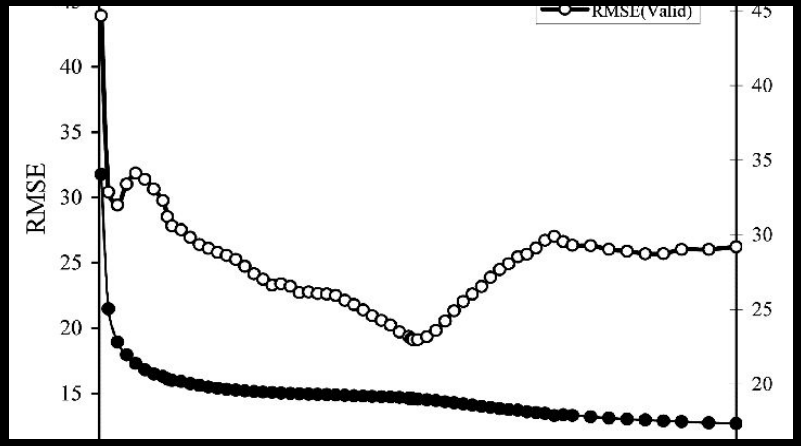
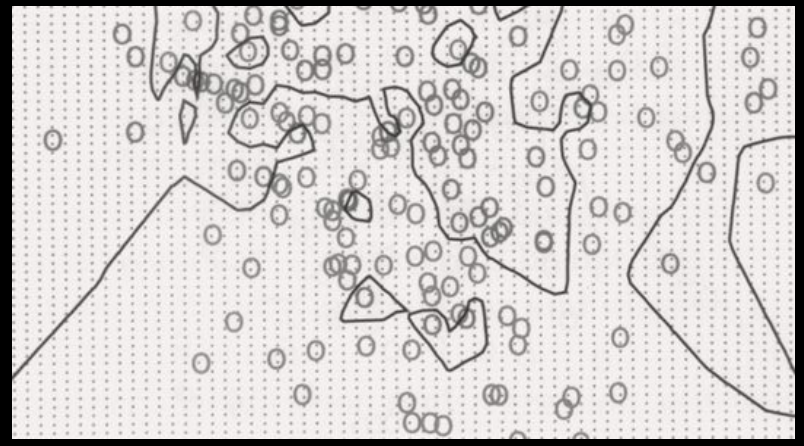
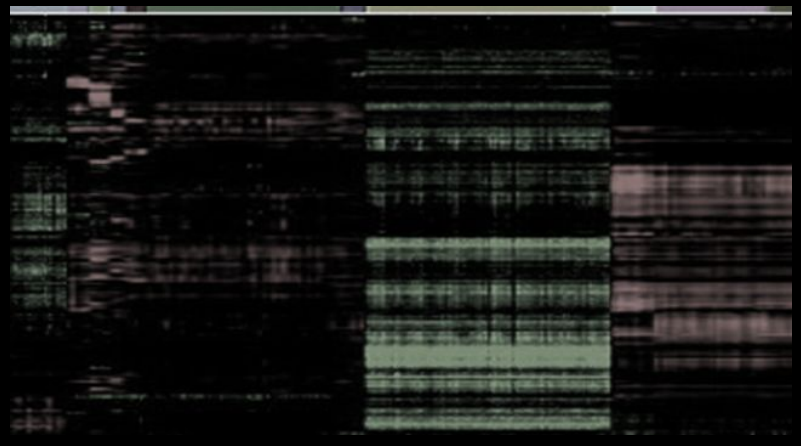
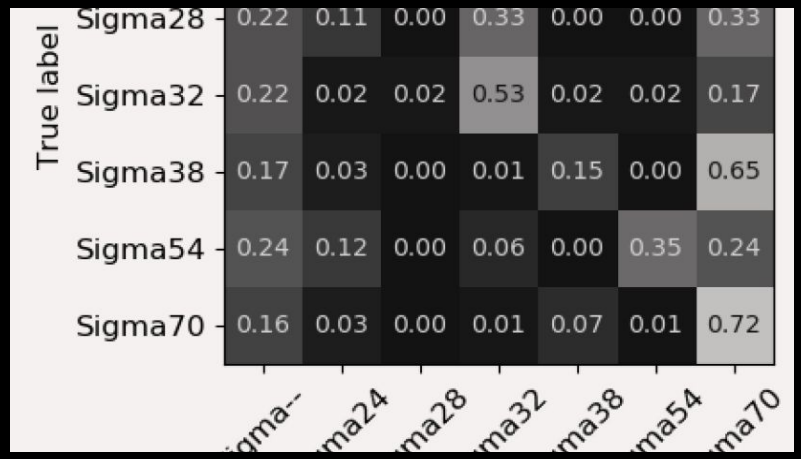
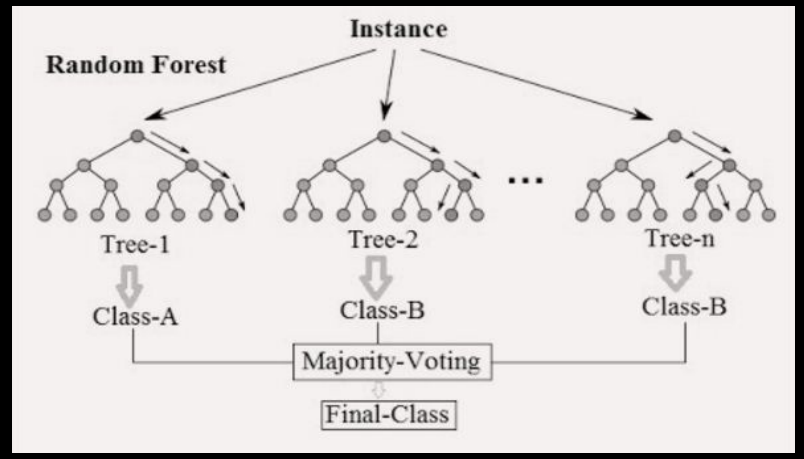
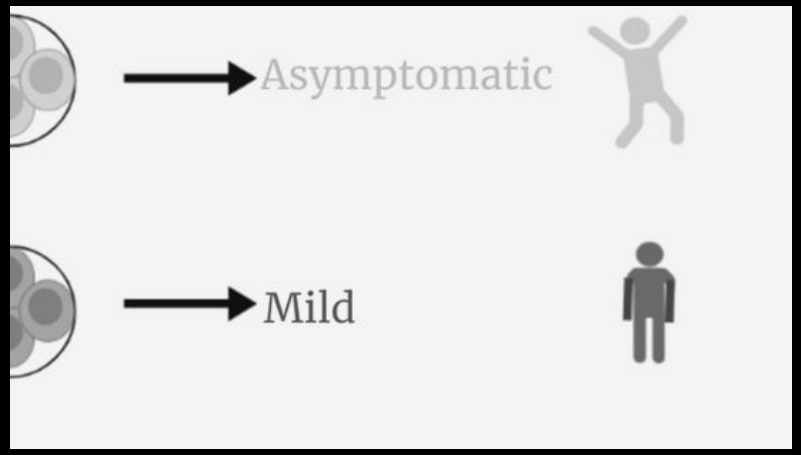












Module 4

Machine Learning

Machine Learning in Bioinformatics: The Idea

- Properties of biological systems at every scale are driven by many interacting factors
- These interactions are complex enough that basic statistical approaches (t-test, multiple regression, etc) are often inadequate
- ~~AI~~ Machine-learning methods can capture these interactions, and are shiny and fun

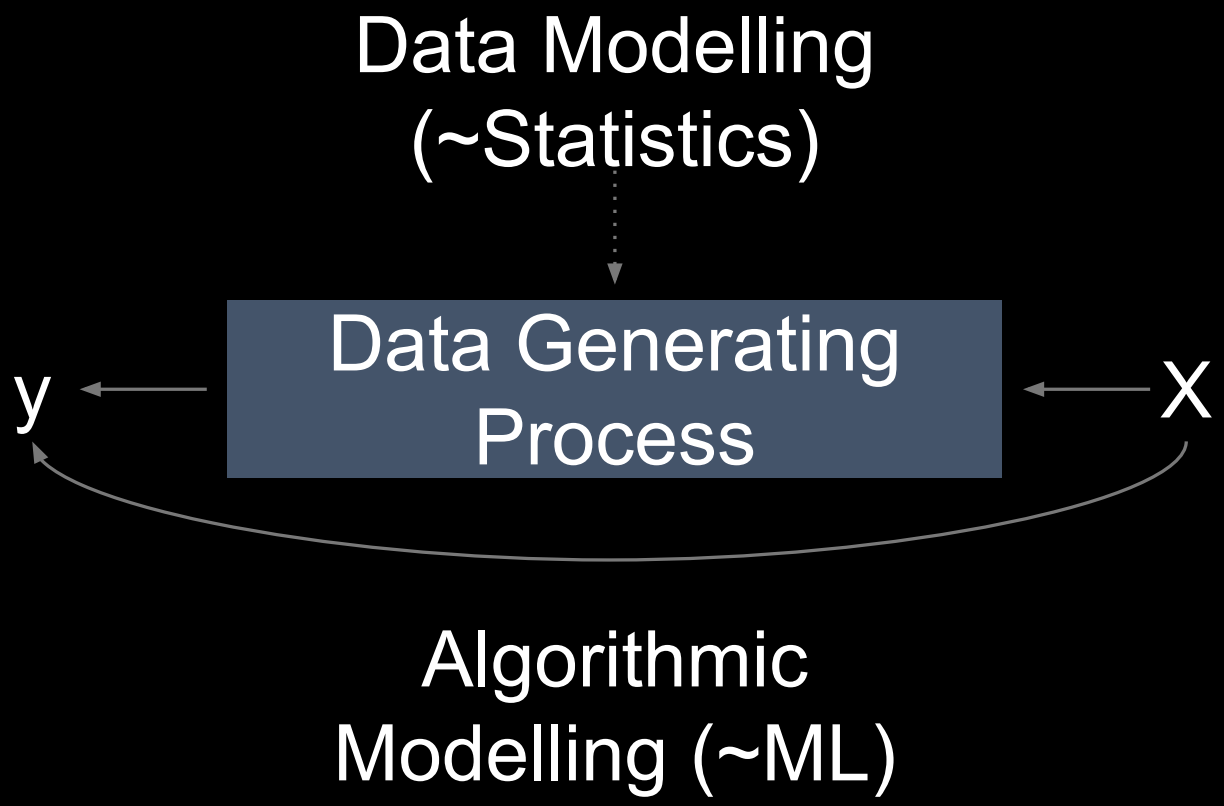
Machine Learning vs. Statistics

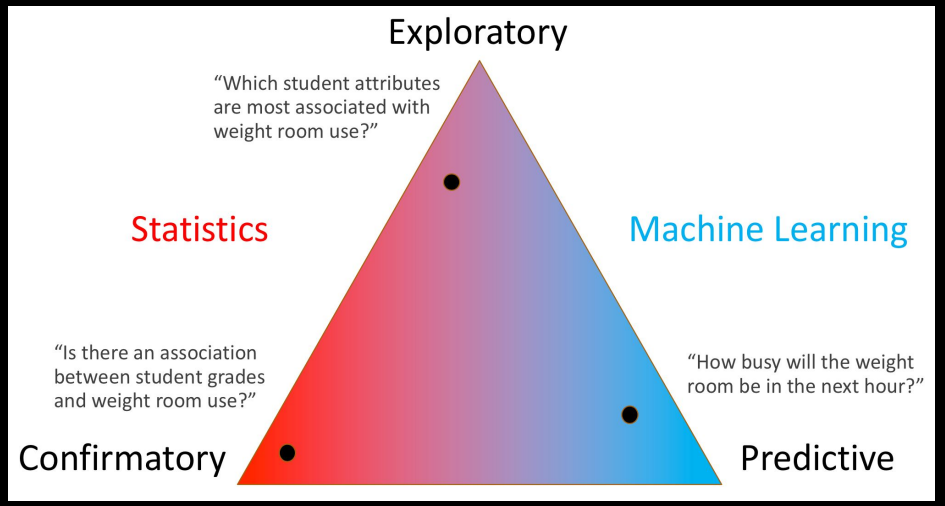
- Definitions and opinions differ, but the two are certainly not independent

- Statistics: Given a dataset, infer parameters that describe its properties and identify key relationships. These properties can be used to make predictions

- Machine Learning: Given a dataset, fit a (potentially very complex) function to generate accurate predictions without necessarily modeling the true causal parameters

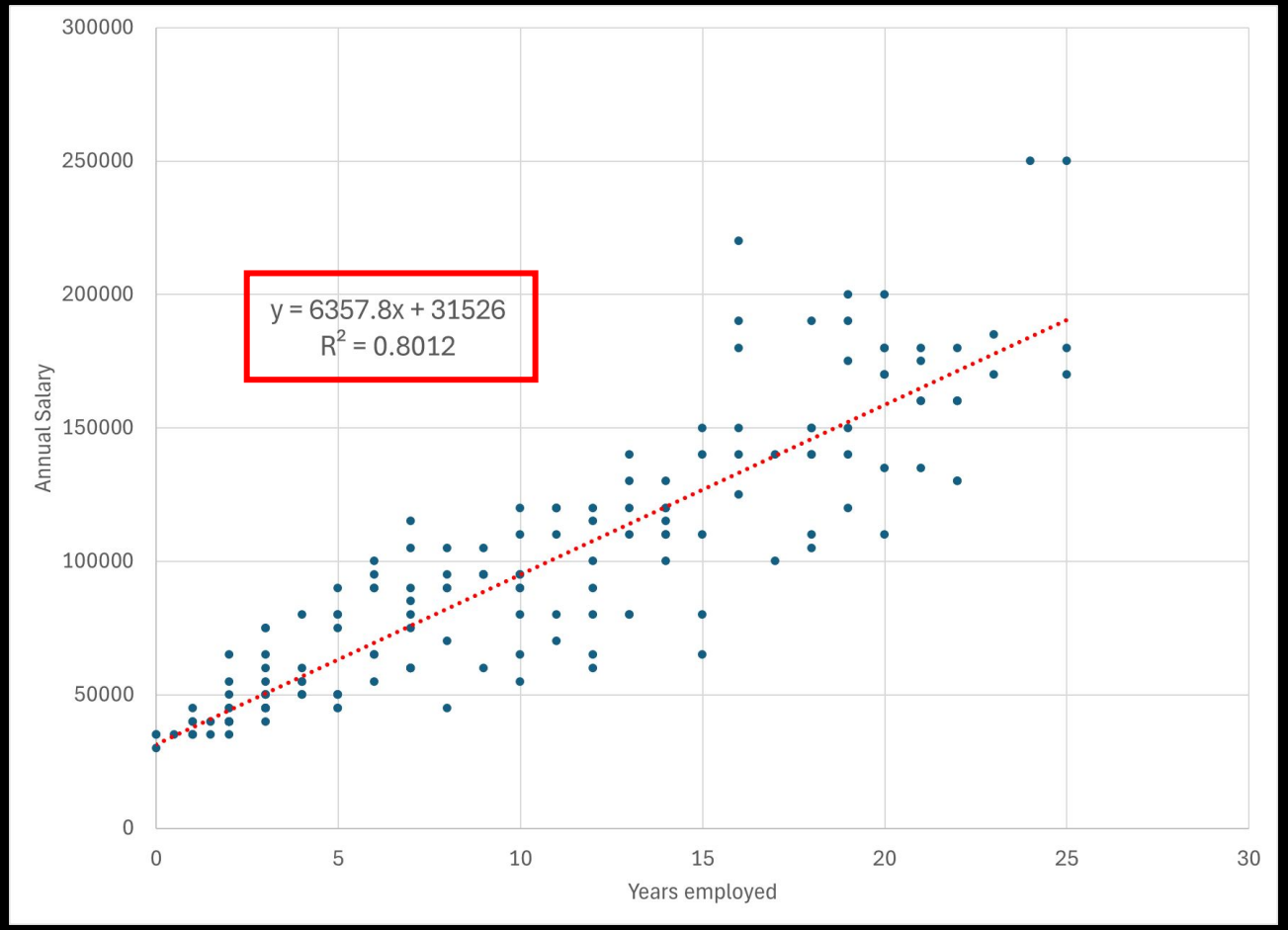
Machine Learning vs. Statistics





Example statistical method: linear regression

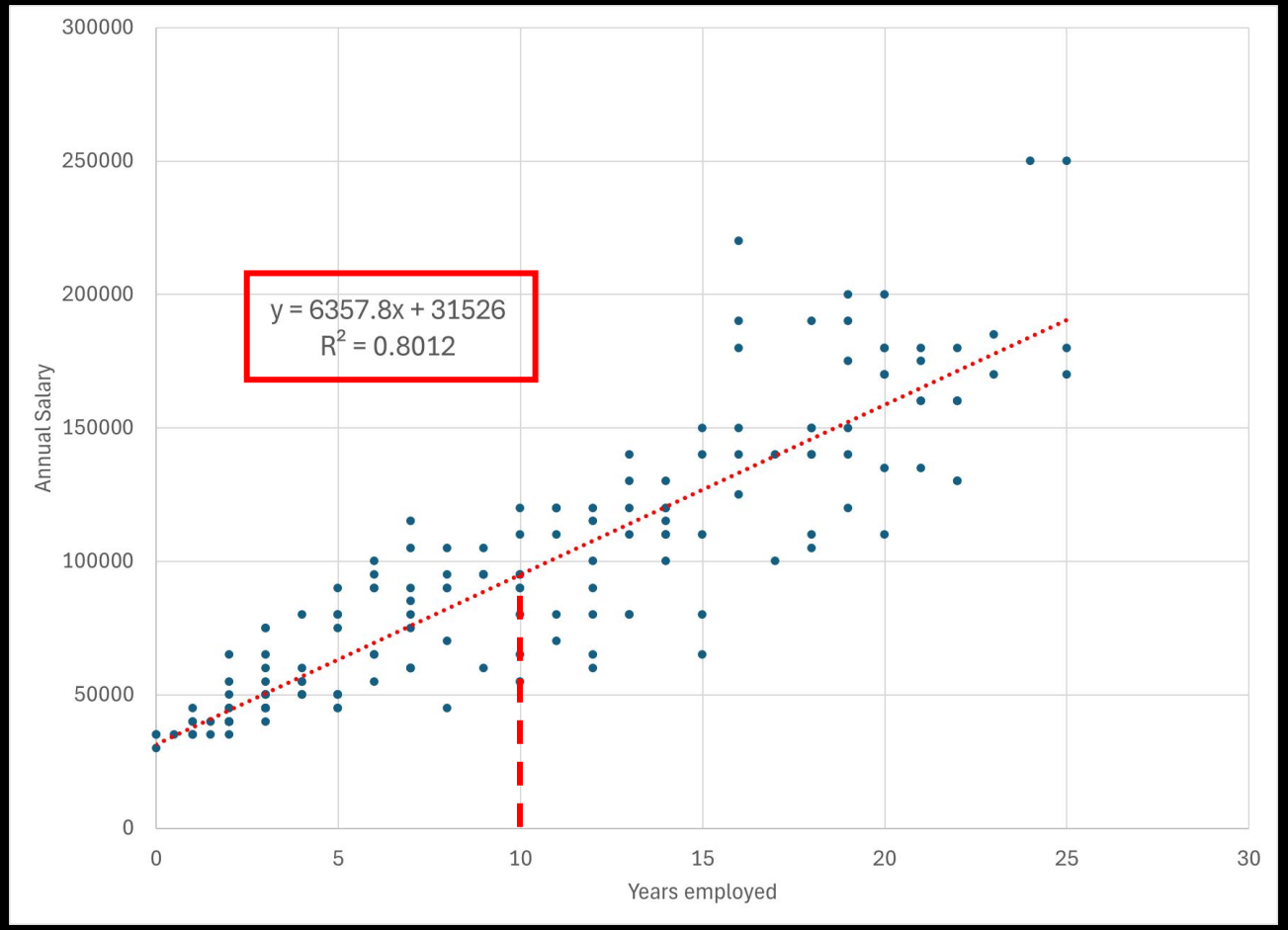
- Given a dataset, fit parameters that describe its properties and identify key relationships
- e.g., relationship between annual salary and years of employment – two parameters



https://www.geeksforgeeks.org/dataset-for-linear-regression/#salary-dataset

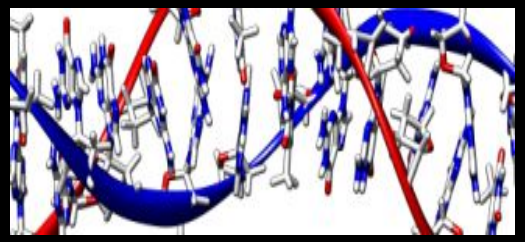
Example machine-learning method: linear regression

- Given a fitted regression model, predict an individual's salary based on their years of experience
- Experience = 10 years
- Predicted salary = $95,104



https://www.geeksforgeeks.org/dataset-for-linear-regression/#salary-dataset

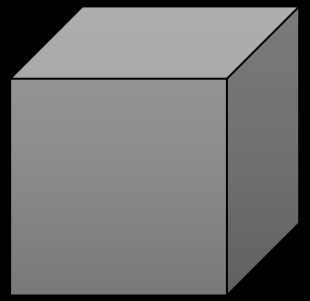
Machine Learning: The big picture



Source Data

(0.5,0.7,…)

Features
(Encoding)



