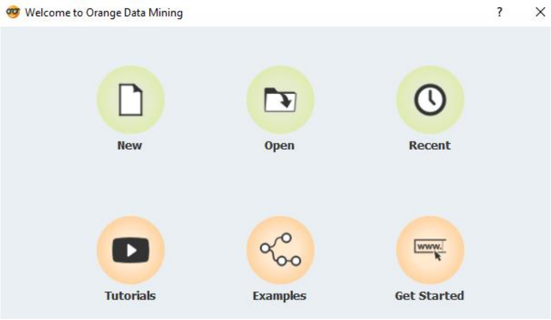
Introduction to Machine Learning with Orange

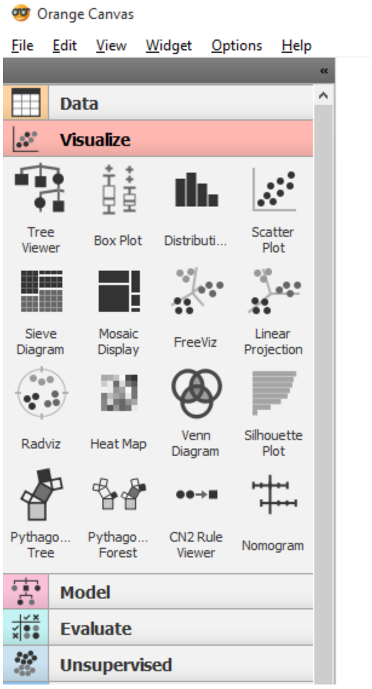
1. Getting Started with Orange

Orange (https://orangedatamining.com) is a machine learning and data visualization tool with interactive data analysis workflows which provides a large variety of tools for different types of data analytics such as data modelling, prediction, clustering, classification, neural networks, etc.

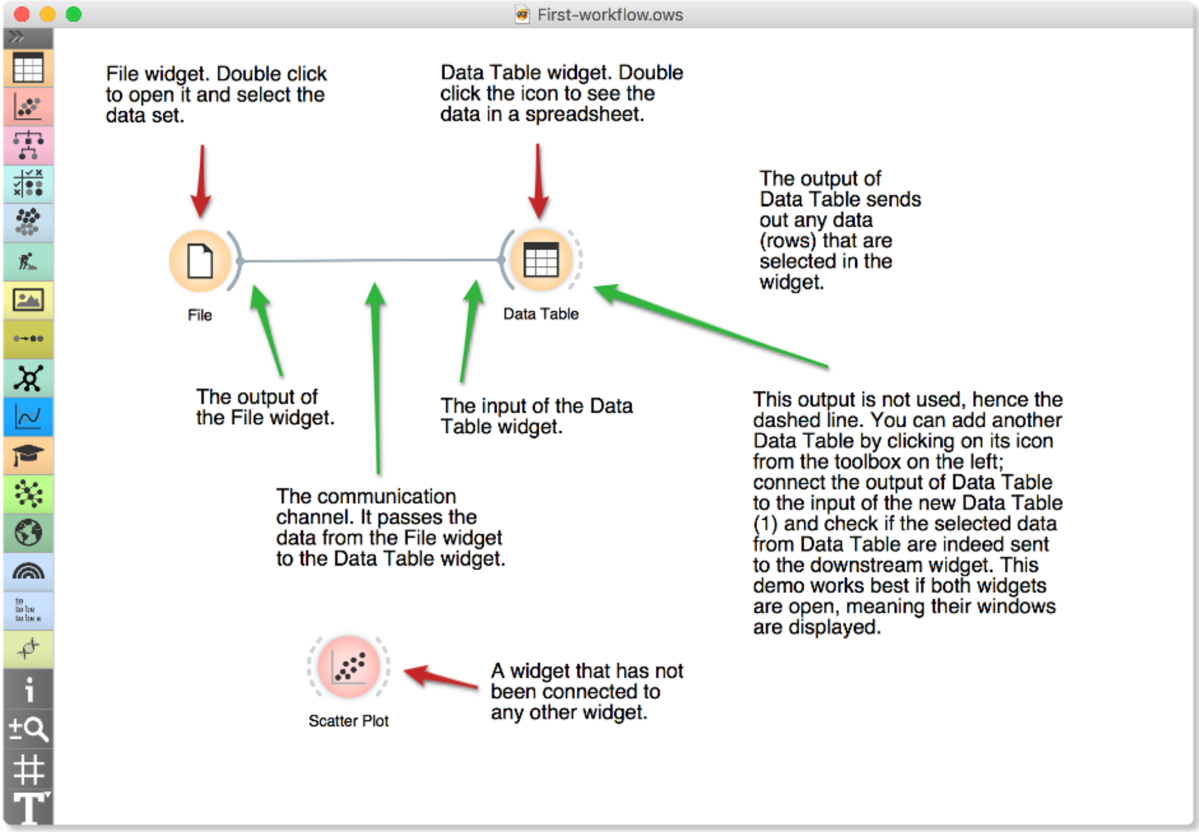
After starting up Orange from the desktop/start menu, you will see the welcome screen when you open Orange the first time. Click **New** to Start a new a recent data analysis workflow.



Orange workflows consist of components that read, process, and visualize data. We refer to these components as **Widgets.** The widgets are organized under different categories/tabs.



We can place the widgets on a drawing board called the "canvas" to design a workflow. Widgets communicate by sending information along their communication channel. Output from one widget can be used as input to another.



File widget. Double click to open it and select the data set.

Data Table widget. Double click the icon to see the data in a spreadsheet.

The output of Data Table sends out any data (rows) that are selected in the widget.

The output of the File widget.

The input of the Data Table widget.

This output is not used, hence the dashed line. You can add another Data Table by clicking on its icon from the toolbox on the left; connect the output of Data Table to the input of the new Data Table (1) and check if the selected data from Data Table are indeed sent to the downstream widget. This demo works best if both widgets are open, meaning their windows are displayed.

The communication channel. It passes the data from the File widget to the Data Table widget.

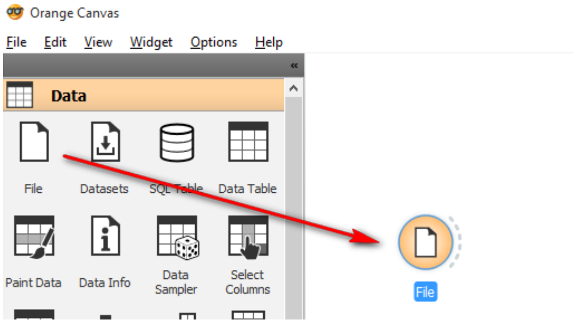
A widget that has not been connected to any other widget.

2. Data Exploration and Visualization

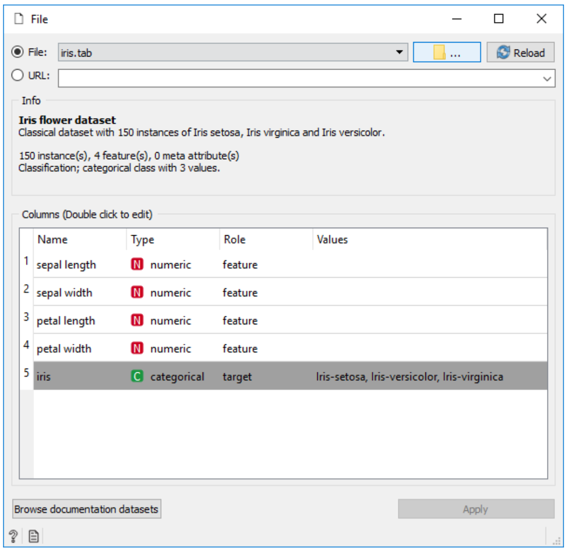
Preparation

In this step, we will set up a data analysis workflow. The famous Iris data set[1] is used for demonstration purpose.

Drag the **File** widget on **Data** tab to the canvas.



Double click the ***File*** widget in the canvas. Click **…** to browse the sample dataset and select the iris dataset (**iris.tab**).



● File: iris.tab

○ URL:

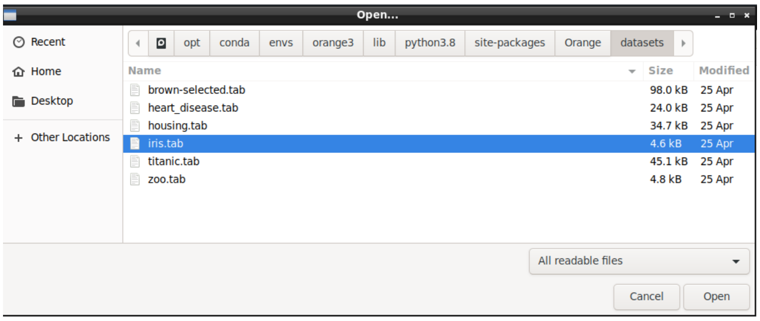
Info

Iris flower dataset
Classical dataset with 150 instances of Iris setosa, Iris virginica and Iris versicolor.

150 instance(s), 4 feature(s), 0 meta attribute(s)
Classification; categorical class with 3 values.

Columns (Double click to edit)

	Name	Type	Role	Values
1	sepal length	N numeric	feature	
2	sepal width	N numeric	feature	
3	petal length	N numeric	feature	
4	petal width	N numeric	feature	
5	iris	C categorical	target	Iris-setosa, Iris-versicolor, Iris-virginica



Name	Size	Modified
brown-selected.tab	98.0 kB	25 Apr
heart_disease.tab	24.0 kB	25 Apr
housing.tab	34.7 kB	25 Apr
iris.tab	4.6 kB	25 Apr
titanic.tab	45.1 kB	25 Apr
zoo.tab	4.8 kB	25 Apr

[1] The Iris data set, introduced in 1936 by Ronald Fisher, a British statistician and biologist, consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor) with 4 measured features from each sample: the length and the width of the sepals and petals, in centimeters.

About the column (variable) types:

- **Categorical**: This refers to a variable with discrete values, for example eye color, nationality, ticket class (first, second, third), and so on. In Orange, these would be marked with a green C.
- **Numeric** or **continuous** variable: This refers to a variable with numbers as values, for example cholesterol level, heart rate, age, and so on. In Orange, these would be marked with a red N.

About the column (variable) roles:

- **Feature**: In statistics they would be called **independent variables**. This all refers to descriptions of data samples. If you have patients in rows, that variables are in columns and they describe these patients (i.e. with name, date of birth, cholesterol level, heart rate, and so on).
- **Target**: Define the target variable or **dependent variable**. This is the variable you are trying to predict, for example the survival of Titanic passengers or the price of a house in Boston.

Close the widget after the file is selected.

Click on ***File** widget* in the data widgets menu. Add a **Data Table** widget to the canvas. Connect the two widgets by dragging a line from the *output channel* of file to the *input channel* of Data Table.

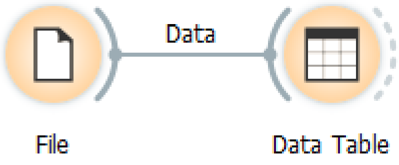
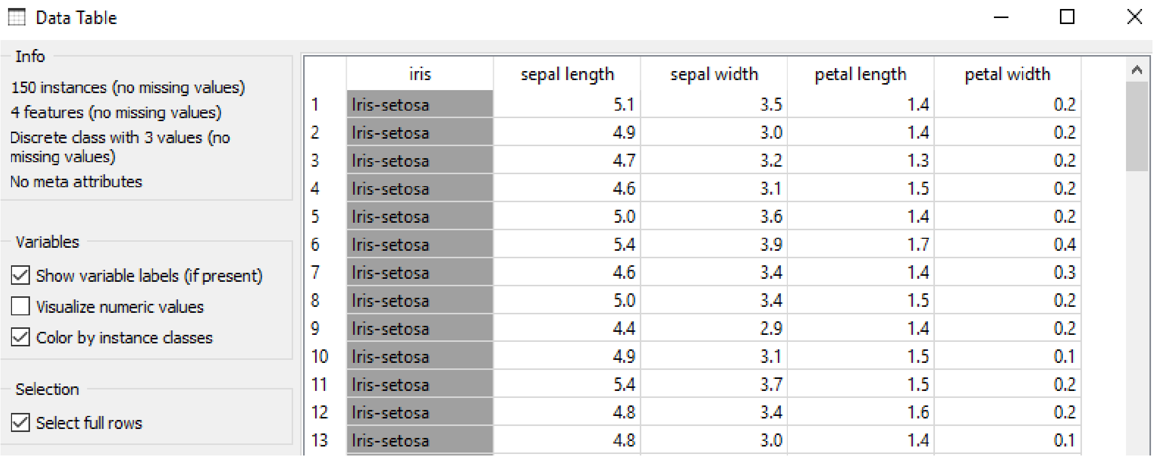


Fig. 1: File to Data Table Canvas Configuration

Double-click the **Data table** widget in the canvas to show the Iris data set. The dataset has 150 instances and 4 features.



