Statistical Methods for Mn/Model Phase 4

Final Report

Prepared by:

Gary W. Oehlert, Ph.D.
Brian Shea

School of Statistics
University of Minnesota
313 Ford Hall
224 Church St. SE
Minneapolis, MN 55455

August 2007

Published by:

Research Services Section
Minnesota Department of Teransportation
Transportation Building, MS 330
395 John Ireland Boulevard
St. Paul, MN 55155

Acknowledgments

We would like to thank Dr. Elizabeth Hobbs at Mn/DOT for providing data, feedback, and the opportunity to work on this project. She was also able to keep us grounded in reality regarding what could and could not be implemented in GIS.

Table of Contents

List of Tables

List of Figures

Executive Summary

Mn/Model is a project that combines landscape and archaeological databases in a Geographic Information System (GIS) with statistical prediction methods to provide an estimate of the risk that a given location contains archaeological artifacts. Planners use these estimates both to seek out areas of low risk and to accommodate areas of high risk when planning transportation projects. Obviously, more accurate risk estimates lead to improved planning and reduced costs.

Mn/Model is about to move into its fourth phase, which will include improved landscape and archaeological data. At this time, Mn/DOT wishes to reconsider the statistical prediction methods used in Phase 3 to determine if better alternatives are available.

This project proposed and compared eight prediction methods, the Phase 3 method and seven alternatives. The methods were logistic regression with BIC model selection (the Phase 3 approach), logistic regression with Bayesian model averaging, naïve Bayes classification, tree-structured regression, "bumped" trees, "bagged" trees, "double bagged" trees, and "boosted" trees. Bumping, bagging, and boosting are examples of "perturb and aggregate methods," which repeatedly modify the data in minor ways and then combine the predictions from the modified data sets. Overall, bagging, double bagging, and boosting had the best predictive ability.

We recommend that bagged trees, or bagging, be the default prediction method for Phase 4. Bagging is easier to do in S-Plus (the statistical software used) than boosting and easier to implement in the GIS framework. Bagging provides substantial improvement in predictive capability over the Phase 3 method. Tree-structured models are also fairly easy to explain to the general public. Double bagging provides a small improvement over bagging, but at the cost of substantially more effort in implementation.

1. Introduction

Mn/Model is a predictive model for precontact archaeological site location in Minnesota (Hudak et al. 2002). Its goal is to use landscape variables such as proximity to water, elevation, aspect, and soil type to classify the landscape into those locations where archaeological sites are more likely and those where they are less likely. The motivation for Mn/Model was, and is, that the Minnesota Department of Transportation (Mn/DOT) is required by Section 106 of the National Historic Preservation Act of 1966, as amended, to make a good faith effort to identify historic properties within any project area and assess the impact of the project on those cultural resources. This assessment must be done before the project can proceed. Accurate prediction of the location of cultural artifacts, or at least of landscapes where those artifacts are more likely to be found, thus speeds the process and provides potential cost savings.

The Mn/Model project began in 1995 and moved through Phase 3 in 1999. It was funded by Mn/DOT using federal money available through the Intermodal Surface Transportation Efficiency Act. The scope of Mn/Model is geographically state-wide in Minnesota and temporally pre-1837 for cultural resources. Mn/Model provides a set of digital maps providing an assessments of both how likely a location is to have historic properties and how likely similar locations are to have been surveyed. These two bits of information help guide planners as to where surveys for cultural artifacts should be concentrated.

Functionally, Mn/Model has landscape and cultural resource data stored as GIS layers, and predictive models are constructed for sub-regions of the state (for example, the Big Woods or the Anoka Sand Plain). The landscape variables in GIS are primarily those that are not generally influenced by humans (for example, soil type, elevation, distance to water). In some cases, there are variables that have changed since European settlement (for example, forest cover type), but data are available for the pre-European settlement era. Because archaeological sites are rare in Minnesota, we compare landscape variables at known sites with landscape variables at random locations, the assumption being that nearly all random locations do not contain archaeological artifacts.

The Mn/Model project builds the regional maps in a sequence of steps:

1. Construct a separate data set for each region. This data set includes the geographic location, site type, and all potential predictor variables. The locations that go into this data set are all locations with known cultural resources, all locations that are known to have been surveyed for cultural resources, and a set of randomly chosen locations from within the region.
2. Export this data set from ArcGIS (ESRI, Inc.) and import it into S-Plus statistical software (Insightful Corp.).
3. Fit a predictive model using these data within S-Plus.
4. Implement in ArcGIS the predictive model determined in S-Plus, producing digital maps for the entire region.

Mn/Model is now entering Phase 4. This phase will bring in additional predictive variables, clean up or provide more accurate versions of current predictive variables, use a spatially corrected database of archaeological sites, and make use of these new data collected since Phase

3 to produce new predictive models. Rather than simply use the prediction schemes previously used in Phase 3, Phase 4 will evaluate other prediction methods for building the predictive maps.

The "Statistics for Mn/Model" project being reported here is the branch of the Phase 4 effort that seeks to evaluate alternate statistical methods for archaeological predictive models. Our principal objectives are:

Objective 1. Find the best prediction method for Mn/Model Phase 4 that can be implemented reasonably within GIS.

Objective 2. Produce S-Plus software (and accompanying documentation) to implement this prediction method in the context of Mn/Model Phase 4.

Secondary objectives for this project include:

Objective 3. Provide Mn/Model professionals with guidance on ways to compare and evaluate prediction methods.

Objective 4. Train Mn/DOT professionals on the new prediction method.

This report address objectives 1 and 3. A separate document, the "User's Guide for Mn/Model Phase 4 S-Plus Software," addresses objective 2. Training will be done in person.

Chapter 2 of this report discusses methods and guidance for comparing prediction methods in the Mn/Model context. These methods include the use of cross-validation for true and false positive rates, as well as graphical methods such as predictive distributions, ROC curves, and "gain" curves.

Chapter 3 discusses potential prediction methods. This project proposed and compared eight prediction methods: logistic regression with BIC model selection (the Phase 3 method), logistic regression with Bayesian model averaging, naive Bayes classification, tree-structured regression, "bumped" trees, "bagged" trees, "double bagged" trees, and "boosted" trees. Bumping, bagging, and boosting are examples of "perturb and aggregate methods," which repeatedly modify the data in minor ways and then combine the predictions from the modified data sets.

Chapter 4 compares the prediction methods on sample Mn/Model data and concludes that bagged trees (bagging) should be the prediction method for Phase 4. Overall, bagging and boosting had the best predictive ability, and bagging is easier to implement in GIS as well as being easier to explain to the public.

Chapter 5 runs through an example with the S-Plus software and explains the output.

Chapter 6 gives conclusions.

2. Model Evaluation

This chapter presents methods of evaluating possible classifiers for use in Mn/Model. It is broken into two sections:

1. Comparing Classifiers, and
2. Cross-Validation.

The sections correspond to two basic questions: what are the properties or characteristics that tell us which classifiers are better, and what is an honest estimate of those properties? Most of the language and ideas of this chapter were originally developed in the context of medical diagnostic tests (Pepe 2003), but they are applicable to any two-category classification problem.

2.1 Comparing Classifiers

This section presents methods of comparing possible two-category classification tools. For example, we would like to predict whether or not a portion of landscape contains an archaeological site. Let's say that we have two competing methods of prediction (rules). Every location is either a site or a non-site, and each rule classifies every location as either a site or a non-site. Thus we can form a 2 by 2 table listing the fractions of locations that fall into the four possibilities for how the rule classified them and their actual site/non-site status, as in Table 2.1.

Rule 1

Truth	Site	Non-Site	Total
Site	0.08	0.02	0.10
Non-Site	0.30	0.60	0.90
Total	0.38	0.62	1.00

Rule 2

Truth	Site	Non-Site	Total
Site	0.06	0.04	0.10
Non-Site	0.20	0.70	0.90
Total	0.26	0.74	1.00

Table 2.1. Two competing classification rules. Rows denote true classes, columns denote classes as determined by the rules.

The true positive rate (TPR), or sensitivity, of the method is the fraction of sites that are correctly predicted to be sites. In probability terms, the sensitivity is the probability that the rule classifies a location as a site, given that the location actually is a site.

$$\text{Sensitivity}_1 = \Pr(\text{Rule 1} = \text{Site} \mid \text{Truth} = \text{Site}) = 0.08 / 0.10 = 0.80$$

$$\text{Sensitivity}_2 = \Pr(\text{Rule 2} = \text{Site} \mid \text{Truth} = \text{Site}) = 0.06 / 0.10 = 0.60$$

In our example, Rule 1 correctly labels 80% of the true sites while Rule 2 correctly labels only 60% of the true sites. That is, Rule 1 is more sensitive to the presence of an archaeological site.

The specificity of the method is the fraction of non-sites that are correctly classified. In probability terms, specificity is the probability that the rule classifies a location as a non-site, given that the location actually is a non-site.

$$\text{Specificity}_1 = \Pr(\text{Rule 1} = \text{Non-Site} \mid \text{Truth} = \text{Non-Site}) = 0.60 / 0.90 \approx 0.67$$

$$\text{Specificity}_2 = \Pr(\text{Rule 2} = \text{Non-Site} \mid \text{Truth} = \text{Non-Site}) = 0.70 / 0.90 \approx 0.78$$

In our example, Rule 1 correctly labels about 67% of non-sites while Rule 2 correctly labels about 78%. That is, Rule 2 is more specific about what locations it labels as sites. A related measurement is the false positive rate (FPR): the fraction of non-sites that are incorrectly categorized as sites. In fact,

$$\text{false positive rate} = 1 - \text{specificity}.$$

A rule that is more specific about what it labels as a site will have fewer false positives.

Ideally, both the sensitivity and the specificity of a method are high. If Rule 1 were both more sensitive and more specific than Rule 2, we would clearly prefer Rule 1. However, when one of the predictors is more sensitive and the other is more specific, we have to decide which errors are more costly before we can choose between the rules. For example, if the costs associated with happening upon an archaeological site at a location that was predicted as a non-site are much greater than the costs of unnecessarily surveying a location or unnecessarily avoiding portions of the landscape (avoiding predicted sites that were actually non-sites), then we will prefer the more sensitive method.

Some predictive methods have a "tuning parameter" that adjusts their sensitivity and specificity. For example, in the case of logistic regression, portions of landscape are ordered according to their estimated likelihood of containing a site. Choosing a point in this continuum as a threshold creates prediction categories: locations above the threshold are labeled sites, the rest are labeled non-sites. Changing the threshold changes the sensitivity and specificity, and examining the sensitivity and specificity provides guidance in choosing the threshold.

Unless we are able to perfectly predict whether a location is a site, there is a necessary trade between sensitivity and specificity when choosing a threshold. The extremes are illustrative. An extremely low threshold (negative infinity, say) would have sensitivity 1, because every site is correctly labeled a site, but this is because every location in the state would be labeled a site. Thus, the sensitivity is 1, but the specificity is 0. A rule based on this threshold would be entirely too sensitive; everything sets it off. Conversely, an extremely high threshold (positive infinity) would have specificity 1 because it correctly labels every non-site a non-site, but it also labels every site a non-site (sensitivity 0). Such a rule is far too specific about what it is willing to label a site.

The trade-off between sensitivity and specificity in assigning a threshold is illustrated by the Receiver Operating Characteristic (ROC) curve. An ROC curve plots the true positive rate (sensitivity) against the false positive rate (1-specificity) as the threshold value varies. Figure 2.1 is a typical ROC curve. The curve passes through the points (0,0) and (1,1). These points correspond to the extreme cases where no location is labeled a site and where every location is labeled a site. The diagonal line connecting these points corresponds to randomly guessing whether a portion of landscape contains an archaeological site; you get the point (p,p) on the line by randomly assigning a fraction p of the locations to be sites. An ROC curve above the diagonal implies that the rule (prediction model) does better than randomly guessing.

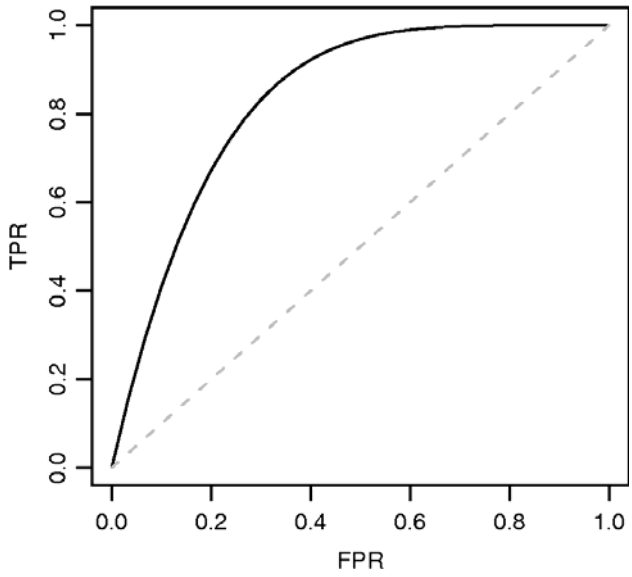


Figure 2.1: An ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) as the threshold level varies.

