trees

Mengqing Zhang

4/26/2020

select the import features

```
bjList$neighbourhood <- as.factor(bjList$neighbourhood)

names(bosList)[33:42] <- c("host_resp_wt_a_day","host_resp_wt_a_few_hrs","host_resp
_wt_an_hr",
  "bed_type_Couch","bed_type_Futon","bed_type_Sofa","bed_type_Bed",
"room_type_Hotel_room","room_type_Private_room","room_type_Shared_room" )

names(bjList)[33:41] <- c("host_resp_wt_a_day","host_resp_wt_a_few_hrs","host_resp_
wt_an_hr",
  "bed_type_Couch","bed_type_Futon",
  "bed_type_Sofa","bed_type_Bed",
  "room_type_Private_room",
  "room_type_Shared_room" )

bosList<- bosList[,-1]
bjList <- bjList[,-1]

bosList <- bosList %>% fastDummies::dummy_cols(select_columns=c('neighbourhood_clea
nsed'), remove_first_dummy = T)

bosList <- bosList %>%
  dplyr::select(host_listings_count,
                   accommodates, bathrooms,bedrooms,beds,
                   guests_included,
                   maximum_nights,number_of_reviews,
                   number_of_reviews_ltm,wifi_available,
                   host_response_time_nodata,
                   cancellation_policy_strict,
                   price, 'neighbourhood_cleansed_Bay Village',
                'neighbourhood_cleansed_Beacon Hill',
                'neighbourhood_cleansed_Mattapan',
                'neighbourhood_cleansed_South Boston',
                'neighbourhood_cleansed_South Boston Waterfront',
                'neighbourhood_cleansed_West End')

names(bosList)[14:19] <- c( 'neighbourhood_cleansed_Bay_Village',
                'neighbourhood_cleansed_Beacon_Hill',
                'neighbourhood_cleansed_Mattapan',
                'neighbourhood_cleansed_South_Boston',
                'neighbourhood_cleansed_South_Boston_Waterfront',
```

```
                'neighbourhood_cleansed_West_End')

bjList <- bjList%>% fastDummies::dummy_cols(select_columns=c('neighbourhood'), remo
ve_first_dummy = T)
bjList <- bjList %>% dplyr::select(host_listings_count,
                  accommodates,bathrooms,bedrooms,beds,minimum_nights,
                  maximum_nights,
                   guests_included,
                   availability_30,availability_90,
                   availability_365,number_of_reviews,
                   number_of_reviews_ltm,TV_available,
                   wc_access,room_type_Shared_room,price,
                  host_resp_wt_a_day,room_type_Private_room)
```

Preparation

```
set.seed(68)
# This will split into train and test 75-25
bosList$train <- sample(c(0, 1), nrow(bosList), replace = TRUE, prob = c(.25, .75))
boslist_test <- bosList %>% filter(train == 0)%>% mutate_if(is.character, as.factor
)
boslist_train <- bosList %>% filter(train == 1)%>% mutate_if(is.character, as.facto
r)

bjList$train <- sample(c(0, 1), nrow(bjList), replace = TRUE, prob = c(.25, .75))
bjList_test <- bjList %>% filter(train == 0)%>% mutate_if(is.character, as.factor)
bjList_train <- bjList %>% filter(train == 1)%>% mutate_if(is.character, as.factor)

# #delete the neighborhood column
# boslist_train <- boslist_train[,-4]
# boslist_test <- boslist_test[,-4]
#
# bjList_train <- bjList_train[,-4]
# bjList_test <-  bjList_test[,-4]

#delete the last train column(0,1)
boslist_train <- boslist_train[,-ncol(boslist_train)]
boslist_test <- boslist_test[,-ncol(boslist_test)]

bjList_train <- bjList_train[,-ncol(bjList_train)]
bjList_test <-  bjList_test[,-ncol(bjList_test)]
```

Regression Tree–Boston

```r
set.seed(68)
#Regression tree
fit.tree <- rpart(price~.,
                  boslist_train,
                  control = rpart.control(cp = 0.0001))
par(xpd = TRUE)

## Printcp will tell you what the cp of spliting into diffrent number layer and the xerror and xstd of each cp.
printcp(fit.tree)
```

