Part IV

Classification Algorithms

10

Decision Trees and Random Forests

10.1 Introduction

In Part III of this book, we introduced feature learning and selection techniques for vibration data. In the fault-detection and -diagnosis problem framework using these vibration data, the next stage is classification. Classification is a typical supervised learning task: it categorises the acquired vibration signal correctly into the corresponding machine condition, which is generally a multiclass classification problem. As described in Chapter 6, supervised learning can be categorised into batch learning (i.e. offline learning) or online learning (i.e. incremental learning). In batch learning, the data points along with their corresponding labels are used together to learn and optimise the parameters of the model of interest. A simple example of classification is to assign a given vibration signal a 'normal' or 'fault' category. This can be seen in Figure 10.1, where the classifier is trained with several vibration signal examples, x_1, x_2, …, x_L along with their predefined labels, i.e. classes, (normal condition [NO], fault condition [FA]). The trained classifier then learns to classify new vibration examples, y_1, y_2, …, y_k to their corresponding classes, i.e. NO or FA.

The literature on classification algorithms has highlighted various techniques to deal with classification problems, e.g. k-nearest neighbours (k-NN) (Duda and Hart 1973); hierarchical-based models, decision trees (DTs) (Quinlan 1986), and random forests (RFs) (Breiman 2001); probability-based models, including naïve Bayes classification (Rish 2001) and logistic regression classification (Hosmer et al. 2013); support vector machines (SVMs) (Cortes and Vapnik 1995); layered models, e.g. artificial neural networks (ANNs) (Jain et al. 1996, 2014) and deep neural networks (DNNs) (Schmidhuber 2015), which can be used for both batch learning and online learning.

Various classification techniques can be used to classify different vibration types based on the features provided. If the vibration signals' features are carefully devised and the parameters of the classifiers are carefully tuned, it is possible to achieve high classification accuracies. This part of the book introduces some widely used state-of-the-art classifiers – decision trees/forests, multinomial logistic regression, SVMs, and ANNs – that have already been used for classification of vibration signal status. In addition to these classifiers, this part also describes recent trends of deep learning in the field of machine condition monitoring and provides an explanation of commonly used techniques and examples of their application in machine fault diagnosis.

To begin with, this chapter introduces DT and RF classifiers by giving the basic theory of the DT diagnosis tool, its data structure, the ensemble model that combines DTs into

Condition Monitoring with Vibration Signals: Compressive Sampling and Learning Algorithms for Rotating Machines, First Edition. Hosameldin Ahmed and Asoke K. Nandi.

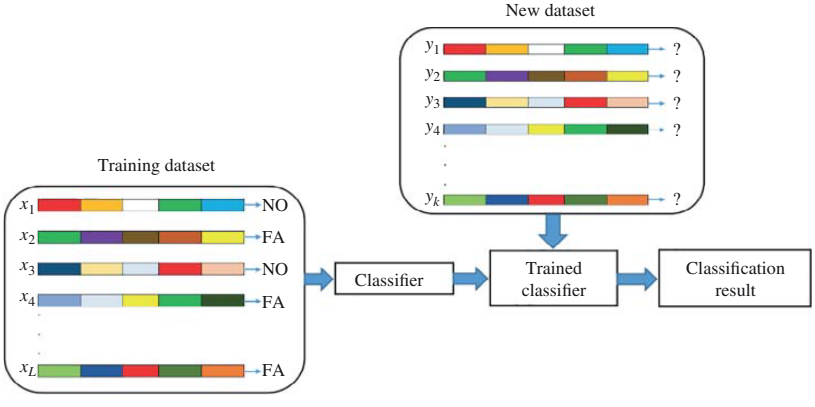

Figure 10.1 A simple example of classification to assign a given vibration signal to the 'normal' or 'fault' category. (See insert for a colour representation of this figure.)

a decision Forest model, and their applications in machine fault diagnosis. The other types of techniques, i.e. Multinomial logistic regression, SVM, ANN, and deep learning, will be covered in details in Chapters 11–14, respectively.

10.2 Decision Trees

The DT is one of the most popular tools in machine learning. There are two main types of DTs: classification trees used when the dependent variable is categorical, and regression trees used when the dependent variable is continuous (Loh 2011; Breiman 2017). Classification trees approximate the discrete classification function by using the tree-based representation. DTs are often constructed recursively using a top-down approach, by partitioning input training data into smaller subspaces until it reaches a subspace representing the most appropriate class label. In fact, most DT algorithms consist of two main phases: a building phase followed by a pruning phase. The stage of constructing a tree from training data is often called *tree induction*, *tree building,* or *tree growing*. Figure 10.2 shows a simple example of two classes, NO and FA, and two X variables, x_i^1 and x_i^2, where Figure 10.2a presents the data points and the partitions, and Figure 10.2b shows the corresponding DT. Using this classifier, we can easily predict the status of a signal of interest by passing it down the tree.

To deal with multiclass classification, we can use one tree and assign each leaf node a class where each leaf is selected by a class label; there also may be two or more leaves with the same class. For instance, Figure 10.3 shows a typical example of a six-class tree for rolling bearing health conditions. These include two normal conditions – brand new (NO), and worn but undamaged (NW) – and four fault conditions – inner race (IR), outer race (OR), rolling element (RE), and cage (CA).

DTs are well suited for those situations where the instances, i.e. the input, are represented by attribute pair values, i.e. data vectors. They can even tolerate some missing feature values, which happens when it comes to real-time data collection and feature

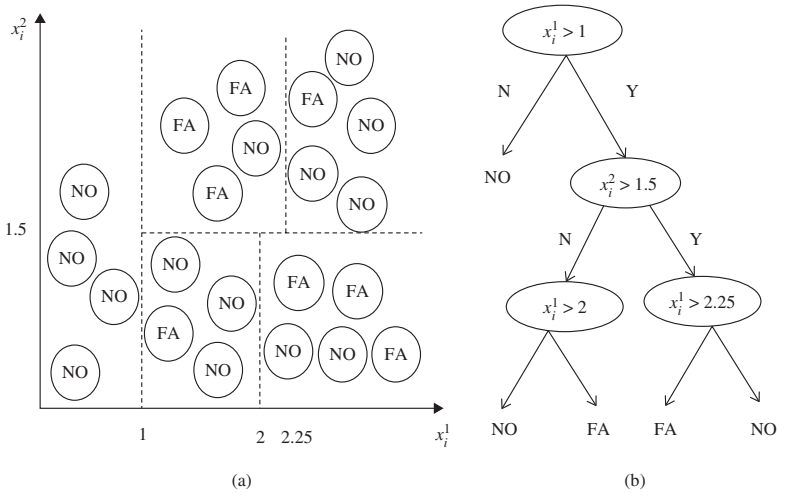

Figure 10.2 A simple example of two classes – normal condition (NO) and fault condition (FA) – and two *X* variables.

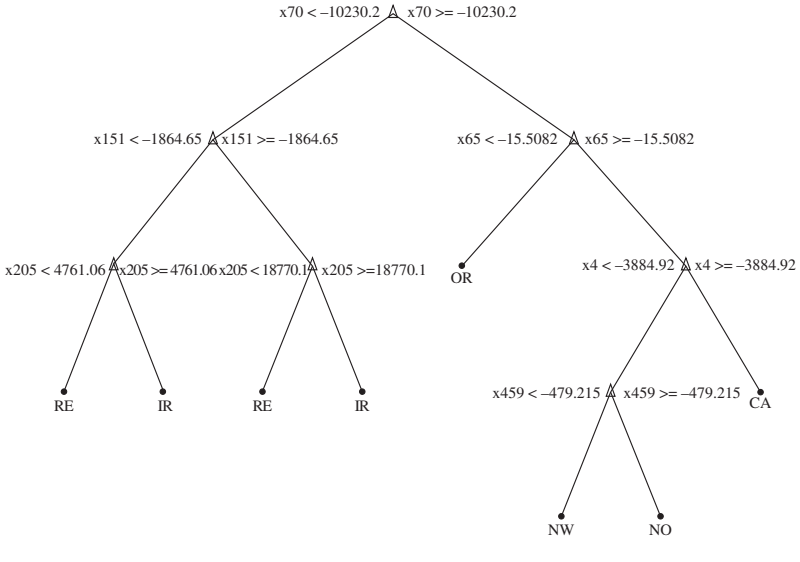

Figure 10.3 Atypical example of a six-class tree for rolling bearing health conditions.

extraction. The structure of a DT is composed of a root, several nodes that are also known as *indicators*, a number of branches, and a number of leaves that also known as *terminals* or *decision nodes*, which include class labels; branches represent the paths from the root to leaves, passing through nodes. The depth of a DT is the longest path from a root to a leaf (Rivest 1987). In the DT algorithm, the tree-based representation is learned from the input data by recognising the best splits and learning to classify the label, i.e. defining when a node is terminal. Each node of DT uses a splitting rule, e.g. GINI index of diversity, information gain, the Marshall correction, or a random selection of attributes, for splitting (Buntine and Niblett 1992; Jaworski et al. 2018), to split the instance space into two or more subspaces.

