Prediction of Housing Prices

Mohomad Sahil, Abhishek Shankarnarayanan

# Introduction

#### Propose a question that you will explore for the final project. What makes it interesting? Difficult?

The question proposed for the final project will be to predict the house prices in the king county and see its relationship with various given predictors.

It is interesting to know which features of the house are more effective on the prices of the house so that we can give more importance on such features while advertising.

There are different regression methods and each method has its own pros and cons. Simple regression methods like linear regression, ridge regression, recursive partioning take significantly less computational time but with the trade-off of accuracy while complex methods like random forest, Multiple Adaptive Regressive Spline(MARS) give more accurate predictions but consume very large computational time. Hence, it’s difficult to pick our best model. We need to be very careful while deciding on our final model. After deciding the final model, we need to see if we can reduce any predictor variables without causing any accuracy changes.

#### Discuss what kind of problem this question is. Who would be the (hypothetical) recipient of this work? Why is the problem important? What decisions would be made by the recipients who use the results.

This problem is of regression type. Zillow will be the recipient of this work. They can use our model to predict the prices of houses before they put them on sale, after taking into consideration all the different factors that might affect it. The main decision would be to come up with a fair price for the household, such that it may attract customers.

# CoNVO Statement

### Context:

The Data Science and Analytics team at Zillow, which is an online real estate database company, will analyze this data from the Kings County housing society to predict house prices for future houses to display them on the web portal.

### Need:

Zillow needs to estimate the correct price of houses such that they don’t lose out on potential buyers by overpricing it.

### Vision:

We hope to analyze the data to achieve some kind of relationship between the prices of houses with the various factors that affect it. The decision that can result from these outcomes is to find the correct price for a house with a given set of attributes.

### Outcome:

The results of this data analysis will be used by Zillow to display the correct prices for new houses. Success will be measured by increase in number of buyers in Kings County from the past years.

kc\_house\_data <- read.csv("~/Downloads/kc\_house\_data.csv", header=T)  
library(lubridate)

##   
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':  
##   
## date

library(rpart)  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(ggplot2)  
library(GGally)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:GGally':  
##   
## nasa

## The following objects are masked from 'package:lubridate':  
##   
## intersect, setdiff, union

## The following objects are masked from 'package:stats':  
##   
## filter, lag

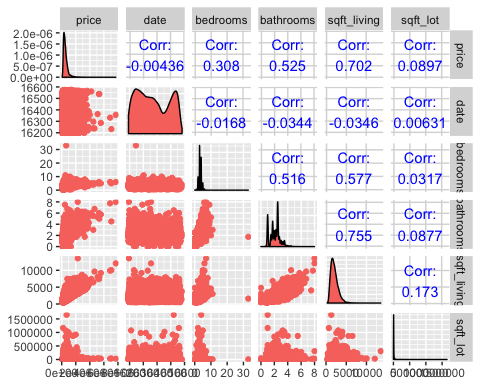
## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

# Convert Date format for regression

kc\_house\_data$date <- (substr(kc\_house\_data$date,1,8))  
kc\_house\_data$date <- ymd(kc\_house\_data$date)  
kc\_house\_data$date<-as.numeric(as.Date(kc\_house\_data$date, origin = "1900-01-01"))

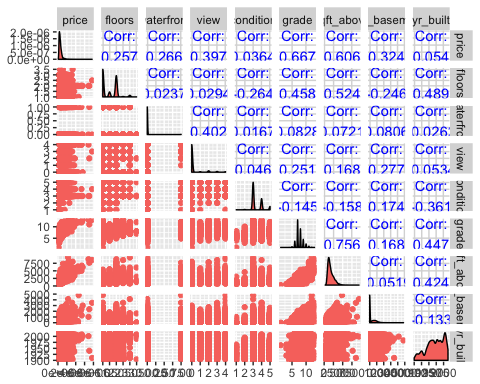
## Checking Relationship between price, date, bedrooms, bathrooms, sqft\_living and sqft lot

plot1<- ggpairs(data=kc\_house\_data, columns=c(3,2,4:7),mapping = aes(color = "blue"),axisLabels="show")  
plot1