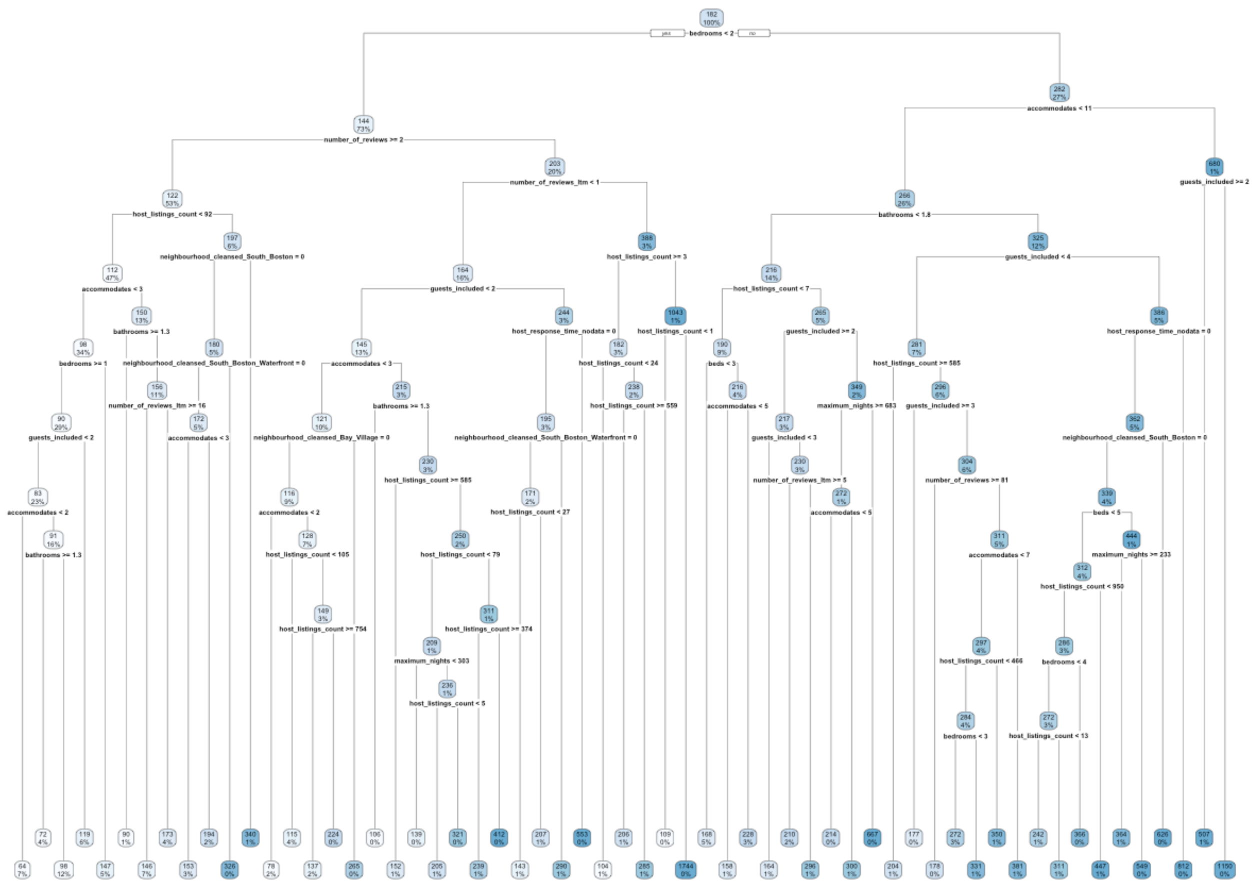
```
##
## Regression tree:
## rpart(formula = price ~ ., data = boslist_train, control = rpart.control(cp = 1e
-04))
##
## Variables actually used in tree construction:
##  [1] accommodates
##  [2] bathrooms
##  [3] bedrooms
##  [4] beds
##  [5] guests_included
##  [6] host_listings_count
##  [7] host_response_time_nodata
##  [8] maximum_nights
##  [9] neighbourhood_cleansed_Bay_Village
## [10] neighbourhood_cleansed_South_Boston
## [11] neighbourhood_cleansed_South_Boston_Waterfront
## [12] neighbourhood_cleansed_West_End
## [13] number_of_reviews
## [14] number_of_reviews_ltm
##
## Root node error: 261549359/2561 = 102128
##
## n= 2561
##
##            CP nsplit rel error  xerror    xstd
## 1  0.03710219      0   1.00000 1.00056 0.52455
## 2  0.03027732      1   0.96290 0.98228 0.52999
## 3  0.01637646      5   0.84179 1.03006 0.53017
## 4  0.00808889      6   0.82541 1.03094 0.53006
## 5  0.00763548      7   0.81732 0.98180 0.50100
## 6  0.00417432      8   0.80969 0.98111 0.50101
## 7  0.00372139     10   0.80134 0.97911 0.50094
## 8  0.00355259     11   0.79762 0.98047 0.50094
## 9  0.00289622     13   0.79051 0.97711 0.50169
## 10 0.00269160     14   0.78762 0.97841 0.50168
## 11 0.00240100     17   0.77954 0.97740 0.50168
## 12 0.00218476     18   0.77714 0.97241 0.50160
```

```
## 13 0.00139790     19   0.77496 0.96466 0.50158
## 14 0.00129089     20   0.77356 0.96042 0.50092
## 15 0.00123318     21   0.77227 0.95978 0.50092
## 16 0.00113419     23   0.76980 0.96057 0.50102
## 17 0.00082532     24   0.76867 0.96015 0.50102
## 18 0.00074251     25   0.76784 0.95884 0.50156
## 19 0.00066091     26   0.76710 0.95917 0.50171
## 20 0.00063267     27   0.76644 0.95940 0.50171
## 21 0.00060236     28   0.76580 0.95977 0.50171
## 22 0.00059644     29   0.76520 0.95928 0.50171
## 23 0.00055109     30   0.76461 0.95972 0.50177
## 24 0.00052288     34   0.76240 0.95936 0.50212
## 25 0.00051752     35   0.76188 0.95895 0.50212
## 26 0.00051298     36   0.76136 0.95883 0.50212
## 27 0.00049632     38   0.76034 0.95870 0.50212
## 28 0.00041868     39   0.75984 0.95896 0.50218
## 29 0.00032869     40   0.75942 0.95868 0.50218
## 30 0.00031814     41   0.75909 0.95848 0.50218
## 31 0.00029664     42   0.75877 0.95868 0.50218
## 32 0.00028673     43   0.75848 0.95865 0.50218
## 33 0.00028122     44   0.75819 0.95862 0.50218
## 34 0.00027103     45   0.75791 0.95869 0.50218
## 35 0.00026260     47   0.75737 0.95881 0.50218
## 36 0.00022880     49   0.75684 0.95875 0.50218
## 37 0.00022800     50   0.75661 0.95848 0.50218
## 38 0.00021375     51   0.75638 0.95812 0.50182
## 39 0.00020857     52   0.75617 0.95826 0.50182
## 40 0.00020504     53   0.75596 0.95824 0.50182
## 41 0.00020160     55   0.75555 0.95819 0.50182
## 42 0.00020132     56   0.75535 0.95816 0.50182
## 43 0.00019798     57   0.75515 0.95822 0.50182
## 44 0.00018840     58   0.75495 0.95802 0.50182
## 45 0.00016673     59   0.75476 0.95805 0.50182
## 46 0.00016394     60   0.75460 0.95844 0.50193
## 47 0.00016020     61   0.75443 0.95838 0.50193
## 48 0.00014897     62   0.75427 0.96001 0.50208
## 49 0.00014027     63   0.75412 0.95992 0.50208
## 50 0.00013862     64   0.75398 0.95993 0.50208
## 51 0.00013214     65   0.75384 0.95985 0.50208
## 52 0.00013165     66   0.75371 0.96000 0.50208
## 53 0.00012761     67   0.75358 0.96002 0.50208
## 54 0.00012680     68   0.75345 0.95991 0.50208
## 55 0.00012612     69   0.75333 0.95989 0.50208
## 56 0.00011677     70   0.75320 0.96087 0.50277
## 57 0.00011550     71   0.75308 0.96121 0.50277
## 58 0.00010928     74   0.75274 0.96126 0.50277
## 59 0.00010824     75   0.75263 0.96150 0.50278
## 60 0.00010496     77   0.75241 0.96163 0.50278
## 61 0.00010211     83   0.75176 0.96157 0.50278
## 62 0.00010000     85   0.75156 0.96141 0.50278
```

```
## We can use the following method to choose the cp with the smallest xerror
fit.tree$cptable[which.min(fit.tree$cptable[,"xerror"]),"CP"]
```

```
## [1] 0.0001884012
```

```
## Build the tree model with the cp which has smallest xerror
tree2 <- prune(fit.tree, cp= fit.tree$cptable[which.min(fit.tree$cptable[,"xerror"]
),"CP"])
## Make the visuallization of regreesion tree
rpart.plot(tree2)
```



```
## MSE of train for Boston
tree.pred.train = predict(tree2,boslist_train)
mean((tree.pred.train-boslist_train$price)^2)
```

```
## [1] 77101.57
```

```
## MSE of test for Boston
tree.pred.test = predict(tree2,boslist_test)
mean((tree.pred.test-boslist_test$price)^2)
```

```
## [1] 47502.59
```

Regression Tree–Beijing

```
#Regression tree
fit.tree <- rpart(price~.,
                  bjList_train,
                  control = rpart.control(cp = 0.0001))
## Printcp will tell you what the cp of spliting into diffrent number layer and the
xerror and xstd of each cp.
printcp(fit.tree)
```

```
##
## Regression tree:
## rpart(formula = price ~ ., data = bjList_train, control = rpart.control(cp = 1e-
04))
##
## Variables actually used in tree construction:
##  [1] accommodates           availability_30        availability_365
##  [4] availability_90        bathrooms              bedrooms
##  [7] beds                   guests_included        host_listings_count
## [10] host_resp_wt_a_day     maximum_nights         minimum_nights
## [13] number_of_reviews      number_of_reviews_ltm  room_type_Private_room
## [16] room_type_Shared_room  TV_available
##
## Root node error: 1.7126e+10/11339 = 1510326
##
## n= 11339
##
##             CP nsplit rel error  xerror    xstd
## 1  0.05910445      0   1.00000 1.00021 0.38630
## 2  0.01825529      1   0.94090 0.94924 0.38588
## 3  0.01777481      2   0.92264 0.93084 0.38579
## 4  0.01675148      3   0.90487 0.98388 0.38619
## 5  0.01454429      4   0.88811 0.97752 0.38470
## 6  0.00597789      9   0.81418 0.97620 0.38512
## 7  0.00421139     10   0.80820 0.99905 0.38521
## 8  0.00355436     11   0.80399 0.99544 0.38428
## 9  0.00300284     12   0.80043 0.99237 0.38409
## 10 0.00282454     13   0.79743 0.99674 0.38404
## 11 0.00266045     14   0.79461 0.99671 0.38404
## 12 0.00258507     15   0.79195 0.99865 0.38404
## 13 0.00208874     16   0.78936 0.99836 0.38428
```

