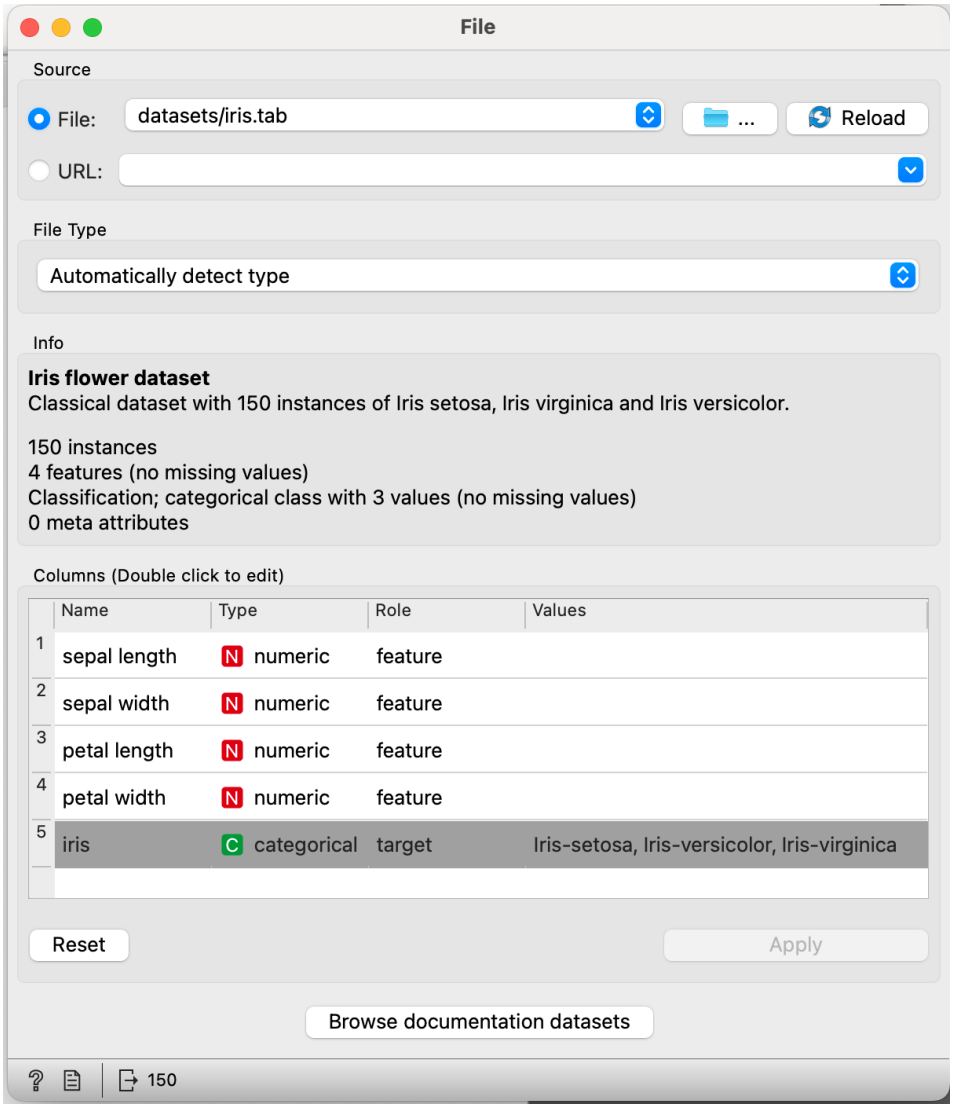
CSCU9M5 Practical 4

Training and Testing a Neural Network model

Now we are going to train and test a Neural Network model using Orange.