We may compare two models (Model 1 and Model 2) by comparing their ROC curves. If the ROC curve of Model 1 is uniformly greater than or equal to the ROC curve of Model 2, then Model 1 is more sensitive than Model 2 at every specificity. (It is also more specific at every sensitivity level.) In that case, Model 1 would clearly be preferable to Model 2.

It may happen that ROC curves intersect, as is shown in Figure 2.2. In that case, one rule is preferable when high specificity is needed, and the other rule is preferable when high sensitivity is needed. For example, suppose that the costs associated with happening upon an archaeological site in a location predicted to be a non-site are much greater than the costs of unnecessarily avoiding portions of the landscape. Then we may fix the sensitivity at a high enough level to control against the costs of falsely labeling sites as non-sites. Once the sensitivity has been fixed, we choose the rule with the best specificity. In the example shown in Figure 2.2, if we require our rule to capture at least 80% of the true sites, then Model 1 is preferably, because it has a higher specificity (lower false positive rate) whenever sensitivity is at least 80%. If we only require 50% of the true sites to be labeled as sites, then the preferable model is Model 2. In this case, our choice of model depends on the percent of true sites that we are willing to mislabel. Thus, in addition to deciding what error is more important to guard against, we generally need to determine the rate at which we are willing to incur that error.

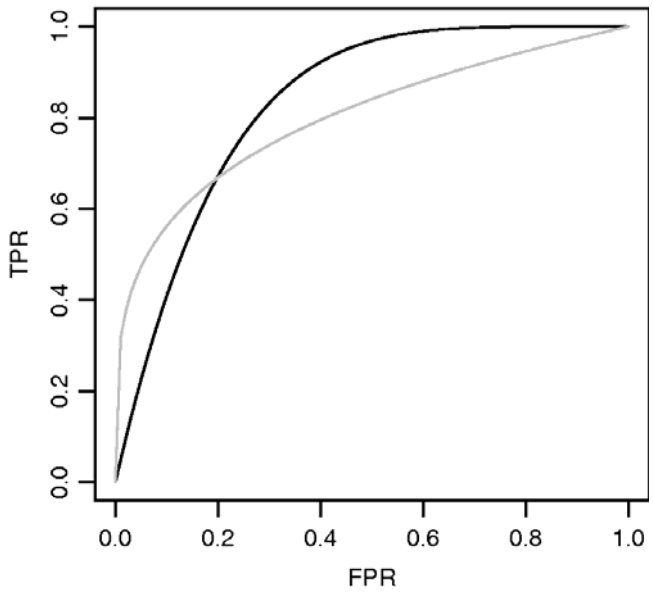


Figure 2.2: An example where the ROC curves for Model 1 (black) and Model 2 (gray) intersect.

Fixing sensitivity, increased specificity corresponds to a smaller fraction of landscape labeled as sites. For example, the two rules in Table 2.2 have the same sensitivity, but Rule 2 is more specific about what locations it labels sites. To capture 70% of the actual sites, Rule 1 has to label 37% of the landscape as sites. To capture the same percentage of actual sites, Rule 2 labels only 27% of the landscape as sites.

Rule 1

Truth	Site	Non-Site	Total
Site	0.07	0.03	0.10
Non-Site	0.30	0.60	0.90
Total	0.37	0.63	1.00

Rule 2

Truth	Site	Non-Site	Total
Site	0.07	0.03	0.10
Non-Site	0.20	0.70	0.90
Total	0.27	0.73	1.00

Table 2.2: An example where sensitivity is the same for both rules. The higher specificity of Rule 2 (0.78 versus 0.67) results in a smaller fraction of the landscape labeled a site (27% versus 37%).

Sensitivity and specificity tell us about a classification rule's performance for sites and non-sites respectively. When the rule is actually put into practice, the truth is unknown, and all we have is the prediction made by our classification rule. Thus, we would like to know how likely we are to encounter a site when the classification rule says that a site will be present, and also when the rule says that a site will not be present.

The positive predictive value (PPV) is the fraction of locations that our rule classifies as sites that actually contain sites. In probability terms, PPV is the probability that a location is a site, given that the rule classifies the location as a site:

$$\text{Positive Predictive Value} = \Pr(\text{Truth} = \text{Site} \mid \text{Label} = \text{Site}).$$

The negative predictive value (NPV) is the fraction of locations that our rule classifies as non-sites that actually are non-sites. In probability terms, NPV is the probability that a location is a non-site, given that the rule classifies the location as a non-site:

$$\text{Negative Predictive Value} = \Pr(\text{Truth} = \text{Non-Site} \mid \text{Label} = \text{Non-Site}).$$

We define a new term, the Unexpected Discovery Rate (UDR), to be the probability of a location being a site, given that the prediction rule classified the location as a non-site:

$$\text{UDR} = \Pr(\text{Truth} = \text{Site} \mid \text{Label} = \text{Non-Site}) = 1 - \text{NPV}.$$

The UDR is the "Oops rate," the fraction of times that we find an archaeological site where we don't expect it.

The PPV, NPV, and UDR depend not only on the sensitivity and specificity but also on the prevalence, which is the fraction of the landscape that contains sites. In probability terms, the prevalence is the probability that a location is a site:

$$\text{Prevalence} = \Pr(\text{Truth} = \text{Site}).$$

One way to think of the prevalence is as the chances of finding a site by choosing a random location. Prevalence is also called the *a priori* probability of a site.

We would like to compare the PPV and NPV to the prevalence to see how much better we do using our prediction rule instead of simply guessing. We define the positive predictive gain (PPG) as the ratio

$$\begin{aligned}\text{PPG} &= \text{Positive Predictive Value} / \text{Prevalence} \\ &= \Pr(\text{Truth=Site} \mid \text{Label=Site}) / \Pr(\text{Truth=Site}) .\end{aligned}$$

The positive predictive gain tells us how many times better the model is at discovering sites than a random survey would be. For example, in Table 2.2, the probability that a random location contains an archaeological site is 0.10. Under Rule 1, the probability that a location labeled a site is in fact a site is about 0.19,

$$\text{PPV}_1 = \Pr(\text{Truth} = \text{Site} \mid \text{Rule 1} = \text{Site}) = 0.07 / 0.37 \approx 0.19 .$$

Thus we find

$$\text{PPG}_1 = 0.19 / 0.10 = 1.9,$$

meaning that locations labeled sites using Rule 1 are about 1.9 times as likely to be sites as a randomly chosen location. Under Rule 2, the PPV is about 0.26,

$$PPV_2 = Pr(Truth = Site|Rule\ 2 = Site) = 0.07 / 0.27 \approx 0.26$$

making the PPG about 2.6,

$$PPG_2 = 0.26 / 0.10 = 2.6\ .$$

A location labeled a site with Rule 2 is 2.6 times as likely as a random location to be an archaeological site.

We also want to compare a rule's ability to avoid sites to the background probability of a site. We define the Negative Predictive Gain (NPG) as the ratio

$$NPG = UDR / Prevalence = Pr(Truth=Site \mid Label=Non\text{-}Site) / Pr(Truth=Site)\ .$$

The negative predictive gain tells us how much less likely we are to discover a site at a location labeled a non-site using the model than if we were surveying randomly. Returning again to Table 2.2, Rule 1 has an unexpected discovery rate of about 0.05,

$$UDR_1 = Pr(Truth = Site \mid Rule\ 1 = Non\text{-}Site) = 0.03 / 0.63 \approx 0.05,$$

which is one half the chances of finding a site at random:

$$NPG_1 = Unexpected\ Discovery\ Rate / Prevalence \approx 0.05 / 0.10 = 0.50.$$

Since Rule 2 has an unexpected discovery rate of about

$$UDR_2 = Pr(Truth = Site \mid Rule\ 2 = Non\text{-}Site) = 0.03 / 0.73 \approx 0.04,$$

its negative predictive gain is about

$$NPG_2 = Unexpected\ Discovery\ Rate / Prevalence \approx 0.04 / 0.10 = 0.40.$$

That is, the chances of unintentionally discovering an archaeological site using Rule 2 are about 40% of what they would be surveying at random.

A good predictive rule will have a large PPG and a small NPG.

Given a prevalence, we can compute the positive predictive gain and the negative predictive gain for every point on an ROC curve. Each different point (rule) on the ROC curve can be identified by its true positive rate (vertical coordinate) or by its false positive rate (horizontal coordinate). We can thus examine positive (or negative) predictive gain by plotting it against either the TPR or the FPR; in fact, it is sometimes useful to do both.

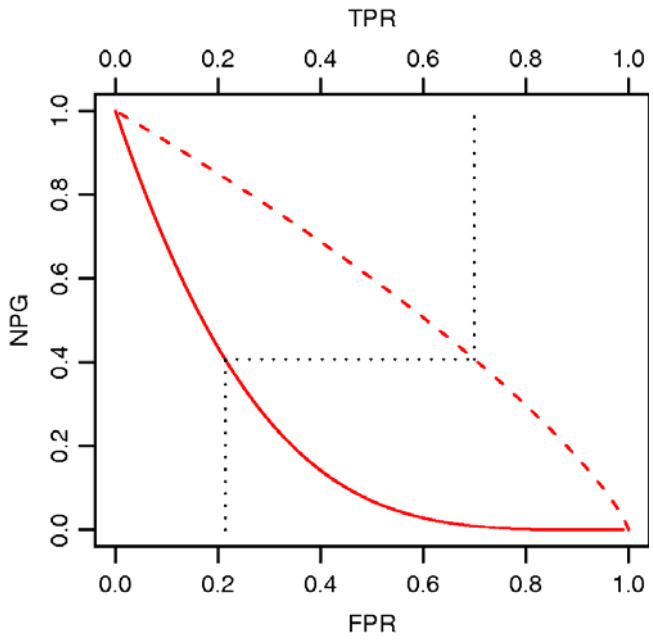


Figure 2.3: Comparison of the negative predictive gain with the false positive rate (solid line) and true positive rate (dashed line).

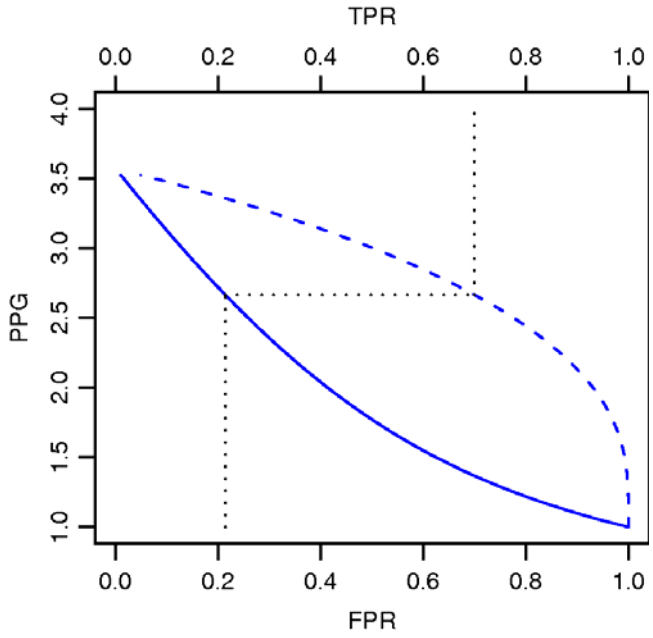
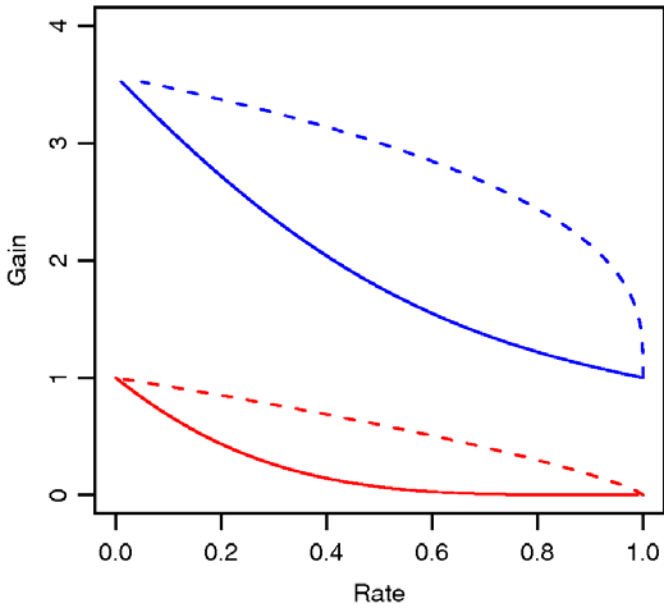


Figure 2.4: Comparison of the positive predictive gain with the false positive rate (solid line) and true positive rate (dashed line).

