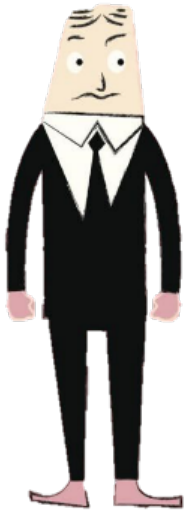
Machine Learning

An Introduction

08/08/2019

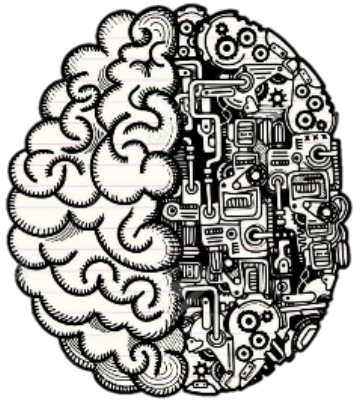
Machine Learning (ML)

Learn from experience

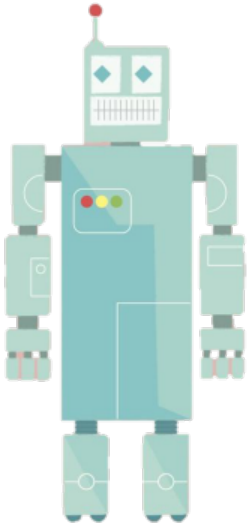


DATA

Learn from ~~experience~~



Follow instructions



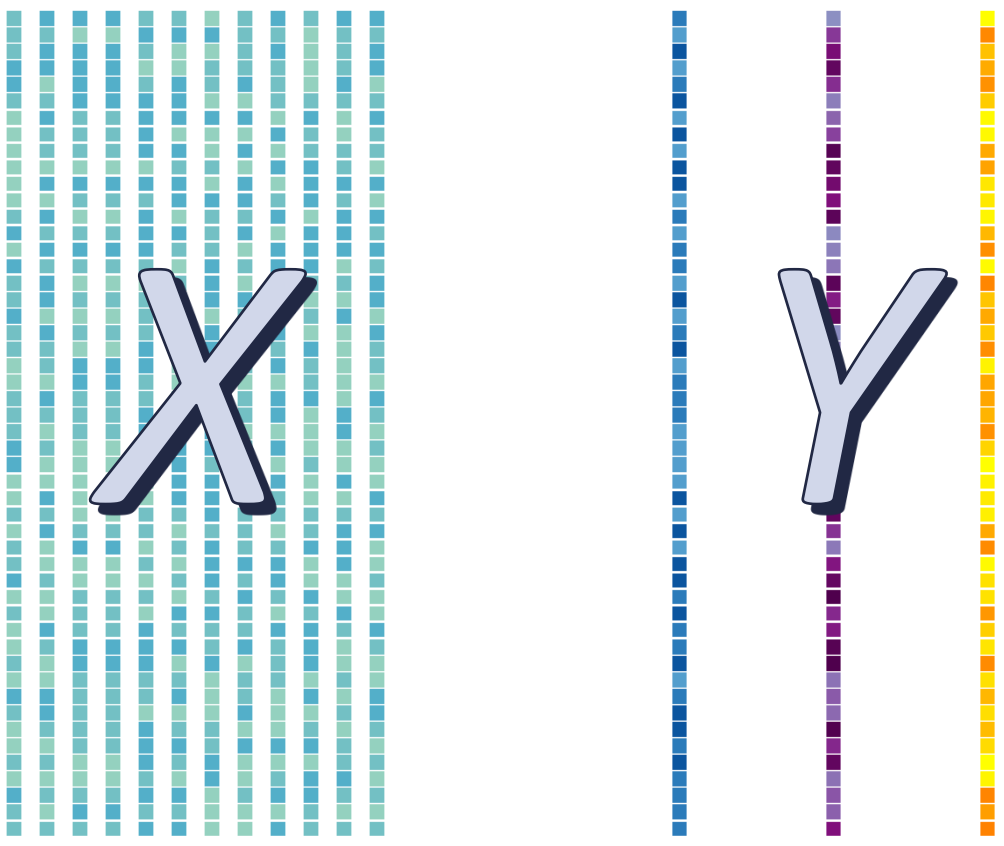
Terminology

features

labels

samples

(observations)



ML algorithms

automatically find patterns in data - **prediction**

Supervised Learning

x, y

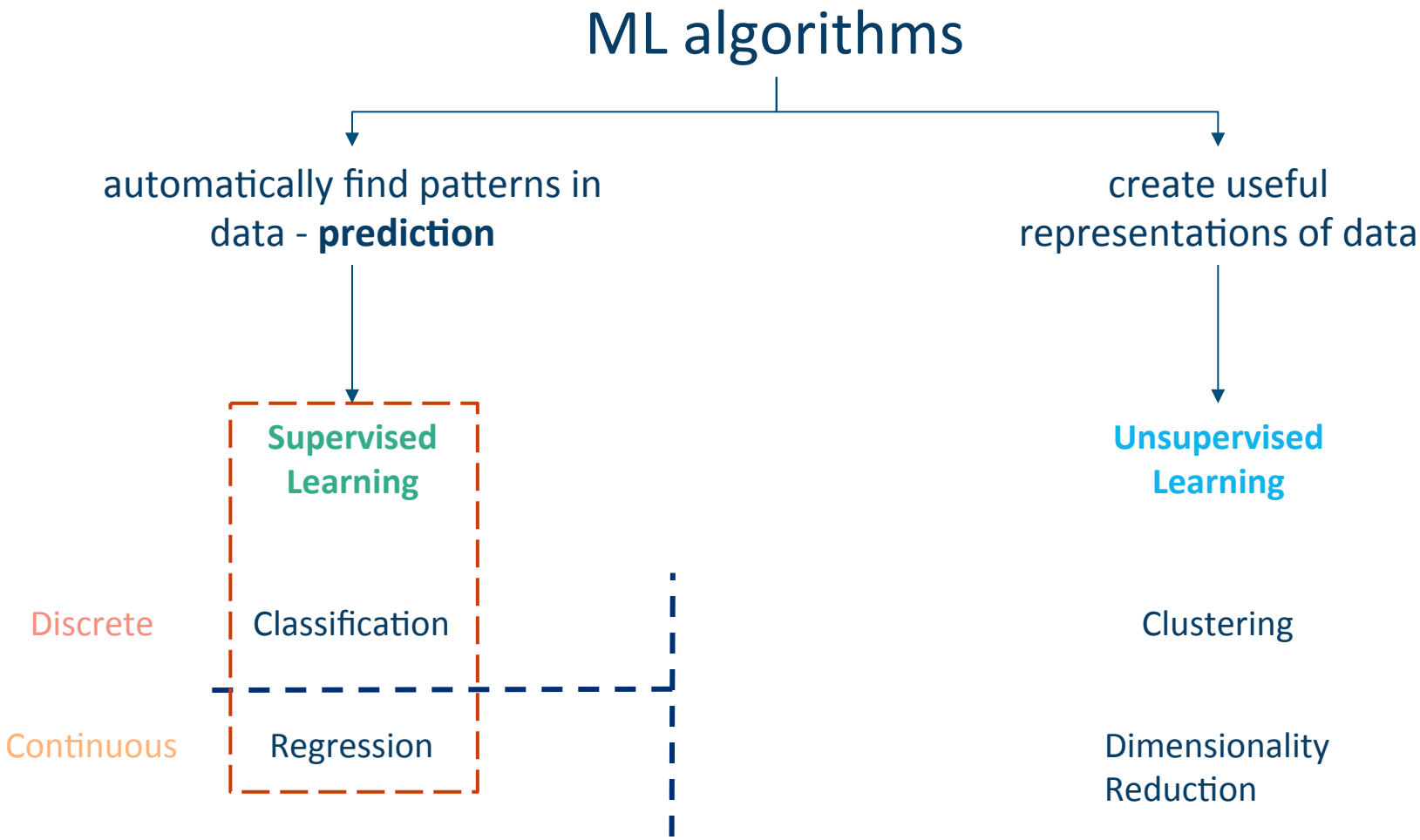
$f: x \rightarrow y$

$y = f(x)$

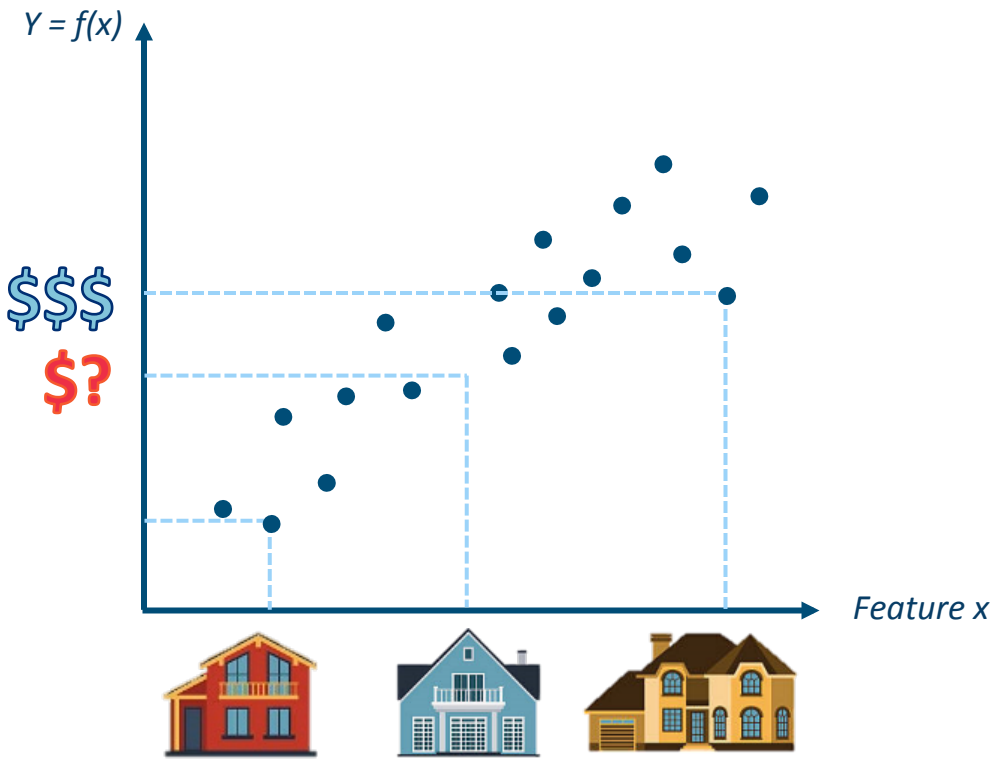
create useful representations of data

Unsupervised Learning

x ~~$, y$~~


ML algorithms
automatically find patterns in data - **prediction**
create useful representations of data
Supervised Learning
Unsupervised Learning
Discrete
Classification
Clustering
Continuous
Regression
Dimensionality Reduction

