Chapter 9
Diving Deeper into Models

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

9.1 Introduction

In the previous chapters of the book, you have been learning the major techniques to prepare your data, develop and validate a clinical prediction model. The workflow generally consists of selecting some input features and combining them to predict relevant clinical outcomes. The way in which features are combined and relationships between data are discovered are several. In fact, several algorithms used to relate features with expected outcome are available. However, in the previous chapters, the focus has been pointed on one specific algorithm: logistic regression. This chapter proposes you an additional list of algorithms that can be used to train a model.

A. Traverso, PhD (✉) · B. Osong, MSc · L. Wee, PhD
Department of Radiation Oncology (MAASTRO), GROW School for Oncology and Developmental Biology, Maastricht University Medical Center+, Maastricht, The Netherlands
e-mail: alberto.traverso@maastro.nl

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

Department of Radiation Oncology, Radboud University Medical Center, Nijmegen, The Netherlands

S. M. J. van Kuijk, PhD
Department of Clinical Epidemiology and Medical Technology Assessment, 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_9

9.2 What Is Machine Learning?

Machine learning is an application of Artificial Intelligence (AI). AI refers to the **ability to program computers (or more in general machines) that are able to solve complicated and usually very time-consuming tasks** [1]. An example of a time consuming and complicated task is the extraction of useful information (data mining) from a large amount of unstructured clinical data ('big data').

9.3 How Do We Use Machine Learning in Clinical Prediction Modelling?

As you learned in previous chapters, after having prepared your data, you develop a clinical prediction model based on available data. In the model, particular properties of your data ('features') will be used to predict your outcome of interest. A particular statistical algorithm is used to learn the 'features' that are most representative and relate them to the predicted outcome. In the previous chapter, only the logistic regression algorithm has been presented to you. However, more complex machine learning-based algorithms exist. These algorithms can be divided into two categories: **supervised and unsupervised** [2].

9.4 Supervised Algorithms

These algorithms apply when learning from 'labelled' data to predict future outcomes [3]. To understand what we mean by labelled data, let us considering the following example. Suppose we are building a model that takes as input some clinical data from the patients (e.g. age, sex, tumor staging) and aims at predicting if a patient will be alive or not (binary outcome) 1 year after receiving the treatment therapy. In our training dataset, the clinical outcome (alive or dead after a certain elapse of time) information is available. These are labelled data. **In supervised learning, the analysis starts from a known training dataset and the algorithm is used to infer the predictions**. The algorithm compares its output to the 'labels' in order to modify it accordingly to match the expected values.

9.5 Unsupervised Algorithms

Unsupervised algorithms are used when the training data is not classified or labelled [4]. A common example of unsupervised learning is trying to cluster a population and see if the clusters share common properties ('features'). This common approach is used in marketing analysis, to see for example if different products might be assigned to different clusters. In summary, **the goal for unsupervised learning is to model**

the underlying structure or distribution in the data in order to learn more about the data. Unsupervised problems can then still be divided into two subgroups:

1. Clustering: the goal is to discover groups that share common features;
2. Association: used to describe rules that explain large portions of data. For example, still from the marketing analysis: people in a certain cluster buy a product X and more likely will also buy a certain product Y.

9.6 Semi-supervised Algorithms

We refer to semi-supervised algorithms **when the number of input data is much greater than the number of output labelled data in the training set** [5]**.** A good example could be a large data set of images, where only few of them are labelled (e.g. dog, cat). Despite this kind of learning problem is not often mentioned, most of the real life machine learning classification problems fall into this category. In fact, labelling data is a very time-consuming task. Imagine in fact a doctor that has to annotate (i.e. delineate anatomical or pathological structures) on hundreds of patients' scans, which you would like to use as your training dataset.

We will now provide an overview of the main algorithms for each presented category.

