24) Bagging, Random Forests, and Boosting

Vitor Kamada

March 2018

Reference

Tables, Graphics, and Figures from

An Introduction to Statistical Learning

James et al. (2017): Chapters: 8.2, 8.3.3, and 8.3.4

Bagging

Set of n independent observations $Z_1, \ldots, Z_n$, each with variance σ^2

$$Var(\bar{Z}) = \frac{\sigma^2}{n}$$

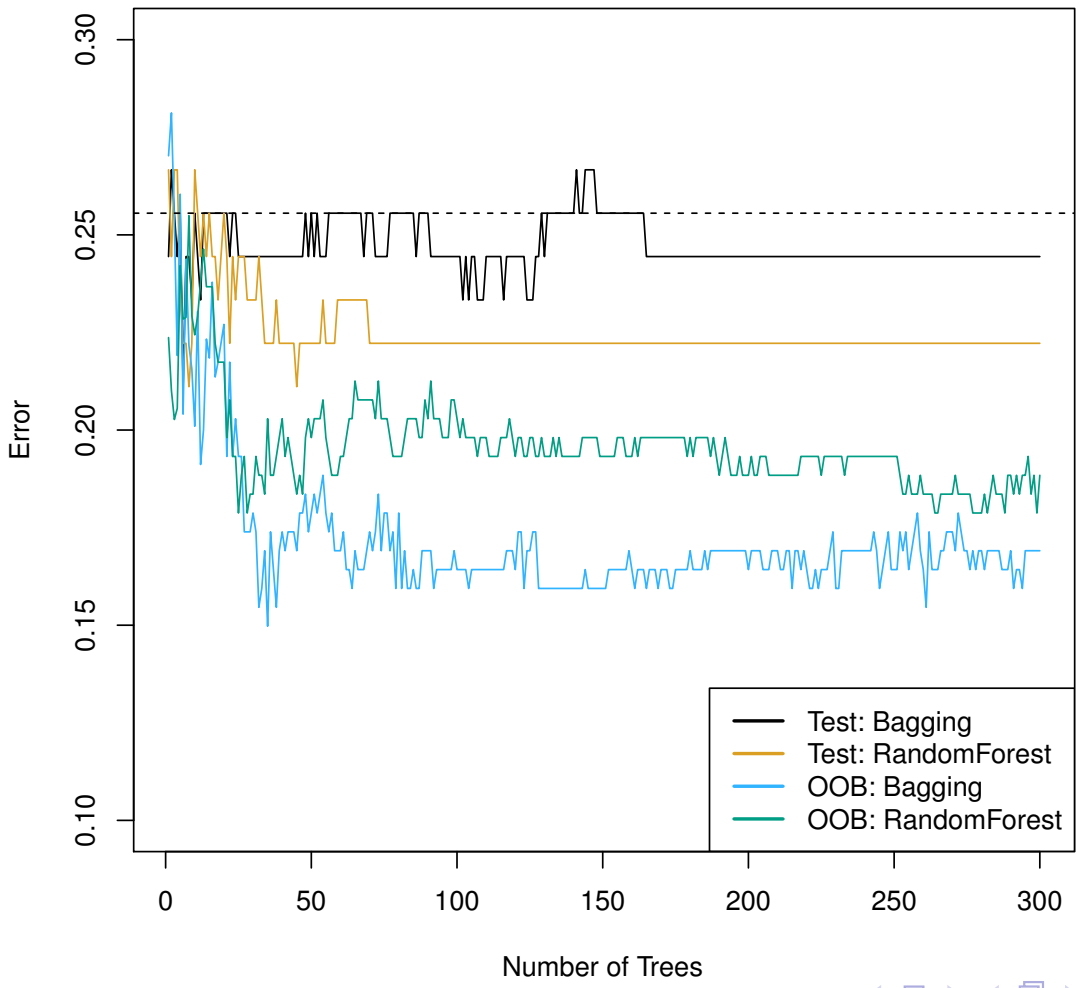
$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x)$$