These splitting rules can be univariate (i.e. consider one attribute) or multivariate (consider multiple attributes) (Brodley and Utgoff 1995; Myles et al. 2004; Lee 2005). The main goal of these splitting rules is to find the best attribute that splits the training tuples into subsets. In DT classification, an attribute of a tuple is often categorical or numerical. However, there are various methods that have been proposed for constructing DTs from uncertain data (e.g. Tsang et al. 2011).

Moreover, DTs can be either binary, in which case the splitting rule splits the subspace into two parts, or multiway. In the binary DT induction method, a bi-partition at a node divides values into two subsets. A multiway split DT use as many partitions as possible (Biggs et al. 1991; Fulton et al. 1995; Kim and Loh 2001; Berzal et al. 2004; Oliveira et al. 2017).

In each DT, the instance is passed down from root to a leaf that includes a final decision, i.e. classification result. In each node in the tree, a test is carried out for a certain attribute of the instance, and each branch of this node corresponds to one possible value of the attribute. To describe the DT classification process in a more understandable way, it is a set of if-then rules. In fact, DTs are typically constructed recursively, following a greedy top-down construction approach. The top-down induction on decision tree (TDIDT) (Quinlan 1986) is a framework that refers to this type of algorithm. Here, the greedy approach defines the best available variable of the current split without considering future splits, which often represents the key idea of learning DTs.

The literature on DTs identified various methods for building DT models such as the classification and regression tree (CART) (Breiman et al. 1984), ID3 (Quinlan 1986), C4.5 (Quinlan 1993), and Chi-square automatic integration detector DT (CHAID) (Berry and Linoff 1997). To build a DT, it is essential to discover at each internal node a rule for splitting the data into subsets. In the case of univariate trees, one finds the attribute that is the most useful in discriminating the input data and finding a decision rule using the attribute. In the case of multivariate trees, one finds a combination of existing attributes that has the most useful discriminatory power (Murthy 1998). The subsequent subsections discuss the algorithms most commonly used by DTs to make splitting decisions. These are univariate and multivariate splitting techniques.

10.2.1 Univariate Splitting Criteria

Univariate splitting criteria consider one attribute to perform the splitting test in an internal node. Hence, the DT learning algorithm, i.e. the inducer, looks for the best available attribute that can be used for splitting. The ideal rule of splitting often results in class purity where all the cases in a terminal node come from the same class. For

example, only one node in Figure 10.2 is pure with NO status for all the instances; three of the four terminal nodes are impure. In order to optimise the DT, one can search for the splitting rule S_r that minimises the impurity function F_{imp}. An *impurity function*, F_{imp}, is a function defined on the set of all k-tuples of numbers $(p(c_1), p(c_2), \ldots, p(c_k))$ satisfying $p(c_i) \geq 0 \; \forall \; i \in \{1, \ldots, k\}$ and $\sum_{i=1}^{k} p(c_i) = 1$ such that

(a) An impurity function (F_{imp}) achieves its maximum only at the point $(\frac{1}{k}, \frac{1}{k}, \ldots, \frac{1}{k})$.
(b) F_{imp} achieves its minimum at the points (1, 0, …, 0), (0, 1, …, 0), …, (0, 0, …, 1), which represent the purity points.
(c) F_{imp} is a symmetric function of $(p(c_1), p(c_2), \ldots, p(c_k))$.

The impurity measure of any node N_1 can be defined using the following equation:

$$i(N_1) = F_{imp}(p(c_1 \mid N_1), p(c_2 \mid N_1), \ldots, p(c_k \mid N_1)) \tag{10.1}$$

If a splitting rule S_r in node N_1 divides all the instances into two subsets N_{1L} and N_{1R}, the decrease in impurity can be defined using the following equation,

$$\Delta i(S_r, N_1) = i(N_1) - P_L i(N_{1L}) - P_R i(N_{1R}) \tag{10.2}$$

where P_L and P_R are proportions of N_{1L} and N_{1R}, respectively. If a test in a node N_j is based on an attribute having n values, Eq. (10.2) can be generalised as Eq. (10.3) (Raileanu and Stoffel 2004):

$$\Delta i(S_r, N_1) = i(N_i) - \sum_{j=1}^{n} P(N_j) i(N_j) \tag{10.3}$$

The literature on DTs has highlighted several impurity-based measures that can be used for splitting. The subsequent subsections discuss the most commonly used measures in more detail.

10.2.1.1 Gini Index

The Gini index is an impurity-based measure for categorical data. It was originally a measure of the probability of misclassification that was developed by the Italian statistician Corrado Gini in 1912. It was proposed for DTs by Breiman et al. (1984) as a measure of node impurity. The Gini index measures the deviations between the probability distributions of the target attribute's values. Given a node N, an impurity function based on Gini index measure assigns a training instance to a class label c_i with probability $p(c_i \mid N)$. Hence, the probability that the instance is class j can be estimated by $p(c_j \mid N)$. Based on this rule, the estimated probability of misclassification is the Gini index, which can be expressed mathematically as in Eq. (10.4):

$$\begin{aligned} Gini(c, N) &= \sum_{i=1}^{k} \sum_{j=1, j \neq i}^{k} p(c_i \mid N) p(c_j \mid N) = \sum_{i=1}^{k} p(c_i \mid N)(1 - p(c_i \mid N)) \\ &= 1 - \sum_{i=1}^{k} (p(c_i \mid N))^2 \end{aligned} \tag{10.4}$$

When a node N is pure, the Gini index value is zero. For example, using the Gini index from Eq. (10.4), the impurity of the pure node in Figure 10.2 described with NO status

for all the instances is $(1-(4/4)^2=0)$. For the root node that satisfies $x_i^1 > 2.25$, which has four instances with NO status and one instance with FA status (see Figure 10.2a), the Gini index is $(1-(4/5)^2-(1/5)^2=0.32)$. If a splitting rule S_r in node N divides all the instances into two subsets N_L and N_R, then the Gini index can be computed using Eq. (10.5),

$$Gini_A(S_r, N) = \frac{|N_1|}{|N|} Gini(N_1) + \frac{|N_2|}{|N|} Gini(N_2) \tag{10.5}$$

The reduction in impurity can be expressed mathematically using Eq. (10.6):

$$\Delta Gini\,(A) = Gini(S_r, N) - (Gini_A(S_r, N) \tag{10.6}$$

An alternative to the Gini index is the twoing criterion, which is a binary criterion that can be defined using the following equation:

$$Twoing(N) = 0.25\left(\frac{|N_0|}{|N|}\right)\left(\frac{|N_1|}{|N|}\right)\left(\sum_{c_i \in C} p_{N_0,c_i} - p_{N_1,c_i}\right)^2 \tag{10.7}$$

10.2.1.2 Information Gain

Information gain (IG) is a measure of how much information a feature provides about the class. It is an impurity-based measure that utilises the entropy as a measure of impurity. The information entropy can be represented mathematically using Eq. (10.8),

$$Info(N) = -\sum_{i=1}^{C} p(N, c_i) log_2(p(N, c_i)) \tag{10.8}$$

where C is the number of classes and $p(N, c_i)$ is the proportion of cases in N that belong to c_i. The corresponding information gained with k outcomes can be defined as the difference between the entropy before the split and the entropy after the split. This can be represented mathematically using Eq. (10.9) (Quinlan 1996a,b):

$$IG(N) = Info(N) - \sum_{i=1}^{k} \frac{|N_i|}{|N|} Info(N_i) \tag{10.9}$$

Even though IG is a good impurity-based measure, it biases toward attributes with many values. To overcome this problem, Quinlan proposed a new measure called the *information gain ratio* that penalises IG for selecting attributes that generate many small subsets (Quinlan 1986). The gain ratio can be expressed mathematically using Eq. (10.10):

$$GR\,(N) = \frac{IG(N)}{SplitINFO} \tag{10.10}$$

Here, *SplitINFO* represents the split information value that denotes the potential information generated by splitting the training data N into k subsets such that

$$SplitINFO\,(N, A) = \sum_{i=1}^{k} \frac{|N_i|}{|N|} \log \frac{|N_i|}{|N|} \tag{10.11}$$

where A is candidate attribute, i is a possible value of A, N is a set of examples, and N_i is subset when $X_A = i$.

