Boring but important disclaimers:

- If you are not getting this from the GitHub repository or the associated Canvas page (e.g. CourseHero, Chegg etc.), you are probably getting the substandard version of these slides Don't pay money for those, because you can get the most updated version for free at

  `https://github.com/julianmak/OCES5303_ML_ocean`

  The repository principally contains the compiled products rather than the source for size reasons.
- Associated Python code (as Jupyter notebooks mostly) will be held on the same repository. The source data however might be big, so I am going to be naughty and possibly just refer you to where you might get the data if that is the case (e.g. JRA-55 data). I know I should make properly reproducible binders etc., but I didn't...
- I do not claim the compiled products and/or code are completely mistake free (e.g. I know I don't write Pythonic code). Use the material however you like, but use it at your own risk.
- As said on the repository, I have tried to honestly use content that is self made, open source or explicitly open for fair use, and citations should be there. If however you are the copyright holder and you want the material taken down, please flag up the issue accordingly and I will happily try and swap out the relevant material.

OCES 5303 :
ML methods in Ocean Sciences

Session 1: basics of ML, Python refresher,
regression + optimisation

Outline

- admin things (canvas, GitHub, Colab)
- approach of the course
  $\rightarrow$ lecture, workshop, assessment
- basics of ML
  $\rightarrow$ unsupervised vs. supervised learning
  $\rightarrow$ ML as an regression/optimisation problem

Content of course is somewhat ocean science motivated, but skills are entirely generic

Practicalities

Instructors:

I Julian Mak (`jclmak@ust.hk`)

TA Jonathan Lee (`hcleear@connect.ust.hk`)

Course grade breakdown:

method	
quizzes	$4\% \times 5 = 20\%$
assignments	$15\% \times 2 = 30\%$
short project	$25\% \times 1 = 25\%$
interview style exam	25%

- pass mark is going to be 50%
- "A" boundary will be around 85%

  (I don't grade to a curve)

- assignments with plagiarism will get a minimum of **zero**

Practicalities

- weekly **Thurs 0900 to 1150**

  → first 60-90 mins is lecture / outline (scaled down as course moves on)

  → rest of the time is the computer workshop, the **main learning part**

- F2F in class

  → it's more for you: us trying to help you code/debug without control/access to your computer is usually frustrating for both parties

- **Prerequisite**:

  → will assume you are familiar with data analysis and some Python

  → some maths background would be helpful

Approach of course

What this course is:

- a hands-on intro to basics of data science and machine learning methods in Python

  $\rightarrow$ on understanding principles, sampling some methodologies, learn by tackling some problems (some contrived, some 'realistic')

  $\rightarrow$ transferable skills

- **you have to try do stuff**

  $\rightarrow$ the lectures are secondary to the computer workshops

  $\rightarrow$ "you get better at cooking by cooking"

- **focus on teaching you to teach yourself**

Approach of course

This course is not:

- a computer science and/or Python course as such

  → we use Python but the focus is the analysis tools
- a statistics/maths course

  → although you will be dealing with numbers quite a bit

(!!! will not deal with text, language processing and LLMs in this course !!!)

After this course you will not be:

- a computer/machine learning wizard

  → it will hopefully give you some mindset and tools to work towards being one (if you want to)
- know how to solve every data problem in ocean science

  → but it should teach you how you might get started on other problems

Material

- most of the material is in the Jupyter notebooks

  → most things in slides are actually taken from the notebooks

  → if you don't want to listen to me you could always go straight to the notebooks

- for extra reading, try Google (search the topic)
- for Python syntax queries, try Google (stack overflow / stack exchange)
- for help with debugging, try Google (stack overflow / stack exchange)
- for help with science queries, try Google

- you could try me but I would probably tell you to try Google first (teaching you to teach yourself)

Syllabus

S01 Basics of ML, Python refresher, `sklearn`, data handling, regression and optimisation

S02 Linear models, dimension reduction, clustering, cross-validation, model hyper-parameters

S03 Classifications, random forests + (gradient) boosting

S04 ANNs/MLPs + PyTorch

S05 CNNs

S06 Autoencoders

S07 RNNs

S08 GANs

S09 PINNs

S10 FNOs / Neural ODEs

- lecture material is somewhat **cumulative**, heavy focus on neural networks
- quizzes every two weeks, short assignments set at S04, S07, project at S10

  $\rightarrow$ marking criteria when assignments are released (project is marked slightly differently)
- material for up to three bonus lectures to be confirmed (e.g. SINDy, TDA, diffusion models, Neural ODES, FNOs etc.)

Some propaganda to start with

ML algorithms are:

- algorithms + tools, and that's it

  $\rightarrow$ very powerful, but context dependent

- usually black box

  $\rightarrow$ it can work wonderfully / fail spectacularly, but you don't necessarily know why...

!!! prudent to do checks!

(see Lec 02, cross-validation)

Figure: Hermeowus Mora, disciple of Hermaeus Mora the Daedric prince of knowledge and memory

