Random forest

If there is a strong predictor in the dataset, the decision trees produced by each of the bootstrap samples in the bagging algorithm becomes very similar: Most of the trees will use the same strong predictor in the top split.

Random forests is a solution to this problem and a method for decorrelating the trees. The hope is to improve the variance reduction.

less gain, the ensemble not independent and diverse

$$\mathrm{Corr}\left(f^{*k}(x), f^{*l}(x)\right) = \rho \quad \forall l \neq k$$

$$\mathrm{Var}\left(\frac{1}{B}\sum_{b=1}^{B} f^{*b}(x)\right) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

$$\rho \to 1 \Rightarrow \mathrm{Var} \to \sigma^2$$

Core modifications to bagging

The idea behind random forest is to *improve the variance reduction of bagging* by reducing the correlation between the trees - while hoping the possible increase in variance in each tree doesn´t cancel the improvement.

The procedure is thus as in bagging, but with the important difference, that

- at each split we are only allowed to consider $m < p$ of the predictors.

A new sample of m predictors is taken at each split and

- typically $m = \text{floor}(\sqrt{p})$ (classificaton) and $m = \text{floor}(p/3)$ (regression)

The general idea is at for very correlated predictors m is chosen to be small.

- so far 08.02.2021 -

Random forest algorithm

(We write out in class.)

- Regression: average of trees
- Classification: majority vote based on vote from each tree.

<u>Group discussion</u>: Study ELS Fig 15.3 (next page)
What do you see? (B, methods, m)

If time: same with ELS Fig 15.9 (next next page)

[Return to main session when ready.]

1990: N= 20.440 neighborhoods
Y= median house value
(ELS 10.14.1)

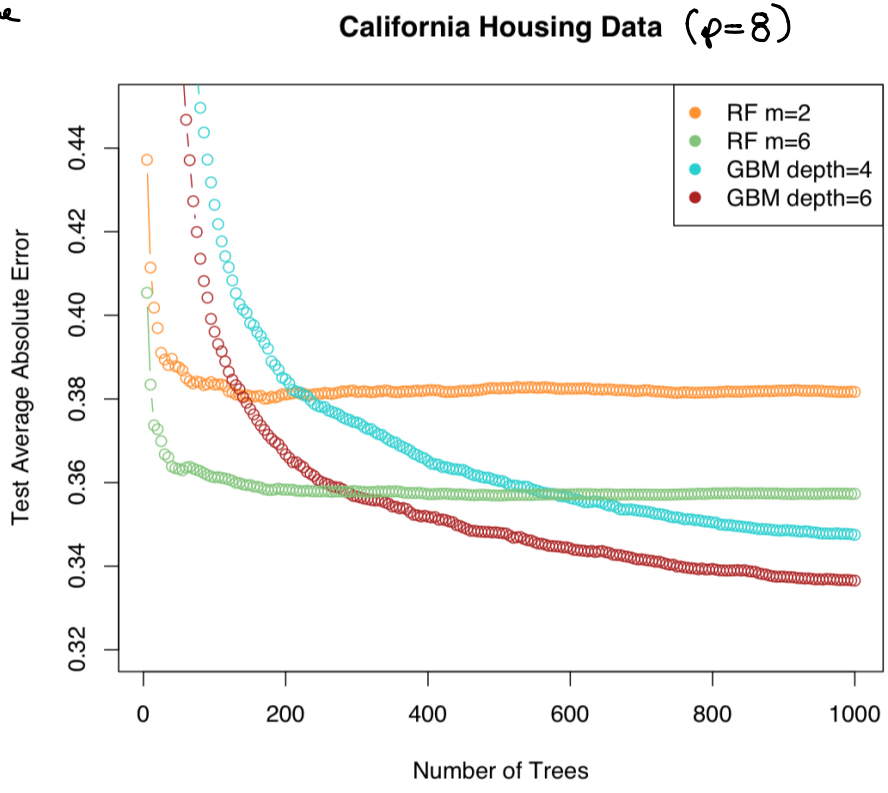


FIGURE 15.3. *Random forests compared to gradient boosting on the California housing data. The curves represent mean absolute error on the test data as a function of the number of trees in the models. Two random forests are shown, with* $m = 2$ *and* $m = 6$*. The two gradient boosted models use a shrinkage parameter* $\nu = 0.05$ *in (10.41), and have interaction depths of* 4 *and* 6*. The boosted models outperform random forests.*

