Chapter 5
Random Forests

5.1 Introduction and Overview

Just as in bagging, imagine growing a large number of classification or regression trees with bootstrap samples from training data. But now, as each tree is grown, take a random sample of predictors before each node is split. For example, if there are 20 predictors, choose a random five as candidates for defining the split. Then construct the best split, as usual, but selecting only from the five chosen. Repeat this process for each prospective split. Do not prune. Thus, each tree is produced from a random sample of cases, and at each split a random sample of predictors. Compute the mean or proportion for each tree's terminal nodes just as in bagging. Finally, for each case, average over trees as in bagging, but only when that case is out-of-bag. Breiman calls such as procedure a "random forest" (Breiman 2001a).

The random forest algorithm is very much like the bagging algorithm. Again let N be the number of observations in the training data and assume for now that the response variable is binary.

1. Take a random sample of size N with replacement from the data.
2. Take a random sample *without* replacement of the predictors.
3. Construct the first recursive partition of the data as usual.
4. Repeat Step 2 for each subsequent split until the tree is as large as desired. Often this leads to one observation in each terminal node. Do not prune. Compute each terminal node proportion as usual.
5. Drop the *out-of-bag* (OOB) data down the tree. Store the class assigned to each observation along with each observation's predictor values.
6. Repeat Steps 1–5 a large number of times (e.g., 500).

The original version of this chapter was revised: See the "Chapter Note" section at the end of this chapter for details. The erratum to this chapter is available at https://doi.org/10.1007/978-3-319-44048-4_10.

R.A. Berk, *Statistical Learning from a Regression Perspective*,
Springer Texts in Statistics, DOI 10.1007/978-3-319-44048-4_5

7. Using only the class assigned to each observation when that observation is OOB, count the number of times over trees that the observation is classified in one category and the number of times over trees it is classified in the other category.
8. Assign each case to a category by a majority vote over the set of trees when that case is OOB. Thus, if 51 % of the time over a large number of trees a given case is classified as a 1, that becomes its assigned classification.

The major differences between the bagging algorithm and the random forests algorithm are the sampling of predictors at each potential split of the training data, and using only the out-of-bag data when fitted values or classes are assigned to each case. Both are in the service of making the output from each tree in the random forest more independent, but there are additional benefits addressed below.

The key output from random forests is the fitted values, which if classes, are displayed in a confusion table. Because the fitted values are for out-of-bag observations, the confusion table effectively is constructed from test data. Still, the black box produced is every bit as opaque as the bagging black box. There are other algorithms that can be used in concert with random forests to provide a peek at what may be going on inside the box.

Finally, even when random forests is being used solely to describe associations in the data, there is more going on than a level I analysis. The use of OOB data to obtain honest fitted values implies level II concerns broadly and concerns about generalization error in particular. Most discussions of random forests consider results from a dataset as estimates so that, for example, fitted proportions from a sample are called probabilities even if it is unclear exactly what is being estimated and where that estimand resides. We will return to these issues toward the end of this chapter but many of the level II perspectives from bagging carry over.

5.1.1 Unpacking How Random Forests Works

Just like for CART and bagging, beneath a simple algorithm are a host of important details and subtleties. To begin, random forests uses CART as a key building block but in this new setting, CART can be made far more effective. Large trees can produce fitted values less biased with respect to the true response surface. With large trees necessarily comes a more complex $\hat{f}(\mathbf{X})$. Ordinarily, however, an increase in the number of terminal nodes leads to a smaller number of observations in each. The fitted values are more vulnerable to instability. From a level II perspective, the bias–variance tradeoff remains a serious problem. But by averaging over trees, the fitted values case-by-case are made more stable. Ideally, both the bias and the variance can be reduced.[1]

[1]One might think weighting trees by some measure of generalization error would help. Better performing trees would be given more weight in the averaging. So far at least, the gains are at best small (Winham et al. 2013). Because better performing trees tend to have more variation over terminal node fitted values, a form of self-weighting is in play.

Another way to make CART more effective is to sample predictors. One benefit is that the fitted values across trees are more independent. Consequently, the gains from averaging over a large number of trees can be more dramatic. Another benefit is that because only a few predictors are considered for each potential partitioning, there can be overall more predictors than observations; p can be very large and even larger than N. This is a major asset in the era of big data. In principle, having access to a very large number of predictors can help legitimately to improve the fit and any forecasting that might follow.

A third benefit can be understood by revisiting the rationale used by CART to determine whether a particular split is to be made for a given tree. Different sets of predictors are evaluated for different splits so that a wide variety of mean functions are evaluated, each potentially constructed from rather different basis functions. Recall the CART splitting criterion for binary response variables:

$$\Delta I(s, A) = I(A) - p(A_L)I(A_L) - p(A_R)I(A_R), \tag{5.1}$$

where $I(A)$ is the value of the parent impurity, $p(A_R)$ is the probability of a case falling in the right daughter node, $p(A_L)$ is the probability of a case falling in the left daughter node, $I(A_R)$ is the impurity of the right daughter node, and $I(A_L)$ is the impurity of the left daughter node. The CART algorithm tries to find the predictor and the split for which $\Delta I(s, A)$ is as large as possible.[2]

The usefulness of a potential split is a function of the two new impurities and the probability of cases falling into either of the prospective daughter nodes. Suppose there is a predictor that could produce splits in which one of the daughter nodes is very homogeneous but has relatively few observations, whereas the other node is quite heterogeneous but has relatively many observations. Suppose there is another predictor that could generate two nodes of about the same size, each of which is only moderately homogeneous. If these two predictors were competing against each other, the second predictor might well be chosen, and the small, relatively homogeneous, region that the first predictor would exploit be ignored. However, if the second predictor were not in the pool of competitors, the first might be selected instead.

Similar issues arise with predictors that are substantially correlated. There may be little difference empirically between the two so that when they compete to be a splitting variable, one might be chosen almost as readily as the other. But they would not partition the data in exactly the same way. The two partitions that could be defined would largely overlap with each partition having unique content as well. The unique content defined by the predictor not chosen would be excluded.

[2]Geurts and his colleagues (2006) have proposed another method for selecting predictors that can decrease dependence across trees and further open up the predictor competition. They do not build each tree from a bootstrap sample of the data. Rather, for each random sample of predictors, they select splits for each predictor at random (with equal probability), subject to some minimum number of observations in the smaller of the two partitions. Then, as in random forests, the predictor that reduces heterogeneity the most is chosen to define the two subsets of observations. They claim that this approach will reduce the overall heterogeneity at least as much as other ensemble procedures without a substantial increase in bias. However, this conclusion would seem to depend on how good the predictors really are.

Moreover, with the shared area now removed from consideration, the chances that the neglected predictor would be selected later for that tree are significantly reduced because its relationship with the response variable has been eroded. But all is not lost. There will be other trees in the forest and other chances to compete. For one or more subsequent trees, the slighted variable will be competing against others, but not the one with which it is strongly correlated.

In practice, the opportunity for weak predictors to contribute is huge. One by one, they may not help much but in the aggregate, their impact can be substantial. Conventional regression models typically exclude such variables by design. The associations with the response one by one are too small to be interesting in subject matter terms. In practice, weak predictors are treated as noise and swept into the disturbance term. But a large number of small associations, when considered as a group, can lead to much better fitted values and much more accurate imputations and forecasts.

5.2 An Initial Random Forests Illustration

Random forests has its roots in CART and bagging. One might expect, therefore, that when random forests is used for classification, a confusion table will be a key output. But for random forests, the confusion table is constructed from the OOB observations so that out-of-sample performance is represented. Such confusion tables can be called "honest."

We revisit the domestic violence example described earlier with an analysis that shows some of the complexities of working with very challenging data. The data are so challenging that some readers may be underwhelmed with how well random forests performs.[3] More definitive performance is illustrated later. The primary goal for the moment is to raise important application issues. The secondary goal is to undercut some of the hype that many associate with the procedures covered in this and the next three chapters. The procedures are very good to be sure. But, compelling results are never guaranteed.

There are a little over 500 observations, and even if just double interactions are considered, there are well over 100 predictors. This time, the goal is not to forecast new calls for service to the police department that likely involve domestic violence, but only those calls in which there is evidence that felony domestic violence has actually occurred. Such incidents represent about 6 % of the cases. They are very small as a fraction of all domestic violence calls for service. As such, they would normally be extremely difficult to forecast with better skill than could be obtained using the marginal distribution of the response alone. One would make only six mistakes in 100 households if one classified all households as not having new incidents of serious domestic violence.

[3] All of the other procedures tried performed even more poorly.

Table 5.1 Confusion table for a serious domestic violence incidents using a 10 to 1 target cost ratio (N = 516)

	No serious DV forecasted	Serious DV forecasted	Model error
No serious DV	341	146	.30
Serious DV	15	14	.52
Use error	.04	.91	Total error = .31

Using the response variable as the only source of information would in this case mean never correctly identifying any serious domestic violence households. The policy recommendation might be for the police to assume that the domestic violence incident to which they had been called would be the last serious one for that household. This would almost certainly be an unsatisfactory result, which implies that there are significant costs from false negatives. The default cost ratio of 1 to 1 is not responsive to the policy setting.

With a target cost ratio of 10 to 1 for false negatives to false positives favored by the police department, one obtains the results in Table 5.1.[4] The empirical cost ratio of false positives to false negatives is 146/15. The cost ratio value of 9.7 to 1 means that each false negative is worth nearly 10 times more than each false positive. This is effectively the 10 to 1 cost ratio sought and in practice, it is very difficult to hit the target cost ratio exactly. In practice, moreover, the differences between a confusion table with a 10 to 1 cost ratio and a confusion table with a 9.7 to 1 cost ratio will usually not matter.

Table 5.1 shows that random forests incorrectly classifies households 31 times out of 100 overall. The value of 31 can be interpreted as the overall average cost of a classification error. But noted earlier, the usual overall error gives all errors equal weight. Having decided from policy considerations that the proper cost ratio is 10 to 1, the proper average cost .57 (i.e., $[(10 \times 15) + 146]/516$). But, the overall measure of cost-weighted generalization error neglects some very important features of the table more relevant to real decision-making.

From the model error, one can see that about 30 % of the of the cases with no subsequent DV are classified incorrectly and about 52 % of the cases with subsequent DV are classified incorrectly. That this application of random forests correctly classifies only about half the DV cases may be disappointing, but without using any of the predictors, no DV cases whatsoever would be correctly classified.[5]

If the results from Table 5.1 are to be used to inform real decisions, use error is especially instructive. When a forecast is for no subsequent DV, the assigned class is incorrect only about 4 % of the time. When a forecast is for subsequent DV, the assigned class is incorrect about 91 % of the time. The large difference in forecasting skill results substantially from the 10 to 1 cost ratio. Implicit is a policy preference

[4] R code is not provided because it is too early in the exposition. Lots of R code is provided later.

[5] It might seem strange that the classification accuracy of one outcome is not 1 minus the classification accuracy of the other. If a case is not classified as DV, it must be classified as a DV case. But one is conditioning on the actual outcome, and the denominators of the model errors differ. Accuracy depends in part on the base.

to accept a relatively large number of false positives (i.e., 146) so that the number of false negatives is relatively low (i.e., 15). The 146 false positives lead to poor accuracy with the DV class that is predicted.

At the same time, the policy preference means that classification accuracy when no DV is the assigned class is improved compared to the baseline. Recall, that if the marginal distribution of the response is used, all 515 cases are predicted to be arrest-free, and 6 % of the cases are not arrest-free. From Table 5.1, 69 % percent of the cases are predicted to be arrest-free, and 4 % of those cases are not arrest-free. Because the base of 6 % is very small, it is impossible to obtain large improvements in percentage units; even perfect forecasting would only reduce the forecasting errors by 6 percentage points. In such circumstances, it is common to report ratios, and then the forecasting errors is reduced by one-third (2 %/6 %).

In short, the procedure accepts much weaker evidence to assign the serious DV class than to assign the no serious DV class. In this illustration, it takes very strong statistical evidence for a case to be classified as no serious DV. High accuracy follows for forecasts of no serious DV.

Although confusion tables using OOB data are an essential feature of random forests output, there are other kinds of output that can be very helpful. These are derived from additional algorithms that will be discussed shortly. In preparation and to help provide readers with better access to the technical literature, we turn to a few formalities.

5.3 A Few Technical Formalities

With some initial material on random forests behind us, it is useful to take a somewhat more formal look at the procedure. We build on an exposition by Breiman (2001a). The concepts considered make more rigorous some ideas that we have used in the past two chapters, and provide important groundwork for material to come. We also consider whether random forests overfits as the number of trees in the forest increases. As before, we emphasize categorical, and especially binary, response variables.

It will be important to keep straight randomness introduced by the algorithm and randomness introduced by the data. One can undertake a level I analysis despite randomness from algorithm because it has nothing to do with how the data were generated. The moment interest includes generalizing beyond the data on hand, randomness in the data are in play, and a level II analysis must follow. Most formal expositions of random forests emphasize level II issues.

In order to help readers who may wish to read Breiman's treatment of random forests or subsequent work that draws directly from it, Breiman's notation is adopted. Bold type is used for vectors and matrices. Capital letters are used for random variables. The terms "predictor" and "input" are used interchangeably.

5.3.1 What Is a Random Forest?

With categorical response variables, a random forest is an ensemble of classifiers. The classifiers are K classification trees, each based in part on chance mechanisms. Like CART, these classifiers can work with more than two response categories. The goal is to exploit the ensemble of K trees to assign classes to observations using information contained in a set of predictors.

We formally represent a random forest as a collection of K tree-structured classifiers $\{f(\mathbf{x}, \Theta_k), k = 1, \ldots\}$, where $\mathbf{x}$ is an input vector of p input values used to assign a class, and k is an index for a given tree. "Each tree casts a unit vote for the most popular class at input x" (Breiman 2001a: 6). As an ensemble of classifiers, a random forest is also a classifier.

Θ_k is a random vector constructed for the kth tree so that it is independent of past random vectors $\Theta_1, \ldots, \Theta_{k-1}$, and is generated from the same distribution. For bagging, it is the means by which observations are selected at random with replacement from the training data. For random forests, it is also the means by which subsets of predictors are sampled without replacement for each potential split. In both cases, Θ_k is a collection of integers. The integers serve as indices determining which cases and which predictors, respectively, are selected. Integers for both sampling procedures can be denoted by Θ_k.

The paramount output from a random forest is an assigned class for each observation determined at its input values $\mathbf{x}_i$. In CART, for example, the class assigned to an observation is the class associated with the terminal node in which an observation falls. With random forests, the class assigned to each observation is determined by a vote over the set of tree classifiers when OOB data are used. Classes are assigned to observations much as they are in bagging. It is important conceptually to distinguish between the class assigned by the kth tree and the class assigned by a forest. It is also important to appreciate that when used as a classifier, random forests does not produce probabilities for the response variable classes. There is not even an analogy to the fitted values from logistic regression.

5.3.2 Margins and Generalization Error for Classifiers in General

For ease of exposition, consider first any ensemble classifier, not random forests in particular. Suppose there is a training dataset with input values and associated values for a categorical response. As before, each observation in the training dataset is realized at random and independently. The set of inputs and a response are random variables.

