code_4360412

Ingrid Shu

4/13/2021

```
library(tidyverse)
library(readr)
library(corrplot)
library(caret)
```

Load the training set and test set.

```
train <- read_csv("Train&Test_FINAL_PROJECT/train.csv") %>%
  mutate(desc = factor(desc),
         exteriorfinish = factor(exteriorfinish),
         rooftype = factor(rooftype),
         state = factor(state))


actual_test <- read_csv("Train&Test_FINAL_PROJECT/test.csv") %>%
  mutate(desc = factor(desc),
         exteriorfinish = factor(exteriorfinish),
         rooftype = factor(rooftype),
         state = factor(state))
```

Let's examine missing values.

```
sum_NA <- function(var){
  sum_NA <- sum(is.na(var))
  return(sum_NA)
}

table(is.na(train)) # 687 N/A in total
```

```
##
## FALSE  TRUE
## 23113   687
```

```
sum_NA(train$fireplaces) # has all 687 N/A
```

```
## [1] 687
```

```r
train <- train %>%
  dplyr::select(-c(fireplaces))

actual_test <- actual_test %>%
  dplyr::select(-c(fireplaces))
```

We will omit fireplaces from our study as it has 687 missing values.

We will also omit zipcode as AvgIncome already records the average household income within the zipcode. Additionally, zipcode can be difficult to interpret.

```r
train <- train %>%
  dplyr::select(-c(zipcode))

actual_test <- actual_test %>%
  dplyr::select(-c(zipcode))
```

We can change yearbuilt to be a categorical variable.

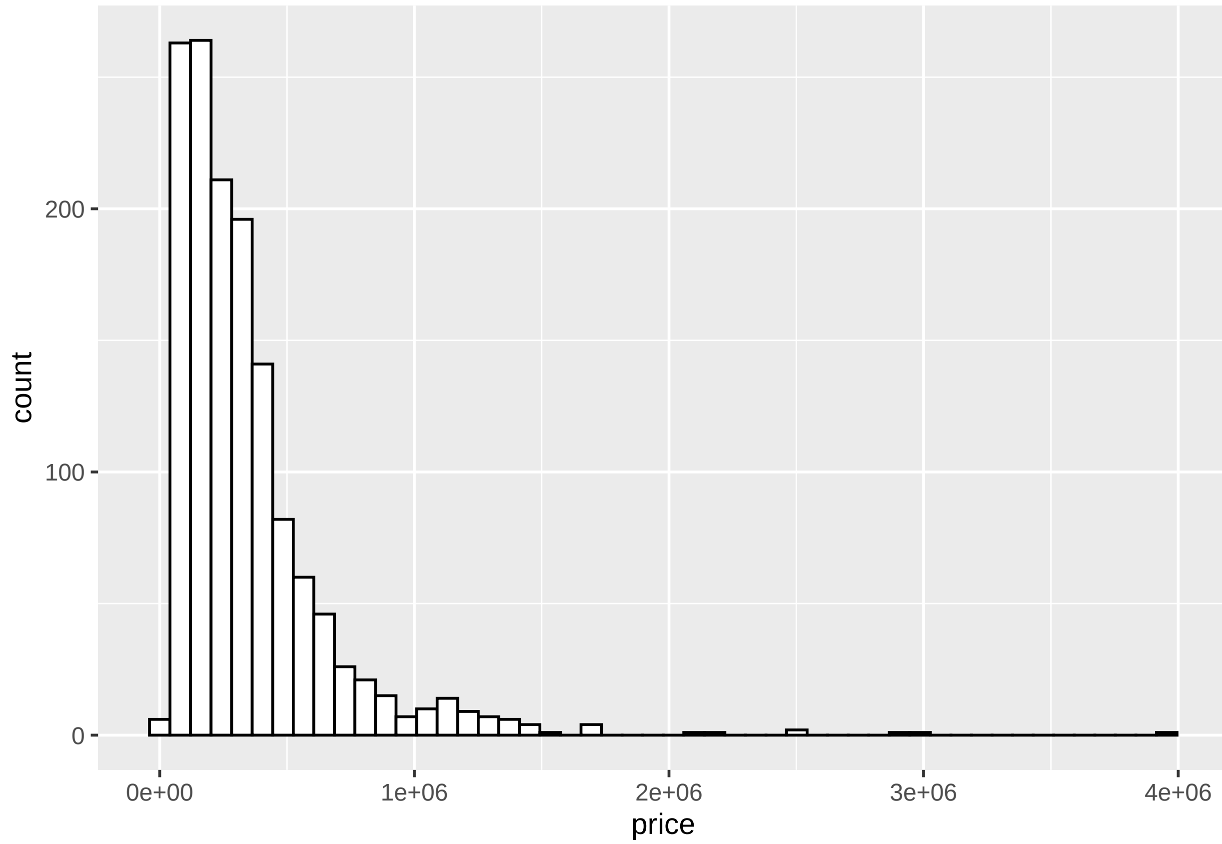
```r
summary(train$yearbuilt)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1805    1922    1940    1944    1960    2017
```

```r
train <- train %>%
  mutate(yearbuilt = ifelse(yearbuilt <= 1940, "old", "new")) %>%
  mutate(yearbuilt = factor(yearbuilt))

actual_test <- actual_test %>%
  mutate(yearbuilt = ifelse(yearbuilt <= 1940, "old", "new")) %>%
  mutate(yearbuilt = factor(yearbuilt))
```

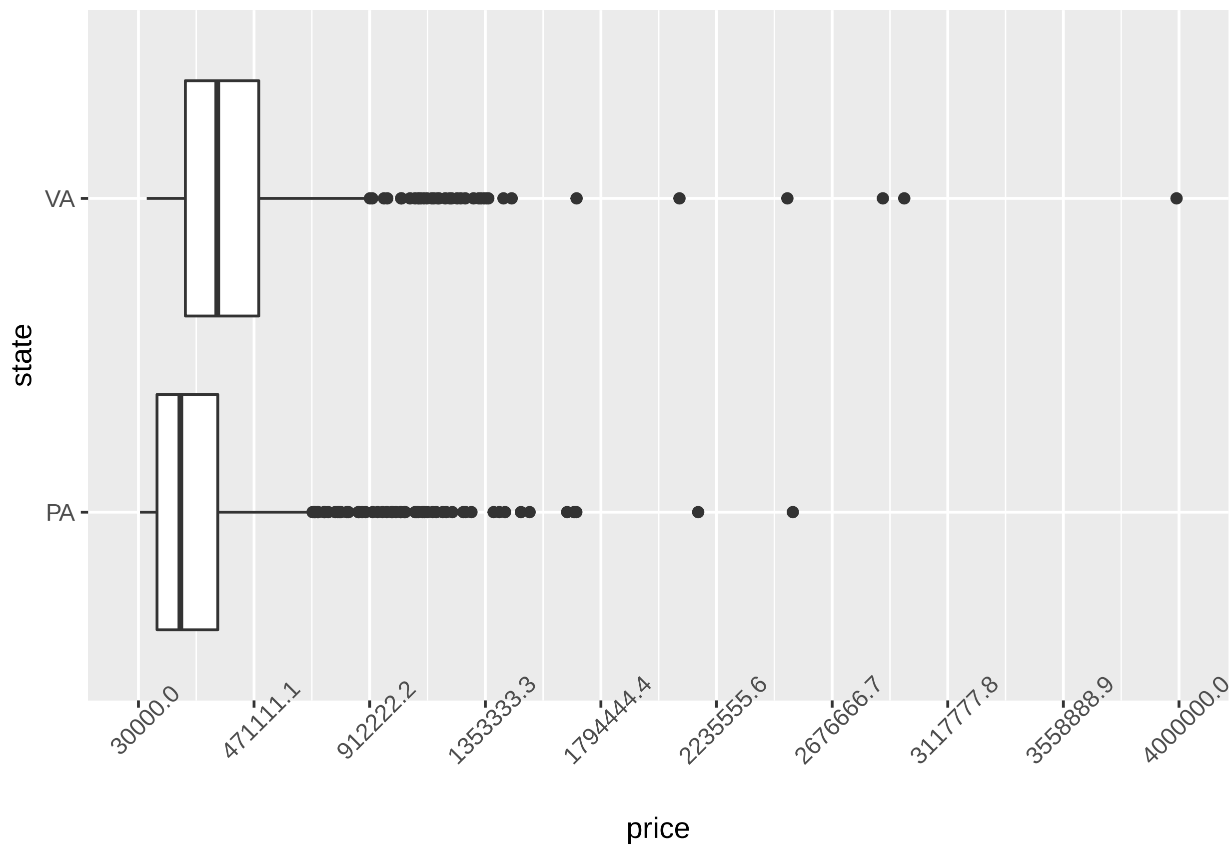
Does our data have any extreme outliers for housing price?

```r
ggplot(train, aes(price)) +
  geom_histogram(color = "black", fill = "white", bins = 50)
```



