8th November 2021

ML 4 Time-series: Essential Methodology

Dr. Andrew P. Creagh | Dr. Anshul Thacker | Prof. David A. Clifton

Institute of Biomedical Engineering (IBME),
Department of Engineering Science,
Old Road Campus Research Building (ORCRB),
University of Oxford

Background

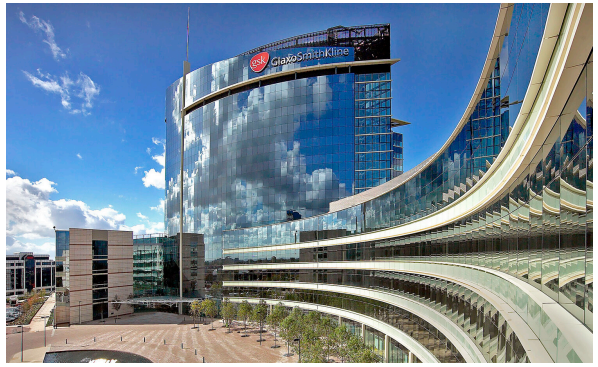
BA, BAI, MAI
Trinity College, Dublin, IE

DPhil.
University of Oxford, UK

F. Hoffmann-La Roche
Basel, CH

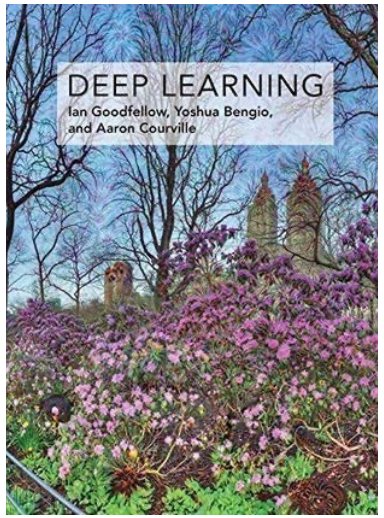
Postdoctoral Research Associate
IBME, Oxford, UK

Junior Research Fellow
St Cross College, Oxford, UK



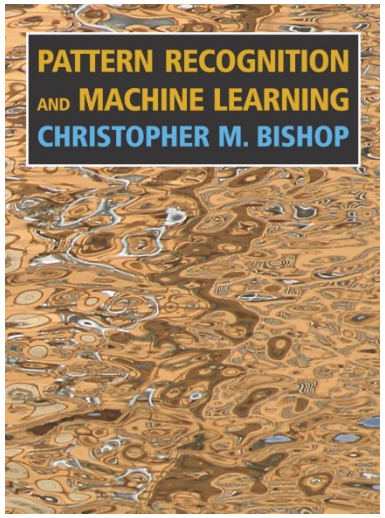
Postdoctoral Fellow
GSK, London, UK

Resources



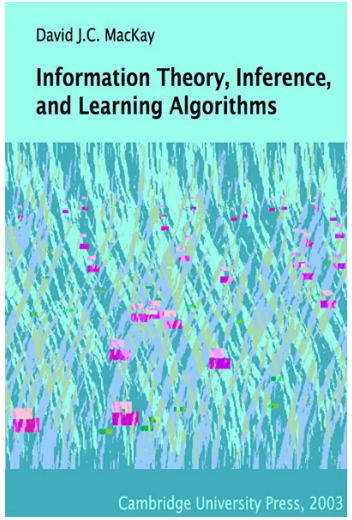
A well-written introduction to all things deep learning (DL), from leaders in the field of theoretical DL.

Very light maths



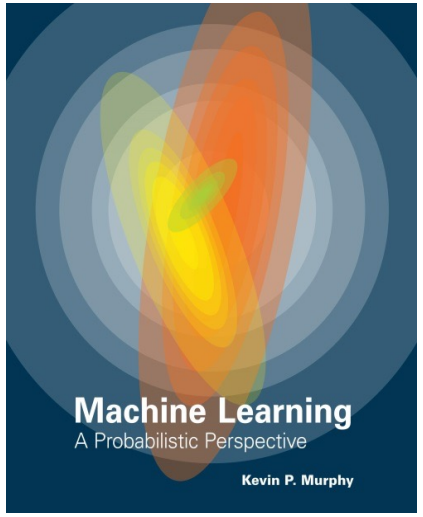
A core classic describing most non-DL algorithms. Very good for one's general understanding.

Reasonably "maths-y"



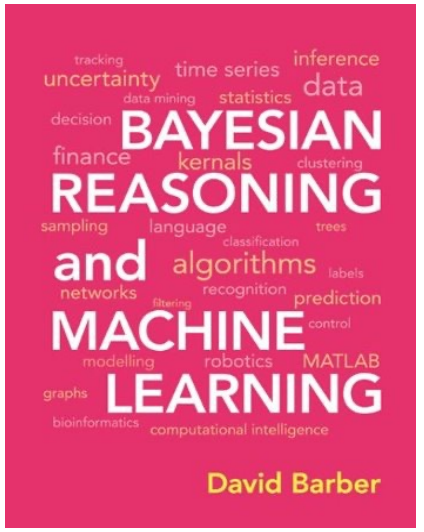
Another core classic, from one of the field's (sadly departed) foundational thinkers – good, even though it's from Cambridge

Reasonably "maths-y"



Seriously comprehensive, one of the best books for general machine learning. Excellent examples.

Really rather "maths-y"



Big, bad, and Bayesian. Everything you'd like to know about Bayesian machine learning. Great for time-series analysis.

Seriously "maths-y"

mild — medium — hot — extra hot

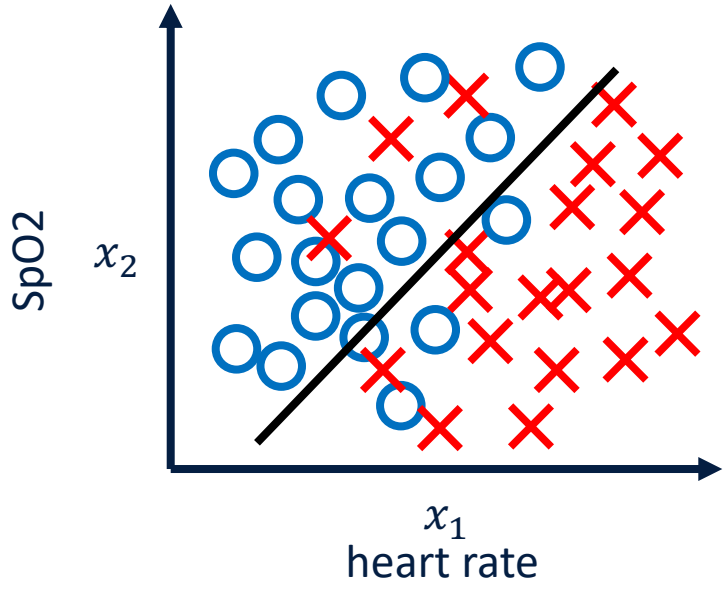
ML applied to healthcare data

- Many (but not all) tasks in healthcare can be phrased as either **classification** or **regression** problems, given some input $\mathbf{X}$

Classification

find: $p(C_i \mid \mathbf{X})$ for various classes C_i

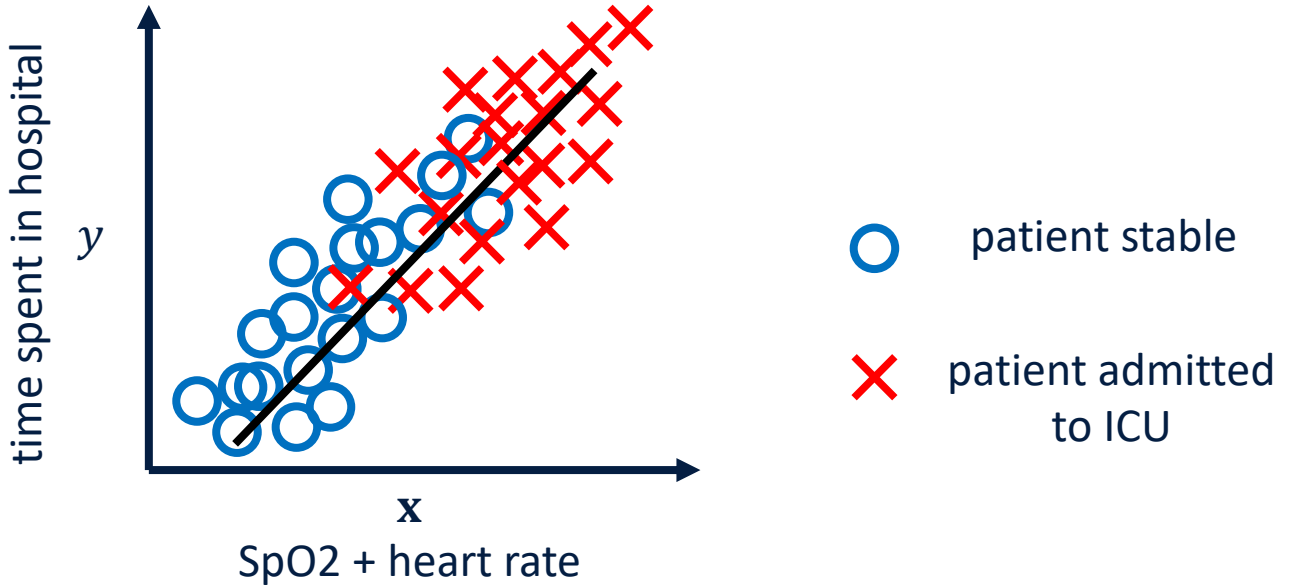
e.g., after measurements $\mathbf{X}$ *have been taken, define* C_1 *to be "patient stable" and define* C_2 *to be "patient admitted to ICU"*



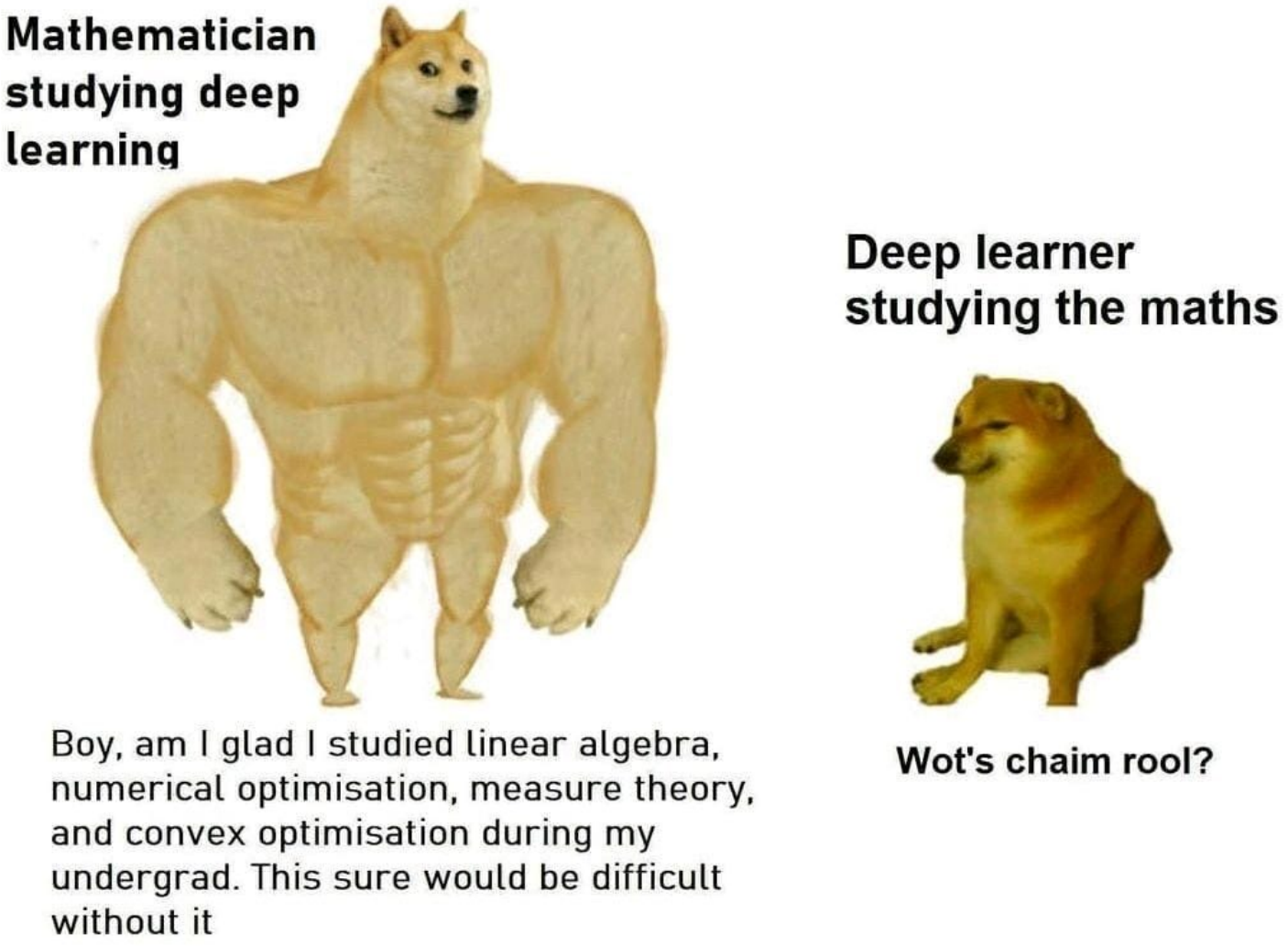
Regression

find $y = f(\mathbf{X})$

e.g., after measurements $\mathbf{X}$ *have been taken, define* y *to be total time spent in hospital, as a function of* $\mathbf{X}$ measurements

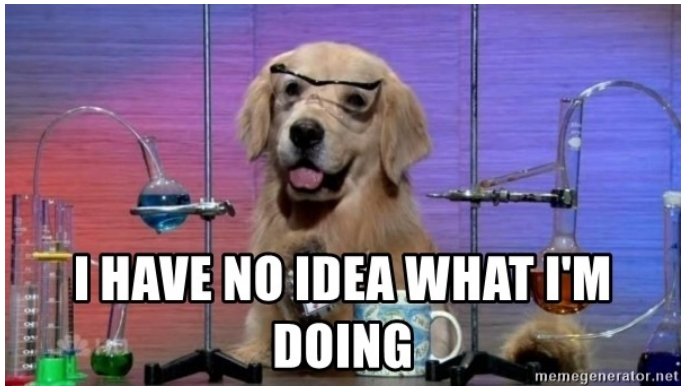


CDT in Health Data Science



Credit: https://www.facebook.com/convolutionalmemes/

Machine Learning 101



Find the function that best explains the data

1) find this: $\hat{\mathbf{y}} = f(\mathbf{x})$

The parameters of the model $f(\mathbf{x})$ are *"learned"* from the data by minimising or maximising an objective function $E(\mathbf{w})$ over some training data, with respect to some loss function $\mathcal{L}(y_i, \hat{y}_i)$

2) using this: $$E(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}\big(y_i, f(\mathbf{x}_i, \mathbf{w})\big)$$

Notation is important	
x, X	Scalar value (variable, parameter or constant)
$\mathbf{x}$	Column or row vector
x_i	The i^{th} element of $\mathbf{x}$
$\mathbf{X}$	A matrix
$\mathbf{x_i}$	The i^{th} element of $\mathbf{X}$
$x^{(k)}$	The k^{th} example, input, or layer of $\mathbf{x}$ or x
$\mathbb{R}^{N\times P}$	Matrix of real numbers, of dimensions $(N\times P)$
$\mathcal{L}(\cdot)$	A loss function
$E(\cdot)$	An objective function

$\mathbf{y} \in \mathbb{R}^{N\times 1}$ are the response labels;

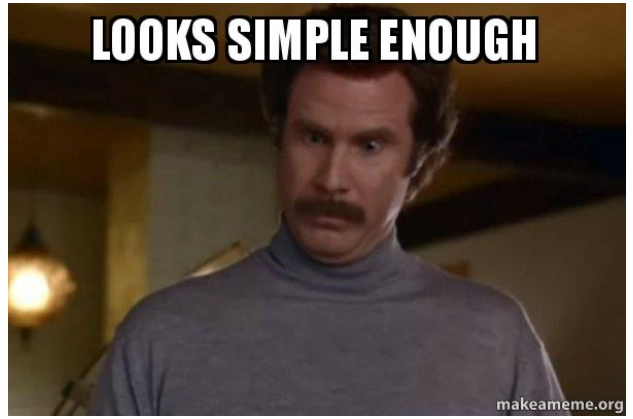
$\hat{\mathbf{y}} \in \mathbb{R}^{N\times 1}$ are the predictions / estimations;

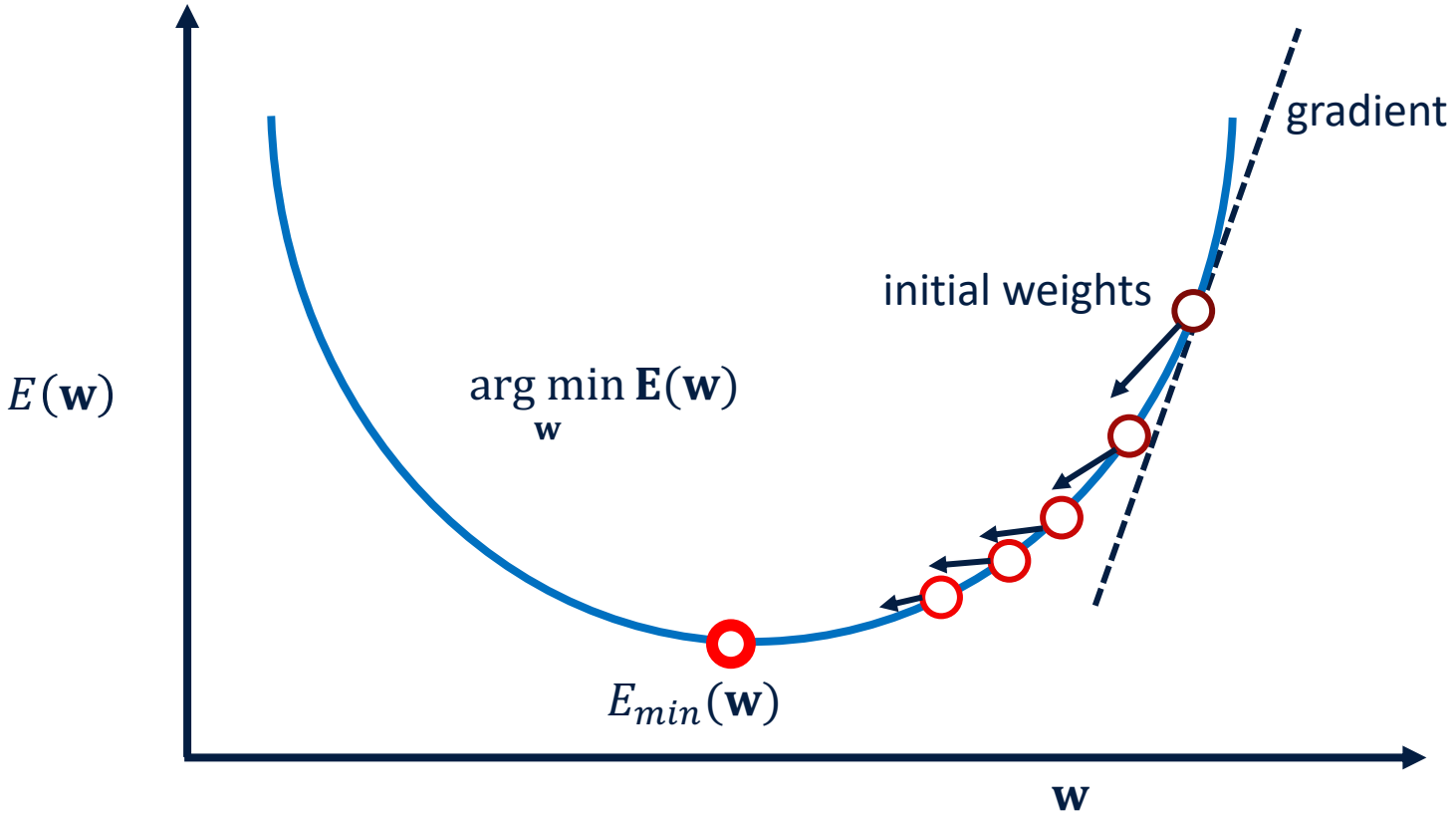
$\mathbf{w} \in \mathbb{R}^{1\times N}$ are the weights or parameters of a model

$\mathbf{X} \in \mathbb{R}^{N\times P}$ are the features / measurements as a (design) matrix

N: number of samples / observations. T: number of timesteps. P: number of features, input channels, etc.

