Predictive Modeling with Random Forests™ in R

A Practical Introduction to R for Business Analysts

by Jim Porzak

Loyalty Matrix

January 2007
Outline

• Part I – Introduction to R

• Part II – Using Random Forests for Classification

• Wrap up & Questions/Discussion

Note: For R setup details see first Appendix slide.
Background on Loyalty Matrix

- Provide customer data analytics to optimize direct marketing resources
- OnDemand platform MatrixOptimizer® (version 3.2)
- Over 20 engagements with Fortune 500 clients
- Experienced team with diverse skills & backgrounds
- 10-person San Francisco firm with an offshore team in Nepal
Part I – Introduction to R

- **Background**
  - Evolution & History
  - Current state of R
  - Resources
    - R-Help
    - Task Views

- **Simple code example**
  - Functions
  - Objects
  - Methods

- **Where to learn more**
Evolution of R from S

• R is the free (GNU), open source, version of S
  – S developed by John Chambers *et al* while at Bell Labs in 80’s
  – For “data analysis and graphics” (with statistics emphasis)
  – Ver.4 defined by the “Green Book” *Programming with Data*, 1998
  – “S-Plus” now owned by Insightful Corp., Seattle, WA

• R was initially written in early 1990’s
  – by Robert Gentleman and Ross Ihaka
  – Statistics Department of the University of Auckland
  – GNU GPL release in 1995
  – “R” is before “S”, as in “HAL” is before “IBM”

• Since 1997 a core group of ± 20 developers
  – Initial V1.0 released in February, 2000
  – Continually developed with a new 0.1 level release ~ 6 months
Current state of R

- V2.4.1 Released December, 2006
- Windows, Mac OS, Linux & Unix ports
- Now 931 submitted packages from “aaMI” to “zoo”
- 18th newsletter (Volume 6/5) published December 2006
- The second useR! conference– Vienna June 2006
- Dozens of texts specifically on R or using R examples
- R language generally accepted to be more powerful than S-Plus
- Some interesting GUI work in progress - JGR
R Resources

• R Homepage: www.r-project.org
  – The official site of R

• R Foundation: www.r-project.org/foundation
  – Central reference point for R development community
  – Holds copyright of R software and documentation

• Local CRAN:
  – Mirror site
    • I use: cran.cnr.berkeley.edu/
    • Find your’s at: cran.r-project.org/mirrors.html
  – Current Binaries
  – Current Documentation & FAQs
  – Links to related projects and sites

• Mailing Lists
  – Best help ever!
R-Help Mailing List Example

Core Developers!
# CRAN Task Views

*Quick start guides to packages by task at hand*

<table>
<thead>
<tr>
<th>Task View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bayesian</strong></td>
<td>Bayesian Inference</td>
</tr>
<tr>
<td><strong>Cluster</strong></td>
<td>Cluster Analysis &amp; Finite Mixture Models</td>
</tr>
<tr>
<td><strong>Econometrics</strong></td>
<td>Computational Econometrics</td>
</tr>
<tr>
<td><strong>Environmetrics</strong></td>
<td>Analysis of ecological and environmental data</td>
</tr>
<tr>
<td><strong>Finance</strong></td>
<td>Empirical Finance</td>
</tr>
<tr>
<td><strong>Genetics</strong></td>
<td>Statistical Genetics</td>
</tr>
<tr>
<td><strong>Graphics</strong></td>
<td><strong>Graphic Displays &amp; Dynamic Graphics &amp; Graphic Devices &amp; Visualization</strong></td>
</tr>
<tr>
<td><strong>MachineLearning</strong></td>
<td>Machine Learning &amp; Statistical Learning</td>
</tr>
<tr>
<td><strong>Multivariate</strong></td>
<td>Multivariate Statistics</td>
</tr>
<tr>
<td><strong>SocialSciences</strong></td>
<td>Statistics for the Social Sciences</td>
</tr>
<tr>
<td><strong>Spatial</strong></td>
<td>Analysis of Spatial Data</td>
</tr>
<tr>
<td><strong>gR</strong></td>
<td>gRaphical models in R</td>
</tr>
</tbody>
</table>