Another technique for measuring the statistical importance of the IG is the likelihood-ratio, which can be defined as (Rokach and Maimon 2005b)

$$G^2(a_i, N) = 2.\ln(2).|N|.IG(a_i, N) \tag{10.12}$$

10.2.1.3 Distance Measure

As an alternative selection criterion, De Mántaras proposed a feature-selection measure for DT induction based on the distance between partitions. Based on this measure, the selected feature in a node induces the partition that is closest (in terms of the distance) to the correct partition of the subset of examples in this node (De Mántaras 1991). The distance measure between partitions generated by feature A and class C can be expressed using Eq. (10.13),

$$(A, C) = 1 - \frac{IG(A, C)}{Info(A, c)} \tag{10.13}$$

where $Info(A, c)$ is a joint entropy of A and C that can be defined using Eq. (10.14):

$$Info(A, C) = \sum_a \sum_c p_{ac} log_2 p_{ac} \tag{10.14}$$

10.2.1.4 Orthogonal Criterion (ORT)

The orthogonal criterion (ORT) is a binary criterion that can be defined using the following equation (Fayyad and Irani 1992):

$$ORT(N) = 1 - COS\theta(p_{c,N_1}, p_{c,N_2}) \tag{10.15}$$

Here, $\theta(p_{c,N_1}, p_{c,N_2})$ is the angle between the probability distribution of the target attribute in partitions N_1 and N_2.

There are also many other univariate splitting measures that have been proposed for DTs. Because of the limitations on space, we cannot detail them here. Readers who are interested are referred to (Rounds 1980; Li and Dubes 1986; Taylor and Silverman 1993; Dietterich et al. 1996; Friedman 1977; Ferri et al. 2002). Moreover, comparative studies of splitting criteria have been conducted (e.g. Safavian and Landgrebe 1991; Buntine and Niblett 1992; Breiman 1996b; Shih 1999; Drummond and Holte 2000; Rokach and Maimon 2005a,b).

10.2.2 Multivariate Splitting Criteria

Unlike univariate splits that consider one attribute to perform the splitting test in an internal node, multivariate splits consider a number of features to be used in a single node split test. Most of the multivariate splits are based on the linear combination of the input features. In fact, finding the best multivariate split is more difficult than finding the best univariate split. In the literature of multivariate splits, various methods have been used for finding the best linear combination. These include greedy search, linear programming, linear discriminant analysis, perceptron training, hill climbing search, etc. Table 10.1 shows a summary of most commonly used methods for multivariate splits in different studies of DTs.

In each case, i.e. univariate or multivariate splitting, the splitting continues until a stopping criterion is satisfied. A simple stopping criterion is the condition of node purity where all the instances in the node belong to only one class. In this case, there

Table 10.1 Summary of the most commonly used methods for multivariate splits in different studies of decision trees.

Method	Studies
Greedy learning	Breiman et al. 1984; Murthy 1998.
Non-greedy learning	Norouzi et al. 2015.
Linear programming	Lin and Vitter 1992; Bennett 1992; Bennett and Mangasarian 1992, 1994; Brown and Pittard 1993; Michel et al. 1998.
Linear discriminant analysis	You and Fu 1976; Friedman 1977; Qing-Yun and Fu 1983; Loh and Vanichsetakul 1988; Likura and Yasuoka 1991; Todeschini and Marengo 1992; John 1996; Li et al. 2003.
Perceptron learning	Utgoff 1989; Heath et al. 1993; Sethi and Yoo 1994; Shah and Sastry 1999; Bennett et al. 2000; Bifet et al. 2010.
Hill-climbing search	Murphy and Pazzani 1991; Brodley and Utgoff 1992; Murthy et al. 1994.

is no need for further splitting. Other stopping rules include: (i) when the number of cases in a node is less than a predefined N_{stop}, the node is declared terminal (Bobrowski and Kretowski 2000); (ii) when the tree depth, i.e. the longest path from a root to a leaf is reached; (iii) when the optimal splitting criterion is not larger than a predefined threshold; and (iv) when the number of nodes, i.e. the predefined maximum number of nodes, has been reached.

10.2.3 Tree-Pruning Methods

As mentioned earlier, the most common DT induction contains two main stages: (i) building a complete tree able to classify all training instances, and (ii) pruning the built tree back. Pruning methods originally proposed in (Breiman et al., 1984) were widely used to avoid overfitting, i.e. obtain the right-sized tree, which is a pruned version of the original built tree, also called a *subtree* (Mingers 1989; Murthy 1998). In reality, using the splitting measures presented, the decision-growing process will continue splitting until a stopping criterion is reached. Nevertheless, the procedure for DT building may lead to a large tree size and/or overfitting. Pruning methods can reduce overfitting and tree size. In fact, pruning methods are basically tree growing in reverse and are also described as *bottom-up pruning* (Kearns and Mansour 1998). Using a tight stopping measure often results in creating small, underfitted DTs, while using a loose stopping measure often result in generating large, overfitted DTs. To solve this problem, the overfitted tree is reduced into a smaller tree (subtree) by removing sub-branches that are not contributing to generalization accuracy (Rokach and Maimon 2005a).

The literature for DTs has defined various methods for pruning DTs. The subsequent subsections detail the most commonly used techniques.

10.2.3.1 Error-Complexity Pruning

Error-complexity pruning, also known as *cost-complexity pruning* (Breiman et al., 1984), is a two-stage error-minimisation pruning method. In the first stage, a series of trees $T_1, T_2, \ldots, T_m$ is generated on training examples, where $T_1$is the original tree before

pruning. Then, T_{i+1} is generated by replacing one or more of the subtrees in the previous T_i where $T_{i+1} < T_i$. For example, we can generate $T_2 < T_1$ by pruning away the branch ζ_{T_1} of T_1 such that,

$$T_2 = T_1 - \zeta_{T_1} \tag{10.16}$$

In general, this can be represented using the following equation:

$$T_{i+1} = T_i - \zeta_{T_i} \tag{10.17}$$

Continuing this way, we generate a decreasing series of subtrees such that $T_1 > T_2 > \ldots > T_m$. For each internal, the pruned branches are those that achieve the lowest increase in apparent error rate per pruned leaf. The cost-complexity criterion for a subtree $T \subset T_0$ can be computed using the following equation,

$$C_a(T) = C(T) + a \mid \widetilde{\zeta}(T) \mid \tag{10.18}$$

where $C(T)$ is the misclassification of a DT T, $\widetilde{\zeta}(T)$ is the number of terminal nodes, T_0 is the original tree before pruning, and a is a complexity parameter. The error-complexity, cost-complexity function C_a of pruned subtree $T - T_\zeta$ to that of the branch at node ζ can be represented using Eq. (10.19):

$$g(\zeta) = \frac{C(T) - C(T_\zeta)}{|\widetilde{\zeta}(T)| - 1} \tag{10.19}$$

In the second stage, for each internal node, the error complexity is computed and the one with the smallest value is converted to a leaf node. The best-pruned tree is then selected.

10.2.3.2 Minimum-Error Pruning

The minimum-error pruning method (Cestnik and Bratko 1991) is a bottom-up technique to select a single tree with the minimum error on the training dataset. With m probability estimates, the expected error in a given node rate can be represented using the following equation,

$$E_\zeta = \frac{N - n_c + (1 - p_{ac})m}{N + m} \tag{10.20}$$

where N is the total number of instances in the node, n_c is the number of instances in class c that minimise the expected error, p_{ac} is the a prior probability of class c, and m is the parameter of the estimation method.

10.2.3.3 Reduced-Error Pruning

The reduced-error pruning is a simple direct technique for tree pruning, proposed by Quinlan (1987). In this technique, rather than generating a series of trees and then selecting one of them, a more direct process is used. The simple idea is to assess each non-terminal node with respect to the classification error in a separate test set, also called the *pruning set*. The assessment examines the change in the classification error over the pruning set that may happen if the non-terminal node is replaced by the best possible leaf. If the new tree would result in an equal or lower number of errors, then the subtree of this non-terminal node is replaced by the leaf.

