Housing Price Analysis using Random Forest

Bangda Sun

May 14, 2017

We see that linear regression model works well on predicting house price, the final model's R^2 we get is approximately 0.874, which is very good in practice.

Next we try to move beyond linearity, one of the most important family of models in machine learning is tree models, includeing decision trees, bagging and random forest etc.

The predictors we used in the linear regression model includes:

sqftliving, bedrooms, bathrooms, grade, floors, waterfront, yrbuilt, lat, long, view, zipcode, condition.

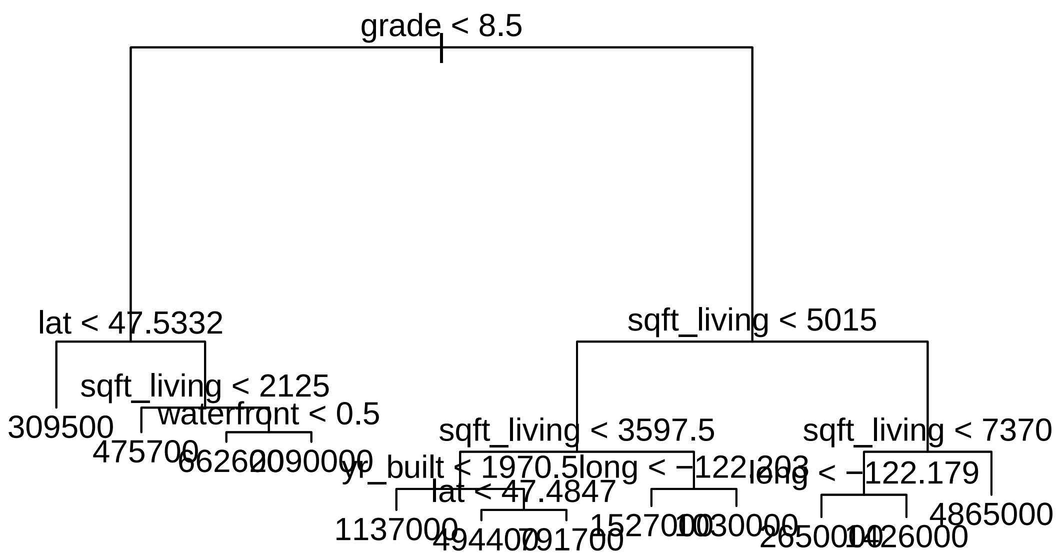
1. Decision Tree

First we start from single decision tree,

```r
library(tree)
tree1 = tree(price ~. -date-zipcode, data = training)
summary(tree1)
```

```
##
## Regression tree:
## tree(formula = price ~ . - date - zipcode, data = training)
## Variables actually used in tree construction:
## [1] "grade"       "lat"         "sqft_living" "waterfront"  "yr_built"
## [6] "long"
## Number of terminal nodes:  12
## Residual mean deviance:  4.304e+10 = 3.046e+14 / 7076
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -4065000   -99520   -25680        0    68920  2195000
```

```r
# Visualize the tree
plot(tree1)
text(tree1)
```




```r
# Text description of tree
tree1
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 7088 9.947e+14  537000
##    2) grade < 8.5 5786 2.312e+14  437800
##      4) lat < 47.5332 2442 3.164e+13  309500 *
##      5) lat > 47.5332 3344 1.300e+14  531500
##       10) sqft_living < 2125 2384 5.235e+13  475700 *
##       11) sqft_living > 2125 960 5.185e+13  670000
##         22) waterfront < 0.5 955 4.025e+13  662600 *
##         23) waterfront > 0.5 5 1.466e+12 2090000 *
##    3) grade > 8.5 1302 4.537e+14  977700
##      6) sqft_living < 5015 1233 2.128e+14  907100
##       12) sqft_living < 3597.5 920 9.172e+13  803100
##         24) yr_built < 1970.5 163 2.071e+13 1137000 *
##         25) yr_built > 1970.5 757 4.892e+13  731200
##           50) lat < 47.4847 154 2.820e+12  494400 *
##           51) lat > 47.4847 603 3.526e+13  791700 *
##       13) sqft_living > 3597.5 313 8.197e+13 1213000
##         26) long < -122.203 115 3.954e+13 1527000 *
##         27) long > -122.203 198 2.447e+13 1030000 *
##      7) sqft_living > 5015 69 1.248e+14 2239000
##       14) sqft_living < 7370 63 5.722e+13 1989000
##         28) long < -122.179 29 2.412e+13 2650000 *
```