Out-of-Bag Error Estimation (OOB)

Each bagged tree makes use $\frac{2}{3}$ of the observations

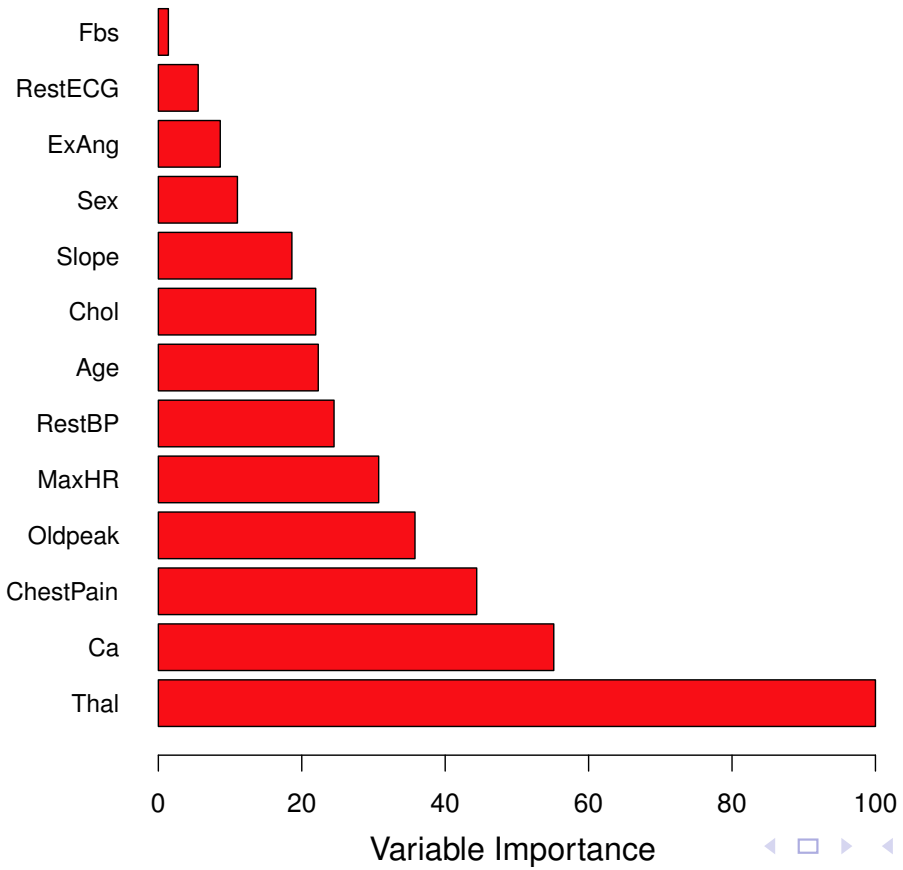
$\frac{1}{3}$ of the observations not used to fit a bagged tree are the out-of-bag (OOB) observations

Heart Data: Single Classification Tree vs Bagging, Random Forest, and Out-of-Bag



The Mean Decrease in Gini Index for each Variable, relative to the Largest

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$



Random Forests

The split only uses m predictors ($m \approx \sqrt{p}$)