There is an ensemble of K classifiers, $f_1(\mathbf{x})$, $f_2(\mathbf{x})$, $\ldots$, $f_K(\mathbf{x})$. For the moment, we do not consider how these different classifiers are constructed. The margin function at *the data point* $\mathbf{X}$, Y is then defined as

$$mg(\mathbf{X}, Y) = av_k I(f_k(\mathbf{X}) = Y) - \max_{j \neq Y} av_k I(f_k(\mathbf{X}) = j), \tag{5.2}$$

where $I(.)$ is an indicator function, j is an incorrect class, av_k denotes averaging over the set of classifiers for a single realized data point, and max denotes the largest value. For a given set of x-values and the associated observed class, the margin is the average number of votes over classifiers for the correct observed class minus the maximum average number of votes over classifiers for any other class. The term "data point" for this discussion is for the same a row in the dataset. Because Eq. 5.2 applies to any row, it applies to all rows.

From the definition of the margin function, generalization error is then,

$$g = P_{\mathbf{X},Y}(mg(\mathbf{X}, Y) < 0), \tag{5.3}$$

where P means probability. In words, Breiman's generalization error is the probability over realizations of a given row of the data that the vote will be won by an incorrect class: the probability that the margin will be negative. Because Eq. 5.3 applies to any row, it applies to realizations for all rows. There are no test data in this formulation of generalization error, and the training data are not fixed. Recall that when Hastie et al. (2009: 220) define generalization error, the training data are fixed and performance is evaluated over realizations of test data. One is, of course, free to define concepts as one wants, but for both definitions, generalization error should be small.

5.3.3 Generalization Error for Random Forests

Now, suppose for the kth classifier $f_k(\mathbf{X}) = f(\mathbf{X}, \Theta_k)$. There are K tree classifiers that comprise a random forest. Breiman proves (2001a) that as the number of trees increases, the estimated generalization error converges to the true generalization error, which is

$$P_{\mathbf{X},y}(P_\Theta(f(\mathbf{X}, \Theta) = Y) - \max_{j \neq Y} P_\Theta(f(\mathbf{X}, \theta) = j) < 0). \tag{5.4}$$

$P_\Theta(f(\mathbf{X}, \Theta) = Y)$ is the probability of a correct classification over trees that differ randomly because of the sampling of the training data with replacement and the sampling of predictors. One can think of this as the proportion of times a classification is correct over a limitless number of trees grown from the same dataset. Parallel reasoning for an incorrect classification applies to $P_\Theta(f(\mathbf{X}, \theta) = j)$. Note that the data are fixed.[6] Then, we are essentially back to Eq. 5.3. As before, $P_{\mathbf{X},Y}$ is the probability of an incorrect classification over realizations of the data themselves. We

[6]The concept of training data gets fuzzy at this point. The training data for a given tree is the random sample drawn with replacement from the dataset on hand. But for the random forest that entire dataset is the training data.

address the uncertainty that is a product of random forests and then the uncertainty that is a product of the random variables. Breiman proves that as the number of trees increase without limit, all of these sources of randomness cancel out leaving the true generalization error shown in Eq. 5.4.

What does one mean in this context by "true" generalization error? No claims are made that the classes assigned by a given forest are "correct." In function estimation language, no claims are made that the true $f(\mathbf{X})$ has been found. Rather, one has once again some approximation of the true response surface, and it is the generalization error of that approximation to which an estimated generalization error converges. It is the "true" generalization error of the approximation.[7]

The importance of the convergence is that demonstrably random forests does not overfit as more trees are grown. One might think that with more trees, one would get an increasingly false sense of how well the results generalize. Breiman proves that this is not true. Given all of the concern about overfitting, this is an important result.

There is some work addressing random forests statistical consistency for what appears to be the true response surface. Even if all of the needed predictors are available, there can be situations in which random forests is not consistent (Biau et al. 2008; Biau and Devroye 2010). However, because this work is somewhat stylized, it is not clear what the implications for practice may be. Work that is more recent and somewhat less stylized proves consistency but among other things, requires sparsity (Biau 2012). This means that only a "very few" of the large set of potential predictors are related to the response (Biau 2012: 1067). We are apparently back to the conventional linear regression formulation in which there are two kinds of predictors: those that matter a lot and those that do not matter at all. A look at the variable importance plots reported later in this chapter shows no evidence of sparsity. There may well be other kinds of data for which sparsity can be plausibly defended (e.g., for genomics research).

Another assumption that all of the recent theoretical work seems to share is that the trees in a random forest are "honest" (Wager 2014: 7). By "honest," one means that the data used to determine the fitted value for each terminal node are not the data used to determined the data partitions. By this definition, the trees in Breiman's random forests are *not* honest so the theoretical work does not directly apply. Moreover, "…the bias of CART trees seems to be subtle enough that it does not affect the performance of random forests in most situations" (Wager 2014: 8). It is probably fair to say that the jury is still out on the formal properties of random forests, but that in practice, there is a consensus that it performs well.

[7]Because of random forests' chance components and random forests' dependence on sample size, the approximation response surface in the joint probability distribution is the expected random forest grown with the same number of observations as available in the data.

5.3.4 The Strength of a Random Forest

The margin function for a given realized data point in a random forest (not just any classifier) is defined as

$$mr(\mathbf{X}, Y) = P_{\Theta}(f(\mathbf{X}, \Theta) = Y) - \max_{j \neq Y} P_{\Theta}(f(\mathbf{X}, \Theta) = j), \tag{5.5}$$

where $f(\mathbf{X}, \Theta)$ denotes random forest classifications for a given row that can vary because of the chance mechanisms represented by Θ. Because of the randomness build into the random forest algorithm, the margin function is defined using the probability of a correct vote and an incorrect vote over forests grown from the same dataset. It appropriates a piece of the definition of random forests generalization error.

It is a short step from a random forest margin function to a definition of the "strength" of a random forest. We take the expectation over realizations of the data. That is, the expected value of Eq. 5.5 is:

$$s = E_{\mathbf{X},\mathbf{y}} mr(\mathbf{X}, Y). \tag{5.6}$$

The strength of a random forest is an expected margin over all possible realizations of the data. And no surprise, strong is good.

5.3.5 Dependence

As previously noted, the effectiveness of averaging over trees using votes depends on the independence of the trees. But, how does one think about that independence in this setting? There is apparently some confusion in the literature (to which I plead guilty). Hastie et al. (2009: 598) stress correctly that "… $\rho(x)$ is the theoretical correlation between a pair of random forest trees evaluated at x, induced by repeatedly making training sample draws $\mathbf{Z}$ from the population, and then drawing randomly a pair of random forest trees." That is, it is the expected value of the correlation between the fitted values from randomly selected pairs of trees over realizations of the data. Ideally, that expected correlation is zero.

5.3.6 Implications

Dependence is important because Breiman shows (Breiman 2001a: 6) that the upper bound for the generalization error is

$$g^* = \frac{\bar{\rho}(1 - s^2)}{s^2}, \tag{5.7}$$

where $\bar{\rho}$ is the expected correlation over pairs of trees as just described, and s is the strength of the random forest. Ideally, the former is small and the latter is large.

Equation 5.7 implies that both the sizes of margins and that ways in which the random forest algorithm introduces randomness are critical. The random forest algorithm already does a pretty good job reducing the expected correlation. But in practice, random forest sometimes can be tuned to help. For example, sampling fewer inputs at each splitting opportunity can in some situations improve performance, and the random forest software in R emphasized here (i.e., *randomForest()*) has a default function for determining the number of predictors that seems to work quite well.[8]

5.3.7 Putting It All Together

Why does random forests work so well as a classifier? Although there are not yet any formal proofs, recent work by Wyner and colleagues (2015) provides a conceptual framework coupled with simulations that support some very instructive intuitions. We will see later that their thinking applies to boosting as well.

It has become common practice, and often the default practice, to grow trees as large as the data will allow. Terminal nodes are often perfectly homogeneous. Indeed, sometimes tuning parameters are set so that each terminal node can contain a single observation. Consequently, in the bootstrap sample used to grow each tree, the fit to the data can be perfect; for each observation, the tree assigns the correct class. When this happens, one has an interpolating classifier (Wyner et al. 2015: 9).

Figure 5.1 provides a very simple illustration. We again have a 3-dimensional scatter plot for a realized training dataset. There are two numerical predictors X and Z, and a binary outcome represented as red, blue, or purple. The red and blue outcome classes are noiseless in the sense that no matter what the realization of the data, observations in those locations are always red or blue respectively. In contrast, the purple circles can be either red or blue depending on the realization.

The box in the upper left-hand corner results from a single classification tree terminal node with $X < 2$ and $Z > 12$. The node has one observation and, therefore, is necessarily classified correctly. With sufficiently fine-grained partitioning of X and Z using a sufficiently large tree, each circle in Fig. 5.1 can reside in its own terminal node. Each would be classified correctly, and its terminal nodes would be perfectly homogeneous. One would have what Wyner and his colleagues call an interpolation of the data.

Things are more complicated in the lower right-hand box. For observations with $X > 7$ and $Z < -4$, the outcome class can vary over realizations. It could be a blue outcome for one data realization and a red outcome for another data realization. Suppose in one realization of the training data, it is red and is classified as red. One still has an interpolation of the realized data. But for test data that classification could

[8]There are other implementations for random forests in R that are briefly discussed later.

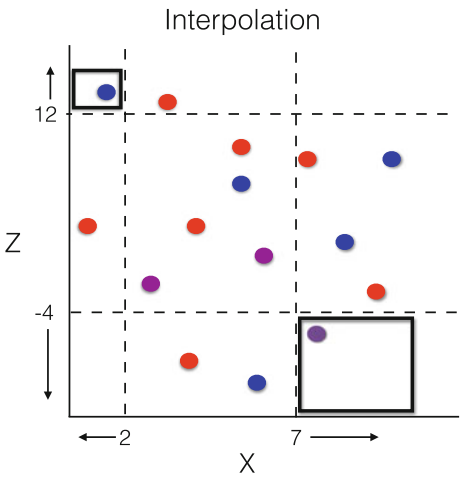


Fig. 5.1 Visualization of interpolation for a single classification tree with *blue* and *red* outcomes having no noise and *purple* outcomes having some noise

be wrong because the realized value in that location is blue. Generalization error has been introduced because of overfitting.

However, *because of the interpolation*, one has "local robustness" (Wyner et al. 2015: 9–10). Whatever the error in the fit caused by the purple circle, it is confined to its own terminal node. That is, all of the classifications for the other circles are made separately from the noisy circle, so that the damage is very limited. In an important sense, overfitting has some demonstrable benefits.

But there is still overfitting for a given tree. Enter random forests. Each tree is grown with a random sample of the training data (with replacement). Whatever the expected value is for the outcome, sometimes the lower right-hand box will contain a red outcome and sometimes it will contain a blue outcome. By the Bayes decision rule, a vote taken over trees for terminal nodes defined just like the lower right-hand box will classify observations that land in that region as red or blue, depending on the majority vote. This is the best one can do with respect to generalization error.

Voting is an averaging process that improves on the generalization error from a single tree. Although each tree will overfit, averaging over trees compensates. Because for real data most points will be purple, the averaging is essential as a form of regularization.

There is even more going on. First, the striving to interpolate is helped by the sampling of predictors. Imagine that in Fig. 5.1 one location has a red circle right on top of a blue circle. There is no way with X and Z alone to arrive at an interpolation point at that location.[9] One solution is to come upon a variable, W, that in combination with X and Z could define two terminal nodes, one with the blue circle and one with the red circle. But, looking back to Eq. 5.1, suppose that for a given prospective partitioning of the data, W does not make a sufficient contribution to homogeneity to be selected as the next partitioning variable. Even if it is strongly related to the

[9] As discussed in Chap. 1, there might be a solution is some linear basis expansion of X and Z.

response, it is also too strongly related to X. An opportunity to distinguish between a red circle and a blue circle is lost. This problem can be solved by sampling predictors. Suppose, there is a predictor U that, like X, is strongly related to the response, but unlike X is only moderately related to Z. If for some split, X is not available as a potential partitioning variable but U is, U, W, and Z can participate sequentially in the partitioning process.[10]

Second, the random sampling helps in the situation just described. Over trees, the covariances among the variables will vary. Consequently, there will be variation in how strongly X and W are related to the response and to each other. This provides an opportunity in some samples for W to be less competitive with X. For example, in some samples, there can be prospective partitions for which W reduces heterogeneity sufficiently, even if X is an earlier partitioning variable. The result will be some different terminal nodes that, in turn, will affect the vote over trees.

Finally, a larger number of training observations can really help. With larger samples, one can grow larger trees. With larger trees, interpolation can better approximated. With a better approximation of interpolation, coupled with averaging over trees, generalization error can be reduced.

Although all of the mechanisms just described are always in play, challenges from real data are substantial. In practice, all of the class labels are noisy so that the circles in Fig. 5.1 would all be purple. A perfect fit in the training data will not lead to a perfect fit in the test data. Moreover, a perfect fit is certainly no guarantee of obtaining unbiased estimates of the true response surface. There will typically be omitted predictors, and the classification trees are still limited by the ways splits are determined and the greedy nature of the tree algorithm. It remains true, however, that "Random forests has gained tremendous popularity due to robust performance across a wide range of data sets. The algorithm is often capable of achieving best-in-class performance with respect to generalization error and is not highly sensitive to choice of tuning parameters, making it an ideal off-the-shelf tool of choice for many applications" (Wyner et al. 2015: 9).

5.4 Random Forests and Adaptive Nearest Neighbor Methods

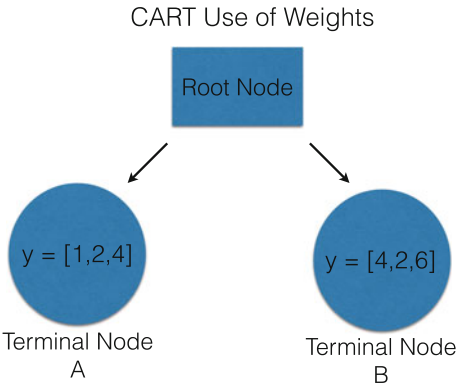
A conceptual link was made earlier between CART and adaptive nearest neighbor methods. Not surprisingly, similar links can be made between random forests and adaptive nearest neighbor methods. But for random forests, there are a number of more subtle issues (Meinshausen 2006; Lin and Jeon 2006). These are important not just for a deeper understanding of random forests, but for recent theoretical treatments (Biau 2013; Wager 2014; Wager et al. 2014; Wager and Walther 2015).

[10] Also, W might be selected farther down in the tree even if it is chosen along with X, depending on the other predictors chosen. They are both in a competition with several other candidate splitting variables.

Recall that in CART, each terminal node represented a region of nearest neighbors. The boundaries of the neighborhood were constructed adaptively when the best predictors and their best splits were determined. With the neighborhood defined, all of the observations inside were used to compute a mean or proportion. This value became the measure of central tendency for the response within that neighborhood. In short, each terminal node and the neighborhood represented had its own conditional mean or conditional proportion.

Consider the case in which equal costs are assumed. This makes for a much easier exposition, and no key points are lost. The calculations that take place within each terminal node implicitly rely on a weight given to each value of the response variable. For a given terminal node, all observations not in that node play no role when the mean or proportion is computed. Consequently, each such observation has a weight of zero. For a given terminal node, all of its observations are used when the mean or proportion is computed. Consequently, each value of the response variable in that node has a weight equal to $1/n_\tau$, where n is the number of observations in terminal node τ. Once the mean or proportion for a terminal node is computed, that mean or proportion can serve as a fitted value for all cases that fall in that terminal node.