The price histogram is severely right skewed. Let's examine if there are more outliers for VA or PA.

```
VA_train <- train %>%
  filter(state == "VA")

PA_train <- train %>%
  filter(state == "PA")

ggplot(train, aes(x = state, y = price))+
  geom_boxplot()+
  coord_flip()+
  scale_y_continuous(breaks = seq(3e4, 4e6, length.out = 10))+
  theme(axis.text.x = element_text(angle = 45))
```




```r
summary(VA_train$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   61728  209083  330816  408062  489166 3990701
```

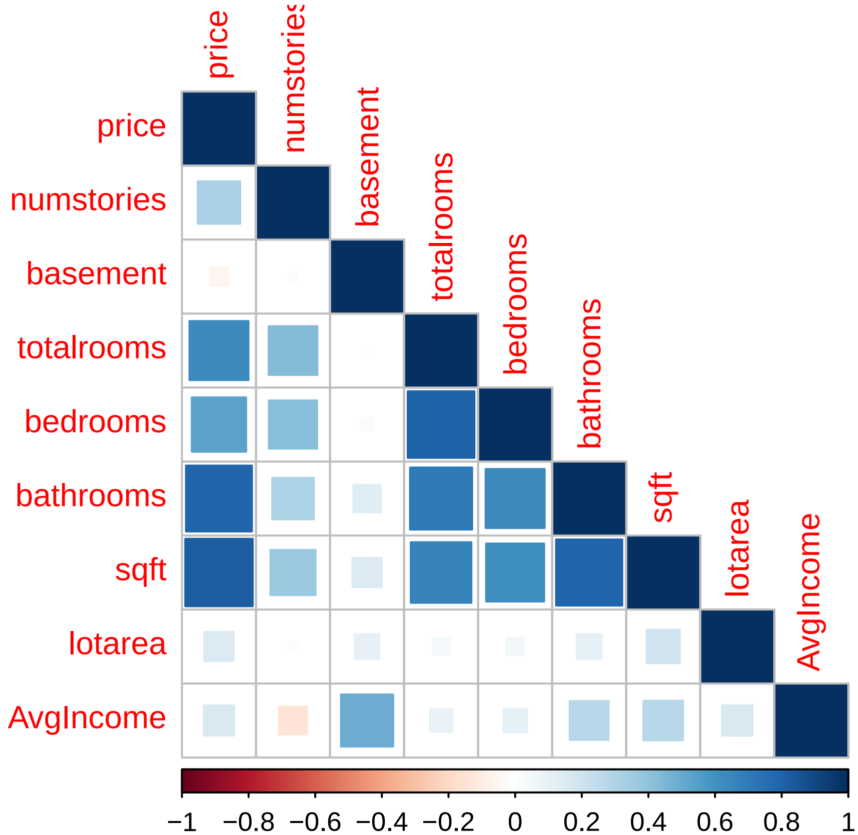
```r
summary(PA_train$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   35847  100935  190048  280591  332769 2526698
```

VA has a slightly higher median price, but both states have their share of ridiculously high housing price outliers.

Let's examine correlations between the continuous variables.

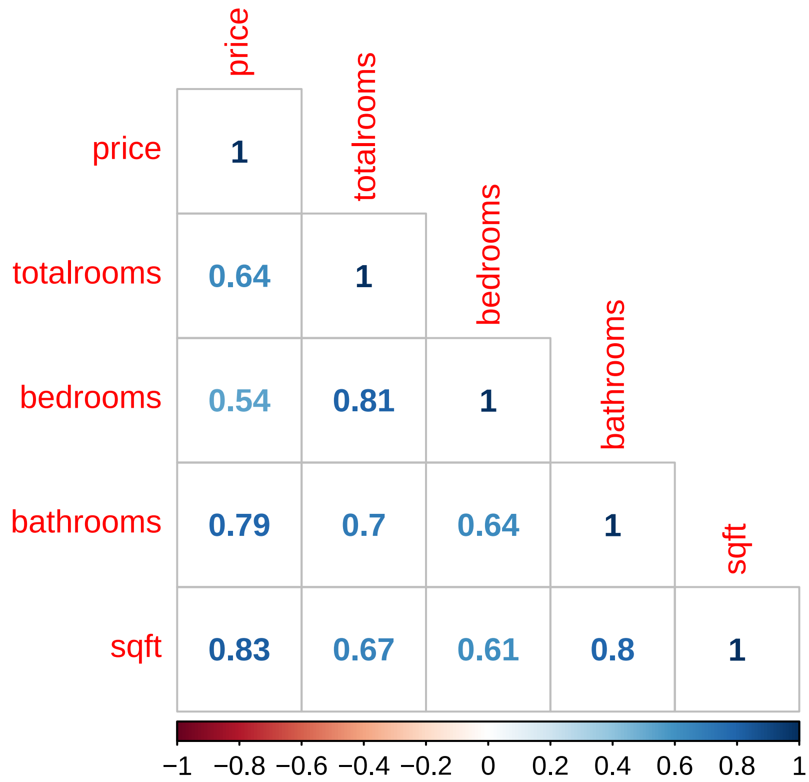
```r
cont_train <- train[-c(1, 3, 5, 6, 7,14)]  # train with only continuous predictors
A <- cor(cont_train)

corrplot::corrplot(A, method = "square", type = "lower")
```



As expected, there is moderately strong positive correlation between totalrooms, bedrooms, bathrooms, and sqft. These variables are also positively correlated with price.

Let's single out totalrooms, bedrooms, bathrooms, and sqft and observe their correlations for possible collinearity.

```
rooms_train <- cont_train[,c(1, 4:7)]
C <- cor(rooms_train)

corrplot::corrplot(C, method = "number", type = "lower")
```



Divide train into a training and testing set. Also, let us identify these outlier observations and create a different version of train with no outliers.

```
set.seed(126)
train1 <- sample_frac(train, 0.7)
test1 <- setdiff(train, train1)


# and a split for the sets with no outliers

outliers <- boxplot.stats(train$price)$out
out_index <- which(train$price %in% c(outliers))

train_no_out <- train[-c(out_index),]

set.seed(3)
train1_NO <- sample_frac(train_no_out, 0.7)
test1_NO <- setdiff(train_no_out, train1_NO)
```

Multiple Regression

```r
fit0 <- lm(price~.-id, data = train1)
summary(fit0)
```

```
##
## Call:
## lm(formula = price ~ . - id, data = train1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -568960  -60815   -3415   56765 1888590
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -2.024e+05  3.806e+04  -5.317 1.31e-07 ***
## descMULTI-FAMILY       -1.277e+05  3.773e+04  -3.386 0.000739 ***
## descROWHOUSE            8.578e+03  5.534e+04   0.155 0.876846
## descSINGLE FAMILY      -2.917e+04  2.357e+04  -1.238 0.216140
## numstories             -2.575e+04  1.167e+04  -2.207 0.027549 *
## yearbuiltold            3.189e+04  1.221e+04   2.611 0.009177 **
## exteriorfinishConcrete  1.791e+04  9.959e+04   0.180 0.857310
## exteriorfinishFrame    -1.562e+04  1.134e+04  -1.378 0.168578
## exteriorfinishLog      -3.974e+04  8.275e+04  -0.480 0.631134
## exteriorfinishStone    -5.690e+04  2.605e+04  -2.184 0.029174 *
## exteriorfinishStucco   -8.848e+04  1.844e+04  -4.797 1.87e-06 ***
## rooftypeROLL            5.852e+03  2.686e+04   0.218 0.827578
## rooftypeSHINGLE        -3.439e+04  1.818e+04  -1.892 0.058747 .
## rooftypeSLATE           4.269e+04  1.468e+04   2.908 0.003723 **
## basement                4.978e+04  1.664e+04   2.992 0.002842 **
## totalrooms             -2.023e+03  4.081e+03  -0.496 0.620213
## bedrooms               -2.040e+04  7.734e+03  -2.638 0.008476 **
## bathrooms               7.365e+04  7.037e+03  10.466  < 2e-16 ***
## sqft                    1.603e+02  6.369e+00  25.175  < 2e-16 ***
## lotarea                 4.799e-02  2.909e-02   1.649 0.099420 .
## stateVA                 2.078e+05  1.941e+04  10.705  < 2e-16 ***
## AvgIncome               1.263e+00  3.915e-01   3.225 0.001301 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 139300 on 958 degrees of freedom
## Multiple R-squared:  0.822,  Adjusted R-squared:  0.8181
## F-statistic: 210.7 on 21 and 958 DF,  p-value: < 2.2e-16
```

From multiple regression including all predictors, we see that significant predictors include desc, numstories, yearbuilt, exteriorfinish, rooftype, basement, bedrooms, bathrooms, sqft, lotarea, state, and AvgIncome.

