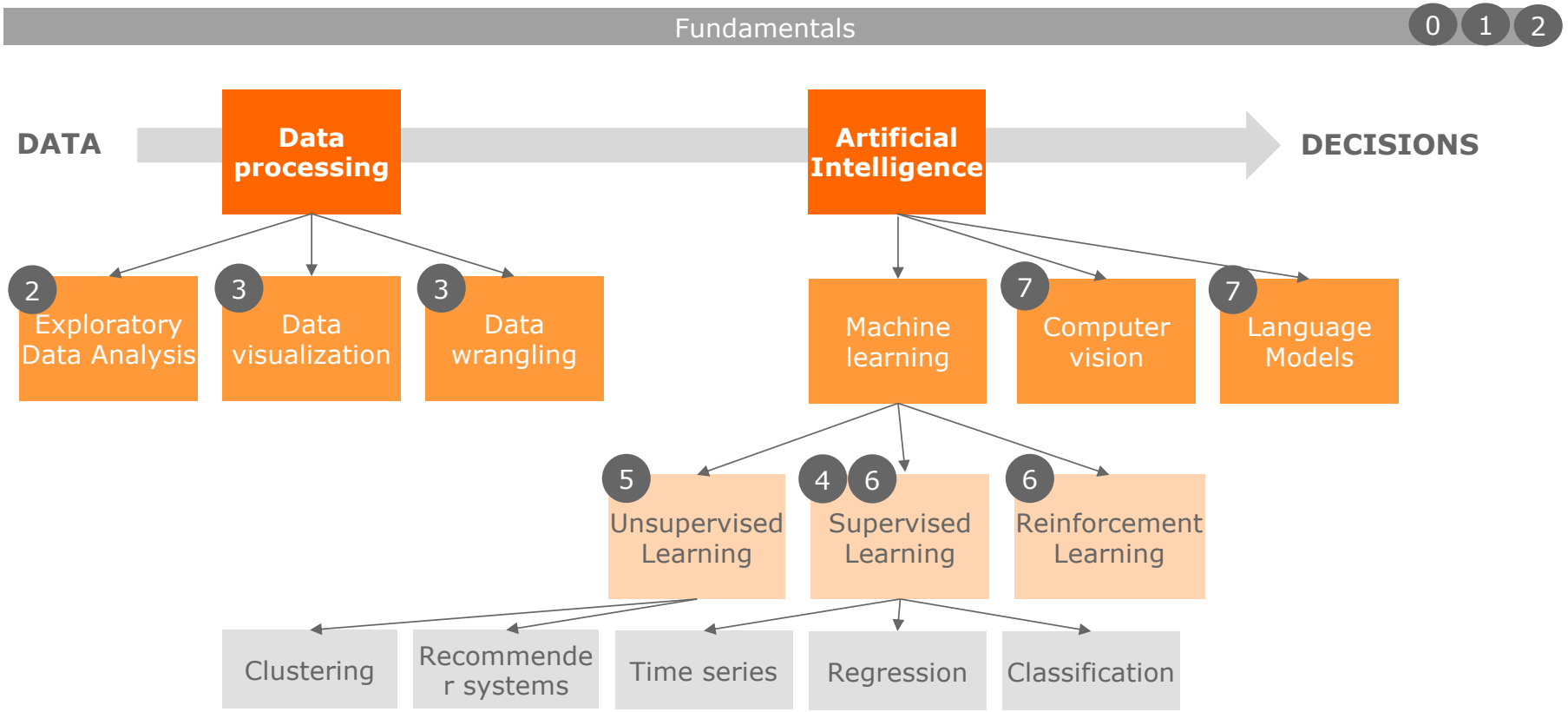
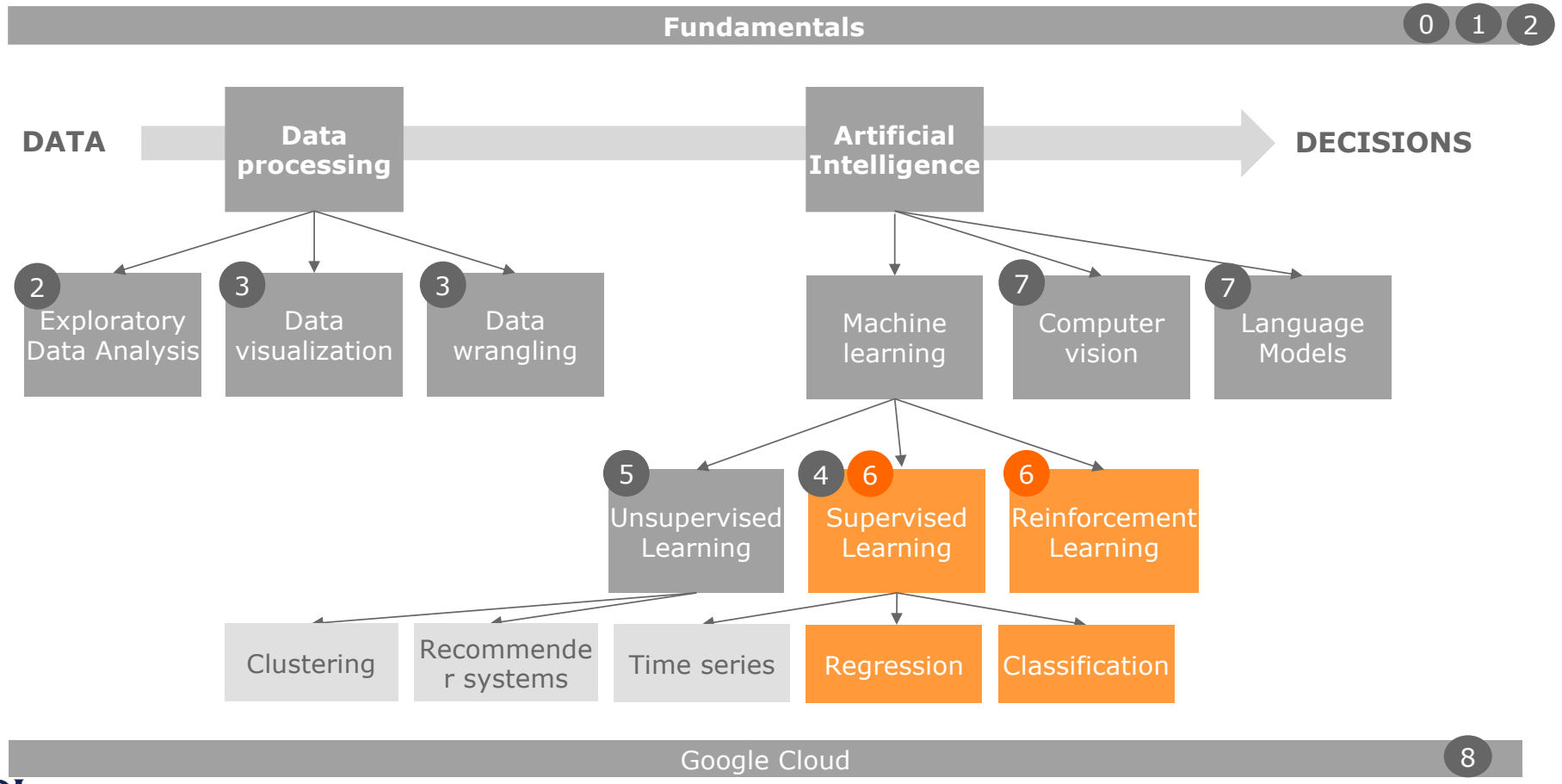

Porto Business School
/ University of Porto
BPI
Grupo CaixaBank
Advanced Learning models
Advanced Predictive Algorithms
Carlos Soares | Lisboa, 23/10/2023
ACCREDITATIONS
MEMBERSHIPS
RANKINGS
Associação Porto Business School (PBS) - U. Porto


Fundamentals
0 1 2
DATA
Data processing
Artificial Intelligence
DECISIONS
2 Exploratory Data Analysis
3 Data visualization
3 Data wrangling
Machine learning
7 Computer vision
7 Language Models
5 Unsupervised Learning
4 6 Supervised Learning
6 Reinforcement Learning
Clustering
Recommender systems
Time series
Regression
Classification
Google Cloud
8


Fundamentals
0
1
2
DATA
Data processing
Artificial Intelligence
DECISIONS
2
Exploratory Data Analysis
3
Data visualization
3
Data wrangling
Machine learning
7
Computer vision
7
Language Models
5
Unsupervised Learning
4
6
Supervised Learning
6
Reinforcement Learning
Clustering
Recommender systems
Time series
Regression
Classification
Google Cloud
8

Advanced Learning models

Session 1	Session 2	Session 3	Session 4
Advanced Predictive Modelling Ensemble Learning AutoML	**Neural Networks** Perceptron RNNs CNNs	**Reinforcement Learning** Multi-Armed bandits Q-Learning	**Responsible Modelling** Responsible AI Interpretability

Advanced Learning models

Session 1

09h30	algorithm bias
10h00	ensemble learning: motivation
10:30	**Break**
10h45	ensemble learning: overview and discussion
11h30	**Break**
11h45	automl
12h45	**Wrap-up**

plan

- why do we need multiple algorithms?
- why do we need multiple models?
- a simpler view of ensembles
- ensembles
  - overview
  - popular methods
  - discussion

based on materials kindly provided by João Mendes Moreira and Eamonn Keogh

goals

- understand the importance of having an intuition about the behavior of algorithms
- understand the basic principles of ensemble learning
- understand the intuition and high-level algorithm of some of the most common ensemble methods

why do we need many algorithms?

so far

- linear/logistic
- trees
- svm
- knn

later

- neural networks

what is the difference between algorithms?

LD

1. Compute the d-dimensional mean vectors for the different classes from the dataset.
2. Compute the scatter matrices (in-between-class and within-class scatter matrix).
3. Compute the eigenvectors ($\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_d$) and corresponding eigenvalues ($\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_d$) for the scatter matrices.
4. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a $d \times k$ dimensional matrix $\boldsymbol{W}$ (where every column represents an eigenvector).
5. Use this $d \times k$ eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the matrix multiplication: $\boldsymbol{Y} = \boldsymbol{X} \times \boldsymbol{W}$ (where $\boldsymbol{X}$ is a $n \times d$-dimensional matrix representing the n samples, and $\boldsymbol{y}$ are the transformed $n \times k$-dimensional samples in the new subspace).

https://sebastianraschka.com/Articles/2014_python_lda.html

DT

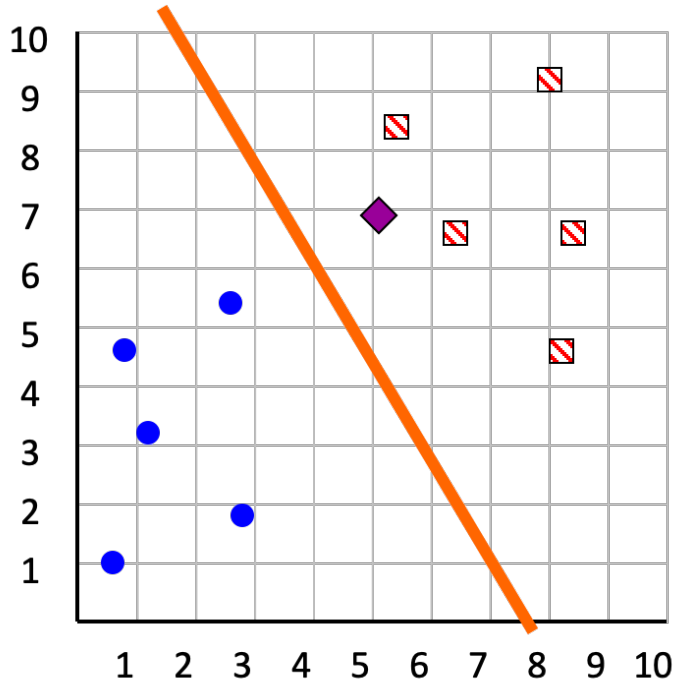
Algorithm Hunt decision tree induction algorithm.

```
1: INPUT D_train current node training set
2: INPUT p the impurity measure
3: INPUT n the number of objects in the training set
4: if all objects in D_train belongs to the same class y then
5:     The current node is a leaf node labeled with class y
6: else
7:     Select a predictive attribute to split D_train using the impurity measure p
8:     Split D_train in subsets according to its current values
9:     Apply Hunt algorithm to each subset
```