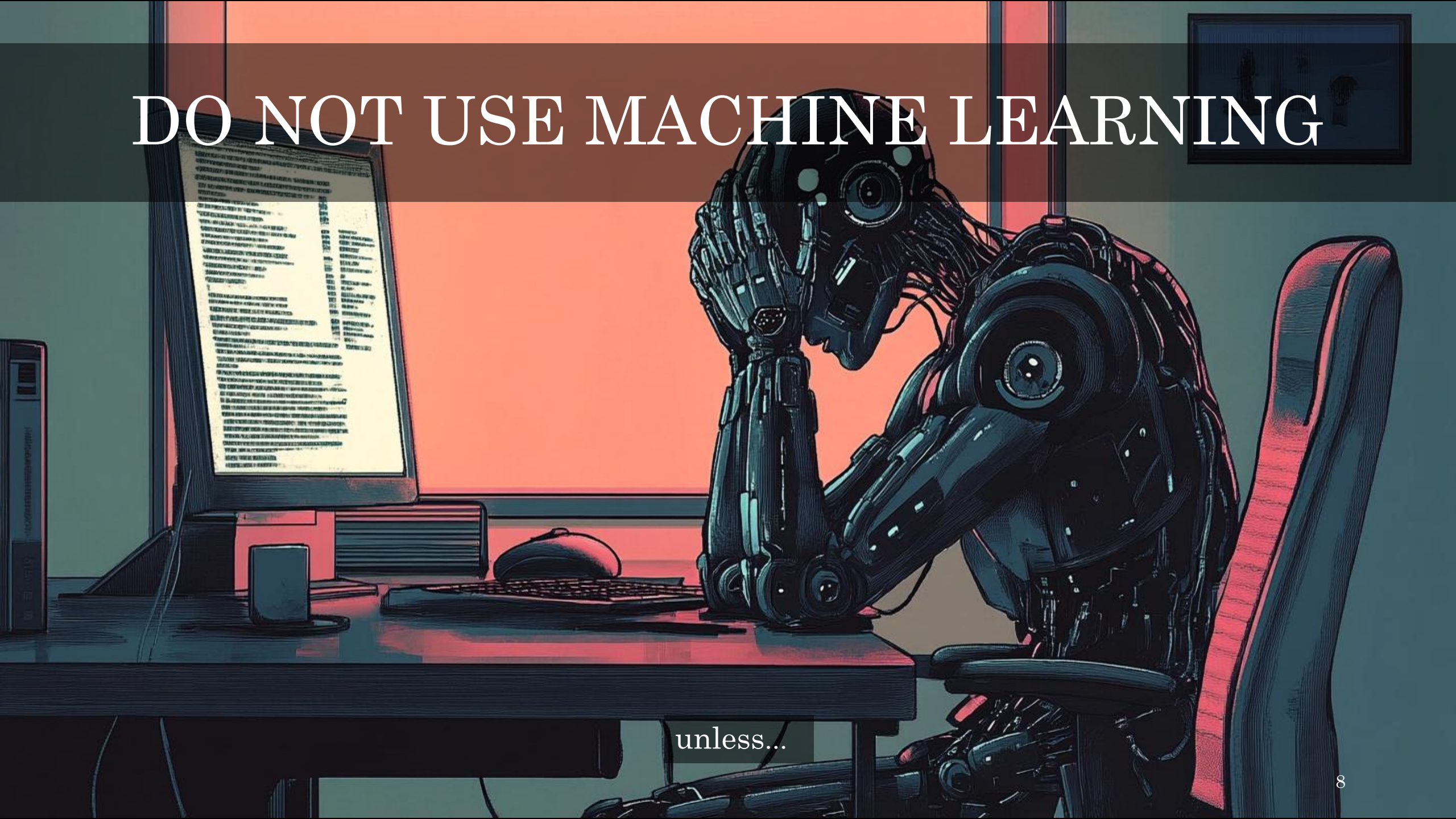
Classifier

Fitted (potentially complex) models

Insights into predictors

Predictions

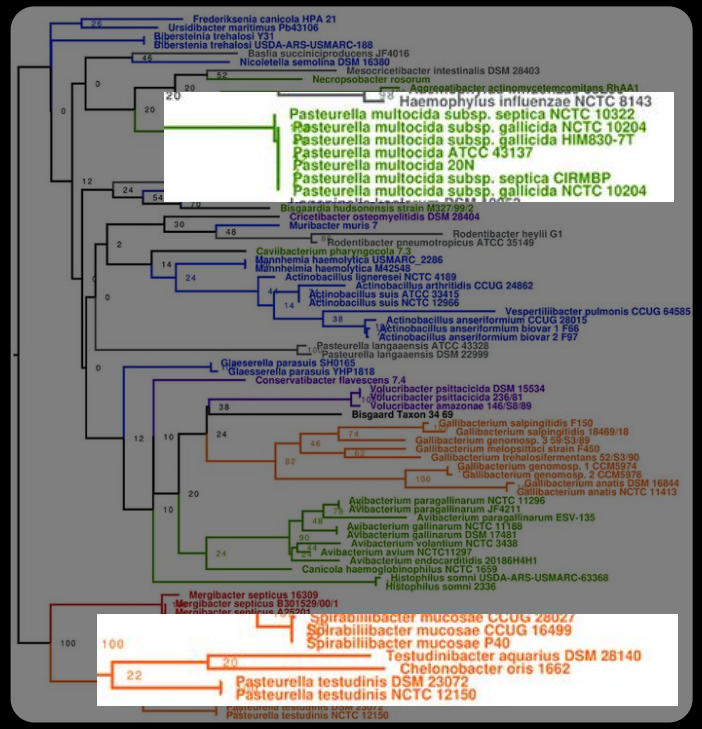
DO NOT USE MACHINE LEARNING



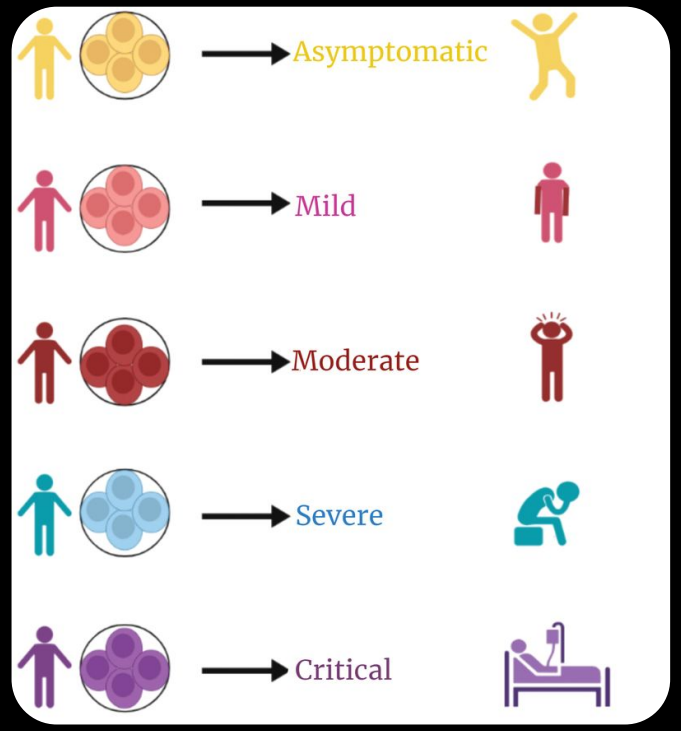
unless...

…unless you understand what you're predicting

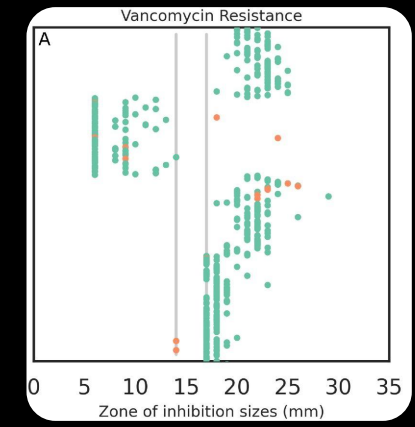
"Protein Structure"



"Species"

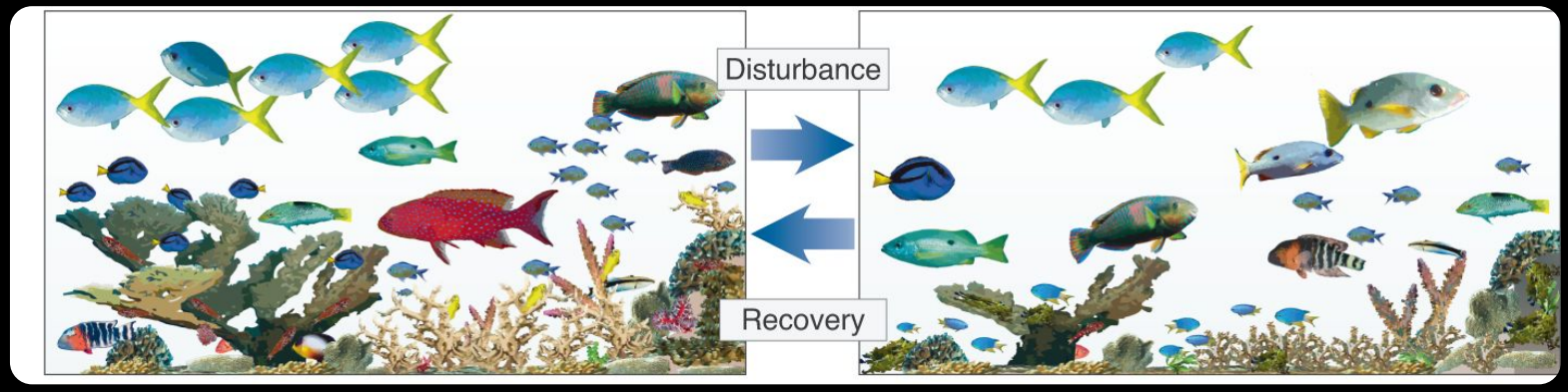


"Disease severity"



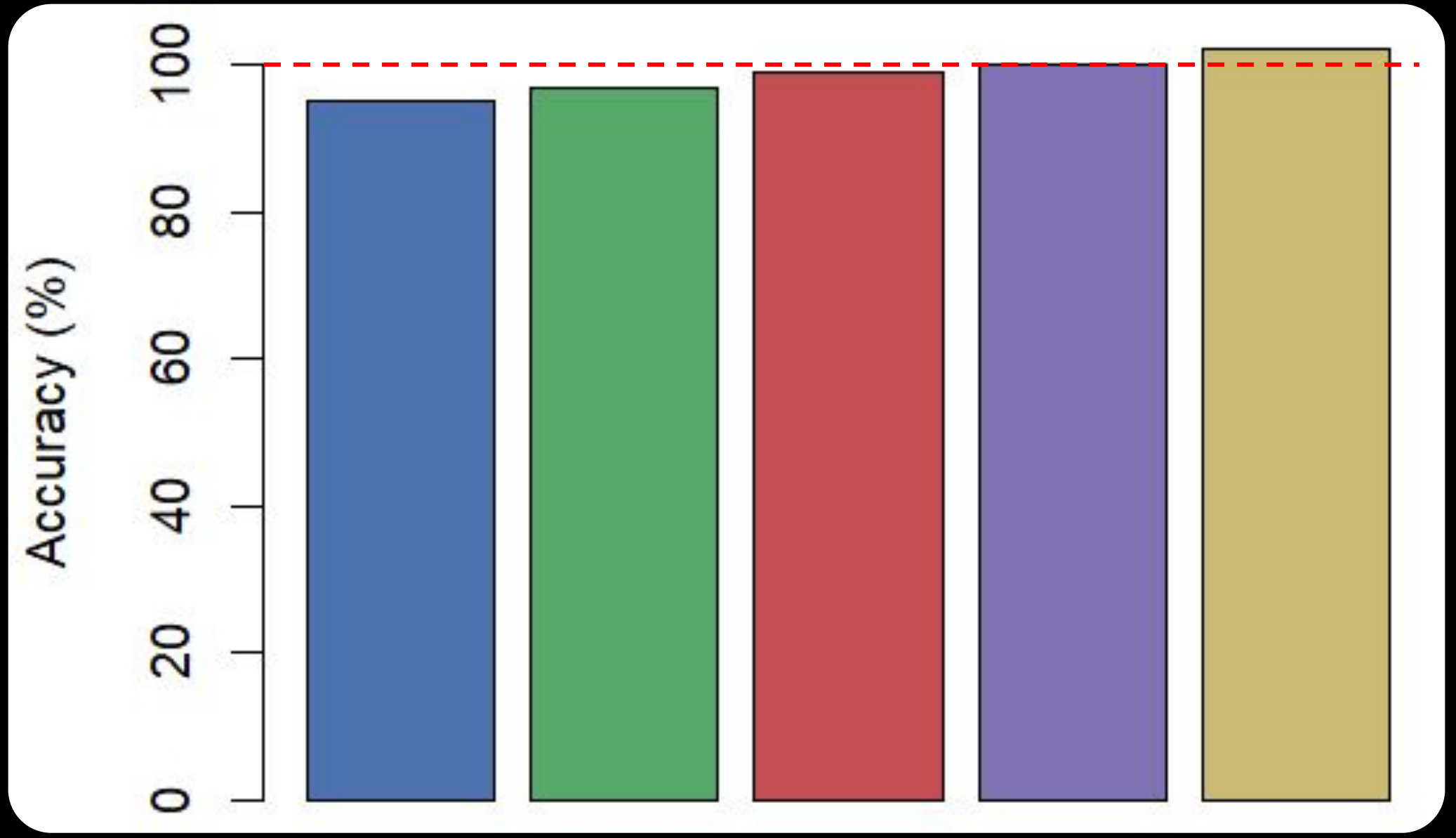
"Antimicrobial resistance"

"Habitat"



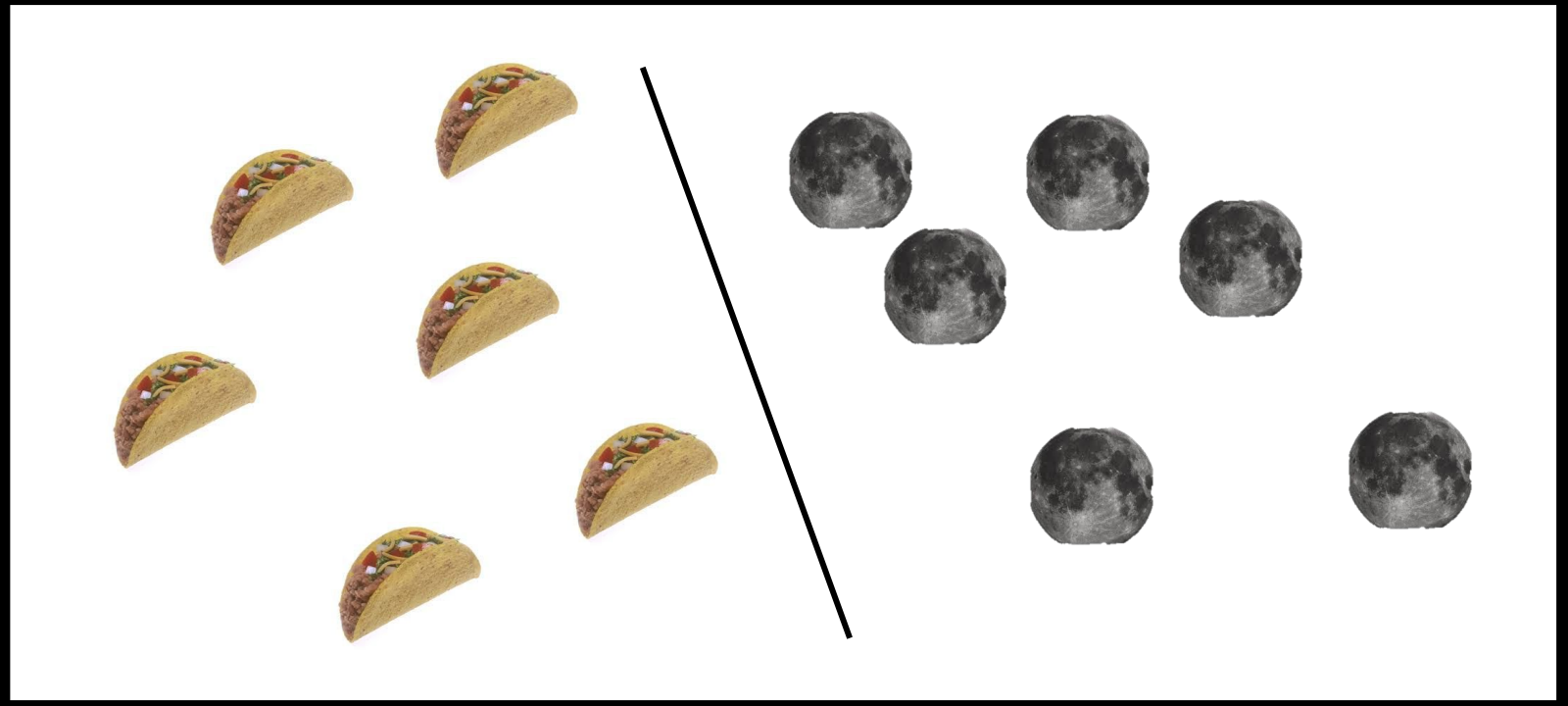
De Luca et al. (2021) *Frontiers in Microbiology*
Beger (2021) *Nature Ecology and Evolution*
Khare and Pandey (2022) *Frontiers in Immunology*
Kim et al. (2024) *Canadian Journal of Microbiology*

Unless you can see past the lies



Module Overview

- Training, testing, and evaluation: How do we manage biases and correctly interpret predictions?
- Feature representations: How do we turn biological data into a reasonable set of features we can feed to a classifier?
- ML methods and case studies

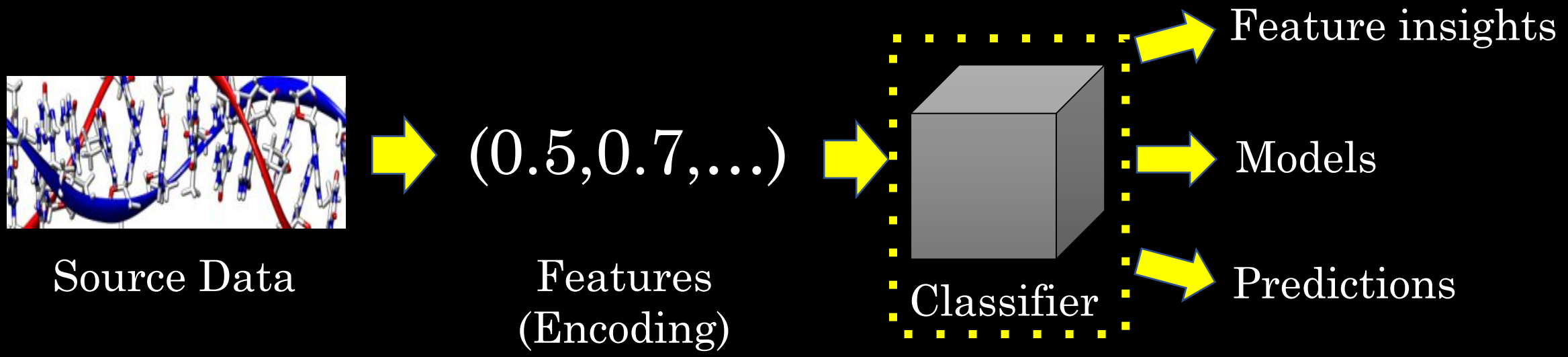


Module 4 Part 1: Introduction, Training, and Testing

Overview

1. General properties of learning problems
2. Training, testing and quantifying accuracy
3. Key criteria for choosing a classifier

The big picture



Types of learning problem

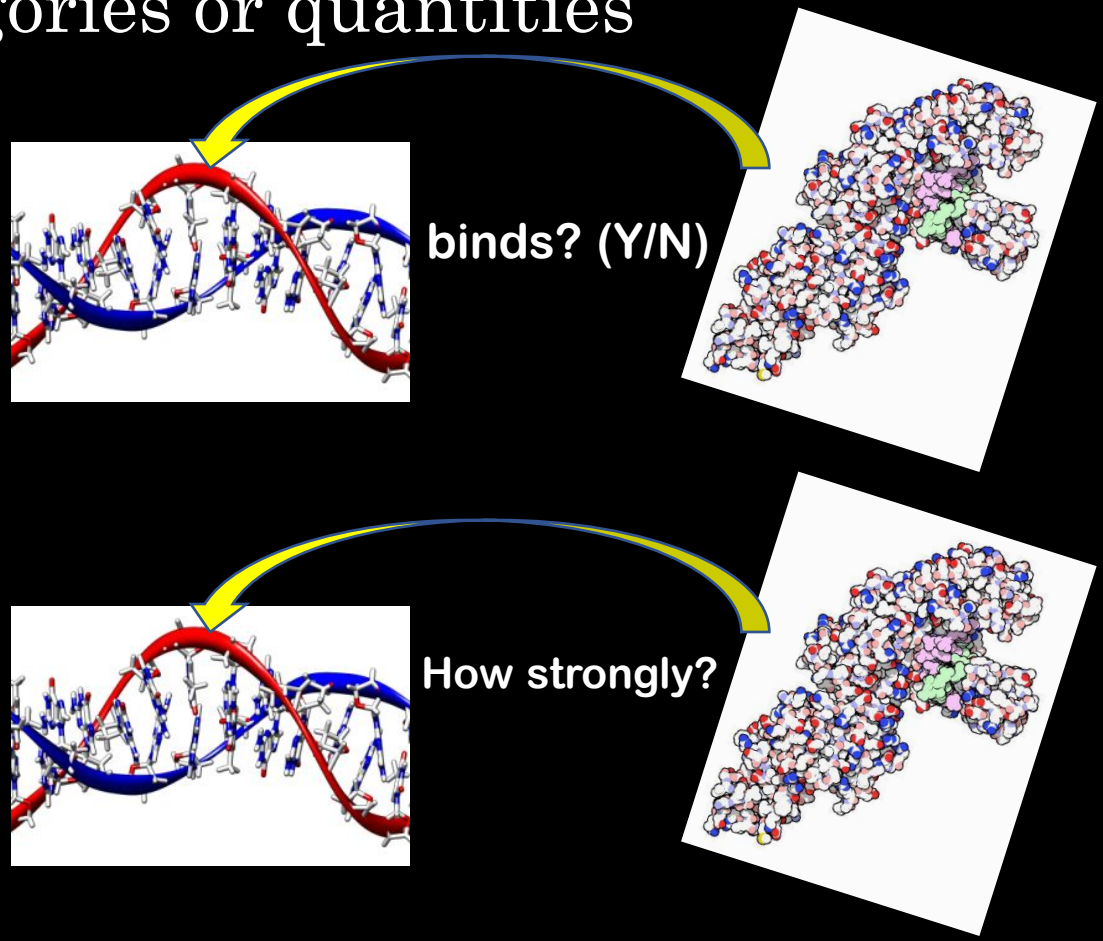
Map input variables to categories or quantities

CLASSIFICATION

Predict qualitative traits (categories)

REGRESSION

(and related methods)
Quantitative predictions



Types of Learning

SUPERVISED (our focus)

- Labeled classes
- Feedback: information about labeling is used to train classifier

UNSUPERVISED

- Classes may be labeled or unlabelled
- Classifier develops the classification scheme independently from class labels

SEMI-SUPERVISED

- Use both labeled and unlabeled data
- Unlabeled data can augment knowledge about probability distributions

REINFORCEMENT

- Identify optimal moves / strategies in a search space
- Good strategies are rewarded (consider short-term vs long-term tradeoffs)

Goal of Supervised Learning

Fit model parameters to minimize error on the training set

e.g., Squared error loss: $\mathrm{EPE}(f) = \mathrm{E}(\mathrm{Y} - f(\mathrm{X}))^2$

Expected prediction error

Difference between The Truth and model prediction

Decision boundaries and loss functions

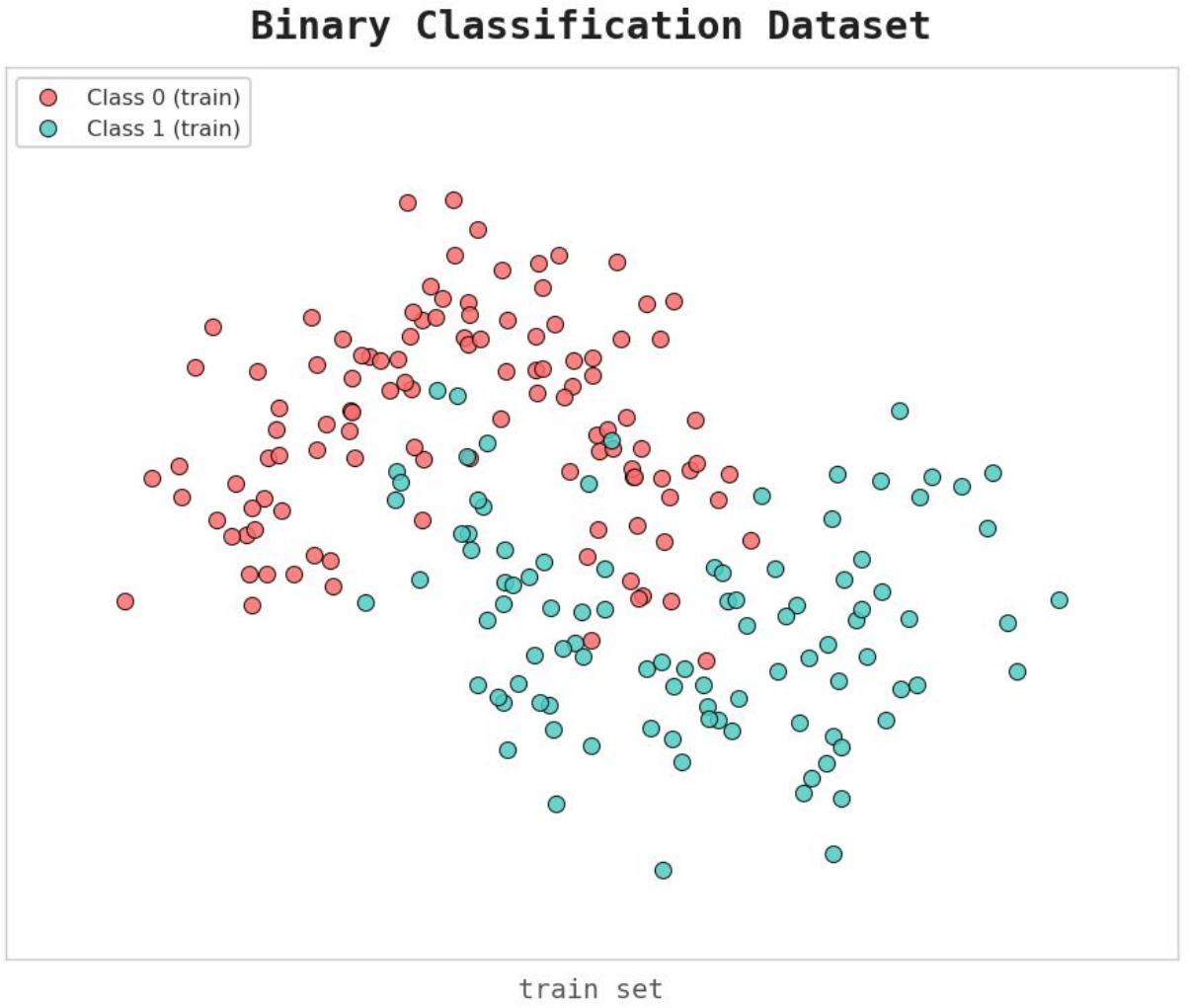
Find β so our function maps x to y

$$\hat{y} = f_\beta(x)$$

In other words: find β that defines boundary between labels

Minimising some loss function like binary cross-entropy (log-loss)

$$\mathcal{L}(\beta) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right]$$