9.7 Supervised Algorithms

9.7.1 Support Vector Machines (SVMs)

SVMs can used for both classification and regression, despite being mostly used in classification problems [6]. The SVM are based on an n-dimensional space where n is the number of features you would like to use as an input. Imagine plotting all your data in the hyperspace, where each point correspond to the n-dimensional feature

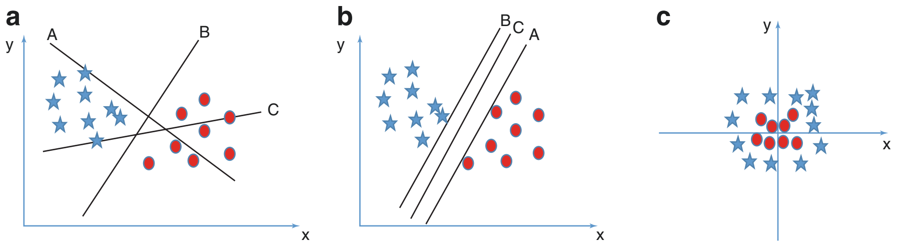

Fig. 9.1 SVMs examples. (**a**) Three different solutions of the problems are drawn. The solution that optimizes the separation betweeb the two clusters of data (stars vs circles) is line B. (**b**) the optimal solution is line C, by keeping into consideration the concept of margin. However, non linear solutions migth be needed as shown in (**c**)

vector of your input data. Therefore, for example, if you have 100 input data and 10 features, 100 vectors of dimension 10 will constitute the hyperspace.

The SVM will try to find the hyperplane / hyperplanes that separate your data into two (or more) classes.

How to find the best hyperplane? If we look at Fig. 9.1a, three possible hyperplanes separate the classes. The key point to be considered is: **"Choose the hyperplane that maximizes the separation between classes"**. Now, in Fig. 9.1a it is easy to affirm that the correct answer is line B. However, what should we choose if we look at Fig. 9.1b?

The definition of **margin** will help us. In SVMs **the margin is defined as the distance between the nearest data point or class (called the "support vector") and hyper-plane**. With this definition in mind, it becomes clear that the best solution in Fig. 9.1b is line C. However, we only have considered problems where the classes were easily separable by linear hyperplanes. What happens if the problem is more complicated like shown in Fig. 9.1c? It is clear that we cannot have a linear hyperplane to separate the classes, but visually it seems that a hyper-circle might work. This relates to the concept of **kernel. A SVMs kernel function gives the possibility to transform non-linear spaces** (Fig. 9.1c) **into linear spaces** (Figs. 9.1a and b)**.** Most common available SVMs computational packages [7] [8] offer different kernels from the most famous radial basis function-based kernel to higher order polynomials or sigmoid functions.

What are the most important parameters in a SVM?

- Kernel: the kernel is one of the most important parameters to be chosen. SVMs offer easier and more complicated kernels. Our suggestion to choose the kernel, is to plot the data projected on some features axis in order to have a visual ideal if the problem can be solved by choosing a linear kernel. In general**, we discourage to start using more complicated kernels from the beginning, since they can easily lead to high probability of overfitting**. It could be a good idea to start with a quadratic polynomial and then increase in complexity. Please keep into consideration that, in general, **complexity also increases computational time** (and required computational power).
- Soft margin constant (C): the "C" parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

Advantages/disadvantages of SVMs

Advantages

1. SVMs can be a useful tool in the case of non-regularity in the data, for example, when the data are not regularly distributed or have an unknown distribution, due to SVMs kernel flexibility.
2. Due to kernels transformations, also not linear classification problems can be solved

3. **SVM delivers an unique solution since the optimization problem is convex**

   Disadvantages

1. **SVM is a non parametric technique, so the results might lack transparency ("black box")**. For example, using a Gaussian kernel each features have a different importance (e.g. different weights in the kernel), therefore it is not trivial to find a straightforward relation between the features and the classification output, like what happens by using logistic regression

9.7.2 *Random Forests (RF)*

RFs are part of the algorithms called decision trees. In decision trees, the goal is to create a prediction model that predicts an output by combining different input variables [9]. In the decision tree, each node corresponds to one of the input variables, Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf. **Why random forests are called random?** The term random is justified by the fact that the random forest algorithm trains different decision trees by using different subsets of the training data. The RF algorithm is depicted in Fig. 9.2. In addition, each node in the decision

