Olga Fink / Alexandre Alahi

ML Basics - Part 2

Last week

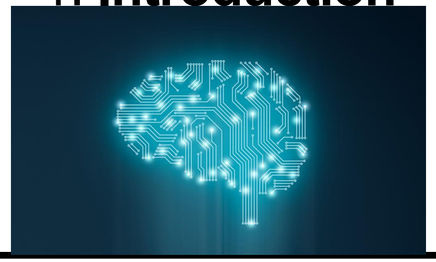
- Define ML ingredients
- Translate a problem to ML
  - Supervised vs. Unsupervised
  - Regression vs. Classification
  - Parametric vs. Non-parametric
- Linear regression
- Non-parametric KNN
- Logistic regression

Today

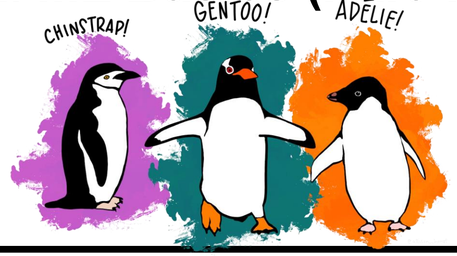
- Quick recap of classification with Logistic Regression
- Underfitting & Overfitting
- Performance Metrics
- Role of representation

Today

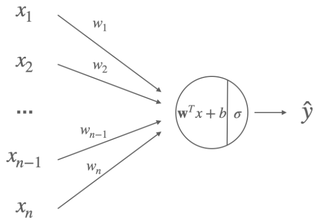
1. Introduction



2. ML Basics (Part 1)

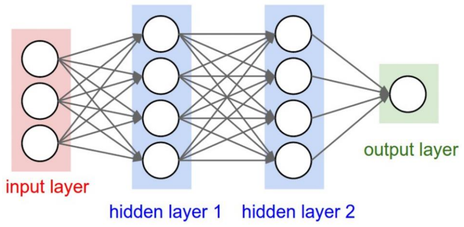


3. ML Basics (Part 2)

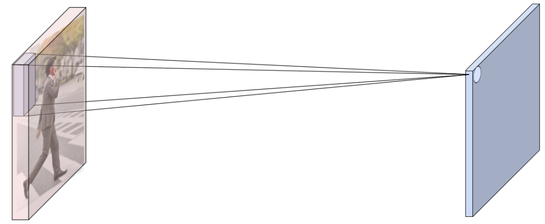


4. Role of Input

5. Deep Learning (Part 1)



6. Deep Learning (Part 2)



Underfitting & Overfitting

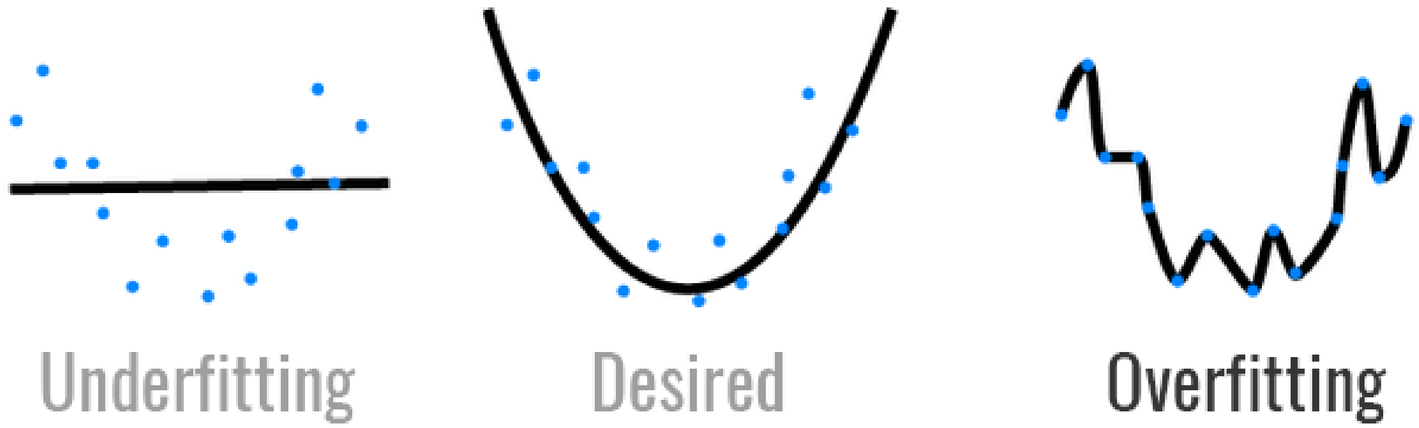
Olga Fink /Alexandre Alahi

Underfitting & Overfitting
Introduction

Goal of supervised ML models: generalise well on new data (based on the patterns learned from known data).

Two situations where it fails:

- Underfitting
- Overfitting



Underfitting & Overfitting

Underfitting

Underfitting: the model doesn't fit well on the training data.

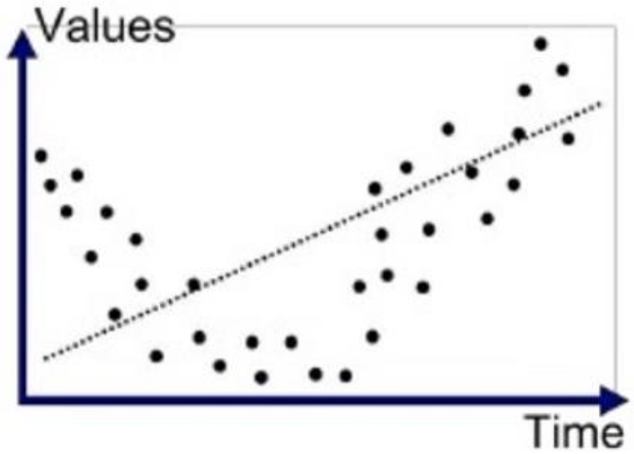
Reason: model too simple → can't capture the underlying patterns within the data.

Underfitting & Overfitting

Underfitting

Underfitting: the model doesn't fit well on the training data.

Reason: model too simple → can't capture the underlying patterns within the data.



The model (the line) doesn't capture the U shape of the data set.

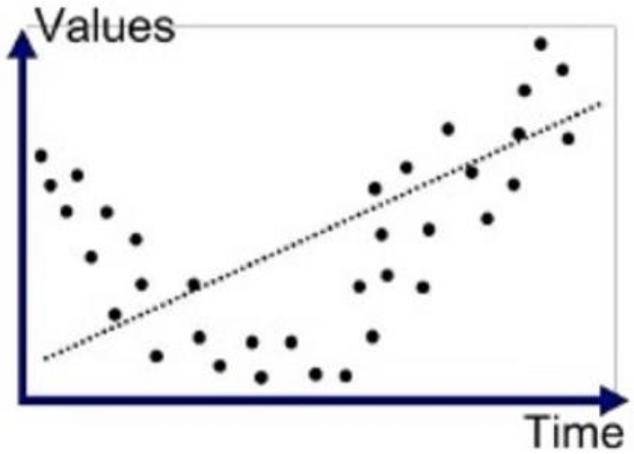
Olga Fink /Alexandre Alahi

Underfitting & Overfitting

Underfitting

Underfitting: the model doesn't fit well on the training data.

Reason: model too simple → can't capture the underlying patterns within the data.



The model (the line) doesn't capture the U shape of the data set.

→ **Solution**: choose a more complex algorithm/model to better fit the data.

Underfitting & Overfitting

Overfitting - Presentation

Q: What if the ML model is too complex?

Underfitting & Overfitting

Overfitting - Presentation

Q: What if the ML model is too complex?

Overfitting model:

- Fits well on training data
- Doesn’t generalise well to unknown data.

Reason: the model is too complex → it fits the noises and errors.

Olga Fink /Alexandre Alahi

Underfitting & Overfitting

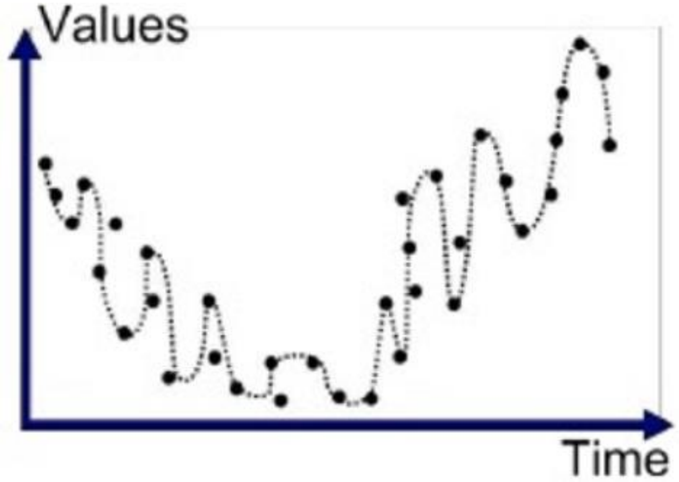
Overfitting - Presentation

Q: What if the ML model is too complex?