Data Table

Info

150 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables

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

Selection

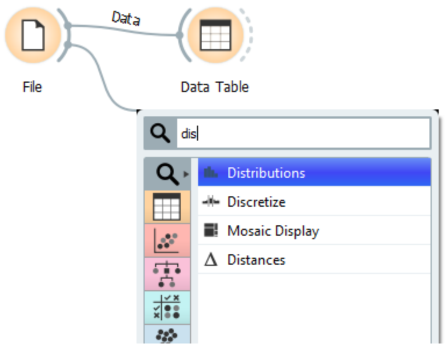
☑ Select full rows

	iris	sepal length	sepal width	petal length	petal width
1	Iris-setosa	5.1	3.5	1.4	0.2
2	Iris-setosa	4.9	3.0	1.4	0.2
3	Iris-setosa	4.7	3.2	1.3	0.2
4	Iris-setosa	4.6	3.1	1.5	0.2
5	Iris-setosa	5.0	3.6	1.4	0.2
6	Iris-setosa	5.4	3.9	1.7	0.4
7	Iris-setosa	4.6	3.4	1.4	0.3
8	Iris-setosa	5.0	3.4	1.5	0.2
9	Iris-setosa	4.4	2.9	1.4	0.2
10	Iris-setosa	4.9	3.1	1.5	0.1
11	Iris-setosa	5.4	3.7	1.5	0.2
12	Iris-setosa	4.8	3.4	1.6	0.2
13	Iris-setosa	4.8	3.0	1.4	0.1

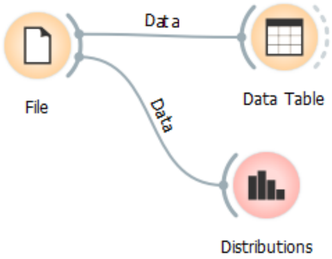
Distribution

Create a **Distributions** widget and link to the **File** widget as follows. From **File** widget, drag from the right side of the widget and type the name of the widget and select the appropriate widget.

In orange, you may extend a workflow by dragging a line from the output channel of a widget and select the widget that you want to add. You may type the name of the widget for filtering.

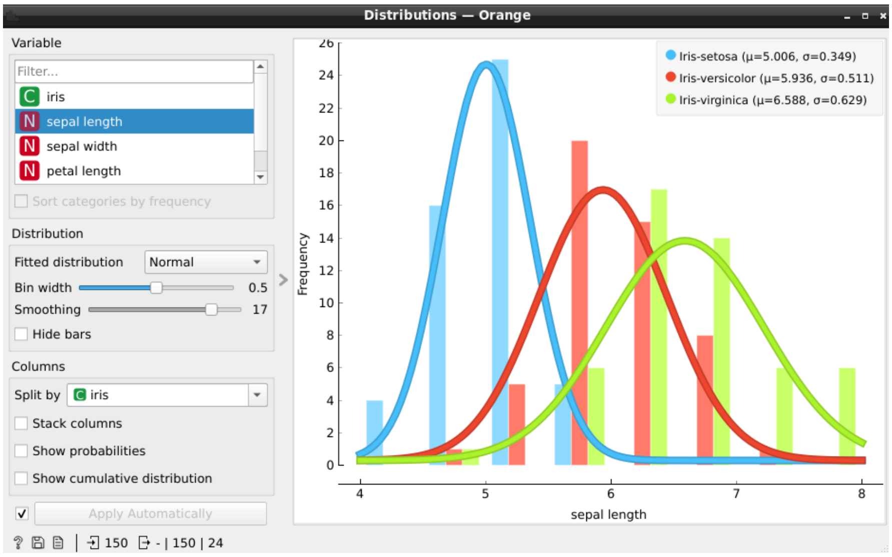


Double click the **Distribution** widget.



Set **Split by** to **iris** and **Fitted distribution** to **Normal**.

- Explore the distribution of the different attributes (e.g. sepal length) across the three different species of Iris.
- Change the bin width and smoothing parameters and observe the output.



Distribution

Fitted distribution: Normal

Bin width: 0.5

Smoothing: 17

□ Hide bars

Columns

Split by: iris

□ Stack columns

□ Show probabilities

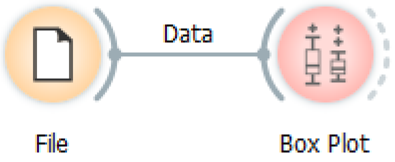
□ Show cumulative distribution

☑ Apply Automatically

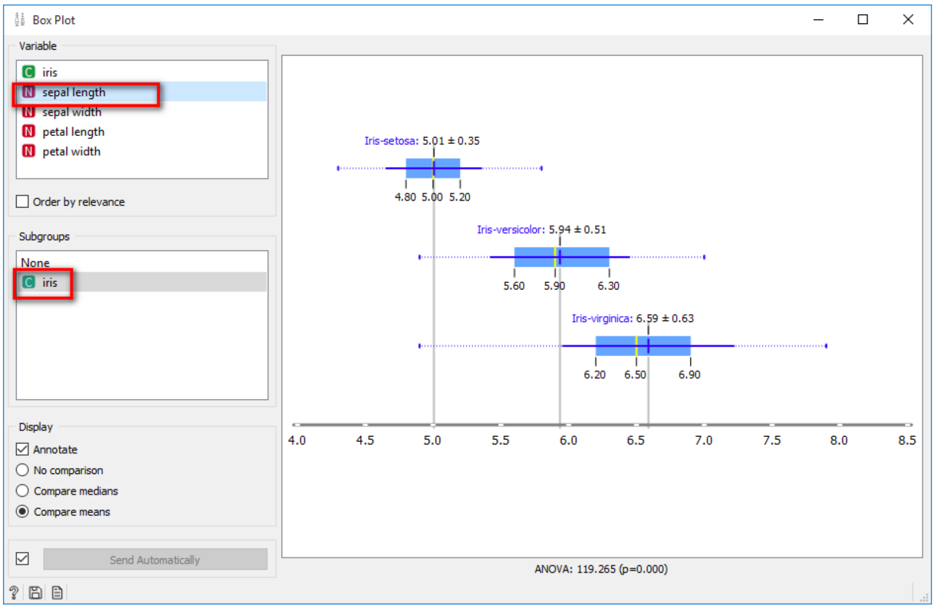
150 | - | 150 | 24

Box Plot

Box plots shows the distribution of attribute values and is useful for identifying outliers. Modify the workflow as follows.



Double click **Box Plot**. You may select an attribute (e.g. sepal length), and analyse the distribution of this attribute across different subgroups (in this example, iris species).


Box Plot

Variable

- C iris
- N sepal length
- N sepal width
- N petal length
- N petal width

☐ Order by relevance

Subgroups

- None
- C iris

Display

☑ Annotate

○ No comparison

○ Compare medians

● Compare means

☑ Send Automatically

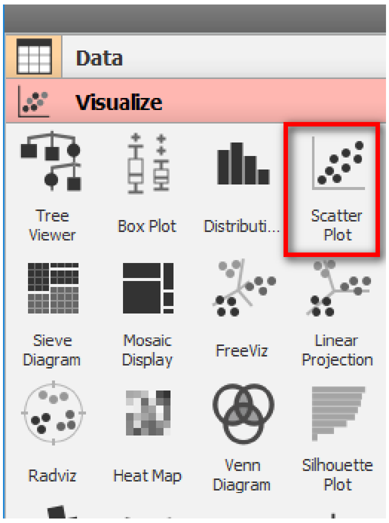
ANOVA: 119.265 (p=0.000)

For continuous attributes, the box plots shows

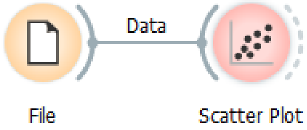
- The mean (blue vertical line)
- The blue highlighted area is the entire standard deviation of the mean.
- The median (yellow vertical line)
- The thin blue line represents the area between the first (25%) and the third (75%) quantile
- The thin dotted line represents the entire range of values (from the lowest to the highest value in the data set for the selected parameter).

Scatter Plot

Under **Visualize** tab in the widget menu, insert a **Scatter plot**.

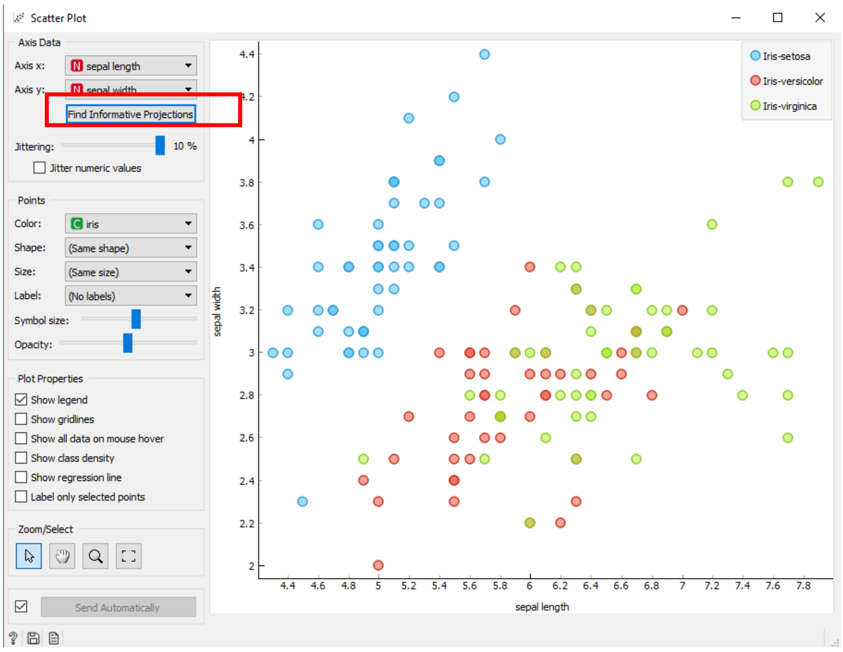


Set up the following workflow.



Double click the scatter plot widget in the canvas. Notice that the three species of iris are not clearly separated from each others.

Click ***Find Information Projections*** in the top left corner. Click **Start**.



Axis x: N sepal length

Axis y: N sepal width

Jittering: 10 %

☐ Jitter numeric values

Color: C iris

Shape: (Same shape)

Size: (Same size)

Label: (No labels)

Symbol size:

Opacity:

☑ Show legend

☐ Show gridlines

☐ Show all data on mouse hover

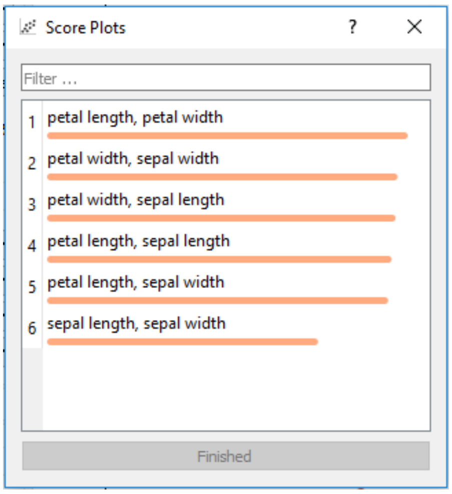
☐ Show class density

☐ Show regression line

☐ Label only selected points

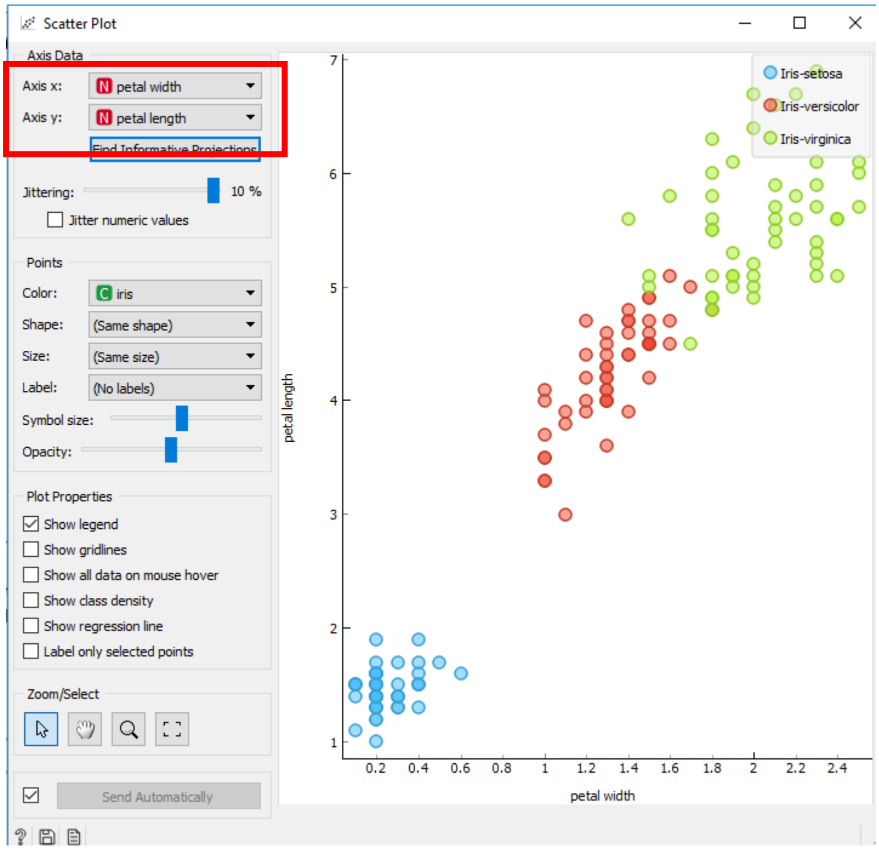
☑ Send Automatically

The rank of the result indicates how good the attribute pair can separate the species in the scatter plot. The pair "Petal length and petal width" fits the best in this case.



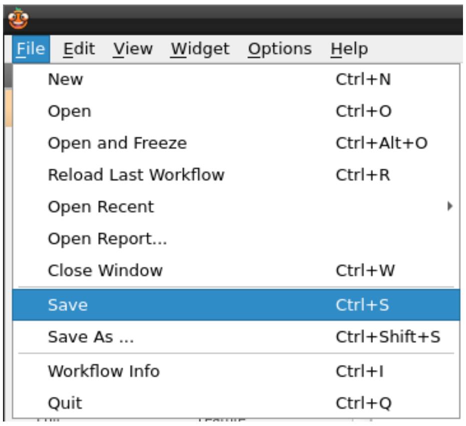
1 petal length, petal width
2 petal width, sepal width
3 petal width, sepal length
4 petal length, sepal length
5 petal length, sepal width
6 sepal length, sepal width

Double click the first entry and close the popup window.