Machine Learning 101





Jacobian

Gradient: $\mathbf{J} = \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \nabla E(\mathbf{w})$

Hessian

$\mathbf{H} = \frac{\partial^2 E(\mathbf{w})}{\partial \mathbf{w} \partial \mathbf{w}^\intercal} = \nabla^2 E(\mathbf{w})$

We can use Newton-Raphson

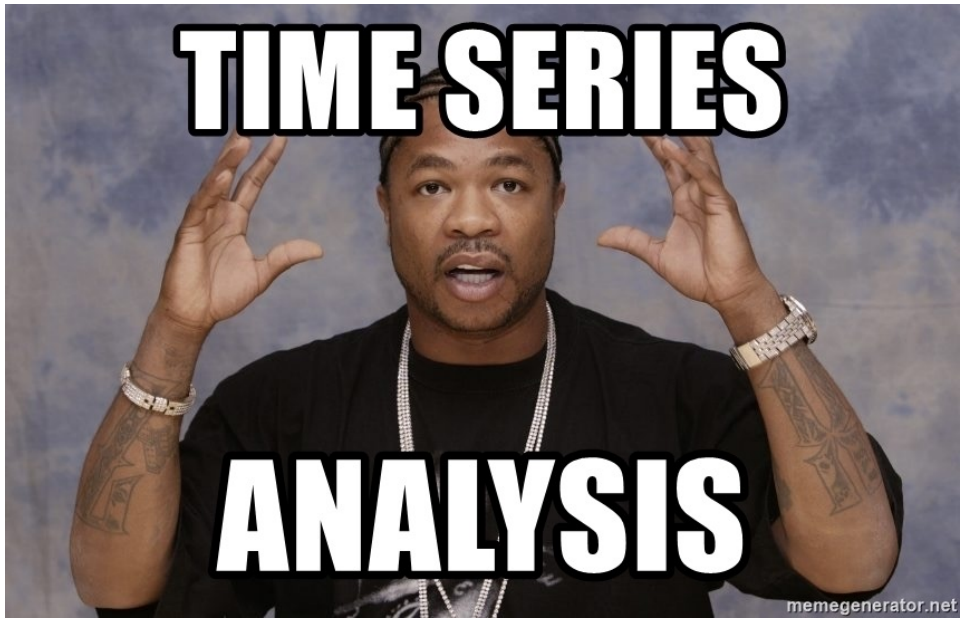
$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

Training a model, *TL;DR:*

To minimise (or maximise) an objective function, take the gradient (partial derivates) of the error function with respect to **w** and set its derivatives to zero.

Why time?

Clinical data is **not** static: clinical measurements are often a time-series, (e.g. heart rate) or repeated measurements taken during visits to the doctor, or a stay in hospital. Clinical events can occur at a point in time (e.g. a heart attack) or can even be causal. These events generally evolve over time (known as **Stochastic processes)**



time

How to implement ML 4 time-series?

Working with time

- Choosing your model
- Why logistic regression still works
- Representing time: extracting time-series features
- Markov chains
- Autoregressive models
- Hidden Markov Models HMM

Choosing your model

Models that don't consider time (i.i.d.)

- Linear / Logistic Regression
- Support Vector Machines (SVM)
- Classification and Regression Trees (CART) (e.g. Random Forest)
- eXtreme Gradient Boosting (XGBoost)
- Deep Neural Networks
- Etc.

Need to incorporate time through the features instead!

Image Credit

Choosing your model

Models that don't consider time (i.i.d.)

- Linear / Logistic Regression
- Support Vector Machines (SVM)
- Classification and Regression Trees (CART) (e.g. Random Forest)
- eXtreme Gradient Boosting (XGBoost)
- Deep Neural Networks
- Etc.

Need to incorporate time through the features instead!

Models that consider time (non i.i.d)

- Markov Chains
- Autoregressive Processes
- Hidden Markov Models (HMM)
- Recurrent Neural Networks (RNN) /
  Long-Short Term Memory (LSTM)
- Transformers,
- Etc.

Image Credit

Choosing your model

Models that don't consider time (i.i.d)

- Linear / Logistic Regression
- Support Vector Machines (SVM)
- Classification and Regression Trees (CART) (e.g. Random Forest)
- eXtreme Gradient Boosting (XGBoost)
- Deep Neural Networks
- Etc.

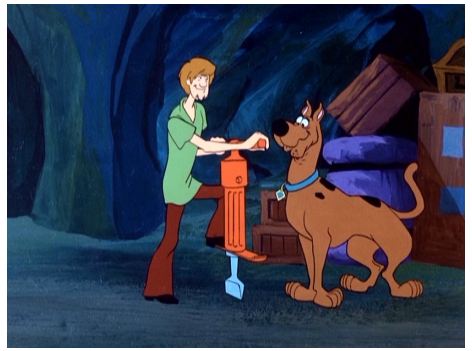
Need to incorporate time through the features instead!

Models that consider time (non i.i.d)

- Markov Chains
- Autoregressive Processes
- Hidden Markov Models (HMM)
- Recurrent Neural Networks (RNN) / Long-Short Term Memory (LSTM)
- Transformers,
- Etc.

Image Credit

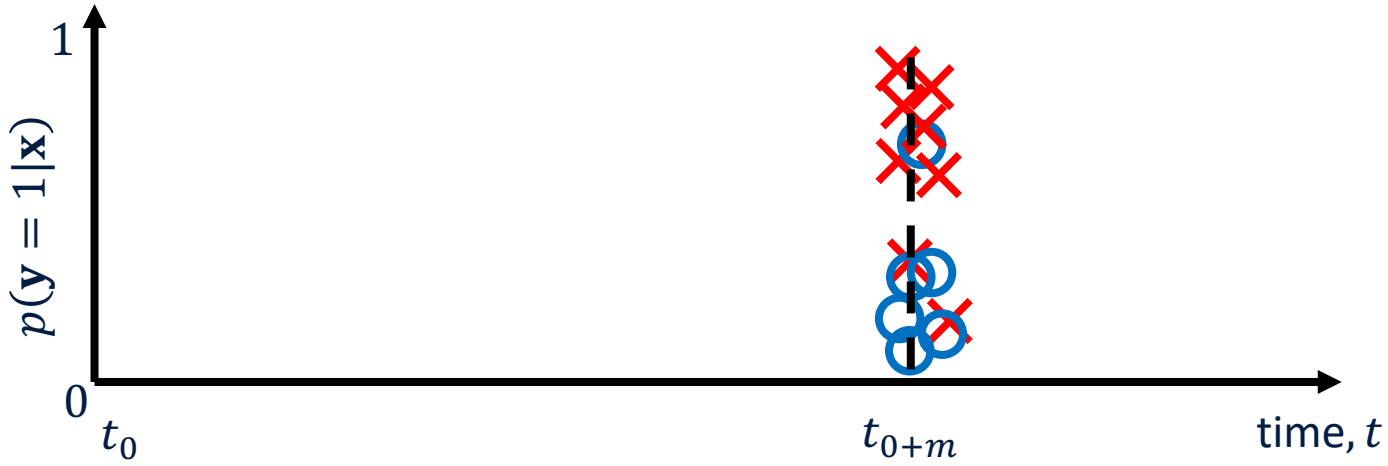
Is a more complex model *necessary*?



The importance of data curation, cleaning, and pre-processing are often overlooked

Why Logistic Regression still works*

**sometimes*: at fixed prediction horizons



After 60 minutes, is the patient alive or dead, m =60 minutes?

- patient stable ($y = 0$)
- mortality ($y = 1$)

linear regression

$$\hat{\mathbf{y}} = \mathbf{w}^\intercal \mathbf{x}$$

logistic regression

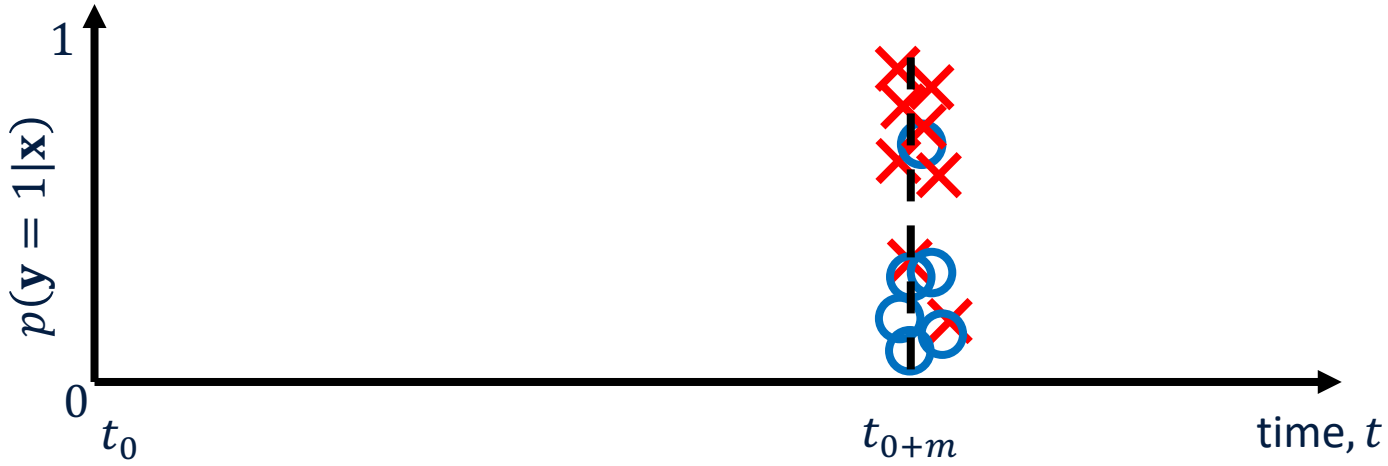
$$p(\mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{-(\mathbf{w}^\intercal \mathbf{x})}}$$

cross-entropy (negative log-likelihood)

$$E(\mathbf{w}) = -\ln p(\mathbf{y} = 1|\mathbf{w}) = -\sum_{i=1}^{N} \{y_i \ln(\hat{y}_i) + (1 - y_i) \ln(1 - \hat{y}_i)\}$$

Why Logistic Regression still works*

**sometimes*: at fixed prediction horizons



After 60 minutes, is the patient alive or dead, m =60 minutes?

- patient stable ($y = 0$)
- mortality ($y = 1$)

linear regression

$$\hat{\mathbf{y}} = \mathbf{w}^\intercal \mathbf{x}$$

logistic regression

$$p(\mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{-(\mathbf{w}^\intercal \mathbf{x})}}$$

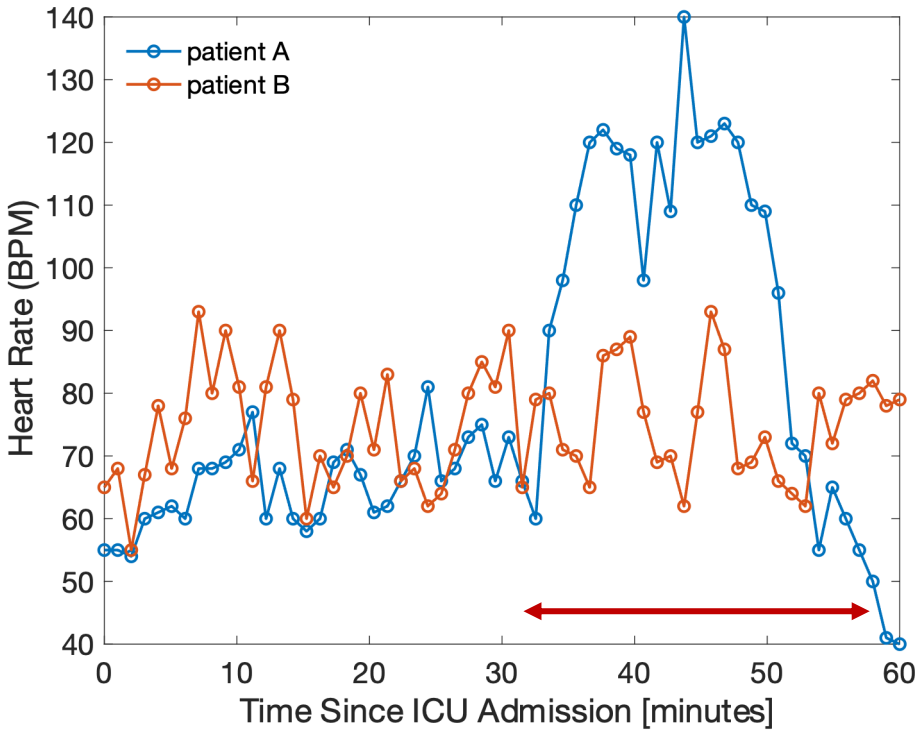
cross-entropy (negative log-likelihood)

$$E(\mathbf{w}) = -\ln p(\mathbf{y} = 1|\mathbf{w}) = -\sum_{i=1}^{N} \{y_i \ln(\hat{y}_i) + (1 - y_i) \ln(1 - \hat{y}_i)\}$$

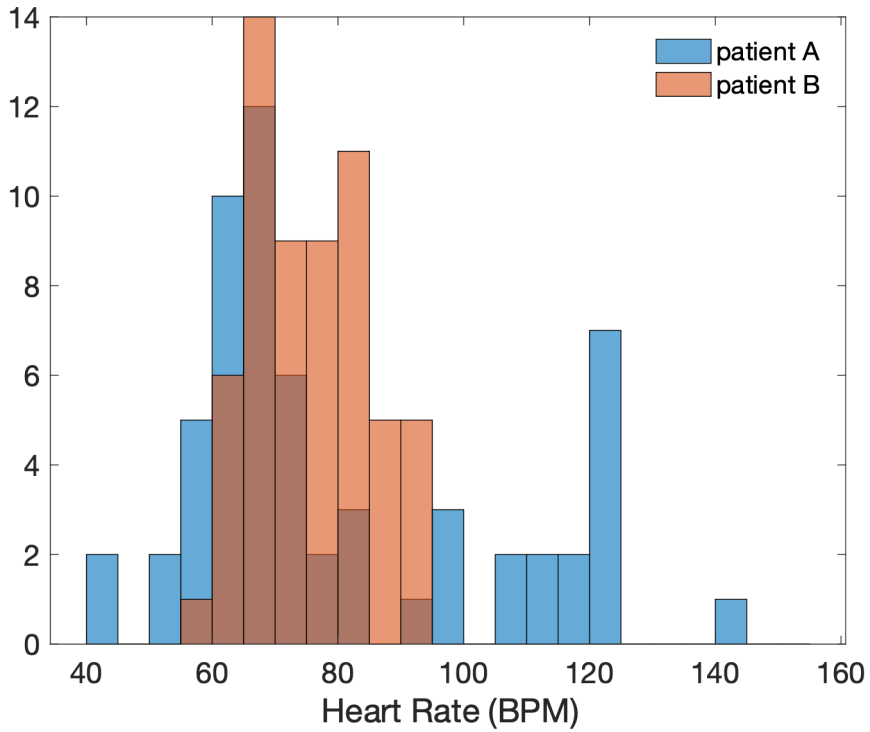
Options:

1. Just consider $\mathbf{x}$ independent and identically distributed (i.i.d.) …
2. Take $\mathbf{x}$ at baseline
3. Summerise $\mathbf{x}$ up until the prediction horizon t_{0+m}

Representing Time



clinical event leading to mortality*



Example feature(s)

- $\bar{x}$ **:** take the average heart rate (HR)

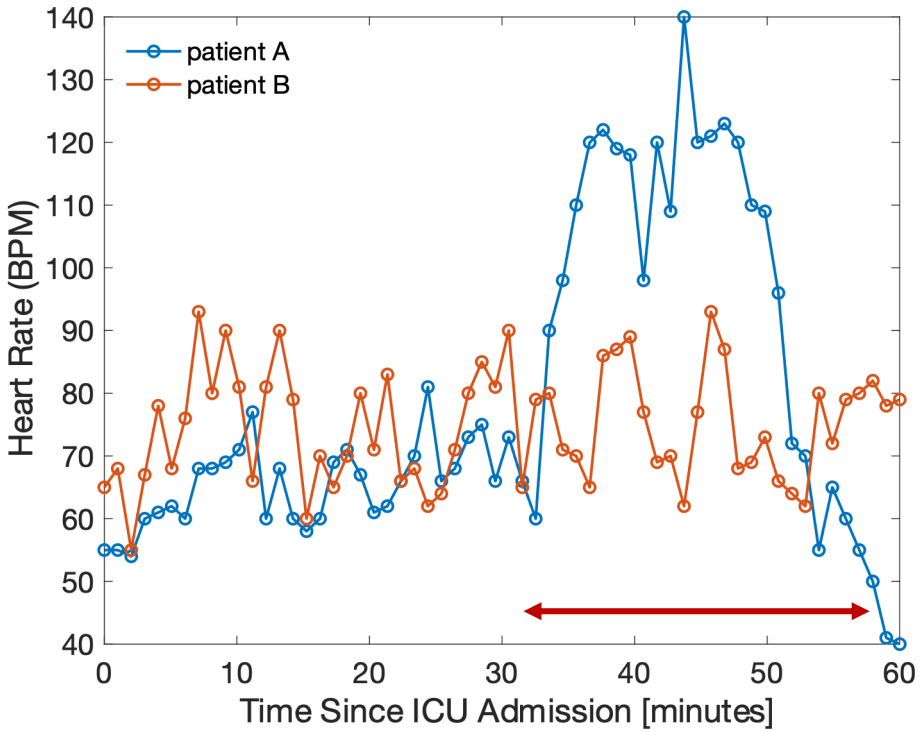
Patient A: 78.4 ± 25 BPM

Patient B: 74.7 ± 9 BPM

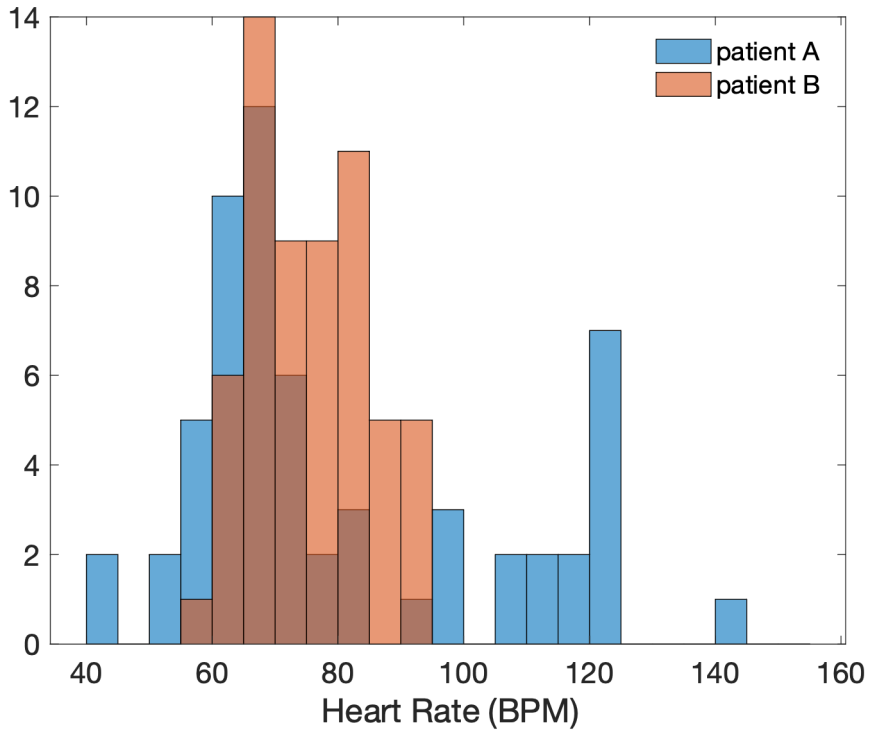
$p = 0.21$

*synthetic data

Representing Time



clinical event leading to mortality*



Example feature(s)

- $\bar{x}$ **:** take the average heart rate (HR)
- $\bar{x} + \sigma$: average HR plus the variability in the HR