Overfitting model:

- Fits well on training data
- Doesn’t generalise well to unknown data.

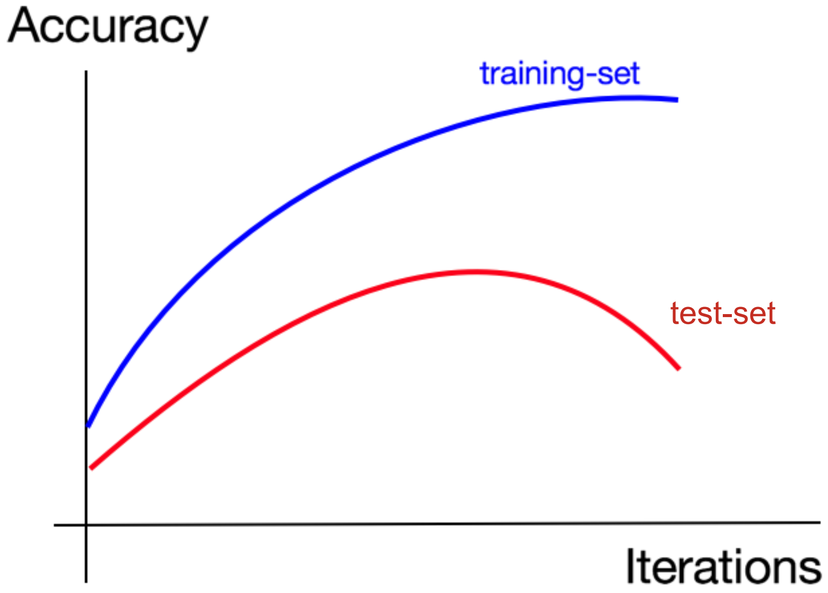
Reason: the model is too complex → it fits the noises and errors.



The model passes by every data point but doesn’t capture the U shape of the data set.

Underfitting & Overfitting

Overfitting - Detection

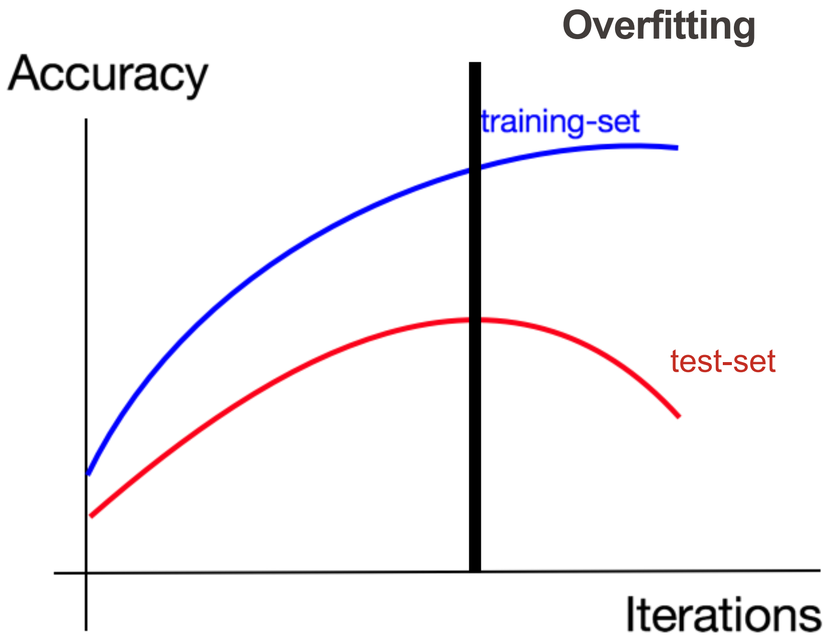


Plot: the evolution of the accuracy on the training set (train data) and the test set (unknown data) over the learning process.

Olga Fink /Alexandre Alahi

Underfitting & Overfitting

Overfitting - Detection



Plot: the evolution of the accuracy on the training set (train data) and the test set (unknown data) over the learning process.

Overfitting: When the accuracy on the training set increases, while the accuracy on the test set decreases.

Olga Fink /Alexandre Alahi

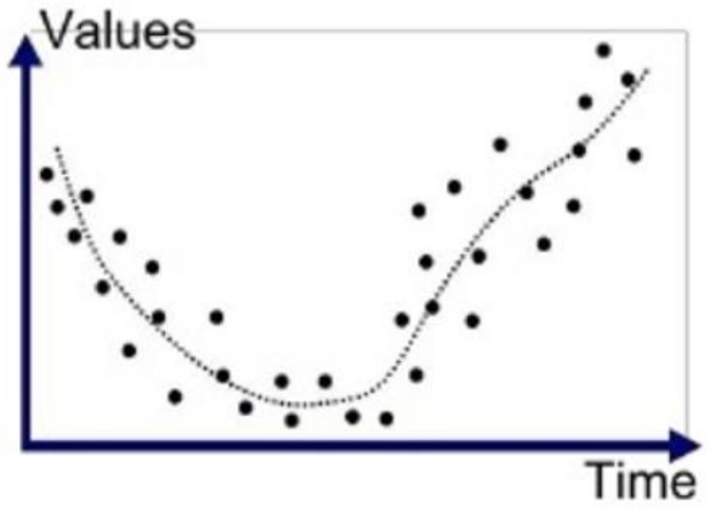
Underfitting & Overfitting

Overfitting - Solutions

Solutions:

- Simpler model → fit the data and not the noises and errors.
- More training data (→ less sampling noise)
- Add a regularisation term (common solution)

Goal: Find the equilibrium between fitting the training data and keeping the model simple enough to ensure it



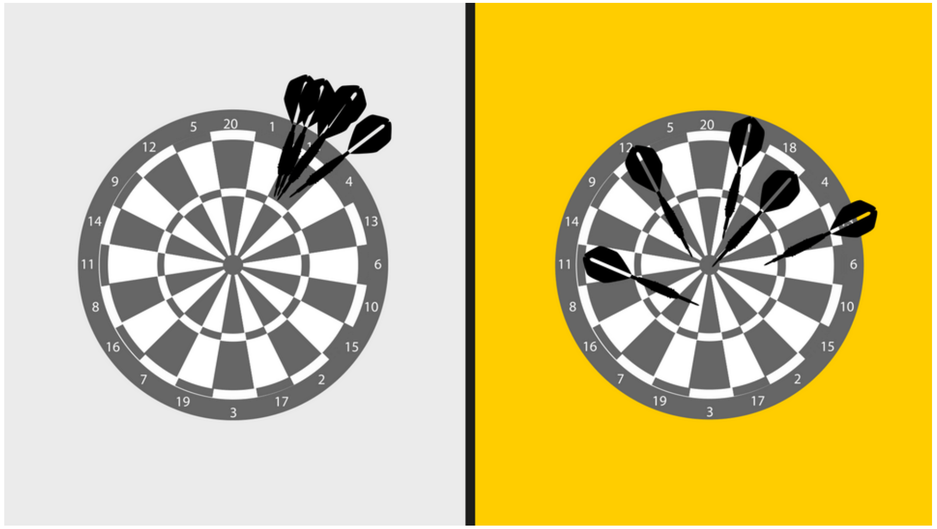
Underfitting & Overfitting

Bias & Variance - Introduction

Another perspective on underfitting and overfitting.

Definitions:

- **Bias** is a model's tendency to consistently learn the same wrong pattern.
- **Variance** is the tendency to learn random patterns unrelated to the underlying relationship in the data.
- **Variance** measures how much a model's predictions vary when trained on different datasets.



High bias, low variance

High variance, low bias

Underfitting & Overfitting
Bias

Bias: difference between the mean prediction of the model and the mean target value.

- The more often the model gives the right prediction, the less biased it is.
- High-bias model → underfit the training data.

Cause: incorrect assumptions in the ML model → miss relations between the data and the output.

- Example: assuming the data is linear when it is quadratic

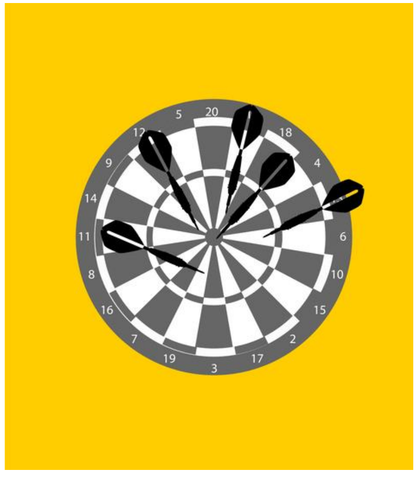
Underfitting & Overfitting
Variance

Variance: fluctuation around the mean target value in response to different training sets.

- The more stable the performance of a learner, the less its variance.
- High-variance → overfit the training data.

Cause: sensitivity to small variations in the training data → model the noise in the training data.

- A model with many degrees of freedom (e.g. a high-degree polynomial model).