Simulation model:

$$Y = \frac{1}{\sqrt{50}} \sum_{j=1}^{50} X_j + \varepsilon$$

500 training sets of N=100 and one test set with 600 obs

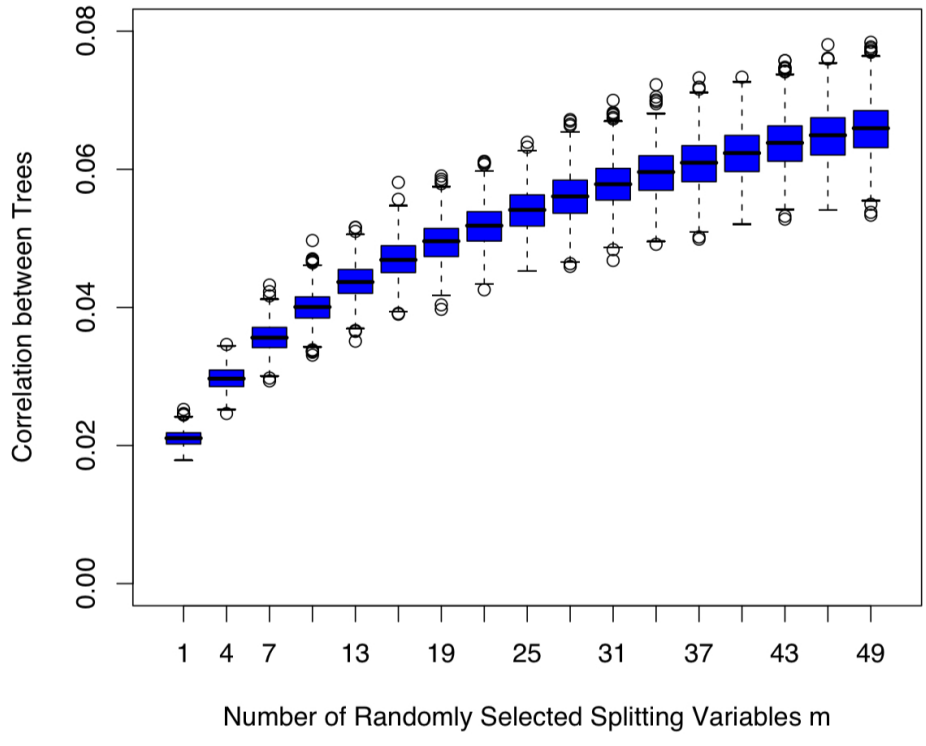


FIGURE 15.9. *Correlations between pairs of trees drawn by a random-forest regression algorithm, as a function of m. The boxplots represent the correlations at 600 randomly chosen prediction points x.*

in the test set

Node size

In addition the recommendations from the Random forest authors were also on *node size* (the minimum number of observations in a leaf node):

- classification: 1
- regression: 5

(ELS page 592)

This is an indication that node size is an hyperparameter, but ESL argue that is is maybe not worth the extra effort to optimize on this parameter.

Study ELS Figure 15.8 for effect of node size.

