Chapter 4
Overfitting, Model Tuning, and Evaluation of Prediction Performance

4.1 The Problem of Overfitting and Underfitting

The *overfitting* phenomenon occurs when the statistical machine learning model learns the training data set so well that it performs poorly on unseen data sets. In other words, this means that the predicted values match the true observed values in the training data set too well, causing what is known as overfitting. Overfitting happens when a statistical machine learning model learns the systematic and noise (random fluctuations) parts in the training data to the extent that it negatively impacts the performance of the statistical machine learning model on new data. This means that the statistical machine learning model adapts very well to the noise as well as to the signal that is present in the training data. The problem is that these concepts do not apply to independent (new) data and negatively affect the model's ability to generalize. Overfitting is more probable when learning a loss function from a complex statistical machine learning model (with more flexibility). For this reason, many nonparametric statistical machine learning models also include constraints in the loss function to improve the learning process of the statistical machine learning models. For example, artificial neural networks (ANN), mentioned later, are a nonparametric statistical machine learning model that is very flexible and is subject to overfitting training data. This problem can be addressed by dropping out (setting to zero) the weights of a certain percentage of hidden units in order to avoid overfitting.

On the other hand, an *underfitted* phenomenon occurs when few predictors are included in the statistical machine learning model, i.e., it is a very simple model that poorly represents the complete picture of the predominant data pattern. This problem also arises when the training data set is too small or not representative of the population data. An *underfitted* model does a poor job of fitting the training data and for this reason it is not expected to satisfactorily predict new data points. This implies that the predictions using unseen data are weak, since individuals are perceived as strangers unfamiliar with the training data set.

O. A. Montesinos López et al., *Multivariate Statistical Machine Learning Methods for Genomic Prediction*, https://doi.org/10.1007/978-3-030-89010-0_4

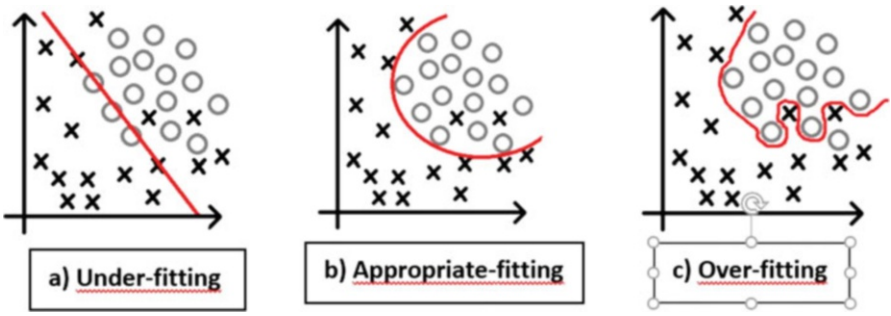

Fig. 4.1 Schematic illustration of three models for classification: (**a**) M1 with underfitting, (**b**) M2 with appropriate fitting, and (**c**) M3 with overfitting

Consider a scattered series of points $(y_1, x_1), \ldots, (y_n, x_n)$, on a plane, to which we want to adjust a statistical machine learning method. This means that we are looking for the best $f(\boldsymbol{x}_i)$ that explains the existing relationship between the response variable (y_i) and the predictors $(x_1, \ldots, x_n)$. We assume that we have three options for $f(\boldsymbol{x}_i)$: M1, the simple model plotted in Fig. 4.1a; M2, an intermediate model shown in Fig. 4.1b; and M3, a complex model shown in Fig. 4.1c.

Under the classification framework, the first panel in Fig. 4.1 (left side, panel a) shows an unsatisfactory fit (underfitted) since the line does not cover most of the points (has high bias) in the plot. As such, we expect that the prediction of unseen data of this model, M1, will perform badly. In contrast, panel c of Fig. 4.1 shows an almost perfect fit, since the predicted line covers all the data points. While at first glance, you may think that model M3 will perform well when predicting unseen data, this is actually untrue since the predicted line covers all points that are noise and those that are signal (overfit); for this reason, this type of model also performs poorly in the prediction of future data due to its complexity and high variance. Therefore, the best option for predicting unseen data is model M2 (Fig. 4.1, panel b) since it represents the predominant (smooth) pattern enough to represent the apparent data pattern while maintaining a balance between bias and variance. For this reason, a well-fitted model is one that faithfully represents the sought-after predominant pattern in the data, while ignoring the idiosyncrasies in the training data. As such, a well-fitted model in the testing set should be in the neighborhood of the model's accuracy based on the training data set, that is, the model's accuracy in the testing set should be approximately equal to that of the model's accuracy in the training set. In contrast, an overfitted model in the testing data set will be far from the neighborhood of the model's accuracy based on the training data set, and usually its prediction performance is very high (good) in the training set and consequently low (bad) in the testing set (Ratner 2017).

The paradox of overfitting is defined as complex models that contain more information about the training data, but less information about the testing data (future data we want to predict). In statistical machine learning, overfitting is a major issue and leads to some serious problems in research: (a) some relationships

that seem statistically significant are only noise, (b) the complexity of the statistical machine learning model is very large for the amount of data provided, and (c) the model in general is not replicable and predicts poorly.

Since the main goal of developing and implementing statistical machine learning methods is to predict unseen data not used for training the statistical machine learning algorithm, researchers are mainly interested in minimizing the testing error (generalization error applicable to future samples) instead of minimizing the training error that is applicable to the observed data used for training the statistical machine learning algorithm.

According to Shalev-Shwartz and Ben-David (2014), if the learning fails, these are some approaches to follow:

1. Increase the sample size of the training set.
2. Modify the hypothesis by (a) enlarging it, (b) reducing it, (c) completely changing it, and (d) changing the parameters being used. We understand a hypothesis as the models and their parameters under evaluation. This point is very important to reach a reasonable model for your data.
3. Change the feature representation of the data.
4. Change the statistical machine learning algorithm used.

4.2 The Trade-Off Between Prediction Accuracy and Model Interpretability

Accuracy is the ability of a statistical machine learning model to make correct predictions and those models with more complexity (called flexible models) are better in terms of accuracy, while the simple, less complex models (called inflexible models) are less accurate but more interpretable. Interpretability indicates to what degree the model allows for human understanding of natural phenomena. For these reasons, when the goal of the study is prediction, flexible models should be used; however, when the goal of the study is inference, inflexible models are more appropriate because they more easily interpret the relationship between the response variables and the predictor variables. As the complexity of the statistical machine learning model increases, the bias is reduced and the variance increases. For this reason, when more parameters are included in the statistical machine learning model, the complexity of the model increases and the variance becomes the main concern while the bias steadily falls. For example, James et al. (2013) state that the linear regression model is a relatively inflexible method because it only generates linear functions, while the support vector machine method is one of the most flexible statistical machine learning methods.

Before providing an analytical interpretation of the trade-off between the bias and variance, we must understand the meaning of both concepts. Bias is the difference between the expected prediction of our statistical machine learning model and the true observed values. For example, assume that you poll a specific city where half of

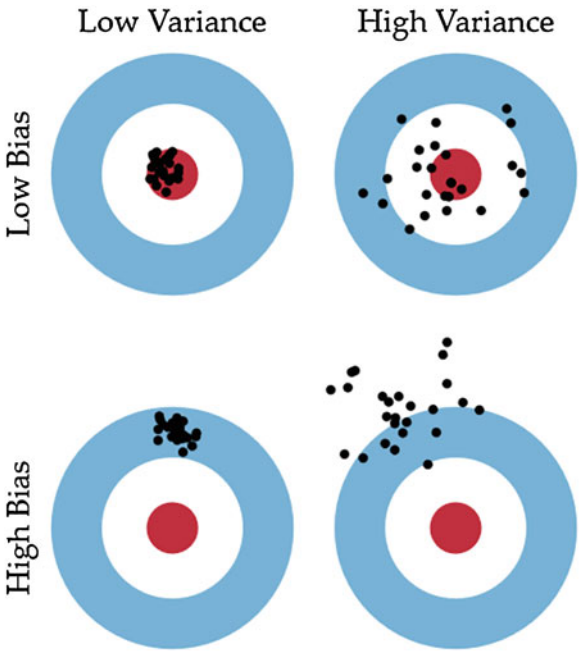

Fig. 4.2 Graphical representations of different levels of bias and variance

the population is high-income and the other half is low-income. If you collected a sample of high-income people, you would conclude that the entire city has high income. This means that your conclusion is heavily biased since you only sampled people with high income. On the other hand, error variance refers to the amount that the estimate of the objective function will change using a different training data set. In other words, the error variance accounts for the deviation of predictions from one repetition to another using the same training set. Ideally, when a statistical machine learning model with low error variance predicts a value, the predicted value should remain almost the same, even when changing from one training data set to another; however, if the model has high variance, then the predicted values of the statistical machine learning method are affected by the values of the data set. We provide a graphical visualization of bias and variance with a bull's eye diagram (see Fig. 4.2). We assume that the center of the target is a statistical machine learning model that perfectly predicts the correct answers. As we move away from the bull's eye, our predictions get worse. Let us assume that we can repeat our entire statistical machine learning model building process to get a number of separate hits on the target. Each hit represents an individual realization of our statistical machine learning model, given the chance variability in the training data we gathered. Sometimes we accurately predict the observations of interest since we captured a representative sample in our training data, while other times we obtain unreliable predictions since our training data may be full of outliers or nonrepresentative values. The four combinations of cases resulting from both high and low bias and variance are shown in Fig. 4.2.

Burger (2018) concludes that the best scenario is the one with *low bias* and *low variance*, since samples are acceptable representatives of the population (Fig. 4.2), while in the case of *high bias* and *low variance*, the samples are fairly consistent, but not particularly representative of the population (Fig. 4.2). However, when there is *low bias* and *high variance*, the samples vary widely in their consistency, and only some may be representative of the population (Fig. 4.2). Finally, with *high bias* and *high variance*, the samples are somewhat consistent, but unlikely to be representative of the population (Fig. 4.2).