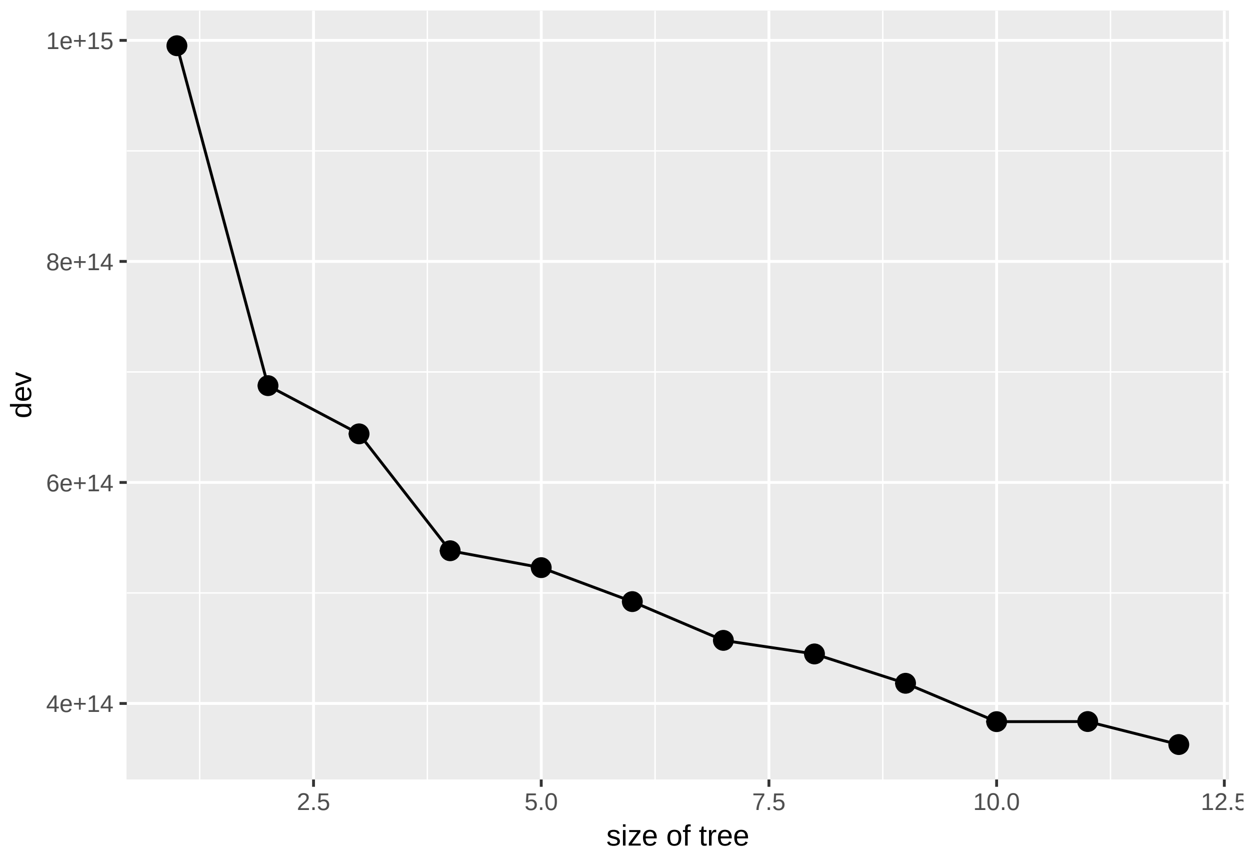
```
##         29) long > -122.179 34 9.649e+12 1426000 *
##       15) sqft_living > 7370 6 2.232e+13 4865000 *
```

Then we use cross validation to see whether pruning the tree will improve performance,

```
cv_tree1 = cv.tree(tree1)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
ggplot() +
  geom_line(mapping = aes(x = cv_tree1$size, y = cv_tree1$dev)) +
  geom_point(mapping = aes(x = cv_tree1$size, y = cv_tree1$dev), size = 3) +
  labs(x = "size of tree", y = "dev")
```