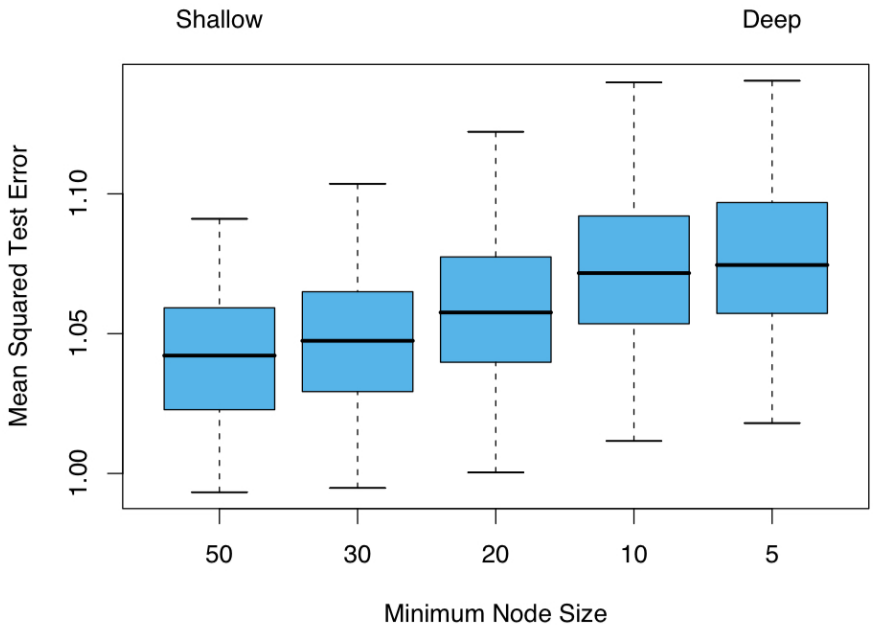
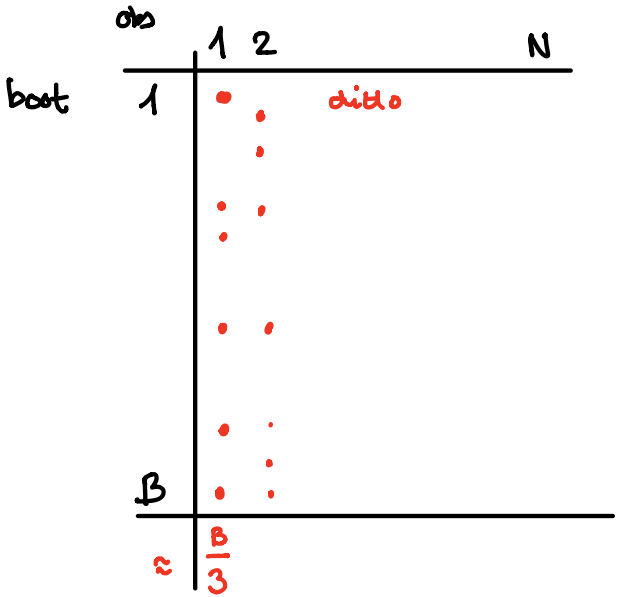


FIGURE 15.8. *The effect of* tree size *on the error in random forest regression. In this example, the true surface was additive in two of the* 12 *variables, plus additive unit-variance Gaussian noise. Tree depth is controlled here by the minimum node size; the smaller the minimum node size, the deeper the trees.*

Out-of-Bag OOB

use obs. not in bootstrap sample as "test observations"



• = not in boot.