Decision boundaries and loss functions

Find β so our function maps x to y

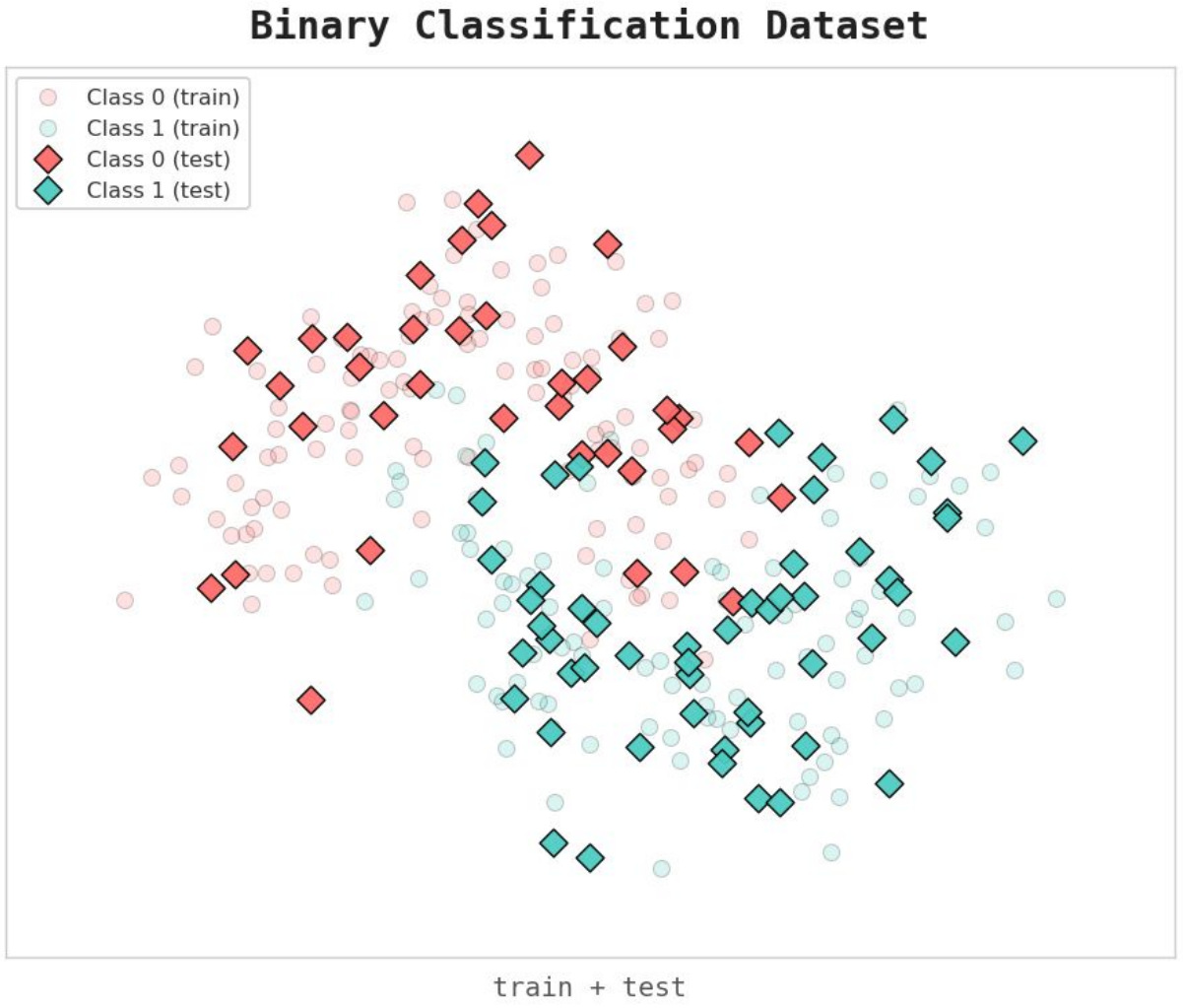
$$\hat{y} = f_\beta(x)$$

In other words: find β that defines boundary between labels

Minimising some loss function like binary cross-entropy (log-loss)

$$\mathcal{L}(\beta) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right]$$

Ideally in a way that generalises to new data!



Decision boundaries and loss functions

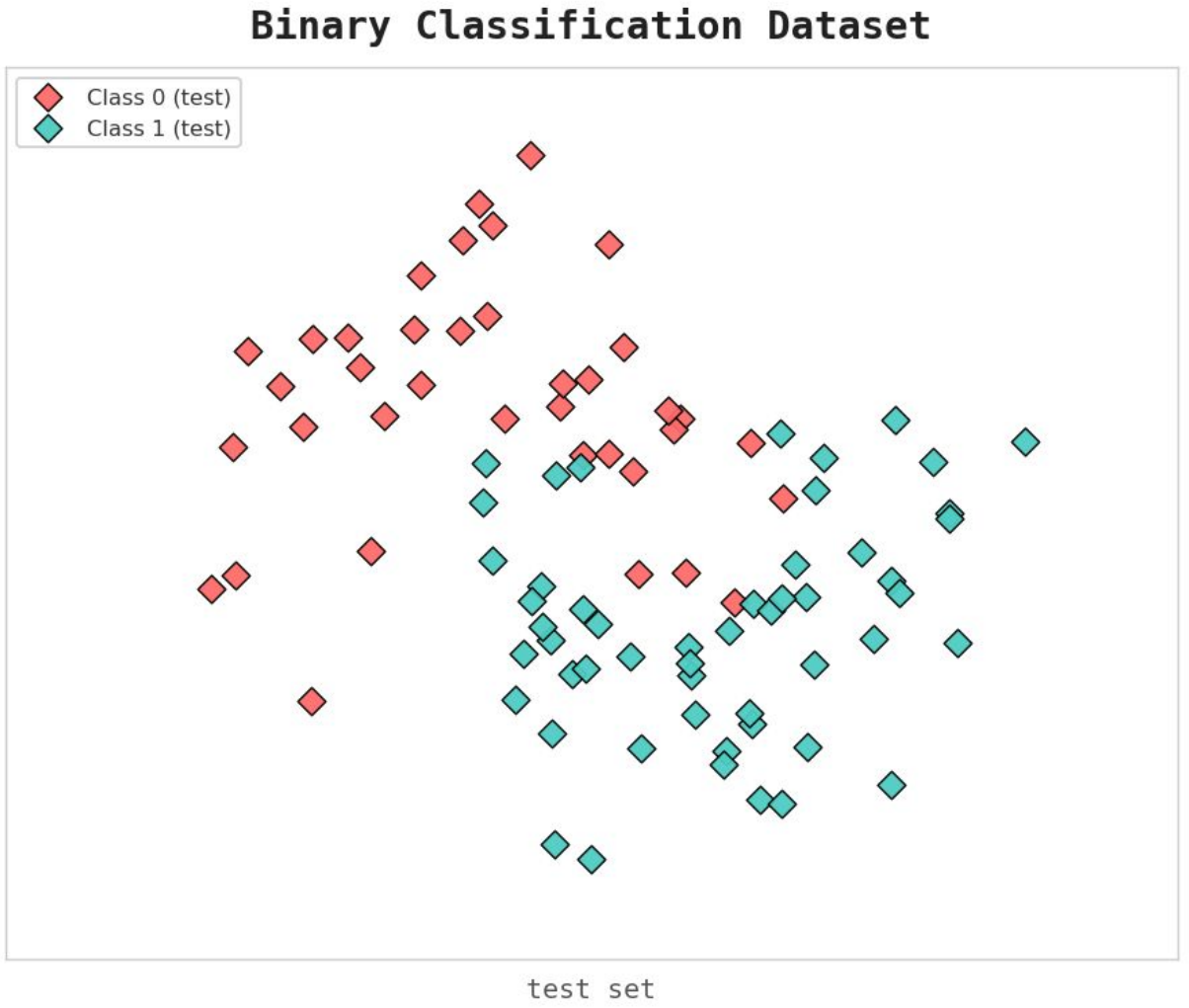
Find β so our function maps x to y

$$\hat{y} = f_\beta(x)$$

In other words: find β that defines boundary between labels

Minimising some loss function like binary cross-entropy (log-loss)

$$\mathcal{L}(\beta) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right]$$



Ideally in a way that generalises to new data!

Training

- How do we optimize the parameters of the model?

- Some methods have analytical solutions that are globally optimal on the cost function
  - e.g. linear regression, discriminant function analysis

- Others must use heuristics (iterative training, greedy approaches)
  - Artificial neural networks
  - Random forests
  - Support vector machines

- Differences between the two can be minor in some cases

What are the properties of a good training set?

- Rnodam spamle from the population
- Sufficiently large
- All classes represented

Generalization

- Typically in statistics we're interested in model parameters, not prediction
- But a classifier is of little use if it can only do well on data it has been trained on
- How well does the classifier handle cases that were not present in the training set?

Decision boundaries and loss functions

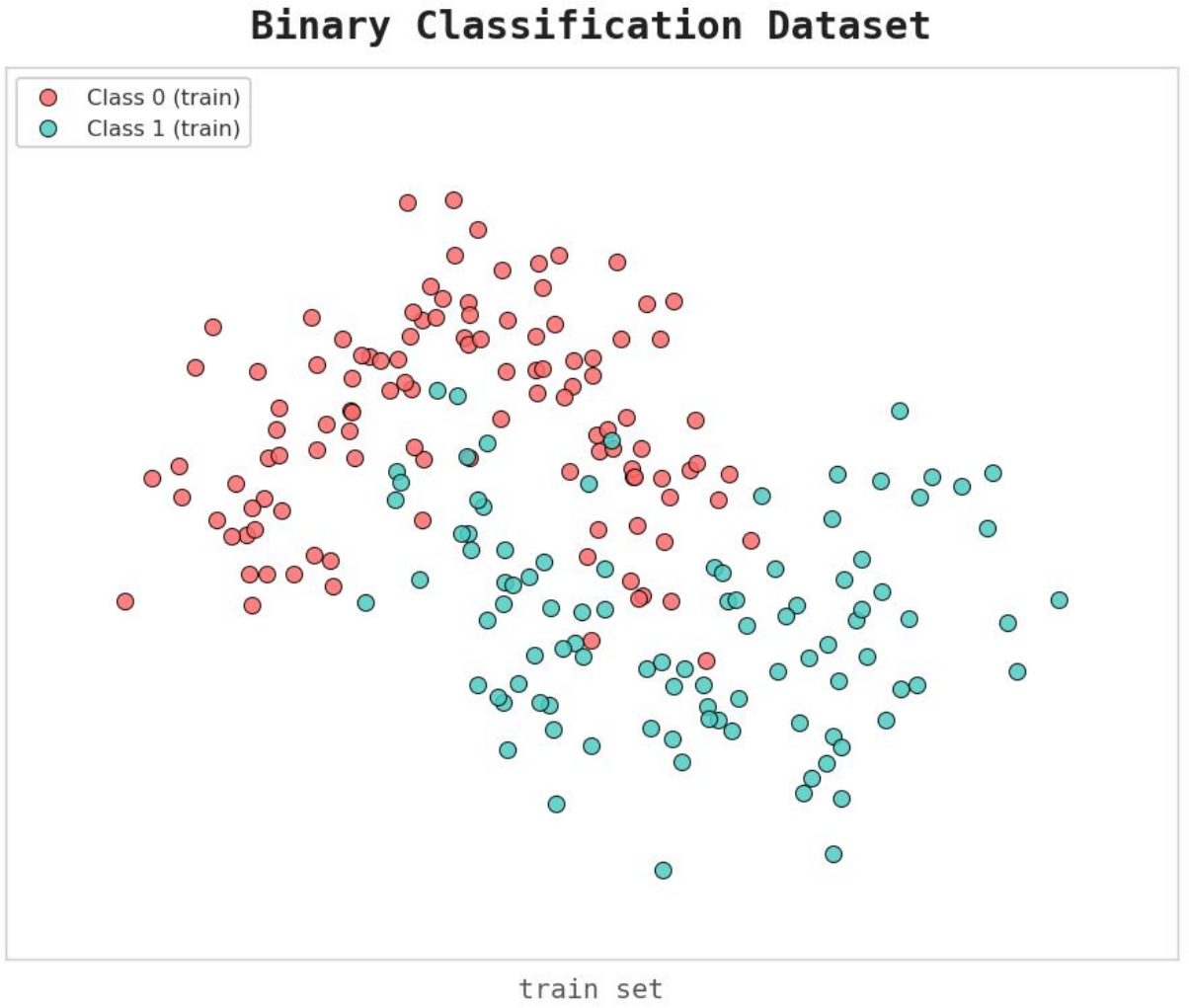
Find β so our function maps x to y

$$\hat{y} = f_\beta(x)$$

In other words: find β that defines boundary between labels

Minimising some loss function like binary cross-entropy (log-loss)

$$\mathcal{L}(\beta) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right]$$



Decision boundaries and loss functions

Find β so our function maps x to y

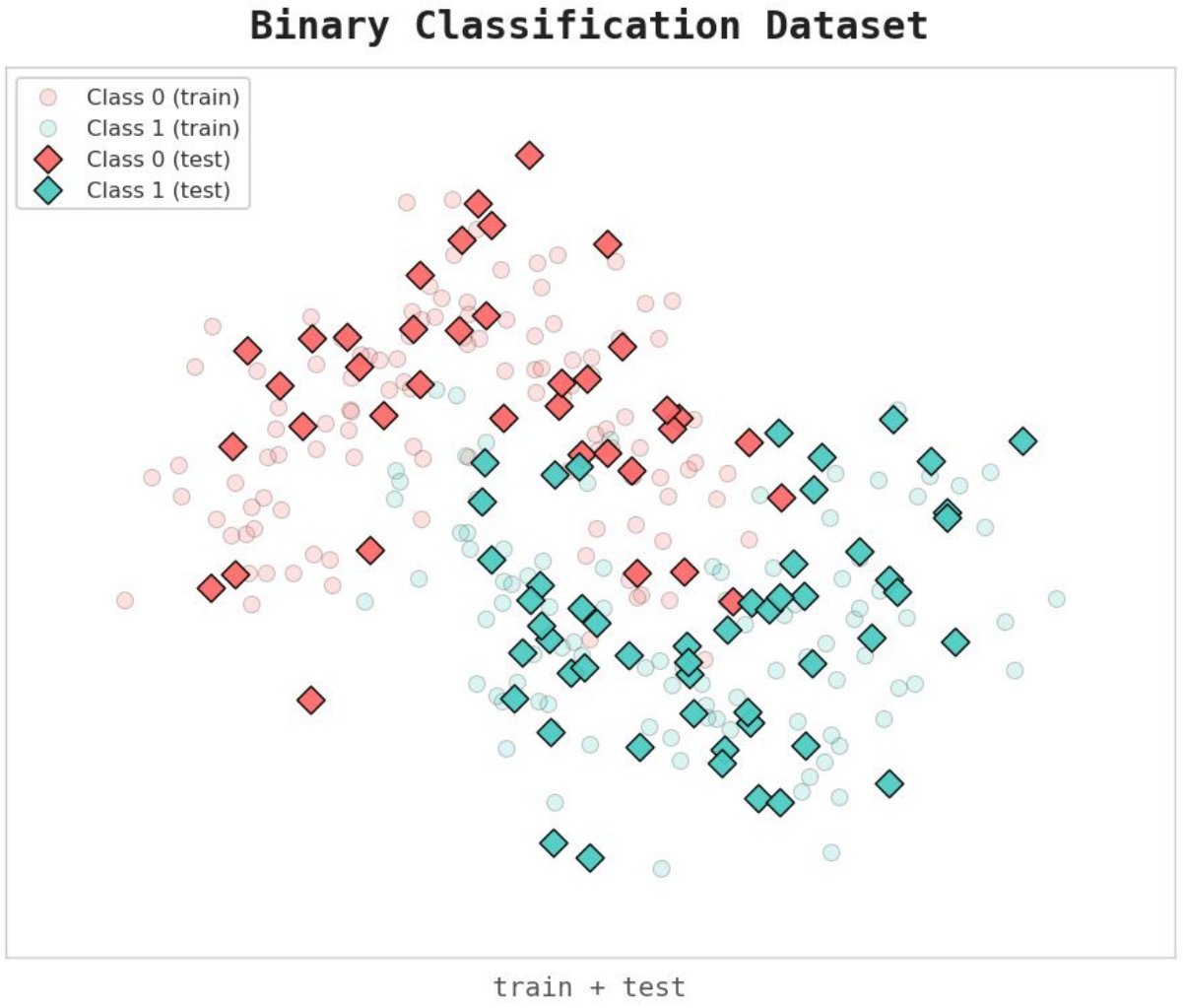
$$\hat{y} = f_\beta(x)$$

In other words: find β that defines boundary between labels

Minimising some loss function like binary cross-entropy (log-loss)

$$\mathcal{L}(\beta) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right]$$

Ideally in a way that generalises to new data!



Decision boundaries and loss functions

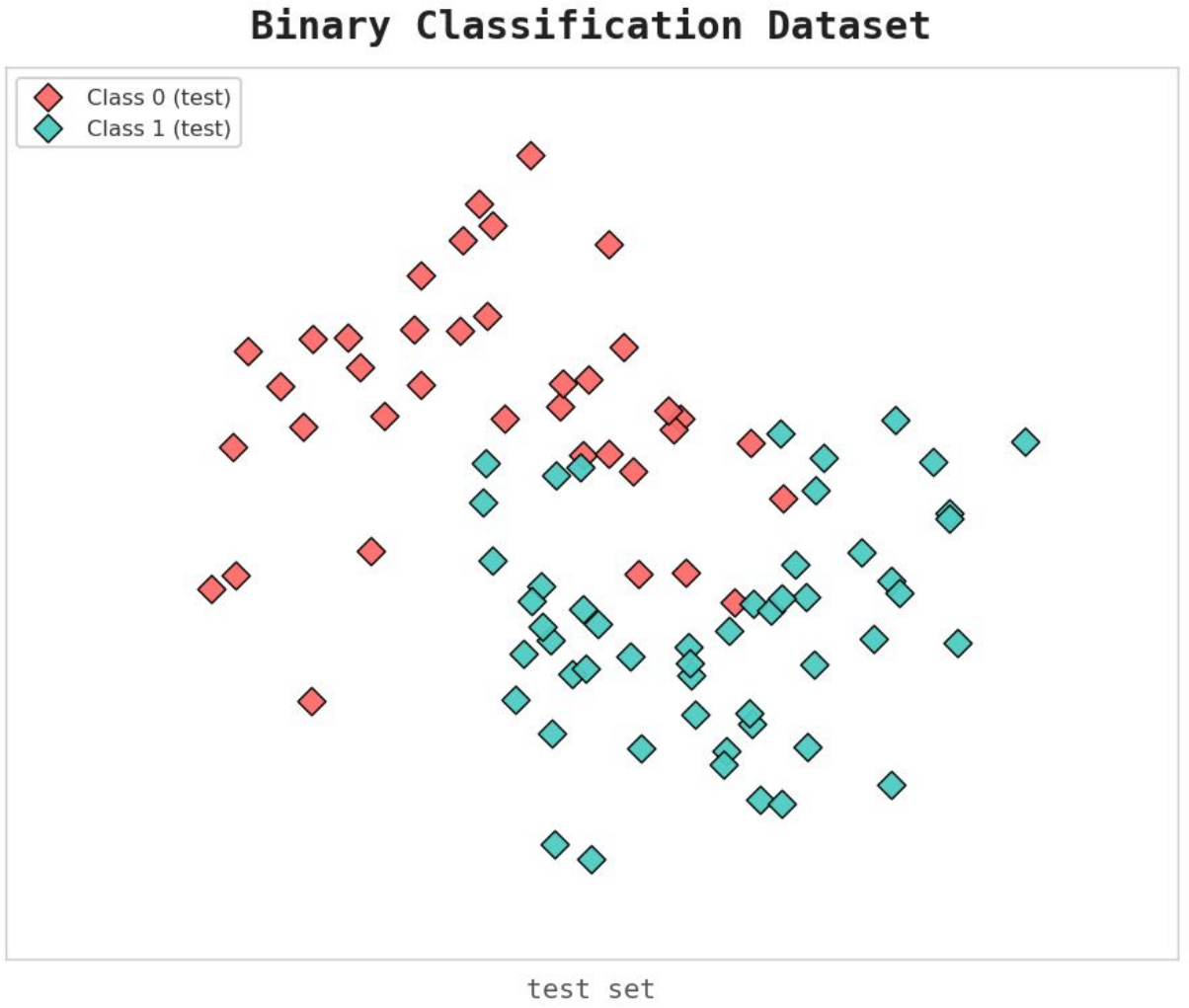
Find β so our function maps x to y

$$\hat{y} = f_\beta(x)$$

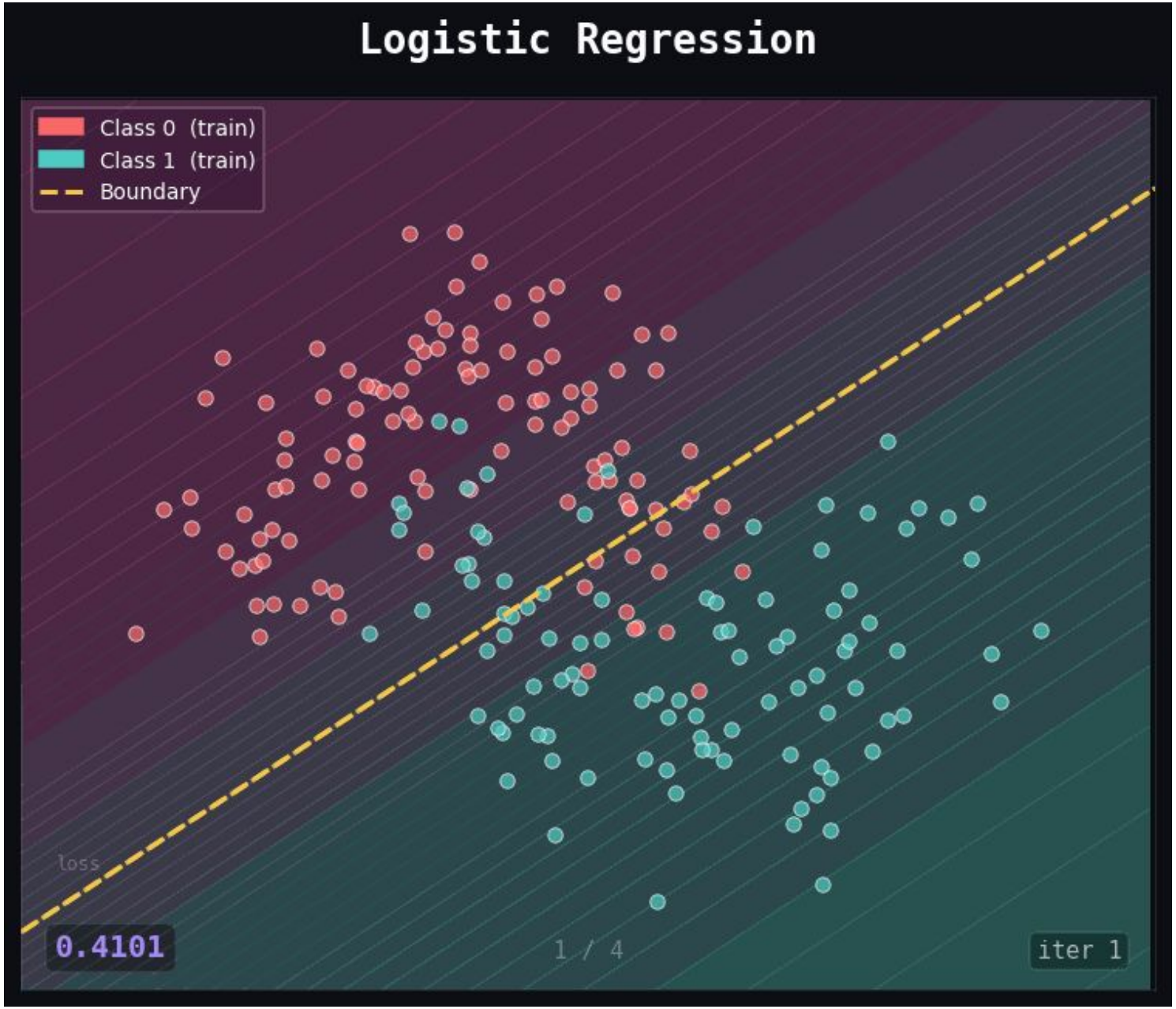
In other words: find β that defines boundary between labels

