Machine Learning 101

ML4HS @UC, lesson 1

Preliminaries

Outline (preliminary)

Lesson 1:
Theory, Introduction to machine learning

Lesson 2:
Practice: an example of supervised learning

Lesson 3:
Theory: Unsupervised learning and Deep Learning introduction

Lesson 4:
Practice: Representation learning, unsupervised learning, latent spaces for simple Neural networks

Lesson 5:
Theory/practice: CNNs theory and exercises

Lesson 6:
Theory/practice: Features selection, cross validation, regularization

Lesson 7:
Theory: Explainability for DL? (with code review)

Lesson 8:
Practice: Segmentation with the MONAI framework

Homeworks and “final” test

Each week I will provide a Jupyter Notebook with guided python exercises.

The homeworks can be done in a group, they are “mandatory” but they are not graded. Each group will create a github/gitlab account and will upload the solved exercise. I will provide solutions/feedbacks and we can discuss about it in class.

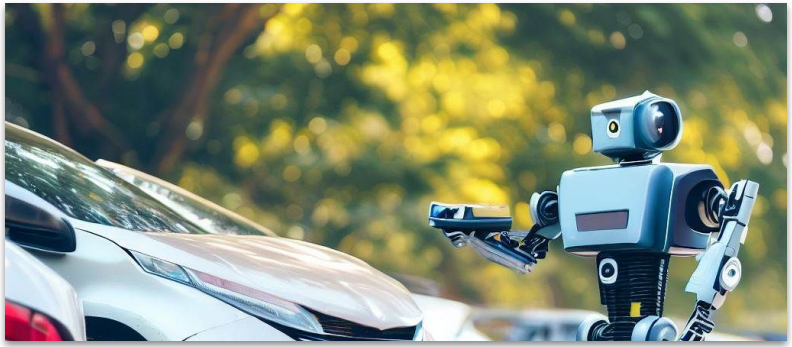
The final test will consist in the application of machine learning to solve a simple problem. Each group will prepare a short presentation about the chosen solution.

Books

- Theobald, O. (2017). *Machine learning for absolute beginners: a plain English introduction* (Vol. 157). London, UK: Scatterplot press
- Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2, pp. 1-758). New York: springer
- Alpaydin, E. (2021). *Machine learning*. Mit Press
- VanderPlas, J. (2016). *Python data science handbook: Essential tools for working with data*. O'Reilly Media, Inc.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press
- ...
- There are many others... and don’t forget the many resources online

Prerequisites

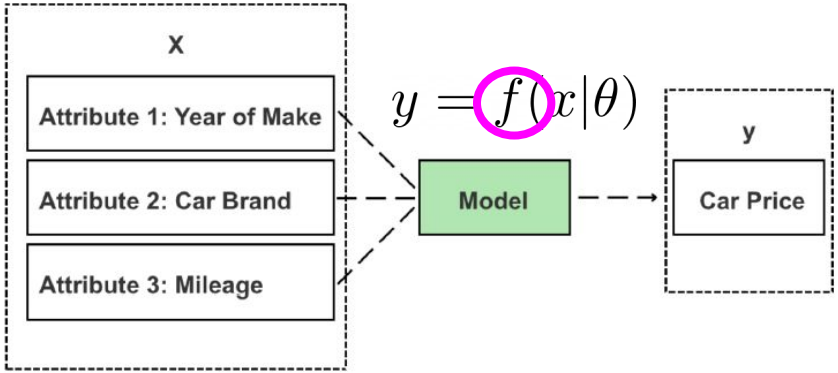
- **Useful**: Linear algebra, basics of statistics and probability
- **Indispensabile**: A bit of understanding of python programming
  - Tutorial video (YouTube): https://t.ly/wYvi
  - Tutorial web: https://t.ly/dKza
  - Tutorial su NumPy, Matplotlib, Pandas: https://t.ly/wp_J



Let’s start with an example

Estimating (predicting) the price of a used car

- We do not know the exact formula for this; at the same time, we know that there should be some rules (“a function f”)
  - Car brand, year of make, mileage, etc.
- Many applications exist where we do not know the rules but have a lot of data
  - Is this a dog or a cat? Will he develop this disease? Is this customer “credit worth”?
  - “I can tell apart dogs and cats!” - ok: you “know” the formula, but you cannot “program” it



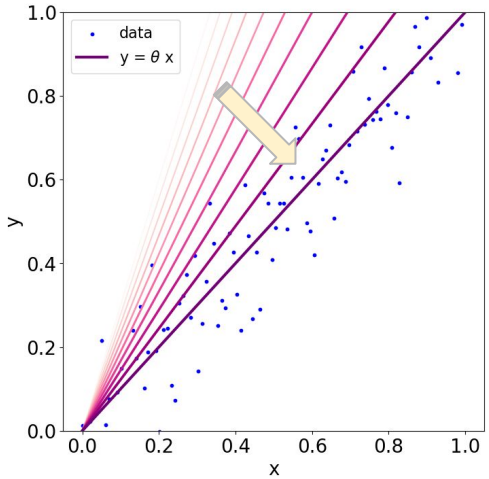
Basically, f is a model is a machine

Programming vs learning

- Traditionally, to make a computer perform a task, you had to give it a set of clear instructions
  - “If X then Y”
- Could we make machines perform a task using input data rather than relying on a direct input command (without being explicitly programmed)?
- That is: can a machine “learn” to do something?
- Learning means getting better through experience
  - “Better” implies **a performance criterion** that is optimized
  - “Experience” implies **data** collected in the past

Learning = adjusting the parameters of your model

$$y = f(x|\theta) \equiv \theta\, x$$



What do we really want? (our taks)

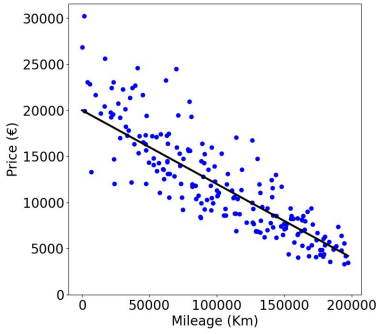
- The task is defined by what we exactly choose to give as input and what we want as output

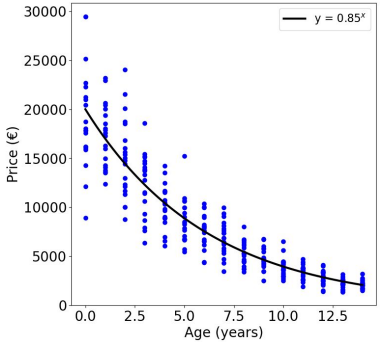
- **Accurate or “robust”?**
  - For example, in representing a car, if we use the brand as an input attribute, we can pin down the price very precisely
  - But if we instead use (few) general attributes (number of seats, engine power, trunk volume, ...), we can learn a more robust estimator
  - Instead of estimating the price, it makes more sense to estimate the percentage of its original price, that is, the effect of depreciation

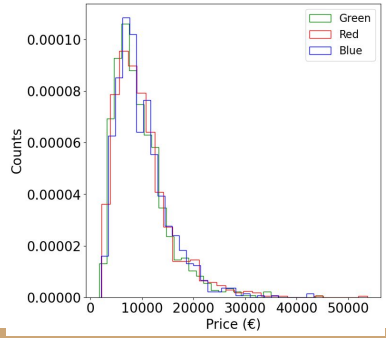
- Yet, beware the trade-off: as you become more general (you broaden your scope), your accuracy (and “depth”) will suffer
  - Do you really need to have one single machine for used cars and used trucks??

