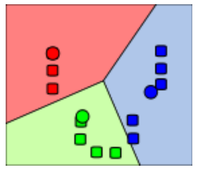
From (chemical) data to information

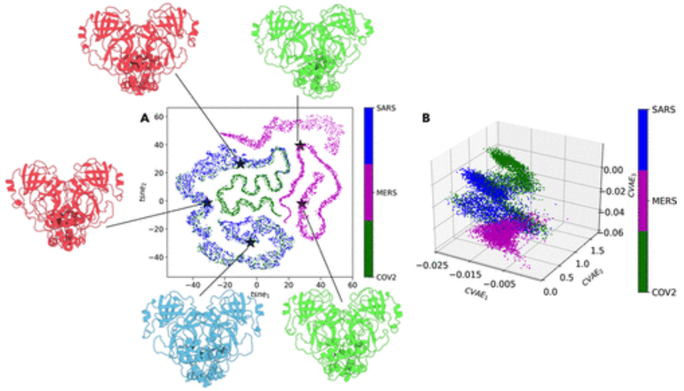
Introduction to Machine Learning Lecture 3

Dr Antonia (Toni) Mey

antonia.mey@ed.ac.uk

Room B23 JBB





Non-linear methods for dimensionality reduction

MATTERS ARISING
https://doi.org/10.1038/s41587-020-00809-z
nature biotechnology
Check for updates

Initialization is critical for preserving global data structure in both *t*-SNE and UMAP

Dmitry Kobak[1] and George C. Linderman[2]

ARISING FROM Becht, E. et al. *Nature Biotechnology* https://doi.org/10.1038/nbt.4314 (2019)

One of the most ubiquitous analysis tools in single-cell transcriptomics and cytometry is *t*-distributed stochastic neighbor embedding (*t*-SNE)[1], which is used to visualize individual cells as points on a two-dimensional scatterplot such that similar cells are positioned close together[2]. A related algorithm, called uniform manifold approximation and projection (UMAP)[3], has attracted substantial attention in the single-cell community[4]. In *Nature Biotechnology*, Becht et al.[4] argued that UMAP is preferable to *t*-SNE because it better preserves the global structure of the data and is more consistent across runs. Here we show that this alleged superiority of UMAP can be entirely attributed to different choices of initialization in the implementations used by Becht et al.: the *t*-SNE implementations by default used random initialization, while the UMAP implementation used a technique called Laplacian eigenmaps (LE)[5] to initialize the embedding. We show that UMAP with random initialization preserves global structure as poorly as *t*-SNE with random initialization, while *t*-SNE with informative initialization performs as well as UMAP with informative initialization. On the basis of these observations, we argue that there is currently no evidence that the UMAP algorithm per se has any advantage over *t*-SNE in terms of preserving global structure. We also contend that these algorithms should always use informative initialization by default.

At the core of both *t*-SNE and UMAP are loss functions that make similar points attract each other and push dissimilar points away from each other. Both algorithms minimize their loss functions by using gradient descent. Gradient descent begins with some initial configuration of points, and with each iteration the points are moved to decrease the loss function. The specific implementations of these algorithms used by Becht et al. differed in how the initial configuration of points was chosen: the *t*-SNE implementations placed the points randomly, whereas the UMAP implementation used LE[5], an algorithm that can often achieve globally accurate embedding on its own[6]. The effect of this difference on how well the two algorithms preserve global structure was not discussed or investigated by Becht et al.

We can illustrate the importance of initialization for both algorithms using a simple toy dataset (Fig. 1). We sampled $n=7{,}000$ points from a circle with some added Gaussian noise and used UMAP and *t*-SNE to construct an embedding. We kept all parameters for both algorithms at their default values and only changed the initialization. The *t*-SNE algorithm with random initialization produced a knot; UMAP with random initialization produced a tangled web with multiple tears. In both cases, the global structure was evidently not preserved; indeed, gradient descent in *t*-SNE and UMAP only pulls close neighbors together and is not much influenced by the global arrangement of points. At the same time, LE recovers the original circle for this toy dataset and, when used for initialization, strongly improves the UMAP result. A method often recommended for initialization in *t*-SNE is principal component analysis (PCA)[2]. Here PCA also recovers the original circle and strongly improves the *t*-SNE result. In both cases, only with informative initialization can UMAP and *t*-SNE produce a faithful representation of the closed one-dimensional manifold—the circle.

Using the code published by Becht et al., we analyzed the separate effects of initialization and algorithm on their results by adding UMAP with random initialization and *t*-SNE (using FIt-SNE[7]) with PCA initialization to the benchmarking comparison. Apart from the initialization, both algorithms were run with the same parameters as in Becht et al. We used all three datasets analyzed in the original publication (sample sizes from 320,000 to 820,000 cells)[8–10]. To quantify preservation of global structure, Becht et al. computed Pearson correlation between pairwise Euclidean distances in high-dimensional space and in the embedding. To quantify the reproducibility of the embedding, the authors embedded random subsamples of the data and measured the correlation of the coordinates of subsample embeddings with the coordinates of the full-dataset embeddings (up to symmetries around the coordinate axes). Our results show that *t*-SNE and UMAP with random initialization perform similarly poorly with regard to both metrics, whereas *t*-SNE and UMAP with PCA and LE initialization, respectively, perform similarly well (Table 1). See Extended Data Figs. 1–6 for the exact analogs of the original figures from Becht et al.

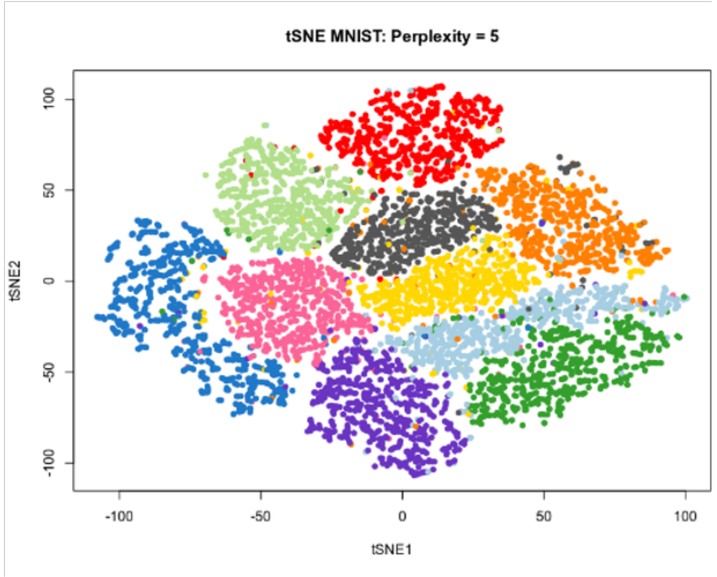
Becht et al. wrote that their findings were "consistent with the idea" that UMAP performs "optimizations that are sensitive to global features of the data, thus reaching similar arrangements more consistently." Our results show that this conclusion can be misleading: their findings were in fact not due to UMAP optimizations but rather to its initialization.

We have recently argued that, for single-cell transcriptomic data, *t*-SNE with PCA initialization produces more meaningful embeddings than *t*-SNE with random initialization[2]. The findings of Becht et al., when interpreted correctly, also underscore the importance of using informative initialization and suggest that it should be used as the default option in *t*-SNE and UMAP implementations. PCA initialization has always been the default in openTSNE[11], a Python reimplementation of FIt-SNE, and FIt-SNE v.1.2 now also uses it by default. OpenTSNE v.0.4 now also supports LE initialization. Importantly, *t*-SNE with non-random initialization should not be considered a new algorithm or even an extension of the original *t*-SNE; it is exactly the same algorithm with the same loss function, and almost any existing implementation trivially allows the use of any given initialization, including the PCA-based one.

[1]Institute for Ophthalmic Research, University of Tübingen, Tübingen, Germany. [2]Applied Mathematics Program, Yale University, New Haven, CT, USA.

- Useful for visualisation, project high-dimensional data in 2 or 3 dimensions.
- Controlled by one main parameters: "perplexity"
- Relative distance between points not quantitatively meaningful

MNIST: database of written digits



https://distill.pub/2016/misread-tsne/

T-distributed Stochastic Neighbour Embedding (t-SNE)

$$X = x_{i=1}^{n}, \quad x_i \in \mathbb{R}^n \longrightarrow Y = y_{i=1}^{n}, \quad y_i \in \mathbb{R}^m$$

1. Modelling high-dimensional space

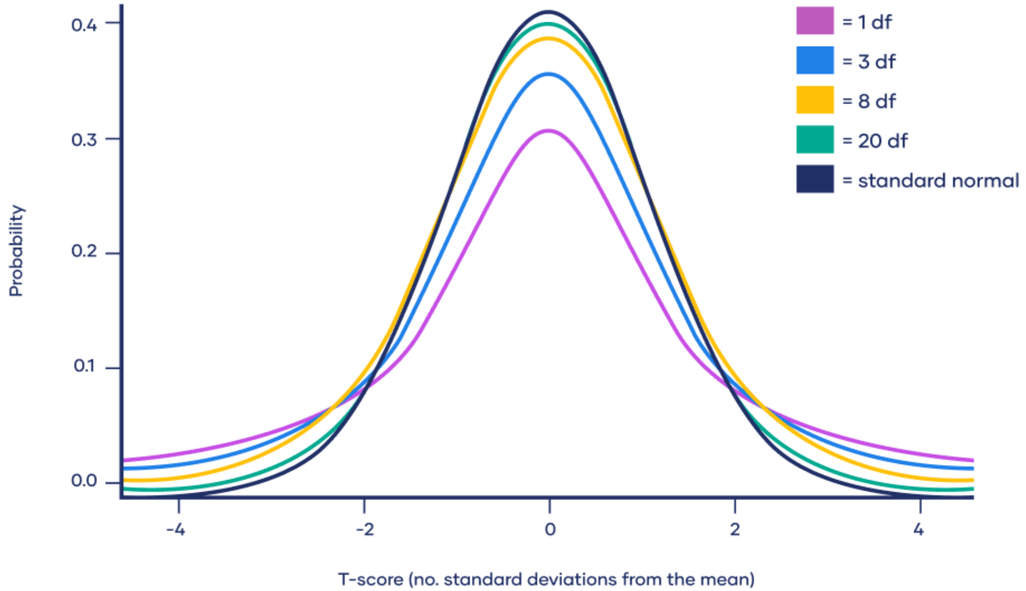
$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/2\sigma_i^2)} \qquad p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

Perplexity: User-defined parameter that influences determination of σ.

