University of Minho
School of Engineering

ISLab

Machine Learning and Decision-Making

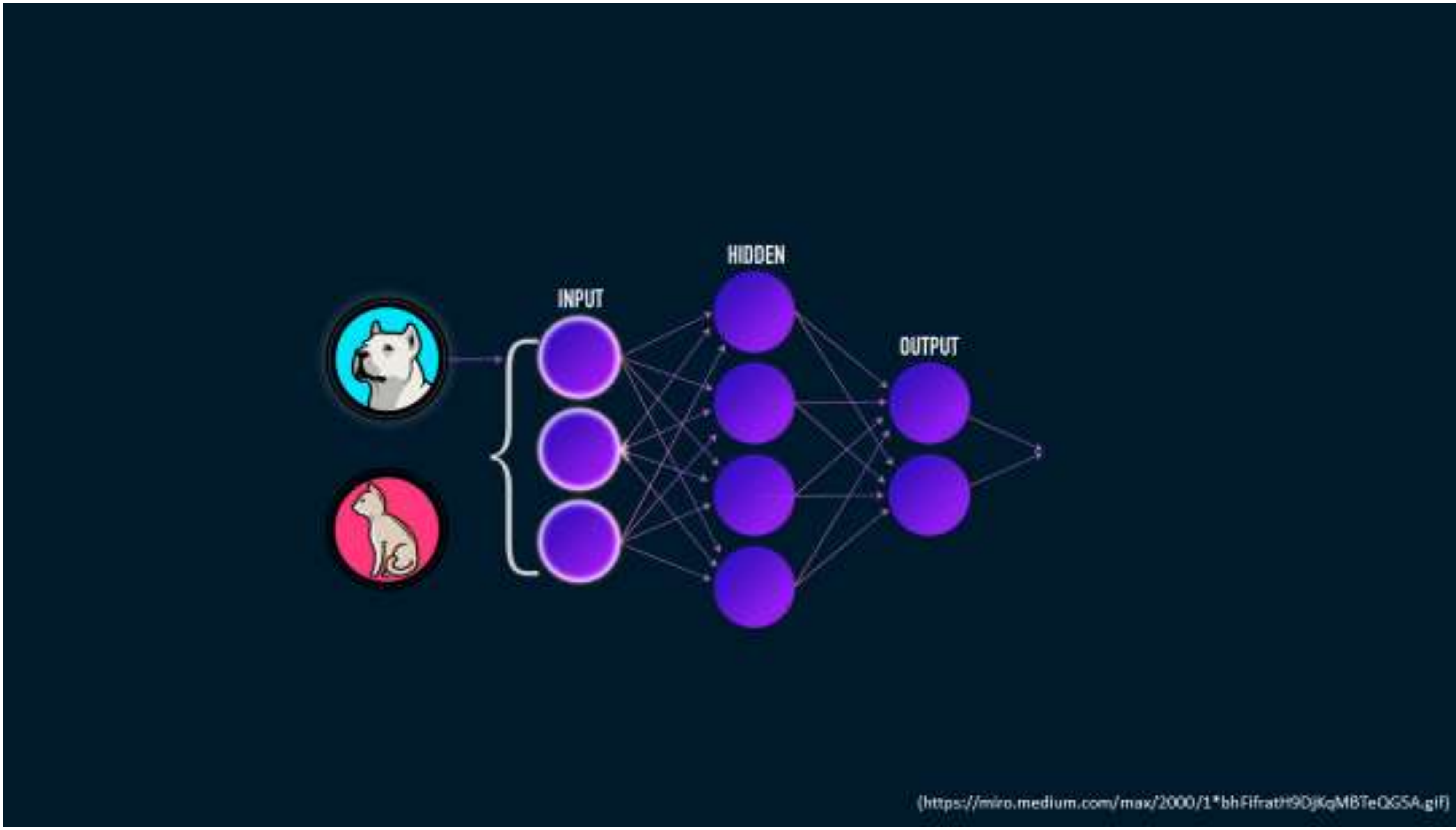
ADI @ LEI/3º, MiEI/4º - 2º Semestre

Filipe Gonçalves, Inês Alves, Cesar Analide

Part IX – April 2022

Contents

Contents


INPUT
HIDDEN
OUTPUT
(https://miro.medium.com/max/2000/1*bhFifratH9DjKqMBTeQG5A.gif)

What about Artificial Neural Networks?

We have already used several learning model techniques... But now let's try using Multilayer Perceptrons (MLPs), a class of Artificial Neural Networks (ANN)!

Artificial Neural Networks are a computational model that consists of several processing elements that receive inputs and deliver outputs based on their predefined activation functions.

Artificial Neural Networks can be applied both for Regression and Classification problems.

To implement our first Artificial Neural Network we will use:

Open for Innovation
KNIME

KNIME for Artificial Neural Networks

Why?

Open for Innovation KNIME

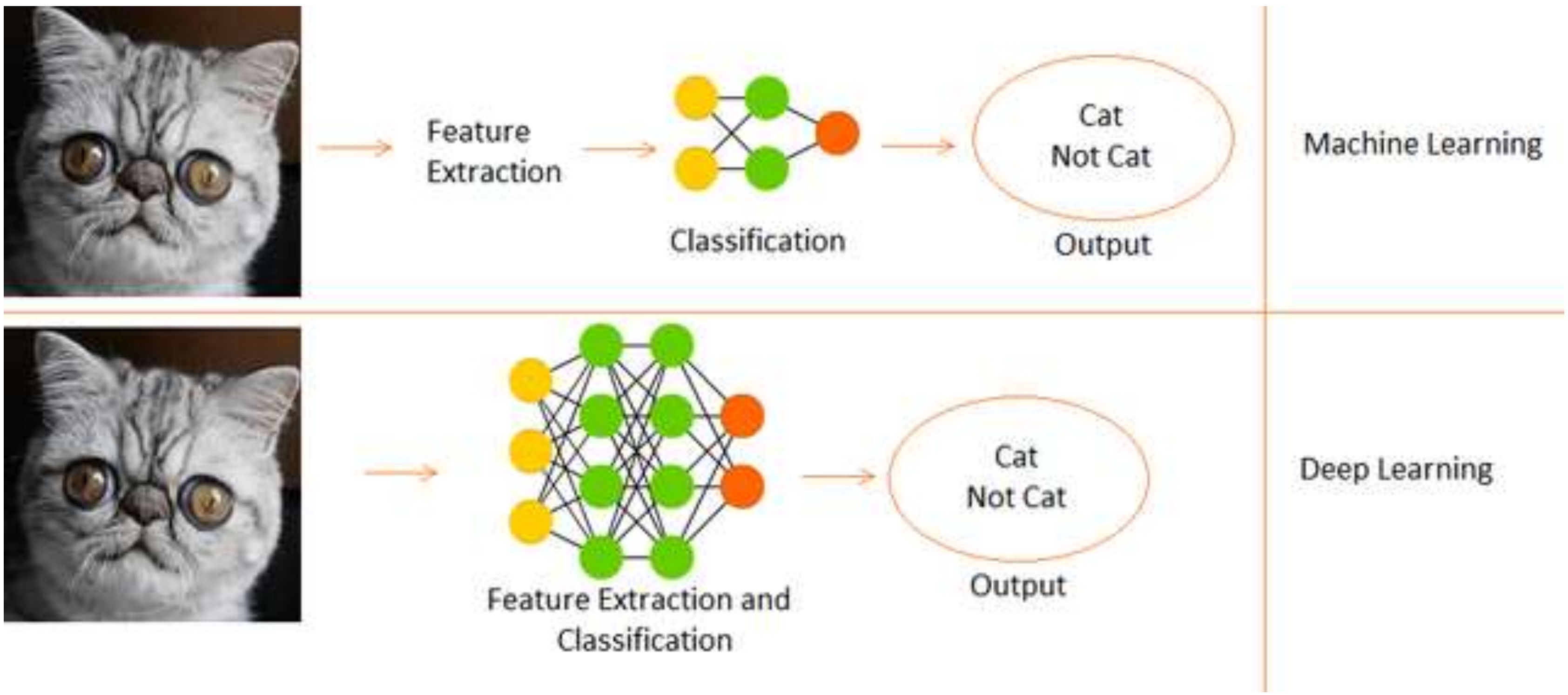
Open-source platform applied for data science