Some propaganda to start with

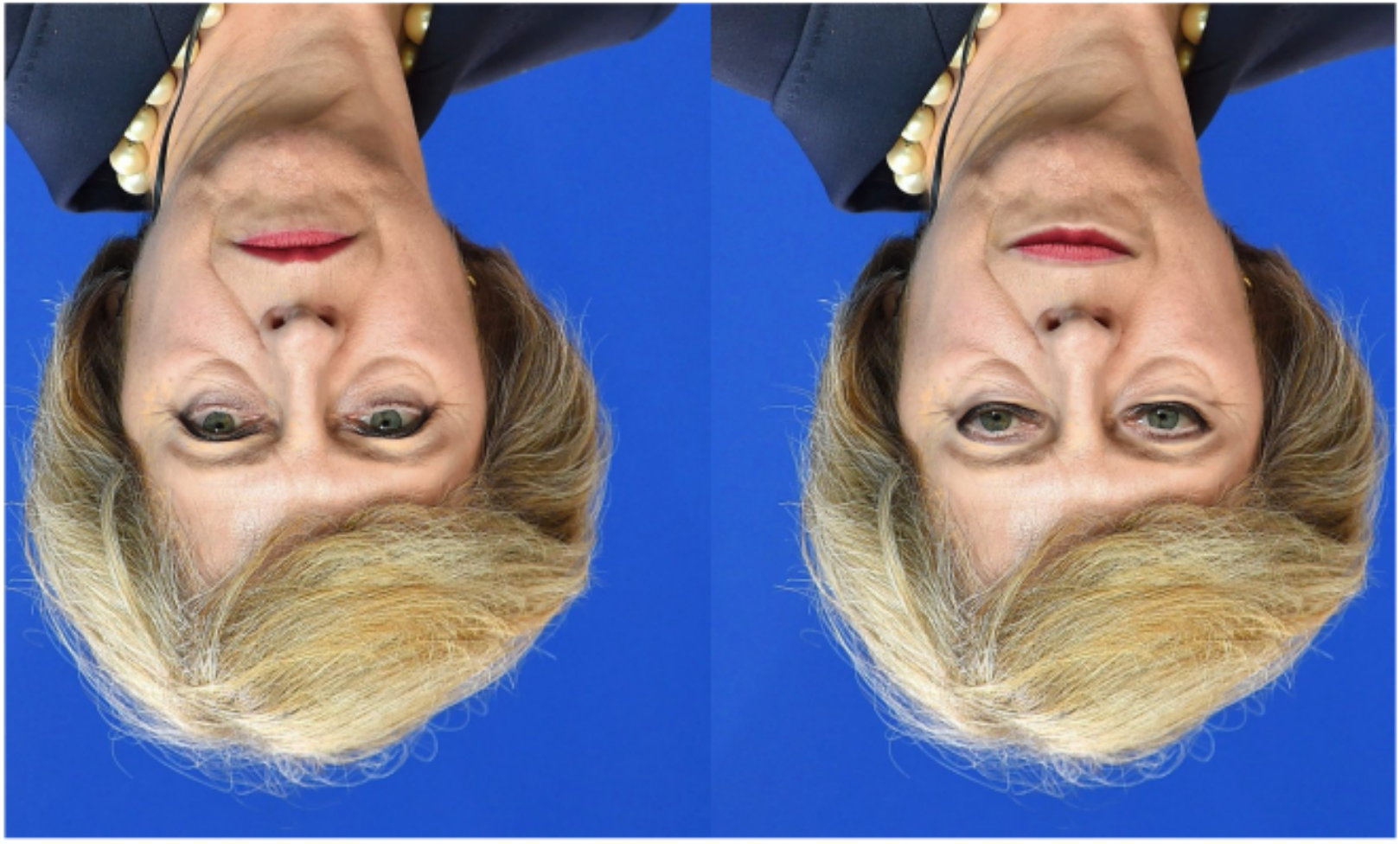

Figure: Cursed beast?

Some propaganda to start with

Figure: Cursed

Machine learning + regression

Recall regression is that, for X the input, y the output, f the model, we want

$$y = f(X)$$

- prediction or forward problem: have X and f, want y
- inference or inverse problem: have y and f, want X
- inference/regression/Machine Learning: have X and y, want f

Unsupervised vs. supervised

- unsupervised ML is where data is unlabelled, and algorithm picks out features by themselves

Figure: Eigenpets essentially. Figure adapted from Fig. 10 of Brunton, Brunton, Proctor & Kutz (2013); will do an easier case of this in Lec 03.

Unsupervised vs. supervised

- supervised ML is where data is labelled, and algorithm fits model between input and output
  $\rightarrow$ often want this for prediction purposes

