Machine Learning

Breeze can use a wide range of generic and specific Machine learning algorithms. Breeze can also import models created in other Machine learning frameworks which can export the models to ONNX format.

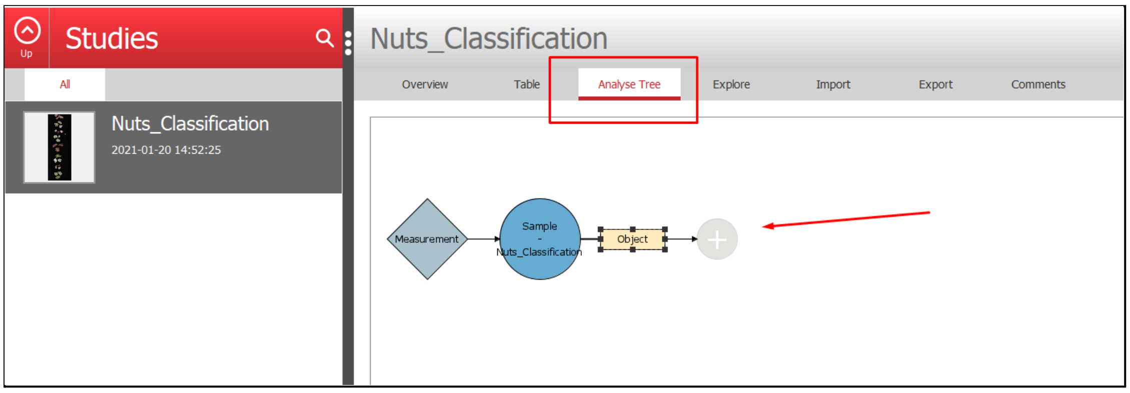
Before doing this tutorial, first do the tutorial Classification of nuts – step 2 advanced.

In this tutorial you will create a "Representative pixel segmentation" to expand on the existing data. Train a machine learning algorithm to perform classification using the *Auto* evaluation mode to have Breeze select and train the most suitable algorithm. Inspect and fine tune the model, and finally have a ready-to-run real time classification workflow.

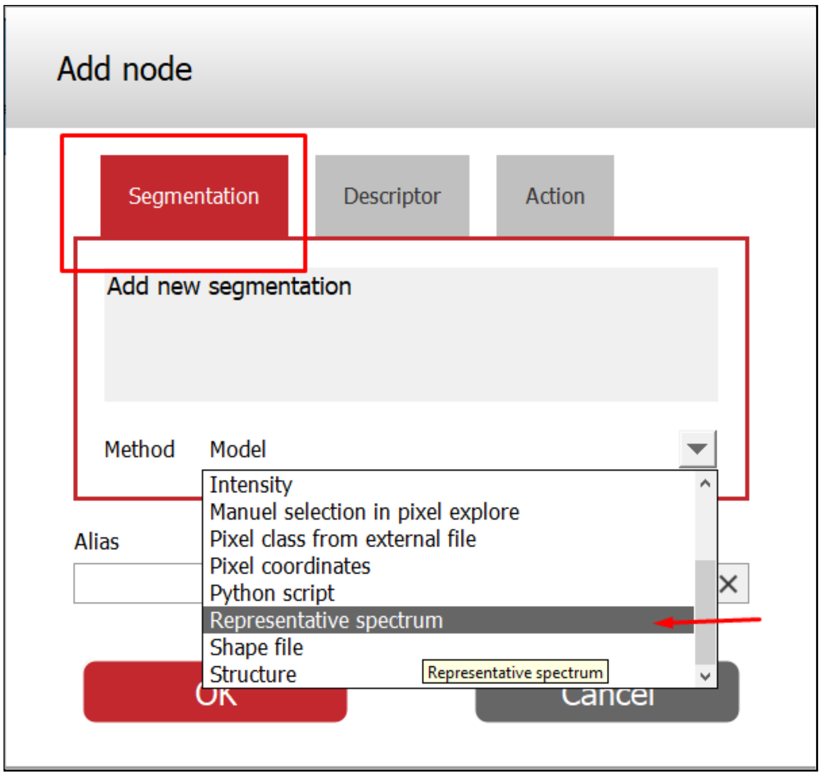
Steps included in tutorial		
Record	**Model**	**Play**
1. Create representative spectrum	2. Create a machine learning classification model	3. Real time predictions

Create representative spectrum

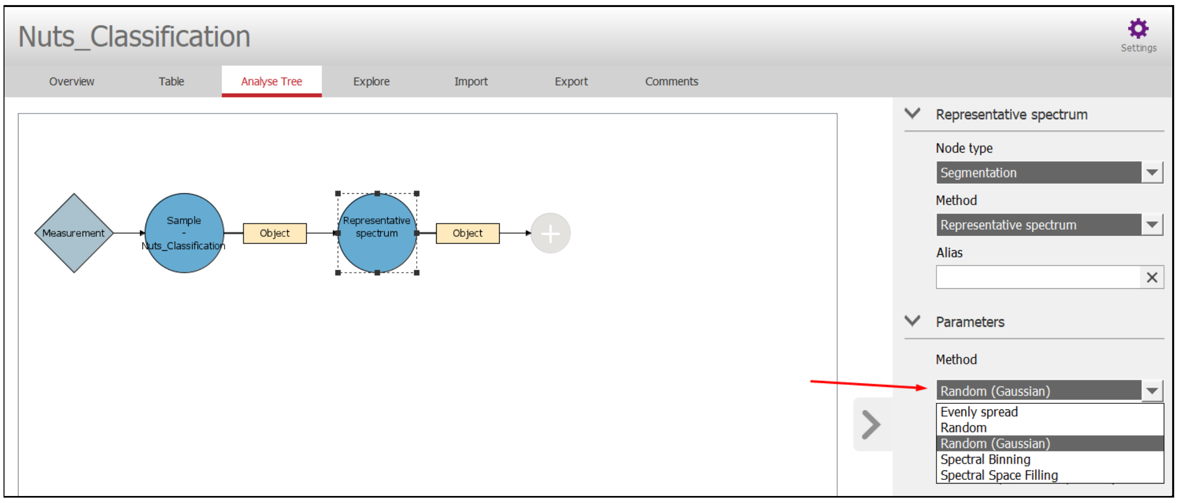
Open the study for the "Classification of Nuts step 2" tutorial that you have previously done. **In Record enter the Analysis Tree** tab and add a new node after the segmentation object node **by pressing the plus sign**.



Select the Segmentation tab in the "Add node" window and **select the "Representative spectrum"** method.



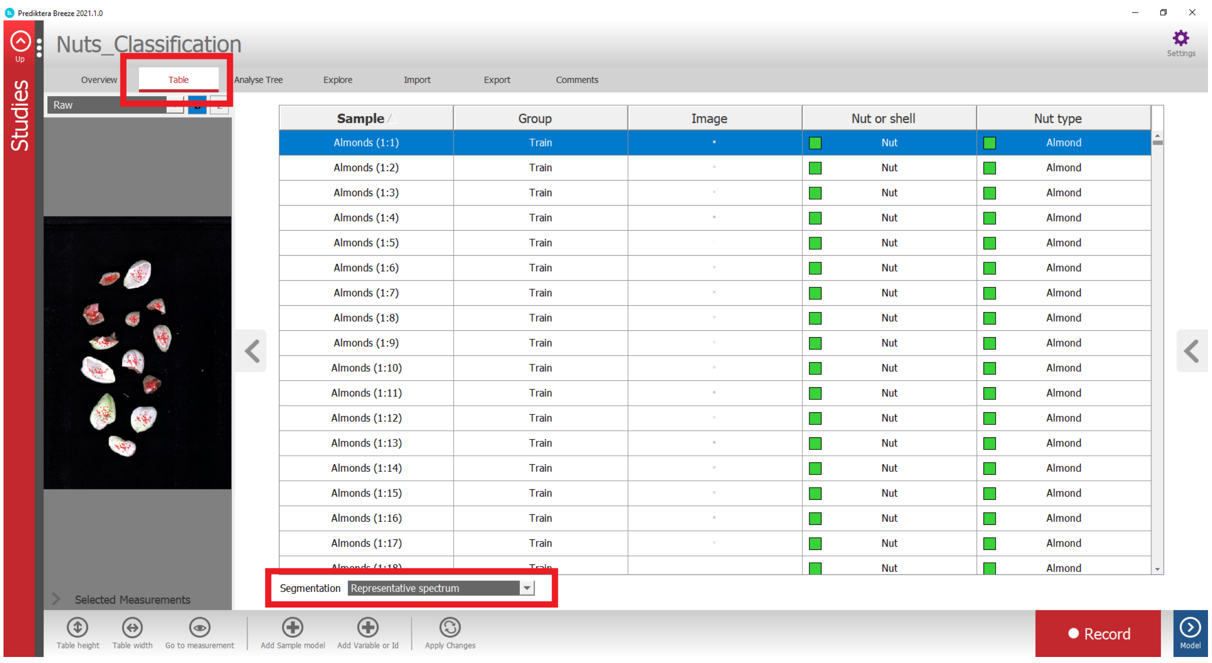
Set Method as "Random (Gaussian)" for a random distribution with a Gaussian (normal) distribution.



Set "Number" to 50 for a total of 50 pixels to be included from each object using the random distribution. **Set "Unique" to "True"** so no pixels are repeated even if all pixels in the object have already been selected.

Press **Apply changes** to apply the new segmentation step

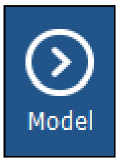
Go to the Table tab and **select the new “Representative spectrum” segmentation** in the drop down menu under the table. In the image on the left of the screen you see that 50 pixels have been selected from each object.