Regression

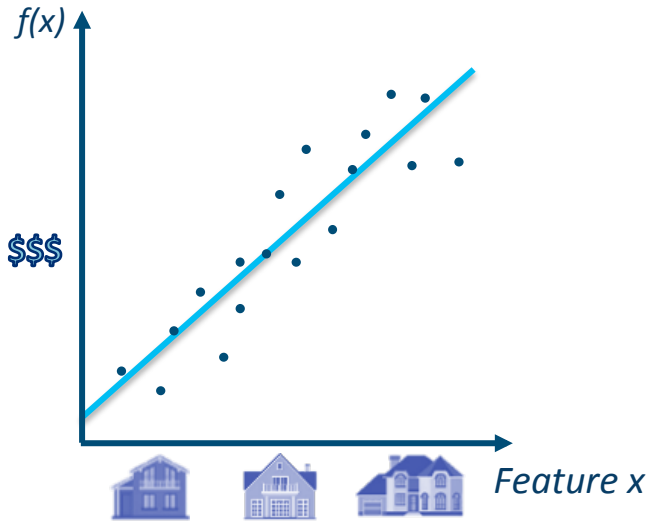


Regression

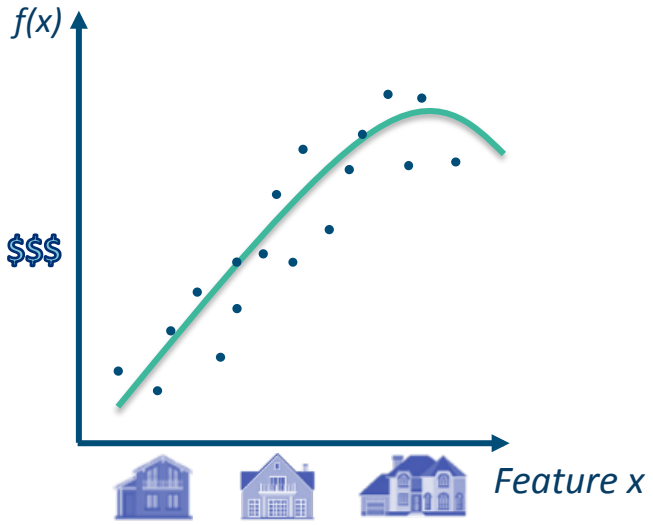
$$f: x \rightarrow y$$

$$y = f(x)$$

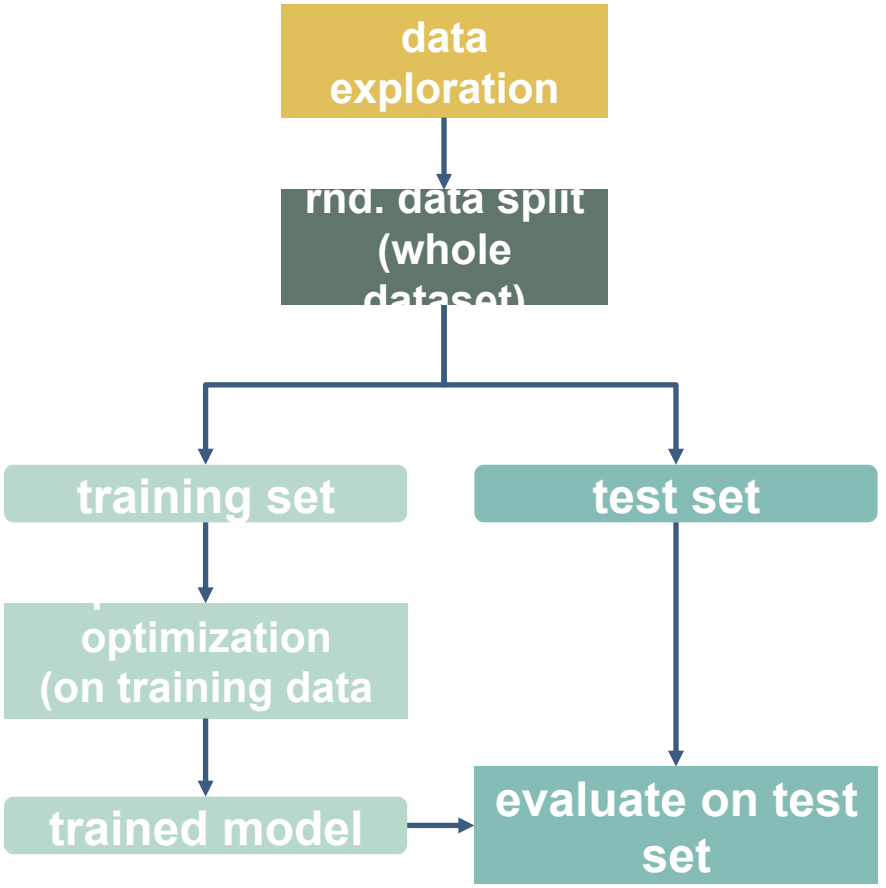
$$y = ax + b$$



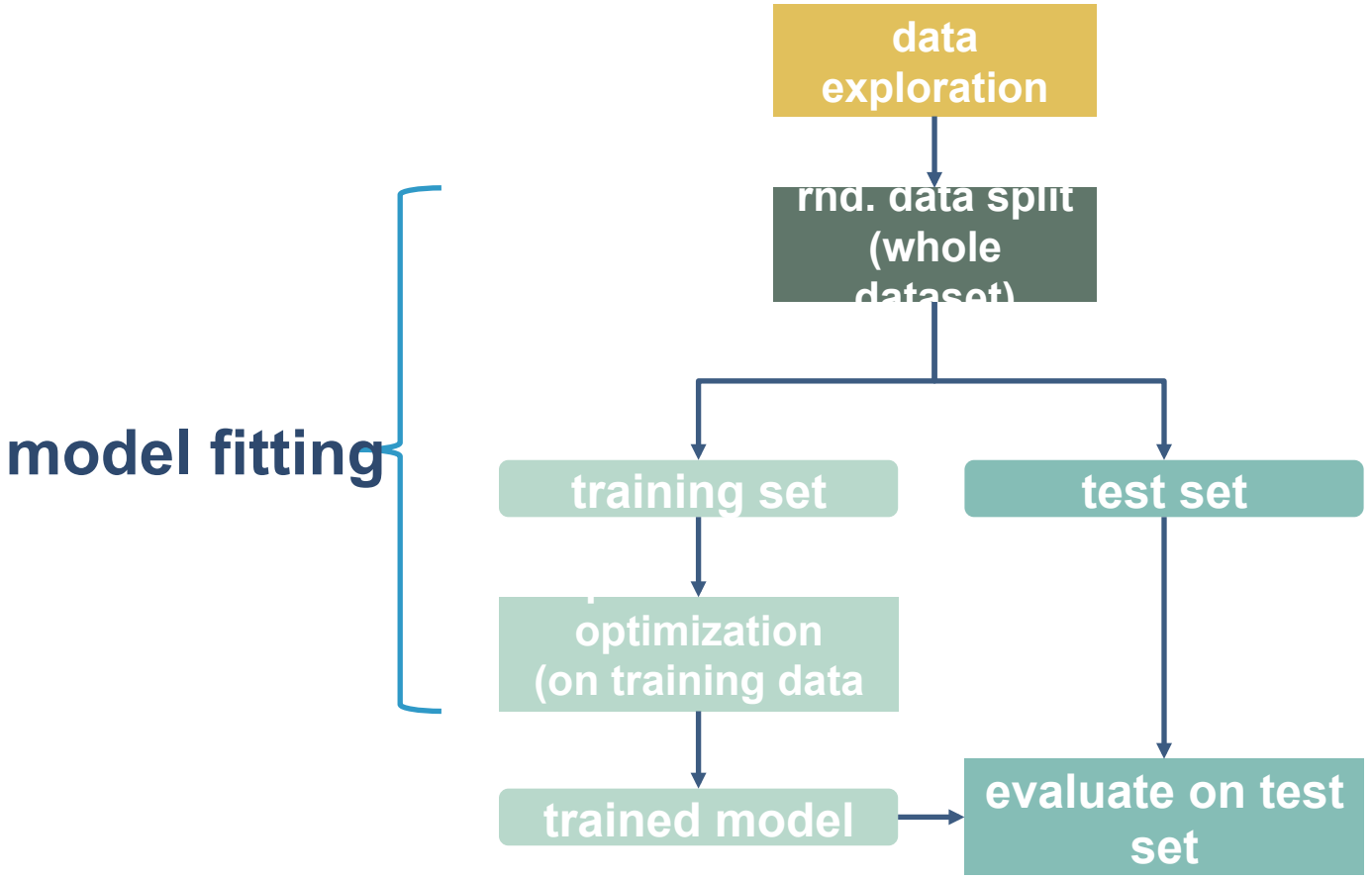
$$y = ax^2 + bx + c$$



Typical ML Pipeline



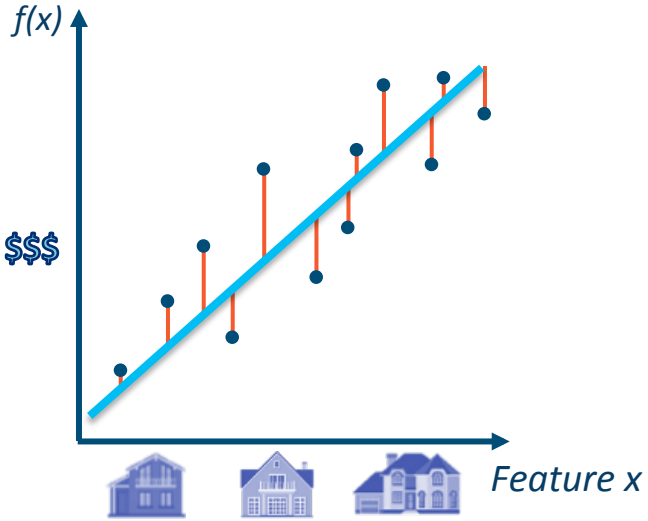
Typical ML Pipeline



Model Fitting

$$f: x \rightarrow y$$

$$y = f(x)$$

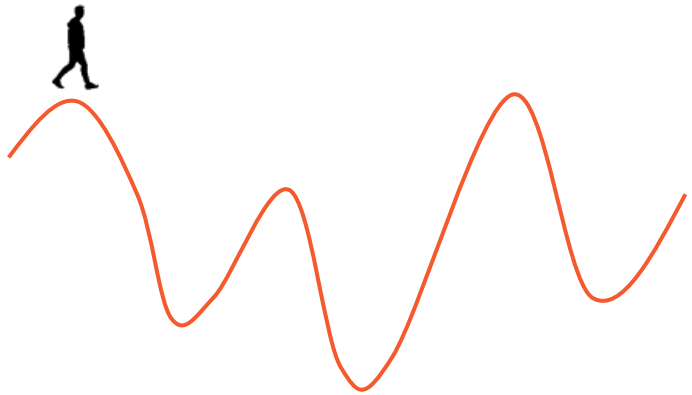


0

1 $f: x \rightarrow y$ → $y = ax + b$

1 $Loss = g(a, b)$ → $MSE = 1/n \sum_{i=1}^{n} (y_i - y$

3 optimization algorithm → gradient descent

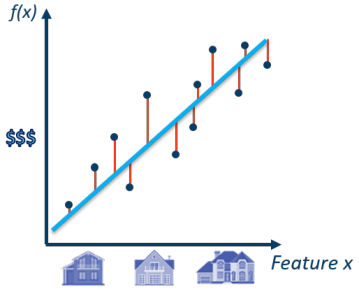


Model Fitting

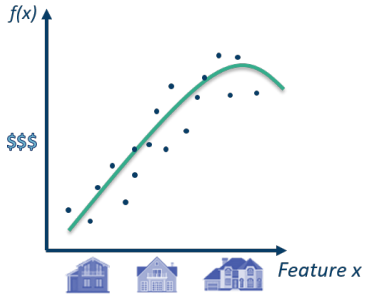
2 Model (*estimator)* selection

linear regression

$y = ax + b$

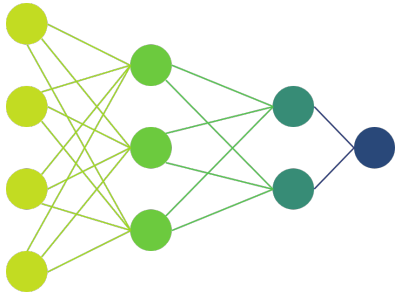


$y = ax^2 + bx + c$



polynomial regression

multi-layer perceptron (ANN)



3 loss *(cost)* function

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

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

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

explained var. $= 1 - \frac{\mathrm{Var}(y - \hat{y})}{\mathrm{Var}(y)}$

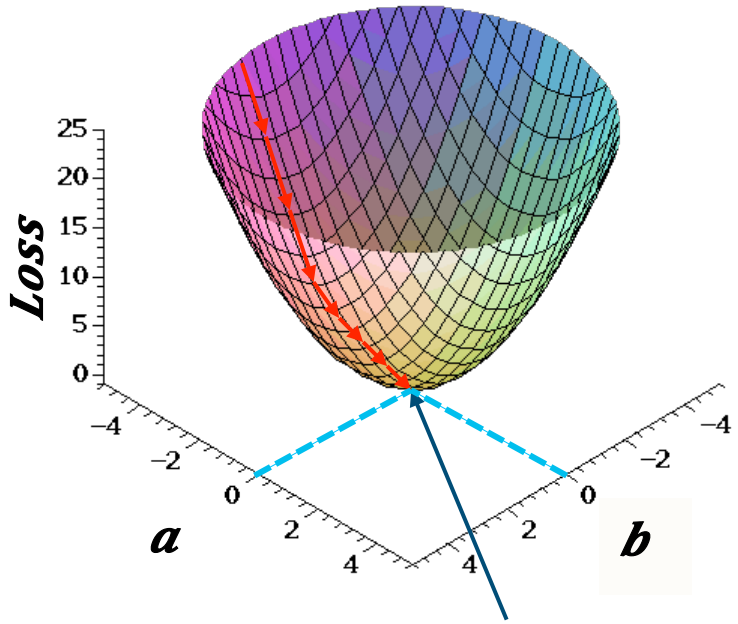
$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})}$

