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 3: classification, random forests
and (gradient) boosting

Outline

- classification

  → predicting **discrete** labels (cf. **continuous** target in regression)

  → TL;DR: finding a separators between data

- random forests and/or (gradient) boosting

  → TL;DR: **ensemble** of decision trees

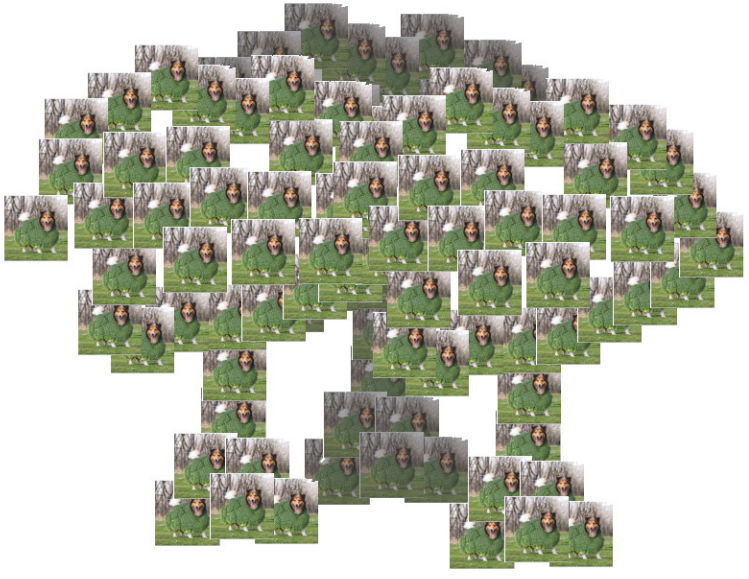


Figure: A forest of broccollies.

Oceanic application

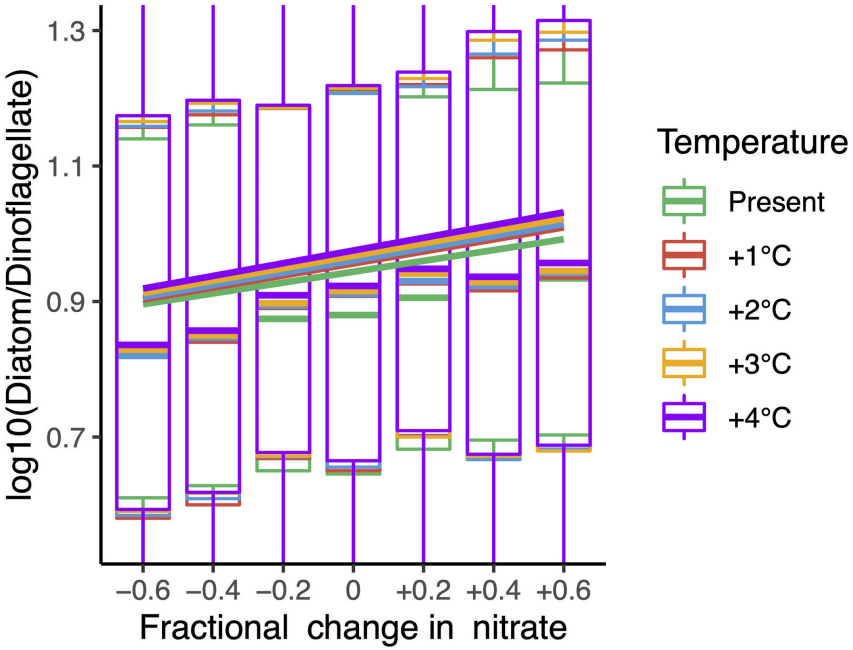


Figure: Predicted log of diatom-dinoflagellate ratio in HK coastal waters using random forests (which is a collection of decision trees) under projected warming scenarios. From Fig. 4 of Cheung *et al.* (2021).

- marine pollution

  → used station data to train up a regression model

  → focused on diatom-dinoflagellate ratio

  → identified key features of importance

  → can do projection (!!!) and find pollution increases under warming

Q. classfier for identifying degrees of toxicity?

(see also Lee *et al.* (2025) from OCES)

Oceanic application

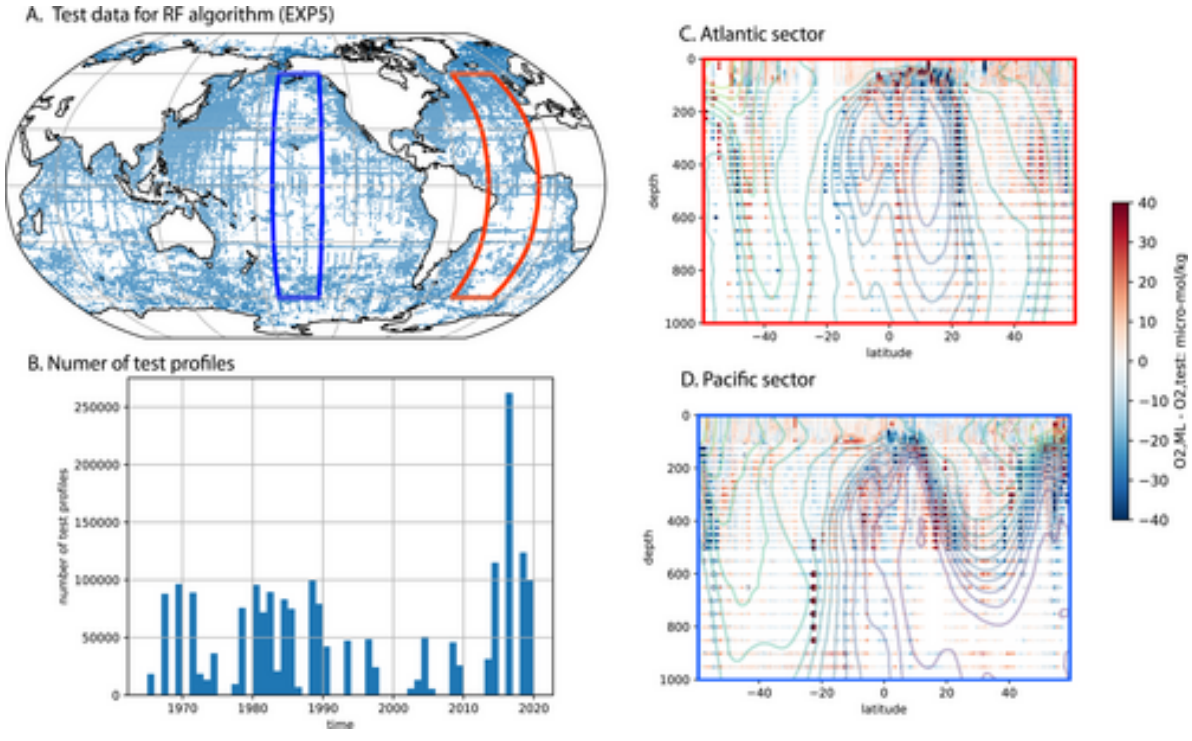


Figure: Sample data of (sparse and incomplete) observed oxygen content, where we want the algorithms to interpolate the data. From Fig. 8 of Ito *et al.* (2024).

- regressor for sparse oxygen observations (also has neural networks here)
  → used ship and argo data
  → identifies **deoxygenation** trends
  → identifies key variabls (cf. their Fig. 11, but how? see later)

Example: predicting label from data

- suppose I have the following data:

bill length (mm)	55.0
bill depth (mm)	20.0
flipper length (mm)	207.0
body mass (g)	4000.0

Q. what is the associated species of penguin?

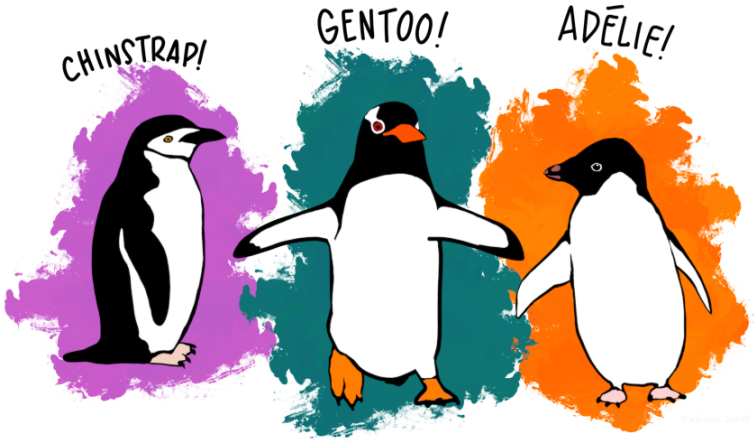


Figure: Palmer penguins logo. From `https://allisonhorst.github.io/palmerpenguins/`

Example: image classification

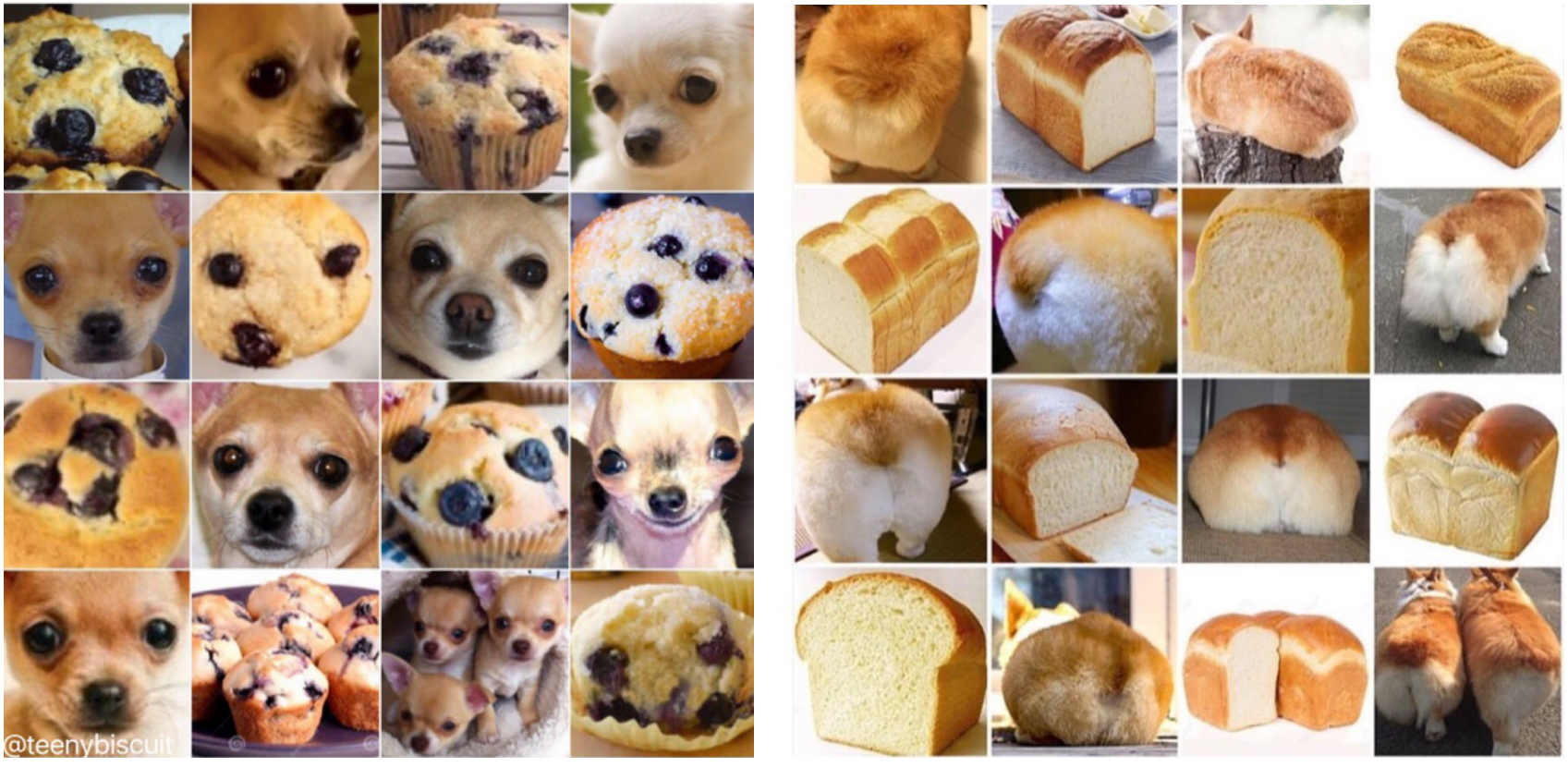


Figure: Various entries from the "animal or things" meme, as found on the internet.

Classification as a geometry problem

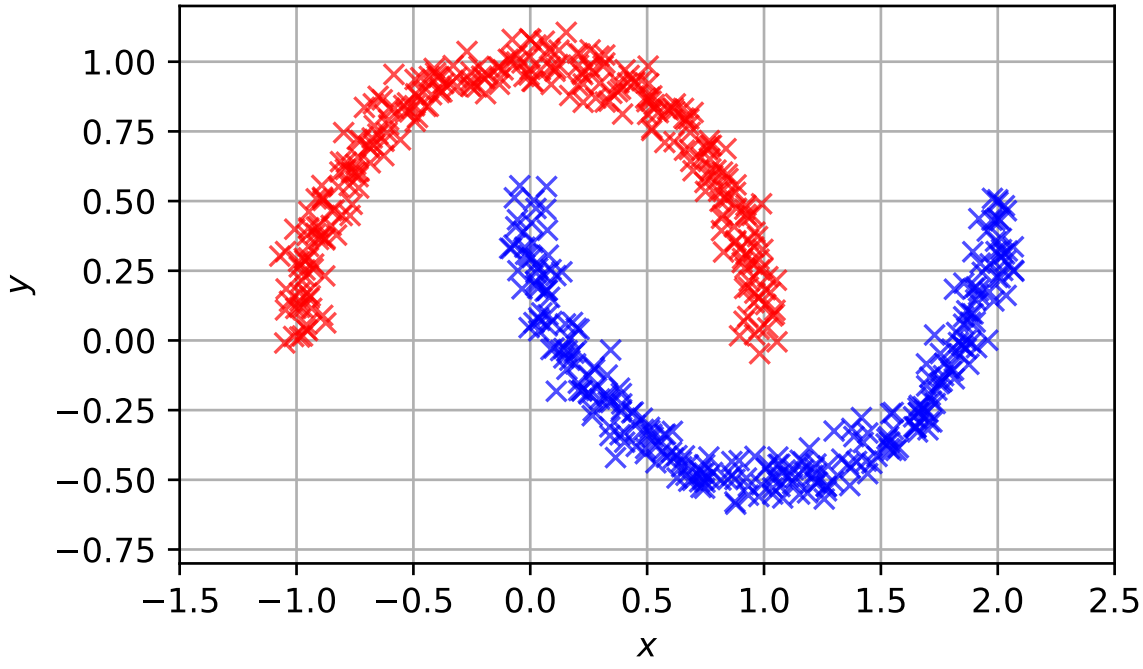


Figure: Moon data coloured by the cluster it is in.

