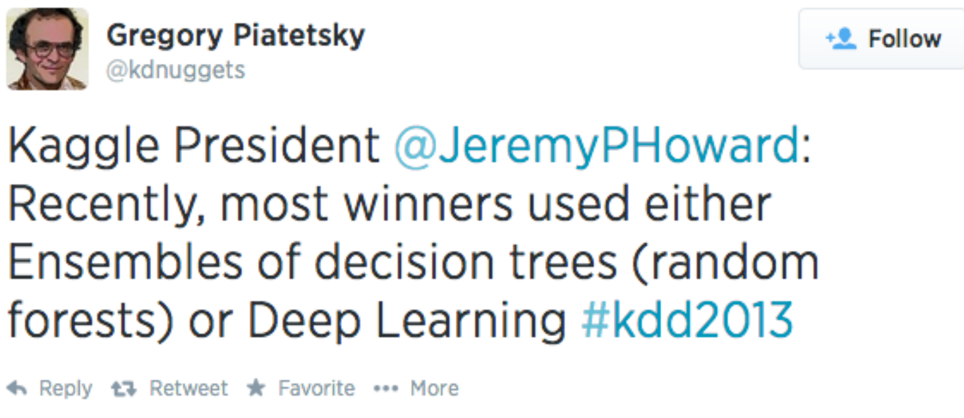
Interpreting Tree Ensembles with *inTrees*

Houtao Deng



Gregory Piatetsky
@kdnuggets

Follow

Kaggle President @JeremyPHoward: Recently, most winners used either Ensembles of decision trees (random forests) or Deep Learning #kdd2013

Reply Retweet Favorite More

source: Twitter

source: StackExchange

How to make Random Forests more interpretable?

6

Are there any methods that one could utilize to make Random Forest more interpretable? Random Forest performs much better than CART but it is a lot less interpretable.

machine-learning random-forest

2

share improve this question

edited Jul 11 '12 at 18:48
chl ♦
30.8k 5 83 190

asked Jul 11 '12 at 18:25
lord12
194 4

I'll introduce the *inTrees* framework and demo the R package

for extracting interpretable information from tree ensembles.

(focus on functions/usability instead of algorithms *)

* Algorithms are described in technical report: "Interpreting Tree Ensembles with inTrees", Houtao Deng, 2014

Supervised learning

Build a model to predict the outcome given a set of predictors.

example: predict if a customer will purchase a product given predictors:

