Machine Learning - Exercise 2

Robin Perälä 175910, Alexander Norrgran 164620, Robert Tommola 194000

12.4.2022

1.

ISLR, Exercise 5.4.2, p. 219-220. *We will now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap sample from a set of n observations.*

(a) *What is the probability that the first bootstrap observation is not the jth observation from the original sample? Justify your answer.*

If we have n observations and draw a random draw from them, the probability to obtain the jth observation (a specific one) is 1/n. The complement of that is the probability to not obtain the jth observation. The complement is defined as $1 - 1/n = (n - 1)/n$.

Give formulas and use correct wording (permutations)...

(b) *What is the probability that the second bootstrap observation is not the jth observation from the original sample?*

Bootstap samples are drawn with replacement, so the probabilities of each draw is the same. The probability is (n-1)/n

(c) *Argue that the probability that the jth observation is not in the bootstrap sample is (1 - 1/n)^n.*

If the probability that the jth observation is (n-1)/n for one specific draw, then we compute the probability that is not in any of the draws by multiplying. $(n - 1)/n * (n - 1)/n * ... * (n - 1)/n$ (not in the first draw, and not in the second draw, ... , an not in the last draw). Bootstrap sampling uses the same sample size as the original sample (n). This means that we have ((n-1)/n)^n. (n-1)/n can also be written as 1 - 1/n, which means that we have (1-1/n)^n.

(d) *When n = 5, what is the probability that the jth observation is in the bootstrap sample?*

We use the complement $1 - (1 - 1/n)^n = 1 - (1 - 1/5)^5 = 1 - (4/5)^5 = 0.67232$

(e) *When n = 100, what is the probability that the jth observation is in the bootstrap sample?*

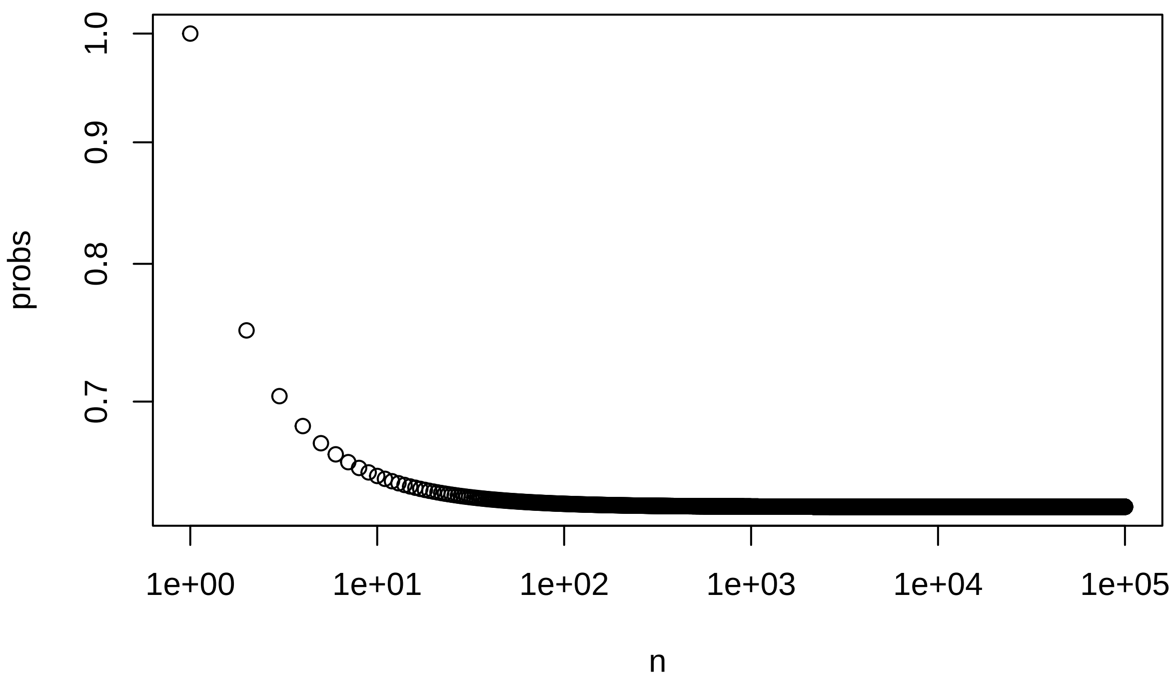
$1 - (99/100)^100 = 0.6339677$

(f) *When n = 10, 000, what is the probability that the jth observation is in the bootstrap sample?*