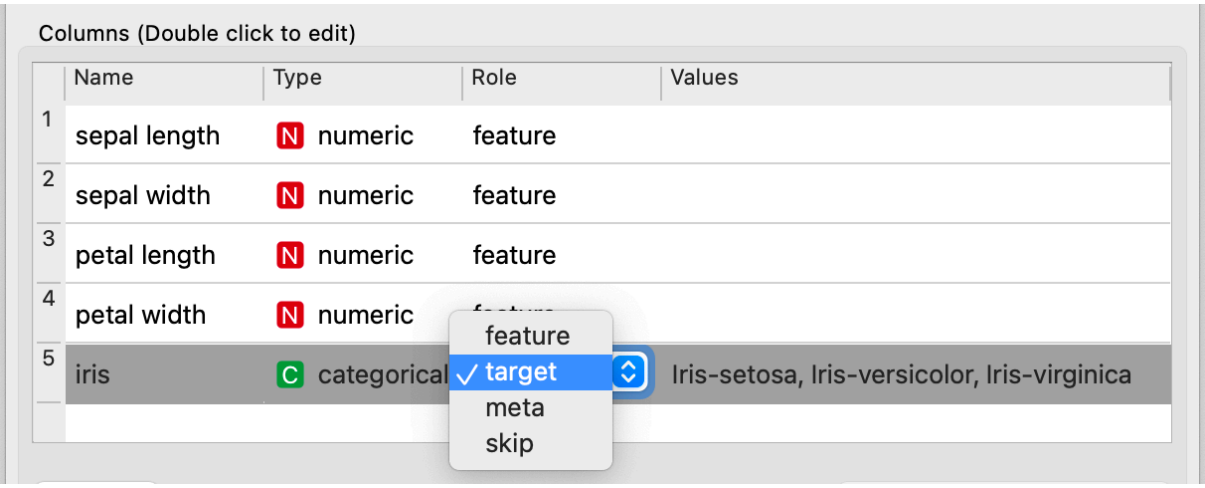
(You can also compare the performance with a logistic regression model)

Create a new workflow and add a file widget as you did previously, and double click it. You should see that the default file is **iris.tab** (if it is not, load in the iris.csv file for this session).



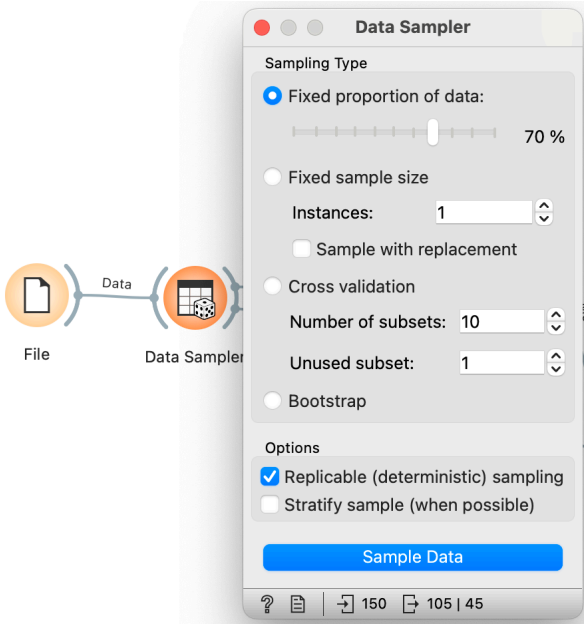
This is the same dataset that was used in previous practical sessions using Orange and is a very well-known dataset in machine learning. The dataset shows sepal length, sepal width, petal length, and petal width for Iris flowers.

You will notice that the ‘iris’ variable role is set to ‘target’, showing the three different species of the Iris flower. If it not is not set automatically in your instance, double click it and change it in the drop down, as seen in the image.

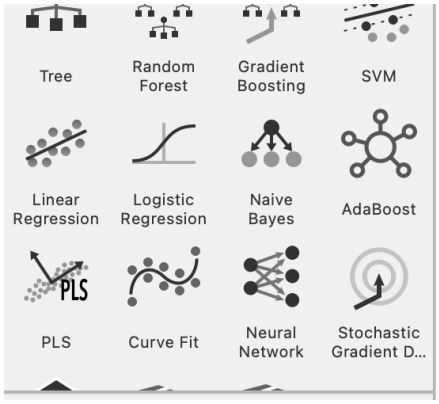


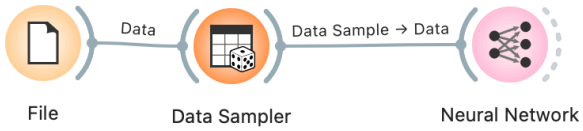
You will notice that the value column displays *Iris-setosa, Iris-versicolor, and Iris-virginica* - the three species of Iris that are labelled in this dataset.