Defining our input

- We have to the choose the attributes that we believe have an effect on the price of a used car
- First we have to choose how to represent them to make them computer-friendly
  - Mileage and age are numbers, we use them as they are
  - A car can be 4WD or not: we use 1 if it is, 0 if it is not
  - We have three colors: {green, blue, red}... What can we do with this?
  - Frame number is kind of a number, but irrelevant - drop it
- Then, we make some plots
  - Mileage: good correlation, ok
  - Age: it seems like a car loses a fraction of its value every year
  - Histogram of the prices (green, red, blue)
    - They seem to be identical: we drop the color feature







“Probably the major driving force of the computing technology is the realization that every piece of information can be represented as numbers”

Grace Hopper

Accepting randomness

- We can work hard as we wish, yet two different cars, having exactly the same values for these attributes, can still go for different prices
  - So, when **testing your machine**, you cannot wish for perfection (rather, you should be afraid of (near) perfection!)
- In the end, we will only be able to provide some sort of “average” or “expected” price
- (or, if we are very smart, an interval in which the unknown price is likely to lie)

Defining parameters, f(x, θ) and minimize the error

- Given the insight gained by our visual exploration, the following could be a good model of your data:

$$y^{\text{pred}} = (\text{price\_new} - \beta\,\text{mileage})\,\alpha^{\text{age}} \qquad \theta = \{\alpha,\ \beta\}$$

- Yet, of course, you can use other models
  - Deep learning is all about forgetting about devising clever models, by using very complex (and adaptable) machines
- Then, we want to find the parameters θ = {α, β} that minimize the error:

$$\text{err}(\theta) = \sum_i \left(y_i - y_i^{\text{pred}}(\theta)\right)^2$$

Knowledge, impermanence, and ego

- With your machine you are “distilling” (or “compressing”) a lot of knowledge: once you learn the rule, you do not need the data anymore
- But be aware that this knowledge could be (and probably is) transient: the rules (can) change (in time and “space”)
  - Market trends, shifts in drivers’ behavior and preference, the economy, new technologies
- And always keep in mind that just because we have a lot of data, it does not mean that there are underlying rules to be learned
  - Phone books contain thousands and thousands of record: would you try and predict a phone number given name and surname of a person??

A bit of history and context

Arthur Samuel - 1959

In 1959, Arthur Samuel published a paper in the IBM Journal of Research and Development with an intriguing and obscure title - “Some Studies in **Machine Learning** Using the Game of Checkers”.

The paper aimed “*to verify the fact that a computer can be programmed so that it will learn to play a better game of checkers than can be played by the person who wrote the program*.”

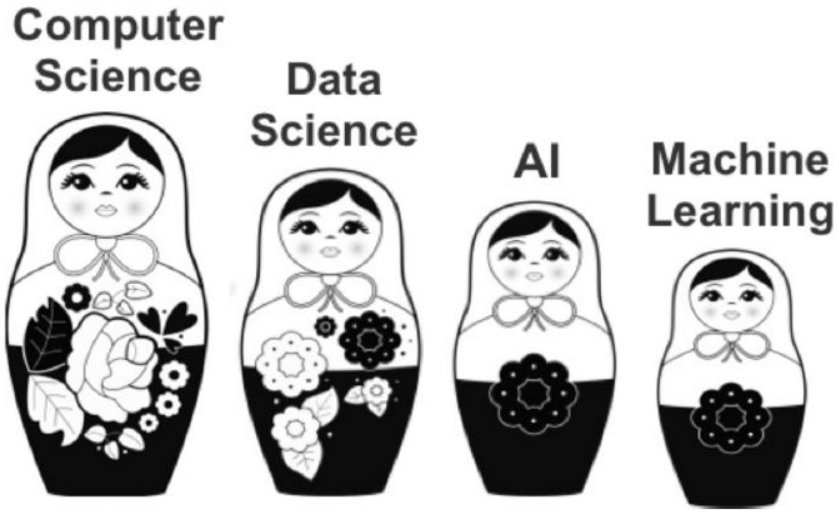
Samuel did not invent the term “machine learning” - but he is credited as the first to give it the meaning we employ today.

And then?

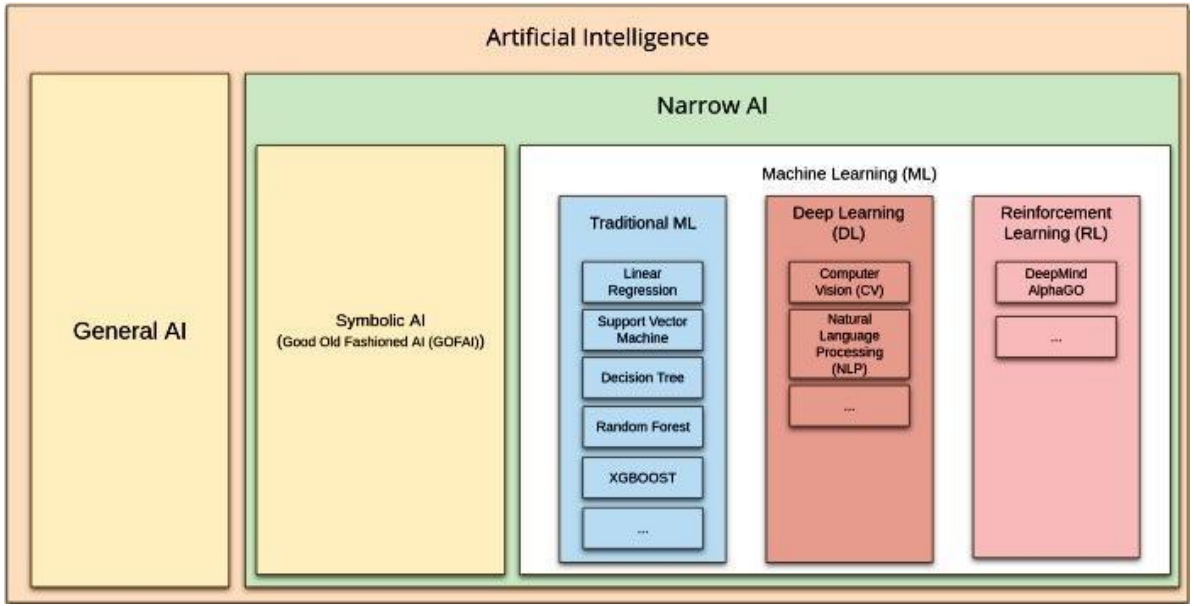
- Computer became more powerful
  - In the mid-1980s, a huge explosion of interest in artificial neural network models
  - Machine learning took inspiration from cognitive science and neuroscience (and it still does)
- Data flooded the (interconnected) world
  - Databases, computers everywhere, the Internet, wearables
- 2006

