CHAPTER

2

Main concepts in machine learning

Sandra Vieira[1], *Walter Hugo Lopez Pinaya*[1,2], *Andrea Mechelli*[1]

[1] **Department of Psychosis Studies, Institute of Psychiatry, Psychology & Neuroscience, King's College London, London, United Kingdom;** [2] **Centre of Mathematics, Computation, and Cognition, Universidade Federal do ABC, Santo André, São Paulo, Brazil**

2.1 Introduction

This chapter introduces the fundamental concepts in machine learning. The aim is to provide the reader with an overview of the machine learning process, before a more in-depth discussion in subsequent chapters. As explained in Chapter 1, we can divide machine learning methods into four categories: supervised, unsupervised, semisupervised, and reinforcement learning. Supervised learning is the most prevalent type of learning in neuroscientific research, including the investigation of brain disorders. Therefore, this chapter will focus on this type of machine learning.

The main steps of the standard supervised learning pipeline in brain disorder studies can be seen in Fig. 2.1. As described in Chapter 1, classical statistics place greater emphasis on the goodness of fit—i.e., the extent to which a statistical model explains the data. On the other hand, machine learning places greater emphasis on predictive ability—i.e., the extent to which a statistical model can infer a characteristic of interest in unseen data. Therefore, the first step is to formulate a well-defined research question that is suitable for machine learning. Second, it is necessary to prepare and explore the data. Just like in classical statistics, getting to know the data well can go a long way when it comes to planning and implementing machine learning models. Next, the prepared raw data undergo several transformations to make it suitable for machine

Machine Learning
https://doi.org/10.1016/B978-0-12-815739-8.00002-X

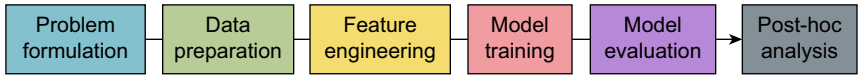

FIGURE 2.1 Main steps of the standard supervised machine learning pipeline in studies of brain disorders.

learning analysis; this procedure is known as *feature engineering*. The resulting data, known as feature set, is then used to train the machine learning models to find relationships between data patterns and the desired outcomes. After training, the overall performance of the model is estimated as a proxy of its generalizability. Lastly, the results of the machine learning model are analyzed. This stage typically involves checking the statistical significance of the model's performance and verifying the most important features driving the model.

In practice, building a successful machine learning model is often an iterative process. It is considered good practice to start with a simple model and then improve its performance by increasing complexity without compromising its generalizability. Comparing different approaches of similar complexity, such as alternative preprocessing or feature engineering methods, is also encouraged.

2.2 Problem formulation

In supervised learning, a well-defined research question requires clearly defined *target variable*, *feature set*, and *task*. The target variable is what the machine learning algorithm will learn to predict (usually referred to as dependent variable in classical statistics); in case of a classification problem, the target variable is commonly known as *labels*. The *feature set* comprises the data that will serve as input to the machine learning algorithm (also known as independent or explanatory variables in classical statistics). The term *feature vector* refers to all features for a single observation (Fig. 2.2A). For example, in the case of structural

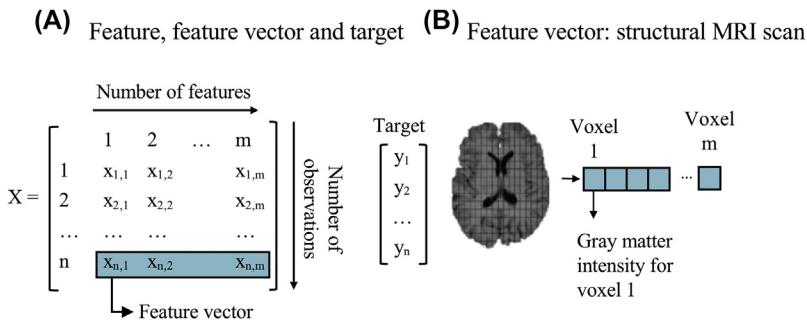

FIGURE 2.2 Terminology in machine learning. (A) A dataset consists of a feature set (X) and target variable (y). The dataset is comprised of one feature vector ($X_{n,1}$, $X_{n,2}$,… $X_{n,m}$), for each observation. (B) Example of a feature vector for a structural magnetic resonance imaging scan. In this case, all voxels in the three-dimensional brain volume are flattened into a one-dimensional vector.

magnetic resonance imaging (MRI), the feature vector will contain the intensity values of all voxels for a neuroanatomical scan (Fig. 2.2B). The total number of features in a dataset is referred to as *dimension* or *dimensionality*. Finally, the task defines what the machine learning algorithm should do with the features and target variable. Putting all these elements together enables one to formulate a succinct problem statement; for example, using functional connectivity data (features) to categorize (task) which patients will and will not benefit from a certain treatment (target).

2.3 Data preparation

Machine learning helps us discover patterns in the data and use these patterns to make predictions about new data. To achieve this, however, it is important to clean, explore, and prepare the data to improve the overall quality of the dataset. This step typically involves using histograms and scatter plots to explore the data and using a range of strategies for minimizing the impact of outliers and missing data (see Chapter 14). Ultimately, a machine learning model is only as good as the data used to develop it.

Also very important is to identify and deal with potential confounding variables. A confounding variable is a variable that influences both the feature set and the target variable. These variables are not directly associated with our formulated problem; however, they may adversely affect the development of the machine learning model, for example, by leading to overoptimistic results. A typical confounding variable in brain disorder studies is the gender of the participants. In a classification problem where we initially want to categorize subjects between healthy controls and patients, if one of the groups is mainly composed by men and the other group by women, the classifier might learn to distinguish the subjects not by the disease mechanisms but by the their gender. For this reason, during the data preparation, we need to verify the presence of any potential source of bias in our dataset using statistical tests (e.g., homogeneity tests) and balance the data (further details about how to deal with confounding variables can be found in Rao, Monteiro, Mourao-Miranda, & Alzheimer's Disease Initiative, 2017).