As we have done previously, we will split our dataset into Training and Testing data to evaluate our model using a **Data Sampler**. Connect a **Data Sampler widget** to the File widget, and double click it to ensure that the data is being split into two fixed proportions: 70%, which we will use to train the model, and 30%, which we will use to test it.

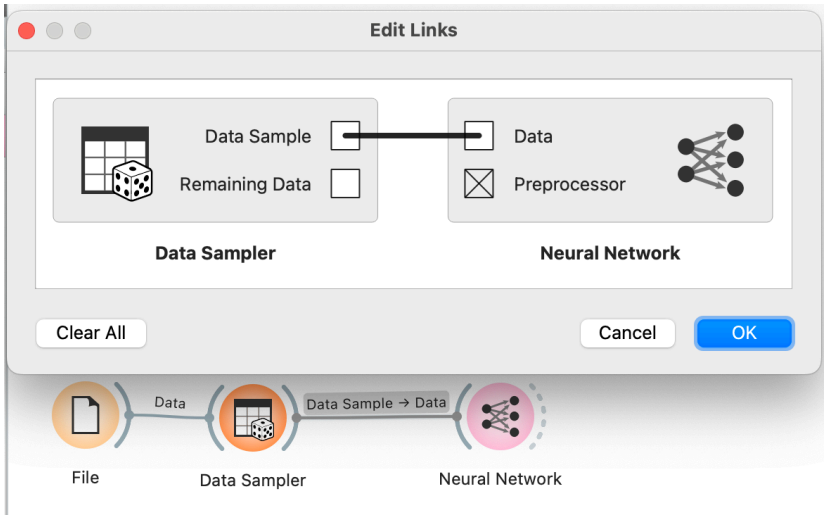


Now that we have our training data, it’s time to create our Neural Network model. Add a Neural Network widget from under the Model section on the left of the screen, and connect the data sampler to it.



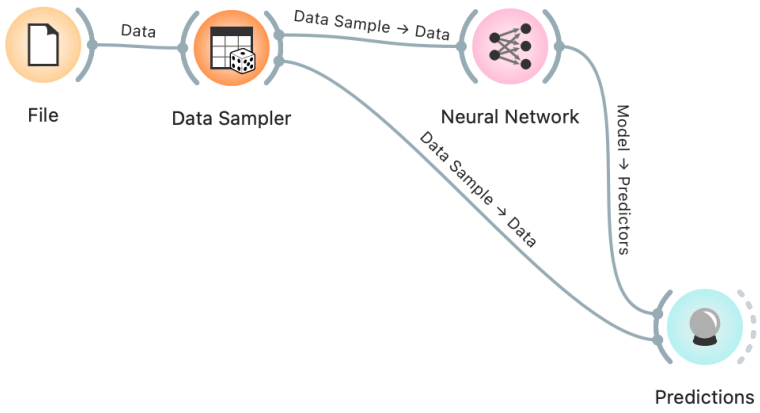


Double click the line joining the data sampler to the Neural Network widget, and ensure that the Data Sample from the Data Sampler is connected to the Data in the Neural Network. This means that the sample consisting of 70% of the overall dataset is now being provided to train the model.

