Machine Learning: Introduction

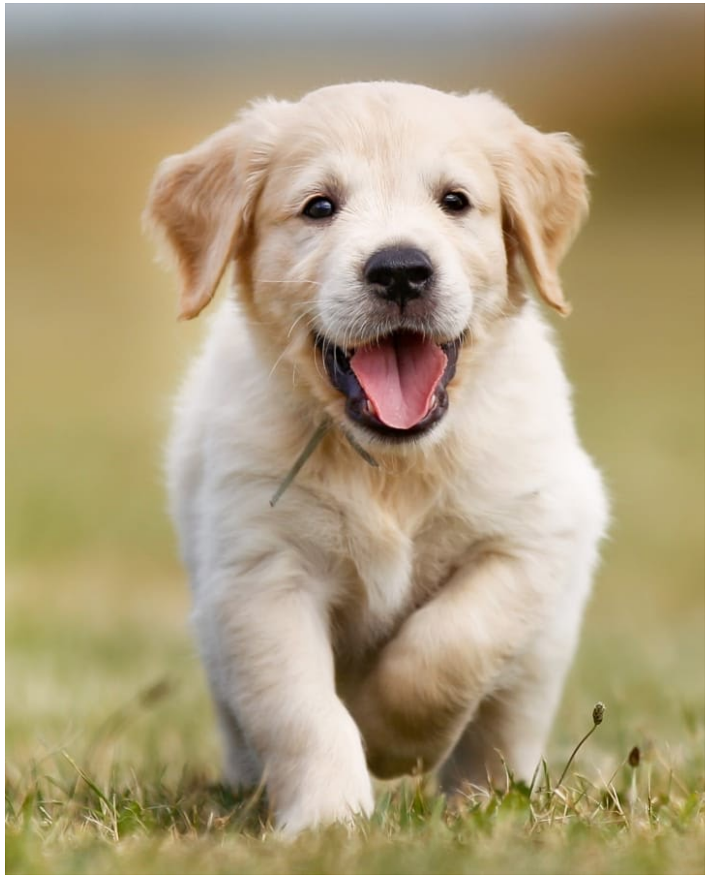
Presenter: Adi Hanuka

Day 3

Outline

- How to learn from data?
- Supervised learning (Linear regression)
- Generalization (over fitting, regularization, cross validation)
- Machine learning life cycle
- Practical concepts (data normalization, rescaling outliers, robustness)

How are you feeling this morning?



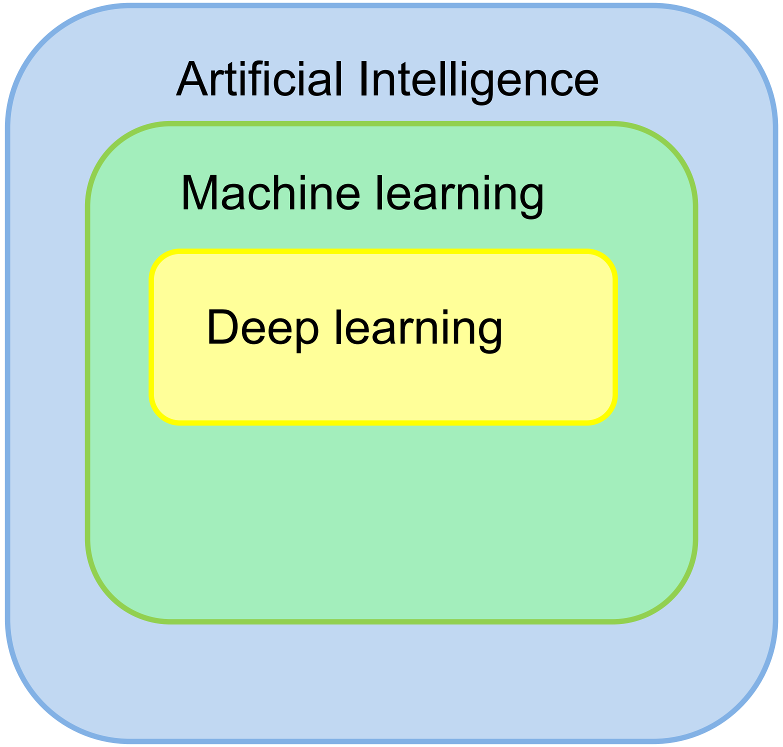
Traditional Programming vs Machine Learning

AI ⊃ ML ⊃ DL

- **Artificial Intelligence (AI)** – mimicking the intelligence or behavioral pattern of humans or living entity.

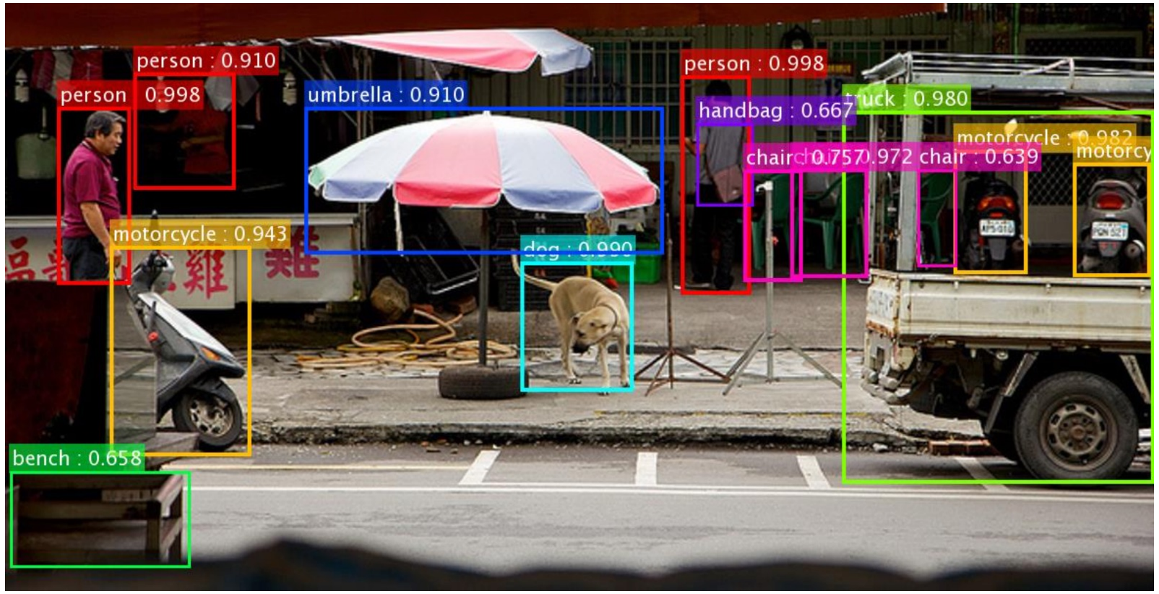
- **Machine Learning (ML)** – computers “learn” from representations to complete specific tasks without being explicitly programmed.

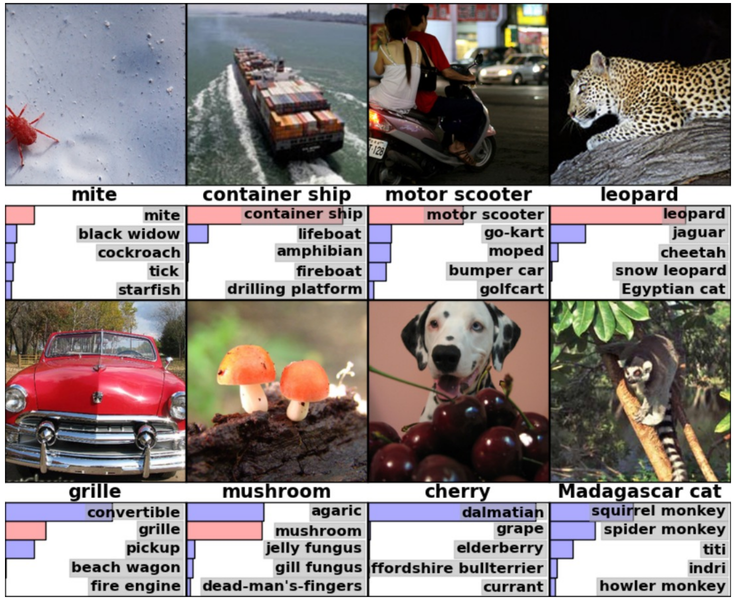
- **Deep Learning (DL)** – ML inspired by our brain’s own neural network to learn hierarchical representations.

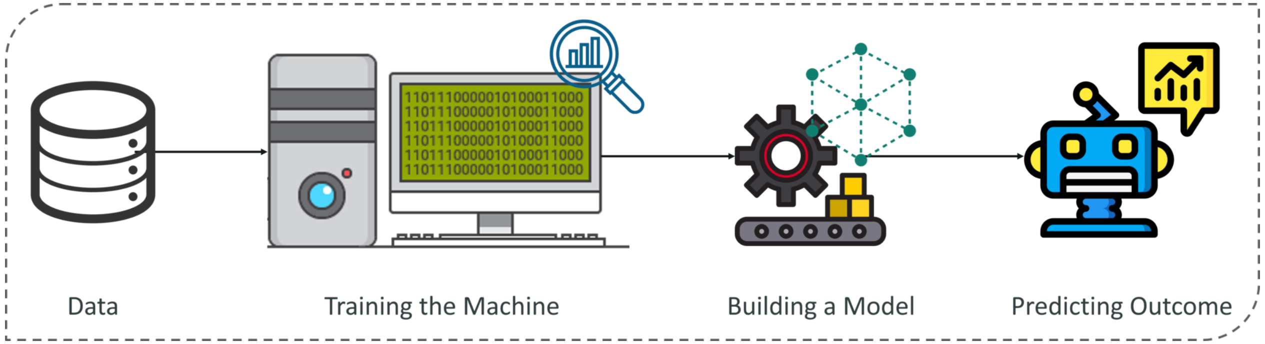


What is Machine Learning?

- Study of an algorithm that is able to *learn* from data.
- A cross-road of statistics (probability) and computer science (algorithms) where learning is casted to an optimization process.







How to learn from data?

Supervised

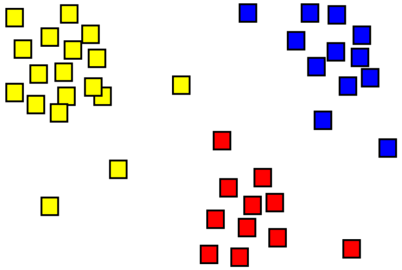
Given data X and label Y & assume an underlying function f(X)=Y, learn an approximate function that mimics f.

Classification

Unsupervised

Given data X only, learn underlying structure.

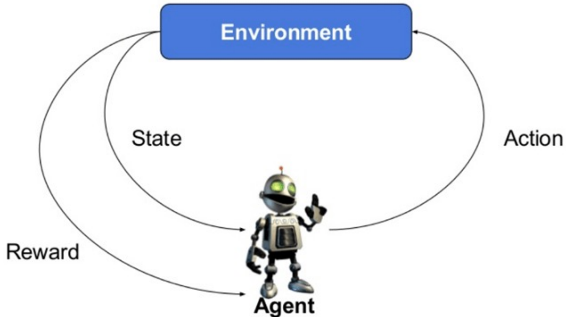
Clustering



Reinforcement

Learn to gain most cumulative reward by interacting with the environment. Data may not be static.

Facility control



How to learn from data? – Examples from Healthcare

Supervised

Task: Predict patient readmission rate.

Data: patients' treatment regime.
Labels: readmissions.

ML model: Build a model that correlates treatment regime with readmissions.

Unsupervised

Task: Categorize MRI data to normal or abnormal.

Data: MRI images.

ML model: Build a model that learns features of images to recognize different patterns (normal/abnormal).

Reinforcement

Task: Allocate scarce medical resources to handle various ER cases.

Data: treatment types, ER cases.

ML model: Build a model that learns treatment strategies for current ER cases.

Outline

- How to learn from data?
- **Supervised learning** (Linear regression)
- Generalization (Over fitting, Regularization, Cross validation)
- Machine learning life cycle
- Practical concepts (data normalization, rescaling outliers, Robustness)

Supervised Learning

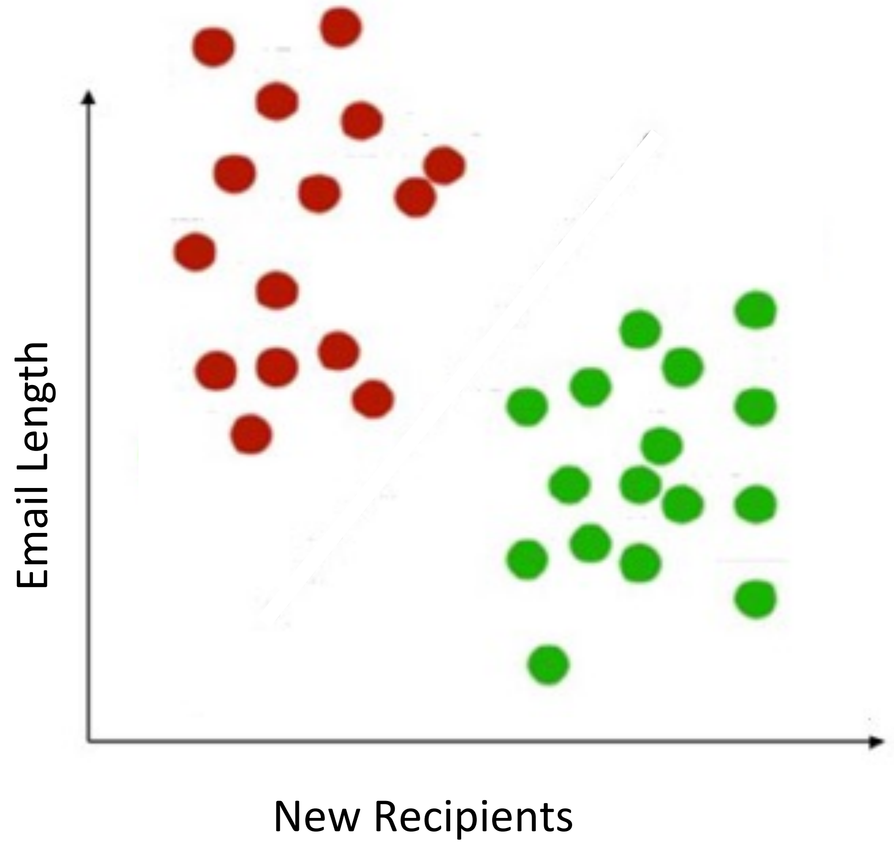
Y

Features | Labels

Instances X

Number of new Recipients	Email Length (K)	Country (IP)	Customer Type	Email Type
0	2	Germany	Gold	Ham
1	4	Germany	Silver	Ham
5	2	Nigeria	Bronze	Spam
2	4	Russia	Bronze	Spam
3	4	Germany	Bronze	Ham
0	1	USA	Silver	Ham
4	2	USA	Silver	Spam

