

Machine Learning using Orange

Xiaojuan Zhu(Julia)

The University of Tennessee, Knoxville
Office of Innovative Technologies
2309 Kingston Pike, KPB Suite 132
Knoxville, TN 37996

HelpDesk: 974-9900

<http://oit.utk.edu/research>



THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

RCS Support

- RCS typically helps UT researchers with:
 - Statistical consulting
 - Research design and implementation
 - Data collection strategies including; developing interview guides, and survey construction
 - Data analysis using qualitative, quantitative, and GIS methods
 - Provide specialized research software support services including:
 - Installation and licensing
 - Akindi exam scoring
- Call OIT helpdesk: 865-974-9900

What is Orange

- Orange is a data mining software created and developed by the Bioinformatics Laboratory at the University of Ljubljana (not a commercial software).
- It can be used for developing and testing machine learning models as well as conducting exploratory data analysis and visualization.
- Orange's components are widget-based and allow user drag and drop to a canvas.
- You can download and install Orange from the link here, <https://orangedatamining.com/download/>

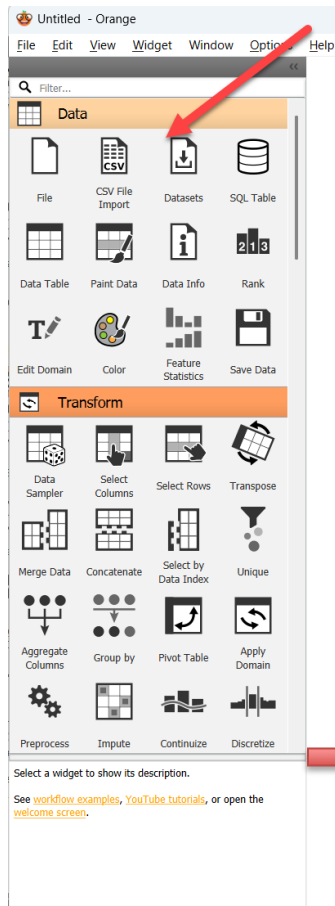
Why is Orange?

- It is easy to implement the machine learning models.
- It is rich visualizations and interactive models.
Wiki says, Orange is “a visual programming front-end for exploratory qualitative data analysis and interactive data visualization.”
- No need to learn Python programming to analyze the data.
- Orange has many add-on toolboxes such as **Bioinformatics, Timeseries, Survival analysis, Text and Spectroscopy**.
- It is open-source platform and **FREE**.

User interface

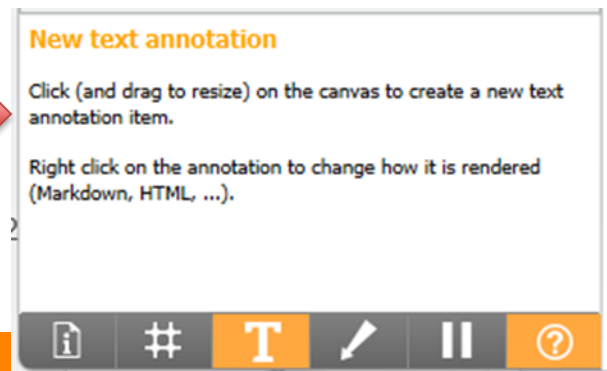
Widget Catalog

• Data • Transform • Visualize • Model • Evaluate •
Unsupervised • Educational • other add-ons



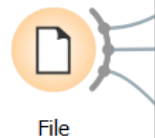
Canvas

Place widgets on the canvas, connect them, load your datasets, and harvest the insight! Interactive Data Visualization. Explore ...



Import data

- File widget to import a dataset on your local drive or online using URL.



File

Feature Statistics

File - Orange

Source

☒ File: Orange 4ML\BreastCancer.csv ... Reload

☐ URL: ...

File Type

Automatically detect type

Info

569 instances
33 features (3.0% missing values)
Data has no target variable.
0 meta attributes

Columns (Double click to edit)

	Name	Type	Role	Values
1	id	N numeric	meta	
2	diagnosis	C categorical	target	B, M
3	radius_mean	N numeric	feature	
4	texture_mean	N numeric	feature	
5	perimeter_mean	N numeric	feature	
6	area_mean	N numeric	feature	

Reset Apply

[Browse documentation datasets](#)

569

Import data

- The data feature can be defined here as well. Diagnosis is set as the target variable. ID can be set as meta.
- **Role:** features, **target**, meta.

Note: Meta variables are [meta data](#), data about data, not used for statistical inference.

- Features can also be defined at Select Columns Widget



Breast Cancer Data

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. A few of the images can be found at <http://www.cs.wisc.edu/~street/images/>. The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features:

Target variable:

- **diagnosis:** M for malignant, B for benign – Acts as the 'Target'
- **area:** tumor area – Acts as the 'Target' or Response for Example2

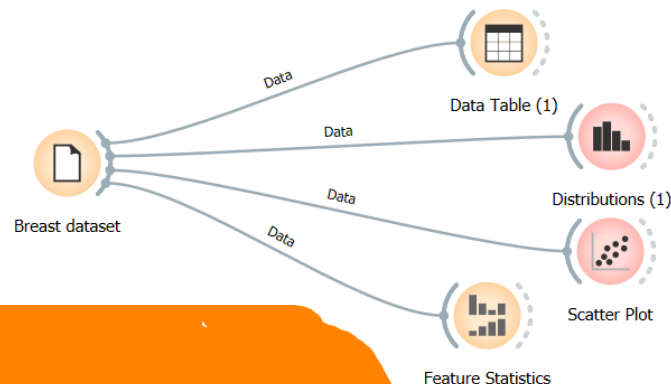
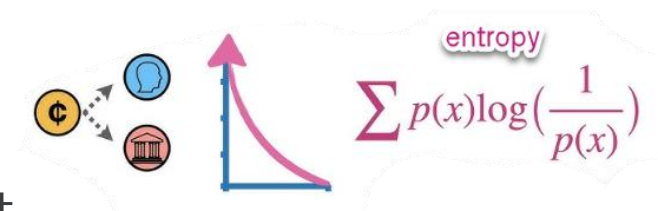
Features:

- **radius:** distances from center to points on the perimeter
- **texture:** standard deviation of gray-scale values
- **perimeter:** tumor perimeter
- **smoothness:** local variation in radius lengths
- **compactness:** $\text{perimeter}^2 / \text{area} - 1.0$
- **concavity:** severity of concave portions of the contour
- **concave points:** number of concave portions of the contour
- **symmetry:** measure of tumor symmetry
- **fractal dimension:** "coastline approximation" – 1

Data source: <https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic>.

Describe the data

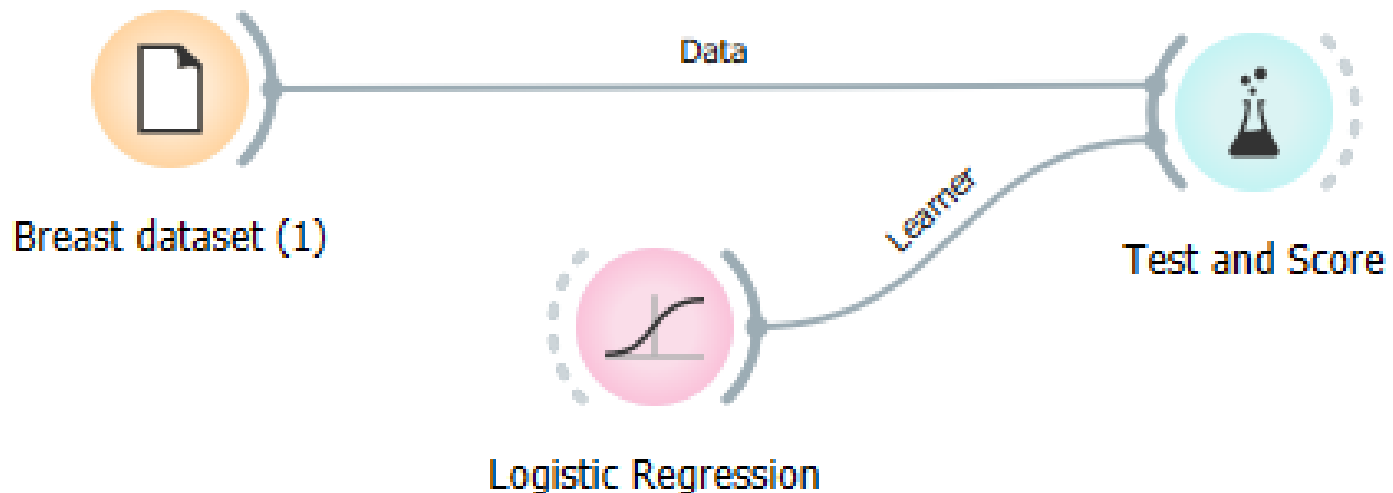
- Two way to find a widget:
 - Drag widget from the widget catalog
 - Drag a line from the file widget the widget options will be shown automatically
- Data table: View the Data
- Distribution: Plot the histogram
- Scatter plot: show x by y scatter plot
- Feature statistics: **Dispersion** (Coefficient of variation-CV for numeric data and entropy for categorical data)
- Other features: box plot, violin plot, line plot, bar plot, heat map and etc.



Fit a simple ML model

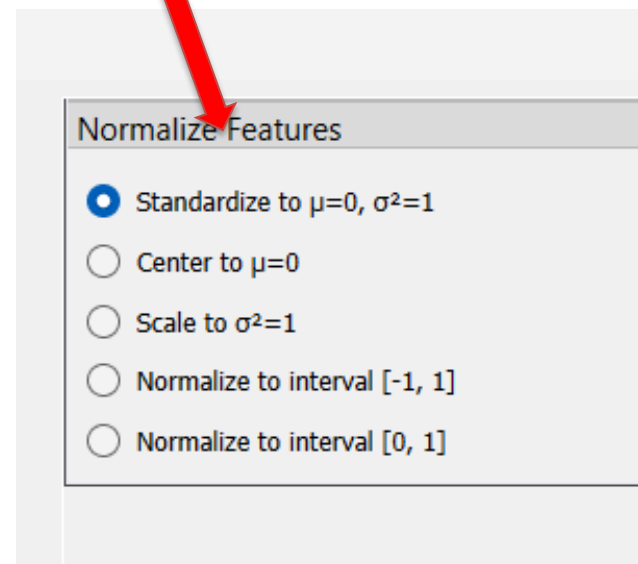
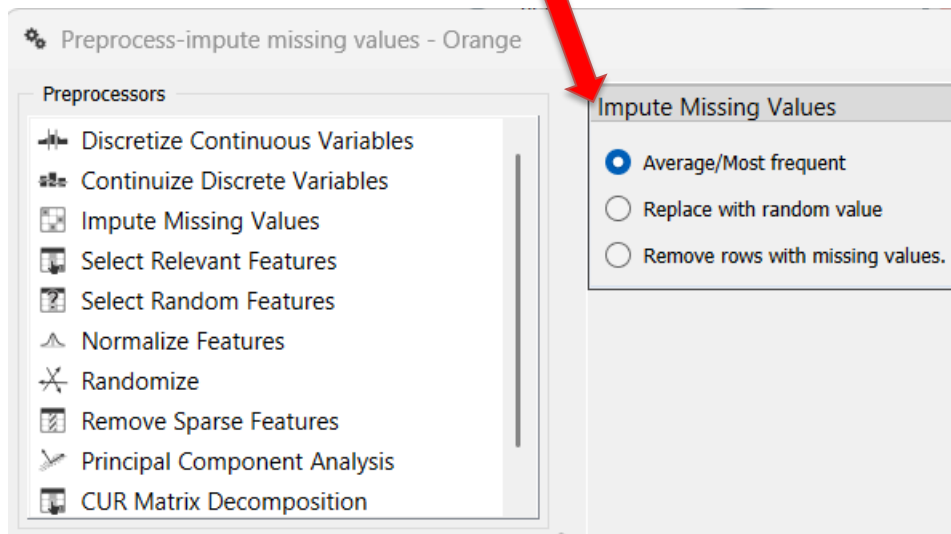
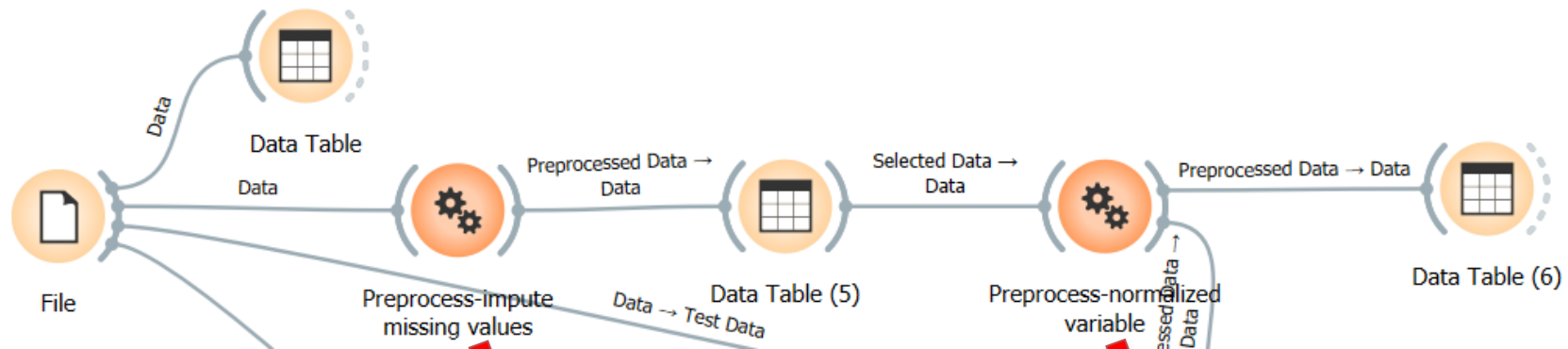
- Fit a logistic regression model
- Data file and logistic regression widget are both connected to test and score.