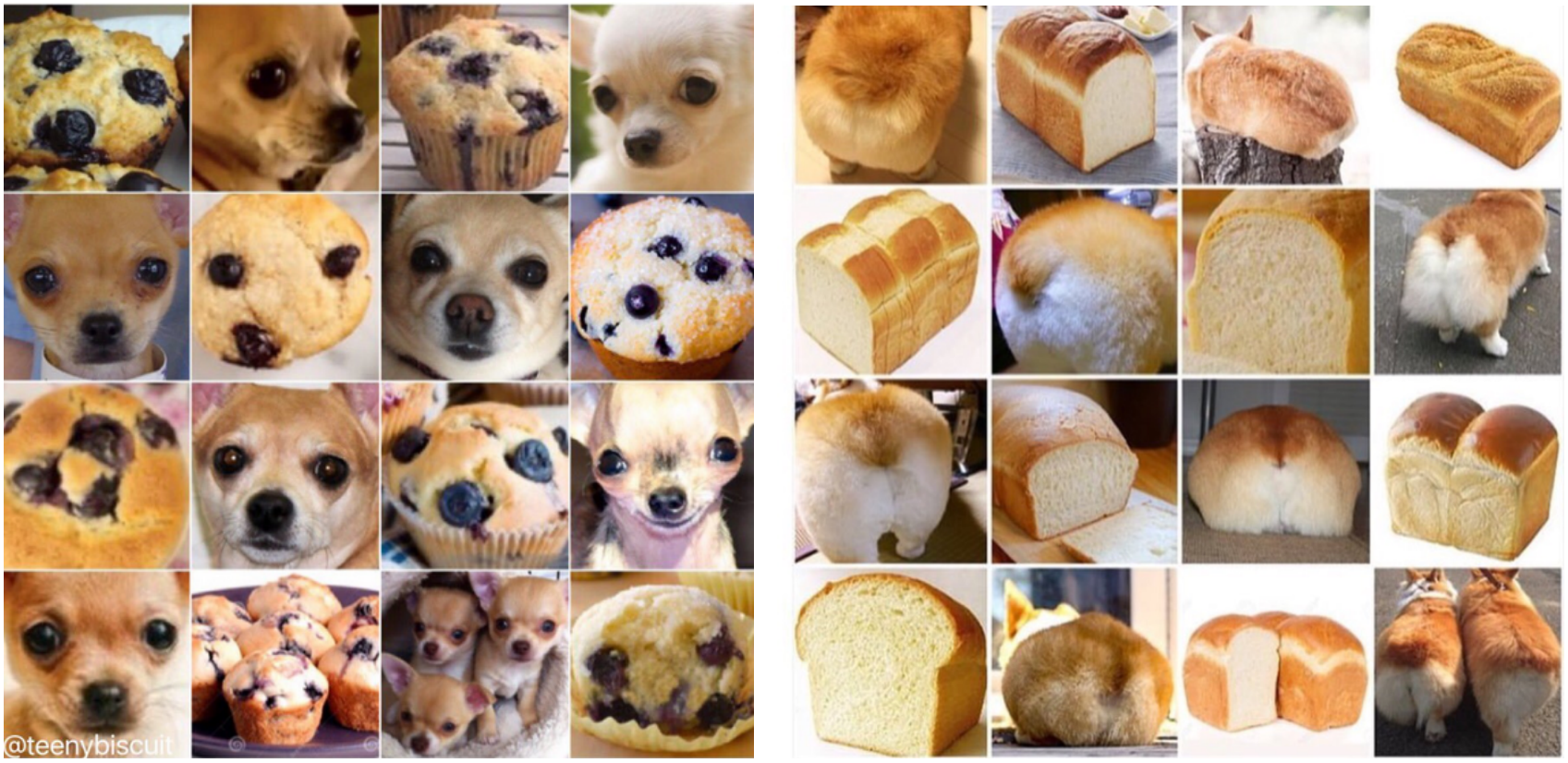

Figure: Various entries from the "animal or things" meme, as found on the internet.

Unsupervised vs. supervised

Figure: Various entries from the "animal or things" meme, as found on the internet.

Data handling: functions

- suppose I take

$$f(t) = \sin(t)$$

$\rightarrow$ then for $t = \{0, 1, 2 \ldots\}$, I store them as an array as

$$\texttt{f} = [\sin(0), \sin(1), \sin(2), \ldots]$$
$$= [0.000, 0.841, 0.909, \ldots]$$

$\rightarrow$ plotted out as the blue curve

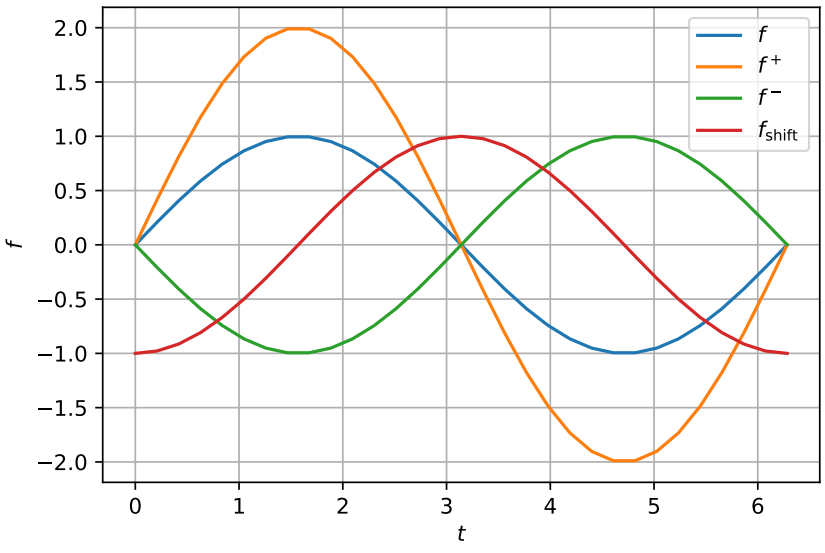

Figure: Some curves.

Data handling: numerical data

	1948											2019
1948	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99
1949	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99
1950	24.55	25.06	25.87	26.28	26.18	26.46	26.29	25.88	25.74	25.69	25.47	25.29
1951	25.24	25.71	26.90	27.58	27.92	27.73	27.60	27.02	27.23	27.20	27.25	26.91
1952	26.67	26.74	27.17	27.80	27.79	27.18	26.53	26.30	26.36	26.26	25.92	26.21
1953	26.74	27.00	27.57	28.04	28.28	28.12	27.43	26.94	27.01	26.87	26.88	27.00
1954	26.98	27.03	26.90	26.64	27.12	26.80	26.11	25.43	25.12	25.23	25.57	25.26
1955	25.61	25.81	26.22	26.60	26.66	26.55	26.15	25.51	25.28	24.41	24.25	24.57
1956	25.34	25.76	26.46	26.85	27.13	26.81	26.23	25.68	25.73	25.75	25.56	25.71
1957	26.04	26.54	27.46	28.23	28.55	28.36	28.17	27.69	27.44	27.42	27.62	27.90
1958	28.33	28.24	28.27	28.27	28.31	27.99	27.32	26.85	26.40	26.45	26.75	26.62
1959	27.07	27.18	27.47	27.88	27.70	27.37	26.44	26.09	25.92	26.24	26.04	26.18
1960	26.27	26.29	26.98	27.49	27.68	27.24	26.88	26.70	26.44	26.22	26.26	26.22
1961	26.23	26.56	26.94	27.36	27.75	27.67	26.89	26.19	25.78	25.71	26.07	25.97
1962	25.96	26.19	26.80	27.13	27.05	27.08	26.76	26.33	25.94	25.97	25.75	25.67
1963	25.77	26.22	27.18	27.78	27.63	27.62	27.78	27.48	27.40	27.36	27.47	27.62
1964	27.34	27.13	27.02	26.95	26.82	26.59	26.33	25.60	25.32	25.37	25.26	25.23
1965	25.66	26.19	26.94	27.38	27.99	28.09	27.90	27.97	28.01	28.17	28.12	27.96
1966	27.67	27.55	28.21	28.16	27.55	27.64	27.33	26.48	26.27	26.22	26.23	26.03
1967	25.88	26.11	26.50	26.74	27.35	27.47	26.97	26.44	25.86	25.97	26.08	25.95
1968	25.69	25.68	26.33	27.10	27.19	27.88	27.58	27.01	26.72	26.75	27.20	27.27
1969	27.50	27.86	27.82	28.13	28.29	27.69	27.08	27.02	27.15	27.34	27.10	26.98

