17-18 July 2019 | Bayer AG R&D, Berlin, Germany

an introduction to machine learning

Marco Chierici
with Margherita Francescatto

Fondazione Bruno Kessler
chierici@fbk.eu

BAYER

FONDAZIONE BRUNO KESSLER | ICT CENTER FOR COMMUNICATION AND INFORMATION TECHNOLOGY

MPBA PREDICTIVE MODELS FOR BIOMEDICINE & ENVIRONMENT

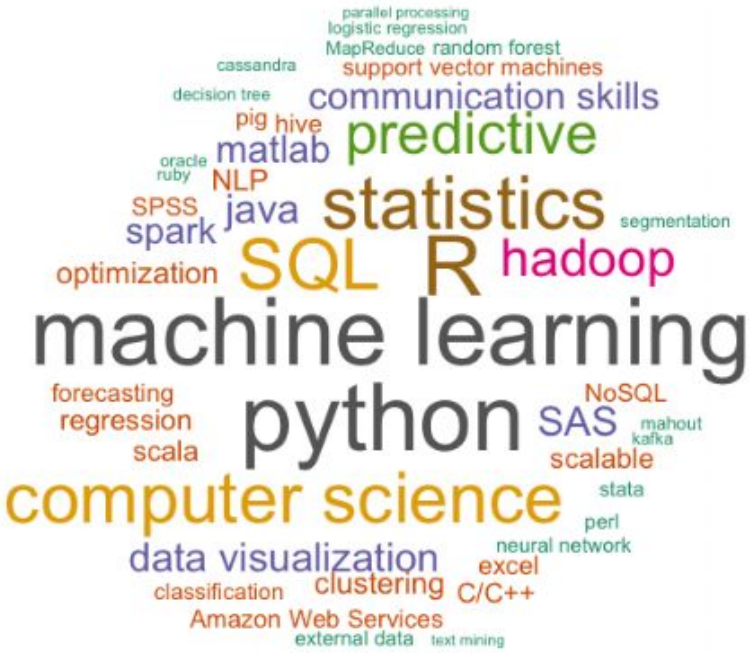
marco chierici

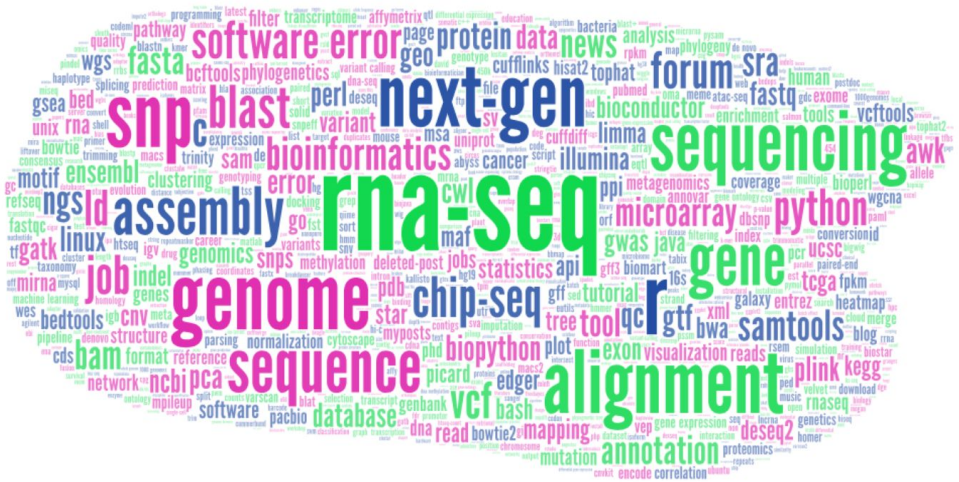
MSc	engineering	2003
PhD	bioengineering	2007
postdoc	bioinformatics	2008
bioinformatician & data scientist		2011-

FONDAZIONE BRUNO KESSLER

chierici@fbk.eu

@MarcoChierici





Fondazione Bruno Kessler
MPBA lab

Trento, Italy

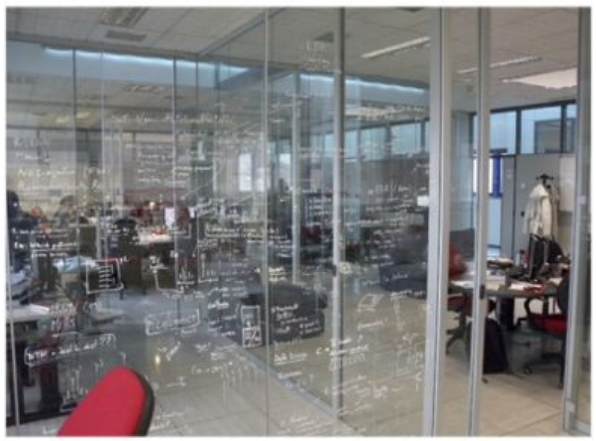
FBK

- >400 researchers
- 7 research units

MPBA lab

- 8 researchers
- 3 PhD students
- 10 MSc students & technologists

https://mpba.fbk.eu - https://mpbalab.fbk.eu



before we start...

how many of you have **heard** about machine learning ?

how many of you **know** about machine learning ?

how many of you **are using** machine learning ?

outline

- General intro: learning & machine learning
- Tools: scikit-learn
- Supervised, unsupervised, reinforcement learning
- Working example: Iris data
- Train and test sets
- k-nearest neighbors (kNN)
- Classification performance
- Cross validation
- Python scikit-learn
- Classical machine learning algorithms

intro:
(machine) learning

***Animal learning**: the alternation of behaviour as a result of individual experience. When an organism can perceive and change its behaviour, it is said to learn.*

Encyclopedia Britannica

bait shyness

eat bait → get sick

↓

~~eat bait~~

[garcia & koelling, 1966]

pigeon superstition

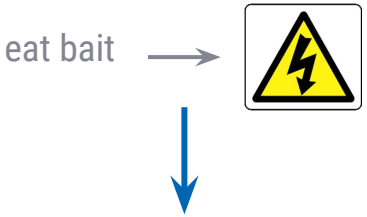
[skinner, 1947]

intro:
(machine) learning

***Animal learning**: the alternation of behaviour as a result of individual experience. When an organism can perceive and change its behaviour, it is said to learn.*

Encyclopedia Britannica

bait shyness revisited



[garcia & koelling, 1966]

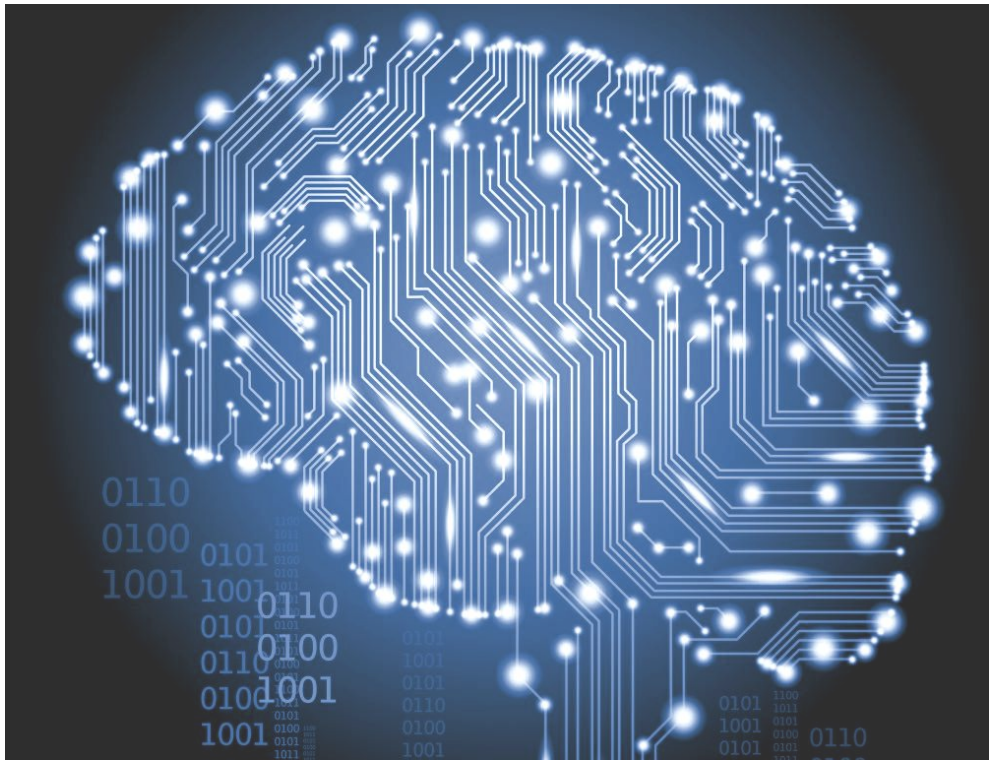
pigeon superstition

[skinner, 1947]

machine learning

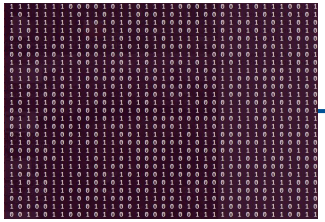
The ability to learn making predictions from training data without being explicitly programmed to perform the task