Strong and comprehensive platform for drag-and-drop analytics, machine & deep learning, statistics, and ETL

Tool of choice for data science starters

Also, no programming background required

Machine Learning vs Deep Learning



ANN on Classification Problem

Let's consider the development & testing of a learning model solution for a binary classification problem – classify as Moon or not Moon given its parameters

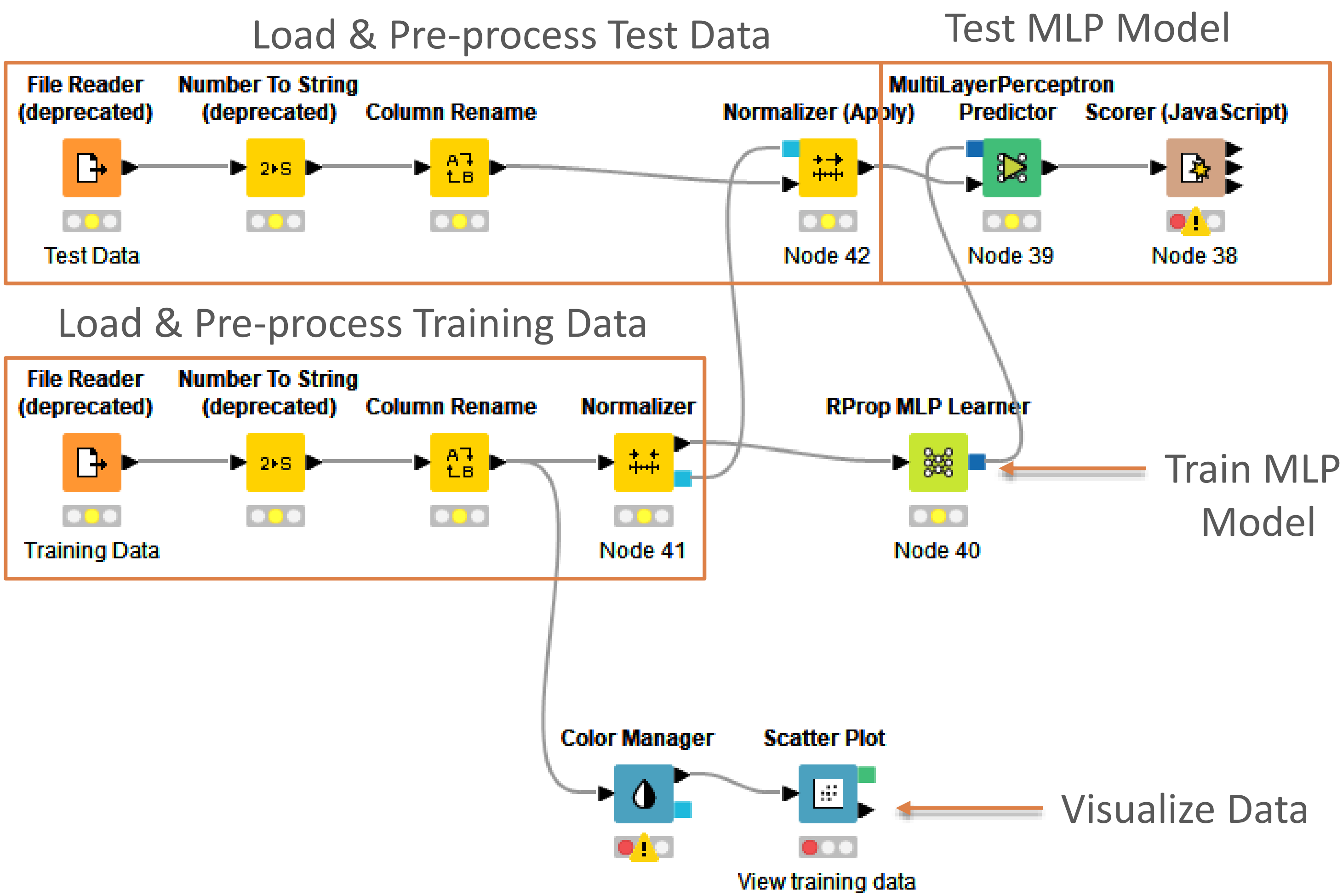
The proposed workflow shows how to create a Multilayer Perceptron with a softmax layer for classification

In this example the MLP is used to classify a simple dataset with two features

Dataset available:

- Training Data: https://bit.ly/36NBOxo
- Test Data: https://bit.ly/3vcAuxd

ANN Workflow Pipeline



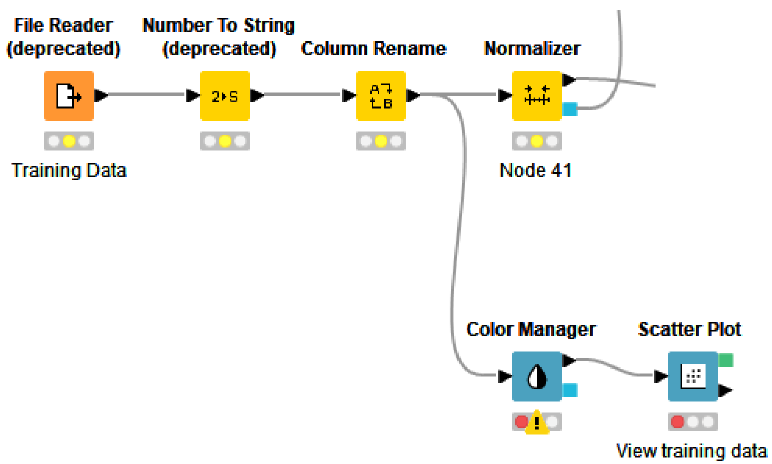
Load & Visualize Training Data

Start by loading and preparing Training Data:

- Col0: binary numeric class (convert to String)
- Col1, Col2: normalize double features

Visualize data distribution per class:

- Class 0: blue
- Class 1: red



File Table - 3:1 - File Reader (deprecated) (Training Data)

File Edit Hilite Navigation View

Table "moon_data_train.csv" - Rows: 2000 | Spec - Columns: 3 | Properties | Flow Variables

Row ID	I Col0	D Col1	D Col2
Row0	1	0.611	-0.46
Row1	1	-0.556	0.731
Row2	0	-0.585	0.663
Row3	1	1.198	-0.695
Row4	0	-0.555	0.797
Row5	0	-0.524	0.883
Row6	1	0.624	-0.782
Row7	0	0.06	0.911
Row8	0	0.443	0.648
Row9	0	0.629	0.893
Row10	0	-0.794	0.244

Normalized table - 0:41 - Normalizer

File Edit Hilite Navigation View

Table "default" - Rows: 2000 | Spec - Columns: 3 | Properties | Flow Variables

Row ID	S Label	D Feature1	D Feature2
Row0	1	0.525	0.226
Row1	1	0.236	0.679
Row2	0	0.229	0.654
Row3	1	0.67	0.137
Row4	0	0.237	0.704
Row5	0	0.244	0.737
Row6	1	0.528	0.104
Row7	0	0.389	0.748
Row8	0	0.483	0.648
Row9	0	0.529	0.741
Row10	0	0.177	0.494