If we denote the variable we are trying to predict as y and our covariates as $\boldsymbol{x}_i$, we may assume that there is a relationship between y and $\boldsymbol{x}_i$, as that given in Eq. (1.1) from Chap. 1, where the error term is normally distributed with a mean of zero and variance σ^2. The expected prediction error for a new observation with value x, using a quadratic loss function, is given by Hastie et al. (2008, page 223):

$$\begin{aligned} E\left(y-\widehat{f}\{\boldsymbol{x}_i\}\right)^2 &= E(y-f\{\boldsymbol{x}_i\})^2 + \left(E\left(\widehat{f}\{\boldsymbol{x}_i\}\right)-f\{\boldsymbol{x}_i\}\right)^2 + E\left\{\widehat{f}(\boldsymbol{x}_i)-E\left(\widehat{f}\{\boldsymbol{x}_i\}\right)\right\}^2 \\ &= \mathrm{Var}(y) + \left[\mathrm{Bias}\left(\widehat{f}\{\boldsymbol{x}_i\}\right)\right]^2 + \mathrm{Var}\left(\widehat{f}(\boldsymbol{x}_i)\right) \\ &= \mathrm{Var}(\epsilon) + \left[\mathrm{Bias}\left(\widehat{f}\{\boldsymbol{x}_i\}\right)\right]^2 + \mathrm{Var}\left(\widehat{f}(\boldsymbol{x}_i)\right), \end{aligned}$$

where Bias is the result of misspecifying the statistical model f. Estimation variance (the third term) is the result of using a sample to estimate f. The first term is the error (irreducible error) that results even if the model is correctly specified and accurately estimated. This irreducible error is the noise term in the true relationship that cannot fundamentally be reduced by any model. Given the true model and infinite data to train (calibrate) it, we should be able to reduce both the bias and variance terms to 0. However, in a world with imperfect models and finite data, there is a trade-off between minimizing the bias and minimizing the variance. The above decomposition reveals a source of the difference between explanatory and predictive modeling: In explanatory modeling, the focus is on minimizing bias to obtain the most accurate representation of the underlying theory. In contrast, predictive modeling seeks to minimize the combination of bias and estimation variance, occasionally sacrificing theoretical accuracy for improved empirical precision (Shmueli 2010).

These four aspects impact every step of the modeling process, such that the resulting f is markedly different in the explanatory and predictive contexts.

Let us assume that f is a reasonable operationalization of the true function (F) relating constructs X and Y. Choosing a function f^* that is intentionally biased in place of f is very undesirable from a theoretical–explanatory standpoint. However, the election of f^* is desirable to f under the prediction approach. We show this using the statistical model $y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon$, which is assumed to be correctly specified with respect to F. Using data, we obtain the estimated model $\widehat{f}$, which has the following properties:

$$\text{Bias} = 0$$

$$\text{Var}\left(\widehat{f}(\boldsymbol{x}_i)\right) = \text{Var}\left(\widehat{\beta}_1 x_1 + \widehat{\beta}_2 x_2 + \widehat{\beta}_3 x_3\right) = \sigma^2 \boldsymbol{x}^{\text{T}} \left(\boldsymbol{X}\boldsymbol{X}^{\text{T}}\right)^{-1} \boldsymbol{x},$$

where $\boldsymbol{x}$ is the vector $\boldsymbol{x} = [x_1, x_2, x_3]^{\text{T}}$ and $\boldsymbol{X}$ is the design matrix based on all predictors. Combining the squared bias with the variance gives, as expected, the prediction error (EPE).

$$E\left(y - \widehat{f}\{\boldsymbol{x}_i\}\right)^2 = \sigma^2 + 0 + \sigma^2 \boldsymbol{x}^{\text{T}} \left(\boldsymbol{X}\boldsymbol{X}^{\text{T}}\right)^{-1} \boldsymbol{x} = \sigma^2 \left[1 + \boldsymbol{x}^{\text{T}} \left(\boldsymbol{X}\boldsymbol{X}^{\text{T}}\right)^{-1} \boldsymbol{x}\right].$$

In comparison, consider the estimated underspecified form $\widehat{f}^*(\boldsymbol{x}_i) = x_1 \widehat{\gamma}$. The bias and variance here are

$$\begin{aligned}\text{Bias} = E\left(\widehat{f}\{\boldsymbol{x}_i\}\right) - f\{\boldsymbol{x}_i\} &= x_1 \left(x_1 x_1^{\text{T}}\right)^{-1} x_1^{\text{T}} (\beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3) \\ &\quad - (\beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3)\end{aligned}$$

$$\text{Var}\left(\widehat{f}(\boldsymbol{x}_i)\right) = \sigma^2 x_1 \left(x_1 x_1^{\text{T}}\right)^{-1} x_1^{\text{T}}$$

Combining the squared bias with the variance EPE is equal to

$$\begin{aligned}E\left(y - \widehat{f}\{\boldsymbol{x}_i\}\right)^2 &= \left[x_1 \left(x_1 x_1^{\text{T}}\right)^{-1} x_1^{\text{T}} (x_2\beta_2 + x_3\beta_3) - (x_2\beta_2 + x_3\beta_3)\right]^2 \\ &\quad + \sigma^2 \left[1 + x_1 \left(x_1 x_1^{\text{T}}\right)^{-1} x_1^{\text{T}}\right].\end{aligned}$$

Although the bias of the underspecified model $f^*(\boldsymbol{x}_i)$ is larger than that of $f\{\boldsymbol{x}_i\}$, its variance can be smaller, and in some cases, so small that the overall EPE will be lower for the underspecified model. Wu et al. (2007) showed the general result for an underspecified linear regression model with multiple predictors. In particular, they showed that the underspecified model that leaves out q predictors has a lower EPE when the following inequality holds:

$$q\sigma^2 > \beta_2^{\text{T}} X_2^{\text{T}} (I - H_1) X_2 \beta_2$$

This means that the underspecified model produces more accurate predictions, in terms of lower EPE, in the following situations: (a) when the data are very noisy (large σ^2); (b) when the true absolute values of the excluded parameters (in our example, β_2 and β_3) are small; (c) when the predictors are highly correlated; and (d) when the sample size is small or the range of left-out variables is small (Shmueli 2010).

Hagerty and Srinivasan (1991) nicely summarize this situation: "We note that the practice in applied research of concluding that a model with a higher predictive validity is "truer," is not a valid inference. This paper shows that a parsimonious but less true model can have a higher predictive validity than a truer but less parsimonious model."

4.3 Cross-validation

Cross-validation (CV) is a strategy for model selection or algorithm selection. CV consists of splitting the data (at least once) for estimating the error of each algorithm. Part of the data (the training set) is used for training each algorithm, and the remaining part (the testing set) is used for estimating the error of the algorithm. Then, CV selects the algorithm with the smallest estimated error. For this reason, CV is used to evaluate the prediction performance of a statistical machine learning model in out-of-sample data. This technique ensures that the data used for training the statistical machine learning model are independent of the testing data set in which the prediction performance is evaluated. It consists of repeating and recording the arithmetic average obtained from the evaluation measures on different partitions. Under k-fold CV, which is explained in greater detail later, this process is repeated a total of k times, with each of the k groups getting the chance to play the role of the test data, and the remaining $k - 1$ groups used as training data. In this way, we obtain k different estimates of the prediction error. As prediction performance is reported the average of these estimates of prediction error. CV is used in data analysis to validate the implemented models where the main objective is prediction and to estimate the prediction performance of a statistical learning model that will be carried out in practice. In other words, CV evaluates how well the statistical machine learning model generalized new data not used for training the model. The results of the CV largely depend on how the division between the training and testing sets is carried out. For this reason, in the following sections, we provide the more popular types of CV used in the implementation of statistical learning models.

4.3.1 The Single Hold-Out Set Approach

The single hold-out set or validation set approach consists of randomly dividing the available data set into a training set and a validation or hold-out set (Fig. 4.3). The statistical machine learning model is trained with the training set while the hold-out

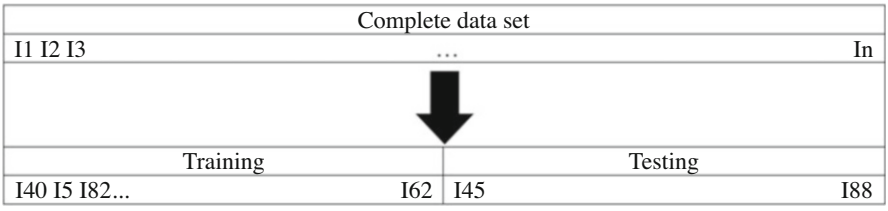

Fig. 4.3 Schematic representation of the hold-out set approach. A set of observations are randomly split into a training set with individuals I40, I5, I82, among others, and into a testing set with observations I45, I88, among others. The statistical machine learning model is fitted on the training set and its performance is evaluated on the validation set (James et al. 2013)

set (testing set) is used to study how well that statistical machine learning model performs on unseen data. For example, 80% of the data can be used for training the model and the remaining 20% of the data for testing it. One weakness of the hold-out (validation) set approach is that it depends on just one training-testing split and its performance depends on how the data are split into the training and testing sets.

4.3.2 *The* k-*Fold Cross-validation*

In k-fold CV, the data set is randomly divided into k complementary folds (groups) of approximately equal size. One of the subsets is used as testing data and the rest $(k - 1)$ as training data. Then $k - 1$ folds are used for training the statistical machine learning model and the remaining fold for evaluating the out-of-sample prediction performance. For these reasons, the statistical machine learning model is fitted k times using a different partition (fold) as the testing set and the remaining $k - 1$ as the training set. Finally, the arithmetic mean of the k folds is obtained and reported as the prediction performance of the statistical machine learning model (see Fig. 4.4). This method is very accurate because it combines k measures of fitness resulting from the k training and testing data sets into which the original data set was divided, but at the cost of more computational resources. In practice, the choice of the number of folds depends on the measurement of the data set, although 5 or 10 folds are the most common choices.

