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/OCES4303_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 4303 :

an introduction to data-driven and ML methods in ocean sciences

Session 5: classifications

Outline

- classification

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

  → e.g. predicting species, image classification, etc.

- task of finding a separators between data

  → Linear/Quadratic Discrimnant Analysis (L/QDA)

  → Support Vector Machine (SVM)

  → general choices of loss function + regularisation through Stochastic Gradient Descent (SGD)

  → digression: links with neural networks later

- demonstration: penguins data

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?

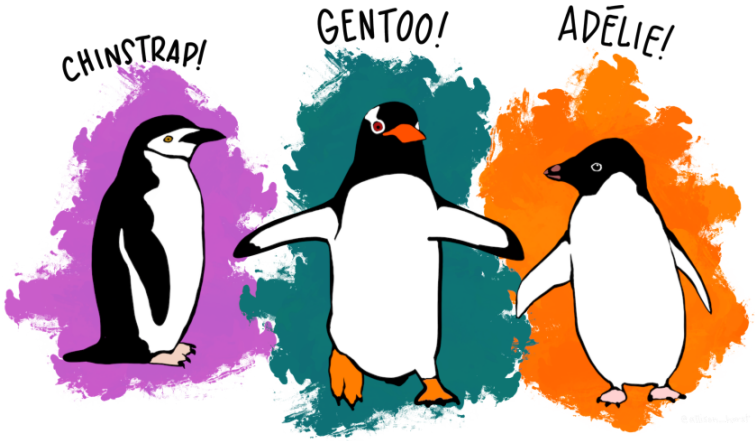


Figure: Palmer penguins logo. From `https://allisonhorst.github.io/palmerpenguins/`

Example: image classification

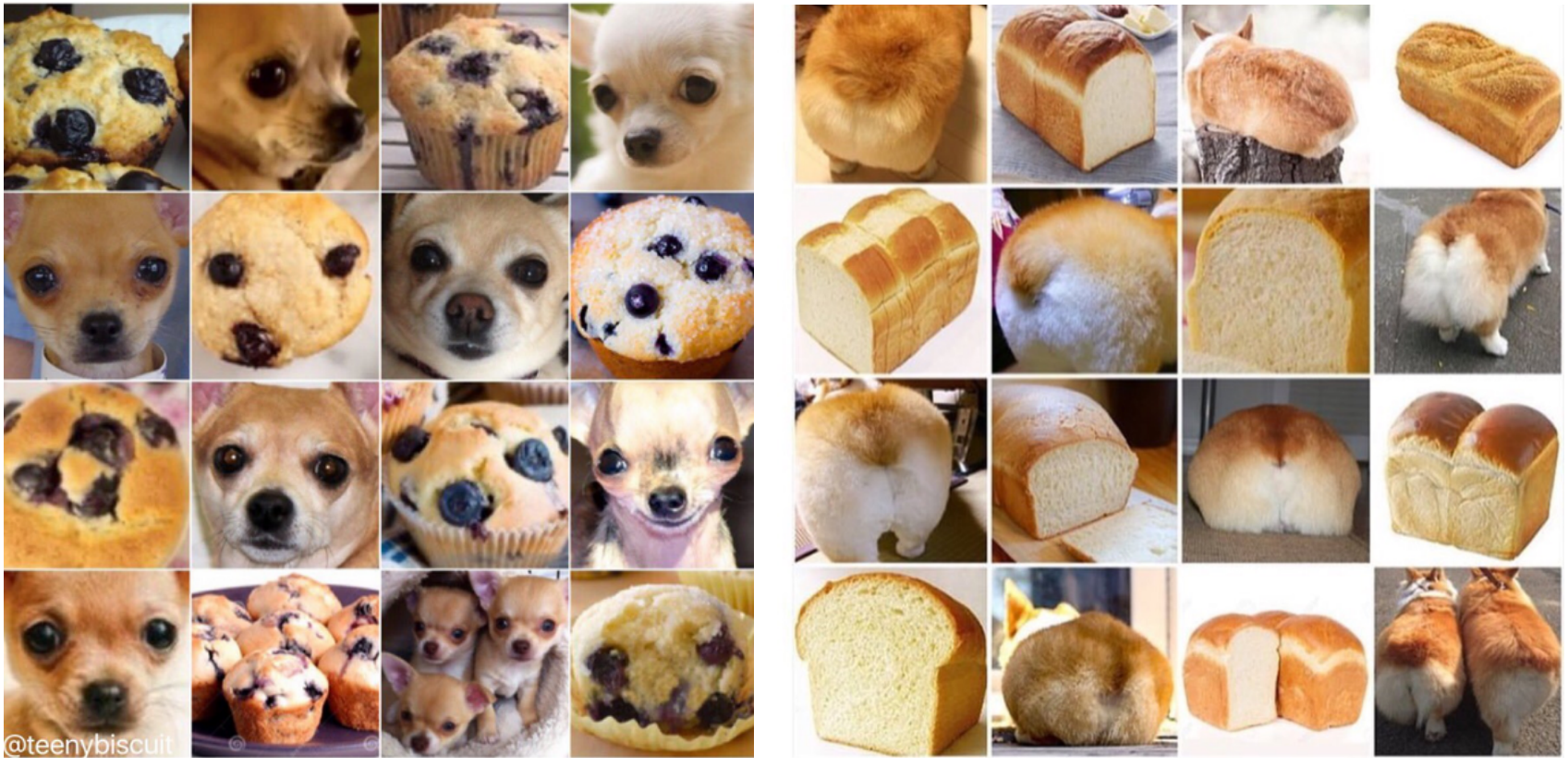


Figure: Various entries from the "animal or things" meme, as found on the internet.

