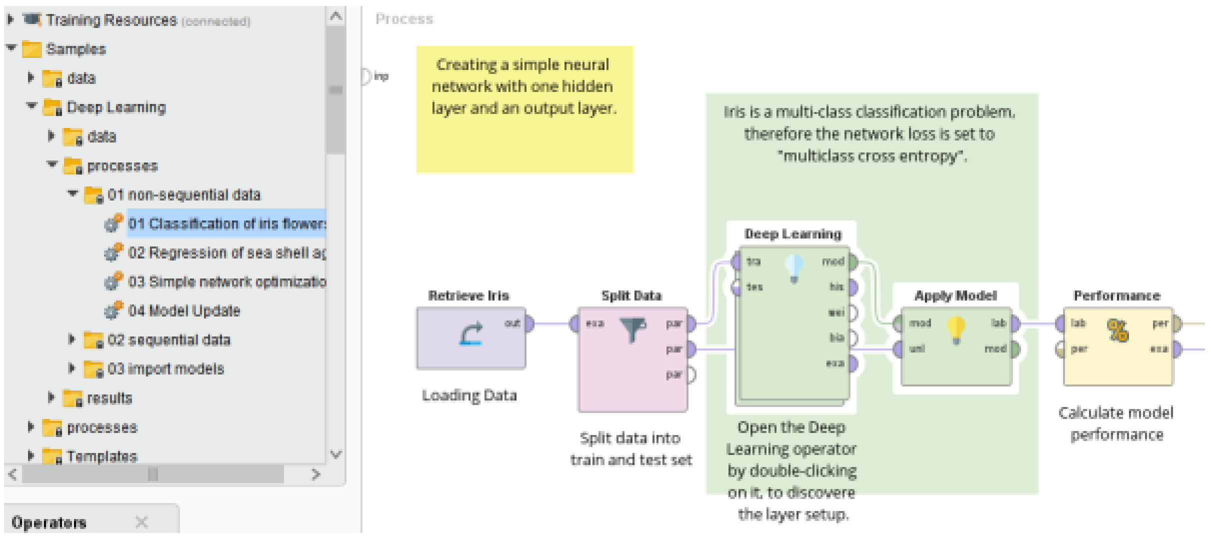
Supervised Learning / Deep Learning

More Deep Learning Examples...

Explore the examples of the Deep Learning Extension.