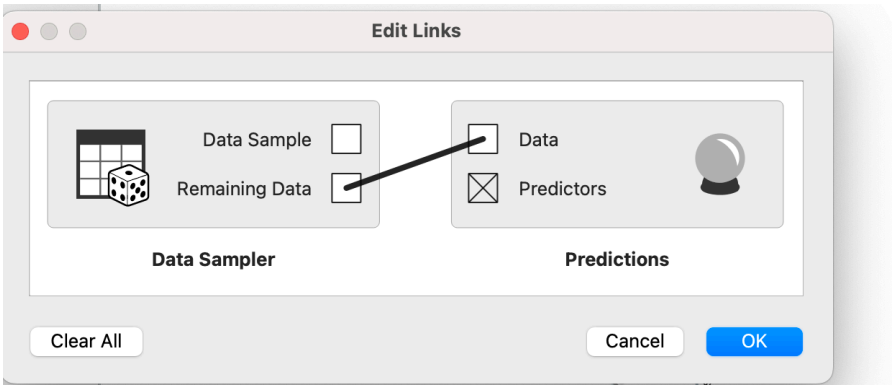


Note: For now, we keep the default parameters in the Neural Network widget. We can configure the number of hidden units and the learning method, but more on this in the later part of this tutorial.

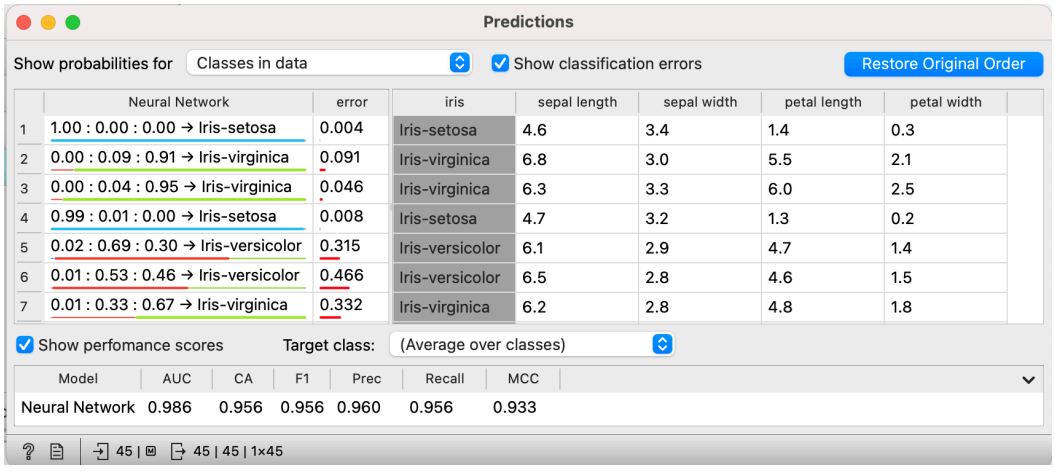
Now that we have a Neural Network model, we need to evaluate it. Do this by adding a predictions widget to the workflow. Connect the model, and connect the Data Sampler to this widget.



Ensure that the data going from the sampler to the prediction widget is the remaining 30% of the data as in the image below.



Double click the Predictions widget to see the predictions made by the Neural network model.



	Neural Network	error	iris	sepal length	sepal width	petal length	petal width
1	1.00 : 0.00 : 0.00 → Iris-setosa	0.004	Iris-setosa	4.6	3.4	1.4	0.3
2	0.00 : 0.09 : 0.91 → Iris-virginica	0.091	Iris-virginica	6.8	3.0	5.5	2.1
3	0.00 : 0.04 : 0.95 → Iris-virginica	0.046	Iris-virginica	6.3	3.3	6.0	2.5
4	0.99 : 0.01 : 0.00 → Iris-setosa	0.008	Iris-setosa	4.7	3.2	1.3	0.2
5	0.02 : 0.69 : 0.30 → Iris-versicolor	0.315	Iris-versicolor	6.1	2.9	4.7	1.4
6	0.01 : 0.53 : 0.46 → Iris-versicolor	0.466	Iris-versicolor	6.5	2.8	4.6	1.5
7	0.01 : 0.33 : 0.67 → Iris-virginica	0.332	Iris-virginica	6.2	2.8	4.8	1.8

