

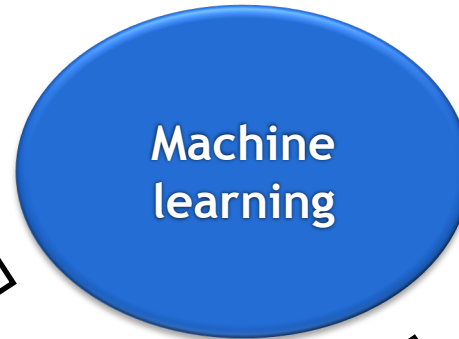
# About data, models and other magic: a brief introduction to machine learning

# WHAT IS MACHINE LEARNING?

# Machine learning components

- Optimise (=learn)
- model parameters
- to minimise a loss function
- using example data

**D.A.T.A!**



Learning algorithm  
(optimization)



Parametric  
mathematical model

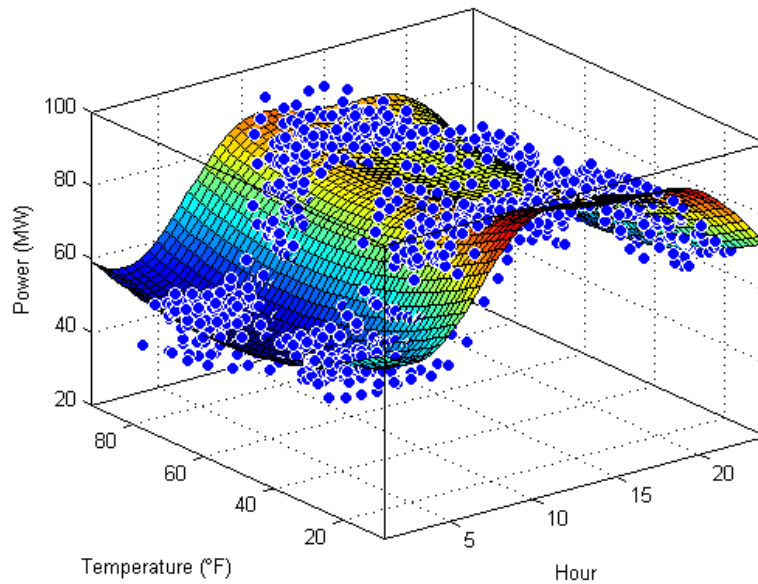


Cost function  
(loss function)

# Types of machine learning

- **Supervised learning:**
  - ◆ Approximate functional relationship between **input (features)** and **desired output (labels)**
  - ◆ Data contains **examples** of input and desired output
- **Unsupervised learning:**
  - ◆ Learn «what normally occurs»:  
model the «structure» or probability density profile of data
  - ◆ Data contains examples of feature combinations (no labels)
- **Reinforcement learning:**
  - ◆ Learn strategy to maximize future «reward»:  
desired output not known/enforced, reward has no closed-form mathematical relation to model output
  - ◆ Data contains no desired outputs

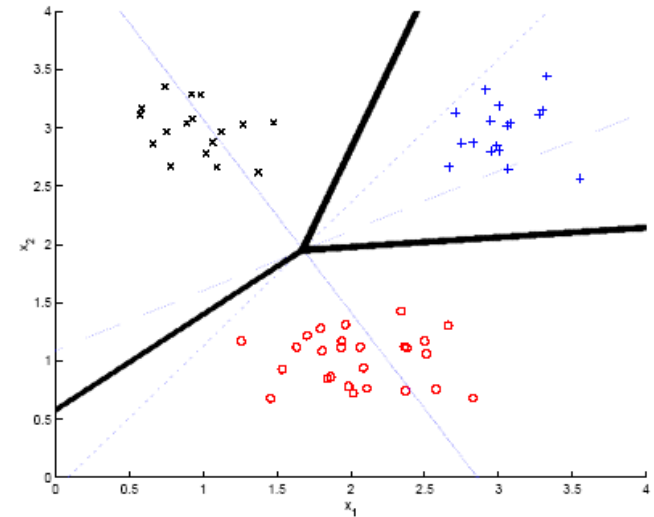
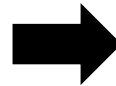
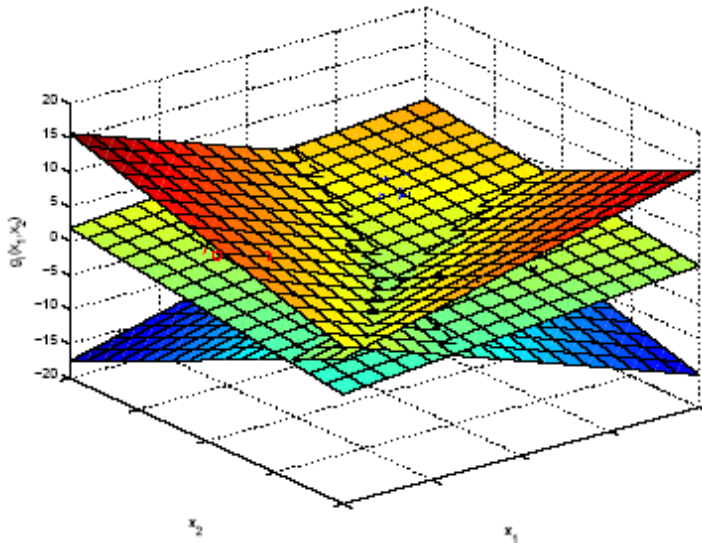
# Types of supervised ML: regression



**Regression:**  
 labels are continuous

**Example (2 features):**  
 "model" is a surface in 3D

# Types of supervised ML: classification



**Classification:**  
labels are discrete (“classes”)

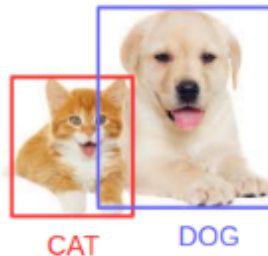
**Example (2 features):**  
“model” gives a surface for each class,  
intersection boundaries between surfaces define classification boundaries

# Complex supervised ML: images



## Image classification:

- Single label: classes are mutually exclusive, label is one category from a set
- Multi-label: Image can belong to multiple classes (label is a vector of 0's and 1's)



## Object detection:

- Find all objects from each class
- Mixture of classification (which class) and regression (bounding box)



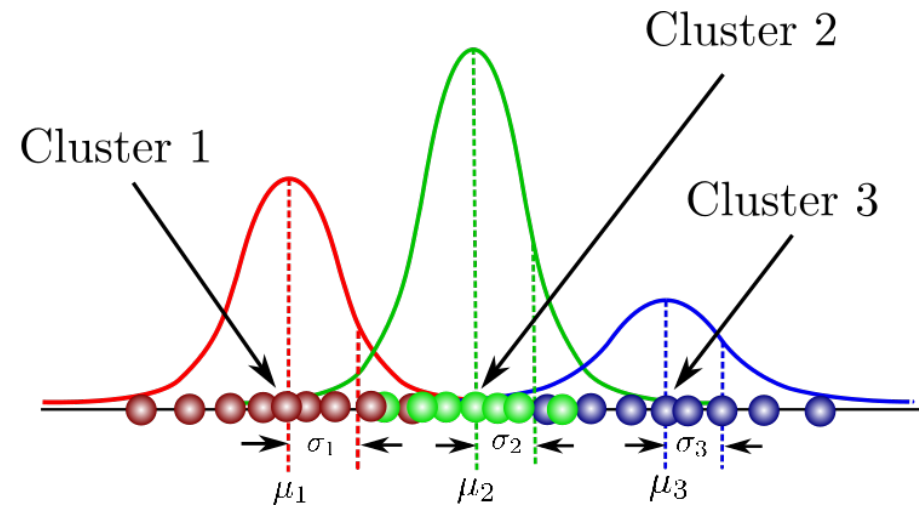
## Image segmentation:

- Find all pixels belonging to (objects of) certain categories
- = classification of each pixel

# Unsupervised ML example: Gaussian mixtures

## Assumptions:

- Data contains  $k$  subgroups
- Each subgroup occurs with a certain probability
- Features in each subgroup follow joint Gaussian distribution



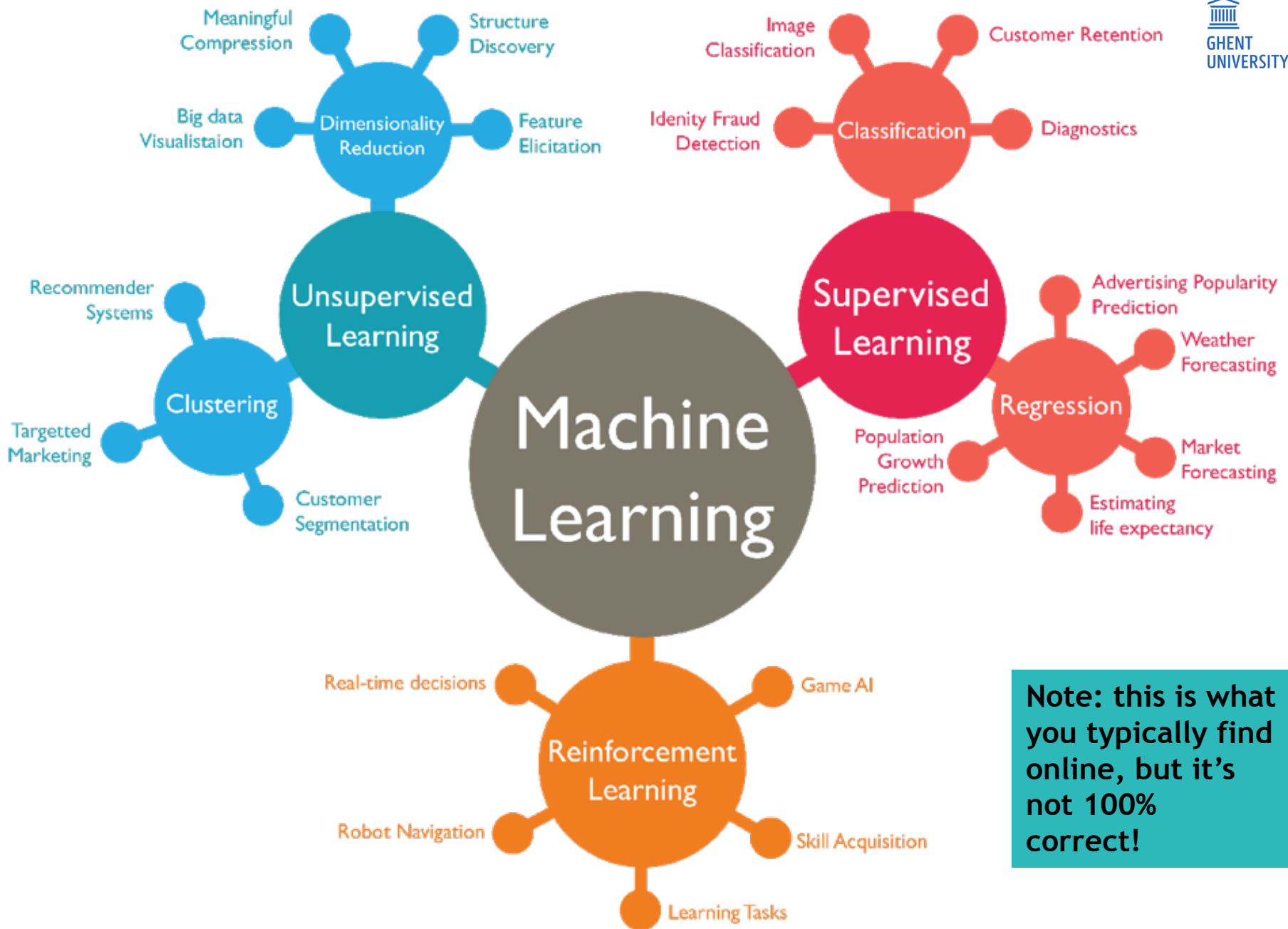
## Model training:

- Subgroup probabilities & Gaussian parameters that best fit the data

## Properties:

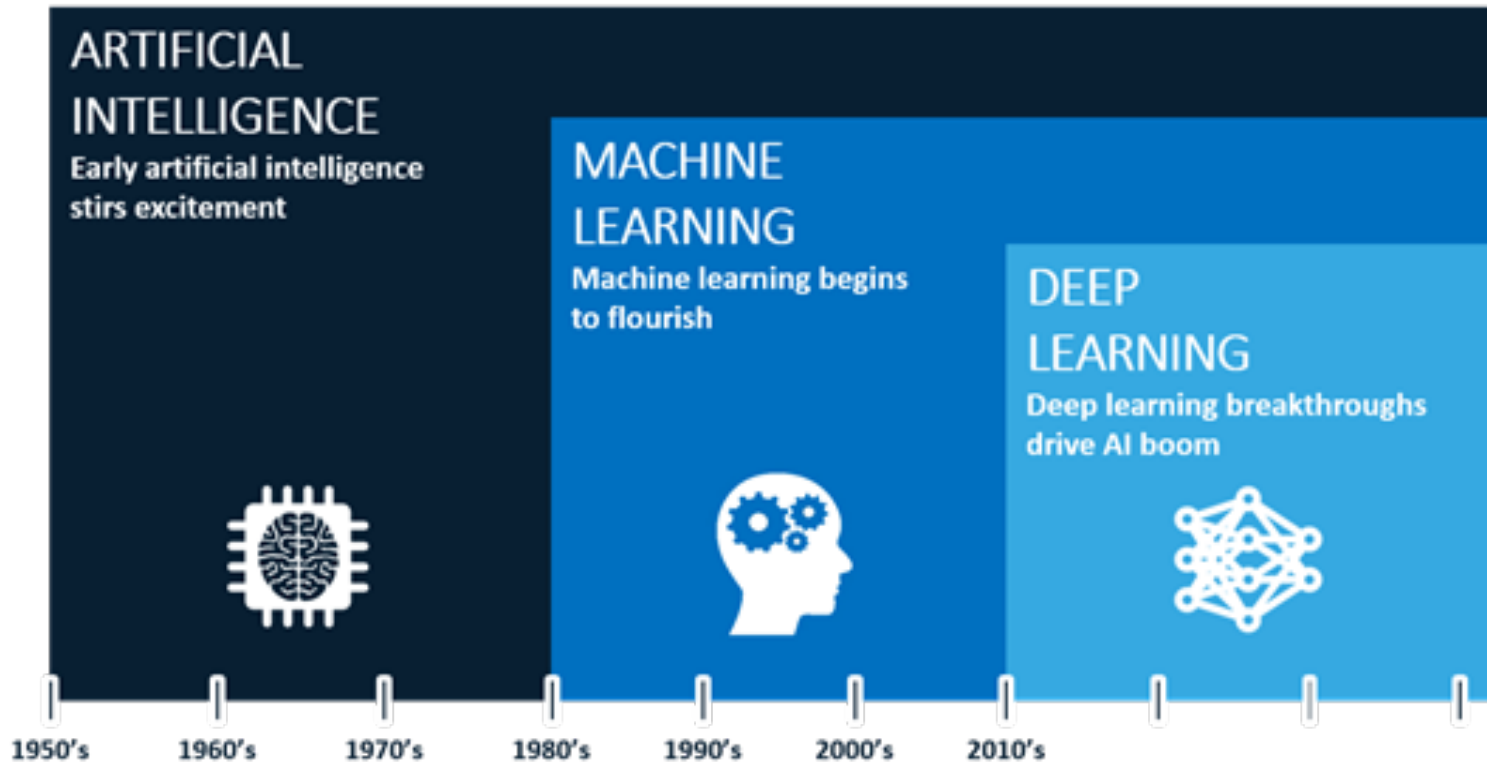
- Model only uses the data (no labels)
- Generative model:  
new “realistic” data can be generated by sampling from the model





**Note: this is what you typically find online, but it's not 100% correct!**

# AI vs machine learning (vs deep learning)



**AI:** Intelligence demonstrated by machines rather than humans or animals.

**ML:** Giving computers the skills to learn without explicit programming

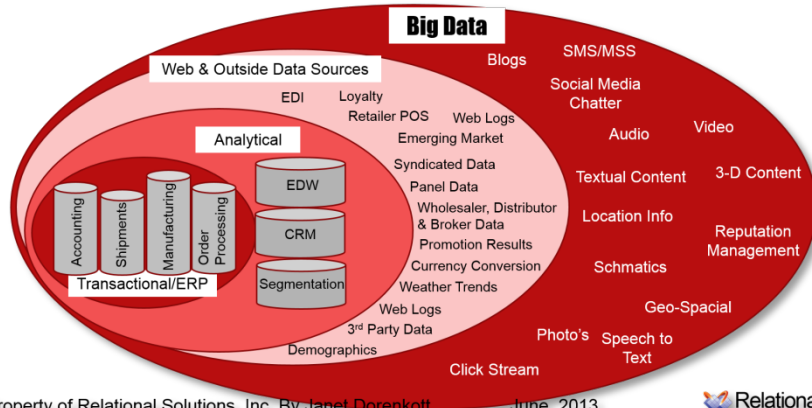
**DL:** Is an ML subset, examining algorithms that learn and improve on their own.

Not shown on the timeline:  
the AI winters!

# Why now?

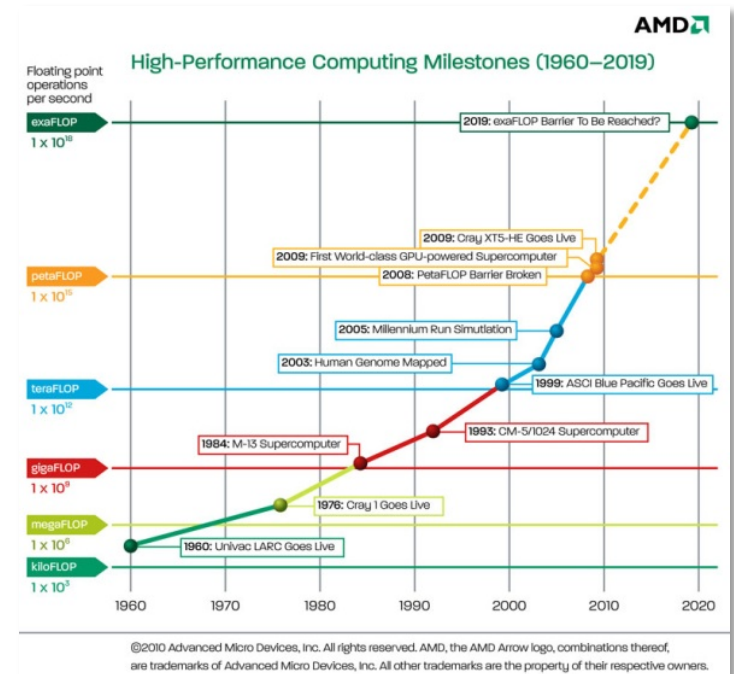
- (Machine learning has been around for more than half a century!)
- But: powerful machine learning needs LOTS of data and serious computing power!