Underfitting & Overfitting

Bias & Variance - Comparison

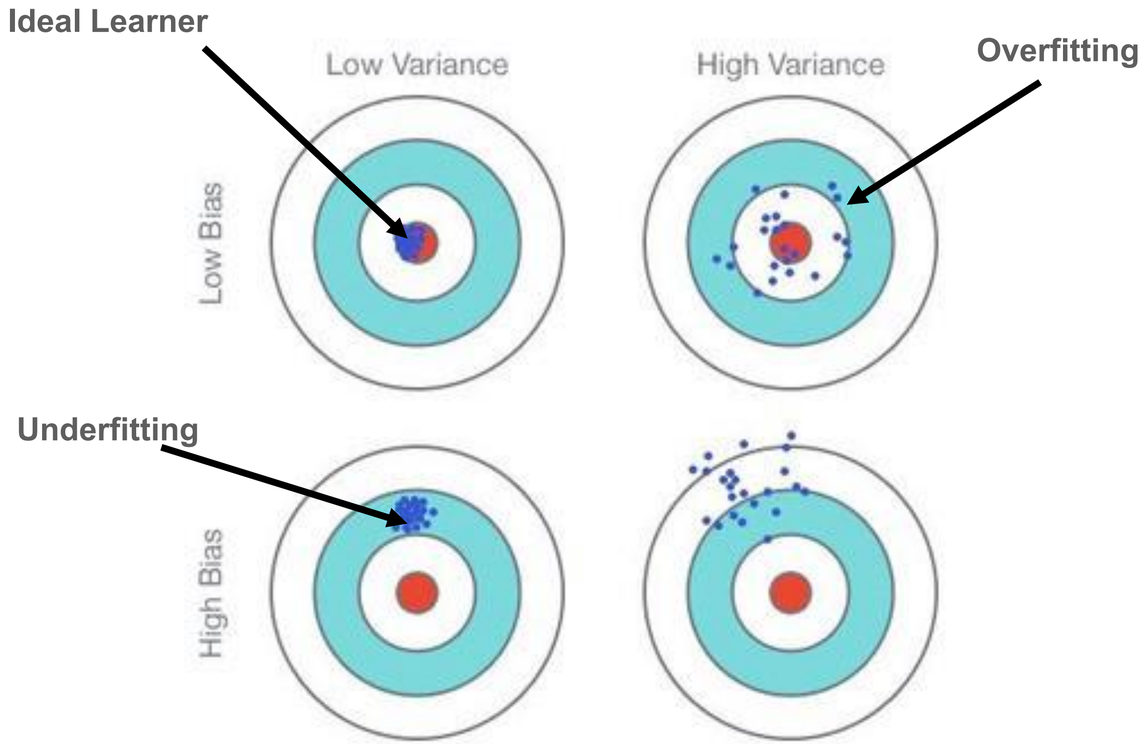


Fig. 1: Graphical Illustration of bias-variance trade-off , Source: Scott Fortmann-Roe., Understanding Bias-Variance Trade-off

Olga Fink /Alexandre Alahi

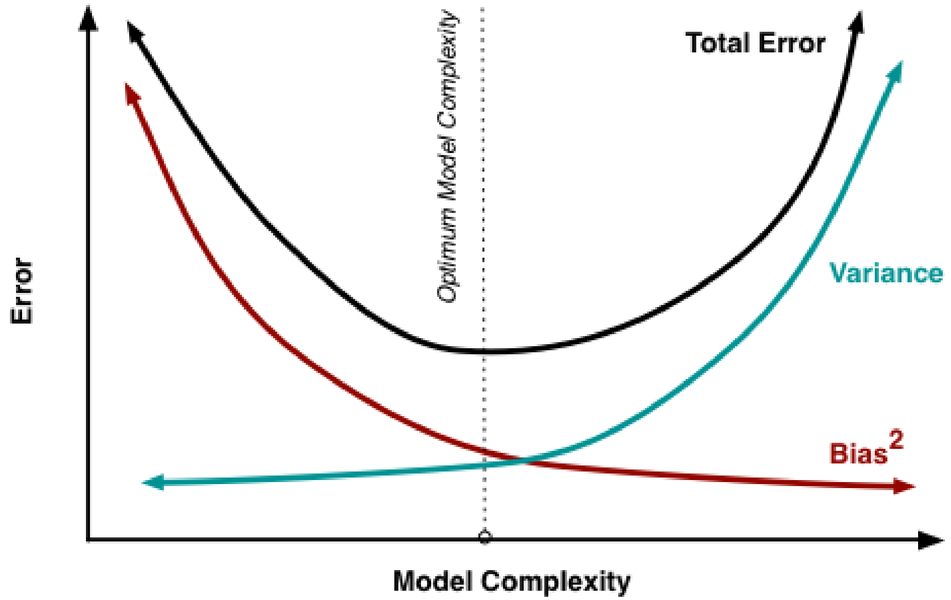
Underfitting & Overfitting
Bias & Variance - Trade-off

A good model should have low bias and low variance.

Trade-off:

- ↗Model's complexity ⇒ ↗variance and ↘bias
- ↘Model's complexity ⇒ ↘variance and ↗bias

→ find the best compromise



Note: To modify a model's complexity:

- Tune its hyper-parameters.

Olga Fink /Alexandre Alahi

Underfitting & Overfitting

Regularisation - Introduction

Q: A technique to prevent overfitting (≡ high variance)?

A: Add a **regularisation term** to the loss function.

$$J(\mathbf{w})_{regularized} = J(\mathbf{w}) + \lambda \sum_{j=1}^{n} w_j^2$$

where λ is the regularisation parameter (controls the strength of the regularisation).

It penalises the model complexity → incentivise to be less complex while fitting the data.

Underfitting & Overfitting
Regularisation - L1 & L2

L2 regularisation (≡ ridge regression):

$$J(\mathbf{w})_{regularized} = J(\mathbf{w}) + \lambda \sum_{j=1}^{n} w_j^2$$

There are other types of regularisation:

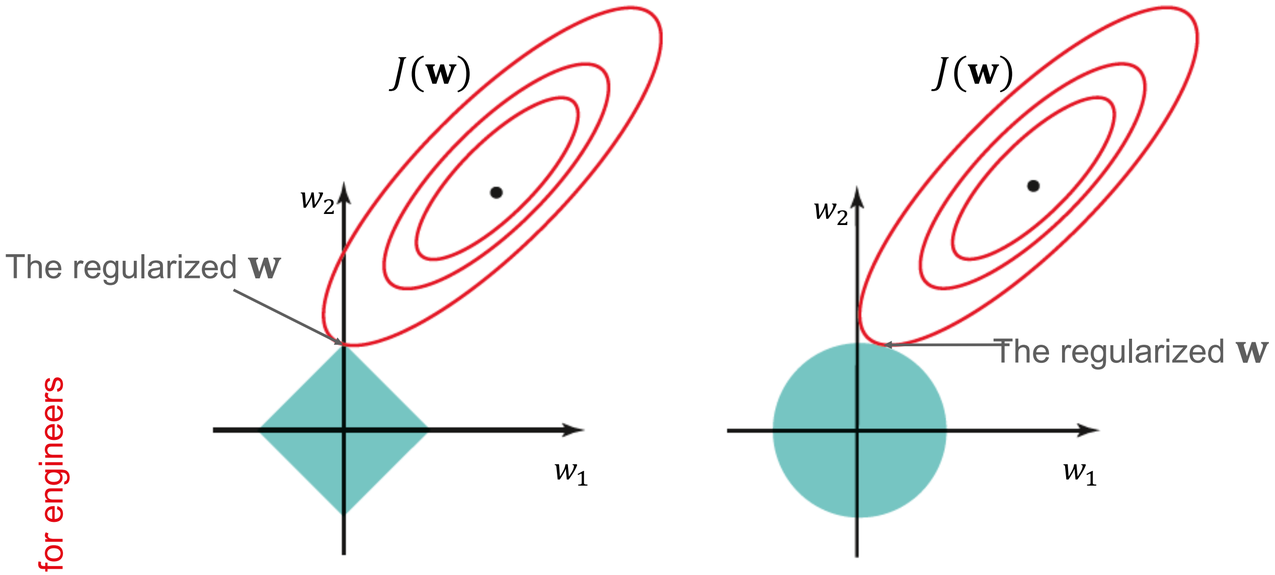
L1 regularisation (≡ lasso regression): $\lambda \sum_{j=1}^{n} w_j^2$ is replaced by $\lambda \sum_{j=1}^{n} |w_j|$.

Olga Fink /Alexandre Alahi

Underfitting & Overfitting
Regularisation - L1 vs. L2

L1 and **L2** both punish high values of $\mathbf{w}$.

L1 tends to set non-relevant features to zero → can be used for feature selection.



L1 Regularization L2 Regularization

- Blue areas: constrained regions by the regularisation.
- Red ellipses: cost as a function of $\mathbf{w}$.

