Machine Learning

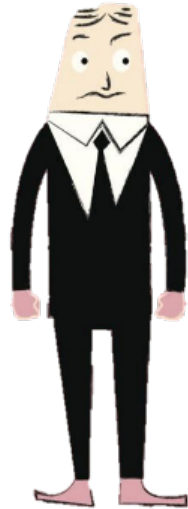## An introduction

by: Estefany Suárez & Jake Vogel

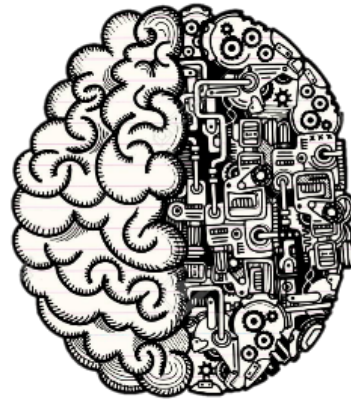29/06/2021

Machine Learning (ML)

Learn from experience
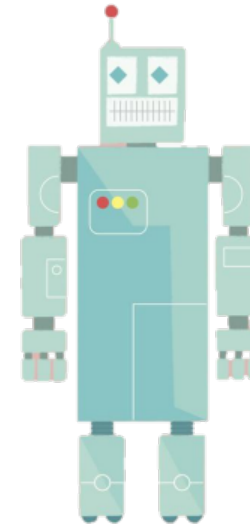
Follow instructions

DATA

Learn from ~~experience~~

Terminology

← *features* →

labels

samples
(observations)

X

Y

ML algorithms

learn a function to map *X* to *Y*
prediction

create useful
representations of data X

**Supervised
Learning**

**Semi-supervised
Learning**

**Unsupervised
Learning**
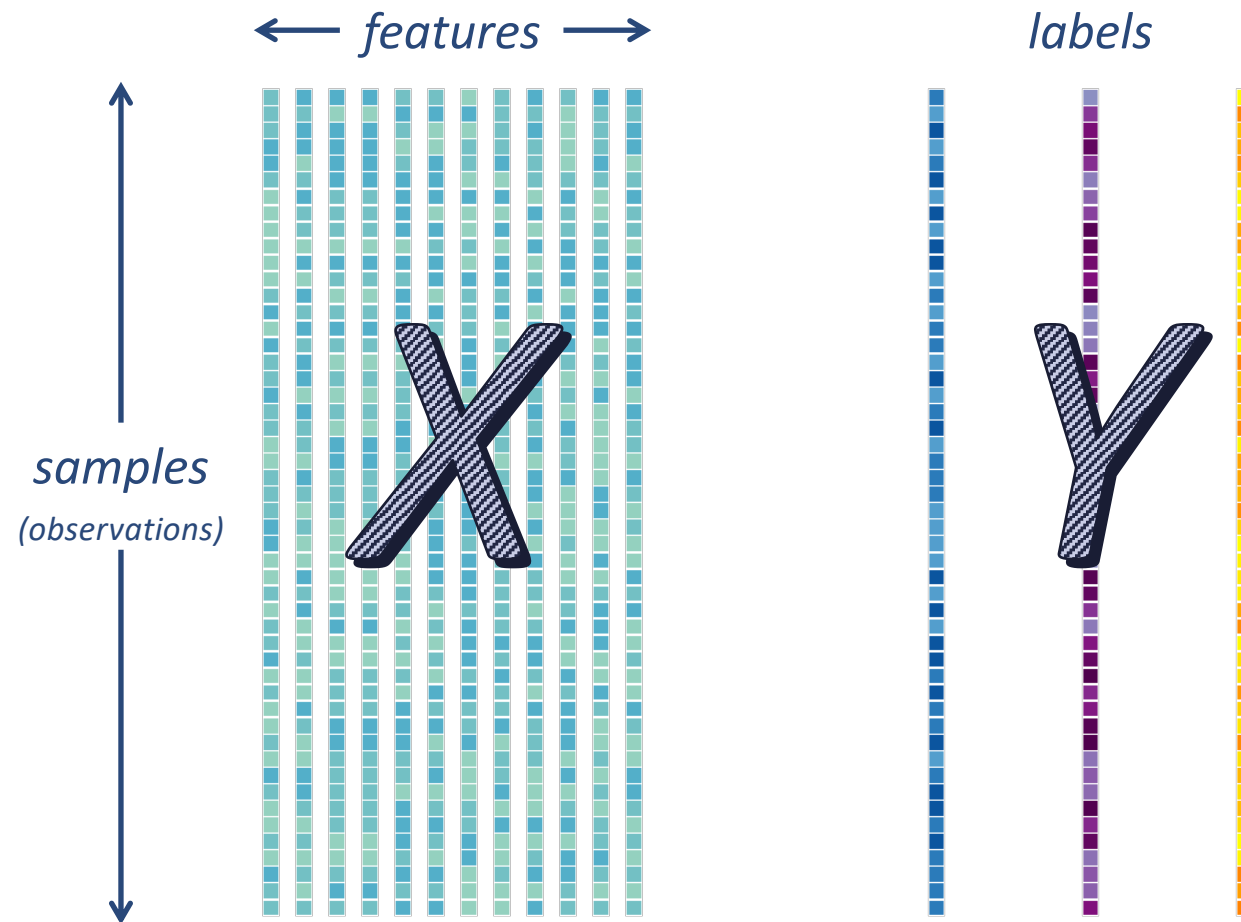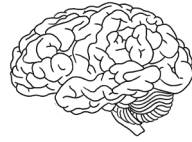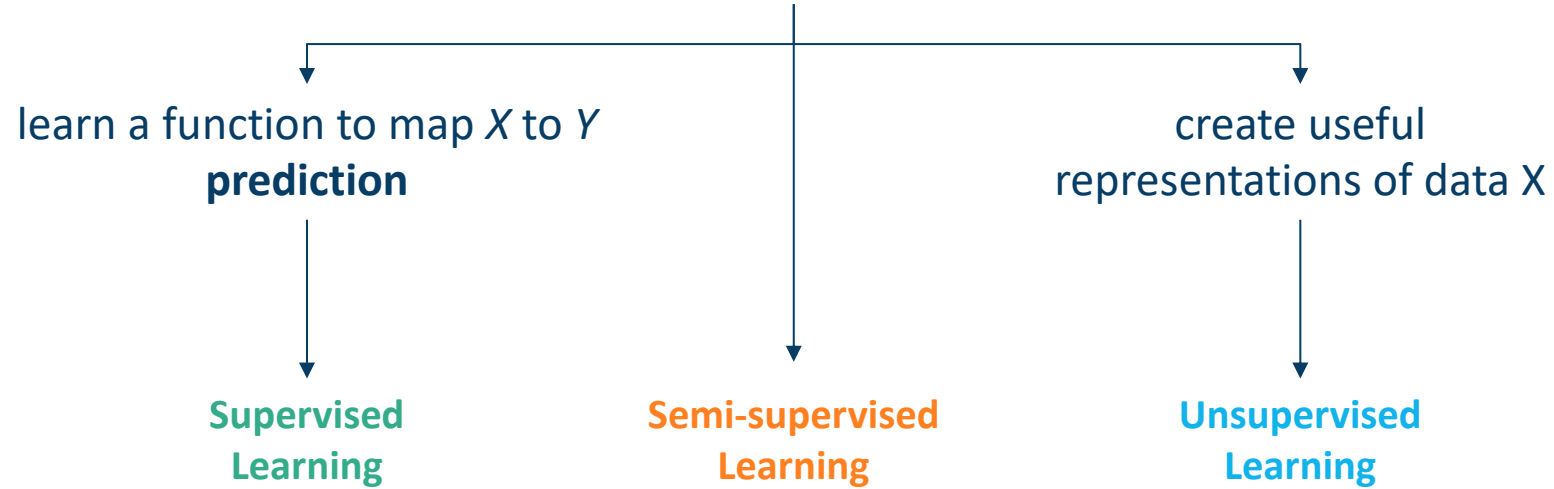
$$x , y$$

$$f : x \rightarrow y$$

$$y = f(x)$$

$$x$$

ML algorithms

learn a function to map *X* to *Y*
prediction

create useful
representations of data *X*

**Supervised
Learning**

**Unsupervised
Learning**

Discrete *Y* | Classification

Continuous *Y* | Regression

Clustering

Dimensionality
Reduction

Regression

$y = f(x)$

feature x
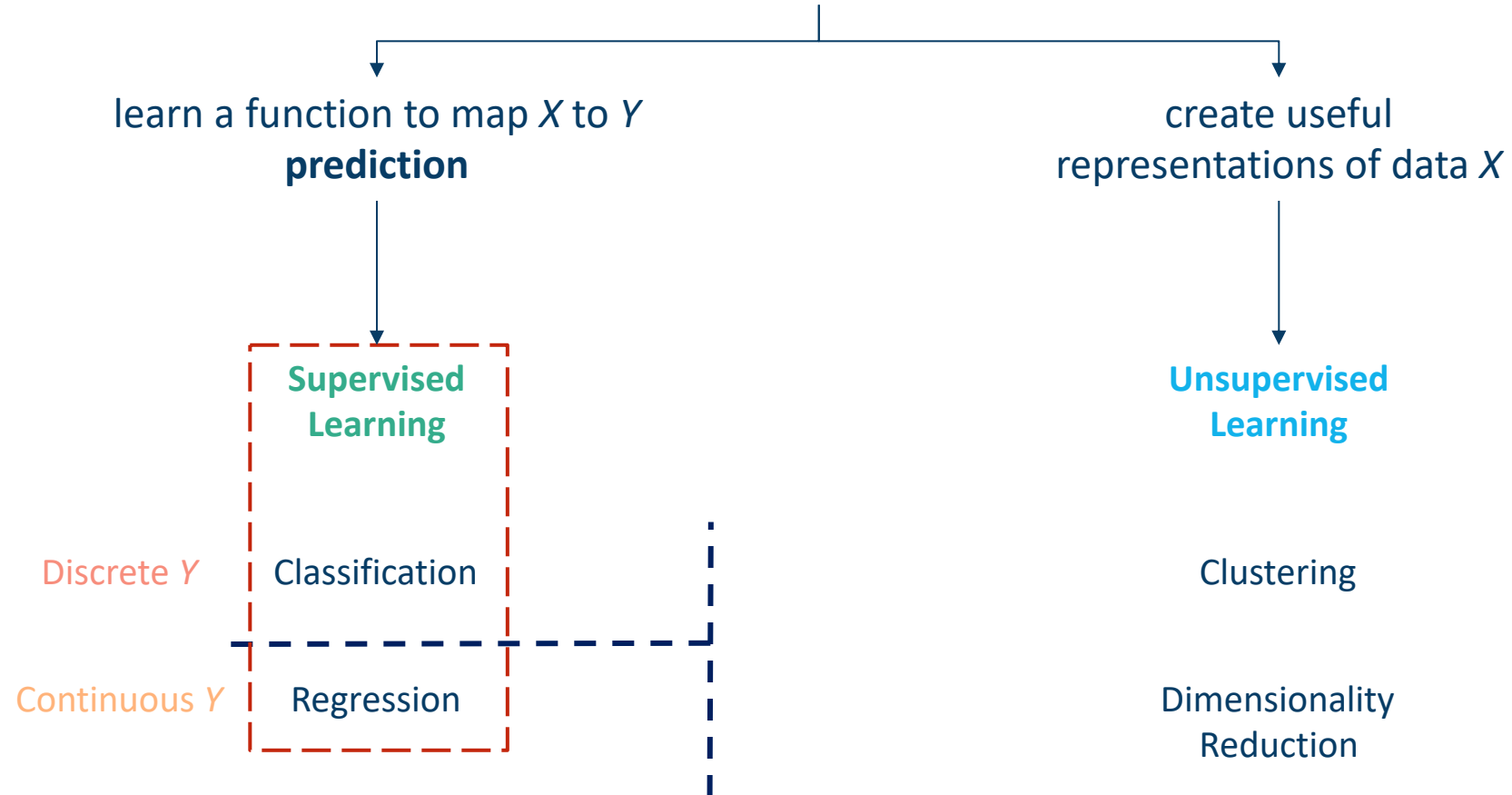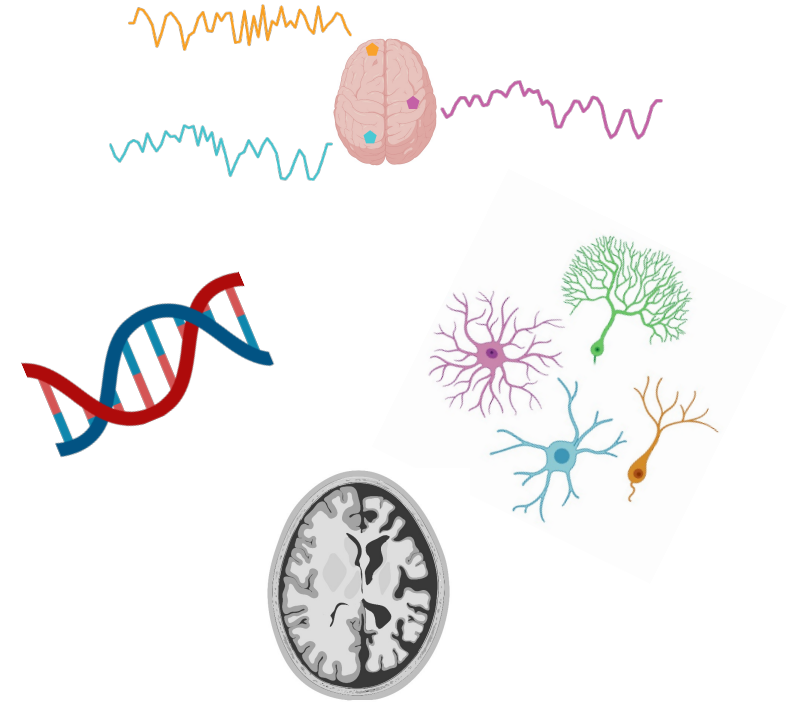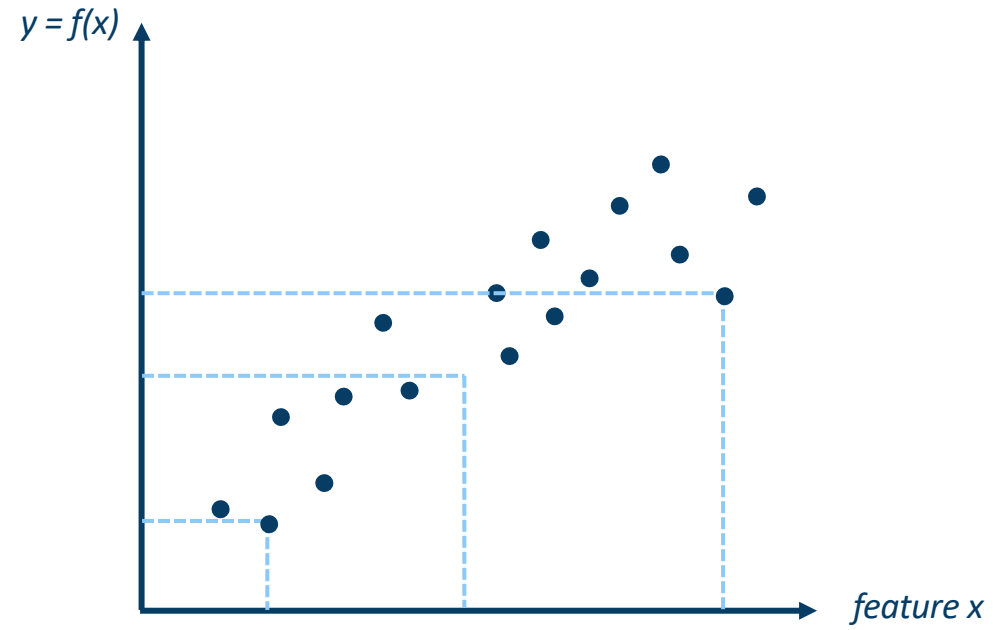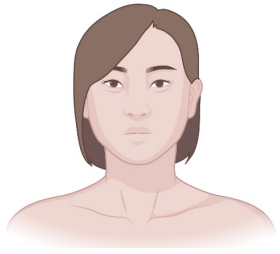
Regression