Sample	Group	Image	Nut or shell	Nut type
Almonds (1:1)	Train		🟩 Nut	🟩 Almond
Almonds (1:2)	Train		🟩 Nut	🟩 Almond
Almonds (1:3)	Train		🟩 Nut	🟩 Almond
Almonds (1:4)	Train		🟩 Nut	🟩 Almond
Almonds (1:5)	Train		🟩 Nut	🟩 Almond
Almonds (1:6)	Train		🟩 Nut	🟩 Almond
Almonds (1:7)	Train		🟩 Nut	🟩 Almond
Almonds (1:8)	Train		🟩 Nut	🟩 Almond
Almonds (1:9)	Train		🟩 Nut	🟩 Almond
Almonds (1:10)	Train		🟩 Nut	🟩 Almond
Almonds (1:11)	Train		🟩 Nut	🟩 Almond
Almonds (1:12)	Train		🟩 Nut	🟩 Almond
Almonds (1:13)	Train		🟩 Nut	🟩 Almond
Almonds (1:14)	Train		🟩 Nut	🟩 Almond
Almonds (1:15)	Train		🟩 Nut	🟩 Almond
Almonds (1:16)	Train		🟩 Nut	🟩 Almond
Almonds (1:17)	Train		🟩 Nut	🟩 Almond
Almonds (1:18)	Train		🟩 Nut	🟩 Almond

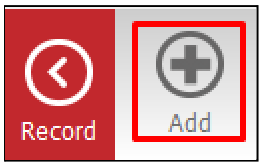
Segmentation: Representative spectrum

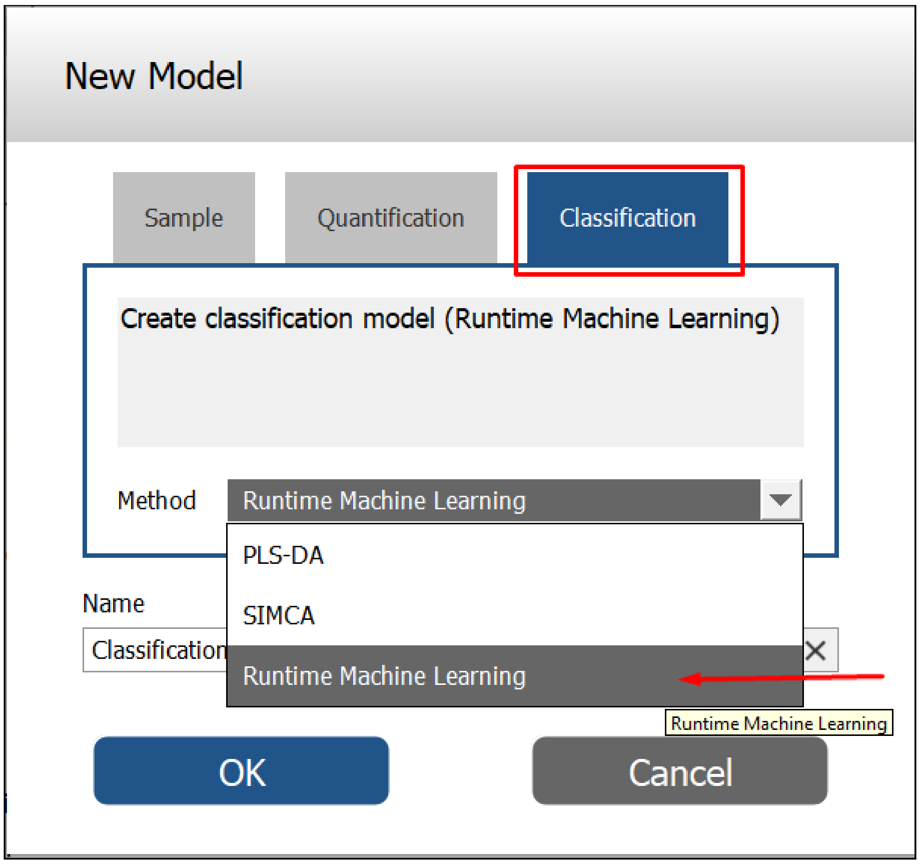
Go to the Model view.



Create a Machine Learning model

Create a new Classification model and select the Method “Runtime Machine Learning”.



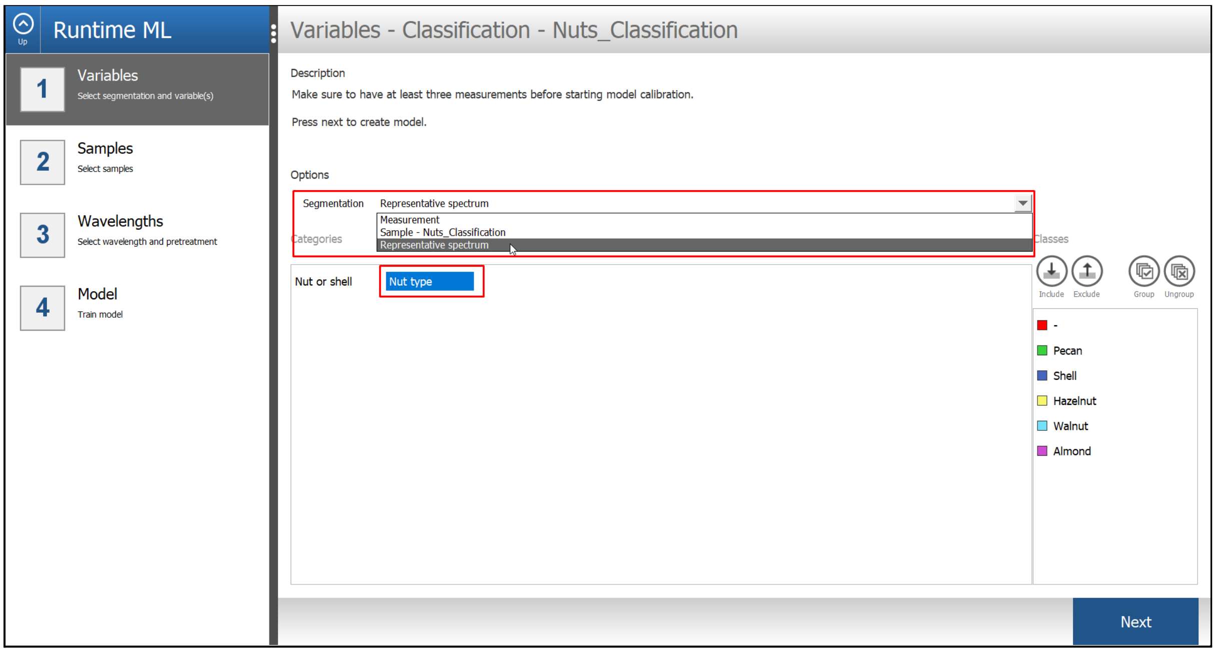

New Model

Create classification model (Runtime Machine Learning)

Method: Runtime Machine Learning

Name: Classification

Select the newly created Segmentation "Representative spectrum" and **"Nut type"** as the variables to classify and click Next.



Variables - Classification - Nuts_Classification

Description

Make sure to have at least three measurements before starting model calibration.

Press next to create model.

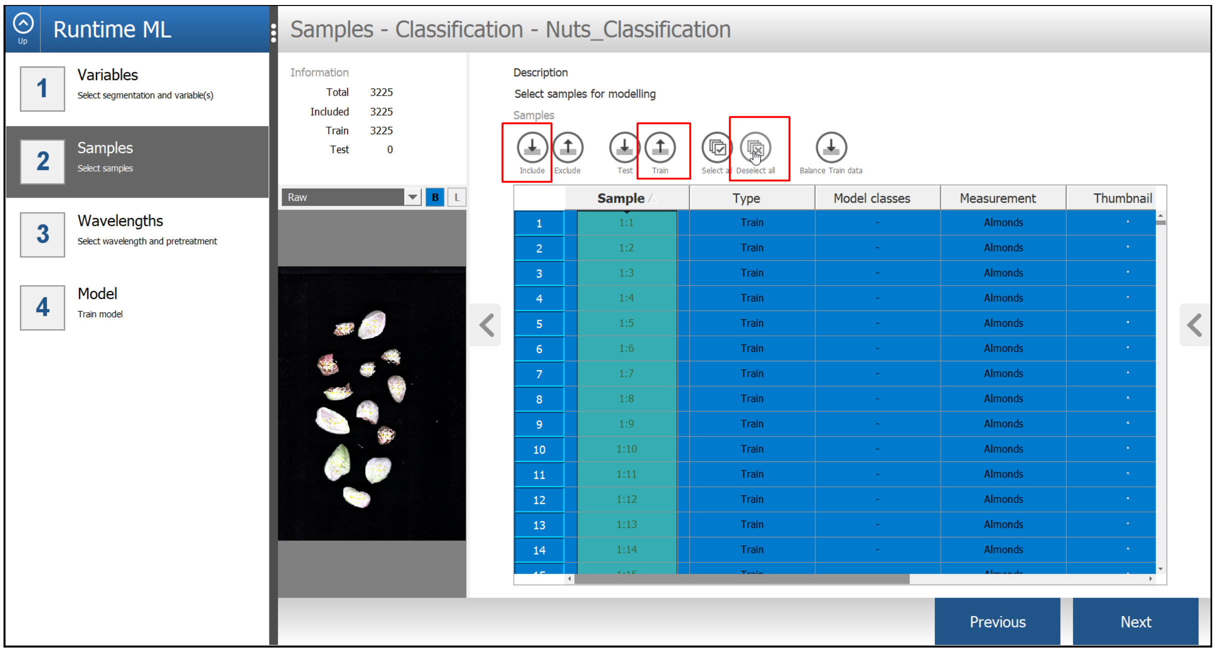
Options

Segmentation: Representative spectrum

Nut or shell: Nut type

- 🟥 -
- 🟩 Pecan
- 🟦 Shell
- 🟨 Hazelnut
- 🟦 Walnut
- 🟪 Almond

Click the Next button. **Click the "Select all button"** to select all samples, **click the "Include" button** to include all samples in the model and **click "Train" button** to add all samples to the training data for the model.



