Building Predictive Models with the *caret* package

Dublin R

July 20, 2014

Contents

1 Introduction to the *caret* package

- The caret package, short for classi

  cation and regression training, contains numer- ous tools for developing predictive models using the rich set of models available in `R`.

- The package focuses on simplifying model training and tuning across a wide variety of modeling techniques.

- It also includes methods for pre-processing training data, calculating variable impor- tance, and model visualizations.

2 Classification and Regression Trees

C&RT, a recursive partitioning method, builds classification and regression trees for predicting continuous dependent variables (regression) and categorical predictor variables (classification). The classic C&RT algorithm was popularized by Breiman et al. *(Breiman, Friedman, Olshen, & Stone, 1984; see also Ripley, 1996).*

2.1 Advantages of C&RT Methods

As mentioned earlier, there are a large number of methods that an analyst can choose from when analyzing classification or regression problems. Tree classification techniques, when they "work" and produce accurate predictions or predicted classifications based on few logical if-then conditions, have a number of advantages over many of those alternative techniques.

Simplicity of results. In most cases, the interpretation of results summarized in a tree is very simple. This simplicity is useful not only for purposes of rapid classification of new observations (it is much easier to evaluate just one or two logical conditions, than to compute classification scores for each possible group, or predicted values, based on all predictors and using possibly some complex nonlinear model equations), but can also often yield a much simpler "model" for explaining why observations are classified or predicted in a particular manner (e.g., when analyzing business problems, it is much easier to present a few simple if-then statements to management, than some elaborate equations).

Tree methods are nonparametric and nonlinear. The final results of using tree methods for classification or regression can be summarized in a series of (usually few) logical if-then conditions (tree nodes). Therefore, there is no implicit assumption that the underlying relationships between the predictor variables and the dependent variable are linear, follow some specific non-linear link function], or that they are even monotonic in nature.

For example, some continuous outcome variable of interest could be positively related to a variable Income if the income is less than some certain amount, but negatively related if it is more than that amount (i.e., the tree could reveal multiple splits based on the same variable Income, revealing such a non-monotonic relationship between the variables).

Thus, tree methods are particularly well suited for data mining tasks, where there is often little a priori knowledge nor any coherent set of theories or predictions regarding which variables are related and how. In those types of data analyses, tree methods can often reveal simple relationships between just a few variables that could have easily gone unnoticed using other analytic techniques.

2.2 Workflow

Modeling in R generally follows the same workflow:

1. Create the model using the basic function:

   ```
   fit <- knn(trainingData, outcome, k = 5)
   ```

2. Assess the properties of the model using `print`, `plot`, `summary` or other methods

3. Predict outcomes for samples using the `predict` method:

   ```
   predict(fit, newSamples).
   ```

   The model can be used for prediction without changing the original model object.

3 Regression Trees

The general approach to derive predictions from few simple if-then conditions can be applied to regression problems as well. This example is based on the data file Poverty, which contains 1960 and 1970 Census figures for a random selection of 30 counties.

The research question (for that example) was to determine the correlates of poverty, that is, the variables that best predict the percent of families below the poverty line in a county. A reanalysis of those data, using the regression tree analysis [and v-fold cross-validation, yields the following results:

Again, the interpretation of these results is rather straightforward: Counties where the percent of households with a phone is greater than 72% have generally a lower poverty rate. The greatest poverty rate is evident in those counties that show less than (or equal to) 72% of households with a phone, and where the population change (from the 1960 census to the 170 census) is less than -8.3 (minus 8.3).

These results are straightforward, easily presented, and intuitively clear as well: There are some affluent counties (where most households have a telephone), and those generally have little poverty. Then there are counties that are generally less affluent, and among those the ones that shrunk most showed the greatest poverty rate.

A quick review of the scatterplot of observed vs. predicted values shows how the discrimination between the latter two groups is particularly well "explained" by the tree model.

4 The Confusion Matrix

The function `confusionMatrix` can be used to compute various summaries for classification models.

4.1 Assessing The Performance of a Predictive Model

caret has several functions that can be used to describe the performance of classification models.

For binary classifiation models, the functions `sensitivity`, `specificity`, `posPredValue` and `negPredValue` can be used to characterize performance.

By default, the first level of the outcome factor is used to define the "positive" result, although this can be changed.