 ##

Checking Relationship between price, floors, waterfront, view, condition,grade,sqft\_above,sqft\_basement,yr\_built

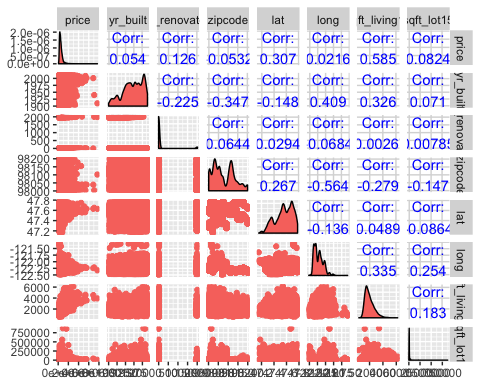
plot2<-ggpairs(data=kc\_house\_data, columns=c(3,8:15),mapping = aes(color = "blue"),axisLabels = "show")  
plot2



## 

## Checking Relationship between price,yr\_renovated,zipcode,lat,long,sqft\_living15,sqft\_lot15

plot3<-ggpairs(data=kc\_house\_data, columns=c(3,15:21),mapping = aes(color = "blue"),axisLabels = "show")  
plot3



## 

## Regression Methods

# Linear Regression

library(caret)  
set.seed(100)  
indx <- createFolds(y=kc\_house\_data$price,k=10,returnTrain=TRUE)  
ctrl <- trainControl(method="cv",index=indx)

set.seed(100)  
lmTune0 <- train(x= kc\_house\_data[,-c(1:3)],y=kc\_house\_data$price,method="lm",trControl = ctrl)

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient  
## fit may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient  
## fit may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient  
## fit may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient  
## fit may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient  
## fit may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient  
## fit may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient  
## fit may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient  
## fit may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient  
## fit may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient  
## fit may be misleading

lmTune0

## Linear Regression   
##   
## 21613 samples  
## 18 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 19452, 19451, 19453, 19451, 19452, 19451, ...   
## Resampling results:  
##   
## RMSE Rsquared   
## 201348.9 0.6997163  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE  
##

summary(lmTune0)

##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1291725 -99229 -9739 77583 4333222   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 6.690e+06 2.931e+06 2.282 0.02249 \*   
## bedrooms -3.577e+04 1.892e+03 -18.906 < 2e-16 \*\*\*  
## bathrooms 4.114e+04 3.254e+03 12.645 < 2e-16 \*\*\*  
## sqft\_living 1.501e+02 4.385e+00 34.227 < 2e-16 \*\*\*  
## sqft\_lot 1.286e-01 4.792e-02 2.683 0.00729 \*\*   
## floors 6.690e+03 3.596e+03 1.860 0.06285 .   
## waterfront 5.830e+05 1.736e+04 33.580 < 2e-16 \*\*\*  
## view 5.287e+04 2.140e+03 24.705 < 2e-16 \*\*\*  
## condition 2.639e+04 2.351e+03 11.221 < 2e-16 \*\*\*  
## grade 9.589e+04 2.153e+03 44.542 < 2e-16 \*\*\*  
## sqft\_above 3.113e+01 4.360e+00 7.139 9.71e-13 \*\*\*  
## sqft\_basement NA NA NA NA   
## yr\_built -2.620e+03 7.266e+01 -36.062 < 2e-16 \*\*\*  
## yr\_renovated 1.981e+01 3.656e+00 5.420 6.03e-08 \*\*\*  
## zipcode -5.824e+02 3.299e+01 -17.657 < 2e-16 \*\*\*  
## lat 6.027e+05 1.073e+04 56.149 < 2e-16 \*\*\*  
## long -2.147e+05 1.313e+04 -16.349 < 2e-16 \*\*\*  
## sqft\_living15 2.168e+01 3.448e+00 6.289 3.26e-10 \*\*\*  
## sqft\_lot15 -3.826e-01 7.327e-02 -5.222 1.78e-07 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 201200 on 21595 degrees of freedom  
## Multiple R-squared: 0.6997, Adjusted R-squared: 0.6995   
## F-statistic: 2960 on 17 and 21595 DF, p-value: < 2.2e-16