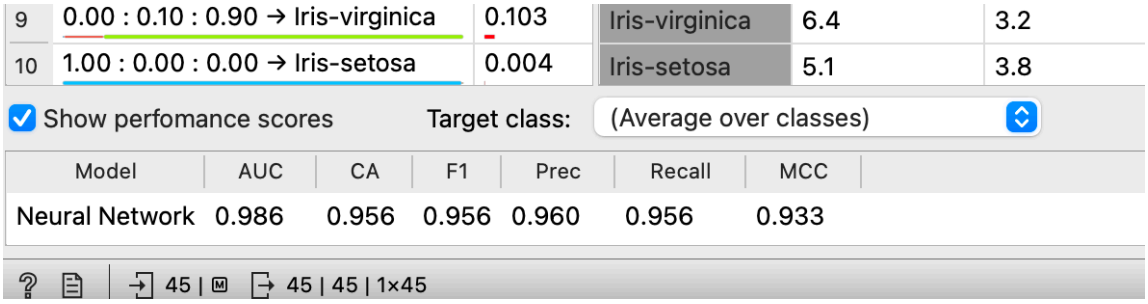
Model	AUC	CA	F1	Prec	Recall	MCC
Neural Network	0.986	0.956	0.956	0.960	0.956	0.933

You can see there is a Neural Network column (on the left), followed by the columns from the test data. In the test data, we can see in the image above that the first flower was a setosa, with a sepal length of 4.6cm, and sepal width of 3.4cm, etc. The corresponding

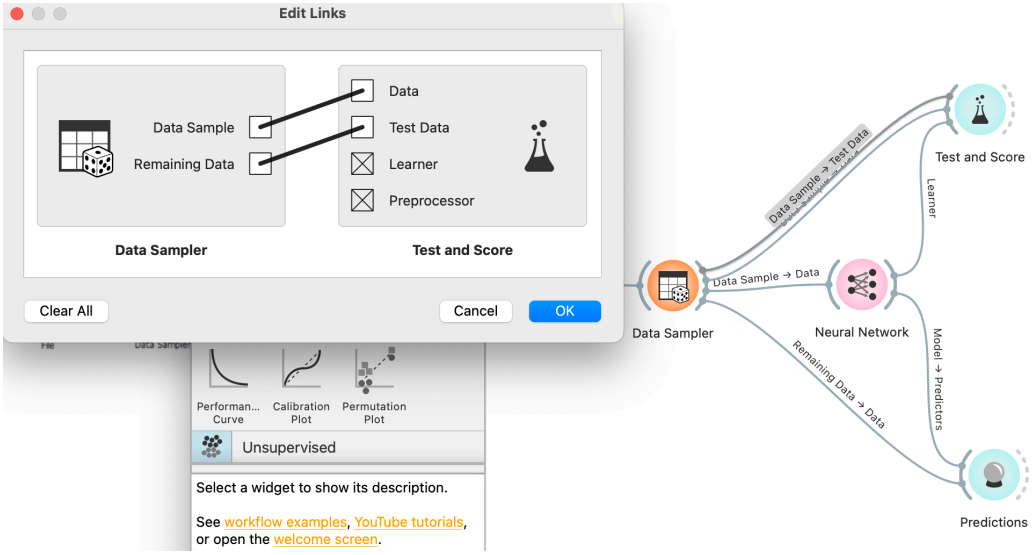
cell in the Neural Network column displays the probability of an Iris flower with these measurements belonging to each class, according to the predictions made by the Neural Network model.

For a more interesting example, look at instance 4. Here, we can see that the Neural Network model predicts a 99% probability that this instance is iris-setosa (the first class), a 1% chance that this instance is of the Iris-versicolor class (the second class), and an 0% chance that belongs to the Iris-virginica class (the third class). And thus, the neural network predicts the first class for this sample. which it then predicts. As we can see, this prediction was correct.

As you can see, there are five different measurements of the model's performance at the bottom of the window. Let's focus on the CA (Classifier Accuracy) for now. In this demonstration, we can see 95.6% of the predictions made on the test data were correct. Does your model also get 95.6%? If it doesn't, don't worry - remember that the 70-30 split of the data was random, and you will likely have trained and tested on slightly different parts of the dataset.

