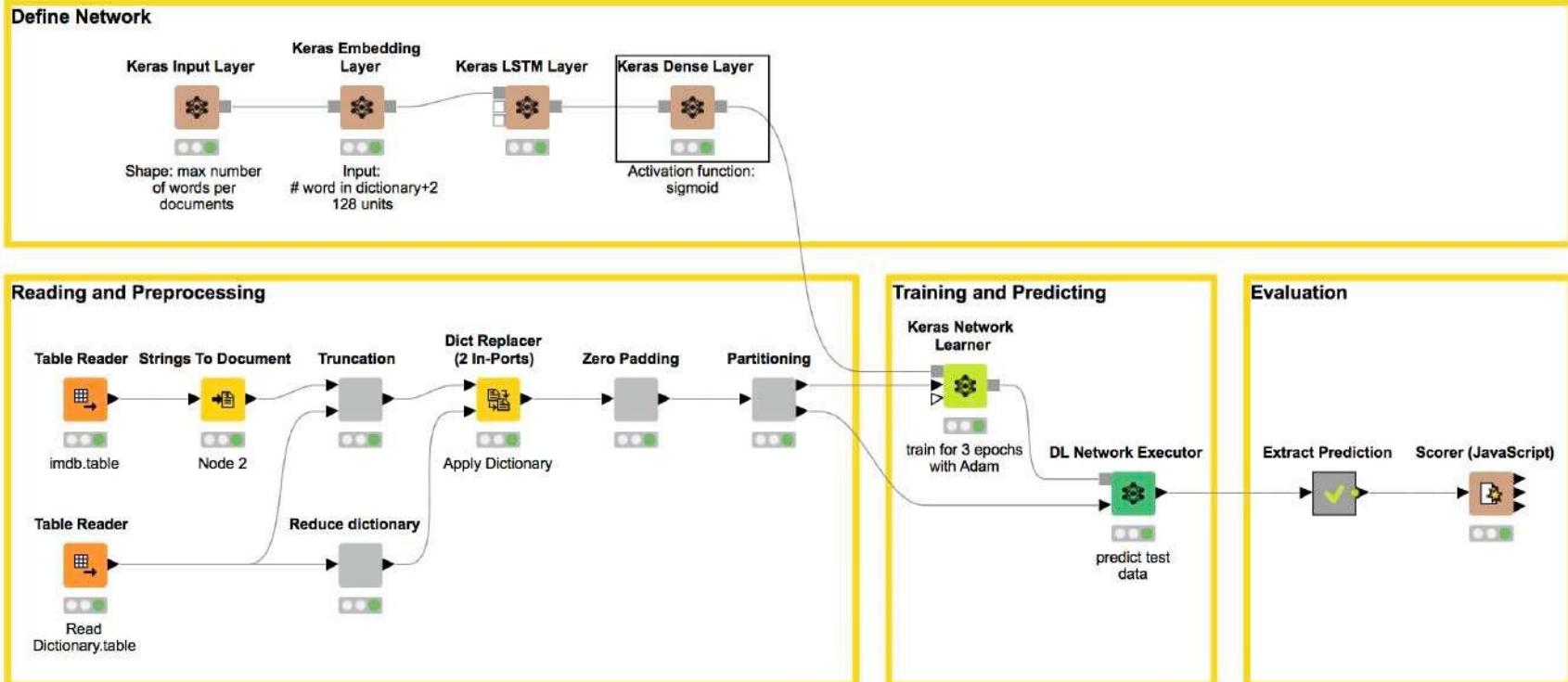


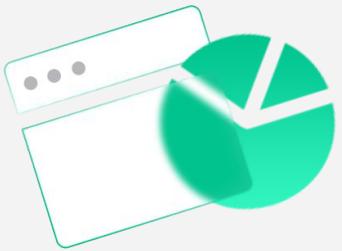
Deep Learning Integration

- Keras integration:
 - Many different layer nodes.
 - Define your network, train and apply a network without a single line of code.
- DL Python integration
- TensorFlow integration



Sentiment Analysis Using Keras





Terima Kasih

SIB Cycle 6 | 2024



www.greatedu.co.id