- Low Perplexity (σ_i^2 small): Focuses on local structures
- High Perplexity (σ_i^2 large): Incorporates more distant points

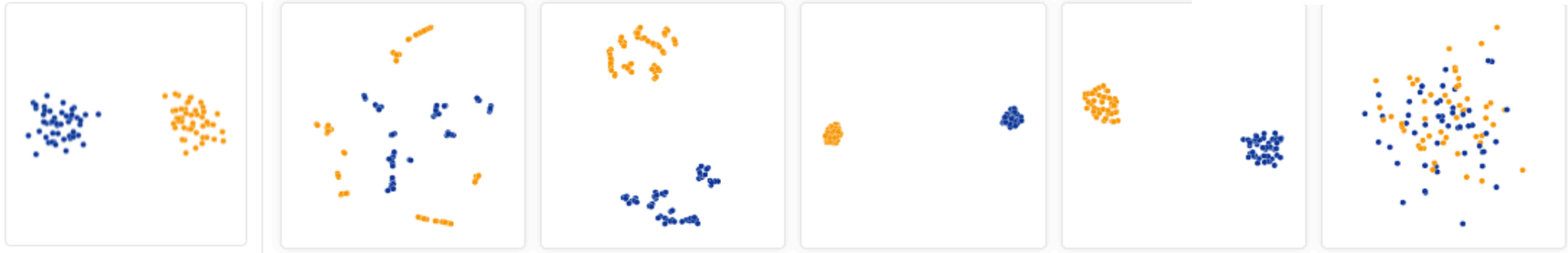
2. Modelling low-dimensional space

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k\neq l}(1 + \|y_k - y_l\|^2)^{-1}}$$



3. Minimize KL-divergence between P and Q

$$C = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$



Original | Perplexity: 2 Step: 5,000 | Perplexity: 5 Step: 5,000 | Perplexity: 30 Step: 5,000 | Perplexity: 50 Step: 5,000 | Perplexity: 100 Step: 5,000

Uniform manifold approximation and projection (UMAP)

UMAP constructs a high-dimensional graph representation of the data and then optimizes a low-dimensional graph to be as structurally similar as possible.

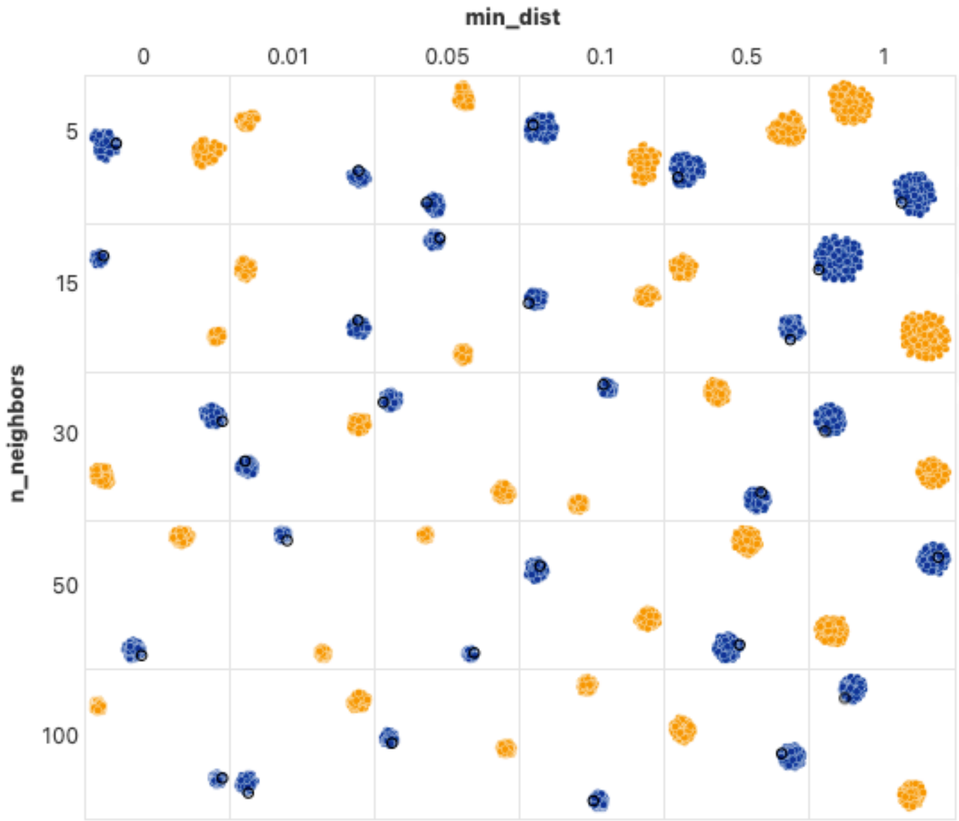
1. Construct a weighted graph in high-dimensional space

$$w_{ij} = \begin{cases} \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma}\right) & \text{if } x_j \in \text{kNN}(x_i) \\ 0 & \text{otherwise} \end{cases}$$

2. Optimize to approximate the topological structure of the high-dimensional graph

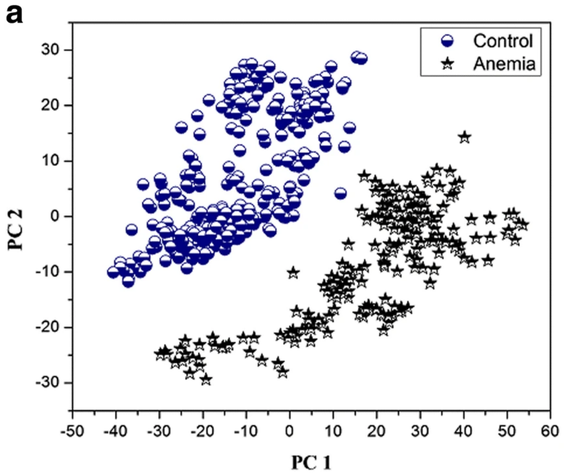
$$C = \sum_{i,j} \left[w_{ij} \log\left(\frac{w_{ij}}{v_{ij}}\right) + (1 - w_{ij}) \log\left(\frac{1 - w_{ij}}{1 - v_{ij}}\right) \right]$$

- *n_neighbors:* local vs. global structure
- *min_distance*: how close points are allowed
- *metric*: quantify the similarity

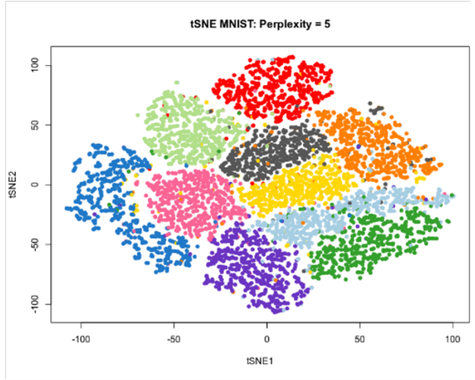


Two clusters with equal numbers of points.
Points Per Cluster: 100; **Dimensions:** 50

Take home from dimensionality reduction



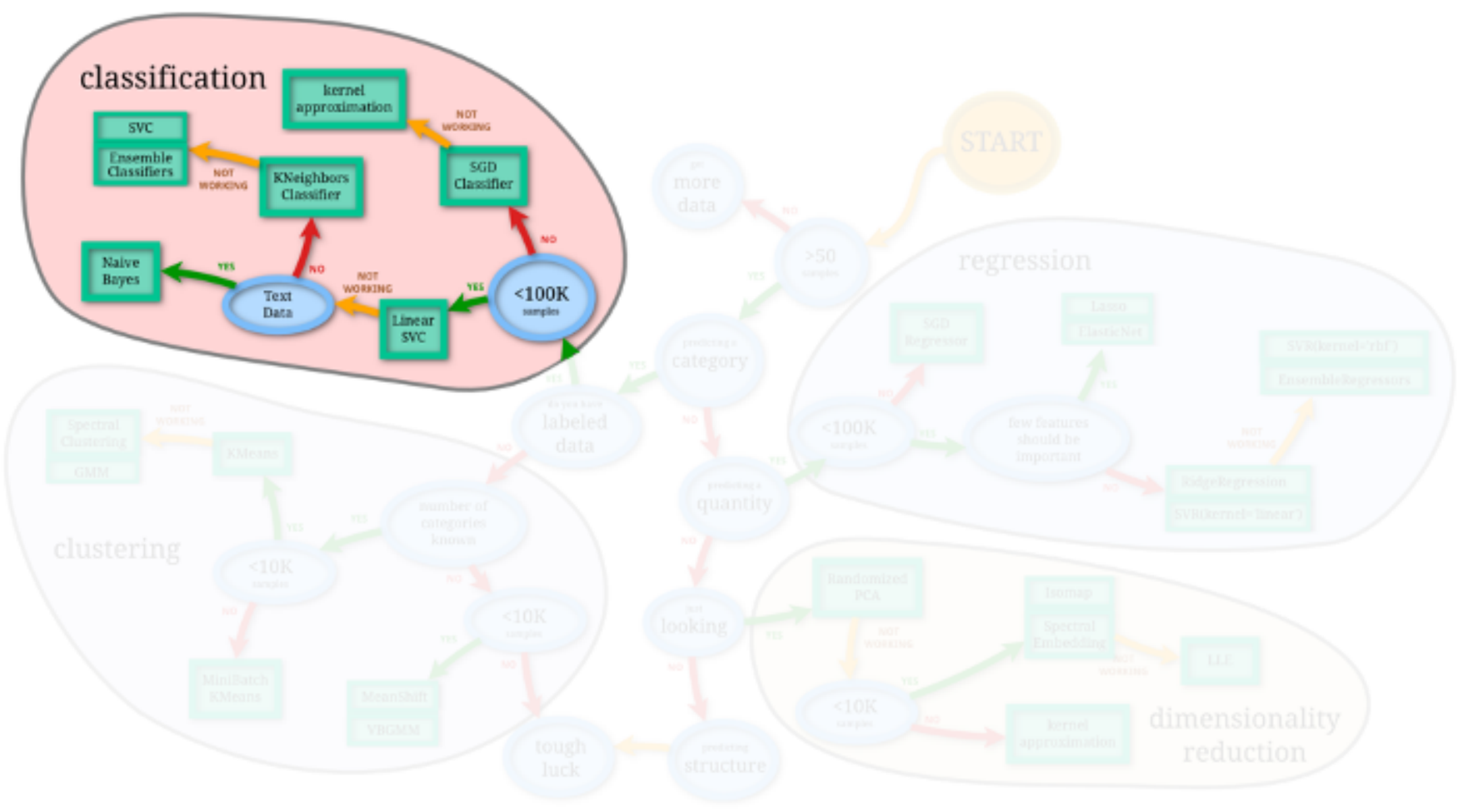
- Principal component analysis provides component ordered by maximum variance.
- Structure of the original data can be preserved
- It can be used as dimensional reduction technique before clustering