$\hat{y}_{(i)}$ based on $B/3$ trees on average

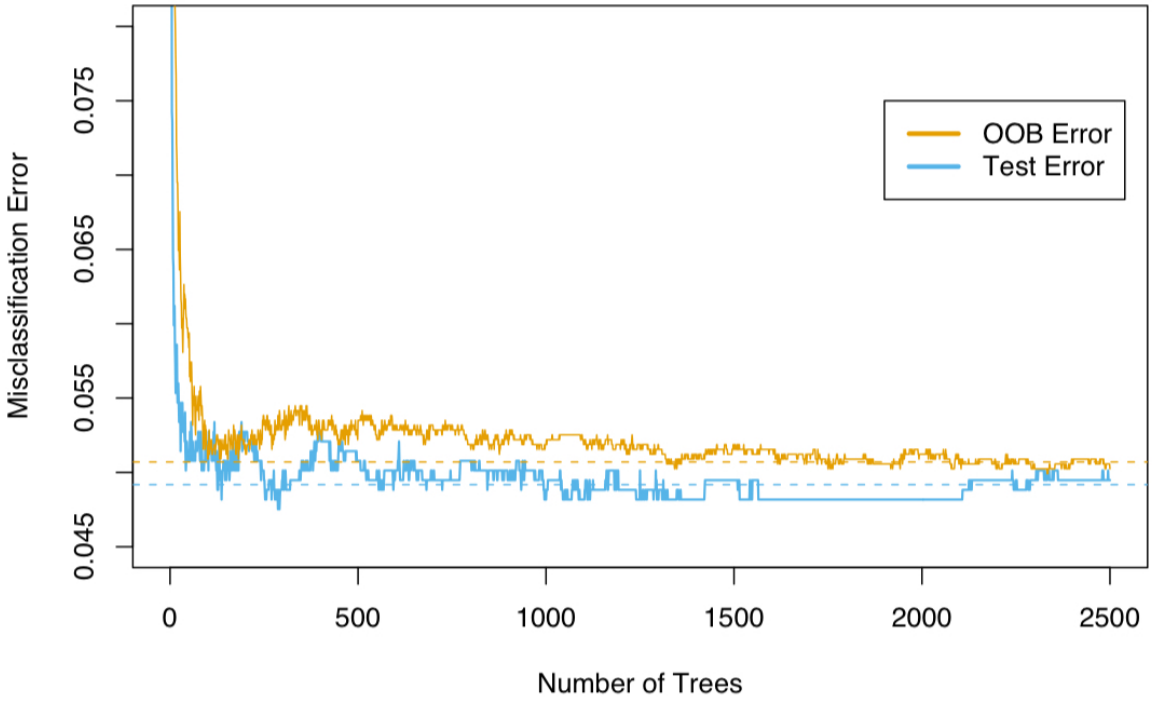
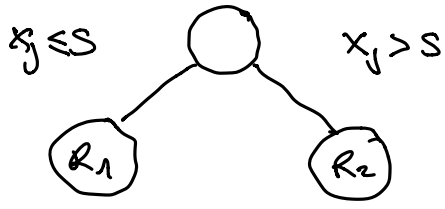


FIGURE 15.4. OOB *error computed on the* `spam` *training data, compared to the test error computed on the test set.*

Variable importance: single tree (10.13.1)

LS:

x_j
split at s



$$\sum_{i=1}^{N} (y_i - \bar{y})^2$$

$$R_1(j,s) = \{x \mid x_j \leq s\}$$

$$R_2(j,s) = \{x \mid x_j > s\}$$

$$\min_{(j,s)} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right]$$

From single to many trees

$$I_l^2 = \frac{1}{B} \sum_{b=1}^{B} I_l^2(T_b)$$

The measure is relative:

- the highest value is set to 100
- the others are scaled according to this

R: `varImpPlot` (or `importance`) in `randomForest` with `type=2`.

Variable importance based on randomization

Variable importance based on randomization is calculated using the OOB sample.

- Computations are carried out for one bootstrap sample at a time.
- Each time a tree is grown the OOB sample is used to test the predictive power of the tree.
- Then for one predictor at a time, repeat the following:
  - permute the OOB observations for the jth variable x_j and calculate the new OOB error.
  - If x_j is important, permuting its observations will decrease the predictive performance.
- The difference between the two is averaged over all trees
- and again highest set to 100, others rescaled.

R: `varImpPlot` (or `importance`) in `randomForest` with `type=1`.