Build a simple ML model



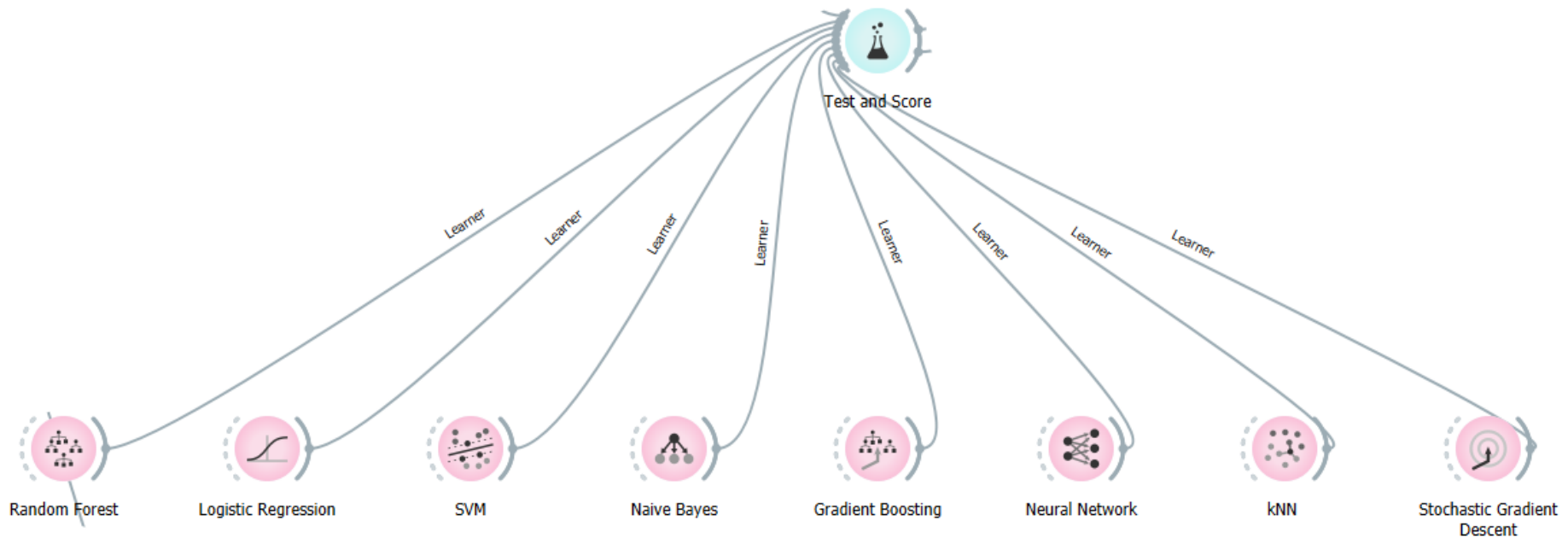
Preprocess widget ()

- Missing values imputation
- Rescale the data



Fit ML models

- All the models can be dragged to the canvas and connected to **Test and Score**



Fit ML models

Random Forest – An ensemble model that builds multiple decision trees and combines their predictions to improve accuracy and control over-fitting. The model also called **classifier or learner** in Orange and Sklearn.

Logistic Regression – A logistic regression that includes a regularization term to prevent overfitting by penalizing large coefficients generated by collinear features/independent variables. Note that regularization is applied by default.

Support Vector Machines (SVM) – A model that finds the optimal hyperplane to separate data points into different classes with the maximum distance in a high dimensional space.

Naïve Bayes – A model works only with discrete attributes. By default, continuous attributes are discretized.

Gradient Boosting (XGBoost) – An ensemble model that builds multiple sequential trees, each correcting errors of previous ones to achieve higher predictive accuracy. (See [here](#))

Fit ML models

K-Nearest Neighbors – A model assumes that similar things exist in close proximity. In other words, similar things are near to each other.

Neural Network Algorithm – A model simulates human brain by weighting “neurons” stored in “hidden layers”. Each training observation adjusts the impact of each neuron, often through “back-propagation”. Deep Learning uses many layers and many neurons per layer. Multi-layer Perceptron (MLP) classifier – this model optimizes the log-loss function using LBFGS or stochastic gradient descent.

Stochastic Gradient Descent – A model uses stochastic gradient descent that minimizes a chosen loss function with a linear function. The algorithm approximates a true gradient by considering one sample at a time, and simultaneously updates the model based on the gradient of the loss function. (See [here](#))

Model Evaluation

Test and scoring

- **Cross validation**: splits the data into a given number of folds (usually 5 or 10) and tested by holding out examples from one fold at a time; the model is induced from other folds and examples from the held out fold are classified. This is repeated for all the folds. **KNIME uses meta nodes.**
- **AUC**: Area Under Curve
- Classification accuracy (**CA**): Accuracy
- **Matthews correlation coefficient (MCC)** : an MCC score closer to 1 indicates better classifier performance, while a score closer to -1 suggests poor performance. In practice, an MCC score above 0.3 is considered moderate, and a score above 0.5 is considered strong.

Test and Score - Orange

☒ Cross validation

Number of folds: 5

☒ Stratified

☐ Cross validation by feature

☐ Random sampling

Repeat train/test: 10

Training set size: 66 %

☒ Stratified

☐ Leave one out

☐ Test on train data

☐ Test on test data

Evaluation results for target B

Model	AUC	CA	F1	Prec	Recall	MCC
Logistic Regression	0.995	0.974	0.979	0.967	0.992	0.944
Random Forest	0.982	0.949	0.960	0.956	0.964	0.891
SVM	0.994	0.977	0.982	0.975	0.989	0.951
Gradient Boosting	0.994	0.956	0.965	0.961	0.969	0.906
AdaBoost	0.911	0.917	0.934	0.933	0.936	0.823
Naive Bayes	0.985	0.937	0.949	0.952	0.947	0.865
Neural Network	0.996	0.981	0.985	0.978	0.992	0.959
kNN	0.986	0.965	0.972	0.959	0.986	0.925
Stochastic Gradient Descent	0.972	0.977	0.982	0.973	0.992	0.951
CN2 Rule Induction	0.922	0.900	0.922	0.899	0.947	0.784



Data Sampler

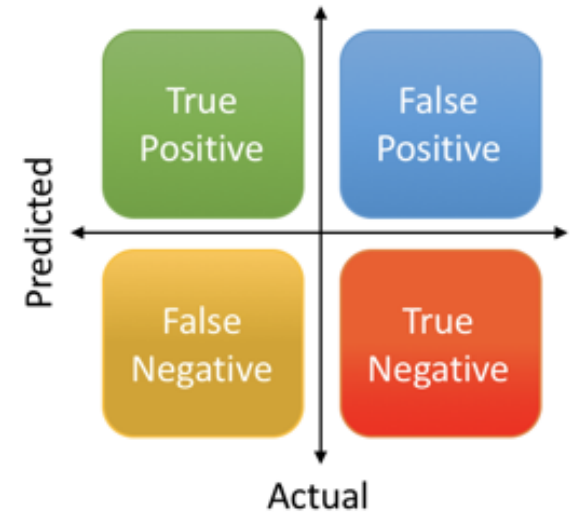
Data Sampler: split the data into training and testing

Test and Score – Model Evaluation

$$\text{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total}}$$



$$F1 \text{ score} = \frac{2 (\text{true positives})}{(2 \text{ true positives} + \text{false positives} + \text{false negatives})}$$

$$MCC = (TP \times TN - FP \times FN) / \sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}$$

Model Evaluation – Confusion Matrix

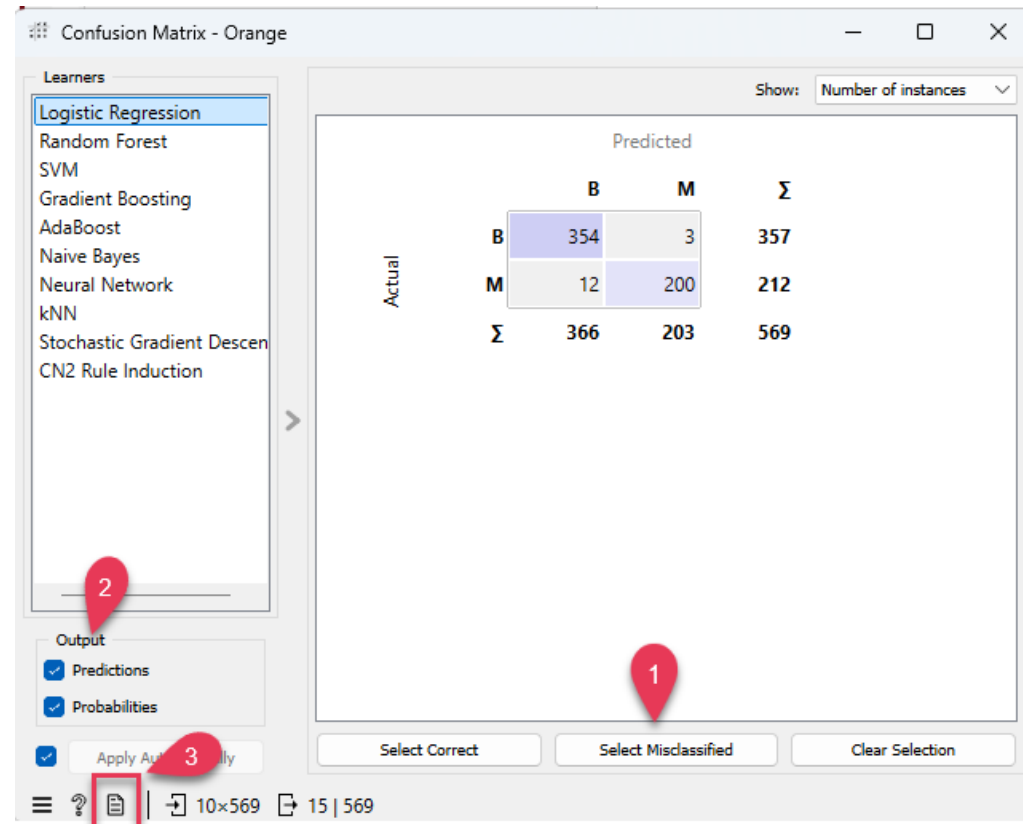
Shows proportions between the predicted and actual class

1. Select: output data

- Correct data
- Misclassified data
- None

2. Add predictions and probabilities in the output

3. Produce a report

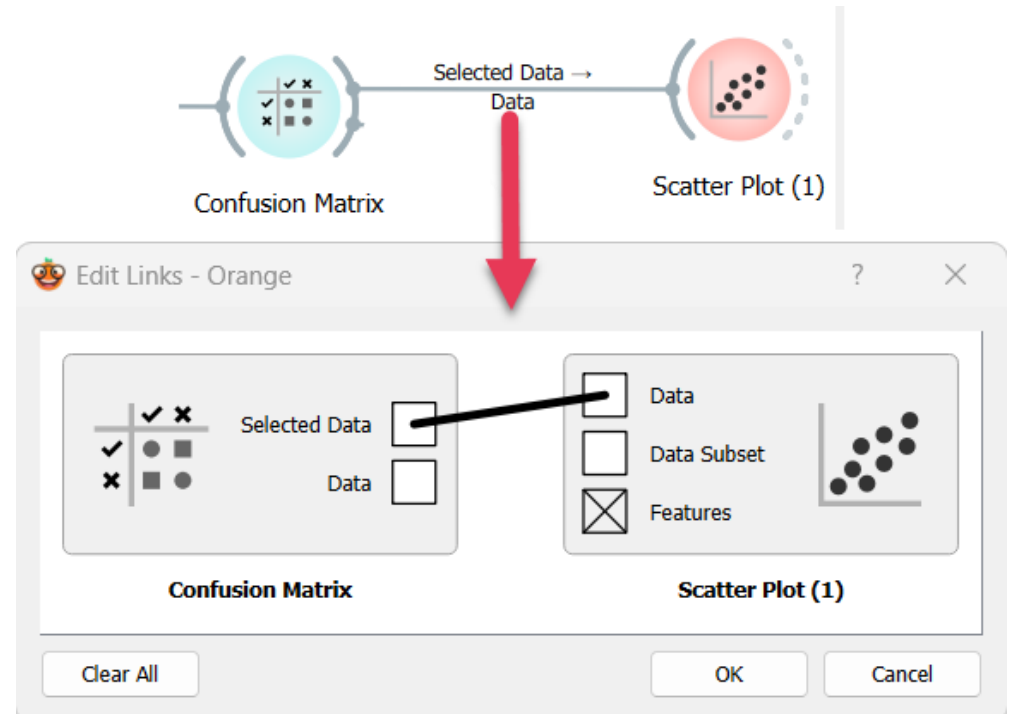


Links

- Links between two widgets can transfer objects from the one widget to the other widget.

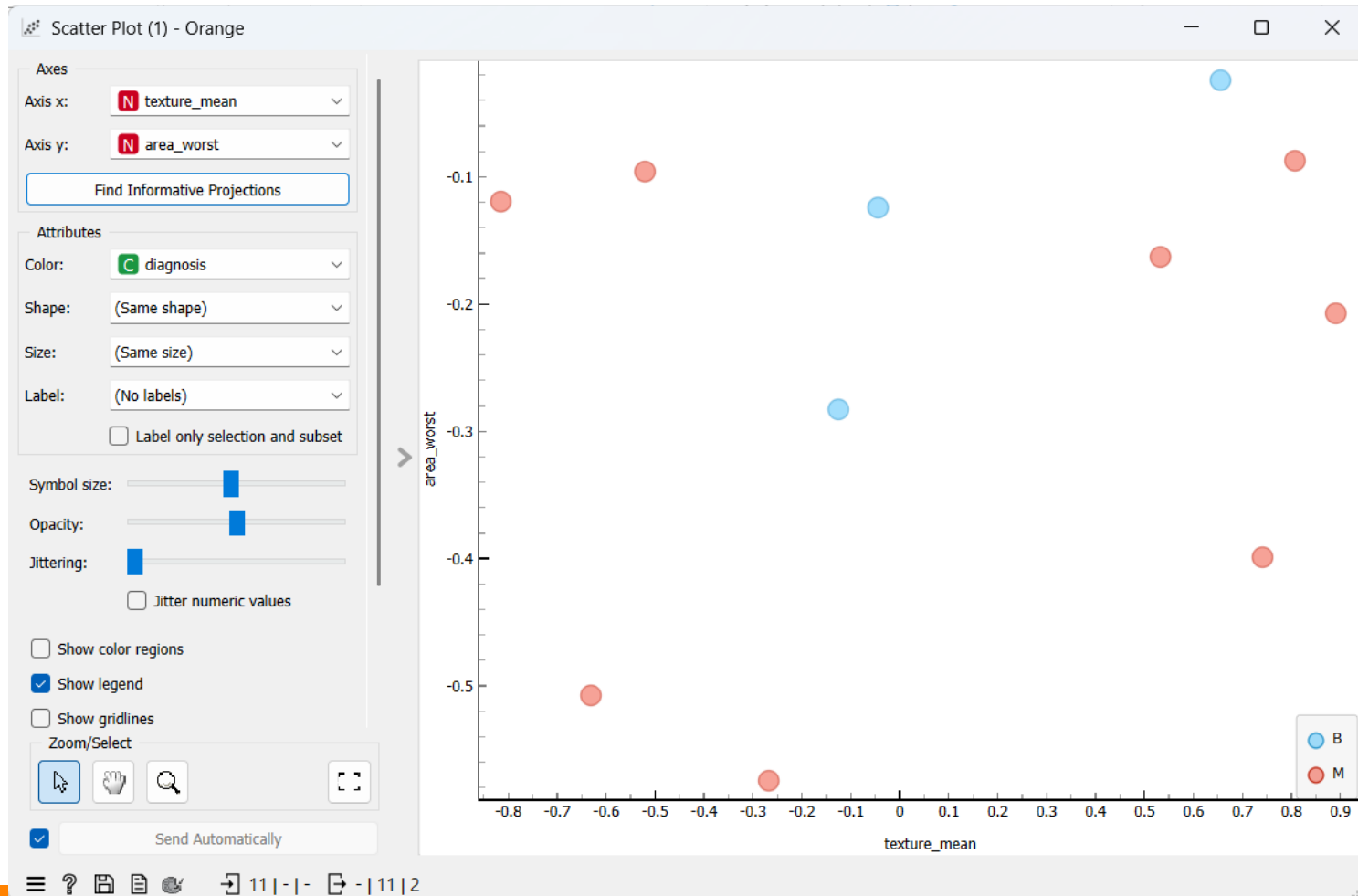
- Classifier
- Learner or Model
- Data
- Predictions
- Objects

KNIME uses different types of nuggets but Orange uses Links



Model Evaluation – Scatter plot

- Scatter plot: visually show misclassified the observations by x and y variables.