Bagging: $m = p$

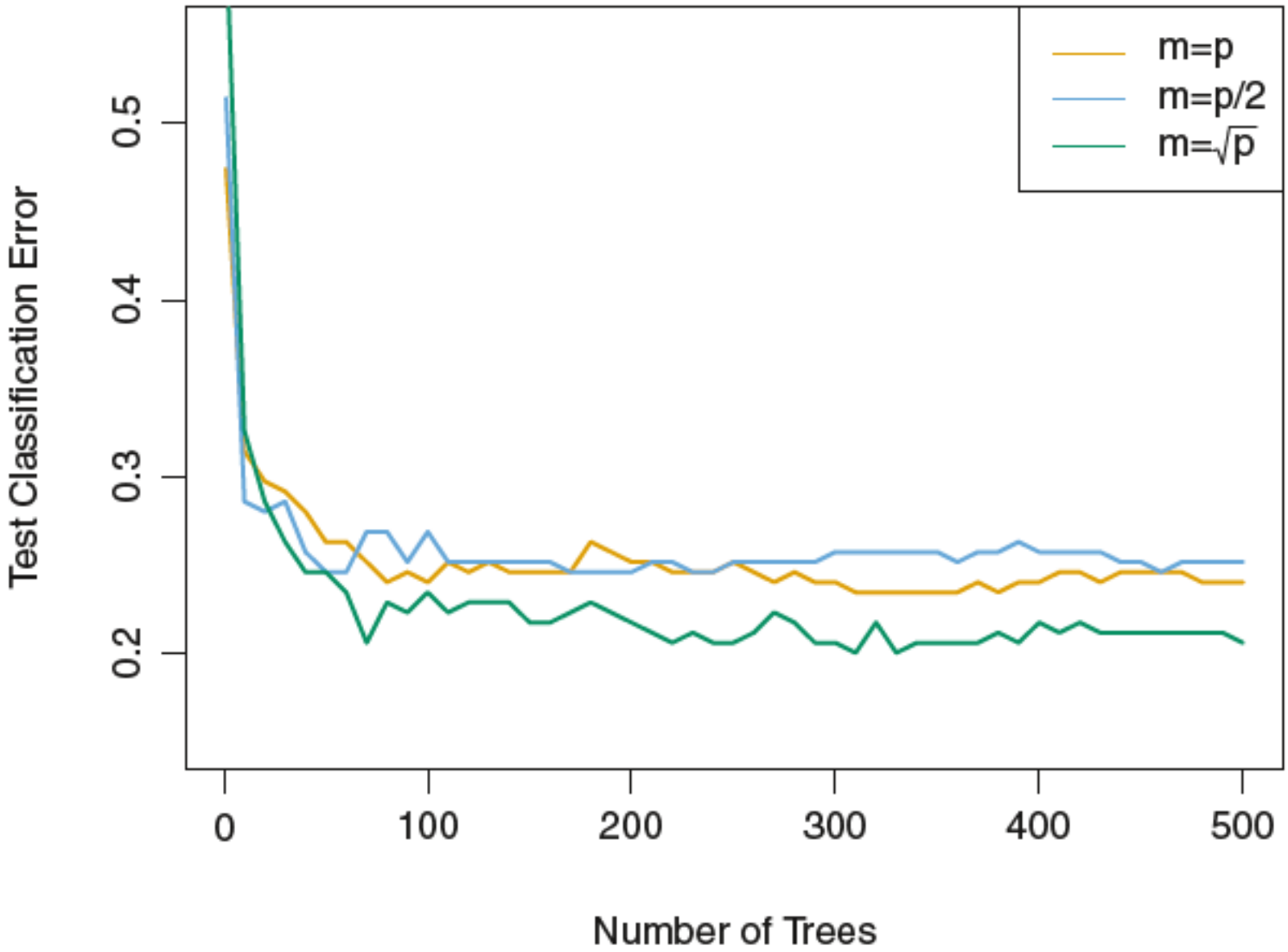
$$Var(X_1 + X_2)$$

$$Var(X_1) + Var(X_2) + 2Cov(X_1, X_2)$$

$$4Var(X) \text{ or } 2Var(X)$$

Gene Expression Data Set with 500 Predictors

A single tree has an error rate of 45.7%



library(MASS); library(stargazer); stargazer(Boston)