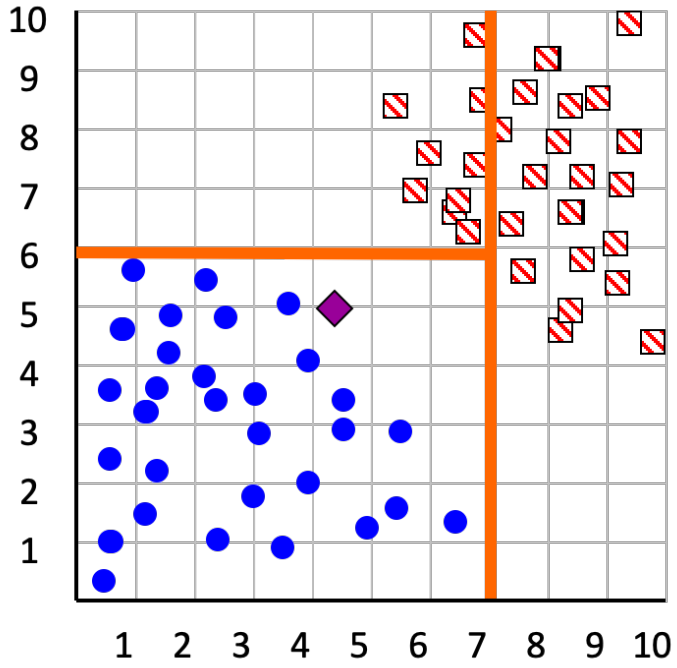
JMM et al (ch. 10)

the difference between algorithms: geometric intuition

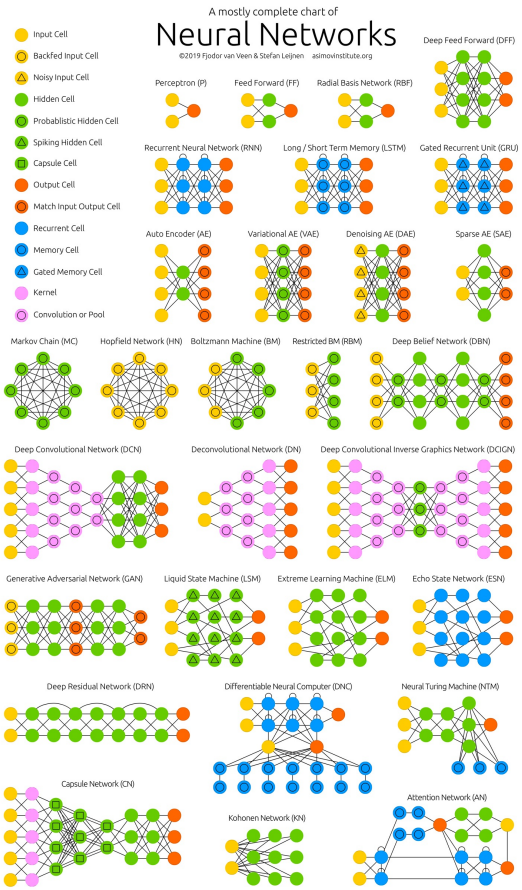
LD

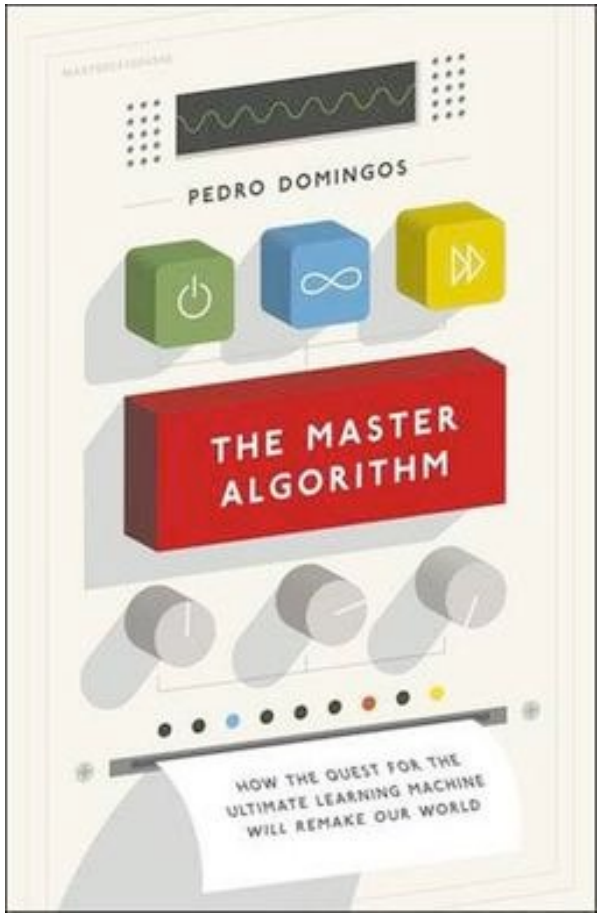


DT



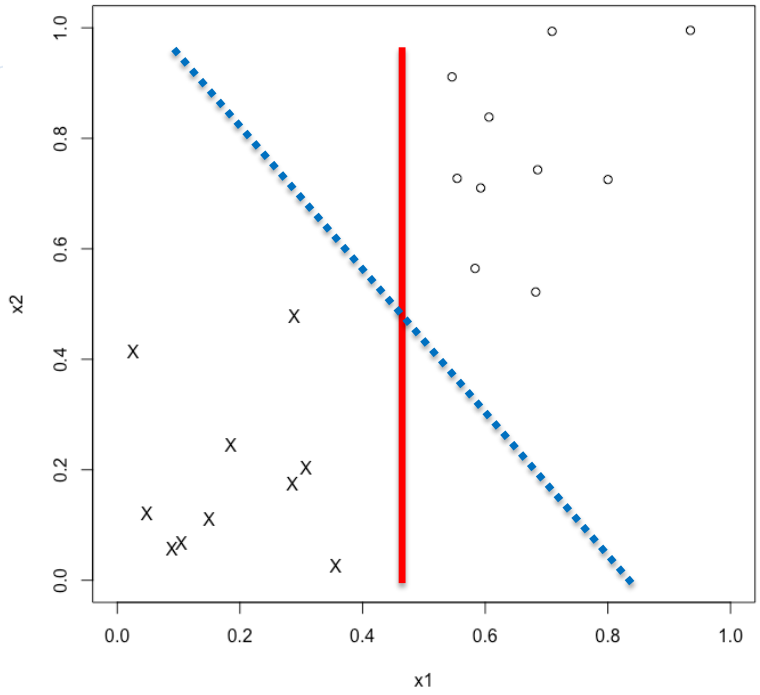
wait, is this really a problem?



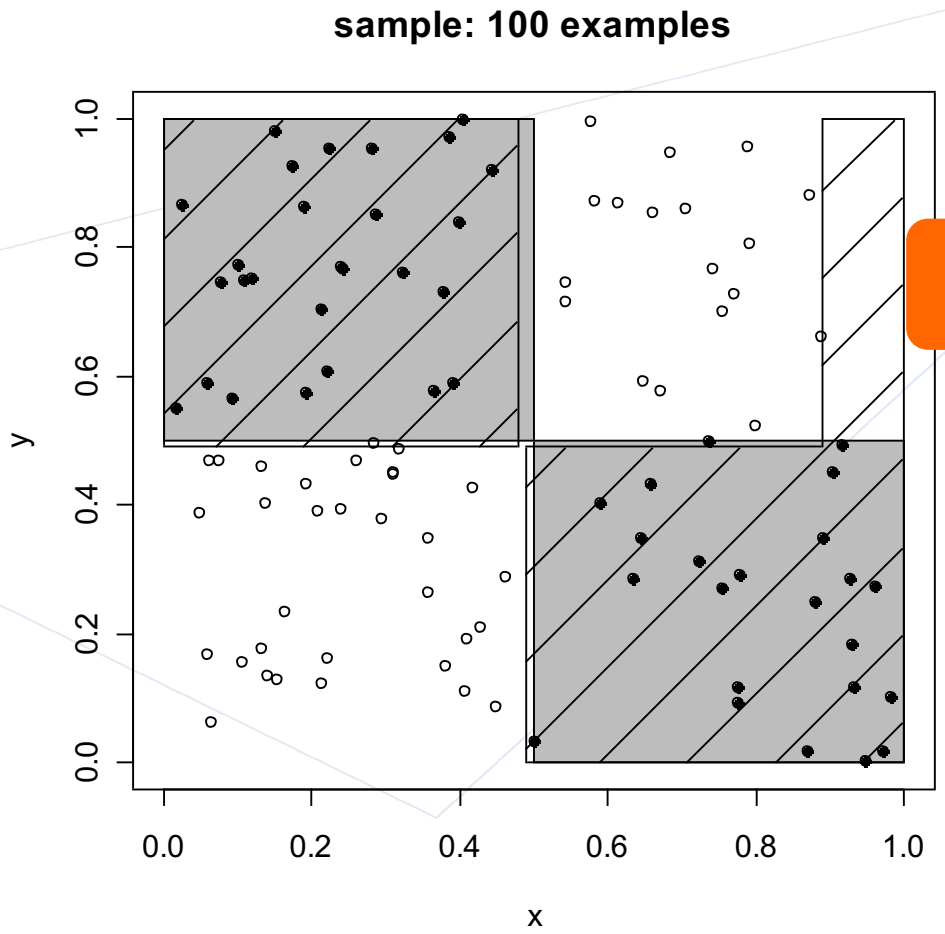


bias: can't live with it

- given
  - dataset
  - learning algorithm
- not every model is possible
  - e.g. DT and LD
  - … but not DT and LD

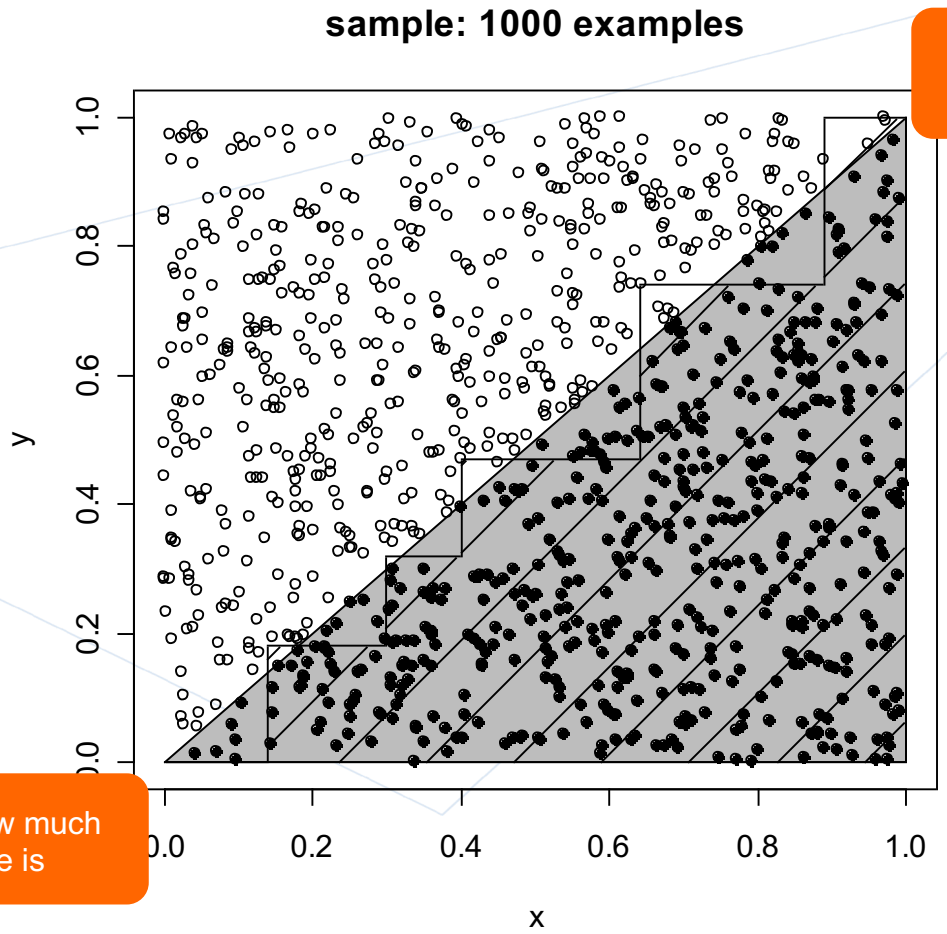


an algorithm may be suitable for a problem



even if not perfect

… or not



… or how simple it is

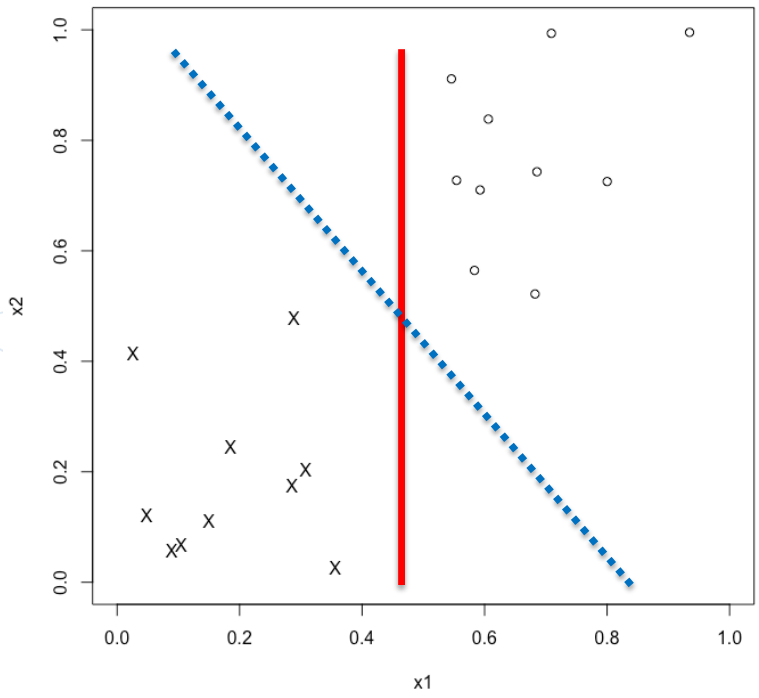
no matter how much data there is

… and can't live without it

- bias-free learning is futile
  - an algorithm that assumes nothing concerning the function it is trying to learn has no rational basis to classify unknown cases
    - Mitchell 97, Ch. 2

- bias = criteria to prefer one model relative to another
  - … so, how to select the best model if all models are considered equally suitable?