Link: cran.cnr.berkeley.edu/src/contrib/Views/
A useful function for DMers...

```r
> prop.test(c(138, 113), c(2500, 2500))
```

2-sample test for equality of proportions with continuity correction

data:  c(138, 113) out of c(2500, 2500)
X-squared = 2.4161, df = 1, p-value = 0.1201
alternative hypothesis: two.sided
95 percent confidence interval:
-0.002501721  0.022501721
sample estimates:
prop 1  prop 2
0.0552  0.0452
R Basics - Objects

• Most functions return an object

```r
pt <- prop.test(c(138, 113), c(2500, 2500))
> str(pt)

List of 9
$ statistic : Named num 2.42
 ..- attr(*, "names")= chr "X-squared"
$ parameter : Named num 1
 ..- attr(*, "names")= chr "df"
$ p.value  : num 0.12
$ estimate : Named num [1:2] 0.0552 0.0452
 ..- attr(*, "names")= chr [1:2] "prop 1" "prop 2"
$ null.value : NULL
$ conf.int : atomic [1:2] -0.0025  0.0225
 ..- attr(*, "conf.level")= num 0.95
$ alternative: chr "two.sided"
$ method   : chr "2-sample test for equality of proportions with continuity correction"
$ data.name : chr "c(138, 113) out of c(2500, 2500)"
- attr(*, "class")= chr "htest"
```
Objects have methods...
One of which we have used already – the default

```r
print(pt)
```

2-sample test for equality of proportions with continuity correction
data:  c(138, 113) out of c(2500, 2500)
X-squared = 2.4161, df = 1, p-value = 0.1201
alternative hypothesis: two.sided
95 percent confidence interval:
-0.002501721  0.022501721
sample estimates:
  prop 1  prop 2
  0.0552  0.0452
R Basics – Learning More

• Wikipedia

• An Introduction to R
  – http://cran.cnr.berkeley.edu/doc/manuals/R-intro.html

• Links to all “official” manuals (htlm & pdf)
  – http://cran.cnr.berkeley.edu/manuals.html

• R Graph Gallery
  – http://addictedtor.free.fr/graphiques/

• R Wiki
  – http://wiki.r-project.org/rwiki/doku.php
Part II – Random Forests

• Background
  – History
  – Advantages
  – Versions

• Example walkthrough using R
  – Problem & Data Descriptions
  – Data Prep
  – Building the Forest
  – Diagnostics
  – Interpretation
  – Prediction
  – Scoring

• And More…
Random Forests - History

- Developed by Leo Breiman of Cal Berkeley, one of the four developers of CART, and Adele Cutler, now at Utah State University.
- An extension of single decision tree methods like CART & CHAID.
- Many small trees are randomly grown to build the forest. All are used in the final result.
- See Wikipedia for more
Random Forests - Advantages

- Accuracy comparable with modern machine learning methods. (SVMs, neural nets, Adaboost)
- Built in cross-validation using “Out of Bag” data. (Prediction error estimate is a by product)
- Large number candidate predictors are automatically selected. (Resistant to over training)
- Continuous and/or categorical predicting & response variables. (Easy to set up.)
- Can be run in unsupervised for cluster discovery. (Useful for market segmentation, etc.)
- Free Prediction and Scoring engines run on PC’s, Unix/Linux & Mac’s. (R version)
Random Forests - Versions

• Original Fortran 77 source code freely available from Breiman & Cutler.
  http://www.stat.berkeley.edu/users/breiman/RandomForests/cc_home.htm
  http://www.math.usu.edu/~adele/forests/

• Commercialization by Salford Systems.
  http://www.salford-systems.com/randomforests.php

