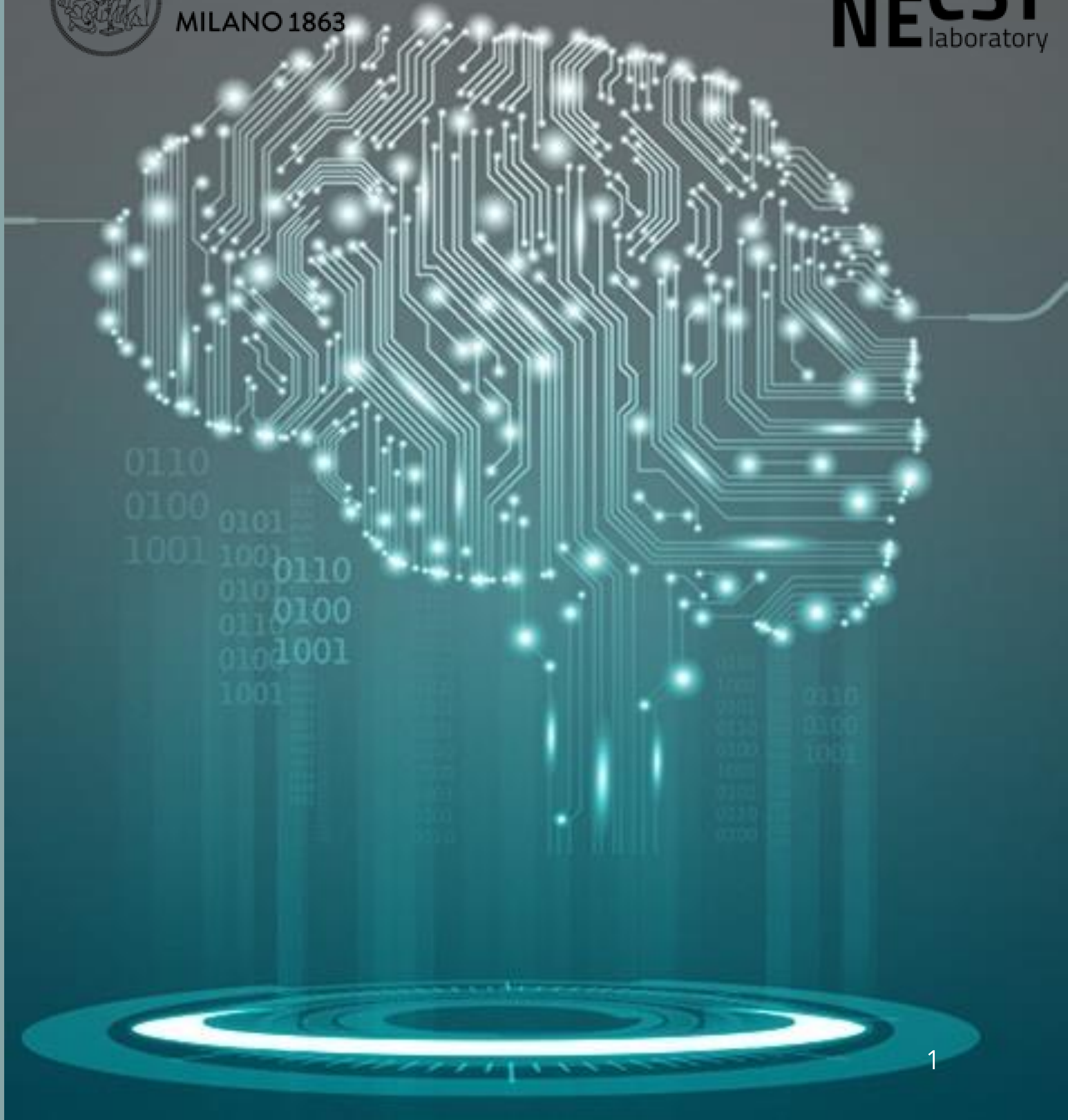
Creativity, Science and **Innovation**

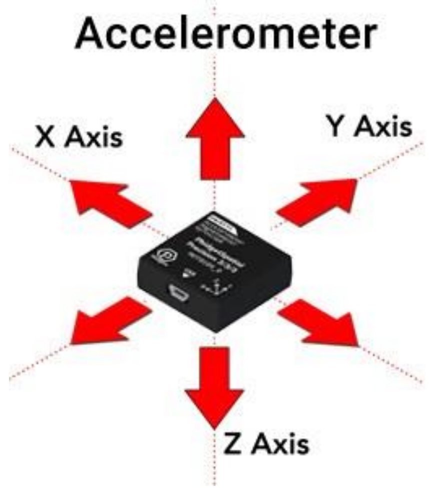
Introduction to Machine Learning for Time Series analysis

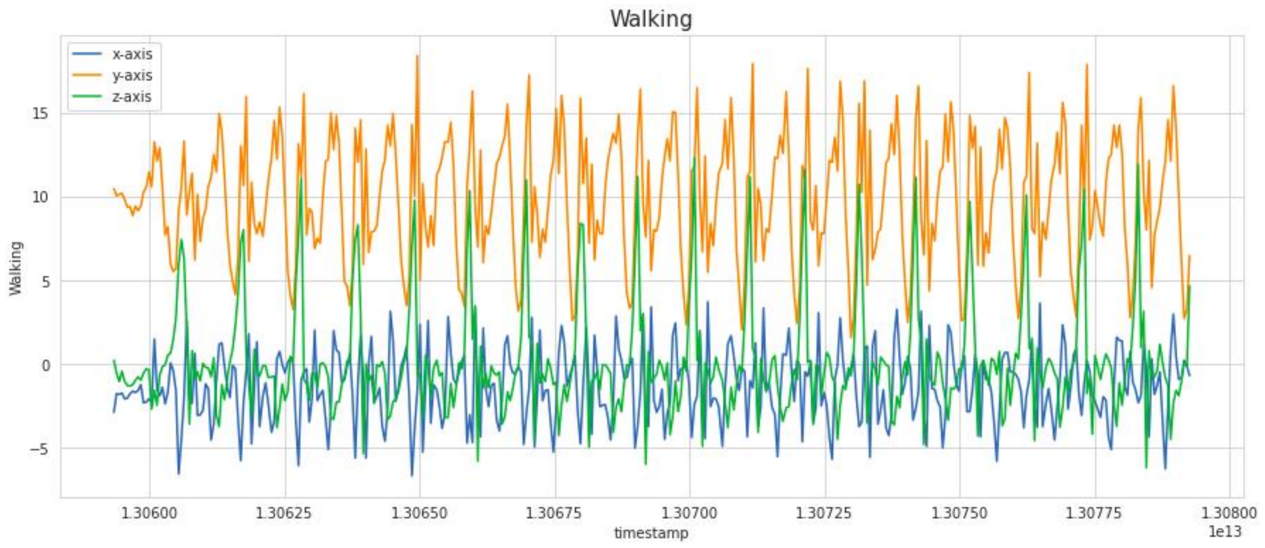
November 24th, 2025

Alessandro Verosimile
alessandro.verosimile@polimi.it



Machine Learning tasks with Time Series: Use case





Objective: classify the activity (walking, running, going upstairs, etc) of a person given data from an accelerometer

Machine Learning tasks with Time Series: Use case

Steps to train a ML model on this task:

1. Collect a dataset with time series (x, y, and z from accelerometer). Each time series has an associated label that indicates the activity performed.

1

X	Y	Z	Label
[2,-1,4,...]	[12,5,8,...]	[9,5,8,...]	Walking (0)
[2,-2,4,...]	[11,2,9,...]	[4,1,5,...]	Running (1)
[1,3,-1,...]	[10,5,7,...]	[4,1,5,...]	Walking (0)
[2,-1,1,...]	[8,11,8,...]	[5,5,7,...]	Walking (0)
[1,5,3,...]	[9,5,8,...]	[8,2,8,...]	Going up (2)
[2,-1,4,...]	[5,5,7,...]	[2,3,4,...]	Walking (0)
[4,5,9,...]	[4,5,9,...]	[4,1,5,...]	Walking (0)

Machine Learning tasks with Time Series: Use case

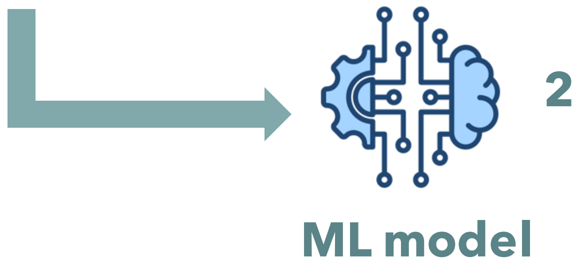
Steps to train a ML model on this task:

1. Collect a dataset with time series (x, y, and z from accelerometer). Each time series has an associated label that indicates the activity performed.

2. Train an ML model on such data. The model will learn to predict the Label given X, Y, and Z

1

X	Y	Z	Label
[2,-1,4,...]	[12,5,8,...]	[9,5,8,...]	Walking (0)
[2,-2,4,...]	[11,2,9,...]	[4,1,5,...]	Running (1)
[1,3,-1,...]	[10,5,7,...]	[4,1,5,...]	Walking (0)
[2,-1,1,...]	[8,11,8,...]	[5,5,7,...]	Walking (0)
[1,5,3,...]	[9,5,8,...]	[8,2,8,...]	Going up (2)
[2,-1,4,...]	[5,5,7,...]	[2,3,4,...]	Walking (0)
[4,5,9,...]	[4,5,9,...]	[4,1,5,...]	Walking (0)



Machine Learning tasks with Time Series: Use case

Steps to train a ML model on this task:

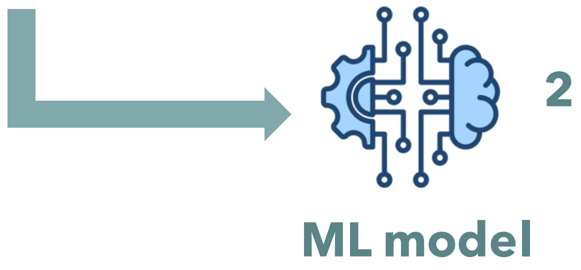
1. Collect a dataset with time series (x, y, and z from accelerometer). Each time series has an associated label that indicates the activity performed.

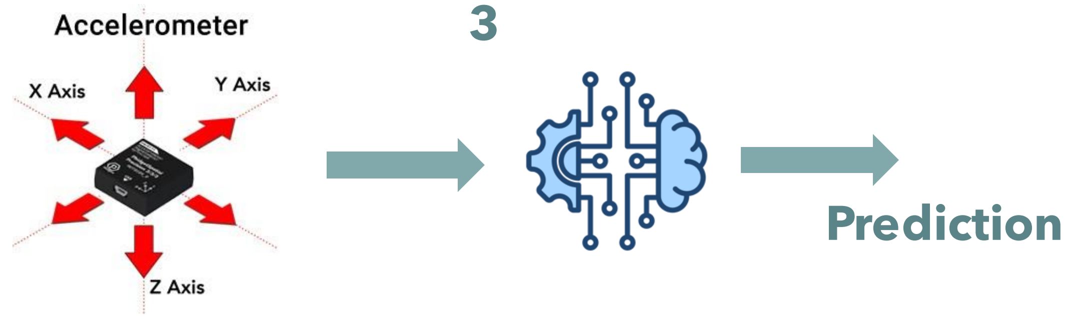
2. Train an ML model on such data. The model will learn to predict the Label given X, Y, and Z

3. When you have the final model, you will be able to use it in **inference** mode on new data coming from the device

1

X	Y	Z	Label
[2,-1,4,...]	[12,5,8,...]	[9,5,8,...]	Walking (0)
[2,-2,4,...]	[11,2,9,...]	[4,1,5,...]	Running (1)
[1,3,-1,...]	[10,5,7,...]	[4,1,5,...]	Walking (0)
[2,-1,1,...]	[8,11,8,...]	[5,5,7,...]	Walking (0)
[1,5,3,...]	[9,5,8,...]	[8,2,8,...]	Going up (2)
[2,-1,4,...]	[5,5,7,...]	[2,3,4,...]	Walking (0)
[4,5,9,...]	[4,5,9,...]	[4,1,5,...]	Walking (0)