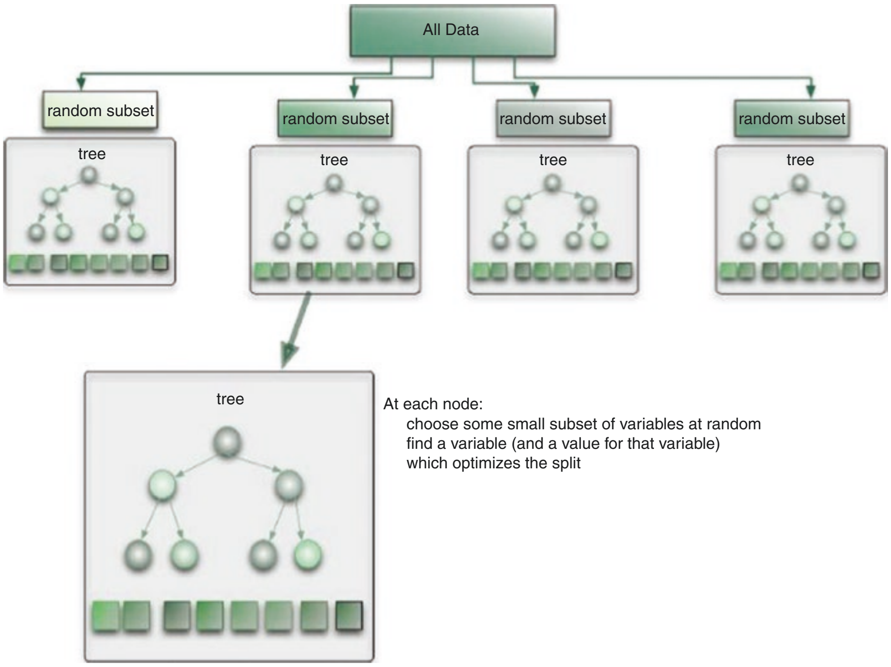
All Data
random subset
random subset
random subset
random subset
tree
tree
tree
tree
tree
At each node:
choose some small subset of variables at random
find a variable (and a value for that variable)
which optimizes the split

Fig. 9.2 Sketch representation of a RF workflow. RFs trained different algorithms by looking at random subsets of the data. The randomness generates models that are not correlated to each other

tree is split by using random selected features from the data. Therefore**, the randomness generates models that are not correlated to each other**.

What are the most important parameters in a RF?

- Maximum features: this is the maximum number of features that a RF is allowed to try in each individual tree. To be noted: **increasing the maximum number of features usually increases the models' performance, but this is not always true since it decreases the diversity of individual trees in the RF**.
- Number of estimators: the number of built trees build before taking the maximum voting or averages of predictions. Higher number of trees give you better performance but makes your code slower. **We suggest keeping this parameter as large as possible to optimize the performances**.
- Minimum sample leaf size: the leaf is the end of a decision tree. A smaller leaf makes the model more prone to capturing noise in train data. Most of the available studies suggest to keep a value larger than 50.

Advantages/disadvantages of RFs

Advantages

- The chance of overfitting decreases, since several different decision trees are used in the learning procedure. This corresponds to train and combine different models.
- RFs apply pretty well when a particular distribution of the data is not required. For example, **no data normalization is needed.**
- Parallelization: the training of multiple trees can be parallelized (for example through different computational slots)

Disadvantages

- **RFs usually might suffer from smaller training datasets**
- Interpretability: **RF is more a predictive tool than a descriptive tool**. It is difficult to see or understand the relationship between the response and the independent variables
- The time to train a RF algorithms might be longer compared to other algorithms. Also, in the case of a categorical variable, the time complexity increases exponentially

9.7.3 Artificial Neural Networks (ANNs)

Finding an agreed definition of ANNs is not that easy. In fact, most of the literature studies only provided graphical representations of ANN. The most used definition is the one by Haykin [10], who defines an ANN architecture as **a massively parallelized combination of very simple learning units that acquire knowledge during the training and store the knowledge by updating their connections to the other simple units**.