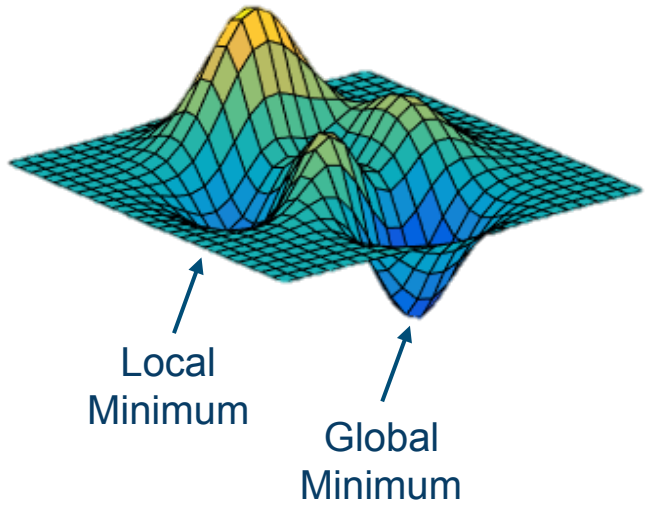
Gradient Descent

4 Optimization of model parameters!

$y = ax + b$

$Loss = f(a,b)$



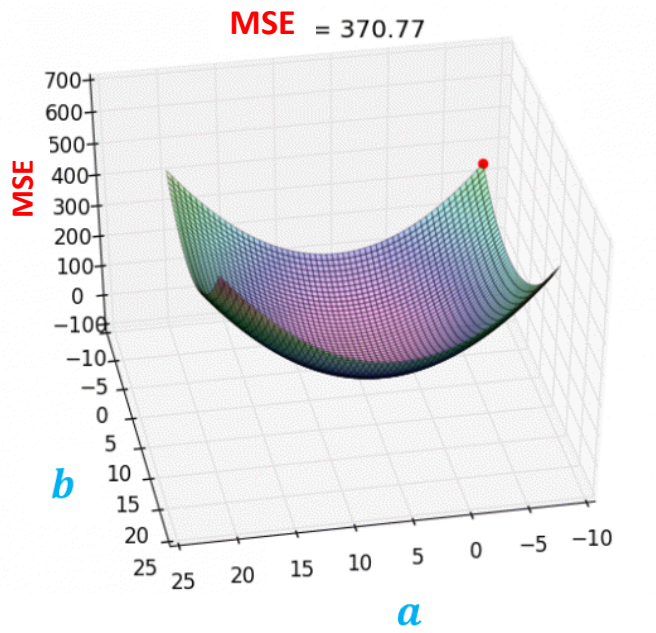


Gradient Descent

Initialize model parameters (a, b) randomly
Iterate between:
1) Compute *loss* (mean square error – MSE)
2) Update model parameters (a, b) in direction of gradient

Training data

$$y = ax + b$$



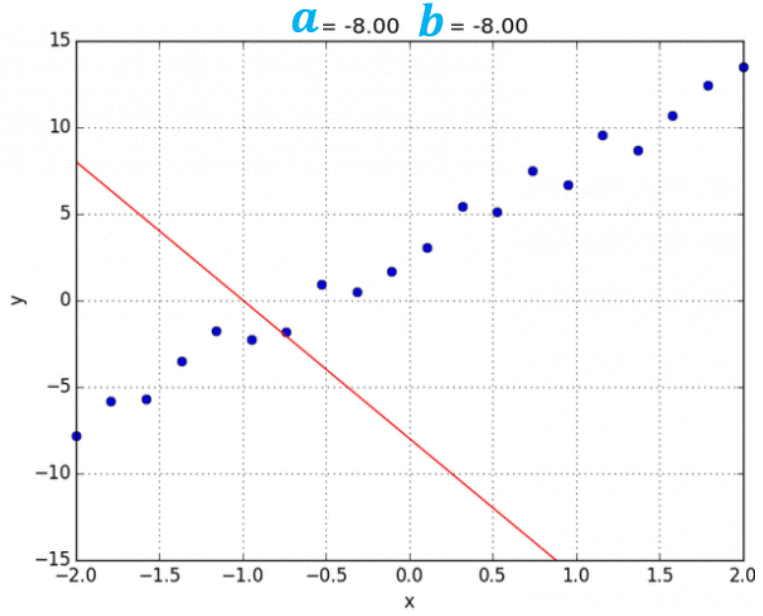
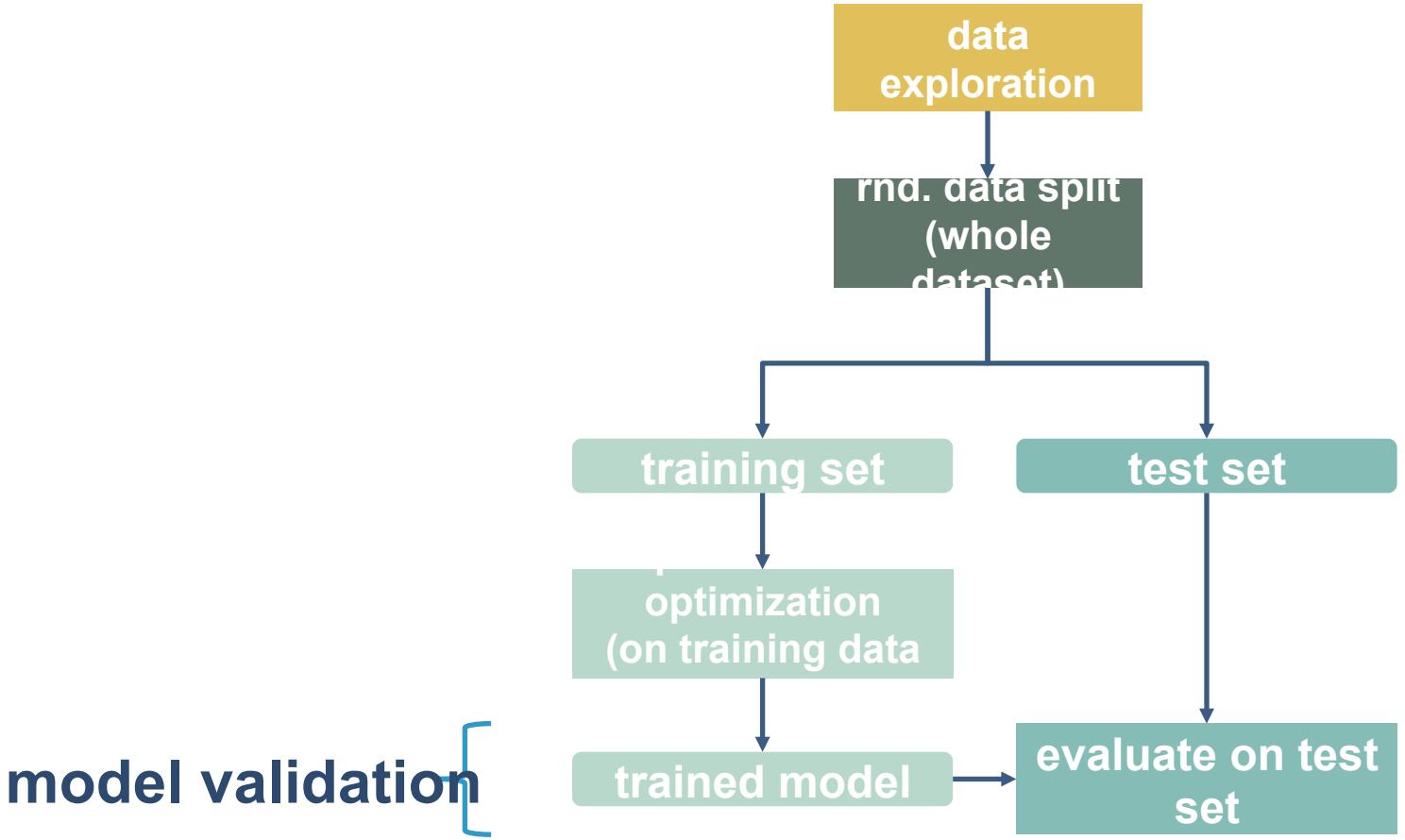
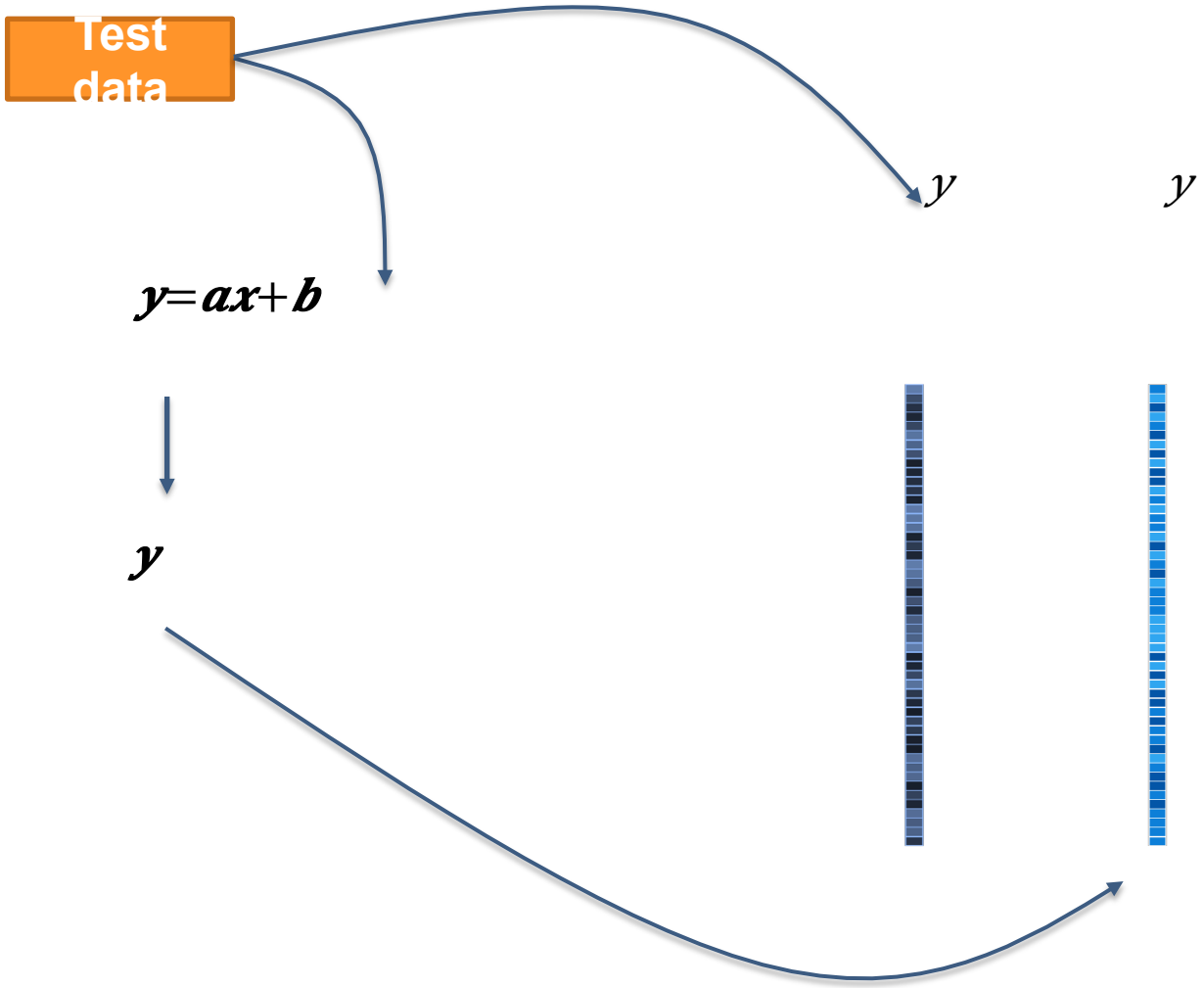


Figure adapted from

Typical ML Pipeline



Model Validation



score function

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

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

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

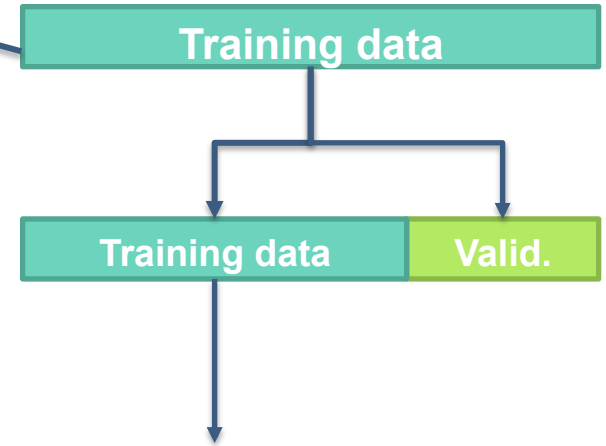
explained var. $= 1 - \frac{\mathrm{Var}(y - \hat{y})}{\mathrm{Var}(y)}$

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

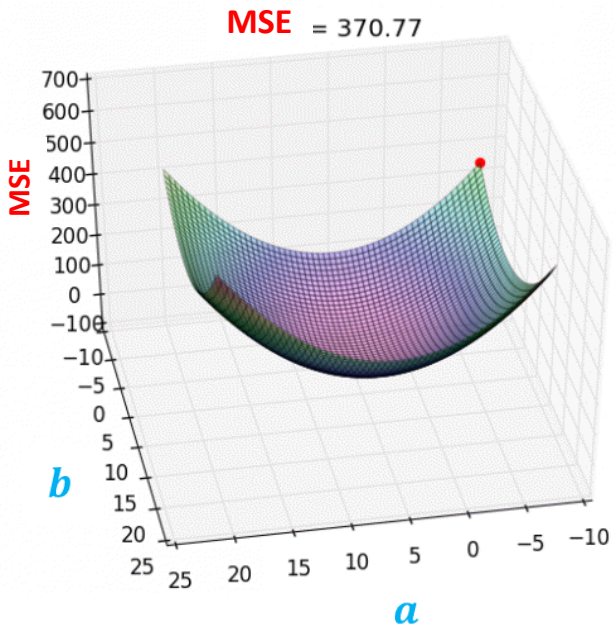
Houston … we have a problem!