Axis x: N petal width
Axis y: N petal length
Jittering: 10 %
☐ Jitter numeric values
Color: C iris
Shape: (Same shape)
Size: (Same size)
Label: (No labels)
Symbol size:
Opacity:
☑ Show legend
☐ Show gridlines
☐ Show all data on mouse hover
☐ Show class density
☐ Show regression line
☐ Label only selected points
☑ Send Automatically

From **File** menu, click **Save**.

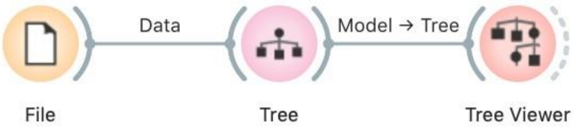


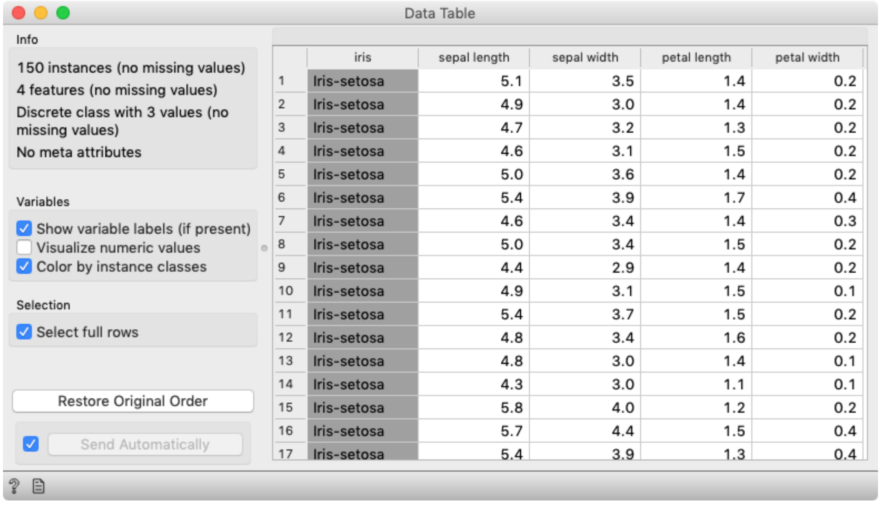
Save the model as “**iris visualization.ows”** on your desktop.

3. Decision Tree Classifier

Decision tree is one of the oldest, but still popular, machine learning methods. We will construct a decision tree to predict the species of iris.

Select **File→New** to start a new canvas.
Let us load *iris* data set, build a tree (widget *Tree*) and visualize it in a *Tree Viewer*.





Info

150 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
No meta attributes

Variables

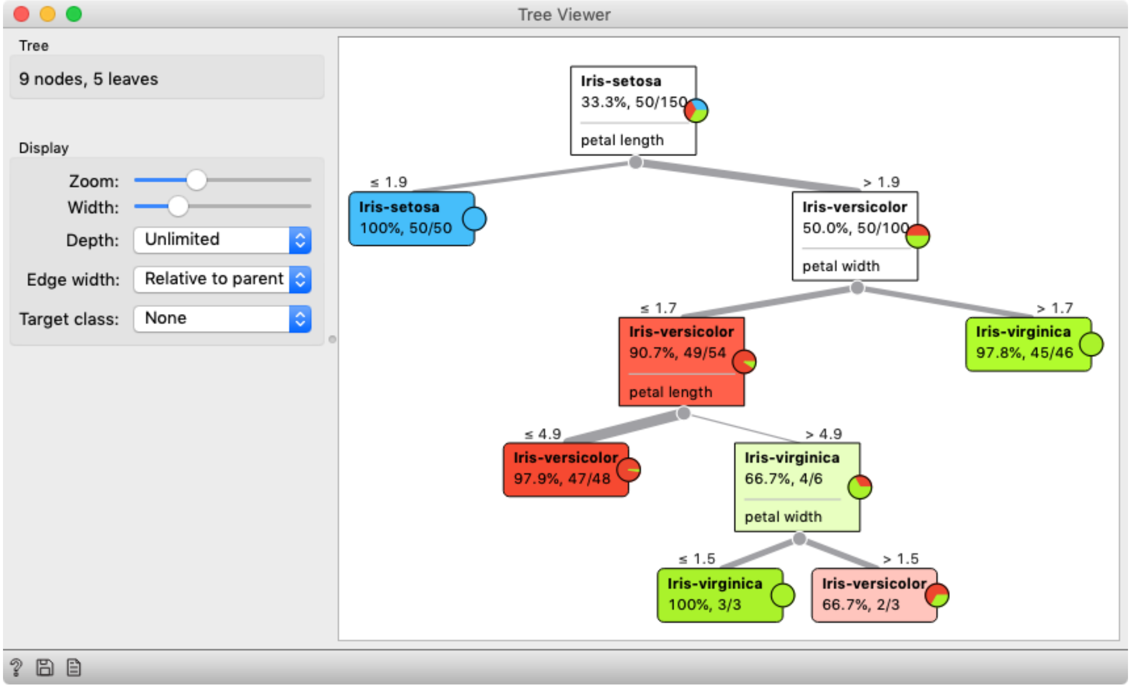
☑ Show variable labels (if present)
□ Visualize numeric values
☑ Color by instance classes

Selection

☑ Select full rows

	iris	sepal length	sepal width	petal length	petal width
1	Iris-setosa	5.1	3.5	1.4	0.2
2	Iris-setosa	4.9	3.0	1.4	0.2
3	Iris-setosa	4.7	3.2	1.3	0.2
4	Iris-setosa	4.6	3.1	1.5	0.2
5	Iris-setosa	5.0	3.6	1.4	0.2
6	Iris-setosa	5.4	3.9	1.7	0.4
7	Iris-setosa	4.6	3.4	1.4	0.3
8	Iris-setosa	5.0	3.4	1.5	0.2
9	Iris-setosa	4.4	2.9	1.4	0.2
10	Iris-setosa	4.9	3.1	1.5	0.1
11	Iris-setosa	5.4	3.7	1.5	0.2
12	Iris-setosa	4.8	3.4	1.6	0.2
13	Iris-setosa	4.8	3.0	1.4	0.1
14	Iris-setosa	4.3	3.0	1.1	0.1
15	Iris-setosa	5.8	4.0	1.2	0.2
16	Iris-setosa	5.7	4.4	1.5	0.4
17	Iris-setosa	5.4	3.9	1.3	0.4

Double click and open the **Tree viewer**.



Tree

9 nodes, 5 leaves

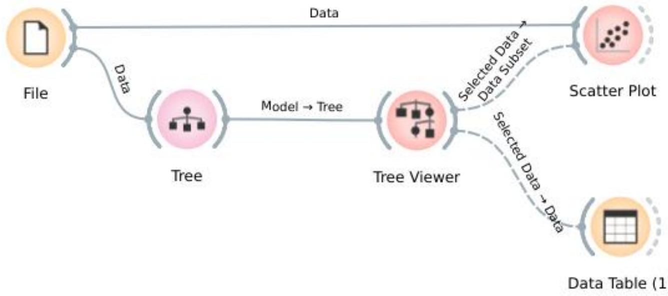
Display

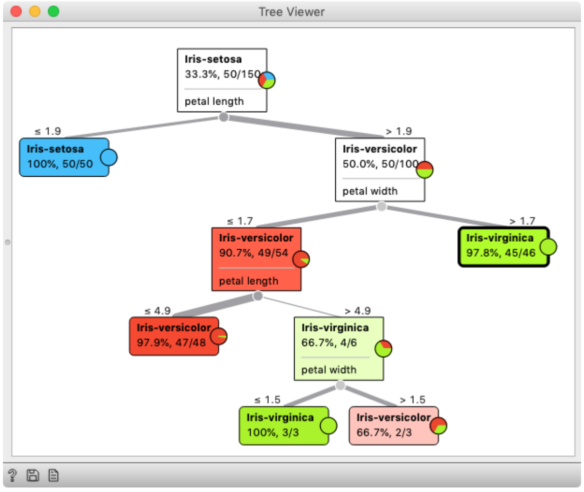
Zoom:
Width:
Depth: Unlimited
Edge width: Relative to parent
Target class: None

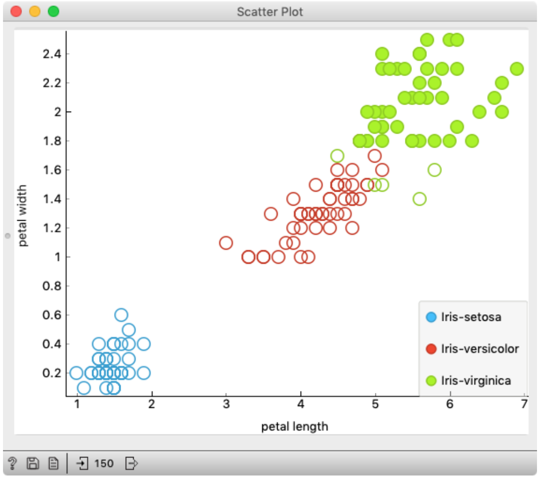
- We can read the tree from top to bottom. In this example, the column *petal length* best separates the iris variety *setosa* from the others, and in the next step, *petal width* then almost perfectly separates the remaining two varieties.
- Trees place the most useful feature at the root based on how well they distinguish between classes. It then splits both subsets further, again by their most useful features, and keeps doing so until it reaches subsets in which all data belongs to the same class (leaf nodes in strong blue or red) or until it runs out of data instances to split or out of useful features (the two leaf nodes in white).

Create the workflow as shown in the diagram below.

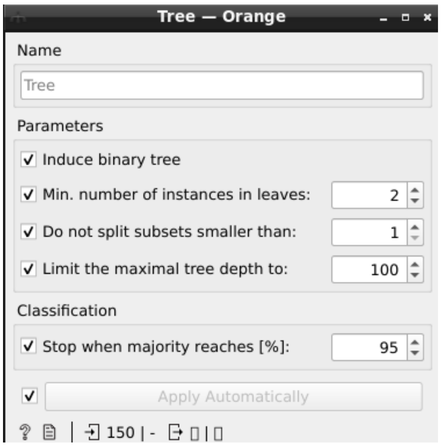
- Select the *Iris* data set in the **File** widget. In the *Scatter Plot*, we first find the best visualization of this data set, that is, the one that best separates the instances from different classes.
- Then we connect the *Tree Viewer* to the *Scatter Plot*. Data instances (particular irises) from the selected node in the *Tree Viewer* are shown in the *Scatter Plot*.







Click the **Tree** widget and review the hyper-parameters of the decision tree. Change the **“Min. number of instances in leaves”** to 1 and observe the change in the decision tree.



Name

Tree

Parameters

☑ Induce binary tree

☑ Min. number of instances in leaves: 2

☑ Do not split subsets smaller than: 1

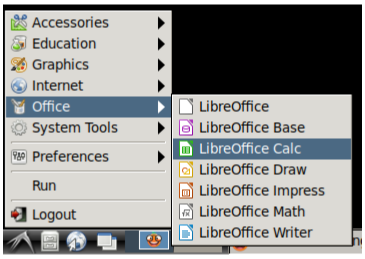
☑ Limit the maximal tree depth to: 100

Classification

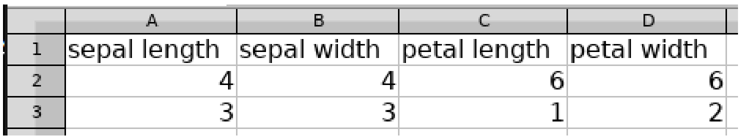
☑ Stop when majority reaches [%]: 95

☑ Apply Automatically

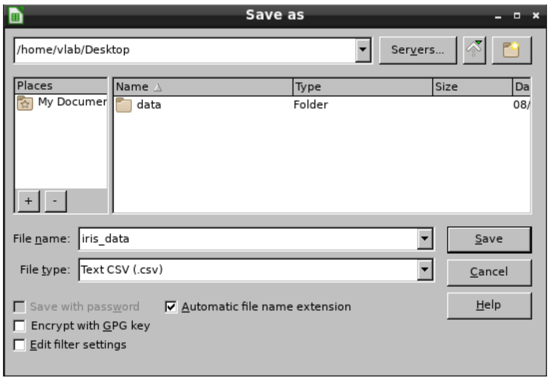
Start **LibreOffice Calc** from Start Menu (If you are using MS Windows, you use may use **MS Excel**).



Create a spreadsheet as follows. Save the file as **data.csv** on your desktop. Click **Use Text CSV Format** (if you are using the OpenOffice calc).



	A	B	C	D
1	sepal length	sepal width	petal length	petal width
2	4	4	6	6
3	3	3	1	2



/home/vlab/Desktop

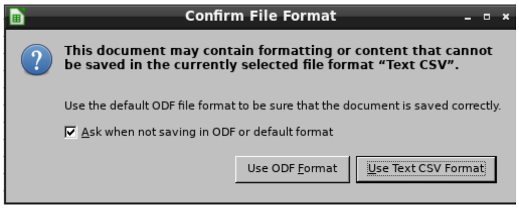
File name: iris_data

File type: Text CSV (.csv)

□ Save with password ☑ Automatic file name extension

□ Encrypt with GPG key

□ Edit filter settings

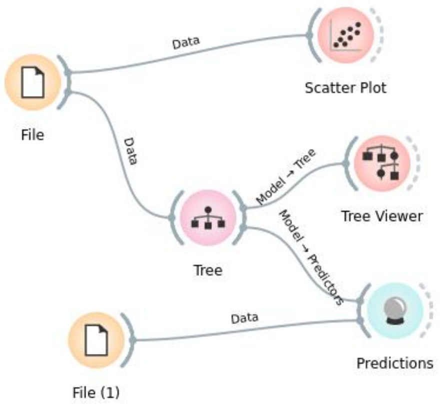


This document may contain formatting or content that cannot be saved in the currently selected file format “Text CSV”.

Use the default ODF file format to be sure that the document is saved correctly.