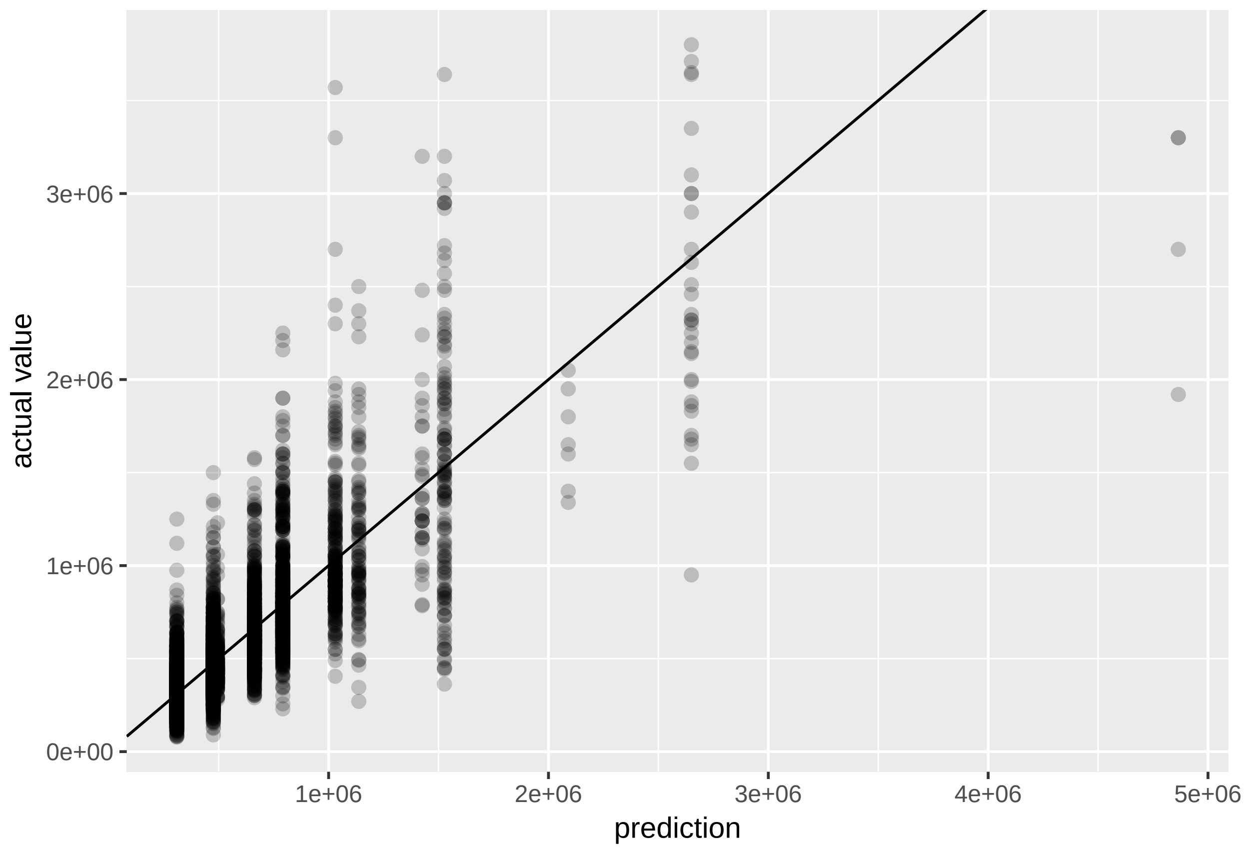
We can see that the dev of tree will get minimum when the tree size is 12. Therefore, the performance of tree doesn't improve much if we prune the tree.