Samples - Classification - Nuts_Classification

Information

Total	3225
Included	3225
Train	3225
Test	0

Description

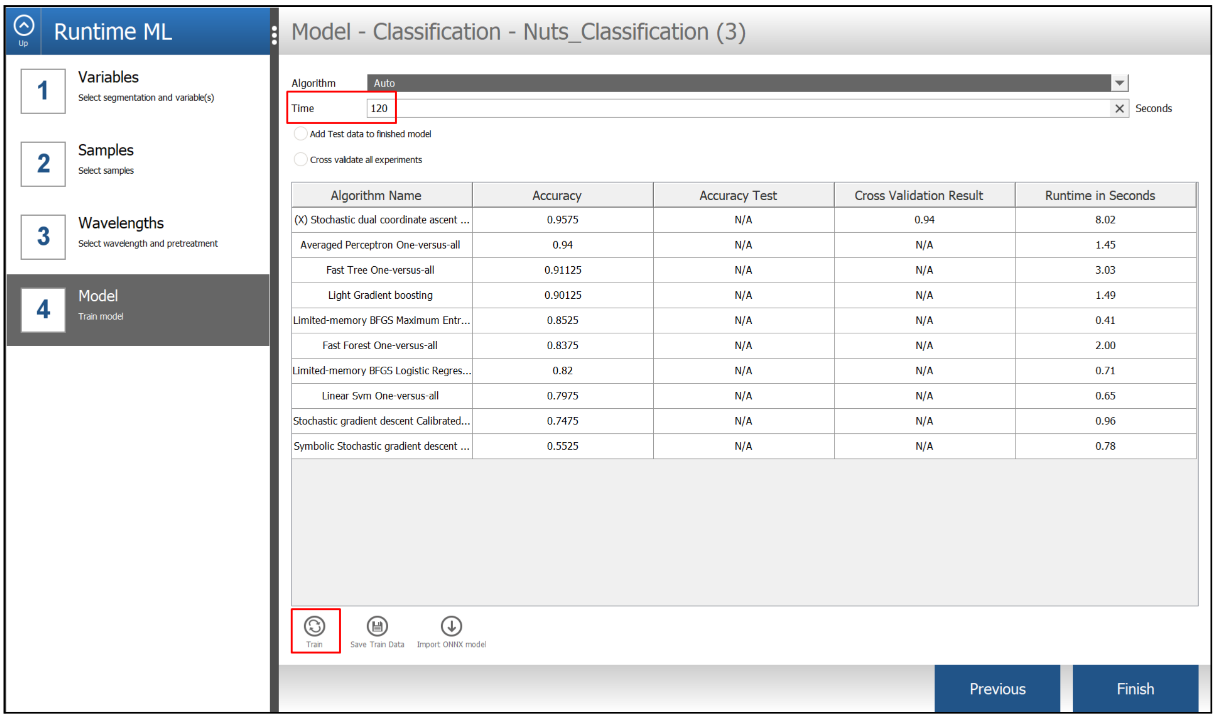
Select samples for modelling

Samples

	Sample	Type	Model classes	Measurement	Thumbnail
1	1:1	Train	-	Almonds	·
2	1:2	Train	-	Almonds	·
3	1:3	Train	-	Almonds	·
4	1:4	Train	-	Almonds	·
5	1:5	Train	-	Almonds	·
6	1:6	Train	-	Almonds	·
7	1:7	Train	-	Almonds	·
8	1:8	Train	-	Almonds	·
9	1:9	Train	-	Almonds	·
10	1:10	Train	-	Almonds	·
11	1:11	Train	-	Almonds	·
12	1:12	Train	-	Almonds	·
13	1:13	Train	-	Almonds	·
14	1:14	Train	-	Almonds	·

Click the "Next" button to inspect which wavelengths will be used in the model. Click the "Next" button to accept the default selections.

In the "Model" step of the wizard, set the training **Time to 100** (seconds) and the **Algorithm to Auto** (default). Then **click "Train"** to have Breeze perform an automatic evaluation of the training data using cross validation to find the best algorithm.



Model - Classification - Nuts_Classification (3)

Algorithm: Auto

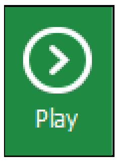
Time: 120 Seconds

Add Test data to finished model

Cross validate all experiments

Algorithm Name	Accuracy	Accuracy Test	Cross Validation Result	Runtime in Seconds
(X) Stochastic dual coordinate ascent ...	0.9575	N/A	0.94	8.02
Averaged Perceptron One-versus-all	0.94	N/A	N/A	1.45
Fast Tree One-versus-all	0.91125	N/A	N/A	3.03
Light Gradient boosting	0.90125	N/A	N/A	1.49
Limited-memory BFGS Maximum Entr...	0.8525	N/A	N/A	0.41
Fast Forest One-versus-all	0.8375	N/A	N/A	2.00
Limited-memory BFGS Logistic Regres...	0.82	N/A	N/A	0.71
Linear Svm One-versus-all	0.7975	N/A	N/A	0.65
Stochastic gradient descent Calibrated...	0.7475	N/A	N/A	0.96
Symbolic Stochastic gradient descent ...	0.5525	N/A	N/A	0.78

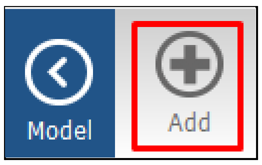
The table is sorted in descending order of performance. The algorithm on the top of the list will be used. **Click “Finish”** to accept the result and create a model of the best performing algorithm



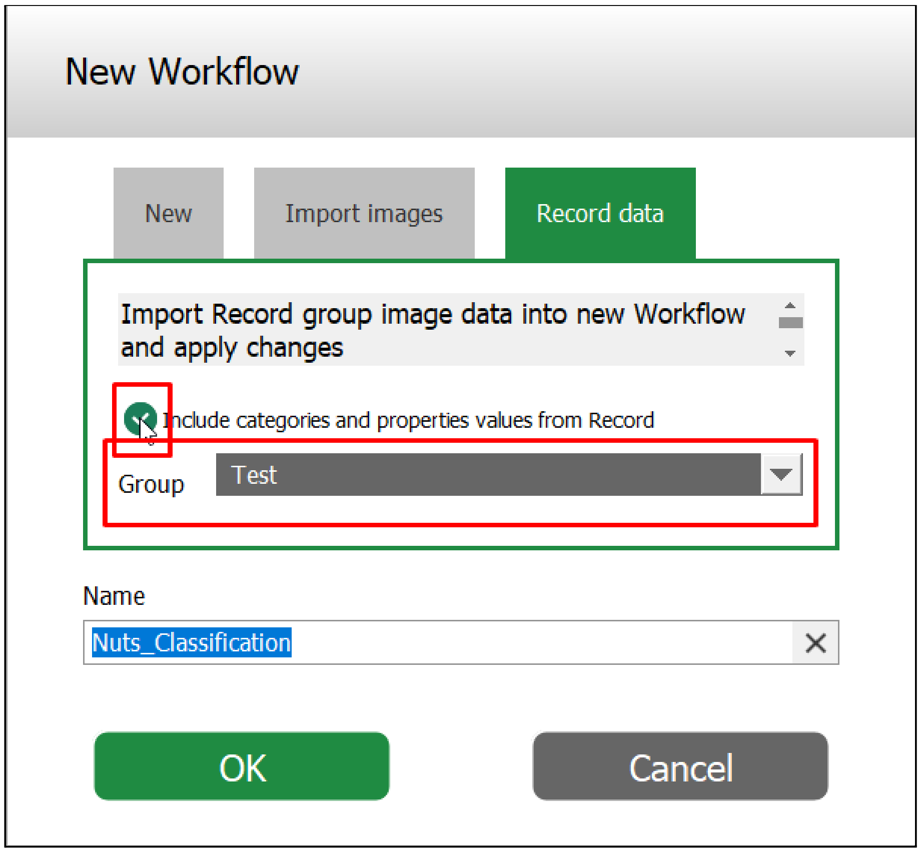
Click the “Play” button.

Real time predictions

Click the “Add” button.



Select the “Record” tab and make sure the **“Test” group is selected, check the “Include categories and properties from Record” option**.



Import Record group image data into new Workflow and apply changes

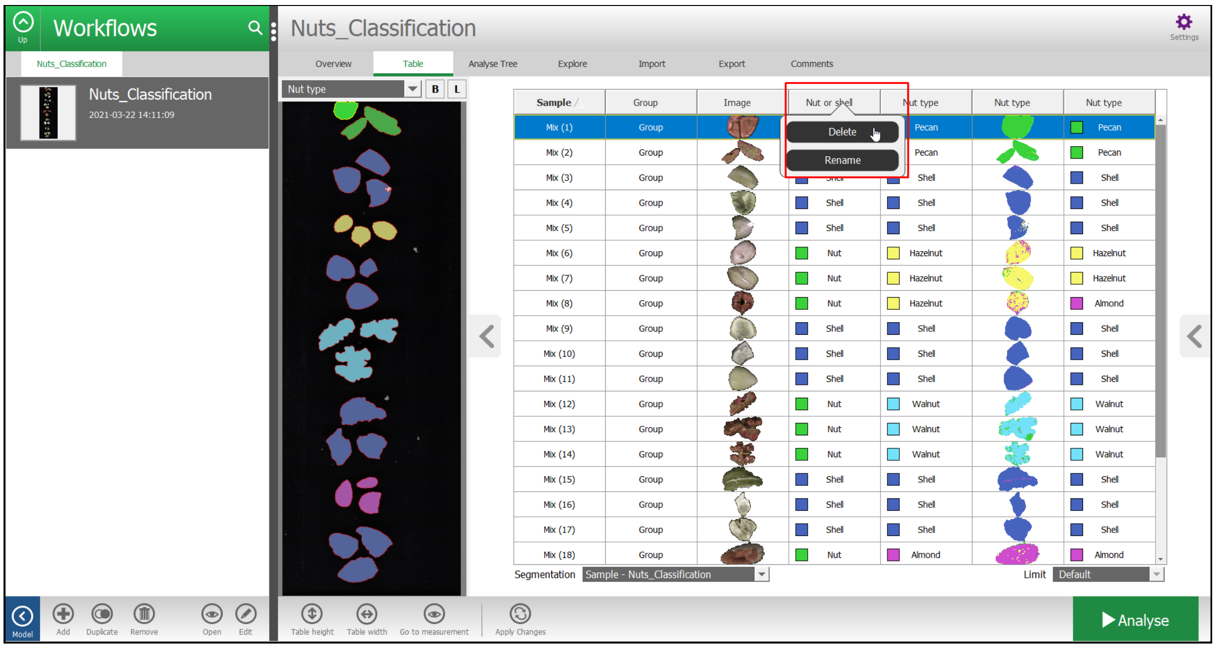
✓ Include categories and properties values from Record

Group: Test

Name: Nuts_Classification

In the “Table” tab you may notice several columns, the first two are the actual (correct) classes that were used for the training. Then you have one column that is the results for using the Machine Learning classification model created above.