- And many small (and a few medium-sized) improvements from then on
  - Sort of accumulated wisdom, in the forms of rules-of-thumb (“this kind of model is good for this”)

Where is machine learning?



And Deep Learning?



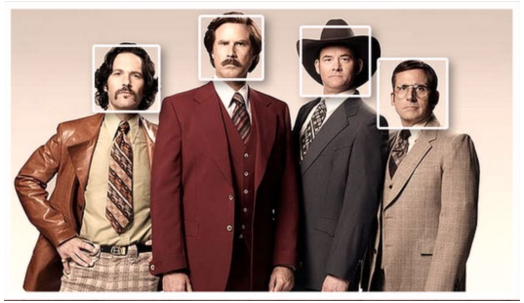
A formal definition of Machine Learning

“A computer program is said to learn from **experience E** with respect to some class of **tasks T** and **performance measure P**, if its performance at tasks in T, as measured by P, improves with experience E”

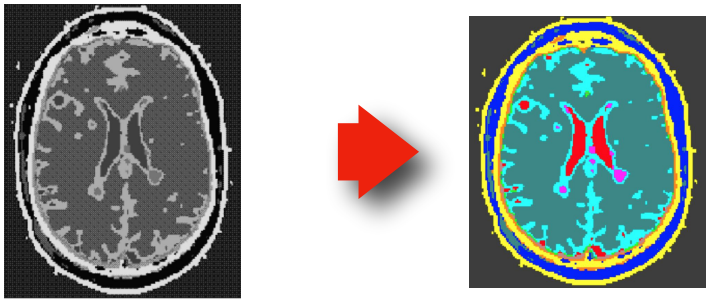
Tom M. Mitchell, 1993

Examples

Face Detection
- Task: face or not face?
- Experience: parts of pictures



Medical Image Detection e Segmentation
- Task: identify different tissues
- Experience: images

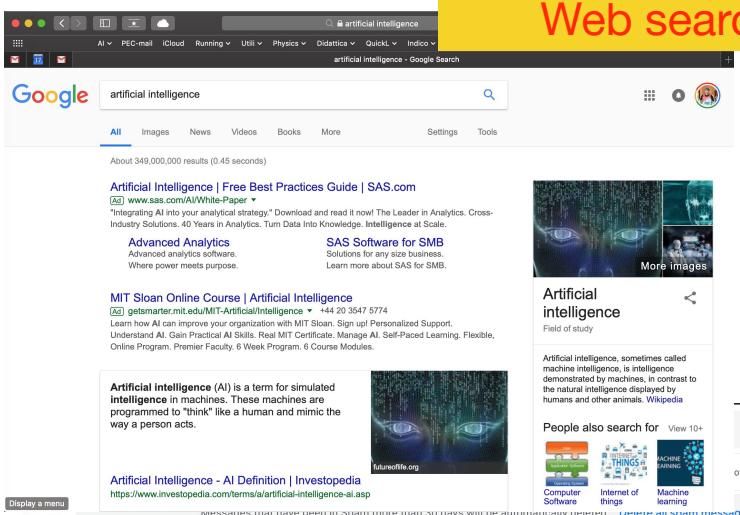


Voice recognition
- Task: identify phonemes
- Experience: acoustic signals

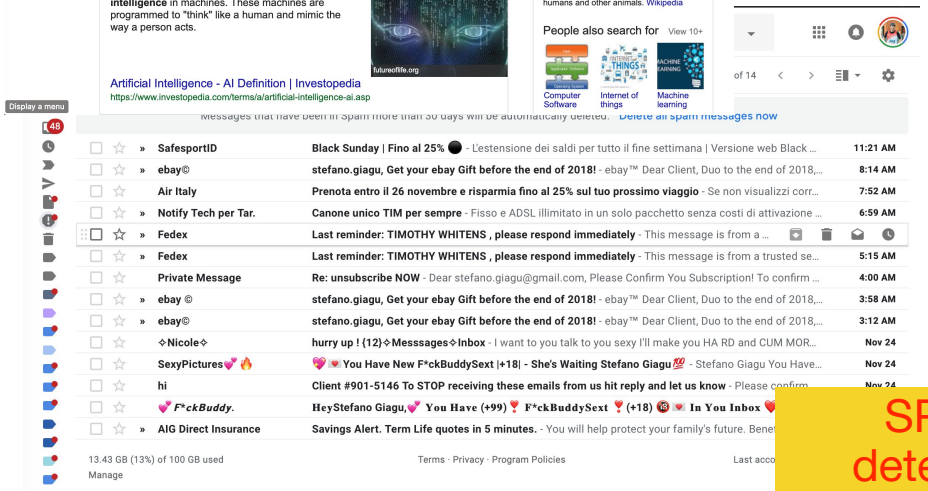
ma-chi-ne-le-ar-nin-g

Industrial applications

Web search



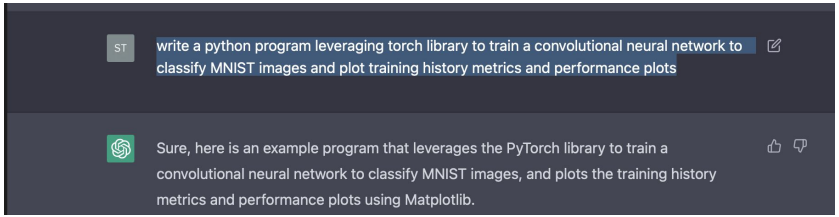
Autonomous cars



SPAM detection

Autonomous drones

LLM - Large Language Models (DALL-E, CHATGPT, ...)