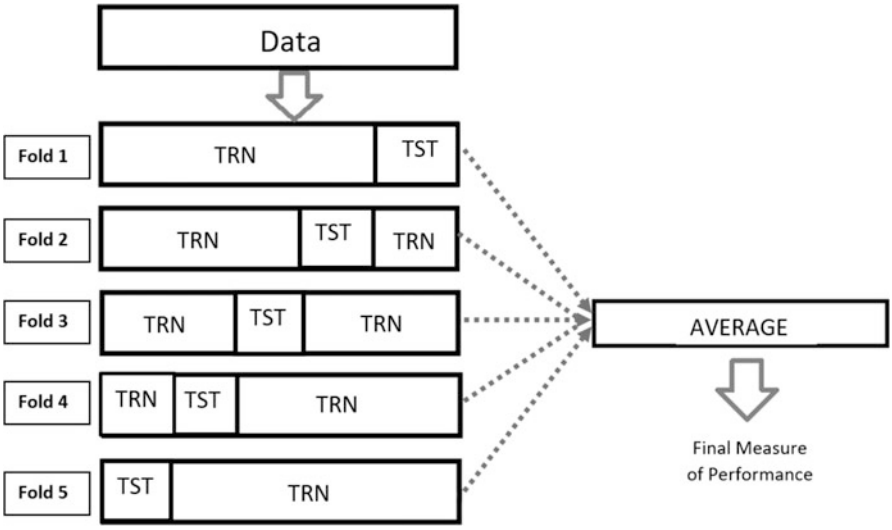

Fig. 4.4 Schematic representation of the k-fold cross-validation with complementary subsets with $k = 5$

It is important to point out that to reduce variability, we recommend implementing the *k*-fold CV multiple times, each time using different complementary subsets to form the folds; the validation results are combined (e.g., averaged) over the rounds (times) to give a better estimate of the statistical machine learning model predictive performance.

4.3.3 The Leave-One-Out Cross-validation

The Leave-One-Out (or LOO) CV is very simple since each training data set is created by including all the individuals except one, while the testing set only includes the excluded individual. Thus, for n individuals in the full data set, we have n different training and testing sets. This CV scheme wastes minimal data, as only one individual is removed from the training set.

Regarding the *k*-fold cross-validation that was just explained, n models are built from n individuals (samples) instead of k models, where $n > k$. Moreover, each model is trained on $n - 1$ samples rather than $(k - 1)n/k$. In both cases, since k is normally not too large and $k < n$, LOO is more computationally expensive than *k*-fold cross-validation. In terms of prediction performance, LOO normally produces high variance for the estimation of the test error. Because $n - 1$ of the n samples is used to build each statistical machine learning model, those constructed from folds are virtually identical to each other and to the model built from the entire training set. However, when the learning curve is steep for the evaluated training size, then five- or ten-fold cross-validation usually overestimates the generalization error.

Learning curves (LC) are considered effective tools to monitor the performance of the employee exposed to a new task. LCs provide a mathematical representation of the learning process that takes place as the task is repeated. In statistical machine learning the LC is a line plot of learning (y-axis) over experience (x-axis). Learning curves are extensively used in statistical machine learning for algorithms that learn (their parameters) incrementally over time, such as deep neural networks. In general, there is considerable empirical evidence suggesting that five- or ten-fold cross-validation should be preferred to LOO.

4.3.4 The Leave-m-Out Cross-validation

The Leave-m-Out (LmO) CV is very similar to LOO as it creates all the possible training/test sets by removing m samples from the complete set. For n samples, this produces $\binom{n}{m}$ train-test pairs. Unlike LOO and *k*-fold, the test sets will overlap for $m > 1$. For example, in a Leave-2-Out CV with a data set with four samples (I1, I2,

I3, and I4), the total number of training-testing sets is equal to $\binom{4}{2} = 6$; this means that the six testing sets are: [I3, I4] [I1, I2], [I2, I4] [I1, I3], [I2, I3] [I1, I4], [I1, I4] [I2, I3],[I1, I3] [I2, I4], and [I1,I2] [I3, I4], while the training sets are the complementary elements of each testing set.

4.3.5 Random Cross-validation

In this type of CV, the number of partitions (independent training-testing data set splits) is defined by the user, and more partitions are better. Each partition is generated by randomly dividing the whole data set into two subsets: the training (TRN) data set and the testing (TST) data set. The percentage of the whole data set assigned to the TRN and TST data sets is also fixed by the user. For example, for each random partition, the user can decide that 80% of the whole data set can be assigned to the TRN data set and the remaining 20% to the TST data set. Random cross-validation is different from *k*-fold cross-validation because the partitions are not mutually exclusive; this means that in the random cross-validation approach, one observation can appear in more than one partition. Consequently, some samples cannot be evaluated, whereas others can be evaluated more than once, meaning that the testing and training subsets can be superimposed (Montesinos-López et al. 2018a, b). To control the randomness for reproducibility, we recommend using a specific seed in the random number generator. We recommend using at least ten random partitions to obtain enough accuracy in the estimate of prediction performance.

4.3.6 The Leave-One-Group-Out Cross-validation

The Leave-One-Group-Out (LOGO) CV is useful when individuals are grouped (in environments or years, or even another criterion), where the number of groups (g) is at least two and the information of $g - 1$ groups are used as the training set while all individuals of the remaining group are used as the testing set. For example, in the context of genomic selection, when the plant breeder is interested in predicting these lines in another environment, the same (or different) lines were frequently evaluated in g environments or years, that represent the groups. Jarquín et al. (2017) denotes this type of CV strategy as CV1 in the context of plant breeding. Under this approach, the predictions are reported for each of the g groups because the scientist is interested in the prediction performance of each environment. Many times, the groups are the years under study and the aim is to predict the information of a complete year. However, when the groups are years and if we suspect that there is a considerable correlation between observations that are near in time. Therefore, it is

Fold 1	TRN	TST			
Fold 2	TRN		TST		
Fold 3	TRN			TST	
Fold 4	TRN				TST
	Year 1	Year 2	Year 3	Year 4	Year 5

Fig. 4.5 Schematic representation of time series data with 5 years, using the previous year for predicting the next year. However, in some practical applications, we are not interested in going too far back in time since the training and testing sets will be less related the farther you go

imperative to evaluate our statistical machine learning model for time series data on "future" observations. In this sense, the training sets are composed of the previous years to predict the subsequent year. This method can also be seen as a variation of k-fold CV, where the first folds are used for training the statistical machine learning model and the fold $(k + 1)$ is the corresponding testing set. The main difference in this CV method is that successive training sets are supersets of those that come before them. Also, it adds all surplus data to the first training partition, which is always used to train the model (see Fig. 4.5).

Figure 4.5 shows that under this type of CV, for time series data, the predictions are for $g - 1$ years; individuals from the first year are not predicted since a training set is not available.

4.3.7 Bootstrap Cross-validation

First, we will define the bootstrapping method to understand how it is used in the CV approach, which should then be straightforward. Bootstrapping is a type of resampling method where, for example, $B = 10$ samples of the same size are repeatedly drawn, with replacement, from a single original sample. Afterward, each of these B samples is used to estimate statistics (for example, the mean, variance, median, minimum, etc.) of a population, and the average of all the B sample estimates of the target statistic is reported as the final estimate. In the context of statistical machine learning, these samples are used to evaluate the prediction performance of the algorithm under study for unseen data. One important difference between this CV approach and all the procedures explained above is that now the training set has the same size (number of observations) as the original sample because the bootstrap method replaced some individuals more than once. According to Kuhn and Johnson (2013), as a result, some observations will be represented multiple times in the bootstrap sample, while others will not be selected at all; those observations not selected are referred to as the testing set, however, this CV strategy is quite different than the previously explained. Efron (1983) pointed out that the prediction performance of the bootstrap samples tends to have less uncertainty than the k-fold cross-validation since on average, 63.2% of the data points are represented (for training) at least once in any sample size. For this reason,

Fold	Training	Testing
1	I7, I4, I6, I9, I2, I3, I4, I4, I8, I6, I8, I7	I1, I5, I10, I11, I12
2	I2, I8, I5, I6, I1, I4, I5, I11, I1 1,I8,I10, I5	I3, I7, I9, I12
3	I5, I9, I11, I3, I10, I5, I7, I2, I 3, I11, I6, I9	I1, I4, I8, I12
4	I10, I12, I9, I7, I4, I3, I1, I9, I3, I2, I1, I6	I5, I8, I11
5	I2, I10, I5, I12, I3, I6, I3, I7, I6, I6, I5, I7	I1, I4, I8, I9

Fig. 4.6 Schematic representation of bootstrap cross-validation

this CV approach has a bias similar to implementing a $k =$ two fold cross-validation, and as the training set becomes smaller, the bias becomes more problematic. To understand this CV method, we provide a simple example of how the training and testing samples are constructed. If we have a sample with 12 individuals denoted as I1, I2, …, I12, we will select $B = 5$ bootstrap samples. Each bootstrap sample is obtained with replacement and the individuals that appear in each one correspond to the training sample; those that are not present will correspond to the testing set. Figure 4.6 provides the five bootstrap samples; each training sample has the same size as the original, however, only some individuals appear in each bootstrap sample, while those individuals that do not appear are included in the testing set. For example, in the first fold, the training bootstrap sample contains seven different individuals (I2, I3, I4, I6, I7, I8, and I9), while the testing set contains five individuals (I1, I5, I10, I11, and I12). It is important to point out that since the training sample has the same size as the original sample, some individuals in the training sample are repeated at least twice; in the first fold, the individual I4 are repeated three times, whereas I6, I7, and I8 are repeated twice. Finally, similar to other methods, the statistical machine learning model is trained with each training set; likewise, the prediction performance of the model is evaluated in each testing set. The average of these sample predictions is reported as the estimated testing error.