Let's fit a multiple regression with only these.

```r
fit00 <- lm(price ~ + numstories + yearbuilt + exteriorfinish + rooftype + basement + bedrooms + bathro

summary(fit00)
```

```
##
## Call:
## lm(formula = price ~ +numstories + yearbuilt + exteriorfinish +
##     rooftype + basement + bedrooms + bathrooms + sqft + lotarea +
##     state + AvgIncome, data = train1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -562250  -61494   -3268   58998 1901785
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -2.183e+05  3.558e+04  -6.136 1.23e-09 ***
## numstories             -2.818e+04  1.162e+04  -2.425 0.015495 *
## yearbuiltold            2.632e+04  1.210e+04   2.175 0.029897 *
## exteriorfinishConcrete  2.913e+04  9.964e+04   0.292 0.770104
## exteriorfinishFrame    -1.536e+04  1.137e+04  -1.351 0.176994
## exteriorfinishLog      -7.723e+04  8.230e+04  -0.938 0.348304
## exteriorfinishStone    -5.711e+04  2.613e+04  -2.185 0.029100 *
## exteriorfinishStucco   -8.698e+04  1.847e+04  -4.710 2.83e-06 ***
## rooftypeROLL            9.195e+03  2.643e+04   0.348 0.727936
## rooftypeSHINGLE        -3.673e+04  1.816e+04  -2.023 0.043360 *
## rooftypeSLATE           4.265e+04  1.475e+04   2.892 0.003910 **
## basement                4.599e+04  1.614e+04   2.850 0.004469 **
## bedrooms               -2.751e+04  6.201e+03  -4.437 1.02e-05 ***
## bathrooms               7.114e+04  6.818e+03  10.435  < 2e-16 ***
## sqft                    1.620e+02  6.254e+00  25.904  < 2e-16 ***
## lotarea                 4.799e-02  2.920e-02   1.643 0.100629
## stateVA                 2.096e+05  1.849e+04  11.338  < 2e-16 ***
## AvgIncome               1.393e+00  3.843e-01   3.624 0.000305 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 140100 on 962 degrees of freedom
## Multiple R-squared:  0.8192, Adjusted R-squared:  0.816
## F-statistic: 256.4 on 17 and 962 DF,  p-value: < 2.2e-16
```

```r
pred00 <- predict(fit00, newdata = test1)
(mse_00 <- mean((pred00 - test1$price)^2))
```

```
## [1] 14294379636
```

14.3 billion test MSE for multiple regression.

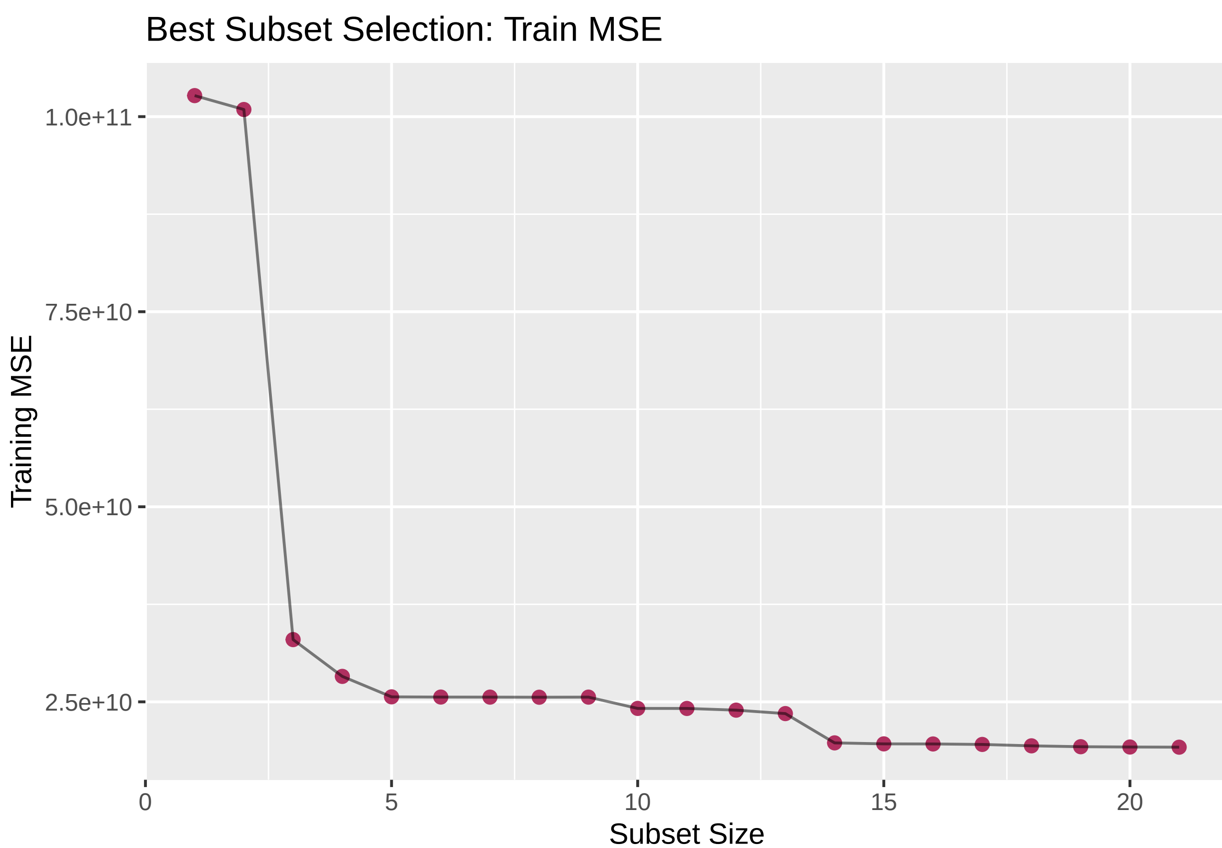
Best Subset Selection

```r
library(leaps)
bestSubset_fit <- regsubsets(price ~., data = train1[,-1], nvmax = 21)
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 1 linear dependencies found
```