Classification as a geometry problem

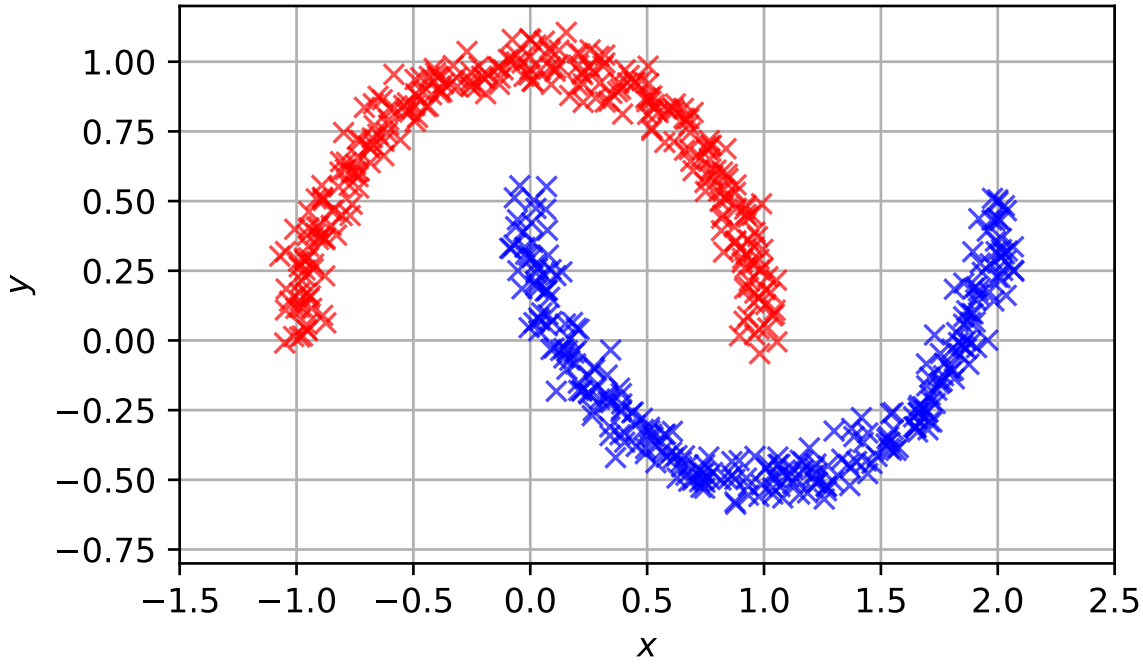


Figure: Moon data coloured by the cluster it is in.

- 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

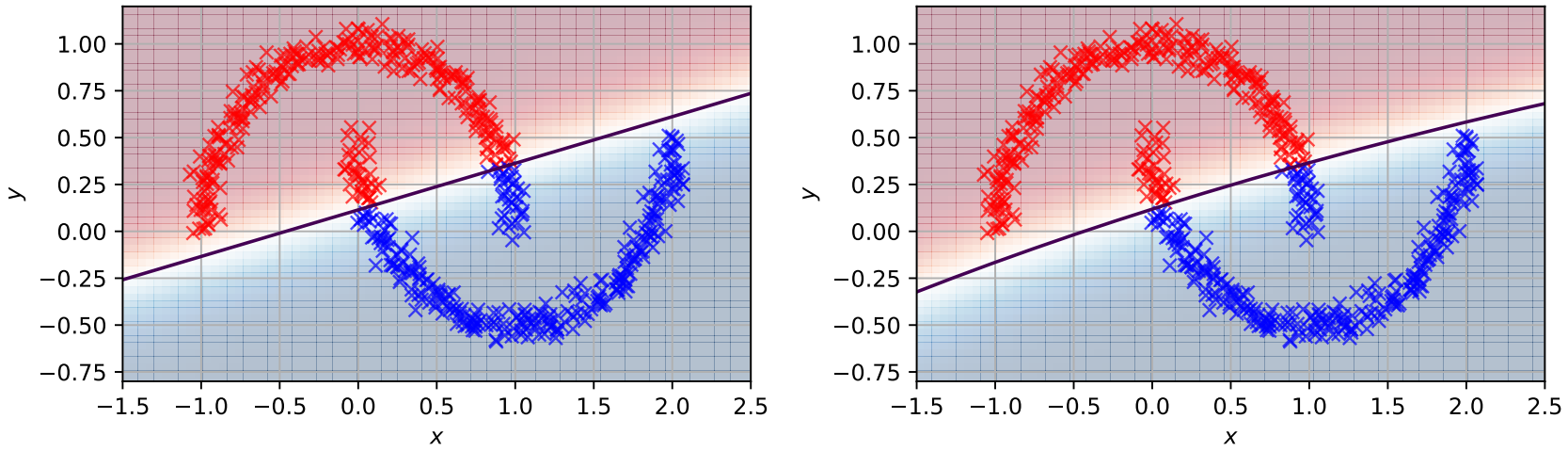


Figure: Classification by a realisation of (left) LDA and (right) QDA, with decision boundaries and probabilities.

- 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

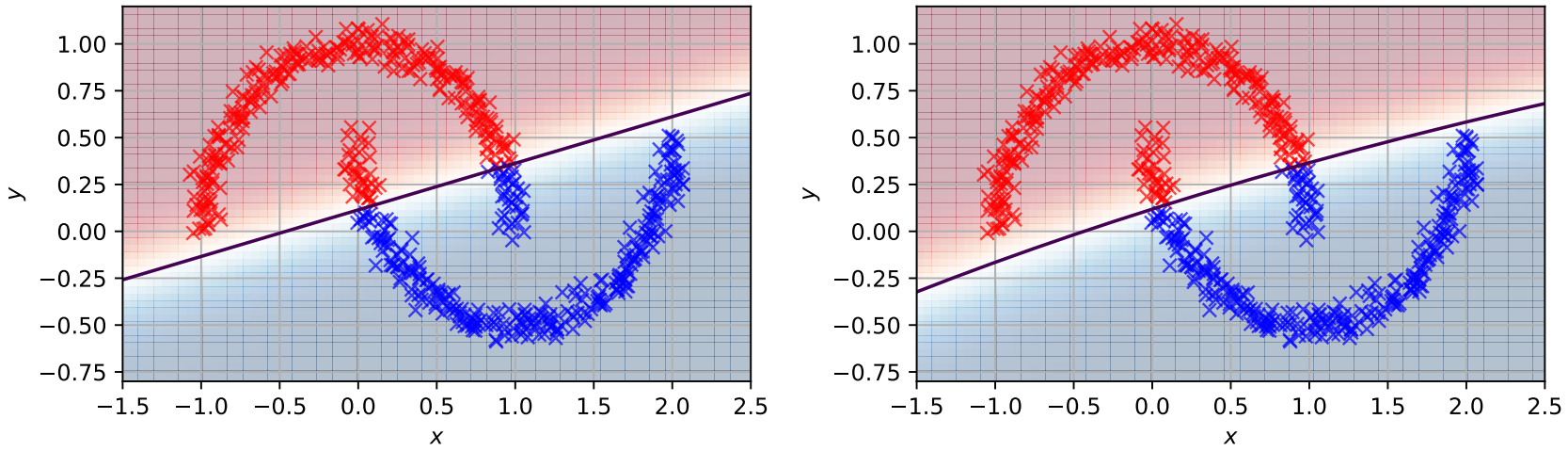


Figure: Classification by a realisation of (left) LDA and (right) QDA, with decision boundaries and probabilities.