Patient A: 78.4 ± 25 BPM

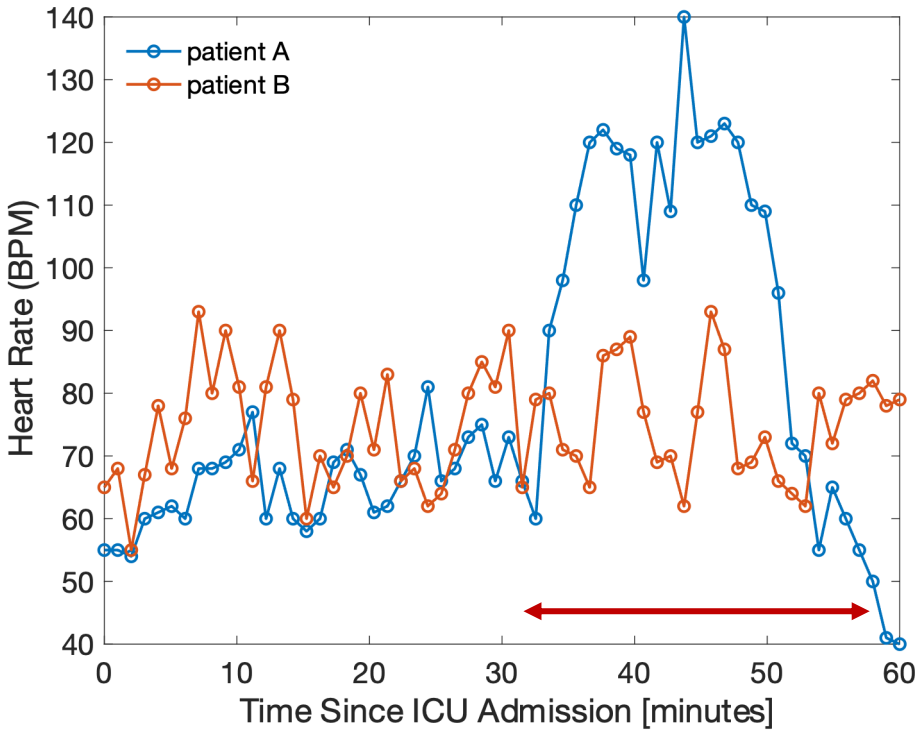
Patient B: 74.7 ± 9 BPM

$p = 0.21$

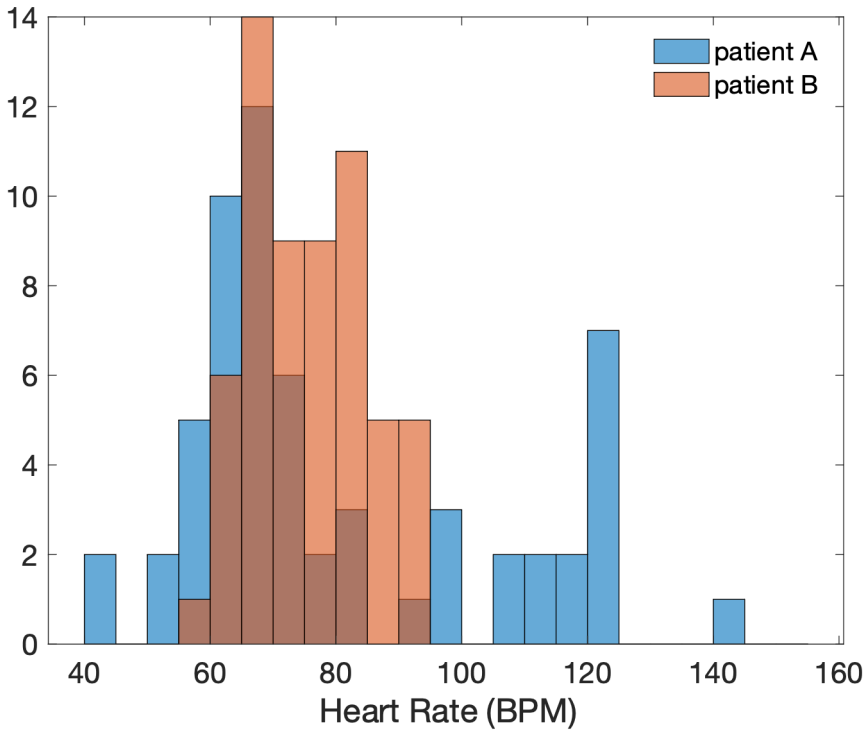
*synthetic data

Representing Time

$$\tilde{x} = \frac{x_T - x_1}{\sigma}$$



clinical event leading to mortality*



Example feature(s)

- $\bar{x}$ **:** take the average heart rate (HR)
- $\bar{x} + \sigma$: average HR plus the variability in the HR
- $\bar{x} + \tilde{x}$: average HR plus the slope of the HR

Patient A: 78.4 ± 25 BPM

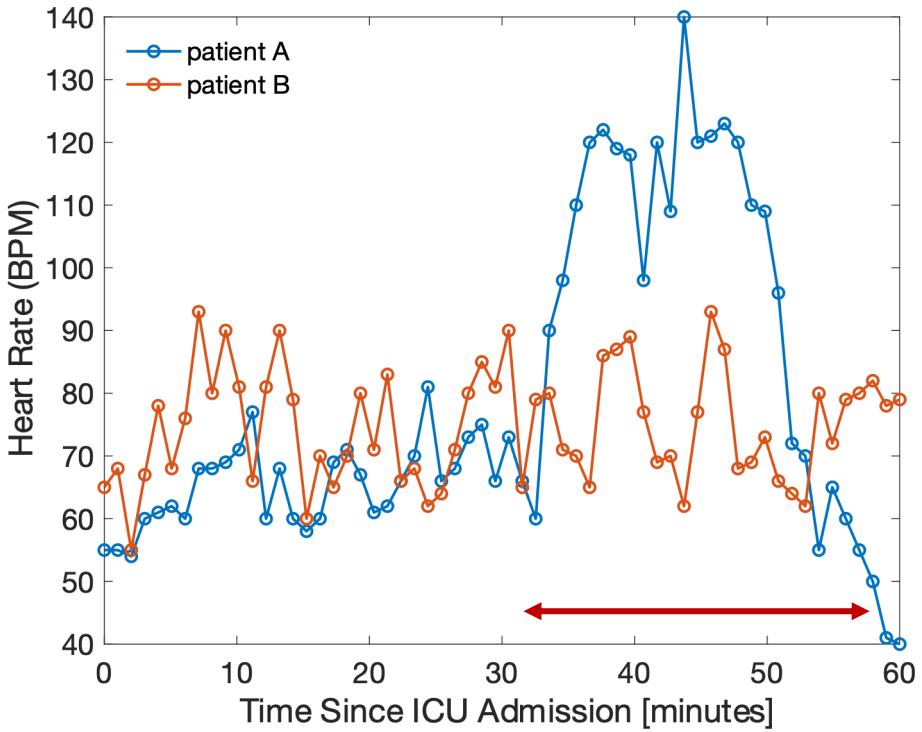
Patient B: 74.7 ± 9 BPM

$p = 0.21$

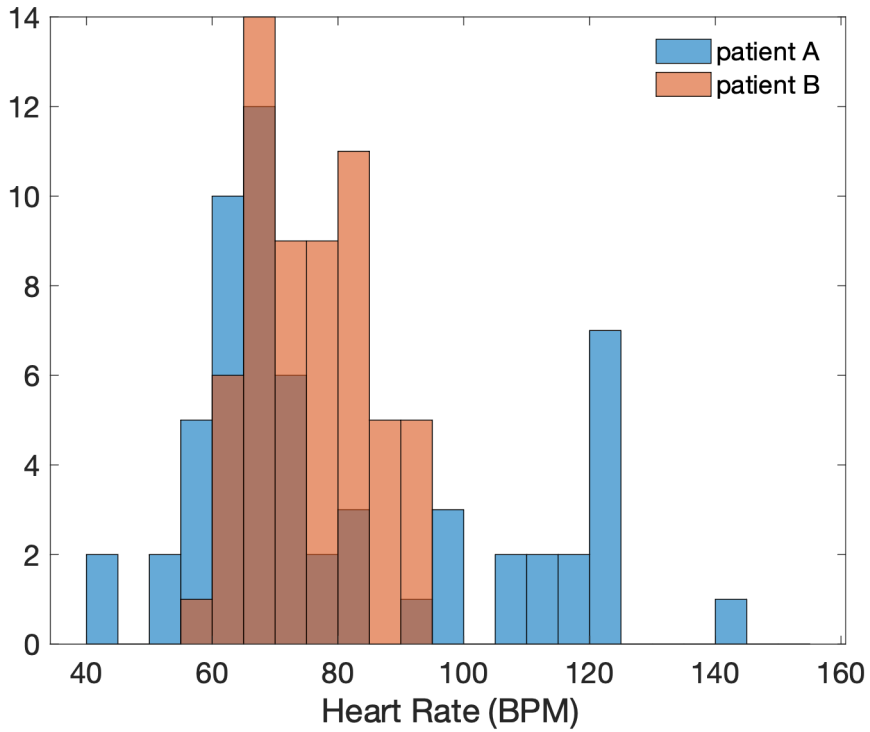
*synthetic data

Representing Time

$$\tilde{x} = \frac{x_T - x_1}{\sigma}$$



clinical event leading to mortality*



Example feature(s)

- $\bar{x}$ **:** take the average heart rate (HR)
- $\bar{x} + \sigma$: average HR plus the variability in the HR
- $\bar{x} + \tilde{x}$: average HR plus the slope of the HR
- $\mathbf{x}$: the raw HR (let model work out time-feature)
- $\mathbf{x}$: the raw HR + HR lag-feature (e.g. x'_{t+1} given x_{t-1})

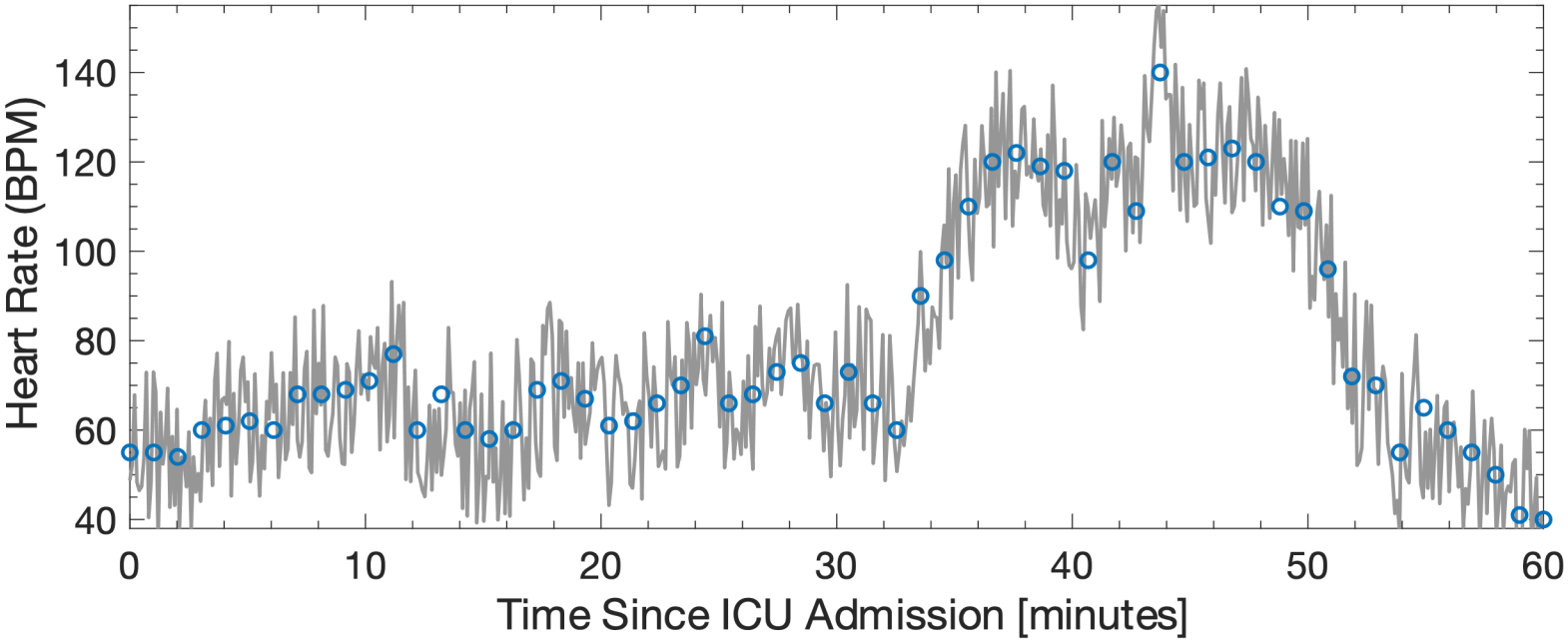
Patient A: 78.4 ± 25 BPM

Patient B: 74.7 ± 9 BPM

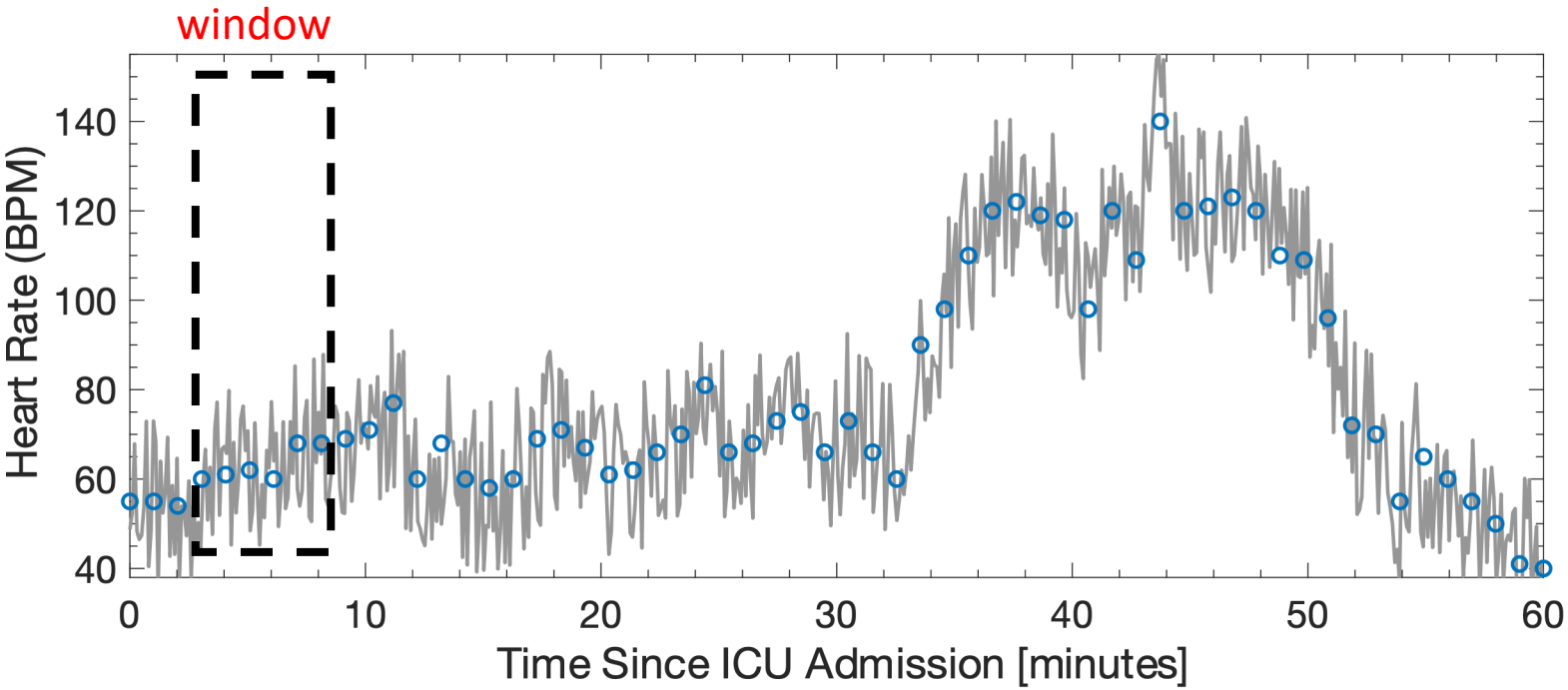
$p = 0.21$

*synthetic data

Discretising Time: Windowing

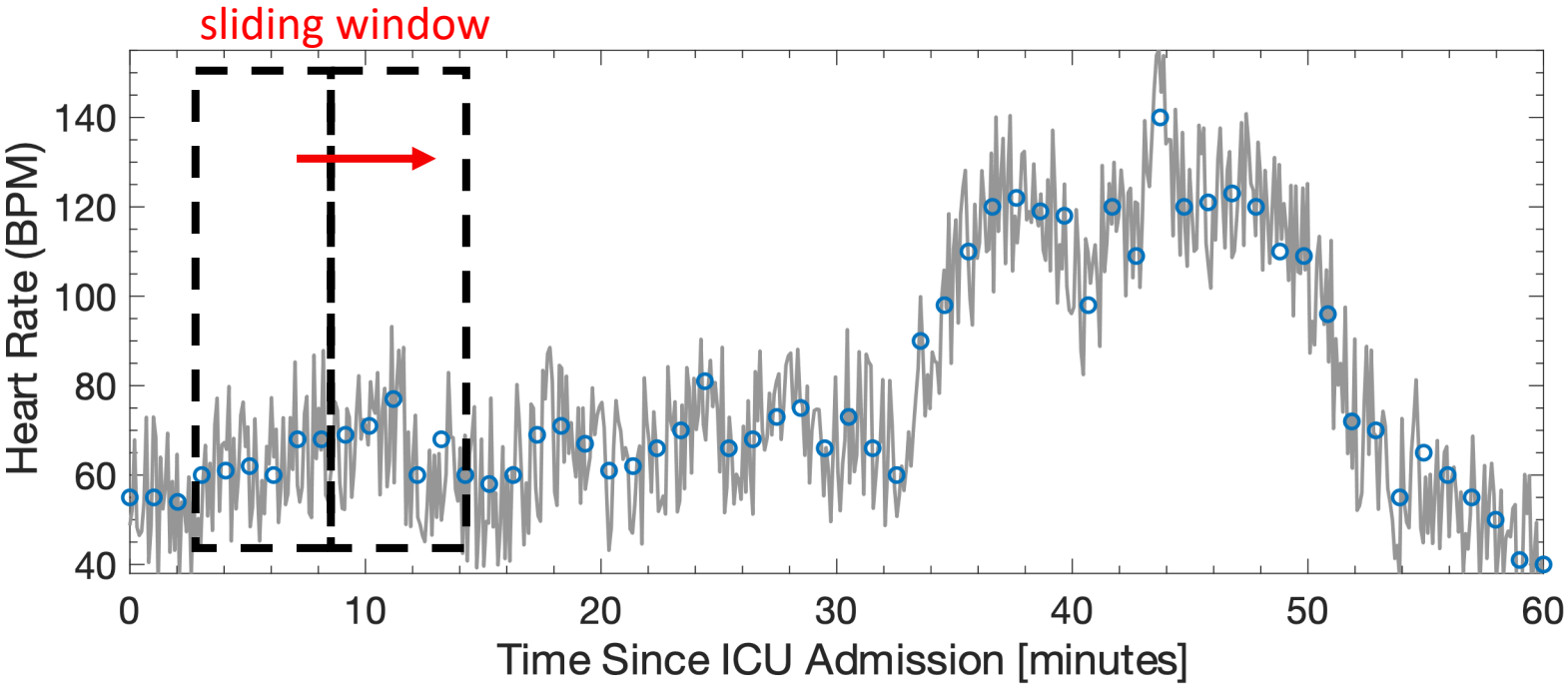


Discretising Time: Windowing



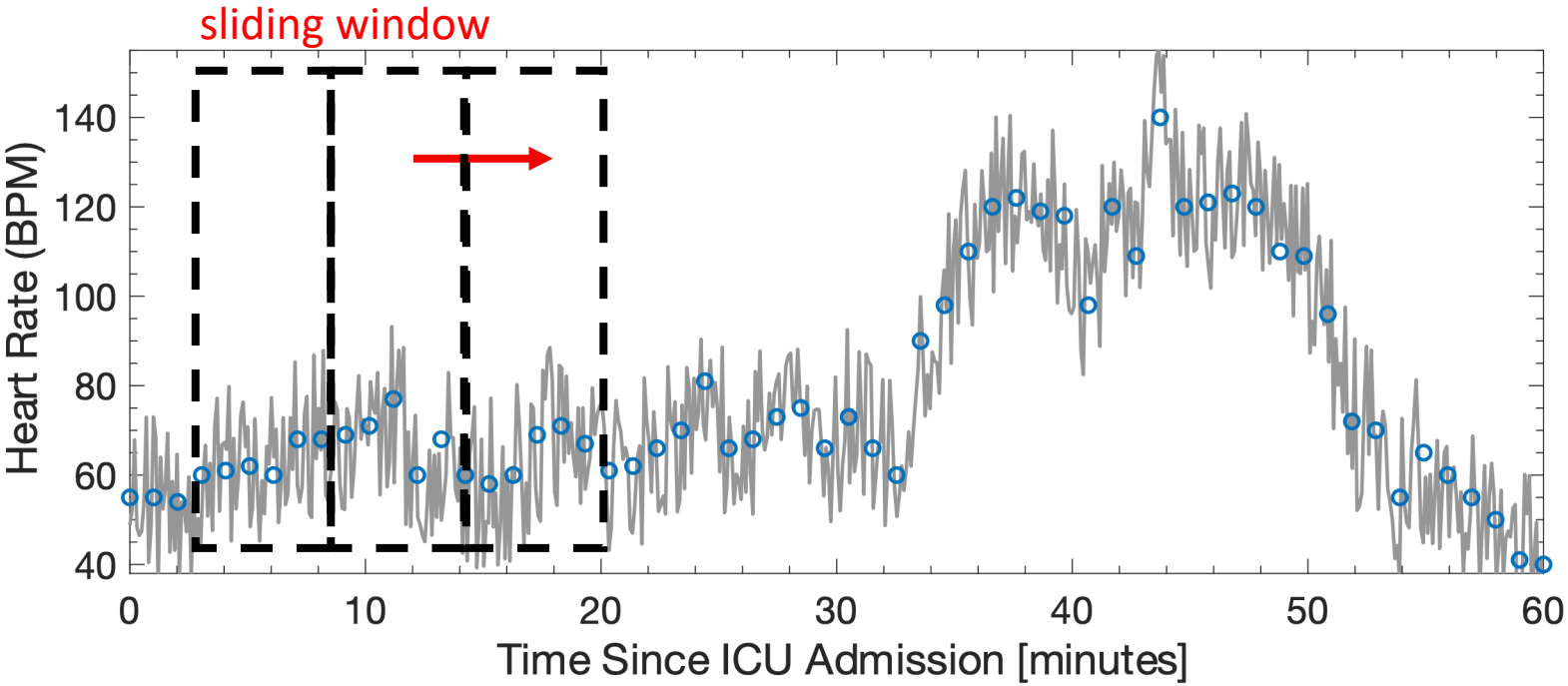
$$\mathbf{z}_1 = \begin{bmatrix} \bar{x}^{(1)} \\ \sigma^{(1)} \\ x_{min}^{(1)} \\ x_{max}^{(1)} \end{bmatrix}$$