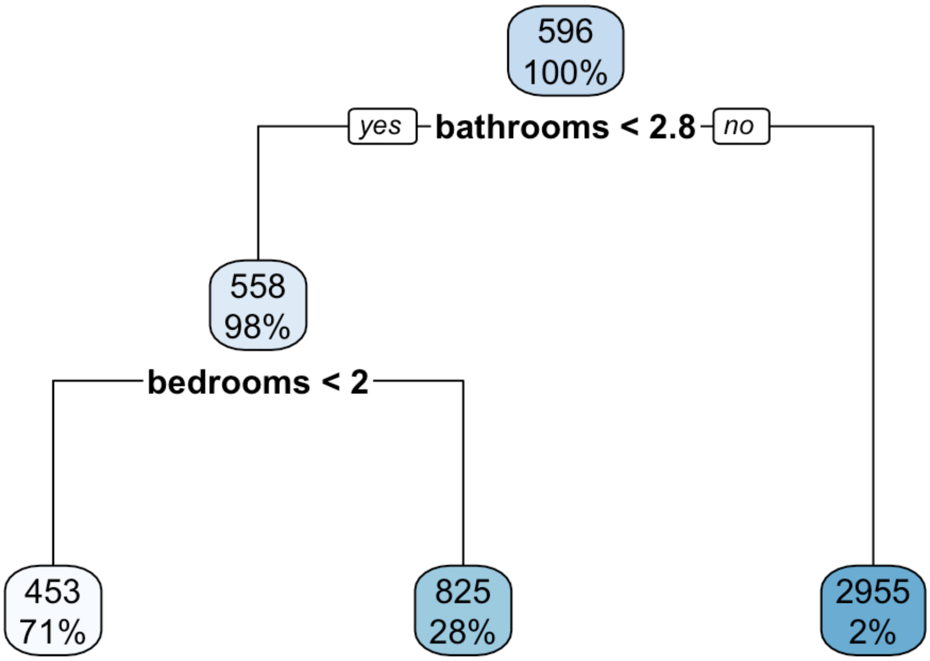
```
## 14 0.00134262     19   0.78309 0.99916 0.38441
## 15 0.00131161     20   0.78175 1.00126 0.38450
## 16 0.00099309     24   0.77651 1.00184 0.38444
## 17 0.00093220     25   0.77551 1.00180 0.38445
## 18 0.00088688     26   0.77458 1.00166 0.38445
## 19 0.00082806     27   0.77369 1.00146 0.38445
## 20 0.00081555     29   0.77204 1.00195 0.38445
## 21 0.00080145     30   0.77122 1.00202 0.38445
## 22 0.00077030     31   0.77042 1.00183 0.38445
## 23 0.00071287     32   0.76965 1.00167 0.38450
## 24 0.00070218     33   0.76894 1.00063 0.38450
## 25 0.00067365     35   0.76753 1.00033 0.38450
## 26 0.00066583     37   0.76619 1.00110 0.38453
## 27 0.00066127     38   0.76552 1.00109 0.38453
## 28 0.00064020     41   0.76354 1.00101 0.38454
## 29 0.00063417     42   0.76290 1.00084 0.38454
## 30 0.00062281     43   0.76226 1.00086 0.38454
## 31 0.00061910     44   0.76164 1.00086 0.38454
## 32 0.00059204     45   0.76102 1.00121 0.38454
## 33 0.00053667     46   0.76043 1.00082 0.38444
## 34 0.00052259     47   0.75989 1.00033 0.38444
## 35 0.00050853     48   0.75937 0.99989 0.38443
## 36 0.00045649     49   0.75886 1.00042 0.38470
## 37 0.00044507     50   0.75840 1.00005 0.38470
## 38 0.00044375     51   0.75796 1.00046 0.38470
## 39 0.00040826     53   0.75707 1.00058 0.38470
## 40 0.00037973     55   0.75625 1.00096 0.38470
## 41 0.00036119     56   0.75587 0.99961 0.38455
## 42 0.00035630     57   0.75551 0.99947 0.38455
## 43 0.00033033     58   0.75516 1.00093 0.38512
## 44 0.00031845     59   0.75483 1.00087 0.38485
## 45 0.00031399     61   0.75419 1.00086 0.38485
## 46 0.00031377     62   0.75388 1.00082 0.38485
## 47 0.00031317     63   0.75356 1.00075 0.38485
## 48 0.00030152     64   0.75325 1.00071 0.38485
## 49 0.00030021     65   0.75295 1.00072 0.38485
## 50 0.00029904     67   0.75235 1.00072 0.38485
## 51 0.00029484     70   0.75145 0.99956 0.38484
## 52 0.00027254     74   0.75027 0.99950 0.38484
## 53 0.00027172     78   0.74918 0.99918 0.38483
## 54 0.00026373     79   0.74891 0.99931 0.38483
## 55 0.00025374     80   0.74864 0.99858 0.38482
## 56 0.00024043     81   0.74839 0.99852 0.38482
## 57 0.00022659     82   0.74815 0.99846 0.38482
## 58 0.00020991     84   0.74770 0.99796 0.38480
## 59 0.00020304     86   0.74728 0.99819 0.38480
## 60 0.00019889     87   0.74707 0.99929 0.38490
## 61 0.00019771     88   0.74688 0.99921 0.38490
## 62 0.00019303     89   0.74668 0.99921 0.38490
## 63 0.00019000     92   0.74610 0.99916 0.38490
## 64 0.00018030     94   0.74572 0.99922 0.38489
```

```
## 65 0.00016907     95   0.74554 0.99964 0.38489
## 66 0.00016705     97   0.74520 0.99971 0.38489
## 67 0.00016307     99   0.74487 0.99955 0.38489
## 68 0.00015872    100   0.74470 0.99987 0.38490
## 69 0.00015402    106   0.74375 0.99987 0.38490
## 70 0.00013946    107   0.74360 0.99975 0.38490
## 71 0.00013665    108   0.74346 1.00044 0.38490
## 72 0.00013193    109   0.74332 1.00033 0.38490
## 73 0.00012682    110   0.74319 1.00050 0.38490
## 74 0.00012306    116   0.74243 1.00049 0.38492
## 75 0.00011927    117   0.74231 1.00064 0.38492
## 76 0.00011162    118   0.74219 1.00090 0.38492
## 77 0.00010910    119   0.74207 1.00097 0.38492
## 78 0.00010792    120   0.74197 1.00097 0.38492
## 79 0.00010582    121   0.74186 1.00089 0.38492
## 80 0.00010409    122   0.74175 1.00085 0.38492
## 81 0.00010000    123   0.74165 1.00053 0.38492
```

```
## We can use the following method to choose the cp with the smallest xerror
fit.tree$cptable[which.min(fit.tree$cptable[,"xerror"]),"CP"]
```

```
## [1] 0.01777481
```

```
## Build the tree model with the cp which has smallest xerror
tree2 <- prune(fit.tree, cp= fit.tree$cptable[which.min(fit.tree$cptable[,"xerror"]
),"CP"])
## Make the visuallization of regreesion tree
rpart.plot(tree2)
```




```
## MSE of train for Beijing
tree.pred.train = predict(tree2,bjList_train)
mean((tree.pred.train-bjList_train$price)^2)
```