10.2.3.4 Critical-Value Pruning

Critical-value pruning (Mingers 1987) sets a threshold, also called the *critical value*, to examine the importance of a non-terminal node in a DT. If the node does not reach the threshold, it will be pruned; and if the node satisfies this condition, it will then be kept. However, if a node meets the pruning condition but the latter nodes, i.e. its children, do not satisfy the pruning condition, this subtree should be kept. Based on this technique, the larger the threshold value selected, the smaller the resulting tree that is formed.

10.2.3.5 Pessimistic Pruning

Pessimistic pruning (Quinlan 1993) uses the pessimistic statistical correlation test instead of the pruning set or cross-validation. It uses a measure called *continuity correction* for a binomial distribution to find a realistic estimate of the misclassification rate $r(\zeta)$ such that,

$$r(\zeta) = \frac{e(\zeta)}{N(\zeta)} \tag{10.21}$$

where $e(\zeta)$ is the number of misclassified instances at node ζ and $N(\zeta)$ is the number of training set instances at node ζ. The misclassification rate using the continuity correction can be represented using the following equation:

$$\widetilde{r}(\zeta) = \frac{e(\zeta) + 1/2}{N(\zeta)} \tag{10.22}$$

The misclassification rate for a subtree T_ζ can be represented using Eq. (10.23):

$$\widetilde{r}(T_\zeta) = \frac{\sum e(i) + \widetilde{\zeta}(T)/2}{\sum N(i)} \tag{10.23}$$

From Eqs. (10.22) and (10.23), $N(\zeta) = \sum N(i)$, as they define the same set of instances. Hence, the misclassification rates in Eqs. (10.22) and (10.23)v can be represented as the number of misclassifications for a node and for a subtree using Eqs. (10.24) and (10.25), respectively:

$$\widetilde{n}(\zeta) = e(\zeta) + 1/2 \tag{10.24}$$

$$\widetilde{n}(T_\zeta) = \frac{\sum e(i) + \widetilde{\zeta}(T)/2}{\sum N(i)} \tag{10.25}$$

Based on this technique, a subtree is kept if its corrected number of misclassifications is less than that of a node by at least one standard error; otherwise it is pruned. The standard error for the number of misclassifications can be computed using the following equation:

$$STE(\widetilde{n}(T_\zeta)) = \sqrt{\frac{\widetilde{n}(T_\zeta)(N(\zeta)) - \widetilde{n}(T_\zeta)}{N(\zeta)}} \tag{10.26}$$

10.2.3.6 Minimum Description Length (MDL) Pruning

The minimum description length (MDL)-based DT pruning method (Mehta et al. 1995) uses the MDL criterion to find a model within a class that allows the shortest encoding of the class series in the training instances. Given a DT node ζ containing N training

examples, belonging to class labels $c_1, c_2, \ldots, c_k$, the encoding length of the classification of all instances of ζ can be represented mathematically using the following equation:

$$L_c(\zeta) = \log\left(\frac{N}{N_{c_1}, N_{c_2}, \ldots, N_{c_k}}\right) + log\left(\frac{N+k-1}{k-1}\right) \tag{10.27}$$

The MDL-based method chooses DTs that can be encoded with fewer bits, i.e. the shortest encoding of the class labels.

There are also many other pruning methods reported in the literature. Readers who are interested are referred to (Wallace and Patrick 1993; Bohanec and Bratko 1994; Almuallim 1996; Fournier and Crémilleux 2002). Moreover, several studies have been conducted to compare the performance of various pruning methods (Quinlan 1987; Mingers 1989; Esposito et al. 1997; Knoll et al. 1994; Patil et al. 2010).

10.2.4 Decision Tree Inducers

DT inducers are methods that can be used to build a DT from a given training dataset. The literature on DTs has highlighted various methods for DT induction. We briefly describe some of the commonly used DT induction methods, including CART (Breiman et al. 1984), ID3 (Quinlan 1986), C4.5 (Quinlan 1993), and CHAID (Berry and Linoff 1997). The procedures for these methods include the application of the splitting criterion and pruning techniques that have been described.

10.2.4.1 CART

CART (Breiman et al. 1984) is one of the most commonly used methods for DT induction. It creates binary trees from training data: i.e. the splitting rule splits each internal node ζ into two parts ζ_L and ζ_R that can be used for both classification and regression problems. Both the Gini index and twoing criteria have been implemented in CART. To handle missing measurement values, CART uses a technique of surrogate splits, i.e. if a case has x_i missing in its measurement, one decides whether it goes to ζ_L and ζ_R by using the best surrogate split. The built tree is then pruned using cost-complexity pruning.

10.2.4.2 ID3

The iterative Dichotomiser 3 (ID3) (Quinlan 1986) is a simple DT induction method that uses IG as splitting criteria, which uses the greatest attribute of IG as a root node of a DT, but does not apply any pruning processes and does not handle missing values. In this method, the DT stops growing when all the instances in the node belong to only one class or when the IG is not bigger than zero. The main disadvantage of the ID3 method is that it requires the data description to include only discrete features.

10.2.4.3 C4.5

C4.5 is a DT induction method proposed by Quinlan in 1993. It is an extension of the DT inducer ID3 (Quinlan 1986) that deals with both continuous and discrete features. Also, unlike ID3, C4.5 handle missing values and apply the pruning process after DT construction. It uses the gain ratio as a splitting criterion and performs error-based pruning after the DT growing phase. In C4.5, the DT stops growing when the number of examples to be split is less than a predefined threshold (Quinlan 1993).

The commercial version of C4.5 is C5.0, which is a developed version that is more efficient than C4.5. Also, J48 is an open source Java implementation of the C4.5 method in the Weka data-mining tool (Xiaoliang et al. 2009; Moore et al. 2009). A good comparative study of ID3, CART, and C4.5 can be found in (Singh and Gupta 2014).

10.2.4.4 CHAID

CHAID, which is a modification of the automatic interaction detector (AID), is one of the earliest tree classification methods (Kass 1980). The basic idea of this method is to find the pair of values with the least significant difference for each input feature x_i with respect to the target attribute using the p value of the statistical test. Pearson's Chi-squared test is used to measure the significant difference. For each selected pair, the obtained p value is compared to a certain merge threshold, i.e. a predefined level of significance. If the p value is larger than the merge threshold, CHAID merges the values and searches for a further possible pair to be merged. This procedure is repeated until the pairs for which the p value is smaller than the defined level of significance are not identified. CHAID does not apply any pruning processes, but it handles missing values by considering them a single category.

Several advantages and disadvantages of DTs have been reported in the literature. The following is a summary of these advantages and disadvantages:

(a) Advantages of DTs:
    1. Easy to build, easy to use, and easy to interpret.
    2. Have the ability to handle both numerical and categorical features, e.g. CART.
    3. Have the ability to handle missing measurement values.
    4. Have the ability to handle outliers.
(b) Disadvantages of DTs:
    1. Some of the DT algorithms, e.g. ID3 and C4.5, deal only with target attributes that have only discrete values.
    2. Cause overfitting.
    3. The prediction model becomes unstable with data that has a small variance.

There are also many other DT inducers reported in the literature, including the fast algorithm for classification trees (Loh and Vanichsetakul., 1988); multivariate adaptive regression splines (Friedman 1991); CAL5 (Muller and Wysotzki 1994); quick, unbiased, efficient, statistical tree (QUEST) (Loh and Shih 1997); PUBLIC (Rastogi and Shim 1998); classification rule with unbiased interaction selection and estimation (CRUISE) (Kim and Loh 2001); and conditional inference trees (CTree) (Hothorn et al. 2015).