Figure 2.3 plots the negative predictive gain against both the true and false positive rates; we have assumed the ROC curve of Figure 2.1 and 10% prevalence. The dashed red line plots the NPG against the true positive rate (top axis). For example, a classification rule that correctly labels 70% of the archaeological sites reduces our chances of unexpected discovery to about 40%

of what it would be with random survey (a reduction of 60%). The solid red line plots the NPG against the false positive rate (bottom axis). Continuing the example, the same rule that reduces our chances of accidental discovery by around 60% has a false positive rate of about 20% (it mislabels about 20% of the non-site landscape as a site).

Figure 2.4 compares the positive predictive gain to the true and false positive rates; we again assume the ROC curve of Figure 2.1 and 10% prevalence. The dashed blue line plots the positive predictive gain against the true positive rate (top axis). The classification rule that correctly labels 70% of the archaeological sites is also about 2.7 times as efficient identifying sites as a random guess would be. The solid blue line plots the positive predictive gain against the false positive rate (bottom axis). Continuing the example, the same rule that gives a 2.7 times gain in efficiency also mislabels about 20% of the non-site landscape as a site.



Line	Comparison
Dashed blue	PPG vs. TPR
Solid blue	PPG vs. FPR
Dashed red	NPG vs. TPR
Solid red	NPG vs. FPR

Figure 2.5. Sample gain plot, showing both positive (blue) and negative (red) predictive gains against the true positive rate (dashed) and the false positive rate (solid).

Gain curves like Figure 2.5 combine the plots in Figures 2.3 and 2.4 into a single graphic. This plot uses a single horizontal axis, but interprets it as FPR for solid lines and TPR for dashed lines. The vertical axis is the gain (either positive or negative).

2.2 Cross-Validation

The goal of our analysis is not to classify the observed locations—we know whether they are sites. Instead, we want to predict where archaeological sites remain uncovered. We create a classification rule based on the sample we have observed and use the information in that sample to predict whether a new location is a site.

Our prediction rules are developed with our data and for our data. This means that they likely work better with this data set than any other. In particular, the sensitivity and specificity we get by simply applying our rules to our data give us an overly optimistic picture of how well the rules will work when applied on new data. Cross-validation directly estimates the prediction error of the classifier on new data; this is called the out-of-sample prediction error of a classifier. Chapters 2 and 8 of Mosteller and Tukey (1977) provide a gentle introduction to the motivation and use of cross-validation.

2.2.1 Standard Cross-Validation

Every real prediction is about a location that was not used in development of the prediction rule. Nonetheless, we need to estimate how well our rules will work on these future locations based on what we know from our current data. Cross-validation simulates this "new location" prediction by setting some of the sample aside to see how well we are able to predict using the remainder of the sample.

	Run 1	Run 2	Run 3	…	Run 10		CV Predicted
Subset 1	Predict	Fit	Fit	…	Fit	→	From run 1
Subset 2	Fit	Predict	Fit	…	Fit	→	From run 2
Subset 3	Fit	Fit	Predict	…	Fit	→	From run 3
…	…	…	…	…	…	→	…
Subset 10	Fit	Fit	Fit	…	Predict	→	From run 10

Figure 2.6: Schematic of subset use and origin of predicted values in a ten-fold cross-validation.

Figure 2.6 illustrates the case of a ten-fold cross-validation. A model is specified up to a set of parameters to be estimated. The data are randomly divided into ten subsets, and we will eventually produce predictions for data in all of the sub sets. To predict for subset 1, we fit the model using subsets 2 through 10 (Run 1). Then we use this model, which was fit without subset 1, to predict subset 1. These predicted values are then carried over to be the cross-validation predicted values for subset 1. To predict subset 2, we fit the model using subsets 1 and 3 through 10 (Run 2), and so on. We leave out each subset once until an out-of-sample prediction has been made for every observation.

Phase 3 of Mn/Model essentially used two-fold cross-validation to evaluate models. For Phase 4, we propose to evaluate the true and false positive rates of our prediction rules using ten-fold cross-validation. We will divide the data into 10 groups at random and do 10 runs, leaving out and predicting one of the groups in each run. The out-of-sample predictions and the truth can be used to estimate the prediction error of the classifier. We call these cross-validated estimates.

Any of the measures outlined in Section 2.1 can be applied to the out-of-sample predictions. Sensitivity, specificity, ROC curves, and gain plots can all be calculated and in general should be calculated. On their own, they provide estimates of the classifier's abilities as a prediction tool. When compared with the analogous measures based on the full sample, they indicate the

sensitivity of the estimation procedure to the data observed.

2.2.2 Spatial Cross-Validation

One potential problem with ordinary cross-validation is that it treats every location in the sample as independent and exchangeable. That is, it assumes that leaving any subset of 10% out is as good as any other. However, we have spatial data, and the first law of geography states that everything is related, but nearby things are more related. This would imply that 10% of the data in a spatial cluster may behave differently than 10% randomly scattered.

To assess the effect of this spatial correlation, we also try a spatial cross-validation. Standard cross-validation sets aside observations randomly to be predicted by those remaining; spatial cross-validation groups data into spatial clusters and then sets aside spatial clusters to be predicted by those remaining. That is, spatial cross-validation estimates the out-of-spatial-cluster prediction error of a classifier.

This is, in general, a much more challenging prediction problem, because we have guaranteed that there will be no nearby (and thus presumably similar) locations in our training data. Spatial cross-validation simulates prediction into an area "unlike" those we have seen before.

In some situations, Mn/Model is used to make predictions into landscapes that have not been surveyed before but are similar to landscapes that have been surveyed before. For those situations, ordinary cross-validation is likely to give the better estimate of prediction accuracy. In other situations, Mn/Model is predicting into landscapes unlike any that have ever been surveyed. In those situations, spatial cross-validation is likely to give the better estimate of prediction accuracy. Finally, spatial cross-validation can be useful in preventing over-optimism about models in regions where there are low site numbers and only certain kinds of landscapes (e.g. lakeshores) have been surveyed. Models in these situations will appear to be quite accurate under cross-validation, but are likely quite poor under spatial cross-validation.

Implement ten-fold spatial cross-validation as follows:

1. Divide the data into K mutually exclusive and exhaustive spatial clusters of locations (sites, surveyed locations, and random locations). By default, we choose K to be 400.
2. Randomly choose 10% of the regional clusters of locations to hold out; predict the locations in those clusters based on a model fitted to the other 90% of the regional clusters.
3. Repeat step 2 nine times until all spatial clusters have been held out and predicted once.
4. Use the spatially cross-validated predictions to compute criteria such as sensitivity, specificity, and so on.

Several questions arise when determining how to construct spatial clusters: Should clusters have an equal number of locations (data points)? Should clusters cover similar areas? If a location is off by itself, should we force it into a cluster with other locations? How many clusters are appropriate? How can we automate this process?

We make the following proposal for automated construction of spatial clusters:

1. Clustering will be based on the UTM easting and northing coordinates of each raster cell in the sample.
2. Clustering will be done via the k-means procedure to produce a vector with elements 1 through 400 specifying the cluster memberships.
3. Ten-fold spatial cross-validation will be done by holding back randomly chosen sets of 10 clusters, until all clusters have been predicted.

Spatial cross-validation necessarily involves a choice on the part of the investigator in defining the spatial clusters. In particular, the above suggestion of 400 clusters is arbitrary. The behavior of spatially cross-validated estimates is highly dependent on the number of spatial clusters defined. On one extreme, the investigator may define as many clusters as there are locations, which is simply standard cross-validation. On the other extreme, one could choose to predict the western half of the locations using only the eastern half of locations. Somewhere between these extremes is an appropriate level of clustering to capture the effect of predicting into portions of landscape with which there is little or no experience.

The k-means procedure works as follows (Hartigan 1975). First we initialize K cluster centers (K is 400 by default); these could be K random points in the region of interest. Then each data location is assigned to the closest cluster center. Then the cluster center centers are moved to positions that minimize a measure of the total distance from all locations to the cluster centers. Then the locations are reassigned to centers, the centers are updated, and this is repeated until no further improvements are possible. If all the locations were uniformly spread over the area of interest, the resulting clusters would be approximately equal in number of locations and area. However, in areas of high density of locations, the clusters will tend to have more locations and smaller areas.

This method to construct spatial clusters is simple to implement and not completely unrealistic, but it may be possible to improve it substantially by instead using expert knowledge to derive the clusters. For example, travel time by foot or by canoe, size of archaeological sites, foraging distances, change of elevation, and other factors may provide better clusters. However, that expert opinion cannot (easily) be automated.

We recommend spatial cross-validation be used as a means of feeling out the possible consequences of extrapolation into portions of landscape where little is known about the presence or absence of sites.

2.3 Summary

Because the costs associated with accidentally discovering an archaeological site are considered large compared to the costs of avoiding some locations unnecessarily, we recommend fixing a minimum sensitivity, as was done in Mn/Model Phase 3. Among the prediction rules attaining that sensitivity level, choose the rule that is most specific. Here we recommend choosing rules that maximize specificity for 85% sensitivity.

Because sensitivity and specificity based on cross-validation directly estimate the predictive

performance of the rule, we recommend that models be compared based on their cross-validated sensitivity and specificity. Similarly, thresholds for classifying locations should come from cross-validation. Note, however, that the actual implemented model should be estimated on all of the data.

3. Classification Methods

This chapter presents several methods to classify locations as having sites or not. The methods fall into three distinct groups: those based on logistic regression, those based on tree-structured regression, and those based on Bayes' rule. There are four sections, three describing the three groups of methods, and a fourth comparing the advantages and disadvantages of the methods. Hastie et al. (2001) is a good general reference for the methods described in this chapter, but it assumes the reader has some comfort with statistical methodology.

3.1 Logistic Regression Methods

Logistic regression methods are based on the assumption that all responses are either 1 (a site) or 0 (a non-site). The probability that a particular location is a site is p(x), where x represents a set of predictor variables such as elevation, aspect, distance to water, and so on. Similarly, the probability that a location is not a site is 1-p(x). If we have k different predictors, we represent these as x_1, x_2, ..., x_k. Since each different location could have different values for the predictors, locations can differ in their probabilities of being sites.

Logistic regression assumes that the relationship between the predictors x and the probability p(x) takes a particular form. The logit transform of the probability p(x) is ln[p(x)/(1-p(x))]; in this expression, ln() indicates the natural logarithm. Logistic regression assumes that the logit of p(x) is a linear combination of the predictor variables:

$$\ln[p(x)/(1-p(x))] = L(x) = a_0 + a_1 x_1 + a_2 x_2 + ... + a_k x_k$$

Here x_1, x_2, ..., x_k are the known predictor values for the location, and a_0, a_1, ..., a_k are unknown coefficients that we will need to estimate. We can call L(x) the logit of the probability or the linear predictor (the sum of products on the right is called a linear combination, hence linear predictor).

Logistic regression takes a set of locations, each with its own 0/1 response and its own set of k predictor values and estimates the values of a_0 through a_k. Then, for a new location with a new set of predictors, we can compute the linear predictor L(x) based on the coefficients we estimated from the training data. We can compute the probability of the location being a site from the linear predictor via

$$p(x) = \exp(L(x))/[1 + \exp(L(x))] .$$

Classification is done by picking a threshold and declaring locations with p(x) greater than the threshold to be sites.

3.1.1 Variable selection

One of the issues with using logistic regression is deciding which set of predictors to use. If we leave out an important predictor, then our estimates of the logit will be biased. If we use extra, unneeded predictors, then our estimates of the logit will have excess variability. Both bias and

excess variability are bad, so we want to reduce them. The trick is to find a set of predictors that is large enough to include all the important variables without including unneeded variables. This is the variable selection problem. Within a given data set, more predictor variables will always fit the data better than fewer predictor variables, but using more predictor variables may make the predictions worse when the procedure is applied to new data. We need to figure out what will work best on new data while using only the data at hand.

Several different variable selection techniques have been proposed. In this project, we will use the Bayesian Information Criterion, also known as BIC (Schwarz, 1973). BIC works by combining a measure of discrepancy between the observed data and the model predictions with a penalty based on the number of predictors and the amount of data. Using more predictors will decrease the discrepancy, but it increases the penalty. What we do is find the subset of predictor variables that minimizes the BIC.

Logistic regression with BIC model selection was the prediction method used in Mn/Model Phase 3.

BIC has a theoretical property that makes its use attractive, although it is not obvious that this property is really active in any actual data set. The property is this: if the "true" model is present as a subset of the variables we are considering, then given enough data, BIC will always choose the correct model. However, we're never sure that the true model is one of our possible models, and we're never sure that we have enough data.

3.1.2 Model Averaging

There may be many models (subsets of variables) with approximately equal values of BIC, that is, there may be many models that fit the data about equally well. In such a case, it is not obvious that the model with the smallest BIC is really any better than a model with a slightly larger BIC. It may be that different models have different virtues that can be exploited by combining them.

