Chapter 8
Prediction Modeling Methodology

Frank J. W. M. Dankers, Alberto Traverso, Leonard Wee, and Sander M. J. van Kuijk

8.1 Statistical Hypothesis Testing

A statistical hypothesis is a statement that can be tested by collecting data and making observations. Before you start data collection and perform your research, you need to formulate your hypothesis. An example hypothesis could be for instance: "If I increase the prescribed radiation dose to the tumor, this will also lead to an increase of side-effects in surrounding healthy tissues". The purpose of statistical hypothesis testing is to find out whether the observations are meaningful or can be attributed to noise or chance.

The original version of this chapter was revised. The correction to this chapter can be found at **https://doi.org/10.1007/978-3-319-99713-1_15**

Frank J. W. M. Dankers (✉)
Department of Radiation Oncology (MAASTRO), GROW School of Oncology and Developmental Biology (GROW), Maastricht University Medical Center, Maastricht, The Netherlands

Department of Radiation Oncology, Radboud University Medical Center, Nijmegen, The Netherlands
e-mail: frank.dankers@maastro.nl

A. Traverso · L. Wee
Department of Radiation Oncology (MAASTRO), GROW School of Oncology and Developmental Biology (GROW), Maastricht University Medical Center, Maastricht, The Netherlands

S. M. J. van Kuijk
Department of Clinical Epidemiology and Medical Technology Assessment (KEMTA), Maastricht University Medical Center, Maastricht, The Netherlands

P. Kubben et al. (eds.), *Fundamentals of Clinical Data Science*,
https://doi.org/10.1007/978-3-319-99713-1_8

The ***null hypothesis*** (often denoted H_0) generally states the currently accepted fact. Often it is formulated in such a way that two measured values have no relation with each other. The alternative hypothesis, H_1, states that there is in fact a relation between the two values. Rejecting or disproving the null hypothesis gives support to the belief that there is a relation between the two values.

To quantify the probability that a measured value originates from the distribution stated under the null statistical hypothesis tests are used that produce a **p-value** (e.g., Z-test or student's t-test). The p-value gives the probability of obtaining a value equal to or greater than the observed value if the null hypothesis is true. A high p-value indicates that the observed value is likely under the null assumption, vice versa a low p-value indicates that the observed value is unlikely given the null hypothesis, which can lead to its rejection.

There are common misconceptions regarding the interpretation of the p-value [1]:

- The p-value is not the probability that the null hypothesis is true
- The p-value is not the probability of falsely rejecting the null hypothesis (type I error, see below)
- A low p-value does not prove the alternative hypothesis

The p-value is to be used in combination with the **α level**. The α level is a predefined significance level by the researcher which equals the probability of falsely rejecting the null hypothesis if it is true (type I error). It is the probability (s)he deems acceptable for making a type I error. If the p-value is smaller than the α level, the result is said to be significant at the α level and the null hypothesis is rejected. Commonly used values for α are 0.05 or 0.001 (Fig. 8.1).

Confidence levels serve a similar purpose as the α level, and by definition the confidence level + α level = 1. So an α level of 0.05 corresponds to a 95% confidence level.

8.1.1 Types of Error

We distinguish between two types of errors in statistical testing [2]. If the null hypothesis is true but falsely rejected, this is called a **type I error** (comparable to a false positive, with a positive result indicating the rejection). The type I error rate, the probability of making a type I error, is equal to the α level since that is the significance level at which we reject the null hypothesis. Likewise, if the null hypothesis is false but not rejected, this is called a **type II error** (comparable to a false negative, with a negative result indicating the failed rejection) (Table 8.1).

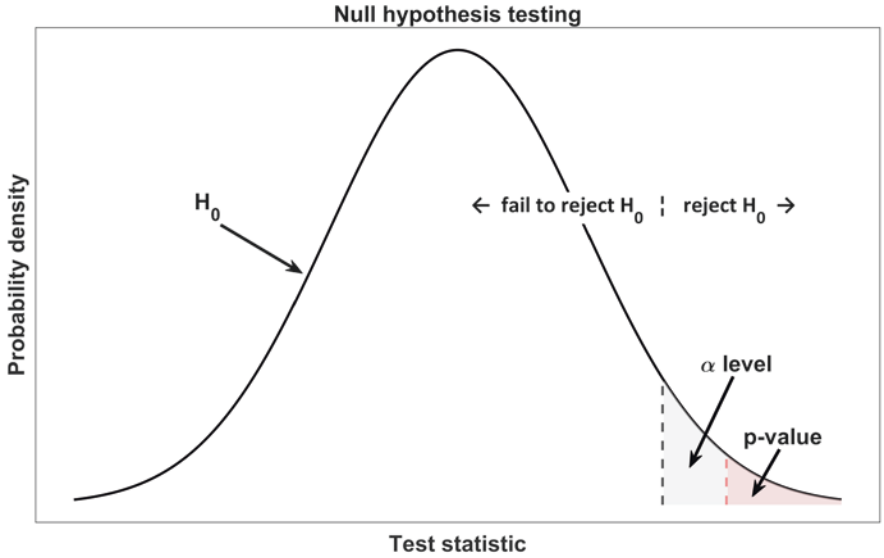

Fig. 8.1 Illustration of null hypothesis testing. The p-value represents the probability of obtaining a value equal or higher than the test value. The α level is predefined by the researcher and represents the accepted probability of making a type I error where the null hypothesis is falsely rejected. If the p-value of a statistical test is larger than the α level the null hypothesis is rejected

Table 8.1 The two types of errors that can be made regarding the acceptance or rejection of the null hypothesis

		Null hypothesis truth	
		True	False
Null hypothesis decision	Fail to reject	Correct	Type II error (false negative)
	Reject	Type I error (false positive)	Correct

8.2 Creating a Prediction Model Using Regression Techniques

8.2.1 Prediction Modeling Using Linear and Logistic Regression

A prediction model tries to stratify patients for their probability of having a certain outcome. The model then allows you to identify patients that have an increased chance of an event and this may lead to treatment adaptations for the individual