2.4 Feature engineering

Feature engineering usually refers to transformations performed on the raw data to generate features. In this section, we present some of the most common feature engineering methods used in brain disorders research:

feature extraction, dimensionality reduction, feature selection, and feature scaling/normalizing.

2.4.1 Feature extraction

Feature extraction refers to the transformations applied to the raw data to extract a feature vector that is suitable for machine learning models. The quality and size of the feature set will have great influence on how well the model will perform. For this reason, the extraction of these features can be very time-consuming and requires significant domain expertise. In brain disorder studies, these transformations are very domain-specific, for example:

- The extraction of brain region connectivity from functional MRI scans
- The extraction of morphological characteristic (volume, thickness, curvature) from structural MRI scans
- The extraction of time−frequency distributions from electroencephalogram signals
- The extraction of the number of occurrences of words from the subject's speech
- The extraction of mobility features from smartphone's GPS signals

2.4.2 Dimensionality reduction

Brain disorder studies often deal with high-dimensional data, i.e., the number of features is substantially larger than the number of observations. Unfortunately, when inputting a small number of observations, there is a high risk of overfitting. Overfitting occurs when the model learns details in the data rather than global generalizable properties; this results in a good performance in the data used to build the model but poor performance when applied to new data. The smaller the dataset, the less likely it is for the model to learn more generalizable patterns, and therefore the higher the risk of overfitting.

An optional step to mitigate this issue is to transform the original extracted features into a substantially smaller set. This procedure is known as *dimensionality reduction*. Dimensionality reduction generates a new feature set while trying not to lose much information and keep (or even improve) the model's performance. A smaller feature set will help reduce computational resources, such as storage space and computational time, and remove redundant features if any; and it will also help with data visualization.

Principal component analysis (PCA) and independent component analysis (ICA) are among the most commonly used techniques. Briefly, PCA

uses an orthogonal transformation to convert a set of observations including possibly correlated variables into a smaller set of uncorrelated variables known as principal components (Jolliffe, 2002). ICA can be seen as an extension to PCA: while the latter only uncorrelates the data, ICA works with higher-order statistics to maximize statistical independence of the estimated components. Autoencoders are a further approach that can be used to extract latent features by applying consecutive nonlinear transformations to the data (Vincent, Larochelle, Lajoie, Bengio, & Manzagol, 2010). Autoencoders and PCA will be covered in more detail in Chapters 11 and 12, respectively.

2.4.3 Feature selection

Feature selection refers to the selection of a subset of data based on the premise that the original feature set contains irrelevant or redundant information. Importantly, feature selection differs from reducing data dimensionality. Although both methods seek to reduce the number of features, in dimensionality reduction new features are created based on, and used instead of, the original features; while feature selection methods eliminate some of the original features.

2.4.3.1 Manual feature selection

Manual feature selection is carried out when there is a priori knowledge about how informative certain features are. For example, in neuroimaging studies, one can select a set of regions of interest which are believed to be implicated in a particular disorder or cognitive task. Likewise, in genetic studies, it is possible to preselect certain single-nucleotide polymorphisms based on existing knowledge.

2.4.3.2 Automated feature selection

In automated feature selection, there is no a priori knowledge about which features will or will not contribute to the task, and therefore the optimal set of features is chosen through a statistical method. Automated feature selection methods can be classified into three classes: filter methods, wrapper methods, and embedded methods (see Guyon & Elisseeff, 2003; Mwangi, Tian, & Soares, 2014; Saeys, Inza, & Larranaga, 2007 for a detailed review) (Table 2.1).

2.4.3.2.1 Filter methods

Filter methods rank features based on simple statistical scores such as mean, variance, and correlation coefficients. Features are then either kept or removed based on their ranking. Filter methods are often univariate and thus consider each feature independently or in terms of their association with the target variable. Examples of filter techniques include the

TABLE 2.1 Automated feature selection methods.

Feature selection	Popular methods	Selection based on machine learning algorithm's performance	Part of the learning process	Computational resources	Univariate/ multivariate	Overfitting
Filter	• Correlation coefficient • t-test • ANOVA	No	No	Low	Univariate	Low
Wrapper	• Recursive feature elimination algorithm • Searchlight	Yes	No	High	Multivariate	Highest
Embedded	• Lasso • Ridge regression • Elastic net	Yes	Yes	Moderate	Multivariate	High

Pearson correlation coefficient and group-level statistics such as t-tests and ANOVA. In addition to their simplicity, filter methods are also computationally inexpensive, easy to interpret, and less prone to overfitting than alternative approaches. However, they fail to take multivariate interactions between features into account, which can lead to the selection of redundant features or the loss of relevant information. Furthermore, filter methods select features without taking the training of the machine learning algorithm into account. Instead, filter methods often choose features based on *P*-values. However, features with small *P*-values at group level do not necessarily equate to large discrimination power (Arbabshirani, Plis, Sui, & Calhoun, 2017).

