Random forests

Hedibert F. Lopes & Paulo Marques
Insper Institute of Education and Research
São Paulo, Brazil

Outline

Outline

boston dataset

Boston Housing data, Harrison and Rubinfeld (1978)[1]
The data consist of 14 characteristics of 506 census tracts in the Boston area.
The response is the logged median value of owner occupied homes in each tract.

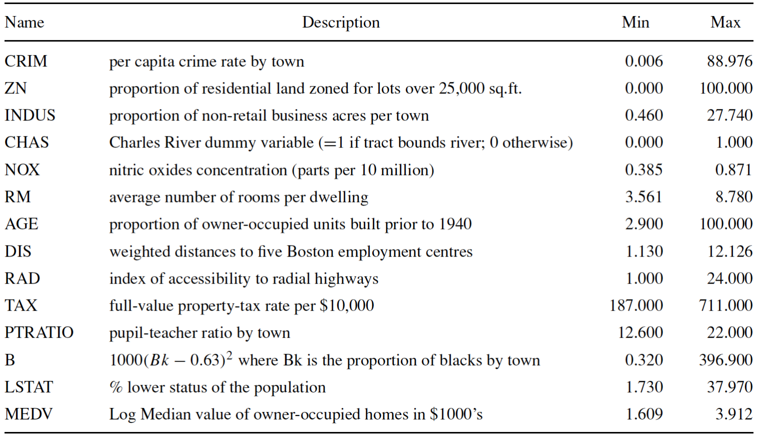


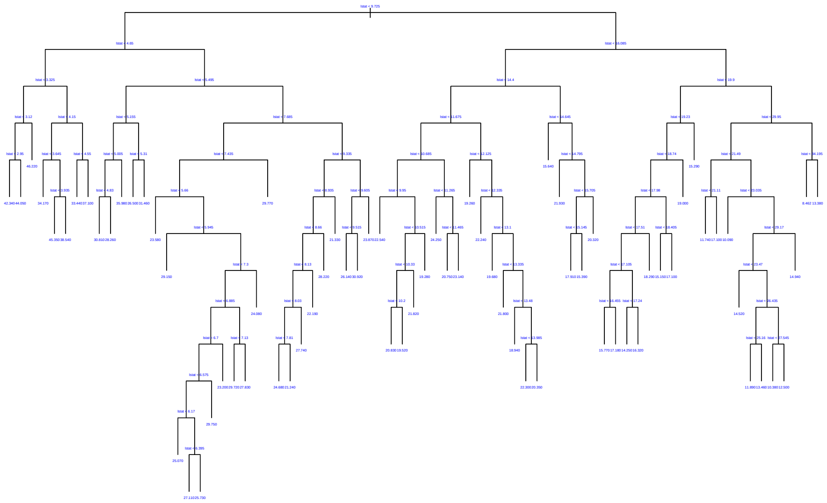
Table 1. Variables in the Boston dataset.

Name	Description	Min	Max
CRIM	per capita crime rate by town	0.006	88.976
ZN	proportion of residential land zoned for lots over 25,000 sq.ft.	0.000	100.000
INDUS	proportion of non-retail business acres per town	0.460	27.740
CHAS	Charles River dummy variable (=1 if tract bounds river; 0 otherwise)	0.000	1.000
NOX	nitric oxides concentration (parts per 10 million)	0.385	0.871
RM	average number of rooms per dwelling	3.561	8.780
AGE	proportion of owner-occupied units built prior to 1940	2.900	100.000
DIS	weighted distances to five Boston employment centres	1.130	12.126
RAD	index of accessibility to radial highways	1.000	24.000
TAX	full-value property-tax rate per $10,000	187.000	711.000
PTRATIO	pupil-teacher ratio by town	12.600	22.000
B	$1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town	0.320	396.900
LSTAT	% lower status of the population	1.730	37.970
MEDV	Log Median value of owner-occupied homes in $1000's	1.609	3.912

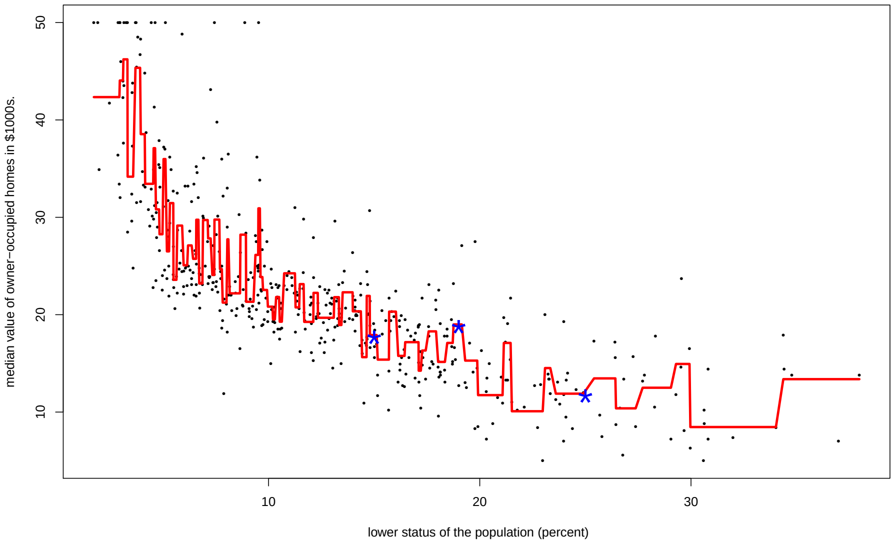
[1] Harrison and Rubinfeld (1978) Hedonic prices and the demand for clean air. *Journal of Environmental Economic and Management*, 5, 81-102.

Let us try $p = 1$

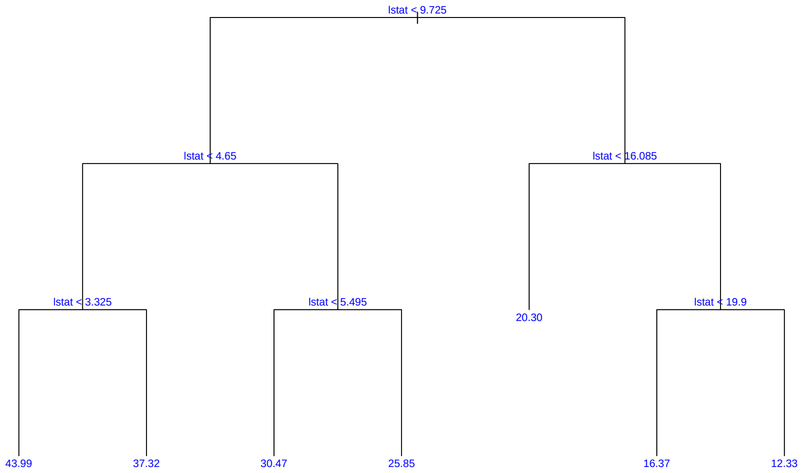
`lstat`: lower status of the population (percent)



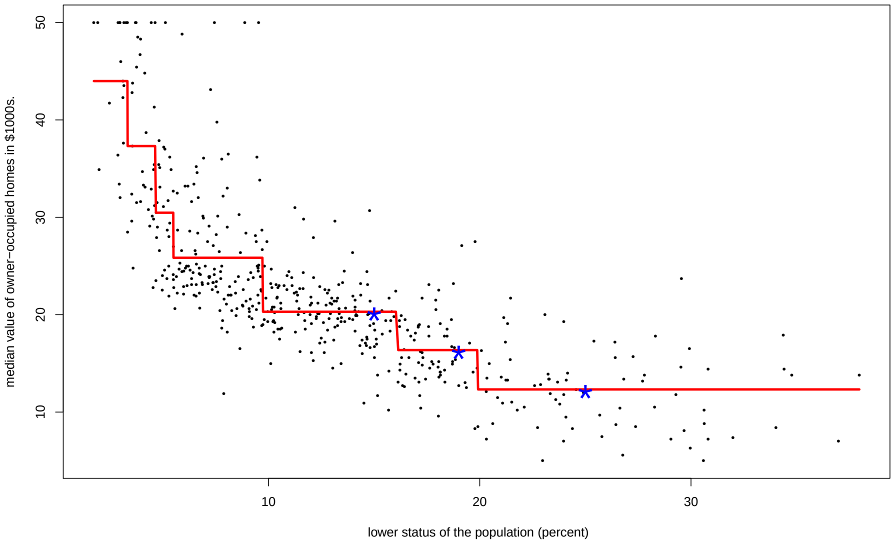
Fitted model



Pruned tree


lstat < 9.725
lstat < 4.65
lstat < 16.085
lstat < 3.325
lstat < 5.495
20.30
lstat < 19.9
43.99
37.32
30.47
25.85
16.37
12.33