Since some models are better than others, we want to acknowledge that when combining various models. BIC can be used to approximate a posterior probability for each model; these posterior probabilities tell us how likely each model is to be the "true" model. The model coefficients are then averaged according to their approximate posterior probabilities: the best models contribute the most to the averaged models. In general, averaged models yield more stable predictions than single selected models.

3.2 Naive Bayes

A premise of the Mn/Model project is that sites and non-sites tend to be at locations with different landscape features. Ultimately, we want to use those differences to predict whether a location contains an archaeological site. Logistic regression yields predictions by estimating the probability that a location is a site given some predictors. Another approach to estimating the probability is to reverse it: estimate the distribution of predictors given that a location is or is not a site. This reversal hinges on a property of conditional probability called Bayes rule.

Bayes rule is written abstractly based on the probability of two events, or conditions, called A and B. To make things concrete, let event A occur when a specific location is a site, and let event B occur when the predictors for the same specific location take a given set of values. We will want an expression for the probability of A given B, that is, the probability that the location is a site given that the location has a certain set of predictors.

The probability Pr(A and B) that both conditions A and B hold is equal to the probability that one of those conditions holds, Pr(A), times the probability of the second condition given the first, Pr(B | A),

$$\Pr(\text{ A and B }) = \Pr(\text{ B } | \text{ A }) \Pr(\text{ A }) .$$

That is, the probability of A and B equals the probability of B given A times the probability of A. Similarly, the probability of A and B equals the probability of A given B times the probability of B:

$$\Pr(\text{ A and B }) = \Pr(\text{ A } | \text{ B }) \Pr(\text{ B }) .$$

Since both of the products—Pr(B | A) Pr(A) and Pr(A | B) Pr(B)—are equal to the joint probability, Pr(A and B), they are equal to one another

$$\Pr(\text{ A } | \text{ B }) \Pr(\text{ B }) = \Pr(\text{ B } | \text{ A }) \Pr(\text{ A }) .$$

Dividing through by Pr(B) gives Bayes rule:

$$\Pr(\text{ A } | \text{ B }) = \Pr(\text{ B } | \text{ A }) \Pr(\text{ A }) / \Pr(\text{ B }) .$$

Bayes rule says that we can express the conditional probability of A given B in terms of the probability of A, the probability of B, and the reversed conditional probability—the probability of B given A. For example, the probability that a location is an archaeological site given that it has some specified landscape features can be rewritten as the probability that the location has the features given that it is a site, scaled by the probabilities of having the features and being a site:

$$\Pr(\text{ Site } | \text{ Feat. }) = \Pr(\text{ Feat. } | \text{ Site }) \Pr(\text{ Site }) / \Pr(\text{ Feat. }) .$$

So, one way to approach estimating the probability of a site given some landscape features is by estimating the three probabilities on the right hand side—Pr(Site), Pr(Feat.), and Pr(Feat. | Site).

An alternative is to use the odds of being a site rather than the probability itself:

$$Odds = \frac{\Pr(Site \mid Feat.)}{\Pr(Nonsite \mid Feat.)} = \frac{\Pr(Feat. \mid Site)\Pr(Site)}{\Pr(Feat. \mid Nonsite)\Pr(Nonsite)}$$

Naïve Bayes makes the simplifying assumption that the distributions of different landscape

features are independent of each other (given site or non-site). That is, for landscape features x_1, x_2, …, x_k, assume that

$$\Pr(x_1,x_2,\ldots,x_k \mid \text{Site}) = \Pr(x_1 \mid \text{Site}) \Pr(x_2 \mid \text{Site}) \ldots \Pr(x_k \mid \text{Site})$$

and

$$\Pr(x_1,x_2,\ldots,x_k \mid \text{Non-site}) = \Pr(x_1 \mid \text{Non-site}) \Pr(x_2 \mid \text{Non-site}) \ldots \Pr(x_k \mid \text{Non-site})$$

Then, the odds are

$$Odds = \frac{\Pr(x_1 \mid Site)}{\Pr(x_1 \mid Nonsite)} \frac{\Pr(x_2 \mid Site)}{\Pr(x_2 \mid Nonsite)} \ldots \frac{\Pr(x_k \mid Site)}{\Pr(x_k \mid Nonsite)} \frac{\Pr(Site)}{\Pr(Nonsite)}$$

This method is "naive" because the simplifying independence assumption is almost surely wrong. Nevertheless, the estimator often performs quite well (see Hand and Yu (2001) for examples and possible explanations).

Estimating Pr(Site) is not really an issue. We either use a prior probability or the sample proportion of sites. The tricky part is how to estimate the conditional probabilities Pr(x_1 | Site) and Pr(x_1 | Non-Site). We do this using the data in the training sample.

We use a kernel density estimator to approximate the conditional probabilities Pr(x_1| Site) and Pr(x_1 | Non-Site). A kernel density estimate is like a smoother version of a histogram. It has all the flexibility of a histogram, but gives a finer estimate of the distribution.

Naïve Bayes can be related back to logistic regression. The logit of a probability is the log of the odds. Logistic regression assumes that this logit is a constant plus a sum of terms of the form $a_i x_i$ --- that is, each predictor enters as a linear function. If we take the log of the odds for the naïve Bayes estimate, we get a constant plus a sum of terms of the form $h_i(x_i)$, where

$$h_i(x_i) = \ln[\ \Pr(x_i \mid \text{Site})\ ] - \ln[\ \Pr(x_i \mid \text{Non-site})\ ]$$

Naive Bayes makes weaker assumptions than logistic regression about how the log odds of a site depend on the location's landscape features. However, we pay for the added flexibility with a loss of efficiency: it is harder to estimate a density function than it is to estimate a constant. If the logistic assumptions are correct, logistic regression is more efficient than naïve Bayes. If naive Bayes improves on logistic regression, it is evidence that we should consider a more flexible relationship between the predictors and the odds that there is a site.

3.3 Tree-Based Methods

Tree methods make predictions by repeatedly splitting the data into smaller and smaller subsets and then predicting based on the average response in each of the final subsets. The splitting method is called recursive partitioning. First, the data set is split into two groups. Then each of these groups is split into two groups, and so on. Each split is based on a single predictor variable and a split point, with data having predictor values less than the split point placed in one

subgroup and all other data placed in the other subgroup. The splits are done to make the responses in each subgroup as homogeneous as possible. The idea is simple, and the heart of the algorithm is deciding what size tree to use (that is, how many splits to make). Breiman et al (1984) is a standard monograph on trees within Statistics.

The most straightforward way to make sense of trees is to see an example, as in Figure 3.1.

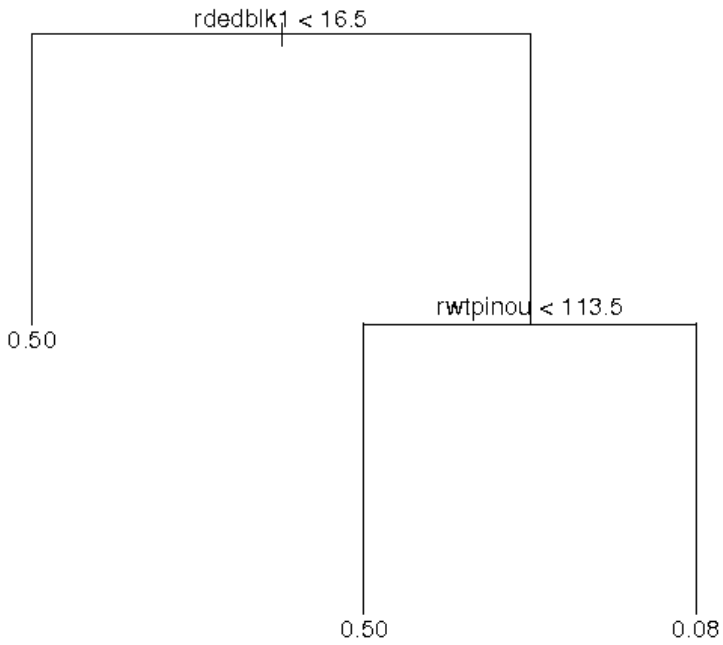


Figure 3.1: A simple regression tree.

In this example, the first split is made in the variable rdedblk1 (distance to the nearest large lake [square root]). The data are divided into locations such that rdedblk1 is less than 16.5 and locations such that rdedblk1 is greater than or equal to 16.5. The second split is made within the group of locations that have rdedblk1 greater than or equal to 16.5. This time the split is made in the variable rwtpinou (distance to nearest permanent wetland inlet/outlet [square root]). These data are divided into locations such that rwtpinou is less than 113.5 and locations such that rwtpinou is greater than or equal to 113.5. Each split is binary (a measurement is either below a threshold or not), and each split is made within a single variable (first rdedblk1, then rwtpinou).

The tree has three terminal nodes. All of the observations in a terminal node have the same estimated probability. In this example, half of the locations with rdedblk1 less than 16.5 are sites, so we estimate a response of 0.50 for all locations with rdedblk1 less that 16.5. Similarly, half of the locations with rdedblk1 greater than or equal to 16.5 and rwtpinou less than 113.5 are also sites, so the estimated response is 0.5. On the other hand, only 4 out of 50 locations with

rdedblk1 greater than or equal to 16.5 and rwtpinou greater than or equal to 113.5 are sites, so this subset has an estimated response of 0.08.

With only three terminal nodes and only two estimated probabilities, this is an unusually stumpy tree. However, even a much larger tree will assign only a few distinct probability estimates when compared, for example, to logistic regression. Because all of the locations in a node have the same estimated probability, trees make very coarse probability estimates.

The compensation for making very coarse estimates comes in the form of versatility. Trees do not assume any particular model for the data; they simply divide the sample into dichotomies that seem promising.

As with the other procedures discussed here, trees have a Goldilocks problem: large trees overfit the data (leading to excess variability in the estimates), small trees under fit the data (missing important features and causing bias), and we want a tree that's just right. Thus, tree size is a tuning parameter. Trees grow until a minimum node size is reached. In our application, all nodes of the tree (subsets of data) must contain at least 20 locations. We then prune the tree: cut back branches that appear to overfit the data. Pruning uses an additional ten-fold cross-validation separate from that we do to assess true and false positive rates.

Trees are grown one split at a time. Branches are formed by optimizing individual splits. This is a "greedy" algorithm that tries to get as much as it can right now by optimizing this split while ignoring all subsequent splits. This algorithm does not necessarily optimize the whole tree. Because the partitioning is recursive, each split is contingent on the previous partitions. Slight variations in the data may lead to noticeably different trees. In order to deal with these issues, it is common to grow a forest of trees and then combine them in some fashion.

3.4 Perturb and aggregate

Trees are flexible and effective predictors, but they suffer from two problems: there are relatively few distinct predicted values (coarseness of predicted values), and the tree produced is very sensitive to minor features of the data. Researchers have proposed several modifications to the basic tree approach that attempt to solve these problems. All of these modification work by repeatedly making slight perturbations to the data, producing a tree for each perturbed data set, and then combining the trees so produced. The names of these procedures are rather whimsical: bagging, boosting, bumping. Methods of this type are called perturb and aggregate methods, and the collection of trees produced is sometimes called a forest.

The principal method to perturb the data is the bootstrap (Efron and Tibshirani 1993). Consider a data set of size 1000. A bootstrap sample is a random sample of size 1000 taken from the original data set *with replacement*. Each of the original data observations may occur, 0, 1, 2, or more times in the bootstrap sample. In effect, this is a random re-weighting of the original data. Because the bootstrap sample was taken at random, it is representative of the original data and predictors built using the bootstrap sample should be applicable to the original data.

Depending on how we apply them, bootstrap replicates of the tree procedure can be used to:

1. Avoid overfitting to accidents of the sample,
2. Reinforce patterns that persist despite small changes in the data.
3. Draw finer distinctions in estimated probabilities.

Bumping (Tibshirani and Knight, 1999) and bagging (Breiman, 1996) are two methods based on bootstrapped trees.

Bootstrapping re-weights the data set at random. We may also consider more systematic re-weighting of the data. Boosting—in particular, AdaBoost (Freund and Schapire, 1996)—is a method that utilizes systematically re-weighted trees. In boosting, observations that were poorly predicted in one step are given higher weight in the next step, so that the procedure eventually concentrates on the difficult to predict cases.

3.4.1 Bumping

An important way that trees are sensitive to the particulars of the data at hand is that they make short-sighted partitions. The algorithm doesn't know what the consequences of it's initial split will be for the subsequent partitions. Consequently, the resulting tree may overlook important features.

Bumping (Bootstrap UMbrella of Predictors) first creates several trees by re-sampling the data via the bootstrap and fitting to the bootstrap samples. It then chooses the single tree with the best performance. Since the result is a single tree, it is interpretable as a tree.

Bumping can help a classifier avoid getting stuck in a poor solution. That is, if there is a good single-tree representation for the data, bumping has more opportunities to find it—or, at least, not to miss it by an accident of the sample. However, because it produces a single tree, bumping still produces a coarse prediction.