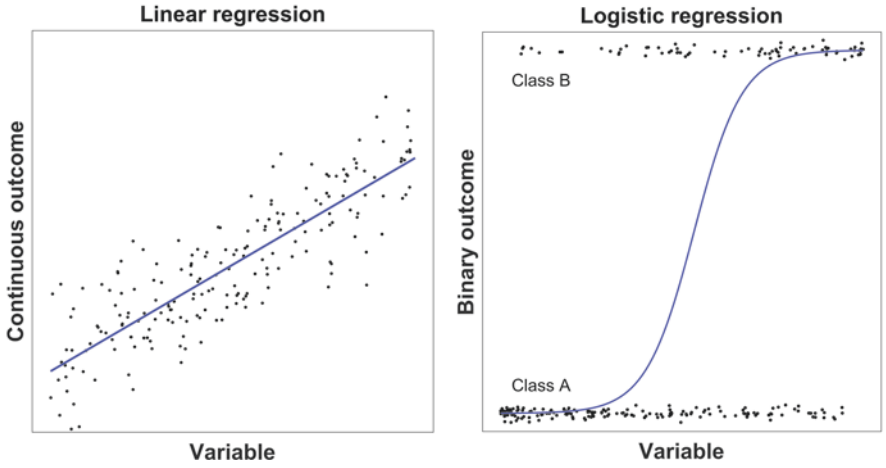

Fig. 8.2 Examples of predictive modeling (blue line) for a continuous outcome using linear regression and for a binary outcome using logistic regression. The predictions in the logistic regression are rounded to either class A or B using a threshold (0.5 by default)

patient. For instance, if a patient has an increased chance of a tumor recurrence the doctor may opt for a more aggressive treatment, or, if a patient has a high risk of getting a side-effect a milder treatment might be indicated.

The outcome variable of the prediction model can be anything, e.g., the risk of getting a side effect, the chance of surviving at a certain time point, or the probability of having a tumor recurrence. We can distinguish outcome variables into **continuous** variables or **categorical** variables. Continuous variables are described by numerical values and **regression** models are used to predict them, e.g., **linear regression**. Categorical variables are restricted to a limited number of classes or categories and we use **classification** models for their prediction. If the outcome has two categories this is referred to as binary classification and typical techniques are decision trees and **logistic regression** (somewhat confusingly, this regression method is well suited for classification due to its function shape).

Fitting or training a linear or logistic prediction model is a matter of finding the function coefficients so that the model function optimally follows the data (Fig. 8.2).

8.2.2 Software and Courses for Prediction Modeling

There are many different software packages available for generating prediction models, all of them with different advantages and disadvantages. Some packages are code-based and programming skills are required, e.g., Python, R or Matlab. There are integrated development environments available for improved

productivity, like RStudio for R, and Spyder for Python. Additionally, they can have rich open-source libraries tailored specifically towards machine learning, for instance Caret for R [3] and Scikit-learn for Python [4]. Other packages have graphical user interfaces and being able to program is not mandatory, like SPSS, SAS or Orange. Some packages are only commercially available, but many are open-source and have a large user base for support.

Preference for a certain software package over others is very personal and the best advice is therefore to simply try several and find out for yourself. A special mention is reserved for the Anaconda Distribution [5], which hosts many of the most widely used software packages for prediction modeling in a single platform (RStudio, Spyder, Jupyter Notebook, Orange and more) (Table 8.2).

There is a wealth of freely available information on the Web to help you get going. Providing a comprehensive overview is therefore an impossible task, but some excellent online courses (sometimes referred to as Massive Open Online Courses or MOOCs) are listed below (Table 8.3).

Table 8.2 A non-exhaustive overview of available software packages for prediction modeling and some of their features

Name	Reference	Coding required	Development environments	Open-source	Learn more (books/tutorials)
R	[6]	Yes	RStudio [7]	Yes	[8]
Python	[9]	Yes	Spyder [10] Jupyter notebooks [11]	Yes	[12]
Matlab	[13]	Yes	Matlab	No	[14]
SPSS	[15]	No	N/A	No	[16]
SAS	[17]	No	N/A	Partly (students)	[18]
Orange	[19]	No	Visual workflows	Yes	
Weka	[20]	No	N/A	Yes	[21]
Rapidminer	[22]	No	Visual workflows	Partly	

N/A not applicable

Table 8.3 Free online courses for prediction modeling and machine learning

Course	Organizer/link
Machine learning	Andrew Ng, Stanford University, Coursera
	https://www.coursera.org/learn/machine-learning
Machine learning	Tom Mitchell, Carnegie Mellon University
	http://www.cs.cmu.edu/~tom/10701_sp11/
Learning from data	Yaser Abu-Mostafa, California Institute of Technology
	https://work.caltech.edu/telecourse.html
Machine learning	Nando de Freitas, University of Oxford
	https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/

8.2.3 A Short Word on Modeling Time-to-Event Outcomes

Many studies are interested not only in predicting a certain outcome, but additionally take into account the time it takes for this outcome to occur. This is referred to as time-to-event analysis and a typical example is survival analysis. Kaplan-Meier curves are widely used for investigation of the influence of categorical variables [23], whereas **Cox regression** (or sometimes called Cox proportional hazards model) additionally allows the investigation of quantitative variables [24].

8.3 Creating a Model That Performs Well Outside the Training Set

8.3.1 The Bias-Variance Tradeoff

The bias-variance tradeoff explains the difficulty of a generated prediction model to generalize beyond the training set, i.e. perform well in an independent test set (also called the out-of-sample performance). The error of a model in an independent test set can be shown to be decomposable into a reducible component and an irreducible component. The irreducible component cannot be diminished, it will always be present no matter how good the model will be fitted to the training data. The origin of the irreducible error can, for instance, be an unmeasured but yet important variable for the outcome that is to be predicted.

The reducible error can be further decomposed into the error due to **variance** and the error due to **bias** [2]. The variance is the error due to the amount of overfitting done during model generation. If you use a very flexible algorithm, e.g., an advanced machine learning algorithm with lots of freedom to follow the data points in the training set very closely, this is more likely to overfit the data. The error in the training set will be small, but the error in the test set will be large. Another way to look at this is that a high variance will result in very different models during training if the model is fitted using different training sets.