"A rabbit detective sitting on a park bench and reading a newspaper in a victorian setting"

"macro 35mm film photography of a large family of mice wearing hats cozy by the fireplace"

Machine Learning *vs* Statistics

The key differences

In machine learning you call “**supervised learning**” what in statistics you call “**regression**”

In machine learning you call “**input/feature**” what in statistics you call “**predictor**” (or independent variable)

In machine learning you call “**output**” what in statistics you call “**response**” (or dependent variable)

Getting philosophical (and simplifying a great deal)

- Science overarching goal: **to predict and influence events**
  - (with “economy”, scope, and depth)
- **Statistics** is more into **influence**
  - The focus is on determining the variables that control a phenomenon
  - “How much smoking affects the probability of lung cancer?”
  - “If I take this pill, will my health get better?”
- **Machine learning** is decidedly more into **prediction**
  - As long as you can tell apart cats and dogs, I don’t care how you do that
  - The “black box” problem
- This leads also to different “technologies”
  - In statistics you favor simple, interpretable models; usually you have less data
  - In ML you need more data, but your models are able to discern very complex (and often non-intuitive) patterns
- Are they converging?
  - Explainability, complex interpretable models

Machine Learning categories

Supervised learning

The process of **understanding input-output relationships** is called supervised learning. The model analyzes and deciphers the relationship between input and output data **to learn the underlying patterns**. Input data is referred to as the **independent variable** (uppercase “X”), while the output data is called the **dependent variable** (lowercase “y”).

The name comes from the supposition that there is a supervisor who can provide us with the desired output for any input.

“Estimating the price of a used car”

Unsupervised learning

Unsupervised learning refers to algorithms that learn patterns from unlabeled data. They do so by learning concise representations of the input data, which can be used for data exploration or to analyze or generate new data.

Basically two approaches

1. **Dimensionality reduction**
    - Your data has N dimensions; find a compressed representation in K (K << N) dimensions
    - Kind of “lossy compression” (like a ZIP file, but yuor txet is a bit mxied up)
    - PCA, autoencoders
2. **Clustering**
    - You have M data points; but you can identify groups of points that are clearly separated
    - “Why there are no human races”
    - K-means, hierarchical, density-based

Applications: visualization, denoising, market segmentation, anomaly detection…

Semi-supervised learning

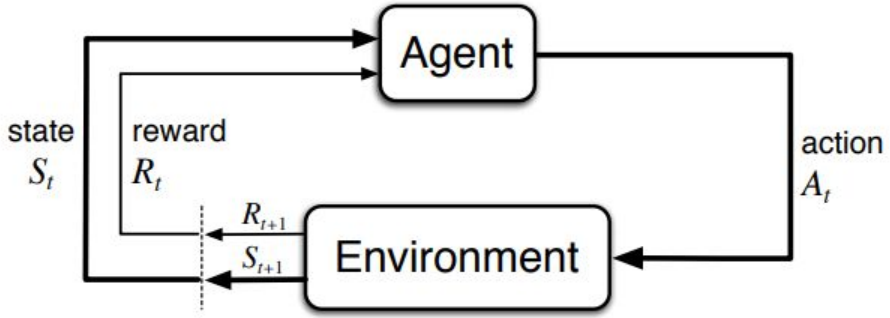
With the “more data the better” as a core motivator, the goal of semi-supervised learning is to leverage unlabeled cases to improve the reliability of the prediction model.

Two approaches:

1. Train, label, retrain (convergence? Convergence to a good model?)
2. Dimensionality reduction (labelled and unlabelled data) + shared internal representations for your prediction (trained on labelled data only)

Reinforcement learning

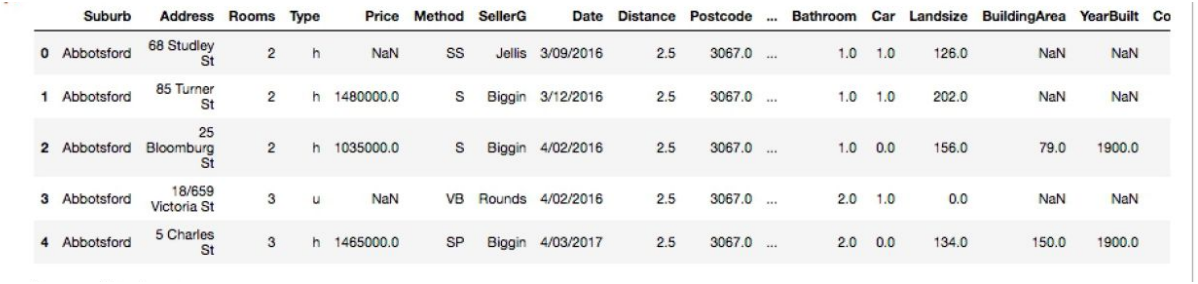
- Reinforcement learning is learning what an intelligent agent ought to do in an environment — how to map situations to actions — so as to **maximize a numerical reward signal…**
- **… by trial-and-error** (observe, act, receive reward, update → observe…)
- NOT supervised NOR unsupervised
- RL is simultaneously a problem, a class of solution methods that work well on the problem



The ML toolbox

Data

- Structured (text) vs unstructured
- A tabular dataset contains data organized in rows and columns



	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	YearBuilt	Co
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	NaN	NaN	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	NaN	NaN	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	79.0	1900.0	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	NaN	NaN	
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	...	2.0	0.0	134.0	150.0	1900.0	

- Columns are features (or variables/dimensions/attributes)
- Rows are examples (or cases/data points)
- Visual exploration
  - Histograms (range, asymmetry, outliers, gaps), scatterplots (correlations)
- You can find hundreds of interesting datasets in CSV format from kaggle.com
- **Advanced**: Big Data (petabytes ~ 1000 of your hard disk)

Infrastructure

- A computer or online platforms (Colab)
- Tools for processing data (*e.g.*, Jupyter Notebook)
- Programming language, like Python
- Numerical libraries, like NumPy, Pandas, Scikit-learn, TensorFlow, Pytorch
  - A collection of pre-compiled programming routines to execute algorithms with minimal use of code
  - Pandas derives from the term “panel data,” similar to “sheets” in Excel and MySQL tables
  - NumPy gives you large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions
  - Scikit-learn provides access to a range of popular shallow algorithms, including linear regression, clustering techniques, decision trees, and support vector machines
  - Pytorch/Tensorflow: make it easy to define and train large neural network models
- Visualization libraries, like Seaborn and Matplotlib
  - data exploration and communication
- **Advanced**: graphics processing unit (GPU) vs central processing unit (CPU) - about 1,000x
  - Amazon Web Services, Microsoft Azure, Alibaba Cloud, Google Cloud Platform, and other cloud providers offer pay-as-you-go GPU resources