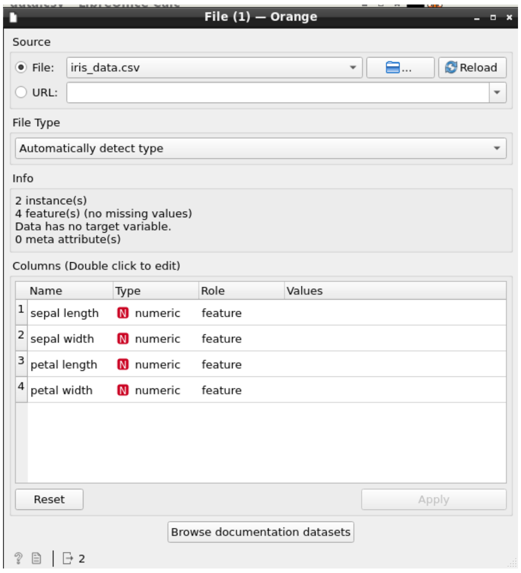
☑ Ask when not saving in ODF or default format

Create the workflow as shown in the figure below.



Add a **Predictions** widget and another **File** widget to the canvas and connect the workflow as follows. There should be two **File** Widgets in the workflow.

- The **File** widget one on the top is used to train a decision tree classifier.
- The **File** widget (File 1) at the bottom is used to provide the data for prediction. Double click it and select **iris_data.csv** that you have created.



Source

● File: iris_data.csv

○ URL:

File Type

Automatically detect type

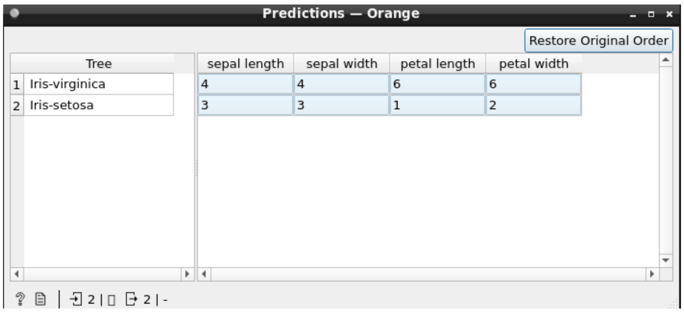
Info

2 instance(s)
4 feature(s) (no missing values)
Data has no target variable.
0 meta attribute(s)

Columns (Double click to edit)

	Name	Type	Role	Values
1	sepal length	N numeric	feature	
2	sepal width	N numeric	feature	
3	petal length	N numeric	feature	
4	petal width	N numeric	feature	

Click the **prediction** widget on the canvas to view the predicted class for the various animals in the given dataset.



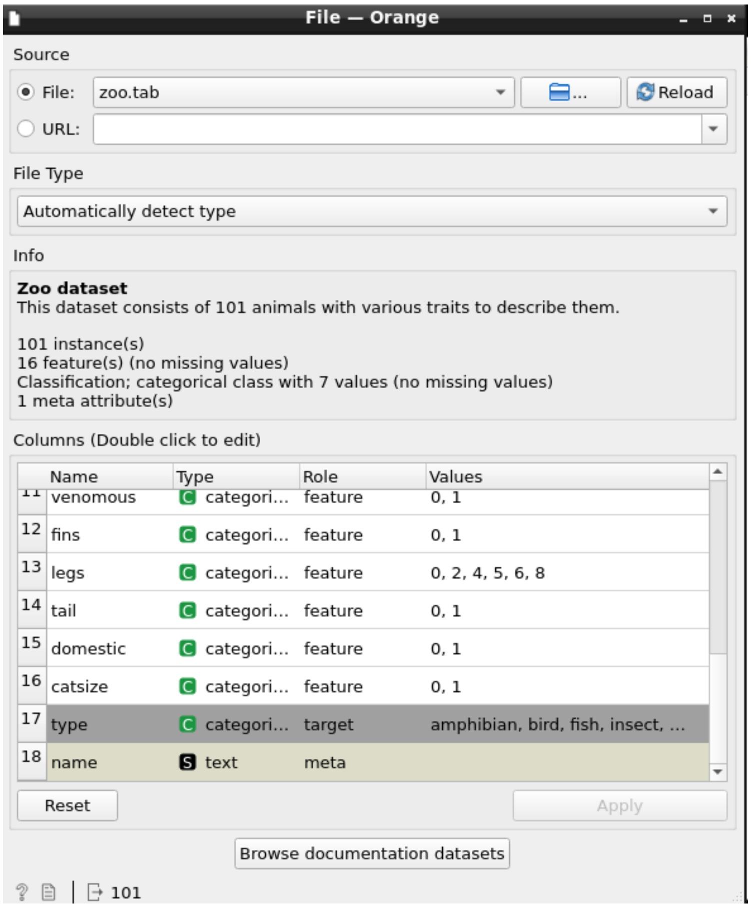
	Tree	sepal length	sepal width	petal length	petal width
1	Iris-virginica	4	4	6	6
2	Iris-setosa	3	3	1	2

4. The Zoo Dataset

In this section, we will explore the zoo dataset and develop a decision tree classification model to predict the type of animal (e.g. Mammal, Bird, Reptile, Fish, Amphibian, Insect and Invertebrate) given various features of the animals.

Start a new workflow (**File -> New**). Add a **File** widget.

Click **…** to browse the sample dataset. Choose the ***zoo data set (zoo.tab)*** as the data source.



Source

● File: zoo.tab

○ URL:

File Type

Automatically detect type

Info

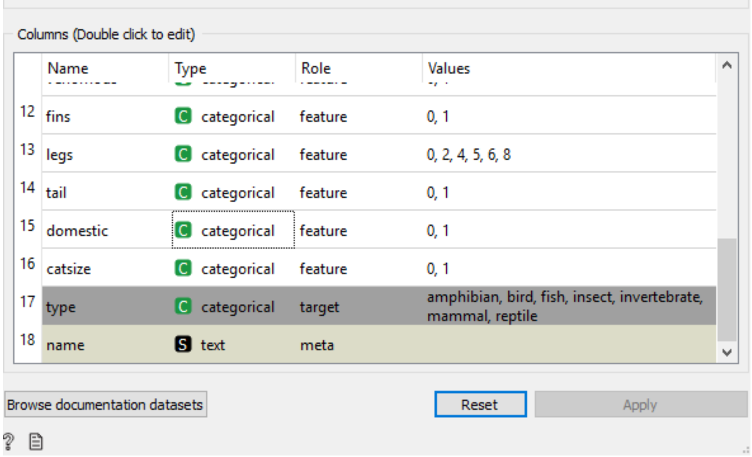
Zoo dataset
This dataset consists of 101 animals with various traits to describe them.

101 instance(s)
16 feature(s) (no missing values)
Classification; categorical class with 7 values (no missing values)
1 meta attribute(s)

Columns (Double click to edit)

	Name	Type	Role	Values
11	venomous	C categori…	feature	0, 1
12	fins	C categori…	feature	0, 1
13	legs	C categori…	feature	0, 2, 4, 5, 6, 8
14	tail	C categori…	feature	0, 1
15	domestic	C categori…	feature	0, 1
16	catsize	C categori…	feature	0, 1
17	type	C categori…	target	amphibian, bird, fish, insect, …
18	name	S text	meta	

Scroll down the column windows to view the “type” column which is a “target” variable. This indicates that we are going to build a machine-learning model to predict the type of animal given the various “feature” columns.



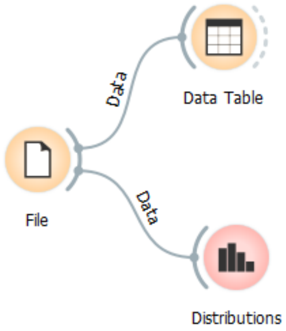
Columns (Double click to edit)

	Name	Type	Role	Values
12	fins	C categorical	feature	0, 1
13	legs	C categorical	feature	0, 2, 4, 5, 6, 8
14	tail	C categorical	feature	0, 1
15	domestic	C categorical	feature	0, 1
16	catsize	C categorical	feature	0, 1
17	type	C categorical	target	amphibian, bird, fish, insect, invertebrate, mammal, reptile
18	name	S text	meta	

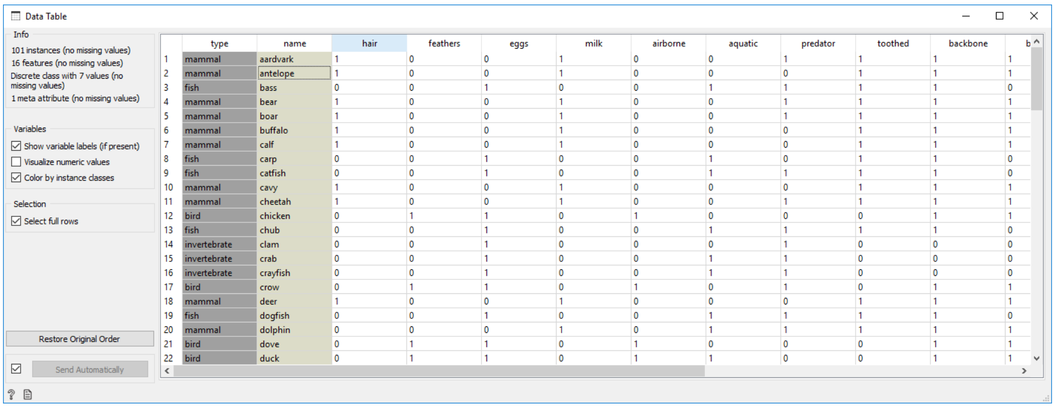
Remark:

- **string** or **text**: variables with text as value, for example the address, the ID of a product, or the name of the person.
- **meta variable**: variables that adds additional information on a data sample, but it is not used in computation (e.g. classification and clustering)

Create the following workflow in the canvas.



Double click the **Data Table** widget to view the data.



Data Table

Info

101 instances (no missing values)
16 features (no missing values)
Discrete class with 7 values (no missing values)
1 meta attribute (no missing values)

Variables

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

Selection

☑ Select full rows

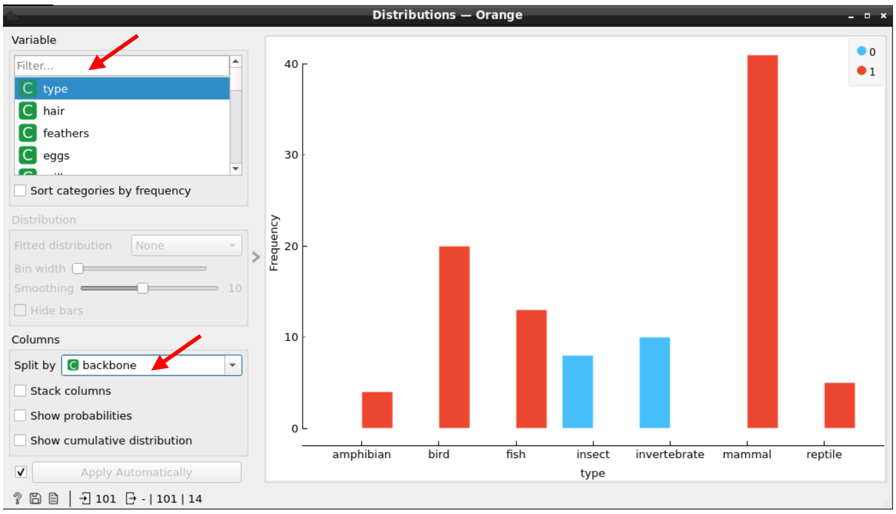
Restore Original Order

☑ Send Automatically

	type	name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	b
1	mammal	aardvark	1	0	0	1	0	0	1	1	1	1
2	mammal	antelope	1	0	0	1	0	0	0	1	1	1
3	fish	bass	0	0	1	0	0	1	1	1	1	0
4	mammal	bear	1	0	0	1	0	0	1	1	1	1
5	mammal	boar	1	0	0	1	0	0	1	1	1	1
6	mammal	buffalo	1	0	0	1	0	0	0	1	1	1
7	mammal	calf	1	0	0	1	0	0	0	1	1	1
8	fish	carp	0	0	1	0	0	1	0	1	1	0
9	fish	catfish	0	0	1	0	0	1	1	1	1	0
10	mammal	cavy	1	0	0	1	0	0	0	1	1	1
11	mammal	cheetah	1	0	0	1	0	0	1	1	1	1
12	bird	chicken	0	1	1	0	1	0	0	0	1	1
13	fish	chub	0	0	1	0	0	1	1	1	1	0
14	invertebrate	clam	0	0	1	0	0	0	1	0	0	0
15	invertebrate	crab	0	0	1	0	0	1	1	0	0	0
16	invertebrate	crayfish	0	0	1	0	0	1	1	0	0	0
17	bird	crow	0	1	1	0	1	0	1	0	1	1
18	mammal	deer	1	0	0	1	0	0	0	1	1	1
19	fish	dogfish	0	0	1	0	0	1	1	1	1	0
20	mammal	dolphin	0	0	0	1	0	1	1	1	1	1
21	bird	dove	0	1	1	0	1	0	0	0	1	1
22	bird	duck	0	1	1	0	1	1	0	0	1	1

Close the **Data Table**.

Double click the **Distribution Table**. Select “type” under **variable** and “backbone” under **Split by**.