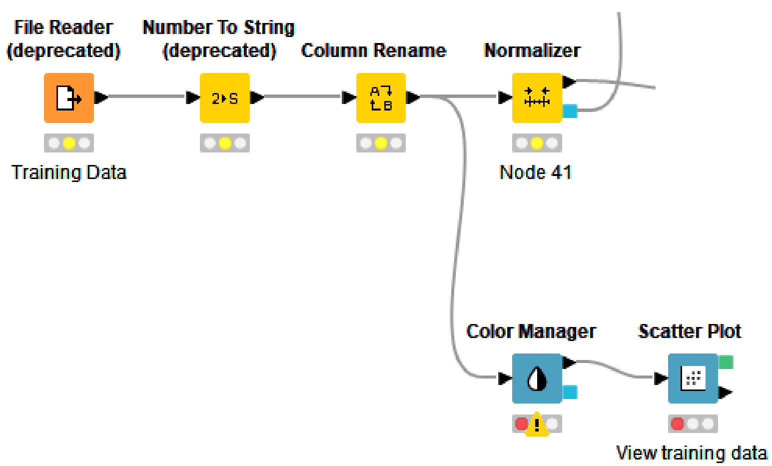
Load & Visualize Training Data

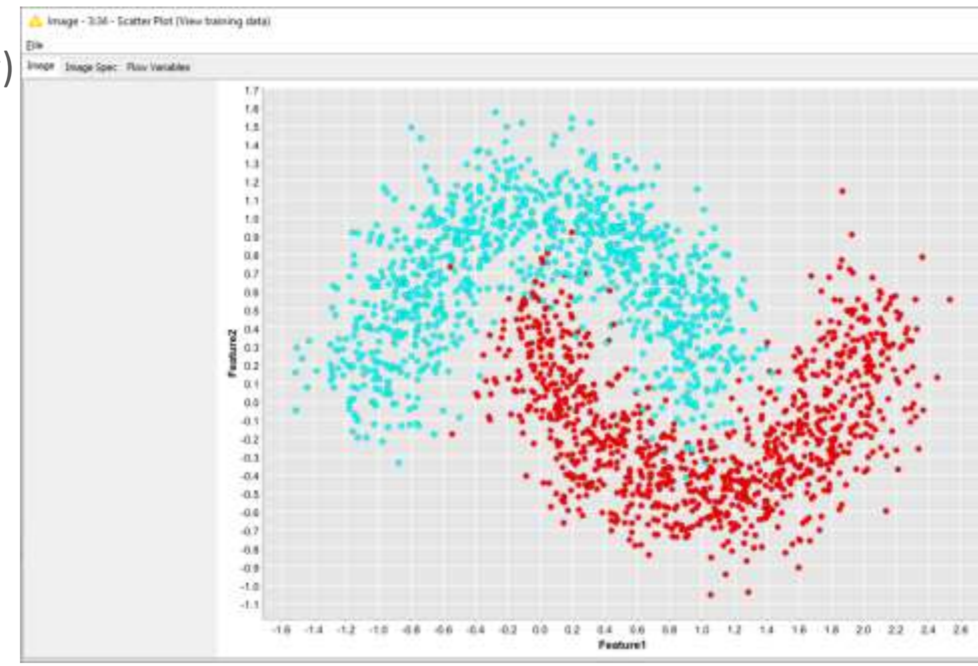
Start by loading and preparing Training Data:

- Col0: binary numeric class (convert to String)
- Col1, Col2: normalize double features

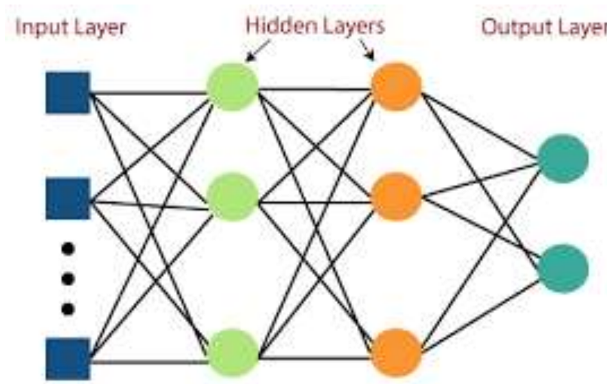
Visualize data distribution per class:

- Class 0: blue
- Class 1: red





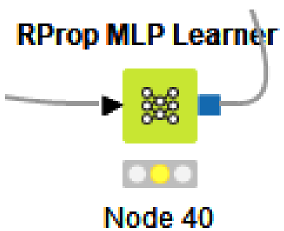
Build & Train MLP

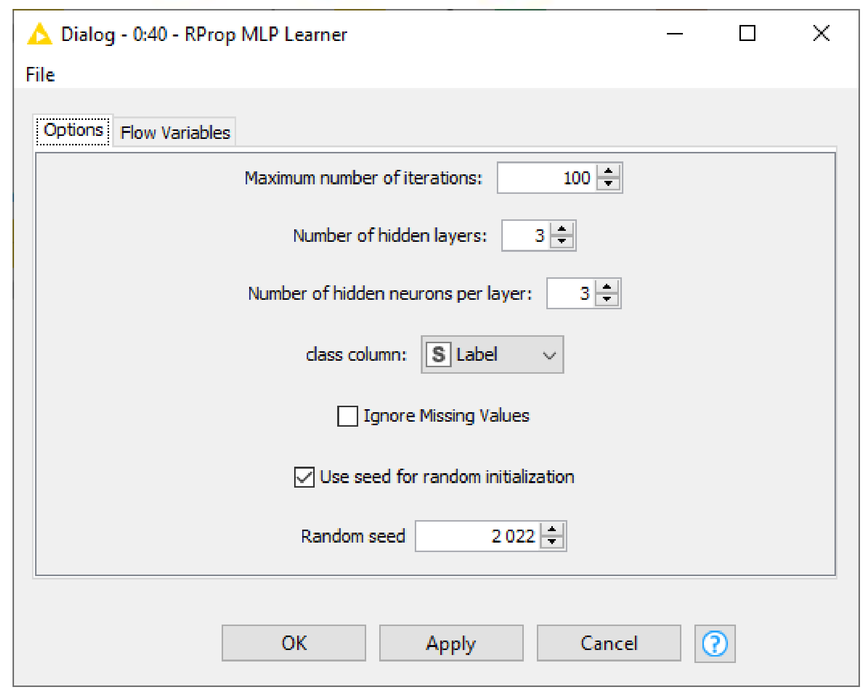


Create a simple general purpose Multilayer Perceptron consisting of three fully connected (FC) layers

Parameters:

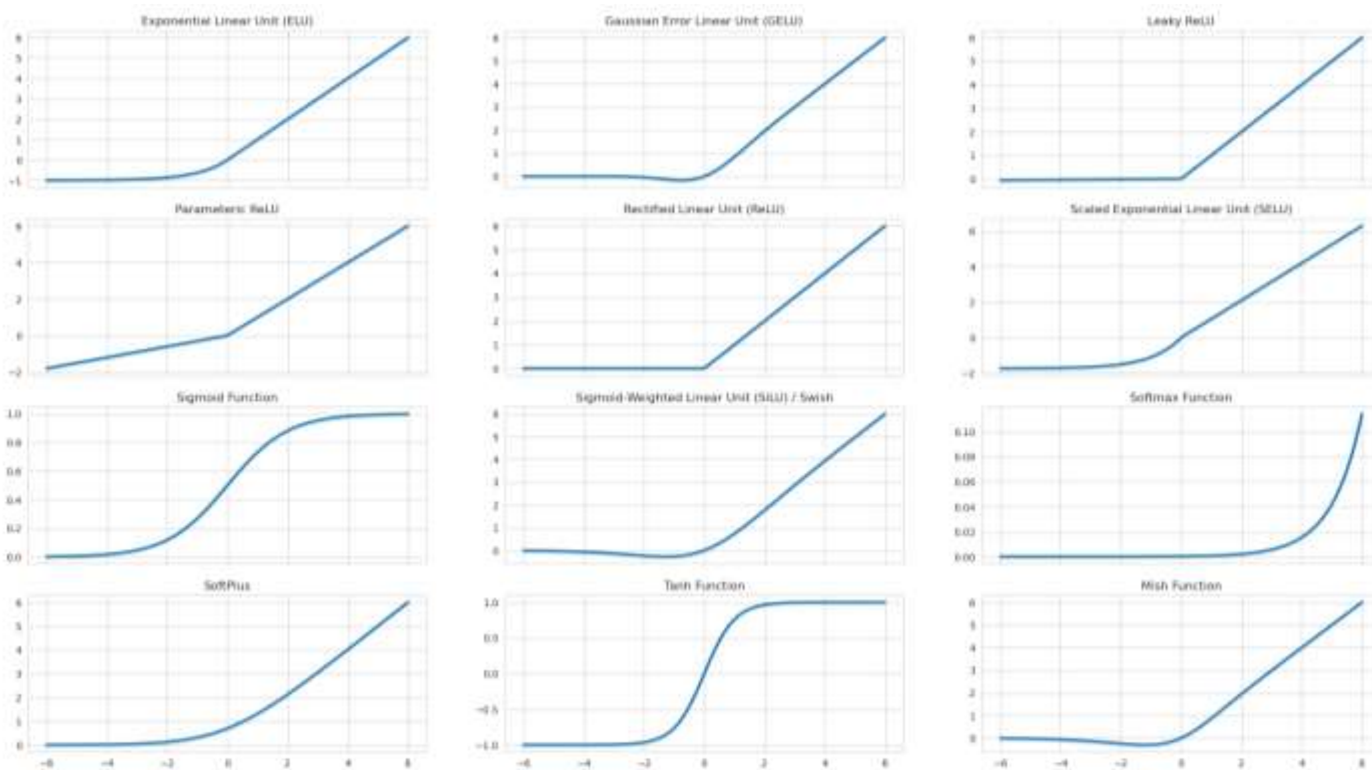
- Number of Iterations: 100
- Number of hidden layers: 3
- Number of hidden neurons per layer: 3
- Class Column: Label
- Random seed: 2022

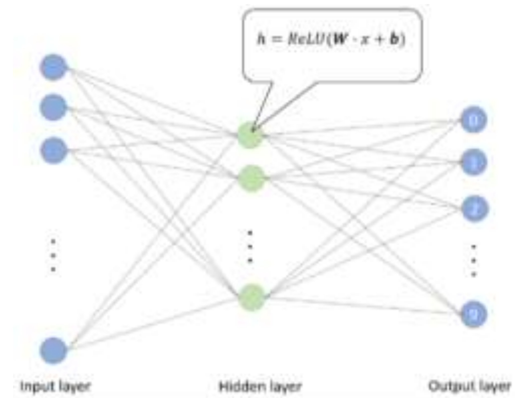


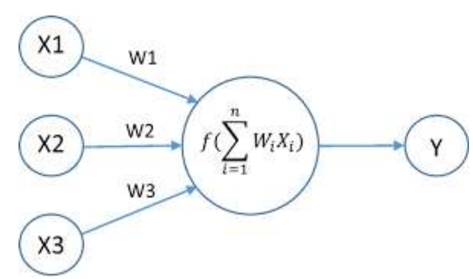


Build & Train MLP

Activation Functions







Load Test Data

Load and prepare Test Data (according to training Data):

- Col0: binary numeric class (convert to String)
- Col1, Col2: normalize double features

Renamed/Retyped table - 3:36 - Column Rename

File Edit Hilite Navigation View

Table "default" - Rows: 1000 | Spec - Columns: 3 | Properties | Flow Variables

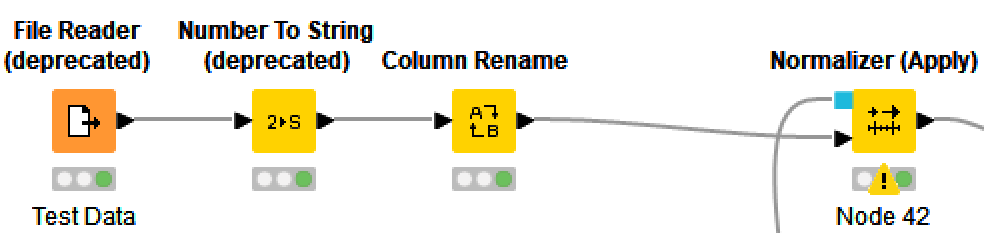
Row ID	S Label	D Feature1	D Feature2
Row0	0	-0.501	0.687
Row1	1	0.19	-0.341
Row2	0	0.995	0.663
Row3	0	-1.031	0.342
Row4	1	0.038	-0.837
Row5	0	-0.114	0.74
Row6	1	0.568	-0.376
Row7	1	0.029	0.066
Row8	1	1.971	0.274
Row9	0	0.709	0.241
Row10	1	0.37	-0.128

Normalized output - 0:42 - Normalizer (Apply)

File Edit Hilite Navigation View

Table "default" - Rows: 1000 | Spec - Columns: 3 | Properties | Flow Variables

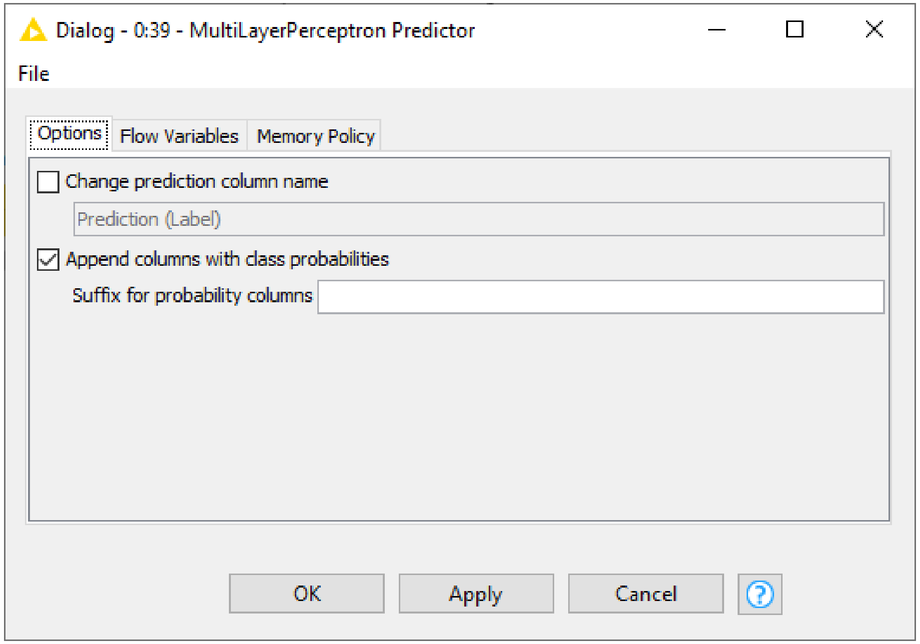
Row ID	S Label	D Feature1	D Feature2
Row0	0	0.25	0.663
Row1	1	0.421	0.271
Row2	0	0.62	0.653
Row3	0	0.119	0.531
Row4	1	0.383	0.083
Row5	0	0.346	0.683
Row6	1	0.514	0.258
Row7	1	0.381	0.426
Row8	1	0.861	0.505
Row9	0	0.549	0.493
Row10	1	0.465	0.352

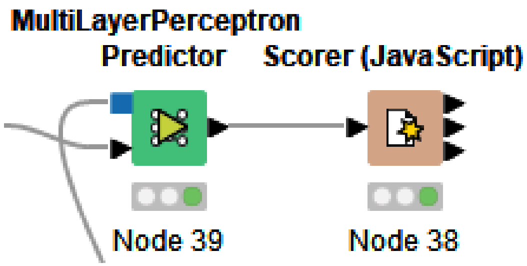


Predict & Evaluate MLP

After training is finished, we can use the MultiLayerPerceptron Predictor Node to create predictions