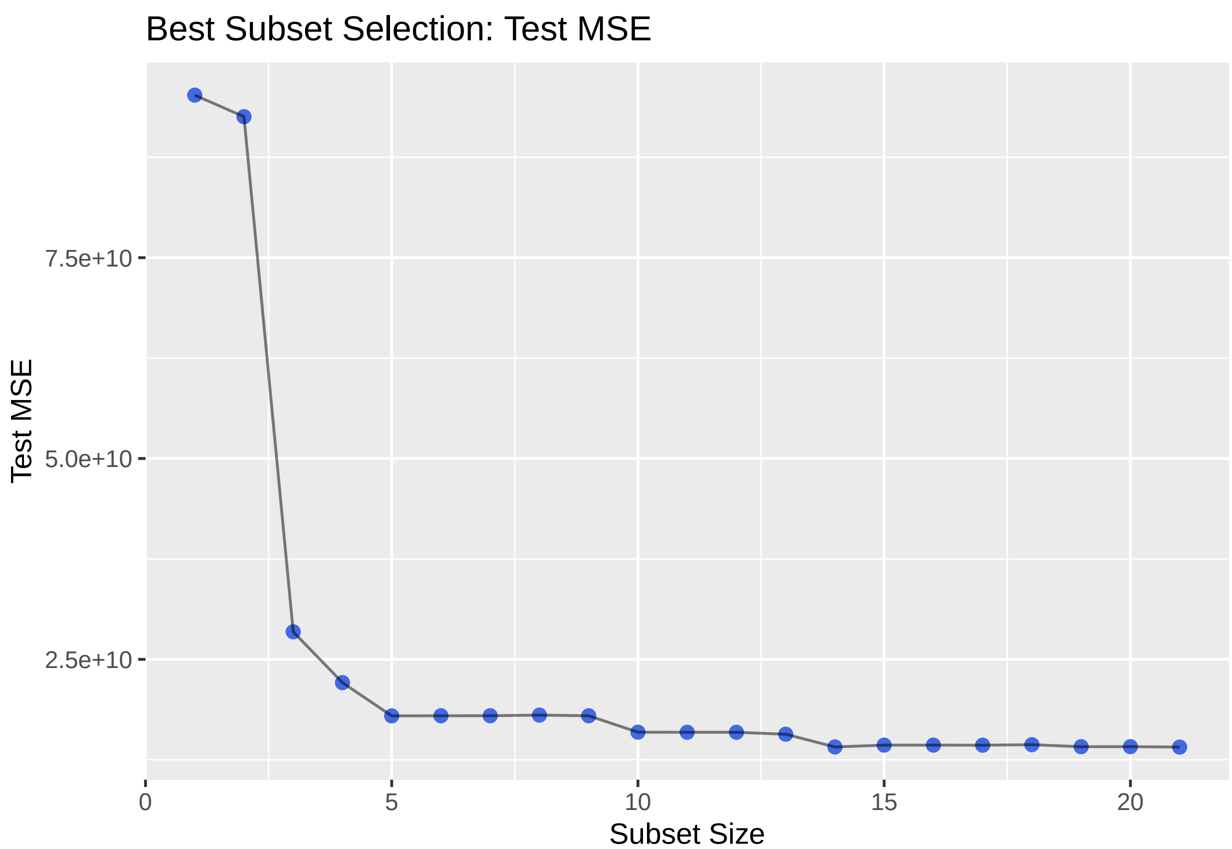
```
## Reordering variables and trying again:
```

```r
train_matrix <- model.matrix(price ~., data = train1[,-1], nvmax = 21)

train_val_errors <- rep(0, 21)
for(i in 1:21){
  coefi <- coef(bestSubset_fit, id = i)
  pred <- train_matrix[, names(coefi)] %*% coefi
  train_val_errors[i] <- mean( (pred - train1$price)^2 )
}

train_val_errors <-  data.frame(train_val_errors)

ggplot(train_val_errors, aes(x = c(1:21), y = train_val_errors))+
  geom_point(color = "maroon", size = 2)+
  geom_line(alpha = 0.5)+
  labs(title = "Best Subset Selection: Train MSE", x = "Subset Size", y = "Training MSE")
```




```r
test_matrix <- model.matrix(price ~., data = test1[,-1], nvmax = 21)

test_val_errors <- rep(0, 21)
for (i in 1:21){
  coefi <- coef(bestSubset_fit, id = i)
  pred <- test_matrix[, names(coefi)] %*% coefi
  test_val_errors[i] <- mean( (pred - test1$price)^2 )
```

```
}

test_val_errors <- data.frame(test_val_errors)

ggplot(test_val_errors, aes(x = c(1:21), y = test_val_errors))+
  geom_point(color = "royal blue", size = 2)+
  geom_line(alpha = 0.5)+
  labs(title = "Best Subset Selection: Test MSE", x = "Subset Size", y = "Test MSE")
```




```
which.min(test_val_errors$test_val_errors)
```

```
## [1] 21
```

```
coef(bestSubset_fit, id = 21)
```

```
##             (Intercept)            descROWHOUSE      descSINGLE FAMILY
##           -2.252649e+05            5.690298e+04           1.966492e+04
##              numstories            yearbuiltold exteriorfinishConcrete
##           -2.817349e+04            2.665452e+04           4.029184e+04
##     exteriorfinishFrame       exteriorfinishLog    exteriorfinishStone
##           -1.618570e+04           -7.416485e+04          -5.955657e+04
##    exteriorfinishStucco            rooftypeROLL        rooftypeSHINGLE
##           -8.754801e+04            1.175382e+04          -3.799658e+04
##           rooftypeSLATE                basement             totalrooms
```

```
##           4.273277e+04           4.587916e+04          -4.970331e+03
##               bedrooms              bathrooms                   sqft
##          -2.178430e+04           7.429146e+04           1.627865e+02
##                lotarea                stateVA              AvgIncome
##           4.571217e-02           2.099513e+05           1.349230e+00
##        descMOBILE HOME
##           0.000000e+00
```

```r
test_val_errors[21,]
```

```
## [1] 14075609601
```

14.1 billion test MSE. Best subset selection did not narrow down the predictors at all.

Ridge Regression

```r
library(glmnet)

train_matrix <- model.matrix(price ~., data = train1[,-1])
test_matrix <- model.matrix(price~., data = test1[,-1])
grid <- 10^seq(10, -2, length = 100)

set.seed(51)

ridge <- glmnet(x = train_matrix, y = train1$price, alpha = 0, lambda = grid)

cv_ridge <- cv.glmnet(x = train_matrix, y = train1$price, alpha = 0, lambda = grid, nfolds = 10, thresh

best_lambda_ridge <- cv_ridge$lambda.min

ridge_pred <- predict(ridge, s = best_lambda_ridge, newx = test_matrix)

(mseRR <- mean( (ridge_pred - test1$price)^2 ))
```

```
## [1] 13513022883
```

13.5 billion test MSE for ridge regression.

Lasso

```r
set.seed(4)

lasso <- glmnet(x = train_matrix, y = train1$price, alpha = 1, lambda = grid)

cv_lasso <- cv.glmnet(x = train_matrix,  y = train1$price, alpha = 1, lambda = grid, nfolds = 10, thres

best_lambda_lasso <- cv_lasso$lambda.min
```

```r
lasso_pred <- predict(lasso, s = best_lambda_lasso, newx = test_matrix)

(lasso_mse <- mean( (lasso_pred - test1$price)^2))
```

```
## [1] 13474099764
```

```r
predict(lasso, s = best_lambda_lasso, type = "coefficients")
```

```
## 24 x 1 sparse Matrix of class "dgCMatrix"
##                                   1
## (Intercept)             -2.006925e+05
## (Intercept)              .
## descMOBILE HOME          .
## descMULTI-FAMILY        -1.116231e+05
## descROWHOUSE             6.828148e+03
## descSINGLE FAMILY       -1.243774e+04
## numstories              -1.848789e+04
## yearbuiltold             2.254833e+04
## exteriorfinishConcrete   .
## exteriorfinishFrame     -1.196143e+04
## exteriorfinishLog        .
## exteriorfinishStone     -4.069537e+04
## exteriorfinishStucco    -8.170566e+04
## rooftypeROLL             .
## rooftypeSHINGLE         -3.675823e+04
## rooftypeSLATE            3.884479e+04
## basement                 2.915399e+04
## totalrooms               .
## bedrooms                -1.886692e+04
## bathrooms                7.330420e+04
## sqft                     1.558596e+02
## lotarea                  4.110070e-02
## stateVA                  1.833604e+05
## AvgIncome                1.026142e+00
```

13.5 billion test MSE for lasso.

