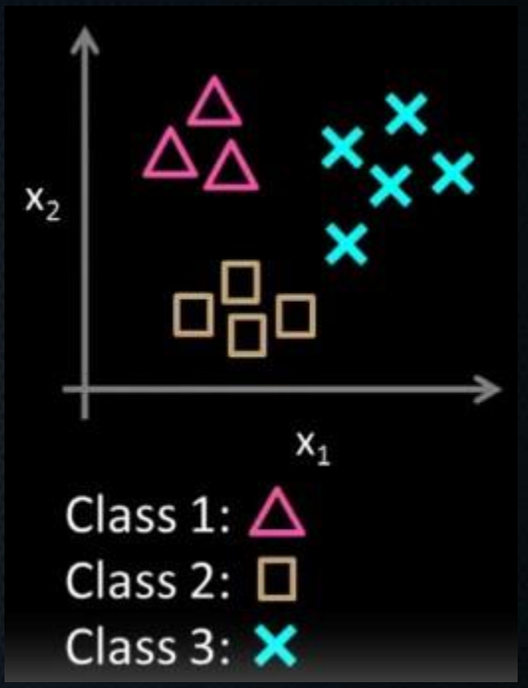
This presentation

- Dealing with > 2 classes in logistic regression
- ROC curves and ROC AUC
- Regularisation
- Getting hyperparameters

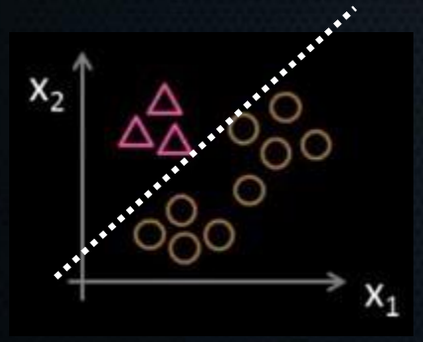
What if we have more than one class?

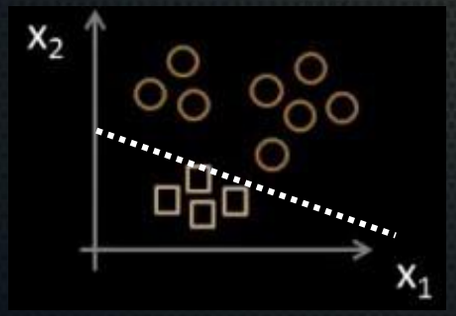
- What do you think we could do?

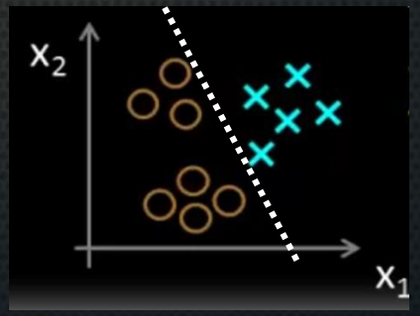


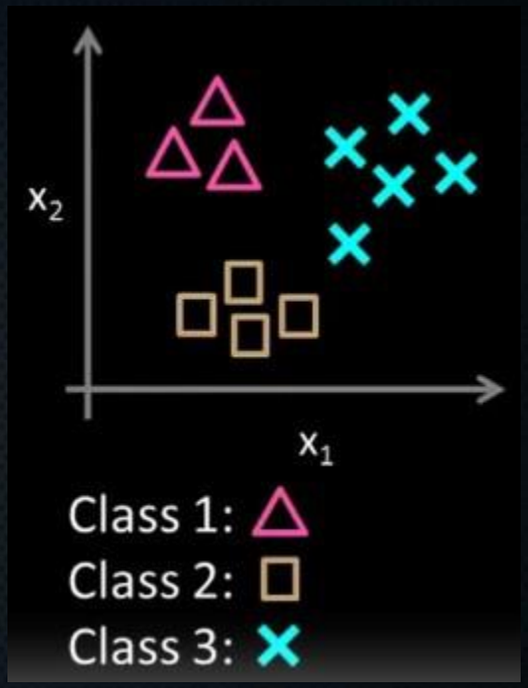
What if we have more than one class?

- What do you think we could do?
- Train a separate binary classifier for each instance:



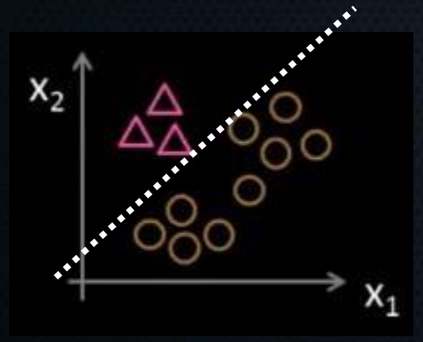




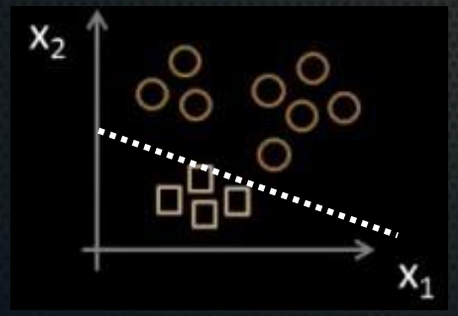


What if we have more than one class?

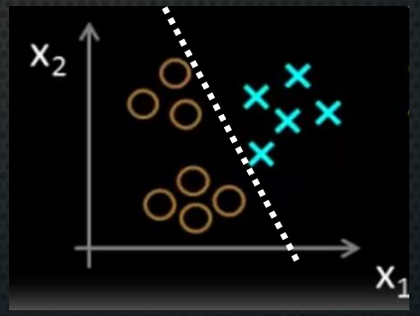
- What do you think we could do?
- Train a separate binary classifier for each instance:



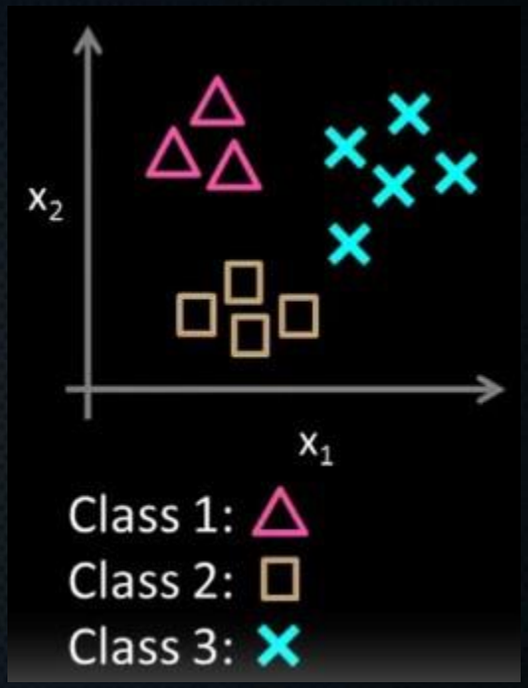
$h_\theta^{(1)}(x)$



$h_\theta^{(2)}(x)$



$h_\theta^{(3)}(x)$

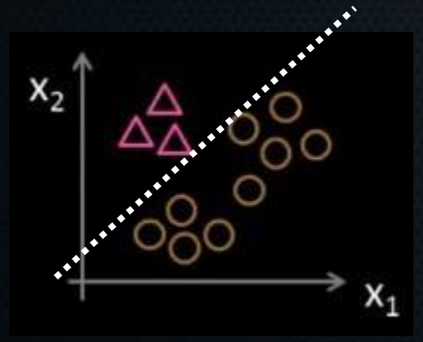


$P(y=i|x;\theta)$

For *i* classes, train *i* binary classifiers to predict that the point is class *i* given the data

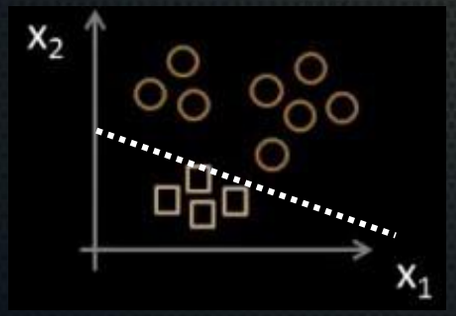
What if we have more than one class?

- What do you think we could do?
- Train a separate binary classifier for each instance:

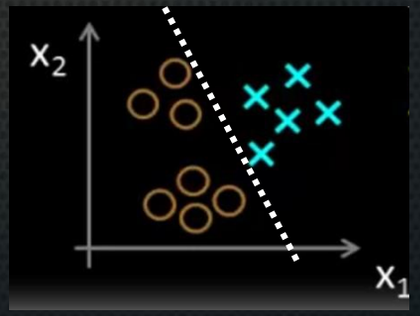


$h_\theta^{(1)}(x)$

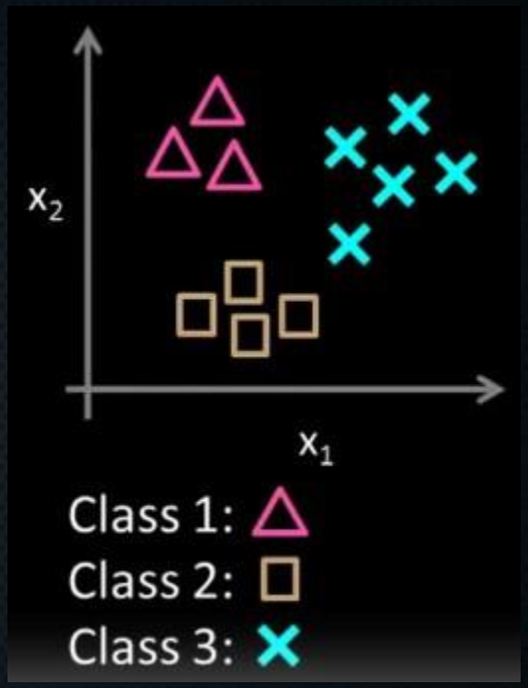
$P(y=i|x;\theta)$



$h_\theta^{(2)}(x)$



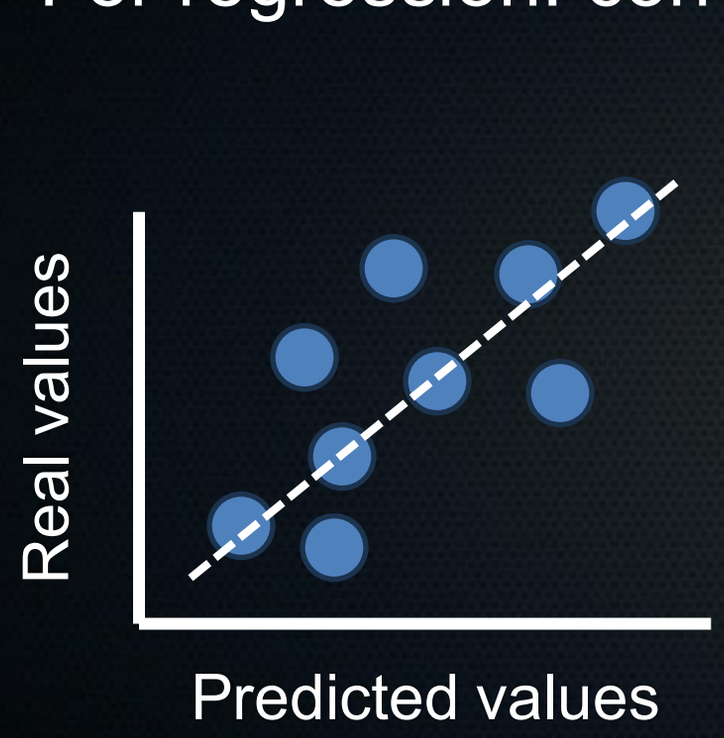
$h_\theta^{(3)}(x)$

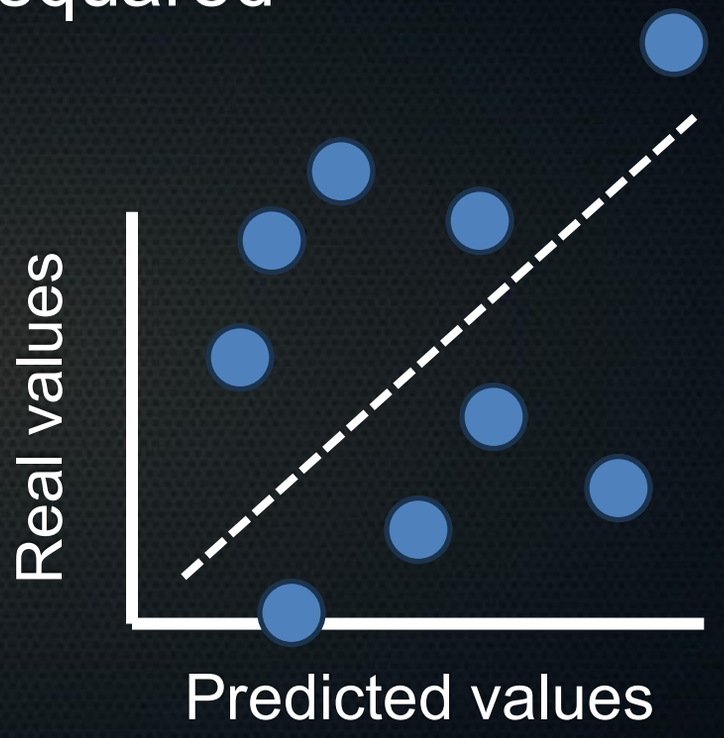


Predicted class = *max*(predictions)

How well are we doing?

- For regression: correlations or R-squared





How well are we doing?

- For regression: correlations or R-squared

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

Where,

r = Pearson Correlation Coefficient

x_i = x variable samples

y_i = y variable sample

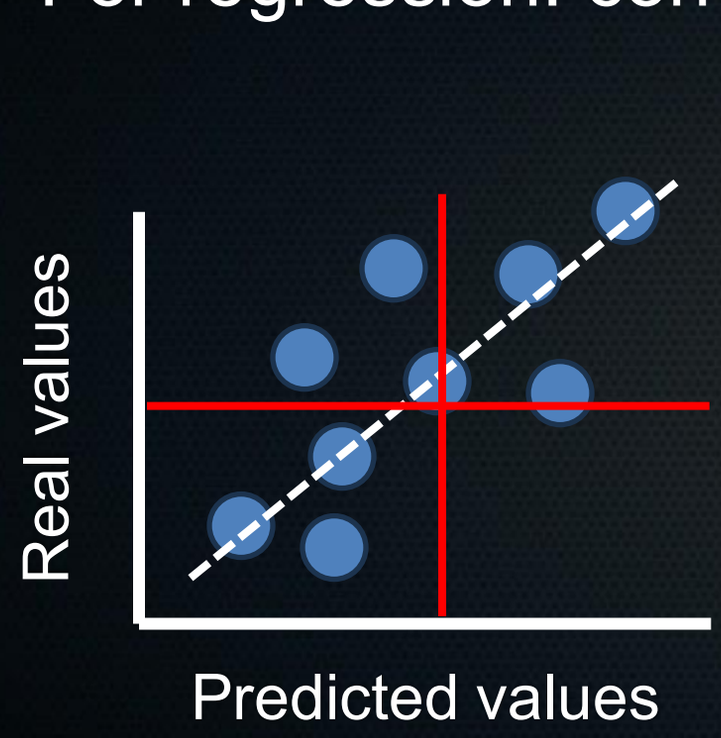
$\bar{x}$ = mean of values in x variable

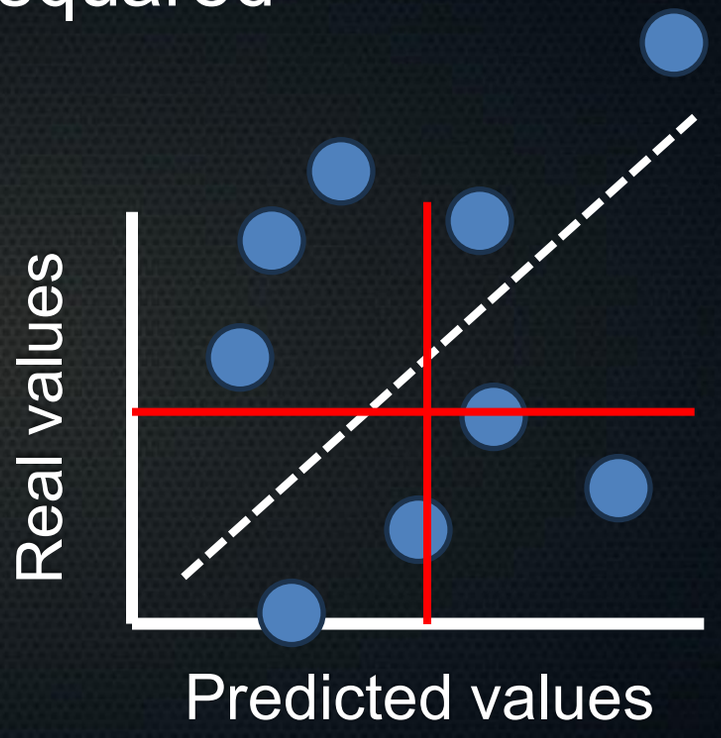
$\bar{y}$ =mean of values in y variable

https://medium.com/@abdallahashraf90x/all-you-need-to-know-about-correlation-for-machine-learning-e249fec292e9

How well are we doing?

- For regression: correlations or R-squared

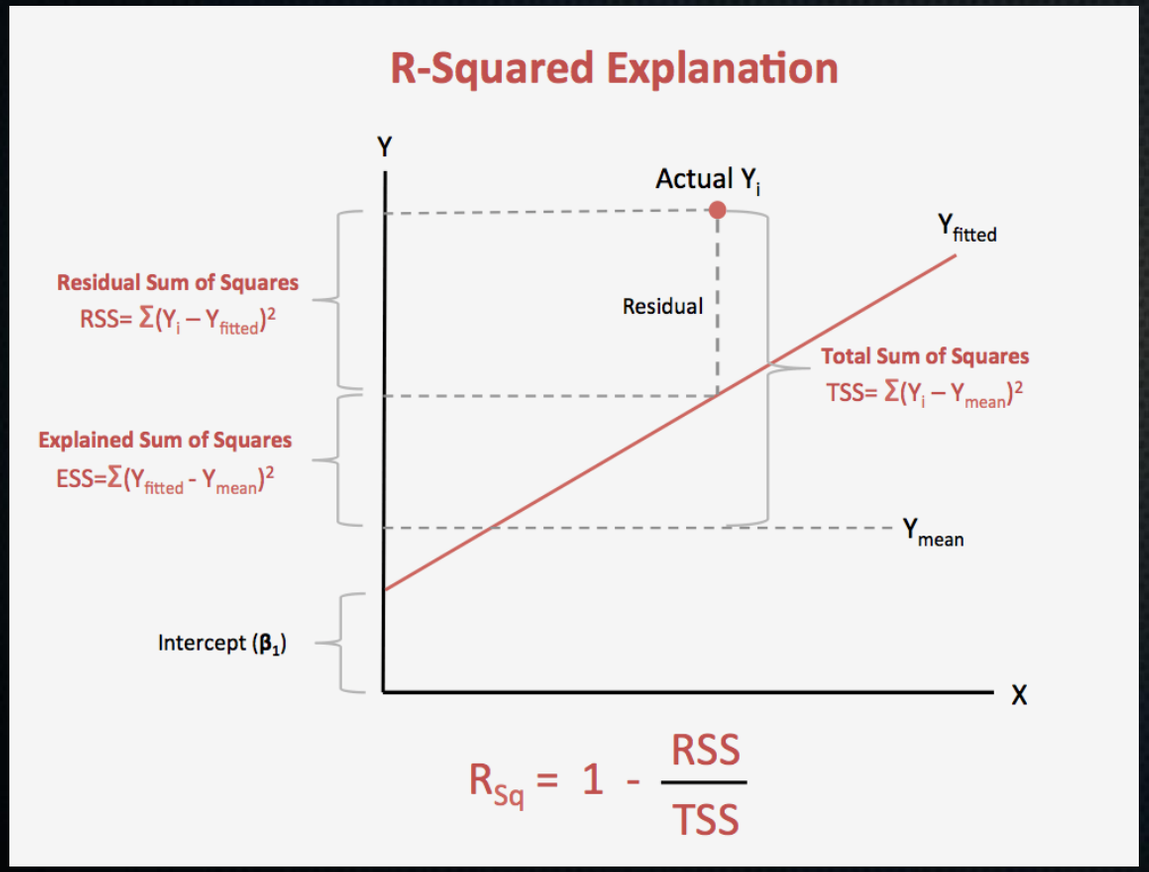




$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{}$$

How well are we doing?

