Machine Learning Kaggle Competition

Ingrid Wijaya

2022-08-21

Data set

```r
# reading the file and removing Id column in train and test data set
readtrain <- read.csv("training.csv")
train <- readtrain[,-1]

readtest <- read.csv("test.csv")
test <- readtest[,-1]
Id <- readtest[,1]

# Splitting the train data set into another training and testing data set to find validation rmse
set.seed(10)
select <- sample(1:nrow(train), 0.7*nrow(train))
data.train <- train[select,]
data.test <- train[-select,]
```

(1) linear regression

```r
library(ModelMetrics)
```

```
##
## Attaching package: 'ModelMetrics'

## The following object is masked from 'package:base':
##
##     kappa
```

```r
# validation rmse
lm <- lm(Y~., data.train)
lm.validation.error <- rmse(data.test$Y, predict(lm, data.test))
lm.validation.error ##1.459112
```

```
## [1] 1.459112
```

```r
# test rmse in kaggle - 1.31123
final.linear <- lm(Y~., train)
pred <- predict(final.linear, test)
lm.test.rmse <- 1.31123
```

(2) Bagging

```r
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.
```

```r
set.seed(10)

# validation rmse
bag.mod <- randomForest(Y~., data=data.train, mtry=15, importance=TRUE)
bag.pred <- predict(bag.mod, data.test)
bag.validation.error <- rmse(data.test$Y, bag.pred)
bag.validation.error ##1.4107
```

```
## [1] 1.4107
```

```r
importance(bag.mod)
```

```
##         %IncMSE IncNodePurity
## X1    2.0640773      67.43981
## X2   13.4218009      94.37083
## X3    6.4874570      87.50805
## X4   22.6027760     129.31325
## X5    2.1663854     100.67869
## X6   -0.8874408      83.64192
## X7   -1.9408445      46.92250
## X8    3.8673854     119.71319
## X9    0.9232775      61.12639
## X10   3.0281752      18.68554
## X11  -0.1518908     108.74616
## X12  20.1521251      74.78531
## X13  11.8343463      99.20869
## X14  14.2035638     114.68889
## X15   7.1876203      69.90163
```

```r
# test rmse in kaggle - 1.26061
library(randomForest)

final.bag <- randomForest(Y~., data=train, mtry=15, importance=TRUE)
pred <- predict(final.bag, test)
bag.test.rmse <- 1.26061

importance(final.bag)
```

```
##        %IncMSE IncNodePurity
## X1    6.392804     109.17133
## X2   19.685512     151.73497
## X3    4.778760     116.29111
## X4   22.653241     169.95847
## X5    2.248705     140.79751
## X6    1.225768     106.77024
## X7   -0.581542      62.85910
## X8    2.863353     162.78128
## X9    2.359059      77.36299
## X10   4.083074      27.52230
## X11   2.048881     173.72241
## X12  22.685994     121.36069
## X13  18.897302     158.10032
## X14  22.633085     196.44564
```

```
## X15 12.896867     131.31153
```

(3) Random Forest

```r
# validation rmse for m = 3
set.seed(10)
rf1 = randomForest(Y~., data.train, mtry=3, importance=T, ntree=500)
rf1.validation.error <- rmse(data.test$Y, predict(rf1, data.test))
rf1.validation.error ## 1.398608
```

```
## [1] 1.398608
```

```r
# validation rmse for m = 4
set.seed(10)
rf2 = randomForest(Y~., data.train, mtry=4, importance=T, ntree=500)
rf2.validation.error <- rmse(data.test$Y, predict(rf2, data.test))
rf2.validation.error ## 1.395356
```

```
## [1] 1.395356
```

```r
# validation rmse for m = 15/3 (recommended)
set.seed(10)
rf3 = randomForest(Y~., data.train, mtry=15/3, importance=T, ntree=500)
rf3.validation.error <- rmse(data.test$Y, predict(rf3, data.test))
rf3.validation.error ## 1.399285
```

```
## [1] 1.399285
```

```r
# final rf model as it has the lowest rmse among model above
set.seed(10)
rf = randomForest(Y~., data.train, mtry=4, importance=T, ntree=500)
rf.validation.error <- rmse(data.test$Y, predict(rf, data.test))
rf.validation.error ## 1.395569
```

```
## [1] 1.395356
```

```r
# test rmse in kaggle - 1.25125
final.rf <- randomForest(Y~., train, mtry=4, importance=T, ntree=500)
pred <- predict(final.rf, test, mtry=4, n.trees = 500)
rf.test.rmse <- 1.25125
```

