Beyond election day: A knowledge Boost so we see the Forest for the Trees while standing on the ISLE with a Bag of voters

Thomas Rusch

19.5.2010

Outline

- The problem and the data
- A nonparametric classifier: Trees
  - Classification and Regression Trees (CART)
  - Conditional Inference Trees (ctrees)
- Improving Performance: Ensemble methods
  - Bagging
  - Random Forests
  - Boosting
  - Importance Sampled Learning Ensembles (ISLE)
- Conclusion

The problem

We want to predict the voter turnout for the 2004 general election in Ohio (USA).

We therefore have a supervised learning problem with an output

$$y_i = \begin{cases} 1 & \text{if person } i \text{ voted} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Each output $i, i = 1, \ldots, N$ depends on p predictor variables x_{ip}, combined to input vector x_i.

We want to predict $\widehat{y} = \widehat{f}(x)$, which will typically lie in $[0, 1]$.
If $\widehat{y} > .5$ it will be assigned to class "1".

The data

We have a data set with $N = 19634$ observations from Ohio with $p = 74$ predictors.

Among them we have for each person

- Voting records from 1990-2004
- Party affiliation, household composition, federal contributor
- Family rank, house owner
- Funding activities
- Gender, education, income, age (in days)
- Overall voting count (simple and eligible)

Our target variable is the individual's voting behavior in the 2004 general election (Bush vs. Gore)

Data description I

Overall, the turnout in 2004's general election was 69.97% (non-information rate)
Some additional descriptive statistics:

PARTY_MIX							
0	1	2	3	4	5	6	7
2180	2246	2358	7021	2590	641	2214	384

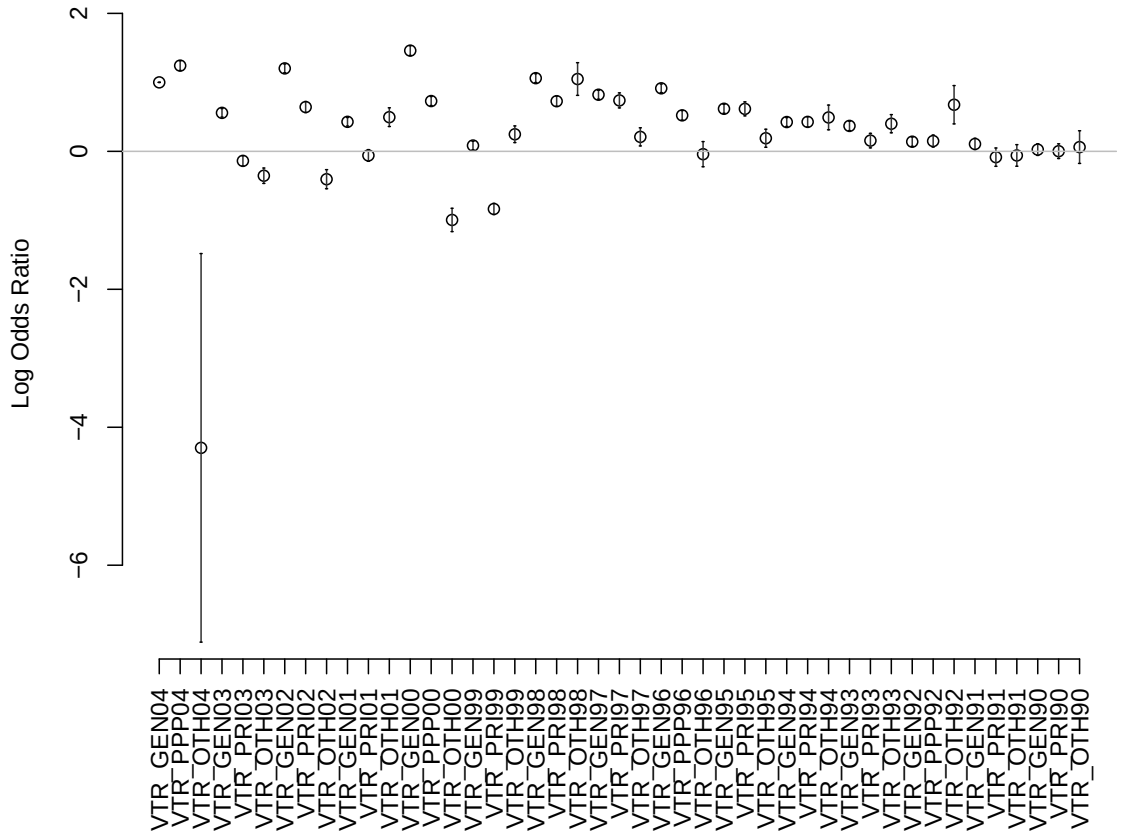
TRAIL_CNT						
1	2	3	4	5	6	7
10567	6963	1547	456	88	12	1

Data description II

SEX		PARTY_CODE		
F	M	D	R	I
10323	9311	4348	4359	10927

ageY				
Min.	1st Qu.	Med.	3rd Qu.	Max.
19	36	45	54	108

Bivariate associations of voting history and target variable



What are Trees?