Numeric Nominal Ordinal



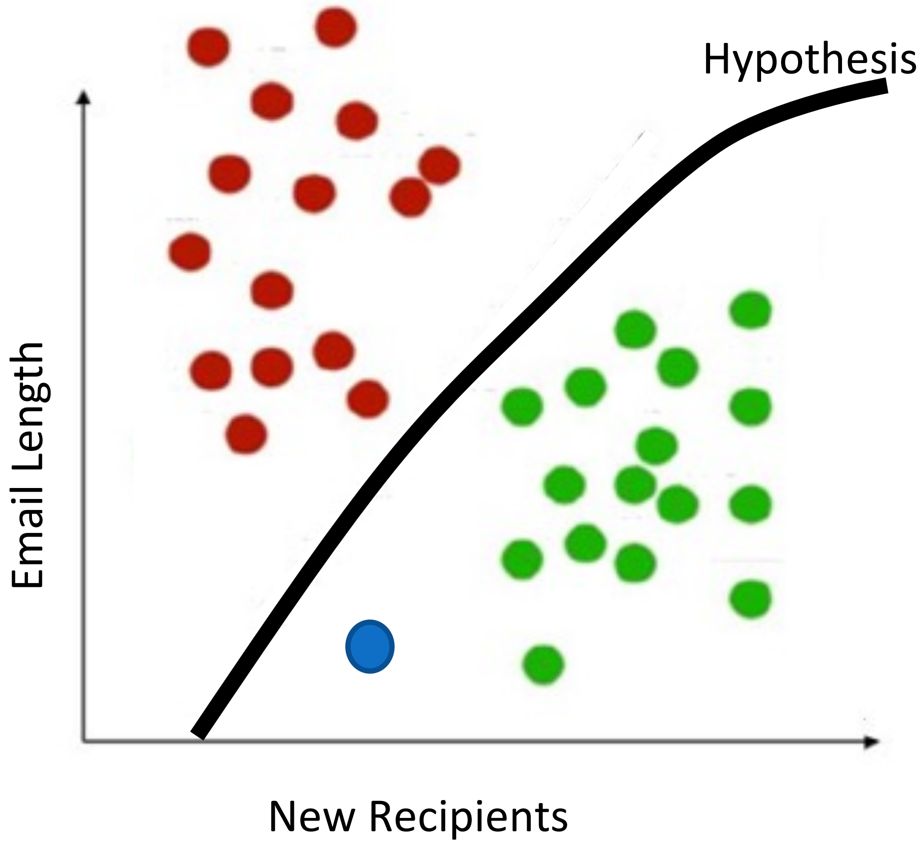
How would you classify this data?

Supervised Learning

When a new email is sent – could we predict if it is ham/spam?

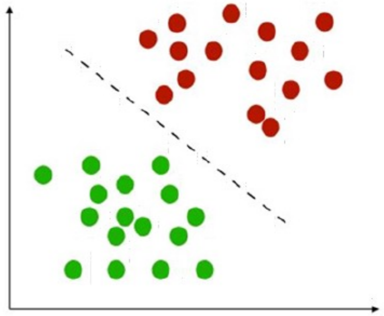
1. We place the new email in the space

2. Classify it according to the sub-space in which it resides.



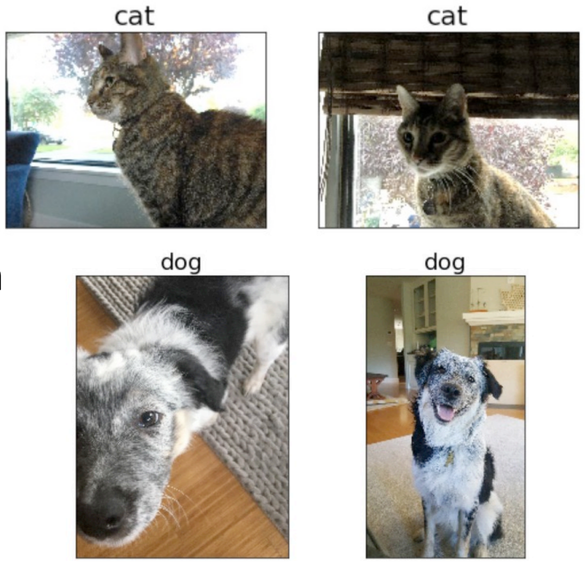
Supervised Learning - Types

Classification

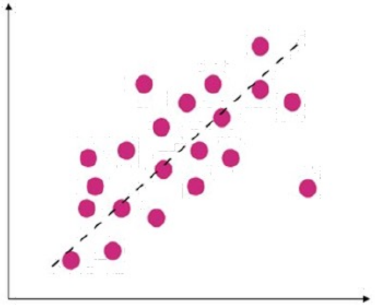


Discrete labels
– dog/cat

Image
classification

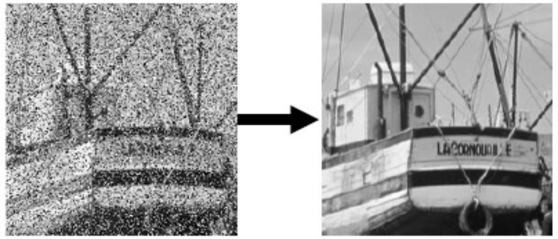


Regression



Continuous
variable – energy
of a particle

Image
denoising



Object
localization

Supervised Learning

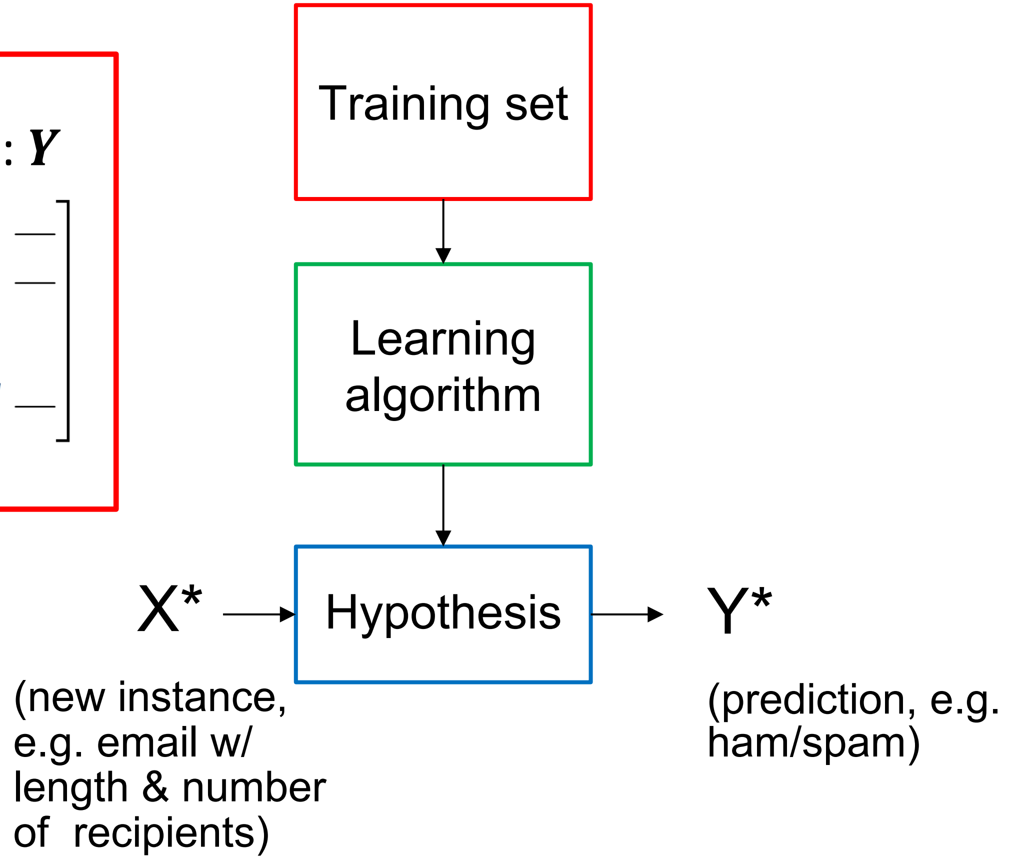
Learning Algorithm　　Training set　　Hypothesis

$$\boxed{A}(\boxed{S}) = \boxed{h}$$

- Hypothesis class $\mathcal{H}=\{h_1,h_2 \ldots\}$ wherein $h_\theta(x) \approx y$
- Loss function
- Optimization method

Inputs: $\boldsymbol{X}$　　Labels: $\boldsymbol{Y}$

$$\begin{bmatrix} \text{———} & \mathbf{x_1}^T & \text{———} \\ \text{———} & \mathbf{x_2}^T & \text{———} \\ & \vdots & \\ \text{———} & \mathbf{x_M}^T & \text{———} \end{bmatrix}, \begin{bmatrix} \text{—} & \mathbf{y_1}^T & \text{—} \\ \text{—} & \mathbf{y_2}^T & \text{—} \\ & \vdots & \\ \text{—} & \mathbf{y_M}^T & \text{—} \end{bmatrix}$$



Linear regression

Learning Algorithm · Training set · Hypothesis

$$\boxed{A}(\boxed{S}) = \boxed{h}$$

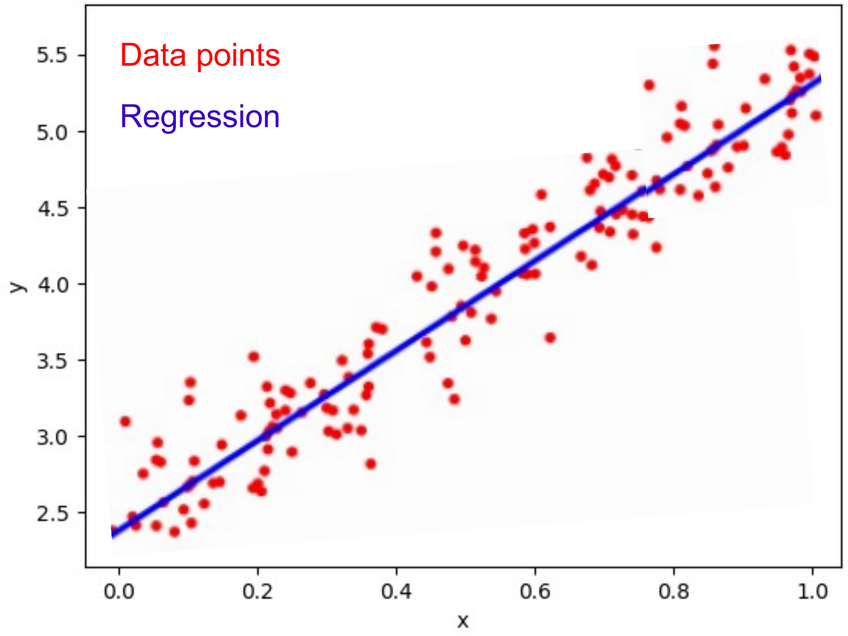
- <u>Hypothesis class:</u> Linear

$$\mathcal{H}=\{{}_{\boldsymbol{\theta}}h \,|\, \mathbb{R}\epsilon\boldsymbol{\theta}^{N+1}\},\ {}_{\boldsymbol{\theta}}h(x)=\theta_0+\tilde{\boldsymbol{\theta}}^T\boldsymbol{x} = \boldsymbol{\theta}^T\begin{pmatrix}1\\ |\\ \boldsymbol{x}\\ |\end{pmatrix}$$

- <u>Loss function:</u> Mean Squared Error

$$\mathcal{L} = \frac{1}{M}\sum_{i=1}^{M}(h_\theta(\boldsymbol{x}_i) - y_i)^2 = \frac{1}{M}\|\boldsymbol{X\theta} - \boldsymbol{y}\|^2$$

- <u>Optimization method:</u> Gradient Descent



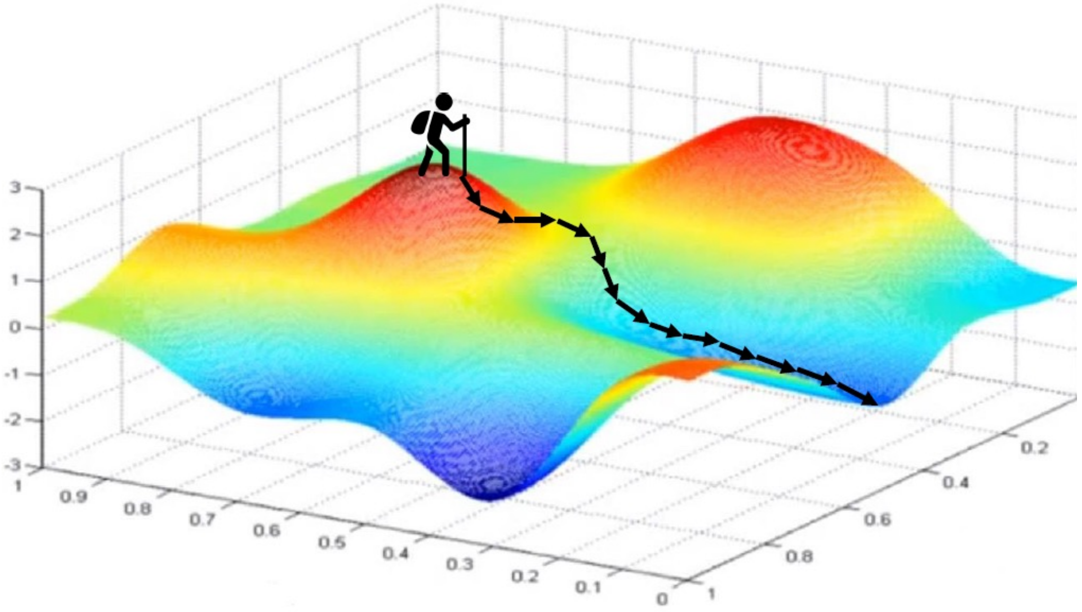
In this case, the exact solution:

$$\nabla_\theta \mathcal{L} = 0 \Longrightarrow \begin{aligned} \theta_0 &= \langle y\rangle + \theta_1\langle x\rangle \\ \theta_1 &= \frac{\sum(x_i - \langle x\rangle)(y_i - \langle y\rangle)}{\sum(x_i - \langle x\rangle)^2} \end{aligned}$$