Bias relates to the error due to the assumptions made by the algorithm that is chosen for model generation. If a linear algorithm is chosen, i.e. a linear relation between the inputs and the outcome is assumed, this may cause large errors (large bias) if the underlying true relation is far from linear. Algorithms that are more flexible (e.g., neural networks) result in less bias since they can match the underlying true but complex relations more closely.

In general it can be said that:

- Flexible algorithms have low bias since they can more accurately match the underlying true relation, but have high variance since they are susceptible to overfitting.
- Inflexible algorithms have low variance since they are less likely to overfit, but have high bias due to their problems of matching the underlying true relationship.

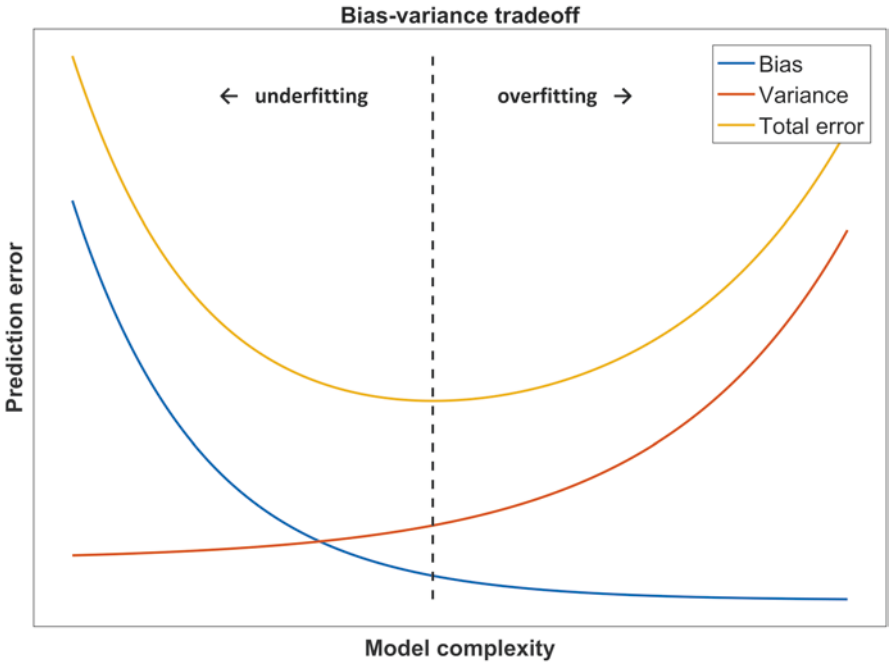

Fig. 8.3 The bias-variance tradeoff. With increased model complexity the model can more accurately match the underlying relation at the risk of increasing the variance (amount of overfitting). The bias-variance tradeoff corresponds to minimizing the total prediction error (which is the sum of bias and variance)

From this we can conclude that the final test set error is a tradeoff between low bias and low variance. It is impossible to simultaneously achieve the lowest possible variance and bias. The challenge is to generate a model with (reasonably) low variance and low bias since that is most likely to generalize well to external sets. This model might have slightly decreased performance in the training set, but will have the best performance in subsequent test sets (Fig. 8.3).

8.3.2 Techniques for Making a General Model

As we collect and expand our datasets we often score many features (parameters) so that we minimize the risk of potentially missing important features, i.e. features that are highly predictive of the outcome. This means that generally we deal with wide sets containing many features. However, many of these features are in fact redundant or not relevant for the outcome at all and can be safely omitted. Including a large number of features during model generation increases the possibility of chance correlations of features with the outcome (overfitting) and this results in models that

do not generalize well. **Dimensionality reduction** [25], reducing the number of features, is therefore an important step prior to model generation. The main advantages are:

- Lowered chance of overfitting and improved model generalization
- Increased model interpretability (depending on the method of dimensionality reduction)
- Faster computation times and reduced storage needs

There are many useful dimensionality reduction techniques. The first category of methods to consider is that of **feature selection**, where we limit ourselves to a subset of the most important features prior to model generation. Firstly, if a feature has a large fraction of missing values it is unlikely to be predictive of the outcome and can often be safely removed. In addition, if a feature has zero or near zero variance, i.e. its values are all highly similar, this again indicates that the feature is likely to be irrelevant. Another simple step is to investigate the inter-feature correlation, e.g., by calculating the Pearson or Spearman correlation matrix. Features that are highly correlated with each other are redundant for predicting the outcome (multicollinearity). Even though a group of highly correlated features may all be predictive of the outcome, it is sufficient to only select a single feature as the others provide no additional information.

Traditionally, further feature selection is then performed by applying stepwise regression. In each step a feature is either added or removed and a regression model is fitted and evaluated based on some selection criterion. There are many choices for the criterion to choose between models, e.g., the Bayesian information criterion or the Akaike information criterion, both of which quantify the measure of fit of models and additionally add a penalty term for complex models comprising more parameters [26]. In forward selection, one starts with no features and the feature that improves the model the most is added to the model. This process is repeated until no significant improvement is observed. In backward elimination, one starts with a model containing all features, and features are removed that decrease the model performance the least, until no features can be removed without significantly decreasing performance.

With feature selection we limit ourselves to a subset of features that are already present in the dataset and this is a special case of dimensionality reduction [27]. In **feature extraction** the number of features are reduced by replacing the existing features by fewer artificial features which are combinations of the existing features. Popular techniques for feature extraction are principle component analysis, linear discriminant analysis and autoencoders [25].

More advanced machine learning algorithms often contain embedded methods for reducing model complexity to improve generalizability. An example is **regularization** where each added feature also comes with an added penalty or cost [8]. The addition of a feature may increase the model performance but, if the added cost is too high, it will not be included in the final model. This effectively performs feature selection and prevents overfitting. The severity of the cost is a hyperparameter that can be tuned (e.g., through cross-validation, see paragraph “Techniques for internal validation”). Popular regularization methods for logistic regression are LASSO (or

L1 regularization) [28], ridge (or L2 regularization), or a combination of both using Elastic Net [29]. The main difference between L1 and L2 regularization is that in L1 regularization the coefficients of unimportant parameters shrink to zero, effectively performing feature selection and simplifying the final model.