2.4.3.2.2 Wrapper methods

In wrapper methods, the machine learning algorithm is wrapped to a feature selection algorithm that screens the original pool of features for the combination of features that yields the highest performance (Guyon & Elisseeff, 2003). Wrapper approaches can be categorized into forward selection and backward elimination methods (Kohavi & John, 1997). In forward selection, the relevant features are iteratively added to an initially empty set, whereas in backward elimination, the search begins with all features and involves iterative removal. In both cases, the search ends until an optimal number of features is found. In contrast to filter methods, wrapper methods are multivariate and therefore can take interactions between features into account. They also tend to select better performing features because their selection is dependent on the algorithm's performance. However, they can be computationally expensive and are more prone to overfitting. An example of a common wrapper method in brain disorders research is the backward elimination technique known as recursive feature elimination algorithm (Guyon, Weston, Barnhill, & Vapnik, 2002).

2.4.3.2.3 Embedded methods

Similar to wrapper methods, embedded approaches also select features based on the performance of the algorithm. However, unlike wrapper methods, embedded approaches are an intrinsic part of the learning process, i.e., they learn what features contribute the most to model performance during the model training. The most common type of embedded feature selection methods in brain disorders research is known as regularization methods. These methods are typically used when there are a large number of features, and all of them are thought to contribute to the task. Regularization techniques select features by shrinking the weight (i.e., importance) of some features to zero, a procedure known as *regularization penalty*; only those features with higher discriminatory power will have a nonzero weight and therefore will be allowed to contribute to

the task. The most popular embedded methods with regularization penalty are the least absolute shrinkage and selection operator (Tibshirani, 1996), ridge regression (Yu & Liu, 2003), and elastic net (Zou & Hastie, 2005).

2.4.4 Feature scaling/normalization

In brain disorders research, we often deal with datasets that contain features that vary in units and range. However, to model the data correctly and effectively, most machine learning algorithms require the data to be on the same scale. There are several ways to achieve this. Two of the most commonly used methods are min-max scaling and z-score normalization (or standardization). The first one transforms the data to a desired range:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

where x' is the new value and $x_{\min}$ and $x_{\max}$ are the lowest and maximum values of the distribution, respectively. Z-score normalization, on the other hand, changes the data so that they resemble a normal distribution.

$$x' = \frac{x - \overline{x}}{\sigma}$$

where $\overline{x}$ and σ are the mean and standard deviation, respectively, for the same feature.

Once the features have been scaled or normalized, data are ready to be analyzed by the machine learning algorithm.

2.5 Model training

In supervised learning, model training refers to the process in which a machine learning algorithm finds a function f that best maps some given input features X and target variable y (please see Part II of this book for an introduction to some of the most commonly used machine learning algorithms in brain disorders research).

$$y = f(X)$$

Importantly, however, the fundamental goal of machine learning is to find a function f capable of high performance when applied to new unseen data. Therefore, the ultimate test of a machine learning algorithm is to see how well the learned function f generalizes to data not used for training.

2.5.1 Training and test sets

The simplest way to assess a model generalization is to randomly divide the whole dataset into two subsets: the *training set* and the *test set*. First, the machine learning algorithm is fitted using the training set; this step involves learning the function *f*. The model is then evaluated using the test set; this step consists in assessing the performance of the learned function in unseen data, which the model did not use for training. However, splitting the data into a single training and test sets, also known as the *hold-out* method, is uncommon in brain disorders research. This is because, as studies often comprise small samples, there would be a high risk that the learned function would be based on variations in the data that are specific to a particular split of training/test sets. This means that, even if the model performs well in a small test set, it might have happened by chance; there is no guarantee it will perform similarly well with a different partition of participants between the training and test sets. The performance of the model would be very vulnerable to chance variations in the training and testing sets, rather than reflecting a robust association between the feature set and the target variable. On the other hand, as sample size increases, it is less likely that the machine learning model will perform well by chance, and we can be more confident whatever the performance turns out to be.

2.5.2 Cross-validation

An alternative approach for estimating the model's performance is splitting the available data into training/test sets several times, each time with a different group of observations in each set. Therefore, instead of having only one measure of performance, we have several, one for each train/test split, or iteration. Once all iterations are completed, we have a better estimate of the overall performance of the machine learning algorithm by calculating the average performance across all iterations. This process also provides us with a measure of variability and stability of the model performance. This method can be executed using the *cross-validation* (*CV*) procedure (Fig. 2.3).

There are several possible ways of resampling the data. A common approach involves using all available data as the training set, except one observation (e.g., data from one participant), which is then used for testing. This process is repeated as many times as there are observations. This approach is known as *leave-one-out CV* (LOOCV) (Fig. 2.4A). Another common method involves leaving out a group of observations—instead of just one observation—at each iteration. To do this, all available data are first randomly partitioned into *k* equal sized subsamples (or folds). At each iteration, one of the subsamples is used for testing while the

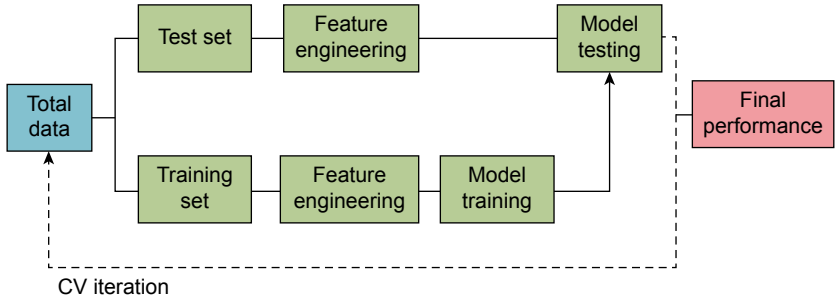

FIGURE 2.3 The cross-validation (CV) procedure. At each iteration of the CV, data are split into two subgroups: training and test sets. The relevant feature engineering methods (e.g., feature selection, dimensionality reduction, and scaling/normalization) are fitted and applied to the training set. Once ready, the transformed training data are used as input to train the machine learning algorithm. The same data transformations are applied to the test set using the same parameters as in the training set. Finally, the trained model is used to predict the target variable in test set. This process is repeated until all iterations are processed. Once all iterations are completed, the overall performance of the machine learning algorithm is estimated.