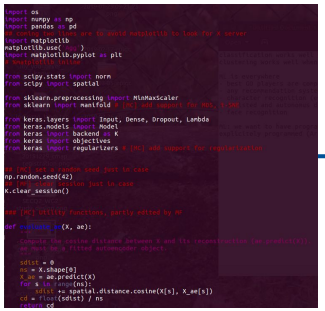
arthur samuel, 1959



traditional vs machine learning programming

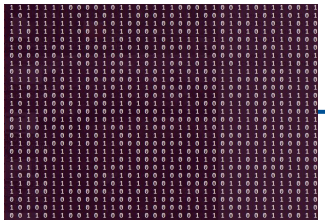
Traditional Programming



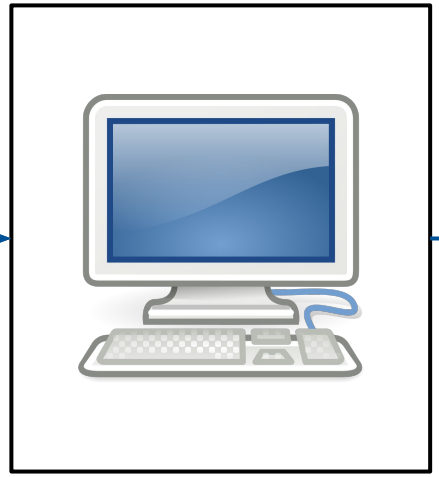


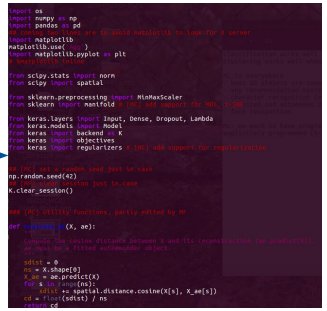
RESULTS

Machine Learning



RESULTS



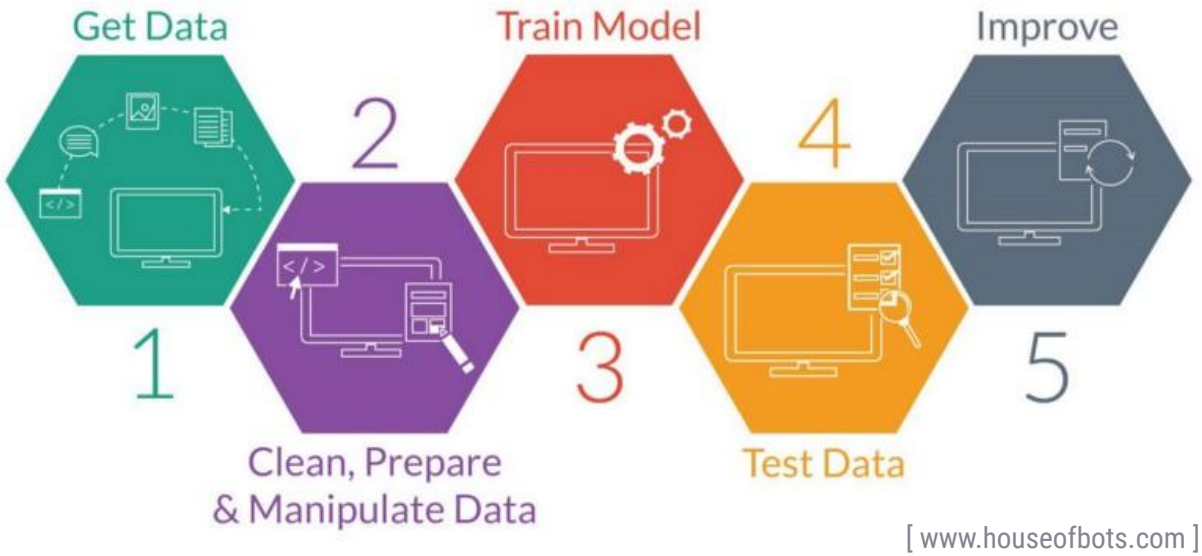


how are things learned?

- Through memorization
  - Memorize as many facts as you can
  - Declarative knowledge, statements of truth
  - Limited by time necessary to observe facts and memory to store them
- Through inference
  - Deduce new information from old knowledge
  - Imperative knowledge
  - Limited by the accuracy of the deduction process: it's essentially a predictive activity and the underlying assumption is that the past can predict the future
- Machine Learning
  - We give data to a program
  - We want the program to infer useful information from implicit patterns in the data
  - Programs able to infer something about the process that generated the data
  - And we want to use this to make predictions on data we haven't seen yet

machine learning
paradigm

- Give the system some observations (training examples)
- The program uses those examples to infer something about the underlying process that gave rise to those observations
- Use what we inferred to make predictions on data that we have never seen before (test examples)



[www.houseofbots.com]

machine learning
the problem setting

A learning problem considers a set of n samples of data with several attributes (features)

Training data

x_tr

and then tries to predict the properties of unknown data

Test data

x_ts

Data can come with or without additional targets (labels) that we want to predict on test data

Training labels

y_tr

reptile example

features | labels

name	egg-laying	scales	poisonous	cold-blooded	# legs	reptile
cobra	True	True	True	True	0	Yes
rattlesnake	True	True	True	True	0	Yes

Our initial model of reptile could be:

- lays eggs
- has scales
- is poisonous
- is cold blooded
- doesn't have legs

More examples needed

reptile example

name	egg-laying	scales	poisonous	cold-blooded	# legs	reptile
cobra	True	True	True	True	0	Yes
rattlesnake	True	True	True	True	0	Yes
boa constrictor	False	True	False	True	0	Yes

Our new example doesn't fit the model

- ~~lays eggs~~
- has scales
- ~~is poisonous~~
- is cold blooded
- doesn't have legs

We can try to refine the model

reptile example

name	egg-laying	scales	poisonous	cold-blooded	# legs	reptile
cobra	True	True	True	True	0	Yes
rattlesnake	True	True	True	True	0	Yes
boa constrictor	False	True	False	True	0	Yes

New model, a reptile:

- has scales
- is cold blooded
- doesn't have legs

More examples needed

reptile example

name	egg-laying	scales	poisonous	cold-blooded	# legs	reptile
cobra	True	True	True	True	0	Yes
rattlesnake	True	True	True	True	0	Yes
boa constrictor	False	True	False	True	0	Yes
alligator	True	True	False	True	4	Yes

Further refined model, a reptile:

- has scales
- is cold blooded

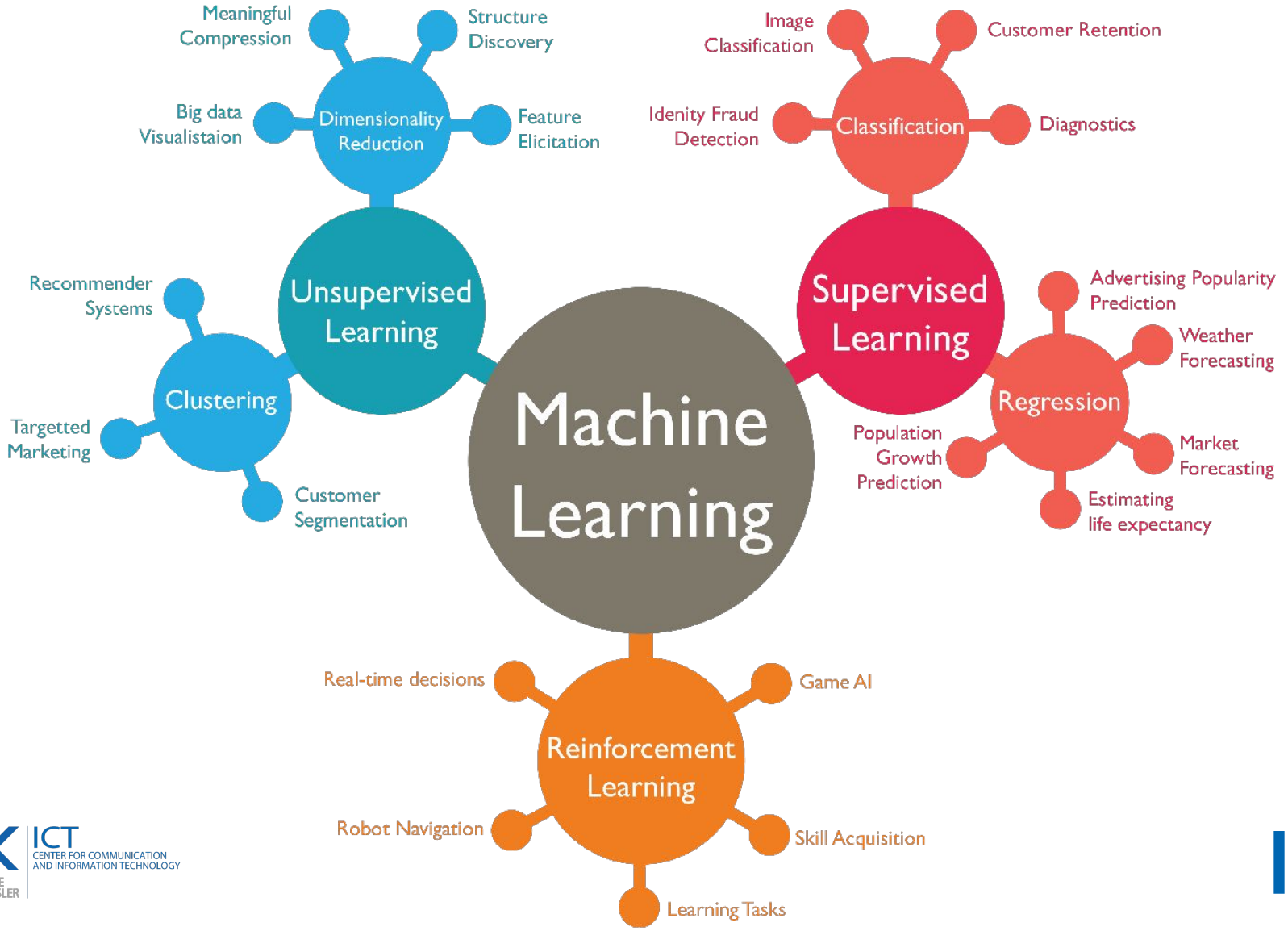
More examples needed

reptile example

name	egg-laying	scales	poisonous	cold-blooded	# legs	reptile
cobra	True	True	True	True	0	Yes
rattlesnake	True	True	True	True	0	Yes
boa constrictor	False	True	False	True	0	Yes
alligator	True	True	False	True	4	Yes
salmon	True	True	False	True	0	No
chicken	Yes	No	False	False	2	No
python	True	True	False	True	0	Yes

- a model doesn't have to be perfect (we accept **false positives**)
- in this case we are happy to see that all reptiles are captured correctly (we have **no false negatives**)

