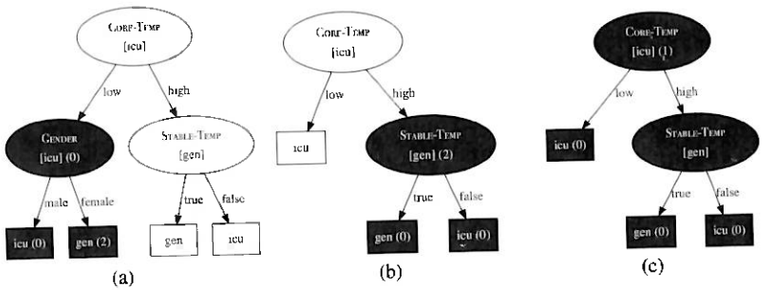

Figure 4.19

The iterations of reduced error pruning for the decision tree in Figure 4.18[161] using the validation set in Table 4.13[161]. The subtree that is being considered for pruning in each iteration is highlighted in black. The prediction returned by each non-leaf node is listed in square brackets. The error rate for each node is given in round brackets.

the predictions made at the leaf nodes of this subtree are incorrect for $\mathbf{d}_2$ and $\mathbf{d}_5$ (because these patients are *female*, the prediction made is *gen* which does not match the validation dataset), so the error rate for the leaf nodes of this subtree is $0 + 2 = 2$. Because the error rate for the leaf nodes is higher than the error rate for the root node of the subtree, this subtree is pruned and replaced by a leaf node. The result of this pruning is visible on the left branch of the tree in Figure 4.19(b)[162].

In the second iteration of the algorithm, the subtree under the STABLE-TEMP node is considered for pruning (highlighted in Figure 4.19(b)[162]). In this instance, the error rate for the root node of this subtree (the STABLE-TEMP node) is 2, whereas the error rate of the leaf nodes of the tree is $0 + 0 = 0$. As the error rate of the root node of the subtree is higher than the error rate of the leaf nodes, the tree is not pruned. Figure 4.19(c)[162] illustrates the final iteration of the algorithm. In this iteration the subtree underneath the root node of the decision tree (the CORE-TEMP node) is considered for pruning (i.e., the full decision tree). In this iteration, the error rate of the root node (1) is greater than the error rate of the three leaf nodes, $(0 + 0 + 0 = 0)$, so the tree is left unchanged.

Post-pruning using an error rate criteria is probably the most popular way to prune decision trees.[24] One of the advantages of pruning decision trees is that

24 See Esposito et al. (1997) and Mingers (1989) for overviews and empirical comparisons of a range of decision tree pruning methods based on error rate.

it keeps trees smaller, which in turn makes them easier to interpret. Another advantage is that pruning often increases the accuracy of the trees when there is noise in the training dataset. This is because pruning typically affects the lower parts of the decision tree, where noisy training data is most likely to cause overfitting. As such, pruning can be viewed as a **noise dampening mechanism** that removes nodes that have been created because of a small set of noisy instances.

4.4.5 Model Ensembles

Much of the focus of machine learning is on developing the single most accurate prediction model possible for a given task. The techniques we introduce in this section take a slightly different approach. Rather than creating a single model, they generate a set of models and then make predictions by aggregating the outputs of these models. A prediction model that is composed of a set of models is called a **model ensemble**.

The motivation behind using ensemble methods is the idea that a committee of experts working together on a problem are more likely to solve it successfully than a single expert working alone. As is always the case when a committee is working together, however, steps should be taken to guard against **group think**. In the context of ensemble models, this means that each model should make predictions independently of the other models in the ensemble. Given a large population of independent models, an ensemble can be very accurate even if the individual models in the ensemble perform only marginally better than random guessing.

There are two defining characteristics of ensemble models:

1. They build multiple different models from the same dataset by inducing each model using a modified version of the dataset.
2. They make a prediction by aggregating the predictions of the different models in the ensemble. For categorical target features, this can be done using different types of voting mechanisms, and for continuous target features, this can be done using a measure of the central tendency of the different model predictions, such as the mean or the median.

There are two standard approaches to creating ensembles: **boosting** and **bagging**. The remainder of this section explains each of these.

4.4.5.1 Boosting

When we use **boosting**,[25] each new model added to an ensemble is biased to pay more attention to instances that previous models misclassified. This is done by incrementally adapting the dataset used to train the models. To do this we use a **weighted dataset** where each instance has an associated weight $\mathbf{w}_i \geqslant 0$, initially set to $\frac{1}{n}$ where n is the number of instances in the dataset. These weights are used as a distribution over which the dataset is sampled to create a **replicated training set**, in which the number of times an instance is replicated is proportional to its weight.