Apply the Scorer Node to evaluate the classification performance of the MLP model





Scorer View

Confusion Matrix

	0 (Predicted)	1 (Predicted)	
0 (Actual)	426	71	85.71%
1 (Actual)	52	451	89.66%
	89.12%	86.40%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
87.70%	12.30%	0.754	877	123

MLP Tips

Regarding Artificial Neural Networks, take into consideration the following good practices:

- ANN are picky - they prefer scaled data! Normalize whenever possible!
- Fine-tune the ANN parameters (e.g., number of iterations, number of hidden layers, number of hidden neurons per layer) using grid search methods
- Never forget to use a specific random seed (replicate learning model train/test)

Open for Innovation KNIME

Hands On

ANN on Classification Problem

Let’s consider the development & testing of a learning model solution for a binary classification problem – classify as Moon or not Moon given its parameters

The proposed workflow shows how to create a Multilayer Perceptron with a softmax layer for classification

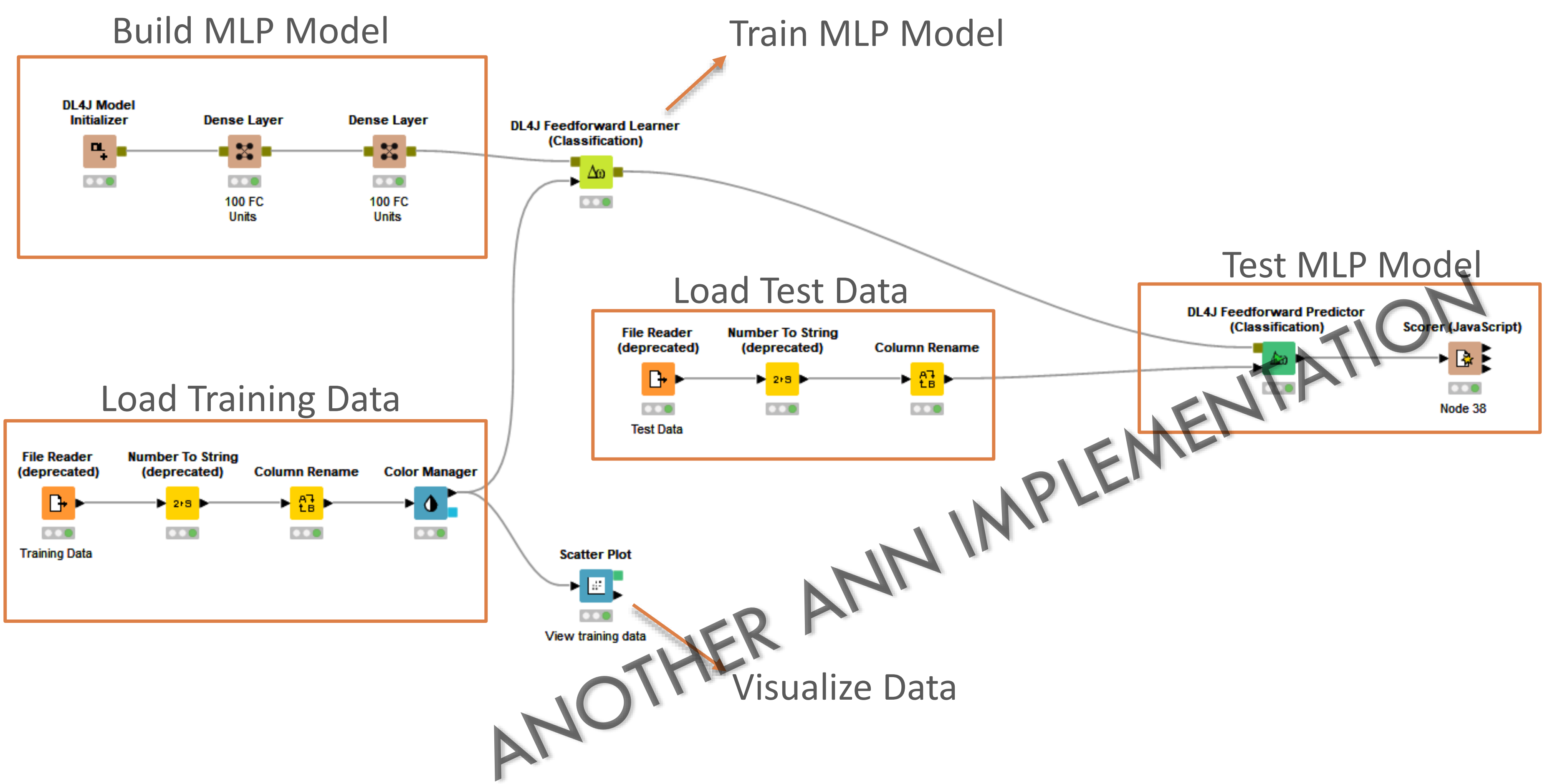
In this example the MLP is used to classify a simple dataset with two features

Dataset available:
- Training Data: knime://knime.workflow/moon_data_train.csv
- Test Data: knime://knime.workflow/moon_data_eval.csv

Workflow Requirements:
- KNIME Deeplearning4J Integration

ANOTHER ANN IMPLEMENTATION

ANN Workflow Pipeline



Load & Visualize Training Data

Start by loading and preparing Training Data:

- Col0: binary numeric class (convert to String)
- Col1, Col2: double features

Visualize data distribution per class:

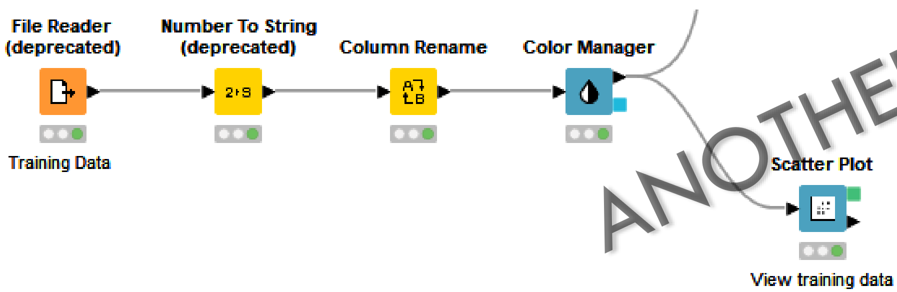
- Class 0: blue
- Class 1: red

File Table - 3:1 - File Reader (deprecated) (Training Data)

File Edit Hilite Navigation View

Table "moon_data_train.csv" - Rows: 2000 | Spec - Columns: 3 | Properties | Flow Variables

Row ID	I Col0	D Col1	D Col2
Row0	1	0.611	-0.46
Row1	1	-0.556	0.731
Row2	0	-0.585	0.663
Row3	1	1.198	-0.695
Row4	0	-0.555	0.797
Row5	0	-0.524	0.883
Row6	1	0.624	-0.782
Row7	0	0.06	0.911
Row8	0	0.443	0.648
Row9	0	0.629	0.893
Row10	0	-0.794	0.244



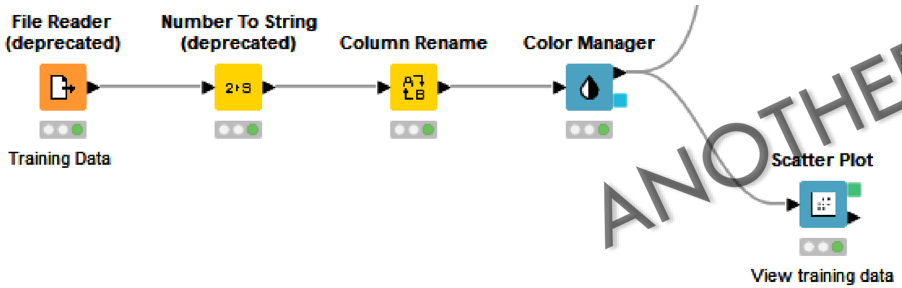
Load & Visualize Training Data

Start by loading and preparing Training Data:

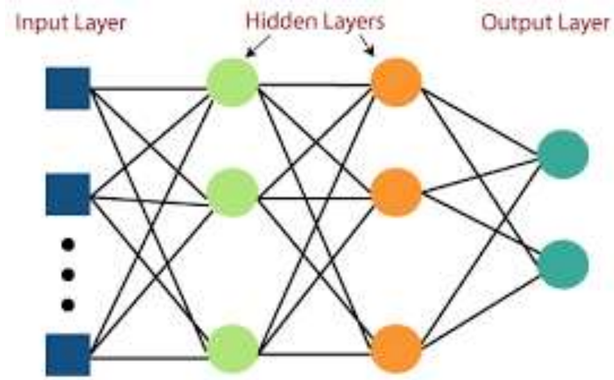
- Col0: binary numeric class (convert to String)
- Col1, Col2: double features

Visualize data distribution per class:

- Class 0: blue
- Class 1: red



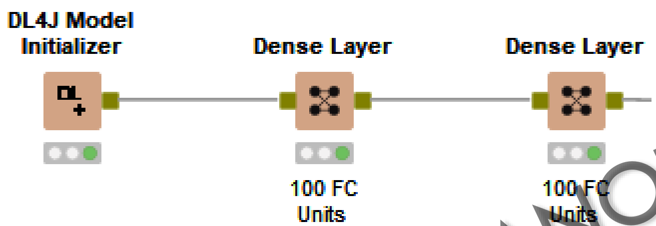
Build MLP

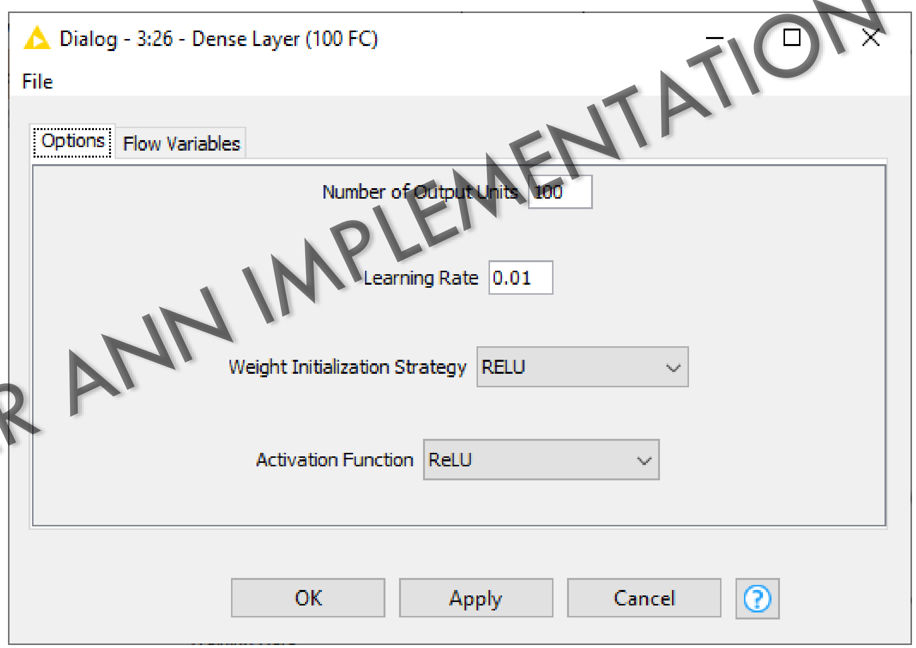


Create a simple general purpose Multilayer Perceptron consisting of two fully connected (FC) layers

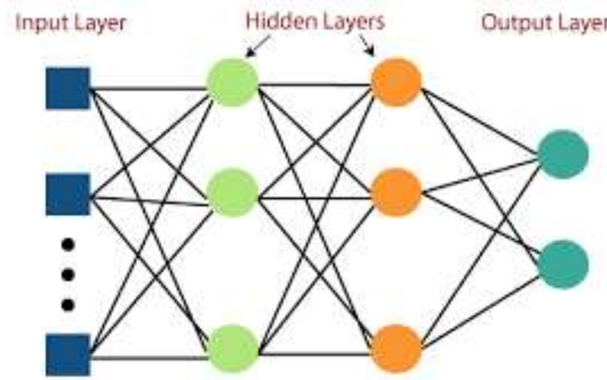
Parameters for each FC layers:

- Number of Units / Nodes = 100
- Learning Rate: 10^-2 = 0.01
- Activation Function: ReLU





Train MLP

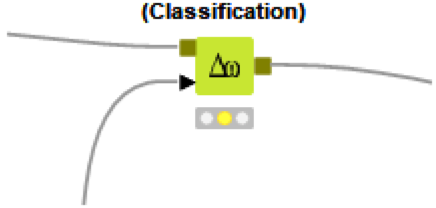


Train the created network using the DL4J Feedforward Learner (Classification) Node

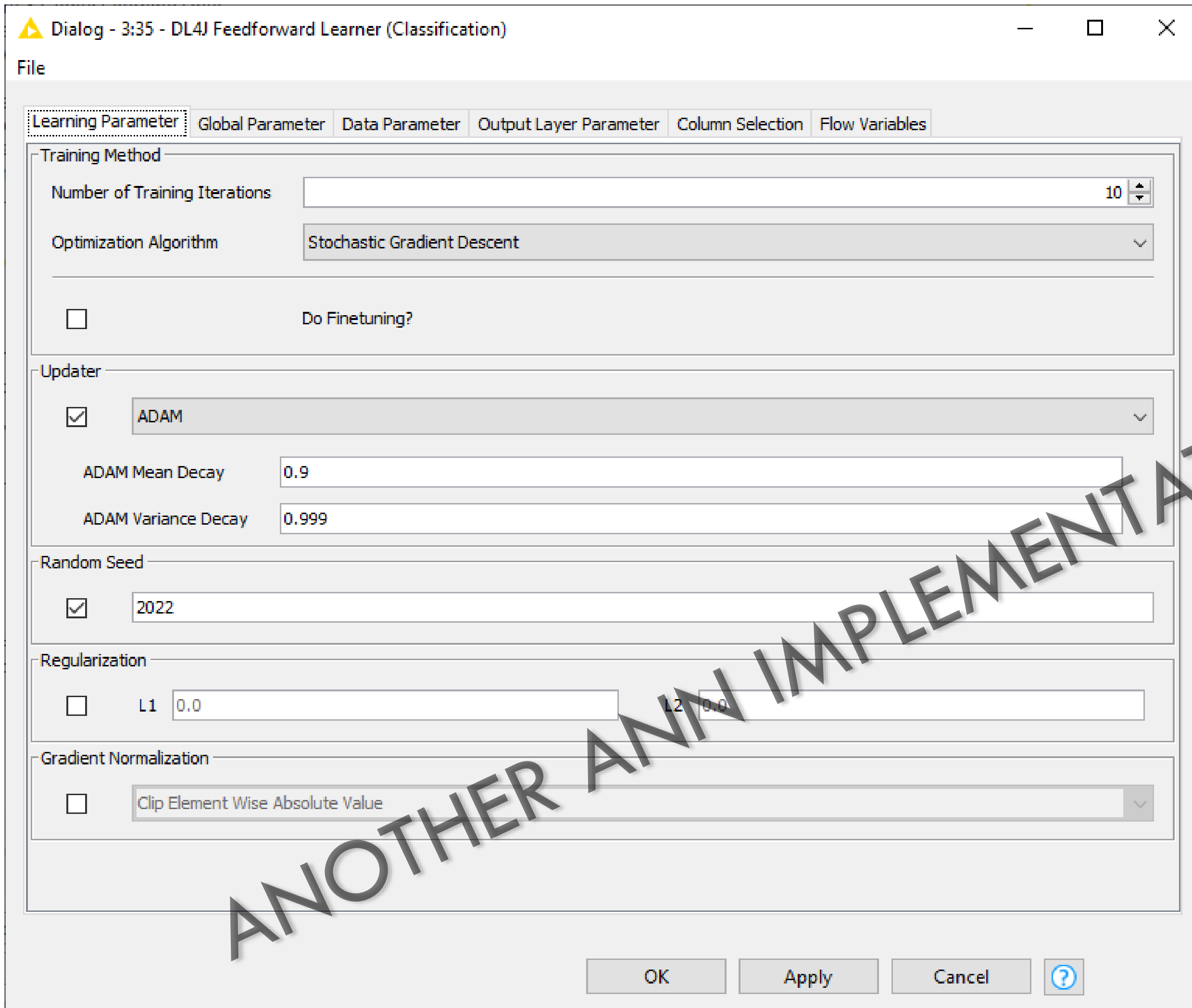
Take into consideration adjusting the learning parameters (e.g., Learning Parameter, Output Layer Parameters, Column Selection, etc.)