Report

- Each widget has a report button in the status bar at the bottom.
- Click **Once!**

Report - Orange

Confusion Matrix

Confusion Matrix

Outline the saved output

Save the output as html, pdf, or report

Back to Last Scheme

Save

Print

Predicted

		B	M	Σ
Actual	B	352	5	357
	M	15	197	212
	Σ	367	202	569

Write a comment...

Confusion Matrix Mon Nov 18 24, 08:51:54

Confusion matrix for Neural Network (showing number of instances)

Predicted

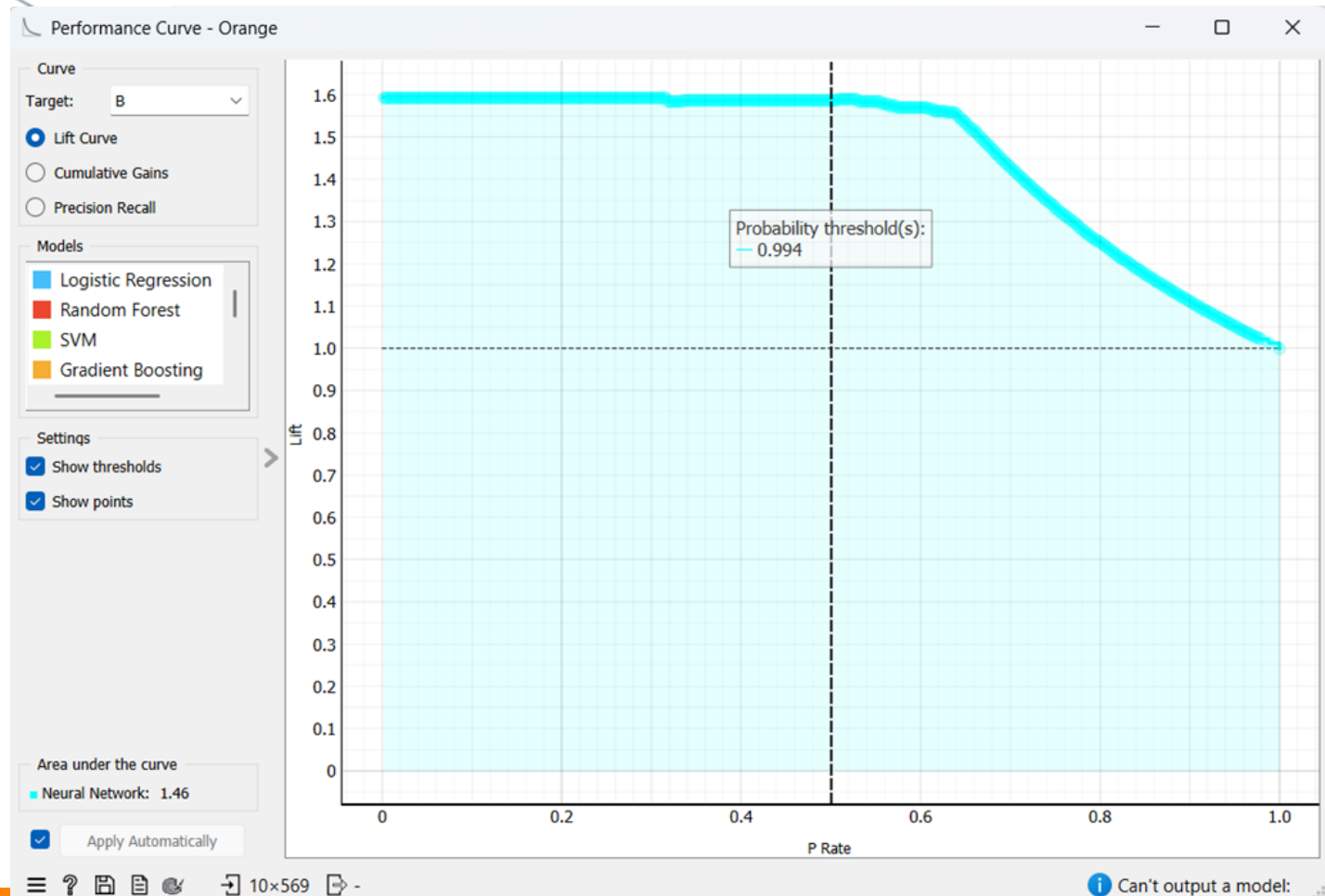
		B	M	Σ
Actual	B	354	3	357
	M	8	204	212
	Σ	362	207	569

Write a comment...

Performance Curve

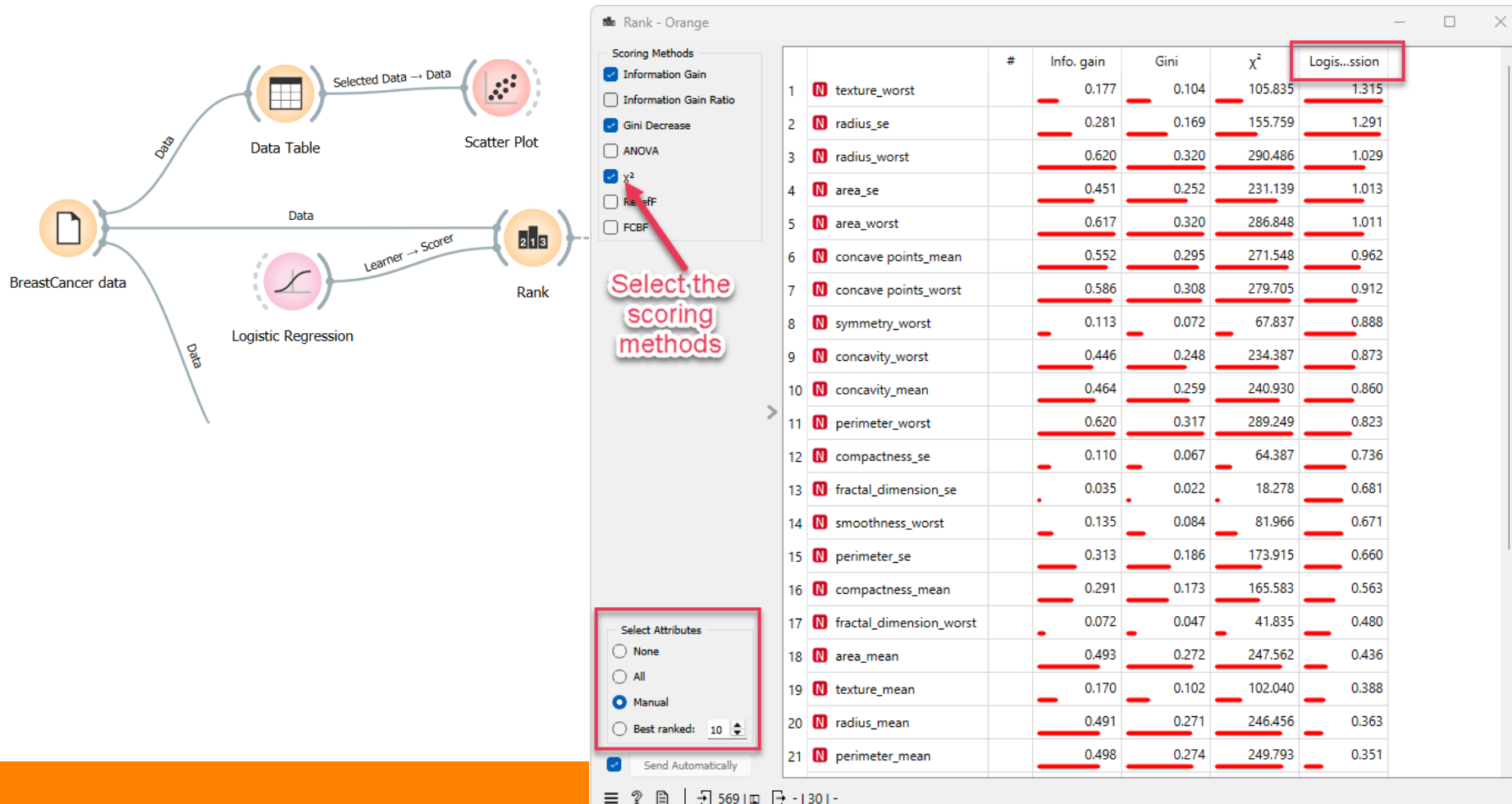
- Analyze the proportion of true positive data instances in relation to the classifier's threshold or the number of instances that we classify as positive. lift curve, cumulative gains, and precision-recall curve.
- **Lift curve** shows the ratio between lift (the proportion of true positive instances to all positive instances in the prediction) and the proportion of positive instances. The higher the initial curve and the longer it is flat, the better the model.
- **Cumulative gains chart** shows the ratio of true positive instances and support, which is the fraction of positively predicted instances, assuming that the instances are ordered according to the model's probability of being positive (e.g. how likely the person has the disease). The greater the area between the curve and the baseline (dashed diagonal line), the better the model.
- **Precision-recall** curve shows the ratio between precision (ratio of true positives in positive predictions) and recall (ratio of true positives in positive class) at different thresholds. Ideally one aims at a high area under the curve.

Performance Curve



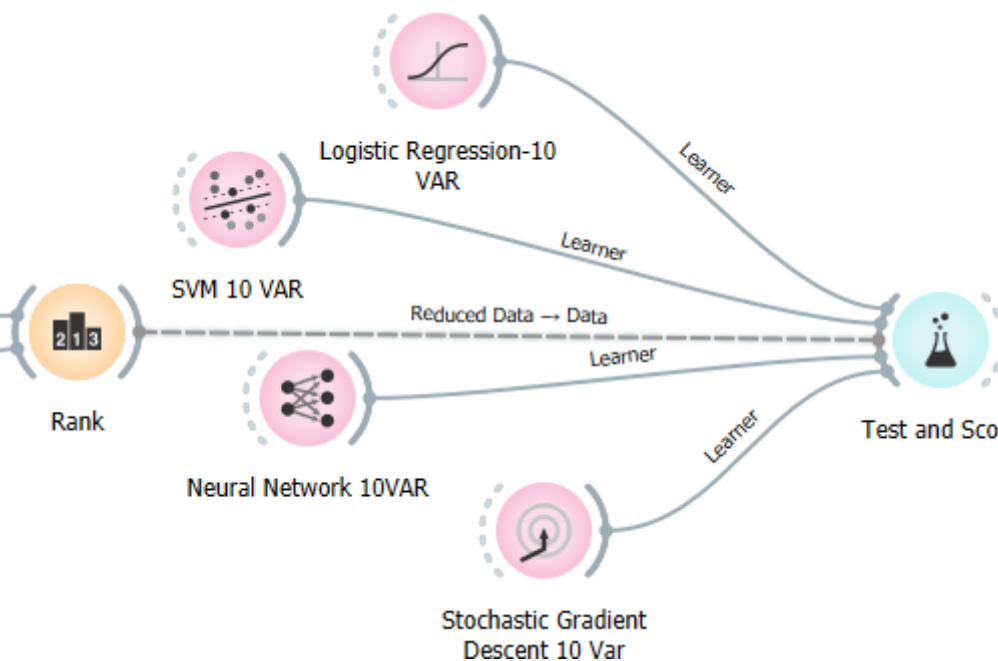
ML Variable Selection – Rank (1 of 2)

- Rank scores variables according to their correlation with discrete or numeric target variable, based on applicable internal scorers (like information gain, chi-square and linear regression) and any connected external models that supports scoring.