Often, ANNs have been compared to biological neural networks. Again, activities of biological neurons can then compared to 'activities' in processing elements

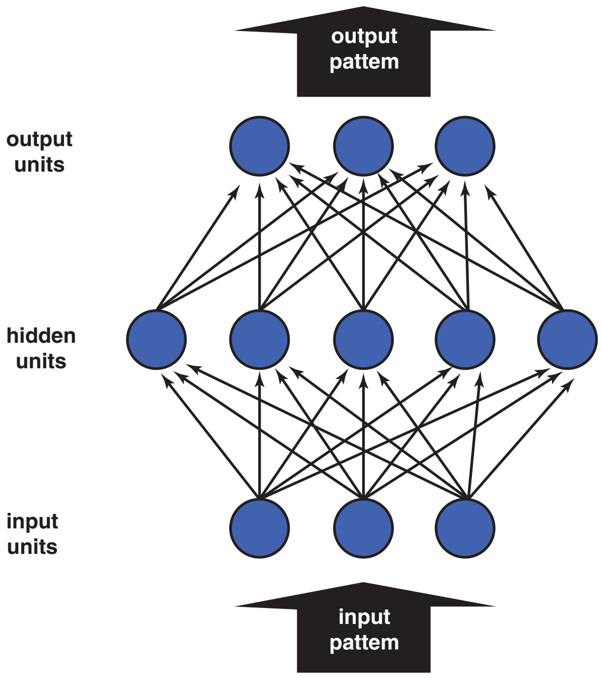

Fig. 9.3 Example architecture for an ANN. The standard architecture has one input layer, output units, and an intermediate layer called 'hidden units'

in ANNs. The process of training an ANN becomes then very similar on the process of learning in our brain: some inputs are applied to neurons, which change their weights during the training to produce the most optimized output.

One of the most important concepts in ANNs is their architecture. With the word **architecture we mean how the ANN is structured, meaning how many neurons are present and how they are connected.**

A typical architecture of ANNs is shown in Fig. 9.3: the input layer is characterized by input neurons, in our case the number of features we would like to input for our model. The output layer corresponds to the desired output of our model. In case of binary classification problems, for example, the output layer will only have two output neurons, but in case of multiple classifications, the number of output neurons can increase up to number of classes. In between, there is a 'hidden layer', where the number of hidden neurons can vary from few to thousands. Sometimes, in more complicated architectures, there might be several hidden layers.

ANNs are also classified according to the flow of the information:

1. Feed-forward neural networks: information travels only in one direction from input to output.
2. Recurrent neural networks: data flows in multiple directions. These are the most common used ANNs due to their capability of learning complex tasks such as for example handwriting or language recognition.

There is a 'hidden layer', where the number of hidden neurons can vary from few to thousands. Sometimes, in more complicated architectures, there might be several hidden layers.

'Deep learning' or 'Convolutional Neural Networks' (CNN) represents an example of very complicated ANNs [11]. The major feature of CNNs is that they can automatically learn significant features. In fact, each layer, during the training procedure, learns which the most representative features are. However, this might lead to use 'deep learning' as a 'black-box'. It is out of the topic of this chapter to dive into properties of CNNs. However, for the interested readers we suggest following references [12, 13].

What are the most important parameters in an ANN?

– Architecture: this is surely the most important parameter to be chosen in your ANN. There are no prescribed rules for choosing the number of neurons in the hidden layer. However, **please note that a very large number of neurons in the hidden layer (compared to the number of features) might increase the risk of overtraining.** We suggest trying different configurations and choosing the one that maximizes the accuracy in the testing set. Conversely, following rules apply for input and output layer: **the number of neurons in the input layer should be the same as the number of desired features**, while the number of output neurons should be equal to the number of classes we want to classify.
– Dropout: we suggest training an ANN **always with dropout set to true**. In fact, drop out is a technique to avoid overfitting [14]. **We suggest setting dropout between 20% and 50%.**
– Activation function: activation functions are used to **introduce nonlinearity** to models. The most common activation function is the rectifier activation function, but **we suggest to use the sigmoid function for binary predictions and the softmax function for multi class classification**.
– Network weight initialization: these are the weights used between neurons when starting the training. **The most common used is to initiate weights from an uniform distribution.**

Advantages/disadvantages of ANNs

Advantages

– **In principle, every kind of data can be used to feed an ANN**. No particular pre-processing of the data is required, but it is still suggested to use data that are normalized [15]. In addition, due to the complex structure of their architectures, ANNs can catch complex non linear relationships between independent and dependent variables
– **Ability to detect all possible interactions between predictor variables**: the hidden layer has the power to detect interrelations between all the input variables. For example, **when important relations are not modelled directly into a logistic regression model, neural networks are expected to perform better than logistic regression**.