$1-(9999/10000)^{10000}=0.632139$ When n grows bigger, the probability is getting ever closer to 1 - 1/e

(g) *Create a plot that displays, for each integer value of n from 1 to 100, 000, the probability that the jth observation is in the bootstrap sample. Comment on what you observe.*

```
n = 1:100000
probs = 1 - ((n-1)/n)^{n}
plot(n, probs, log='xy')
```



(h) *We will now investigate numerically the probability that a bootstrap sample of size n = 100 contains the jth observation. Here j = 4. We repeatedly create bootstrap samples, and each time we record whether or not the fourth observation is contained in the bootstrap sample. Comment on the results obtained.*

```
set.seed(1)
store=rep(NA, 10000)
for(i in 1:10000){
  store[i]=sum(sample(1:100, rep=TRUE)==4)>0
}
mean(store) #0.6405
```

```
## [1] 0.6417
```

```r
#Which is getting ever closer to 1 - 1/e
1-1/exp(1) #0.63212...
```

```
## [1] 0.6321206
```

2.

Suppose that n = 10 and the observations are 6.45, 1.28, -3.48, 2.44, -5.17, -1.67, -2.03, 3.58, 0.74, -2.14 *Write a script in R to simulate the fraction of the original observations not contained in a bootstrap sample. Use B = 10000 bootstrap replications. Compare with the approximation 10/3.*

```
set.seed(1)
n=10
obs = c(6.45, 1.28, -3.48,
        2.44, -5.17, -1.67,
        -2.03, 3.58, 0.74, -2.14)

store=rep(NA, 10000)
for(i in 1:10000){
  store[i]= (n - sum(obs %in% sample(obs, size=10, rep=TRUE)))/n
}
mean(store)
```

```
## [1] 0.35057
```

```
#If we compare the mean value of observations included in the bootstrap (instead of the fraction) it is
mean(store)*n
```

```
## [1] 3.5057
```

```
#Which is close to
10/3
```

```
## [1] 3.333333
```

3

ISLR, Exercise 8.4.2, p. 361 *It is mentioned in Section 8.2.3 that boosting using depth-one trees (or stumps) leads to an additive model: that is, a model of the form*

$$f(X) = \sum_{j=1}^{p} f_j(X_j).$$

Explain why this is the case. You can begin with (8.12) in Algorithm 8.2.

Equation 8.12:

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

Because the tree contains just one split (stump), and as such the shrinkage parameter is 1? / 0?

Because the trees are only stumps, when we update $\hat{f}^b$ by adding in a shrunken version of the new tree

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

, it only results in...

4.

Use the data USCompaniesdata.dta. Create a training set containing half of the observations, and a test set containing the remaining observations. Fit a tree with Return on Assets (roa_w) as the response and the other variables as predictors.

```
library(haven) #For importing Stata data
library(tree) #For fitting trees
library(randomForest) #For bagging and randomforests
library(gbm) #For boosting

#Load Stata data with read_dta function from haven
USCompanies_data_winsorized = read_dta("USCompanies_data_winsorized.dta")
USData = subset(USCompanies_data_winsorized, select = -conm)
USData = na.omit(USData)

#Split into training and test set
set.seed(1)
train <- sample(1:nrow(USData), nrow(USData)/2)
test <- (-train)

# set.seed(1)
# train=sample(c(TRUE, FALSE), nrow(USData), rep=TRUE)
# test = !train
```

```
#Fit a tree using function "tree" (in library "tree")
treeROA = tree(roa_w~. , USData, subset=train)
summary(treeROA)
```

