**Supplementary Material**

Nest site selection patterns of a local Egyptıan Vulture *Neophron* *percnopterus* populatıon in Turkey

BİLGECAN ŞEN, JOSÉ P. TAVARES and C. CAN BİLGİN

**Contents**

Classification and Regression Trees (CART)

Random Forests

**Classification and Regression Trees (CART)**

In classification trees, we start with a root node in which the sample is not partitioned yet and the error rate is highest. For instance, in the nest site selection sample, there were a total of 69 points with 39 nests and 30 random points, and the root node error was 30/69=0.43 at the beginning. The aim is to reduce this error rate by splitting the sample depending on the explanatory variables and forming a tree. The commonly used algorithm to do so is called Gini index, and for a two category nominal response variable it is formulated as:

$$G=\sum\_{j\ne k}^{}p\_{ij}×p\_{ik}$$

where *k* and *j* refer to categories as presence/absence and *p* is the proportion of that category at the *i*th leaf after splitting the data depending on an explanatory variable. The aim is to minimize the *G* value on the respective leaves after each split.

Data, either on the root node or on the other nodes following it, are always split in two, so a nod after a split forms two leaves. The splitting process differs depending on the type of the explanatory variable:

* When the explanatory variable is nominal with equal to or more than two categories, every permutation of the categories is tried out to split the response variable. For example with a 3 category nominal variable a, b and c, permutations of ab-c, ac-b and bc-a are considered, resulting in two leaves, and a G value for every permutation is calculated.
* When the explanatory variable is continuous, the response variable can only be partitioned in the original order of the continuous variables. For example, with a continuous variable of values 1, 2 and 3, splitting can only carried out in two ways: either 12 in one leaf and 3 in the other, or 1 in one leaf and 23 in the other. Again, for every possible split G values are calculated.

At the root node, every explanatory variable is used to split the data in two by following the methods explained above and the variable and its appropriate permutation causing the lowest G value after the split is chosen. The data is now divided in two and two leaves have been formed. The process is repeated for the two leaves again using all the variables and for every other leaf afterwards until a pre-defined stop criterion is met or splitting is no longer feasible. When a leaf is split, it is called a node. The final leaves (or terminal nodes) that have not been split show how many splits were carried out when constructing the tree; if *m* is the number of leaves then number of splits is always *m*-1. Data is predicted from the final leaves. For example 11/2 in a leaf means that there are 11 nests and 2 random points in that leaf’s prediction, hence error rate is 2/13=0.15. The error rate of the tree is calculated through all leaves and is usually presented as a “relative error rate” which is relative to the root node error.

It can be deduced that the earlier a variable is selected for a split, then the more important it is on affecting the response variable. Also, variables that are tied to each other through nodes throughout the tree are assumed to be interacting.

Much like in GLM, variable selection that will be included in the final model is an internal part of the CART. Without any criterion, the tree will overgrow and therefore will be an overfitting model (Figure 3.2). The relative error rate is not a good indicator when deciding the tree size, because it always decreases as the tree grows. The criterion to decide the size of the tree is the cost-complexity parameter (*cp*). The parameter cp is inversely correlated with the number of splits in a tree. As *cp* decreases, tree size increases and as a result explanatory power of the tree also increases but there is a chance that the model will overfit. As *cp* increases, tree becomes smaller, only a few variables are included in the final tree and the explanatory power of the tree is reduced. The aim then is to find the optimal *cp* for the constructed tree.

Cross validation is used to compare the effect of the different number of splits or *cp* on the error rate. Data is divided randomly into *k* subsets (default *k* is 10). Then *k* trees are grown fully without constraints, but at each tree one subset is left out and resulting tree is used to predict the subset that was left out. Error rate from that prediction is called cross validation error rate and it is unbiased when compared to the relative error rate. The data though is not predicted just once with the full tree but rather predicted after every split, so following any split we have *k* number of cross validation error rates. Therefore the mean and the standard deviation can be calculated using those error rates for every split and every *cp*. Finally the tree is pruned back to lowest number of splits or highest number of *cp* which is within one standard error of the split, or the *cp* with the lowest cross validation error rate. This way a simpler model is preferred over more complex ones that do not have a significant effect on the model’s prediction performance. This is a direct analogy to what Richard (2008) suggested when selecting models in multi model inference.