- 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

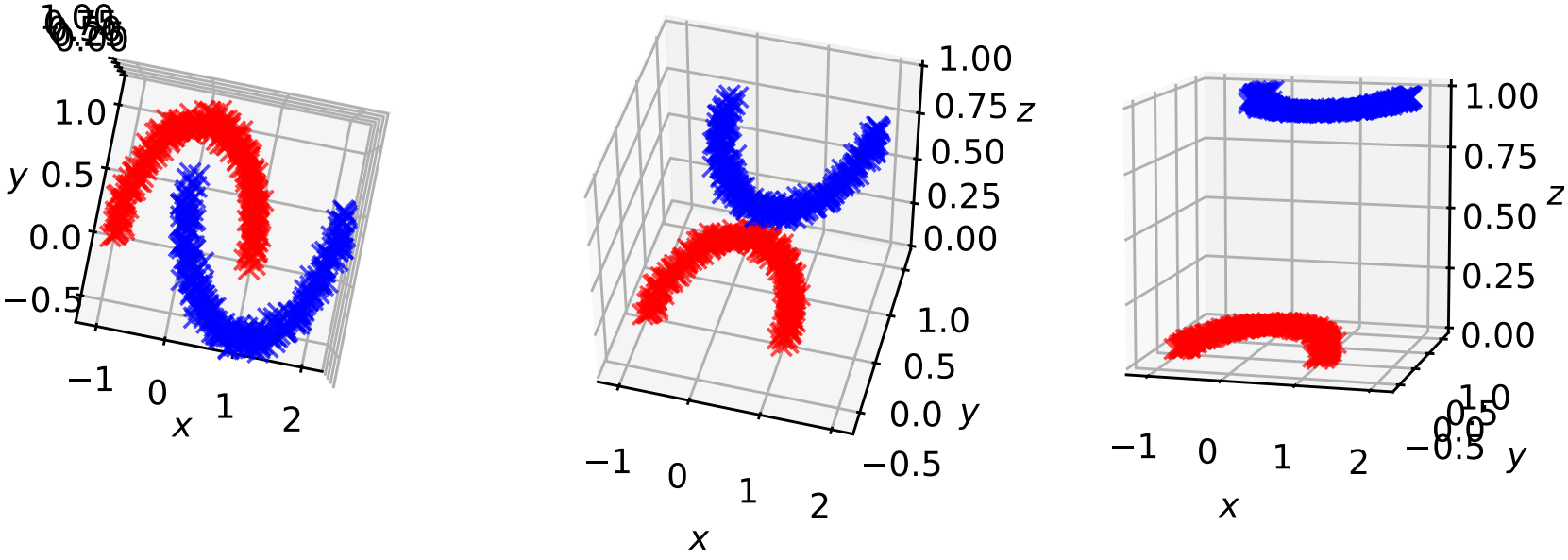


Figure: Moon data where I artificially 'lift' one of the clusters.

- 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

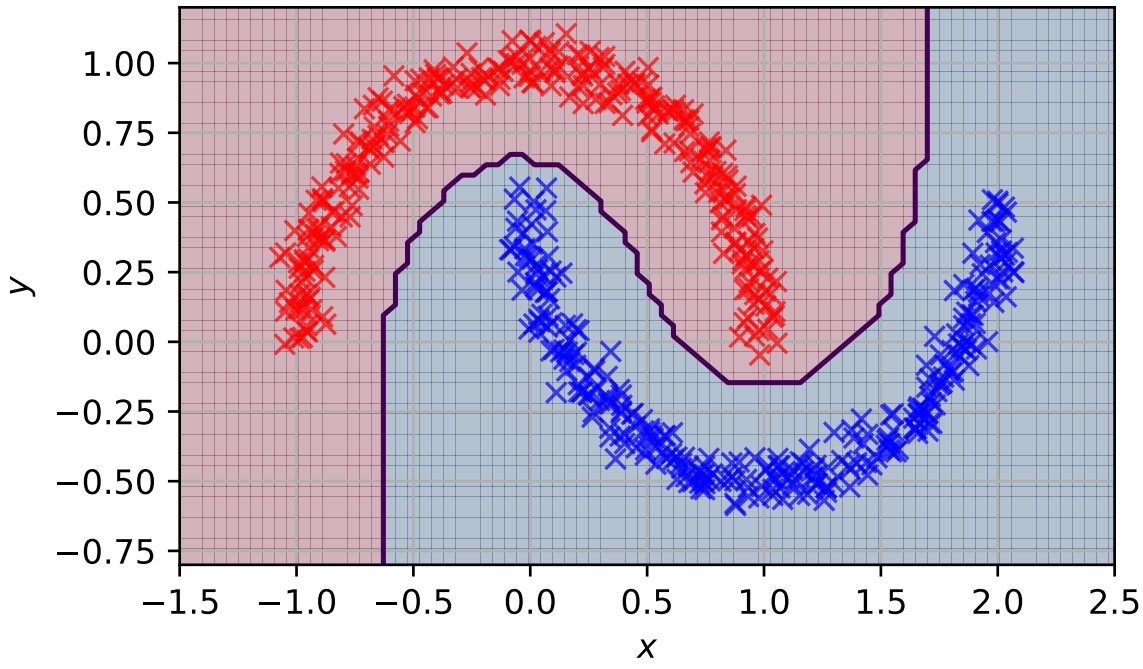


Figure: Moon data using a SVM with the radial basis function as the kernel.