FIGURE 2.4 Three types of cross-validation (CV). (A) Two types of CV are displayed according to the number of observations in the training/test sets at each iteration: leave-one-out CV (LOOCV) and k-fold CV; (B) nested CV.

remaining subsamples are used for training; as there are *k* subsamples, the process is repeated *k* times. This type of CV is known as *k-fold CV* (Fig. 2.4A). One of the most commonly used *k*-fold CV schemes is the 10-fold CV. Here, data are divided into 10 subgroups. At every iteration, nine groups are used for training and the remaining group is used for testing; this is repeated 10 times, each time with a different group as the test set.

How do we know which CV scheme to use?—LOOCV? or 5-fold CV? or 10-fold? There is no absolute answer to this question. The main thing to consider is how the number of iterations impacts the bias and variance of the testing process. On one side of the spectrum is the twofold CV. As we have already seen, a model with just one split is very prone to chance—the performance may be very poor, good, or very good, depending on what participants end up in the training and test sets. This approach introduces bias to our testing process. On the other side of the spectrum is LOOCV. Here, because the difference in size between the training set used in each iteration and the entire dataset is only a single observation, the overall performance measure will be approximately unbiased. However, because the performance at each iteration corresponds to how well the model did for the one testing participant in that iteration, there will be more variability in the test error (especially if the dataset has multiple outliers). This high variance is not a problem in the context of twofold CV, where half of the total dataset is being used for training and the remaining half is being used for testing.

From here we conclude that there is a bias−variance trade-off associated with the choice of *k*: the closer *k* is to N (sample size), the lower the bias and higher the variance; on the other hand, the closer k is to 2, the higher the bias and lower the variance. Typically, a value of *k* of 5 or 10 is usually suggested as they have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance. LOOCV is one of the most commonly used CV schemes in the brain disorders literature. However, in addition to having high variance, it can also be more expensive; applying more sophisticated machine learning models to complex data (e.g., high-dimensional neuroimaging data, multimodal data, large consortium data) may not be feasible with a LOOCV.

2.5.2.1 Nested cross-validation

The inner workings of a machine learning algorithm depend on a set of parameters that determine the properties of the model. These parameters are called *hyperparameters* to distinguish them from standard model parameters adjusted during training. For example, in neural networks, the number of layers and units are hyperparameters, while the weights connecting the layers involved in the internal data processing are the parameters of the model.

To maximize the performance of a machine learning model, it is common practice to test a range of values for a given hyperparameter. This could be done by running the CV procedure several times, where different hyperparameters are tested each time, and then selecting the ones that yield the best performance. However, this approach would result in a biased estimation of performance as the final model is carefully chosen based on its performance on the test set. One crucial rule in machine learning is "Never tune the machine learning model on the test set."

To avoid this issue, the training set can be further divided into two sets of data: *training* and *validation*. Here, we can perform a single training/validation split. However, it will likely result in the same problem of using a single split for training/test, as previously discussed. For this reason, it is considered good practice to use an additional CV, called inner CV. The several iterations of this inner CV are used to train and test different hyperparameters. The best performing hyperparameter in the inner CV is then used to train a model in the whole training set defined by the outer CV. Finally, this model is used to predict the target variable in the test set of the outer CV. The final performance is estimated by averaging the performance in the test set across all outer CV iterations (Figs. 2.4B and 2.5).

Hyperparameter tuning is often described as being part of the *model selection* process. Building a successful machine learning model often involves testing different approaches. Model selection refers to the process of choosing between different models based on their performance with different machine learning algorithms, sets of features, or, just as we described here, choosing between different hyperparameters.

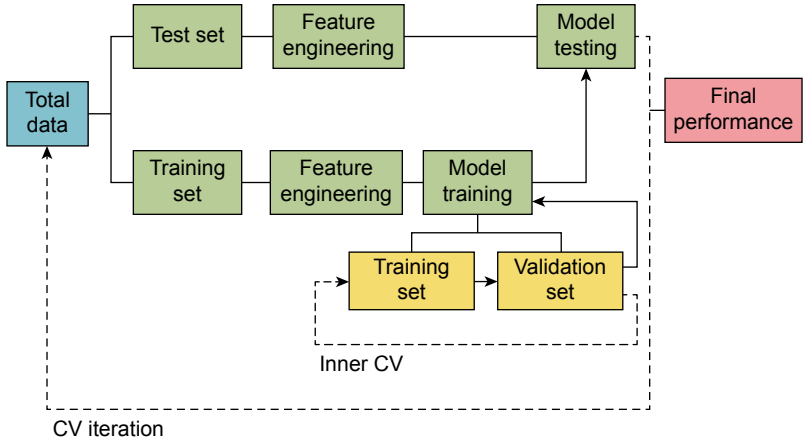

FIGURE 2.5 The cross-validation (CV) procedure with a nested CV.

2.5.2.2 *Stratified cross-validation*

Another critical issue in brain disorders research is the fact that datasets are often imbalanced. Suppose we are interested in classifying 100 individuals who met criteria for high risk for psychosis into those who did and did not transition to full-blown psychosis. Assuming 20 participants will have transitioned over 1 year (Fusar-Poli et al., 2012), the ratio between labels would be roughly 1:5. A 10-fold CV, for example, would randomly divide the total sample into 10 equal subgroups, which is likely to result in a disproportionate number of transition/nontransition labels at different iterations of the CV. This would be most likely to introduce some sampling bias, as it is difficult for a machine learning model to learn a decision function without seeing enough examples of one of the labels. To avoid this issue, it is possible to sample training and test sets in such a way that the same ratio between the different labels is kept throughout the iterations. This approach is known as *stratified CV* (Fig. 2.6).