- For regression: correlations or R-squared

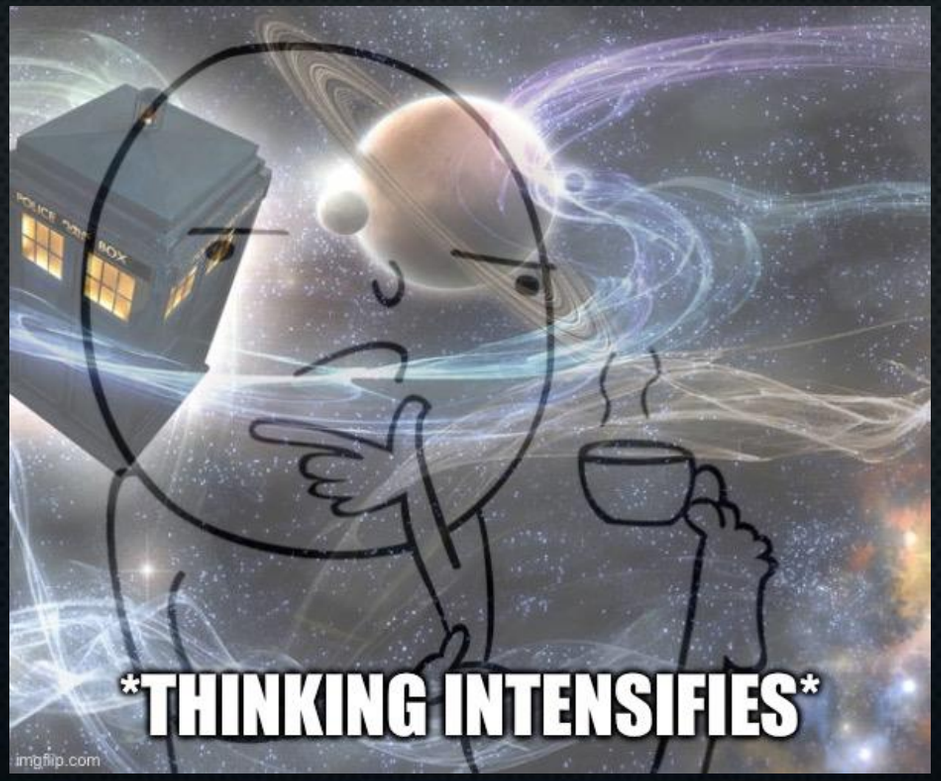


- R-squared of 1: every point perfectly on the line → the regression perfectly explains everything
- R-squared of 0: line explains absolutely nothing. Sorry bro.

https://www.machinelearningplus.com/machine-learning/complete-introduction-linear-regression-r/

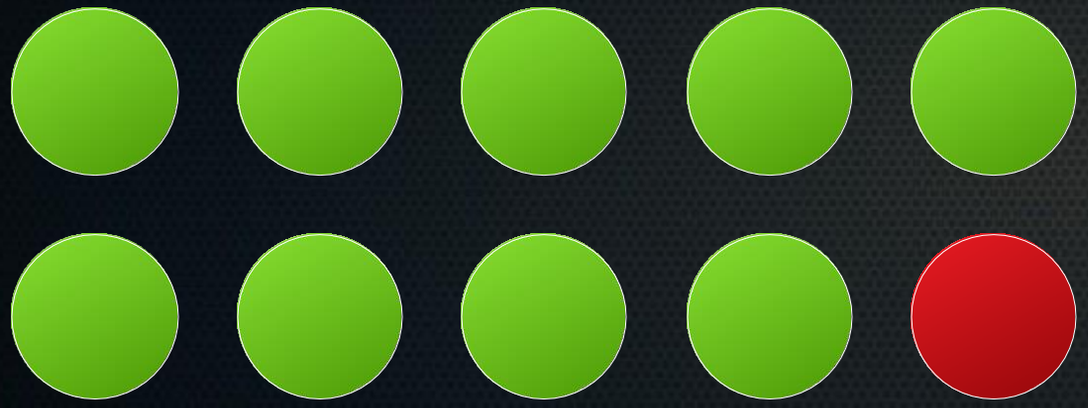
How well are we doing?

- And for classification?



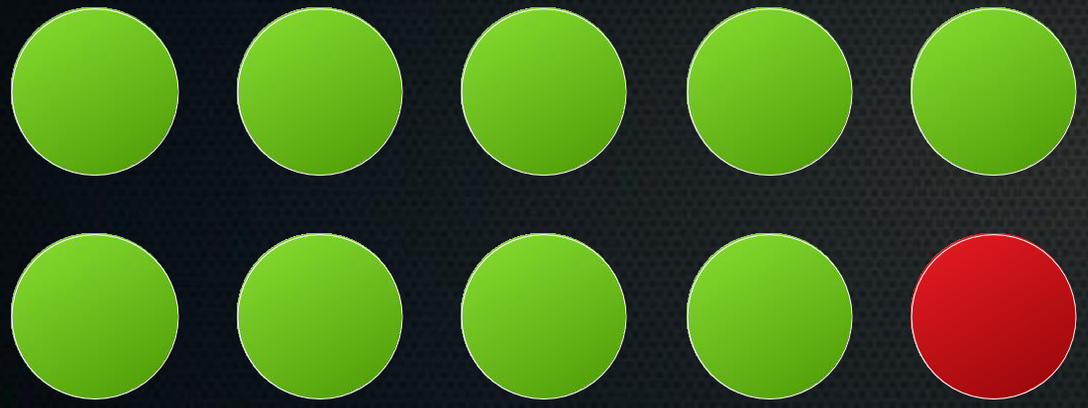
How well are we doing?

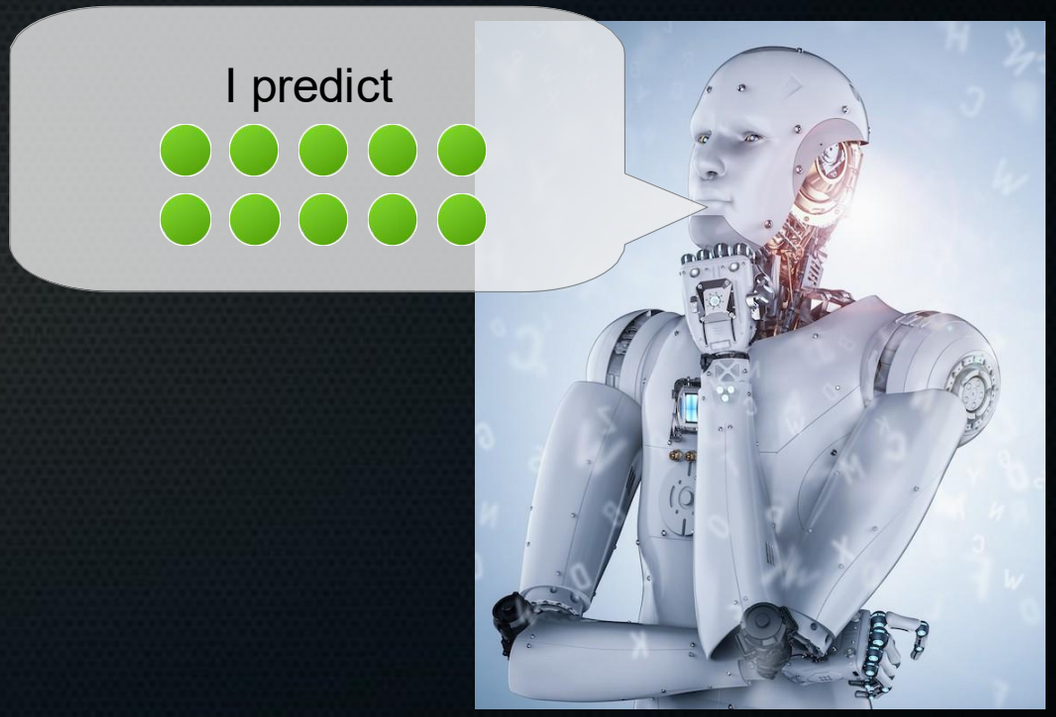
- What can we use?
- Idea: accuracy.



How well are we doing?

- What can we use?
- Idea: accuracy.

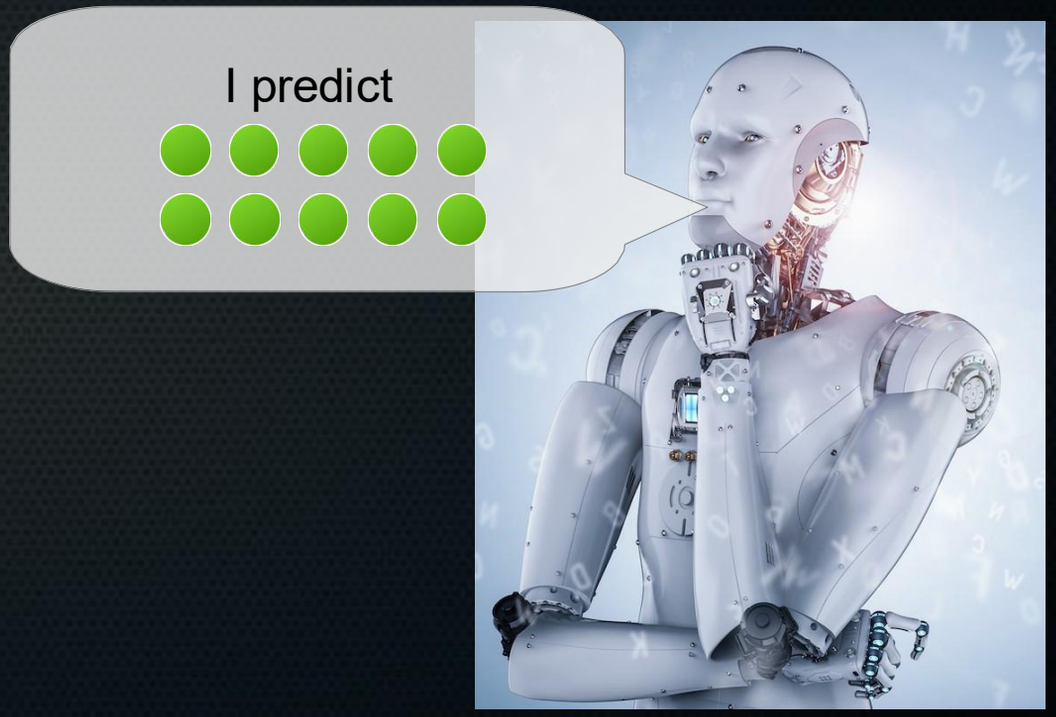




How well are we doing?

- What can we use?
- Idea: accuracy.

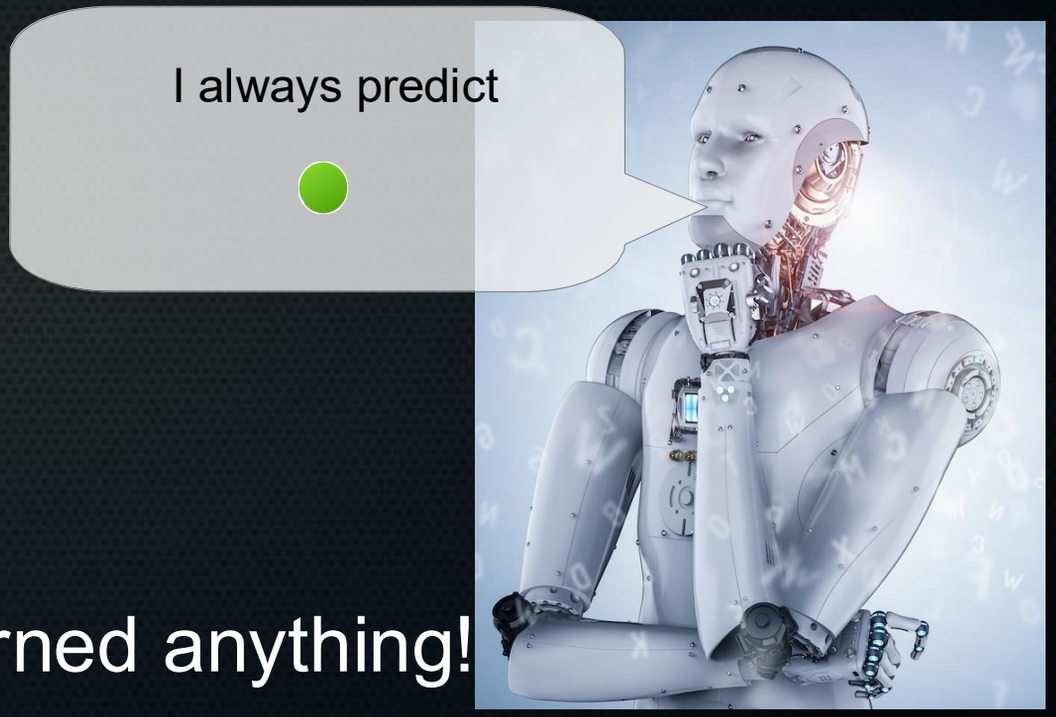
- Accuracy = 90% (9/10 correct).
  → seems pretty good!



How well are we doing?

- What can we use?
- Idea: accuracy.

- Accuracy = 90% (9/10 correct).
  → Lucky break! Classifier hasn‘t learned anything!



Need something else to measure performance

- Have 4 types of predictions:

		Reality	
		Cancer	**Not Cancer**
Model Prediction	**Cancer**	True Positive	False Positive
	Not Cancer	False Negative	True Negative

Need something else to measure performance

- Have 4 types of predictions:
  Call this a **confusion matrix**

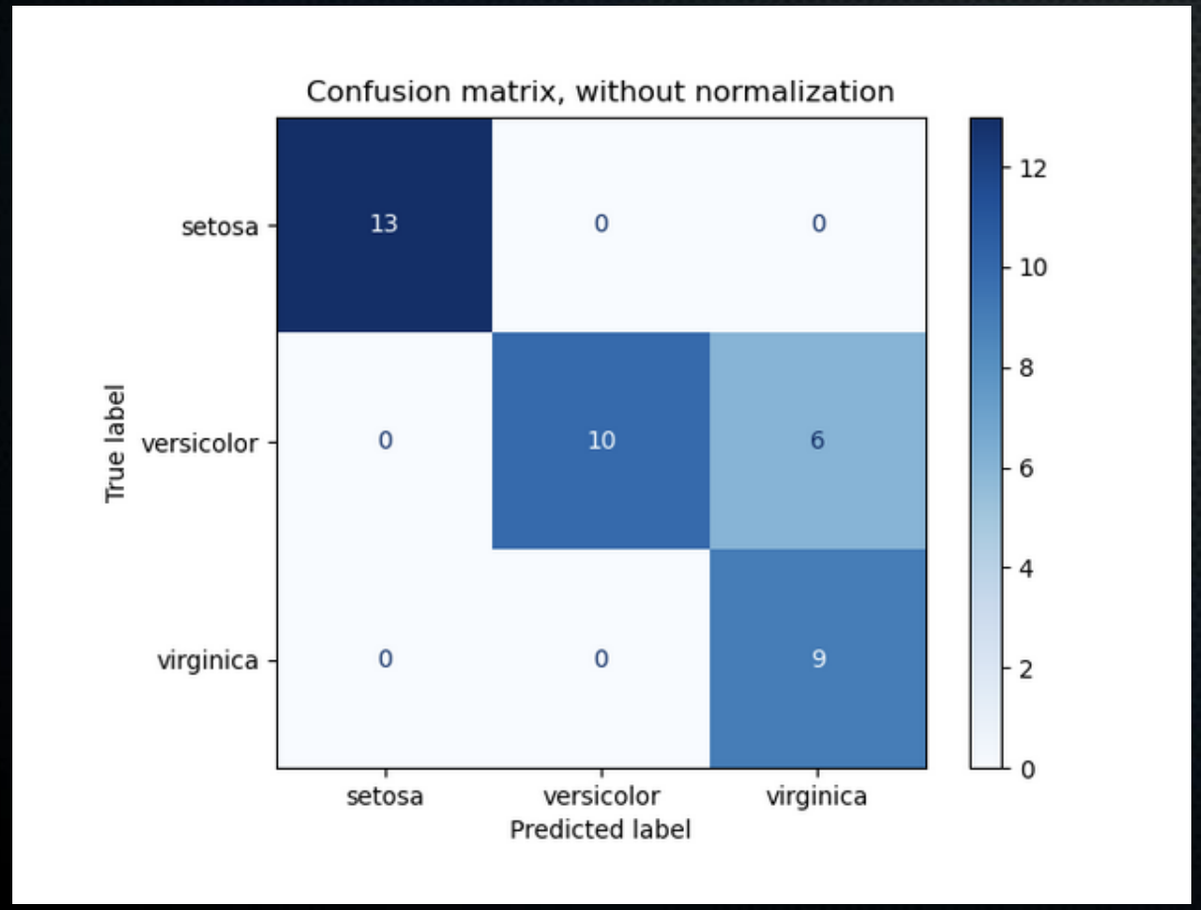
Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

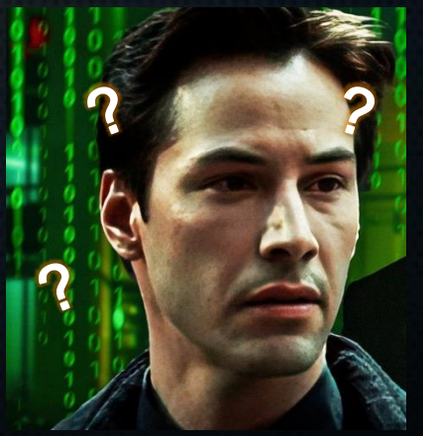
confused matrix noises

Need something else to measure performance

- Have 4 types of predictions:
  Can have multiple classes



*even more
confused matrix noises*



https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

Need something else to measure performance

- Have 4 types of predictions:
- We want to know both how many true positives we pick out from the test data (sensitivity) and how many true negatives we correctly classify as negative (specificity).

		Reality	
		Cancer	**Not Cancer**
Model Prediction	**Cancer**	True Positive	False Positive
	Not Cancer	False Negative	True Negative

Need something else to measure performance

- Have 4 types of predictions:
- We want to know both how many true positives we pick out from the test data (sensitivity) and how many true negatives we correctly classify as negative (specificity).

		Reality	
		Cancer	**Not Cancer**
Model Prediction	**Cancer**	True Positive	False Positive
	Not Cancer	False Negative	True Negative

Sensitivity (true positive rate)

$$\frac{True\ Positives}{True\ Positives + False\ Negatives}$$

→ What proportion of positives in the data do we correctly predict?

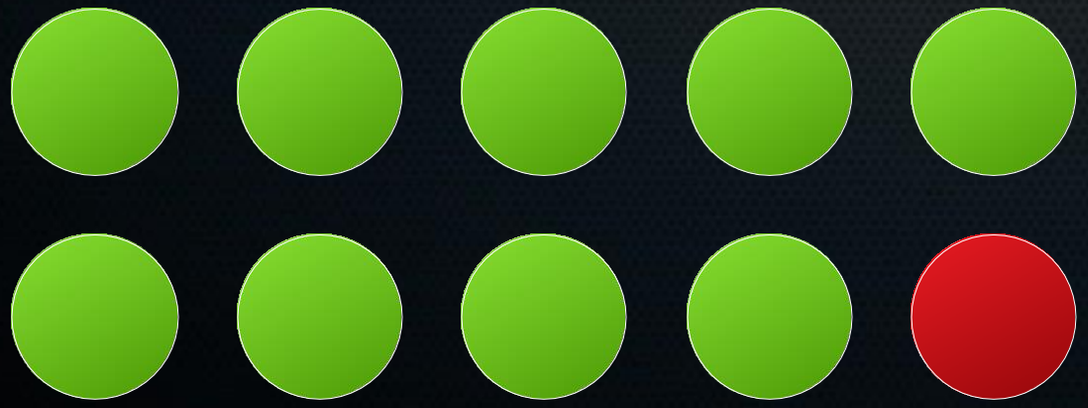
Specificity (true negative rate)

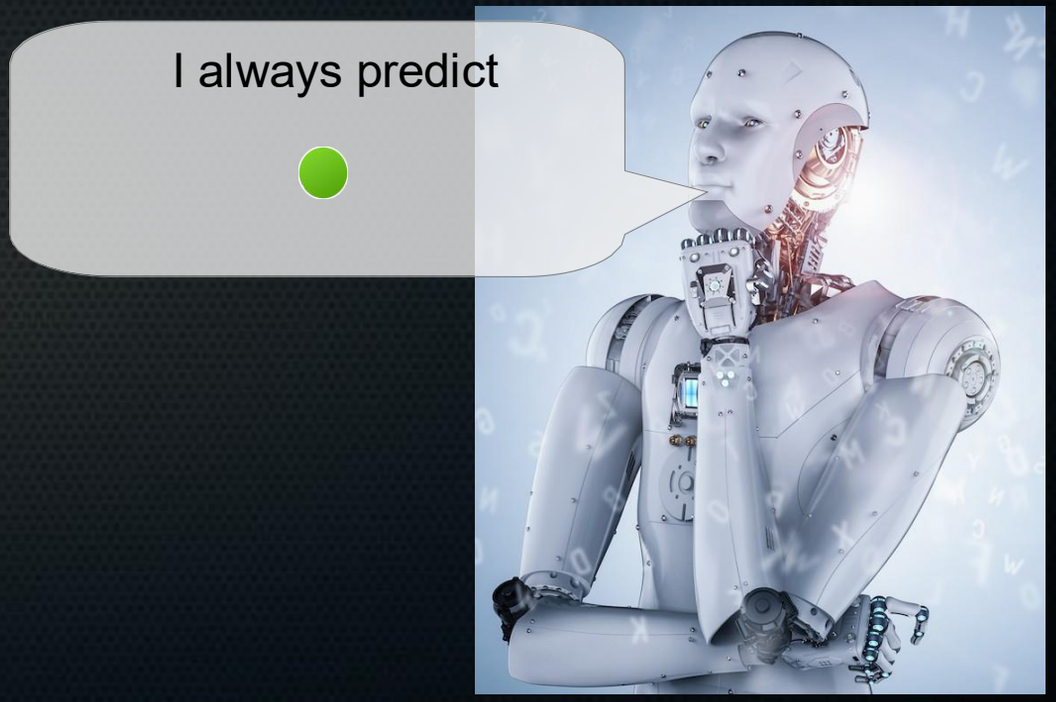
$$\frac{True\ Negatives}{True\ Negatives + False\ Positives}$$

→ What proportion of negatives in the data do we correctly predict?