Figure 5.2 shows a toy rendering. The tree has a single partitioning of the data. There happen to be three values of the response variable in each terminal node. Consider terminal node A. The mean for terminal node A is 2.33, computed with weights of 1/3 for the values in that node and weights of 0 otherwise; the values of the response variable in terminal node B play no role when the mean of node A is computed. Each of the three observations landing in terminal node A are assigned a value of 2.33 as their fitted value. If the response variable had been binary, the numbers in the two terminal nodes would have been replaced by 1s and 0s. Then a conditional proportion for terminal node A would be the outcome of the weighted



$$\bar{y}_A = (1/3 \times 1) + (1/3 \times 2) + (1/3 \times 4) + (0 \times 4) + (0 \times 2) + (0 \times 6) = 2.33$$

$$\bar{y}_B = (0 \times 1) + (0 \times 2) + (0 \times 4) + (1/3 \times 4) + (1/3 \times 2) + (1/3 \times 6) = 4.00$$

Fig. 5.2 CART weighting used to assign a mean or proportion to a terminal node A or B

averaging. And from this, an assigned class could be determined as usual. The same reasoning applies to terminal node B.

A bit more formally, a conditional mean or proportion for any terminal node τ is

$$\bar{y}_\tau | x = \sum_{i=1}^{N} w_{(i,\tau)} y_i, \tag{5.8}$$

where the sum is taken over the *entire* training dataset, and w_i is the weight for each y_i. The sum of the weights over all observations is 1.0. In practice, most of the weights for the calculations in any terminal will be zero because they are not associated with the terminal node τ. This is no different from the manner in which nearest neighbor methods can work when summary measures of a response variable are computed.

There are two important features of the weighting terminal node by terminal node. First, each terminal node defines a neighborhood. The x-values for each observation determine in which neighborhood the observation belongs. It will often turn out that observations with somewhat different sets of x-values land in the same neighborhood. For example, a partition may be defined by a threshold of 25 years of age. All ages less than 25 sent to one neighborhood and all ages 25 and above are sent to another.

Second, for any given tree, each of the N observations will have a single, nonzero weight because each observation must land in one (and only one) of the terminal nodes. It is in that node that the single weight is determined as the reciprocal of the number of observations. In our toy example, each of the six observations happen to have a weight of 1/3 because both terminal nodes have three observations.

Now imagine that the tree is grown as an element of a random forest. The form of the calculations shown for terminal nodes A and B still apply with the fitted values, in this instance, a conditional mean. However, there are now a large number of such trees. For each observation, random forests average the weights obtained from each tree (Lin and Jeon 2006: 579–580). Consequently, the ith fitted value from a random forest is a weighted average of N values of the response, much like in Fig. 5.2, but using average weights.[11] That is,

$$\hat{y}_i = \sum_{i=1}^{N} \bar{w}_i y_i, \tag{5.9}$$

where $\bar{w}_i$ is the average weight.

The weights can serve another important purpose. Suppose for a given neighborhood defined by $\mathbf{x}_0$, there are 10 observations and, therefore, 10 values for a quantitative response. There are also 10 average weights as just described. If one orders the response values from low to high, the weights conceptualized as

[11] For example, if there is a tiny random forest of 3 trees (more like very small stand), and the ith observation has 3 weights of .2, .3, and .1, the average weight over the 3 trees is .2.

Table 5.2 Weights and cumulative weights for a target value $\mathbf{x}_0$

Average weight	Response value	Cumulative weight
.10	66	.10
.11	71	.21
.12	74	.33
.08	78	.41
.09	82	.50
.10	85	.60
.13	87	.73
.07	90	.80
.11	98	.91
.09	99	1.0

probabilities can be used to compute other summary measures than the mean. Table 5.2 can be used to illustrate.

From left to right, there are ten average weights that sum to 1.0, ten response values available for $\mathbf{x}_0$, listed in order, and then cumulative weights. The mean is computed by multiplying each response value by its average weight and adding the products. In this case, the mean is 83. Quantiles are also available. The 10th percentile is 66. The 50th percentile (the median) is 82. The 90th percentile is a little less than 98. In short, one is not limited to the mean of each $\mathbf{x}_0$.

Suppose one has a random forest and a variety of predictor profiles of x-values. When the response is quantitative, there routinely is interest in determining the fitted conditional mean for each profile. But sometimes, there will be interest in fitted conditional medians to "robustify" random forest results or to consider a central tendency measure unaffected by the tails of the distribution. Sometimes, there is subject matter interest in learning about a conditional quantile such as the 25th percentile or the 90th percentile.

For example, in today's world of school accountability based on standardized tests, perhaps students who score especially poorly on standardized tests respond better to smaller classroom sizes than students who excel on standardized tests. The performance distribution on standardized tests, conditioning on classroom size, differs for good versus poor performers. Building on work of Lin and Jeon just discussed, Meinshausen (2006) alters the random forests algorithm so that conditional quantiles can be provided as fitted values. An application is provided later in this chapter using *quantregForest()* in R.[12]

But there are caveats. In particular, each tree is still grown with a conventional impurity measure, which for a quantitative response is the error sum of squares (Meinshausen 2006: Sect. 3). If one is worried about the impact of a highly skewed response variable distribution, there may well be good reason to worry about the splitting criterion too. For example, one might prefer to minimize the sum of the

[12]The package *quantregForest()* is authored by Nicolai Meinshausen and Lukas Schiesser.

absolute values of the residuals (i.e., L_1 loss) rather than the sum of squared residuals (i.e., L_2 loss) as an impurity measure. This was originally proposed by Breiman and his colleagues in 1984 (Chap. 8), and there have been interesting efforts to build on their ideas (Chaudhuri and Loh 2002; Loh 2014). But L_1 loss does not seem to have yet been incorporated into random forests. We will see later that L_1 loss has been implemented in stochastic gradient boosting.

5.5 Introducing Misclassification Costs

Just as in CART, there is a need when random forests is used as a classifier to consider the relative costs of false negatives and false positives. Otherwise, for each tree, one again has to live with results that depend on the default of equal costs and a prior distribution for the response variable that is the same as its marginal distribution in the data.

Perhaps the most conceptually direct method would be to allow for a cost matrix just as CART does. To date, this option is not available in random forest software, and there is evidence that it might not work effectively if it were.

There are four approaches that have been seriously considered for the binary class case. They differ by whether costs are imposed on the data before each tree is grown, as each tree is grown, or at the end when classes are assigned. Although binary outcomes will be emphasized, the lessons for response variables with more than two categories will be covered as well.

1. Just as in CART, one can use a prior distribution to capture costs as each tree is grown. This has the clear advantages of being based on the mechanics of CART and capitalizing on a straightforward way to translate costs into an appropriate prior.
2. After all of the trees are grown, one can differentially weight the classification votes over trees. For example, one vote for classification in the less common category might count the same as two votes for classification in the more common category. This has the advantage of being easily understood.
3. After all of the trees are grown, one can abandon the majority vote rule and use thresholds that reflect the relative costs of false negatives and false positives. For instance, rather than classifying as 1 all observations when the vote is larger than 50 %, one might classify all observations as 1 when the vote is larger than 33 %. This too is easy to understand.
3. When each bootstrap sample is drawn before a tree is grown, one can oversample cases from one class relative to cases from the other class, in much the same spirit as disproportional stratified sampling used for data collection (Thompson 2002: Chap. 11). Before a tree is grown, one oversamples the cases for which forecasting errors are relatively more costly. Conceptually, this is a lot like altering the prior distribution.

All four approaches share the problem that the actual ratio of false negatives to false positives in the confusion table probably will not sufficiently correspond to the target cost ratio. In practice, this means that whatever method is used to introduce relative costs, that method is simply considered a way to "tune" the results. With some trial and error, an appropriate ratio of false negatives to false positives can usually be achieved. All four also share the problem mentioned earlier that when there are more than two response categories, none of the methods introduce enough new information to directly control all of the relevant cost ratios. But also as before, with some trial and error it is usually possible to approximate well enough the target cost in the confusion table.

Although experience suggests that in general all four methods can tune the results as needed, there may be some preference for tuning by the prior or by stratified bootstrap sampling. Both of these methods will affect the confusion table through the trees themselves. The structure of the trees themselves responds to the costs introduced. Changing the way votes are counted or the thresholds used only affects the classes assigned, and leaves the trees unchanged. The defaults of equal costs and the empirical prior remain in effect. By allowing the trees to respond directly to cost considerations, more responsive forecasts should be produced. Moreover, any output beyond a confusion table will reflect properly the desired costs. More is said about such output shortly.

There is one very important situation in which the stratified sampling approach is likely to be demonstrably superior to the other three approaches. If the response variable is highly unbalanced (e.g., a 95–5 split), any given bootstrap sample may fail to include enough observations for the rare category. Then, a useful tree will be difficult to grow. As observed earlier, it will often be difficult under these circumstances for CART to move beyond the marginal distribution of the response. Oversampling rare cases when the bootstrap sample is drawn will generally eliminate this problem. Using a prior that makes the rare observations less rare can also help, but that help applies in general and will not be sufficient if a given bootstrap sample makes the rare cases even more rare.

We consider some applications in depth shortly. But a very brief illustration is provided now to prime the pump.

5.5.1 A Brief Illustration Using Asymmetric Costs

Table 5.3 was constructed using data from the prison misconduct study described earlier. In this example, the response is incidents of very serious misconduct, not the garden variety kind. As noted previously, such misconduct is relatively rare. Less than about 3 % of the inmates had such reported incidents. So, just as for the domestic violence data shown in Table 5.1, it is extremely difficult to do better than the marginal distribution under the usual CART defaults. In addition, there is simply not a lot of misconduct cases from which the algorithm can learn. Trees in the forest will be unable to partition the rare cases as often as might be desirable; tree depth

Table 5.3 Confusion table for forecasts of serious prison misconduct with a 20 to 1 target cost ratio (N = 4806)

	Forecast no misconduct	Forecast misconduct	Model error
No misconduct	3311	1357	.29
Misconduct	58	80	.42
Use error	.02	.94	Overall error = .29

may be insufficient. In short, when a response distribution is highly unbalanced, very large samples can sometimes help too.

Suppose that the costs of forecasting errors for the rare cases were substantially higher than the costs of forecasting errors for the common cases. These relative costs can be introduced effectively by drawing a stratified bootstrap sample, oversampling the rare cases. And by making the rare cases less rare, problems that might follow from the highly unbalanced response variable can sometimes be overcome.

For Table 5.3, the bootstrap samples for each of the two response categories was set to equal 100.[13] The "50–50" bootstrap distribution was selected by trial and error to produce an empirical cost ratio of false negatives to false positives of about 20 to 1 (actually 23 to 1 here). The cost ratio may be too high for real policy purposes, but it is still within the range considered reasonable by prison officials.

Why 100 cases each? Experience suggests that the sample size for the less common response category should equal about two-thirds of the number of cases in that class. If a larger fraction of the less common cases is sampled, the out-of-bag sample size for that class may be too small. The OOB observations may not be able to provide the quality of test data needed.

With the number of bootstrap observations for the less common category determined to be 100, the 50–50 constraint leads to 100 cases being sampled for the more common response category. In practice, one determines the sample size for the less common outcome and then adjusts the sample size of the more common outcome as needed.

Table 5.3 can be interpreted just as any of the earlier confusion tables. For example, the overall proportion of cases incorrectly identified is 0.29, but that fails to take the target costs of false negatives to false positives (i.e., 20 to 1) into account. Random forests classify 42 % of the incidents of misconduct incorrectly and 29 % of the no misconduct cases incorrectly. Should prison officials use these results for forecasting,

[13] Using the procedure *randomForest()* in R written by Leo Breiman and Ann Culter, and later ported to R by Andy Liaw and Matthew Wiener, the stratified sampling argument was *sampsize=c(100,100)*. The order of the two sample sizes depends on the order of the response variable categories. They are ordered alphabetical or numerically low to high depending on how the variable is coded. For classification procedures in R, it is a good idea to always construct the outcome variable as a factor. The procedure *randomForest()* will automatically know that the task is classification. If a binary response variable is defined as numeric with a value of 0 and a value of 1, and if the type of procedure within *randomForest()* is not identified as classification, *randomForest()* will proceed with regression. This is a common error.

a forecast of no serious misconduct would be wrong only 2 times out of 100, and a forecast of serious misconduct would be wrong 94 times out of 100. The very large number of false positives results substantially from the target 20 to 1 cost ratio. But, for very serious inmate misconduct, having about 1 true positive for about 17 false positives (1357/80) may be an acceptable trade-off. The misconduct represented can include homicide, assault, sexual assault, and narcotics trafficking. If not, the cost ratio could be made more symmetric.

To summarize, random forests provides several ways to take the costs of false negatives and false positives into account. Introducing stratified bootstrap sampling seems to work well in practice. Ignoring the relative costs of classification errors does not mean that costs are not affecting the results. The default is equal costs and using the marginal distribution of the response variable as the empirical prior.

5.6 Determining the Importance of the Predictors

Just as for bagging, random forests leaves behind so many trees that collectively they are useless for interpretation. Yet, a goal of statistical learning can be to explore how inputs are related to outputs. Exactly how best to do this is currently unresolved, but there are several useful options available. We begin with a discussion of "variable importance."

5.6.1 Contributions to the Fit

One approach to predictor importance is to record the decrease in the fitting measure (e.g., Gini index, mean square error) each time a given variable is used to define a split. The sum of these reductions for a given tree is a measure of importance for that variable when that tree is grown. For random forests, one can average this measure of importance over the set of trees.

As with conventional variance partitions, however, reductions in the fitting criterion ignore the prediction skill of a model, which many statisticians treat as the gold standard. Fit measures are computed with the data used to build the classifier (i.e., in-sample). They are not computed from test data (i.e., out-of-sample).

Moreover, it can be difficult to translate contributions to a fit statistic into practical terms. Simply asserting that a percentage contribution to a fit statistic is a measure of importance is circular. Importance must be defined outside of the procedure used to measure it. And what is it about contributions to a measure of fit that makes a predictor more or less important? Even if an external definition is provided, is a predictor important if it can account for, say, 10 % of the reduction in impurity?

One also must be fully clear that contributions to the fit by themselves are silent on what would happen if in the real world a predictor is manipulated. Causality can only be established by how the data were generated, and causal interpretations depend on there being a real intervention altering one or more predictors (Berk 2003).

5.6.2 Contributions to Prediction

Breiman (2001a) has suggested another form of randomization to assess the role of each predictor. This method is implemented in *randomForest()*. It is based on the reduction in what Breiman calls prediction accuracy when a predictor is shuffled so that the predictor cannot make a systematic contribution to a prediction. For categorical response variables, it is the reduction in *classification* accuracy with OOB data. One conditions on the actual outcome to determine the proportion of times the wrong class is assigned. As such it has a very grounded interpretation that can be directly linked to the rows of a confusion table. For numeric response variables, the standard approach is to use the increase in mean squared error for the OOB data. One is again conditioning on the actual value of Y. The term "prediction", can be a little misleading, but to be consistent with Breiman, we will stick with it here.

Breiman's approach has much in common with the concept of Granger causality (Granger and Newbold 1986: Sect. 7.3). Imagine two times series, Y_t and X_t. If the future conditional distribution of Y given current and past values of Y is the same as the future conditional distribution of Y given current and past values of Y and X, X does not Granger-cause Y.[14] If the two future conditional distributions differ, X is a Granger-cause of Y.

These ideas generalize so that for the baseline conditional distribution, one can condition not just on current and past values of Y but on current and past values of other predictors (but not X). Then X Granger-causes Y, conditional on the other predictors, if including X as a predictor changes the future conditional distribution of Y. In short, the idea of using forecasting performance as a way to characterize the performance of predictors has been advanced in both the statistical and econometrics literature.

