

Big data science Day 2

F. Legger - INFN Torino

<https://github.com/leggerf/MLCourse-2021>



Yesterday

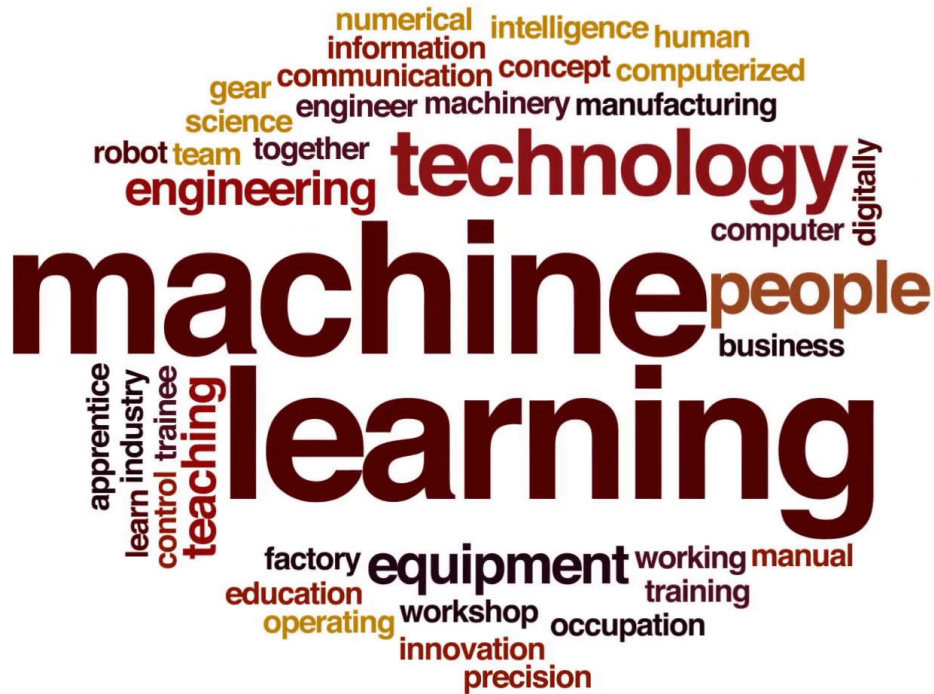
- Big data
- Analytics

Today

- Machine learning

Next

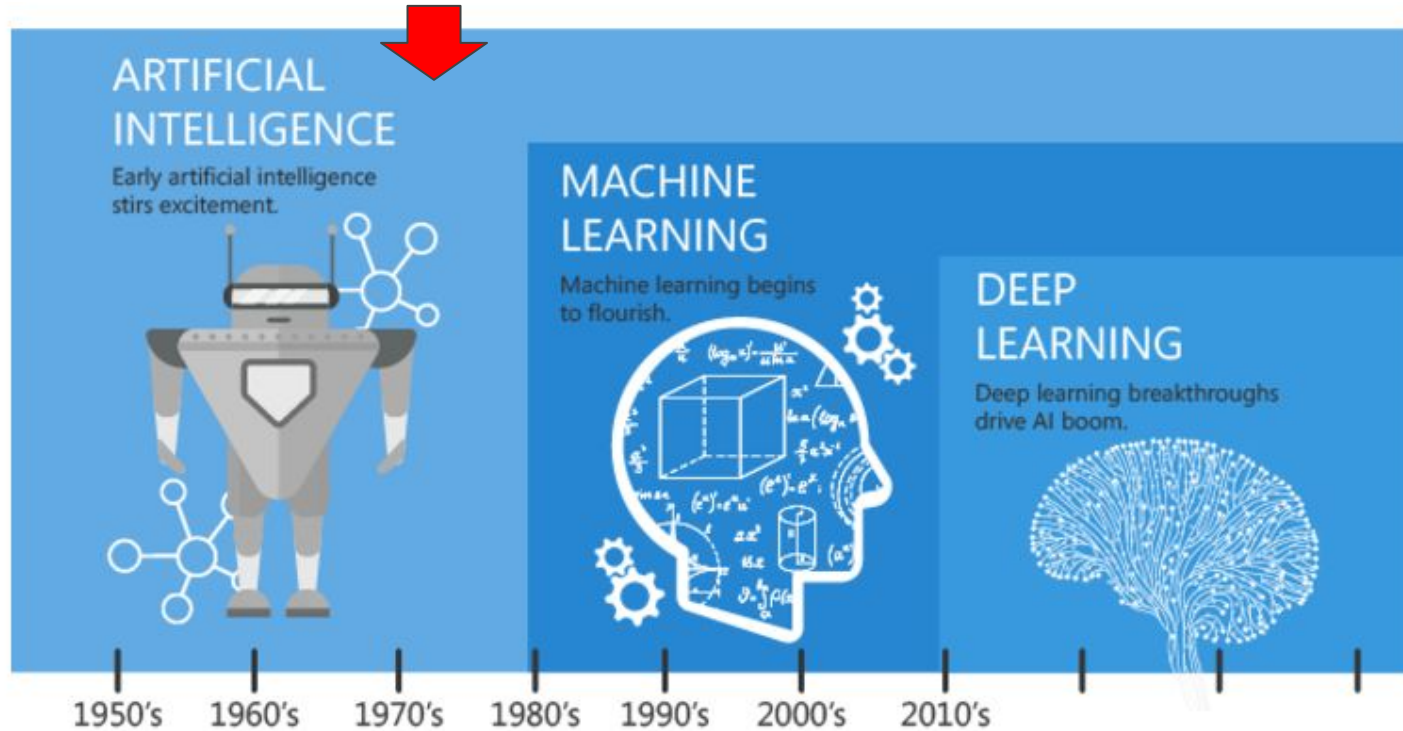
- Deep learning
- Parallelisation
- Heterogeneous architectures



Hands on

Apache MLlib library and Apache Spark

Our ultimate objective is to make programs that learn from their experience as effectively as humans do
[John McCarthy, 1958]



Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.

Machine Learning

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead

[Wikipedia]

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at task in T , as measured by P , improves with experience E

[Tom Mitchell, 1997]

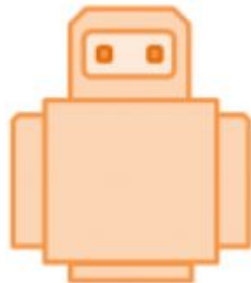
Machine Learning is the science of getting computers to act without being explicitly programmed

[Andrew Ng]

Machine Learning

Input Data

Information (+ Answers)