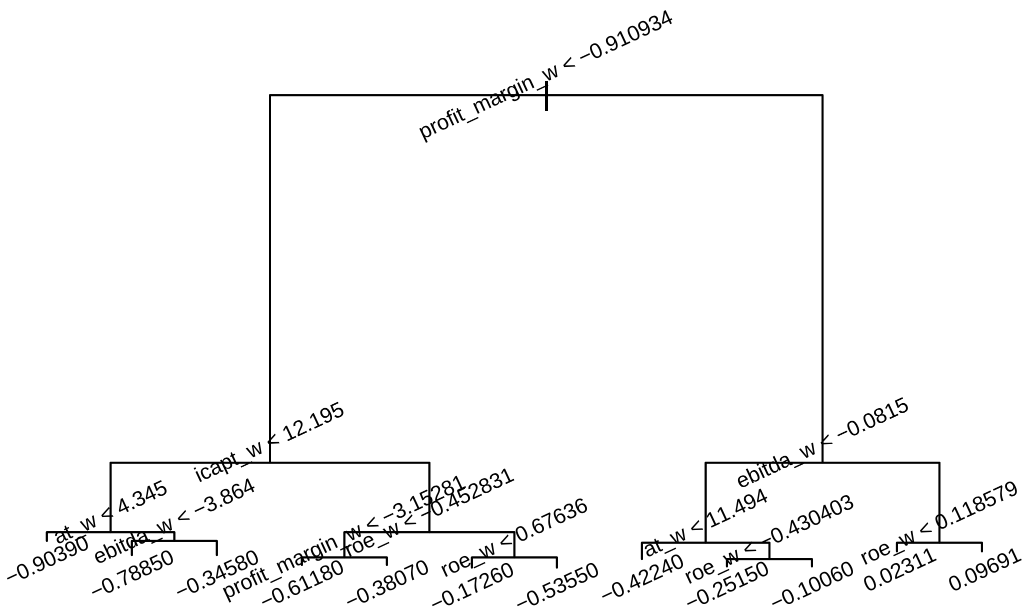
```
##
## Regression tree:
## tree(formula = roa_w ~ ., data = USData, subset = train)
## Variables actually used in tree construction:
## [1] "profit_margin_w" "icapt_w"         "at_w"            "ebitda_w"
## [5] "roe_w"
## Number of terminal nodes:  12
## Residual mean deviance:  0.01093 = 18.31 / 1676
## Distribution of residuals:
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -0.6040000 -0.0382000  0.0001005  0.0000000  0.0439200  0.8494000
```

```
treeROA
```