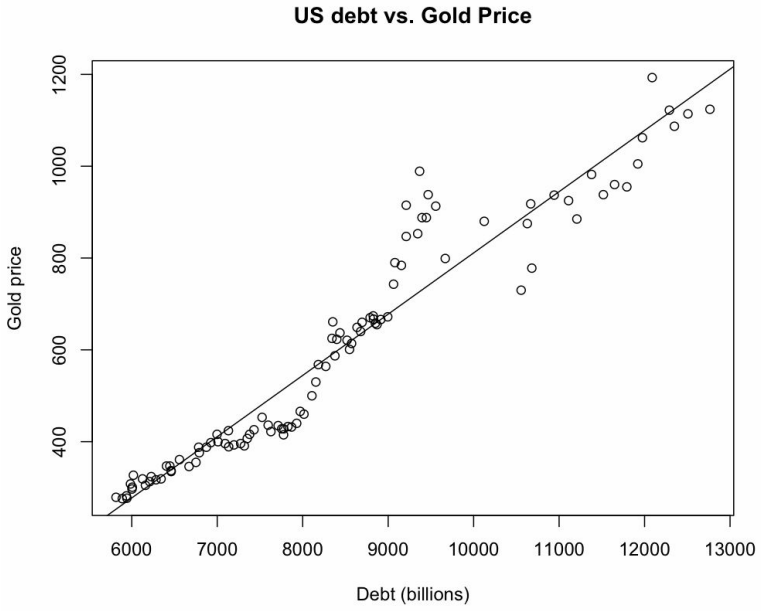
machine learning
classes



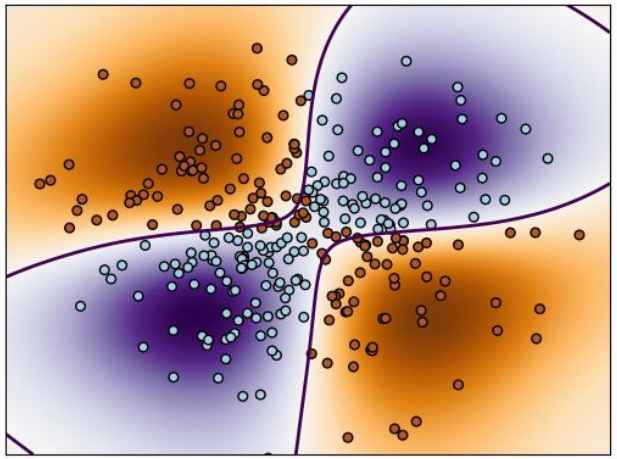
supervised learning

- Data has **target** (labeled input data)
- Given a set of feature/label pairs we want to **find a rule that predicts the label** of unseen input
- Relies on **labeled input data** to learn a function that produces an appropriate output when given new unlabeled data (e.g. regression, classification)

numeric target: regression



categorical target: classification



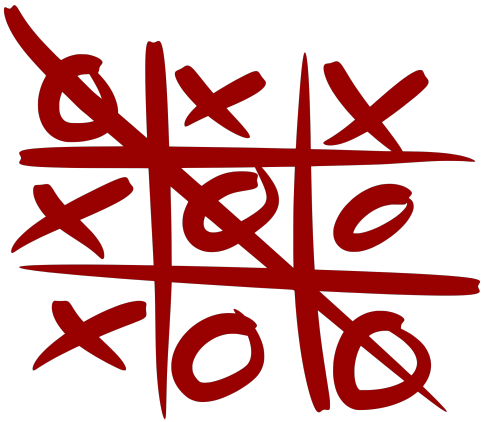
unsupervised learning

- Data has **no target**
- Given a set of observations without labels we want to **group** them i**nto natural clusters** (or find labels for them)
- Examples: clustering (e.g. k-means), dimensionality reduction (e.g. PCA, UMAP), ...

reinforcement learning

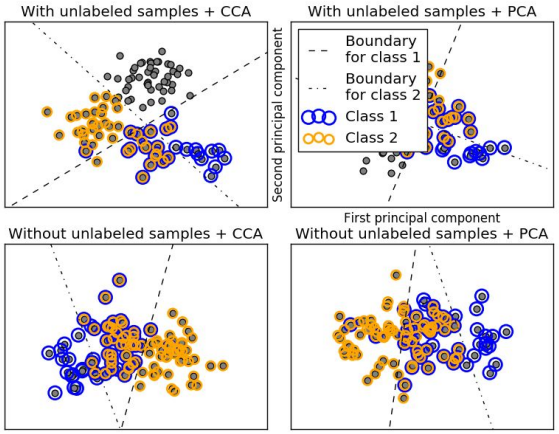
- Data has **partial target**
- How software agents ought to **take actions in an environment** so as to **maximize** some notion of cumulative **reward**

multi-armed bandits

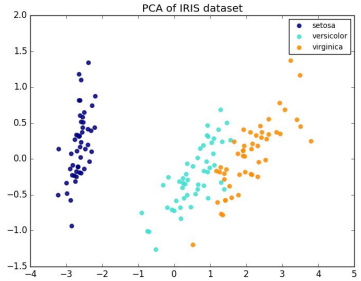


AlphaGo

tools
scikit-learn



- Free machine learning library for **Python**
- Open Source (BSD)
- Simple **fit / predict / transform** methods
- Python / NumPy
- Features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means…



python™

scikit learn

other python tools
that pop-up in the tutorial

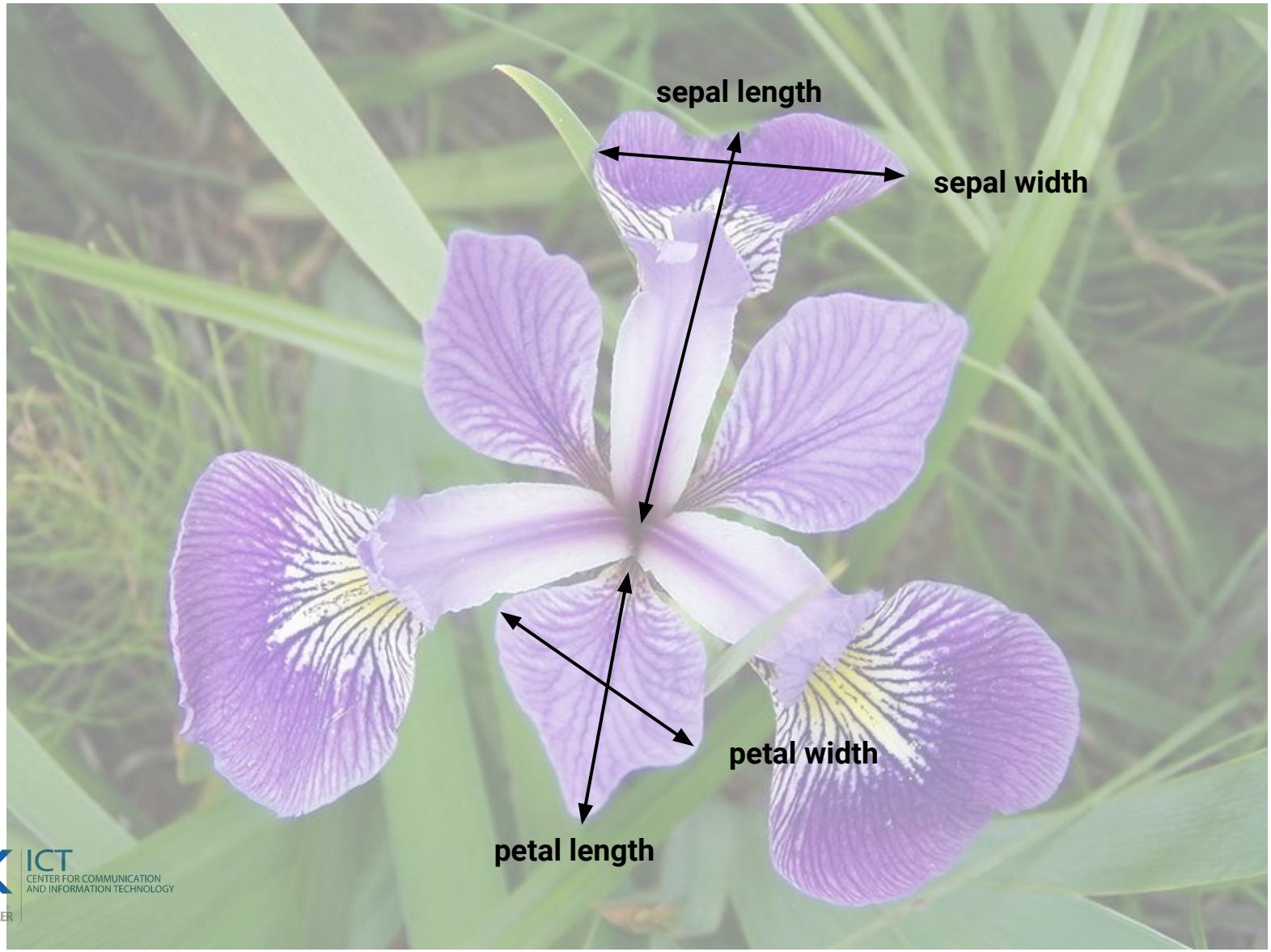
pandas

$$y_i t = \beta' x_{it} + \mu_i + \epsilon_{it}$$

working example
iris flower data set

- Or Fisher's *Iris* data set
- Or Anderson's *Iris* data set
- Describes a set of measurements collected for three distinct Iris species (*Iris setosa*, *Iris virginica* and *Iris versicolor*)
  - Sepal width and length
  - Petal width and length
- Introduced to describe Fisher's linear discriminant model, became a classical machine learning example