Trees are a simple and intuitive form of additive models
They partition the feature space into disjoint rectangles and fit a constant in every partition
Advantages are

- Non-parametric and robust method
- Non-linear decision boundary (focus on interactions)
- Interpretability
- Missing data (surrogate variables) and mixed input handling

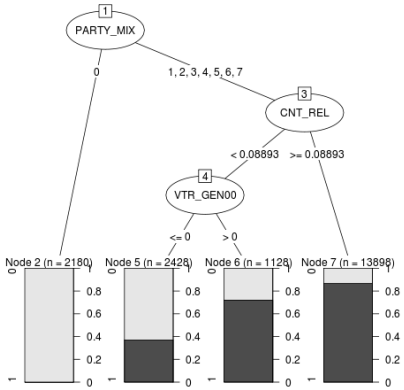
Disadvantages are

- Relatively low predictive power
- High variance

Binary Trees: Idea and Visualization

- One starts with the whole feature space of inputs.
- That one is split into two regions based on a split point of a single variable.
- The split point and variable is chosen to optimize a fit criterion.
- The procedure is then repeated recursively in each region until a stopping rule applies.

It looks like this:



Binary Trees: Formalization I

Say we have a M partitions, $R_1, \ldots, R_M$ and each response is predicted by a region specific constant c_m

The tree model is then

$$\hat{f}(x) = \sum_{m=1}^{M} c_m I(x \in R_m) \tag{2}$$

In our binary classification case, c_m is the relative frequency of class "1".
Starting with all data, the CART algorithm splits it into two daughter nodes according to the splitting variable j and the split point s.

$$R_1(j,s) = \{X|X_j \leq s\} \text{ and } X_2(j,s) = \{X|X_j > s\} \tag{3}$$

denote the two daughter nodes (half-planes of the starting node).

Binary Trees: Formalization II

The splitting variable j and the split point s are found by using e.g. the Gini index and therefore solving

$$\min_{j,s} \left[N_1 c_1 (1 - c_1) + N_2 c_2 (1 - c_2) \right] \tag{4}$$

with

$$\widehat{c}_k = \frac{1}{N_k} \sum_{x_i \in R_k(j,s)} I(y_i = 1); \qquad k = 1, 2. \tag{5}$$

Other loss functions can be used such as the misclassification error or the deviance.

Binary Trees: Pruning

Trees may overfit, hence keep them from growing:

Grow large tree T_0 with minimum node size and find $T \subset T_0$ with size $|T|$

- Determine depth via minimal missclassification rate $Q_m(T) = 1 - max(c_m, 1 - c_m)$ on test set (via CV)

- Cost-complexity pruning: Find $T_\alpha \subseteq T_0$ that minimizes

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T| \tag{6}$$

One can also use Gini index or deviance for $Q_m(T)$ and α can be chosen adaptively via CV.

Binary trees: Conditional Inference Trees I

CART favor variables with many splits and complete data
Alternative: Conditional Inference Trees (Hothorn et. al., 2006)

Unbiasedness achieved by the following idea:

- Selecting the splitting variable and split point based on statistical criterion
- Criterion: Measure of association between the x_{ip} and y_i, utilizing a linear statistic and its asymptotic distribution of the permutation test framework of Strasser & Weber (1999).

Binary trees: Conditional Inference Trees II

Specifically, the approach is like this

- Test global null hypothesis of independence in terms of the p partial hypotheses $H_0^p : D(y|x_p) = D(y)$.
- Calculate the linear association statistic for the output variable and every input variable
- Choose variable with lowest (significant) p-value for splitting
- Split by maximizing a linear statistic between all binary possible partitions
- If H_0 is not rejected at a pre-specified level α, the algorithm stops.

Binary trees: Analysis - Setup

We used `caret` to conduct model selection and validation throughout.

- Estimated expected prediction accuracy via 10-fold cross validation for different tuning parameters
- Take configuration with maximal accuracy
- Use the average missclassification of the final model on all test sets as expected test error

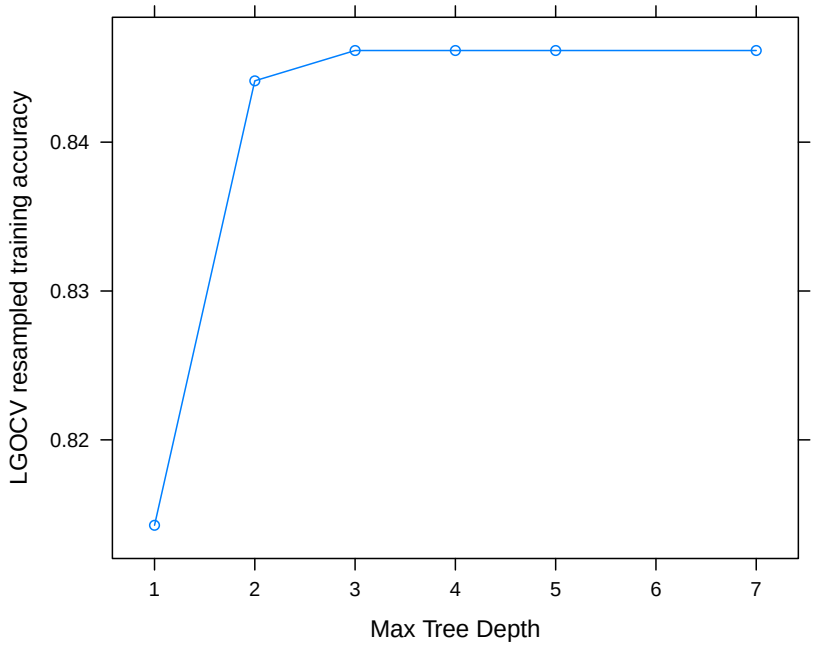
For trees specifically

- For CART: Tuned by maximum depth of tree from 1 to 15 in 10-fold CV
- For Ctree: Tuned by global significance value from 0.001 to 0.99 in 10-fold CV