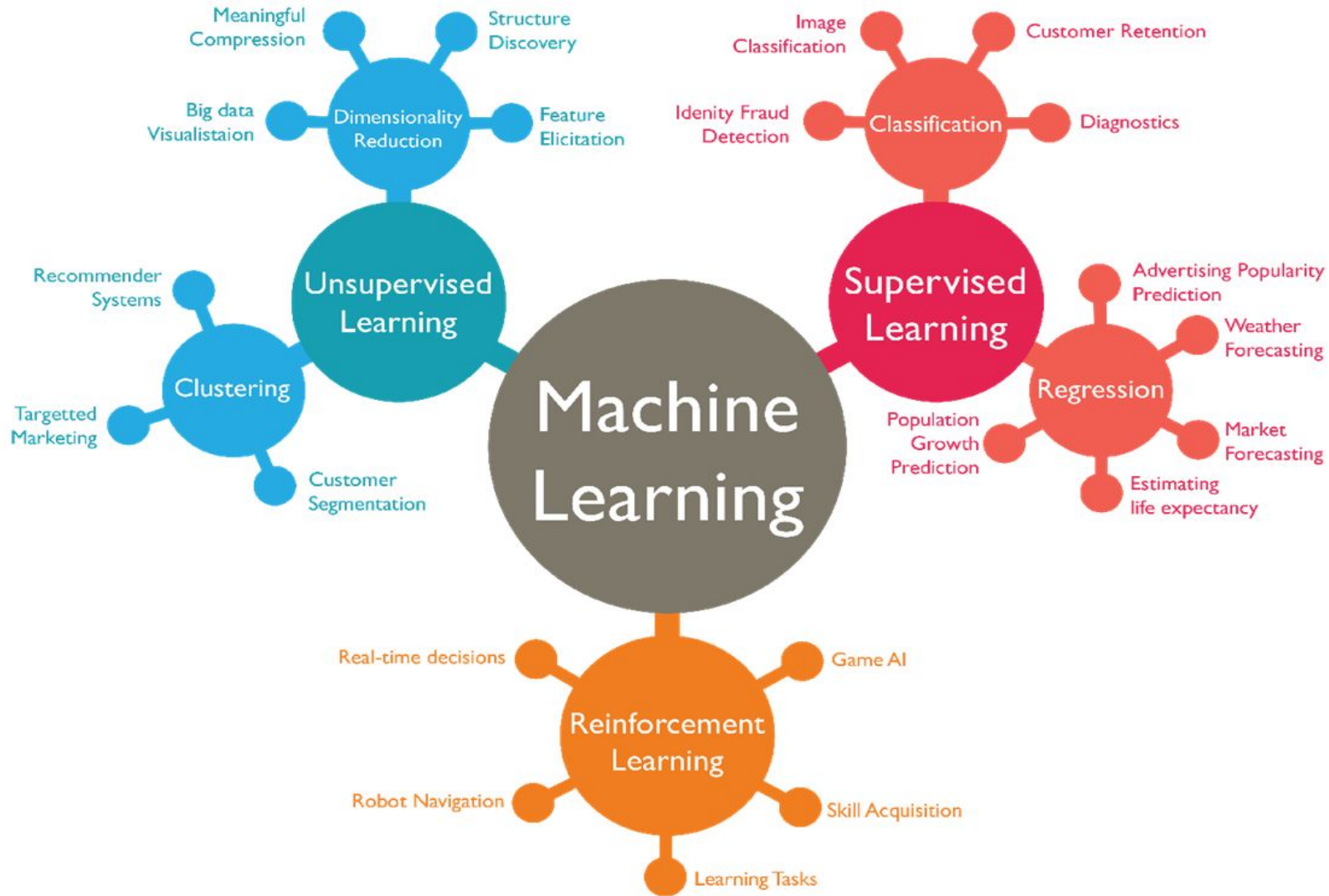


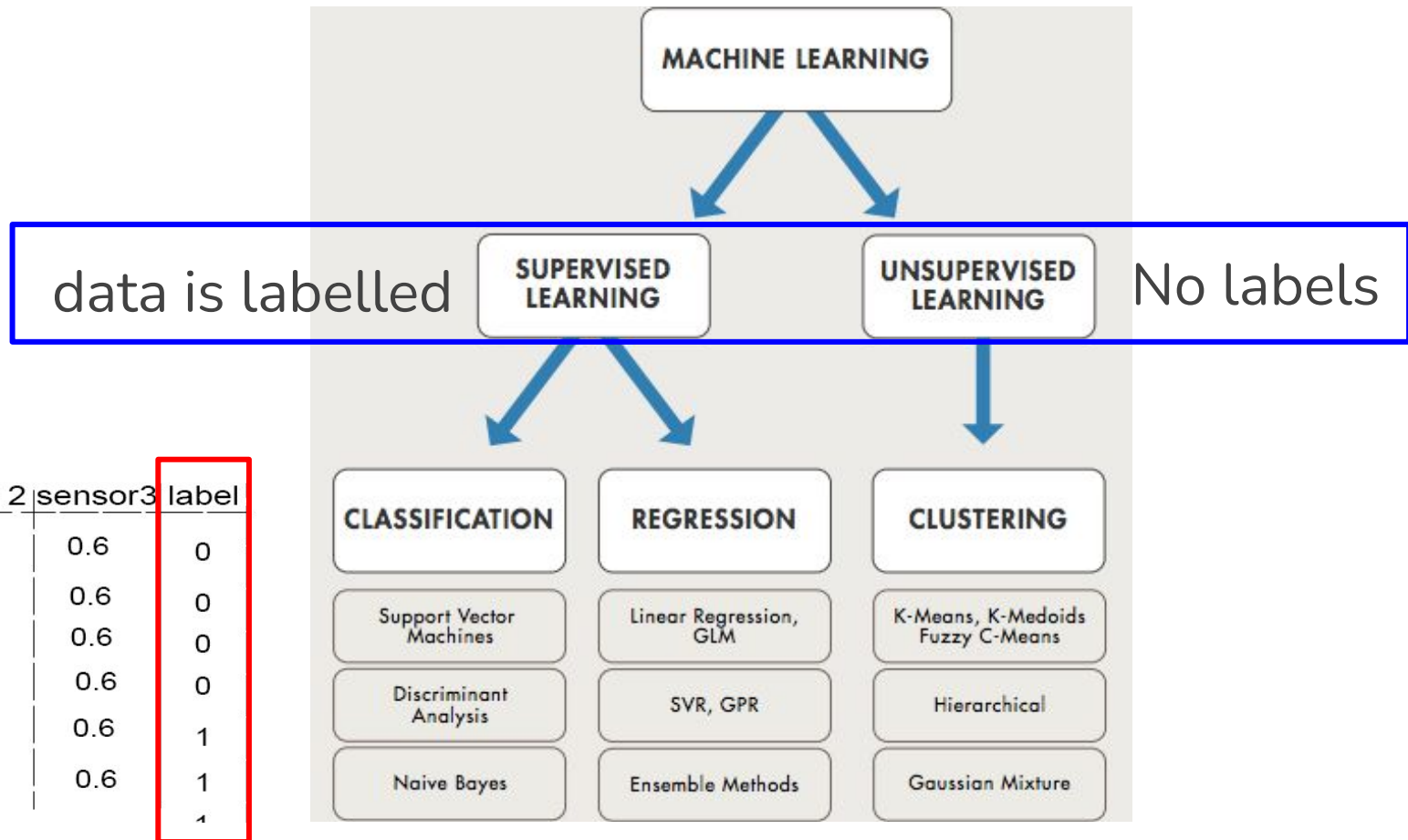
Output

Optimum Model

- Relationships
- Patterns
- Dependencies
- Hidden structures

Algorithms + Techniques



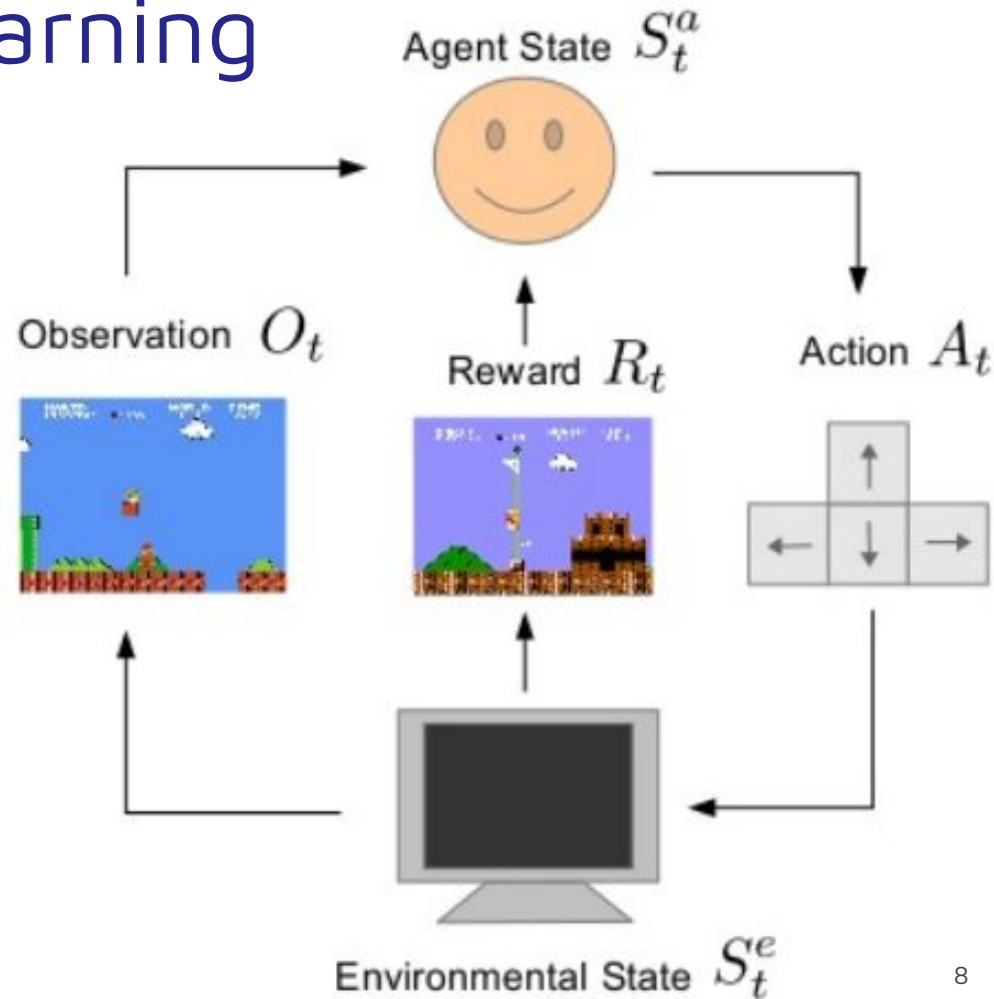


Discrete
labels

Continuous
labels

Reinforcement learning

- getting an agent to act in the world so as to maximize its rewards
- sparse and time delayed labels (**rewards**)



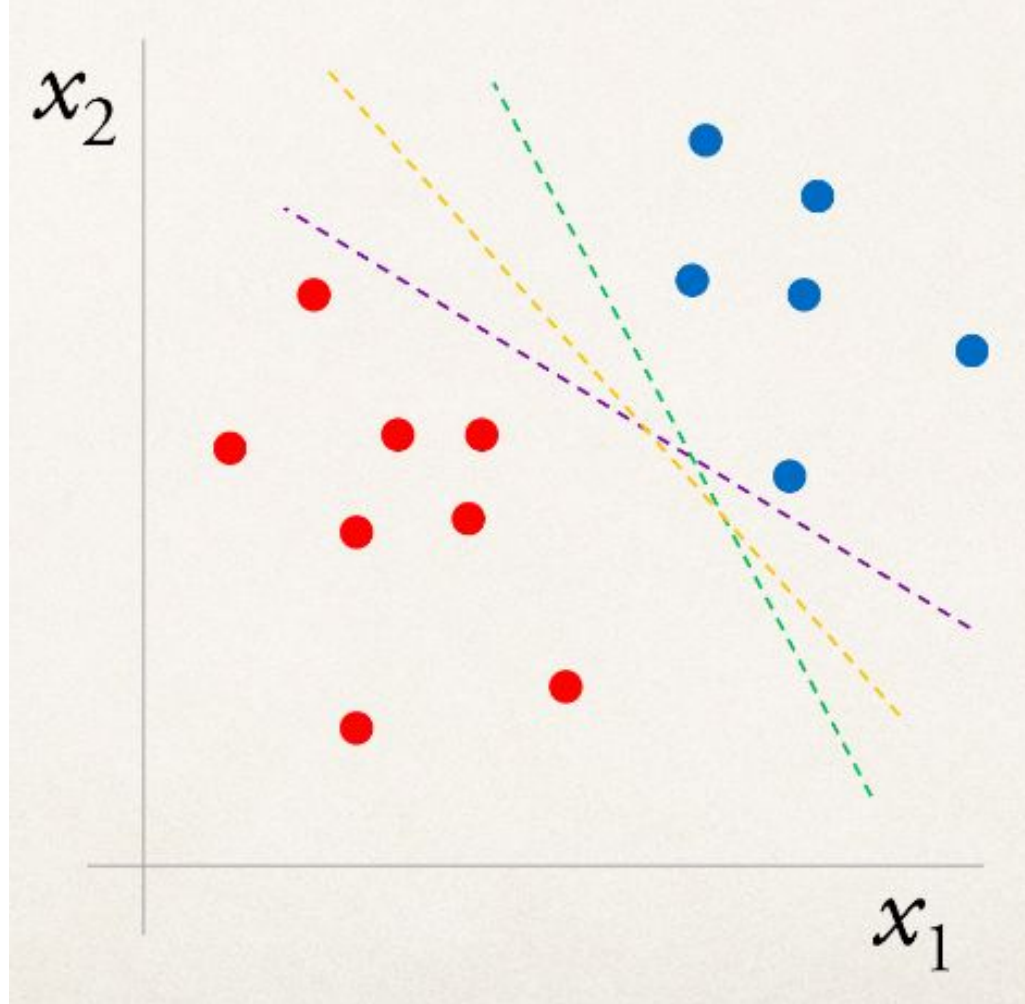
Classification

- Businesses who target customers: good vs bad, stay or leave
- **Signal vs background**
-