Boosting works by iteratively creating models and adding them to the ensemble. The iteration stops when a predefined number of models have been added. During each iteration the algorithm does the following:

1. Induces a model using the weighted dataset and calculates the total error, ϵ, in the set of predictions made by the model for the instances in the training dataset.[26] The ϵ value is calculated by summing the weights of the training instances for which the predictions made by the model are incorrect.
2. Increases the weights for the instances misclassified by the model using

$$\mathbf{w}[i] \leftarrow \mathbf{w}[i] \times \left(\frac{1}{2 \times \epsilon}\right) \tag{4.12}$$

   and decreases the weights for the instances correctly classified by the model using[27]

$$\mathbf{w}[i] \leftarrow \mathbf{w}[i] \times \left(\frac{1}{2 \times (1-\epsilon)}\right) \tag{4.13}$$

3. Calculates a **confidence factor**, α, for the model such that α increases as ϵ decreases. A common way to calculate the confidence factor is

$$\alpha = \frac{1}{2} \times log_e\left(\frac{1-\epsilon}{\epsilon}\right) \tag{4.14}$$

Once the set of models has been created, the ensemble makes predictions using a weighted aggregate of the predictions made by the individual models.

25 Schapire (1999) gives a readable introduction to boosting by one of the originators of the technique.

26 Normally in machine learning, we do not test a model using the same dataset that we use to train it. Boosting, however, is an exception to this rule.

27 Updating the weights using Equations (4.12)[163] and (4.13)[164] ensures that the weights always sum to 1.

The weights used in this aggregation are the confidence factors associated with each model. For categorical target features, the ensemble returns the majority target level using a weighted vote, and for continuous target features, the ensemble returns the weighted mean.

4.4.5.2 Bagging

When we use **bagging** (or **bootstrap aggregating**), each model in the ensemble is trained on a random sample[28] of the dataset where, importantly, each random sample is the same size as the dataset and **sampling with replacement** is used. These random samples are known as **bootstrap samples**, and one model is induced from each bootstrap sample. The reason that we sample with replacement is that this will result in duplicates within each of the bootstrap samples, and consequently, every bootstrap sample will be missing some of the instances from the dataset. As a result, each bootstrap sample will be different, and this means that models trained on different bootstrap samples will also be different.[29]

Decision tree induction algorithms are particularly well suited to use with bagging. This is because decision trees are very sensitive to changes in the dataset: a small change in the dataset can result in a different feature being selected to split the dataset at the root, or high up in the tree, and this can have a ripple effect throughout the subtrees under this node. Frequently, when bagging is used with decision trees, the sampling process is extended so that each bootstrap sample only uses a randomly selected subset of the descriptive features in the dataset. This sampling of the feature set is known as **subspace sampling**. Subspace sampling further encourages the diversity of the trees within the ensemble and has the advantage of reducing the training time for each tree.

Figure 4.20[166] illustrates the process of creating a model ensemble using bagging and subspace sampling. The combination of bagging, subspace sampling, and decision trees is known as a **random forest** model. Once the individual models have been induced, the ensemble makes predictions by returning the majority vote or the median depending on the type of prediction required. For continuous target features, the median is preferred to the mean because the mean is more heavily affected by outliers.

28 See Section 3.6.3[98].

29 If we have a very large dataset we may—for computational reasons—want to create bootstrap samples that are smaller than the original dataset. If this is the case, then **sampling without replacement** is preferred. This is called **subagging**.

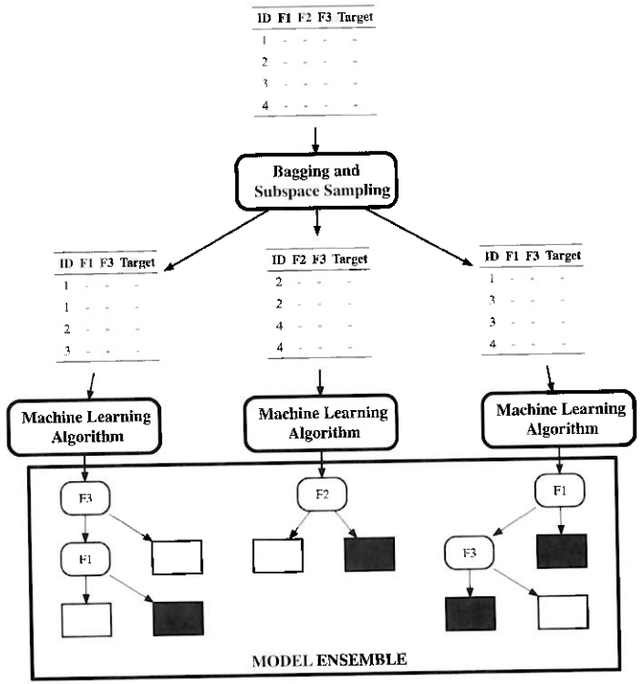

Figure 4.20
The process of creating a model ensemble using bagging and subspace sampling.