Disadvantages

– **'Black box' approach**: in a logistic regression model, the developer is able to verify which variables are most predictable by looking at the coefficients of the odds ratios. Neural networks, compared to logistic regression are black boxes. In fact, after setting up the training data and the network parameters, the ANNs 'learn' by themselves which input variables are the most important. It is therefore impossible to determine how the variables contribute to the output. There is interest in the community to develop regression-like techniques to examine the connection weights and the relations between input features [16].

– **Large computational power required**: with a standard personal computer and with back propagation activated the training of a network might require from hours to some months compared to logistic regression.
– **Prone to overfitting**: the ability of an ANN to model interactions and non-linearity implicitly might represent a risk of overfitting. Suggestions to limit overfitting are: limiting the number of hidden layers and hidden neurons, adding a penalty function to the activation function for large weights, or limiting the amount of training using cross validation [17].

9.8 Unsupervised Algorithms

9.8.1 K-means

The goal of this algorithm is to find groups (or clusters) in data. The number of chosen group is defined by the variable K. The basic idea of the algorithm is to iteratively assign the data point to one of the K groups based on the features used as input [18]. The groups are assigned by similarities in the features values. The outputs of the algorithm are the centroids of each cluster K, and the labels for training data (each data point).

The algorithm workflow can be summarized as:

– data assignment step: each data is assigned to the nearest centroid based on the squared Euclidean distance
– centroid update step: the centroids are recomputed by taking the mean of data points assigned to a specific centroid's cluster.

The algorithm iterates between those two steps until the optimal solution (i.e. no data points change clusters) is found. Please note that there result is not a local optimum. The algorithm workflow is depicted in Fig. 9.4.

What are the most important parameters in k-means?

– Number of clusters K: there is no pre-defined rule to find the optimal number of K. Our suggestion is to iterate the algorithm several times and compare the results to find the best parameter**. One of the most common metrics used to compare results is the mean distance between the data points and their corresponding cluster centroids. However, this metric cannot be used as only indicator.** In fact, increasing the number of K will always decrease the distance until the extreme case where the distance is zero (K = number of data points). We suggest to plot the mean distance as a function of K; then the 'elbow point', where the

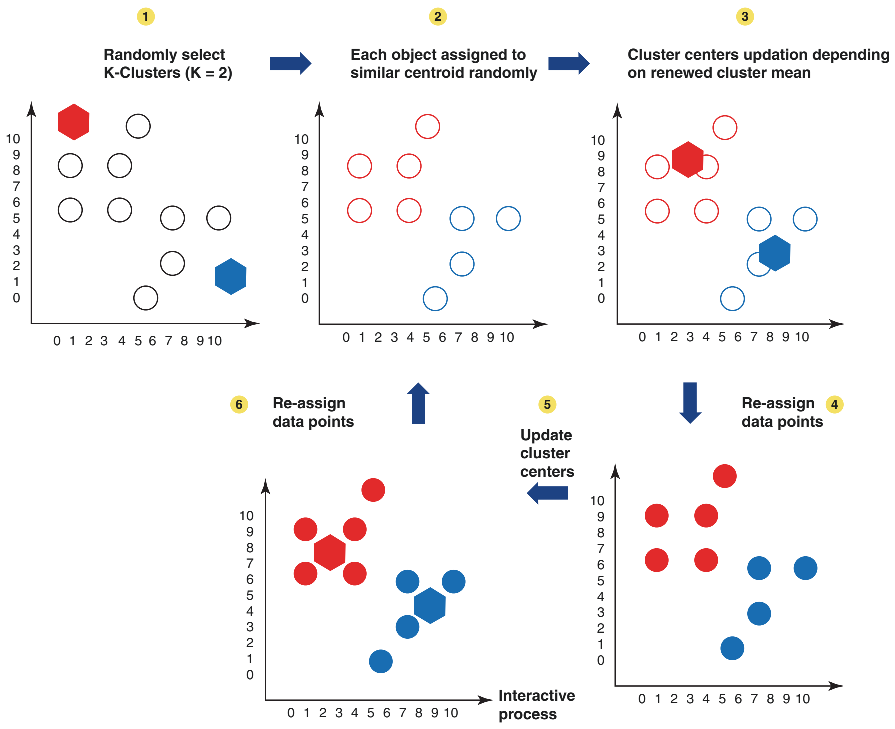
1
Randomly select K-Clusters (K = 2)
2
Each object assigned to similar centroid randomly
3
Cluster centers updation depending on renewed cluster mean
4
Re-assign data points
5
Update cluster centers
6
Re-assign data points
Interactive process
0 1 2 3 4 5 6 7 8 9 10