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

  $\rightarrow$ no probability measure here for decision

SVM

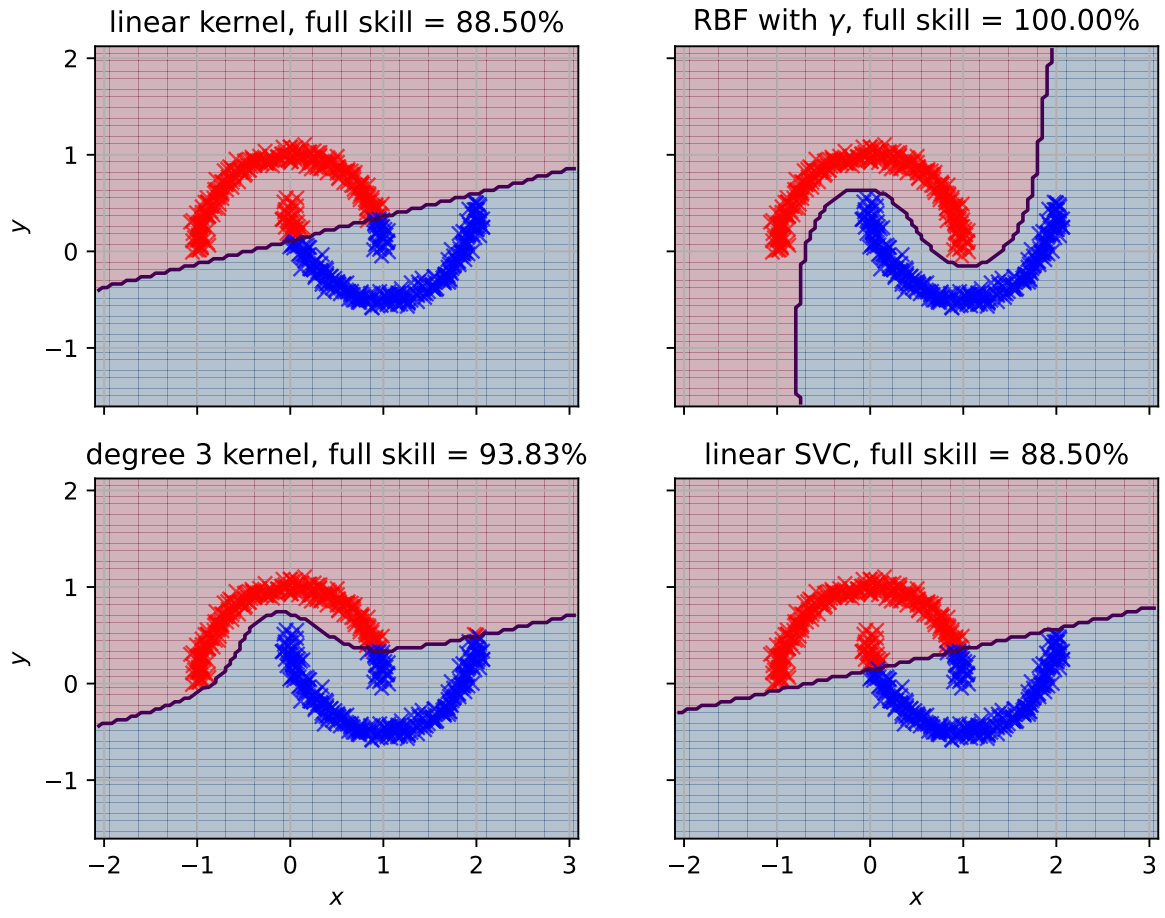


Figure: SVM on moon data with different choices of kernels and regularisations.

Issues with SVM: dependence on scaling

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

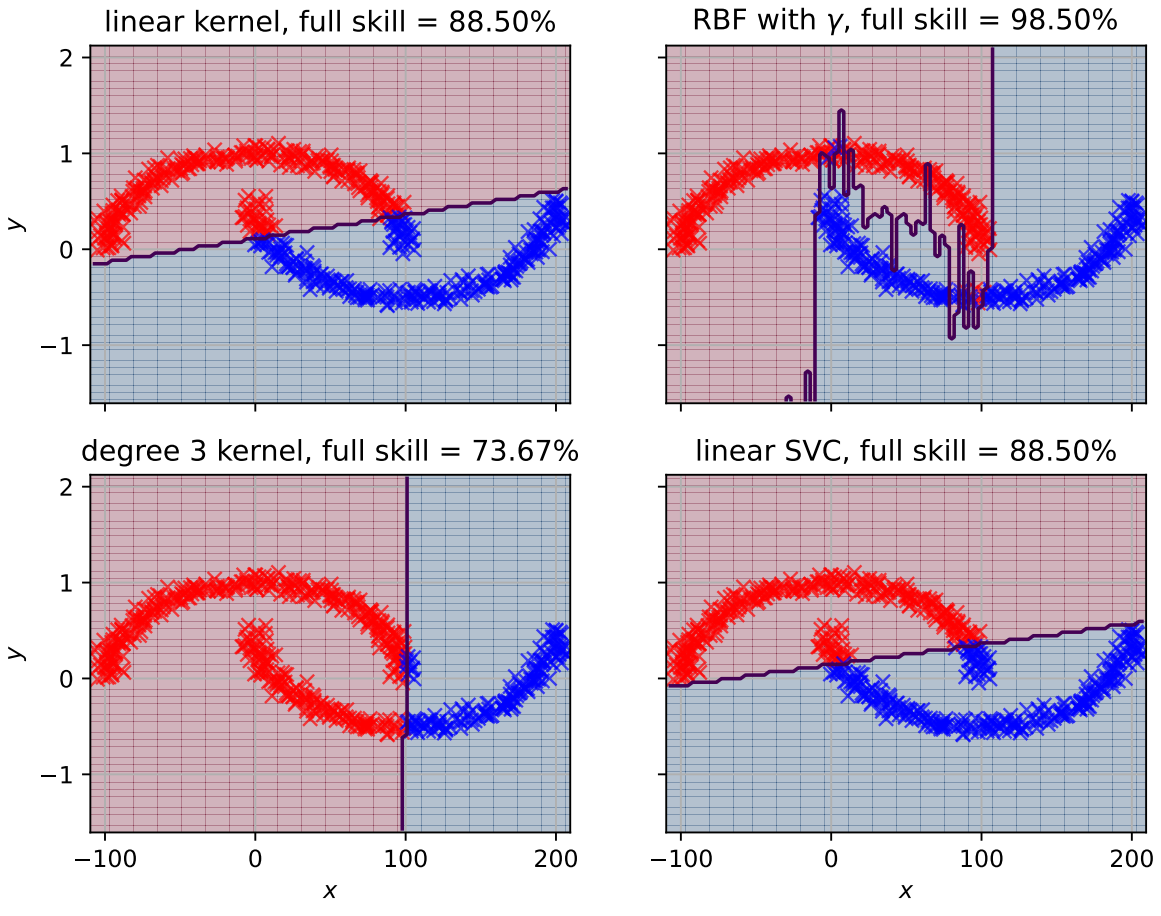


Figure: SVM on stretched moon data with different choices of kernels and regularisations.

