Categorical soil attribute modeling and mapping: C5 decision tree models.

Soil Security Laboratory

2018

1 C5 decision trees

The C5 decision tree model is available in the `C50` package. The function `C5.0` fits classification tree models or rule-based models using Quinlans's C5.0 algorithm (Quinlan, 1993).

Essentially we will go through the same process as we did for the multinomial logistic regression. The `C5.0` function and its internal parameters are similar in nature to that of the `Cubist` function for predicting continuous variables. The `trials` parameter lets you implement a "boosted" classification tree process, with the results aggregated at the end. There are also many other useful model tuning parameters in the `C5.0Control` parameter set too that are worth a look. Just see the help files for more information.

First lets prepare the data.

```
library(ithir)
library(sp)
library(raster)
library(rasterVis)

data(hvTerronDat)
data(hunterCovariates)
```

Transform the `hvTerronDat` data to a `SpatialPointsDataFrame`.

```
names(hvTerronDat)
```

```
## [1] "x"      "y"      "terron"
```

```
coordinates(hvTerronDat) <- ~x + y
```

As these data are of the same spatial projection as the `hunterCovariates`, there is no need to perform a coordinate transformation. So we can perform the intersection immediately.

```
DSM_data <- extract(hunterCovariates, hvTerronDat, sp = 1, method = "simple")
DSM_data <- as.data.frame(DSM_data)
str(DSM_data)
```

```
## 'data.frame': 1000 obs. of  8 variables:
##  $ x                : num  346535 334760 340910 336460 344510 ...
##  $ y                : num  6371941 6375841 6377691 6382041 6378116 ...
##  $ terron           : Factor w/ 12 levels "1","2","3","4",..: 3 4 12 5 6 11 3 10 3 5 ..
##  $ AACN             : num  37.544 25.564 32.865 0.605 9.516 ...
##  $ Drainage.Index   : num  4.72 4.78 2 4.19 4.68 ...
##  $ Light.Insolation : num  1690 1736 1712 1712 1677 ...
##  $ TWI              : num  11.5 13.8 13.4 18.6 19.8 ...
##  $ Gamma.Total.Count: num  380 407 384 388 454 ...
```

It is always good practice to check to see if any of the observational data returned any NA values for any one of the covariates. If there is NA values, it indicates that the observational data is outside the extent of the covariate layers. It is best to remove these observations from the data set.

```
which(!complete.cases(DSM_data))
```

```
## integer(0)
```

```
DSM_data <- DSM_data[complete.cases(DSM_data), ]
```

Now we perform a random selection of data to hold back from the model fitting process.

```
set.seed(655)
training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
```

With this, the modeling can begin.

```
library(C50)
hv.C5 <- C5.0(x = DSM_data[training, c("AACN", "Drainage.Index", "Light.Insolation",
    "TWI", "Gamma.Total.Count")], y = DSM_data$terron[training], trials = 1,
    rules = FALSE, control = C5.0Control(CF = 0.95, minCases = 20,
    earlyStopping = FALSE))
```

By calling the `summary` function, some useful model fit diagnostics are given. These include the tree structure, together with the covariates used and mis-classification error of the model, as well as a rudimentary confusion matrix. A useful feature of the C5 model is that it can omit in an automated fashion, unnecessary covariates.

The `predict` function can either return the predicted class or the confidence of the predictions (which is controlled using the `type=‘‘prob’’` parameter). The probabilities are quantified such that if an observation is classified by a single leaf of a decision tree, the confidence value is the proportion of training cases at that leaf that belong to the predicted class. If more than one leaf is involved (i.e., one or more of the attributes on the path has missing values), the value is a weighted sum of the individual leaves' confidences. For rule-classifiers, each applicable rule votes for a class with the voting weight being the rule's confidence value. If the sum of the votes for class C is W(C), then the predicted class P is chosen so that W(P) is maximal, and the confidence is the greater of (1), the voting weight of the most specific applicable rule for predicted class P; or (2) the average vote for class P (so,

W(P) divided by the number of applicable rules for class P).Boosted classifiers are similar, but individual classifiers vote for a class with weight equal to their confidence value. Overall, the confidence associated with each class for every observation are made to sum to 1.

```
# return the class predictions
predict(hv.C5, newdata = DSM_data[training, ])

# return the class probabilities
predict(hv.C5, newdata = DSM_data[training, ], type = "prob")
```

So lets look at the calibration and validation statistics. First, the calibration statistics:

```
C.pred.hv.C5 <- predict(hv.C5, newdata = DSM_data[training, ])
goofcat(observed = DSM_data$terron[training], predicted = C.pred.hv.C5)
```

```
## $confusion_matrix
##     1 2  3  4  5  6  7  8  9 10 11 12
## 1  17 9  4  4  0  0  0  0  1  0  0  1
## 2   0 0  0  0  0  0  0  0  0  0  0  0
## 3   0 0 42  4  0  0 13  0  4  3  7 14
## 4   2 0 12 37  0  1  5  6  9  1  9  1
## 5   0 0  0  0 48  2  1  9 14 18  0  0
## 6   0 0  0  0 22 55  0  3  2  1  1  0
## 7   0 0  7  6  7  0 95 18  9 12  5 14
## 8   0 0  0  4 12  8  2 61  3  4  9  2
## 9   0 0  0  0  0  0  0  0  0  0  0  0
## 10  0 0  0  0  5  0  8  6  0 14  7  0
## 11  0 0  0  0  0  0  0  0  0  0  0  0
## 12  0 0  0  0  0  0  0  0  0  0  0  0
##
## $overall_accuracy
## [1] 53
##
## $producers_accuracy
##  1  2  3  4  5  6  7  8  9 10 11 12
## 90  0 65 68 52 84 77 60  0 27  0  0
##
## $users_accuracy
##   1   2   3   4   5   6   7   8   9  10  11  12
##  48 NaN  49  45  53  66  55  59 NaN  35 NaN NaN
##
## $kappa
## [1] 0.4618099
```