Note that bumping depends on random perturbations of the sample. Running the same procedure twice on the same data set is liable to create two different estimates. It also depends on how many perturbations we choose to make.

3.4.2 Bagging

Bagging (Bootstrap AGgregation) creates several trees by re-sampling the data via the bootstrap and fitting trees to the bootstrap samples, but bagging predicts using the average of the predictions from the multiple trees. Averaging the perturbed trees results in a more stable predictor. Similarities across trees reinforce one another, but accidents of the algorithm tend to cancel one another out. Averaging also allows for greater nuance in the predicted probabilities through less coarse predictions.

Bagging maintains all the flexibility of trees while generating more nuanced and more stable predictors, but the result is no longer interpretable as a tree. In fact, it may be difficult to interpret the results at all.

As with bumping, bagging depends on random perturbations of the sample. Running the same

procedure twice on the same data set is liable to create two different estimates. The user must also choose the number of trees to produce.

3.4.3 Double bagging

Double bagging runs the bagging prediction technique twice. First run bagging to predict the response Y with a bagging prediction $\hat{y}_1$. Then compute residuals $r = Y - \hat{y}_1$. Now use the same predictors as before, but use bagging to estimate the residuals *r* obtaining a second bagging prediction $\hat{y}_2$. Our final double bagging prediction is $\hat{y} = \hat{y}_1 + \hat{y}_2$.

When applied to Mn/Model data, a double bagging prediction using K trees in each of two passes generally gives slightly better predictions than a single bagging prediction using 2K trees. It may seem odd that refitting the residuals can give improved fits, but that is not uncommon for nonlinear procedures.

3.4.4 Boosting

Boosting is another method that creates a predictor by averaging a collection of trees. However, boosting does not form the collection of trees by re-sampling. Instead, it forms a sequence of trees by re-weighting misclassified observations; observations misclassified in one tree have greater weight, and thus a better chance of being correctly classified, in the next tree. The resulting classifier is an average of the sequence of trees in which trees with lower error rates receive more weight.

The basic recipe for boosting is as follows:

1. Fit a tree to the data.
2. Weight the tree according to its misclassification rate—trees with fewer errors receive more weight.
3. Re-weight the data so that:
   (a) The incorrectly classified observations receive more weight.
   (b) The correctly classified observations receive less weight.
4. Repeat.

If the recipe repeated forever, we would never get a predictor. Therefore, the number of iterations to use is a tuning parameter that needs to be specified by the user. Also, the re-weighting scheme needs to be specified.

The observations that our tree mislabels are harder to account for. So, the next pass puts more emphasis on trying to get those right. That is, it assigns increased weight to the misclassified locations and decreased weight to correctly classified locations. It then fits a tree to the re-weighted data. Each iteration focuses its attention on those observations that the previous rule misclassified. The result is a collection of classifiers with different strengths. In fact, each tree is designed to be strongest where the previous tree was weak. Because the trees have focused their attention on different parts of the sample, their strengths tend to complement one another.

The boosted classifier averages the trees together in order to combine their various strengths, like

bagging. Unlike bagging, boosting assigns different weights to the trees according to their various error rates. Furthermore, there is nothing random about boosting; it does not rely on bootstrapping the sample.

Boosting suffers from the same issues of interpretation as bagging. Averages of trees do not result in new trees. The boosted classifier is not clearly expressible as an equation or as a tree. It often results in a very effective predictor, but an uninterpretable predictor.

3.5 Pros and Cons

This section summarizes advantages and disadvantages of each of the prediction methods outlined above.

3.5.1 Logistic Regression with BIC Selection

- Because the form of the model is assumed known up to a set of coefficients, logistic regression is readily interpretable.
- For the same reason, logistic regression is very restrictive. If the true behavior of the population is well approximated by one of the specified models, this is not an issue. However, this procedure has no way to discover patterns in the data.
- Because a single model is selected, only a few of the predictors will tend to have non-zero coefficients. This tends to ease interpretation.
- Because of the large number of possible models to consider, selecting a logistic regression using BIC will tend to be very slow.

3.5.2 Averaged Logistic Regression

- Because a linear combination of linear models is another linear model, averaged logistic regression is also readily interpretable and similarly restrictive.
- Because we average over several models, many predictor variables will tend to have non-zero coefficients.
- Averaged models tend to be more stable predictors than individual models.
- Because of the large number of possible models to consider, averaging logistic regressions will tend to be very slow.

3.5.3 Naive Bayes

- Naive Bayes fits a more flexible model than logistic regression, but assumes more form than trees.
- The resulting “model” is difficult to interpret than logistic regression because the estimated density functions g1, ..., gk are unrestricted.
- Density functions are difficult to estimate and may require a lot of data to estimate well.

3.5.4 Trees

- Trees are highly interpretable.
- Trees have a very flexible form.
- Trees are computationally convenient and efficient.
- Trees are discrete, which limits our ability to distinguish between locations.
- Trees are highly sensitive to particulars of the sample, which may limit their ability to find interesting patterns and affects their stability as predictors.

3.5.5 Bumping

- Bumping results in a tree, so it inherits the flexibility and interpretability but also the discreteness of trees.
- Bumping guards against overlooking good predictors because of the myopic way that trees are grown—one split at a time.
- Bootstrap re-sampling can be computationally expensive.
- Estimates are not unique.

3.5.6 Bagging

- Bagging inherits the flexibility of trees.
- Bagging alleviates the discreteness of trees.
- Bagging generally results in very stable predictors.
- Bagged trees are basically uninterpretable in form.
- Bootstrap re-sampling can be computationally expensive.
- Estimates are not unique.
- Double bagging improves prediction, but at the cost of additional computation.

3.5.7 Boosting

- Like bagging, boosting inherits the flexibility of trees, fits a smoother set of probabilities, and generally results in very stable predictors.
- Also like bagging, boosted trees are basically uninterpretable.
- Boosting requires tuning by the user.
- There is no existing function in S-plus for boosting. Furthermore, it creates a very complicated prediction rule making it difficult to translate into Python. This makes boosting an extremely inconvenient method to implement. We will not attempt to implement boosting unless it clearly outperforms alternative methods. It's too easy to make errors and too hard to find them.

4. Comparison and Recommendations

The only true test of a prediction method is how well it predicts. Thus, we will compare the prediction methods described in the previous chapter by applying them to Mn/Model data and measuring the quality of their predictions. As described in Chapter 2, our primary criteria will be the false positive rates obtained via cross-validation for 70% and 85% true positive rates. A secondary criterion will be the area under the ROC curve. Gain curves and other graphical comparisons were made, but they will not be described or displayed here in the interests of space.

In order to compare classification methods, we fit a model using each method to data from three regions in eastern Minnesota: Big Woods, Anoka Sand Plain, and Mille Lacs Uplands. For these regions, two data sets were considered—an original set and an extended set with approximately twice as many random locations. Table 4.1 summarizes the number of sites and random locations in each data set.

	Original			Extended		
	Big Woods	Anoka	Mille Lacs	Big Woods	Anoka	Mille Lacs
Sites	862	724	1037	863	728	1037
Random	1615	1112	3203	3402	2247	6427

Table 4.1: Number of Sites and Random Locations by Data Set

Each classification method orders locations according to their predicted probability of a site. Using these orderings, we call the set that captures 70% of all sites High Probability. The set that captures the next 15% of sites we call Medium Probability. The High Probability locations are the locations classified as sites to attain a 70% true positive rate. The High Probability and Medium Probability locations together give an 85% true positive rate. Within these categories, the lower the false positive rate, the better the method.

In the examples below, bumping and bagging were done with 11 trees: the original data and 10 bootstrap replications. Double bagging was done with 11 trees per pass. Calculations for boosting were made external to S-Plus in a program called R.

Tables 4.2, 4.3, and 4.4 collect the results of applying the predictors to Phase 4 site data (the `mod4.site` subset, which includes site centroids, secondary site locations, and random points). Tables 4.2 and 4.3 show the false positive rates at 70% and 85% true positive rate, respectively. Table 4.4 shows the areas under the ROC curves. All results are based on ten-fold cross-validated predictions.

Method	Big Woods	Original Anoka	Mille Lacs	Big Woods	Extended Anoka	Mille Lacs
BIC Logit	.19	.06	.03	.18	.06	.03
BMA Logit	.19	.06	.03	.18	.07	.03
Naïve Bayes	.19	.07	.04	.19	.06	.04
Tree	.17	.11	.02	.20	.08	.01
Bumping	.18	.10	.02	.19	.07	.01
Bagging	.15	.05	.01	.12	.04	.01
Boosting	.14	.05				
Double Bag.	.12	.03	.01	.09	.03	.01

Table 4.2: Cross-validated False Positive Rates for 70% True Positive Rate.

Method	Big Woods	Original Anoka	Mille Lacs	Big Woods	Extended Anoka	Mille Lacs
BIC Logit	.35	.15	.08	.34	.15	.07
BMA Logit	.34	.14	.08	.33	.15	.07
Naïve Bayes	.36	.20	.11	.37	.18	.11
Tree	.36	.18	.07	.36	.21	.06
Bumping	.37	.19	.08	.38	.20	.06
Bagging	.29	.13	.04	.26	.12	.04
Boosting	.28	.12				
Double Bag.	.24	.11	.03	.20	.11	.03

Table 4.3: Cross-validated False Positive Rates for 85% True Positive Rate.

Method	Big Woods	Original Anoka	Mille Lacs	Big Woods	Extended Anoka	Mille Lacs
BIC Logit	.827	.907	.954	.831	.904	.956
BMA Logit	.829	.908	.955	.833	.905	.957
Naïve Bayes	.821	.895	.939	.825	.900	.938
Tree	.820	.875	.947	.807	.880	.956
Bumping	.809	.878	.945	.812	.887	.950
Bagging	.862	.929	.973	.875	.934	.976
Double Bag.	.875	.939	.971	.890	.940	.973

Table 4.4: Cross-validated areas under the ROC curve.

Double bagging is consistently the best, followed by boosting (where available), and bagging. After these three, there is usually a large step down in quality of prediction, although which method is the next best varies between data sets. Because boosting is more difficult to implement in S-Plus and GIS, and because boosting is sandwiched between bagging and double bagging in terms of quality of prediction, boosting will not be considered further.

Examination of Tables 4.2 through 4.4 also shows that bagging and double bagging consistently benefit from additional random locations. This suggests that regional models in Mn/Model Phase 4 should be fit with more random locations than was done in Phase 3.

If we use bagging or double bagging, we must decide how many trees to use. In general, more passes is better (double better than single, triple better than double), and more trees per pass is better, but there are diminishing returns for both the number of passes and the number of trees. In addition, implementing the bagging procedures in GIS is possible, but tedious, and doubling the number of trees used doubles the amount of effort required to implement them. Therefore, control on the number of trees is needed. Table 4.5 shows the cross-validated false positive rates at 85% true positive rate for various numbers of passes (single, double, and triple bagging) and numbers of trees per pass. For single bagging, there is relatively little improvement after 10 or 11 trees. Double and triple bagging may improve the results slightly for the same total number of trees, but they show the largest improvements when using two or three passes with a fairly large number of trees each.

Bagging	Trees/pass	Big Woods	Big Woods ext.	Anoka
Single	3	.33	.34	.19
Single	6	.30	.28	.14
Single	11	.27	.26	.12
Single	16	.27	.24	.14
Single	21	.27	.24	.12
Single	24	.23	.25	.12
Double	5	.27	.25	.12
Double	6	.29	.24	.11
Double	8	.28	.25	.11
Double	10	.26	.22	.12
Double	12	.24	.22	.12
Triple	8	.24	.22	.09
Triple	10	.24	.22	.11
Triple	12	.25	.22	.10

Table 4.5: Cross-validated False Positive Rates for 85% True Positive Rate for various numbers of passes and trees per pass.

Based on its performance in these regions and the resources required to implement the prediction in GIS, we recommend that Bagging with 11 total trees be used to classify locations.

5. Example

In this example, we will go through a mock analysis using the new Phase 4 S-Plus functions and extended data from the Big Woods region. (In the software, regions are indicated by a region abbreviation, which for the extended Big Woods is `bgwd4ex`. This abbreviation will appear repeatedly.) We will look at a subset of the output produced and describe how to interpret it. Much more complete documentation on these functions and the output is available in the "User's Guide for Mn/Model Phase 4 S-Plus Software."

In the example below, S-Plus commands appear after a prompt "> ", and commands, variable names, and output are shown in a `monospaced` font.