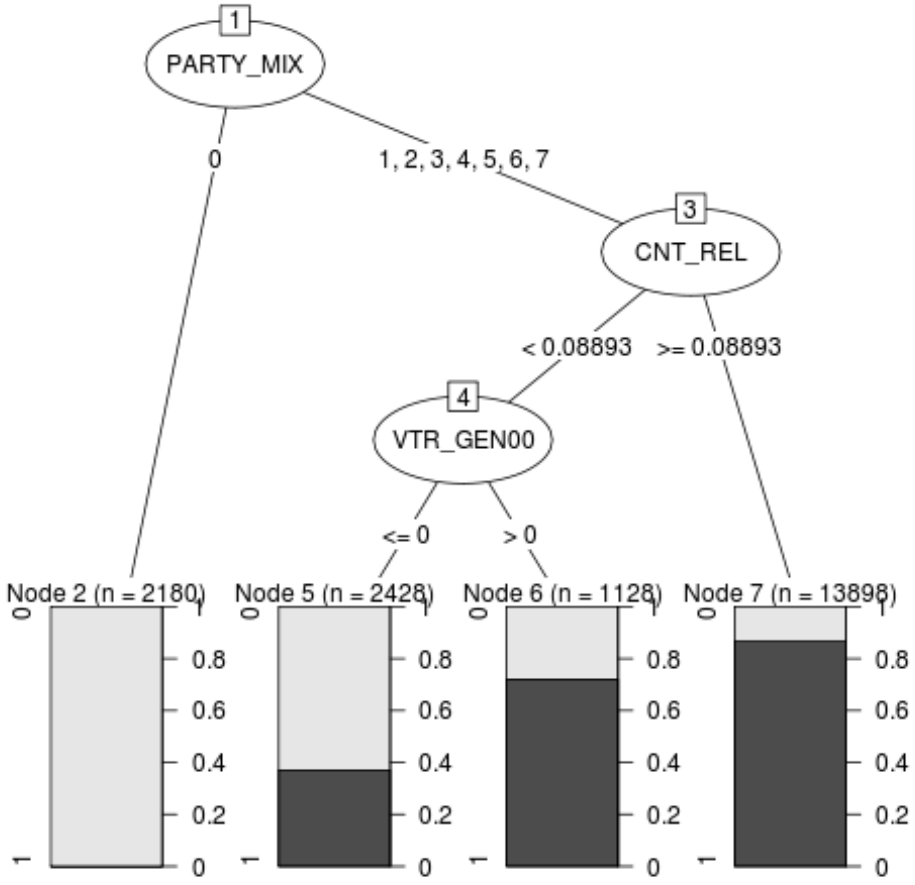
We used `rpart` and `party` to fit the trees.

Binary trees: Analysis - Results CART I

```
> optRPART <- train(predictorVot, predictionVot, method = "rpart",
+     tuneGrid = rpartGrid, trControl = trainControl(method = "LGOCV",
+         number = 10, p = 0.66, returnResamp = "all", verboseIter = TRUE))
```



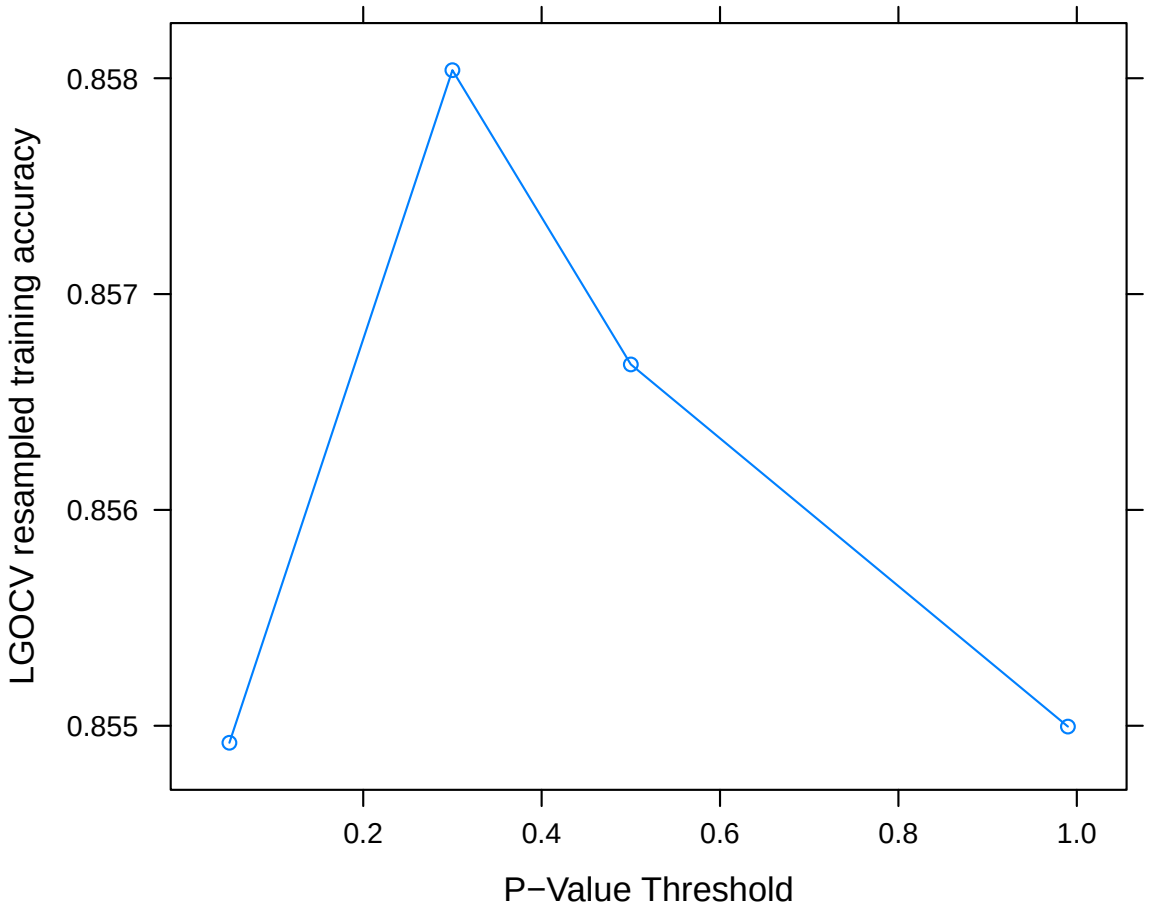
Binary Trees: Analysis - Results CART II