10.3 Decision Forests

As described, DTs are easy to build, easy to use, and easy to interpret. However, in practice, they have two main issues: (i) using one tree easily causes overfitting, and (ii) they are unstable. Significant improvements in classification accuracy have resulted from building an ensemble of trees and allowing them to vote for the most popular class (Breiman 2001). A number of independent methods have been proposed to build ensembles of trees. In these methods, (i) the original training dataset is randomly divided into several subsets that can be used to induce multiple classifiers. Then, (ii) a combination process is used to produce a single classification for a given example. Various techniques

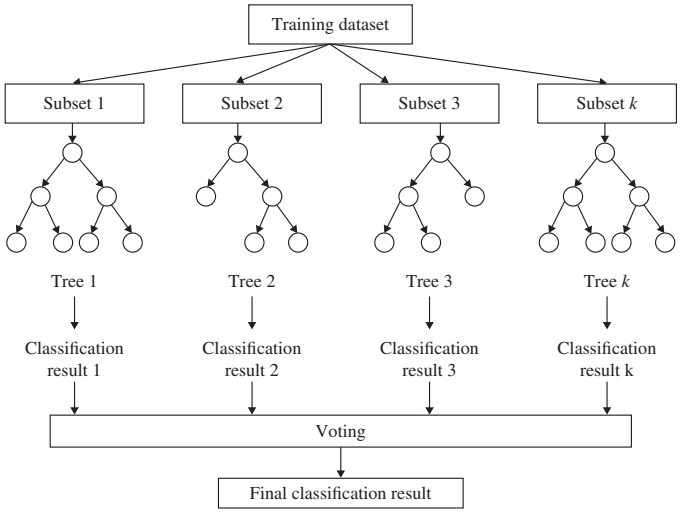

Figure 10.4 Atypical example of a random forest classifier.

have been used for building ensembles, e.g. bagging (Breiman 1996a); wagging, which is a variant of bagging (Bauer and Kohavi 1999); and boosting-based methods, including AdaBoost and Arc-x4 (Quinlan 1996a,b; Freund and Schapire 1997). An empirical comparison of these methods, i.e. bagging, boosting, and their variants, has been conducted by Freund and Schapire (Freund and Schapire 1997). The basic idea of all these methods is that for the kth tree, a random vector v_k is produced, independent of the past random vectors $v_1, v_2, v_3, \ldots, v_{k-1}$ but with the same distribution. With these generated random vectors, the generated k trees are used to vote for the most popular class label (Breiman 2001). These processes are called random forests (RFs). In other words, an RF (see Figure 10.4) is an ensemble learning method that constructs multiple DTs during the training process; the final output is the mode of the classes output by each individual DT (Nandi et al. 2013).

Breiman showed that RFs are an effective tool in prediction. Also, he explained why RFs do not overfit as more trees are added, but generate a limiting value of the generalization error (Breiman 2001).

Due to the compressibility and ease of interpreting their results, DTs are being considered in a large diversity of applications, e.g. automatic modulation recognition (Nandi and Azzouz 1995, 1998; Azzouz and Nandi 1996); bioinformatics (Che et al. 2011); image processing (Lu and Yang 2009); and medicine (Podgorelec et al. 2002). The following subsection presents their application in machine fault diagnosis.

10.4 Application of Decision Trees/Forests in Machine Fault Diagnosis

DTs and RFs are popular techniques that have been applied to machine fault diagnosis. Many studies related to machine fault diagnosis have been published. In this section,

a brief discussion of the application of DTs and RFs in machine fault diagnosis is presented.

The early application of DTs for fault diagnosis was proposed by Patel and Kamrani in 1996 (Patel and Kamrani 1996). The study presented an intelligent decision support system for diagnosis and maintenance of automated systems called ROBODOC, which is a DT-based system that uses the shallow knowledge of a maintenance expert in the form of impulse factor (IF) … THEN rules. Yang et al. (2005) proposed an expert system called VIBEX for vibration fault diagnosis of rotating machinery using DT and decision tables. In this method, a decision table based on the cause-symptom matrix is used as the probabilistic method for diagnosing abnormal vibrations; and a DT is introduced to build a knowledge base, which is essential for vibration expert systems. VIBEX embeds a cause-result matrix containing 1800 confidence factors, which makes it suitable to monitor and diagnosis rotating machinery.

Tran et al. (2006) proposed the application of the DT method to classify the faults of induction motors. In this method, feature extraction is applied beforehand to extract useful information from the raw data using statistical feature parameters from the time domain and frequency domain. Then, a DT method is applied as a classification model for fault diagnosis of induction motors with vibration signals and current signals. The experimental results showed that the DT achieved high fault-classification accuracies for induction motors. Sugumaran and Ramachandran proposed automatic rule learning using ID3 DTs for a fuzzy classifier in fault diagnosis of roller bearings (Sugumaran and Ramachandran 2007). In this method, a DT is used to generate rules automatically from the feature set. Statistical features are extracted from the collected vibration data, and good features that discriminate the different fault conditions of the bearing are selected using a DT. The rule set for the fuzzy classifier is obtained once again using the DT. The results are found to be encouraging. However, this method requires a large number of data points in the dataset to achieve good results.

Sun et al. (2007) proposed a method based on C4.5 DTs and principal component analysis (PCA) for rotating machinery fault diagnosis. In this method, PCA is used to extract the features of the acquired data. Then C4.5 is trained by using the samples to generate a DT with diagnosis knowledge. Finally, the tree model is used to make a diagnosis. Six kinds of running conditions including normal, unbalance, rotor radial rub, oil whirl, shaft crack, and a simultaneous state of unbalance and radial rub, are simulated to validate the proposed method. The result showed the efficiency of the proposed method for rotating machinery fault diagnosis.

Yang et al. (Yang et al. 2008) investigated the possibilities of applying the RF algorithm in machine fault diagnosis and proposed a hybrid method combined with the genetic algorithm (GA) to improve classification accuracy. The proposed method is demonstrated by a case study of induction motor fault diagnosis. (i) The experimental results showed that the normal RF achieved satisfactory fault diagnosis accuracy. (ii) By applying GA to do the parameter optimisation of the number of trees and random split number of the RF, the experimental results showed that classification accuracy achieved 98.89%, which is 3.33% higher than the best classification accuracy achieved by the normal RF.

Yang and colleagues (Yang et al. 2009) proposed a fault-diagnosis method based on an adaptive neuro-fussy inference system (ANFIS) in combination with DTs. In this

method, CART, which is one of the DT methods, is used as a feature-selection procedure to select pertinent features from the dataset.

Saravanan and Ramachandran (2009) proposed a method for fault diagnosis of a spur bevel gearbox using a discrete wavelet and DT. In this method, the discrete wavelet is used for feature extraction and a J48 DT is used for feature selection as well as for classification. The analysis of vibration signals produced by a bevel gearbox in various conditions and faults is used to validate this method. (i) Statistical features are extracted for all the wavelet coefficients of the vibration signals using Daubechies wavelet db1–db15. (ii) J48 is used for feature selection and classification of various conditions of the gearbox. The experimental results showed that the maximum average classification accuracy achieved was 98.57%.

Sakthivel et al. (2010a,b) proposed the use of a C4.5 DT algorithm for fault diagnosis of monoblock centrifugal pumps through statistical features extracted from vibration signals under good and faulty conditions. The experimental results showed that C4.5 and the vibration signals are good candidates for practical applications of fault diagnosis of monoblock centrifugal pumps. Sakthivel et al. (2010b) also presented the utilization of DTs and rough sets to generate the rules from statistical features extracted from vibration signals under normal and faulty conditions of a monoblock centrifugal pump. In this study, a fuzzy classifier is built using a DT and rough set rules and tested using test data. The experimental results showed that overall classification accuracy obtained using the DT-fuzzy hybrid system is better than the overall classification accuracy obtained using the rough set-fuzzy hybrid system.

Saimurugan et al. (2011) presented the use of c-support vector classification (c-SVC) and nu-SVC models of SVM with four kernel functions for classification of faults using statistical features extracted from vibration signals under good and faulty conditions of the rotational mechanical systems. The DT algorithm (C.4.5 algorithm) was used to select the prominent features. These features were given as input for training and testing the c-SVC and nu-SVC models of SVM, and their fault classification accuracies were compared. Seera et al. (2012) proposed a method called FMM-CART to detect and classify comprehensive fault conditions of induction motors using a hybrid fuzzy min-max (FMM) neural network and CART. In this method, the motor current harmonics are employed to form the input features to FMM-CART for detection and diagnosis of individual and multiple motor fault conditions. The process of FMM-CART starts with FMM, which is formed using hyper-box fuzzy sets; then the hyper boxes generated from FMM training serve as input to the CART DT. The final tree is then used for fault classification. The experimental results showed that FMM-CART is able to achieve 98.25% accuracy for five motor conditions in noise-free conditions. Moreover, Seera et al. (2017) proposed a hybrid model, FMM-RF, for classification of ball bearing faults with vibration signals. This method is based on the FMM neural network and the RF model. In this method, power spectrum and sample entropy features are used during feature extraction, where important features are extracted. With these features, the resulting hyper boxes from FMM are used as input to CART, where the centroid point and the confidence factor are used for tree building. Then, bagging is conducted for RF, and the majority voting scheme is used to combine the predictions from an ensemble of trees. Experiments of benchmark and real-world dataset showed accurate performance using the FMM-RF model. The best classification accuracies obtained from benchmark and real-world datasets were 99.9% and 99.8%, respectively.