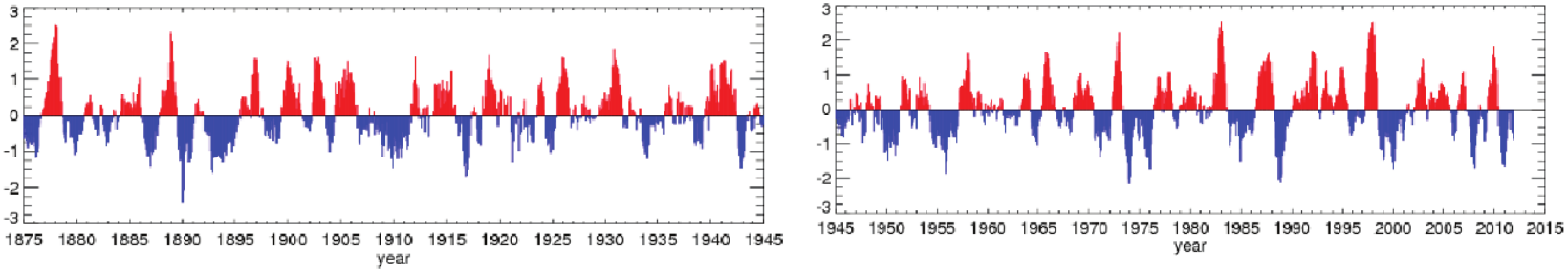

Figure: El-Niño 3.4 data (with masking values and averaged accordingly), and plotting the anomalies (relative to the de-trended signal).

Data handling: numerical data

- multi-dimensional arrays can be visualised as images

  $\rightarrow$ simulation data, satellite data, etc.

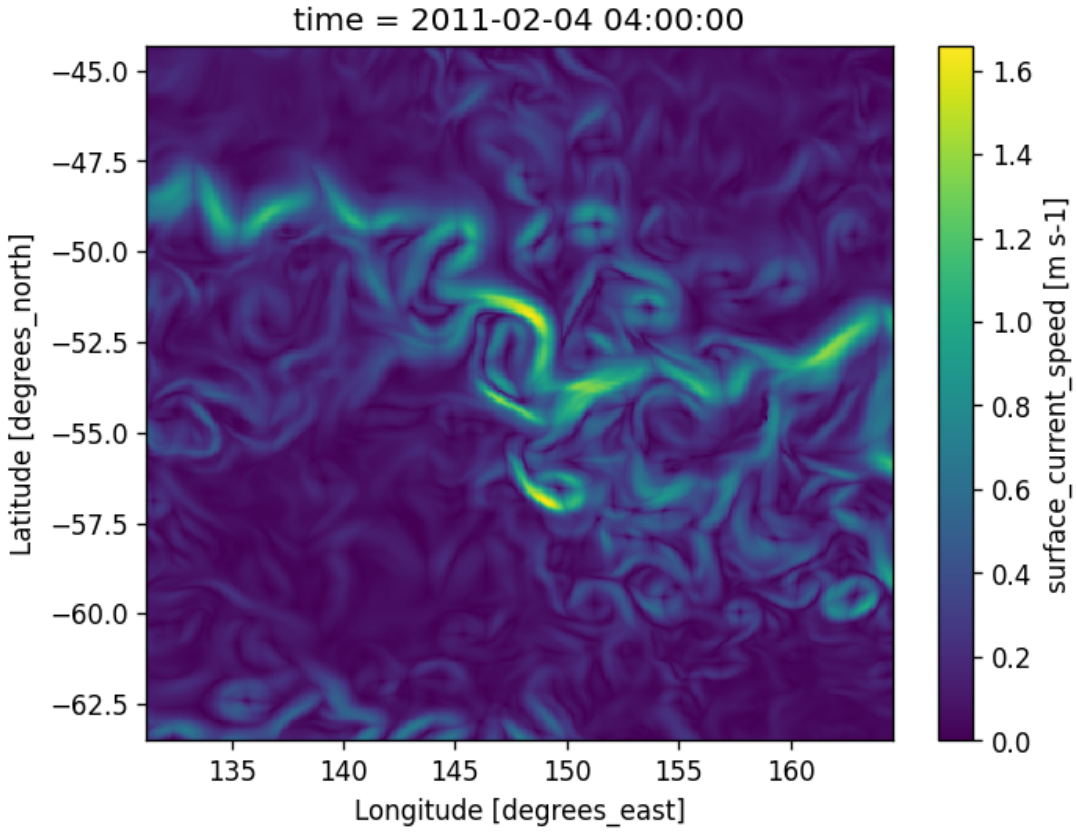

Figure: Surface current speed from simulation data (NEMO ORCA0083-N01), somewhere in the Southern Ocean.

Data handling: images

Figure: Pictures of the other three ad hoc TAs in the course, with special appearance from (in my opinion) the OG AI assistants. These guys will show up throughout the course.

- (64, 64, 1) arrays

  $\rightarrow$ entry denotes how 'light' it is (255 = white, 0 = black, gray in between)

  $\rightarrow$ converted from RGB, i.e. (64, 64, 3) arrays

Data handling: text + sound

Figure: The legend himself.