## Recursive Partitioning

set.seed(123)  
cartTune <- train(x= kc\_house\_data[,-c(1:3)],y=kc\_house\_data$price,method = "rpart",tuneLength = 25,trControl = ctrl)

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =  
## trainInfo, : There were missing values in resampled performance measures.

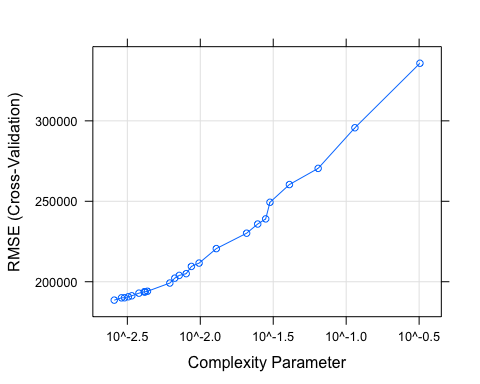
cartTune

## CART   
##   
## 21613 samples  
## 18 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 19452, 19451, 19453, 19451, 19452, 19451, ...   
## Resampling results across tuning parameters:  
##   
## cp RMSE Rsquared   
## 0.002571405 188565.7 0.7370914  
## 0.002883034 189988.8 0.7332492  
## 0.003025883 190042.0 0.7330008  
## 0.003211805 190657.5 0.7311106  
## 0.003391812 191255.6 0.7294279  
## 0.003795596 192889.0 0.7245934  
## 0.004125403 193708.2 0.7222608  
## 0.004172721 193708.2 0.7222608  
## 0.004194801 193775.6 0.7219990  
## 0.004341524 194057.6 0.7210698  
## 0.006180285 199178.1 0.7060982  
## 0.006688211 202223.8 0.6969940  
## 0.007173163 203939.3 0.6918447  
## 0.008008066 205019.3 0.6884694  
## 0.008691089 209515.0 0.6748434  
## 0.009813059 211589.2 0.6686036  
## 0.012878915 220585.0 0.6400012  
## 0.020774840 230161.6 0.6072999  
## 0.024744699 235857.8 0.5873241  
## 0.028072436 239060.8 0.5753306  
## 0.030034119 249386.8 0.5378707  
## 0.040822368 260397.7 0.4974046  
## 0.064259982 270459.4 0.4559730  
## 0.114781587 295714.5 0.3506940  
## 0.320270438 335793.9 0.3033326  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was cp = 0.002571405.

