Lones, M. A. (2024). Avoiding common machine learning pitfalls. *Patterns*, *5*(10), 101046.

Omics Journal Club

Presented by Margaret Ho

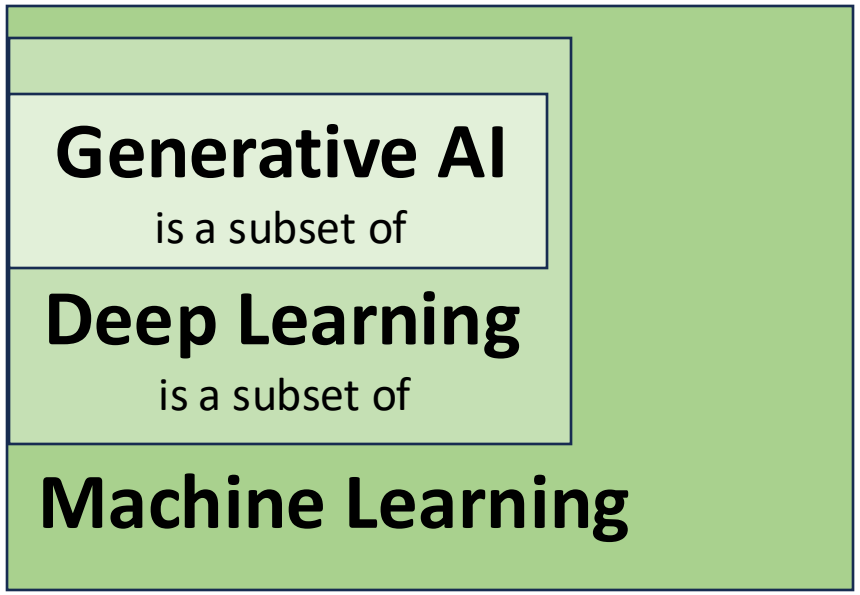
Dec 2 2025

Motivation

- Many traditional machine learning techniques are already used in genomics, some are relatively established
- We are seeing more complex “deep learning” methods using neural networks or even “generative” methods published
  - So many papers – can be a bit overwhelming
    - Sequence models moving from CNNs to pre-trained “foundation” models using transformer architecture
    - Dimension reduction, feature selection and data integration with variational autoencoders
    - Graph neural networks for modeling genetic interactions
    - Representation learning for automated feature extraction
    - Diffusion modeling for denoising

I have been trying to find papers that can provide **best practices** and **overall guidance** rather just than papers where they just try to impress reader with their latest model

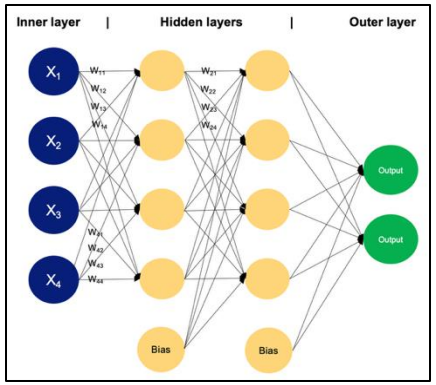
- Interpretability is key for biology



Related categories of computer science that are sometimes easily confused

Machine Learning vs Deep Learning

	Traditional Machine Learning	Deep Learning (subset of ML)
Feature Engineering	Uses **manual feature extraction** e.g., selecting relevant genes, motifs, or summary statistics	Learns **hierarchical features automatically** from raw data (e.g., DNA sequences, images)
Model Complexity	Models are usually **shallow** with few layers, fewer parameters	**Neural networks** with many layers, blocks and millions of parameters
Data	Works well with **small to moderate** datasets	Needs **large datasets** to learn effectively
Compute	Runs efficiently on CPUs	Needs **GPUs/TPUs** due to high computational cost
Interpretability	Often more interpretable (decision trees, linear models)	Significantly more opaque/black box, though some efforts being made to improve
Examples	Linear regression, random forests, SVMs, k-nn	CNNs, RNNs, Transformers, Autoencoders, Diffusion Models



Generative AI – subset of Deep Learning – is tasked with generating new content that resembles and is based on training data

Full disclosure, this table was generated by ChatGPT though I have tweaked it slightly

Patterns

CellPress
OPEN ACCESS

Tutorial

Avoiding common machine learning pitfalls

Michael A. Lones[1,*]
[1]School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK
*Correspondence: m.lones@hw.ac.uk
https://doi.org/10.1016/j.patter.2024.101046

Written as a guide of do's and don'ts in the 5 stages of ML:

- Before you build models
- How to reliably build models
- How to robustly evaluate models
- How to compare models fairly
- How to report your results

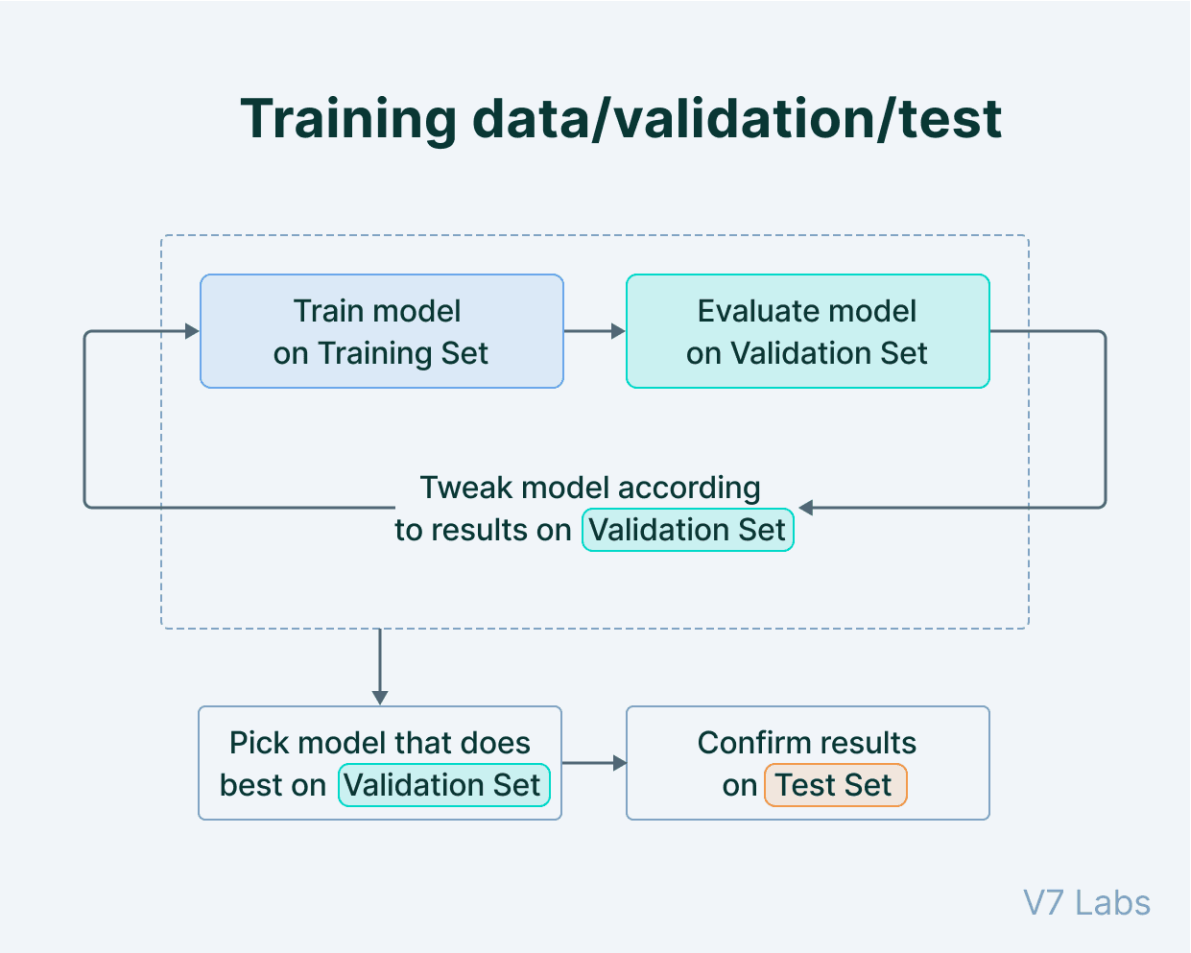
Before you build models

- **Do think about how & where you will use data**
- **Do take the time to understand your data**
  - Start with some exploratory data analysis
  - Data quality matters
- **Do clean your data**
  - Missing values, outliers
- **“Do not look at all of your data”**
  - Avoid making untestable assumptions that will later feed into the model
- **Do make sure you have enough data**
  - Depends on signal to noise ratio of the data
  - Enough for training, validation(s), and test
  - More than one dataset to mitigate bias
  - Carefully repartition datasets if small and biased
- **Do talk to domain experts**
  - In biology, we need the results to be interpretable
- **Do survey the literature**
  - What’s already been tried
- **Do think about how your model will be deployed**
  - Cost, constraints, how will it tie with other tools