Important Parameters:

- Global Learning Rate (Global Parameters): configure the learner node to use the specified learning rate for all layers
- Output Layer Parameter: configure the loss function suitable for the learner node ('Negative Log Likelihood' paired with 'Softmax' is usually a good choice for classification)

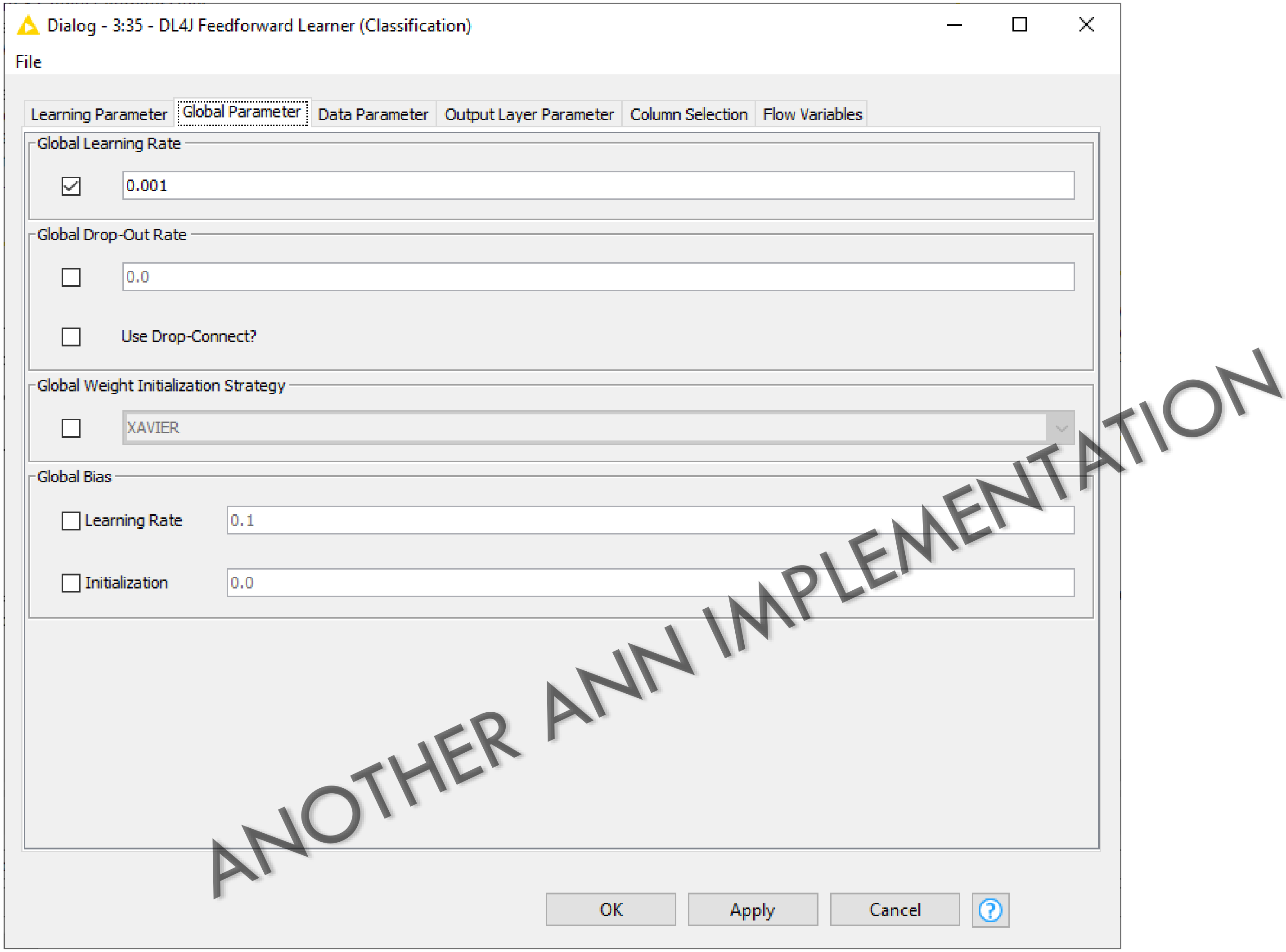


Train MLP

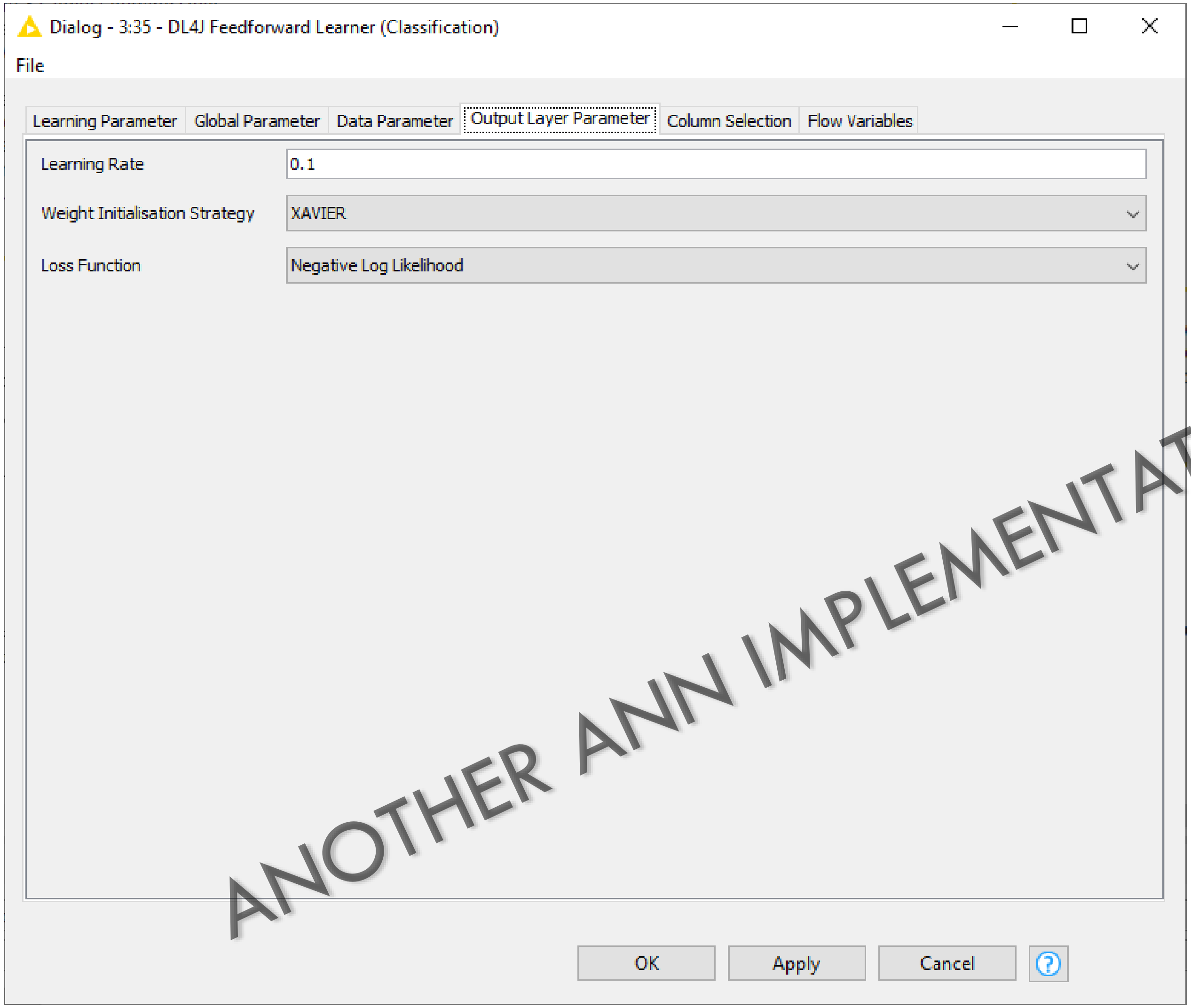


ANOTHER ANN IMPLEMENTATION

Train MLP

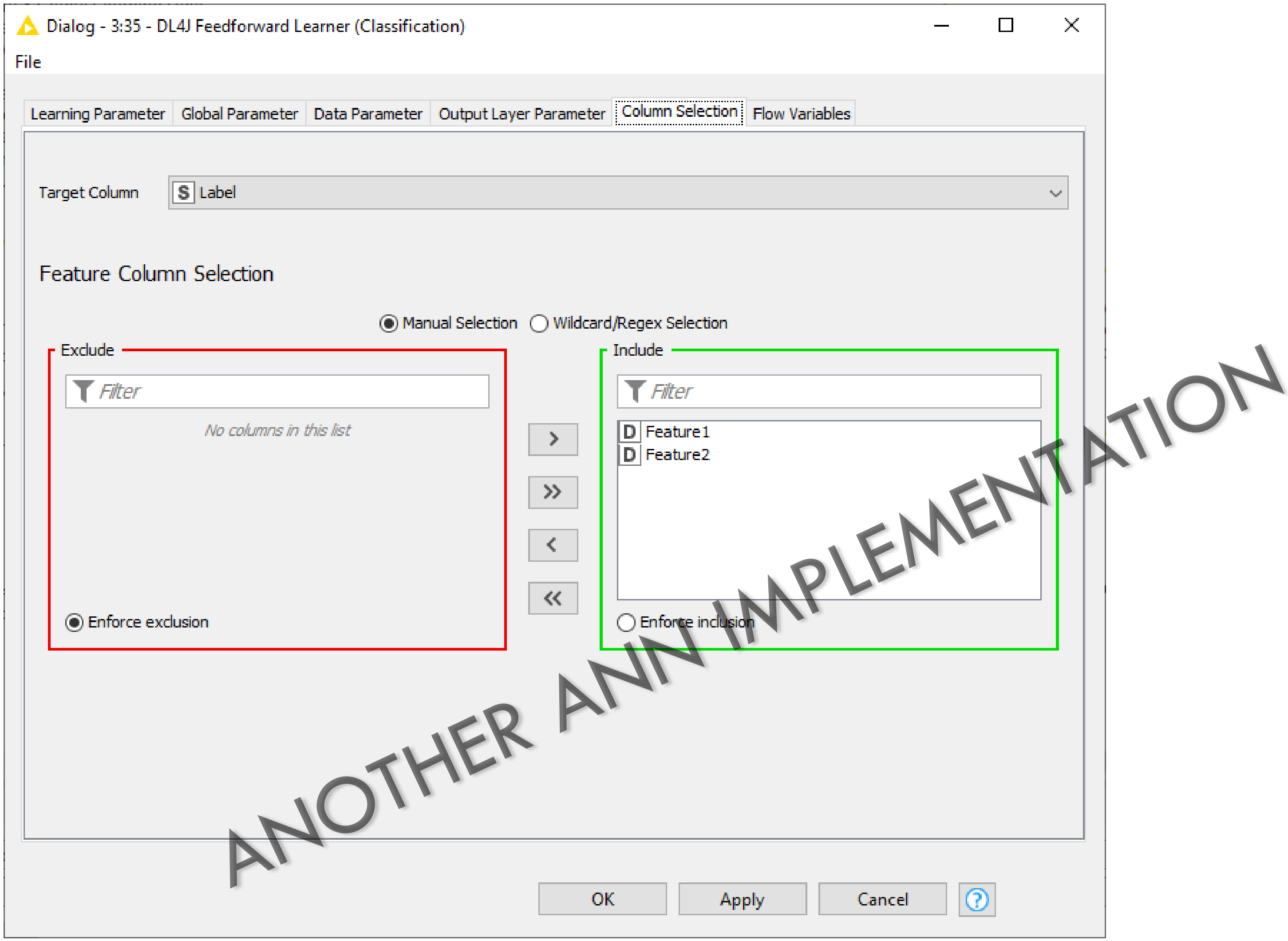

Dialog - 3:35 - DL4J Feedforward Learner (Classification)
File
Learning Parameter
Global Parameter
Data Parameter
Output Layer Parameter
Column Selection
Flow Variables
Global Learning Rate
0.001
Global Drop-Out Rate
0.0
Use Drop-Connect?
Global Weight Initialization Strategy
XAVIER
Global Bias
Learning Rate
0.1
Initialization
0.0
OK
Apply
Cancel
ANOTHER ANN IMPLEMENTATION

Train MLP



ANOTHER ANN IMPLEMENTATION

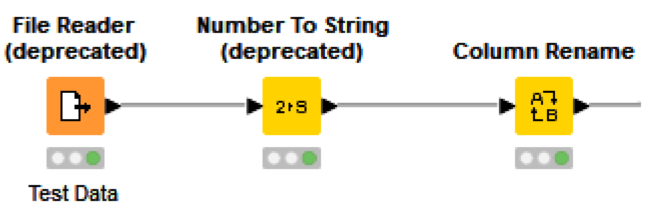
Train MLP


Dialog - 3:35 - DL4J Feedforward Learner (Classification)
File
Learning Parameter
Global Parameter
Data Parameter