4.3.8 Incomplete Block Cross-validation

Incomplete block (IB) CV should be used when there are J treatments evaluated in I blocks and the same treatments are evaluated in all the blocks. The idea behind this

Table 4.1 TRN data set for $I = 3$ blocks and $J = 10$ treatments with 70% of data for training

Block	Treatments per block						
1	1	3	5	7	8	9	10
2	2	3	4	5	6	7	9
3	1	2	4	6	7	8	10

CV method is that some treatments should be present in some blocks but absent in others, whereas the same treatment should be present in at least one environment (block). The theory of incomplete block designs developed in the experimental designs statistical area can be used to construct the training set. For example, under a balanced incomplete block (BIB) design, the term incomplete means that all treatments in each block cannot be evaluated, whereas balanced means that each pair of treatments occur together λ times. The training set is constructed by first defining the % of individuals in the TRN set using the equation $sI = Jr = N_{\text{TRN}}$, where J represents the number of treatments under study, I represents the number of blocks under study, r denotes the number of repetitions of each treatment, and s denotes the treatments per block. For example, suppose that we had $J = 10$ treatments and $I = 3$ blocks (that is, 30 individuals), and we decided to use $N_{\text{TRN}} = 21$ (70%) of the total individuals in the TRN set. Therefore, the number of treatments by block can be obtained by solving $(sI = N_{\text{TRN}})$ for s, which results in $s = N_{\text{TRN}}/I$. This means that $s = 21/3 = 7$ treatments per block. Then, the corresponding elements for the training set can be obtained with the function find.BIB(10, 3, 7) of the package crossdes of the R statistical software. The numbers used in the function find.BIB() denote the treatments, the blocks, and the treatments per block, respectively. Finally, the treatments that make up the TRN set are shown in Table 4.1.

According to Table 4.1, it is clear that each treatment is present in two blocks and missing in one block. For example, in Block 1 the testing set includes treatments 2, 4, and 6; in Block 2, the testing set is composed of treatments 1, 8, and 10; and in Block 3, the testing set is composed of treatments 3, 5, and 9.

4.3.9 Random Cross-validation with Blocks

Random cross-validation with blocks was proposed by Lopez-Cruz et al. (2015) and belongs to the so-called replicated TRN-TST cross-validation that appears in the publication of Daetwyler et al. (2012), since some individuals can never be part of the training set. This algorithm, like the incomplete block cross-validation, is appropriate when we are interested in evaluating J lines in I blocks or environments and tries to mimic a prediction problem faced by breeders in incomplete field trials where lines are evaluated in some, but not all, target environments. The algorithm for constructing the TRN-TST sets is described by the following steps: Step 1. Calculate the total number of observations under study as $N = J \times I$; Step 2. Define the proportion of observations used for training and testing, that is, P_{TRN} and P_{TST}; Step 3. Calculate the size of the testing set $N_{\text{TST}} = N \times P_{\text{TST}}$; Step 4. Choose N_{TST} lines at

Table 4.2 TRN-TST data sets for $J = 10$ lines and $I = 3$ environments

Environment	Lines									
	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10
1	TRN	TST	TRN	TRN	TST	TRN	TRN	TST	TRN	TRN
2	TST	TRN	TRN	TRN	TRN	TRN	TST	TRN	TST	TRN
3	TRN	TRN	TST	TST	TRN	TRN	TRN	TRN	TRN	TST

random without replacement if $J \geq N_{TST}$, and with replacement otherwise; Step 5. Each chosen line will then be assigned to one of the I environments chosen at random without replacement; Step 5. All the selected lines and environments will form the training set, while the lines and environments that were not chosen will form the corresponding testing set; and Step 6. Steps 1–5 are repeated depending on the number of TRN-TST partitions required (Lopez-Cruz et al. 2015). This CV is called CV2 in Jarquín et al. (2017). Next, we assume that we have ten lines or treatments and three environments or blocks that will form the corresponding training testing sets for only one partition: Step 1. The total number of observations under study is $N = 10 \times 3 = 30$; Step 2. We define $P_{TRN} = 0.7$ and $P_{TST} = 0.3$; Step 3. The size of the testing set is $N_{TST} = 30 \times 0.3 = 9$; Step 4. Since $J = 10 \geq N_{TST} = 9$, we selected the following lines at random without replacement: L1, L2, L3, L4, L5, L6, L7, L8, L9, and L10; and Step 5. Each chosen line was assigned to one of the $I = 3$ environments randomly chosen without replacement, as shown in Table 4.2. It is important to point out that this CV strategy only differs from the incomplete block cross-validation in the way the lines are allocated to blocks.

4.3.10 Other Options and General Comments on Cross-validation

It is important to highlight that when the data set is considerably large, it is better to randomly split it into three parts: a training set, a validation set (or tuning set), and a testing set. The training set and testing set are used as explained before, while the validation (tuning set) set is used to estimate the prediction error for model selection, which is the process of estimating the performance of different models in order to choose the best one, or to evaluate the chosen statistical machine learning model with a range of values of tuning hyperparameters to select the combination of hyperparameters with the best prediction performance and then use these hyperparameters (or best model) to evaluate the prediction performance in the testing set (Fig. 4.7). It is important to point out that Fig. 4.7 shows only one random split of the data in terms of the training, testing, and validation sets.

For example, assume that our data set has 50,000 rows (observations) and that we have decided to use 5000 of them for the testing set and another 5000 for the validation set. This means that 40,000 rows are left for the training set. At this point, we train our statistical machine learning model with each component of the

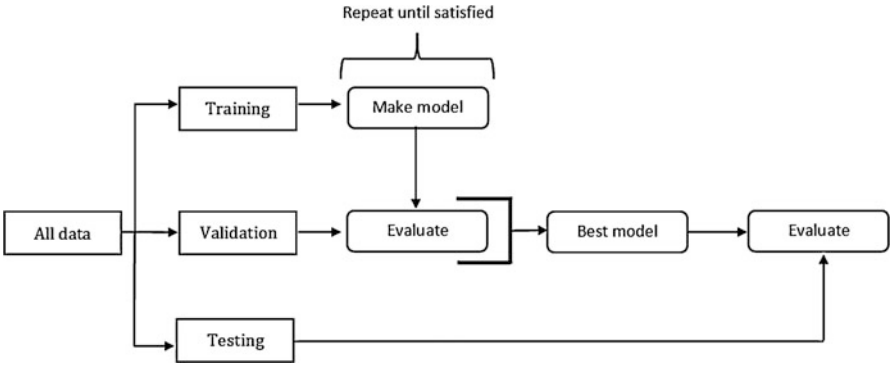

Fig. 4.7 Schematic representation of the training, validation (tuning), and testing sets proposed by Cook (2017)

grid of hyperparameters of the training set and evaluate the prediction performance on the validation set, as shown in the middle of Fig. 4.7. Finally, we will pick the best model (best subset of hyperparameters) in terms of prediction performance in the validation set and we are ready to evaluate the prediction performance in the testing set. Then, we will report the testing error on the testing set, as can be observed at the bottom of Fig. 4.7 (Cook 2017). Under this approach, the testing and validation sets have approximately the same size to guarantee a similar out-of-sample prediction performance. Since it is difficult to give general rules on how to choose the number of observations in each of the three parts, a typical number might be 50% for training, and 25% each for validation and testing (Hastie et al. 2008) or 70%, 15%, and 15% for training, validation, and testing, respectively. Another way to find the optimal setting of hyperparameters using the grid search, which is very common in deep learning, consists of picking values for each parameter from a finite set of options [e.g., number of epochs (100, 150, 200, 250, 300); batch sizes (25, 50, 75, 100, 125); number of layers (1, 2, 3, 4, 5); and types of activation functions (RELU, Sigmoid), ...] and training the statistical machine learning model with every permutation of hyperparameter choices using the training set. Then, the combination of hyperparameters with the best prediction performance on the validation set is chosen, and we report the prediction performance of the best selected model (set of hyperparameters) in the testing set (Buduma 2017). The aforementioned examples use the validation data set as a proxy measure of the accuracy during the hyperparameter optimization process. However, it can also be used as a proxy measure of the accuracy for model selection, and instead of using a grid of hyperparameters, we can use a set of different statistical machine learning models; here, the best model is chosen instead of the best combination of hyperparameters. It is important to understand that when you have more than one random partition (training-testing-validation), as shown in Fig. 4.8, the same process provided in Fig. 4.7 is followed, however, the average of all the partitions is reported as a measure of prediction performance. Also, if more precision is required in the estimated prediction performance, you can repeat the process given in Fig. 4.8

Partition 1	TRN						TST	VAL
Partition 2	TRN				TST		VAL	TRN
Partition 3	TRN		TST		VAL		TRN	
Partition 4	TST		VAL		TRN			
Partition 5	VAL		TRN					TST

Fig. 4.8 Five partitions of training-validation-testing

multiple times and report the average of all repetitions as a measure of prediction performance.

As mentioned above, this approach (Figs. 4.7 and 4.8) is used for a large data set however, in the case of a smaller data set, we suggest modifying this approach to avoid wasting too much training data in validation sets. This modification consists of performing an inner cross-validation approach since the training set is split into complementary subsets, and each model is trained against a different combination of these subsets, which is subsequently validated against the remaining parts. Once the model hyperparameters have been selected, a final model is trained with the whole training set (refitted) and the generalized prediction performance is measured on the testing set. This approach was applied by Montesinos-López et al. (2018a, b), who also split the original data into training and testing sets. Subsequently, each training set was split again and 80% of the data was used for training a grid of hyperparameters while the remaining 20% was used for validating (tuning) the prediction performance and selecting the best combination of hyperparameters with the best prediction performance. At this point, the deep learning algorithm was refitted with the whole training set, and with this they evaluated the out-of-sample prediction. They called the conventional training-testing partition outer cross-validation, while the split performed in each training set used for hyperparameter tuning was called inner cross-validation. It should be highlighted that there are no differences between outer and inner CV and training-validation-test. Finally, it should also be mentioned that any type of the cross-validation strategies mentioned in this section (random CV, *k*-fold CV, Bootstrap CV, IB CV, etc.,) can be used in both outer and inner CV, such as is the case with the five-fold CV.