Gradient descent - recap

Iteratively reduce loss

$$\nabla\mathcal{L}(\theta_0, \theta_1 \dots \theta_N) = \begin{pmatrix} \frac{\partial \mathcal{L}}{\partial \theta_0} \\ \frac{\partial \mathcal{L}}{\partial \theta_1} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \theta_N} \end{pmatrix}$$



1. Initialize θ randomly
2. Repeat until convergence:

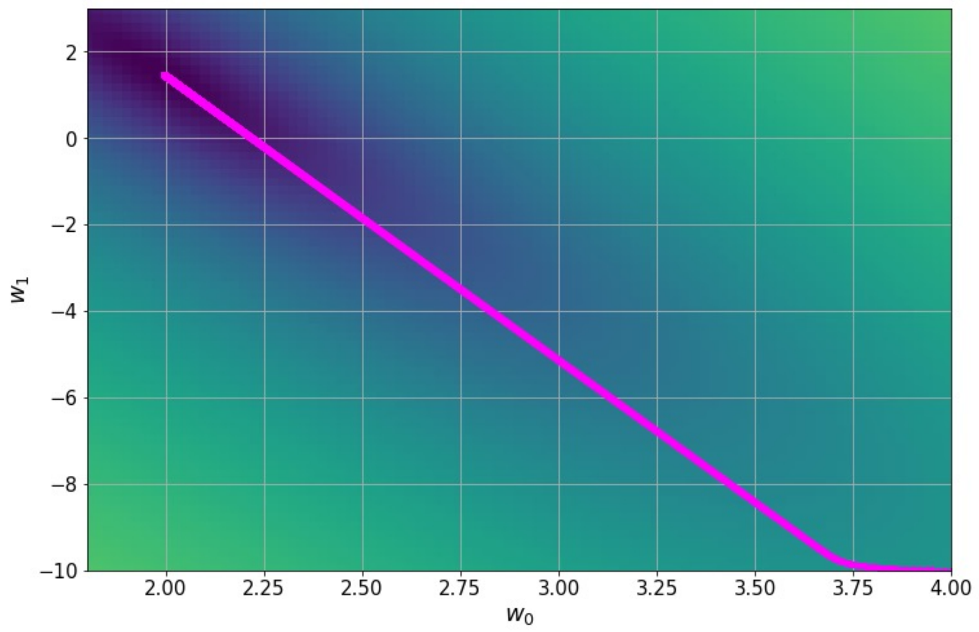
$$\boldsymbol{\theta} \coloneqq \boldsymbol{\theta} - \alpha\nabla\mathcal{L}(\boldsymbol{\theta})$$

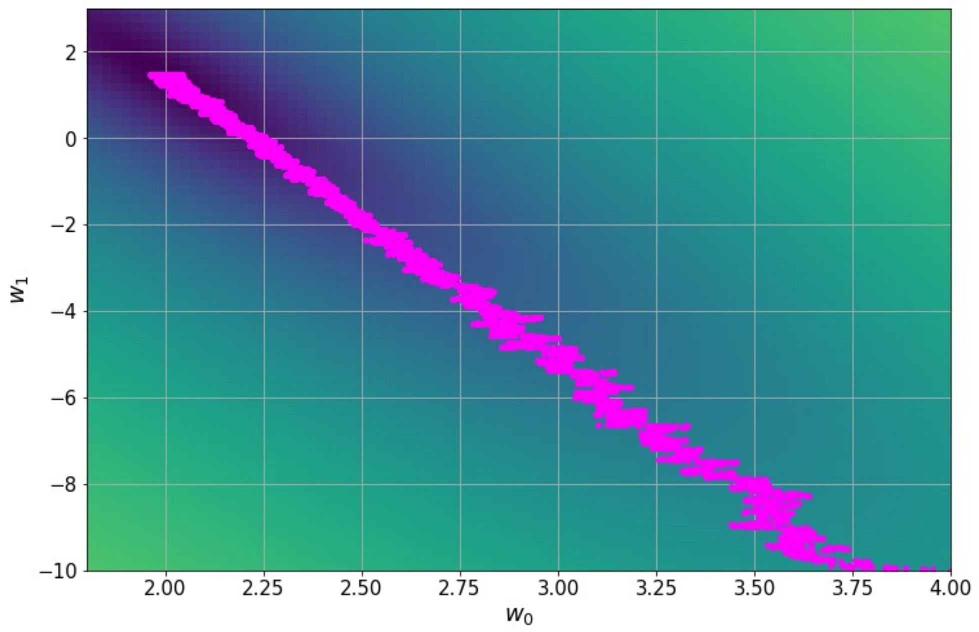
α: Learning rate

Optimize faster? Mini-batch Stochastic GD

SGD uses a subset of data for gradient calculation:

1. Create a batch = random subset of data.
2. Compute the gradient for the batch and update the parameters.

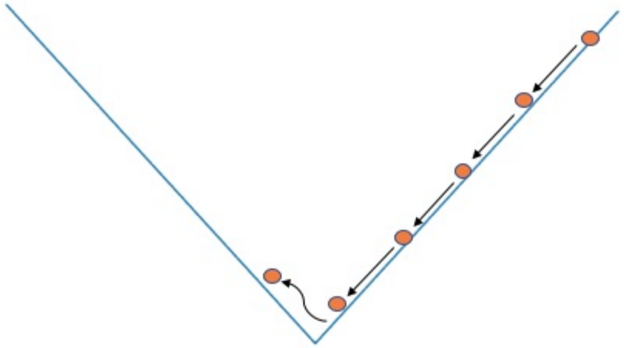




Loss functions for regression

- **Mean Absolute Error** (MAE ,L1 loss)

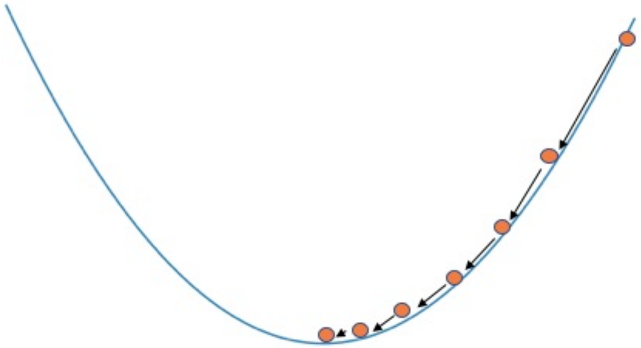
$$\mathrm{MAE} = \frac{1}{m}\sum_{i=1}^{m} \left| y_i - h_\theta(\boldsymbol{x}_i) \right|$$



- **Mean Squared Error** (MSE, L2 loss)

$$\mathrm{MSE} = \frac{1}{m}\sum_{i=1}^{m} \left(y_i - h_\theta(\boldsymbol{x}_i) \right)^2$$

Loss gets small when 1> but
may explode when1 <<



- **Huber Loss**

$$\mathrm{Huber} = \begin{cases} \frac{1}{2}a^2 \ldots \text{for } a \leq \delta \\ \delta|a| - \frac{1}{2}\delta^2 \ldots \text{otherwise} \end{cases}$$

combines them together: L1
when the loss is large,
L2 when it's small.
(hyperparameter: δ)

Outline

- How to learn from data?
- Supervised learning (Linear regression)
- **Generalization** (Over fitting, Regularization, Cross validation)
- Machine learning life cycle
- Practical concepts (data normalization, rescaling outliers, robustness)

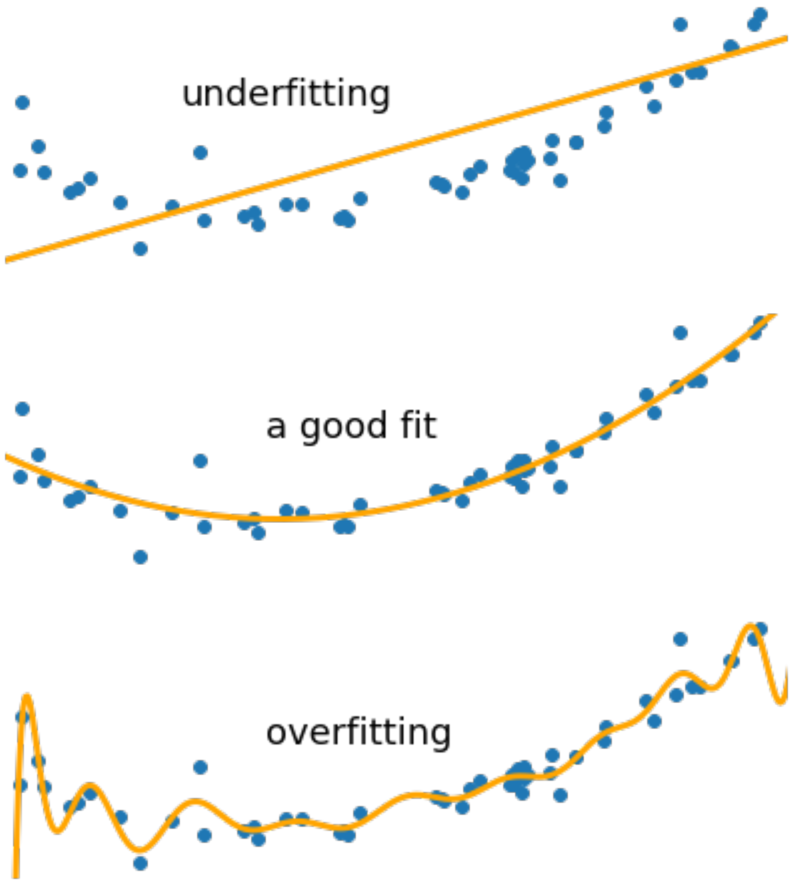
Generalization

Generalization: model works well equally on the train and unseen datasets.

Overfitting: model "memorized data".

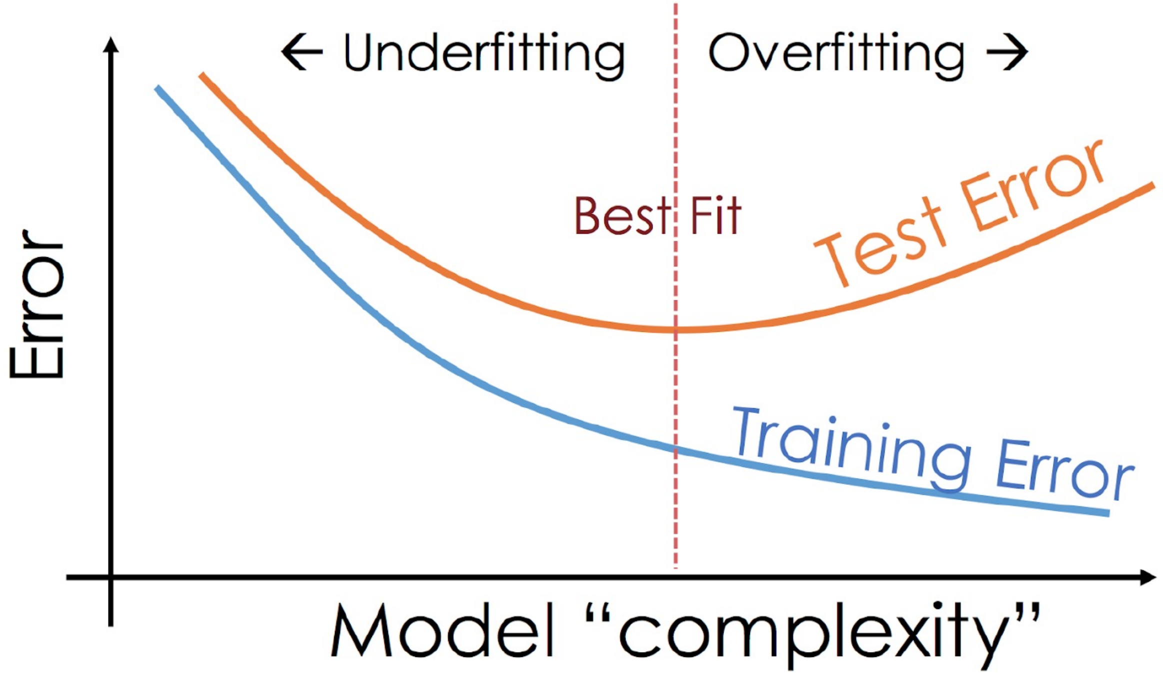
- Works well on train data but poorly on unseen data (test set).
- Typical for complex model + low data statistics.