It will be noticed that some of the Terron classes failed to be predicted by the fitted model. For example Terron classes 2, 9, 11, and 12 were all predicted as being a different class. All observations of Terron class 2 were predicted as Terron class 1. Doing the external validation we return the following:

```
V.pred.hv.C5 <- predict(hv.C5, newdata = DSM_data[-training, ])
goofcat(observed = DSM_data$terron[-training], predicted = V.pred.hv.C5)
```

```
## $confusion_matrix
##    1 2  3 4  5  6  7  8 9 10 11 12
## 1  7 3  5 1  0  0  1  0 2  1  0  1
## 2  0 0  0 0  0  0  0  0 0  0  0  0
## 3  0 0 14 1  0  0  4  0 2  2  2  6
## 4  1 1  5 9  0  0  4  3 7  0  0  0
## 5  0 0  0 0 20  2  0  7 5  2  0  1
## 6  0 0  0 0 10 22  0  7 0  1  0  0
## 7  0 0 10 5  0  0 35 11 2  7  5  3
## 8  0 0  0 3  6  2  4 16 2  5  2  2
## 9  0 0  0 0  0  0  0  0 0  0  0  0
## 10 0 0  0 1  2  0  3  3 4  7  1  0
## 11 0 0  0 0  0  0  0  0 0  0  0  0
## 12 0 0  0 0  0  0  0  0 0  0  0  0
##
## $overall_accuracy
## [1] 44
##
## $producers_accuracy
##  1  2  3  4  5  6  7  8  9 10 11 12
## 88  0 42 45 53 85 69 35  0 29  0  0
##
## $users_accuracy
##   1   2   3   4   5   6   7   8   9  10  11  12
##  34 NaN  46  30  55  56  45  39 NaN  34 NaN NaN
##
## $kappa
## [1] 0.3565075
```

And finally, creating the map derived from the `hv.C5` model using the raster `predict` function (Figure 1). Note that the C5 model returned 0% producer's accuracy for Terron classes 2, 9, 11 and 12. These data account for only a small proportion of the data set, and/or, they may be similar to other existing Terron classes (based on the available predictive covariates). Consequently, they did not feature in the `hv.C5` model and ultimately, the final map.

```
# class prediction
map.C5.c <- predict(hunterCovariates, hv.C5, type = "class",
filename = "hv_C5_class.tif",
    format = "GTiff", overwrite = T, datatype = "INT2S")

map.C5.c <- as.factor(map.C5.c)
rat <- levels(map.C5.c)[[1]]
rat[["terron"]] <- c("HVT_001", "HVT_003", "HVT_004", "HVT_005", "HVT_006",
    "HVT_007", "HVT_008", "HVT_010")
levels(map.C5.c) <- rat
```

```
# plot
area_colors <- c("#FF0000", "#73DFFF", "#FFEBAF", "#A8A800", "#0070FF", "#FFA77F",
    "#7AF5CA", "#CCCCCC")
levelplot(map.C5.c, col.regions = area_colors, xlab = "", ylab = "")
```

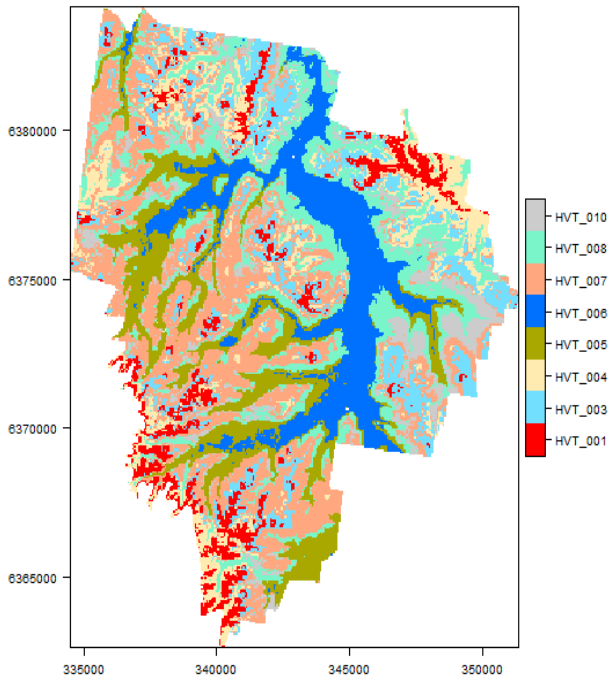

Figure 1: Hunter Valley Terron class map created using C5 decision tree model.

References

Quinlan, J. R.
1993. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, California.