Breiman's importance measure of prediction accuracy differs perhaps most significantly from Granger cause in that Breiman does not require time series data and randomly shuffles the values of predictors rather than dropping (or adding) predictors from a procedure. The latter has some important implications discussed shortly.

For Breiman's approach with a categorical response variable, the following algorithm is used to compute each predictor's importance.

1. Construct a measure prediction error ν for each tree as usual by dropping the out-of-bag (OOB) data down the tree. Note that this is out-of-sample because data not used to grow the tree are used to evaluate its predictive skill.
2. If there are p predictors, repeat Step 1 p times, but each time with the values of a given predictor randomly shuffled. The shuffling makes that predictor on the average unrelated to the response and all other predictors. For each shuffled predictor j, compute new measure of prediction error, ν_j.
3. For each of the p predictors, average over trees the difference between the prediction error with no shuffling and the prediction error with the jth predictor shuffled.

[14] Sometimes Granger-cause is called predictive cause.

The average increase in prediction error when a given predictor j is shuffled represents the importance of that predictor. That is,

$$\mathrm{I}_j = \sum_{k=i}^{K} \left[\frac{1}{K} (\nu_j - \nu) \right], \quad j = 1, \ldots, p, \tag{5.10}$$

where there are K trees, ν_j is the prediction error with predictor j shuffled, and ν is the prediction error with none of the predictors shuffled. It is sometimes possible for prediction accuracy to improve slightly when a variable is shuffled because of the randomness introduced. A negative measure of predictor importance follows. Negative predictor importance can be treated as no decline in accuracy or simply can be ignored.

As written, Eq. 5.10 is somewhat open-ended. The measures of prediction error (ν and ν_j) are not defined. As just noted, for a quantitative response variable, the MSE is an obvious choice. There are more options for categorical response variables: the deviance, percentage of cases classified incorrectly, average change in the margins, or some other measure. Currently, the preferred measure is the proportion (or percentage) of cases misclassified. This has the advantage of allowing direct comparisons between the increases in misclassification and all of the row summaries in a confusion table. In addition, all of the other measures considered to date have been found less satisfactory for one reason or another. For example, some measures are misleadingly sensitive; small changes in the number of classification errors can lead to large changes in the importance measure.

When used with categorical response variables, a significant complication is that Eq. 5.10 will almost always produce different importance measures for given predictors for different categories of the response. That is, there will be for any given predictor a measure of importance for each response class, and the measures will not generally be the same. For example, if there are three response classes, there will be three measures of importance for each predictor that will generally differ. Moreover, this can lead to different rankings of predictors depending on which response category is being considered. Although this may seem odd, it follows directly from the fact that the number of observations in each response class and the margins for each class will typically differ. Consequently, a given increase in the number of misclassifications can have different percentage impacts. A detailed illustration is presented shortly.

Partly in response to such complications, one can standardize the declines in performance. The standard deviation of $(\nu_j - \nu)$ over trees can be computed. In effect, one has a bootstrap estimate over trees of the standard error associated with the increase in classification error, which can be used as a descriptive measure of stability. Larger values imply less stability.

Then, one can divide Eq. 5.10 by this value. The result can be interpreted as a z-score so that importance measures are now all on the same scale. And with a bit of a stretch, confidence intervals can be computed and conventional hypothesis tests performed. It is a stretch because the sampling distribution of the predictor

importance measure is usually not known. Perhaps more important, the descriptive gains from standardization are modest at best, as the illustrations that follow make clear.

One of the drawbacks of the shuffling approach to variable importance is that only one variable is shuffled at a time. There is no role for joint importance over several predictors. This can be an issue when predictors are correlated. There will be a contribution to prediction accuracy that is uniquely linked to each predictor and joint contributions shared between two or more predictors. This can also be an issue when a single categorical predictor is represented by a set of indicator variables. The importance of the set is not captured.

There is currently no option in the random forest software to shuffle more than one variable at a time. However, it is relatively easy to apply the prediction procedure in random forests using as input the original dataset with two or more of the predictors shuffled. Then, Eq. 5.10 can be employed as before, where j would now be joined by other predictor subscripts. The main problem is that the number of potential joint contributions can be very large. In practice, some subset selection procedure is likely to be needed, perhaps based on substantive considerations.

It might seem that Granger's approach of examining forecasting skill with and without a given predictor included is effectively the same as Breiman's shuffling approach. And if so, one might consider, for instance, dropping sets of predictors to document their joint contribution. But actually, the two strategies are somewhat different. In Granger's approach, dropping or adding predictors to the model means that the model itself will be reestimated each time. So, the comparisons Granger favors are the result of different predictors being included *and different models*. The impact of neutralizing a predictor and changing the model are confounded. Under Breiman's approach, the model is not reconstructed. The shuffling is undertaken as an additional procedure with the model fixed.

In summary, for many scientists the ability to predict accurately in Brieman's sense is an essential measure of a model's worth. If one cannot predict well, it means that the model cannot usefully reproduce the empirical world. It follows that such a model has little value. And as now stressed a number of times, a model that fits the data well will not necessarily predict well. Put another way, out-of-sample performance is far more compelling than in-sample performance. The take-home message is simple: if prediction skill is the gold standard (or even just a very important criterion by which to evaluate a model), then a predictor's contribution to that skill is surely one reasonable measure of that predictor's importance.

5.6.2.1 Some Examples of Importance Plots with Extensions

Consider now a random forests analysis of data from an educational and job training program for homeless individuals. Because providing such services to homeless individuals was expensive, administrators wanted to know in advance which individuals

referred to the program would not likely be helped. For example, they may have had more fundamental needs such as treatment for drug dependence. At the same time, they wanted to make a special effort to identify individuals with promise and were prepared to accept a substantial number of individuals who would not find a steady job when predicted to do so. A provisional cost ratio of 4 to 1 was determined. It was 4 times worse to overlook a promising individual than to mistakenly decide that an individual was promising.

Random forests was applied to training data on a little less than 7000 individuals who had gone through their program. One of the primary outcomes was whether after finishing the program steady employment followed. It did for about 27 % of the graduates of the program. The response variable is still unbalanced, not nearly so seriously as in the past two examples.

Table 5.4 shows the confusion table that resulted. Consistent with stakeholder preferences, there are about 4 false positives for every false negative (i.e., 2626/606). 68 % of those who would find employment were accurately identified in advance. However, because of the imposed 4 to 1 cost ratio, a prediction of success on the job market would be wrong 67 % of the time.

However, we are focusing now on the variable importance plots shown in Fig. 5.3. Reduction in prediction accuracy is shown on the horizontal axis. The code for the random forest analysis and the subsequent importance plots is shown in Fig. 5.4. Keep in mind that although we are using Breiman's term "prediction accuracy," we are actually considering classification accuracy in OOB data.

The upper left figure shows unstandardized reductions in prediction accuracy for employment when each predictor is in turn randomly shuffled. The age at which an individual enters the program is the most important input. When that variable is shuffled, prediction accuracy declines about 2.5 percentage points (i.e., from 68 % to 65.5 %). The importance of all of the other predictors can be interpreted in the same fashion. The bottom four predictors make no contribution to predictive accuracy. Recall that contributions less than 0.0 result from the noise built into the random forests algorithm and in practice are taken to be equal to 0.0.

Predictor importance does not show *how* an input is related to the response. The functional form is not revealed, nor are any of the likely interaction effects with other inputs. Going back to our bread baking metaphor, each input is but an ingredient in a recipe. We can learn how important an input is for prediction, but nothing more.

Table 5.4 Confusion table for employment after training with a 4 to 1 target cost ratio (N = 6723)

	Forecast not employed	Forecast employed	Model error
Not employed	2213	2626	.54
Employed	606	1278	.32
Use error	.21	.67	Overall error = .48

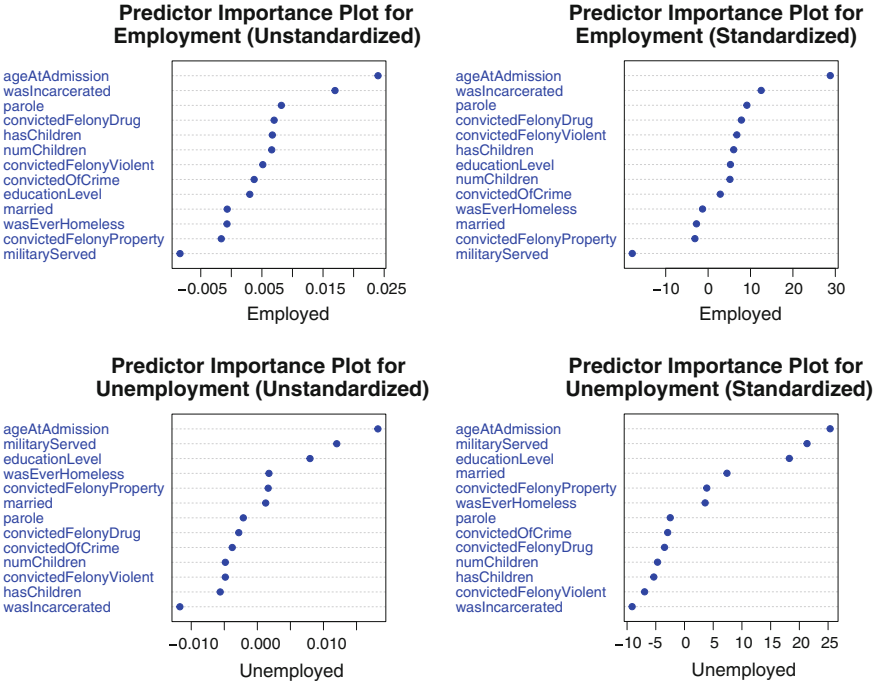


Fig. 5.3 Variable importance plots for employment outcome with a 4 to 1 target cost ratio (N = 6723)

Also not shown is prediction accuracy that is shared among inputs. Consequently, the sum of the individual contributions can be substantially less than 68 %.

The upper right figure shows the standardized contributions to employment prediction accuracy in standard deviation units.

The ordering of the inputs in the upper right figure has changed a bit because of the standardization, and as a descriptive summary, it is not clear what has been gained. It may be tempting to use each input's standard deviation, which can be easily extracted from the output, to construct confidence intervals. But, for a variety of technical reasons, this is not a good idea (Wager et al. 2014).

The bottom two figures repeat the same analyses, but using unemployment as the outcome. One might think that the figures in the bottom row would be very similar to the figures in the top row. Somewhat counterintuitively, they are not. But, recall how classification is accomplished in random forests. For a binary outcome, the class is assigned by majority vote. Two important features of those votes are in play here: the voting margin and the number of actual class members.

Consider a simple example in which there are 500 trees in the random forest. Suppose a given individual receives a vote of 251 to 249 to be assigned to the employment class category. The margin of victory is very small. Suppose that in fact that individual does find a job; the forecast is correct. Now a predictor is shuffled.

The vote might be very different. But suppose it is now just 249 to 251. Only two votes over tress have changed. Yet, the individual is now placed incorrectly in the unemployed class. This increases the prediction error by one individual.

Is that one individual increase in misclassifications enough to matter? Probably not given the usual sample sizes. But if a substantial number of the votes over trees is close, a substantial increase in the number of classification errors could result. And if the sample size is relatively small, the accuracy decline in units of percentage points could be relatively large. Such results are potentiated if the votes in terminal nodes within trees are close as well. Perhaps the key point is that these processes can differ depending on the outcome class, which explains why predictor importance can vary by the outcome class.

In Fig. 5.3, the prediction contributions are generally larger for the upper two figures than for the lower two figures. This is substantially a consequence of the marginal distribution of the response. There are more than twice as many individuals who do not find work compared to individuals who do. As a result, it would take many more classification changes after shuffling for the smaller prediction accuracy declines shown in the lower figures to approximate the larger predictor accuracy declines shown in the upper figures.

5.7 Input Response Functions

Predictor importance is only part of the story. In addition to knowing the importance of each input, it can be very useful to have a description of how each predictor is related to the response. The set of response functions needs to be described.

One useful solution based on an earlier suggestion by Breiman and his colleagues (1984) is "partial dependence plots" (Friedman 2001; Hastie et al. 2009: Sect. 10.13.2). For tree-based approaches, one proceeds as follows.

1. Grow a forest.
2. Suppose x_1 is the initial predictor of interest, and it has v distinct values in the training data. Construct v datasets as follows.
    a. For each of the v distinct values of x_1, make up a new dataset where x_1 only takes on that value, leaving all other variables untouched.
    b. For each of the v datasets, predict the response for each tree in the random forest. There will be for each tree a single value averaged over all observations. For numeric response variables, the predicted value is a mean. For categorical response variables, the predicted value is a proportion.

c. Average each of these predictions over the trees. The result is either an average mean or an average proportion over trees.
d. Plot the average prediction for each value for each of the v datasets against the v values of x_1

3. Go back to Step 2 and repeat for each predictor.

There is a lot going on in this algorithm that may not be immediately apparent. Partial dependence plots show the average relationship between a given input and the response within the fixed, joint distribution of the other inputs. For each of the v values, the values of all other inputs are always the same. Therefore, variation in these other inputs cannot explain away an empirically determined response function for those v values.

Perhaps here is a way to think about it. Suppose the selected input is age. One asks what would the average outcome be if everyone were 21 and nothing else changed? Then one asks, what would the average outcome be if everyone were 22 and nothing else changed? The same question is asked for age 23, age 24 and so on. All that changes is the single age assigned to each case: 21, 22, 23, 24 . . . .

Now in more detail, suppose the response is binary and initially everyone is assigned the age of 20. Much as one would do in conventional regression, the outcome for each case is predicted using all of the other inputs as well. Those fitted values can then be averaged. For a binary outcome, an average class is computed. That average is a proportion.[15] Next, everyone is assigned the age of 21, and the same operations are undertaken. For each age in turn, the identical procedure is applied. One can then see how the outcome class proportions change with age alone because as age changes, none of the other variables do. Should the response be quantitative, each fitted average is a conditional mean. One can then see how the conditional mean changes with age alone.

Commonly, the partial dependence is plotted. Unlike the plots of fitted values constructed from smoothers (e.g., from the generalized additive model), partial dependence plots usually impose no smoothness constraints, and the underlying tree structure tends to produce somewhat bumpy results. In practice, one usually overlays an "eyeball" smoother when the plot is interpreted. Alternatively, it is often possible to overlay a smoother if the software stores the requisite output. In R, *randomForest()* does.

For quantitative response variables, the units on the vertical axis usually are the natural units of the response, whatever they happen to be. For categorical response variables, the units of the response on the vertical axis are centered logits. Consider first the binary case.

Recall that logistic regression equation commonly is written as

$$\log\left(\frac{p}{1-p}\right) = \mathbf{X}\boldsymbol{\beta}, \tag{5.11}$$

[15] For example, if getting a job is coded 1 and not getting a job is coded 0, the mean of the 1s and 0s is the proportion who got a job.

where p is the probability of a success. The term on the left-hand side is the log of the odds of a success, often called the "logit". The change in the response for a unit change in a predictor, all other predictors held constant, is in "logits".

For the multinomial case, the most common approach is to build up from the familiar binary formulation. If there are K response categories, there are $K - 1$ equations, each of the same general form as Eq. 5.11. One equation of the K possible equations is redundant because the response categories are exhaustive and mutually exclusive. Thus, if an observation does not fall in categories $1, \ldots, K - 1$, it must fall in the Kth category. This implies that a single category can be chosen as the reference category, just as in the binomial case (i.e., there are two possible outcomes and one equation). Then, for each of the $K - 1$ equations, the logit is the log of the odds for a given category compared to the reference category.