(4) Boosting

```r
# for loop to find the best
library(gbm)
lambda <- seq(0.0001,0.5,0.01)
numtree <- c(500, 1000, 1500, 2000, 2500, 3000)
depth <- c(1,2,3,4)
y.train <- train$Y
validation.error <- matrix(0, nrow = length(numtree), ncol = length(lambda))
rownames(validation.error) <- c("500","1000", "1500", "2000", "2500", "3000")
validation.error.depth <- list()

for(inter.depth in depth){
  for (ntree in numtree){
    for (i in lambda){
      set.seed(10)
```

```
      boost <- gbm(Y~., data=data.train, distribution="gaussian",
                   n.trees=ntree, interaction.depth=inter.depth, shrinkage=i)

      validation.error[as.character(ntree), which(i==lambda)] <- rmse(data.test$Y, predict(boost,
                                                                                   data.test))
    }
  }
  # Each Component 1, 2, 3, 4 in validation.error.depth represents the
  # validation error rate for all ntree-shrinkage pair for the respective
  # interaction.depth value.
  validation.error.depth[[inter.depth]] <- validation.error
}

# the ntree-shrinkage pair that gives the with the lowest validation error rate
lowest.validation.1 <- which(validation.error.depth[[1]] == min(validation.error.depth[[1]]), arr.ind =
lowest.validation.1
```

```
##     row col
## 500   1  13
```

```
lowest.validation.2 <- which(validation.error.depth[[2]] == min(validation.error.depth[[2]]), arr.ind =
lowest.validation.2
```

```
##     row col
## 500   1  20
```

```
lowest.validation.3 <- which(validation.error.depth[[3]] == min(validation.error.depth[[3]]), arr.ind =
lowest.validation.3
```

```
##     row col
## 500   1   5
```

```
lowest.validation.4 <- which(validation.error.depth[[4]] == min(validation.error.depth[[4]]), arr.ind =
lowest.validation.4
```

```
##     row col
## 500   1   6
```

```
# lowest validation rmse from each interaction.depth value (1, 2, 3, 4)
val.rmse.1 <- validation.error.depth[[1]][lowest.validation.1]
val.rmse.2 <- validation.error.depth[[2]][lowest.validation.2]
val.rmse.3 <- validation.error.depth[[3]][lowest.validation.3]
val.rmse.4 <- validation.error.depth[[4]][lowest.validation.4]

# summary of lowest validation rmse from all three tuning parameters
summary <- rbind(c(1, val.rmse.1,  numtree[lowest.validation.1[1]], lambda[lowest.validation.1[2]]),
                 c(2, val.rmse.2,  numtree[lowest.validation.2[1]], lambda[lowest.validation.2[2]]),
                 c(3, val.rmse.3,  numtree[lowest.validation.3[1]], lambda[lowest.validation.3[2]]),
                 c(4, val.rmse.4,  numtree[lowest.validation.4[1]], lambda[lowest.validation.4[2]]))
rownames(summary) <- c("1","2", "3", "4")
colnames(summary) <- c("interaction.depth","validation rmse", "ntree", "shrinkage")
summary
```

```
##   interaction.depth validation rmse ntree shrinkage
## 1                 1        1.403593   500    0.1201
## 2                 2        1.375583   500    0.1901
## 3                 3        1.369416   500    0.0401
```