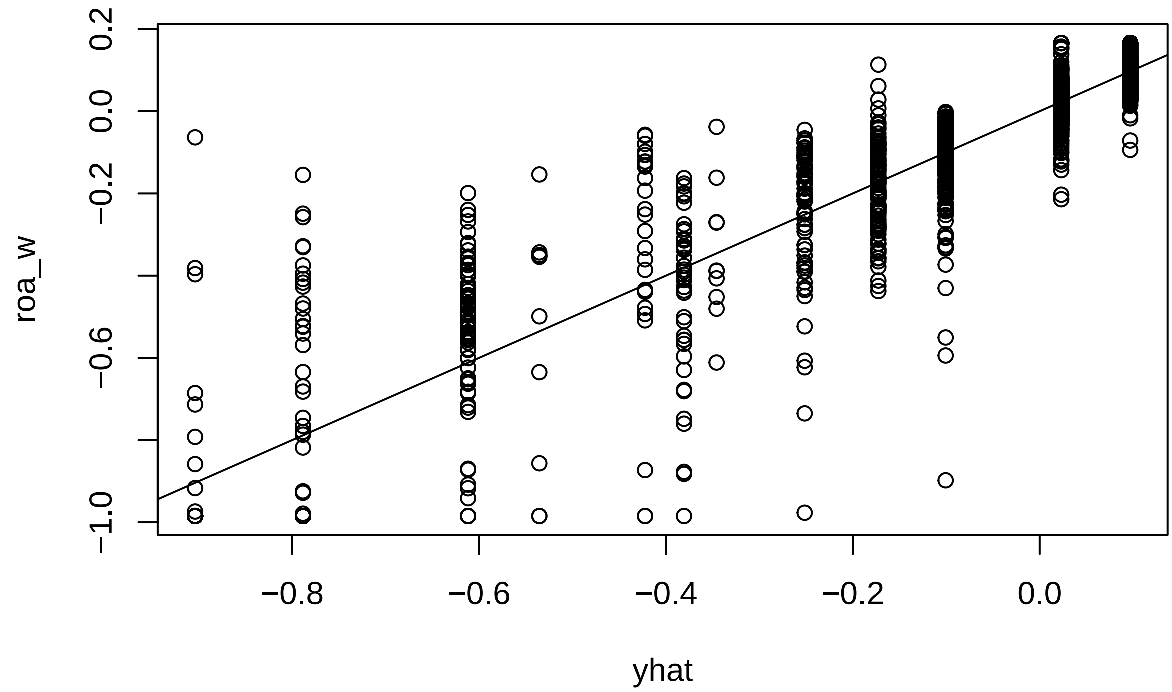
```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 1688 121.2000 -0.085170
##    2) profit_margin_w < -0.910934 311  33.0700 -0.486700
##      4) icapt_w < 12.195 102   8.1060 -0.763400
##        8) at_w < 4.345 43   2.1370 -0.903900 *
##        9) at_w > 4.345 59   4.5000 -0.660900
##         18) ebitda_w < -3.864 42   1.6620 -0.788500 *
```

```
##         19) ebitda_w > -3.864 17   0.4666 -0.345800 *
##      5) icapt_w > 12.195 209  13.3500 -0.351700
##       10) roe_w < -0.452831 96   5.3960 -0.505900
##         20) profit_margin_w < -3.15281 52   2.4670 -0.611800 *
##         21) profit_margin_w > -3.15281 44   1.6560 -0.380700 *
##       11) roe_w > -0.452831 113   3.7360 -0.220800
##         22) roe_w < 0.67636 98   0.9485 -0.172600 *
##         23) roe_w > 0.67636 15   1.0750 -0.535500 *
##    3) profit_margin_w > -0.910934 1377  26.6800  0.005529
##      6) ebitda_w < -0.0815 319   9.5560 -0.174000
##       12) at_w < 11.494 39   2.9000 -0.422400 *
##       13) at_w > 11.494 280   3.9150 -0.139400
##         26) roe_w < -0.430403 72   1.5760 -0.251500 *
##         27) roe_w > -0.430403 208   1.1220 -0.100600 *
##      7) ebitda_w > -0.0815 1058   3.7430  0.059660
##       14) roe_w < 0.118579 534   1.1950  0.023110 *
##       15) roe_w > 0.118579 524   1.1070  0.096910 *
```

```
#Plot
plot(treeROA); text(treeROA, pretty=0, cex=0.7, srt=25)
```




```
#MSE
yhat=predict(treeROA, newdata=USData[-train,])
ROAtest=USData[-train, "roa_w"]
plot(yhat, ROAtest$roa_w, ylab = "roa_w")
abline(0,1)
```




```
mean((yhat-ROAtest$roa_w)^2)
```

```
## [1] 0.01246236
```

5.

Apply bagging to USCompaniesdata.dta. Compare the MSE of the tree in Exercise 4 with the MSE of the bagged trees.