- text and sounds can be digitised and manipulated also (e.g., Shazam)
  $\rightarrow$ not to much on text here (don't touch on LLMs; some in RNNs maybe)

Machine learning + regression

Recall regression is that, for X the input, Y the output, f the model, we want

$$y = f(X)$$

- prediction or forward problem: have X and f, want Y
- inference or inverse problem: have Y and f, want X
- inference/regression/Machine Learning: have X and Y, want f

Q. really want the "best" f, but how to define "best", and then obtain "best" f?

Linear regression as an optimisation problem

Recall linear regression takes the model as

$$\hat{Y} = aX + b$$

where $\hat{Y}$ is the prediction

- idea: given a measure of error $J = J(\hat{Y}, Y)$, find a and b such that J is minimised

  $\rightarrow$ i.e. we have an optimisation problem

Linear regression as an optimisation problem

Recall linear regression takes the model as

$$\hat{Y} = aX + b$$

where $\hat{Y}$ is the prediction

- idea: given a measure of error $J = J(\hat{Y}, Y)$, find a and b such that J is minimised

  $\rightarrow$ i.e. we have an optimisation problem

  $\rightarrow$ J is sometimes called the objective/mismatch/loss function (or functional depending on context)

  $\rightarrow$ a and b would be the model parameters of f, and control variables of the problem

!!! a and b would depend on choice of J

Linear regression as an optimisation problem

- one choice for J is the L^p family of norms

$$\|\hat{Y} - Y\|_{L^p} = \left(\int |\hat{Y} - Y|^p \, \mathrm{d}\mu \right)^{1/p}$$

$\rightarrow$ μ is a measure (but not going to elaborate what that is...)

$\rightarrow$ in practice we almost always deal with with sums instead of integrals...

- the two often encountered cases are L^2 as Mean Squared Error (MSE)

$$\mathrm{MSE} \sim \|\hat{Y} - Y\|_{L^2}^2 \sim \frac{1}{N} \sum_i^N |\hat{Y}_i - Y_i|^2$$

and L^1 as Mean Absolute Error (MAE)

$$\mathrm{MAE} \sim \|\hat{Y} - Y\|_{L^1} \sim \frac{1}{N} \sum_i^N |\hat{Y}_i - Y_i|.$$

Linear regression as an optimisation problem

- the standard linear regression uses L^2 as the loss function
  $\rightarrow$ finds the standard line of best fit (LOBF)
  $\rightarrow$ actually have close form solutions for a and b

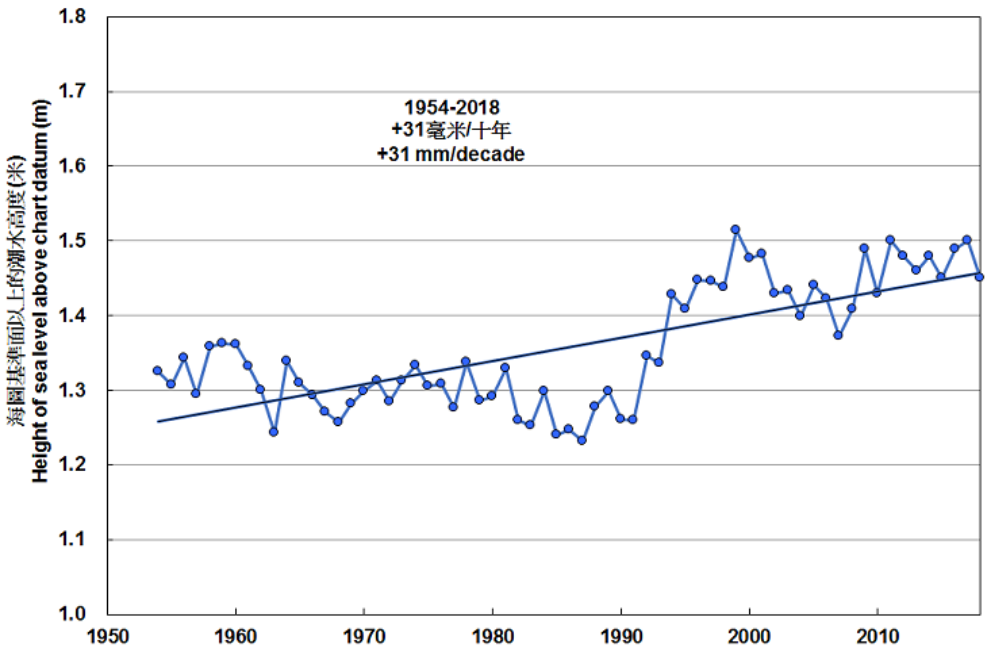

Figure: Figure from HKO.

- can go beyond linear
  $\rightarrow$ e.g. could argue figure on left is problematic...
- can also change choice of loss in principle
  $\rightarrow$ L^1 optimisation reduces effect of outliers

Train/test split + model skill

Good practice to not throw all the data in (overfitting etc.). Split (X, Y) into:

- training data $(X_{\text{train}}, Y_{\text{train}})$ (most data should be here)

  $\rightarrow$ exposed to ML algorithms for training the model

  $\rightarrow$ used to compute loss function
- test data $(X_{\text{test}}, Y_{\text{test}})$

  $\rightarrow$ **NOT** exposed to ML algorithm

  $\rightarrow$ used to test performance of model

Train/test split + model skill

Good practice to not throw all the data in (overfitting etc.). Split (X, Y) into:

- training data $(X_{\text{train}}, Y_{\text{train}})$ (most data should be here)

  $\rightarrow$ exposed to ML algorithms for training the model

  $\rightarrow$ used to compute loss function

- test data $(X_{\text{test}}, Y_{\text{test}})$

  $\rightarrow$ **NOT** exposed to ML algorithm

  $\rightarrow$ used to test performance of model

- sometimes have validation data $(X_{\text{val}}, Y_{\text{val}})$

  $\rightarrow$ subset of training data

  $\rightarrow$ exposed to ML algorithms to tune model hyperparameters and/or model selection