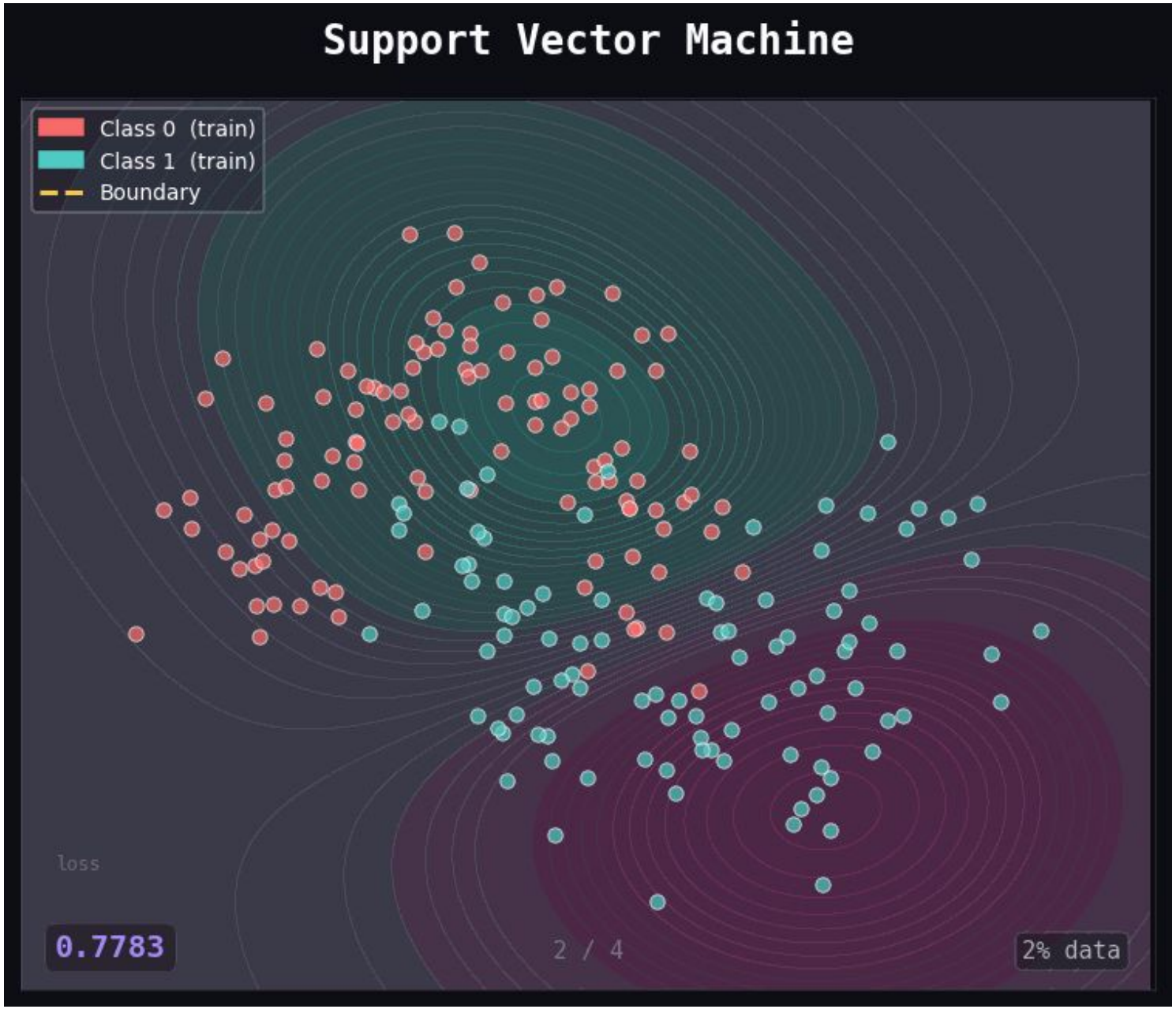
Minimising some loss function like binary cross-entropy (log-loss)

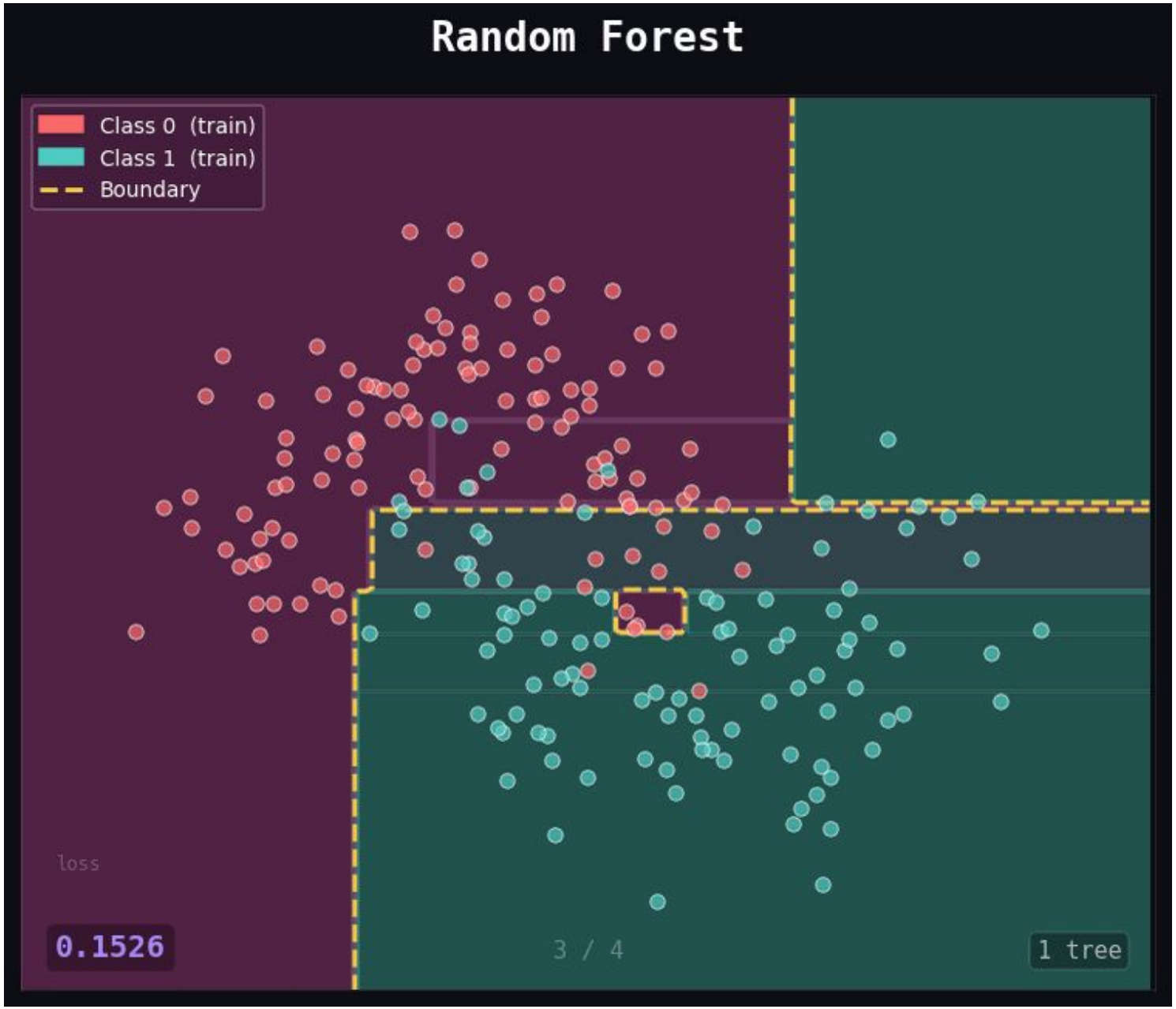
$$\mathcal{L}(\beta) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right]$$

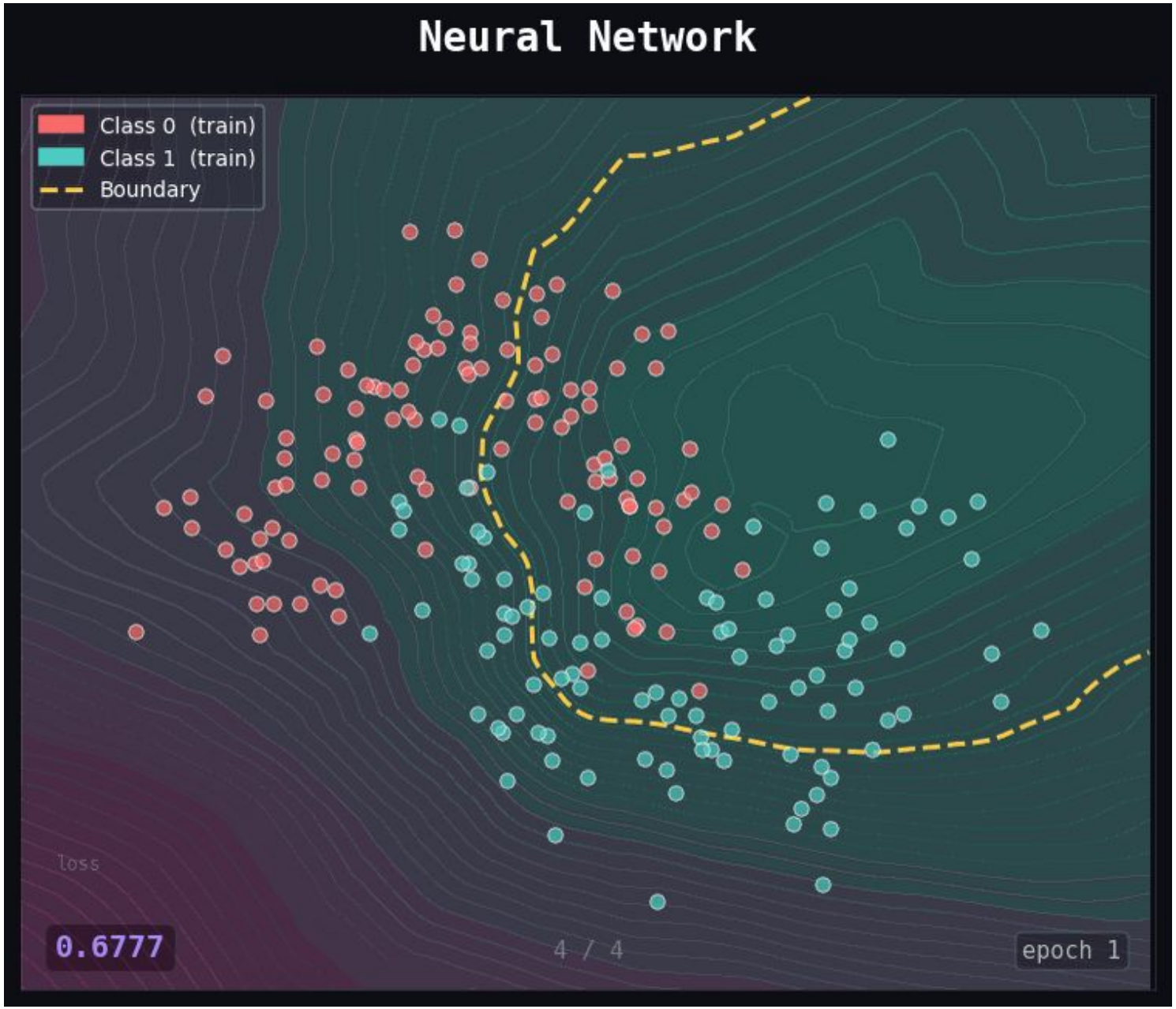


Ideally in a way that generalises to new data!


Logistic Regression
Class 0 (train)
Class 1 (train)
Boundary
loss
0.4101
1 / 4
iter 1

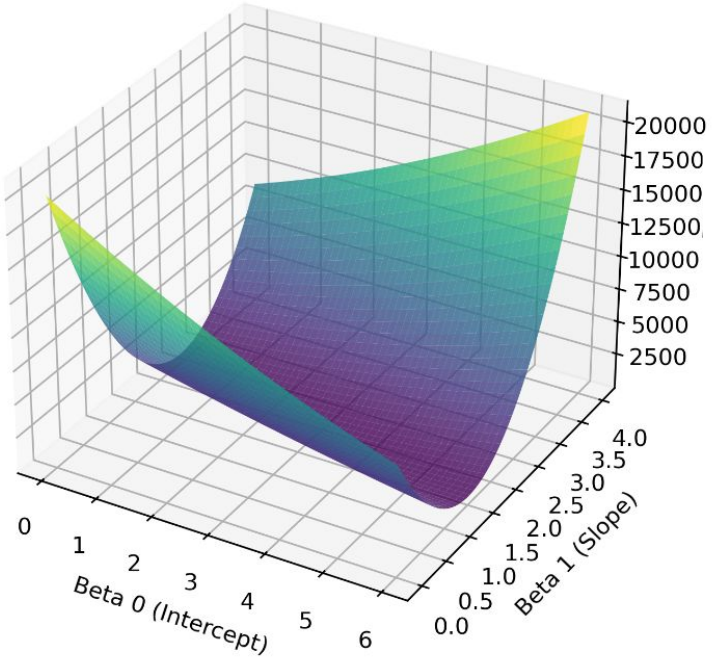

Support Vector Machine
Class 0 (train)
Class 1 (train)
Boundary
loss
0.7783
2 / 4
2% data

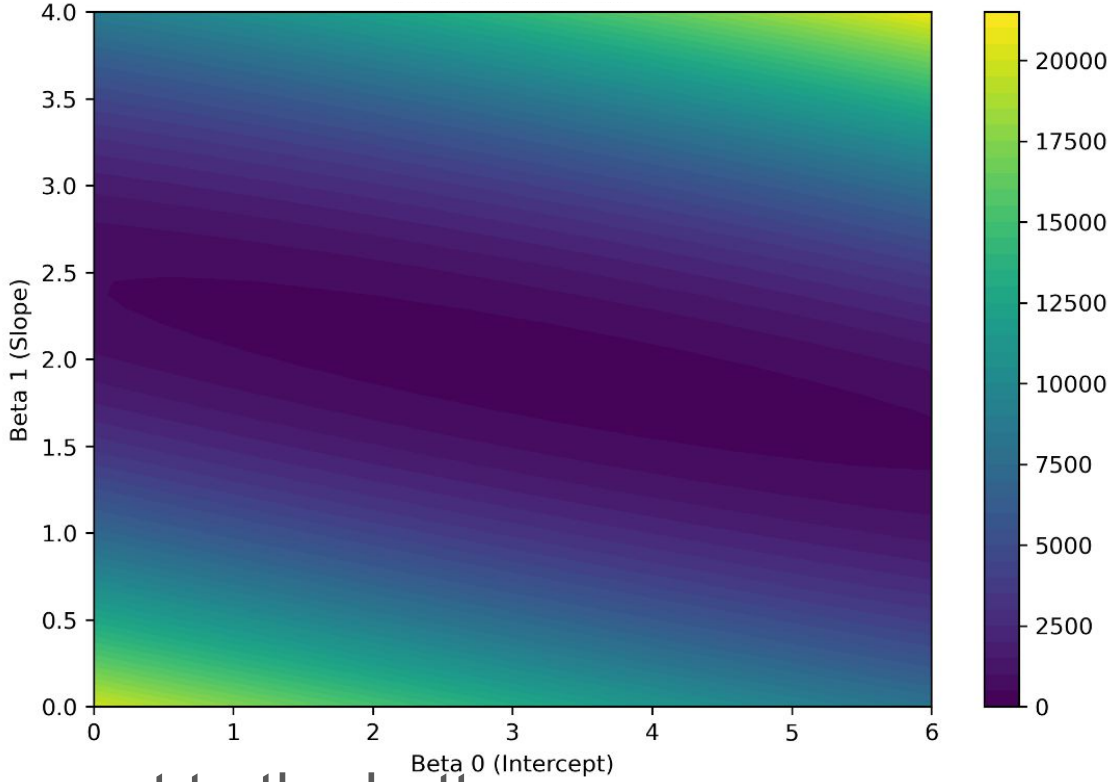

Random Forest
Class 0 (train)
Class 1 (train)
Boundary
loss
0.1526
3 / 4
1 tree


Neural Network
Class 0 (train)
Class 1 (train)
Boundary
loss
0.6777
4 / 4
epoch 1

Relationship between loss and model parameters

$$\hat{y} = \frac{1}{1 + e^{-(\beta^\top \mathbf{x})}}$$





Minimise loss: go down this surface until we get to the bottom.

$$\mathcal{L}(\beta) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)]$$

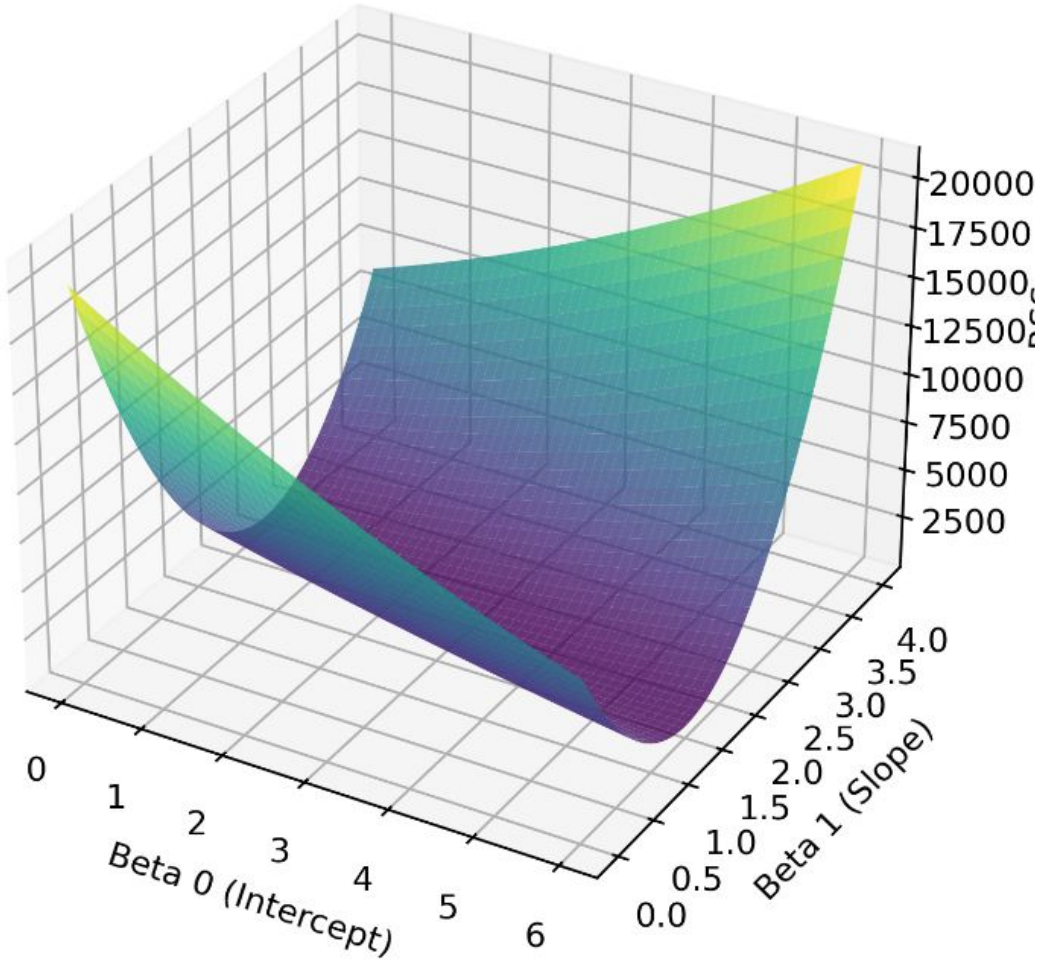
But how do we update model parameters to achieve this?

Gradient descent using partial derivatives of loss

Calculate partial derivative of:

$$\mathcal{L}(\beta) = -\frac{1}{N}\sum_{i=1}^{N} [y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i)]$$

with respect to each β parameter:



Gradient descent using partial derivatives of loss

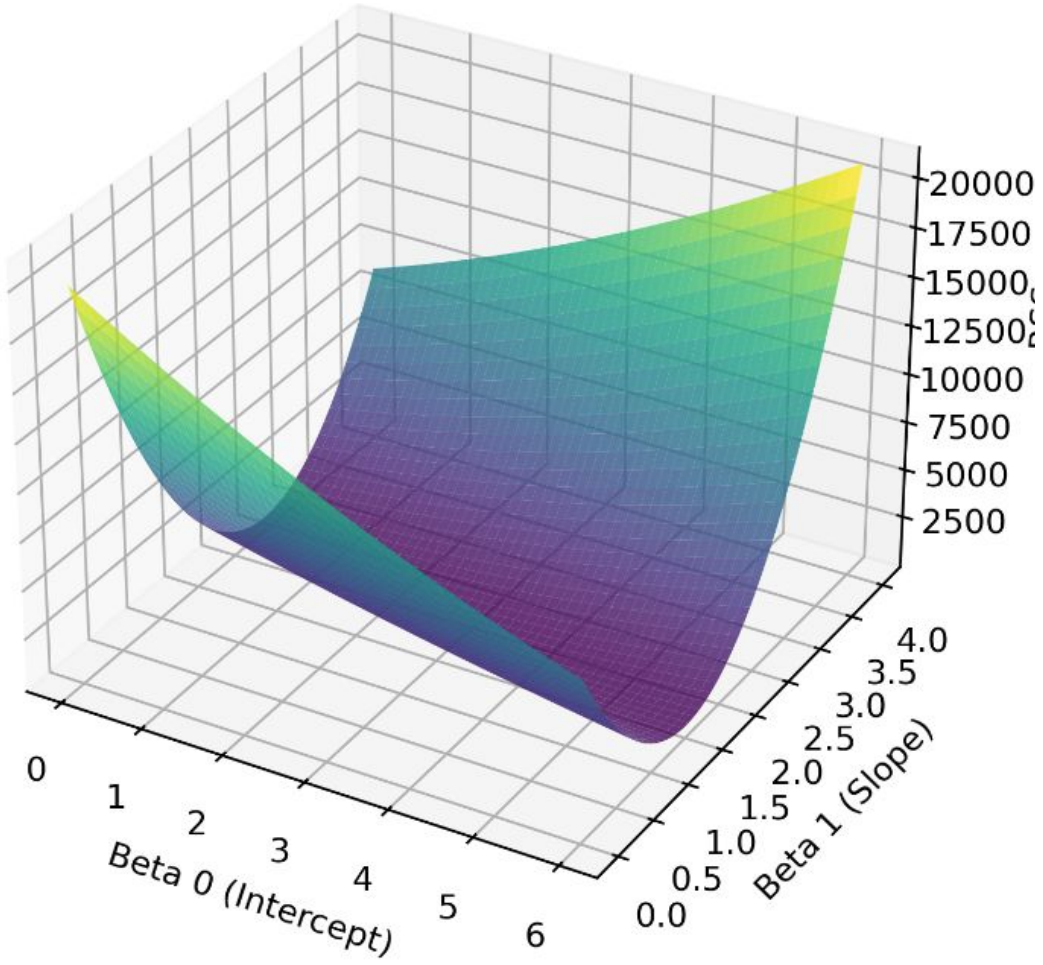
Calculate partial derivative of:

$$\mathcal{L}(\beta) = -\frac{1}{N}\sum_{i=1}^{N} \left[y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i) \right]$$

with respect to each β parameter:

$$\frac{\partial \mathcal{L}}{\partial \beta_0} = \frac{1}{N}\sum_{i=1}^{N} (\hat{y}_i - y_i)\, x_{i,0}$$

$$\frac{\partial \mathcal{L}}{\partial \beta_1} = \frac{1}{N}\sum_{i=1}^{N} (\hat{y}_i - y_i)\, x_{i,1}$$