$$f: x \rightarrow y$$

$$y = f(x)$$
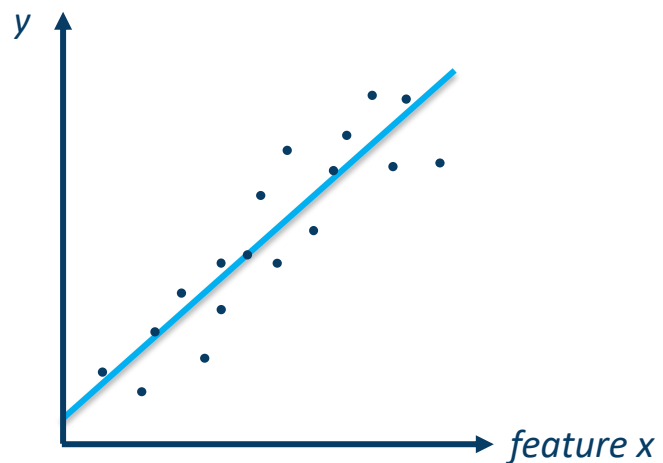
$$y = \boldsymbol{a}x + \boldsymbol{b}$$

$$y = \boldsymbol{a}x^2 + \boldsymbol{b}x + \boldsymbol{c}$$
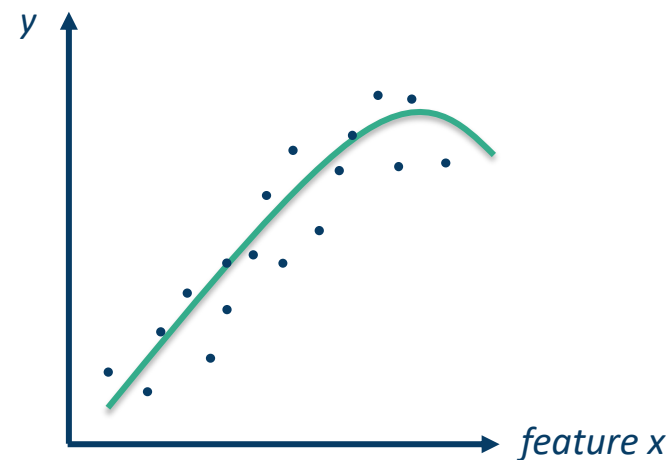
Typical ML Pipeline

Why cross-validation (CV)?
- Avoid overfitting
- Improve model generalizability

```
            ┌─────────────────────┐
            │  data exploration   │
            └─────────────────────┘
                      │
                      ▼
            ┌─────────────────────┐
            │   rnd. data split   │
            │   (whole dataset)   │
            └─────────────────────┘
                      │
          ┌───────────┴───────────┐
          ▼                       ▼
   ┌──────────────┐        ┌──────────────┐
   │ training set │        │   test set   │
   └──────────────┘        └──────────────┘
          │                       │
          ▼                       │
  ┌──────────────────┐            │
  │ parameters       │            │
  │ optimization     │            │
  │ (on training     │            │
  │  data only)      │            │
  └──────────────────┘            │
          │                       ▼
          ▼              ┌──────────────────┐
   ┌──────────────┐      │  evaluate/report │
   │ trained model│─────▶│ model performance│
   └──────────────┘      └──────────────────┘
```

model fitting
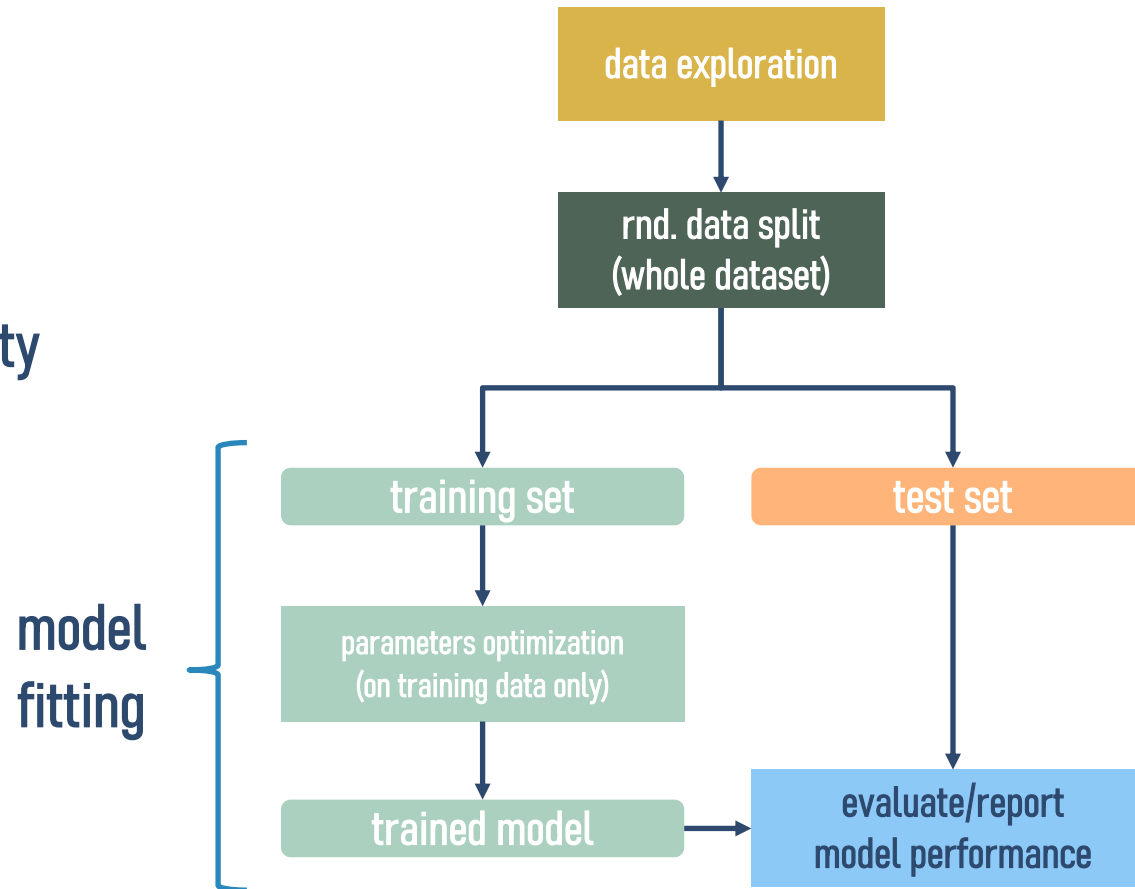
ML Basic Ingredients

$$f: x \rightarrow y$$
$$y = f(x)$$

f(x)

feature x

Training data	Test data

$x_{\text{train}}, y_{\text{train}}$ $x_{\text{test}}, y_{\text{test}}$
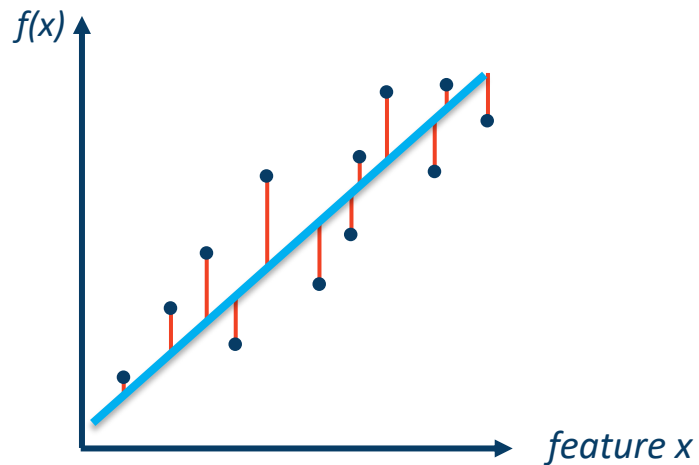
(1) Model or estimator $\longrightarrow$ $y = ax + b$
   $(f: x \rightarrow y)$

(2) Loss function $\longrightarrow$ $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