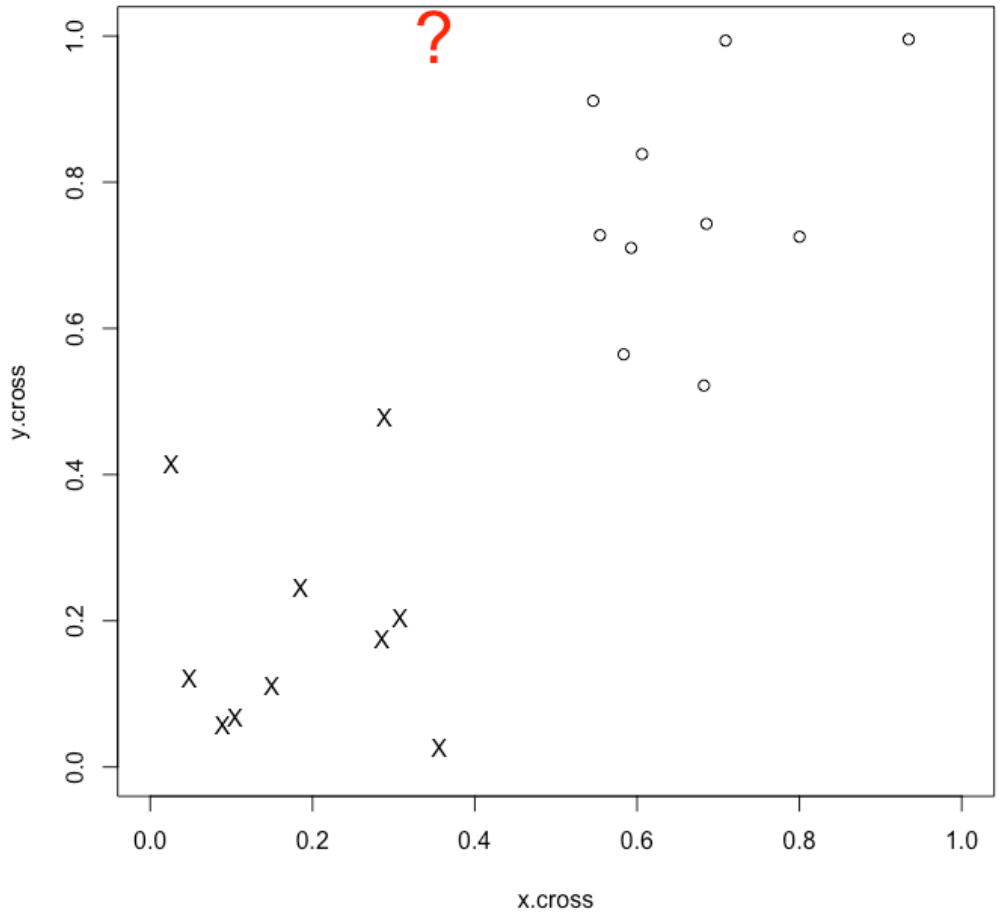
the bias-free algorithm



- … can learn any model
  - e.g. the DT and the LD
- … but doesn't have any way to preference for one over the other
  - … or for one over the other

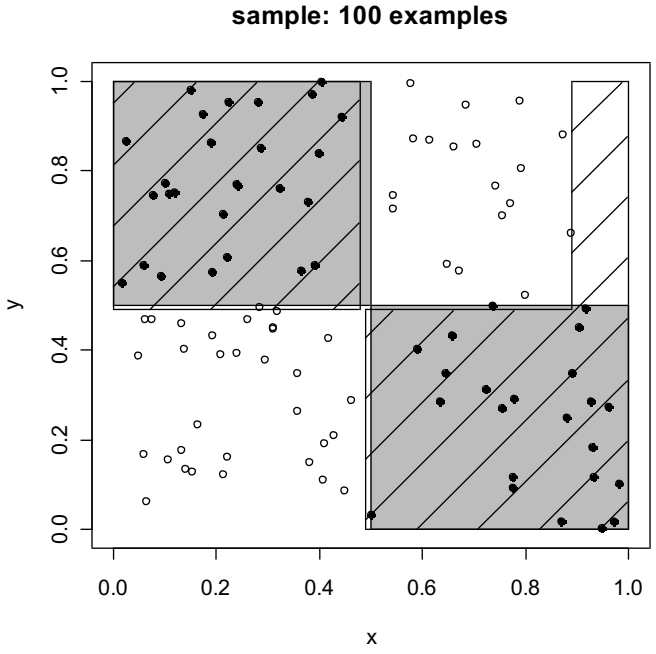
- on the other hand, algorithms with bias can!
  - DT prefers one
  - LD prefers the other

… right?

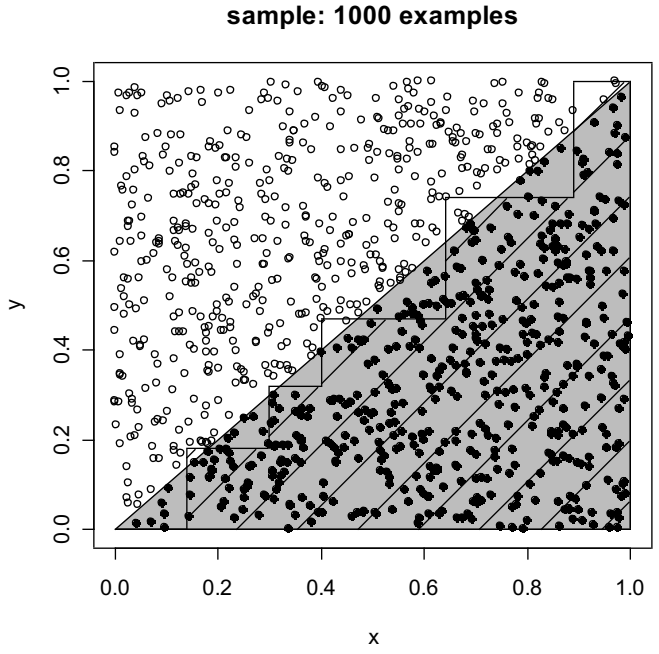


(im)perfect solution: empirical evaluation

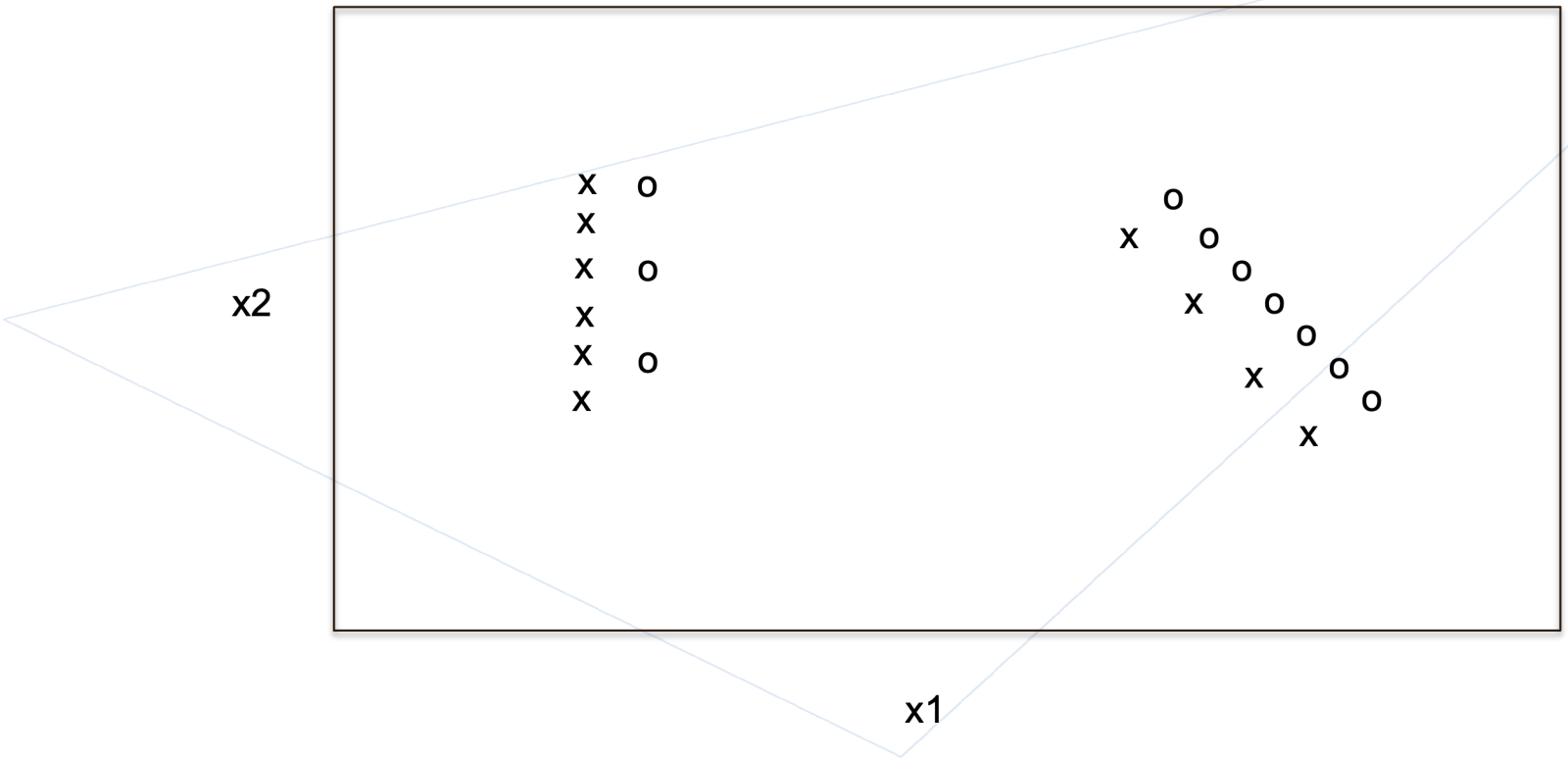
DT is better than LD



LD is better than DT

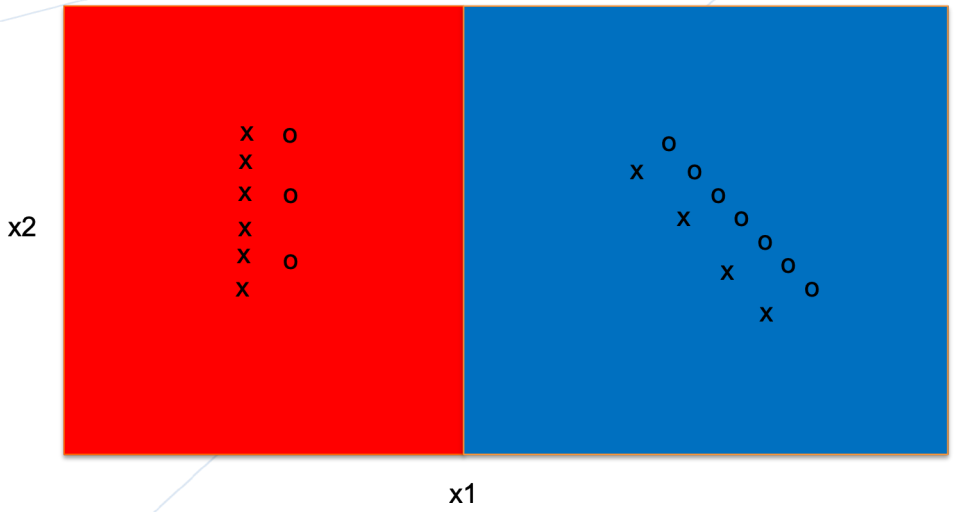


but it is possible to learn the best model for a problem with a single algorithm?



we need to integrate multiple models

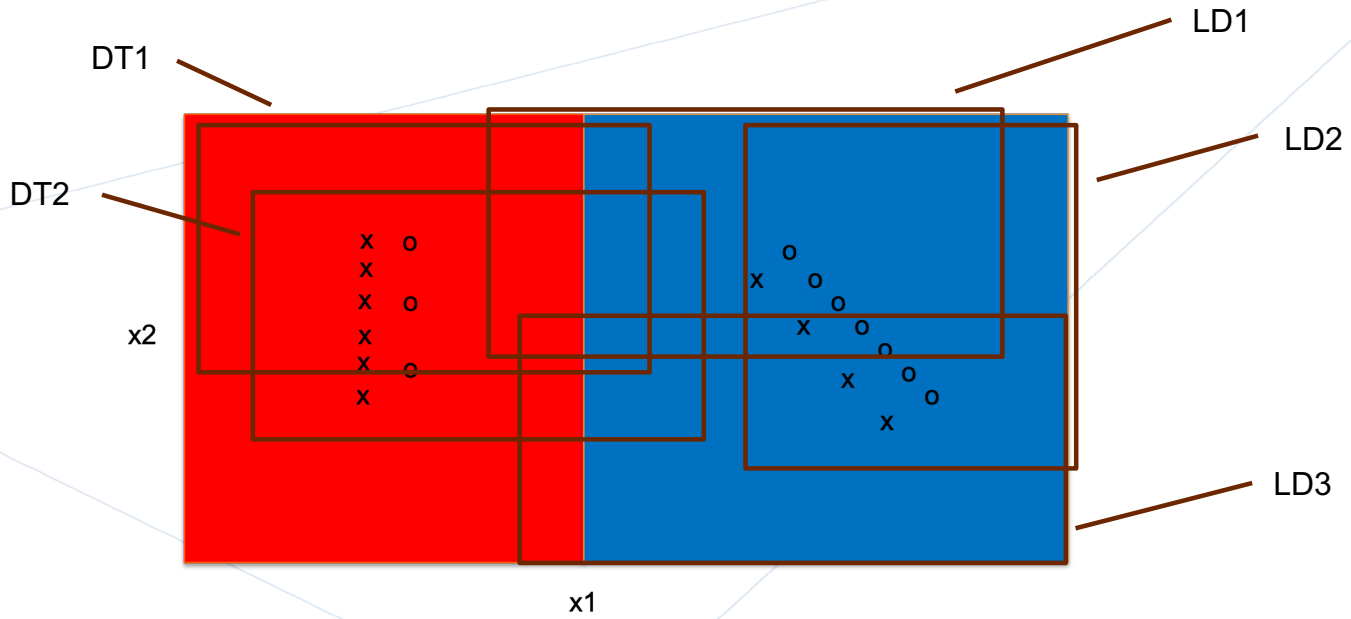
- if x1 < 0.4
  - DT
- else
  - LD



- … requires solving two very hard problems
  - find best bias for a region
  - find best model for the selected bias on the selected region

a simpler approach: the wisdom of the crowds

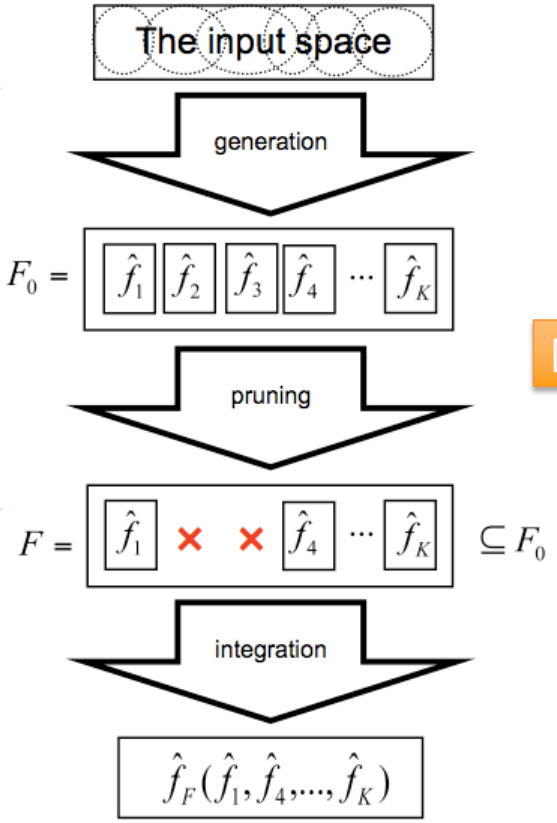
- different models have different areas of expertize