8.4 Model Performance Metrics

8.4.1 General Performance Metrics

The performance of a prediction model is evaluated by the calculation of performance metrics. We want our model to have high discriminative ability, i.e. high probabilities should be predicted for observations having positive classes (e.g., alive after 2 years or treatment) and low probabilities for negative classes (dead after 2 years of treatment). There is no general best performance metric for model evaluation as this depends strongly on the underlying data as well as the intended application of the model.

Other often-used overall performance metrics are **R-squared** measures of goodness of fit (or R^2, also called the coefficient of determination). The R^2 can be interpreted as the amount of variance in the data that is explained by the model (explained variation). Higher R^2s correspond to better models. Examples are Cox and Snell's R^2 or Nagelkerke's R^2. R-squared values are mainly used in regression models; for classification models it is more appropriate to look at performance metrics derived from the confusion matrix.

Another popular overall performance measure is the **Brier score** (or mean squared error) and it is defined as the average of the square of the difference between the predictions and observations. A low Brier score indicates that predictions match observations and we are dealing with a good model.

8.4.2 Confusion Matrix

The **confusion matrix** is a very helpful tool in assessing model performance. It lists the correct and false predictions versus the actual observations and allows for the calculation of several insightful performance metrics. If the output of your prediction model is a probability (e.g., the output of a logistic regression model), then it needs to be dichotomized first by applying a threshold (typically 0.5) before the confusion matrix can be generated. An exemplary confusion matrix is shown in Table 8.4.

True positives, called hits, are cases that are correctly classified. True negatives are correctly rejected. False positives, or false alarm, are equivalent to a type I error. False negatives, or misses, are equivalent to a type II error.

Prevalence is defined as the number of positive observations with respect to the total observations. A balanced dataset has a prevalence close to 0.5, or 50%. Often, we have to deal with imbalanced datasets and this can lead to difficulties when

Table 8.4 Confusion matrix showing predictions and observations. Many useful performance metrics are derived from the values in the confusion matrix

		Observation			
		True	False		
Prediction	True	True positive (TP)	False positive (FP)	→	Positive predictive value (PPV)
	False	False negative (FN)	True negative (TN)	→	Negative predictive value (NPV)
		↓ Sensitivity (TPR)	↓ Specificity (TNR)		

interpreting certain performance metrics. Performance metrics can be high for poor models that are trained and tested on imbalanced datasets.

$$\text{Prevalence} = \left(\text{TP} + \text{FN}\right) / \left(\text{TN} + \text{TP} + \text{FP} + \text{FN}\right)$$

8.4.3 Performance Metrics Derived from the Confusion Matrix

Accuracy, defined as the proportion of correct predictions, is often reported in literature. Care has to be taken when using this metric in highly imbalanced datasets. Consider a dataset with only 10% positive observations. If the prediction model simply always predicts the negative class it will be correct in 90% of the cases. The accuracy is high, but the model is useless since it cannot detect any positive cases.

$$\text{Accuracy} = \left(\text{TN} + \text{TP}\right) / \left(\text{TN} + \text{TP} + \text{FP} + \text{FN}\right)$$

Another option is to look at the proportion of correct positive predictions for the total number of positive observations. This is called the **Positive Predictive Value** (PPV) or *precision*. Similarly, the proportion of correct negative predictions for the total number of negative observations is called the **Negative Predictive Value** (NPV), respectively. These metrics are of interest to patients and clinicians as they give the probability that the prediction matches the observation (truth) for a patient. PPV and NPV are dependent on the prevalence in the dataset making their interpretation more difficult. A high prevalence will increase PPV and decrease NPV (while keeping other factors constant).

$$\text{PPV} = \text{TP} / \left(\text{TP} + \text{FP}\right)$$

$$\text{NPV} = \text{TN} / \left(\text{TN} + \text{FN}\right)$$

If we want to consider characteristics not of the population but of the prediction model when applied as a clinical test, we can evaluate **sensitivity** and **specificity**.

Sensitivity, or True Positive Rate (TPR, or sometimes called *recall* or probability of detection), is defined as the probability of the model to make a positive prediction for the entire group of positive observations. It is a measure of avoiding false negatives, i.e. not missing any diseased patients.

Similarly, specificity is defined as the probability of the model to make a negative prediction for the entire group of negative observations. It is a measure of avoiding false positives, i.e. not including non-diseased patients.

$$\mathrm{TPR} = \mathrm{TP} / \left(\mathrm{TP} + \mathrm{FN}\right) \qquad \left(\text{sensitivity}\right)$$

$$\mathrm{TNR} = \mathrm{TN} / \left(\mathrm{TN} + \mathrm{FP}\right) \qquad \left(\text{specificity}\right)$$

Additionally, we can determine the False Positive Rate (FPR), or fall-out, and the False Negative Rate (FNR), which are the opposites of TPR and FPR, respectively. Note that FPR is used in the next paragraph for the construction of the Receiver Operating Characteristic curve.

$$\mathrm{FNR} = 1 - \mathrm{TPR} = 1 - \text{sensitivity}$$

$$\mathrm{FPR} = 1 - \mathrm{TNR} = 1 - \text{specificity}$$

The **F1-score**, or F-score, is a metric combining both PPV (precision) and TPR (recall or sensitivity). Unlike PPV and TPR separately, it takes both false positives and false negatives into account simultaneously. It does however still omit the true negatives. It is typically used instead of accuracy in the case of severe class imbalance in the dataset.

$$\mathrm{F1} = 2 \cdot \left(\mathrm{PPV} \cdot \mathrm{TPR}\right) / \left(\mathrm{PPV} + \mathrm{TPR}\right)$$

8.4.4 Model Discrimination: Receiver Operating Characteristic and Area Under the Curve

The performance of a prediction model is always a tradeoff between sensitivity and specificity. By changing the threshold that we apply to round our model predictions to positive or negative classes, we can change the sensitivity and specificity of our model. By decreasing this threshold, we are making it easier for the model to make positive predictions. The number of false negatives will go down but false positives will go up, increasing sensitivity but lowering specificity. By increasing the threshold, the model will make fewer positive predictions, the number of false negatives will go up and false positive will go down, decreasing sensitivity and increasing specificity (Fig. 8.4).