Fig. 9.4 Sketch of the k-means clustering algorithm

rate of decrease sharply shifts can used to determine K. Additional more advanced methods can be the silhouette method [19], and the G-means algorithm [20].

Advantages/disadvantages of k-means

Advantages

– In case of a large amount of data, **K-means is the faster algorithm between the families of unsupervised algorithms used for clustering problems**. For example, it is faster than hierarchical clustering. However, increasing K might increase the computational time.
– **K-means produce tighter clusters than the other algorithms in the category**

Disadvantages

– It is in general difficult to predict the optimal K and results might be strongly affected by different Ks
– If there is a big unbalance between data (or high number of outliers) the algorithm does not work well

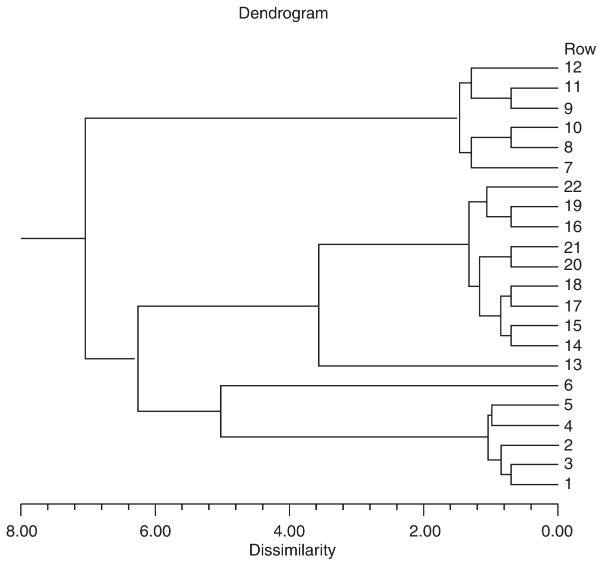

Fig. 9.5 Example of a dendrogram

9.8.2 Hierarchical Clustering

Compared to k-means, hierarchical clustering starts by assigning all data points as their own cluster. The basic idea of the algorithm is to build hierarchies and then assign points to clusters [21]. The workflow can be summarized as:

– Assign each data point to its own cluster
– Find the closest pair of cluster using Euclidean distance and merge them into one single cluster
– Calculate distance between two nearest clusters and combine until all items are clustered in to a single cluster.

What are the most important parameters in hierarchical clustering?

– Number of clusters: again, there is no general recipe on how to find the optimal number of clusters. However, we suggest to notice which vertical lines can be cut by horizontal line without intersecting a cluster and covers the maximum distance. This can be done by building a dendrogram [22]. An example of dendrogram is shown in Fig. 9.5.

Advantages/disadvantages of hierarchical clustering

Advantages

– The dendogram as algorithm output is quite understandable and easy to visualize it.

Disadvantages

– Compared to k-means, Time complexity of at least O(n^2 *log n*) is required, where *'n'* is the number of data points
– Much more sensitive to noise outliers

9.9 Conclusion

– Two major classes of machine learning algorithms exist**: supervised and unsupervised learning**. The first class is mainly used to predict outcomes by using some input features, the second class is used to cluster 'unlabeled data'.
– **There is no recipe for choosing a specific algorithm and there is no perfect algorithm**. You have been presented to major advantages and disadvantages of all the listed algorithms. **It is useful to remember that an extreme complexity of the algorithm might increase the risk of overfitting**
– We recommend the user to focus **on a very careful preparation of the data** before building a model (see previous chapters). In fact, a recent review [23] pointed out how classification algorithms suffer from quality of the input data.