- ensemble
  - ask all
  - combine predictions

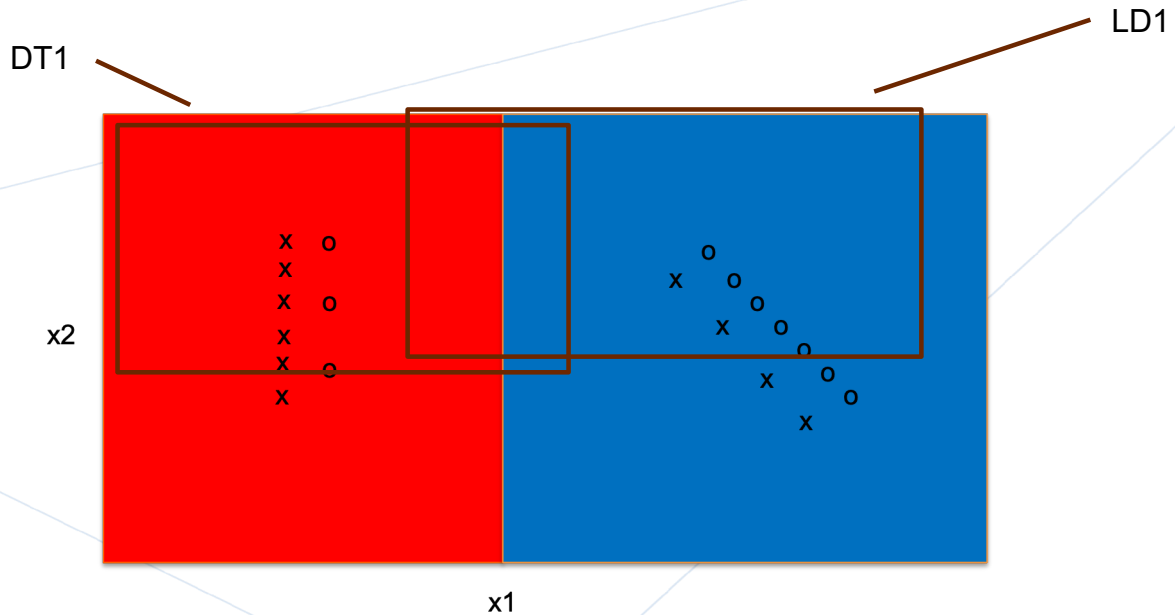
ensemble learning

- multiple models
  - base models
- … each of them obtained by applying a different learning process to a given problem
  - e.g. different algorithms
- … combined to make a single prediction
  - e.g. in classification, each model makes a prediction
  - … then combined to obtain the final prediction of the ensemble



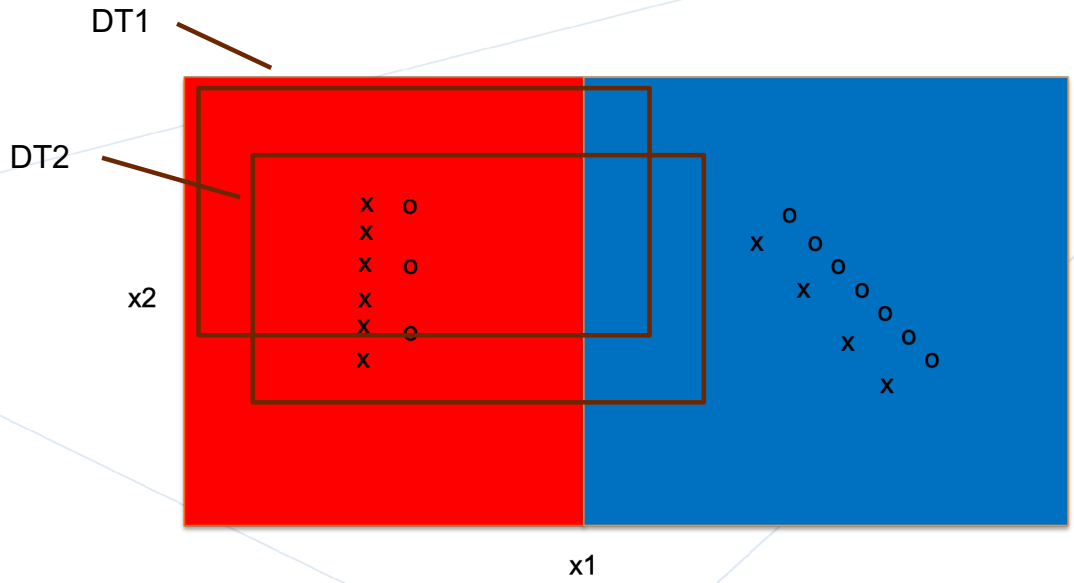
different algorithms, different models

heterogeneous ensembles



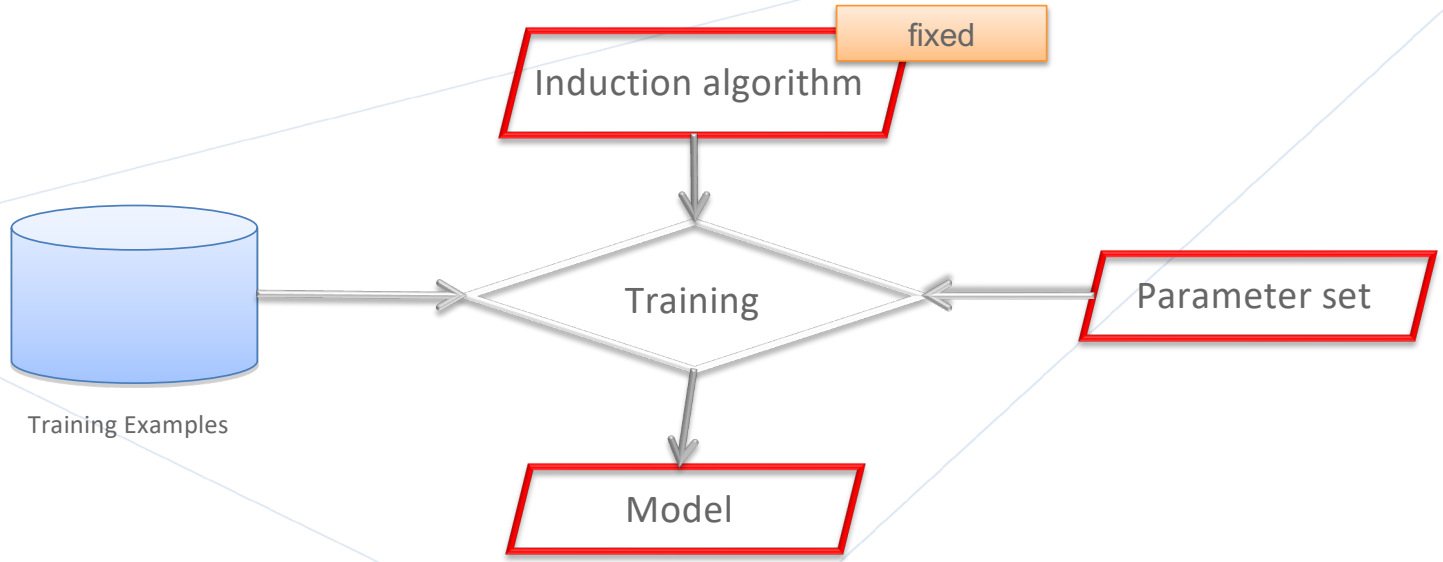
… but we don't have that many algorithms

- homogeneous ensembles
  - how?

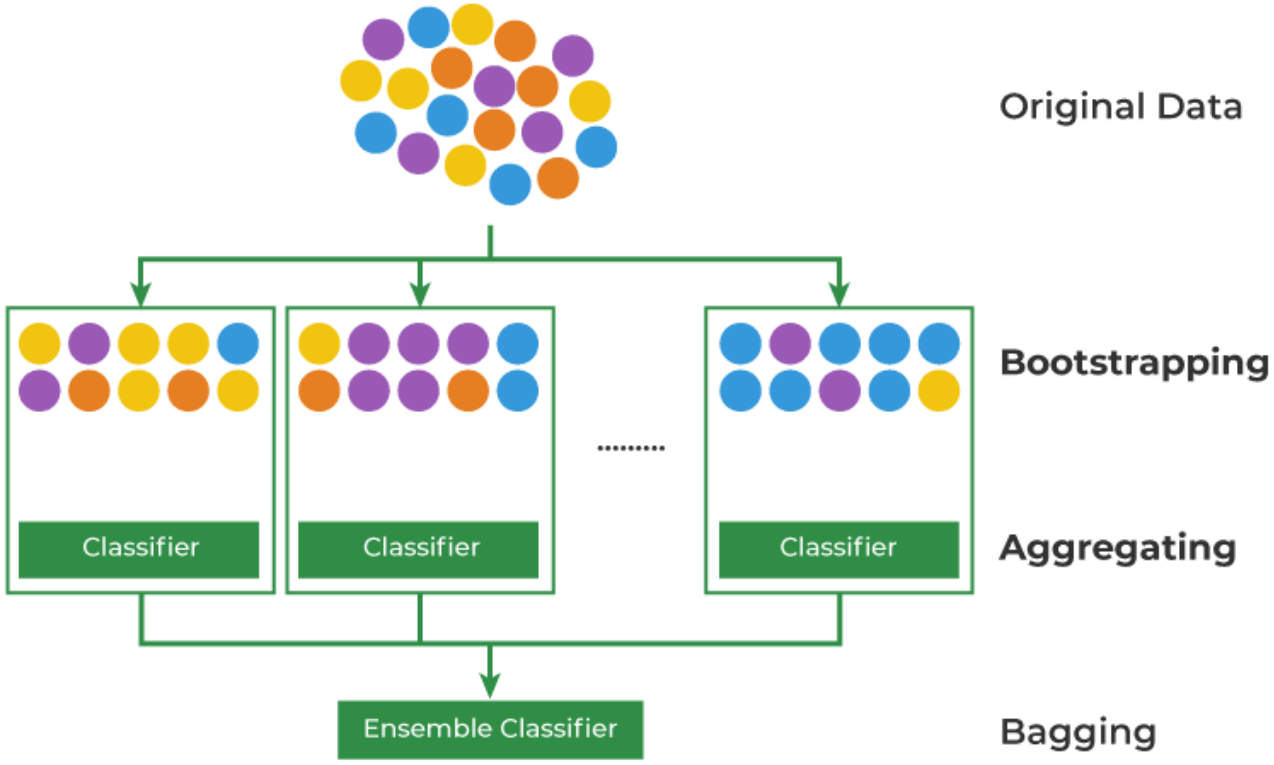


- the most common type!

homogeneous ensembles: how to generate different models?



bagging: a (simple) classic



https://www.geeksforgeeks.org/ml-bagging-classifier/

exercise

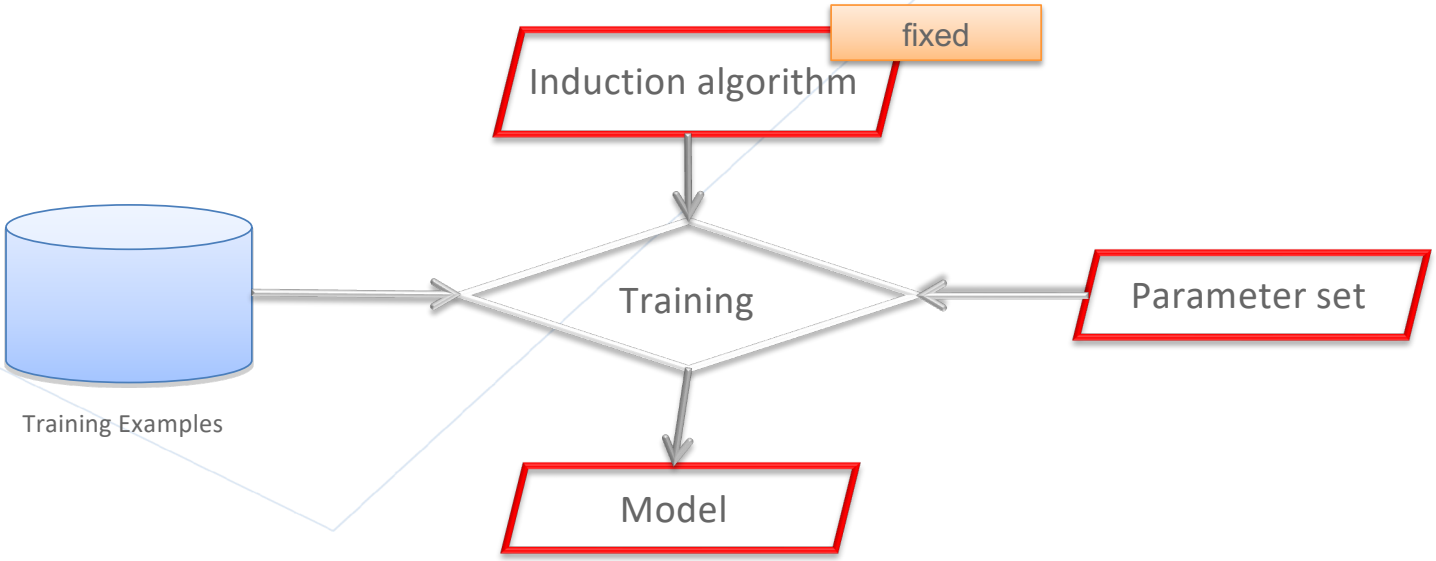
- implement bagging
  - notebook: Bagging (dev) exercise

gps

- why do we need multiple algorithms?
  - right bias depends on problem
- why do we need multiple models?
  - right bias depends on the region
- a simpler view of ensembles
  - wisdom of the crowds-ish
  - exercise: implement bagging
- ensembles
  - overview
  - popular methods
  - discussion

homogeneous ensembles: how to generate different models?