Fitted model

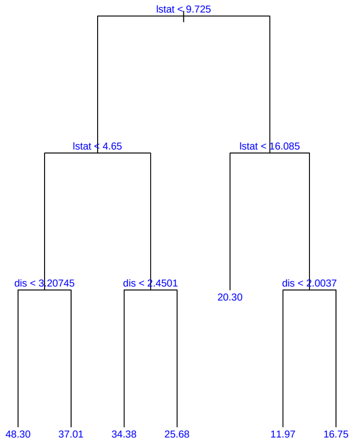


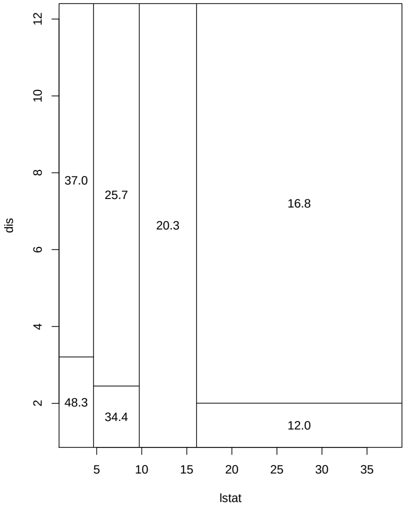
R code

```
install.packages("tree")
library(tree)
library(MASS)
attach(Boston)

# Fitting a large tree
# -----------------------------
temp.tree = tree(medv~lstat,data=Boston,mindev=.0001)
plot(temp.tree,type="uniform")
text(temp.tree,col="blue",label=c("yval"),cex=.3)

temp.fit = predict(temp.tree)
oo=order(lstat)
preddf = data.frame(lstat=c(15,19,25))
yhat = predict(temp.tree,preddf)

plot(lstat,medv,cex=.5,pch=16)
lines(lstat[oo],temp.fit[oo],col="red",lwd=3)
points(preddf$lstat,yhat,col="blue",pch="*",cex=3)

# Pruning the large tree
# -----------------------------
boston.tree=prune.tree(temp.tree,best=7)
plot(boston.tree,type="uniform")
text(boston.tree,col="blue",label=c("yval"),cex=.8)

boston.fit = predict(boston.tree)
oo=order(lstat)
preddf = data.frame(lstat=c(15,19,25))
yhat = predict(boston.tree,preddf)

plot(lstat,medv,cex=.5,pch=16)
lines(lstat[oo],boston.fit[oo],col="red",lwd=3)
points(preddf$lstat,yhat,col="blue",pch="*",cex=3)
```

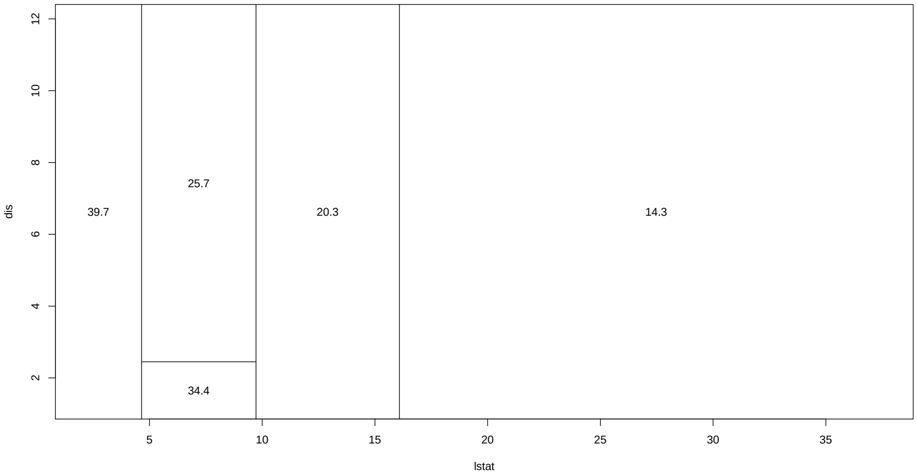
Let us try $p = 2$

`lstat`: lower status of the population (percent)
`dis`: weighted mean of distances to five Boston employment centres.

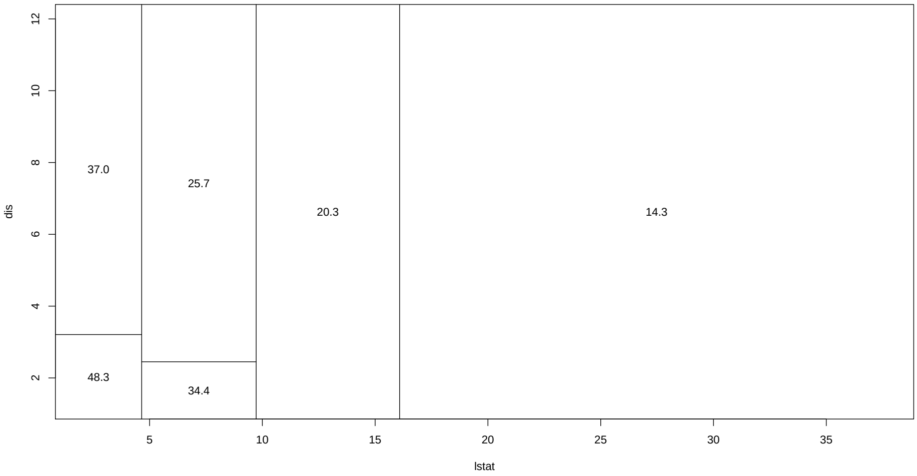




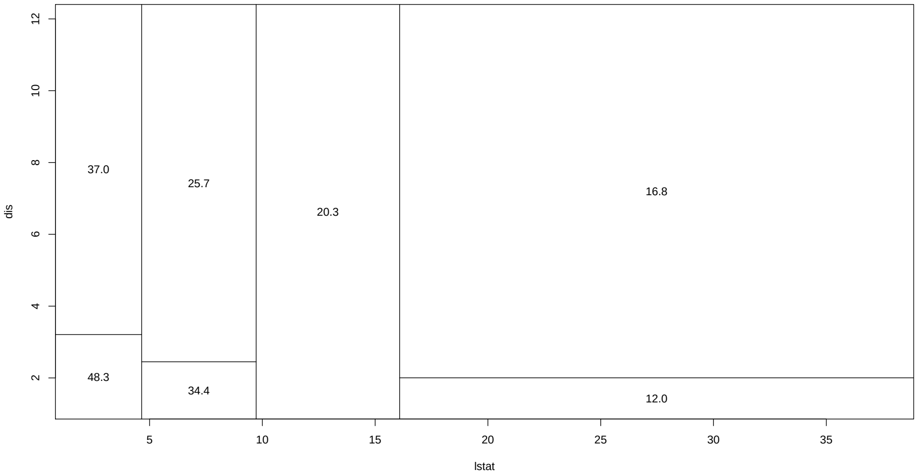
Terminal nodes: 5



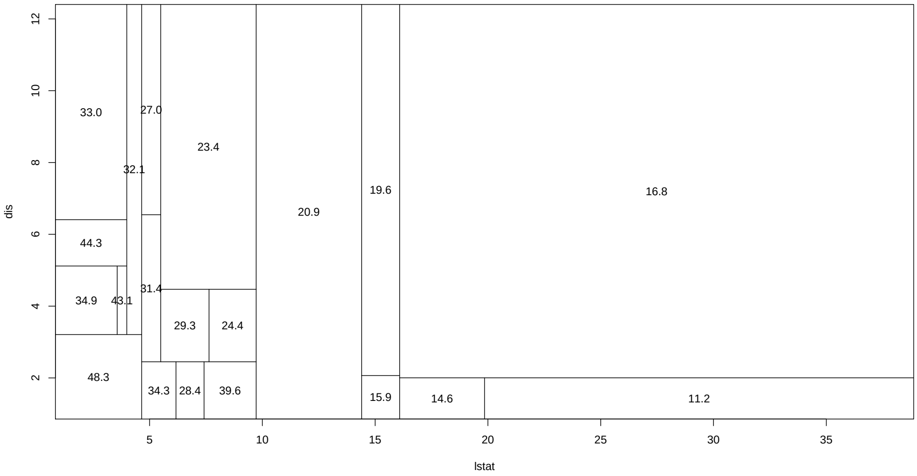
Terminal nodes: 6



Terminal nodes: 7



Terminal nodes: 20



R code

```
df2=Boston[,c(8,13,14)]

temp.tree = tree(medv˜.,df2,mindev=.0001)

boston.tree=prune.tree(temp.tree,best=7)

par(mfrow=c(1,2))

plot(boston.tree,type="u")

text(boston.tree,col="blue",label=c("yval"),cex=.8)

partition.tree(boston.tree)
```