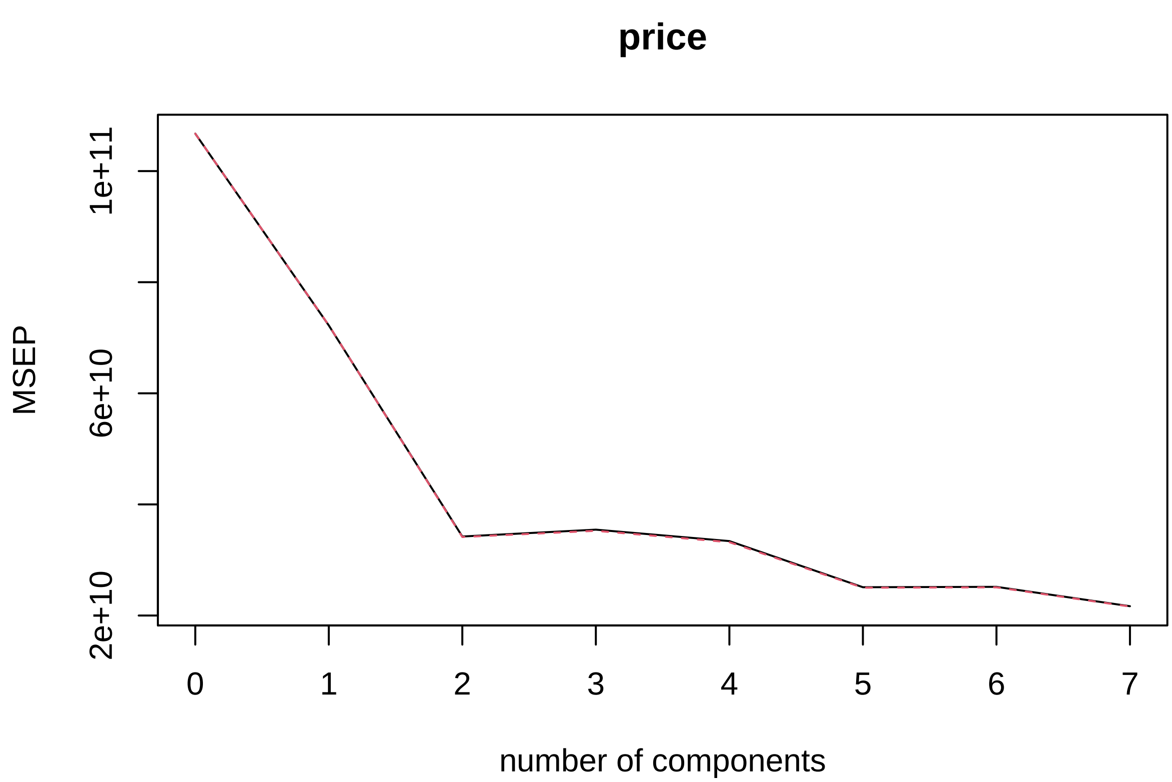
PCA

```r
set.seed(8)
library(pls)
pcr_fit <- pcr(price ~ state + basement + bedrooms + bathrooms + sqft + lotarea + AvgIncome, data = tra
summary(pcr_fit)
```

```
## Data:    X dimension: 980 7
##  Y dimension: 980 1
## Fit method: svdpc
## Number of components considered: 7
##
```

```
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          326712   268737   184995   188298   182743   158397   158617
## adjCV       326712   268788   184744   187751   182200   158127   158346
##        7 comps
## CV      147227
## adjCV   147031
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X        39.39    68.26    81.70    89.02    94.54    97.80   100.00
## price    33.63    69.43    69.47    71.14    77.65    77.65    80.64
```

```
validationplot(pcr_fit, val.type = "MSEP")
```




```
pcr_pred <- predict(pcr_fit, newdata = test1, ncomp = 5)
(pcr_mse <- mean (( pcr_pred - test1$price)^2 ))
```

```
## [1] 15353856315
```

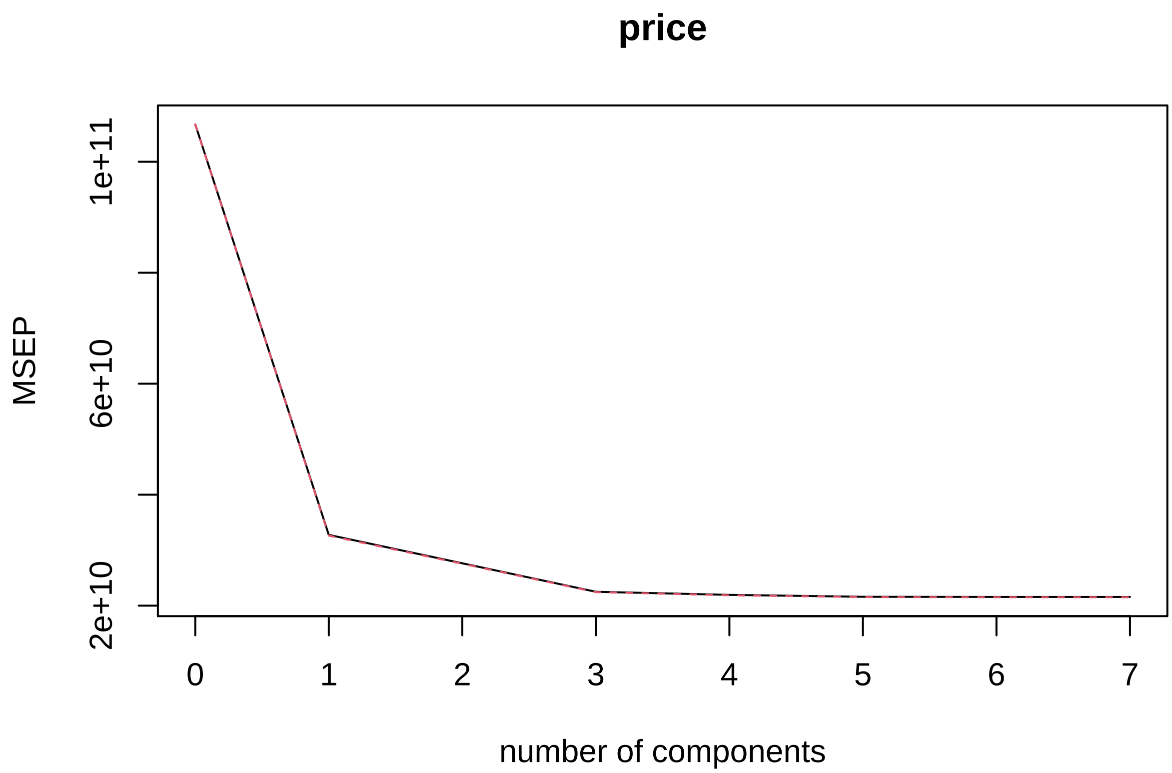
15.4 billion test MSE for PCA.

PLS

```r
set.seed(24)
pls_fit <- plsr(price ~ state + basement + bedrooms + bathrooms + sqft + lotarea + AvgIncome, data = tra
summary(pls_fit)
```

```
## Data:    X dimension: 980 7
##  Y dimension: 980 1
## Fit method: kernelpls
## Number of components considered: 7
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          326712   180993   166240   150003   148131   146961   146866
## adjCV       326712   180621   165985   149813   147890   146799   146696
##        7 comps
## CV      146858
## adjCV   146689
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X        35.45    65.31    73.98    80.09    89.44    96.73   100.00
## price    70.75    75.17    79.75    80.42    80.60    80.64    80.64
```

```r
validationplot(pls_fit, val.type = "MSEP")
```




```r
pls_pred <- predict(pls_fit, newdata= test1[,-1], ncomp = 5)
(pls_mse <- mean( (pls_pred - test1$price)^2 ))
```

```
## [1] 15370635778
```

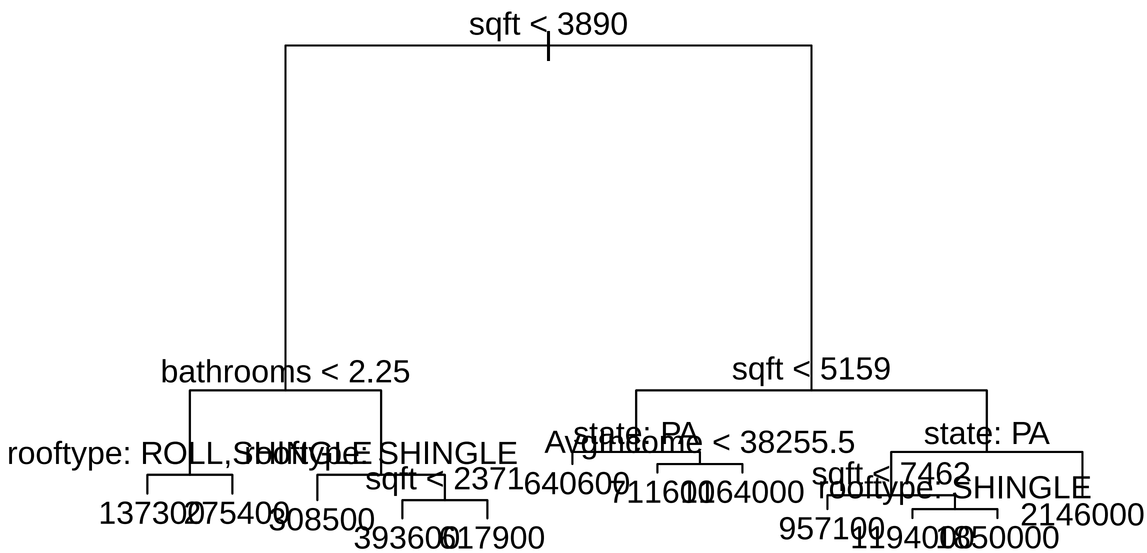
15.4 billion test MSE for PLS.