Issues with SVM: unbalanced data

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

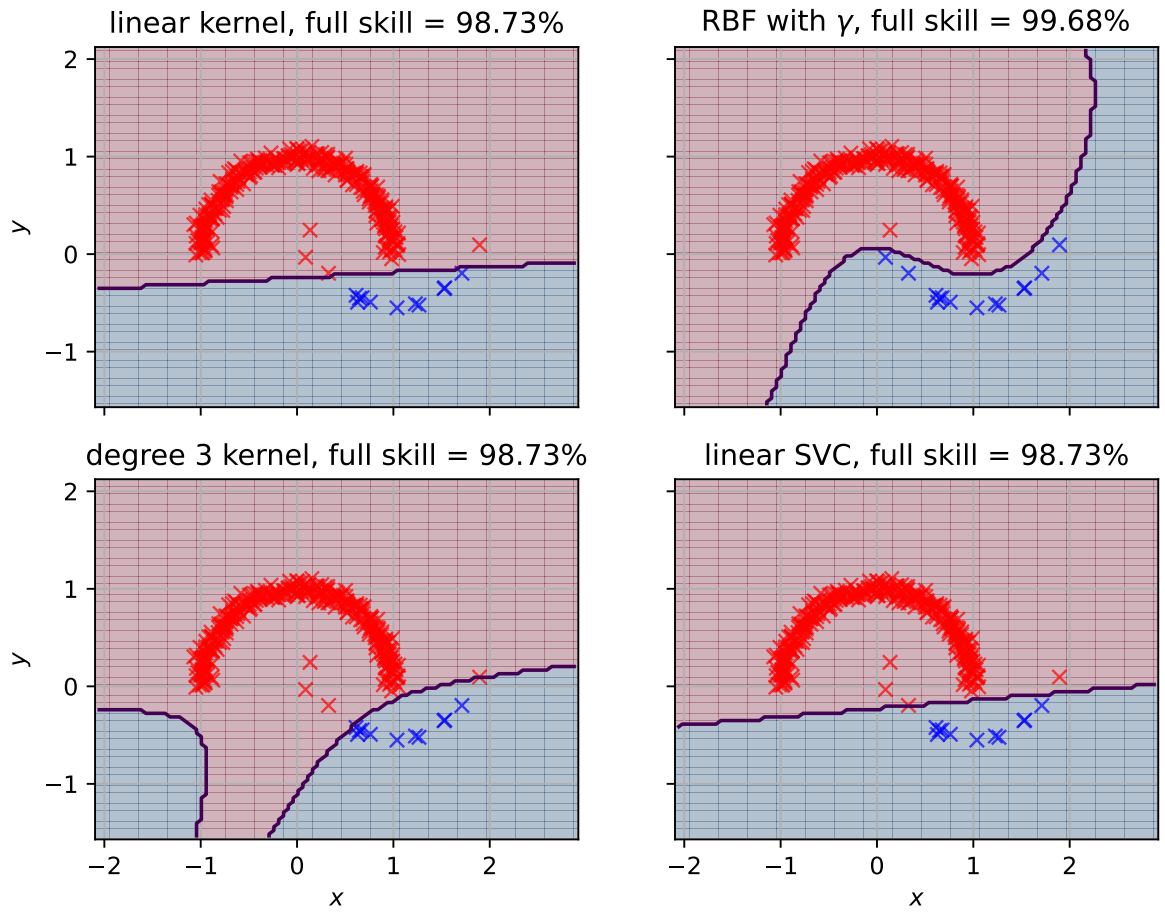


Figure: SVM on unbalanced moon data with different choices of kernels and regularisations.

Issues with SVM: unbalanced data with re-weighting

- one way is to change the weighting on the data

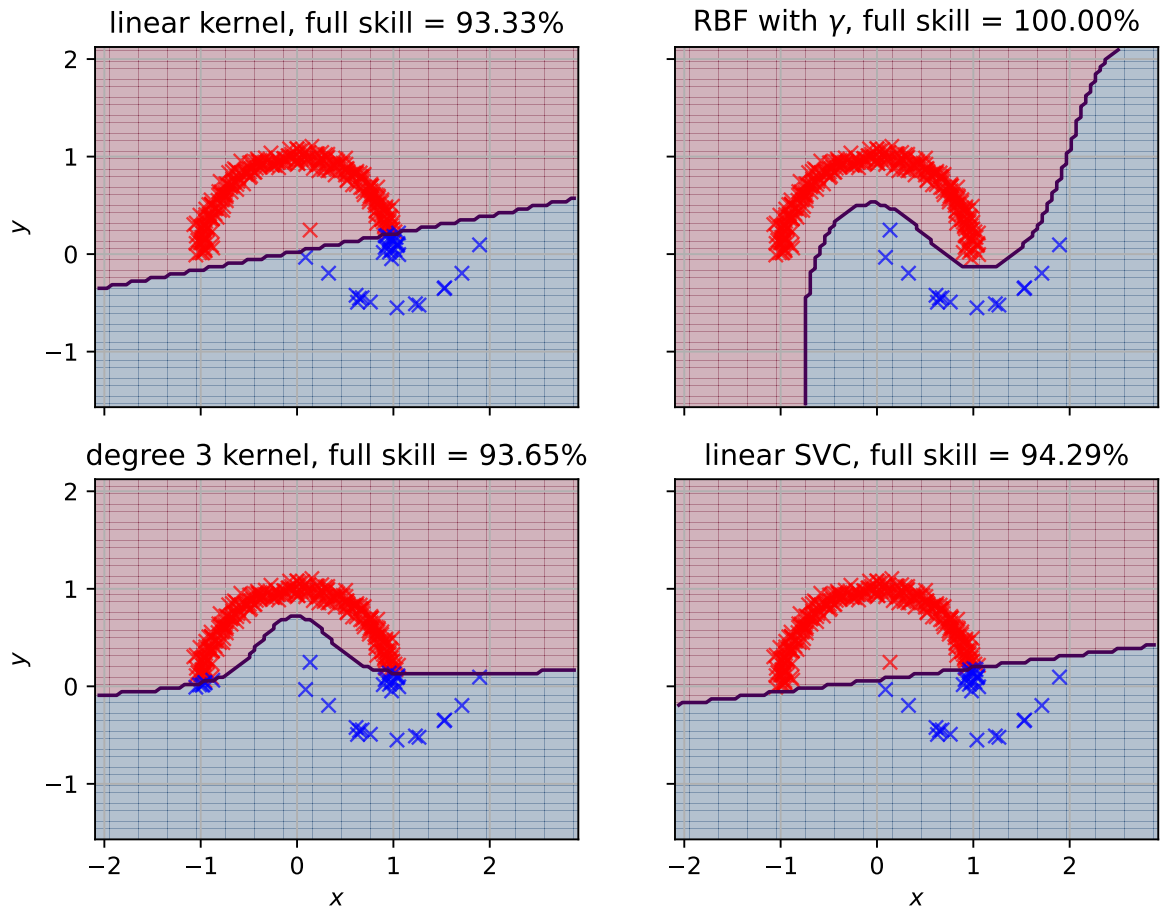


Figure: SVM on unbalanced moon data but with weight rebalancing, with different choices of kernels and regularisations.