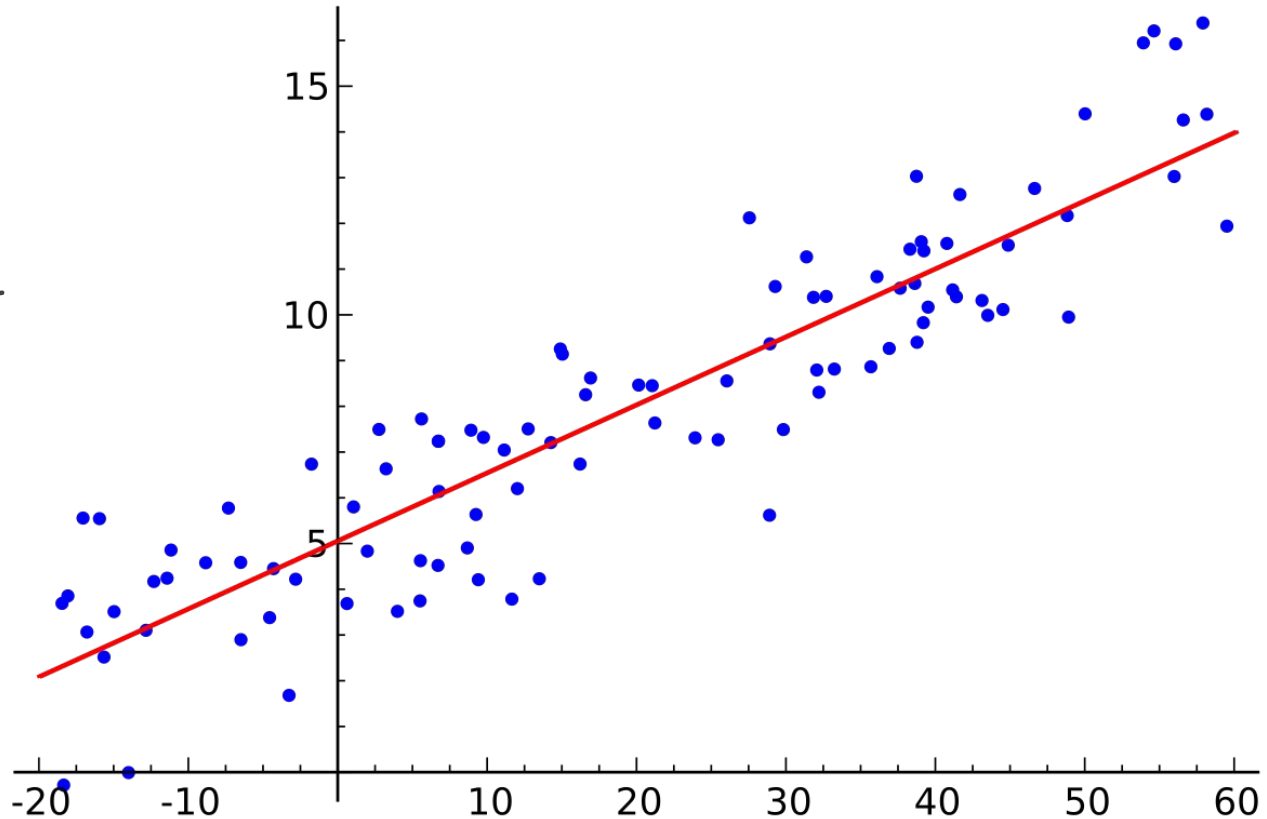
hands-on

Supervised, discrete labels



Regression

- Businesses who predict customer behavior: e.g. house prices, ...

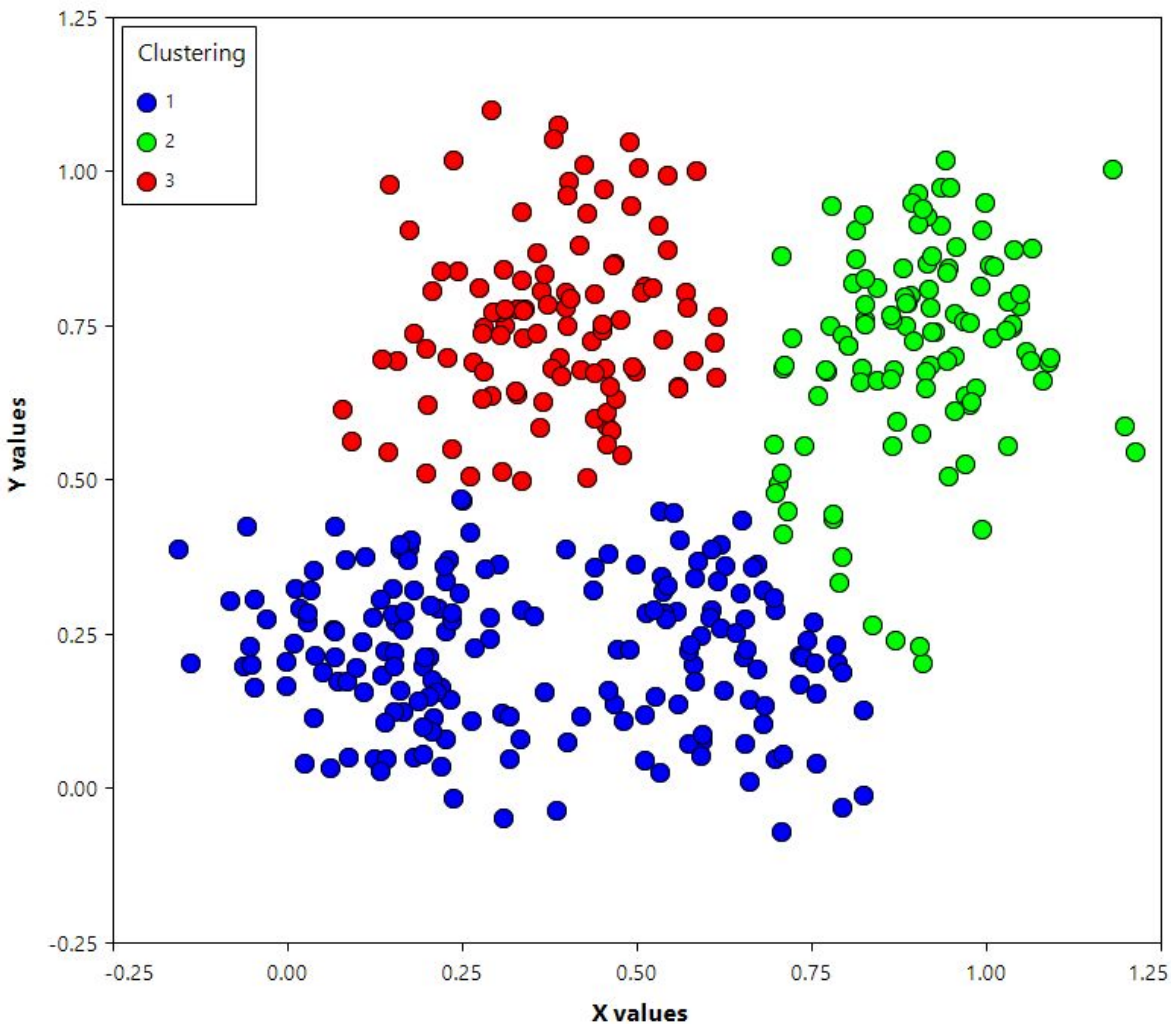


Supervised, continuous labels

Clustering

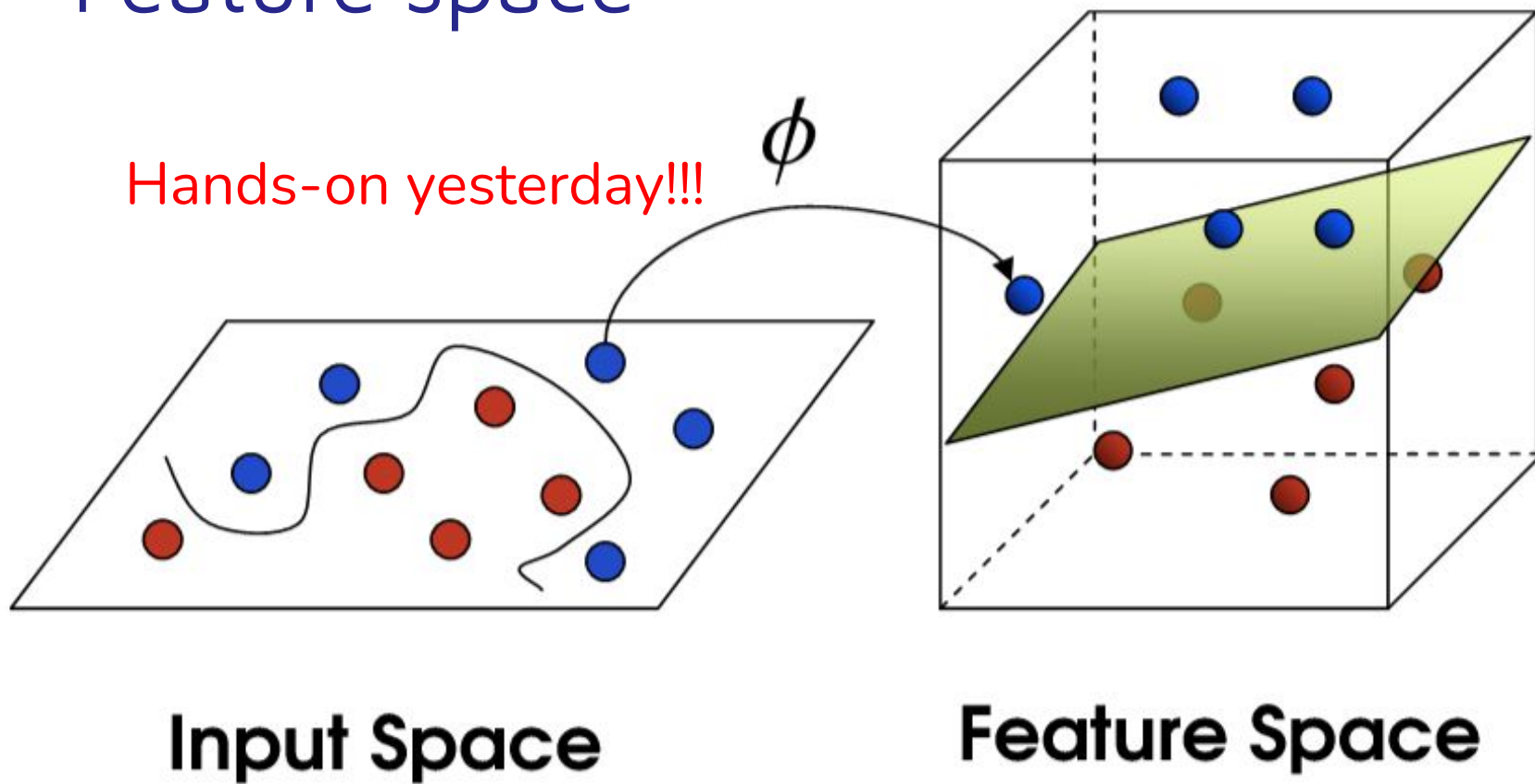
- Businesses who identify customer categories
- Light vs heavy flavour jets
-