Train/test split

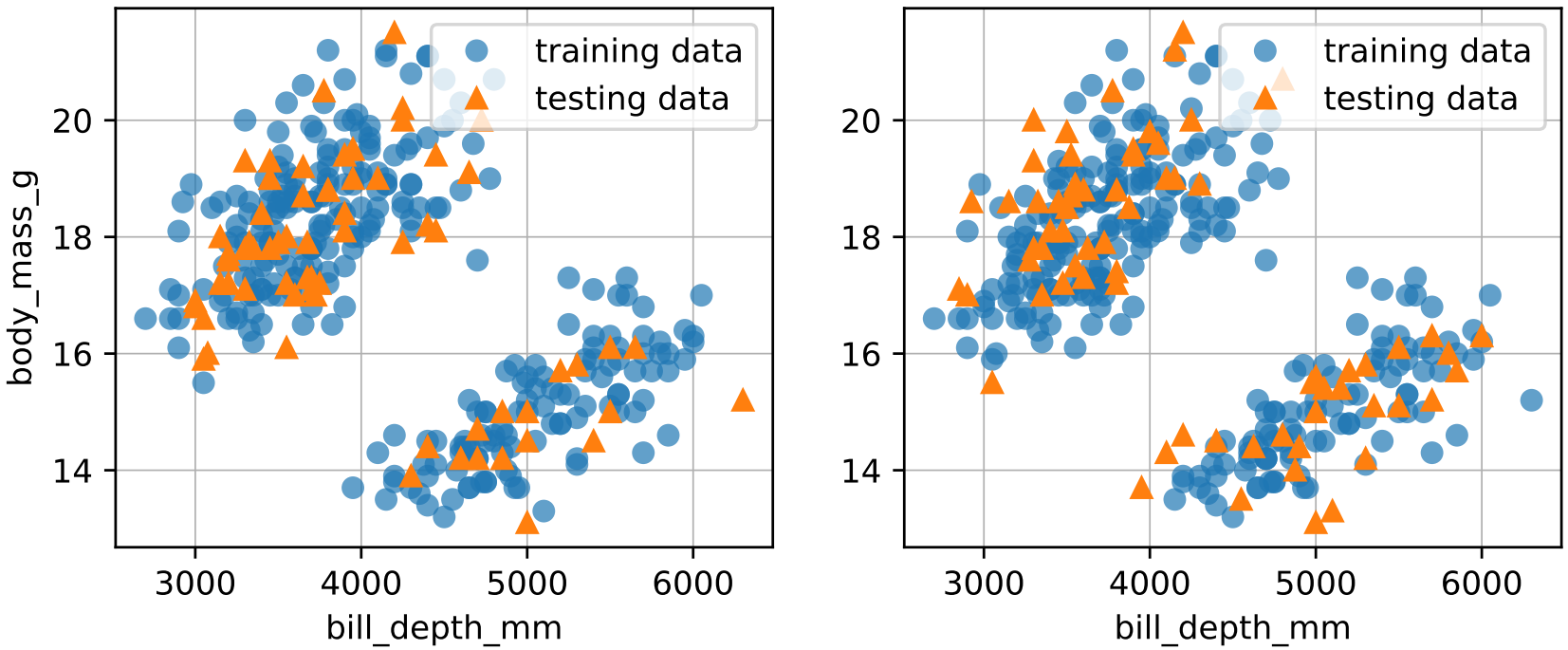

Figure: Sample of train/test split from penguins data.

Train/test split + model skill

- train/test split would be 'random'
  $\rightarrow$ i.e. sampling from a uniform distribution (see later)
  $\rightarrow$ introduces inherent randomness however (also see later)

- how to judge model skill on test data?
  $\rightarrow$ could use the L^p norms again
  $\rightarrow$ other choices, e.g. AIC and BIC that penalises complexity (see OCES 3301, and information/cross entropy later in course)

Ultimately there is a choice in J, and a lot of things boil down to how you choose that

(e.g. different linear models is more or less an exercise in varying J...see Lec 02)

Data scaling

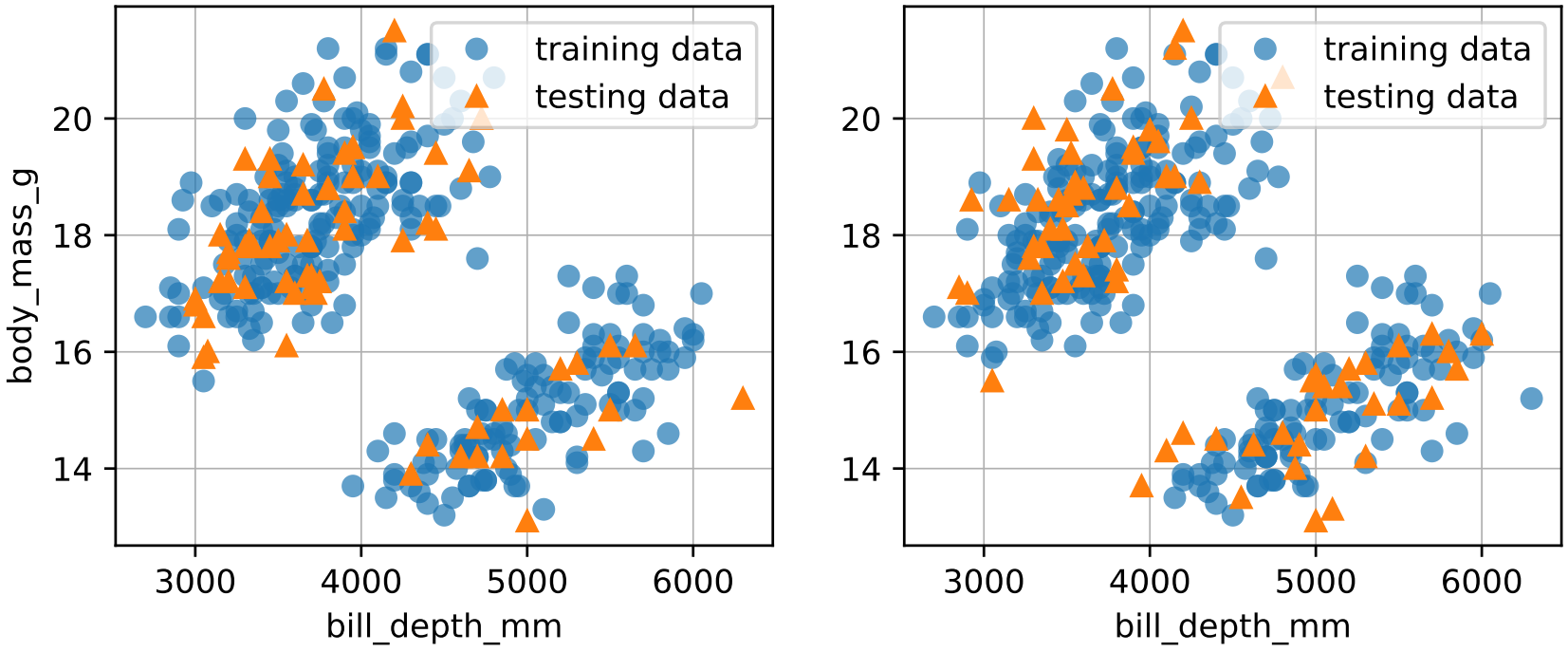

Figure: Sample of train/test split from penguins data.