Samples

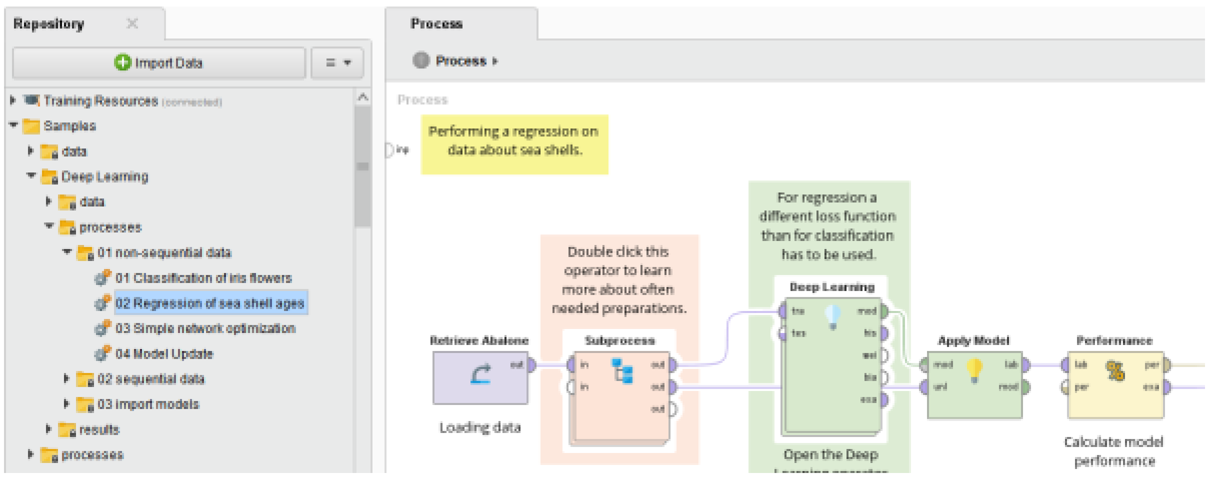

Training Resources (connected)
Samples
data
Deep Learning
data
processes
01 non-sequential data
01 Classification of iris flowers
02 Regression of sea shell ag
03 Simple network optimizatio
04 Model Update
02 sequential data
03 import models
results
processes
Templates
Operators
Process
Creating a simple neural network with one hidden layer and an output layer.
Iris is a multi-class classification problem, therefore the network loss is set to "multiclass cross entropy".
Retrieve Iris
Loading Data
Split Data
Split data into train and test set
Deep Learning
Open the Deep Learning operator by double-clicking on it, to discovere the layer setup.
Apply Model
Performance
Calculate model performance

1. Non-sequential data - 01 Classification of iris flowers

Close

[Open 01 Classification of iris flowers]

A first process introducing classification of tabular data with a fully-connected neural network. This process is a good introduction coming from a classification using another machine learning model using the well known iris data set with characteristics of three different flower types.

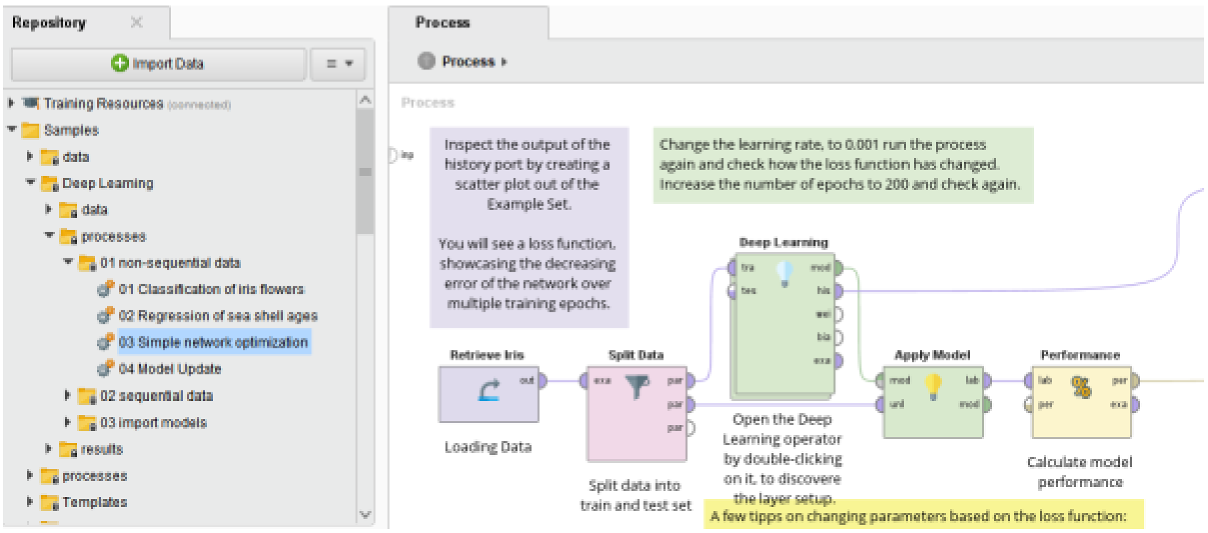

Repository
Import Data
Process
Training Resources (connected)
Samples
data
Deep Learning
data
processes
01 non-sequential data
01 Classification of iris flowers
02 Regression of sea shell ages
03 Simple network optimization
04 Model Update
02 sequential data
03 import models
results
processes
Performing a regression on data about sea shells.
Double click this operator to learn more about often needed preparations.
For regression a different loss function than for classification has to be used.
Retrieve Abalone
Loading data
Subprocess
Deep Learning
Open the Deep
Apply Model
Performance
Calculate model performance

2. Non-sequential data - 02 Regression of sea shell ages

Close ^

02 Regression of sea shell ages

This process is very similar to the “Classification of iris flowers” and differs only in the main objective, to regress numeric values instead of classifying types. The process uses a data set about characteristics of sea shells. For some time sea shells had to be killed in order to estimate their age. Moving over to modelling their age based on other characteristics prohibits this. This process explores sea shell characteristics and highlights what changes to a neural network need to be done in order to move from classification to regression.



