CART and Random Forests

Classification and regression trees (CART) and random forests are both tree-based methods. Trees are flexible data-driven methods to determine an outcome using splits, or logical rules, on the independent variables. Trees have the ability to more easily capture nonlinear relationships than linear and logistic regression, and can be used for both a continuous outcome (like in linear regression) and a categorical outcome (like in logistic regression).

CART

The simpler of the two methods is CART. In a CART model, a single tree is constructed, and for this reason, CART models are very easy to interpret. It is a popular model in applications where the method needs to be easy to explain. This interpretability and the ability to capture nonlinearities in the data are two major benefits of CART.

Each split in a CART tree is always based on only one independent variable, and a CART tree can be given as many independent variables as you have available. If a particular independent variable is not a good predictor of the dependent variable, the CART tree will not use that variable at all to make a split. For this reason, a CART tree is often useful to help find the important variables when you have tens, or even hundreds, of independent variables available.
The splits used in a CART model divide the observations into smaller groups, called leaves or buckets. These groups are selected to be as homogenous or "pure" as possible, meaning that we want each of the groups to contain observations belonging to just one class. This is not always possible, since we might have two observations with exactly the same values of the independent variables, but belonging to different classes.

Even if it is possible to divide the observations into pure groups, it is typically not a good idea. It leads to overfitting of the training data, and the model will not extend well to data that it has never seen before. One popular method to prevent overfitting from happening is to set a minimum number of observations that must be in each final group of the tree. The choice of this parameter can often influence the accuracy of the model, and several different parameter choices should be tested. One way of doing this is with a technique called cross-validation.

For each bucket or leaf of the tree, the typical prediction is to predict the majority outcome of all of the observations in the training set that belong in that bucket. We can instead assign a probability to each observation of belonging to each class by computing the percentage of observations in the corresponding bucket that belong to that class. Then these probabilities can be thresholded to capture different error preferences, similarly to logistic regression (note that predicting the majority outcome is like using a threshold of 0.5).

CART can easily be used to predict an outcome with more than two classes, or a continuous outcome. With a continuous outcome, the prediction for each group is the average value of the dependent variable over all training points that were classified into that group. In a CART model, regardless of the outcome type, keep in mind that all observations that are classified into the same group, or bucket, will have the same prediction.

After building a cart model, a couple of steps should be taken to assess the validity and evaluate the performance of the tree:

- Plot the tree and look at each of the decision rules. Check to see if they make sense intuitively. Rules that seem strange could still be accurate, or it could be a sign that the data needs to be examined. This is also a good way to see which variables are the most important for the model, by seeing which ones were used in the tree.
- Assess the accuracy of the tree on a test set. The predictions should be compared to the actual values by using a classification matrix (for a categorical outcome) or by computing the R^2 value (for a continuous outcome).

Random Forests

The method of random forests was designed to improve the prediction accuracy of CART. It works by building a large number of CART trees which each "vote" on the outcome to make a final prediction. Unfortunately, this makes the method less interpretable than CART, so often you need to decide which you value more: the interpretability of the model, or attaining the maximum possible accuracy.

In a random forest, each tree is given a random subset of the independent variables from which it can select the splits, and a training set that is a bagged or bootstrapped sample of the full dataset. This means that the data used as the training data for each tree is selected randomly with replacement from the full dataset. As an example, if we have five data points in our training set, labeled {1,2,3,4,5} we might use the five observations {3,2,4,3,1} to build the first tree, the five observations {5,1,3,3,2} to build the second tree, etc. Notice that some observations are repeated, and some are not included in certain training sets. This gives each tree a slightly different training set with which to build the tree, and helps make the trees see different things in the data.

Just like in CART, there are some parameters that need to be selected for a random forest model. However, random forest models have been shown to be less sensitive to the parameters of the model, so as long as the parameters are set to reasonable values, the tree is most likely close to the optimal model.

Trees in R

We will use the packages "rpart", "rpart.plot" and "randomForest" to build tree models.

Suppose your training data set is called "TrainingData", your dependent variable is called "DependentVar", and you have two independent variables called "IndependentVar1" and "IndependentVar2". If your dependent variable is categorical or binary, you can build a CART model with the following command:

```
CartModel = rpart(DependentVar ~ IndependentVar1 + IndependentVar2, data=TrainingData,
method="class", minbucket=25)
```

The value selected for minbucket can be any number you choose, or a value selected through cross-validation. You can also instead set a value for the "cp" parameter in a similar way. If your dependent variable is continuous, you just need to remove the method="class" argument.

You can plot the tree with the prp() function:

```
prp(CartModel)
```

Suppose your test set is called "TestingData". You can make predictions on the test set using the predict() function:

```
TestPredictions = predict(CartModel, newdata=TestingData, type="class")
```

If you have a continuous dependent variable or you want probability predictions, you should remove the type="class" argument.

If you have a classification problem, you can generate an ROC curve for your model with the following commands:

```
TestProbabilities = predict(CartModel, newdata=TestingData)
PredROC = prediction(TestProbabilities[,2], TestingData$DependentVar)
PerfROC = performance(PredROC, "tpr", "fpr")
plot(PerfROC)
```

To build a random forest model, you can use the following command:

```
ForestModel = randomForest(DependentVar ~ IndependentVar1 + IndependentVar2, data=TrainingData)
```

Be sure your outcome variable is a factor variable if you have a classification problem, which can be done with the following command:

```
TrainingData$DependentVar = as.factor(TrainingData$DependentVar)
```

You can also change the parameter settings of the model by adding additional arguments. For example, you can use the following command to build a random forest model with 200 trees and at least 25 observations in each bucket:

```
ForestModel = randomForest(DependentVar ~ IndependentVar1 + IndependentVar2, data=TrainingData,
ntree=200, nodesize=25)
```

You can make predictions on your test set with the following command:

```
TestPredictions = predict(ForestModel, newdata=TestingData)
```