Next we calculate the test error of this tree,

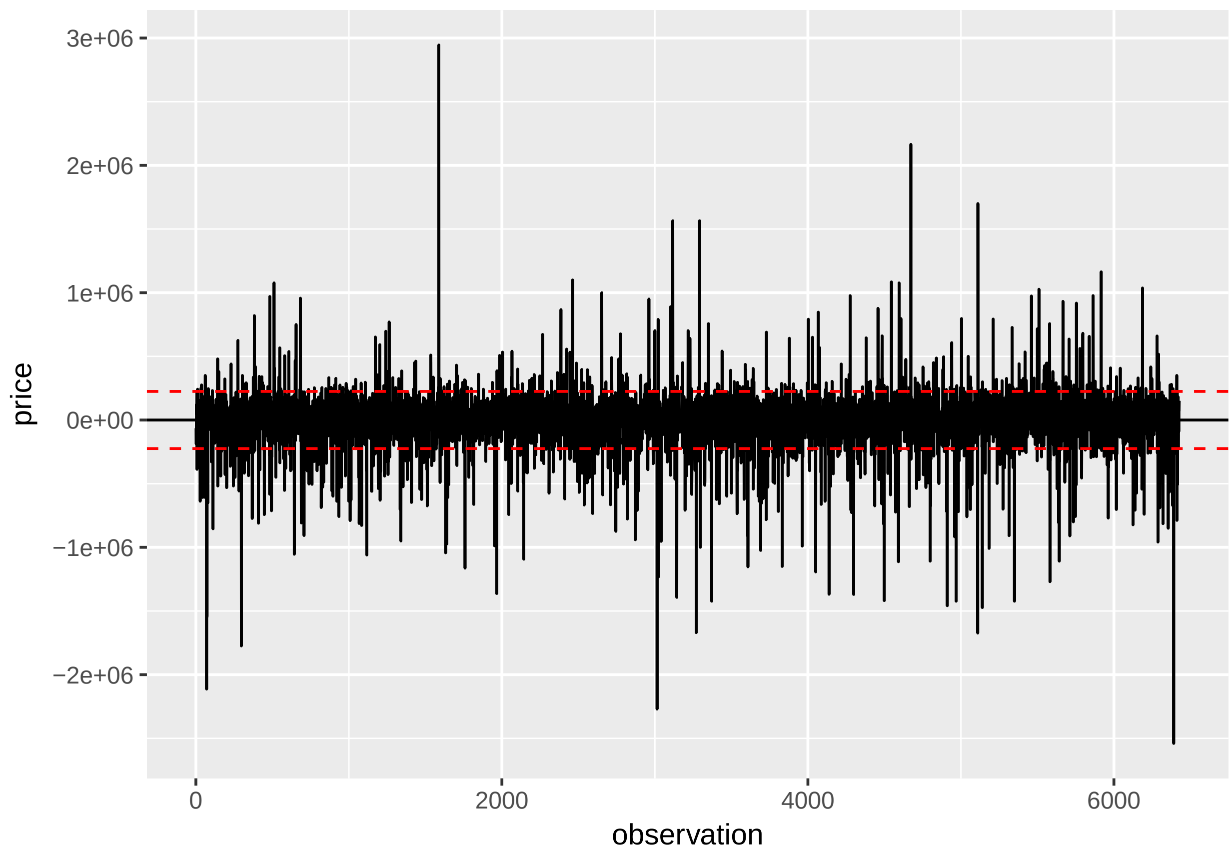
```
predOfTree1 = predict(tree1, newdata = testing)
rmse1 = sqrt(mean((predOfTree1 - testing$price)^2))
rmse1
```

```
## [1] 224316.3
```

```
ggplot() +
  geom_point(mapping = aes(x = predOfTree1, y = testing$price), alpha = .2, size = 2) +
  geom_abline(slope = 1, intercept = 0) +
  labs(x = "prediction", y = "actual value")
```




```
ggplot() +
  geom_line(mapping = aes(x = 1:length(predOfTree1), y = predOfTree1 - testing$price)) +
  geom_abline(slope = 0, intercept = 0) +
  geom_abline(slope = 0, intercept = rmse1, linetype = "dashed", color = "red") +
  geom_abline(slope = 0, intercept = -rmse1, linetype = "dashed", color = "red") +
  labs(x = "observation", y = "price")
```



From this plot, we can see that many predictions are far away from their actual value. The error is still high. We need to seek models to predict more precisely.