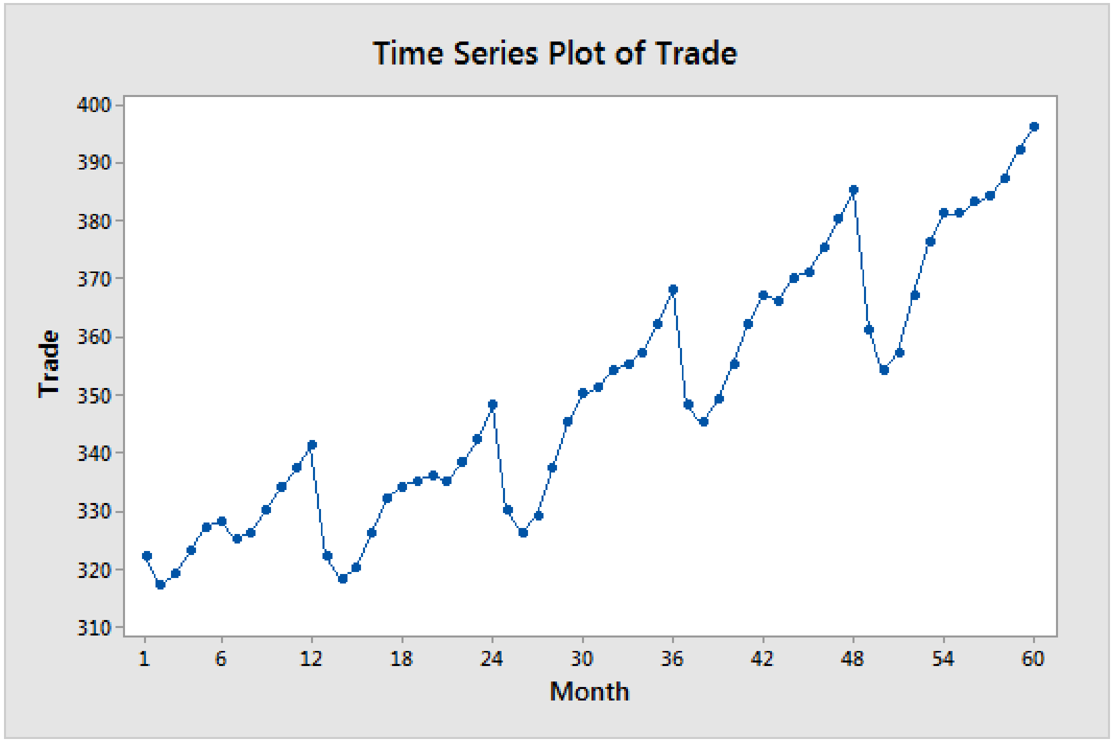




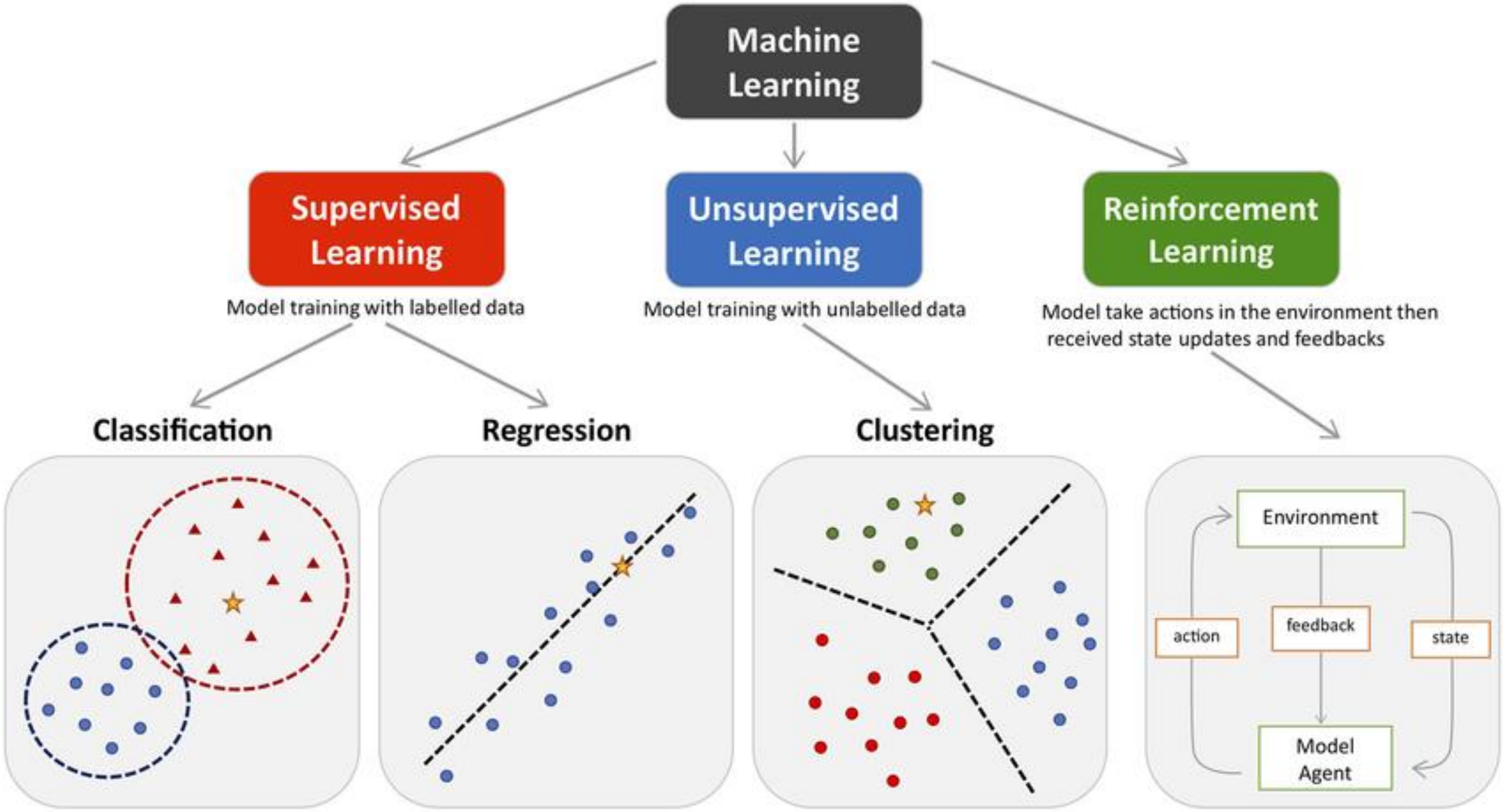
Machine Learning tasks with Time Series

- Forecasting: You have ***n*** values of the time series in input, and you want to predict the value at time *n+1*. (example: bitcoin forecasting, weather forecasting)

- Classification: You have n values of the time series in input, and you want to predict the class it belongs to. (example: activity recognition from accelerometer data)



Machine Learning categories



Supervised Machine Learning: an example

	Weight	Blood Pr.
Patient 1	54	112
Patient 2	68	130

Supervised Machine Learning: an example

	Weight	Blood Pr.
Patient 1	54	112
Patient 2	68	130

$$\text{Blood Pr} = W_1 * \text{Weight} + W_0$$

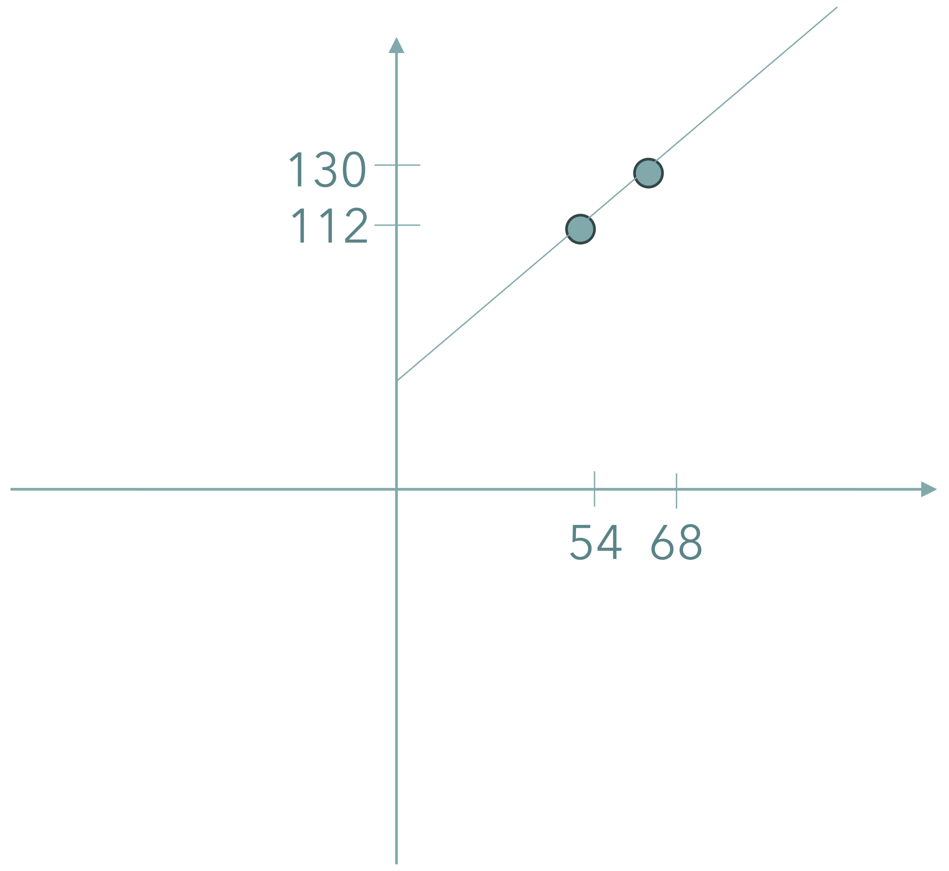
$$\begin{cases} 112 = W_1 * 54 + W_0 \\ 130 = W_1 * 68 + W_0 \end{cases}$$

Supervised Machine Learning: an example

	Weight	Blood Pr.
Patient 1	54	112
Patient 2	68	130

Blood Pr = W_1 * Weight + W_0

$$\begin{cases} 112 = W_1 * 54 + W_0 \\ 130 = W_1 * 68 + W_0 \end{cases}$$

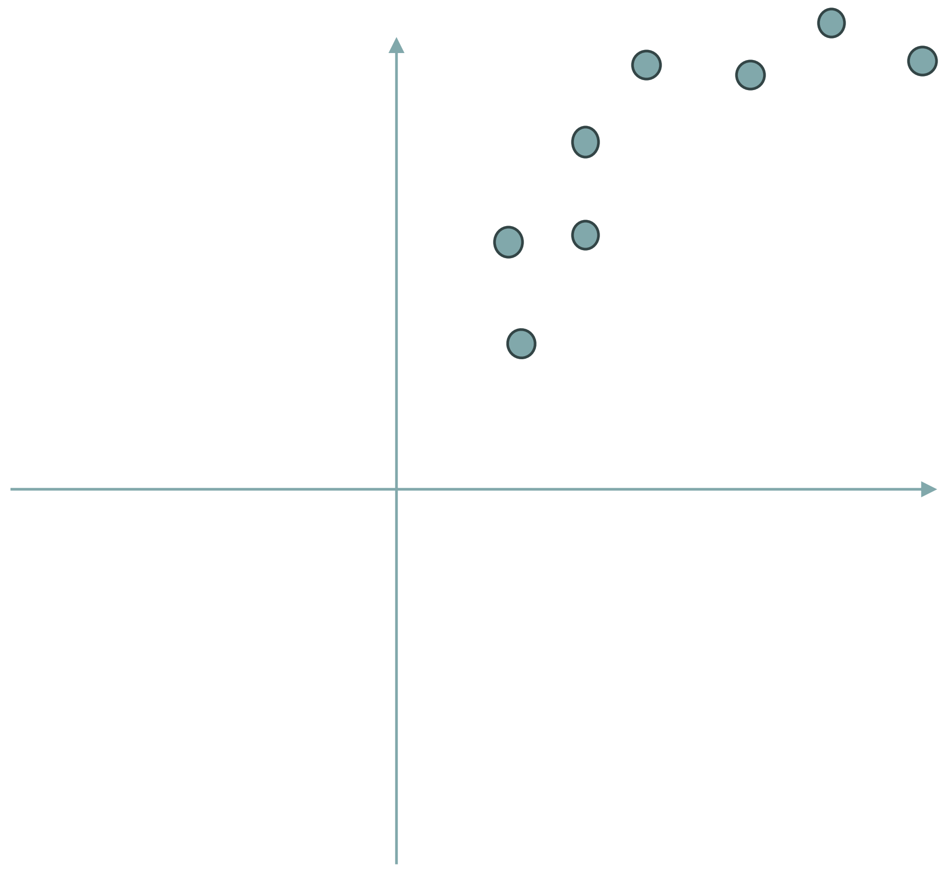


Supervised Machine Learning: an example

	Weight	Blood Pr.
Patient 1	54	112
Patient 2	68	130
Patient 3	57	115
Patient 4	56	116
Patient 5	77	132
Patient 6	81	138
Patient 7	74	130
Patient 8	66	122

Supervised Machine Learning: an example

	Weight	Blood Pr.
Patient 1	54	112
Patient 2	68	130
Patient 3	57	115
Patient 4	56	116
Patient 5	77	132
Patient 6	81	138
Patient 7	74	130
Patient 8	66	122

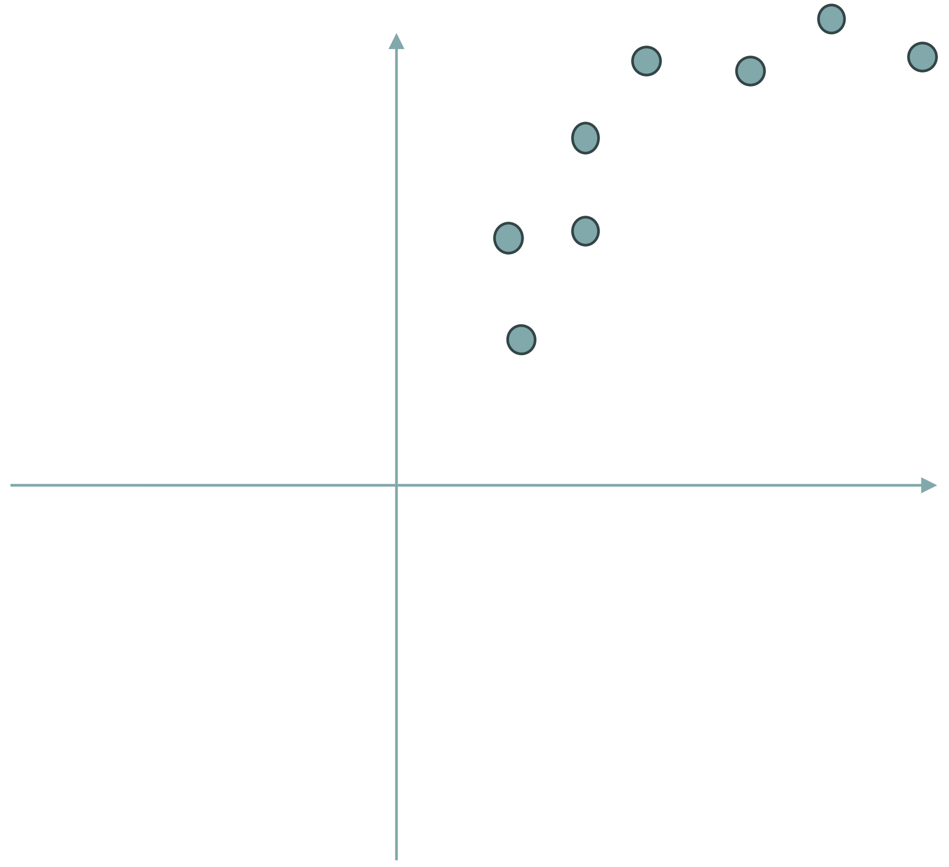


Supervised Machine Learning: an example

	Weight	Blood Pr.
Patient 1	54	112
Patient 2	68	130
Patient 3	57	115
Patient 4	56	116
Patient 5	77	132
Patient 6	81	138
Patient 7	74	130
Patient 8	66	122

$$\begin{cases} 112 = W_1 * 54 + W_0 \\ 130 = W_1 * 68 + W_0 \\ \dots \\ 122 = W_1 * 66 + W_0 \end{cases}$$

?