Karabadji et al. (2012) developed a GA-based technique for DT selection in industrial machine fault diagnosis. In this technique, for DT selection, GAs are used to validate DT performance on training and testing sets to achieve the most robustness. Several trees are generated, and then the GA is used to find the best representative tree. The experimental results of vibration signals from an industrial fan using RF for fault diagnosis demonstrated robustness and good performance. An enhanced version has the DT evolve using a genetic programming paradigm, which has been used for fault classification (Guo et al. 2005). Cerrada et al. (2016) proposed a two-step fault-diagnosis method for spur gears. In this method, the diagnostic system is performed using a GA and a classifier based on the RF.

Muralidharan and Sugumaran (2013) proposed a fault-diagnosis algorithm for monoblock centrifugal pumps using wavelets and DT. In this method, the continuous wavelet transform (CWT) is calculated from the acquired vibration signals using different families and different levels, which form the feature set. These features are then fed as input to the classifier J48, and the classification accuracies are calculated. The experimental results showed the efficiency of the proposed method in monoblock fault diagnosis. Jegadeeshwaran and Sugumaran proposed a vibration-based monitoring technique for automobile hydraulic brake systems. In this technique, the C4.5 DT algorithm is used to extract statistical features from vibration signals; feature selection is also carried out. The selected features are classified using the C4.5 DT algorithm and best-first DT algorithm, with pre-pruning and post-pruning techniques. The experimental results showed that the best-first DT classifier with pre-pruning is more accurate compared to the best-first DT classifier with post-pruning and C4.5 DT classifier (Jegadeeshwaran and Sugumaran 2013). Saimurugan et al. (2015) conducted a study of the classification ability of the DT and SVM in gearbox fault detection. This study includes two fault classes, two gear speeds (first and fourth gear), and three loading conditions. Feature extraction was performed using statistical features, and classification accuracies were obtained using both the SVM and J48 DT methods.

Li et al. (2016) presented a method called deep RF fusion (DRFF) for gearbox fault diagnosis using acoustic and vibration signals simultaneously. In this method, the statistical parameters of the wavelet packet transform (WPT) are first obtained from the acoustic and vibration signals respectively. Then, two deep Boltzmann machines (DBMs) are developed for deep representations of the WPT statistical parameters. Finally, the RF is used to fuse the outputs of the two DBMs. Experimental results of gearbox fault diagnosis under different operating conditions showed that the DRFF has the ability to improve the performance of gearbox fault diagnosis.

Zhang et al. (2017) proposed a fault-diagnosis method based on the fast-clustering algorithm and DT for rotating machinery with imbalanced data. In this method, the fast-clustering algorithm is used to search for essential samples from the majority data of the imbalanced fault sample set. Hence, the balanced fault sample set comprises the clustered data, and the minority data is built. Then, a DT is trained using the balanced fault sample set to obtain the fault diagnosis model. The experimental results of the gearbox fault dataset and rolling bearing fault dataset showed that the proposed fault-diagnosis model has the ability to diagnose rotating machinery faults for imbalanced data.

Table 10.2 Summary of some of the introduced techniques and their publically accessible software.

Algorithm name	Platform	Package	Function
Fit binary classification decision tree for multiclass classification	MATLAB	Statistics and Machine Learning Tool-box – Classification – Classification trees	fitctree
Produce sequence of subtrees by pruning			prune
Compact tree			compact
Mean predictive measure of association for surrogate splits in decision tree			surrogateAssociation
Decision tree and decision forest	MATLAB	Wang (2014)	RunDecisionForest TrainDecisionForest

10.5 Summary

This chapter has provided a review of DTs and RFs and their application in machine fault diagnosis, including several DT analysis techniques. The algorithms most commonly used by DTs to make splitting decisions were described. These include univariate splitting criteria, e.g. Gini index, information gain, distance measure, and orthogonal criterion; and multivariate splitting criteria, e.g. greedy learning, linear programming, linear discriminant analysis, perceptron learning, and hill-climbing search. Moreover, DT pruning methods including error-complexity pruning, minimum-error pruning, critical-value pruning, etc., were presented. In addition, the most commonly used DT inducers, i.e. methods that can be used to build a DT from a given training dataset, e.g. CART, the iterative Dichotomiser 3 (ID3), C4.5, and CHAID were briefly described. The procedures for these methods include the application of the splitting criteria and pruning techniques that have been described.

Also, a brief discussion of the application of DTs and RFs in machine fault diagnosis was provided in this chapter. Some of the introduced techniques and their publicly accessible software are summarised in Table 10.2.

References

Almuallim, H. (1996). An efficient algorithm for optimal pruning of decision trees. *Artificial Intelligence* 83 (2): 347–362. Elsevier.

Azzouz, E.E. and Nandi, A.K. (1996). Modulation recognition using artificial neural networks. In: *Automatic Modulation Recognition of Communication Signals*, 132–176. Boston, MA: Springer.

Bauer, E. and Kohavi, R. (1999). An empirical comparison of voting classification algorithms: bagging, boosting, and variants. *Machine Learning* 36 (1–2): 105–139. Springer.

Bennett, K.P. (1992). *Decision Tree Construction Via Linear Programming*, 97–101. Center for Parallel Optimization, Computer Sciences Department, University of Wisconsin.

Bennett, K.P. and Mangasarian, O.L. (1992). Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software* 1 (1): 23–34. Taylor & Francis.

Bennett, K.P. and Mangasarian, O.L. (1994). Multicategory discrimination via linear programming. *Optimization Methods and Software* 3 (1–3): 27–39. Taylor & Francis.

Bennett, K.P., Cristianini, N., Shawe-Taylor, J., and Wu, D. (2000). Enlarging the margins in perceptron decision trees. *Machine Learning* 41 (3): 295–313. Springer.

Berry, M.J. and Linoff, G. (1997). *Data Mining Techniques: For Marketing, Sales, and Customer Support*. Wiley.

Berzal, F., Cubero, J.C., Marın, N., and Sánchez, D. (2004). Building multi-way decision trees with numerical attributes. *Information Sciences* 165 (1–2): 73–90. Elsevier.

Bifet, A., Holmes, G., Pfahringer, B., and Frank, E. (2010). Fast perceptron decision tree learning from evolving data streams. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 299–310. Berlin, Heidelberg: Springer.

Biggs, D., De Ville, B., and Suen, E. (1991). A method of choosing multiway partitions for classification and decision trees. *Journal of Applied Statistics* 18 (1): 49–62. Taylor & Francis.

Bobrowski, L. and Kretowski, M. (2000). Induction of multivariate decision trees by using dipolar criteria. In: *European Conference on Principles of Data Mining and Knowledge Discovery*, 331–336. Berlin, Heidelberg: Springer.

Bohanec, M. and Bratko, I. (1994). Trading accuracy for simplicity in decision trees. *Machine Learning* 15 (3): 223–250. Springer.

Breiman, L. (1996a). Bagging predictors. *Machine Learning* 24: 123–140. Springer.

Breiman, L. (1996b). Some properties of splitting criteria. *Machine Learning* 24 (1): 41–47. Springer.

Breiman, L. (2001). Random forests. *Machine Learning* 45 (1): 5–32. Springer.

Breiman, L. (2017). *Classification and Regression Trees*. Routledge Taylor & Francis.

Breiman, L., Friedman, J.H., Olshen, R.A. et al. (1984). Classification and regression trees. Wadsworth International Group. LHCb collaboration.

Brodley, C.E. and Utgoff, P.E. (1992). *Multivariate Versus Univariate Decision Trees*. Amherst, MA: University of Massachusetts, Department of Computer and Information Science.

Brodley, C.E. and Utgoff, P.E. (1995). Multivariate decision trees. *Machine Learning* 19 (1): 45–77. Springer.

Brown, D.E. and Pittard, C.L. (1993). Classification trees with optimal multi-variate splits. In: *International Conference on Systems, Man and Cybernetics, 1993.'Systems Engineering in the Service of Humans,' Conference Proceedings*, vol. 3, 475–477. IEEE.

Buntine, W. and Niblett, T. (1992). A further comparison of splitting rules for decision-tree induction. *Machine Learning* 8 (1): 75–85. Springer.

Cerrada, M., Zurita, G., Cabrera, D. et al. (2016). Fault diagnosis in spur gears based on genetic algorithm and random forest. *Mechanical Systems and Signal Processing* 70: 87–103. Elsevier.

Cestnik, B. and Bratko, I. (1991). On estimating probabilities in tree pruning. In: *European Working Session on Learning*, 138–150. Berlin, Heidelberg: Springer.

Che, D., Liu, Q., Rasheed, K., and Tao, X. (2011). Decision tree and ensemble learning algorithms with their applications in bioinformatics. In: *Software Tools and Algorithms for Biological Systems*, 191–199. New York, NY: Springer.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning* 20 (3): 273–297. Springer.

