Big data science
Day 2

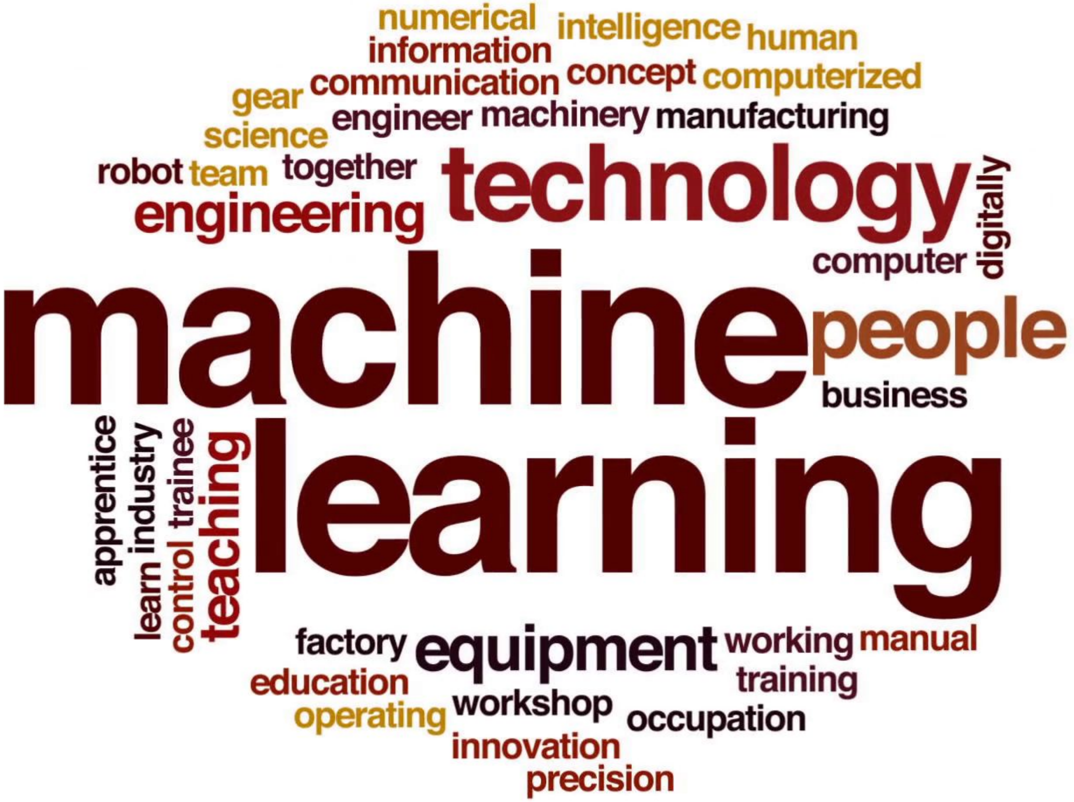
F. Legger - INFN Torino
https://github.com/leggerf/MLCourse-2022

Last lecture

- Big data
- Analytics

Today

- **Machine learning**



A PROPOSAL FOR THE

DARTMOUTH SUMMER RESEARCH PROJECT

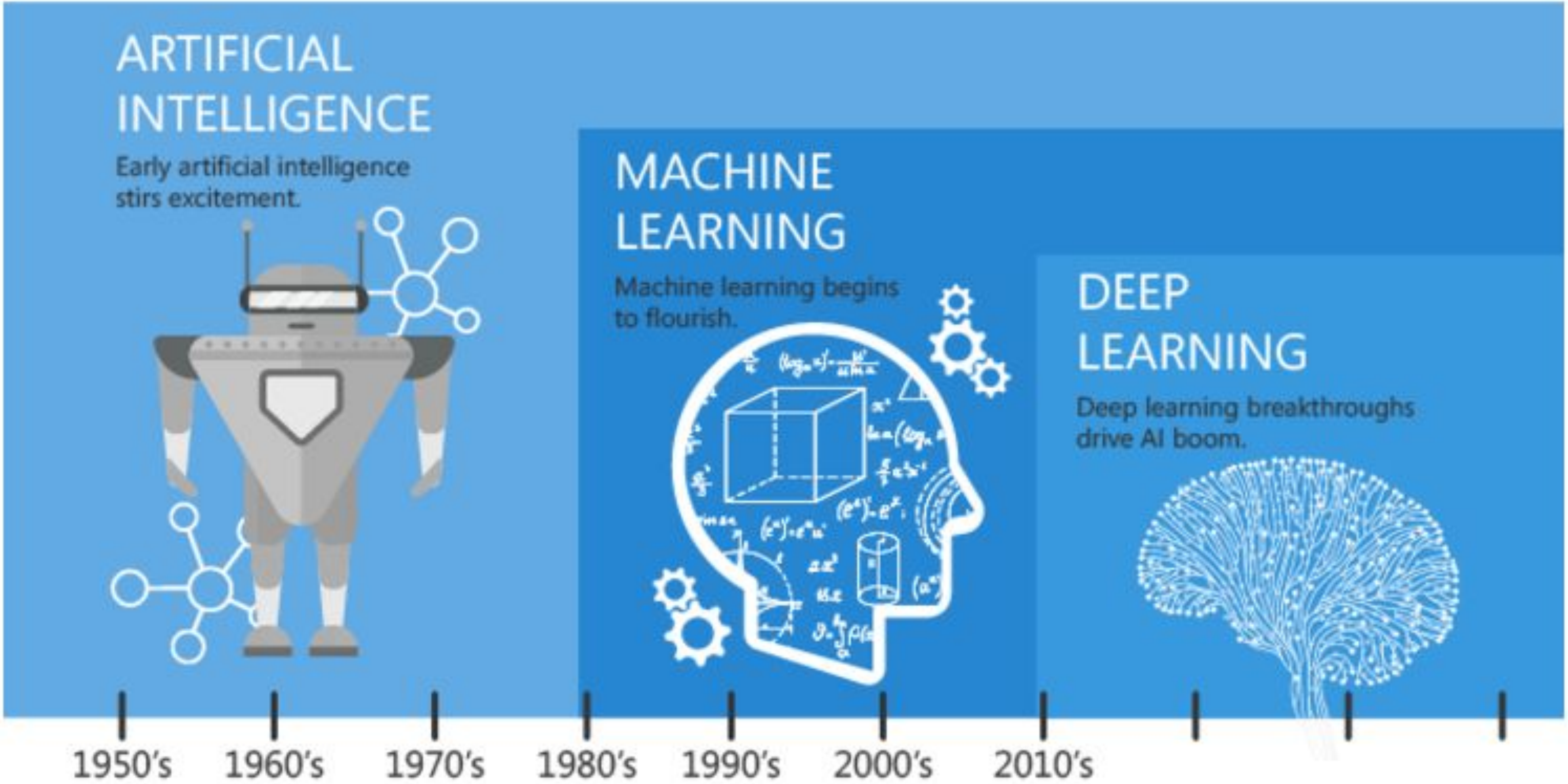
ON ARTIFICIAL INTELLIGENCE

J. McCarthy, Dartmouth College
M. L. Minsky, Harvard University
N. Rochester, I. B. M. Corporation
C. E. Shannon, Bell Telephone Laboratories

August 31, 1955

Our ultimate objective is to make programs that learn from their experience as effectively as humans do

[John McCarthy, 1958]



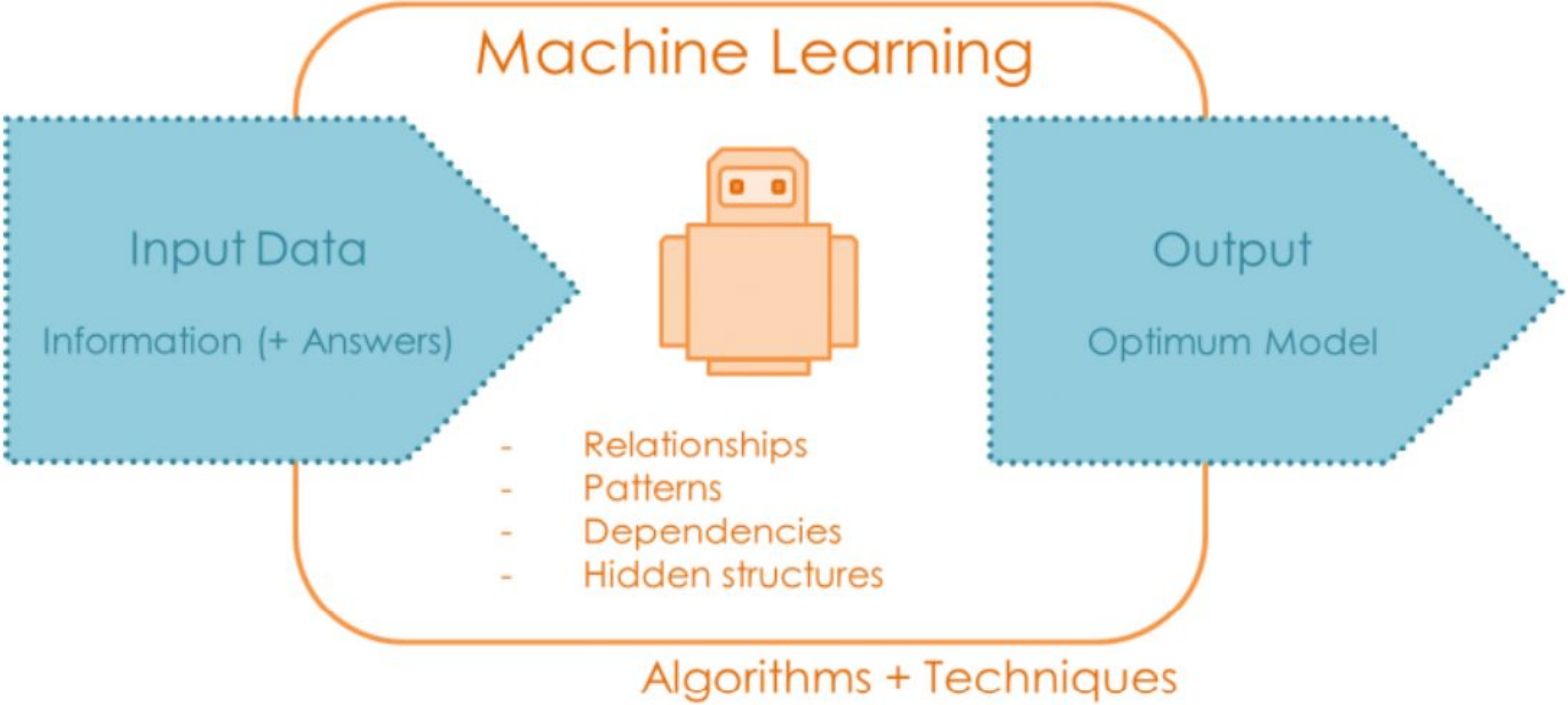
ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