- moon data as an example
- task: find the line/curve that separates the data clusters

  → a separator (there are many of these)

  → higher dimension analog: (hyper-)planes/surfaces

L/QDA

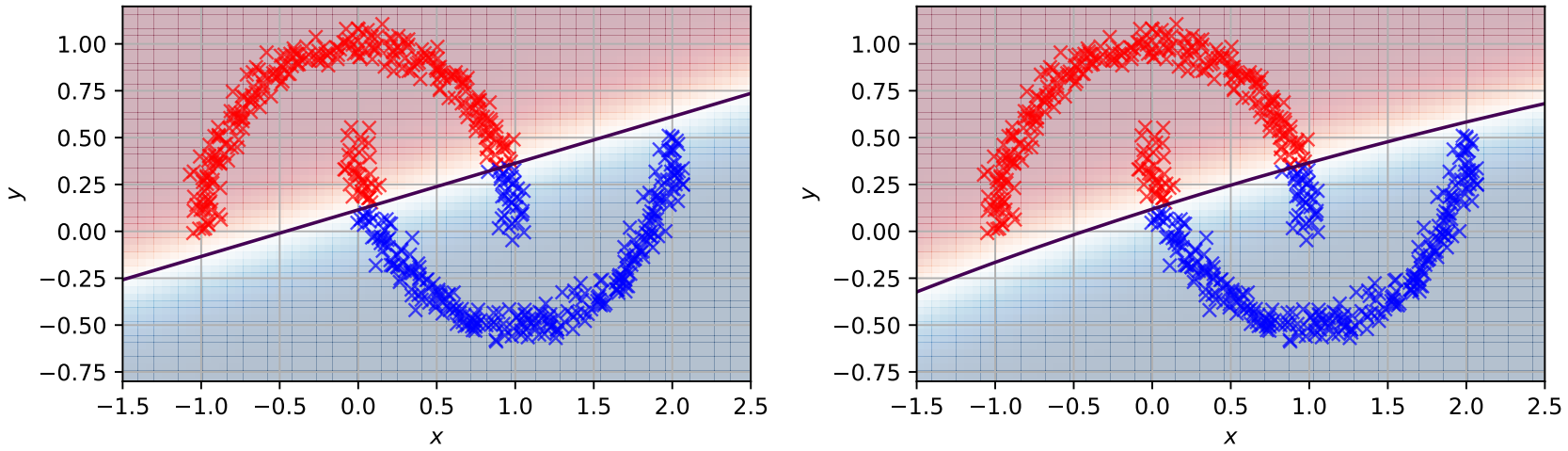


Figure: Classification by a realisation of (left) LDA and (right) QDA, with decision boundaries and probabilities.

- Linear/Quadratic Discriminant Analysis (L/QDA)
  → finds a linear/quadratic separator
  → closed form solutions, quite robust
  → an approach related to statistics, has probability measures associated with this (the shading)

L/QDA

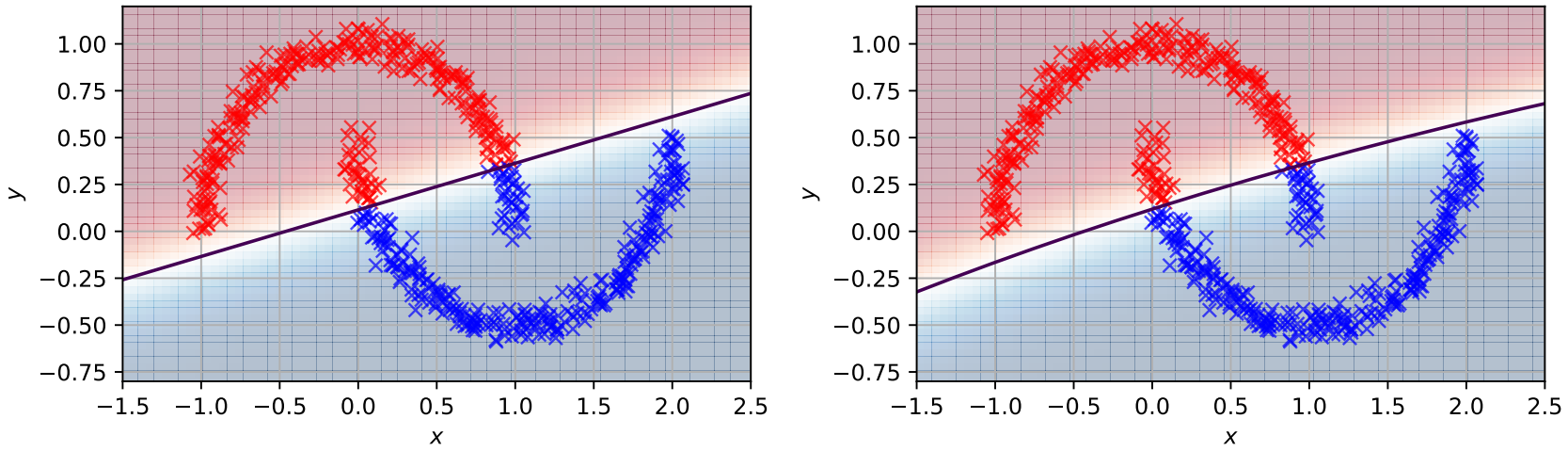


Figure: Classification by a realisation of (left) LDA and (right) QDA, with decision boundaries and probabilities.

- Linear/Quadratic Discriminant Analysis (L/QDA)
  → finds a linear/quadratic separator
  → closed form solutions, quite robust
  → an approach related to statistics, has probability measures associated with this (the shading)

Q. more complex curve needed?

SVM

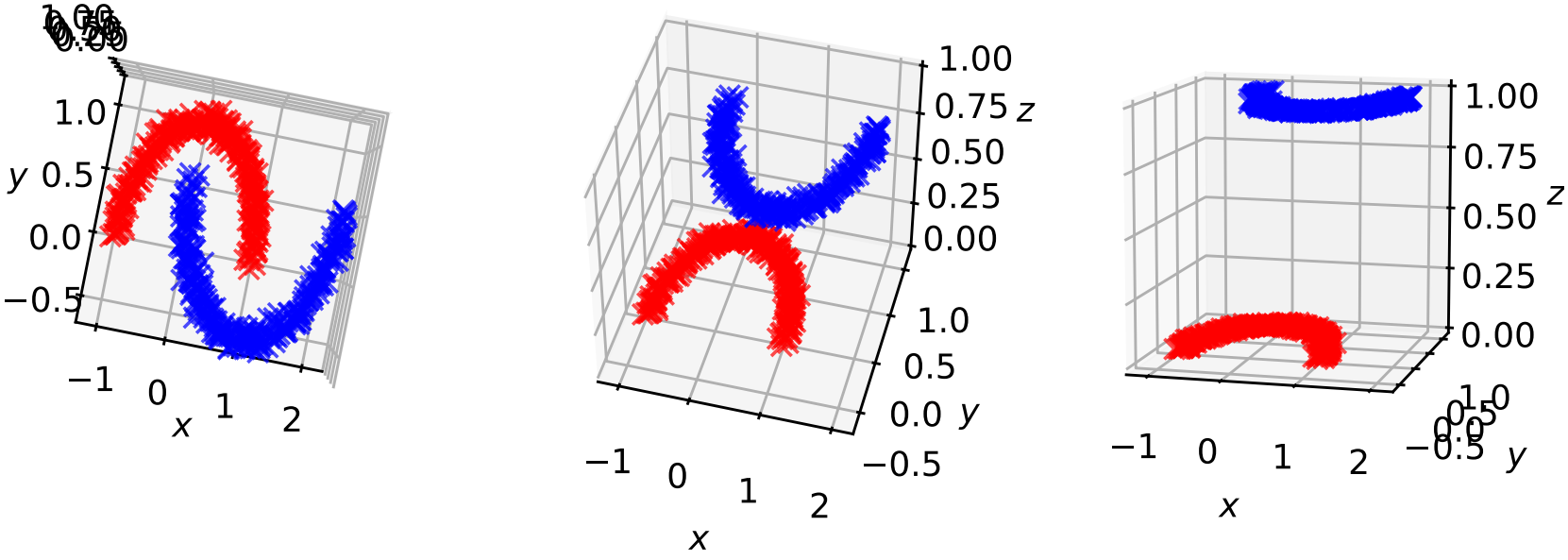


Figure: Moon data where I artificially 'lift' one of the clusters.

- instead of more complex curve, what about simpler separator but 'lift' the data instead to a higher dimension?

  $\rightarrow$ basic geometric idea of Support Vector Machine (SVM)

SVM

- with enough complexity the data can almost always be separable

  $\rightarrow$ then easy to find separating hyper-plane

  $\rightarrow$ can arbitrarily find complicated dimension promotions though?

  $\rightarrow$ mapping can be expensive computationally?

SVM

- with enough complexity the data can almost always be separable

  $\rightarrow$ then easy to find separating hyper-plane

  $\rightarrow$ can arbitrarily find complicated dimension promotions though?

  $\rightarrow$ mapping can be expensive computationally?

- idea: find **maximal** hyper-plane separator given a remapping into higher-dimensions, but **penalise** the complexity of that transformation

  $\rightarrow$ an optimisation problem

(actually uses the kernel trick and solving the dual problem; not going to elaborate on those)

SVM

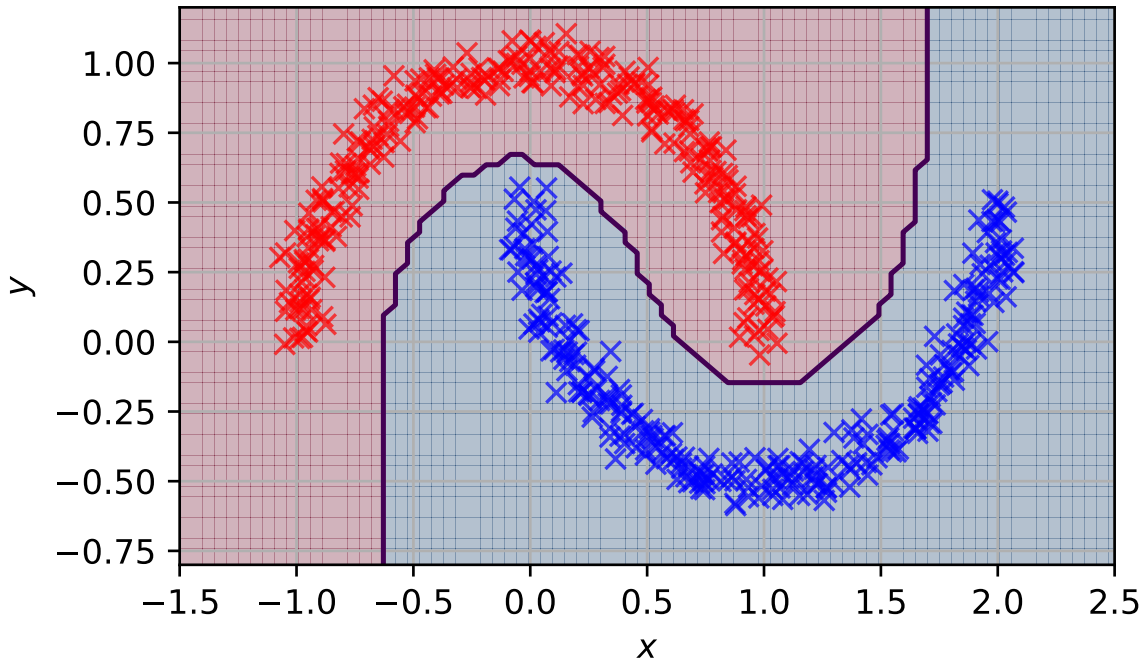


Figure: Moon data using a SVM with the radial basis function as the kernel.

- roughly: transform data, find hyper-plane, undo transformation

  $\rightarrow$ no probability measure here for decision

SVM

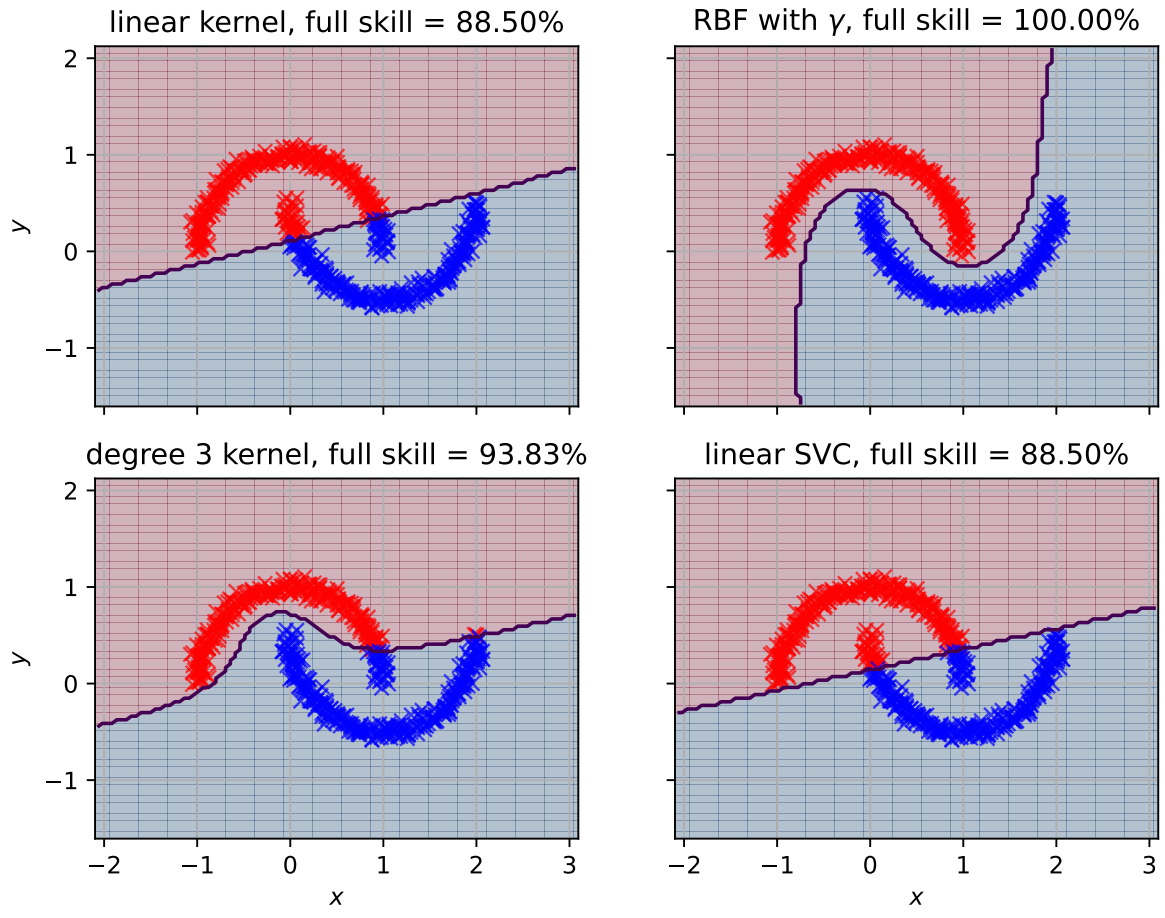


Figure: SVM on moon data with different choices of kernels and regularisations.