How well are we doing?

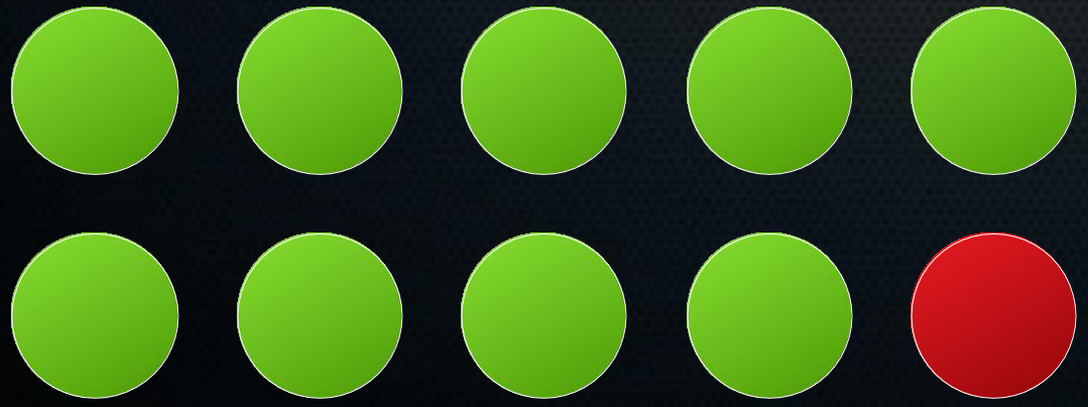
- Accuracy = 90% (9/10 correct).
  → Lucky break! Classifier hasn‘t learned anything!
- Sensitivity = 100%
- Specificity = 0%

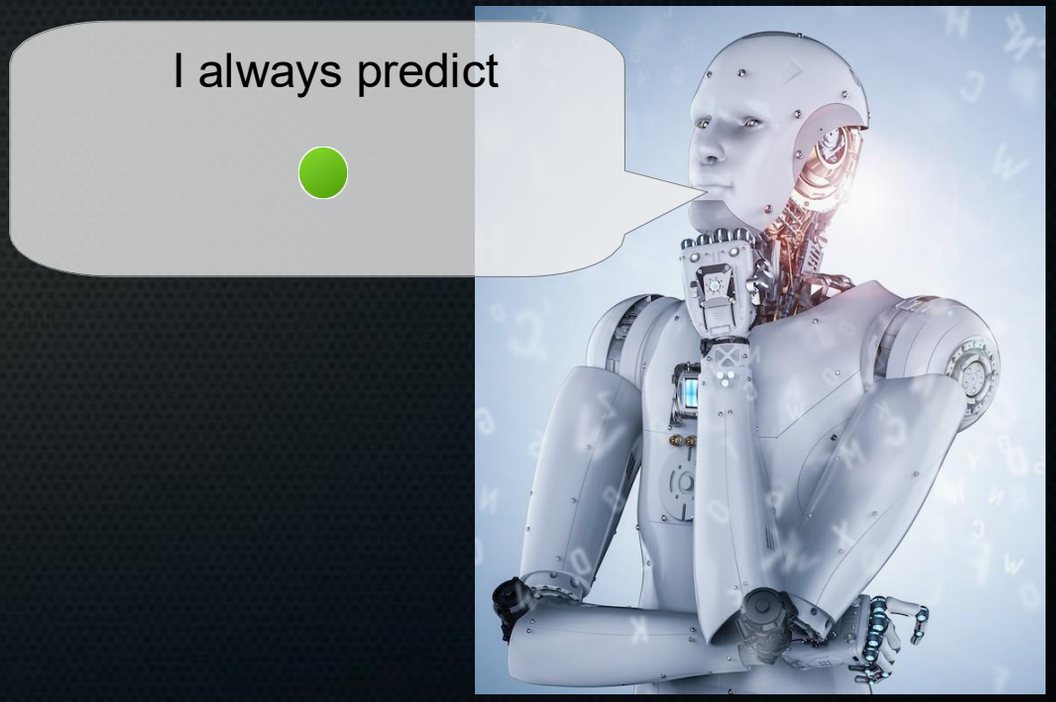




How well are we doing?

- Accuracy = 90% (9/10 correct).
  → Lucky break! Classifier hasn‘t learned anything!
- Sensitivity = 100% } Found all positives
- Specificity = 0% } By having 0 discerning ability





How well are we doing?

- Accuracy = 90% (9/10 correct).
  → Lucky break! Classifier hasn‘t learned anything!
- Sensitivity = 100%
- Specificity = 0%

Question:
So what’s the best specificity/sensitivity trade-off?

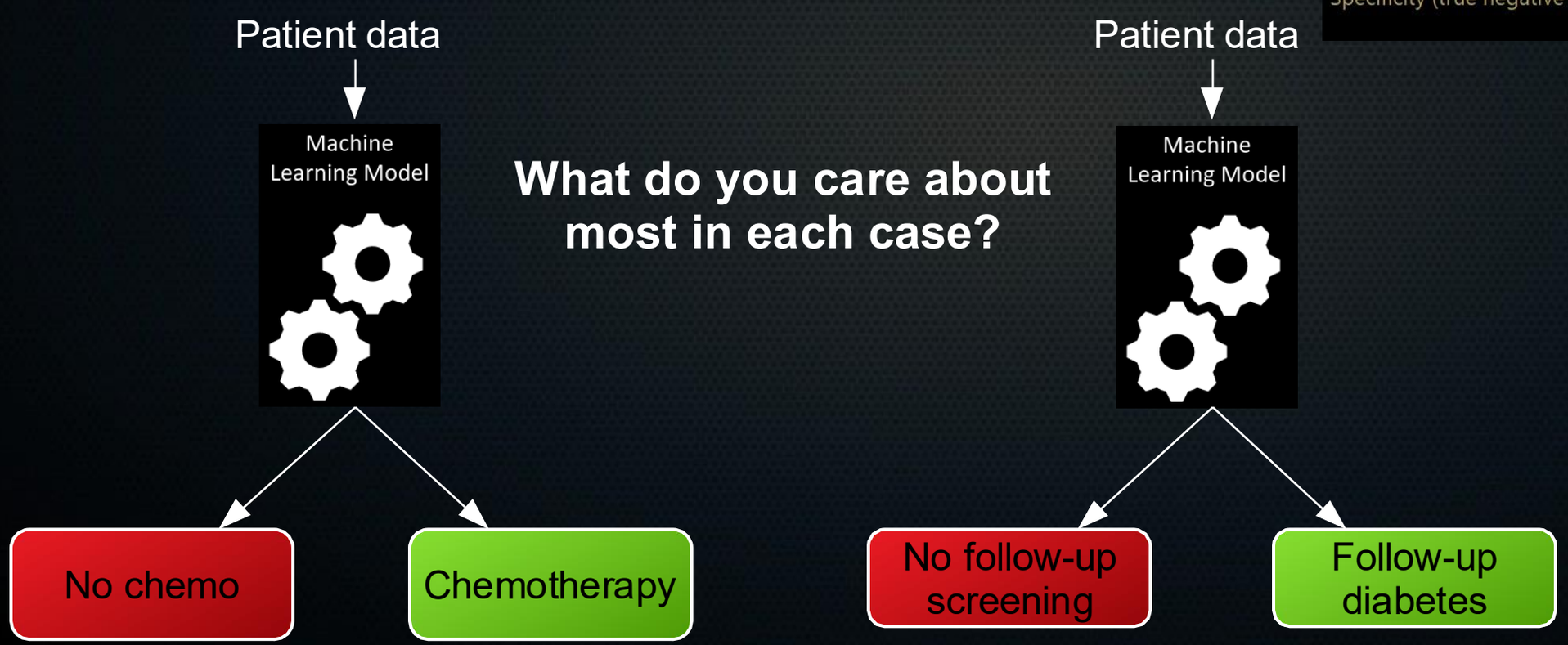
Sensitivity (true positive rate) $\frac{True\ Positives}{True\ Positives + False\ Negatives}$

Specificity (true negative rate) $\frac{True\ Negatives}{True\ Negatives + False\ Positives}$

ROC curve

- What is the best balance between sensitivity and specificity?
  - Depends on your application:

Sensitivity (true positive rate)	$\frac{True\ Positives}{True\ Positives + False\ Negatives}$
Specificity (true negative rate)	$\frac{True\ Negatives}{True\ Negatives + False\ Positives}$



ROC curve

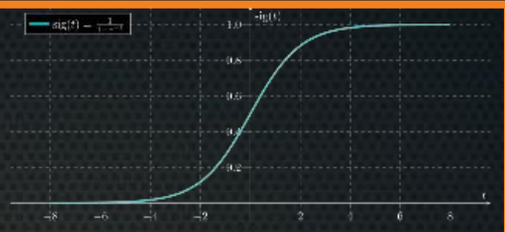
- What is the best balance between sensitivity and specificity?
  - Depends on your application:

Sensitivity (true positive rate)	$\frac{True\ Positives}{True\ Positives + False\ Negatives}$
Specificity (true negative rate)	$\frac{True\ Negatives}{True\ Negatives + False\ Positives}$

ROC curve

- How do you implement a focus on specificity or sensitivity?

  - How do we work with this? $h_\theta(x) = \frac{1}{1+e^{(\theta^T \cdot x)}}$

    - Interpret outcome of $h_\theta(x)$ as probability that class = 1 given the features. Example:

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{Tumor size} \\ \text{Neovascularisation level} \end{bmatrix}$$

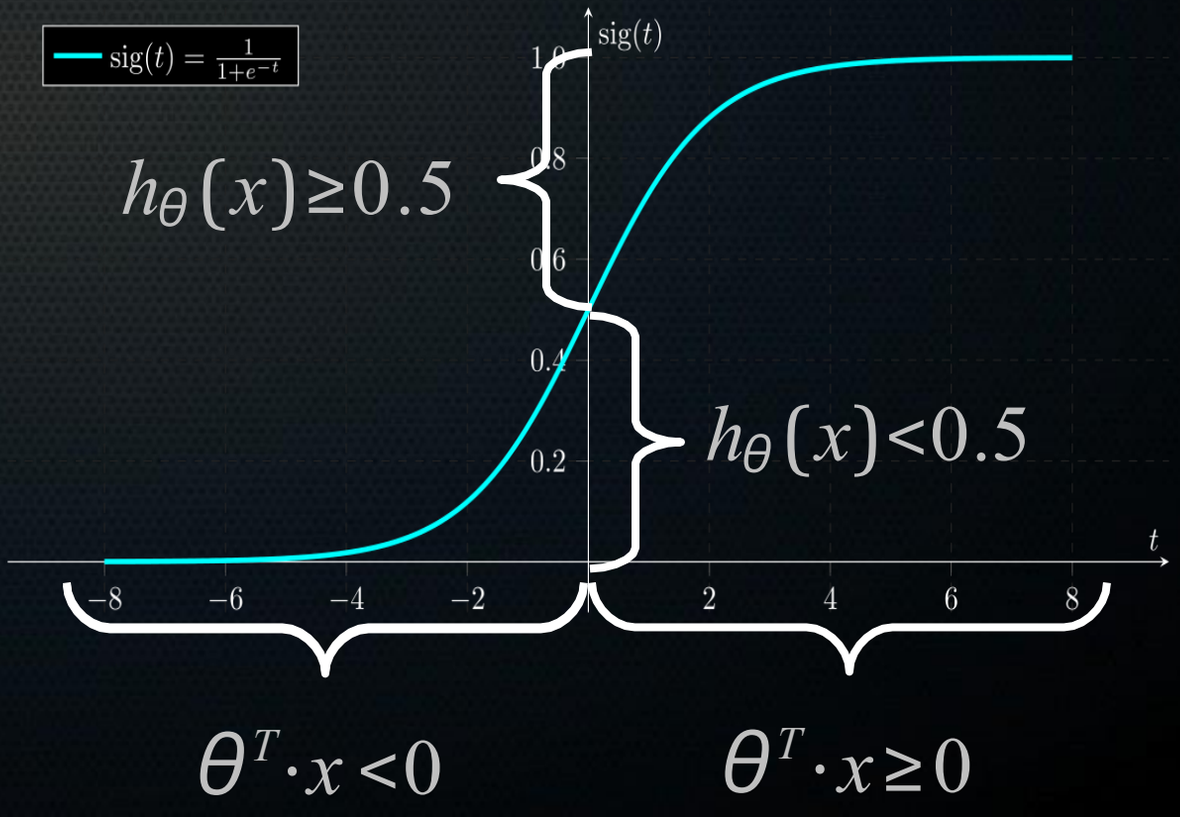
$h_\theta(x) = 0.8 \longrightarrow$ 80% chance of tumor being malignant (class 1)

100% - 80% → 20 % chance of being benign (class 0)

ROC curve

How do you implement a focus on specificity or sensitivity?

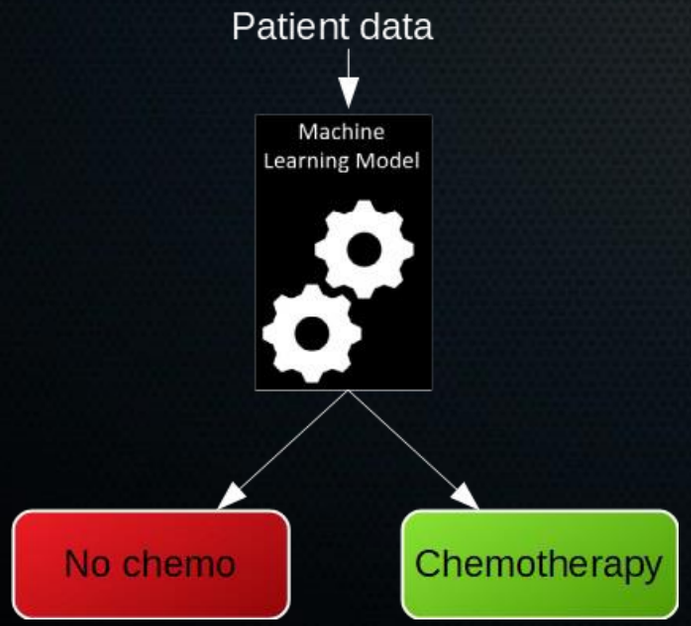
You tell me!

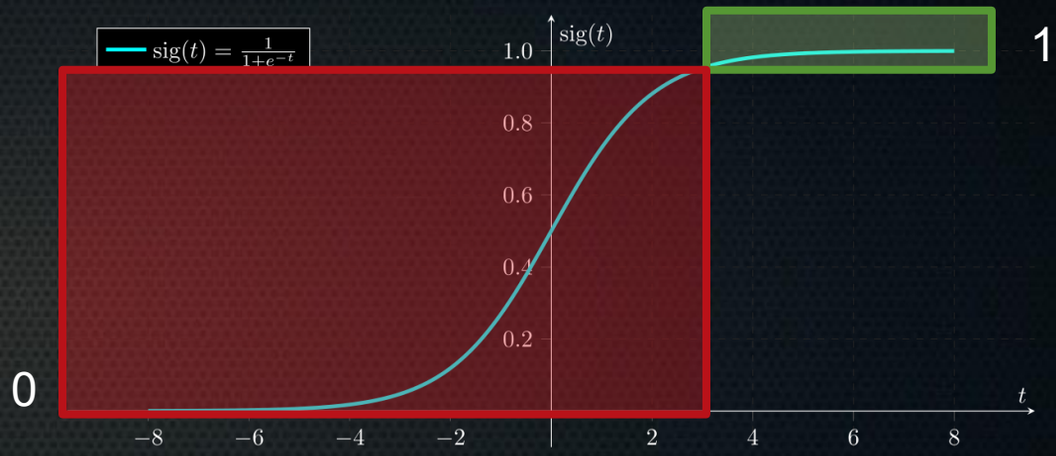


ROC curve

- How do you implement a focus on specificity or sensitivity?
- We could say:

$$\text{if } h_\theta(x) \geq 0.95 \text{ classify as positive}$$
$$\text{else classify as negative}$$

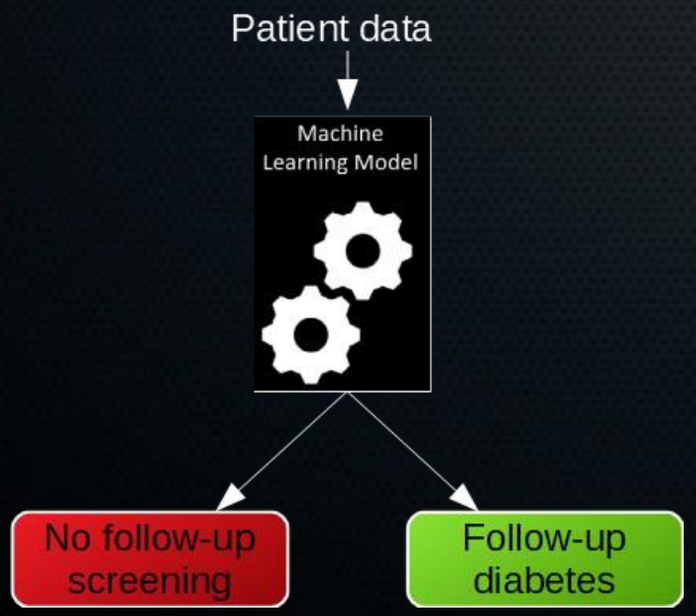


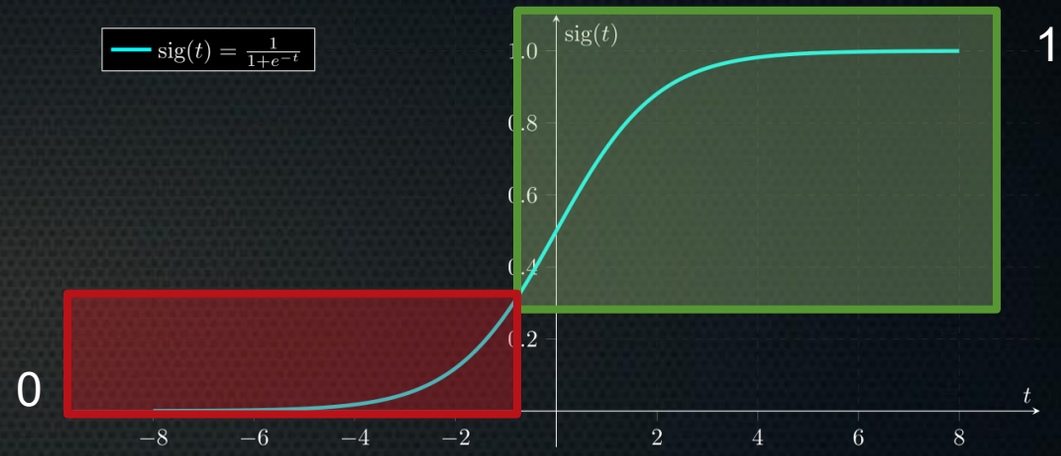


ROC curve

- How do you implement a focus on specificity or sensitivity?
- We could say:

$$\text{if } h_\theta(x) \geq 0.3 \text{ classify as positive}$$
$$\text{else classify as negative}$$

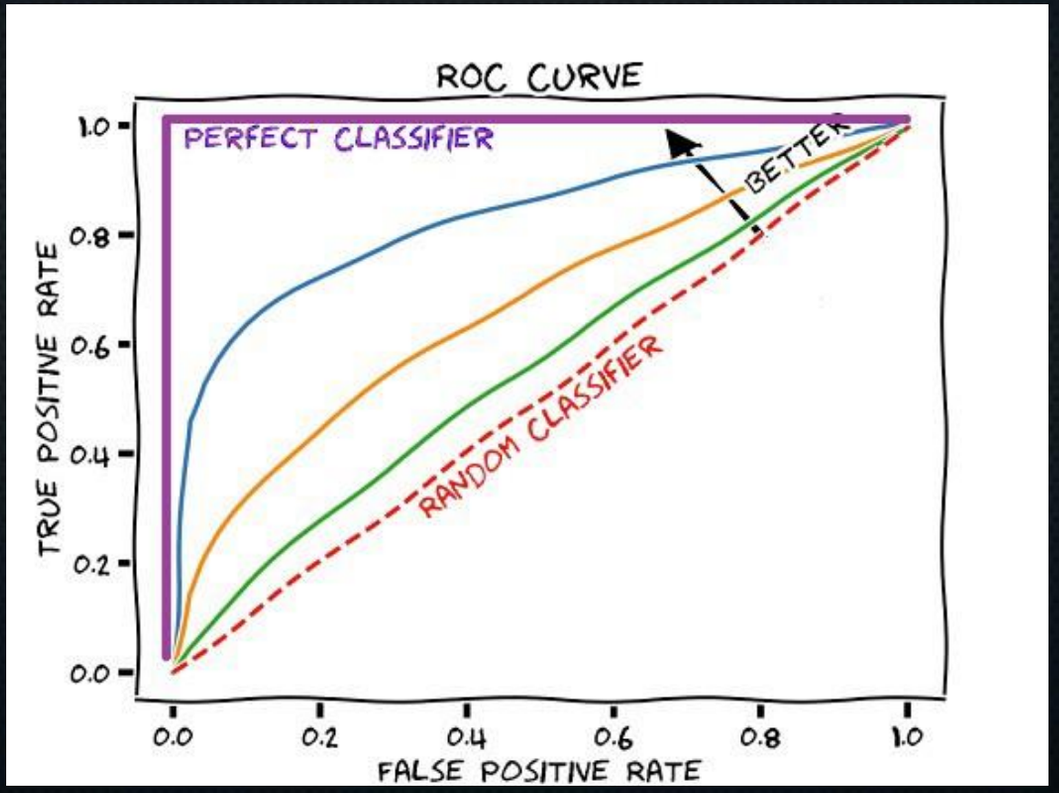




ROC curve

- What if we see how our classifier performs for all possible thresholds?