Suppose there are four response categories, and the fourth is chosen as the reference category. There would then be three equations with three different responses, one for $\log(p_1/p_4)$, one for $\log(p_2/p_4)$, and one for $\log(p_3/p_4)$. The predictors would be the same for each equation, but each equation would have its own set of regression coefficients differing in values across equations.

One might think that partial dependence plots would follow a similar convention. But they do not. The choice of the reference category determines which logits will be used, and the logits used affect the regression coefficients that result. Although the overall fit is the same no matter what the reference category, and although one can compute from the set of estimated regression coefficients what the regression coefficients would be were another reference category used, the regression coefficients reported are still different when different reference categories are used.

There is usually no statistical justification for choosing one reference category or another. The choice is usually made on subject matter grounds to make interpretations easier, and the choice can easily vary from data analyst to data analyst. So, the need for a reference category can complicate interpretations of the results and means that a user of the results has to undertake additional work if regression coefficients using another reference category are desired.

Partly in response to these complications, partial dependence plots are based on a somewhat different approach. There are K, rather than $K - 1$, response functions, one for each response variable class. For the logistic model and class k, these take the form of

$$p_k(X) = \frac{e^{f_k(X)}}{\sum_{k=1}^{K} e^{f_k(X)}}. \tag{5.12}$$

There is still a redundancy problem to solve. The solution employed by partial dependence plots is the constraint $\sum_{k=1}^{K} f_k(X) = 0$. This leads to the multinomial deviance loss function and the use of a rather different kind of baseline.

Instead of using a given category as the reference, the unweighted mean of the proportions in the K categories is used as the reference. In much the same spirit as analysis of variance, the response variable units are then in deviations from a mean. More specifically, we let

$$f_k(X) = \log[\mathrm{p}_k(X)] - \frac{1}{K}\sum_{k=1}^{K}\log[\mathrm{p}_k(X)]. \tag{5.13}$$

Thus, the response is the difference between the logged proportion for category k and the average of the logged proportions for all K categories. The units are essentially logits but with the mean over the K classes as the reference. Consequently, each response category can have its own equation and, therefore, its own partial dependence plot. This approach is applied even when there are only two response categories, and the conventional logit formulation might not present interpretive problems.

To illustrate, consider once again the employment data. Suppose age in years is the predictor whose relationship with the binary employment response variable is of interest. And suppose for an age of say, 25 years, the proportion of individuals finding a job is .20 (computed using the partial dependence algorithm). The logit is $\log(.2) - [\log(.2) + \log(.8)]/2 = -0.693$ (using natural logarithms). This is the value that would be plotted on the vertical axis corresponding to 25 years of age on the horizontal axis. It is the log of the odds with mean proportion over the K categories as the reference.

The same approach can be used for the proportion of 25 year old individuals who do not find a job. That proportion is necessarily .80, so that value plotted is $\log(.8) - [\log(.2) + \log(.8)]/2 = 0.693$. In the binary case, essentially the same information is obtained no matter which response class is examined.

As required, $0.693 - 0.693 = 0$. This implies that one response function is the mirror image of the other. Thus, one partial dependence plot is the mirror image of the other partial dependence plot, and only one of the two is required for interpretation.

It is easy to get from the centered log odds to more familiar units. The values produced by Eq. 5.13 are half the usual log of the odds. From that, one can easily compute the corresponding proportions. For example, multiplying $-.693$ by 2 and exponentiating yields an odds of .25. Then, solving for the numerator proportion results in a value of .20. We are back where we started.[16]

Equation 5.13 would be applied for each year of age. Thus, for 26 years, the proportion of individuals finding a job might be .25. Then the value plotted on the horizontal axis would be 26, and the value on the vertical axis would be $\log(.25) - [\log(.25) + \log(.75)]/2 = -.549$.

The value of $-.549$ is in a region where the response function has been increasing. With one additional year of age, the proportion who find work increases from 0.20 to 0.25, which becomes log odds of -0.693 and -0.549 respectively. All other values produced for different ages can be interpreted in a similar way. Consequently, one can get a sense of how the response variable changes with variation in a given predictor, all other predictors held constant.

[16] $e^{[2(-.693)]}/(1 + (e^{[2(-.693)]}) = .20$.

5.7.1 Partial Dependence Plot Examples

Figure 5.5 shows two partial dependence plots constructed from the employment data with code that can be found toward the bottom of Fig. 5.4. The upper plot shows the relationship between the input age and the centered log odds of employment.[17] Age is quantitative. Employment prospects increase nearly linearly with age until about age 50 at which time they begin to gradually decline. The lower plot shows the relationship between the input education and the centered log odds of employment. Education is a factor. The employment prospects are best for individuals who have a high school or GED certificate. They are worst for individuals who have no high school degree or GED. Individuals with at least some college fall in between.[18] Program administrators explained the outcome for those with at least some college as a result of the job market in which they were seeking work. The jobs were for largely unskilled labor. There were not many appealing jobs for individuals with college credits.

In order to get a practical sense of whether employment varies with either input, it can be useful to transform the logits back to proportions. Suppose that one were interested in the largest gap in the prospects for employment. From the upper plot, the largest logit is about 0.39, and the smallest logit is about −0.42. These become proportions of 0.69 and 0.30 respectively. The proportion who find work nearly doubles. Age seems to be strongly associated with employment.

It is important not to forget that the response functions displayed in partial dependence plots reflect the relationship between a given predictor and the response, conditioning on all other predictors. All other predictors are being "held constant" in the manner discussed above. The code in Fig. 5.4 shows that there are a substantial number of such predictors.

When the response has more than three outcome categories, there are no longer the symmetries across plots that are found for binary outcomes. For a binary response variable, it does not matter which of the two categories is used when the partial dependence plot is constructed. One plot is the mirror image of the other. Figure 5.6 shows what can happen with three outcome categories.

The three employment outcomes in Fig. 5.6 are "no salary," "hourly salary," "yearly salary." The first category means that no job was found. For those who found a job, a yearly salary is for these individuals associated with higher status positions compared to positions offering an hourly salary. The plot at the top, for the yearly salary outcome, looks rather like the plot for finding any job. Prospects are bleak for those under 20, peak around 50 and then gradually taper off. The plot in the middle, for the hourly salary outcome, has the same general shape, but peaks around 40 and then falls of far more abruptly. The plot at the bottom, for no salary, looks much like the top plot, but with a sign reversal. No plot is the mirror image of another because if the outcome in question is does not occur, *one of two* other

[17] The term "centered" is used because mean of the K proportions is the reference.

[18] The bars use the value of zero as the base and move away from 0.0 upwards or downwards.

```
library(randomForest)
# random forests
rf1<-randomForest(Employed~ageAtAdmission+
convictedOfCrime+convictedFelonyViolent+
convictedFelonyProperty+convictedFelonyDrug+
wasIncarcerated+numChildren+parole
married+hasChildren+educationLevel+
wasEverHomeless+militaryServed,
data=TestData,importance=T,sampsize=c(1200,1100))

par(mfrow=c(2,2))

# Variable Importance Plots
varImpPlot(rf1,type=1,scale=F,class="Employed",
main="Forecasting Importance Plot for Employment
(Unstandardized)",col="blue",cex=1,pch=19)

varImpPlot(rf1,type=1,scale=T,class="Employed",
main="Forecasting Importance Plot for Employment
(Standardized)", col="blue",cex=1,pch=19)

varImpPlot(rf1,type=1,scale=F,class="Unemployed",
main="Forecasting Importance Plot for Unemployment
(Unstandardized)", col="blue",cex=1,pch=19)

varImpPlot(rf1,type=1,scale=T,class="Unemployed",
main="Forecasting Importance Plot for Unemployment
(Standardized)", col="blue",cex=1,pch=19)

# Partial Dependence Plots
part1<-partialPlot(rf1,pred.data=TestData,x.var=ageAtAdmission,
rug=T,which.class="Employed")

par(mfrow=c(2,1))

scatter.smooth(part1$x,part1$y,span=1/3,xlab="Age at Admission",
ylab="Centered Log Odds of Employment", main="Partial
Dependence Plot for Employment on Age",col="blue",pch=19)

part2<-partialPlot(rf1,pred.data=TestData,x.var=educationLevel,
rug=T,which.class="Employed", main="Partial Dependence
Plot for Employment on Education", xlab="Educational Level,
ylab="Centered Log Odds of Employment"),ylim=c(-.05,.25))
```

Fig. 5.4 R code for random forests analysis of an employment outcome

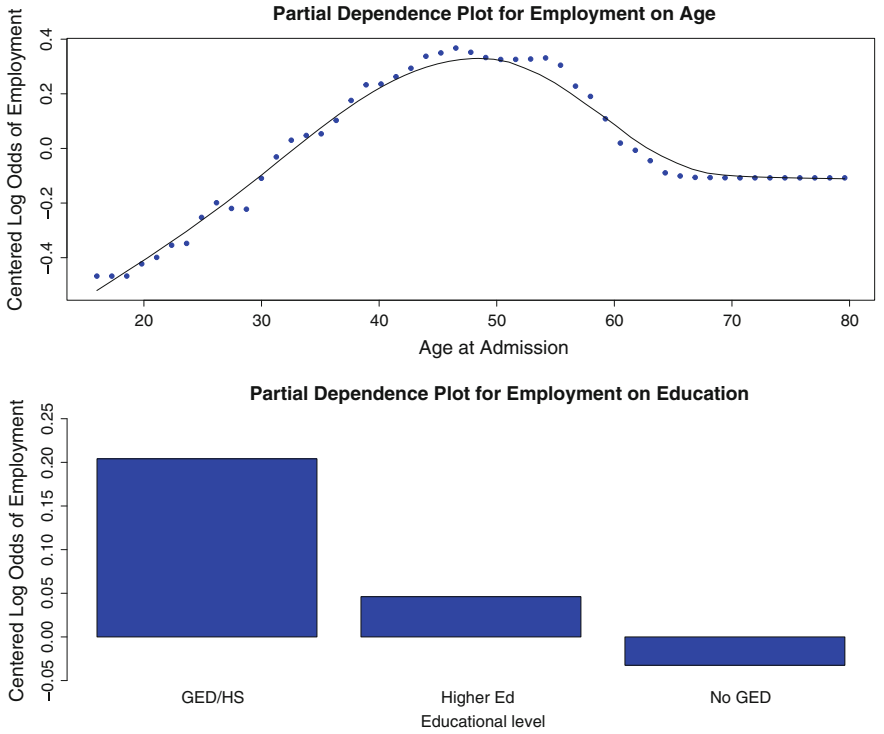


Fig. 5.5 Partial response plots for employment on age and education (N = 6723)

categories will. Moreover, the two other categories change depending on the single outcome whose centered logits are plotted.

In practice, each plot can have a story to tell. For example, comparing the top two plots, being over 50 years old is associated with a small decline in prospects for higher status jobs, In contrast, being over 50 years old is associated with a dramatic decline in prospects for lower status jobs. Part of the explanation can probably be found in the nature of the job. Hourly jobs will often require more physical capacity, which can decrease dramatically starting around age 50.

When there are more than two classes, working with units other than logits is more limited. Suppose the respective proportions at age 30 for the three classes represented in Fig. 5.5 are 0.15, 0.35, and 0.50 respectively. The three centered logits computed for the three response categories are respectively −0.68, 0.16, .52. As before, the sum of the values is 0.0. But there is no mirror image, and one cannot easily get from a single partial dependence plot back to underlying proportions. But one can work with odds.

For example, the largest logit in the top figure is about 0.19. The odds are 1.21. The smallest logit in the top figure is about −.61. The odds are 0.54. For individuals around 50 years old, their odds of getting a job with a yearly salary are about 1.2

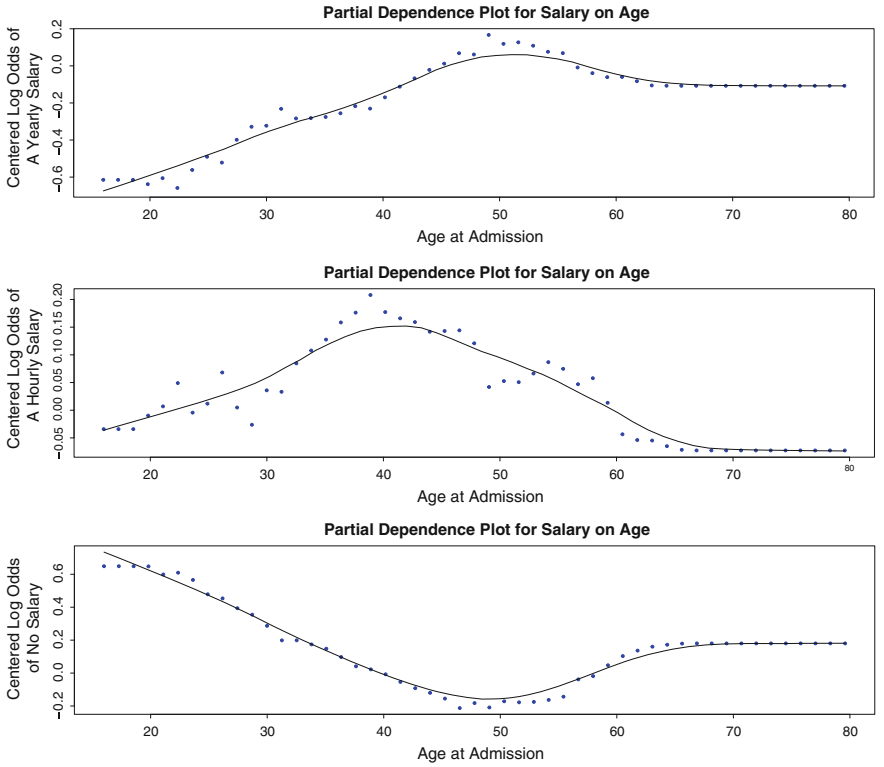


Fig. 5.6 Partial response plots for employment of yearly salary, weekly salary, or no salary on age (N = 6723)

odds units better than average. For individuals around 20 years of age, the odds of getting a job with a yearly salary are about 1.85 odds units worse than average (i.e., 1/.54). Once again, age at the extremes seems to matter substantially.

Finally, just as with variable importance plots, one does not really know *how* each input is linked to the outcome. In particular, variation in an input may well be partitioned between a large number of interaction effects. In econometric language, something akin to reduced form relationships are being represented.[19]

5.8 Classification and the Proximity Matrix

It can be interesting to determine the degree to which individual observations tend to be classified alike. In random forests, this information is contained in the "proximity matrix." The proximity matrix is constructed as follows.

[19]They are not literally reduced forms results because there is no structural model.

1. Grow tree as usual.
2. Drop all the training data (in-bag and out-of-bag) down the tree.
3. For all possible pairs of cases, if a pair lands in the same terminal node, increase their proximity by one.
4. Repeat Steps 1–4 until the designated number of trees has been grown.
5. Normalize by dividing by the number of trees.

The result is an $N \times N$ matrix with each cell showing the proportion of trees for which each pair of observations lands in the same terminal node. The higher that proportion, the more alike those observations are in how the trees place them, and the more "proximate" they are.

As noted earlier, working with a very large number observation can improve how well random forests performs because large trees can be grown. Large trees can reduce bias. For example, working with 100,000 observations rather than 10,000 can improve classification accuracy by as much as 50 %. However, because a proximity matrix is $N \times N$, storage can be a serious bottleneck. Storage problems can be partly addressed by only storing the upper or lower triangle, and there are other storage-saving procedures that have been developed. But large datasets still pose a significant problem.