How to reliably build models

- Do not allow the test data to leak into the training process*
- Do be careful where and how you do feature selection*
- Do try a range of different models
- Do not use inappropriate models
- Do keep up with progress in deep learning (and its pitfalls)*
- Do not assume deep learning will be the best approach*
- Do optimize your model's hyperparameters
- Do avoid learning spurious correlations*

Splitting Data for Training, Validation and Test



Do not allow test data to leak into the training process

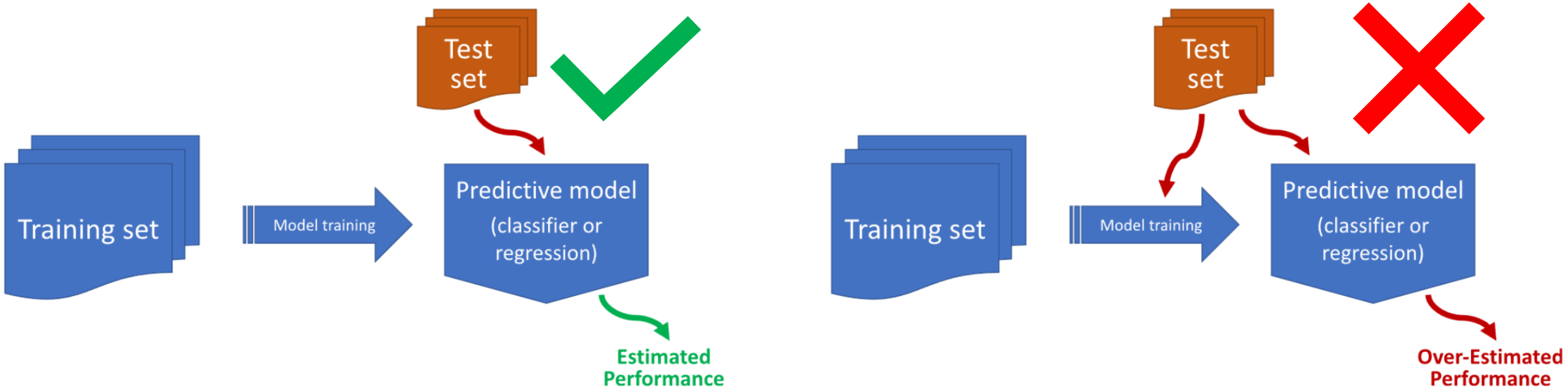
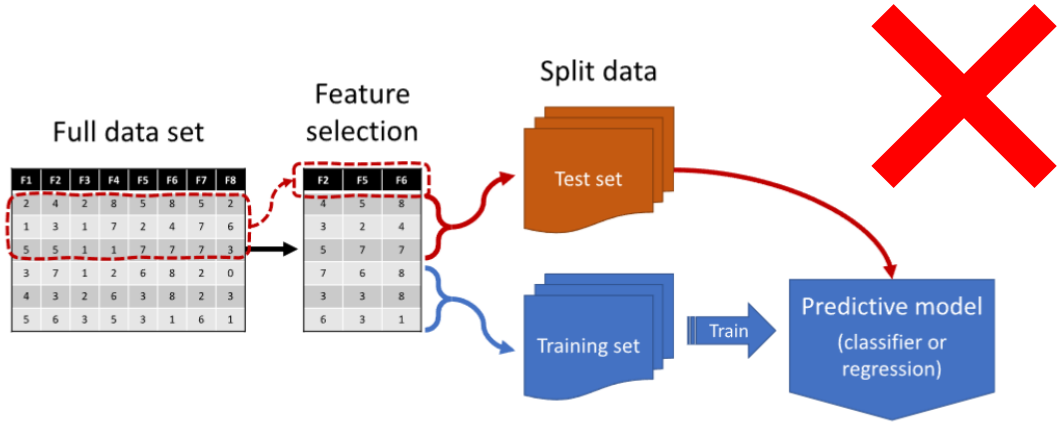


Figure 1. See "do not allow test data to leak into the training process"
(Left) How things should be, with the training set used to train the model and the test set used to measure its generality. (Right) When there is a data leak, the test set can implicitly become part of the training process, meaning that it no longer provides a reliable measure of generality.

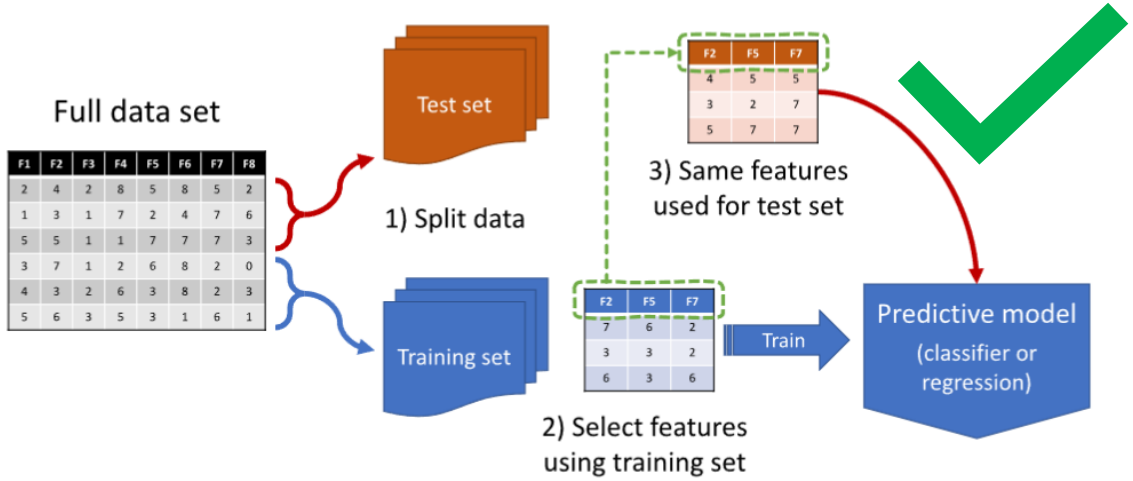
- Validation and Test data needs to be uncompromised and **used only for validation or testing, not for training**
- **Unseen test data** is needed to accurately assess how well the model generalizes
- Data leakage can occur in many scenarios including data normalization, imputation, feature selection, which need to be performed only on training data and not the validation or test data

Do be careful where and how you do feature selection

An important step in training is to select features:



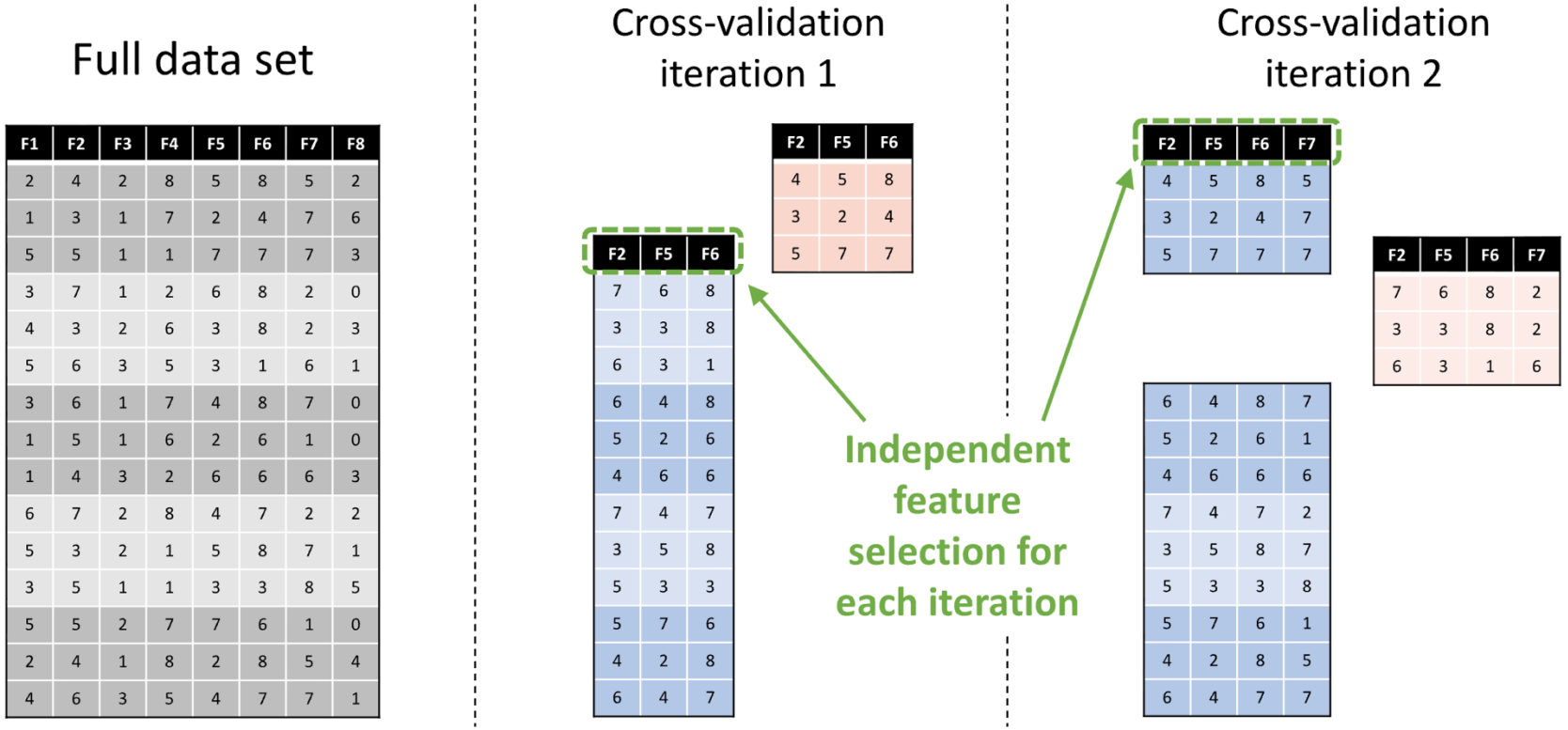
If you select features on the whole dataset before splitting, information can leak from test set to training set