ML Variable Selection – Rank (2 of 2)

- Fit 4 ML models using first 10 selected variables

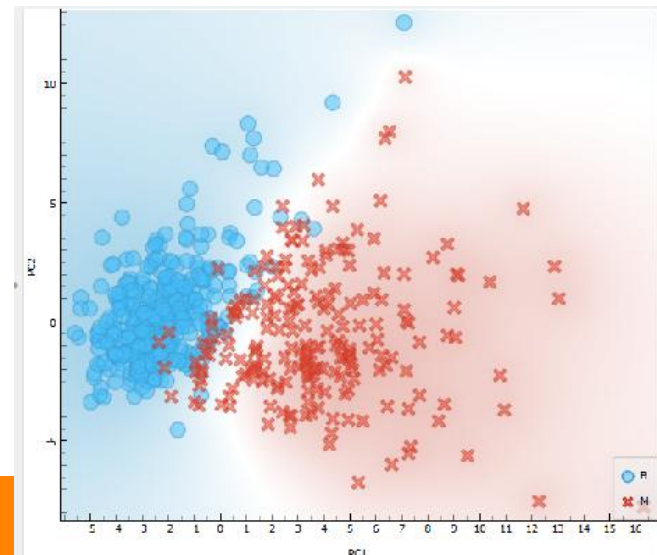
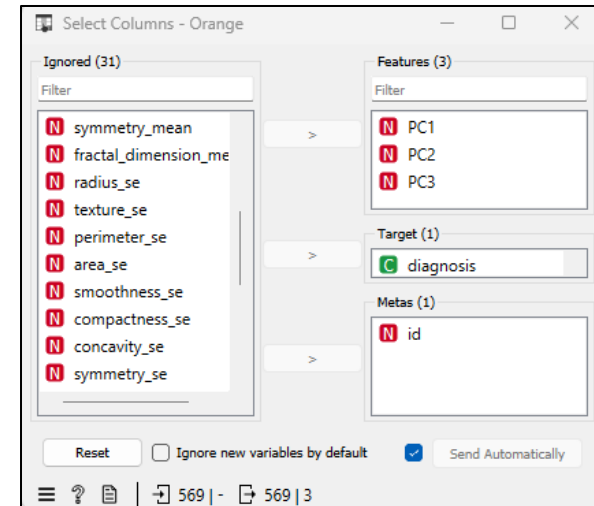
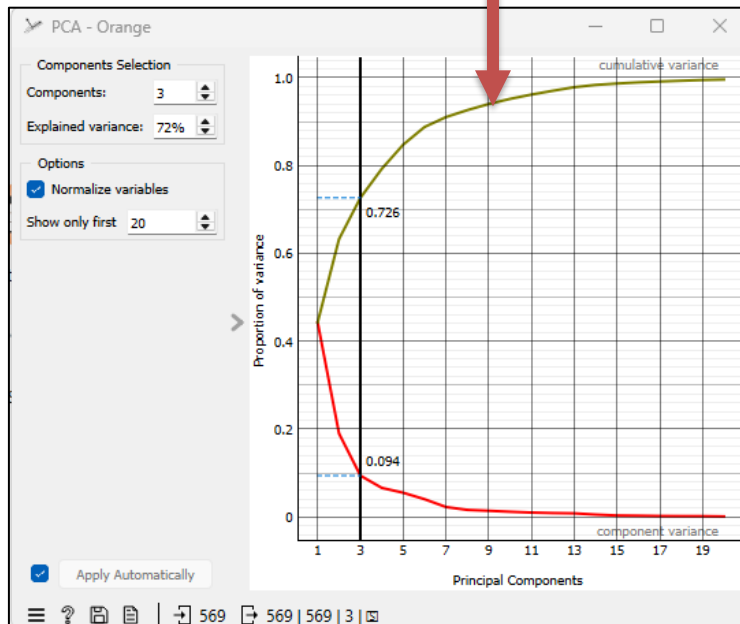
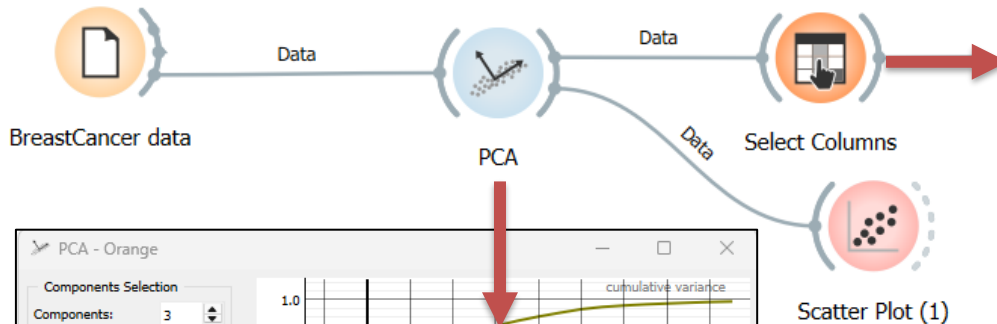


Evaluation results for target (None, show average over classes) ▾

Model	AUC	CA	F1	Prec	Recall	MCC
Logistic Regression-10 VAR	0.990	0.954	0.954	0.954	0.954	0.902
SVM 10 VAR	0.994	0.968	0.968	0.968	0.968	0.932
Neural Network 10VAR	0.995	0.968	0.968	0.968	0.968	0.932
Stochastic Gradient Descent 10 Var	0.963	0.967	0.967	0.967	0.967	0.928

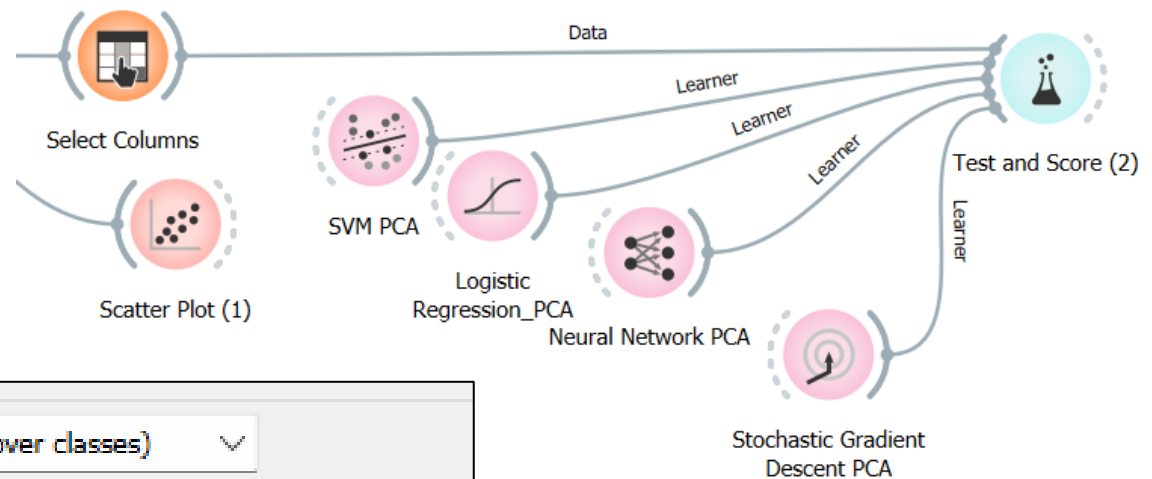
Principle Component Analysis (1 of 2)

- PCA -> Select Columns (3PCA) -> Fit models -> Test and Score



Principle Component Analysis

- PCA-> Select Columns (3PCA) -> Fit models -> Test and Score



Evaluation results for target (None, show average over classes) ▼

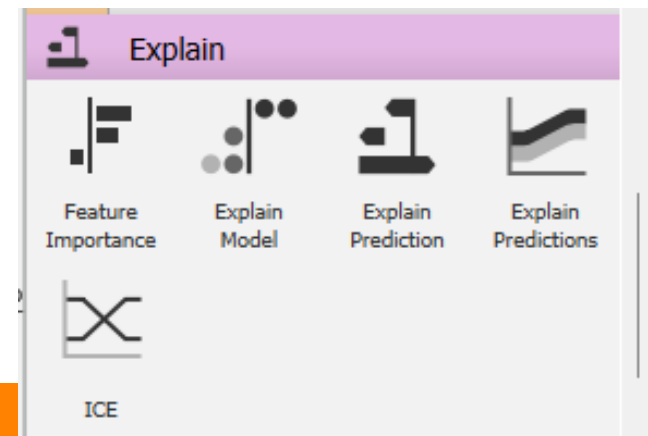
Model	AUC	CA	F1	Prec	Recall	MCC
Neural Network PCA	0.991	0.951	0.951	0.951	0.951	0.894
SVM PCA	0.987	0.947	0.947	0.947	0.947	0.887
Logistic Regression_PCA	0.990	0.944	0.944	0.944	0.944	0.880
Stochastic Gradient Descent PCA	0.941	0.944	0.944	0.944	0.944	0.880

Add-on: Explain

- Orange3 Explain is an add-on for the Orange3 data mining suite. It provides extensions for explanatory AI. (See [more](#)).
- Option-> add-ons->Explain

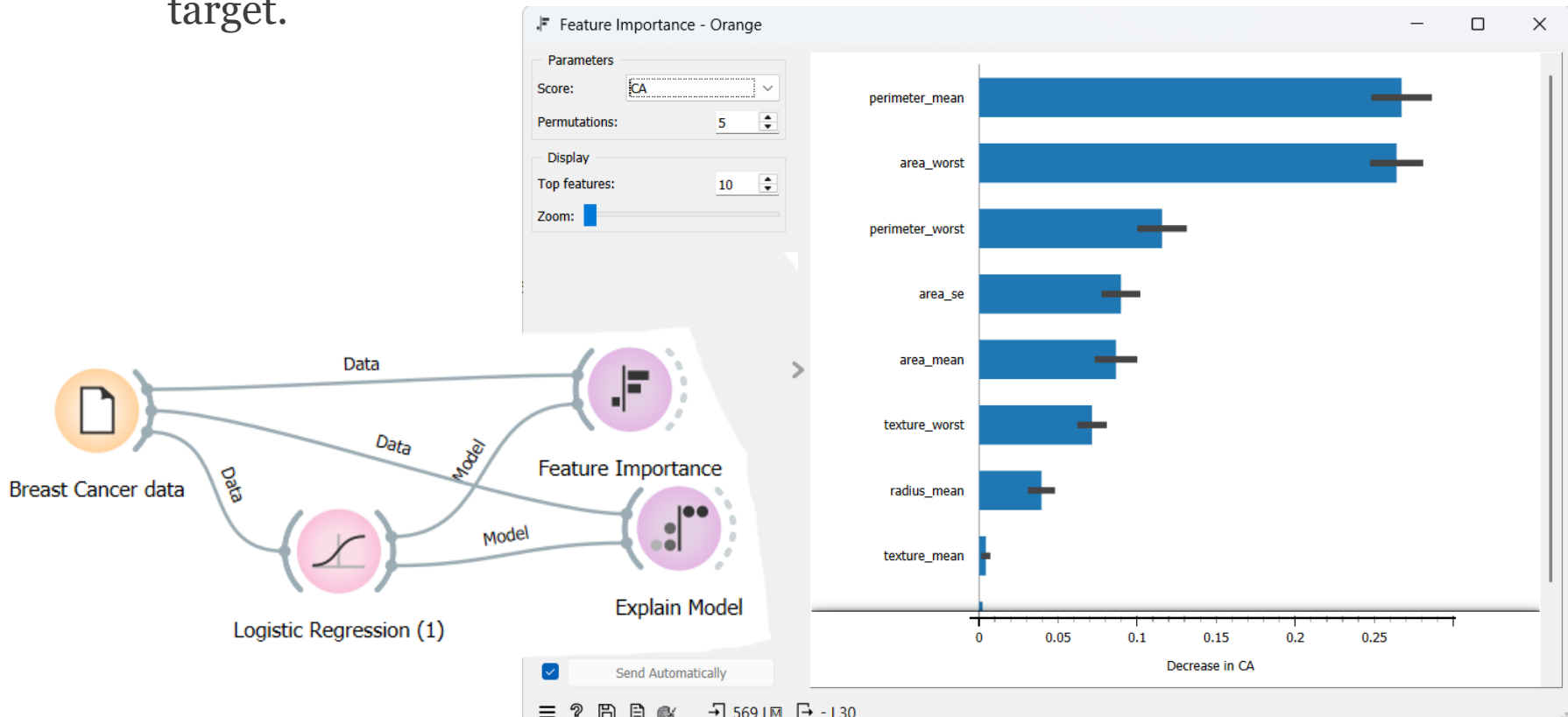
Features

- Explains a classification or regression model. Explains which features contribute the most and how they contribute toward the prediction for a specific class.
- Explains which features contribute the most to the prediction for a single instance based on the model and how they contribute.



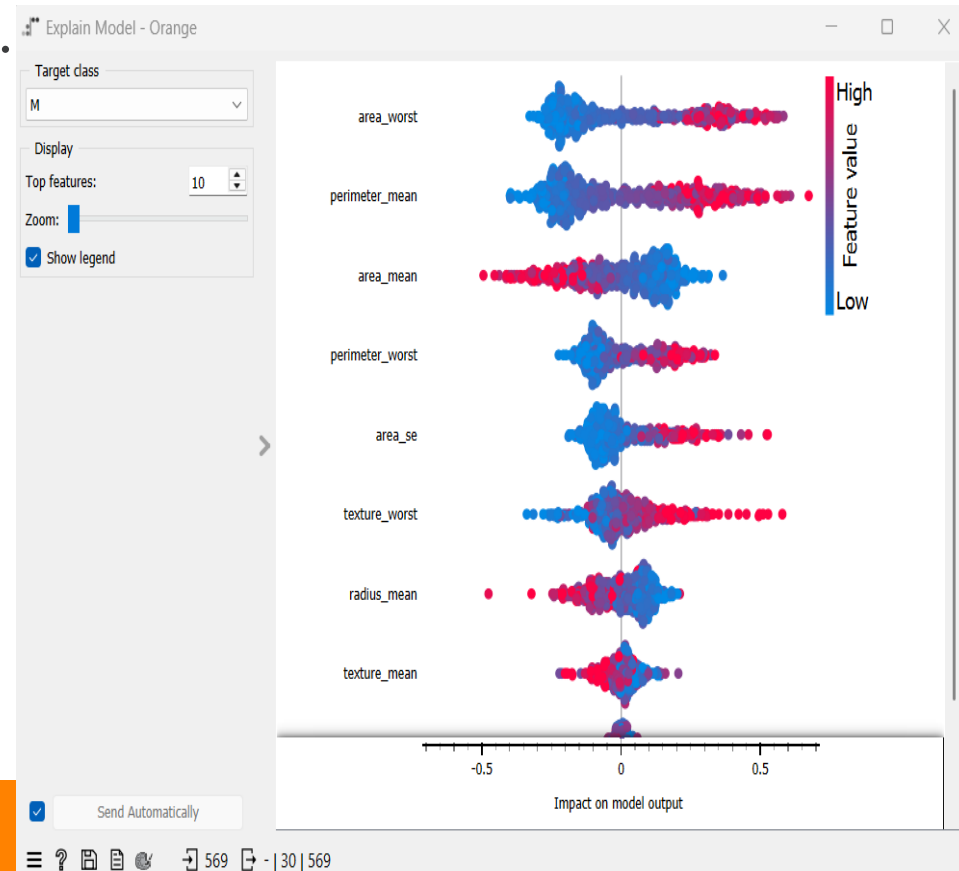
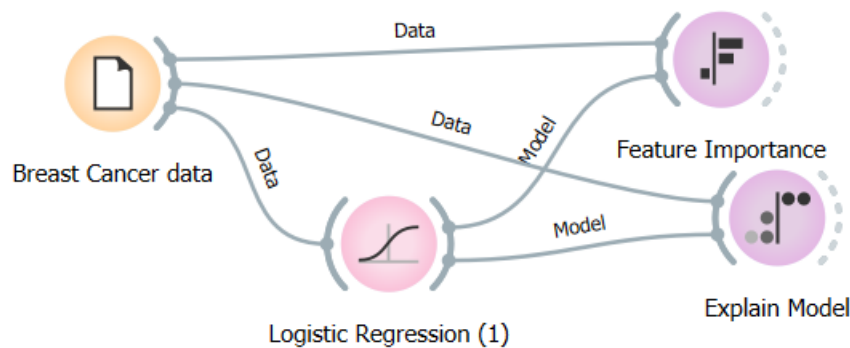
Feature Importance

- Option-> add-ons-> Explain
- Compute the contribution of each feature toward the prediction, by measuring the increase in the prediction error of the model after we permuted the feature's values, which breaks the relationship between the feature and the target.