Statistic	N	Mean	St. Dev.	Min	Max
crim	506	3.614	8.602	0.006	88.976
zn	506	11.364	23.322	0.000	100.000
indus	506	11.137	6.860	0.460	27.740
chas	506	0.069	0.254	0	1
nox	506	0.555	0.116	0.385	0.871
rm	506	6.285	0.703	3.561	8.780
age	506	68.575	28.149	2.900	100.000
dis	506	3.795	2.106	1.130	12.127
rad	506	9.549	8.707	1	24
tax	506	408.237	168.537	187	711
ptratio	506	18.456	2.165	12.600	22.000
black	506	356.674	91.295	0.320	396.900
lstat	506	12.653	7.141	1.730	37.970
medv	506	22.533	9.197	5.000	50.000

library(randomForest); set.seed(1)

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

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

          Mean of squared residuals: 11.02509
                    % Var explained: 86.65
```

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

boston.test=Boston[-train,"medv"]

mean((yhat.bag-boston.test)^2)

13.43

set.seed(1)

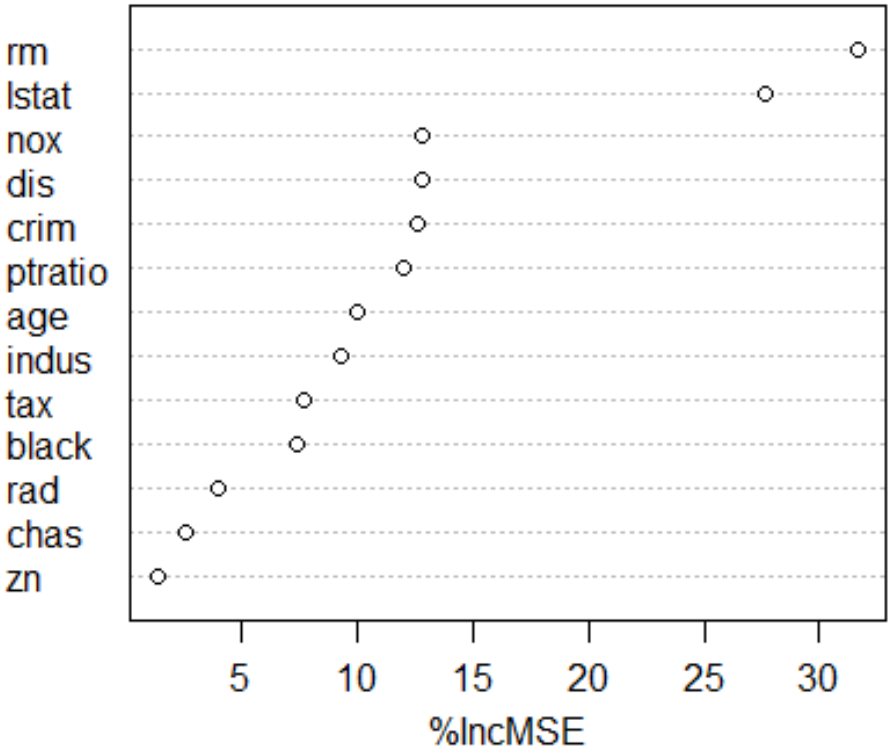
```
rf.boston=randomForest(medv~.,data=Boston,
subset=train,mtry=6,importance=TRUE)