By evaluating different thresholds for rounding our model predictions, we can determine many sensitivity and specificity pairs. If we plot the sensitivity versus

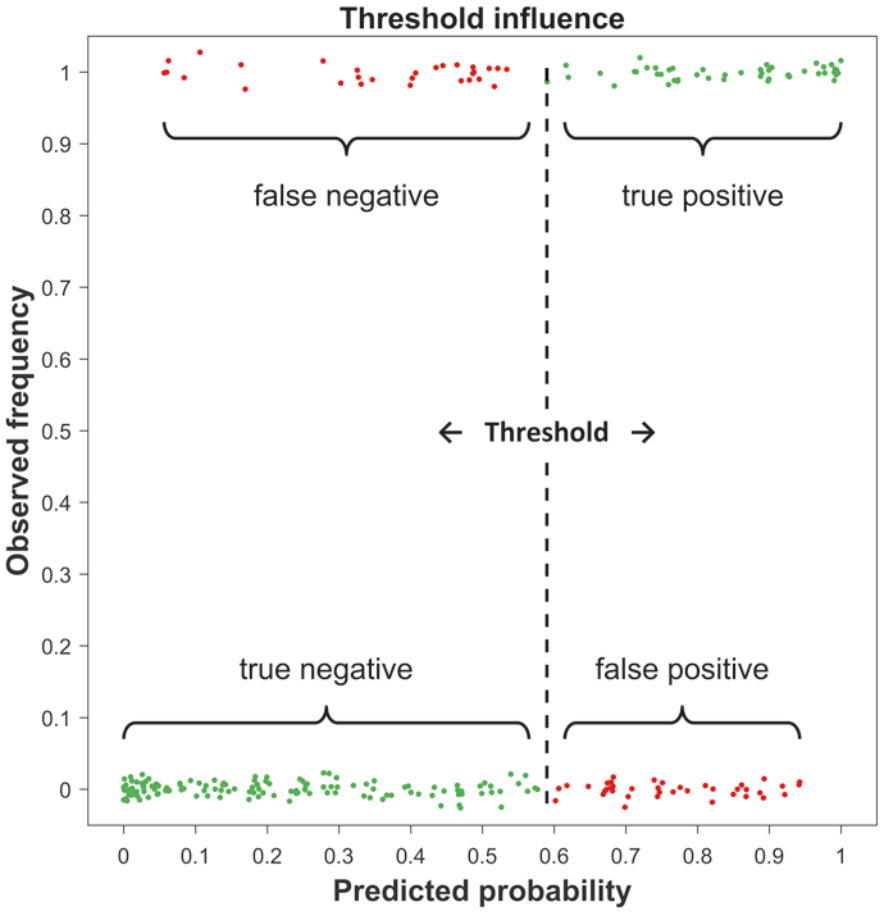

Fig. 8.4 Influence of the threshold that is used to round model prediction probabilities to 0 or 1. By using a low threshold the model will detect most of the patients with the outcome (high sensitivity), but many patients without the outcome will also be included (low specificity). For each value of the threshold sensitivity and specificity values can be calculated

(1 – specificity) for all these pairs, i.e. the true positive rate versus the false positive rate, we obtain the **Receiver Operating Characteristic** curve (ROC) [30]. This curve can give great insight into model discrimination performance. It allows for determining the optimal sensitivity/specificity pair of a model so that it can support decision making, and also allows comparison of different models with each other.

Powerful models have ROC curves that approach the upper left corner, which indicates that the model achieves the maximum of 100% sensitivity and 100% specificity simultaneously. Conversely, a poor model with no predictive value will have an ROC curve close to the y = x or 45 degree line. This has led to the use of the **Area**

Under the Curve (AUC) of the ROC curve (or concordance statistic, c) as a widely used metric for interpreting individual model performance but also for comparing between models. Strong performing models have higher ROC curves and thus larger AUC values. A perfect model making correct predictions for every patient has an AUC of 1, whereas a useless model giving random predictions results in an AUC of 0.5. The AUC can be interpreted as the probability that the model will give a higher predicted probability to a randomly chosen positive patient than a randomly chosen negative patient (Fig. 8.5).

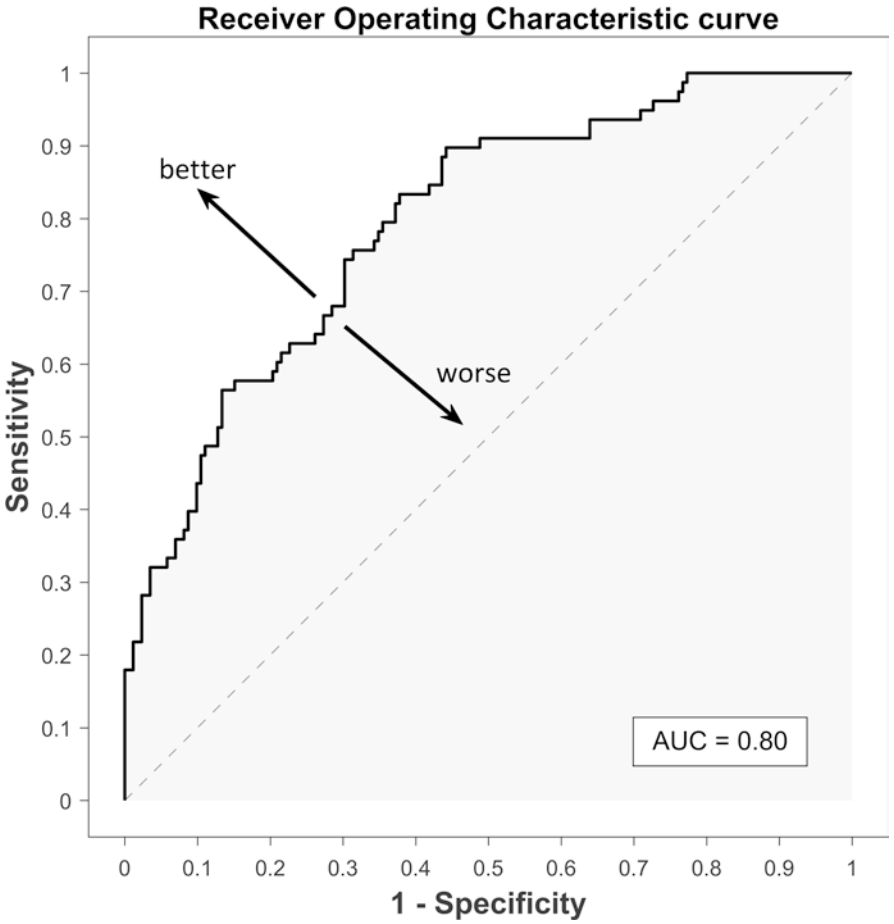

Fig. 8.5 ROC curve indicating discriminating performance of the model. Model predictions are rounded to 0 or 1 using many different thresholds resulting in the sensitivity and specificity pairs that form the curve. AUC is indicated by the gray area under the curve. Higher values correspond to better model discrimination performance