- Non-linear methods such as t-SNE and UMAP **are not distance** preserving
- Not ideal for clustering data

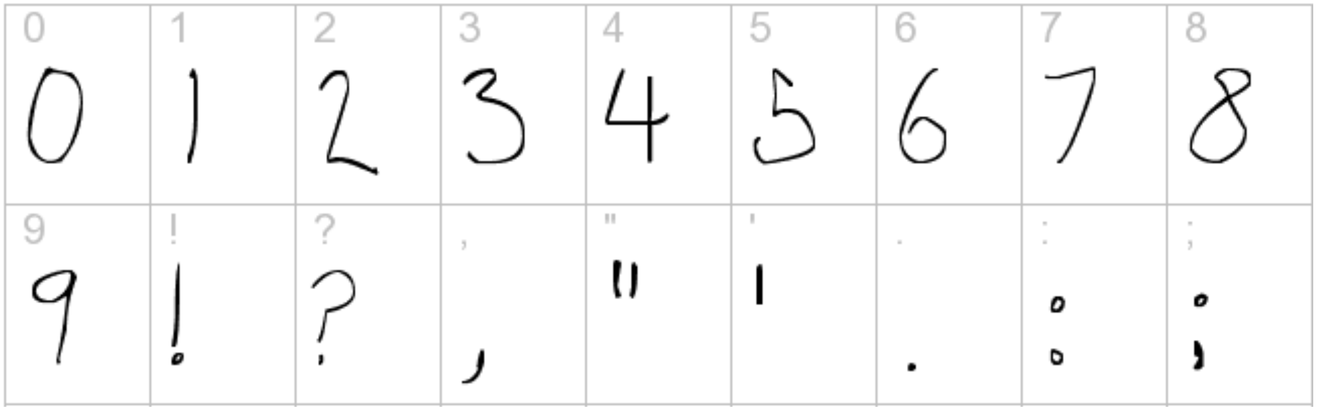
Lasers Med Sci 34, 35–40 (2019)