Algorithms

- Basically, your f(x|θ) can take zillions of forms
  - Linear regression, logistic regression, decision trees, support vector machines, k-nearest neighbors…
  - And, of course, **neural networks** (yes, neural networks, we will see, are just functions, defined in a very peculiar way)
- For unsupervised learning
  - k -means, PCA…
  - And, of course, **neural networks**
- Ensemble models
- Ok, we have simplified a bit too much…
- Parametric models
  - We estimate (the parameters of) a single, global model for all our data
  - Regressions, neural networks
- Non-parametric models
  - The only information we use is the most basic assumption—namely, that similar inputs have similar outputs (or a “metric”, if you want to be fancy)
  - K-nearest neighbors

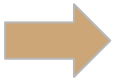
ML Workflow

Massaging the data

- Cleaning the data
  - Modifying and removing incomplete, incorrectly formatted, irrelevant or duplicated data
  - Converting text-based data to numeric values and the redesigning of features
- Feature Selection (column compression)
  - Drop irrelevant, weakly correlated, duplicate features
- Row Compression
  - Feature aggregation (non-numeric and categorical values can be problematic)
- Converting text-based values into numbers
  - True/False $\rightarrow$1/0
  - One-hot Encoding
- Binning
  - Home prices: the exact size of the pool is not that relevant (“small,” “medium,” and “large”)
- Normalization
  - Rescaling the range of values for a given feature into a set range ([0, 1])
- Standardization (z-scoring)
  - Feature a $\rightarrow$ â, so that $\langle \hat{a} \rangle = 0$, $\mathrm{Var}[\hat{a}] = 1$
- Missing Data
  - Replace with mode, median; remove row

One-hot encoding

Name in English	Speakers	Degree of Endangerment
South Italian	7500000	Vulnerable
Sicilian	5000000	Vulnerable
Low Saxon	4800000	Vulnerable
Belarusian	4000000	Vulnerable
Lombard	3500000	Definitely endangered
Romani	3500000	Definitely endangered
Yiddish	3000000	Definitely endangered
Gondi	2713790	Vulnerable
Picard	700000	Severely endangered



Name in English	Speakers	Vulnerable	Definitely Endangered	Severely Endangered
South Italian	7500000	1	0	0
Sicilian	5000000	1	0	0
Low Saxon	4800000	1	0	0
Belarusian	4000000	1	0	0
Lombard	3500000	0	1	0
Romani	3500000	0	1	0
Yiddish	3000000	0	1	0
Gondi	2713790	1	0	0
Picard	700000	0	0	1

How much data do we need?

- ML works best when your training dataset includes a full range of feature combinations (“**curse of dimensionality**”)
- Absolute minimum: ten times as many data points as the total number of features
- ~10,000: clustering and dimensionality reduction algorithms can be highly effective
- < 100,000: regression analysis and classification algorithms
- > 100,000: neural networks more cost-effective and time-efficient for working with massive quantities of data

Choosing your model

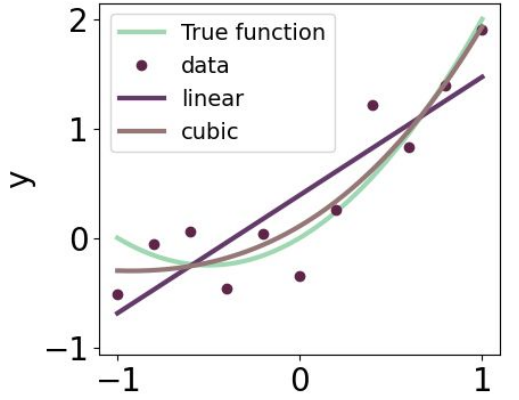
"Nature does not make jumps"
GW von Leibniz

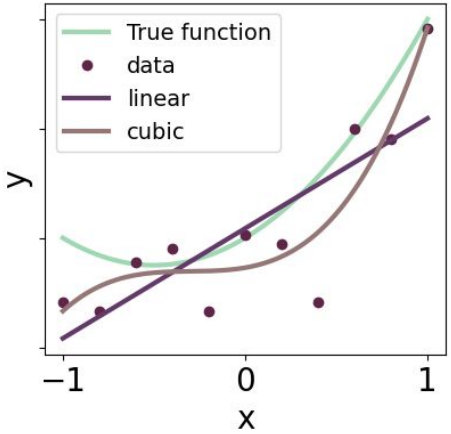
- There are no rules... but we want a model that **generalizes** well
  - Generalization refers to your model's ability to predict new, unseen data

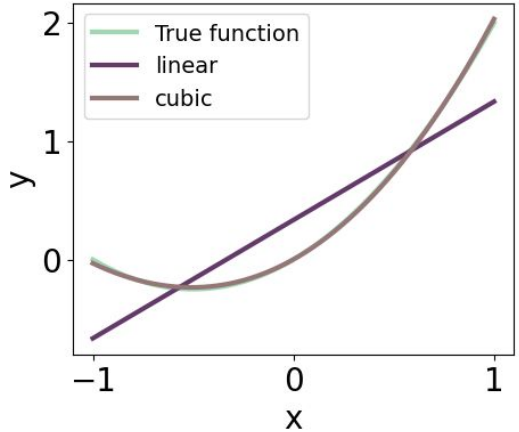
$$y_{\text{new}}^{\text{pred}}(\hat{\theta}) = f(x_{\text{new}}|\hat{\theta}) \text{ is close to } y_{\text{new}}?$$

- Each model - each form of $f(x|\theta)$ - comes with a set of assumptions: its **inductive bias**
- Bias and variance
  - Some models are more "stiff" - they don't follow closely the data (pronte to **underfit**), but learn well with fewer examples
  - Some models are more "malleable" - they can adapt to the data (prone to **overfit**), but require a lot of examples

Bias and Variance (and regularization)



