

# How Machine Learning Techniques Could Help in Control Systems

-

## Workshop

Alessia Bozzini



**UNIVERSITÀ**  
**di VERONA**  
Dipartimento  
di **INFORMATICA**

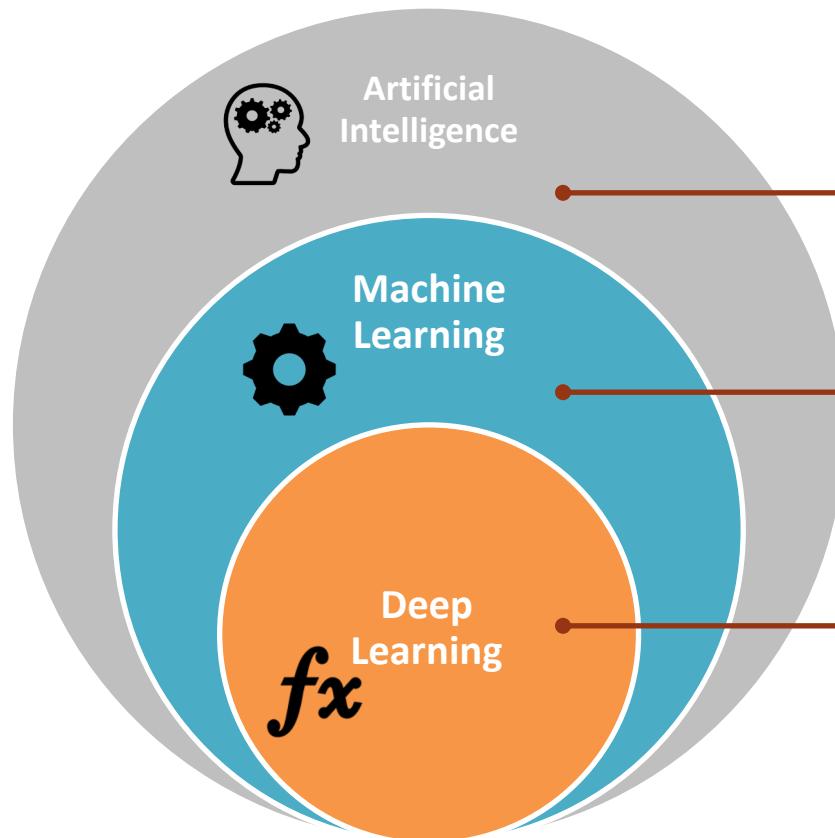
# Agenda

- Machine learning Supervised
- Control system
- Methodology
- MATLAB and Simulink
- Example

Machine learning Supervised  
Control system  
Methodology

## **BACKGROUND**

# Machine Learning



## Artificial Intelligence

Any technique which enables computers to mimic human behavior

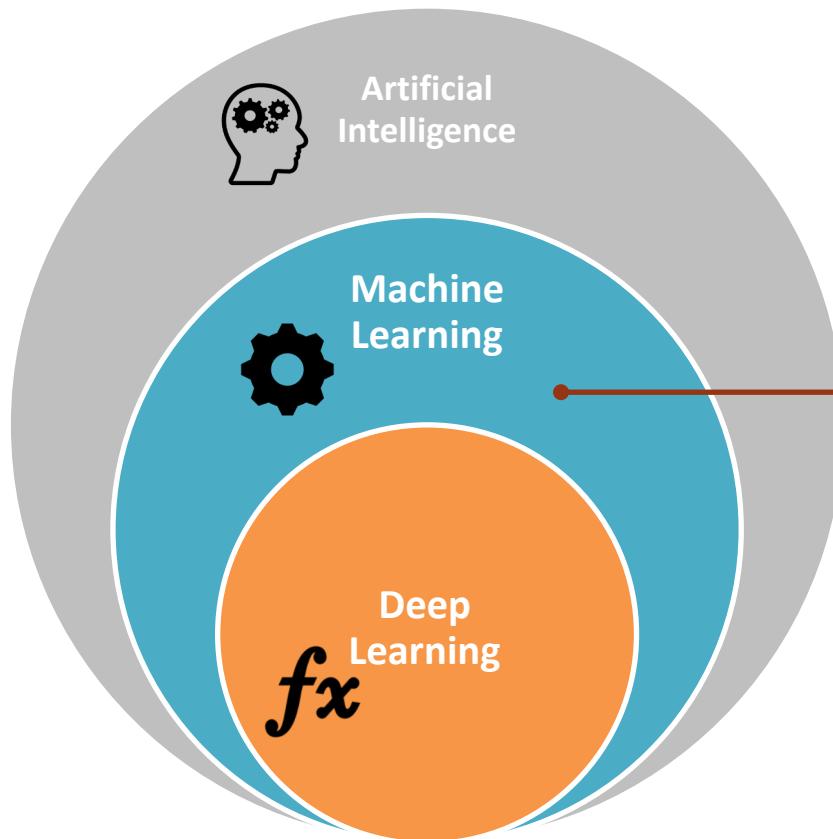
## Machine Learning

Subset of AI techniques which use statistical methods to enable machines to improve with experience

## Deep Learning

Subset of ML which make the computation of multi-layer neural networks feasible

# Machine Learning



## Machine Learning

### Supervised

- Example + solution

### Unsupervised

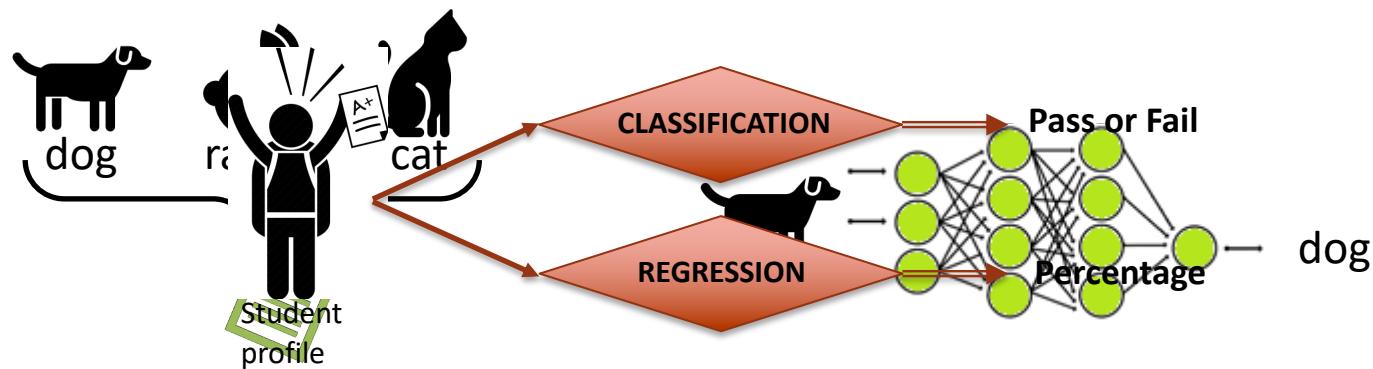
- Example without solution

### Reinforcement learning

- Example + goal

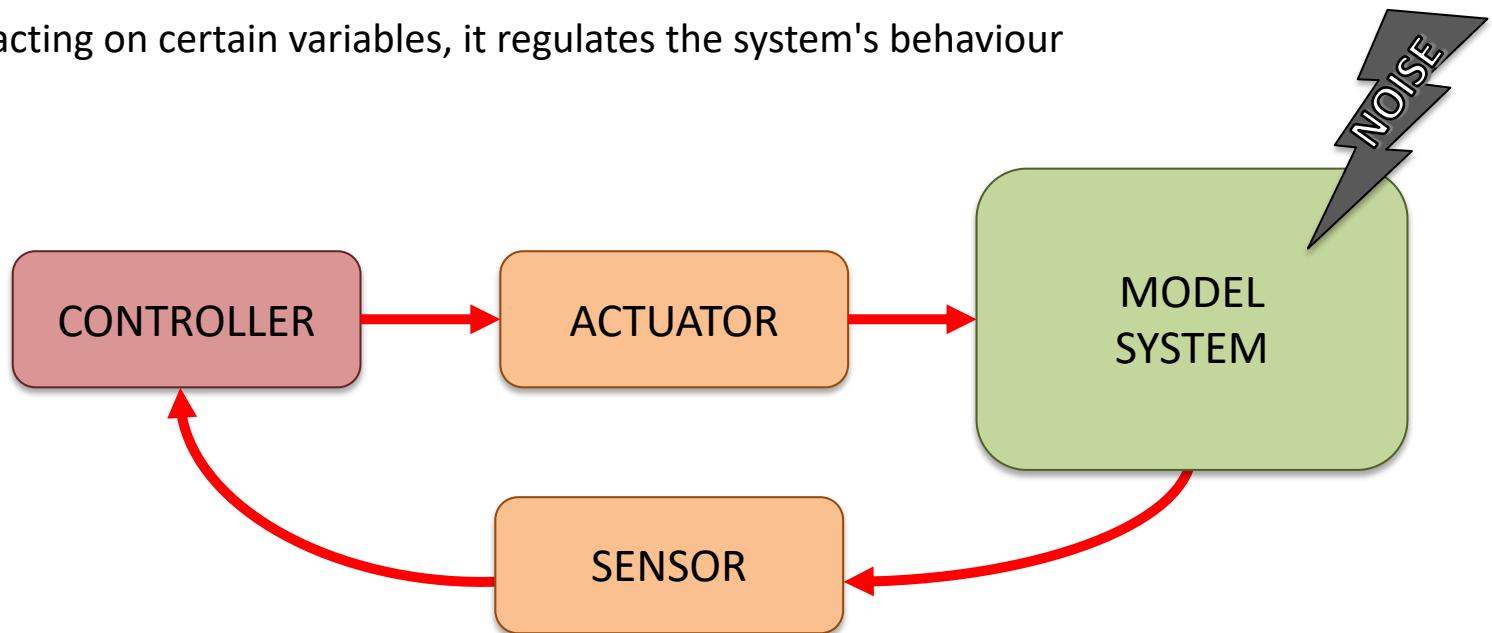
# Machine Learning: Supervised

- Having **inputs** and related **outputs**, the rule that associates the inputs to the outputs is extracted
  - Classification → *discrete response* (i.e. ON/OFF, 0/1/2, dog/cat/rabbit ...)
  - Regression → *continuous response*