Discretising Time: Windowing



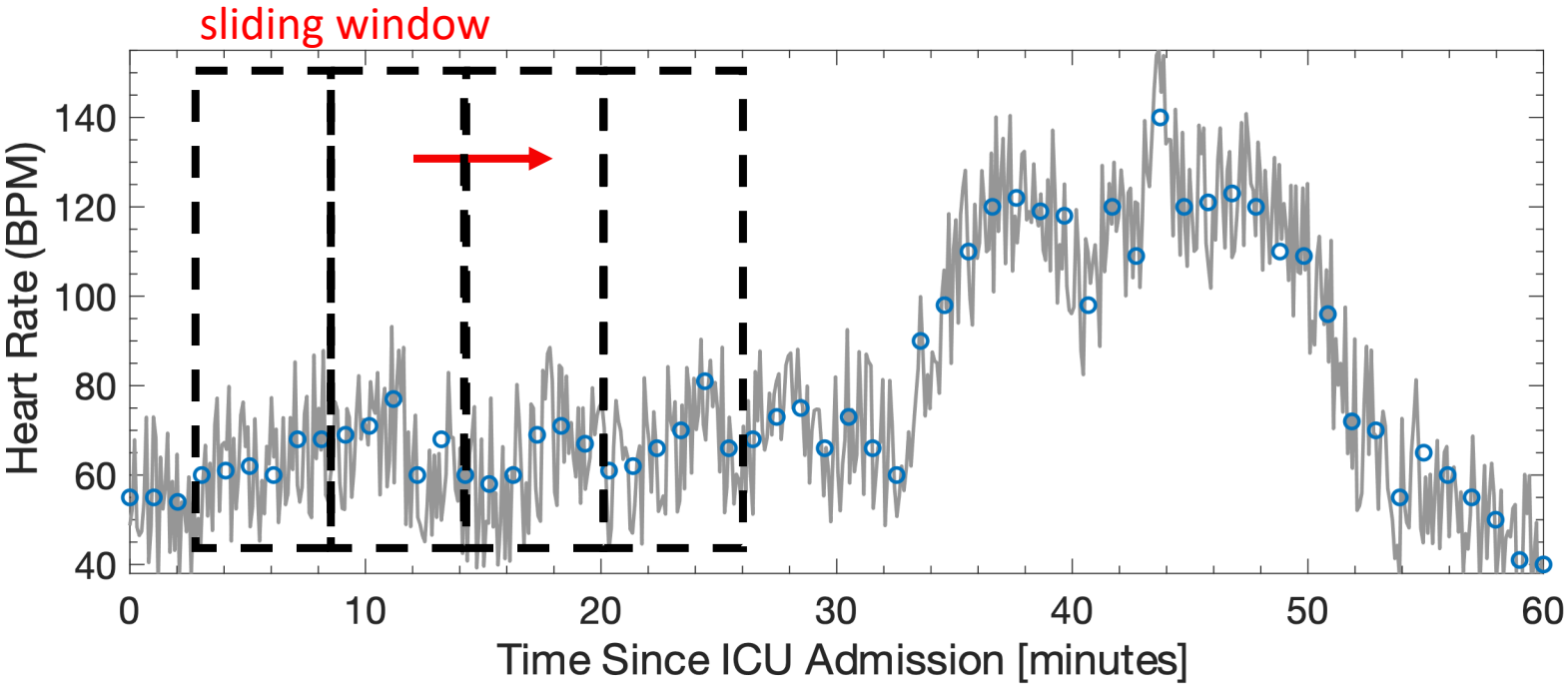
$$\mathbf{z}_2 = \begin{bmatrix} \bar{x}^{(2)} \\ \sigma^{(2)} \\ x_{min}^{(2)} \\ x_{max}^{(2)} \end{bmatrix}$$

Discretising Time: Windowing



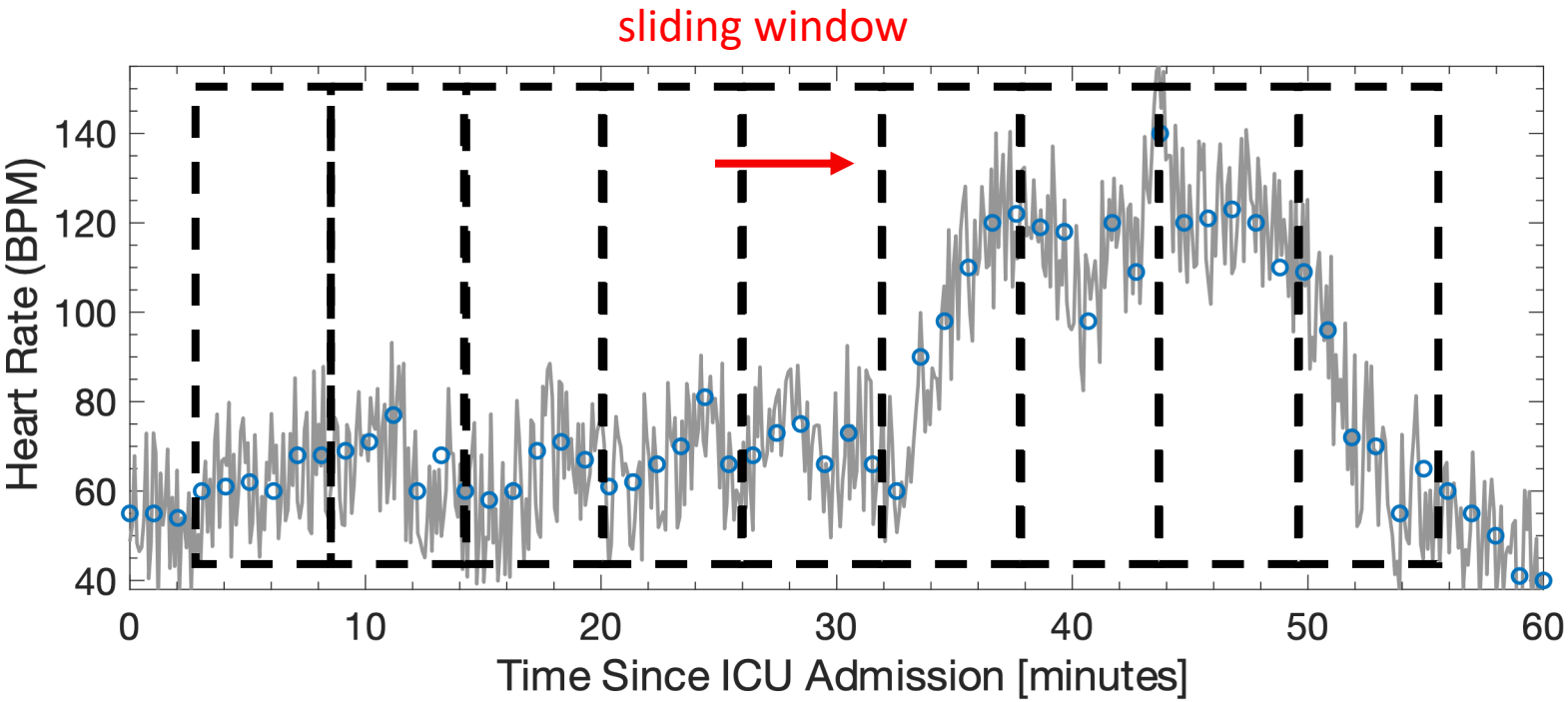
$$\mathbf{z}_3 = \begin{bmatrix} \bar{x}^{(3)} \\ \sigma^{(3)} \\ x_{min}^{(3)} \\ x_{max}^{(3)} \end{bmatrix}$$

Discretising Time: Windowing



$$\mathbf{z}_3 = \begin{bmatrix} \bar{x}^{(3)} \\ \sigma^{(3)} \\ x_{min}^{(3)} \\ x_{max}^{(3)} \end{bmatrix}$$

Discretising Time: Windowing



$$\mathbf{z}_m = \begin{bmatrix} \bar{x}^{(m)} \\ \sigma^{(m)} \\ x_{min}^{(m)} \\ x_{max}^{(m)} \end{bmatrix}$$

$$\mathbf{z}_m = \left[\bar{x}^{(m)}, \sigma^{(m)}, x_{min}^{(m)}, x_{max}^{(m)}\right]^{\mathrm{T}}$$

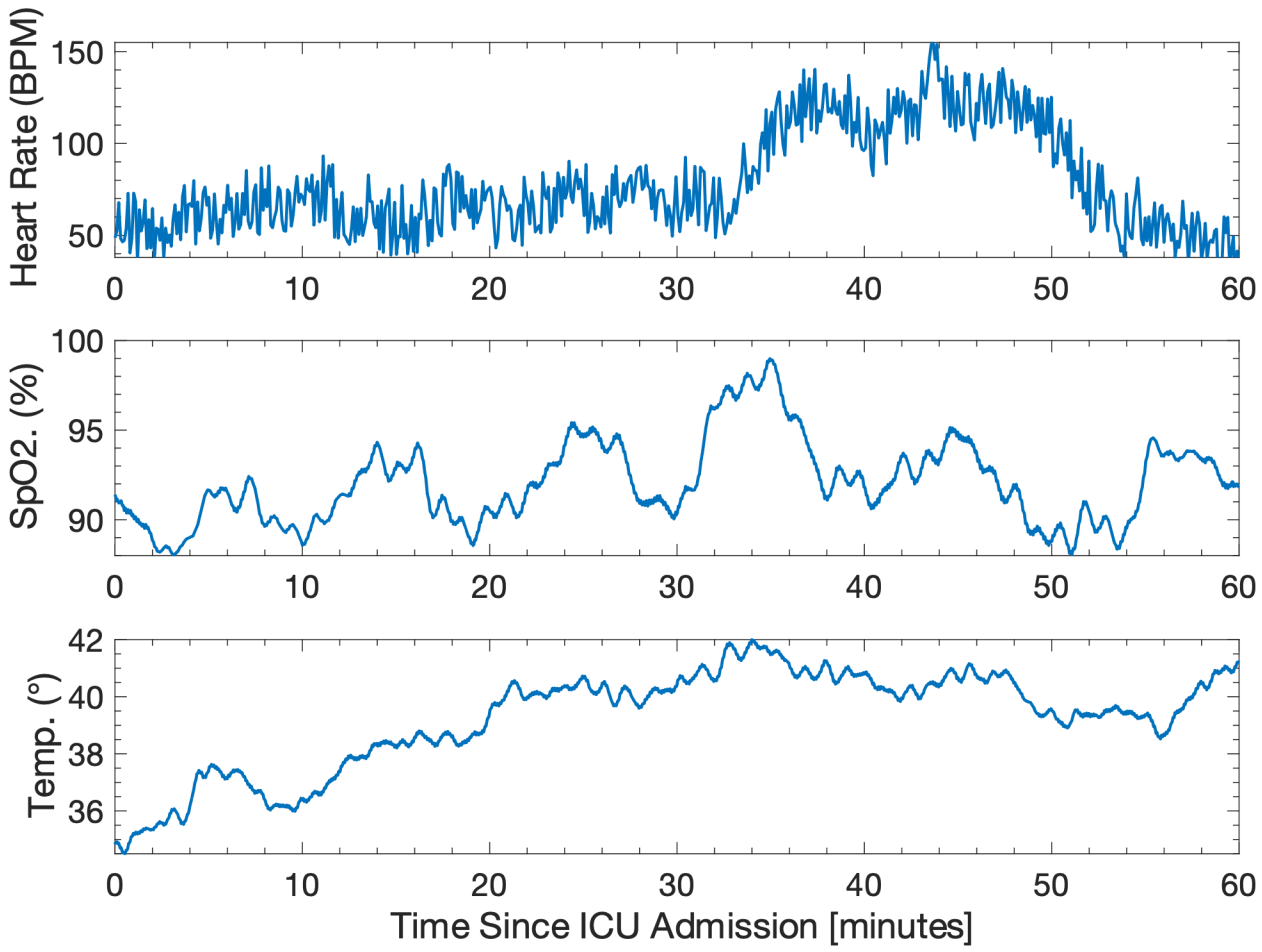
$$\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \cdots, \mathbf{z}_T]$$

$$\mathbf{Z} \in \mathbb{R}^{P \times T}$$

P: number of features, input channels, etc.

T: number of timesteps.

Handling Multivariate Time-Series Data



$$\mathbf{x}_1 = \{x_1, x_2, \cdots x_m\}$$

$$\mathbf{x}_2 = \{x_1, x_2, \cdots x_n\}$$

$$\mathbf{x}_3 = \{x_1, x_2, \cdots x_n\}$$

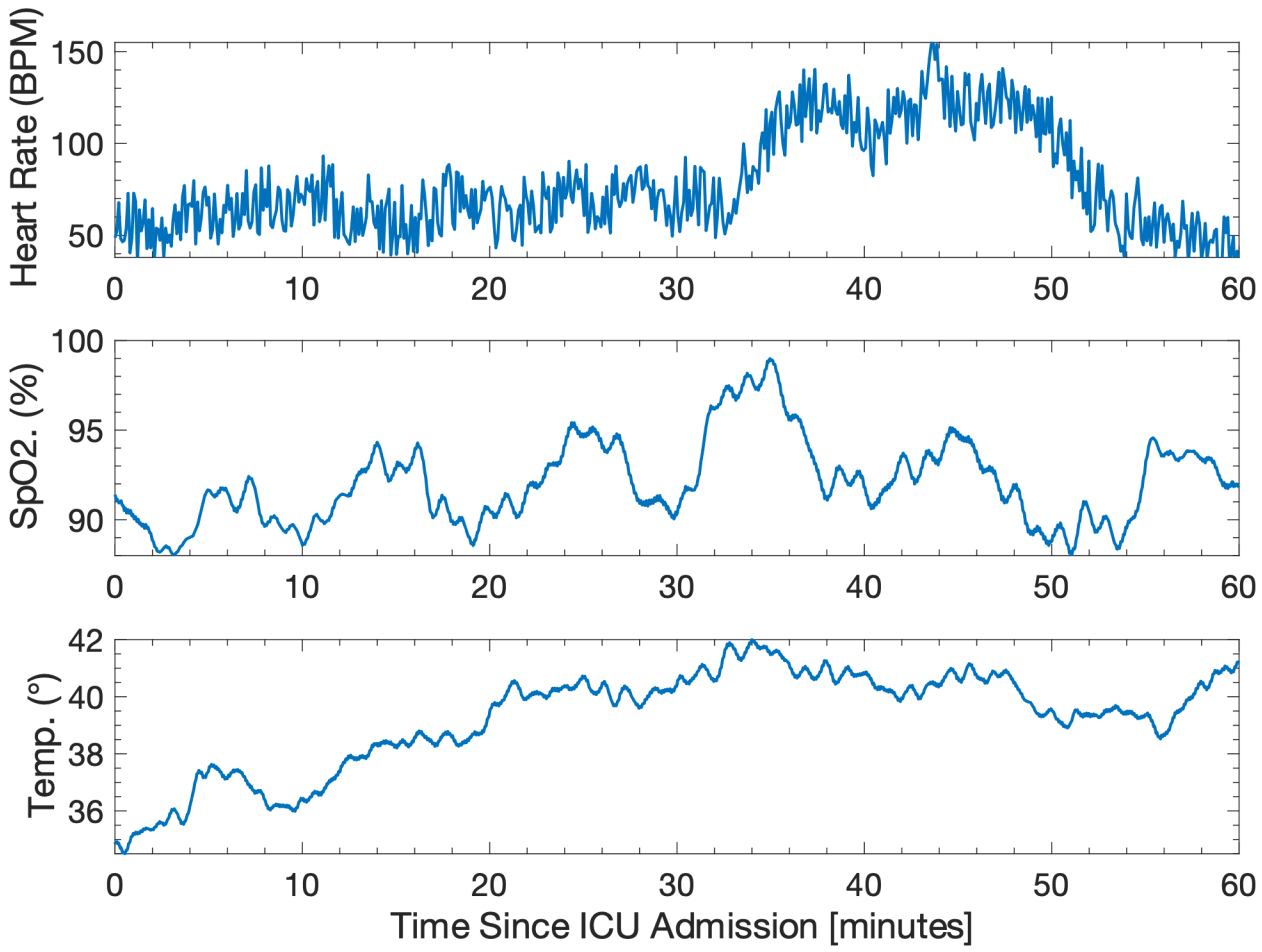
$$m > n$$

Handling uneven / irregularly sampled data:

Resampling

- Downsample (or window)
- Don't up-sample (if you can avoid it); you're interpolating information that you don't know, which is always dangerous

Handling Multivariate Time-Series Data



$$\mathbf{x}_1 = \{x_1, x_2, \cdots x_m\}$$

$$\mathbf{x}_2 = \{x_1, x_2, \cdots x_n\}$$

$$\mathbf{x}_3 = \{x_1, x_2, \cdots x_n\}$$

$$m > n$$

Handling uneven / irregularly sampled data:

Resampling

- Downsample (or window)
- Don't up-sample (if you can avoid it); you're interpolating information that you don't know, which is always dangerous

$$\mathbf{Z} \in \mathbb{R}^{P \times T}$$

for each time-series feature (e.g. HR, SpO2, Temp.)

$$\mathbf{Z} \in \mathbb{R}^{(P \times 3) \times T}$$

Markov Chains

For a timeseries: $\mathbf{x} = \{x_t, x_{t+1}, x_{t+2}, \cdots, x_T\}$, we need a model $p(\mathbf{x})$:

$$p(\mathbf{x}) = p(x_1) \prod_{t=2}^{T} p(x_t|x_{1:t-1}) \qquad p(x_t|x_{1:t-1}) = p(x_1) \text{ for } t = 1$$

initial transition

$$p(x_{1:T}) = p(x_1)p(x_2|x_1)\, p(x_3|x_2) \cdots p(x_T|x_{T-1})$$

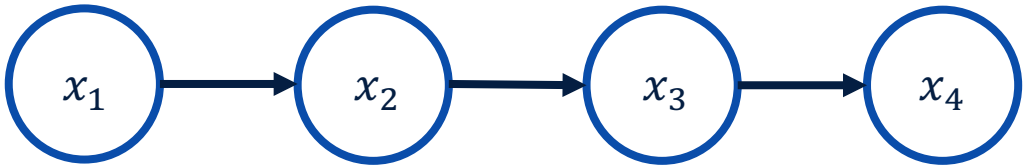
Independence assumptions
We only assume that the influence of the immediate past is more relevant than the remote past and in Markov models only a limited number of previous observations are required to predict the future.

Fit by Maximum Likelihood

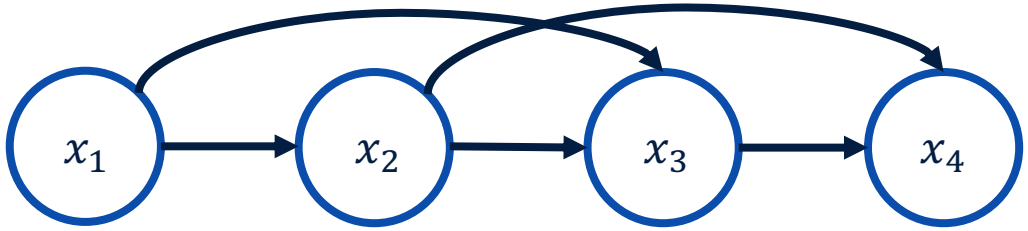
Assuming only the recent past is relevant:

$$p(x_t|x_{1:t-1}, \cdots x_{t-1}) = p(x_t|x_{1:t-L}, \cdots, x_{t-1})$$

where $L \geq 1$ is the *order* of the Markov chain



(a) First order Markov Chain



(b) Second order Markov Chain

Auto-Regressive Models

(are Continuous-state Markov Models)

The timeseries value x_t can modelled by a weighted sum of previous values:

$$x_t \approx \sum_{l=1}^{L} a_l x_{t-l} + c$$

a_l we can call the 'AR coefficients'

We can view this then as a form of regression, and find the AR coefficients by minimising the squared loss:

$$\sum_t \left(x_t - \sum_{l=1}^{L} a_l x_{t-l} \right)^2$$

This is a quadratic function, we can easily solve this!