Sensitivity

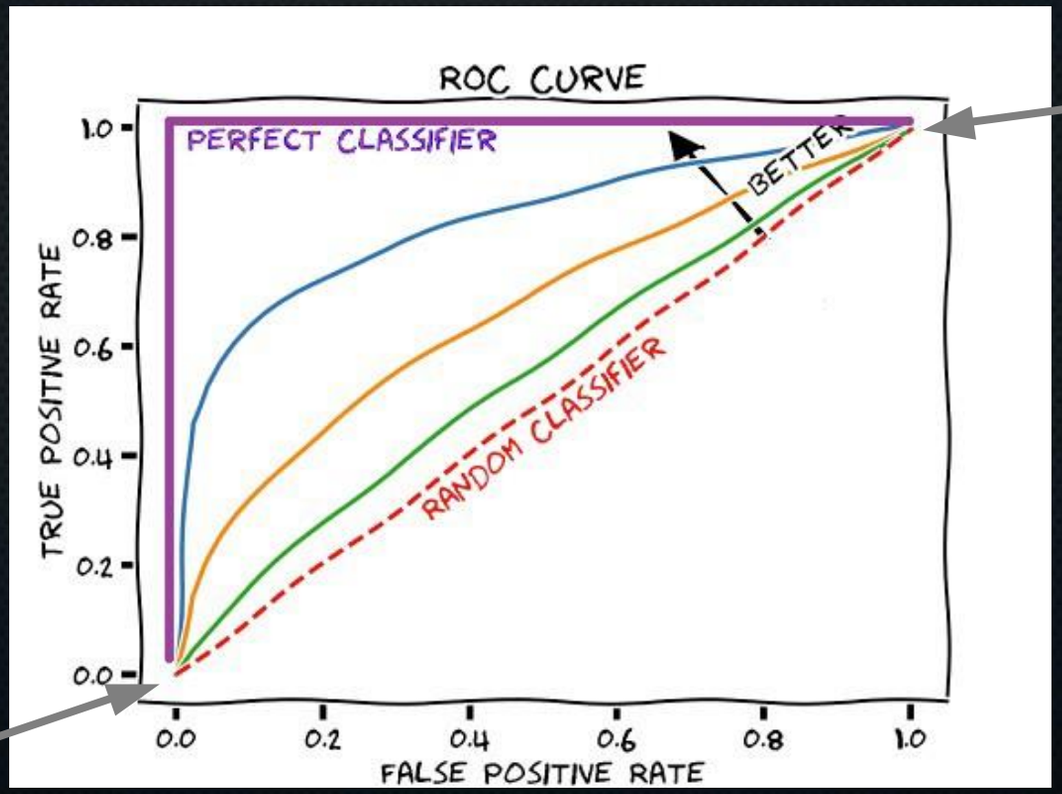


1-specificity

Source: https://glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc/

ROC curve

- What if we see how our classifier performs for all possible thresholds?



Sensitivity

1-specificity

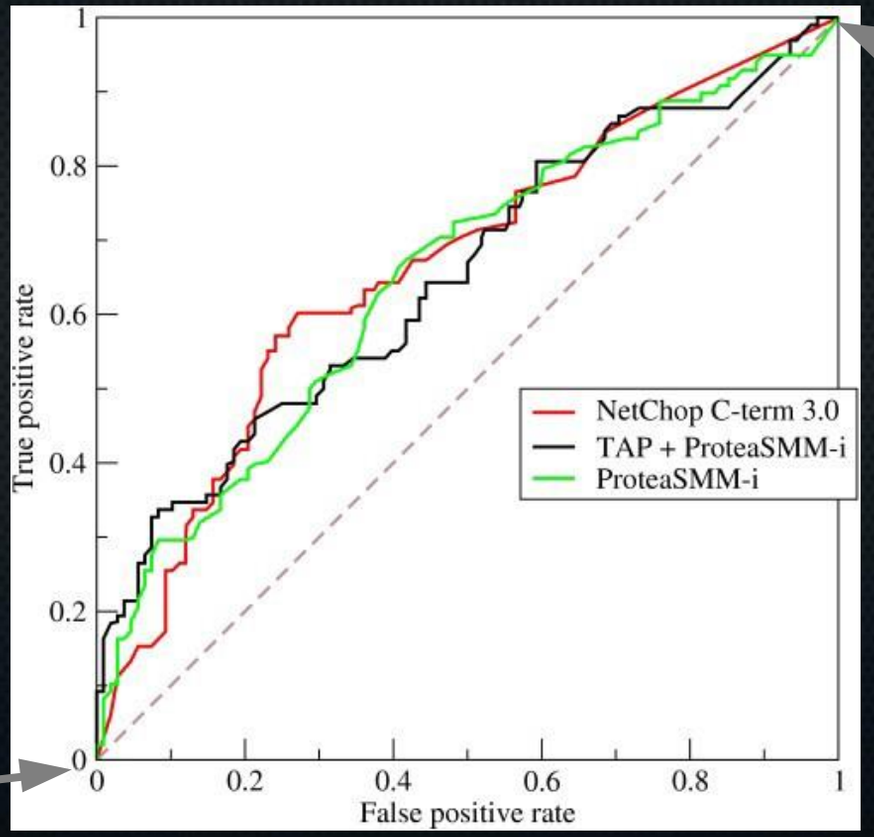
Classify as 1 only if sigmoid >= 0
(i.e. classify everything as positive)

Classify as 1 only if sigmoid > 1
(i.e. classify everything as negative)

Source: https://glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc/

ROC curve

- What if we see how our classifier performs for all possible thresholds?



Sensitivity

1-specificity

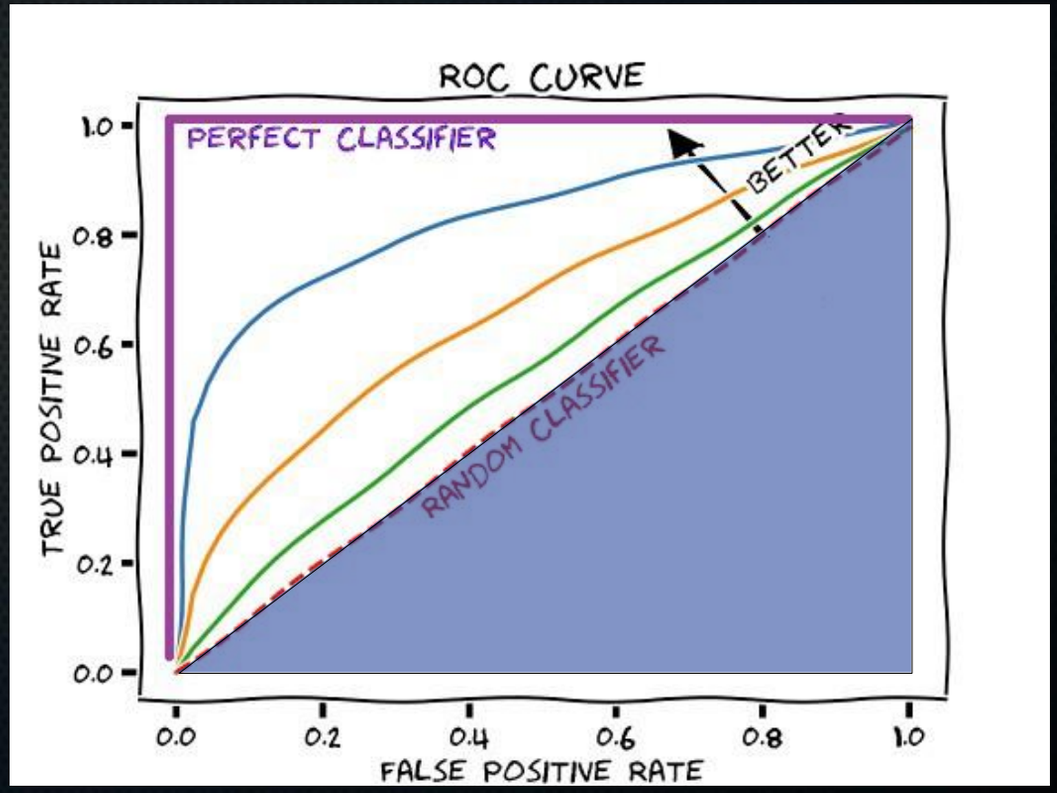
Classify as 1 only if sigmoid >= 0
(i.e. classify everything as positive)

Classify as 1 only if sigmoid > 1
(i.e. classify everything as negative)

Source: https://en.wikipedia.org/wiki/Receiver_operating_characteristic#/media/File:Roccurves.png

Area under the ROC curve (AUC)

- How to compare classifiers numerically?
- Coin-flip classifier (random guess)
  AUC = 0.5



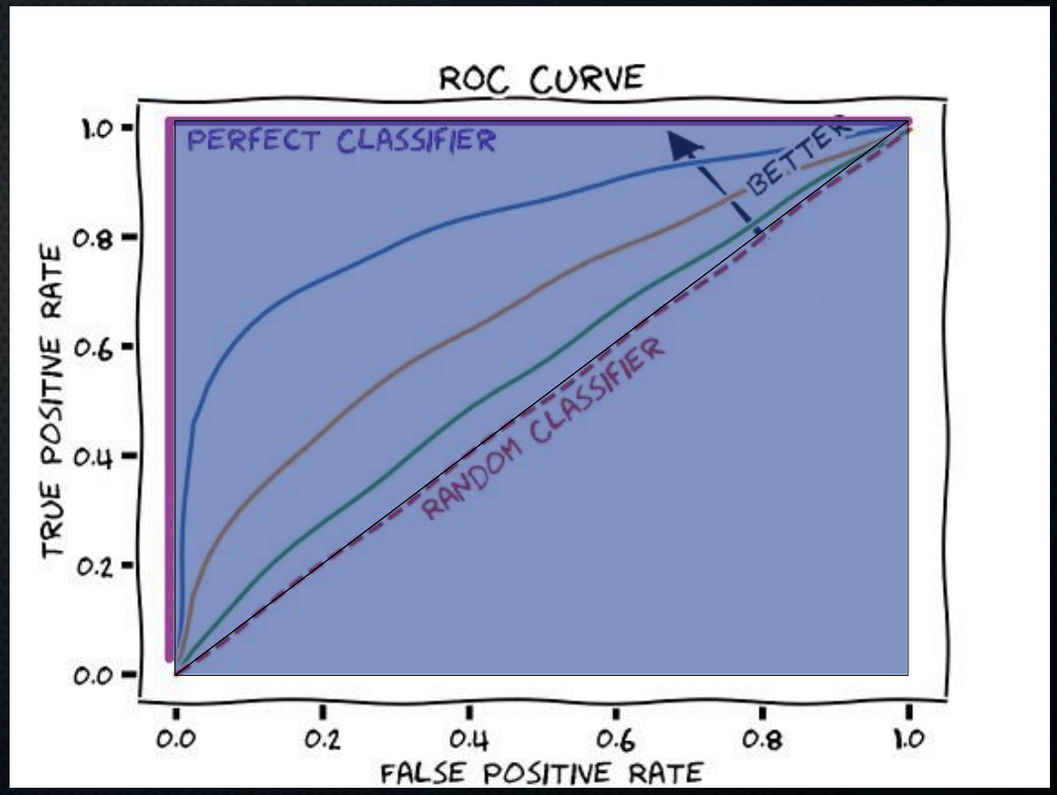
Sensitivity

1-specificity

Source: https://glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc/

Area under the ROC curve (AUC)

- How to compare classifiers numerically?
- Coin-flip classifier (random guess)
  AUC = 0.5
- Best possible classifier (positive cases all predicted 1)
  AUC = 1



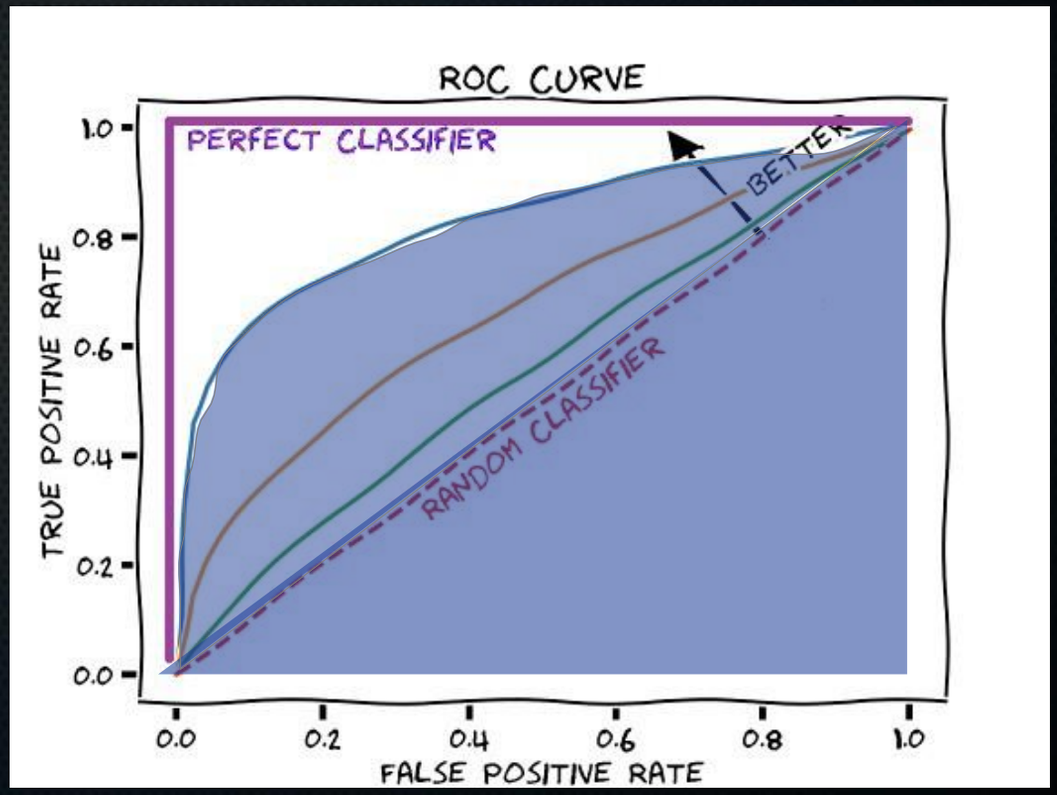
Sensitivity

1-specificity

Source: https://glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc/

Area under the ROC curve (AUC)

- How to compare classifiers numerically?
- Coin-flip classifier (random guess)
  AUC = 0.5
- Best possible classifier (positive cases all predicted 1)
  AUC = 1
- In-between: ~0.8, for instance



Sensitivity

1-specificity

Source: https://glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc/

Pitfall AUC (of ROC curve)

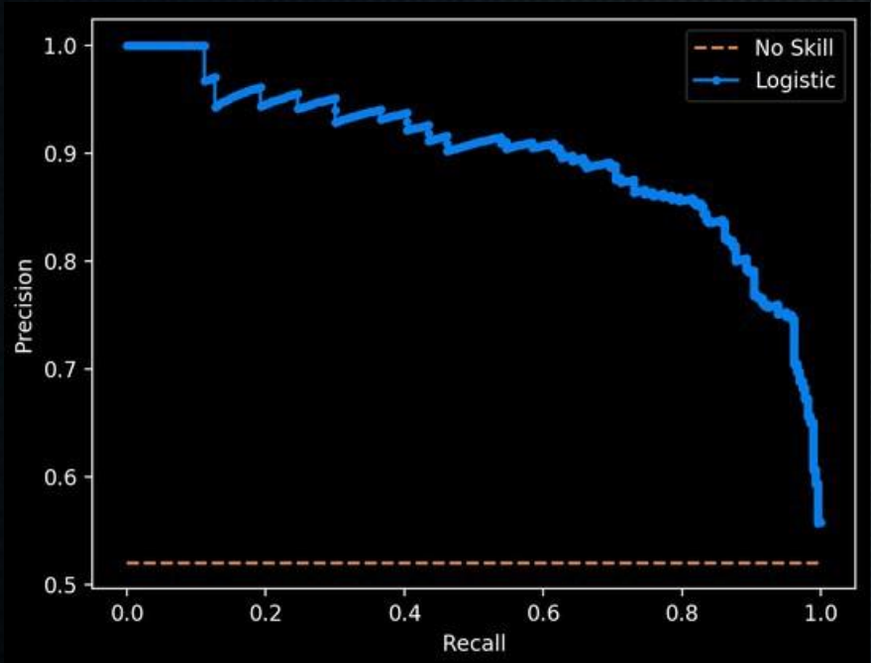
- All these values are determined by ratios
- If data is sampled from general population, $N_{negative} >> N_{positive}$ for a disease
- For specificity, because true negatives is a *huge number*, more false positives matter much less. Specificity becomes overly optimistic (especially if, later, you run your classifier in a clinical setting where $N_{negative}$ is much smaller!)

		Reality	
		Cancer	**Not Cancer**
Model Prediction	**Cancer**	True Positive	False Positive
	Not Cancer	False Negative	True Negative

Sensitivity (true positive rate)	$\frac{True\ Positives}{True\ Positives + False\ Negatives}$
Specificity (true negative rate)	$\frac{True\ Negatives}{True\ Negatives + False\ Positives}$

Pitfall AUC (of ROC curve)

- Instead, should then look at precision-recall curve.



Precision-Recall Plot for a No Skill Classifier and a Logistic Regression Model

Source: https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

		Reality: Cancer	Reality: Not Cancer
Model Prediction	Cancer	True Positive	False Positive
	Not Cancer	False Negative	True Negative

Recall =

Sensitivity (true positive rate) $\frac{True\ Positives}{True\ Positives + False\ Negatives}$

Specificity (true negative rate) $\frac{True\ Negatives}{True\ Negatives + False\ Positives}$

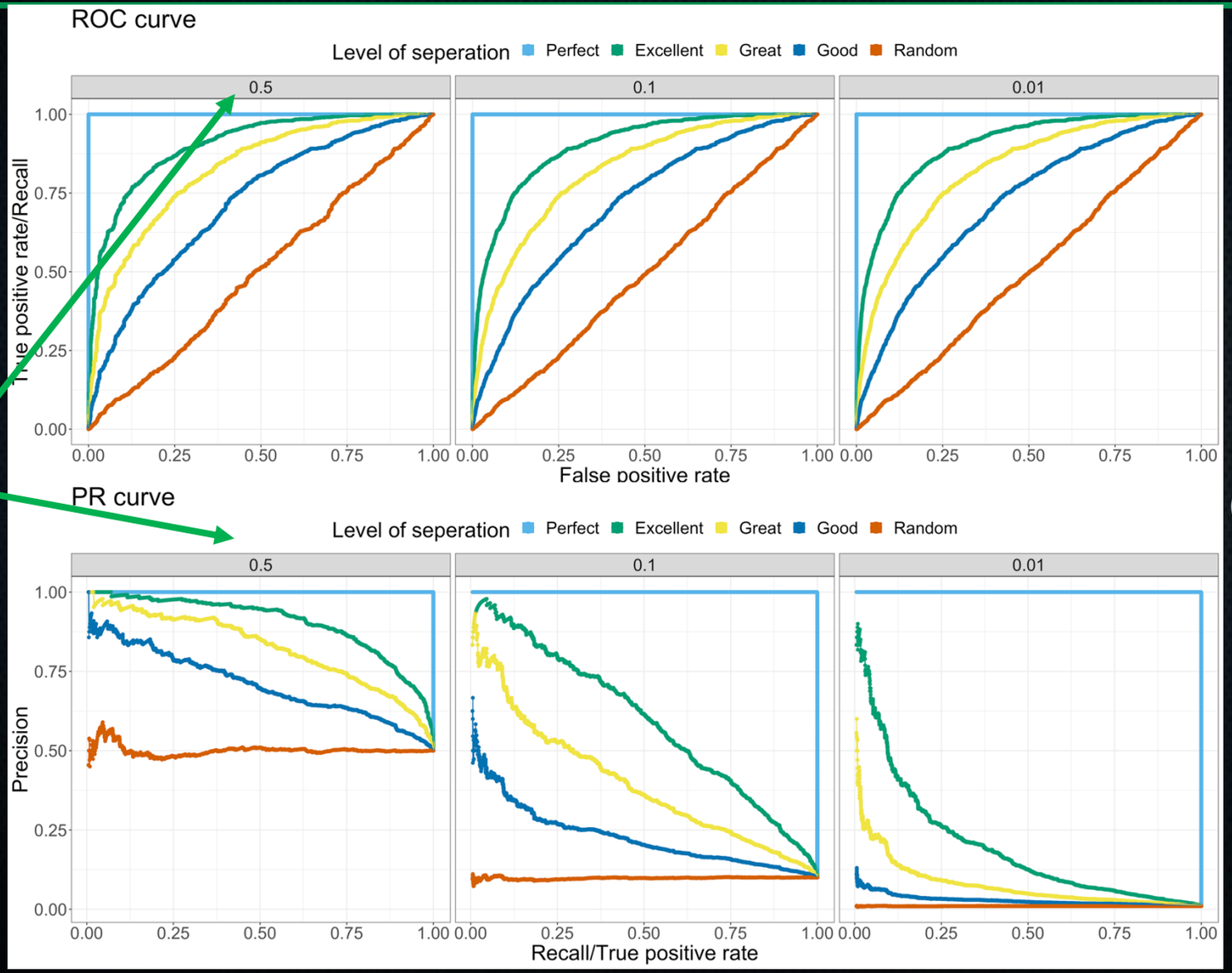
$$\underbrace{\frac{True\ Positives}{(True\ Positives + False\ Positives)}}_{\text{Precision}}$$

- Or make sure to train on balanced data!