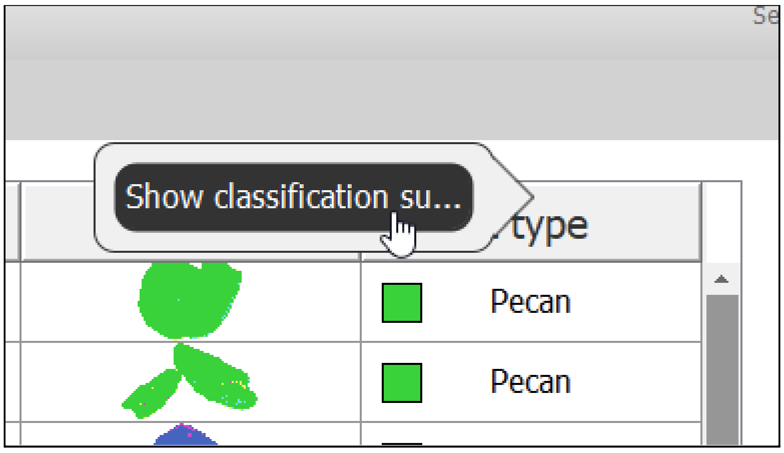
You can right click on the column header and **delete the “Nut or Shell” column** as we are not going to look at this classification now.



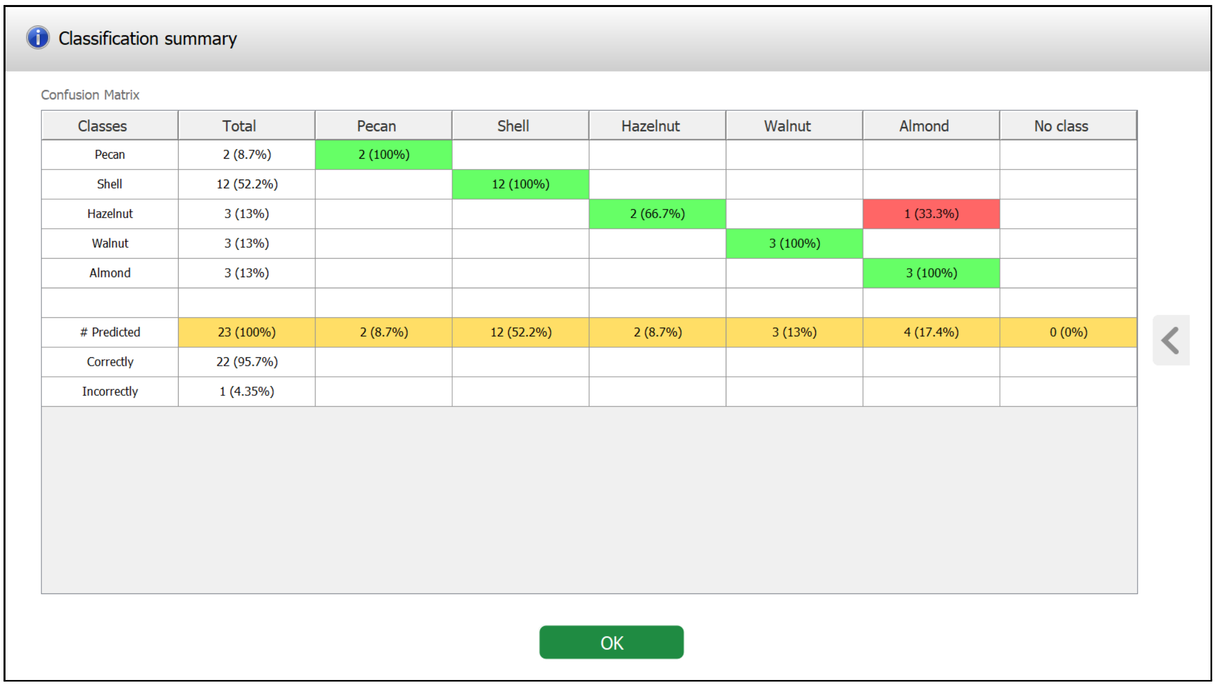
Sample	Group	Image	Nut or shell	Nut type	Nut type	Nut type
Mix (1)	Group		[illegible]	Pecan		🟩 Pecan
Mix (2)	Group		[illegible]	Pecan		🟩 Pecan
Mix (3)	Group		[illegible]	🟦 Shell		🟦 Shell
Mix (4)	Group		🟦 Shell	🟦 Shell		🟦 Shell
Mix (5)	Group		🟦 Shell	🟦 Shell		🟦 Shell
Mix (6)	Group		🟩 Nut	🟨 Hazelnut		🟨 Hazelnut
Mix (7)	Group		🟩 Nut	🟨 Hazelnut		🟨 Hazelnut
Mix (8)	Group		🟩 Nut	🟨 Hazelnut		🟪 Almond
Mix (9)	Group		🟦 Shell	🟦 Shell		🟦 Shell
Mix (10)	Group		🟦 Shell	🟦 Shell		🟦 Shell
Mix (11)	Group		🟦 Shell	🟦 Shell		🟦 Shell
Mix (12)	Group		🟩 Nut	🟦 Walnut		🟦 Walnut
Mix (13)	Group		🟩 Nut	🟦 Walnut		🟦 Walnut
Mix (14)	Group		🟩 Nut	🟦 Walnut		🟦 Walnut
Mix (15)	Group		🟦 Shell	🟦 Shell		🟦 Shell
Mix (16)	Group		🟦 Shell	🟦 Shell		🟦 Shell
Mix (17)	Group		🟦 Shell	🟦 Shell		🟦 Shell
Mix (18)	Group		🟩 Nut	🟪 Almond		🟪 Almond

Segmentation: Sample - Nuts_Classification
Limit: Default

Inspect the results between the predicted Nut type and actual. **Right click on the prediction column "Nut type" and click "Show classification summary"**.



The summary will indicate if there are any misclassifications of the objects. In this example one Hazelnut is misclassified as an Almond.

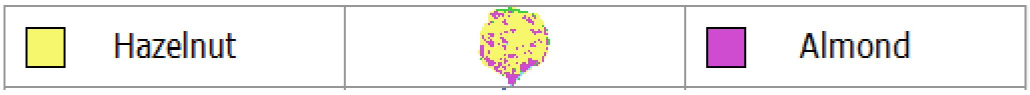


Classification summary

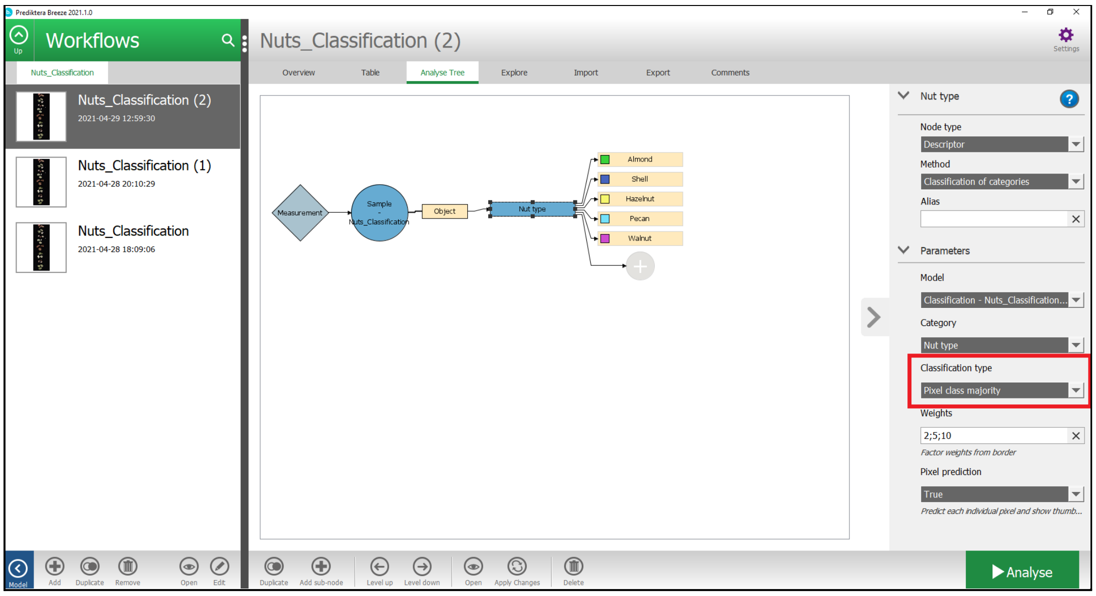
Confusion Matrix

Classes	Total	Pecan	Shell	Hazelnut	Walnut	Almond	No class
Pecan	2 (8.7%)	2 (100%)					
Shell	12 (52.2%)		12 (100%)				
Hazelnut	3 (13%)			2 (66.7%)		1 (33.3%)	
Walnut	3 (13%)				3 (100%)		
Almond	3 (13%)					3 (100%)	
# Predicted	23 (100%)	2 (8.7%)	12 (52.2%)	2 (8.7%)	3 (13%)	4 (17.4%)	0 (0%)
Correctly	22 (95.7%)						
Incorrectly	1 (4.35%)						

When inspecting the table we can see the misclassified nut (i.e. Hazelnut misclassified as Almond) we can see that the majority of the pixels are classified correctly but the object classification is incorrect (this may not be the case in your model). The misclassification of these pixels is due to the training data where we used “Representative pixels” and the prediction which is done at an object level using the object's average spectrum.