Maximum likelihood estimation (MLE)

Autoregression (AR) models can also be generalised to the order l by

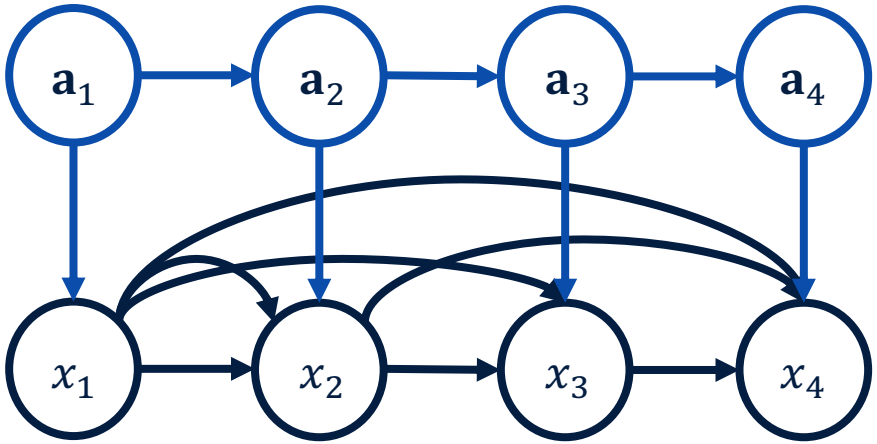
$$AR(l) = x_t \approx a_1\, x_{t-1} + a_2\, x_{t-2} + \cdots + a_p\, x_{t-l+c}$$

Auto-Regressive Models: Assumptions

A scalar time-invariant auto-regressive model is defined by

$$x_t = \sum_{l=1}^{L} a_l x_{t-l} + \eta_t \qquad \eta_t \sim \mathrm{N}(\eta_t | \mu, \sigma^2)$$

Where $\mathbf{a} = (a_1, \cdots, a_L)^\intercal$ we can call the 'AR coefficients', and σ^2 is called the *innovation* noise. The model predicts the future based on a linear combination of previous L observations.



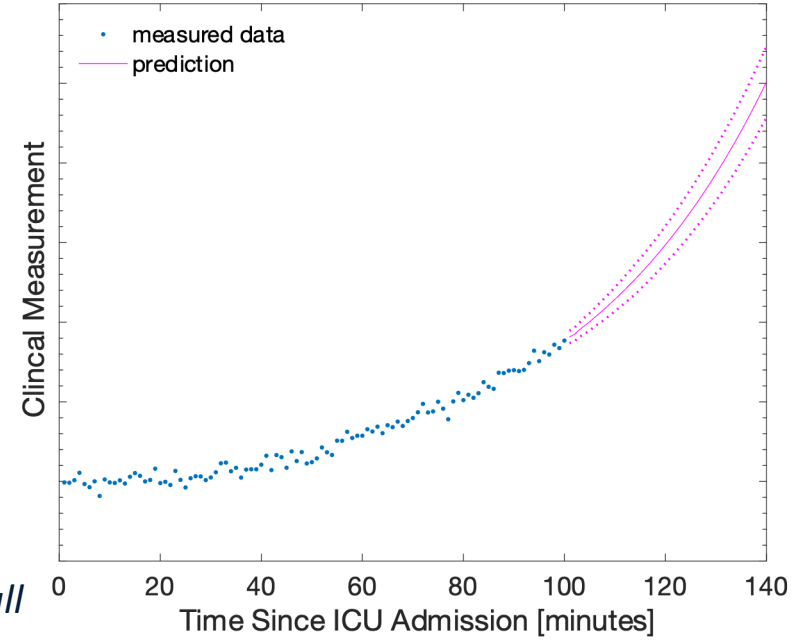
A time-varying AR model

This is an L^{th} order Markov model!

$$p(\mathbf{x}) = p(x_1) \prod_{t=2}^{T} p(x_t | x_{t-1}, \cdots, x_{t-L}) \qquad x_i = \emptyset \text{ for } i \leq 0$$

with

$$p(x_t | x_{t-1}, \cdots x_{t-L}) = \mathrm{N}\left(x_t \middle| \sum_{l=1}^{L} a_l x_{t-l}, \sigma^2 \right)$$



Practically speaking, we can use the AR coefficients as features, or use the AR model overall

Hidden Markov Models (HMM)

Markov property that only the present influences the future

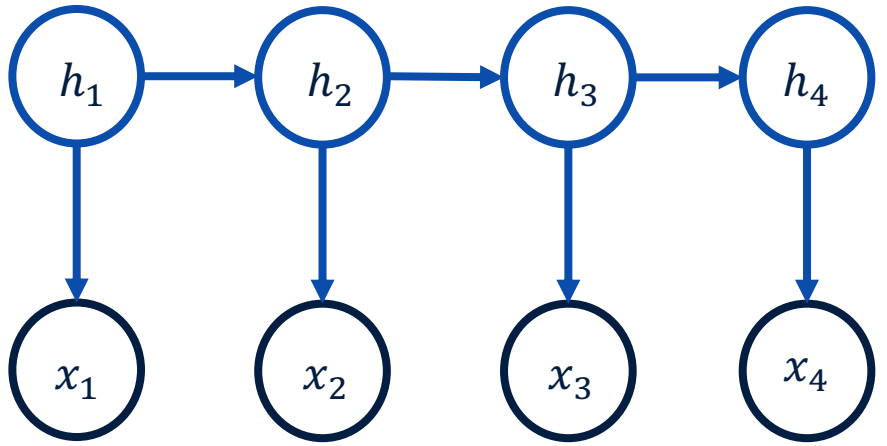
The Hidden Markov Model (HMM) defines a Markov chain on hidden (or `latent') variables $h_t = \{h_1, h_2, \cdots, h_H\}$

The observed (or `visible') variables are dependent on the hidden variables through an emission $p(x_t|h_t)$. This defines a joint distribution:

$$p(\mathbf{h}, \mathbf{x}) = p(x_1|h_1)p(h_1)\prod_{t=2}^{T} p(x_t|h_t)p(h_t|h_{t-1})$$

emission prob

transition prob

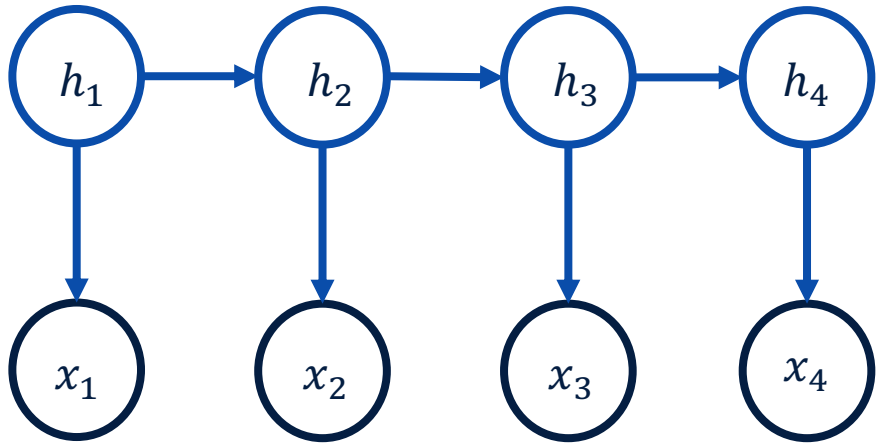


Hidden Markov Models (HMM)

Markov property that only the present influences the future

The Hidden Markov Model (HMM) defines a Markov chain on hidden (or `latent') variables $h_t = \{h_1, h_2, \cdots, h_H\}$

The observed (or `visible') variables are dependent on the hidden variables through an emission $p(x_t|h_t)$. This defines a joint distribution:



$$p(\mathbf{h}, \mathbf{x}) = p(x_1|h_1)p(h_1)\prod_{t=2}^{T} p(x_t|h_t)p(h_t|h_{t-1})$$

emission prob

transition prob

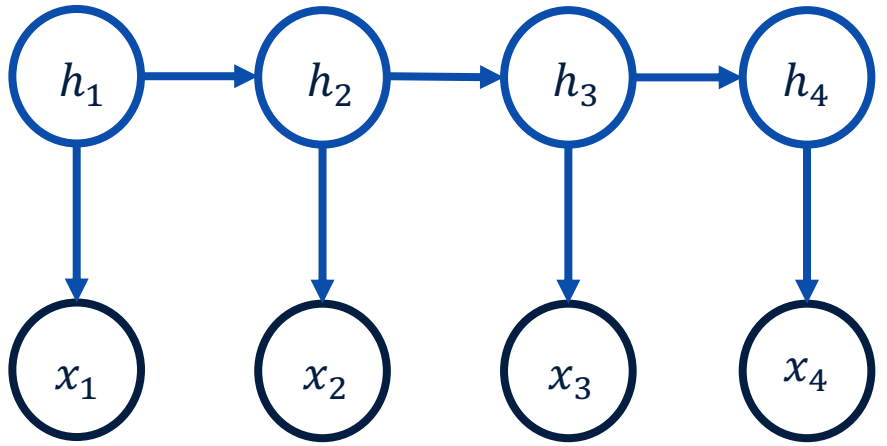
1. The transition distribution $p(h_{t+1}|h_t)$ is defined by a $H \times H$ **transition** matrix: $\mathbf{A}_{i',i} = p(h_{t+1} = i'|h_t = i)$
2. The emission distribution, $p(x_t|h_t)$ has discrete states $x_t \in \{1, \cdots, V\}$, we can define a $V \times H$ **emission** matrix: $\mathbf{B}_{i,j} = p(x_t = i|h_t = k)$

   For continuous outputs, h_t selects one of H possible output distributions $p(x_t|h_t), h_t \in \{1, \cdots, H\}$.

Hidden Markov Models (HMM)

Markov property that only the present influences the future

The Hidden Markov Model (HMM) defines a Markov chain on hidden (or `latent') variables $h_t = \{h_1, h_2, \cdots, h_H\}$



The observed (or `visible') variables are dependent on the hidden variables through an emission $p(x_t|h_t)$. This defines a joint distribution:

$$p(\mathbf{h}, \mathbf{x}) = p(x_1|h_1)p(h_1)\prod_{t=2}^{T} p(x_t|h_t)p(h_t|h_{t-1})$$

emission prob

transition prob

1. The transition distribution $p(h_{t+1}|h_t)$ is defined by a $H \times H$ **transition** matrix: $\mathbf{A}_{i',i} = p(h_{t+1} = i'|h_t = i)$
2. The emission distribution, $p(x_t|h_t)$ has discrete states $x_t \in \{1, \cdots, V\}$, we can define a $V \times H$ **emission** matrix: $\mathbf{B}_{i,j} = p(x_t = i|h_t = k)$

   For continuous outputs, h_t selects one of H possible output distributions $p(x_t|h_t), h_t \in \{1, \cdots, H\}$.
3. Most likely Hidden path, $\arg\max_{\mathbf{h}} p(\mathbf{h}, \mathbf{x})$ is found via the ***Viterbi*** algorithm (*don't worry too much about this*)

How to implement ML 4 time-series?

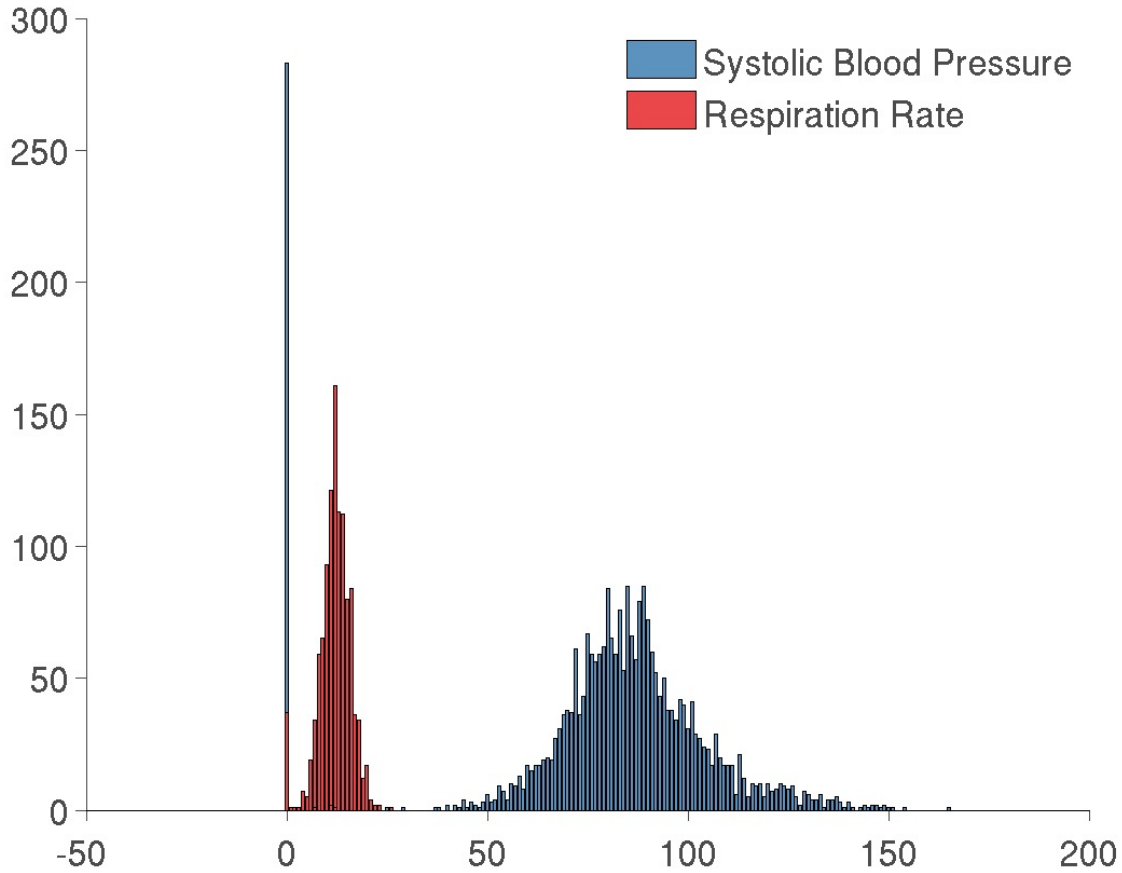
Pre-processing steps:

- Outlier removal
- Normalisation
- Dealing with missing values

Credit: me

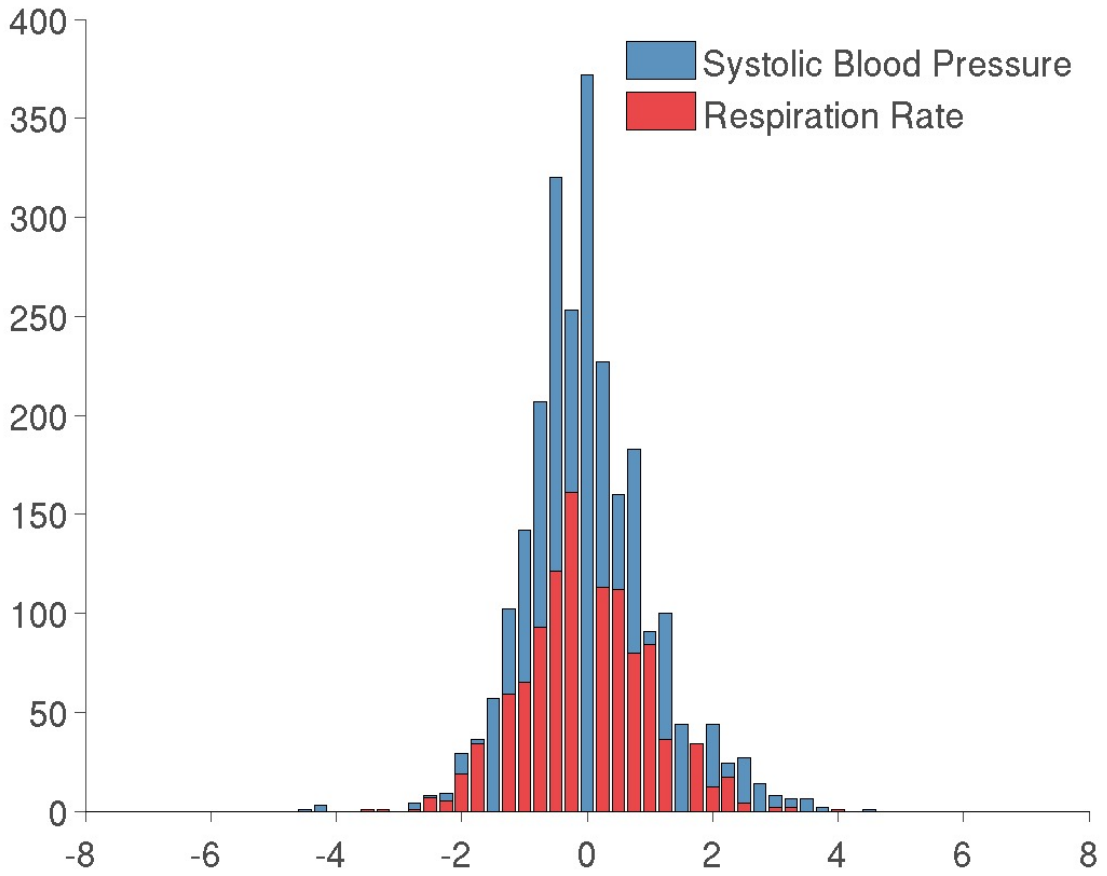
Pre-processing

- A trivial way in which clinical data are corrupted might be through artefactually-low or –high data
- These are typically easy to identify, and can therefore be removed straightforwardly
- In the example shown here, it is unlikely that (living) patients had a systolic blood pressure of 0 mmHg or a respiratory rate of 0 breaths/min…
- Remove outliers greater than a certain number of standard deviations



Normalisation

- Normalisation is effectively a transformation of the horizontal axis
- It is usually a linear transformation, such that the shape of the distribution is not distorted
- Common normalisations include:



0-1 normalisation:

$$\mathbf{x}' = \frac{\mathbf{x} - \min(\mathbf{x})}{\max(\mathbf{x}) - \max(\mathbf{x})}$$

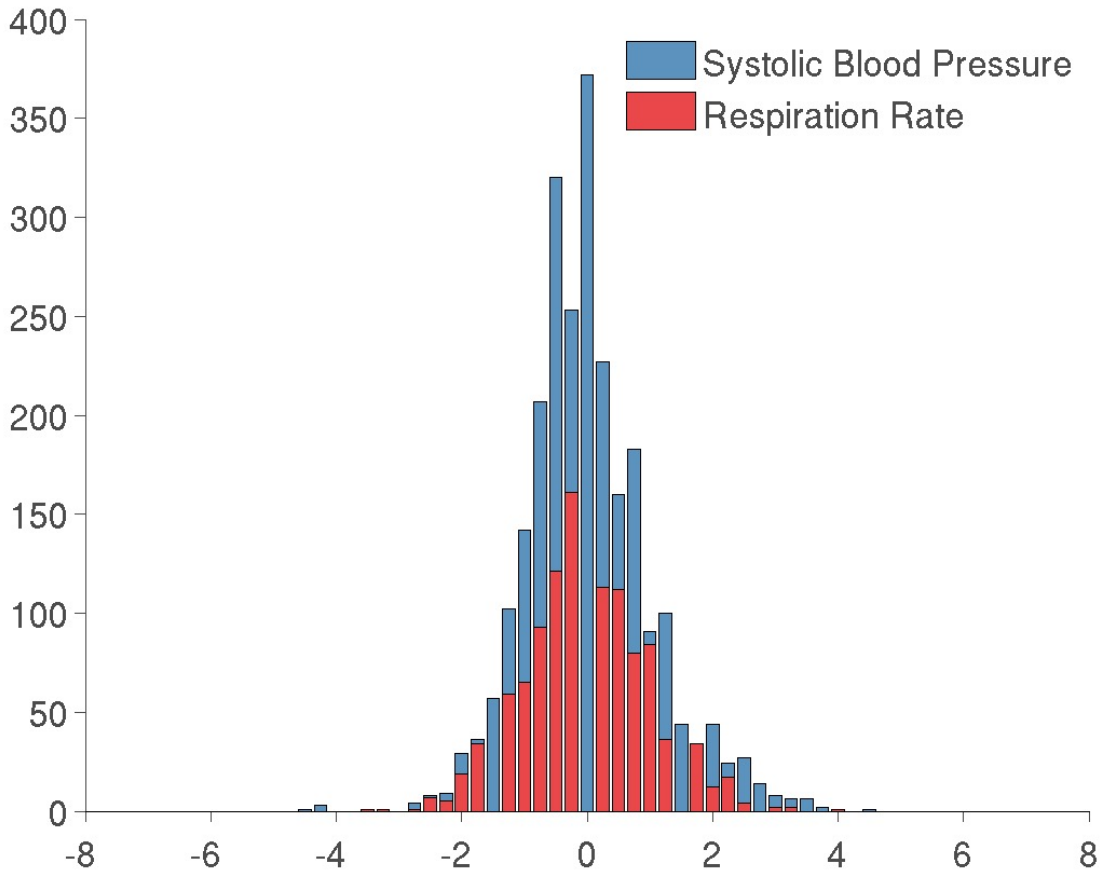
standardisation / zscore:

$$\mathbf{x}' = \frac{\mathbf{x} - \mu}{\sigma}$$

μ: mean of $\mathbf{x}$; σ : standard deviation of $\mathbf{x}$

Normalisation

- The results of zscore normalisation are shown here, where the mean μ and s.d. σ are calculated separately for each variable (blood pressure, respiratory rate, etc…)
- It is called standardisation because IFF the original data distribution were some Gaussian $\mathrm{N}(\mu, \sigma)$, then the distribution of the result will be a standard Gaussian, $\mathrm{N}(0, 1)$
- It is a commonly-made error to assume that standardisation assumes normality / Gaussianity of the data distribution



standardisation / zscore:

$$\mathbf{x}' = \frac{\mathbf{x} - \mu}{\sigma}$$

μ: mean of $\mathbf{x}$; σ : standard deviation of $\mathbf{x}$

Pre-processing

- Clinical data can be particularly **noisy** and **missing**
- **Missingness** is often either expressed as:
  - missing-at-random, perhaps due to sensor failure, data drop-out, etc.
  - missing-not-at-random, where the missingness itself may be informative.
- Examples of the latter include the fact that *measuring a physiological parameter is often itself an indicator of concern*
- Conversely, if data are missing, it might indicate that a clinician did not deem them to be sufficiently at-risk to make a measurement
- (Thus, we sometimes include “number of measurements in some period” or “time since last measurement” as a feature in our models, such that the model can have some exposure to this oft-ignored but potentially useful information.)