Pitfall AUC (of ROC curve)

https://sinyi-chou.github.io/classification-pr-curve/

Proportion of positive class



$$\frac{\textit{True Positives}}{\textit{(True Positives + False Positives)}}$$

Precision

Pitfall AUC (of ROC curve)

Use precision recall curve if you *care about the positive class* and your *data is imbalanced*. ROC weighs positive and negative predictions equally, so large proportion of negatives makes it unsuitable

Knock knock: reality is complex

Deep ROC Analysis and AUC as Balanced Average Accuracy, for Improved Classifier Selection, Audit and Explanation

Publisher: IEEE | Cite This | PDF

André M. Carrington ; Douglas G. Manuel ; Paul W. Fieguth ; Tim Ramsay ; Venet Osmani ; Bernhard Wernly ; Carol Bennett ; Steve… All Authors

15 Paper Citations | 694 Full Text Views

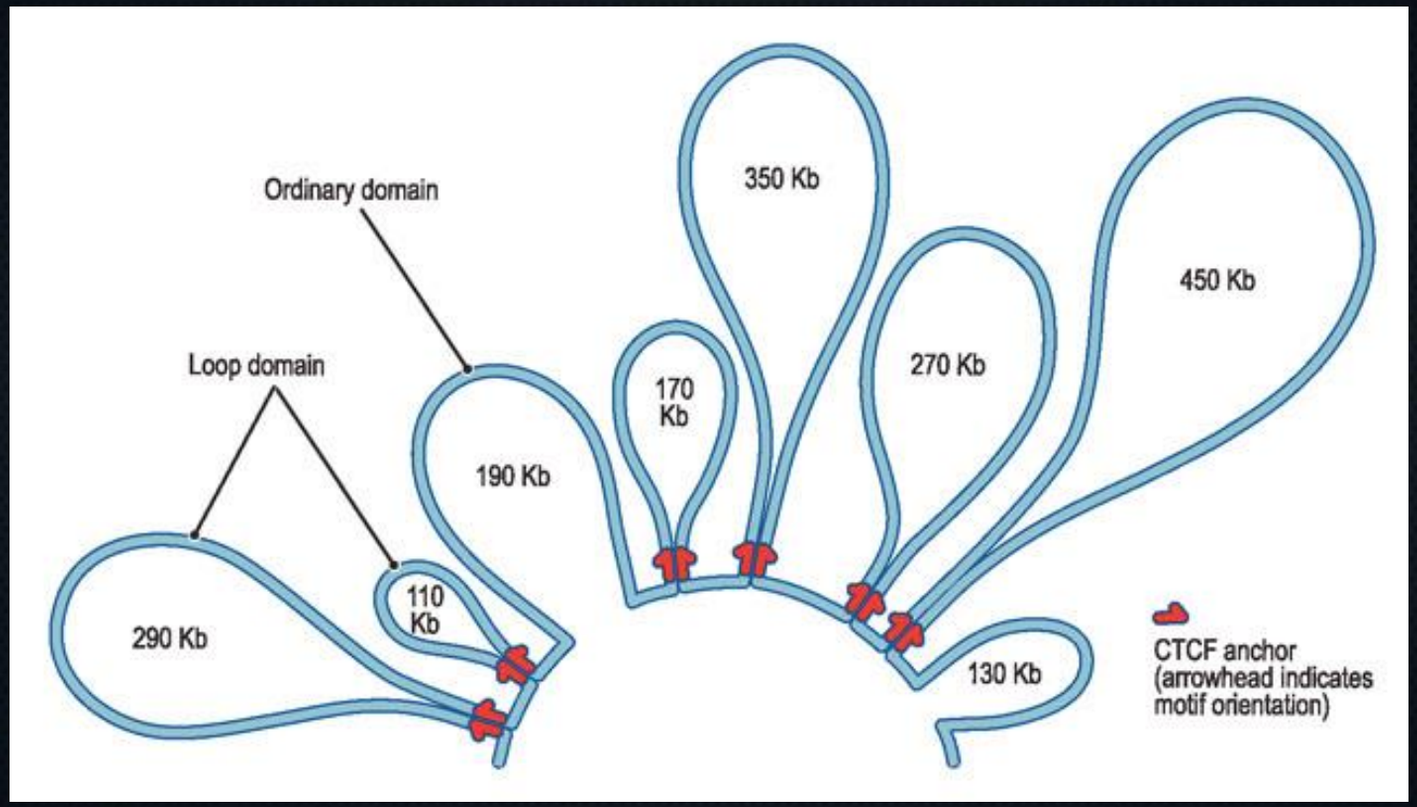
The area under the ROC curve (AUC) measures performance over the whole ROC curve, considering every possible decision threshold, which is too general and includes thresholds that would never be used. Accuracy, F1 score, sensitivity and specificity measure performance at a single decision threshold (point) on an ROC curve, which is too specific and ignores information. **Deep ROC analysis (paper) (presentation) (code) [1]** permits in-depth analysis of classifier performance in groups of predicted risk or probability that span the ROC curve. Previous attempts to represent AUC by partial AUC, the standardized partial AUC or the two-way AUC were flawed.

http://www.deeproc.org/

Summary

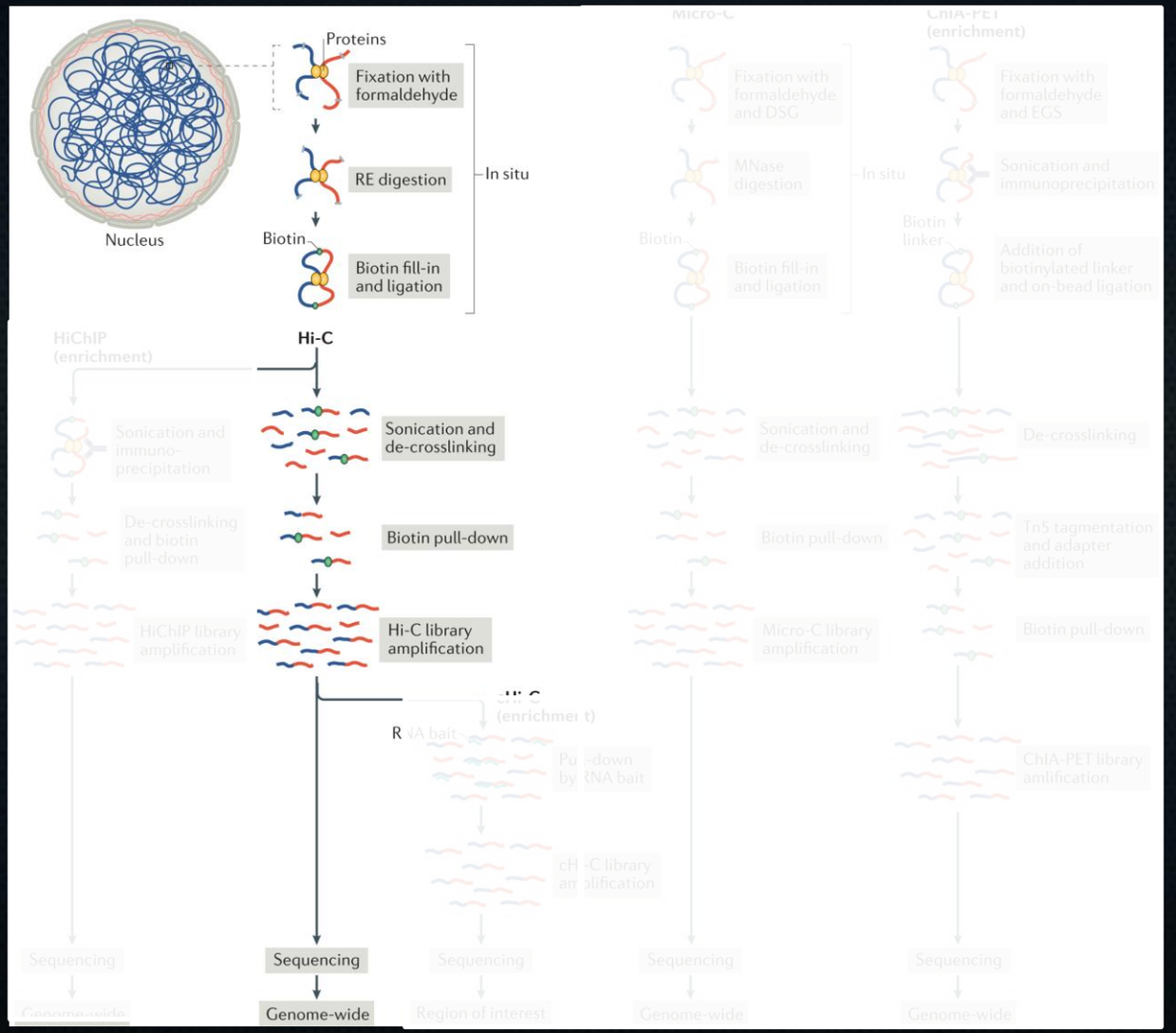
- Multiclass logistic regression: train *n_classes* binary classifiers
- Can tailor threshold depending on if you want to favour sensitivity or specificity
- ROC curve gives performance for all possible decision thresholds (also unreasonable ones)
- ROC curve not fit for imbalanced data (many more negatives) where you care about positive class: use PR curve instead

Metrics in reality



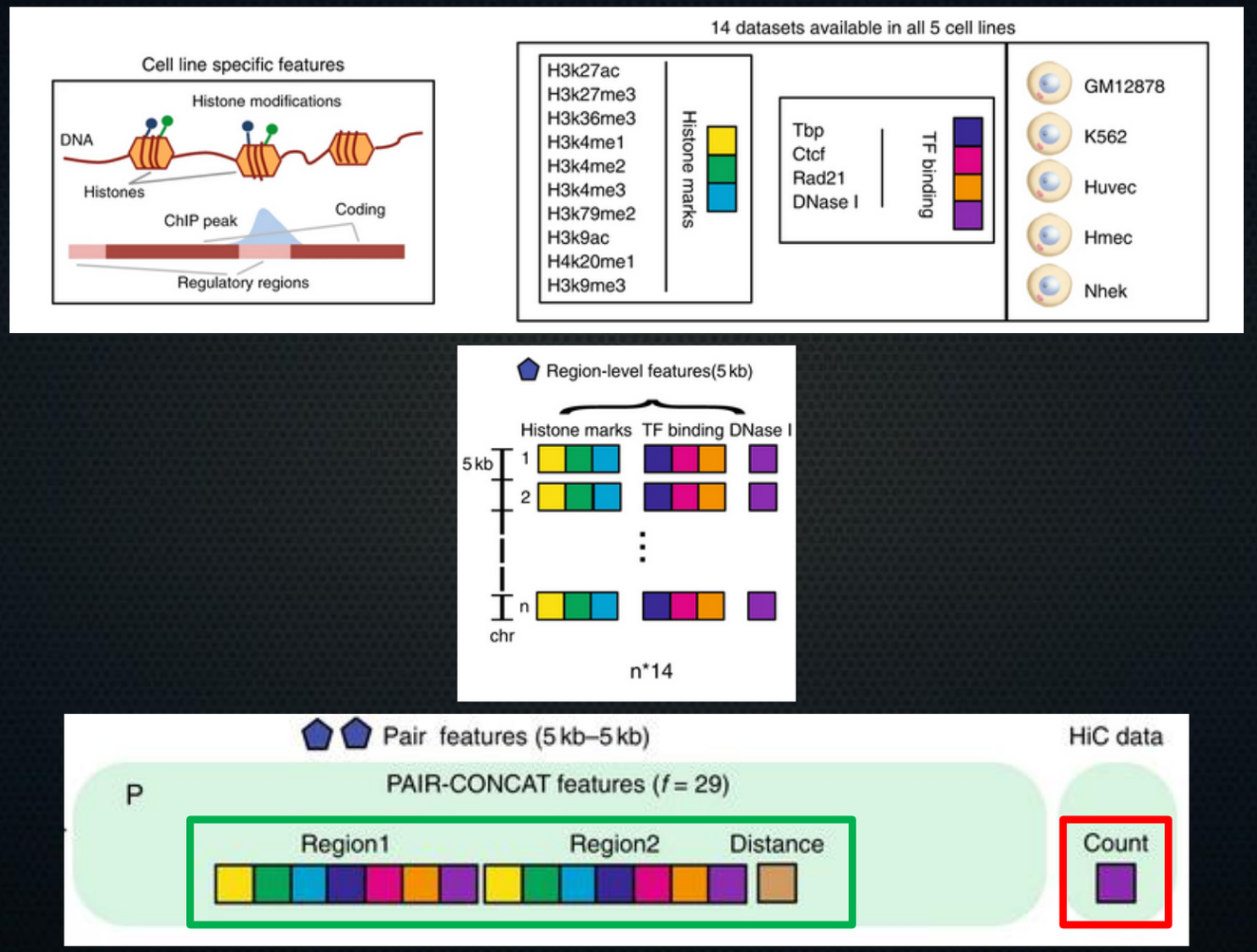
https://jonlieffmd.com/wp-content/uploads/2015/04/DNA-loops-Harvard-.jpg

Metrics in reality



https://www.nature.com/articles/s41580-021-00362-w

Metrics in reality



https://www.nature.com/articles/s41467-019-13423-8

Metrics in reality

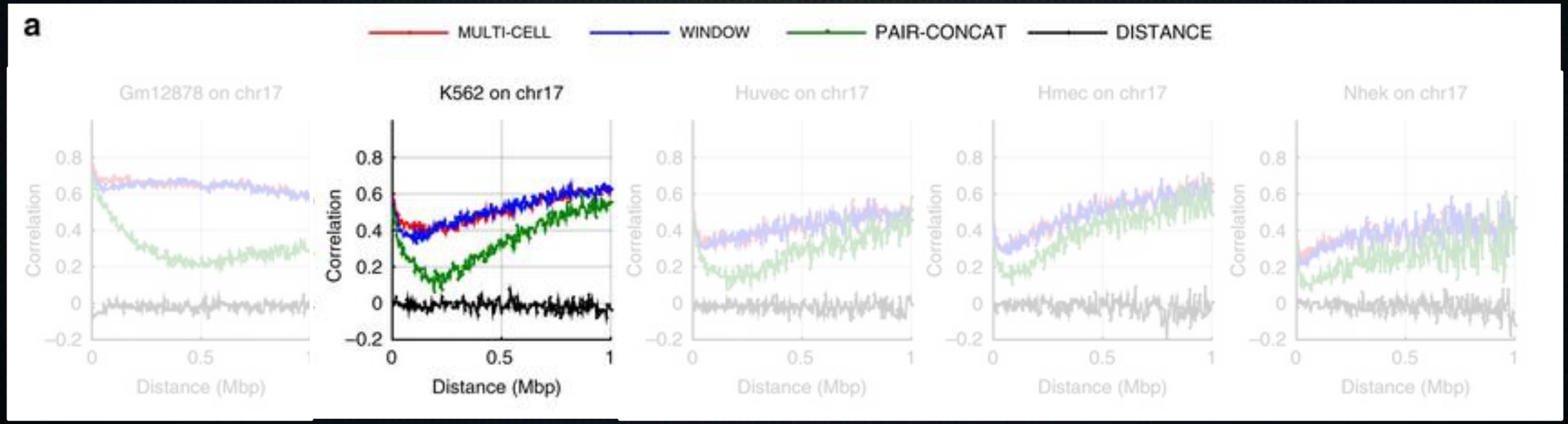
Question:
Can we predict how often two bins interact?

What would be a good metric?

What do you need to correct for/
take into account?

Metrics in reality

Question:
What did they end up using here?



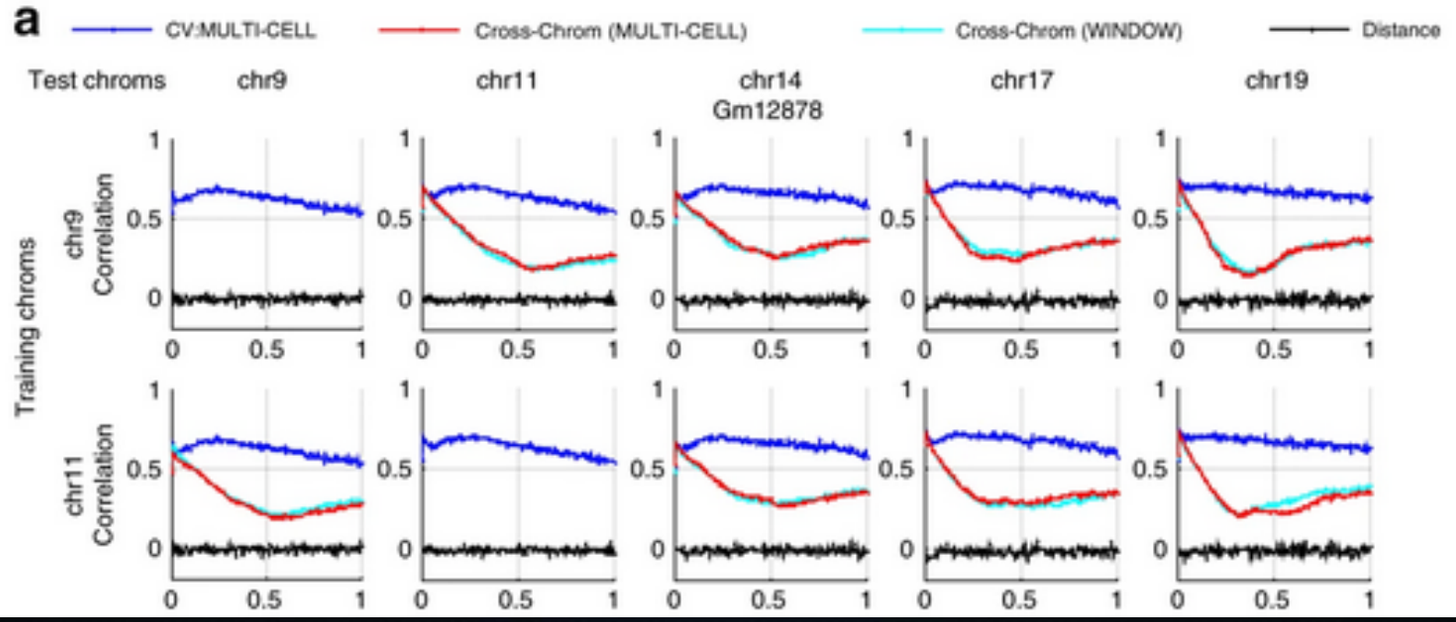
https://www.nature.com/articles/s41467-019-13423-8

Metrics in reality

Question:
What are they testing now?

Fig. 3

From: In silico prediction of high-resolution Hi-C interaction matrices



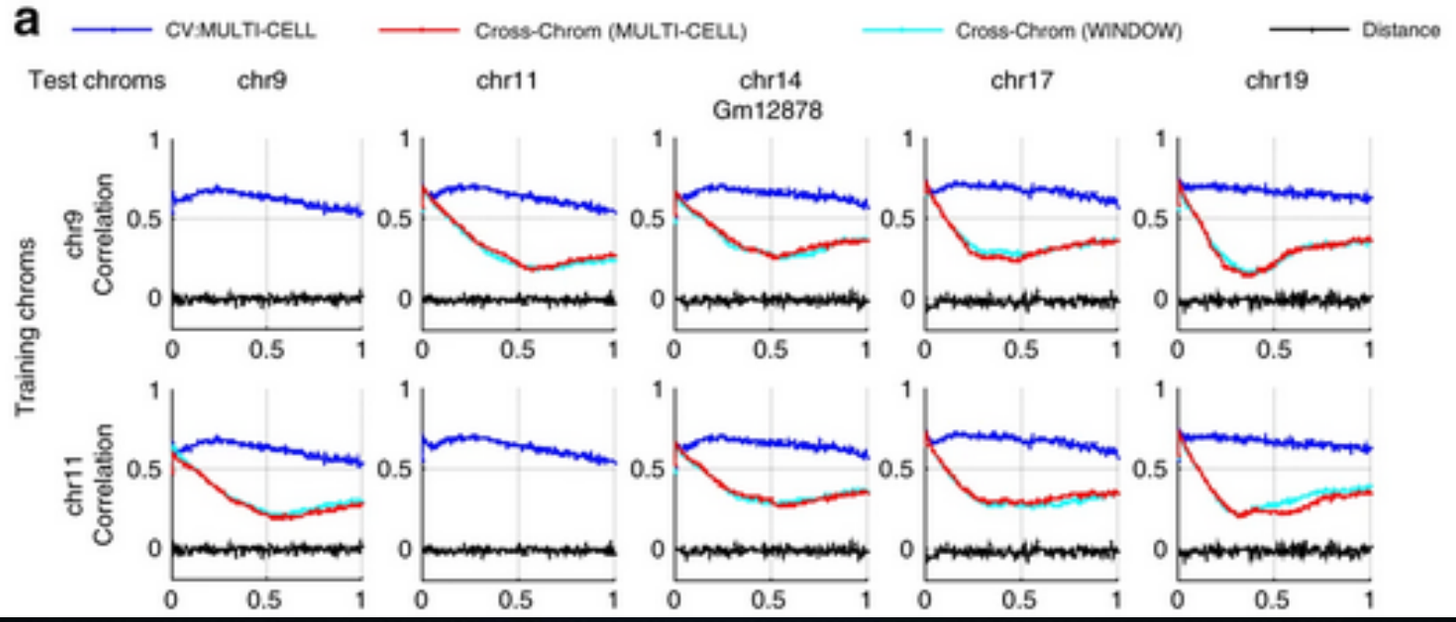
https://www.nature.com/articles/s41467-019-13423-8

Metrics in reality

Question:
What are they testing now?