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

mean((yhat.rf-boston.test)^2)
```

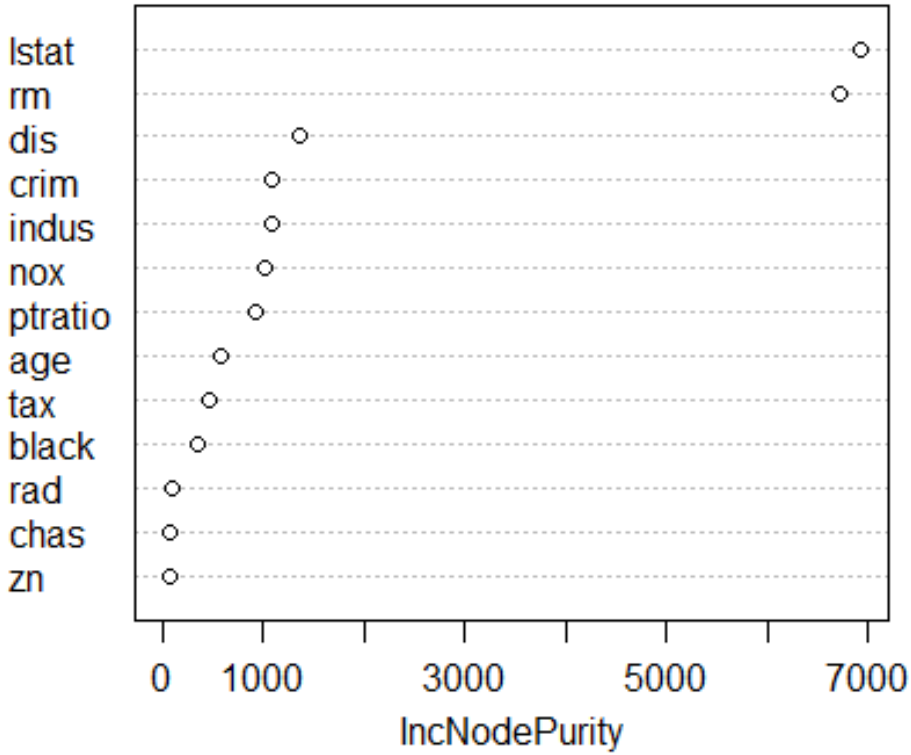
11.48

importance(rf.boston)

	%IncMSE	IncNodePurity
crim	12.547772	1094.65382
zn	1.375489	64.40060
indus	9.304258	1086.09103
chas	2.518766	76.36804
nox	12.835614	1008.73703
rm	31.646147	6705.02638
age	9.970243	575.13702
dis	12.774430	1351.01978
rad	3.911852	93.78200
tax	7.624043	453.19472
ptratio	12.008194	919.06760
black	7.376024	358.96935
lstat	27.666896	6927.98475

varImpPlot(rf.boston)





Boosting

For $b = 1, 2, ..., B$, repeat:

a) Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to the training data (X, r)