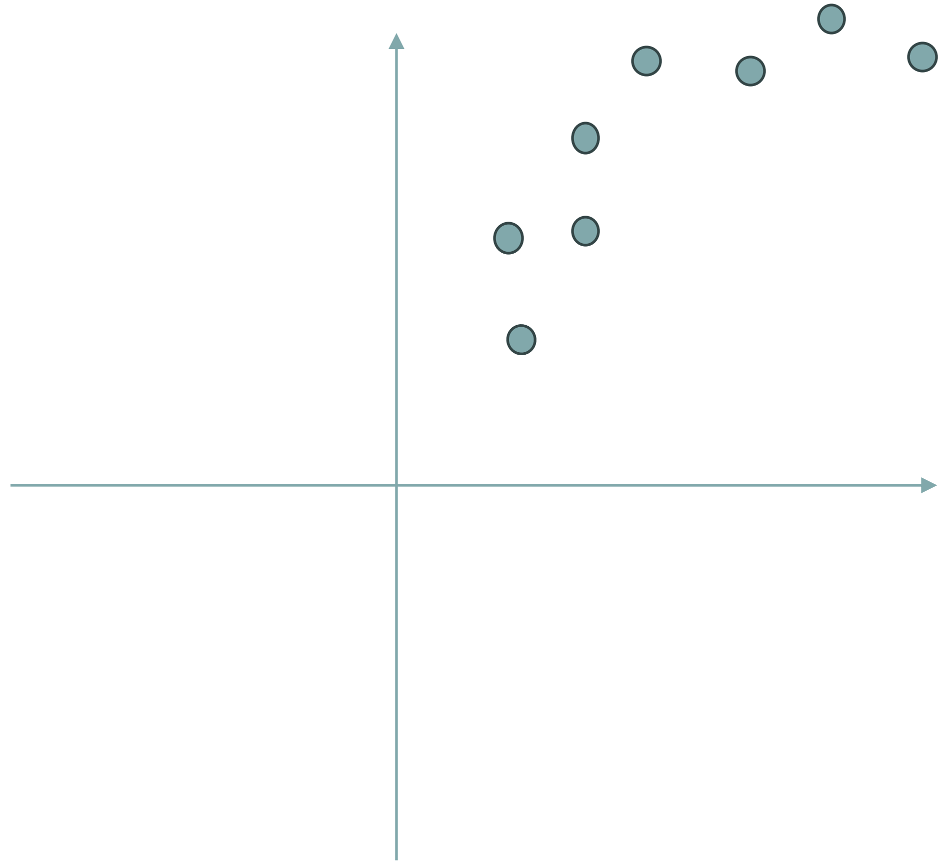


Supervised Machine Learning: an example

	Weight	Blood Pr.
Patient 1	54	112
Patient 2	68	130
Patient 3	57	115
Patient 4	56	116
Patient 5	77	132
Patient 6	81	138
Patient 7	74	130
Patient 8	66	122

$$\begin{cases} 112 = W_1 * 54 + W_0 \\ 130 = W_1 * 68 + W_0 \\ \dots \\ 122 = W_1 * 66 + W_0 \end{cases}$$

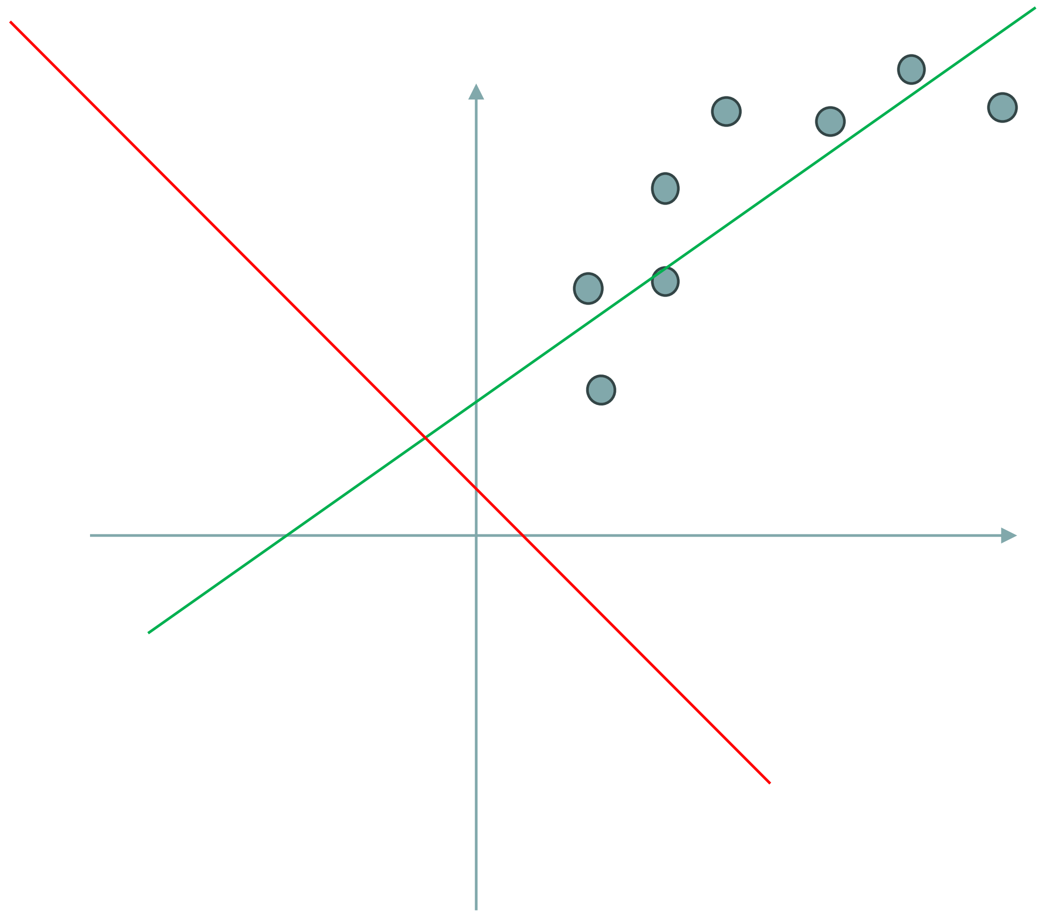
No solution, but…



Supervised Machine Learning: an example

	Weight	Blood Pr.
Patient 1	54	112
Patient 2	68	130
Patient 3	57	115
Patient 4	56	116
Patient 5	77	132
Patient 6	81	138
Patient 7	74	130
Patient 8	66	122

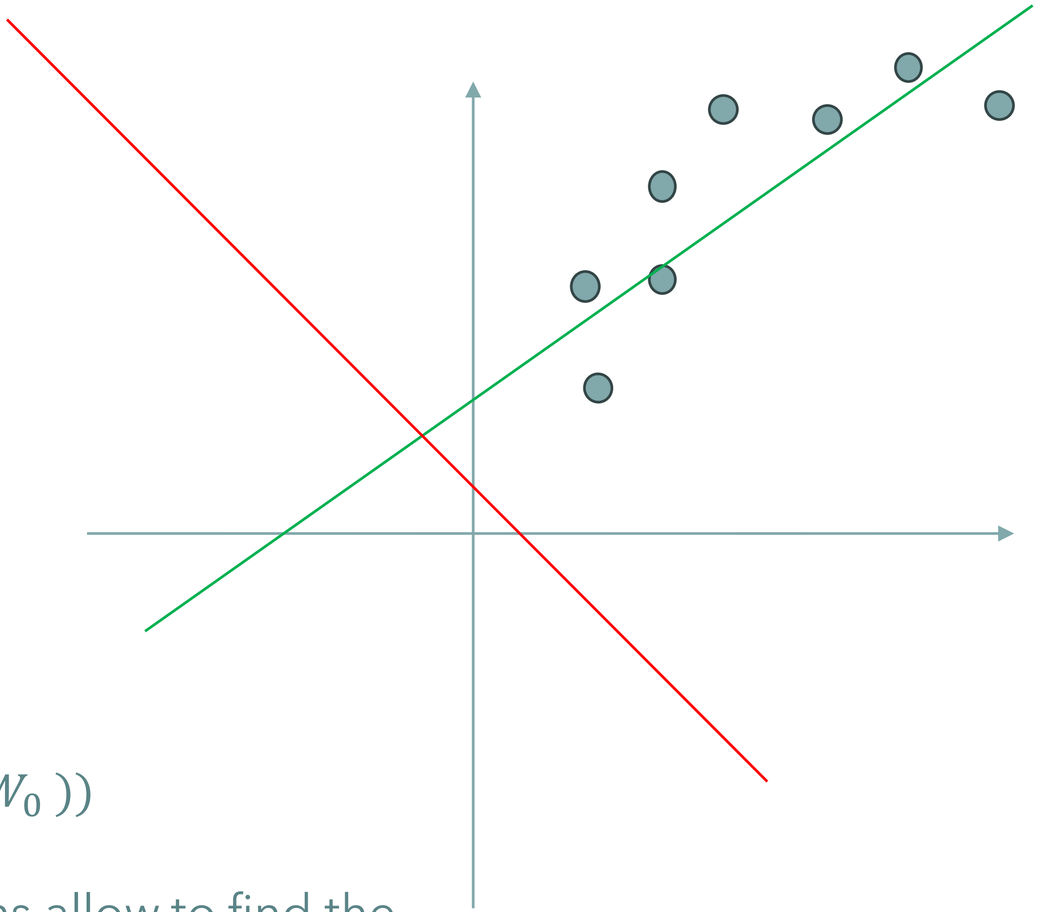
$$\left\{\begin{array}{l} 112 = W_1 * 54 + W_0 \\ 130 = W_1 * 68 + W_0 \\ \quad\ldots \\ 122 = W_1 * 66 + W_0 \end{array}\right.$$



Supervised Machine Learning: an example

	X = Weight	Y = Blood Pr.
Patient 1	54	112
Patient 2	68	130
Patient 3	57	115
Patient 4	56	116
Patient 5	77	132
Patient 6	81	138
Patient 7	74	130
Patient 8	66	122

$$\begin{cases} 112 = W_1 * 54 + W_0 \\ 130 = W_1 * 68 + W_0 \\ \dots \\ 122 = W_1 * 66 + W_0 \end{cases}$$



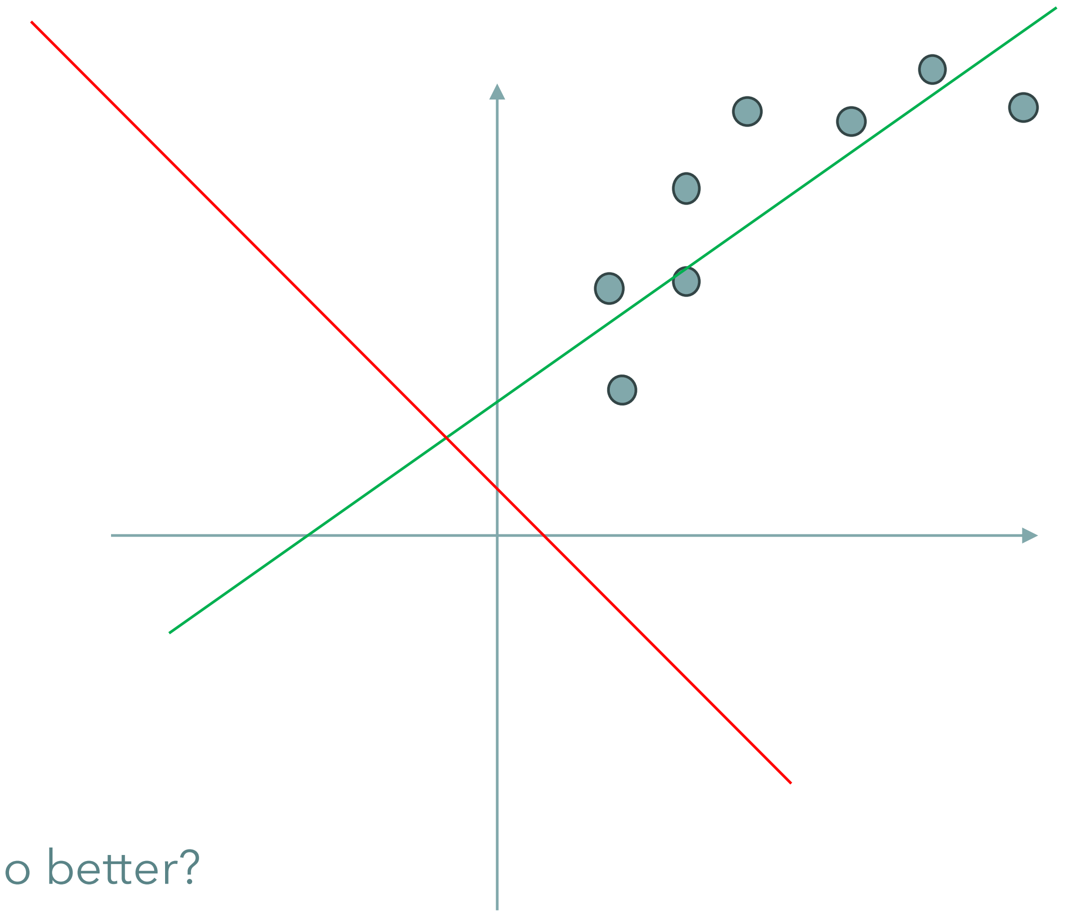
$$L = \sum_i (y_i - (W_1 X_i + W_0))$$

Some mathematical algorithms allow to find the weights that minimize such quantity, called **loss function**