Issues with SVM: dependence on scaling

- SVM not invariant to re-scaling
  $\rightarrow$ highly recommended to scale/standardise data

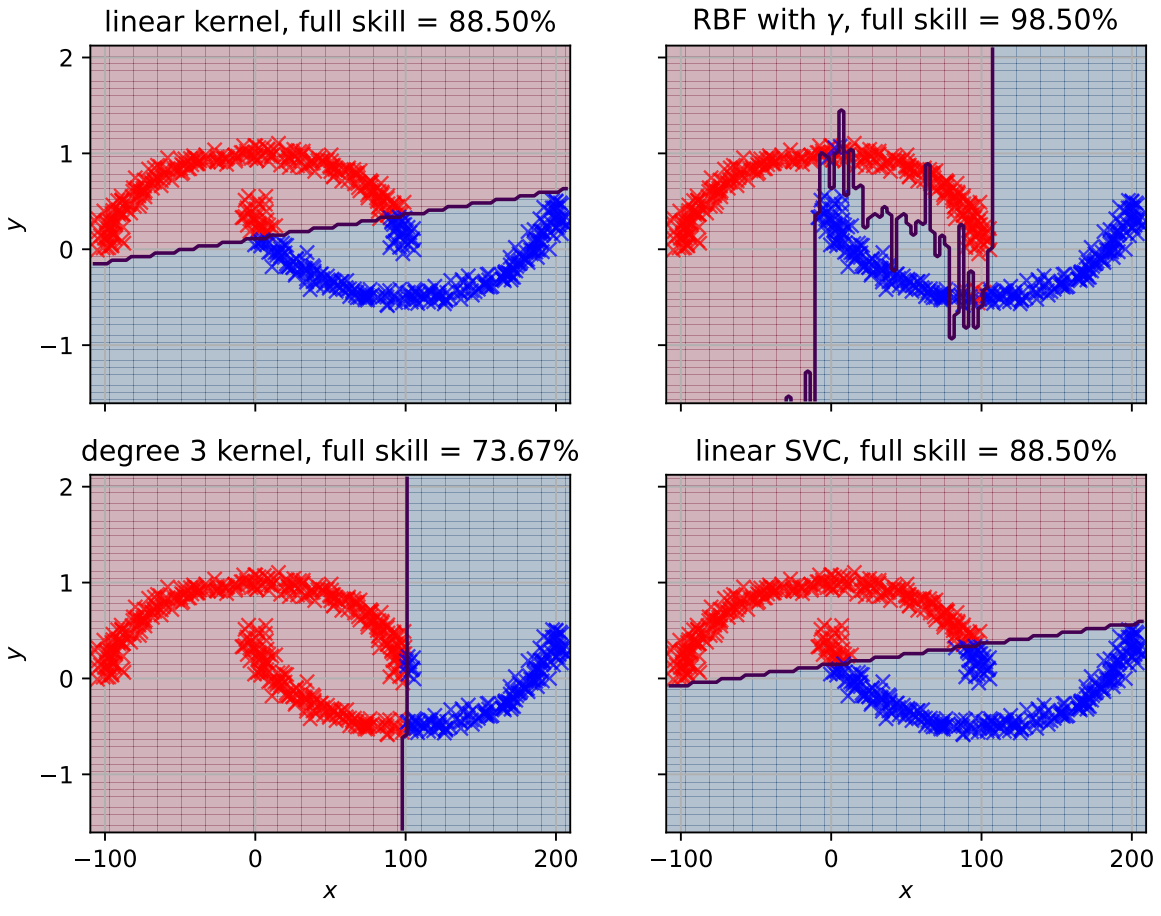


Figure: SVM on stretched moon data with different choices of kernels and regularisations.

Issues with SVM: unbalanced data

- SVM has dependence on sample size, biased towards the larger class

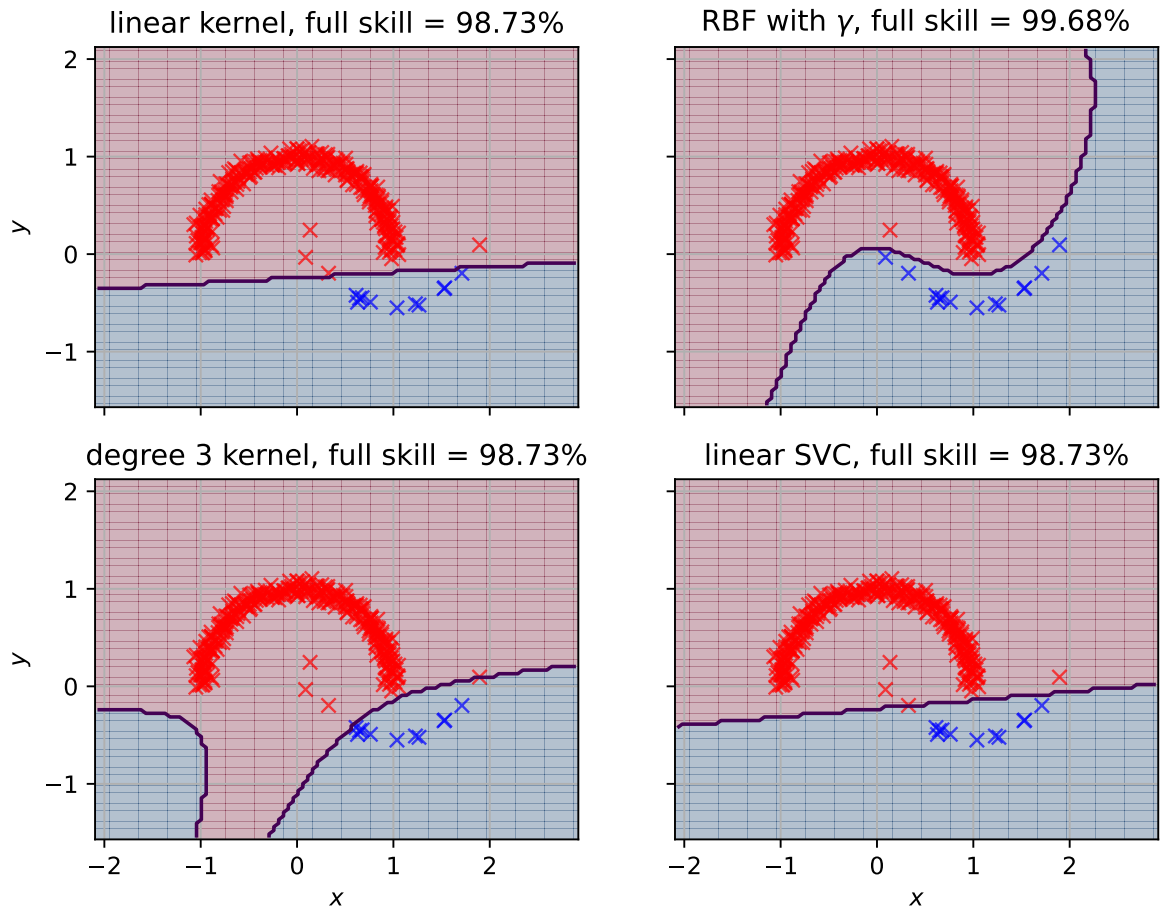


Figure: SVM on unbalanced moon data with different choices of kernels and regularisations.

Issues with SVM: unbalanced data with re-weighting

- one way is to change the weighting on the data

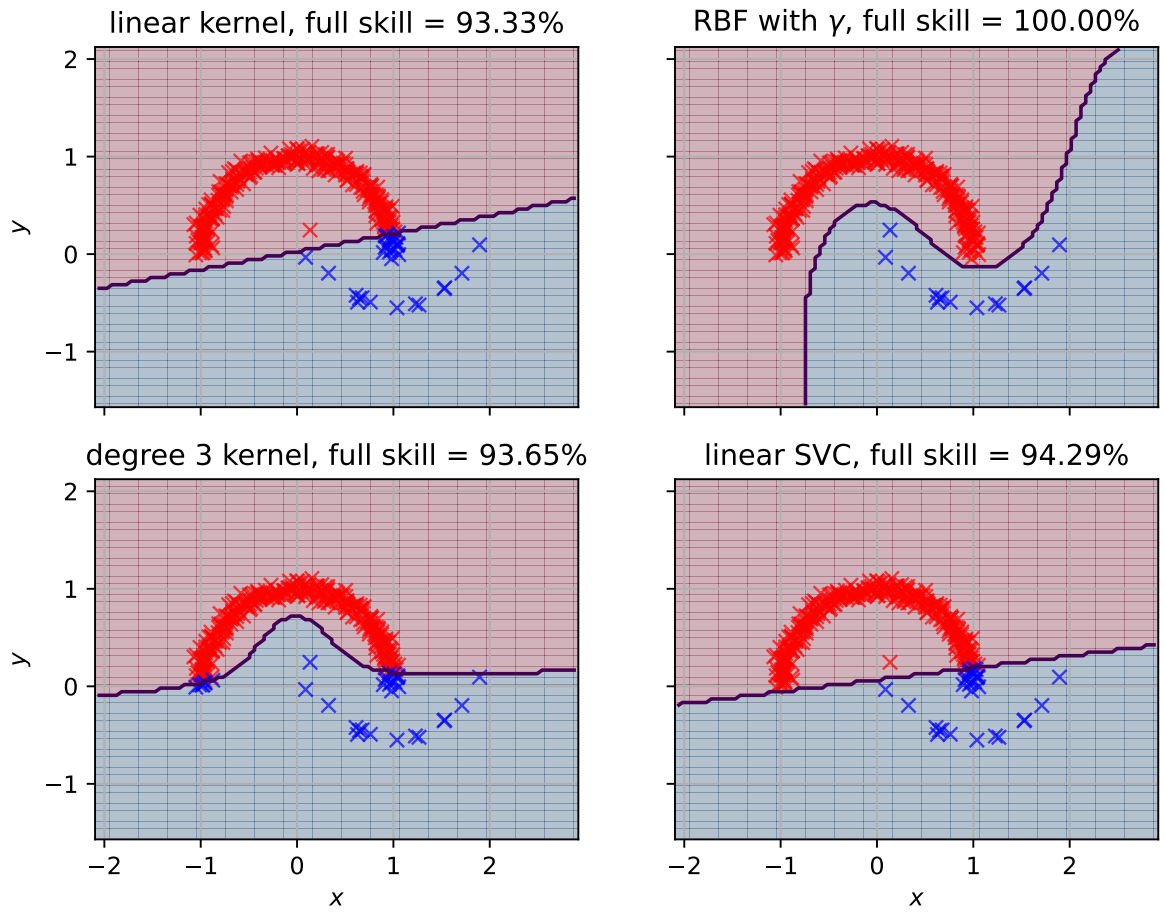


Figure: SVM on unbalanced moon data but with weight rebalancing, with different choices of kernels and regularisations.

Digression: Stochastic Gradient Descent (SGD)

(Relates later to Neural Networks)

- recall loss function J as function of model parameters θ traces out some "landscape" (a hyper-surface), and we want to find the global minimum

  $\rightarrow$ local minimums can exist

Digression: Stochastic Gradient Descent (SGD)

(Relates later to Neural Networks)

- recall loss function J as function of model parameters θ traces out some "landscape" (a hyper-surface), and we want to find the global minimum

  $\rightarrow$ local minimums can exist

- gradient descent

  $\rightarrow$ compute **full** gradient, find fastest descent

  $\rightarrow$ can be costly, can get stuck in local minimum

Digression: Stochastic Gradient Descent (SGD)

(Relates later to Neural Networks)

- recall loss function J as function of model parameters θ traces out some "landscape" (a hyper-surface), and we want to find the global minimum

  $\rightarrow$ local minimums can exist

- gradient descent

  $\rightarrow$ compute **full** gradient, find fastest descent

  $\rightarrow$ can be costly, can get stuck in local minimum

- **stochastic** gradient descent

  $\rightarrow$ **approximates** full gradient by evaluating by computing gradient in only one or a few directions

  $\rightarrow$ take longer to converge, potentially cheaper, could get 'kicked' out of local minimum (that may be a good thing)

Classification as an optimisation problem

- want one-sided loss functions to punish wrong classifications

  $\rightarrow$ samples below; see notebook for description

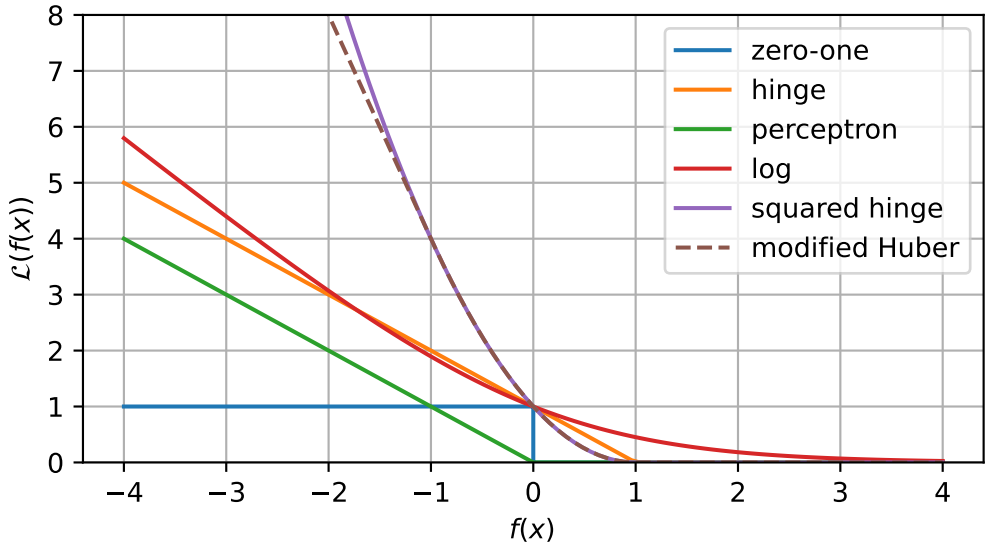


Figure: Selection of loss functions for classification.

- probably want a penalisation too (e.g. L^2, L^1, elastic net)

Special cases

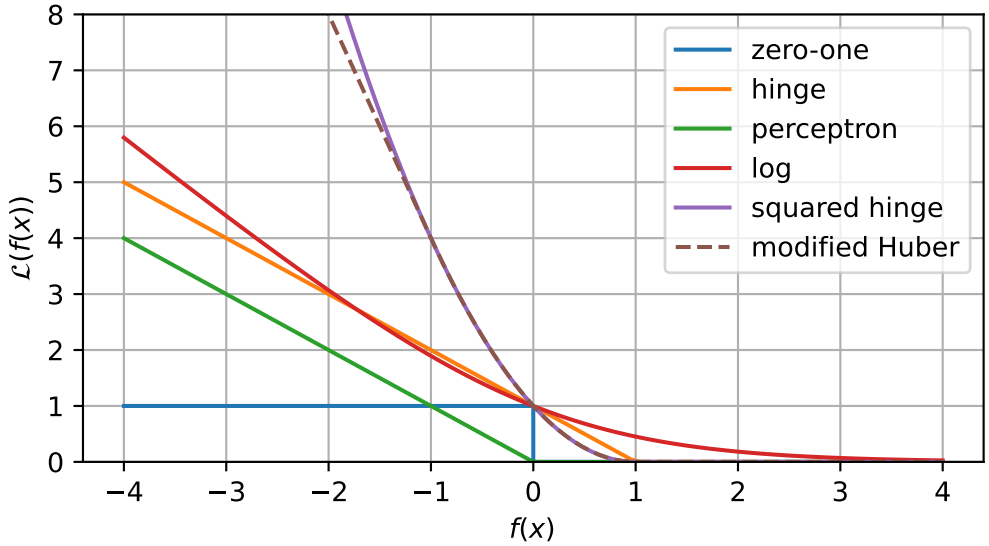


Figure: Selection of loss functions for classification.