- two variables here are different (e.g. units)

  $\rightarrow$ what to do about it?

  $\rightarrow$ how to compare different quantities in general? e.g. T and S and contribution to ρ

Recap: basics of probability

- idea: make the data distribution comparable

- recall that we say a random variable (just think data) X follows a Gaussian/normal distribution

$$X \sim \mathcal{N}(\mu, \sigma)$$

if it is described by the probability distribution function (pdf)

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

$\rightarrow$ just the 'bell' curve, with mean μ and variance σ^2

Recap: basics of probability

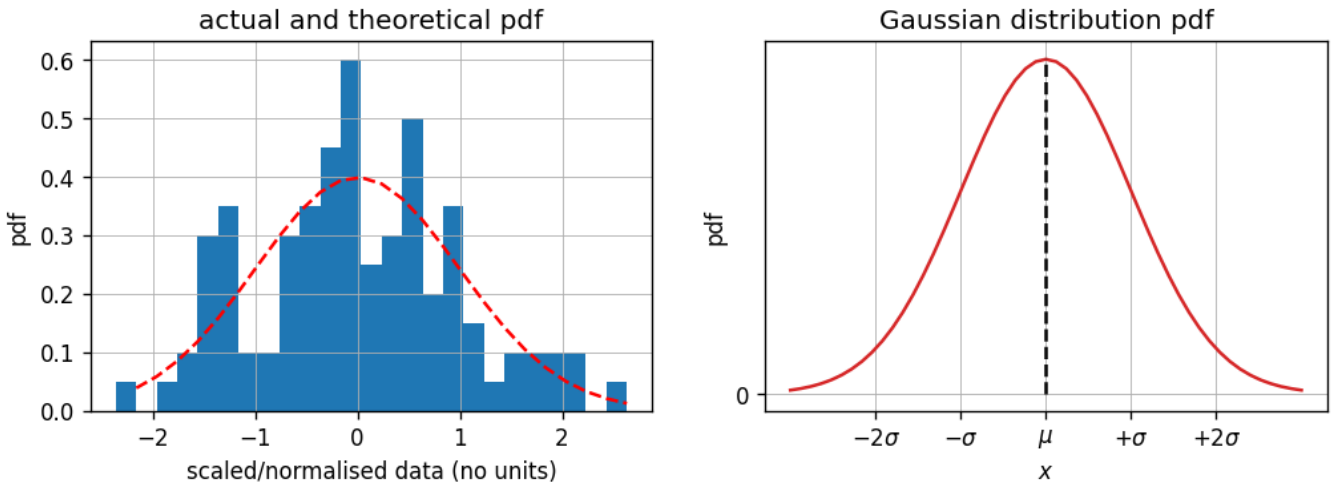

Figure: The Gaussian pdf (with units deliberately omitted). Obtain probability from an integral.

- CLT tells you with enough samples most distributions follow a Gaussian
- **68-95-99.7 rule**, 68, 95 and 99.7% of the data lies within 1, 2 and 3 std of the mean

Data scaling

- then for data $X \sim \mathcal{N}(\mu_X, \sigma_X)$ and $Y \sim \mathcal{N}(\mu_Y, \sigma_Y)$, just do Z-score standardisation (cf. OCES 3301 lec 05):

$$\tilde{X} = \frac{X - \mu_X}{\sigma_X}, \qquad \tilde{Y} = \frac{Y - \mu_Y}{\sigma_Y},$$

then $\tilde{X}$ and $\tilde{Y}$ both follow $\mathcal{N}(0, 1)$

Data scaling

- then for data $X \sim \mathcal{N}(\mu_X, \sigma_X)$ and $Y \sim \mathcal{N}(\mu_Y, \sigma_Y)$, just do Z-score standardisation (cf. OCES 3301 lec 05):

$$\tilde{X} = \frac{X - \mu_X}{\sigma_X}, \qquad \tilde{Y} = \frac{Y - \mu_Y}{\sigma_Y},$$