petal vs sepal



iris dataset in practice

FEATURES

LABELS

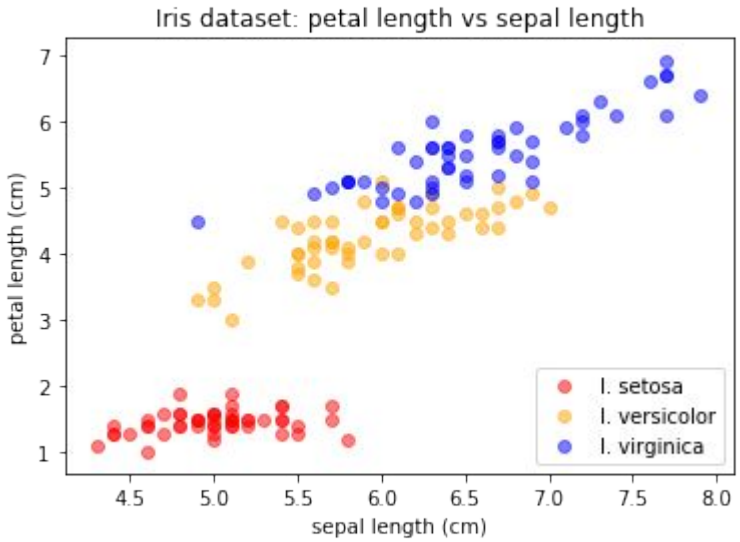
SAMPLES

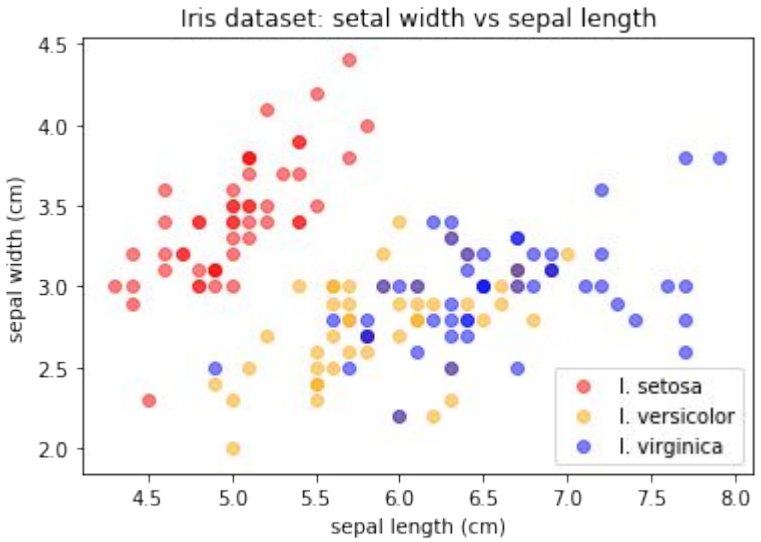
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	*Iris setosa*
4.7	3.2	1.3	0.2	*Iris setosa*
7.0	3.2	4.7	1.4	*Iris versicolor*
6.4	3.2	4.5	1.5	*Iris versicolor*
5.8	2.7	5.1	1.9	*Iris virginica*
7.1	3.0	5.9	2.1	*Iris virginica*
...	...	...	...	...

- 4 **features**: sepal length and width, petal length and width
- 3 **labels**: I. setosa, I. versicolor, I. virginica
- 150 **samples**, 50 for each label (class)

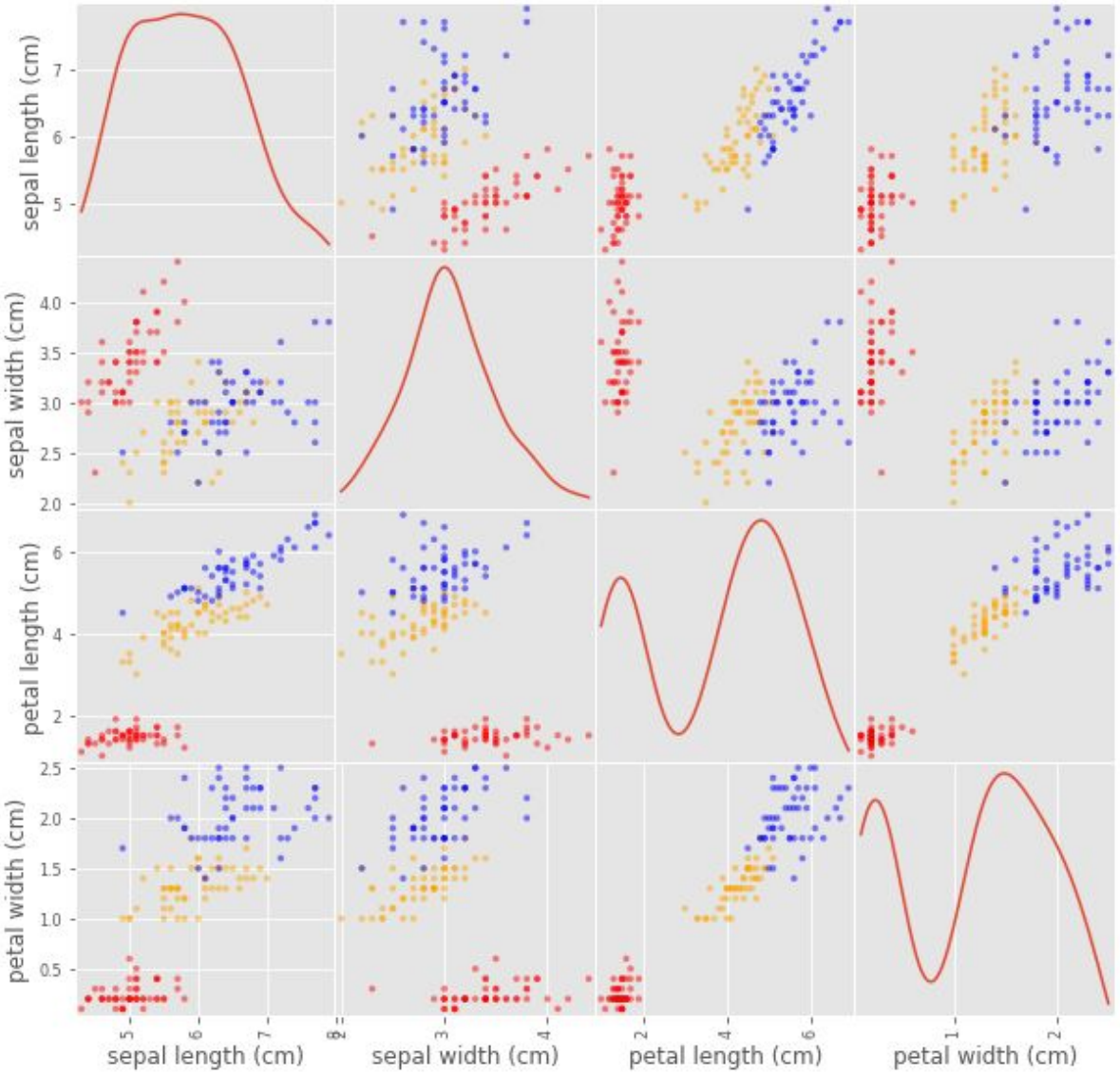
```
from sklearn import datasets
iris = datasets.load_iris()
```

feature relationships



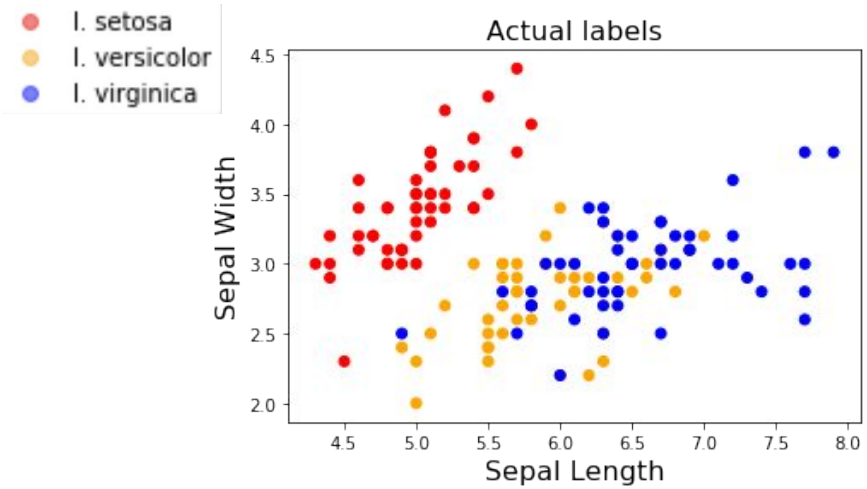


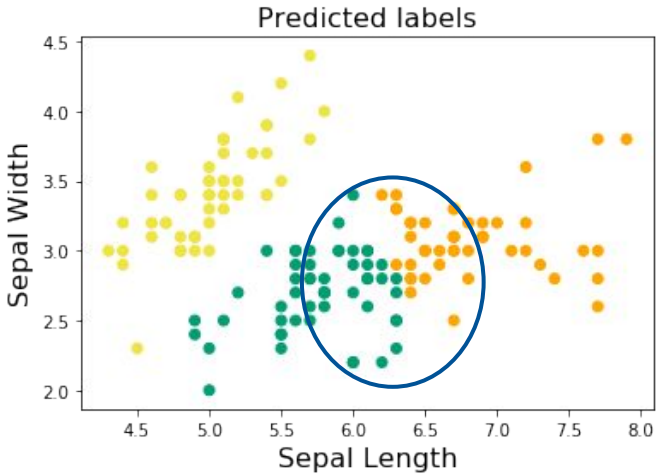
feature relationships



iris unsupervised learning example

- Recap: unsupervised = no labels
- Problem statement: given the iris sepal lengths and widths, assume we know that there are 3 species, but we don't know how to label them
- We want to try splitting the observations into 3 groups (clusters)
- K-means: simplest clustering approach