X1: income(K/year); X2: 1: California resident
0: non-California resident

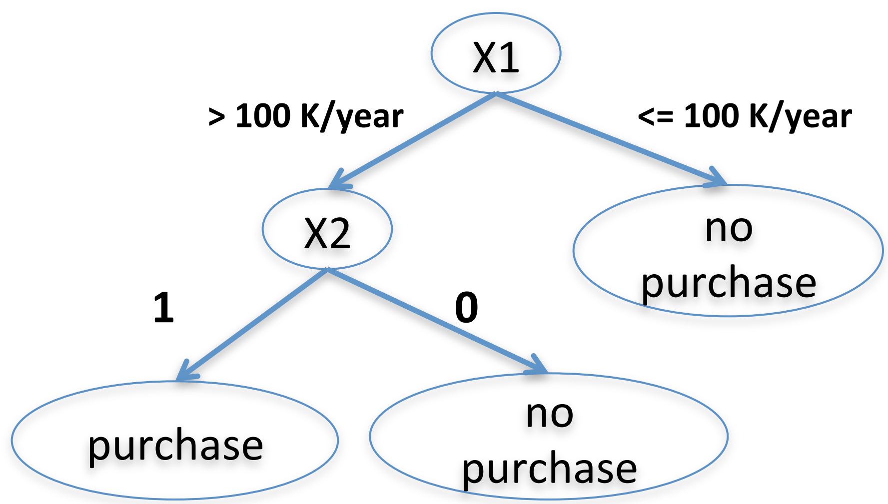
Logistic regression model

prediction based on the value of a linear combination of the predictors

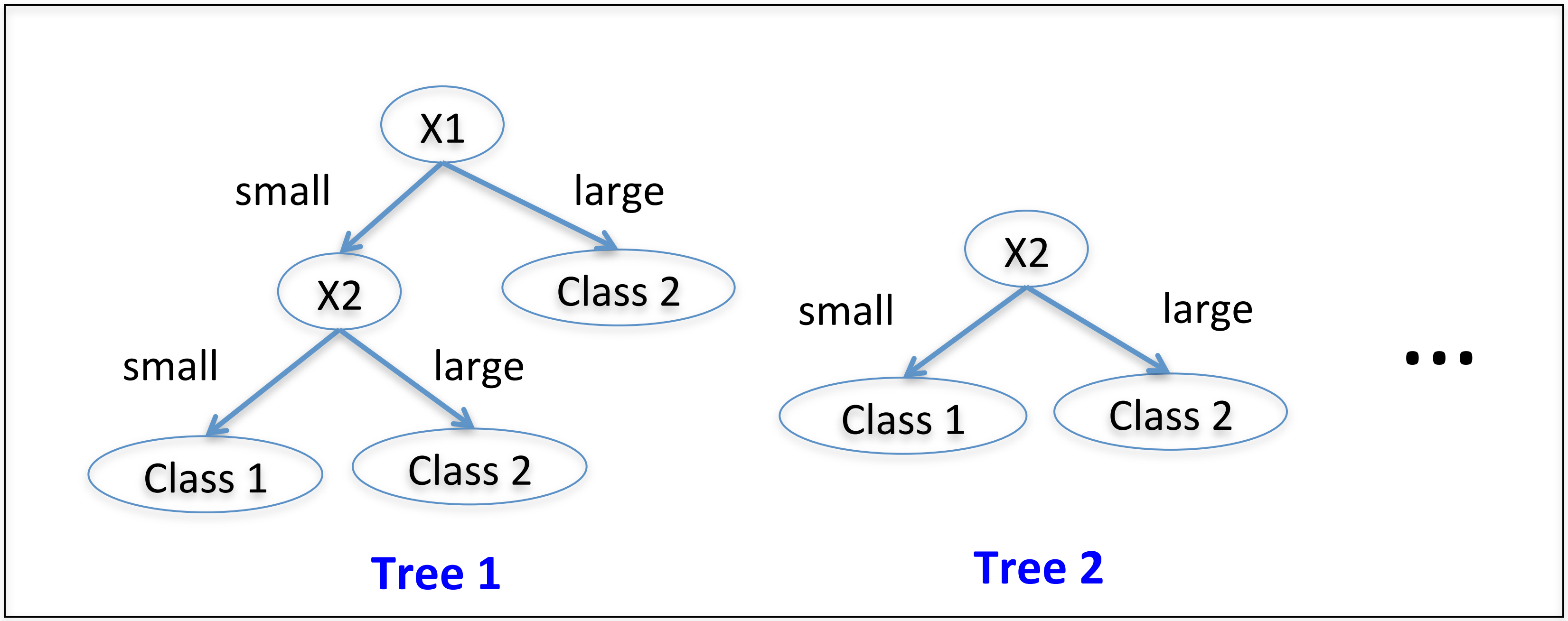
e.g., a customer is predicted to purchase if

$$0.01 * X1 + 2 * X2 > v$$

Decision tree



Tree ensembles are considered the most accurate learners



Accuracy is not the only goal in practice

Tree ensembles are

- difficult to understand and communicate
- difficult to debug, i.e., find "modeling bugs"
  - an example of modeling bugs: In customers' profile one includes the predictor "whether a customer has made the payment" to predict "whether a customer purchases a product" (the model has no value)
- difficult to deploy
  - particularly when you train a model with one programming language but apply the model with another

Tree ensembles are popular in data competitions where data are often well-prepared and the predictive accuracy is the only goal.

Thus the *inTrees* framework!

- Extract interpretable information from a tree ensemble, particularly,
  - Extract rules
  - Measure rules
  - Prune rules
  - Select rules
  - Summarize rules
  - Discover frequent variable interactions
- Applicable to many tree ensembles such as as random forests, regularized random forests and boosted trees

The team-optimization problem

- 10 players are chosen from 20 to play a game. The team would win only if
  - either player 1 or player 2 is in the team and
  - Player 1 and player 2 do not play together

- X1, ..., X20, respectively, represent player 1, ..., player 20.