Missingness

- There is no “right” way to handling missing data. Typical approaches include:

1. Simply disregarding any data (e.g., patients) that have missing **any** of their variables. This is the standard approach taken in much of medical statistics.

   Advantage: very simple, and imposes no distortion on the data

   Disadvantage: very wasteful of data, especially when the number of variables measured for the individual patient is very high – one might end up retaining only a small fraction of the available data.

   (Reducing the size of a dataset makes the resulting model typically less effective than if more data were used; it also makes the evaluation of the model less reliable.)

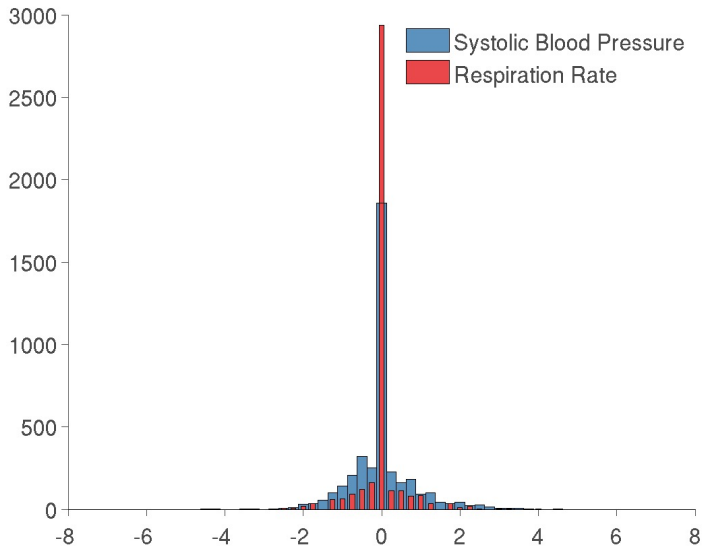
Missingness

- There is no “right” way to handling missing data. Typical approaches include:

2. Mean imputation – this involves stating that a missing value simply takes the mean of that variable across the whole dataset. If ZMUV normalisation has been used, this is equivalent to setting missing values equal to zero.

   Advantage: simple, and removes a variable from (for example) contributing to a logistic regression output (because it is effectively an input of zero to the score)

   Disadvantage: biases the data towards normality, because missing values are assumed to be directly in the middle of the “normal” range



Missingness

- There is no “right” way to handling missing data. Typical approaches include:

3. Nearest-neighbour imputation – this involves finding the “nearest neighbour” (or set of neighbours) to a data item based on those variables that are available. The missing variables are then set to the values of the nearest neighbours

   Advantage: avoids bias towards normality, because you are assuming that the data item should behave as its near neighbours

   Disadvantage: complex, requiring large numbers of comparisons between variables; there is also the potential for biasing the data towards existing examples (by assuming that data behave like their near neighbours)

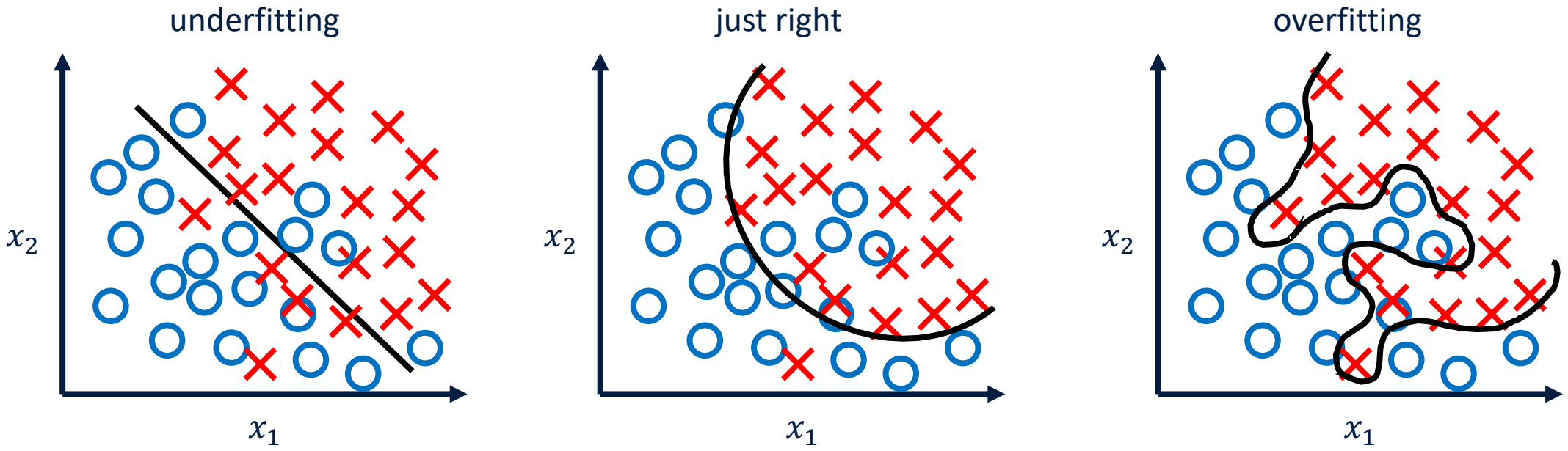
Some models intrinsically incorporate missingness, unevenly sampled data, and uncertainty
(cc’ Prof. D. Clifton, Dr . T Zhu and Gaussian Processes, who you will get a lecture from on Thursday)

How to implement ML 4 time-series?

How do I generalize my results?

- Overfitting / Underfitting, etc.
- Regularisation
- Feature Selection
- Cross-validation
- Stratification

Avoiding overfitting



Why is my model overfitting?

1. **Your model is too complex** – you have too many degrees-of-freedom in your model, and it is learning the noise in the training data, rather than the underlying structure that you wish to learn
2. **Cross validation –** you have not implemented, or you could have leakage of information between your train, validation and testing sets
3. **Dataset drift** – practice changes over time
4. **Cohort differences** – datasets may come from patients with different "case mix" (i.e., different diseases), or with different patient characteristics (e.g., different demographics)

Regularisation

Many statistical and machine learning models can easily overfit to the training data, resulting in poorer estimations, models that are not generalisable or too complex. Regularisation is often introduced to mitigate against this.

For example, large coefficient values in a regression can be penalised by adding a regularisation term to a loss function, or through reducing the number of parameters or features used in a model (i.e. feature selection).

The most common regularisers use the l_p-norm defined by:

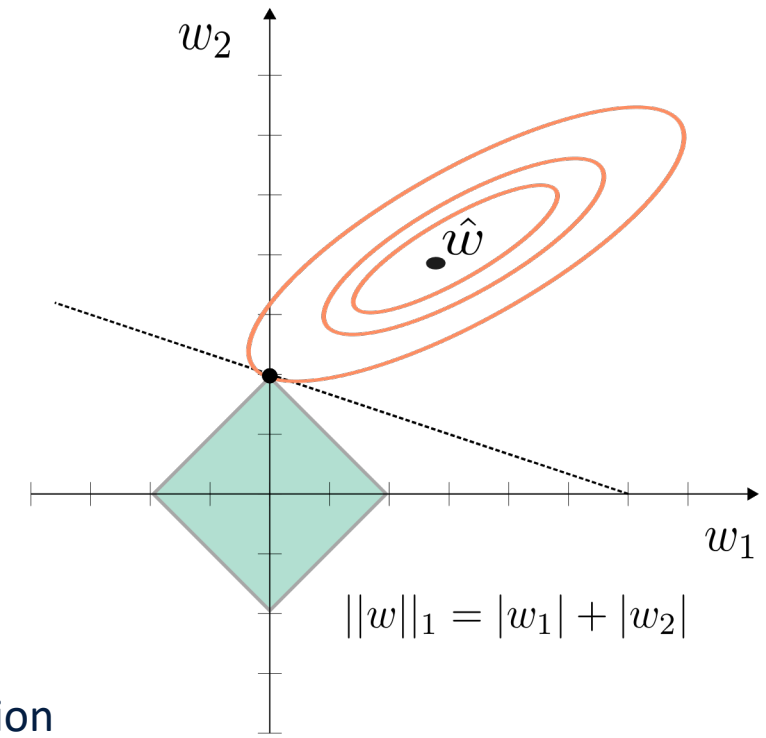
$$\|\mathbf{w}\|_P = \left(\sum_{i=1}^{N} |w_i|^P \right)^{1/P}$$

Find the parameters $\mathbf{w}$ that minimise the objective function $E(\mathbf{w})$:

$$\arg\min_{\mathbf{w}} \left\{ \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\big(y_i, f(\mathbf{x}_i, \mathbf{w})\big) + \lambda \|\mathbf{w}\|_p \right\}$$

Where $\lambda \geq 0$ is a tuning parameter defining how much the model is regularised; the larger the value of λ, the greater the amount of shrinkage.

Regularisation→Feature Selection

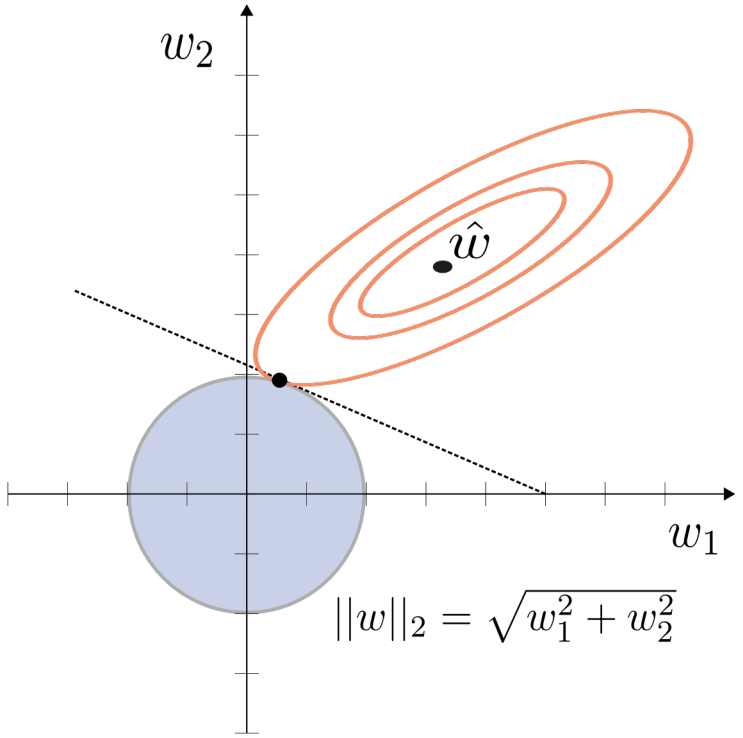


sparse solution

l_1-regularisation

shrinks non-important weights $\mathbf{w}$ (or features) to zero → feature selection

a.k.a. Least Absolute Shrinkage and Selection Operator (LASSO)



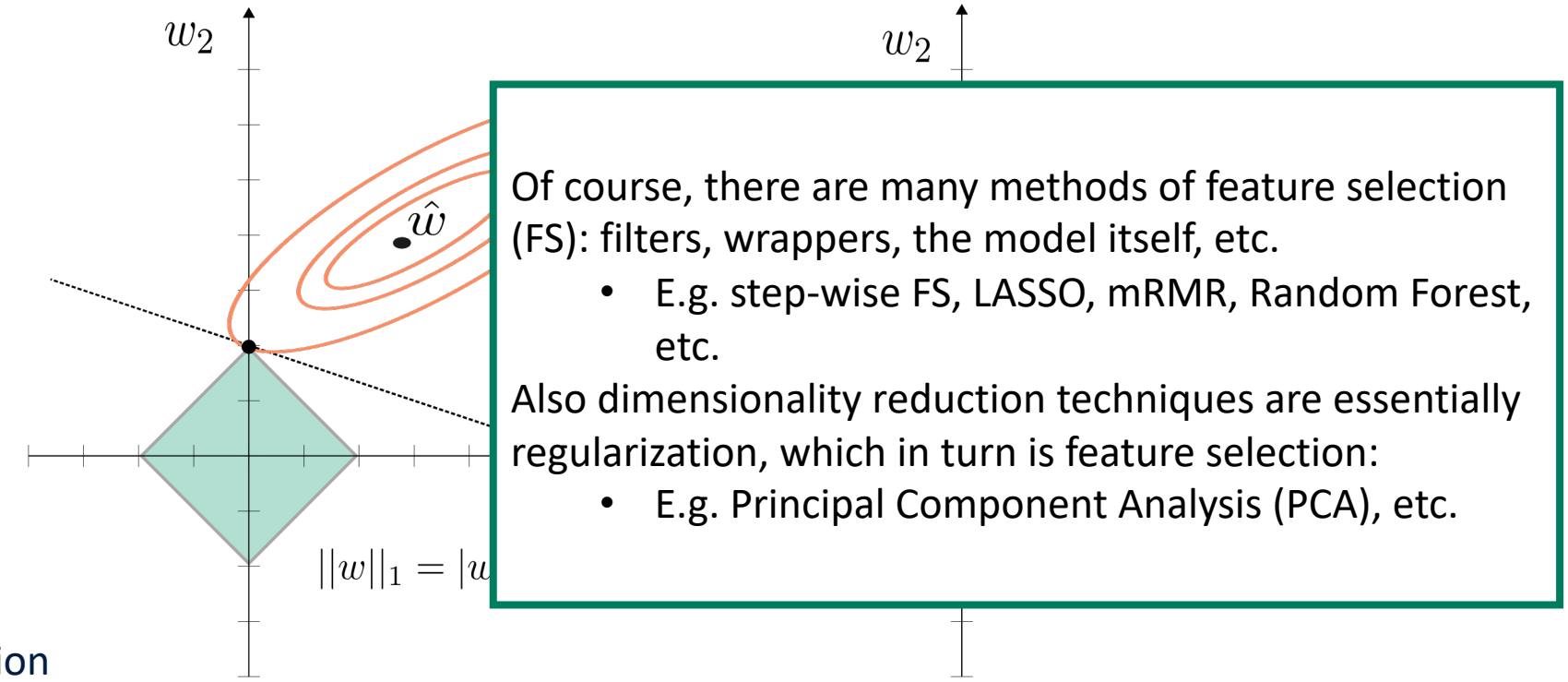
l_2-regularisation

weights the features by importance

a.k.a. Ridge Regression

Diagram adapted from: Trevor Hastie, Robert Tibshirani, and Martin Wainwright. Statistical learning with sparsity: the lasso and generalizations. CRC press, 2015

Regularisation→Feature Selection



Of course, there are many methods of feature selection (FS): filters, wrappers, the model itself, etc.

- E.g. step-wise FS, LASSO, mRMR, Random Forest, etc.

Also dimensionality reduction techniques are essentially regularization, which in turn is feature selection:

- E.g. Principal Component Analysis (PCA), etc.

sparse solution

l_1-regularisation

shrinks non-important weights $\mathbf{w}$ (or features) to zero → feature selection

a.k.a. Least Absolute Shrinkage and Selection Operator (LASSO)

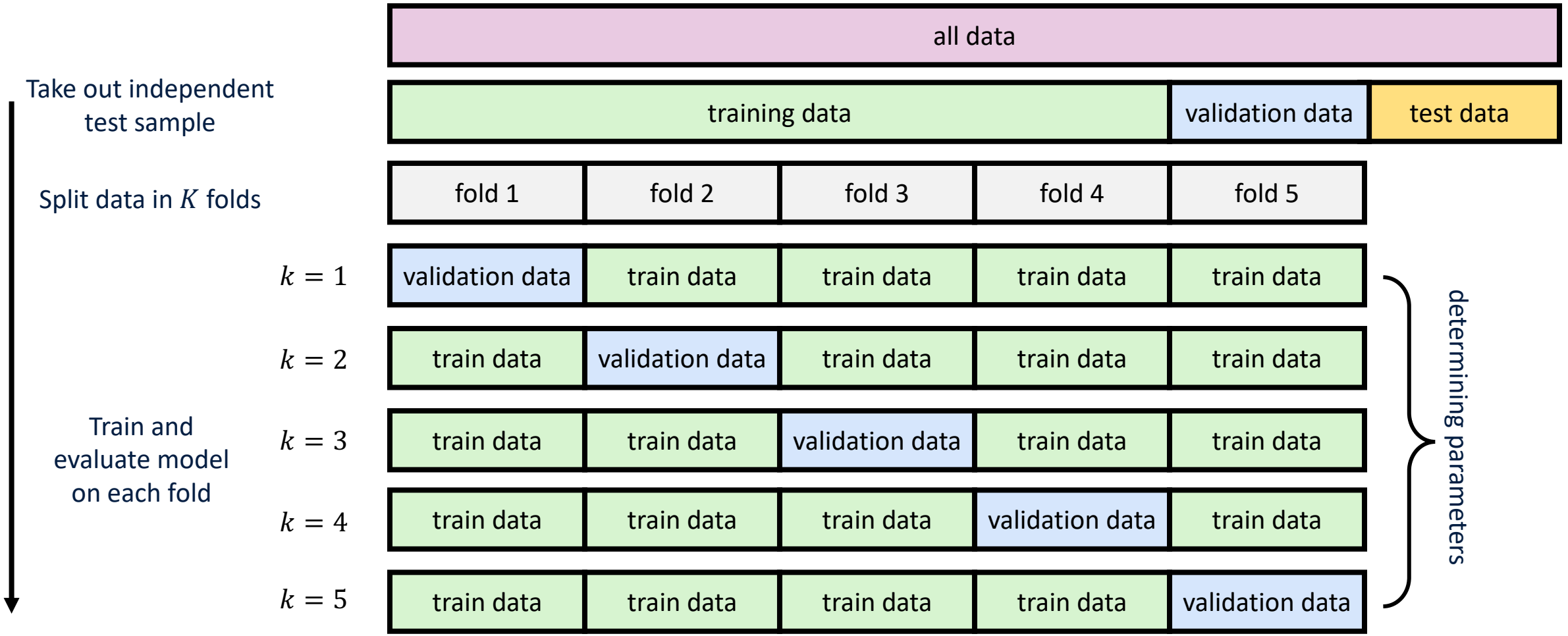
l_2-regularisation

weights the features by importance

a.k.a. Ridge Regression

Cross-validation (CV):