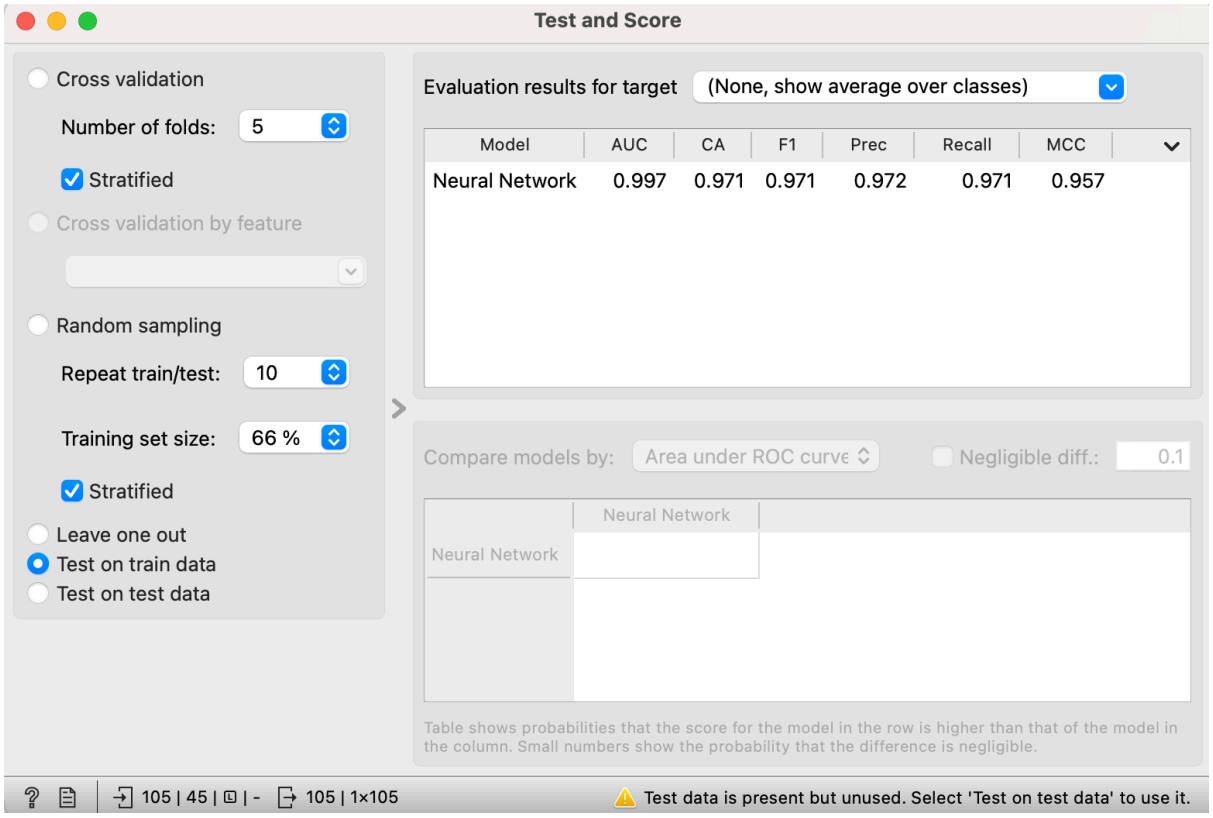


Let's have a look at how the model performs in more detail. Add a Test and Score widget, and ensure that the Data Sample is connected to the Data (telling Test and Score what data was used for training the model), and the Remaining Data is being used as Test Data.