Inspect the output of the history port by creating a scatter plot out of the Example Set.

You will see a loss function, showcasing the decreasing error of the network over multiple training epochs.

Change the learning rate, to 0.001 run the process again and check how the loss function has changed. Increase the number of epochs to 200 and check again.

Open the Deep Learning operator by double-clicking on it, to discovere the layer setup.

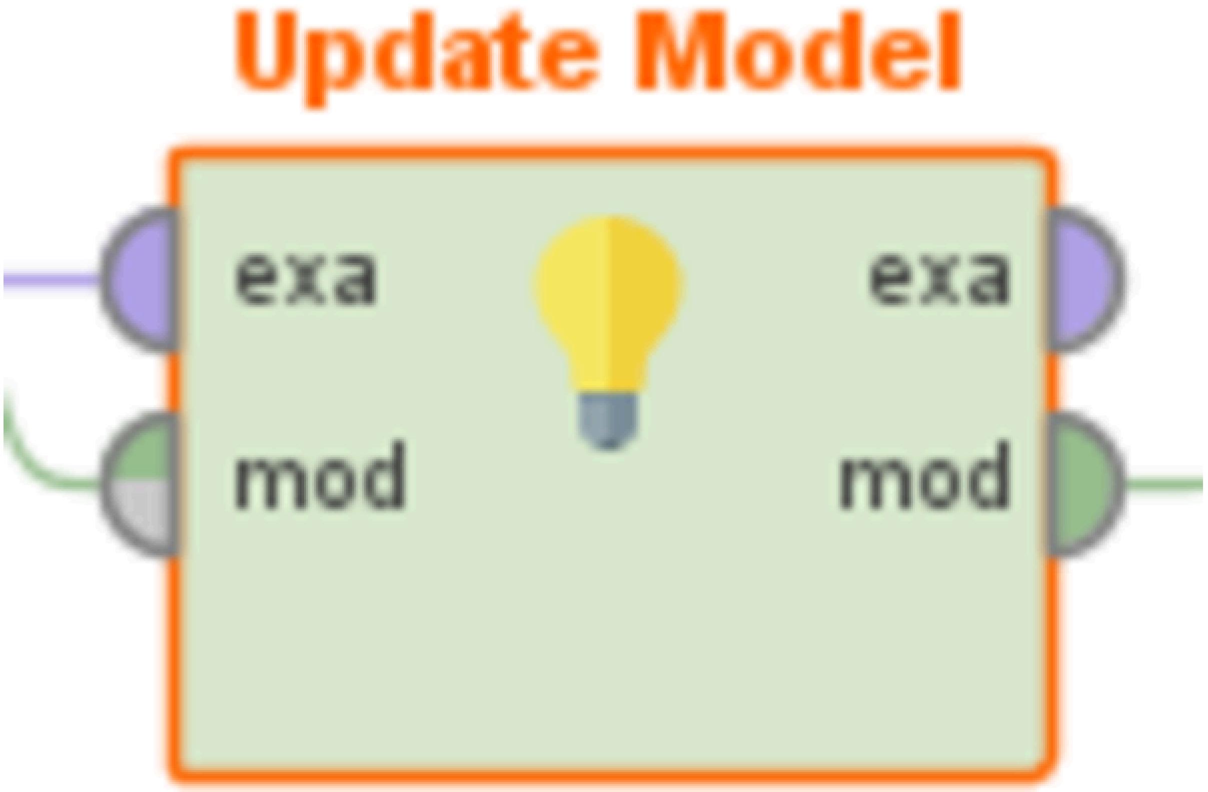
A few tipps on changing parameters based on the loss function:

3. Non-sequential data - 03 Simple network optimization

Close ^

03 Simple network optimization

This process continues the analysis from “Classification of iris flowers” and introduces ways of optimizing a network. Checking out the training history and learning how to tune operators based on its behavior is explained in this process.