Instead, one should only use the training set to select features used in training set and then later in the test set

- Similarly, component weights for features should be determined only by training data, and same weights should apply to test data later

Do be careful where and how you do feature selection

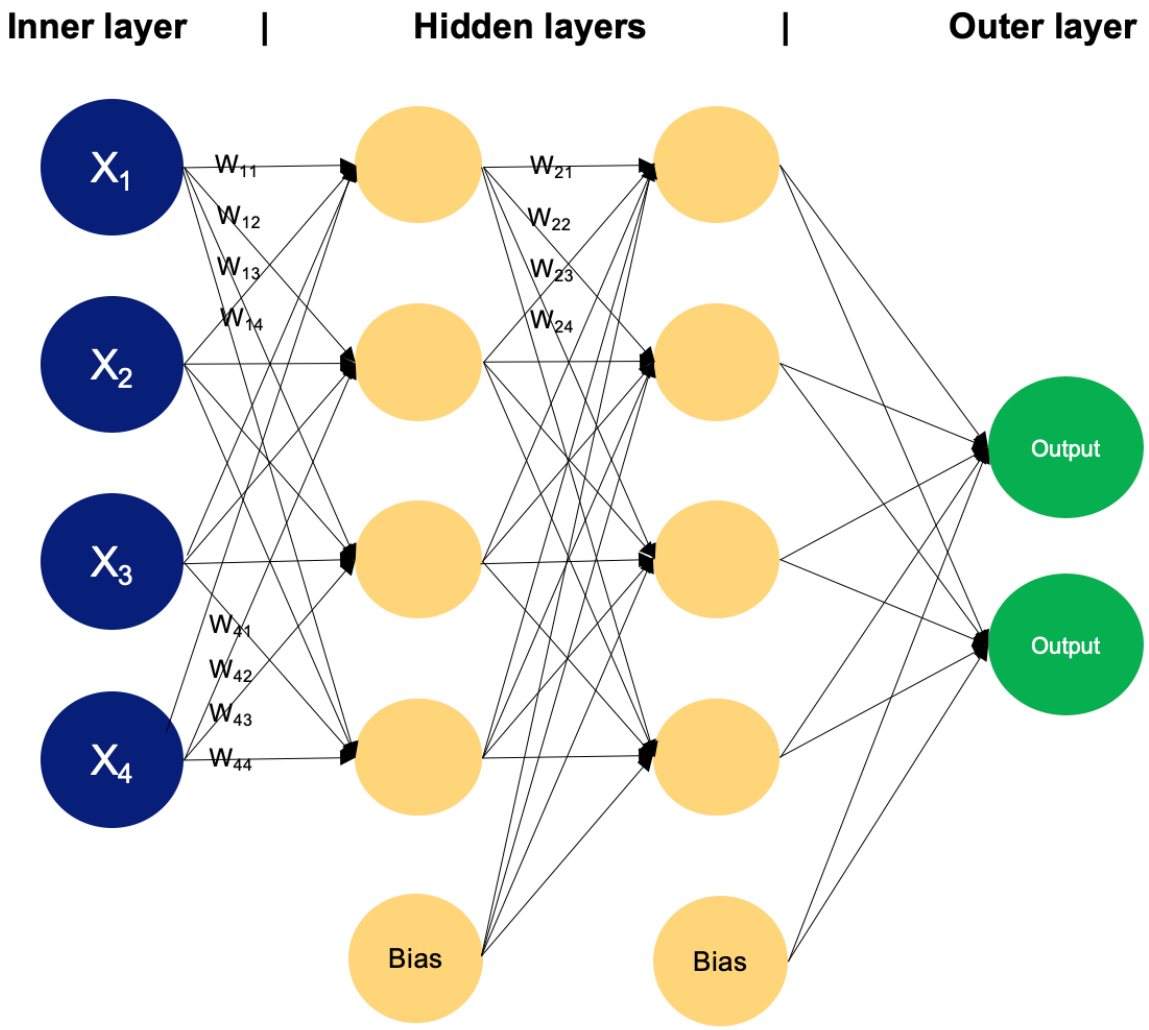


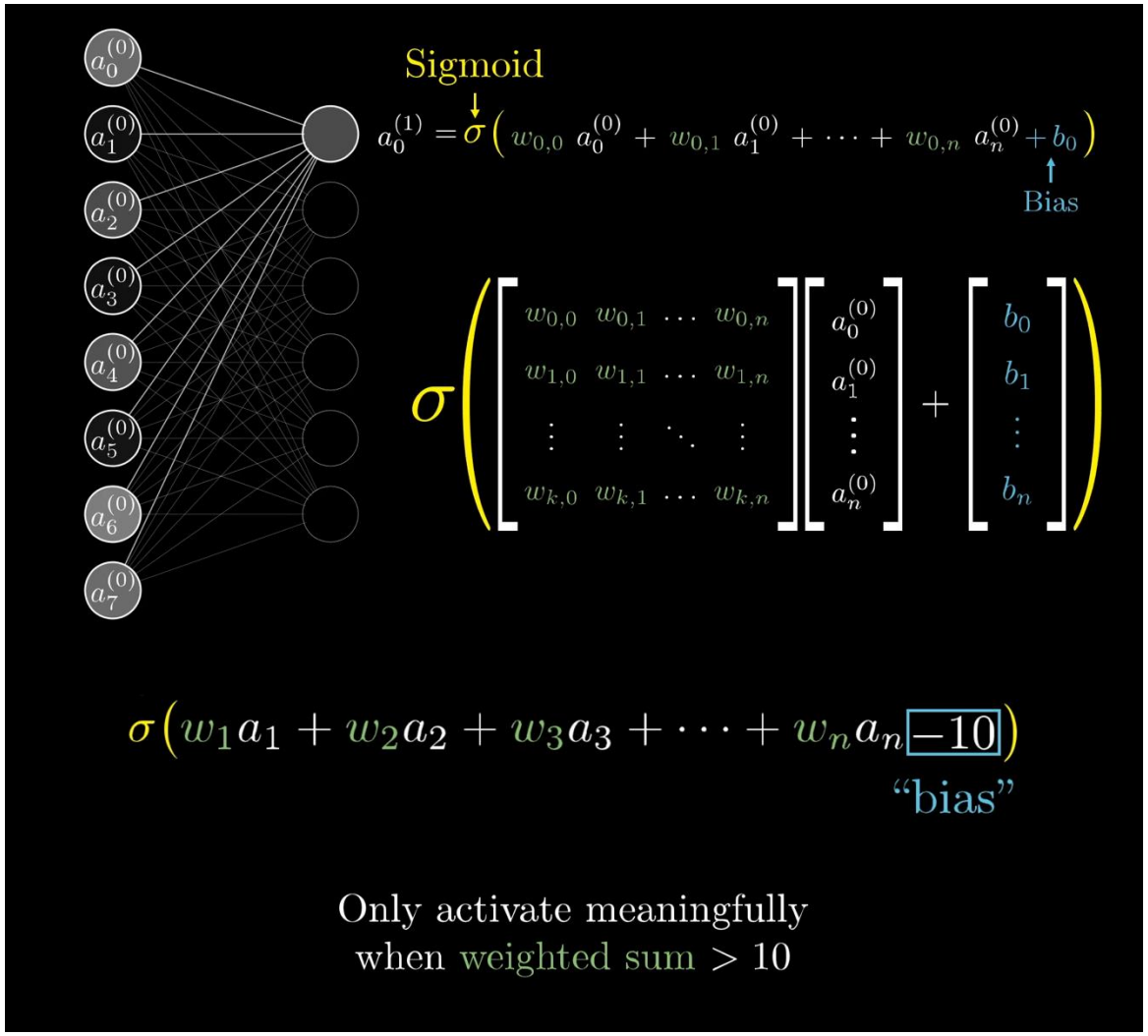
In cross validation (where you might train the model on different iterations of the training data), feature selection should be performed independently for each iteration, based only on the subset of the data used in that iteration

Do try out a range of different models

- The “no free lunch theorem” applied to ML: no single ML approach is best when applied over every possible problem
  - Your goal is to find the ML model that works well for your problem
  - ML models will always have inductive biases that make them incapable of modeling certain kinds of relationships
- Modern ML libraries (*scikit-learn* in Python, *tidymodels* in R etc) allow experimenting with many models
  - But be aware of model appropriateness, evaluate with multiple validation sets, and and vary hyperparameters to give different model types “a fair chance”