• R package, randomForest. An adaptation by Andy Liaw of Merck.
  http://cran.cnr.berkeley.edu/src/contrib/Descriptions/randomForest.html
• Sample Data from a sports club
• Challenge – predict “at-risk” members based on membership usage data & simple demographics
• Training & Test data sets provided:
  – MemberTrainingSet.txt (1916 records)
  – MemberTestSet.txt (1901 records)
### RF Example – Columns

- **MembID** (identifier)
- **Status = M or C**
  
  (Member or Cancel)
- **Gender**
- **Age**
- **MembDays**
- **NumUses1st30d**
- **NumUsesLast30d**
- **TotalUses**
- **FirstCkInDay**

- **LastCkInDay**
- **DaysSinceLastUse**
- **TotalPaid**
- **MonthlyAmt**
- **MilesToClub**
- **NumExtras1st30d**
- **NumExtrasLast30d**
- **TotalExtras**
- **DaysSinceLastExtra**
RF Example – Getting Started

• Load the randomForest package

```r
> ## CIwR_rf.R
> require(randomForest)
```

Loading required package: randomForest
randomForest 4.5-18
Type rfNews() to see new features/changes/bug fixes.
[1] TRUE

• Point to working environment

```r
> setwd("c:/Projects/CIwR/R")
> dir("Data")
```

[1] "CruiseReservationEvents.txt"  "KeyCustomers.txt"
[3] "MemberTestSet.txt"            "MemberTrainingSet.txt"
[5] "NewSubscribers.txt"           "orders.txt"
[7] "ZipPopDist.txt"
```r
> Members <- read.delim("Data/MemberTrainingSet.txt", row.names = "MembID")
> str(Members)

'data.frame': 1916 obs. of 17 variables:
$ Status : Factor w/ 2 levels "C","M": 1 1 1 1 1 1 1 1 1 1 ... 
$ Gender : Factor w/ 3 levels "F","M","U": 2 2 1 2 2 1 1 2 ... 
$ Age : int 21 18 21 21 45 25 21 20 35 15 ... 
$ MembDays : int 92 98 30 92 31 249 1 92 322 237 ... 
$ NumUses1st30d : int 11 11 3 6 24 2 0 16 12 6 ... 
$ NumUsesLast30d : int 6 6 3 1 24 0 0 4 0 0 ... 
$ TotalUses : int 28 31 3 9 24 6 0 30 38 26 ... 
$ FirstCkInDay : Factor w/ 556 levels "","2004-01-04",...: 132 264 ... 
$ LastCkInDay : Factor w/ 489 levels "","2004-01-15",...: 134 356 ... 
$ DaysSinceLastUse : int 3 2 9 11 4 196 NA 12 138 65 ... 
$ TotalPaid : int 149 136 100 129 75 134 138 149 582 168 ... 
$ MonthlyAmt : int NA 27 NA NA 31 30 NA NA 10 ... 
$ MilesToClub : int 4 0 0 5 2593 4 5 4 NA 2 ... 
$ NumExtras1st30d : int 0 0 0 0 0 0 0 0 1 0 ... 
$ NumExtrasLast30d : int 0 0 0 0 0 0 0 0 0 0 ... 
$ TotalExtras : int 0 0 0 0 0 0 0 0 6 0 ... 
$ DaysSinceLastExtra: int NA NA NA NA NA NA NA NA 253 NA ... 
```
RF Example – Quick Look at Data (1 of 2)

```r
> summary(Members)
```