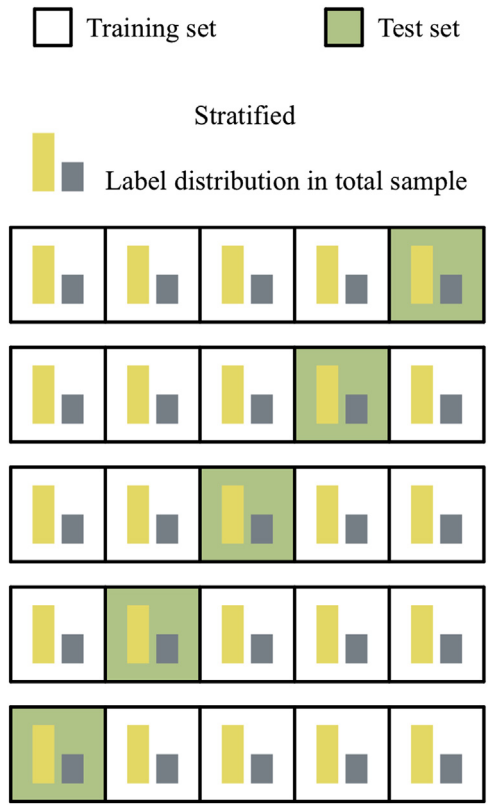

FIGURE 2.6 Stratified cross-validation.

2.5.3 Implementing data transformations in the machine learning pipeline

Dimensionality reduction, automated feature selection methods, and scaling/normalization (or other data transformations) result in a different set of features compared to the original data. Therefore, it is important to consider these steps as part of the model selection process. For example, although it may seem intuitive to first select the most informative features and only then split the data into training/test sets and implement the algorithm, doing so would introduce bias. If any data transformation is applied to the total available data, then the data set aside to test the model, i.e., the test set, would no longer be entirely independent from the training set. Therefore, the performance of the algorithm in the test set would no longer be considered an unbiased measure of generalizability. This issue is commonly known as *double dipping* and tends to yield overoptimistic results (see Mwangi et al. (2014) for a comparison of feature selection with and without double dipping). To avoid this source of bias, data transformations such as feature selection, dimensionality reduction, or normalization should be included in the CV framework, as shown in Fig. 2.3. At each iteration of the CV, feature selection/dimensionality reduction/normalization are implemented on the training data and the resulting features are used to train the algorithm; once trained, the algorithm is tested using the test set after the same set of features have been selected/extracted/normalized. For example, if features A and B were removed from the training set, the same features should be removed from the test set.

2.5.4 Bias–variance trade-off

Building a successful algorithm is often a continuous process, where increasingly more complex models (e.g., models with large number of parameters to estimate) are developed to achieve better performances. As complexity increases, however, there are two main sources of error—bias and variance—that need to be balanced (Fig. 2.7). It should be noted that this trade-off between bias and variance is different from the one discussed in Section 2.5.2: the former refers to the learning taking place during training of the model, whereas the latter is concerned with the testing of the model. Bias arises when the model learns a faulty assumption in the data. In the presence of high bias, the algorithm will not be able to model the relationship between features and target correctly; this is known as *underfitting*. As shown in Fig. 2.7A, although data tend to *plateau*, the model assumes there is a linear association between the variables. A model that uses inefficient or uninformative features, a very small number of observations, or a too simple algorithm, for example, is

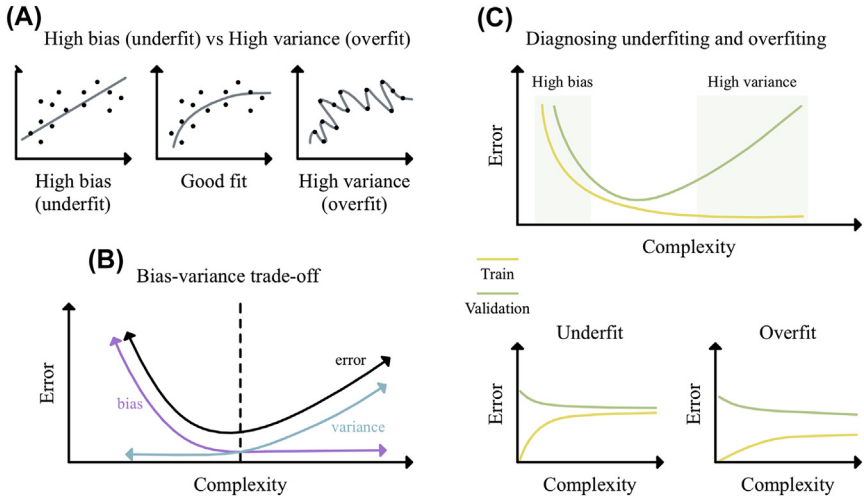

FIGURE 2.7 (A) Three models with different levels of bias and variance. The model in the left has low variance but high bias, while the one on the right has high variance but low bias. An optimal solution would be a model with a good balance between bias and variance. (B) Bias–variance trade-off. As model complexity increases, variance increases and bias decreases. Ideally, bias and variance can be balanced, and the algorithm will achieve convergence (when additional training will not improve the model). (C) Diagnosing underfitting and overfitting. By assessing the error in the training and validation sets, it is possible to establish whether a model is under- or overfitting the data.

likely to be too simple to capture any meaningful patterns and result in a highly biased model. On the other hand, variance results from modeling detailed fluctuations in the data. An algorithm with high variance will capture specific aspects of the training data that do not generalize well in the test set; this is known as *overfitting*. This happens when, for example, there are too many features relative to the number of observations or when very complex models are implemented (Fig. 2.7A). The way in which different levels of model complexity affect bias and variance is known as the bias–variance trade-off (Fig. 2.7B).