Explain Model

Explain Model: explains classification and regression models with SHAP library. SHAP value is a measure of how much each feature affect the model output. The highest ranked variable are perimeter mean and area_worst ([the worst areas](#))- the variables with the highest impact on the prediction. Having a higher value in area_worst (red dots on the right) means the tumor is likely to be M.



Hyperparameter Tuning

Two kinds of parameters in machine learning models

1. Model Parameters

e.g. intercept (a) and slope (b) in a regression model $y = bx + a$

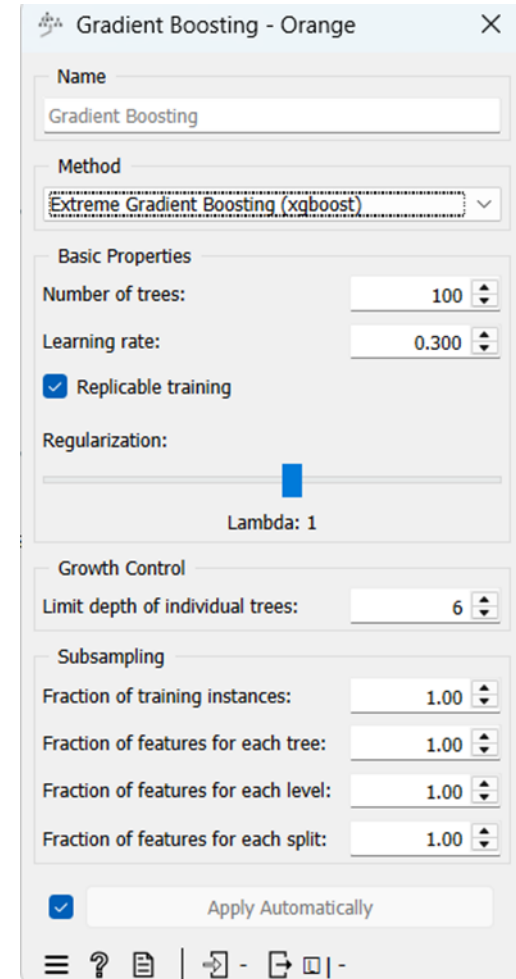
2. Hyperparameters

It directly control model structure, function, and performance.

We tune the hyperparameters to get the best fit and efficient model. It allow users to tweak model performance for optimal results.

- Hyperparameter tuning is the process of selecting the optimal set of hyperparameters for a machine learning model

It is an important step in the model development process, as the choice of hyperparameters can have a significant impact on the model's performance.



What are the hyperparameter tuning techniques?

5 Hyperparameter Optimization Techniques

- Manual Search

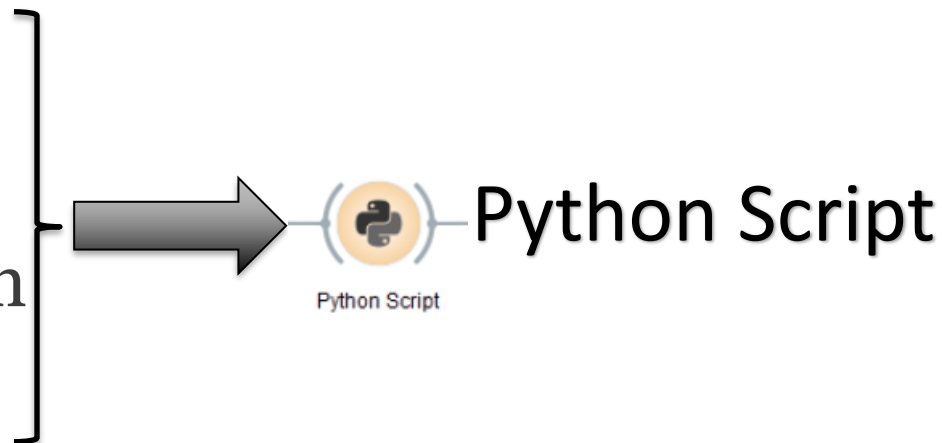
Manually selects and adjusts the hyperparameters of the model

- Grid Search

- Random Search

- Bayesian Optimization

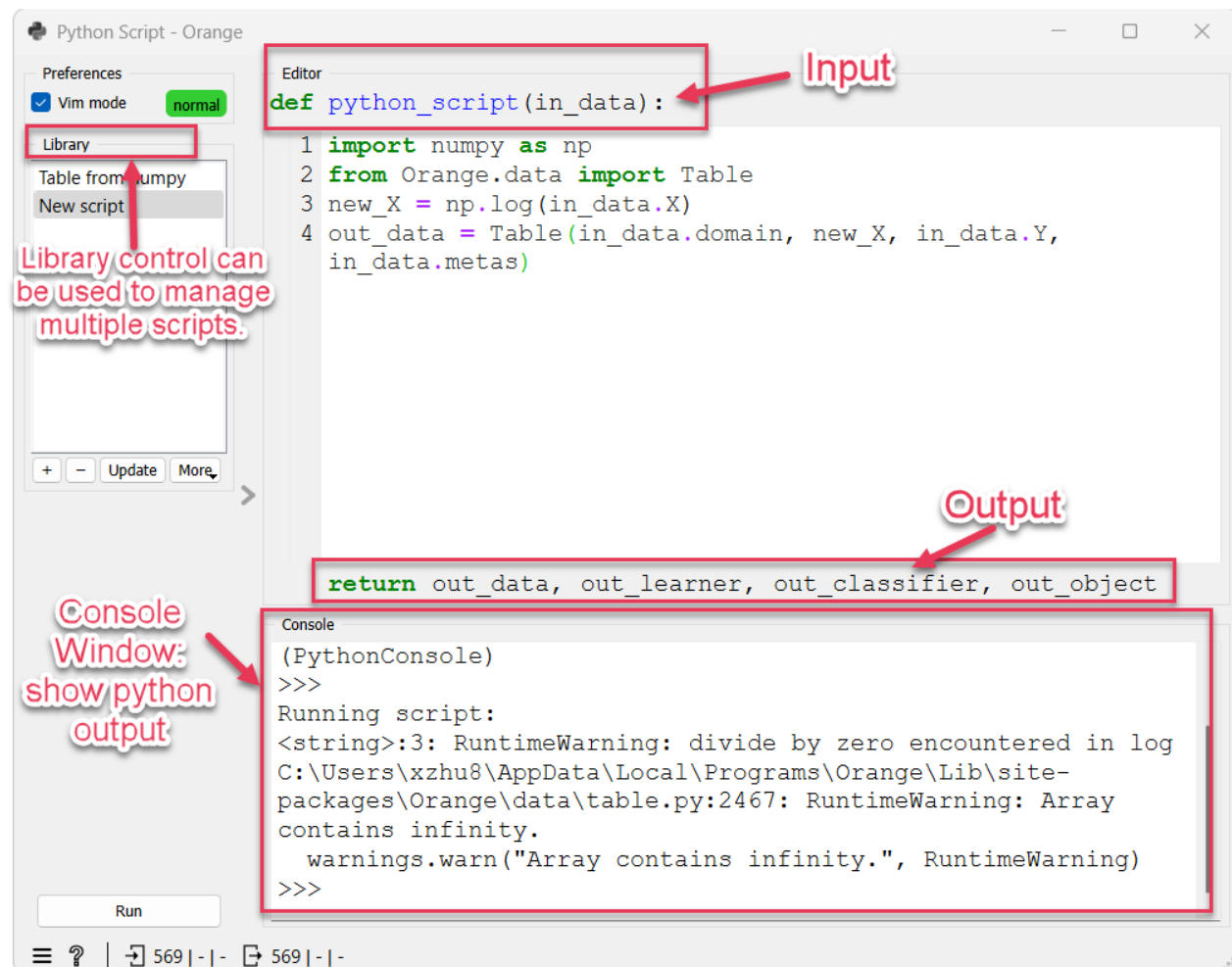
- Hyperband



Reference: <https://www.run.ai/guides/hyperparameter-tuning> or
https://scikit-learn.org/1.5/modules/grid_search.html

Hyperparameter Tuning - Python Script

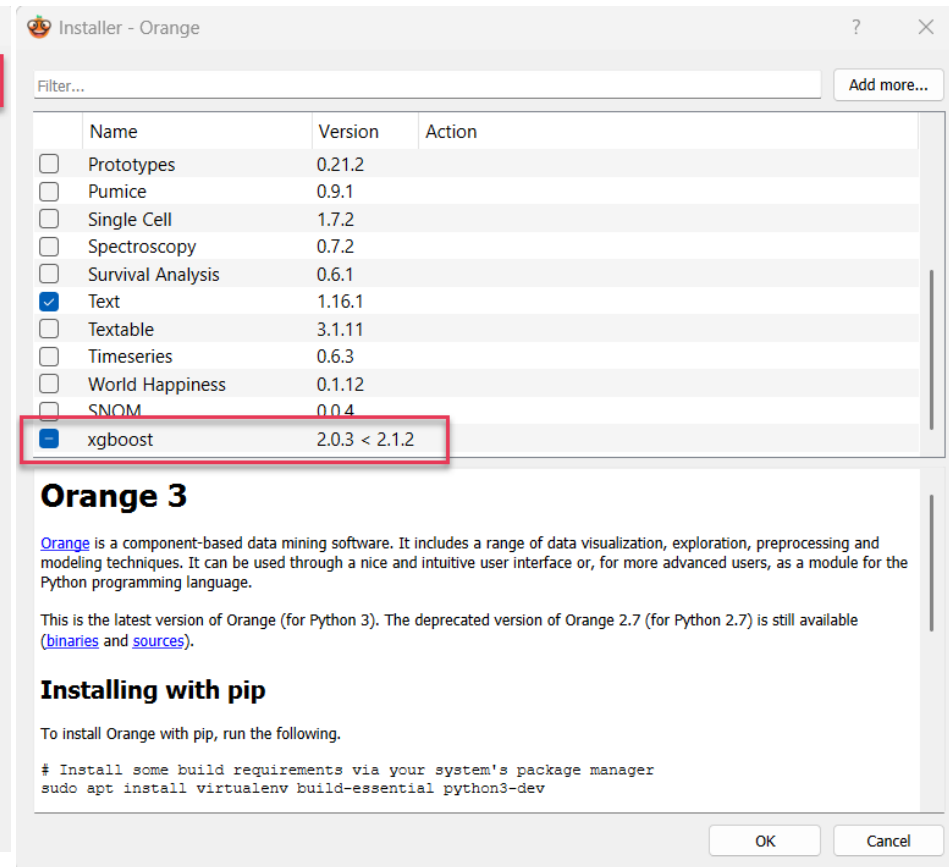
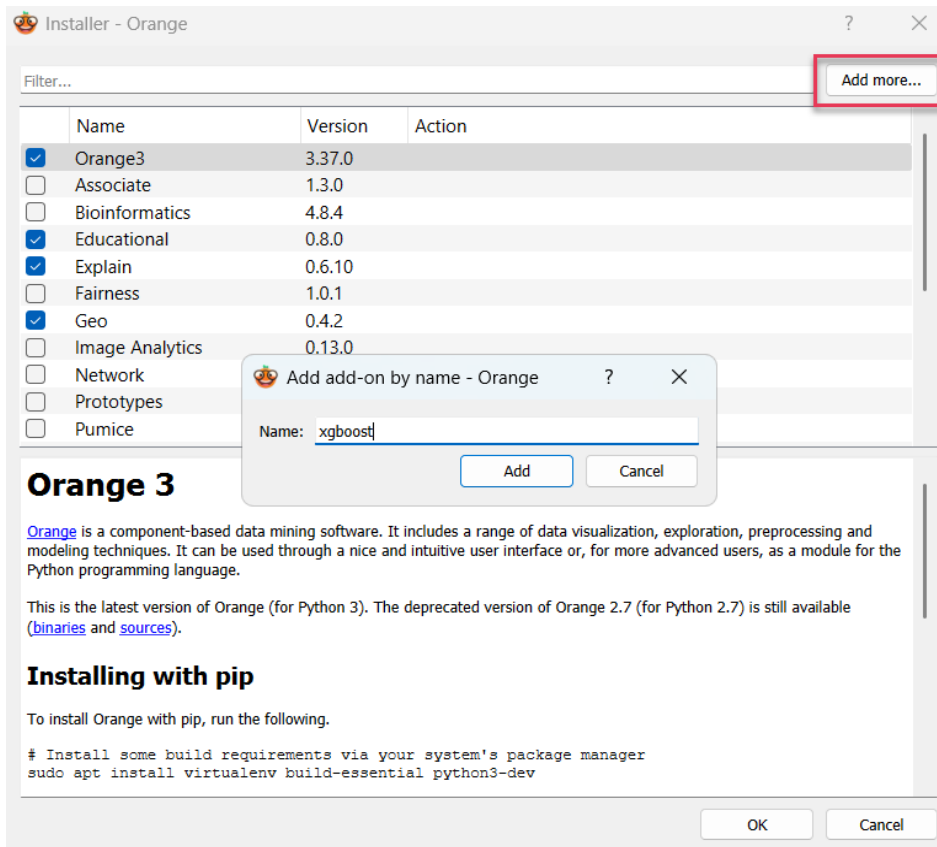
- When a suitable functionality is not implemented in an existing widget, we need to use Python Script
- Python scripts: define a function with input parameters and output parameters.



Note: Learner is a model, like logistic regression. When we give a Learner a class-labeled dataset, the learner will return a classifier.

