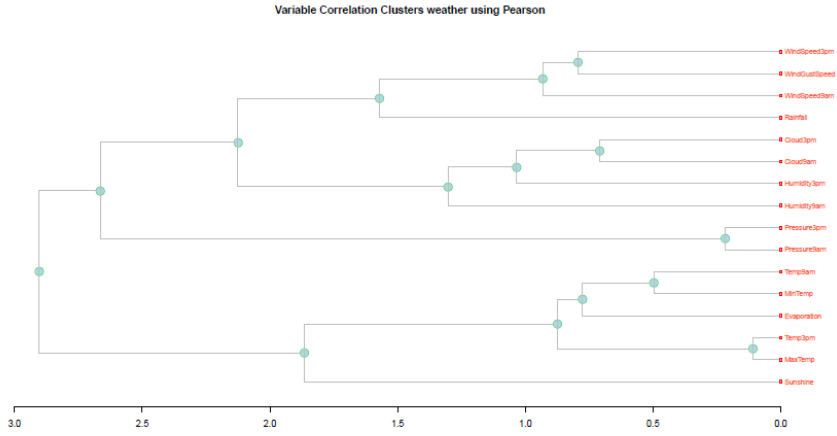
REVOLUTION ANALYTICS

# Modeling in Revolution R Enterprise
# Module 7: Decision Trees

July 23, 2014

# Decision Trees

Decision trees are effective algorithms widely used for classification and regression. More specifically, they show the potential consequences of decisions and display them in a convenient tree-like graph.



# Decision Trees

Building a decision tree generally requires that all continuous variables be sorted in order to decide where to split the data. To main techniques to overcome this obstacle exist:

1. Performing data pre-sorting, or
2. Using approximate summary statistic of the data

Pre-sorting techniques often cannot accommodate very large data sets. Big data decision trees are normally parallelized to enable large scale learning: task parallelism builds different tree nodes on different processors, or in Hadoop, on different compute nodes.

# Tree Modeling Algorithm

The underlying algorithm is based on recursive partitioning:

- Start with all the data at the root of the tree
- Partition the data set according to some criterion of “best” partition
- Do the same for each of the two new subsets
- Once a partition is made, stick with it (greedy approach)
- Repeat until desired depth has been reached

# rxDTree

The rxDTree function fits a classification or regression tree by specifying a model function and data. The function is especially designed for handling large data sets, and in particular is an approximate decision tree algorithm with horizontal parallelism (i.e. partitions the data horizontally for parallelization).

```
rxDTree(formula, data, ...)
```

The algorithm works by using histograms as the approximate compact representation of the data and builds the decision tree in a breadth-first fashion. The algorithm is executed in parallel, where each processor receives a subset of data, yet still has a view of the complete tree built so far.

# rxDTree

- The user may control the balance between time complexity and prediction accuracy when using rxDTree by specifying the maximum number of bins for the histogram.

In other words, you can set the number of bins in the histograms to control the trade-off between accuracy and speed:

- A large number of bins allows a more accurate description of the data and thus more accurate results
- A small number of bins reduces time complexity and memory usage.

# Example: Tree Modeling

Using the Bank data again, let's construct a tree model to observe whether the client owns a home based on the variables age and balance.

Let's also speed up the processing time by requiring a smaller number of bins. One way to accomplish this is using the cp subcall, or the complexity parameter. The cp is a numeric scalar, requiring that any split that does not decrease the overall lack-of-fit by at least the value of cp is not attempted.

The output is automatically generated:

```
infile <- file.path("data", "BankXDF.xdf")
BankDS <- RxXdfData(file = infile)

Tree <- rxDTree(housing ~ age + balance, data = BankDS, cp = 0.01)
```

# Example: Tree Modeling

The output is…

```
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.002 seconds
## Rows Read: 10000, Total Rows Processed: 20000, Total Chunk Time: 0.004 seconds
## Rows Read: 10000, Total Rows Processed: 30000, Total Chunk Time: 0.005 seconds
## Rows Read: 10000, Total Rows Processed: 40000, Total Chunk Time: 0.005 seconds
## Rows Read: 5211, Total Rows Processed: 45211, Total Chunk Time: 0.005 seconds
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.002 seconds
...
```

```
## Call:
## rxDTree(formula = housing ~ age + balance, data = BankDS, cp = 0.01)
## File: D:\Github\Legacy_Course_Materials\modules\Modeling\Decision_Trees_with_RRE\doc\data\BankXDF.xdf
## Number of valid observations:  45211
## Number of missing observations:  0
##
...
```

Note that no classification was provided for balance given the complexity parameter we set.