Unsupervised



Feature space

Hands-on yesterday!!!



Feature engineering

Raw Data

```
0: {  
  house_info: {  
    num_rooms: 6  
    num_bedrooms: 3  
    street_name: "Shorebird Way"  
    num_basement_rooms: -1  
    ...  
  }  
}
```

Raw data doesn't come to us as feature vectors.

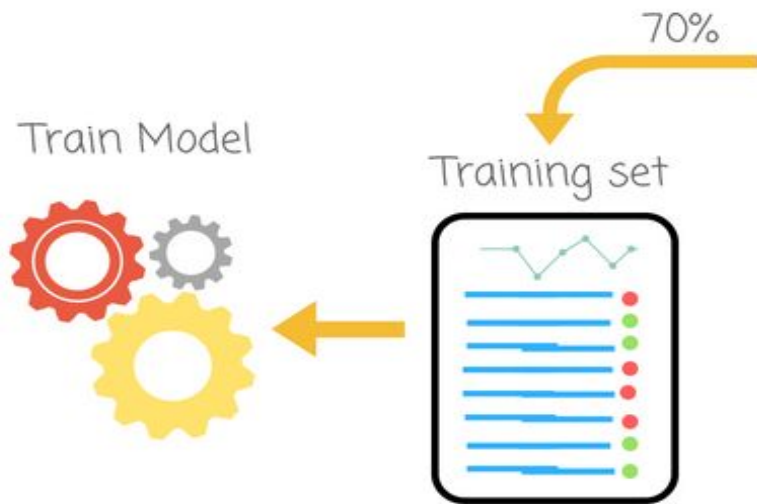
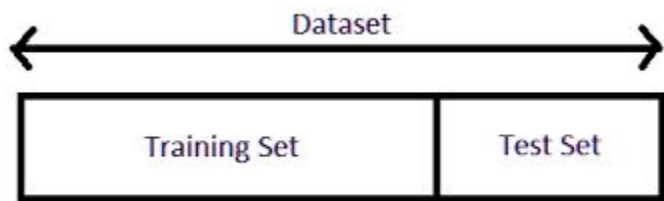
Feature Engineering

Feature Vector

```
[  
  6.0,  
  1.0,  
  0.0,  
  0.0,  
  0.0,  
  9.321,  
  -2.20,  
  1.01,  
  0.0,  
  ...  
]
```

Process of creating features from raw data is **feature engineering**.

Training and test set



Entire Dataset



30%

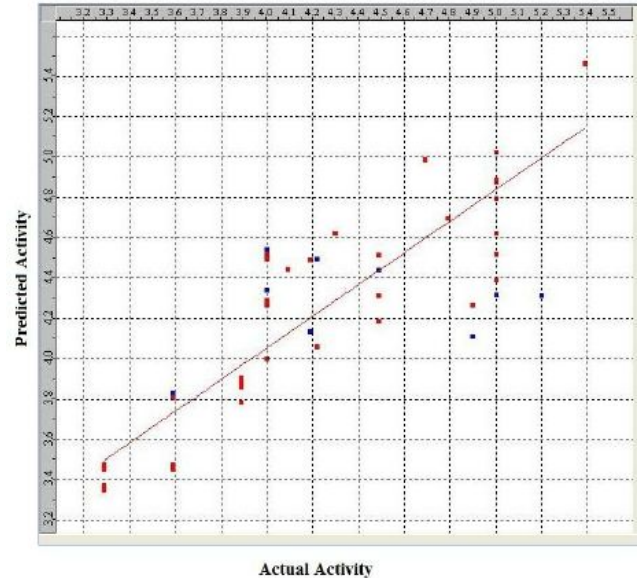


Test set

Test labels



Used later for testing



Example: supervised classification

Ingredients

- **Inputs:** X , e.g. timestamp, price, color, size, etc.
- **Features:** X , transformed inputs
- **Labels:** y
- **Training** and **test** datasets

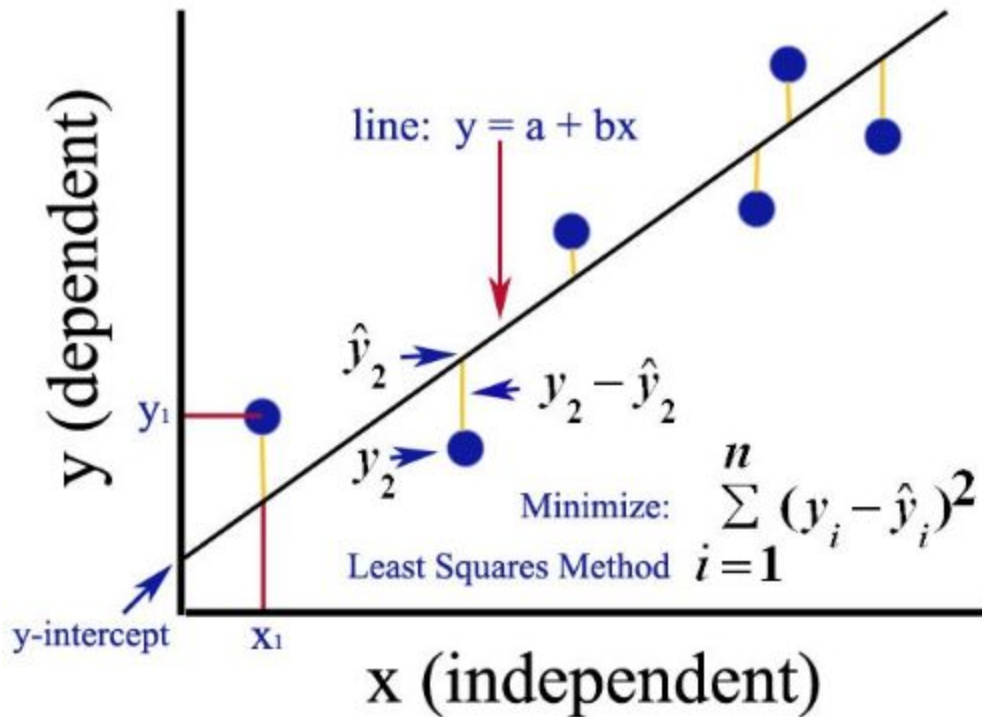
Recipe

- **Weights:** W (matrix) parameters to be found by the model
- **Activation function:** ϕ (step function, e.g. sigmoid)
- **Predictions:** $z = \phi(W^T X)$ yields $(0,1)$
- **Cost function == loss function == prediction error:** $J(W)$, e.g. $\sum (y_i - z_i)^2 / 2$
- **Aim:** find weights W that minimize cost function & give best separation



Another example, linear regression

- Inputs (features): x_i
- Labels: y_i
- Model: $y = a + bx$
- Weight+bias (parameters to be found): a, b
- Cost function: **Mean Square Error (MSE)**
- No **activation function**: problem is linear