- hinge loss + L^2 penalisation $\Rightarrow$ linear SVM
- log loss $\Rightarrow$ logistic regression (a.k.a. logit model, or Maximum Entropy model (not going to elaborate on why)
  $\rightarrow$ it's called 'regression' but it's really classification

- Note: flipping some of these around gives candidates for activation functions (an important component of Networks)

Logistic Regression

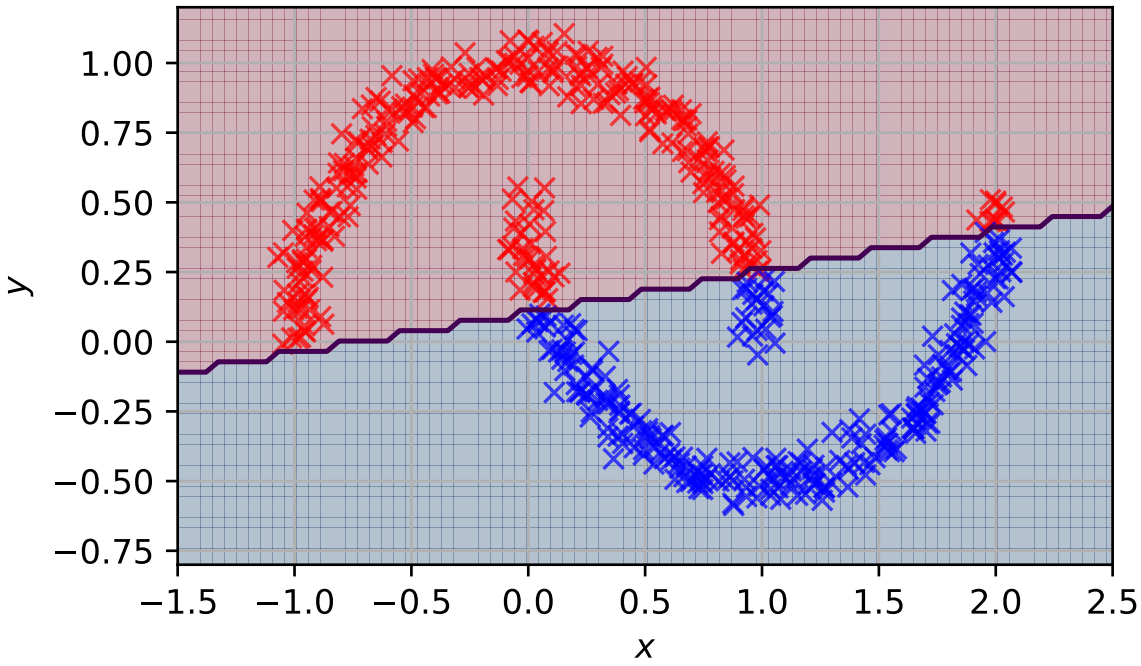


Figure: Logistic regression on moon data with default settings.

- above one is done with `SGDClassifier`

  → more options through `LogisticRegression`

Demonstration: penguins data

- SVM (with rbf) kernel on penguins data

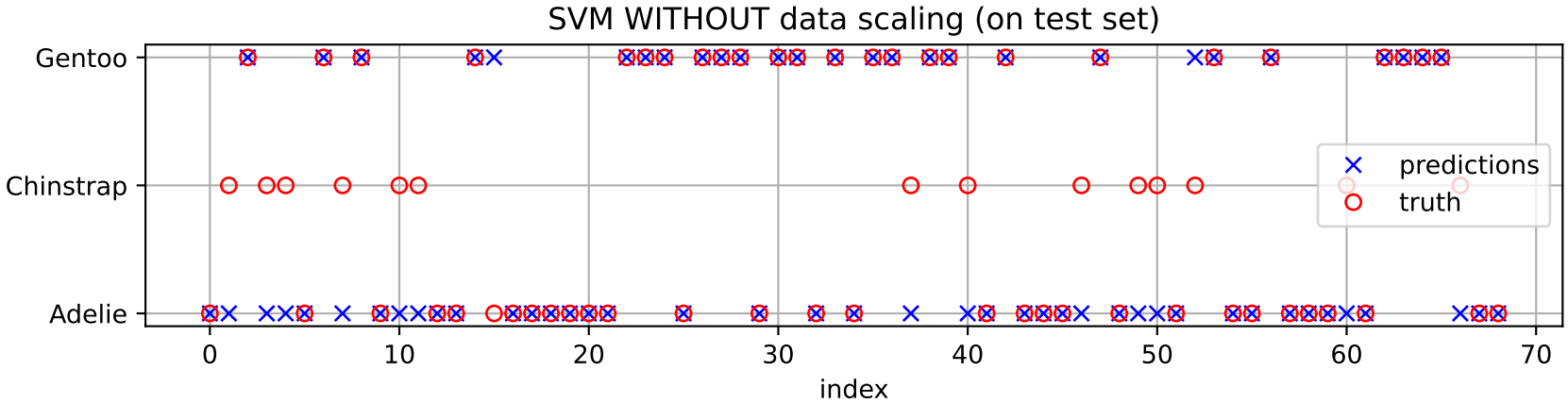
  $\rightarrow$ convert `species` labels to numerical values

  $\rightarrow$ do this wrong first by not scaling the data, using two features (chose the worse combo deliberately)

  $\rightarrow$ scale data, still using two features

  $\rightarrow$ scale data, using all four features

Demonstration: penguins data



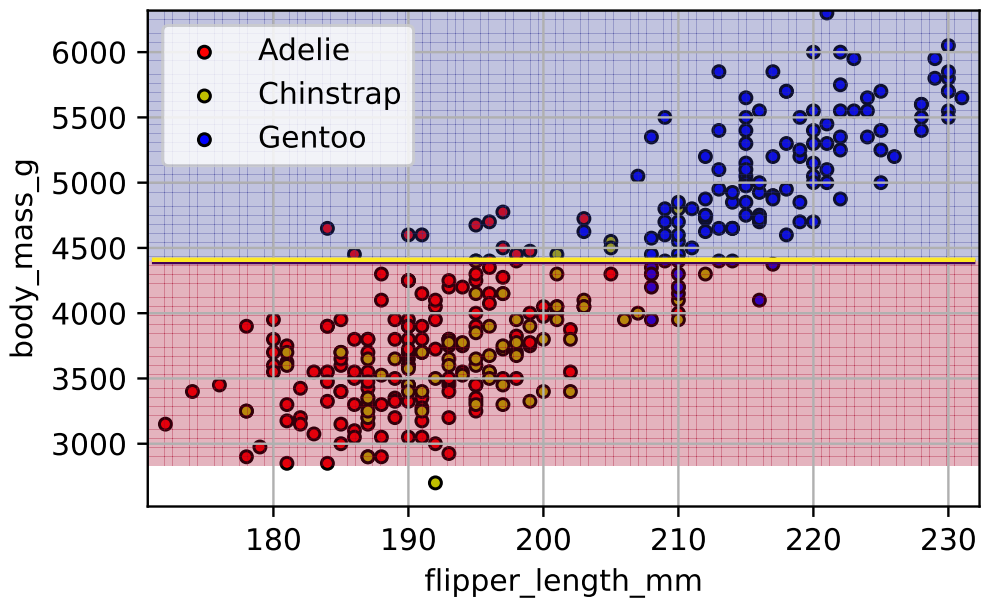


Figure: SVM (with rbf kernel) on **unscaled** penguins data using two features.

Demonstration: penguins data

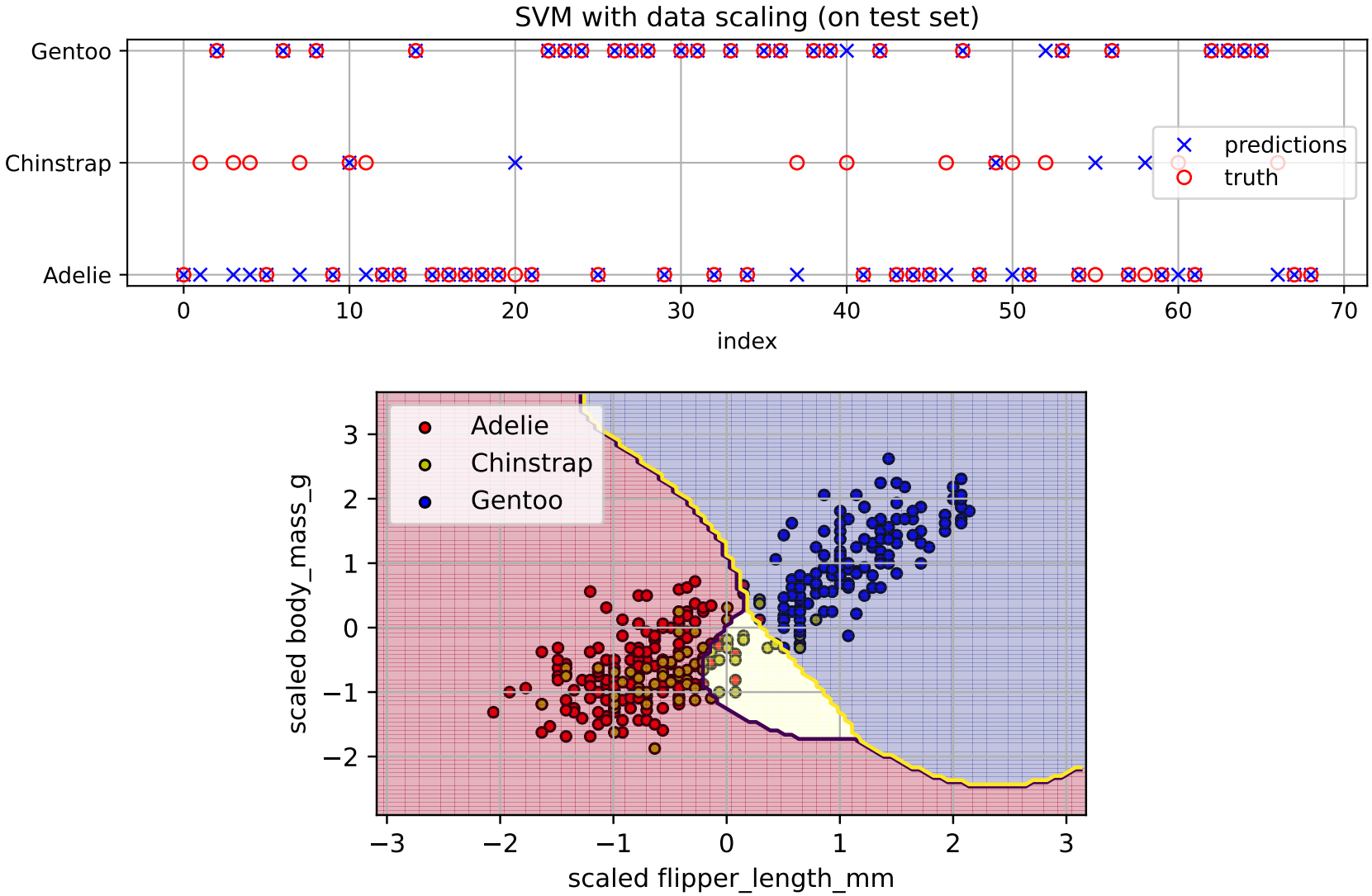


Figure: SVM (with rbf kernel) on scaled penguins data using two features.

Demonstration: penguins data

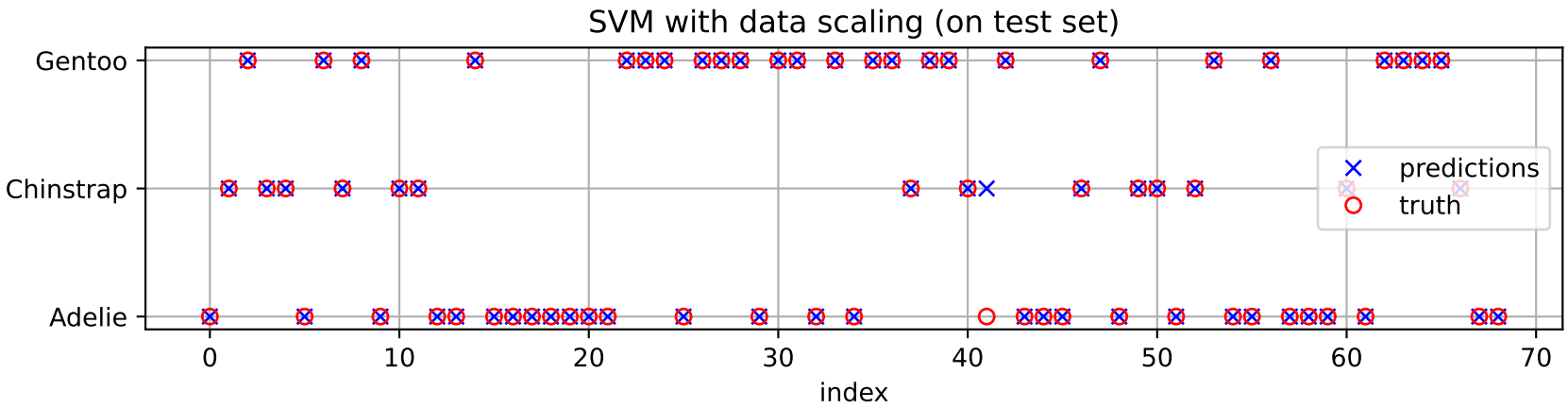


Figure: SVM (with rbf kernel) on scaled penguins data using all features.

- can't show decision boundary here in an obvious way because it lives in 4d

Decision trees: Anatomy

Decision trees: Anatomy

- **root**: head node
- **leaves**: terminal nodes
- **nodes**: the boxes
- **branches**: connectors of nodes with decision

  $\rightarrow$ strictly speaking the 'yes' and 'no' should be branches
- **levels/depth**: how far you are from root

  $\rightarrow$ here it could be tree of depth 1 or 2
- **parent/child**: to talk about nodes, parent is one level closer to root

My Cat's Decision-Making Tree.

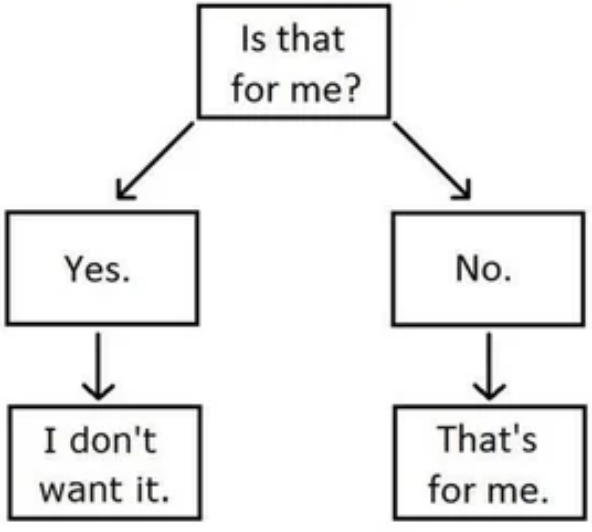


Figure: Example of a decision tree.

Quick (!) recap: probability

- main goal: how do we decide on the criterion to split data (i.e. branching)?

  $\rightarrow$ want the data to tell us, rather than us doing it manually

Quick (!) recap: probability

- main goal: how do we decide on the criterion to split data (i.e. branching)?

  $\rightarrow$ want the data to tell us, rather than us doing it manually

- need some concepts in probability, do this through examples

  $\rightarrow$ e.g. for a **fair** 6-side dice, the possible events are

  $$X = \{1, 2, 3, 4, 5, 6\},$$

  and assigned to these events are probabilities $p_i \in [0, 1]$

  $\rightarrow$ all probabilities should sum to 1, i.e. $\sum_i p_i = 1$

  $\rightarrow$ **fair** $\Rightarrow$ uniform distribution $\Rightarrow p_i = 1/N = 1/6$

Quick (!) recap: probability

- there is a measure called the information entropy defined as

$$H = \sum_i H_i = -\sum_i p_i \log_a p_i$$

$\rightarrow$ called Shannon entropy/index (from ecology)

$\rightarrow$ base a is usually 2, e or 10; the value itself doesn't actually matter too much (I will take $a = \mathrm{e}$ so $\log = \ln$)

Quick (!) recap: probability

- there is a measure called the information entropy defined as

$$H = \sum_i H_i = -\sum_i p_i \log_a p_i$$

  $\rightarrow$ called Shannon entropy/index (from ecology)

  $\rightarrow$ base a is usually 2, e or 10; the value itself doesn't actually matter too much (I will take $a = \mathrm{e}$ so $\log = \ln$)

- measure of species diversity (in ecology); think of it as measure of **surprise**, e.g.