Fig. 3

From: In silico prediction of high-resolution Hi-C interaction matrices



https://www.nature.com/articles/s41467-019-13423-8

Metrics in reality

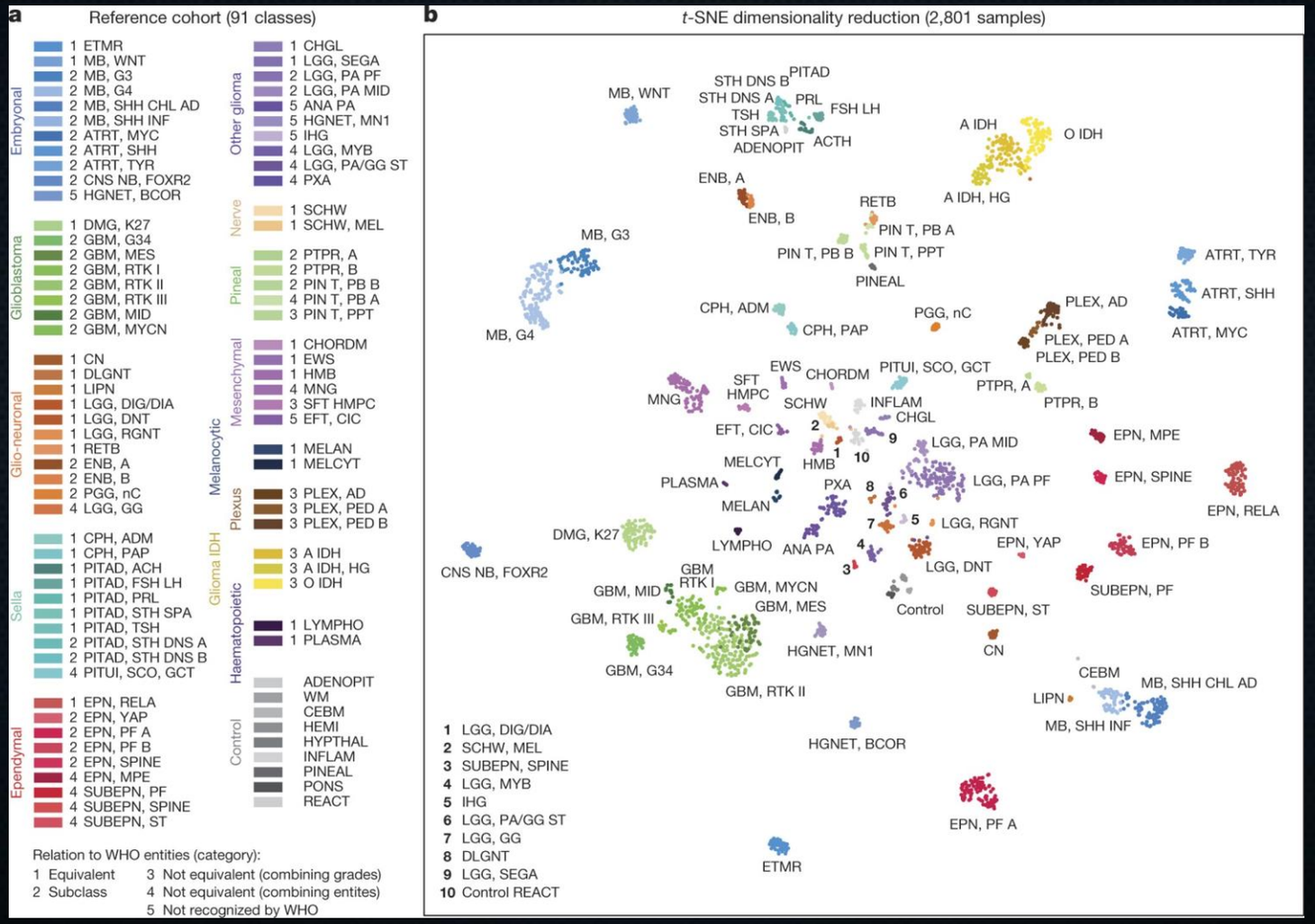
Cross-validate over **biological entities** and **distance-stratified**:

- Same chromosome
- Cross-chromosome

Every metric answers a different question!
Or: investigates a *different level* of generalisation

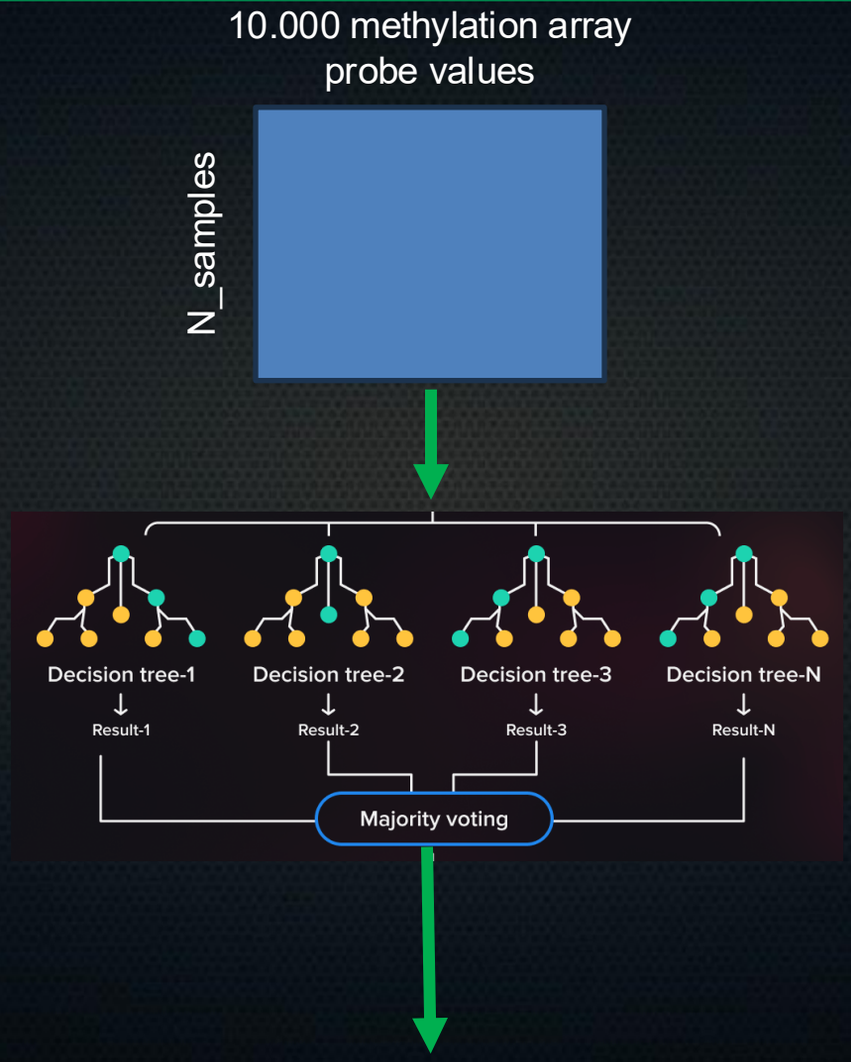
https://www.nature.com/articles/s41467-019-13423-8

Metrics in reality



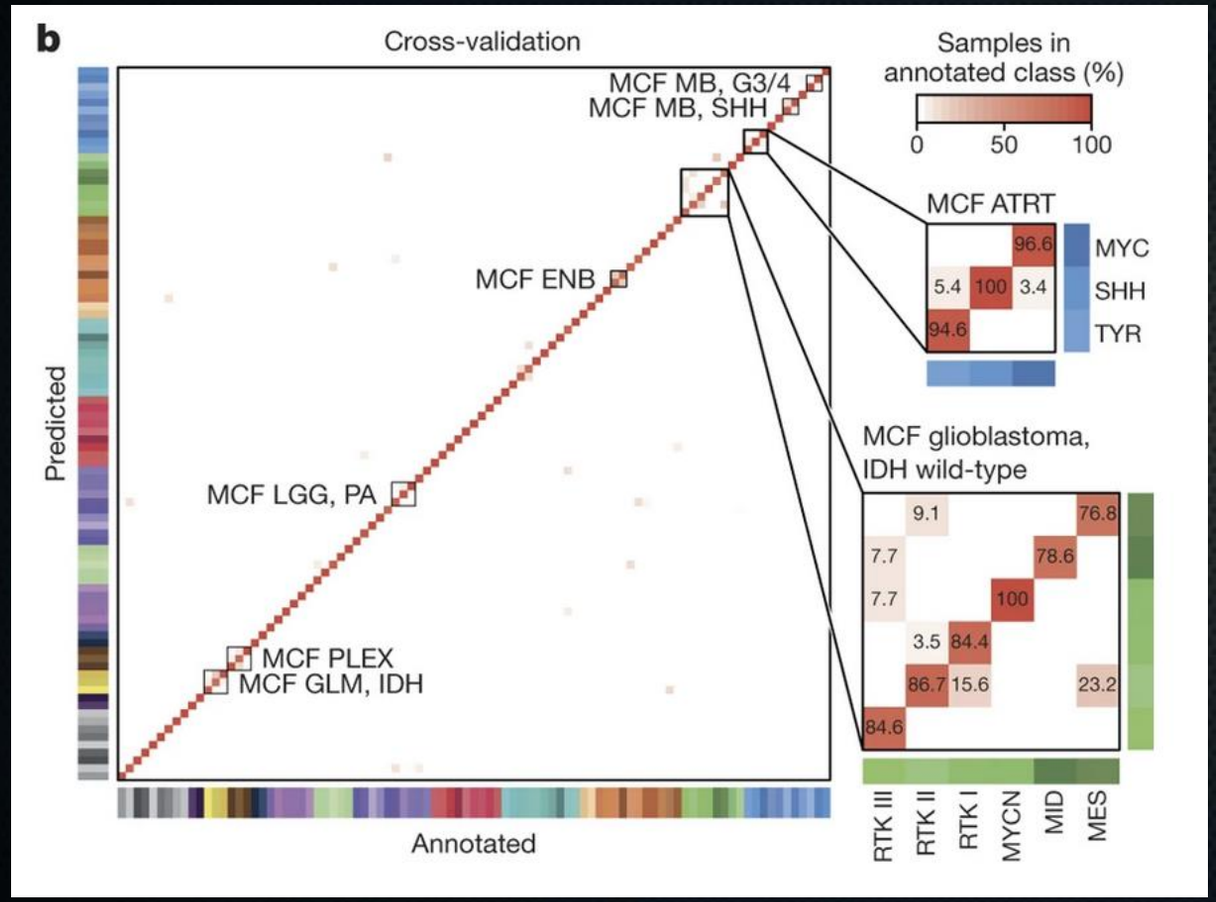
https://www.nature.com/articles/nature26000

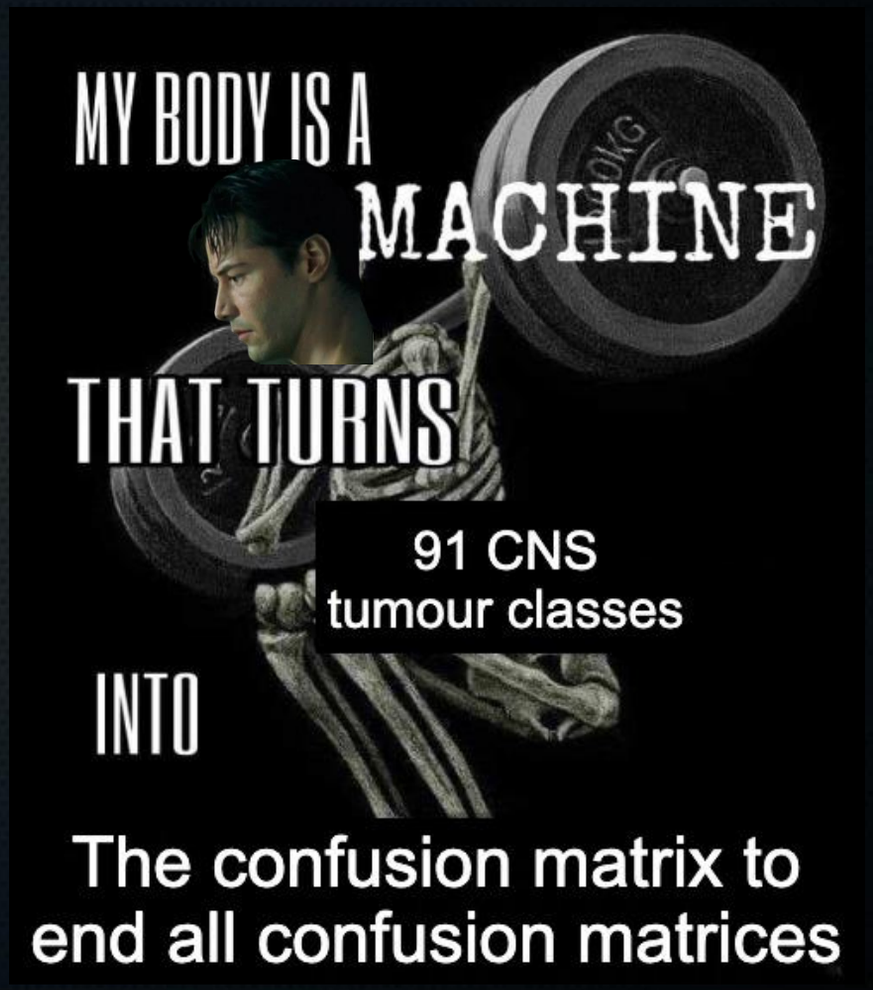
Metrics in reality



Prediction: 1 of 91 possible tumour types

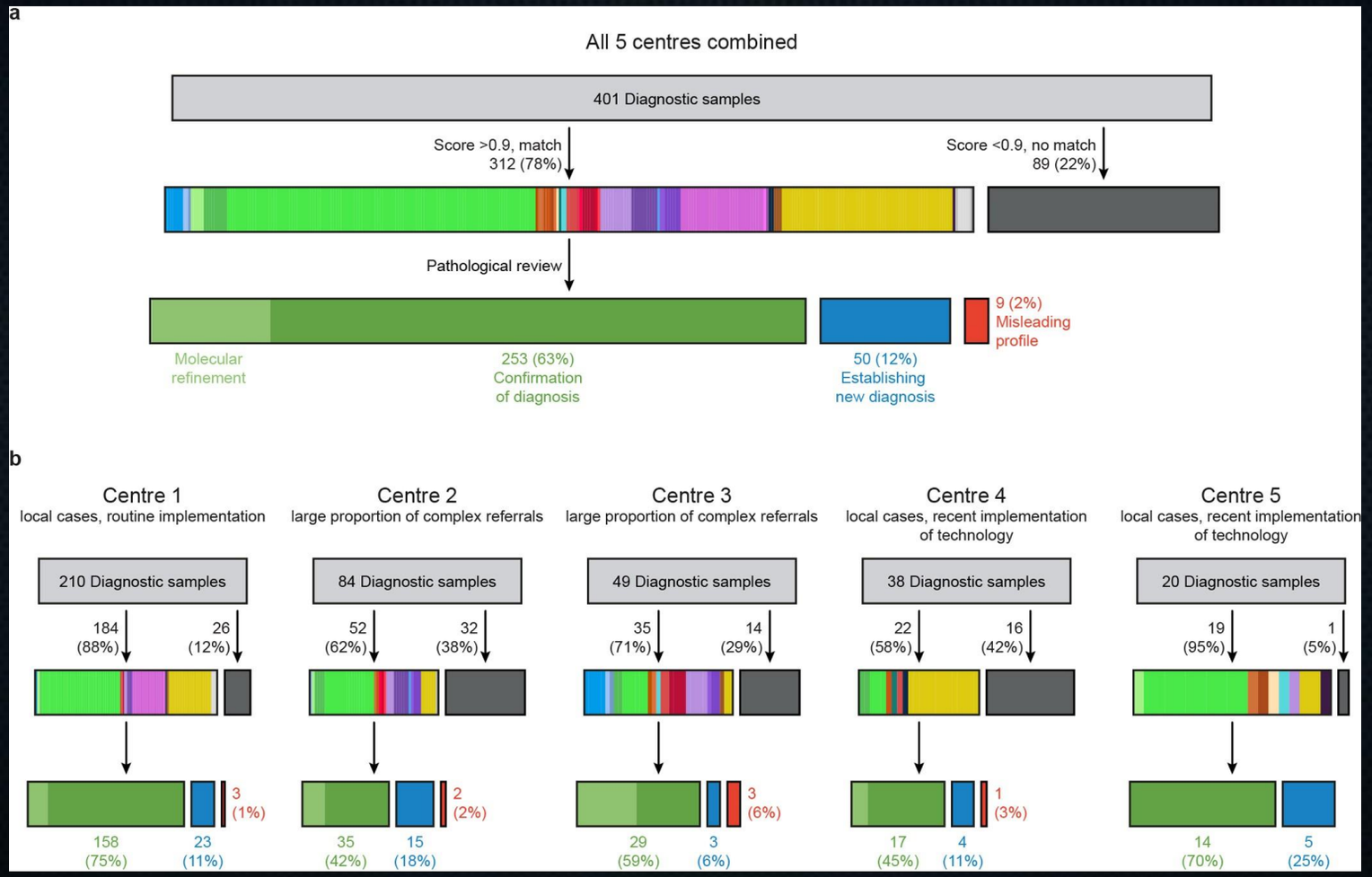
Metrics in reality





https://www.nature.com/articles/nature26000

Metrics in reality



https://www.nature.com/articles/nature26000

Metrics in reality

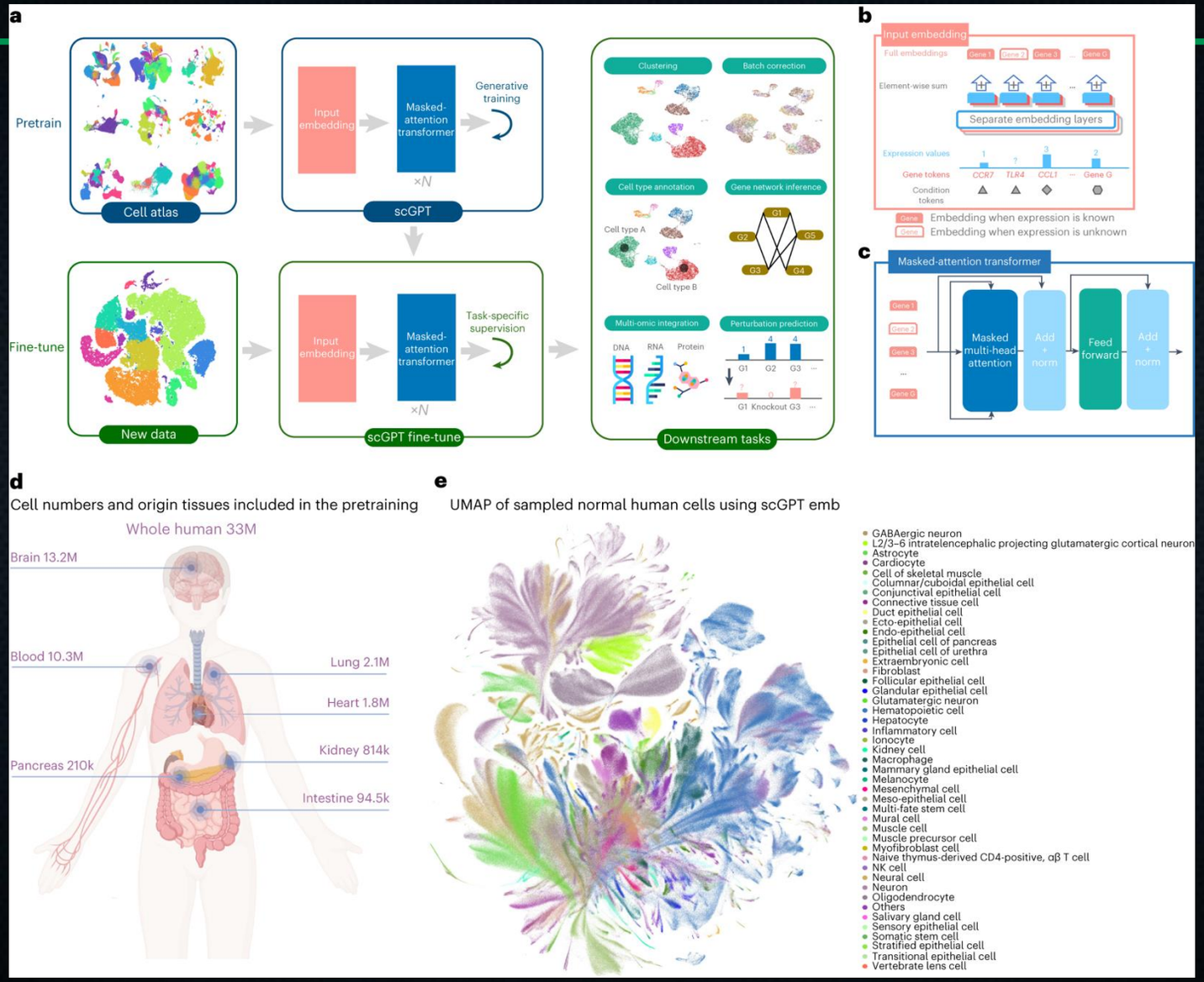
Usual cross-validation and confusion matrix