Little subject matter sense can be made of an $N \times N$ matrix of even modest size. Consequently, additional procedures usually need to be applied. We turn to one popular option: multidimensional scaling.

5.8.1 Clustering by Proximity Values

The proportions in a proximity matrix can be seen as measures of similarly, and the matrix is symmetric with 1s along the main diagonal. Consequently, a proximity matrix can be treated as a similarity matrix in much the same spirit as some kernel matrices discussed earlier. As such, it is subject to a variety of clustering procedures with multidimensional scaling (MDS) the one offered by *randomForests()*. The results can be shown in a 2-dimensional plot. The two axes are the first and second principal components (i.e., with the two largest eigenvalues) derived from the proximity matrix. Observations closer in their Euclidian distances are more alike.[20]

Figure 5.7 shows an MDS plot from the proximity matrix constructed for the employment analysis. The results for the first two principle components of the proximity matrix are shown and clearly, there are discernible pattens. The red dots are for individuals who found employment, and the blue dots are for individuals who did

[20]The calculations are done by *cmdscale()*, and the plotting is done by *MDSplot()*. The latter automatically calls the former. However, with more than several hundred observations, there is so much overplotting that making sense of the results is very difficult. Labeling the points could help, but not with so much overplotting. Finally, the computational challenges are substantial. For the employment data (N = 6723), doing the MDS using *cmdscale()* on an iMac with an 3.4 GHz intel Core i7 and 32 GB of memory using took about 30 min.

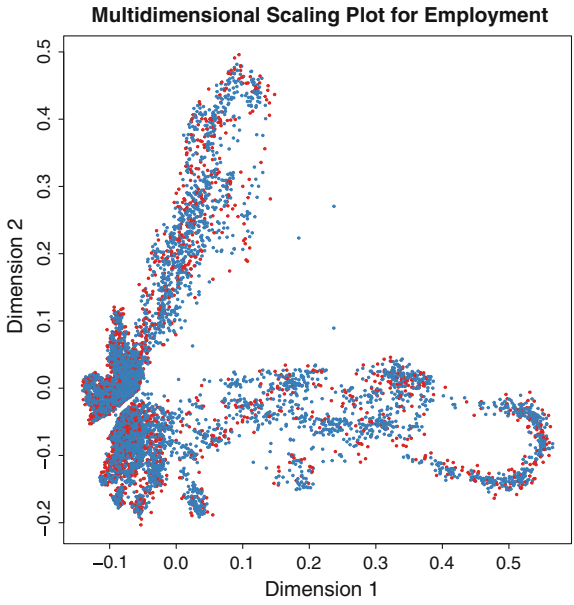


Fig. 5.7 Multidimensional scaling plot of the proximity matrix from the employment analysis: *red dots* for employed and *blue dots* for unemployed

not. Ideally, the individuals who found employment would tend to be alike, and the individuals who did not find employment would tend to be alike. That is, there would be lots of terminal nodes dominated by either employed individuals or unemployed individuals so that they would tend to fall in similar neighborhoods. However, it is difficult to extract much from the figure because of the overplotting. Working with a small random sample of cases would allow for a plot that could be more easily understood, but there would be a substantial risk that important patterns might be overlooked.

In short, better ways need to be found to visualize the proximity matrix. Multidimensional scale has promise, but can be very difficult to interpret with current graphical procedures and large Ns.[21]

5.8.1.1 Using Proximity Values to Impute Missing Data

There are two ways in which random forests can impute missing data. The first and quick method relies on a measure of location. If a predictor is quantitative, the median of the available values is used. If the predictor is categorical, the modal category from the available data is used. Should there be small amounts of missing data, this method may be satisfactory, especially given the computational demands of the second method.

[21] This is not to say that MDS is inappropriate in principle or that it will not work well for other kinds of applications.

The second method capitalizes on the proximity matrix in the following manner.

1. The "quick and dirty" method of imputation is first applied to the training data, a random forest is grown, and the proximity values computed.
2. If a missing value is from a quantitative variable, a weighted average of the values for the non-missing cases is used. The proximity values between the case with a missing value and all cases with non-missing values are used as the weights. So, cases that are more like the case with the missing value are given greater weight. All missing values for that variable are imputed in the same fashion.
3. If a missing value is from a categorical value, the imputed value is the most common non-missing value for the variable, with the category counts weighted, as before, by proximity. Again, cases more like the case with the missing value are given greater weight. All missing values for that variable are imputed in the same fashion.

The step using proximity values is then iterated several times. Experience to date suggests that four to six iterations is sufficient. But the use of imputed values tends to make the OOB measures of fit too optimistic. There is really less information being brought to bear in the analysis than the random forest algorithm knows about. The computational demands are also quite daunting and may be impractical for many datasets until more efficient ways to handle the proximities are found. Finally, imputing a single weighted value for each missing observation papers over chance variation in the imputation. How much this matters depends on whether a level II analysis is being undertaken. More will be said about that later.[22]

5.8.1.2 Using Proximities to Detect Outliers

The proximity matrix can be used to spot outliers in the space defined by the predictors. The basic idea is that outliers are observations whose proximities to all other observations in the data are small. The procedures in *randomForest()* to detect outliers are not implemented for quantitative response variables. For categorical response variables, outliers are defined within categories of the response variable. Within each observed outcome class, each observation is given a value for its "outlyingness" computed as follows.

1. For a given observation, compute the sum of the squares of the proximities with all of the other observations in the same outcome class. Then take the inverse. A large value will indicate that on the average the proximities are small for that observation; the observation is not much like other observations. Do the same for all other observations in that class. One can think of these values as unstandardized.
2. Within each class, compute the median and mean absolute deviation around the median of the unstandardized values.

[22]The assumed missing data mechanism is much like missing conditionally at random because of the proximity weighting.

3. Subtract the median from each of the unstandardized values and divide by the mean absolute deviation. In this fashion, the unstandardized values are standardized.
4. Values less than zero are set to 0.0.

These steps are then repeated for each class of the response variable. Observations with values larger than 10 can be considered outliers. Note that an observation does not become suspect because of a single atypical x-value. An outlier is defined by *how it is classified*, which is a function of all of its x-values. It typically lands in terminal nodes where it has little company.

Figure 5.8 is an index plot of outlier values for the employment data, with employed cases represented by red spikes and unemployed cases represented as blue spikes. The R code is shown in Fig. 5.9.

For this analysis, there are perhaps 4 to 6 observations of possible concern, but the outlier measures are close to 10 and with over 9000 observations overall, they would make no material difference in the results if dropped. It might be different if the outlier measures were in the high teens, and there were only a few hundred observations in total.

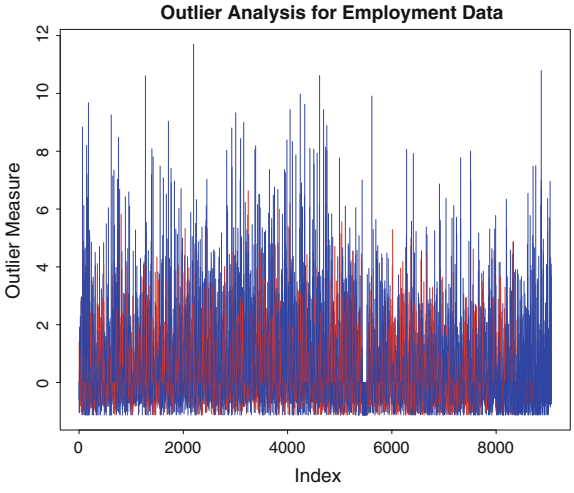


Fig. 5.8 Index plot of outlier measures for employment data with values greater than 10 candidates for deletion (*Red spike* are for employed and *blue spikes* are for unemployed)

```
plot(outlier(rf1),type="h", ylab="Outlier Measure",
     main="Outlier Analysis for Employment Data",
     col=c("red","blue")[as.numeric(TestData$Employed)])
```

Fig. 5.9 R code for index plot of outlier measures

When the data analyst considers dropping one or more outlying cases, a useful diagnostic tool can be a cross tabulation of the classes assigned for the set of observations that two random forest analyses have in common: one with all of the data and one with the outliers removed. If the common observations are, by and large, classified in the same way in both analyses, the outliers do not make an important difference to the classification process.

5.9 Empirical Margins

Recall Breiman's definition of a margin: the difference between the proportion of times over trees that an observation is correctly classified minus the largest proportion of times over trees that an observation is incorrectly classified. That definition can be used for each observation in a dataset. Positive values represent correct classifications, and negative values represent incorrect classifications.

Figure 5.10 shows histograms for two empirical margins. The upper plot is for individuals who were employed. The lower plot is for individuals who are not employed. We are conditioning on the actual response class much as we do for rows of a confusion table. For both histograms, the red, vertical line is located at a margin value of 0.0.

For employed individuals, the median margin is .18 (i.e., .59–.41), which translates into a solid majority vote or more for half of the correctly classified cases. Moreover, 68 % of the employed individuals are correctly classified. For unemployed individuals, the median margin is −.07. About 55 % of the unemployed individuals are incorrectly classified, although many by close votes.

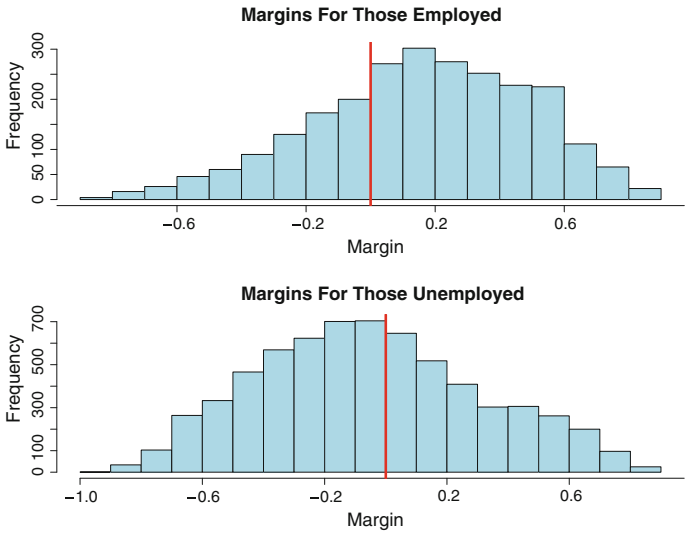


Fig. 5.10 Distribution of margins for employed and unemployed individuals (The *red vertical line* is located at a margin value of 0.0)

The margins can serve an additional performance diagnostic to supplement the rows of a confusion table. In addition to classification accuracy computed from each row, one can learn how definitive both the correct and incorrect classifications are. Ideally, the votes are decisive for the correctly classified cases and razor thin for the incorrectly classified cases. The results in Fig. 5.10 have some of this pattern, but only weakly so. Perhaps the major use of margins is to compare the stability of results from different datasets. One might have results from two datasets that classify with about the same accuracy, but for one of those, the classifications are more stable. The results from that dataset should be preferred because they are less likely to be a result of a random forest luck of the draw.

The stability is with respect to the random forest algorithm itself because margins are derived from votes over trees. The data are fixed. If a new random forest were grown with the same data, the classes assigned to all cases would not likely be the same. The uncertainty captured is created by the random forest algorithm itself. As such, it represents the in-sample reliability of the algorithm and says nothing about accuracy. Indeed, it is possible to have results that are reliably wrong.

There can be no margins in forecasting settings because the outcomes are not yet known. But the votes are easily retrieved by *randomForest()*, and can be very instructive. More decisive votes imply more reliable forecasts. For example, a school administrator may be trying to project whether a particular student will drop out of high school. If so, there may be good reason to intervene with services such as tutoring. The case for intervening should be seen as more credible if the drop out forecast is coupled with high reliability.

There is no clear threshold beyond which reliability is automatically high enough. That will be a judgment call for decision-makers. Moreover, that line may be especially hard to draw when there are more than two outcome classes. Suppose the vote proportions are 0.25, 0.30, and 0.45. The class with 45 % of the votes wins by a substantial plurality. But the majority of votes is cast against that class.

For real world settings in which forecasts from a random forest are made, the best advice may be to use the votes to help determine how much weight should be given to the forecast compared to the weight given to other information. In the school drop out example, an overwhelming vote projecting a drop out may mean discounting a student's apparent show of contrition and promises to do better. If the vote is effectively too close to call, the show of contrition and promises to do better may quite properly carry the day.

5.10 Quantitative Response Variables

There is not very much new that needs to be said about quantitative response variables once one appreciates that random forests handles quantitative response variables much as CART does. Even through for each tree there are still binary partitions of the data, there are no response variable classes and no response class proportions from which to define impurity. Traditionally, impurity is defined as the within-node

error sum of squares. A new partition of the data is determined by the split that would most reduce the sum, over the two prospective subsets, of the within-partition error sums of squares. Predicted values are the mean of the response variable in each of the terminal nodes. For each OOB observation, the mean of its terminal node is the fitted value assigned.

For regression trees, therefore, there are no classification errors, only residuals. Concerns about false negatives and positives and their costs are no longer relevant. There are no confusion tables and no measures of importance based on classification errors. To turn a regression tree into a fully operational random forest, the following steps are required.

1. Just as in the classification case, each tree is grown from a random sample (with replacement) of the training data.
2. Just as in the classification case, for each potential partitioning of the data, a random sample (without replacement) of predictors is used.
3. The value assigned to each terminal node is the mean of the response variable values that land in that terminal node.
4. For each tree in the random forest, the fitted value for each OOB case is the mean previously assigned to the terminal node in which it lands.
5. As before, random forest averages over trees. For a given observation, the average of the tree-by-tree fitted values is computed using only the fitted values from trees in which that observation was not used to grow the tree. This is the fitted value that random forest returns.
6. Deviations between these averaged, fitted values and the response variable observed values are used to construct the mean square error reported for the collection of trees that constitutes a random forest. The value of the mean square error can be used to compute a "pseudo" R^2 as $(1 - \text{MSE})/\text{Var}(Y)$.
7. Construction of partial dependence plots is done in the same manner as for classification trees, but now the fitted response is the set of conditional means for different predictor values, not a set of transformed fitted proportions.
8. Input variable importance is computed using the shuffling approach as before. And as before there is a "resubstitution" (in-sample) measure and a OOB (out-of-sample) measure. For the resubstitution measure, each time a given variable is used to define a partitioning of the data for a given tree, the reduction in the within-node error sum of squares is recorded. When the tree is complete, the reductions are summed. The result is a reduction in the error sum of squares that can be attributed to each predictor. These totals, one for each predictor, are then averaged over trees.

   The out-of sample importance measure is also an average over trees. For a given tree, the OOB observations are used to compute each terminal node's error sum of squares. From these, the mean squared error for that tree is computed. Then a designated predictor is shuffled, and mean square error for that tree is computed again. An increase in this mean square error is a decrease in accuracy. The same steps are applied to each tree, and the accuracy decreases are averaged over trees to get an average decrease in accuracy for that predictor. The standard deviation

of these decreases over trees can be used to standardize the average decrease, if that is desirable. The same process is employed for each predictor.