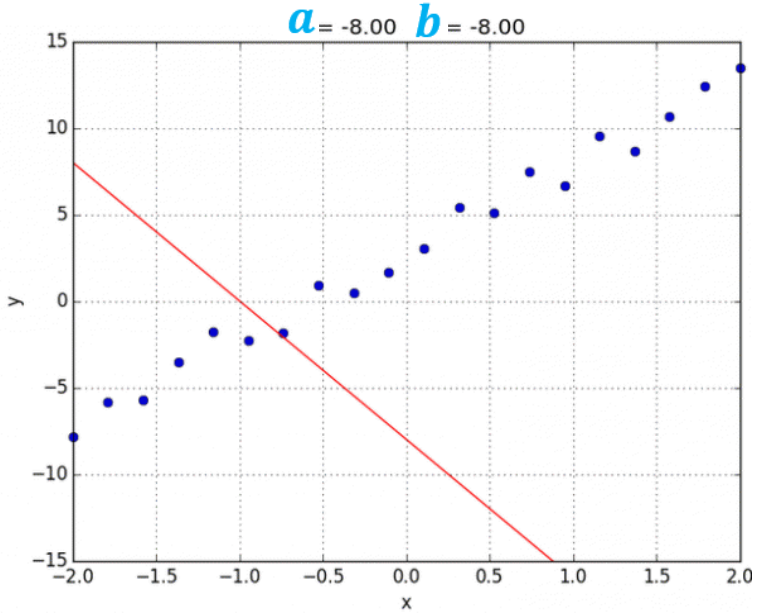
Initialize model parameters (a, b) randomly
Iterate between:
1) Compute *loss* (mean square error – MSE)
2) Update model parameters (a, b) in direction of gradient

$$y = ax + b$$



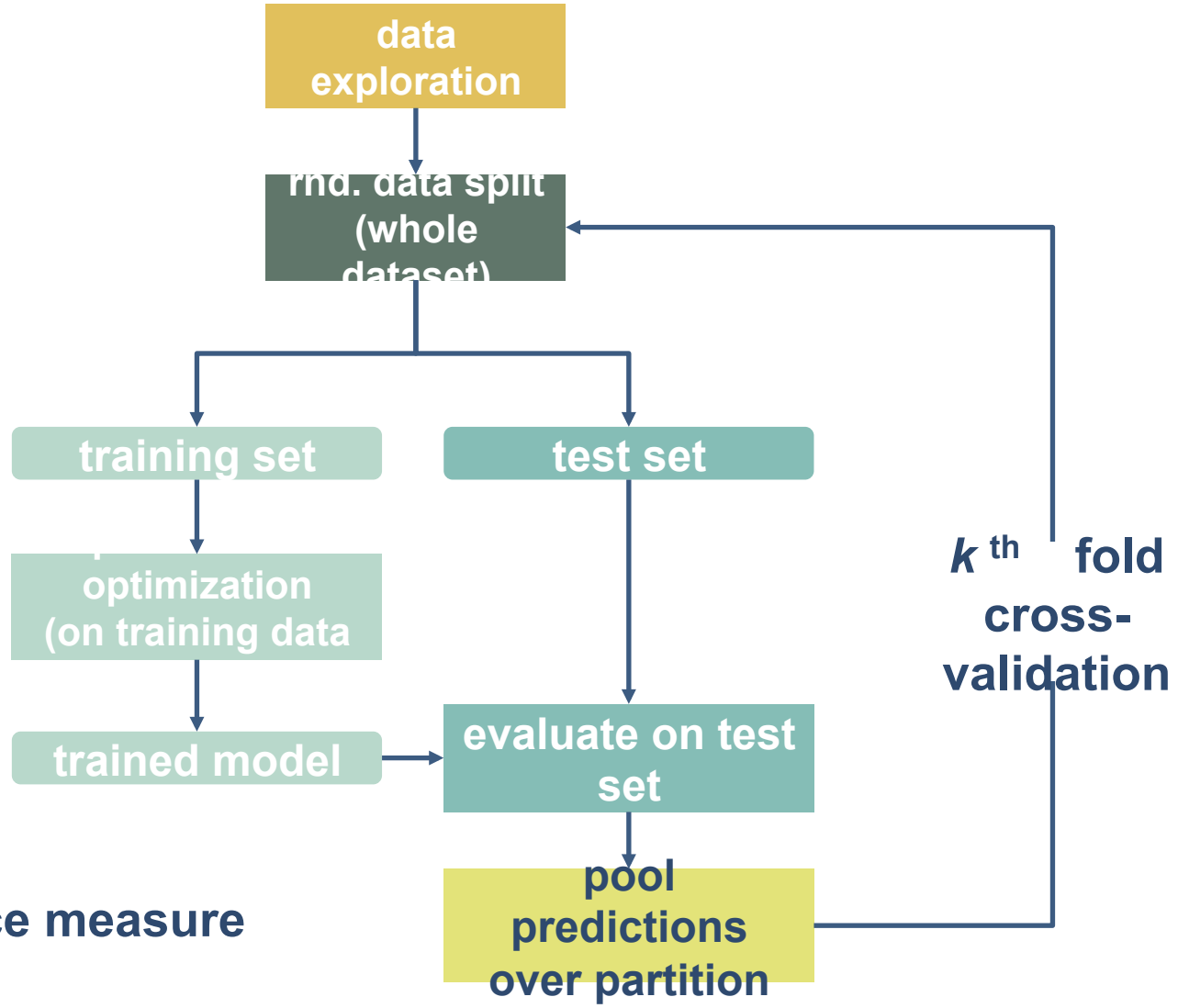
multiple training epochs!





How many TE?

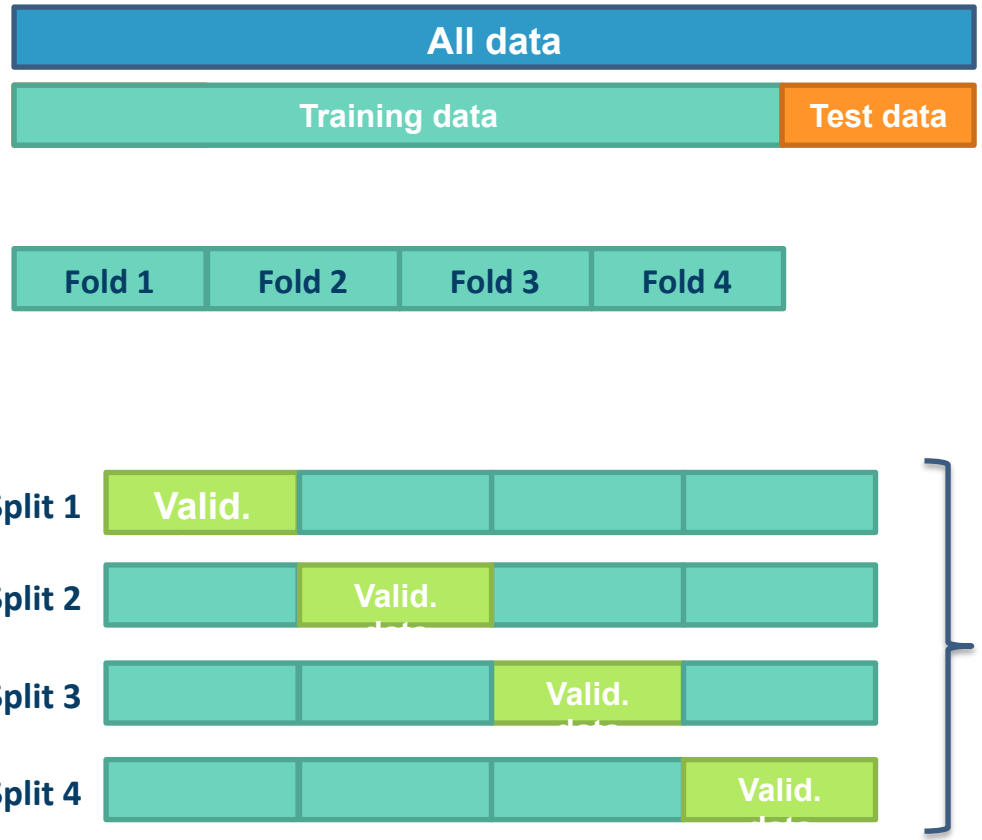
Typical ML Pipeline



WHY CV?

- **Avoid overfitting**
- **Improve model generalizability**
- **Provide a more accurate performance measure**
- ***Hyper*-parameter tuning**

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



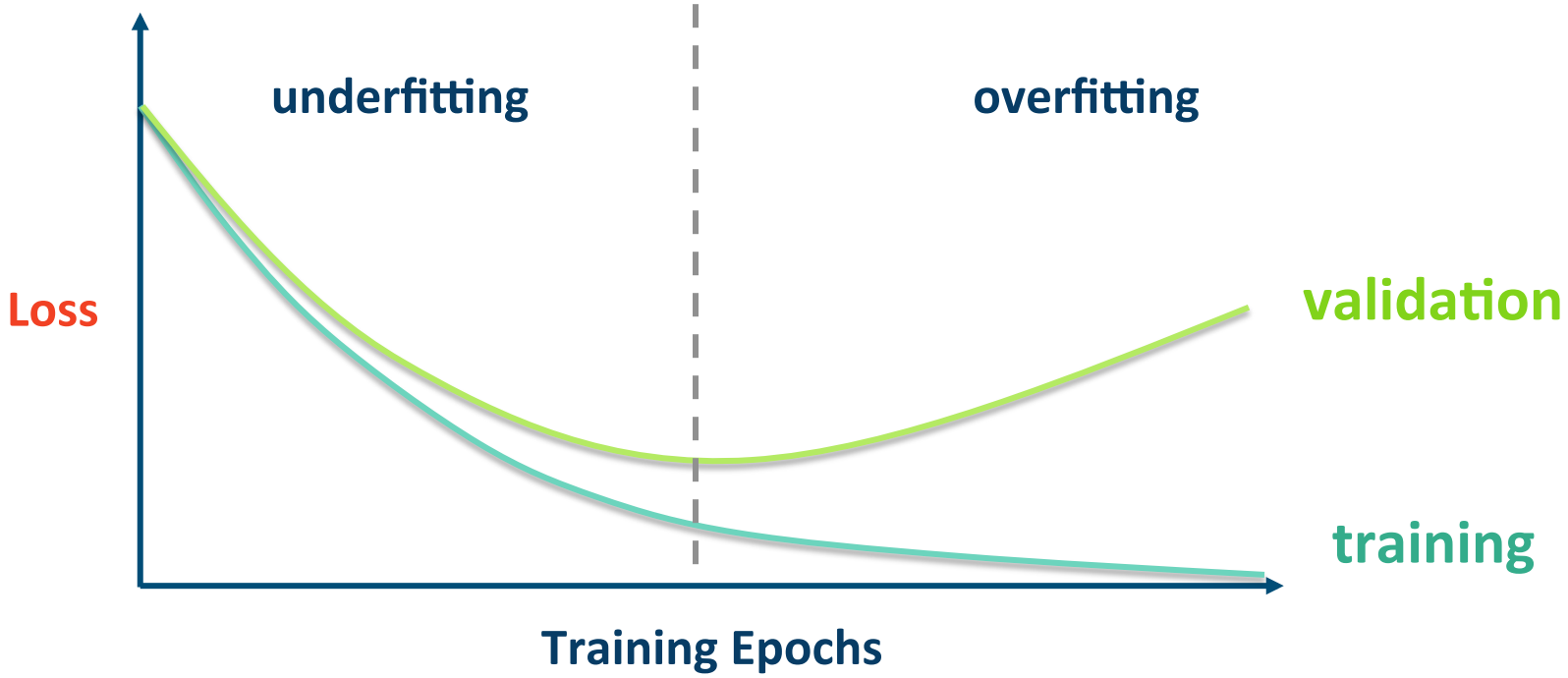
***Hyper*-parameter tuning!**

1) Train model parameters on **training** set
2) Evaluate training with the **validation** set
3) Report error on **test** set