2. Random Forest

We use ensemble methods to improve the performance of the tree model. One of the popular methods is random forest.

```
library(randomForest)
```

```
## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(1)
randForest = randomForest(I(log(price)) ~. -id-zipcode-date, data = training, mtry = 5,
                          importance = TRUE)
randForest
```

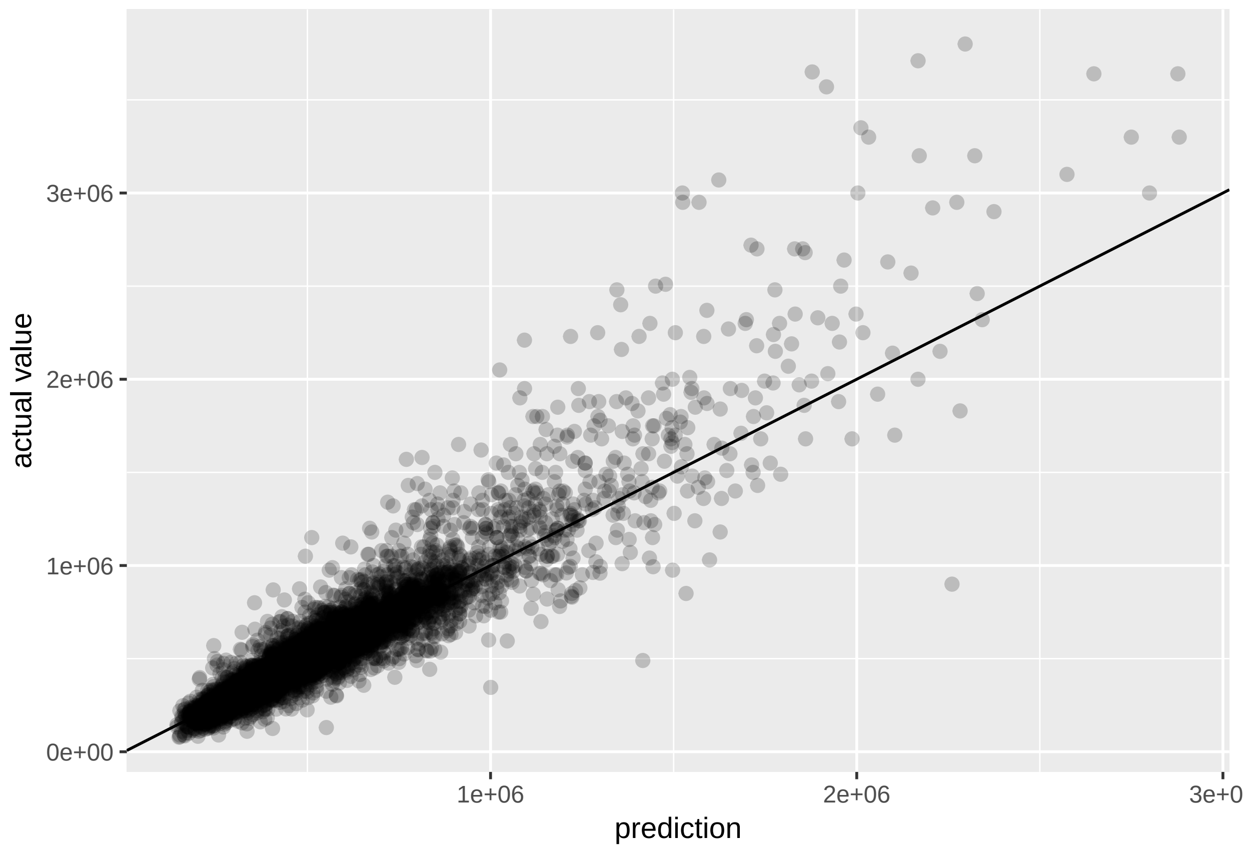
```
##
## Call:
```

```
##  randomForest(formula = I(log(price)) ~ . - id - zipcode - date,      data = training, mtry = 5, impo
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 0.03586097
##                     % Var explained: 87.29
```