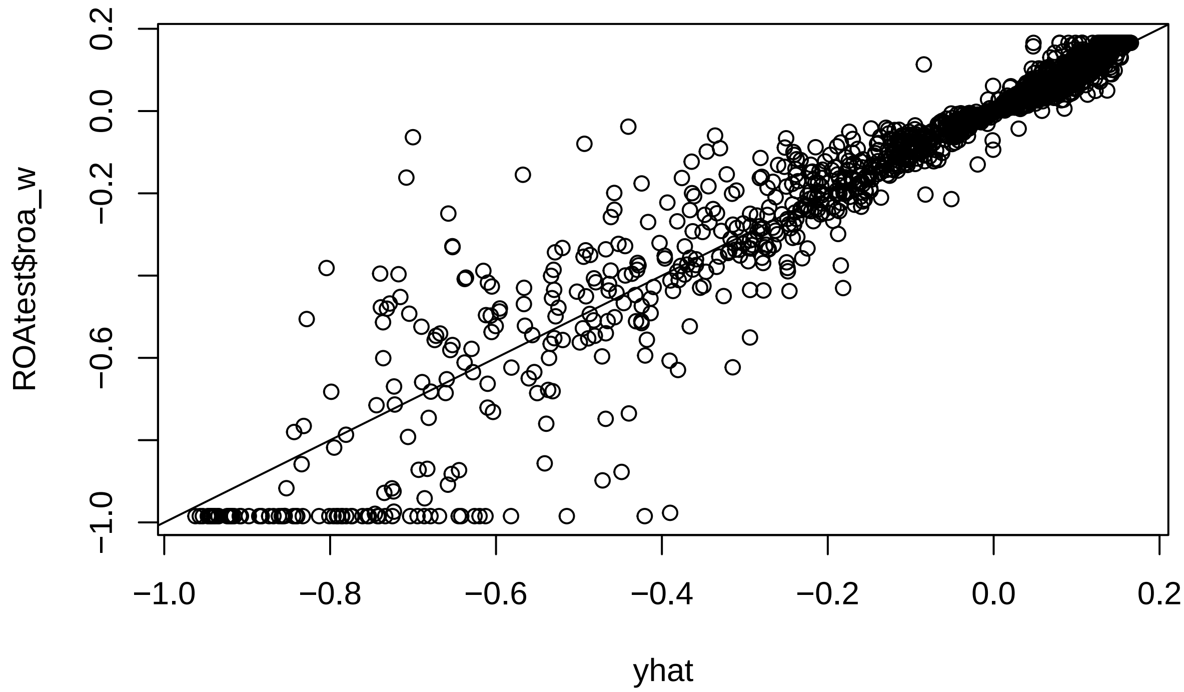
```
#Bagging

#Bagging using the function randomForest,
#inside the randomForest library. When mtry is for all variables, it is bagging.
bagROA = randomForest(roa_w~., USData, subset=train, mtry=ncol(USData)-1, importance =TRUE)
summary(bagROA)
```

```
##                 Length Class  Mode
## call               6   -none- call
## type               1   -none- character
## predicted       1688   -none- numeric
## mse              500   -none- numeric
## rsq              500   -none- numeric
## oob.times       1688   -none- numeric
## importance        88   -none- numeric
## importanceSD      44   -none- numeric
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            11   -none- list
## coefs              0   -none- NULL
## y               1688   -none- numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

```
bagROA
```

```
##
## Call:
##  randomForest(formula = roa_w ~ ., data = USData, mtry = ncol(USData) -      1, importance = TRUE, s
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 44
##
##           Mean of squared residuals: 0.007524909
##                     % Var explained: 89.52
```

```
#MSE
yhat=predict(bagROA, newdata=USData[-train,])
ROAtest=USData[-train, "roa_w"]
plot(yhat, ROAtest$roa_w)
abline(0,1)
```




```
mean((yhat-ROAtest$roa_w)^2)
```

```
## [1] 0.006178823
```

6.

Apply random forests to USCompaniesdata.dta. Does random forests provide an improvement over the bagged trees in Exercise 5?