8.4.5 Model Calibration

Historically, the focus in evaluating model performance has primarily been on discriminative performance, e.g., by calculating R^2 metrics, confusion matrix metrics and performing ROC/AUC analysis. Model calibration is however as important as discrimination and should always be evaluated and reported. Model calibration refers to the agreement between subgroups of predicted probabilities and their observed frequencies. For example, if we collect 100 patients for which our model predicts 10% chance of having the outcome, and we find that in reality 10 patients actually have the outcome, then our model is well calibrated. Since the predicted probabilities can drive decision-making it is clear that we want the predictions to match the observed frequencies.

A widely-used (but no so effective) way of determining model calibration is by performing the **Hosmer-Lemeshow test** for goodness of fit of logistic regression models. The test evaluates the correspondence between predictions and observations by dividing the probability range [0-1] into *n* subgroups. Typically, 10 subgroups are chosen, but this number is arbitrary and can have a big influence on the final p-value of the test.

A better approach is to generate a **calibration plot** [31–33]. It is constructed by ordering the predicted probabilities, dividing them into subgroups (again, typically 10 is chosen) and then plotting the average predicted probability versus the average outcome for each subgroup. The points should lie close to the ideal line of y = x indicating agreement between predictions and observed frequencies for each subgroup. Helpful additions are error bars, a trend line (often a LOESS smoother [34]), individual patient predictions versus outcomes and/or histograms of the distributions of positive and negative observations (the graph is then sometimes called a validation plot) (Fig. 8.6).

8.5 Validation of a Prediction Model

8.5.1 The Importance of Splitting Training/Test Sets

In the previous paragraphs different metrics for evaluation of model performance have been discussed. As briefly discussed in paragraph "The bias-variance tradeoff" it is important to compute performance metrics not on the training dataset but on data that was not seen during the generation of the model, i.e. a test or validation set. This will ensure that you are not mislead into thinking you have a good performing model, while it may in fact be heavily overfitted on the training data. Overfitting means that the model is trained *too well* on the training set and starts to follow the noise in the data. This generally happens if we allow too many parameters in the final model. The performance on the training set is good, but on new data the model will fail.

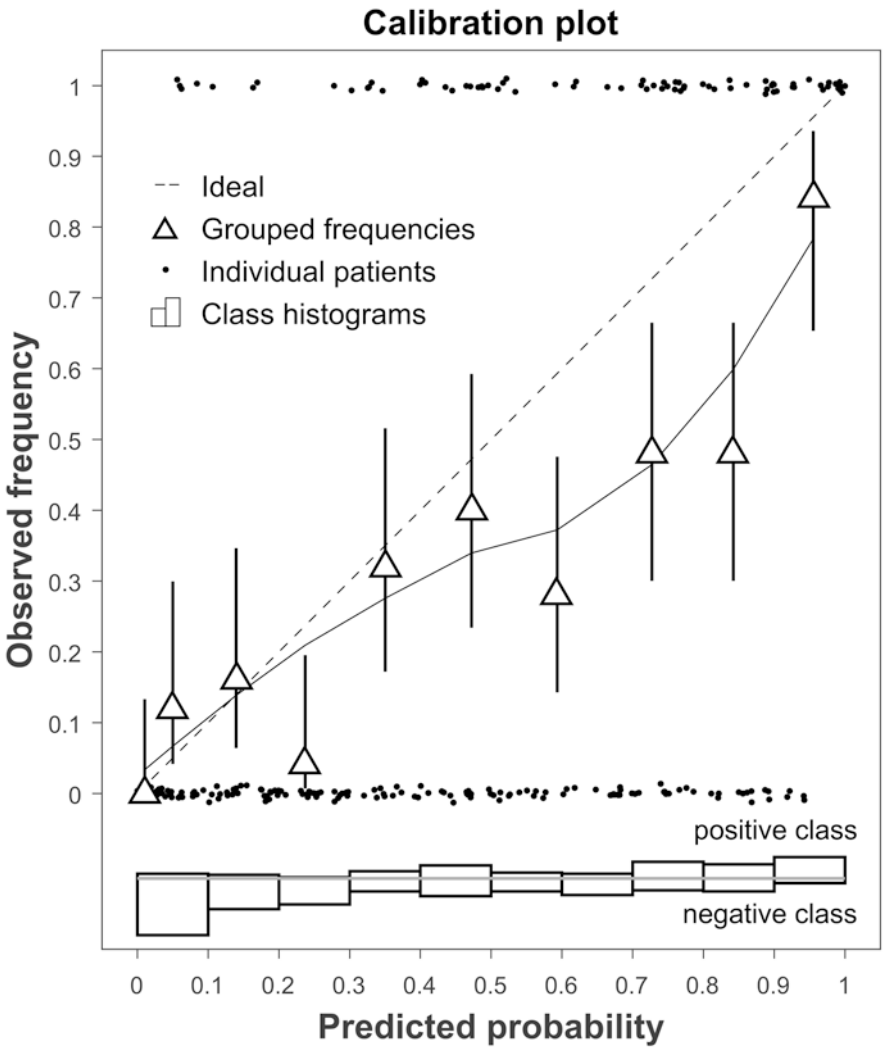

Fig. 8.6 Calibration plot indicating agreement between model predictions and observed frequencies. Data points are subdivided into groups for which the mean observation is plotted against the mean model probability. Perfect model calibration corresponds with the y = x line. Additionally, individual data points are shown (with some added y-jitter to make them more clear), as well as histograms for the positive and negative classes [0,1]

Underfitting corresponds to models that are too simplistic and do not follow the underlying patterns in the data, again resulting in poor performance in unseen data.

