Title
Navigating the pitfalls of applying machine learning in genomics.

Journal
Nature reviews. Genetics, 23(3)

ISSN
1471-0056

Authors
Whalen, Sean
Schreiber, Jacob
Noble, William S
et al.

Publication Date
2022-03-01

DOI
10.1038/s41576-021-00434-9

Peer reviewed

Navigating the pitfalls of applying machine learning in genomics

Sean Whalen[1,6], *Jacob Schreiber*[2,6], *William S. Noble*[3] *and Katherine S. Pollard*[1,4,5]✉

Abstract | The scale of genetic, epigenomic, transcriptomic, cheminformatic and proteomic data available today, coupled with easy-to-use machine learning (ML) toolkits, has propelled the application of supervised learning in genomics research. However, the assumptions behind the statistical models and performance evaluations in ML software frequently are not met in biological systems. In this Review, we illustrate the impact of several common pitfalls encountered when applying supervised ML in genomics. We explore how the structure of genomics data can bias performance evaluations and predictions. To address the challenges associated with applying cutting-edge ML methods to genomics, we describe solutions and appropriate use cases where ML modelling shows great potential.

Examples
Also known as 'samples' or 'observations'. The primary data objects being manipulated by a machine learning system. They are the basic units being measured.

Training set
Examples and associated outcomes that are used to fit a supervised machine learning model.

[1]*Gladstone Institutes, San Francisco, CA, USA.*

[2]*Department of Genetics, Stanford University, Stanford, CA, USA.*

[3]*Department of Genome Science, University of Washington, Seattle, WA, USA.*

[4]*Department of Epidemiology & Biostatistics, University of California, San Francisco, CA, USA.*

[5]*Chan Zuckerberg Biohub, San Francisco, CA, USA.*

[6]*These authors contributed equally: Sean Whalen, Jacob Schreiber.*

✉*e-mail: katherine.pollard@gladstone.ucsf.edu*

https://doi.org/10.1038/s41576-021-00434-9

As the amount and complexity of genomic data rapidly increase, machine learning (ML) tools are being used for a wide array of analytical tasks. Examples include processing and normalizing raw data, integrating heterogeneous sources of genomic information, exploring data structure, predictive modelling, generative modelling and prioritizing experiments. In this Review, we discuss several pitfalls that one might encounter when applying ML tools to genomics data. We focus on supervised ML in which the goal is to learn a model that makes accurate predictions on new data and/or yields mechanistic insight via identification of genomic features that underlie a predictive model.

A crucial first step in applying supervised learning in genomics is identifying the most useful formulation of the scientific question as a ML task. Some tasks have limited scientific value. For example, predicting a cheap form of experimental data from a more expensive form may not be practically useful unless the expensive data are already available or the model itself provides some specific biological insight. Once one has identified a general problem for which ML can be helpful, some specific formulations of the problem may be more useful than others. For instance, a model trained to predict gene expression from promoter sequences using data in one cell type may not generalize to a new cell type in which expression levels are driven by other transcription factors. In addition, circular problems should be avoided, such as predicting protein functions from a protein interaction database that links two proteins if they share a Gene Ontology (GO) category or other functional annotation. Here, the circular issue is that the label being predicted (protein function) is actually directly encoded in the features used for prediction.

Once the analyst has identified the specific ML problem that will be solved, they must choose a model and determine how to properly evaluate its performance. Performance evaluation is often executed using cross-validation (BOX 1), whereby examples are iteratively randomized into a training set used to fit a model and a held-out test set used to quantify model performance. This is where one typically encounters the pitfalls that we discuss. The primary problem is that examples are assumed to be independent and identically distributed (IID). But genomics is replete with violations of these assumptions, such as adjacent genomic positions that exhibit correlated activity, or proteins in the same family, pathway or complex that have very similar functions. If modelling assumptions are inaccurate, then the reported predictive accuracy of a model may be substantially inflated compared with the true generalization error the model would have on a completely independent prediction set. Unfortunately, ML software cannot detect this; instead, it is the job of the modeller to check assumptions and adjust their statistical procedures accordingly. Our goal is to highlight specific scenarios where model and performance evaluation assumptions are violated, along with solutions for mitigation.

This Review is organized around five pitfalls that arise when applying supervised ML models in genetics and genomics (FIG. 1). Several problems we explore have been touched upon by Teschendorff[1], Minhas et al.[2] and in reviews of genomics applications for deep learning[3–6]. We cover these alongside many new topics to provide a complete picture of the interrelated pitfalls that arise when applying supervised ML in genomics. Owing to their connections, the pitfalls share some mitigation strategies (see the Conclusions section), but we discuss each one separately because they can occur in isolation.

Box 1 | **Terminology of performance evaluation**

Cross-validation

In this procedure for estimating the performance of a supervised machine learning (ML) model on held-out data, examples are randomly split into *k* groups, known as folds. The model is fit on *k*–1 groups (the training set) and then applied to the remaining group (the test set). By comparing predictions with actual outcomes across many random test sets, model performance on unseen data can be estimated. Group *k*-fold cross-validation, or blocking, is a variant of cross-validation (CV) that takes into account information about groups of dependent examples, such as which chromosome a gene is located on or the patient from which a sample was derived. In group *k*-fold CV, when splitting into *k* folds, all examples belonging to the same group are assigned to the same fold. In this way, examples that belong to the same group cannot cross the train–test divide.

Performance statistics

We can summarize the performance of a binary classifier in various ways.

- The true positive rate (also known as 'recall' and 'power') is the number of true positives divided by the total number of positives.
- The false positive rate (FPR) is the number of false positives divided by the total number of negatives.
- Precision is the number of true positives divided by the total number of predicted positives (true positives plus false positives). The false discovery proportion (FDP) is 1 – precision. The expected (or average) FDP over many iterations of an experiment is the false discovery rate (FDR).
- Accuracy is the proportion of correct predictions made by a classifier. It is calculated as the number of correct predictions (true positives plus true negatives) divided by the total number of predictions.

Visualizing performance

Most classifiers allow the user to vary how many examples get classified as positive (for example, by varying a threshold). As the number of predictions changes, we can plot two types of curves.

- A receiver operating characteristic (ROC) curve plots the true positive rate as a function of the FPR. The area under this curve (auROC) quantifies the performance of the classifier, with an area of 1.0 corresponding to perfect performance and 0.5 corresponding to random chance. The auROC does not depend on the ratio of class sizes in the test set.
- A precision–recall (PR) curve plots precision as a function of recall (also known as the true positive rate). Perfect performance corresponds to an area (auPR) of 1.0, and random chance corresponds to an area of (number of positives/total number of examples). For a fixed classifier, the auPR will change depending on the ratio of the class sizes in the test set.

auROC measures how recall increases as FPR increases, whereas auPR measures how recall increases as FDP increases. FPR and FDP both have the number of false positives as their numerator, but their denominators are different: the number of negatives for FPR versus the number of predictions for FDP. As a consequence, auPR can be low when auROC is high.

Test set
Examples and associated outcomes that are used to evaluate model performance. Training and test sets are disjoint.

Independent
The value of one example does not depend on the value of others.

Identically distributed
Generated by the same underlying distribution, with a particular mean, variance and shape.

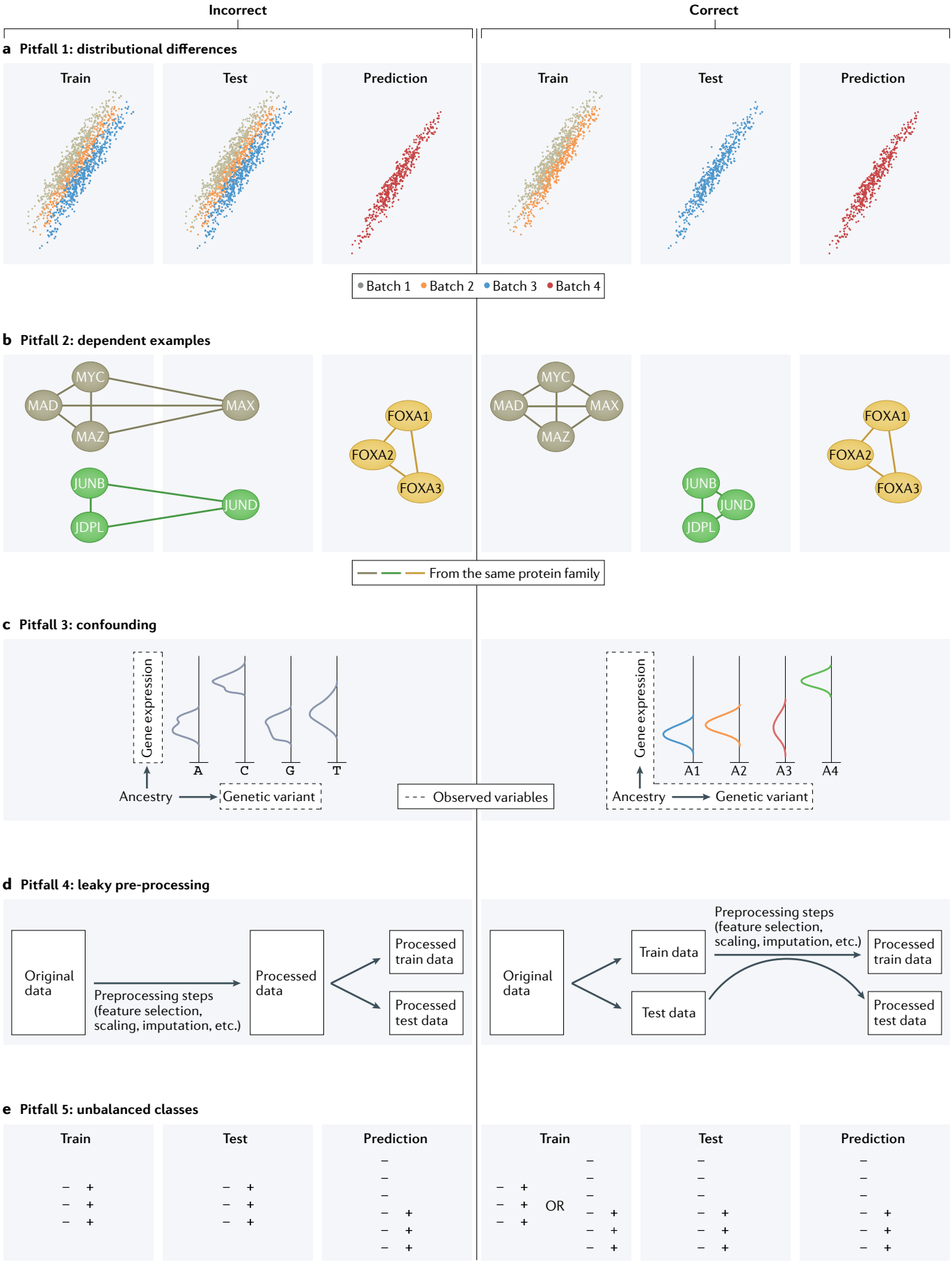
Each problem is described in generality and through illustrative examples selected from many in the literature. To further demonstrate these ideas quantitatively, we developed interactive notebooks with data and code that can be downloaded and run locally or in a web browser without installing any software. These notebooks use epigenomic data or simulations, some with no true associations between features and the outcome, to quantify the effects of the pitfalls and demonstrate their mitigation. Satisfying model assumptions is hard in complex biological systems, but we hope that the strategies presented here increase the usefulness of ML in genomics.