Multilayer perceptron (MLP) as the basis for deep learning / neural network architecture





https://blog.gopenai.com/day-12-multi-level-perceptron-mlp-and-its-role-in-llms-a942e4a9e0c8

https://www.3blue1brown.com/lessons/neural-networks

Do keep up with progress in deep learning (and its pitfalls)

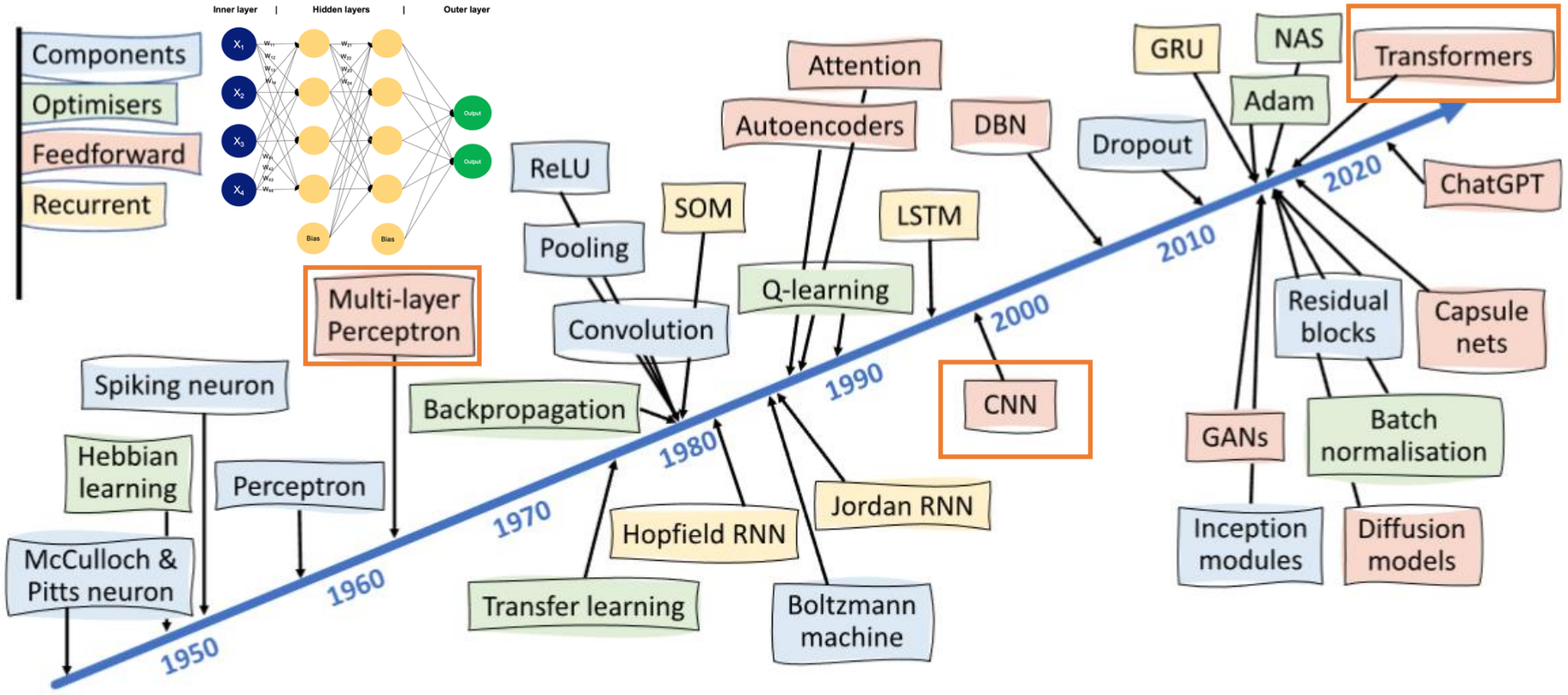


Figure 2. See "do keep up with progress in deep learning (and its pitfalls)"
A rough history of neural networks and deep learning showing what I consider to be the milestones in their development. For a far more thorough account of the field's historical development, take a look at Schmidhuber.[37,38]

Multi-layer perceptrons and **recurrent neural networks** have been around for decades but have been subsumed by **convolutionary neural networks** and **transformers**. However, CNNs and transformers require **a lot** of data to train. Larger pretrained models are often called "foundational models", but these have downsides like memorizing data.

Do not assume deep learning will be the best approach

- There are many examples of deep learning being outperformed by traditional ML models such as random forests or support vector machines

A deep learning model is unlikely to be a good choice if you have limited data, domain knowledge suggest the underlying pattern is quite simple, or the model needs to be interpretable.

The last point is particularly worth considering: a deep neural network is essentially a very complex piece of decision making that emerges from interactions between a large number of non-linear functions. Non-linear functions are hard to follow at the best of times, but when you start joining them together, their behavior gets very complicated very fast.

While explainable AI (xAI) methods can shine some light on the workings of deep neural networks, they can also mislead you by ironing out the true complexities of the decision space.

- High stakes or safety critical decisions are not well suited

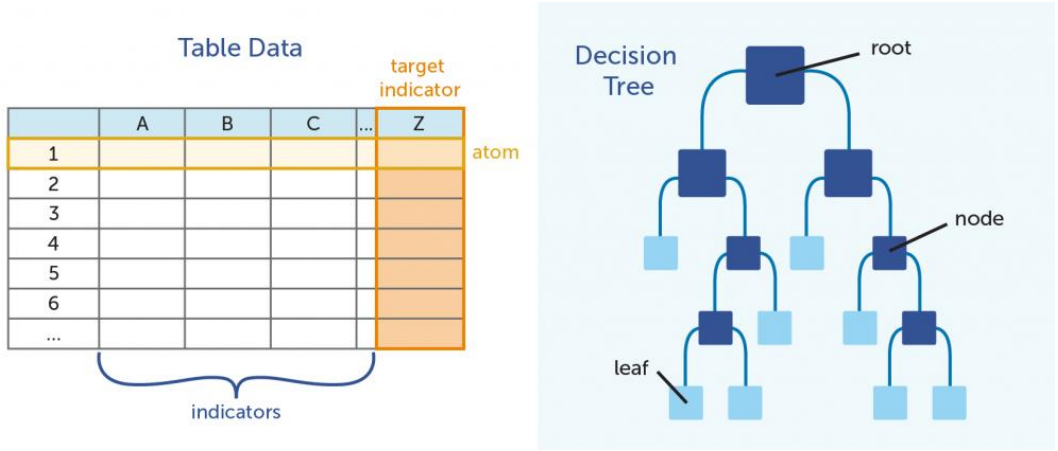
Do not assume deep learning will be the best approach

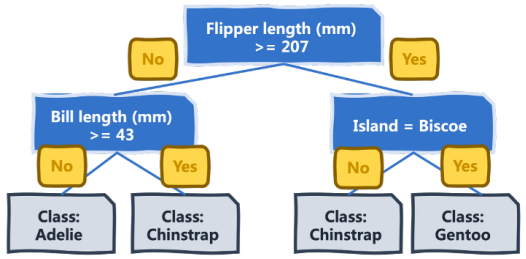
Why do tree-based models still outperform deep learning on tabular data?

Léo Grinsztajn
Soda, Inria Saclay
leo.grinsztajn@inria.fr

Edouard Oyallon
ISIR, CNRS, Sorbonne University

Gaël Varoquaux
Soda, Inria Saclay





Position: Why Tabular Foundation Models Should Be a Research Priority

Boris van Breugel [1] Mihaela van der Schaar [1 2]

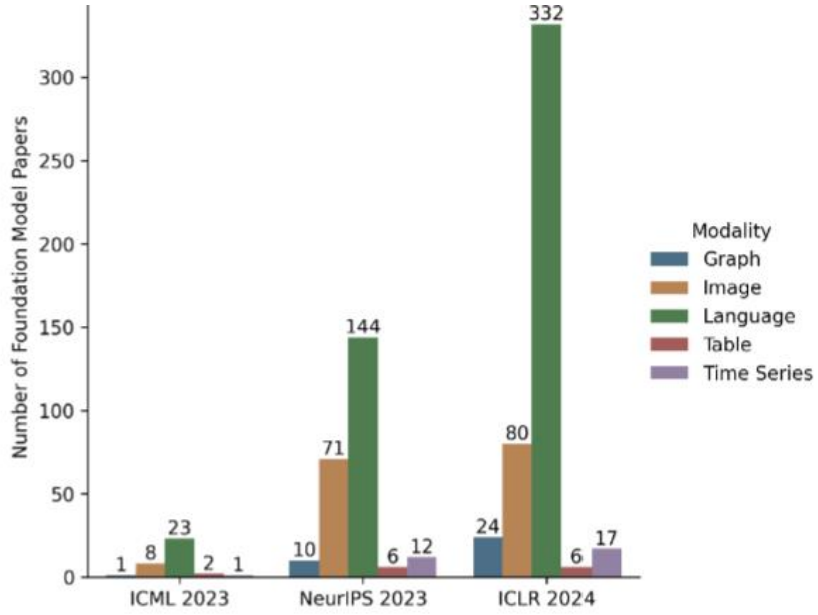


Figure 1. Representation of different modalities in foundation model research across recent ML conferences, roughly estimated as the number of accepted papers with abstracts containing keywords (see Appendix A). LLMs are booming and tabular data is heavily underrepresented.