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

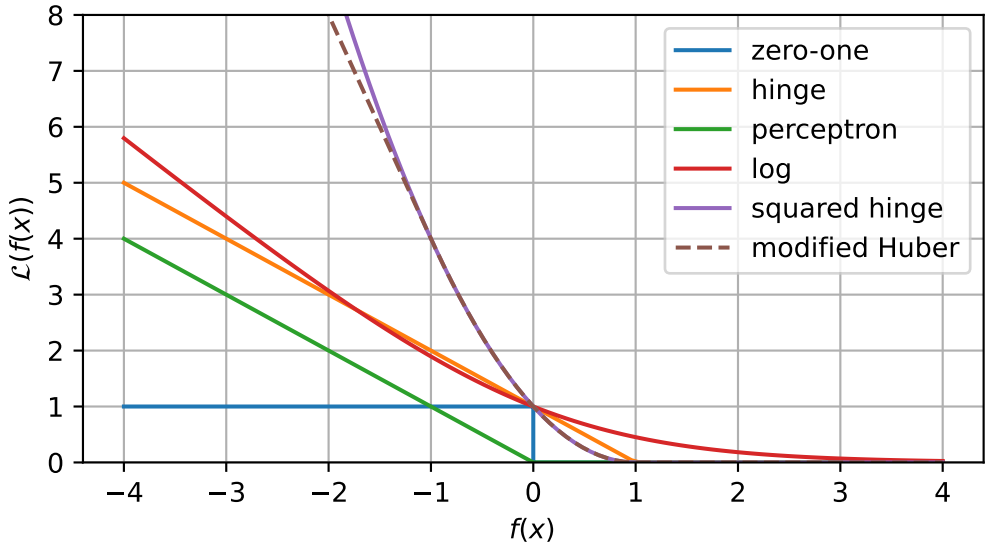


Figure: Selection of loss functions for classification.

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

Special cases

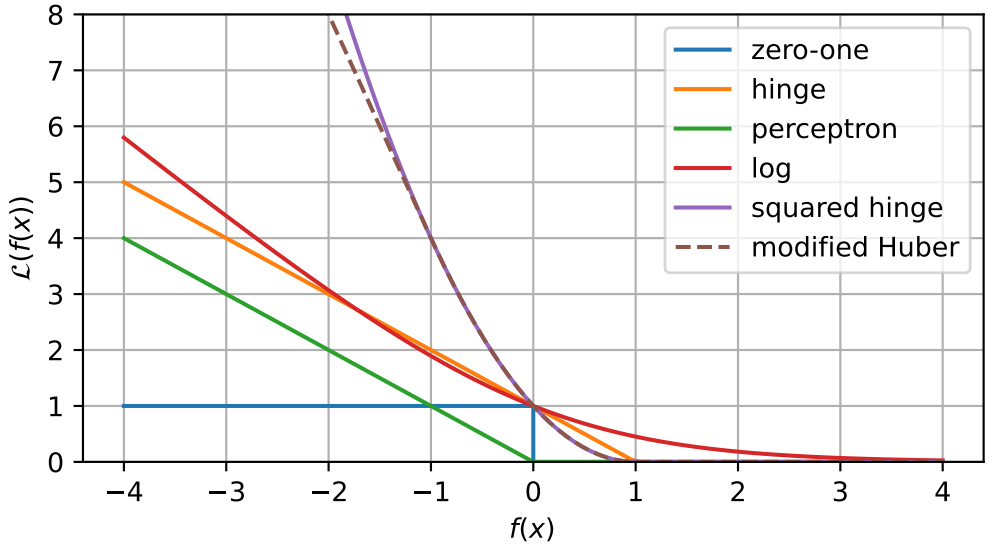


Figure: Selection of loss functions for classification.

- 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

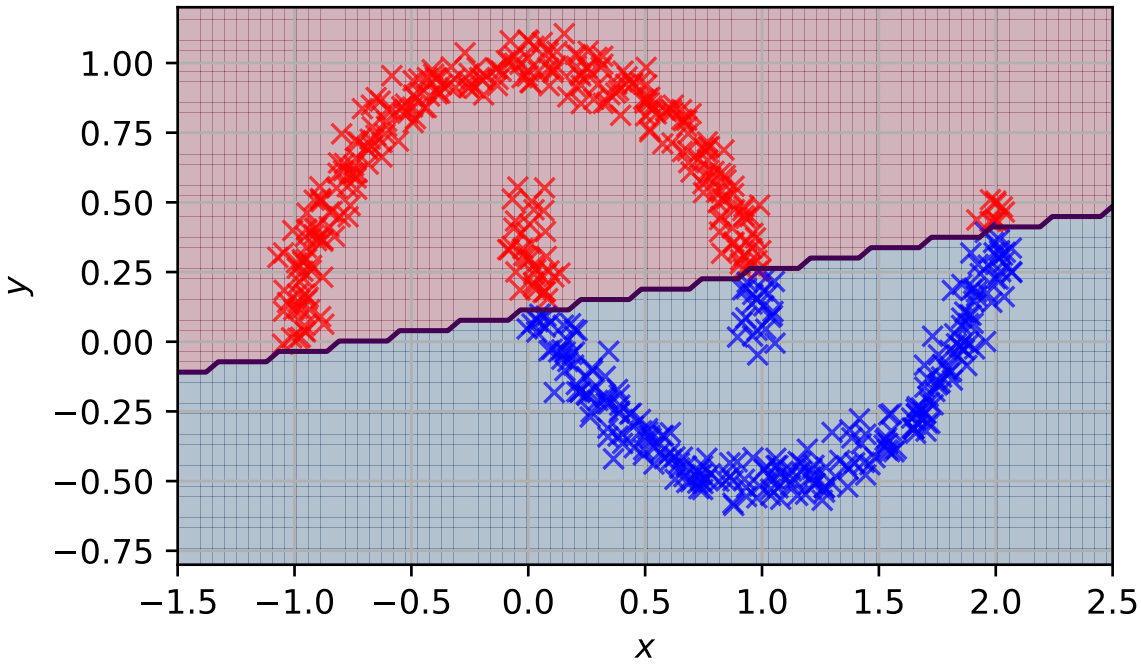


Figure: Logistic regression on moon data with default settings.