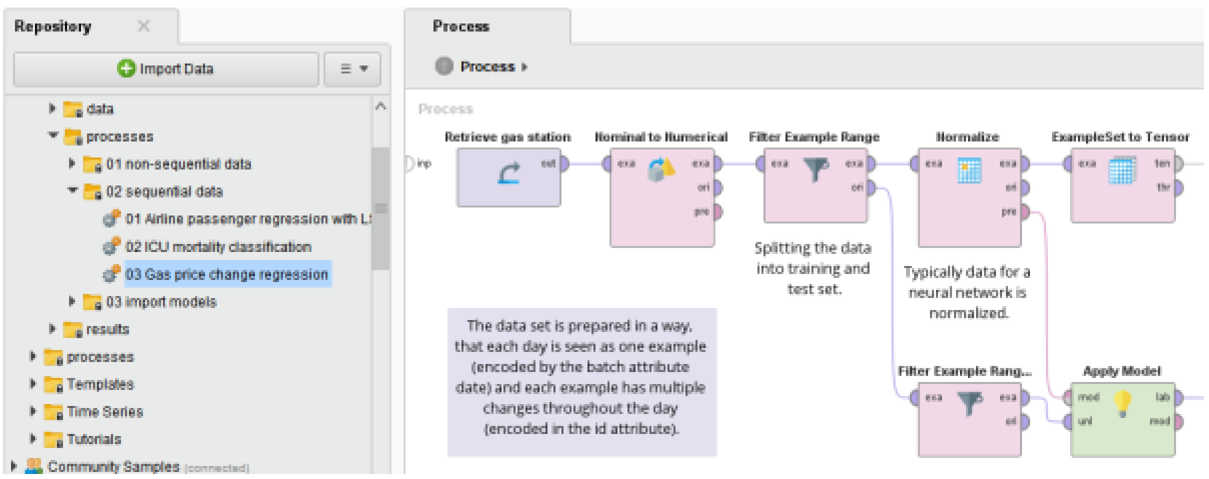


4. Non-sequential data - 04 Model Update

Close ^

04 Model Update

Within this example it is shown, how to take an initially trained model and slightly update it on newly added data, without having to train it completely anew. Deep Learning models are updatable models, in that sense, that we can continue training a model, since in its very nature neural networks are trained in an iterative fashion.



Splitting the data into training and test set.

Typically data for a neural network is normalized.

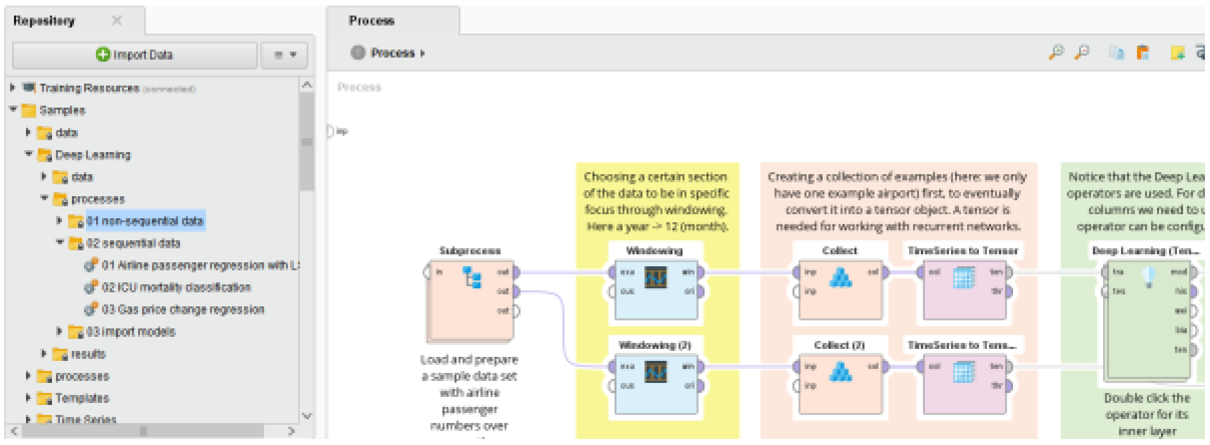
The data set is prepared in a way, that each day is seen as one example (encoded by the batch attribute date) and each example has multiple changes throughout the day (encoded in the id attribute).

5. Sequential data - 01 Gas price change regression

Close ^

01 Gas price change regression

In this example we are exploring a prediction use-case of modelling the up-coming gas price based on historic data of a local gas station. This process not only show's how to perform many-to-one regression using LSTMs, but also how to prepare sequential data for this task, without relying on collections. The shown method using the "ExampleSet to Tensor" operator is the recommended method, when obtaining your data from a data base, or other means, where you are facing a long data sets that contains identifiers for both the time-steps and the grouping of samples you want to classify, e.g. days of gas prices.