4.4 Model Tuning

A hyperparameter is a parameter whose value is set before the learning process begins. Hyperparameters govern many aspects of the behavior of statistical machine learning models, such as their ability to learn features from data, the models' exhibited degree of generalizability in performance when presented with new data, as well as the time and memory cost of training the model, since different hyperparameters often result in models with significantly different performance. This means that tuning hyperparameter values is a critical aspect of the statistical machine learning training process and a key element for the quality of the resulting

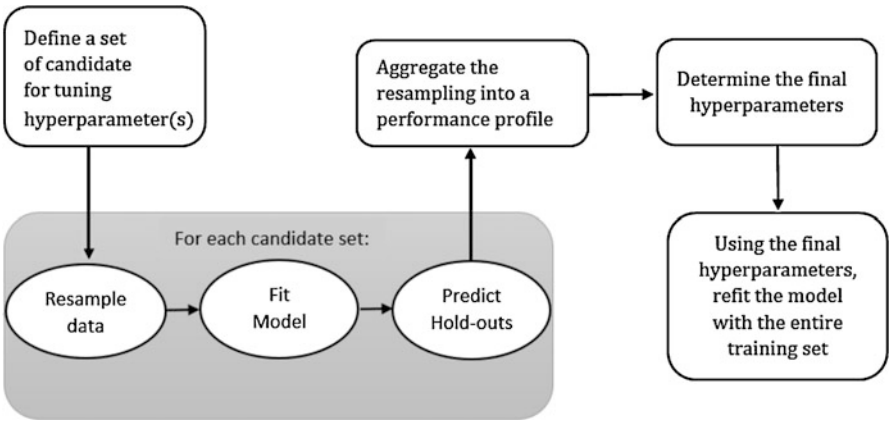

Fig. 4.9 Schematic representation of the tuning process proposed by Kuhn and Johnson (2013)

prediction accuracies. However, choosing appropriate hyperparameters is challenging (Montesinos-López et al. 2018a). Hyperparameter tuning finds the best version of a statistical machine learning model by running many training sets on the original data set using the algorithm and ranges of values of hyperparameters as specified. The hyperparameter values that provide the best performance in out-of-sample prediction evaluated by the chosen metric are then selected.

There are many ways of searching for the best hyperparameters. However, a general approach defines a set of candidate values for each hyperparameter. Each value of this set of candidate values is then applied with a resample of the training set of the chosen statistical machine learning method, where we aggregate all the hold-out predictions from which the best hyperparameters are chosen and refit the model with the entire set (Kuhn and Johnson 2013). A schematic representation of the tuning process proposed by Kuhn and Johnson (2013) is given in Fig. 4.9. It is important to highlight that this process should be performed correctly because when the same data are used for training and evaluating the prediction performance, the prediction performance obtained is extremely optimistic.

For example, suppose a breeder is interested in developing an algorithm to classify unseen plants as diseased or not diseased with an available training data set. The goal is to minimize the rate of misclassification or to maximize the percentage of cases correctly classified (PCCC). Also, assume that you are new to the world of statistical machine learning and that you only understand the k-nearest neighbor method. Since this algorithm depends only on the hyperparameter called the number of neighbors (k), the question is which value of k to choose in such a way that the prediction performance of this algorithm will be the best in the sample prediction of plants. To find the best value of the k hyperparameter, you must specify a range of values for k (for example, from 1 to 60 with increments of 1), then with a part of your training data set, called the training-inner (or tuning that corresponds to the training data in the inner loop) set, which is randomly selected. You proceed to evaluate the 60 values of k with the k-nearest neighbor method and evaluate the

prediction performance in the remaining part of the training set (validation set). Next, you select the value of k from this range of values that best predicts (according, for example, to the PCCC) out-of-sample data (validation set) and use this value to perform the prediction of the unseen plants not used for training the model (testing set). This is a widely adopted practice that consists of searching for the parameter (usually through brute force loops) that yields the best performance over a validation set. However, the process illustrated here is very simple because the k-nearest neighbor model only depends on a unique hyperparameter; however, there are other statistical machine learning algorithms (for example, deep learning methods) where the tuning process is required for a considerable amount of hyperparameters. For this reason, we encourage caution when choosing the statistical machine learning algorithm, since the amount of work required for performing the tuning process depends on the chosen method.

4.4.1 Why Is Model Tuning Important?

Tuning the hyperparameters of the models is a key element to optimize your statistical machine learning model to perform well in out-of-sample predictions. The tuning process is more an art than a science because there is no unique formal scientific procedure available in the literature. Nowadays, the tuning process is trial and error that consists of implementing the statistical machine learning model many times with different values of the hyperparameters and then comparing its performance on the validation set in order to determine which set of hyperparameters results in the most accurate model; for the final implementation, the set of hyperparameters of the best model is used. As mentioned above, for the k-nearest neighbor classifier, we need to choose the number of neighbors (k) using the tuning process to obtain the optimal prediction performance of this algorithm, while for conventional Ridge regression, the parameter lambda (λ) is obtained by tuning to improve the out-of-sample predictions. These two statistical machine learning algorithms that we just mentioned need only one hyperparameter; however, other statistical machine learning methods may require more hyperparameters, as exemplified by deep learning models that require at least three hyperparameters (number of neurons, number of hidden layers, type of activation function, batch size, etc.). After tuning the required hyperparameters, the statistical machine learning model learns the parameters from the data to be used for the final prediction of the testing set. The choice of hyperparameters significantly influences the time required to train and test a statistical machine learning model. Hyperparameters can be continuous or of the integer type; for this reason, there are mixed-type hyperparameter optimization methods.

4.4.2 Methods for Hyperparameter Tuning (Grid Search, Random Search, etc.)

Manual tuning of statistical machine learning models is of course possible, but relies heavily on the user's expertise and understanding of the underlying problem. Additionally, due to factors such as time-consuming model evaluations, nonlinear hyperparameter interactions in the case of large models that consist of tens or even hundreds of hyperparameters, manual tuning may not be feasible since it is equivalent to brute force. For this reason, the four most common approaches for hyperparameter tuning reported in the literature are (a) grid search, (b) random search, (c) Latin hypercube sampling, and (d) optimization (Koch et al. 2017).

In the *grid search* method, each hyperparameter of interest is discretized into a desired set of values to be studied where the models are trained and assessed for all combinations of the values across all hyperparameters (that is, a "grid"). Although fairly simple and straightforward to carry out, a grid search is appropriate when there are only a few values for a limited number of hyperparameters. However, although this is a comprehensive way of assessing different hyperparameter values, when there are many values for some or many hyperparameters, it quickly becomes quite costly due to the number of hyperparameters and the number of discrete levels of each. For example, in Ridge regression, this approach is implemented as follows: since λ is the hyperparameter to be tuned, we first propose, for example, a grid of 100 values for this hyperparameter from $\lambda = 10^{10}$ to $\lambda = 10^{-2}$; then we divide the training set into five inner training sets and five inner testing (tuning) sets, where each of the 100 values of the grid is fitted using the inner training sets and the testing error is evaluated with the inner testing sets. Then we get the average predicted test error and pick one value out of the 100 values of the grid that produces the best prediction performance. Next, we refit the statistical machine learning method to the whole training set using the picked value of λ, and finally perform the predictions for the testing set using the learned parameters of the training set with the best picked value of λ. In all the models with one hyperparameter, it is practical to implement the *grid search* method, but for example, in deep learning models, which many times require six hyperparameters to be tuned, if only three values are used for each hyperparameter, there are $3^6 = 729$ combinations that need to be evaluated, quickly becoming computationally impracticable.

A *random search* differs from a *grid search* in that rather than providing a discrete set of values to explore each hyperparameter, we determine a statistical distribution for each hyperparameter from which values may be randomly sampled. This affords a much greater chance of finding effective values for each hyperparameter. While Latin hypercube sampling is similar to the previous method, it is a more structured approach (McKay 1992) since it is an experimental design in which samples are exactly uniform across each hyperparameter but random in combinations. These so-called low-discrepancy point sets attempt to ensure that points are approximately equidistant from one another in order to fill the space

efficiently. This sampling supports coverage across the entire range of each hyperparameter and is more likely to find good values of each hyperparameter.

The previous two methods for hyperparameter tuning are used to perform individual experiments by building models with various hyperparameter values and recording the model performance for each. Because each experiment is performed in isolation, this process is parallelized, but is unable to use the information from one experiment to improve the next experiment. Optimization methods, on the other hand, consist of sequential model-based optimization where the results of previous experiments are used to improve the sampling method of the next experiment. These methods are designed to make intelligent use of fewer evaluations and thus save on the overall computation time (Koch et al. 2017). Optimization algorithms that have been used in statistical machine learning generally for hyperparameter tuning include Broyden–Fletcher–Goldfarb–Shanno (BFGS) (Konen et al. 2011), covariance matrix adaptation evolution strategy (CMA-ES) (Konen et al. 2011), particle swarm (PS) (Renukadevi and Thangaraj 2014), tabu search (TS), genetic algorithms (GA) (Lorena and de Carvalho 2008), and more recently, surrogate-based Bayesian optimization (Dewancker et al. 2016). Also, recently the use of the response surface methodology has been explored for tuning hyperparameters in random forest models (Lujan-Moreno et al. 2018). However, the implementation of these optimization methods is not straightforward because it requires expensive computation; also, software development is required for implementing these algorithms automatically. There have been advances in this direction for some machine learning algorithms in the statistical analysis system (SAS), R and Python software (Koch et al. 2017). An additional challenge is the potential unpredictable computation expense of training and validating predictive models using different hyperparameter values. Finally, although it is challenging, the tuning process often leads to hyperparameter settings that are better than the default values, as it provides a heuristic validation of these settings, giving greater assurance that a model configuration with a higher accuracy has not been overlooked.

4.5 Metrics for the Evaluation of Prediction Performance

The quality of prediction performance of any statistical machine learning method in a given data set consists of evaluating how close the predicted values are to the true observed ones. In other words, the prediction performance quantifies the matching degree between the predicted response value for a given observation and the true response value for that observation (James et al. 2013). However, the metrics used for quantifying the prediction performance depend on the type of response variable under study; for this reason, we subsequently give the most popular metrics used for this goal for four types of response variables.