Binary trees: Analysis - Results Ctree I

```
> optCTREE <- train(VTR_GEN04 ~ SEX + PARTY_CODE + PARTY_MIX +
+     HOH_FLAG + TRAIL_CNT + INCOME + HOME_OWNER + MAIL_ORD + COMP_OWN +
+     ED_LEVEL + FUND_RELIG + FUND_POLIT + FUND_ARTS + FUND_ANIML +
+     FUND_CHILD + FUND_HELTH + FUND_PVRTY + FUND_ENVIR + FEC_01_02 +
+     FEC_99_00 + FEC_97_98 + FEC_95_96 + FEC_93_94 + FEC_03_04 +
+     FEC_05_06 + PHONE_SUPR + ageD + VTR_CNT + VTR_GEN90 + VTR_PRI90 +
+     VTR_OTH90 + VTR_GEN91 + VTR_PRI91 + VTR_OTH91 + VTR_GEN92 +
+     VTR_PPP92 + VTR_OTH92 + VTR_GEN93 + VTR_PRI93 + VTR_OTH93 +
+     VTR_GEN94 + VTR_PRI94 + VTR_OTH94 + CNT_ELL + CNT_REL + VTR_GEN95 +
+     VTR_PRI95 + VTR_OTH95 + VTR_GEN96 + VTR_PPP96 + VTR_OTH96 +
+     VTR_GEN97 + VTR_PRI97 + VTR_OTH97 + VTR_GEN98 + VTR_PRI98 +
+     VTR_OTH98 + VTR_GEN99 + VTR_PRI99 + VTR_OTH99 + VTR_GEN00 +
+     VTR_PPP00 + VTR_OTH00 + VTR_GEN01 + VTR_PRI01 + VTR_OTH01 +
+     VTR_GEN02 + VTR_PRI02 + VTR_OTH02 + VTR_PRI03 + VTR_GEN03 +
+     VTR_OTH03 + VTR_PPP04 + VTR_OTH04, data = sele, method = "ctree",
+     tuneGrid = ctreeGrid, trControl = trainControl(method = "LGOCV",
+         number = 10, returnResamp = "all", p = 0.66, verboseIter = TRUE))
```

Binary trees: Analysis - Results Ctree II



Idea of Ensemble Methods

Ensemble methods try to improve a prediction by using a number of (weak) learners and letting them all vote on the prediction

– Wisdom of Crowds, Democratic learning
– Knowledge of a collective exceeds knowledge of an "individual"

Idea works best with independent, diverse "individuals"

- Advantage: Extremely high prediction accuracy
- Disadvantage: Very low interpretability, computationally intensive

For all of the following methods, trees (or stumps) are the preferred base learners

Bagging: Idea and Formalization I

Reduces the variance of an unstable, nonlinear or adaptive predictor with low bias

- Draw B bootstrap samples $(x_i*, y_i*)^b, i = 1, \ldots, N$ from original data
- Fit the model to get the predictions $\widehat{f}^{*b}(x)$
- Average the predictions over all B bootstrap samples, i.e

$$\widehat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \widehat{f}^{*b}(x) \tag{7}$$

Bagging: Idea and Formalization II

For our classification problem this means selecting the class with the most votes from the bootstrap samples for an observation with inputs x

$$\widehat{G}_{bag}(x) = argmax_{\{1,2\}}[p_1(x), p_2(x)] \tag{8}$$

with $p_k(x), k = 1, 2$ being the proportion of prediction models from the bootstrap samples predicting class k for input x