**Random Forests**

The biggest difference of RF comes from the method which trees are grown. Before splitting a node in two, instead of contesting all variables for their *G* values as in CART, a pre-determined and randomly selected number of variables are tried out. This number is usually determined as the squared root of *m* (number of variables) and is called *mtry*. For example if *mtry* of a RF is 5, in every tree and in every split of those trees 5 variables are randomly selected and contested over their *G* value, and the one giving the lowest *G* value is selected as splitting criterion.

Another important difference of RF from CART is that a random bootstrap sample with replacement of the original data is used when growing each tree. In this way an average of 33% of all cases are left out during the construction of each tree. These cases that are left out are called “out of bag” (OOB) and are used for validating the data, an analogy of cross validation. When a case is OOB from a tree, it is dropped down from that tree and a prediction is made through the final leaves of the tree. For example, when a case of an actual nest is OOB from a tree, it will be predicted as an actual nest or a random point. Since in every tree an average of 33% of cases are left out, then in average a case will be left out in 33% of all trees. If we grow the forest with 1000 trees, our aforementioned case of actual nest site will be OOB from (on average) 333 trees and all of those trees will have a prediction on whether the case is a nest and or a random point. When the number of trees predicting the case to be an actual nest is higher than the number of trees predicting it to be a random point then the model’s prediction for that case is actual nest, if the opposite were true then the model would predict it to be a random point, which would be an error. This is the majority vote aspect of the RF. The rate of all the mis-predicted cases is called OOB error rate and it is an unbiased estimation for the model’s predictive capabilities.

The most important aspect of Random Forests is the variable importance measures in which we can deduce which variables have more effect on the prediction of cases. There are several importance measures, but we used what is usually defined as the “mean decrease in accuracy”. Breiman simply named this as “measure 1”.

When estimating measure 1, OOB error rates for each tree in the forest are calculated. These error rates are not majority votes based, and simply refer to classification error for each individual tree. Then the variable under study, say *m*, is permuted over all OOB cases for each tree, and a new OOB error rate is calculated for every individual tree. If the variable *m* has an effect over predictions different than random, then the new error rates after the permutation should be higher. The difference between the original OOB error rate and permuted OOB error rate is calculated for each tree and the sum of these differences is divided by the number of trees in the forest. After normalizing this value, it gives us the mean decrease in accuracy when the values of variable *m* are random. This process is repeated for every variable in the study. If a variable caused more decrease when random, then it has more impact on the model’s predictive capabilities. This way, every variable in the study can be compared with each other in terms of importance. In our study, we assumed that important variables reflected the nest site selection or habitat features that affected breeding performance of Egyptian vultures.

The only parameters to adjust when constructing random forests are the number of trees in the forest (*ntree*) and previously defined *mtry*. Because there is a significant random aspect of this method, OOB error rates and scores of variable importance measures can differ in every model. It has been shown that higher number of trees in a forest leads to better variable importance score stability (Genuer et al., 2010), so we used 2000 trees to construct every forest. We used the tuneRF function in the randomForest package in R to find the best *mtry* value. This function, starting with a pre-determined value of *mtry*, multiplies or divides *mtry* by a factor and builds forests until OOB error rate does not improve. In our case, tuneRF (with a factor of 2) built 3 forests with *mtry* values of 2, 4 and 8 in the initial model construction and 1, 2 and 4 in the final forest construction for both nest site selection and breeding success assessment. From these forests we chose the *mtry* value of the forest with the lowest OOB error rate. If OOB error rates were equal we chose the default *mtry* value. Then we grew the initial forest using the chosen *mtry* value. In the second step, to improve the OOB error rate of the initial forest, we eliminated variables which had a negative variable importance score. A negative score means that a random permutation of a variable performs better than its original combination, indicating that the variable is increasing the OOB error rate. After the elimination we used *tuneRF* for the second time and chose the best *mtry* value for the final forest, repeating the aforementioned method. We constructed 10 forests using those *mtry* values and chose the forest with the lowest OOB-error rate to be the final forest.

We used partial dependence plots to illustrate the individual effects of the variables on the probability of a point being a nest (Hastie et al., 2009). The function to draw partial dependence plots is as follows:

$$f\left(x\right)=logp\_{k}\left(x\right)-1/K×\sum\_{j=1}^{K}\left(logp\_{j}(x)\right)$$

*K* is the number of classes. *pk* is the fraction of votes for the class that is being plotted and *pj* is the fraction of votes for class *j*.