De Mántaras, R.L. (1991). A distance-based attribute selection measure for decision tree induction. *Machine Learning* 6 (1): 81–92. Springer.

Dietterich, T., Kearns, M., and Mansour, Y. (1996). Applying the weak learning framework to understand and improve C4. 5. In: *International Conference on Machine Learning (ICML)*, 96–104.

Drummond, C. and Holte, R.C. (2000). Exploiting the cost (in) sensitivity of decision tree splitting criteria. *Proceedings of the International Conference on Machine Learning* 1 (1): 239–246.

Duda, R.O. and Hart, P.E. (1973). *Pattern Classification and Scene Analysis. A Wiley-Interscience Publication*. New York: Wiley.

Esposito, F., Malerba, D., Semeraro, G., and Kay, J. (1997). A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19 (5): 476–491.

Fayyad, U.M. and Irani, K.B. (1992). The attribute selection problem in decision tree generation. In: *American Association for Artificial Intelligence (AAAI)*, 104–110.

Ferri, C., Flach, P., and Hernández-Orallo, J. (2002). Learning decision trees using the area under the ROC curve. In: *International Conference on Machine Learning (ICML)*, vol. 2, 139–146.

Fournier, D. and Crémilleux, B. (2002). A quality index for decision tree pruning. *Knowledge-Based Systems* 15 (1–2): 37–43. Elsevier.

Freund, Y. and Schapire, R.E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55 (1): 119–139. Elsevier.

Friedman, J.H. (1977). A recursive partitioning decision rule for nonparametric classification. *IEEE Transactions on Computers* (4): 404–408.

Friedman, J.H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*: 1–67.

Fulton, T., Kasif, S., and Salzberg, S. (1995). Efficient algorithms for finding multi-way splits for decision trees. In: *Machine Learning Proceedings 1995*, 244–251.

Guo, H., Jack, L.B., and Nandi, A.K. (2005). Feature generation using genetic programming with application to fault classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 35 (1): 89–99.

Heath, D., Kasif, S., and Salzberg, S. (1993). Induction of oblique decision trees. In: *International Joint Conferences on Artificial Intelligence (IJCAL)*, vol. 1993, 1002–1007.

Hosmer, D.W. Jr., Lemeshow, S., and Sturdivant, R.X. (2013). *Applied Logistic Regression*, vol. 398. Wiley.

Hothorn, T., Hornik, K. and Zeileis, A. (2015). ctree: Conditional Inference Trees. The Comprehensive R Archive Network.

Jain, A.K., Mao, J., and Mohiuddin, K.M. (1996). Artificial neural networks: a tutorial. *Computer* 29 (3): 31–44.

Jain, L.C., Seera, M., Lim, C.P., and Balasubramaniam, P. (2014). A review of online learning in supervised neural networks. *Neural Computing and Applications* 25 (3–4): 491–509. Springer.

Jaworski, M., Duda, P., and Rutkowski, L. (2018). New splitting criteria for decision trees in stationary data streams. *IEEE Transactions on Neural Networks and Learning Systems* 29 (6): 2516–2529.

Jegadeeshwaran, R. and Sugumaran, V. (2013). Comparative study of decision tree classifier and best first tree classifier for fault diagnosis of automobile hydraulic brake system using statistical features. *Measurement* 46 (9): 3247–3260. Elsevier.

John, G.H. (1996). Robust linear discriminant trees. In: *Learning from Data*, 375–385. New York, NY: Springer.

Karabadji, N.E.I., Seridi, H., Khelf, I., and Laouar, L. (2012). Decision tree selection in an industrial machine fault diagnostics. In: *International Conference on Model and Data Engineering*, 129–140. Berlin, Heidelberg: Springer.

Kass, G.V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied statistics*: 119–127. Wiley.

Kearns, M.J. and Mansour, Y. (1998). A fast, bottom-up decision tree pruning algorithm with near-optimal generalization. In: *International Conference on Machine Learning (ICML)*, vol. 98, 269–277.

Kim, H. and Loh, W.Y. (2001). Classification trees with unbiased multiway splits. *Journal of the American Statistical Association* 96 (454): 589–604. Taylor & Francis.

Knoll, U., Nakhaeizadeh, G., and Tausend, B. (1994). Cost-sensitive pruning of decision trees. In: *European Conference on Machine Learning*, 383–386. Berlin, Heidelberg: Springer.

Lee, S.K. (2005). On generalized multivariate decision tree by using GEE. *Computational Statistics & Data Analysis* 49 (4): 1105–1119. Elsevier.

Li, X. and Dubes, R.C. (1986). Tree classifier design with a permutation statistic. *Pattern Recognition* 19 (3): 229–235. Elsevier.

Li, X.B., Sweigart, J.R., Teng, J.T. et al. (2003). Multivariate decision trees using linear discriminants and tabu search. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 33 (2): 194–205.

Li, C., Sanchez, R.V., Zurita, G. et al. (2016). Gearbox fault diagnosis based on deep random forest fusion of acoustic and vibratory signals. *Mechanical Systems and Signal Processing* 76: 283–293. Elsevier.

Likura, Y. and Yasuoka, Y. (1991). Utilization of a best linear discriminant function for designing the binary decision tree. *International Journal of Remote Sensing* 12 (1): 55–67. Taylor & Francis.

Lin, J.H. and Vitter, J.S., 1992. Nearly optimal vector quantization via linear programming.

Loh, W.Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1 (1): 14–23. Wiley.

Loh, W.Y. and Shih, Y.S. (1997). Split selection methods for classification trees. *Statistica Sinica*: 815–840.

Loh, W.Y. and Vanichsetakul, N. (1988). Tree-structured classification via generalized discriminant analysis. *Journal of the American Statistical Association* 83 (403): 715–725.

Lu, K.C. and Yang, D.L. (2009). Image processing and image mining using decision trees. *Journal of Information Science & Engineering* (4): 25. Citeseer.

Mehta, M., Rissanen, J., and Agrawal, R. (1995). MDL-based decision tree pruning. In: *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD)*, vol. 21, No. 2, 216–221.

Michel, G., Lambert, J.L., Cremilleux, B., and Henry-Amar, M. (1998). A new way to build oblique decision trees using linear programming. In: *Advances in Data Science and Classification*, 303–309. Berlin, Heidelberg: Springer.

Mingers, J. (1987). Expert systems—rule induction with statistical data. *Journal of the Operational Research Society* 38 (1): 39–47. Springer.

Mingers, J. (1989). An empirical comparison of pruning methods for decision tree induction. *Machine Learning* 4 (2): 227–243. Springer.

Moore, S.A., D'addario, D.M., Kurinskas, J., and Weiss, G.M. (2009). Are decision trees always greener on the open (source) side of the fence? In: *Proceedings of DMIN*, 185–188.

Muller, W. and Wysotzki, F. (1994). A splitting algorithm, based on a statistical approach in the decision tree algorithm CAL5. In: *Proceedings of the ECML-94 Workshop on Machine Learning and Statistics* (eds. G. Nakhaeizadeh and C. Taylor).

Muralidharan, V. and Sugumaran, V. (2013). Feature extraction using wavelets and classification through decision tree algorithm for fault diagnosis of mono-block centrifugal pump. *Measurement* 46 (1): 353–359. Elsevier.

Murphy, P.M. and Pazzani, M.J. (1991). ID2-of-3: Constructive induction of M-of-N concepts for discriminators in decision trees. In: *Machine Learning Proceedings 1991*, 183–187. Elsevier.

Murthy, S.K. (1998). Automatic construction of decision trees from data: a multi-disciplinary survey. *Data Mining and Knowledge Discovery* 2 (4): 345–389. Springer.

Murthy, S.K., Kasif, S., and Salzberg, S. (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research* 2: 1–32.

Myles, A.J., Feudale, R.N., Liu, Y. et al. (2004). An introduction to decision tree modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society* 18 (6): 275–285. Wiley.

Nandi, A.K. and Azzouz, E.E. (1995). Automatic analogue modulation recognition. *Signal Processing* 46 (2): 211–222. Elsevier.

Nandi, A.K. and Azzouz, E.E. (1998). Algorithms for automatic modulation recognition of communication signals. *IEEE Transactions on Communications* 46 (4): 431–436.

Nandi, A.K., Liu, C., and Wong, M.D. (2013). Intelligent vibration signal processing for condition monitoring. In: *Proceedings of the International Conference Surveillance*, vol. 7, 1–15.