When an algorithm performs poorly, this will most likely be due to either a high bias or high variance issue. Therefore, to improve the model's performance, it is necessary first to identify whether a model is underfitting or overfitting the data (Fig. 2.7C). When the model is too simple, error in both training and validation sets is high, indicating that the model is not a good fit, i.e., the model is underfitting the data. On the other hand, when the model is too complex, it fits the training data very well, i.e., the error is close to zero; however, the error in the validation set is high, indicating overfitting and poor generalizability. Ideally, a good machine learning model will have a small error in the training set and a slightly larger, although still small, error in the validation set.

2.6 Model evaluation

Once the algorithm has been trained and then applied to the test set, the next step is to decide how to quantify its performance. In this section, we introduce some of the most prominent evaluation metrics for classification and regression.

2.6.1 Classification

In classification, performance is calculated by comparing the predicted labels against the true labels in each test set of the CV. The final performance is then estimated by averaging the performance in the test sets across all iterations of the CV (Fig. 2.8).

Imagine a study that aims to develop a model to distinguish 100 depressive patients in two groups: those who responded (respondents) and did not respond (nonrespondents) to antidepressant medication. The predictions made by the classifier would fall under four categories:

- *True positives* (*TP*): proportion of respondents who were correctly identified as respondents
- *False positives* (*FP*): proportion of nonrespondents classified as respondents
- *True negatives* (*TN*): proportion of nonrespondents identified as nonrespondents
- *False negatives* (*FN*): proportion of respondents classified as nonrespondents

These categories can be easily organized in a summary table known as the *confusion matrix* (Fig. 2.9). From this matrix it is possible to calculate several metrics to describe the performance of a classifier. Each metric highlights slightly different aspects of the model. For this reason, it is

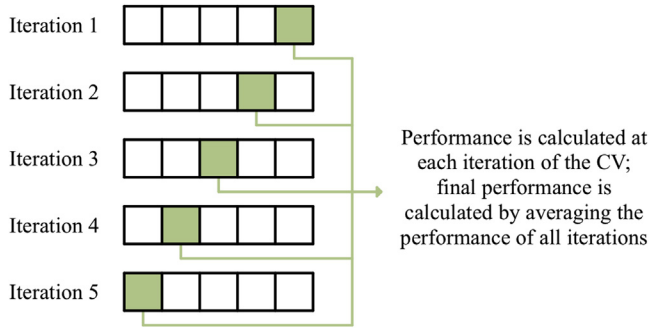

FIGURE 2.8 Final performance in a cross-validation (CV) scheme.

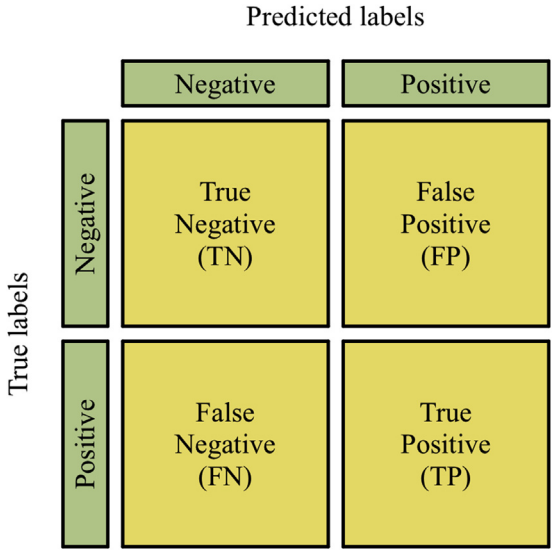

FIGURE 2.9 Confusion matrix.

considered good practice to report the confusion matrix. This way, the reader has access to a range of metrics and not just the ones the authors decide to report.

2.6.1.1 Accuracy

Accuracy refers to the proportion of correct predictions. Simple to calculate and intuitive to understand, accuracy is one of the most commonly used metrics.

$$\text{Acc} = \frac{TP + TN}{TP + FP + TN + FN}$$

However, a key limitation of this metric is that it does not account for imbalanced dataset, i.e., different number of observations for each label. To illustrate how such imbalance could be problematic, imagine that the accuracy for our treatment study was 75%. In theory, this could be a reasonable result; however, after checking the individual predictions made by the classifier, it turns out all participants were classified as nonrespondents. This happened because the algorithm learned that, as only 25 patients responded to treatment, the most efficient way to achieve a high accuracy was to always predict nonresponse, as this is by far the most likely outcome. This limitation can have major consequences in brain disorders research, where datasets are often imbalanced and biased results might preclude individuals from getting appropriate diagnosis or treatment. A more appropriate metric is therefore *balanced accuracy*, which

is derived by computing the average between the accuracies for each class (Brodersen, Ong, Stephan, & Buhmann, 2010).

$$\text{Balanced Acc} = \frac{\frac{TP}{TP+FN} + \frac{TN}{TN+FP}}{2}$$

In the same example, the balanced accuracy would be 50% because nonrespondents are identified with an accuracy of 100%, but respondents are identified with an accuracy of 0%.

2.6.1.2 Sensitivity/true positive rate/recall

Sensitivity is the proportion of positives correctly identified as positives (e.g., the proportion of patients identified as such in a case–control study).

$$\text{Sens} = \frac{TP}{TP + FN}$$

Sensitivity is a useful metric in situations where we do not want to miss any positive cases. For example, when building an algorithm to detect individuals at risk of developing a mental disorder, we want to minimize false negatives, i.e., instances when someone is classified as a control when they are a case.