Separating the two species below is difficult

supervised learning
k-Nearest Neighbors (k-NN)

k-NN

non-parametric method used for classification and regression

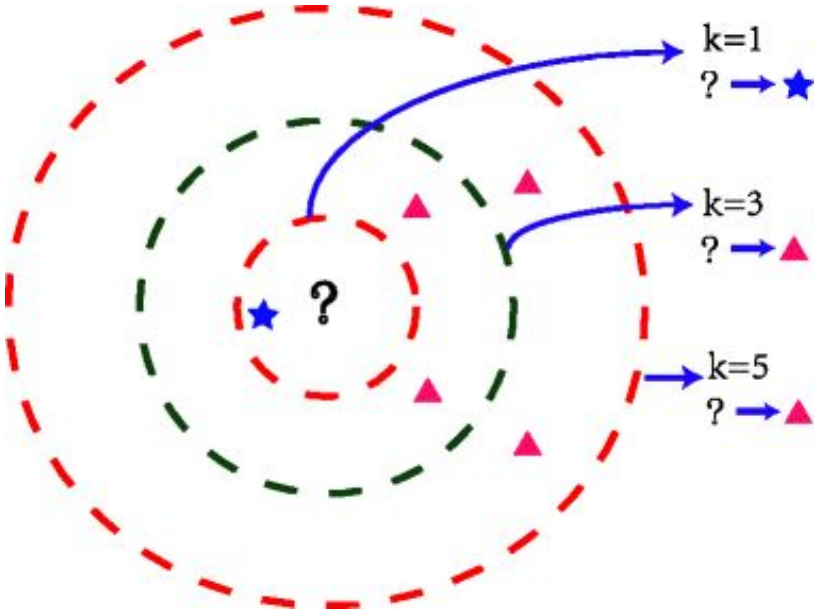
input

the k closest training examples in the feature space

output

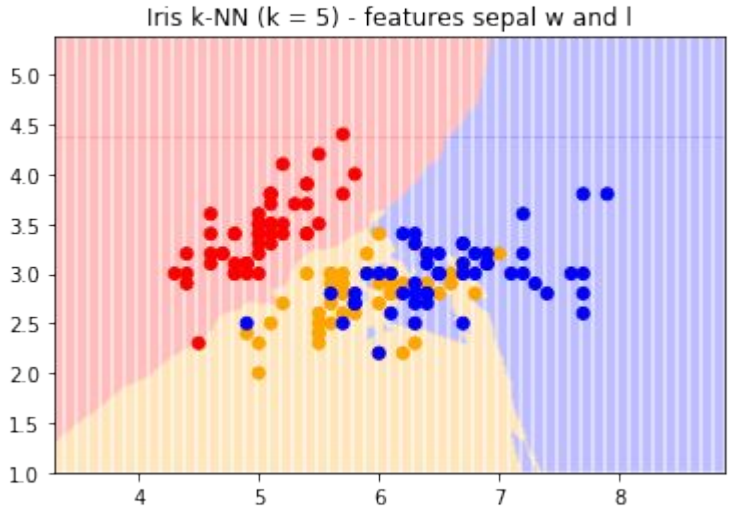
class membership

An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors

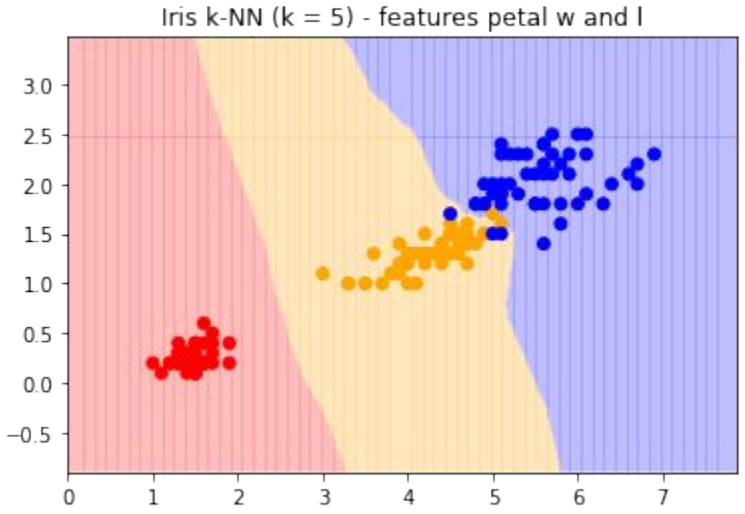


k-NN
in scikit-learn

```
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(n_neighbors=5)
clf.fit(x, y)
y_pred = clf.predict(x_test)
```



decision boundaries one-shot k-NN, k = 5,
features **sepal** width and length



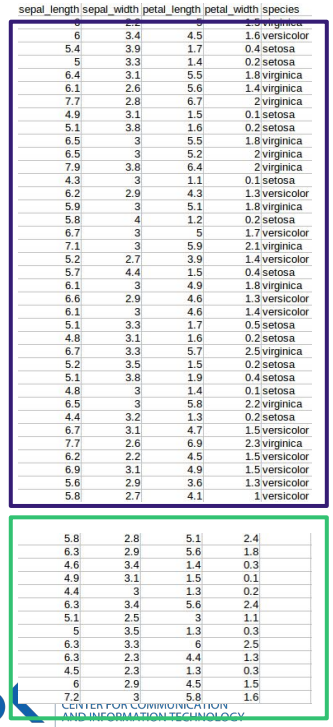
decision boundaries one-shot k-NN, k = 5,
features **petal** width and length

train and test

For supervised classification we need:

labelled data to **train** the model, unlabelled data to **test** the model

⇒ we need to **partition** the Iris dataset into **train and test sets**



sepal_length	sepal_width	petal_length	petal_width	species
6	3.4	4.5	1.6	versicolor
5.4	3.9	1.7	0.4	setosa
5	3.3	1.4	0.2	setosa
6.4	3.1	5.5	1.8	virginica
6.1	2.6	5.6	1.4	virginica
7.7	2.8	6.7	2	virginica
4.9	3.1	1.5	0.1	setosa
5.1	3.8	1.6	0.2	setosa
6.5	3	5.5	1.8	virginica
6.5	3	5.2	2	virginica
7.9	3.8	6.4	2	virginica
4.3	3	1.1	0.1	setosa
6.2	2.9	4.3	1.3	versicolor
5.9	3	5.1	1.8	virginica
5.8	4	1.2	0.2	setosa
6.7	3	5	1.7	versicolor
7.1	3	5.9	2.1	virginica
5.2	2.7	3.9	1.4	versicolor
5.7	4.4	1.5	0.4	setosa
6.1	3	4.9	1.8	virginica
6.6	2.9	4.6	1.3	versicolor
6.1	3	4.6	1.4	versicolor
5.1	3.3	1.7	0.5	setosa
4.8	3.1	1.6	0.2	setosa
6.7	3.3	5.7	2.5	virginica
5.2	3.5	1.5	0.2	setosa
5.1	3.8	1.9	0.4	setosa
4.8	3	1.4	0.1	setosa
6.5	3	5.8	2.2	virginica
4.4	3.2	1.3	0.2	setosa
6.7	3.1	4.7	1.5	versicolor
7.7	2.6	6.9	2.3	virginica
6.2	2.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
5.6	2.9	3.6	1.3	versicolor
5.8	2.7	4.1	1	versicolor

Train data

5.8	2.8	5.1	2.4
6.3	2.9	5.6	1.8
4.6	3.4	1.4	0.3
4.9	3.1	1.5	0.1
4.4	3	1.3	0.2
6.3	3.4	5.6	2.4
5.1	2.5	3	1.1
5	3.5	1.3	0.3
6.3	3.3	6	2.5
6.3	2.3	4.4	1.3
4.5	2.3	1.3	0.3
6	2.9	4.5	1.5
7.2	3	5.8	1.6

Test data

- The partition must be fair
  - Few samples for training ⇒ difficult to capture the complexity of the data
  - One classical partitioning is 80/20 ⇒ 80% of the samples used for training, 20% for test
- Homogeneity between train and test ⇒ balancing
  - E.g. wouldn't be good to have most versicolor and setosa in the train set and most virginica in the test set

classifier evaluation

We need **measures** helping us evaluate **how well we learned**

In a classification problem, we can evaluate this considering the **confusion matrix**, where:

- TP = True Positive
- FP = False Positive
- FN = False Negative
- TN = True Negative

Prediction \ Reality	Class 0	Class 1
Class 0	TN	FN
Class 1	FP	TP

classification metrics

Reality

Prediction

	Class 0	Class 1
Class 0	TN	FN
Class 1	FP	TP

- Sensitivity
  - TP/P = TP/(TP + FN)
- Specificity
  - TN/N = TN/(FP + TN)
- Accuracy
  - (TP + TN)/(P + N)
- Matthew's Correlation Coefficient (MCC)
  - $$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

MCC properties

- Range [-1,+1]
- MCC = 1 → perfect classification
- MCC = 0 → random prediction
- MCC = -1 → perfect misclassification
- Can be generalized to more than 2 classes

classification metrics

Prediction \ Reality	Class 0	Class 1
Class 0	TN	FN
Class 1	FP	TP

Matthew's Correlation Coefficient (MCC)

$$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

MCC is good for unbalanced classes

Example: 15 samples of class 1, 5 samples of class 0.
Suppose the smaller class gets almost entirely misclassified:

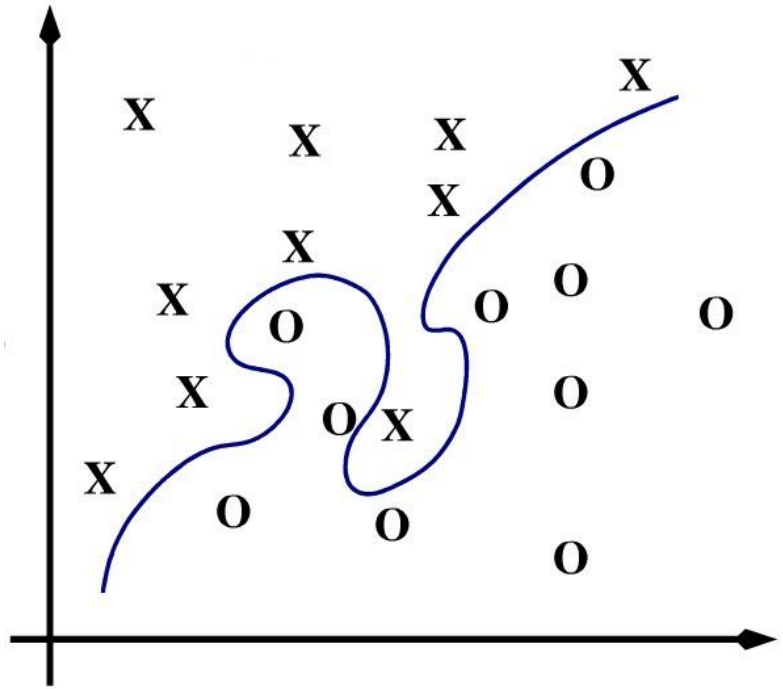
True	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
Pred	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1

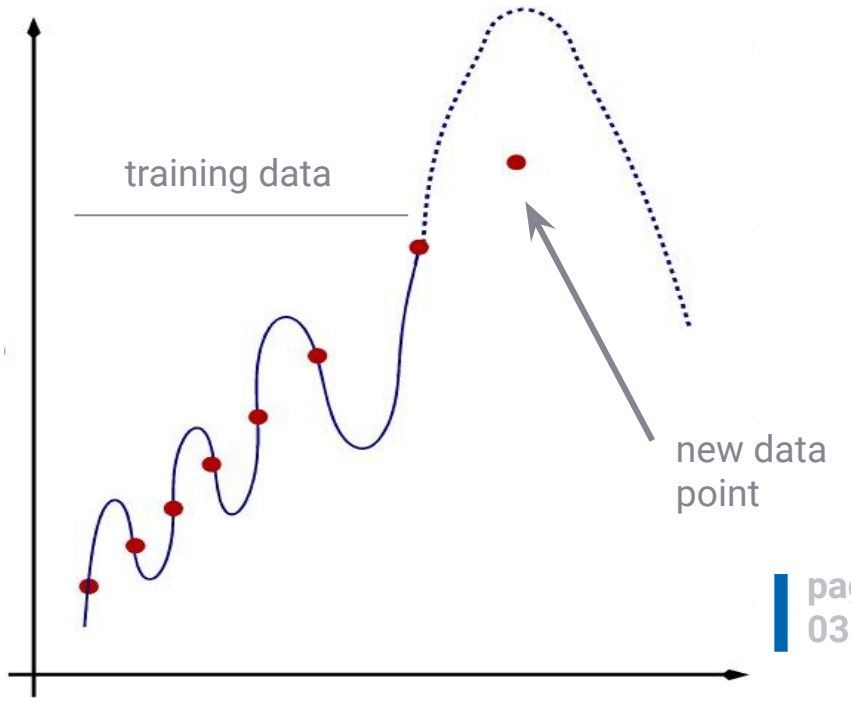
Accuracy = 0.75 (seems good!)