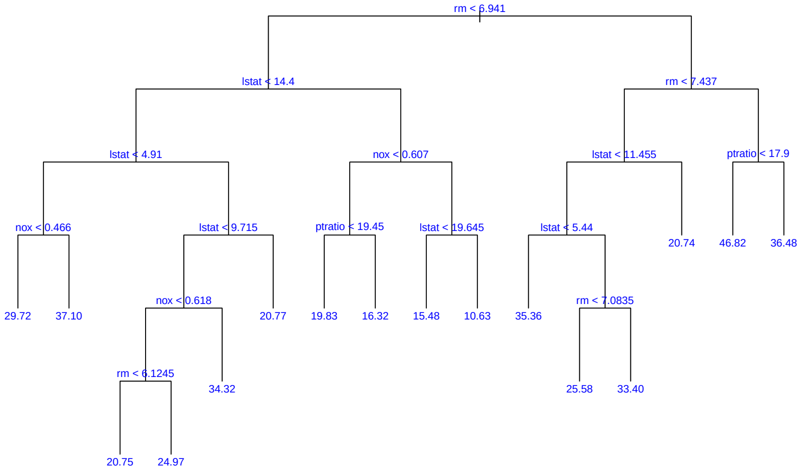
Comparing in-sample fits

```
boston.fit2 = predict(boston.tree)

fmat = cbind(medv,boston.fit,boston.fit2)

colnames(fmat)=c("y=medv","treel","treeld")

print(cor(fmat))
```

```
          y=medv     treel    treeld
y=medv 1.0000000 0.8338431 0.8560183
treel  0.8338431 1.0000000 0.9364702
treeld 0.8560183 0.9364702 1.0000000
```

Let us try $p = 4$

`nox`: nitrogen oxides concentration (parts per 10 million)
`rm`: average number of rooms per dwelling
`ptratio`: pupil-teacher ratio by town
`lstat`: lower status of the population (percent)

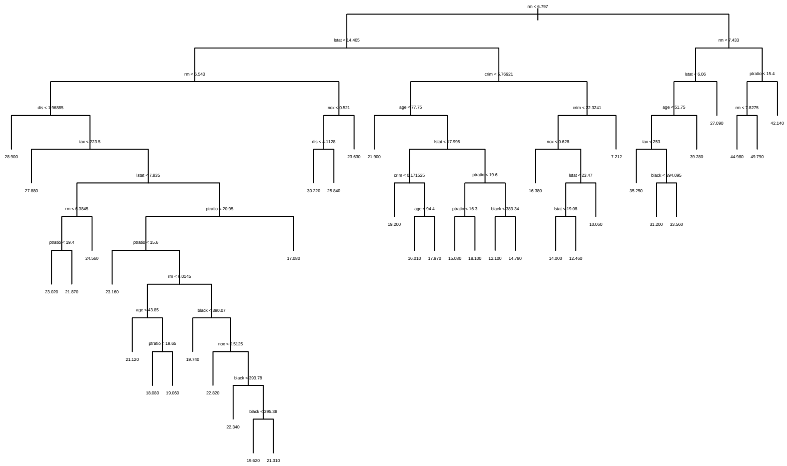


R code

```
df4=Boston[,c(5,6,11,13,14)]
temp = tree(medv~.,df4,mindev=.0001)
boston.tree4=prune.tree(temp,best=15)

par(mfrow=c(1,1))
plot(boston.tree4,type="u")
text(boston.tree4,col="blue",label=c("yval"),cex=.8)

fmat4=cbind(fmat,predict(boston.tree4))
colnames(fmat4)[4]="tree4"

print(cor(fmat4))

          y=medv     treel    treeld     tree4
y=medv 1.0000000 0.8338431 0.8560183 0.9266955
treel  0.8338431 1.0000000 0.9364702 0.8783099
treeld 0.8560183 0.9364702 1.0000000 0.8830659
tree4  0.9266955 0.8783099 0.8830659 1.0000000
```

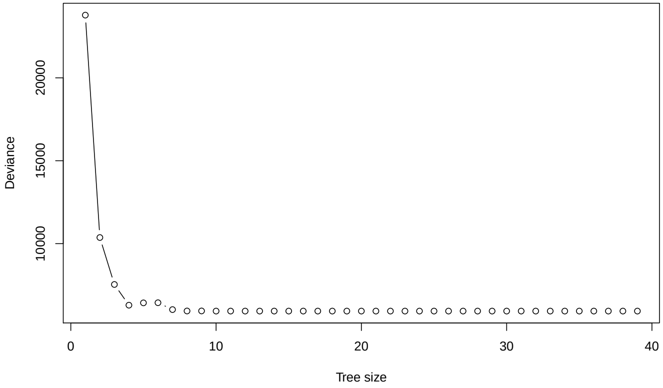
Tree with all $p = 13$ predictors

```
library (MASS)
set.seed (31415)
train = sample (1: nrow(Boston ), nrow(Boston )/2)
tree.boston = tree(medv~.,Boston ,subset =train,mindev=.0001)
plot(tree.boston,type="u")
text(tree.boston,pretty=0,cex=0.3)
```



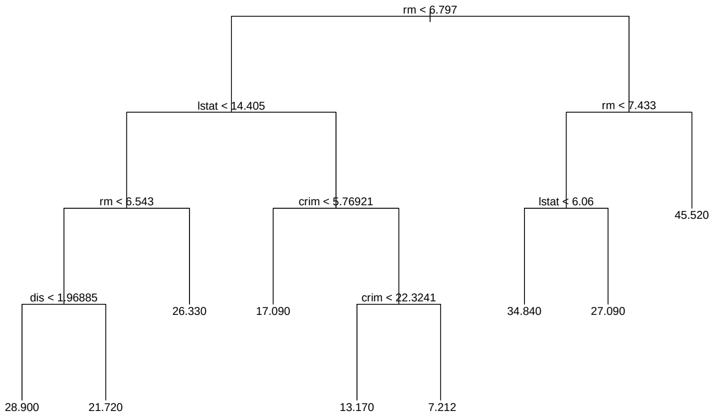
Cross-validation

```
cv.boston = cv.tree(tree.boston)
plot(cv.boston$size,cv.boston$dev,type="b")
```



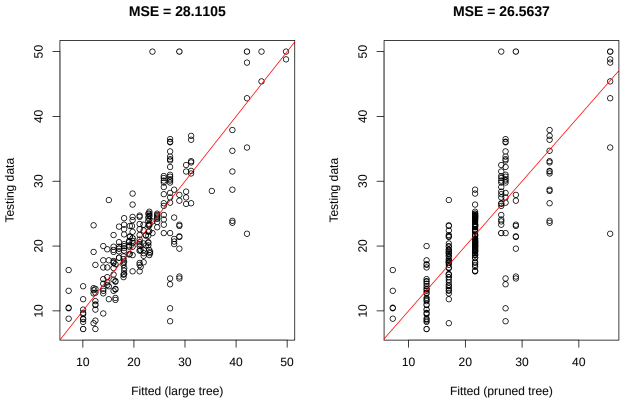
Pruned tree

```
prune.boston = prune.tree(tree.boston,best=9)
plot(prune.boston,type="u")
text(prune.boston,pretty=0)
```



Large and pruned trees

```
y        = Boston[-train,"medv"]
yhat    = predict(tree.boston,newdata=Boston[-train,])
yhat1 = predict(prune.boston,newdata=Boston[-train,])
corr(y,yhat) = 0.798, corr(y,yhat1) = 0.806, corr(yhat,yhat1) =  0.946

par(mfrow=c(1,2))
plot(yhat,boston.test,xlab="Fitted (large tree)",ylab="Testing data")
title(paste("MSE = ",round(mean((yhat-y)^2),4),sep=""))
abline(0,1,col=2)
plot(yhat1,boston.test,xlab="Fitted (pruned tree)",ylab="Testing data")
title(paste("MSE = ",round(mean((yhat1-y)^2),4),sep=""))
abline(0,1,col=2)
```



Outline

Trees versus linear models[2]

Regression and classification trees have a very different flavor from the more classical approaches for regression and classification.

In particular, linear regression assumes a model of the form

$$f(x) = \sum_{j=1}^{p} \beta_j x_j$$

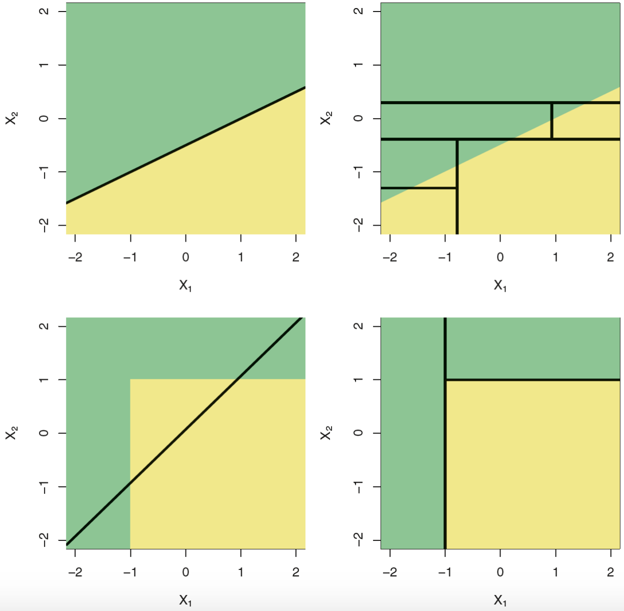
whereas regression trees assume a model of the form

$$f(x) = \sum_{m=1}^{M} c_m 1_{x \in R_m}$$

where $R_1, \ldots, R_m$ represent a partition of feature space.