  $\rightarrow$ if $X = \{1, 1, 1, 1, 1, 1\}$, then there is no 'surprise' on what you should draw ($H = 0$)

  $\rightarrow$ if $X = \{1, 2, 3, 4, 5, 6\}$, events are maximally different and maximally 'surprising', ($H = \log N = \log 6$ here)

Quick (!) recap: probability

- from that, the information gain is defined as

$$\mathrm{IG} = H_p - \sum_{i=0}^{N} p_{c,i} H_{c,i}$$

$\rightarrow$ think entropy of parent class minus the weighted averages of entropy of child class

Quick (!) recap: probability

- from that, the information gain is defined as

$$\mathrm{IG} = H_p - \sum_{i=0}^{N} p_{c,i} H_{c,i}$$

  $\rightarrow$ think entropy of parent class minus the weighted averages of entropy of child class
- e.g., if I have cats and dogs of various colours, hair length...

  $\rightarrow$ my parent class could be 'cats' + 'dogs' (2 classes)

  $\rightarrow$ if I segment on **three** 'colours', then I want
  - probabilities of the resulting **six** classes (probably)
  - entropy of that

Quick (!) recap: probability

$$\mathrm{IG} = H_p - \sum_{i=0}^{N} p_{c,i} H_{c,i}$$

- positive values mean entropy of data at the next level has **decreased**

  $\rightarrow$ data is getting more 'pure'

  $\rightarrow$ if proceeding to $H = 0$ then we have maximum purity with no uncertainty

  $\rightarrow$ no uncertainty = max information, and information **gain**

- want to split according to maximum information gain

(or maximise entropy decrease)

Quick (!) recap: probability

- Gini index measure a similar thing to entropy, and is defined as

$$G = \sum_i G_i = \sum_i p_i(1 - p_i)$$

  $\rightarrow$ single species gives $G = 0$, maximally pure

- can do a similar thing to information gain to minimise the Gini index with splitting

  $\rightarrow$ used more in decision trees (computationally faster)

  $\rightarrow$ entropy measures used more in things like neural networks

Demonstration: penguins data

- consider **classification** problem first
  $\rightarrow$ predict species from other features

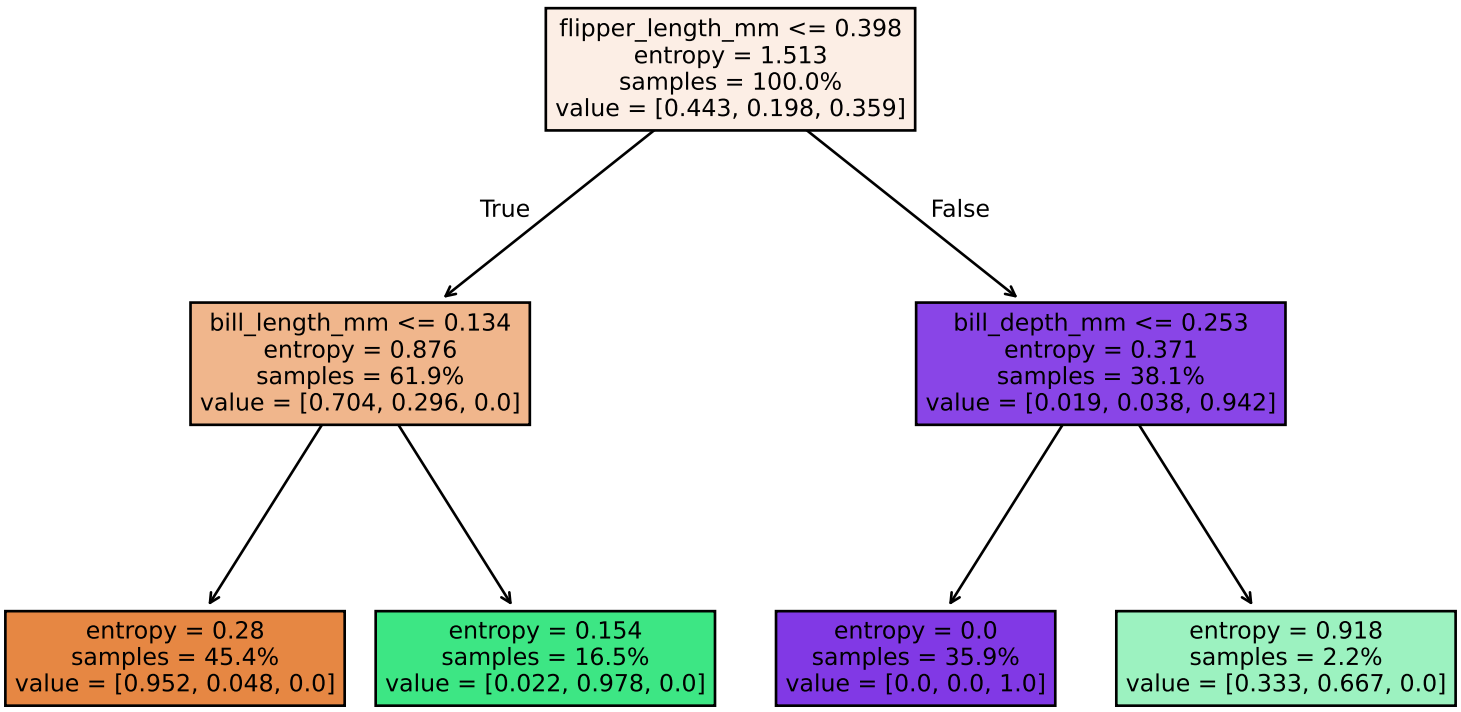


Figure: Tree classifier for penguins data. Maximum depth 2 and uses information entropy criterion.

Demonstration: penguins data

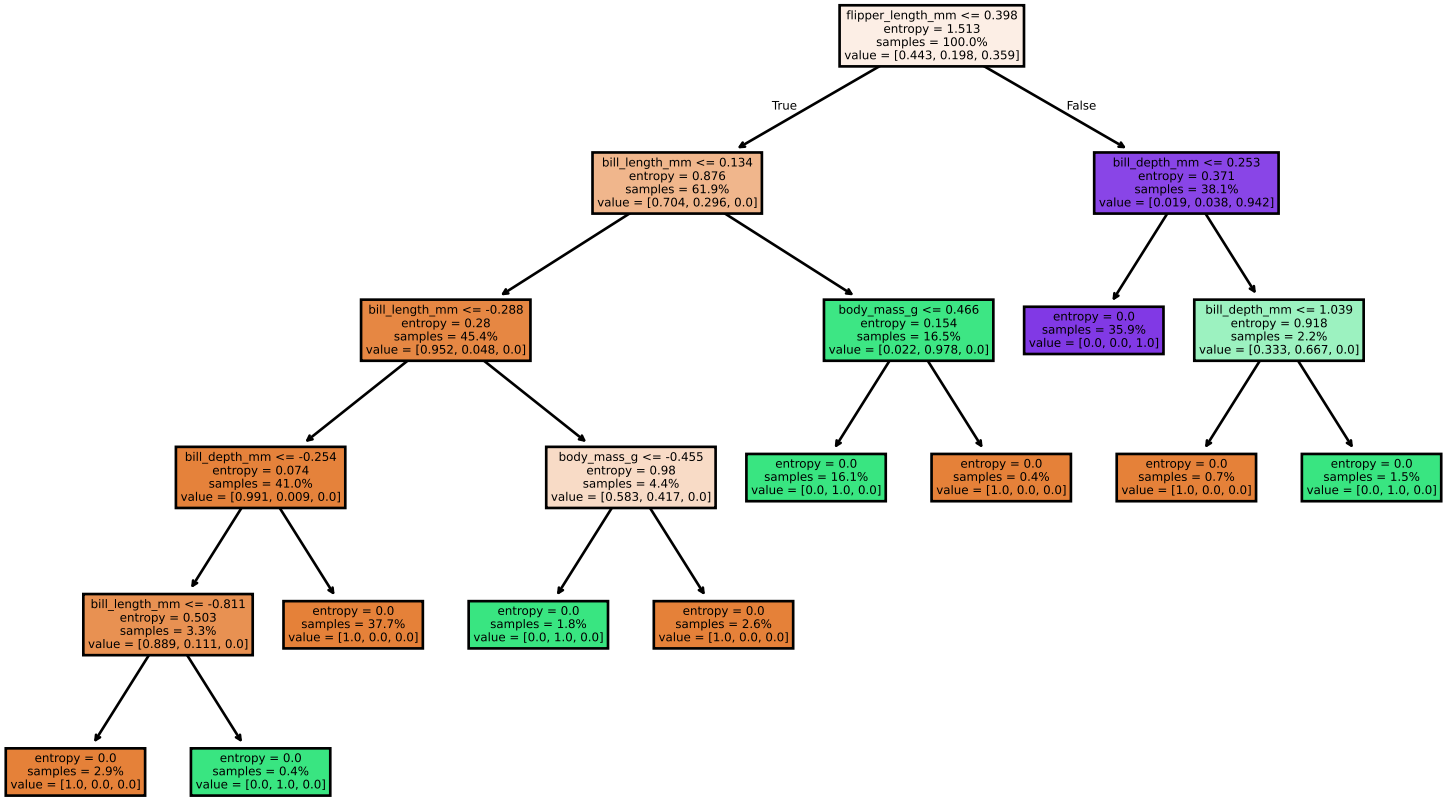


Figure: Tree classifier for penguins data. No maximum depth, and uses information entropy criterion.

Demonstration: penguins data

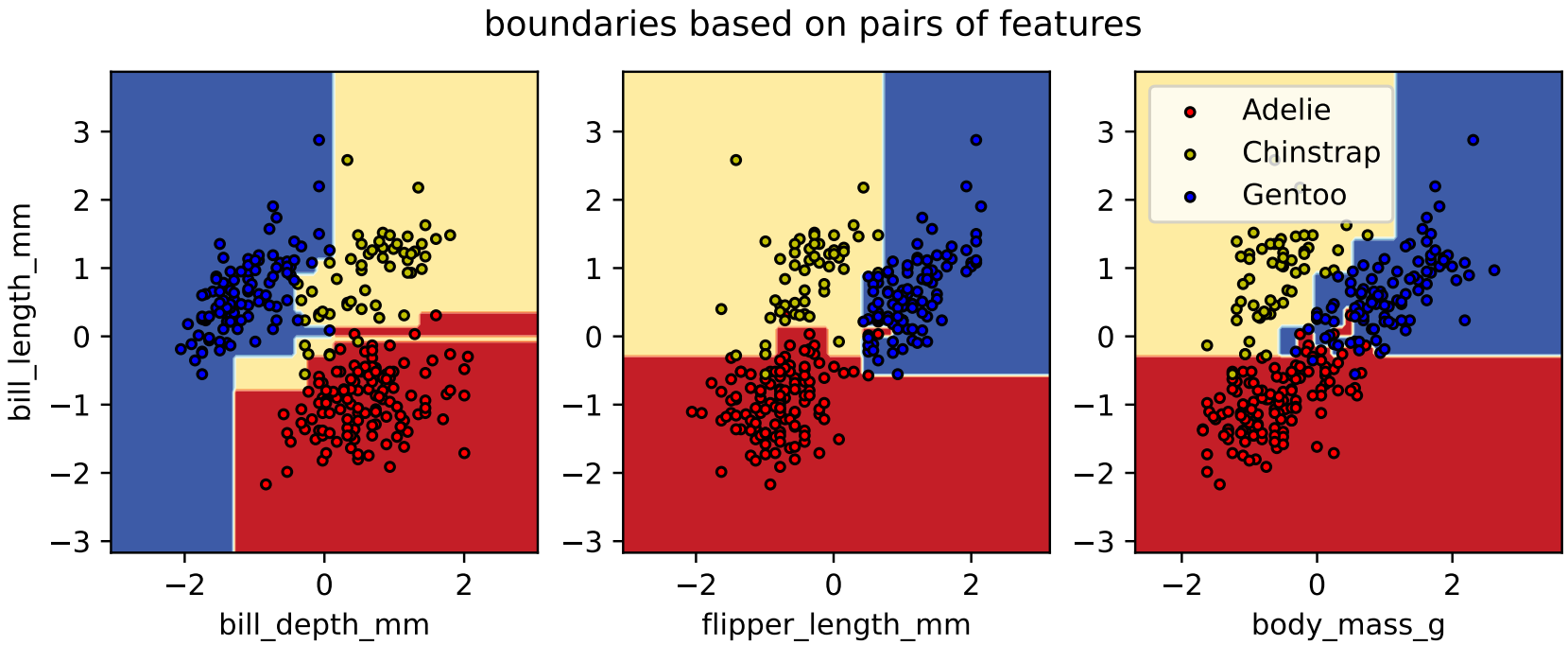


Figure: Tree classifier decision boundaries based on two features for penguins data.

- notice decision trees basically do piecewise constant predictions (see this again when doing regression)

Demonstration: penguins data

- consider now the **regression** problem

  → same problem identified in the second session

  → include species as an input feature?

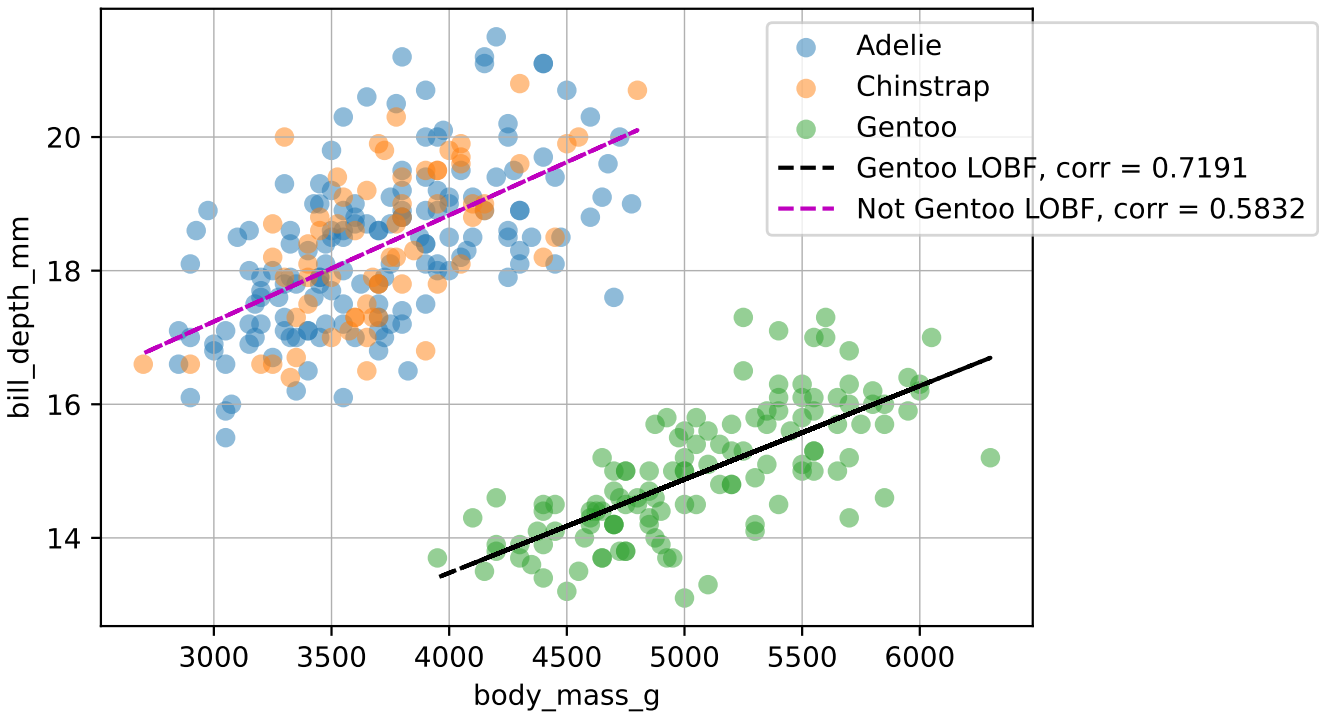


Figure: Instead of models for each species, one model that does different decisions based on species?