The first step is to read the data into S-Plus using the `mnmodel.readdata()` function:

```
> mnmodel.readdata("bgwd4ex",rootvars="none")
Variables in bgwd4ex.data.all are:
 [1] "Id"          "X"           "Y"           "Abl"         "Alluv"
 [6] "Blg"         "Ded.blk1"    "Ded.cors"    "Ded.or30"    "Ded.priv"
[11] "Ded.swm"     "Dedbwet1"    "Dint"        "Dir.ww"      "Dislksed"
[16] "Dis.asbi"    "Dis.br"      "Dis.bw"      "Dis.con"     "Dis.hdw"
[21] "Dis.maj"     "Dis.min"     "Dis.mix"     "Dis.ok"      "Dis.pap"
[26] "Dis.pibf"    "D.dra30"     "Dis.pr"      "Dis.rb"      "Dis.sug"
[31] "Ht90"        "Lk1.size"    "Maj.area"    "Min.area"    "Mrdiv990"
[36] "Plk1size"    "Rel90a"      "Rgh90"       "Slp"         "Terr"
[41] "Vaw1"        "Vpw1"        "Lk.inout"    "Lkpinout"    "Wtpinout"
[46] "Site.type"   "Phase3"      "Phase4"      "sin.Dir.ww"  "cos.Dir.ww"

Variables in bgwd4ex.subsets are:
 [1] "p3.cent"        "p3.sec"         "p3.neg"         "p3.aux"
 [5] "p4.cent"        "p4.sec"         "p4.line"        "p4.poly"
 [9] "p4.surv"        "p4.aux"         "all.rand"       "no.rand"
[13] "all.locations"  "all3.sites"     "all4.sites"     "all3.survey"
[17] "all4.survey"    "mod3.cent"      "mod3.site"      "mod3.surv"
[21] "mod4.cent"      "mod4.site"      "mod4.surv"      "CVsets"
[25] "SCVsets"        "clusters"

Variables in bgwd4ex.transformed.vars are:
[1] "Dir.ww"

Total number of locations: 9420

    Number of Phase 4 centroid sites: 712
    Number of Phase 4 secondary sites: 111
    Number of Phase 4 line sites: 0
    Number of Phase 4 polygon sites: 40
Total number of Phase 4 sites: 863

    Number of Phase 3 negative surveys: 1274
    Number of Phase 4 DOT surveys: 3710
    Number of Phase 4 sites as surveys: 171
Total number of Phase 4 non-site survey locations: 5155

Total number of random: 3402

Total number for Phase 4 site-centroid models: 4114
Total number for Phase 4 site models: 4265
Total number for Phase 4 survey models: 9420
```

The function establishes some S-Plus data sets and prints some informative output about the data. First, it prints the names of all the variables present in the data set. Next, it prints the

names of subsetting variables that it created to permit analysis of various subsets of data. The subsetting variables most likely to be used are `mod4.site` and `mod4.surv`, which indicate the data to be used in Phase 4 models of potential for sites and survey bias, respectively. Third, it prints the names of any variables that it transformed. For example, direction variables are converted to their sines and cosines, and, by default, a list of variables with distributions skewed to the right are transformed to their square roots (there is no need to use the square roots when using tree-based methods such as bagging). Finally, it prints out counts of data falling into different categories: total number, different types of sites, different types of negative surveys, random points, and the number of points used Phase 4 site and survey models.

The next step is optional. A user can request that descriptive statistics and graphics be produced for the variables in the data set. Some of these summaries are done separately for sites, negative surveys, and random points, so this step should only be done on subsets that contain all three types. Usually this means that the selected subset will be `mod4.surv`.

This step produces many pages of output and many plots. To manage this avalanche, results from different analyses are placed in different directories on the computer. The user indicates the directory with a user-selected label. We will use the label "example", so output will appear in the directory `BGWD4NEW.example.summaries`. (Note: you should use different labels for site and survey models; if you don't, the results of your second analysis will over write those of your first analysis.)

Variable summaries are produced using the function `mnmodel.var.summaries()`. We must indicate the subset we will use (`mod4.surv`) and the label we want (`example`).

```
> mnmodel.var.summaries("BGWD4NEW",mod4.surv,"example")
```

There is no output visibly printed by S-Plus, because all output has been redirected to files in the `BGWD4NEW.example.summaries` directory. The first summary is descriptive statistics. All descriptive statistics are placed in the file `variable.summary.txt` in the output directory. The output to this file begins with means and standard deviations of the variables, reported separately for sites, negative survey points, and random points. In our example, the file begins with Abl (elevation) and Alluv (on alluvium) before continuing through the remainder of the variables.

```
Means and standard deviations of variables:

, , Abl
         Sites Negative Surveys Random Points
mean 942.92816         927.4675     987.26543
  sd  91.70153          98.5058      77.79021

, , Alluv
          Sites Negative Surveys Random Points
mean 0.07763615       0.09253152    0.02557319
  sd 0.26775334       0.28980292    0.15788138
```

From this output we can see the mean elevations for sites, negative surveys, and random points are 942.9, 927.5, and 987.3 respectively, and the standard deviations of elevation for these subsets are 91.7, 98.5, and 77.8. Thus sites and negative surveys generally occurred at lower elevations (probably near water), and there is a bit less variability in the elevations of the random

points.

After means and variances come two-sample Wilcoxon rank-sum tests for each variable comparing site data to random location data, and negative survey data to random location data. The two-sample Wilcoxon test is a nonparametric test of location (center or median). That is, it asks whether there is evidence that two groups (that are otherwise comparable) are centered at different values, and the test does not rely on the shape of the underlying group distributions. The p-value answers the question "Assuming that the two groups really had the same center, how likely are we to have observed data with centers this far apart (as measured by the Wilcoxon)?" Small p-values indicate that the groups have different centers. For both comparisons, the Wilcoxon statistic and the p-value are reported. Our file continues

```
Wilcoxon tests of variables:

, , Abl
         Sites vs Random Surveys vs Random
Wilcoxon   -1.350454e+01     -2.957183e+01
 p-value    1.470355e-41     3.442133e-192

, , Alluv
         Sites vs Random Surveys vs Random
Wilcoxon    7.321045e+00      1.221593e+01
 p-value    2.460474e-13      2.555507e-34
```

We show only the first two variables, and we see the results of comparing site to random and negative survey to random for elevation and alluvium. All of these p-values are tiny, so there is overwhelming evidence against the four hypotheses that (for this region) these variables are equally distributed for sites, negative surveys, and random points. For a site model, this is all to the good, because the whole foundation of Mn/Model is that sites are generally located on recognizable landscapes. However, this also shows that there is bias in the selection of survey locations.

Finally, the Spearman rank correlation matrix of the variables based on all of the data is printed. The Spearman rank correlation coefficient indicates how two variables vary together: positive values indicate that they vary directly (go up and down together); negative values indicate that they vary inversely (one goes up while the other goes down). The further the coefficient is from zero, the stronger the relationship. The rank correlation coefficient is bounded between one and negative one. A subset of the correlation matrix is presented here:

```
Spearman rank correlations:
             Abl  Alluv    Blg Ded.blk1 Ded.cors
       Abl  1.000 -0.415  0.013   -0.307    0.161 ...
     Alluv -0.415  1.000  0.063    0.324   -0.108 ...
       Blg  0.013  0.063  1.000    0.021    0.074 ...
  Ded.blk1 -0.307  0.324  0.021    1.000    0.131 ...
  Ded.cors  0.161 -0.108  0.074    0.131    1.000 ...
             .      .      .        .        .    .
             .      .      .        .        .      .
             .      .      .        .        .        .
```

We see, for example, that Abl (elevation) is negatively related to Ded.blk1 (the distance to edge of the nearest big lake)---locations at higher elevations are closer to big lakes.

After summary statistics, `mnmodel.var.summaries()` produces a pdf graphics file for each

variable. This file contains a stack of three histograms showing the distribution of the variable separately for sites, negative surveys, and random locations. Histograms represent data by the area of the bars in the plot, so the bigger the bar at a given value of the variable, the more data are in that range. The files are named `var.1.pdf`, where "var" is the variable.

Figure 5.1 shows Abl.1.pdf (elevation). In this figure, we can see that negative survey points have a fairly sharp peak around 950 to 970 feet of elevation, but sites are more spread out in the middle. All three groups show a tail on the left, indicating a fair number of points at elevations considerably lower than the bulk of the data.

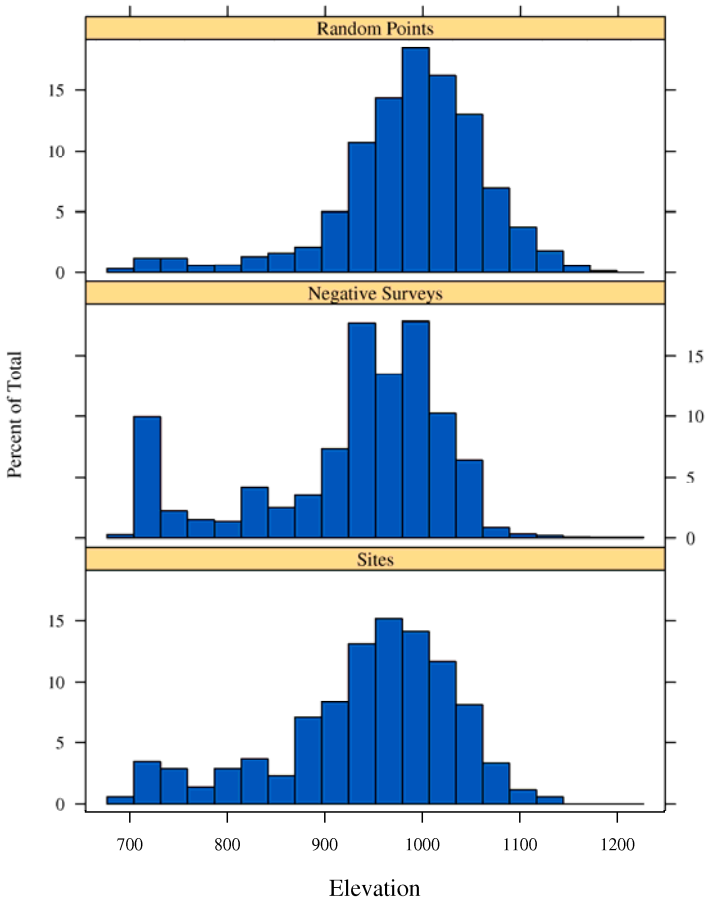


Figure 5.1. Sample histograms for elevation separately for sites, negative surveys, and random locations.

It is now time to fit a model. To do this we use the function `mnmodel.fit()`. This function needs to know what dataset we are using, what subset (type of model) we are doing, what label to use for output, and which fitting method to use. Thus, a typical command might be

```
> mnmodel.fit("bgwd4ex",mod4.surv,"example","bagging")
Constructing bgwd4ex.example.results

Available predictors:
 [1] "Id"          "X"           "Y"           "Abl"         "Alluv"
```

```
 [6] "Blg"         "Ded.blk1"    "Ded.cors"    "Ded.or30"    "Ded.priv"
[11] "Ded.swm"     "Dedbwet1"    "Dint"        "Dir.ww"      "Dislksed"
[16] "Dis.asbi"    "Dis.br"      "Dis.bw"      "Dis.con"     "Dis.hdw"
[21] "Dis.maj"     "Dis.min"     "Dis.mix"     "Dis.ok"      "Dis.pap"
[26] "Dis.pibf"    "D.dra30"     "Dis.pr"      "Dis.rb"      "Dis.sug"
[31] "Ht90"        "Lk1.size"    "Maj.area"    "Min.area"    "Mrdiv990"
[36] "Plk1size"    "Rel90a"      "Rgh90"       "Slp"         "Terr"
[41] "Vaw1"        "Vpw1"        "Lk.inout"    "Lkpinout"    "Wtpinout"
[46] "Site.type"   "Phase3"      "Phase4"      "sin.Dir.ww"  "cos.Dir.ww"


Total number of locations: 9420
Total number of sites: 863
Total number of survey points: 5155
Total number of non-random: 6018
Total number of random: 3402
Eliminating non-predictors

Eliminating previously transformed variables from predictors:
[1] "Dir.ww"

Eliminating collinear and/or nearly constant variables from predictors:
[1] "Dis.mix"

Fitting with these variables:
[1] "Abl"         "Alluv"       "Blg"         "Ded.blk1"    "Ded.cors"
 [6] "Ded.or30"    "Ded.priv"    "Ded.swm"     "Dedbwet1"    "Dint"
[11] "Dislksed"    "Dis.asbi"    "Dis.br"      "Dis.bw"      "Dis.con"
[16] "Dis.hdw"     "Dis.maj"     "Dis.min"     "Dis.ok"      "Dis.pap"
[21] "Dis.pibf"    "D.dra30"     "Dis.pr"      "Dis.rb"      "Dis.sug"
[26] "Ht90"        "Lk1.size"    "Maj.area"    "Min.area"    "Mrdiv990"
[31] "Plk1size"    "Rel90a"      "Rgh90"       "Slp"         "Terr"
[36] "Vaw1"        "Vpw1"        "Lk.inout"    "Lkpinout"    "Wtpinout"
[41] "sin.Dir.ww"  "cos.Dir.ww"

Doing bagging ... done.
Cross-validating, please be patient: 1  2  3  4  5  6  7  8  9  10   done.
Spatially cross-validating, please be patient: 1  2  3  4  5  6  7  8  9  10   done.
```

The output begins with a repetition of summary information about variables present, variables transformed, variables used, and how many data of various types are used. Then it gets down to the business of fitting the model. Because we are doing ordinary and spatial cross-validation, it actually does the fitting 21 times. This could take some time, so it prints progress information letting you know which fit it is doing.