<table>
<thead>
<tr>
<th>Status</th>
<th>Gender</th>
<th>Age</th>
<th>MembDays</th>
<th>NumUses1st30d</th>
</tr>
</thead>
<tbody>
<tr>
<td>C: 809</td>
<td>F: 870</td>
<td>Min. : 13.00</td>
<td>Min. : 1.0</td>
<td>Min. : 0.000</td>
</tr>
<tr>
<td>M: 1107</td>
<td>M: 832</td>
<td>1st Qu.: 23.00</td>
<td>1st Qu.: 92.0</td>
<td>1st Qu.: 1.000</td>
</tr>
<tr>
<td>U: 214</td>
<td>Median: 29.00</td>
<td>Median: 220.0</td>
<td>Median: 4.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean: 32.72</td>
<td>Mean: 247.8</td>
<td>Mean: 5.385</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3rd Qu.: 40.00</td>
<td>3rd Qu.: 365.0</td>
<td>3rd Qu.: 8.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Max. : 82.00</td>
<td>Max. : 668.0</td>
<td>Max. : 36.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NA's : 1.00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NumUsesLast30d</th>
<th>TotalUses</th>
<th>FirstCkInDay</th>
<th>LastCkInDay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. : 0.000</td>
<td>Min. : 0.00</td>
<td>2004-06-01: 10</td>
<td>2005-10-28: 56</td>
</tr>
<tr>
<td>1st Qu.: 0.000</td>
<td>1st Qu.: 3.00</td>
<td>2004-06-23: 10</td>
<td>2005-10-27: 55</td>
</tr>
<tr>
<td>Median : 0.000</td>
<td>Median : 12.00</td>
<td>2004-11-01: 10</td>
<td>2005-10-30: 52</td>
</tr>
<tr>
<td>Mean : 2.125</td>
<td>Mean : 26.73</td>
<td>2005-02-02: 10</td>
<td>2005-10-26: 47</td>
</tr>
<tr>
<td>3rd Qu.: 3.000</td>
<td>3rd Qu.: 33.00</td>
<td>2005-09-13: 9</td>
<td>2005-10-29: 42</td>
</tr>
<tr>
<td>Max. :26.000</td>
<td>Max. :340.00</td>
<td>(Other) :1631</td>
<td>(Other) :1428</td>
</tr>
</tbody>
</table>

*Continued on next slide…*
RF Example – Quick Look at Data (2 of 2)

```
... Continued from above

<table>
<thead>
<tr>
<th>DaysSinceLastUse</th>
<th>TotalPaid</th>
<th>MonthlyAmt</th>
<th>MilesToClub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>1.00</td>
<td>Min.</td>
<td>4.00</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>7.00</td>
<td>1st Qu.</td>
<td>21.00</td>
</tr>
<tr>
<td>Median</td>
<td>32.00</td>
<td>Median</td>
<td>28.00</td>
</tr>
<tr>
<td>Mean</td>
<td>75.51</td>
<td>Mean</td>
<td>28.50</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>106.00</td>
<td>3rd Qu.</td>
<td>35.00</td>
</tr>
<tr>
<td>Max.</td>
<td>624.00</td>
<td>Max.</td>
<td>94.00</td>
</tr>
<tr>
<td>NA's</td>
<td>236.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NumExtras1st30d</th>
<th>NumExtrasLast30d</th>
<th>TotalExtras</th>
<th>DaysSinceLastExtra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>0.0000</td>
<td>Min.</td>
<td>0.0000</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>0.0000</td>
<td>1st Qu.</td>
<td>0.0000</td>
</tr>
<tr>
<td>Median</td>
<td>0.0000</td>
<td>Median</td>
<td>0.0000</td>
</tr>
<tr>
<td>Mean</td>
<td>0.4128</td>
<td>Mean</td>
<td>1.324</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>0.0000</td>
<td>3rd Qu.</td>
<td>0.0000</td>
</tr>
<tr>
<td>Max.</td>
<td>13.0000</td>
<td>Max.</td>
<td>121.0000</td>
</tr>
<tr>
<td>NA's</td>
<td>1646.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

- Absolute Dates not useful (at least down to day level)
- RF does not like NA’s!
  - Day’s Since Last xxx is NA when no event, use large # days
  - Impute remaining NA’s
RF Example – Prepping the data set

- Subset out the absolute dates:
  ```r
  > Members <- subset(Members, select = -c(FirstCkInDay, LastCkInDay))
  ```

- Replace days since last NA’s with 999:
  ```r
  > Members$DaysSinceLastUse[is.na(Members$DaysSinceLastUse)] <- 999
  > Members$DaysSinceLastExtra[is.na(Members$DaysSinceLastExtra)] <- 999
  ```

- Impute remaining NA’s with Random Forests’ impute:
  ```r
  > Members <- rfImpute(Status ~ ., data = Members)          ## 70 sec
  ```