[2]ISLR, Section 8.1.3

Trees versus linear models



Advantages of trees[3]

- Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches.
- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- Trees can easily handle qualitative predictors without the need to create dummy variables.

[3] ISLR, Section 8.1.4

Disadvantages of trees

- Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches.
- Additionally, trees can be very non-robust. In other words, a small change in the data can cause a large change in the final estimated tree.

Bagging, random forests and boosting

However, by aggregating many decision trees, using methods like bagging, random forests, and boosting, the predictive performance of trees can be substantially improved.

Outline

Bootstrap aggregation: bagging[4]

Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method.

A natural way to reduce the variance and hence increase the prediction accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions. In other words, we could calculate

$$\hat{f}^1(x), \hat{f}^2(x), \dots, \hat{f}^B(x)$$

using B separate training sets, and average them in order to obtain a single low-variance statistical learning model given by

$$\hat{f}_{avg}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(x)$$

[4] ISLR 8.2.1

Bagging

Of course, this is not practical because we generally do not have access to multiple training sets.

Instead, we can bootstrap, by taking repeated samples from the (single) training data set.

In this approach we generate B different bootstrapped training data sets.

We then train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$, and finally average all the predictions, to obtain

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

This is called bagging.

Bagging for regression trees

Construct B regression trees using B bootstrapped training sets, and average the resulting predictions.

These trees are grown deep, and are not pruned.

Hence each individual tree has high variance, but low bias.

Averaging these B trees reduces the variance.

Bagging has been demonstrated to give impressive improvements in accuracy by combining together hundreds or even thousands of trees into a single procedure.

Bagging the boston data

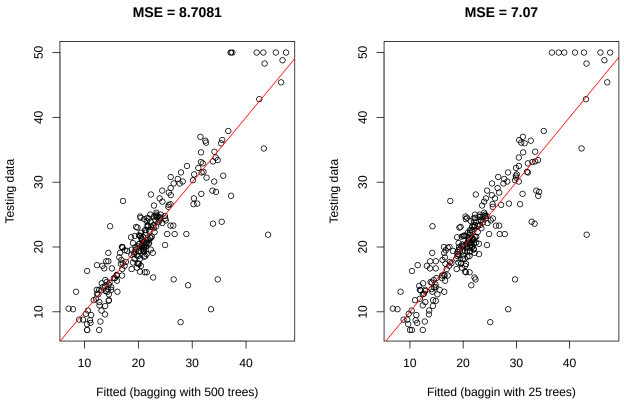
```
bag.boston = randomForest(medv~.,data=Boston,subset=train,mtry=13,importance=TRUE)
```

```
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 13

          Mean of squared residuals: 15.33724
                    % Var explained: 83.59
```

```
bag.boston = randomForest(medv~.,data=Boston,subset=train,mtry=13,ntree=25)
```

```
               Type of random forest: regression
                     Number of trees: 25
No. of variables tried at each split: 13

          Mean of squared residuals: 16.46512
                    % Var explained: 82.38
```

Fitted models



```
cor(cbind(boston.test,yhat,yhat1,yhat.bag,yhat.bag1))
            boston.test      yhat     yhat1  yhat.bag yhat.bag1
boston.test   1.0000000 0.7982040 0.8057710 0.8686673 0.8932177
yhat          0.7982040 1.0000000 0.9461125 0.9317866 0.9363935
yhat1         0.8057710 0.9461125 1.0000000 0.9357755 0.9395889
yhat.bag      0.8686673 0.9317866 0.9357755 1.0000000 0.9893311
yhat.bag1     0.8932177 0.9363935 0.9395889 0.9893311 1.0000000
```

R code

```
library(randomForest)
set.seed(31415)

bag.boston = randomForest(medv~.,data=Boston,subset=train,mtry=13,importance=TRUE)
bag.boston

bag.boston = randomForest(medv~.,data=Boston,subset=train,mtry=13,ntree=25)
bag.boston

yhat.bag = predict (bag.boston ,newdata =Boston [-train ,])
yhat.bag1 = predict (bag.boston ,newdata =Boston [-train ,])

par(mfrow=c(1,2))
plot(yhat.bag,boston.test,xlab="Fitted (bagging 1)",ylab="Testing data")
title(paste("MSE = ",round(mean((yhat-yhat.bag)^2),4),sep=""))
abline(0,1,col=2)

plot(yhat.bag1,boston.test,xlab="Fitted (pruned tree)",ylab="Testing data")
title(paste("MSE = ",round(mean((yhat1-yhat.bag1)^2),4),sep=""))
abline(0,1,col=2)

cor(cbind(boston.test,yhat,yhat1,yhat.bag,yhat.bag1))
```

Outline

Random forests[5]

Random forests provide an improvement over bagged trees by way of a random small tweak that decorrelates the trees.

As in bagging, we build a number forest of decision trees on bootstrapped training samples.

But when building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors.

The split is allowed to use only one of those m predictors. A fresh sample of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$.

[5]ISLR, Section 8.2.2

When does random forests beat bagging?

One very strong predictor + many moderately strong predictors.

Bagging:

- Most or all of the trees will use this strong predictor in the top split.
- All of the bagged trees will look quite similar to each other.
- The predictions from the bagged trees will be highly correlated.
- Averaging many highly correlated quantities does not reduce the variance.

Random forests:

- Random forests forces each split to consider only m predictors.
- On average $(p - m)/p$ of the splits will not consider the strong predictor.
- Other predictors will have more of a chance.
- This process decorrelates the trees.
- The average of the resulting trees less variable and hence more reliable.

Random Forest for Regression or Classification[6]

Algorithm 15.1 *Random Forest for Regression or Classification.*

```
1. For $b = 1$ to $B$:
    (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size $N$ from the training data.
    (b) Grow a random-forest tree $T_b$ to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{min}$ is reached.
        i. Select $m$ variables at random from the $p$ variables.
        ii. Pick the best variable/split-point among the $m$.
        iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.
```

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

[6]Hastie, Tibshirani and Friedman (2008) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*, page 588.

Out-of-bag error[7]

The out-of-bag (OOB) estimate, for response observation y_i in the training data, is obtained by averaging those $T_b(x_i)$ for which observation i was NOT in the bootstrap sample:

$$\hat{f}_{rf}^{(i)} = \frac{1}{B_i} \sum_{b: i \notin Z_b^*} T_b(x_i),$$

where B_i is the number of times observation i was not in the bootstrap sample.

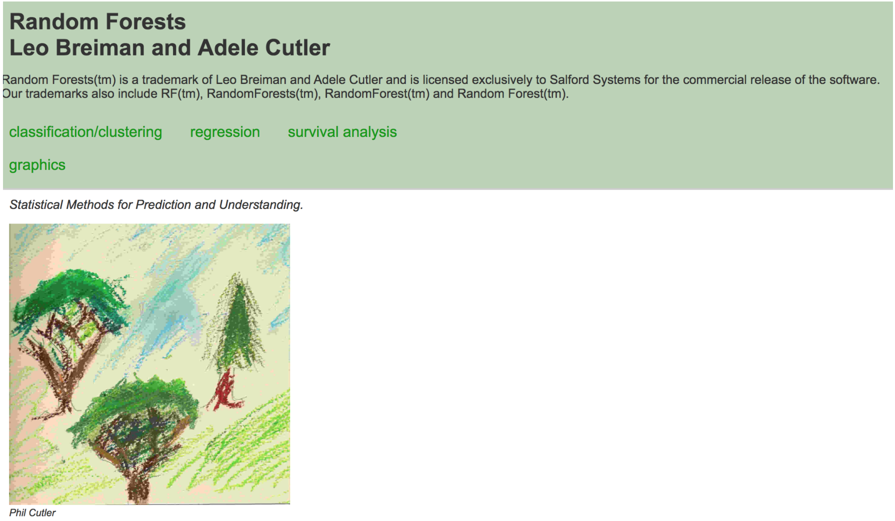
The overall out-of-bag (OOB) error estimate is

$$\text{err}_{OOB} = \frac{1}{N} \sum_{i=1}^{N} L[y_i, \hat{f}_{rf}^{(i)}]$$

- Random forests deliver cross-validated error estimates at virtually no extra cost.
- If B is sufficiently large (about three times the number needed for the random forest to stabilize), we can see that the OOB error estimate is equivalent to leave-one-out cross-validation error.

[7] Efron and Hastie (2016) *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*, page 327-330.

Adele Cutler's Random Forests page[8]



Random Forests
Leo Breiman and Adele Cutler

Random Forests(tm) is a trademark of Leo Breiman and Adele Cutler and is licensed exclusively to Salford Systems for the commercial release of the software. Our trademarks also include RF(tm), RandomForests(tm), RandomForest(tm) and Random Forest(tm).

classification/clustering   regression   survival analysis

graphics

Statistical Methods for Prediction and Understanding.