2.6.1.3 Specificity/true negative rate

Specificity refers to the proportion of negatives that are correctly identified as negatives (e.g., the proportion of healthy controls identified as such in a case–control study).

$$\text{Spec} = \frac{TN}{TN + FP}$$

Specificity is a useful metric where we do not want to miss any negative cases. For example, when developing an algorithm to predict who is likely to respond to treatment, we want to minimize false positives, i.e., instances when someone is classified as a respondent when they are a nonrespondent.

2.6.1.4 Area under the receiver operating characteristic curve (AUC-ROC)

In addition to providing a categorical prediction about class membership, models can also output the probability of belonging to a certain group. In a binary classification task, most classifiers assign observations to one class or another based on a probability threshold (known as classification threshold) of 0.5. For example, in a binary classification task with classes A and B, if the algorithm outputs a probability of a

participant belonging to class A higher than 0.5, this participant will be assigned to this class; otherwise the participant will be assigned to the class B. We can manipulate the classification threshold and calculate the sensitivity and specificity at each possible value of the classification threshold (from 0 to 1). If we then plot 1-specificity (false positive rate) against the sensitivity (true positive rate) for each possible classification threshold level, this generates a curve known as *receiving operating curve* (*ROC*). The performance of the classifier can then be summarized in one number by calculating the area under the ROC curve (AUC-ROC) (Fig. 2.10). The AUC-ROC for a perfect classifier is 1 because the true positive rate and false positive rate are 1 and 0, respectively. Typically, an AUC-ROC of 0.5 suggests no discrimination, 0.7–0.8 is considered acceptable, 0.8–0.9 is considered excellent, and more than 0.9 is considered outstanding.

2.6.1.5 Multiclass classification

Multiclass classification refers to a situation in which a classifier has to predict more than two classes. This is a crucial issue in brain disorders as, in practice, most decisions made by clinicians are nonbinary. For example, when a clinician assesses a patient, the aim is not only to determine whether the patient has a mental disorder but also to ascertain which disorder best explains their symptoms. Although some algorithms allow multiclass classification directly, several of the most commonly used algorithms do not. In such cases, multiclass classification is often addressed by transforming the problem into a series of binary

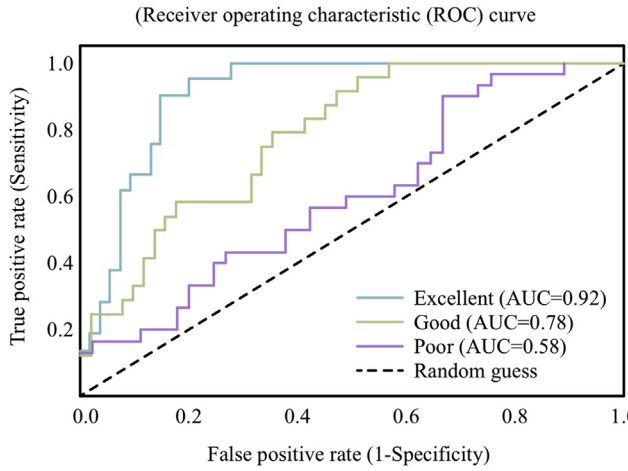

FIGURE 2.10 Example of three AUC-ROC curves displaying increasingly levels of performance.

classifications, either by performing pairwise comparisons (for example, treatment A vs. no treatment, treatment B vs. no treatment, and treatment A vs. treatment B) (one-vs-one classification) or comparing one class against the combination of all other classes (one-vs-all classification).

2.6.2 Regression

In a regression problem, performance is estimated by calculating how much the predicted scores ($\widehat{y}$) deviate from the actual target value (y) in the test set. Similarly to classification, there are several possible ways of measuring performance.

2.6.2.1 Mean absolute error, mean squared error, and root mean squared error

The mean absolute error (MAE) is the average of the absolute differences between predictions and actual values.

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}\left|y_j - \widehat{y}_j\right|$$

The mean squared error (MSE) is the sum of the square of the difference between the predicted and target variable, divided by the number of observations. By squaring the difference between predicted and target variable, MSE solves the issue of positive and negative numbers canceling each other out when added. However, the final measure of error is now the squared of the original units, which hinders interpretability.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{y}_i\right)^2$$

The square root of the MSE (root mean squared error or RMSE) solves this problem by converting the units back to the original units of the output variable.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{y}_i\right)^2}$$

All these metrics measure the error of the model. Therefore, the smaller the values of MAE, MSE, and RMSE, the better the performance of model. MAE is more robust to outliers than MSE and RMSE, as it does not square the differences. On the other hand, by squaring the errors before they are averaged, RMSE gives more weight to large errors than MAE and MSE. Therefore, the RMSE should be more useful when large errors are particularly undesirable.

2.6.2.2 R-squared

The R-squared (R^2) or coefficient of determination metric indicates how well the predictions made by the algorithm fit the true values. The R^2 can range between 0 and 1, indicating no-fit and perfect fit, respectively.

2.7 Post hoc analysis

Once the model training and testing are completed, we can run additional statistical analyses to improve our understanding of the final model(s). Two of the most common analyses include testing if the model's performance is statistically significant and identifying the features that made the greatest contribution to prediction.

2.7.1 Significance testing

In addition to estimating metrics to quantify the performance of an algorithm, it is considered good practice to verify the statistical significance of these metrics to assess whether the model's performance is better than chance level. This step is crucial when dealing with small sample sizes to minimize the risk of overoptimistic conclusions. In this context, significance testing is usually carried out on the performance metric using permutation testing.