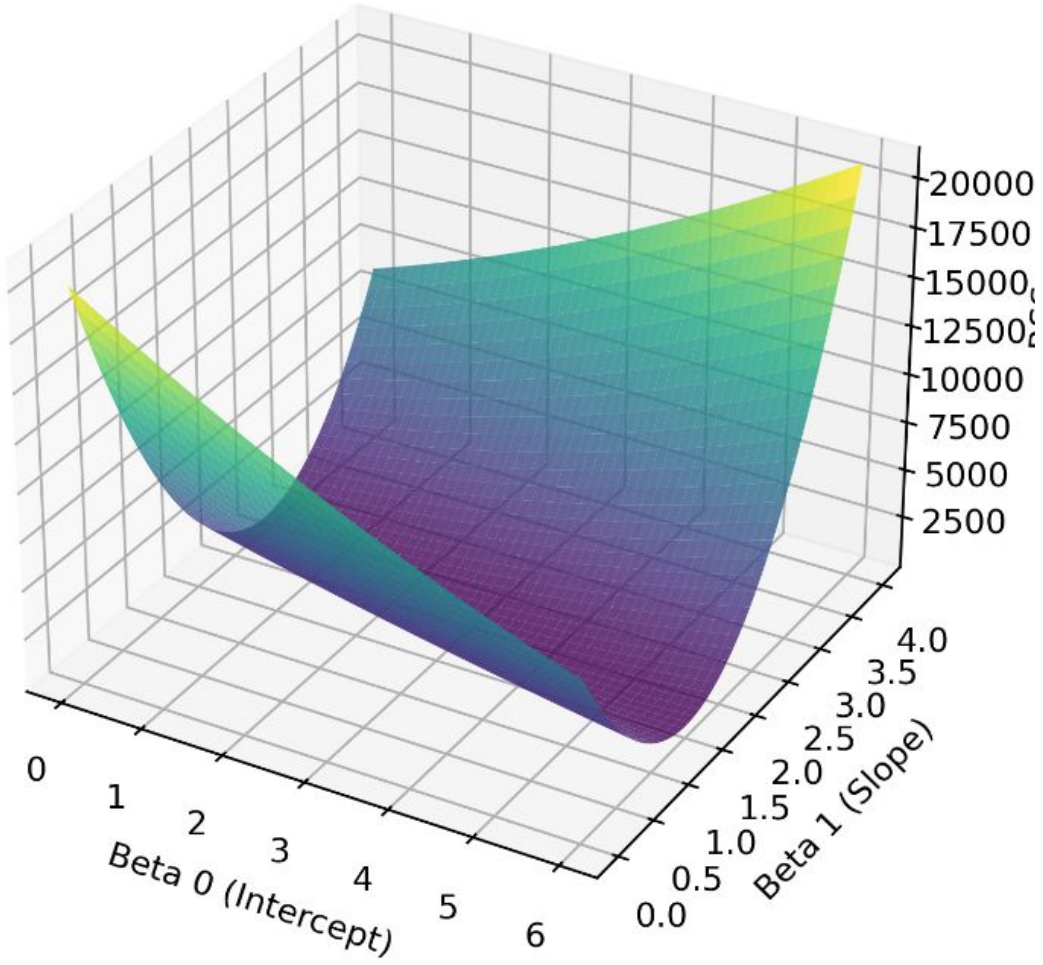
Gradient descent using partial derivatives of loss

Calculate partial derivative of:

$$\mathcal{L}(\beta) = -\frac{1}{N}\sum_{i=1}^{N} [y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i)]$$

with respect to each β parameter:

$$\nabla_\beta \mathcal{L} = \frac{1}{N} X^\top (\hat{y} - y)$$



Learning Rate is an important parameter

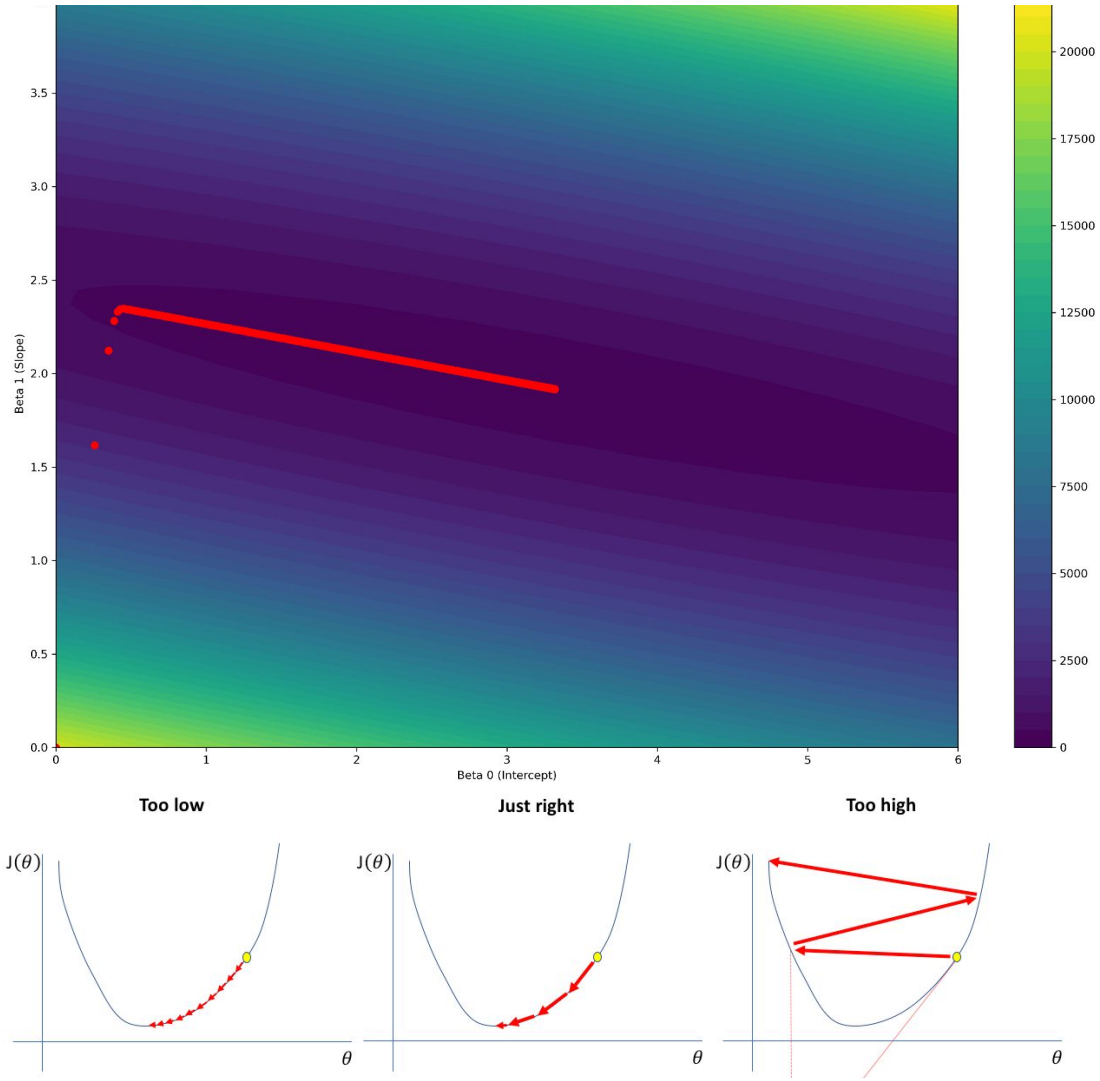
Learning rate is essentially relative “step” size when sliding down the gradient.

Learning rate too small:

- Convergence becomes extremely slow
- Stuck in local minima
- Can run out of iterations!

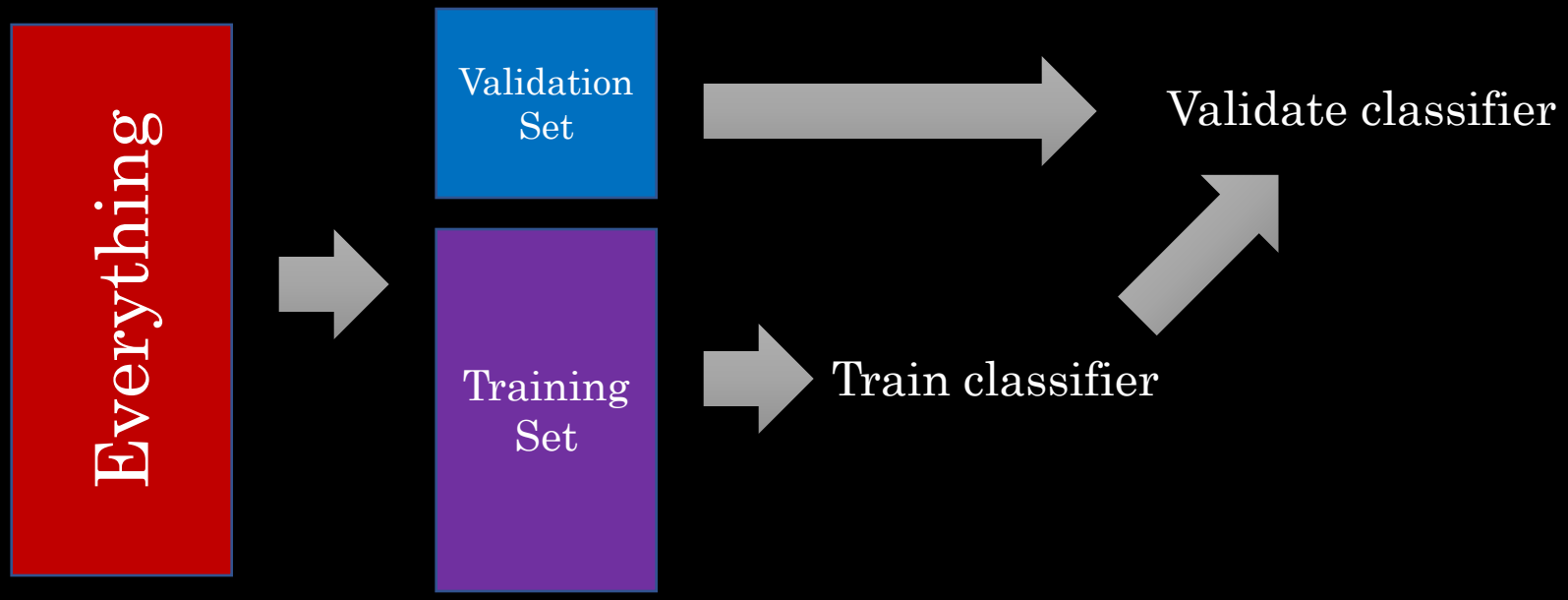
Learning rate too large:

- Overshoot optimal value and oscillate
- Failure to converge
- Float overflows



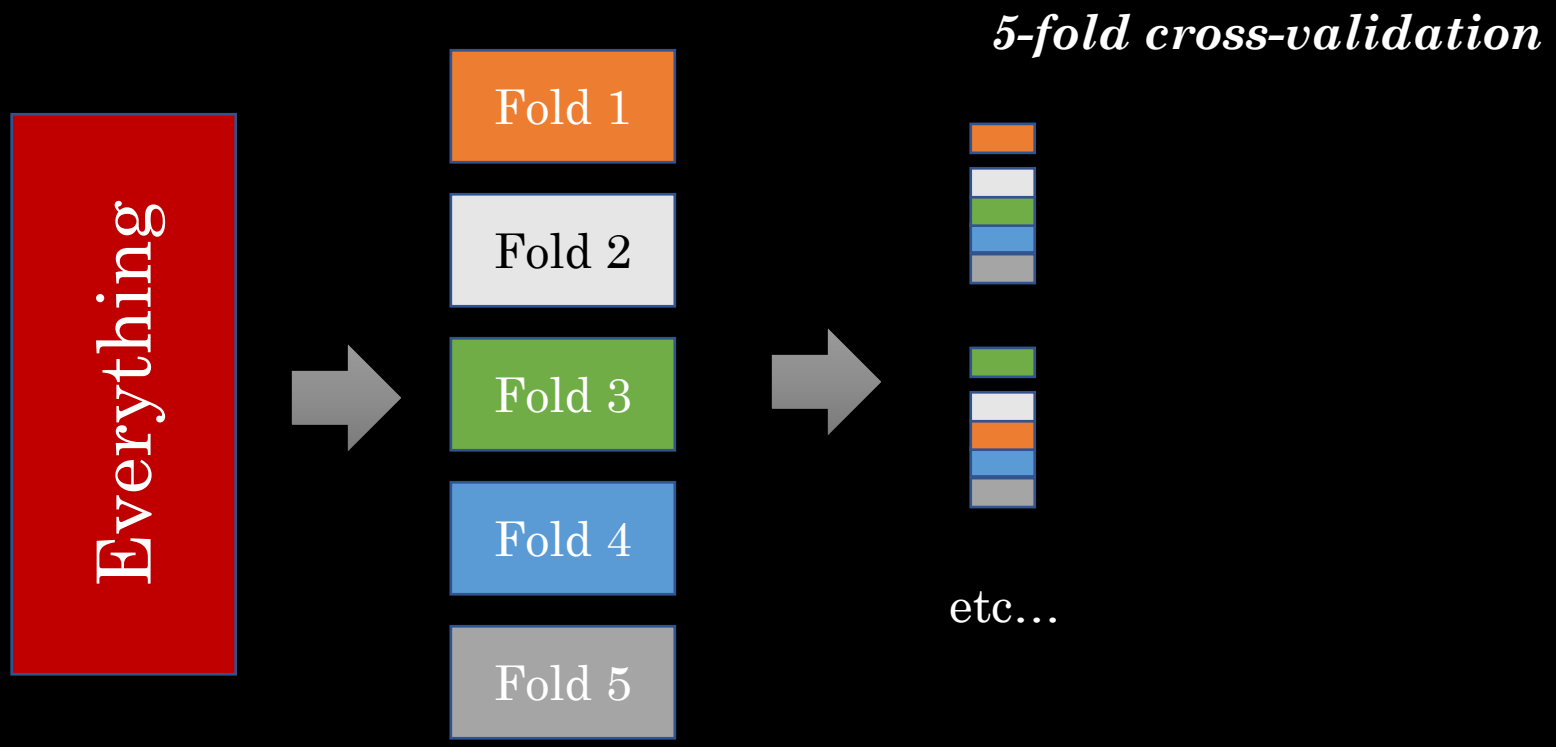
Data set splitting 1 – Validation

Use a fraction of available cases as the *training set*, reserve the remainder for a *validation* set



Cross-validation

Repeated training with different subsets

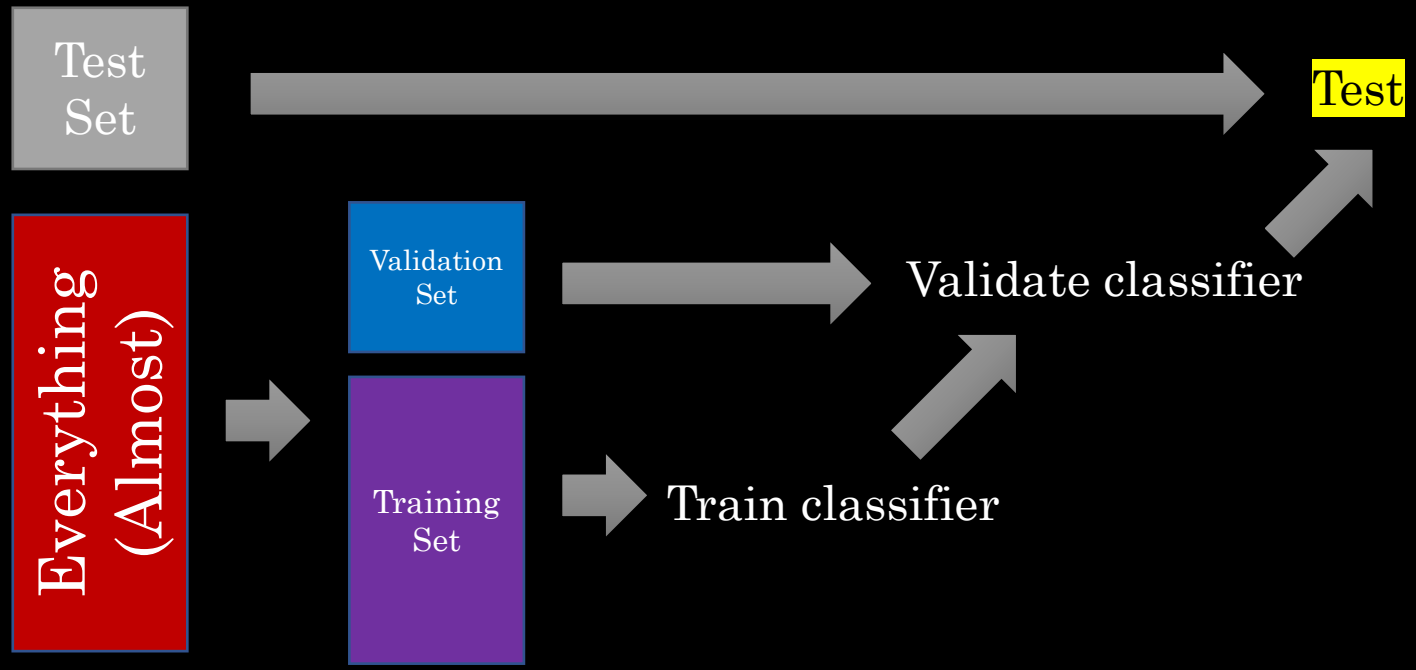


The cross-validation score is the average performance on all validation sets

Often used to optimize model design (e.g., hyperparameters)

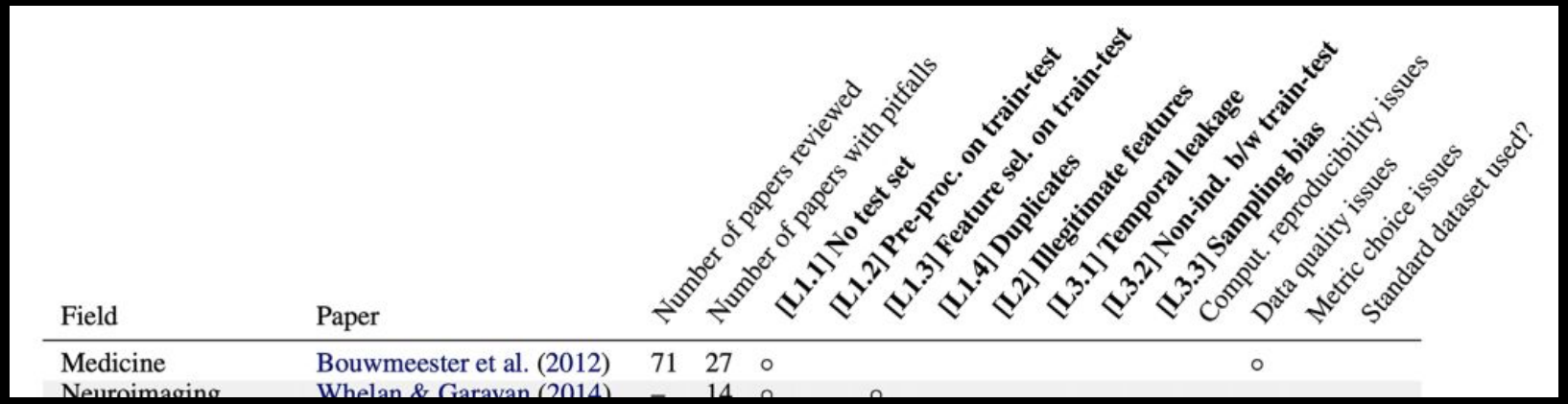
Data Set Splitting 2 – Testing

- With k-fold cross-validation, all data are used in training k-1 times
- Test set: Hold out part of the data entirely and assess only AFTER cross-validation / parameter selection has been completed



Constructing sets

- Sample sets at random, ensuring that every class is represented
- Each class has a distinct identifier. Two-class problems are often referred to positive and negative sets
- Data leakage can allow the classifier to "cheat" and get a misleadingly high accuracy on validation and test sets



Field	Paper	Number of papers reviewed	Number of papers with pitfalls	[L1.1] No test set	[L1.2] Pre-proc. on train-test	[L1.3] Feature sel. on train-test	[L1.4] Duplicates	[L2] Illegitimate features	[L3.1] Temporal leakage	[L3.2] Non-ind. b/w train-test	[L3.3] Sampling bias	Comput. reproducibility issues	Data quality issues	Metric choice issues	Standard dataset used?
Medicine	Bouwmeester et al. (2012)	71	27	○								○			
Neuroimaging	Whelan & Garavan (2014)	–	14	○		○									

Kapoor and Narayanan (2023). "Leakage and the reproducibility crisis in machine-learning-based science" *Patterns*

Measuring Accuracy

Confusion matrix for a two-class (positive and negative set) problem:

True \ Predicted	+	-
+	True positive	False negative
-	False positive	True negative

Many tools make quantitative predictions, even with categorical problems: accuracy calculations may require thresholding of these predictions

Quantifying Accuracy

Sensitivity, recall, true positive rate	$= \frac{TP}{TP+FN} = \frac{TP}{n^+}$	Specificity, true negative rate	$= \frac{TN}{TN+FP} = \frac{TN}{n^-}$
Positive predictive value, precision	$= \frac{TP}{TP+FP}$	Negative predictive value	$= \frac{TN}{TN+FN}$
False positive rate, fallout	$= \frac{FP}{FP+TN} = \frac{FP}{n^-}$	False discovery rate	$= \frac{FP}{FP+TP}$

Chicco and Jurman (2020) *BMC Genomics*

Combined performance on both classes

- Measures need to correct for set size
- Balanced accuracy: intuitive, all classes are weighted equally
- F1: Harmonic mean of precision and recall, no consideration of TN
- MCC: Balanced consideration, can behave weirdly with extreme values (good thing?)

Balanced Accuracy

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right)$$

F1 Score

$$F1 = \frac{2 \times TP}{2 \times TP + FP + FN}$$

Matthews Correlation Coefficient (MCC)

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

Vigorous comparison (and possibly overly bold assertions): Chicco and Jurman (2020) *BMC Genomics*

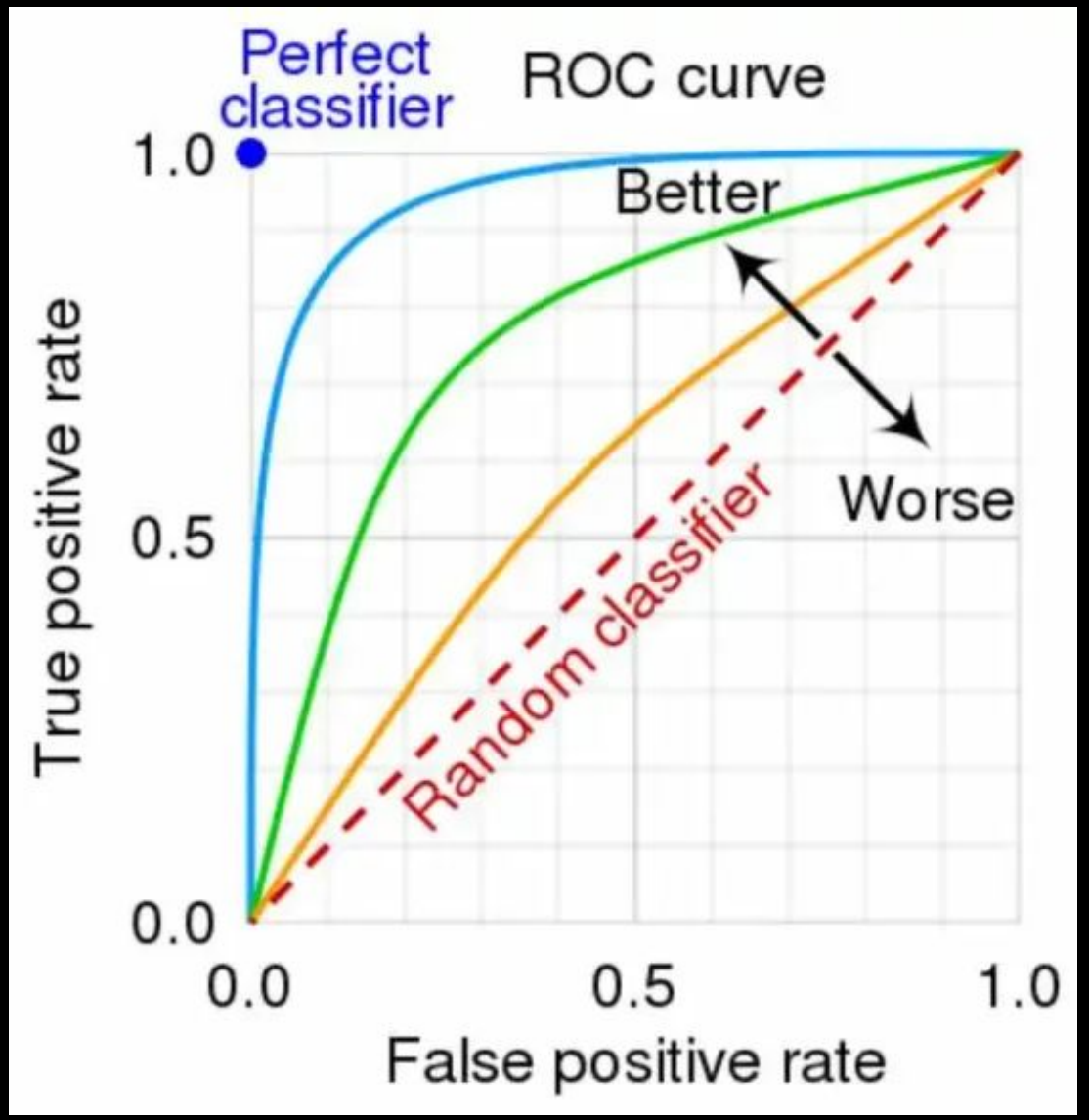
Aaaargh!