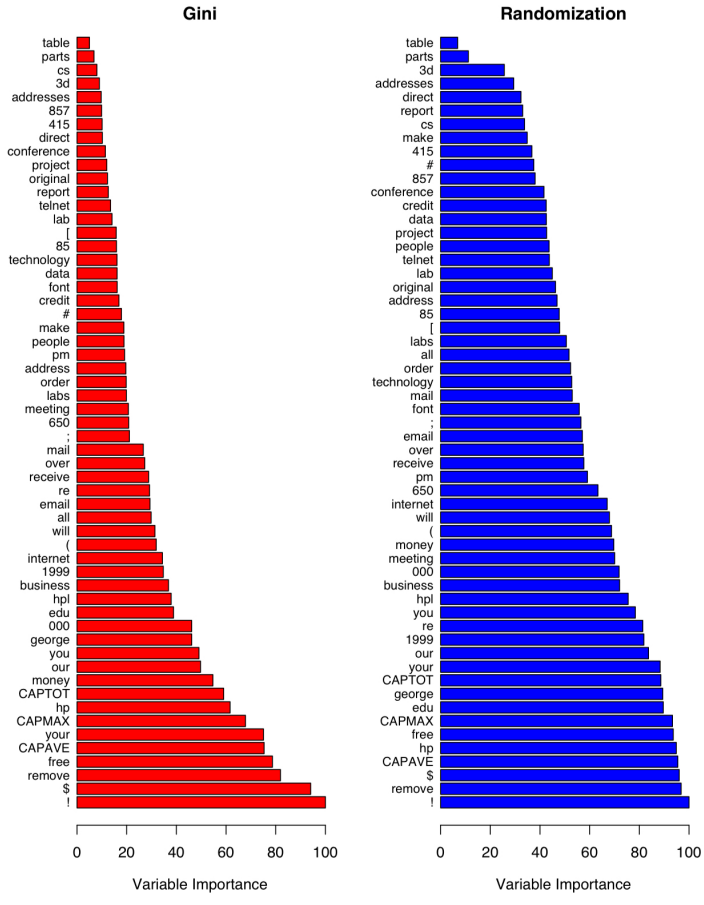


FIGURE 15.5. *Variable importance plots for a classification random forest grown on the* `spam` *data. The left plot bases the importance on the Gini splitting index, as in gradient boosting. The rankings compare well with the rankings produced by gradient boosting (Figure 10.6 on page 354). The right plot uses* OOB *randomization to compute variable importances, and tends to spread the importances more uniformly.*

Variable importance based on randomization

Variable importance based on randomization is calculated using the OOB sample.

- Computations are carried out for one bootstrap sample at a time.
- Each time a tree is grown the OOB sample is used to test the predictive power of the tree.
- Then for one predictor at a time, repeat the following:
  - permute the OOB observations for the jth variable x_j and calculate the new OOB error.
  - If x_j is important, permuting its observations will decrease the predictive performance.
- The difference between the two is averaged over all trees
- and again highest set to 100, others rescaled.

R: `varImpPlot` (or `importance`) in `randomForest` with `type=1`.

Study Figure 15.5 in ELS.