Essentially, permutation testing measures the likelihood that the model's performance would be observed by chance. Specifically, for a supervised algorithm, target variables are randomly shuffled along all subjects (usually 10,000 times) until any statistical relationship between them and the input features is lost. The cross-validated performance is then calculated for each permutation, resulting in a statistical distribution of measures, which reflects the null hypothesis that the model behaves by chance. The number of times the performance is greater than or equal to the original performance is then divided by the number of permutations to estimate a *P*-value (Good, 1994).

2.7.2 Identifying the most informative features

In brain disorders research, a model that performs well but provides little or no information about how input features are contributing to the predictions would be of limited clinical utility. The knowledge of the pattern in the data driving the predictions can provide meaningful clinical information, for instance, with respect to treatment development and optimization, as well the underlying disease mechanisms. Identifying the most important features is also useful to exclude the possibility that the

algorithm is being driven by artifacts in the data (e.g., movement artifacts in imaging data).

In most cases, the relative importance of input features is estimated during the training process, whereby features are assigned weight values corresponding to their relative contribution to the decision function. For example, in Mechelli et al. (2017), in addition to predicting later transition to psychosis from clinical information with 65% accuracy, the authors also reported that high disorder of thought content and low functioning at baseline were the features that provided the most significant contribution to the overall discrimination function.

2.8 Conclusion

In this chapter, we have introduced some of the fundamental concepts in machine learning and discussed the main steps of a standard pipeline. A good understanding of the rationale for each of these steps, as well as the main challenges associated with them, is essential to minimize the risk of spurious results. Building a successful machine learning model also relies on a good understanding of the multitude of possible methods available that can be used at different steps of the process. Deciding which approach to implement at each stage of the analysis should be an informed process based on several factors, such as type and number of features, type of target variable (i.e., is it a classification or regression problem?), available computational resources, and an in-depth appreciation of the relative advantages and disadvantages of each method. All concepts introduced in this chapter are used throughout this book. Several aspects, such as dimensionality reduction and the implementation of different machine learning algorithms, are covered in detail in later chapters.

2.9 Key points

- The supervised machine learning pipeline is comprised of six main stages: (i) problem formulation, (ii) data preparation, (iii) feature engineering, (iv) model training, (v) model evaluation, and (vi) post hoc analysis.
- A well-defined supervised machine learning problem should include a clear definition of a feature set, target variable, and task.
- Careful data preparation, including checking for outliers and possible confounding variables, should be carried out to avoid spurious interpretations.

- Feature engineering is a critical step in the standard machine learning pipeline; most of the time devoted to machine learning is spent on crafting useful features.
- Data transformations such as dimensionality reduction, automated feature selection, and scaling/normalization should be implemented as part of the CV scheme.
- There are several performance metrics to choose from; it is good practice to present the confusion matrix for classification problems, from which most classification metrics can be derived.
- Testing performance for statistical significance is especially important in brain disorder studies where samples tend to be small resulting in higher risk of overoptimistic results.

References

Arbabshirani, M. R., Plis, S., Sui, J., & Calhoun, V. D. (2017). Single subject prediction of brain disorders in neuroimaging: Promises and pitfalls. *NeuroImage, 145*, 137–165. https://doi.org/10.1016/j.neuroimage.2016.02.079.

Brodersen, K. H., Ong, C. S., Stephan, K. E., & Buhmann, J. M. (2010). The balanced accuracy and its posterior distribution. In *2010 20th international conference on pattern recognition* (pp. 3121–3124). IEEE. https://doi.org/10.1109/ICPR.2010.764.

Fusar-Poli, P., Bonoldi, I., Yung, A. R., Borgwardt, S., Kempton, M. J., Valmaggia, L., et al. (2012). Predicting psychosis: meta-analysis of transition outcomes in individuals at high clinical risk. *Archives of General Psychiatry, 69*(3), 220–229. https://doi.org/10.1001/archgenpsychiatry.2011.1472.

Good, P. (1994). *Permutation tests: A practical guide to resampling methods for testing hypotheses*. New York: Springer.

Guyon, I., & Elisseeff, A. (March 2003). An introduction to variable and feature selection. *Journal of Machine Learning Research, 3*, 1157–1182.

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning, 46*(1/3), 389–422. https://doi.org/10.1023/A:1012487302797.

Jolliffe, I. (2002). *Principal component analysis*. Berlin: Springer.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence, 97*(1–2), 273–324. https://doi.org/10.1016/S0004-3702(97)00043-X.

Mechelli, A., Lin, A., Wood, S., McGorry, P., Amminger, P., Tognin, S., et al. (2017). Using clinical information to make individualized prognostic predictions in people at ultra-high risk for psychosis. *Schizophrenia Research, 184*, 32–38. https://doi.org/10.1016/j.schres.2016.11.047.

Mwangi, B., Tian, T. S., & Soares, J. C. (2014). A review of feature reduction techniques in neuroimaging. *Neuroinformatics, 12*(2), 229–244. https://doi.org/10.1007/s12021-013-9204-3.A.

Rao, A., Monteiro, J. M., Mourao-Miranda, J., & Alzheimer's Disease Initiative. (2017). Predictive modelling using neuroimaging data in the presence of confounds. *NeuroImage, 150*, 23–49. https://doi.org/10.1016/j.neuroimage.2017.01.066.

Saeys, Y., Inza, I., & Larranaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics, 23*(19), 2507–2517. https://doi.org/10.1093/bioinformatics/btm344.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 267–288. https://doi.org/10.2307/2346178.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. A. (December 2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research, 11*, 3371–3408.

Yu, L., & Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning* (pp. 856–863). ICML-03.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67*(2), 301–320. https://doi.org/10.1111/j.1467-9868.2005.00503.x.