Distributions — Orange

Variable

Filter...

type
hair
feathers
eggs

☐ Sort categories by frequency

Distribution

Fitted distribution: None

Bin width

Smoothing 10

☐ Hide bars

Columns

Split by: backbone

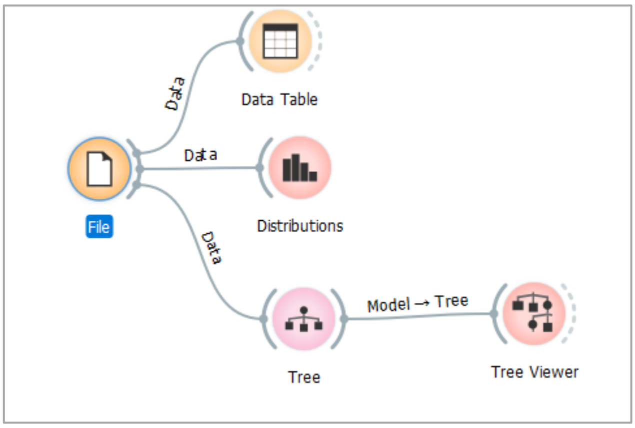
☐ Stack columns
☐ Show probabilities
☐ Show cumulative distribution

☑ Apply Automatically

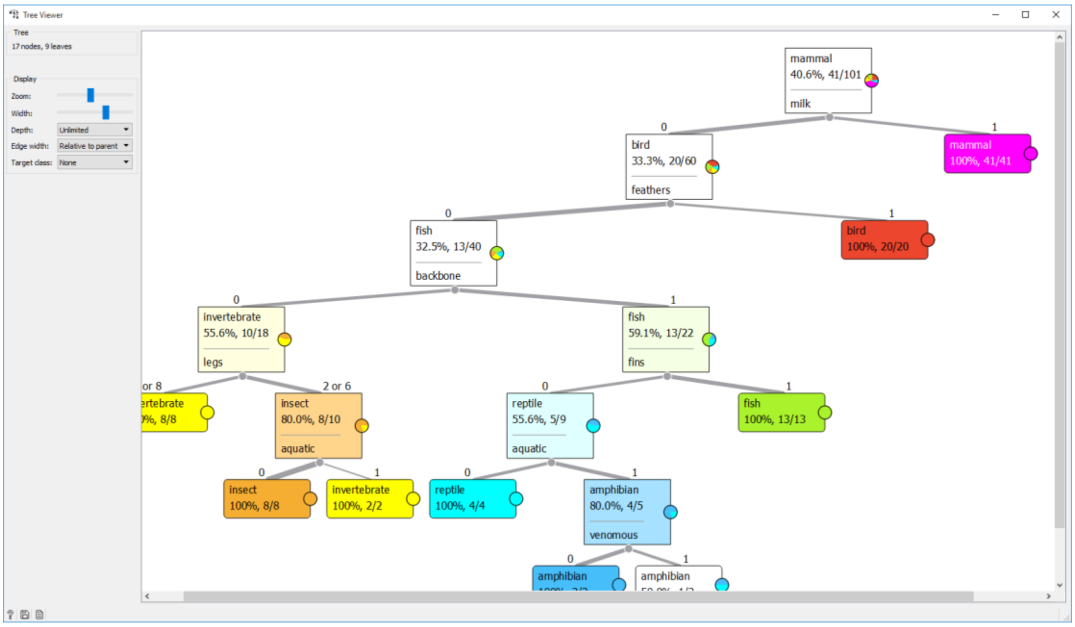
101 | - | 101 | 14

Close the **Distribution Table**.

Feed the data from the zoo dataset to the ***Tree*** widget (under the ***Model*** *category* from the *widget menus)*. Connect the output of the tree widget to a tree viewer widget (under the **visualize** category from the widget menu). The resulting configuration is shown below.



Click the **Tree Viewer** to view the classification tree.



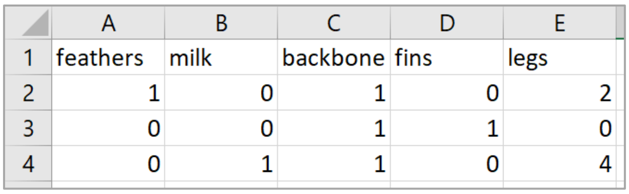
Tree Viewer

Tree
17 nodes, 9 leaves

Display
Zoom:
Width:
Depth: Unlimited
Edge width: Relative to parent
Target class: None

Save the model as “**zoo classification.ows**”.

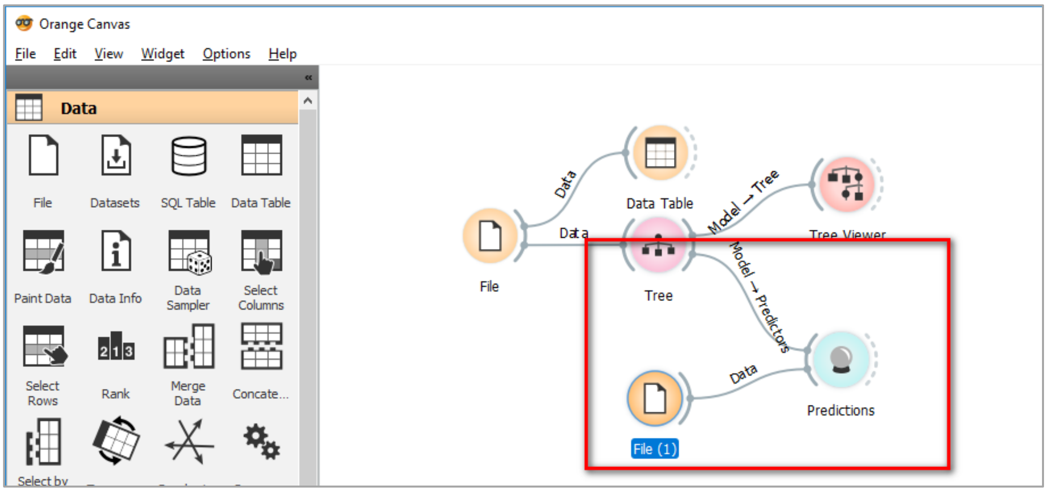
Use **OpenOffice Calc** or **Excel** to create a spreadsheet as follows.



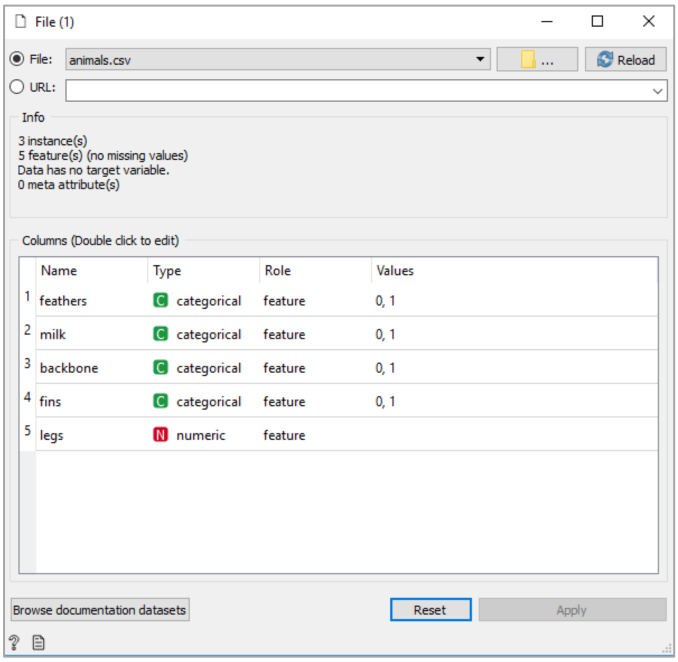
	A	B	C	D	E
1	feathers	milk	backbone	fins	legs
2	1	0	1	0	2
3	0	0	1	1	0
4	0	1	1	0	4

Save the file as **animals.csv** on your desktop.

Add a **Predictions** widget and another **File** widget to the canvas and connect the workflow as follows.



Double click the newly created **File** widget (File (1)) and select **animals.csv** that you have created.



● File: animals.csv

○ URL:

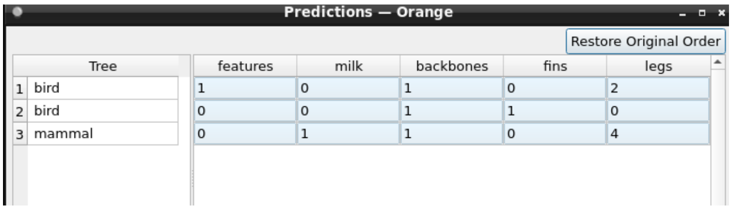
Info

3 instance(s)
5 feature(s) (no missing values)
Data has no target variable.
0 meta attribute(s)

Columns (Double click to edit)

	Name	Type	Role	Values
1	feathers	C categorical	feature	0, 1
2	milk	C categorical	feature	0, 1
3	backbone	C categorical	feature	0, 1
4	fins	C categorical	feature	0, 1
5	legs	N numeric	feature	

Click the **prediction** widget on the canvas to view the predicted class for the various animals in the given dataset.



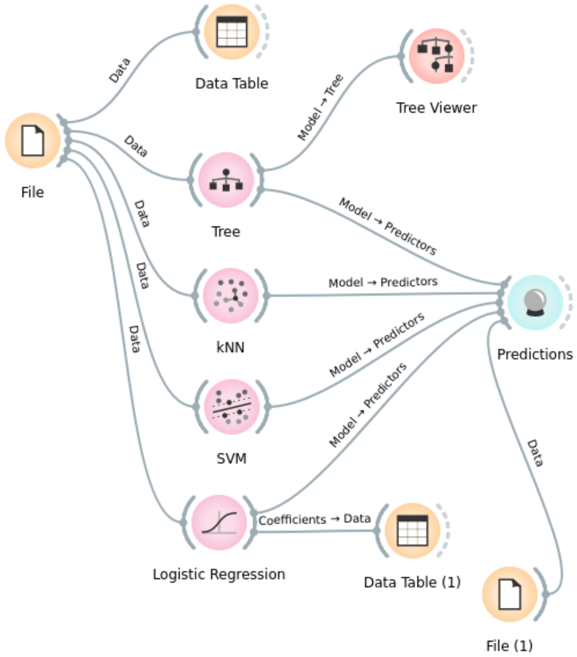
	Tree	features	milk	backbones	fins	legs
1	bird	1	0	1	0	2
2	bird	0	0	1	1	0
3	mammal	0	1	1	0	4

Note: Your predictions may be different than the one shown in the diagram, which depend on the setting of the hyper-parameters in the Tree widgets, which may affect the rules in the decision tree model.

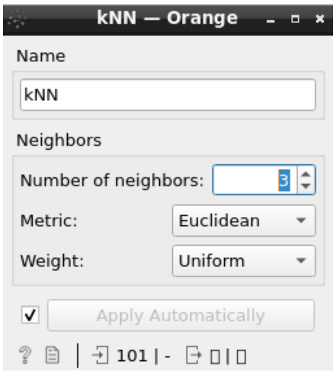
Use the **Tree Viewer** to understand how the predicted type of animal is computed.

Modify the workflow as follows

- Add a **KNN**, **SVM**, and **Logistic Regression** classifiers widgets. Connect the output of the dataset (the **File** widget) to the input of the classifiers. Connect the output of the classifier widgets to the prediction.
- Connect the output of Logistic Regression to a **Data Table**.



Double click the kNN widgets to review the hyper-parameters. Set the Number of **neighbors** to **3**.


kNN — Orange

Name

kNN

Neighbors

Number of neighbors: 3

Metric: Euclidean

Weight: Uniform

☑ Apply Automatically

101 | -

Click the **prediction** widget to examine the predicted animal types.

Save your workflow as **zoo.ows**.

5. Evaluation of classification models

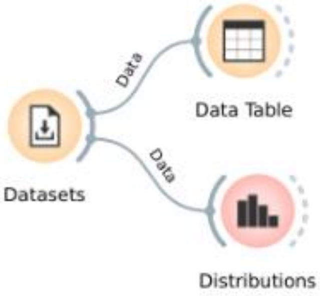
On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren’t enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew.

In this section, we will analyse a simplified version of the Titanic Dataset provided by Orange. We will understand what sort of people were more likely to survive the Titanic sinking, develop a classification model to predict whether passengers will survive by building classification models and evaluate the model’s accuracy.

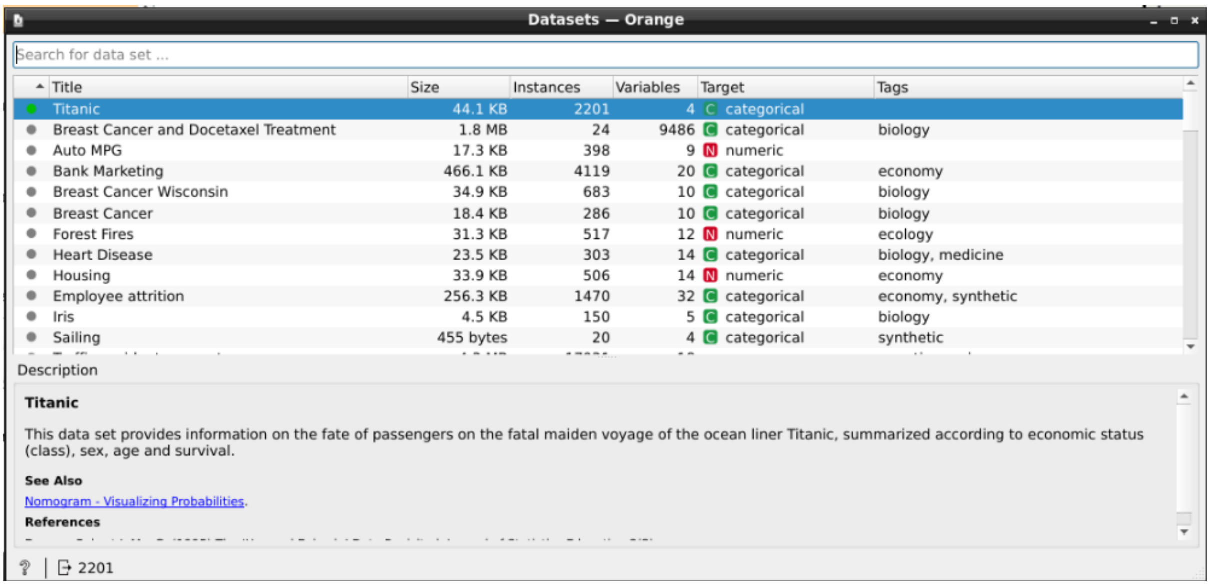
Start a new workflow (**File -> New**).