		Predicted condition		Sources: [4][5][6][7][8][9][10][11][12] view · talk · edit	
	Total population $= P + N$	**Predicted Positive (PP)**	**Predicted Negative (PN)**	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	**Positive (P)** [a]	**True positive** (TP), hit[b]	**False negative** (FN), miss, underestimation	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate type II error [c] $= \frac{FN}{P} = 1 - TPR$
	Negative (N)[d]	**False positive** (FP), false alarm, overestimation	**True negative** (TN), correct rejection[e]	False positive rate (FPR), probability of false alarm, fall-out type I error [f] $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN} = 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2\,PPV \times TPR}{PPV + TPR} = \frac{2\,TP}{2\,TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

https://en.wikipedia.org/wiki/F-score

ROC Curves

- Plot the true positive rate and false positive rate at every threshold
- Area under the curve (AUC) is a measure of performance
- In a two-class problem:
  - 1.0: perfect
  - 0.5: random
  - < 0.5: worse than random (?)



https://spotintelligence.com/2024/06/17/roc-auc-curve-in-machine-learning/

Regression problems

- RMSD is more commonly used (for example, in regression)

- RMSD assigns disproportionately larger penalties to larger errors than MAE

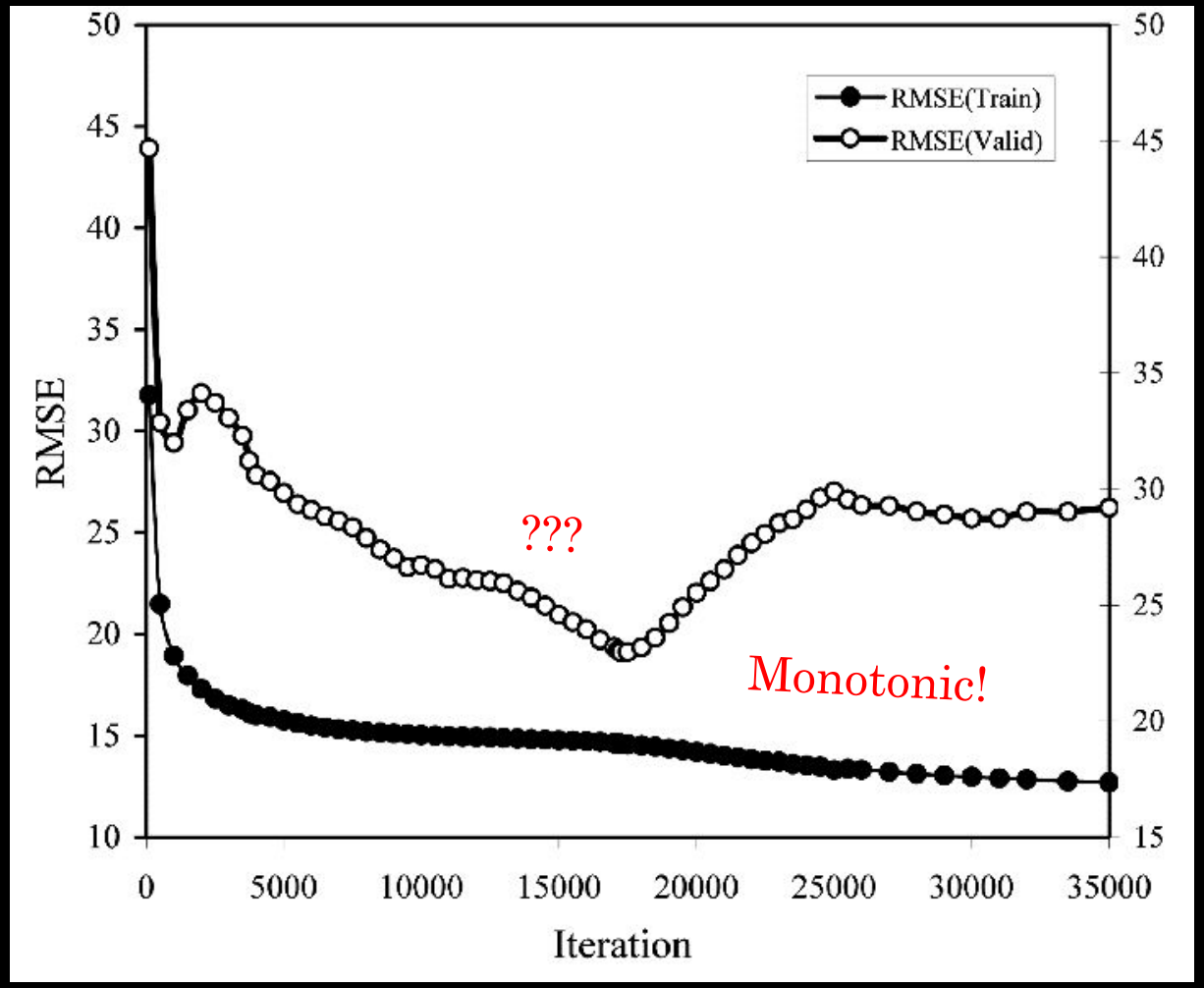
Mean absolute error

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

Root mean square deviation

$$\mathrm{RMSD} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Iterative training



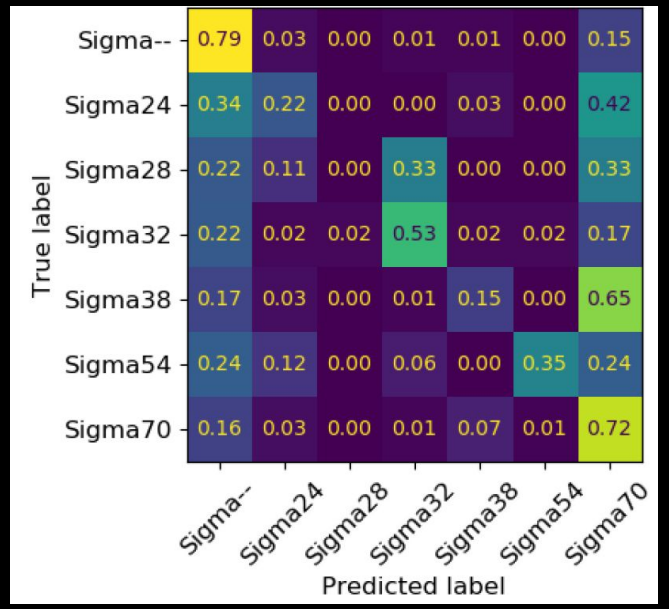
- Error on the training set tends to decrease monotonically
- Validation set error will often decrease, then jump due to overfitting
- Stopping criteria for iterative training often make use of this

Aziz and Esmailian (2007) *Bulletin of the Korean Chemical Society*

Expanding to multiple classes

- Accuracy measures can generalize (e.g., balanced accuracy) or be applied to each class individually (e.g., sensitivity of each class relative to everything else)
- Do we weight all classes equally, or do we weight by abundance?

Promoter predictions by class – different promoter types are active at different times



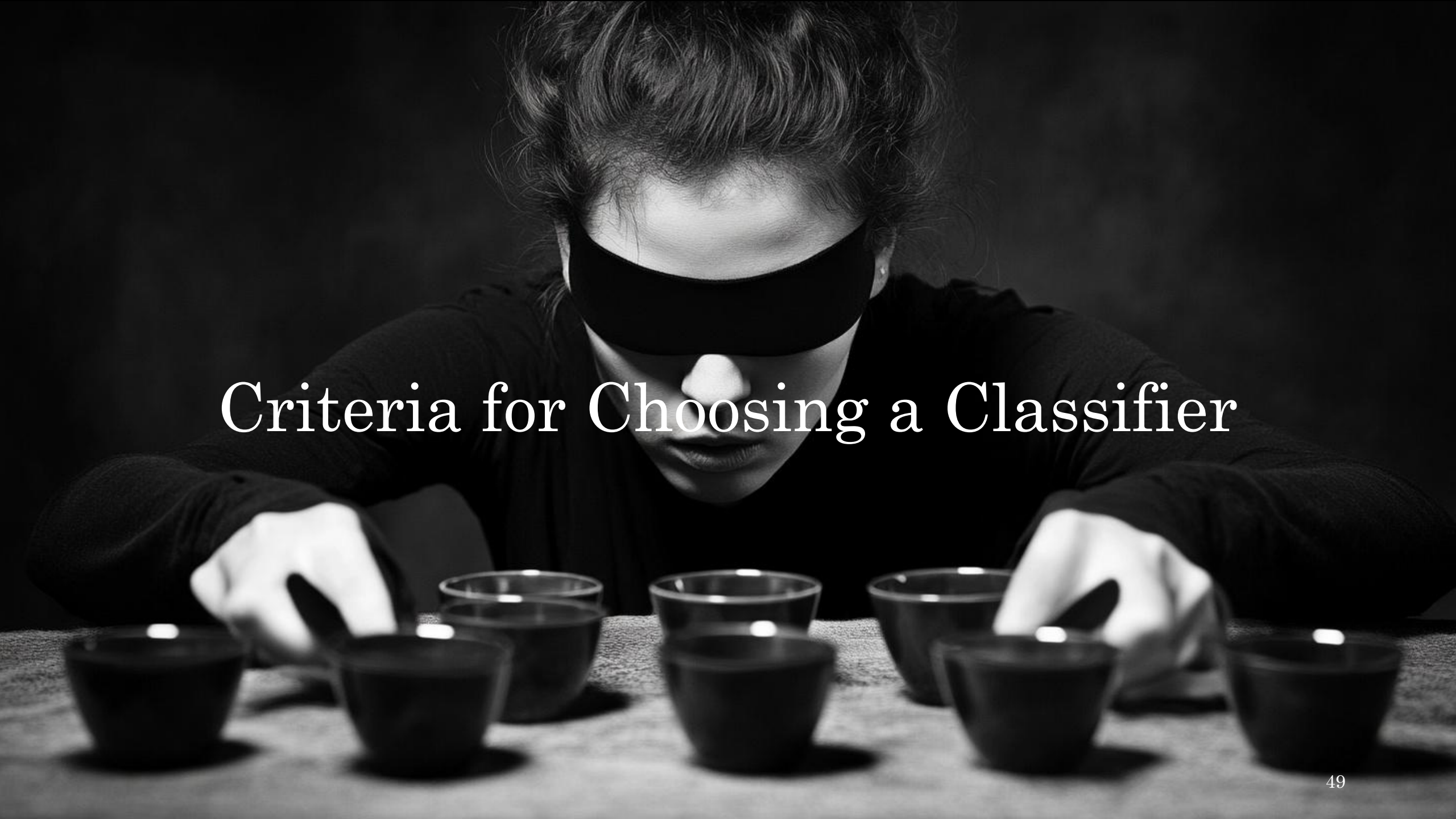
Bernardino and Beiko (2022) *BioSystems*

Regularization

- In basic terms, one or more procedures that rewards models that are simpler
- Need to balance accuracy with complexity
- Super-super general form:

$$Score = Accuracy - \lambda \times Complexity$$

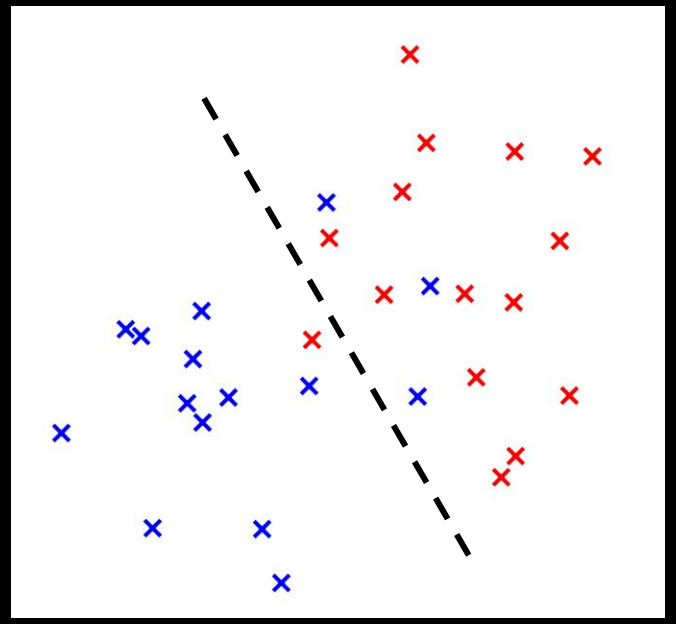
- The magnitude of λ determines the severity of the penalty



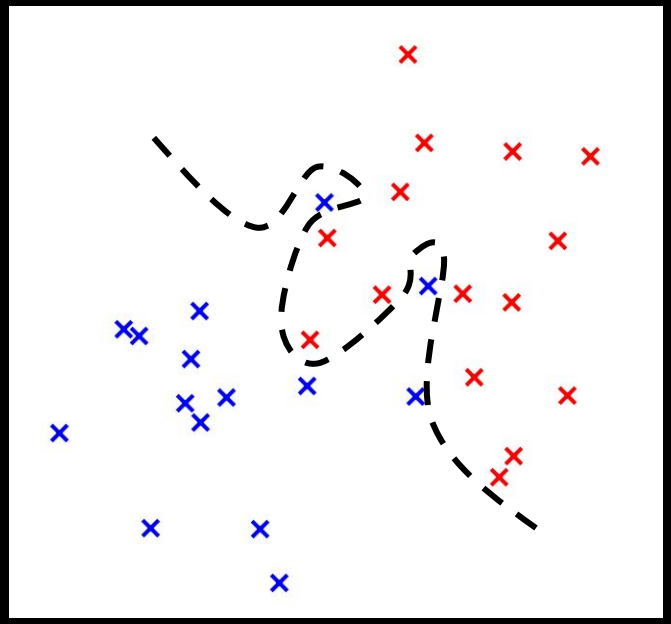
Criteria for Choosing a Classifier

Bias-Variance Tradeoff

- Do we want a classifier that is as simple as possible, or one that can make complex decisions?

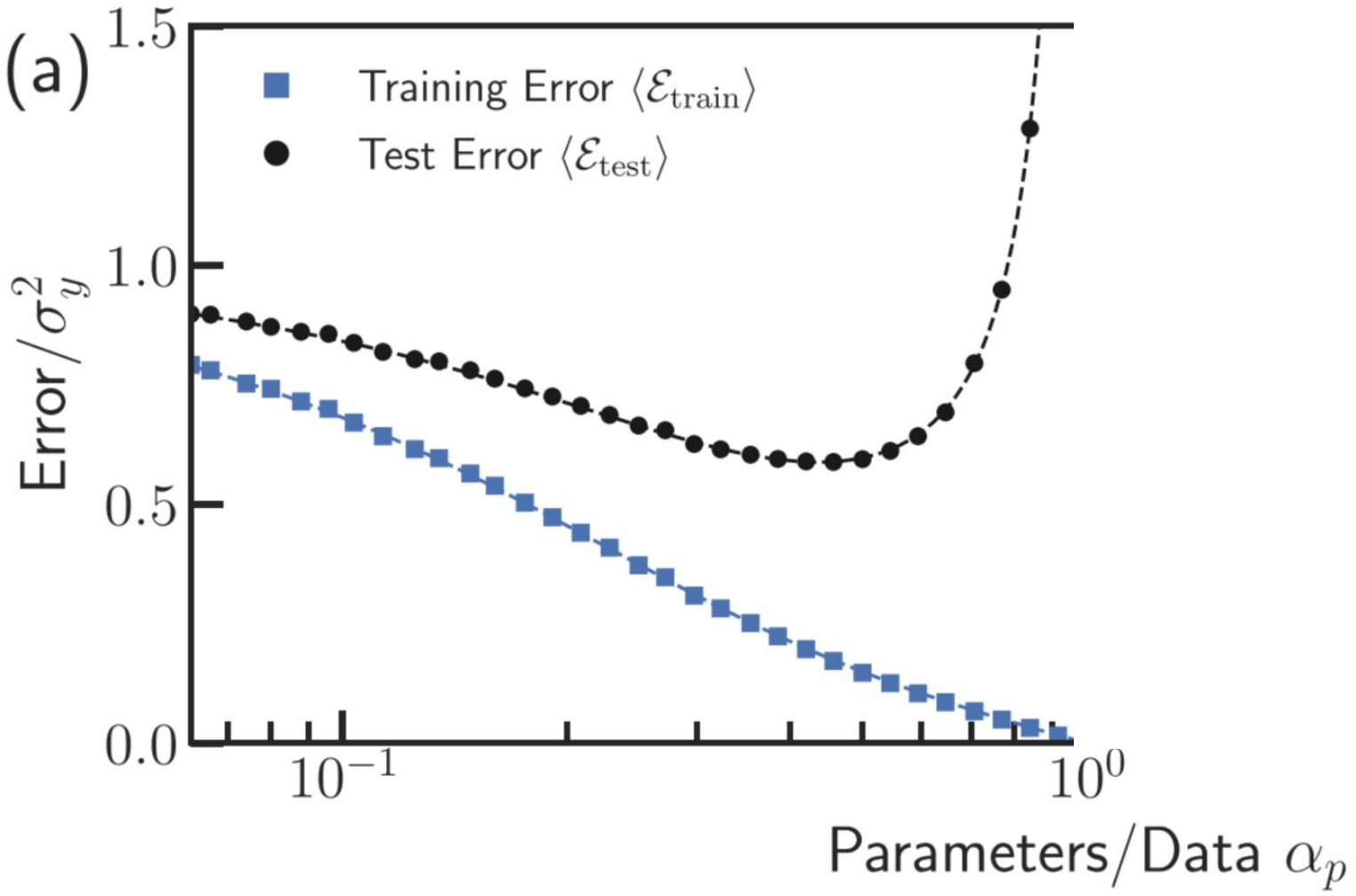


High bias (simple model)

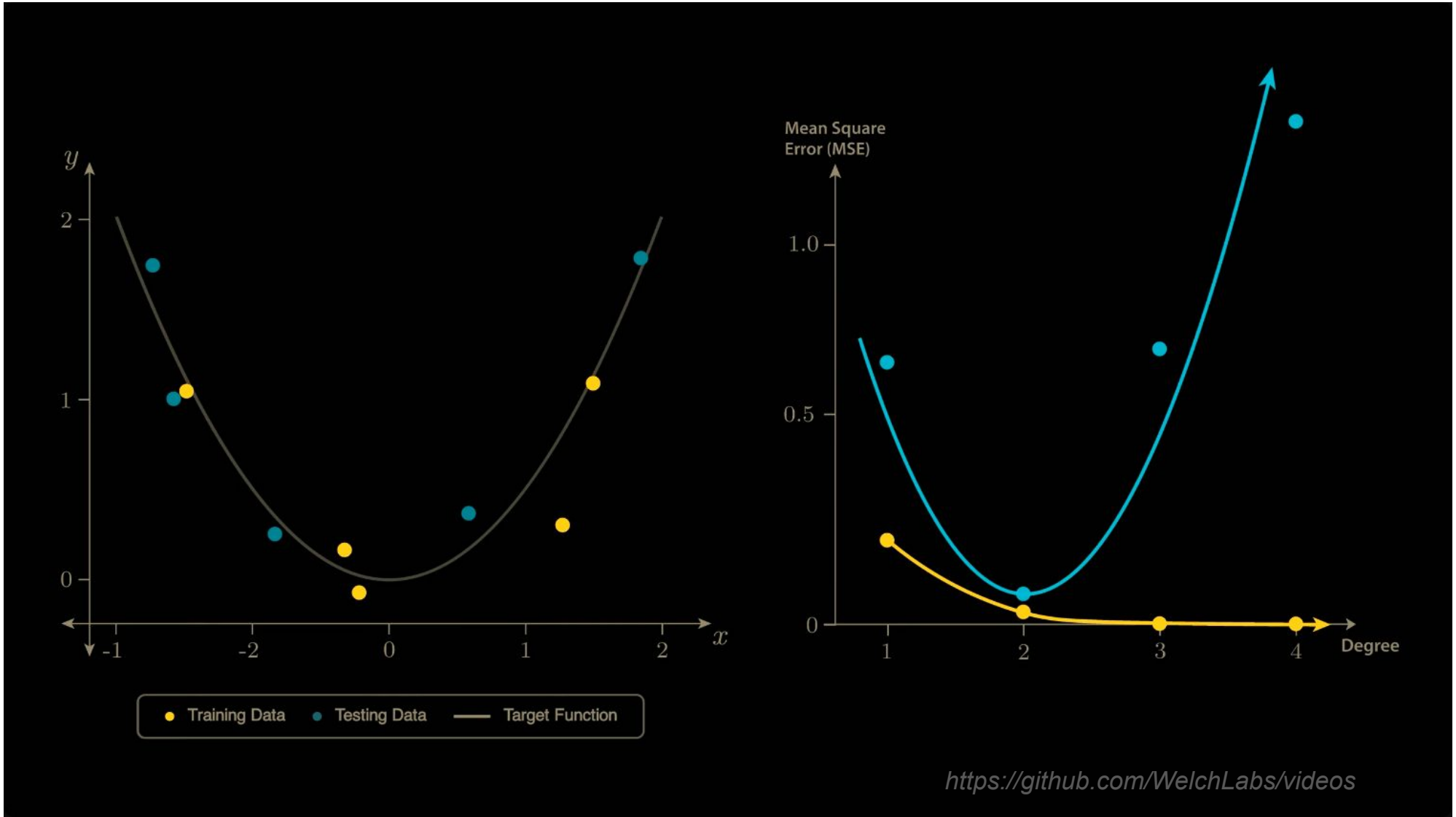


High variance (complex model)