cartTune$finalModel

## n= 21613   
##   
## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 21613 2.912917e+15 540088.1   
## 2) grade< 8.5 17362 6.656597e+14 437284.0   
## 4) lat< 47.53435 7304 1.003407e+14 315438.4   
## 8) sqft\_living< 1938 4566 3.155358e+13 269848.4 \*  
## 9) sqft\_living>=1938 2738 4.347075e+13 391466.1 \*  
## 5) lat>=47.53435 10058 3.781350e+14 525766.8   
## 10) sqft\_living< 2035 6761 1.361306e+14 462801.4   
## 20) sqft\_living< 1455.5 3312 4.810585e+13 410143.4 \*  
## 21) sqft\_living>=1455.5 3449 7.002213e+13 513367.7   
## 42) lat>=47.69585 1146 9.277631e+12 433105.3 \*  
## 43) lat< 47.69585 2303 4.968824e+13 553307.2 \*  
## 11) sqft\_living>=2035 3297 1.602317e+14 654887.1   
## 22) waterfront< 0.5 3279 1.324656e+14 648655.0   
## 44) lat>=47.7123 848 1.029196e+13 518145.2 \*  
## 45) lat< 47.7123 2431 1.026914e+14 694180.4   
## 90) zipcode>=98004.5 2354 8.549691e+13 682650.4 \*  
## 91) zipcode< 98004.5 77 7.314424e+12 1046669.0 \*  
## 23) waterfront>=0.5 18 4.439324e+12 1790167.0 \*  
## 3) grade>=8.5 4251 1.314336e+15 959962.4   
## 6) sqft\_living< 4185 3667 4.934437e+14 848043.0   
## 12) lat< 47.5231 810 3.289091e+13 557955.7 \*  
## 13) lat>=47.5231 2857 3.730659e+14 930286.8   
## 26) long>=-122.1865 1583 6.476840e+13 805650.8   
## 52) sqft\_living< 3155 892 2.181817e+13 726982.2 \*  
## 53) sqft\_living>=3155 691 3.030373e+13 907202.7 \*  
## 27) long< -122.1865 1274 2.531521e+14 1085152.0   
## 54) sqft\_living< 3045 737 7.221140e+13 891035.6   
## 108) yr\_built>=1977.5 436 3.332748e+13 784938.5 \*  
## 109) yr\_built< 1977.5 301 2.686696e+13 1044718.0 \*  
## 55) sqft\_living>=3045 537 1.150553e+14 1351566.0   
## 110) sqft\_living15< 3405 435 6.538577e+13 1256048.0   
## 220) lat>=47.69365 83 1.050495e+13 962905.7 \*  
## 221) lat< 47.69365 352 4.606667e+13 1325169.0 \*  
## 111) sqft\_living15>=3405 102 2.877472e+13 1758925.0 \*  
## 7) sqft\_living>=4185 584 4.865430e+14 1662717.0   
## 14) sqft\_living< 7940 574 3.372812e+14 1603157.0   
## 28) long>=-122.1875 315 7.499599e+13 1281833.0   
## 56) waterfront< 0.5 304 4.360610e+13 1224531.0   
## 112) grade< 11.5 255 2.285437e+13 1136878.0 \*  
## 113) grade>=11.5 49 8.596943e+12 1680682.0 \*  
## 57) waterfront>=0.5 11 2.805271e+12 2865455.0 \*  
## 29) long< -122.1875 259 1.902060e+14 1993957.0   
## 58) lat< 47.52195 27 6.625440e+12 882916.7 \*  
## 59) lat>=47.52195 232 1.463726e+14 2123260.0   
## 118) sqft\_above< 4755 202 8.983921e+13 1967657.0   
## 236) sqft\_living< 5005 151 5.461881e+13 1824722.0   
## 472) lat>=47.70175 14 1.411618e+12 1086993.0 \*  
## 473) lat< 47.70175 137 4.480915e+13 1900110.0 \*  
## 237) sqft\_living>=5005 51 2.300129e+13 2390859.0 \*  
## 119) sqft\_above>=4755 30 1.871090e+13 3170982.0   
## 238) sqft\_lot15< 14539 14 2.404687e+12 2573982.0 \*  
## 239) sqft\_lot15>=14539 16 6.950490e+12 3693356.0 \*  
## 15) sqft\_living>=7940 10 3.034967e+13 5081430.0 \*

plot(cartTune, scales = list(x = list(log = 10)))



# Ridge Regression

ridgeGrid <- expand.grid(lambda = seq(0, 0.1, by = 0.005))  
set.seed(100)  
ridgeTune <- train(x= kc\_house\_data[,-c(1:3)],y=kc\_house\_data$price,method ="ridge",tuneGrid = ridgeGrid,trControl = ctrl,preProc = c("center", "scale"))