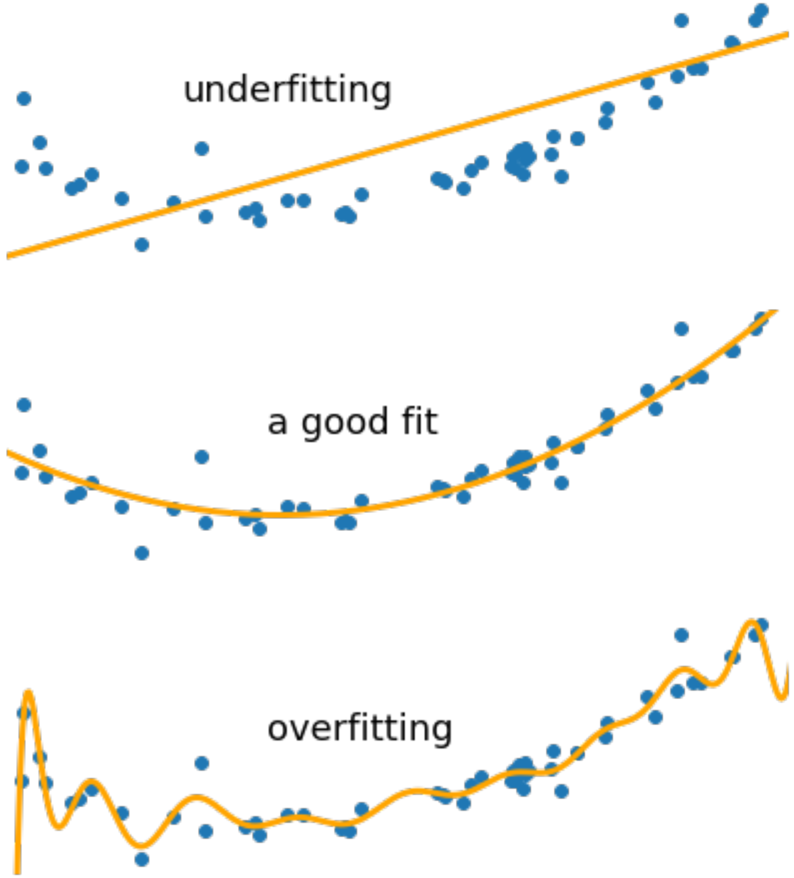
For example: A polynomial of a higher power makes the model more complex, or flexible, and as a result a model can overfit.



Generalization

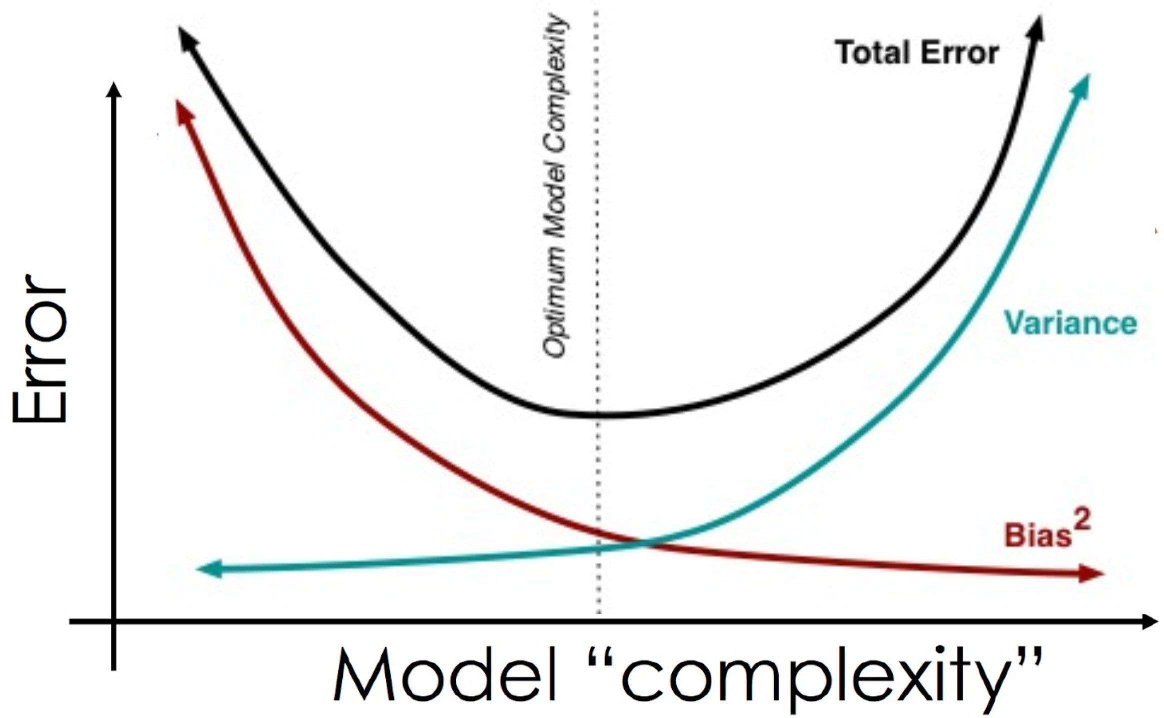
Generalization: model works well equally on the train and unseen datasets.

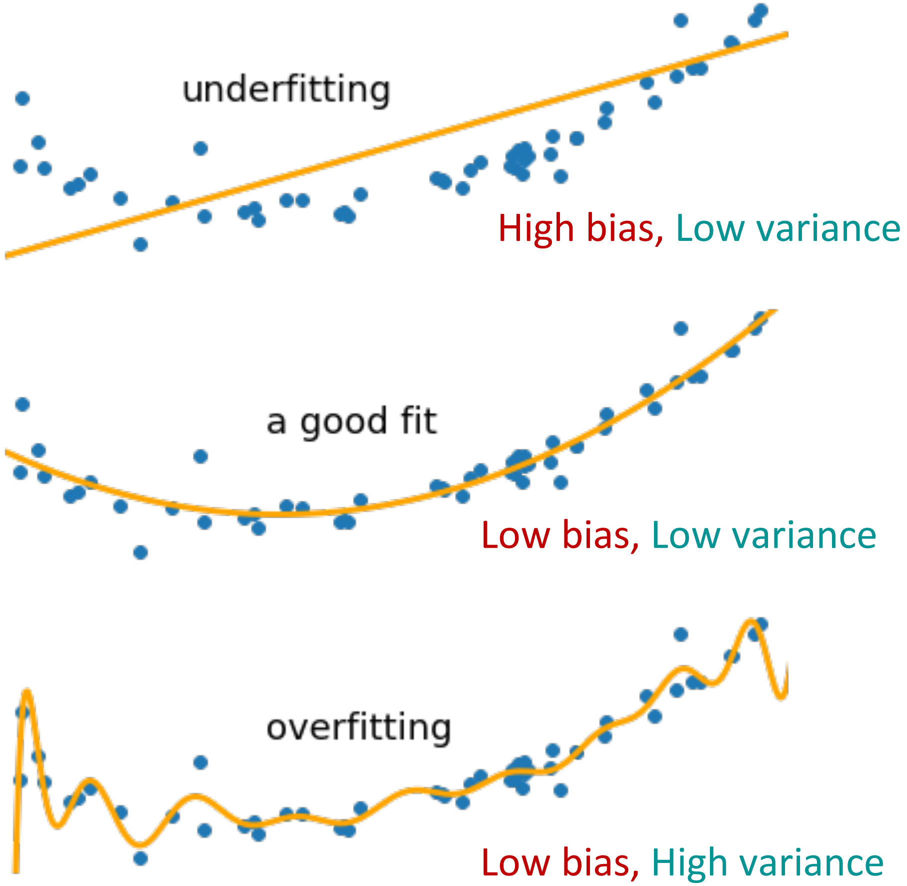




Bias-Variance tradeoff

Bias: simplifying assumptions to make the model easier to approximate.
Variance: how much the model will change given different training data.
Trade-off: tension between the error introduced by both.





Regularization

- Additional constraints on model parameters.
- Can help avoiding overfitting - prefer a simpler solution over complicated ones.

$$\mathcal{L}_{\text{total}} = \mathcal{L}\left(\mathbf{y}, h(\mathbf{x}, \theta)\right) + \lambda R(\theta)$$

model loss

regularization loss

λ: regularization parameter

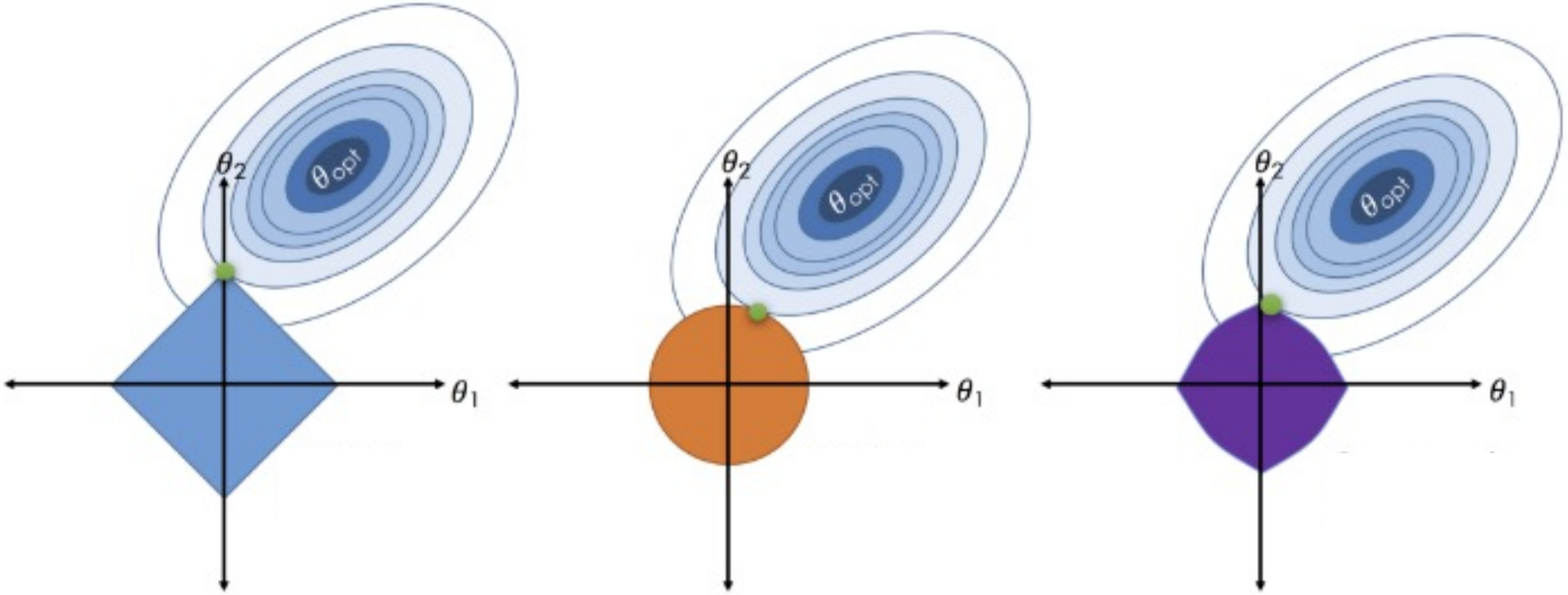
- Basic regularization terms:

L_1: $R(\theta) = \left\|\theta\right\| = \sum|\theta_i|$ Lasso - favors sparse solutions

L_2: $R(\theta) = \left\|\theta\right\|^2 = \sum{\theta_i}^2$ Ridge - favors smaller values

L_{1+2}: $R(\theta) = \sum|\theta_i| + \beta{\theta_i}^2$ Elastic net

Regularization



L_1: $R(\theta) = ||\theta|| = \sum|\theta_i|$

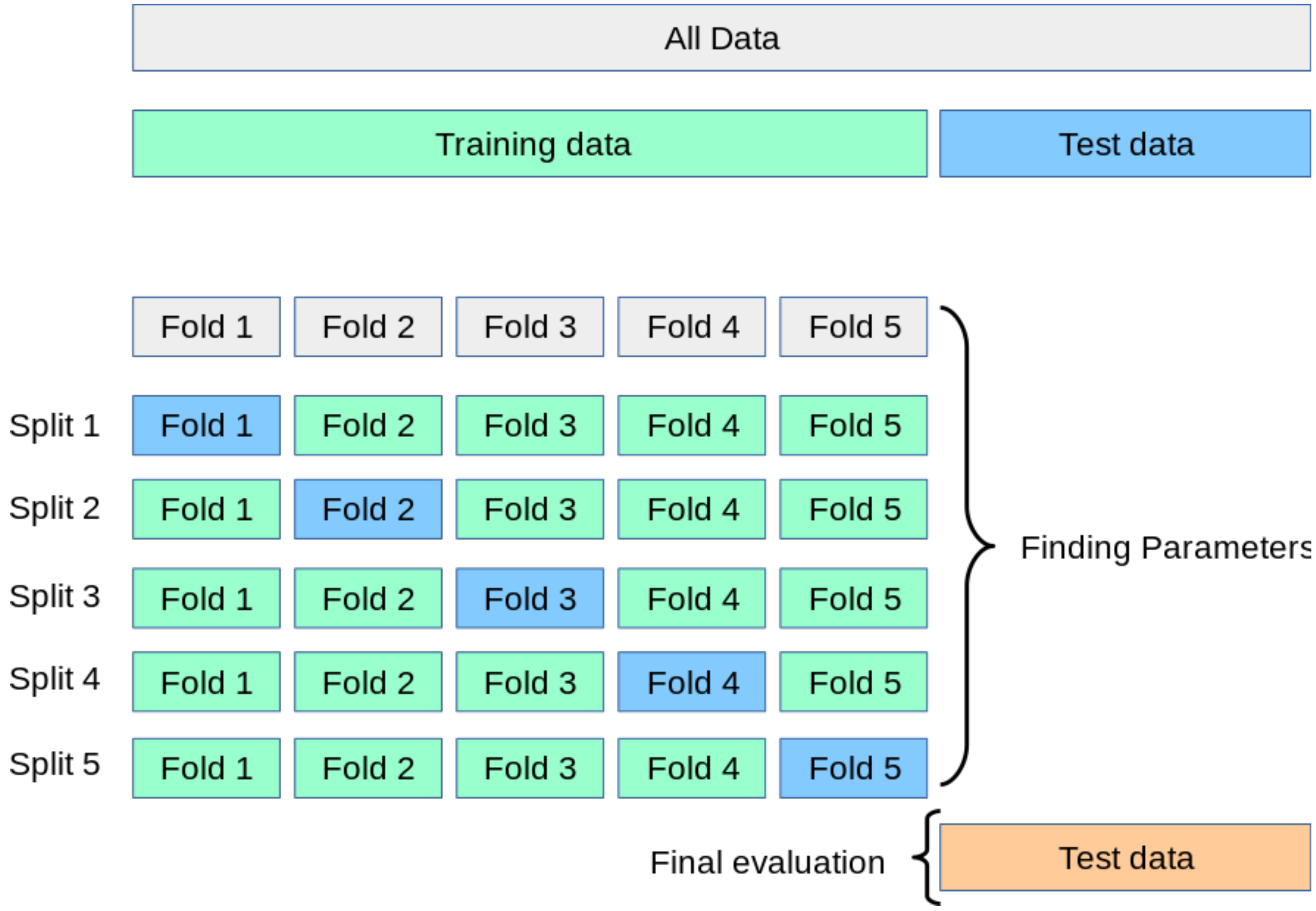
L_2: $R(\theta) = ||\theta||^2 = \sum{\theta_i}^2$

L_{1+2}: $R(\theta) = \sum|\theta_i| + \beta{\theta_i}^2$

Cross Validation

2 Datasets:
- Train set to optimize the model.
- Test set to evaluate performance after the model is tuned.

How to evaluate the model during optimization?
- Split the train set into k-folds.
- Use *i*-th set as a "validation set" to measure the performance of a model trained on the rest (k-1 combined).
- Repeat *k*-times.
- Take the mean as a performance.