Properly evaluating your model on new/unseen data will improve the generalizability of the model. We differentiate between **internal validation**, where the data-

set is split into a training set for model generation and a test set for model validation, and **external validation**, where the complete dataset is used for model generation and separate/other datasets are available for model validation.

8.5.2 Techniques for Internal Validation

If you only have a single dataset available you can generate a test set by slicing of a piece of the training set. The simplest approach is to use a **random split**, e.g., using 70% of the data for training and 30% for evaluation (sometimes called a hold-out set). It is important to stratify the outcome over the two sets, i.e. make sure the prevalence in both sets remains the same. The problem with this method is that we can never be sure that the calculated performance metric is a realistic estimate of the model performance on new data or due to a(n) '(un)lucky' randomization. This can be overcome by repeating for many iterations and averaging the performance metrics. This method is called **Monte Carlo cross-validation** (Fig.8.7) [35].

Another approach is the method of ***k*-fold cross-validation** [36]. In this method the data is split into *k* stratified folds. One of these folds is used as a test set, the others are combined and used for model training. We then iterate and use every fold as a test set once. A better estimate of the true model performance can be achieved by averaging the model performances on the test set. Typically, k = 5 or k = 10 is chosen for the number of folds (Fig. 8.8).

The advantage of *k*-fold cross-validation is that each data point is used in a test set only once, whereas in Monte Carlo cross-validation it can be selected multiple times (and other points are not selected for a test set at all), possibly introducing bias. The disadvantage of *k*-fold cross-validation is that it only evaluates a limited number of splits whereas Monte Carlo cross-validation evaluates as many split as you desire by increasing the number of iterations (although you could iterate the entire *k*-fold cross-validation procedure as well which is commonly called repeated *k*-fold cross-validation).

Note that in both Monte Carlo cross-validation and *k*-fold cross-validation we are generating many models instead of a single final model, e.g., because the feature

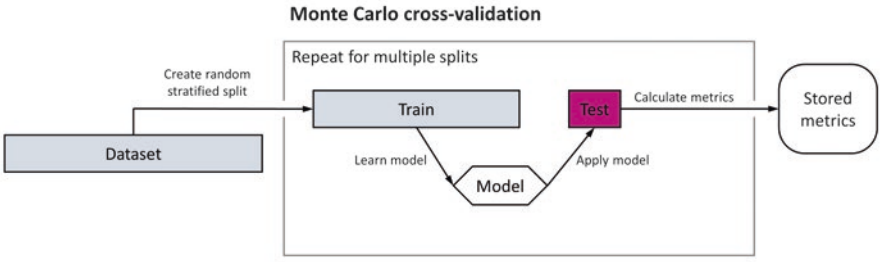

Fig. 8.7 Schematic overview of a Monte Carlo cross-validation. A random stratified split is applied to separate a test set from the training set. A prediction model is trained on the training set and performance metrics on the test set are stored after which the process is repeated

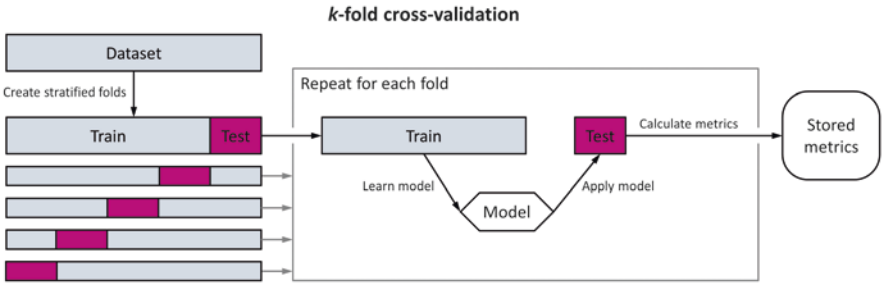

Fig. 8.8 Schematic overview of k-fold cross-validation. The dataset is randomly split into k stratified folds. Each fold is used as a test set once, while the other folds are temporarily combined to form a training set for model generation. Performance metrics on the test set are calculated and stored, and the process is repeated for the number of folds that have been generated

selection algorithm might select different features or the regression produces different coefficients due to different training data. Cross-validation is used to identify the best method (i.e. data pre-processing, algorithm choice etc.) that is to be used to construct your final model. When you have identified the optimal method you can then train your model accordingly on all the available data.

A common mistake in any method where the dataset is split into training and test sets is to allow **data leakage** to occur [37]. This refers to using any data or information during model generation that is not part of the training set and can result in overfitting and overly optimistic model performance. It can happen for example when you do feature selection on the total dataset before applying the split. In general it is advised to perform any data pre-processing steps after the data has been split and using only information available in the training set.

8.5.3 External Validation

The true test of a prediction model is to evaluate its performance under external validation, or separate datasets from the training dataset. Preferably, this is performed on new data acquired from a different institution. It will indicate the generalizability of the model and show whether it is overfitted on the training data. If this can be performed on multiple external validation sets, this further strengthens the acceptance of the prediction model under evaluation.

It has to be noted that if the datasets intended to be used for external validation are collected by the same researchers that built the original prediction model, this is still not an *independent* validation. Independent external validation, by other researchers, is the ultimate test of the model generalizability. This requires open and transparent reporting of the prediction model, of inclusion and exclusion criteria for the training cohort and of data pre-processing steps. Additionally, it is encouraged to make the training data publicly available as this allows other researchers to verify your methodology and results and greatly improves reproducibility.

8.6 Summary Remarks

8.6.1 What Has Been Learnt

In this chapter you have learnt about the importance of the bias-variance tradeoff in prediction modeling applications. You have learnt how to generate a simple logistic regression model and what metrics are available to evaluate its performance. It is important to not limit the evaluation to model discrimination only, but also include calibration as well. Finally, we have discussed the importance of separating training and test sets so that we protect ourselves from overfitting. Internal validation strategies such as cross-validation are discussed, and the ultimate test of a prediction model, independent external validation, has been emphasized.