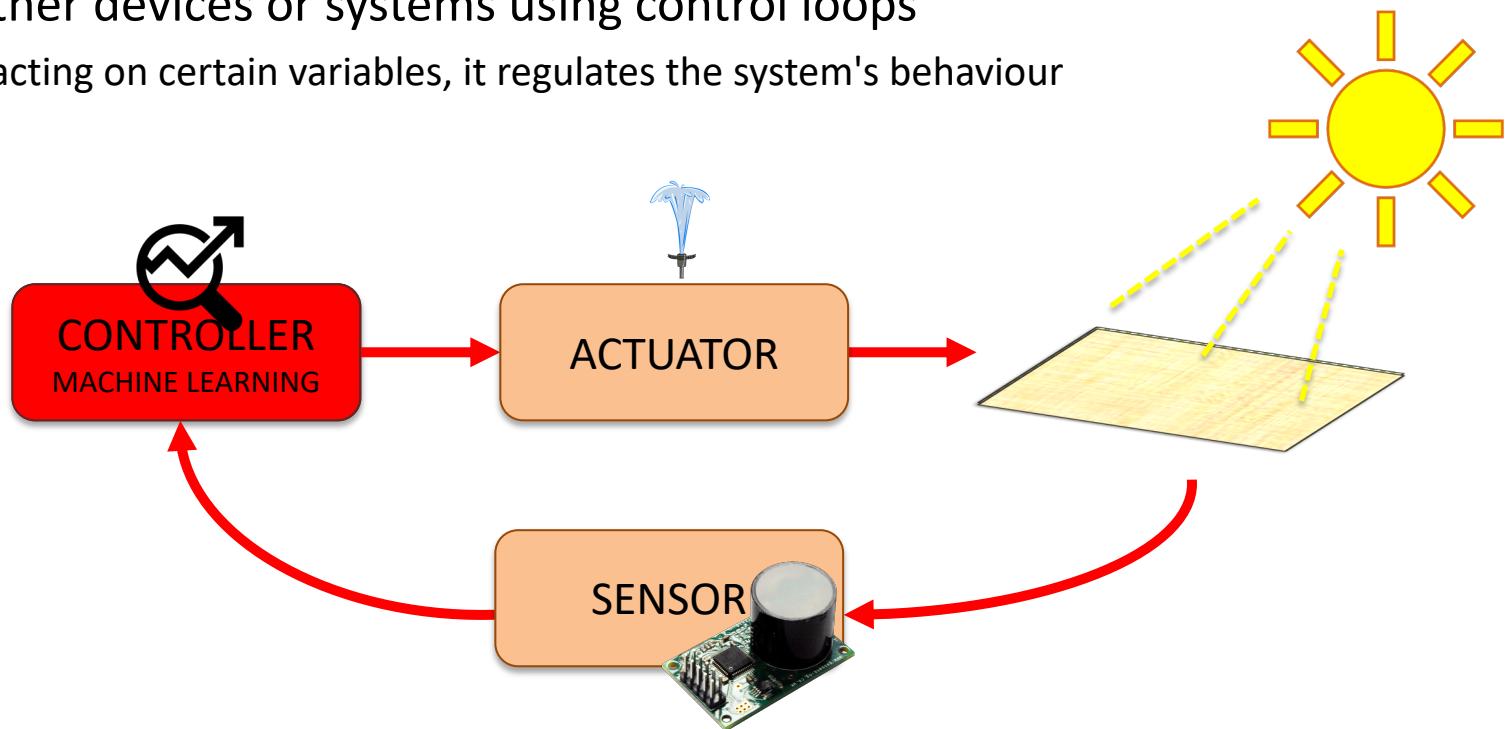
# Control system

- A control system manages, commands, directs, or regulates the behavior of other devices or systems using control loops
  - acting on certain variables, it regulates the system's behaviour



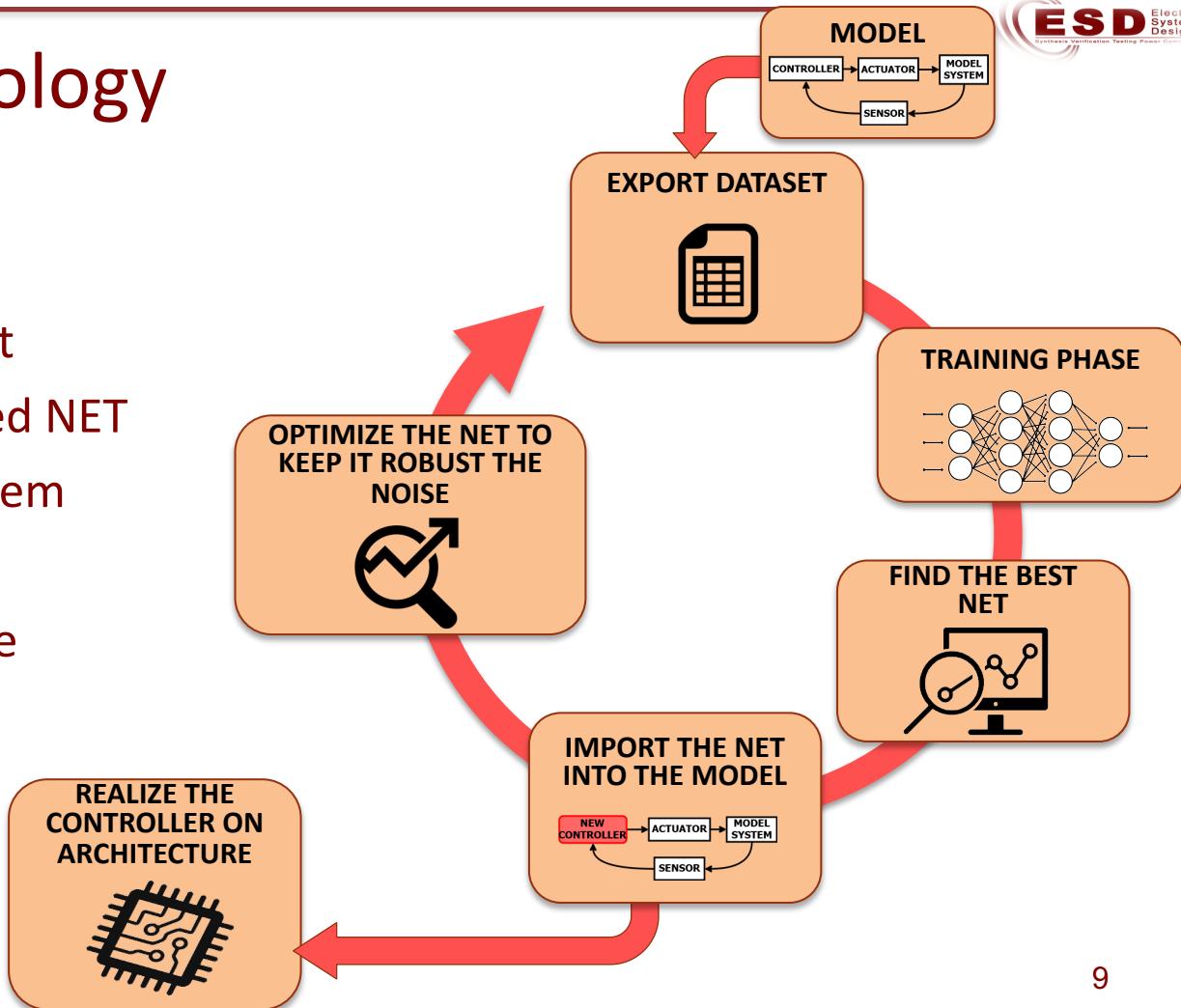
# Control system

- A control system manages, commands, directs, or regulates the behavior of other devices or systems using control loops
  - acting on certain variables, it regulates the system's behaviour



# Methodology

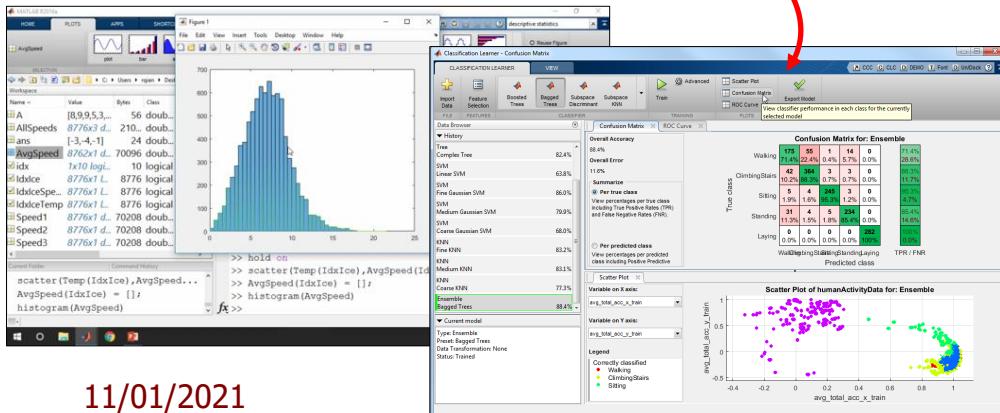
- Main steps are:
  - Export the dataset
  - Build of the trained NET
  - Implement in system
  - Optimize the NET
  - Generate the code



# Development environment

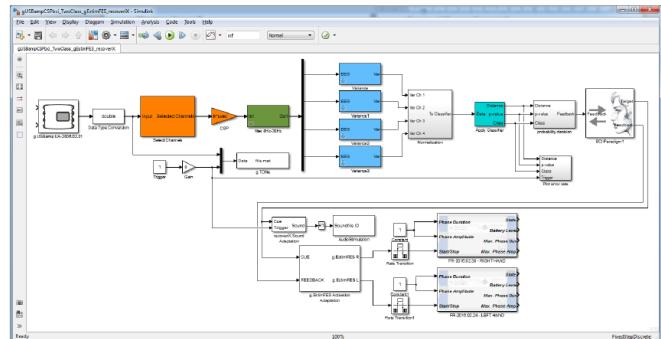
## The MATLAB environment

- Is an environment for numerical calculation and statistical analysis
- Allows the use of machine learning techniques
  - Classification Learner App
  - Regression Learner App



## The Simulink environment

- Allows construction of the simulation environment for the hardware and software design of a system
- Is closely integrated with MATLAB