Supervised Machine Learning: an example

	Weight	Blood Pr.
Patient 1	54	112
Patient 2	68	130
Patient 3	57	115
Patient 4	56	116
Patient 5	77	132
Patient 6	81	138
Patient 7	74	130
Patient 8	66	122

$$\begin{cases} 112 = W_1 * 54 + W_0 \\ 130 = W_1 * 68 + W_0 \\ \dots \\ 122 = W_1 * 66 + W_0 \end{cases}$$

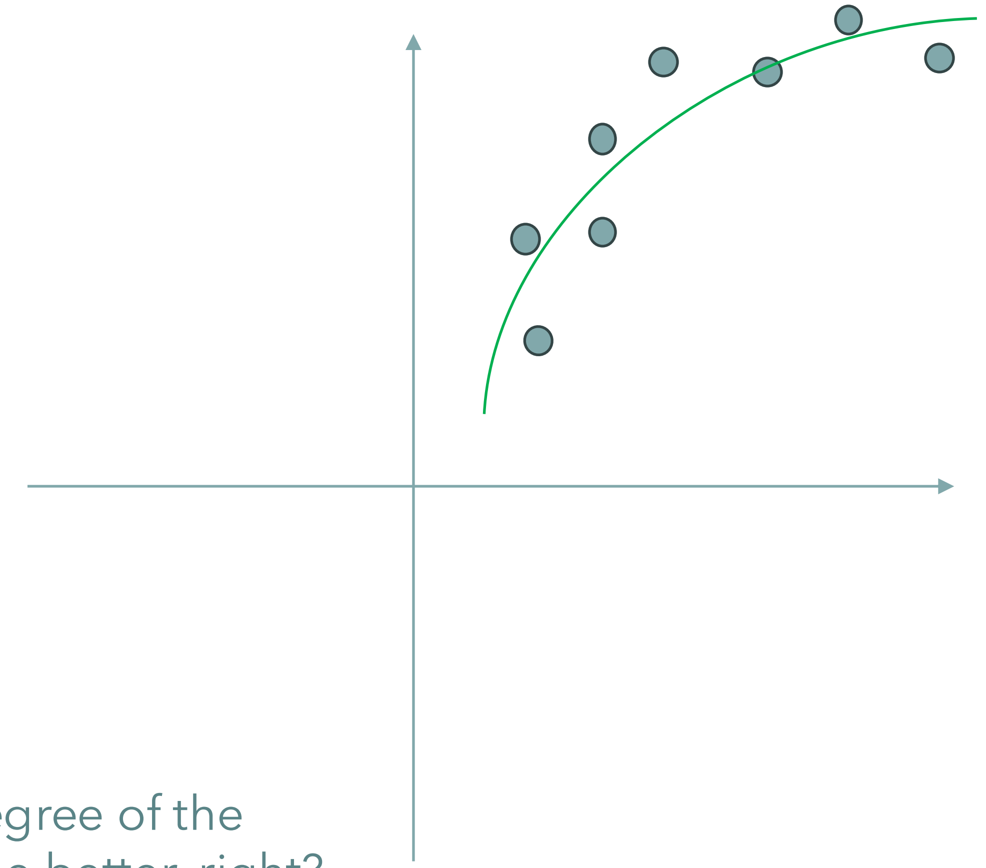


Can we do better?

Supervised Machine Learning: an example

	Weight	Weight^2	Blood Pr.
Patient 1	54	54^2	112
Patient 2	68	68^2	130
Patient 3	57	57^2	115
Patient 4	56	56^2	116
Patient 5	77	77^2	132
Patient 6	81	81^2	138
Patient 7	74	74^2	130
Patient 8	66	66^2	122

$$\left\{\begin{array}{l} 112 = W_2 * 54^2\ W_1 * 54 + W_0 \\ 130 = W_2 * 68^2\ W_1 * 68 + W_0 \\ \quad \dots \\ 122 = W_2 * 66^2\ W_1 * 66 + W_0 \end{array}\right.$$

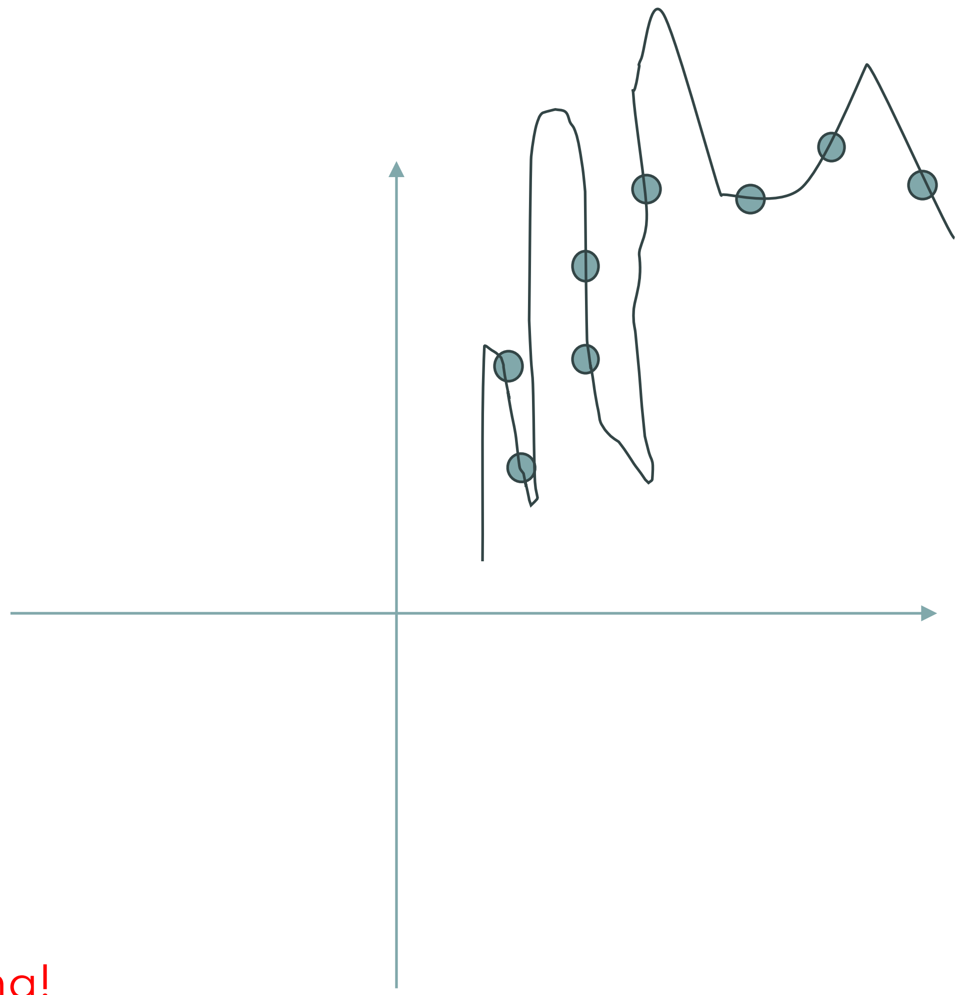


So increasing the degree of the polynomial will always be better, right?

Supervised Machine Learning: an example

	Weight	Weight^2	Blood Pr.
Patient 1	54	54^2	112
Patient 2	68	68^2	130
Patient 3	57	57^2	115
Patient 4	56	56^2	116
Patient 5	77	77^2	132
Patient 6	81	81^2	138
Patient 7	74	74^2	130
Patient 8	66	66^2	122

$$\begin{cases} 112 = W_2 * 54^2\ W_1 * 54 + W_0 \\ 130 = W_2 * 68^2\ W_1 * 68 + W_0 \\ \quad\ldots \\ 122 = W_2 * 66^2\ W_1 * 66 + W_0 \end{cases}$$



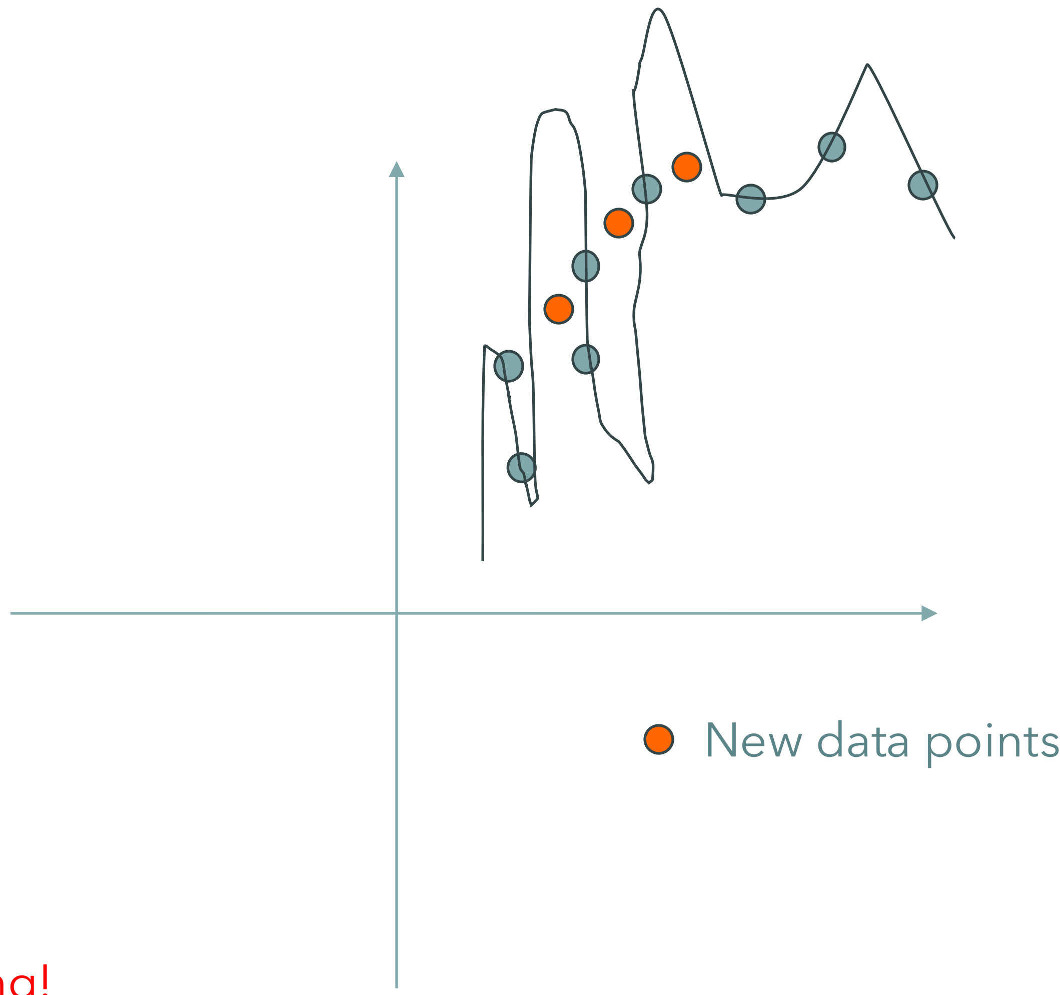
NO, overfitting!

Supervised Machine Learning: an example

	Weight	Weight^2	Blood Pr.
Patient 1	54	54^2	112
Patient 2	68	68^2	130
Patient 3	57	57^2	115
Patient 4	56	56^2	116
Patient 5	77	77^2	132
Patient 6	81	81^2	138
Patient 7	74	74^2	130
Patient 8	66	66^2	122

$$\begin{cases} 112 = W_2 * 54^2\ W_1 * 54 + W_0 \\ 130 = W_2 * 68^2\ W_1 * 68 + W_0 \\ \quad\dots \\ 122 = W_2 * 66^2\ W_1 * 66 + W_0 \end{cases}$$

NO, overfitting!