Boston

```
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)
```

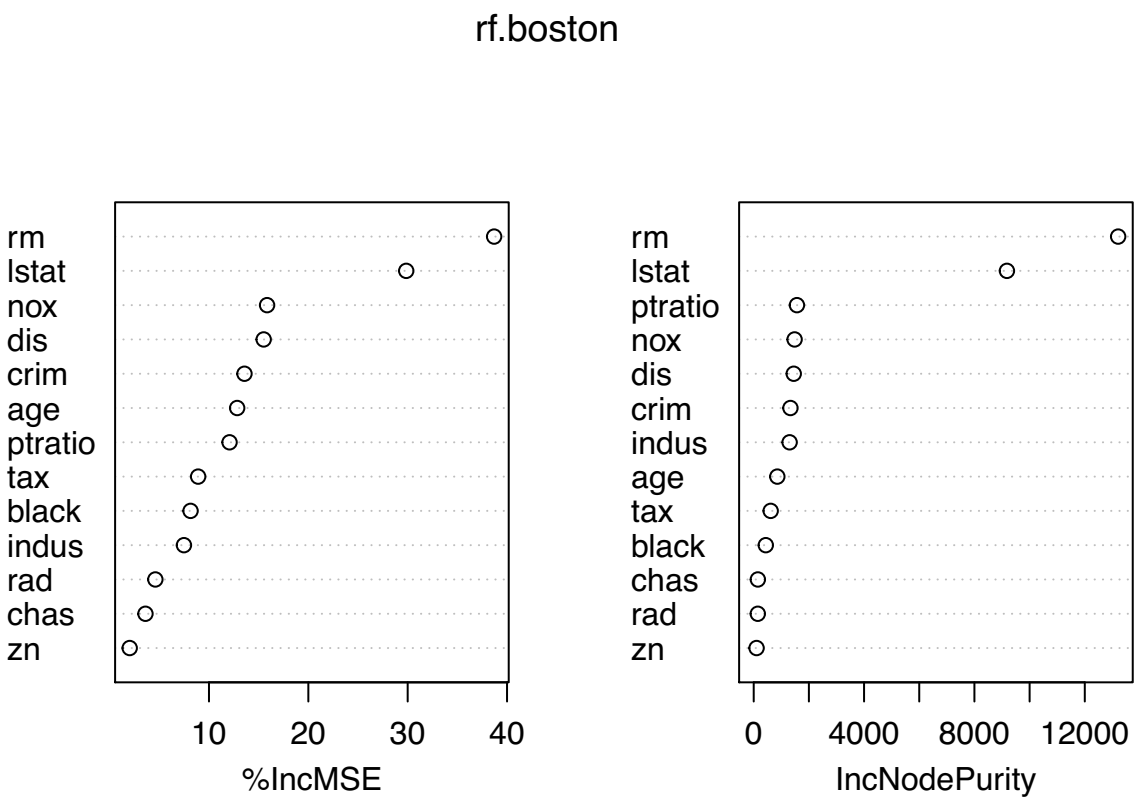
```
## [1] 15.77329
```

OOB not OOB

```
importance(rf.boston)
```

```
##           %IncMSE IncNodePurity
## crim    13.571040     1331.4214
## zn       2.018357      103.3764
## indus    7.478237     1301.3529
## chas     3.604777      150.6007
## nox     15.847850     1481.9064
## rm      38.703015    13209.6852
## age     12.837457      856.1236
## dis     15.505816     1450.6934
## rad      4.600793      147.7769
## tax      8.910426      615.1269
## ptratio 12.069248     1566.8163
## black    8.144727      438.1747
## lstat   29.854464     9177.8663
```

```
varImpPlot(rf.boston)
```



Pima indians

We decorrelate the trees by using the *randomForest()* function again, but this time we set *mtry=3*. This means that the algorithm only considers three of the predictors in each split. We choose 3 because we have 10 predictors in total and $\sqrt{10} \approx 3$.

```
set.seed(1)
rf=randomForest(factor(diabetes)~npreg+glu+bp+skin+bmi+ped+age,data=ctrain,mtry=3,importance=TRUE) #def
rf
```

```
##
## Call:
##  randomForest(formula = factor(diabetes) ~ npreg + glu + bp +      skin + bmi + ped + age, data = ct
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 22%
## Confusion matrix:
##     0  1 class.error
## 0 172 28        0.14
## 1  38 62        0.38
```

```
test.x=ctest[,-1]
test.y=ctest[,1]
train.y=ctrain[,1]
train.x=ctrain[,-1]
```

```r
train.res=predict(rf,type="prob")[,2]
test.res=predict(rf,newdata=test.x,type="prob")[,2]
train.class=ifelse(train.res>=0.5,1,0)
#train.class2=predict(rf,type="response") #same as train.class
test.class=ifelse(test.res>=0.5,1,0)
print("Evaluation on training data")
```

```
## [1] "Evaluation on training data"
```

```r
confusionMatrix(factor(train.class),factor(train.y))$overall[1]
```

```
##  Accuracy
## 0.7733333
```

```r
print("Evaluation on test data")
```

```
## [1] "Evaluation on test data"
```