Do avoid learning spurious correlations

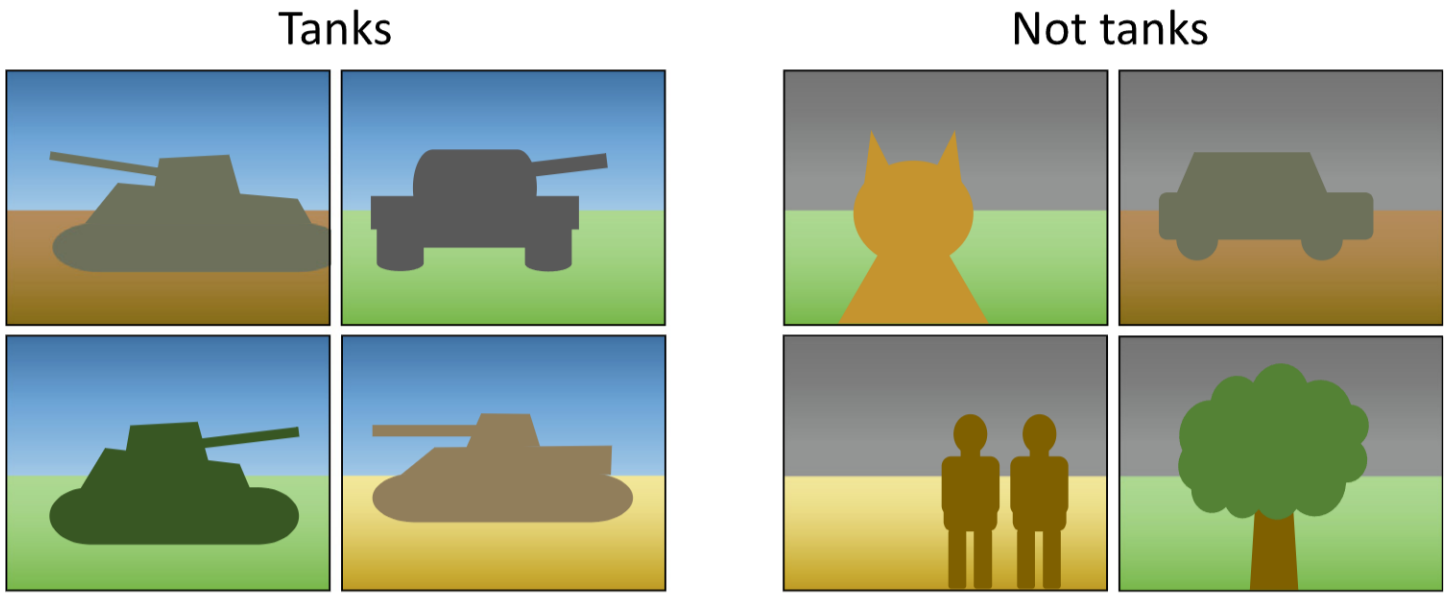


Figure 4. See "do avoid learning spurious correlations"
The problem of spurious correlations in images as illustrated by the tank problem. The images on the left are tanks, and those on the right are not tanks. However, the consistent background (blue for tanks, gray for others) means that these images can be classified by merely looking at the colors of pixels toward the top of the images rather than having to recognize the objects in the images, resulting in a poor model.

Spurious correlations are "red herring" features within data that are correlated with target variable but have no semantic meaning. ML models can inadvertently train on them and fail to generalize to other scenarios or real-life data. More complex data have more spurious correlations, more complex models have ability to fit spurious correlations.

Strategies to mitigate:

regularization - penalizes overly complex models which can cause overfitting or early stopping of training

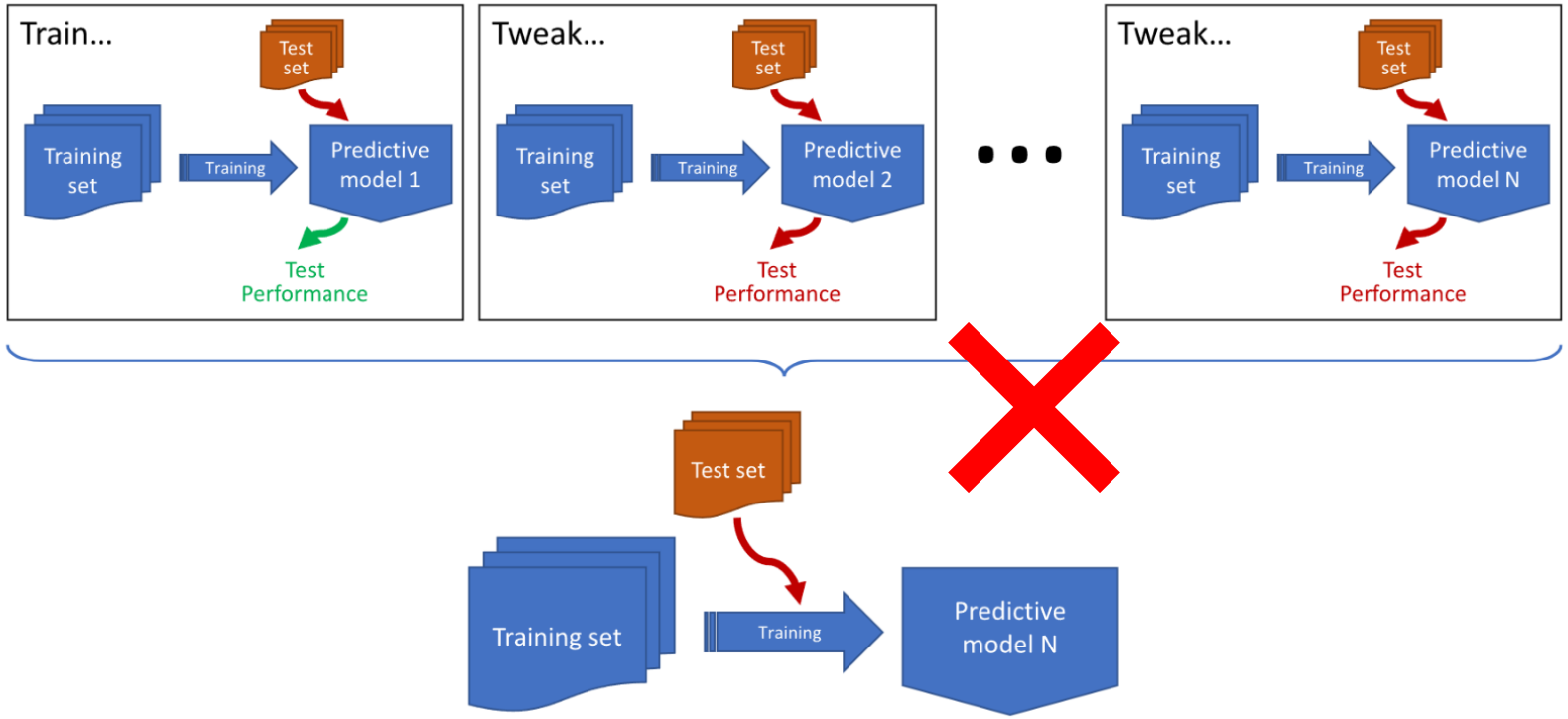
data augmentation - create varied versions of the data for training with transformations e.g. rotation, color swapping, adding noise - the model is exposed to a wider variety of patterns to learn general patterns rather than dataset-specific noise

How to robustly evaluate models

- **Do use an appropriate test set**
  - Should not overlap with training set
  - Should be representative of the wider population
  - Beware when a single piece of equipment is used to collect all the data
  - If using public datasets, be wary of "Frankenstein" datasets assembled from datasets which may overlap with training data
  - Keep datapoints for a subject together (ie. different modalities) to prevent data leakage
- **Do not do data augmentation before splitting your data**
  - Model might overfit augmented data rather than original samples
  - Or worse, augmented versions of data might end up in test set (data leakage)
- **Do avoid sequential overfitting***
- **Do evaluate a model multiple times**
  - Models are often stochastic, small changes to training data will vary performance significantly (use cross validation to split training data into folds)
  - Measure model performance with multiple evaluations and report average + stdev
- **Do save data to evaluate final model instance**
- **Do choose metrics carefully***
- **Do consider model fairness**
- **Do not ignore temporal dependencies in time-series data***