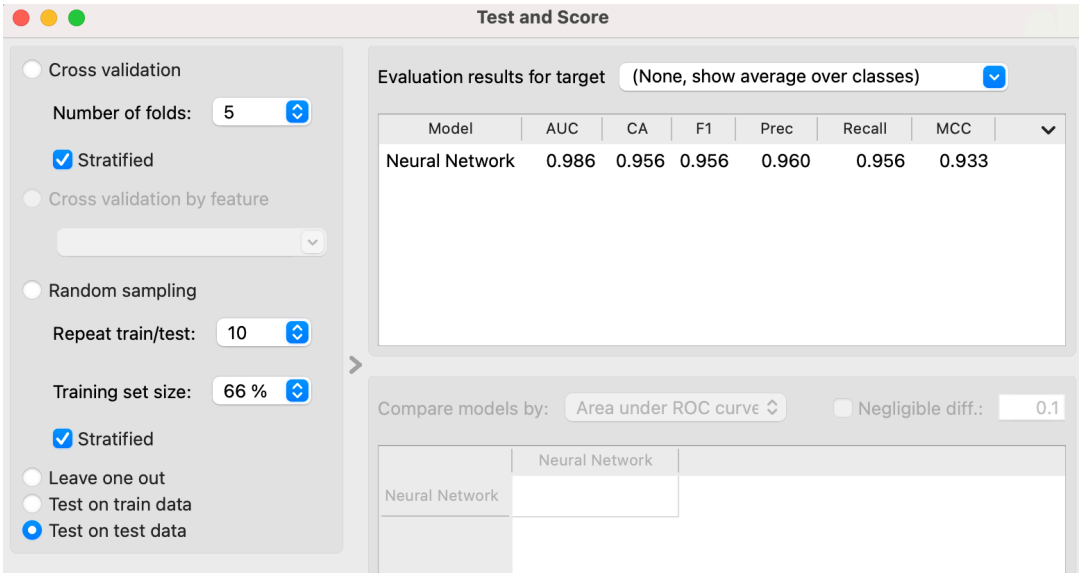
Click on Test and Score, and you will see a number of different options for evaluating your

Neural Network model. Begin by choosing the option Test on train data, and inspecting the Classifier Accuracy (CA). Here, we get a classification accuracy of 97.1% on the train data.



Now try evaluating it on the test data (which your model has never seen before).

Here, we get an accuracy of 95.6% on the test data (slightly lower but not much different than we got on the train data).