Load and prepare a sample data set with airline passenger numbers over

Choosing a certain section of the data to be in specific focus through windowing. Here a year -> 12 (month).

Creating a collection of examples (here: we only have one example airport) first, to eventually convert it into a tensor object. A tensor is needed for working with recurrent networks.

Notice that the Deep Lea operators are used. For d columns we need to u operator can be config

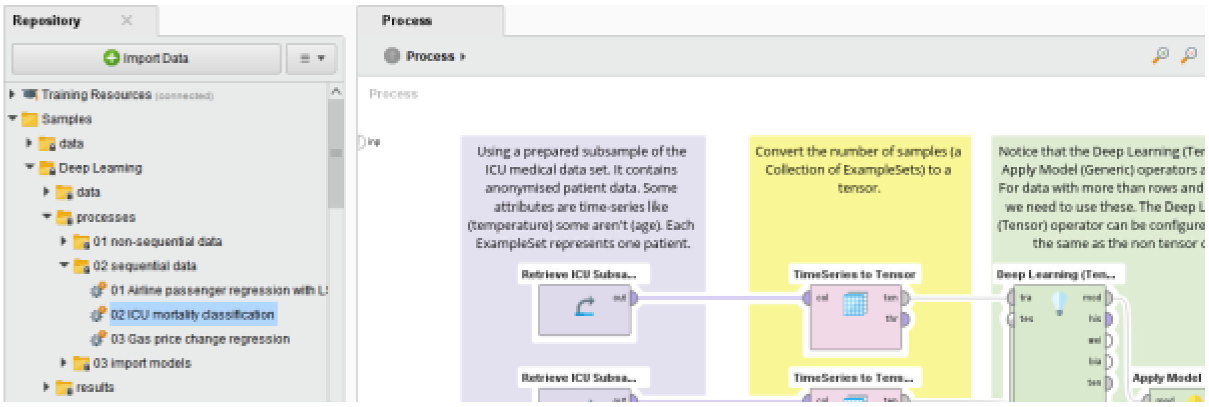
Double click the operator for its inner layer

6. Sequential data - 02 Airline passenger regression with LSTM

Close ^

02 Airline passenger regression with LSTM

Previous sample processes have covered handling of non-sequential data sets. Meaning, that one example is independent from the others. In this example, we're now having a look at sequential data sets, where examples are depending on each other. Here in a timely manner.
The example uses the development of airline passenger numbers over time and explores how this type of data can be prepared for being used with a neural network, that is using recurrent layers, that are capable of dealing with sequences and connection between various time-steps.
Currently we support two ways of preparing sequential data for neural networks. This process highlights how collections of examples can be used. Which might be a helpful mechanism, when creating the time sequences on the fly. Explore the "Gas price change regression" for an example more suited for data coming from data bases.



Using a prepared subsample of the ICU medical data set. It contains anonymised patient data. Some attributes are time-series like (temperature) some aren't (age). Each ExampleSet represents one patient.

Convert the number of samples (a Collection of ExampleSets) to a tensor.