Regression Tree

```r
library(tree)
library(randomForest)
tree0 <- tree(price ~.-id, data = train1)
summary(tree0)
```

```
##
## Regression tree:
## tree(formula = price ~ . - id, data = train1)
## Variables actually used in tree construction:
## [1] "sqft"      "bathrooms" "rooftype"  "state"     "AvgIncome"
## Number of terminal nodes:  12
## Residual mean deviance:  2.123e+10 = 2.055e+13 / 968
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -921900  -70330  -11050       0   62970 1844000
```

```r
plot(tree0)
text(tree0, pretty = 0)
```



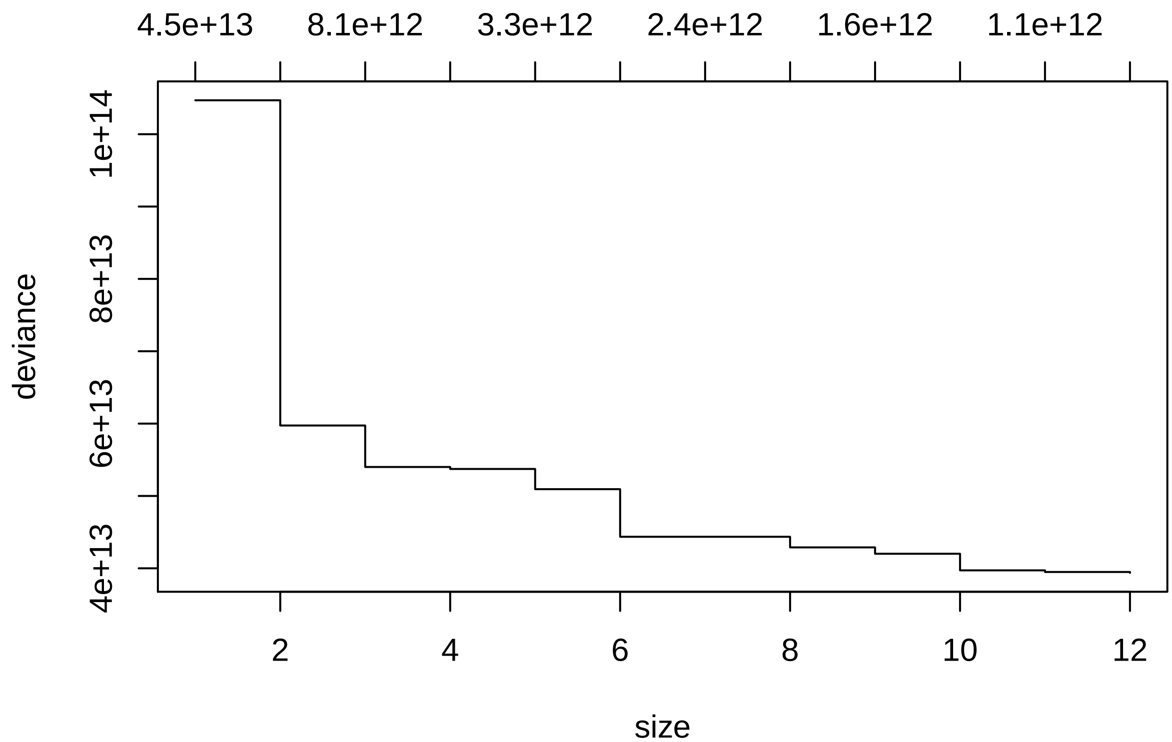

```r
pred <- predict(tree0, newdata = test1)

mse_tree <- mean((pred - test1$price)^2 )
mse_tree
```

```
## [1] 23960379105
```

24.0 billion test MSE for a regression tree.

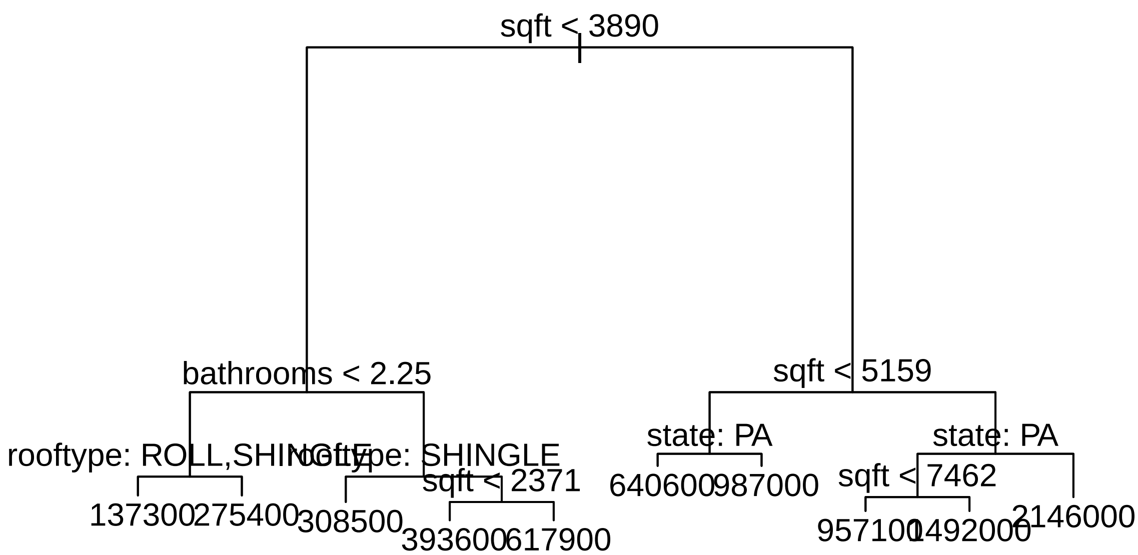
Pruned regression tree

```r
set.seed(15)
cv_tree0 <- cv.tree(tree0)
plot(cv_tree0)
```



Cross-validation has selected a tree of size 10.

```
prune_tree0 <- prune.tree(tree0, best = 10)
plot(prune_tree0)
text(prune_tree0, pretty = 0)
```




```r
prune_pred <- predict(prune_tree0, newdata = test1)
mse_prune <- mean((prune_pred - test1$price)^2)
mse_prune
```

```
## [1] 22030295509
```

22.0 billion test MSE for a pruned regression tree. Pruning the tree did improve the test MSE.