4.4.5.3 Summary

Which approach should we use? Bagging is simpler to implement and parallelize than boosting, so it may be better with respect to ease of use and training time. With respect to the general ability of bagging and boosting ensembles to make accurate predictions, the results reported in Caruana et al. (2008) indicate that boosted decision tree ensembles were the best performing model of those tested for datasets containing up to 4,000 descriptive features. For datasets containing more that 4,000 features, random forest ensembles (based on bagging) performed better. Caruana et al. (2008) suggest that a potential explanation for this pattern of results is that boosted ensembles are prone to overfitting,

and in domains with large numbers of features, overfitting becomes a serious problem.[30]

4.5 Summary

We have introduced information theory as a method of determining the shortest sequence of descriptive feature tests required to make a prediction. We have also introduced **decision tree** models, which make predictions based on sequences of tests on the descriptive feature values of a query. Consequently, decision trees naturally lend themselves to being trained using information-based metrics. We also introduced the **ID3** algorithm as a standard algorithm for inducing decision trees from a dataset. The ID3 algorithm uses a top-down, recursive, depth-first partitioning of the dataset to build a tree model beginning at the root node and finishing at the leaf nodes. Although this algorithm works quite well as presented, it assumes categorical features with no missing values and clean data. The algorithm can, however, be extended to handle continuous descriptive features and continuous target features. We also discussed how **tree pruning** can be used to help with the problem of overfitting.

The **C4.5** algorithm is a well-known variant of the ID3 algorithm that uses these extensions to handle continuous and categorical descriptive features and missing features. It also uses post-pruning to help with overfitting. **J48** is an open source implementation of the C4.5 algorithm that is used in many data analytics toolkits. Another well-known variant of the ID3 algorithm is the **CART** algorithm. The CART algorithm uses the **Gini index** (introduced in Section 4.4.1[144]) instead of information gain to select features to add to the tree. This algorithm can also handle continuous target features. The variant of the decision tree algorithm that should be used for a particular problem depends on the nature of the problem and the dataset being used. Performing evaluation experiments using different model types is really the only way to determine which variant will work best for a specific problem.

The main advantage of decision tree models is that they are interpretable. It is relatively easy to understand the sequences of tests a decision tree carried out in order to make a prediction. This interpretability is very important in some domains. For example, if a prediction model is being used as a diagnostic tool in a medical scenario, it is not sufficient for the system to simply

30 Question 5, at the end of this chapter, explores model ensembles in more detail, and worked examples are provided in the solution.

return a diagnosis. In these contexts both the doctor and the patient would want the system to provide some explanation of how it arrives at the predictions it makes. Decision tree models are ideal for these scenarios.

Decision tree models can be used for datasets that contain both categorical and continuous descriptive features. A real advantage of the decision tree approach is that it has the ability to model the interactions between descriptive features. This arises from the fact that the tests carried out at each node in the tree are performed in the context of the results of the tests on the other descriptive features that were tested at the preceding nodes on the path from the root. Consequently, if there is an **interaction effect** between two or more descriptive features, a decision tree can model this. It is worth noting that this ability is diminished if **pre-pruning** is employed, as pre-pruning may stop subtrees that capture descriptive feature interactions from forming. Finally, as noted earlier, decision tree induction is, relatively, robust to noise in the dataset if **pruning** is used.

There are, however, some situations where decision tree models are not the best option. Although decision trees can handle both categorical and continuous features, they tend to become quite large when dealing with continuous descriptive features. This can result in trees becoming difficult to interpret. Consequently, if dealing with purely continuous data, other prediction models may be more appropriate, for example, the error-based models we will see in Chapter 7[323].

Decision trees also have difficulty with domains that have a large number of descriptive features, particularly if the number of instances in the training dataset is small. In these situations overfitting becomes very likely. The probability-based models we will see in Chapter 6[247] do a better job of handling high-dimensional data.

Another potential issue with decision trees is that they are eager learners. As such, they are not suitable for modeling concepts that change over time, because they will need to be retrained. In these scenarios, the similarity-based prediction models that are the topic of the next chapter, Chapter 5[179], perform better, as these models can be incrementally retrained.

We concluded this chapter by explaining **model ensembles**. We can build a model ensemble using any type of prediction model—or, indeed, a mixture of model types. We don't have to use decision trees. However, decision trees are often used in model ensembles, due to the sensitivity of tree induction to changes in the dataset, and this is why we have introduced model ensembles in this chapter. Model ensembles are amongst the most powerful machine learning algorithms: Caruana and Niculescu-Mizil (2006) report a large-scale comparison between seven different types of prediction model in which bagged and boosted tree ensembles are reported as among the best performing. The cost of this high performance, however, is increased learning and model complexity.

4.6 Further Reading

Gleick (2011) provides an excellent and accessible introduction to information theory and its history. Shannon and Weaver (1949) is taken as the foundational book in information theory, and Cover and Thomas (1991) is a well-regarded textbook on the topic. MacKay (2003) is an excellent textbook on information theory and machine learning.

Quinlan (1986) originally described the ID3 algorithm, and Quinlan (1993) and Breiman (1993) are two of the best-known books on decision trees. Loh (2011) provides a good overview of more recent developments in tree induction algorithms.

Schapire (1990) is an example of some of the early work on weak learners and computational learning theory. Freund and Schapire (1995) introduced the **AdaBoost** algorithm, which is one of the seminal boosting algorithms. Friedman et al. (2000) generalized the AdaBoost algorithm and developed another popular boosting algorithm, the **LogitBoost** algorithm. Breiman (1996) developed the use of bagging for prediction, and Breiman (2001) introduced **random forests**. Kuncheva (2004) and Zhou (2012) both provide good overviews of ensemble learning.