Olga Fink /Alexandre Alahi

Underfitting & Overfitting
Regularisation - Next

- **L1** and **L2** are popular regularisation techniques
- In the following lectures, we will cover specific techniques of regularisation

Performance Metrics

Performance Metrics

Confusion Matrix - Binary Case

For categorical binary classification, the usual metrics are based on the confusion matrix:

- **TP**: True Positives (positive examples classified as positive)
- **TN**: True Negatives (negative examples classified as negative)
- **FP**: False Positives (negative examples classified as positive)
- **FN**: False Negatives (positive examples classified as negative)

		Class	
		Positive	Negative
Prediction	Positive	**TP**	**FP**
	Negative	**FN**	**TN**

Olga Fink /Alexandre Alahi

Performance Metrics

Confusion Matrix - Binary Case

There are 2 additional values:

- **P = TP + FN**: Condition positive (positive examples)
- **N = FP + TN**: Condition negative (negative examples)

		Class	
		Positive	Negative
Prediction	Positive	**TP**	**FP**
	Negative	**FN**	**TN**
		P = TP + FN	**N** = FP + TN

Olga Fink /Alexandre Alahi

Performance Metrics

Confusion Matrix - Binary Case

Example - Spam filter that categorises email as spam (true) or non-spam (false):

- 200 spams are classified as spams.
- 100 non-spams are classified as spams.
- 300 spams are classified as non-spams.
- 400 non-spams are classified as non-spams.

		Class	
		Spam	Non-Spam
Prediction	Spam	**200**	**100**
	Non-Spam	**300**	**400**
		P = 500	**N** = 500

Olga Fink /Alexandre Alahi

Performance Metrics

Confusion Matrix - Multi-class case

- The confusion matrix also works for multi-class datasets.
- Example: Confusion matrix of a palmer penguins classifier (species: Chinstrap, Gentoo, Adélie).

		Class		
		Chinstrap	Gentoo	Adélie
Prediction	Chinstrap	**12**	**1**	**1**
	Gentoo	**0**	**16**	**0**
	Adélie	**0**	**1**	**16**

Olga Fink /Alexandre Alahi

Olga Fink /Alexandre Alahi

Performance Metrics

Accuracy - Theory

Accuracy: Percentage of correctly classified samples.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{P + N}$$

Appropriate metric when:

- Classes are not unbalanced
- The errors FP and FN have the same importance

Performance Metrics

Accuracy - Example

Accuracy: Percentage of correctly classified spams.

$$Accuracy = \frac{TP + TN}{P + N}$$

$$= \frac{200+400}{500+500}$$

$$= 0.6$$

		Class	
		Spam	Non-Spam
Prediction	Spam	200	100
	Non-Spam	300	400
		P = 500	N = 500

Olga Fink /Alexandre Alahi

Performance Metrics

Accuracy - Drawback

- Accuracy doesn't totally translate the behaviour of classifiers.
- **Example**: The 2 classifiers below have the same accuracy (0.6) but behave differently:
  - Left: Classify everything as Non-Spam.
  - Right: Strong non-spam recognition rate, but weak spam recognition rate

		Class	
		Spam	Non-Spam
Prediction	Spam	**0**	**0**
	Non-Spam	**100**	**900**
		P = 100	N = 900

		Class	
		Spam	Non-Spam
Prediction	Spam	**50**	**50**
	Non-Spam	**50**	**850**
		P = 100	N = 900

Olga Fink /Alexandre Alahi

Performance Metrics

Precision - Theory

Precision: Percentage of samples classified as positives that are truly positives.

$$Precision = \frac{TP}{TP + FP}$$

Olga Fink /Alexandre Alahi

Performance Metrics
Precision - Example

Precision: Percentage of samples classified as spams that are truly spams.

$$Precision = \frac{TP}{TP + FP}$$

$$= \frac{200}{200+100}$$

$$= 0.667$$

		Class	
		Spam	Non-Spam
Prediction	Spam	200	100
	Non-Spam	300	400
		P = 500	N = 500

Olga Fink /Alexandre Alahi

Performance Metrics

Recall - Theory

Recall: Percentage of positive samples that are correctly classified as positives.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{P}$$

Performance Metrics
Recall - Example

Recall: Percentage of spams that are correctly classified as spams.

$$Recall = \frac{TP}{P}$$

$$= \frac{200}{200+300}$$

$$= 0.4$$

		Class	
		Spam	Non-Spam
Prediction	Spam	200	100
	Non-Spam	300	400
		P = 500	N = 500

Olga Fink /Alexandre Alahi

Performance Metrics

Precision & Recall - Drawback

- Precision and recall don't totally translate the behaviour of classifiers.
- **Example**: The 2 classifiers below have the same precision (0.667) and recall (0.4) but behave differently:
  - Very different negative recognition rates (strong on left, nil on right).
- Accuracy would clearly show this.

		Class: Spam	Class: Non-Spam
Prediction	Spam	200	100
	Non-Spam	300	**400**
		P = 500	N = 500

		Class: Spam	Class: Non-Spam
Prediction	Spam	400	300
	Non-Spam	100	**0**
		P = 500	N = 100

Olga Fink /Alexandre Alahi

Performance Metrics

F1-Score - Theory

F1 score: Single number that combines precision and recall

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Precision and recall can be weighted differently, if one is more important than the other.

Olga Fink /Alexandre Alahi

Performance Metrics
F1-Score - Example

F1 score: Single number that combines precision and recall

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

$$= 2 \cdot \frac{0.667 \cdot 0.4}{0.667 + 0.4}$$

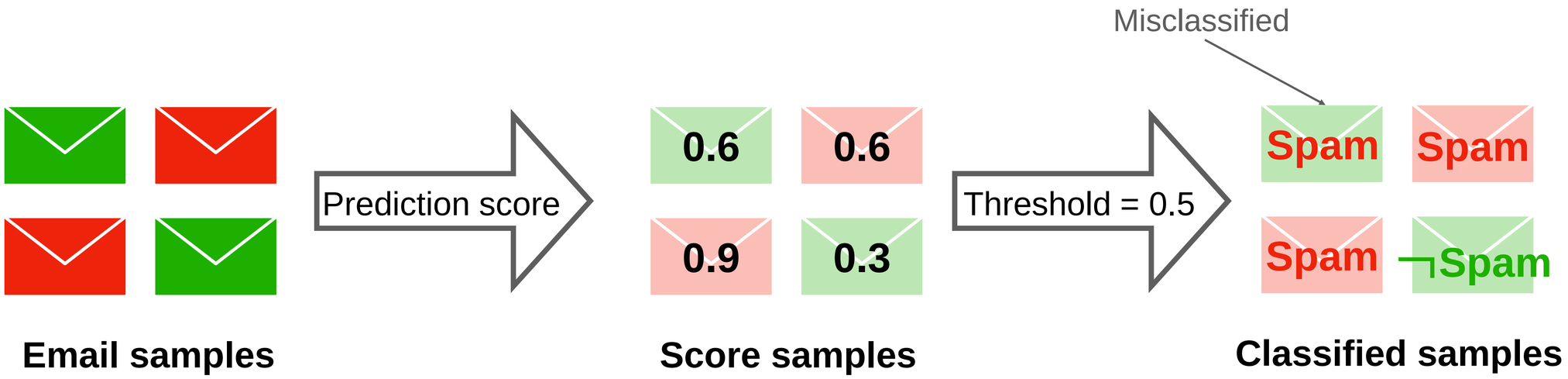
$$= 0.5$$

		Class	
		Spam	Non-Spam
Prediction	Spam	200	100
	Non-Spam	300	400
		P = 500	N = 500

Olga Fink /Alexandre Alahi

Performance Metrics
ROC Curve

- Many classifiers output a prediction score for every sample.
- Then, it uses a threshold to categorise the samples. If the score's sample is higher than the threshold, it is categorise as positive (negative if it is lower).



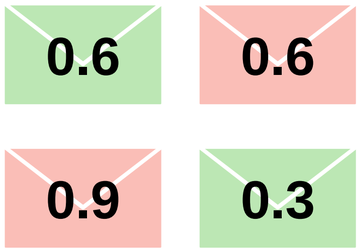
Olga Fink /Alexandre Alahi

Performance Metrics
ROC Curve

- Increasing the threshold, decrease the true positive rate but also decrease the false positive rate.
- Example: spam = positive.