Pitfall 1: distributional differences

This first pitfall concerns examples that are not identically distributed, meaning that the probability of observing a given value is not the same across examples (FIG. 1a). Repeated coin tosses are identically distributed (the probability of heads is the same each time), whereas online search results are not (they vary by season). Distributional differences can affect the features (whose marginal distribution is denoted as $P(x)$), the outcome ($P(y)$), and/or the relationship between features and outcome (the conditional distribution $P(y|x)$).

In genomics, distributional differences arise for many reasons. A common cause is inherent biological structure in the data. For example, epigenetic profiles differ between euchromatin and heterochromatin. Proteins belong to functional categories, each with distinct expression patterns and physical interactions. Population genetic structure creates distributional differences known as ascertainment bias in genome-wide association studies (GWAS) when variants are discovered or models are trained on individuals with different ancestry from the population being tested for genotype–phenotype associations[7–10]. Distributional differences also arise when a model is trained and applied in different biological contexts (for example, different cell types, different species or in vitro versus in vivo[11]). Lastly, distributions may differ owing to study design and technical factors

Fig. 1 | **An overview of five common pitfalls. a** | Distributional differences can arise from various sources, such as batch effects. If the training and test sets are a mixture of examples from every batch (left), performance on the testing set will be much higher than on a new batch. To fit a model that will generalize to new batches, training and test sets should be composed of different batches (right). **b** | Dependency structure arises when biological groups exhibit similar feature–outcome relationships, such as correlated functions of proteins from the same family or complex. Predicting outcomes is easy if a model is trained using other entities in the same group (left). To ensure that the model can generalize, one should instead partition entire biological groups into either the training or test set (right). **c** | Confounding variables are unobserved variables that alter dependence structures between the observed variables. In this example, the unmeasured ancestry of individuals is a confounder of the relationship between genetic variants and gene expression (left), causing the C allele to appear associated with higher expression. After adjusting for the ancestry of individuals (right), we see that expression is higher for individuals from one ancestry group (A4). The association between the C allele and expression is only due to C being more common in individuals from A4. Although difficult to discover, confounders should be explicitly included in the modelling approach and construction of training and test sets. **d** | Information leakage can happen when information is leaked from the test set into the training as a result of the training and test sets being preprocessed together (left). Instead, the raw data should be split into training and test sets with preprocessing performed separately (right). **e** | Unbalanced data can make model training and evaluation difficult. If the training and test sets are balanced but the prediction set is unbalanced, test set performance will not reflect prediction set performance (left). Regardless of whether a balanced or unbalanced training set is used, the imbalance in the test set should be reflective of the imbalance in the prediction set (right). Ideally, one should also use a performance measure that can handle imbalance.


Incorrect
Correct
a Pitfall 1: distributional differences
Train
Test
Prediction
Train
Test
Prediction
Batch 1
Batch 2
Batch 3
Batch 4
b Pitfall 2: dependent examples
MYC
MAD
MAX
MAZ
JUNB
JUND
JDPL
FOXA1
FOXA2
FOXA3
From the same protein family
c Pitfall 3: confounding
Gene expression
A
C
G
T
Ancestry
Genetic variant
Observed variables
A1
A2
A3
A4
d Pitfall 4: leaky pre-processing
Original data
Preprocessing steps (feature selection, scaling, imputation, etc.)
Processed data
Processed train data
Processed test data
Train data
Test data
e Pitfall 5: unbalanced classes
OR

such as time, personnel, reagents or instruments[12]. These batch effects are extremely common in genomics, including single-cell experiments[13], and can bias both the mean and the variability of measurements. Most genomic datasets have some distributional differences from these various sources.

Violations of the assumption that examples are identically distributed create several problems for ML modelling. In this pitfall, we focus on when the training and test sets have one distribution and the prediction set has a different distribution. In this case, one should expect that performance will be higher in cross-validation (same setting) than on the prediction set (different setting). Because the prediction and test sets have distributional differences, the relationships between features and outcomes that are learned during model fitting may not hold in the prediction setting.

Fortunately, some distributional differences are simple to identify. Ideally, the marginal distributions of both outcomes and features should be examined. But the outcome is typically unknown in the prediction setting, so one can only assess feature distributions. A straightforward approach is visualization, either by projecting the data into two dimensions and making scatter plots or by comparing histograms of feature values. A more sophisticated approach uses statistical tests to detect when distributions differ[14]; for example, the binomial test for binary features, the Kolmogorov–Smirnov test for univariate continuous features or Maximum Mean Discrepancy for multivariate continuous features[14,15]. Model-based techniques for outlier and anomaly detection can also be used[16–18].

Accounting for distributional differences is still an area of open research. Various batch correction methods are commonly used, such as quantile normalization, empirical Bayes adjustment for measured variables with ComBat[19], surrogate variable analysis for estimating and correcting for unknown sources of noise[20], and canonical correlation analysis, which enables the identification of common patterns across batches[21,22]. It is worth noting that sometimes performing this correction can inadvertently cause information to leak between training and test splits in cross-validation (see pitfall 4). More elaborate approaches from domain adaptation and transfer learning[23] include unsupervised feature transformations and supervised learning of robust feature representations[24,25]. Another solution is adversarial learning, a set of techniques that attempt to fool models by providing them with deceptive inputs. Specifically, a model trained to predict the dataset that each example came from can be used to generate penalties for the primary prediction task[14,26,27].

Case study. Various ML approaches have been used to model the sequence preferences of RNA- and DNA-binding proteins. Transcription factor binding motifs, for instance, are estimated using data from a variety of assays, both in vitro[28,29] (for example, protein-binding microarrays[30–34], HT-SELEX[35–37], MITOMI or HiTS-Flip[38]) and in vivo (for example, ChIP–seq or ChIP–exo). These measurement techniques each have unique biases, including which sequences are assayed and which other proteins, if any, are present during the experiment. Hence, the observed distribution of bound sequences for a given transcription factor is different across assays. When a comprehensive evaluation of methods for modelling mouse transcription factor binding sites was conducted, the authors noted the effect of these distributional differences as a major source of performance disparities[11]. They show that a model learned from in vitro protein-binding microarray data typically performs much better in cross-validation than it does on in vivo ChIP–seq data, with area under the receiver operating characteristic (auROC) curve differing by up to 0.4 (0.5 being the difference between perfect prediction and random guessing). The reverse is also true. Neither setting is superior; they are simply different. The study showed that performance also drops when transcription factor binding models are applied to make predictions on a different species from the one on which they were trained. One should be aware of these performance differences across contexts, but they are not inherently bad. In fact, analysing the sources of differential performance can reveal interesting biological differences between the settings, such as the presence of cofactors or cooperativity in ChIP–seq that is absent from protein-binding microarray data.

Other examples. Distributional differences are common. GWAS data can have differences in allele frequency distributions arising from ancestry. Single-cell and bulk gene expression measurements can have systematic differences across batches. In proteomics, distributional differences across mass spectrometers mean that reproducibility is higher on the same instrument than across instruments[39]. Supervised ML models that predict protein function are frequently applied to proteins from different protein families than those represented in the training data. Outside molecular biology, machine- and hospital-based biases have been observed in medical images and electronic health records[40–45], complicating the development of methods that aim to be deployed to entire medical systems. Another example is when drug repurposing models do not perform well on new drugs or rare diseases where distributions differ from databases on which the models were trained[46].

Pitfall 2: dependent examples

The mathematics of commonly used ML models and cross-validation depends upon the assumption of independence, meaning that the values of one example are not dependent on another example (FIG. 1b). To illustrate the concept, repeated draws from a card deck without replacing the drawn card are dependent, because the probability of the next card depends on what has already been drawn.

In genomics, dependence is pervasive yet can be challenging to recognize. When predicting protein–protein interactions, examples are pairs of proteins. When pairs are represented in a dataset with unique identifiers they may appear to be independent, but all pairs that share a given protein are correlated with each other. Dependent examples are similarly created with enhancer–promoter, regulator–gene and drug–protein interactions[47–50].

Generalization error
A measure of how accurately a model predicts outcomes in data it has never seen before.

Prediction set
A third set of examples whose associated outcomes are truly not known, where a fitted model is applied to make predictions. Also known as a prospective validation set.

True negatives
Negatives whose labels are correctly predicted.

Features
Properties of a given example, for example, the gene expression values associated with a gene or the sequence patterns associated with a genomic window. Also known as 'covariates'.

Outcome
Outcomes are what we want to predict in supervised learning, for example, the functional class assigned to a gene or the binary classification of whether a given genomic window contains a promoter. Categorical outcomes are often referred to as 'labels'. In regression settings, the outcome is a real number.

Ascertainment bias
Examples in a study are not representative of the general population.