## THE BIG DATA EXPLOSION!

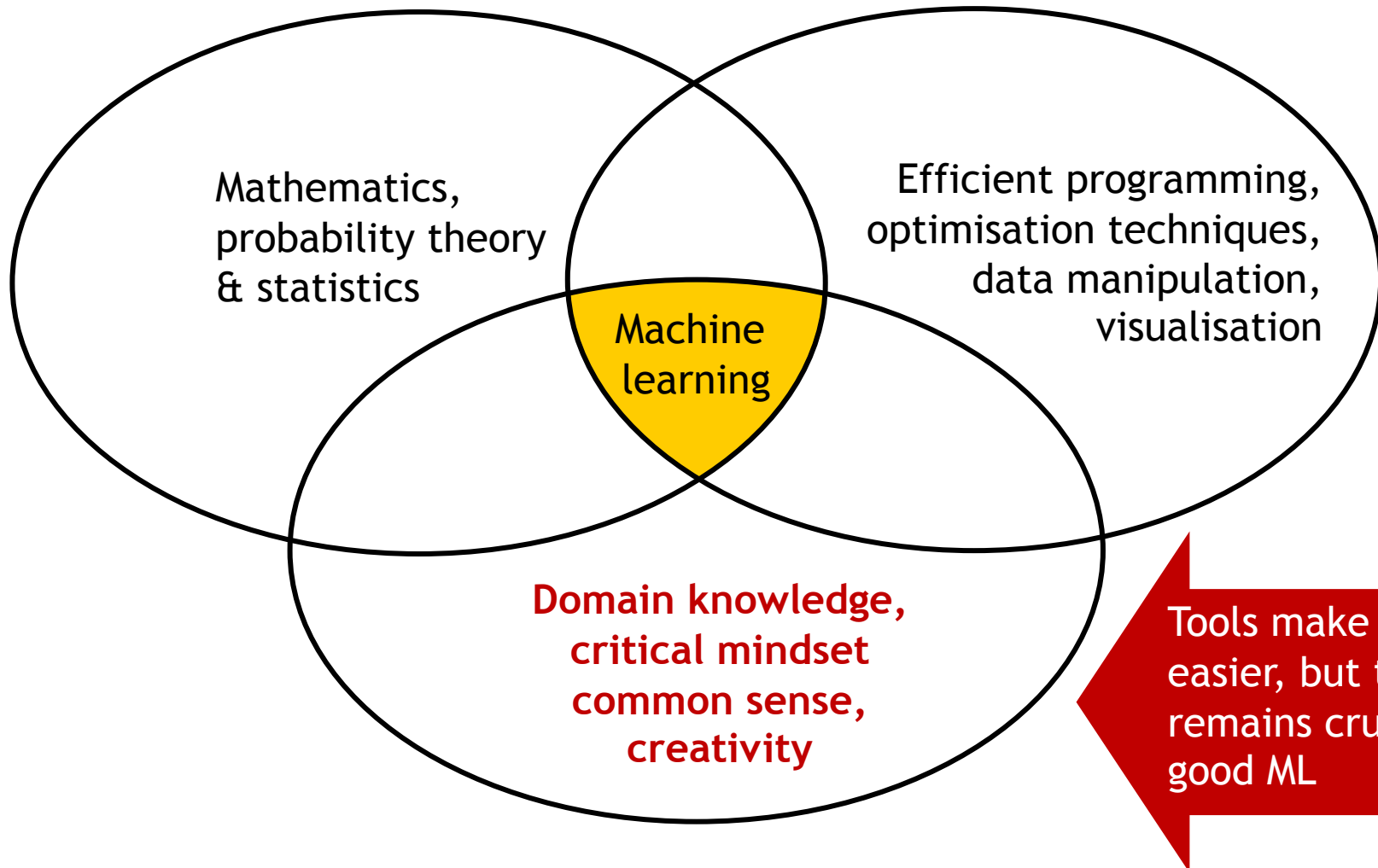


Property of Relational Solutions, Inc. By Janet Dorenkott June, 2013,

 Relational Solutions



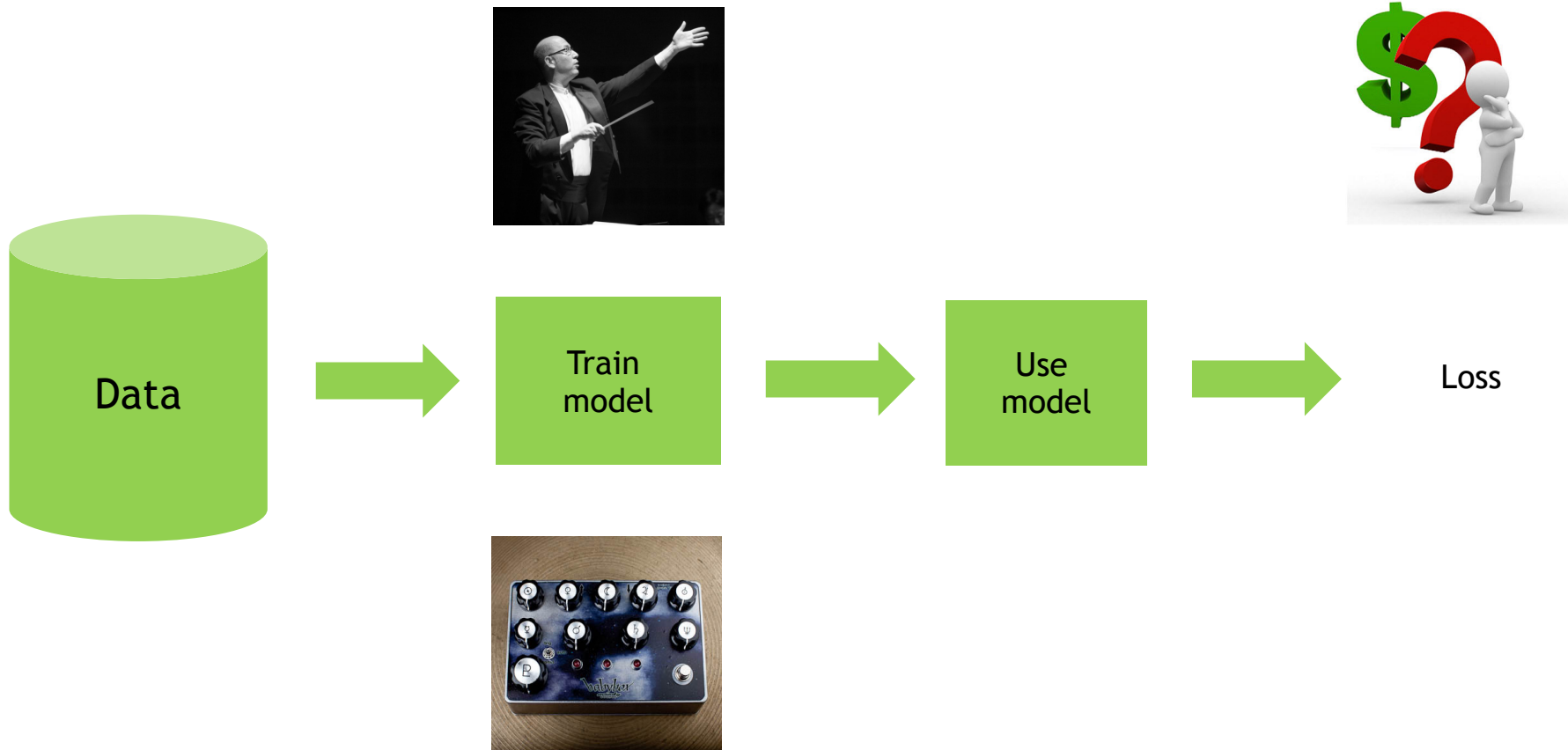
# What skills do you need?



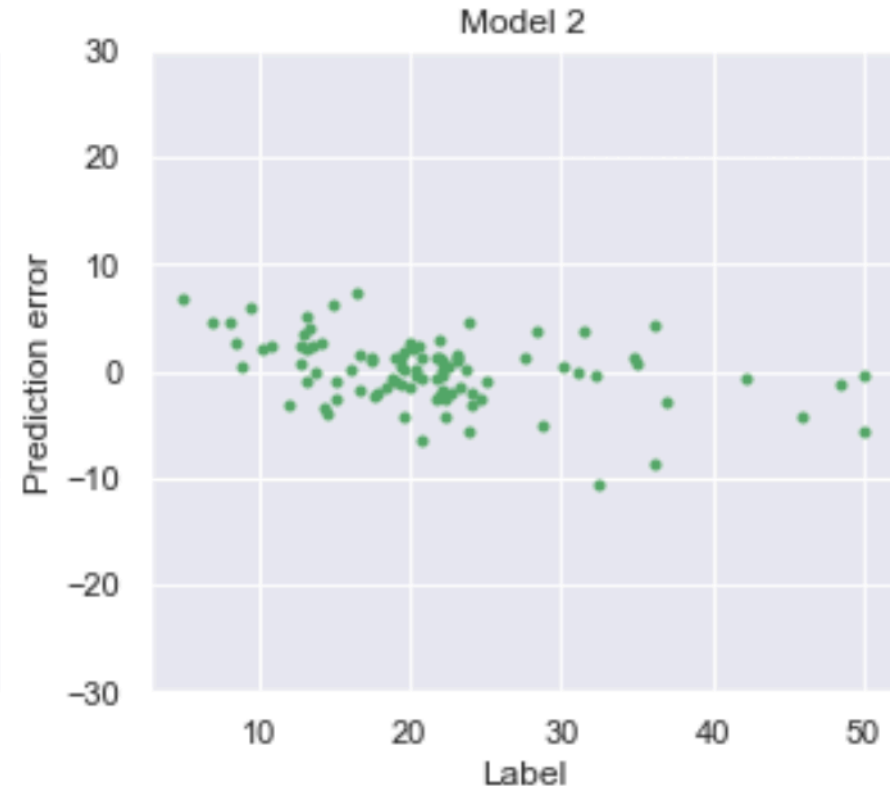
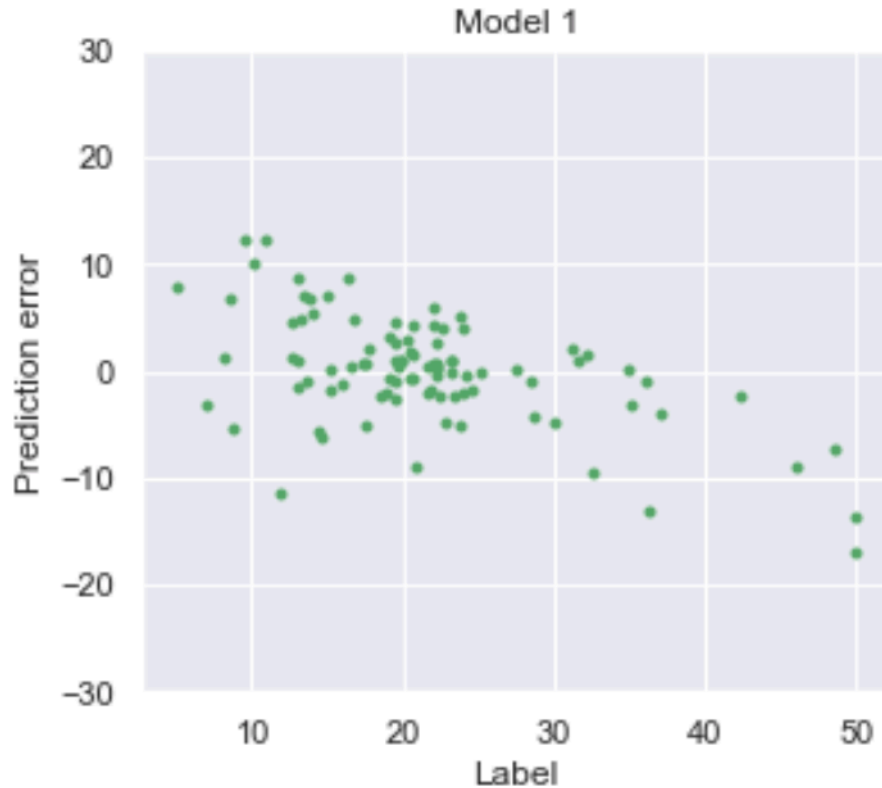
# THE ULTIMATE GOAL: GENERALISATION

# What is a good model?

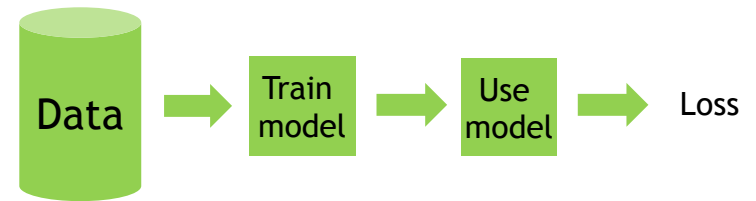
What we know thus far ...