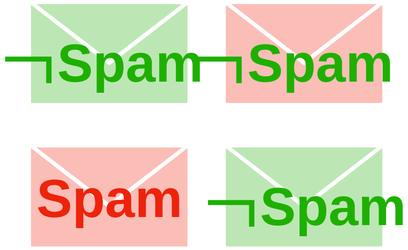
Email samples

Prediction score



Score samples

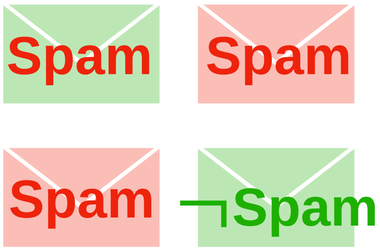
Threshold = 0.7



$$Recall = \frac{TP}{P} = \frac{1}{2}$$

$$FPRate = \frac{FP}{N} = 0$$

Threshold = 0.5



$$Recall = \frac{TP}{P} = \frac{2}{3}$$

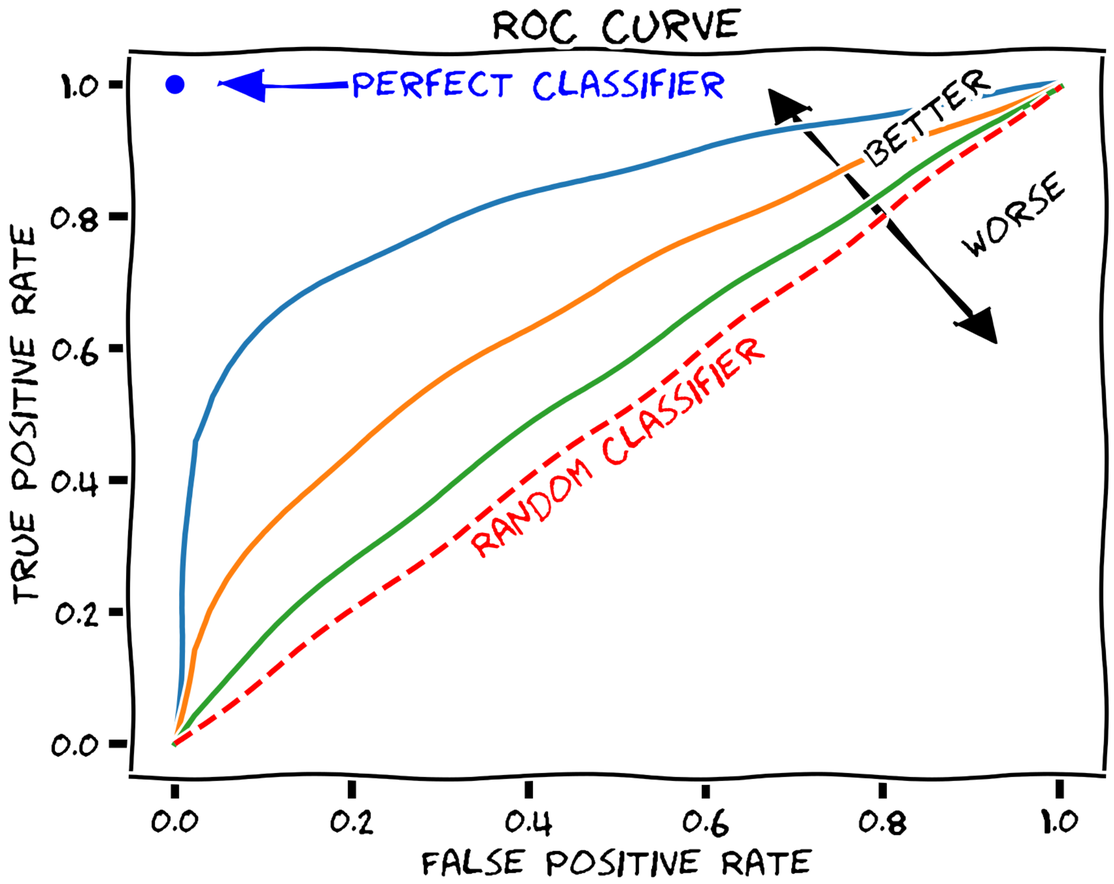
$$FPRate = \frac{FP}{N} = \frac{1}{2}$$

Classified samples

Olga Fink /Alexandre Alahi

Performance Metrics
ROC Curve

- **ROC**: plot that illustrates the ability of a binary classifier system as its threshold is varied.
- The **a**rea **u**nder the ROC **c**urve (AUC) can act as a metric.
  - The maximum AUC is 1.
  - Random predictions give AUC = 0.5



http://playground.tensorflow.org/

Now

- What is an input representation?
- What is a feature?
- Handling different types of features
- Missing values
- Feature expansion
- Examples of representations

What is an input representation in ML?

A representation is a mathematical form (*e.g.*, a vector)

- It describes an observation in the real-world (*e.g.*, an image, waveforms, signals, …)
- It is used for subsequent steps (*e.g.*, a classifier) to produce the outcome of interest (*e.g.*, recognizing objects)
- It is often more compact than the original observation (lower dimensionality)
- It is potentially more robust to nuisances
- With a good representation, subsequent steps should be easier

“Solving a problem simply means representing it so as to make the solution transparent.”

- H. Simon, Sciences of the Artificial -

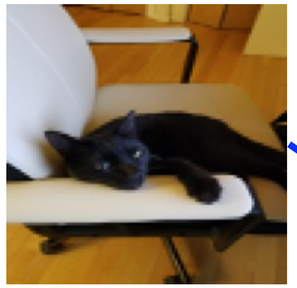
Olga Fink /Alexandre Alahi

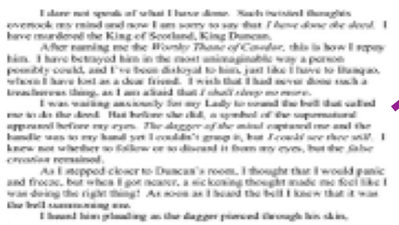
What is an input representation in ML?

“Solving a problem simply means representing it so as to make the solution transparent.”

- H. Simon, Sciences of the Artificial -

The machine learning pipeline

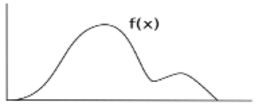


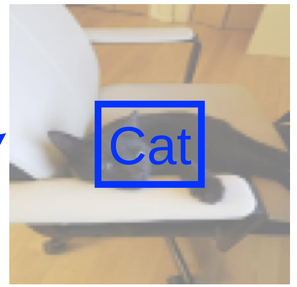


Representation

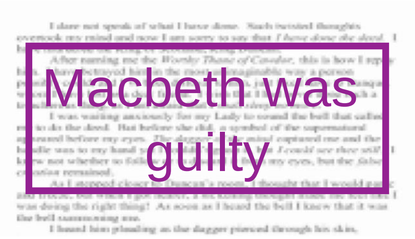
[81 20 84 64 58 39 17 54 72 15]

ML algorithm





Hey, what's the weather today?

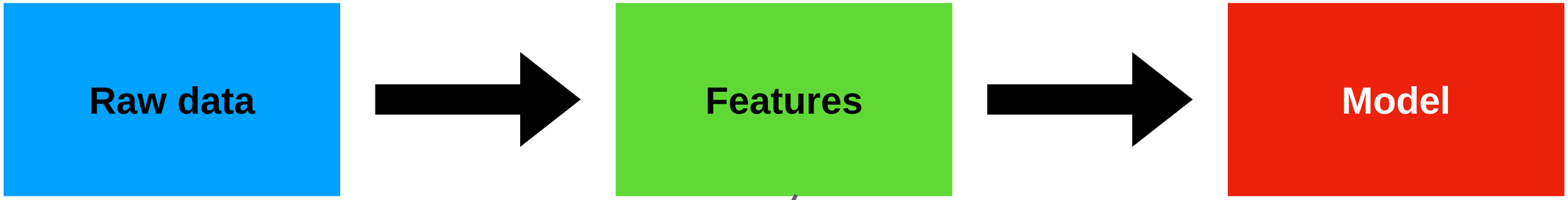


What is a Feature?

A feature vector is a representation,
***i.e.*, a mathematical form that describes an observation in the real-world…**

What is a Feature?
Feature engineering

Feature engineering means transforming **raw data** into a **feature vector** that represents the underlying data well