Machine Learning

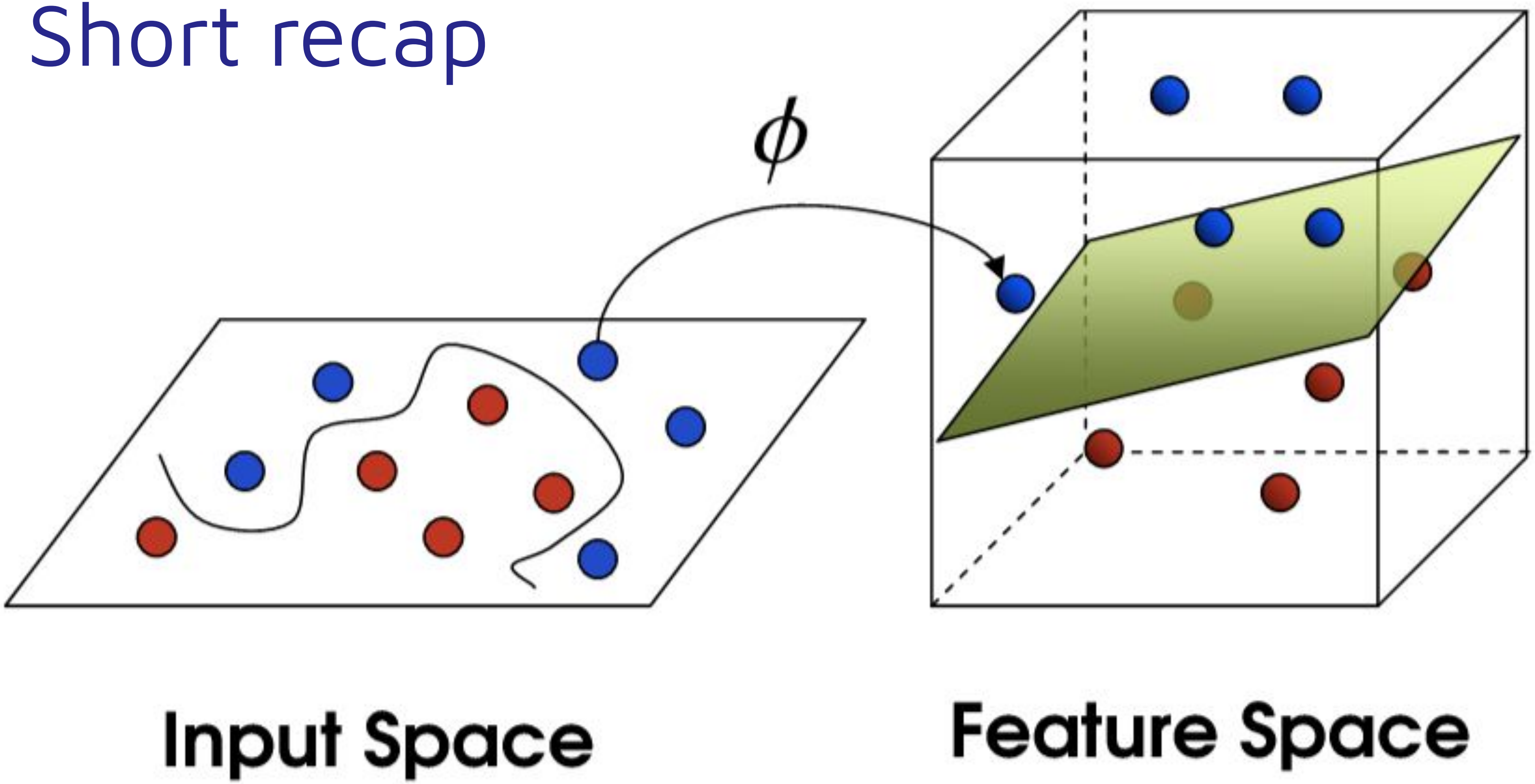
Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead [Wikipedia]

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]

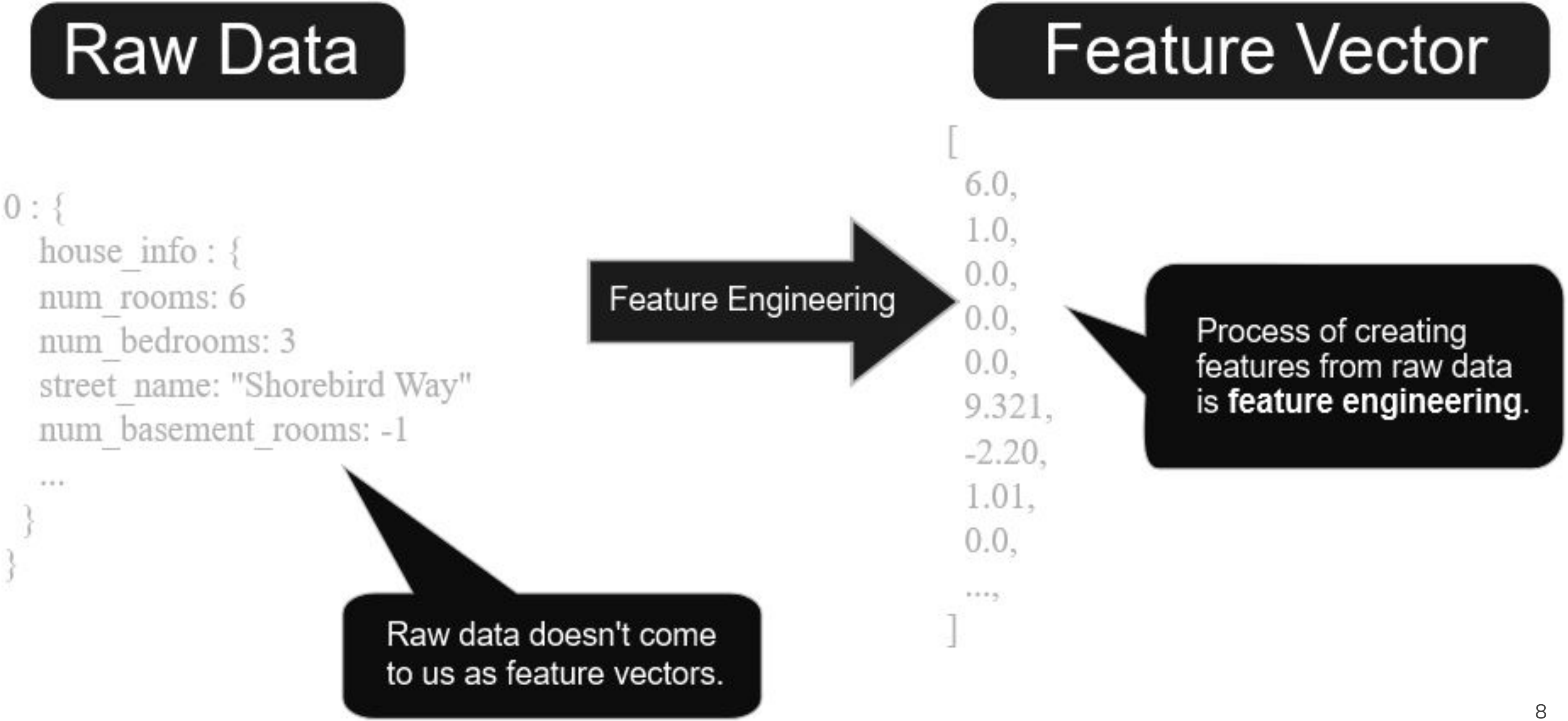

Machine Learning
Input Data
Information (+ Answers)
Output
Optimum Model
- Relationships
- Patterns
- Dependencies
- Hidden structures
Algorithms + Techniques