Cross-validation (CV): K-Fold



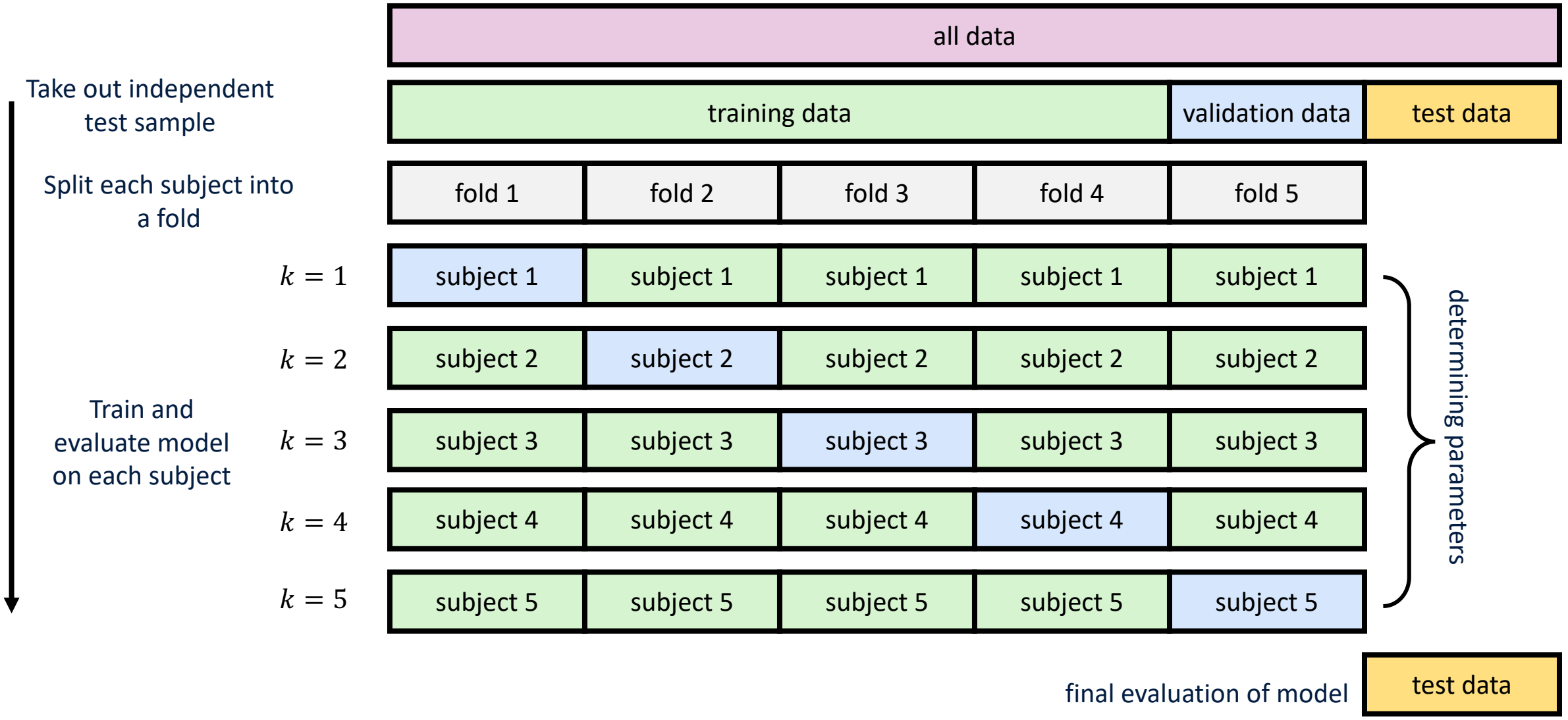
N.B: we must always split folds *subject-wise* to ensure no information leakage

Credit: https://scikit-learn.org/stable/modules/cross_validation.html

final evaluation of model

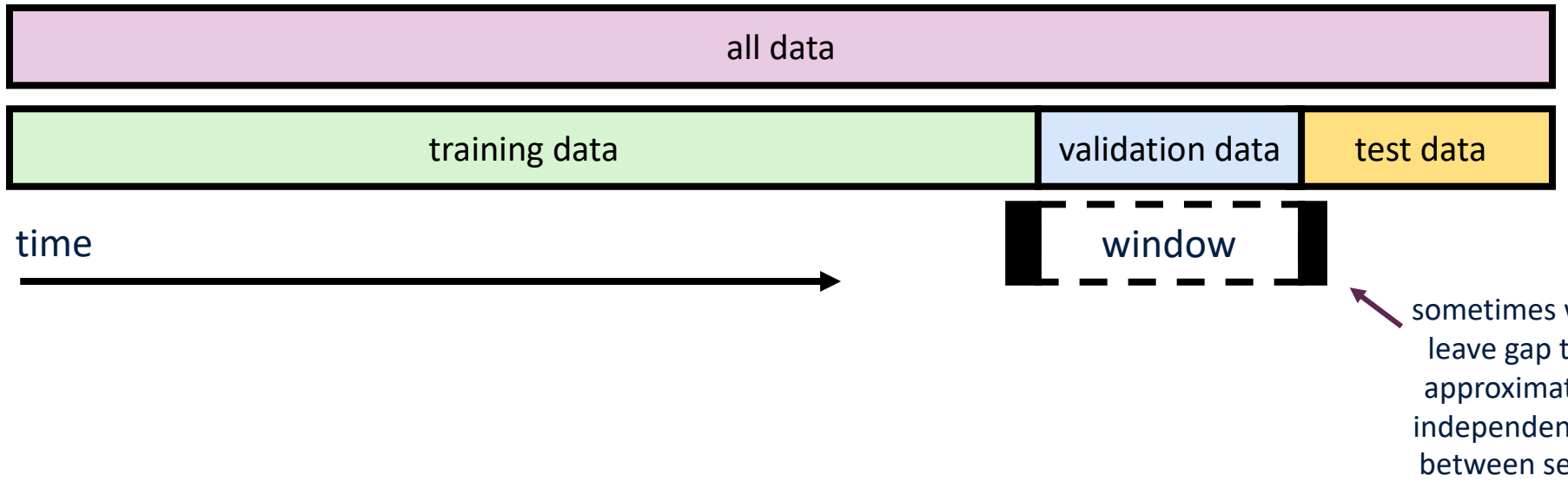
test data

Cross-validation (CV): Leave-one-out



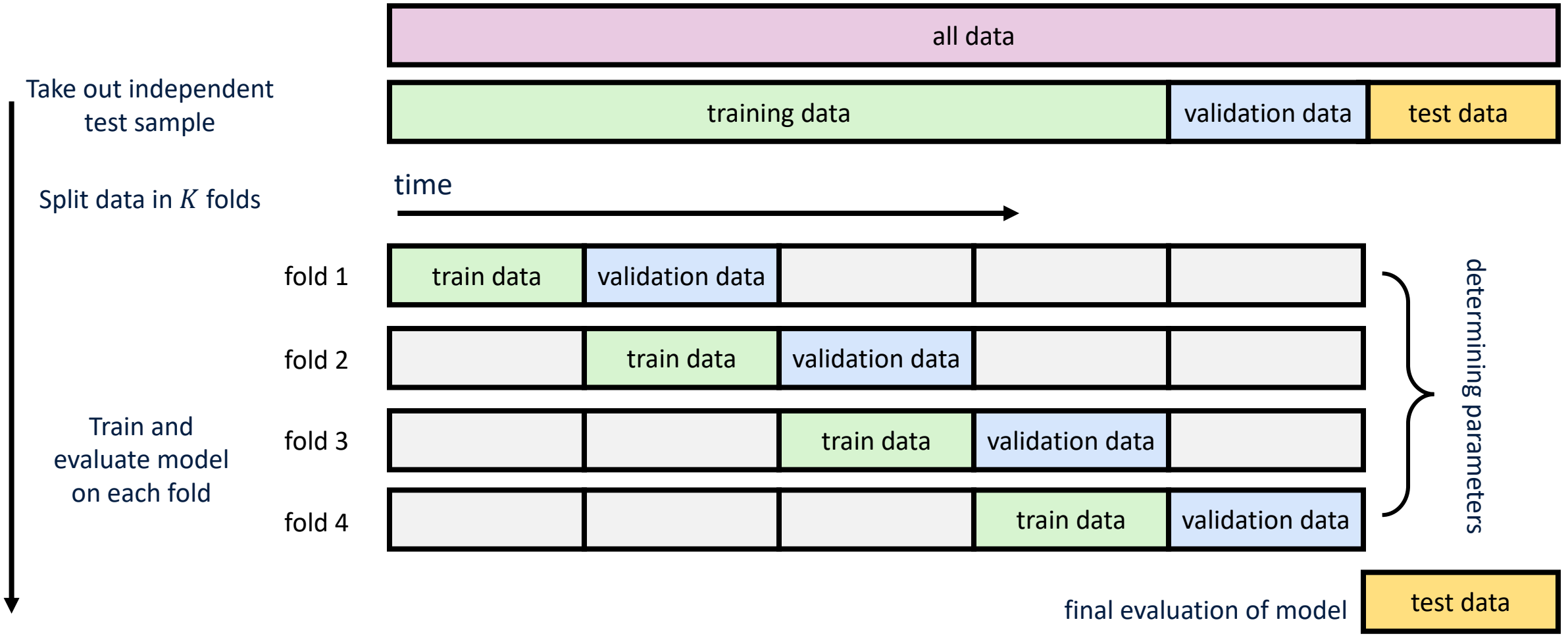
Cross-validation (CV): In Time

Take out independent test sample

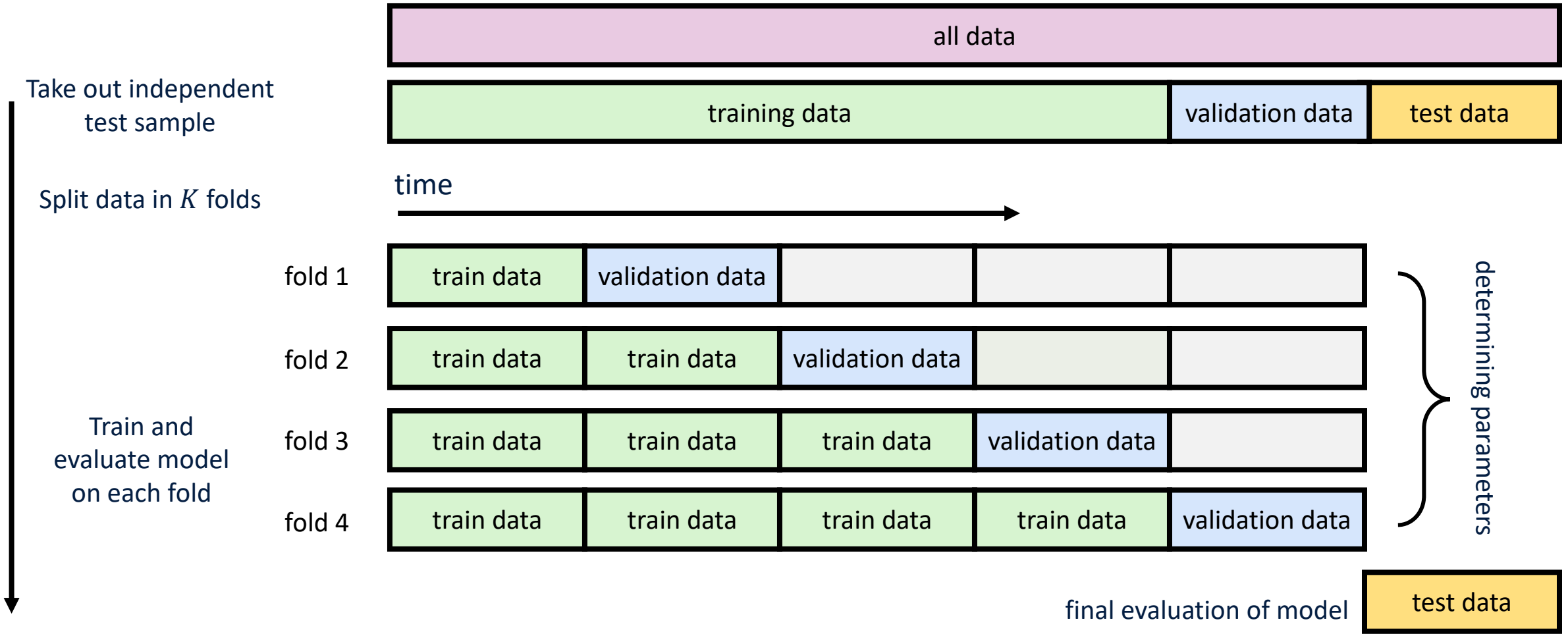


See Cerqueira et al. (2019) https://arxiv.org/pdf/1905.11744.pdf

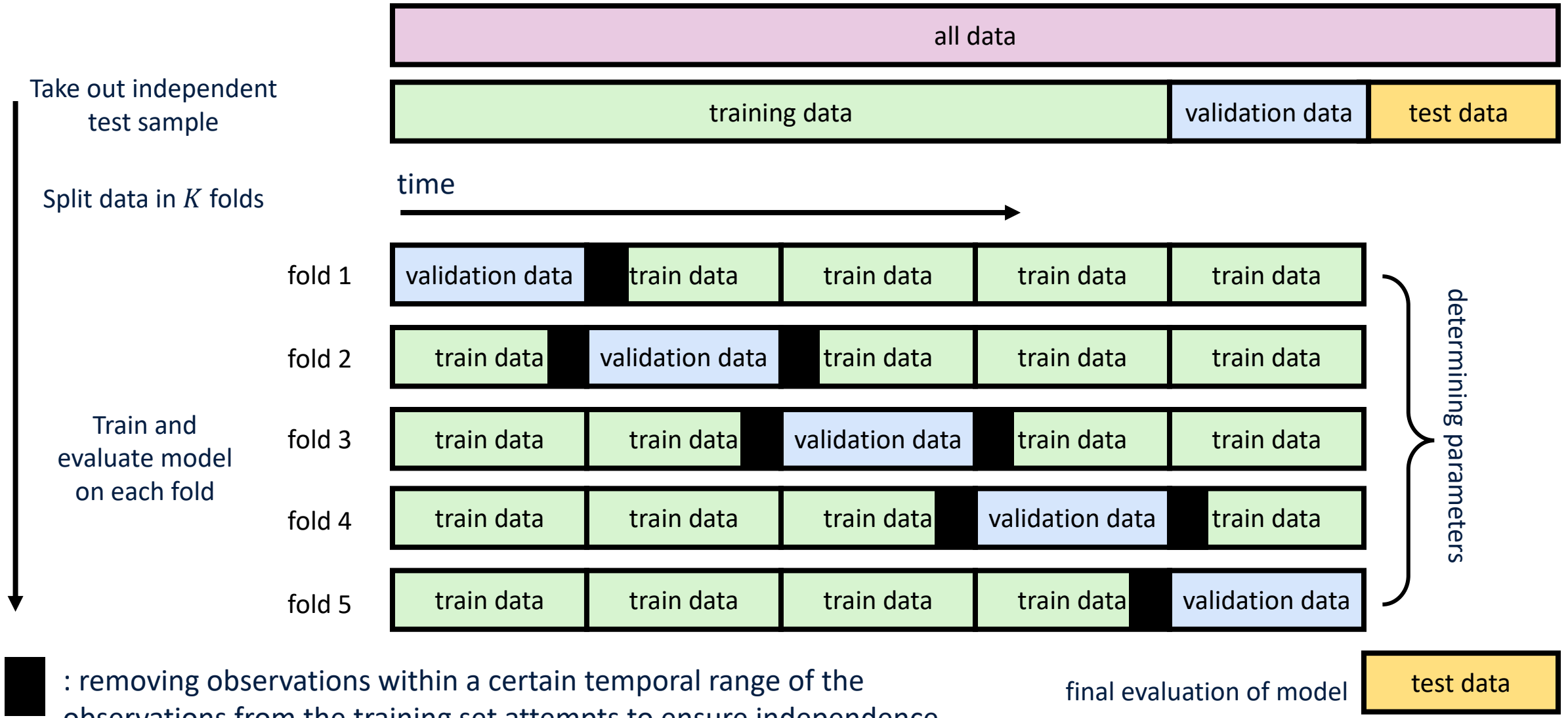
Cross-validation (CV): K-Fold



Cross-validation (CV): Forward Chaining

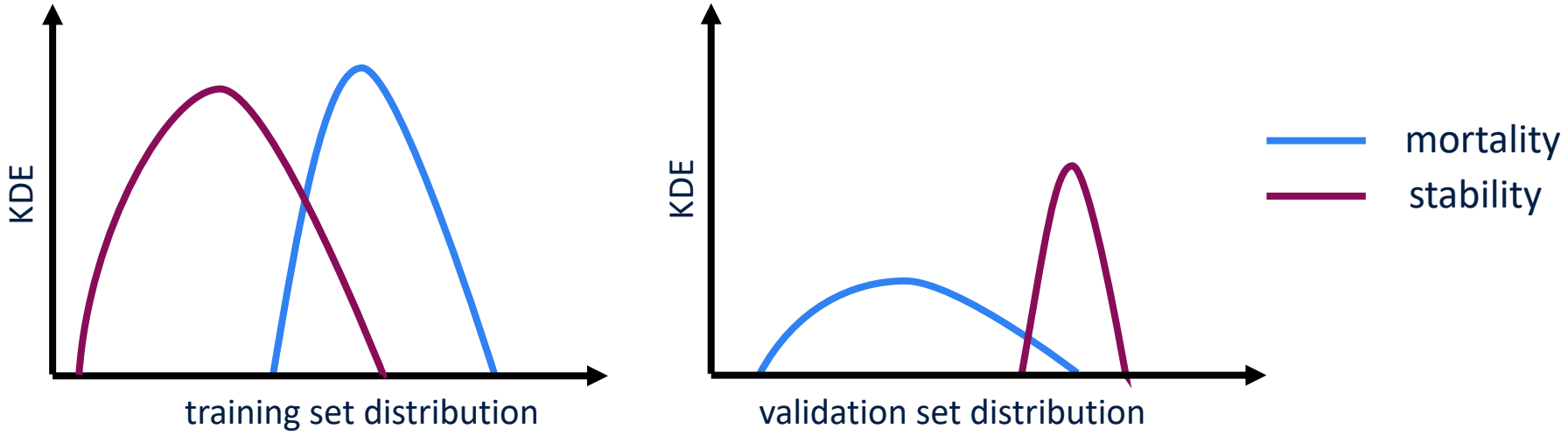


Cross-validation (CV): HV-Blocked



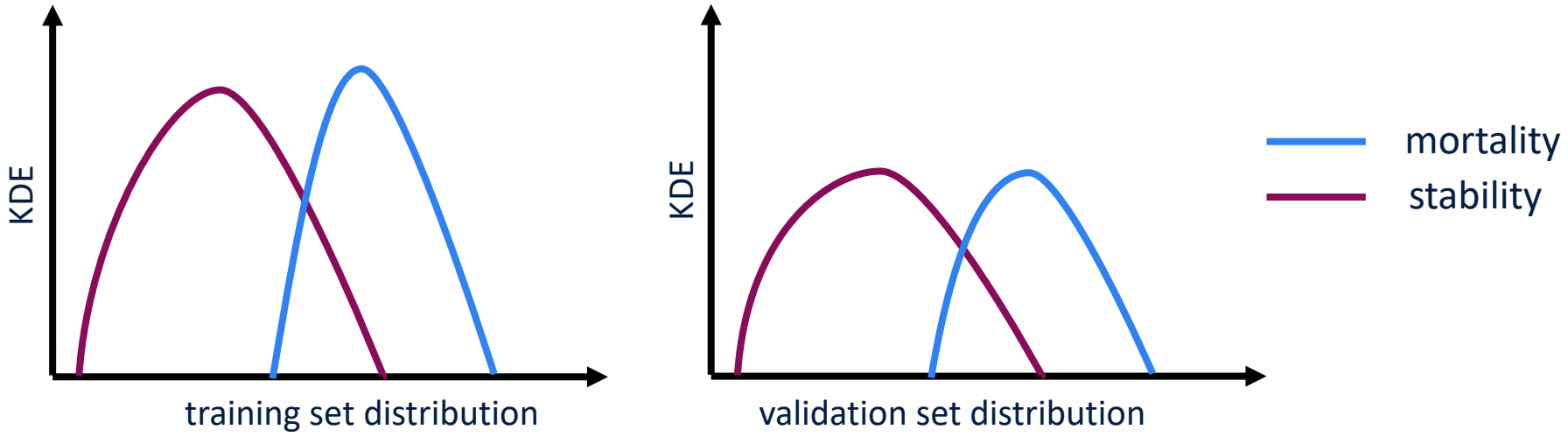
Cross-validation (CV): Stratification

- For a model to be evaluated in an unbiased manner, we need to ensure that the training and validation sets are *roughly* similar
  - Such as equal representation from each class
  - The same distribution of participants and attributes between train and validation (e.g. age, sex, clinical condition)
- Stratification and balancing, example:
  - Over-sample the under-represented class (e.g., mortality)
  - Under-sample the over-represented class (e.g., stability)



Cross-validation (CV): Stratification

- For a model to be evaluated in an unbiased manner, we need to ensure that the training and validation sets are *roughly* similar
  - Such as equal representation from each class
  - The same distribution of participants and attributes between train and validation (e.g. age, sex, clinical condition)
- Stratification and balancing, example:
  - Over-sample the under-represented class (e.g., mortality)
  - Under-sample the over-represented class (e.g., stability)



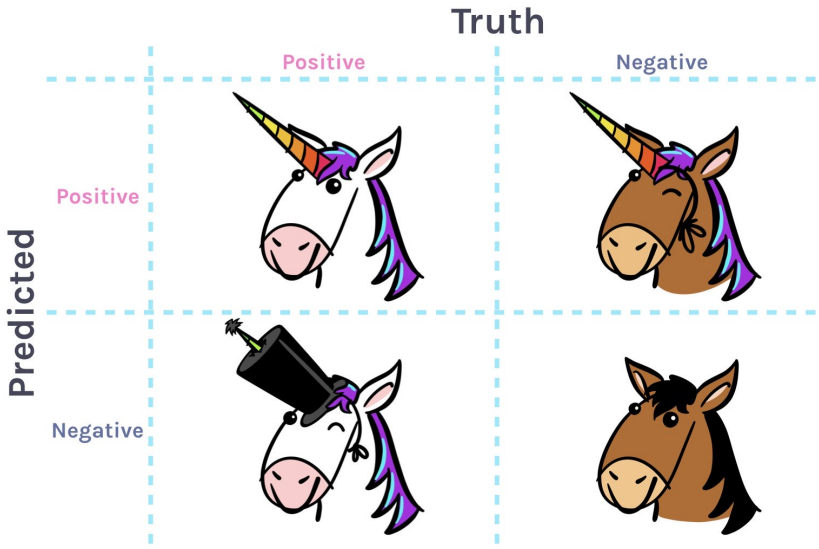
How to implement ML 4 time-series?