Alternatively, the average predicted class membership probabilities may be used

We used the package `ipred` and $B = 100$ bootstrap samples for the example

Bagging: Results

```
> (optBAG <- train(predictorVot, predictionVot, method = "treebag",
+     nbagg = 100, trControl = trainControl(method = "LGOCV", number = 10,
+         returnResamp = "all", p = 0.66, verboseIter = TRUE)))


Call:
train.default(x = predictorVot, y = predictionVot, method = "treebag",
    nbagg = 100, trControl = trainControl(method = "LGOCV", number = 10,
        returnResamp = "all", p = 0.66, verboseIter = TRUE))

19634 samples
74 predictors

summary of leave group out cross-validation (10 reps) sample sizes:
    12959, 12959, 12959, 12959, 12959, 12959, ...

LGOCV resampled training results

  Accuracy  Kappa  Accuracy SD  Kappa SD
  0.874     0.689  0.00372      0.00972
```

Random Forests: Idea

Improve the performance of bagging by using a lot of decorrelated unpruned trees

- Draw B bootstrap samples $(x_i*, y_i*)^b, i = 1, \ldots, N$ from original data
- Fit a tree T_b to the bootstrap sample by repeating recursively for each terminal node
  - Randomly select m variables of the p predictors
  - Split node at best splitting variable and point of the m candidates
- Average the tree ensemble T_{b1}^B

For classification the prediction will be

$$\hat{C}_{rf}^B(x) = majorityvote\{\hat{C}_b(x)\}_1^B \tag{9}$$

with $\hat{C}_b(x)$ being the prediction of the bth tree

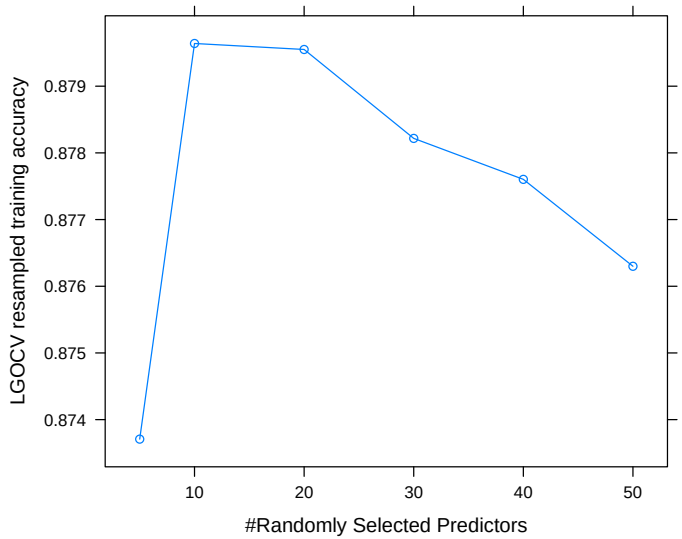
Random Forests: Usage

- Needs high number of bootstrap samples and a small number of m
- Better preformance by reducing variance, bias remains (tends to be bigger for smaller m)
- Very high prediction accuracy
- Not much tuning needed (m and maybe B)
- Overfitting is not much of a problem (occurs only because of unpruned trees)
- Inherent "CV" via Out-Of-Bag samples: Predict each observation (x_i, y_i) by averaging those trees corresponding to samples in which (x_i, y_i) did NOT occur
- Measure of variable importance: At each split in each tree, use improvement in split criterion for every variable and accumulate it over trees