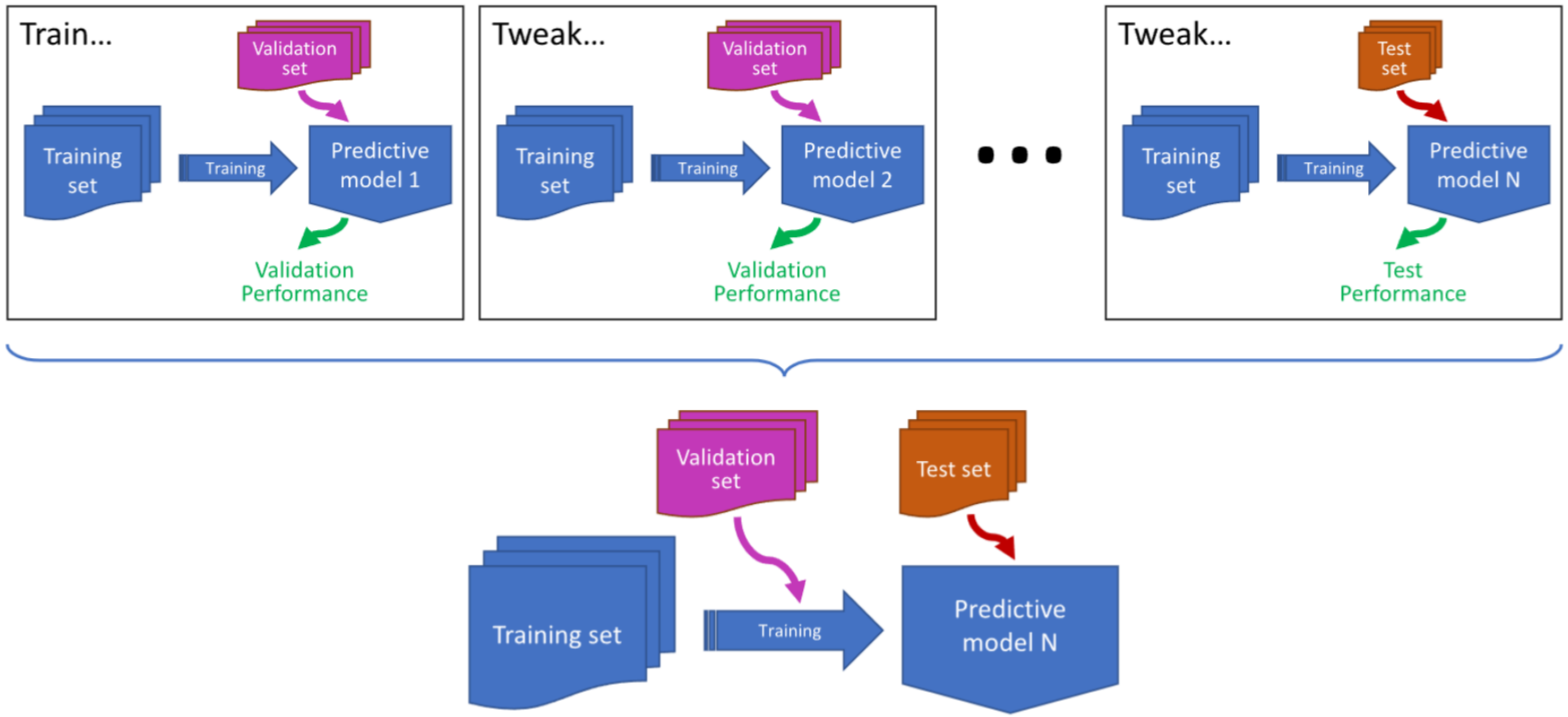
Do avoid sequential overfitting

Also called over-hyping: overfitting of hyperparameters



- **Sequential overfitting** occurs when you train multiple models in succession using knowledge gained about each model's performance to tweak/guide the configuration of each one, and you use the same test set to evaluate each model
- Often done informally and without documentation, where one tries different models and hyperparameters until you get a good result on the test set.
- The test set ends up becoming an implicit part of the training process and the model trains to fit the test set ☹

Do avoid sequential overfitting



It is better to use **validation dataset** to tweak parameters (separate samples not directly used in training but used to guide training) and then later use a final holdout dataset to test the final model ☺

Do not ignore temporal dependencies in time- series data

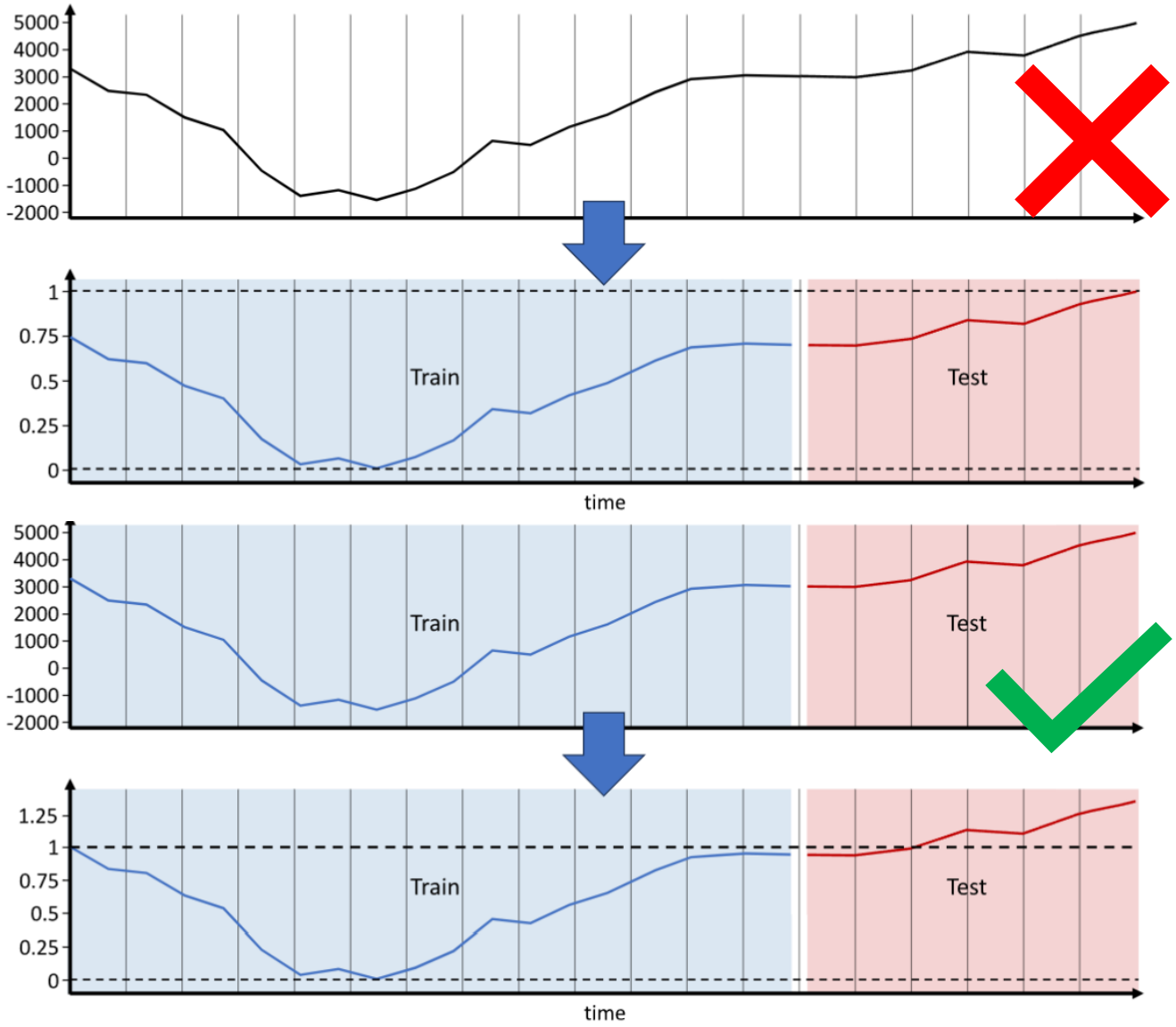


Figure 7. See "do not ignore temporal dependencies in time-series data"
(Top) A time series is scaled to the interval [0, 1] before splitting off the test data (shown in red). This could allow the model to infer that values will increase in the future, causing a potential look ahead bias. (Bottom) Instead, the data should be split before doing scaling so that information about the range of the test data cannot leak into the training data.

The order of datapoints is important in time-series data. Time series data is subject to a particular kind of data leakage known as **"look ahead" bias** and occurs when data points used in training occur later than those used to test the model. This can allow knowledge of the future to leak into training.

This can be avoided when taking care when doing cross validation to use methods of CV (e.g. blocked CV) that avoid training with data that occurs later

← Another form of look ahead bias where time-series data is scaled (0 to 1) prior to splitting, which could allow model to infer values will increase in future (top). Instead, the data should be split before scaling (bottom)

Do choose metrics carefully

F1	F2	F3	F4	F5	Class
2	4	2	8	5	A
1	3	1	7	2	A
5	5	1	1	7	A
3	7	1	2	6	A
4	3	2	6	3	A
5	6	3	5	3	B
3	6	1	7	4	B
1	5	1	6	2	B
1	4	3	2	6	B
6	7	2	8	4	B

Always predict class A

Predicted	Correct?
A	TRUE
A	TRUE
A	TRUE
A	TRUE
A	TRUE
A	FALSE
A	FALSE
A	FALSE
A	FALSE
A	FALSE

Number of correct classifications

$$\text{Accuracy} = \frac{5}{10} = 50\%$$

Total number of classifications

Figure 6. See "do choose metrics carefully"
The problem with using accuracy as a performance metric on imbalanced data. Here, a dummy model that always predicts the same class label has an accuracy of 50% or 90% depending on the distribution of class labels within the data.