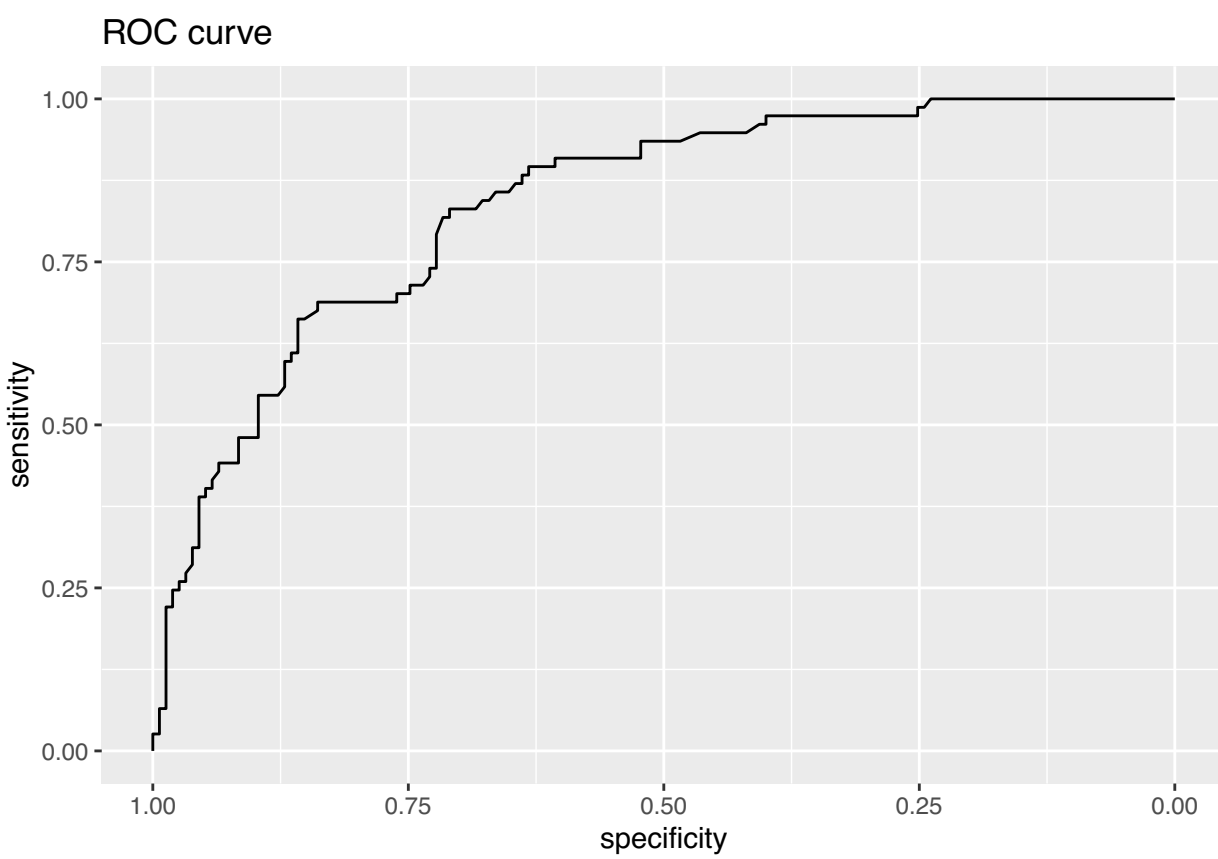
```r
confusionMatrix(factor(test.class),factor(test.y))$overall[1]
```

```
##  Accuracy
## 0.7801724
```