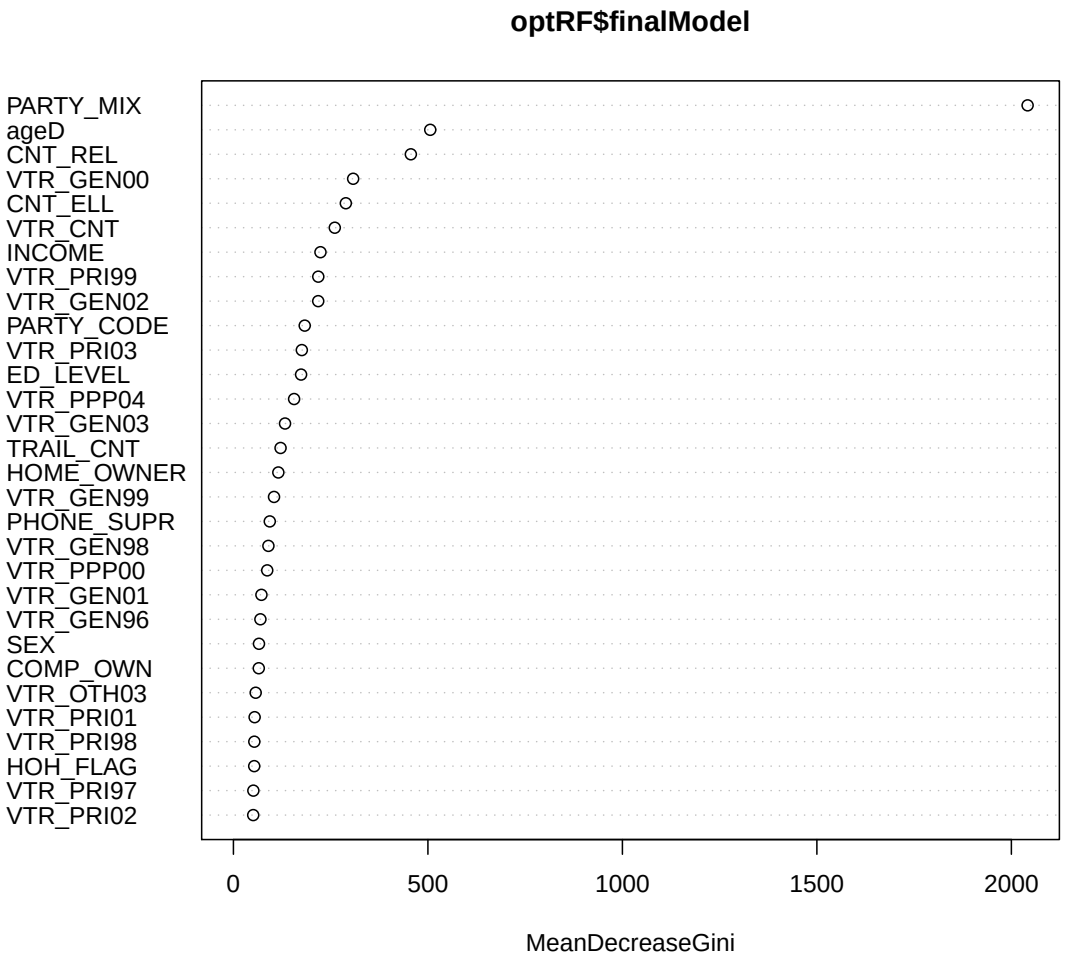
Random Forests: Results with CART I

We used the package `randomForests` with $B = 500$ and $m = 5, 10, 20, 30, 40, 50$

```
> optRF <- train(predictorVot, predictionVot, method = "rf", tuneGrid = rfGrid,
+     trControl = trainControl(method = "LGOCV", returnResamp = "final",
+         number = 10, p = 0.66, verboseIter = TRUE))
```



Random Forests: Results with CART II



Boosting: Idea

Uses the idea of combining weak learners (slightly better than random guess) to produce a "commitee" vote
Differs from bagging mainly by iteratively reweighting the missclassified sample points, so the commitee evolves and all “individuals” cast a weighted vote

- Sequentially apply a weak learner (e.g. stump) to repeatedly modified versions of the data
- Combine the produced sequence of learners via a weighted majority vote to get the final prediction
- Data are modified such that the missclassified observations get higher weights in the next iteration
- Each successive classifier must focus more and more on the difficult points

Boosting: Formalization

Boosting fits an additive model in a base learner

$$\widehat{f}(x) = \sum_{l=1}^{L} \beta_l b(x; \gamma_l) \tag{10}$$

with

$$(\beta_l, \gamma_l) = min_{\{\beta_l, \gamma_l\} 1 \ldots L} \sum_{i=1}^{N} L(y_i, \widehat{f}(x)) \tag{11}$$

Different loss functions for binary classification include

- Exponential loss ("AdaBoost"): $L(y, f(x)) = \exp(-yf(x))$
- Deviance loss: $L(y, f(x)) = -\log(1 + \exp\{-2yf(x)\})$
- More robust loss functions: Missclassification, Hinge loss, Huber loss, etc.

For classification, boosting rarely overfits.

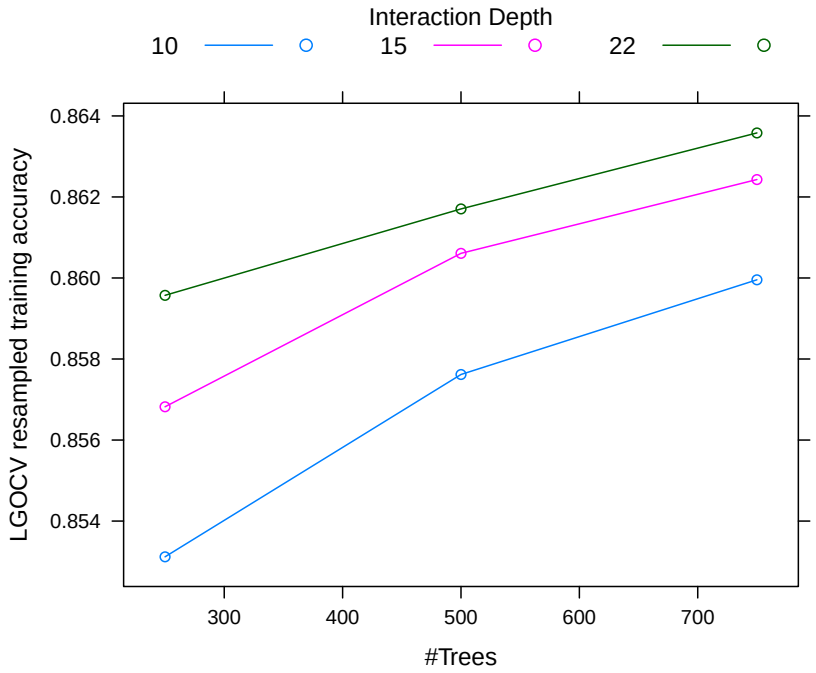
Boosting: Usage

- Mostly Trees (or stumps) are employed as the weak learner (basis expansion)
- Often (apart from AdaBoost) the model in (10) and (11) with trees as base learner has to be approximated by the "Gradient Boosting Algorithm" (includes shrinkage)
- The depth of the trees (interactions) can be used as tuning parameter
- Additional tuning can be done by number of iterations and shrinkage
- Variable importance can be assessed as in Random Forests
- Very high predictive performance

We used the package `gbm`, interaction depths were 1,2,5,10,17,22 and iterations were 50,100,250,500,750,1000,2000. Shrinkage was 0.1.