Data Exploration

Start a new workflow (**File -> New**). Create the following workflow.



Double click the Dataset widget. Use the **titanic dataset.**



Title	Size	Instances	Variables	Target	Tags
● Titanic	44.1 KB	2201	4	C categorical	
● Breast Cancer and Docetaxel Treatment	1.8 MB	24	9486	C categorical	biology
● Auto MPG	17.3 KB	398	9	N numeric	
● Bank Marketing	466.1 KB	4119	20	C categorical	economy
● Breast Cancer Wisconsin	34.9 KB	683	10	C categorical	biology
● Breast Cancer	18.4 KB	286	10	C categorical	biology
● Forest Fires	31.3 KB	517	12	N numeric	ecology
● Heart Disease	23.5 KB	303	14	C categorical	biology, medicine
● Housing	33.9 KB	506	14	N numeric	economy
● Employee attrition	256.3 KB	1470	32	C categorical	economy, synthetic
● Iris	4.5 KB	150	5	C categorical	biology
● Sailing	455 bytes	20	4	C categorical	synthetic

Description

Titanic

This data set provides information on the fate of passengers on the fatal maiden voyage of the ocean liner Titanic, summarized according to economic status (class), sex, age and survival.

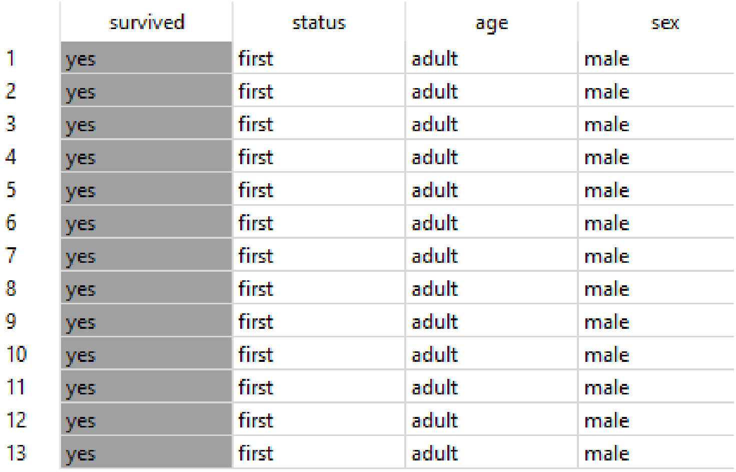
See Also

Nomogram - Visualizing Probabilities.

References

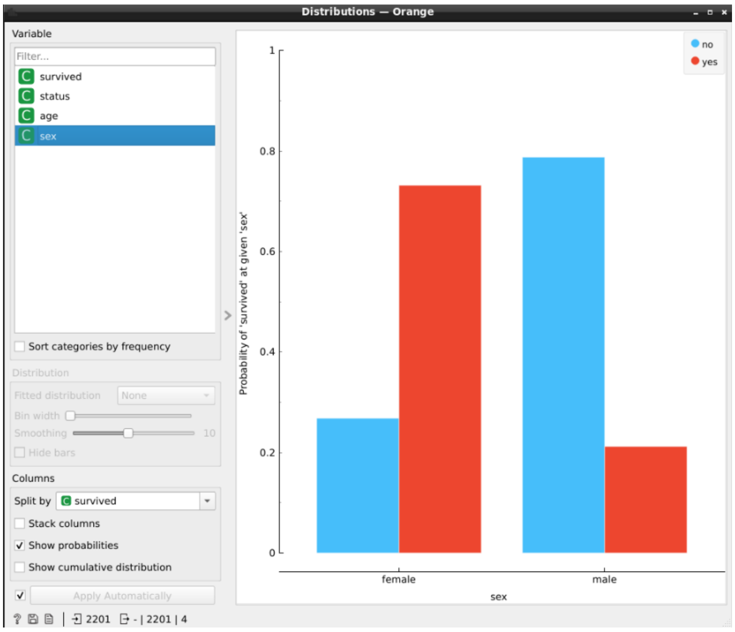
Close the widget.

Double click **Data Table** to review the dataset.



	survived	status	age	sex
1	yes	first	adult	male
2	yes	first	adult	male
3	yes	first	adult	male
4	yes	first	adult	male
5	yes	first	adult	male
6	yes	first	adult	male
7	yes	first	adult	male
8	yes	first	adult	male
9	yes	first	adult	male
10	yes	first	adult	male
11	yes	first	adult	male
12	yes	first	adult	male
13	yes	first	adult	male

Double click the **Distribution** widget for exploration of the data. For the **Split by** option, Select **survived**



Distributions — Orange

Variable

- survived
- status
- age
- sex

□ Sort categories by frequency

Distribution

Fitted distribution: None

Bin width

Smoothing 10

□ Hide bars

Columns

Split by: survived

□ Stack columns

☑ Show probabilities

□ Show cumulative distribution

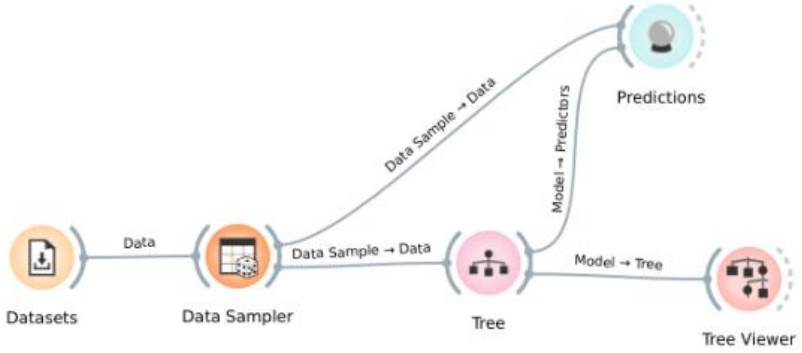
☑ Apply Automatically

2201 | - | 2201 | 4

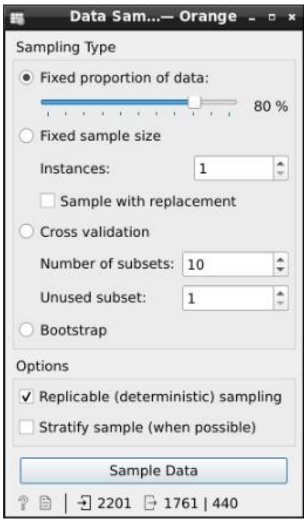
Select the different variable and explore which attribute will best split the data to predict the survival of a passenger.

Training and evaluating the classification model

We will split the data into training set (80%) and test set (20%) and train a decision tree classifier on the training set.



For click the **Data Sampler**. Select Fixed proportion of data and select 80%. To ensure that that the result can be replicated, check the option **Replicable (deterministic) sampling**. If not, the data will be sampled randomly and you may get different results every time.



Data Sam...— Orange

Sampling Type

● Fixed proportion of data: 80 %

○ Fixed sample size

Instances: 1

□ Sample with replacement

○ Cross validation

Number of subsets: 10

Unused subset: 1

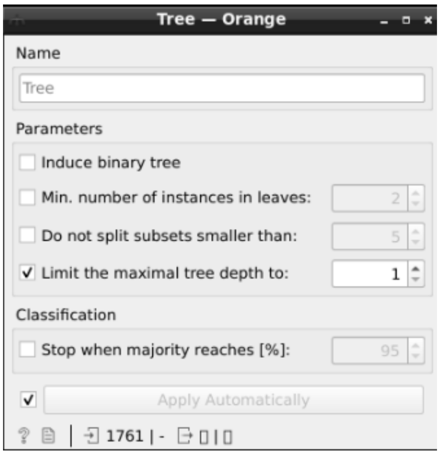
○ Bootstrap

Options

☑ Replicable (deterministic) sampling

□ Stratify sample (when possible)

Double click the **Tree** widget. Check the option **Limit the maximal tree depth** to **1**. Uncheck all other options.



Tree — Orange

Name

Tree

Parameters

□ Induce binary tree

□ Min. number of instances in leaves: 2

□ Do not split subsets smaller than: 5

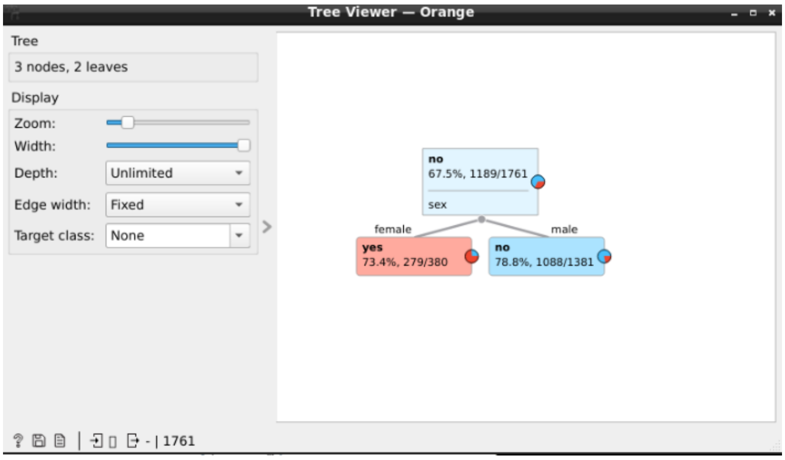
☑ Limit the maximal tree depth to: 1

Classification

□ Stop when majority reaches [%]: 95

☑ Apply Automatically

Click the **Tree Viewer** to view the decision tree.



Tree

3 nodes, 2 leaves

Display

Zoom:

Width:

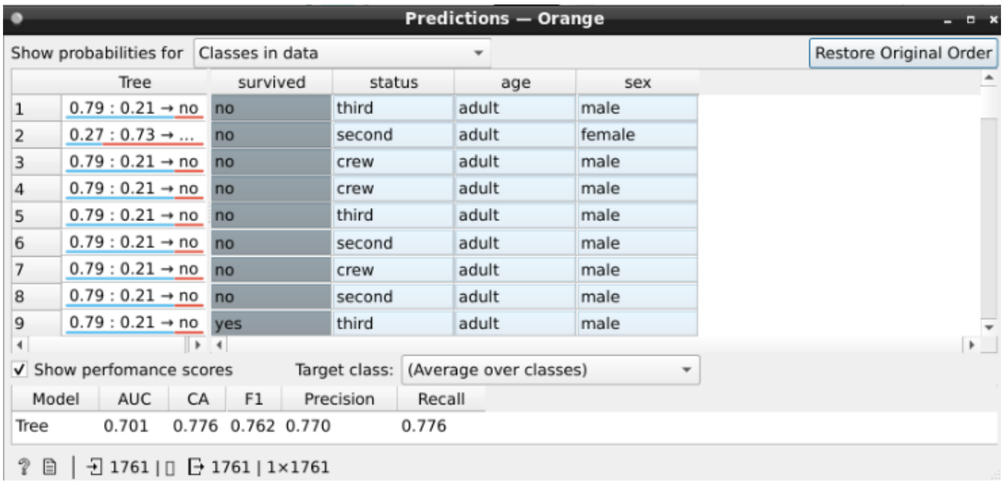
Depth: Unlimited

Edge width: Fixed

Target class: None

1761

Double click the **Predictions** widget. Check **Show performance scores** and **select (Average over classes)**.



Show probabilities for Classes in data

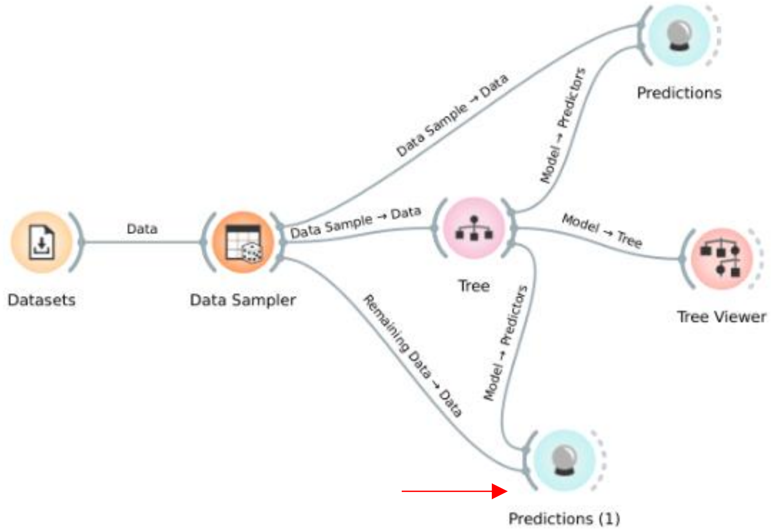
	Tree	survived	status	age	sex
1	0.79 : 0.21 → no	no	third	adult	male
2	0.27 : 0.73 → ...	no	second	adult	female
3	0.79 : 0.21 → no	no	crew	adult	male
4	0.79 : 0.21 → no	no	crew	adult	male
5	0.79 : 0.21 → no	no	third	adult	male
6	0.79 : 0.21 → no	no	second	adult	male
7	0.79 : 0.21 → no	no	crew	adult	male
8	0.79 : 0.21 → no	no	second	adult	male
9	0.79 : 0.21 → no	yes	third	adult	male

☑ Show perfomance scores Target class: (Average over classes)

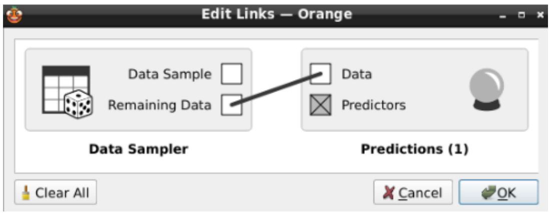
Model	AUC	CA	F1	Precision	Recall
Tree	0.701	0.776	0.762	0.770	0.776

1761 | 1761 | 1×1761

What is the **Classification Accuracy (CA)** of the model on the training data?