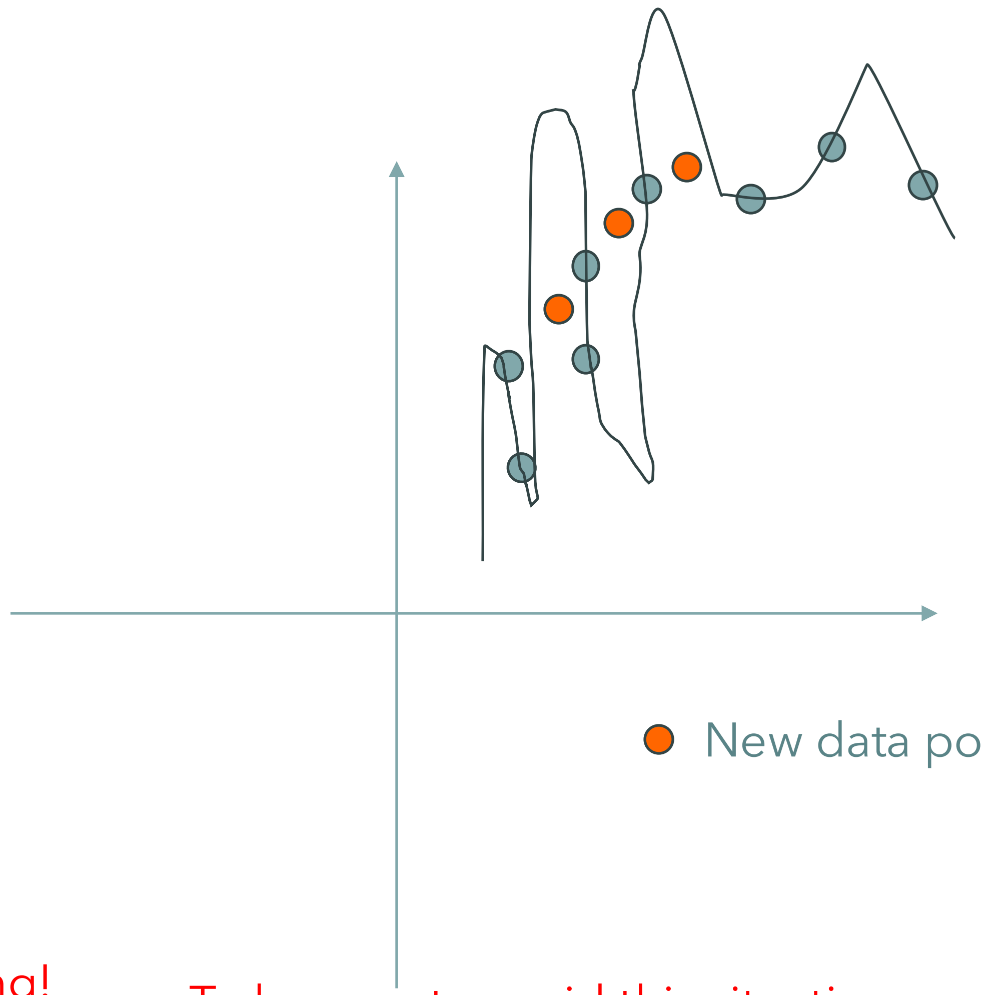


Supervised Machine Learning: an example

	Weight	Weight^2	Blood Pr.
Patient 1	54	54^2	112
Patient 2	68	68^2	130
Patient 3	57	57^2	115
Patient 4	56	56^2	116
Patient 5	77	77^2	132
Patient 6	81	81^2	138
Patient 7	74	74^2	130
Patient 8	66	66^2	122

$$\begin{cases} 112 = W_2 * 54^2\ W_1 * 54 + W_0 \\ 130 = W_2 * 68^2\ W_1 * 68 + W_0 \\ \dots \\ 122 = W_2 * 66^2\ W_1 * 66 + W_0 \end{cases}$$

NO, overfitting!



● New data points

To be sure to avoid this situation, we always need to take some data apart to validate our model

Some important rules

- Always split your dataset to be able to evaluate the model and control overfitting. Usually, for ML tasks, you use 70% of the data for the training, 10% for the validation, and 20% for the test set. More complex strategies to better utilize your data are **K-Fold Cross Validation** or **Leave One Out**

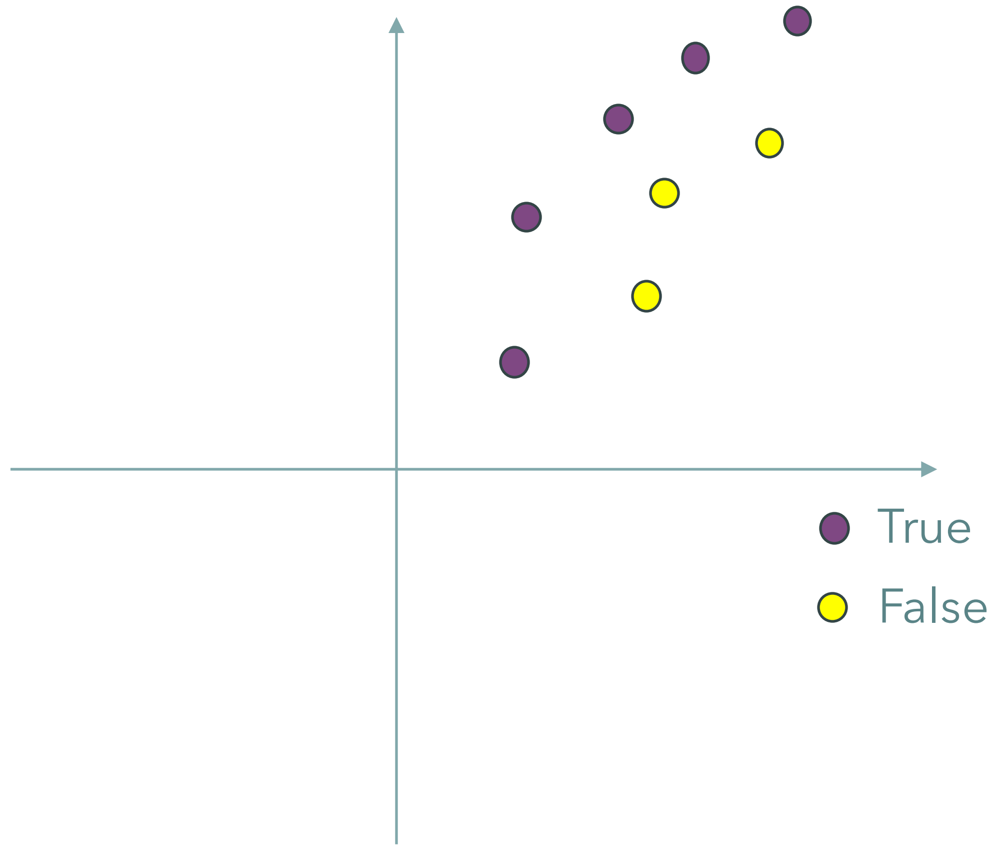
- To avoid overfitting you use **regularization techniques**; some models in the online libraries such as Scikit Learn use regularization techniques by default.

- The validation set is used to choose the best **hyperparameters** (for instance, the grade of the polynomial in the simple example before). In complex models you have a huge number of parameters and you have tu carefully choose them.

- Be careful to the difference between **parameters** and **hyperparameters**. The parameters are learnt from the model (the coefficients W_i of the previous linear regression), the hyperparameters determine the model's shape and complexity, and they are decided by the developer (the degree of the polynomial).

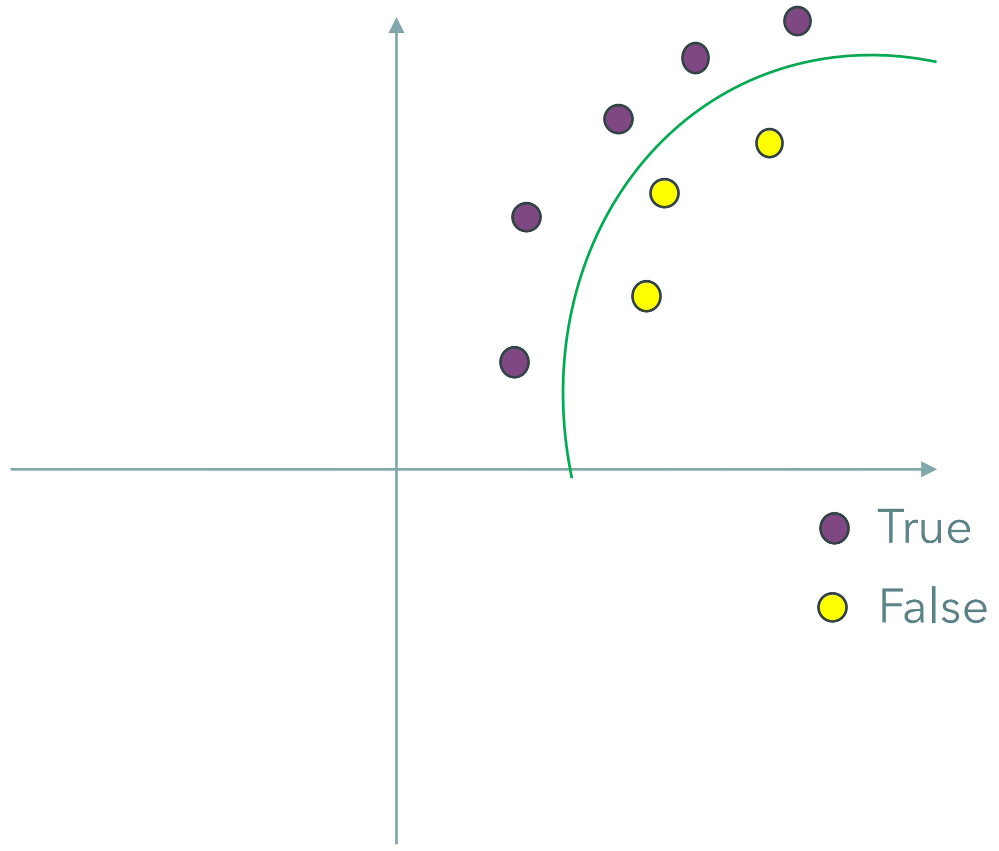
Supervised Machine Learning: classification

	Weight	Weight^2	Pathology
Patient 1	54	54^2	True
Patient 2	68	68^2	False
Patient 3	57	57^2	False
Patient 4	56	56^2	False
Patient 5	77	77^2	True
Patient 6	81	81^2	False
Patient 7	74	74^2	False
Patient 8	66	66^2	True



Supervised Machine Learning: classification

	Weight	Blood Pr	Pathology
Patient 1	54	54^2	True
Patient 2	68	68^2	False
Patient 3	57	57^2	False
Patient 4	56	56^2	False
Patient 5	77	77^2	True
Patient 6	81	81^2	False
Patient 7	74	74^2	False
Patient 8	66	66^2	True



Most famous ML models that work with tabular data

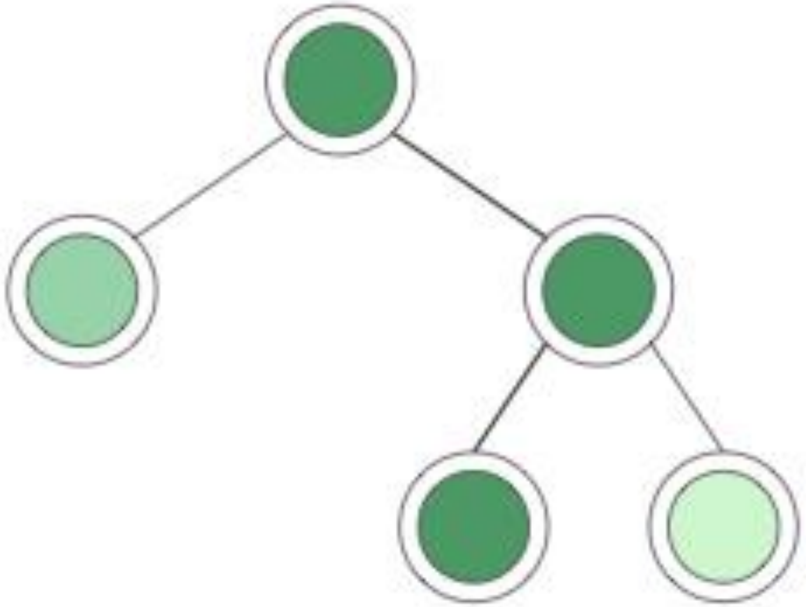
Regression

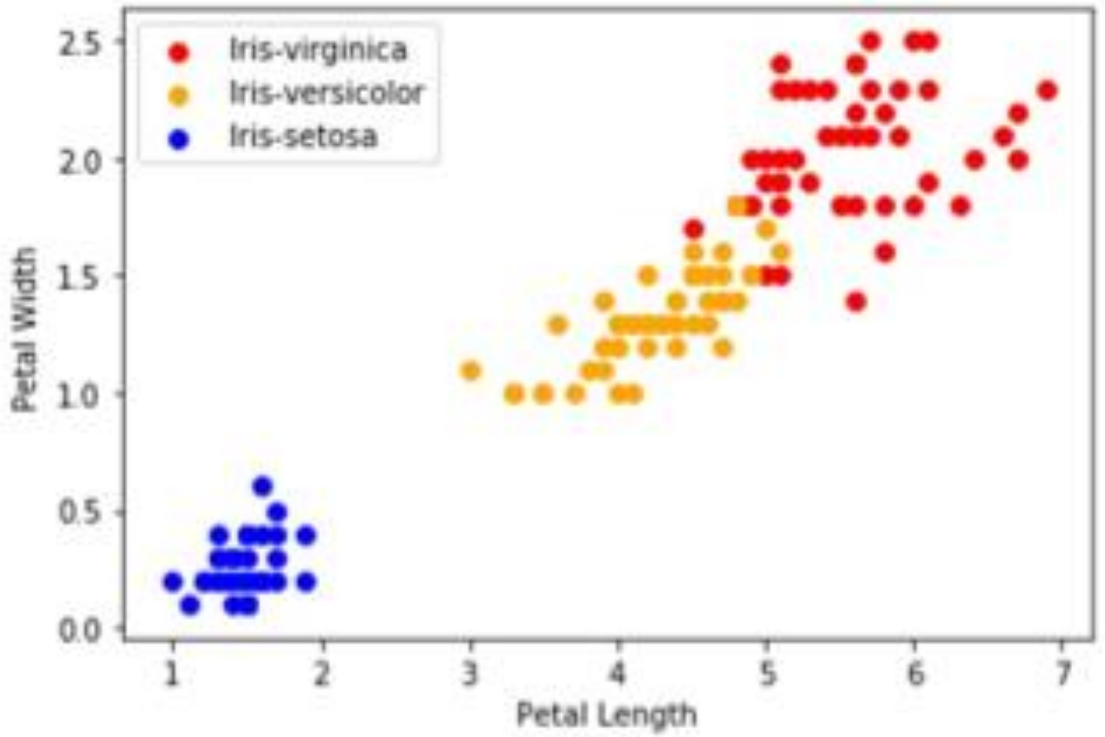
- Linear regression, and all its regularized variants (Ridge regression, Lasso regression, ecc)
- Decision Tree based models and ensembles (Random Forests, Xgboost, ecc)
- K-Nearest-neighbours