From scikit webpage

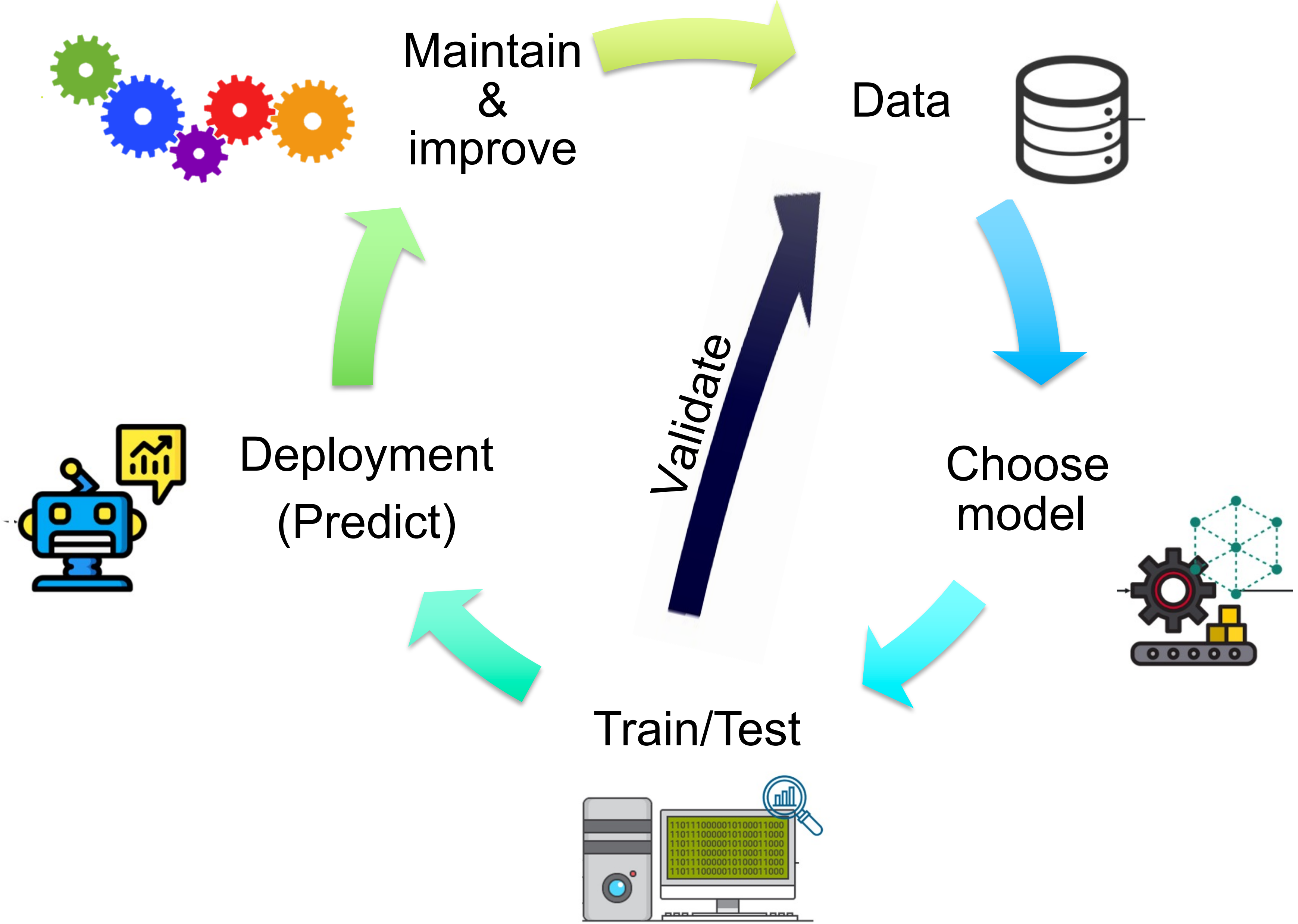
Intermediate summary

- Supervised learning – ML task of learning a function that maps an input to an output based on example input-output pairs.
  - Define a model, loss function and optimization method.
  - Popular loss functions = Mean Squared Error (MSE), Mean Absolute Error (MAE).
  - Popular optimization method = Gradient Descent (GD) or Stochastic GD (SGD) which uses a random subset of train data.
- Two datasets:
  - Train set = dataset used to optimize models parameters.
  - Test set = dataset used to benchmark the performance of the model.
  - Features: traits/attributes that can be used to describe each data sample in a quantitative manner.
- Generalization = model performs on the test dataset as well as the train dataset
  - Overfit = *memorization* of data, a model performs well on train set but poorly on test set
- Regularization = additional constraints on model parameters, can help avoiding overfitting.
  - L2 prefers smaller weight values, L1 may lead to a sparse solution
- Cross validation = splitting the train set to k-folds to create validation set(s) and measure model performance and/or tune hyperparameters.

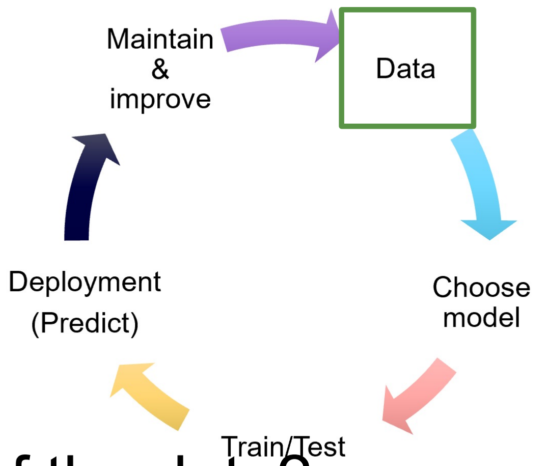
Outline

- How to learn from data?
- Supervised learning (Linear regression)
- Generalization (Over fitting, Regularization, Cross validation)
- **Machine learning life cycle**
- Practical concepts (data normalization, rescaling outliers, Robustness)

ML life cycle

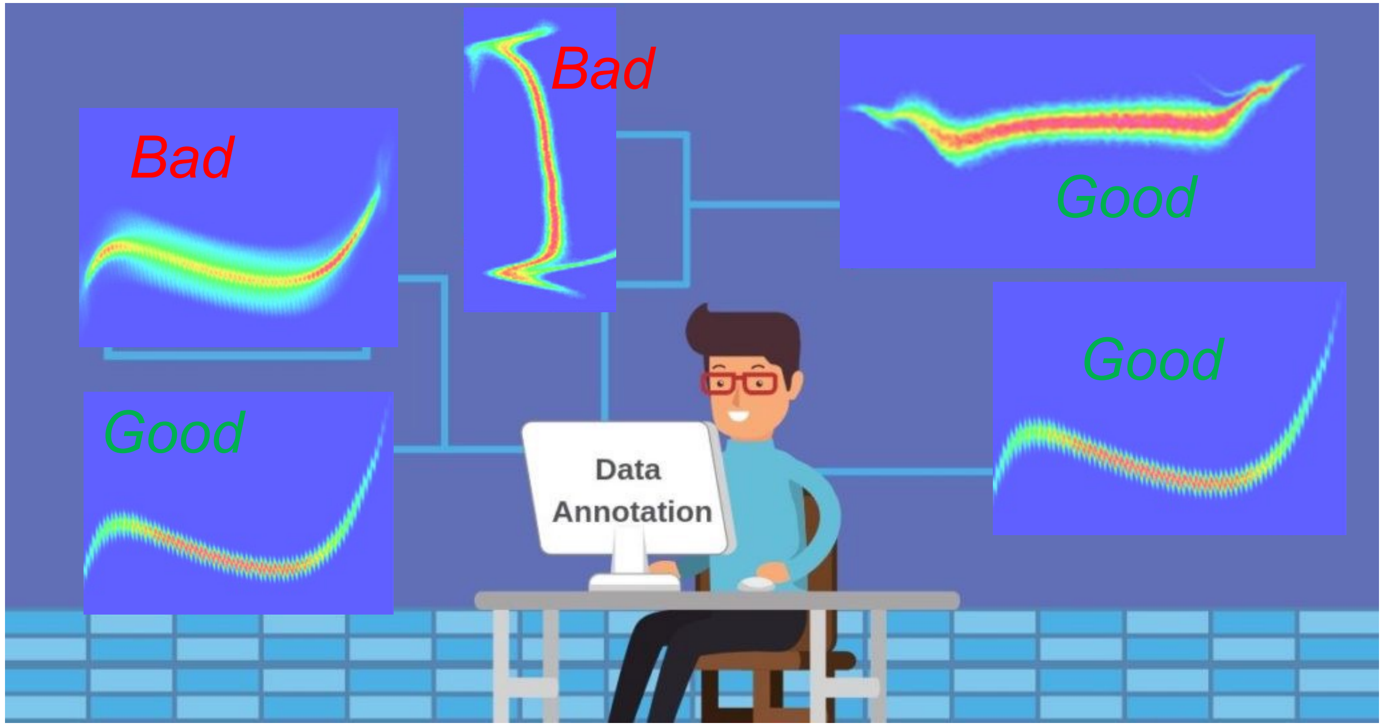


ML Life Cycle – Dataset Labeling

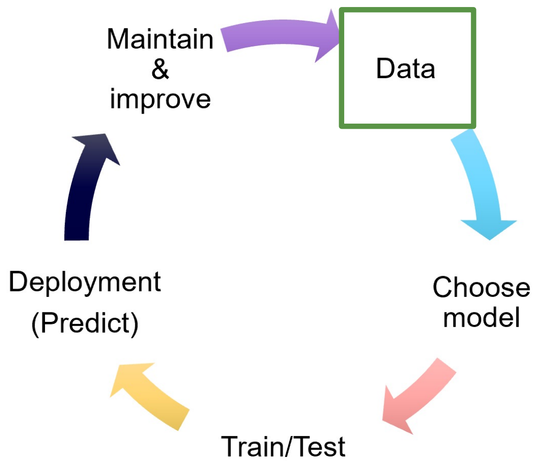


What purpose will this dataset serve?

- will the data be used for training, testing, both?
- will the data be used to evaluate an existing algorithm?
- will the labels be used to determine the underlying distribution of the data?



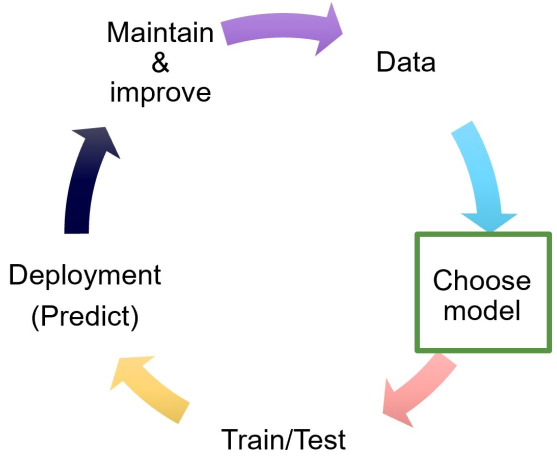
ML Life Cycle – Dataset Cleaning



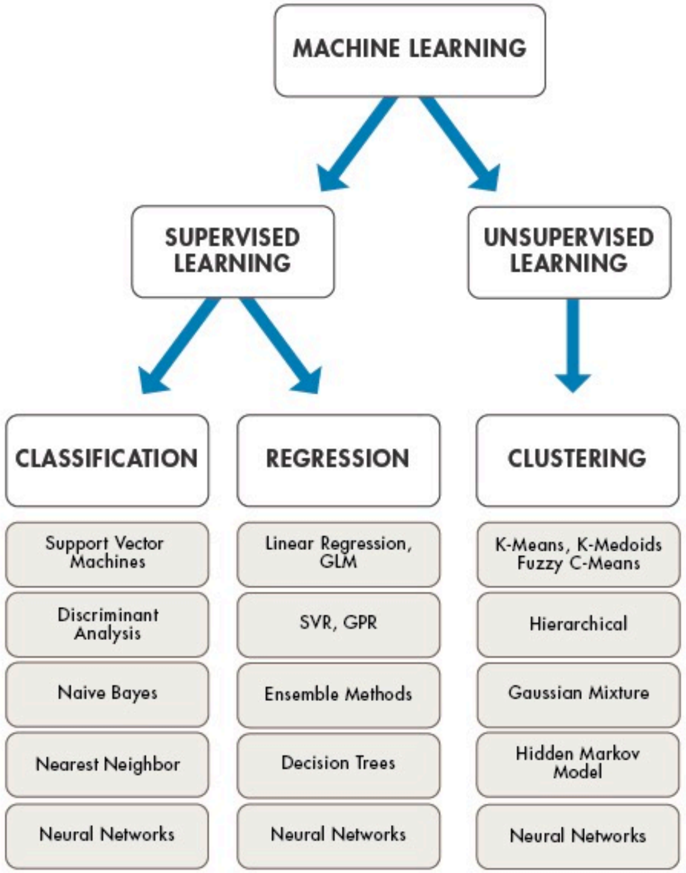
Prepare the data for training and/or testing

- Remove unwanted observations
- Combine labels from multiple people (measure label quality)
- Remove (or add) bias for training/testing purposes
- Augmentation

ML Life Cycle – Choosing a model

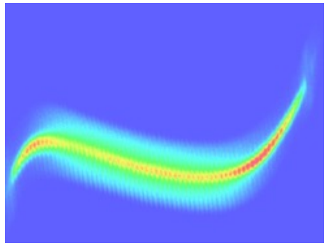


1. *What is our task?*



2. *Choose simple models first, don't overkill: Occam's razor rule*

Maybe a simple linear regression on one feature will do.