The Data Mining World — Classification problem

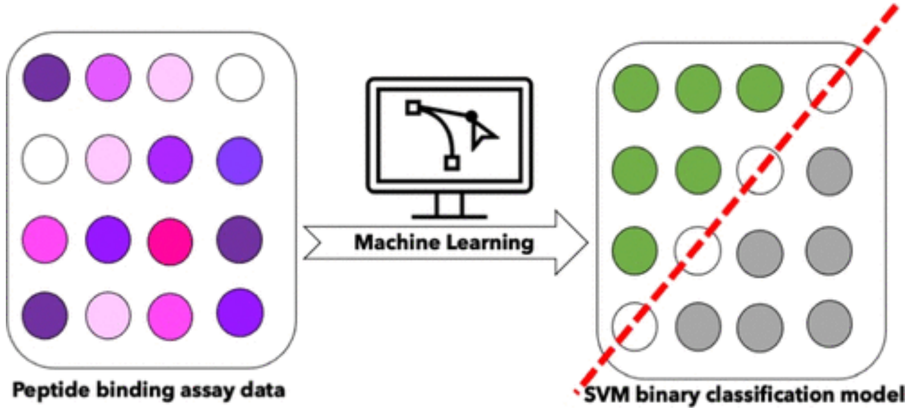


From **scikit-learn.org**

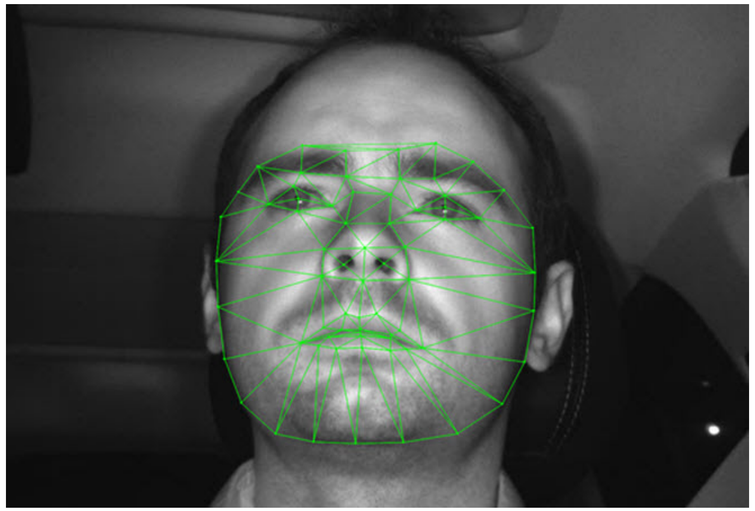
Classification problems are ubiquitous

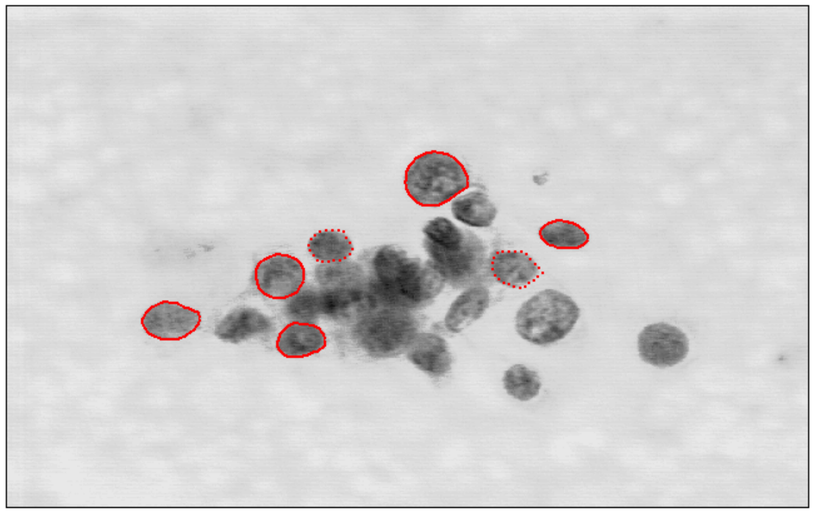


http://yann.lecun.com/exdb/mnist/

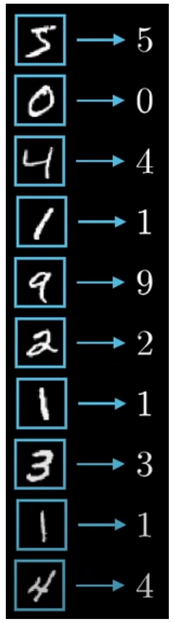


ACS Omega 2022, 7, 16, 14069–14073



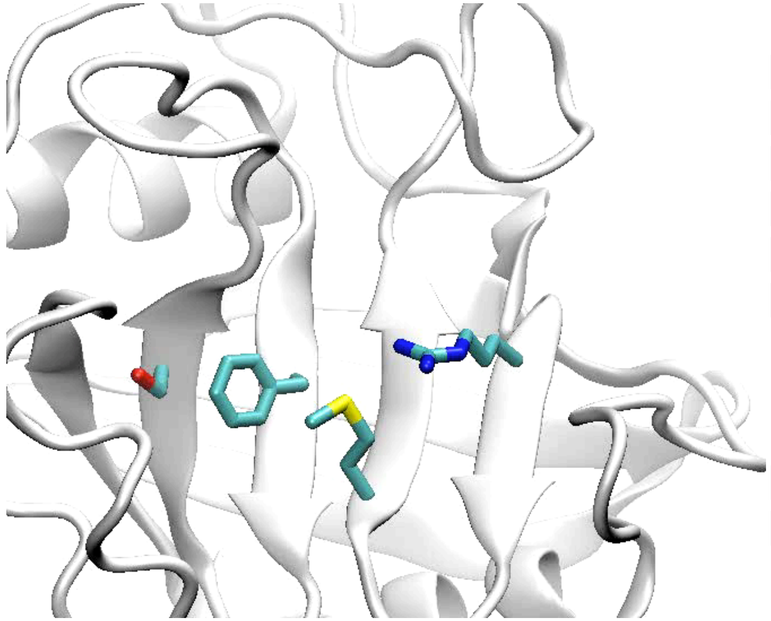


Labels are needed for supervised learning tasks



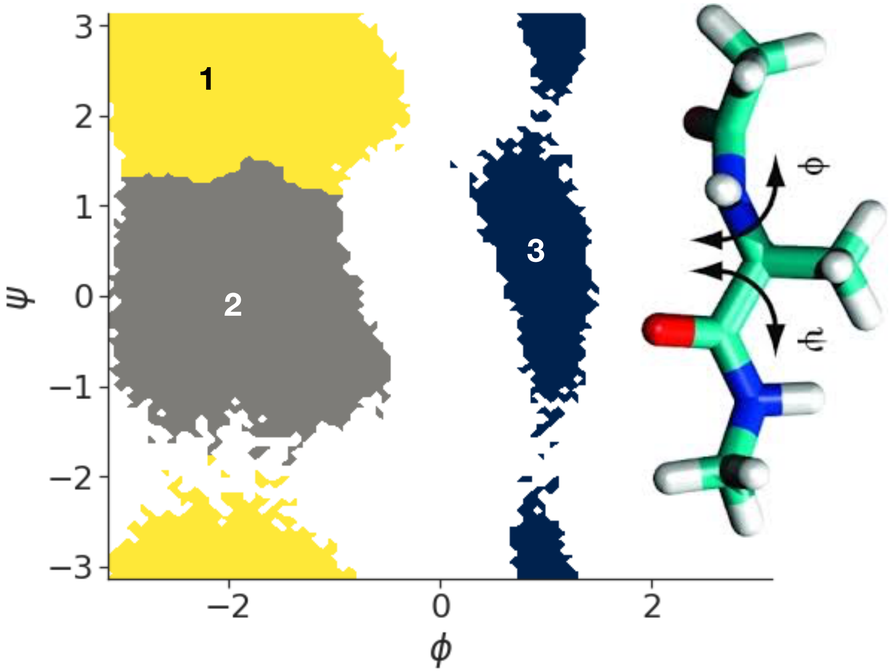
Label

Labelled energies



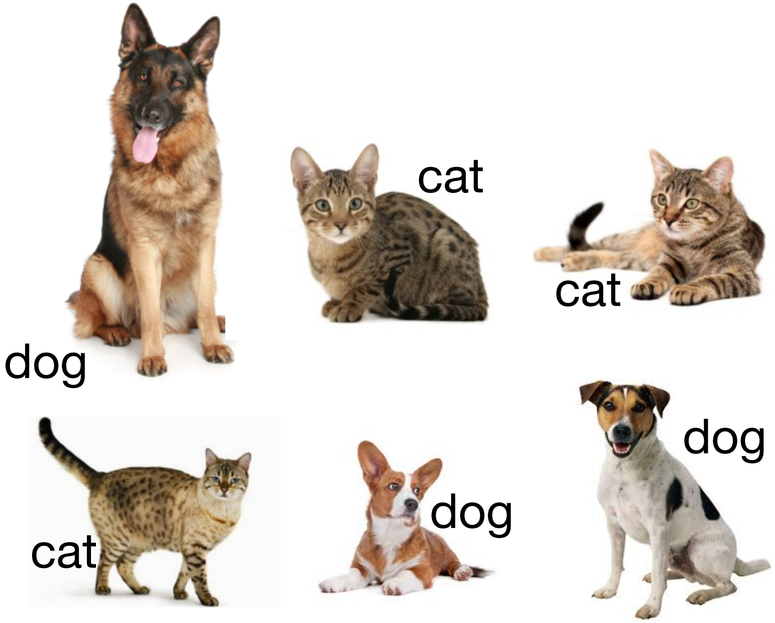
[-12008 kJ]
[-12078 kJ]
[-12045 kJ]
[-12083 kJ]
[-12062 kJ]
[-12058 kJ]
.
.
.
[-12099 kJ]
[-12093 kJ]

Labelled states

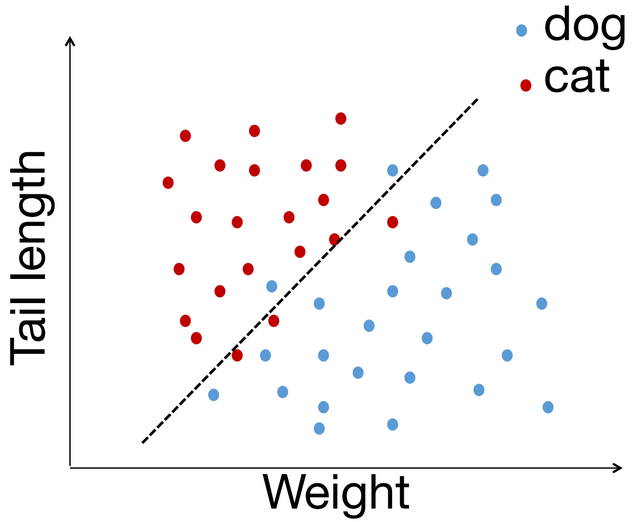


Data Classification via Supervised Learning

- Take **labelled data**
- Create an n-dimensional **feature vector** from data
- Separate “feature space” in different regions

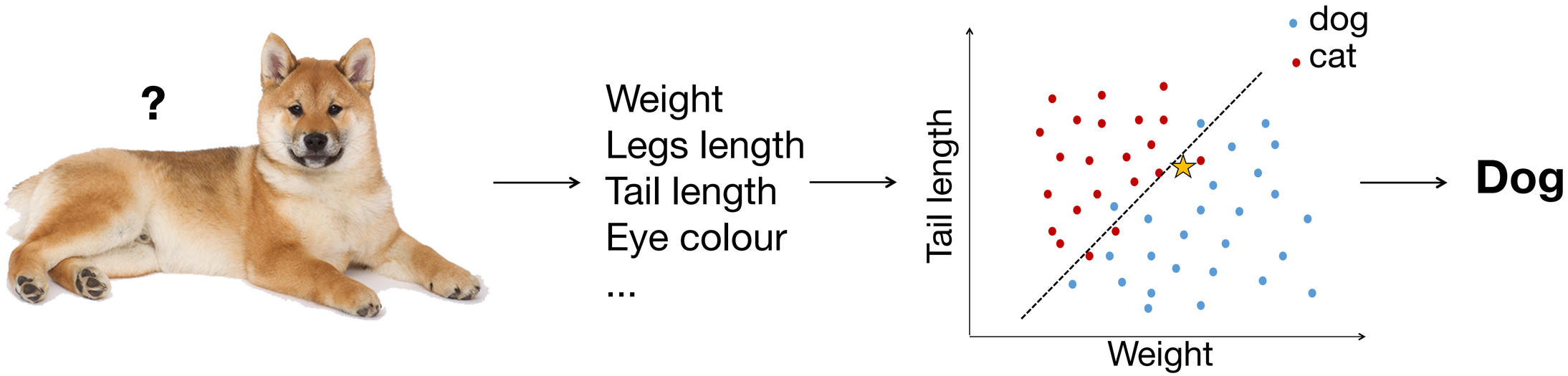


Weight
Legs length
Tail length
Eye colour
...



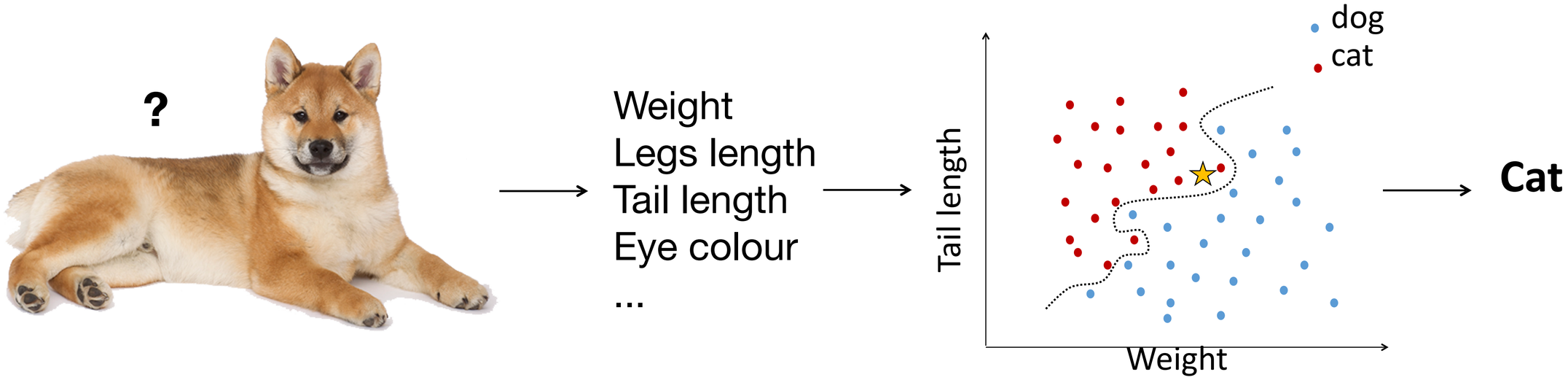
Data Classification via Supervised Learning

- Take **labelled data**
- Create an n-dimensional **feature vector** from data
- Separate “feature space” in different regions



Data Classification via Supervised Learning

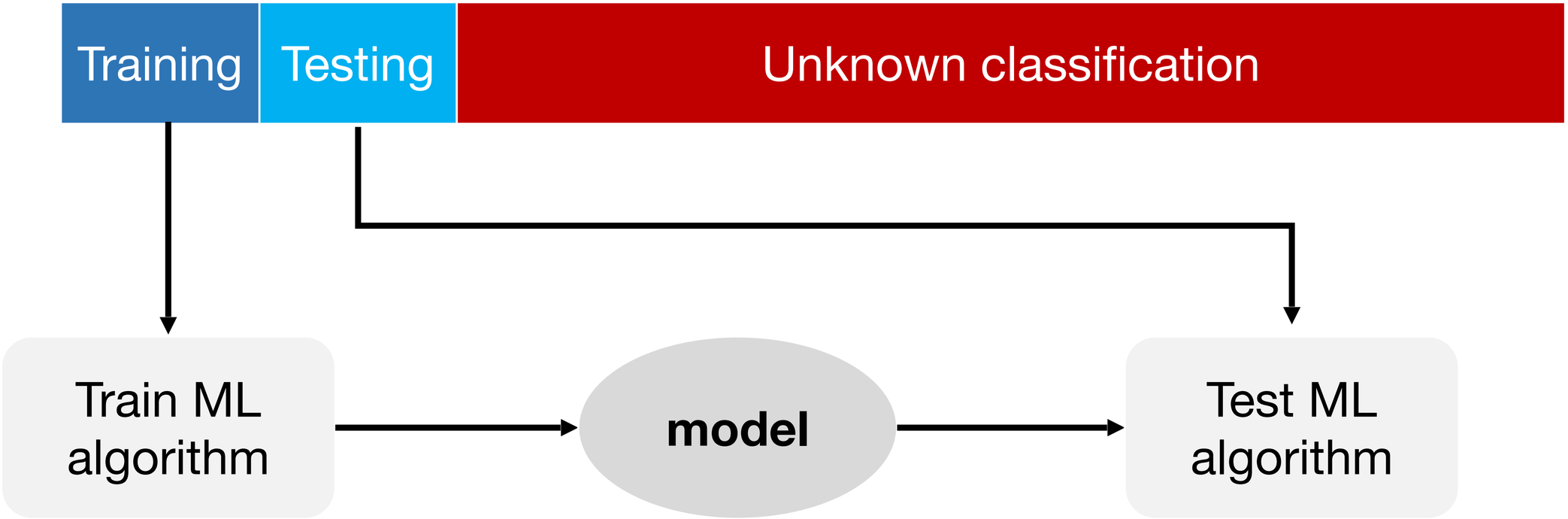
- Take **labelled data**
- Create an n-dimensional **feature vector** from data
- Separate “feature space” in different regions
- Warning: a too precise classification of examples might sacrifice generality (**overfitting**)