Install a Python Package in Orange3

- Add-on -> Add more... -> Install xgboost package



Hyperparameter Tuning-GridSearch

- Sklearn – GridSearchCV to tune xgboost model

Gradient Boosting - Orange

Name
Gradient Boosting

Method
Extreme Gradient Boosting (xgboost)

Basic Properties

Number of trees: 100

Learning rate: 0.300

☒ Replicable training

Regularization: Lambda: 1

Growth Control

Limit depth of individual trees: 6

Subsampling

Fraction of training instances: 1.00

Fraction of features for each tree: 1.00

Fraction of features for each level: 1.00

Fraction of features for each split: 1.00

☒ Apply Automatically

Grid Search for Xgboost - Orange

Editor

```
def python_script(in_data):  
    3 import pandas as pd  
    4  
    5 # Define the hyperparameter grid  
    6 param_grid = {  
    7     'n_estimators': [100, 400, 800],  
    8     'max_depth': [3, 5, 7, 9],  
    9     'learning_rate': [0.3, 0.2, 0.1, 0.01, 0.001],  
   10     'min_child_weight': [1, 10, 100]  
   11     #'subsample': [0.2, 0.3, 0.5, 0.7, 1]  
   12 }  
   13  
   14 # Create the XGBoost model object  
   15 xgb_model = xgb.XGBClassifier()  
   16  
   17 # Create the GridSearchCV object  
   18 grid_search = GridSearchCV(xgb_model, param_grid, cv=5, scoring='accuracy')  
   19  
   20 # Fit the GridSearchCV object to the training data  
   21 grid_search.fit(in_data.X, in_data.Y)  
   22  
   23 # Print the best set of hyperparameters and the corresponding score  
  
   return out_data, out_learner, out_classifier, out_object
```

Console

```
NameError: name 'X_train' is not defined  
>>>  
Running script:  
Best set of hyperparameters: {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight':  
1, 'n_estimators': 400}  
Best score: 0.9771464058376029  


|   | mean_fit_time | std_fit_time | ... | std_test_score | rank_test_score |
|---|---------------|--------------|-----|----------------|-----------------|
| 0 | 0.045853      | 0.003881     | ... | 0.011991       | 34              |
| 1 | 0.096114      | 0.011287     | ... | 0.012988       | 24              |
| 2 | 0.151291      | 0.002994     | ... | 0.012988       | 24              |
| 3 | 0.024217      | 0.001008     | ... | 0.022463       | 45              |
| 4 | 0.073794      | 0.003089     | ... | 0.022463       | 45              |

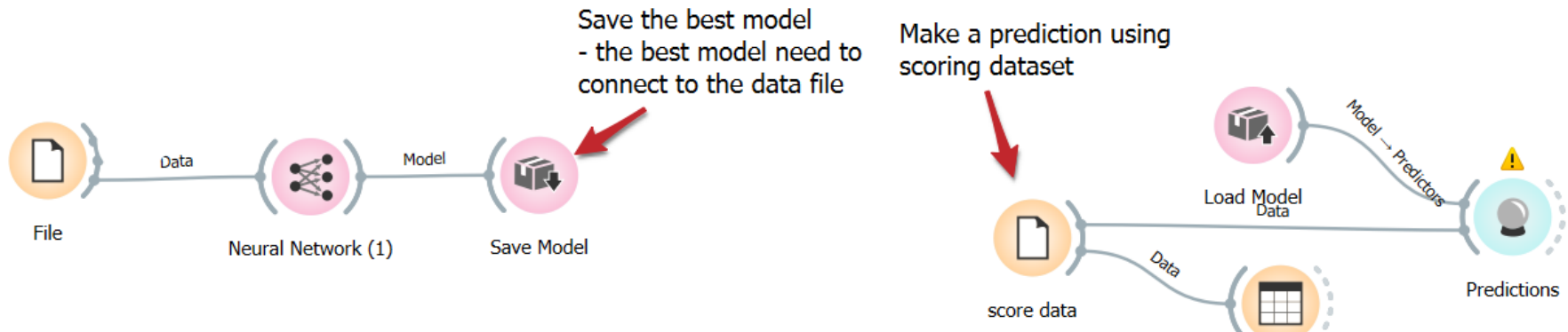

```

Run

Make Predictions Using Scoring Data

Two ways to make a prediction

1. Save the best model, then load the model and make predictions
2. Use the best model and make prediction directly – See Regression



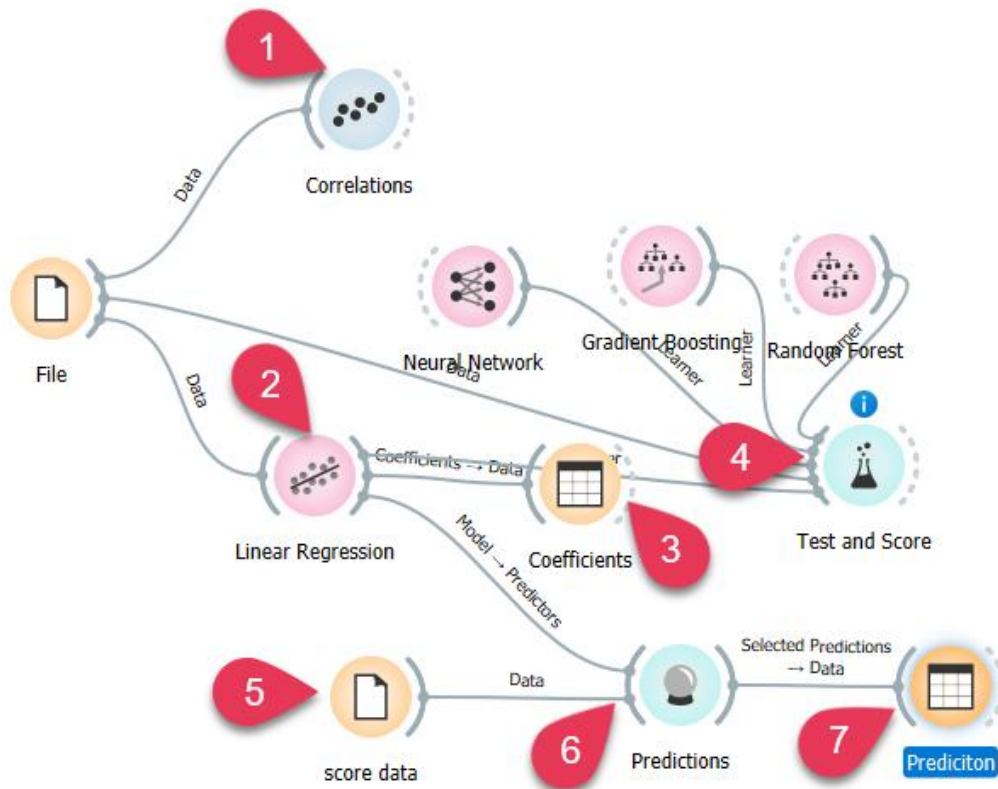
Predictions - Orange

Show probabilities for ☐ Classes known to the model ☒ Show classification errors

	Neural Network	error	diagnosis	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry
1	1.00 : 0.00 → B		?	857343	11.760	21.60	74.72	427.9	0.08637	0.04966	0.01657	0.01115	0.14
2	1.00 : 0.00 → B		?	857373	13.640	16.34	87.21	571.8	0.07685	0.06059	0.01857	0.01723	0.13
3	1.00 : 0.00 → B		?	857374	11.940	18.24	75.71	437.6	0.08261	0.04751	0.01972	0.01349	0.18
4	0.00 : 1.00 → M		?	857392	18.220	18.70	120.30	1033.0	0.11480	0.14850	0.1772	0.106	0.20
5	0.00 : 1.00 → M		?	857438	15.100	22.02	97.26	712.8	0.09056	0.07081	0.05253	0.03334	0.16
6	1.00 : 0.00 → B		?	85759902	11.520	18.75	73.34	409.0	0.09524	0.05473	0.03036	0.02278	0.19
7	0.00 : 1.00 → M		?	857637	19.210	18.57	125.50	1152.0	0.10530	0.12670	0.1323	0.08994	0.19
8	0.00 : 1.00 → M		?	857793	14.710	21.59	95.55	656.9	0.11370	0.13650	0.1293	0.08123	0.20
9	1.00 : 0.00 → B		?	857810	13.050	19.31	82.61	527.2	0.08060	0.03789	0.000692	0.004167	0.18
10	1.00 : 0.00 → B		?	858477	8.618	11.79	54.34	224.5	0.09752	0.05272	0.02061	0.007799	0.16
11	1.00 : 0.00 → B		?	858970	10.170	14.88	64.55	311.9	0.11340	0.08061	0.01084	0.0129	0.27
12	1.00 : 0.00 → B		?	858981	8.598	20.98	54.66	221.8	0.12430	0.08963	0.03	0.009259	0.18
13	0.00 : 1.00 → M		?	858986	14.250	22.15	96.42	645.7	0.10490	0.20080	0.2135	0.08653	0.19

Example 2: Numeric Target

- Set **area_mean** as target variable (y)





File - Orange

Source

☒ File:

Orange 4ML\breast cancer dataset ▾

 ...

 Reload

☐ URL:

▾






File Type

Automatically detect type ▾

Info

569 instances
32 features (no missing values)
Data has no target variable.
0 meta attributes





Columns (Double click to edit)

	Name	Type	Role	Values
4	texture_mean	 numeric	feature	
5	perimeter_mean	 numeric	feature	
6	area_mean	 numeric	target	
7	smoothness_me...	 numeric	feature	
8	compactness_m...	 numeric	feature	

Reset

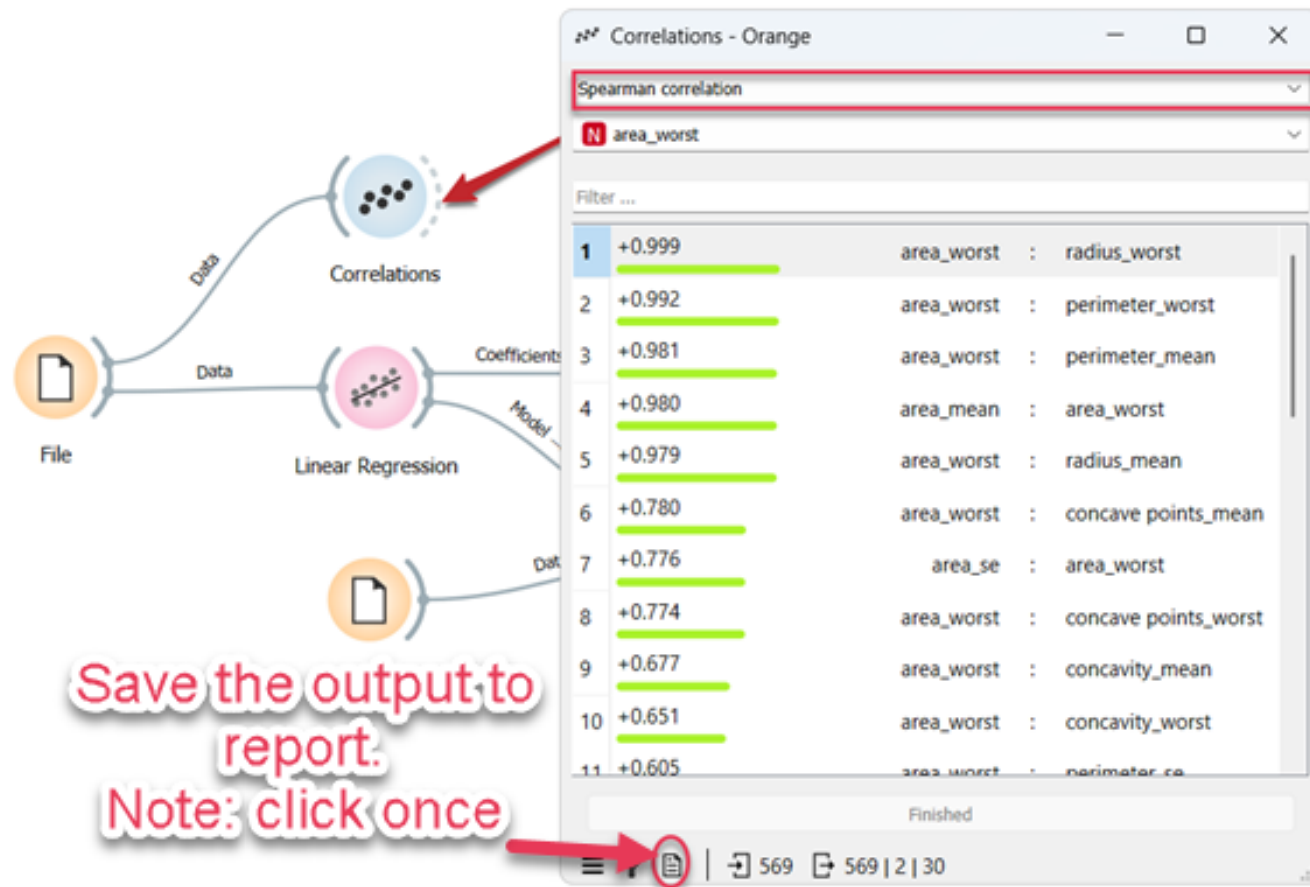
Apply

Browse documentation datasets

   |  569

Correlations

- Show Pearson or Spearman correlation with area_mean. You can also pick the other variables for correlations, e.g. area_worst.
- Click **report** to save the output



The image shows an Orange Data Mining workflow and the output of the Correlations widget. The workflow consists of a File widget connected to both a Correlations widget and a Linear Regression widget. The Correlations widget is highlighted with a red arrow. The Linear Regression widget outputs Coefficients and a Model. A red arrow points from the Correlations widget to the Correlations - Orange window. The Correlations - Orange window shows the Spearman correlation method selected, with area_worst as the variable. The output table lists 11 correlations with their coefficients and the variables involved. A red arrow points to the report icon in the bottom toolbar of the Correlations window.