To remedy this we can change to object classification method, switching to the Analys Tree tab and selecting the classification node. **Change the model “Classification type” from “Object average spectrum” to “Pixels class majority”**.



Nut type

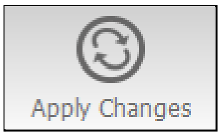
Node type: Descriptor
Method: Classification of categories
Alias:

Parameters

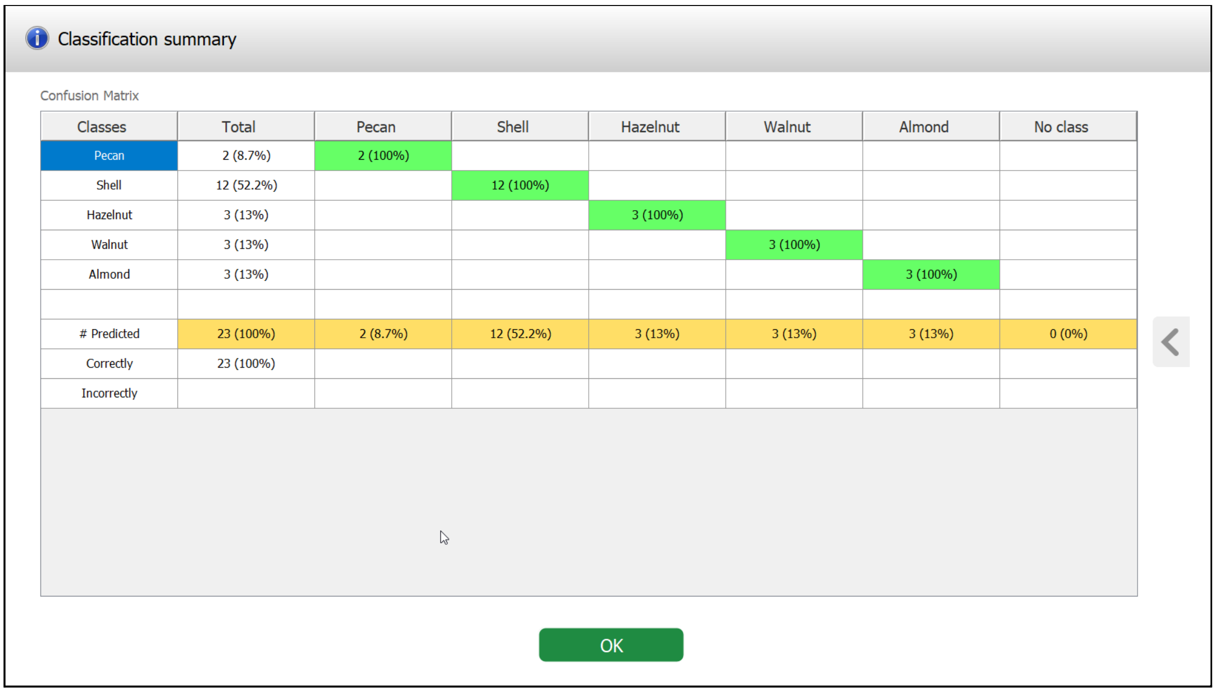
Model: Classification - Nuts_Classification...
Category: Nut type
Classification type: Pixel class majority
Weights: 2;5;10
Factor weights from border
Pixel prediction: True
Predict each individual pixel and show thumb...

You can **keep the default "Weights"** (2;5;10) which are 2 - the weight of the pixels closest to the edge of the object, 5 - second closest to the edge of the object and 10 - the center of the object. The default weight implice the center of the object is most representative of the spectrum and closer to the edge is less so.

Click "Apply Changes".



Now let us inspect the Table and the "Show classification summary" column context option.



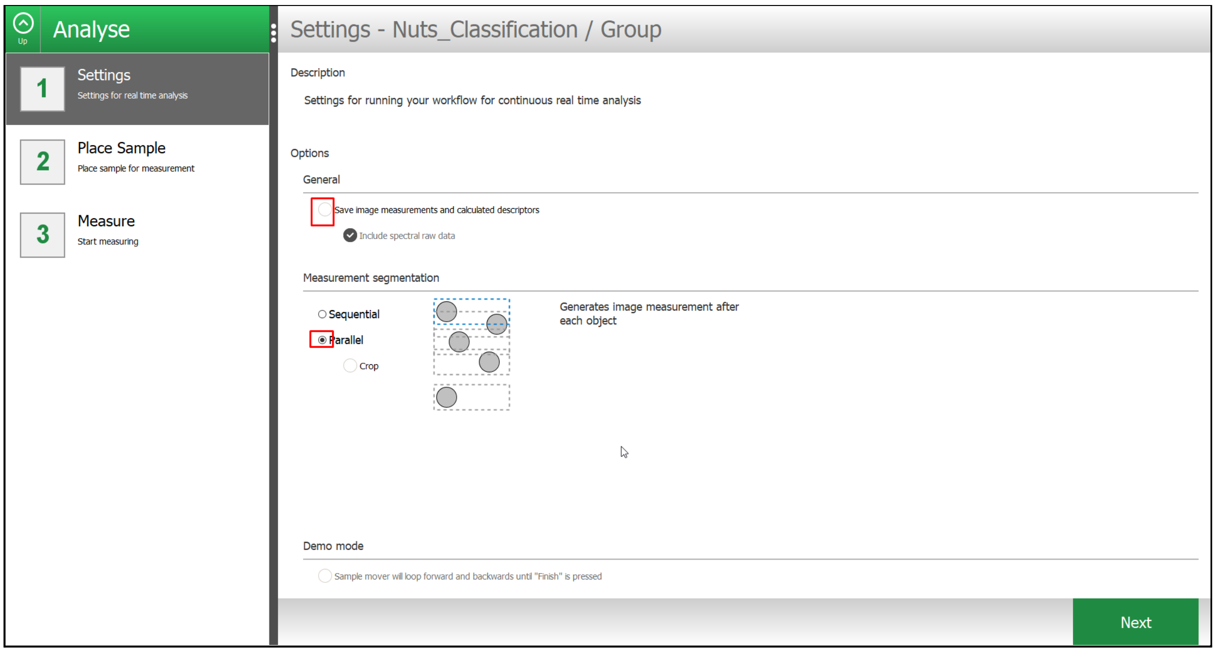
Confusion Matrix

Classes	Total	Pecan	Shell	Hazelnut	Walnut	Almond	No class
Pecan	2 (8.7%)	2 (100%)					
Shell	12 (52.2%)		12 (100%)				
Hazelnut	3 (13%)			3 (100%)			
Walnut	3 (13%)				3 (100%)		
Almond	3 (13%)					3 (100%)	
# Predicted	23 (100%)	2 (8.7%)	12 (52.2%)	3 (13%)	3 (13%)	3 (13%)	0 (0%)
Correctly	23 (100%)						
Incorrectly							

Your model should hopefully also receive a perfect object classification score. The pixel classification will not be uniformed but using tools available in Breeze the results can be tweaked to near perfection.

The model classification is now ready for real time predictions. To test this, **click on Analys** and the same data used in training the model and test classification will be evaluated in a real time scenario.

Uncheck the "Save image measurements and calculated descriptors" option and select "Parallel" Measurement segmentation.



Settings - Nuts_Classification / Group

Description

Settings for running your workflow for continuous real time analysis

Options

General

☐ Save image measurements and calculated descriptors

✓ Include spectral raw data

Measurement segmentation

○ Sequential

● Parallel

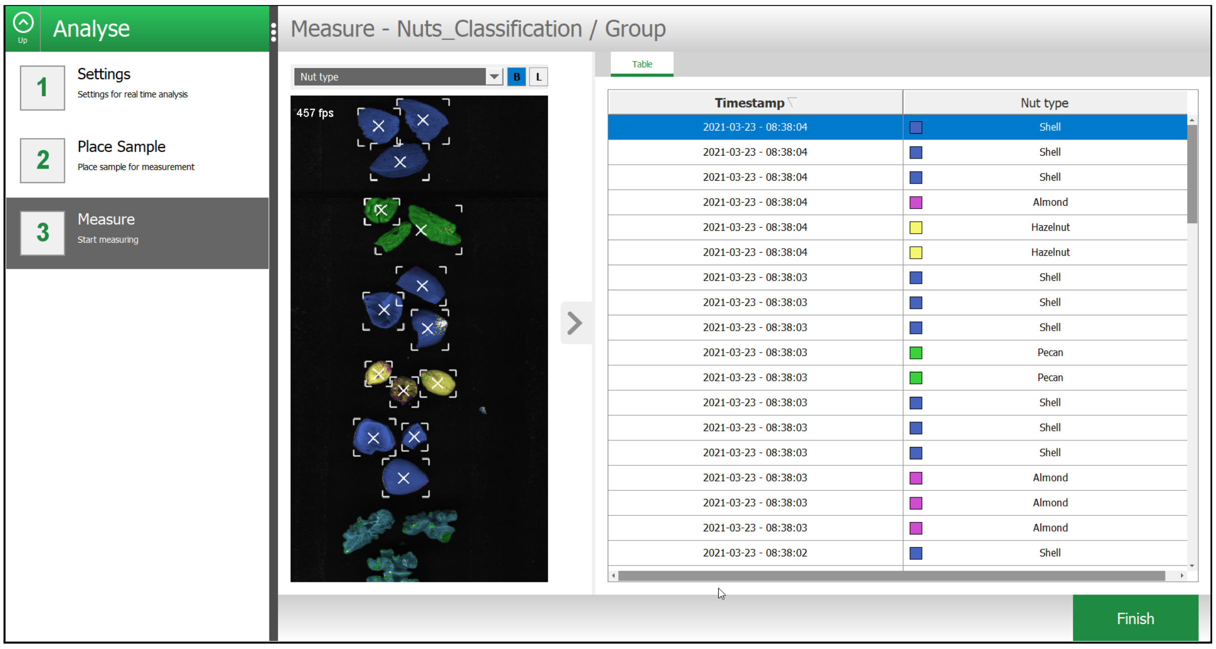
☐ Crop

Generates image measurement after each object

Demo mode

☐ Sample mover will loop forward and backwards until "Finish" is pressed

Click on “Next”.
Click on “Start”.