Next, add another **Predictions** widget (**Predictions (1)** in the diagram below). To send the remaining 20% of the data to the second **Predictions** widget, double click the connection line and connect **Remaining Data** output of the Data Sampler widget to the Data input of the **Predictions (1)** widget.



Click the **Predictions(1)** widget to view the **classification accuracy** metrics on the **test set**.

Exercise

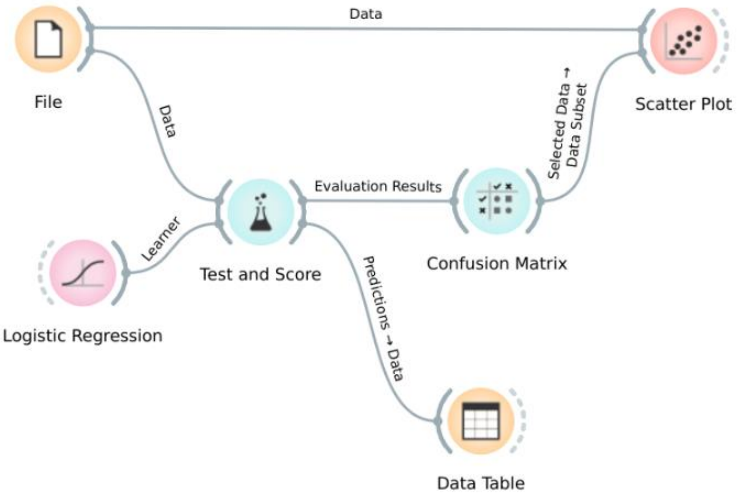
Change the **Limit the maximal tree depth** to 2,3, … and observe the change in decision Tree. Observe the change in the classification accuracy on the training and test set.

6. Cross validation

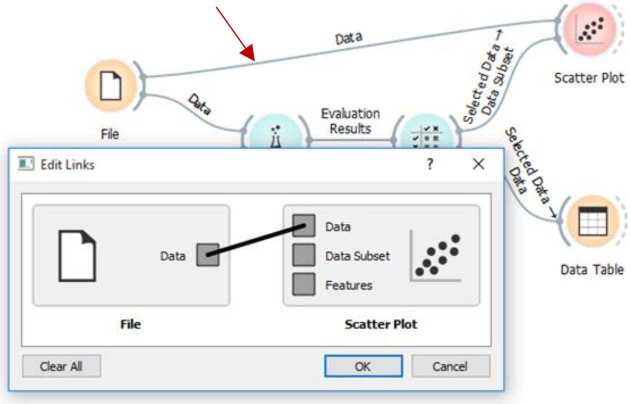
In this section, we will use logistic regression to classify the type of iris species. The performance of the model will be evaluated using cross validation and compared with other classification models.

Start a new workflow (**File -> New**).

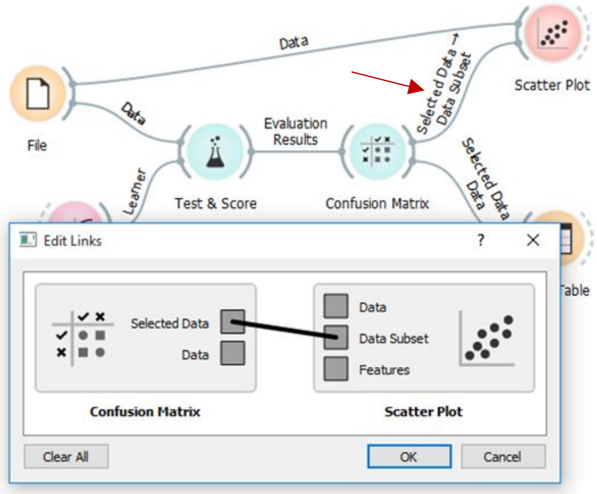
Create a **data source** using the **Iris data set** and feed the data to a **scatter plot**. Add the **logistic regression widget** in the canvas and connect the **output channel** to the input channel of the **test & score widget (**available under the **evaluate category** in the menu**)**. Connect the output of **test & score widget** to the input channel of the **confusion matrix widget** (available under the **evaluate category** in the menu). The workflow configuration is illustrated below.



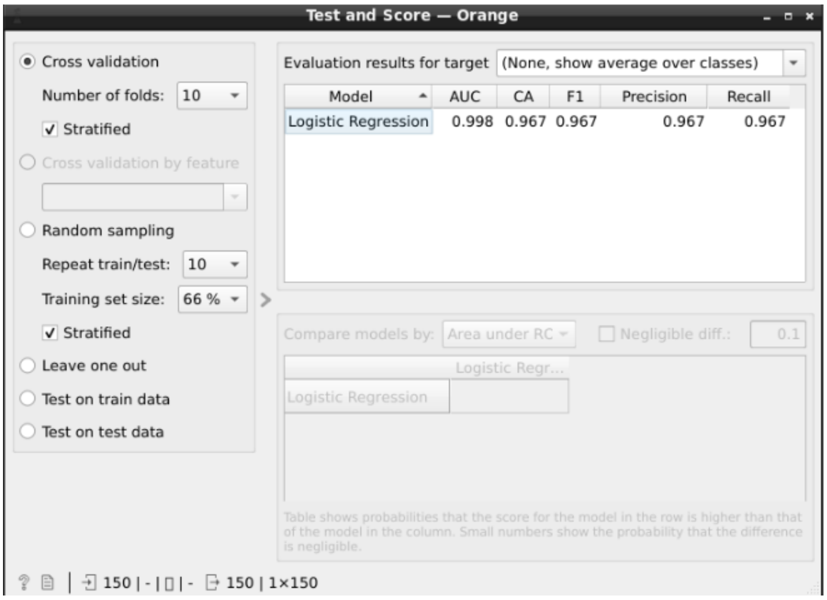
Double click the line between **File** and **Scatter Plot** and configure as follows.



Double click the line between **Confusion Matrix** and **Scatter Plot** and configure as follows.



Double click **Test & Score** to open the widget.



● Cross validation
Number of folds: 10
☑ 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 (None, show average over classes)

Model	AUC	CA	F1	Precision	Recall
Logistic Regression	0.998	0.967	0.967	0.967	0.967

Compare models by: Area under RC ☐ Negligible diff.: 0.1

	Logistic Regr...
Logistic Regression	

Table shows probabilities that the score for the model in the row is higher than that of the model in the column. Small numbers show the probability that the difference is negligible.

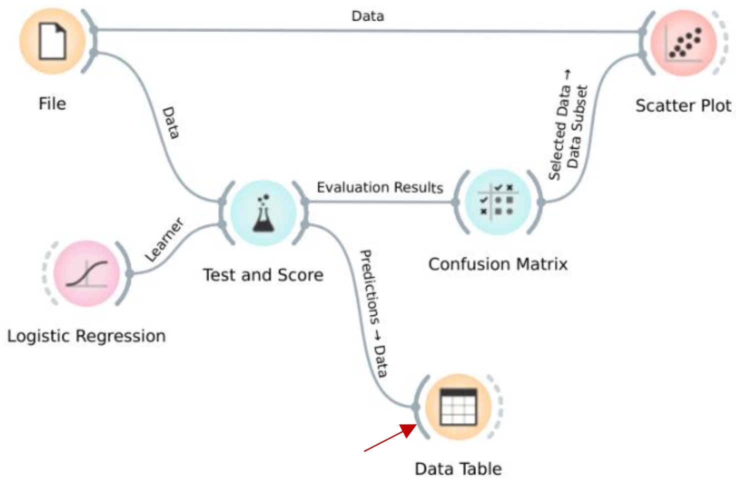
150 | - | | - 150 | 1×150

Under Sampling, select **Cross validation** and choose **10**. Check the **Stratified** option.

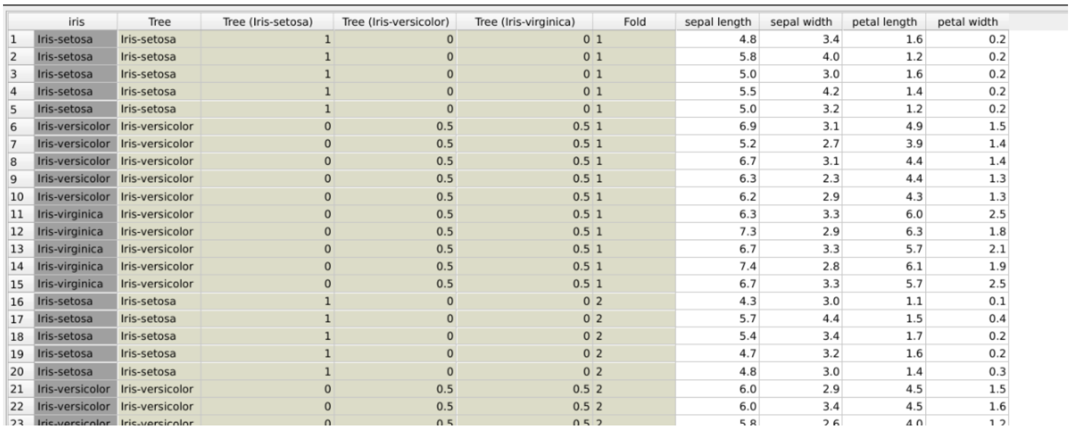
- A 10-fold cross validation is used to evaluation the predictive model. The dataset is first shuffled randomly and split into 10 groups. For each group (1/10 of the data), the data is used as the **validation set**. The remaining data is used as the **training set** to fit the model and evaluate it on the test set.
- In **stratified sampling**, samples for the training/validation/test set are selected in the same proportion as in the dataset.

The various performance metrics of logistics regression such as **classification accuracy (CA)**, **Precision** and **Recall** is shown in **Evaluation Results**.

Double click the **Data table** (with input from **Test and Score**).

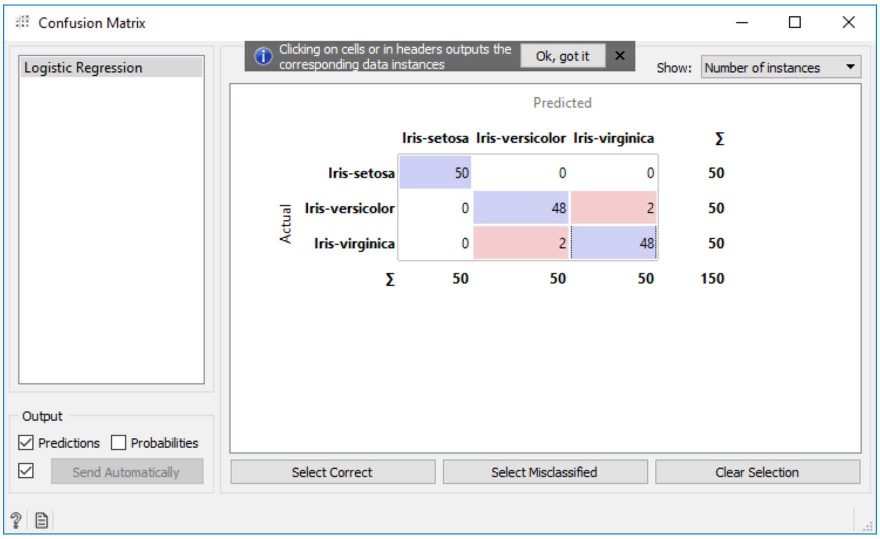


You may view the **Fold** of each sample and their predicted probability for each class.



	iris	Tree	Tree (Iris-setosa)	Tree (Iris-versicolor)	Tree (Iris-virginica)	Fold	sepal length	sepal width	petal length	petal width
1	Iris-setosa	Iris-setosa	1	0	0	1	4.8	3.4	1.6	0.2
2	Iris-setosa	Iris-setosa	1	0	0	1	5.8	4.0	1.2	0.2
3	Iris-setosa	Iris-setosa	1	0	0	1	5.0	3.0	1.6	0.2
4	Iris-setosa	Iris-setosa	1	0	0	1	5.5	4.2	1.4	0.2
5	Iris-setosa	Iris-setosa	1	0	0	1	5.0	3.2	1.2	0.2
6	Iris-versicolor	Iris-versicolor	0	0.5	0.5	1	6.9	3.1	4.9	1.5
7	Iris-versicolor	Iris-versicolor	0	0.5	0.5	1	5.2	2.7	3.9	1.4
8	Iris-versicolor	Iris-versicolor	0	0.5	0.5	1	6.7	3.1	4.4	1.4
9	Iris-versicolor	Iris-versicolor	0	0.5	0.5	1	6.3	2.3	4.4	1.3
10	Iris-versicolor	Iris-versicolor	0	0.5	0.5	1	6.2	2.9	4.3	1.3
11	Iris-virginica	Iris-versicolor	0	0.5	0.5	1	6.3	3.3	6.0	2.5
12	Iris-virginica	Iris-versicolor	0	0.5	0.5	1	7.3	2.9	6.3	1.8
13	Iris-virginica	Iris-versicolor	0	0.5	0.5	1	6.7	3.3	5.7	2.1
14	Iris-virginica	Iris-versicolor	0	0.5	0.5	1	7.4	2.8	6.1	1.9
15	Iris-virginica	Iris-versicolor	0	0.5	0.5	1	6.7	3.3	5.7	2.5
16	Iris-setosa	Iris-setosa	1	0	0	2	4.3	3.0	1.1	0.1
17	Iris-setosa	Iris-setosa	1	0	0	2	5.7	4.4	1.5	0.4
18	Iris-setosa	Iris-setosa	1	0	0	2	5.4	3.4	1.7	0.2
19	Iris-setosa	Iris-setosa	1	0	0	2	4.7	3.2	1.6	0.2
20	Iris-setosa	Iris-setosa	1	0	0	2	4.8	3.0	1.4	0.3
21	Iris-versicolor	Iris-versicolor	0	0.5	0.5	2	6.0	2.9	4.5	1.5
22	Iris-versicolor	Iris-versicolor	0	0.5	0.5	2	6.0	3.4	4.5	1.6
23	Iris-versicolor	Iris-versicolor	0	0.5	0.5	2	5.8	2.6	4.0	1.2

Open the **Confusion Matrix** by double-clicking the widget.