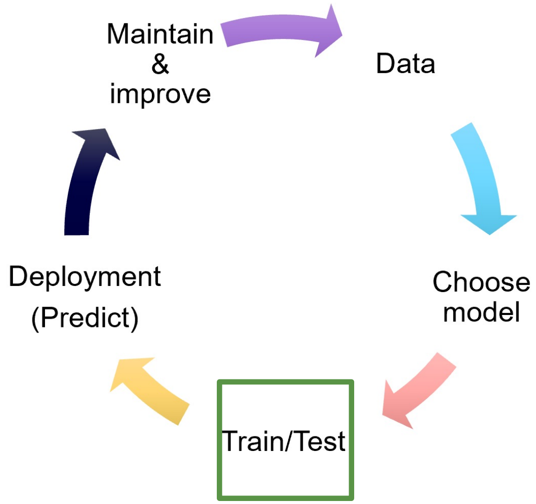


Consider the input (e.g. pixels, derived features, etc.)

$$y = a_0 + a_1 x$$

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_n x^n$$

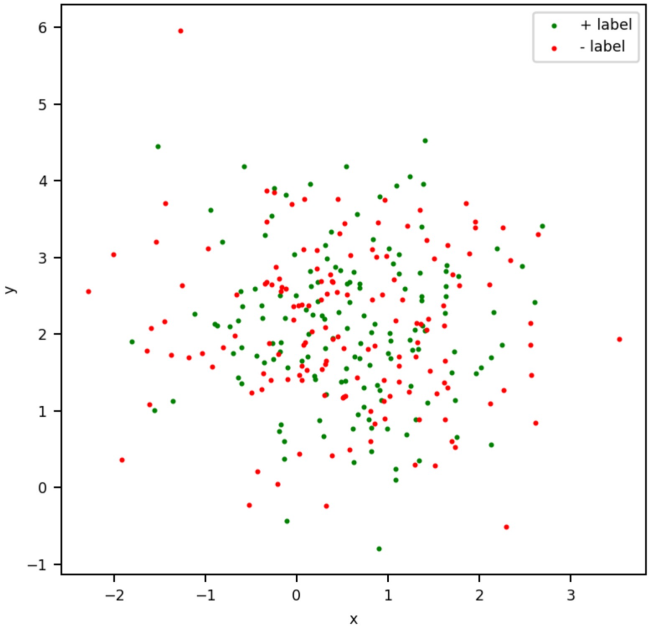
ML Life Cycle – Train/Test



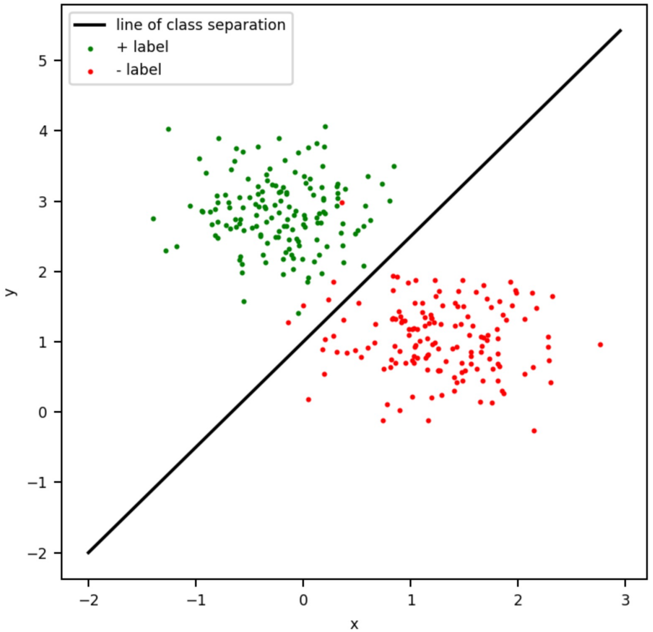
Model training: modify model parameters so that they describe the data.

Model testing: How general is the model when evaluated on new data?

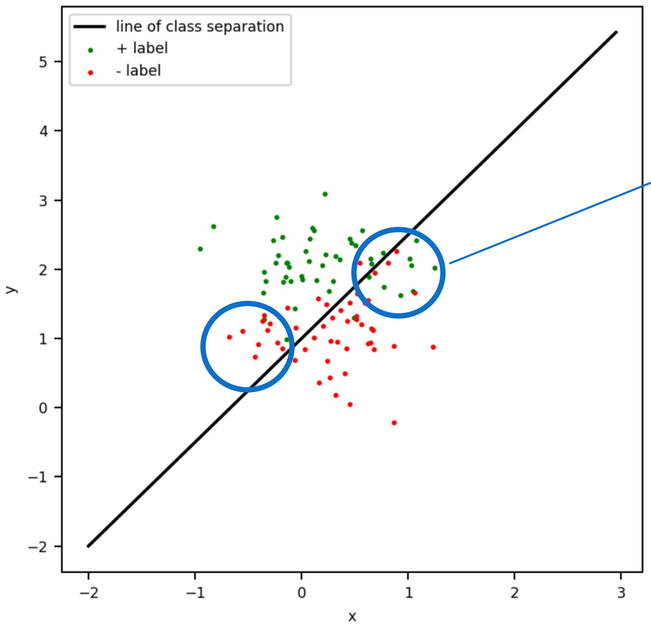
Before training



After training



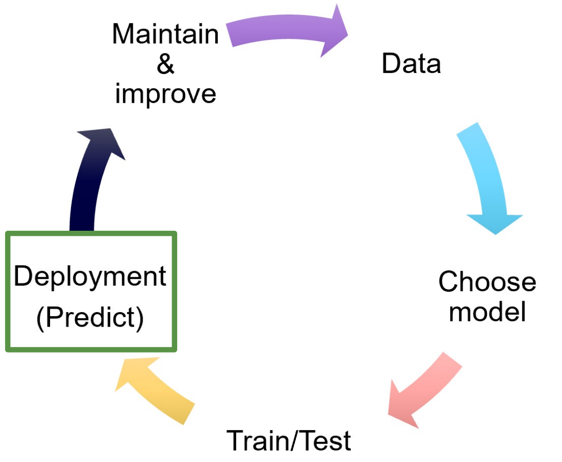
Testing



Performs poorly on the test data

Make sure to choose a test set that is representative of the real population!

ML Life Cycle – Deployment (MLOps)



The model is trained and performs well on unseen data
→ it's time to go on production (deployment)

1. New data in

01100
10110
11110

01100
10110
11110

01100
10110
11110

2. Data cleansing

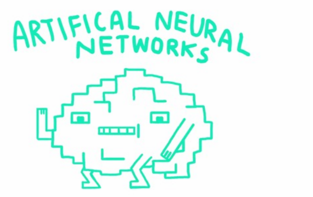
- Remove outliers
- Build features needed
- Standarise (scale) data

3. Data storage (databases)

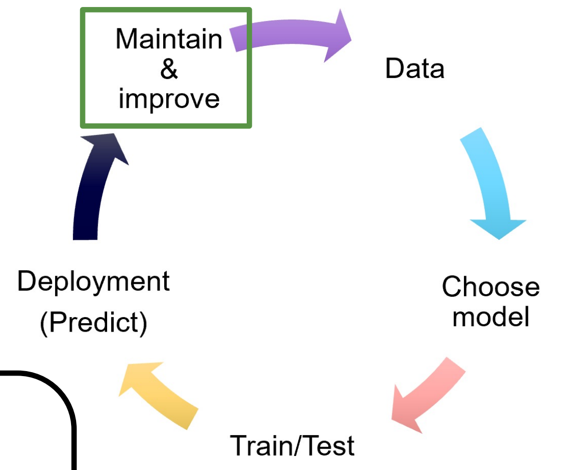
- Availiable: Fast read/write of data
- Consistent: data won't change

4. Prediction: ML inference

Data is ready to be consumed by the trained model:



ML Life Cycle – Maintain & Improve (MLOps)



But what happens if data changes? *MLOps: make sure the deployed model is correct and predicts well*

5. Test again

Is your performance on new test similar to the original test?

Old test set

01100
10110
11110

Old performance

New test set

01100
10110
11110

New performance

Similar performance

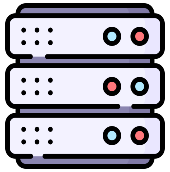
5. Retrain with the new data:

Retrain and substitute old model by the newly trained one

Different performance

Report to human...

You better have a look at this...



Write a story on Jira, will do tomorrow

Outline

- How to learn from data?
- Supervised learning (Linear regression)
- Generalization (over fitting, regularization, cross validation)
- Machine learning life cycle
- **Practical concepts** (data normalization, rescaling outliers, robustness)

Practical concepts – data scaling

Features have different ranges

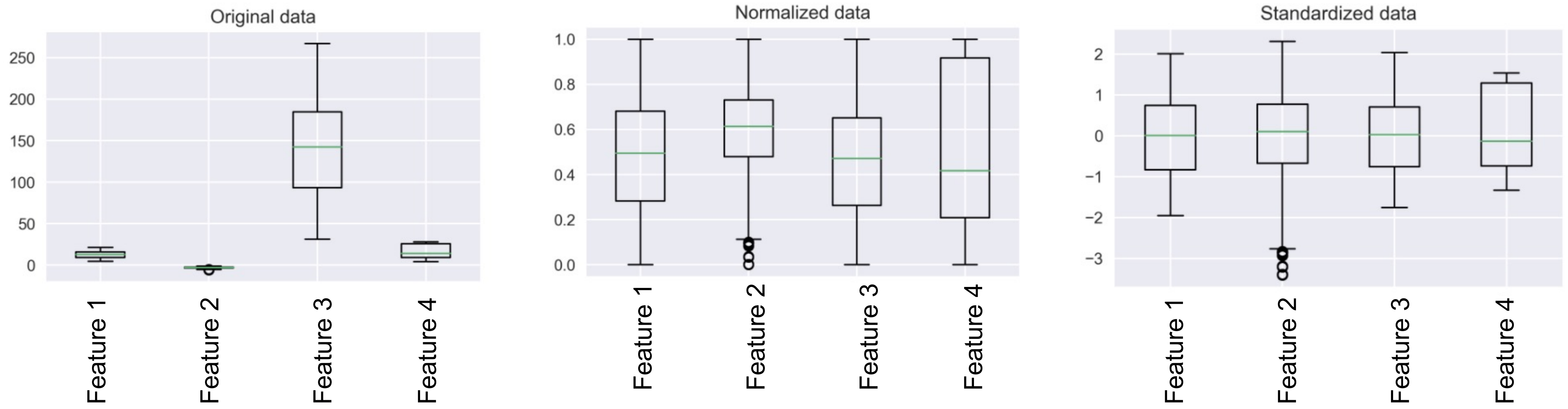
Name	Weight	Price
Orange	15	1
Apple	18	3
Banana	12	2
Grape	10	5

“Weight” > “Price”

The algorithm assumes that “Weight,” is more important than “Price.”

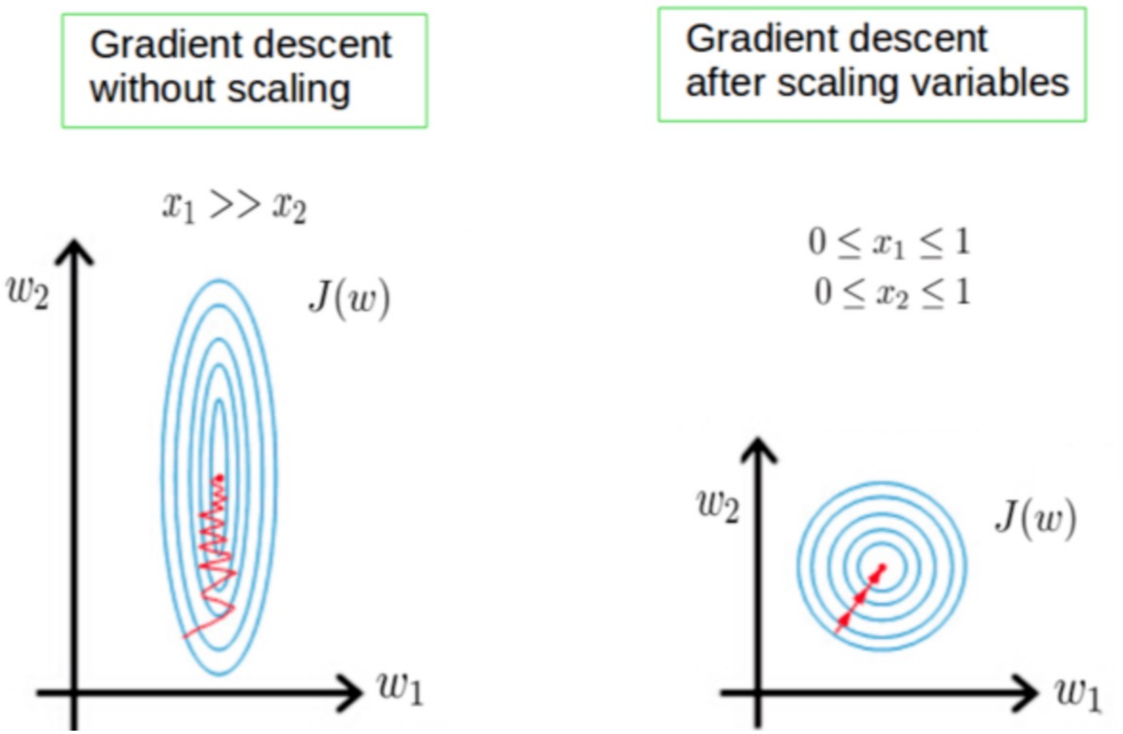
Practical concepts – data scaling

Features have different ranges → Scaling the data so that all the features will be comparable and have a similar effect on the learning models.