Now that we've fit the model, we need to get summaries of how well the model worked as well as information we will use to actually implement the model in GIS. We use the `mnmodel.fit.summaries()` function to do this. The ordinary usage of this function takes three arguments: a region, a label, and a method. You may optionally set the *a priori* probability for sites, which will affect only the gain plots shown below. The default for the *a priori* probability is .01.

```
> mnmodel.fit.summaries("BGWD4NEW","example","bagging")
```

Output from `mnmodel.fit.summaries()` goes into the `REGIONABBR.label.summaries` directory, which in our example is `bgwd4ex.example.summaries`. The output consists of a text file named `method.summary.txt` (`bagging.summary.txt` in the example) and either 8 or 13 summary graphics in pdf format, the number depending on whether spatially cross-validated predictions are available.

Let us begin with a discussion of the graphs, which are produced for the predictions, cross-validated predictions, and, if available, spatially cross-validated predictions. For these kinds of predictions we produce cumulative plots, ROC curves, and gain curves (Chapter 2). For cross-validated data, we also produce graphs with multiple cumulative curves and ROC curves, one for each cross-validation subset. The graphs are stored in the files with the names:

appcumpred.method.pdf, appgain.method.pdf, approc.method.pdf.

cvcumpred.method.pdf, cvgain.method.pdf, cvroc.method.pdf, cvcumpred.multi.method.pdf, cvroc.multi.method.pdf.

scvcumpred.method.pdf, scvgain.method.pdf, scvroc.method.pdf, scvcumpred.multi.method.pdf, scvroc.multi.method.pdf.

The "app" prefix stands for "apparent" and indicates results for predictions constructed using all the data. The "cv" and "scv" prefixes indicate results for cross-validated and spatially cross-validated predictions. The "multi" indicates separate curves for each cross-validation subset. In all cases, the "method" is replaced with the method of interest, for example, `approc.bagging.pdf`. The cumulative predicted plots allow us to see the actual values predicted for different locations and to compare the distributions of predictions for sites and non-sites. An example cumulative predicted plot is shown in Figure 5.2.

We see in Figure 5.2 two bands of points, the lower band for random points and the upper band for non-random points (sites and surveys). The curves show the cumulative distributions, that is, fractions of points with values less than or equal to the current value, for random points (dashed) and sites (solid). For a good prediction, the random points should cluster to the left and their cumulative should rise steeply and then flatten near the top of the graph; and the sites should cluster to the right, with their cumulative staying low and then rising sharply on the right. The greater the area between the two curves, the better the prediction method is doing.

The "multi" form of the plot shows the same points, but the cumulative curves are plotted separately for the 10 cross-validation subsets. This illustrates the variability in the quality of the prediction. This is illustrated in Figure 5.3. In this example, the curves are fairly stable across the subsets.

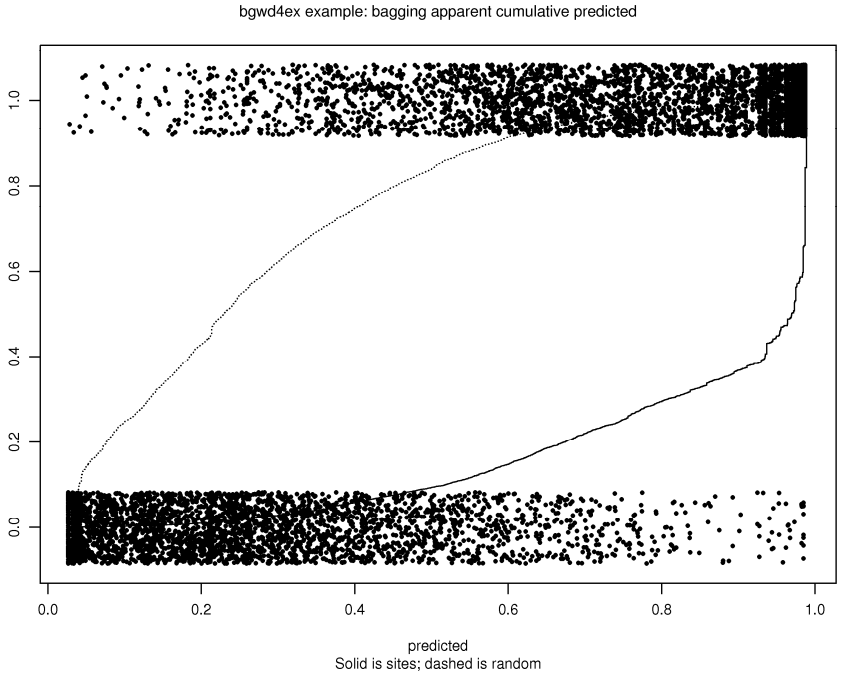


Figure 5.2. Cumulative apparent predicted plot showing separate curves for sites (solid) and non-sites (dotted).

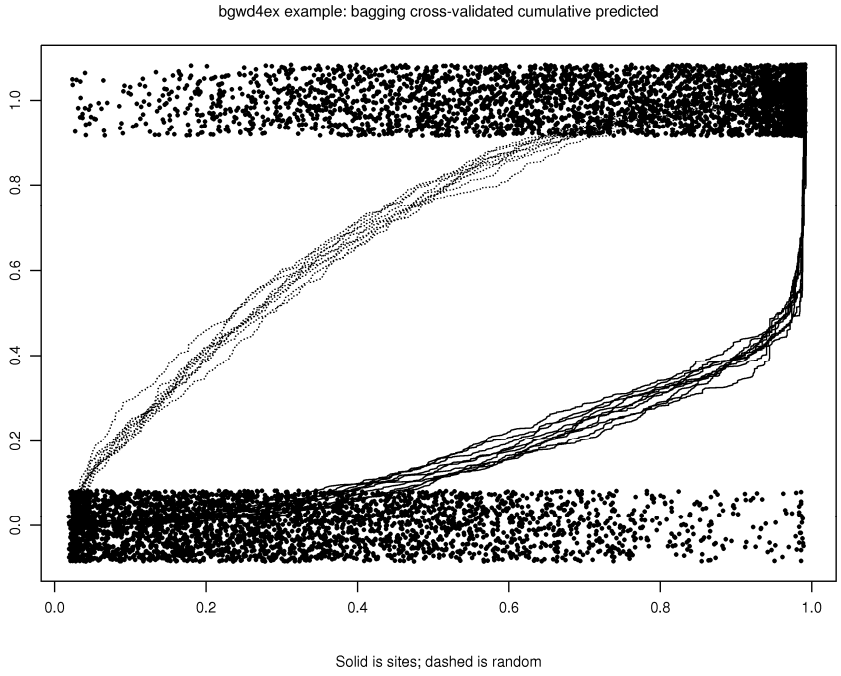


Figure 5.3. Multiple form of cumulative predicted plots, showing separate curves for each cross-validation subset.

The ROC curve plots the true positive rate (vertical axis) against the false positive rate (horizontal axis) for a large number of potential thresholds. As you adjust these thresholds, you change the fractions of sites and non-sites that are classified as sites (these are the true and false positive rates). The curve starts in the lower left hand corner and moves to the upper right hand corner. Ideally, the curve should move up to the top very quickly and then move along the upper boundary. This gives us maximum true positive rate with minimum false positive rate. The diagonal line corresponds to randomly guessing site versus random. The false positive rates for .7 and .85 true positive rates are highlighted. While our principal figure of merit is the false positive rate at a .85 true positive rate, the area under the ROC curve gives an overall summary of the quality of the prediction. Figure 5.4 shows the ROC curve based on cross-validated predictions.

The "multi" form of the graph plots the ROC curve separately for each cross-validation subset. Figure 5.5 shows an example of these multiple ROC curves for the same data shown in Figure 5.4. Again we see that the ROC curves are fairly stable.

The final graph is the gain curves. These curves are already somewhat complex, so we only plot them for the full data set, and not separately by cross-validation subsets. The gain curve depends on the *a priori* probability used when the summary function was called. The gain referred to in the name of the curve is the ratio of the posterior probability of a location being a site to the prior probability of a location being a site. "Posterior" means after seeing the data and doing the classification. The upper curves are probabilities for locations that were classified to be sites; we want these curves to be as high as possible so that as large a fraction as possible of those locations classified as sites actually are sites. Conversely, the lower curves are probabilities for locations classified as non-sites; we want these curves to be as low as possible. The curves show these ratios as a function of the true and false positive rates.

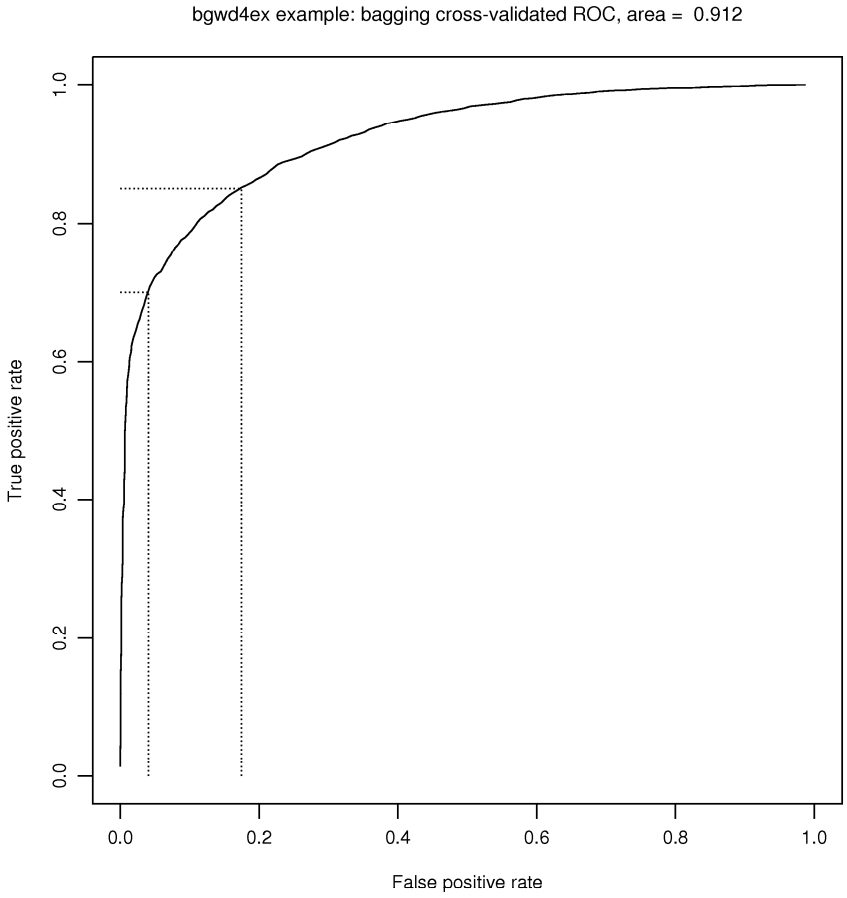


Figure 5.4. ROC curve for cross-validated predictions.

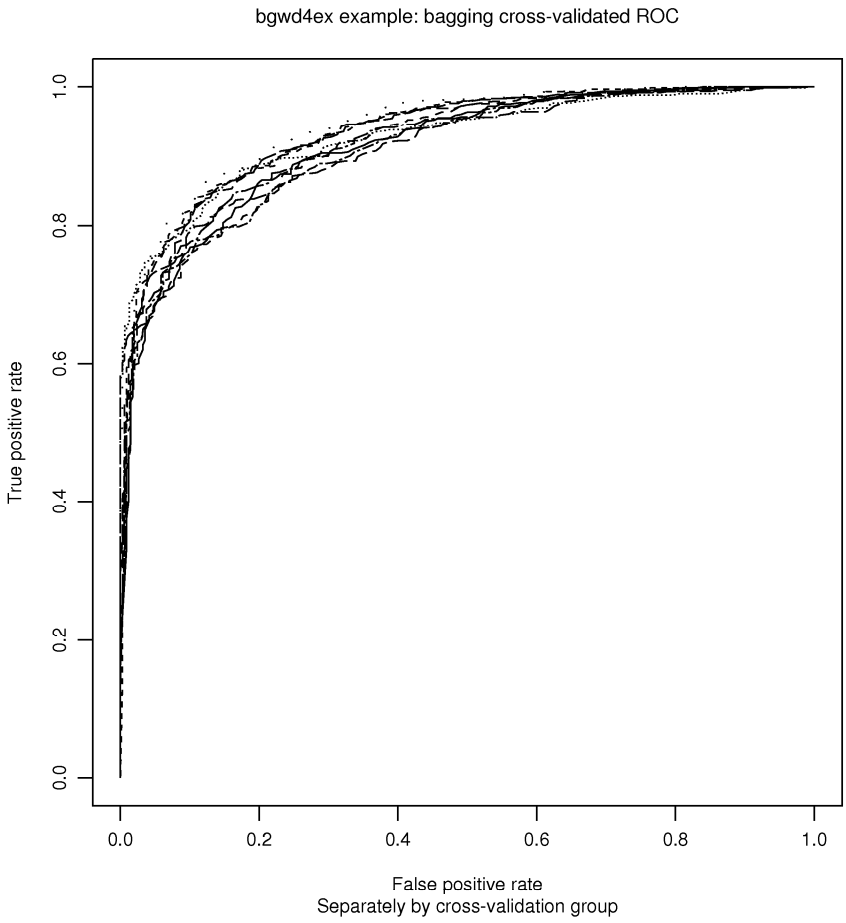


Figure 5.5. ROC curves separately for each cross-validation subset.