Classification

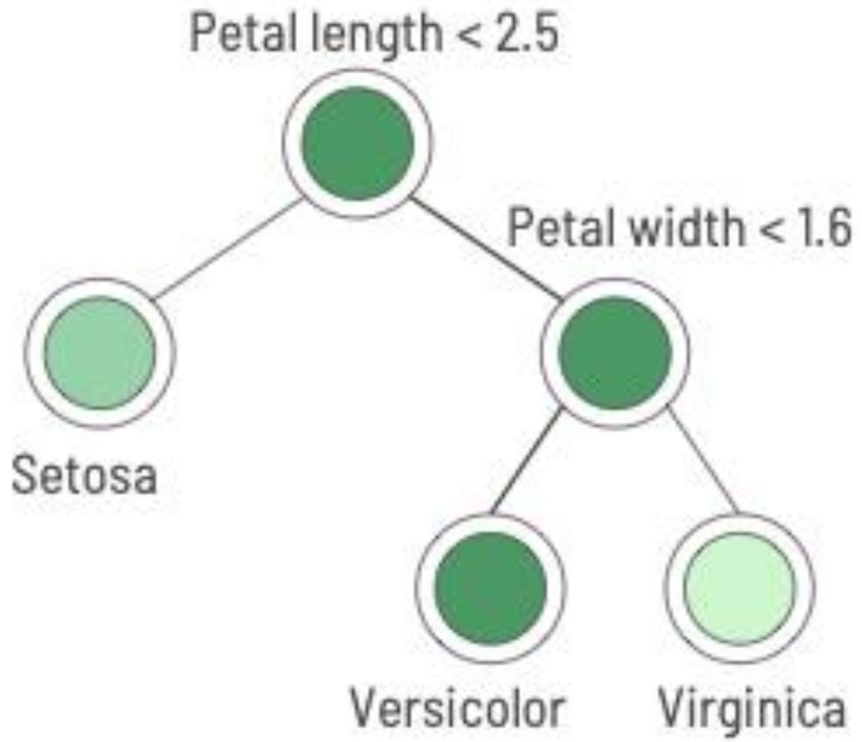
- Logistic regression
- Decision Tree based models and ensembles (Random Forests, Xgboost, ecc)
- K-Nearest-neighbours
- Support Vector Machines
- Naive Bayes

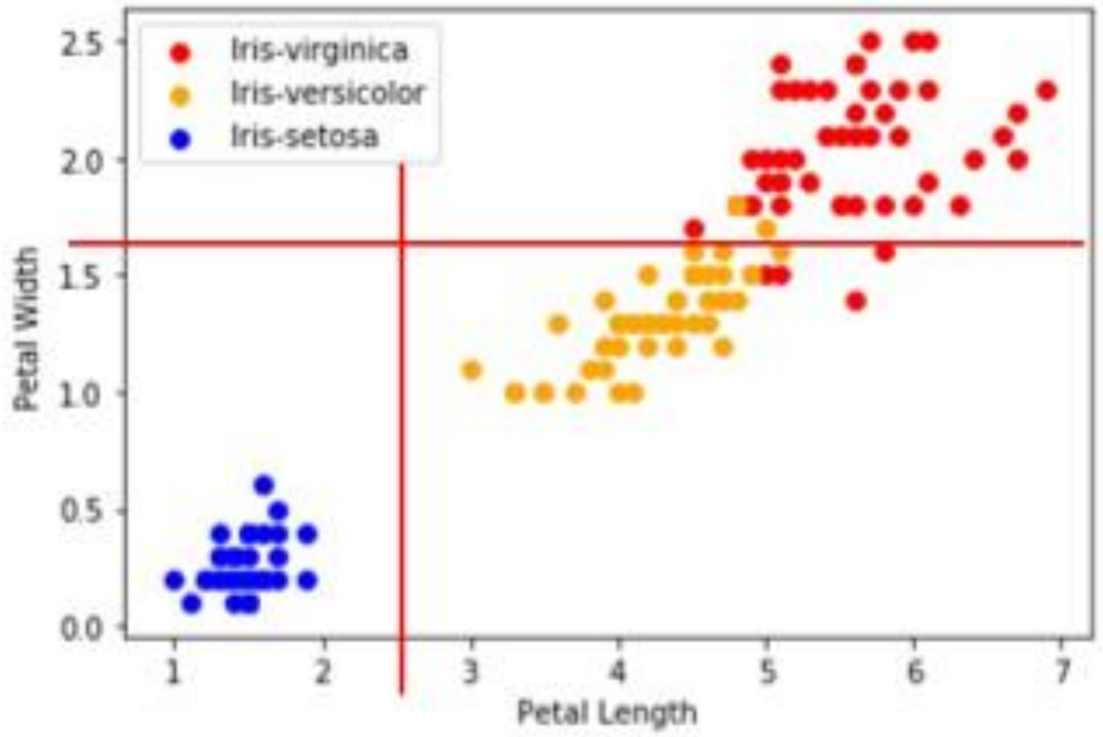
Decision Trees



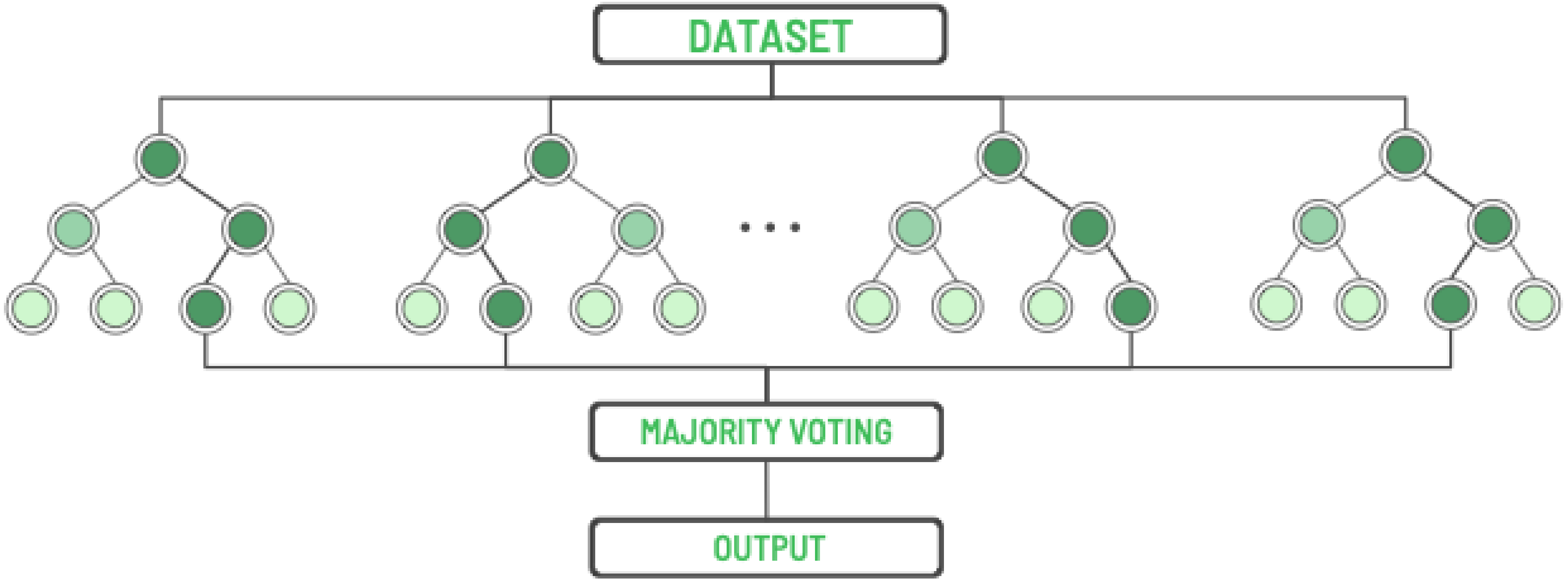


Decision Trees

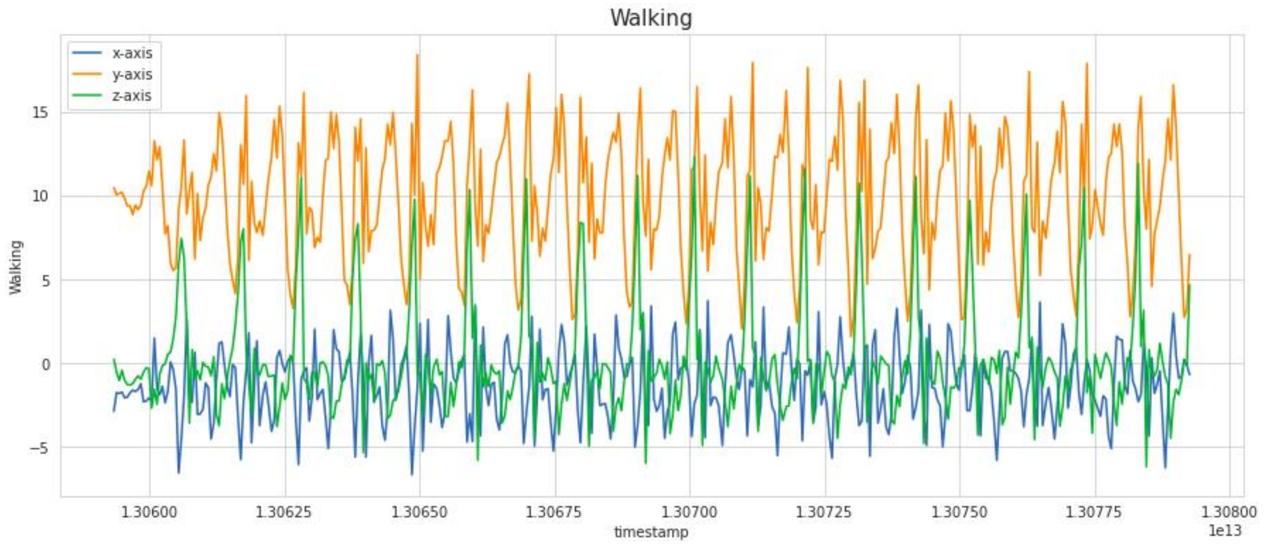




Random Forests



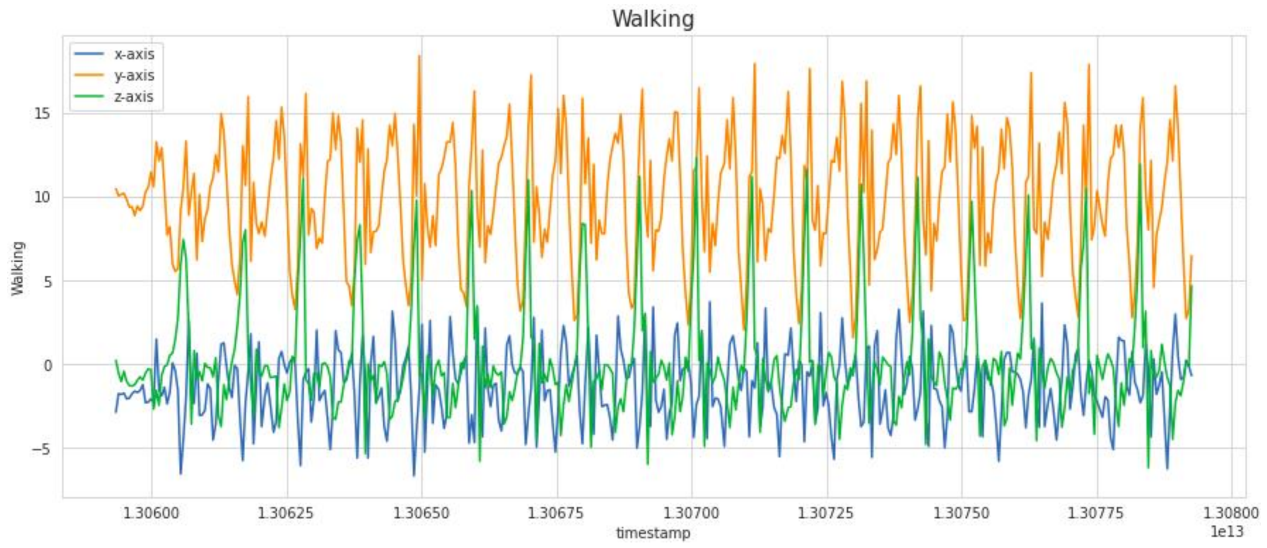
How can we use this models for dealing with Time series?



How can we use this models for dealing with Time series?

1st alternative

- Extract features: Mean, Standard Deviation, etc…
- Build a tabular dataset
- Apply the previously mentioned models

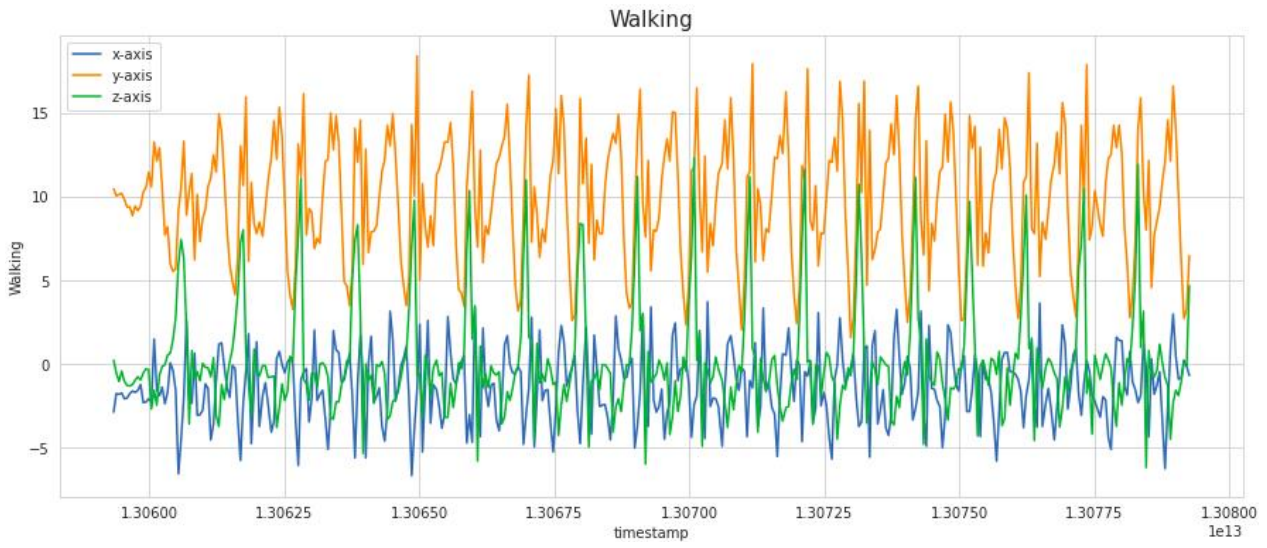


How can we use this models for dealing with Time series?