Demonstration: penguins data

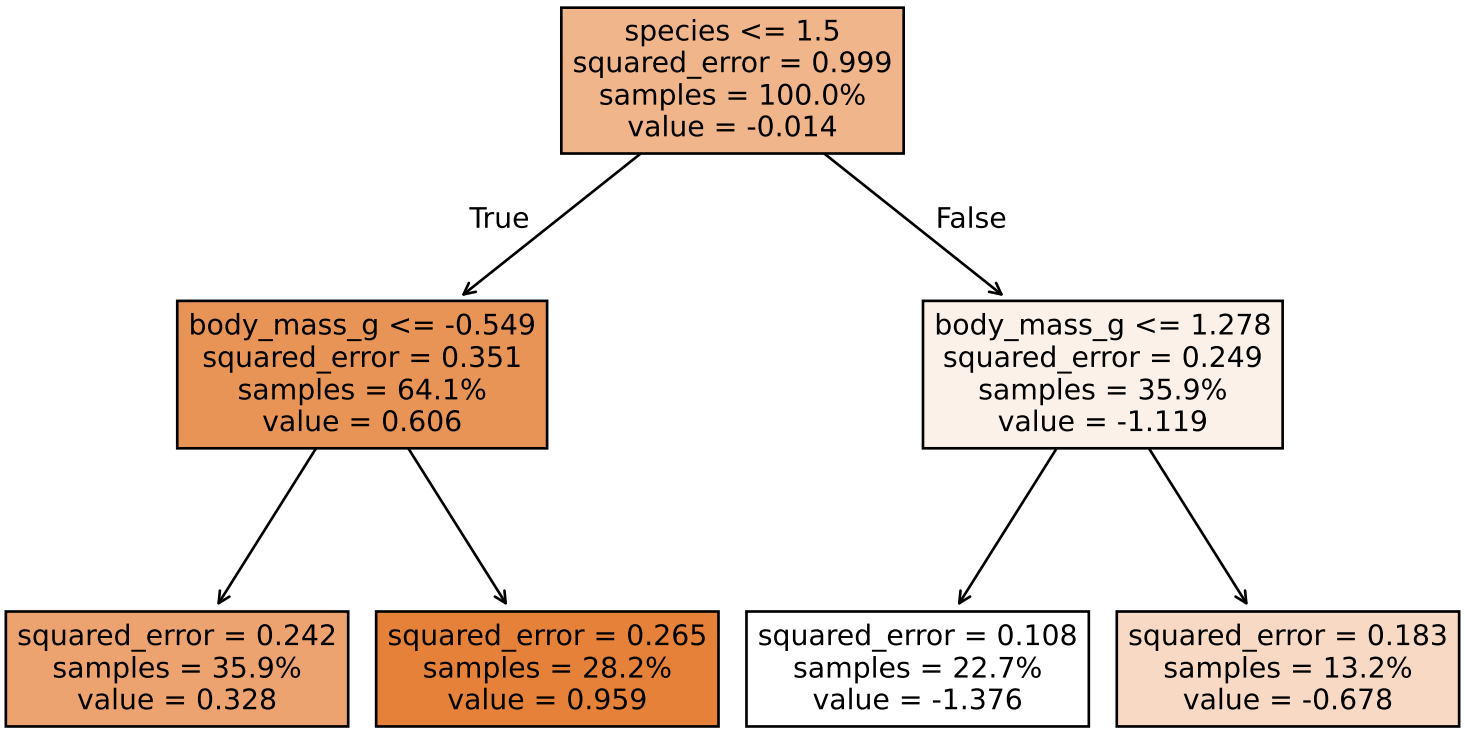


Figure: Tree regressor for penguins data. Maximum depth two, squared loss.

Demonstration: penguins data

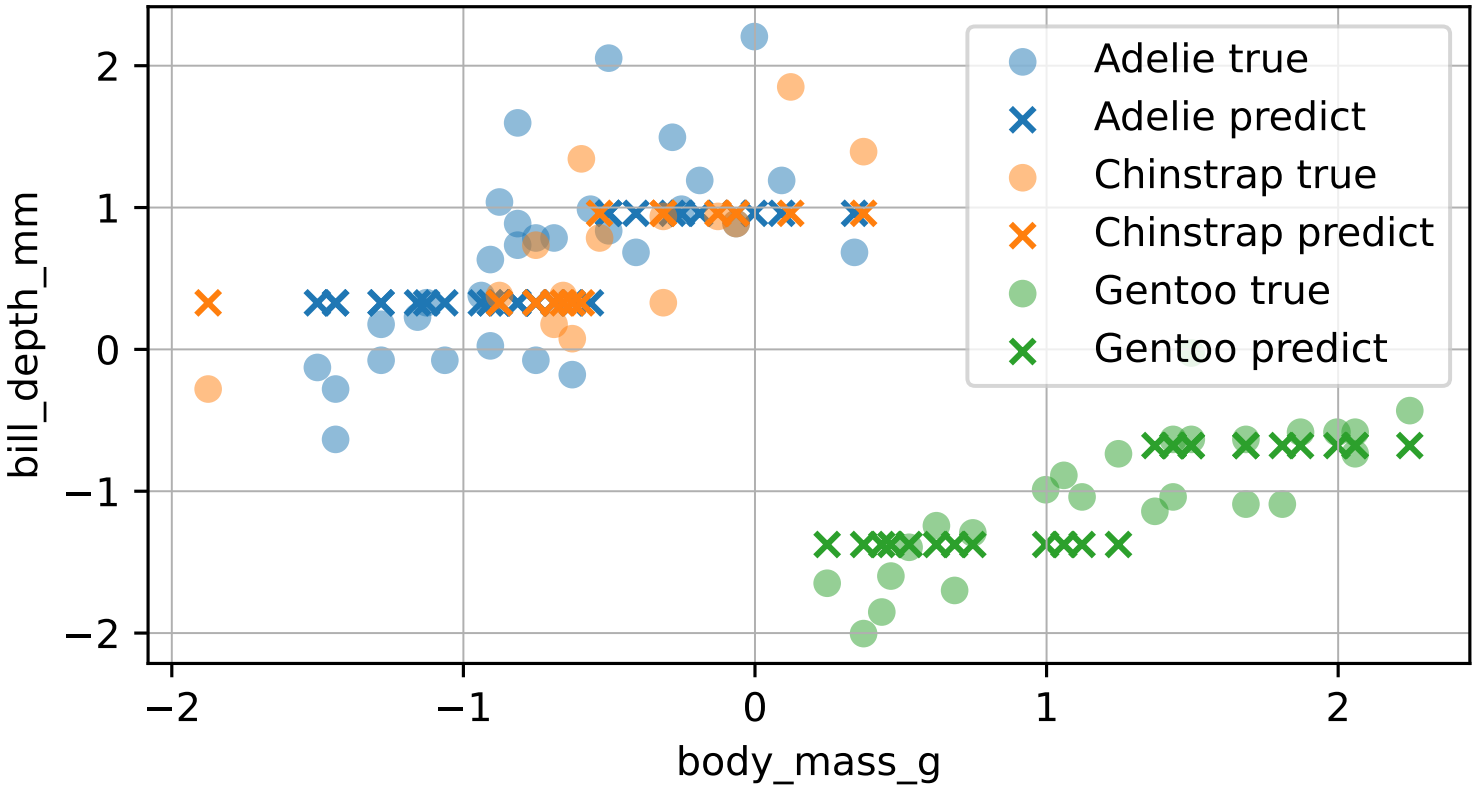


Figure: Tree regressor for penguins data. Maximum depth two, squared loss.

- notice the species predictions are clustered accordingly

  → notice the piecewise constant predictions (model too shallow)

Demonstration: penguins data

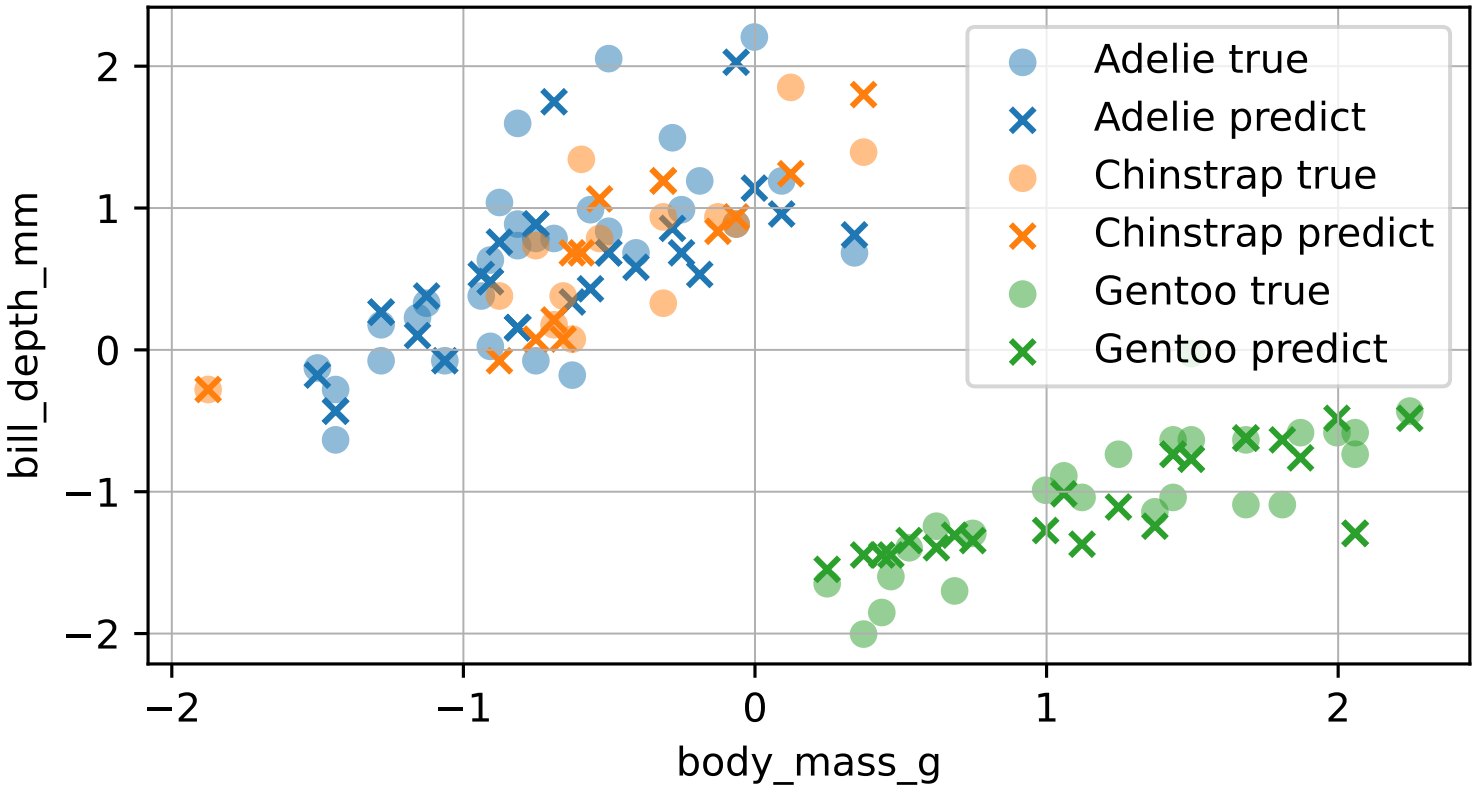


Figure: Tree regressor for penguins data. No maximum depth, squared loss.

- recover variability by having more complexity

Demonstration: penguins data

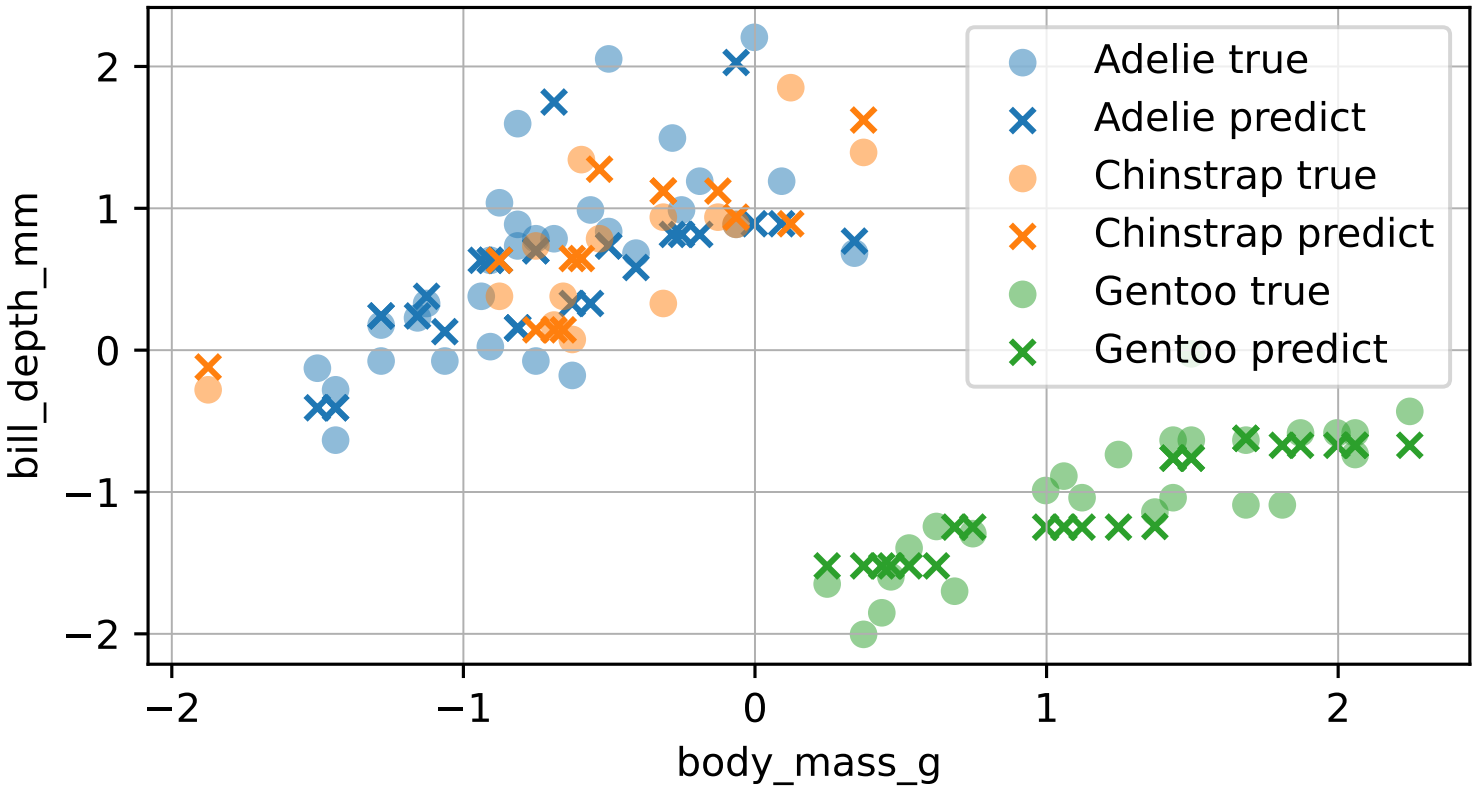
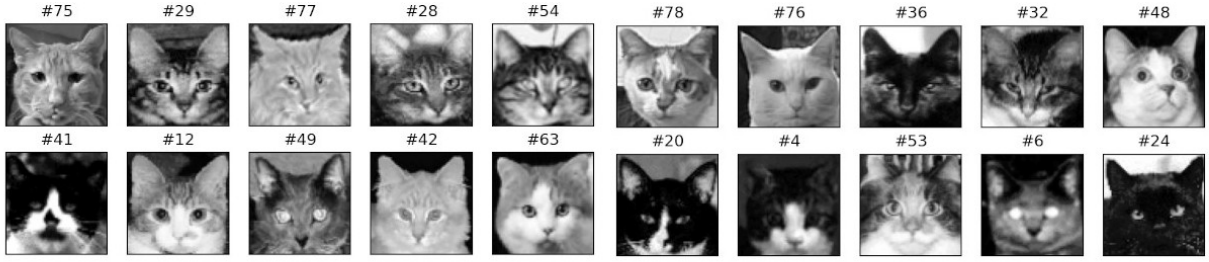


Figure: Tree regressor for penguins data with post-pruning. No maximum depth, squared loss.