MCC = 0.19 (ouch!)

caveat
training error

Minimizing the training error is prone to overfitting and it results in poor predictive performance





bias/variance dilemma

Components of the training error

bias

error from simplifying assumptions in the model

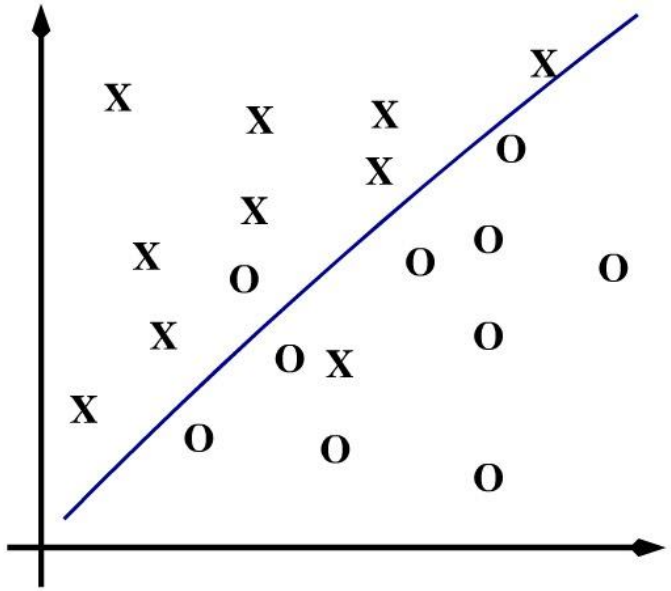
variance

sensitivity to small fluctuations in the training data

[noise]

irreducible error: cannot be minimized

The optimal solution should be a **compromise** between complexity of the solution and error on the training set



a good model doesn't have to be perfect

beware of models that perfectly classify the input data - unless they generalize well on different data

metrics
in scikit-learn /1

```
from sklearn import metrics
```

Classification metrics

See the Classification metrics section of the user guide for further details.

metrics.accuracy_score (y_true, y_pred[, ...])	Accuracy classification score.
metrics.auc (x, y[, reorder])	Compute Area Under the Curve (AUC) using the trapezoidal rule
metrics.average_precision_score (y_true, y_score)	Compute average precision (AP) from prediction scores
metrics.brier_score_loss (y_true, y_prob[, ...])	Compute the Brier score.
metrics.classification_report (y_true, y_pred)	Build a text report showing the main classification metrics
metrics.confusion_matrix (y_true, y_pred[, ...])	Compute confusion matrix to evaluate the accuracy of a classification
metrics.f1_score (y_true, y_pred[, labels, ...])	Compute the F1 score, also known as balanced F-score or F-measure
metrics.fbeta_score (y_true, y_pred, beta[, ...])	Compute the F-beta score
metrics.hamming_loss (y_true, y_pred[, classes])	Compute the average Hamming loss.
metrics.hinge_loss (y_true, pred_decision[, ...])	Average hinge loss (non-regularized)
metrics.jaccard_similarity_score (y_true, y_pred)	Jaccard similarity coefficient score
metrics.log_loss (y_true, y_pred[, eps, ...])	Log loss, aka logistic loss or cross-entropy loss.
metrics.matthews_corrcoef (y_true, y_pred)	Compute the Matthews correlation coefficient (MCC) for binary classes
metrics.precision_recall_curve (y_true, ...)	Compute precision-recall pairs for different probability thresholds
metrics.precision_recall_fscore_support (...)	Compute precision, recall, F-measure and support for each class
metrics.precision_score (y_true, y_pred[, ...])	Compute the precision
metrics.recall_score (y_true, y_pred[, ...])	Compute the recall
metrics.roc_auc_score (y_true, y_score[, ...])	Compute Area Under the Curve (AUC) from prediction scores
metrics.roc_curve (y_true, y_score[, ...])	Compute Receiver operating characteristic (ROC)
metrics.zero_one_loss (y_true, y_pred[, ...])	Zero-one classification loss.
metrics.brier_score_loss (y_true, y_prob[, ...])	Compute the Brier score.

metrics
in scikit-learn /2

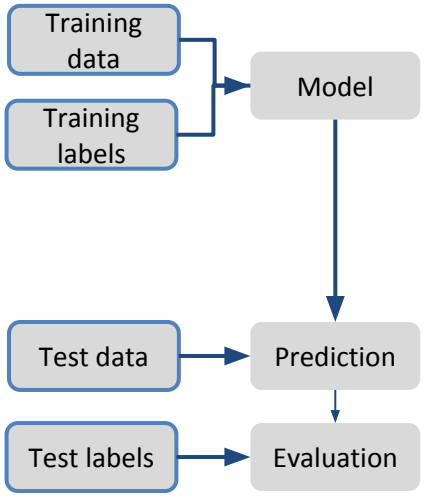
```
from sklearn import metrics

clf = RandomForestClassifier()

clf.fit(x_tr, y_tr)

y_pred = clf.predict(x_ts)

accuracy = metrics.accuracy_score(y_ts, y_pred)
```




```
metrics.accuracy_score(y_ts, y_pred)
```

0.948529411765

```
metrics.matthews_corrcoef(y_ts, y_pred)
```

0.88787329751

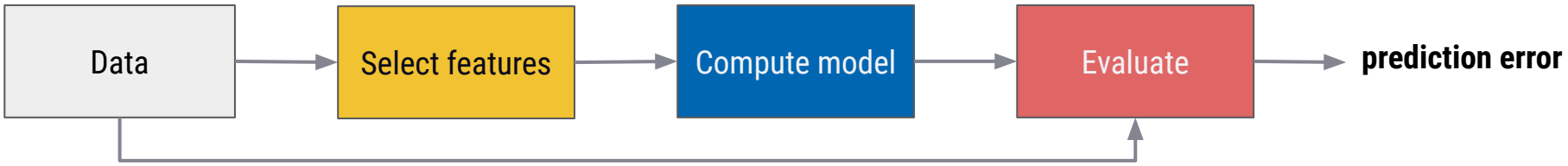
The model performs well!

BUT...

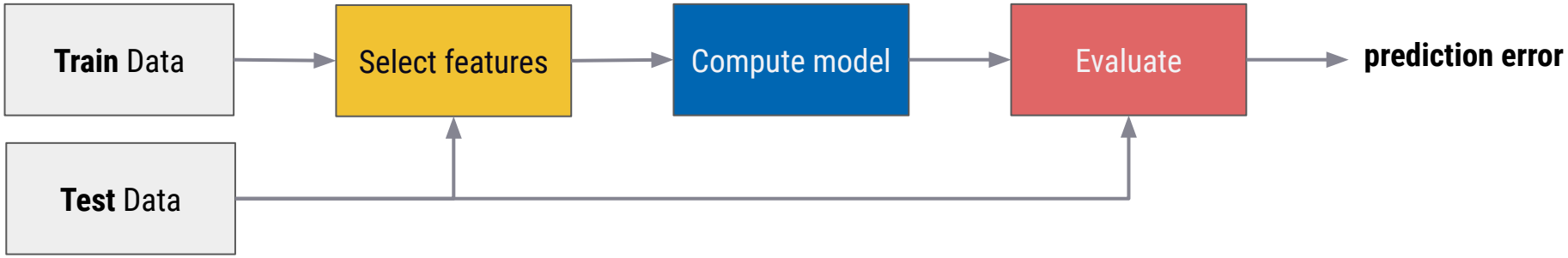
… how this model will generalize to an independent dataset?

model evaluation strategies

OVERFITTING !!