- above one is done with `SGDClassifier`

  → more options through `LogisticRegression`

Demonstration: penguins data

- SVM (with rbf) kernel on penguins data

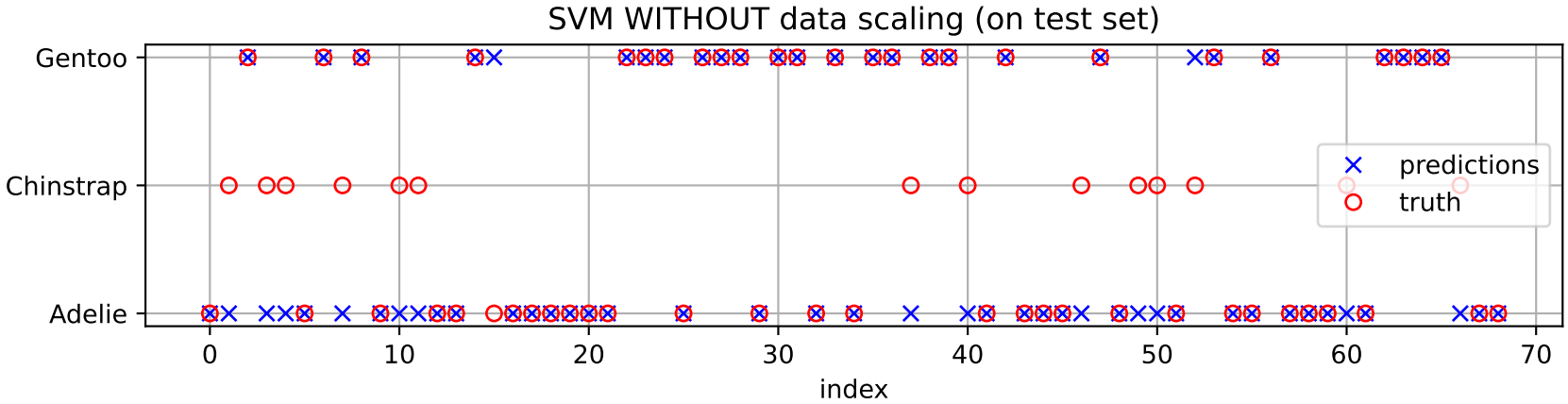
  $\rightarrow$ convert `species` labels to numerical values

  $\rightarrow$ 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



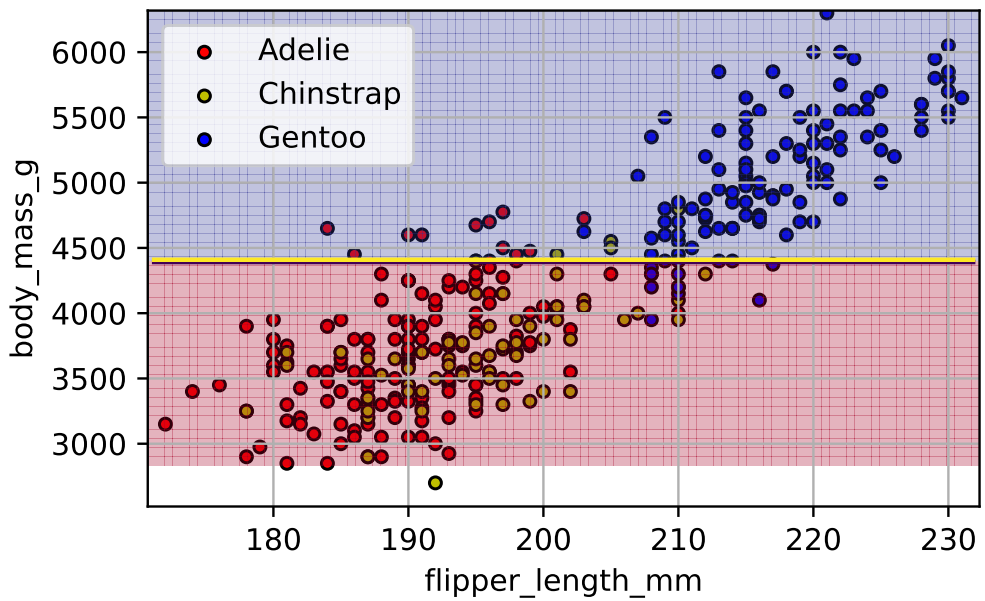


Figure: SVM (with rbf kernel) on **unscaled** penguins data using two features.