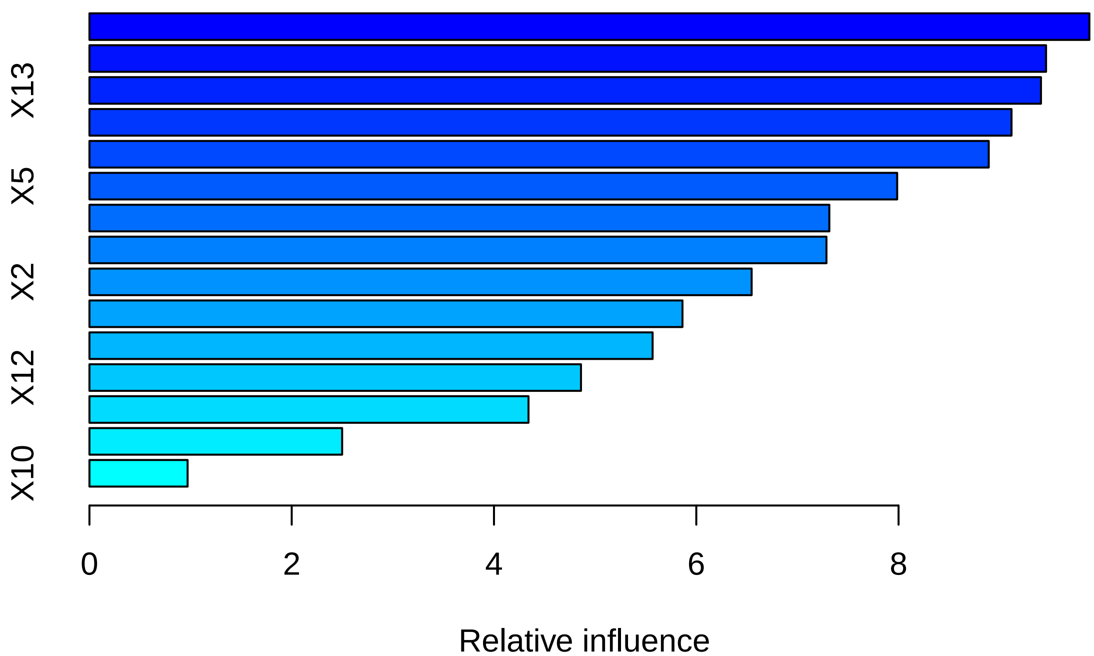
```
## 4                  4        1.364077    500     0.0501
```

```r
bestntree <- 500
bestlambda <- 0.0501
bestinteraction.depth <- 4

# validation rmse
set.seed(10)
boost.mod <- gbm(Y~., data=data.train, distribution="gaussian",
                 n.trees=bestntree, interaction.depth=bestinteraction.depth, shrinkage=bestlambda)
boost.validation.error <- rmse(data.test$Y, predict(boost.mod, data.test))
boost.validation.error ## 1.308332
```

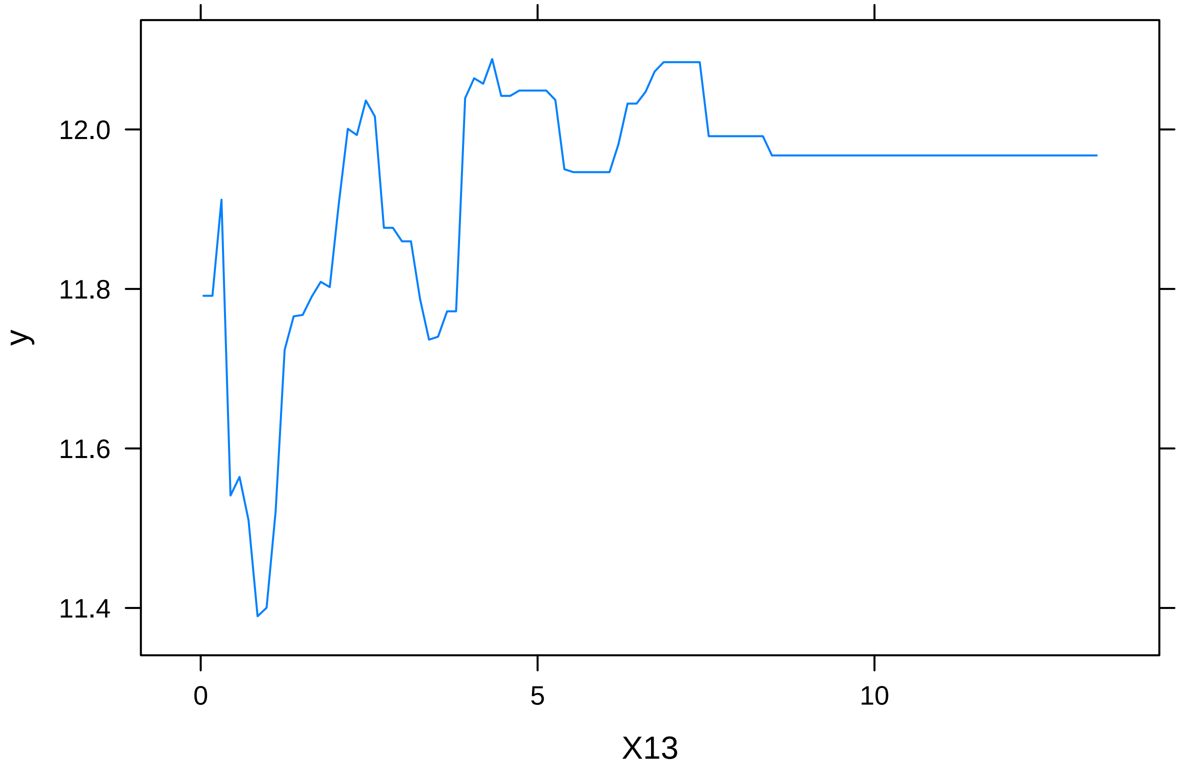
```
## [1] 1.364077
```

```r
# relative influence plot and relative influence statistics.
summary(boost.mod)
```