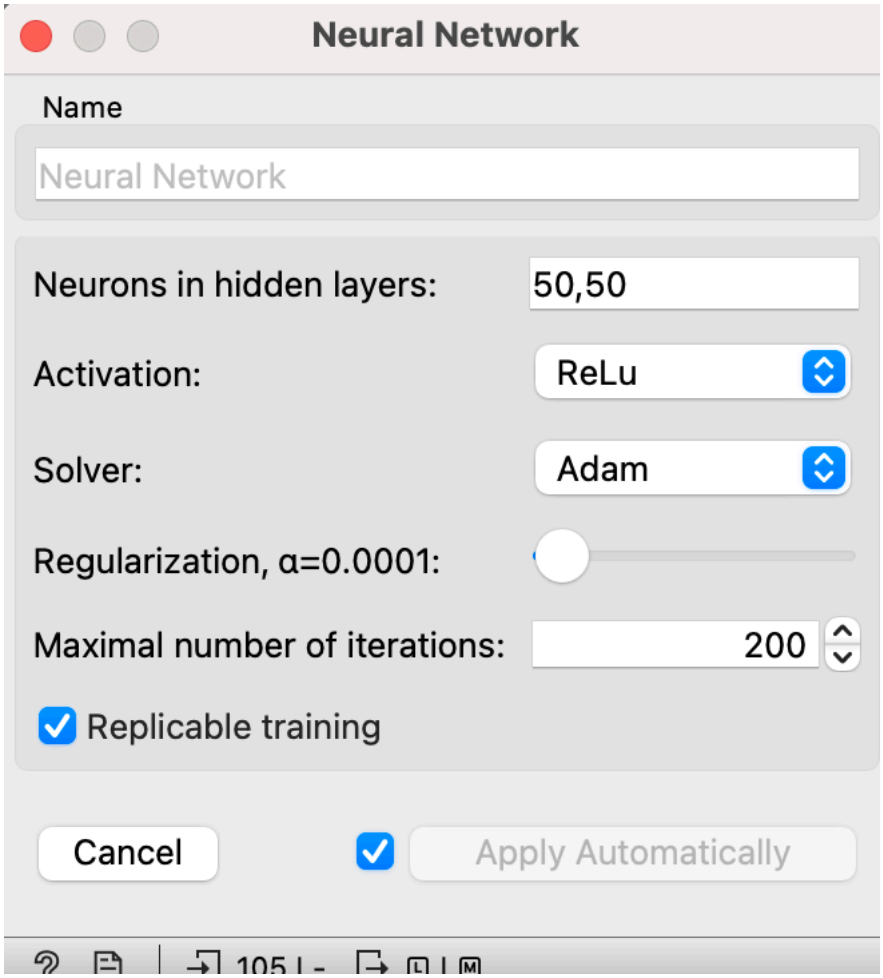


Depending upon your case and how good the model has trained, the Test on test data might give you a classification accuracy that will be very different than the accuracy on the train data. Here, the classification accuracy on the test data is very much similar (or at least very close) to the accuracy on the train data. Think about it. It is actually because the Iris data is very simple in comparison to most classification problems and we aren't seeing what normally happens in machine learning.

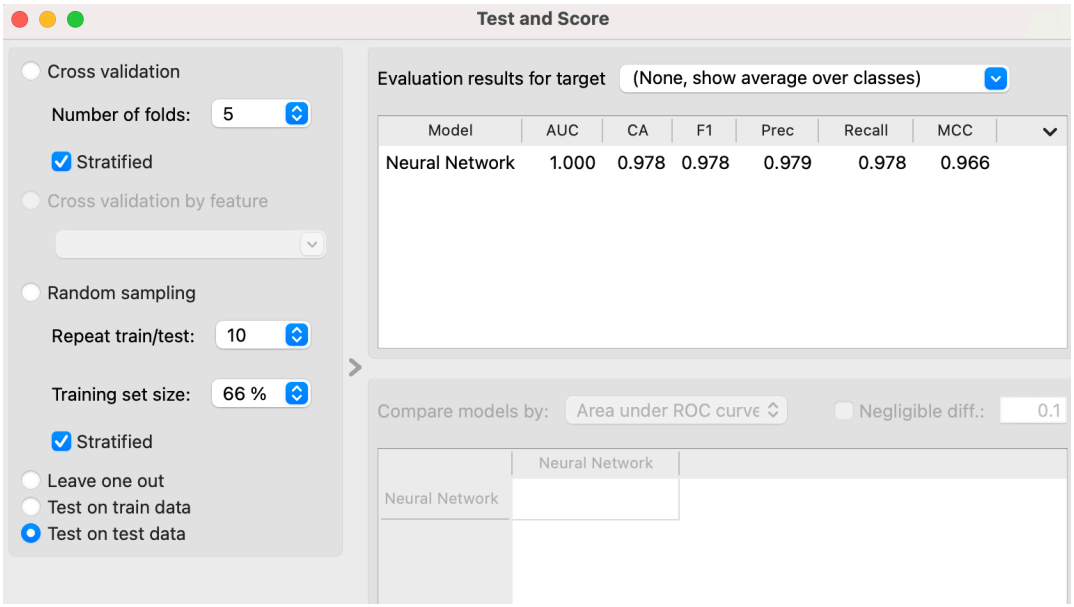
Additional task 1:

Next, we can configure the neural network and change the number of hidden units (neurons), and even the learning method. You can try many different values.

Click on the Neural Network widget and change the number of neurons in the hidden layers. By default, it could be set to 100.



Now, evaluate the model and check the Test and Score. Does it change the performance in terms of the classification accuracy on the test data. Yes, it does.



Additional task 2: Change the optimizer and observe any changes in the results.

Additional task 3:

Additional task 2: Use a logistic regression model instead of a neural network model and compare the performance with the neural network.