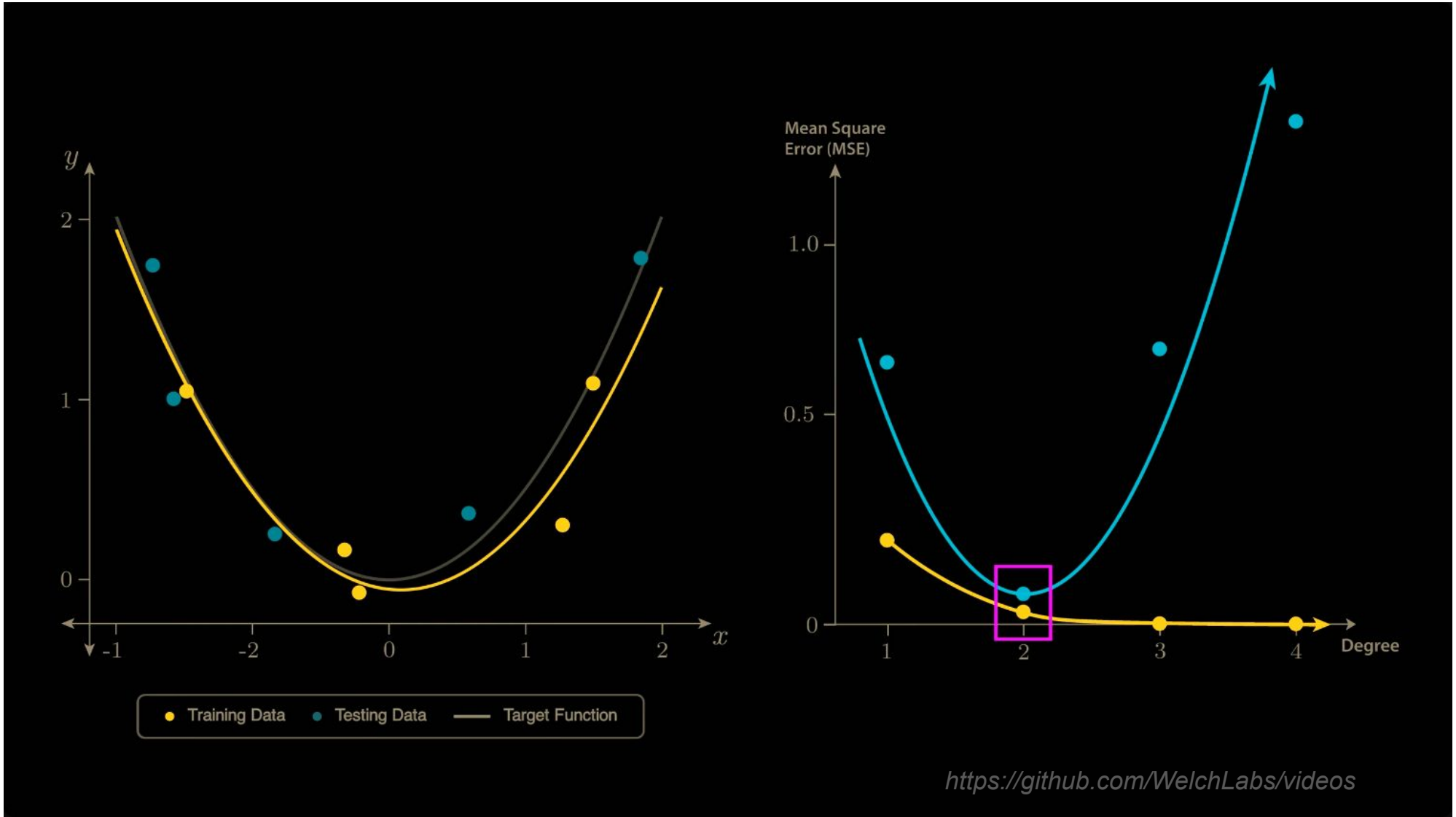
Bias-Variance Trade-off



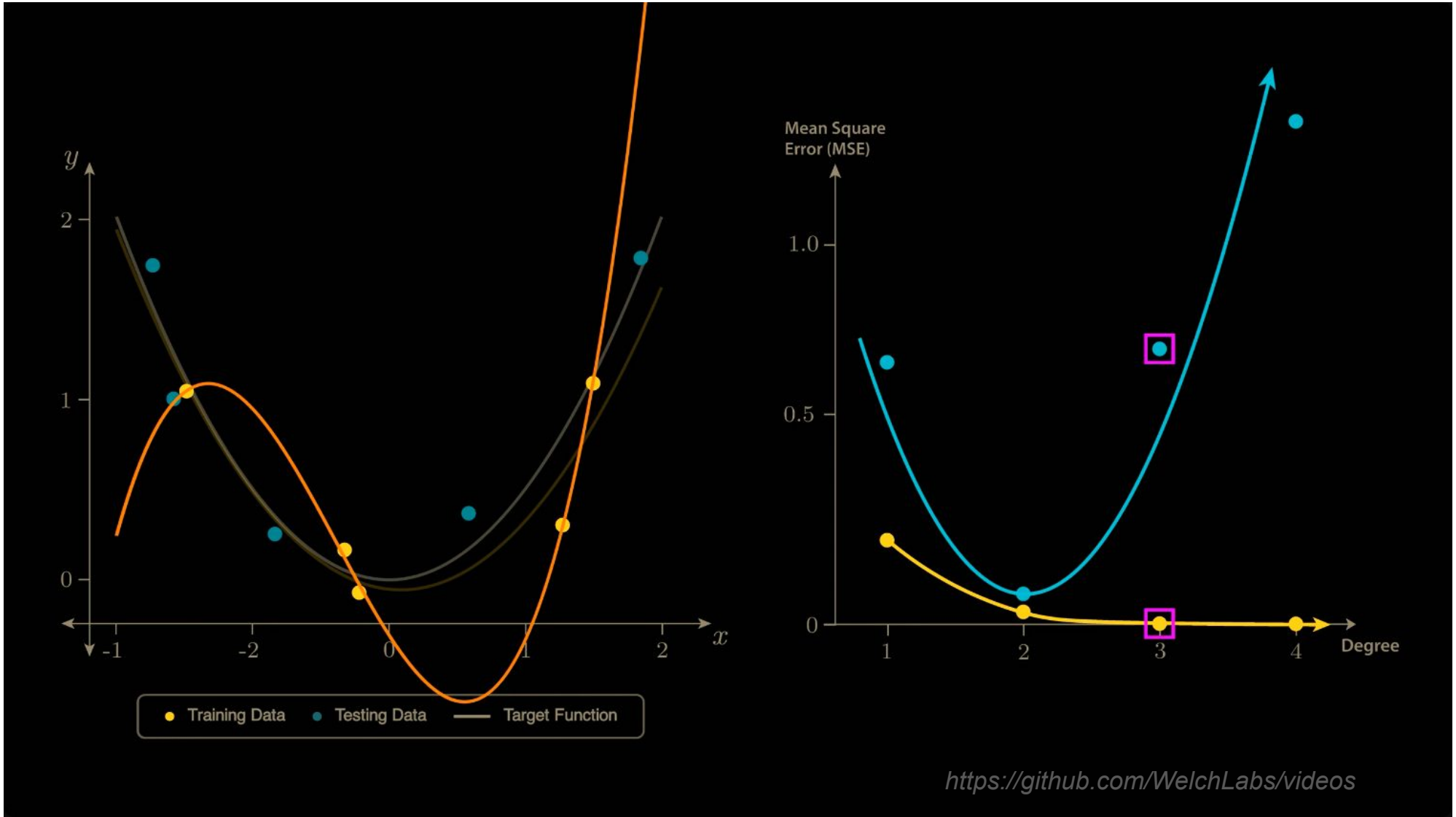
Bias-variance trade-off as model “complexity” increases



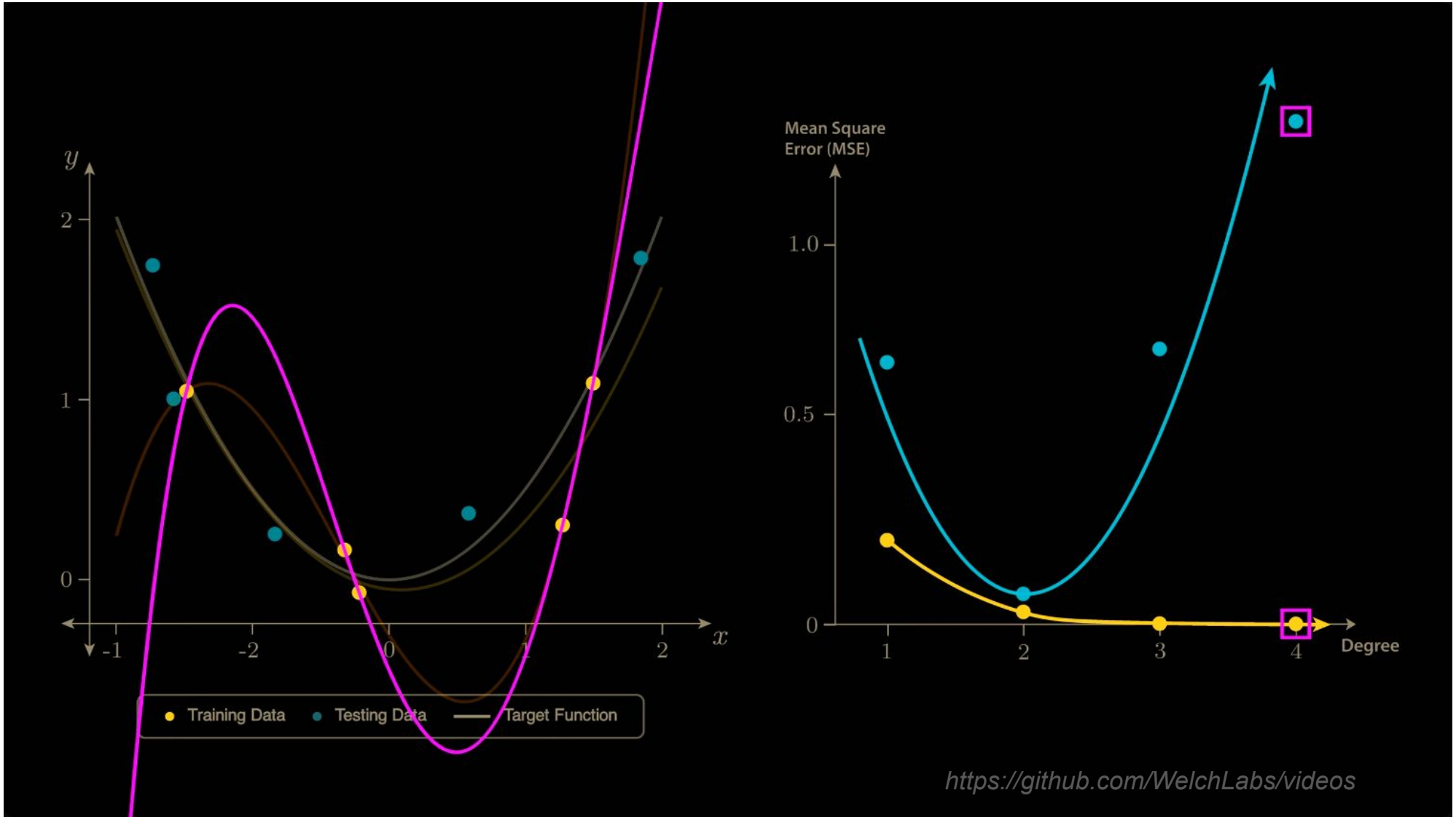
Bias-variance trade-off as model “complexity” increases



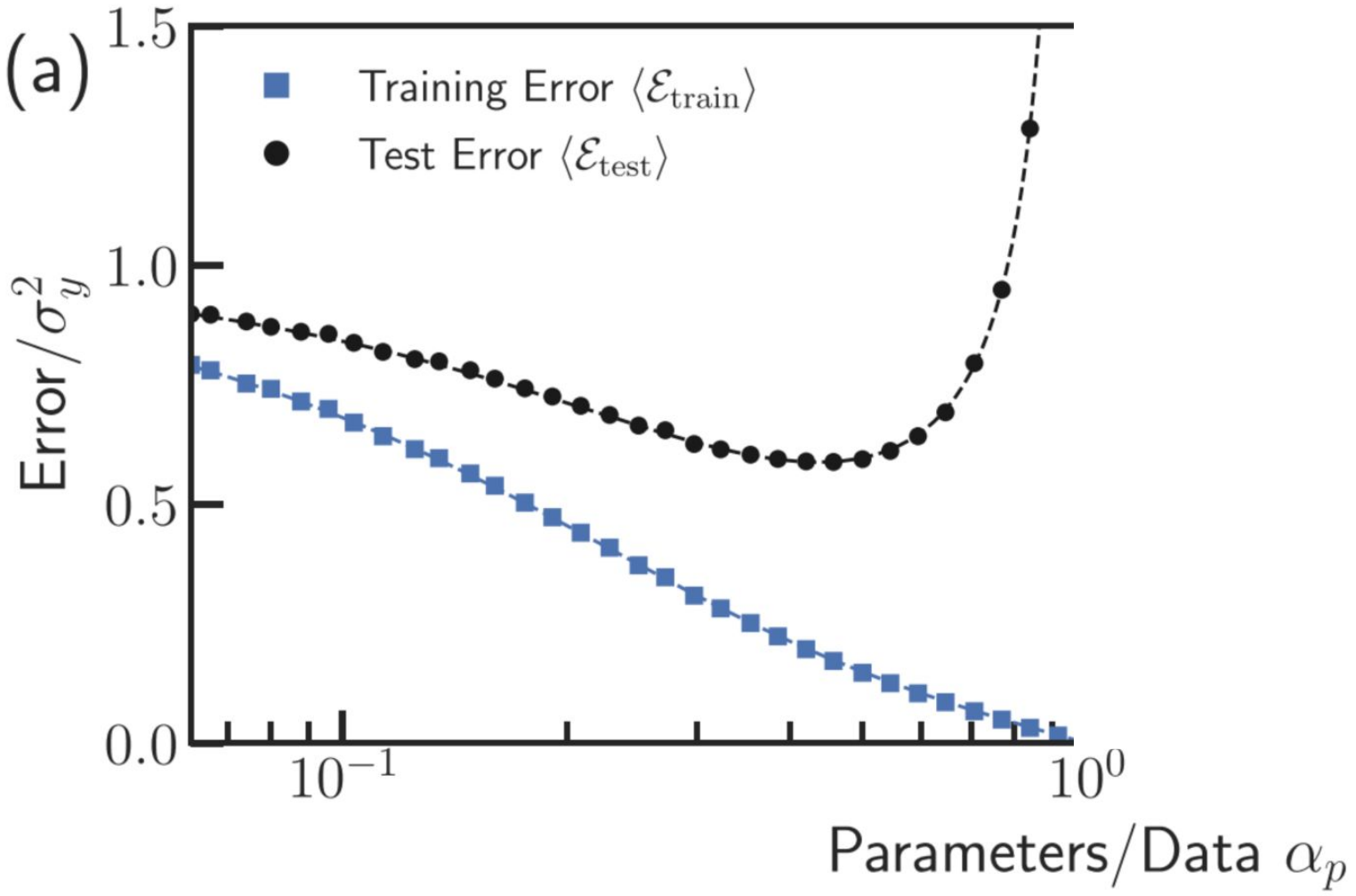
Bias-variance trade-off as model “complexity” increases



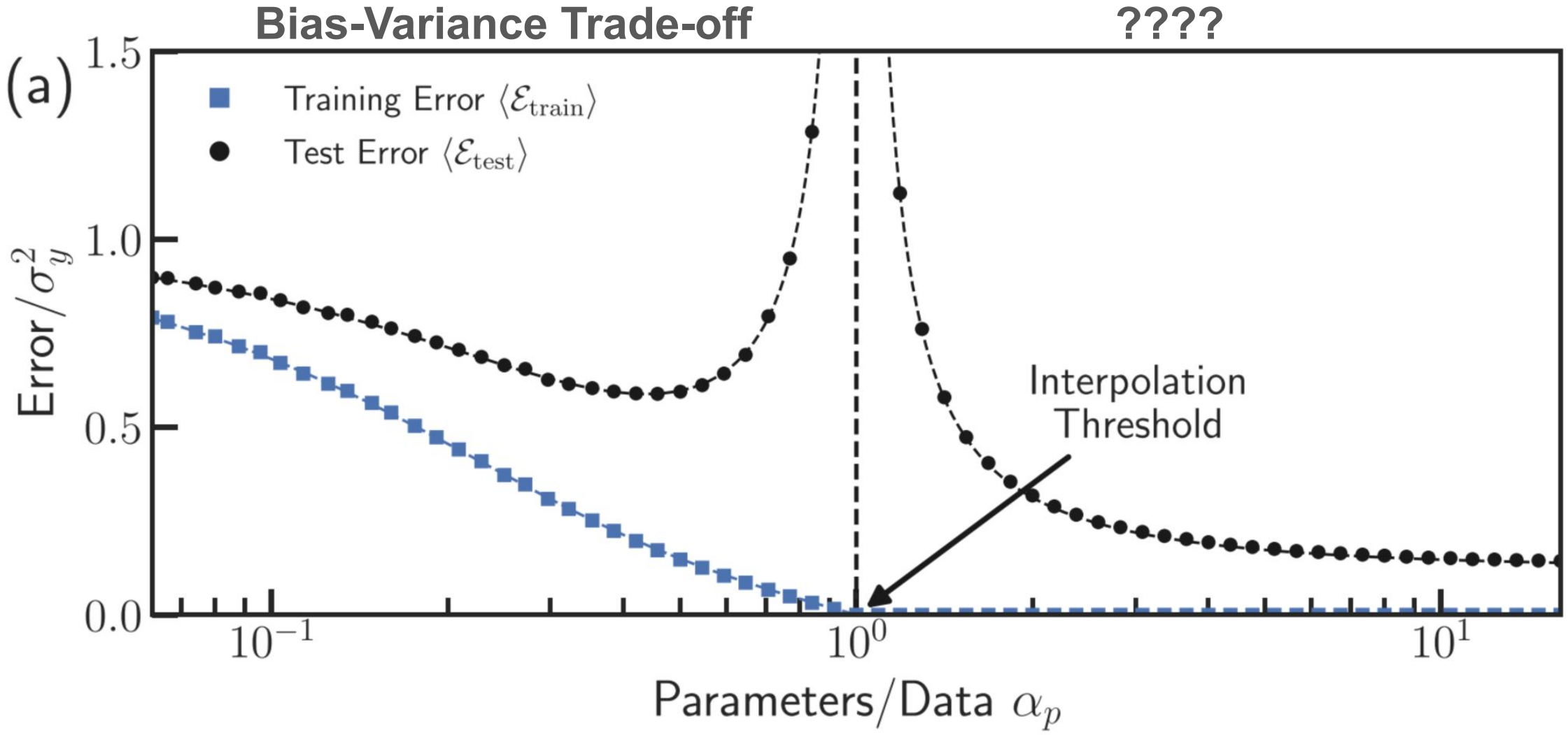
Bias-variance trade-off as model “complexity” increases



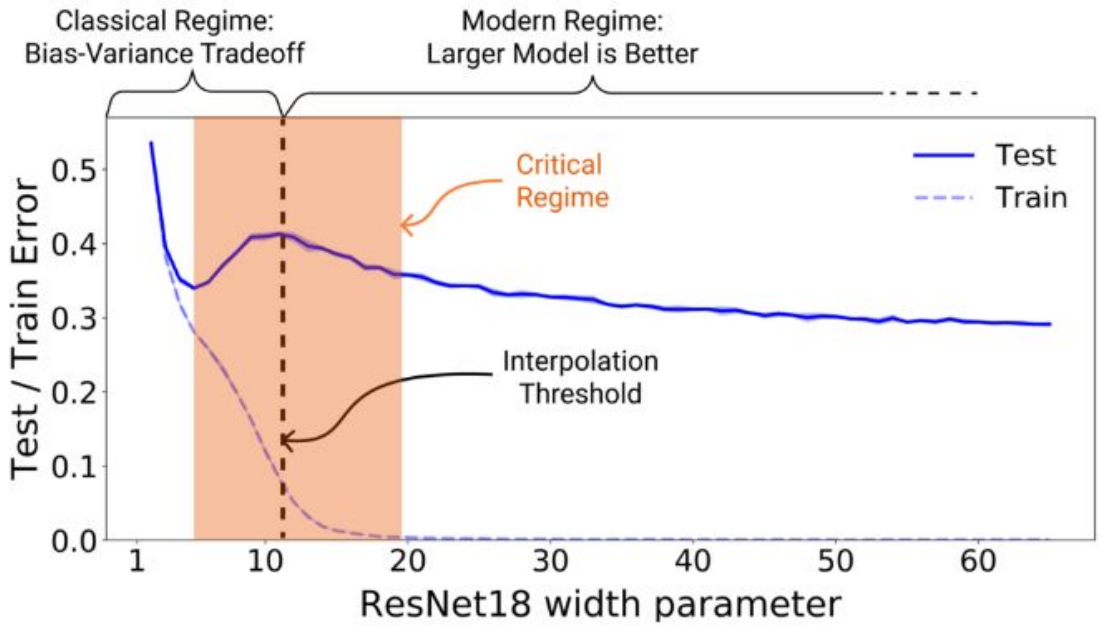
Bias-Variance Trade-off

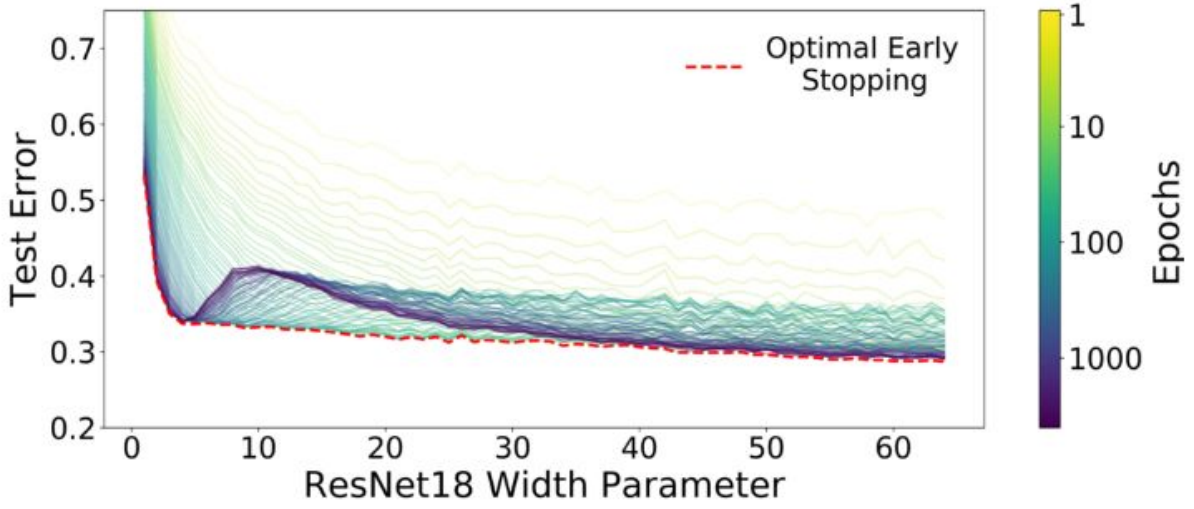


“Double Descent” beyond interpolation threshold



Empirically observed double descent with training length in large neural networks