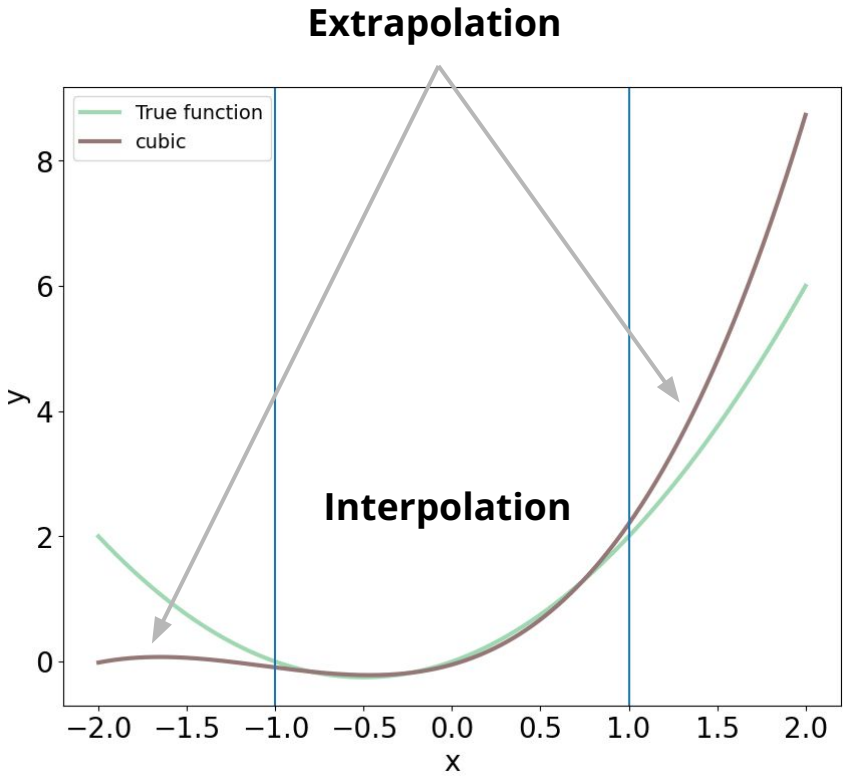




Regularization

- Techniques to make your model “stiffer”
- L1 (Lasso) and L2 (ridge) norms, early stopping...

Interpolation *vs* extrapolation (out-of-sample)



Take-home message: extrapolation is way harder than interpolation

Linear regression *vs* k-nearest neighbors

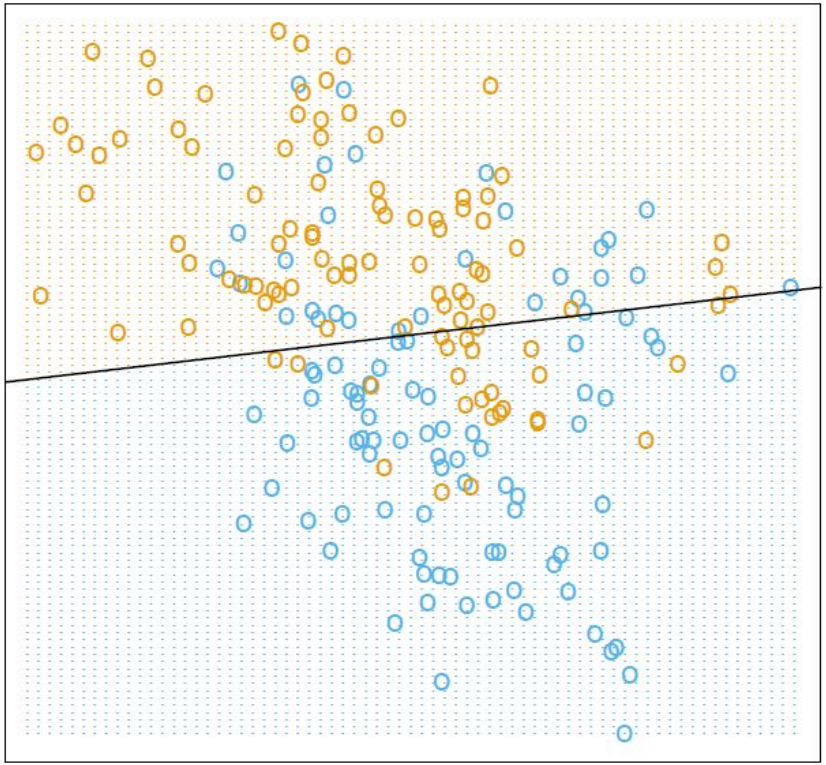
- A the extreme ends of the spectrum
  - The linear model makes huge assumptions about structure and yields stable but possibly inaccurate predictions
  - The method of k-nearest neighbors makes very mild structural assumptions: its predictions are often accurate but can be unstable
  - Parametric *vs* non-parametric
    - Effective number of parameters scales with number of examples

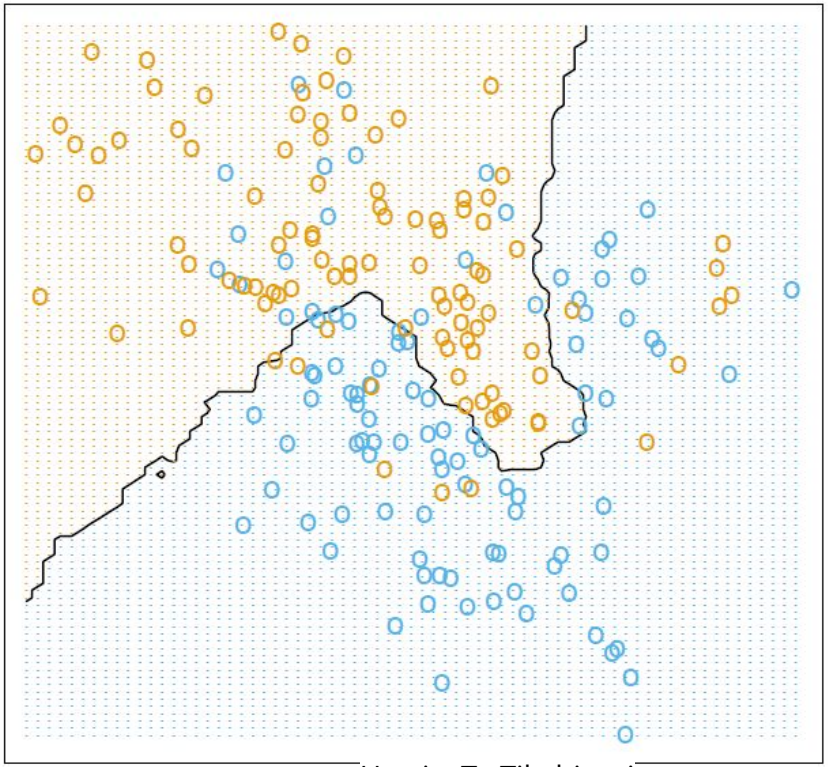
$$y_i^{\text{pred}} = \beta_0 + \sum_{j=1}^{p} \beta_j \, x_{ij}$$

$$y_i^{\text{pred}} = \frac{1}{k} \sum_{x_i \in \mathcal{N}_k(x)} y_i$$

- Two (limit) scenarios
  - The training data in each class were generated from bivariate Gaussian distributions with uncorrelated components and different means (good for linear)
  - The training data in each class came from a mixture of 10 low variance Gaussian distributions, with individual means themselves distributed as Gaussian (good for k-nearest neighbors