Logistic Regression

Clicking on cells or in headers outputs the corresponding data instances

Show: Number of instances

Predicted

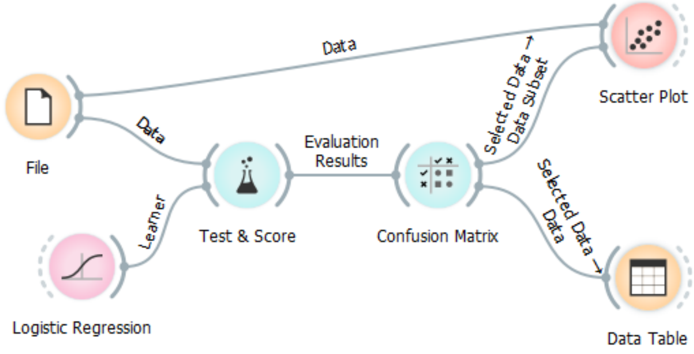
		Iris-setosa	Iris-versicolor	Iris-virginica	Σ
Actual	Iris-setosa	50	0	0	50
	Iris-versicolor	0	48	2	50
	Iris-virginica	0	2	48	50
	Σ	50	50	50	150

Output

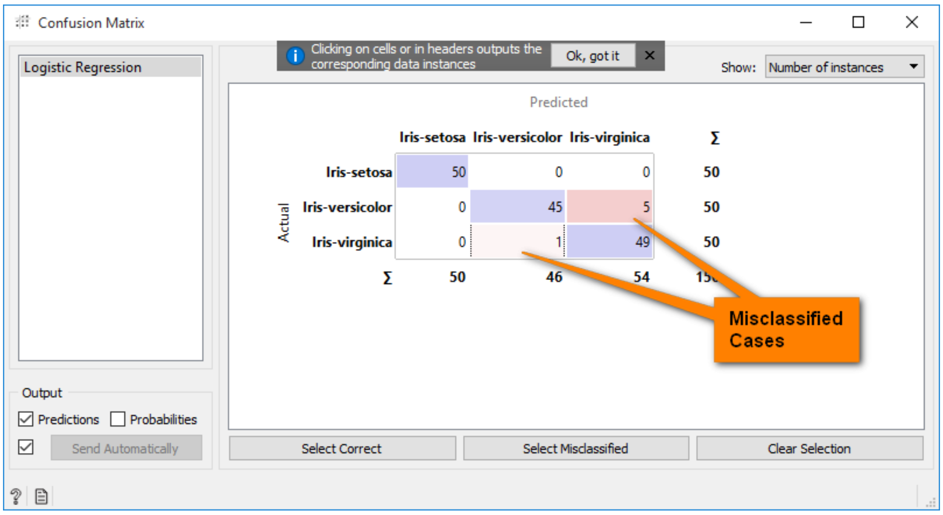
☑ Predictions □ Probabilities

☑ Send Automatically

Refine the workflow as follows.

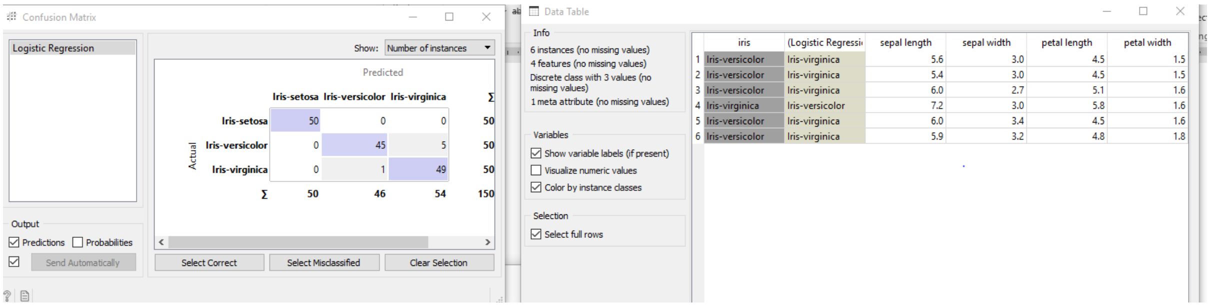


To further explore the data, select the **misclassified cases** on the confusion matrix shown by holding "**Ctrl**" and click on the misclassified cases which are **highlighted in red** in the confusion matrix. We can view the data subset (the misclassified cases) in the data table and scatter plot.



Predicted

Actual	Iris-setosa	Iris-versicolor	Iris-virginica	Σ
Iris-setosa	50	0	0	50
Iris-versicolor	0	45	5	50
Iris-virginica	0	1	49	50
Σ	50	46	54	150



Predicted

Actual	Iris-setosa	Iris-versicolor	Iris-virginica	Σ
Iris-setosa	50	0	0	50
Iris-versicolor	0	45	5	50
Iris-virginica	0	1	49	50
Σ	50	46	54	150

Info

6 instances (no missing values)
4 features (no missing values)
Discrete class with 3 values (no missing values)
1 meta attribute (no missing values)

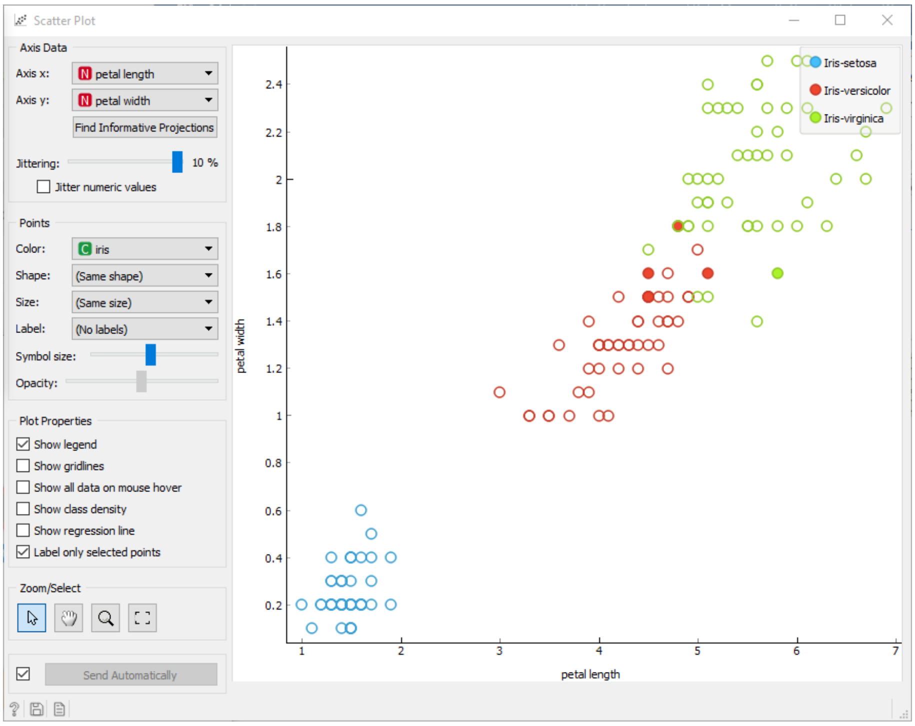
Variables

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

Selection

☑ Select full rows

	iris	(Logistic Regressi	sepal length	sepal width	petal length	petal width
1	Iris-versicolor	Iris-virginica	5.6	3.0	4.5	1.5
2	Iris-versicolor	Iris-virginica	5.4	3.0	4.5	1.5
3	Iris-versicolor	Iris-virginica	6.0	2.7	5.1	1.6
4	Iris-virginica	Iris-versicolor	7.2	3.0	5.8	1.6
5	Iris-versicolor	Iris-virginica	6.0	3.4	4.5	1.6
6	Iris-versicolor	Iris-virginica	5.9	3.2	4.8	1.8



Scatter Plot

Axis Data

Axis x: petal length

Axis y: petal width

Find Informative Projections

Jittering: 10 %

□ Jitter numeric values

Points

Color: iris

Shape: (Same shape)

Size: (Same size)

Label: (No labels)

Symbol size:

Opacity:

Plot Properties

☑ Show legend

□ Show gridlines

□ Show all data on mouse hover

□ Show class density

□ Show regression line

☑ Label only selected points

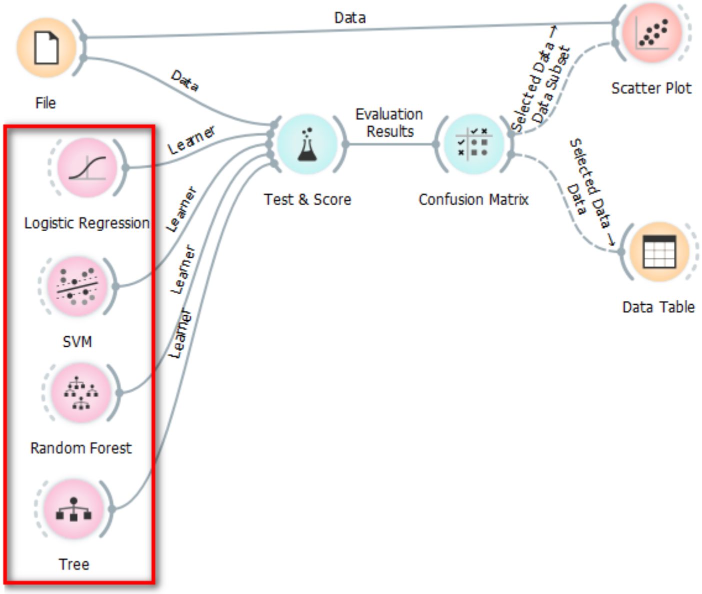
Zoom/Select

☑ Send Automatically

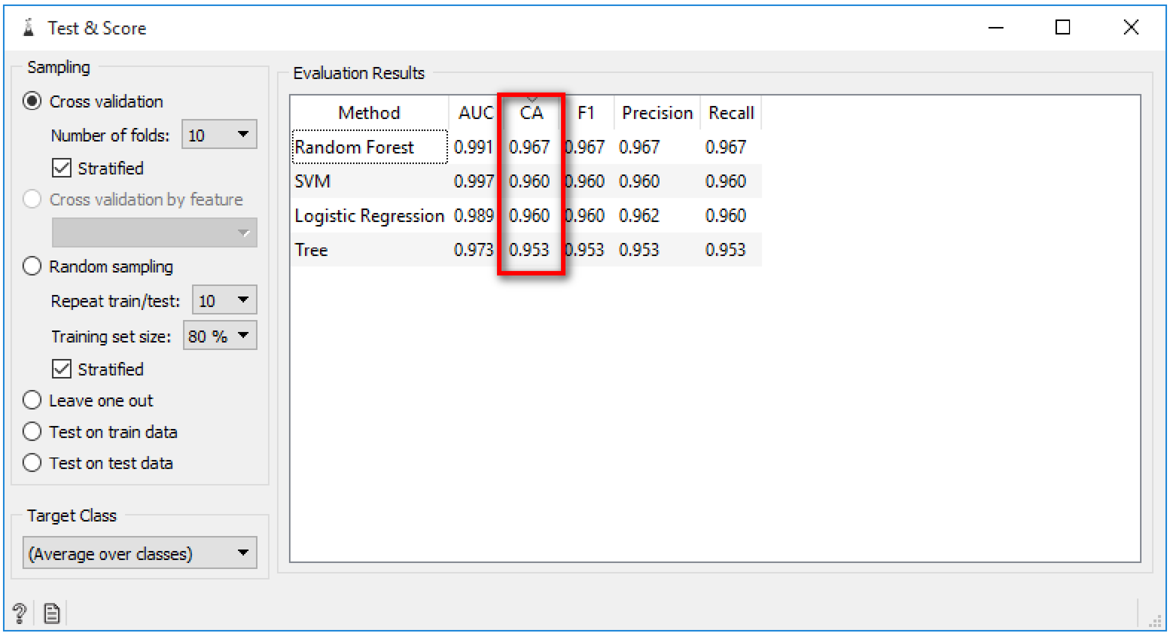
In the scatter plot above, the misclassified flowers are shown as filled circles.

Comparing different models

You may compare the performance of logistic regression with other classification algorithms (SVM, Decision Tree, Random Forest) by refining the workflow as follows.



Open the **Test & Score** to and compare the **Classification Accuracy** (CA) and other metrics of the various algorithms.



Test & Score

Sampling

● Cross validation

Number of folds: 10

☑ Stratified

○ Cross validation by feature

○ Random sampling

Repeat train/test: 10

Training set size: 80 %

☑ Stratified

○ Leave one out

○ Test on train data

○ Test on test data

Target Class

(Average over classes)

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
Random Forest	0.991	0.967	0.967	0.967	0.967
SVM	0.997	0.960	0.960	0.960	0.960
Logistic Regression	0.989	0.960	0.960	0.962	0.960
Tree	0.973	0.953	0.953	0.953	0.953

Select the option **Test on train data**. Compare the classification accuracy of the model.

Saving the model

Save the model as "**iris cross validation.ows**".

7. References

1. Orange Documentation
   https://orange.biolab.si/docs/
2. Getting Started with Orange
   https://www.youtube.com/playlist?list=PLmNPvQr9Tf-ZSDLwOzxpvY-HrE0yv-8Fy
3. Machine Learning Jargon
   https://orangedatamining.com/blog/2022/2022-02-01-machine-learning-jargon
4. The Kaggle titanic dataset
   https://www.kaggle.com/c/titanic