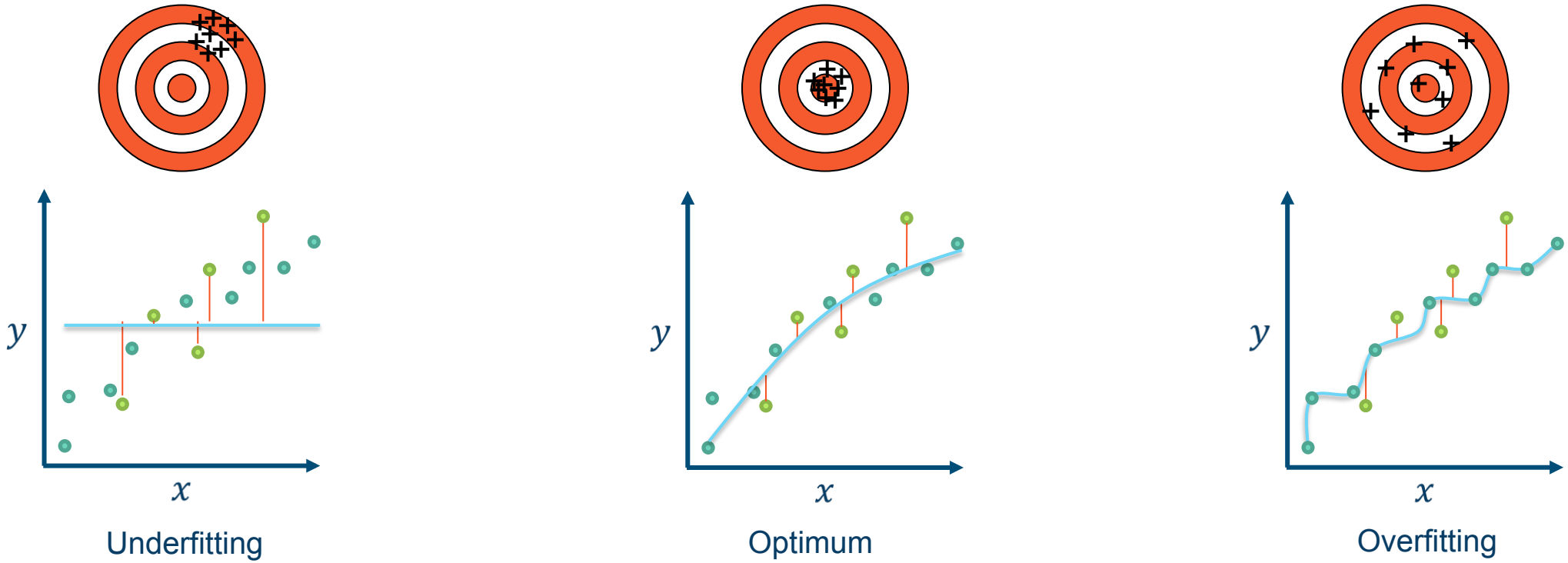
Overfitting

k^{th}-fold

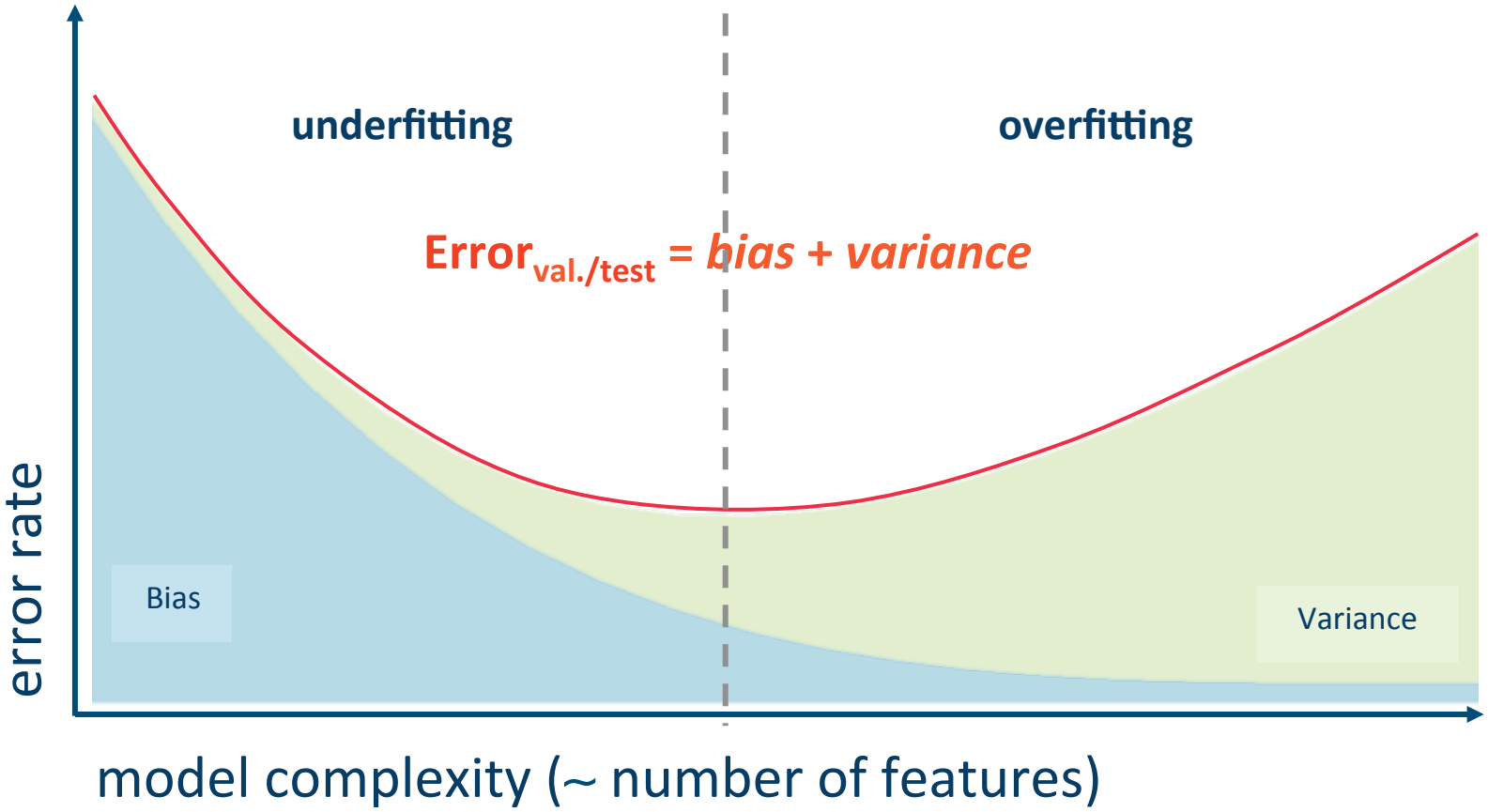
How many TE?



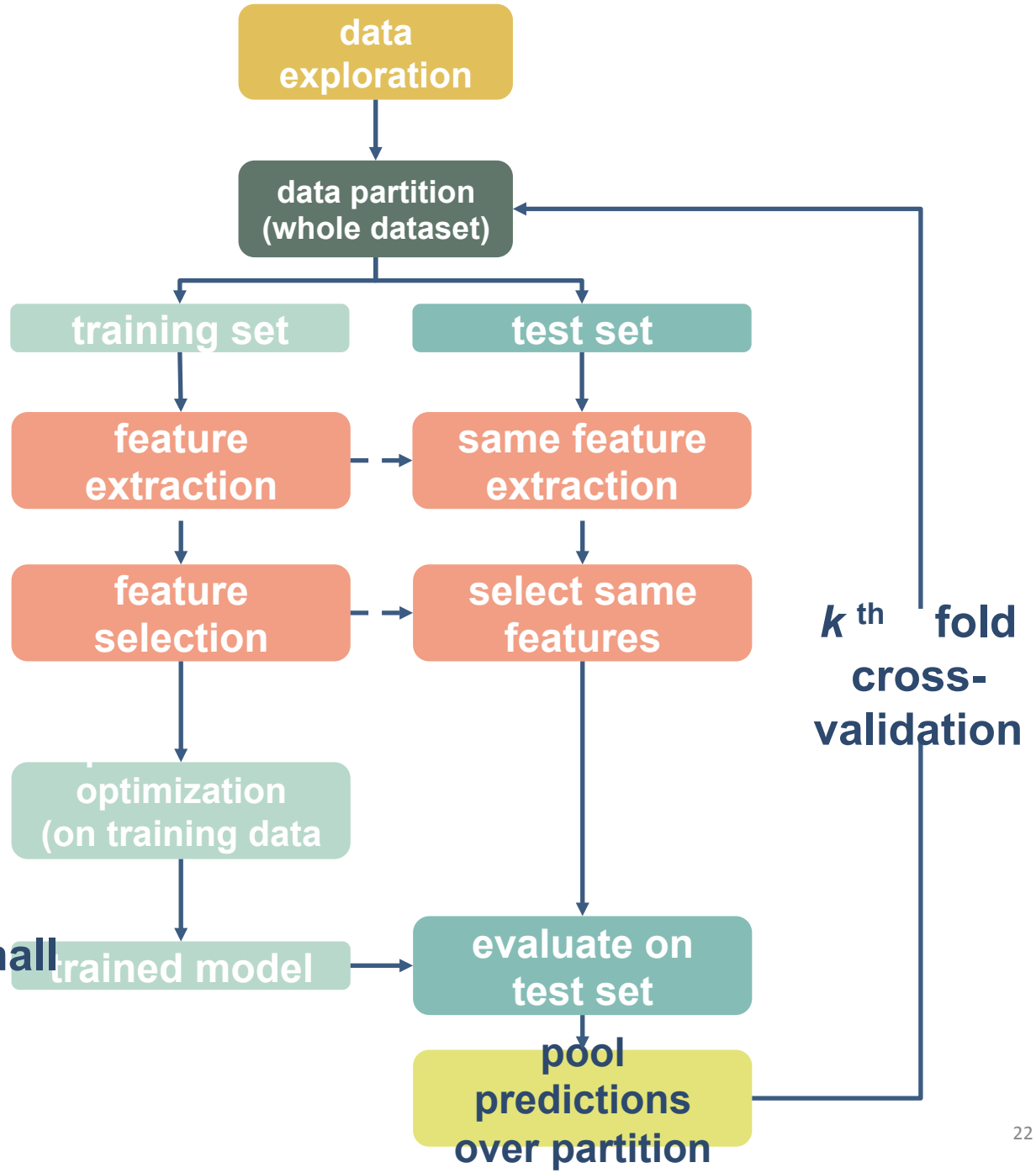
Bias-Variance Trade-Off



Bias-Variance Trade-Off



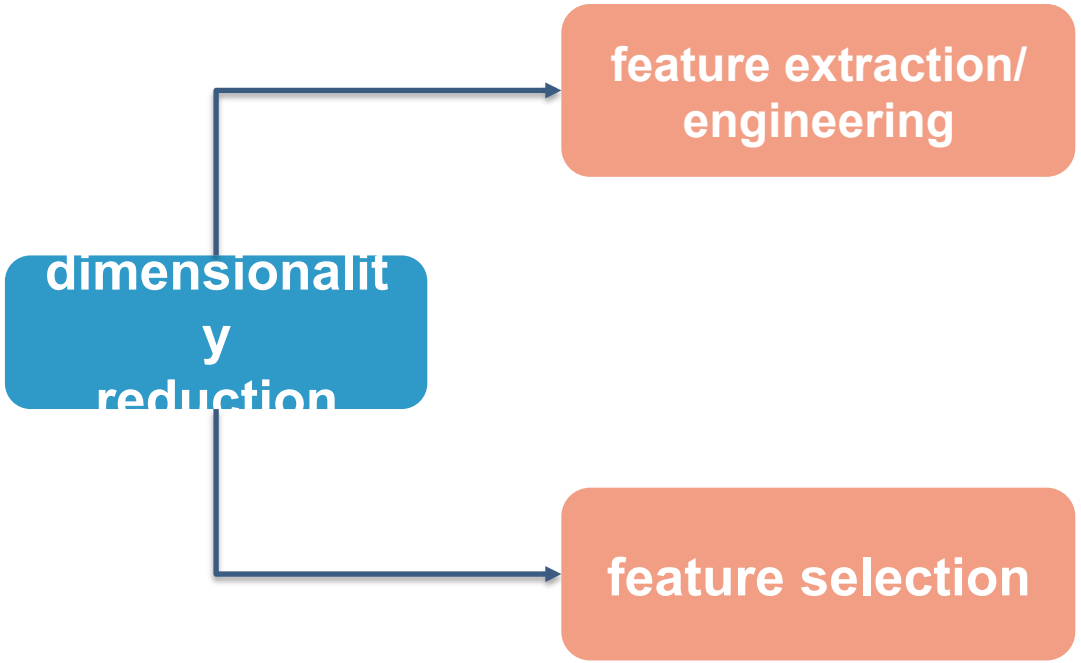
Typical ML Pipeline



WHY DR ?