- "Xi = Y" means player i plays, "Xi = N" means player i does not play

Traditional models couldn't capture the true patterns

Logistic regression model
glmnet R package

```
21 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept) -0.16553963
X1           .
X2           0.06863864
X3           0.32582543
X4           0.22011867
X5           0.48373506
X6           .
X7           .
X8           .
X9          -0.18818627
X10          .
X11         -0.08020167
X12         -0.34149303
X13          .
X14          0.42807918
X15          .
X16          .
X17          .
X18          0.22368637
X19         -0.35657330
X20          .
```

Single decision tree
rpart R package

```
n= 100

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 100 44 win (0.4400000 0.5600000)
   2) X12=Y 50 24 lose (0.5200000 0.4800000)
     4) X19=Y 24  9 lose (0.6250000 0.3750000)
       8) X11=Y 12  2 lose (0.8333333 0.1666667) *
       9) X11=N 12  5 win (0.4166667 0.5833333) *
     5) X19=N 26 11 win (0.4230769 0.5769231)
      10) X5=N 14  6 lose (0.5714286 0.4285714) *
      11) X5=Y 12  3 win (0.2500000 0.7500000) *
   3) X12=N 50 18 win (0.3600000 0.6400000)
     6) X5=N 22 11 lose (0.5000000 0.5000000)
      12) X19=Y 10  3 lose (0.7000000 0.3000000) *
      13) X19=N 12  4 win (0.3333333 0.6666667) *
     7) X5=Y 28  7 win (0.2500000 0.7500000) *
```

Use regularized random forests

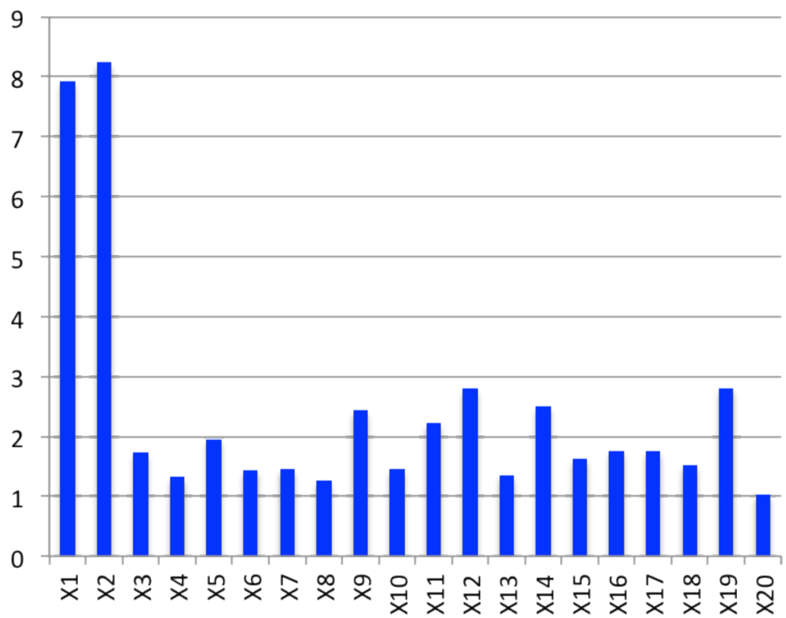
Tree format:

	left daughter	right daughter	split var	split point	status	prediction
1	2	3	8	1	1	0
2	4	5	5	1	1	0
3	6	7	5	1	1	0
4	8	9	6	1	1	0
5	10	11	14	1	1	0
...	...	...	...	...	...	...

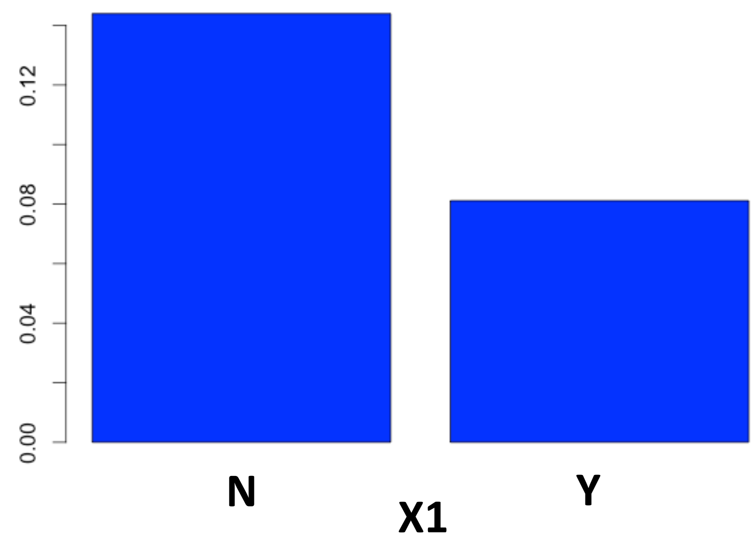
- RRF with 100 trees has 100 such matrices
- Each tree is built based on partial data and partial features, and thus could make un-reliable decisions, or include "noise" variables