References

1. Michalski RS, Carbonell JG, Mitchell TM. Machine learning: an artificial intelligence approach [Internet]. Berlin/Heidelberg: Springer Berlin Heidelberg; 1983. [cited 2018 Jun 5]. Available from: http://public.eblib.com/choice/publicfullrecord.aspx?p=3099788
2. Kubat M. An introduction to machine learning [Internet]. Cham: Springer International Publishing; 2017. [cited 2018 June 5]. Available from: http://link.springer.com/10.1007/978-3-319-63913-0
3. Caruana R, Niculescu-Mizil A. An empirical comparison of supervised learning algorithms. In ACM Press; 2006. p. 161–8. [cited 2018 June 5]. Available from: http://portal.acm.org/citation.cfm?doid=1143844.1143865
4. Hastie T, Tibshirani R, Friedman J. The elements of statistical learning [Internet]. New York: Springer New York; 2009 [cited 2018 Jun 5]. (Springer Series in Statistics). Available from: http://link.springer.com/10.1007/978-0-387-84858-7
5. Zhou X, Belkin M. Semi-supervised learning. In: Academic Press Library in Signal Processing [Internet]. Elsevier; 2014. p. 1239–69. [cited 2018 June 5]. Available from: http://linkinghub.elsevier.com/retrieve/pii/B978012396502800022X
6. Hearst MA, Dumais ST, Osuna E, Platt J, Scholkopf B. Support vector machines. IEEE Intell Syst Their Appl. 1998;13(4):18–28.
7. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in python. ArXiv12010490 Cs [Internet]; 2012. [cited 2018 June 5]. Available from: http://arxiv.org/abs/1201.0490
8. Paluszek M, Thomas S. MATLAB machine learning. Berkeley: Apress; 2017. p. 326.
9. Safavian SR, Landgrebe D. A survey of decision tree classifier methodology. IEEE Trans Syst Man Cybern. 1991;21(3):660–74.

10. Haykin SS, Haykin SS. Neural networks and learning machines. 3rd ed. New York: Prentice Hall; 2009. p. 906.
11. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436–44.
12. Schmidhuber J. Deep learning in neural networks: an overview. Neural Netw. 2015;61:85–117.
13. Greenspan H, van Ginneken B, Summers RM. Guest editorial deep learning in medical imaging: overview and future promise of an exciting new technique. IEEE Trans Med Imaging. 2016;35(5):1153–9.
14. Schittenkopf C, Deco G, Brauer W. Two strategies to avoid overfitting in feedforward networks. Neural Netw. 1997;10(3):505–16.
15. Basheer I, Hajmeer M. Artificial neural networks: fundamentals, computing, design, and application. J Microbiol Methods. 2000;43(1):3–31.
16. Schumacher M, Roßner R, Vach W. Neural networks and logistic regression: part I. Comput Stat Data Anal. 1996;21(6):661–82.
17. Smith M. Neural networks for statistical modeling. London: International Thomson Computer Press; 1996. p. 235.
18. Jain AK. Data clustering: 50 years beyond K-means. Pattern Recogn Lett. 2010;31(8):651–66.
19. Rousseeuw PJ. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math. 1987;20:53–65.
20. Hamerly G. Learning the k in k-means. Adv Neural Inf Process Syst. 2004;17:281–8.
21. Kaufman L, Rousseeuw PJ. Finding groups in data: an introduction to cluster analysis. Hoboken: Wiley; 2005. p. 342. (Wiley series in probability and mathematical statistics).
22. Hierarchical clustering. In: Wiley series in probability and statistics [Internet]. Chichester: John Wiley & Sons, Ltd; 2011. p. 71–110. [cited 2018 June 5]. Available from: http://doi.wiley.com/10.1002/9780470977811.ch4
23. Deist TM, Dankers FJWM, Valdes G, Wijsman R, Hsu I-C, Oberije C, et al. Machine learning algorithms for outcome prediction in (chemo)radiotherapy: an empirical comparison of classifiers. Med Phys [Internet]; 2018. [cited 2018 June 18]. Available from: http://doi.wiley.com/10.1002/mp.12967