(3) Optimization algorithm $\longrightarrow$ gradient descent

9
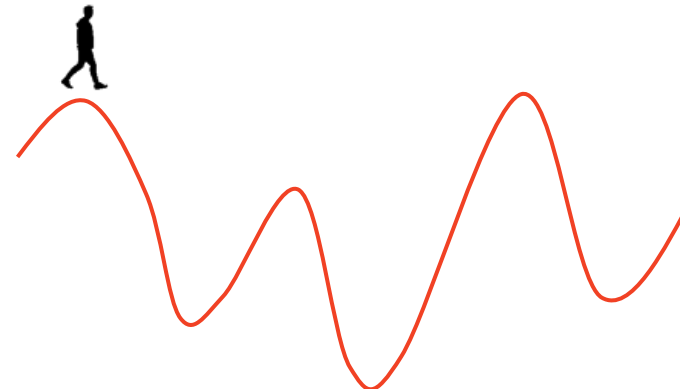
ML Basic Ingredients

① Model (*estimator*) selection

$$y = ax + b$$

linear regression

$$y = ax^2 + bx + c$$
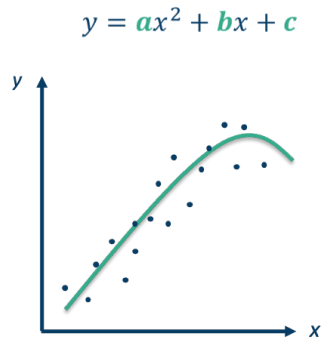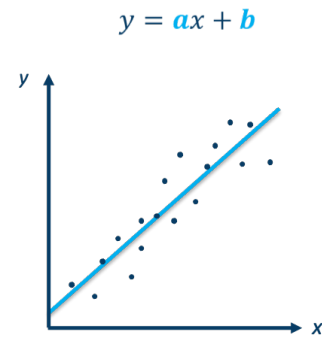
polynomial regression

multi-layer perceptron (ANN)
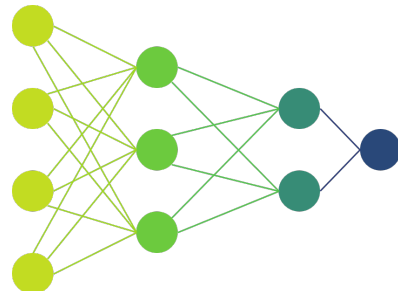
② Loss *(cost)* function

$$\text{mean squared error} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$$

$$\text{mean absolute error} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y}_i|$$

$$\text{max error} = \max(|y_i - \widehat{y}_i|)$$

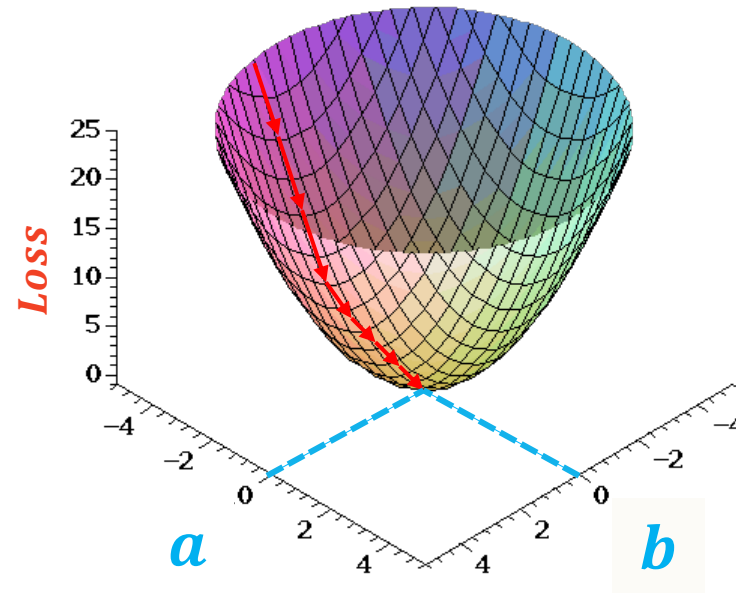$$\text{explained var.} = 1 - \frac{\text{Var}(y - \widehat{y})}{\text{Var}(y)}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})}$$

ML Basic Ingredients

③ Optimization algorithm for model parameters!

$$y = ax + b$$

$$Loss = f(a, b)$$

Gradient Descent

$$x_{\text{train}} \qquad y_{\text{train}}$$
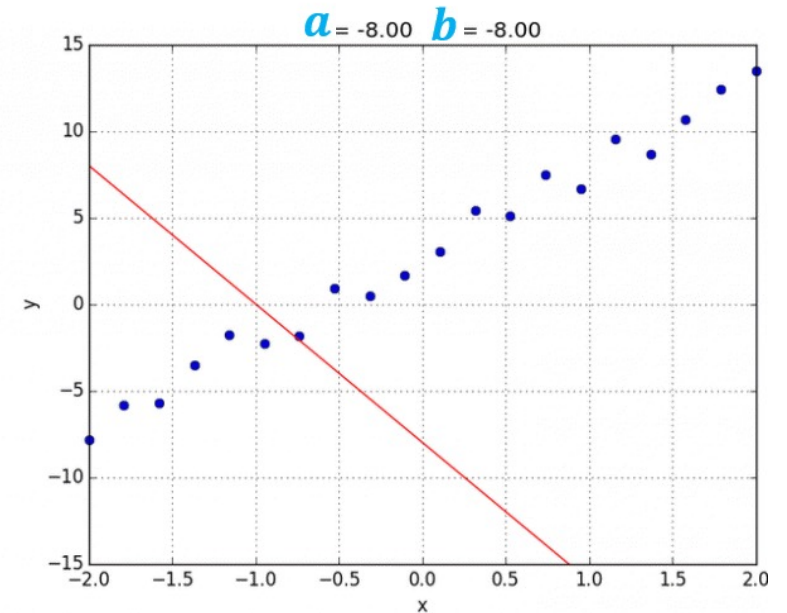
Initialize model parameters (a,b) and
iterate between:

1) Estimate output $\qquad \widehat{y}_i = ax_i + b$

2) Compute *loss* $\qquad$ MSE $= \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$

3) Compute the gradient

4) Update parameters (a,b)

MSE $= 370.77$

$a = -8.00 \quad b = -8.00$

Figure adapted from: https://giphy.com/gifs/gradient-O9rcZVmRcEGqI

12
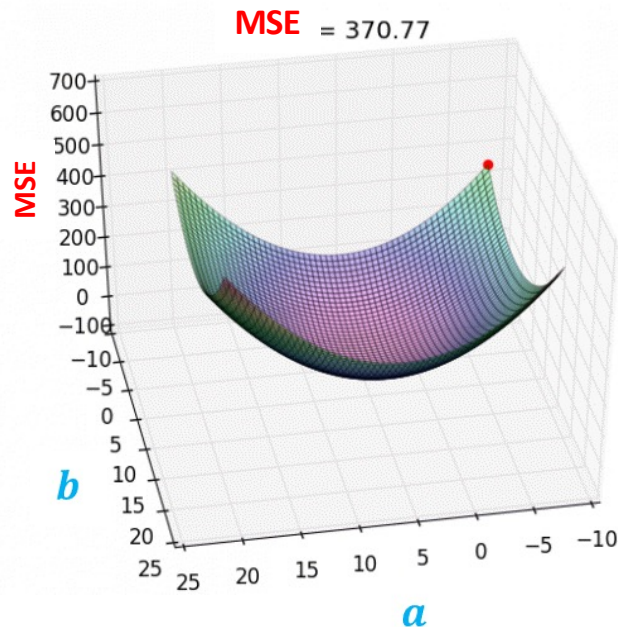
There is a problem though …

simple loss function
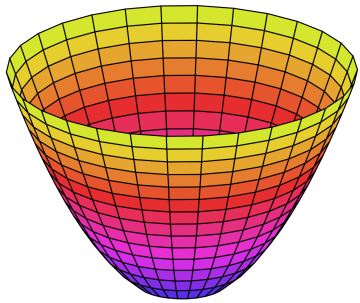
global
minimum

convoluted loss function
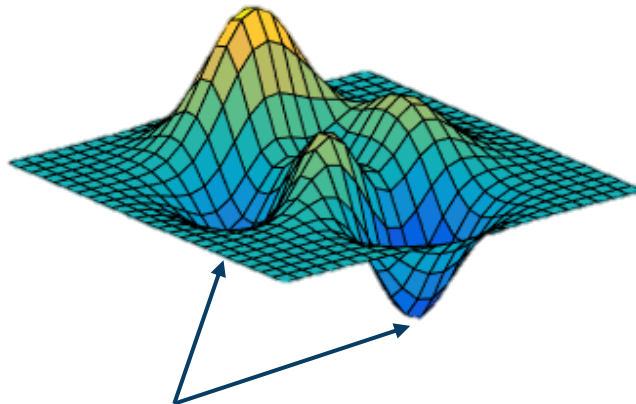
multiple local
minima

Me
WANTS
THE
DATA

*

? ? ?

Re-use your data . . .

multiple training epochs !

Typical ML Pipeline

Model Validation

Test data

x_{test} y_{test}

$\widehat{y}_i = ax + b$

$\widehat{y}$ y
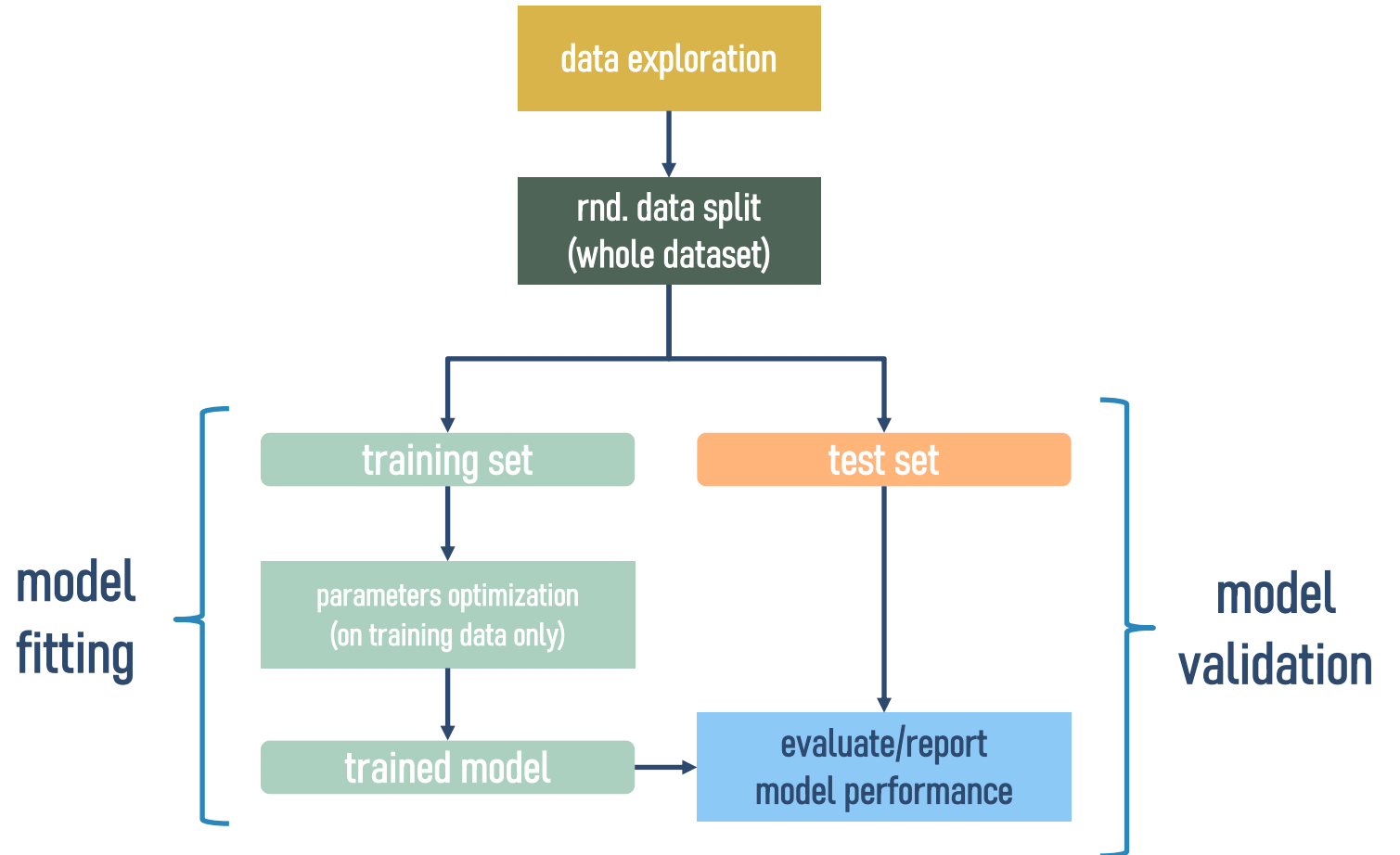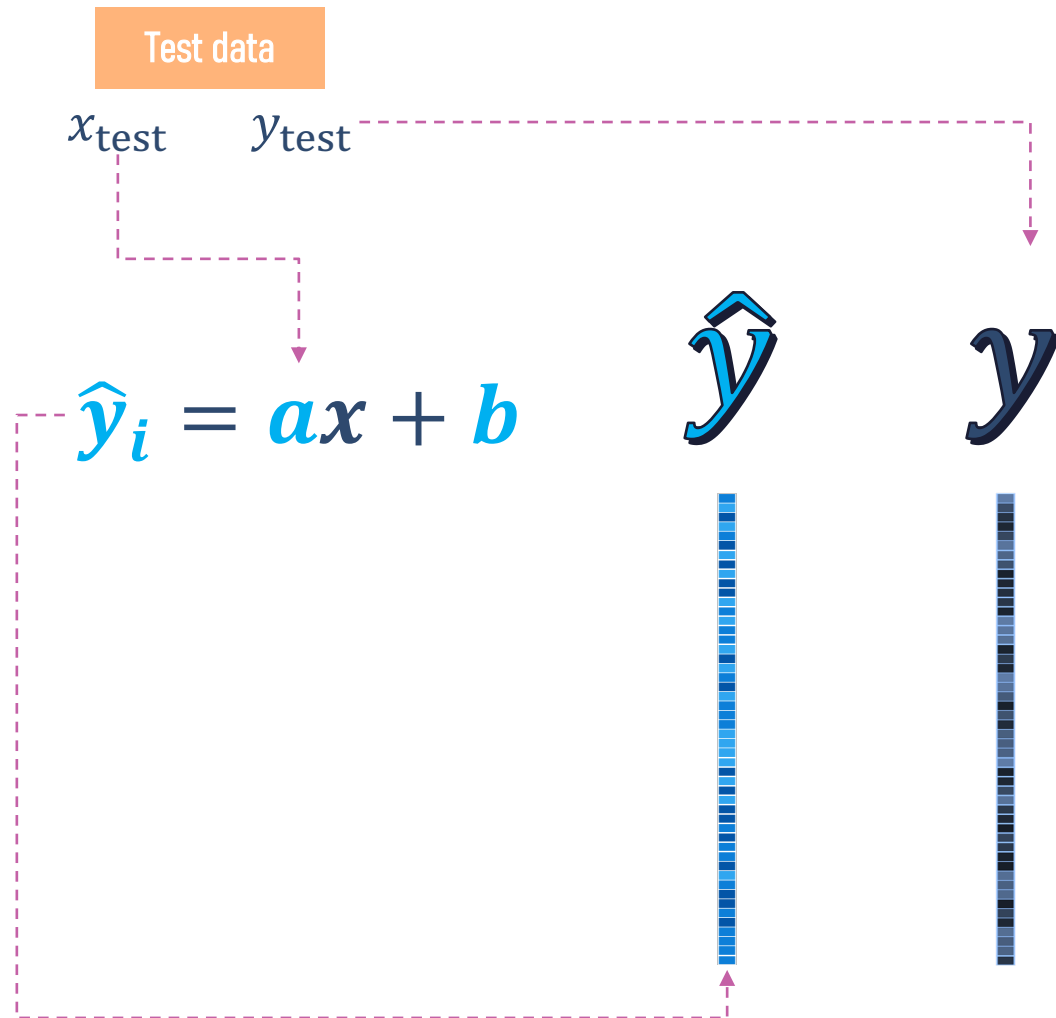
score function