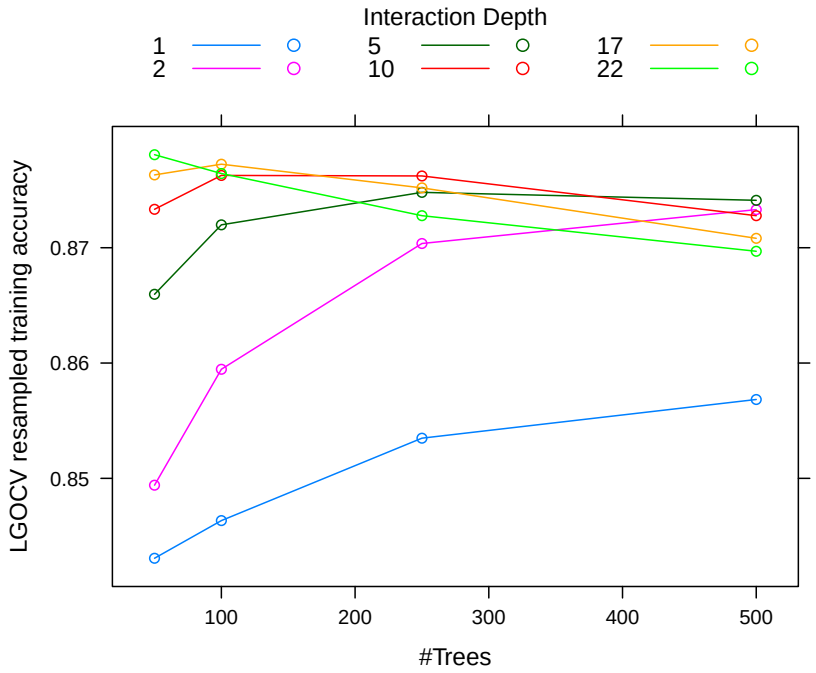
Boosting: Results with AdaBoost

```
> optADA <- train(predictorVot, predictionVot, method = "gbm",
+     distribution = "adaboost", tuneGrid = adagrid, trControl = trainControl(meth
+         number = 10, returnResamp = "final", p = 0.66, verboseIter = TRUE))
```



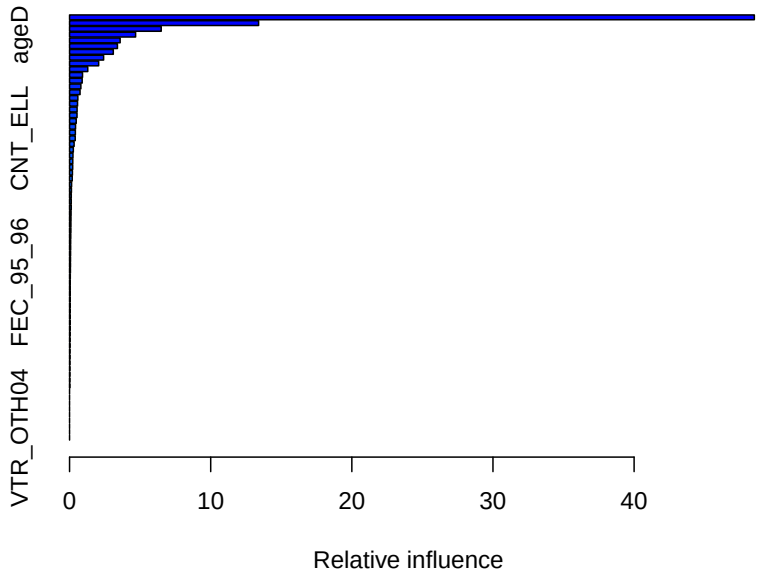
Boosting: Results with Binomial Deviance I

```
> optGBM <- train(predictorVot, predictionVot, method = "gbm",
+     tuneGrid = gbmgrid, distribution = "bernoulli", trControl = trainControl(met
+         returnResamp = "final", number = 10, p = 0.66, verboseIter = TRUE))
```



Boosting: Results with Binomial Deviance II

```
> summary(optGBM$finalModel, plotit = TRUE)[1:4, ]
        var   rel.inf
1 PARTY_MIX 48.519061
2   CNT_REL 13.394770
3 VTR_GEN00  6.494238
4      ageD  4.679659
```



Importance Sampled Learning Ensembles: Idea

There are some similarities between boosting and LASSO in terms of regularization and L_1 loss.

Led to the idea to post process an ensemble method by fitting a LASSO path into the ensemble. This will lead to a set with less trees.