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

Tip 1: Dimensionality Reduction



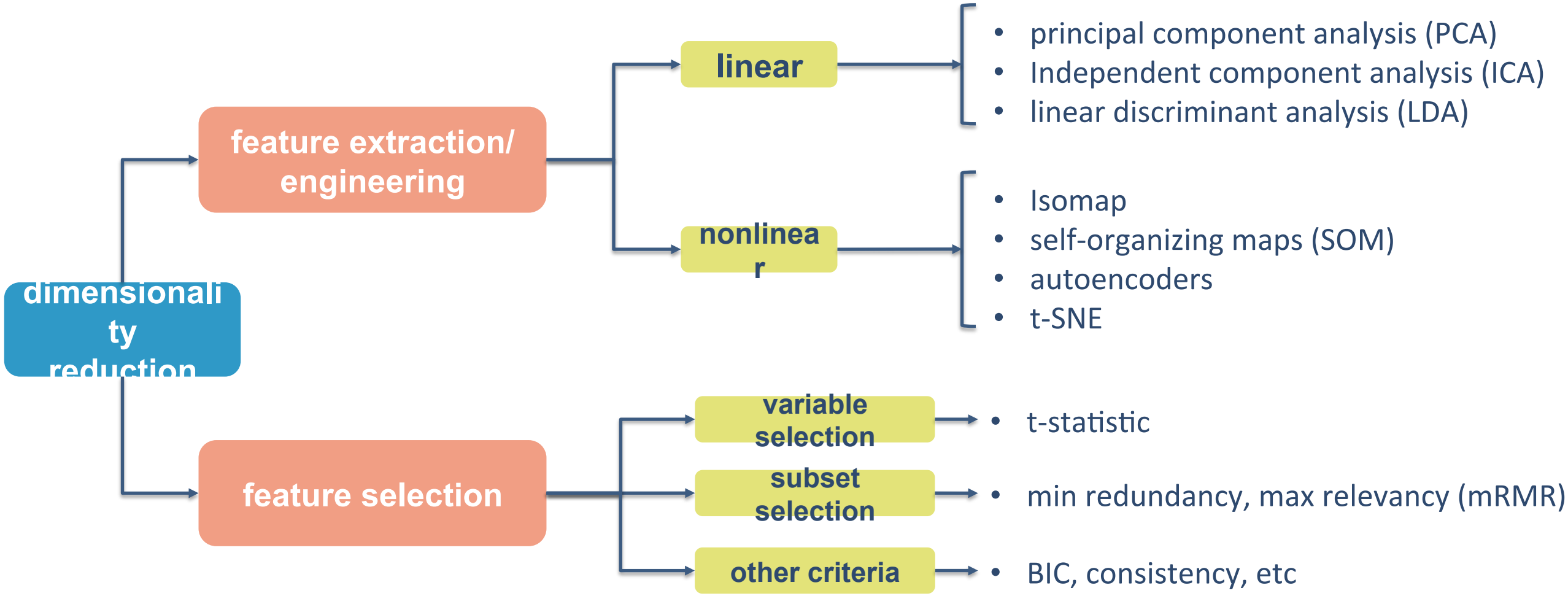
- Compact representation of the data
- Maps input features into a lower dimensional space

e.g. If features are $X=[x\downarrow 1 , x\downarrow 2 , x\downarrow 3 , x\downarrow 4]$
then $Z=T(X)=[c\downarrow 1\ x\downarrow 1 + c\downarrow 2\ x\downarrow 2 , \quad x\downarrow 3 * x\downarrow 4]$

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

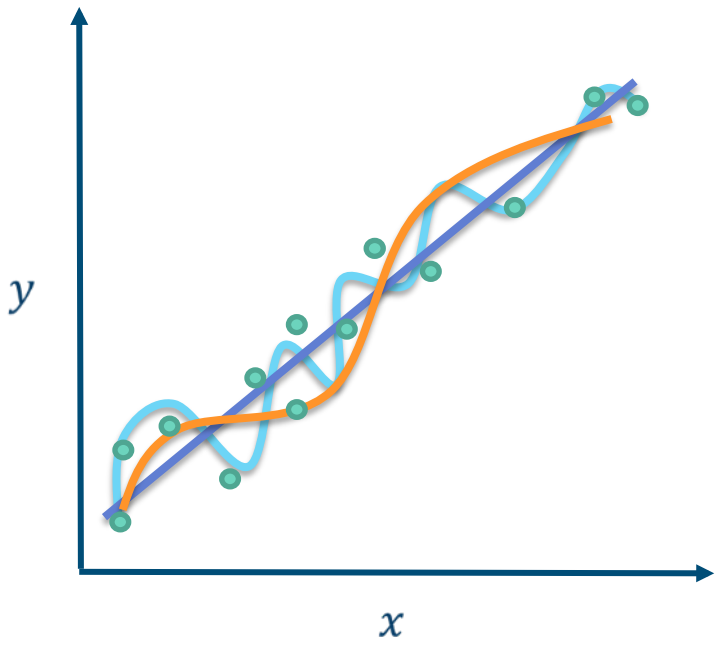
e.g. $Z=S(X)=[x\downarrow 2 , \ x\downarrow 3]$

Tip 1: Dimensionality Reduction



Tip 2: Regularization

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



Penalties on the ***LOSS*** function to prevent overfitting!

1) L1/Lasso: constrains parameters to be **sparse**

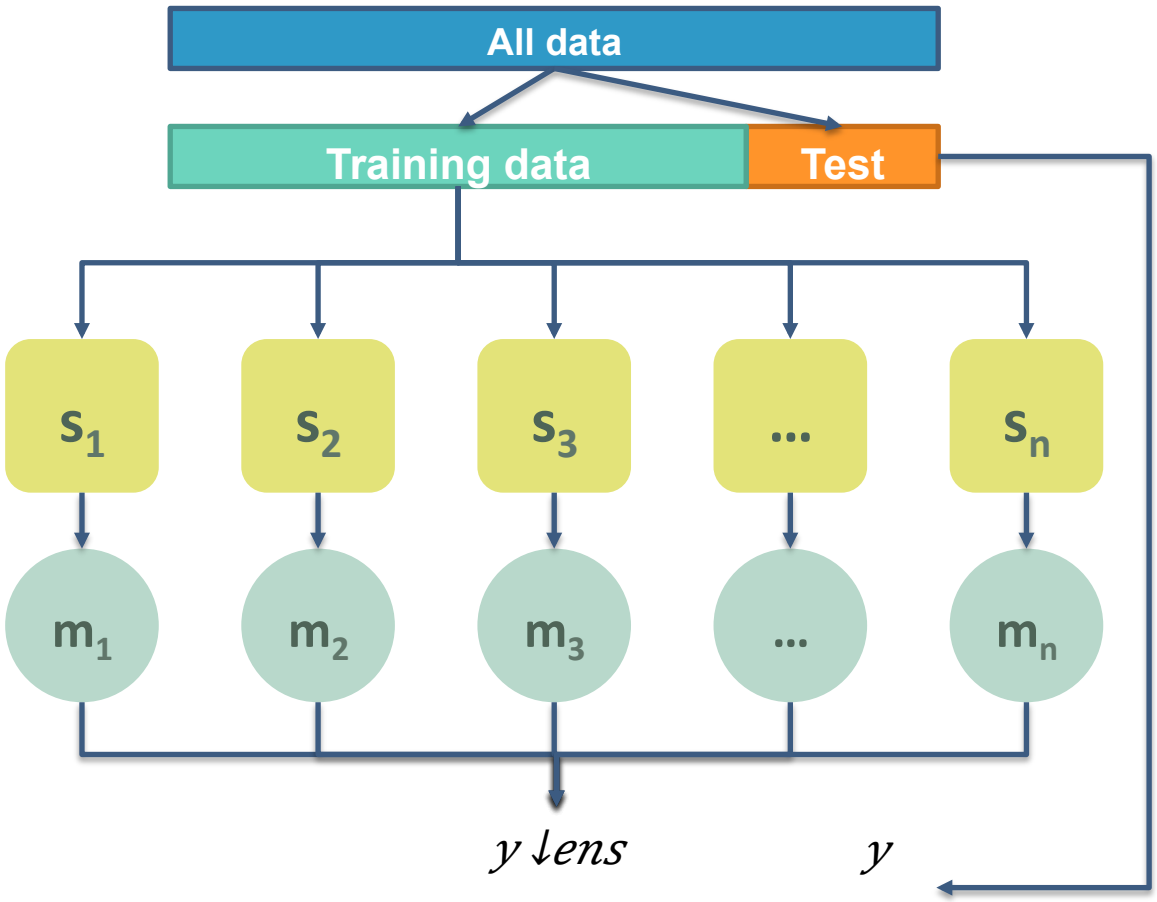
$$MSE = \sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

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

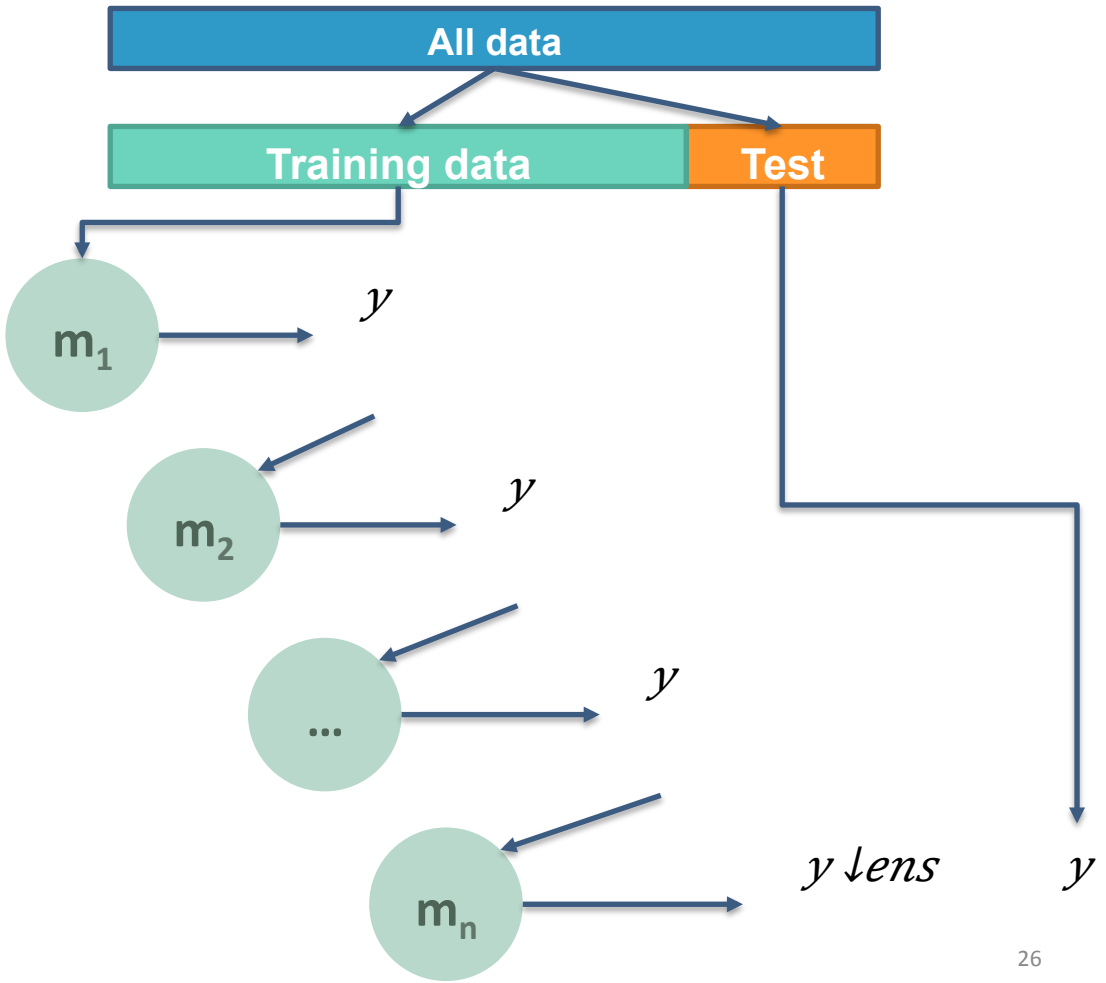
$$MSE = \sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} \beta_j{}^2$$