then $\tilde{X}$ and $\tilde{Y}$ both follow $\mathcal{N}(0, 1)$

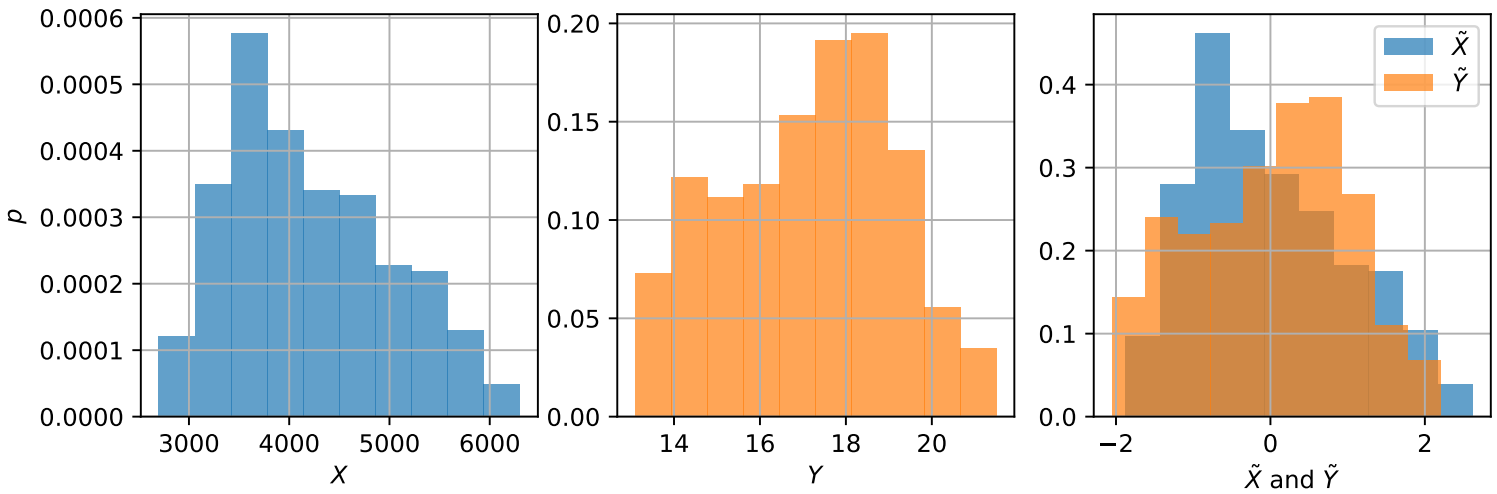

Figure: Raw and re-scaled pdfs as histograms.

Data scaling

- not the only way to do it, and not always the best way of doing it

  $\rightarrow$ e.g. not all data follows Gaussian distribution

  $\rightarrow$ e.g. sometimes there are principles to guide non-dimensionalisation

  $\rightarrow$ e.g. min/max $\rightarrow$ [0, 1] scaling for example in images

- if we don't have further information, could try this first

- aside: train/test split splits usually done via some uniform distribution, but can also do things differently

Q: model robustness

- since we are talking about probability, there is inherent randomness

  → resulting models have some randomness also

  → how do we know our model 'works' not just because we got 'lucky'?

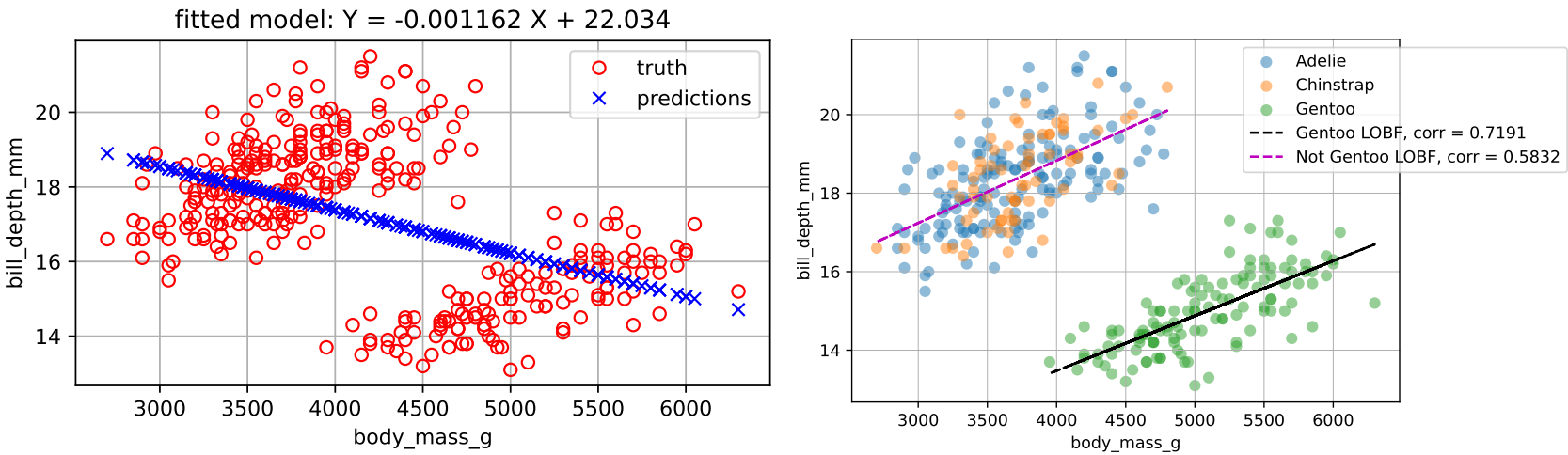

Figure: Contrived example of completely different models depending on data selection.

Demonstration

- GitHub repository
- Jupyter notebooks
- Python (+ Anaconda and/or Google Colab)
- xarray, pandas etc.
- data manipulation, plotting etc.
- `sklearn`

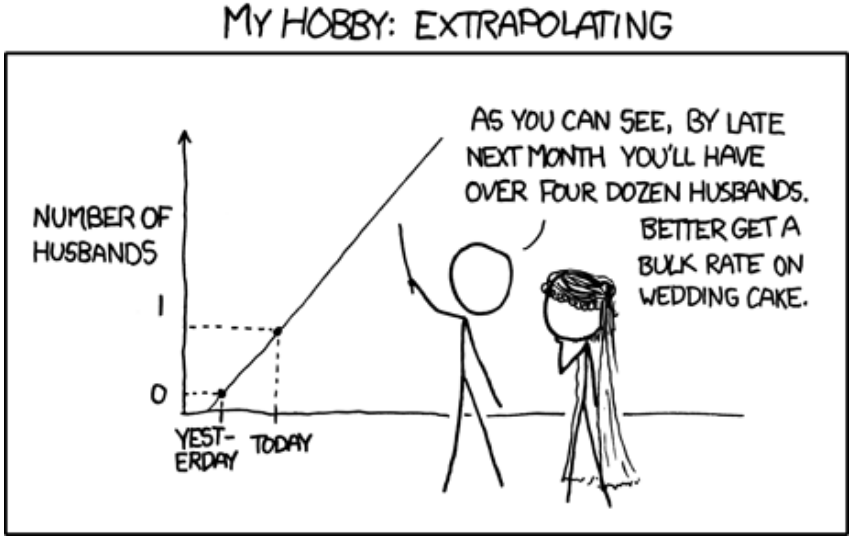

Figure: How not to do extrapolation. From XKCD.

what you wouldn't do with statistics
you should not do with machine learning

Moving to a Jupyter notebook →