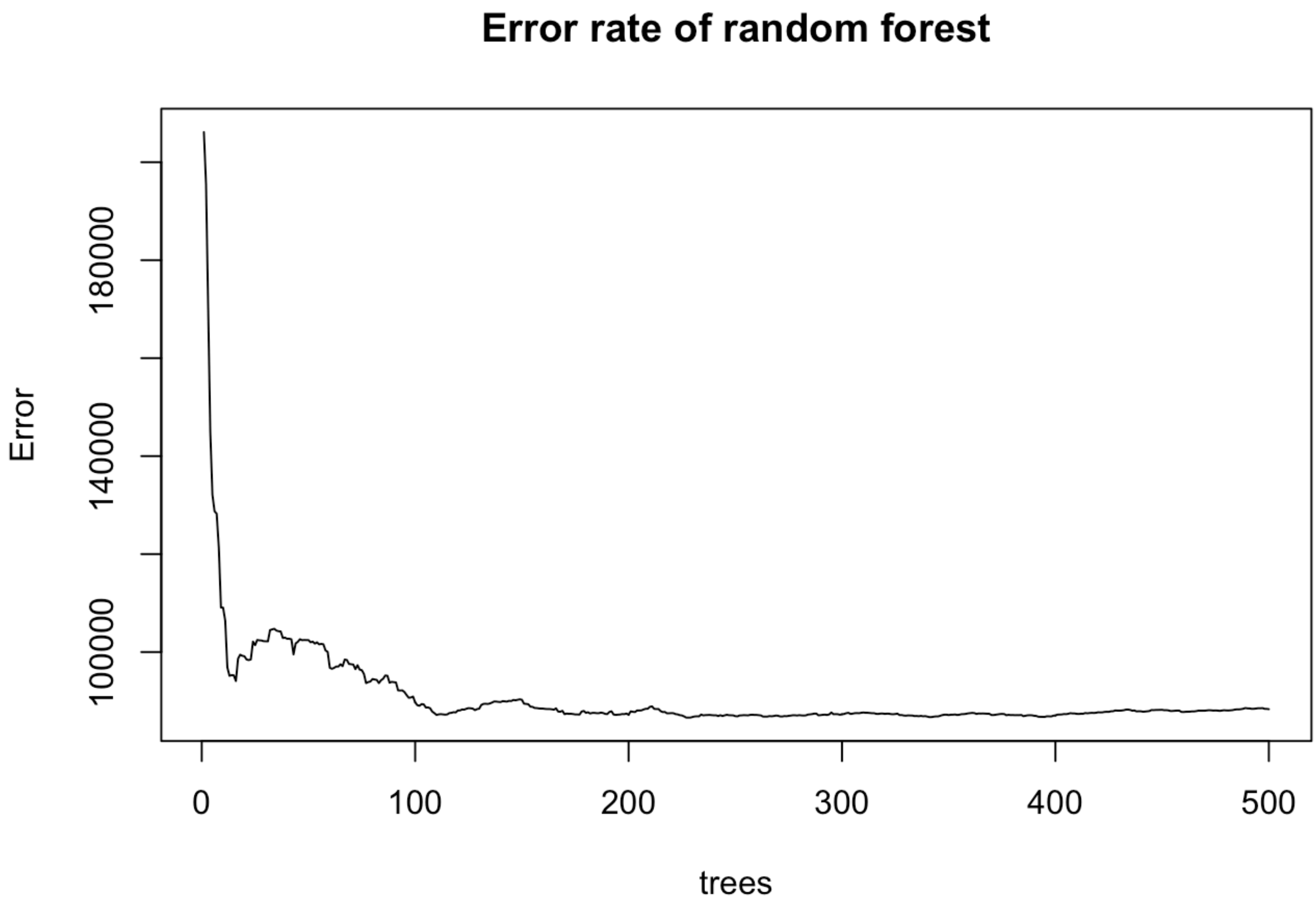
```
## [1] 1393488
```

```
## MSE of test for Beijing
tree.pred.test = predict(tree2,bjList_test)
mean((tree.pred.test-bjList_test$price)^2)
```

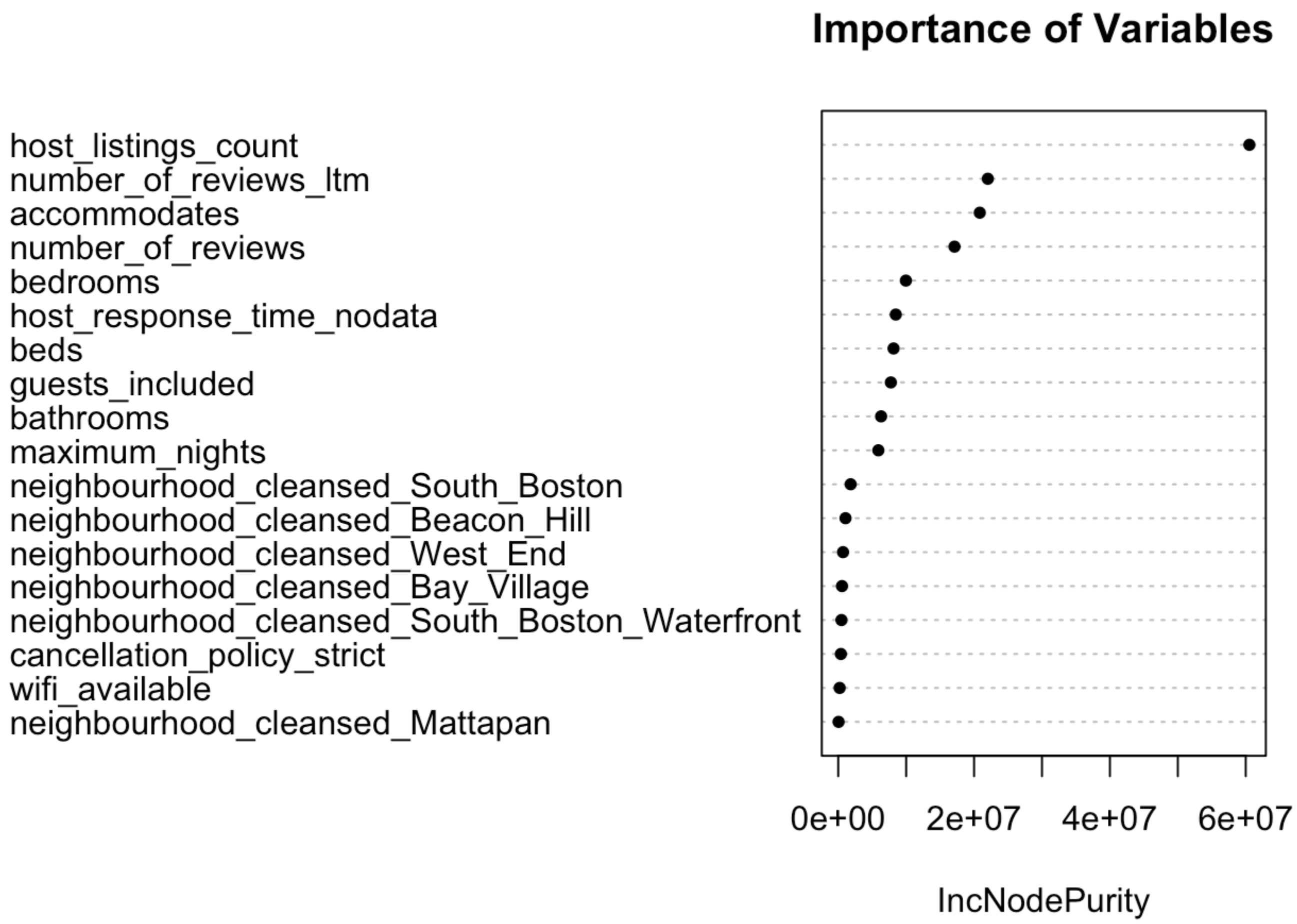
```
## [1] 3758002
```

Random Forest–Boston

```
##Random Forest
#decide ntree by the plot of error vs ntree
error_rf <- randomForest(price ~.,data=boslist_train)
plot(error_rf,main = "Error rate of random forest")
```




```
fit_rf <- randomForest(price~.,
                       boslist_train,
                       ntree=100,
                       do.trace=F)

varImpPlot(fit_rf,pch = 20, main = "Importance of Variables")
```




```
## MSE of train for Boston
yhat_rf <- predict(fit_rf, boslist_train)
train_mse_rf <- mean((yhat_rf - boslist_train$price) ^ 2)
print(train_mse_rf)
```