Existing ways for interpreting tree ensembles I

Importance score
RRF.importance



Partial dependence plot
RRF.partialPlot



Both methods provide insights to individual variables, but can not tell how multiple variables interact.

Existing ways for interpreting tree ensembles II

Form a linear combination of the rules (rule ensemble a.k.a. RuleFit*).

Imp.	Coeff.	sup.	Rule
100	0.57	0.49	$0.25 \leq x_6 < 0.75$
99	0.79	0.15	$x_1 \geq 0.35$ & $x_2 \geq 0.45$ & $x_3 \geq 0.45$
83	-0.81		linear: x_7
63	0.61		linear: x_8
61	0.34	0.51	$0.35 \leq x_6 < 0.85$
58	-0.38	0.25	$x_4 < 0.35$ & $x_5 \geq 0.45$

inTrees **provides a more general framework for rule analysis for a broad set of tree ensembles**

* my colleague Giovanni Seni has developed Rego, an open-source command-line batch interface for RuleFit

inTrees: extract conditions

X1 = Y & X2 = Y -> win

rule condition

rule outcome

1923 (1835 distinct) conditions are extracted from RRF (100 trees).

For example:

condition
X[,1] %in% c('N') & X[,2] %in% c('N') & X[,19] %in% c('N')
X[,1] %in% c('Y') & X[,2] %in% c('N') & X[,19] %in% c('N')

These conditions are R-executable

inTrees: assign outcomes

condition
X[,1] %in% c('N') & X[,2] %in% c('N') & X[,19] %in% c('N')
X[,1] %in% c('Y') & X[,2] %in% c('N') & X[,19] %in% c('N')

condition	pred
X[,1] %in% c('N') & X[,2] %in% c('N') & X[,19] %in% c('N')	lose
X[,1] %in% c('Y') & X[,2] %in% c('N') & X[,19] %in% c('N')	win

inTrees: measure rules

- Length: # variable-value pairs in a rule condition
- Frequency: proportion of instances that satisfy a rule condition
- Error: the error rate of a rule

condition	pred
X[,1] %in% c('N') & X[,2] %in% c('N') & X[,19] %in% c('N')	lose
X[,1] %in% c('Y') & X[,2] %in% c('N') & X[,19] %in% c('N')	win

⬇

len	freq	err	condition	pred
3	0.07	0	X[,1] %in% c('N') & X[,2] %in% c('N') & X[,19] %in% c('N')	lose
3	0.16	0	X[,1] %in% c('Y') & X[,2] %in% c('N') & X[,19] %in% c('N')	win

inTrees: prune each rule

len	freq	err	condition	pred
3	0.07	0	X[,1] %in% c('N') & X[,2] %in% c('N') & X[,19] %in% c('N')	lose
3	0.16	0	X[,1] %in% c('Y') & X[,2] %in% c('N') & X[,19] %in% c('N')	win

irrelevant variables

len	freq	err	condition	pred
2	0.22	0	X[,1] %in% c('N') & X[,2] %in% c('N')	lose
2	0.24	0	X[,1] %in% c('Y') & X[,2] %in% c('N')	win

inTrees: select a compact rule set

len	freq	err	condition	pred
2	0.22	0	X[,1] %in% c('N') & X[,2] %in% c('N')	lose
2	0.24	0	X[,1] %in% c('Y') & X[,2] %in% c('N')	win