4.5.1 Quantitative Measures of Prediction Performance

Before implementing a statistical machine learning model, we assume that we have our training observation $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ and we estimate f, as $\widehat{f}$, with the chosen statistical machine learning model. Then we can make predictions for each of the response values (y_i) with $\widehat{f}(x_i)$ and compute the predicted values for each of the n observations in the training set; with these values we can calculate the ***mean square error*** (MSE) for the training data set as $E = \frac{1}{n}\left(\sum_{i=1}^{n}\left(y_i - \widehat{f}(x_i)\right)\right)^2$; however, what we really want to predict are the values for unseen test observations that were not used to train the statistical machine learning model. Assuming that the unseen testing set is equal to $\{(x_{n+1}, y_{n+1}), (x_{n+2}, y_{n+2}), \ldots, (x_{n+T}, y_{n+T})\}$, the MSE for the testing data set should be calculated as

$$\mathrm{MSE}_{\mathrm{TST}} = \frac{1}{T}\sum_{i=n+1}^{n+T}\left(y_i - \widehat{f}(x_i)\right)^2, \tag{4.1}$$

where $\widehat{f}(x_i)$ is the prediction that $\widehat{f}$ gives to the ith observation. The $\mathrm{MSE}_{\mathrm{TST}}$ with a lower value will have better predictions, which means that the predicted values are very close to the true observed values. Also, the square root of $\mathrm{MSE}_{\mathrm{TST}}$ can be used as a measure of prediction performance and is called root mean square error (RMSE).

Pearson's correlation coefficient is a very popular measure of prediction performance in plant breeding and can be calculated as

$$r_{\mathrm{TST}} = \frac{\sum_{i=n+1}^{n+T}\left(\widehat{f}(x_i) - \overline{\widehat{f}(x_i)}\right)(y_i - \overline{y}_i)}{\sqrt{\sum_{i=n+1}^{n+T}\left(\widehat{f}(x_i) - \overline{\widehat{f}(x_i)}\right)^2}\sqrt{\sum_{i=n+1}^{n+T}(y_i - \overline{y}_i)^2}}, \tag{4.2}$$

where $\overline{\widehat{f}(x_i)}$ is the average of the T predictions that conform to the testing set, and $\overline{y}_i$ is the average of the T true observed values. In this case, the closer that the predictions are to 1, the better the implemented statistical machine learning model will perform. It is important to point out that Pearson's correlation is defined between -1 and 1. However, to be convinced that the observed and predicted values match, it is a common practice to perform a scatter plot of predicted versus observed (or vice versa) values and when the observed and predicted values follow a straight line (45° diagonal line) from the bottom left corner to the top right corner, this indicates a perfect match between the observed and predicted values. For this reason, Pearson's correlation as a metric should be complemented with the intercept and slope, since the slope and intercept describe the consistency and the model bias, respectively (Smith and Rose 1995; Mesple et al. 1996). To obtain the slope and intercept, the observed values (as y) versus the predicted values (as x) are regressed and in addition

to Pearson's correlation, these values should also be reported (slope and intercept). The expected intercept should be zero and the slope 1, if the correlation obtained between the observed and predicted values is high. It is important to avoid carrying out the regression in the opposite way, i.e., using predicted values as y's, and observed values as x's, since this leads to incorrect estimates of the slope and the y-intercept. This denotes that a spurious effect is added to the regression parameters when regressing predicted versus observed values and comparing them against the 1:1 line. The user should also remember that underestimation of the slope and overestimation of the y-intercept increase as Pearson's correlation values decrease. We strongly recommend that scientists evaluate their models by regressing observed versus predicted values and test the significance of slope $= 1$ and intercept $= 0$ (Piñeiro et al. 2008). Finally, it is important to recall that the square of Pearson's correlation can also be used as a metric for measuring prediction performance since it represents the proportion of the total variance explained by the regression model and is called the coefficient of determination denoted as R^2.

Next, we present the mean absolute error (MAE) metric that measures the difference between two continuous variables (observed and predicted). The MAE can be calculated with the following expression:

$$\mathrm{MAE}_{\mathrm{TST}} = \frac{1}{T}\sum_{i=n+1}^{n+T}\left|y_i - \widehat{f}(x_i)\right| \tag{4.3}$$

Below we present another metric used to evaluate the prediction performance of any statistical machine learning model; it was proposed by Kim and Kim (2016) and is called ***mean arctangent absolute percentage error*** (MAAPE) that is calculated as follows:

$$\mathrm{MAAPE}_{\mathrm{TST}} = \frac{\sum_{i=n+1}^{n+T}\arctan\left(\left|\frac{y_i - \widehat{f}(x_i)}{y_i}\right|\right)}{T} \tag{4.4}$$

Although MAAPE is finite when the response variable (i.e., $y_i = 0$) equals zero, since it has a satisfactory trigonometric representation. However, because MAAPE's value is expressed in radians, it is less intuitive, in addition to being scale-free. This metric is a modification of the mean absolute percentage error (MAPE) which is problematic because it is undefined when the response variable is equal to zero ($y_i = 0$). MAAPE is also asymmetric since division by zero is defined and is not a problem. It is important to point out that there are other metrics for measuring prediction accuracy for continuous data, but we only presented the most popular ones.

The distinction between the training and test MSE is important since we are not interested in how well the statistical machine learning method performs in the training data set, due to the fact that our main goal is to perform accurate predictions in the unseen test data. For example, a plant breeder may be interested in developing an algorithm to predict disease resistance of a plant in new environments based on

records that were collected in a set of environments. We can train the statistical machine learning method with the information collected in the set of environments, but the interest is not in how well the statistical machine learning method predicts in those previously collected environments. An environmental scientist can also be interested in predicting the average annual rainfall in a municipality in Mexico, using data from the last 20 years to train the model. Such measures could include sea surface temperature, time, the yearly rotation of the earth, among others. In this case, the scientist is really interested in predicting the average rainfall of the next 1 or 2 years, not in accurately predicting the years measured in the training set.

4.5.2 *Binary and Ordinal Measures of Prediction Performance*

The binary and ordinal response variables are very common in classification problems, where the goal is to predict which category something falls into. An example of a classification problem is analyzing financial data to determine if a client will be granted credit or not. Another example is analyzing breeding data to predict if an animal is at high risk for a certain disease or not. Below, we provide some popular metrics to evaluate the prediction performance of this type of data.

The first metric is called a ***confusion matrix***, which is a tool to visualize the performance of a statistical machine learning algorithm that is used in supervised learning for classifying categorical and binary data. Each column of the matrix represents the number of predictions in each class, while each row represents the instances in the real class. One of the benefits of confusion matrices is that they make it easy to determine whether the system is confusing classes. Table 4.3 shows a sample format of a confusion matrix of C classes.

With Eqs. (4.5–4.8) we can calculate the total number of false negatives (TFN), false positives (TFP), true negatives (TTN) for each class i, and the total true positives in the system, respectively:

$$\text{TFN}_i = \sum_{\substack{j=1 \\ j \neq i}}^{C} n_{ij} \tag{4.5}$$

Table 4.3 Confusion matrix with more than two classes

		Predicted values			
		Class 1	Class 2	...	Class C
Observed values	Class 1	n_{11}	n_{12}	...	n_{1C}
	Class 2	n_{21}	n_{22}	...	n_{2C}
	⋮	⋮	⋮	⋮	⋮
	Class C	n_{C1}	n_{C2}	...	n_{CC}

$$\mathrm{TFP}_i = \sum_{\substack{j=1 \\ j \neq i}}^{C} n_{ji} \tag{4.6}$$

$$\mathrm{TTN}_i = \sum_{\substack{j=1 \\ j \neq i}}^{C} \sum_{\substack{k=1 \\ k \neq i}}^{C} n_{ji} \tag{4.7}$$

$$\mathrm{TTP}_{\mathrm{all}} = \sum_{j=1}^{C} n_{jj} \tag{4.8}$$

Below, we define the sensitivity (S_e), precision (P), and specificity (S_p). The sensitivity indicates the ability of our statistical learning algorithm to determine the proportion of true positives that are correctly identified by the test. The precision is the proportion of correct classification of our statistical machine learning model and represents the proportion of cases correctly classified, while the specificity is the ability of our statistical machine learning model to classify the true negative cases, that is, the specificity is the proportion of true negatives that are correctly identified by the test. Under the "one-versus-all basis," where each category is compared with the composed information of the remaining categories, we provide the expressions for computing the generalized precision, sensitivity, and specificity for each class i:

$$P_i = \frac{\mathrm{TTP}_{\mathrm{all}}}{\mathrm{TTP}_{\mathrm{all}} + \mathrm{TFP}_i} \tag{4.9}$$

$$S_{ei} = \frac{\mathrm{TTP}_{\mathrm{all}}}{\mathrm{TTP}_{\mathrm{all}} + \mathrm{TFN}_i} \tag{4.10}$$

$$S_{pi} = \frac{\mathrm{TTN}_{\mathrm{all}}}{\mathrm{TTN}_{\mathrm{all}} + \mathrm{TFP}_i} \tag{4.11}$$

$$\mathrm{pCCC} = \frac{\mathrm{TTN}_{\mathrm{all}}}{\sum_{i=1}^{C} \sum_{j=1}^{C} n_{ij}} \tag{4.12}$$

The term pCCC denotes the ***proportion of cases correctly classified***, which is a measure of the overall accuracy, and when multiplied by 100, denotes the percentage of cases correctly classified. Many times, this is the only metric reported for measuring prediction performance in multi-class problems. However, PCCC alone is sometimes quite misleading as there may be a model with relatively "high" accuracy, but it predicts the "unimportant" class labels fairly accurately (e.g., "unknown bucket"). However, the model may be making all sorts of mistakes on the classes that are actually critical to the application. This problem is serious when in the input data the number of samples of different classes is very unbalanced. For

Table 4.4 Confusion matrix with two classes

		Predicted values		
		True	False	Sum
Observed values	True	tp	fn	tp + fn
	False	fp	tn	fp + tn
	Sum	tp + fp	fn + tn	n

tp denotes true positives, fp denotes false positives, fn denotes false negatives, tn denotes true negatives, and n denotes total number of individuals

example, if there are 990 samples of class 1 and only 10 of class 2, the classifier can easily have a bias toward class 1. If the classifier classifies all samples as class 1, its accuracy will be 99%. This does not mean that it is an appropriate classifier, as it had a 100% error when classifying the samples of class 2. For this reason, reporting this metric with those reported in Eqs. (4.9–4.11) is recommended in order to have a better picture of the prediction performance of any statistical machine learning method (Ratner 2017). Also, it is important to highlight that when the problem only has two classes, the confusion matrix is reduced to Table 4.4.

