

SVM

Support Vector Machines map inputs to higher-dimensional feature spaces.

Inputs

- Data: input dataset
- Preprocessor: preprocessing method(s)

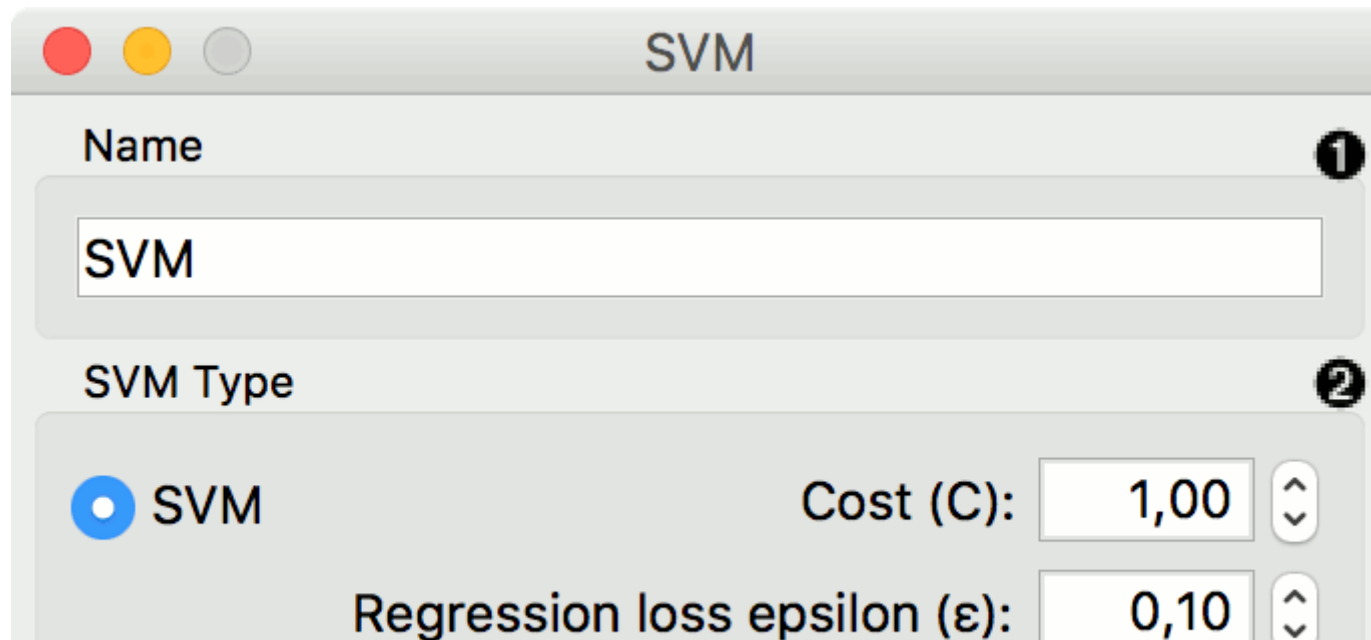
Outputs

- Learner: linear regression learning algorithm
- Model: trained model
- Support Vectors: instances used as support vectors

Support vector machine (SVM) is a machine learning technique that separates the attribute space with a hyperplane, thus maximizing the margin between the instances of different classes or class values. The technique often yields supreme predictive performance results. Orange embeds a popular implementation of SVM from the **LIBSVM** package. This widget is its graphical user interface.

For regression tasks, **SVM** performs linear regression in a high dimension feature space using an ϵ -insensitive loss. Its estimation accuracy depends on a good setting of C , ϵ and kernel parameters. The widget outputs class predictions based on a **SVM Regression**.

The widget works for both classification and regression tasks.



☐ v-SVM

Regression cost (C): 1,00

Complexity bound (ν): 0,50

Kernel

3

☐ LinearKernel: $\exp(-g|x-y|^2)$ ☐ Polynomial

g: auto

☒ RBF☐ Sigmoid

Optimization Parameters

4

Numerical tolerance:

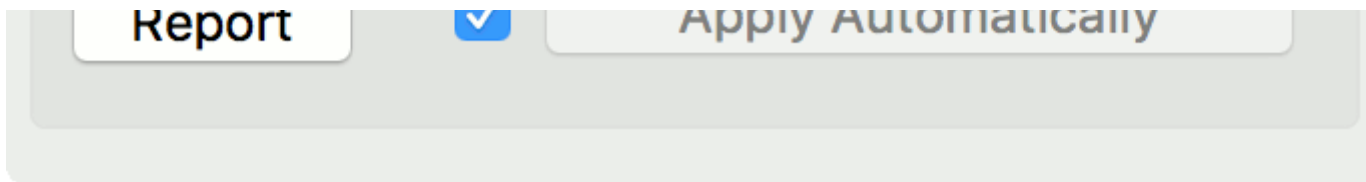
0,0010

☒ Iteration limit:

100

5

6



1. The learner can be given a name under which it will appear in other widgets. The default name is “SVM”.
2. SVM type with test error settings. *SVM* and *v-SVM* are based on different minimization of the error function. On the right side, you can set test error bounds:
 - **SVM**:
 - **Cost**: penalty term for loss and applies for classification and regression tasks.
 - ϵ : a parameter to the epsilon-SVR model, applies to regression tasks. Defines the distance from true values within which no penalty is associated with predicted values.
 - **v-SVM**:
 - **Cost**: penalty term for loss and applies only to regression tasks
 - v : a parameter to the v-SVR model, applies to classification and regression tasks. An upper bound on the fraction of training errors and a lower bound of the fraction of support vectors.
3. Kernel is a function that transforms attribute space to a new feature space to fit the maximum-margin hyperplane, thus allowing the algorithm to create the model with **Linear**, **Polynomial**, **RBF** and **Sigmoid** kernels. Functions that specify the kernel are presented upon selecting them, and the constants involved are:
 - **g** for the gamma constant in kernel function (the recommended value is $1/k$, where k is the number of the attributes, but since there may be no training set given to the widget the default is 0 and the user has to set this option manually),
 - **c** for the constant c_0 in the kernel function (default 0), and
 - **d** for the degree of the kernel (default 3).
4. Set permitted deviation from the expected value in *Numerical Tolerance*. Tick the box next to *Iteration Limit* to set the maximum number of iterations permitted.
5. Produce a report.
6. Click *Apply* to commit changes. If you tick the box on the left side of the *Apply* button, changes will be communicated automatically.

Examples

In the first (regression) example, we have used *housing* dataset and split the data into two data subsets (*Data Sample* and *Remaining Data*) with **Data Sampler**. The sample was sent to SVM which produced a *Model*, which was then used in **Predictions** to predict the values in *Remaining Data*. A similar schema can be used if the data is already in two separate files; in this case, two **File** widgets would be used instead of the **File** - **Data Sampler** combination.

Workflow Diagram:

```

graph LR
    File[File] -- Data --> DS[Data Sampler]
    DS -- "Data Sample → Data" --> SVM[SVM]
    DS -- "Remaining Data → Data" --> Data[Data]
    SVM -- "Model → Predictors" --> Predictions[Predictions]
  
```

SVM Widget Settings:

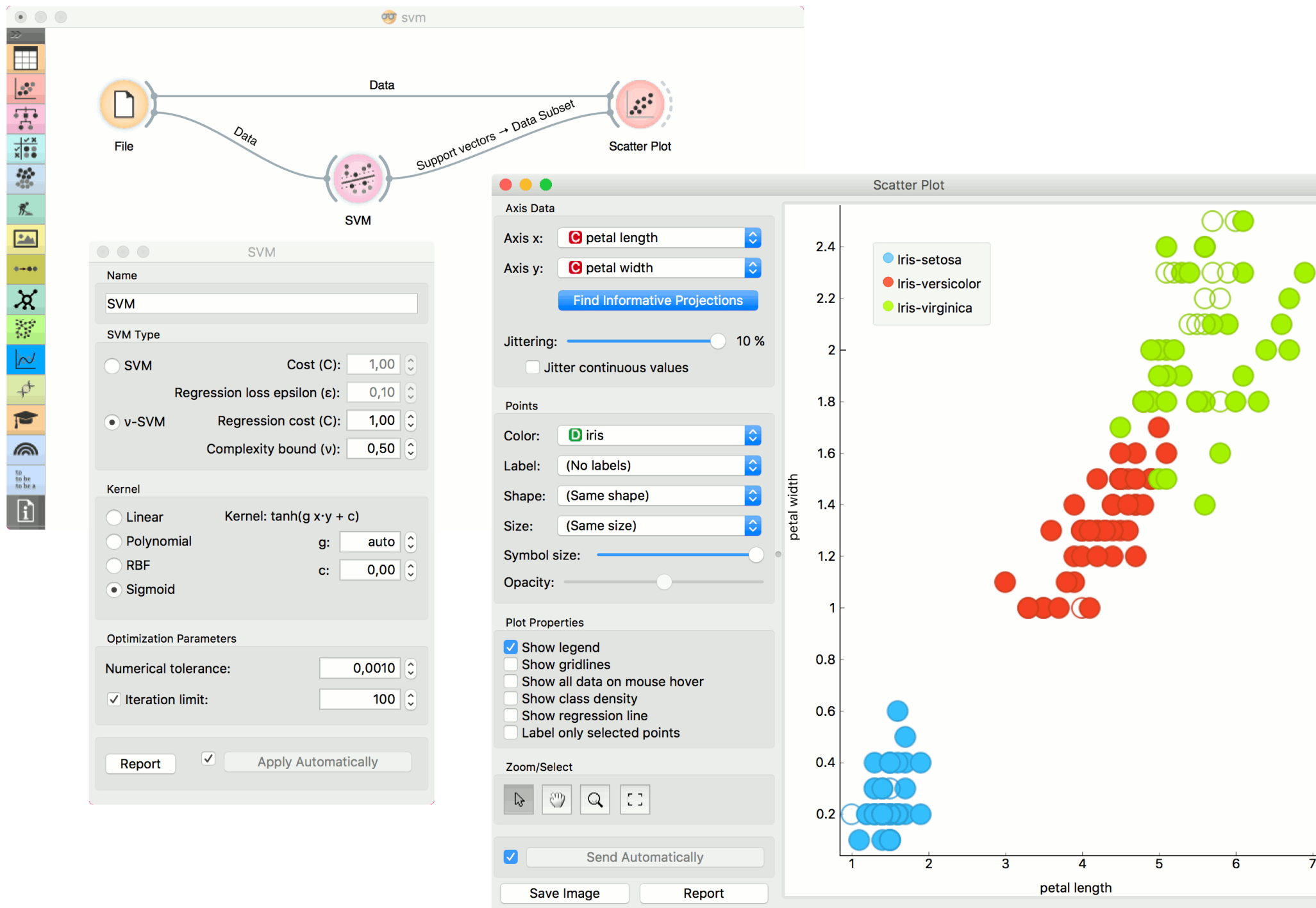
- Name: SVM
- SVM Type:
 - ☒ SVM
 - Cost (C): 1,00
 - Regression loss epsilon (ε): 0,10
 - ☐ v-SVM
 - Regression cost (C): 1,00
 - Complexity bound (v): 0,50
- Kernel:
 - ☐ Linear
 - ☐ Polynomial
 - ☐ RBF
 - ☒ Sigmoid
 Kernel: $\tanh(g \cdot x \cdot y + c)$
 - g: auto
 - c: 0,00
- Optimization Parameters:
 - Numerical tolerance: 0,0010
 - ☒ Iteration limit: 100
- Buttons: Report, ☒ Apply Automatically

Predictions Widget Output:

	SVM	MEDV	CRIM	ZN	INDUS	CHAS
1	19.380	21.500	0.111	0.000	13.890	1.000
2	26.655	23.600	0.057	0.000	3.410	0.000
3	12.934	7.500	10.834	0.000	18.100	0.000
4	20.883	19.500	0.034	0.000	5.190	0.000
5	11.949	13.800	2.379	0.000	19.580	0.000
6	21.513	29.600	0.060	0.000	2.460	0.000
7	13.875	19.300	0.376	0.000	10.590	1.000
8	17.586	16.800	0.224	0.000	9.690	0.000
9	12.252	13.500	1.613	0.000	8.140	0.000
10	22.721	22.000	0.058	12.500	6.070	0.000
11	22.204	20.100	0.020	80.000	1.760	0.000
12	23.655	22.000	0.110	0.000	11.930	0.000
13	8.167	8.800	20.085	0.000	18.100	0.000

The second example shows how to use **SVM** in combination with **Scatter Plot**. The following workflow trains a SVM model on *iris* data and outputs support vectors, which are those data instances that were used as support vectors in the learning phase. We can observe which are these data instances in a scatter plot visualization. Note that for the workflow to work correctly, you must set the links between widgets as

demonstrated in the screenshot below.



References

Introduction to SVM on StatSoft.