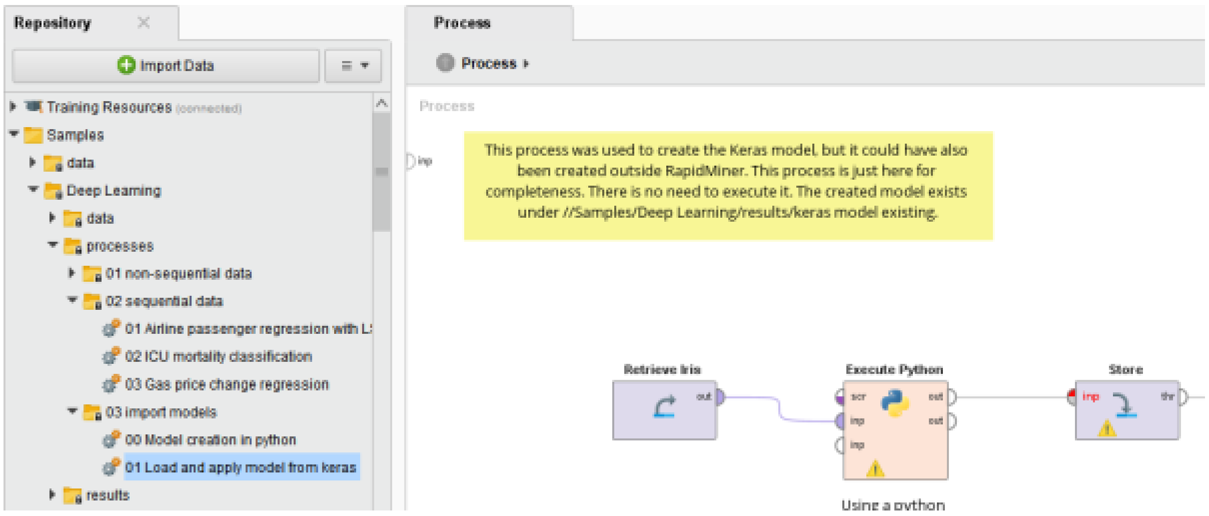
Notice that the Deep Learning (Ter Apply Model (Generic) operators a For data with more than rows and we need to use these. The Deep L (Tensor) operator can be configure the same as the non tensor c

Close ^

7. Sequential data - 03 ICU mortality classification

03 ICU mortality classification

This process is very similar to "Airline passenger regression with LSTM" but showcases how to handle classification tasks, when dealing with sequential data. The example shown performs a many-to-one classification, where for many time-steps the model should predict one class.



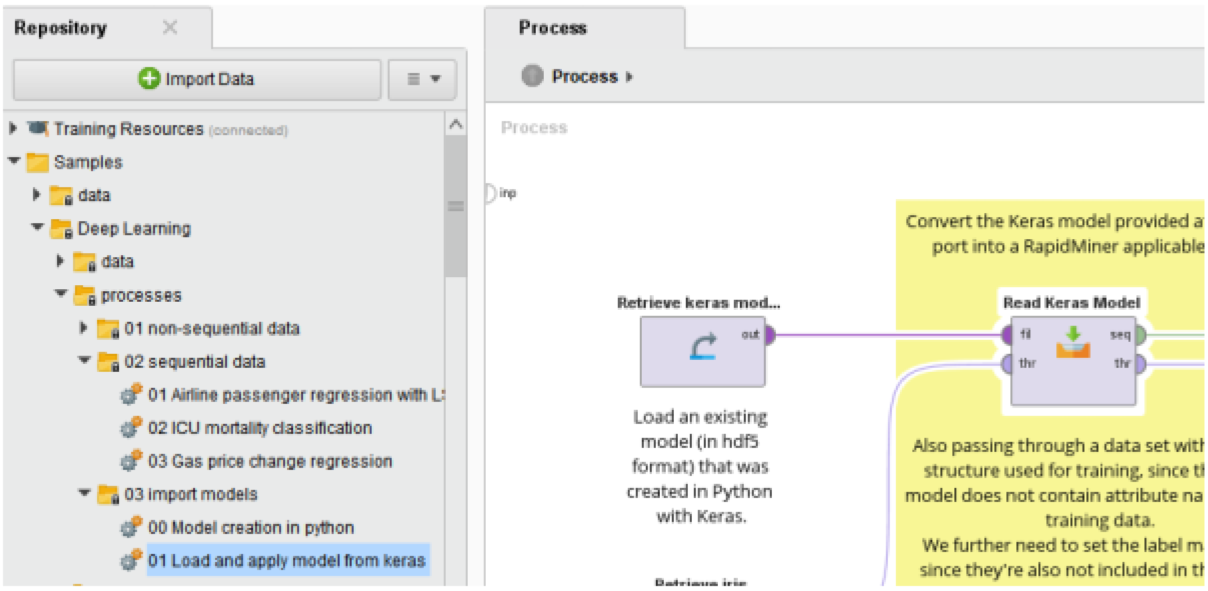
This process was used to create the Keras model, but it could have also been created outside RapidMiner. This process is just here for completeness. There is no need to execute it. The created model exists under //Samples/Deep Learning/results/keras model existing.

Close ^

8. Import models - 00 Model creation in python

00 Model creation in python

This process uses the python extension and relies on having a python environment and the Keras library installed. It shows how a model can be trained in python and stored for later use in RapidMiner.