In classification models, the most commonly used metric is accuracy (proportion of samples in data correctly classified). This works if your classes are balanced but could be misleading otherwise.

F1	F2	F3	F4	F5	Class
2	4	2	8	5	A
1	3	1	7	2	A
5	5	1	1	7	A
3	7	1	2	6	A
4	3	2	6	3	A
5	6	3	5	3	A
3	6	1	7	4	A
1	5	1	6	2	A
1	4	3	2	6	A
6	7	2	8	4	B

Always predict class A

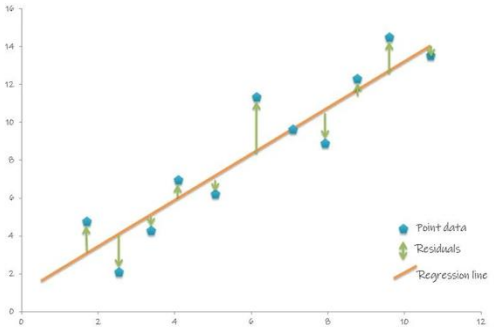
Predicted	Correct?
A	TRUE
A	TRUE
A	TRUE
A	TRUE
A	TRUE
A	TRUE
A	TRUE
A	TRUE
A	TRUE
A	FALSE

Number of correct classifications

$$\text{Accuracy} = \frac{9}{10} = 90\%$$

Total number of classifications

Consider a model that always predicts one class, it would show high accuracy on a dataset that is 90% that class. Another metric that is relatively insensitive to class imbalance would be better (e.g. F1 score, Cohen's kappa coefficient or Matthew's correlation coefficient).



In regression, overrelying on RMSE (a bit like accuracy) is susceptible to models that predict no change (e.g. previous value or average of dataset values), these will often have relatively high RMSE even though the model has predicted no relationship.

How to compare models fairly

- **Do not assume a bigger number means a better model**
  - Comparing published models may result in unfair comparisons, esp if one model has been optimized more than the other
  - Instead: fresh implementation, optimization to similar degree, multiple evaluations, and statistical tests to see if the difference in performance is significant
  - Be careful with foundation models since in many cases the original training data is unknown
- **Do use meaningful baselines***
- **Do use statistical tests when comparing models**
- **Do correct for multiple comparisons**
- **Do not always believe results from community benchmarks***
- **Do combine models (carefully)***

Do use meaningful baselines

nature methods

Brief Communication

https://doi.org/10.1038/s41592-025-02772-6

Deep-learning-based gene perturbation effect prediction does not yet outperform simple linear baselines

Received: 11 October 2024

Accepted: 24 June 2025

Published online: 4 August 2025

Check for updates

Constantin Ahlmann-Eltze[1,2,3] ✉, Wolfgang Huber[2] & Simon Anders[1]

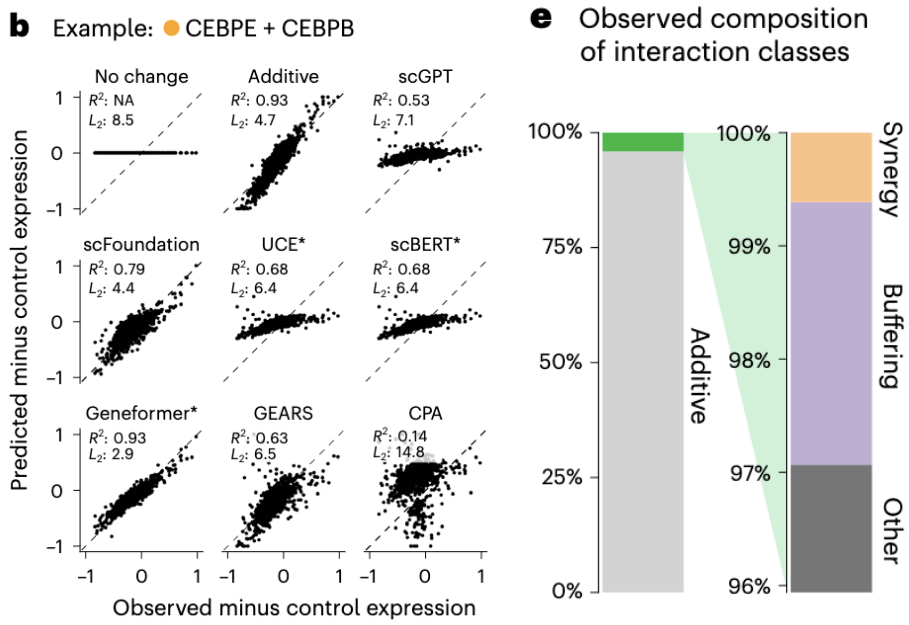
Recent research in deep-learning-based foundation models promises to learn representations of single-cell data that enable prediction of the effects of genetic perturbations. Here we compared five foundation models and two other deep learning models against deliberately simple baselines for predicting transcriptome changes after single or double perturbations. None outperformed the baselines, which highlights the importance of critical benchmarking in directing and evaluating method development.

We fine-tuned the models on all 100 single perturbations and on 62 of the double perturbations and assessed the prediction error on the remaining 62 double perturbations. For robustness, we ran each analysis five times using different random partitions.

For comparison, we included two simple baselines: (1) the 'no change' model that always predicts the same expression as in the control condition and (2) the 'additive' model that, for each double perturbation, predicts the sum of the individual logarithmic fold changes (LFCs). Neither uses the double perturbation data.

All models had a prediction error substantially higher than the additive baseline (Fig. 1a,b).

Do compare against established approaches / baseline models. Baseline models are chosen to demonstrate that complexity in the new model is necessary.



The 'no change' baseline cannot, by definition, find synergistic interactions, but also the deep learning models rarely predicted synergistic interactions, and it was even rarer that those predictions were correct (Fig. 1f).

Comparison examples:

Tabular data – compare w/ SVM and decision trees

Regression – compare w/ logistic regression

Time series – compare w/ naïve baseline where next value is no change

Classification – compare w/ naïve baseline where most frequent class label is chosen

Do not always believe results from community benchmarks

- The idea behind community benchmarks is that everyone is using same data to train and test their models
  - But if access to the test set is unrestricted, then you cannot assume that people have not used it as part of their training process ie. “training to the test set” (esp an issue with foundation models)
  - More subtle issue: even if everyone who uses test set only uses it once, *collectively the test set is still being used many times by the community*. It then becomes increasingly likely that the best model “just happens” to fit the test set and does not necessarily generalize any better
  - Therefore, be careful how much you read into any one benchmark dataset

Do combine models (carefully)

- ML is not always about choosing between models, often in makes sense to use combinations of models since each has different tradeoff of strengths and weaknesses
- **Ensembles** are a well-established type of composite models
  - Can be composed out of same base model type (e.g. random forest, bagging, boosting)
  - Can be composed of aggregated outputs of a group of different kinds of base models (e.g. stacked generalization, or stacking)
- There may be other ways of combining models, such as by using one model to feed input to another model

How to report results

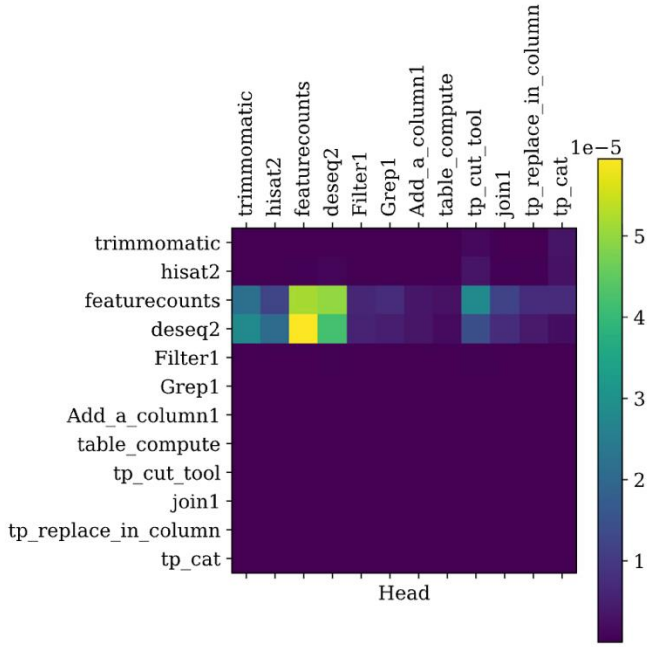
- **Do be transparent**
- **Do report performance in multiple ways**
- **Do not generalize beyond the data**
  - Be careful not to make general statements not supported by data used to train and evaluate models
  - Be aware of bias or sampling error (may not fully represent real world)
  - Multiple datasets may not be independent and be biased
  - In deep learning datasets, the need for data quantity can supersede data quality metrics
- **Do be careful when reporting statistical significance**
  - The difference between two models may not be that large
  - Other than p-values, effect size is important. Also consider reporting Bayesian statistics
- **Do look at your models***
- **Do use an ML checklist**
  - REFORMS checklist for consensus-based recommendations for doing ML-based science and practice

Do look at your models

- Try to understand how models reach a decision
- Visualize models and try different xAI tools or approaches
  - **Saliency maps** in CNNs and vision transformers can give sense of the important parts of the input
  - In non-vision transformers, look at the **attention weights**
  - Model agnostic methods like **LIME** (local-interpretable model-agnostic explanations) and Shapley additive explanations (**SHAP**) can pinpoint which features are important for a model

"While xAI techniques can give you useful insights into a model's behavior, it is important to bear in mind that they are unlikely to tell you exactly what a model is doing. This is particularly the case for deep learning models, whose complexity makes their behavior inherently difficult to analyze."