Phil Cutler

[8] http://www.math.usu.edu/∼adele/forests

Outline

Revisiting the spam dataset

Overall misclassification erros: 9.3% (tree) vs 5.7% (lasso with interactions)

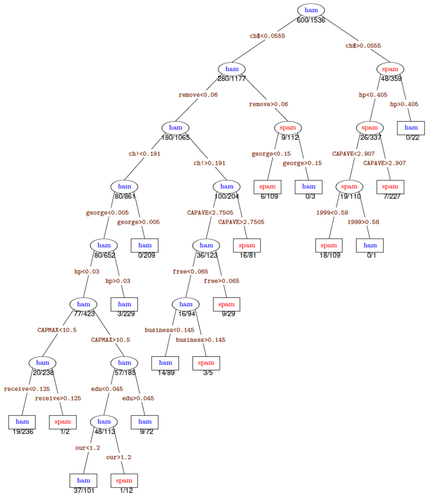


Figure 17.1 Regression tree fit to the binary `spam` data, a bigger version of Figure 8.7. The initial trained tree was far bushier than the one displayed; it was then *optimally* pruned using 10-fold cross-validation.

Test error

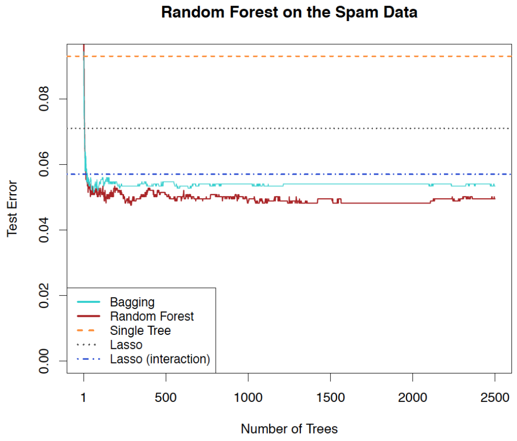


Figure 17.2 Test misclassification error of random forests on the `spam` data, as a function of the number of trees. The red curve selects $m = 7$ of the $p = 57$ features at random as candidates for the split variable, each time a split is made. The blue curve uses $m = 57$, and hence amounts to bagging. Both bagging and random forests outperform the lasso methods, and a single tree.

OOB error

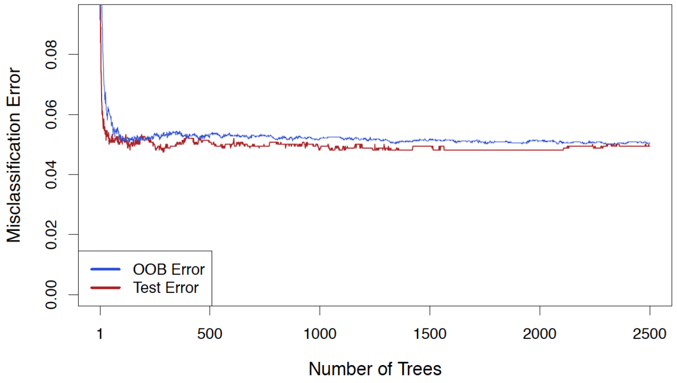


Figure 17.3 Out-of-bag misclassification error estimate for the `spam` data (blue) versus the test error (red), as a function of the number of trees.

Variable-importance plots

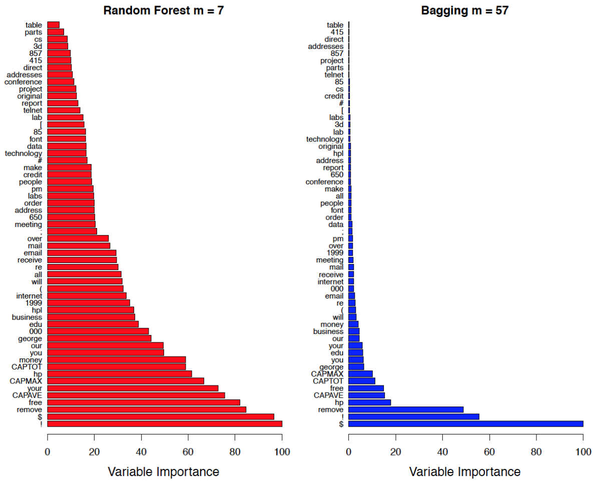
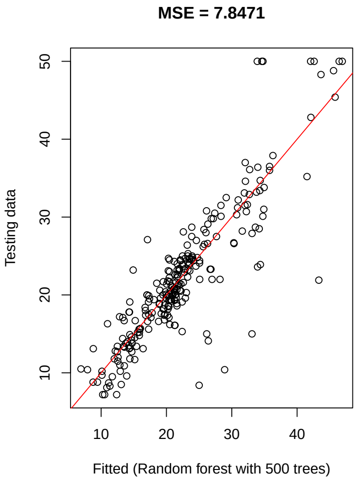
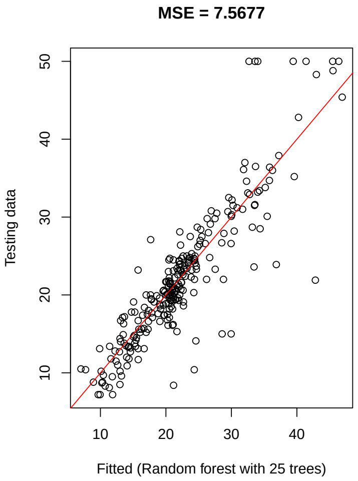


Figure 17.5 Variable-importance plots for random forests fit to the spam data. On the left we have the $m = 7$ random forest; due to the split-variable randomization, it spreads the importance among the variables. On the right is the $m = 57$ random forest or bagging, which focuses on a smaller subset of the variables.

Random forest: the `boston` data

`randomForest()` defaults:
$m = p/3$ variables (regression trees)
$m = \sqrt{p}$ variables (classification trees)





R code

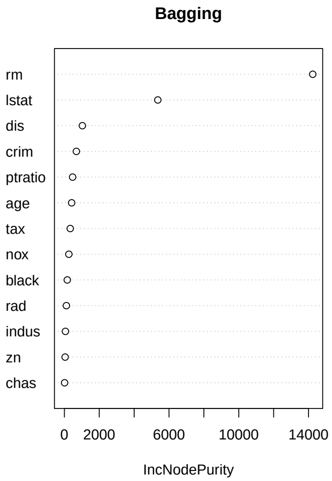
```
set.seed(31415)
rf.boston = randomForest(medv~.,data=Boston,subset=train,mtry=10,importance=TRUE)
rf.boston
yhat.rf = predict (rf.boston ,newdata =Boston [-train ,])

rf.boston = randomForest(medv~.,data=Boston,subset=train,mtry=10,ntree=25,importance=TRUE)
rf.boston
yhat.rf1 = predict (rf.boston ,newdata =Boston [-train ,])



par(mfrow=c(1,2))
plot(yhat.rf,boston.test,xlab="Fitted (Random forest with 500 trees)",ylab="Testing data")
title(paste("MSE = ",round(mean((yhat-yhat.rf)^2),4),sep=""))
abline(0,1,col=2)

plot(yhat.rf1,boston.test,xlab="Fitted (Random forest with 25 trees)",ylab="Testing data")
title(paste("MSE = ",round(mean((yhat1-yhat.rf1)^2),4),sep=""))
abline(0,1,col=2)


round(cor(cbind(boston.test,yhat,yhat1,yhat.bag,yhat.bag1,yhat.rf,yhat.rf1)),4)
            boston.test   yhat  yhat1 yhat.bag yhat.bag1 yhat.rf yhat.rf1
boston.test      1.0000 0.7982 0.8058   0.8687    0.8932  0.8801   0.8754
yhat             0.7982 1.0000 0.9461   0.9318    0.9364  0.9367   0.9306
yhat1            0.8058 0.9461 1.0000   0.9358    0.9396  0.9431   0.9401
yhat.bag         0.8687 0.9318 0.9358   1.0000    0.9893  0.9963   0.9874
yhat.bag1        0.8932 0.9364 0.9396   0.9893    1.0000  0.9902   0.9802
yhat.rf          0.8801 0.9367 0.9431   0.9963    0.9902  1.0000   0.9939
yhat.rf1         0.8754 0.9306 0.9401   0.9874    0.9802  0.9939   1.0000
>
```

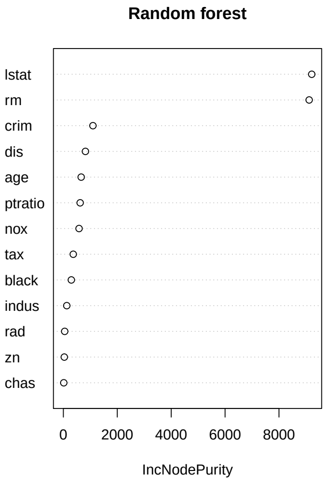
IncNodePurity

"`IncNodePurity` *is a measure of the total decrease in node impurity that results from splits over that variable, averaged over all trees. In the case of regression trees, the node impurity is measured by the training RSS, and for classification trees by the deviance.*" (ISLR, page 330)

```
cbind(importance(bag.boston),importance(rf.boston))
        IncNodePurity IncNodePurity
crim        686.54263    1096.69409
zn           42.13764      34.86653
indus        59.16361     126.10467
chas         12.19279      15.02267
nox         255.99583     582.10465
rm        14242.36653    9118.38936
age         416.35425     661.10425
dis        1032.03573     818.34677
rad         113.72632      49.43791
tax         335.51800     367.36610
ptratio     471.69760     623.26081
black       165.22122     295.30598
lstat      5357.56649    9214.00802
```