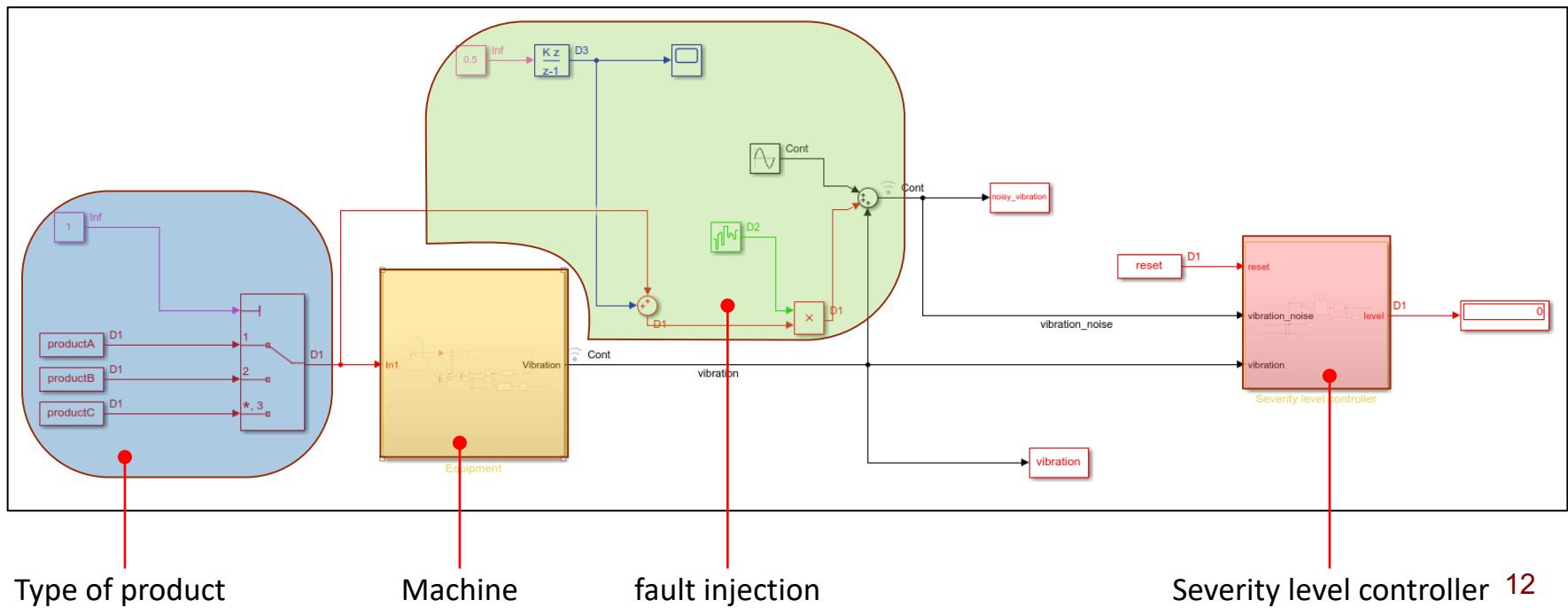
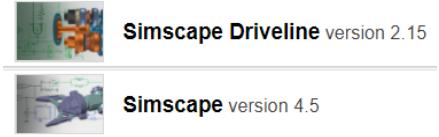


# **CASE STUDY: FAILURE OF A DRIVE SHAFT OF A MACHINE**

# Model scheme

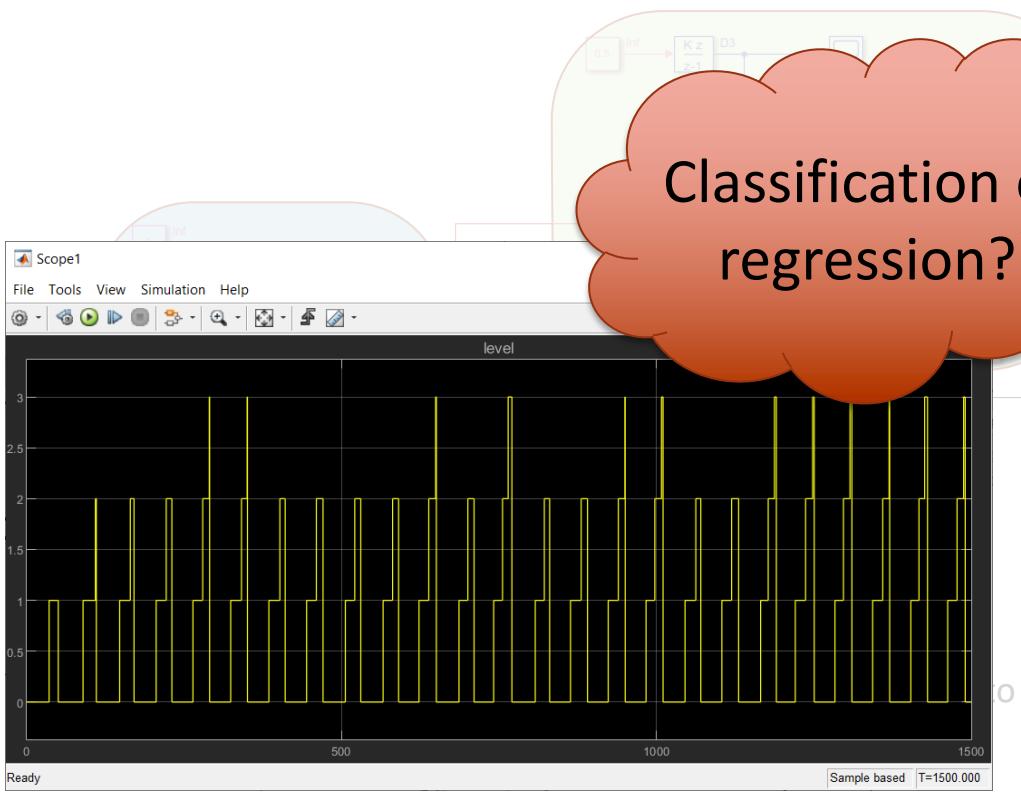
- The machine produces a product in 50 seconds
- There are 10 seconds between one production and the next
- The vibration of the machine is monitored

## MATLAB tools dependency



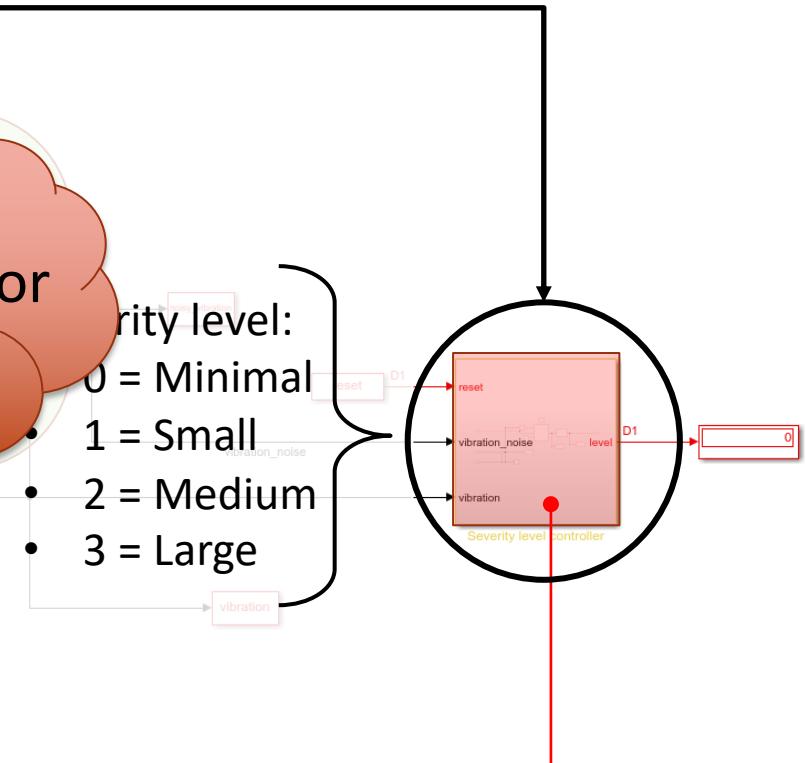
# Goal

Replicate the behavior



Classification or regression?

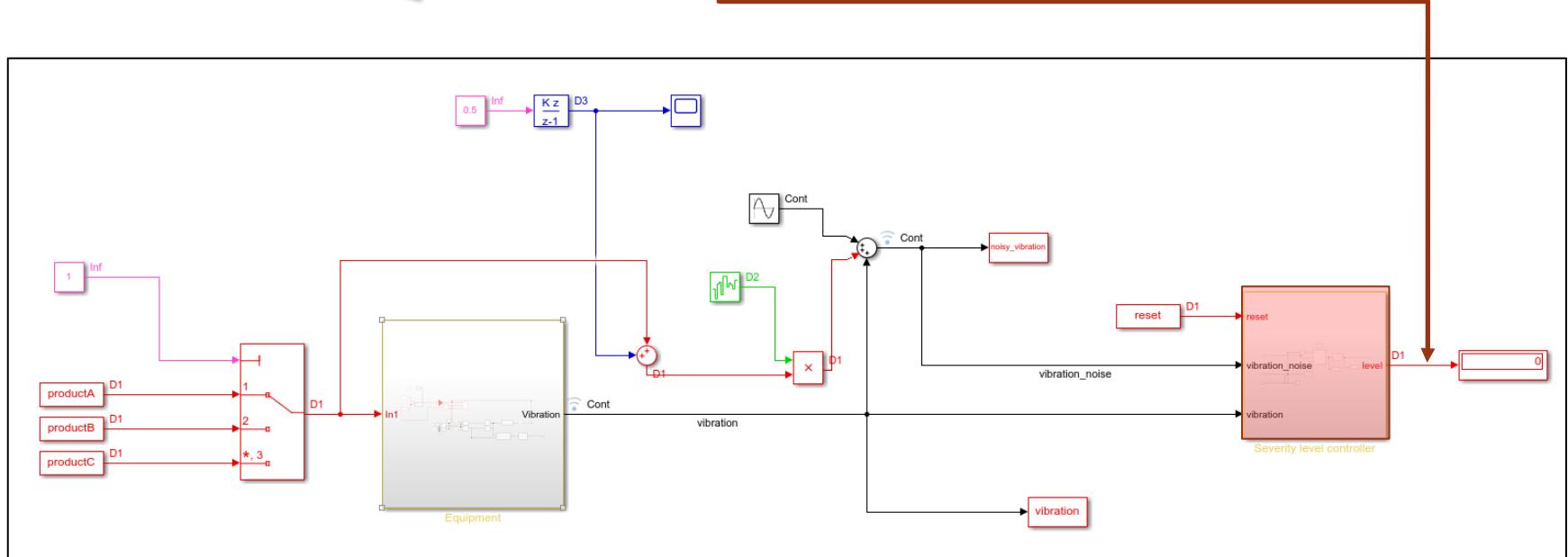
- Severity level:
- 0 = Minimal
- 1 = Small
- 2 = Medium
- 3 = Large