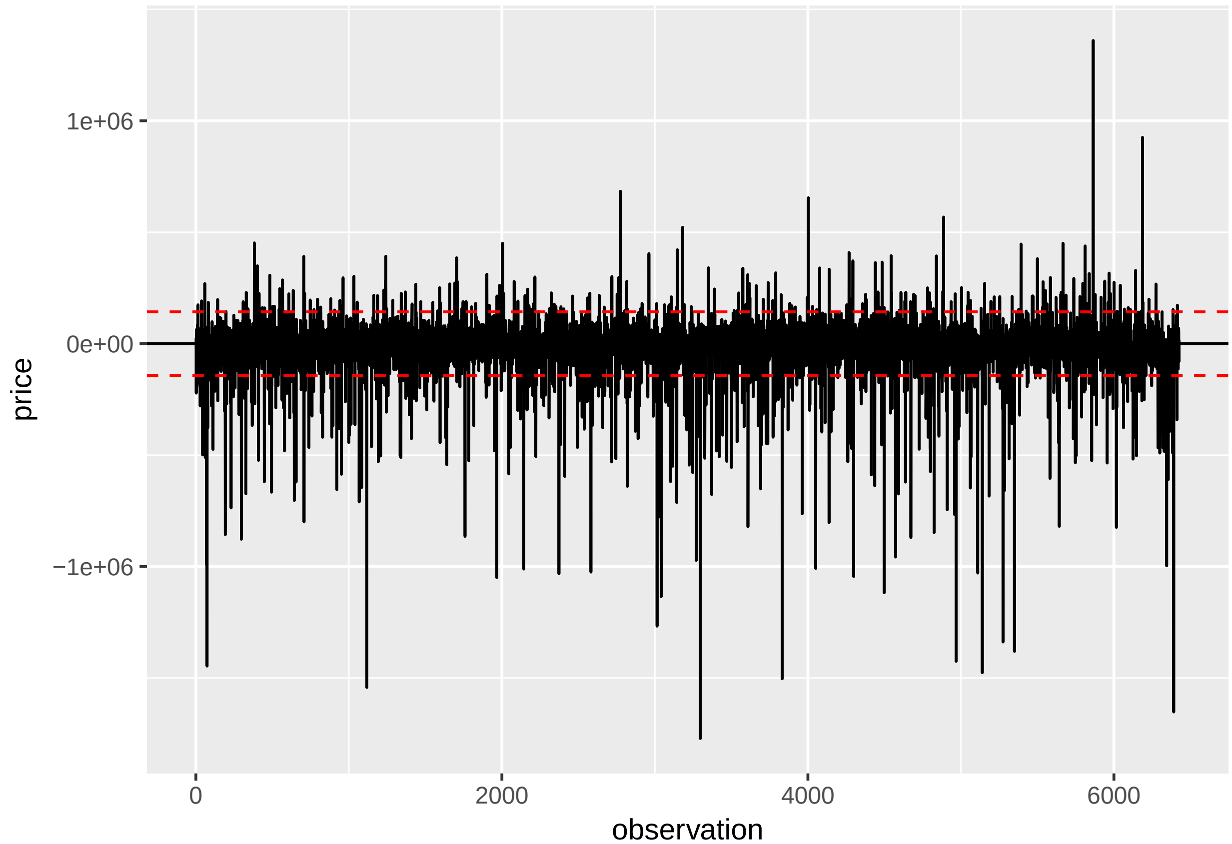
```
predOfrandForest = predict(randForest, newdata = testing)
rmse2 = sqrt(mean((testing$price - exp(predOfrandForest))^2))
rmse2
```

```
## [1] 142801.7
```

```
ggplot() +
  geom_point(mapping = aes(x = exp(predOfrandForest), y = testing$price), alpha = .2, size = 2) +
  geom_abline(slope = 1, intercept = 0) +
  labs(x = "prediction", y = "actual value")
```