Repository
Import Data
Training Resources (connected)
Samples
data
Deep Learning
data
processes
01 non-sequential data
02 sequential data
01 Airline passenger regression with L
02 ICU mortality classification
03 Gas price change regression
03 import models
00 Model creation in python
01 Load and apply model from keras
Process
Process
inp
Retrieve keras mod...
out
Read Keras Model
fil
thr
seq
thr

Convert the Keras model provided a
port into a RapidMiner applicable

Load an existing model (in hdf5 format) that was created in Python with Keras.

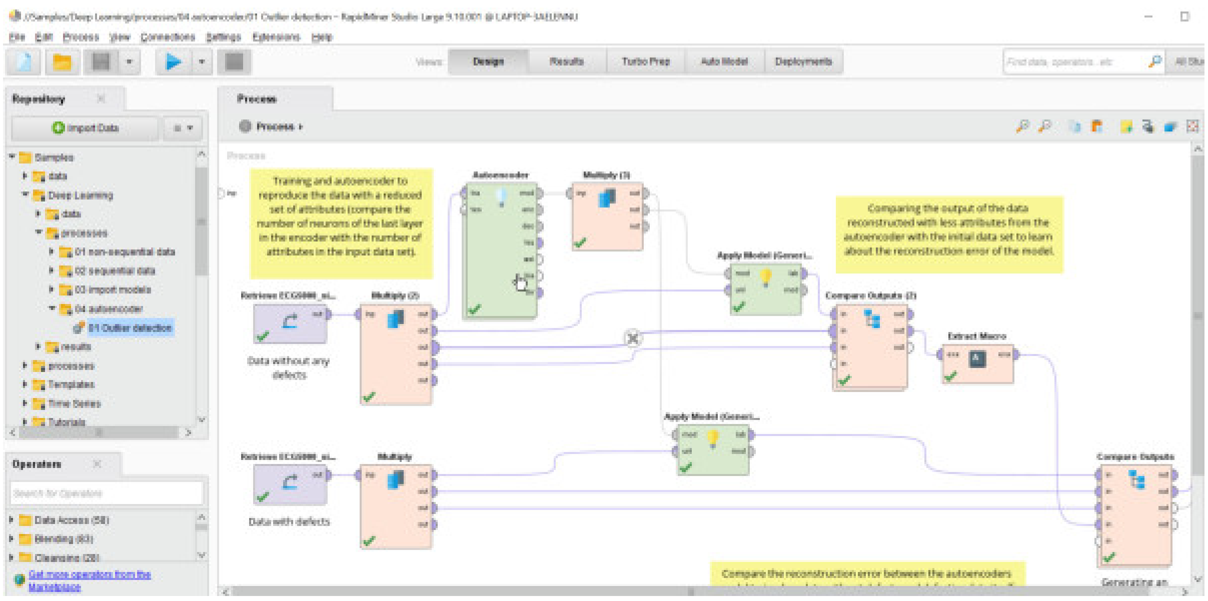
Also passing through a data set wit
structure used for training, since t
model does not contain attribute na
training data.
We further need to set the label m
since they're also not included in t

9. Import models - 01 Load and apply model from keras

Close ^

01 Load and apply model from keras

This process shows how a model created with the Keras library in python and stored in an HDF5 file, can be loaded into RapidMiner Studio and used to make predictions on ExampleSets. The read-in model can be applied using RapidMiner without having to have knowledge about Python or even having it installed. Furthermore the process shows how to use a sample data sets to adjust the Keras model, so that in case of classifications the predictions are already reflecting the actual class value and not just some number representing them.



Training and autoencoder to reproduce the data with a reduced set of attributes (compare the number of neurons of the last layer in the encoder with the number of attributes in the input data set).

Comparing the output of the data reconstructed with less attributes from the autoencoder with the initial data set to learn about the reconstruction error of the model.

Compare the reconstruction error between the autoencoders

10. autoencoder - 01 Outlier detection

Close ^

01 Outlier detection

This process shows use of the autoencoder operator on data that is not explicitly labeled, but is all defect free. By applying the autoencoder model to new unlabeled data, the reconstruction error is used to generate an anomaly score that can be used to detect defects.