# Comparing models, based on loss

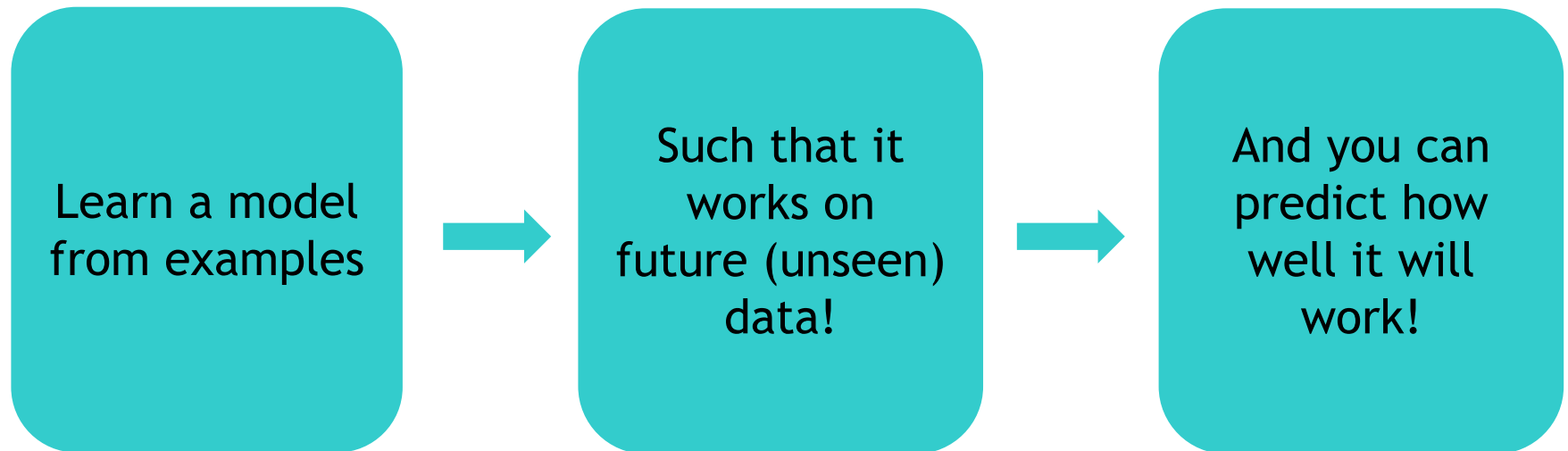


Model 1 SSE = 26.67  
 Model 2 SSE = 9.83



**Initial conclusion:** model 2 is the better model?

# ML is more than just optimisation!

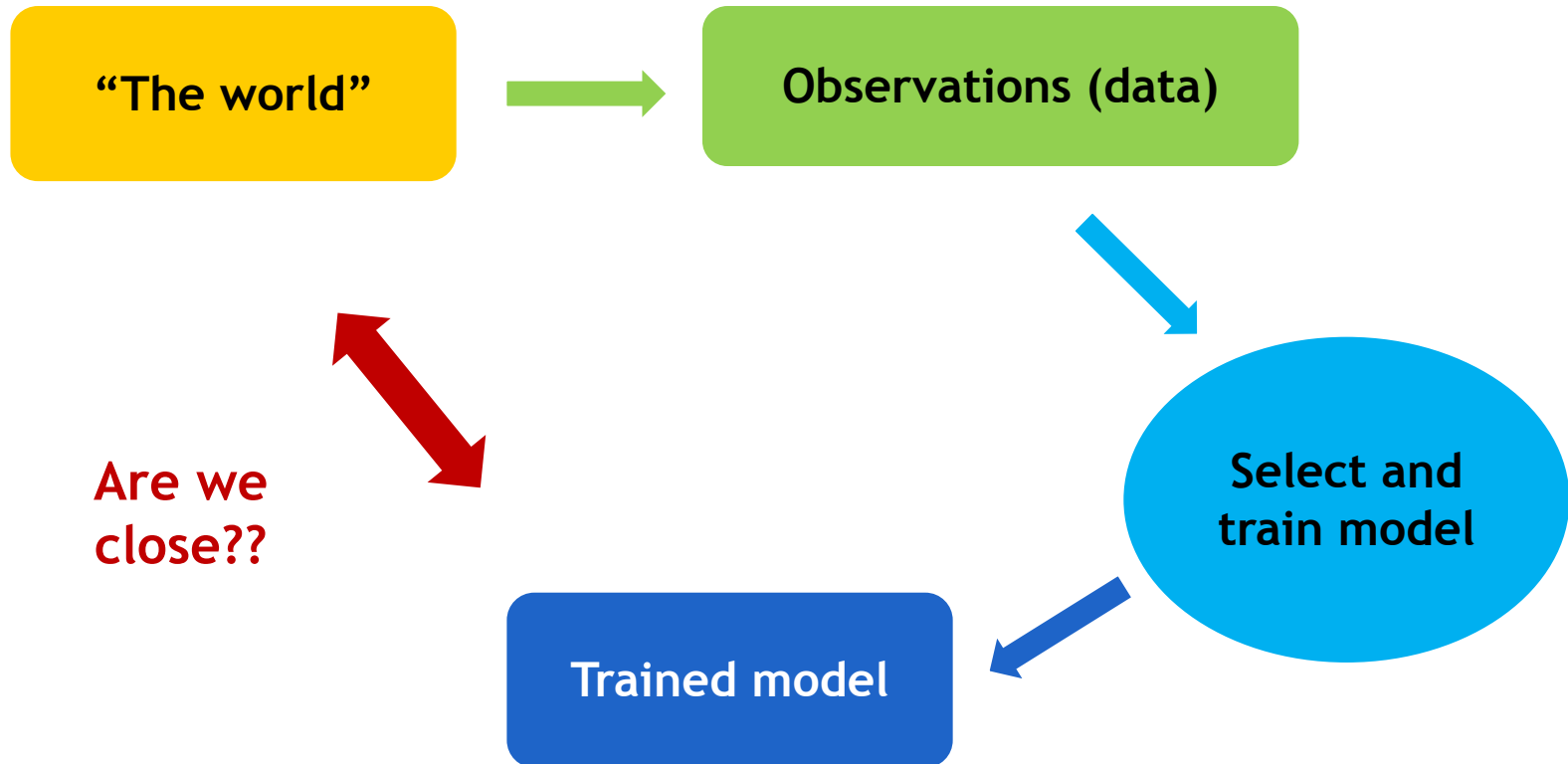




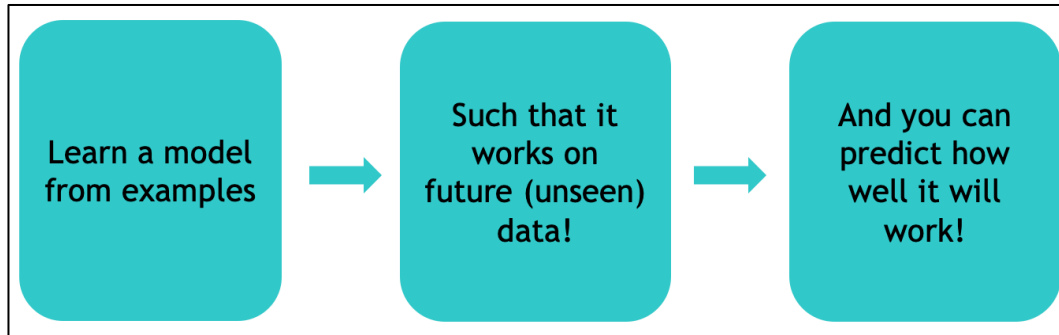
# The goal is **GENERALISATION!!!**

- Machine learning learns from examples:
  - ◆ A finite set of observations (**the observed features**) and what the model should output for them (**the labels**)
  - ◆ For each model we make, we can **only** measure how well it performs on any example from this set
  - ◆ And from all possible models, we want to choose the best
  - ◆ **But ‘the best’ means a model that performs well for data that we haven’t seen yet**
  - ◆ And the problem with a finite set is ... that **it’s finite**

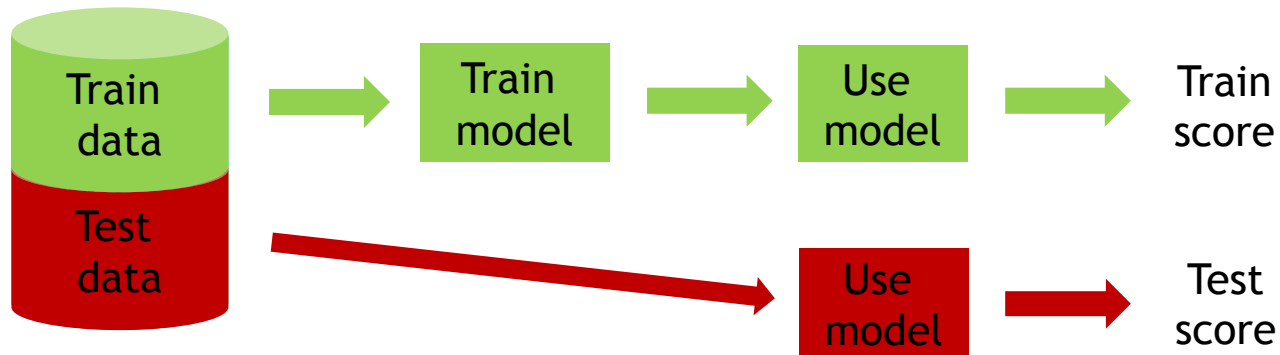
# How to evaluate generalization?



*Based on Abu-Mostafa's MOOC "Learning from Data"*

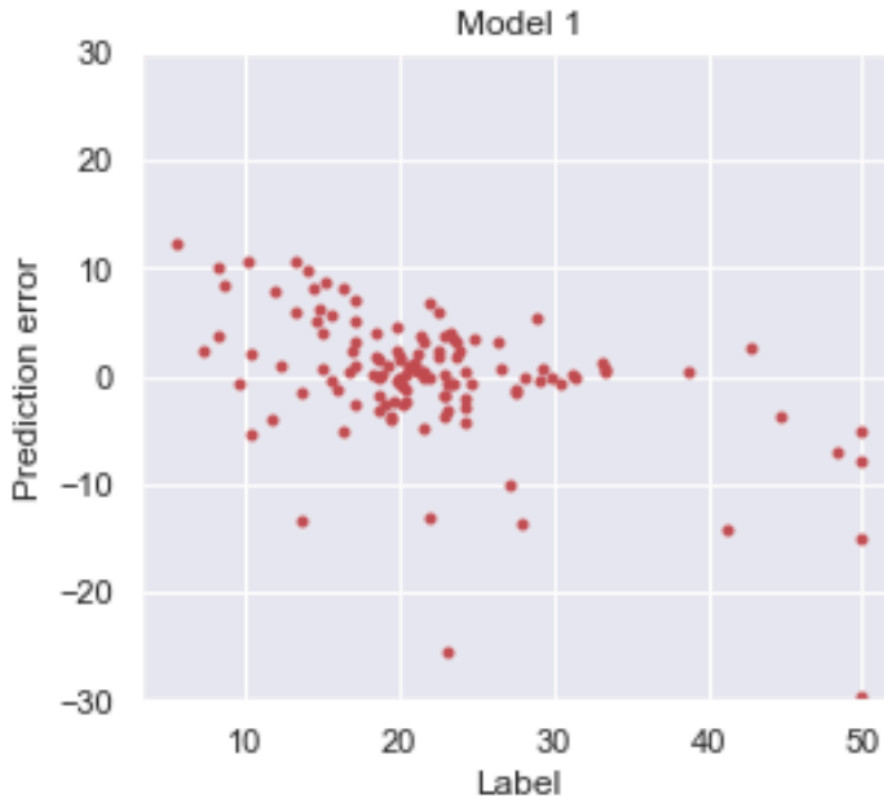


Let's not trust our models ... and test them on new data!



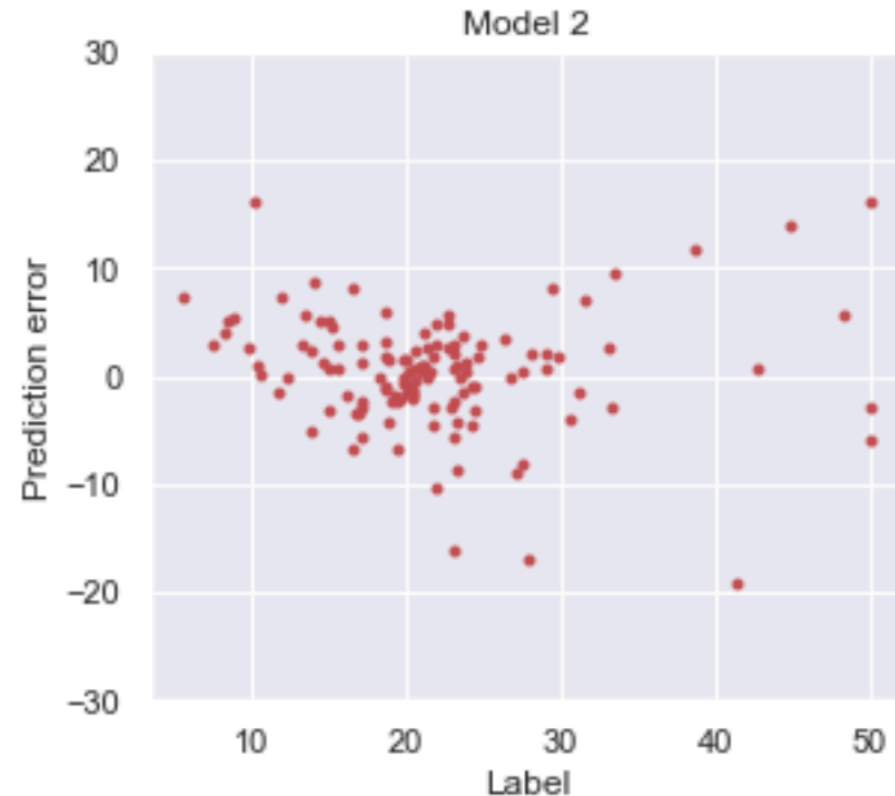
Ideally: both scores should be similar!