mean squared error $= \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$

mean absolute error $= \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y}_i|$

max error $= \mathbf{max}(|y_i - \widehat{y}_i|)$

explained var. $= 1 - \frac{\mathbf{Var}(y - \widehat{y})}{\mathbf{Var}(y)}$

$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})}$

We have yet another problem …

All data

Training data | Test data

$x_{\text{train}}, y_{\text{train}}$ $x_{\text{test}}, y_{\text{test}}$
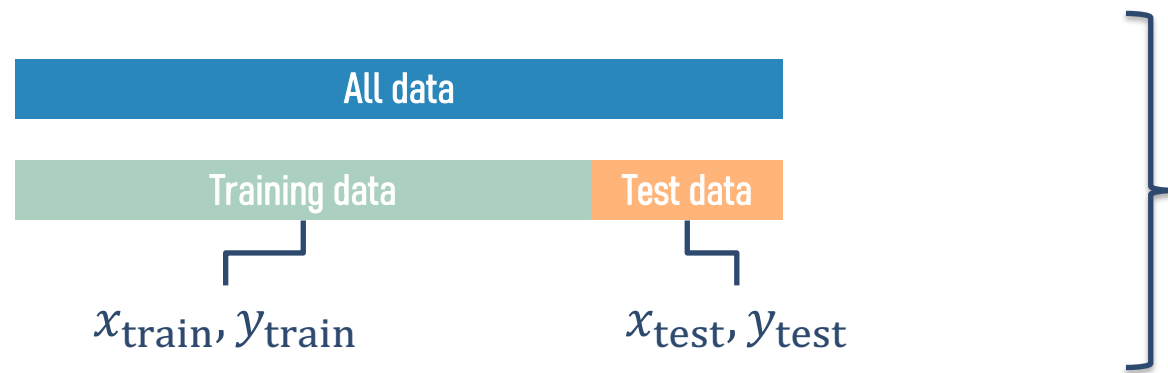
BUT … What if your data is very heterogeneous and the *"outlier"* samples/observations happen to be in the test set?

This would greatly affect the generalizability of your model!!!

Typical ML Pipeline

data exploration

data split
(whole dataset)

training set

validation set

test set

k^{th}- fold
cross-validation

parameters optimization

trained model

evaluate
model performance

pool predictions over
partitions

select best
model/hyperparameters

report
model performance
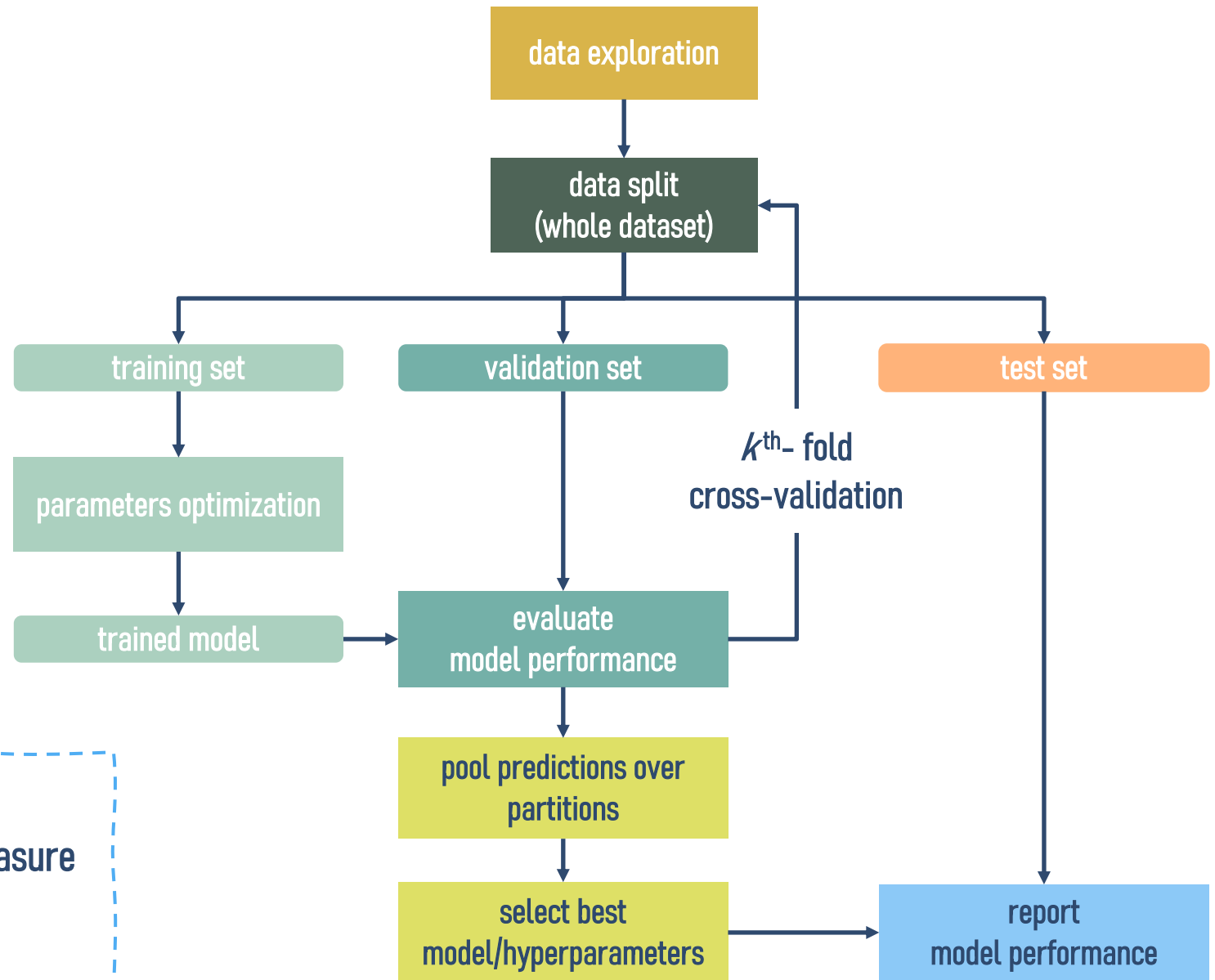
Why k^{th}-fold CV?

- Provide a more accurate performance measure

    [average, standard deviation, range]

- Model selection

- *Hyper*-parameter tuning

17

[*k-fold*] Cross-Validation (CV)

All data

Training data | Test data

k = 4 folds

| Fold 1 | Fold 2 | Fold 3 | Fold 4 |

Fold 1 Valid. data

Fold 2 Valid. data

Fold 3 Valid. data

Fold 4 Valid. data
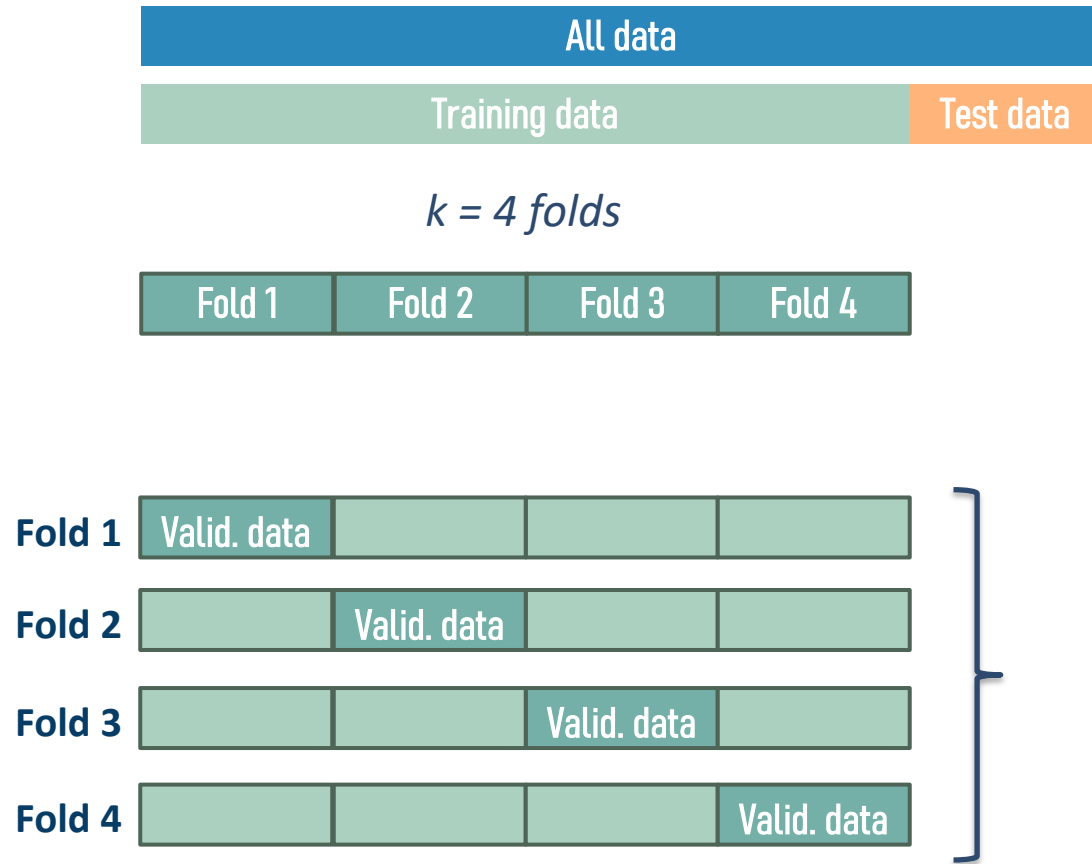
1. For each fold:

   I.   Train model parameters with **training** set

   II.   Evaluate training with **validation** set

2. Based on **validation** set, select *optimal* model and hyper-parameters

3. Only at the very, very **END**, report error on **test** set

Diagnosing Overfitting with *Learning Curves*

k^{th}-fold

How many TE?

underfitting

overfitting

Loss

validation

training

Training Epochs (TE)

Bias-Variance Trade-Off

Underfitting

Optimum

Overfitting
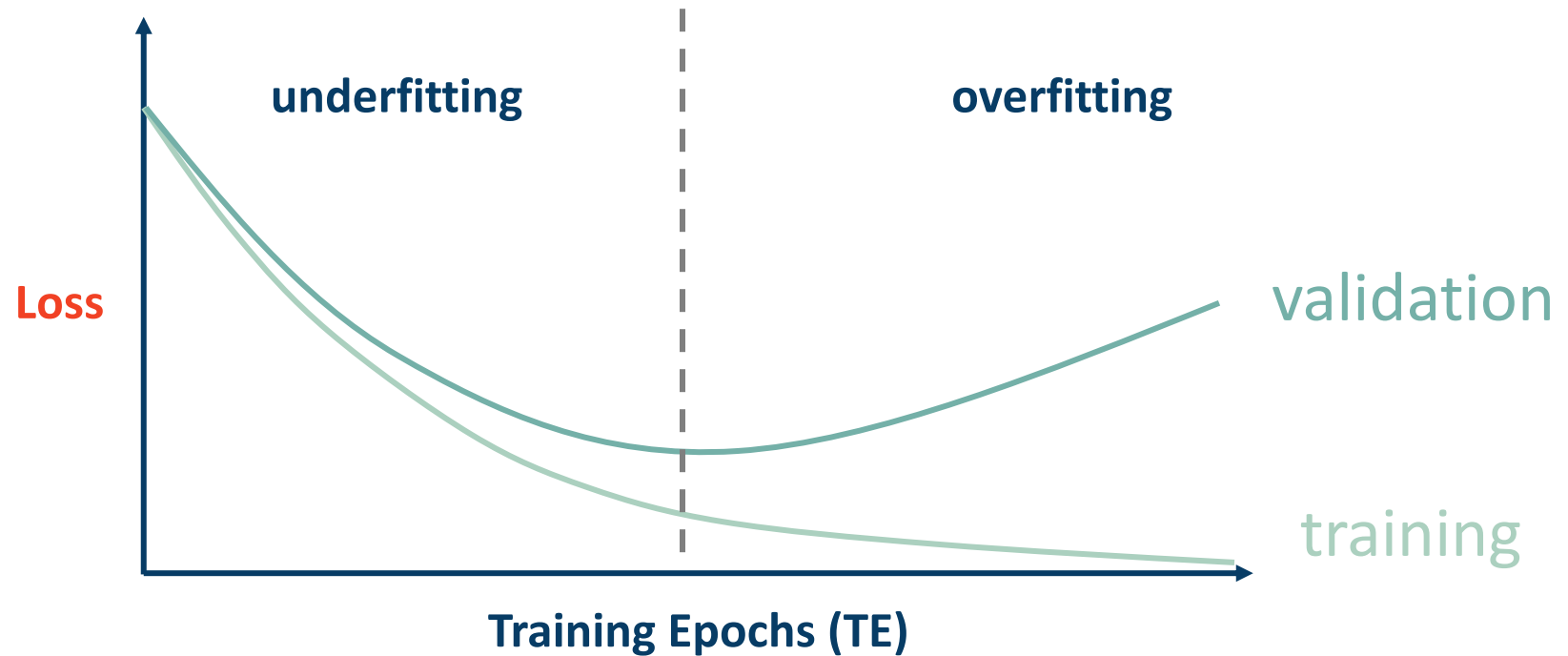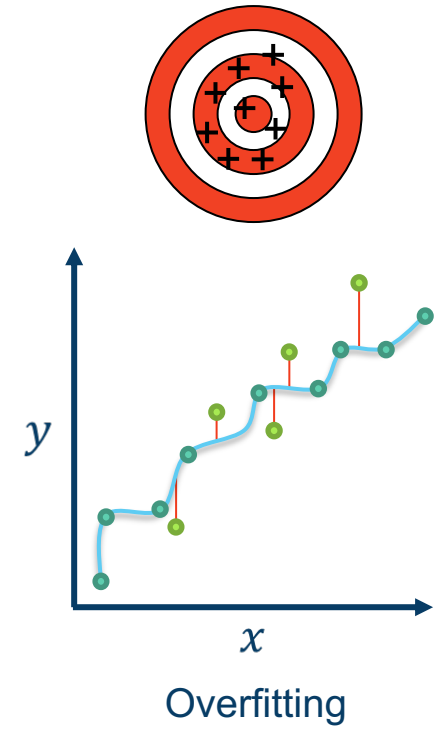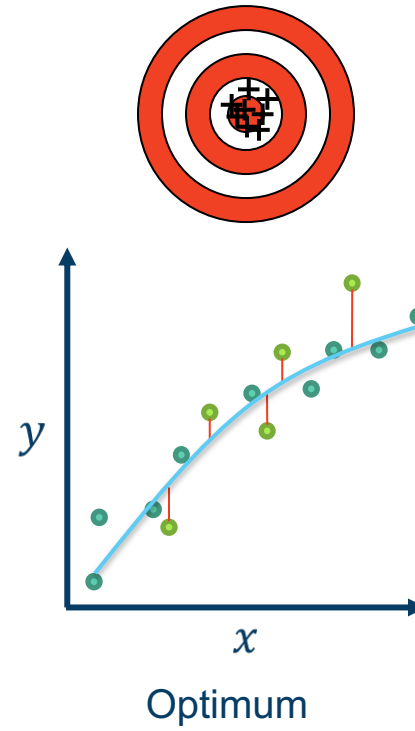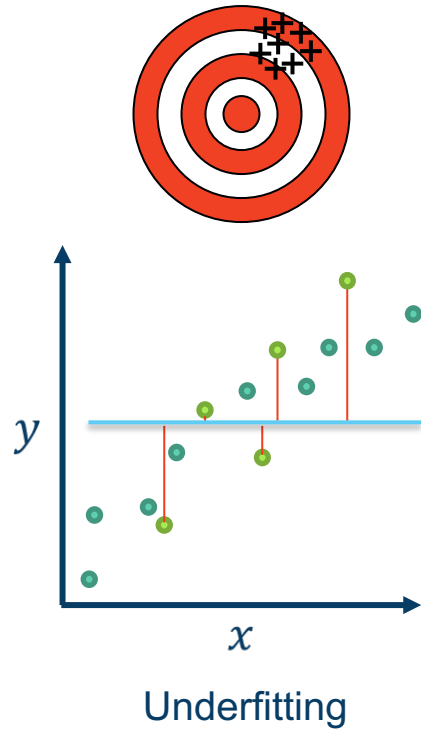
Training set   Test set

Bias-Variance Trade-Off

$$\text{Error}_{val./test} = bias + variance$$

underfitting

overfitting

error rate

Bias

Variance

model complexity (~ number of features)

learning epochs

Typical ML Pipeline

```
                          data exploration

                          data split
                          (whole dataset)

          training set      validation set              test set

          feature          apply same feature
          extraction/      extraction/selection    k^th- fold
          selection                                cross-validation

          parameters
          optimization

          trained model    evaluate
                           model performance

                           pool predictions over
                           partitions

                           select best model        report
                                                     model performance
```

dimensionality reduction

WHY DR ?
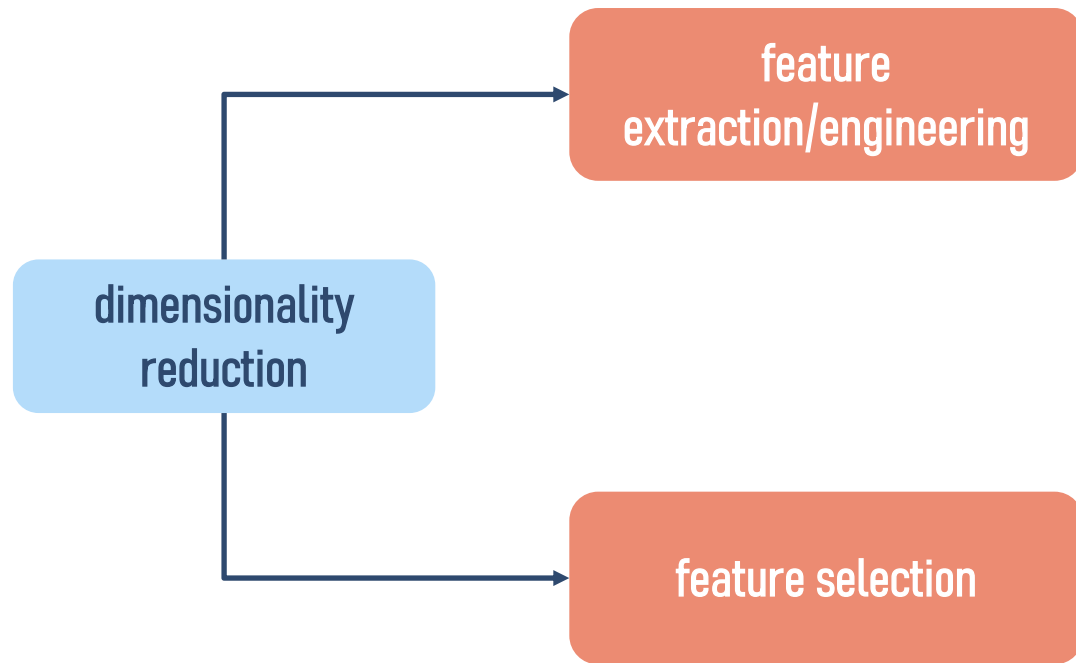- Curse of dimensionality (more features than samples)
- Intrinsic dimension may actually be small (redundant data)
- Extract "salient" features
- Remove noisy features
- VISUALIZATION!!!!

22

Tip 1: Dimensionality Reduction

dimensionality reduction

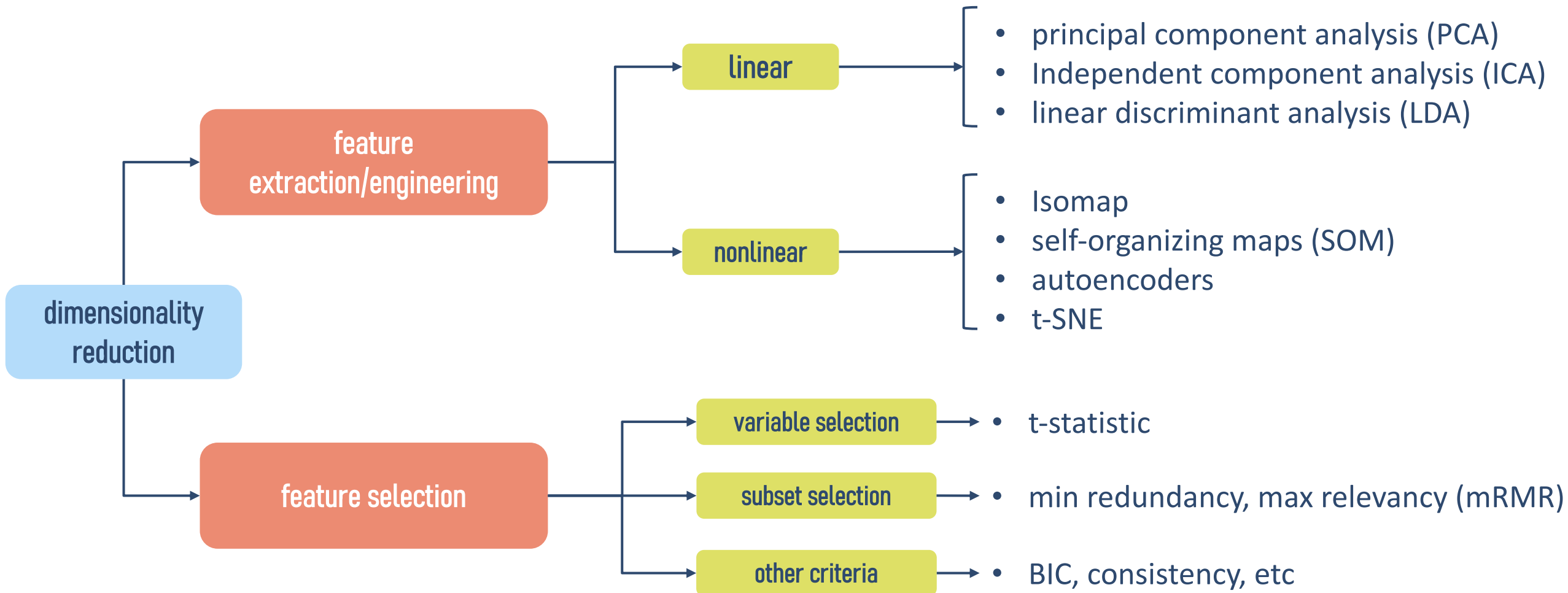feature extraction/engineering

feature selection

- Compact representation of the data
- Maps input features into a lower dimensional space

$$\text{e.g. If features are} \quad X = [x_1, x_2, x_3, \text{x}_4]$$
$$\text{then } Z = T(X) = [c_1 x_1 + c_2 x_2, \qquad x_3 * x_4]$$