## Loading required package: elasticnet

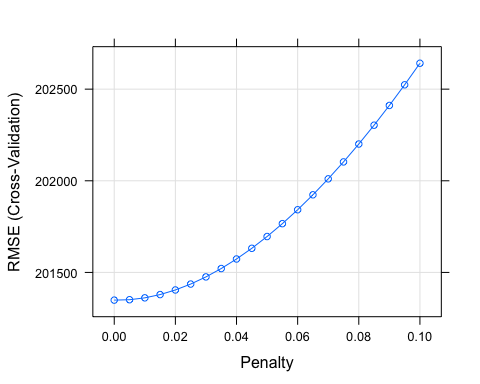
## Loading required package: lars

## Loaded lars 1.2

ridgeTune

## Ridge Regression   
##   
## 21613 samples  
## 18 predictor  
##   
## Pre-processing: centered (18), scaled (18)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 19452, 19451, 19453, 19451, 19452, 19451, ...   
## Resampling results across tuning parameters:  
##   
## lambda RMSE Rsquared   
## 0.000 201348.9 0.6997163  
## 0.005 201351.1 0.6997229  
## 0.010 201361.3 0.6997145  
## 0.015 201379.1 0.6996924  
## 0.020 201404.3 0.6996576  
## 0.025 201436.5 0.6996113  
## 0.030 201475.6 0.6995544  
## 0.035 201521.2 0.6994876  
## 0.040 201573.3 0.6994117  
## 0.045 201631.5 0.6993275  
## 0.050 201695.9 0.6992356  
## 0.055 201766.1 0.6991365  
## 0.060 201842.0 0.6990307  
## 0.065 201923.6 0.6989189  
## 0.070 202010.6 0.6988013  
## 0.075 202102.9 0.6986785  
## 0.080 202200.5 0.6985509  
## 0.085 202303.3 0.6984186  
## 0.090 202411.0 0.6982822  
## 0.095 202523.7 0.6981419  
## 0.100 202641.2 0.6979979  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was lambda = 0.

print(update(plot(ridgeTune), xlab = "Penalty"))



## Principle Component Analysis

set.seed(100)  
pcrTune <- train(x= kc\_house\_data[,-c(1:3)],y=kc\_house\_data$price,method = "pcr",tuneGrid = expand.grid(ncomp = 1:18),trControl = ctrl)

## Loading required package: pls

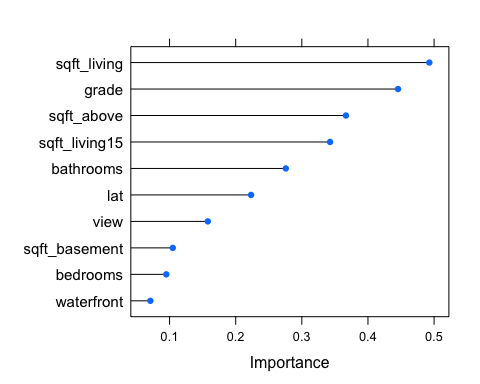
##   
## Attaching package: 'pls'

## The following object is masked from 'package:stats':  
##   
## loadings

pcrTune

## Principal Component Analysis   
##   
## 21613 samples  
## 18 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 19452, 19451, 19453, 19451, 19452, 19451, ...   
## Resampling results across tuning parameters:  
##   
## ncomp RMSE Rsquared   
## 1 365218.8 0.009732164  
## 2 365302.2 0.009334006  
## 3 263937.6 0.482891756  
## 4 257817.8 0.506568429  
## 5 256875.0 0.510074147  
## 6 256324.3 0.512241293  
## 7 253416.4 0.523310295  
## 8 247780.4 0.544312629  
## 9 232713.2 0.598418058  
## 10 226070.0 0.621046889  
## 11 221132.1 0.637589001  
## 12 221085.7 0.637744402  
## 13 220740.5 0.638857879  
## 14 220748.4 0.638835182  
## 15 209092.7 0.676125432  
## 16 206388.4 0.684575054  
## 17 201348.9 0.699716295  
## 18 201325.1 0.699741722  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was ncomp = 18.