# Comparing models - revisited



Model 1:  
 train SSE = 26.67  
 test SSE = 43.12

↔ **Small gap!**



Model 2:  
 train SSE = 9.83  
 test SSE = 52.02

↔ **Big gap!**

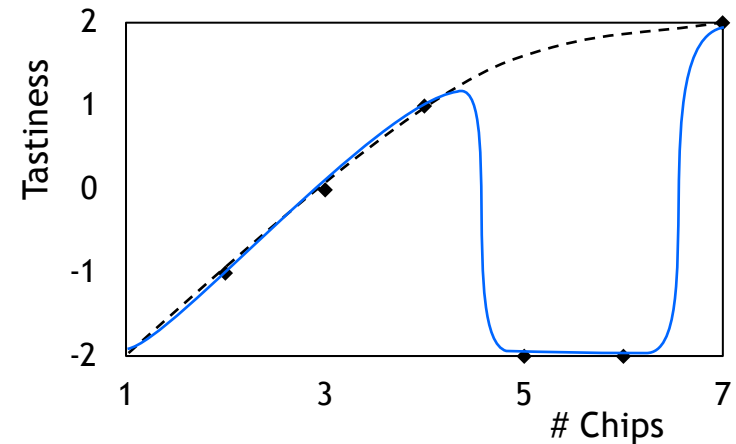
**Updated conclusion: model 1 is the better model!**

**WHERE DOES THIS GAP COME FROM?**

# Bobby's cookie experiment



	# Chips	Tastiness
Sunday 1	7	Awesome
Sunday 2	4	Good
Sunday 3	2	Bad
Sunday 4	5	Terrible
Sunday 5	3	Average
Sunday 6	6	Terrible

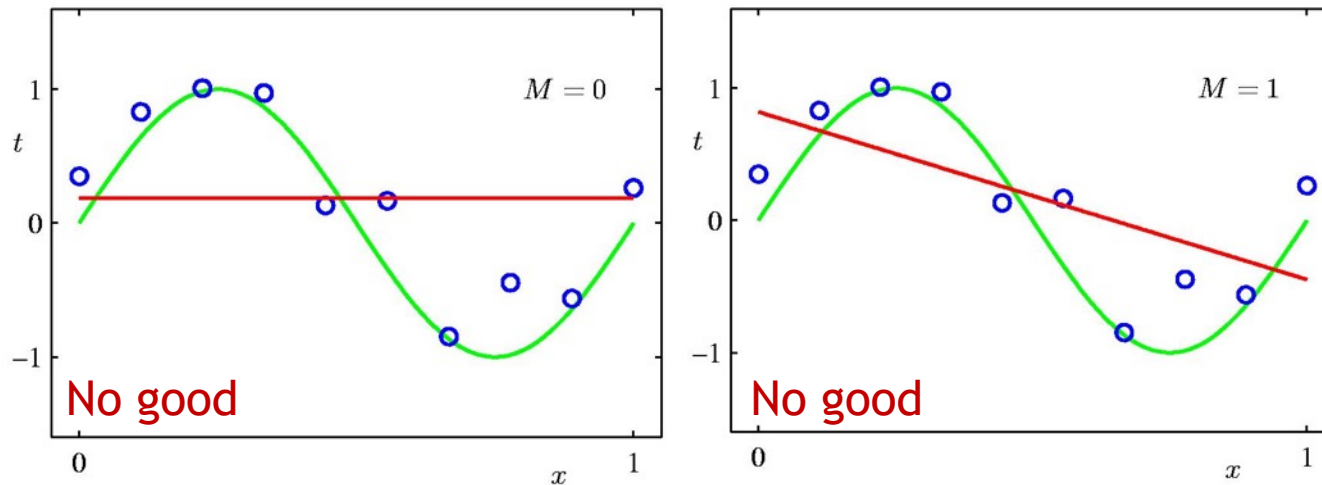


## Data is never perfect:

- Some information may be missing from your features
- Your features may contain noise
- Your labels may contain noise

# Simple and complex models

Best fit to training data, for different model complexities

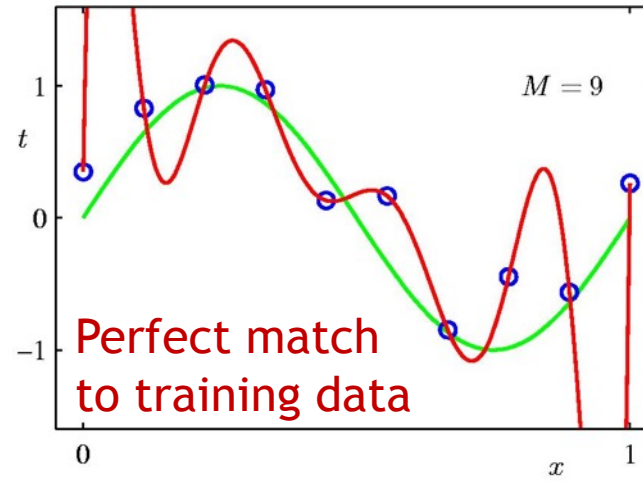
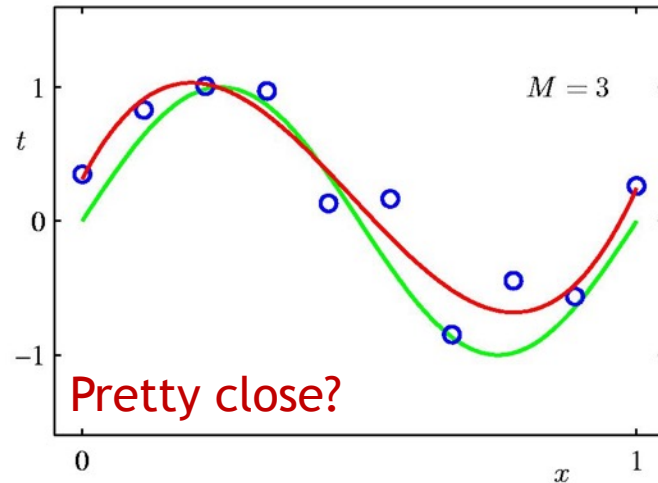


These models are **underfitting**:

- they are not complex enough
- they can not approximate the training data
- they can not approximate the ground truth in any decent way!

# Simple and complex models

Best fit to training data, for different model complexities

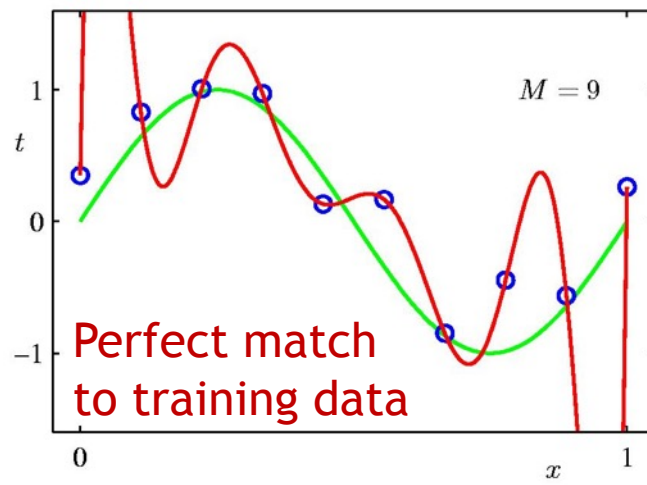
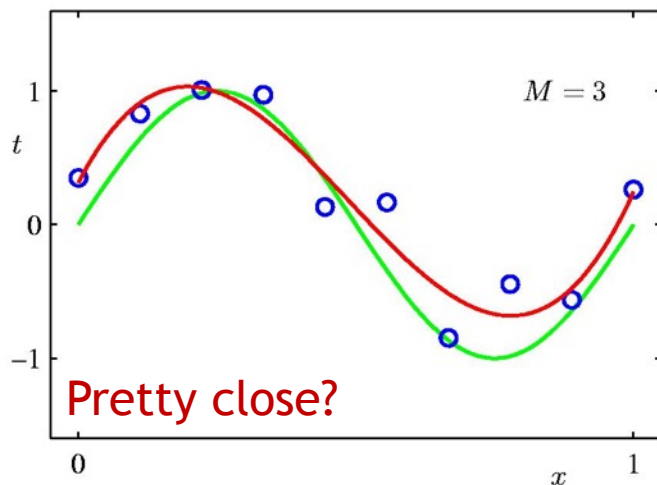
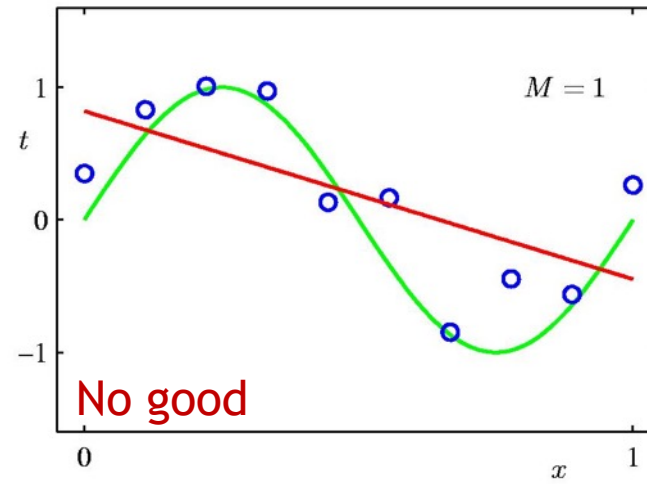
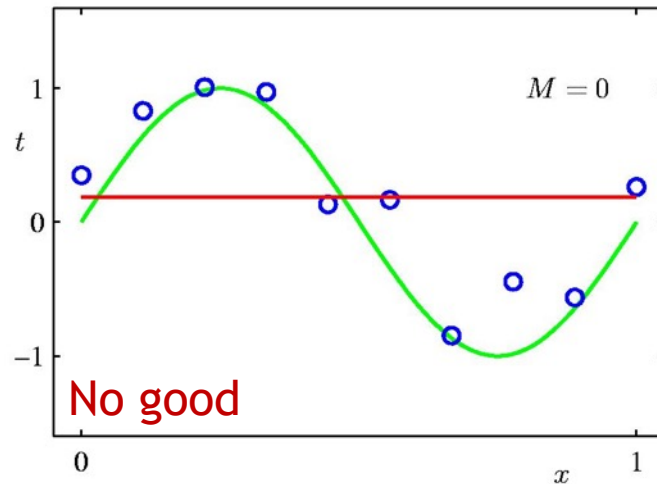


- Model on the right: **overfitting!**
- Good model: constrain the complexity by **regularization!**
- **Tune complexity of (potentially) powerful models with hyperparameters**

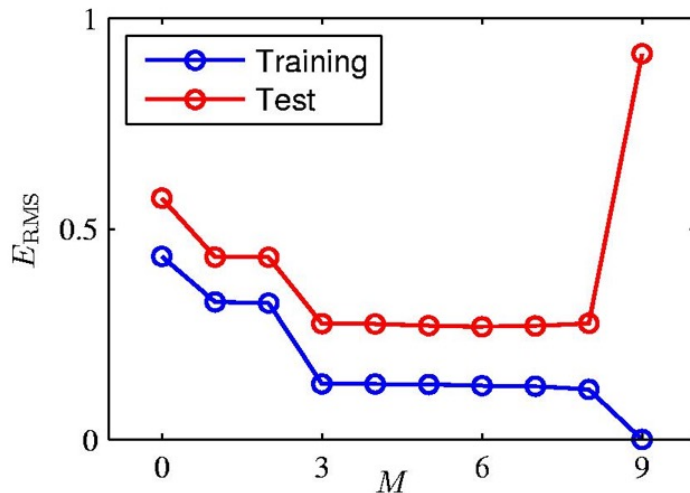
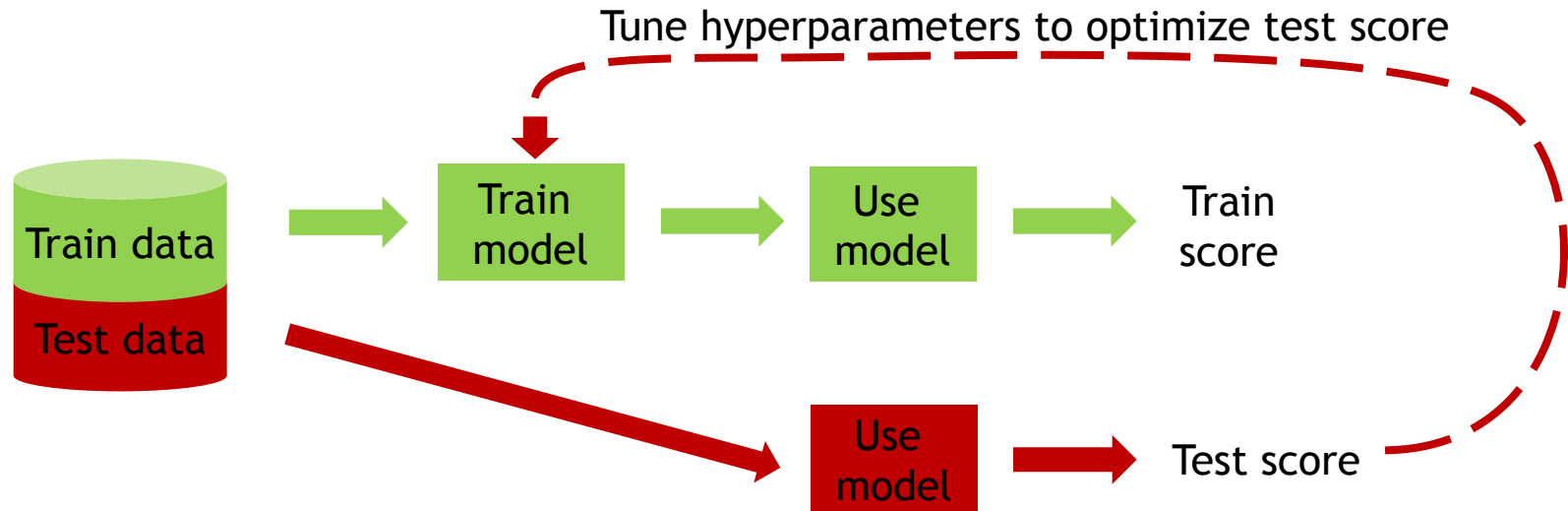


# Which hyperparameters to choose??

In real life we don't know the ground truth!