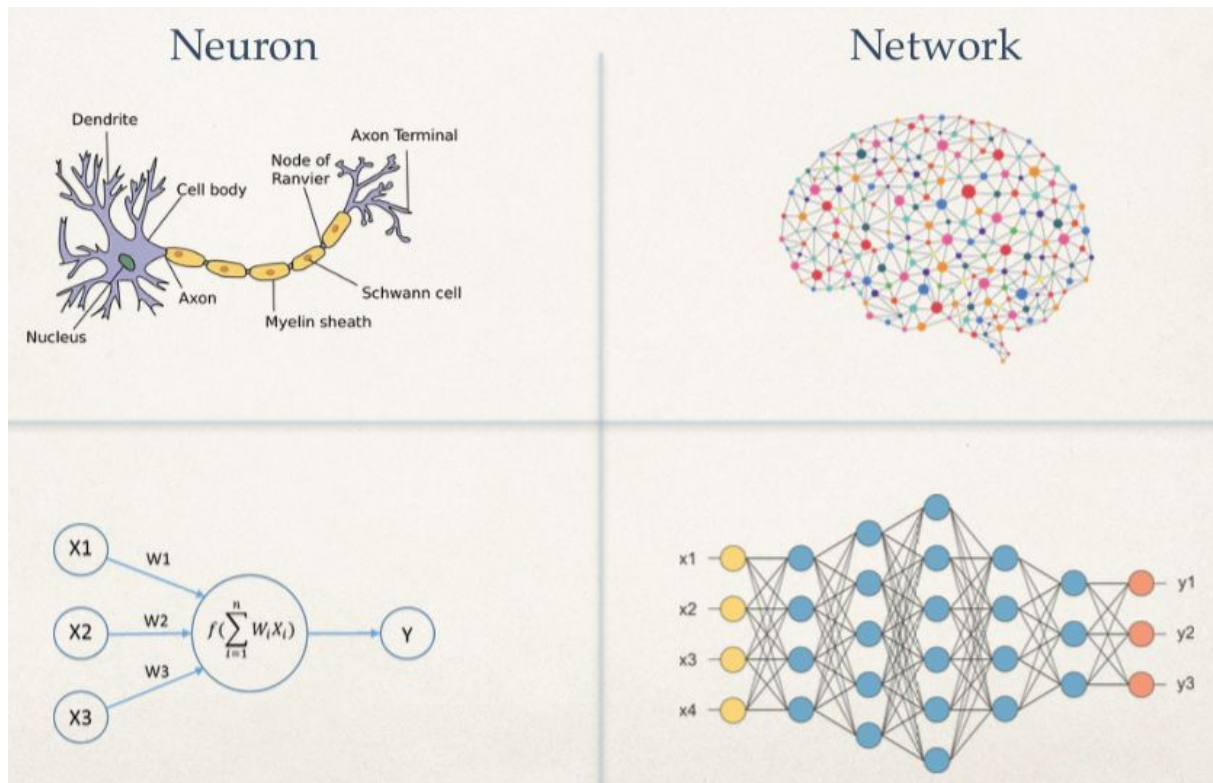


$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2.$$

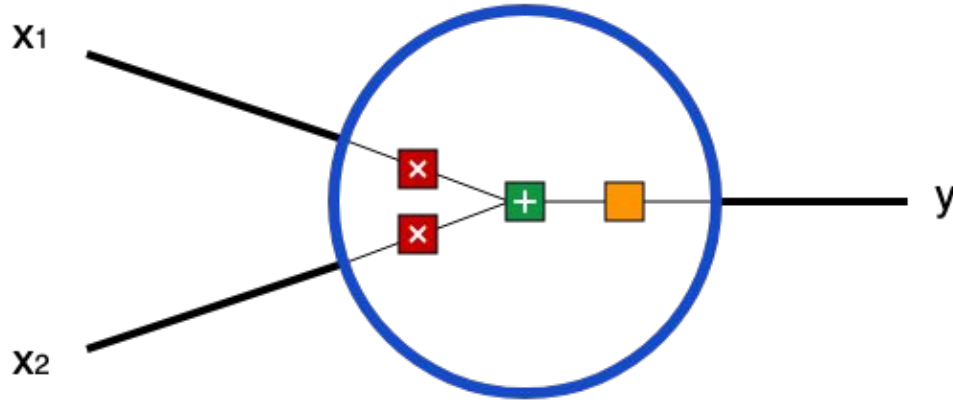
Neural networks: supervised classification

- Basic unit: **Neuron**
- A neuron takes inputs, does some math with them, and produces **one** output
- **Feedforward:** process of passing inputs forward to get an output

Hands-on today!



Neuron



- each **input** x is multiplied by a **weight** w

$$x_1 \rightarrow x_1 * w_1$$



$$x_2 \rightarrow x_2 * w_2$$

- all the weighted inputs are added together with a **bias** b

$$(x_1 * w_1) + (x_2 * w_2) + b$$



- the sum is passed through an **activation function** f

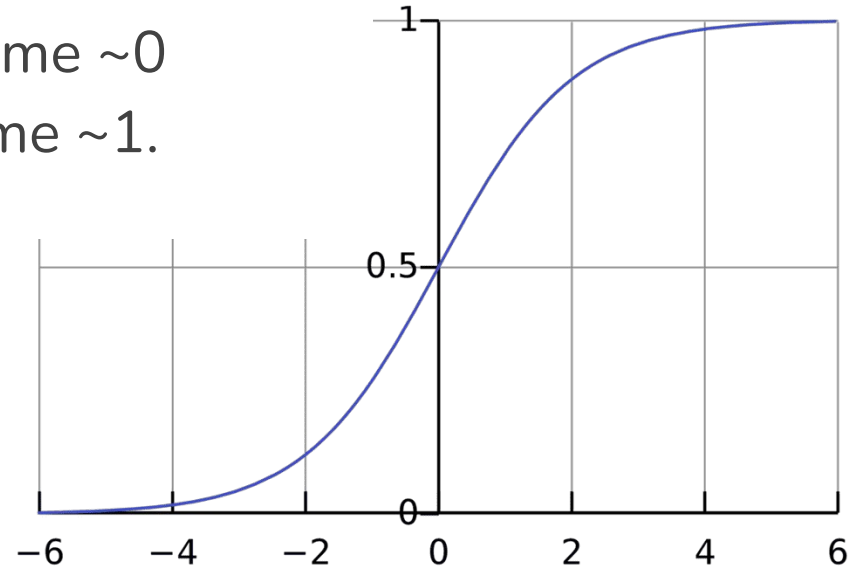
$$y = f(x_1 * w_1 + x_2 * w_2 + b)$$



Activation function

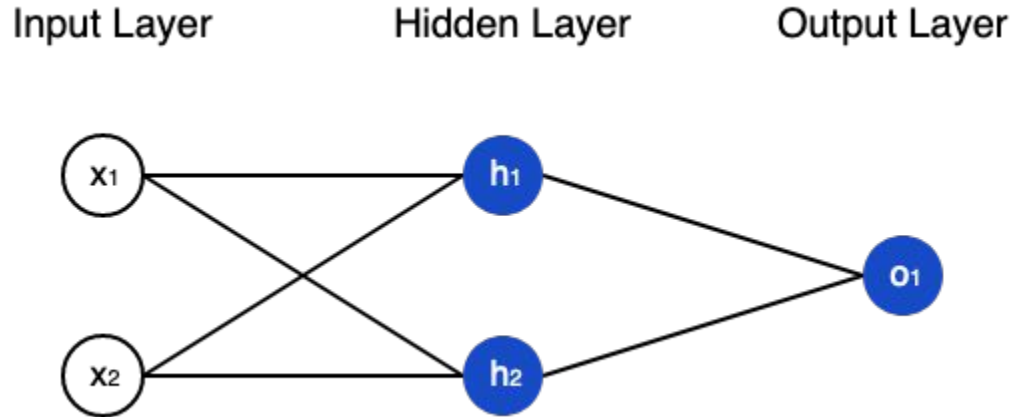
- Turns unbounded output into a known range/shape
- For example, **sigmoid** function only outputs numbers in the range (0, 1)
 - big negative numbers become ~0
 - big positive numbers become ~1.

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}.$$



Neural network

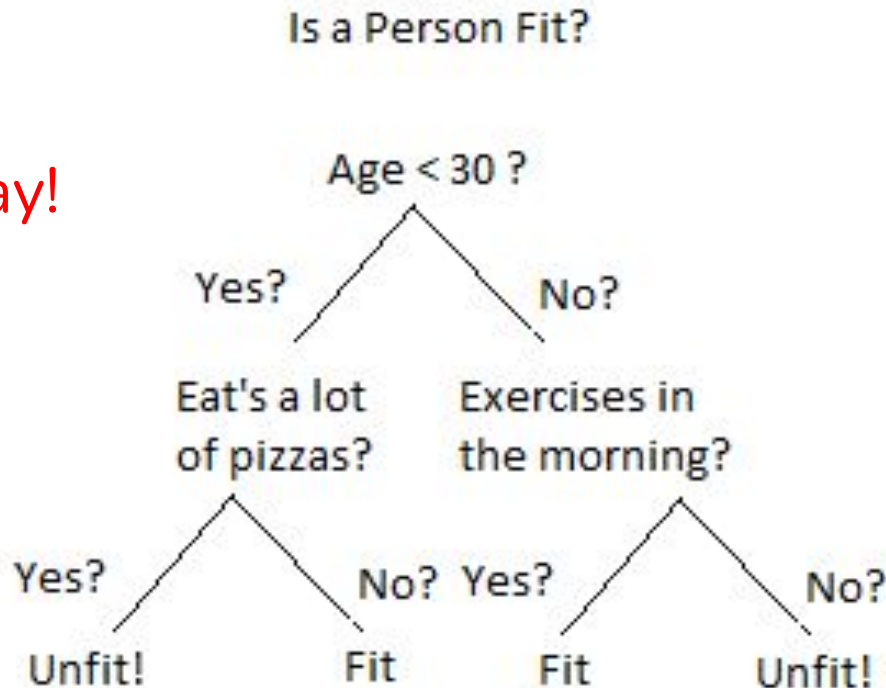
- Combining more neurons
- A **hidden layer** is any layer between the input (first) layer and output (last) layer
 - There can be multiple hidden layers
- This network has:
 - one **input** layer with 2 inputs
 - one **hidden** layer with 2 neurons
 - one **output** layer with 1 neuron



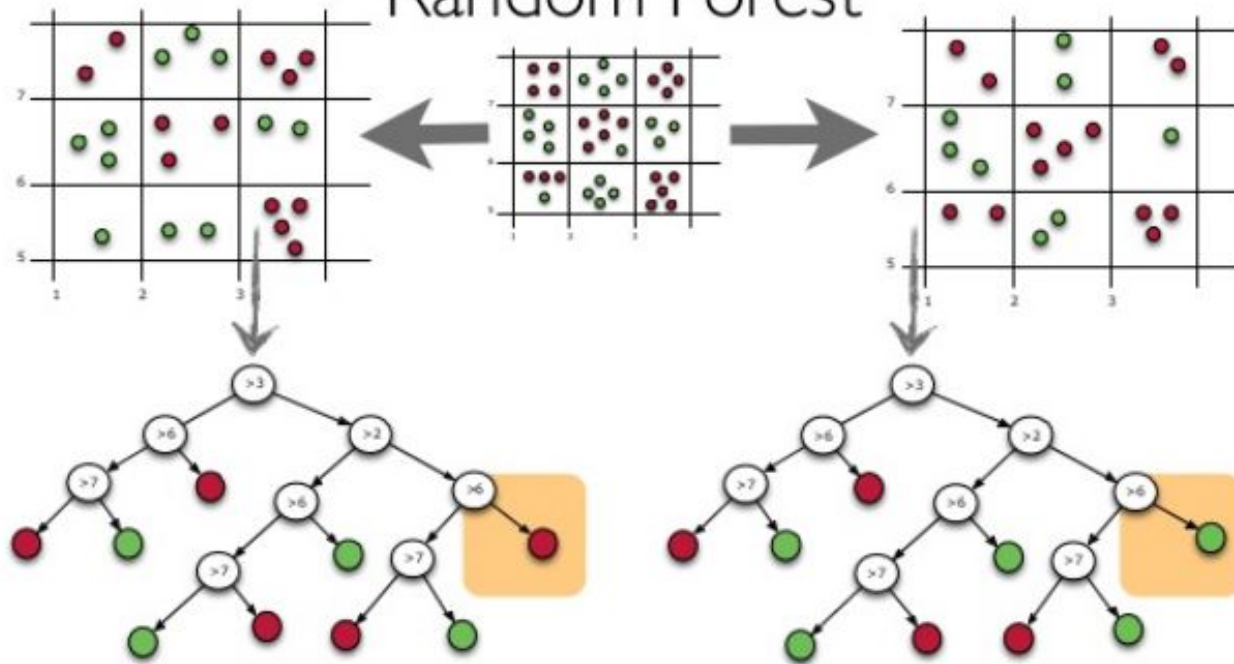
Decision trees

Typically used in combinations (Random forest, Gradient Tree Boosting)

Hands-on today!