Adversarial learning
Machine learning techniques for improving model robustness to distributional differences, such as those caused by batch effects or other confounders.

Frequently, the independence and identical distribution assumptions are entangled. For instance, genotyping results for family members are dependent and also may differ distributionally from other families. Unfortunately, dependence relationships are not always known. Even known dependencies have a tendency to be ignored in supervised ML analyses. When data are formatted as a table with one example in each row, the standard for most ML toolkits, it is easy to proceed with model fitting and cross-validation without checking whether examples are independent (FIG. 2a).

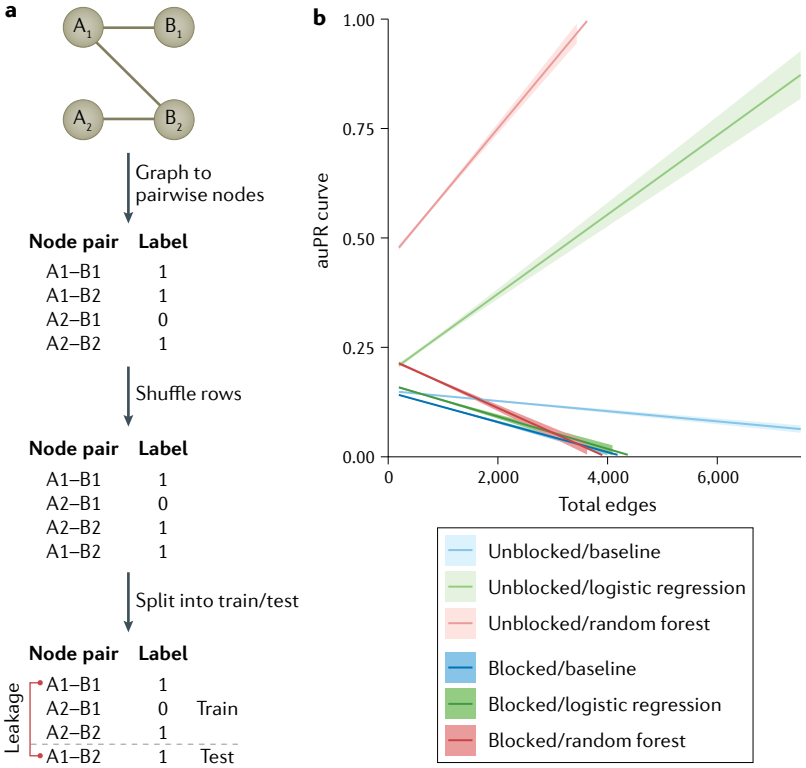


Fig. 2 | **Pairs of nodes in biological networks are not independent. a** | A bipartite graph with interactions connecting nodes from set A to set B. Each node contains a vector of features, and existing (interaction) and non-existing (no interaction) edges are encoded by concatenating the features of the node pairs that define them. The resulting feature matrix and labels are usable by standard machine learning (ML) algorithms. To estimate model performance, edges are randomly split into training and test sets. By chance, one edge from node A_1 is in the training set and the other in the test set. These samples are dependent and thus allow information to leak across the train–test divide, inflating performance. This problem (pitfall 2) has affected various areas in genomics, where sets A and B could be proteins and ligands, drugs and target genes or enhancers and promoters. **b** | Performance inflation on graph-based datasets grows with the number of edges. We demonstrate this phenomenon by simulating many realizations of random bipartite graphs with power law degree distributions (notebook A). Each graph is encoded as in panel **a** using random node feature vectors so that there is no relationship between features and edge presence. Models that assess area under the precision–recall (auPR) curve using blocking perform similarly to the baseline model that guesses randomly. This is an accurate assessment of performance, because there is no relationship between features and the outcome in the data. Without blocking, the random forest and logistic regression classifiers learn which nodes have a large number of edges in the training set, and make stronger predictions when the features of those nodes are present for edges in the test set. As a consequence, auPR values for unblocked cross-validation are falsely inflated; the outcome is random and not associated with the features, but the model can make accurate predictions on the test set owing to dependence between node pairs in the training and test sets. The problem increases with the number of edges, because the probability of edges with shared nodes crossing the train–test divide grows.

Failing to account for dependencies between examples can lead to biased models and overly optimistic estimates of model performance. Randomized cross-validation (BOX 1) does not protect against this problem and overestimates performance because examples in the test set can be correlated with training examples and bring information into the test set that should not be there. For instance, a model that predicts protein–protein interactions would be likely to perform better in cross-validation than on novel proteins, because proteins with more than one interaction can appear in both the training and test sets of each fold. The scale of this problem increases with the level of dependence; it is worse in highly connected graphs and graphs with hub nodes where area under the precision–recall (auPR) curve can be elevated by more than 0.5 (FIG. 2b, notebook A).

To check for this pitfall, we recommend explicitly considering the underlying dependencies in your data before applying ML tools. One intuitive way of doing this is to visualize the dependencies as a graph in which nodes represent biological entities (for example, genes, proteins, regulatory elements or chemicals) and edges represent associations or interactions between nodes. Cytoscape, R and Python all have tools for rendering tabular data as a graph and computing summary statistics. Edges can be binary (presence or absence of relationship) or quantitative (strength of association). Genomic proximity, protein complexes, transcriptional networks and metabolic pathways are all examples of biological phenomena that generate edges. Nodes with many edges (high degree) create groups of correlated nodes, which are common in genome biology. Examples of such hub nodes are a promoter interacting with many enhancers, a transcription factor regulating many genes and a protein involved in many different complexes. For this pitfall, the key point is that directly — and even indirectly — connected nodes are dependent, as are edges that share a node.

Several approaches exist for mitigating the effects of dependent examples on ML models, and these are not mutually exclusive. The best solution may be to acknowledge dependence and mitigate overfitting at the model evaluation stage. Group *k*-fold cross-validation (BOX 1), also known as blocking[51], is a bulwark against non-independent examples crossing the train–test divide. It does not reduce dependence and is not a universal solution; some dependence structures are too complex to address with blocking. But it does prevent inflated performance metrics caused by dependent examples in the training and test sets, bringing performance closer to what would be expected on an independent prediction set (notebook A). As another approach, it is tempting to directly reduce dependence by downsampling edges (for example, only including one edge per high-degree node) or downweighting nodes with high degree (for example, using node propensity scores). However, these strategies can make the modified data biologically unrealistic, for example, by removing highly interactive proteins or genomic regions. A third alternative is to use methods that explicitly model the covariance between examples, such as mixed effects models from biostatistics[52], time-series

Positive
Positives are examples with the outcome of interest in a binary classifier.

Negative
Negatives are examples with the alternative outcome in a binary classifier. In genomics, negatives often outnumber positives.

models and autocorrelation models from spatial statistics[53]. Unfortunately, these models may not scale to large genomic datasets and are rarely implemented by common ML toolkits, although progress is being made in this area[54,55]. Finally, if specific features are found to be responsible for creating dependent examples, it may be possible to reformulate the problem to have less dependence.

Case study. To illustrate these points, we review an example from our own work on predicting 3D enhancer–promoter interactions using features derived from functional genomics measurements[56]. The examples are enhancer–promoter pairs, which are labelled as physically interacting or not, using Hi-C data. Promoters that interact with many enhancers are over-represented in the positive class and under-represented in the negative class. Applying randomized (*k*-fold) cross-validation produces overly optimistic performance measurements owing to the same promoter appearing in enhancer–promoter pairs in both the training and test sets[47,48]. Shuffled measurements result in a model with equivalent performance. We reformulated the problem to make pairs of genomic windows the examples rather than enhancer–promoter pairs, which dramatically reduces hub nodes in the dataset. This plus blocking by chromosome in cross-validation dramatically reduced dependence and produced a generalizable model[49]. In this case, the important features learned with randomized versus group cross-validation were similar, demonstrating that a ML model can be useful even when performance evaluations are incorrect. Nonetheless, group cross-validation is preferred.

Other examples. The interconnected and, at times, redundant nature of biological systems means that dependencies can be found in most prediction problems. Two genomic loci being in linkage disequilibrium creates a dependency between variants at those positions, requiring the development of methods that can disambiguate these effects for the analysis of GWAS data[57]. When using functional activity measurements from several cell types or tissues, genomic loci themselves are dependent across samples because the underlying functional activity is generally shared[58]. The family of a protein is predictive of its function[59], and so measuring true generalization performance of a computational method may require ensuring that a protein and its entire family fall on the same side of the train–test split.

Pitfall 3: confounding

One of the hardest pitfalls to diagnose involves data in which an unmeasured or artefactual variable ('confounder') creates or masks associations with an outcome. This occurs because the confounder induces dependence between features and the outcome (FIG. 1c). The error is that the confounder is not measured or not thought to be important and is therefore not included in the model. This may have little or no effect on the accuracy of predictions, but it leads to incorrect interpretations of the learned feature–outcome relationships and poor performance when the model is applied in a new context in which the confounder is absent or is distributed differently than in the original context.

The lack of easily interpretable cues in genomics data means that confounders are difficult to identify. In image analysis, where confounders are in front of our eyes, they can still be hard to recognize. Examples include background scenery confounding prediction of types of animals[60] and radiology scanner type confounding prediction of hip fractures[42]. Confounding in genetic studies can arise from unmodelled environmental factors and population structure[8], as well as other factors. An example from genomics involves predicting 3D chromatin interaction data (for example, Hi-C) from epigenetic features. Pairs of loci close to each other along the linear genome have similar epigenetic marks and also interact frequently in 3D owing to polymer physics, making genomic distance a confounder. Another common example is inadvertently confounding the data by sampling or processing samples with different outcomes (for example, diseased versus healthy or different treatment groups) in different batches. When integrating data from multiple studies, differences in genomic assays or bioinformatics pipelines across conditions or cell types also creates confounding. Unfortunately, this means that some degree of confounding can potentially be present in any genomic dataset.