Short recap

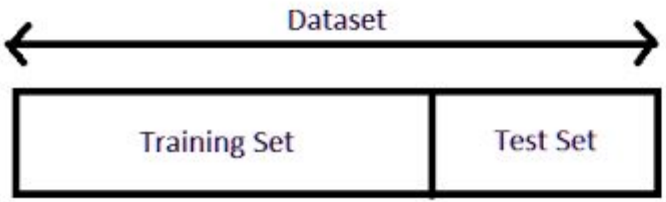


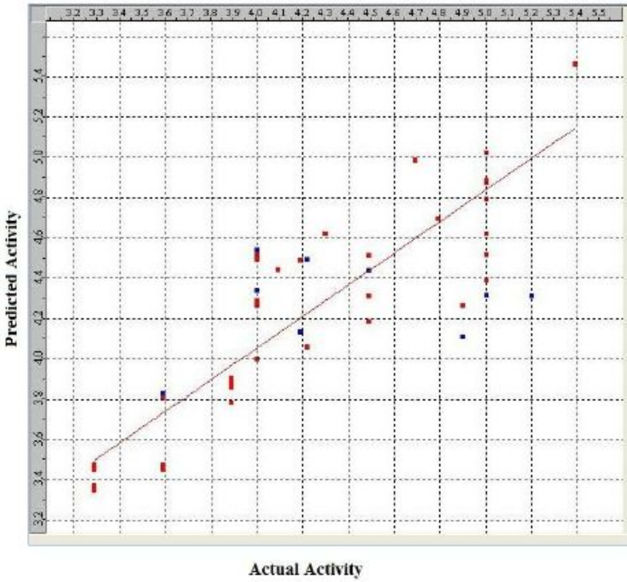
Raw data → **Preprocessing**

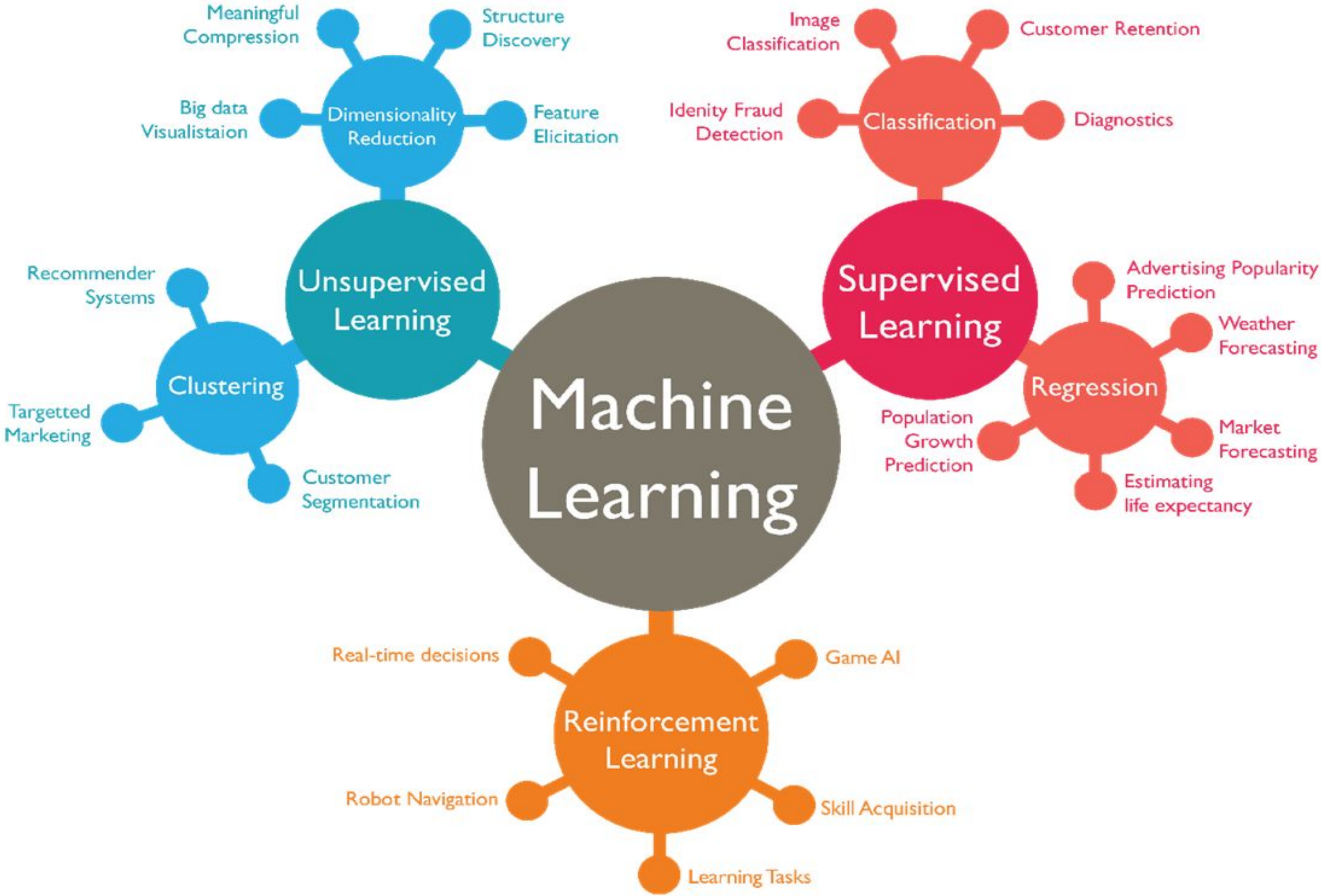
Feature engineering



Training and test set

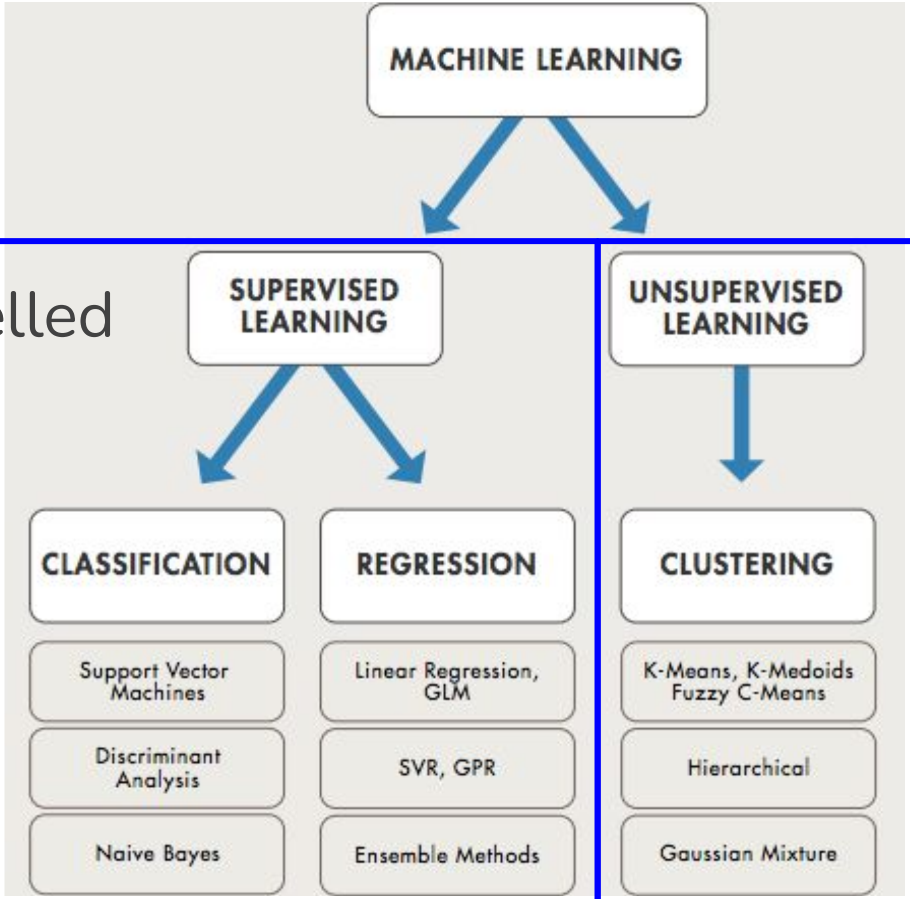





Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

Training data is labelled

sensor1	sensor 2	sensor3	label
0.3	0.2	0.6	0
0.3	0.2	0.6	0
0.3	0.2	0.6	0
0.3	0.2	0.6	0
0.3	0.2	0.6	1
0.3	0.2	0.6	1