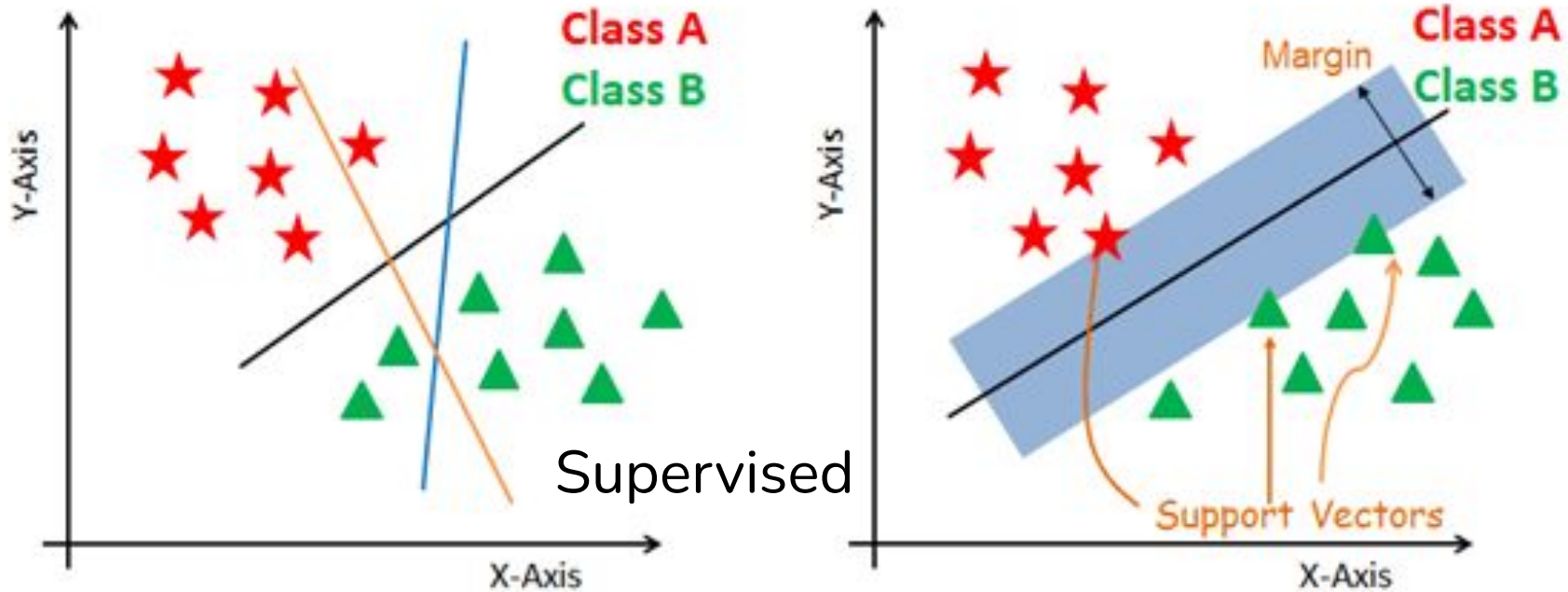


Random Forest



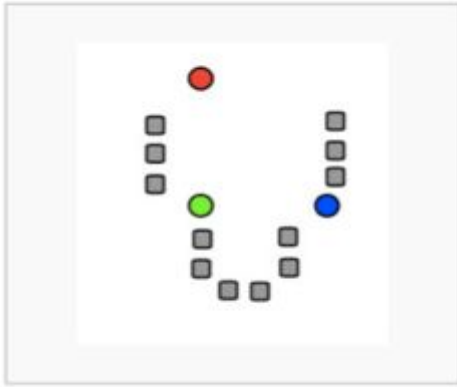
- Each tree sees part of the training sets and captures part of the information it contains

Support vector machines (SVG)

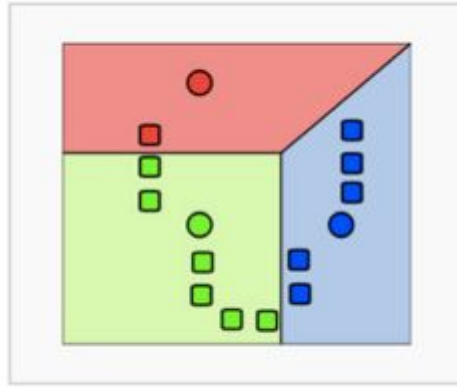


<https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>

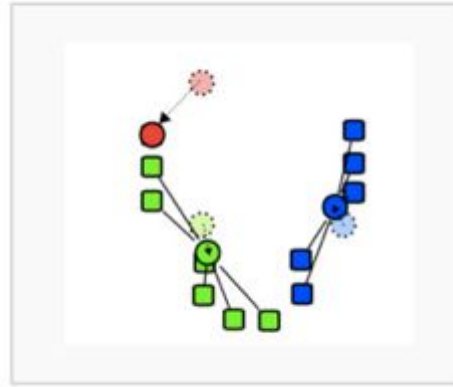
K-means clustering



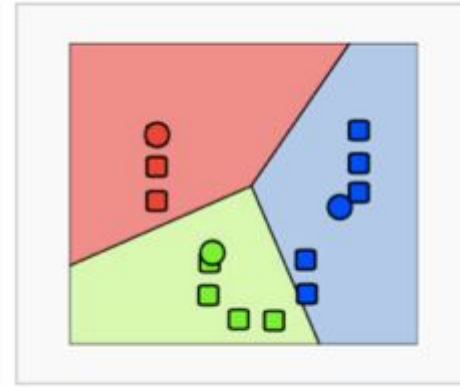
1. k initial "means" (in this case $k=3$) are randomly generated within the data domain (shown in color).



2. k clusters are created by associating every observation with the nearest mean. The partitions here represent the **Voronoi diagram** generated by the means.



3. The **centroid** of each of the k clusters becomes the new mean.

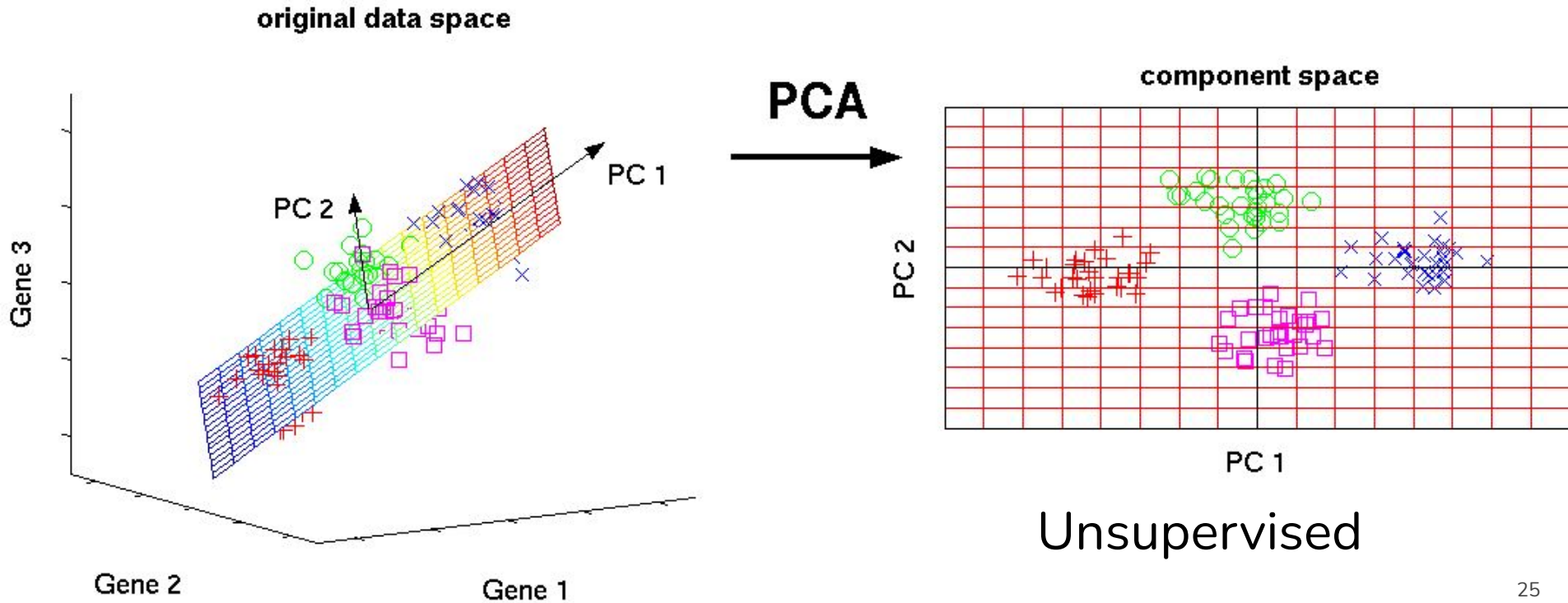


4. Steps 2 and 3 are repeated until convergence has been reached.

Unsupervised

Principal Component Analysis (PCA)