```
ntree  OOB  1  2
  300: 21.82% 31.64% 14.63%
ntree  OOB  1  2
  300: 22.44% 33.13% 14.63%
ntree  OOB  1  2
  300: 21.76% 31.89% 14.36%
ntree  OOB  1  2
  300: 21.45% 32.14% 13.64%
ntree  OOB  1  2
  300: 20.72% 31.64% 12.74%
```
RF Example – One Last Look at Data & Save It

```r
> summary(Members)

<table>
<thead>
<tr>
<th>Status</th>
<th>Gender</th>
<th>Age</th>
<th>MembDays</th>
<th>NumUses1st30d</th>
</tr>
</thead>
<tbody>
<tr>
<td>C: 809</td>
<td>F: 870</td>
<td>Min. : 13.00</td>
<td>Min. : 1.0</td>
<td>Min. : 0.000</td>
</tr>
<tr>
<td>M: 1107</td>
<td>M: 832</td>
<td>1st Qu.: 23.00</td>
<td>1st Qu.: 92.0</td>
<td>1st Qu.: 1.000</td>
</tr>
<tr>
<td>U: 214</td>
<td>Median : 29.00</td>
<td>Median : 220.0</td>
<td>Median : 4.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean : 32.71</td>
<td>Mean : 247.8</td>
<td>Mean : 5.385</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3rd Qu.: 40.00</td>
<td>3rd Qu.: 365.0</td>
<td>3rd Qu.: 8.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Max. : 82.00</td>
<td>Max. : 668.0</td>
<td>Max. : 36.000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NumUsesLast30d</th>
<th>TotalUses</th>
<th>DaysSinceLastUse</th>
<th>TotalPaid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. : 0.000</td>
<td>Min. : 0.00</td>
<td>Min. : 1.0</td>
<td>Min. : 0.00</td>
</tr>
<tr>
<td>1st Qu.: 0.000</td>
<td>1st Qu.: 3.00</td>
<td>1st Qu.: 9.0</td>
<td>1st Qu.: 70.75</td>
</tr>
<tr>
<td>Median : 0.000</td>
<td>Median : 12.00</td>
<td>Median : 47.0</td>
<td>Median : 135.00</td>
</tr>
<tr>
<td>Mean : 2.125</td>
<td>Mean : 26.73</td>
<td>Mean : 189.3</td>
<td>Mean : 188.75</td>
</tr>
<tr>
<td>3rd Qu.: 3.000</td>
<td>3rd Qu.: 33.00</td>
<td>3rd Qu.: 172.0</td>
<td>3rd Qu.: 232.25</td>
</tr>
<tr>
<td>Max. : 26.000</td>
<td>Max. : 340.00</td>
<td>Max. : 999.0</td>
<td>Max. : 961.00</td>
</tr>
</tbody>
</table>

... more cut ...

> save(Members, file = "MemberTrainingSetImputed.rda")
RF Example – Building the Forest!

```r
> Members.rf <- randomForest(Status ~ ., data = Members, importance = TRUE, proximity = TRUE) # 30 sec
> Members.rf

Call:
  randomForest(x = Members[-1], y = Members$Status, ntree = 500, 
  mtry = 3, importance = TRUE, proximity = TRUE, data = Members)

Type of random forest: classification
  Number of trees: 500
  No. of variables tried at each split: 3

  OOB estimate of error rate: 21.4%

Confusion matrix:
   C  M class.error
  C 546 263 0.3250927
  M 147 960 0.1327913
```

- Rather good results. Only ~20% overall error rate.
  - 33% false positive
  - 13% false negative
RF Example – Tuning the Forest