- Selection of a subset of input features
- Features are still in original space

$$\text{e.g. } Z = S(X) = [x_2, \ x_3]$$

Tip 1: Dimensionality Reduction

dimensionality reduction

feature extraction/engineering

- linear
  - principal component analysis (PCA)
  - Independent component analysis (ICA)
  - linear discriminant analysis (LDA)
- nonlinear
  - Isomap
  - self-organizing maps (SOM)
  - autoencoders
  - t-SNE

feature selection

- variable selection → t-statistic
- subset selection → min redundancy, max relevancy (mRMR)
- other criteria → BIC, consistency, etc
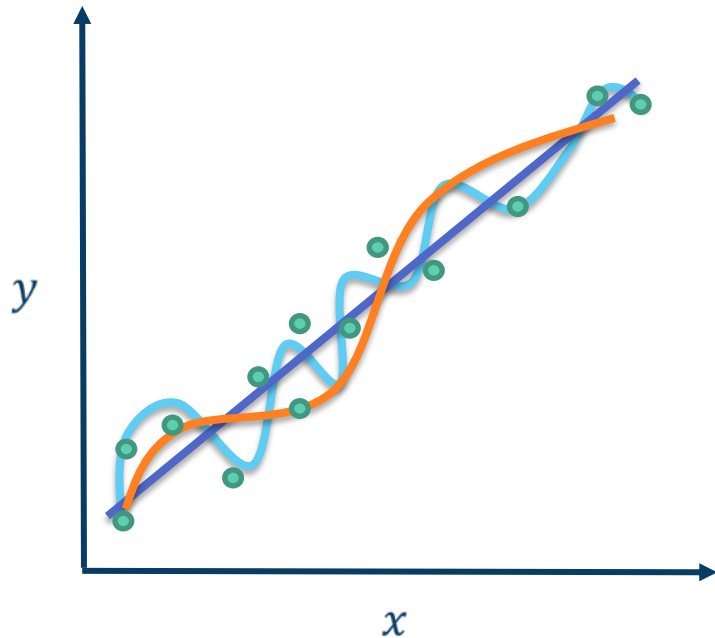
Tip 2: Regularization

$$y = \cancel{\beta_0} + \beta_1 x + \cancel{\beta_2} x^2 + \cdots + \cancel{\beta_p} x^p$$

Penalties on the *LOSS* function to prevent overfitting!

1) L1/Lasso: constrains parameters to be **sparse**
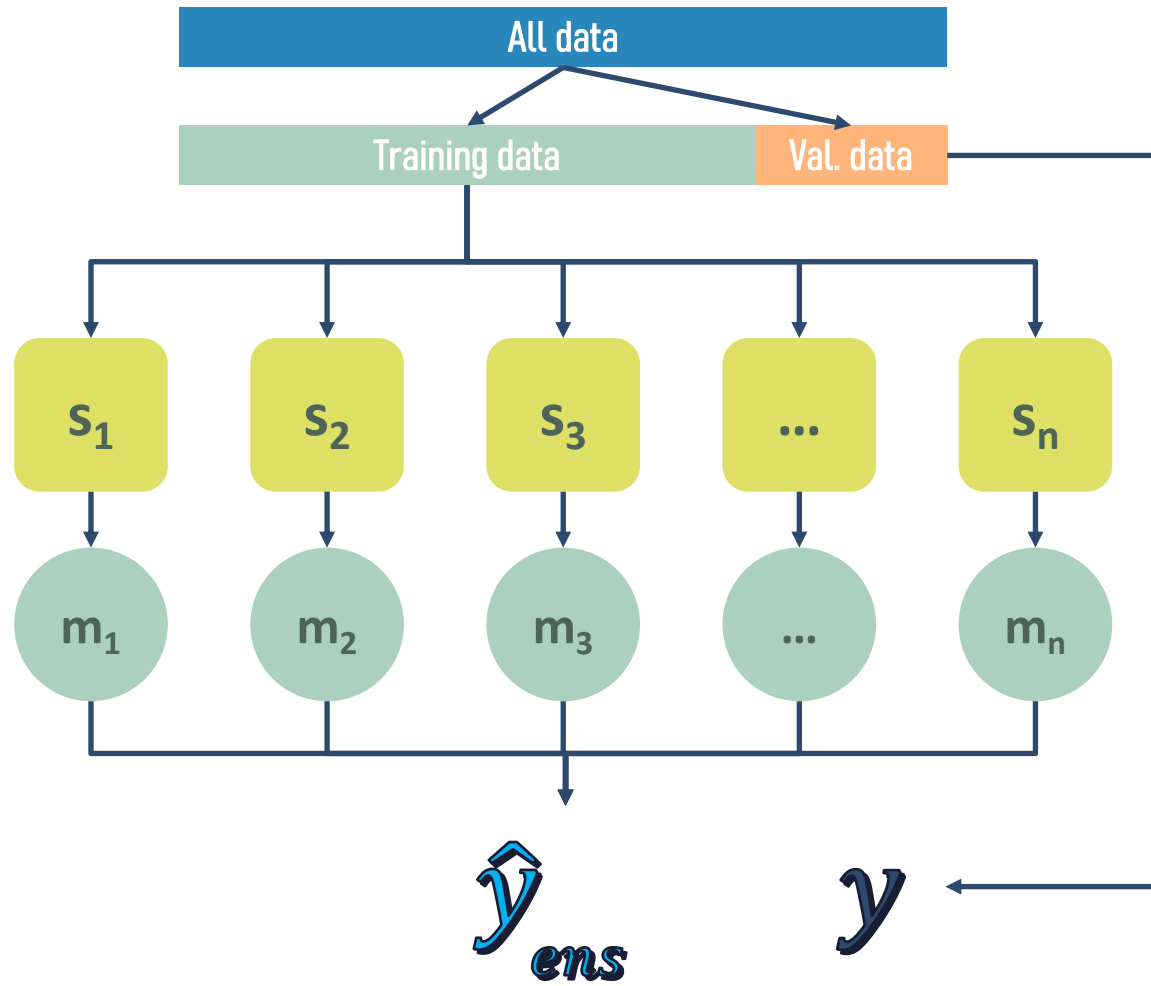
$$MSE = \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

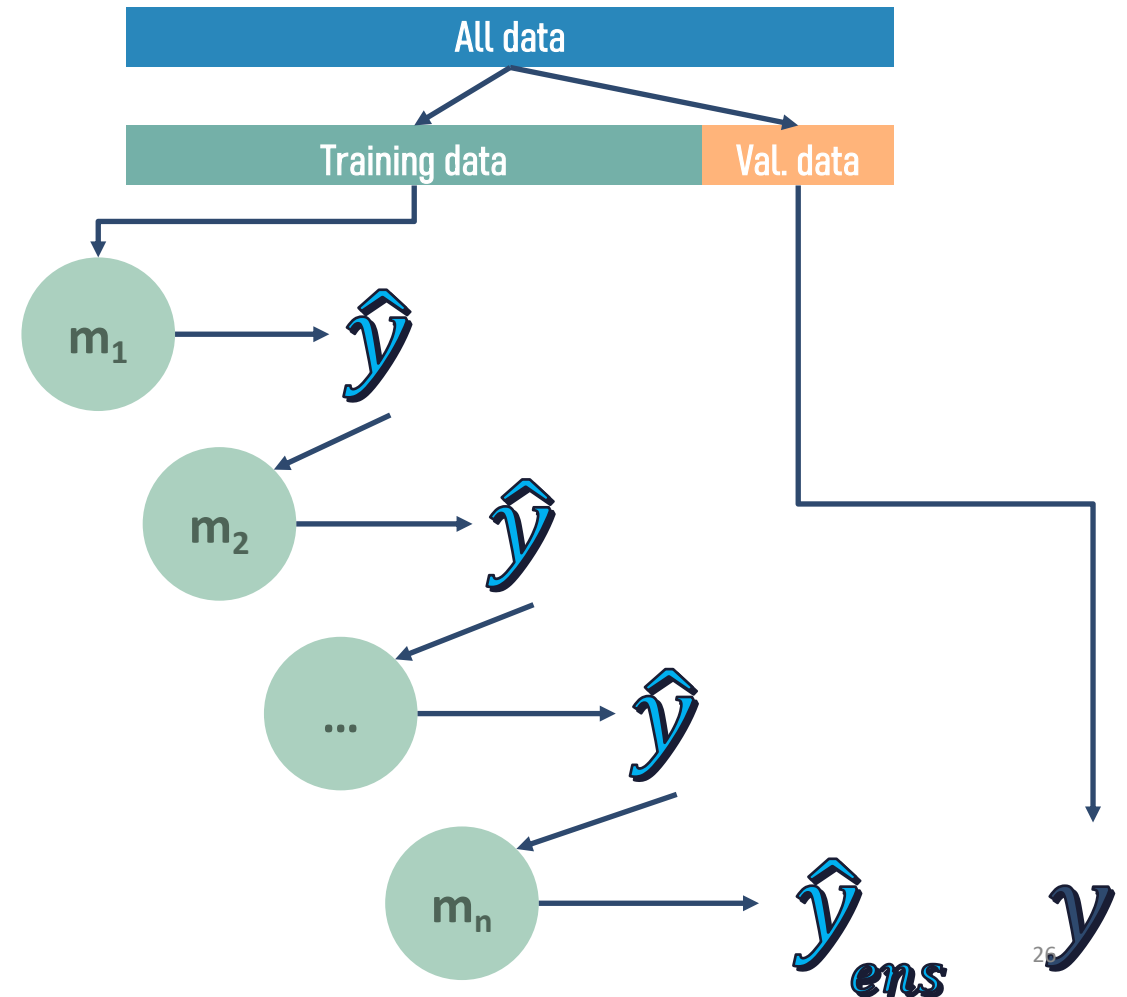2) L2/Ridge: constrains parameters to be **small**

$$MSE = \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Tip 3: Ensemble Methods