The main problem with this pitfall is that ML models will estimate an association between the outcome and features that depends on the confounder, but the modeller will wrongly interpret this as a direct biological effect. The issue is that we are unaware of the confounder's effect on the observed data, and we do not include it in the ML model. Importantly, cross-validation does not protect against confounded effects, because the confounding is present in both the training and test sets. Confounded associations often come to light when the fitted model makes inaccurate predictions in a new context where the confounder is absent or has a different relationship with the measured features and outcomes. For instance, if ancestry confounds genotype–phenotype relationships in a training cohort, then the fitted ML model will perform poorly when applied to a cohort in which ancestry is randomized. In addition to creating false associations, confounding can mask a real relationship. The variability introduced by the confounder makes it hard to learn true relationships between features and an outcome.

Several statistical approaches help to prevent these problems. Ideally, examples should be randomized with respect to potential confounders, such as experimental batches. When this is not possible, one solution is to use principal components, probabilistic estimation of expression residuals (PEER)[61] or other statistics that summarize structure in high-dimensional data to capture unmeasured confounders[62,63]. One may also try to measure potentially confounding variables. In both cases, including the variable in the ML model will adjust for its effect and reduce confounding[64]. In the example of predicting 3D chromatin interactions from epigenetic features, including genomic distance in the model clarifies whether epigenetic marks are more correlated with 3D interaction than expected by chance. Many regression

Collider
A variable causally influenced by two variables, for example, both a feature and the outcome in predictive modelling.

models are amenable to always including a variable and thereby adjusting for it. But ML approaches that choose features randomly make it hard to force the confounder into the model. To solve this problem, adversarial strategies (mentioned in pitfall 1) for supervised and unsupervised ML models are being developed[65]. Alternatively, a measured variable can be used as a baseline predictor with which models with other features can be compared. It is important to note that adding a variable to a model to reduce confounding can induce bias when it acts as a collider.

Case study. A common confounding variable in genomics experiments is the number of reads sequenced. The outputs from genomic experiments often take the form of signal tracks, where each position in the genome is assigned the number of aligned reads or a processed version thereof. Deeper sequencing means higher average signal. This effect is clear in data from a publicly available ChIP–seq experiment profiling histone H3 lysine 27 acetylation (H3K27ac), which we analysed using random subsets of reads to simulate a range of sequencing depths (notebook B). The effect of sequencing depth on peak height is nonlinear, with some peaks showing more pronounced differences than others (FIG. 3a) and a larger genome-wide effect at lower sequencing depth (FIG. 3b). Applying a ML model trained on data with one sequencing depth to a prediction context with a different depth will result in systematic misprediction of signal values.

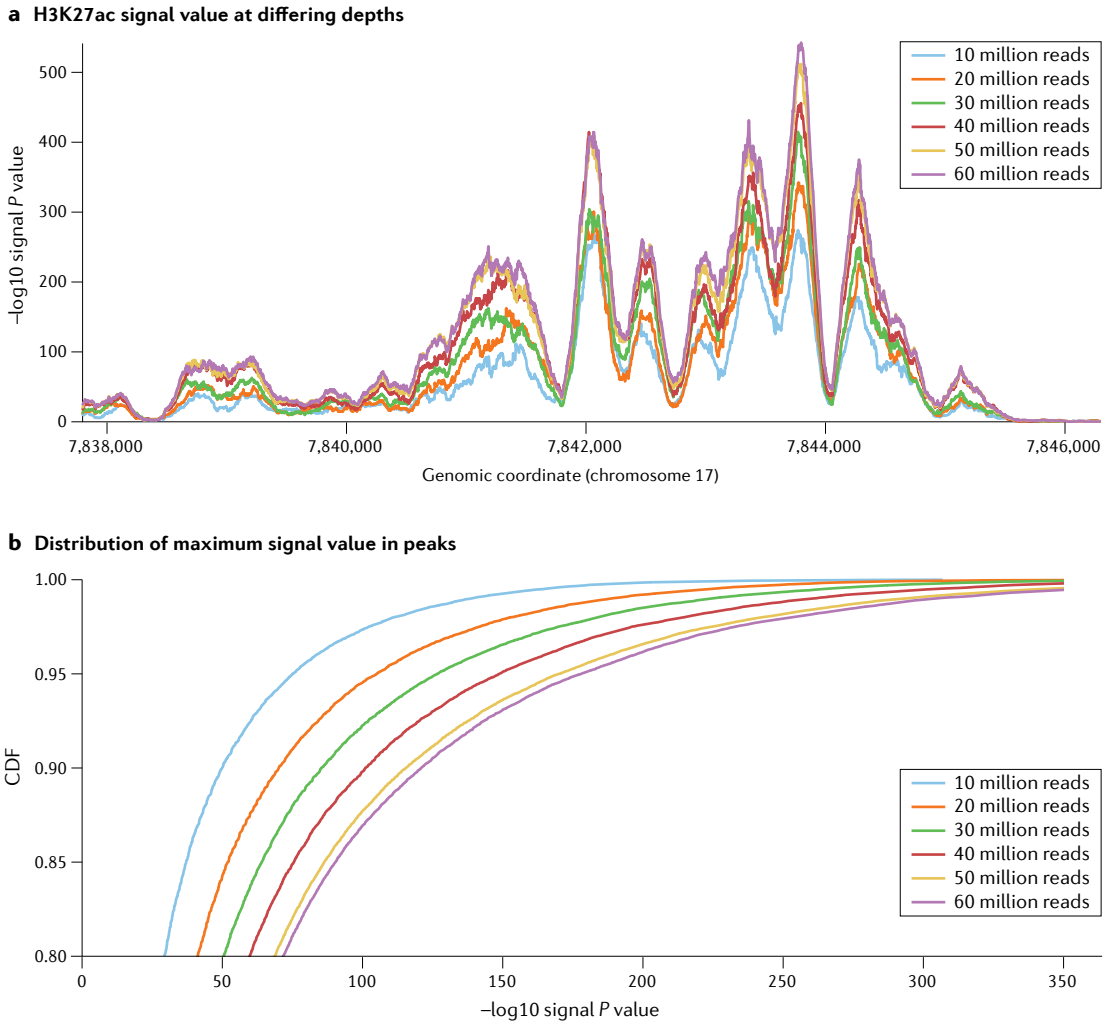


Fig. 3 | **Sequencing depth as a confounding variable.** An experiment profiling histone H3 lysine 27 acetylation (H3K27ac) in smooth muscle cells (ENCODE accession: ENCSR210ZPC) reprocessed at six read depths (notebook B). **a** | The −log10 *P* value signal at one locus on chromosome 17. Peaks are higher as sequencing depth increases, making depth a confounder if not accounted for in machine learning modelling (pitfall 3). This trend is greater in some of the peaks than in others, indicating that the confounding effect is nonlinear. **b** | A cumulative distribution function (CDF) of the maximum signal value (−log10 *P* value) in each peak across all peaks in the genome. The CDFs show what percentage of peaks have a −log10 *P* value less than or equal to every threshold (going from least to most significant from left to right). The same set of genome-wide peaks, those called using 10 million reads, is used for all sequencing depths. The CDF curves for low read depths being above those for high read depths means that fewer peaks would be called significant at any given *P*-value threshold. Machine learning models that aim to predict peaks or learn features enriched in significant peaks would be biased when applied to data with a different sequencing depth if they did not account for the confounding.

Clustering
Unsupervised learning, where there is no measured outcome, although the cluster assignment is an estimate of an unobserved label. The goal is to organize examples on the basis of pairwise similarities of their features, for example, into groups ('clusters') or a hierarchical tree.

Other examples. Confounding arises in GWAS when population genetic structure is systematically different between cases and controls. This problem has arisen in studies of many phenotypes. One notable example is a ML model trained to predict autism spectrum disorder (ASD) using genotyping data, which initially appeared successful[66]. However, the results were generally irreproducible after accounting for population structure[67] and further analysis suggested that the variants were more indicative of shared ancestry than they were of ASD[68,69].

Confounding is also frequently introduced through the data collection process. Technical artefacts are a common source of confounding in genomics, especially when datasets from different sources are integrated into one model. For example, co-expression networks are a powerful tool for predicting gene functions or disease associations. But batch effects induce spurious correlations between sets of genes that confound these predictions[63]. Unmeasured population structure can be associated with both genotypes and gene expression, making it a common confounder in models used to identify expression quantitative trait loci (eQTLs)[62]. Technical artefacts can operate in a similar way in eQTL studies. As an additional example, the data derived from electronic health records (EHRs) can be confounded by socioeconomic status, because poorer patients may be more likely to be seen at clinics where the authors of the EHRs are less experienced[70], as well as by severity of disease and ability to access health care[71].

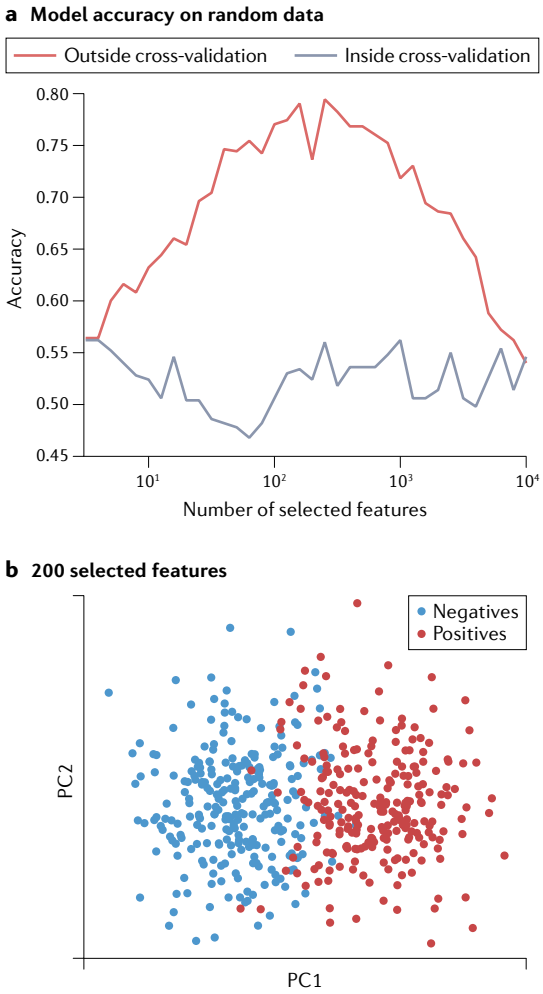


Fig. 4 | **Performing feature selection outside cross-validation yields unrealistically high model accuracy.** We demonstrate the consequences of improperly using supervised feature selection approaches (pitfall 4) on a synthetic dataset and label vector composed entirely of random values (notebook C). **a** | Because the data have no real signal by design, a random forest classifier evaluated with feature selection carried out properly does not perform significantly above random chance (grey); however, when feature selection is erroneously performed before cross-validation on all the folds together (red), the model exhibits a striking increase and subsequent decrease in performance as the number of features increases. This problematic behaviour occurs with any machine learning (ML) model. **b** | A principal component analysis (PCA) projection of the same dataset after 200 features are chosen using supervised feature selection, coloured by class label (blue = negatives, red = positives). Despite the lack of real differences between the two classes, the feature selection procedure corrupted the subsequent unsupervised PCA approach. Outside the scope of ML analyses, preprocessing and visualizing the entire dataset is not necessarily problematic. But if one subsequently chooses to fit an ML model, preprocessing should be redone using only the training set.