• • • 1000+ rules

len	freq	err	condition	pred	impRRF
2	0.22	0	X[,1] %in% c('N') & X[,2] %in% c('N')	lose	1
2	0.22	0	X[,1] %in% c('Y') & X[,2] %in% c('Y')	lose	0.97
2	0.24	0	X[,1] %in% c('Y') & X[,2] %in% c('N')	win	0.60
2	0.32	0	X[,1] %in% c('N') & X[,2] %in% c('Y')	win	0.42

inTrees: summarize rules (simplified tree ensemble learner)

len	freq	err	condition	pred
2	0.22	0	X[,1] %in% c('N') & X[,2] %in% c('N')	lose
2	0.24	0	X[,1] %in% c('Y') & X[,2] %in% c('N')	win

• • • 1000+ rules

len	freq	err	condition	pred
2	0.32	0	X[,1] %in% c('N') & X[,2] %in% c('Y')	win
2	0.24	0	X[,1] %in% c('Y') & X[,2] %in% c('N')	win
1	0.44	0	X[,1]==X[,1]	lose

More readable format

len	freq	err	condition	pred
2	0.32	0	X1 %in% c('N') & X2 %in% c('Y')	win
2	0.24	0	X1 %in% c('Y') & X2 %in% c('N')	win
1	0.44	0	Else	lose

inTrees: discover frequent variable interactions

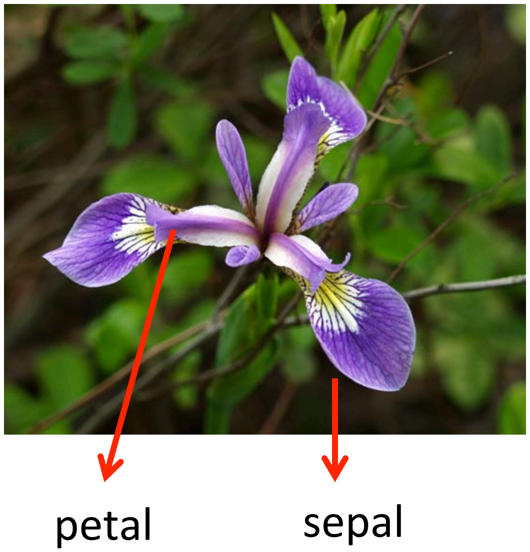
- support: proportion of tree ensemble rules containing the condition
- confidence: proportion of rules containing the condition and the rule outcome

Top variable interactions (large support) with length >= 2

len	sup	conf	condition	pred
2	0.046	1	X[,1] %in% c('N') & X[,2] %in% c('N')	lose
2	0.044	1	X[,1] %in% c('Y') & X[,2] %in% c('N')	win
2	0.041	1	X[,1] %in% c('N') & X[,2] %in% c('Y')	win
2	0.039	1	X[,1] %in% c('Y') & X[,2] %in% c('Y')	lose
2	0.034	0.699	X[,12] %in% c('N') & X[,5] %in% c('Y')	win
2	0.032	0.667	X[,12] %in% c('Y') & X[,19] %in% c('Y')	lose
2	0.029	0.696	X[,11] %in% c('Y') & X[,19] %in% c('Y')	lose
2	0.028	0.635	X[,11] %in% c('Y') & X[,12] %in% c('Y')	lose
2	0.026	0.595	X[,12] %in% c('N') & X[,5] %in% c('N')	lose
2	0.025	0.615	X[,19] %in% c('Y') & X[,9] %in% c('Y')	lose

Real-life example: Iris data

setosa



virginica

versicolor

simplified tree ensemble learner (STEL)

len	freq	err	condition	pred
1	0.37	0	petal.width<=0.7	setosa.
3	0.31	0	petal.len>2.6 & petal.len<=4.85 & petal.width<=1.6	versicolor.
2	0.25	0	petal.len>4.85 & petal.width>1.7	virginica.
3	0.03	0	sepal.width<=3.1 & petal.len<=4.85 & petal.width>1.6	virginica.
3	0.02	0	sepal.width>2.25 & petal.width>0.8 & petal.width<=1.75	versicolor.
1	0.01	0	petal.len>4.85	virginica.
1	0.01	0	Else	versicolor.