Tip 3: Ensemble Methods

bagging = bootstrapp resampling + aggregation



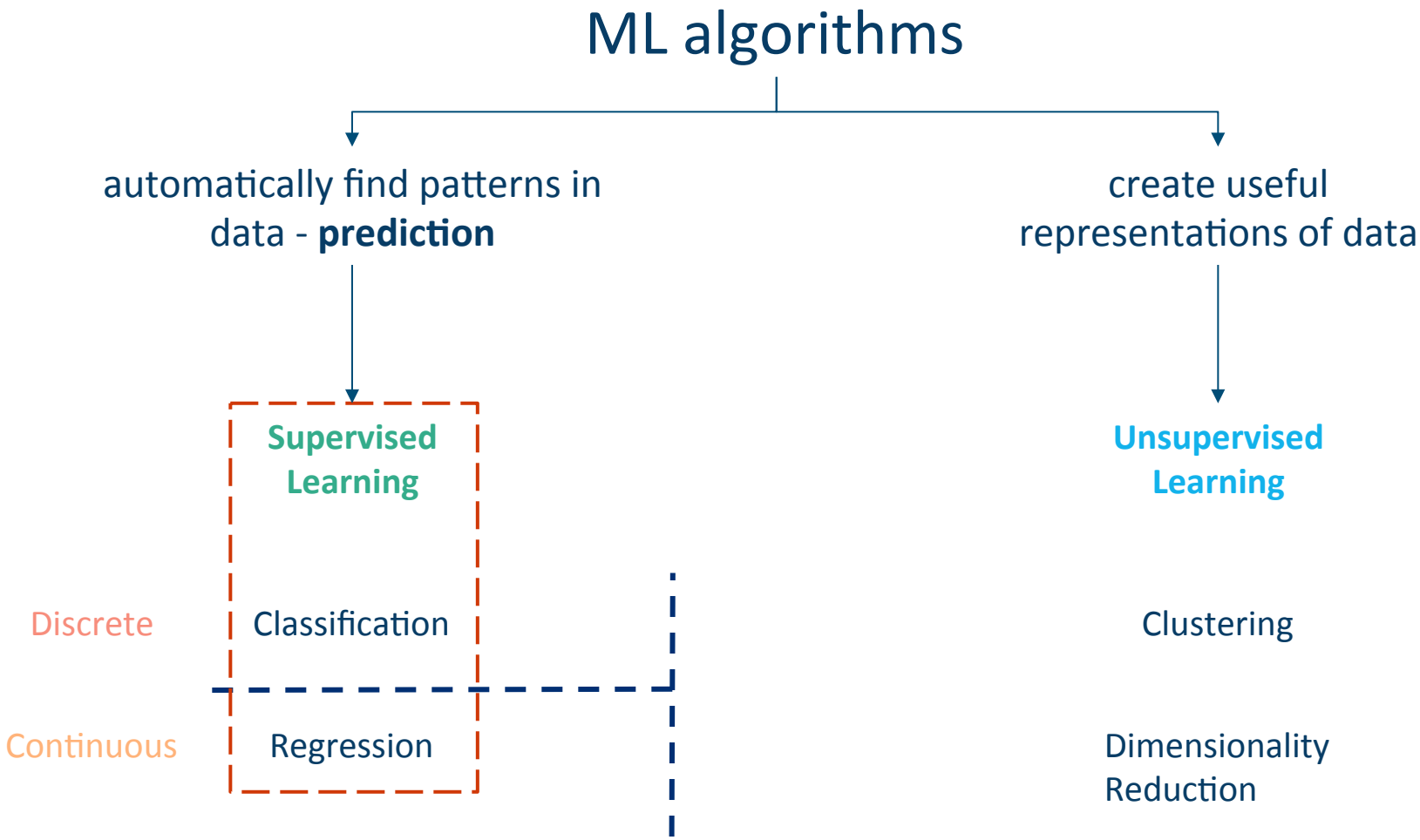
boosting



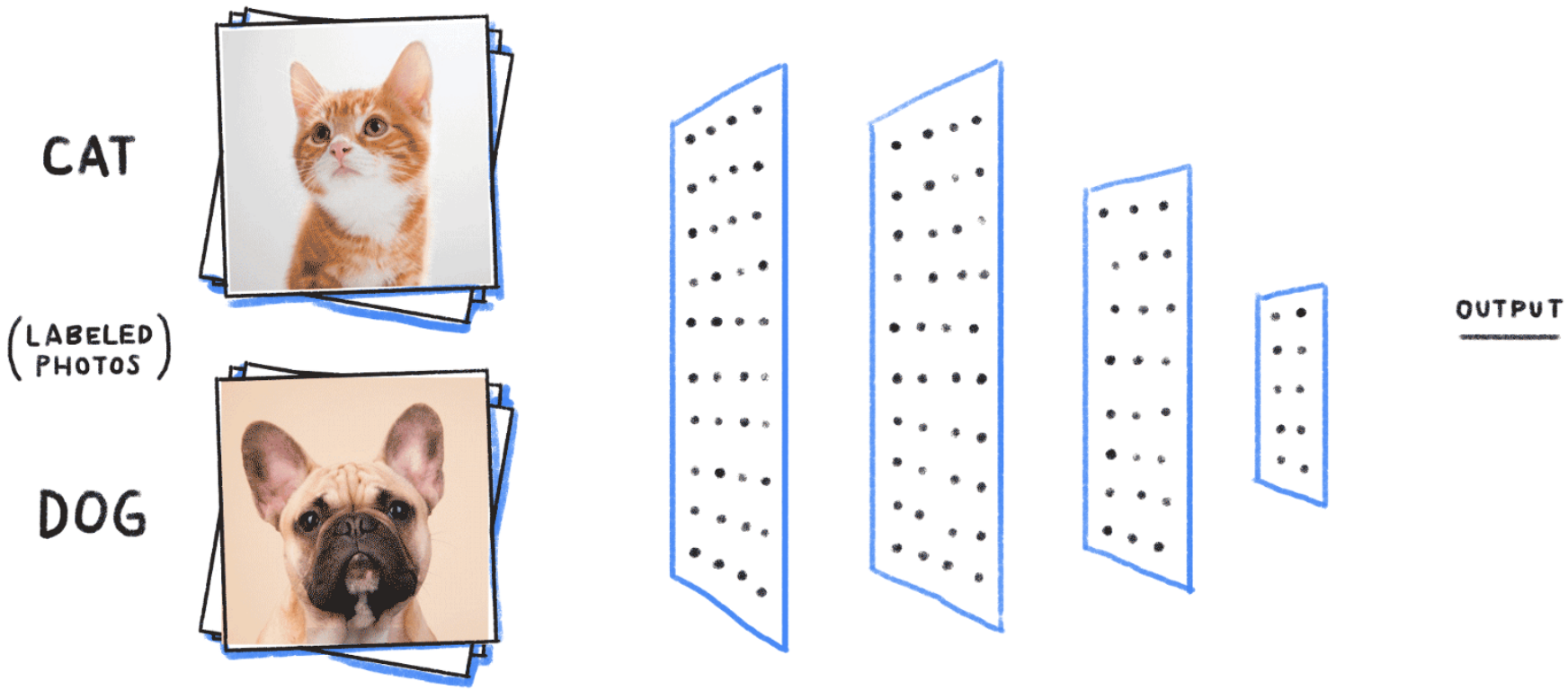
Sources of Bias

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

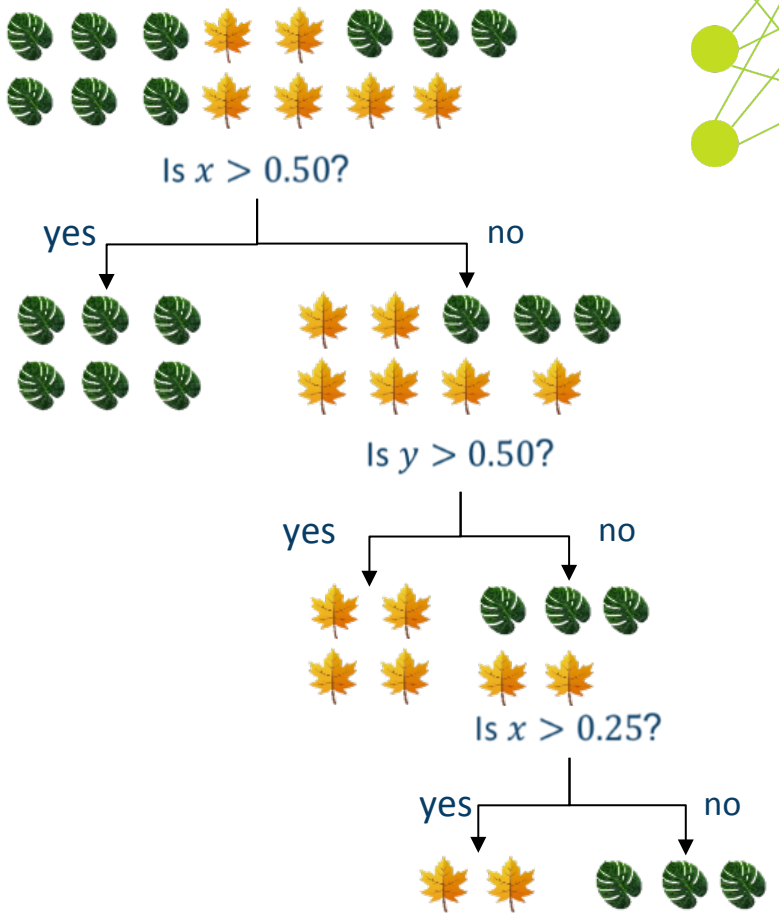
Pradeep Reddy Raamana (slide 63 - https://drive.google.com/file/d/1jXWQT9P4CmJgBShjwqymKiI_jOH-bb7q/view)

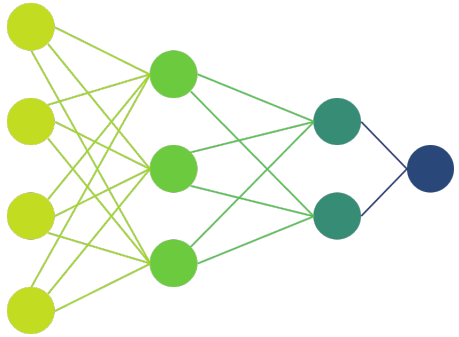

ML algorithms
automatically find patterns in data - **prediction**
create useful representations of data
Supervised Learning
Unsupervised Learning
Discrete
Classification
Clustering
Continuous
Regression
Dimensionality Reduction

Classification



Classification

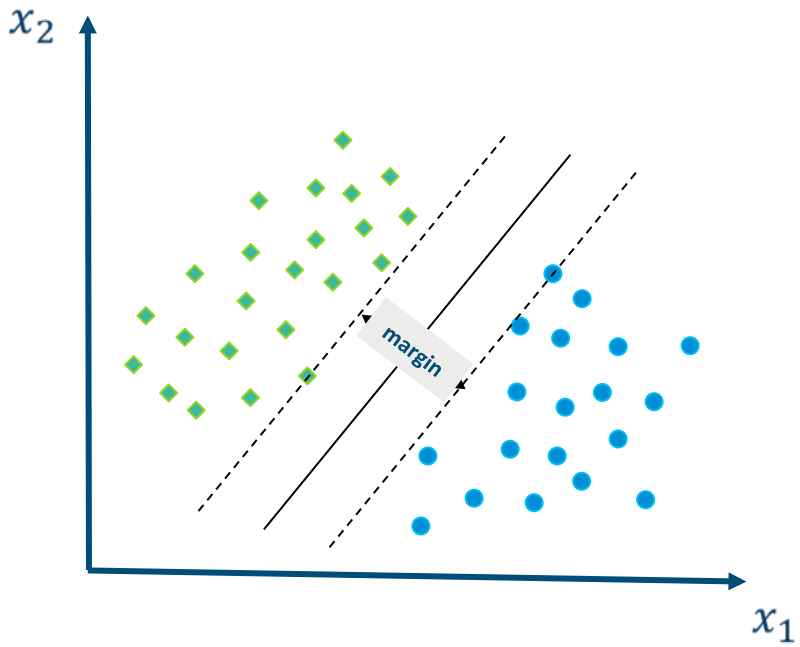




- Support Vector Machine (SVM)

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

} probabilistic classifiers



Performance Metrics – Binary Classification

confusion matrix

		True condition: POSITIVE	True condition: NEGATIVE
Predicted condition	POSITIVE	TRUE positive - $T\downarrow p$	FALSE positive - $F\downarrow p$ (Type I error)
	NEGATIVE	FALSE negative - $F\downarrow n$ (Type II error)	TRUE negative - $T\downarrow n$

score function

$$accuracy = T\downarrow p + T\downarrow n \,/ T\downarrow p + T\downarrow n + F\downarrow p + F\downarrow n$$

$$precision = T\downarrow p \,/ T\downarrow p + F\downarrow p$$

$$recall = T\downarrow p \,/ T\downarrow p + F\downarrow n$$

Appropriate when classes are imbalanced!