Pitfall 4: leaky preprocessing

A subtle yet pervasive problem in ML analyses is data processing that inadvertently causes information to leak from the test set into the training set. Information leakage ('double dipping') occurs when the training set is processed in a manner that depends on data from the test set, which induces dependence between examples (a special case of pitfall 2) and interferes with the utility of the test set for evaluating model performance (FIG. 1d).

Leaky preprocessing is pervasive in genomics. Any data transformation that looks at multiple examples together can be problematic. Specific methods include standardization and principal component analysis (PCA), plus various other scalings and unsupervised embedding approaches. Supervised feature selection, which involves filtering features that are based on association with the outcome, is another form of preprocessing that can cause leakage when performed outside cross-validation. More broadly, recent work on post-selection inference highlights the problem of performing statistical analyses such as differential expression after clustering[72], even if the clusters were defined on independent datasets[73].

The consequence of preprocessing the whole dataset is an overly optimistic cross-validation performance estimate (notebooks C and D) that is easily interpreted as indicating a true biological relationship. This can happen even when there is no association in the data (FIG. 4a, notebook C). Information leakage is also a problem for unsupervised ML methods, such as clustering or visualization techniques. For instance, examples with similar outcome values may incorrectly group together if supervised feature selection is applied to data before they are explored with unsupervised ML (FIG. 4b). More generally, any downstream analysis will be impacted by information leakage.

False negatives
Positives whose labels are incorrectly predicted as negative.

Most ML toolkits enable leakage-free application of supervised and unsupervised transformations by learning parameters from the training set only, then applying the transformation to the training and test sets independently. For example, standardization involves subtracting the mean from a variable and dividing by its standard deviation. The mean and standard deviation are parameters that can be learned from and applied to the training set, and those same parameters can be re-used to transform the test set. This is in contrast to learning the mean and standard deviation from the entire dataset, before splitting into training and test sets. Fortunately for users, this process is already included in scikit-learn's pipeline and transformer API[74], as well as the preProcess parameter of the caret package's train() function[75].

Case study. In ML analysis of DNA methylation assays, researchers are faced with more features (tens of thousands of probes) than examples (typically 100 or fewer samples). To build a robust ML model of an outcome across samples using probes as features it is essential to perform feature selection. If this is performed on the entire dataset before cross-validation, leakage will occur. Identifying probes that exhibit high univariate predictive power across the entire dataset is one way that information is leaked[76–80]. Another way is through identifying 'differentially methylated probes/positions' (DMPs), or aggregates of DMPs called 'differentially methylated regions' (DMRs)[81–85], where only probes with large differences across outcome labels (typically 0.2) are used. Information is leaked because the labels are explicitly used to define how differential each probe is. When studies find high predictive accuracy using DMPs or DMRs defined on the whole dataset, it is unclear whether this signal is real biology or a consequence of train–test leakage. Unfortunately, many overviews of methylation analysis and tools that implement methylation data processing do not explicitly consider downstream ML analysis[86]. We note that identification of DMPs and/or DMRs on an entire dataset is reasonable when these are the final result; the pitfall arises when they are subsequently used as features to build a predictive model[87].

Other examples. Leaky preprocessing was so prevalent in microarray analysis that the journal *Bioinformatics* released an editorial that encouraged scientists to be more critical of their own papers[88]. Pulini et al.[89] refer to this pitfall as 'circular analysis' and point out that it affects a large number of surveyed studies that classify cases of attention deficit hyperactivity disorder (ADHD). Other examples include prediction of clinical or cellular outcomes from functional MRI data[90], gene expression[91,92], metabolomics[93] or DNA methylation[76–85]. Identifying motifs that predict gene expression is another setting in which performing feature selection outside cross-validation has been a problem. Here, feature selection refers to identifying motifs that are enriched in the promoters of co-expressed genes, and leakage occurs when this is performed prior to random assignment of genes to training and test sets in cross-validation[94]. Leakage also occurs when evolutionary profiles are used to predict protein structures or other properties. The problem is that profile parameters are chosen using all proteins before cross-validation[95].

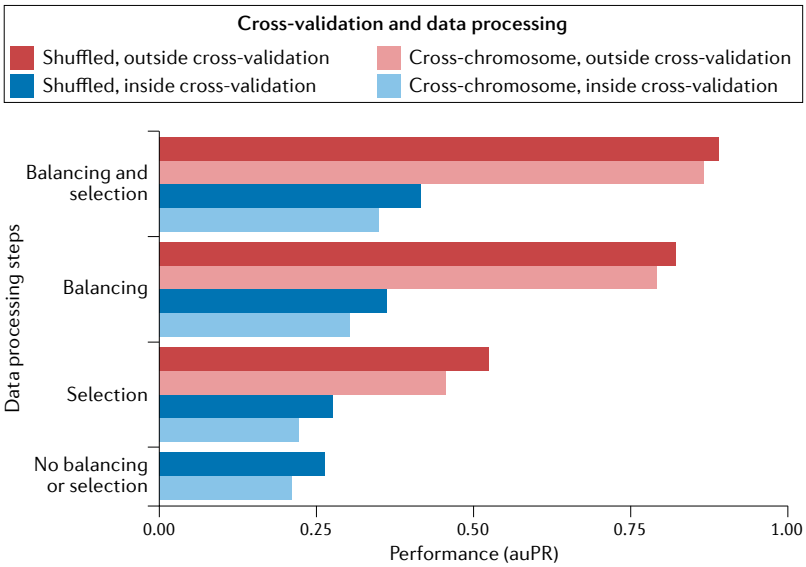


Fig. 5 | **Balancing classes inflates performance when applied outside cross-validation.** Predicting enhancers genome-wide (notebook D) is an example of a very unbalanced classification problem, because most genomic windows do not contain an active enhancer (pitfall 5). Making the positive (enhancer) and negative (not enhancer) classes equal in size ('balancing') can be helpful in the training step of cross-validation. But balancing should be performed inside cross-validation (blue) so that the test set used to evaluate performance reflects the actual imbalance in the genome. Failing to do so inflates performance (red). (In the graph, performance is quantified according to the area under the precision–recall (auPR) curve.) Similarly, preprocessing to select a subset of features ('selection') will inflate performance when performed outside cross-validation owing to leakage (pitfall 4), and performing both balancing and selection outside cross-validation leads to the most inflated performance (red versus blue). These trends hold for two types of cross-validation (shuffled (dark shades) versus cross-chromosome (light shades)).

Pitfall 5: unbalanced classes

A supervised learning task is balanced if examples are evenly distributed across values of the outcome and unbalanced otherwise. Few real datasets are perfectly balanced, and some problems in genomics exhibit extreme imbalance (FIG. 1e). For example, when applying ML to millions of genomic windows to predict whether a given window contains an enhancer, windows with validated examples (positives) may constitute ~1% of the total (FIG. 5). In the context of predicting patient disease risk, Khalilia et al.[96] reported disease prevalence ranging from 0.01% to 29%. A study predicting deleterious non-coding variants used 400 positives with 14 million negatives for Mendelian disease and 2,000 positives with 1.4 million negatives for complex disease[97].

In these scenarios, models are at risk of over-learning the majority class and under-learning the minority class. This is particularly problematic when the minority class is of primary interest and false negatives can be high cost, such as in detecting disease from medical scans or predicting side effects of drug combinations. Importantly, unbalanced data also affect the performance of regression tasks in which labels are continuous values rather than discrete classes (notebook E).

Class imbalance is addressed with a range of strategies. Within the modelling procedure, one can put prior

probability distributions on the classes. More commonly, researchers use rebalancing approaches to increase performance on the minority class[98–102]. There are three basic strategies: oversampling the minority class, undersampling the majority class and weighting examples. Each of these approaches emphasizes the importance to the model of the minority class. Oversampling either duplicates existing data (sampling with replacement)[103] or synthesizes 'plausible' examples of the minority class using interpolation of existing examples (for example, SMOTE[104] or ADASYN[105]). Conversely, undersampling takes a random subset of the majority class, matching the size of the minority class. Weighting examples involves using the inverse of the class proportion so that the ML model gives more emphasis to errors on the minority class. Each approach makes different trade-offs: oversampling retains all data but increases computation time, undersampling decreases computation time but discards some data and sample weighting retains all data but requires determination of the optimal weights. The imbalanced-learn package[106] for scikit-learn[74], as well as the weights argument for train and the sampling argument for trainControl in caret[75], provide practitioners in Python and R with off-the-shelf methods for handling class imbalance.

All three approaches tend to decrease accuracy for the majority class while increasing accuracy for the minority class, which is often a desirable trade-off. However, there are two important considerations to keep in mind. First, balancing should always be performed only within the training fold, so that the fitted model is evaluated against the distribution of classes expected in the prediction set (FIG. 5, notebook D). Second, rebalancing the classes will cause the estimator to become uncalibrated in the sense that the distribution of probabilities predicted will more closely match the balanced training set instead of the unbalanced test set. This is not necessarily a problem when the ranking of examples is more important than the predicted probability itself, and estimators can be recalibrated using a post-processing step, but the issue should be kept in mind.