Measure - Nuts_Classification / Group

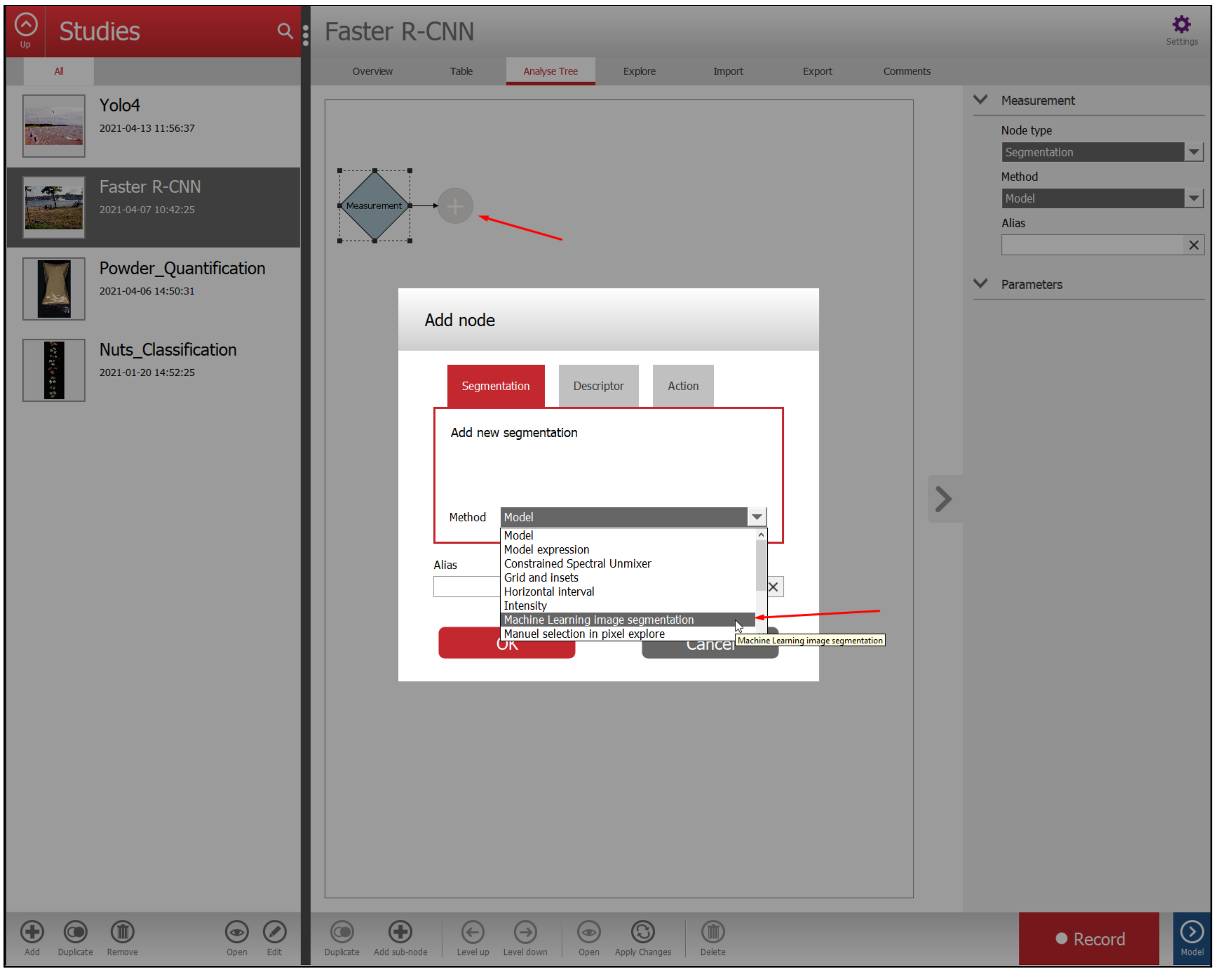
Timestamp		Nut type
2021-03-23 - 08:38:04	🟦	Shell
2021-03-23 - 08:38:04	🟦	Shell
2021-03-23 - 08:38:04	🟦	Shell
2021-03-23 - 08:38:04	🟪	Almond
2021-03-23 - 08:38:04	🟨	Hazelnut
2021-03-23 - 08:38:04	🟨	Hazelnut
2021-03-23 - 08:38:03	🟦	Shell
2021-03-23 - 08:38:03	🟦	Shell
2021-03-23 - 08:38:03	🟦	Shell
2021-03-23 - 08:38:03	🟩	Pecan
2021-03-23 - 08:38:03	🟩	Pecan
2021-03-23 - 08:38:03	🟦	Shell
2021-03-23 - 08:38:03	🟦	Shell
2021-03-23 - 08:38:03	🟦	Shell
2021-03-23 - 08:38:03	🟪	Almond
2021-03-23 - 08:38:03	🟪	Almond
2021-03-23 - 08:38:03	🟪	Almond
2021-03-23 - 08:38:02	🟦	Shell

Good job! Watch the classified nuts roll by and receive a perfect classification score. Click “Finish” to stop the predictions.

Appendix

ONNX image segmentation

Breeze supports some object detection algorithms via ONNX, currently *YOLOv4* and *Faster R-CNN*. To utilize this feature add a segmentation node to the Analys Tree, **Machine Learning image segmentation.**



Add new segmentation

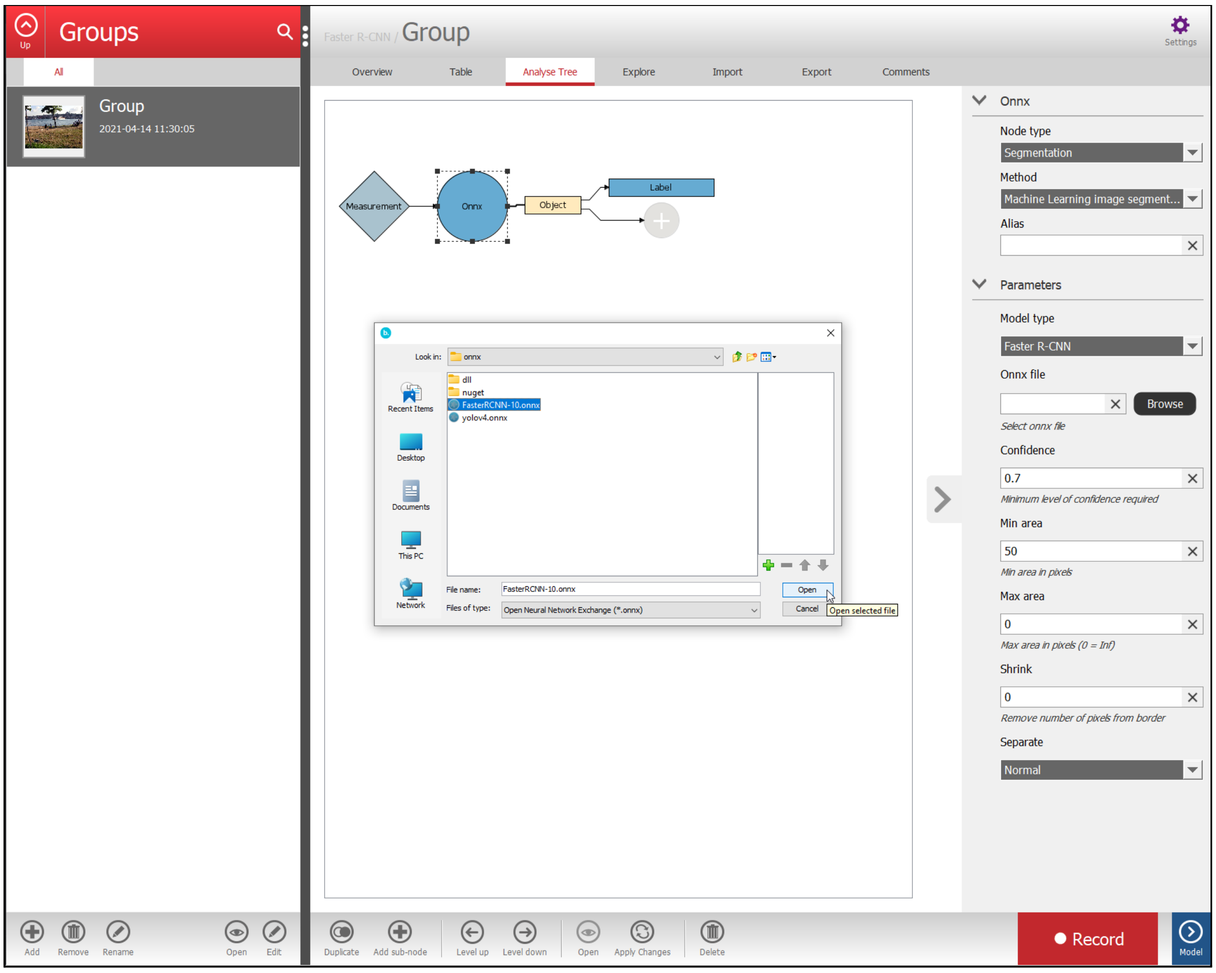
Method: Model

Model
Model expression
Constrained Spectral Unmixer
Grid and insets
Horizontal interval
Intensity
Machine Learning image segmentation
Manuel selection in pixel explore

Alias

Measurement
Node type: Segmentation
Method: Model
Alias
Parameters

Select your pretrained ONNX model type in the Model Type drop down and browse to and select the model file, in this case a *Faster R-CNN* model file and segmentation.