Despite the tight connection between regression trees and random forests, there are features found in some implementations of regression trees that have yet to be introduced into random forests, at least within R. But change is underway. Extensions to Poisson regression seem imminent (Mathlourthi et al. 2015), and Ishwaran and colleagues (2008) provide in R a procedure to do survival analysis (and lots more) with random forests using *randomForestSRC()*. Both alter the way splits within each tree are determined so that the reformulation is fundamental. For example, *randomForestSRC()* can pick the predictor and split that maximizes the survival difference in the two offspring nodes. There is also the option to do the analysis with competing risks (Ishwaran et al. 2014) and various weighting options that can be applied to the splitting rule (Ishwaran 2015).

Alternatively, *quantregForest()* only changes how values in each terminal node are used. The intent is to compute quantiles. Instead of storing only the mean of each terminal node as trees are grown, the entire distribution is stored. Recall the earlier discussion surrounding Table 5.2. Once the user decides which quantiles are of interest, they can be easily computed.

If one is worried about the impact of within-node outliers on the conditional mean, the conditional median can be used instead. If for substantive reasons there is interest in, say, the first or third quartile, those can be used. Perhaps most importantly, the quantile option provides a way to take the costs of forecasting errors into account. For example, if the 75th quantile is chosen, the consequences of underestimates are three times more costly than the consequences of overestimates (i.e., $75/25 = 3$).

However, such calculations only affect what is done with the information contained in the terminal nodes across trees. They do not require that the trees themselves be grown again with a linear loss function, let alone a loss function with asymmetric costs. In other words, the trees grown under quadratic loss are not changed. If there are concerns about quadratic loss, they do not apply to each of the splits. Moreover, all of the usual random forests outputs (e.g., variable importance plots) are still a product of a quadratic loss function.

5.11 A Random Forest Illustration Using a Quantitative Response Variable

Several years ago, an effort was made to count the number of homeless in Los Angeles County (Berk et al. 2008). There are over 2000 census tracts in the county, and enumerators were sent to a sample of a little over 500. Their job was to count "unsheltered" homeless who were not to be found in shelters or other temporary housing. Shelter counts were readily available. The details of the sampling need not trouble us here, and in the end, the overall county total was estimated to be about 90,000. We focus here on the street counts only.

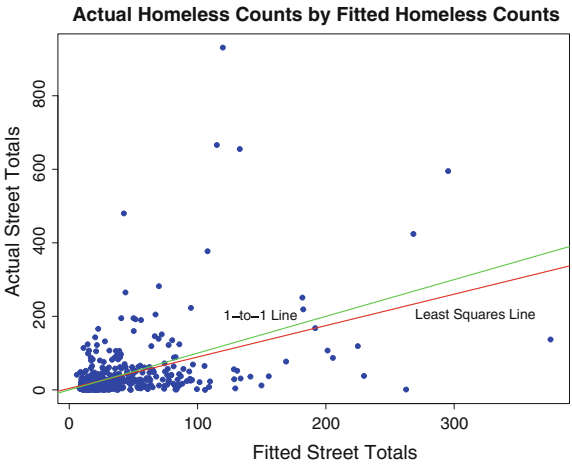


Fig. 5.11 For Los Angeles county census tracts a plot of actual homeless street counts against the random forest fitted homeless street counts (Least squares line is in *red*, the 1-to-1 line in *green*, N = 504)

In addition to countywide totals, there was a need to have estimated counts for tracts not visited. Various stakeholders might wish to have estimates at the tract level for areas to which enumerators were not sent. Random forests was used with tract-level predictors to impute the homeless street counts for these tracts. Here, we focus on the random forests procedure itself, not the imputation. About 21 % of the variance in the homeless street counts was accounted for by the random forests application with the follow inputs.[23]

1. MedianInc – median household income
2. PropVacant – proportion of vacant dwellings
3. PropMinority – proportion minority
4. PerCommercial – percentage of land used for commercial purposes
5. PerIndustrial – percentage of the land used for industrial purposes
6. PerResidential – percentage of the land used for residential purposes.

Figure 5.11 is a plot of the actual street totals against the fitted street totals in the data. One can see that there are a number of outliers that make any fitting exercise challenging. In Los Angeles county, the homeless are sprinkled over most census tracts, but a few tracts have very large homeless populations. The green 1-to-1 line summarizes what a good fit would look like. Ideally the fitted street counts and the actual street counts should be much the same. The red line summarizes with a least squares line the quality of the fit actually obtained. Were the horizontal axis extended to allow for fitted counts with the same large values as the actual counts, the two lines would diverge dramatically.

The counts in the highly populated census tracts are often badly underestimated. For example, the largest fitted count is around 400. There are 5 census tracts with

[23] As is often the case with quantitative response variables, the defaults in *randomForest()* worked well.

actual street counts over 400, one of those with a count of approximately 900. From a policy point of view this really matters. The census tracts most in need of services are not accurately characterized by random forests.

Figure 5.12 shows two variable importance plots. On the left, the percentage increase in average OOB mean squared error for each predictor is plotted. On the right, the increase in the average in-sample node impurity for each predictor is plotted. For example, when the percentage of land that is commercial is shuffled, the OOB mean squared error increases by about 7 %, and the proportion of the residential population that is minority has no predictive impact whatsoever. When household median income is shuffled, average node impurity increases by 7e+05, and all of the predictors have some impact on the fit. The ranking of the variables changes depending on the importance measure, but for the reasons discussed earlier, the out-of-sample measures is preferable, especially if forecasting is an important goal.

Figure 5.13 is a partial dependence plot showing a positive association between the percentage of the residential dwellings that are vacant and the number of homeless counted. When vacancy is near zero, the average number of homeless is about 20 per tract. When the vacancy percent is above approximately 10 %, the average count increases to between 60 and 70 (with a spike right around 10 %). The change is very rapid. Beyond 10 % the relationship is essentially flat. At that point, perhaps the needs of squatters are met.

In response to the poor fit for the few census tracts with a very large number of homeless individuals, it is worth giving *quantregForest()* a try. As already noted, a random forest is grown as usual, but the distributions in the terminal nodes of each

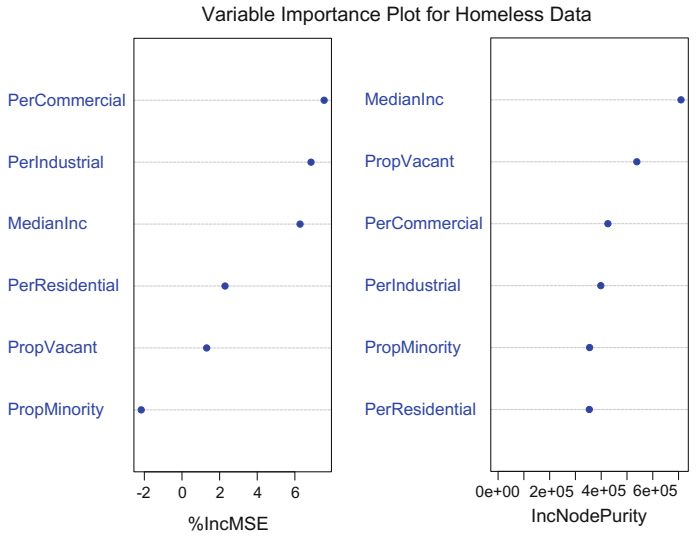


Fig. 5.12 Variable importance plots for street counts (On the *left* is the percent increase in the OOB mean squared error, and on the *right* is the in-sample increase in node impurity. N = 504)

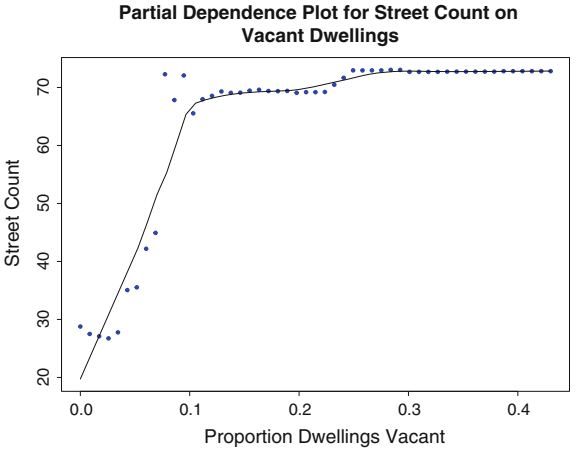


Fig. 5.13 The partial response plot for street counts on the proportion of vacant dwellings in a census tract (N = 504)

tree are retained for analysis. We will consider the impact of using three different quantiles: 0.10, 0.50, and 0.90. For these data, quantiles substantially larger than .50 would in practice usually be the focus. The code used in the analyses to follow is shown in Fig. 5.16.[24]

Figure 5.14 shows three plots laid out just like Fig. 5.11, for the conditional quantiles of 0.10, 0.50 and 0.90. All three fit the data about equally well if the adjusted R^2 is the measure because the outliers are very few. But should there be special concerns about the half dozen or so census tracts with very large homeless counts, it may make sense to fit the 90th percentile. For example, the largest fitted value for conditional 10th percentile is about 37. The largest fitted value for conditional 50th percentile is about 220. The largest fitted value for conditional 90th percentile is nearly 700. Several tracts with large homeless counts are still badly underestimated, but clearly, the small number of tracts with substantial number of homeless is better approximated. Whether this is appropriate depends on the relative costs of underestimates to overestimates. We are again faced with the prospect of asymmetric costs, but for quantitative response variables.

When for the homeless data the 90th percentile is used, the cost of underestimating the number of homeless in a census tract is 9 times more costly than overestimating the number of homeless in a census tract (i.e., $0.90/0.10 = 9$). Whether the 9-to-1 cost ratio makes sense is a policy decision. What are the relative costs of underestimating the number of homeless compared to overestimating the number of homeless? More will be said about these issues when quantile boosting is discussed in the next chapter. But just as for classification, there is no reason to automatically impose symmetric costs.

[24]The authors are Nicolai Meinshausen and Lukas Schiesser. The version used for these analyses (version 1.1) seems to have some bugs in the plotting routines, which is why the code shown in Fig. 5.16 is so lengthy and inelegant. The plots had to be constructed from more basic procedures.

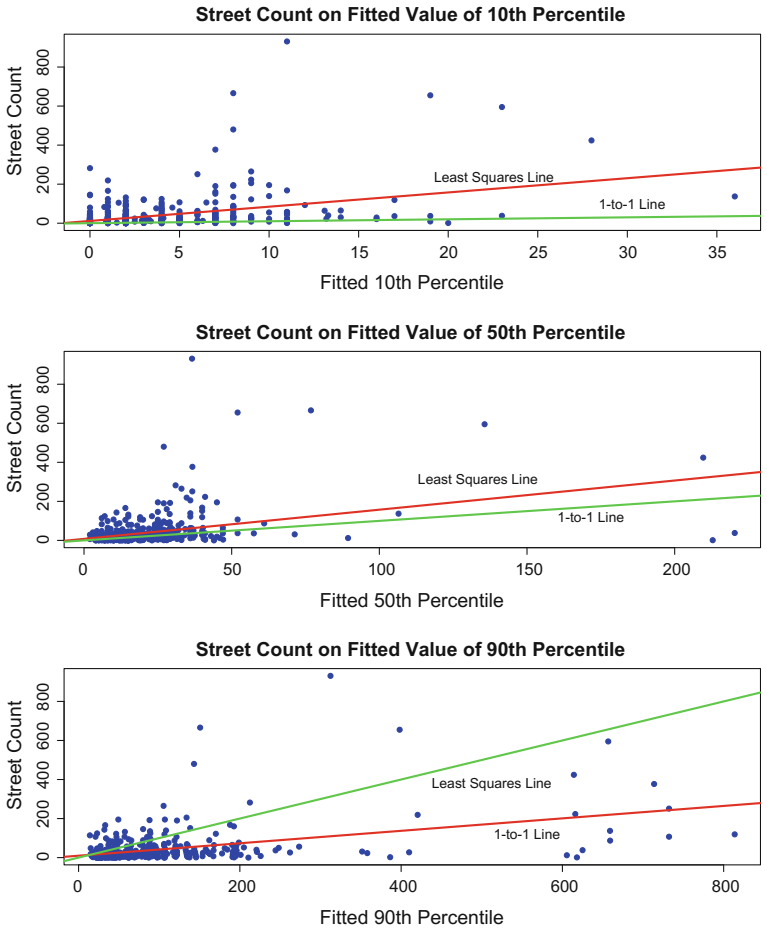


Fig. 5.14 Plot of quantile random forest fitted values against the actual values (Least squares line in *red* and 1-to-1 line in *green*. Quantiles = 0.05, 0.50, 0.90. N = 504)

Figure 5.15 shows three variable importance plots, one for each of the three quantiles being used: 0.10, 0.50, 0.90. The percentage point increase in quantile (linear) out-of-sample loss is the measure of importance. For example, for the conditional 90th percentile, the most important input is the proportion of vacant dwellings, which when shuffled increased the out-of-sample L_1 loss by about 16 percentage points. As before, negative values are treated as 0.0.

Perhaps the most important message is that by this measure of importance, the order of the variables can vary by the conditional quantile estimated. Inputs that are most important for out-of-sample performance when the 90th percentile is the fitted value may not be the most important for out-of-sample performance when the 10th percentile is the fitted value. That is, predictors that help to distinguish between

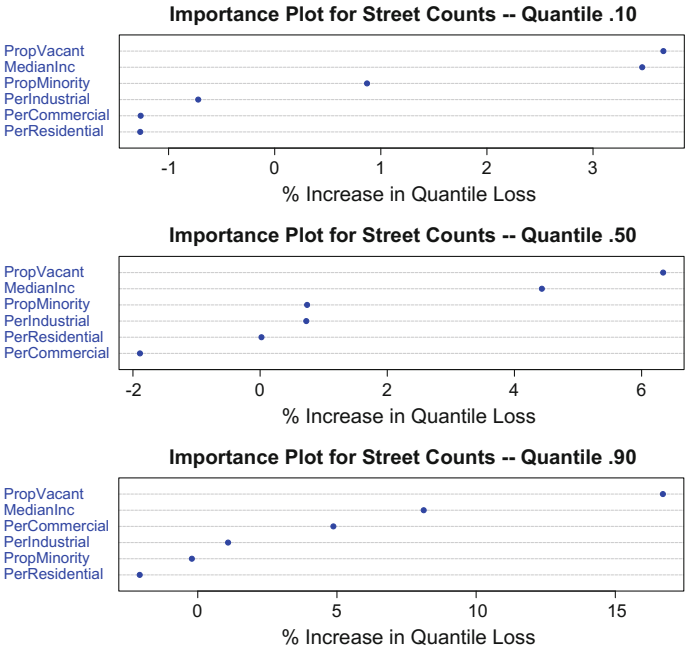


Fig. 5.15 Street count variable importance plots for quantiles of 0.10, 0.50, and 0.90 (Importance is measured by the OOB percentage point increase in the quantile loss after shuffling. N = 504)

census tracts with large numbers of homeless may be of no help distinguishing between census tracts with small numbers of homeless. Different processes may be involved.

There are apparently no partial dependence plots of quantile regression forests. It seems that relatively modest changes in the partial dependence plot algorithm could accommodate conditional quantiles. However, the computation burdens may be substantially increased.

Quantile random forests has some nice features and in particular, the ability to introduce asymmetric costs when the response variable is quantitative. However, the random forest is grown as usual with each split based on quadratic loss. Were one truly committed to linear loss, it would make sense to revise the splitting criterion accordingly. An R library called yarp(), by Adam Kapelner, is under development that among other features will allow for a wide variety of split loss functions. Quantile random forests is a strange hybrid and so far at least, does not seem to be widely used.