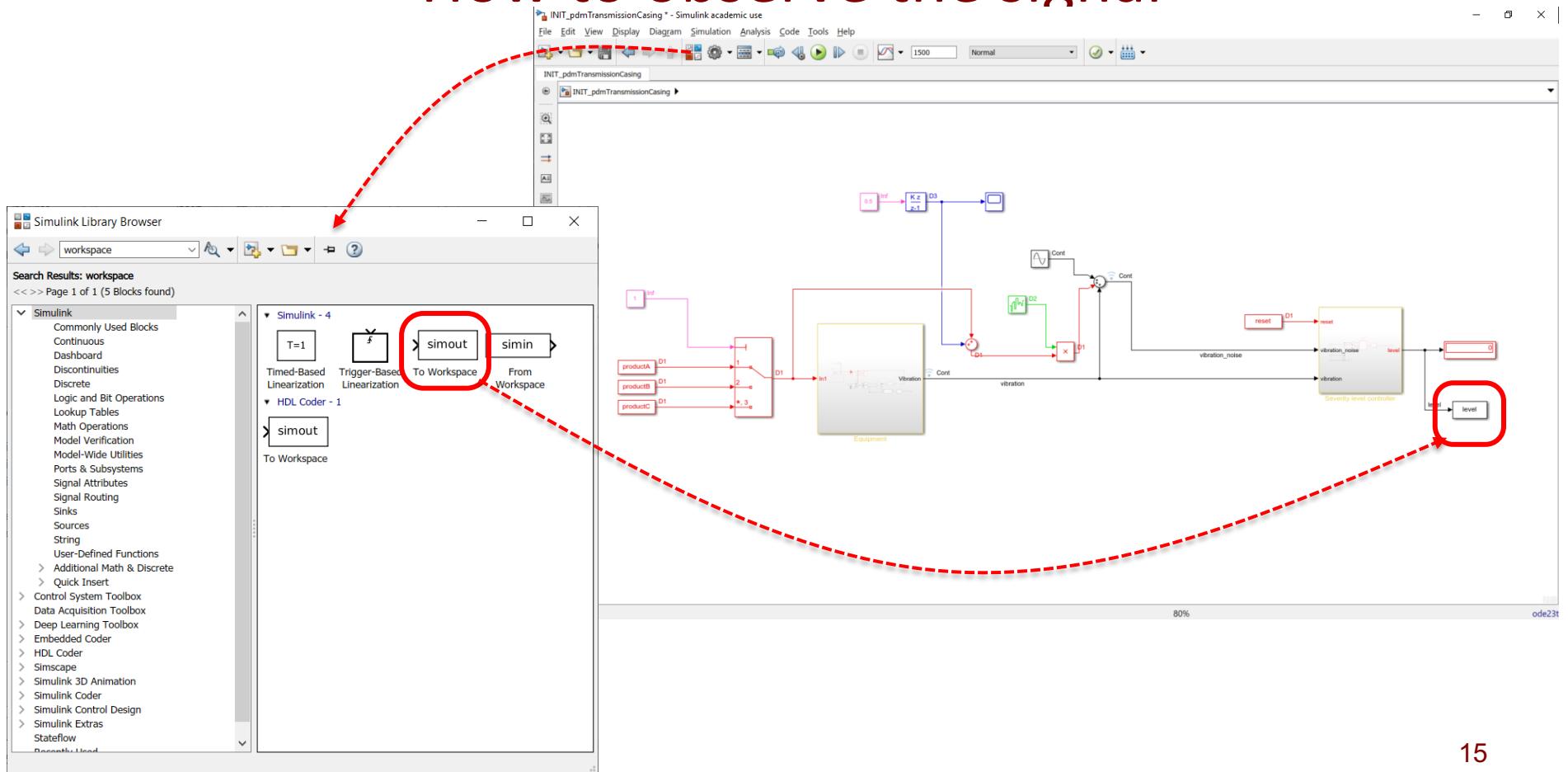
# Dataset generation

- Training dataset

[ inputs | level ]



# How to observe the signal



# How to observe the signal

The screenshot illustrates the MATLAB/Simulink environment for observing signals.

**Left Panel (MATLAB R2018b - academic use):**

- Current Folder:** C:\Users\aless\OneDrive - Università degli Studi di Verona\ASEM\_lesson\laboratorio\asemLab
- Workspace:** Shows variables like brake, ClutchRadius, DampingCoeff, gearA, gearB, gearC, level, productA, productB, productC, tempA, tempB, tempC, and init.
- Command Window:** Displays the command `>> init`.
- Editor:** Shows the code for `pdmTransmissionCasing22.slx`:

```

1 Name -
2 slprj
3 Soluzioni
4 brake.mat
5 createtablefromtimeseries.m
6 init.m
7 INIT_pdmTransmissionCasing.six
8 INIT_pdmTransmissionCasing.slx
9 INIT_pdmTransmissionCasing.slvx
10 out.mat
11 pdmTransmissionCasing22.slx
12 productA.mat
13 productB.mat
14 productC.mat
15 reset.mat
16 tempA.mat
17 tempB.mat
18 tempC.mat
19 Untitled.m

```

**Middle Panel (Simulink Model):**

The model `INIT_pdmTransmissionCasing` contains the following components and connections:

- A `vibration_noise` source block connects to a `vibration` block.
- The `vibration` block connects to a `Severity level controller` block.
- The `Severity level controller` block has two outputs: one to a `reset` block and one to a `level` scope block.
- A `reset` block receives a signal from the `Severity level controller` and provides a feedback signal to the `vibration` block.
- A `level` scope block displays the signal from the `Severity level controller`.

**Right Panel (Variables - level):**

This window shows a time series named `level`. The data table is as follows:

Time	Data:1
0	0
1.0000	0
0.0020	0
0.0030	0
0.0040	0
0.0050	0
0.0060	0
0.0070	0
0.0080	0
0.0090	0
0.0100	0
0.0110	0
0.0120	0
0.0130	0
0.0140	0
0.0150	0
0.0160	0
0.0170	0
0.0180	0
0.0190	0
0.0200	0

The current time is uniform from 0 to 143.8 seconds.

**Bottom Right:** Page number 16.

# Dataset generation

- Dataset
  1. Build a matrix

```
A = [signal.Data    signal2.Data    ... ]);
```

2. Save dataset in a CSV file

```
csvwrite("dataset.csv", [ double(signal.Data), double(signal2.Data) ... ]);
```

# Training phase

Training phase

The screenshot shows the MATLAB R2018b interface with the Classification Learner app open. The 'Classification Learner' button in the Apps tab is circled in red. A red arrow points from this button to the Classification Learner app window. Another red arrow points from the Classification Learner app window down to the 'PREDICTORS' and 'RESPONSE' labels.

**PREDICTORS**

**RESPONSE**

Classification Learner

New Session

Data set

Workspace Variable

dataset4

Response

Predictors

	Name	Type	Range
<input checked="" type="checkbox"/>	VarName1	double	-1.8157 .. 1.8157
<input checked="" type="checkbox"/>	VarName2	double	-14.808 .. 14.125
<input checked="" type="checkbox"/>	VarName3	double	-14.966 .. 13.106
<input type="checkbox"/>	VarName4	double	0 .. 3

Add All Remove All

How to prepare data

Validation

Cross-Validation

Protects against overfitting by partitioning the data set into folds and estimating accuracy on each fold.

Cross-validation folds: 5 folds

Holdout Validation

Recommended for large data sets.

Percent held out: 25%

No Validation

No protection against overfitting.

Read about validation

Start Session Cancel

Response variable is numeric. Distinct values will be interpreted as class labels.

# Training phase

Classification Learner

CLASSIFICATION LEARNER

New Session Feature Selection PCA All Quick-To...

From Workspace Start a new session by selecting data from the workspace and specifying a validation set.

From File Start a new session by importing data from a file and specifying a validation set.

Current Model

Model 1: Draft

Model Type Preset: Fine Tree Maximum number of splits: 100 Split criterion: Gini's diversity index Surrogate decision splits: Off

Feature Selection All features used in the model, PCA PCA disabled

Data set: dataset4 Observations: 15001 Size: 471 kB Predictors: 3 Response: VarName4 Response Classes: 4

Classification Learner - Scatter Plot

CLASSIFICATION LEARNER

New Session Feature Selection PCA All Quick-To...

GET STARTED

Data Browser All Quick-To...

History 1 Tree Last change: Disabled PCA

DECISION TREES

- Fine Tree
- Medium Tree

DISCRIMINANT ANALYSIS

- Linear Discriminant
- Quadratic Discriminant

LOGISTIC REGRESSION CLASSIFIER

- Logistic Regression

SUPPORT VECTOR MACHINES

- Linear SVM
- Quadratic SVM
- All SVMs

NEAREST NEIGHBOR CLASSIFIERS

- Fine KNN
- Medium KNN
- All KNNs

ENSEMBLE CLASSIFIERS

- Boosted Trees
- Bagged Trees

CLASSIFICATION LEARNER

New Session Feature Selection PCA All Quick-To...