4.2 Appraising Regression Models

- For regression models, performance is calculated using the *root mean squared error* and R^2 instead of accuracy and the Kappa statistic.
- However, there are many models where there is no notion of model degrees of freedom (such as random forests) or where there are more parameters than training set samples.
- Given this, `train` ignores degrees of freedom when computing performance values. For example, to compute R^2, the correlation coeficient is computed between the observed and predicted values and squared.
- When comparing models, the performance metrics will be on the same scale, but these metrics do not penalize model complexity (as *adjusted* R^2 does) and will tend to favor more complex fits over simpler models.

5 The ROC Curve

The ***aSAH*** dataset summarizes several clinical and one laboratory variable of 113 patients with an aneurysmal subarachnoid hemorrhage.

```
Xavier Robin, Natacha Turck, Alexandre Hainard, et al. (2011) \pROC: an open-source packa
```

```
> tail(aSAH)
    gos6 outcome gender age wfns s100b  ndka
136    5    Good Female  68    4  0.47 10.33
137    4    Good   Male  53    4  0.17 13.87
138    1    Poor   Male  58    5  0.44 15.89
139    5    Good Female  32    1  0.15 22.43
140    5    Good Female  39    1  0.50  6.79
141    5    Good   Male  34    1  0.48 13.45
```

-
-

```
library(pROC)
data(aSAH)
# Basic example with 2 roc objects
roc1 <- roc(aSAH$outcome, aSAH$s100b)
```

```
> roc1

Call:
roc.default(response = aSAH$outcome, predictor = aSAH$s100b)

Data: aSAH$s100b in 72 controls (aSAH$outcome Good) < 41 cases (aSAH$outcome Poor).
Area under the curve: 0.7314
```

6 Resampling Methods

7 Resampling Methods

The resampling methods used by **caret** are:

- bootstrapping,
- k-fold crossvalidation,
- leave-one-out cross-validation,
- leave-group-out cross-validation (i.e., repeated splits without replacement).

7.1 Avoiding Over-Fitting

A major issue that arises when applying regression or classification trees to "real" data with much random error noise concerns the decision when to stop splitting. For example, if you had a data set with 10 cases, and performed 9 splits (determined 9 if-then conditions), you could perfectly predict every single case. In general, if you only split a sufficient number of times, eventually you will be able to "predict" ("reproduce" would be the more appropriate term here) your original data (from which you determined the splits). Of course, it is far from clear whether such complex results (with many splits) will replicate in a sample of new observations; most likely they will not.

This general issue is also discussed in the literature on tree classification and regression methods, as well as neural networks, under the topic of **overlearning** or **overfitting**. If not stopped, the tree algorithm will ultimately "extract" all information from the data, including information that is not and cannot be predicted in the population with the current set of predictors, i.e., random or noise variation.

The general approach to addressing this issue is first to stop generating new split nodes when subsequent splits only result in very little overall improvement of the prediction. For example, if you can predict 90% of all cases correctly from 10 splits, and 90.1% of all cases from 11 splits, then it obviously makes little sense to add that 11th split to the tree. There are many such criteria for automatically stopping the splitting (tree-building) process.

7.1.1 Pruning

Once the tree building algorithm has stopped, it is always useful to further evaluate the quality of the prediction of the current tree in samples of observations that did not participate in the original computations. These methods are used to "prune back" the tree, i.e., to eventually (and ideally) select a simpler tree than the one obtained when the tree building algorithm stopped, but one that is equally as accurate for predicting or classifying "new" observations.

7.1.2 Crossvalidation

One approach is to apply the tree computed from one set of observations (learning sample) to another completely independent set of observations (testing sample). If most

or all of the splits determined by the analysis of the learning sample are essentially based on "random noise," then the prediction for the testing sample will be very poor. Hence one can infer that the selected tree is not very good (useful), and not of the "right size."

7.1.3 V-fold crossvalidation

Continuing further along this line of reasoning (described in the context of crossvalidation above), why not repeat the analysis many times over with different randomly drawn samples from the data, for every tree size starting at the root of the tree, and applying it to the prediction of observations from randomly selected testing samples. Then use (interpret, or accept as your final result) the tree that shows the best average accuracy for cross-validated predicted classifications or predicted values.

In most cases, this tree will not be the one with the most terminal nodes, i.e., the most complex tree. This method for pruning a tree, and for selecting a smaller tree from a sequence of trees, can be very powerful, and is particularly useful for smaller data sets. It is an essential step for generating useful (for prediction) tree models, and because it can be computationally difficult to do, this method is often not found in tree classification or regression software.