The performance metrics used to evaluate ML models are differentially affected by imbalance. As a classic example, 99% accuracy is achieved on a training set with 100 positives and 9,900 negatives by always predicting the negative class. Traditional alternatives are auROC and auPR curves (BOX 1). Both quantify recall (also known as power and true positive rate) as the number of positive predictions changes. auROC does so as a function of false positive rate (FPR). In many genomics problems, high recall can be achieved at a very low FPR owing to the large number of negatives in the test set[107], making it easy to obtain a high auROC even when false positives vastly outnumber true positives (that is, high false discovery rate). Alternatively, auPR compares recall with precision, which is one minus the false discovery rate and does not depend on the number of negatives. When false positives dominate true positives, auPR will be low. Thus, for imbalanced problems where the positive minority class is of primary interest, auPR is generally preferred (notebook D). No performance metric is universally best, and the analyst should carefully evaluate options for their setting.

True positives
Positives whose labels are correctly predicted.

False positives
Negatives whose labels are incorrectly predicted as positive.

Class imbalance can also occur in the regression setting, although this is discussed less frequently. For instance, when a model makes predictions of multiple outputs for a single example (the multi-task setting), the distribution of values in these outputs can influence both training and evaluation. As a hypothetical example, if one task has values entirely between 1,000 and 10,000 and the other task has values between 0.01 and 0.1, then evaluating the model simply using mean squared error (MSE) across the two tasks will largely ignore the second task. A distributional difference of that magnitude may be unlikely in genomics, but varying patterns of sparsity are common across genomic assays. For example, some RNA sequencing (RNA-seq) assays exhibit signal primarily at the ends of transcribed genes. Evaluation of a model that tries to predict those assays alongside ChIP–seq assays, where signal is exhibited over a much larger portion of the genome, may be biased (notebook E). Fortunately, a simple workaround in the regression setting is to evaluate each output individually, or as part of a group of outputs with similar distributions (for example, RNA-seq separately from ChIP–seq) instead of aggregating overall outputs regardless of distribution.

Case study. ML models are used to predict protein functions from heterogeneous features such as expression, protein domains, orthology patterns and protein–protein interactions. This is a very unbalanced problem for most annotation labels with fewer than 100 positives (for example, proteins with a given GO term) and thousands of negatives (the rest of the proteins). Hence, a model that always predicts the negative class will have high accuracy. Furthermore, the set of predicted positives at a fixed FPR may actually contain mostly negatives (false positives). At FPR = 0.05, for example, we expect only 5% of the negatives to be in the predictions, but 5% of a large number can easily exceed the number of true positives. For these reasons, a critical assessment of ML methods for mouse protein function prediction used precision at fixed recall in addition to auROC[108]. Reporting auPR or applying rebalancing methods are other strategies that are useful for this problem.

Other examples. In genomics, imbalance is the default state. Indeed, many problems are so imbalanced that researchers have to use statistical approaches simply to ensure that the small number of positive examples that they find are real. For instance, selecting conservative significance thresholds in alignment queries, GWAS projects[109] and motif scanning[110] is crucial for properly controlling the number of false positives because the number of true positives is very low compared with the size of the genome. Given the size of the human genome, problems involving non-coding variants are often highly imbalanced. Examples include prediction of chromatin states, gene expression and disease status from sequence. In the last setting, researchers have developed a balancing algorithm that oversamples the negative class and undersamples the majority class[97]. A common strategy for training models to predict functional peaks from ChIP–seq or chromatin accessibility assays is to use all the peaks and an equal number of

negative regions[29], effectively undersampling the majority class. Databases for drug–target interactions generally do not contain negative interactions, and so one must construct their own negative sets during the training of predictive models[70]. This pitfall focuses on classification but similar problems arise when predicting a quantitative outcome with regression models. Performance can be poor in areas of the quantitative signal where data are sparse, such as genomic regions or genes with low read counts in single-cell genomics assays.

Conclusions

ML shows great potential in genomics, but applying it effectively can be challenging owing to the inherent complexities of biological systems. In this Review, we use general concepts, examples, case studies and computational notebooks to illustrate five common pitfalls that reduce the value of supervised ML. An unfortunate reality is that these mistakes are often easy to make yet subtle enough that we may not realize we are making them, as we have learned in our own work[47–49,56,58]. Although guaranteeing that one has avoided these pitfalls is difficult, we have found that the best guard is to be constructively sceptical and do a thorough inspection of the results to ensure that they make sense.

We have emphasized that the pitfalls can occur independently but are interconnected. Indeed, several of them involve failure to properly evaluate a model in the setting where it will be used in practice. An alternative way to view pitfalls 1–4 is through the lens of artefactual variables that induce relationships in the data that we fail to account for in the model. Confounding (pitfall 3) is a specific example of the dependence structures and graphs discussed in pitfall 2: confounder variables are nodes that create an indirect path between a feature and the outcome, thereby changing the feature–outcome relationship[111]. Leaky preprocessing (pitfall 4) also relates to dependence, in this case between examples in the training and test sets. Leakage can also be thought of as test data confounding the relationship between features and outcome in the training data. The distributional differences from pitfall 1 arise when variables that differ systematically between the training and prediction sets (for example, batch or ancestry) are not modelled. This relates to pitfall 3, because confounder variables shift the distribution of the observed data away from the distribution that we think we are modelling. The difference is that pitfall 3 is about the training data, whereas pitfall 1 focuses on when feature–outcome relationships are different between the prediction and training sets.

Although this unified view is abstract, it explains why the remedy is the same in each case: construct your training, test and prediction sets such that the relationship between the training and test sets is the same as the relationship between the training and the prediction set. For example, if a model is trained on in vitro data but the prediction setting is in vivo, then the test set should also be in vivo; avoid training and testing on data arising from a mixture of all existing batches if the prediction setting is a new batch; avoid splitting dependent examples into your training and test sets if the prediction set will not include similarly dependent examples; avoid performing preprocessing steps on your training and test data together because you cannot carry them out at the same time on your prediction set.

Performing research is rarely a straightforward process, and sometimes the winding path one takes can lead to a pitfall. For instance, given gene expression values from samples under two conditions, one may initially be interested in identifying differentially expressed genes. Here, it is appropriate to analyse the entire dataset. However, if later on, the researcher decides to add a ML analysis using differentially expressed genes as features, they would have inadvertently fallen into pitfall 4. Hence, we recommend considering the entire path your data has taken before applying ML.

Beyond these five pitfalls, there are other considerations to keep in mind for the most effective application of ML. Among these is using appropriate baselines to ensure that your model has not learned simple rules that are not biologically interesting. We discussed predicting chromatin interactions from genomic distance as an example of a baseline model, and FIG. 2 uses random guessing as a baseline. Another issue not directly covered in our pitfalls is using performance on test folds to tune model parameters or perform model selection, which leads to elevated performance estimates. A further consideration, touched on in pitfall 5, is using informative performance measures. Global performance measures (for example, auROC or correlation across the entire test set) are informative but frequently obscure interesting details that can be captured by finer-grained measures, such as breaking performance down by demographics or genomic annotations. When it is unclear what the best fine-grained measures are beforehand, we recommend looking closely at examples that are poorly predicted.

In this Review, we primarily illustrate the direct effects of ML pitfalls on model performance. However, ML is often employed for gaining biological insights rather than prediction per se. In these cases, one will generally interpret a trained model, after validating that it performs well, to extract the relationships it has learned. Unfortunately, models that exhibit good performance may have learned meaningless relationships if one or more pitfalls has influenced the analysis. Our suggestions for checking the pitfalls are important even for model interpretation. However, in some cases an overfitted or biased model can produce accurate biological conclusions; insufficient for generalizable predictions but sufficient for attribution[112]. For example, the model may overestimate the strength of association between a feature and outcome owing to confounding or dependence. When this is a true biological relationship, an effect size error has been made but the association is not a false discovery.

We as modellers and reviewers are the key to ensuring effective use of ML in genomics. Before accepting ML outputs at face value, we should become familiar with the data, apply healthy scepticism and conduct robust follow-up analyses[113]. This is easier said than done, but the trustworthiness of ML in biomedical research depends on these strategies.