Bagging

```r
set.seed(1)
bag_tree0 <- randomForest(price ~.-id, data = train1, mtry = 13, ntree = 500, importance = TRUE)
bag_pred <- predict(bag_tree0, newdata = test1)
mse_bagg <- mean((bag_pred - test1$price)^2)
mse_bagg
```

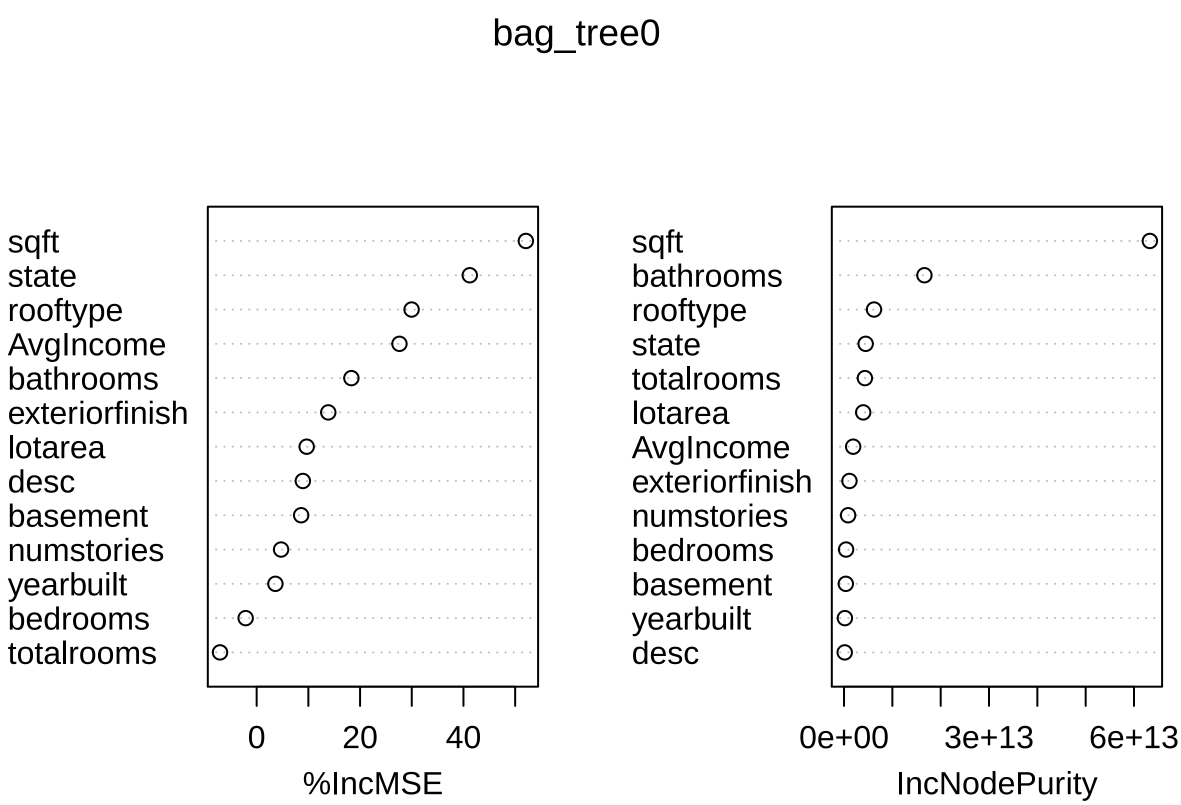
```
## [1] 8230509836
```

```r
importance(bag_tree0)
```

```
##                    %IncMSE IncNodePurity
```

```
## desc             8.935890  1.425121e+11
## numstories       4.724863  8.429528e+11
## yearbuilt        3.624978  1.826971e+11
## exteriorfinish 13.854770  1.124424e+12
## rooftype        29.967479  6.207184e+12
## basement         8.606347  3.654349e+11
## totalrooms      -7.084873  4.317312e+12
## bedrooms        -2.128140  4.231098e+11
## bathrooms       18.318051  1.663663e+13
## sqft            52.069314  6.328286e+13
## lotarea          9.677375  3.967403e+12
## state           41.219300  4.477730e+12
## AvgIncome       27.619165  1.888225e+12
```

```
varImpPlot(bag_tree0)
```



8.2 billion test MSE for bagging.

Boosting

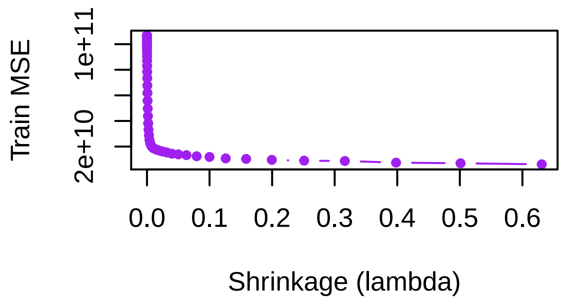
```
library(gbm)
set.seed(2)

powers <- seq(-10, -0.2, by = 0.1)
```

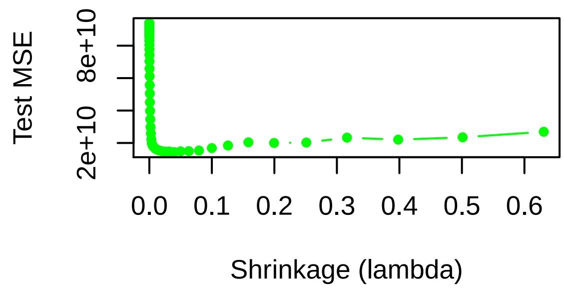
```r
lambdas <-  10^powers
length_lambdas <- length(lambdas)

train_errors <- rep(0, length_lambdas)
test_errors <- rep(0, length_lambdas)

for(i in 1:length_lambdas){
  boost = gbm(price ~.-id, data = train1, distribution = "gaussian", n.trees = 1000, shrinkage = lambdas
  train_pred = predict(boost, newdata = train1, n.trees = 1000)
  test_pred = predict(boost, newdata = test1, n.trees = 1000)
  train_errors[i] = mean((train_pred - train1$price)^2)
  test_errors[i] = mean((test_pred - test1$price)^2)
}


par(mfrow = c(2,2))
plot(lambdas, train_errors, type = "b", xlab = "Shrinkage (lambda)", ylab = "Train MSE", col = "purple"
plot(lambdas, test_errors, type = "b", xlab = "Shrinkage (lambda)", ylab = "Test MSE", col = "green", p

lambdas[which.min(test_errors)]
```

```
## [1] 0.03981072
```

```r
boost_mse <- min(test_errors)
boost_mse
```

```
## [1] 14390722934
```





14.4 billion test MSE (minimum test MSE) for boosting when lambda = 0.04.

RandomForests

```
set.seed(0)
rf_tree0 <- randomForest(price ~.-id, data = train1, mtry = 4, ntree = 500, importance = TRUE)
rf_pred <- predict(rf_tree0, newdata = test1)

mse_rf <- mean((rf_pred - test1$price)^2)
mse_rf
```

```
## [1] 6246182391
```

```
importance(rf_tree0)
```

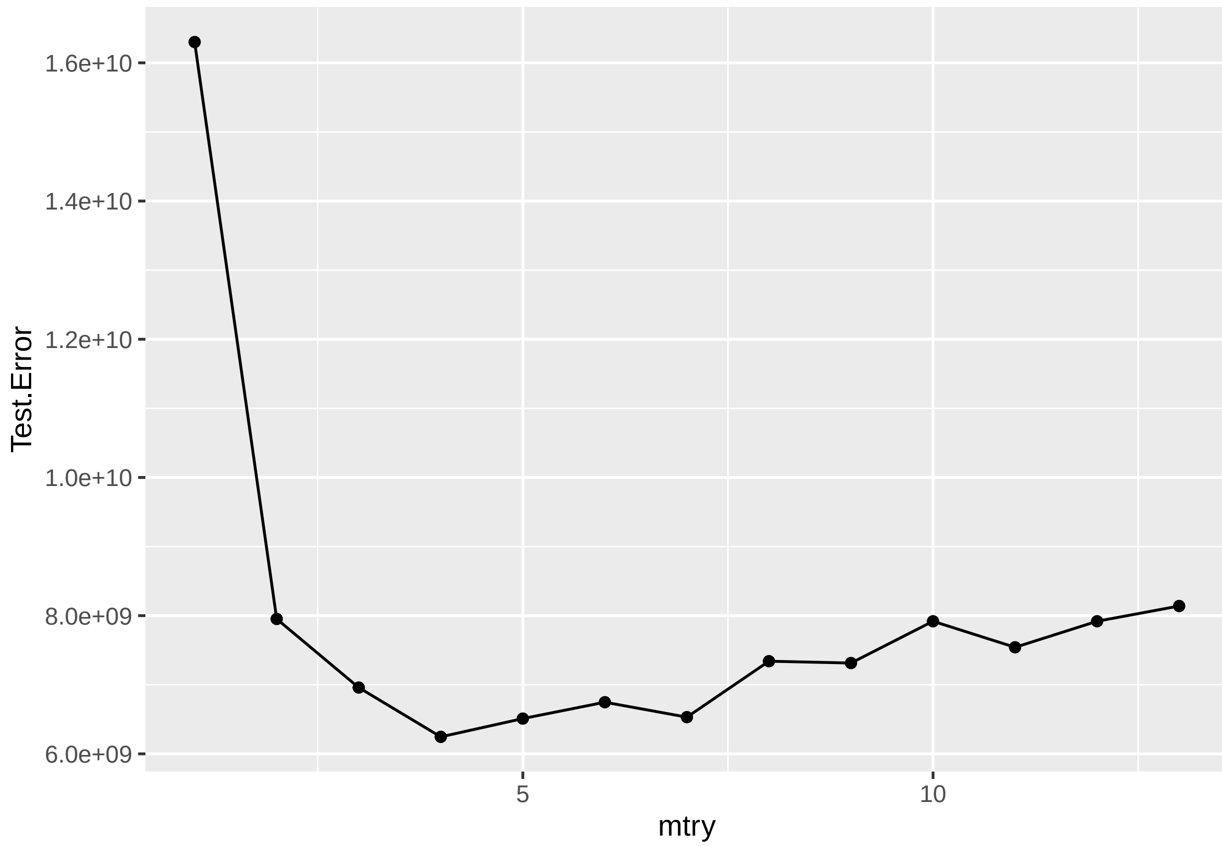
```
##                    %IncMSE IncNodePurity
## desc             6.6615744  2.573388e+11
## numstories       3.9073348  1.982209e+12
## yearbuilt        7.9436598  5.052209e+11
## exteriorfinish   7.2976858  1.542806e+12
## rooftype        23.3003911  5.899239e+12
## basement         9.5005876  9.038959e+11
## totalrooms       8.7603451  1.108992e+13
```

```
## bedrooms        0.1666041  7.116147e+12
## bathrooms      23.2217753  2.420166e+13
## sqft           32.3762802  3.380392e+13
## lotarea        12.1878489  8.596187e+12
## state          26.1032829  3.364472e+12
## AvgIncome      16.7482334  3.526986e+12
```

```r
varImpPlot(rf_tree0)
```