Norouzi, M., Collins, M., Johnson, M.A. et al. (2015). Efficient non-greedy optimization of decision trees. In: *NIPS'15 Proceedings of the 28th International Conference on Neural Information Processing Systems* - Volume 1, 1729–1737.

Oliveira, W.D., Vieira, J.P., Bezerra, U.H. et al. (2017). Power system security assessment for multiple contingencies using multiway decision tree. *Electric Power Systems Research* 148: 264–272. Elsevier.

Patel, S.A. and Kamrani, A.K. (1996). Intelligent decision support system for diagnosis and maintenance of automated systems. *Computers & Industrial Engineering* 30 (2): 297–319. Elsevier.

Patil, D.D., Wadhai, V.M., and Gokhale, J.A. (2010). Evaluation of decision tree pruning algorithms for complexity and classification accuracy. *International Journal of Computer Applications* 11 (2).

Podgorelec, V., Kokol, P., Stiglic, B., and Rozman, I. (2002). Decision trees: an overview and their use in medicine. *Journal of Medical Systems* 26 (5): 445–463. Springer.

Qing-Yun, S. and Fu, K.S. (1983). A method for the design of binary tree classifiers. *Pattern Recognition* 16 (6): 593–603. Elsevier.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning* 1 (1): 81–106. Springer.

Quinlan, J.R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies* 27 (3): 221–234. Elsevier.

Quinlan, J.R. (1993). *C4. 5: Programs for Machine Learning*. Elsevier.

Quinlan, J.R. (1996a). Bagging, boosting, and C4. 5. In: *AAAI'96 Proceedings of the thirteenth national conference on Artificial intelligence* - Volume 1, 725–730.

Quinlan, J.R. (1996b). Improved use of continuous attributes in C4. 5. *Journal of Artificial Intelligence Research* 4: 77–90.

Raileanu, L.E. and Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence* 41 (1): 77–93. Springer.

Rastogi, R. and Shim, K. (1998). PUBLIC: a decision tree classifier that integrates building and pruning. In: *Proceedings of the International Conference on Very Large Data Bases (VLDB)* , vol. 98, 24–27.

Rish, I. (2001). An empirical study of the naive Bayes classifier. In: *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, vol. 3, No. 22, 41–46. New York: IBM.

Rivest, R.L. (1987). Learning decision lists. *Machine Learning* 2 (3): 229–246. Springer.

Rokach, L. and Maimon, O. (2005a). Decision trees. In: *Data Mining and Knowledge Discovery Handbook*, 165–192. Boston, MA: Springer.

Rokach, L. and Maimon, O. (2005b). Top-down induction of decision trees classifiers-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 35 (4): 476–487.

Rounds, E.M. (1980). A combined nonparametric approach to feature selection and binary decision tree design. *Pattern Recognition* 12 (5): 313–317. Elsevier.

Safavian, S.R. and Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics* 21 (3): 660–674.

Saimurugan, M., Ramachandran, K.I., Sugumaran, V., and Sakthivel, N.R. (2011). Multi component fault diagnosis of rotational mechanical system based on decision tree and support vector machine. *Expert Systems with Applications* 38 (4): 3819–3826. Elsevier.

Saimurugan, M., Praveenkumar, T., Krishnakumar, P., and Ramachandran, K.I. (2015). A study on the classification ability of decision tree and support vector machine in gearbox fault detection. In: *Applied Mechanics and Materials*, vol. 813, 1058–1062. Trans Tech Publications.

Sakthivel, N.R., Sugumaran, V., and Babudevasenapati, S. (2010a). Vibration based fault diagnosis of monoblock centrifugal pump using decision tree. *Expert Systems with Applications* 37 (6): 4040–4049. Elsevier.

Sakthivel, N.R., Sugumaran, V., and Nair, B.B. (2010b). Comparison of decision tree-fuzzy and rough set-fuzzy methods for fault categorization of mono-block centrifugal pump. *Mechanical Systems and Signal Processing* 24 (6): 1887–1906. Elsevier.

Saravanan, N. and Ramachandran, K.I. (2009). Fault diagnosis of spur bevel gear box using discrete wavelet features and decision tree classification. *Expert Systems with Applications* 36 (5): 9564–9573. Elsevier.

Schmidhuber, J. (2015). Deep learning in neural networks: an overview. *Neural Networks* 61: 85–117. Elsevier.

Seera, M., Lim, C.P., Ishak, D., and Singh, H. (2012). Fault detection and diagnosis of induction motors using motor current signature analysis and a hybrid FMM–CART model. *IEEE Transactions on Neural Networks and Learning Systems* 23 (1): 97–108.

Seera, M., Wong, M.D., and Nandi, A.K. (2017). Classification of ball bearing faults using a hybrid intelligent model. *Applied Soft Computing* 57: 427–435. Elsevier.

Sethi, I.K. and Yoo, J.H. (1994). Design of multicategory multifeature split decision trees using perceptron learning. *Pattern Recognition* 27 (7): 939–947. Elsevier.

Shah, S. and Sastry, P.S. (1999). New algorithms for learning and pruning oblique decision trees. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 29 (4): 494–505.

Shih, Y.S. (1999). Families of splitting criteria for classification trees. *Statistics and Computing* 9 (4): 309–315. Springer.

Singh, S. and Gupta, P. (2014). Comparative study ID3, cart and C4. 5 decision tree algorithm: a survey. *International Journal of Advanced Information Science and Technology (IJAIST)* 27 (27): 97–103.

Sugumaran, V. and Ramachandran, K.I. (2007). Automatic rule learning using decision tree for fuzzy classifier in fault diagnosis of roller bearing. *Mechanical Systems and Signal Processing* 21 (5): 2237–2247. Elsevier.

Sun, W., Chen, J., and Li, J. (2007). Decision tree and PCA-based fault diagnosis of rotating machinery. *Mechanical Systems and Signal Processing* 21 (3): 1300–1317. Elsevier.

Taylor, P.C. and Silverman, B.W. (1993). Block diagrams and splitting criteria for classification trees. *Statistics and Computing* 3 (4): 147–161. Springer.

Todeschini, R. and Marengo, E. (1992). Linear discriminant classification tree: a user-driven multicriteria classification method. *Chemometrics and Intelligent Laboratory Systems* 16 (1): 25–35. Elsevier.

Tran, V.T., Yang, B.S., and Oh, M.S. (2006). Fault diagnosis of induction motors using decision trees. In: *Proceeding of the KSNVE Annual Autumn Conference*, 1–4.

Tsang, S., Kao, B., Yip, K.Y. et al. (2011). Decision trees for uncertain data. *IEEE Transactions on Knowledge and Data Engineering* 23 (1): 64–78.

Utgoff, P.E. (1989). Perceptron trees: a case study in hybrid concept representations. *Connection Science* 1 (4): 377–391. Taylor & Francis.

Wallace, C.S. and Patrick, J.D. (1993). Coding decision trees. *Machine Learning* 11 (1): 7–22.

Wang, Q. (2014). Decision tree and decision forest. Mathworks File Exchange Center. https://uk.mathworks.com/matlabcentral/fileexchange/39110-decision-tree-and-decision-forest.

Xiaoliang, Z., Hongcan, Y., Jian, W., and Shangzhuo, W. (2009). Research and application of the improved algorithm C4. 5 on decision tree. In: *International Conference on Test and Measurement, 2009. ICTM'09*, vol. 2, 184–187. IEEE.

Yang, B.S., Lim, D.S., and Tan, A.C.C. (2005). VIBEX: an expert system for vibration fault diagnosis of rotating machinery using decision tree and decision table. *Expert Systems with Applications* 28 (4): 735–742. Elsevier.

Yang, B.S., Di, X., and Han, T. (2008). Random forests classifier for machine fault diagnosis. *Journal of Mechanical Science and Technology* 22 (9): 1716–1725. Springer.

Yang, B.S., Oh, M.S., and Tan, A.C.C. (2009). Fault diagnosis of induction motor based on decision trees and adaptive neuro-fuzzy inference. *Expert Systems with Applications* 36 (2): 1840–1849. Elsevier.

You, K.C. and Fu, K.S. (1976). An approach to the design of a linear binary tree classifier. In: *Proc. Symp. Machine Processing of Remotely Sensed Data*, 3A–10A.

Zhang, X., Jiang, D., Long, Q., and Han, T. (2017). Rotating machinery fault diagnosis for imbalanced data based on decision tree and fast clustering algorithm. *Journal of Vibroengineering* 19 (6): 4247–4259.