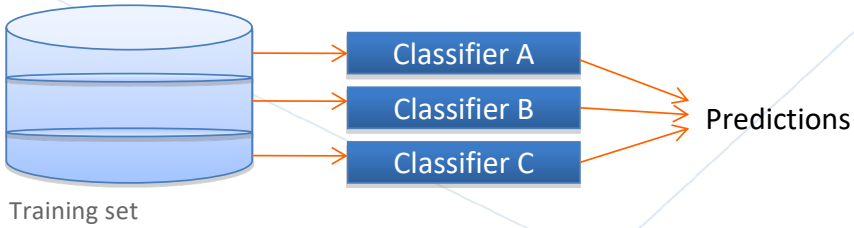
- Data manipulation
  - training set
- Modeling process manipulation
  - induction algorithm
    - variant of the selected algorithm
  - parameter set
  - model
    - uncommon



data manipulation

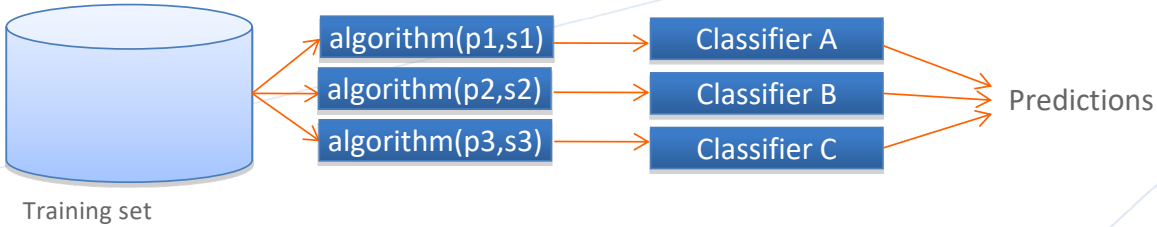
- Manipulating the input features

- Sub-sampling from the training set

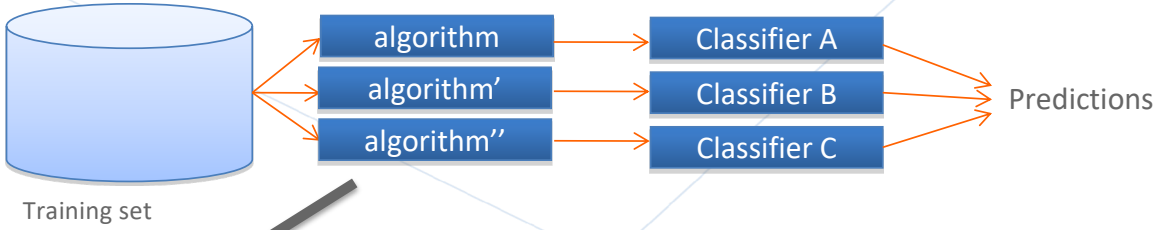


modeling process manipulation

- Manipulating the parameter sets



- Manipulating the induction algorithm



still homogeneous: *algorithm'* and *algorithm''* are variations of *algorithm*

gps

- why do we need multiple algorithms?
  - right bias depends on problem
- why do we need multiple models?
  - right bias depends on the region
- a simpler view of ensembles
  - wisdom of the crowds-ish
  - exercise: implement bagging
- ensembles
  - overview
  - popular methods
    - bagging
    - boosting
    - random forest
    - negative correlation
  - discussion

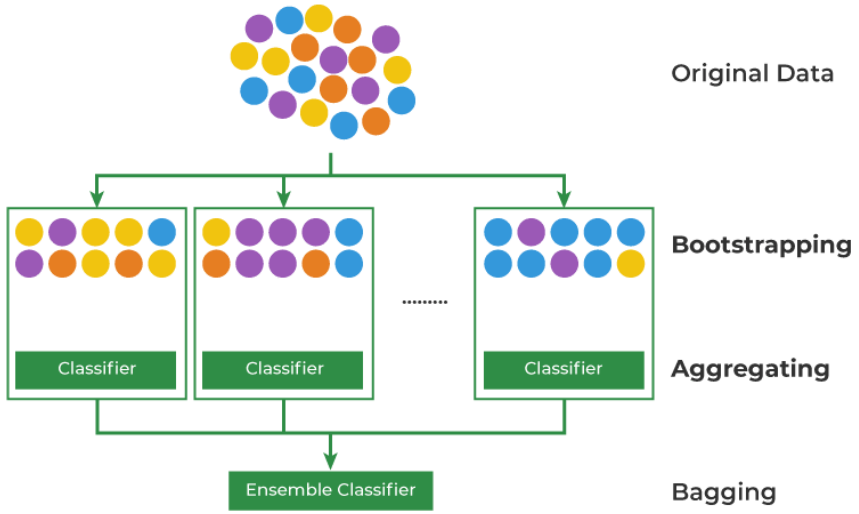
bagging: Bootstrap AGGregatING

training

- given a set D of d tuples
- at each iteration i
  - training set D_i of d tuples is sampled with replacement from D
    - i.e. bootstrap
  - model M_i is learned for training set D_i

prediction

- given an observation X
- for each classifier M_i
  - make a prediction
- an aggregation of the predictions is the prediction of the bagged model M^* for X



http://en.wikibooks.org/wiki/File:DTE_Bagging.png

bagging

analogy

- diagnosis based on the majority vote of multiple doctors
- trained in slightly different contexts

accuracy

- often significantly better than a single classifier derived from *D*
- robust to noise

… if classifier is unstable!

- unstable means a small change to the training data may lead to major decision changes
  - decision trees
  - neural networks

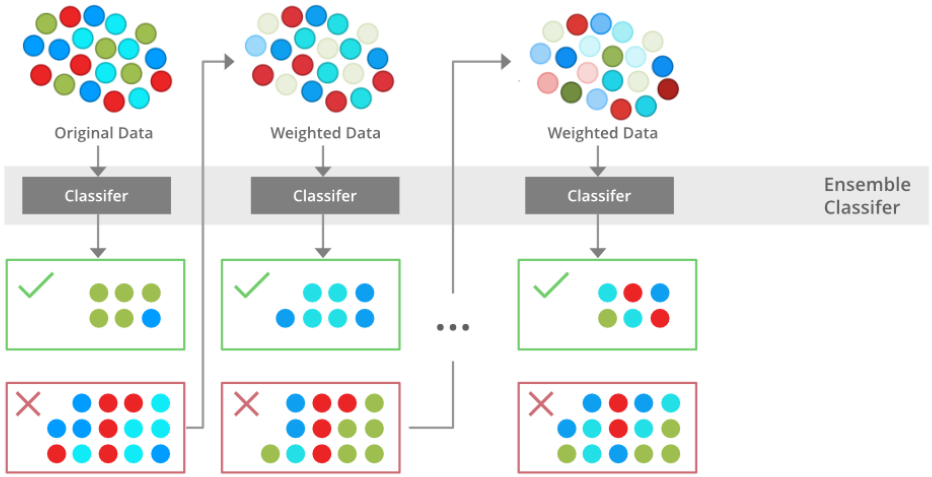
boosting

training

- equal weights are assigned to each training example
- learn model M_1
- learn additional *k-1* models
  - give more weight to the examples that were incorrectly predicted by M_i
  - learn model M_{i+1}

prediction

- given an observation *X*
- for each classifier M_i
  - make a prediction
- an aggregation of the predictions is the prediction of the bagged model *M** for *X*
  - the weight of each classifier's vote is a function of its accuracy



https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/

boosting: discussion

analogy

- diagnosis based on the majority vote of multiple doctors
- trained sequentially to solve the errors of the previous one
- weighted by ability

boosting vs. bagging

- differences
  - independent sampling vs. error-dependent sampling
  - uniform aggregation vs. weighted aggregation

… so

- boosting tends to achieve greater accuracy
- … but it also risks overfitting the model to misclassified data

random forest

training

- learn *k* models
- … with changed algorithm
  - at each split
    - randomly select a subset of the original features during the process of tree generation

prediction

- given an observation *X*
- for each classifier M_i
  - make a prediction
- an aggregation of the predictions is the prediction of the bagged model *M** for *X*

random forest: discussion

analogy

- diagnosis based on the majority vote of multiple doctors
- trained to use different(-ish) tools

RF vs adaboost

- comparable in accuracy
- more robust to errors and
- … outliers

… vs bagging and adaboost

- RF is insensitive to the number of attributes selected for consideration at each split and
- faster

negative correlation learning

training

- learn *k* models
- … with changed algorithm
  - trained to minimize the error function of the ensemble
    - i.e., it adds to the error function a penalty term with the average error of the models already trained

prediction

- given an observation *X*
- for each classifier M_i
  - make a prediction
- an aggregation of the predictions is the prediction of the bagged model *M** for *X*

negative correlation learning: discussion

analogy

- diagnosis based on the majority vote of multiple doctors
- trained with awareness of the quality of the group

only regression

- algorithms that try to minimize/maximize a given objective function
  - e.g., neural networks, support vector regression

models negatively correlated with the averaged error of the previously generated models

exercise

- compare popular ensemble methods
  - notebook: Compare ensemble methods exercise

ensembles methods for...

classification

regression

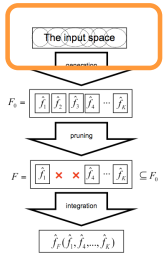
clustering

- aka consensual clustering

label ranking

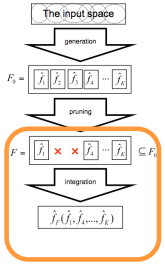
...

anything, really



types of ensembles: how to combine models

regression	classification
average	**majority voting**
weighted average	**weighted majority voting**
sum	borda count
weighted sum	
product	
maximum	
minimum	
median	



gps

- why do we need multiple algorithms?
  - right bias depends on problem
- why do we need multiple models?
  - right bias depends on the region
- a simpler view of ensembles
  - wisdom of the crowds-ish
  - exercise: implement bagging
- ensembles
  - overview
  - popular methods
  - discussion

exercise

- understanding bagging
  - notebook: solution to the Bagging (dev) exercise
  - what is the impact of varying the number of models?
    - repeat many times with 5 and 50 models
    - analyze the distribution of results
  - what is the impact of varying the size of the sample
    - sample 5% of the data
    - repeat many times with 5 models
    - analyze the distribution of results

characteristics of the base models: classification

base classifiers should be as accurate as possible and

- although there is "the strength of weak classifiers"
  - R.E. Schapire. 1990. The Strength of Weak Learnability. *Mach. Learn.* 5, 2 (July 1990), 197-227

… having diverse errors

- Brown, G. & Kuncheva, L., "Good" and "Bad" Diversity in Majority Vote Ensembles, *Multiple Classifier Systems, Springer,* **2010**, *5997*, 124-133

characteristics of the base models: regression

more amenable to theoretical analysis

- the error of an ensemble $\hat{f}_F$ with *K* base learners in relation to the true values given by f is:
  - $E(\hat{f}_F - f)^2 = \overline{bias}^2 + \frac{1}{K} \times \overline{var} + \left(1 - \frac{1}{K}\right) \times \overline{covar}$
  - … assuming the integration function is the average

the goal is to minimize $E(\hat{f}_F - f)^2$, so

- the average bias of the base learners should be as small as possible
  - i.e. the base learners should be as accurate (on average) as possible
- the average variance of the base learners should be as small as possible
  - i.e. the base learners should be as robust to small changes on the training data (on average) as possible
- the average covariance of the base learners should be as low as possible
  - i.e. the base learners should have negative correlation

pros & cons

+

accuracy
- majority compensates for individual errors

diversity is key
- individual models specialize in different areas of the data space
- how?

... but must be reasonably accurate
- ... and by “reasonable” we mean...?