Published online: 26 November 2021

1. Teschendorff, A. E. Avoiding common pitfalls in machine learning omic data science. *Nat. Mater.* **18**, 422–427 (2019).
This Comment article talks about cross-validation and independent test sets as solutions to two pitfalls encountered when applying supervised ML in genomics: the 'curse of dimensionality' and confounding.
2. Minhas, F., Asif, A. & Ben-Hur, A. Ten ways to fool the masses with machine learning. Preprint at *arXiv* https://arxiv.org/abs/1901.01686 (2019).
3. Eraslan, G., Avsec, Ž., Gagneur, J. & Theis, F. J. Deep learning: new computational modelling techniques for genomics. *Nat. Rev. Genet.* **20**, 389–403 (2019).
4. Ching, T. et al. Opportunities and obstacles for deep learning in biology and medicine. *J. R. Soc. Interface* **15**, 20170387 (2018).
5. Zou, J. et al. A primer on deep learning in genomics. *Nat. Genet.* **51**, 12–18 (2019).
6. Flagel, L., Brandvain, Y. & Schrider, D. R. The unreasonable effectiveness of convolutional neural networks in population genetic inference. *Mol. Biol. Evol.* **36**, 220–238 (2019).
7. Liu, J., Lewinger, J. P., Gilliland, F. D., Gauderman, W. J. & Conti, D. V. Confounding and heterogeneity in genetic association studies with admixed populations. *Am. J. Epidemiol.* **177**, 351–360 (2013).
8. Vilhjálmsson, B. J. & Nordborg, M. The nature of confounding in genome-wide association studies. *Nat. Rev. Genet.* **14**, 1–2 (2013).
9. Hellwege, J. N. et al. Population stratification in genetic association studies. *Curr. Protoc. Hum. Genet.* **95**, 1.22.1–1.22.23 (2017).
10. Sul, J. H., Martin, L. S. & Eskin, E. Population structure in genetic studies: confounding factors and mixed models. *PLoS Genet.* **14**, e1007309 (2018).
11. Weirauch, M. T. et al. Evaluation of methods for modeling transcription factor sequence specificity. *Nat. Biotechnol.* **31**, 126–134 (2013).
12. Leek, J. T. et al. Tackling the widespread and critical impact of batch effects in high-throughput data. *Nat. Rev. Genet.* **11**, 733–739 (2010).
This Review documents the prevalence of batch effects in genomic data and shows how these can confound statistical inferences.
13. Tran, H. T. N. et al. A benchmark of batch-effect correction methods for single-cell RNA sequencing data. *Genome Biol.* **21**, 12 (2020).
14. Rabanser, S., Günnemann, S. & Lipton, Z. Failing loudly: an empirical study of methods for detecting dataset shift. in *Advances in Neural Information Processing Systems* (NeurIPS 2019) (eds Wallach, H. et al.) Vol. 32, 1396–1408 (Curran Associates, Inc., 2019).
15. Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B. & Smola, A. A kernel two-sample test. *J. Mach. Learn. Res.* **13**, 723–773 (2012).
16. Ren, J. et al. in *Advances in Neural Information Processing Systems* (NeurIPS 2019) (eds Wallach, H. et al.) Vol. 32, 14707–14718 (Curran Associates, Inc., 2019).
17. Kingma, D. P. & Welling, M. Auto-encoding variational Bayes. Preprint at *arXiv* https://arxiv.org/abs/1312.6114# (2013).
18. Liu, F. T., Ting, K. M. & Zhou, Z. in *IEEE International Conference on Data Mining* 413–422 (IEEE, 2008).
19. Johnson, W. E., Li, C. & Rabinovic, A. Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics* **8**, 118–127 (2007).
20. Leek, J. T. & Storey, J. D. Capturing heterogeneity in gene expression studies by surrogate variable analysis. *PLoS Genet.* **3**, 1724–1735 (2007).
21. Butler, A., Hoffman, P., Smibert, P., Papalexi, E. & Satija, R. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nat. Biotechnol.* **36**, 411–420 (2018).
22. Stuart, T. et al. Comprehensive integration of single-cell data. *Cell* **177**, 1888–1902.e21 (2019).
23. Wang, T. et al. BERMUDA: a novel deep transfer learning method for single-cell RNA sequencing batch correction reveals hidden high-resolution cellular subtypes. *Genome Biol.* **20**, 165 (2019).
24. Pan, S. J. & Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **22**, 1345–1359 (2010).
25. Kouw, W. M. & Loog, M. A review of domain adaptation without target labels. *IEEE Trans. Pattern Anal. Mach. Intell.* **43**, 766–785 (2019).
26. Shimodaira, H. Improving predictive inference under covariate shift by weighting the log-likelihood function. *J. Stat. Plan. Inference* **90**, 227–244 (2000).
This paper discusses distributional differences, also known as covariate shift, and proposes several weighting schemes for adjusting for this pitfall.
27. Bickel, S., Brückner, M. & Scheffer, T. Discriminative learning under covariate shift. *J. Mach. Learn. Res.* **10**, 2137–2155 (2009).
28. Orenstein, Y. & Shamir, R. Modeling protein-DNA binding via high-throughput in vitro technologies. *Brief. Funct. Genomics* **16**, 171–180 (2017).
29. Alipanahi, B., Delong, A., Weirauch, M. T. & Frey, B. J. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nat. Biotechnol.* **33**, 831–838 (2015).
30. Berger, M. F. & Bulyk, M. L. Universal protein-binding microarrays for the comprehensive characterization of the DNA-binding specificities of transcription factors. *Nat. Protoc.* **4**, 393–411 (2009).
31. Annala, M., Laurila, K., Lähdesmäki, H. & Nykter, M. A linear model for transcription factor binding affinity prediction in protein binding microarrays. *PLoS ONE* **6**, e20059 (2011).
32. Agius, P., Arvey, A., Chang, W., Noble, W. S. & Leslie, C. High resolution models of transcription factor-DNA affinities improve in vitro and in vivo binding predictions. *PLoS Comput. Biol.* **6**, e1000916 (2010).
33. Riley, T. R., Lazarovici, A., Mann, R. S. & Bussemaker, H. J. Building accurate sequence-to-affinity models from high-throughput in vitro protein-DNA binding data using FeatureREDUCE. *Elife* **4**, e06397 (2015).
34. Wong, K.-C., Li, Y., Peng, C. & Wong, H.-S. A comparison study for DNA motif modeling on protein binding microarray. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **13**, 261–271 (2016).
35. Rastogi, C. et al. Accurate and sensitive quantification of protein-DNA binding affinity. *Proc. Natl Acad. Sci. USA* **115**, E3692–E3701 (2018).
36. Im, J., Park, B. & Han, K. A generative model for constructing nucleic acid sequences binding to a protein. *BMC Genomics* **20**, 967 (2019).
37. Ishida, R. et al. RaptRanker: in silico RNA aptamer selection from HT-SELEX experiment based on local sequence and structure information. *Nucleic Acids Res.* **48**, e82 (2020).
38. Nutiu, R. et al. Direct measurement of DNA affinity landscapes on a high-throughput sequencing instrument. *Nat. Biotechnol.* **29**, 659–664 (2011).
39. Tabb, D. L. et al. Repeatability and reproducibility in proteomic identifications by liquid chromatography-tandem mass spectrometry. *J. Proteome Res.* **9**, 761–776 (2010).
40. Pooch, E. H. P., Ballester, P. L. & Barros, R. C. Can we trust deep learning models diagnosis? The impact of domain shift in chest radiograph classification. Preprint at *arXiv* https://arxiv.org/abs/1909.01940# (2019).
41. Zech, J. R. et al. Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: a cross-sectional study. *PLoS Med.* **15**, e1002683 (2018).
42. Badgeley, M. A. et al. Deep learning predicts hip fracture using confounding patient and healthcare variables. *NPJ Digit. Med.* **2**, 31 (2019).
43. Antun, V., Renna, F., Poon, C., Adcock, B. & Hansen, A. C. On instabilities of deep learning in image reconstruction and the potential costs of AI. *Proc. Natl Acad. Sci. USA* **117**, 30088–30095 (2020).
44. Geis, J. R. et al. Ethics of artificial intelligence in radiology: summary of the joint european and north american multisociety statement. *Radiology* **293**, 436–440 (2019).
45. Larrazabal, A. J., Nieto, N., Peterson, V., Milone, D. H. & Ferrante, E. Gender imbalance in medical imaging datasets produces biased classifiers for computer-aided diagnosis. *Proc. Natl Acad. Sci. USA* **117**, 12592–12594 (2020).
46. Guney, E. in *Biocomputing 2017: Proceedings of the Pacific Symposium* (eds Altmann, R. B. et al.) 132–143 (World Scientific, 2016).
47. Xi, W. & Beer, M. A. Local epigenomic state cannot discriminate interacting and non-interacting enhancer-promoter pairs with high accuracy. *PLoS Comput. Biol.* **14**, e1006625 (2018).
48. Cao, F. & Fullwood, M. J. Inflated performance measures in enhancer-promoter interaction-prediction methods. *Nat. Genet.* **51**, 1196–1198 (2019).
49. Whalen, S. & Pollard, K. S. Reply to 'Inflated performance measures in enhancer-promoter interaction-prediction methods'. *Nat. Genet.* **51**, 1198–1200 (2019).
50. Eid, F.-E. et al. Systematic auditing is essential to debiasing machine learning in biology. *Commun. Biol.* **4**, 183 (2020).
This article proposes a set of data modifications that can be used to identify overestimated performance in supervised ML with paired-input data, such as protein–protein interactions, where examples occur in many pairs.
51. Roberts, D. R. et al. Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. *Ecography* **40**, 913–929 (2017).
This study demonstrates blocking as an effective strategy for estimating the performance of ML models on data with complex dependency structures.
52. Korte, A. et al. A mixed-model approach for genome-wide association studies of correlated traits in structured populations. *Nat. Genet.* **44**, 1066–1071 (2012).
53. Stucki, S. et al. High performance computation of landscape genomic models including local indicators of spatial association. *Mol. Ecol. Resour.* **17**, 1072–1089 (2017).
54. Runcie, D. E. & Crawford, L. Fast and flexible linear mixed models for genome-wide genetics. *PLoS Genet.* **15**, e1007978 (2019).
55. Jiang, L. et al. A resource-efficient tool for mixed model association analysis of large-scale data. *Nat. Genet.* **51**, 1749–1755 (2019).
56. Whalen, S., Truty, R. M. & Pollard, K. S. Enhancer–promoter interactions are encoded by complex genomic signatures on looping chromatin. *Nat. Genet.* **48**, 488–496 (2016).
57. Brzyski, D. et al. Controlling the rate of GWAS false discoveries. *Genetics* **205**, 61–75 (2017).
58. Schreiber, J., Singh, R., Bilmes, J. & Noble, W. S. A pitfall for machine learning methods aiming to predict across cell types. *Genome Biol.* **21**, 282 (2020).
59. Lee, D., Redfern, O. & Orengo, C. Predicting protein function from sequence and structure. *Nat. Rev. Mol. Cell Biol.* **8**, 995–1005 (2007).
60. Ribeiro, M. T., Singh, S. & Guestrin, C. in *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 1135–1144 (Association for Computing Machinery, 2016).
61. Stegle, O., Parts, L., Piipari, M., Winn, J. & Durbin, R. Using probabilistic estimation of expression residuals (PEER) to obtain increased power and interpretability of gene expression analyses. *Nat. Protoc.* **7**, 500–507 (2012).
62. Listgarten, J., Kadie, C., Schadt, E. E. & Heckerman, D. Correction for hidden confounders in the genetic analysis of gene expression. *Proc. Natl Acad. Sci. USA* **107**, 16465–16470 (2010).
63. Parsana, P. et al. Addressing confounding artifacts in reconstruction of gene co-expression networks. *Genome Biol.* **20**, 94 (2019).
64. Dinga, R., Schmaal, L., Brenda, W. J., Veltman, D. J. & Marquand, A. F. Controlling for effects of confounding variables on machine learning predictions. Preprint at *bioRxiv* https://doi.org/10.1101/2020.08.17.255034 (2020).
65. Dincer, A. B., Janizek, J. D. & Lee, S.-I. Adversarial deconfounding autoencoder for learning robust gene expression embeddings. *Bioinformatics* **36**, i573–i582 (2020).
66. Skafidas, E. et al. Predicting the diagnosis of autism spectrum disorder using gene pathway analysis. *Mol. Psychiatry* **19**, 504–510 (2014).
67. Robinson, E. B. et al. Response to 'Predicting the diagnosis of autism spectrum disorder using gene pathway analysis'. *Mol. Psychiatry* **19**, 859–861 (2014).
68. Keys, K. L. et al. On the cross-population generalizability of gene expression prediction models. *PLoS Genet.* **16**, e1008927 (2020).
69. Belgard, T. G., Jankovic, I., Lowe, J. K. & Geschwind, D. H. Population structure confounds autism genetic classifier. *Mol. Psychiatry* **19**, 405–407 (2014).
70. Chen, X. et al. Drug-target interaction prediction: databases, web servers and computational models. *Brief. Bioinform.* **17**, 696–712 (2016).
71. Brookhart, M. A., Stürmer, T., Glynn, R. J., Rassen, J. & Schneeweiss, S. Confounding control in healthcare database research: challenges and potential approaches. *Med. Care* **48**, S114–S120 (2010).
72. Zhang, J. M., Kamath, G. M. & Tse, D. N. Valid post-clustering differential analysis for single-cell RNA-seq. *Cell Syst.* **9**, 383–392.e6 (2019).
73. Gao, L. L., Bien, J. & Witten, D. Selective Inference for hierarchical clustering. Preprint at *arXiv* https://arxiv.org/abs/2012.02936 (2020).
74. Pedregosa, F. et al. Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).
75. Kuhn, M. Building predictive models in R using the caret package. *J. Stat. Softw. Artic.* **28**, 1–26 (2008).