$$F1\ score = 2T\downarrow p \,/ 2T\downarrow p + F\downarrow p + F\downarrow n$$

Performance Metrics – Binary Classification

		True condition: POSITIVE	True condition: 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)

$$TPR = T_p / P = T_p / (T_p + F_n)$$

$$FPR = F_p / N = F_p / (T_n + F_p)$$

Specificity (or selectivity)

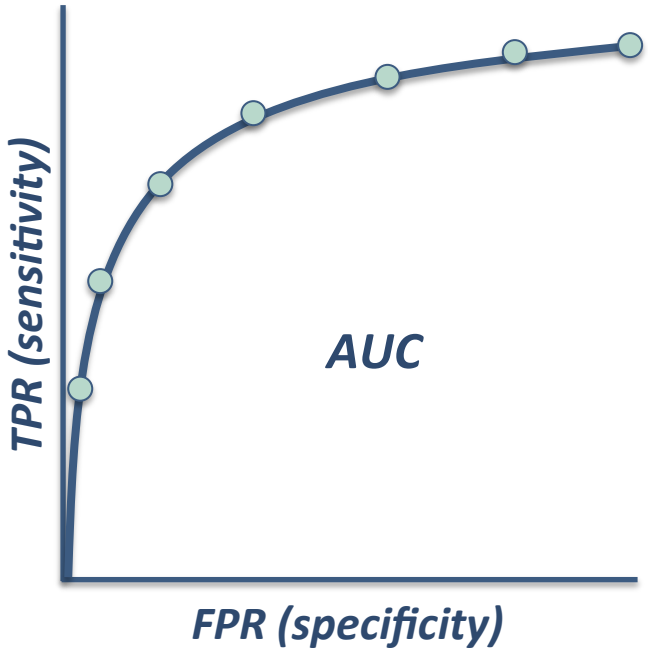
$$FNR = F_n / P = F_n / (T_p + F_n)$$

$$TNR = T_n / N = T_n / (T_n + F_p)$$

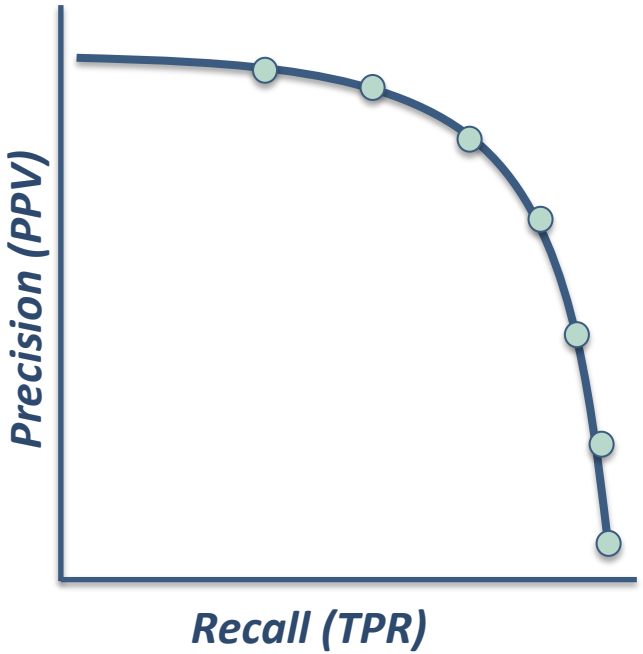
Performance Metrics – Binary Classification

(probabilistic classifiers)

ROC curve



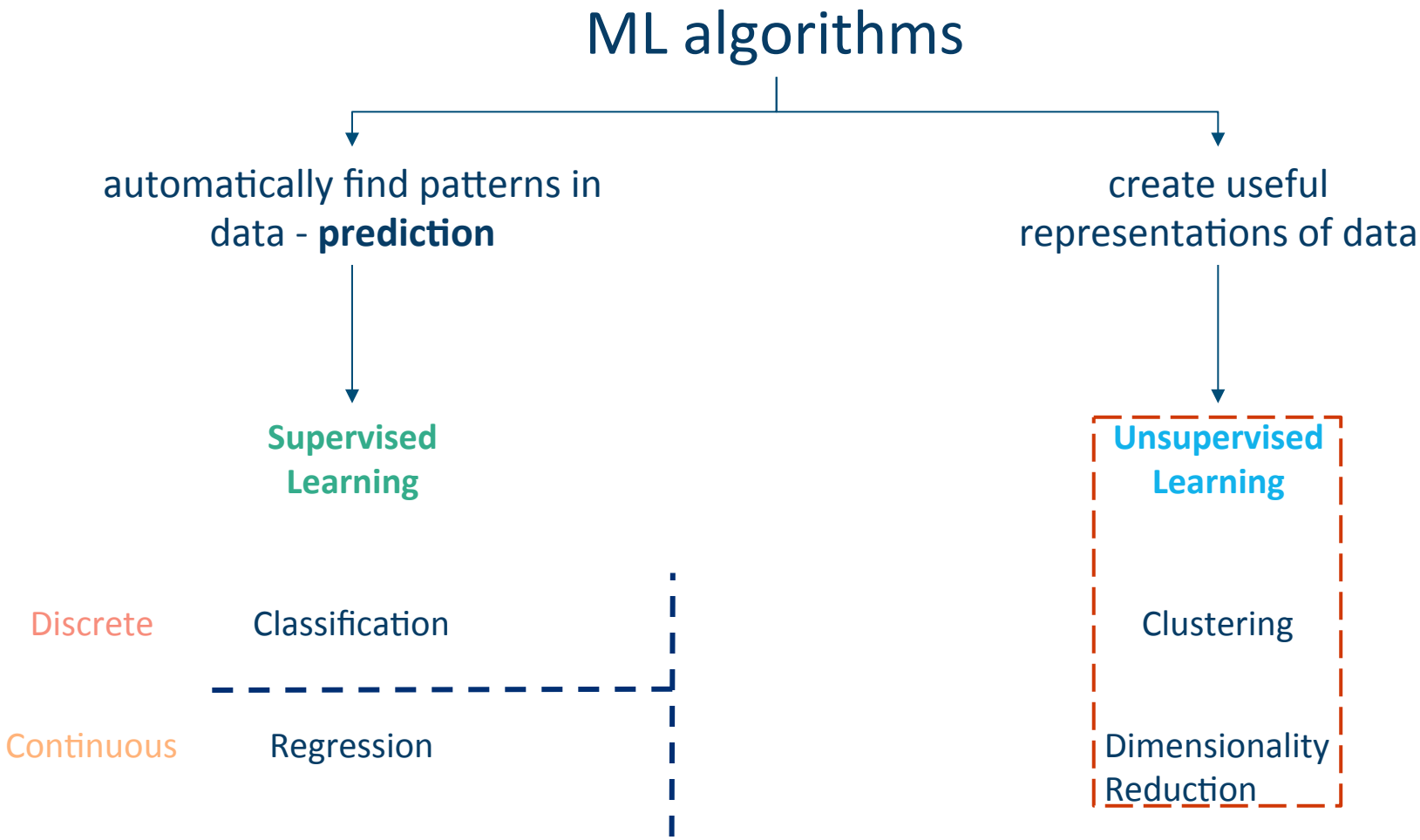
precision-recall curve



Multiclass Prediction (≠ Multilabel Prediction)

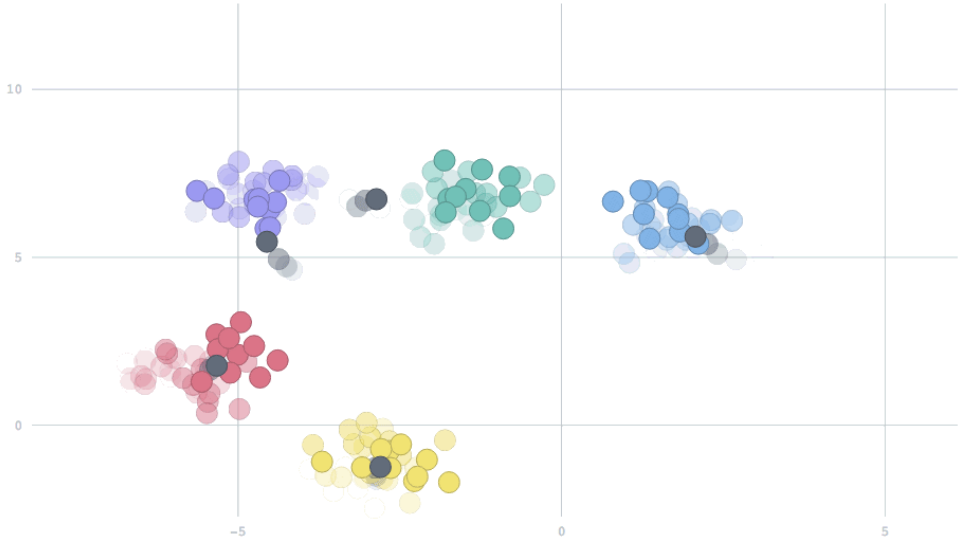
		True class		
		CAT	DOG	BIRD
Predicted class	CAT	13	0	0
	DOG	0	10	6
	BIRD	0	0	9

- 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.


ML algorithms
automatically find patterns in data - **prediction**
create useful representations of data
Supervised Learning
Unsupervised Learning
Discrete
Classification
Continuous
Regression
Clustering
Dimensionality Reduction

Clustering

- *K*-means
- Hierarchical clustering
- Mixture of Gaussians



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

statistical inference

Goal:

- Identify significant contributing variables (statistical null-hypothesis testing, *p*-values)

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.

pattern recognition

Goal:

- Identify most predictive variable sets (out-of-sample prediction performance)

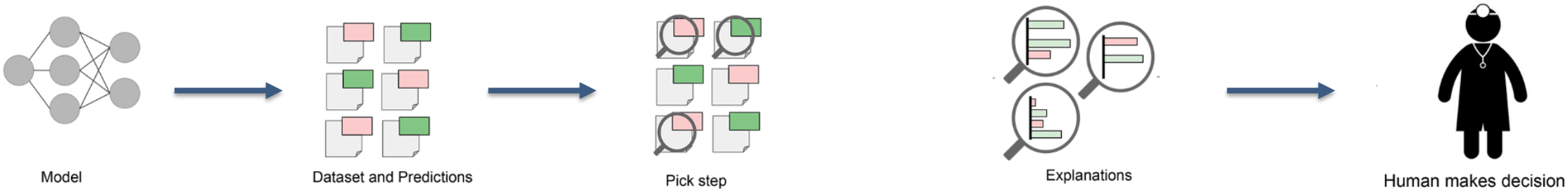
Uses:

- Pragmatic forecasting of biological processes.
- Tends to concern less regarding the data-generating process.

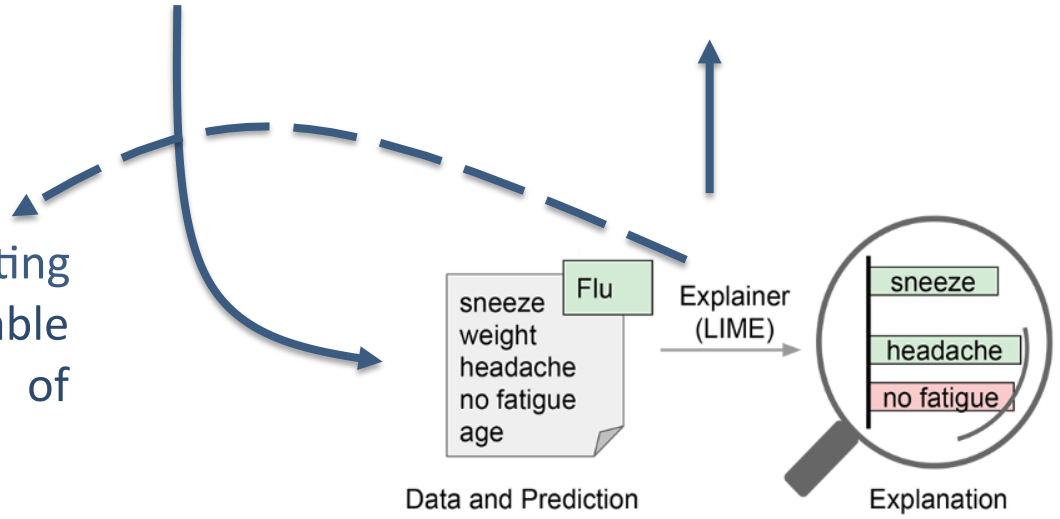
(Bzdok, et al., 2018, *bioRχiv)*

Diagnosing Features (≈ Interpretability)

Local Interpretable Model-Agnostic Explanations [***LIME***]



- 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.



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