6.2 billion test MSE for random forest with mtry = 4.

```r
testError <- rep(0, 13)
for(i in 1:13){
  set.seed(0)
  rf <- randomForest(price ~.-id, data = train1, mtry = i, ntree = 500, importance = TRUE)
  pred <- predict(rf, newdata = test1)
  testError[i] <- mean((pred - test1$price)^2)
}

df <- data.frame(
  "mtry" = c(1:13),
  "Test Error" = testError
)

ggplot(df, aes(mtry, Test.Error)) +
  geom_point() +
  geom_line()
```




```
which.min(df$Test.Error)
```

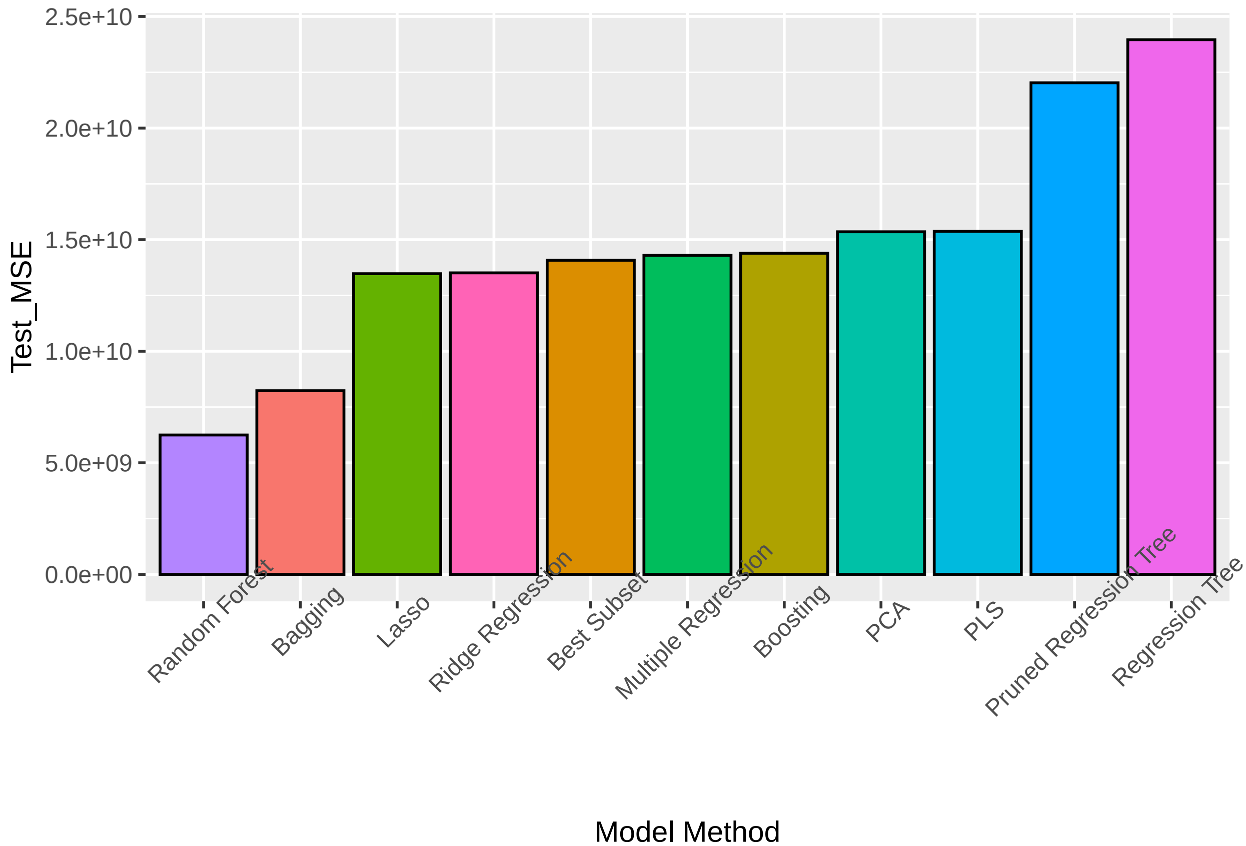
```
## [1] 4
```

The random forest that we tried (mtry = 4) is already the one with the lowest test MSE for random forests.

```
mse_df <- data.frame(
  "Method" = c("Multiple Regression",
               "Best Subset",
               "Ridge Regression",
               "Lasso",
               "PCA",
               "PLS",
               "Regression Tree",
               "Pruned Regression Tree",
               "Bagging",
               "Boosting",
               "Random Forest"
               ),
  "Test_MSE" = c(mse_00, test_val_errors[21,], mseRR, lasso_mse, pcr_mse, pls_mse, mse_tree, mse_prune,
)

mse_df
```

```
##                     Method    Test_MSE
## 1     Multiple Regression 14294379636
## 2             Best Subset 14075609601
## 3        Ridge Regression 13513022883
## 4                   Lasso 13474099764
## 5                     PCA 15353856315
## 6                     PLS 15370635778
## 7         Regression Tree 23960379105
## 8  Pruned Regression Tree 22030295509
## 9                 Bagging  8230509836
## 10               Boosting 14390722934
## 11          Random Forest  6246182391
```

```r
ggplot(mse_df, aes(x = reorder(Method, Test_MSE), y = Test_MSE)) +
  geom_col(aes(fill = Method), color = "black")+
  theme(axis.text.x = element_text(angle = 45))+
  theme(legend.position = "none") +
  labs(x = "Model Method", "Test MSE")
```




```r
rf_no_out <- randomForest(price ~.-id, data = train1_NO, mtry = 4, ntree = 500, importance = TRUE)
rf_no_out_pred <- predict(rf_no_out, newdata = test1_NO)

mean((rf_no_out_pred - test1_NO$price)^2)
```

```
## [1] 3999167411
```

When using data with no outliers, the test MSE for random forest with mtry = 4 is 4.0 billion.

Predicting on test dataset

```
actual_test$price <- predict(rf_tree0, newdata = actual_test[,-2])
actual_test$student_id <- rep(4360412, 600)


testing_predictions_4360412 <- actual_test %>%
  select(id, price, student_id)

write.csv(testing_predictions_4360412, 'testing_predictions_4360412.csv')
```

Baseline comparison

```
avgPrice <- mean(train1$price)
baseMSE <-  mean((avgPrice - test1$price)^2)
baseMSE
```

```
## [1] 93780086525
```

93.8 billion