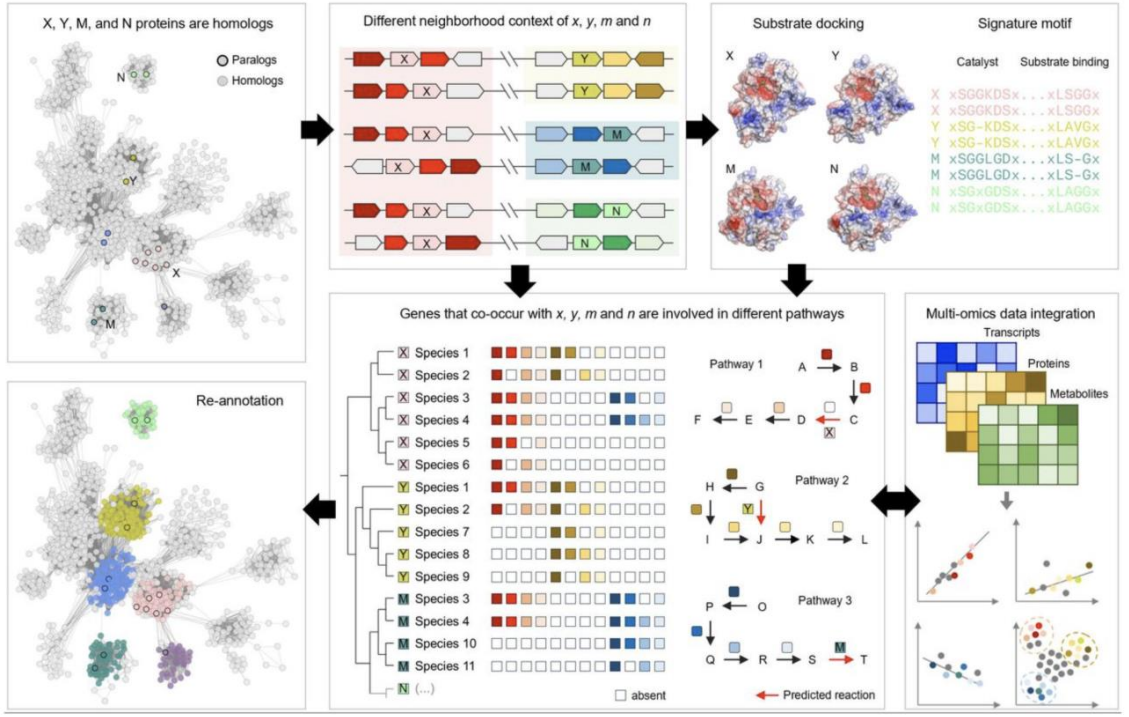
- **Ablation** studies can be useful for complex models
  - Successively remove parts of the model to see what is important
  - May result in simpler models, which is better



Deep learning gets the glory, deep fact checking gets ignored

When impressive AI biology results are full of errors

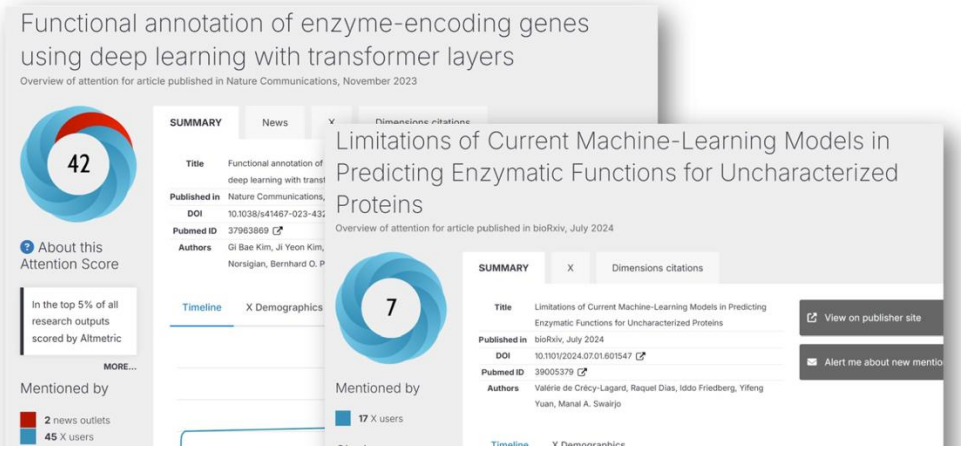
https://rachel.fast.ai/posts/2025-06-04-enzyme-ml-fails/



It is important to look at multiple types of evidence when classifying enzyme function (Fig 2 from de Crecy, et al.)

Hundreds of Likely Erroneous Results

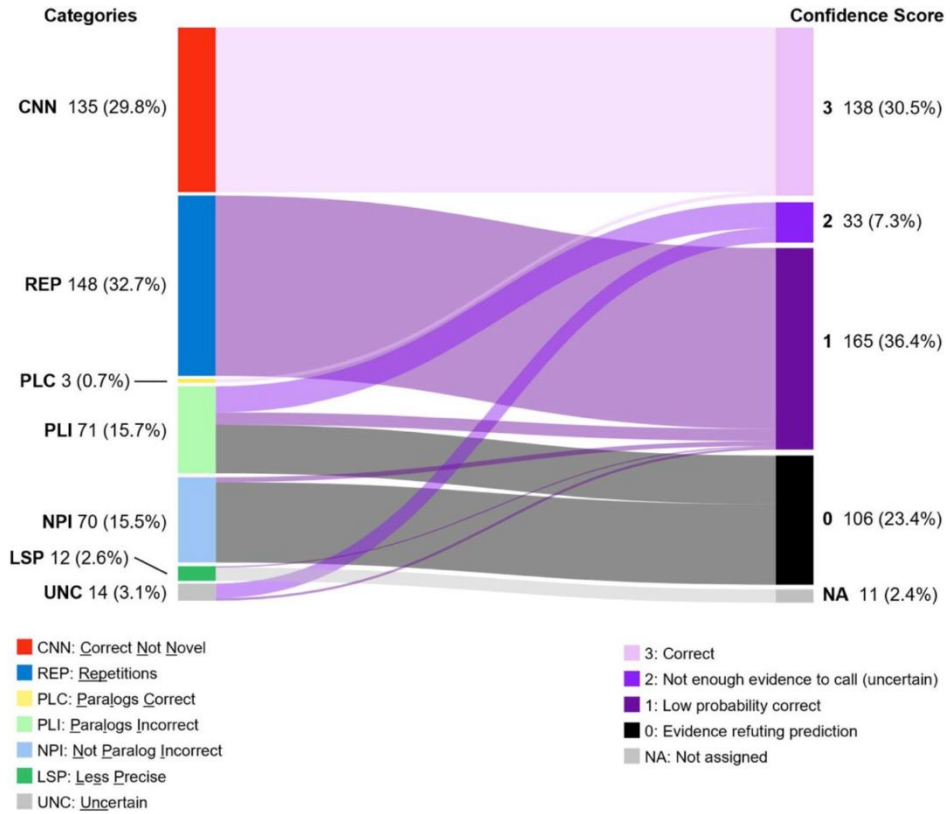
Spotting this one error inspired de Crécy-Lagard and her co-authors to take a closer look at all of the enzymes found to have novel results in the Kim, et al, paper. They found that 135 of these results were already listed in the online database used to build the training set and thus not actually novel. An additional 148 of the results contained a very high level of repetition, with the same highly specific functions reappearing up to 12 times. Biases, data imbalance, lack of relevant features, architectural limitations, or poor uncertainty calibration can all lead models to "force" the most common labels from the training data.



The Errors

The Transformer model in the Nature Communications paper made hundreds of "novel" predictions that are almost certainly erroneous. The paper had followed a standard methodology of evaluating performance on a held-out test set, and did quite well on that (although later investigation suggests there may have been data leakage). The results claimed for enzymes where no ground truth is known were full of errors.

Of the 450 "novel" results given in the paper, 135 of these results were not novel at all; they were already listed in the online database UniProt. Another 148 showed unreasonably high levels of repetition, with the same very specific enzyme functions reappearing up to 12 times for genes of *E. coli*, which biologically implausible.



Most of the "novel" results from the transformer paper were either not novel, unusually repetitious, or incorrect paralogs (Fig 5 from de Crecy, et al.)

Takeaways

- **Complex is not necessarily better**
- **Interpretability is key for scientific insight**
- Many easy mistakes can cause an ML model to appear to perform well when, in reality, it does not
- This can cause misinformation and harm
- Fairness, transparency and avoidance of bias are important
- Fast-moving research – “the theory of how to do ML almost always lags behind the practice”