- ntree = 500 & mtry = 3 are defaults. Try tuning them.

```r
> Members.rf <- randomForest(Members[-1], Members$Status, data = Members,
mtry = 4, ntree = 1000, importance = TRUE, proximity = TRUE)     ## 50 sec
> Members.rf
```

Call:
randomForest(x = Members[-1], y = Members$Status, ntree = 1000, mtry = 4,
importance = TRUE, proximity = TRUE, data = Members)

Type of random forest: classification
Number of trees: 1000
No. of variables tried at each split: 4

OOB estimate of error rate: 21.14%
Confusion matrix:

```
C     M class.error
C 556 253   0.3127318
M 152 955   0.1373080
```

- No real difference (probably within random effects)
RF Example – Diagnostics (1 of 3)

- RF Diagnostics - OOB errors by # trees
  > Plot(Members.rf, lty = 1)
RF Example – Diagnostics (2 of 3)

- RF Diagnostics – Margin Plot
  
  ```r
  > plot(margin(Members.rf, Members$Status))
  ```
RF Example – Diagnostics (3 of 3)

- RF Diagnostics – MDS Plot
  
  ```
  > MDSplot(Members.rf, Members$Status, k = 3)
  ```

![MDS Plot](image)
RF Example – Interpretation (1 of 5)

• Variable Importance Plot

```R
> varImpPlot(Members.rf)
```

Members.rf
RF Example – Interpretation (2 of 5)

- RF Diagnostics – Partial Dependence 1
  - `partialPlot(Members.rf, Members[-1], MembDays)`
  - `abline(h=0, col = "blue")`
RF Example – Interpretation (3 of 5)

• RF Diagnostics – Partial Dependence 2
  - `partialPlot(Members.rf, Members[-1], DaysSinceLastUse)`
  - `abline(h=0, col = "blue")`
RF Example – Interpretation (4 of 5)

- RF Diagnostics – Partial Dependence 3
  - partialPlot(Members.rf, Members[-1], Age)
RF Example – Interpretation (5 of 5)

- RF Diagnostics – Partial Dependence 3

> partialPlot(Members.rf, Members[-1], Gender)
RF Example – Prediction (1 of 2)

- Need to do same variable selection & conditioning as we did for training set:

```r
> MembersTest <- read.delim("Data/MemberTestSet.txt", row.names = "MembID")
> MembersTest <- subset(MembersTest, select = -c(FirstCkInDay, LastCkInDay))
> MembersTest$DaysSinceLastUse[is.na(MembersTest$DaysSinceLastUse)] <- 999
> MembersTest$DaysSinceLastExtra[is.na(MembersTest$DaysSinceLastExtra)] <- 999
> MembersTest <- rfImpute(Status ~ ., data = MembersTest)
```

- Then we can use the “predict” method of our forest on the test data:

```r
> MembersTest.pred <- predict(Members.rf, MembersTest[-1])
> str(MembersTest.pred)
```

Factor w/ 2 levels "C","M": 2 2 2 2 2 1 1 2 2 1 ...
• Some basic R gives the actual error:

```r
> ct <- table(MembersTest[[1]], MembersTest.pred)
> cbind(ct, class.error = c(ct[1,2]/sum(ct[1,]), ct[2,1]/sum(ct[2,])))

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>M</td>
<td>class.error</td>
</tr>
<tr>
<td>C</td>
<td>511</td>
<td>295</td>
</tr>
<tr>
<td>M</td>
<td>144</td>
<td>951</td>
</tr>
</tbody>
</table>
```

```r
> (ct[1, 2] + ct[2, 1]) / length(MembersTest$Status)    # Test Set Error

[1] 0.2309311
```

• Recall our original OOB error estimates:
  – 21% overall error rate.
  – 33% false positive
  – 13% false negative
RF Example – Scoring

• Need a score? Count the trees.

```r
AtRiskScore <- floor(9.99999 * Members.rf$votes[, 1]) + 1
barplot(table(AtRiskScore), col = "yellow",
       ylab = "# Members", main = "Distribution of At-Risk Scores")
```
More Random Forests

• More capability in randomForest package
  – Regression Forest
  – Unsupervised Classification
  – Outlier measures
  – Prototypes

• Other Random Forests in R world
  – cforest in party package
    • Hothorn, Hornik & Zeileis; Vienna
  – varSelRF uses RF for variable selection
    • Ramón Díaz-Uriarte; Madrid
Conclusion - Random Forest Summary

- Has yielded practical results in number of cases
- Minimal tuning, no pruning required
- Black box, with interpretation
- Scoring fast & portable
Questions? Comments?