- pruning to avoid over-fitting + promote robustness
  → remove nodes that add extra complexity but not that much skill (cf. A/BIC)

Schematic of random forests



full data

Figure: Schematic of random forests: full data.

Schematic of random forests

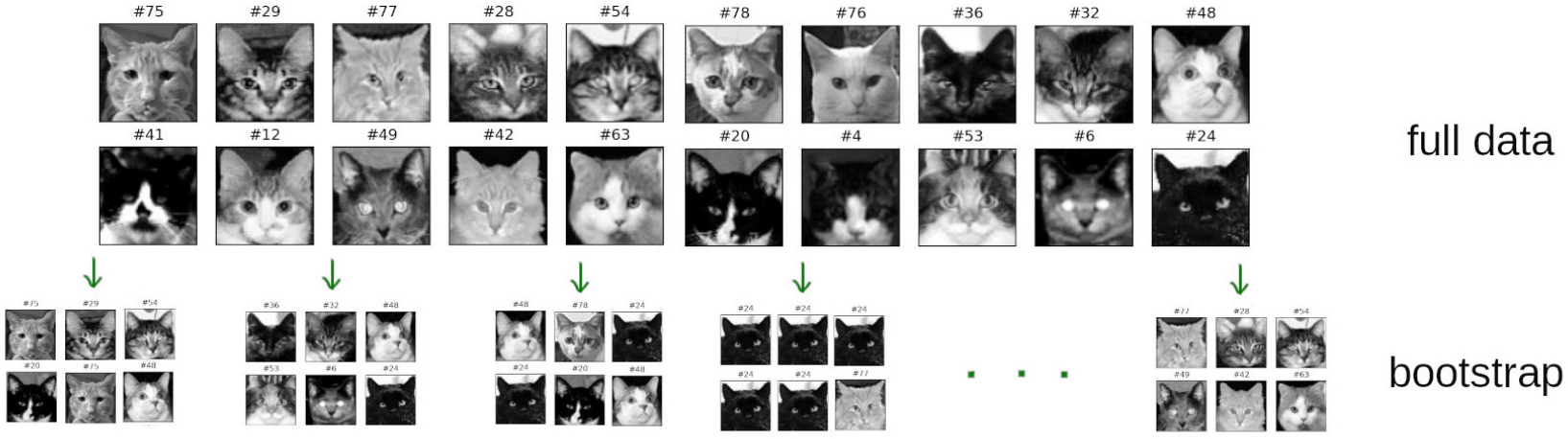


Figure: Schematic of random forests: bootstrap sampling, sub-sampling **with replacement** (!!!).

Schematic of random forests

Figure: Schematic of random forests: bagging (or bootstrap aggregation).

Schematic of random forests

Figure: Schematic of random forests: bagging (or bootstrap aggregation).

Schematic of random forests

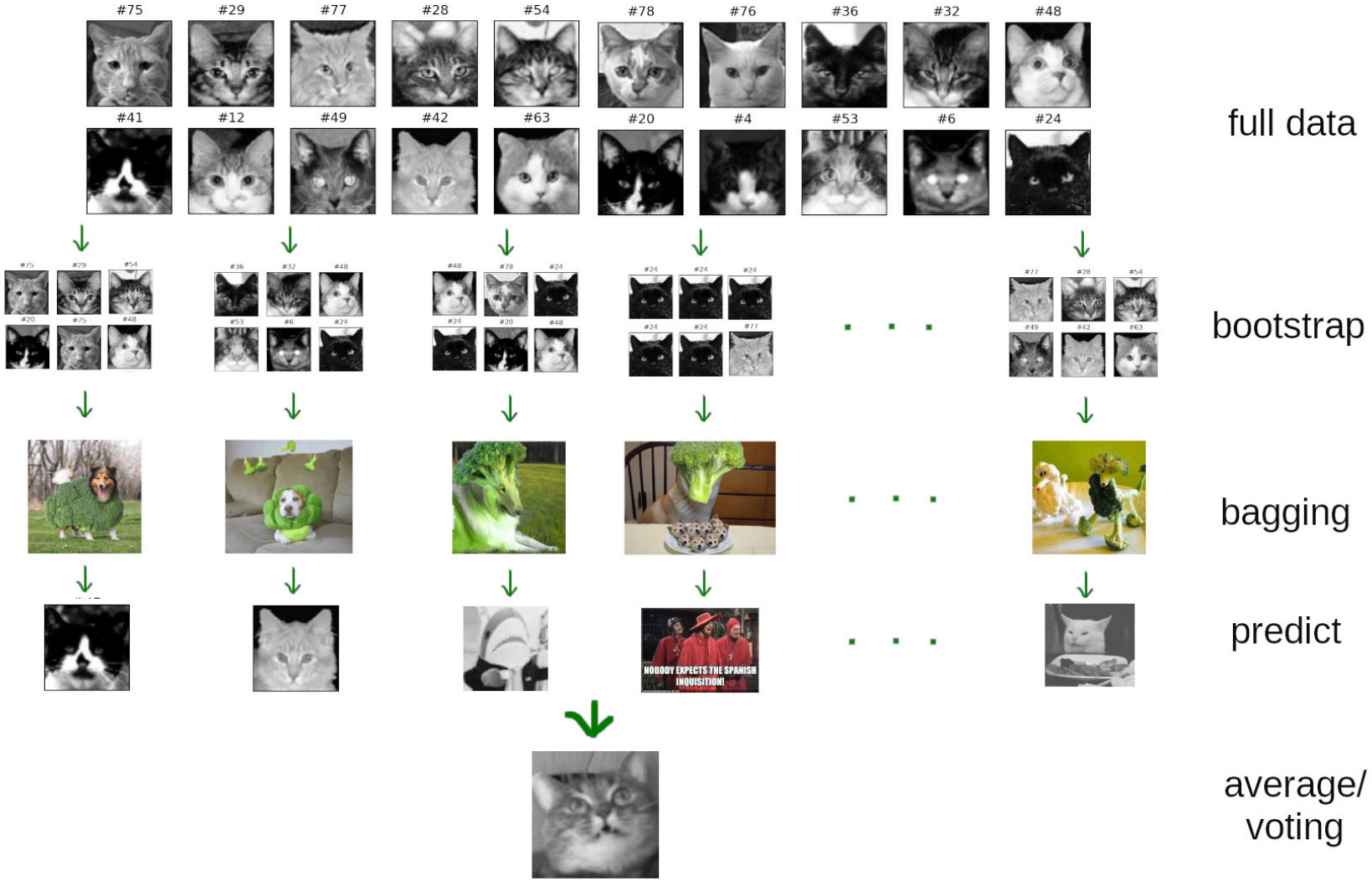


Figure: Schematic of random forests: averaging/voting (depending on regressor or classifier).

Schematic of boosting

Figure: Schematic of boosting: train a weak model, and compute mismatches (e.g. wrong colour).

Schematic of boosting

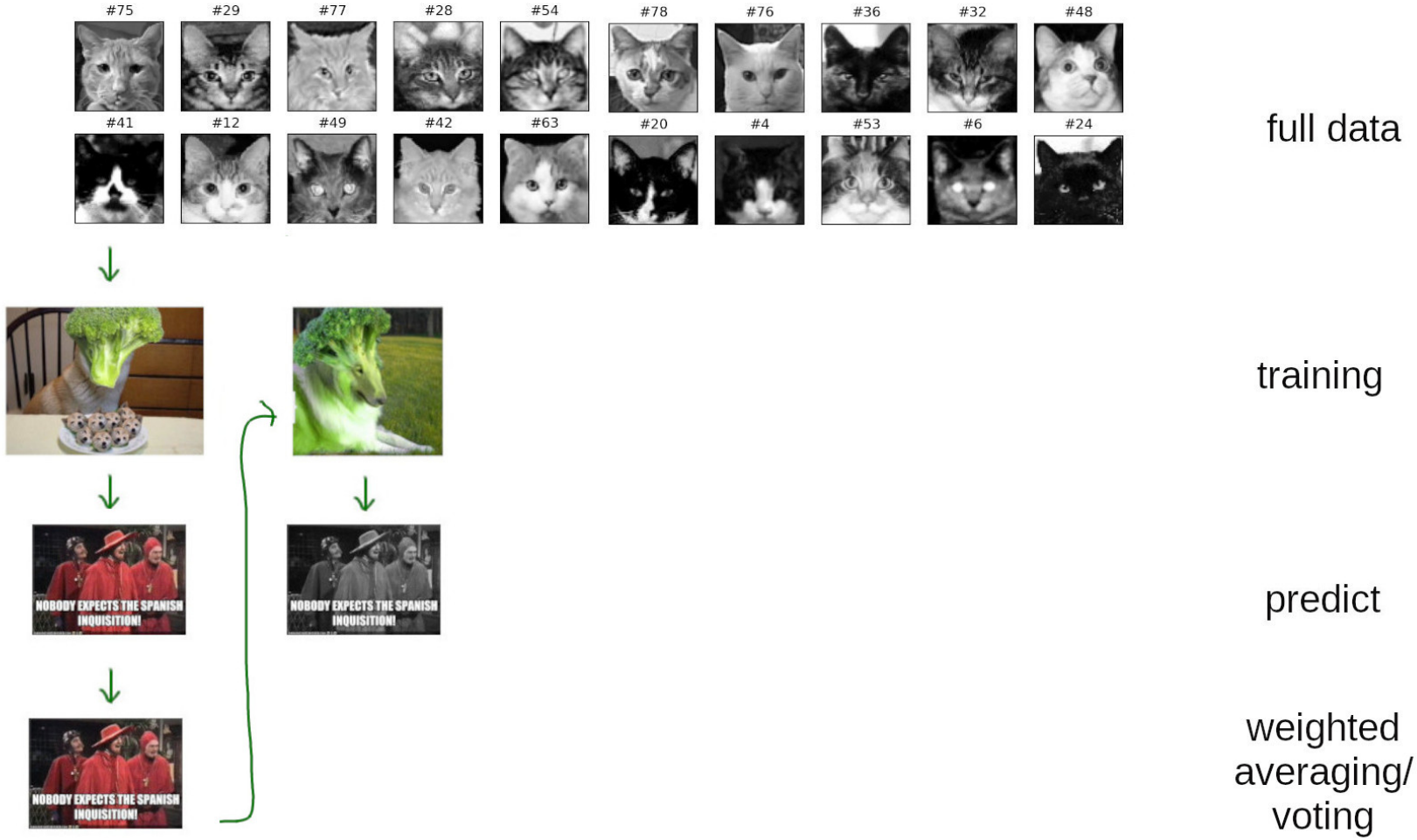


Figure: Schematic of boosting: train new model to reduce previous mismatches (e.g. predicting a grayscale image).

Schematic of boosting

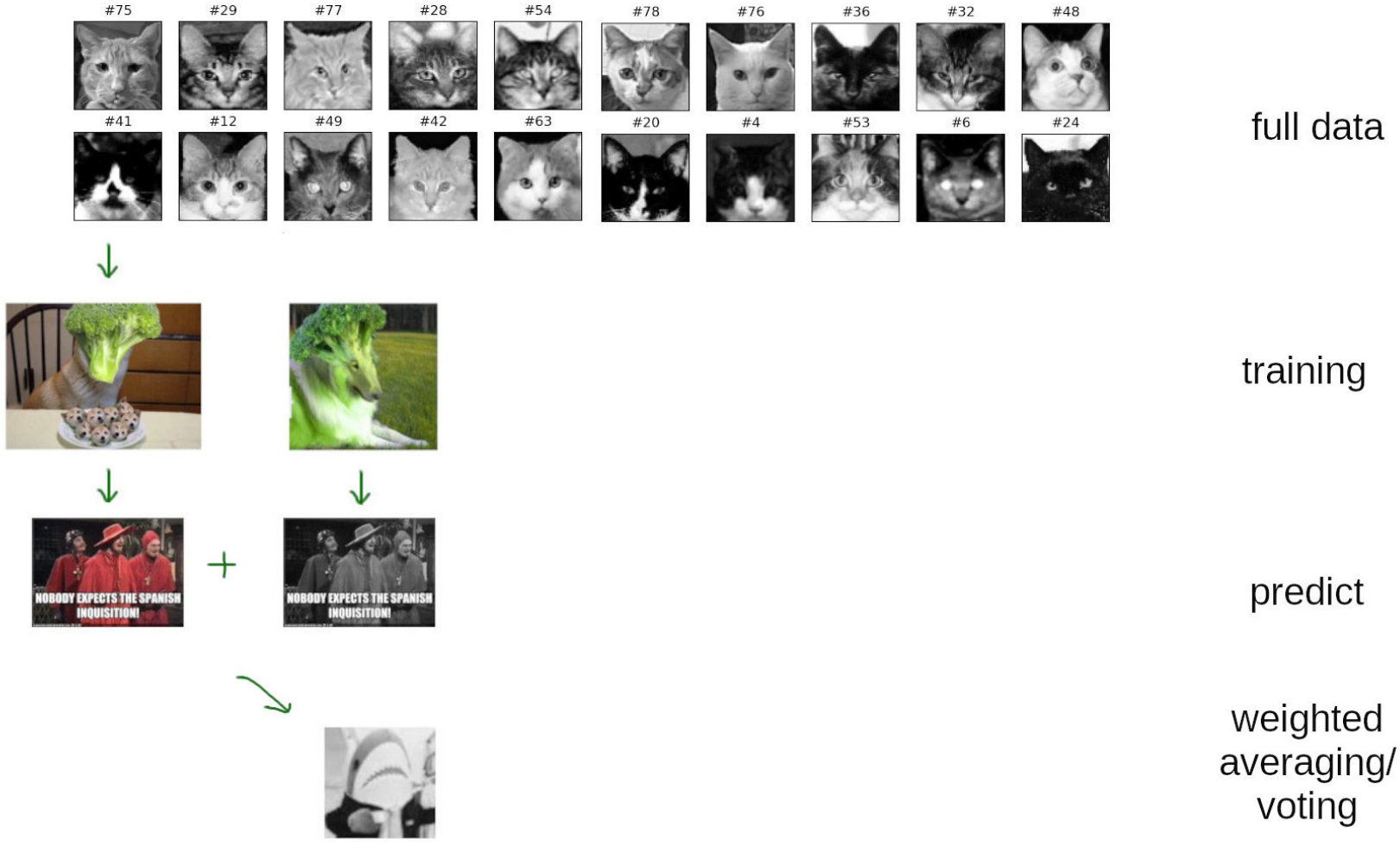


Figure: Schematic of boosting: re-weigh, predict, compute mismatches again (e.g. wrong animal).