5.12 Statistical Inference with Random Forests

As long as users of random forests are content to describe relationships in the data on hand, random forests is a level I procedure. But the use of OOB data to get honest performance assessments and measures of predictor performance speaks to level II

```
library(quantregForest)
X<-as.matrix(HData[,2:7]) # predictors as matrix
Y<-as.numeric(as.matrix(HData[,1])) # response as vector

# Quantile Random Forests
out2<-quantregForest(x=X,y=Y,nodesize=10,importance=T,
                     quantiles = c(.10))
preds<-predict(out2) # Fitted OOB values

# Fitted value plots
par(mfrow=c(3,1))
plot(preds[,1],Y,col="blue",pch=19,
     xlab="Fitted 10th Percentile",
     ylab="Street Count",
     main="Street Count on Fitted Value of 10th Percentile")
abline(lsfit(preds[,1],Y),col="red",lwd=2)
abline(0.0,1.0,lwd=2,col="green")
text(22,220,"Least Squares Line",cex=1)
text(30,90,"1-to-1 Line",cex=1)

plot(preds[,2],Y,col="blue",pch=19,
     xlab="Fitted 50th Percentile",
     ylab="Street Count",
     main="Street Count on Fitted Value of 50th Percentile")
abline(lsfit(preds[,2],Y),col="red",lwd=2)
abline(0.0,1.0,lwd=2,col="green")
text(130,300,"Least Squares Line",cex=1)
text(170,110,"1-to-1 Line",cex=1)

plot(preds[,3],Y,col="blue",pch=19,
     xlab="Fitted 90th Percentile",
     ylab="Street Count",
     main="Street Count on Fitted Value of 90th Percentile")
abline(lsfit(preds[,3],Y),col="red",lwd=2)
abline(0.0,1.0,lwd=2,col="green")
text(500,370,"Least Squares Line",cex=1)
text(550,110,"1-to-1 Line",cex=1)

# Importance Plots
par(mfrow=c(3,1))
imp10<-sort(out2$importance[,1])
dotchart(imp10,col="blue",pch=19,xlab="% Increase
         in Quantile Loss",
         main="Importance Plot for Street Counts -- Quantile .10")
imp50<-sort(out2$importance[,2])
dotchart(imp50,col="blue",pch=19,xlab="% Increase
         in Quantile Loss",
         main="Importance Plot for Street Counts -- Quantile .50")
imp90<-sort(out2$importance[,3])
dotchart(imp90,col="blue",pch=19,xlab="% Increase
         in Quantile Loss",
         main="Importance Plot for Street Counts -- Quantile .90")
```

Fig. 5.16 R code for quantile random forests

concerns and generalization error in particular. If the forecasting is an explicit goal, a level II analysis is being undertaken.

Taking level II analyses a step farther, there have been some recent efforts to provide a rationale and computational procedures for random forests statistical inference (Wager 2014; Wager et al. 2014; Wager and Walther 2015; Mentch and Hooker 2015). The issues are beyond the scope of our discussion in part because the work is at this point still very much in progress. Key complications are the inductive nature of random forests, tree depth dependence on sample size, the sampling of predictors, and summary values for terminal nodes computed from the same data used to grow the trees.

However, if one has test data, one can proceed in the same spirit as in CART. The estimation target is the fitted values from the very same random forest grown with the training data. The test data can be used to estimate generalization error or other features of a confusion table. One can then apply the pairwise (nonparametric) bootstrap to the test data in the fashion discussed in earlier chapters.

5.13 Software and Tuning Parameters

In this chapter, all empirical work has been done with the R procedure *randomForest()*, which works well even for large datasets. But there are some disciplines in which the datasets are extremely large (e.g., 1,000,000 observations, and 5000 predictors) and working with subsets of the data can be counterproductive. For example, in genomics research there may be thousands of predictors.

Over the past few years, new implementations of random forests have been written for R, and some are remarkably fast (Ziegler and König 2014). A recent review by Wright and Ziegler (2015:1) confirms that *randomForest()* is "feature rich and widely used." But the code has not been optimized for high dimensional data.

Wright and Ziegler describe their own random forests implementation, *ranger()*, in some depth. It is indeed very fast, but lacks a number of features that can be very important in practice. All of the other implementations considered are either no more optimized than *randomForest()*, or run faster but lack important features (e.g., partial dependence plots). No doubt, at least some of these packages will add useful features over time. One possible candidate in R is *Rborist()* (Seligman 2015). The programs *rforest()* (Zhang et al. 2009) and *rjungle()* (Schwartz et al. 2010) are also candidates, but neither is currently available in R.[25] There are, in addition, efforts to reconsider random forests more fundamentally for very high dimensional data. For example, Xu and colleagues (2012) try to reduce the number of input dimensions by taking into account information much like that assembled in the *randomForest()* proximity matrix. Readers intending to use random forests should try to stay informed about these developments. Here, we will continue with *randomForest()*.

[25] *Rborist* wins the award for the most clever name.

Despite the complexity of the random forest algorithm and the large number of potential tuning parameters, most of the usual defaults work well in practice. However, if one tunes from information contained in the OOB confusion table, the OOB data will slowly become tainted. For example, if for policy or subject matter reasons one needs to tune to approximate a target asymmetric cost ratio in a confusion table, model selection is in play once again. Still, when compared to the results from true test data, the OOB results usually hold up well if the number of cost ratios estimated is modest (e.g., <10) and the sample size is not too small (e.g., >1000). The same holds if on occasion some of the following tuning parameters have to be tweaked.

1. *Node Size* — Unlike in CART, the number of observations in the terminal nodes of each tree can be very small. The goal is to grow trees with as little bias as possible. The high variance that would result can be tolerated because of the averaging over a large number of trees. In the R implementation of random forests, the default sample sizes for the terminal nodes are one for classification and five for regression. These seem to work well. But, if one is interested in estimating a quantile, such as in quantile random forests, then terminal node sizes at least twice as large will often be necessary for regression. If there are only five observations in a terminal node, for instance, it will be difficult to get a good read on, say, the 90th percentile.
2. *Number of Trees* — The number of trees used to constitute a forest needs to be at least several hundred and probably no more that several thousand. In practice, 500 trees is often a good compromise. It sometimes makes sense to do most of the initial development (see below) with about 500 trees and then confirm the results with a run using about 3000 trees. But, the cost is primarily computational time and only if the number of inputs and number of observations are large, do computational burdens become an issue. For example, if there are 100 inputs and 100,000 observations, the number of trees grown becomes an important tuning parameter.
3. *Number of Predictors Sampled* — The number of predictors sampled at each split would seem to be a key tuning parameter that should affect how well random forests performs. Although it may be somewhat surprising, very few predictors need to be randomly sampled at each split, and within sensible bounds on the number sampled, it does not seem to matter much for the OOB error estimates. With a large number of trees, each predictor will have an ample opportunity to contribute, even if very few are drawn for each split. For example, if the average tree in a random forest has ten terminal splits, and if there are 500 trees in the random forest, there will be 5000 chances for predictors to weigh in. Sampling two or three each time should then be adequate unless the number of predictors is quite large (e.g., > 100).
   But a lot depends on the number of predictors and how strongly they are related. If the correlations are substantial, it can be useful to reduce the number of predictors sampled for each partitioning decision. In the original manual for the FORTRAN version of random forests, Breiman recommends starting with the

number of predictors sampled equal to the square root of the number of predictors available. Then, trying a few more or a few less as well can be instructive.

The feature of random forests that will usually make the biggest difference in the results is how the costs of false negatives and false positives are handled, or for quantile random forests, the quantile used. Even through asymmetric costs are introduced by altering one or more of the arguments in *randomForest()*, one should not think of the target cost ratio as a tuning parameter. It is a key factor in the fitting process determined in advance from substantive and policy considerations. However, to arrive at a good approximation of the target cost ratio, some tuning of one or more arguments will usually be necessary (e.g., *sampsize()*).

Computational burdens can be an issue when the training data have a very larger number of observations (e.g., $>100,000$), when the number of inputs is large (e.g., >100), and when a substantial number of inputs are categorical with many classes.[26] It is difficult to tune one's way out of letting the algorithm grind for hours in part because with each new set of tuning values, the algorithm has to run again. Sometimes a better strategy is to work with a random, modest sized subset of training data for tuning, saving the bulk of the data for results that will be used. Doing some initial screening of the predictors to be used can also help, as long as one is aware of the risks. Combining some of the categories for factors with many levels is worth a try. Finally, many of the computational steps in random forests are easily parallelized and will run well on computers with multiple processors. Soon, software with these capabilities and others that increase processing speed will be routinely available and be richly endowed with desirable features.

Also, a cautionary note. Random forests is not designed to be a variable selection procedure. Nevertheless, it can be temping to use the variable importance plots to discard weak predictors. There are at least four problems with this approach. First, there is rarely any need to reduce the number of predictors. The way in which splits are determined for each tree in the forest is a kind of provisional, variable selection method that performs well. In other words, there is almost never a need to drop the unimportant variables and rerun random forests. Second, some argue that if multicolinearity is a serious problem, random forests results can be unstable. But that concern refers primarily to estimates of variable importance. Should that form of instability become an issue, any dimension reduction in the set of predictors is probably best done before the random forests analysis begins. However, one is back in the model selection game. Third, if the goal is to use variable importance to determine the predictors to be introduced into some other procedure, performance in prediction may not be what is needed. For example, prediction importance may be unrelated to causal importance. Finally, as discussed earlier, there are lots of worthy procedures designed for variable/model selection as long as one is prepared to address the damage usually done to level II analyses.

[26] Currently, up to 53 classes are allowed for any given categorical input in *randomForest()*.

5.14 Summary and Conclusions

There is substantial evidence that random forests is a very powerful statistical learning tool. If forecasting accuracy is one's main performance criterion, there are no other general purpose tools that have been shown to consistently perform any better. Moreover, random forests comes with a rich menu of post-processing procedures and simple means with which to introduce asymmetric costs. We consider a chief competitor in the next chapter.

But a lot depends on the data analysis task. We will later briefly address deep learning, which for specialized applications has enormous promise with very high dimensional data when great precision in the fitted values is needed. But one must be prepared to invest substantial time (e.g., weeks) in tuning, even when there is access to a very large number of CPUs and GPUs. As some advertisements warn,"don't try this at home." The most powerful desktop and laptop computers can be overmatched by deep learning.

Random forests seems to get its leverage from five features of the algorithm:

1. growing large, low bias trees;
2. using bootstrap samples as training data when each tree is grown;
3. using random samples of predictors for each partitioning of the data;
4. constructing fitted values and output summary statistics from the out-of-bag data; and
5. averaging over trees.

At the same time, very few of random forests' formal properties have been proven. At a deeper level, the precise reasons why random forests performs so well are not understood. There is some hard work ahead for theoretical statisticians. Nevertheless, random forests is gaining in popularity because it seems to work well in practice, provides lots of flexibility, and in R at least, comes packaged with a number of supplementary algorithms that provide a range of useful output.

Exercises

Problem Set 1

The goal of this first exercise is to compare the performance of linear regression, CART, and random forests. Construct the following dataset in which the response is a quadratic function of a single predictor.

```
x1=rnorm(500)
x12=x1^2
y=1+(-5*x12)+(5*rnorm(500))
```

1. Plot the $1 + (-5 \times$ x12$)$ against x1. This is the "true" relationship between the response and the predictor without the complication of the disturbances. This is the $f(X)$ you hope to recover from the data.
2. Proceed as if you know that the relationship between the response and the predictor is quadratic. Fit a linear model with x12 as the predictor. Then plot the fitted values

against x1. The results show how the linear model can perform when you know the correct function form.
3. Now suppose you do not know that the relationship between the response and the predictor is quadratic. Apply CART to the same response variable using rpart() and x1 as the sole predictor. Use the default settings. Construct the predicted values, using predict(). Then plot the fitted values against x1. How do the CART fitted values compare to the linear regression fitted values? How well does CART seem to capture the true $f(X)$?
4. Apply random forests to the same response variable using randomForests() and x1 as the sole predictor. Use the default settings. Construct the predicted values using predict(). Then plot the fitted values against x1. How do the random forest fitted values compare to the linear regression fitted values? How well does random forests seem to capture the true $f(X)$?
5. How do the fitted values from CART compare to the fitted values from random forests? What feature of random forests is highlighted?
6. Construct a partial dependence plot with x1 as the predictor. How well does the plot seem to capture the true $f(X)$?
7. Why in this case does the plot of the random forest fitted values and the partial dependence plot look so similar?

5.14.1 Problem Set 2

Load the dataset SLID from the *car* library. Learn about the data set using the help() command. Treat the variable "wages" as the response and all other variables as predictors. The data have some missing values you will want to remove. Try using na.omit().

1. Using the default settings, apply random forests and examine the fit quality.
2. Set the argument *mtry* at 4. Apply random forests again and examine fit quality. What if anything of importance has changed?
3. Now set ntrees at 100 and then at 1000 applying random forests both times. What if anything of importance has changed?
4. Going back to the default settings, apply random forests and examine the variable importance plots with no scaling for each predictor's standard deviation. Explain what is being measured on the horizontal axis on both plots when no scaling for the standard deviation is being used. Interpret both plots. If they do not rank the variables in the same way, why might that be? Now scale the permutation-based measure and reconstruct that plot. Interpret the results. If the ranks of the variables differ from the unscaled plot, why might that be? Focusing on the permutation-based measures (scaled and unscaled) when might it be better to use one rather than the other?
5. Construct partial dependence plots for each predictor and interpret them.

5.14.2 Problem Set 3

Load the *MASS* library and the dataset called Pima.tr. Read about the data using help().

1. Apply random forests to the data using the diagnosis of diabetes as the response. Use all of the predictors and random forest default settings. Study the confusion table.
   a. How accurately does the random forests procedure forecast overall?
   b. How accurately does the random forests procedure forecast each of the two outcomes separately (i.e., *given* each outcome)? (Hint: you get this from the rows of the confusion table.)
   c. If the results were used to forecast either outcome (i.e., *given* the forecast), what proportions of the time would each of the forecasts be incorrect? (Hint: you get this from the columns of the confusion table.)
2. Construct variable importance plots for each of the two outcomes. Use the unscaled plots of forecasting accuracy. Compare the two plots.
   a. Which predictors are the three most important in forecasts of the presence of diabetes compared to forecasts of the absence of diabetes? Why might they not be the same?
3. Construct and interpret partial dependence plots of each predictor.
4. Suppose now that medical experts believe that the costs of failing to identify future cases of diabetes are four times larger than the costs of falsely identifying future cases of diabetes. For example, if the medical treatment is to get overweight individuals to lose weight, that would likely be beneficial even if the individuals were not at high risk for diabetes. But failing to prescribe a weight loss program for an overweight individual might be an error with very serious consequences. Repeat the analysis just completed but now taking the costs into account by using the stratified bootstrap sampling option in random forests.
   a. How has the confusion table changed?
   b. How have the two variable importance plots changed?
   c. How have the partial dependence plots changed?
5. Plot the margins to consider the reliability of the random forests classifications. You will need at least *margin()* followed by the *plot()*. Are the two classes correctly classified with about the same reliability? If so, why might a physician want to know that? If not, why might a physician want to know that?
6. The votes are stored as part of the random forests object. Construct a histogram of the votes separately for each of the two outcome classes. How do votes differ from margins?

7. Now imagine that a physician did not have the results of the diabetes test but wanted to start treatment immediately, if appropriate. Each of the predictors are known for that patient but not the diagnosis. Using the predictor values for that patient, a random forests forecast is made. What should the physician use to measure the reliability of that forecast? Given some examples of high and low reliability.