VIEW

FILE FEATURES

MODEL TYPE

TRAINING

Scatter Plot

Original data set: dataset4

Plot Data Model p

Predictors X: VarName1 Y: VarName2

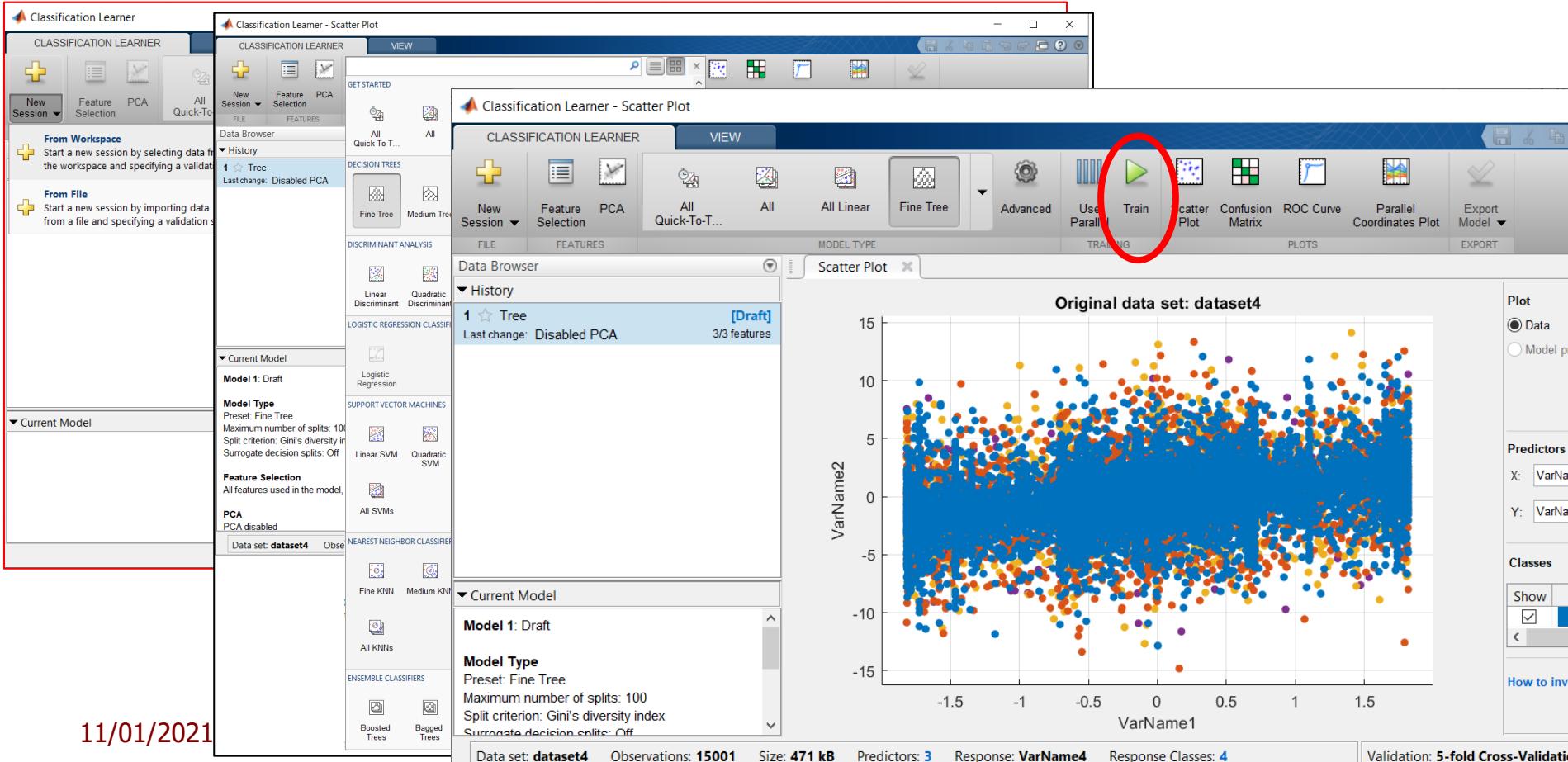
Classes Show < How to invert

VarName2

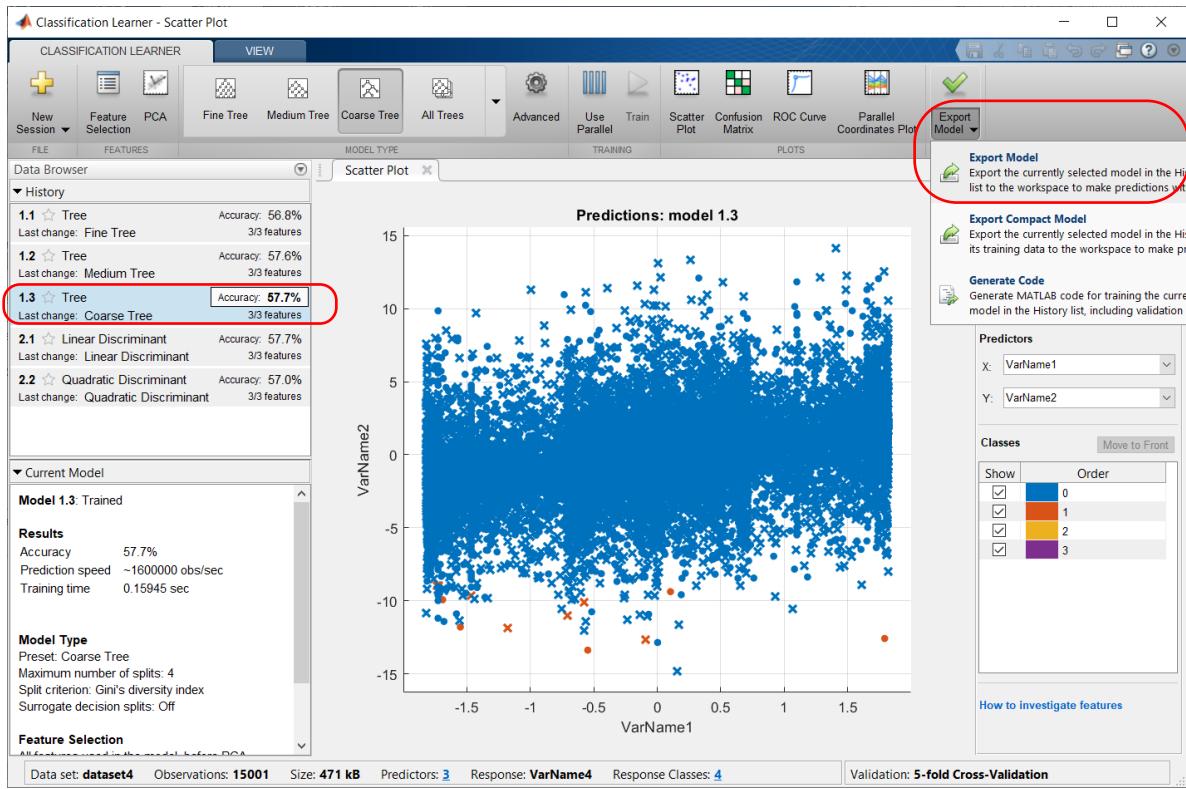
VarName1

11/01/2021

Validation: 5-fold Cross-Validation

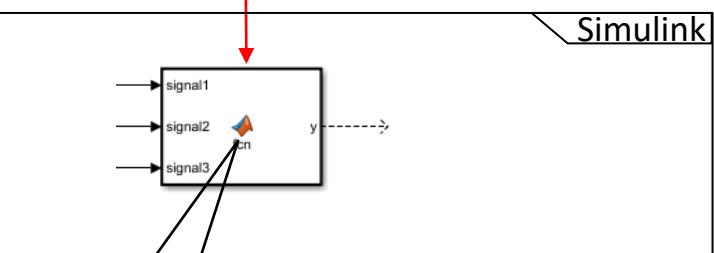


# Training phase and prediction phase



MATLAB

```
saveCompactModel(
    trainedModel.ClassificationTree,
    "myclass.mat")
```



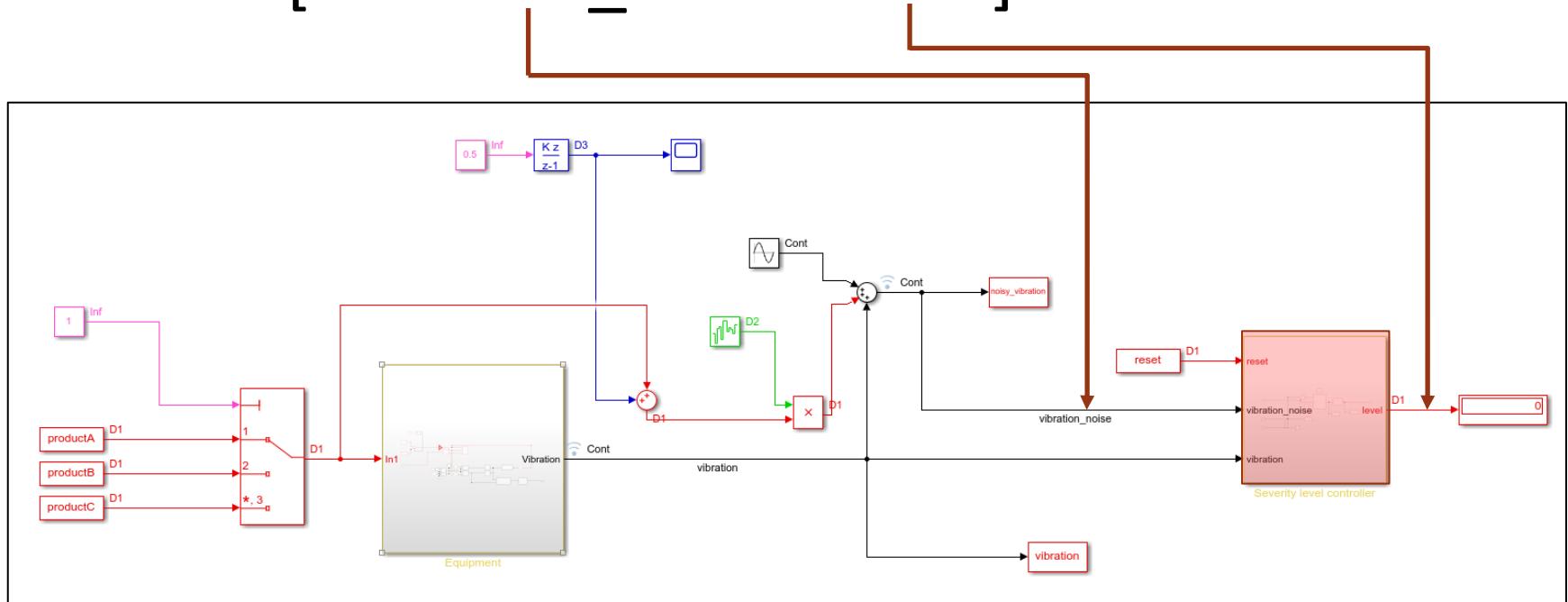
```
function y = fcn(signal1, signal2, signal3)

mdl = loadCompactModel("myclass.mat");
u = [signal1 signal2 signal3];
y = predict(mdl,u);
```

# Solution 1

- Build the following dataset

[vibration\_noise level]