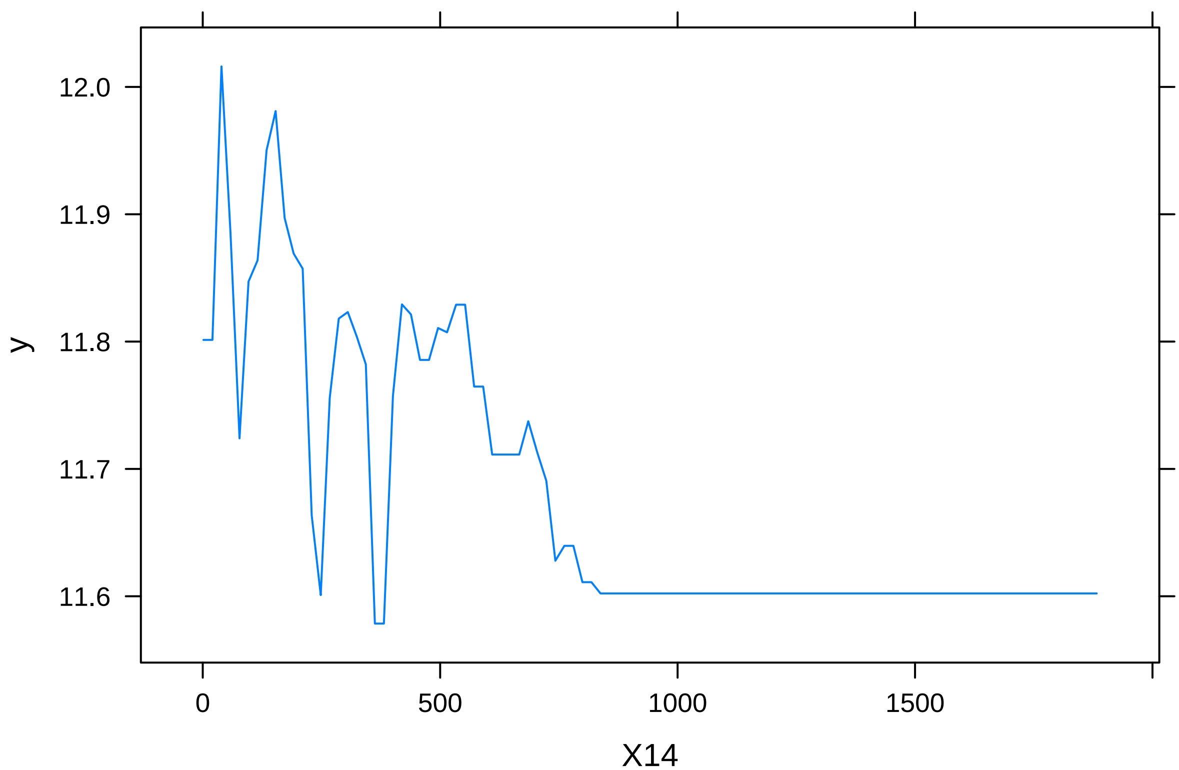



```
##     var   rel.inf
## X14 X14 9.8868827
## X8   X8 9.4582918
## X13 X13 9.4080768
## X11 X11 9.1169055
## X15 X15 8.8917930
## X5   X5 7.9863464
## X4   X4 7.3156479
## X3   X3 7.2870674
## X2   X2 6.5474341
## X1   X1 5.8634476
## X6   X6 5.5680826
## X12 X12 4.8598875
## X9   X9 4.3410971
## X7   X7 2.4985273
## X10 X10 0.9705122
```

```
par(mfrow=c(1,2))
plot(boost.mod ,i="X13") # 1st most impt var
```




```
plot(boost.mod ,i="X14") # 2nd most impt var
```