bagging = bootstrapp resampling + aggregation
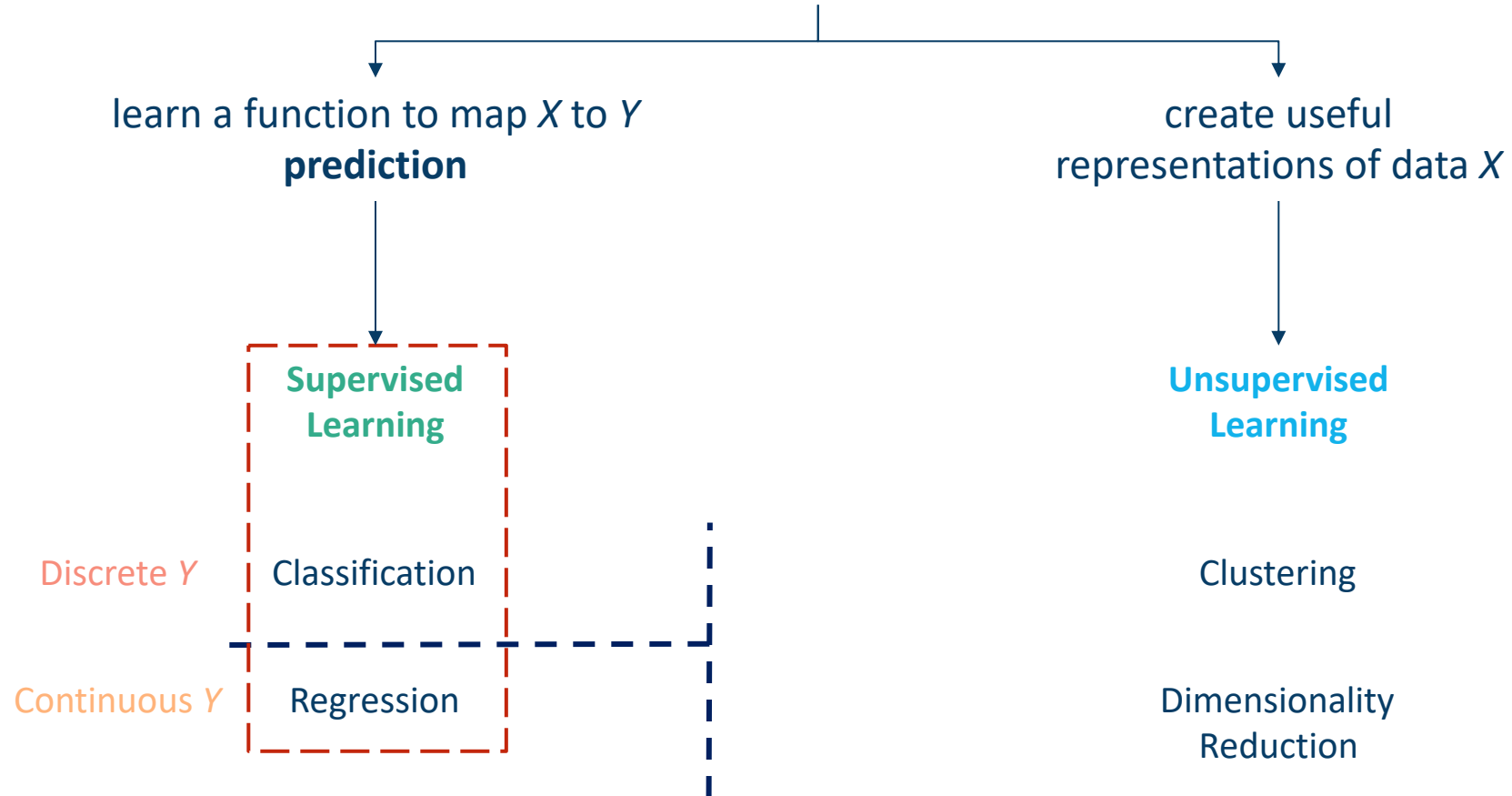
boosting

Sources of Bias

Type	DO NOT ☹	Sexy name	DO ☺
k-hacking	Try many *k*'s in *k*-fold CV (or different training %) and report only the best	*k*-hacking	Pick *k*=10, repeat It many times (>200 or as many as possible!), and report the full distribution (NOT boxplots!)
metric-hacking	Try **different performance metrics** (e.g., accuracy, F1, AUC, error rate, etc.) and report the best	*m*-hacking	Choose the most appropriate and recognized metric for the problem (e.g., AUC for binary classification)
feature/dataset-hacking	Try **subsets of feature(s) or subsamples of dataset(s)**, but report only the best	*d*-hacking	Use and report on everything: all analyses on all datasets

Machine Learning tools for neuroimaging … in Python!

ML algorithms

learn a function to map *X* to *Y*
prediction

create useful
representations of data *X*

**Supervised
Learning**

**Unsupervised
Learning**

Discrete *Y*   Classification

Clustering

Continuous *Y*   Regression

Dimensionality
Reduction

Classification

CAT

(LABELED PHOTOS)

DOG

OUTPUT

Classification

Is $x > 0.50$?

yes   no

Is $y > 0.50$?

yes   no

Is $x > 0.25$?
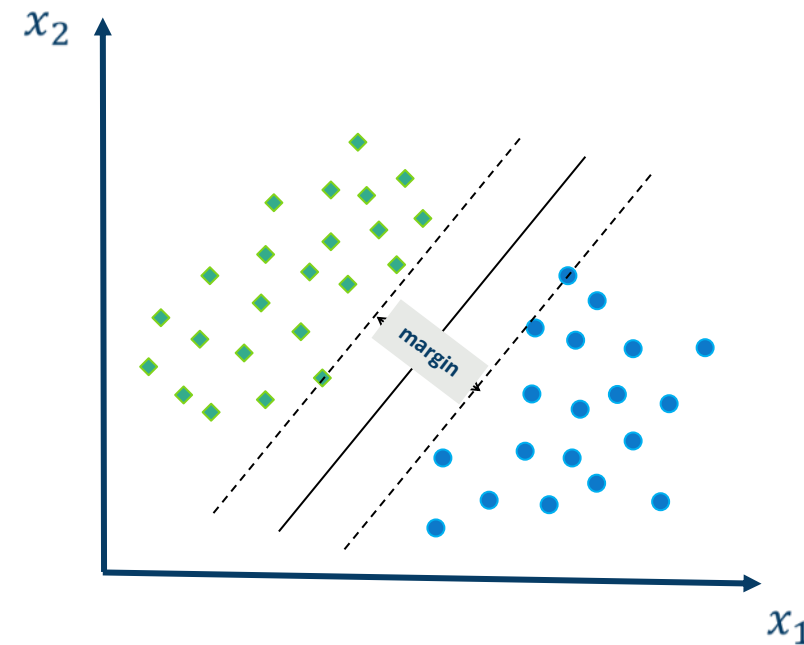
yes   no
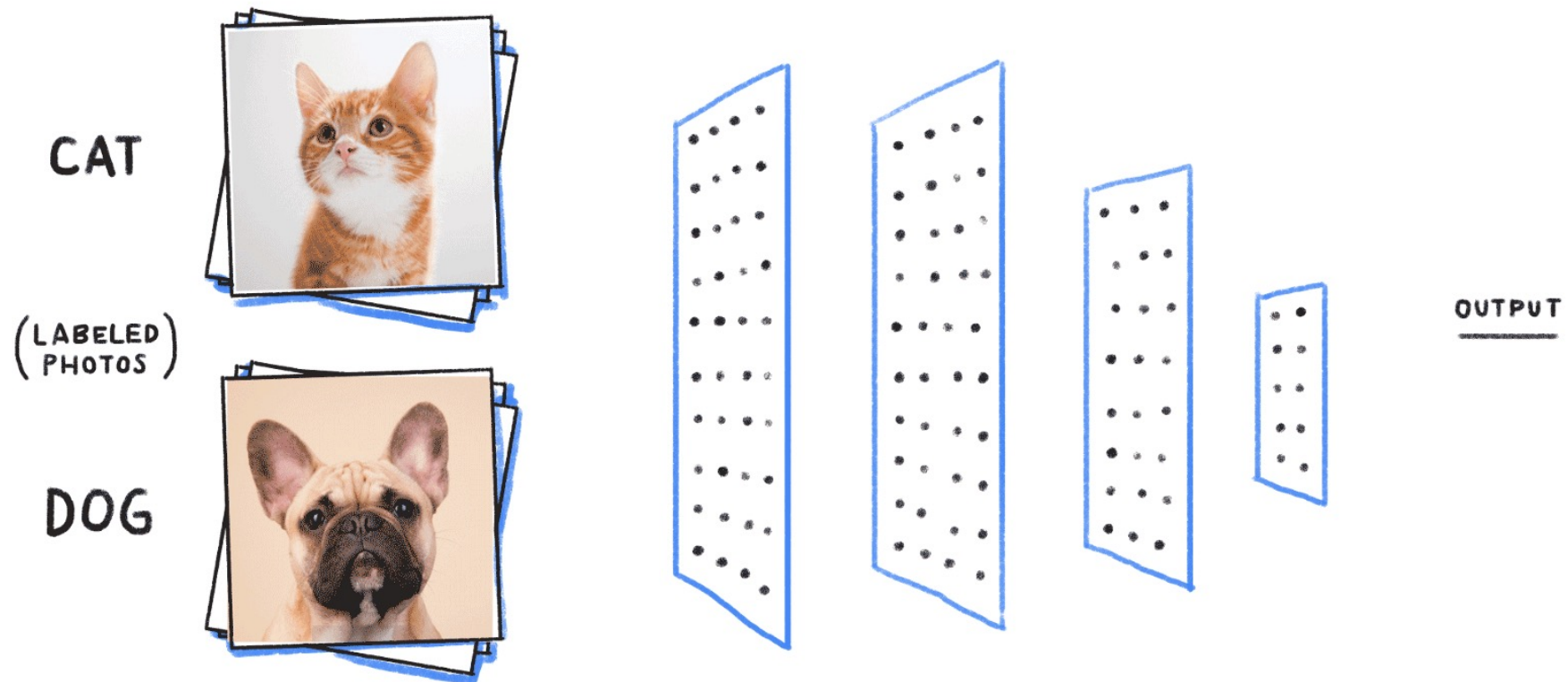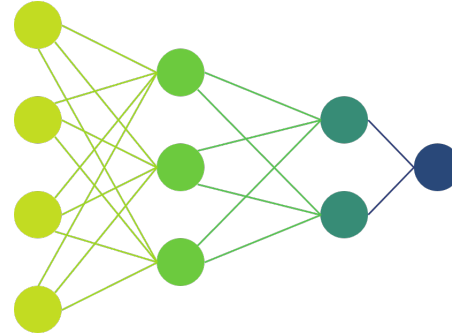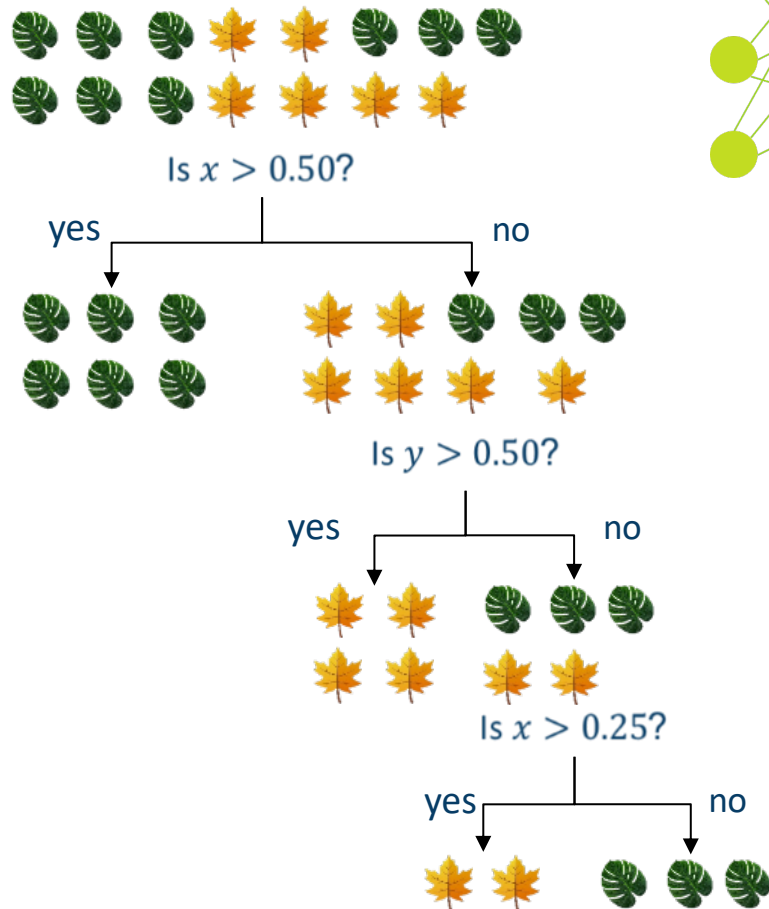
- Support Vector Machine (SVM)

- -

- Artificial Neural Networks
- Logistic regression
- Decision Trees
- Random Forests

probabilistic classifiers

x_2

margin

x_1

Performance Metrics – Binary Classification

confusion matrix

		True condition	
		POSITIVE	**NEGATIVE**
Predicted condition	**POSITIVE**	TRUE positive - T_p	**FALSE positive - F_p (Type I error)**
	NEGATIVE	**FALSE negative - F_n (Type II error)**	TRUE negative - T_n

score function

$$accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$precision = \frac{T_p}{T_p + F_p}$$

$$recall = \frac{T_p}{T_p + F_n}$$

Appropriate when classes are imbalanced!

$$F1\ score = \frac{2T_p}{2T_p + F_p + F_n}$$

Multiclass Prediction (≠ Multilabel Prediction)

		True class		
		CAT	**DOG**	**BIRD**
Predicted class	**CAT**	13	0	0
	DOG	0	10	6
	BIRD	0	0	9

- To extend a binary metric to multiclass problems, the data is treated as a collection of binary problems, one for each class.

- The binary metric is then averaged across the set of classes, each of which may be useful in some scenario.