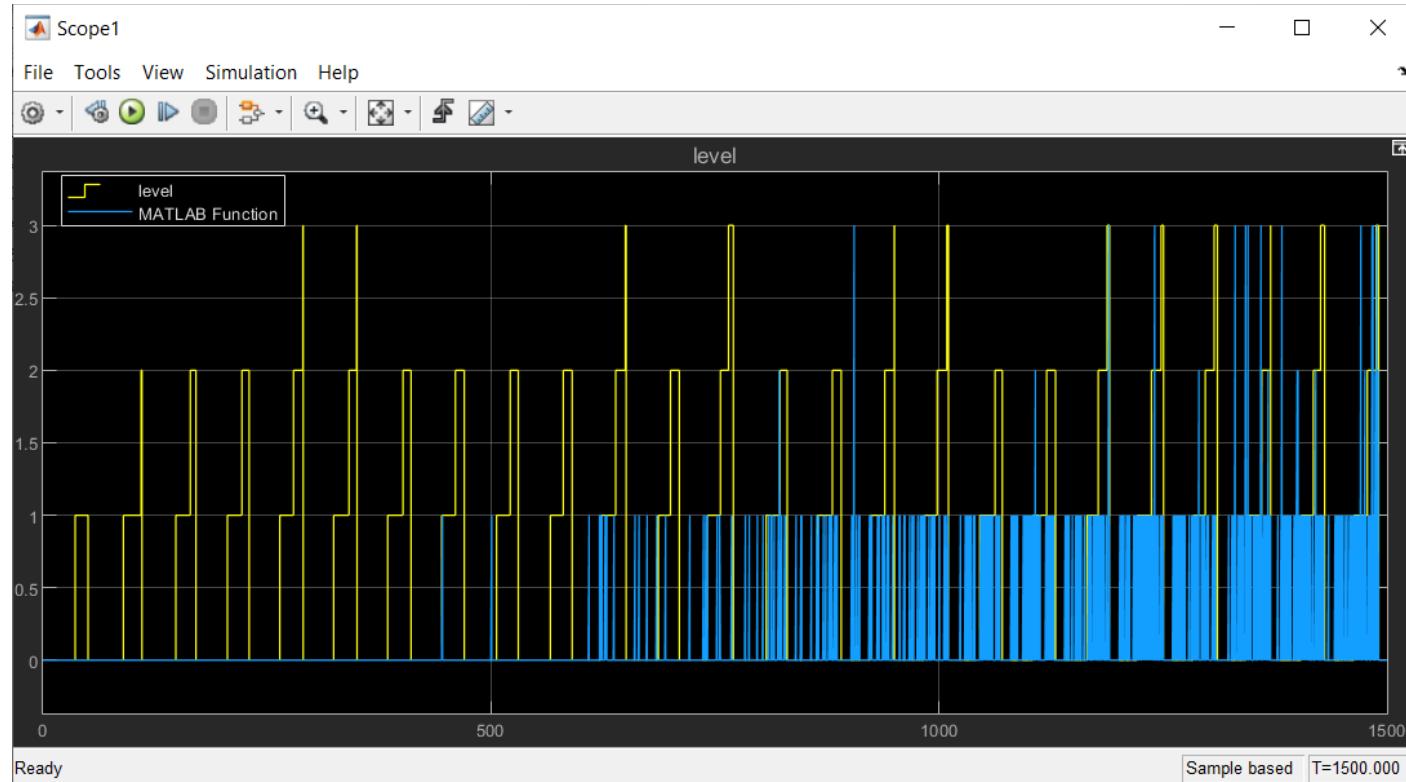


# MATLAB exercise

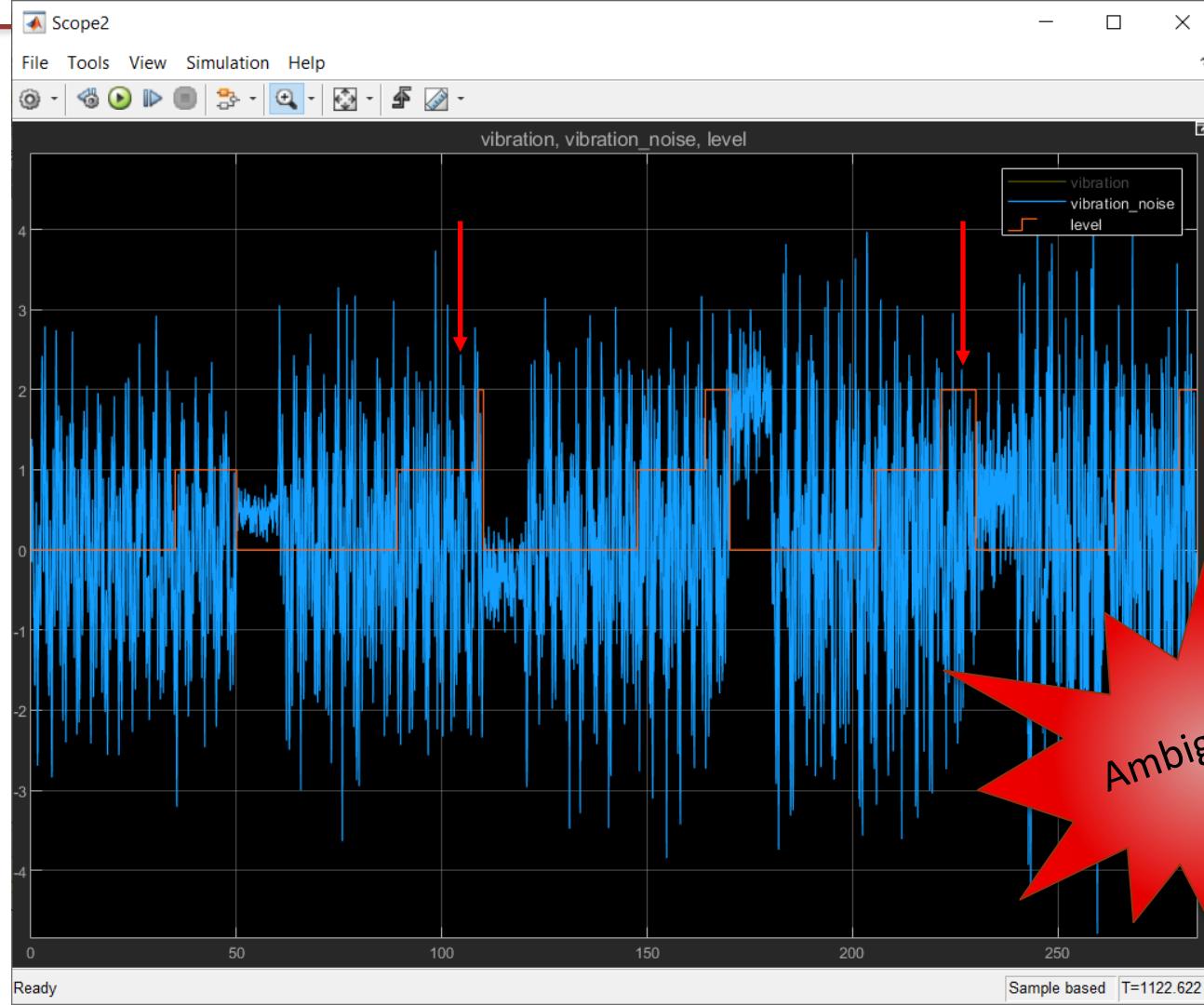
- Run MAT file  
init.mat
- Run model  
INIT\_pdmTransmissionCasing.slx

1

# Dataset: [vibration\_noise; level]



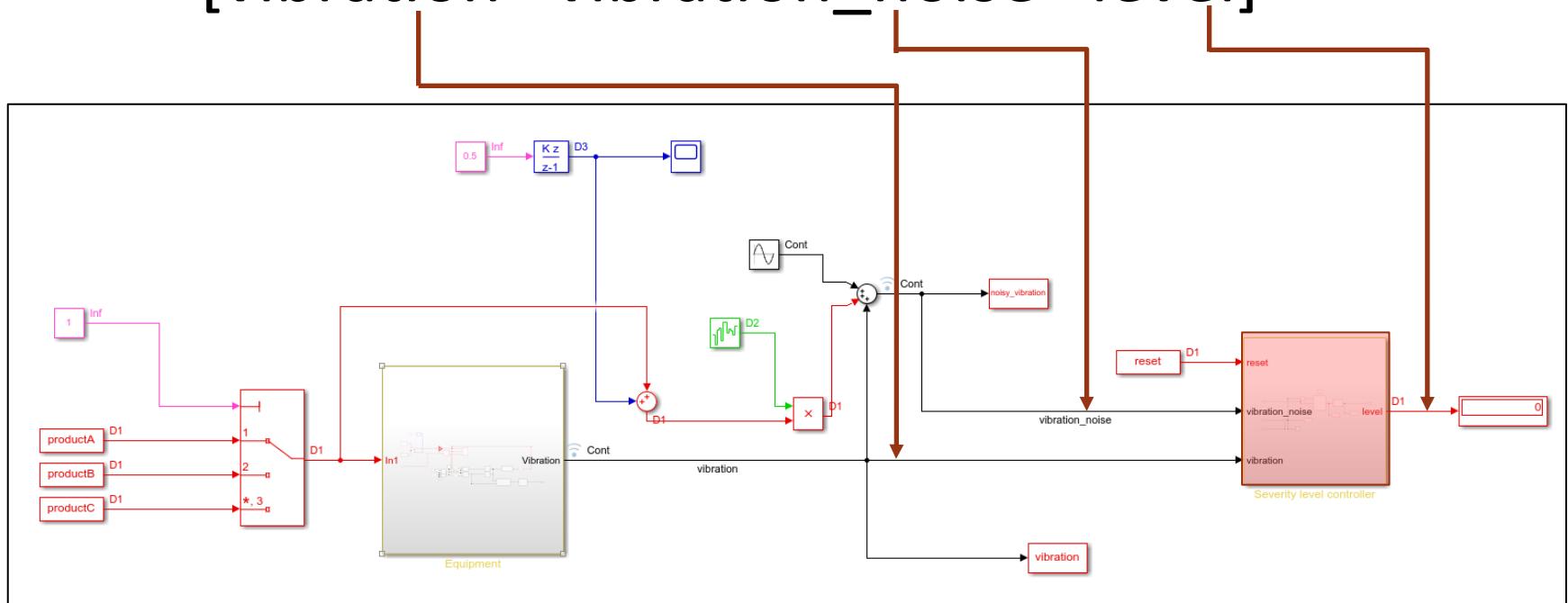
1



# Solution 2

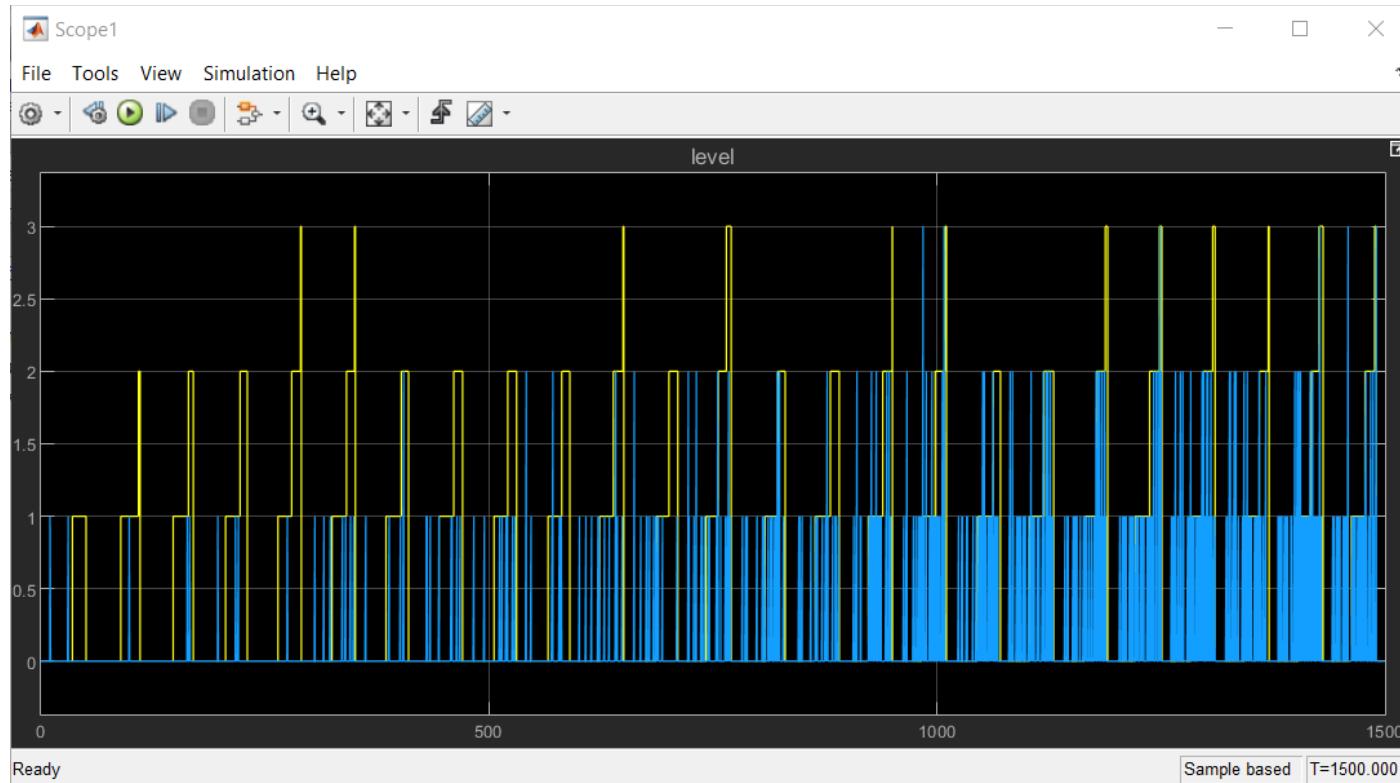
- Build the following dataset

[vibration    vibration\_noise    level]