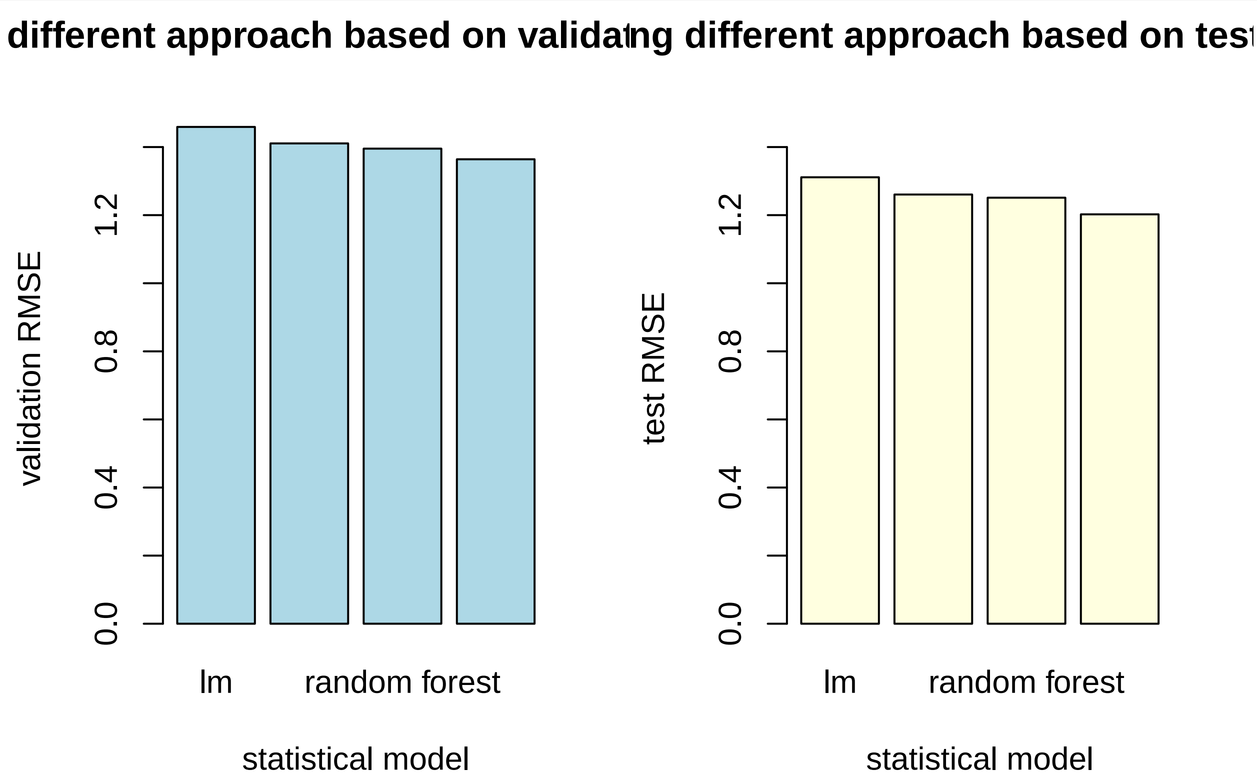
```
# test rmse in kaggle - 1.20241
boost.best <- gbm(Y~., data=train, distribution = "gaussian", n.trees =
                    bestntree, interaction.depth = bestinteraction.depth, shrinkage = bestlambda)

pred <- predict(boost.best, test,n.trees = bestntree)
boost.test.rmse <- 1.20241
```