Schematic of boosting

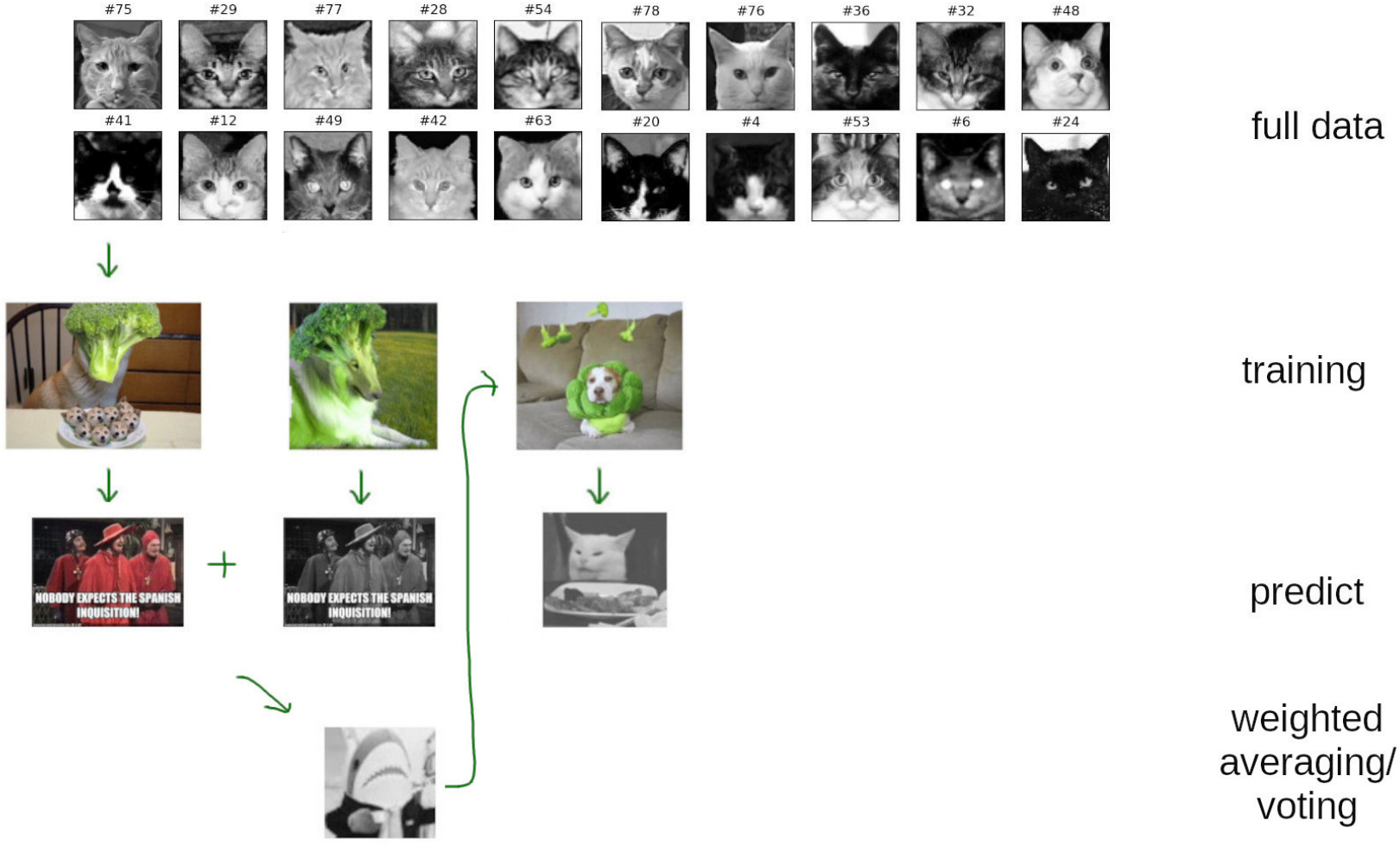


Figure: Schematic of boosting: train new model aiming to further reduce mismatches (e.g. predicting a cat).

Schematic of boosting

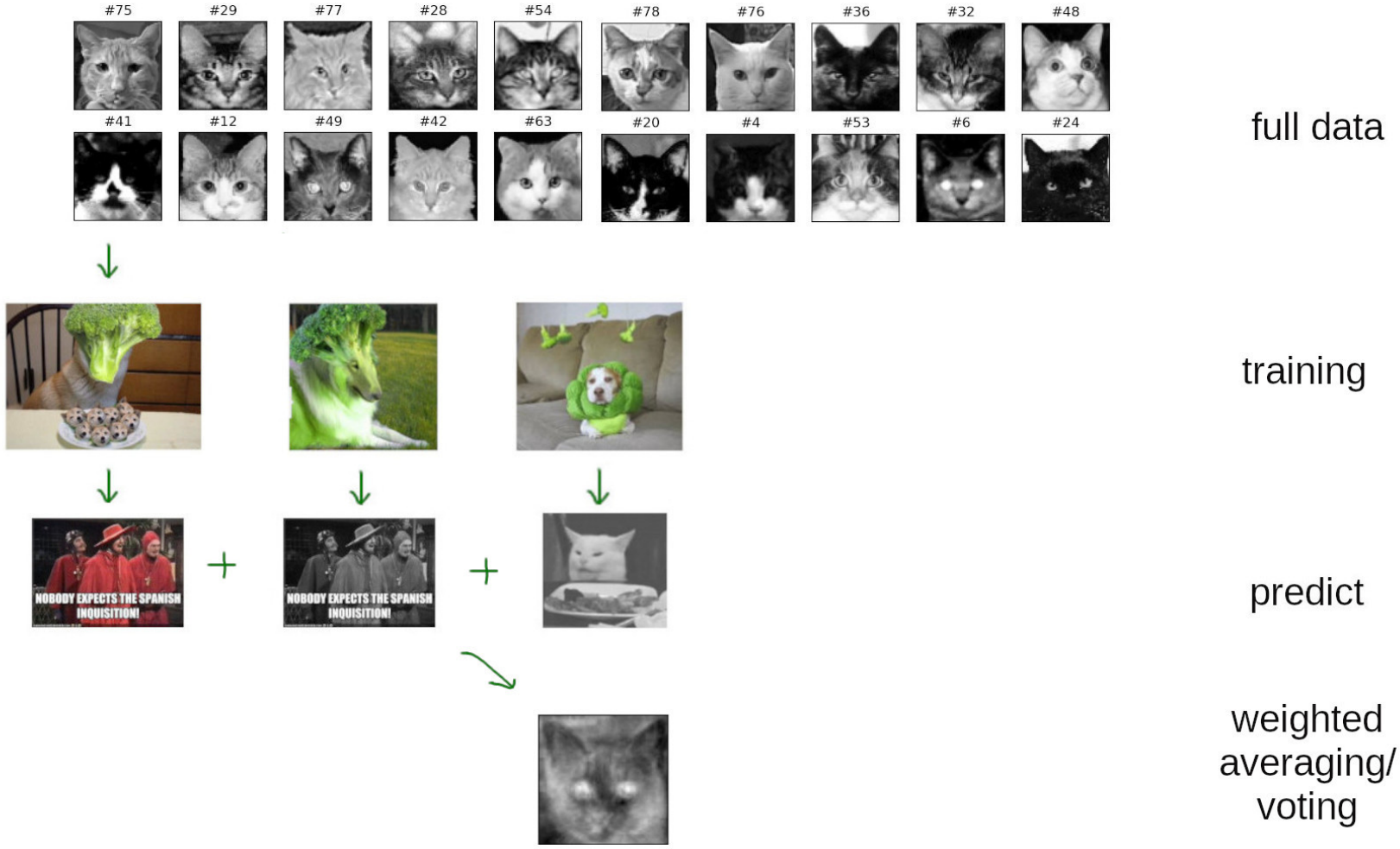


Figure: Schematic of boosting: adjust weight and continue...(e.g. cursed cat)

Schematic of boosting

Figure: Schematic of boosting: adjust weight and continue...(e.g. blurry Miffy)

Schematic of boosting

Figure: Schematic of boosting: stop at some point (e.g. recovered a Miffy).

Gradient boosting

- boosting is sequential: builds tree to build on previous issues, then take a sum, aims to reduce **bias**

  $\rightarrow$ bagging is done in **parallel**, and averaging reduces **variance**
- aims to minimise errors in predictions by targeting the bad ones

  $\rightarrow$ i.e. an optimisation problem (again!)

  $\rightarrow$ the **gradient** part is then it uses gradient-based methods to find minimiser (e.g. SGD-based methods)

Example: penguins data

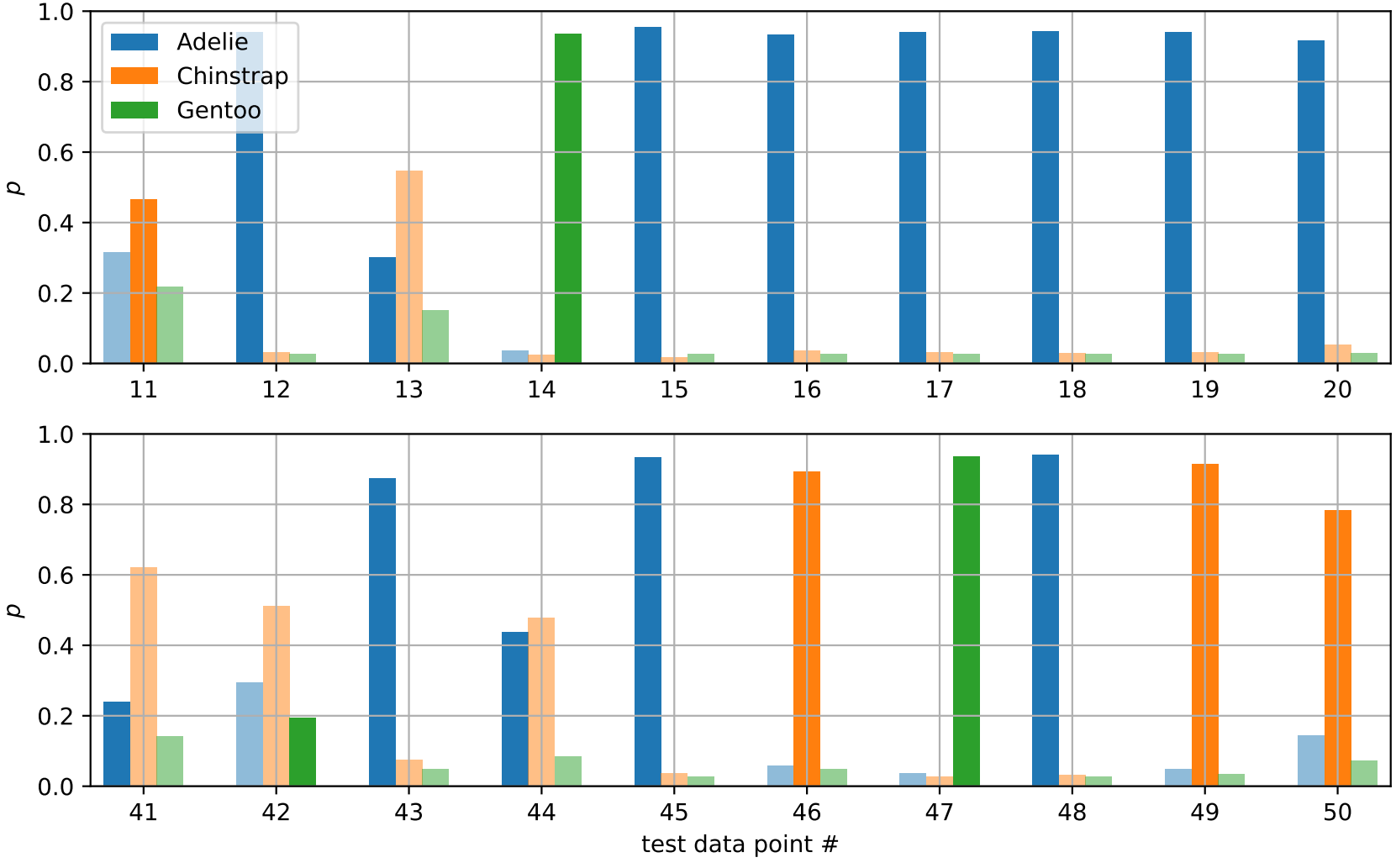


Figure: Probabilities associated with the predictions on test set with gradient boosting. Highlighted bar denotes truth, and note highest bar (which are higher than in random forests) is not always highlighted bar (e.g. index 13, 41, 44 as in random forests actually...)

Demonstration

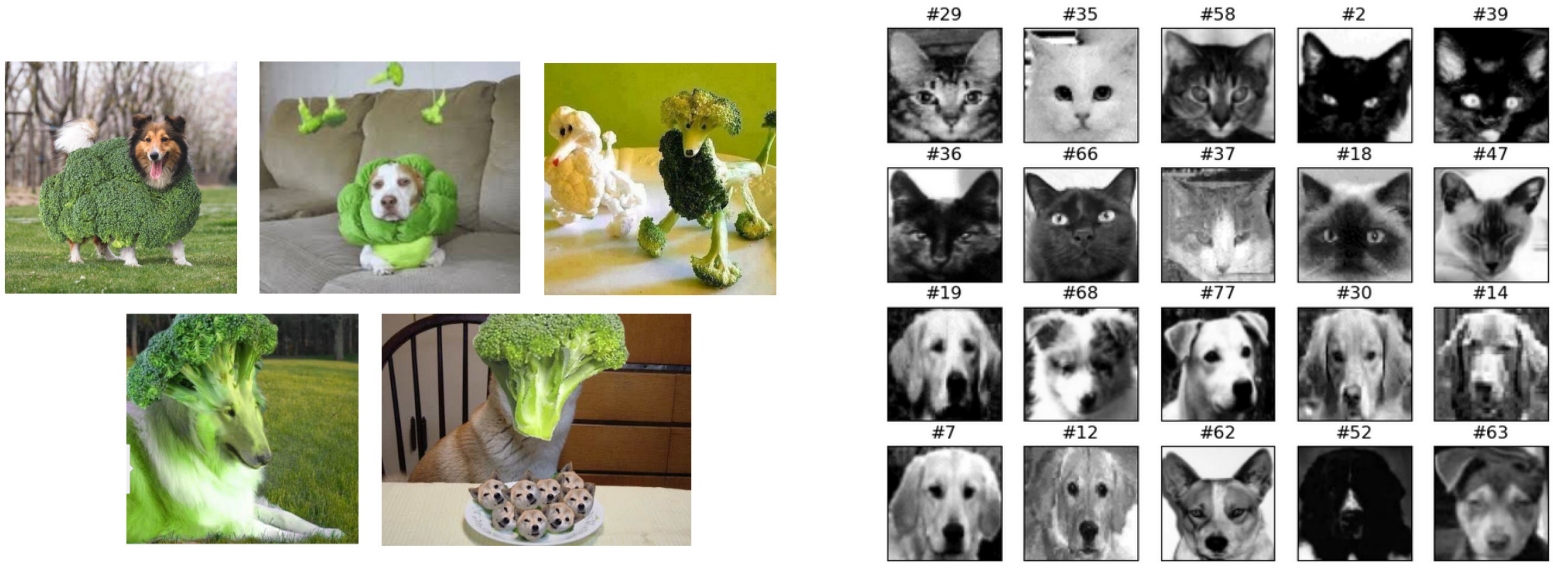


Figure: Forest of brocollies to classify/predict cats and dogs?

- classification tasks as finding separators for data
- ensemble methods as classfiers/regressors
  $\rightarrow$ bagging/bossting applicable to non-trees in principle
- need to cross-validate and tune hyper-parameters accordingly!

Moving to a Jupyter notebook $\rightarrow$