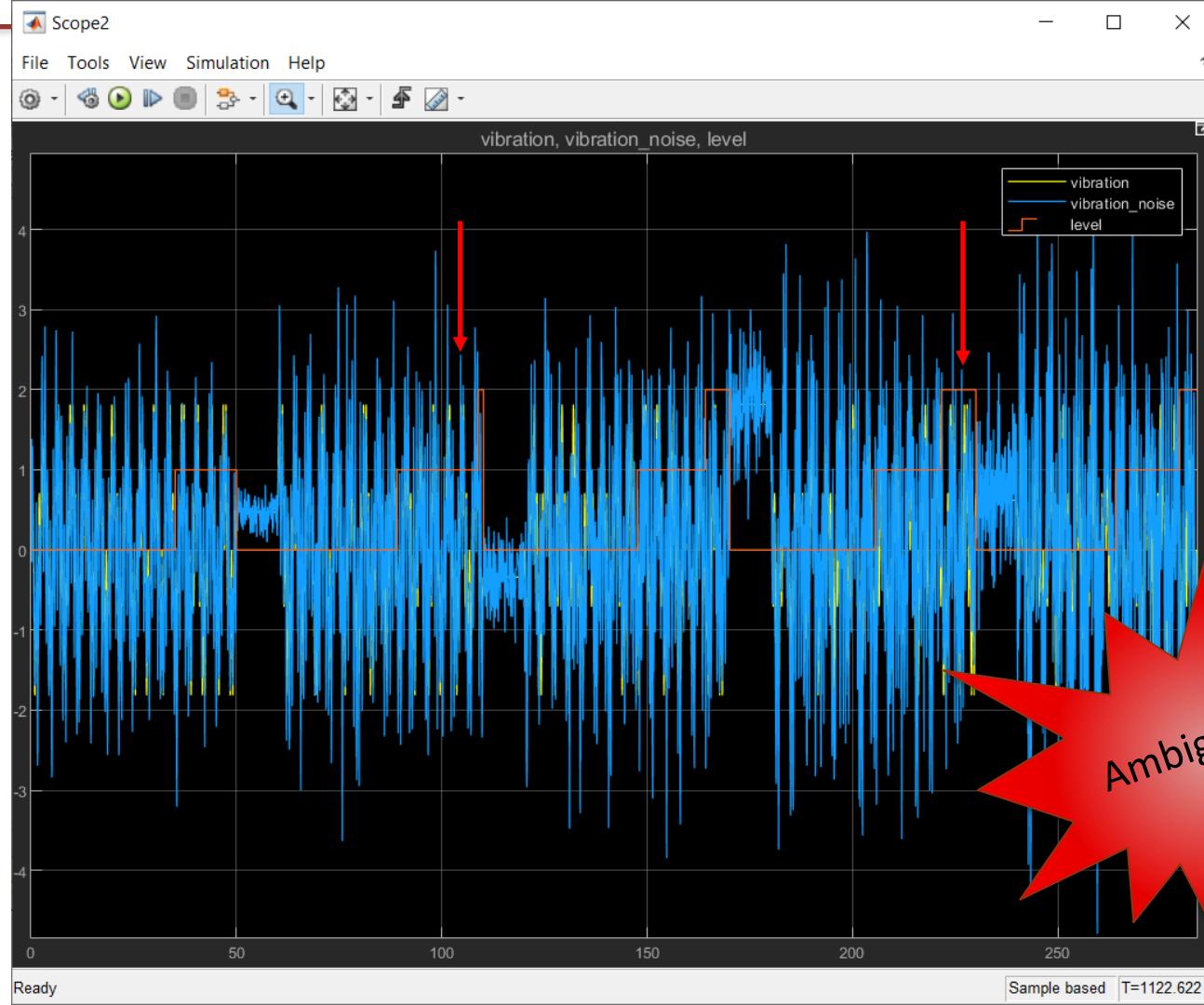


2

# Dataset: [vibration; vibration\_noise; level]

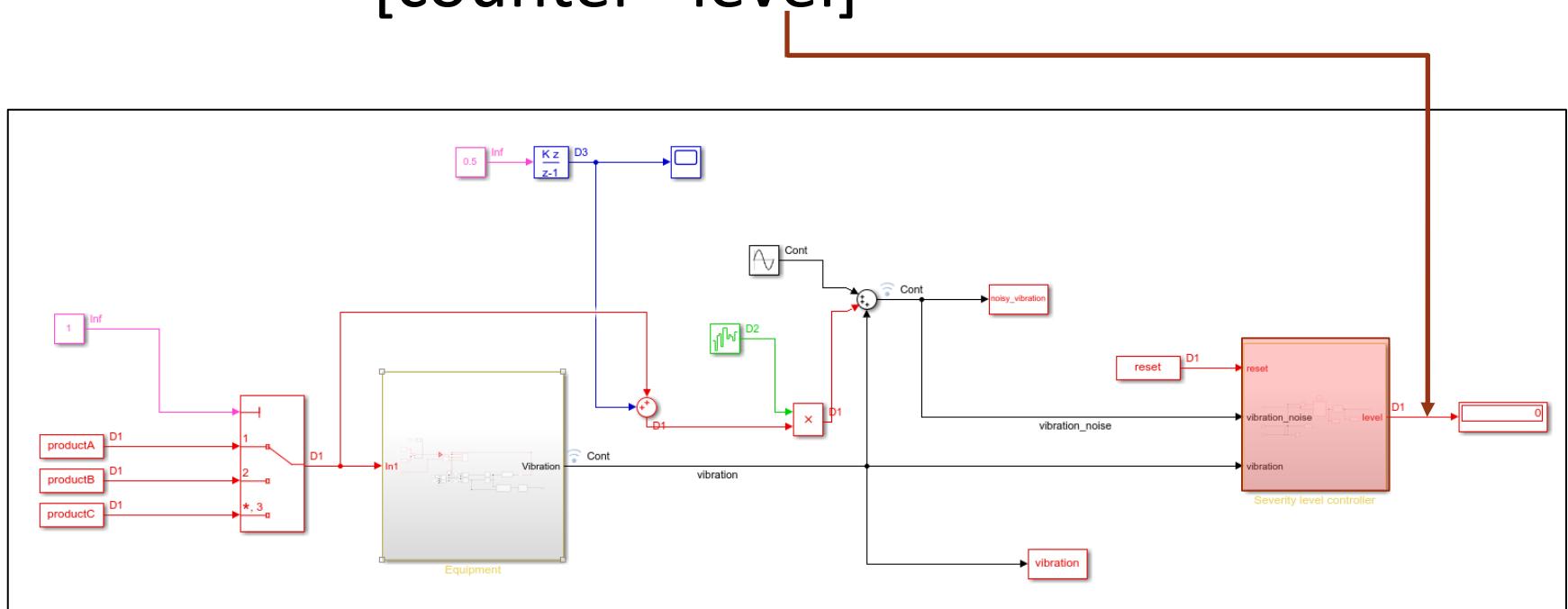


2



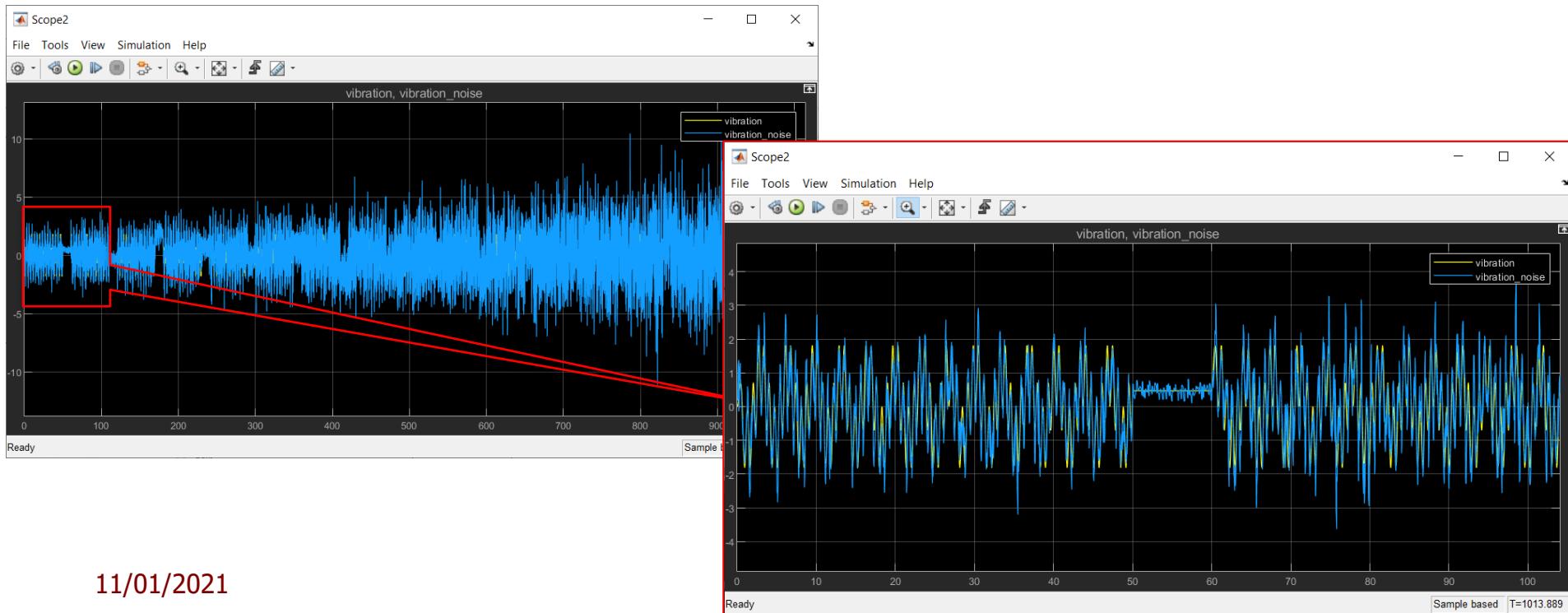
# Solution 3

- Build the following dataset  
[counter level]



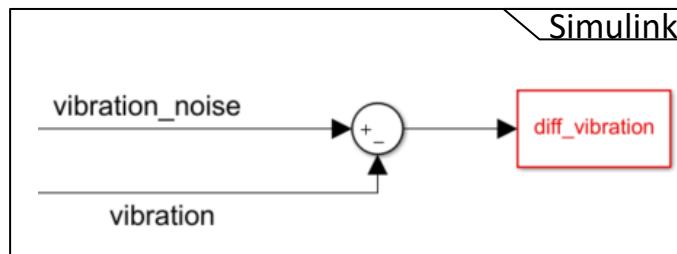
# Solution 3

- Counter: how many times the vibration with the fault deviates from the desired one, within a certain tolerance