- Same chromosome
- Cross-chromosome

Every metric answers a different question!
Or: investigates a *different level* of generalisation

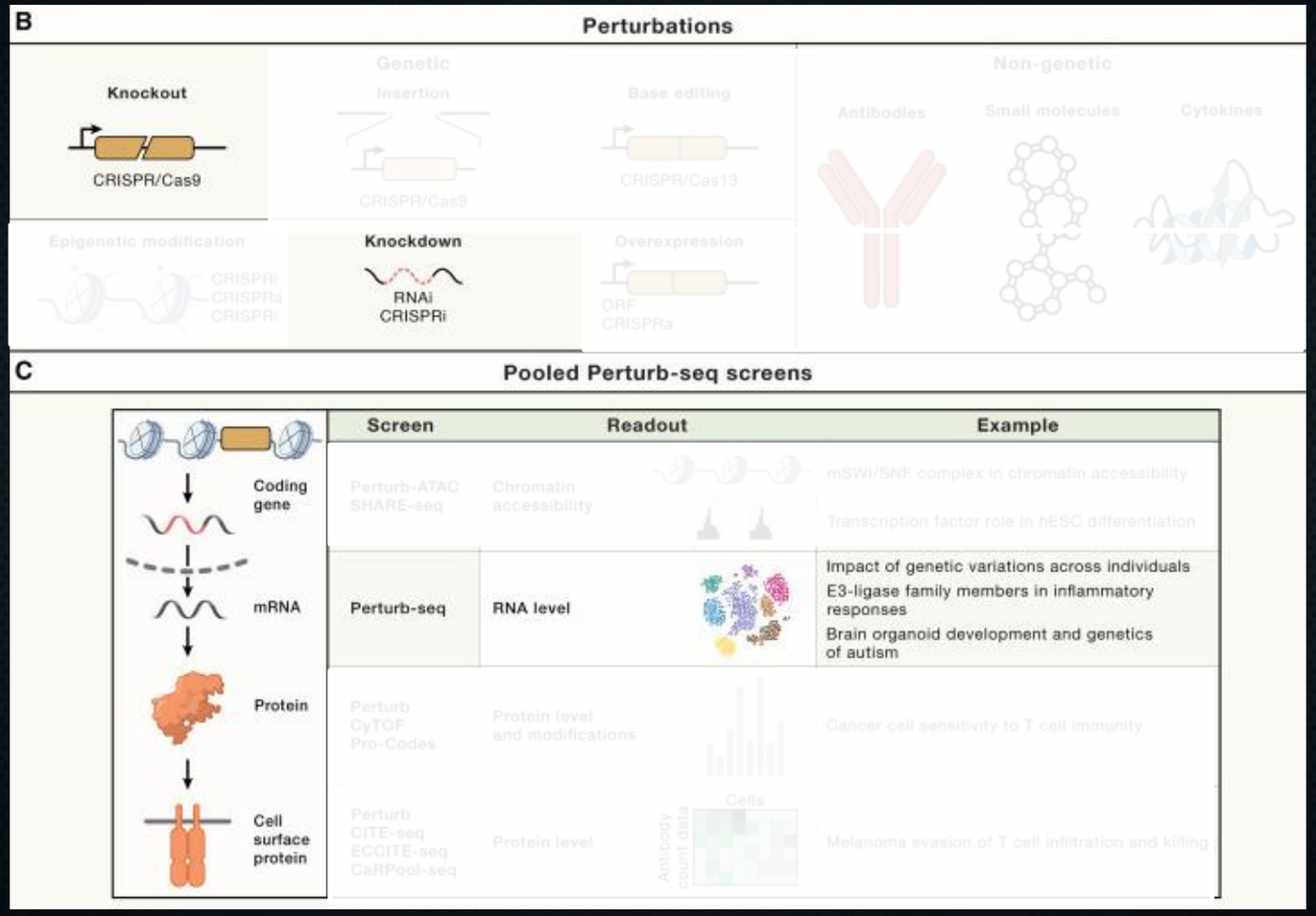
https://www.nature.com/articles/nature26000

Metrics in reality : scGPT blunder



https://www.nature.com/articles/s41592-024-02201-0

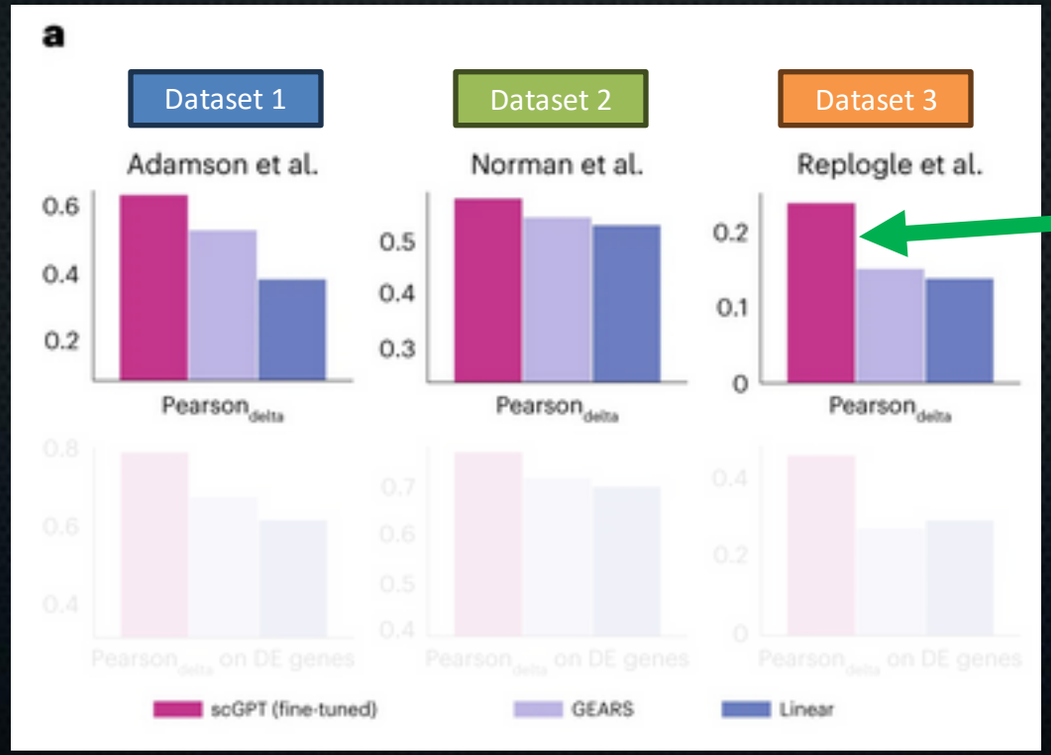
Metrics in reality : scGPT blunder

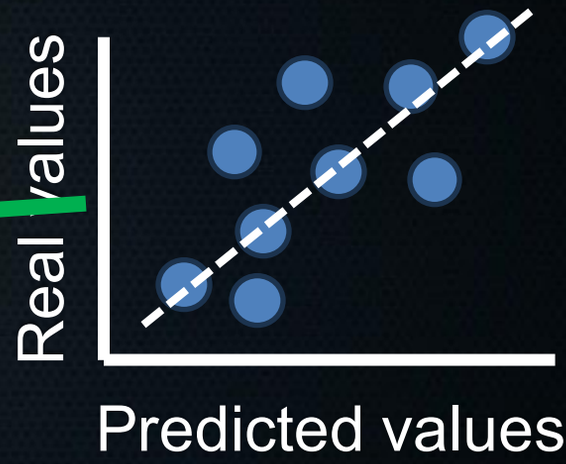


https://www.cell.com/cell/fulltext/S0092-8674(24)00829-8

Metrics in reality : scGPT blunder

Pearson correlation of predicted and true expression

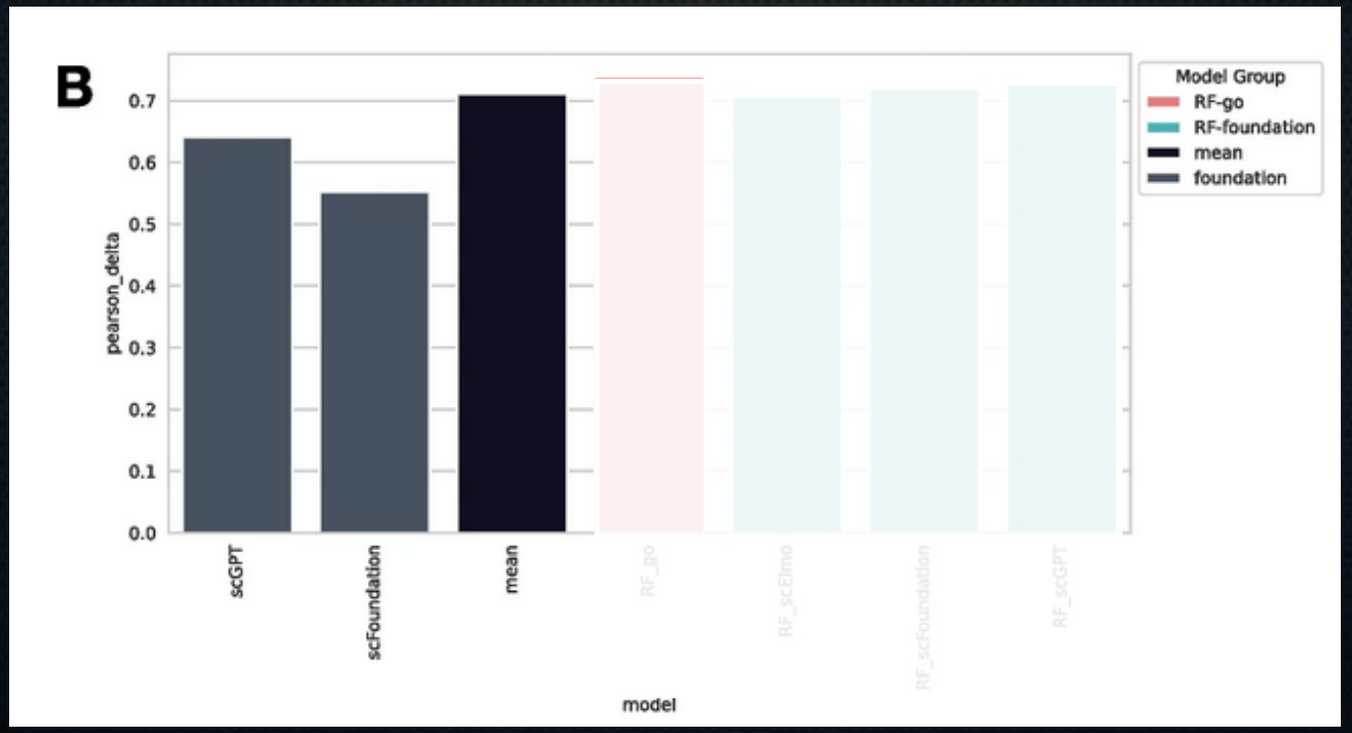




https://www.nature.com/articles/s41592-024-02201-0

Metrics in reality : scGPT blunder

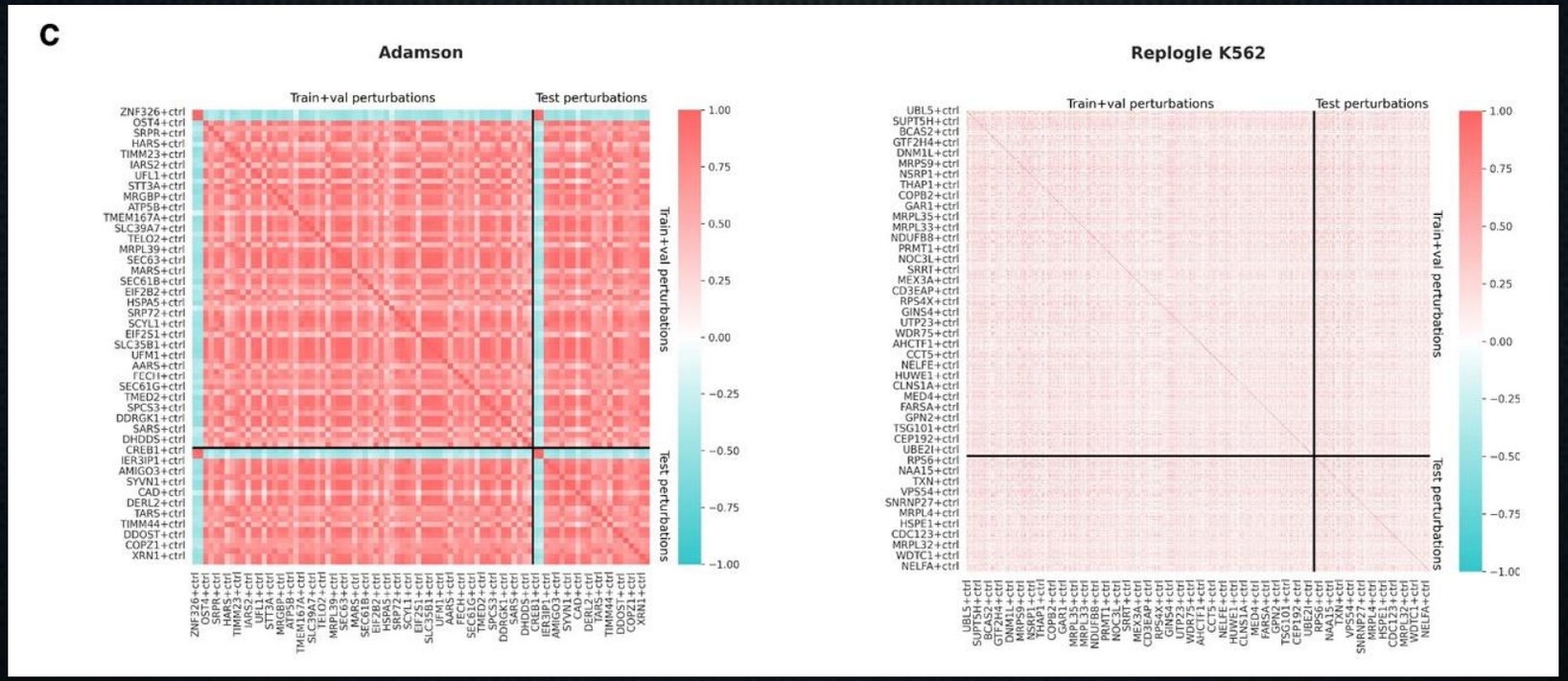
Question:
What are they testing now?



https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-025-11600-2

Metrics in reality : scGPT blunder

Question:
What are they testing now?



https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-025-11600-2

Metrics in reality

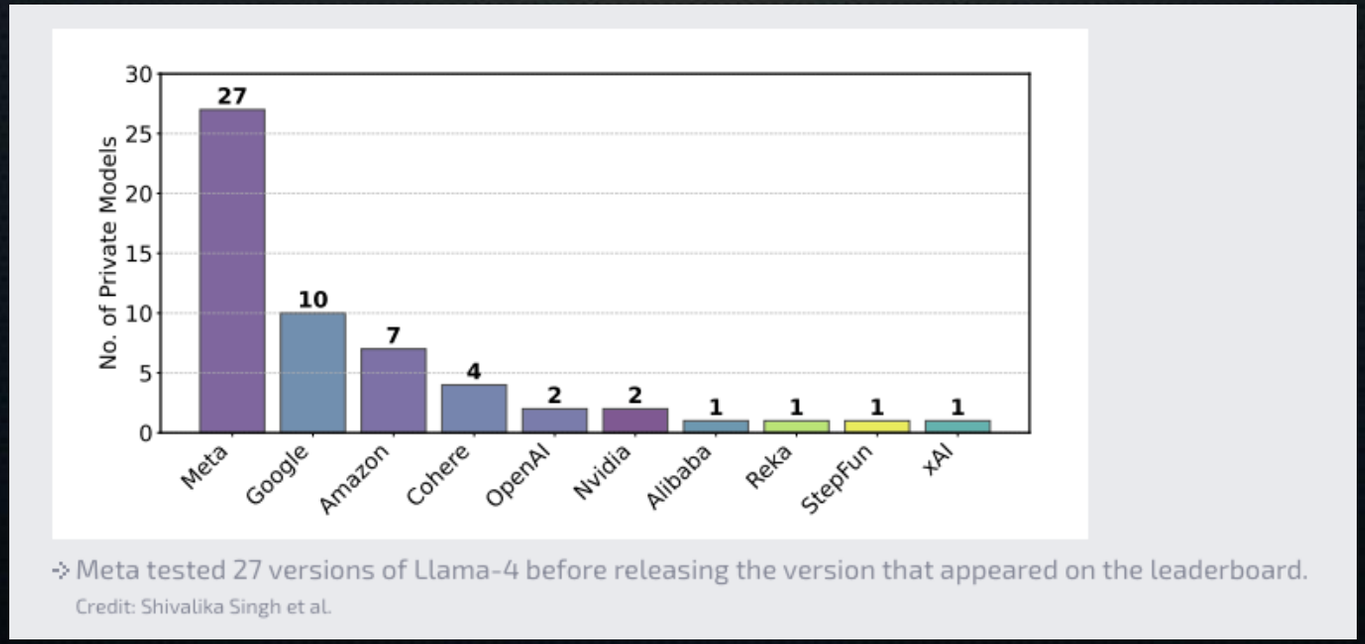
Almost wilful ignorance of the most stupid baseline possible.

- Always test against *many* baselines
- **Know your data: garbage in, garbage out!**

"When a measure becomes a target, it ceases to be a good measure"

Metrics in reality

Goodhart’s law:
"When a measure becomes a target, it ceases to be a good measure"



-> Meta tested 27 versions of Llama-4 before releasing the version that appeared on the leaderboard.