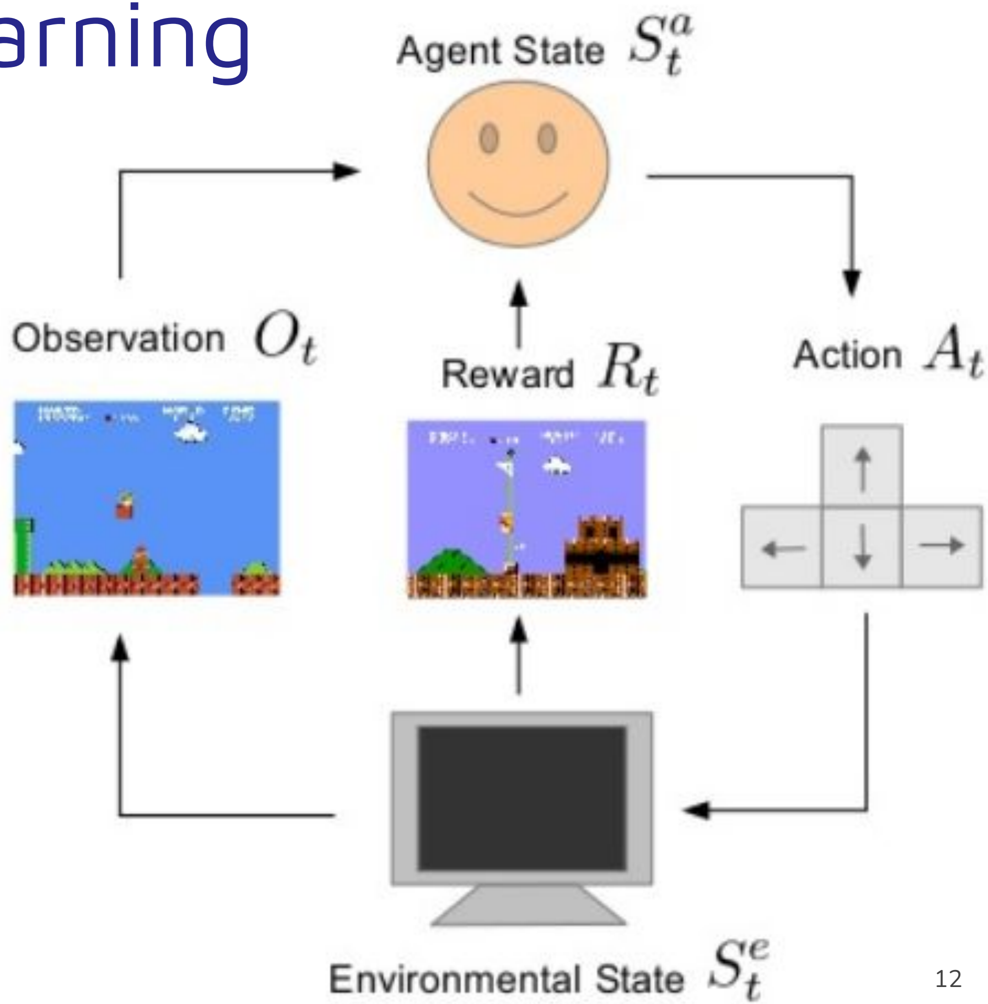
No labels in training data

Discrete labels

Continuous labels

Reinforcement learning

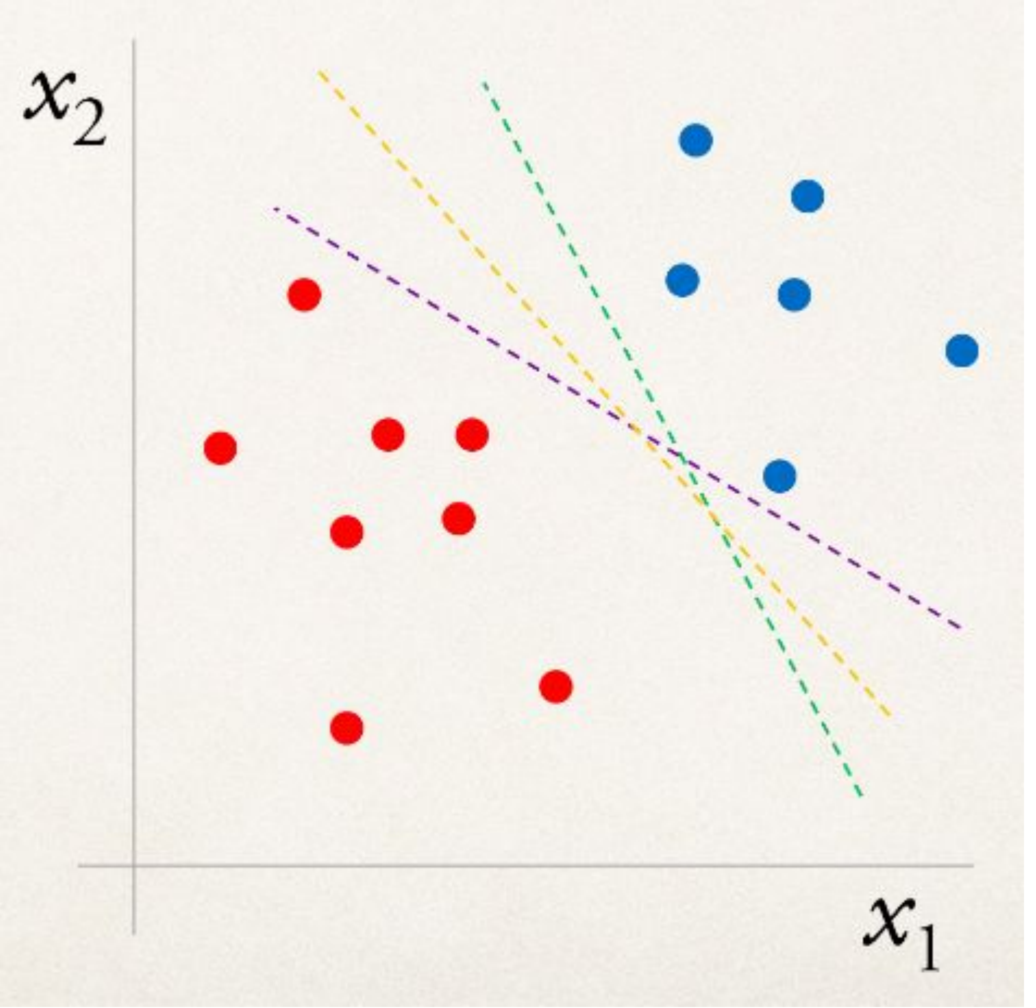
- getting an agent to act in the world so as to maximize its rewards
- sparse and time delayed labels (**rewards**)



Classification

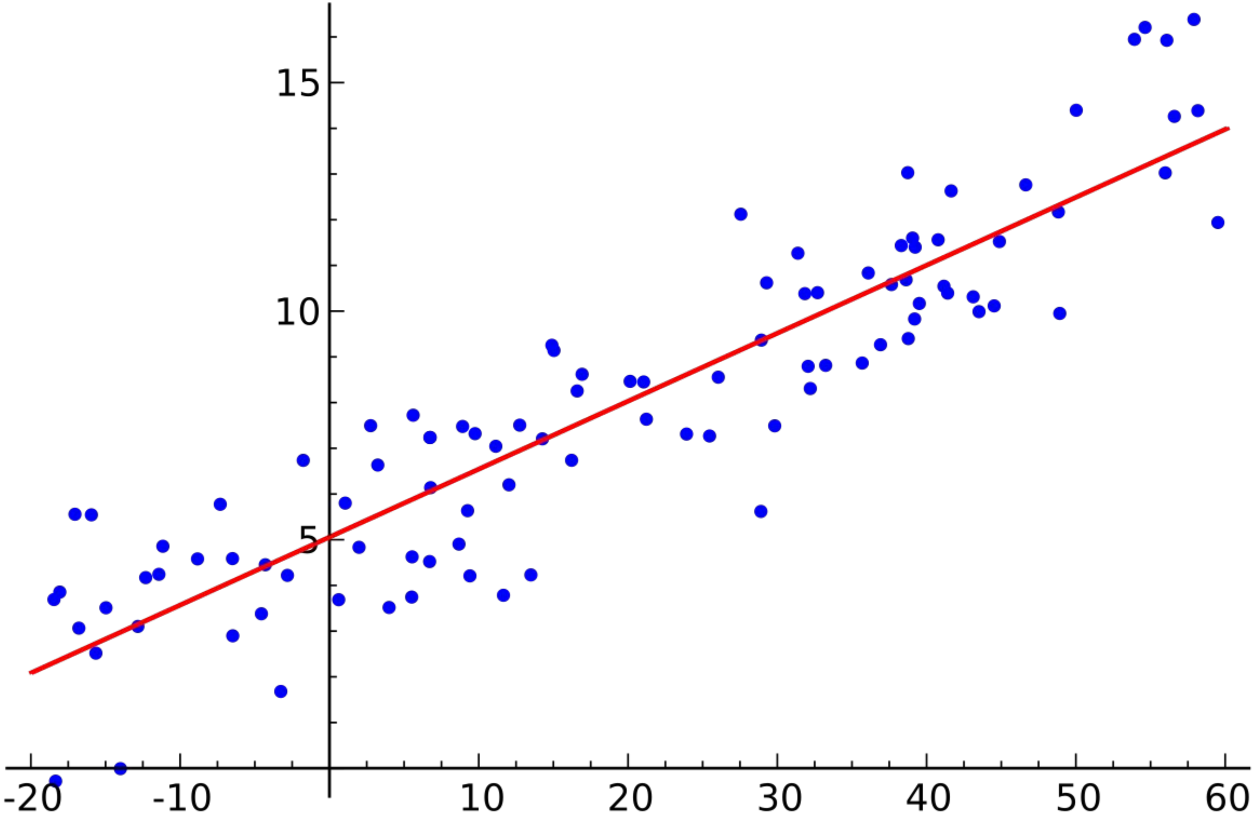
- Businesses who target customers: good vs bad, stay or leave
- **Signal vs background**
- ....

Supervised, discrete labels



Regression

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

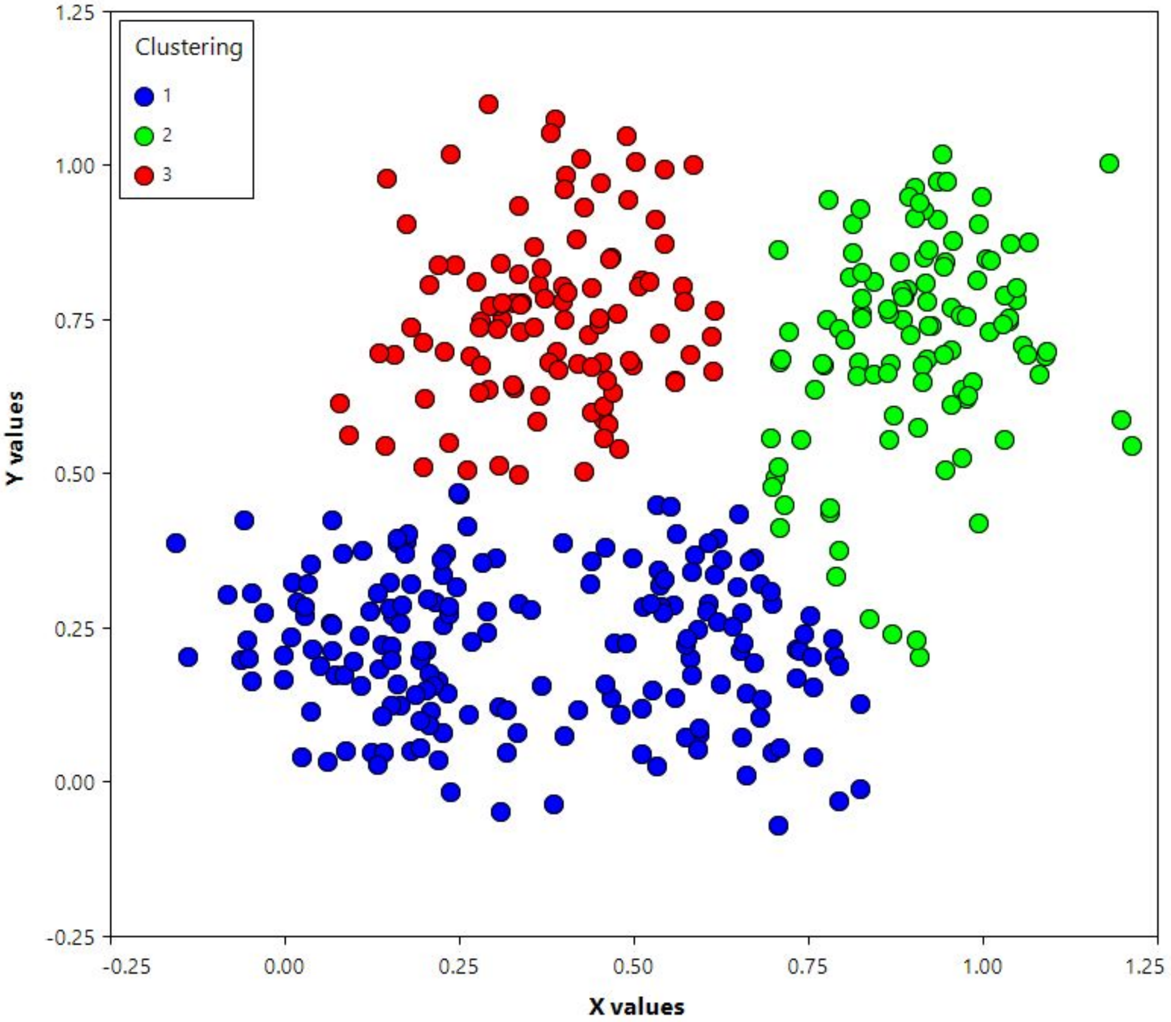


Supervised, continuous labels

Clustering

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

Unsupervised



Example: supervised classification

Ingredients

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

Recipe

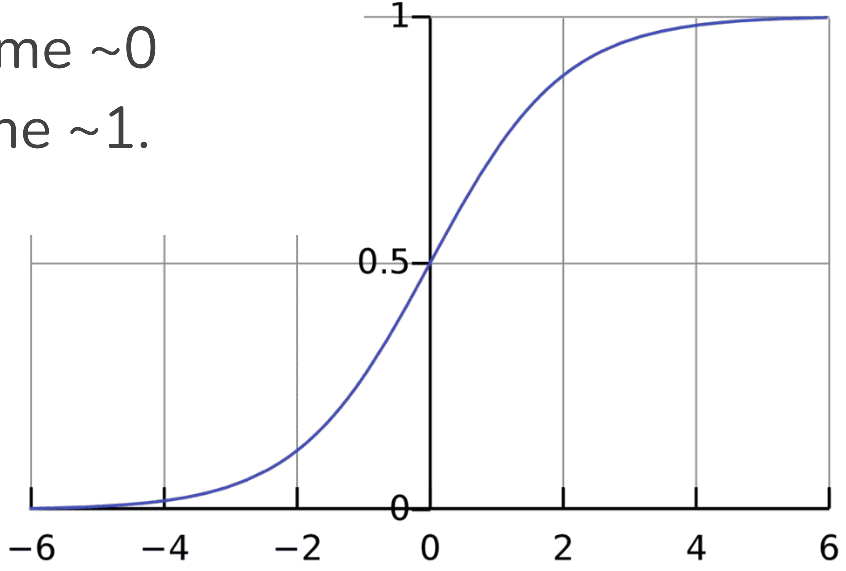
- **Predictions: z = φ(W^T *X*)** yields (0,1)
  - **Weights: W** (matrix) parameters to be found by the model
  - **Activation function: φ** (step function, e.g. sigmoid)
- **Cost function == loss function == prediction error: J(W)**
  - e.g. $\sum(y_i - z_i)^2 /2$