1st alternative

- Extract features: Mean, Standard Deviation, etc…
- Build a tabular dataset
- Apply the previously mentioned models

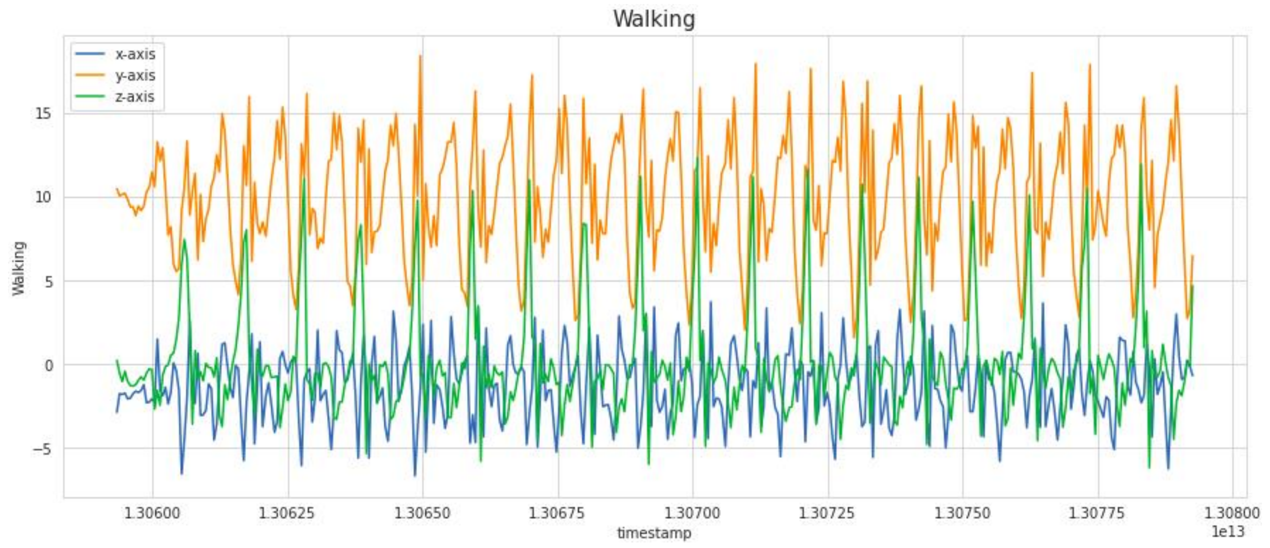
Coding time



How can we use this models for dealing with Time series?

1st alternative

- Extract features: Mean, Standard Deviation, etc…
- Build a tabular dataset
- Apply the previously mentioned models



2nd alternative

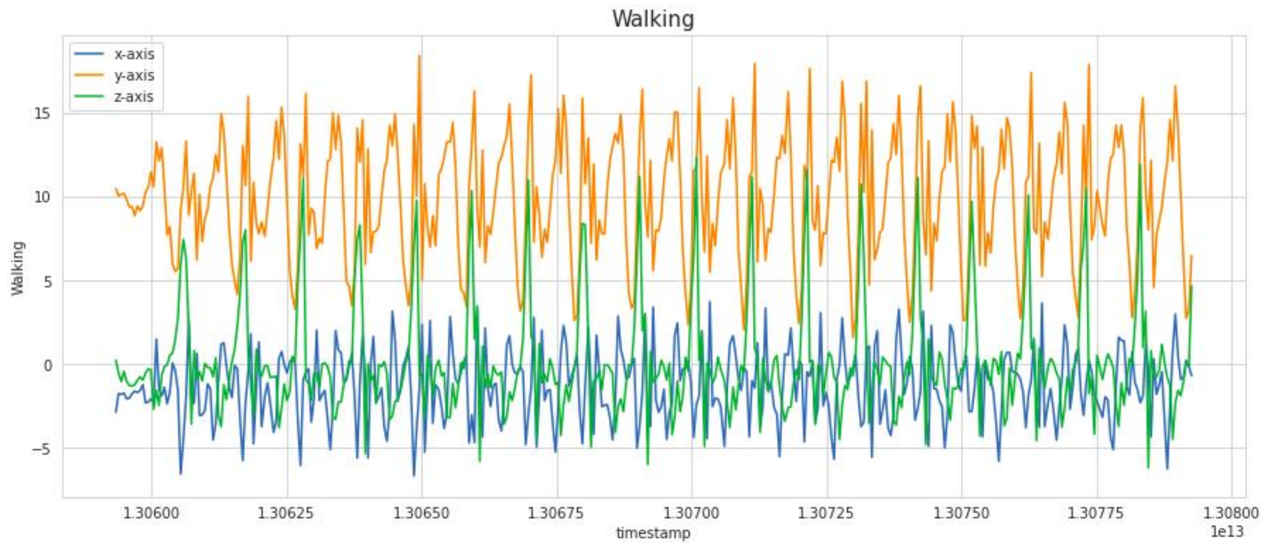
- Use models that directly take as input the data in the format of a time series.
- Which models can do this?

POLITECNICO MILANO 1863

How can we use this models for dealing with Time series?

1st alternative

- Extract features: Mean, Standard Deviation, etc…
- Build a tabular dataset
- Apply the previously mentioned models

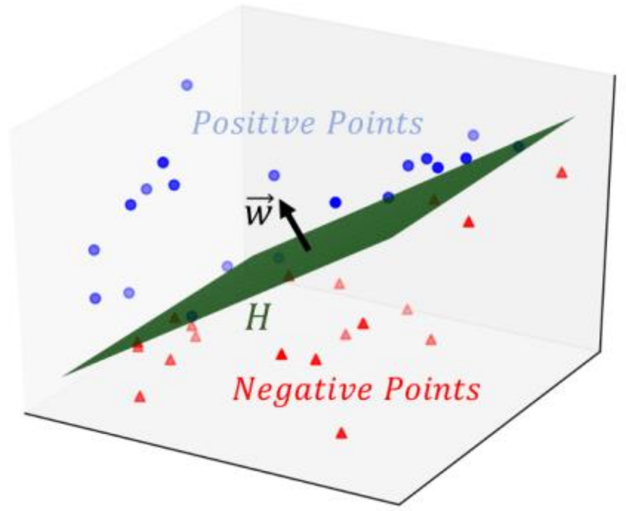


2nd alternative

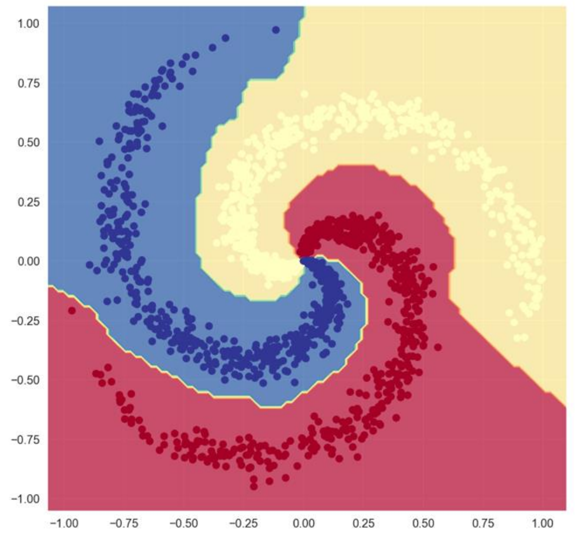
- Use models that directly take as input the data in the format of a time series.
- Which models can do this? **Neural Networks**

Supervised Machine Learning: Neural networks

All the models we have seen until now are linear in the feature space.

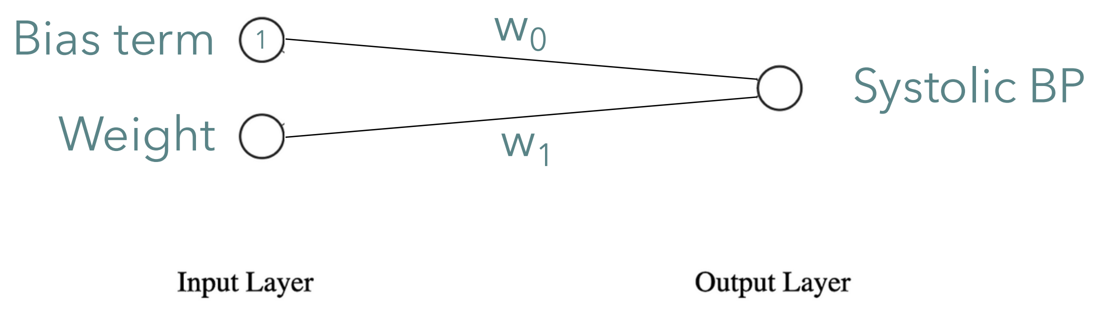


The main characteristic of Neural Networks is the introduction of Non-linearities



Supervised Machine Learning: Neural networks

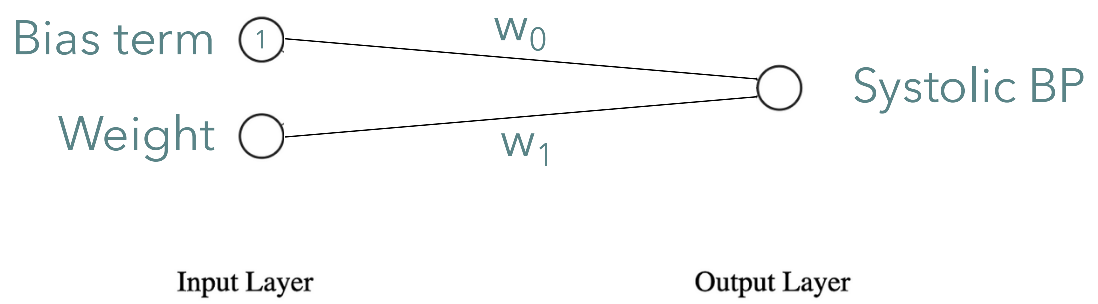
How is this linearity introduced?



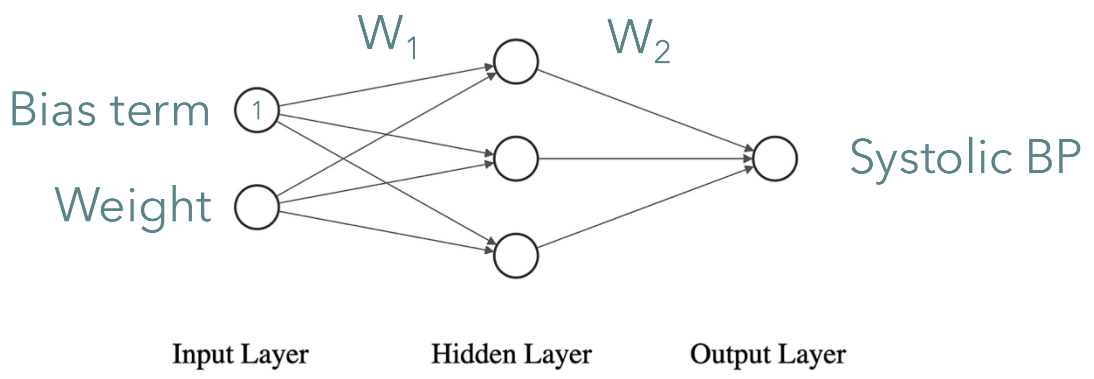
Classical Linear Regression

Supervised Machine Learning: Neural networks

How is this linearity introduced?



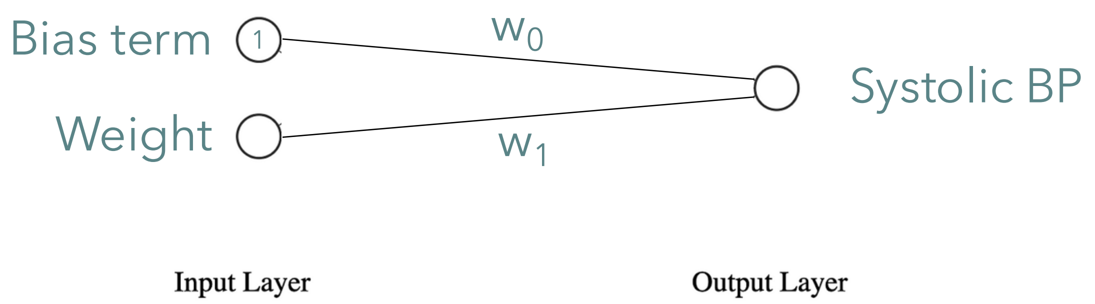
Classical Linear Regression



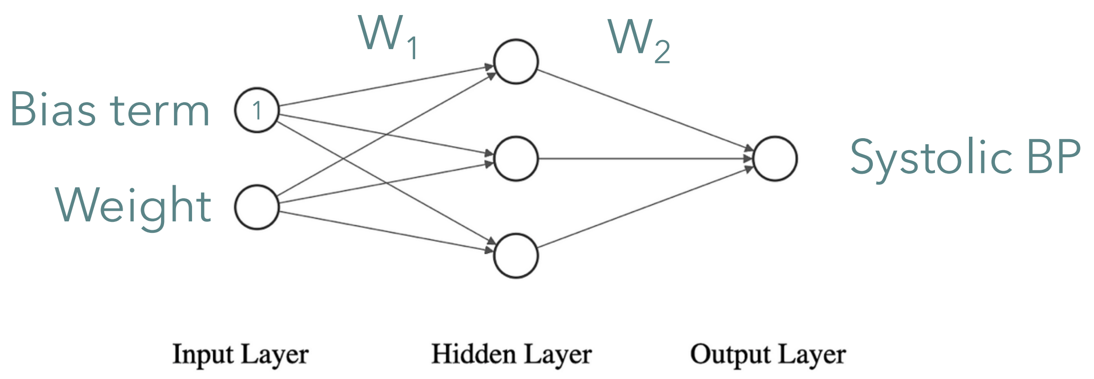
Non-linear?