-

complexity
- understanding the global model
- explaining decisions
- computational

remember Occam’s Razor
- simplicity leads to greater accuracy
- identifying the best model requires identifying the proper "model complexity”

- … and the NFL theorem

Wrap-up

better results with combination of multiple models: majority compensates for individual errors

individual models specialize in different areas of the data space: diversity is key

increase in complexity

plan

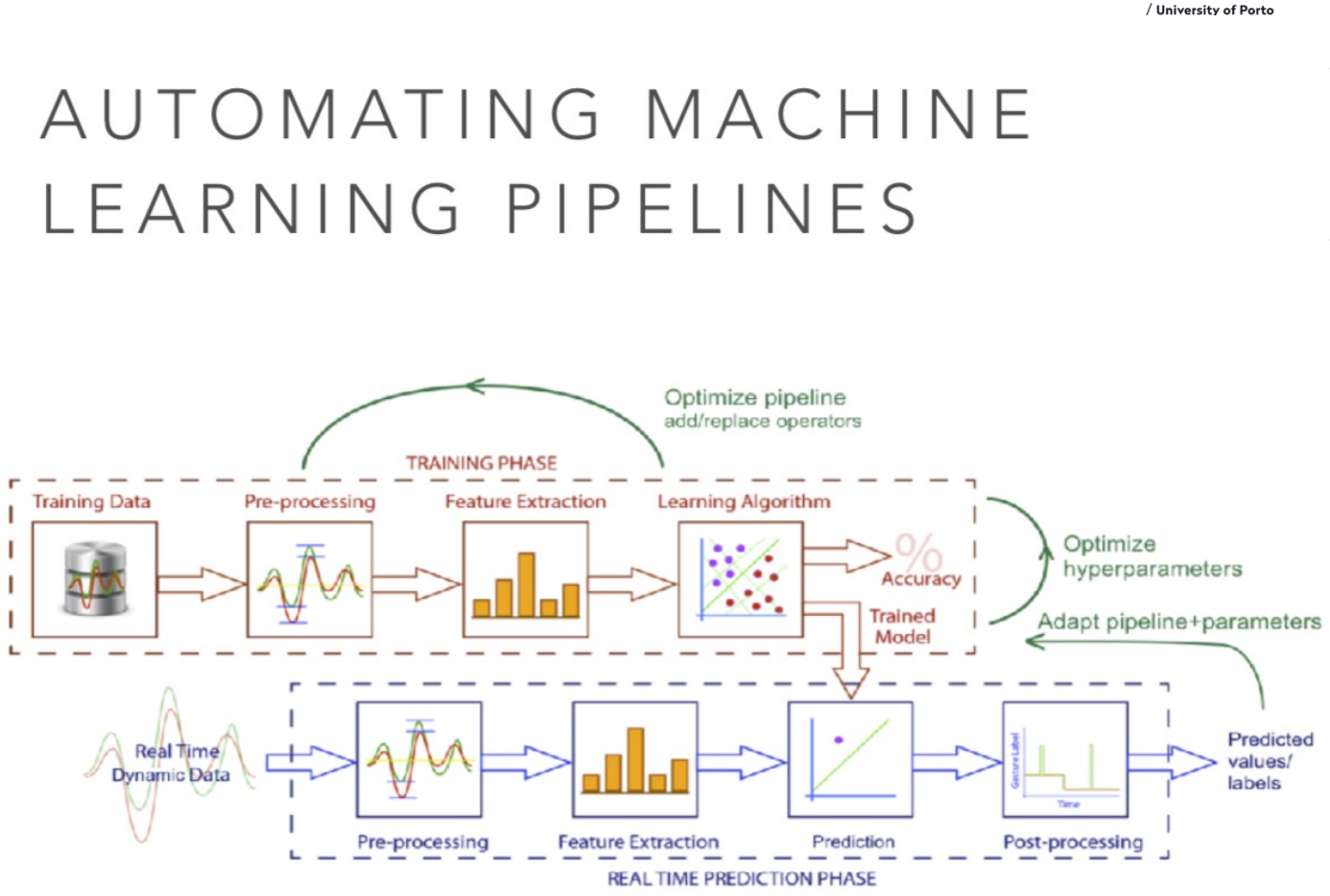
- why do we need automl
- hyperparameter tuning
- neural architecture search
- metalearning

based on materials kindly provided by Joaquin Vanschoren

goals

- understand the importance of having an intuition about the behavior of algorithms
- understand the basic principles of ensemble learning
- understand the intuition and high-level algorithm of some of the most common ensemble methods

designing the optimal pipeline is impossible!



https://openml.github.io/openml-tutorial/slides_pdf/AutoML%20Introduction.pdf

automl today

tasks

- hyperparameter tuning
  - most common
- neural architecture search
- pipeline configuration
  - well, essentially algorithm selection

approaches

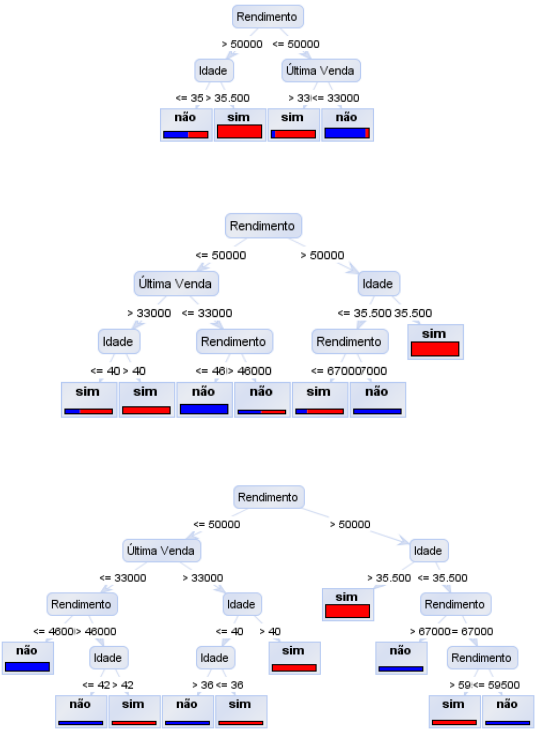
- search
- ... and/or metalearning

gps

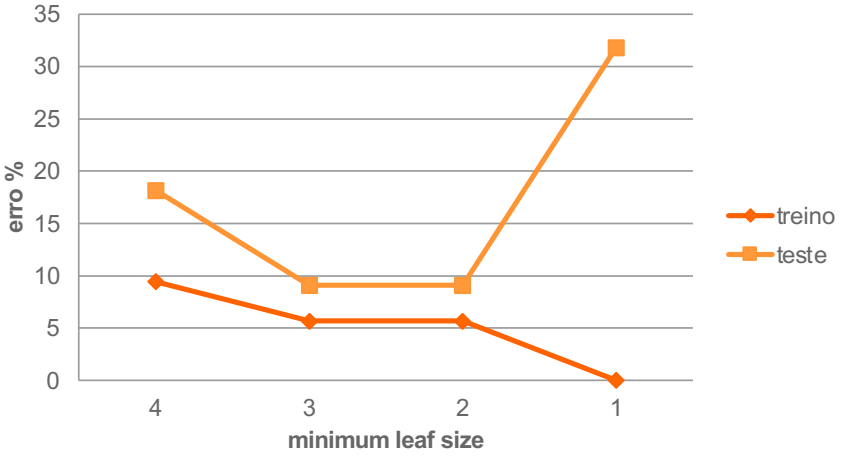
- why do we need automl
  - best pipeline varies depending on data
  - space is huge
- hyperparameter tuning
  - simple search: grid vs random
  - exercise: grid search vs random search
  - the need for validation
  - smarter searches
    - model-based
    - hyperband
- neural architecture search
- metalearning

the importance of hyperparameter tuning: the fine line between under- and overfitting