Save the output to report.
Note: click once

Rank	Correlation	Variable 1	Variable 2
1	+0.999	area_worst	radius_worst
2	+0.992	area_worst	perimeter_worst
3	+0.981	area_worst	perimeter_mean
4	+0.980	area_mean	area_worst
5	+0.979	area_worst	radius_mean
6	+0.780	area_worst	concave points_mean
7	+0.776	area_se	area_worst
8	+0.774	area_worst	concave points_worst
9	+0.677	area_worst	concavity_mean
10	+0.651	area_worst	concavity_worst
11	+0.605	area_worst	perimeter_mean

Fit a Regression Model

- Fit a regression model with options
- Parameters: Fit intercept.
Unchecking the option forces the intercept to 0.
- Choose a model to train:
 - no regularization
 - a Ridge regularization (L2-norm penalty)
 - a Lasso bound (L1-norm penalty)
 - an Elastic net regularization

Linear Regression - Orange

Name
Linear Regression

Parameters
☒ Fit intercept (unchecking it fixes it to zero)

Regularization
☒ No regularization
☐ Ridge regression (L2)
☐ Lasso regression (L1)
☐ Elastic net regression

Regularization strength:
Alpha: 0.0001

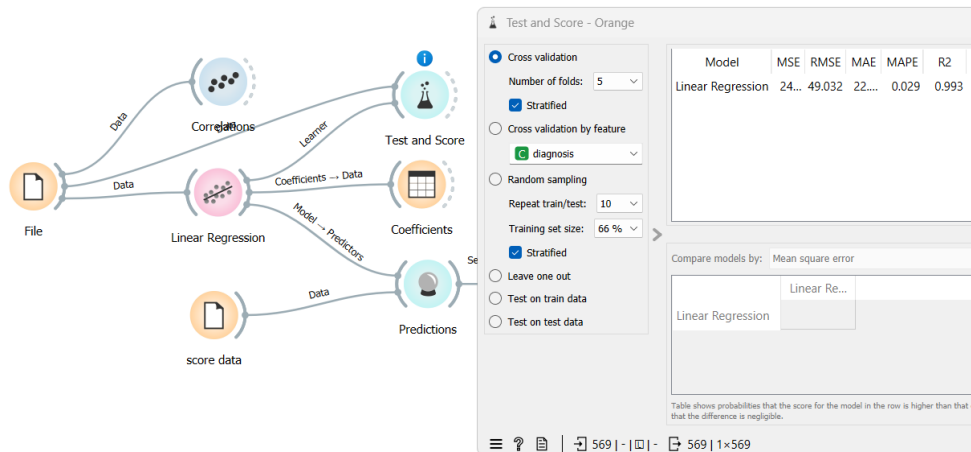
Elastic net mixing:
L1 0.50 : 0.50 L2

☒ Apply Automatically

569 | - 30 |

Fit a Regression Model

- **Data Table** to view coefficients
- **No p-values** for each variable
- **Test and score** to view MSE RMSE and R2



Coefficients - Orange

Info
30 instances (no missing data)
1 feature
No target variable.
1 meta attribute

Variables
☒ Show variable labels (if present)
☐ Visualize numeric values
☒ Color by instance classes

Selection
☒ Select full rows

	name	coef
1	intercept	-337.955
2	radius_mean	98.2609
3	texture_mean	-0.819478
4	perimeter_mean	3.00974
5	smoothness_m...	-393.299
6	compactness_m...	-444.146
7	concavity_mean	95.471
8	concave points_...	202.146
9	symmetry_mean	-121.373
10	fractal_dimensi...	1184.77
11	radius_se	63.8898
12	texture_se	-8.75167
13	perimeter_se	18.0575
14	area_se	-0.620185
15	smoothness_se	1339.85
16	compactness_se	-3.10487
17	concavity_se	317.422
18	concave points_...	-3134.36
19	symmetry_se	-236.712
20	fractal_dimensi...	-831.491
21	radius_worst	-62.1988
22	texture_worst	1.0392
23	perimeter_worst	-1.88815
24	area_worst	0.491379

Restore Original Order

☒ Send Automatically

30 30 | 30

Predict the Scoring Dataset

- Set **area_mean** as target
- Connect prediction with regression and score data
- Performance scores are shown at the bottom
- Table Data to view selected or all predicted values

Columns (Double click to edit)			
	Name	Type	Role
1	id	N numeric	meta
2	diagnosis	C categorical	meta
3	radius_mean	N numeric	feature
4	texture_mean	N numeric	feature
5	perimeter_mean	N numeric	feature
6	area_mean	N numeric	target

Predictions - Orange

Shown regression error: Absolute difference Restore Original Order

	Linear Regression	error	area_mean	id
1	435.2	7.3	427.9	857343
2	588.4	16.6	571.8	857373
3	443.4	5.8	437.6	857374
4	1053.5	20.5	1033.0	857392
5	716.5	3.7	712.8	857438
6	412.5	3.5	409.0	85759902
7	1182.5	30.5	1152.0	857637
8	676.5	19.6	656.9	857793
9	539.6	12.4	527.2	857810
10	224.0	0.5	224.5	858477
11	313.1	1.2	311.9	858970
12	190.8	31.0	221.8	858981
13	659.2	13.5	645.7	858986
14	230.0	30.9	260.9	859196
15	463.1	35.9	499.0	85922302

☒ Show performance scores

Model	MSE	RMSE	MAE	MAPE	R2
Linear Regression	483.601	21.991	16.761	0.029	0.996

50 | 50 | 50 | 1x50

Unsurprised Learning

- When dealing with real-world problems, most of the time, data will not come with predefined labels, but we still want to develop machine learning models that can correctly classify the data by finding some commonality in the features to predict the classes on new data.
- Two main types of problems in unsupervised learning:
 - Clustering
 - Dimension Reduction--PCA

Clustering Analysis

- Cluster analysis or clustering is to group a set of objects that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).



sample



Cluster/group

Clustering

- Clustering, however, has many different names (with respect to the fields it is being applied):

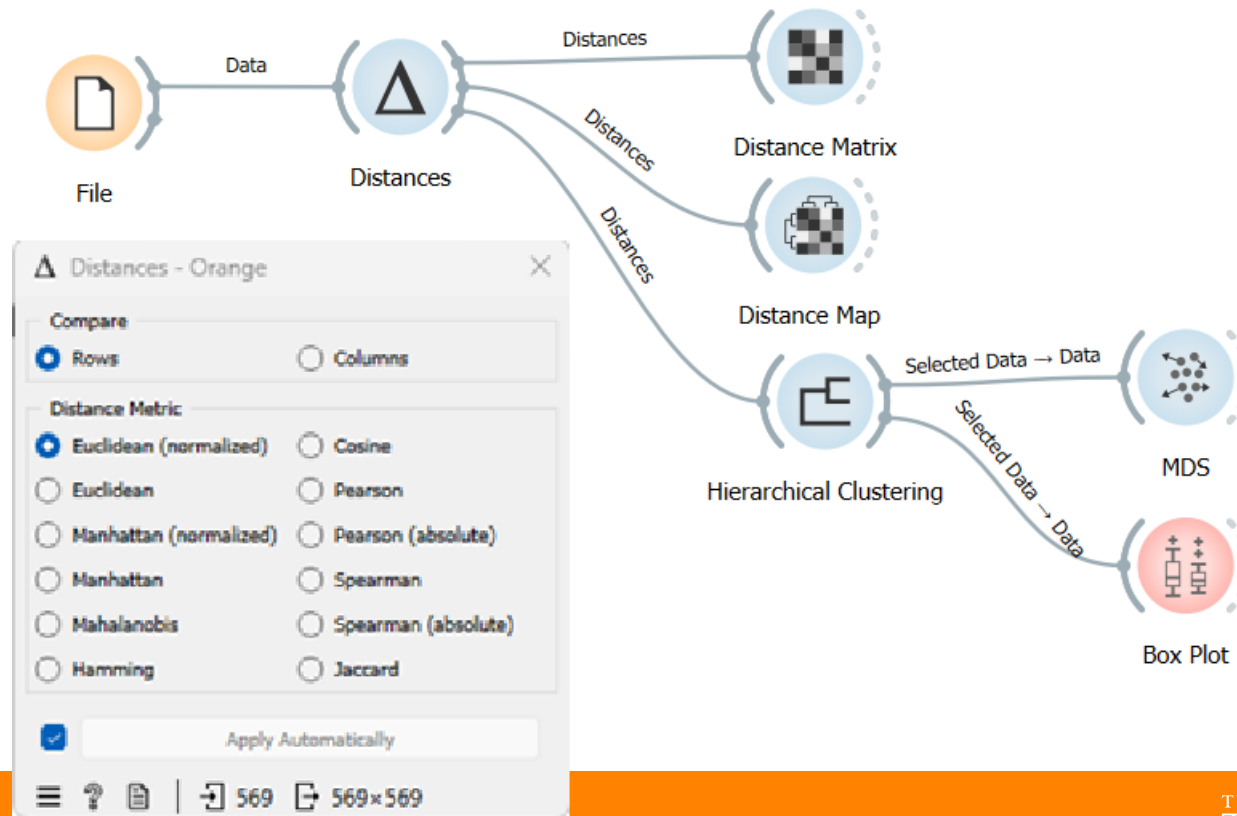
- Cluster analysis
- Automatic classification
- Data segmentation

All the above names essentially mean clustering.

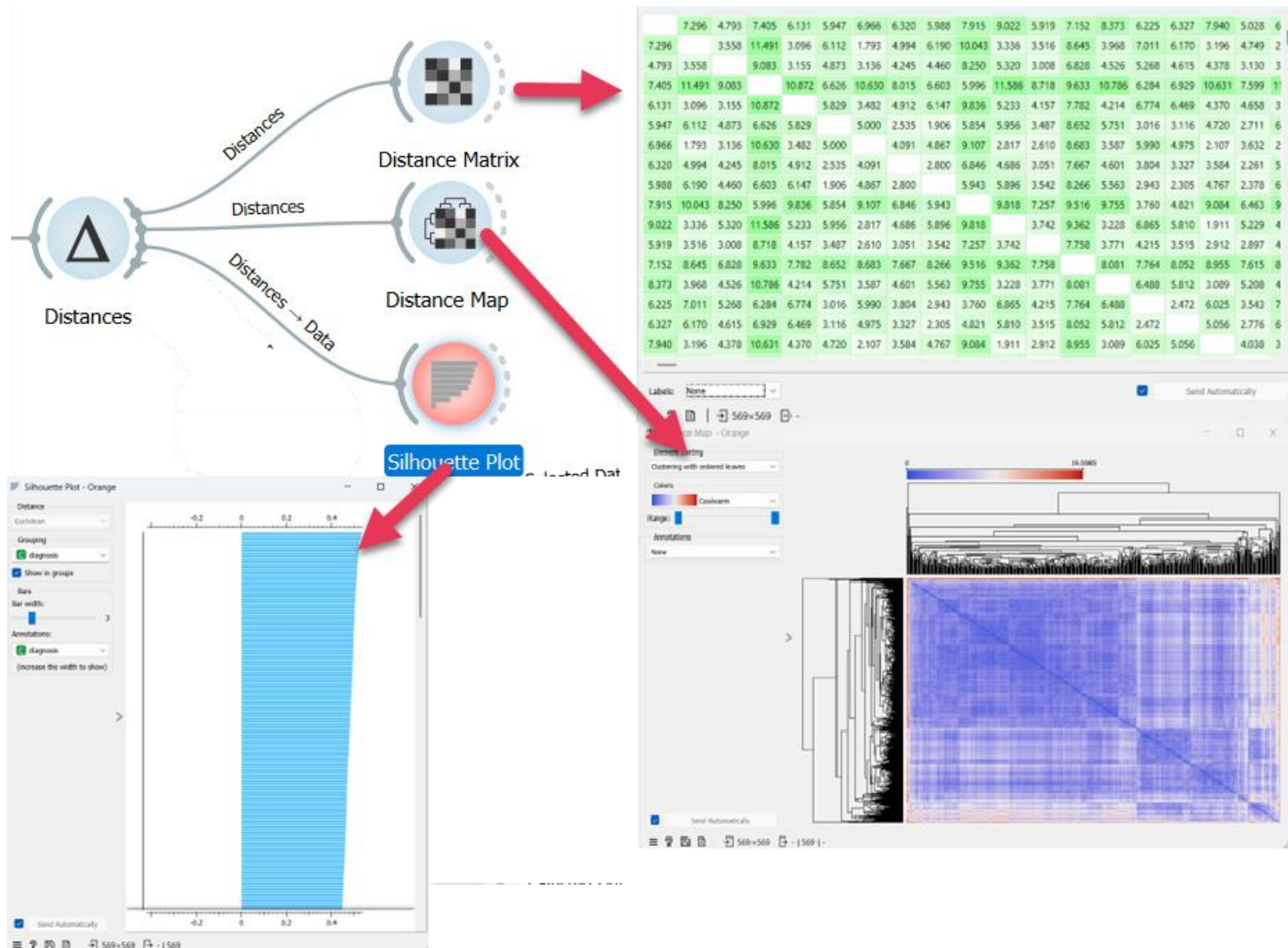
- Cluster analysis have an incredible wide range of applications and are quite useful to solve real world problems such as anomaly detection, recommending systems, documents grouping, or finding customers with common interests based on their purchases. Some of the most common clustering algorithms will be explored in the workshop, are:
 - K-Means
 - Hierarchical Clustering

Hierarchical Clustering

- Breast Cancer dataset
- **Distances** computes distances between rows or columns in a dataset. By default, the data will be normalized to ensure equal treatment of individual features. Normalization is always done column-wise.



View distance: How?