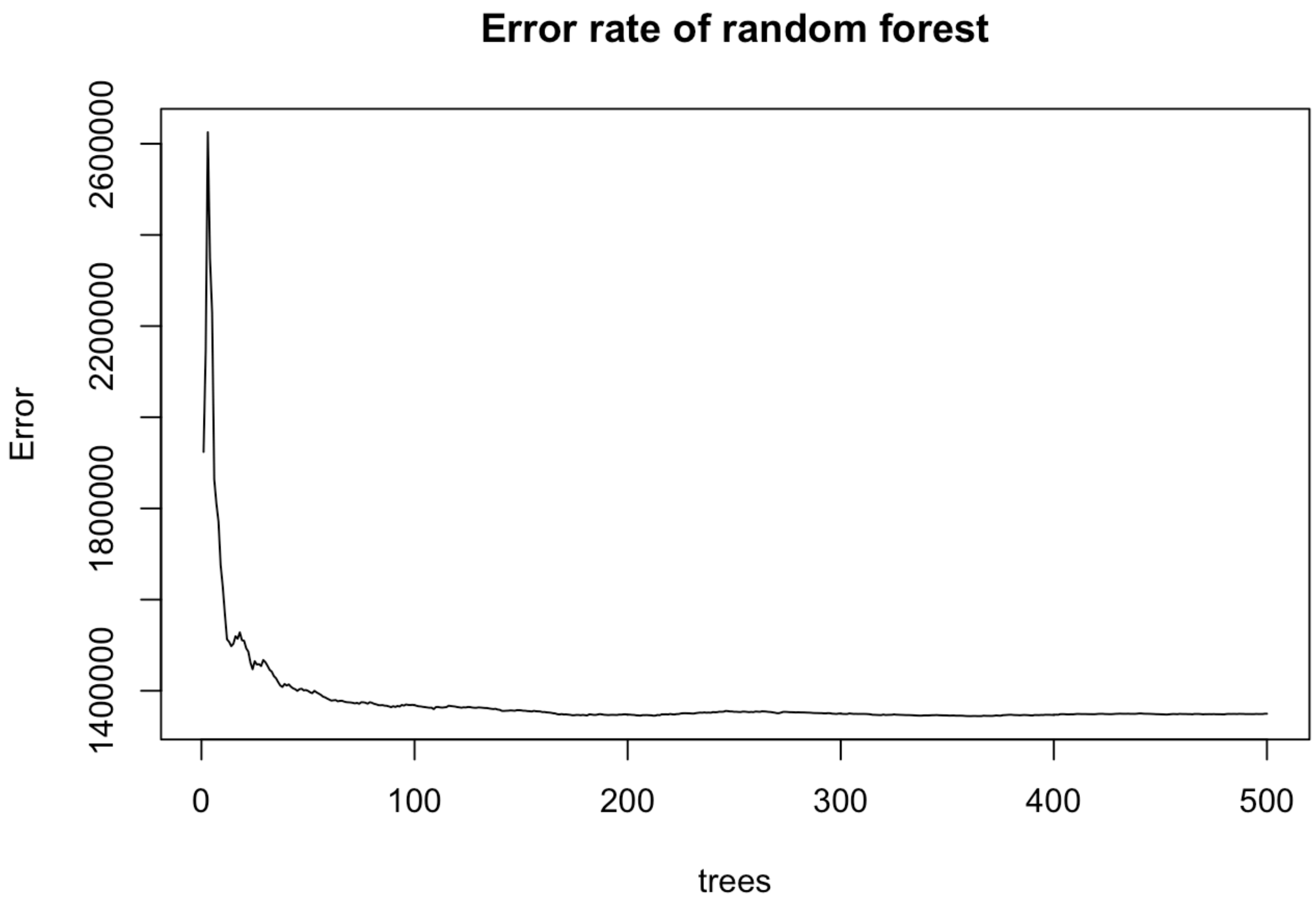
```
## [1] 41871.5
```

```
#levels(boslist_test$neighbourhood_cleansed) = levels(boslist_train$neighbourhood_c
leansed)

## MSE of Test for Boston
yhat_rf <- predict(fit_rf, boslist_test)
test_mse_rf <- mean((yhat_rf - boslist_test$price) ^ 2)
print(test_mse_rf)
```