From Table 4.4 the PCCC is calculated as $\frac{tp+tn}{n}$, while the $S_e = \frac{tp}{tp+fn}$, $S_p = \frac{tn}{tn+fp}$, and $P = \frac{tp}{tp+fp}$. Also, when there are only two classes, González-Camacho et al. (2018) suggest calculating the ***Kappa coefficient*** (κ) or Cohen's Kappa, which is defined as

$$\kappa = \frac{P_0 - P_e}{1 - P_e},$$

where P_0 is the agreement between observed and predicted values and is computed by the PCCC described above for two classes; P_e is the probability of agreement calculated as $P_e = \frac{tp+fn}{n} \times \frac{tp+fp}{n} + \frac{fp+tn}{n} \times \frac{fn+tn}{n}$, where fp is the number of false positives, and fn is the number of false negatives (Table 4.4). This statistic can take on values between -1 and 1; a value of 0 means there is no agreement between the observed and predicted classes, while a value of 1 indicates perfect agreement between the model prediction and the observed classes. Negative values indicate that a prediction may be incorrect; however large negative values seldom occur when working with predictive models. Depending on the context, a Kappa value from 0.30 to 0.50 indicates reasonable agreement (Kuhn and Johnson 2013). The Kappa coefficient is appropriate when data are unbalanced, because it estimates the proportion of cases that were correctly identified by taking into account coincidences expected from chance alone (Fielding and Bell 1997). It is important to point out that this statistic was originally designed to assess the agreement between two raters (Cohen 1960).

Another popular metric for binary data is the ***Area Under the receiver operating characteristic*** Curve (AUC–ROC) and it ranks the positive predictions higher than the negative. The ROC curve is defined as a plot of 1 − specificity or false positive rate (FPR) as the x-axis versus its model sensitivity as the y-axis. For a given set of

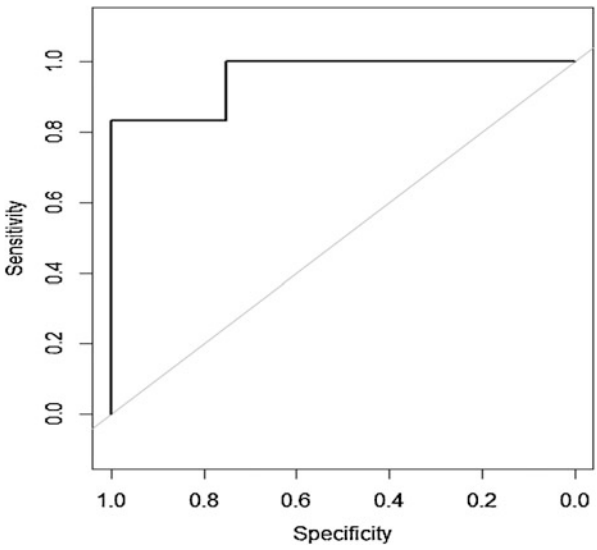

Fig. 4.10 ROC curve for the synthetic data

thresholds τ, it is an effective method for evaluating the quality or performance of diagnostic tests, and is widely used in statistical machine learning to evaluate the prediction performance of learning algorithms. Since the mathematical construction of this metric is not required in this book, we illustrate its calculation with one simple example. Assume that the observed (y), predicted probabilities (*pi*) and predicted values ($\hat{y}$) obtained after implementing a statistical machine learning model are $y = \{1, 0, 1, 1, 0, 0, 1, 1, 0, 1\}$, $pi = \{0.6, 0.55, 0.8, 0.78, 0.3, 0.42, 0.9, 0.45, 0.3, 0.88\}$, and $\widehat{y} = \{1, 1, 1, 1, 0, 0, 1, 0, 0, 1\}$; then, using the following R code, we can obtain many metrics for binary data (Fig. 4.10):

```
#########################Libraries required#########################
   library(caret)
   library(pROC)
   ##Observed (y), predicted probability (pi) and predicted values of
synthetic data##
   y=c(1, 0,1, 1, 0, 0, 1, 1, 0, 1)
   pi=c(0.6, 0.55, 0.8, 0.78, 0.3, 0.42, 0.9, 0.45, 0.3, 0.88)
   yhat=c(1, 1, 1, 1, 0, 0, 1, 0, 0, 1)
   xtab <- table(y,yhat)
   confusionMatrix(xtab)

   plot.roc(y,pi) #####This make the ROC curve plot

   Confusion Matrix and Statistics

     yhat
   y  0 1
    0 3 1
    1 1 5
```

```
               Accuracy : 0.8
                  95% CI : (0.4439, 0.9748)
    No Information Rate : 0.6
    P-Value [Acc > NIR] : 0.1673

                  Kappa : 0.5833
 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.7500
            Specificity : 0.8333
         Pos Pred Value : 0.7500
         Neg Pred Value : 0.8333
             Prevalence : 0.4000
         Detection Rate : 0.3000
   Detection Prevalence : 0.4000
      Balanced Accuracy : 0.7917

       'Positive' Class : 0
```

It is important to point out that when there are more than two classes, these metrics (accuracy, sensitivity, specificity, etc.) can be calculated on a "one-versus-all" basis that consists of using each class versus the pool of the remaining classes, as was illustrated for the confusion matrix with more than two classes (James et al. 2013).

When the statistical machine learning model discriminates correctly between the two groups, it produces a curve that coincides with the left and top sides of the plot. Under this scenario, the perfect model would have 100% sensitivity and specificity, and the ROC curve would be a single step between (0,0) and (0,1) and would remain constant from (0,1) to (1,1); this implies that the area under the ROC curve of the model would be 1. In general, the larger the area under the ROC curve, the better the model in terms of prediction performance. On the other hand, a completely useless statistical machine learning algorithm would give a straight line (45° diagonal line) from the bottom left corner to the top right corner of the plot. Different statistical machine learning models or the same model with different training sets or hyperparameters can be compared by superimposing their ROC curves in the same graph. In practice, most of the time the values in the two groups overlap, so the curve often lies between these extremes.

From this ROC curve, we can obtain a global assessment of the prediction performance of the statistical machine learning method by measuring the area under the receiver operating characteristic curve. This area is equal to the probability that a random individual (person) of the sample with the presence of the target has a higher value of the measurement than a random individual without the target.

When a statistical machine learning model is unable to discriminate between the positive and negative classes, this means that it has low discriminatory power. Therefore, only in statistical machine learning models that had good discriminatory power we can be confident of the predictions they provide and furthermore those models that provide a curve that lies considerably above the curve will be better.

Matthews correlation coefficient (MCC). Introduced in 1975 by Brian Matthews (1975) and regarded by many scientists as the most informative score that connects all four measures in a confusion matrix, the Matthews Correlation Coefficient is typically used in statistical machine learning to measure the quality of binary classifications and it is particularly useful when there is a significant imbalance in class sizes (data). MCC is calculated according to the following expression:

$$\text{MCC} = \frac{\text{tp} \times \text{tn} - \text{fp} \times \text{fn}}{\sqrt{(\text{tp} + \text{fp}) \times (\text{tp} + \text{fn}) \times (\text{tn} + \text{fp}) \times (\text{tn} + \text{fn})}} \tag{4.13}$$

If any of the denominator terms equals zero in (4.13) it will be set to 1 and MCC becomes zero, which has been shown to be the correct limiting value. It returns a value between -1 and 1, where 1 means a perfect prediction, 0 means no better than random and -1 means a total disagreement between predicted and observed values.

Next, we present the ***Brier score*** (Brier 1950) for categorical or binary data that can be computed as

$$\text{BS} = T^{-1} \sum_{i=n+1}^{n+T} \sum_{c=1}^{C} (\widehat{\pi}_{ic} - d_{ic})^2, \tag{4.14}$$

where $\widehat{\pi}_i$ denotes the estimated probabilities (predictive distribution) derived from the estimated model for observation i and d_{ic} takes a value of 1 if the categorical response observed for individual i falls into category c; otherwise, $d_{ic} = 0$. The range of BS in Eq. (4.14) for categorical data is between 0 and 2. For this reason, we suggest dividing by 2, that is, BS/2, to obtain the Brier score bound between 0 and 1; lower scores imply better predictions (Montesinos-López et al. 2015a, b).