Performance Metrics – Binary Classification

		True condition	
		POSITIVE	**NEGATIVE**
Predicted condition	**POSITIVE**	TRUE positive - T_p	FALSE positive - F_p (Type I error)
	NEGATIVE	FALSE negative - F_n (Type II error)	TRUE negative - T_n

Sensitivity (or recall)

$$TPR = \frac{T_p}{P} = \frac{T_p}{T_p + F_n}$$

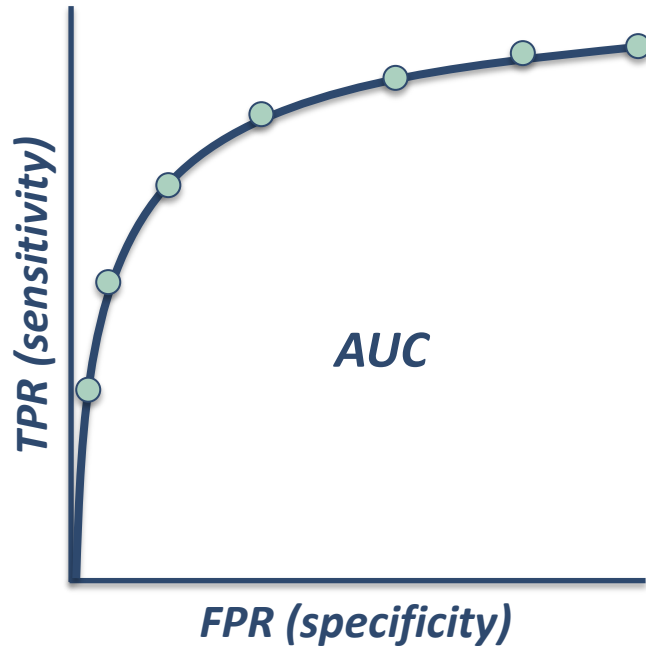$$FPR = \frac{F_p}{N} = \frac{F_p}{T_n + F_p}$$

Specificity (or selectivity)

$$FNR = \frac{F_n}{P} = \frac{F_n}{T_p + F_n} \qquad TNR = \frac{T_n}{N} = \frac{T_n}{T_n + F_p}$$
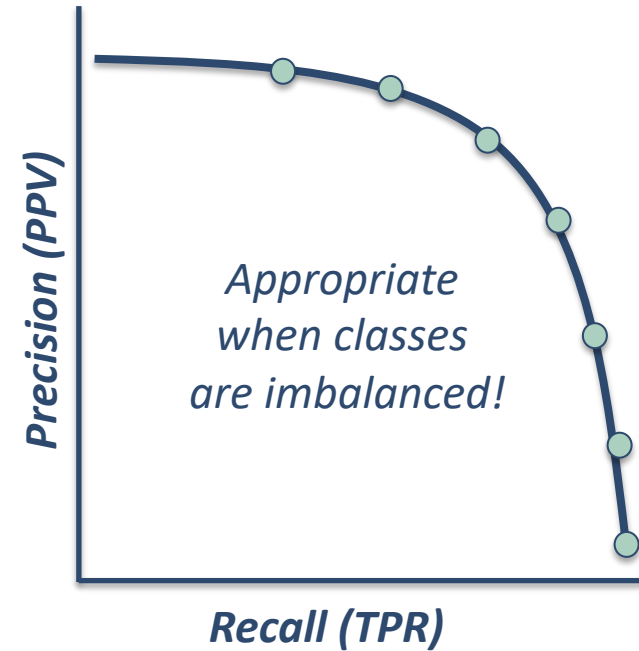
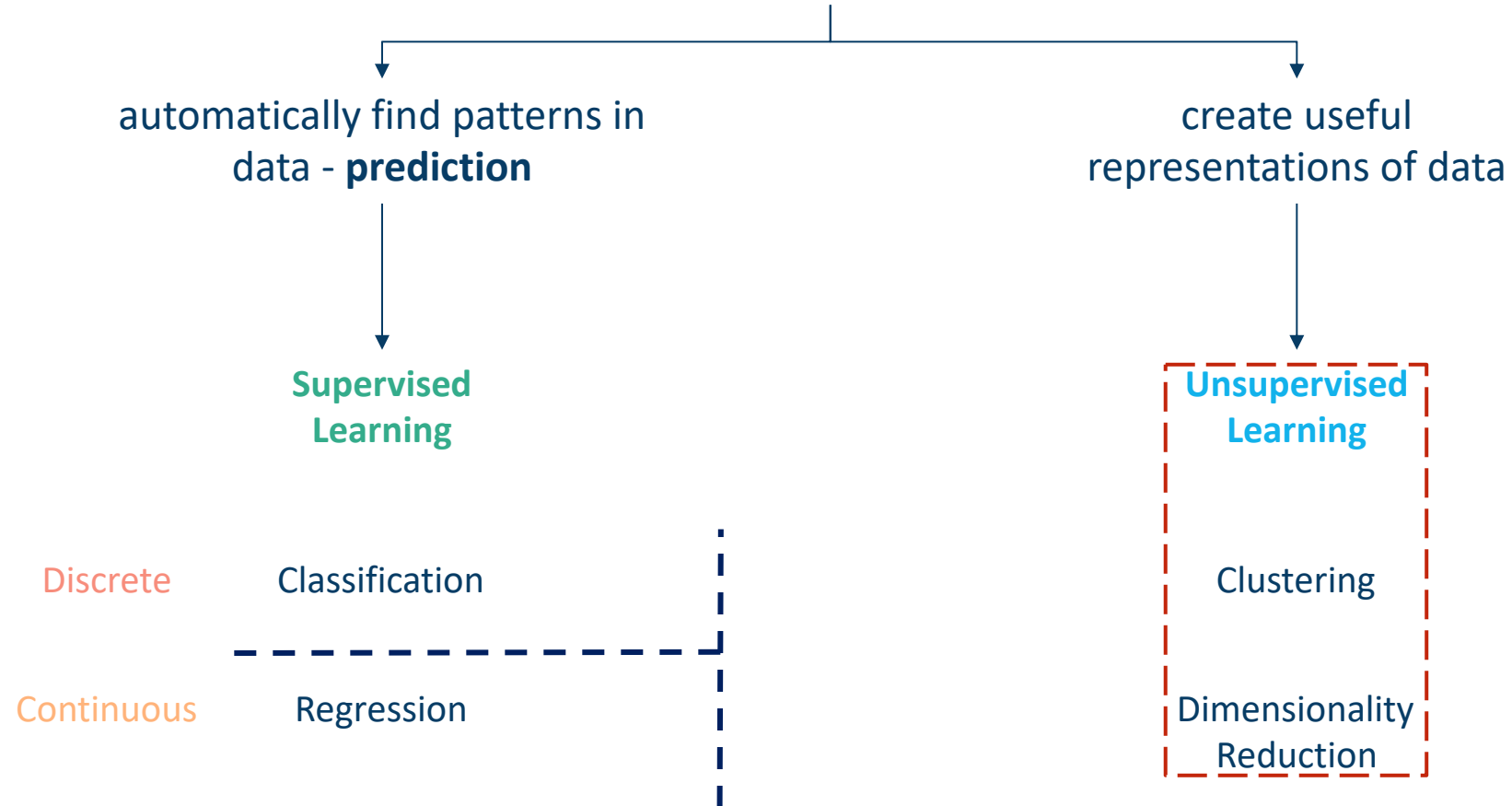# Performance Metrics – Binary Classification

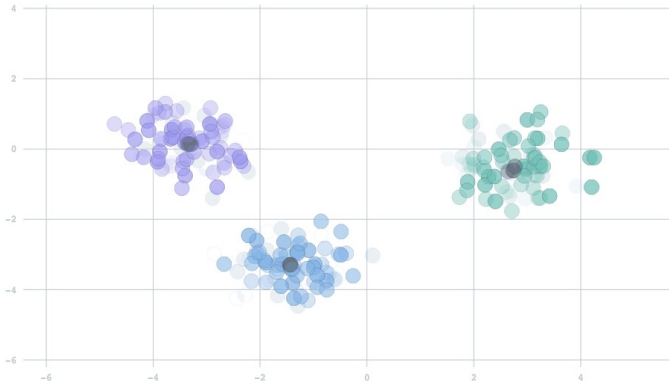*(probabilistic classifiers)*

ROC curve

precision-recall curve

TPR (sensitivity)

AUC

FPR (specificity)

Precision (PPV)

Appropriate when classes are imbalanced!

Recall (TPR)

ML algorithms

automatically find patterns in data - **prediction**

create useful representations of data

Supervised Learning

Unsupervised Learning

Discrete — Classification

Clustering

Continuous — Regression

Dimensionality Reduction
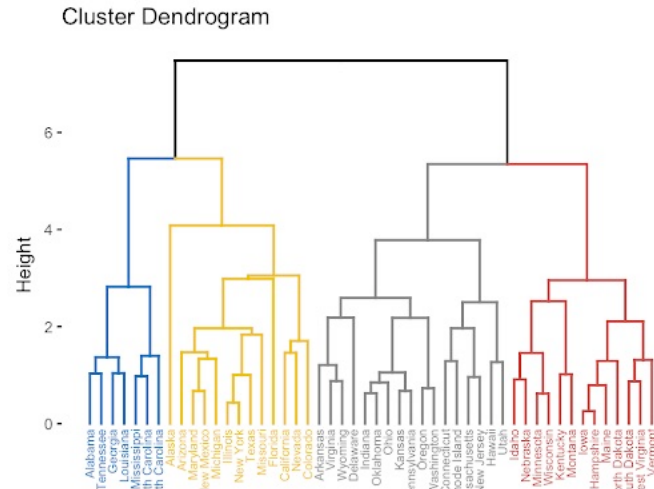
Clustering

K-means

Hierarchical clustering
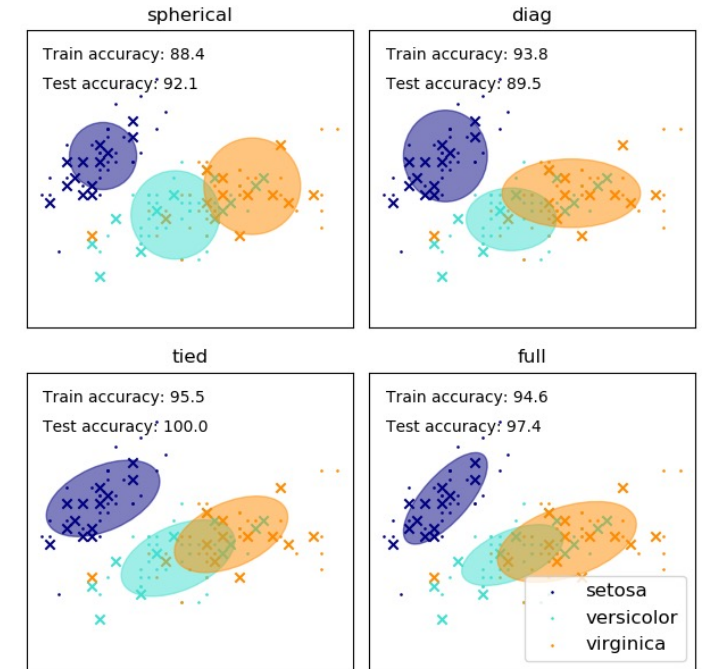
Gaussian mixture models

Inference vs *Prediction* – Linear Model $\mathbf{Y = \beta X + \epsilon}$

Statistics

Goal:
- Draw population inferences from a sample
- Identify significant contributing variables (statistical null-hypothesis testing, *p*-values, variance explained)

Uses:
- Scientific discovery. Ideal to uncover characteristics or true properties of the biological processes of the studied phenomenon.
- Useful to judge the individual relevance of each quantitative measure in impacting the response of interest.

Machine Learning

Goal:
- Find generalizable predictive patterns
- Identify most predictive variables set (out-of-sample prediction performance)

Uses:
- Pragmatic forecasting of biological processes.
- Tends to concern less about the data-generating process.

Diagnosing Features (≈ Interpretability)

Local Interpretable Model-Agnostic Explanations [*LIME*]

Model

Dataset and Predictions

Pick step

Explanations

Human makes decision

- Local perturbation of the input
- Estimate how the predictions change
- Generate an explanation by approximating the underlying model by an interpretable one, learned on local perturbations of (interpretable aspects of) the input.

sneeze
weight
headache
no fatigue
age

Flu

Explainer (LIME)

sneeze
headache
no fatigue

Data and Prediction

Explanation

(Tulio-Ribeiro, et al., 2016, *bioRχiv*)