Experiments on 20 publicly available data sets show that: simplified tree ensemble learner (STEL) outperforms decision tree rpart

	numInst	numFea	rpart	STEL	difference(%)
anneal	898	38	0.098	0.070 ●	28.7%
austra	690	14	0.145	0.157 ○	8.0%
auto	205	25	0.376	0.262 ●	30.2%
breast	699	10	0.058	0.048 ●	17.4%
crx	690	15	0.148	0.159 ○	7.2%
german	1000	20	0.274	0.286 ○	4.3%
glass	214	9	0.342	0.310 ●	9.2%
heart	270	13	0.219	0.224	2.1%
hepati	155	19	0.209	0.211	0.6%
horse	368	22	0.164	0.197 ○	16.6%
iris	150	4	0.064	0.047 ●	26.6%
labor	57	16	0.223	0.148 ●	33.7%
led7	3200	7	0.318	0.269 ●	15.3%
lymph	148	18	0.268	0.209 ●	21.9%
pima	768	8	0.260	0.272 ○	4.4%
tic-tac	958	9	0.094	0.002 ●	97.9%
vehicle	846	18	0.325	0.285 ●	12.2%
waveform	5000	21	0.262	0.198 ●	24.2%
wine	178	13	0.122	0.086 ●	29.8%
zoo	101	16	0.211	0.061 ●	71.3%

Results with statistically significant differences are marked with circles (rpart wins) or filled circles (STEL wins).

- STEL outperforms rpart in 13 data sets and loses in only 5 (with statistically significant differences)
- When STEL wins, the improvements are greater (most data sets have more than 10% of improvements)

Most accurate rule for each data set with minimum freq of 0.1

	len	freq	err	condition	pred
anneal	5	0.342	0	X4<=1.5 & X5<=82.5 & X7 %in% c('S') & X8<=2.5 & X33<=0.7995	3
austra	5	0.181	0	X5<=7.5 & X7<=3.375 & X8<=0.5 & X13<=415.5 & X14<=251	0
auto	5	0.195	0	X1>71.5 & X2 %in% c('bmw','honda','isuzu', 'jaguar','mazda','nissan','peugot','subaru','toyota') & X5 %in% c('four') & X10<=187.25 & X21>69.5	0
breast	3	0.591	0	X3<=3.5 & X7<=2.5 & X9<=3.5	benign
crx	4	0.188	0	X3>1.5625 & X6 %in% c('aa','c','d','ff','i','j', 'k','m','r') & X9 %in% c('f') & X15<=492	no
german	5	0.132	0.015	X1 %in% c('no-account') & X5<=4103.5 & X10 %in% c(' guarantor',' none') & X13>33.5 & X14 %in% c(' none')	good
glass	5	0.136	0	X1<=1.517325 & X3>2.7 & X4>1.42 & X7>7.82 & X9<=0.16	bwnfp
heart	4	0.2	0	X1<=55.5 & X4>119 & X10<=1.7 & X13<=4.5	1
hepati	6	0.542	0	X1<=61.5 & X11>1.5 & X13>1.5 & X14<=3.7 & X15<=218.5 & X18>40.5	2
horse	5	0.188	0	X3<=38.45 & X3>37.25 & X4<=126 & X10>2.5 & X12>2.5	1
iris	1	0.333	0	X3<=2.55	setosa
labor	4	0.614	0	X2>2.75 & X7 %in% c('empl_contr','ret_allw') & X8>5 & X13 %in% c('yes')	good
led7	3	0.103	0.211	X1<=0.5 & X2<=0.5 & X6>0.5	1
lymph	4	0.284	0	X2>1.5 & X13>2.5 & X13<=3.5 & X18<=2.5	2
pima	3	0.124	0	X2<=106.5 & X6<=29.95 & X8<=28.5	0
tic-tac	3	0.225	0	X1 %in% c('b','x') & X5 %in% c('b','x') & X9 %in% c('b','x')	positive
vehicle	5	0.102	0	X3>71.5 & X6>8.5 & X7>142.5 & X12<=376.5 & X14>63.5	4
waveform	5	0.102	0.059	X6<=1.655 & X9<=2.99 & X11>3.415 & X12>2.49 & X14>2.075	2
wine	3	0.337	0	X1<=12.78 & X2<=4.575 & X10<=4.84	2
zoo	1	0.406	0	X4>0.5	1

To Latex users: this table was produced by
"print(xtable(*learner*), include.rownames=FALSE)"

Demo*: using inTrees for 3 tree ensembles

- Random forest ("randomForest" R package)
- Regularized random forest ("RRF" R package)
- Boosted trees ("gbm" R package)

* Demo code is available at: sites.google.com/site/houtaodeng/intrees