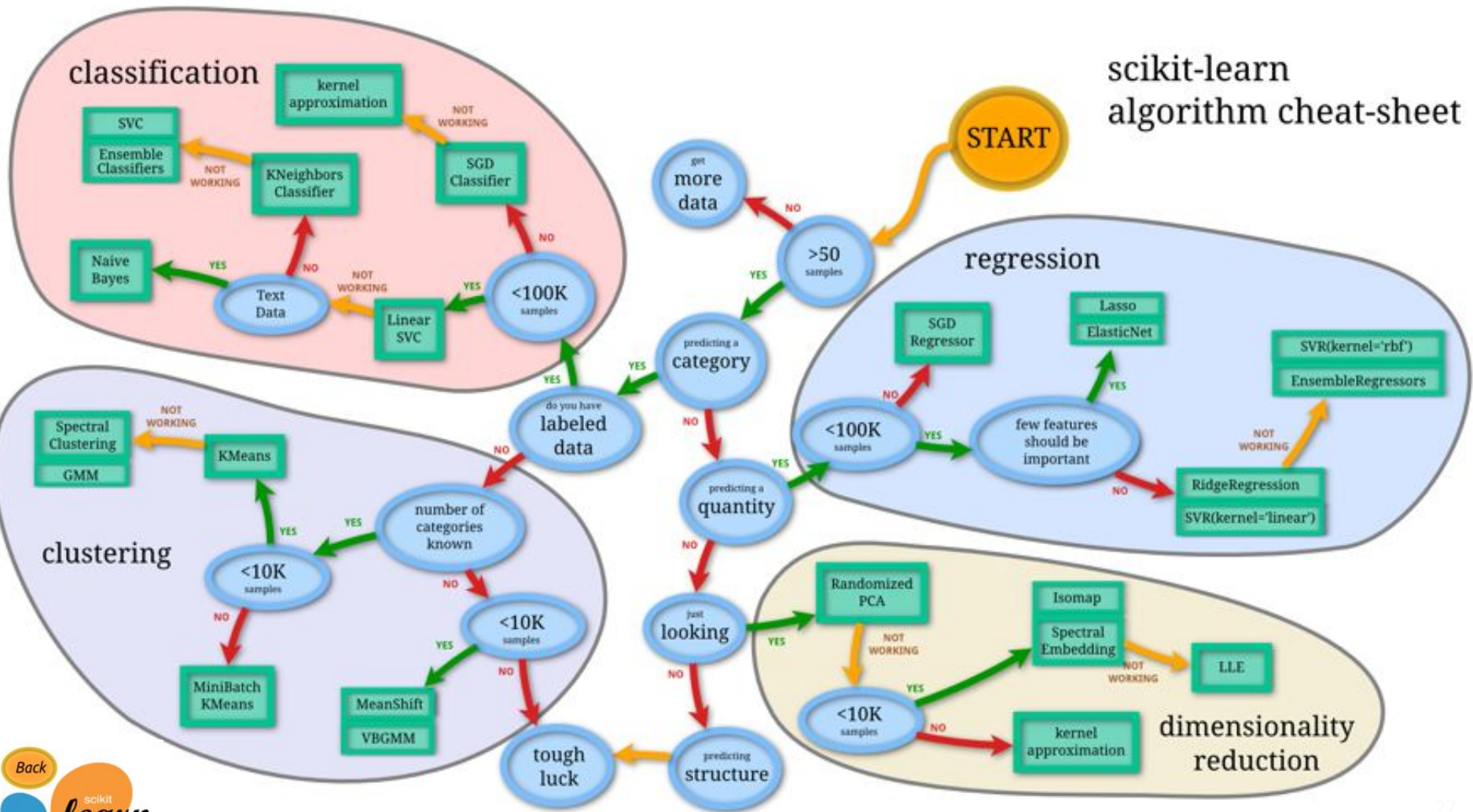
- Used for dimensionality reduction









Ensembles

- **Bagging**
 - building multiple models (typically of the *same* type) from different subsamples of the training dataset
- **Boosting**
 - building multiple models (typically of the *same* type) each of which learns to fix the predictions errors of a prior model in the chain
- **Stacking**
 - building multiple models (typically of *different* types) and supervisor model that learns how to best combine the predictions of the primary model
- **Weighting|Blending**
 - combine multiple models into single prediction using different weight functions

scikit-learn algorithm cheat-sheet



	TYPE	NAME	DESCRIPTION	ADVANTAGES	DISADVANTAGES
Linear		Linear regression	The “best fit” line through all data points. Predictions are numerical.	Easy to understand – you clearly see what the biggest drivers of the model are.	<ul style="list-style-type: none"> ✗ Sometimes too simple to capture complex relationships between variables. ✗ Tendency for the model to “overfit”.
		Logistic regression	The adaptation of linear regression to problems of classification (e.g., yes/no questions, groups, etc.)	Also easy to understand.	<ul style="list-style-type: none"> ✗ Sometimes too simple to capture complex relationships between variables. ✗ Tendency for the model to “overfit”.
Tree-based		Decision tree	A graph that uses a branching method to match all possible outcomes of a decision.	Easy to understand and implement.	<ul style="list-style-type: none"> ✗ Not often used on its own for prediction because it's also often too simple and not powerful enough for complex data.
		Random Forest	Takes the average of many decision trees, each of which is made with a sample of the data. Each tree is weaker than a full decision tree, but by combining them we get better overall performance .	A sort of “wisdom of the crowd”. Tends to result in very high quality models. Fast to train.	<ul style="list-style-type: none"> ✗ Can be slow to output predictions relative to other algorithms. ✗ Not easy to understand predictions.
		Gradient Boosting	Uses even weaker decision trees, that are increasingly focused on “hard” examples .	High-performing.	<ul style="list-style-type: none"> ✗ A small change in the feature set or training set can create radical changes in the model. ✗ Not easy to understand predictions.
Neural networks		Neural networks	Mimics the behavior of the brain. Neural networks are interconnected neurons that pass messages to each other. Deep learning uses several layers of neural networks put one after the other.	Can handle extremely complex tasks - no other algorithm comes close in image recognition.	<ul style="list-style-type: none"> ✗ Very, very slow to train, because they have so many layers. Require a lot of power. ✗ Almost impossible to understand predictions.

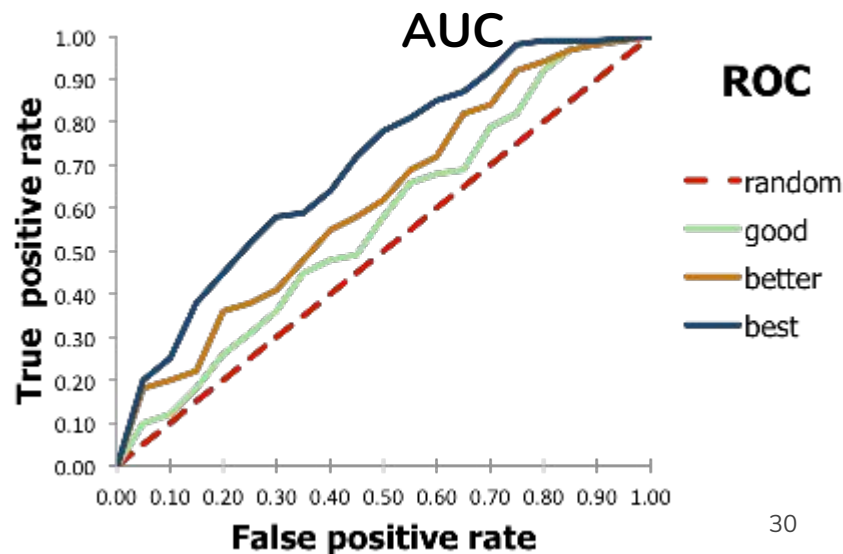
*All models are wrong, but
some are useful (George Box)*

Classification metrics

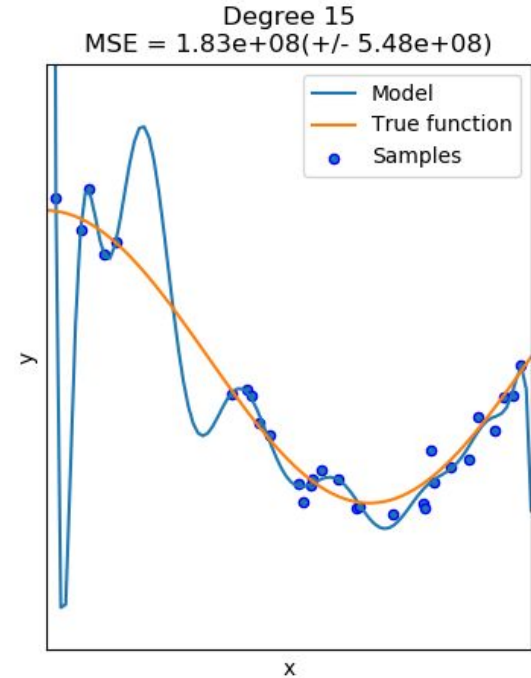
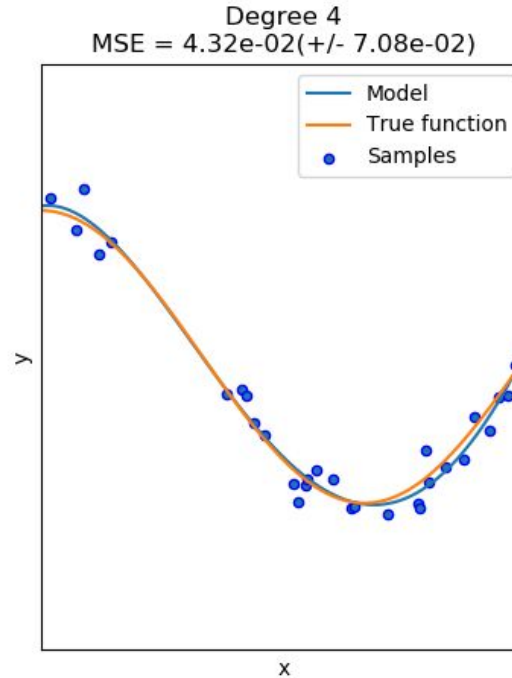
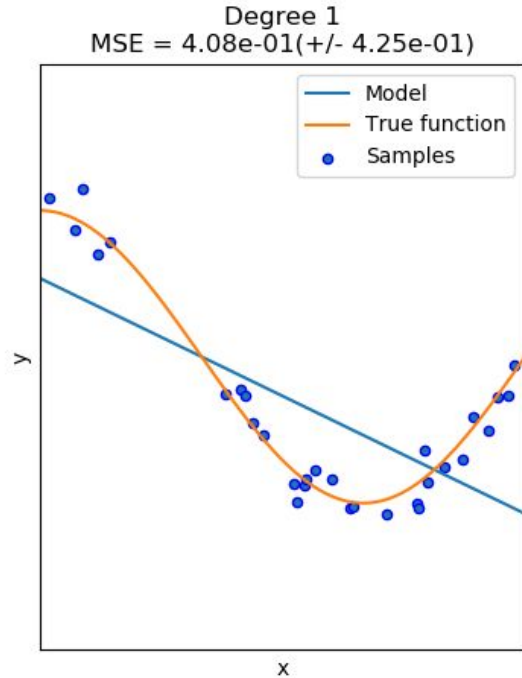
- **ROC**: Receiver Operating Characteristics
- **AUC**: Area under the curve
- **TPR**: True positive rate
- **FPR**: False positive rate
- **TNR/FNR**: True/False negative rate