- Email JPorzak@LoyaltyMatrix.com
- Archive http://porzak.com/JimArchive/
- Call 415-296-1141
- Visit http://www.LoyaltyMatrix.com
- Come by at:
  580 Market Street, 6th Floor
  San Francisco, CA 94104
R Setup for Tutorial

This is the setup I will be using during the tutorial, you may, of course, change OS, editor, paths to match your own preferences.

- Windows XP SP2 on 3GHz P4 w/ 1G RAM.
- R Version 2.4.0
- RWinEdt & WinEdt V5.4 or JGR
- Following packages will be used
  - randomForest
- Directory Structure
  - R’s working directory & source code: C:\Projects\CIwR\R
  - Tutorial data loaded in: C:\Projects\CIwR\R\Data
  - Plots will be stored in: C:\Projects\CIwR\R\Plots
- Other tools I like to use
  - TextPad: www.TextPad.com
  - DbVisualizer: http://www.dbvis.com/products/dbvis/
- Download data/code from my archive: http://porzak.com/JimArchive/
RDBMS Datamart using a Star Schema

- See Ralph Kimball: http://www.kimballgroup.com
- Holds “Analysis Ready” data
Staging Data for Analysis – Moving to R

• Use RODBC to load directly from datamart

```r
require(RODBC)
cODBC <- odbcConnect("KeyCustomers") # in Windows: odbcConnect("") works
myQuery <- readChar("../SQL/MyQuery.sql", nchars = 99999) # use cat(myQuery) to view
MyDataFrame <- sqlQuery(cODBC, myQuery)
# Fix up datatypes, factors if necessary
MyDataFrame$DatePch <- as.Date(MyDataFrame$DatePch)
str(MyDataFrame)
head(MyDataFrame)
```

• Use SQL export & read.table
  – We’ll use read.delim for tutorial (I like tab delimited)

```r
KeyCustomers <- read.delim("Data/KeyCustomers.txt", row.names = "ActNum")
```

• Sampling large data sets
  – RANDOM table trick (two columns: integer identity & runif [0, 9999])

```r
SELECT SUBT_ID, etc...
FROM NewSubscribers ns
JOIN Random r
  ON r.identity_key = ns.SUBT_ID
AND r.random <= 100 -- for 10% sample
```
Profiling Raw Data in R

• Profile staged raw data to check assumptions about data made when defining problem

### TriRaw\_raw_redemptions\_POINTS\_USED

<table>
<thead>
<tr>
<th>#</th>
<th>Rows</th>
<th>Nulls</th>
<th>Distinct</th>
<th>Empty</th>
<th>Numeric</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>100.00</td>
<td>0.00</td>
<td>2.632</td>
<td>0.00</td>
<td>90.651</td>
<td>29.983</td>
</tr>
<tr>
<td>Min.</td>
<td>80</td>
<td>1st Qu.</td>
<td>3,500</td>
<td>Median</td>
<td>20,000</td>
<td>Mean</td>
</tr>
</tbody>
</table>

Distribution of POINTS\_USED

### AMA\_Stage\_CUSTOMER\_GENDER

<table>
<thead>
<tr>
<th>#</th>
<th>Rows</th>
<th>Nulls</th>
<th>Distinct</th>
<th>Empty</th>
<th>Numeric</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>100.00</td>
<td>44.41</td>
<td>0.001</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Categories in GENDER

Details in useR! 2006 talk