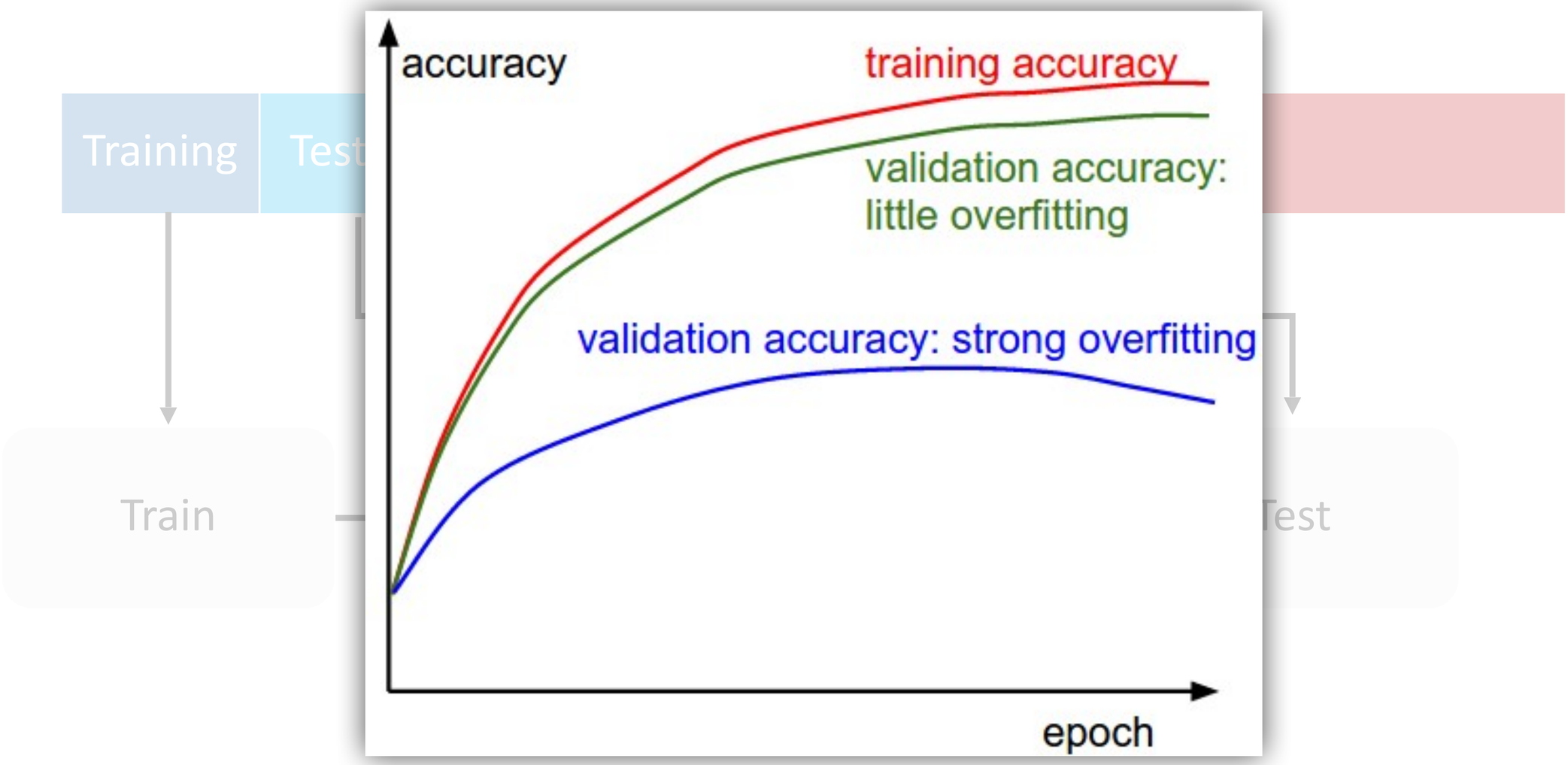
How to handle the data for classification

Data

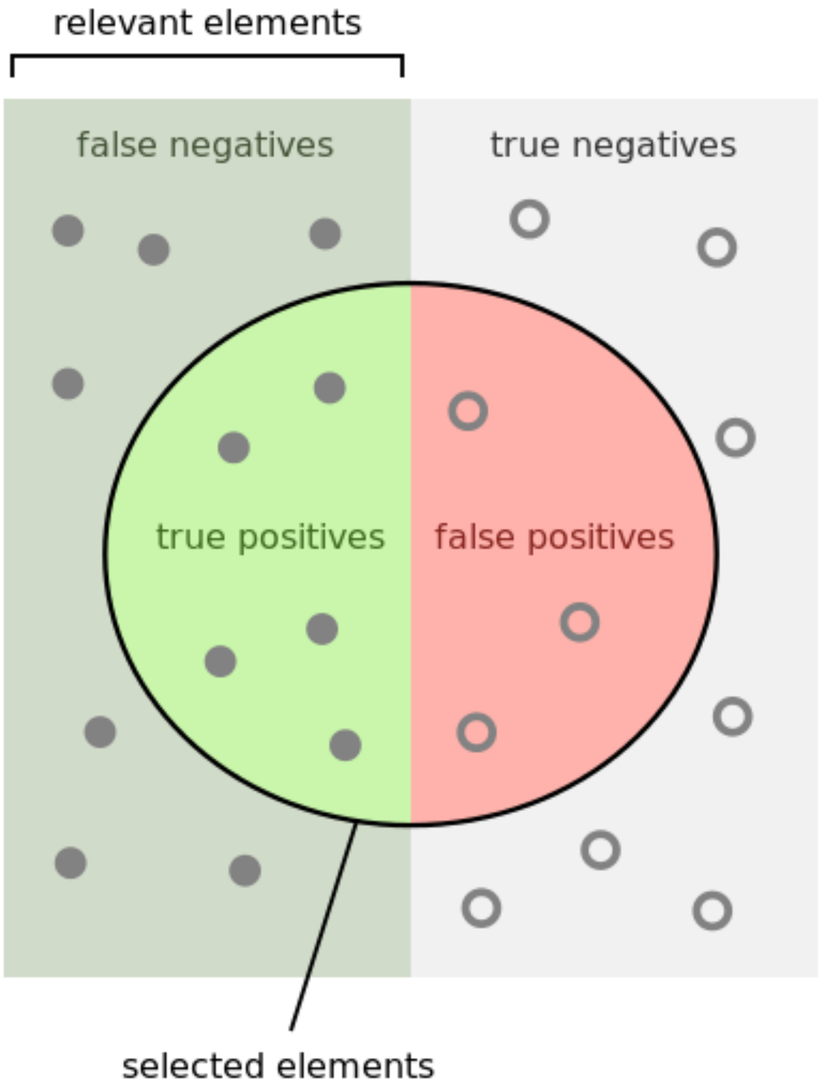
How to handle the data for classification



Data Classification using supervised learning



More nomenclature



- **Confusion Matrix:**
  describes classification results
  can also describe n classes

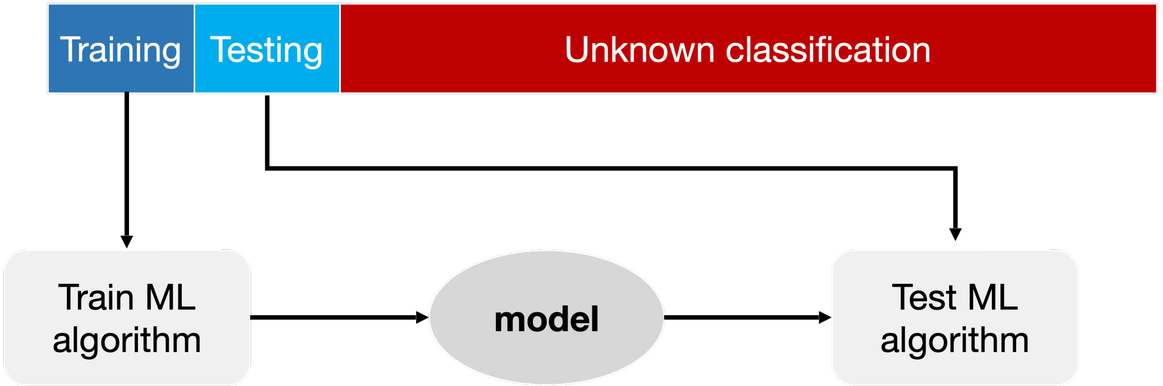
result \ real	Dog	Cat
Dog	90	10
Cat	12	88

- $\textbf{precision} = \dfrac{\text{true positives}}{\text{selected elements}} =$
- $\textbf{sensitivity} = \textbf{recall} = \dfrac{\text{true positives}}{\text{relevant elements}} =$
- $\textbf{accuracy} = \dfrac{\text{true positives} + \text{true negatives}}{\text{total population}}$