How do I evaluate my time-series model?

- Confusion matrices
- The receiver operating characteristic (ROC)
- Dealing with imbalanced data
- Regression evaluation
- Event prediction

Contingency tables

a.k.a. confusion matrices



Credit: https://twitter.com/apreshill/status/1199727355251445760?s=20

Predicted \ Truth	True	False
True	**True positive**	**False positive**[1]
False	**False negative**[2]	**True negative**

TP / (TP + FP) = Positive Predictive Value

sometimes called precision

TN / (TN + FN) = Negative Predictive Value

TP / (TP + FN) = Sensitivity

sometimes called recall

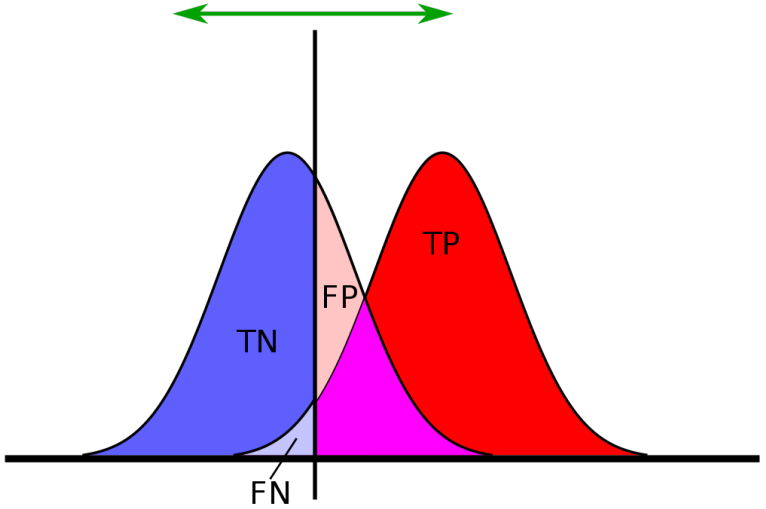
TN / (TN + FP) = Specificity

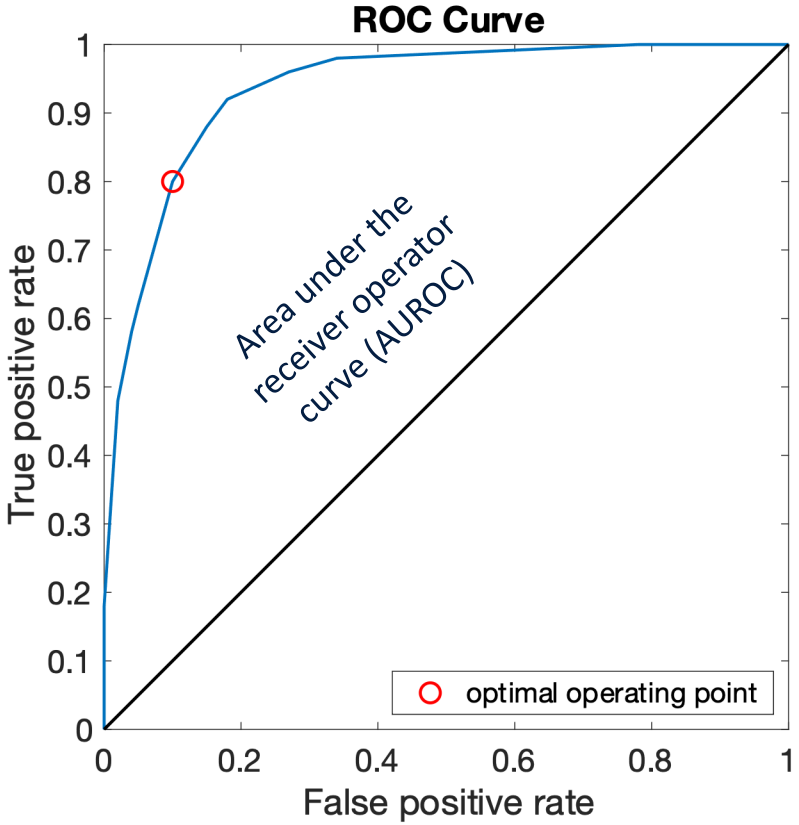
[1] or type I error, or α-error
[2] or type II error, or β-error

The receiver operating characteristic

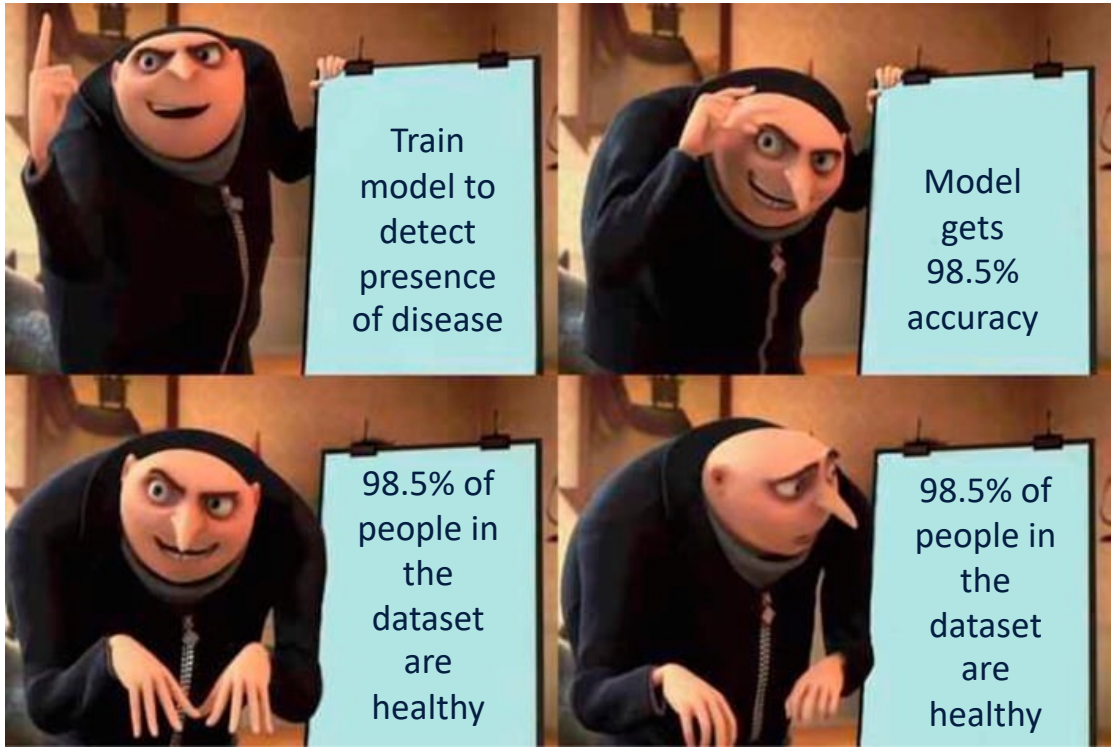
- Most medical evidence considers some sort of ROC curve
- The ROC curve is a series of (sensitivity, specificity) pairs, typically joined together, often leading to a “stepwise” nature of the graph
- We obtain the ROC curve by changing the threshold on our score in steps:

low threshold = more alarms = higher sensitivity
high threshold = fewer alarms = higher specificity





Imbalanced Data



Credit: me

Imbalanced Data

*remember, the minority example should be the positive class!

How many of the positive predictions are correct?

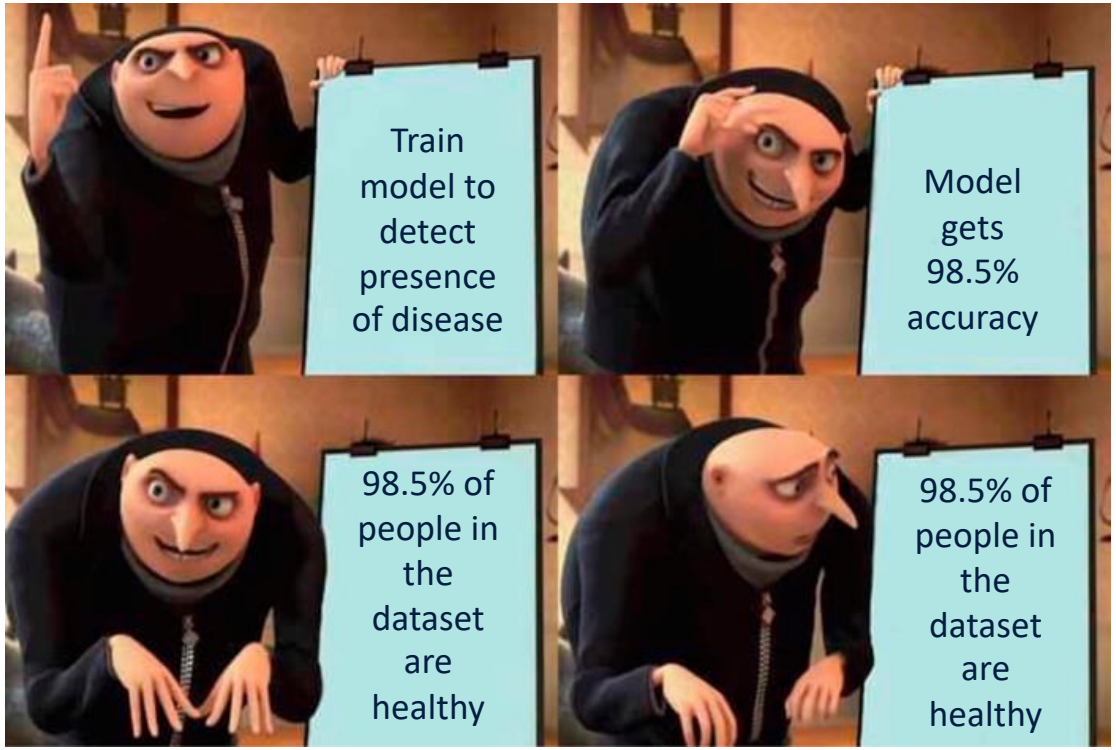
$$precision = TP \,/\, (TP \; + FP)$$

How many actual positive cases are identified?

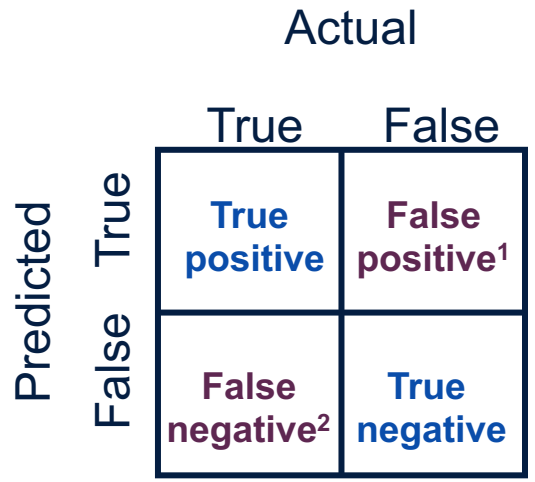
$$recall = TP \,/\, (TP \; + \; FN)$$

F-score: harmonic mean between precision and recall

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision \times recall}$$



Credit: me



Imbalanced Data

*remember, the minority example should be the positive class!

How many of the positive predictions are correct?

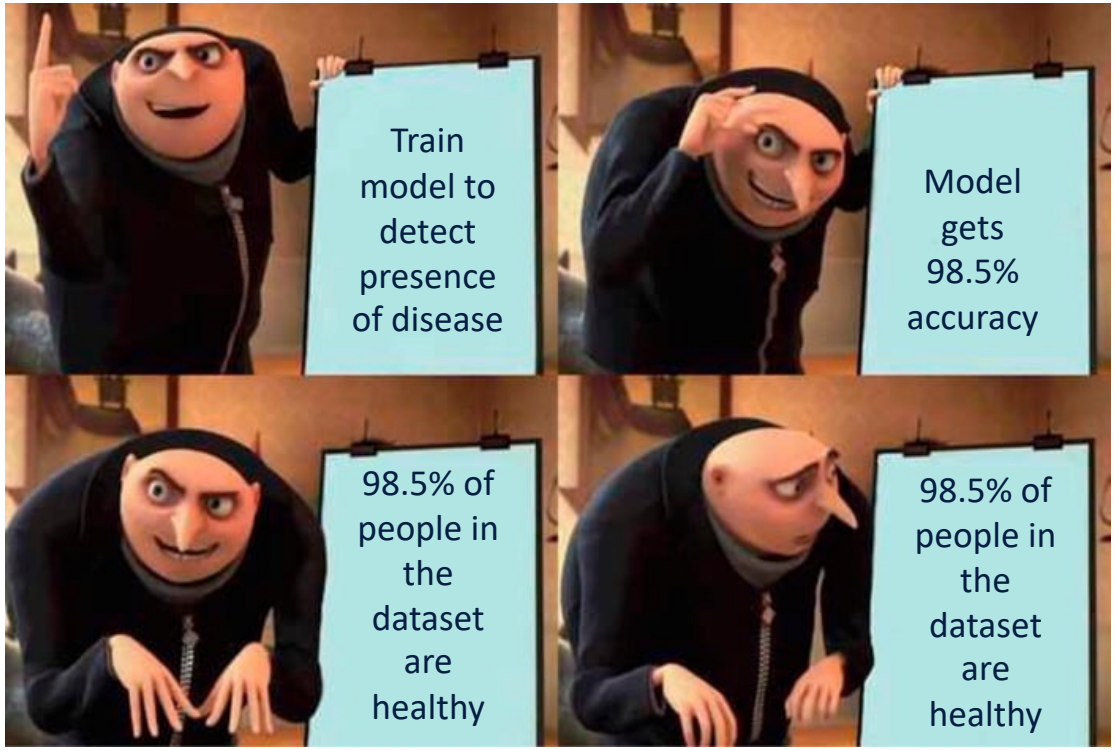
$$precision = TP \,/\, (TP + FP)$$

How many actual positive cases are identified?

$$recall = TP \,/\, (TP + FN)$$

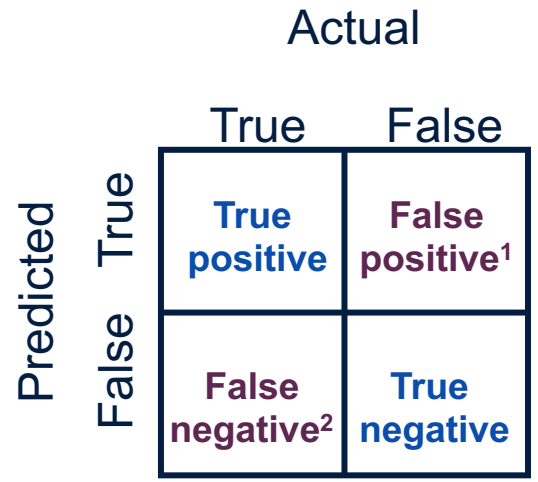
F-score: harmonic mean between precision and recall

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision \times recall}$$

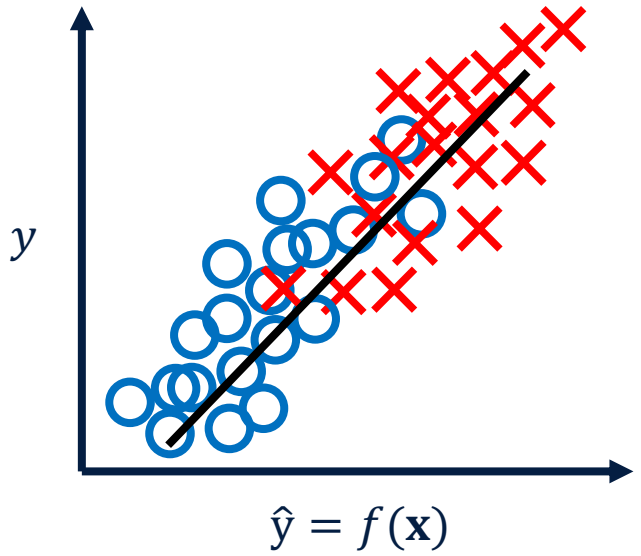


Credit: me

- For multi-class problems, the confusion matrix can simply be expanded and the macro-average of each metric (i.e. for each binary classification task) can be reported instead (e.g. the macro F1-score (MF1))
- We could also re-sample (bootstrap) the data by taking balanced subsampled and reporting balance-sensitive measures (i.e. accuracy)
- *Many other imbalanced data metrics exist!*

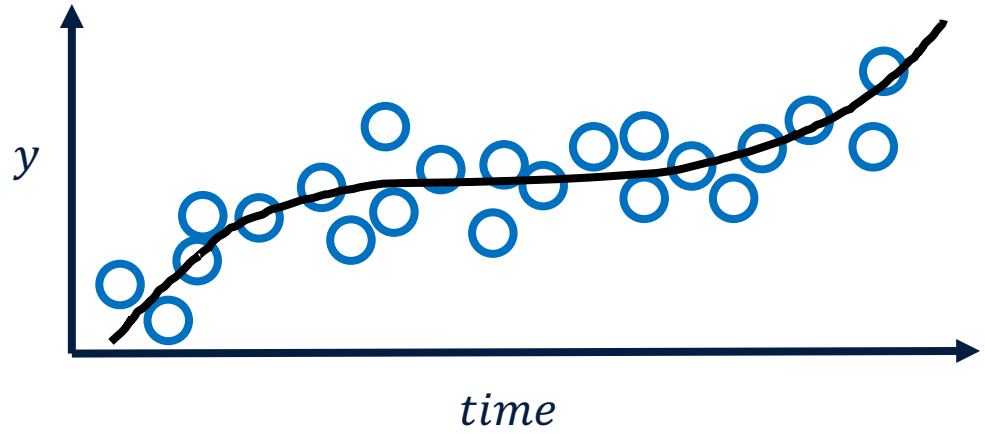


Regression based Evaluation



patient stable

patient admitted to ICU



**N.B. both metrics are reported in the original units, which helps interpretating the results*

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

penalises all deviations equally

$$RMSE = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

penalises large deviations more

ML 4 Time-series: Essential Methodology

What have we learned?

1. How to implement ML 4 time-series:
    - How to work with time, choose your model and setting up your task
    - How to preprocess your data
2. How to generalize your results
    - Regularisation
    - Cross-validation
3. How do I evaluate your time-series model
    - Classification vs. regression
    - Dealing with imbalanced data

Lab 1

ML 4 Timeseries: Mortality Prediction

In this lab you will learn and implement a basic machine learning pipeline for time-series analysis mortality prediction using ScikitLearn.

Your tasks:

1. Visualise the distribution of features & an example over time
2. Generate statistical summary features: $\bar{\mathbf{x}}; \overline{\boldsymbol{x}} + \boldsymbol{\sigma}; \overline{\boldsymbol{x}} + \widetilde{\boldsymbol{x}}$
3. Implement stratified k-fold cross validation
4. Apply the necessary pre-processing steps
   - normalisation
   - outlier removal: mean imputation
5. Train two off-the-shelf machine learning models for mortality prediction.
   - Train a linear model, e.g. Logistic Regression
   - Train a non-linear model, e.g. Random Forest
6. Evaluate the model
   - Compare between model performance
   - Determine the important features for mortality prediction
   - Compare model performance between various types of features
7. Re-implement time-series specific features
   - $\mathbf{a}_l$ the autocorrelation coefficients at various time lags, l
   - feel free to come up with your own, for example windowing the time-series
   - train your models again and compare model performance between statistical and time-series feature types