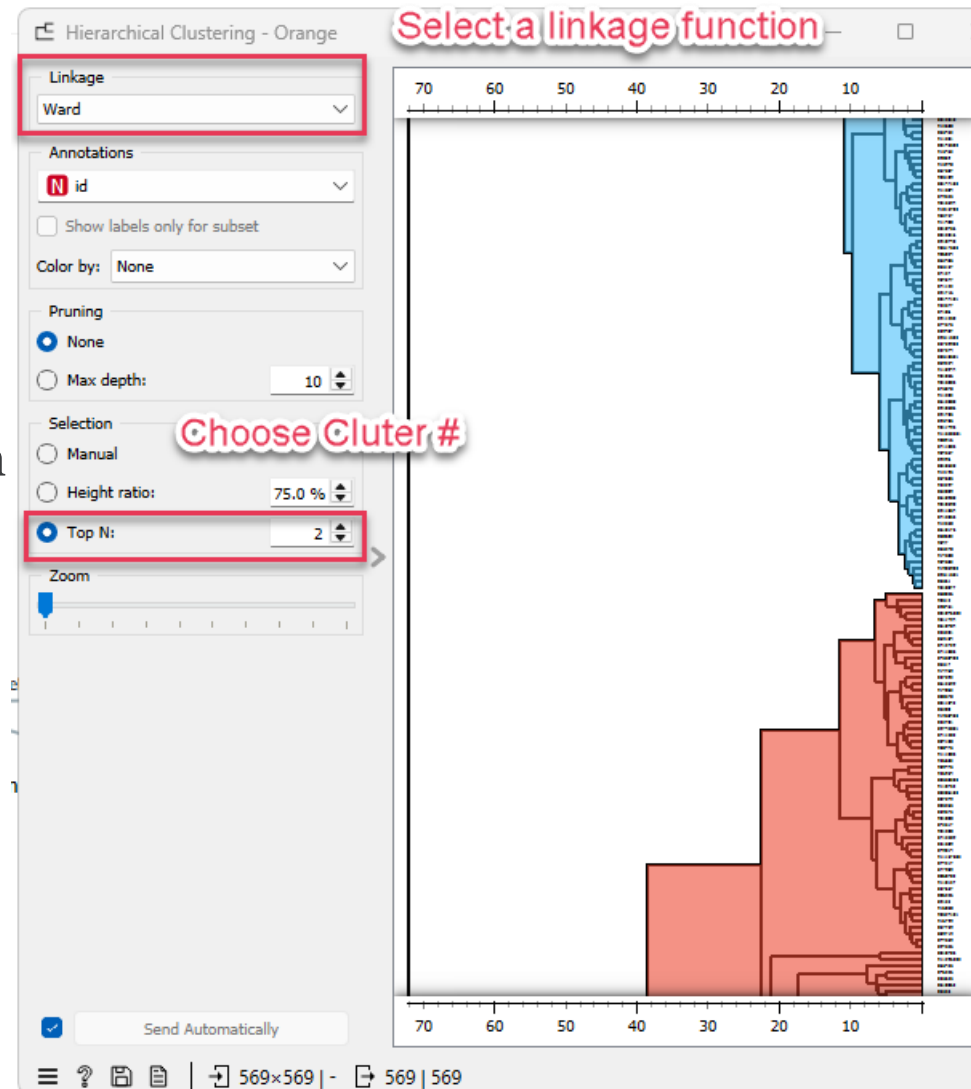


Silhouette score

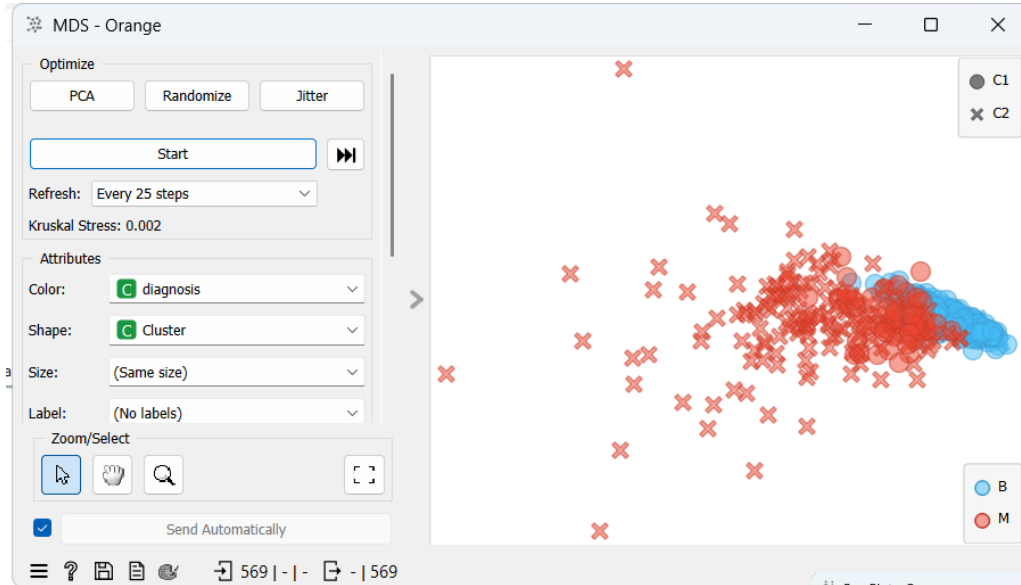
Dendrogram

Three different selection methods:

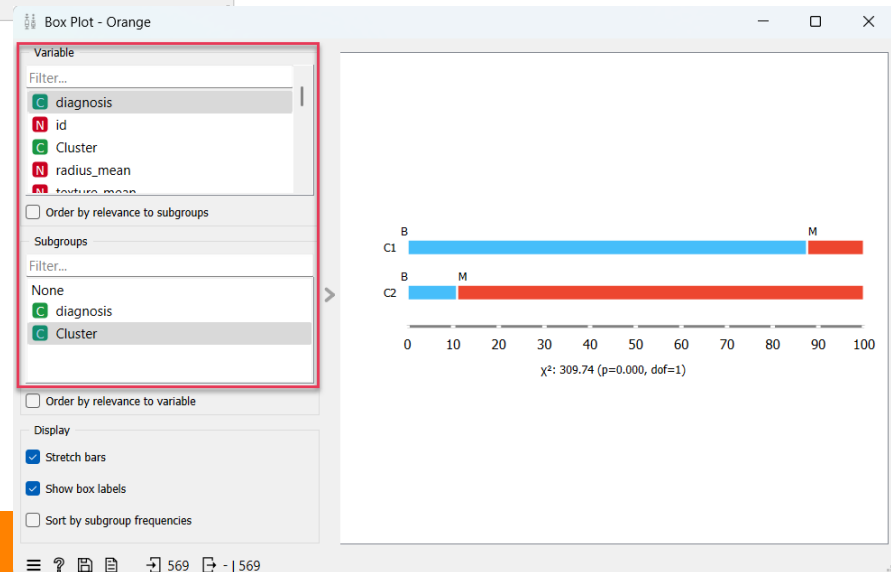
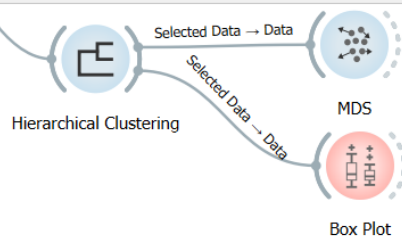
- **Manual:** clicking inside the dendrogram will select a cluster. Multiple clusters can be selected by holding Ctrl/Cmd. Each selected cluster is shown in a different color and is treated as a separate cluster in the output.
- **Height ratio:** clicking on the bottom or top ruler of the dendrogram places a cutoff line in the graph. Items to the right of the line are selected.
- **Top N:** selects the number of top nodes. Top N: 2 – two clusters



View Clusters-MDS and Box Plot

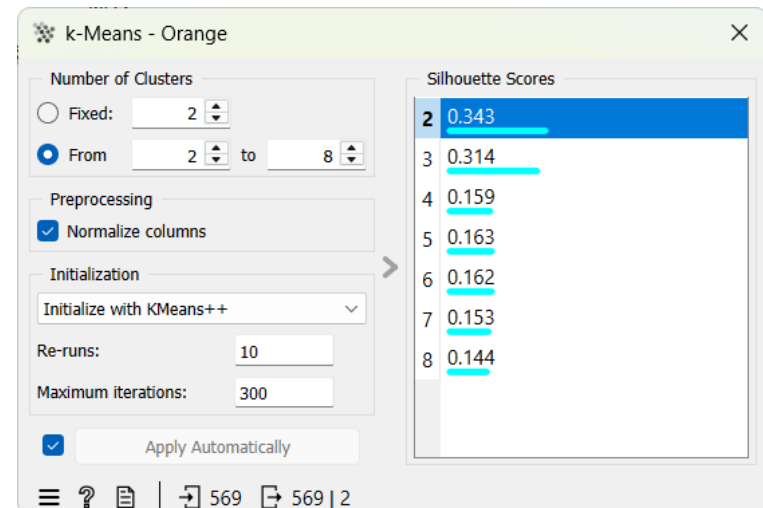
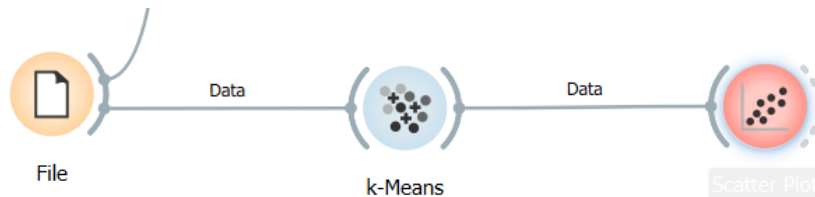


Multidimensional scaling (MDS) is a technique which finds a low-dimensional (in our case a two-dimensional) projection of points, where it tries to fit distances between points as well as possible.



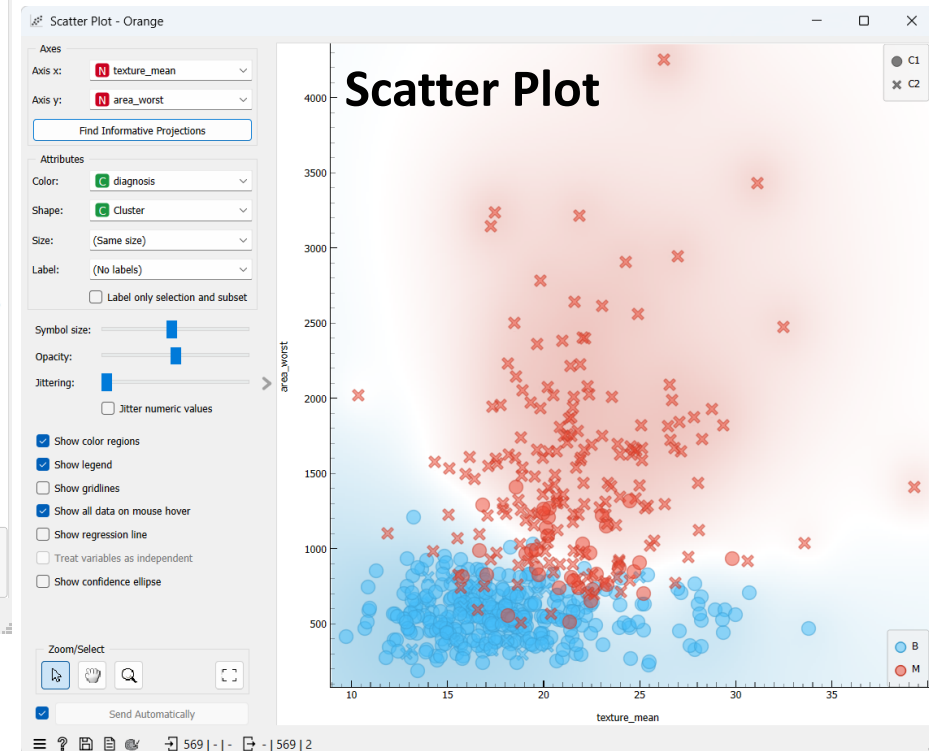
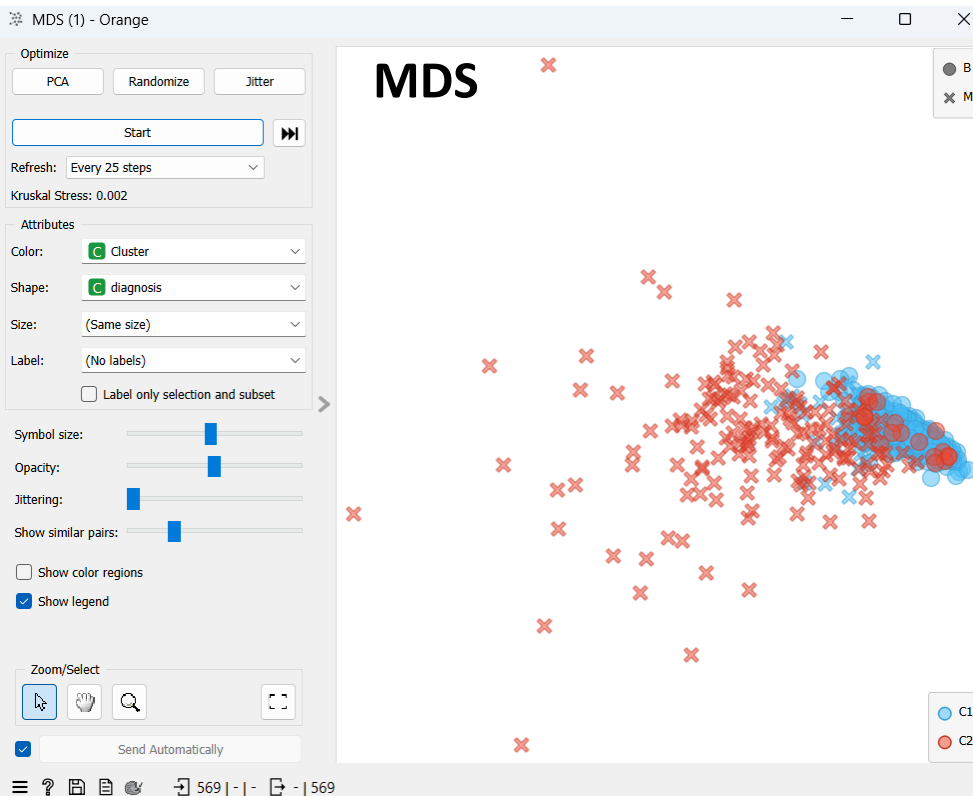
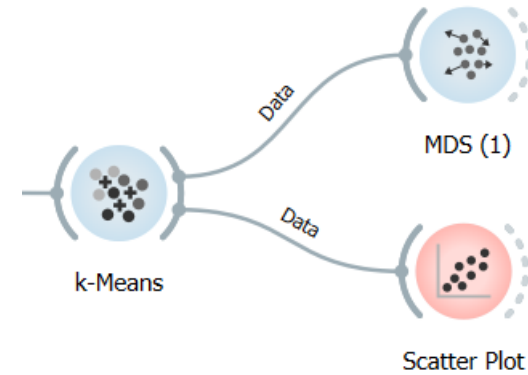
K-means Clustering

- Silhouette scores shows 2 is the best # of cluster
- The higher the silhouette score, the better the clustering.
- Preprocessing: If the option is selected, columns are normalized (mean centered to 0 and standard deviation scaled to 1).
 - Initialization method (the way the algorithm begins clustering)



View Clusters

- MDS and Scatter plot



Summary

Pros

- Open-source library
- User friendly and easy implement– visualize the data, ML procedures and interact with models
- Useful documentations, tutorials and examples
- Good for beginner or education purpose usage
- Many add-ons and still in development
- Fast computation speed

Cons

- Still has a lot of limitations (e.g. feature (domains and table), no built in Karas package, and not for large dataset)
- Python script console is hard to visualize the output. Jupyter notebook or spyder will be better to view the python commands and outputs
- Orange2.7 still available but not be included in Orange3
Orange2.7 contains Hyperparameter tuning, but Orange3 does not and cannot run hyperparameter tuning in Orange3
- Not for Advanced user. Some user says “Real life work is not like that, these tools (Orange) are just not enough.”

Thank You!

- For more information visit Orange3 website at:
 - <https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/index.html>
- To request RCS services, call the HelpDesk at:
 - [865-974-9900](tel:865-974-9900)

Questions?