Linear regression *vs* k-nearest neighbors





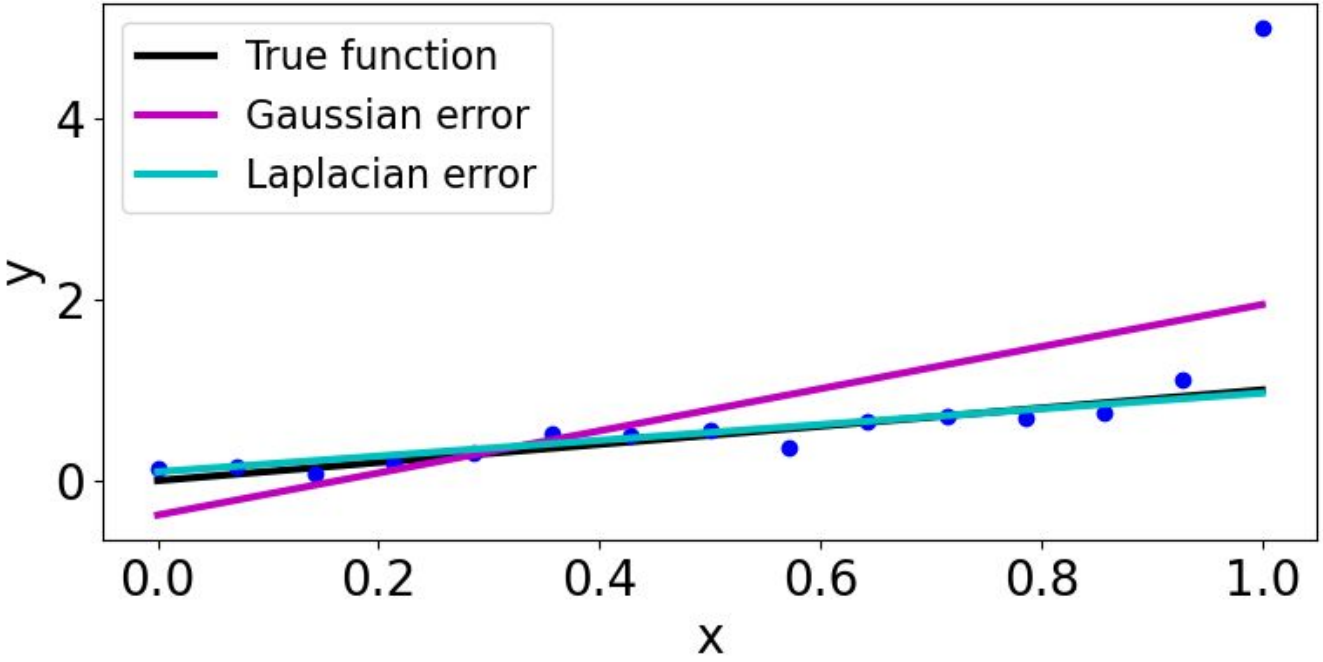
Hastie, T., Tibshirani

Choosing your objective

$$\hat{e}(y,\, y^{\mathrm{pred}})$$

- Accepting randomness (do you remember?) means interpreting errors in a probabilistic sense (-log-likelihood)
  - Mean squared error, $\hat{e} = (y - \tilde{y})^2$: y has a Gaussian distribution around the expected (predicted) value $\tilde{y}$
  - Mean absolute error, $\hat{e} = |y - \tilde{y}|$: y is Laplacian (two-sided exponential) around $\tilde{y}$
  - Cross entropy, $\hat{e} = y \log(\tilde{y}) + (1-y) \log(1-\tilde{y})$: y (y= 0 or 1) is a Bernoulli variable with $\langle y \rangle = \tilde{y}$
  - And so all the others (more or less)
- Reasoning probabilistically helps you understand your objective
- Changing objective function can lead to very different results!

Objective function matters a lot



False-positive *vs* False-negative

- How good a classifier is depends on the cost associated with different errors
- False-positive rate: FP/P
  - To put in jail an innocent is worse than leaving a criminal free
- False-negative rate: FN/N
  - To give parole to a re-offender is very costly (in societal terms)

	Action		
Truth	***Choose positive (start treatment)***	***Choose negative (discharge the patient)***	***Sum***
Positive (the patient has cancer)	TP: True positive	FN: False negative	P
Negative (the patient does not have cancer)	FP: False positive	TN: True negative	N
Sum	P′	N′	

Training your model

- Basically you need to find the parameters that minimize your error

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, E(\theta) \equiv \sum_i \hat{e}(y_i,\, y_i^{\mathrm{pred}}) \qquad y_i^{\mathrm{pred}}(\theta) = f(x_i|\theta)$$

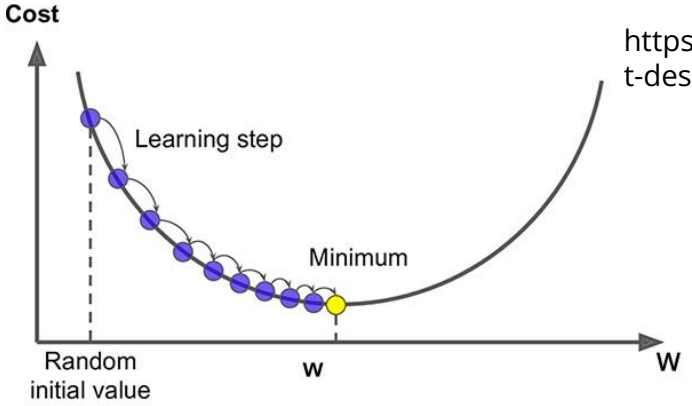
- In very few case you have analytical solutions
- For the rest, you need a numerical method that iteratively makes the error a little smaller at each step (steps can be called **epochs**)

$$\theta_k \to \theta_{k+1} \qquad E(\theta_{k+1}) < E(\theta_k)$$

- Basically all methods are variation of the **gradient descent**
  - For **learning rate** γ small enough, E(θ) descreases

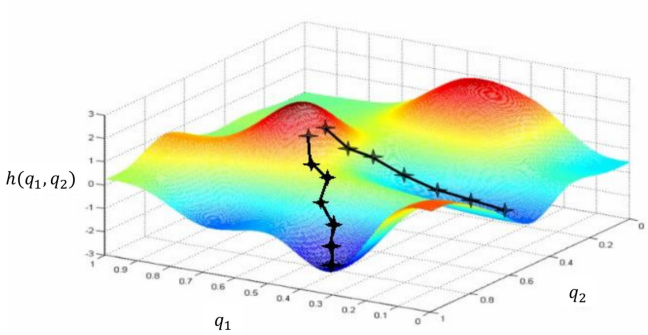
$$\theta_{k+1} = \theta_k - \gamma \nabla E(\theta)$$