76. Vidaki, A. et al. DNA methylation-based forensic age prediction using artificial neural networks and next generation sequencing. *Forensic Sci. Int. Genet.* **28**, 225–236 (2017).
77. Kimura, R. et al. An epigenetic biomarker for adult high-functioning autism spectrum disorder. *Sci. Rep.* **9**, 13662 (2019).
78. Levy, J. J. et al. MethylNet: an automated and modular deep learning approach for DNA methylation analysis. *BMC Bioinforma.* **21**, 108 (2020).
79. Rauschert, S., Raubenheimer, K., Melton, P. E. & Huang, R. C. Machine learning and clinical epigenetics: a review of challenges for diagnosis and classification. *Clin. Epigenetics* **12**, 51 (2020).
80. Capper, D. et al. DNA methylation-based classification of central nervous system tumours. *Nature* **555**, 469–474 (2018).
81. Bahado-Singh, R. O. et al. Deep learning/artificial intelligence and blood-based dna epigenomic prediction of cerebral palsy. *Int. J. Mol. Sci.* **20**, 2075 (2019).
82. Mohandas, N. et al. Epigenome-wide analysis in newborn blood spots from monozygotic twins discordant for cerebral palsy reveals consistent regional differences in DNA methylation. *Clin. Epigenetics* **10**, 25 (2018).
83. Crowgey, E. L., Marsh, A. G., Robinson, K. G., Yeager, S. K. & Akins, R. E. Epigenetic machine learning: utilizing DNA methylation patterns to predict spastic cerebral palsy. *BMC Bioinforma.* **19**, 225 (2018).
84. Aref-Eshghi, E. et al. Genomic DNA methylation-derived algorithm enables accurate detection of malignant prostate tissues. *Front. Oncol.* **8**, 100 (2018).
85. Luo, R. et al. Identifying CpG methylation signature as a promising biomarker for recurrence and immunotherapy in non-small-cell lung carcinoma. *Aging* **12**, 14649–14676 (2020).
86. Wilhelm-Benartzi, C. S. et al. Review of processing and analysis methods for DNA methylation array data. *Br. J. Cancer* **109**, 1394–1402 (2013).
87. Peters, T. J. et al. De novo identification of differentially methylated regions in the human genome. *Epigenetics Chromatin* **8**, 6 (2015).
88. Rocke, D. M., Ideker, T., Troyanskaya, O., Quackenbush, J. & Dopazo, J. Papers on normalization, variable selection, classification or clustering of microarray data. *Bioinformatics* **25**, 701–702 (2009).
89. Pulini, A. A., Kerr, W. T., Loo, S. K. & Lenartowicz, A. Classification accuracy of neuroimaging biomarkers in attention-deficit/hyperactivity disorder: effects of sample size and circular analysis. *Biol. Psychiatry Cogn. Neurosci. Neuroimaging* **4**, 108–120 (2019).
90. Poldrack, R. A., Huckins, G. & Varoquaux, G. Establishment of best practices for evidence for prediction: a review. *JAMA Psychiatry* **77**, 534–540 (2020).
91. Ambroise, C. & McLachlan, G. J. Selection bias in gene extraction on the basis of microarray gene-expression data. *Proc. Natl Acad. Sci. USA* **99**, 6562–6566 (2002).
The authors present prediction of cancer outcome from expression of a small number of genes as an example of how supervised feature selection performed before cross-validation leads to performance overestimation.
92. van Eyk, C. L. et al. Analysis of 182 cerebral palsy transcriptomes points to dysregulation of trophic signalling pathways and overlap with autism. *Transl. Psychiatry* **8**, 88 (2018).
93. Alakwaa, F. M., Chaudhary, K. & Garmire, L. X. Deep learning accurately predicts estrogen receptor status in breast cancer metabolomics data. *J. Proteome Res.* **17**, 337–347 (2018).
94. Yuan, Y., Guo, L., Shen, L. & Liu, J. S. Predicting gene expression from sequence: a reexamination. *PLoS Comput. Biol.* **3**, e243 (2007).
95. Urban, G., Torrisi, M., Magnan, C. N., Pollastri, G. & Baldi, P. Protein profiles: biases and protocols. *Comput. Struct. Biotechnol. J.* **18**, 2281–2289 (2020).
This study demonstrates how protein profiles cause leakage of information between the training and test sets, and hence performance overestimation, in the context of protein structure prediction.
96. Khalilia, M., Chakraborty, S. & Popescu, M. Predicting disease risks from highly imbalanced data using random forest. *BMC Med. Inform. Decis. Mak.* **11**, 51 (2011).
97. Schubach, M., Re, M., Robinson, P. N. & Valentini, G. Imbalance-aware machine learning for predicting rare and common disease-associated non-coding variants. *Sci. Rep.* **7**, 2959 (2017).
98. Japkowicz, N. & Stephen, S. The class imbalance problem: a systematic study1. *Intell. Data Anal.* **6**, 429–449 (2002).
99. Barandela, R., Sánchez, J. S., Garca, V. & Rangel, E. Strategies for learning in class imbalance problems. *Pattern Recognit.* **36**, 849–851 (2003).
This work explores the negative consequences of imbalanced data as well as several common strategies for mitigating this pitfall.
100. Batista, G. E. A. P. A., Prati, R. C. & Monard, M. C. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.* **6**, 20–29 (2004).
101. Buda, M., Maki, A. & Mazurowski, M. A. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Netw.* **106**, 249–259 (2018).
This article explores performance measures and mitigation strategies for class imbalance specifically in the context of prediction with convolutional neural networks.
102. Cui, Y., Jia, M., Lin, T.-Y., Song, Y. & Belongie, S. Class-balanced loss based on effective number of samples. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2019)
103. Nguyen, H. M., Cooper, E. W. & Kamei, K. Borderline over-sampling for imbalanced data classification. *Int. J. Knowl. Eng. Soft Data Paradig.* **3**, 4 (2011).
104. Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **16**, 321–357 (2002).
105. Haibo H., Yang B., Garcia, E. A. & Shutao L. in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)* 1322–1328 (IEEE,2008).
106. Lemaître, G., Nogueira, F. & Aridas, C. K. Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. *J. Mach. Learn. Res.* **18**, 559–563 (2017).
107. Davis, J. & Goadrich, M. in *Proc. 23rd International Conference on Machine Learning* 233–240 (Association for Computing Machinery, 2006).
108. Peña-Castillo, L. et al. A critical assessment of *Mus musculus* gene function prediction using integrated genomic evidence. *Genome Biol.* **9**, S2 (2008).
109. Kaler, A. S. & Purcell, L. C. Estimation of a significance threshold for genome-wide association studies. *BMC Genomics* **20**, 618 (2019).
110. Grant, C. E., Bailey, T. L. & Noble, W. S. FIMO: scanning for occurrences of a given motif. *Bioinformatics* **27**, 1017–1018 (2011).
111. VanderWeele, T. J. & Shpitser, I. On the definition of a confounder. *Ann. Stat.* **41**, 196–220 (2013).
112. Efron, B. Prediction, estimation, and attribution. *J. Am. Stat. Assoc.* **115**, 636–655 (2020).
113. Yu, B. & Kumbier, K. Veridical data science. *Proc. Natl Acad. Sci. USA* **117**, 3920–3929 (2020).

Acknowledgements
The authors thank P. Baldi, M. Beer, A. Ben-Hur, J. Ernst, E. Eskin, G. Haliburton, H. Huang, S.-I. Lee, M. Libbrecht, J. Majewski, Q. Morris, S. Mostafavi, J.-P. Vert, W. Wang, B. Yu and M. Zitnik for recommending examples and for helpful suggestions on how to review this topic.

Author contributions
S.W. and J.S. researched data for article. All authors substantially contributed to the discussion of content, wrote the article and reviewed and or edited the manuscript before submission.

Competing interests
The authors declare no competing interests.

Peer review information
Nature Reviews Genetics thanks A. Gitter, J. Gagneur and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

Publisher's note
Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

RELATED LINKS
Interactive notebooks of the pitfalls discussed in this Review: https://github.com/shwhalen/ml-pitfalls