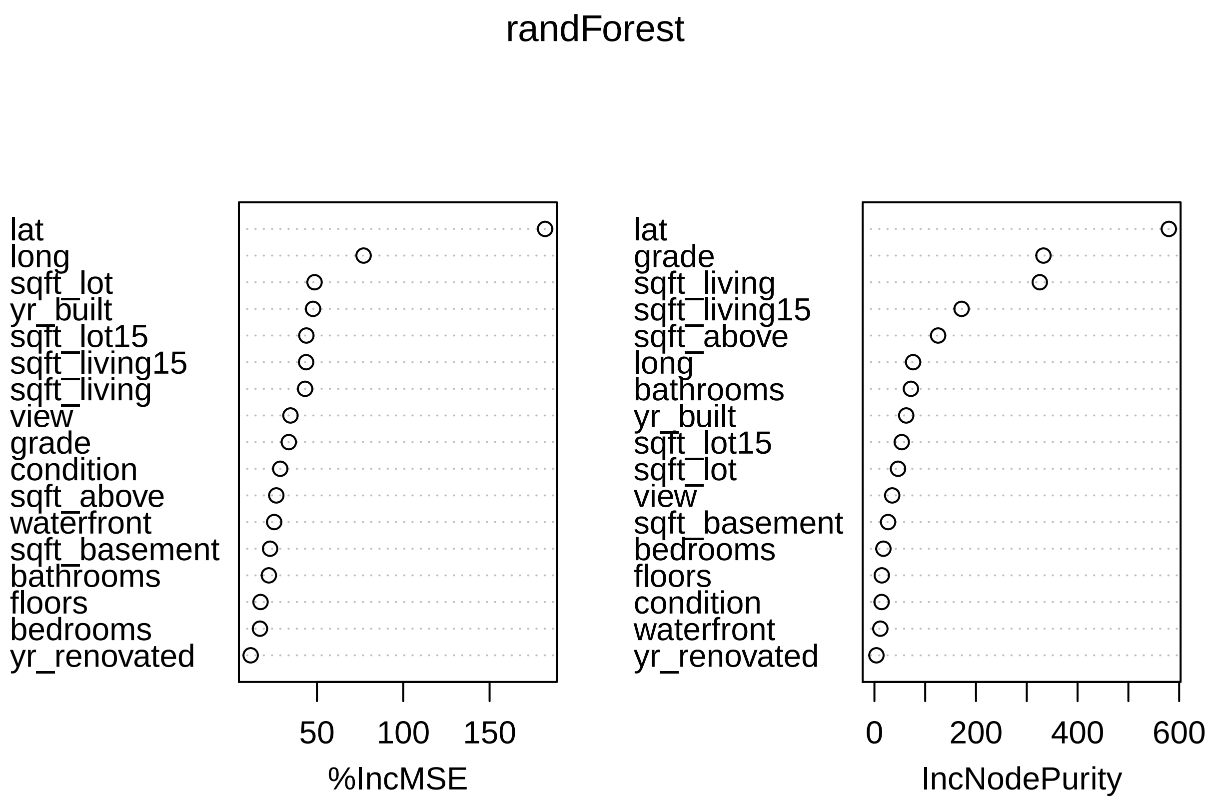

```
ggplot() +
  geom_line(mapping = aes(x = 1:length(predOfrandForest), y = exp(predOfrandForest) - testing$price)) +
  geom_abline(slope = 0, intercept = 0) +
  geom_abline(slope = 0, intercept = rmse2, linetype = "dashed", color = "red") +
  geom_abline(slope = 0, intercept = -rmse2, linetype = "dashed", color = "red") +
  labs(x = "observation", y = "price")
```




```r
importance(randForest)
```

```
##                  %IncMSE IncNodePurity
## bedrooms        16.98279     17.682193
## bathrooms       22.14704     71.858680
## sqft_living     43.12953    325.628456
## sqft_lot        48.61245     46.454551
## floors          17.32741     14.573919
## waterfront      25.23851     11.689565
## view            34.66992     35.028867
## condition       28.71623     14.200844
## grade           33.68949    332.797907
## sqft_above      26.46916    125.406629
## sqft_basement   22.85204     26.940917
## yr_built        47.72134     62.636548
## yr_renovated    11.61159      3.968069
## lat            182.13863    579.685747
## long            77.01030     76.154332
## sqft_living15   43.72745    171.641281
## sqft_lot15      43.87485     53.835669
```

```r
varImpPlot(randForest)
```



randForest
lat
long
sqft_lot
yr_built
sqft_lot15
sqft_living15
sqft_living
view
grade
condition
sqft_above
waterfront
sqft_basement
bathrooms
floors
bedrooms
yr_renovated
50
100
150
%IncMSE
lat
grade
sqft_living
sqft_living15
sqft_above
long
bathrooms
yr_built
sqft_lot15
sqft_lot
view
sqft_basement
bedrooms
floors
condition
waterfront
yr_renovated
0
200
400
600
IncNodePurity