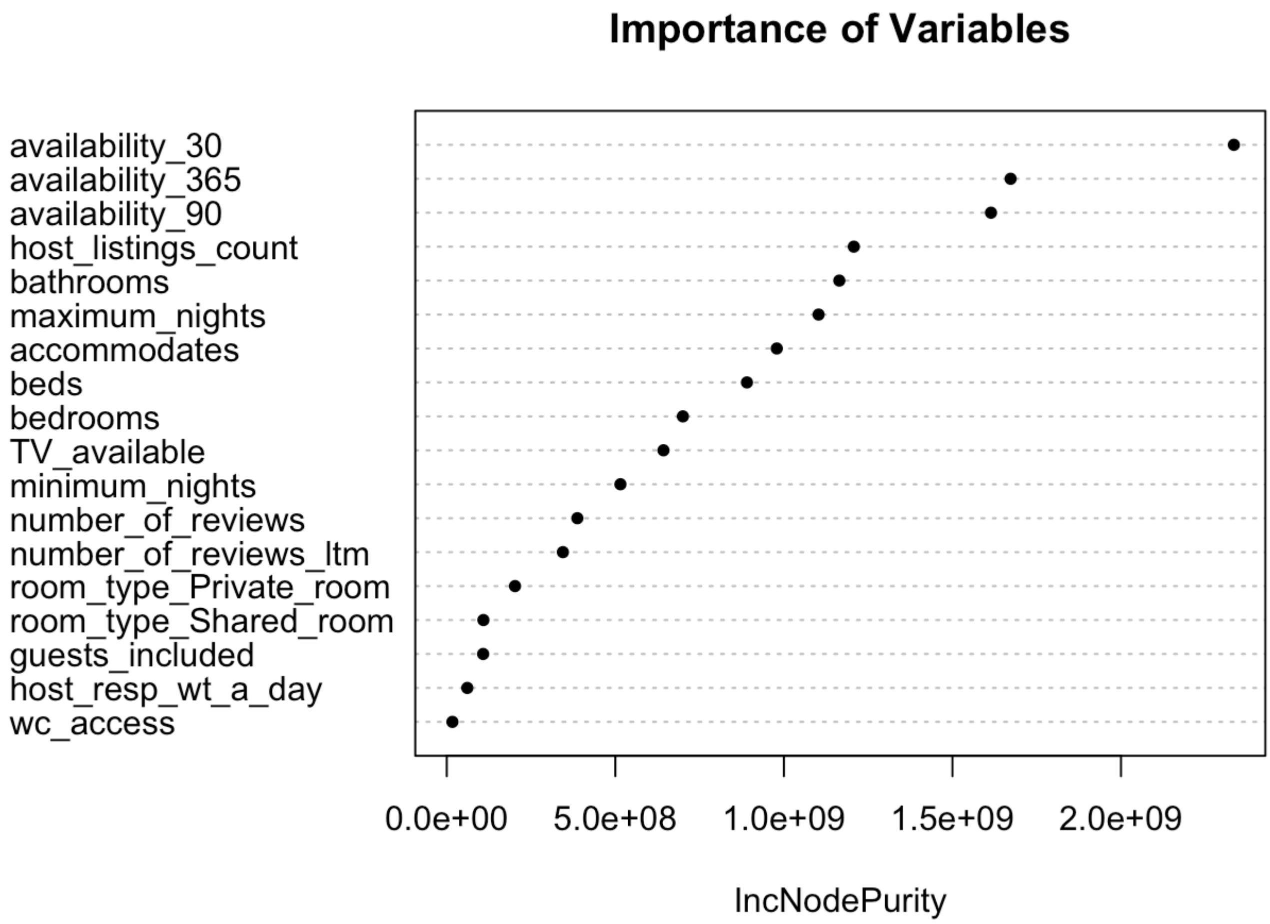
```
## [1] 43083.78
```

Random Forest–Beijing

```
##Random Forest
#decide ntree by the plot of error vs ntree
error_rf <- randomForest(price ~.,data=bjList_train)
plot(error_rf,main = "Error rate of random forest")
```




```
fit_rf <- randomForest(price~.,
                       bjList_train,
                       ntree=,
                       do.trace=F)

varImpPlot(fit_rf,pch = 20, main = "Importance of Variables")
```




```
## MSE of train for Beijing
yhat_rf <- predict(fit_rf, bjList_train)
train_mse_rf <- mean((yhat_rf - bjList_train$price) ^ 2)
print(train_mse_rf)
```

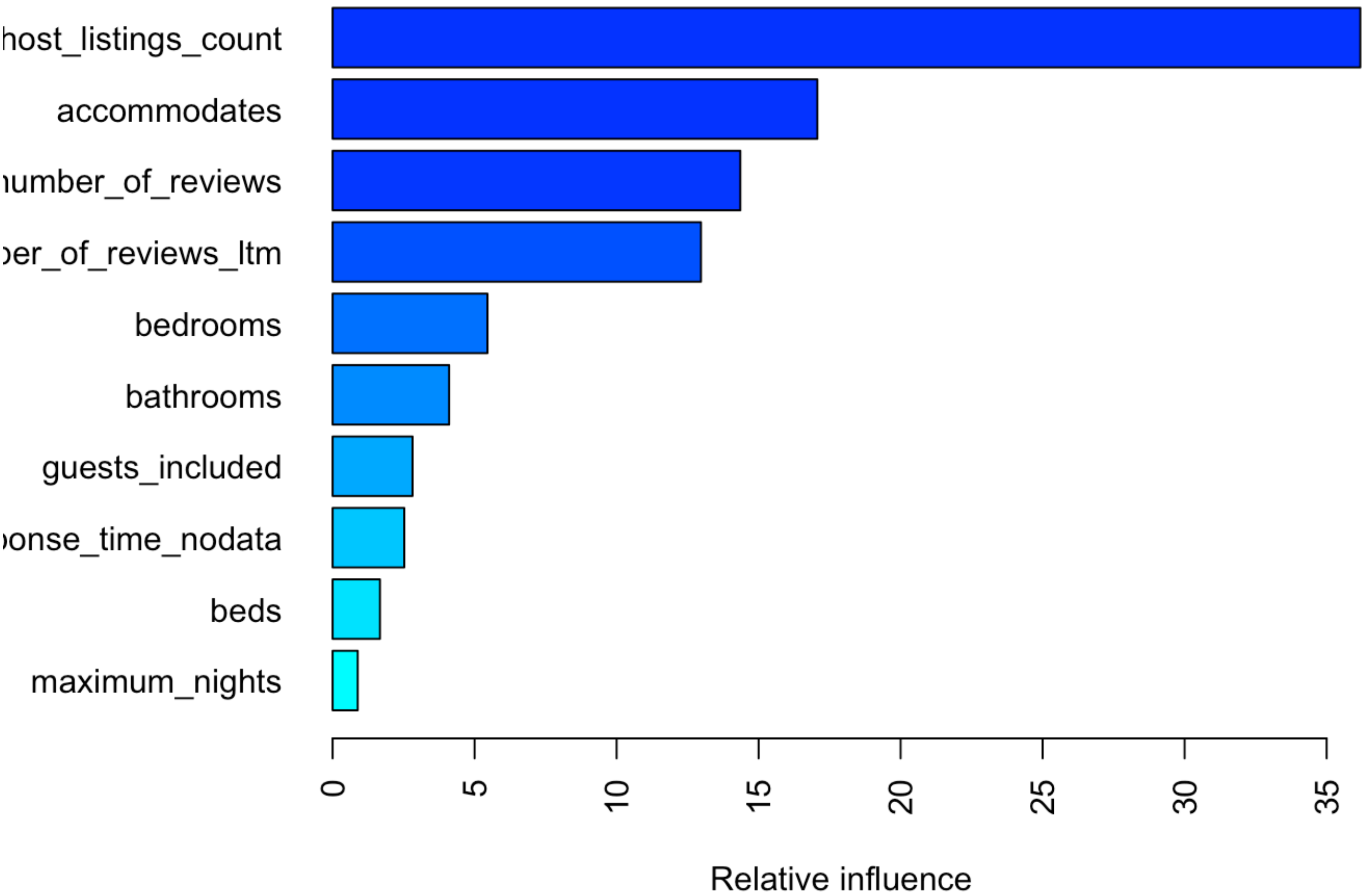
```
## [1] 385519.8
```

```
#levels(boslist_test$neighbourhood_cleansed) = levels(boslist_train$neighbourhood_c
leansed)

## MSE of Test for Beijing
yhat_rf <- predict(fit_rf, bjList_test)
test_mse_rf <- mean((yhat_rf - bjList_test$price) ^ 2)
print(test_mse_rf)
```

```
## [1] 3683702
```