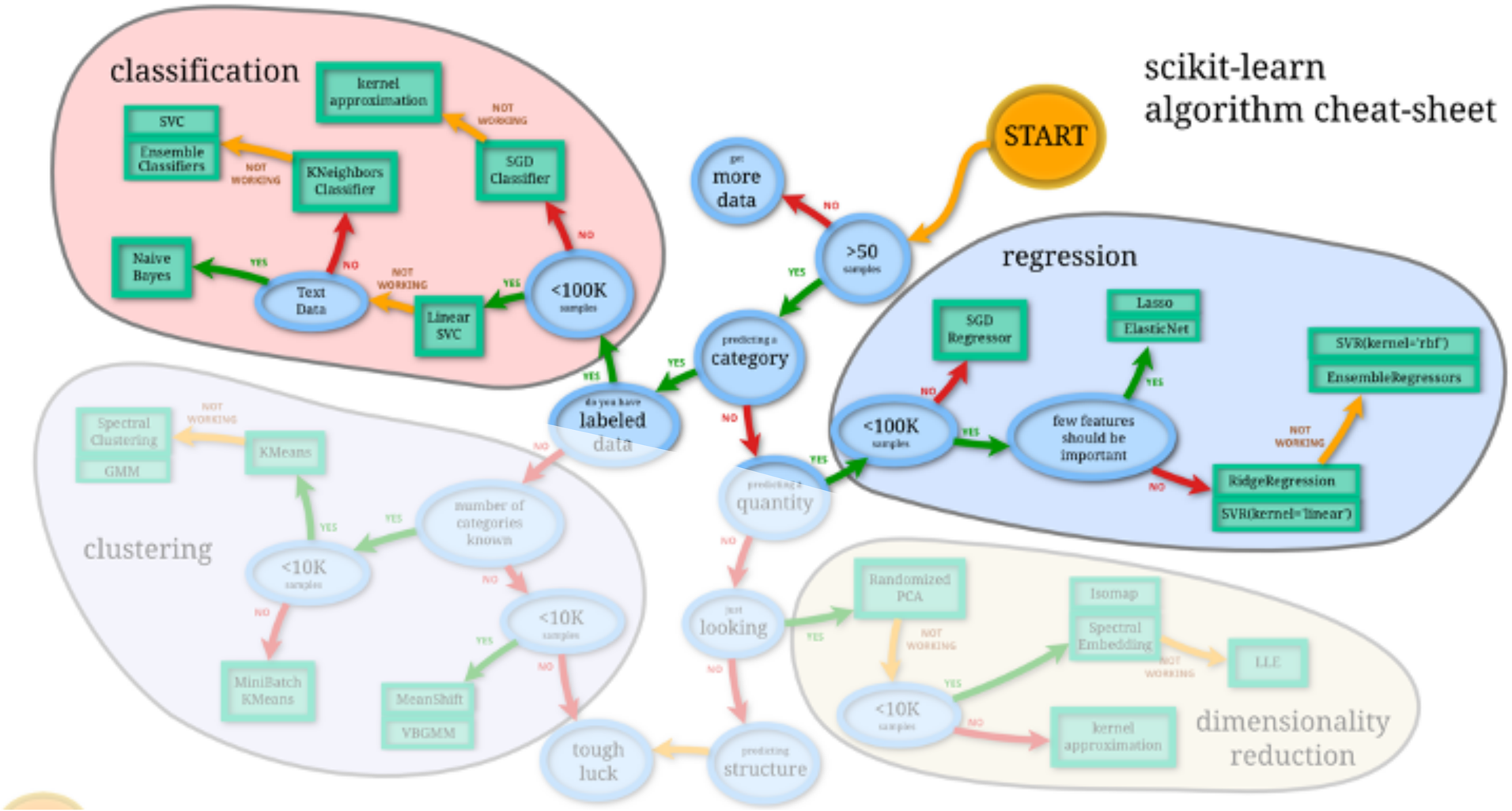
en.wikipedia.org/wiki/Precision_and_recall

Learning Algorithms

- Decision Tree (DT)
- Random Forests (RF)
- Artificial Neural Network (ANN)
- Support Vector Machine (SVM)
- Logistic Regression (LOGRES)
- Naïve Bayes (NB)
- K Nearest Neighbor (KNN)
- ...



ML Algorithms for classification and regression



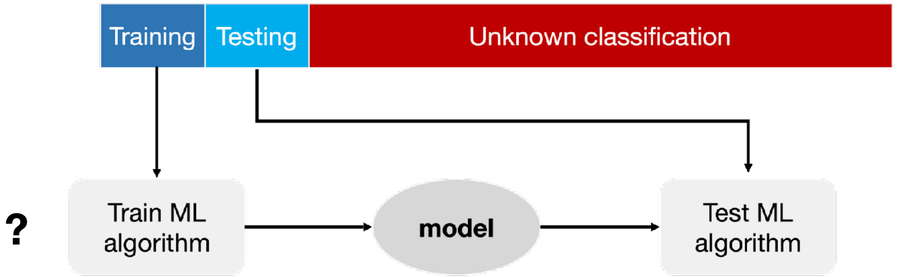
Learning Algorithms

Classification task

- **Decision Tree (DT)**
- **Random Forests (RF)**
- **Artificial Neural Network (ANN)**
- Support Vector Machine (SVM)
- Logistic Regression (LOGRES)
- Naïve Bayes (NB)
- K Nearest Neighbour (KNN)
- ...

Regression task

- **Decision Tree Regression (DT)**
- **Random Forests (RF)**
- **Neural Network Regressions (ANN)**
- Support Vector Machine (SVM)
- Linear Regression (LR)
- K Nearest Neighbour (KNN)
- Gaussian Regression
- ...

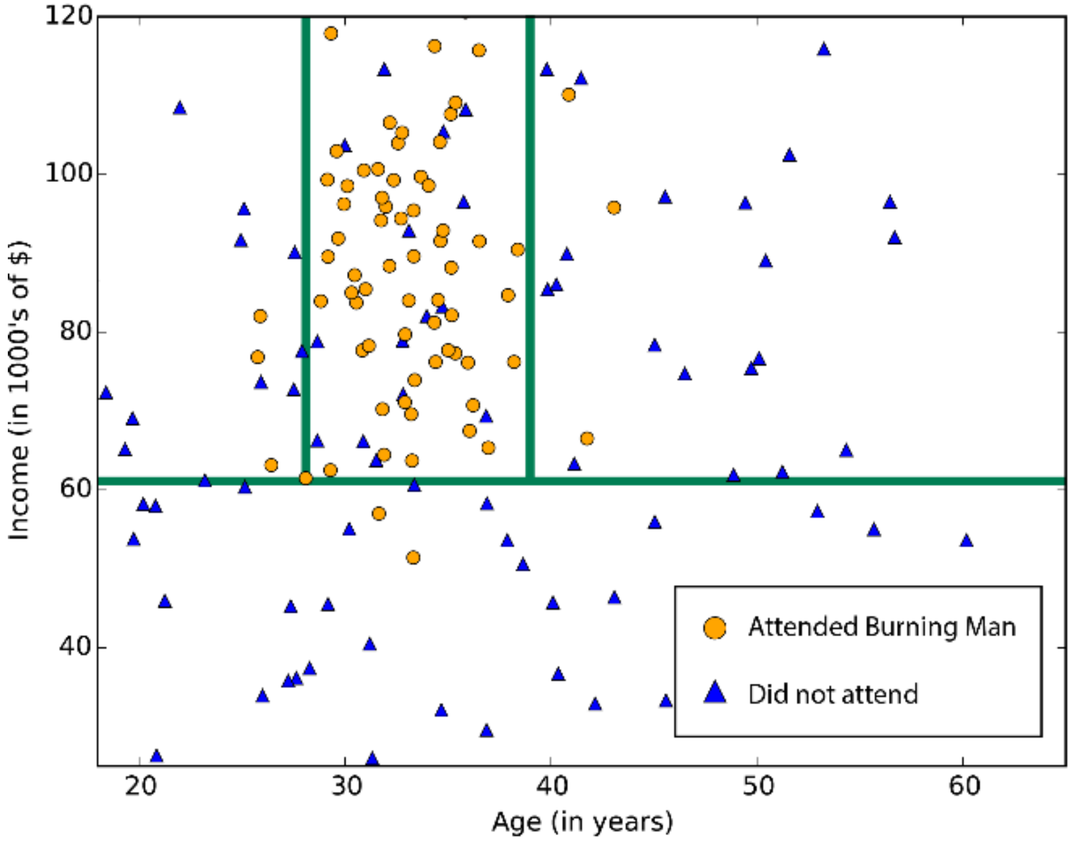


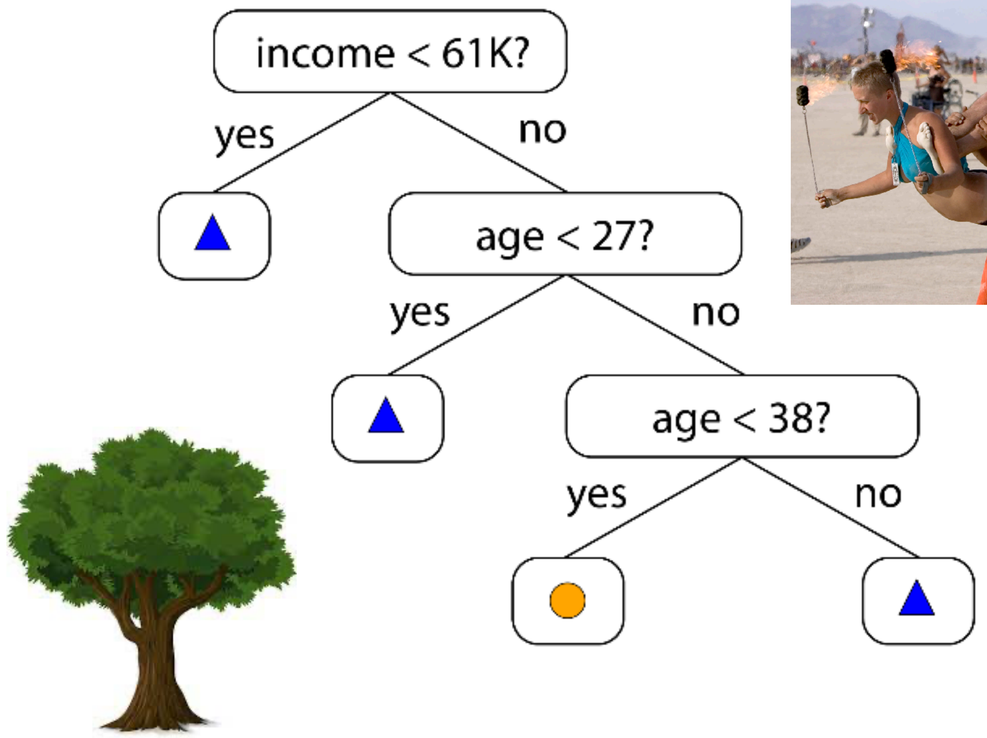
How do I pick the best learning algorithm?