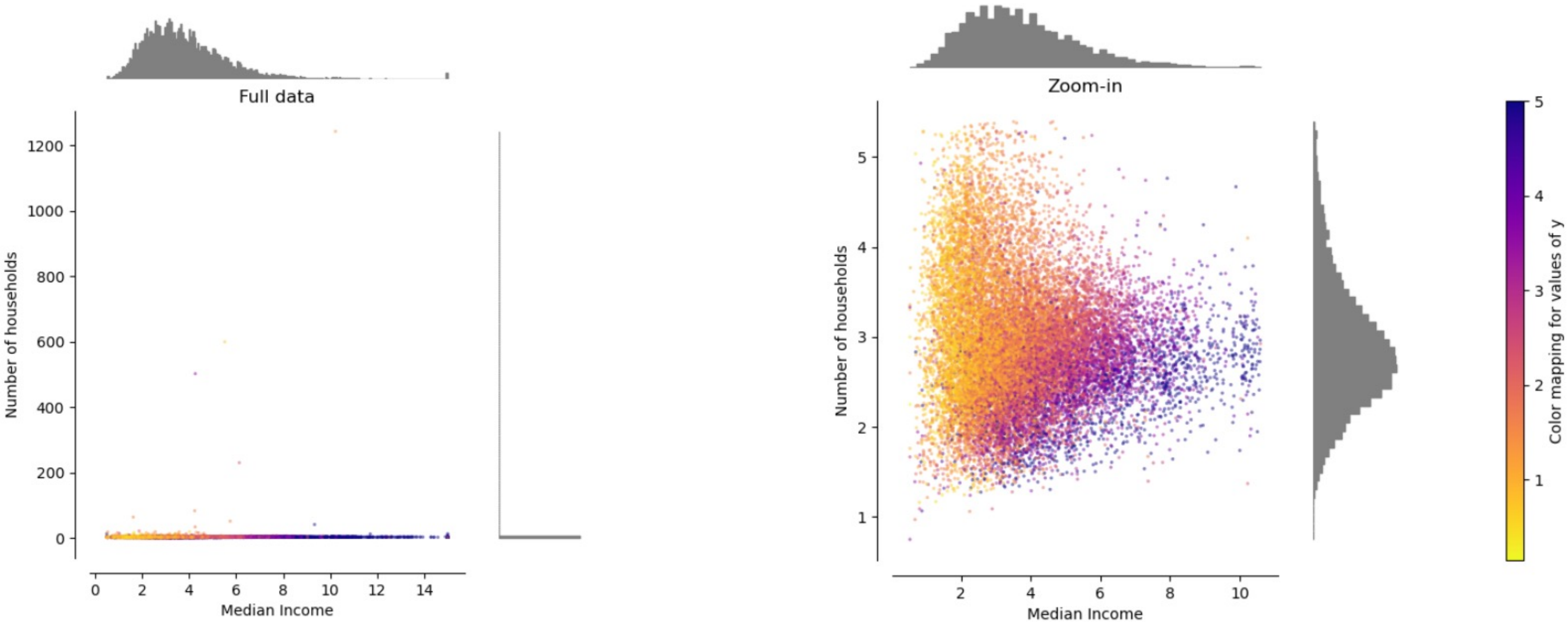
Practical concepts – data scaling

Another reason for feature scaling is that some algorithms converge much faster with feature scaling than without it.



Practical concepts – scaling data with outliers

- Data has marginal outliers → pre-processing can be very beneficial.

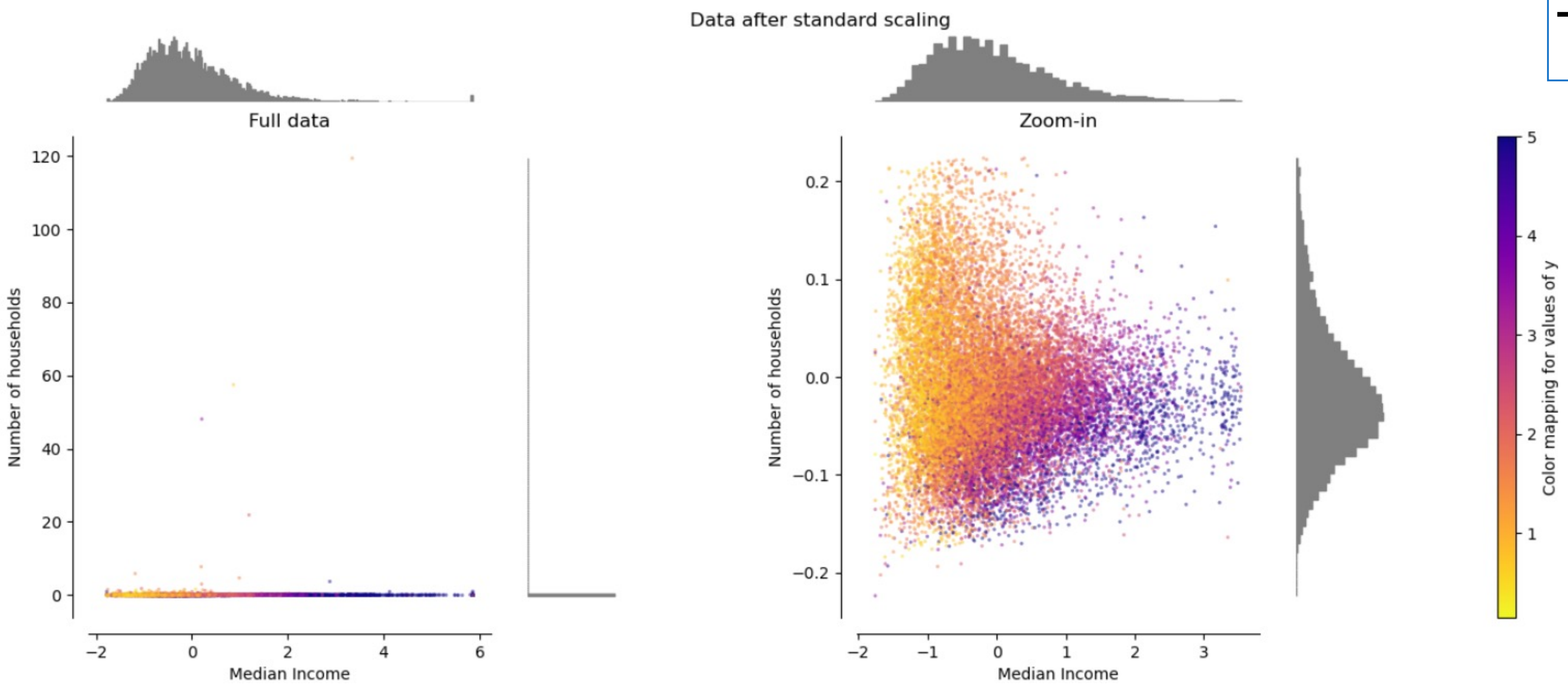


From scikit webpage

Practical concepts – scaling data with outliers

- Standard Scaler removes the mean and scales the data to unit variance.
  - outliers have an influence when computing the mean & std.
  - → cannot guarantee balanced feature scales in the presence of outliers.

$$\boxed{\frac{x - \mathrm{mean}(x)}{\mathrm{std}(x)}}$$

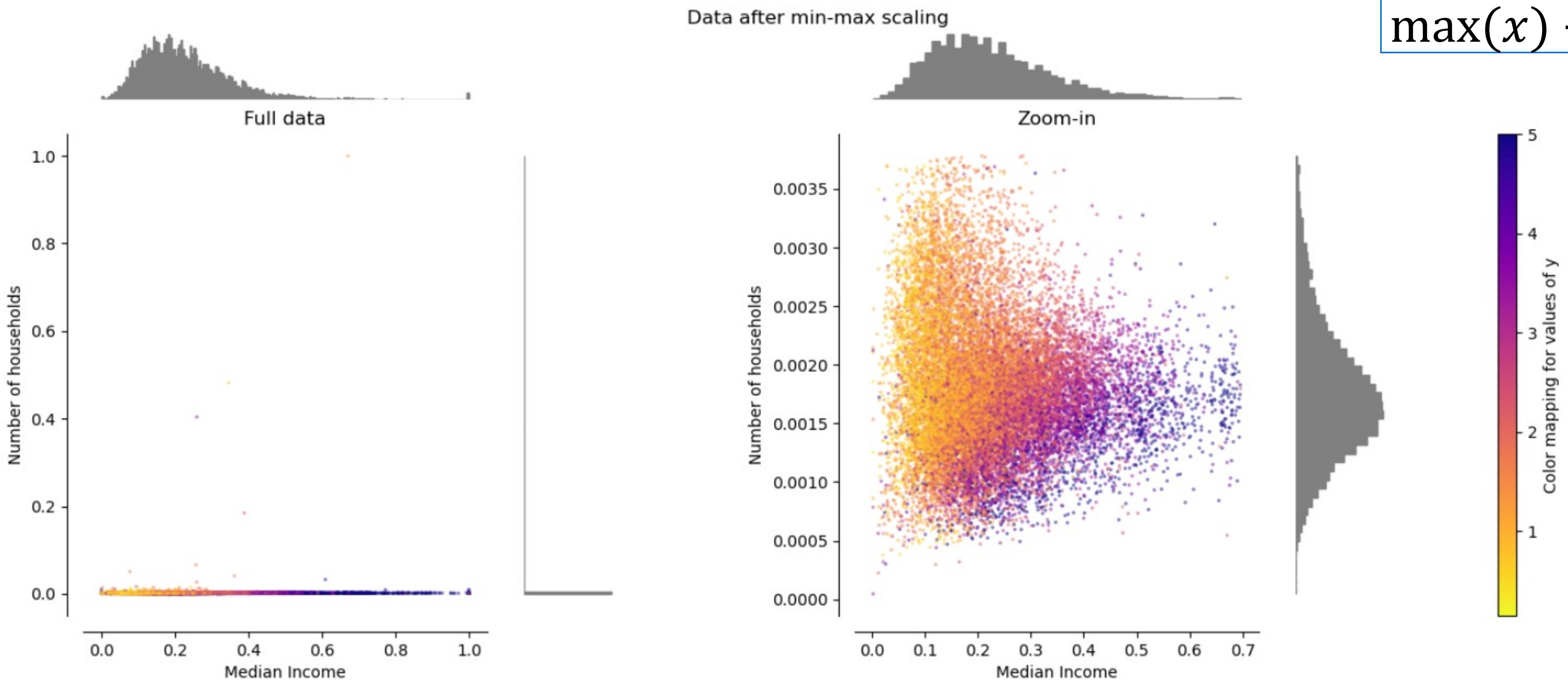


From scikit webpage

Practical concepts – scaling data with outliers

- MinMax Scaler rescales the data set such that all feature values are in the range [0, 1] → very sensitive to the presence of outliers.

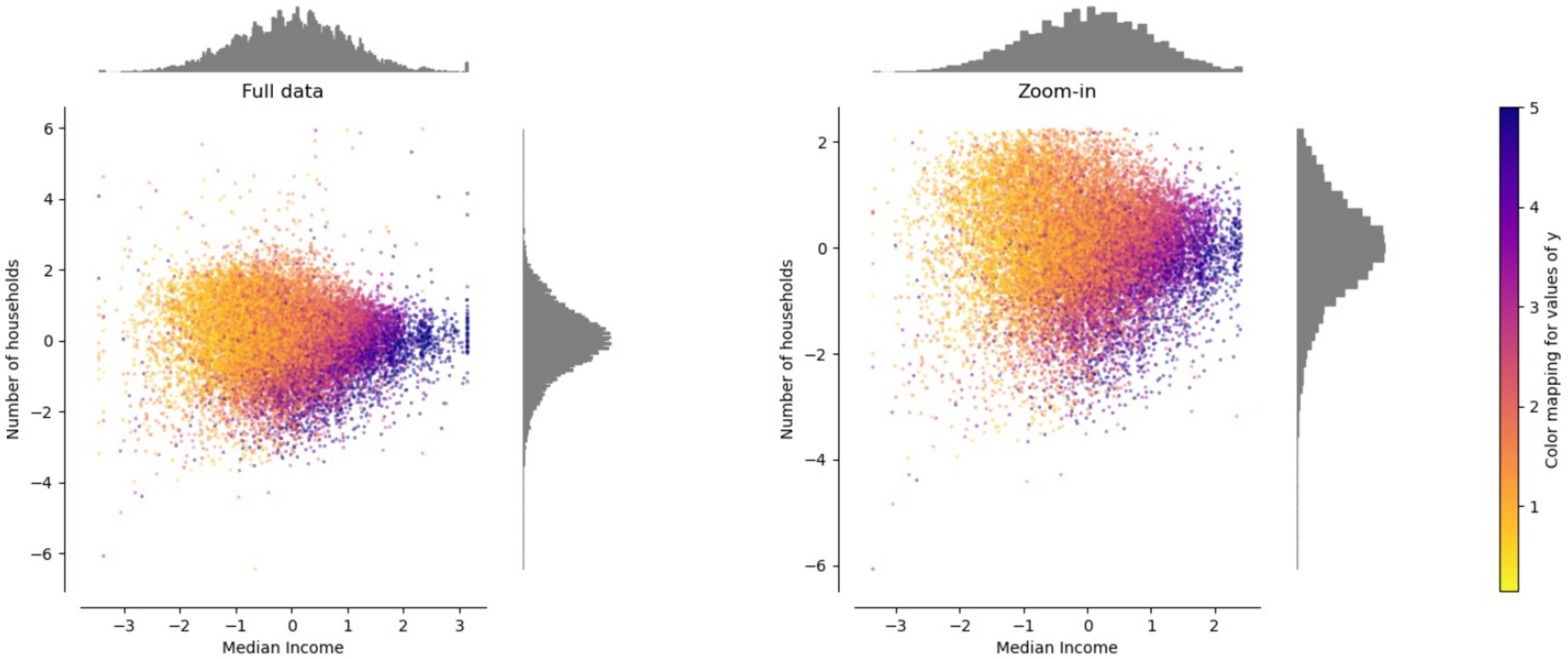
$$\frac{x - \min(x)}{\max(x) - \min(x)}$$