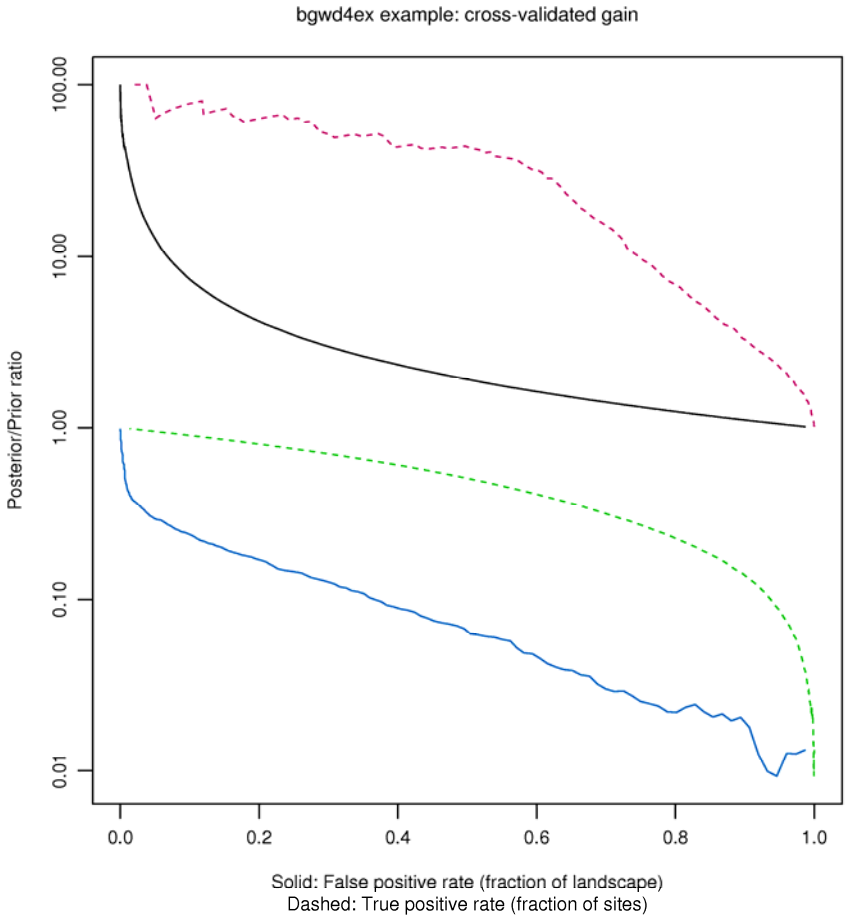


Figure 5.6. Gain curves for sites and non-sites.

In addition to graphical summaries, we also have text/numeric summaries of the results in a text file named `method.summary.txt` (`bagging.summary.txt` in the example). These files consist of a method-specific part and a generic part. We begin with an example of the generic part and describe the contents.

```
Apparent false positive rates and cutoffs
at .70 and .85 true positive rates
          Cutoff  TPR  FPR
  HIGH 0.8075281 0.70 0.02
MEDIUM 0.6026849 0.85 0.09
```

The output begins with apparent false positive rates, which are based on predictions from the full data set. Here is how to interpret the output. For this prediction (bagging in this example), if we use a cutoff of .8075 and declare locations with predictions greater than the cutoff to be sites, then we capture 70% of the sites (true positive rate of 70%) but only 2% of the random points (false positive rate of 2%). Similarly, if we choose a cutoff of .6027, then we capture 85% of the sites and 9% of the random points.

The apparent error rates are too optimistic, so we also compute error rates using cross-validation. Our 10-fold cross-validation fits the model using 90% of the data and then uses that model to predict the 10% of the data held out. We cycle through 10 different subsets of hold out data until all locations were predicted using models fit excluding the points of interest. Cross-validated results are summarized next.

```
Cross-validated true and false positive rates and
cutoffs at nominal .70 and .85 true positive rates

, , HIGH
            Cutoff   TPR   FPR
        1 0.8096584 0.660 0.040
        2 0.7989974 0.710 0.030
        3 0.8027825 0.680 0.020
        4 0.8059832 0.710 0.040
        5 0.8101571 0.690 0.030
        6 0.8073968 0.690 0.050
        7 0.8142347 0.660 0.040
        8 0.8082299 0.720 0.030
        9 0.8160169 0.670 0.040
       10 0.8104025 0.670 0.030
 Average 0.8083859 0.686 0.035
Combined 0.7833581 0.700 0.040

, , MEDIUM
            Cutoff   TPR   FPR
        1 0.6041536 0.770 0.120
        2 0.6076615 0.830 0.120
        3 0.5999492 0.830 0.110
        4 0.5936856 0.850 0.140
        5 0.6077833 0.830 0.120
        6 0.5994719 0.820 0.140
        7 0.5954879 0.800 0.150
        8 0.5961668 0.840 0.110
        9 0.6019120 0.820 0.190
       10 0.6043290 0.800 0.120
 Average 0.6010601 0.819 0.132
Combined 0.5484972 0.850 0.170
```

Begin with the "High" results, which are supposed to capture 70% of the sites, and consider group 1. We fit a model to 90% of the data and find the cutoff that gives us 70% true positive rate in the data used to fit the model. The cutoff is .8097. We now apply this model and cutoff to the 10% of the data held back. When we do this, we find that we actually obtain a 66% true positive rate (instead of the nominal 70%) and a 4% false positive rate. Similarly, when we hold back group 10, we select a cutoff of .8104 and obtain a TPR of 67% and a FPR of 3%. Averaging across the 10 groups (the line labeled "Average"), the cutoff is .8083, the TPR is 69%, and the FPR is 3.5%. The average cutoff of .8083 is pretty close to the apparent cutoff from above (.8075), so that is reassuring, but the TPR that we actually attain is below the 70% that we wanted. To get 70% TPR, we must reduce the cutoff to .7834 (the line labeled "Combined"), which gives us a FPR of 4%. Thus the apparent cutoffs are too optimistic, and we really should use a lower cutoff to attain 70% TPR when predicting to new data. (Most site models show differences between apparent and cross-validated results much greater than this example survey model.)

The "Medium" results tell a similar story. They should have a TPR of 85%, but when cross-validated, the TPR only averages about 82%. We need to lower the threshold to .5485 to capture 85% of the sites when predicting to new data.

Results from spatial cross-validation are even more pessimistic, but that is because they simulate a much more challenging modeling situation. In cross-validation, we predict using models fit without benefit of the data we are trying to predict. However, since the 10 subsets are chosen randomly, there is a good chance that landscapes similar to where we are trying to predict are included in the modeling subset. In spatial cross-validation, we exclude data in small spatial clusters rather than one point at a time. Spatial cross-validation simulates predicting into landscapes where we've never had any data before! We have to expect that our predictions will work more poorly, and, unfortunately, our expectations are met.

Spatial cross-validation results are reported just like the cross-validation results. For example, when trying to capture 85% of the sites, our average threshold is .606 and we achieve an average TPR of 52% with a FPR of 12%. To get our desired TPR of 85%, we must lower the threshold to .356, which gives us a FPR of 38%. These results should be compared with .548 and FPR of 17% for simple cross-validation.

```
Spatial cross-validated true and false positive rates and
cutoffs at nominal .70 and .85 true positive rates

, , HIGH
            Cutoff   TPR   FPR
       1 0.8002406 0.560 0.020
       2 0.8380684 0.050 0.000
       3 0.8259919 0.350 0.020
       4 0.8474061 0.060 0.030
       5 0.8163326 0.080 0.010
       6 0.8298684 0.020 0.000
       7 0.7338127 0.380 0.090
       8 0.8054010 0.210 0.020
       9 0.7975499 0.370 0.030
      10 0.8104839 0.390 0.040
 Average 0.8105155 0.247 0.026
Combined 0.4460000 0.700 0.250

, , MEDIUM
```

```
            Cutoff   TPR   FPR
         1 0.6085676 0.750 0.090
         2 0.6184312 0.290 0.070
         3 0.6082854 0.590 0.080
         4 0.6292347 0.260 0.110
         5 0.6230563 0.430 0.140
         6 0.6030965 0.440 0.120
         7 0.5632239 0.490 0.190
         8 0.6111954 0.720 0.150
         9 0.5863917 0.730 0.100
        10 0.6082661 0.460 0.110
   Average 0.6059749 0.516 0.116
  Combined 0.3556858 0.850 0.380
```

We recommend using the "combined" cross-validated cutoffs to do classification. If you know that you are predicting into a landscape unlike the rest of your data set, you could consider the more pessimistic spatially cross-validated threshold.

The output file also contains method-specific information for how to implement the prediction routine. Calling this part of the output "arcane" would be too generous. Details of how to interpret and use the method-specific output can be found in the "User's Guide for Mn/Model Phase 4 S-Plus Software."

6. Conclusion

Phase 4 of Mn/Model provides the opportunity to improve both the archaeological data bases and statistical prediction methods that were used in Phase 3. Among the methods considered here, bagging (bagged trees), double bagging (bagged trees twice), and boosting consistently outperformed the alternatives. Boosting does not have a native implementation in S-Plus, and it would appear fairly challenging to port the implementation from R to S-Plus. For this reason, we do not recommend boosting. Double bagging generally outperformed bagging, although typically not by a large amount. However, double bagging does best with two passes each with a fairly large number of trees, so double bagging will, in general, require doubling the already considerable effort needed to implement bagging in GIS. For this reason, we cannot wholeheartedly recommend double bagging.

The best compromise between accuracy and efficiency is bagging, using a single pass with roughly 10 total trees (11 is the default in the S-Plus software supplied with this project). As was shown in Chapter 4, bagging provides a 15% to 50% improvement in false positive rate over the method used in Phase 3 (logistic regression with BIC selection) depending on data set and true positive rate. Software to implement bagging has been provided as part of this project and is documented in "User's Guide for Mn/Model Phase 4 S-Plus Software."

7. References

Breiman, L. (1996). "Bagging Predictors," *Machine Learning* 26: 123-140.

Breiman, L., Friedman, J., Olshen, R. and Stone, C. (1984). *Classification and Regression Trees*, Wadsworth, Belmont, CA.

Efron, B., and Tibshirani, R. (1993). *An Introduction to the Bootstrap*, Chapman and Hall, London.

Freund, Y., and Schapire, R. (1996) Experiments with a New Boosting Algorithm, *Machine Learning: Proceedings of the Thirteenth Annual Conference*, Morgan Kauffman, San Francisco, CA, pp. 148-156.

Hand, D., and Yu, K. (2001). "Idiot's Bayes: Not So Stupid after All?," *International Statistical Review*, 69: 385-398.

Hartigan, J. A. (1975). *Clustering Algorithms*, Wiley, New York, NY.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*, Springer, New York, NY.

Hudak, G. J., Hobbs, E., Brooks, A., Sersland, C. A., and Phillips, C. (2002). *Mn/Model: A Predictive Model of Precontact Archaelogical Site Location for The State of Minnesota*, final report to Mn/DOT 73217, Minnesota Department of Transportation, St. Paul, MN.

Mosteller, F., and Tukey, J. W. (1977). *Data Analysis and Regression*, Addison-Wesley, Reading, MA.

Pepe, M. S. (2003). *The Statistical Evaluation of Medical Tests for Classification and Prediction*, Oxford, New York, NY.

Schwarz, G. (1978). "Estimating the Dimension of a Model," *The Annals of Statistics*, 6: 461-464.

Tibshirani, R., and Knight, K. (1999). "Model Search by Bootstrap "Bumping"," *Journal of Computational and Graphical Statistics*, 8: 671-686.

Yerushalmy, J. (1947). "Statistical Problems in Assessing Methods of Medical Diagnosis with Special Reference to X-Ray Techniques," *Public Health Rep.* 62: 1432-1449.