# Example: Tree Modeling

```
library(RevoTreeView)
plot(createTreeView(Tree))
```

# Example: Interpretation

The first node encompasses all observations. Since the majority of bank clients in our data own homes, the decision tree's first attempt at classification places every client in the "owns a home" category.

- However, we can see that this method incorrectly classifies 20,081 clients in our sample (approximately 44-percent of our data). Accordingly, only 56-percent of our clients are correctly classified.

# Example: Interpretation

The second node splits the home classification based on age. In this case, clients with ages less than 54.5 years are classified as owning a home, whereas clients with ages greater than 54.5 years are classified as not owning a home. This classification yields:

- For the clients greater than or equal to 54.5 years of age, 1758 are incorrectly classified (approximately 31-percent), while 3948 are correctly classified (about 69-percent).
- For clients less than 54.5 years of age, 16,133 are incorrectly classified (approximately 41-percent), while 23,372 are correctly classified (about 59-percent).

Obviously the second iteration yields a slight improvement over the first, with both 31-percent and 41-percent being lower than the original incorrectly classified rate of 44-percent.

# Exercise: Tree Modeling

Using the Bank data, construct a tree model to observe whether or not a client has purchased a term deposit(y), based on the variables age and duration (i.e. duration of the advertising phone call, in seconds). Also, let's speed up the computation by requiring a complexity parameter equaling 0.01. Given an interpretation for each node of the result.

# Exercise: Solution

Constructing the tree model, and analyzing the results…

```
Tree2 <- rxDTree(y ~ age + duration, data = BankDS, cp = 0.01)
```

# Exercise: Solution

```
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.002 seconds
## Rows Read: 10000, Total Rows Processed: 20000, Total Chunk Time: 0.005 seconds
## Rows Read: 10000, Total Rows Processed: 30000, Total Chunk Time: 0.005 seconds
## Rows Read: 10000, Total Rows Processed: 40000, Total Chunk Time: 0.005 seconds
## Rows Read: 5211, Total Rows Processed: 45211, Total Chunk Time: 0.005 seconds
## Rows Read: 10000, Total Rows Processed: 10000, Total Chunk Time: 0.002 seconds
...
```

# Example: Tree Modeling

```
plot(createTreeView(Tree2))
```

# Exercise: Solution

For our interpretation, given the first node:

- All clients are classified as not subscribing to a term deposit.
  - 5289 are incorrectly classified (approximately 12-percent)
  - 39922 are correctly classified (approximately 88-percent)

Given the second node, there is no change in the classification rate:

- Clients on the advertising call for less than 520 seconds are classified as not subscribing to a term deposit.
  - 3128 are incorrectly classified (approximately 8-percent)
- Clients on the advertising call for greater than or equal to 520 seconds are still classified as not subscribing to a term deposit.
  - 2161 are incorrectly classified (approximately 44-percent)

# Exercise: Solution

Given the third node, a change is made to the clients on the advertising phone call for greater than or equal to 520 seconds:

- Clients on the advertising call for less than 787 seconds (but still greater than or equal to 520 seconds) are classified as not subscribing to a term deposit.
  - 1068 are incorrectly classified (approximately 36-percent)
- Clients on the advertising call for greater than or equal to 787 seconds are classified as subscribing to a term deposit.
  - 882 are incorrectly classified (approximately 43-percent)

So, as one may expect, clients on the advertising call for an extended duration (a little more than 13 minutes) are classified as subscribing to a term deposit, where as clients on class shorter than that are less interested and usually do not subscribe according to the above classification.

# Recap

Let's review some of the concepts covered in this module:

- What is a Decision Tree?
- How do you interpret its output?
- What is the complexity parameter?

REVOLUTION ANALYTICS

# Thank you

**Revolution Analytics is the leading commercial provider of software and support for the popular open source R statistics language.**

**www.revolutionanalytics.com, 1.855.GET.REVO, Twitter: @RevolutionR**