File name: FasterRCNN-10.onnx
Files of type: Open Neural Network Exchange (*.onnx)

Onnx

Node type: Segmentation

Method: Machine Learning image segment...

Alias:

Parameters

Model type: Faster R-CNN

Onnx file: *Select onnx file*

Confidence: 0.7 — *Minimum level of confidence required*

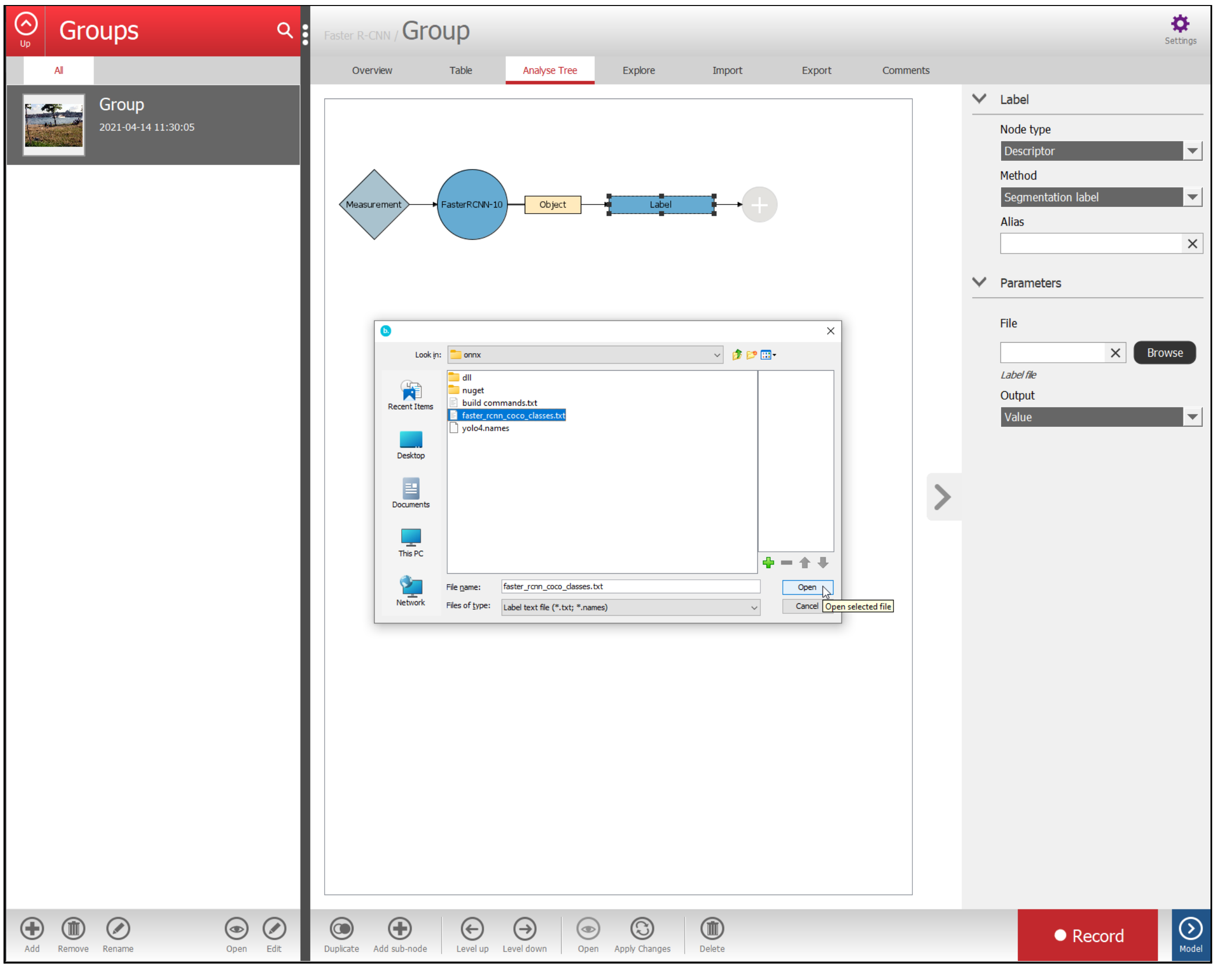
Min area: 50 — *Min area in pixels*

Max area: 0 — *Max area in pixels (0 = Inf)*

Shrink: 0 — *Remove number of pixels from border*

Separate: Normal

A Label classification node is automatically added when adding the machine learning segmentation. Add a new line separated class file to the Label node. May be in either *.txt* or *.names* files.



Label

Node type: Descriptor

Method: Segmentation label

Alias:

Parameters

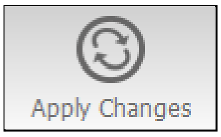
File:

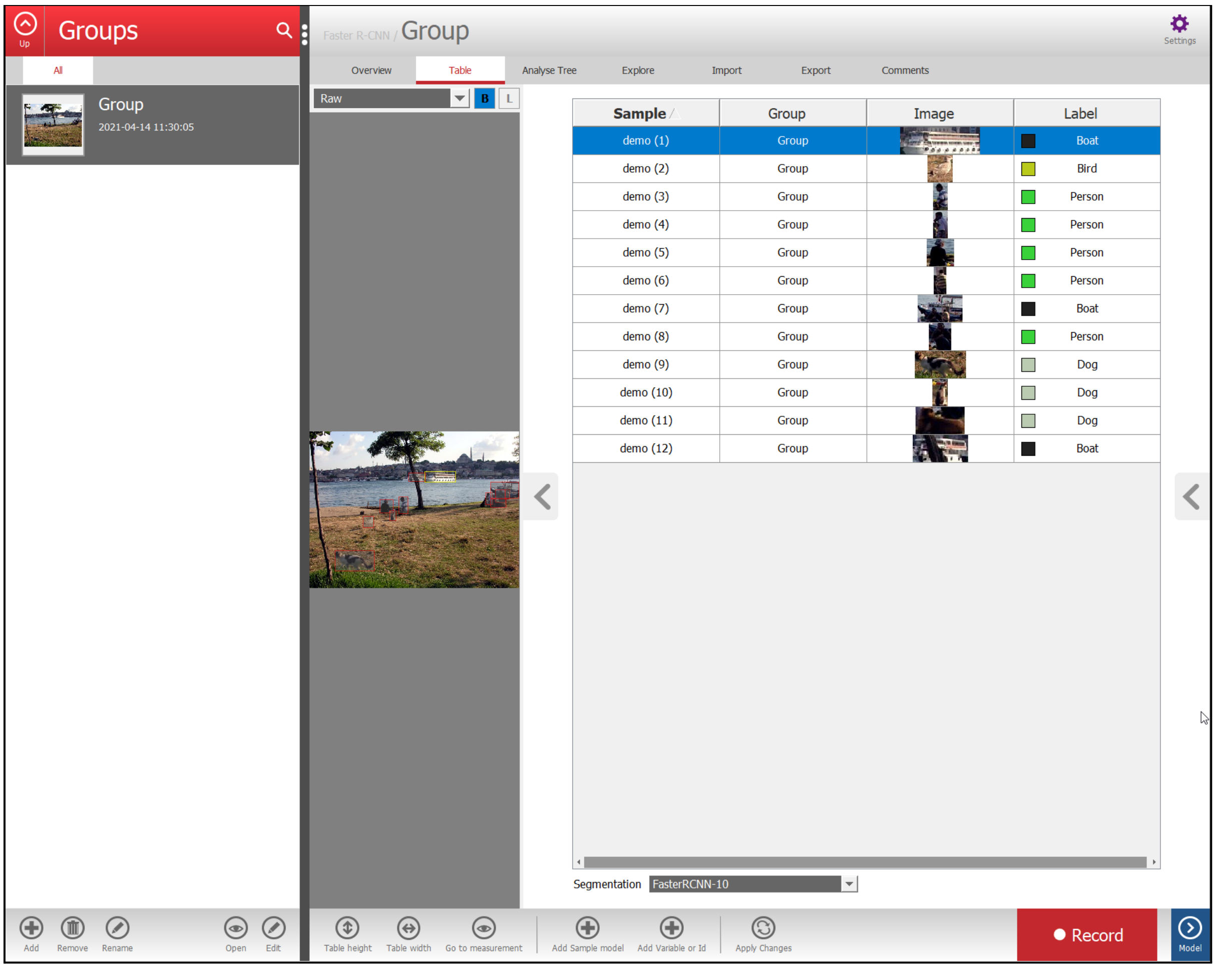
Output: Value

File name: faster_rcnn_coco_classes.txt

Files of type: Label text file (*.txt; *.names)

Click Apply Changes to evaluate the segmentation, go to Table tab to see the result.