(5) Summary of figures in the report

```
par(mfrow=c(1,2))
# Fig.1 Comparing validation error rate for 4 different approaches above
ynames <- c("lm", "bagging", "random forest", "boosting")
validation <- c(lm.validation.error, bag.validation.error, rf.validation.error, boost.validation.error)
barplot(height = validation, names = ynames,
        main ="Fig.1 Comparing different approach based on validation error rate (RMSE)",
        xlab = "statistical model",ylab = "validation RMSE",
        ylim = c(0, 1.5), col = "light blue")

# Fig. 2 Comparing test error rate for 4 different approaches above
test <- c(lm.test.rmse, bag.test.rmse, rf.test.rmse, boost.test.rmse)
barplot(height = test, names = ynames,
        main ="Fig.2 Comparing different approach based on test error rate (RMSE)",
        xlab = "statistical model",ylab = "test RMSE",
        ylim = c(0, 1.5), col = "light yellow")
```




```
# Fig. 3 relative influence plot
par(mfrow=c(1,1))
set.seed(10)
boost.mod <- gbm(Y~., data=data.train, distribution="gaussian",
```

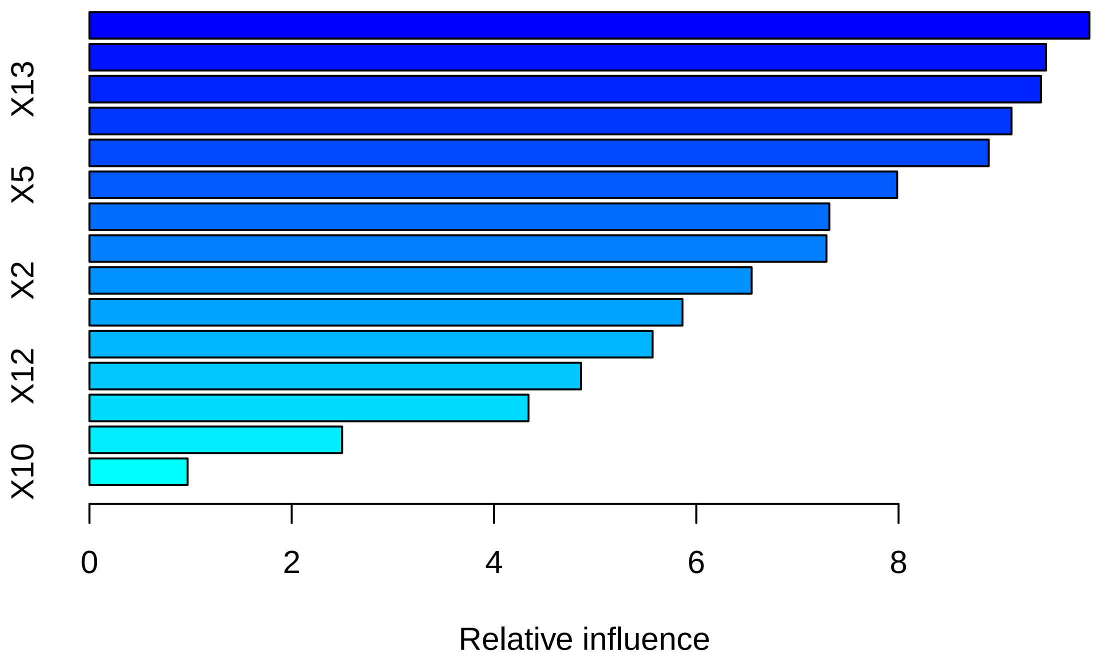
```
                  n.trees=bestntree, interaction.depth=bestinteraction.depth, shrinkage=bestlambda)
boost.validation.error <- rmse(data.test$Y, predict(boost.mod, data.test))
```

```
## Using 500 trees...
```

```
boost.validation.error
```