gradient boosting–Boston

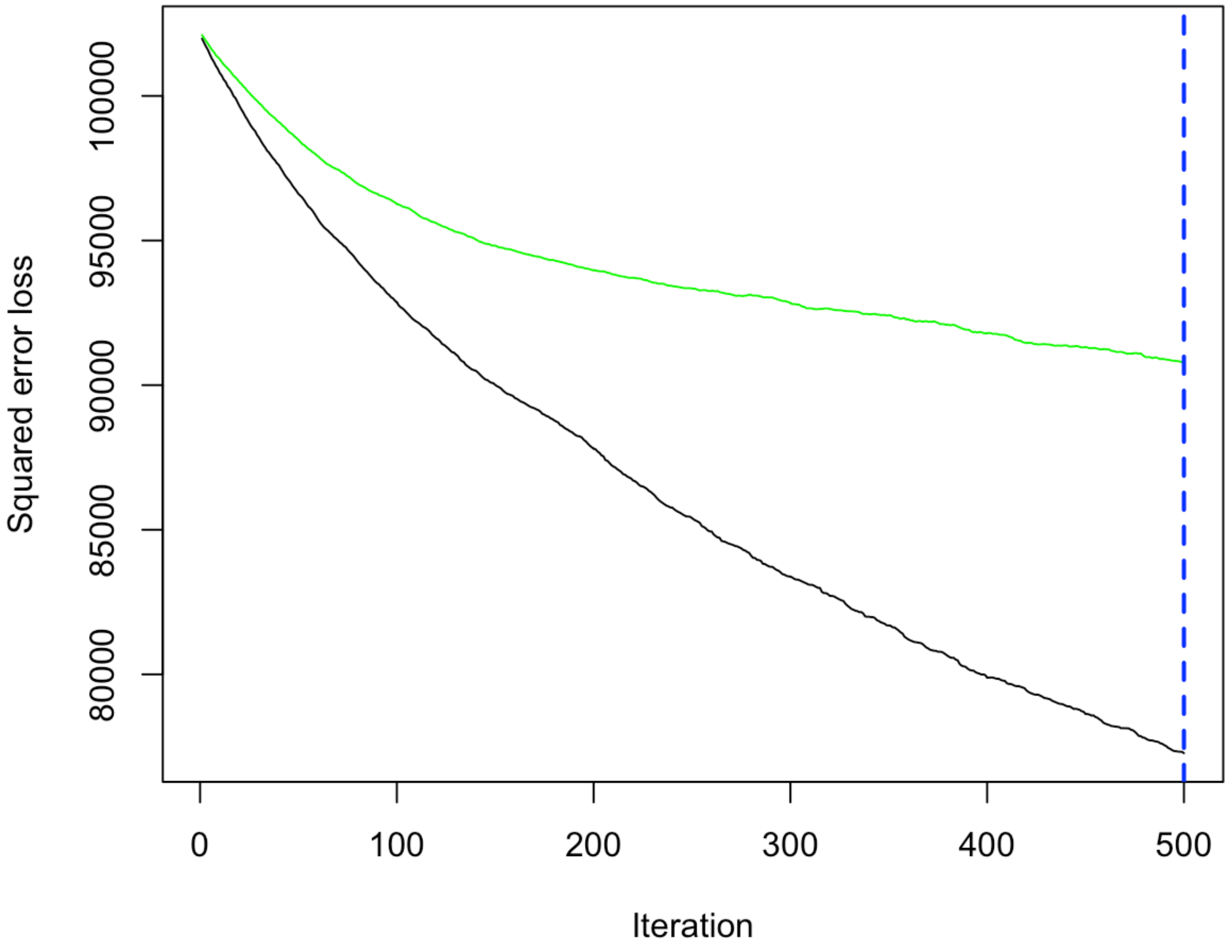
```
Boston.boost=gbm(formula = price~., distribution = "gaussian", data = boslist_train
, n.trees = 500,interaction.depth = 15, shrinkage = 0.005,cv.folds = 5)

# A gradient boosted model with gaussian loss function.
# 10000 iterations were performed.
# There were 13 predictors of which 13 had non-zero influence.
par(mar = c(5, 8, 1, 1))
summary(
  Boston.boost,
  cBars = 10,
  method = relative.influence, # also can use permutation.test.gbm
  las = 2
)
```




```
##
var
## host_listings_count                                                       host_l
istings_count
## accommodates
accommodates
## number_of_reviews                                                           numb
er_of_reviews
```

```
## number_of_reviews_ltm                                                    number_o
f_reviews_ltm
## bedrooms
bedrooms
## bathrooms
bathrooms
## guests_included                                                                gu
ests_included
## host_response_time_nodata                                            host_respons
e_time_nodata
## beds
beds
## maximum_nights                                                                  m
aximum_nights
## neighbourhood_cleansed_South_Boston                            neighbourhood_cleansed
_South_Boston
## neighbourhood_cleansed_Beacon_Hill                             neighbourhood_cleanse
d_Beacon_Hill
## cancellation_policy_strict                                             cancellation_
policy_strict
## neighbourhood_cleansed_Bay_Village                             neighbourhood_cleanse
d_Bay_Village
## neighbourhood_cleansed_South_Boston_Waterfront neighbourhood_cleansed_South_Bost
on_Waterfront
## neighbourhood_cleansed_West_End                                   neighbourhood_clea
nsed_West_End
## neighbourhood_cleansed_Mattapan                                   neighbourhood_clea
nsed_Mattapan
## wifi_available                                                                  w
ifi_available
##                                                      rel.inf
## host_listings_count                             36.186459551
## accommodates                                    17.068875967
## number_of_reviews                               14.355901530
## number_of_reviews_ltm                           12.970606801
## bedrooms                                         5.452900424
## bathrooms                                        4.103573539
## guests_included                                  2.814886718
## host_response_time_nodata                        2.523359946
## beds                                             1.668039106
## maximum_nights                                   0.884321814
## neighbourhood_cleansed_South_Boston              0.843548476
## neighbourhood_cleansed_Beacon_Hill               0.772269237
## cancellation_policy_strict                       0.104151179
## neighbourhood_cleansed_Bay_Village               0.091705097
## neighbourhood_cleansed_South_Boston_Waterfront  0.091672380
## neighbourhood_cleansed_West_End                  0.058995749
## neighbourhood_cleansed_Mattapan                  0.008732485
## wifi_available                                   0.000000000
```

```
perf_gbm1 = gbm.perf(Boston.boost, method = "cv")
```




```
boostpre <- predict(
                    # the model from above
                    object = Boston.boost,
                    # the testing data
                    newdata = boslist_train,
                    # this is the number we calculated above
                    n.trees = perf_gbm1)
rmse_fit <- Metrics::rmse(actual = boslist_train$price,
                           predicted = boostpre)
## MSE of train for Boston
rmse_fit^2
```

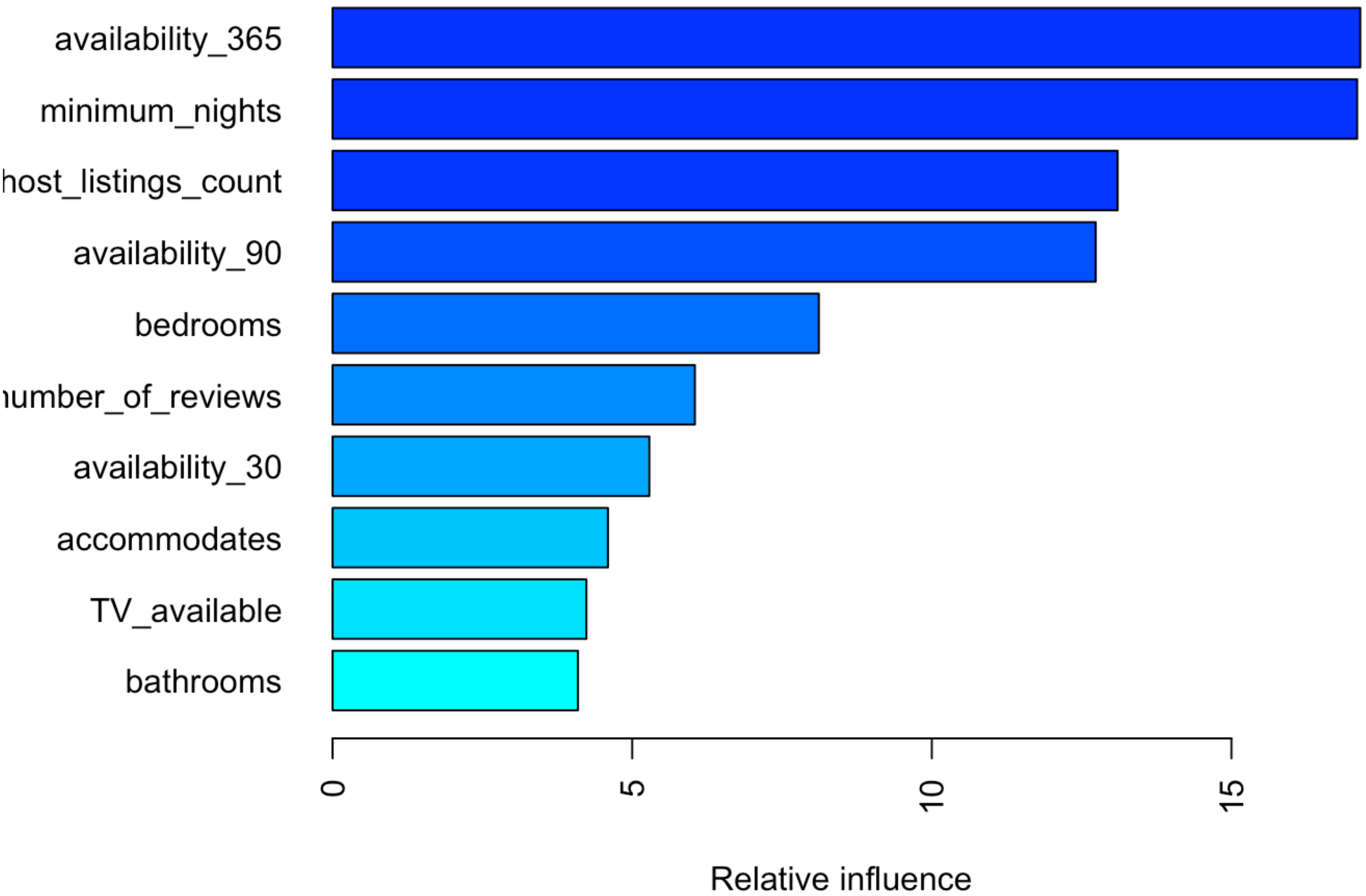
```
## [1] 77275.66
```

```
boostpre <- predict(
  # the model from above
  object = Boston.boost,
  # the testing data
  newdata = boslist_test,
  # this is the number we calculated above
  n.trees = perf_gbm1)
rmse_fit <- Metrics::rmse(actual = boslist_test$price,
                          predicted = boostpre)
## MSE of test for Boston
rmse_fit^2
```

```
## [1] 44983.96
```