```
#Random forests

set.seed(1)
#Fit a random forest using the function randomForest,
#inside the randomForest library. When mtry is less than
#than all variables in the data, it is a random forests model
forestROA = randomForest(roa_w~., USData, subset=train, importance = TRUE)
# forestROA = randomForest(roa_w~., USData, subset=train, mtry=ncol(USData)/2, importance = TRUE)
# forestROA = randomForest(roa_w~., USData, subset=train, mtry=ncol(USData)/3, importance = TRUE)
#Should be done on several different mtry

summary(forestROA)
```

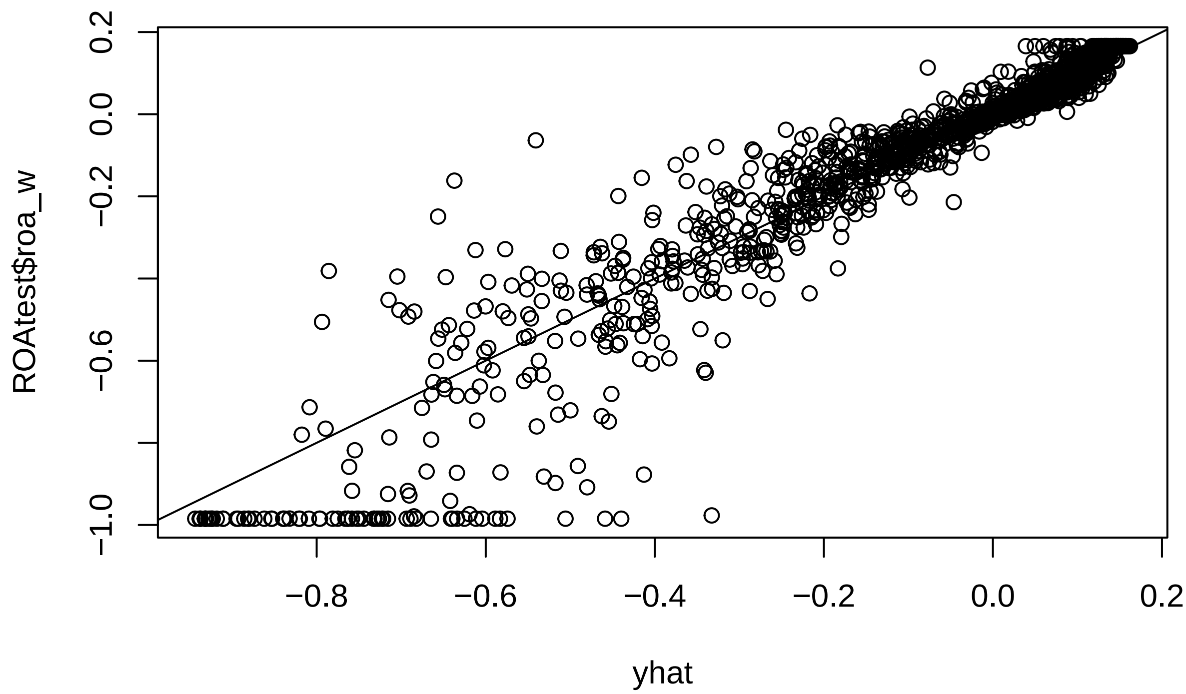
```
##                 Length Class  Mode
## call               5   -none- call
## type               1   -none- character
## predicted       1688   -none- numeric
## mse              500   -none- numeric
## rsq              500   -none- numeric
## oob.times       1688   -none- numeric
## importance        88   -none- numeric
## importanceSD      44   -none- numeric
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            11   -none- list
## coefs              0   -none- NULL
## y               1688   -none- numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

```
forestROA
```

```
##
## Call:
##  randomForest(formula = roa_w ~ ., data = USData, importance = TRUE,      subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 14
##
##           Mean of squared residuals: 0.007853636
##                     % Var explained: 89.06
```

```
#MSE
yhat=predict(forestROA, newdata=USData[-train,])
```

```r
ROAtest=USData[-train, "roa_w"]
plot(yhat, ROAtest$roa_w)
abline(0,1)
```