Aim: find weights W that <u>minimize cost function</u> & give best separation

Activation function

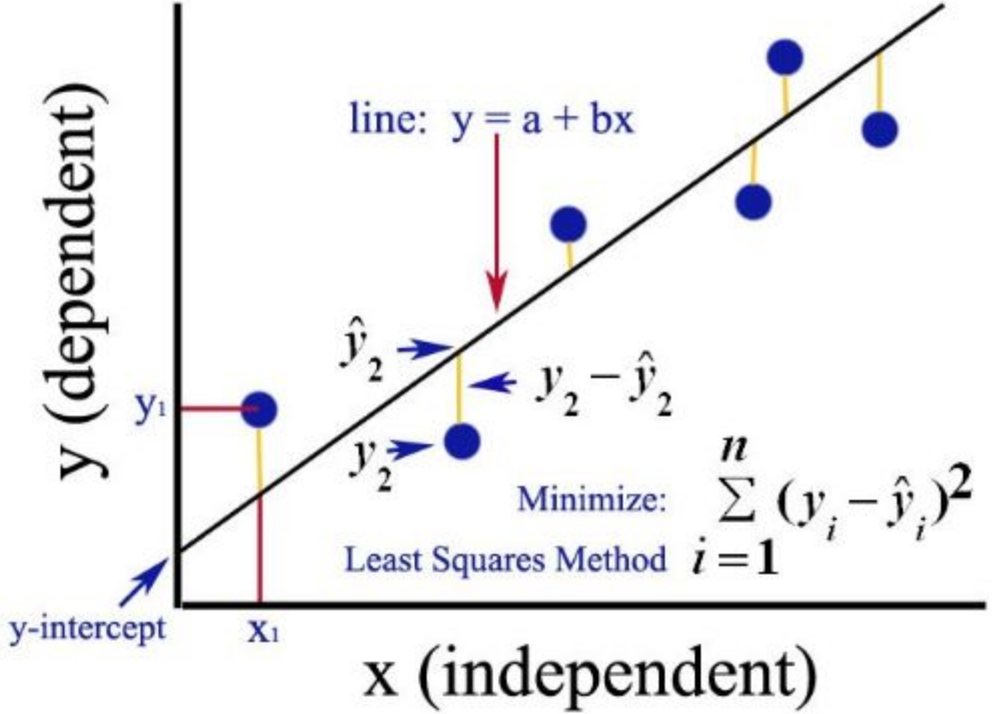
- Turns unbounded output into a known range/shape
- For example, **sigmoid** function only outputs numbers in the range (0, 1)
  - big negative numbers become ~0
  - big positive numbers become ~1.

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



Another example, linear regression

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

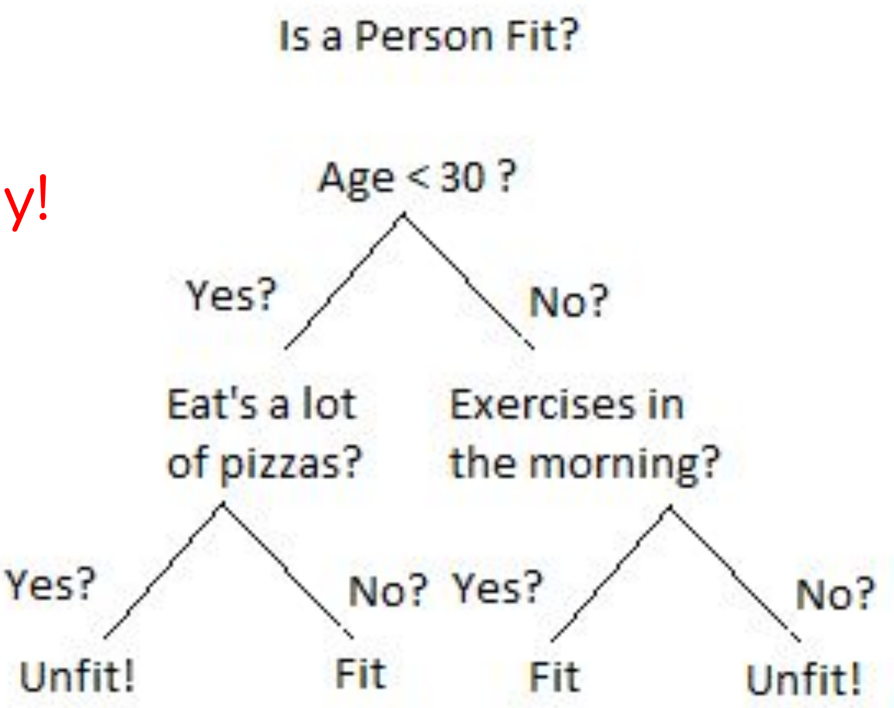


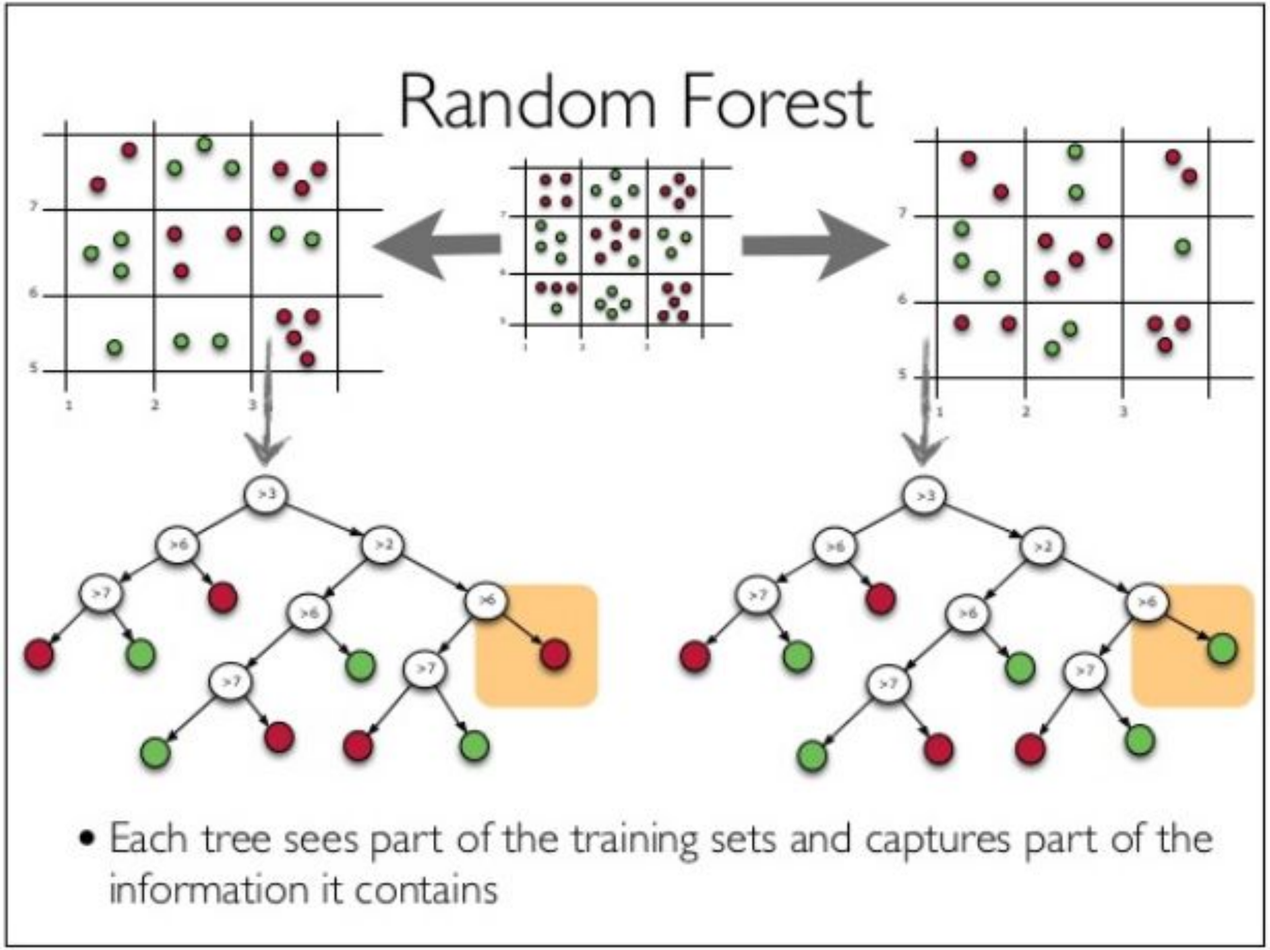
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