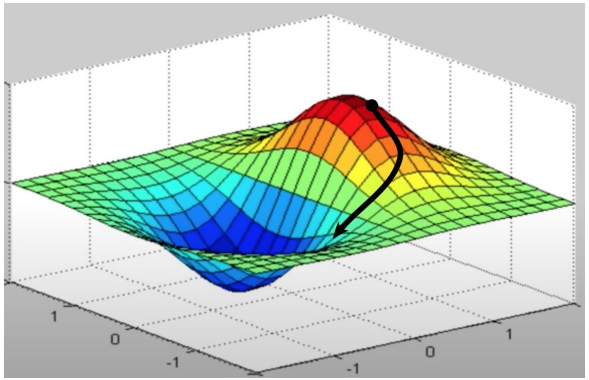
Training your model (visuals)



https://saugatbhattarai.com.np/what-is-gradient-descent-in-machine-learning/

Non-convex Example



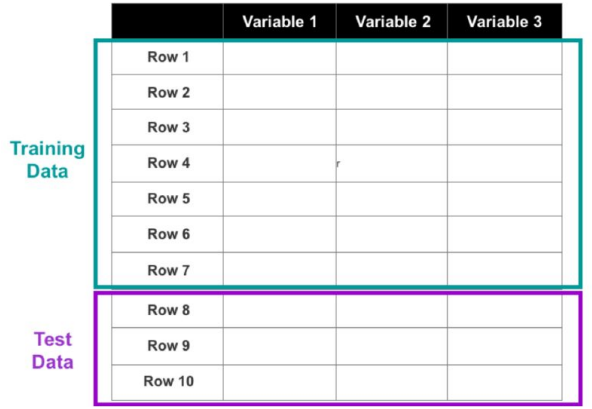


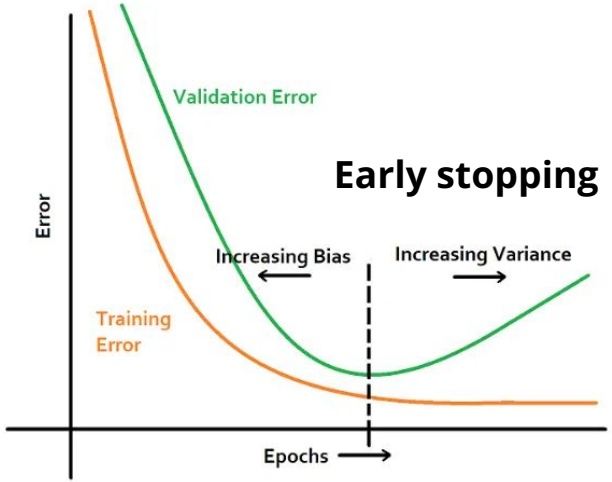
https://shashank-ojha.github.io/ParallelGradientDescent/

https://mriquestions.com/back-propagation.html

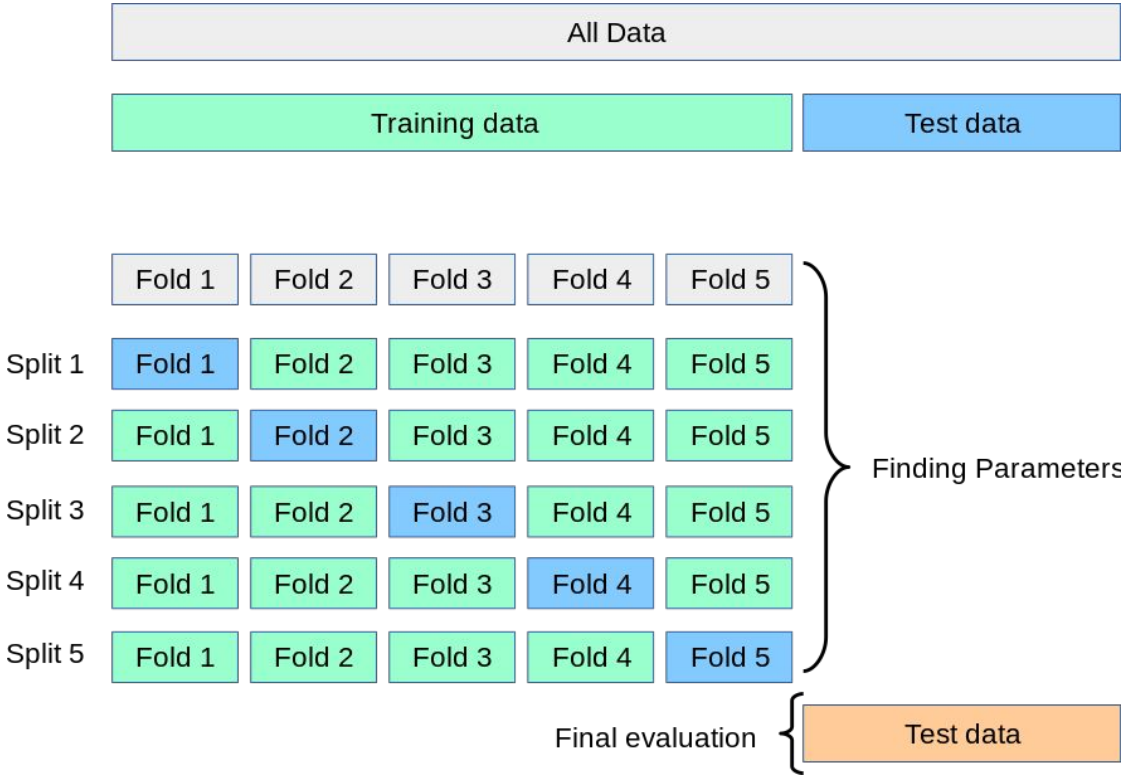
Testing your model

- Training, test, and validation set (~80/10/10)
  - Training: to determine your parameters
  - Validation: **early stopping**, to tune hyperparameters (k in k-nearest neighbors)
  - Test: to check if your model generalizes
- How to choose the sets
  - Basic: randomize the rows in you data table
  - More in general: you use a “hidden” feature (*e.g.*, patient, experiment, time) to segment your data
- Cross-validation (testing on steroids)
  - exhaustive cross validation
  - k-fold validation (k buckets… k patients/esperiments)





K-fold validation



What we'll spend our time on

Rarely mentioned, yet data preparation/exploration is the most important step for the final result, and the most time-demanding of the workflow (even if less "cool")

3 take-home messages…

… and three little stories

- Know your data
  - Understand to the maximum extent possible what they mean, how they are generated, how data points are “interconnected”
  - “Just a guy in a garage” and the Netflix prize - https://www.wired.com/2008/02/mf-netflix/
- Reflect on your task
  - What exactly do I want to predict? Can I make it more general? Are there constraints/prior expectations that I can incorporate?
  - Amazon discriminating against women - https://finance.yahoo.com/news/amazon-reportedly-killed-ai-recruitment-100042269.html
- Test your model carefully
  - Ask yourself: what generalization really means in my task? Besides performance, what should you check in the answers from your model? Are you sure your model has not seen any of the test data?
  - Identify suicidal ideation from fMRI - https://twitter.com/KordingLab/status/1644143576182579200

See you tomorrow!