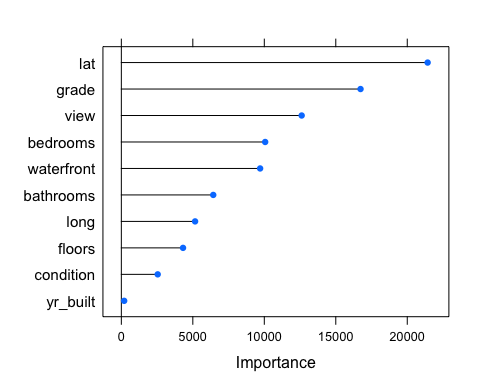
pcrImp <- varImp(pcrTune, scale = FALSE)  
plot(pcrImp, top = 10, scales = list(y = list(cex = .95)))

 ## Partial Least Squares

set.seed(100)  
plsTune <- train(x= kc\_house\_data[,-c(1:3)],y=kc\_house\_data$price,method = "pls",tuneGrid = expand.grid(ncomp = 1:18),trControl = ctrl)  
plsTune

## Partial Least Squares   
##   
## 21613 samples  
## 18 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 19452, 19451, 19453, 19451, 19452, 19451, ...   
## Resampling results across tuning parameters:  
##   
## ncomp RMSE Rsquared   
## 1 3.650951e+05 0.01052339  
## 2 3.449337e+05 0.13810801  
## 3 2.617864e+05 0.49128530  
## 4 2.564127e+05 0.51191290  
## 5 2.560819e+05 0.51315603  
## 6 2.535649e+05 0.52285462  
## 7 2.507277e+05 0.53341971  
## 8 2.477204e+05 0.54453383  
## 9 2.206967e+05 0.63898222  
## 10 2.186016e+05 0.64589168  
## 11 2.102745e+05 0.67239754  
## 12 2.046334e+05 0.68979381  
## 13 2.031801e+05 0.69426845  
## 14 2.028876e+05 0.69514967  
## 15 2.023715e+05 0.69666662  
## 16 2.014443e+05 0.69945798  
## 17 2.013489e+05 0.69971629  
## 18 1.317923e+17 0.01062210  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was ncomp = 17.

plsImp <- varImp(plsTune, scale = FALSE)  
plot(plsImp, top = 10, scales = list(y = list(cex = .95)))



# Random Forest

mtryGrid <- data.frame(mtry = floor(seq(1,ncol(kc\_house\_data[,-c(1:3)]),length = 10)))  
rfTune <- train(x= kc\_house\_data[,-c(1,3)],y=kc\_house\_data$price,method = "rf",tuneGrid = mtryGrid,ntree = 50,importance = TRUE,trControl = ctrl)

## Loading required package: randomForest

## randomForest 4.6-12

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

##   
## Attaching package: 'randomForest'

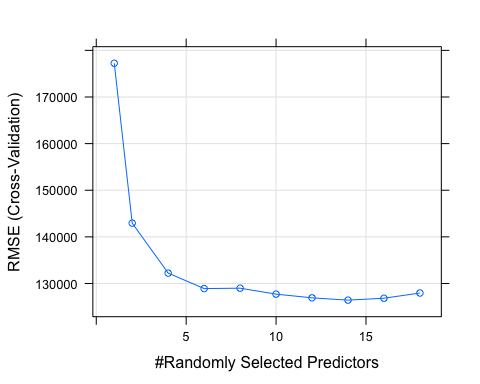
## The following object is masked from 'package:dplyr':  
##   
## combine

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

summary(rfTune)