Finally, we describe the use of negative log-likelihood (MLL) to evaluate the prediction performance. This metric has the characteristic that better forecasts have lower values and for this reason, it is analogous to the MSE. For categorical data,

$$\text{MLL} = -\frac{1}{T}\left[\sum_{i=n+1}^{n+T} \sum_{c=1}^{C} 1\{y_i = k\} \log(\widehat{\pi}_{ic})\right],$$

where $1\{y^{(i)} = k\}$ is an indicator variable taken the value of 1 when the ith observation is assigned to category c, for $c = 1, 2, \ldots, C$ takes place in the ith observation. When the data are binary, the MLL is reduced to $\text{MLL} = -\frac{1}{T}\left[\sum_{i=n+1}^{n+T}[y_i \log(\widehat{\pi}_i) + (1 - y_i) \log(1 - \widehat{\pi}_i)]\right]$. Following, we provide some advantages of using the MML as a measure of prediction performance: (a) it has a simple definition that, from a purely intuitive point of view, seems to be a reasonable basis on which to compare forecasts; (b) it is mathematically optimal in the sense that estimates of parameters of calibration models fitted by maximizing the likelihood are usually the most accurate possible estimates (see Cassella and Berger 2002); (c) it is a generalization to probabilistic forecasts of the most commonly used skill score for single forecasts; (d) the properties of the likelihood have been studied

at great length over the last 90 years, and as such is well understood; (e) it is both a measure of resolution and reliability; (f) likelihood can be used for both calibration and assessment: this creates consistency between these two operations; (g) use of the likelihood also creates consistency with other statistical modeling activities, since most other statistical modeling uses the likelihood, which is important in cases where the use of forecasts is simply a small part of a larger statistical modeling effort, as is the case of our particular business; (h) likelihood can be used for all types of response variables; and (i) likelihood can be used to compare multiple leads, multiple variables, and multiple locations at the same time in a sensible way with a single score even when these leads, variables, and locations are cross-correlated.

4.5.3 Count Measures of Prediction Performance

Spearman's correlation and the MML are recommended to measure the prediction performance for count data.

For the application of the ***Spearman's correlation***, the formula given in Eq. (4.2) for Pearson's correlation can be used; however, instead of using the observed and predicted values directly, these are replaced by their corresponding ranks. For example, assuming that the observed and predicted values are $y = \{15, 9, 12, 27, 6, 3, 36, 15, 21, 30\}$ and $\widehat{y} = \{20, 17, 24, 25, 3, 3, 34, 22, 21, 33\}$, we can thus show how to get the rank for the observed values: $\text{rango}_y = \{5, 3, 4, 8, 2, 1, 10, 6, 7, 9\}$. However, in this vector, observations 1 and 8 are the same, and as such, their positions are added and divided by two, that is, $\frac{5+6}{2} = 5.5$. Therefore, the final rank for the observed values is $\text{rango}_y = \{5.5,3, 4, 8, 2, 1, 10,5.5,7, 9\}$. Now the range for the predicted values is $\text{rango}_{\widehat{y}} = \{4, 3, 7, 8, 1, 2, 10, 6, 5, 9\}$, but again, since values 5 and 6 are the same, we add their ranges, and as this is repeated twice, it is divided by two and we get $\frac{1+2}{2} = 1.5$. Therefore, the final range of the predicted values is $\text{rango}_{\widehat{y}} = \{4, 3, 7, 8, 1.5, 1.5, 10, 6, 5, 9\}$. Finally, to obtain the Spearman correlation, we used the expression given in Eq. (4.2) for Pearson's correlation, and instead of using the original observed and predicted values, we used rango_y and $\text{rango}_{\widehat{y}}$. The interpretation of this metric is equal to that of the Pearson correlation, that is, when it is closer to 1, the prediction performance of the implemented statistical learning method is better. It should be noted that when the number of repeated values in the observed and predicted values is greater than two, the adjusted range is the sum of the repeated ranges divided by the number of repeated values; this new range is then given to the repeated values. In this case, it is also important to regress the observed versus the predicted values to obtain the intercept and slope using the ranges of the observed and predicted values.

It is also possible to use the MLL criteria to assess the prediction performance for count data; however, the new expression is now based on minus the log-likelihood of a Poisson distribution, which is equal to

$$\text{MLL} = \frac{1}{T}\sum_{i=n+1}^{n+T}\left[-\widehat{f}(x_i) + y_i \log\left(\widehat{f}(x_i)\right)\right]$$

Again, when the values of MLL are lower, the observed and predicted values are closer to one another.

References

Brier GW (1950) Verification of forecasts expressed in terms of probability. Mon Weather Rev 78: 1–3

Buduma M (2017) Fundamentals of deep learning, 1st edn. O'Reilly, Sabastopol, CA

Burger SV (2018) Introduction to machine learning with R. Rigorous mathematical analysis, 1st edn. O'Reilly, Sabastopol, CA

Cassella G, Berger RL (2002) Statistical inference. Duxbury, Belmont, CA

Cohen J (1960) A coefficient of agreement for national data. Educ Psychol Meas 20:37–46

Cook D (2017) Practical machine learning with H_2O. O'Reilly Media, Inc, Sabastopol, CA

Daetwyler HD, Calus MPL, Pong-Wong R, de los Campos G, Hickey JM (2012) Genomic prediction in animals and plants: simulation of data, validation, reporting and benchmarking. Genetics 193:347–365

Dewancker I, McCourt M, Clark S, Hayes P, Johnson A, Ke G (2016) A stratified analysis of Bayesian optimization methods. arXiv:1603.09441v1

Efron B (1983) Estimating the error rate of a prediction rule: improvement on cross-validation. J Am Stat Assoc 78(382):316–331

Fielding AH, Bell JF (1997) A review of methods for the assessment of prediction errors in conservation presence/absence models. Environ Conserv 24:38–49. https://doi.org/10.1017/S0376892997000088

González-Camacho JM, Ornella L, Pérez-Rodríguez P, Gianola D, Dreisigacker S, Crossa J (2018) Applications of machine learning methods to genomic selection in breeding wheat for rust resistance. Plant Genome 11(2):1–15. https://doi.org/10.3835/plantgenome2017.11.0104

Hagerty MR, Srinivasan S (1991) Comparing the predictive powers of alternative multiple regression models. Psychometrika 56:77–85. MR1115296

Hastie T, Tibshirani R, Friedman J (2008) The elements of statistical learning: data mining, inference, and prediction, Springer series in statistics, 2nd edn. Springer, New York

James G, Witten D, Hastie T, Tibshirani R (2013) An introduction to statistical learning: with applications in R. Springer, New York

Jarquín D, Lemes da Silva C, Gaynor RC, Poland J, Fritz AR et al (2017) Increasing genomic-enabled prediction accuracy by modeling genotype × environment interactions in Kansas wheat. Plant Genome 10(2):1–15. https://doi.org/10.3835/plantgenome2016.12.0130

Kim S, Kim H (2016) A new metric of absolute percentage error for intermittent demand forecasts. Int J Forecast 32(3):669–679

Koch P, Wujek B, Golovidov O, Gardner S (2017) Automated hyperparameter tuning for effective machine learning. In: Proceedings of the SAS global forum 2017 conference. SAS Institute Inc, Cary, NC. http://support.sas.com/resources/papers/proceedings17/SAS514-2017.pdf

Konen W, Koch P, Flasch O, Bartz-Beielstein T, Friese M, Naujoks B (2011) Tuned data mining: a benchmark study on different tuners. In: Proceedings of the 13th annual conference on genetic and evolutionary computation (GECCO-2011). SIGEVO/ACM, New York

Kuhn M, Johnson K (2013) Applied predictive modeling. Springer, New York

Lopez-Cruz M, Crossa J, Bonnett D, Dreisigacker S, Poland J, Jannink J-L, Singh RP, Autrique E, de los Campos, G. (2015) Increased prediction accuracy in wheat breeding trials using a marker × environment interaction genomic selection method. G3 5(4):569–582

Lorena AC, de Carvalho ACPLF (2008) Evolutionary tuning of SVM parameter values in multiclass problems. Neurocomputing 71:3326–3334
Lujan-Moreno GA, Howard PR, Rojas OG, Montgomery DC (2018) Design of experiments and response surface methodology to tune machine learning hyperparameters, with a random forest case-study. Expert Syst Appl 109:195–205
Matthews BW (1975) Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochim Biophys Acta Protein Struct 405:442–451
McKay MD (1992) Latin hypercube sampling as a tool in uncertainty analysis of computer models. In: Swain JJ, Goldsman D, Crain RC, Wilson JR (eds) Proceedings of the 24th conference on winter simulation (WSC 1992). ACM, New York, pp 557–564
Mesple F, Troussellier M, Casellas C, Legendre P (1996) Evaluation of simple statistical criteria to qualify a simulation. Ecol Model 88:9–18
Montesinos-López OA, Montesinos-López A, Pérez-Rodríguez P, de los Campos G, Eskridge KM, Crossa J (2015a) Threshold models for genome-enabled prediction of ordinal categorical traits in plant breeding. G3 5(1):291–300
Montesinos-López OA, Montesinos-López A, Crossa J, Burgueño J, Eskridge K (2015b) Genomic-enabled prediction of ordinal data with Bayesian logistic ordinal regression. G3 5(10):2113–2126. https://doi.org/10.1534/g3.115.021154
Montesinos-López A, Montesinos-López OA, Gianola D, Crossa J, Hernández-Suárez CM (2018a) Multi-environment genomic prediction of plant traits using deep learners with a dense architecture. G3 8(12):3813–3828. https://doi.org/10.1534/g3.118.200740
Montesinos-López OA, Montesinos-López A, Crossa J, Gianola D, Hernández-Suárez CM et al (2018b) Multi-trait, multi-environment deep learning modeling for genomic-enabled prediction of plant traits. G3 8(12):3829–3840. https://doi.org/10.1534/g3.118.200728
Piñeiro G, Perelman S, Guerschman JP, Paruelo JM (2008) How to evaluate models: observed vs. predicted or predicted vs. observed? Ecol Model 216:316–322
Ratner B (2017) Statistical and machine-learning data mining. Techniques for better predictive modelling and analysis of big data, 3rd edn. CRC Press Taylor & Francis Group, Boca Raton, FL
Renukadevi NT, Thangaraj P (2014) Performance analysis of optimization techniques for medical image retrieval. J Theor Appl Inf Technol 59:390–399
Shalev-Shwartz S, Ben-David S (2014) Understanding machine learning from theory to algorithms. Cambridge University press, New York
Shmueli G (2010) To explain or to predict? Stat Sci 25(3):289–310
Smith EP, Rose KA (1995) Model goodness-of-fit analysis using regression and related techniques. Ecol Model 77:49–64
Wu S, Harris T, Mcauley K (2007) The use of simplified or misspecified models: linear case. Canad J Chem Eng 85:386–398