Confusion matrix

		True class			
		p	n		
Hypothesized class	Y	True Positives	False Positives	fp rate = $\frac{FP}{N}$	tp rate = $\frac{TP}{P}$
	N	False Negatives	True Negatives	precision = $\frac{TP}{TP+FP}$	recall = $\frac{TP}{P}$
Column totals:		P	N	accuracy = $\frac{TP+TN}{P+N}$	
				F-measure = $\frac{2}{1/precision + 1/recall}$	



Overfitting / underfitting



Underfitting

Model doesn't have enough parameters to describe data

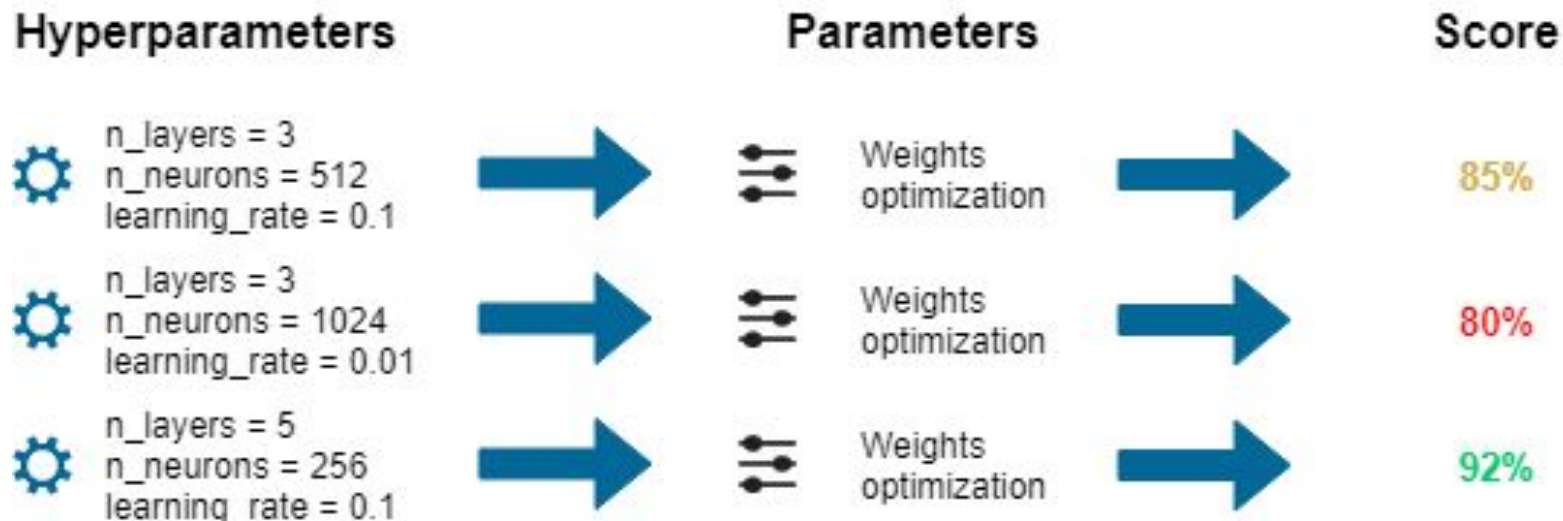


Overfitting

Model has too many parameters

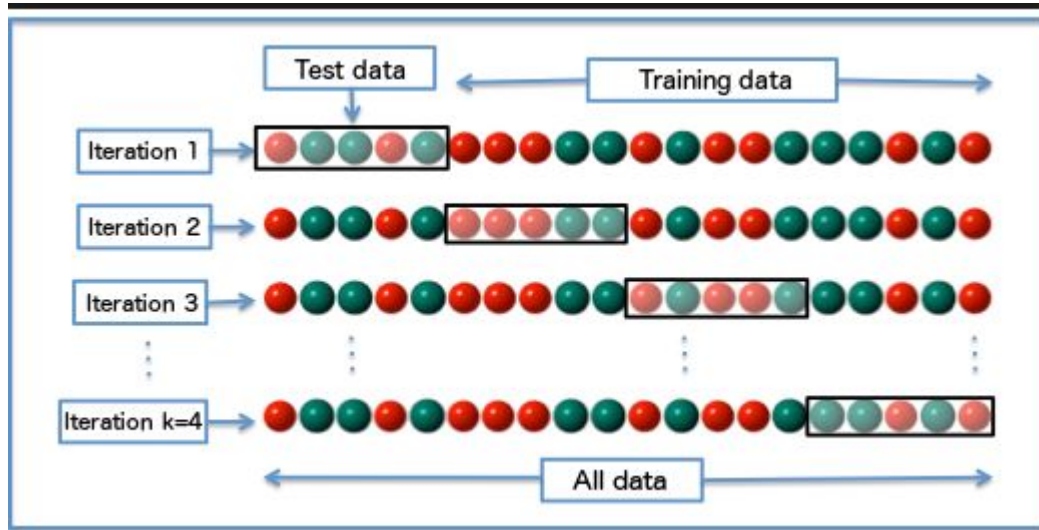
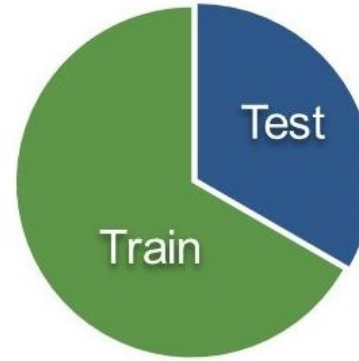
Hyperparameters vs parameters

- Model **parameters** are learned during training when we optimize a loss function
- **Hyperparameters** are not model parameters and they cannot be directly trained from the data



Cross-validation

- Train/test split
- K-folds cross validation



Modeling Algorithm

Tune

hyperparameters (tuning options)

- Polynomial order, penalty parameter, ...
- Network configuration, solver options, ...
- Max tree depth, splitting criterion, ...

Model

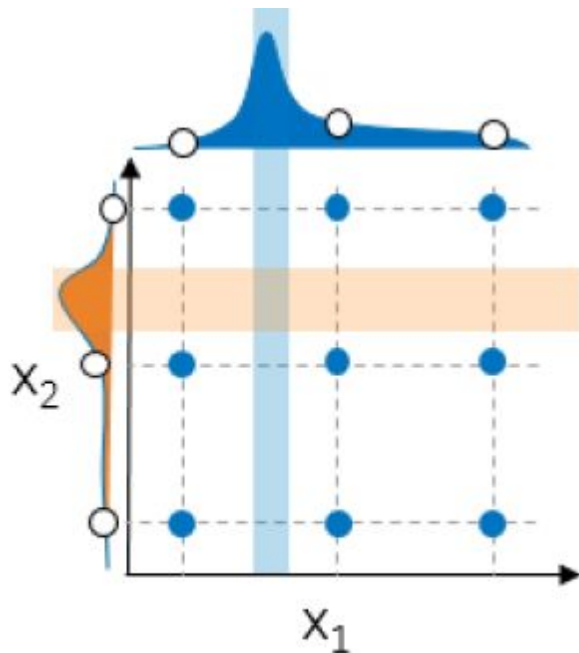
Train

model parameters

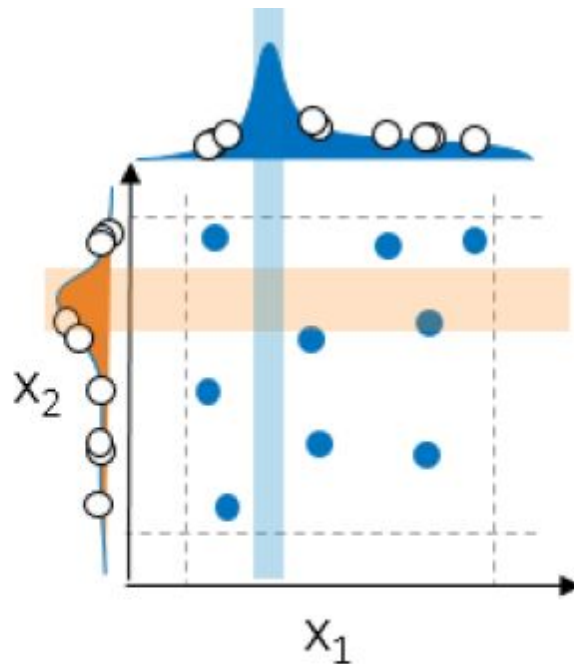
- Regression coefficients
- Neural net weights
- Tree splitting rules

...

Hyperparameter tuning



(a) Standard Grid Search



(b) Random Search

Hands-on today

1. **Point your browser to:** <https://yoga.to.infn.it>
2. **Open a terminal:**
 - `cd MLCourse-2021`
 - `git pull`
 - `cp Notebooks/Day2/* ../`
3. **From JupyterHub Home tab:**
 - start and run *ML_GBT.ipynb*
 - Apache ML Library [MLLib](#)
 - Gradient Boosting Trees (GBT)
 - Hyperparameter optimisation
 - Multilayer Perceptron Classifier (MPC) - Bonus