Designing clever features is a key part of the machine learning pipeline

For simple models, most of the “heavy lifting” is done there

Olga Fink /Alexandre Alahi

Features

Feature engineering

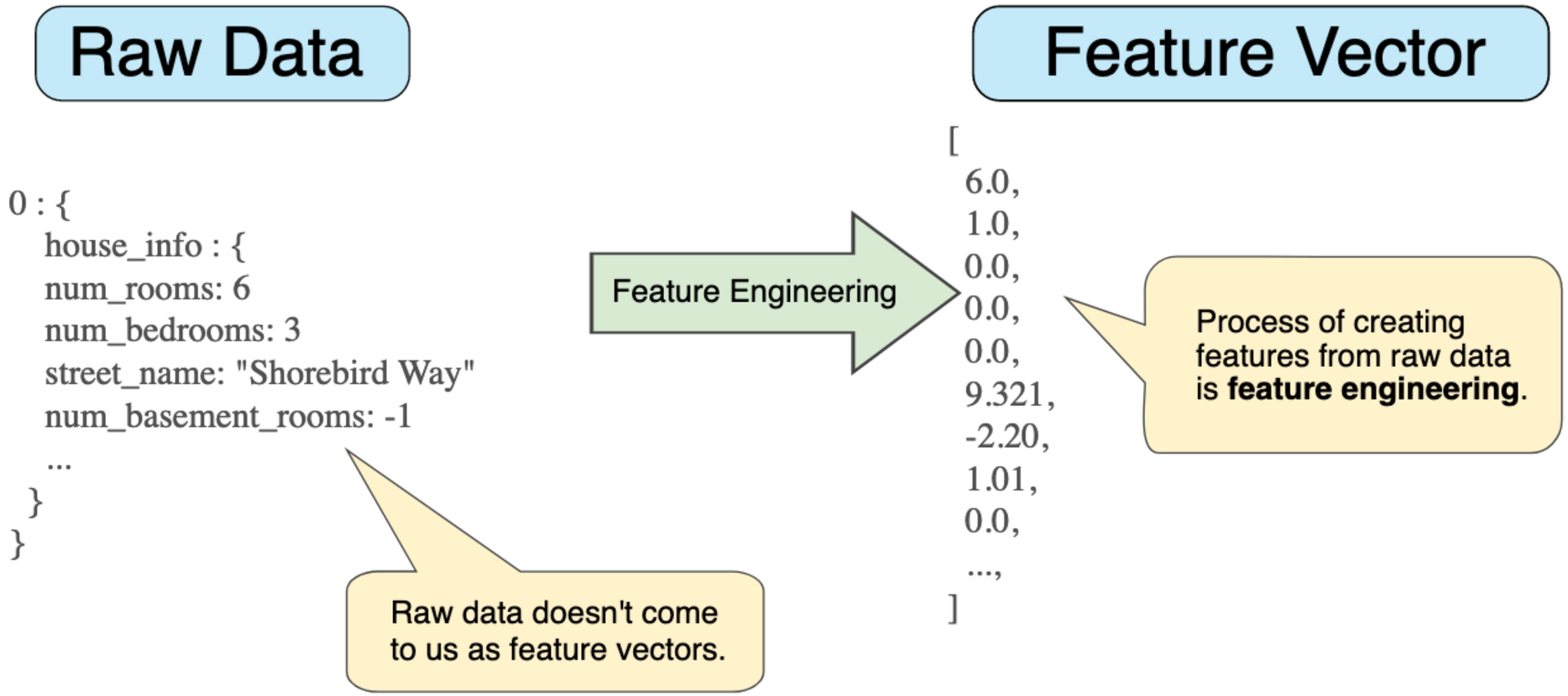


Image credit: [Google Machine Learning Crash Course]

Features
Types of features

Different types of features:

- **Numerical / continuous**
  - *e.g.,* height, temperature, price, …
- **Ordinal**
  - *e.g.,* “like”, “somewhat like”, “neutral”, “somewhat dislike”, “dislike”
- **Categorical**
  - *e.g.,* color, species, …

Feature type	Order	Scale
Numerical	Yes	Yes
Ordinal	Yes	No
Categorical	No	No

Features
Preprocessing

Preprocessing: the process of transforming raw feature vectors into a representation that is more suitable for ML algorithms

Techniques differ depending on type of feature:

- Numerical, ordinal and categorical features need to be handled differently

Olga Fink /Alexandre Alahi

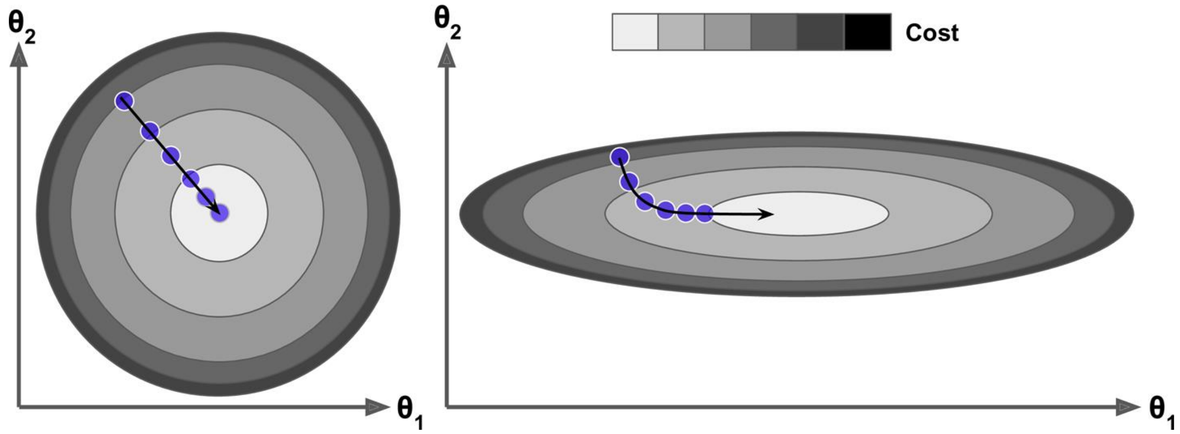
Numerical features

Numerical features
Data normalization

Data normalization / feature scaling: Normalize features (bring them all to the same scale)

Crucial step in preprocessing:

- Many classifiers (such as KNN) rely on distance metrics
- Gradient descent will converge faster
- Coefficients are penalized appropriately (in the case where regularization is applied)



A. Géron "Hands-on Machine Learning with Scikit-Learn and TensorFlow", 2017

Numerical features
Data normalization - Example

Example:

KNN with Palmer Penguins dataset

Features: body mass & bill length

Q: Which feature matters the most for the distance metric?

A:

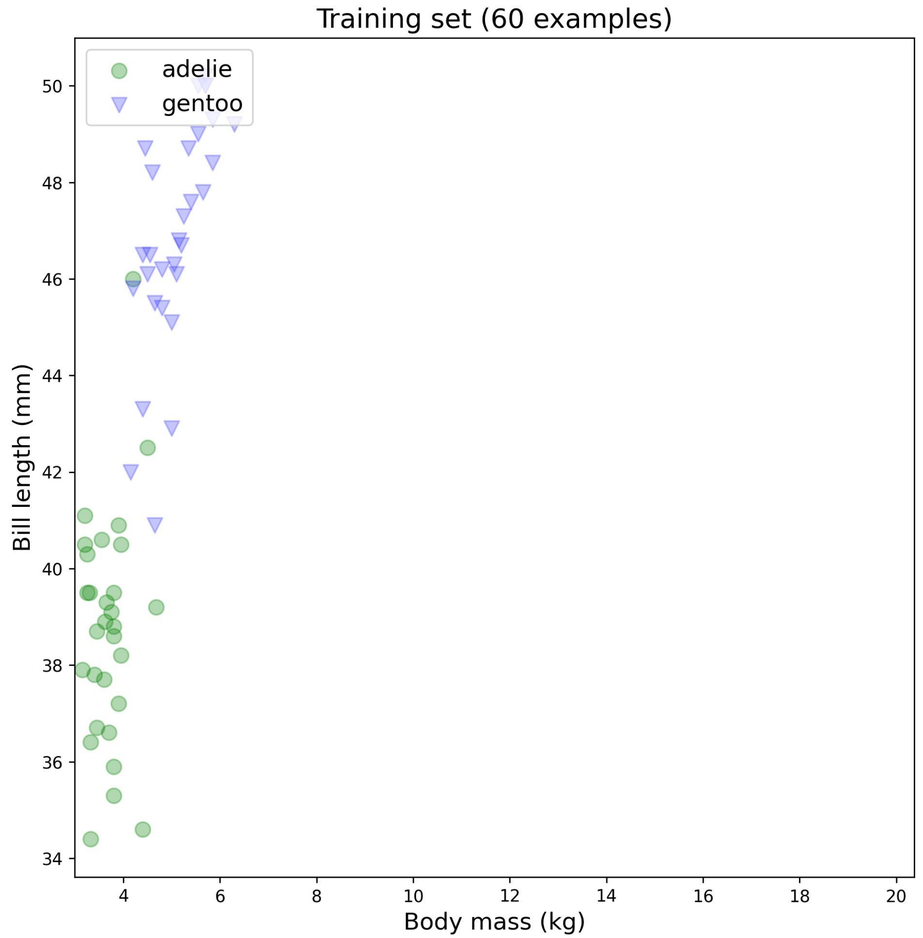
- If body mass in **g** and bill length in **m**