Demonstration: penguins data

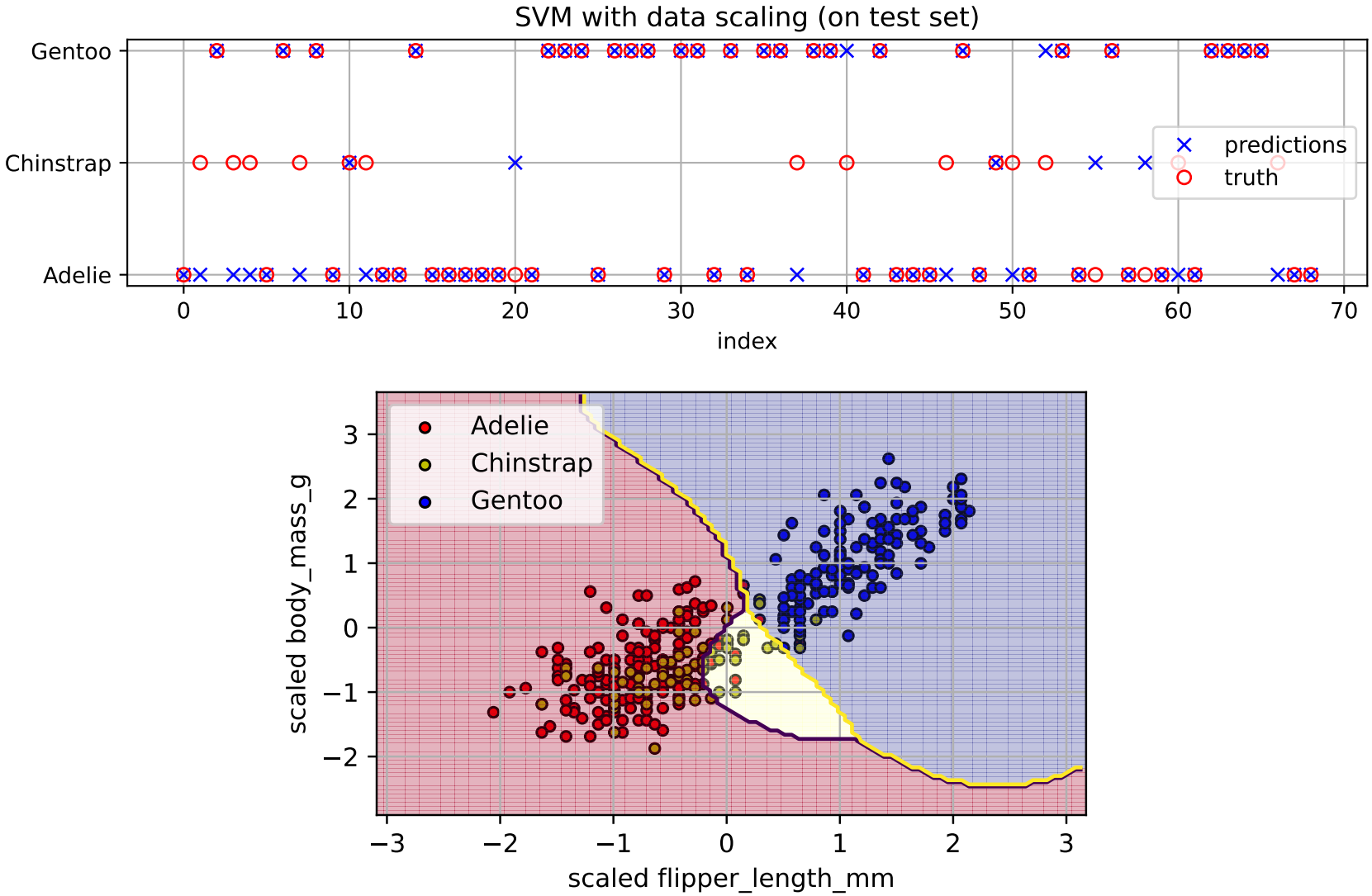


Figure: SVM (with rbf kernel) on scaled penguins data using two features.

Demonstration: penguins data

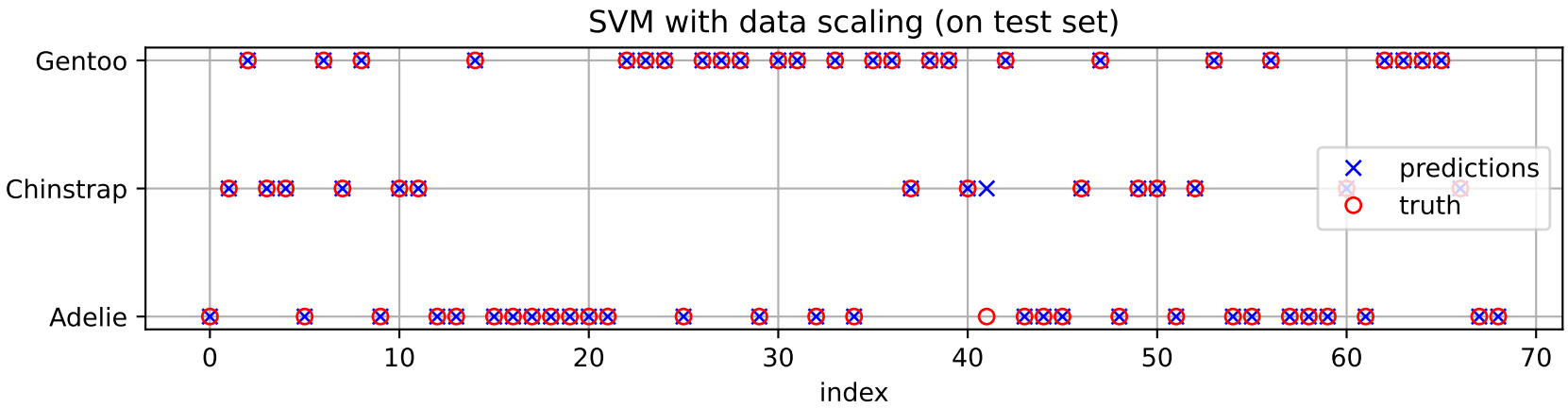


Figure: SVM (with rbf kernel) on scaled penguins data using all features.

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

Demonstration

- classification tasks as finding a separator between data

  $\rightarrow$ label prediction
- need to cross-validate and tune hyper-parameters accordingly!

- hard exercise: do image classification

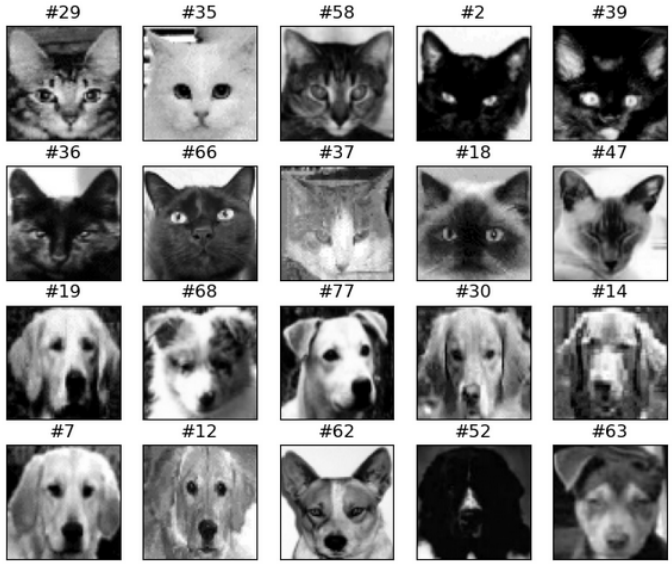


Figure: Selection of cats and dogs as a hard classification task.

Moving to a Jupyter notebook $\rightarrow$