Learning algorithms quality criteria:

- **accuracy**: percentage of correct classification
- **robustness**: handling noise and missing values
- **efficiency**: time to construct and use the model
- **scalability**: efficiency in memory requirements
- **interpretability**: how much the model is understandable

Decision Trees (DT)





- Subdivides features space in sectors
- Can overfit if space subdivision becomes too fine

azure.microsoft.com/en-us/documentation/articles/machine-learning-algorithm-choice

Bootstrap Aggregating (Bagging)

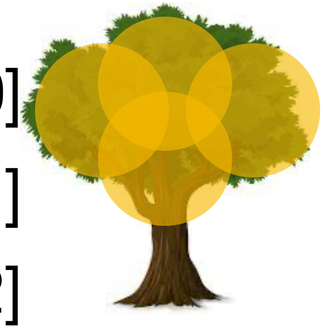
A weighted sum of weak classifiers creates a single strong classifier

Useful when a small change to training set causes large change in the output classifier ("learner is unstable")

training set *D* with *m* examples

Create *N* bootstrap samples S drawing *m* random examples from *D* *with replacement*

D=	1	2	3	4	5	6	7	8	9	
S[0]=	5	1	7	2	7	9	2	6	5	→C[0]
S[1]=	9	4	7	1	2	8	9	7	6	→C[1]
S[2]=	0	8	2	0	9	7	7	0	1	→C[2]
					...					
S[*N*]=	1	2	3	4	5	6	7	8	9	→C[*N*]



Training: for every S, build a distinct classifier C using the same learning algorithm

L. Breiman, *Bagging Predictors,* in *Machine* Learning, Springer, 1996

Random Forests (RF)

- **Data bagging**: creates *N* decision trees trained on bagged data
- **Feature bagging**: Given *M* features, every tree learns on $m \ll M$ randomly selected features
- Classification based on **voting** of resulting *forest*

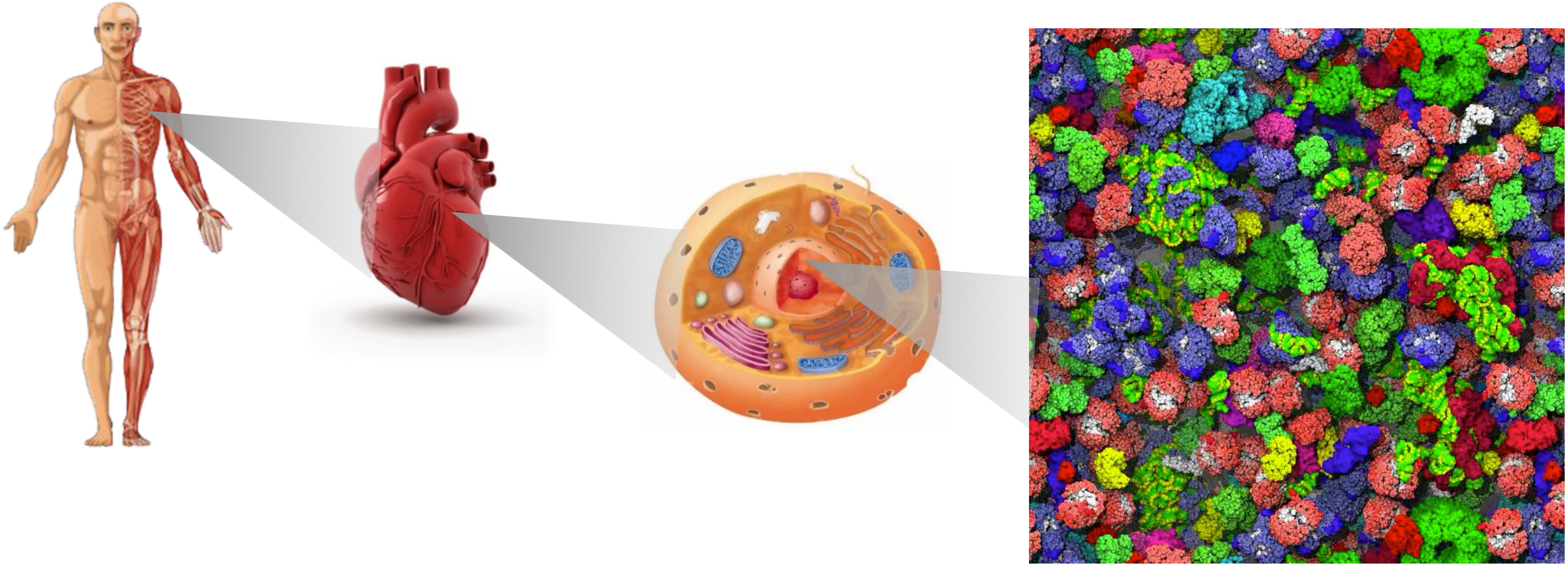
Advantages:

- does *not* overfit easily
- can handle thousands of features
- estimates what variables are important for classification

L. Breiman, *Random Forests*, Machine Learning, 2001

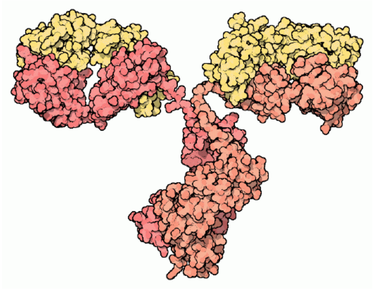
Classification of conformations of a protein

Structure determines the function?