Not all ensembles may be good to post-process (the basis functions should cover the space well and be distinct)
Use importance sampling for all the tree parameters (splitting variable and point and node values)

ISLE: Algorithm

Let $S_l(\eta)$ be a subsample of length $N*\eta$ (sampled without repacement)

- Set $f_0(x) = argmin_c \sum_{i=1}^{N} L(y_i, c)$
- Repeat for $l = 1$ to L
  - Calculate $\gamma_l = argmin_\gamma \sum_{i \in S_l(\eta)} L(y_i, f_{m-1}(x_i) + b(x_i; \gamma))$ (Importance sampling)
  - Calculate $f_l(x) = f_{m-1}(x) + \nu b(x; \gamma_m)$ (Regularization)
- The ISLE is then $\{b(x; \gamma_1), \ldots, b(x; \gamma_L)\}$

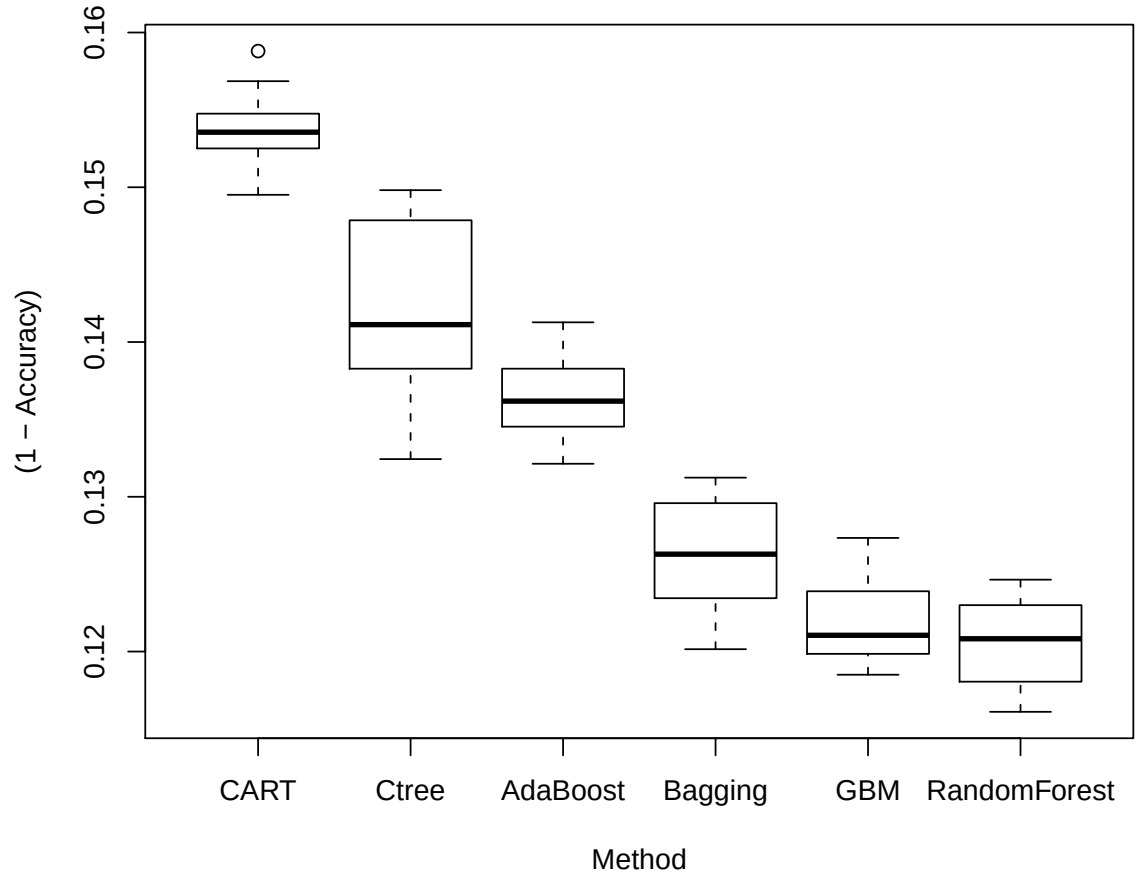
Bagging, Random forests, Gradient Boosting are special cases of ISLE (also known as stochastic gradient boosting)

Conclusion: Comparison of Results I

We estimate the expected test error by looking at the average misclassification rate over all 10 test sets for every optimal model.

Method	Av. Misclass	Sd	Parameter
CART	.154	.003	max.dept= 3
Ctree	.142	.006	mincriterion= .3
AdaBoost	.136	.003	iter= 750,interact= 22
Bagging	.126	.004	-
GBM	.122	.003	iter= 50,interact= 22
RandomForest	.120	.003	mtry= 10

Conclusion: Comparison of Results II



What else can be called an ensemble method?

Many procedures use a "commitee" idea

- Neural Networks
- Support Vector Machines
- Stacking
- Mixture of experts
- Rule Ensembles

But if basis functions are considered as "individuals", nearly everything is an ensemble

- Linear models, power functions or other tranformations in linear regression
- Trees or Regression splines (and therefore MARS or GAMs)
- Bayesian Nonparametric Regression

Therefore, in statistics, thinking in ensembles is very natural

Further Infos:

- Hastie, Tibshirani & Friedman (2009). Elements of Statistical Learning. 2nd Edition. New York: Springer.
- Hothorn, Hornik & Zeileis (2006). Unbiased Recursive Partitioning: A Conditional Inference Framework. Journal of Computational and Graphical Statistics.
- Strasser & Weber (1999). On the Asymptotic Theory of Permutation Statistics. Mathematical Methods of Statistics.