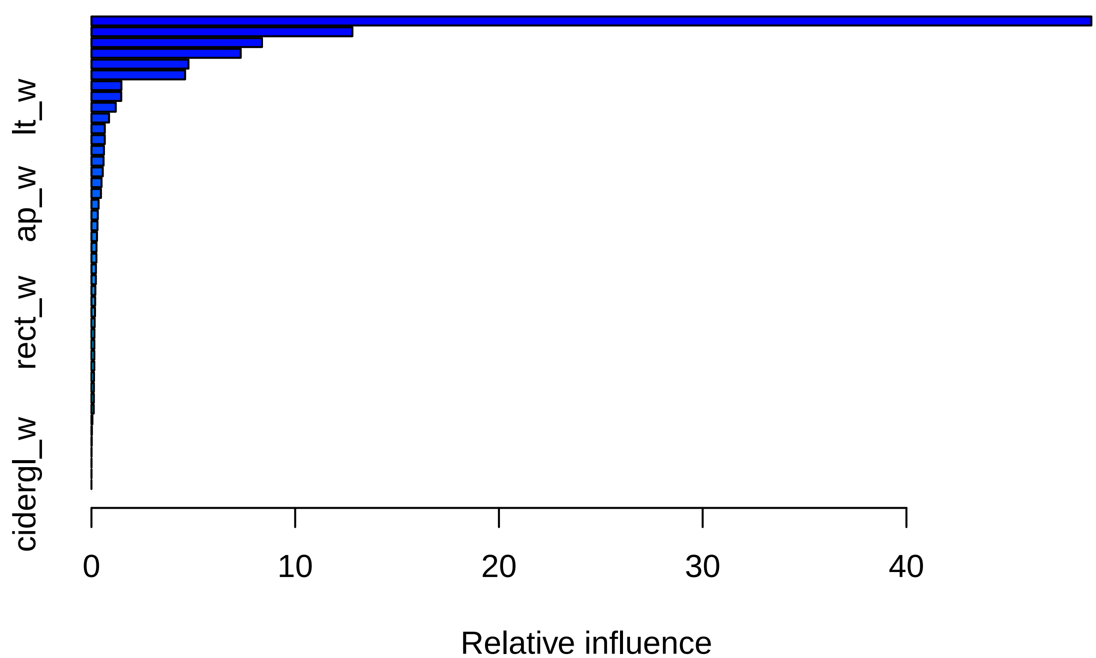

```r
mean((yhat-ROAtest$roa_w)^2)
```

```
## [1] 0.006525587
```

7.

Apply boosting to USCompaniesdata.dta. Which variables are the most important predictors in the boosted model?

```
#Boosting
set.seed(1)
#Boosting using the "gbm" function inside the "gbm" library
boostROA = gbm(roa_w~., USData, distribution="gaussian", n.trees=500, interaction.depth=4) #5000


summary(boostROA)
```