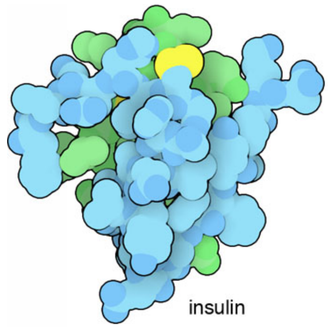


Proteins are molecular machines managing life

Antibody

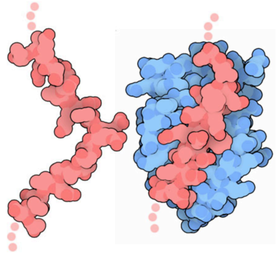


disease



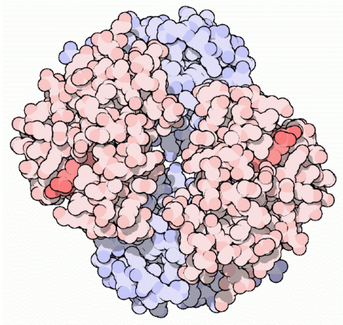
hormones

NOXA



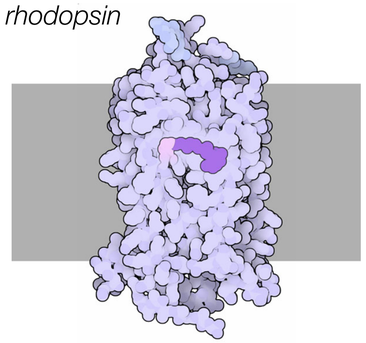
Cell regulation

hemoglobin



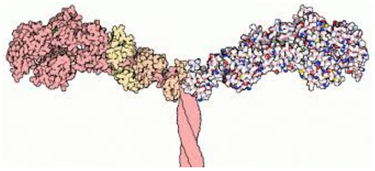
transport

rhodopsin



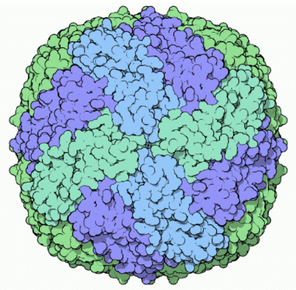
receptors

myosin



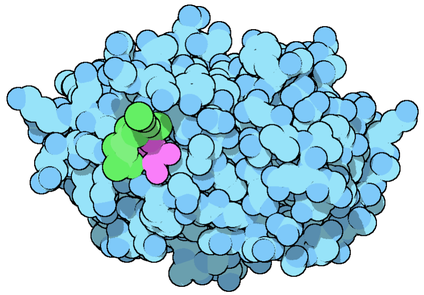
energy

ferritin



storage

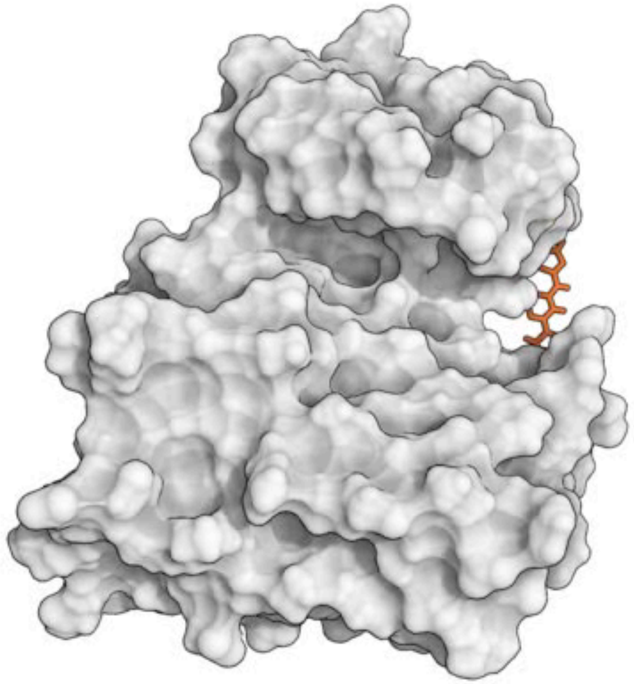
trypsin



enzymes

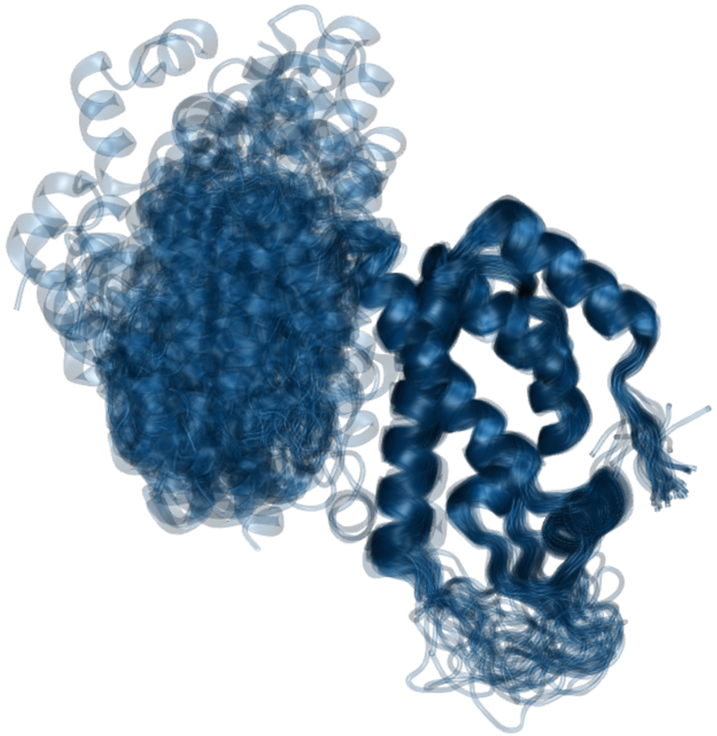
https://pdb101.rcsb.org/

Dynamics determines the function



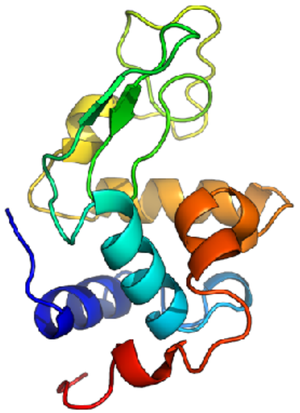
Shan Y, et al. *JACS* **133,** 9181 (2011)

Mey et al. *Living J. Comp. Mol. Sci. 2 (1), 18378 (2020)*



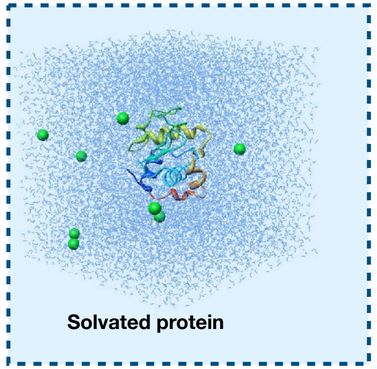
Degiacomi, *Structure* **27**, 1034 (2019)

Molecular dynamics simulations can provide quantitative observables

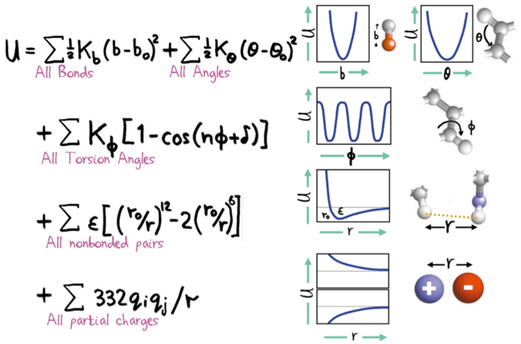


Disulphide bridges
Protonation
Crystal Waters
Cofactors
Alternative side chains

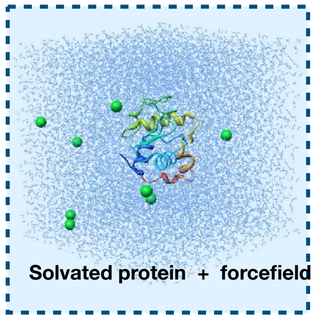
Solvate in water and add ions



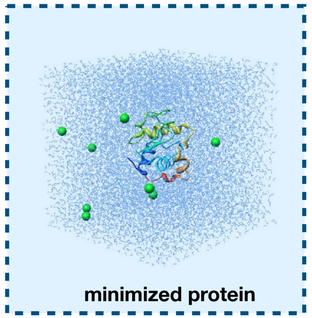
Apply forcefield description



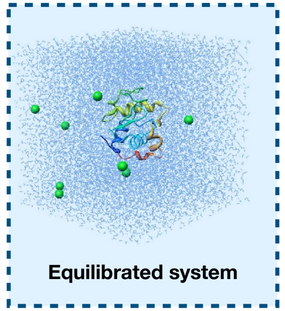
$$\vec{F}_i = -\frac{dU}{d\vec{x}_i}$$



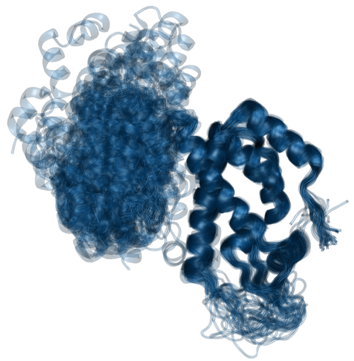
Minimize the positions



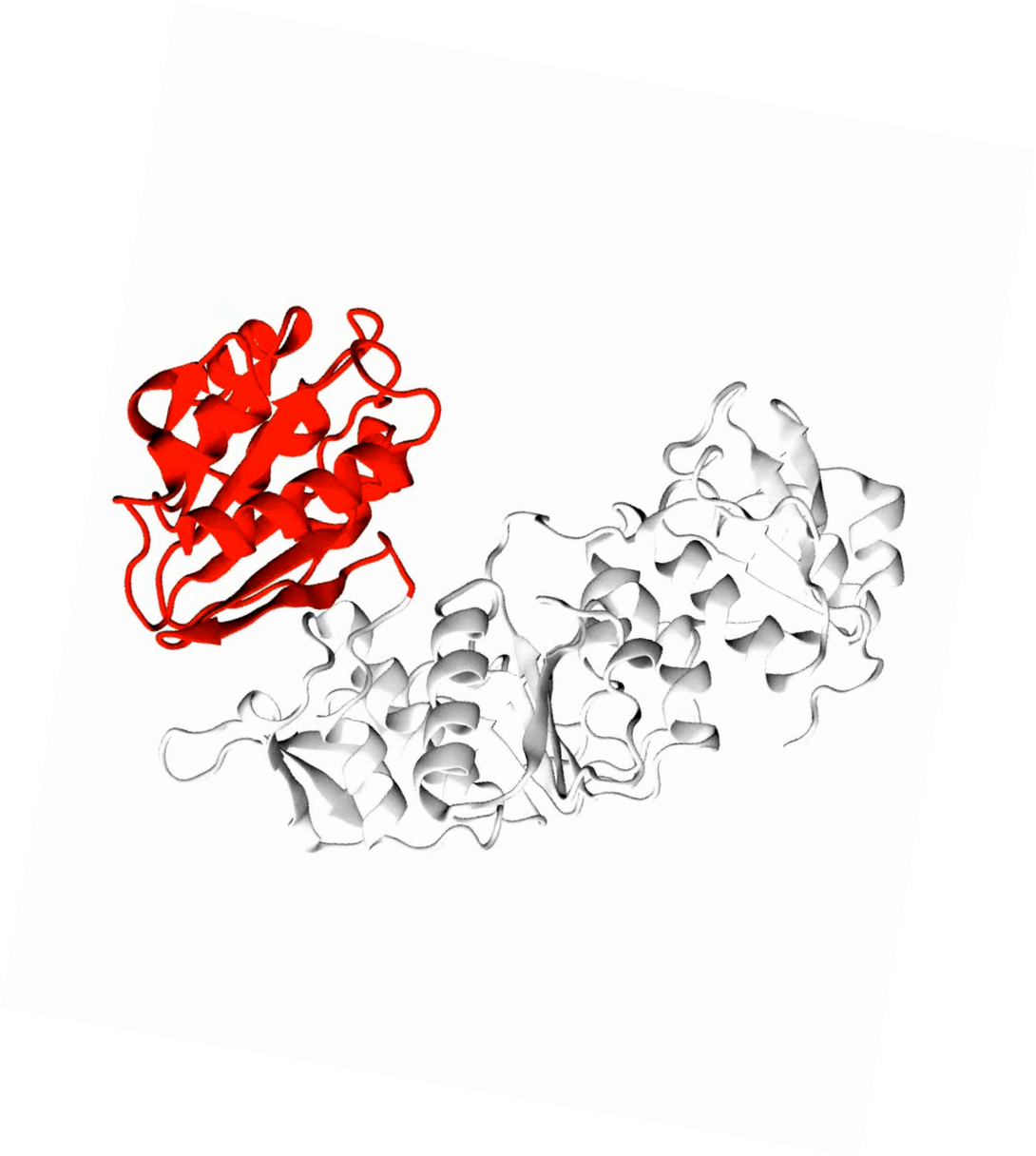
NVT/NPT equilibration



Production simulation



Classification of conformations of a protein



- Labels are given: open or closed
- What are good features?
- Try different features with a random forest

Take homes

- Classification requires the **knowledge of labels** for training data and they are categorical
- Non-linear regressions will also have labels but they **are not categorical**
- Many different algorithms exist for classification and regression
- Both are forms of **supervised machine learning**