Output Layer Parameter
Column Selection
Flow Variables
Target Column
S Label
Feature Column Selection
Manual Selection
Wildcard/Regex Selection
Exclude
Filter
No columns in this list
Enforce exclusion
Include
Filter
D Feature1
D Feature2
Enforce inclusion
OK
Apply
Cancel
ANOTHER ANN IMPLEMENTATION

Load Test Data

Load and prepare Test Data (according to training Data):

- Col0: binary numeric class (convert to String)
- Col1, Col2: double features



Renamed/Retyped table - 3:36 - Column Rename

File Edit Hilite Navigation View

Table "default" - Rows: 1000 | Spec - Columns: 3 | Properties | Flow Variables

Row ID	S Label	D Feature1	D Feature2
Row0	0	-0.501	0.687
Row1	1	0.19	-0.341
Row2	0	0.995	0.663
Row3	0	-1.031	0.342
Row4	1	0.038	-0.837
Row5	0	-0.114	0.74
Row6	1	0.568	-0.376
Row7	1	0.029	0.066
Row8	1	1.971	0.274
Row9	0	0.709	0.241
Row10	1	0.37	-0.128

Predict & Evaluate MLP

After training is finished, we can use the DL4J Feedforward Predictor (Classification) Node to create predictions

Apply the Scorer Node to evaluate the classification performance of the MLP model