b) Update $\hat{f}$ by adding in a shrunken version of the new tree:

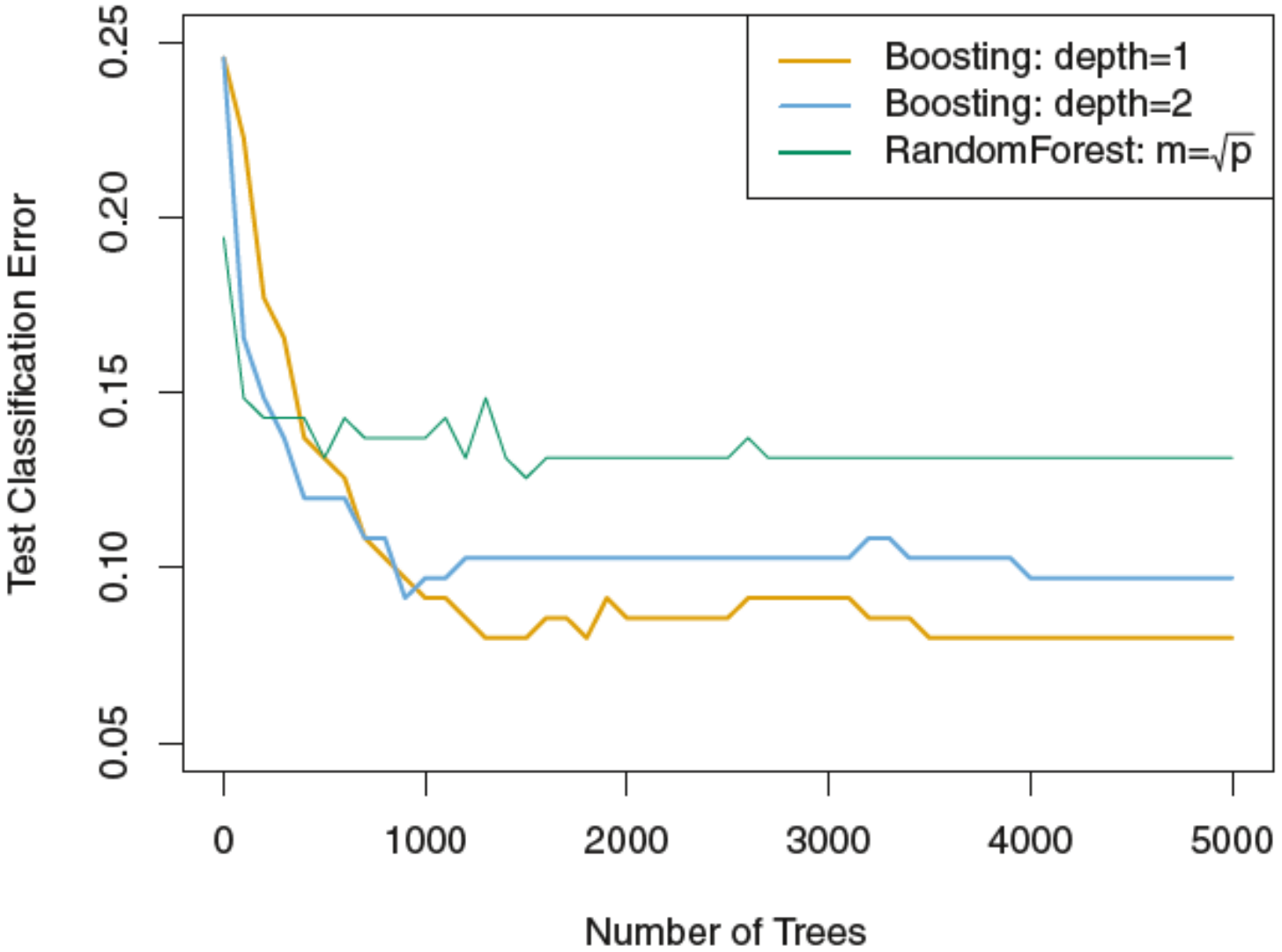
$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

c) Update the residuals:

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$$

Boosting ($\lambda = 0.01$) vs Random Forests

Test error rate for a single tree is 24%



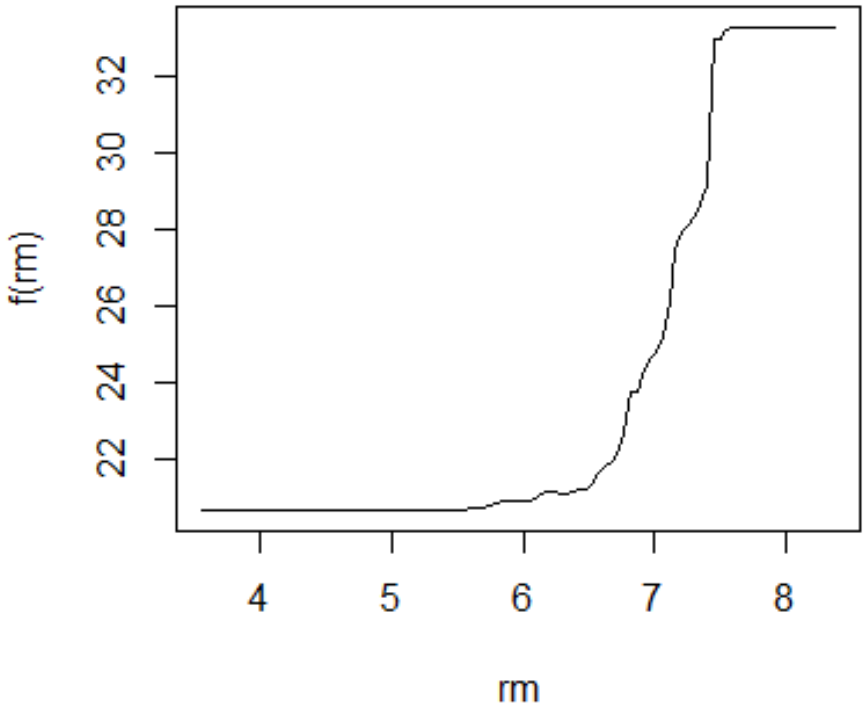
library(gbm); set.seed(1)

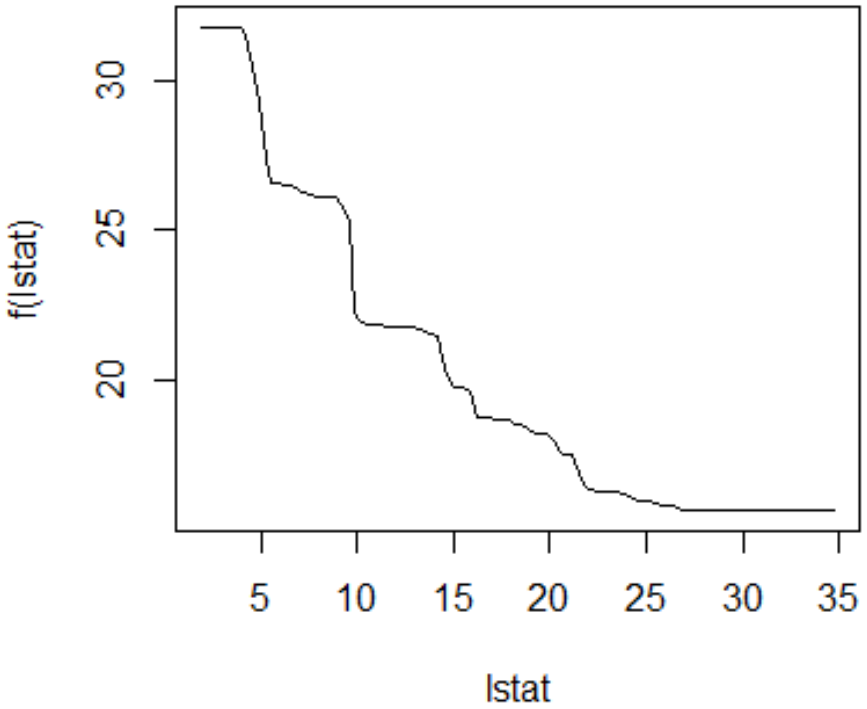
boost.boston=gbm(medv~.,data=Boston[train,], distribution="gaussian",n.trees=5000, interaction.depth=4); summary(boost.boston)

	var	rel.inf
lstat	lstat	45.9627334
rm	rm	31.2238187
dis	dis	6.8087398
crim	crim	4.0743784
nox	nox	2.5605001
ptratio	ptratio	2.2748652
black	black	1.7971159
age	age	1.6488532
tax	tax	1.3595005
indus	indus	1.2705924
chas	chas	0.8014323
rad	rad	0.2026619
zn	zn	0.0148083

par(mfrow=c(1,2))

```
plot(boost.boston,i="rm")
plot(boost.boston,i="lstat")
```





yhat.boost=predict(boost.boston, newdata=Boston[-train,],n.trees=5000)

```
mean((yhat.boost-boston.test)^2)
```

11.84

```
boost.boston=gbm(medv~.,data=Boston[train,],
distribution="gaussian",n.trees=5000,
interaction.depth=4, shrinkage=0.2,verbose=F)
yhat.boost=predict(boost.boston,
newdata=Boston[-train,],n.trees=5000)
mean((yhat.boost-boston.test)^2)
```

11.51