# Choosing the best model

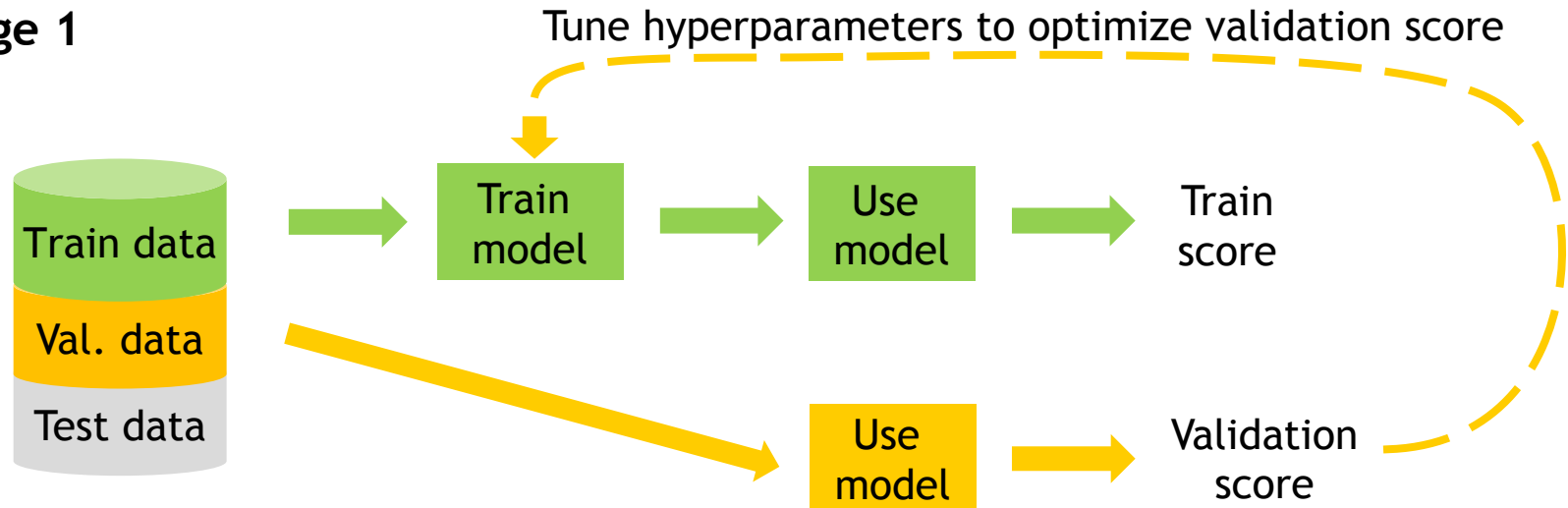


Choose the model that performs best on unseen data

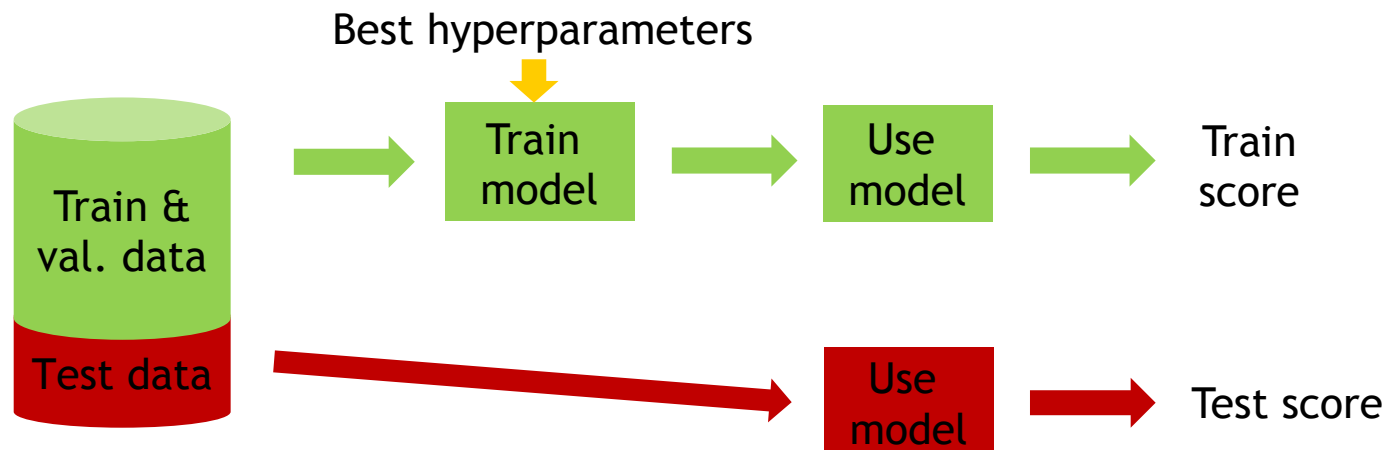
But now, your test data is “spoiled”: test scores will likely be optimistic!

# Training, validation and testing

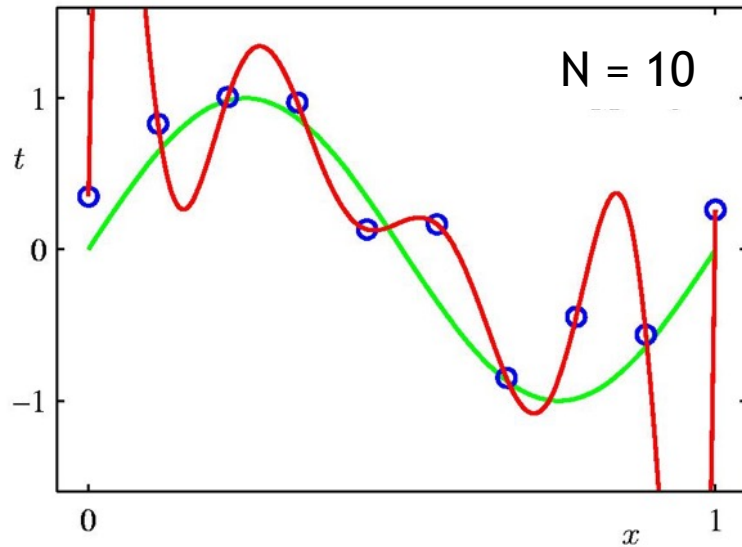
## Stage 1



## Stage 2

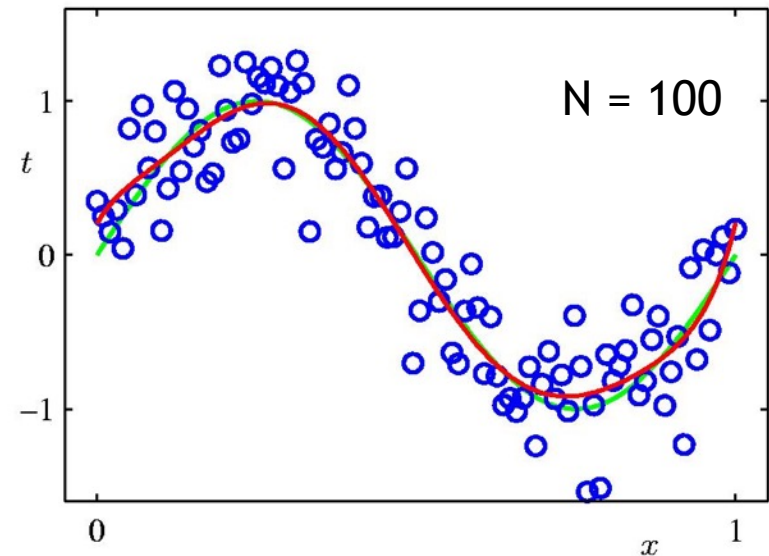
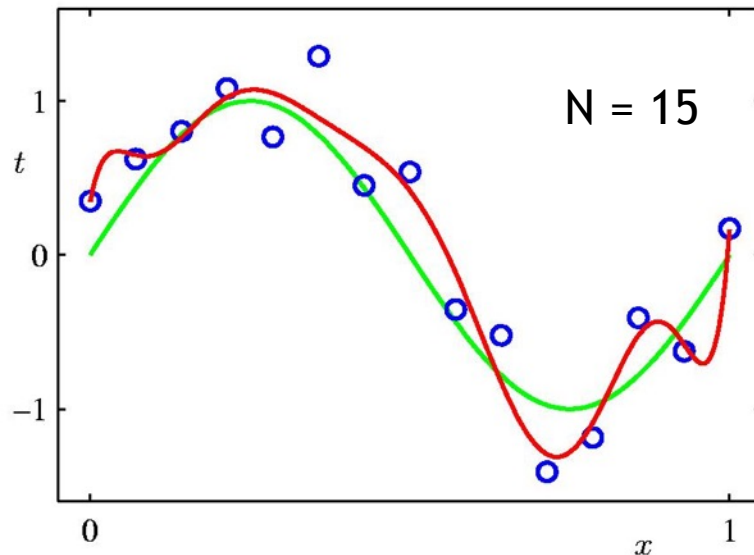


# Get more data if you can!!

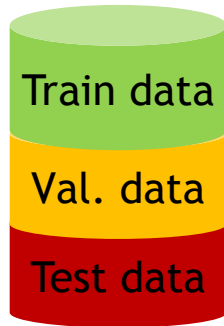


3 times the same model,  
trained with different  
amounts of data

**Overfitting will decrease  
if you get more data!**



# Training, validation and testing



## Train set:

- Must be large enough to avoid overfitting
- Must be **representative** for data your model will be used on (features & labels)
- Must be representative for what you want your model to do

## Validation and test sets:

- Must fulfill same conditions as train set
- & Data in each set must be independent
- In order to give an accurate estimate of model quality on unseen data

# PROBLEMS WITH DATA

# Data issues

- Not enough data
  - ◆ to learn the desired relation
  - ◆ AND average out all irrelevant effects
- Incomplete information
  - ◆ E.g. Cookie example:  
some information that determines the label is missing
- Data is inaccurate
  - ◆ e.g. measurement noise
  - ◆ e.g. mistakes in the labels
- Data bias
  - ◆ this part of the talk!

# Some innocent examples ...



A close up of a lush green field  
Tags: grass, field, sheep, standing, rainbow, man

<http://aiweirdness.com/post/171451900302/do-neural-nets-dream-of-electric-sheep>



# Some innocent examples ...

<http://aiweirdness.com/post/171451900302/do-neural-nets-dream-of-electric-sheep>



Cloudy hills  
=  
Probably sheep  
around

A herd of sheep grazing on a lush green hillside  
Tags: grazing, sheep, mountain, cattle, horse

# Some innocent examples ...



Most sheep in the data occur in “sheepy” landscapes  
Model has learned connection between landscape and sheep

Overfitting on **context** in training data!

# Some innocent examples ...



Too many giraffes  
in the data set!!