  → **body mass** matters more

- If body mass in **kg** and bill length in **mm**

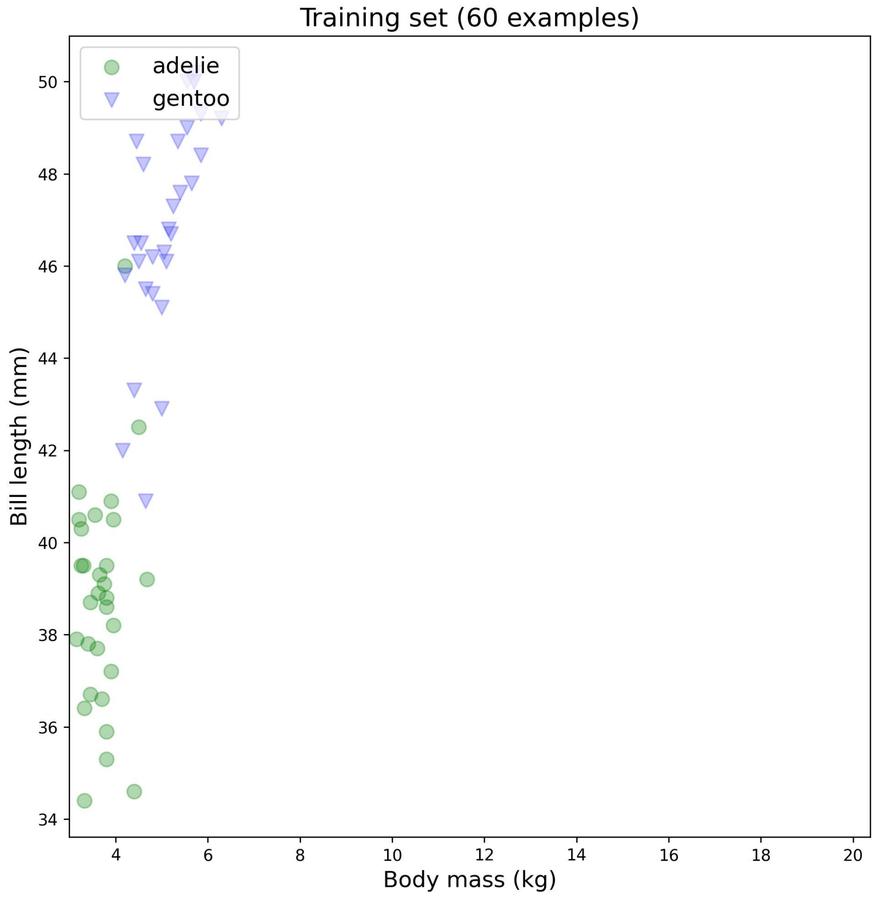
  → **bill length** matters more

With normalization, the units of the features stop playing an important role in the model accuracy

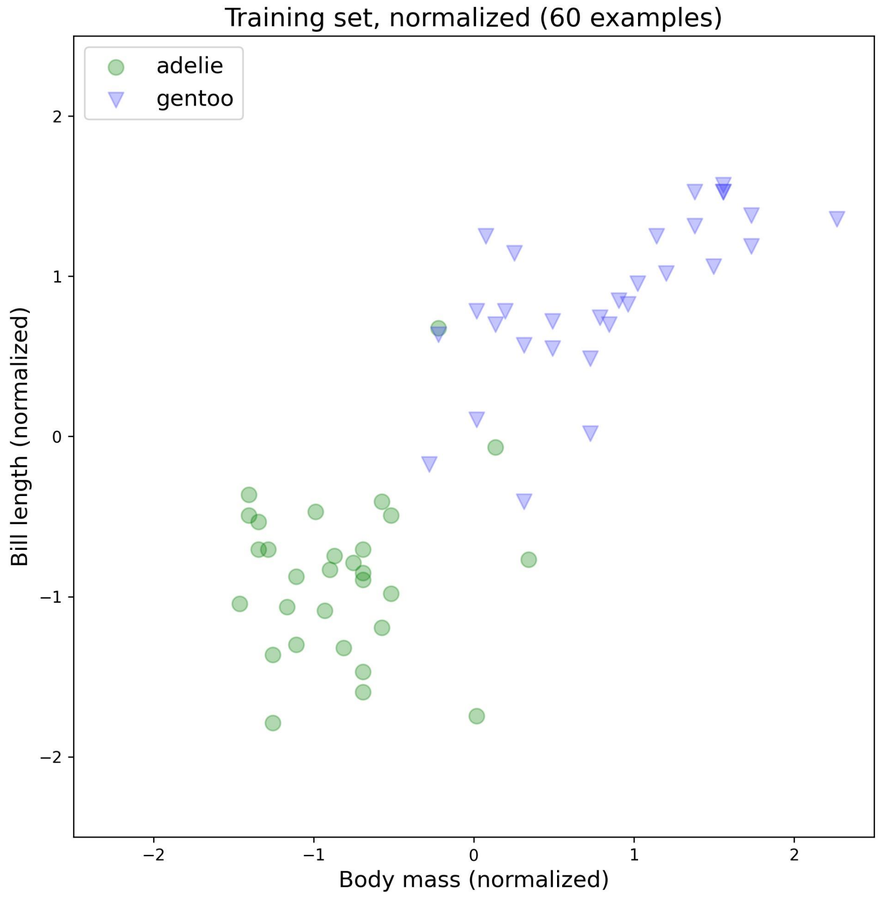


Numerical features

Data normalization - Example



Normalization



Olga Fink /Alexandre Alahi

Numerical features

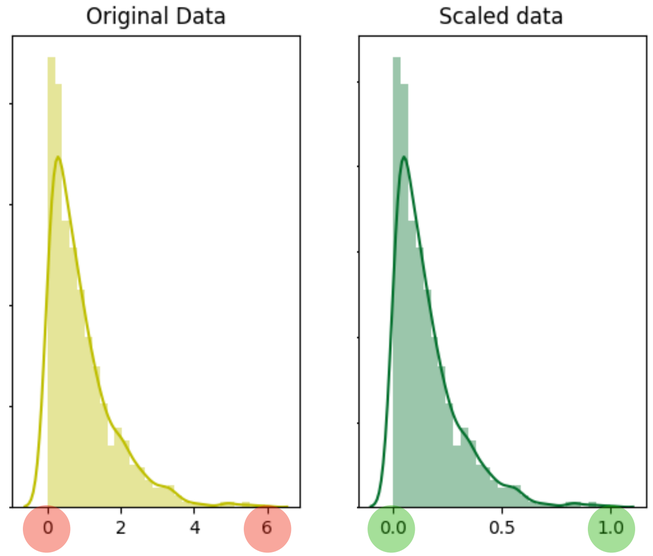
Data normalization - Methods

Min-max scaling:

Scale each feature to the range [0, 1]

For each feature:

$$x_{norm} = \frac{x - min(x)}{max(x) - min(x)}$$

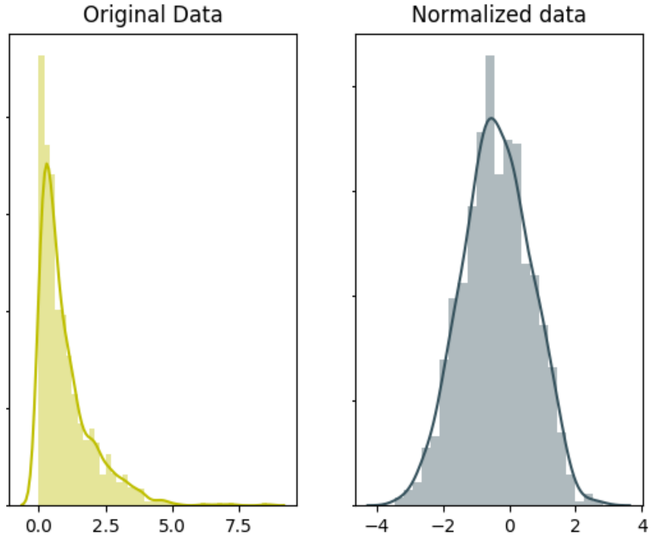


Z-Score Normalization / Standardization:

Set mean of each feature (**μ**) to 0, standard deviation (**σ**) to 1

For each feature:

$$x_{norm} = \frac{x - \mu_x}{\sigma_x}$$



Numerical features
Data normalization - Outliers

Features may have outliers or follow some heavy-tailed distribution

- e.g., **urban area population**:
  - most urban areas have a few thousands inhabitants
  - a handful of urban areas have tens of millions of inhabitants (New York, Tokyo, …)

Numerical features
Data normalization - Outliers

If the dataset has outliers, then:

- With **min-max** scaling:
  - typical values are scaled to a very small interval

- With **Z-score** standardization:
  - mean & standard deviation is not meaningful for heavy-tailed data
  - outliers will still have very large values