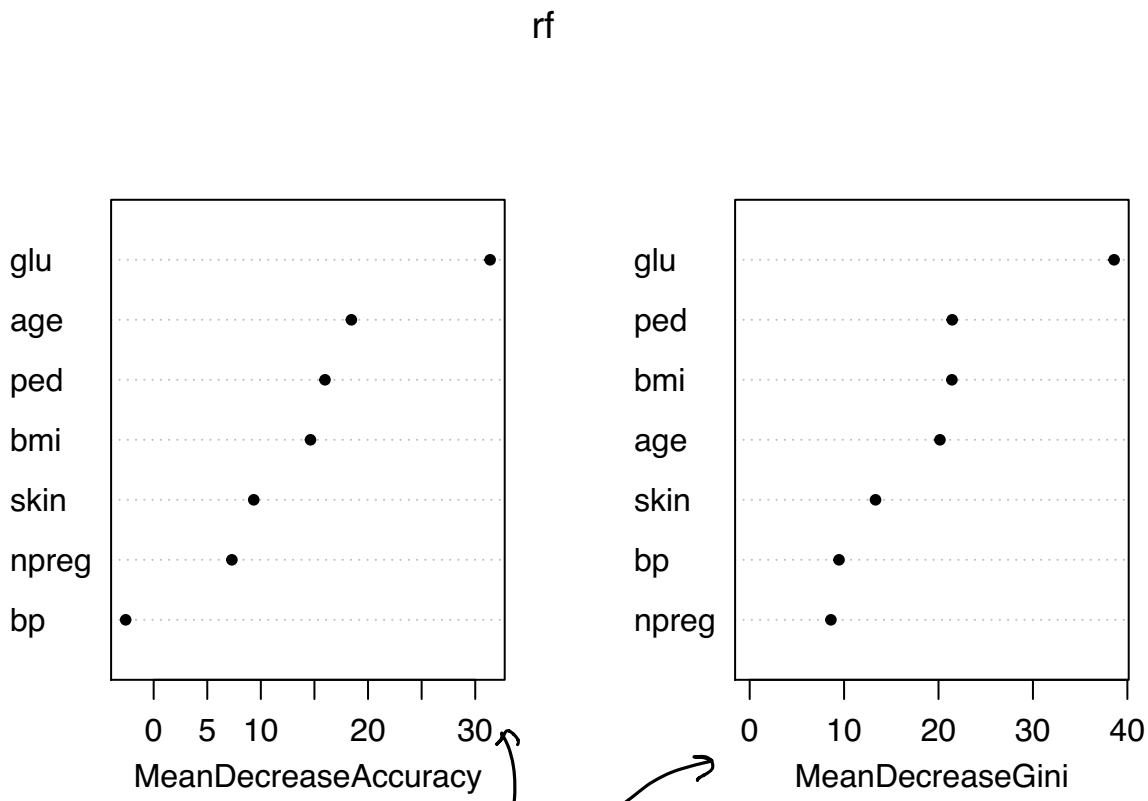
```r
roc.rf = roc(test.y,test.res,legacy.axes=TRUE)
print(auc(roc.rf))
```

```
## Area under the curve: 0.8379
```

```r
ggroc(roc.rf)+ggtitle("ROC curve")
```



```r
varImpPlot(rf,pch=20)
```




```r
#varImpPlot(rf,pch=20,type=1)
#varImpPlot(rf,pch=20,type=2)
```

= OOB

= not OOB

Forward - boosting next

Study ELS Figure 15.7 for comparing random forest with boosting as a function of relevant variables.

When the number of relevant predictors * is high, random forest performs well. * is small, random forest performance deteriorate with many noisy variables

Conclusions

Exercises

Small tasks

Look through the many problems sets presented in this document. (Solutions provided for some of the problems.)

Prove the formula

for the variance of the mean with compound symmetry correlation

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

This is also Exercise 15.1 in ELS. Link to solutions

Conclusions