Credit: Shivalika Singh et al.

https://arstechnica.com/ai/2025/05/researchers-claim-lm-arenas-ai-leaderboard-is-biased-against-open-models/

Regularisation

- Change the cost function to apply a cost for complexity

Regularisation

- Change the cost function to apply a cost for complexity

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} - y^{(i)} \cdot \log\left(h_\theta(x^{(i)})\right) - (1 - y^{(i)}) \cdot \log(1 - h_\theta(x^{(i)}))$$

Regularisation

- Change the cost function to apply a cost for complexity

$$J(\theta_0 \ldots \theta_n) = \frac{1}{m} \sum_{i=1}^{m} -y^{(i)} \cdot \log\left(h_\theta(x^{(i)})\right) - (1 - y^{(i)}) \cdot \log\left(1 - h_\theta(x^{(i)})\right)$$

$$J(\theta_0 \ldots \theta_n) = \frac{1}{m} \sum_{i=1}^{m} \left(-y^{(i)} \cdot \log\left(h_\theta(x^{(i)})\right) - (1 - y^{(i)}) \cdot \log\left(1 - h_\theta(x^{(i)})\right)\right) + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

- Make J a function of both the error of predictions given some parameters *and* the magnitude of those parameters themselves

Regularisation

- Change the cost function to apply a cost for complexity

$$J(\theta_0 \ldots \theta_n) = \frac{1}{m} \sum_{i=1}^{m} -y^{(i)} \cdot \log\left(h_\theta(x^{(i)})\right) - (1 - y^{(i)}) \cdot \log(1 - h_\theta(x^{(i)}))$$

$$J(\theta_0 \ldots \theta_n) = \frac{1}{m} \sum_{i=1}^{m} \left(-y^{(i)} \cdot \log\left(h_\theta(x^{(i)})\right) - (1 - y^{(i)}) \cdot \log(1 - h_\theta(x^{(i)}))\right) + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

- Make J a function of both the error of predictions given some parameters *and* the magnitude of those parameters themselves
- By convention: don‘t shrink bias/intercept term

Easiest to visualise how it works in linear regression

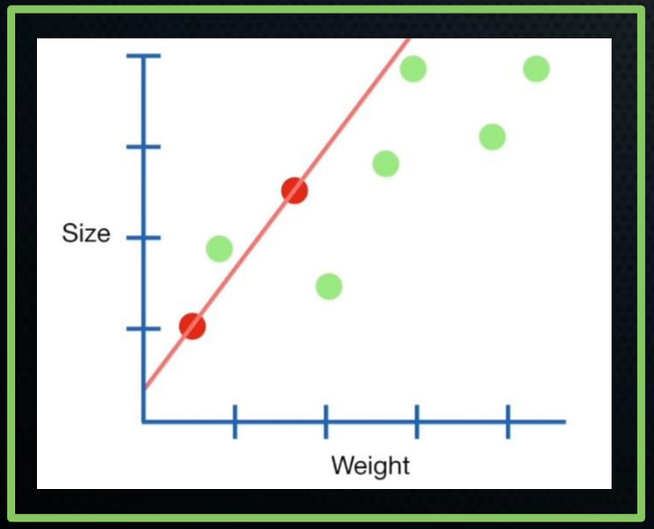
$$J(\theta_0, \ldots, \theta_n) = \frac{1}{2m}\sum_{i=1}^{m} (h_\theta(x) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} (\theta_j)^2 \qquad h_\theta(x) = \theta_0 + \theta_1 x$$

Regularisation: ridge regression

- Change the cost function to apply a cost for complexity

$$J(\theta_0, \ldots, \theta_n) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}(\theta_j)^2 \qquad h_\theta(x) = \theta_0 + \theta_1 x$$

- Add some **bias** (constrain hypothesis to a set with small parameter values) but reduces **variance**:



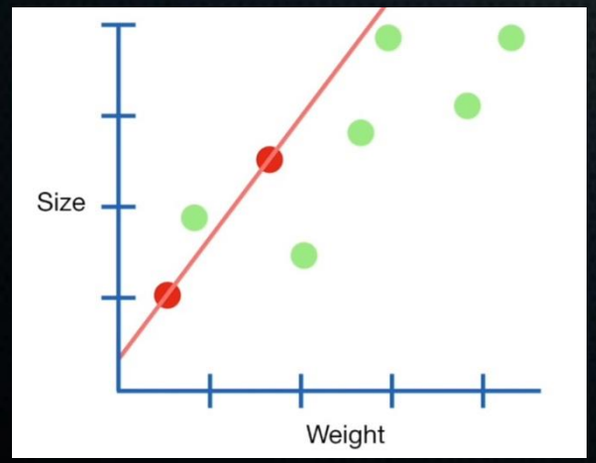
Source: StatQuest. “Regularization Part 1: Ridge (L2) Regression” YouTube, Joshua Starmer, 24 Sep. 2018,https://www.youtube.com/watch?v=Q81RR3yKn30

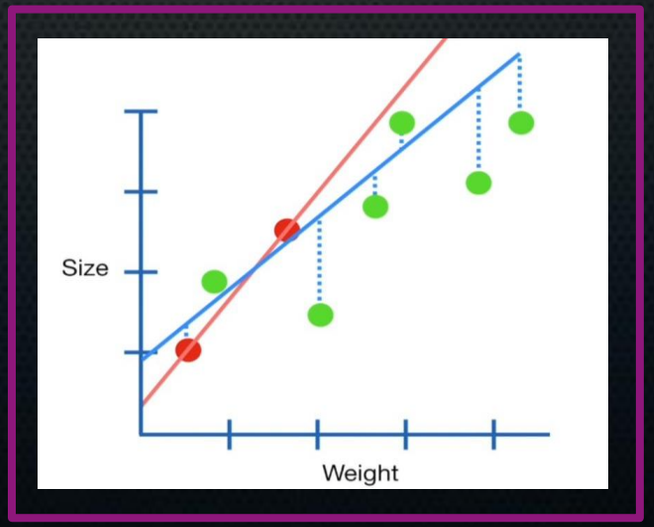
Regularisation: ridge regression

- Change the cost function to apply a cost for complexity

$$J(\theta_0, \ldots, \theta_n) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}(\theta_j)^2 \qquad h_\theta(x) = \theta_0 + \theta_1 x$$

- Add some **bias** (constrain hypothesis to a set with small parameter values) but reduces **variance**:

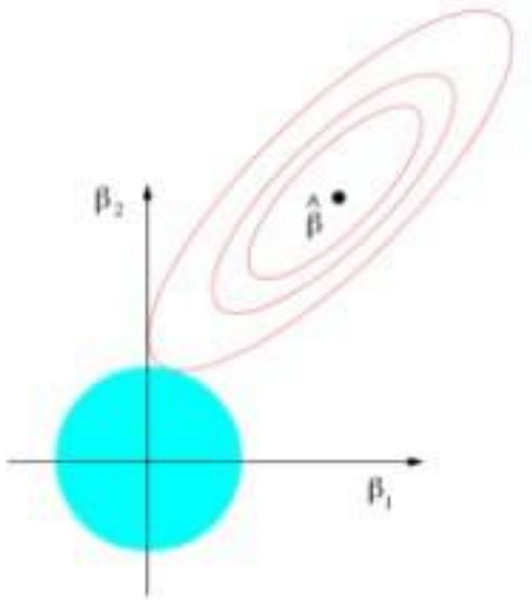




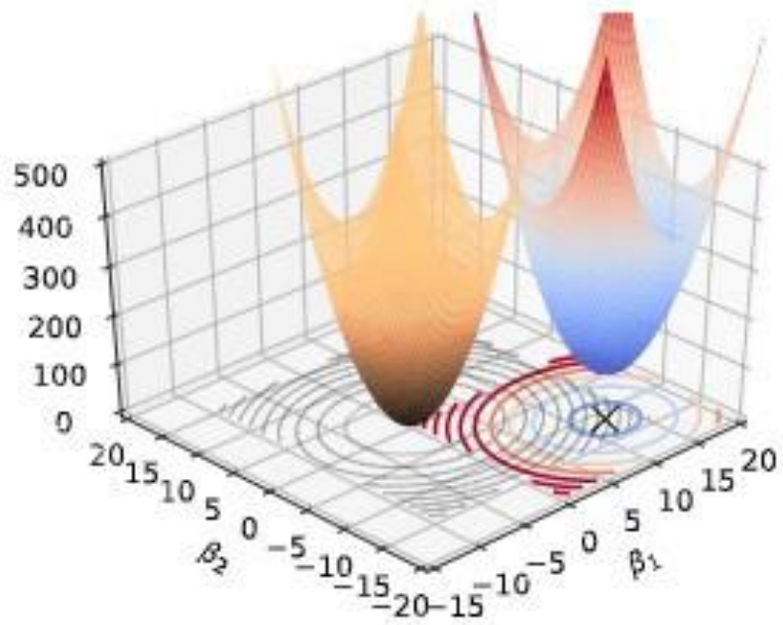
Constrained how much the line may increase with Weight (biased) → generalises better to test set

Source: StatQuest. "Regularization Part 1: Ridge (L2) Regression" YouTube, Joshua Starmer, 24 Sep. 2018,https://www.youtube.com/watch?v=Q81RR3yKn30

Regularisation visually



Hard constraint illustration from ESL page 71.



Soft constraint with non-regularized loss function (blue-red) term and penalty term (orange).

https://explained.ai/regularization/

Regularisation taken together

- Changes your cost function to add penalty for too large parameters. Slightly increases bias, decreases variance.
- Adds another hyperparameter, **λ**
- Can be used for linear regression, logistic regression, neural networks, etc.

$$J(\theta_0 \ldots \theta_n) = \frac{1}{m}\sum_{i=1}^{m}\left(-y^{(i)}\cdot\log\left(h_\theta(x^{(i)})\right) - \left(1-y^{(i)}\right)\cdot\log\left(1-h_\theta(x^{(i)})\right)\right) + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

- What is the cost I calculate on the validation set?

Regularisation taken together

- Changes your cost function to add penalty for too large parameters. Slightly increases bias, decreases variance.
- Adds another hyperparameter, **λ**
- Can be used for linear regression, logistic regression, neural networks, etc.

$$J(\theta_0 \ldots \theta_n) = \frac{1}{m}\sum_{i=1}^{m}(-y^{(i)} \cdot \log(h_\theta(x^{(i)})) - (1-y^{(i)}) \cdot \log(1-h_\theta(x^{(i)}))) + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

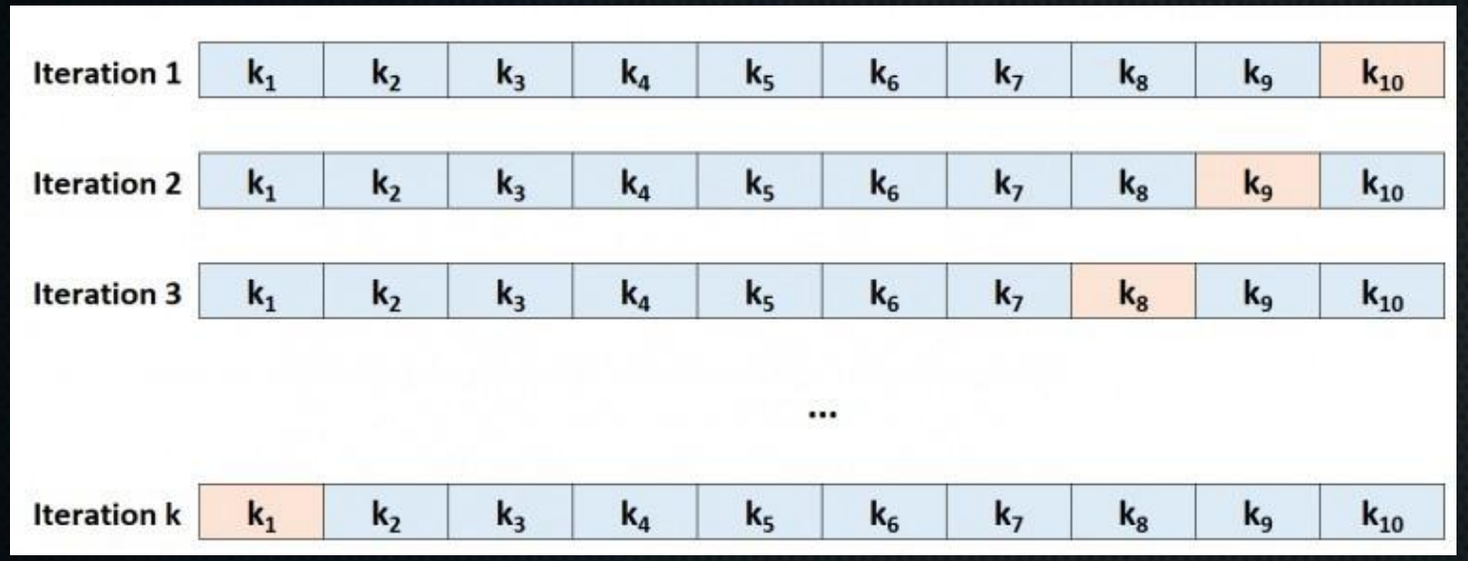
- **Important note: don't use the regularisation term when calculating on validation set!**

What about those hyperparameters?

- We've now encountered λ for regularisation and α, the learning rate.
- How do we pick these hyperparameters?

What about those hyperparameters?

- Back to cross-validation:

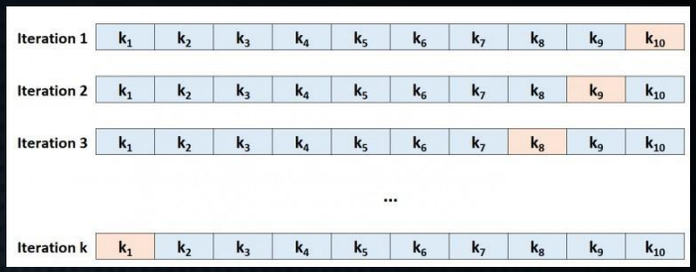


Source:https://www.statology.org/k-fold-cross-validation/

- Here, we train our model on 9 folds, and test it on a hold-out set of data to get a feel for the generalisation error (or the *real* performance on unseen data) by averaging the 10 values we get.

What about those hyperparameters?

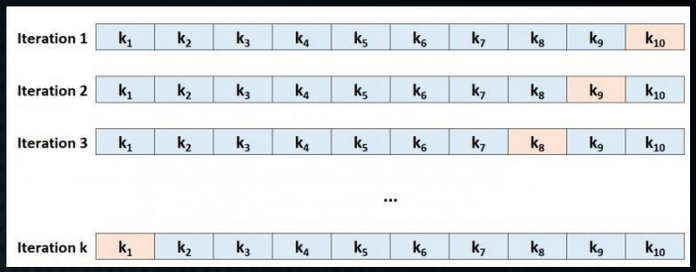
- Naive idea for hyperparameters:
  - Say we want to pick alpha from [0.001, 0.01, 0.1, 1, 10]
  - Take each alpha, train your model with those using cross-validation, and pick whichever set performs best on average out of the 5 * 10 models you made.



Source:https://www.statology.org/k-fold-cross-validation/

What about those hyperparameters?

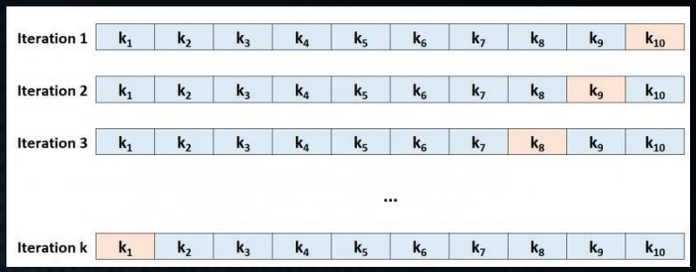
- Naive idea for hyperparameters:
  - Say we want to pick alpha from [0.001, 0.01, 0.1, 1, 10]
  - Take each alpha, train your model with those using cross-validation, and pick whichever set performs best on average out of the 5 * 10 models you made.



Source:https://www.statology.org/k-fold-cross-validation/

What about those hyperparameters?

- Naive idea for hyperparameters:
  - Say we want to pick alpha from [0.001, 0.01, 0.1, 1, 10]
  - Take each alpha, train your model with those using cross-validation, and pick whichever set performs best on average out of the 5 * 10 models you made.
- But there's something wrong here:
  - If you do this, you select hyperparameters that give the best score on the validation set → so it's fitted to your data in a way.
  - You want to prevent this leak of information.

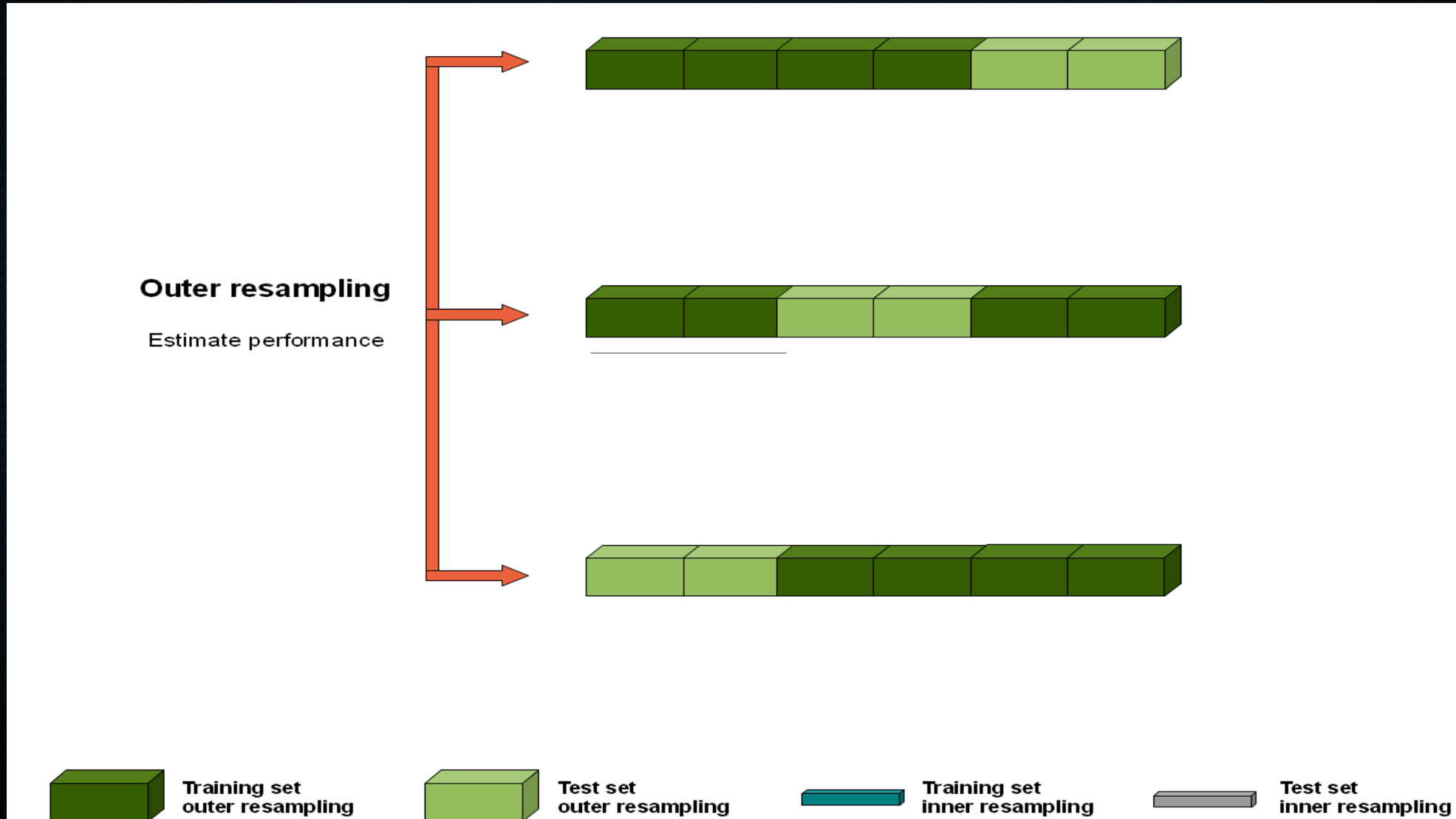


Source:https://www.statology.org/k-fold-cross-validation/

Nested cross-validation

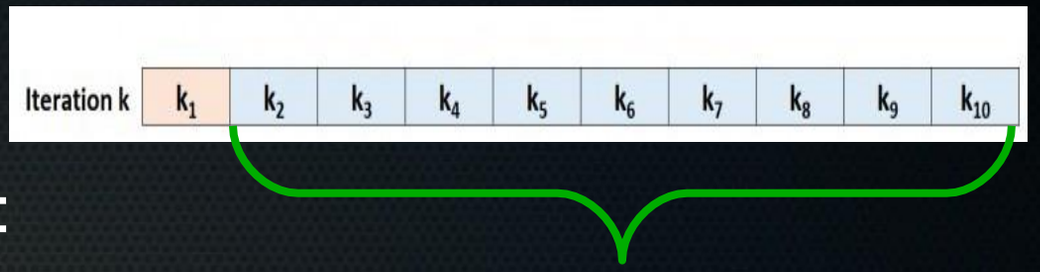
- What we want: pick best hyperparameters automatically *and* independently from validation performance.

Nested cross-validation



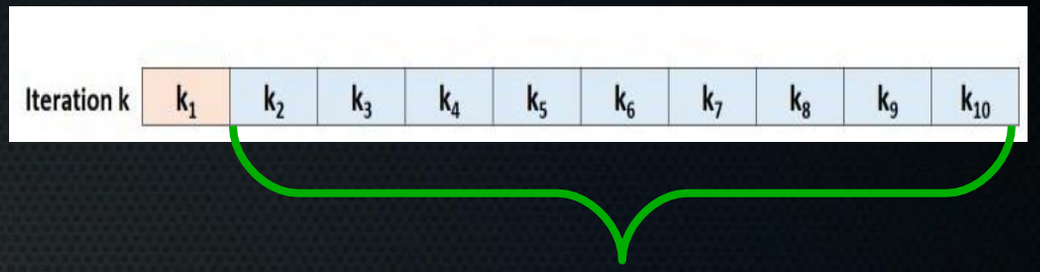
Nested cross-validation

- So: α = [0.001, 0.01, 0.1, 1, 10]
  - Make outer folds, say from 1 to *k* (10)
  - Within each of the 10 training data sets:
    - Make inner folds, say from 1 to *l* (also 10 here)



Nested cross-validation

- So: $\alpha = [0.001, 0.01, 0.1, 1, 10]$
  - Make outer folds, say from 1 to *k* (10)
  - Within each of the 10 training data sets:
    - Make inner folds, say from 1 to *l* (also 10 here)
    - For each of the 10 training data sets:
      - For each of our alpha values:
        - Train a model with those alpha values
        - Test it on the validation set
    - End up with: average performance of 5 alpha values over 10 folds.
    - Pick alpha with best average perfomance (and record it)
  - For the outer fold, train with that best alpha and predict on held out data



>1 hyperparameters

- Just train models with all possible (or a subset of combinations of them)
- Note: nested cross-validation takes *a lot* of computing power. For the example on the previous slide:
  10 upper folds *
  10 lower folds *
  5 alpha values
  = *500 models!*

Summary

- Multi-class classification: simply train *n* independent binary logistical regressors for your *n* classes, run them all on the data, pick for each sample the class with the highest probability over the regressors
- Performance metrics: Accuracy, sensitivity and specificity, ROC curve and ROC AUC (or PRC AUC for imbalanced data)
- Regularisation: add a cost to making parameters too large (i.e. fitting them too precisely to the data). Forces the model to only increase those parameters that really improve the fit (less fine-tuning exactly to the training data)!
- Nested cross-val for finding hyperparameters.

BREAK FOR PRACTICAL