Decision trees: supervised classification

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

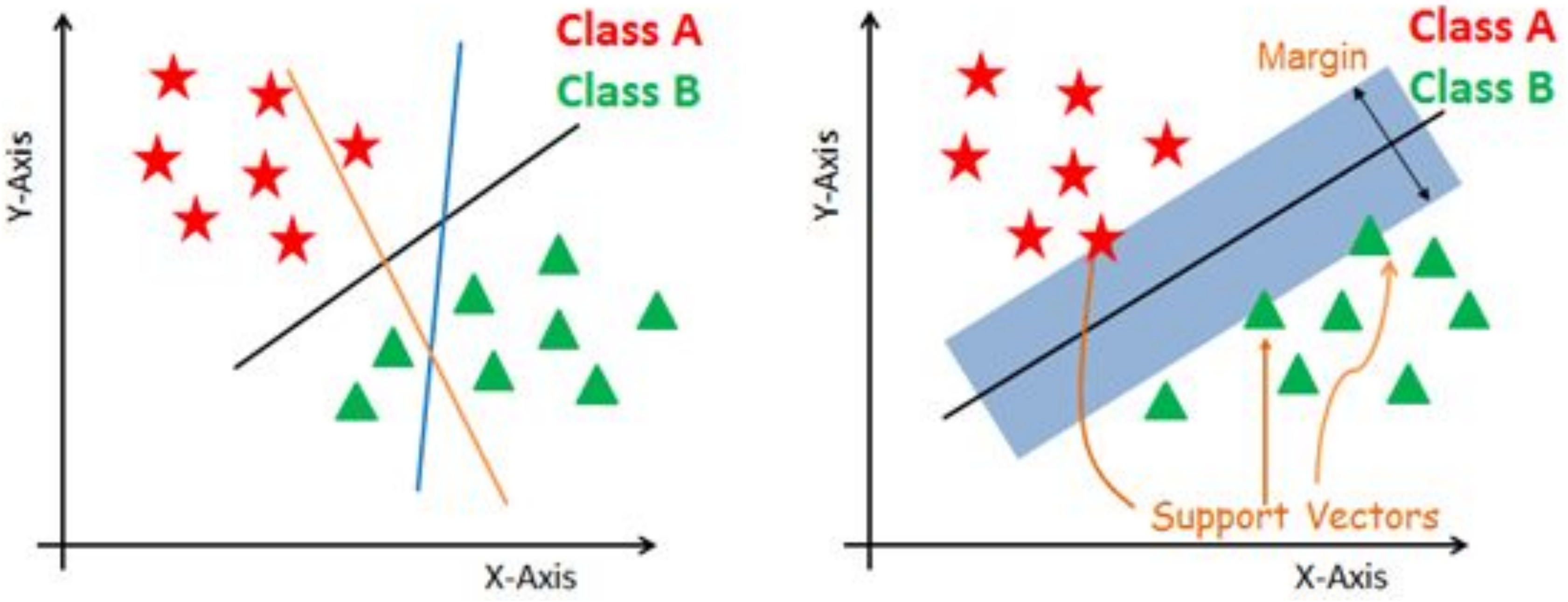
Hands-on today!





https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd

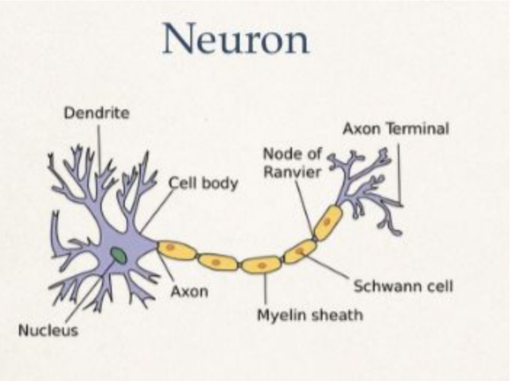
Support vector machines (SVG), supervised classification

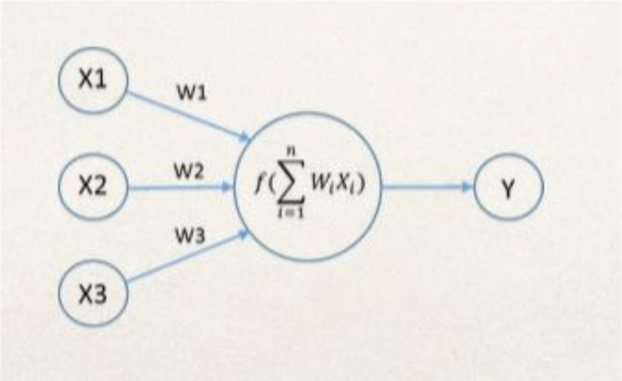


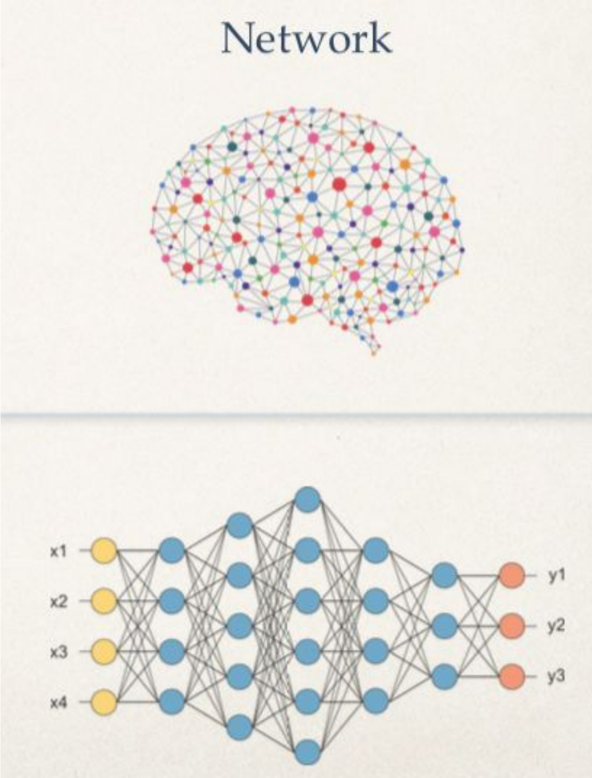
https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Neural networks: supervised classification

- Basic unit: **Neuron.** A neuron takes inputs, does some math with them, and produces **one** output



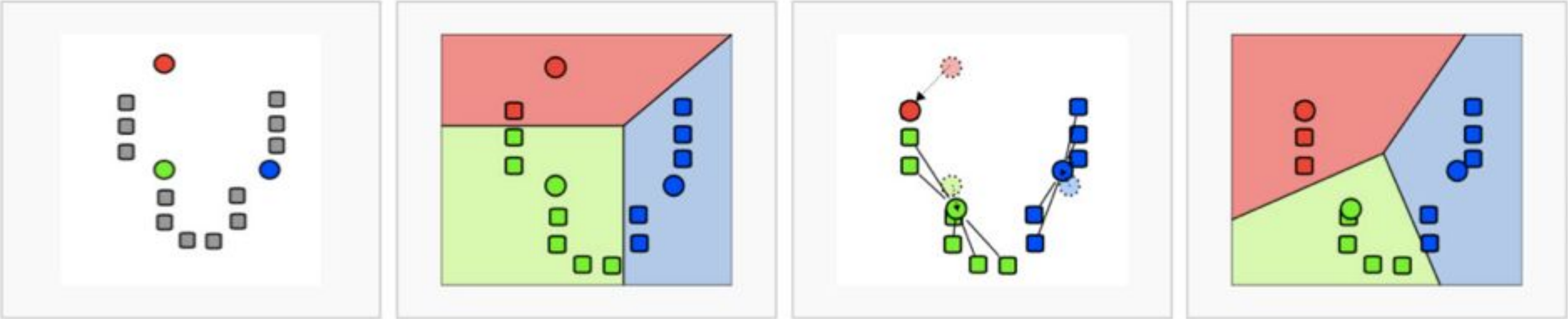




Hands-on today!

https://victorzhou.com/blog/intro-to-neural-networks/

K-means clustering, unsupervised



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

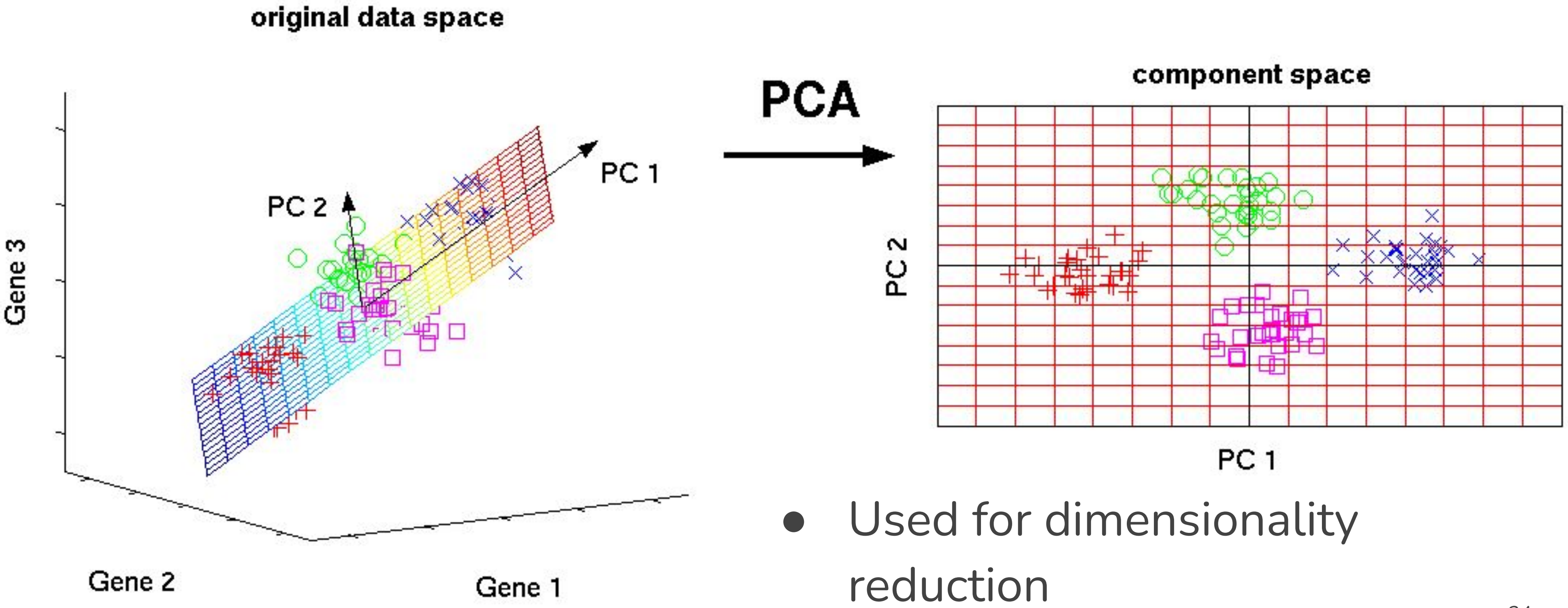
2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

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.