Sample	Group	Image	Label
demo (1)	Group		⬛ Boat
demo (2)	Group		🟨 Bird
demo (3)	Group		🟩 Person
demo (4)	Group		🟩 Person
demo (5)	Group		🟩 Person
demo (6)	Group		🟩 Person
demo (7)	Group		⬛ Boat
demo (8)	Group		🟩 Person
demo (9)	Group		⬜ Dog
demo (10)	Group		⬜ Dog
demo (11)	Group		⬜ Dog
demo (12)	Group		⬛ Boat

Segmentation FasterRCNN-10

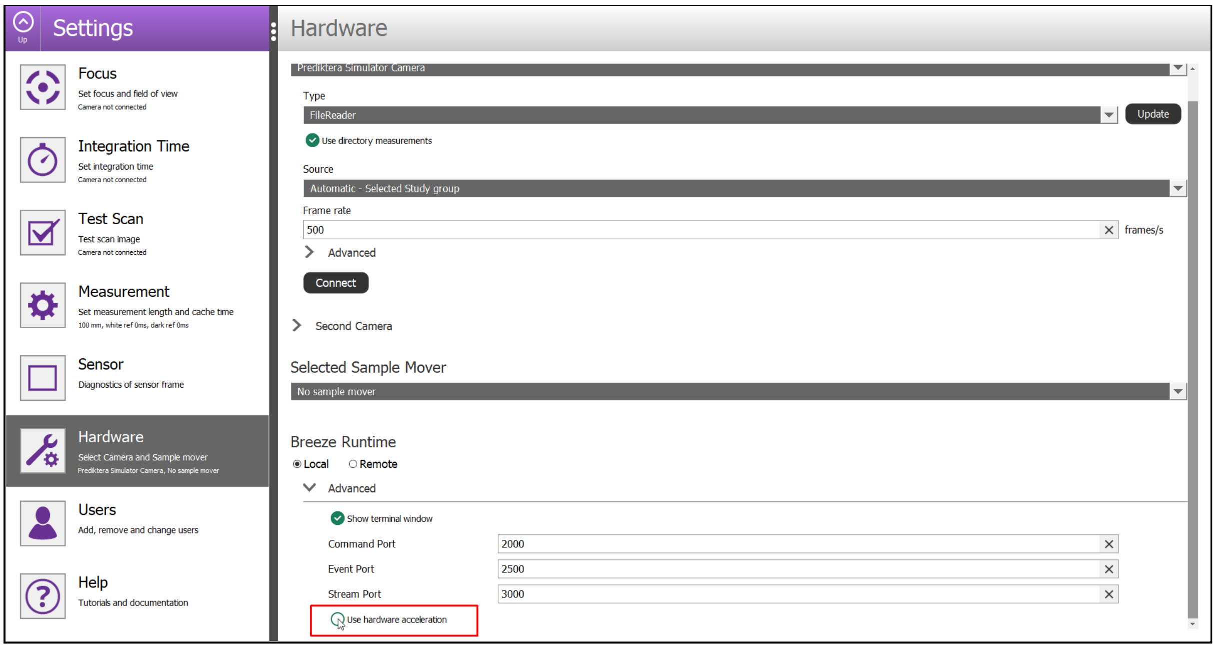
To train a custom object detection model we recommend using TensorFlow, here is an example which trains a YOLOv4 model: Tianxiaomo/pytorch-YOLOv4: PyTorch ,ONNX and TensorRT implementation of YOLOv4

Advanced

Enabling GPU acceleration

Disclaimer: This functionality is currently in an *experimental phase* and may be used for evaluation purposes but should not be deployed in a production environment.

To enabled is check the box in Settings:



Up
Settings
Focus
Set focus and field of view
Camera not connected
Integration Time
Set integration time
Camera not connected
Test Scan
Test scan image
Camera not connected
Measurement
Set measurement length and cache time
100 mm, white ref 0ms, dark ref 0ms
Sensor
Diagnostics of sensor frame
Hardware
Select Camera and Sample mover
Prediktera Simulator Camera, No sample mover
Users
Add, remove and change users
Help
Tutorials and documentation

Hardware

Prediktera Simulator Camera

Type: FileReader [Update]

✓ Use directory measurements

Source: Automatic - Selected Study group

Frame rate: 500 frames/s

Advanced

[Connect]

Second Camera

Selected Sample Mover: No sample mover

Breeze Runtime: [✓ Local] Remote

Advanced

✓ Show terminal window

Command Port	2000
Event Port	2500
Stream Port	3000

Use hardware acceleration

Windows

In Windows no additional steps need to be taken to enable GPU accelerated computing via the Microsoft DirectML which is included in Windows 10, version 1903 and newer.

To enable GPU acceleration via NVIDIA CUDA and TensorRT this option needs to be selected during the installation of Breeze. The required library files will then be downloaded and installed as part of the installation wizard.

Linux

NVIDIA CUDA can be enabled in Linux (tested for Ubuntu 18.04) using the script `enable-cuda.sh` found under the Breeze/Runtime directory. This will download the required library files.

Inference

When applicable to following order of priority is used:

1. ~~TensorRT[1]~~
2. DirectML[2]
3. CUDA[3]
4. Generic CPU (MLAS)

TensorRT / CUDA

~~NVIDIA TensorRT is a library that facilitates high performance inference on NVIDIA GPUs, and can be used on Windows OS~~. For Linux x64 OS generic CUDA can be used in a similar way.

Supported matrix (see NVIDIA support hardware) :

CUDA Toolkit	**Linux x86_64 Driver Version**	**Windows x86_64 Driver Version**
CUDA 11.2.0 GA	≥ 460.27.04	≥ 460.89

TensorRT requires that all inputs of the subgraphs have *shape* specified.

DirectML

On Windows 10 computer Breeze Runtime can utilize Direct Machine Learning (DirectML) GPU accelerated computing for faster prediction results.

Examples of compatible hardware include:

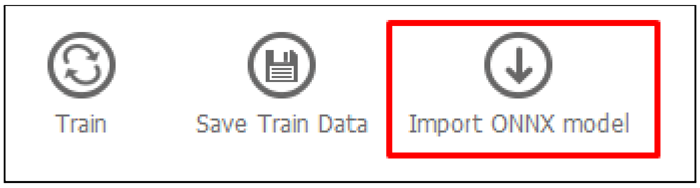
- NVIDIA Kepler (GTX 600 series) and above
- AMD GCN 1st Gen (Radeon HD 7000 series) and above
- Intel Haswell (4th-gen core) HD Integrated Graphics and above

DirectML is compatible with Windows 10, version 1709 (10.0.16299; RS3, “Fall Creators Update”) and newer.

[1] Not applicable in current version
[2] Windows only
[3] Linux x64 only

Import of ONNX model

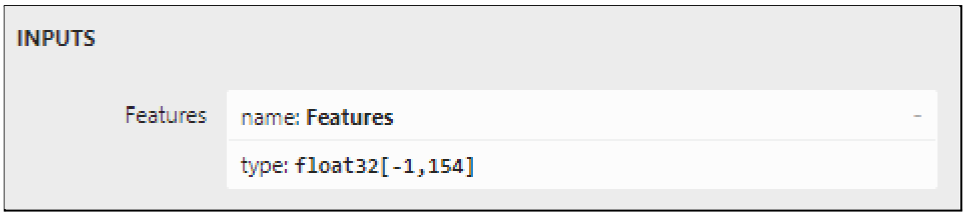
ONNX models created outside Breeze can be used to make classification and quantification predictions during runtime. To import an existing model use the *Import ONNX model* in Model view in Breeze.



The model must follow the standard format specified below for both name and datatype. Inputs and outputs are bound by name so the names must match exactly according to the specification below.

Classification model

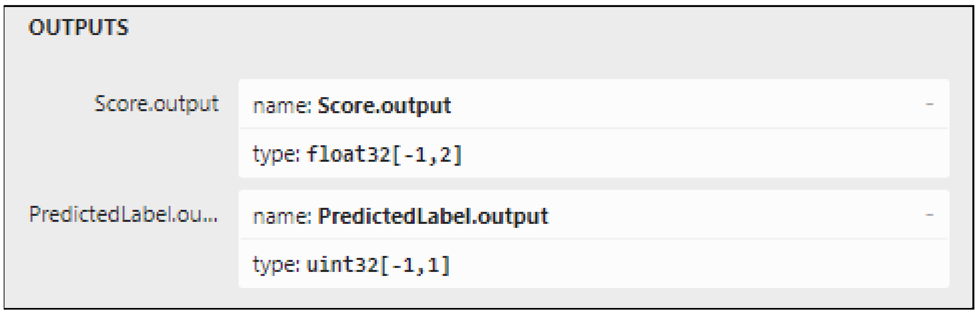
The model INPUTS and OUTPUTS of the model should be as below, the names and data types must match. The dimensions for the `INPUTS.Features` must be the same as the number of wavelengths in the Breeze model. In the example below *154* wavelengths are included in the Breeze model and also then in the ONNX model.



INPUTS

Features
name: Features
type: float32[-1,154]

The INPUTS must have the name *Features*.



OUTPUTS

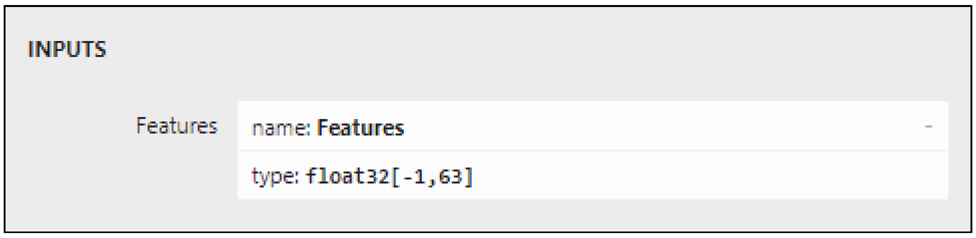
Score.output
name: Score.output
type: float32[-1,2]

PredictedLabel.ou...
name: PredictedLabel.output
type: uint32[-1,1]

For the classification model the above OUTPUTS must exist in the listed format. The dimension for the `Score.output` variable must be the same as the number of classes available in Breeze. `PredictedLabel.output` is the prediction result in a single dimension variable.

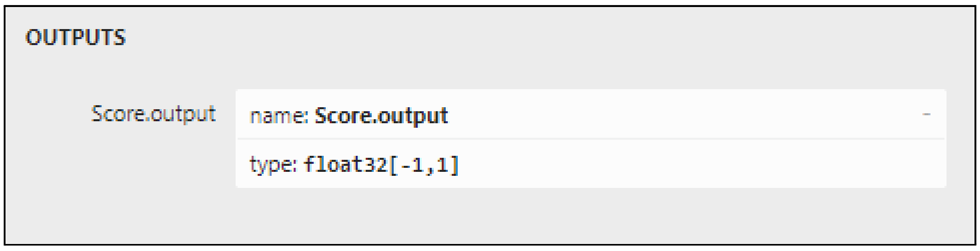
Quantification model

The INPUTS are the same as for Classification (wavelengths in model).


INPUTS

Features
name: **Features**
type: `float32[-1,63]`

For OUTPUTS only `Score.output` is applicable for quantification. The dimensions are the number of variables used in the Breeze model.


OUTPUTS

Score.output
name: **Score.output**
type: `float32[-1,1]`