SELECTION BIAS !!



model evaluation
~~strategies~~ mistakes

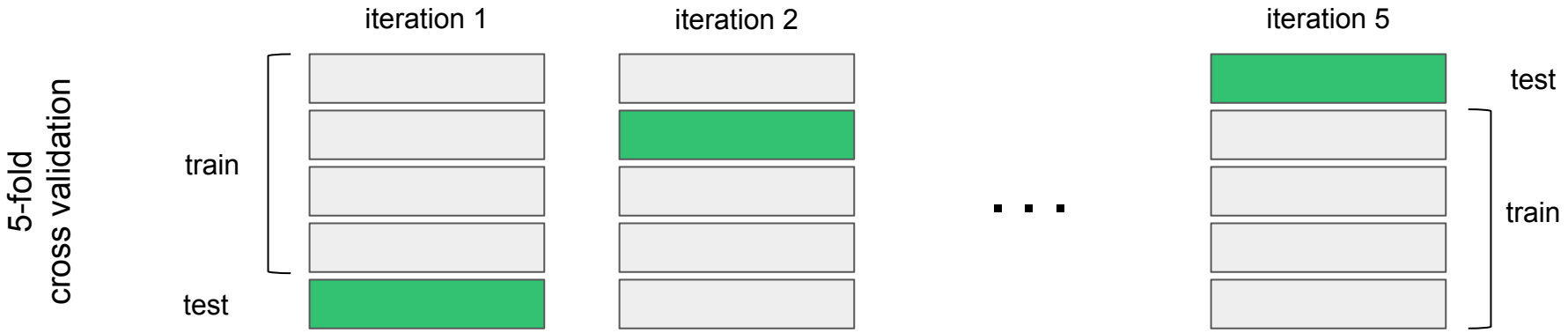
prevent overfitting: [k-fold] cross-validation

01 split data

in k smaller portions

02 use iteratively

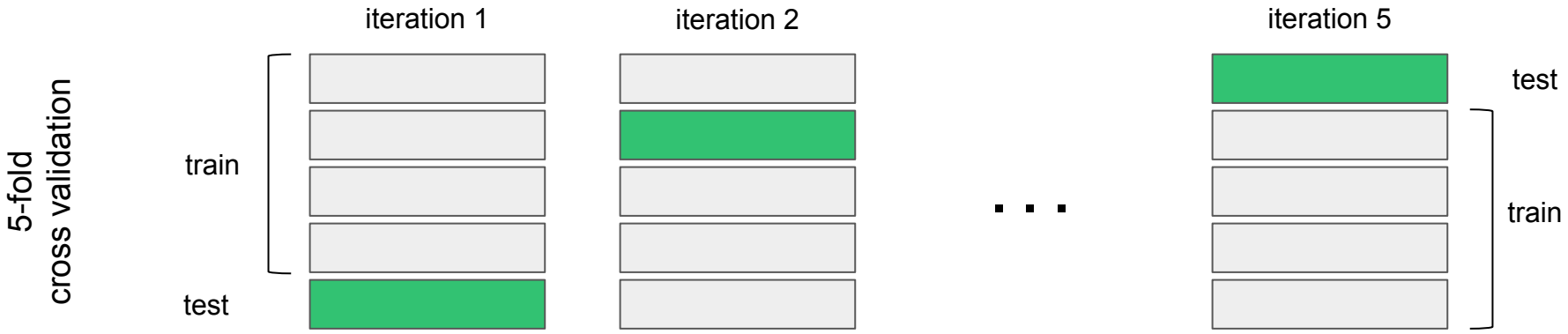
- k-1 folds to **learn** the classification rules
- 1 fold to **evaluate** the classification performance



if the data is stratified, the folds have to reflect this

E.g. in the Iris dataset we want each fold to contain balanced amounts of the three Iris species

k-fold cross-validation in scikit-learn




```
from sklearn import model_selection
skf = model_selection.StratifiedKFold(n_splits=5, shuffle=True, random_state=0)

for idx_tr, idx_ts in skf.split(x_tr, y_tr):
    X_train, Y_train = x_tr[idx_tr], y_tr[idx_tr]
    X_test, Y_test = x_tr[idx_ts], y_tr[idx_ts]
```

need for guidelines!
to ensure unbiased model estimates

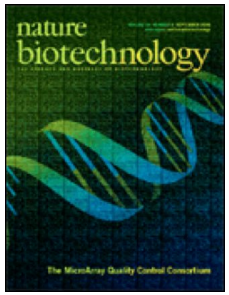
```
                    from sklearn.ensemble import RandomForestClassifier
                    From sklearn import metrics
```


```
y_pred = clf.predict(
                                                                    True, random_state=0)
    for idx_tr, idx_ts in skf:
        X_train, Y_train = x_tr[idx_tr], y_tr[idx_tr]
        X_test, Y_test = x_tr[idx_ts], y_tr[idx_ts]


    metrics.matthews_corrcoef(y_ts, y_pred)
```

Microarray and Sequencing Quality Control

MAQC: FDA-led community-wide effort
AIM: assess technical performance and safety for clinical application of genomics technologies in precision medicine



...

2005-2006

MAQC-I

Differentially expressed genes in microarray data

2007-2010

MAQC-II

Predictive models in microarray studies

2008-2014

MAQC-III (SEQC)

Next-Generation Sequencing quality control (RNA-Seq)

2014-ongoing

MAQC-IV (SEQC2)

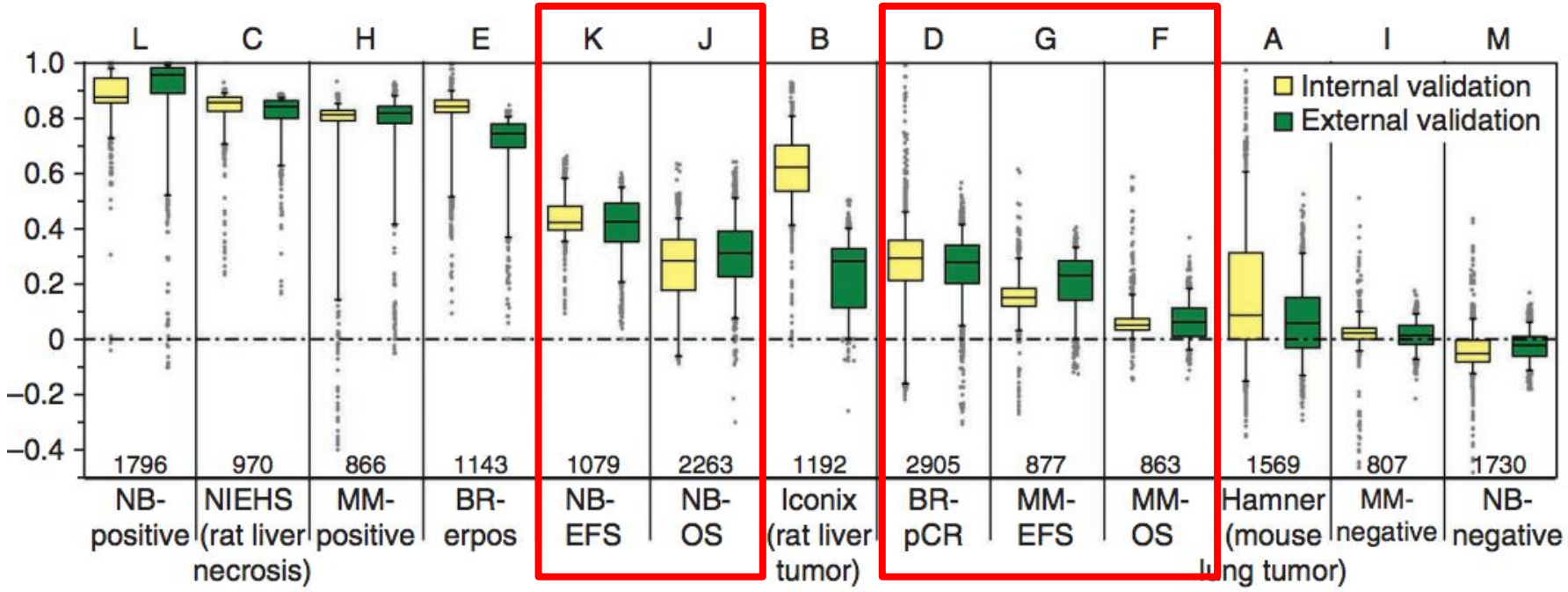
Reproducible genomics: use of DNA-Seq to enhance research in **precision medicine**

> 10 years
> 100 organizations
> 300 participants (FBK since 2007)
IMPACT: 33 papers

prediction performance
is endpoint/dataset dependent

>30,000 models

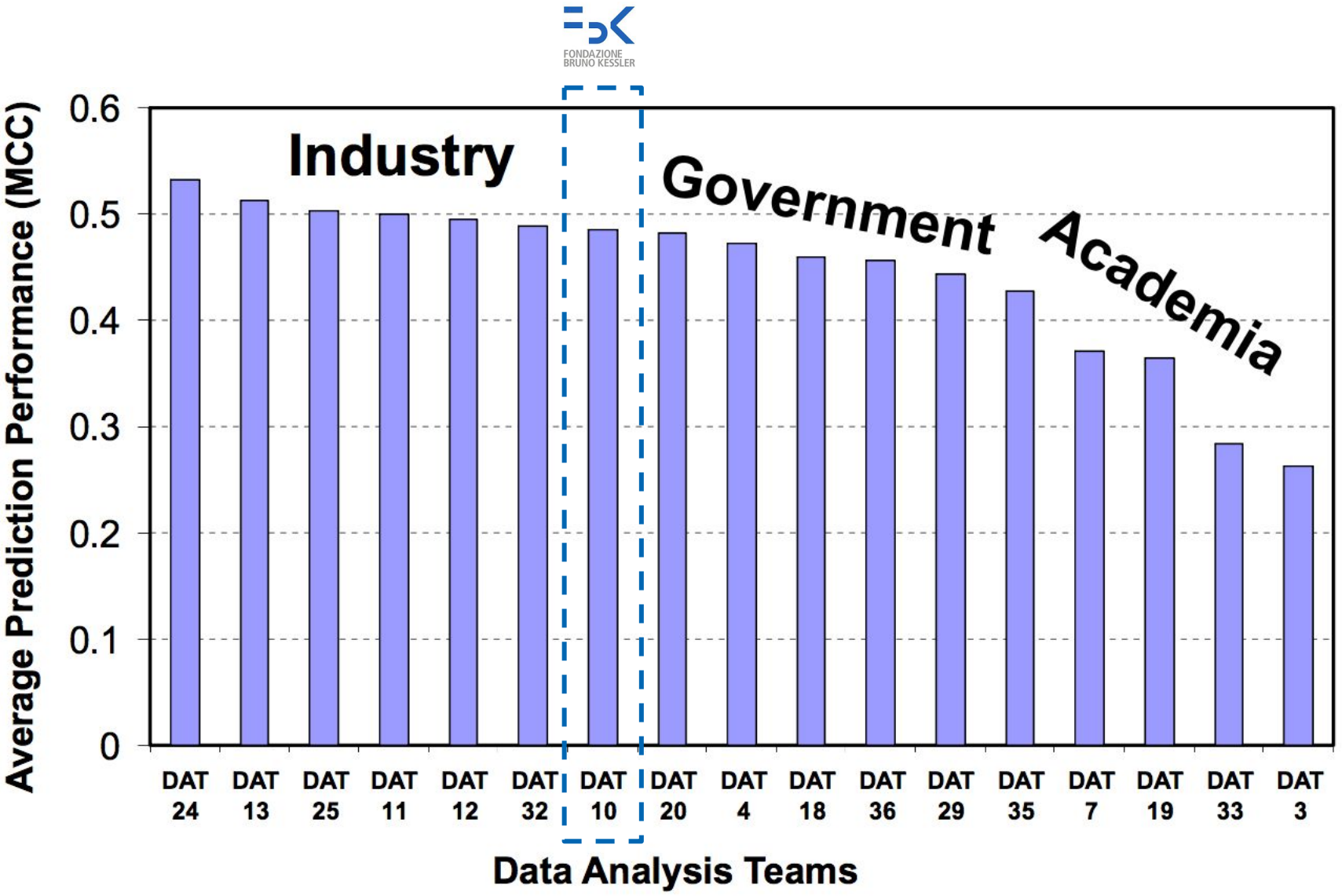
The MAQC Consortium. *Nature Biotechnology*, 2010