8.6.2 Further Reading

The field of prediction modeling and machine learning is extremely broad and in this chapter we have only scratched the surface. A good place to start with further reading on the many aspects of prediction modeling is the book "*Clinical Prediction Models – A Practical Approach to Development, Validation, and Updating*" by Steyerberg [38]. If you are looking to improve your knowledge and simultaneously improve your practical modeling skills the book "*An Introduction to Statistical Learning – with Applications in R*" by James et al. is highly recommended [8]. Finally, if you want to go in-depth and understand the underlying principles of the many machine learning algorithms the go-to book is "*The Elements of Statistical Learning – Data Mining, Inference, and Prediction*" by Hastie et al. [39].

References

1. Goodman S. A dirty dozen: twelve P-value misconceptions. Semin Hematol. 2008;45(3):135–40.
2. Banerjee A, Chitnis UB, Jadhav SL, Bhawalkar JS, Chaudhury S. Hypothesis testing, type I and type II errors. Ind Psychiatry J. 2009;18(2):127–31.
3. Kuhn M, et al. Caret: classification and regression training, 2016.
4. Pedregosa F, et al. Scikit-learn: machine learning in Python. J Mach Learn Res. 2011;12:2825–30.
5. Anaconda distribution: The most popular Python/R data science distribution. [Online]. Available: https://www.anaconda.com/distribution/.
6. R: The R Project for Statistical Computing. [Online]. Available: https://www.r-project.org/.
7. RStudio: Open source and enterprise-ready professional software for R, *RStudio*. [Online]. Available: https://www.rstudio.com/products/rstudio/.

8. James G, Witten D, Hastie T, Tibshirani R. An introduction to statistical learning – with applications in R. 1st ed. New York: Springer; 2013.
9. Python: The official home of the Python Programming Language. [Online]. Available: https://www.python.org/.
10. Spyder: The Scientific PYthon Development EnviRonment. [Online]. Available: https://pythonhosted.org/spyder/index.html.
11. Jupyter: Open-source web application for live coding, data visualizations, numerical simulation, statistical modeling and more. [Online]. Available: http://jupyter.org/.
12. Müller A, Guido S. Introduction to machine learning with Python. Sebastopol: O'Reilly Media; 2016.
13. Matlab: The easiest and most productive software environment for engineers and scientists. [Online]. Available: https://www.mathworks.com/products/matlab.html.
14. Murphy P. Machine learning: a probabilistic perspective. Cambridge: The MIT Press; 2012.
15. SPSS: The world's leading statistical software used to solve business and research problems by means of ad-hoc analysis, hypothesis testing, geospatial analysis and predictive analytics. [Online]. Available: https://www.ibm.com/analytics/spss-statistics-software.
16. George D, Mallery P. IBM SPSS statistics 23 step by step: Pearson Education; 2016.
17. SAS: SAS/STAT State-of-the-art statistical analysis software for making sound decisions. [Online]. Available: https://www.sas.com/en_us/software/stat.html.
18. SAS/STAT® 13.1 User's Guide. SAS Institute Inc, 2013.
19. Orange: Open source machine learning and data visualization for novice and expert. [Online]. Available: https://orange.biolab.si/.
20. Weka: Data mining software in Java. [Online]. Available: https://www.cs.waikato.ac.nz/ml/index.html.
21. Witten I, Frank E, Hall M, Pal C. Data mining: practical machine learning tools and techniques. Burlington: Morgan Kaufmann; 2016.
22. RapidMiner Studio: Visual workflow designer for data scientists. [Online]. Available: https://rapidminer.com/products/studio/.
23. Efron B. Logistic regression, survival analysis, and the Kaplan-Meier curve. J Am Stat Assoc. 1988;83(402):414–25.
24. Walters SJ. Analyzing time to event outcomes with a Cox regression model. Wiley Interdiscip Rev Comput Stat. 2012;4(3):310–5.
25. van der Maaten LJP, Postma EO, van den Herik HJ. Dimensionality reduction: a comparative review. Tilburg University Technical Report TiCC TR 2009-005; 2009.
26. Vrieze SI. Model selection and psychological theory: a discussion of the differences between the Akaike information criterion (AIC) and the Bayesian information criterion (BIC). Psychol Methods. 2012;17(2):228–43.
27. Dash M, Liu H. Feature selection for classification. Intell Data Anal. 1997;1(3):131–56.
28. Tibshirani R. Regression shrinkage and selection via the Lasso. J R Stat Soc Ser B Methodol. 1996;58(1):267–88.
29. Zou H, Hastie T. Regularization and variable selection via the elastic net. J R Stat Soc Ser B Stat Methodol. 2005;67(2):301–20.
30. Fawcett T. An introduction to ROC analysis. Pattern Recogn Lett. 2006;27(8):861–74.
31. Steyerberg EW, et al. Assessing the performance of prediction models: a framework for some traditional and novel measures. Epidemiology. 2010;21(1):128–38.
32. Moons KGM, et al. Risk prediction models: I. Development, internal validation, and assessing the incremental value of a new (bio)marker. Heart. 2012;98(9):683–90.
33. Steyerberg EW, Vergouwe Y. Towards better clinical prediction models: seven steps for development and an ABCD for validation. Eur Heart J. 2014;35(29):1925–31.
34. Austin PC, Steyerberg EW. Graphical assessment of internal and external calibration of logistic regression models by using loess smoothers. Stat Med. 33(3):517–35.
35. Xu Q-S, Liang Y-Z. Monte Carlo cross validation. Chemom Intell Lab Syst. 2001;56(1):1–11.

36. Rodriguez JD, Perez A, Lozano JA. Sensitivity analysis of k-fold cross validation in prediction error estimation. IEEE Trans Pattern Anal Mach Intell. 2010;32(3):569–75.
37. Luo W, et al. Guidelines for developing and reporting machine learning predictive models in biomedical research: a multidisciplinary view. J Med Internet Res. 2016;18(12)
38. Steyerberg E. Clinical prediction models: a practical approach to development, validation, and updating. New York: Springer-Verlag; 2009.
39. Hastie T, Tibshirani R, Friedman J. The elements of statistical learning: data mining, inference, and prediction. New York: Springer-Verlag; 2001.