From scikit webpage

Practical concepts – scaling data with outliers

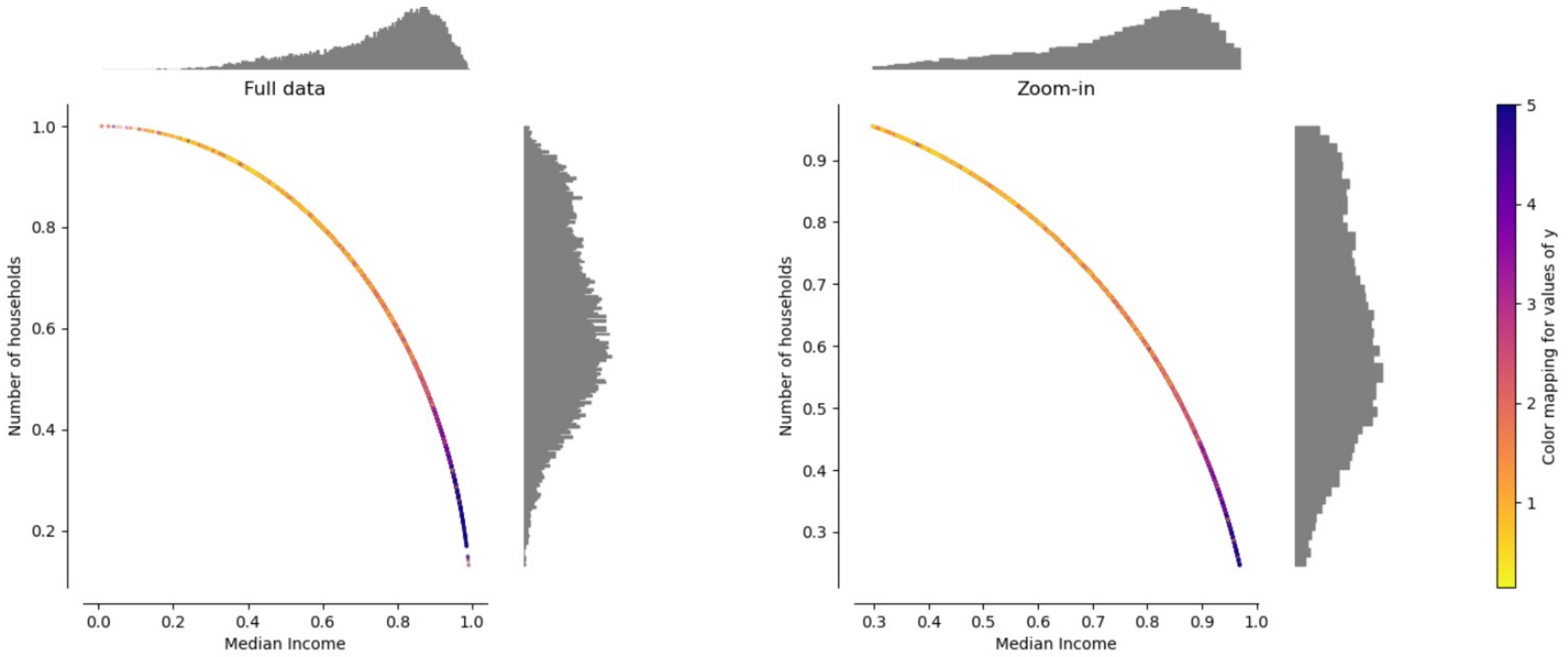
- Power transformer applies a power transformation to each feature to make the data more Gaussian-like in order to stabilize variance and minimize skewness.



From scikit webpage

Practical concepts – scaling data with outliers

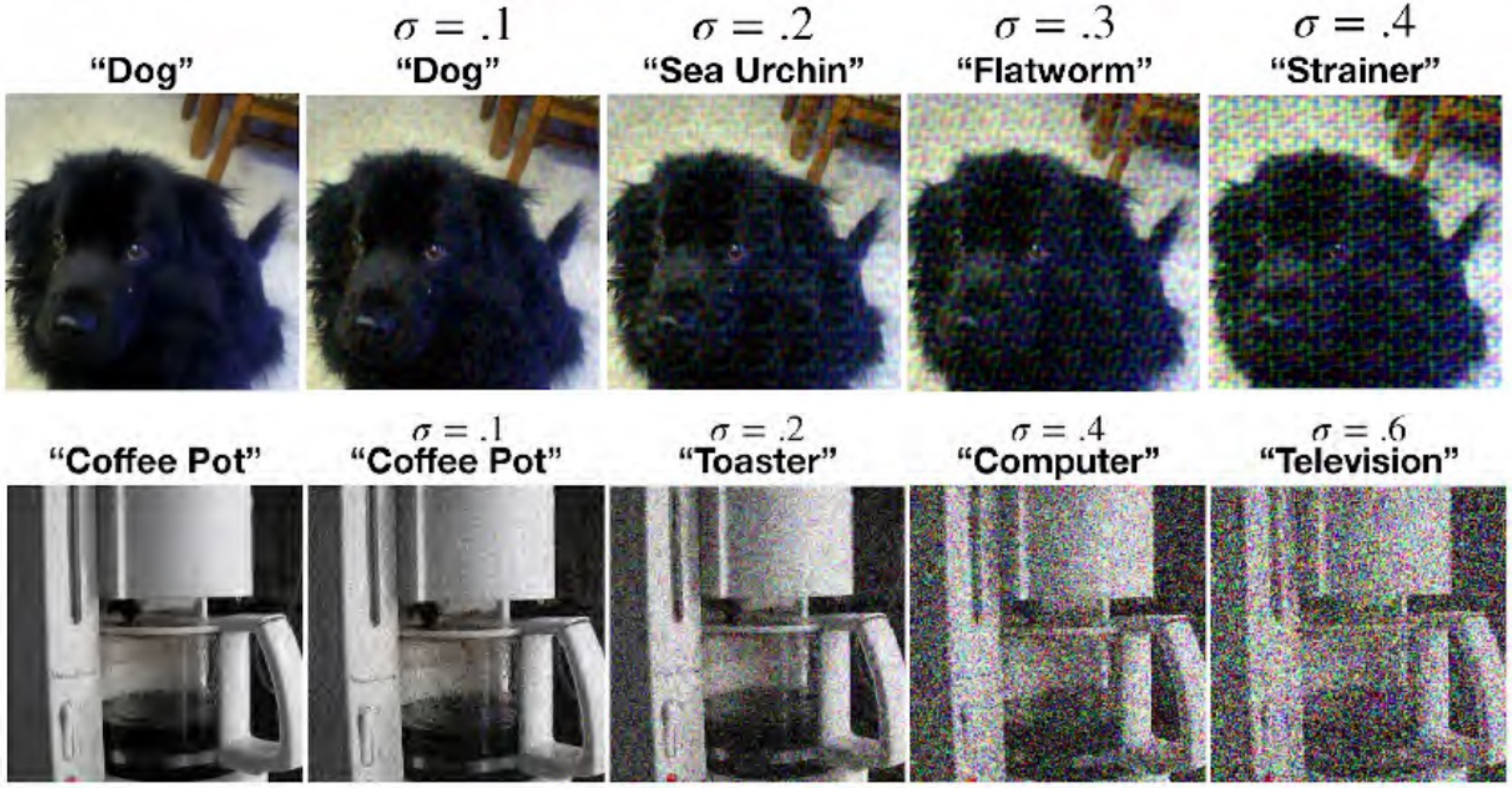
- Normalizer rescales the vector for each sample to have unit norm, independently of the distribution of the samples → all samples are mapped onto the unit circle.



From scikit webpage

Robustness

- Models are often not robust to small shifts in the distribution, especially for high-dimensional data.



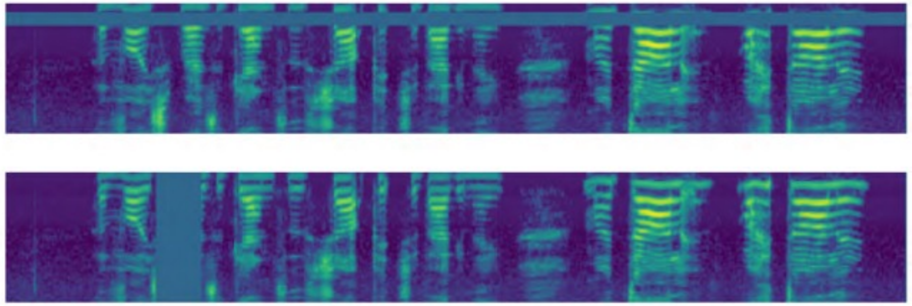
Yin et al. ; arXiv:1906.08988
Lopez et al. ; arXiv:1906.02611

Data augmentation can help

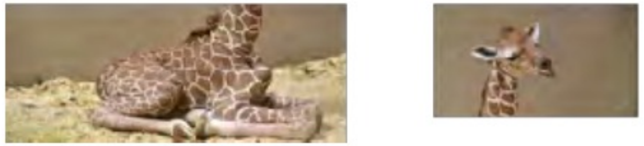
- Augmentation strategies don't need to be “physical"

Random flip left-right:

Cutout / Random erasing:



Random shifts/ crops/ color operations:

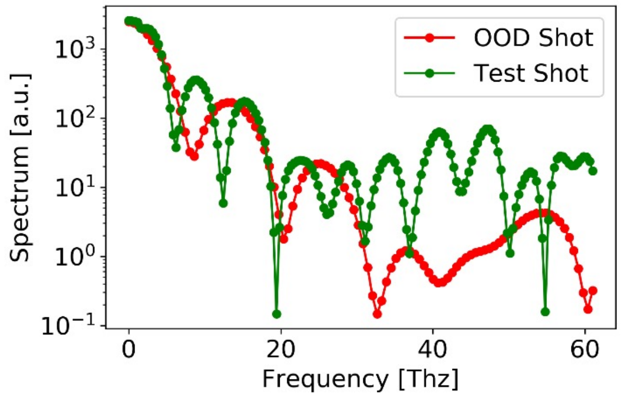


Mixup / Pairing images:

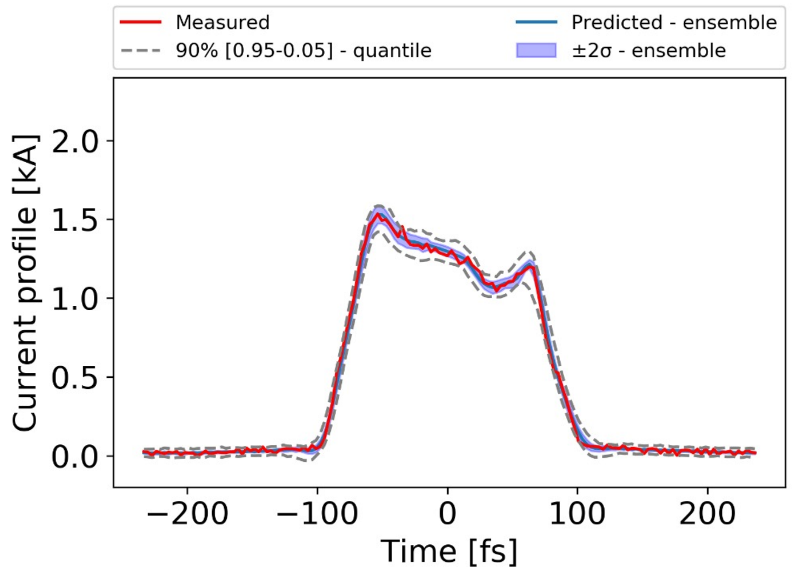
$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j$$
$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j$$

Out-of-Distribution (OOD) Robustness

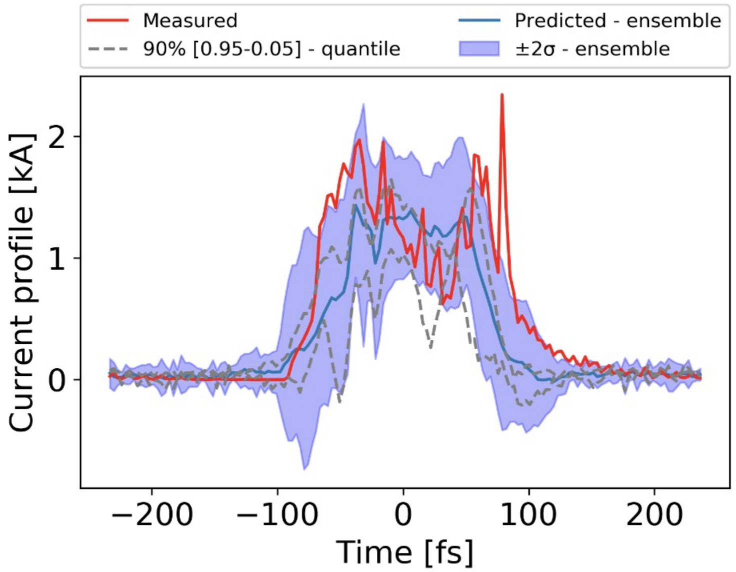
- Given OOD inputs (e.g. using the same machine in a different operation mode), it is necessary to understand how robust the ML model is and how well it generalizes on unfamiliar data.



Test shot within the trained distribution



Out-of-distribution



Out-of-Distribution → Higher Uncertainty

Convery, arxiv 2105.04654 (accepted to PRAB)

Other learning tasks

- Other supervised learning settings:
  - Multi-class or Multi-label.
  - Semi-supervised: make use of labeled and un-labeled data.
- Incremental learning – learns one instance at a time.
- Active learning - learning algorithm interactively query the system to get new data points.
- Transfer learning - model developed for a task is reused as the starting point for a model on a second task

Key points

- Data integration, selection, cleaning and pre-processing (normalization, outliers).
- Models – favor simple over complex.
- Interpreting results - avoid GIGO, uncertainty, robustness.

Thank you for your attention!

Questions?