# Dataset generation

- Build dataset: [counter level]

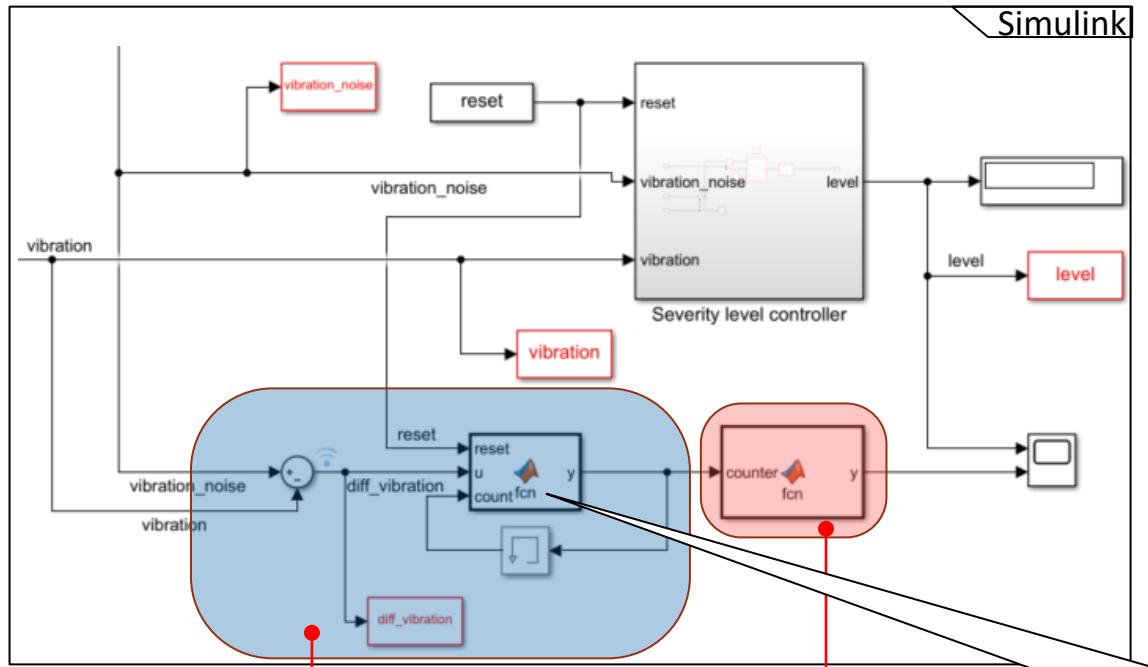


MATLAB

```
[n,m] = size(diff_vibration.Data)
tolerance = 0.1;
counter = zeros(n,1);
j = 1;
for i = 2:n
    j = mod(i,60001)+1;
    if (reset.signals.values(j)) == 1
        counter(i) = 0;
    elseif diff_vibration.Data(i) > tolerance
        counter(i) = counter(i-1) +1 ;
    else
        counter(i) = counter(i-1);
    end
end
A = [counter level.Data];
```

Find the best  
tolerance

# Prediction phase



To calculate  
counter at runtime

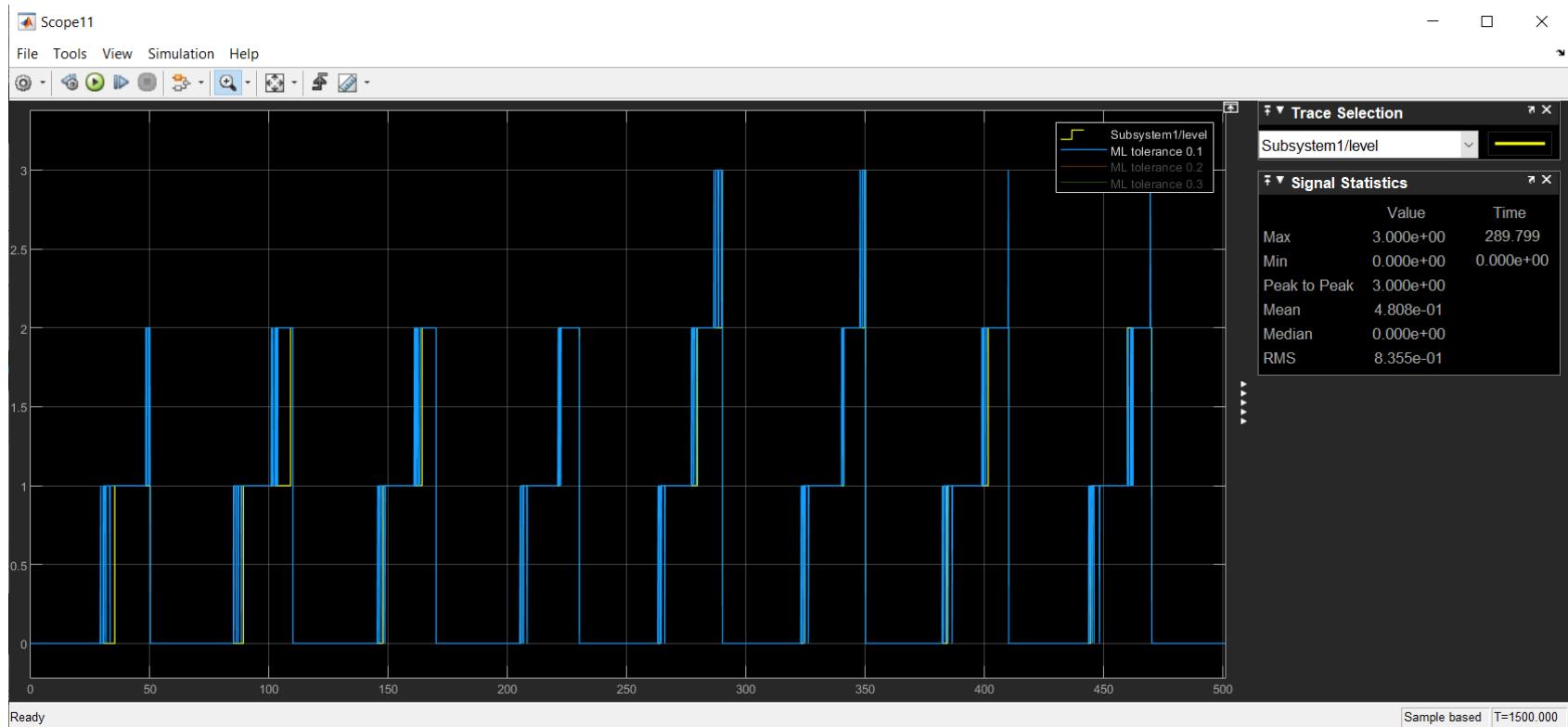
contains the machine  
learning algorithm

```
function y = fcn(reset, u, count)
tolerance = 0.1;
if reset==1
    count=0;
end

if u > tolerance
    count = count+1;
else
    count=count+0;
end
y = count;
```

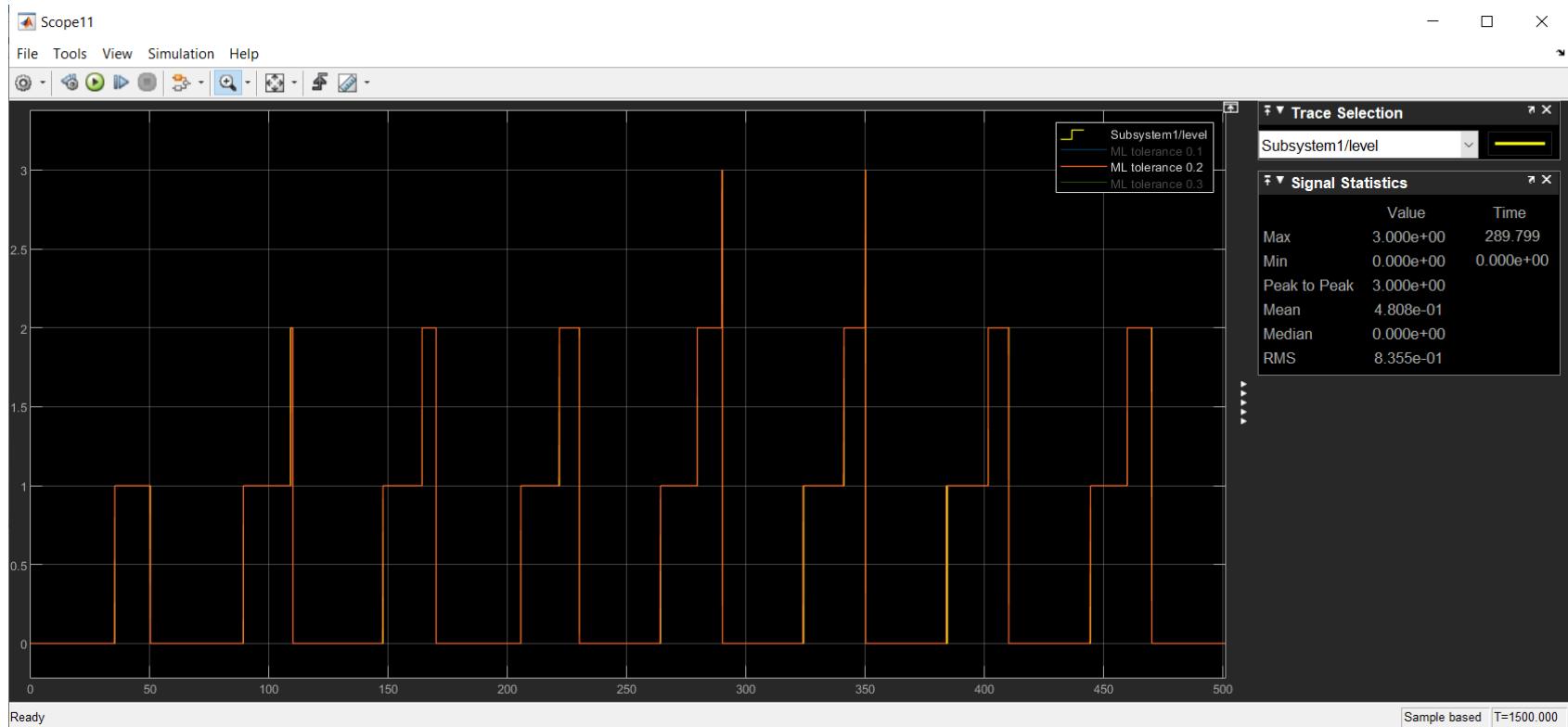
3

# Dataset: [counter; level] with tolerance = 0.1



3

# Dataset: [counter; level] with tolerance = 0.2



3

# Dataset: [counter; level] with tolerance = 0.3