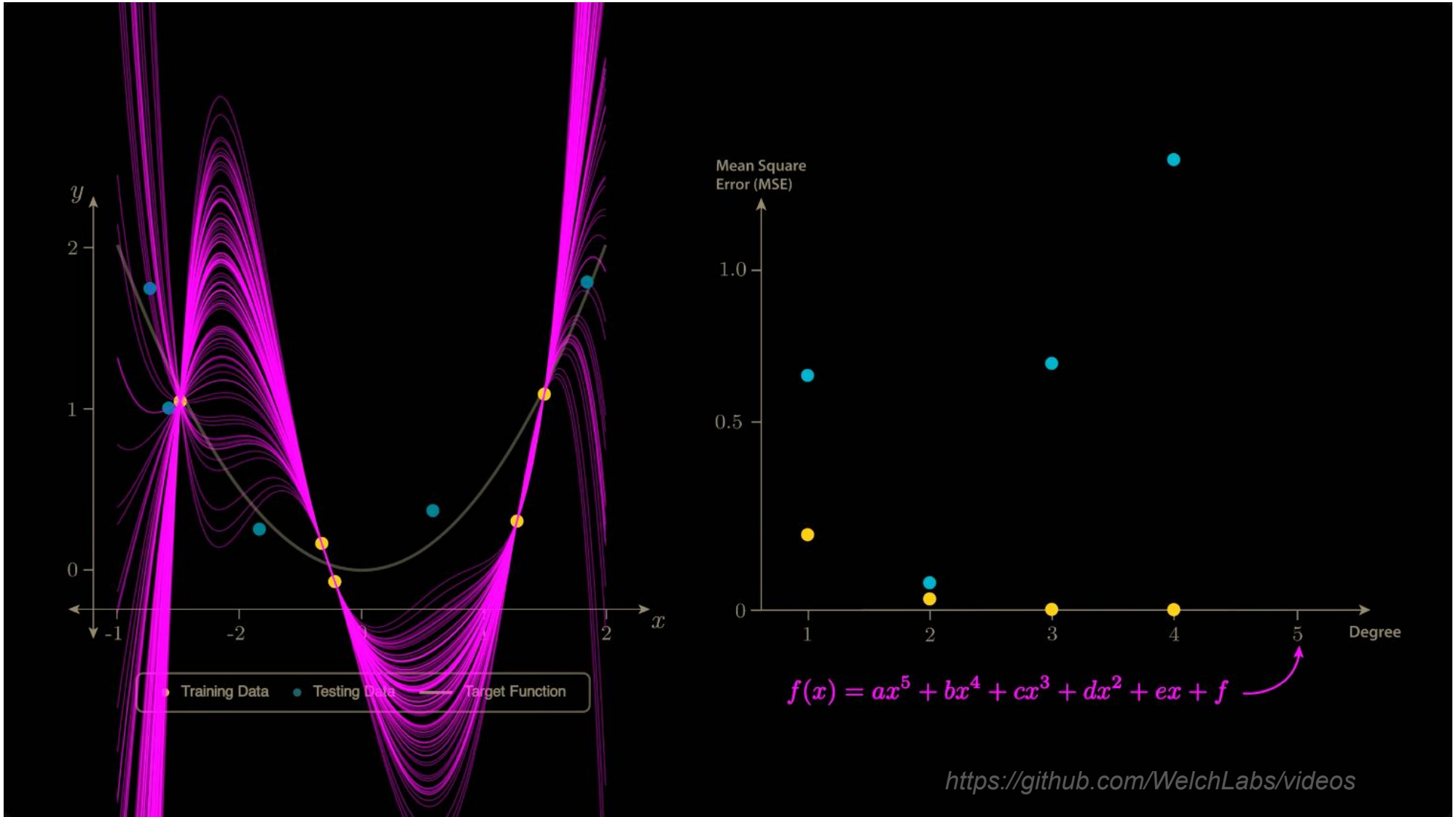




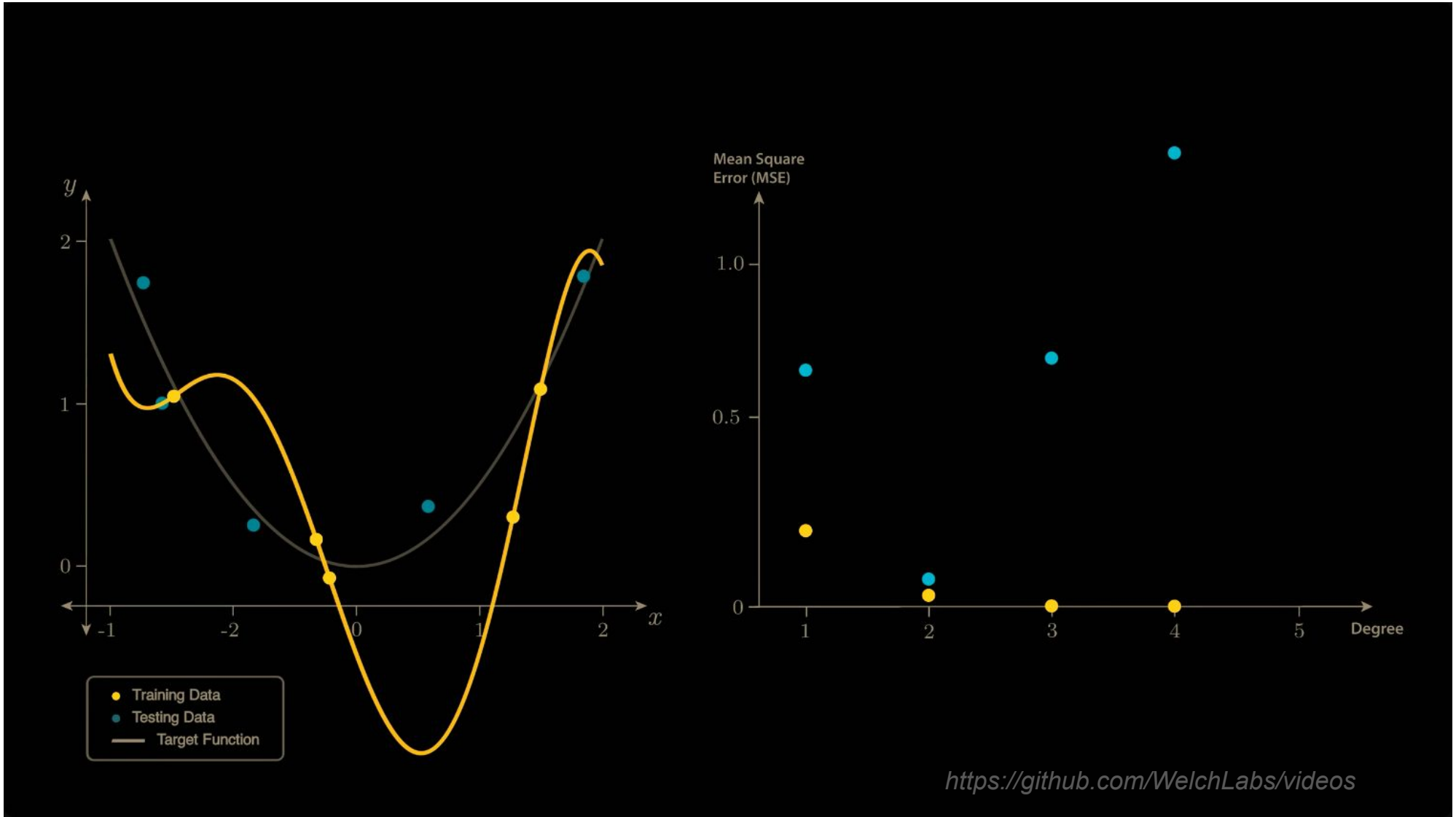
Nakkiran, Preetum, et al. "Deep double descent: Where bigger models and more data hurt." Journal of Statistical Mechanics: Theory and Experiment 2021.12 (2021): 124003.

Belkin, Mikhail, et al. "Reconciling modern machine-learning practice and the classical bias–variance trade-off." *Proceedings of the National Academy of Sciences* 116.32 (2019): 15849-15854.

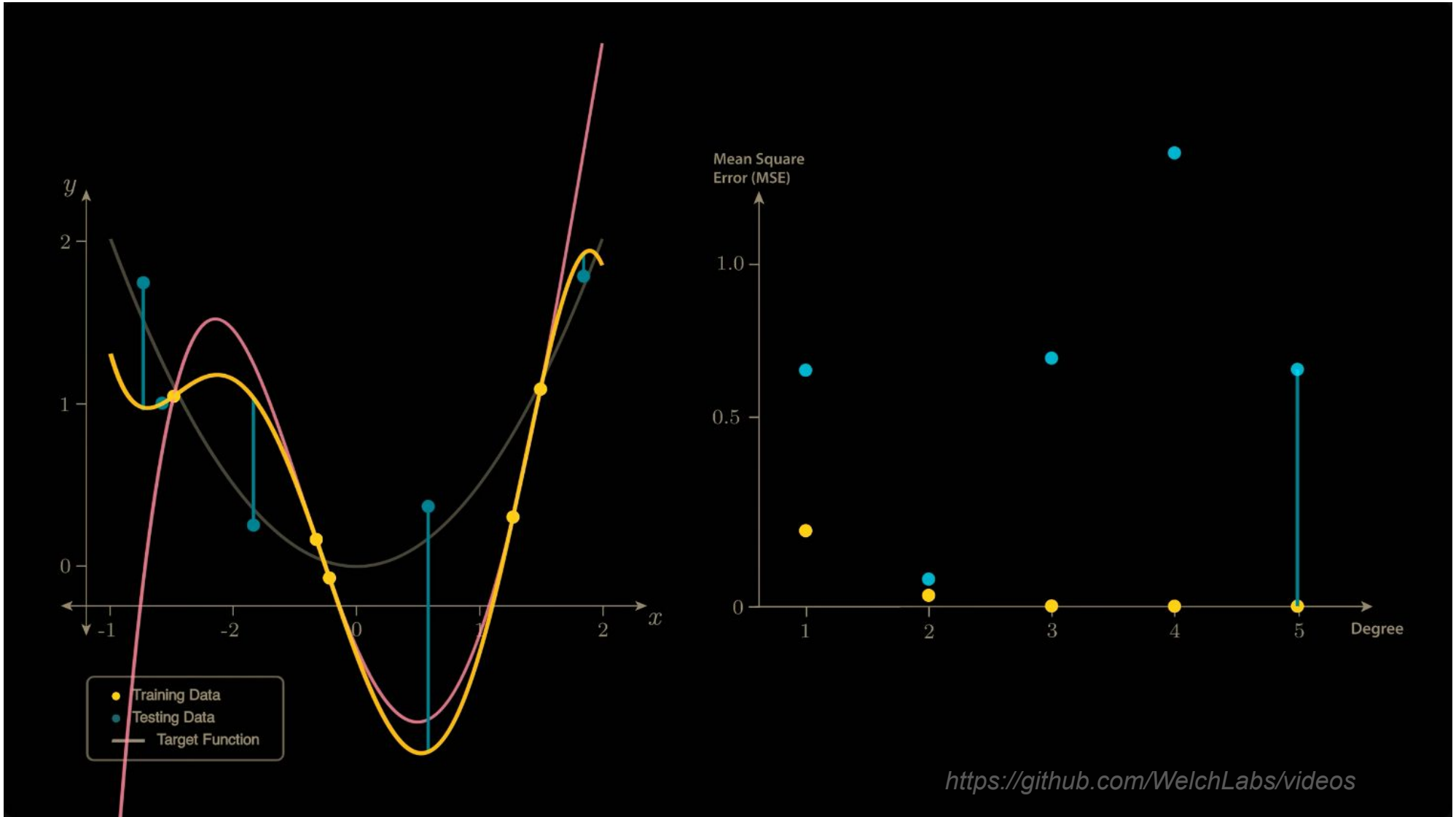
Regularised polynomial regression can show double descent



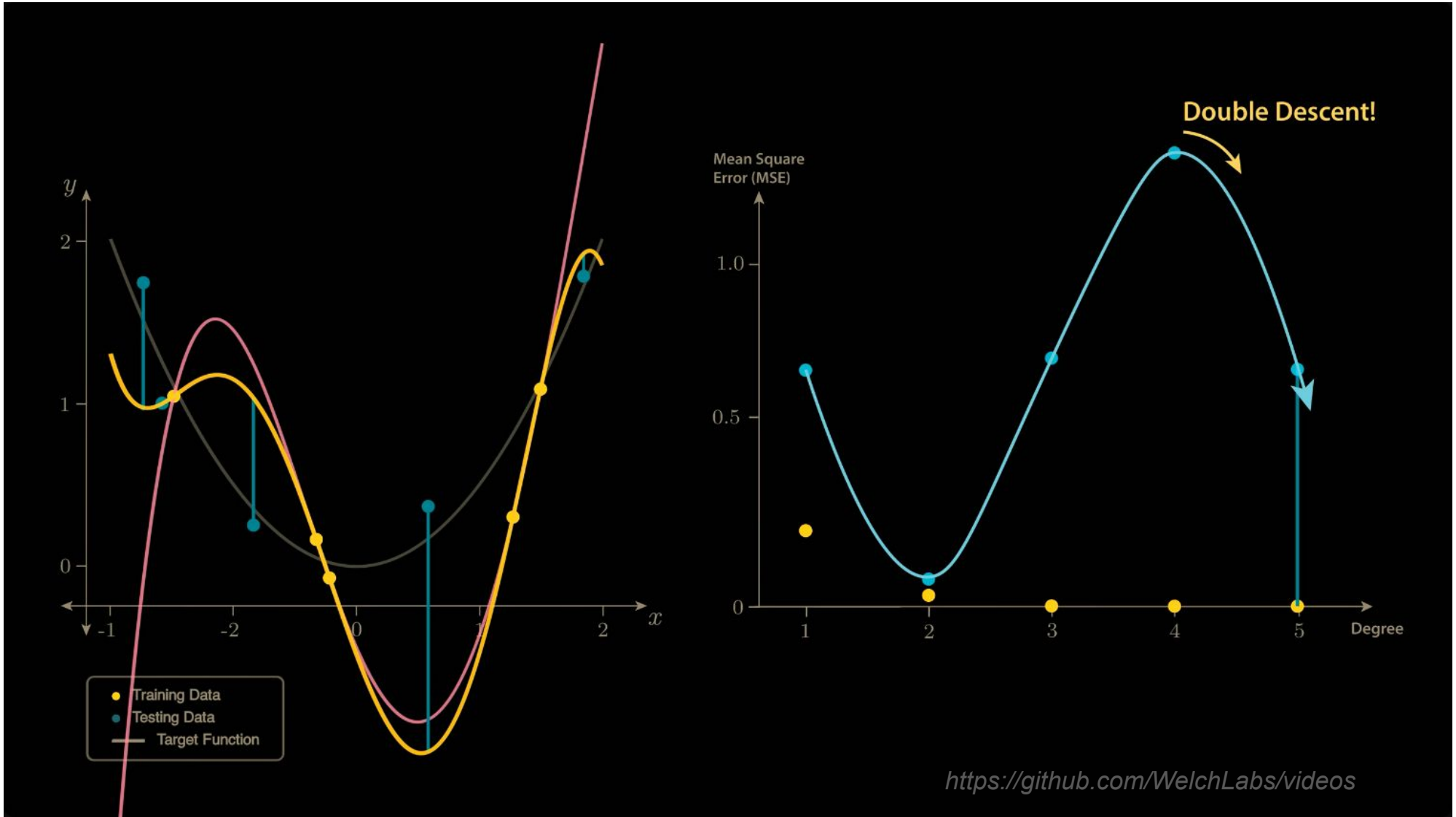
Regularised polynomial regression can show double descent



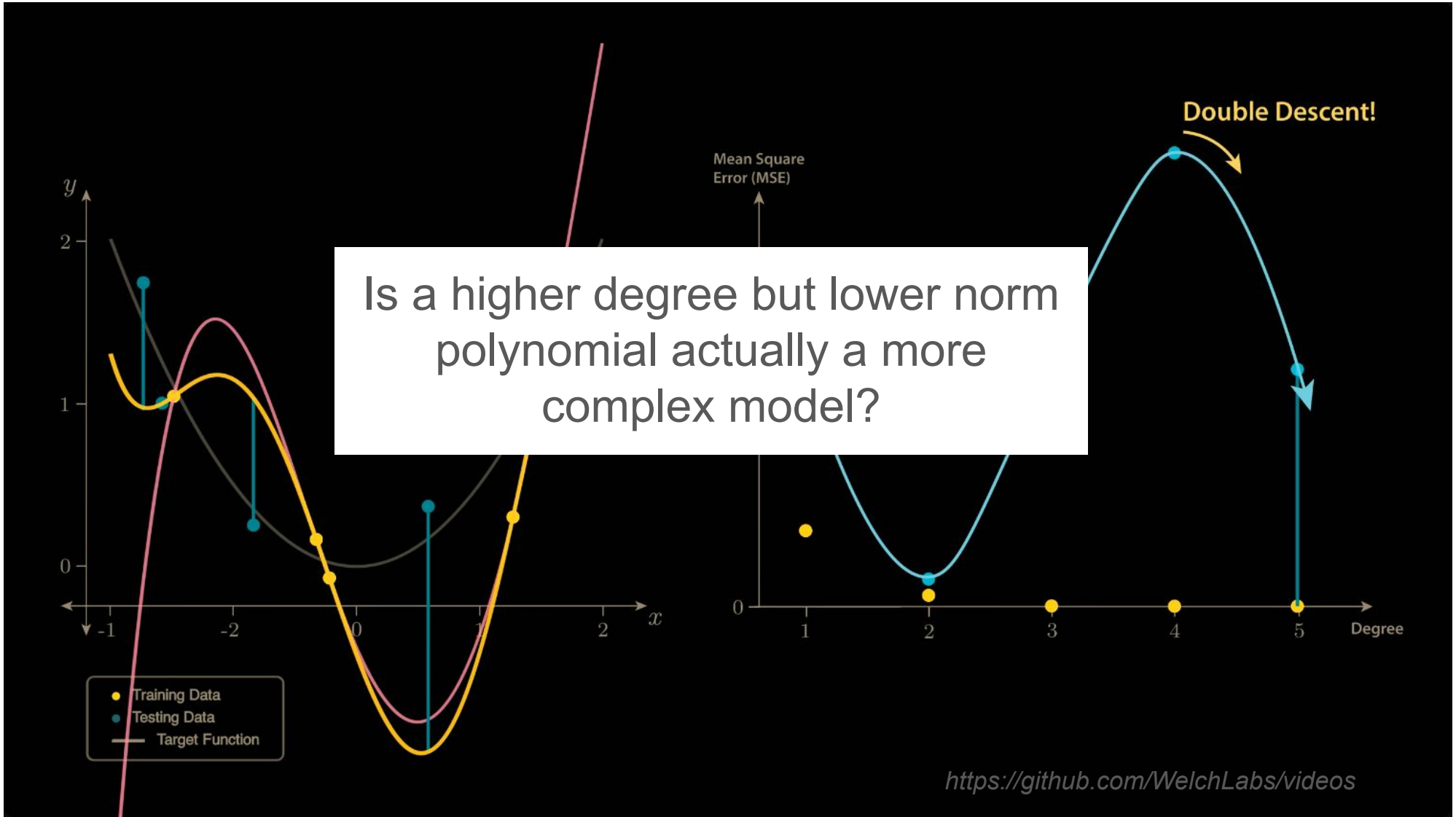
Regularised polynomial regression can show double descent



Regularised polynomial regression can show double descent

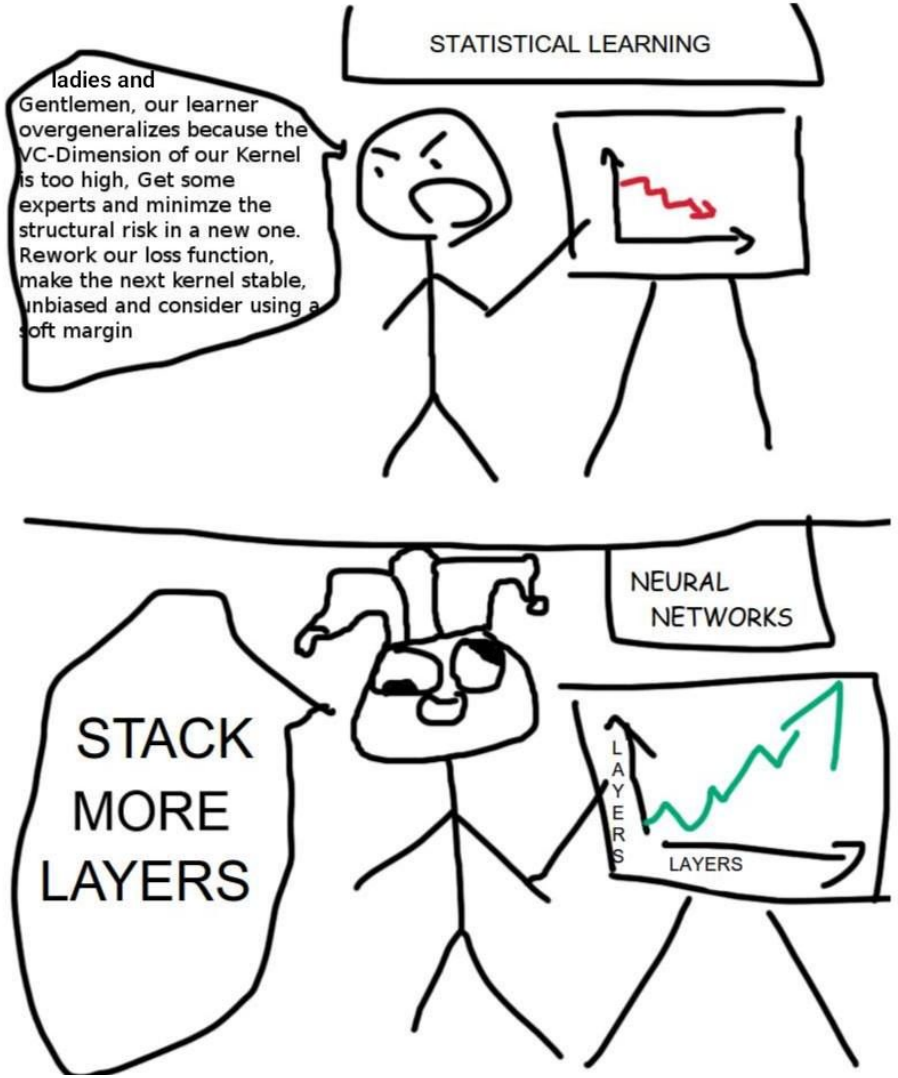


Regularised polynomial regression can show double descent



Is a higher degree but lower norm polynomial actually a more complex model?

High dimensional models are counter-intuitive…

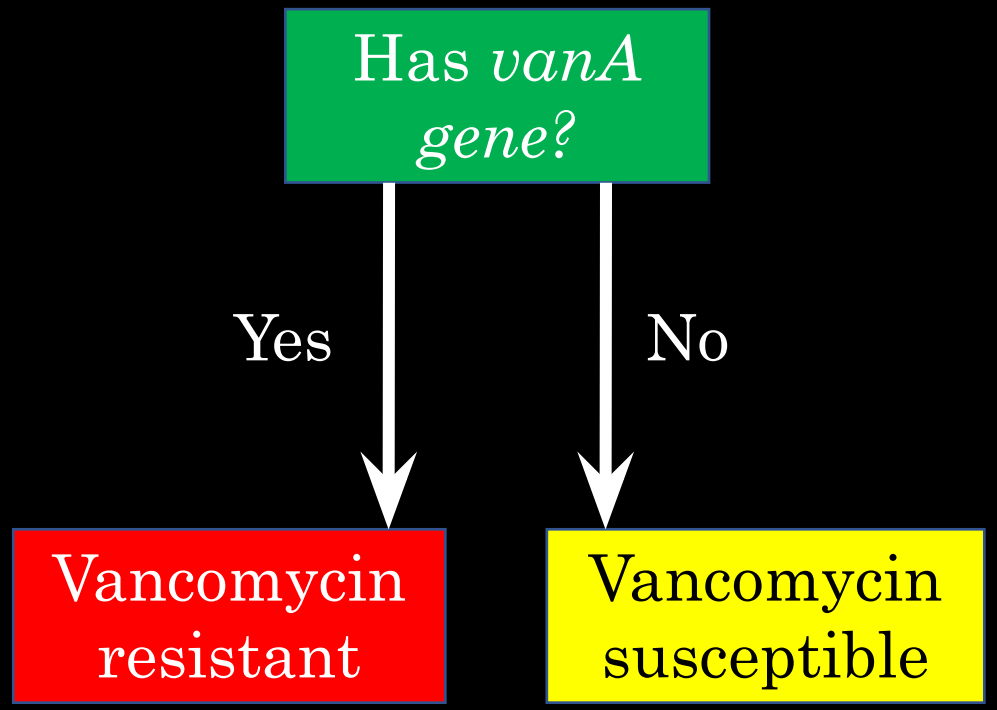


https://moultano.wordpress.com/2020/10/18/why-deep-learning-works-even-though-it-shouldnt/

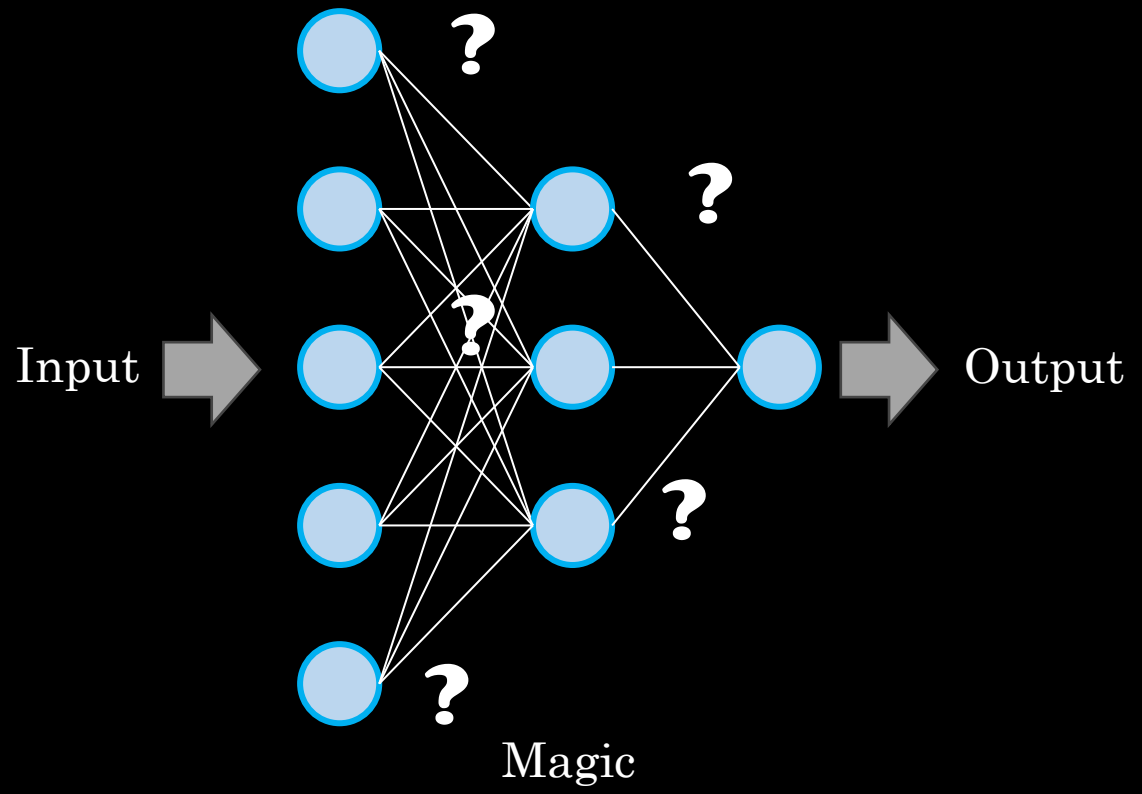
Interpretability

Some methods yield understandable (or almost understandable) rules, others do not

Decision tree



Artificial neural network



Tractability

- If the training data are necessarily high-dimensional, then a simpler classifier may be necessary

- (or we need to be more aggressive in our feature selection / extraction – see next lecture!)

Summary

- Choosing a classifier is non-trivial
- Choosing a training / testing strategy is non-trivial
- Evaluating accuracy is non-trivial
- ML in general must be approached with skepticism!