A close up of a hillside next to a rocky hill  
Tags: hillside, grazing, sheep, giraffe, herd

<http://aiweirdness.com/post/171451900302/do-neural-nets-dream-of-electric-sheep>

# Some innocent examples ...



A group of orange flowers in a field  
Image credit: Richard Leeming @RM\_Leeming - CC-BY license

<http://aiweirdness.com/post/171451900302/do-neural-nets-dream-of-electric-sheep>

# Some innocent examples ...



Decision based on  
context  
(and giraffes again)

NeuralTalk2: A flock of birds flying in the air  
Microsoft Azure: A group of giraffe standing next to a tree  
Image: Fred Dunn, <https://www.flickr.com/photos/gratapictures> - CC-BY-NC

<http://aiweirdness.com/post/171451900302/do-neural-nets-dream-of-electric-sheep>

# Automated Inference on Criminality using Face Images

Xiaolin Wu

Shanghai Jiao Tong University

xwu510@gmail.com

Xi Zhang

Shanghai Jiao Tong University

zhangxi\_19930818@sjtu.edu.cn

Distinguish between criminals and non-criminals, based on images from government data bases



(a) Three samples in criminal ID photo set  $S_c$ .

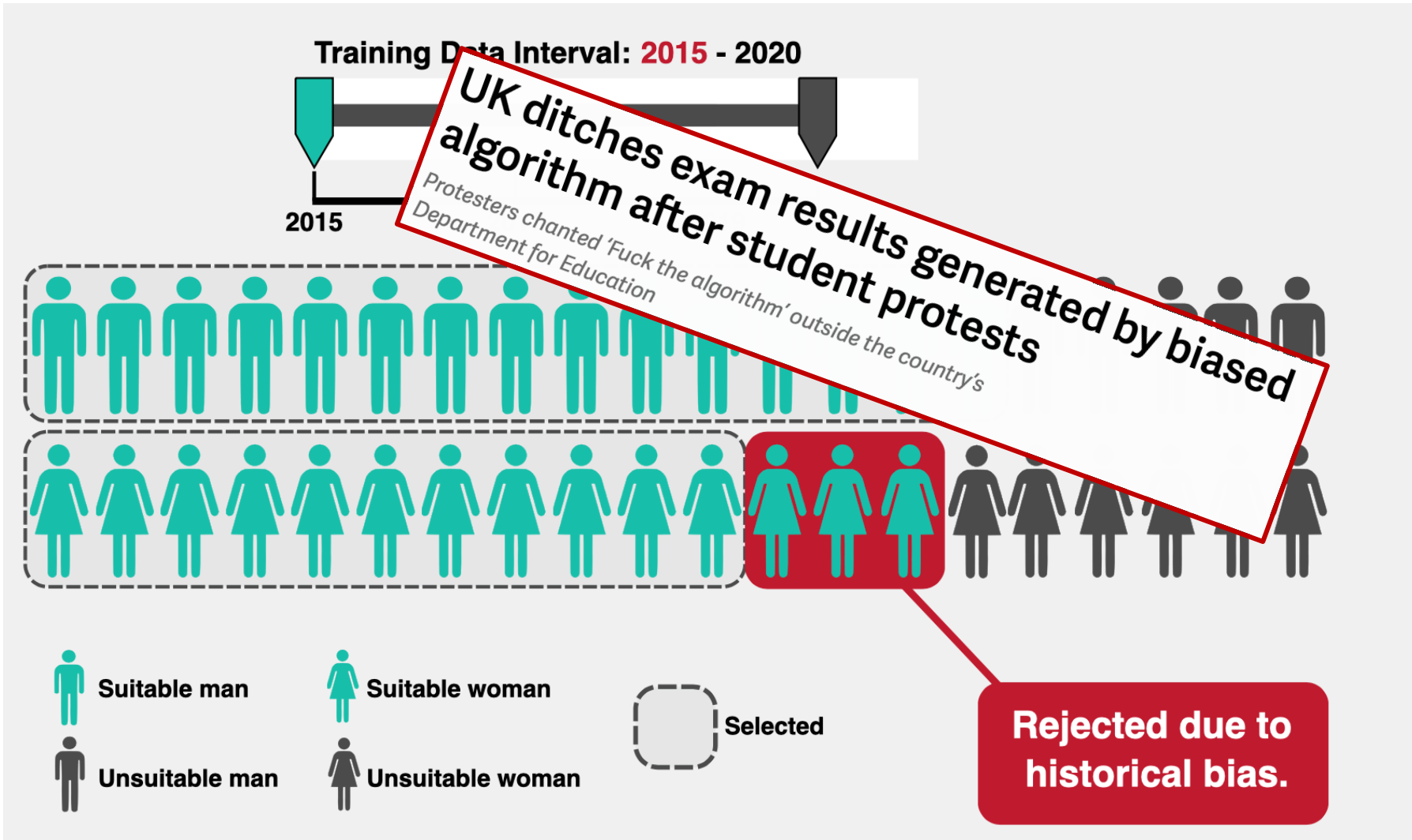


(b) Three samples in non-criminal ID photo set  $S_n$ .

**Model concluded that angle of mouth corners was crucial determining factor!**

Sounds stupid, but similar mistakes happen all the time!

# Impact of historical bias



# Image Representations Learned With Unsupervised Pre-Training Contain Human-like Biases

Ryan Steed

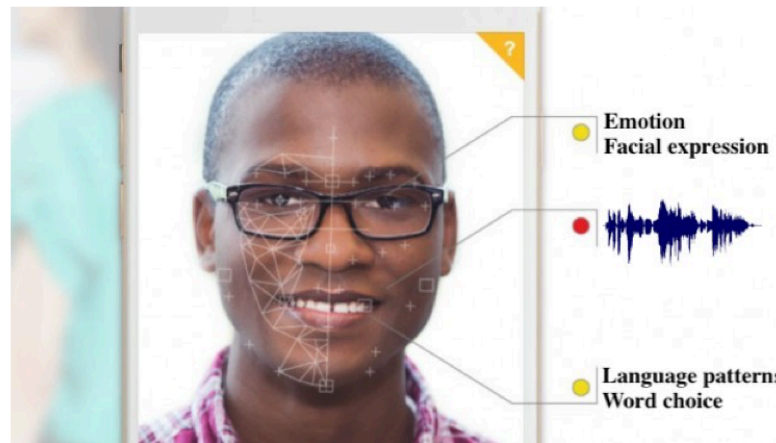
ryansteed@cmu.edu

Carnegie Mellon University  
Pittsburgh, Pennsylvania, USA

Aylin Caliskan

aylin@gwu.edu

George Washington University  
Washington, District of Columbia, USA



arXiv:2010.15052v3

**Figure 1: Unilever using AI-powered job candidate assessment tool HireVue [35].**

## “Historical bias”:

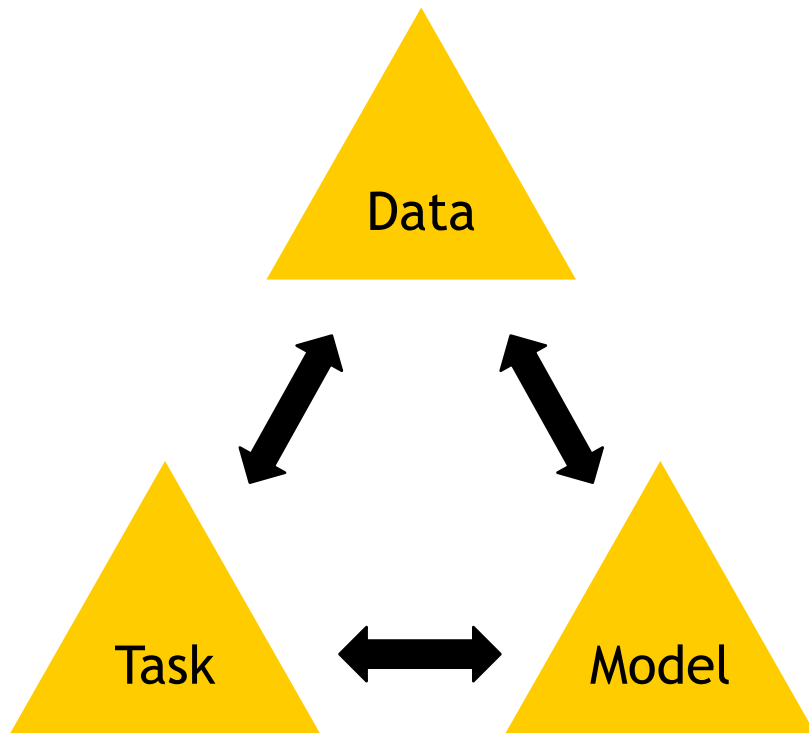
Models learn to mimic the patterns in the data

If these patterns are racist or discriminatory, the model will pick them up



# THE INTERPLAY BETWEEN MODEL, TASK AND DATA

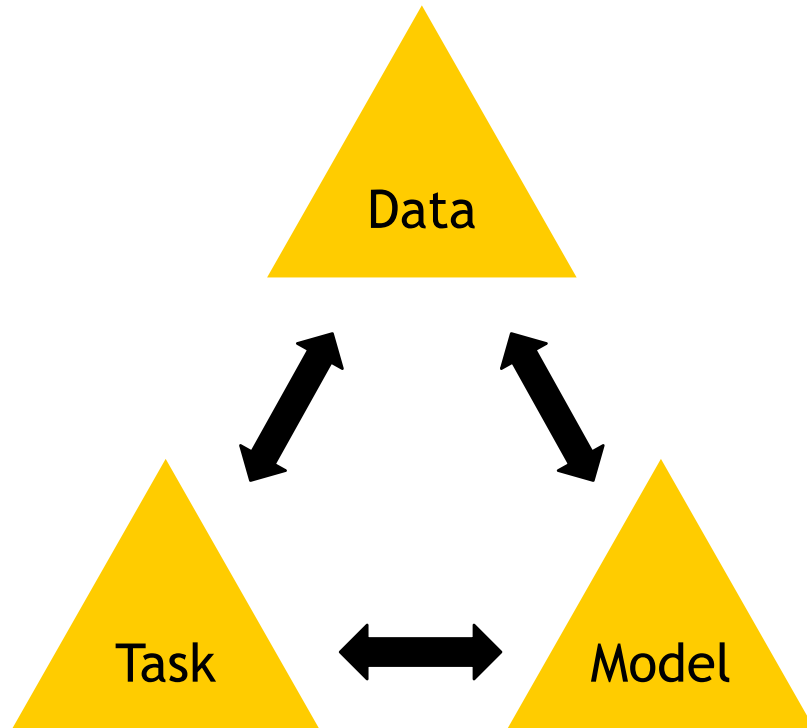
# Which model for my task?