https://www.saedsayad.com/clustering_kmeans.htm

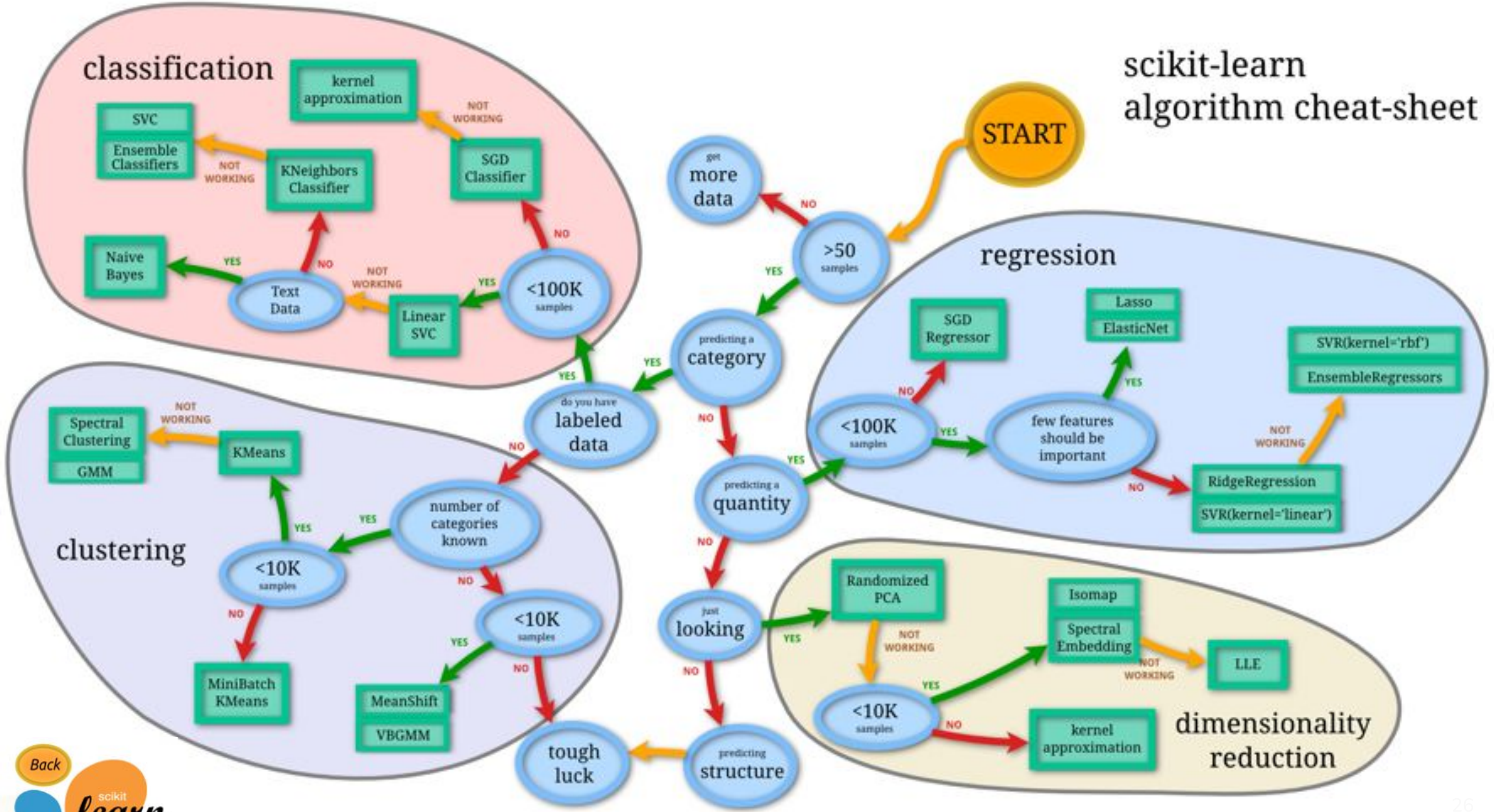
Principal Component Analysis (PCA), unsupervised



- 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
START
>50 samples
get more data
predicting a category
do you have labeled data
<100K samples
predicting a quantity
just looking
predicting structure
tough luck
classification
kernel approximation
SVC
Ensemble Classifiers
KNeighbors Classifier
SGD Classifier
Naive Bayes
Text Data
Linear SVC
clustering
Spectral Clustering
GMM
KMeans
number of categories known
<10K samples
MiniBatch KMeans
MeanShift
VBGMM
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
few features should be important
RidgeRegression
SVR(kernel='linear')
dimensionality reduction
Randomized PCA
Isomap
Spectral Embedding
LLE
kernel approximation
YES
NO
NOT WORKING
Back

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.	X Sometimes too simple to capture complex relationships between variables. X Tendency for the model to "overfit".
Linear	Logistic regression	The adaptation of **linear regression** to problems of classification (e.g., yes/no questions, groups, etc.)	Also easy to understand.	X Sometimes too simple to capture complex relationships between variables. X 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.	X Not often used on its own for prediction because it's also often too simple and not powerful enough for complex data.
Tree-based	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.	X Can be slow to output predictions relative to other algorithms. X Not easy to understand predictions.
Tree-based	Gradient Boosting	Uses even weaker decision trees, that are increasingly **focused on "hard" examples.**	High-performing.	X A small change in the feature set or training set can create radical changes in the model. X 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.	X Very, very slow to train, because they have so many layers. Require a lot of power. X 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

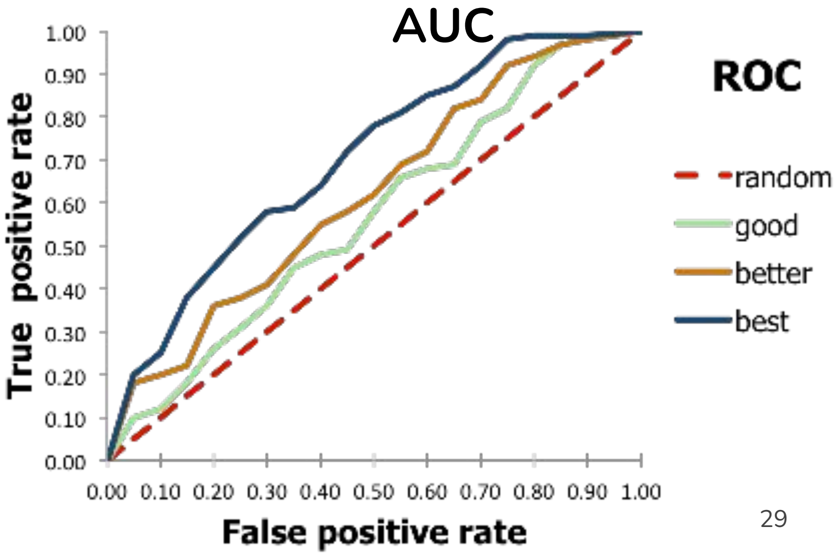
		True class p	True class n
Hypothesized class	Y	True Positives	False Positives
	N	False Negatives	True Negatives
Column totals:		P	N