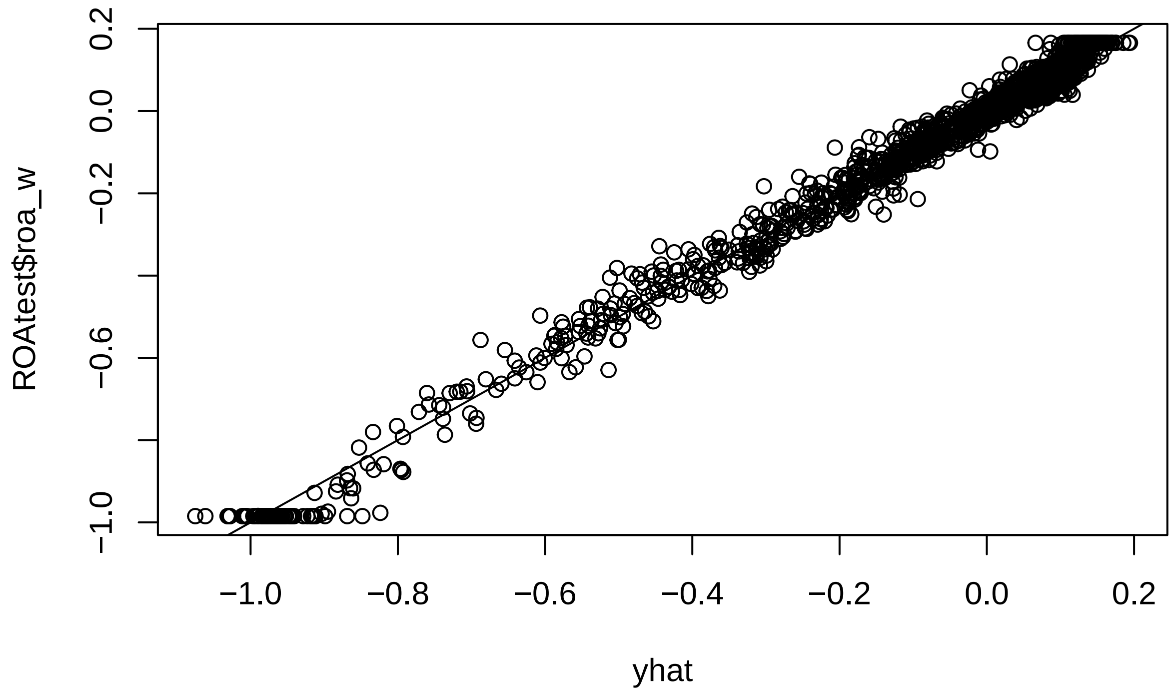

```
##                                    var      rel.inf
## profit_margin_w       profit_margin_w 49.077220066
## ebitda_w                     ebitda_w 12.808796933
## roe_w                           roe_w  8.370929651
## icapt_w                       icapt_w  7.328434141
## at_w                             at_w  4.762816673
## teq_w                           teq_w  4.598167229
## ib_w                             ib_w  1.470522692
## asset_turnover_w     asset_turnover_w  1.463603848
## lt_w                             lt_w  1.194587644
## pe_w                             pe_w  0.863484862
## ch_w                             ch_w  0.656509777
```

```
## sale_w                           sale_w  0.655561154
## oancf_w                         oancf_w  0.613927694
## cogs_w                           cogs_w  0.595359456
## operating_margin_w operating_margin_w  0.555910076
## bkvlps_w                       bkvlps_w  0.495882428
## gdwl_w                           gdwl_w  0.467283587
## ap_w                               ap_w  0.352596398
## dp_w                               dp_w  0.312692577
## dlc_w                             dlc_w  0.296799663
## fopo_w                           fopo_w  0.272657843
## lco_w                             lco_w  0.247277590
## txt_w                             txt_w  0.243039866
## ppent_w                         ppent_w  0.219257314
## ceq_w                             ceq_w  0.217857303
## aco_w                             aco_w  0.190576224
## chech_w                         chech_w  0.182728402
## ci_w                               ci_w  0.174844669
## rect_w                           rect_w  0.158690123
## intano_w                       intano_w  0.147219421
## caps_w                           caps_w  0.142781679
## ivncf_w                         ivncf_w  0.141636410
## capx_w                           capx_w  0.139970966
## re_w                               re_w  0.125825184
## np_w                               np_w  0.120447364
## epspi_w                         epspi_w  0.119101764
## invt_w                           invt_w  0.113414751
## fiao_w                           fiao_w  0.046955740
## tstk_w                           tstk_w  0.022022188
## dvt_w                             dvt_w  0.016238275
## ivst_w                           ivst_w  0.006382126
## aqc_w                             aqc_w  0.005583599
## siv_w                             siv_w  0.004404648
## cidergl_w                     cidergl_w  0.000000000
```

```r
boostROA
```

```
## gbm(formula = roa_w ~ ., distribution = "gaussian", data = USData,
##     n.trees = 500, interaction.depth = 4)
## A gradient boosted model with gaussian loss function.
## 500 iterations were performed.
## There were 44 predictors of which 43 had non-zero influence.
```

```r
#all or only training set?


#MSE
yhat=predict(boostROA, newdata=USData[-train,])
ROAtest=USData[-train, "roa_w"]
plot(yhat, ROAtest$roa_w)
abline(0,1)
```




```
mean((yhat-ROAtest$roa_w)^2)
```

```
## [1] 0.0008299824
```