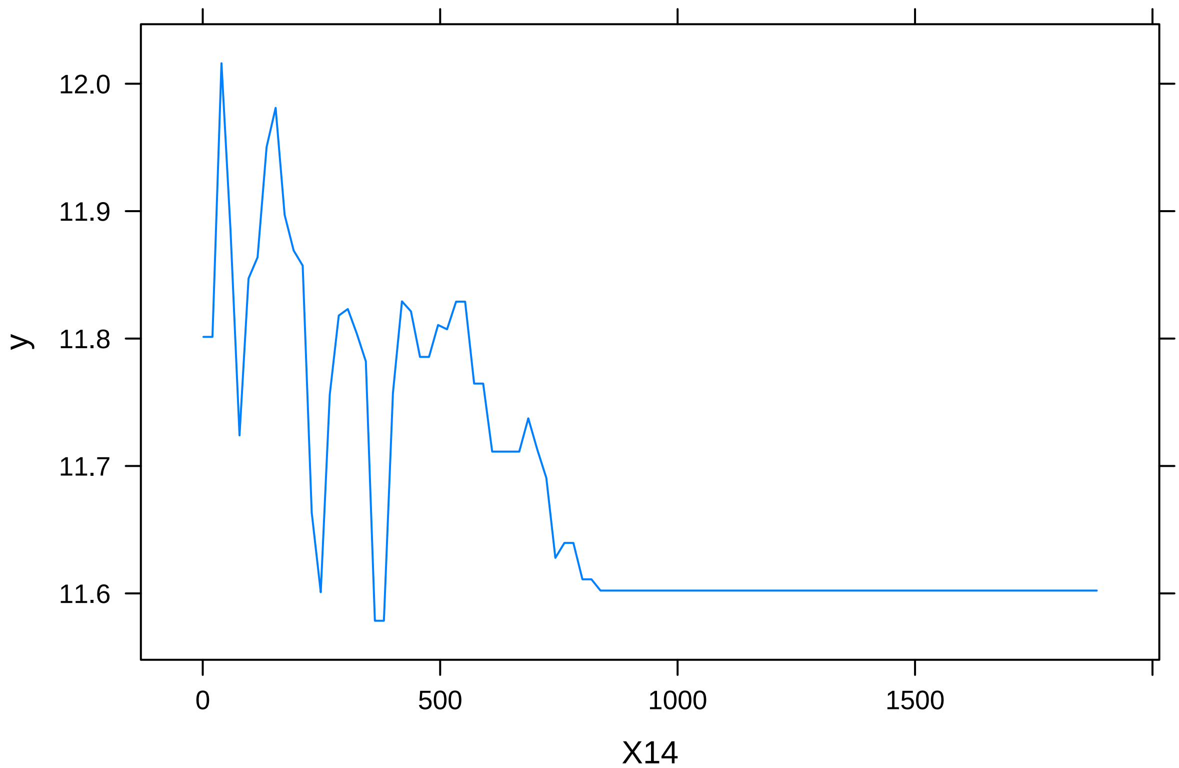
```
## [1] 1.364077
```

```
summary(boost.mod)
```




```
##     var   rel.inf
## X14 X14 9.8868827
## X8   X8 9.4582918
## X13 X13 9.4080768
## X11 X11 9.1169055
## X15 X15 8.8917930
## X5   X5 7.9863464
## X4   X4 7.3156479
## X3   X3 7.2870674
## X2   X2 6.5474341
## X1   X1 5.8634476
## X6   X6 5.5680826
## X12 X12 4.8598875
## X9   X9 4.3410971
## X7   X7 2.4985273
## X10 X10 0.9705122
```

```
# Fig. 4 partial dependence plot
par(mfrow=c(1,2))
plot(boost.mod ,i="X14") # 1st most impt var
```




```
plot(boost.mod ,i="X8") # 2nd most impt var
```