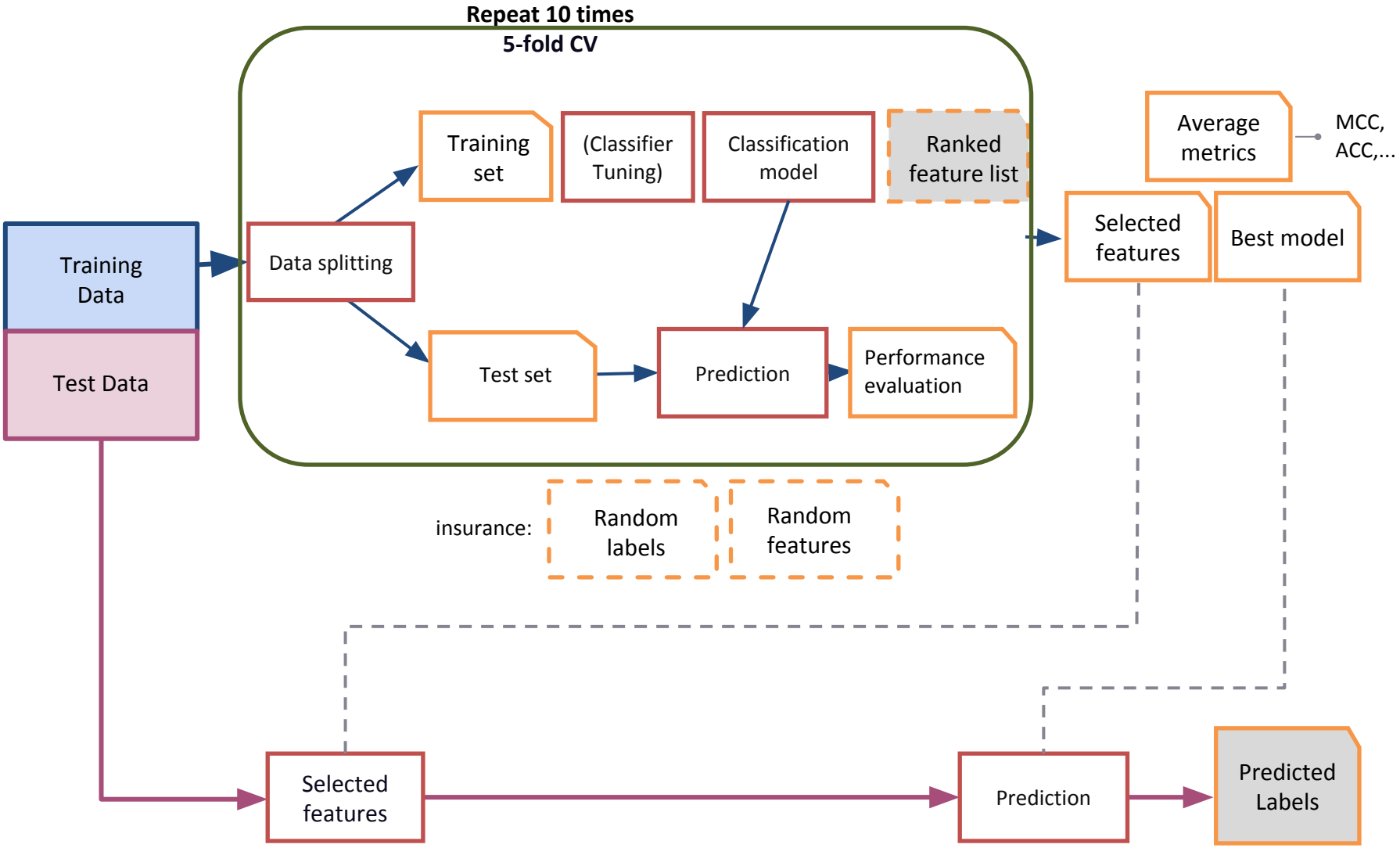
BR: Breast Cancer
MM: Multiple Myeloma
NB: Neuroblastoma

Limited predictability for clinically relevant endpoints based on gene expression alone

different proficiency
among data analysis teams



The MAQC-II/SEQC Data Analysis Plan



recap
take-home messages

learning

- supervised (teacher)
- unsupervised (no teacher)
- reinforcement (learn from experience)

model

- knowledge
- algorithms

training data

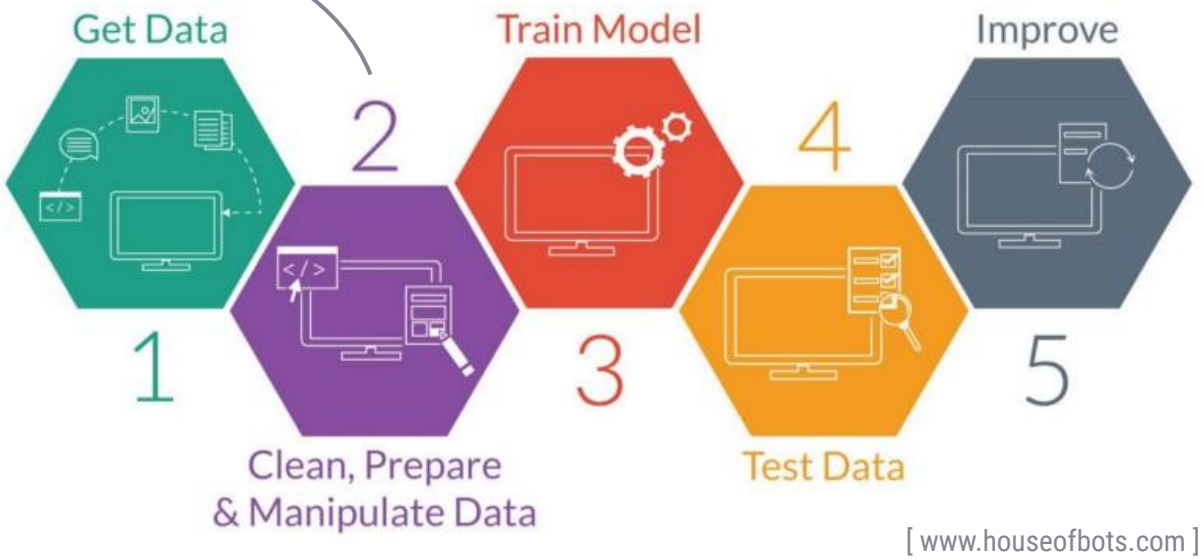
- a set of examples represented by feature vectors
- [optional] target values for each example

generalization

reasonable / unbiased predictions on unseen data

data preprocessing

- Convert raw data into a clean data set
- Transform the features
  - Rescale to an interval: e.g. [-1, 1] or [0, 1]
  - Standardize (mean 0, std 1)



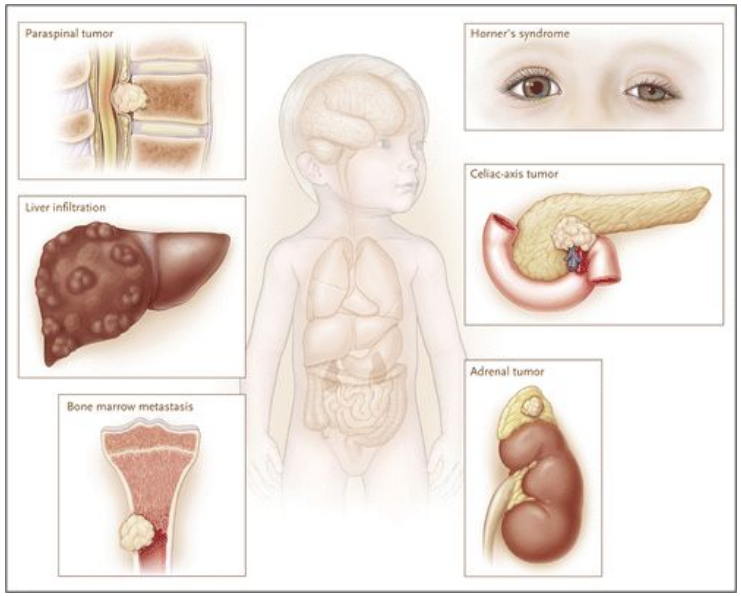
[www.houseofbots.com]

hands-on intro

A first classification task with Python and scikit-learn

- DATA: RNA-seq expression data of neuroblastoma patients
- Neuroblastoma (NB)
  - Pediatric tumor of the sympathetic nervous system
  - Develops from immature neuroblasts
  - It's the most common cancer of childhood
- Dataset
  - 272 samples
- Classification tasks
  - Favorable/unfavorable (CLASS)
  - Gender (SEX)
  - Unknown label (RND)

[Maris, *NEJM*, 2010]



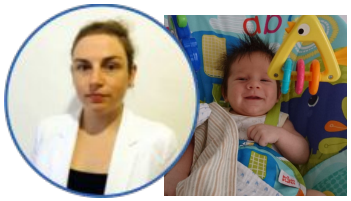
questions?

Photo by Emily Morter on Unsplash

thank you / danke / grazie !

Cesare Furlanello

Head of Data Science
@furlan



Margherita Francescatto

Giuseppe Jurman

FONDAZIONE BRUNO KESSLER | ICT CENTER FOR COMMUNICATION AND INFORMATION TECHNOLOGY