varyoing minimum leaf size



… affects predictive performance of DT model



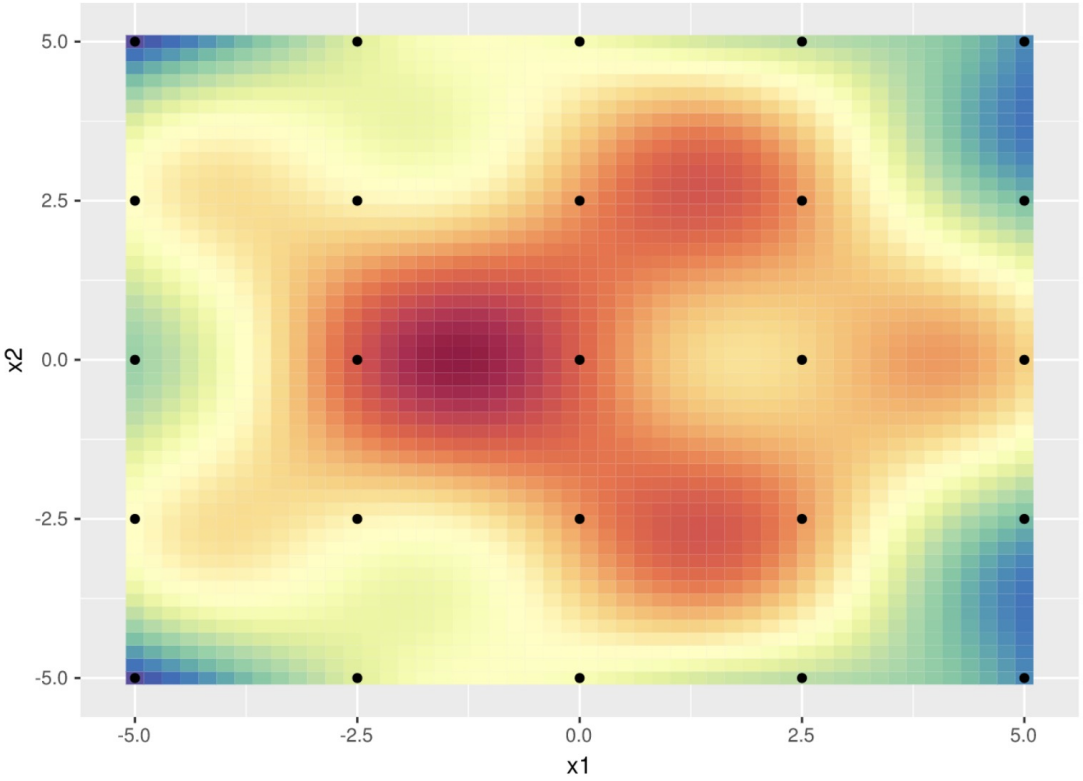
grid search

define grid

- e.g. minimum leaf size: {5, 25, 50, 100}
- … maximum depth: {3, 5, 10}

learn and evaluate models for all combinations

choose best



https://openml.github.io/openml-tutorial/slides_pdf/AutoML%20Introduction.pdf

random search

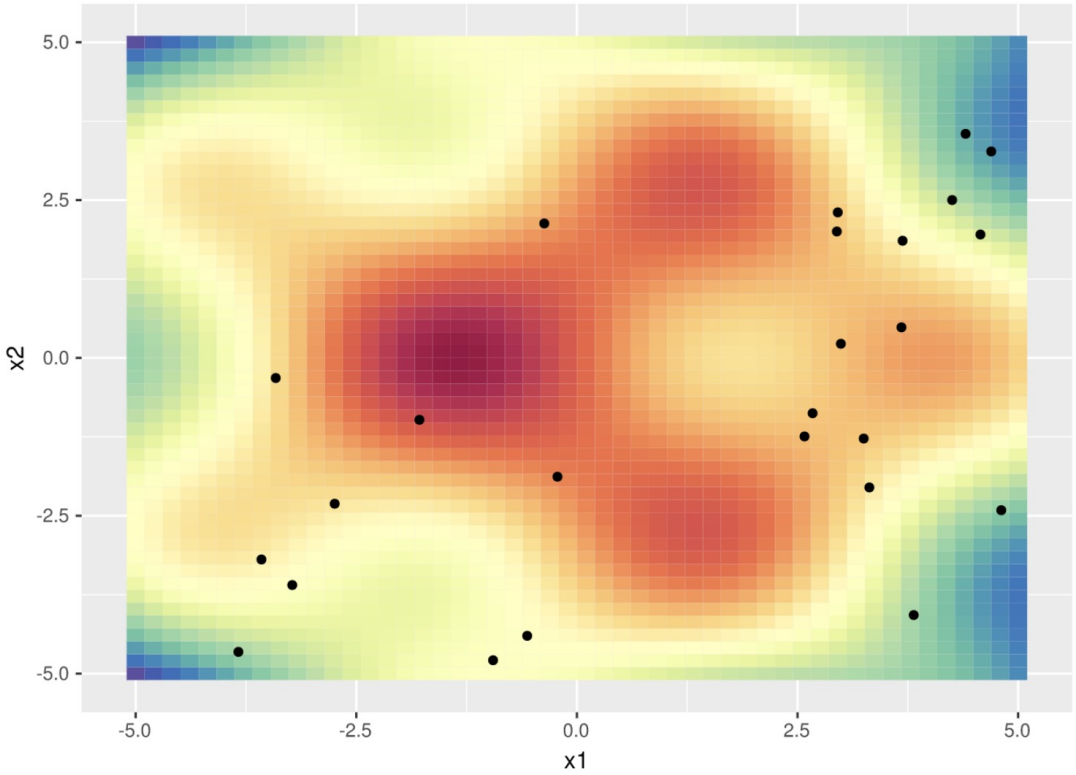
define domain

- e.g. minimum leaf size: {5,… 100}
- … maximum depth: {3, …, 10}

generate combinations randomly

learn and evaluate models for all combinations

choose best

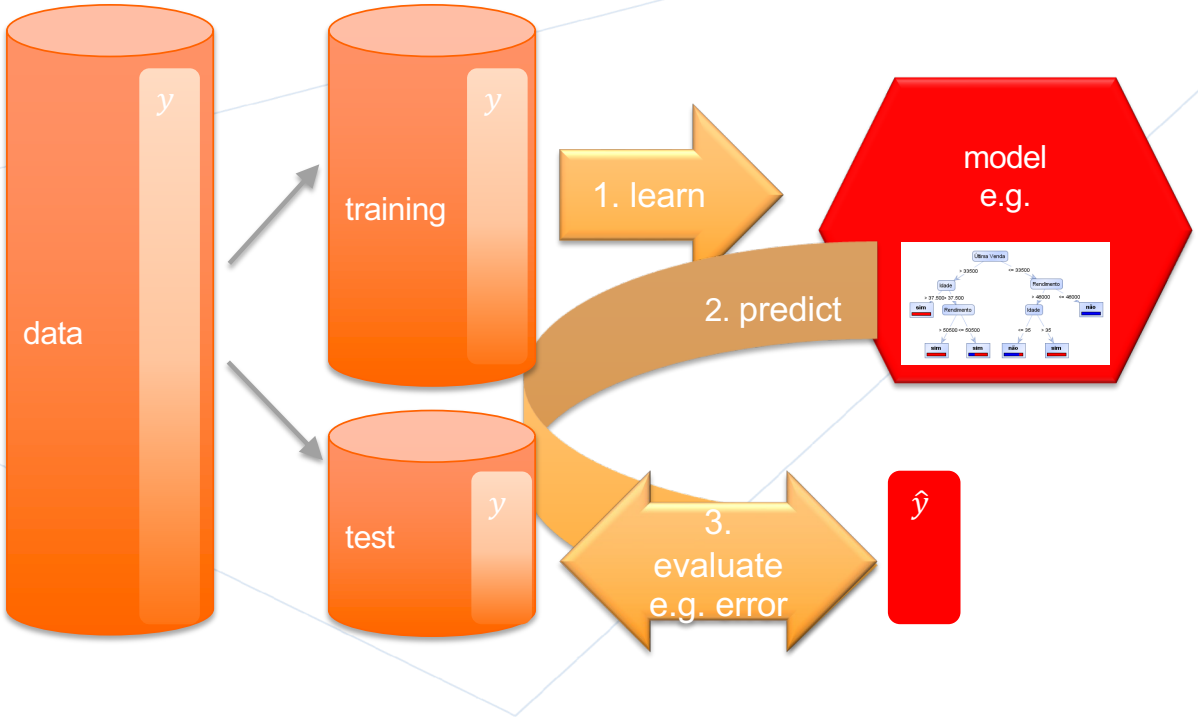


https://openml.github.io/openml-tutorial/slides_pdf/AutoML%20Introduction.pdf

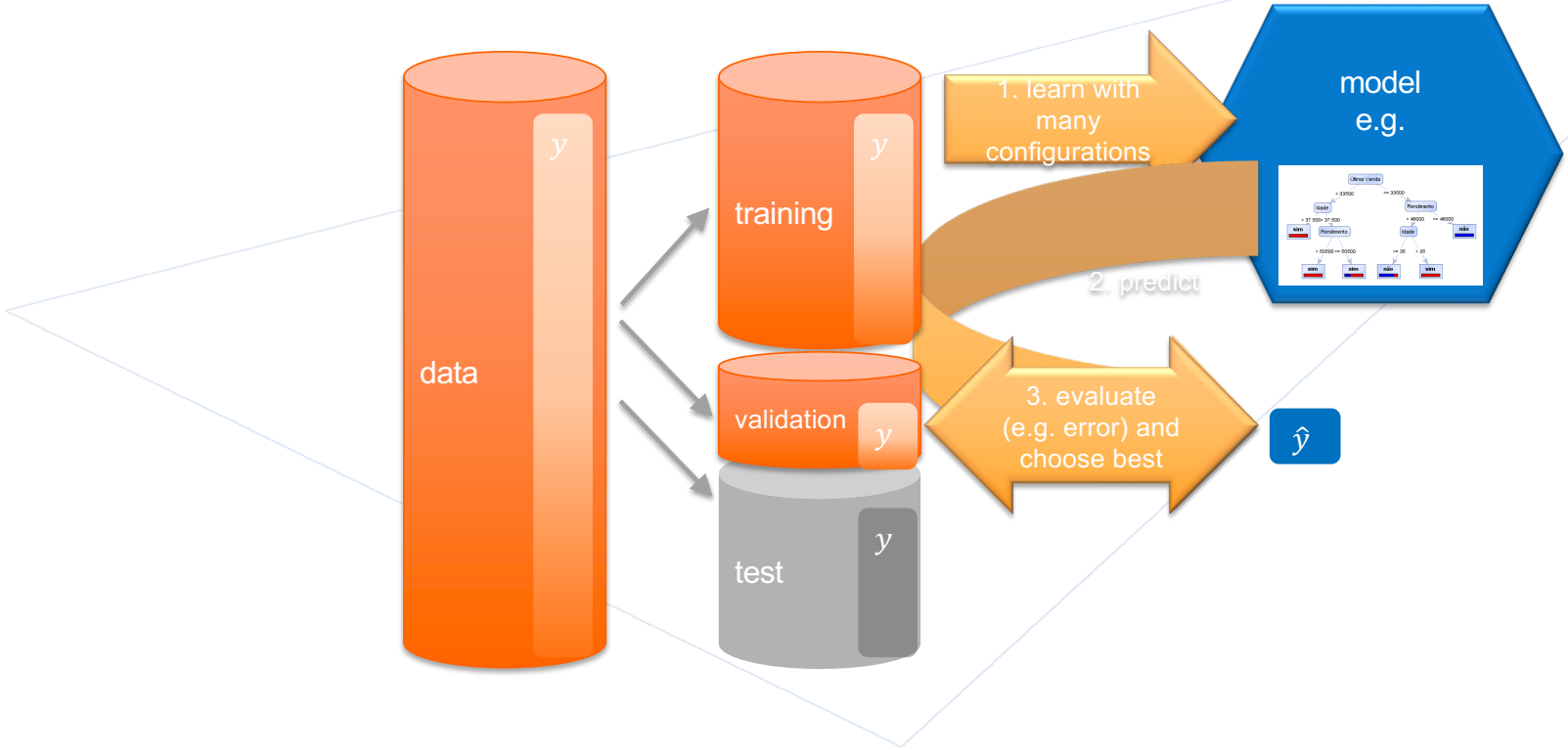
exercise

- compare grid and random search
  - notebook: Hyperparameter tuning DT exercise

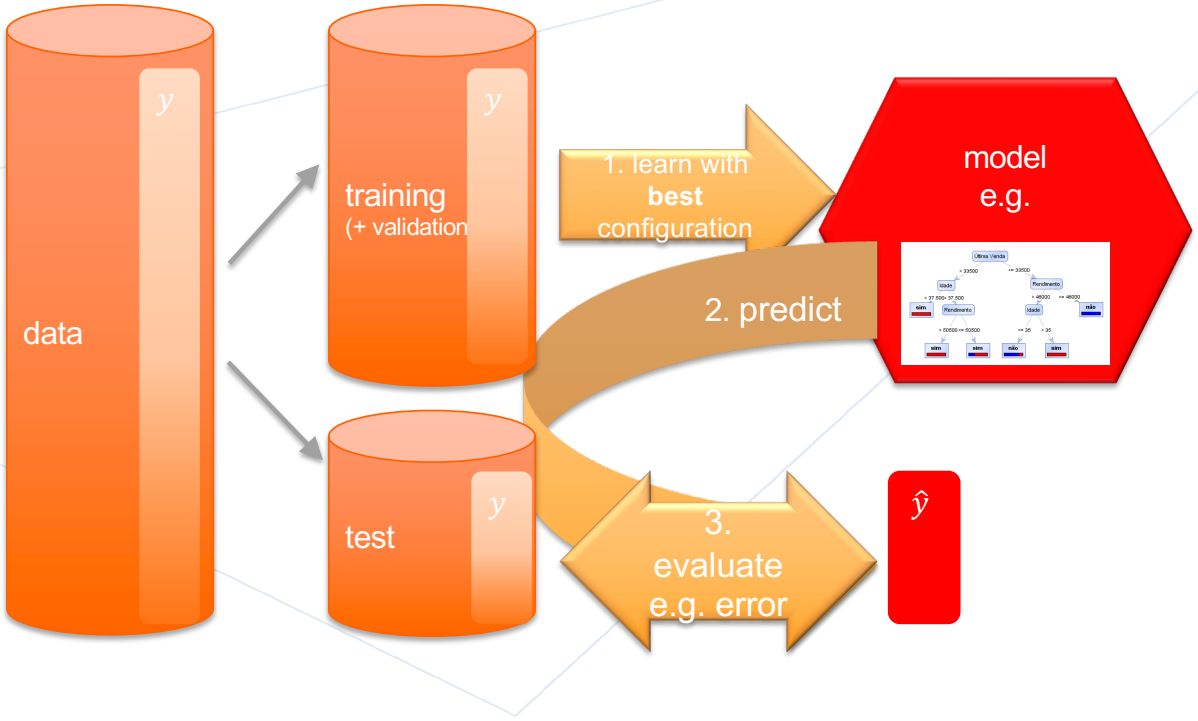
evaluation methodology: is this still suitable?



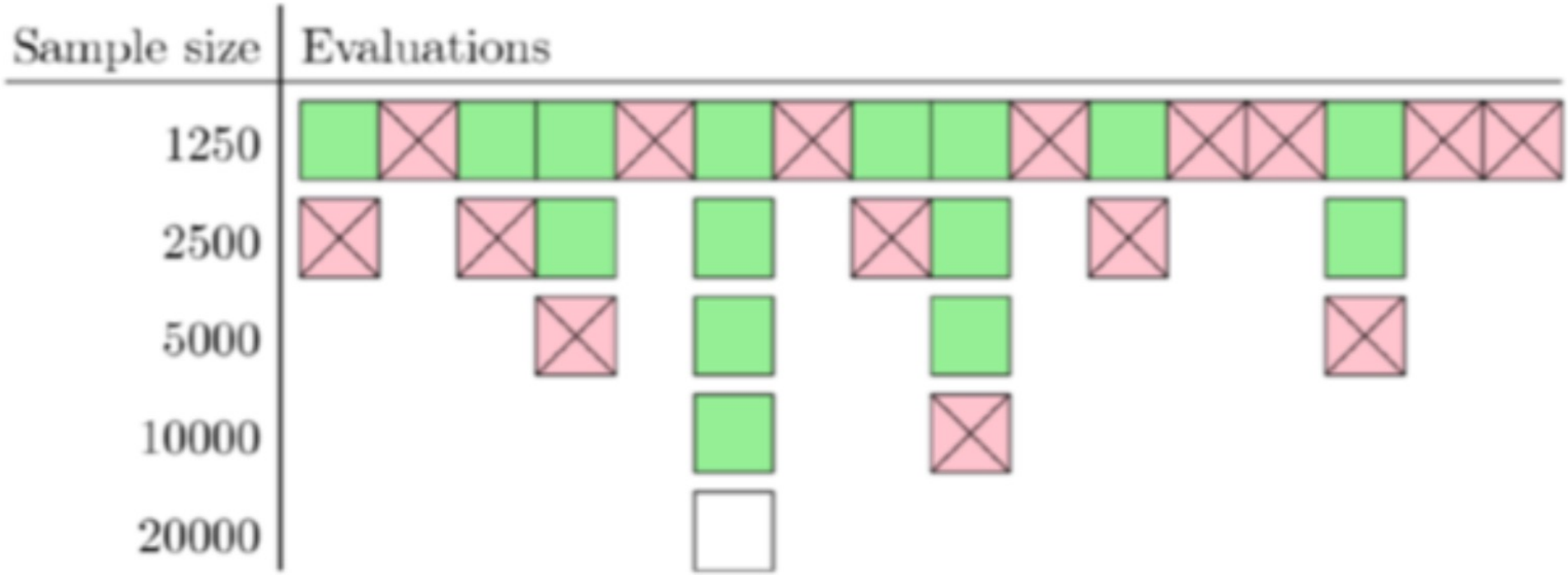
evaluation methodology with hyperparameter tuning (1/2)



evaluation methodology with hyperparameter tuning (2/2)



smarter search: hyperband



https://openml.github.io/openml-tutorial/slides_pdf/AutoML%20Introduction.pdf

guided search

define domain

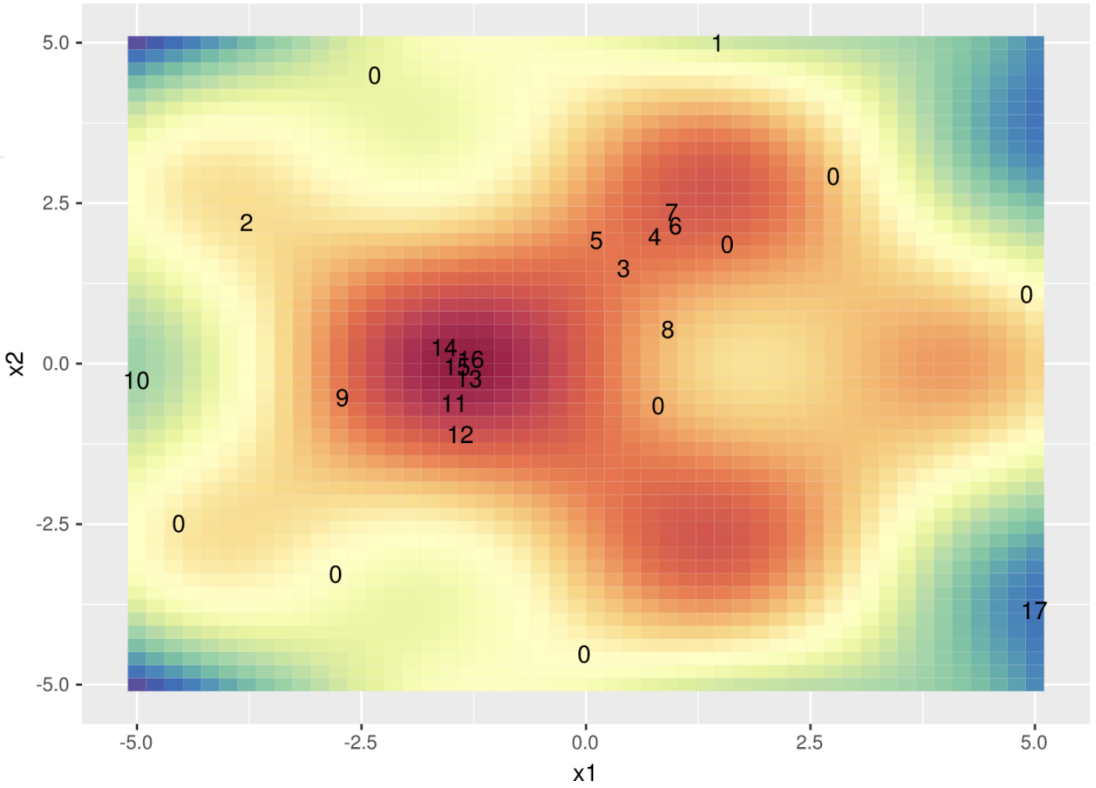
- e.g. minimum leaf size: {5,… 100}
- … maximum depth: {3, …, 10}