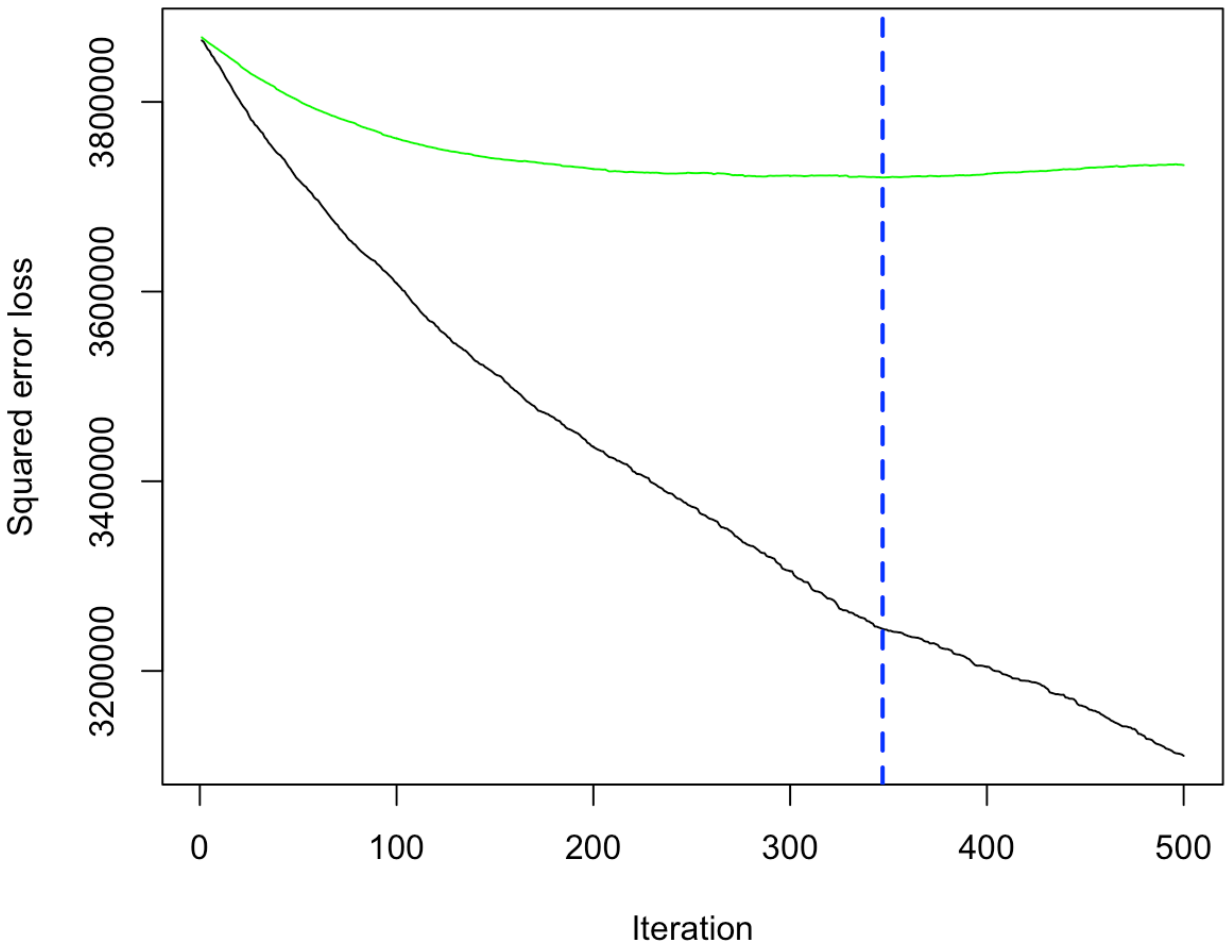
gradient boosting–Beijing

```
beijing.boost=gbm(formula = price~., distribution = "gaussian", data = bjList_test,
n.trees = 500,interaction.depth = 15, shrinkage = 0.005,cv.folds = 5)

# A gradient boosted model with gaussian loss function.
# 10000 iterations were performed.
# There were 13 predictors of which 13 had non-zero influence.
par(mar = c(5, 8, 1, 1))
summary(
  beijing.boost,
  cBars = 10,
  method = relative.influence, # also can use permutation.test.gbm
  las = 2
)
```




```
##                                             var      rel.inf
## availability_365             availability_365 1.715104e+01
## minimum_nights                 minimum_nights 1.709822e+01
## host_listings_count       host_listings_count 1.310128e+01
## availability_90               availability_90 1.273510e+01
## bedrooms                             bedrooms 8.115077e+00
## number_of_reviews           number_of_reviews 6.046532e+00
## availability_30               availability_30 5.286247e+00
## accommodates                     accommodates 4.595104e+00
## TV_available                     TV_available 4.235140e+00
## bathrooms                           bathrooms 4.094413e+00
## maximum_nights                 maximum_nights 2.497031e+00
## room_type_Private_room room_type_Private_room 1.903939e+00
## number_of_reviews_ltm   number_of_reviews_ltm 1.683983e+00
## beds                                     beds 6.814985e-01
## wc_access                           wc_access 3.231867e-01
## room_type_Shared_room   room_type_Shared_room 2.524845e-01
## guests_included               guests_included 1.988538e-01
## host_resp_wt_a_day         host_resp_wt_a_day 8.650184e-04
```

```
perf_gbm1 = gbm.perf(beijing.boost, method = "cv")
```




```
boostpre <- predict(
                    # the model from above
                    object = beijing.boost,
                    # the testing data
                    newdata = bjList_train,
                    # this is the number we calculated above
                    n.trees = perf_gbm1)
rmse_fit <- Metrics::rmse(actual = bjList_train$price,
                          predicted = boostpre)
## MSE of train for Beijing
rmse_fit^2
```

```
## [1] 1448112
```

```
boostpre <- predict(
  # the model from above
  object = beijing.boost,
  # the testing data
  newdata = bjList_test,
  # this is the number we calculated above
  n.trees = perf_gbm1)
rmse_fit <- Metrics::rmse(actual = bjList_test$price,
                          predicted = boostpre)
## MSE of test for Beijing
rmse_fit^2
```

```
## [1] 3244364
```