Variable importance





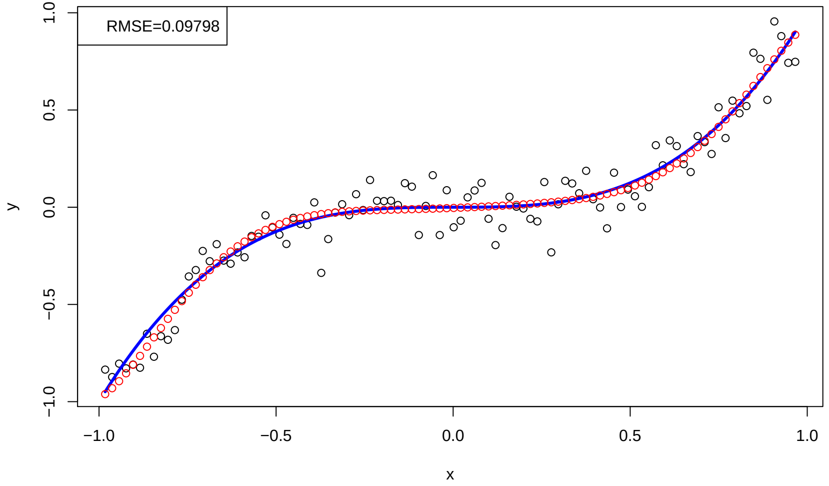
"The results indicate that across all of the trees considered in the random forest, the wealth level of the community (lstat) and the house size (rm) are by far the two most important variables." (ISLR, page 330)

Outline

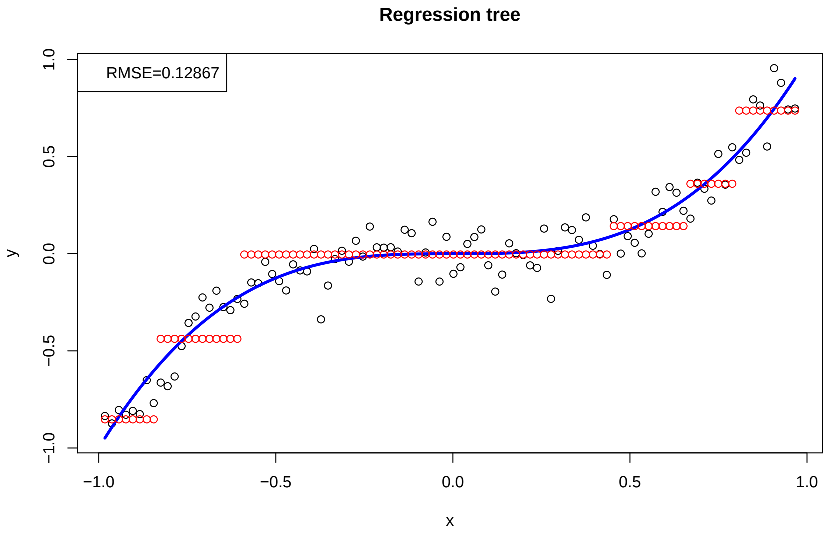
Simulated exercise

$n = 200$ (100 for training and 100 for testing)
$x_i \sim U(-1, 1)$
$y_i \sim N(x_i^3, 0.1^2)$