$$\text{fp rate} = \frac{FP}{N} \qquad \text{tp rate} = \frac{TP}{P}$$

$$\text{precision} = \frac{TP}{TP+FP} \qquad \text{recall} = \frac{TP}{P}$$

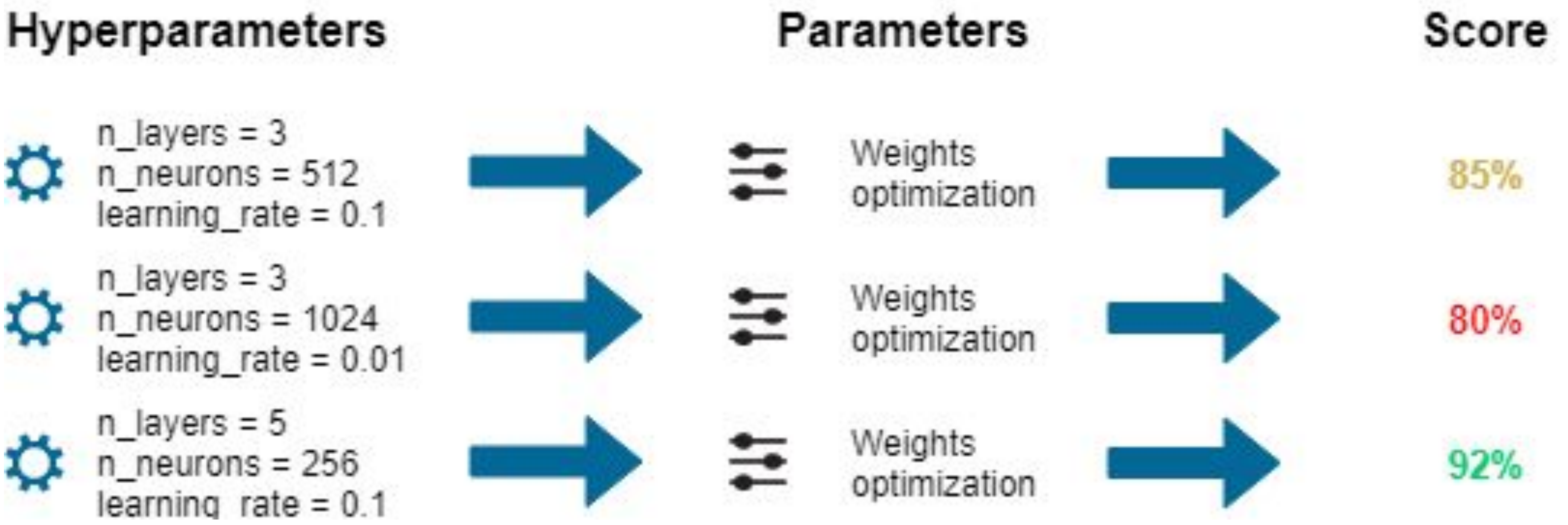
$$\text{accuracy} = \frac{TP+TN}{P+N}$$

$$\text{F-measure} = \frac{2}{1/\text{precision}+1/\text{recall}}$$



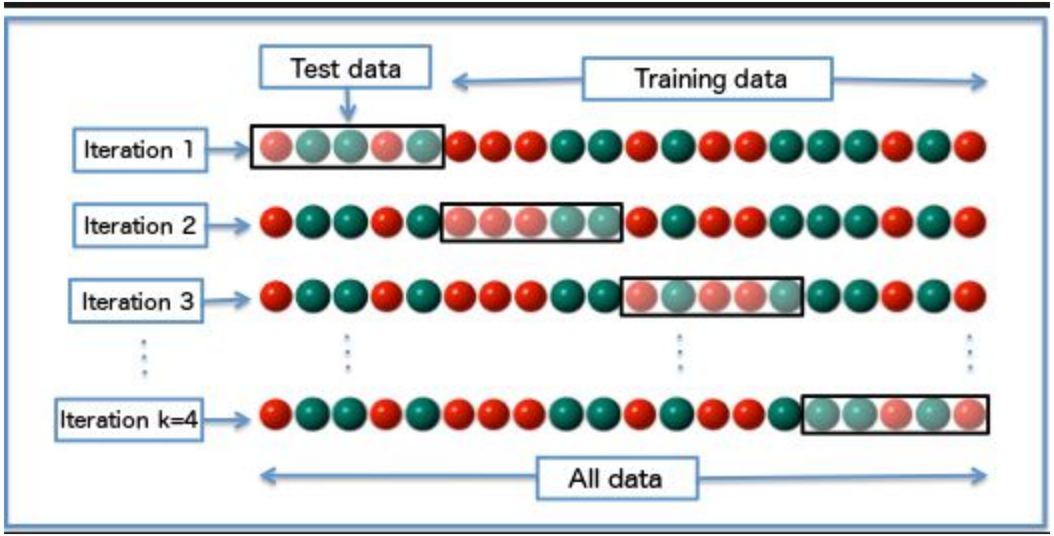
Hyperparameters vs parameters

- 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

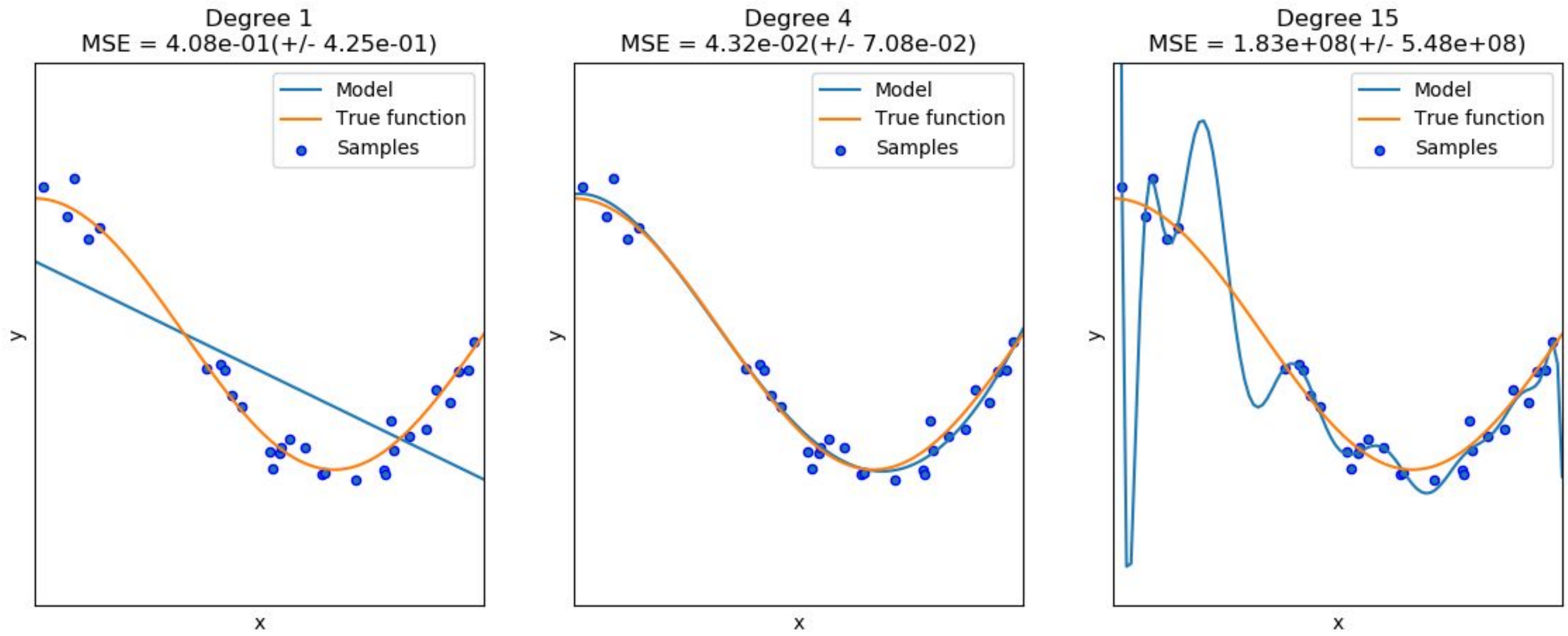


Cross-validation: is you model robust?

- Train/test split
- K-folds cross validation



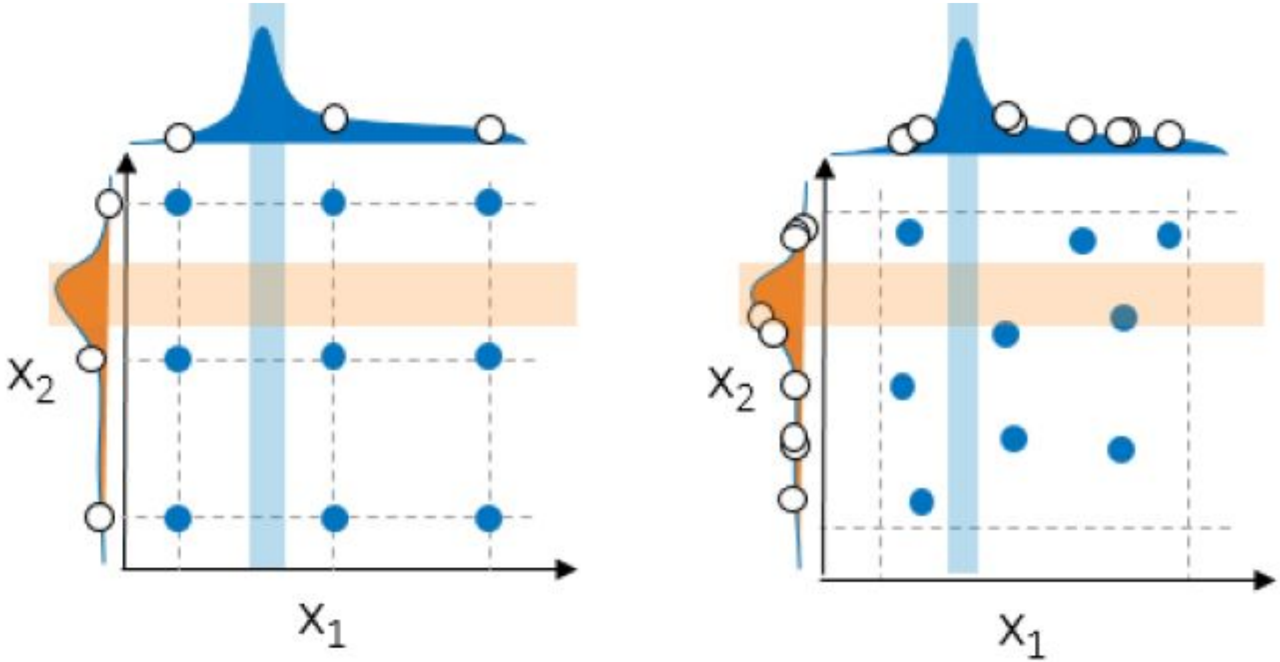
Overfitting / underfitting



Underfitting
Model doesn’t have enough (hyper-)parameters to describe data

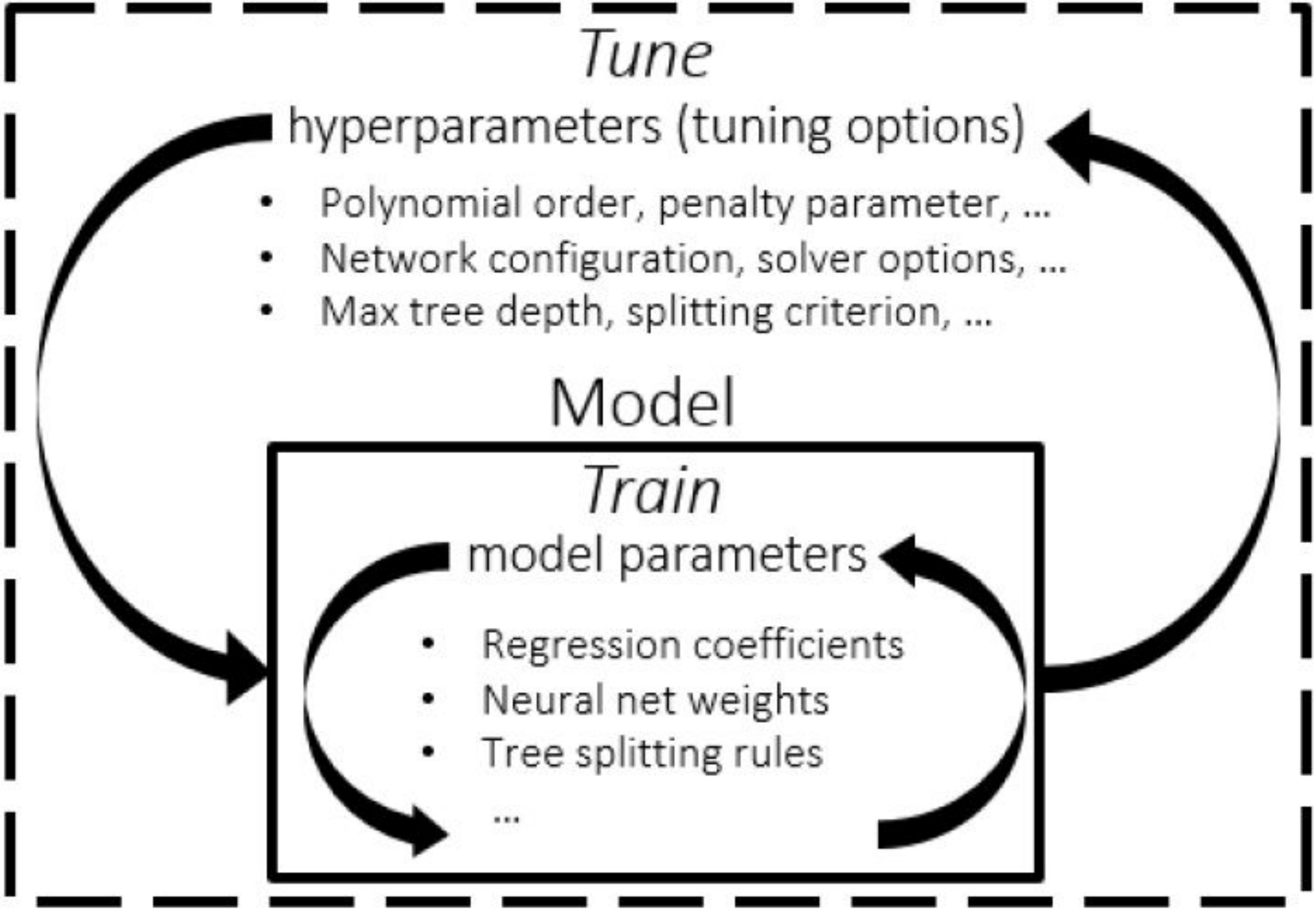
Overfitting
Model has too many (hyper-)parameters

Hyperparameter tuning



(a) Standard Grid Search (b) Random Search

Modeling Algorithm



Hands-on today

1. **Point your browser to:** https://yoga.to.infn.it
2. **Open a terminal:**
   - cd MLCourse-2122
   - 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