=> **Outliers affect the quality of the scaling**

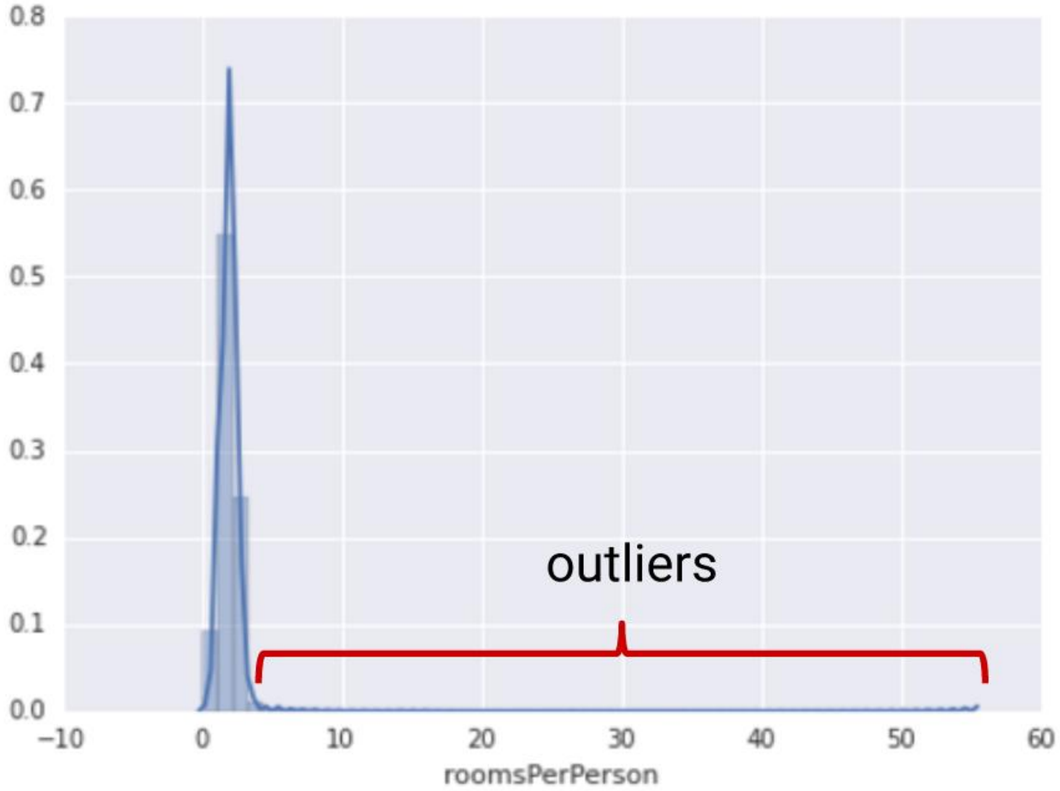


Image credit: [Google Machine Learning Crash Course]

Numerical features
Handling outliers

To handle outliers:

Logarithmic scaling: take the log of every value

- ⚠ make sure all values are positive
- e.g., $\boldsymbol{x} = [1,10,5,600]$

  $\rightarrow \log(\boldsymbol{x}) = [0,2.3,1.6,6.4]$

Value clipping: clip all values above / below an arbitrary threshold

- e.g., for clip value of [-10, 10]:
  - $\boldsymbol{x} = [0,-3,-1294,5,10320]$

    $\rightarrow clip(\boldsymbol{x}) = [0,-3,-10,5,10]$

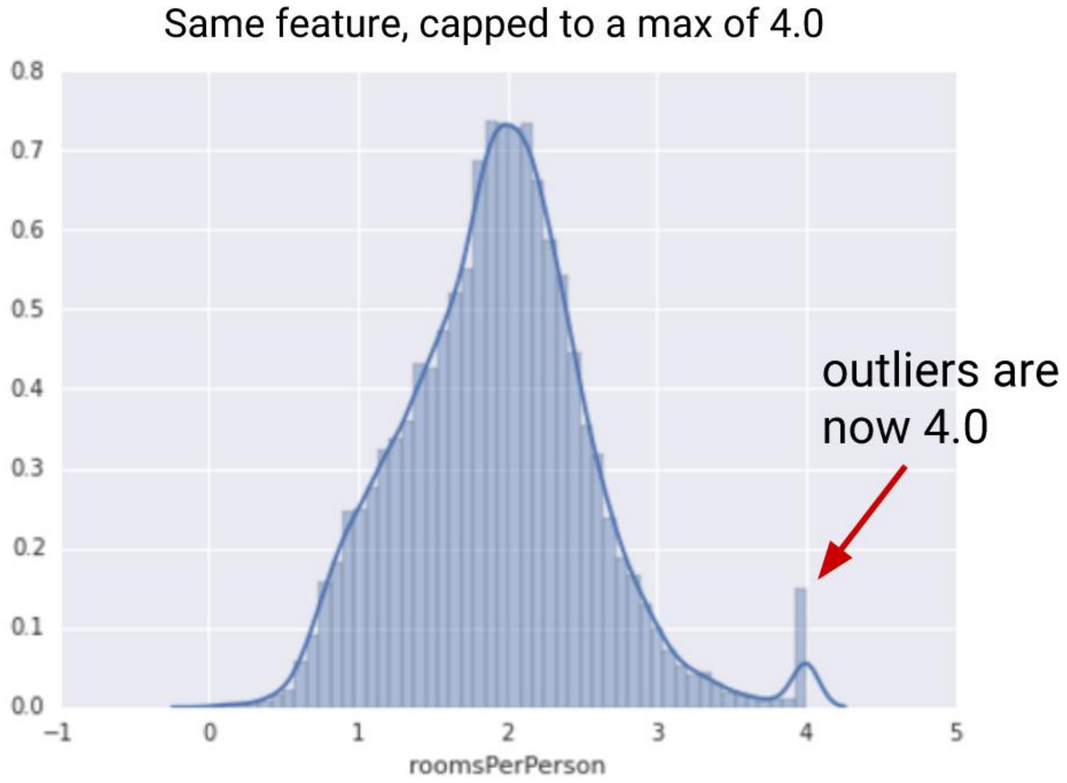


Image credit: [Google Machine Learning Crash Course]

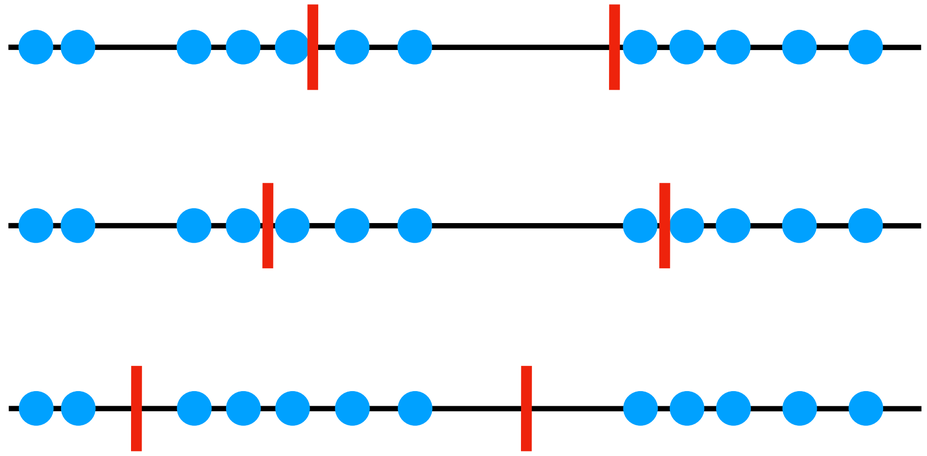
Numerical features
Binning

Binning / discretization: partition continuous features into discrete values

- e.g., **age:** instead of using each person's age as a number, we may want to use age ranges instead: 0-9, 10-19, 20-35, …

Common types of binning:

- **Uniform:** all bins have identical widths
- **Quantile:** all bins have the same number of points
- **Clustered:** a clustering algorithm (seen later in this course) is used to separate values



Next week

- Continue on the role of representation
- Deep learning

End

Features
Resources

- Google Machine Learning Crash Course:
  https://developers.google.com/machine-learning/crash-course/representation/feature-engineering

- scikit-learn preprocessing documentation:
  https://scikit-learn.org/stable/modules/preprocessing.html

- Kaggle feature engineering tutorial:
  https://www.kaggle.com/learn/feature-engineering

Glossary

- **KNN: K-Nearest Neighbors**
- **ML: Machine Learning**
- **RMSE: Root Mean Square Error**
- **MSE: Mean Square Error**
- Toy problem/example: a problem without scientific interest but useful for illustrating a concept

Appendix
Notation

$\mathbb{X}$: A set (ex: dataset)

$\mathbf{X}$: Features of $\mathbb{X}$

$\mathbf{Y}$: Labels of $\mathbb{X}$

$\boldsymbol{x}^{(i)}$: Element i of the set $\mathbb{X}$ (ex: datapoint)

a : A scalar

$\mathbf{a}$: A vector

$\mathbf{A}$: A matrix or a tensor

a_i : Element i of vector $\mathbf{a}$, with indexing starting at 1

$A_{i,j}$: Element (i, j) of matrix $\mathbf{A}$

$\mathbf{A}_{i,:}$: Row i of matrix $\mathbf{A}$

$\mathbf{A}_{:,i}$: Column i of matrix $\mathbf{A}$

$A_{i,j,k}$: Element (i, j, k) of a 3D tensor $\mathbf{A}$

$\mathbf{A}_{:,:,i}$: 2D slice of a 3D tensor $\mathbf{A}$

Appendix
Cost Function - Disambiguation

In this class:

J : cost function

=> average of loss over a single iteration

In ML literature:

- loss function L
- cost function J
- error function E

are sometimes used interchangeably, and

sometimes used like they are in this class