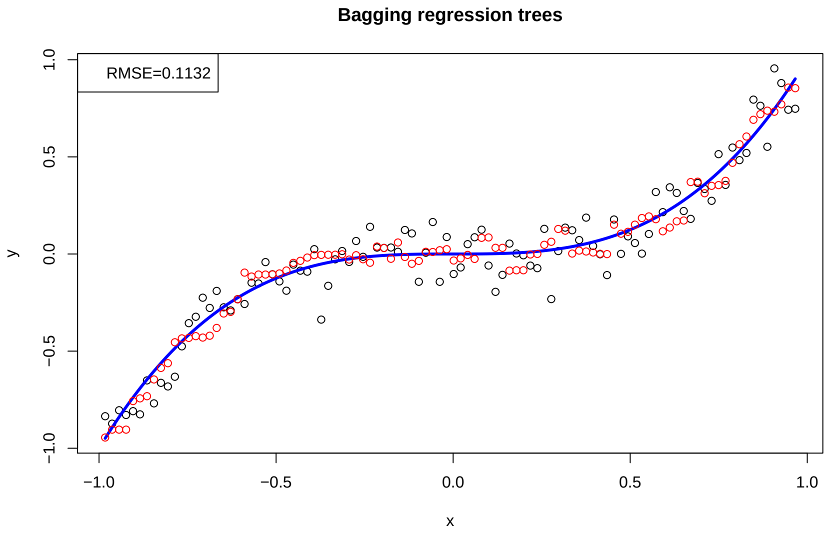
OLS − polynomial regression



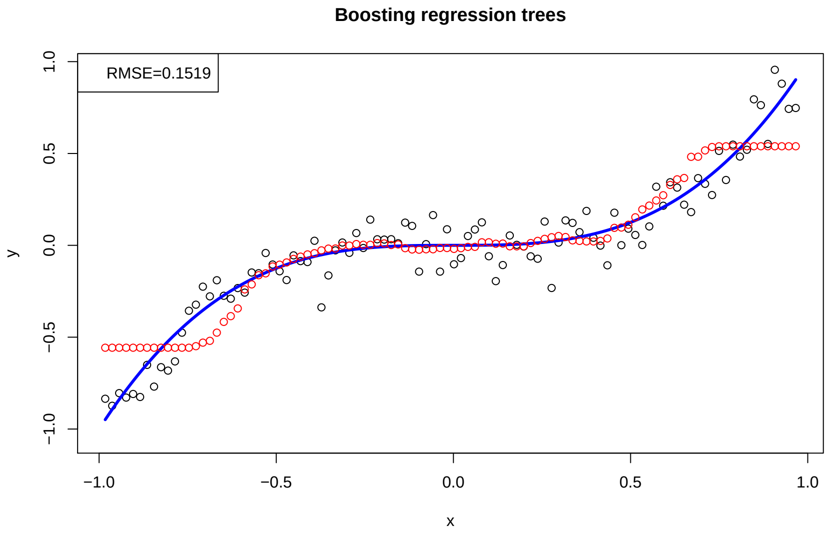
Regression tree



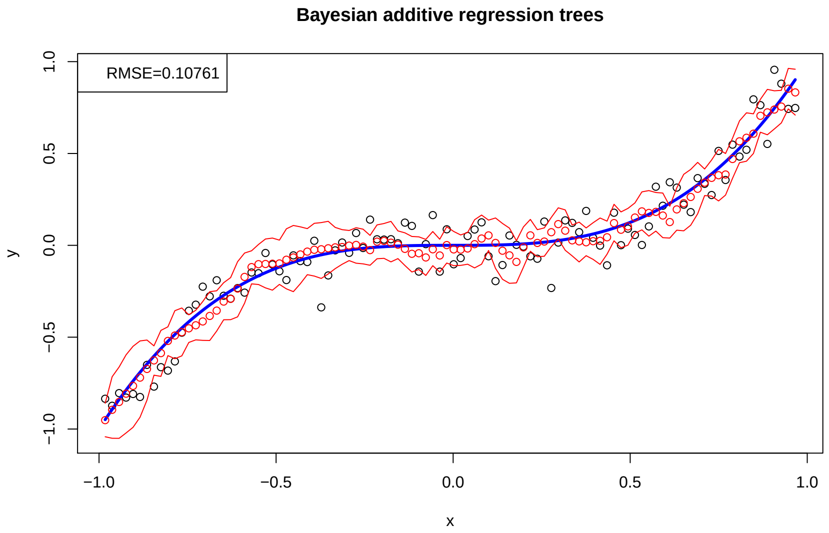
Bagging regression trees



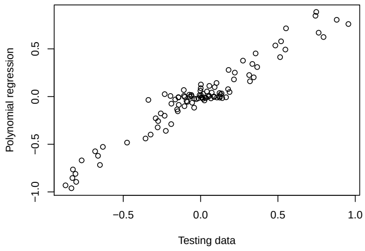
Boosting regression trees

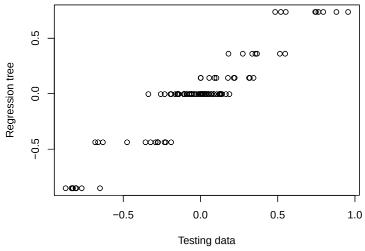


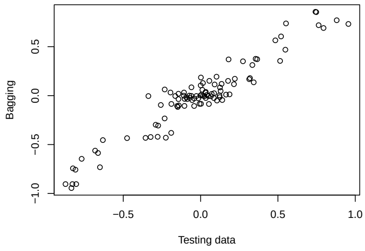
Bayesian additive regression trees

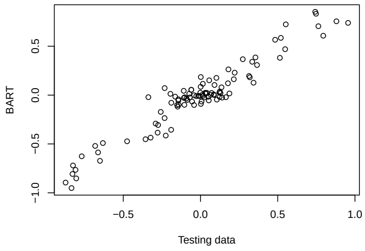


True versus fitted



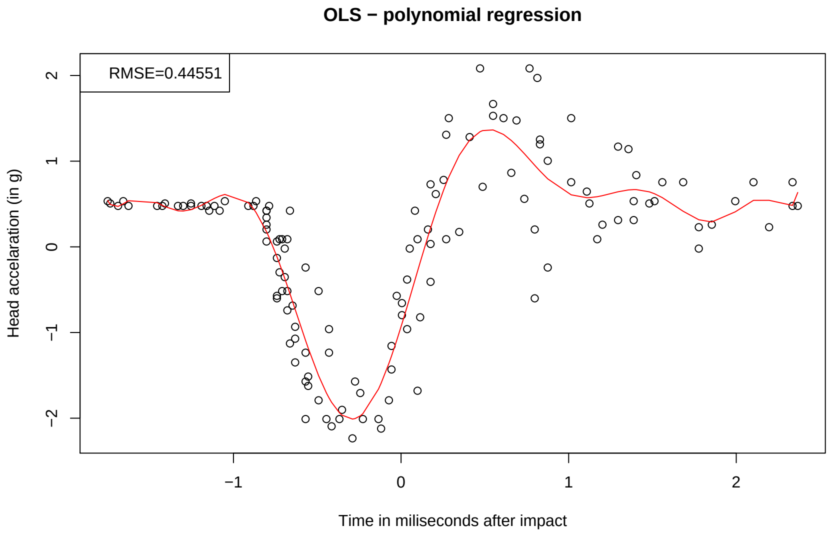




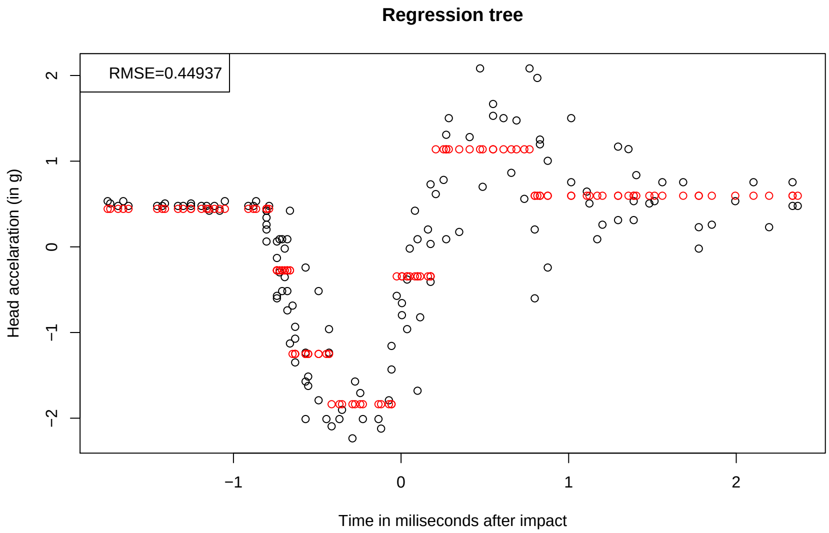


Outline

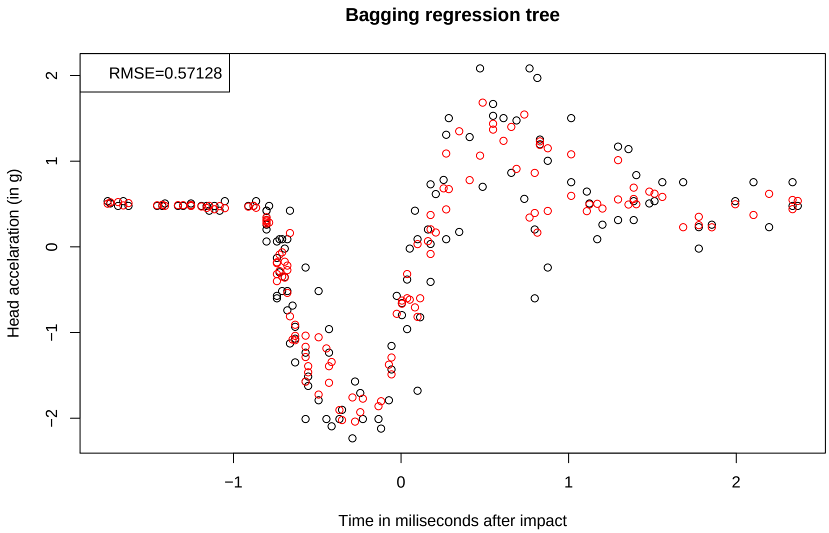
Revisiting the `motorcycle` data



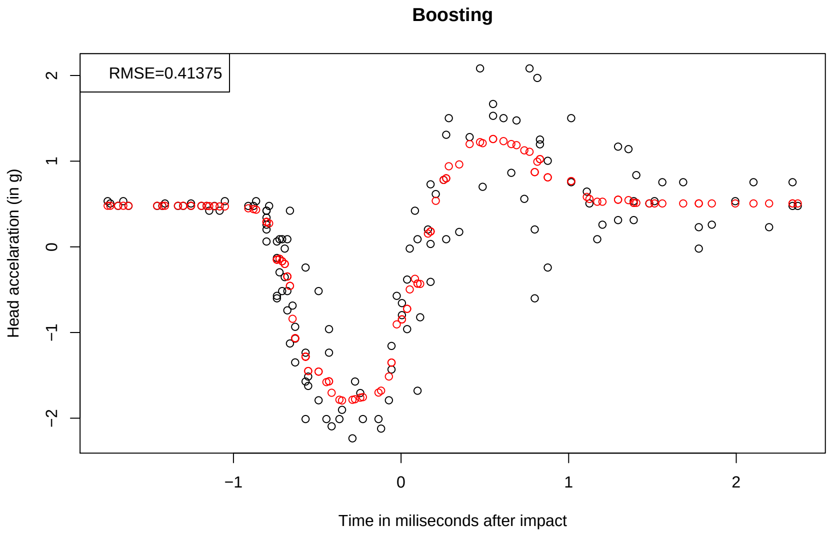
Regression tree



Bagging regression tree

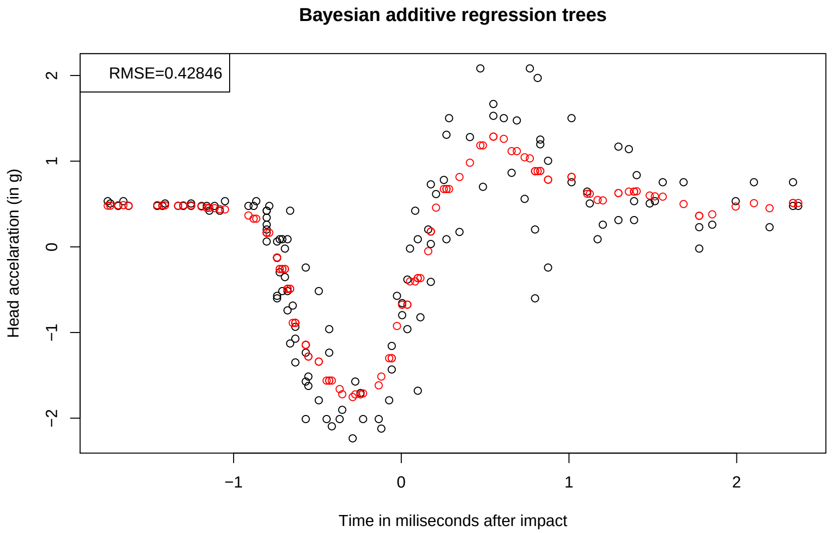


Boosting



Boosting
RMSE=0.41375
Head accelaration (in g)
Time in miliseconds after impact
−2
−1
0
1
2

BART



Bayesian additive regression trees
RMSE=0.42846
Head accelaration (in g)
Time in miliseconds after impact
−2
−1
0
1
2
−1
0
1
2

Outline

The ALS data[9]

These data represent measurements on patients with amyotrophic lateral sclerosis (Lou Gehrig's disease). The goal is to predict the rate of progression of an ALS functional rating score (FRS). There are 1197 training measurements on 369 predictors and the response, with a corresponding test set of size 625 observations.