- **Variability in the data:**  
model needs to be able to “average out” what is not important
- **Complex (nonlinear) relation:**  
model needs to be able to mimic complex (highly nonlinear) functions
- **BUT:**  
More complex models  
“overfit faster”  
OR “need more data to avoid overfitting”

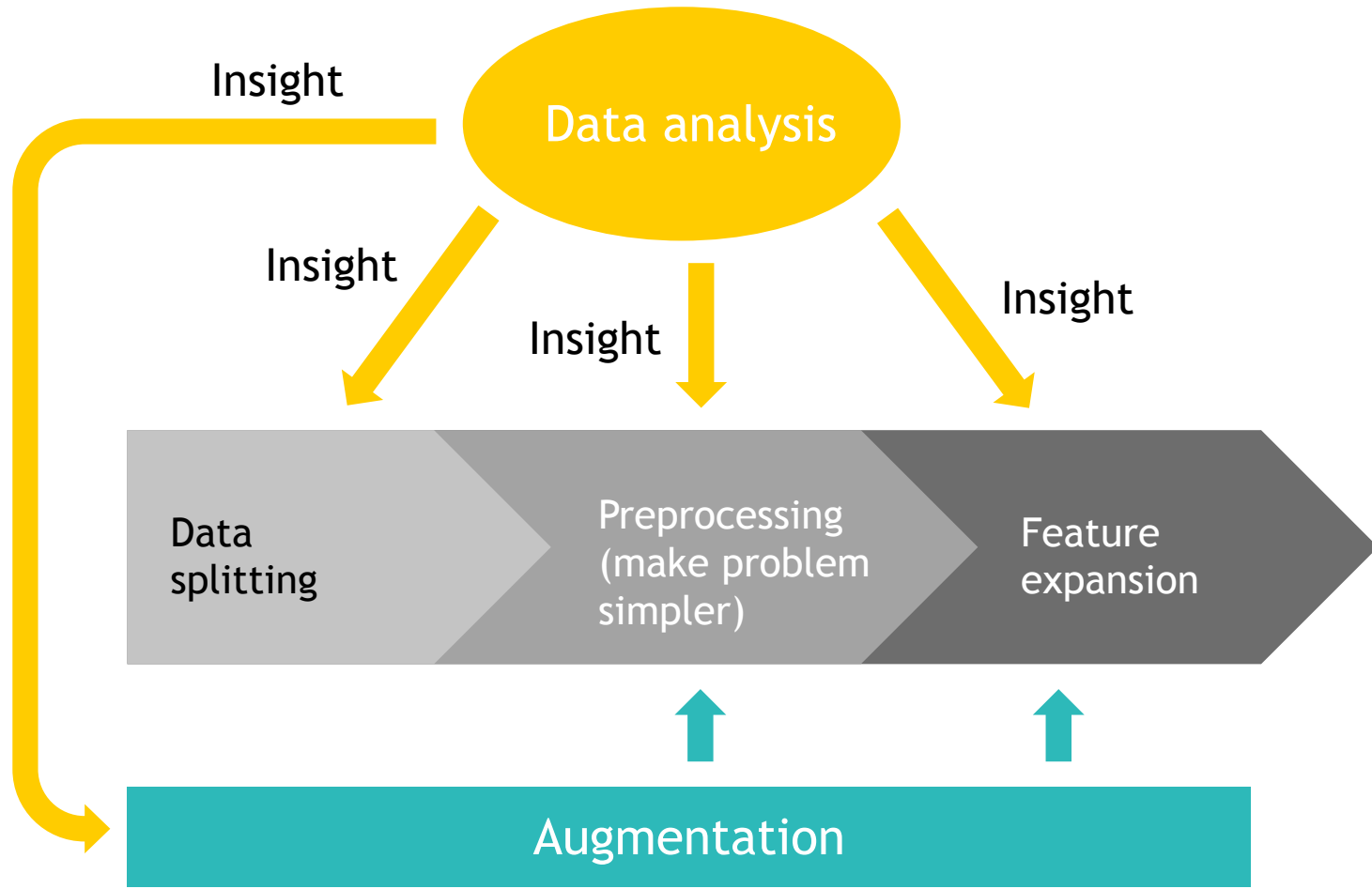
**Identify the least complex model that is capable of solving the task**

# Everything is connected!



- **Think carefully about how to set up your experiment**
- **Try to make your task easier:** data cleaning, preprocessing, possibly feature engineering
- **Try to have a good baseline:** how far do I get with a simple model?

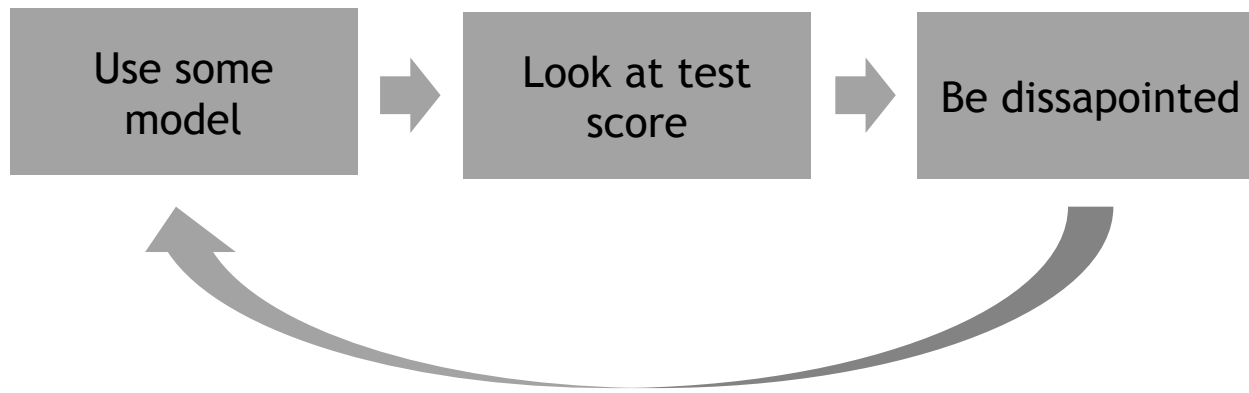
# Adapting your data to task and model



# GETTING STARTED

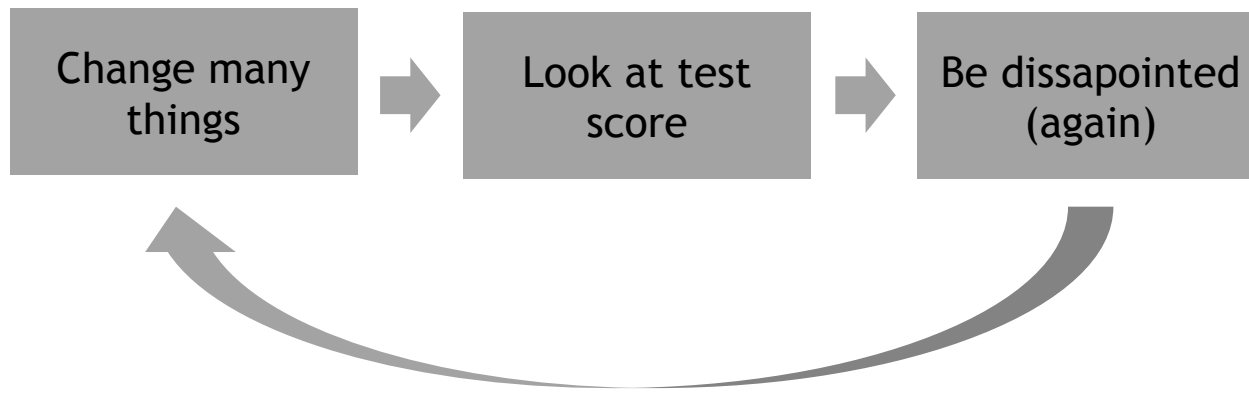
# The importance of a systematic approach

Many beginners do this:



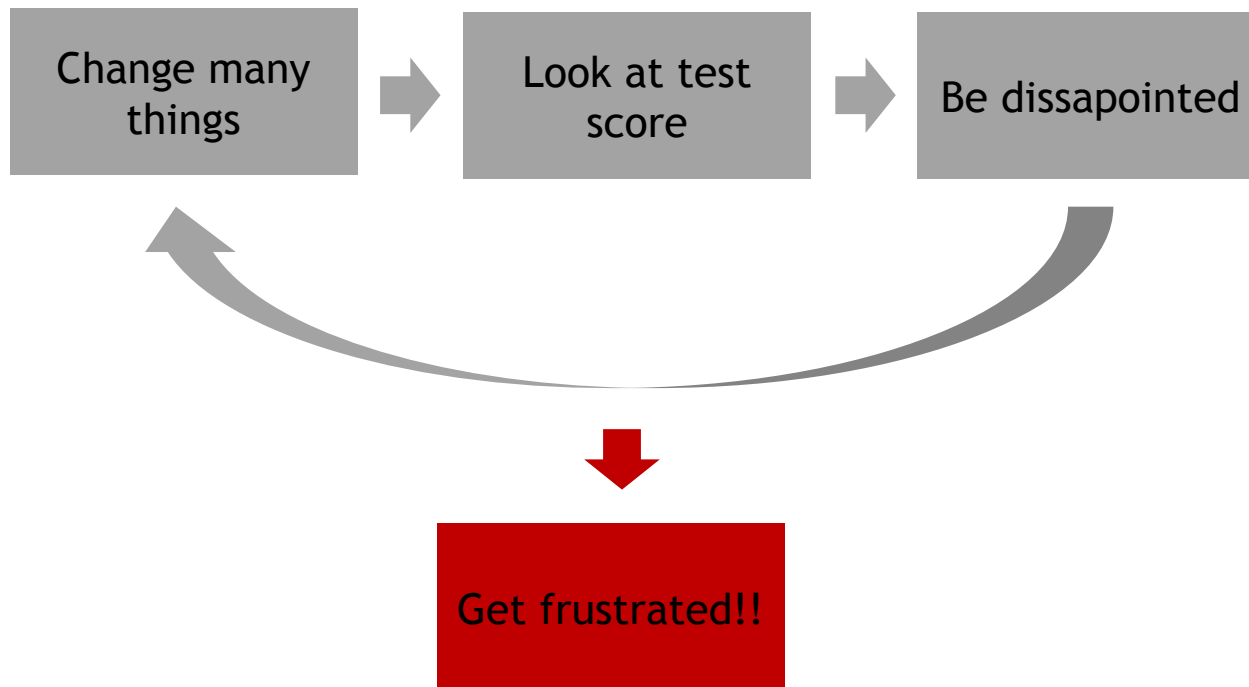
# The importance of a systematic approach

Many beginners do this:



# The importance of a systematic approach

Many beginners do this:





# Summary: data, task and model

## Data

- Must contain enough information to solve the task
- Must be representative sample from “the world”
- Enough data to cover all kinds of variability
- **Understand (difficulties in) your data!**

## Labels

- Define the task
- Must be sufficiently accurate

## Model features

- Extracted from original data:  
cleaning, transformations, compression
- Final remaining inputs to the model

## Model

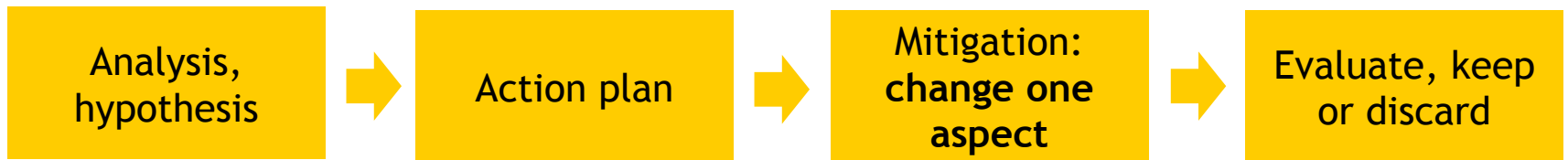
- Mathematical transformation
- Transforms features into predictions
- Must be sufficiently “powerful” to approximate labels
- **Understand the models and hyperparameters you use!**

# Scientific approach

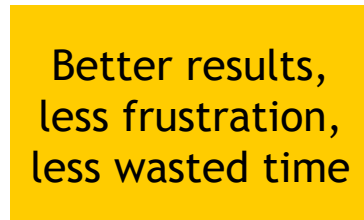
What you SHOULD do after your first dissapointment is this:

**Hypothesis:** your explanation for observations (errors, gaps, ...)

**Action plan:** best way to improve



**If your hypothesis was not correct:**  
find an explanation before moving on



Better results,  
less frustration,  
less wasted time

**Understand the models you use and their properties**

**Understand your data**

**Analyse your model errors:**  
visualise, understand what is happening!

# Where to start

1. Learn about the models
2. Learn about your data (and if necessary: first learn how to use data analysis and visualisation libraries)
3. Easily accessible ML libraries:
  - sklearn: common “traditional” ML techniques, preprocessing, data splitting, ...
  - Tensorflow: specifically for neural networks and deep learning



The screenshot shows the scikit-learn website interface. At the top left is the scikit-learn logo. To its right are navigation buttons for Home, Installation, Documentation, and Examples. Further right is a search bar with a 'Search' button and a 'Fork me on GitHub' banner. The main content area features a grid of 24 small plots showing various data distributions and model results. To the right of the grid, the text 'scikit-learn' is displayed in a large font, followed by the subtitle 'Machine Learning in Python'. Below this, a list of bullet points describes the library's features.

**scikit-learn**  
Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license