Supervised Machine Learning: Neural networks

How is this linearity introduced?



Classical Linear Regression



Non-linear?

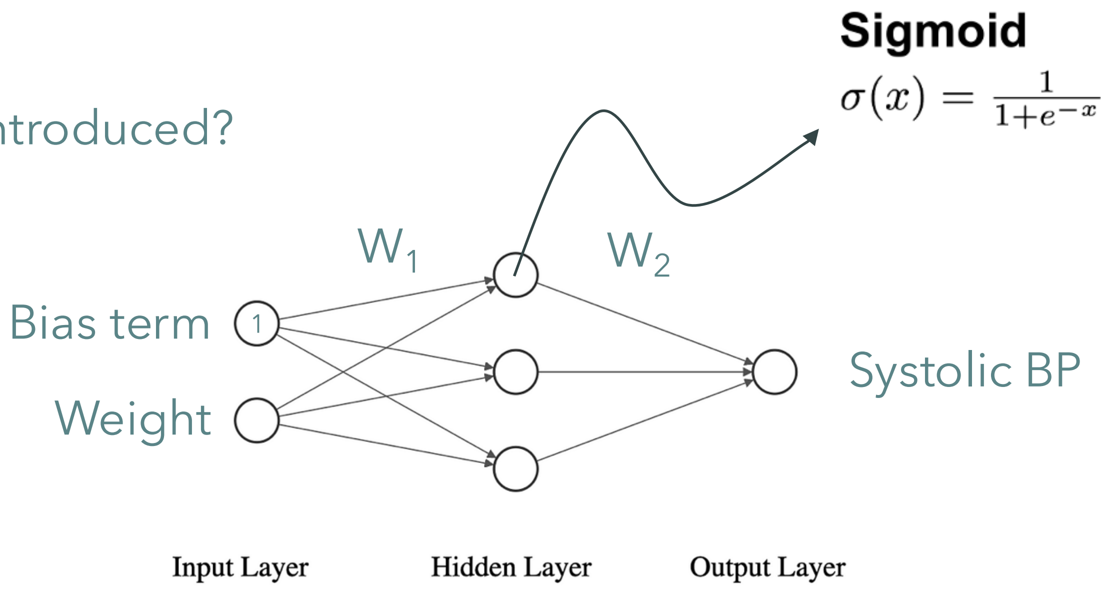
No!
W_1 is a 2x3 matrix
W_2 is a 3x1 matrix

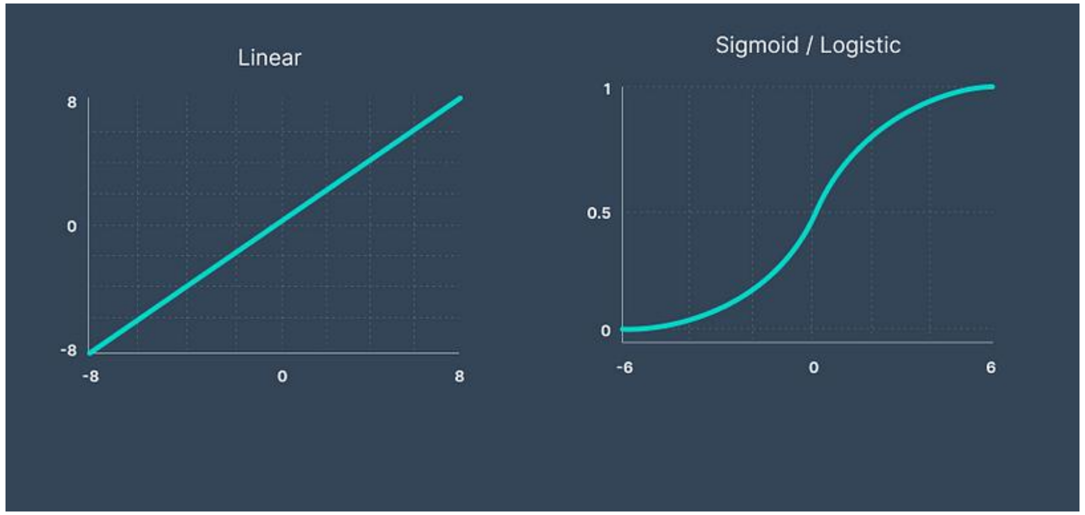
In the end, the result is just a 2x1 matrix, just the two same parameter of the classical linear regression

Supervised Machine Learning: Neural networks

How is this linearity introduced?

The **nonlinearity** comes from a new element that is introduced in the hidden neurons, called **activation function**

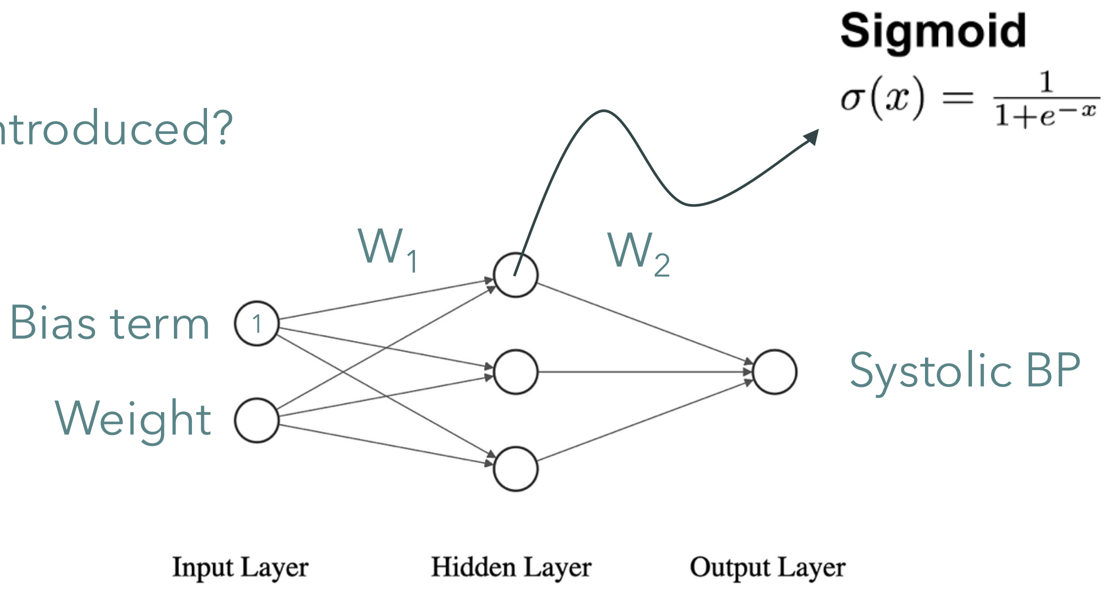


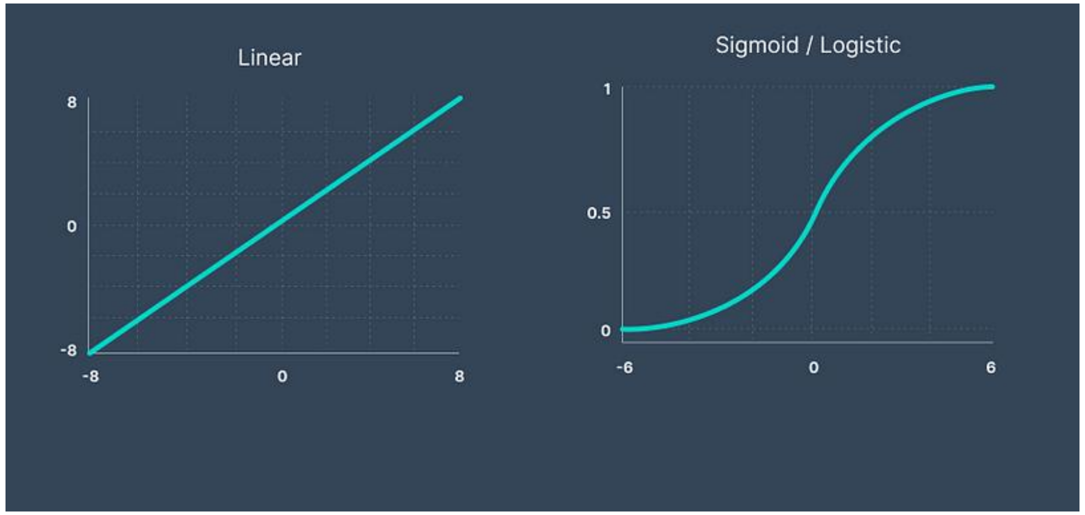


Supervised Machine Learning: Neural networks

How is this linearity introduced?

The **nonlinearity** comes from a new element that is introduced in the hidden neurons, called **activation function**





Neural networks can learn **complex and nonlinear patterns**.
However, you have to be careful: they tend to **overfit** much easier and, therefore, need a lot of data and careful regularization techniques to work well.

Best Deep Learning models for time series

Many variants of classical Neural Networks exist to deal with complex and structured data (such as images, graphs, or time series). These are the most suitable for the latter:

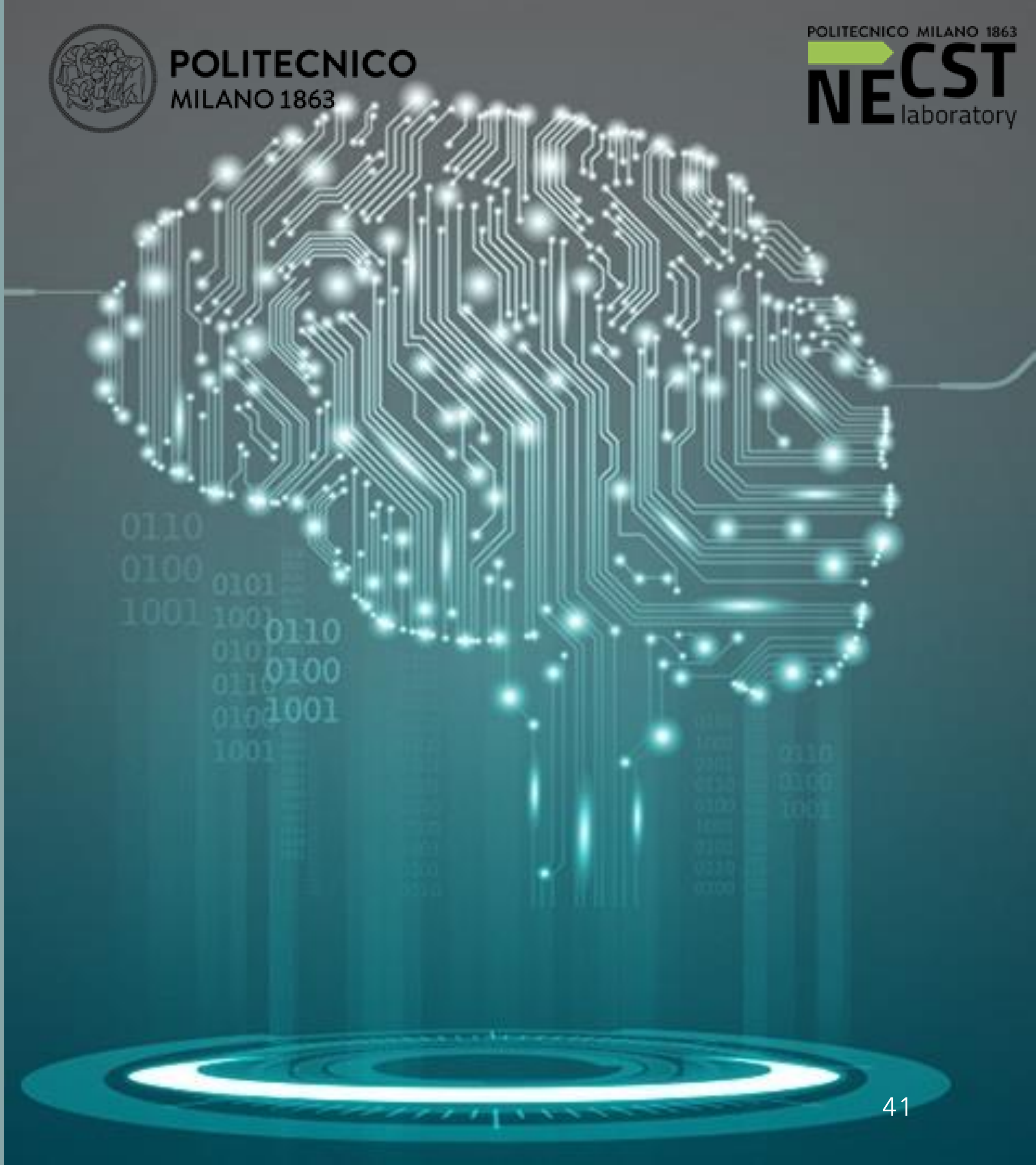
- Recurrent Neural Networks (RNN)
- 1D Convolutional Neural Networks (1D CNN)
- Long-Short Time Memory (LSTM)
- Bidirectional LSTM
- Gated Recurrent Units (GRU)

Creativity, Science and **Innovation**

Thank you for your attention

November 24th, 2025

Alessandro Verosimile
alessandro.verosimile@polimi.it



Creativity, Science and **Innovation**

Thank you for your attention

Coding time

November 24th, 2025

Alessandro Verosimile
alessandro.verosimile@polimi.it