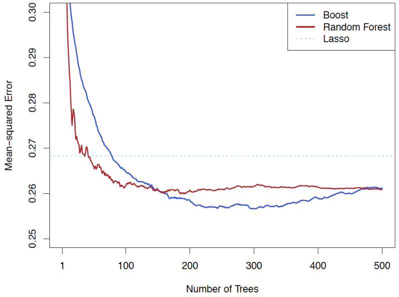


Figure 17.6 Test performance of a boosted regression-tree model fit to the **ALS** training data, with $n = 1197$ and $p = 369$. Shown is the mean-squared error on the 625 designated test observations, as a function of the number of trees. Here the depth $d = 4$ and $\epsilon = 0.02$. Boosting achieves a lower test MSE than a random forest. We see that as the number of trees B gets large, the test error for boosting starts to increase—a consequence of overfitting. The random forest does not overfit. The dotted blue horizontal line shows the best performance of a linear model, fit by the lasso. The differences are less dramatic than they appear, since the vertical scale does not extend to zero.

[9] Computer Age Statistical Inference, pages 334-337.

Variable-importance plot

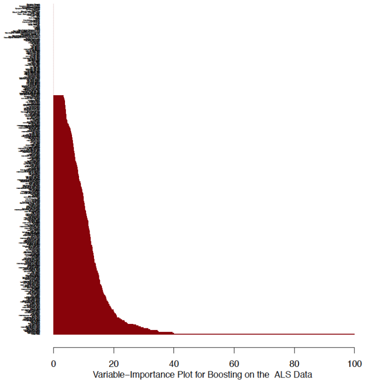
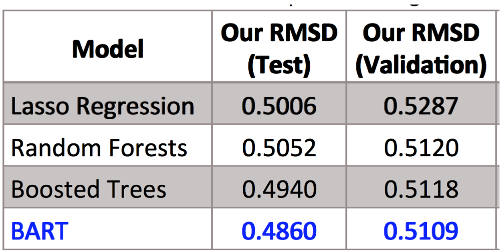


Figure 17.7 Variable importance plot for the `ALS` data. Here 267 of the 369 variables were used in the ensemble. There are too many variables for the labels to be visible, so this plot serves as a visual guide. Variable `Onset.Delta` has relative importance 100 (the lowest red bar), more than double the next two at around 40 (`last.slope.weight` and `alsfrs.score.slope`). However, the importances drop off slowly, suggesting that the model requires a significant fraction of the variables.

Only 267 of the 369 variables were ever used, with one variable `Onset.Delta` standing out ahead of the others. This measures the amount of time that has elapsed since the patient was first diagnosed with ALS, and hence a larger value will indicate a slower progression rate.

The DREAM challenge[10]

The ALS data were kindly provided by Lester Mackey and Lilly Fang, who won the DREAM challenge prediction prize in 2012 (Kuffner et al., 2015). It includes some additional variables created by them.
Their winning entry used Bayesian trees, not too different from random forests.



Model	Our RMSD (Test)	Our RMSD (Validation)
Lasso Regression	0.5006	0.5287
Random Forests	0.5052	0.5120
Boosted Trees	0.4940	0.5118
BART	0.4860	0.5109

[10]https://web.stanford.edu/ lmackey/papers/alsprize4life-slides.pdf

The DREAM challenge[11]

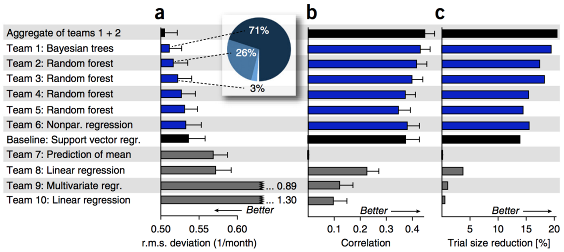


Figure 2 Performance of methods. (**a**,**b**) We compared the approaches of the ten teams that submitted executable R code in the final phase of the challenge and a baseline approach designed by the challenge organizers. All the solvers' algorithms had to be compatible with R version 2.13.1. The teams are numbered according to their ranking in r.m.s. deviation performance (**a**) or Pearson Correlation (**b**). They are colored blue if they performed better than, and gray if they performed worse than, the baseline. In addition, an aggregate of the predictions of teams 1 and 2 is shown. Whiskers indicate bootstrapped s.d. (inset). The frequency with which methods were ranked first is estimated across different bootstrap samples of patients. Teams 1 and 2 were ranked first in 71% and 26%, respectively, of the bootstrap samples (percentage rounded to the nearest integer). (**d**) By simulation, we estimated to what extent clinical trials can be reduced in size by each of the participating approaches corresponding to their improved prediction of disease progression.

Küffner, Zach, Norel, Hawe, Schoenfeld, Wang, Li, Fang, Mackey, Hardiman, Cudkowicz, Sherman, Ertaylan, Grosse-Wentrup, Hothorn, van Ligtenberg, Macke, Meyer, Schölkopf, Tran, Vaughan, Stolovitzky and Leitner (2014) Crowdsourced analysis of clinical trial data to predict amyotrophic lateral sclerosis progression. *Journal Nature Biotechnology*, 33, 51.

[11] http://neurology.arizona.edu/sites/default/files/dr_scherer-nbt_3051.pdf

Institutions

1 Institute of Bioinformatics and Systems Biology, German Research Center for Environmental Health, Munich, Germany.

2 Department of Informatics, Ludwig-Maximilians-University, Munich, Germany.

3 Prize4Life, Tel Aviv, Israel and Cambridge, Massachusetts, USA.

4 IBM T.J. Watson Research Center, Yorktown Heights, New York, USA.

5 MGH Biostatistics Center, Massachusetts General Hospital, Boston, Massachusetts, USA.

6 Harvard Medical School, Charlestown, Massachusetts, USA.

7 Sentrana Inc., Washington, DC, USA.

8 Latham&Watkins LLP, Silicon Valley, California, USA.

9 Department of Statistics, Stanford University, Stanford, California, USA.

10 Department of Neuroscience, Beaumont Hospital and Trinity College Dublin, Dublin, Ireland.

11 Neurological Clinical Research Institute, Massachusetts General Hospital, Charlestown, Massachusetts, USA.

12 Luxembourg Centre for Systems Biomedicine, University of Luxembourg, Esch/Alzette, Luxembourg.

13 Max Planck Institute for Intelligent Systems, Tübingen, Germany.

14 Institute of Social- and Preventive Medicine, University of Zïrich, Zürich, Switzerland.

15 Orca XL Problem Solvers, Amsterdam, the Netherlands.

16 Max Planck Institute for Biological Cybernetics and Bernstein Center for Computational Neuroscience, Tübingen, Germany.

17 Berkeley School of Public Health, University of California, Berkeley, California, USA.

18 ALS Innovation Hub, Biogen Idec, Cambridge, Massachusetts, USA.

Outline

Bibliographic Notes[12]

Random forests as described here were introduced by Breiman (2001), although many of the ideas had cropped up earlier in the literature in different forms.

Notably Ho (1995) introduced the term “random forest,” and used a consensus of trees grown in random subspaces of the features.

The idea of using stochastic perturbation and averaging to avoid overfitting was introduced by Kleinberg (1990), and later in Kleinberg (1996).

Amit and Geman (1997) used randomized trees grown on image features for image classification problems.

Breiman (1996) introduced bagging, a precursor to his version of random forests.

Dietterich (2000) also proposed an improvement on bagging using additional randomization. His approach was to rank the top 20 candidate splits at each node, and then select from the list at random. He showed through simulations and real examples that this additional randomization improved over the performance of bagging.

Friedman and Hall (2007) showed that sub-sampling (without replacement) is an effective alternative to bagging. They showed that growing and averaging trees on samples of size $N/2$ is approximately equivalent (in terms bias/variance considerations) to bagging, while using smaller fractions of N reduces the variance even further (through decorrelation).

[12] ESL, page 602.

References

Breiman (2001) Random forests. *Machine Learning*, 45, 5-32.

Ho (1995) Random decision forests. In Kavavaugh and Storms (eds), *Proc. Third International Conference on Document Analysis and Recognition*, Vol. 1, IEEE Computer Society Press, New York, pp. 278-282.

Kleinberg (1990) Stochastic discrimination. *Annals of Mathematical Artificial Intelligence*, 1, 207-239.

Kleinberg (1996) An overtraining-resistant stochastic modeling method for pattern recognition. *Annals of Statistics*, 24, 2319-2349.

Amit and Geman (1997) Shape quantization and recognition with randomized trees. *Neural Computation*, 9, 1545-1588.

Breiman (1996) Bagging predictors. *Machine Learning*, 26, 123-140.

Dietterich (2000) An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning*, 40(2), 139-157.

Friedman and Hall (2007) On bagging and nonlinear estimation. *Journal of Statistical Planning and Inference*, 137, 669-683.

Criminisi, Shotton and Konukoglu (2011) Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2-3), 81-227.

Denil, Matheson and de Freitas (2013) Consistency of Online Random Forests

Lakshminarayanan, Roy and Teh (2015) Mondrian Forests: Efficient Online Random Forests