generate random combination

learn and evaluate model

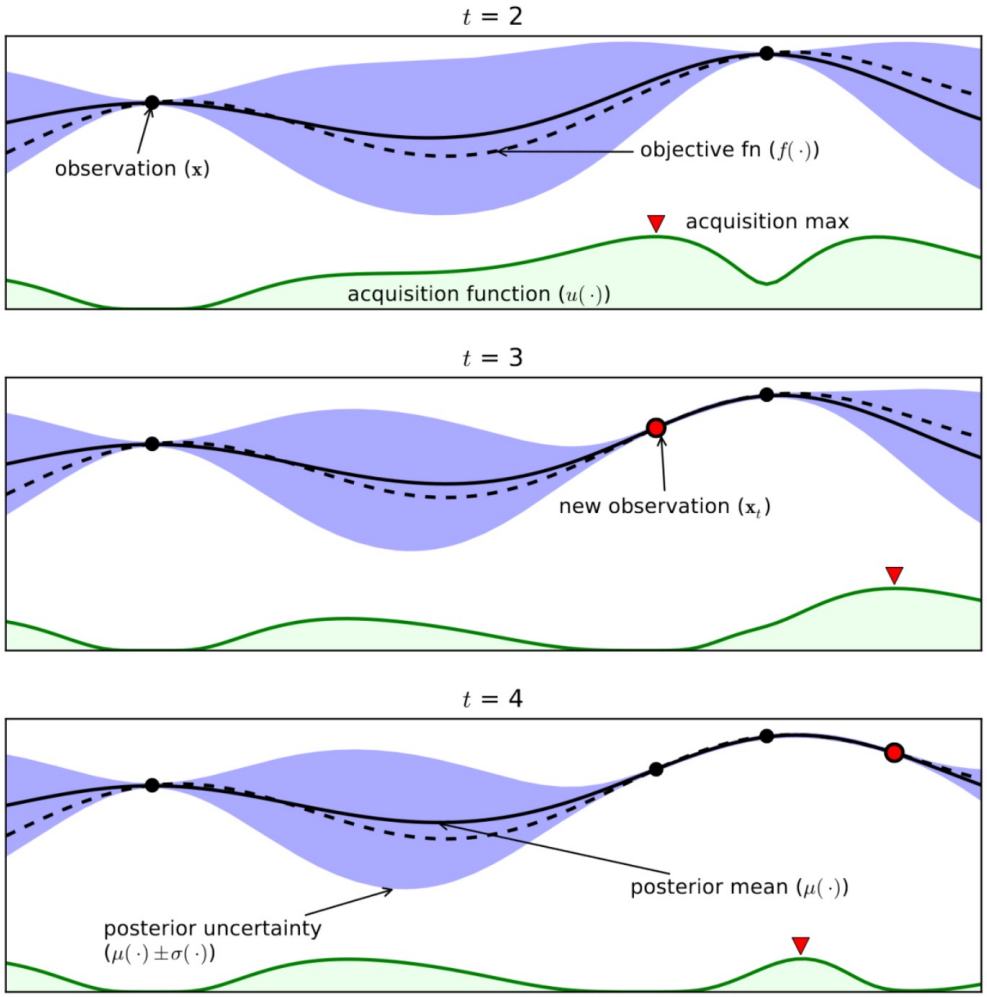
generate next combination and return to previous step until stopping criterion

choose best



https://openml.github.io/openml-tutorial/slides_pdf/AutoML%20Introduction.pdf

guided search: Bayesian optimization



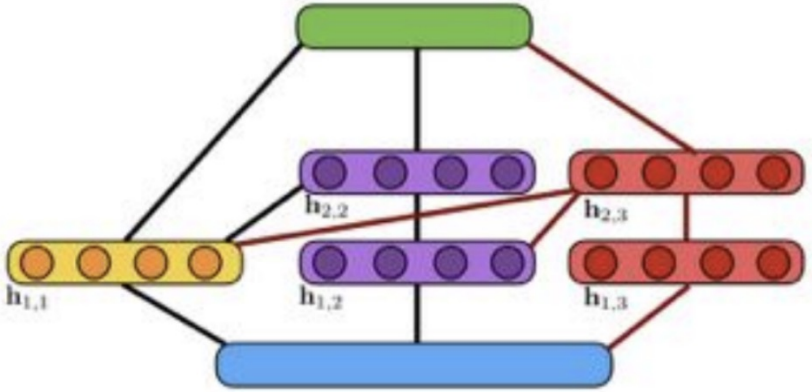
https://openml.github.io/openml-tutorial/slides_pdf/AutoML%20Introduction.pdf

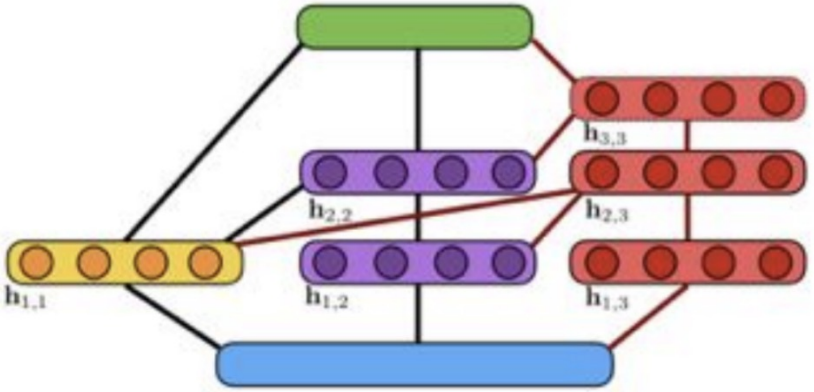
how bad can the problem be? pretty bad...

	Name	Range	Default	log scale	Type	Conditional
Network hyperparameters	batch size	$[32, 4096]$	32	✓	float	-
	number of updates	$[50, 2500]$	200	✓	int	-
	number of layers	$[1, 6]$	1	-	int	-
	learning rate	$[10^{-6}, 1.0]$	10^{-2}	✓	float	-
	L_2 regularization	$[10^{-7}, 10^{-2}]$	10^{-4}	✓	float	-
	dropout output layer	$[0.0, 0.99]$	0.5	✓	float	-
	solver type	{SGD, Momentum, Adam, Adadelta, Adagrad, smorm, Nesterov }	smorm3s	-	cat	-
	lr-policy	{Fixed, Inv, Exp, Step}	fixed	-	cat	-
Conditioned on solver type	β_1	$[10^{-4}, 10^{-1}]$	10^{-1}	✓	float	✓
	β_2	$[10^{-4}, 10^{-1}]$	10^{-1}	✓	float	✓
	ρ	$[0.05, 0.99]$	0.95	✓	float	✓
	momentum	$[0.3, 0.999]$	0.9	✓	float	✓
Conditioned on lr-policy	γ	$[10^{-3}, 10^{-1}]$	10^{-2}	✓	float	✓
	k	$[0.0, 1.0]$	0.5	-	float	✓
	s	$[2, 20]$	2	-	int	✓
Per-layer hyperparameters	activation-type	{Sigmoid, TanH, ScaledTanH, ELU, ReLU, Leaky, Linear}	ReLU	-	cat	✓
	number of units	$[64, 4096]$	128	✓	int	✓
	dropout in layer	$[0.0, 0.99]$	0.5	-	float	✓
	weight initialization	{Constant, Normal, Uniform, Glorot-Uniform, Glorot-Normal, He-Normal, He-Uniform, Orthogonal, Sparse}	He-Normal	-	cat	✓
	std. normal init.	$[10^{-7}, 0.1]$	0.0005	-	float	✓
	leakiness	$[0.01, 0.99]$	$\frac{1}{3}$	-	float	✓
	tanh scale in	$[0.5, 1.0]$	$2/3$	-	float	✓
	tanh scale out	$[1.1, 3.0]$	1.7159	✓	float	✓

https://openml.github.io/openml-tutorial/slides_pdf/AutoML%20Introduction.pdf

neural architecture search





https://openml.github.io/openml-tutorial/slides_pdf/AutoML%20Introduction.pdf

gps

- why do we need automl
- hyperparameter tuning
- neural architecture search
- metalearning
  - general
  - metalearning for pipeline configuration
  - exercise: pipeline search with TPOT

the algorithm selection problem

DT ? RF

how can I use previous experience to help me choose the best algorithm?

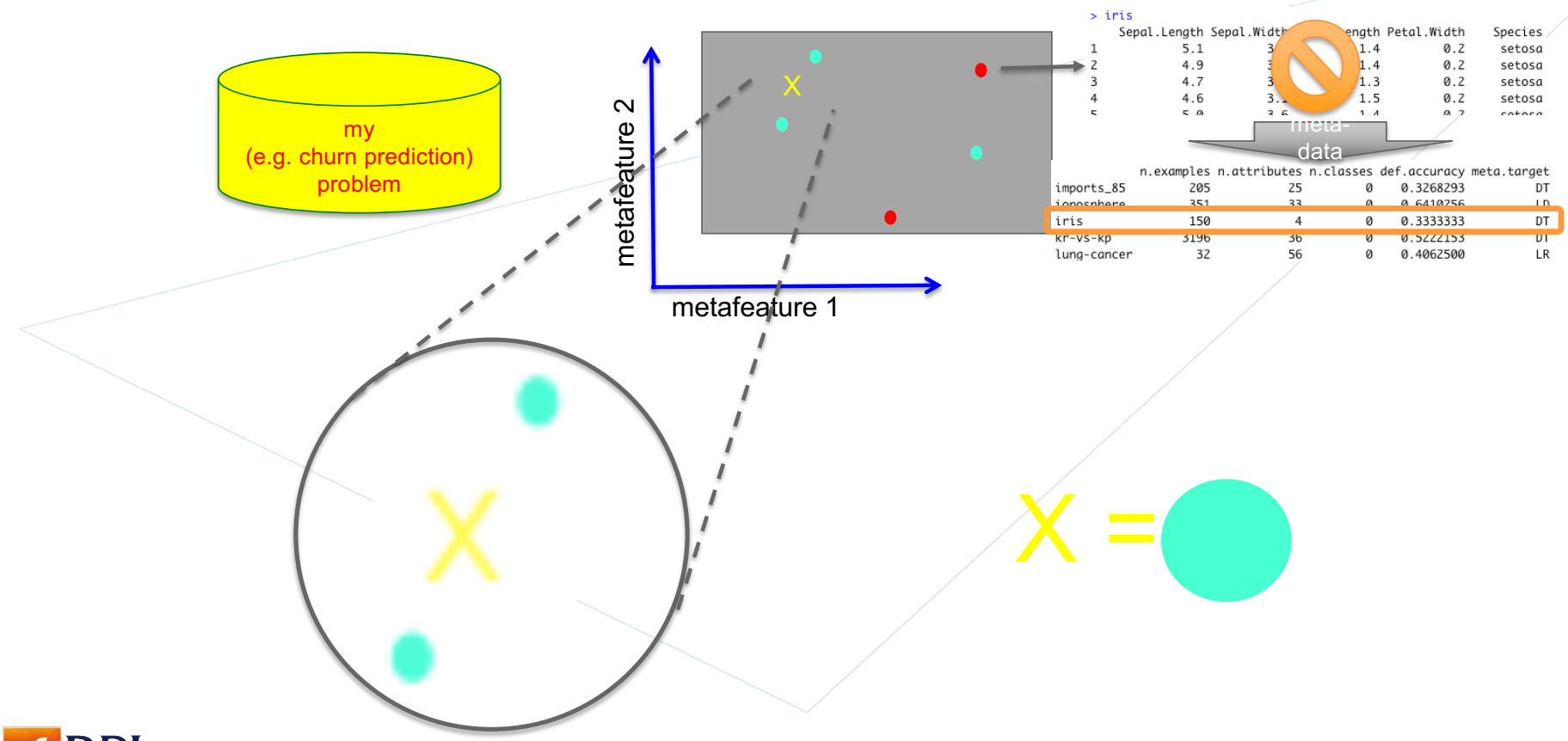
```
> iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
```

DT

```
> house_votes_84
     V1   V2 V3   V4   V5   V6   V7   V8   V9 V10  V11  V12  V13  V14  V15  V16        V17
1     n    y  n    y    y    y    n    n    n   y <NA>    y    y    y    n    y republican
2     n    y  n    y    y    y    n    n    n   n    n    y    y    y    n <NA> republican
3  <NA>    y  y <NA>    y    y    n    n    n   n    y    n    y    y    n    n   democrat
4     n    y  y    n <NA>    y    n    n    n   n    y    n    y    n    n    y   democrat
```

RF

metalearning for algorithm selection



exercise

- optimize pipeline with auto-sklearn
  - notebook: tpot exercise

Wrap-up

automl aims to automate the boring part of data science
standard evaluation does not work with automl
hyperparameter tuning is currently the most important task

algorithm selection/pipeline configuration are under strong development

Make change happen