## Length Class Mode   
## call 6 -none- call   
## type 1 -none- character  
## predicted 21613 -none- numeric   
## mse 50 -none- numeric   
## rsq 50 -none- numeric   
## oob.times 21613 -none- numeric   
## importance 38 -none- numeric   
## importanceSD 19 -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 21613 -none- numeric   
## test 0 -none- NULL   
## inbag 0 -none- NULL   
## xNames 19 -none- character  
## problemType 1 -none- character  
## tuneValue 1 data.frame list   
## obsLevels 1 -none- logical

plot(rfTune)



rfTune

## Random Forest   
##   
## 21613 samples  
## 19 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 19452, 19451, 19453, 19451, 19452, 19451, ...   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared   
## 1 177234.3 0.8191911  
## 2 142946.4 0.8604216  
## 4 132227.8 0.8748278  
## 6 128909.0 0.8797568  
## 8 128991.7 0.8787783  
## 10 127714.6 0.8809891  
## 12 126924.1 0.8821206  
## 14 126431.9 0.8830863  
## 16 126841.3 0.8822263  
## 18 127958.6 0.8798717  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 14.

# MARS

library(earth)

## Loading required package: plotmo

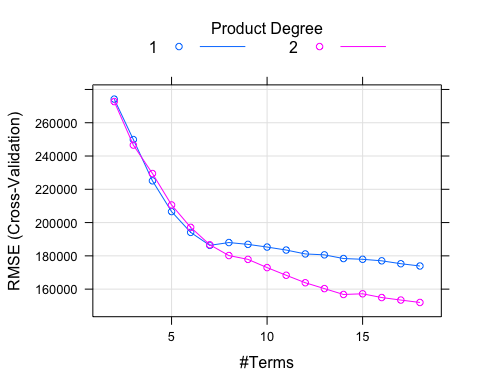
## Loading required package: plotrix

## Loading required package: TeachingDemos

marsGrid <- expand.grid(.degree = 1:2, .nprune = 2:18)  
set.seed(100)  
  
marsTuned <- train(x=kc\_house\_data[,-c(1:3)],y=kc\_house\_data$price, method = "earth",tuneGrid = marsGrid,trControl = ctrl)  
marsTuned

## Multivariate Adaptive Regression Spline   
##   
## 21613 samples  
## 18 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 19452, 19451, 19453, 19451, 19452, 19451, ...   
## Resampling results across tuning parameters:  
##   
## degree nprune RMSE Rsquared   
## 1 2 274195.0 0.4426346  
## 1 3 249881.9 0.5363547  
## 1 4 225093.0 0.6240259  
## 1 5 206612.4 0.6832630  
## 1 6 194103.4 0.7205775  
## 1 7 186310.8 0.7427537  
## 1 8 187995.2 0.7420285  
## 1 9 186899.3 0.7453746  
## 1 10 185289.2 0.7497198  
## 1 11 183487.9 0.7543182  
## 1 12 181155.2 0.7601430  
## 1 13 180602.3 0.7618252  
## 1 14 178384.9 0.7672942  
## 1 15 177932.2 0.7691194  
## 1 16 177017.2 0.7714519  
## 1 17 175278.7 0.7756524  
## 1 18 173899.8 0.7788536  
## 2 2 272726.1 0.4485349  
## 2 3 246571.4 0.5483444  
## 2 4 229468.0 0.6076553  
## 2 5 210596.1 0.6702919  
## 2 6 197131.8 0.7111430  
## 2 7 186680.6 0.7409882  
## 2 8 180184.6 0.7583728  
## 2 9 177884.2 0.7647039  
## 2 10 172943.9 0.7782279  
## 2 11 168330.4 0.7900343  
## 2 12 163841.7 0.8006576  
## 2 13 160314.0 0.8097485  
## 2 14 156791.8 0.8177308  
## 2 15 157201.4 0.8173221  
## 2 16 154940.7 0.8227315  
## 2 17 153434.0 0.8259294  
## 2 18 151983.4 0.8292709  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were nprune = 18 and degree = 2.

plot(marsTuned)



# Comparison of all the above models

lmTune0$results

## intercept RMSE Rsquared RMSESD RsquaredSD  
## 1 TRUE 201348.9 0.6997163 11180.77 0.01532767

cartTune$results

## cp RMSE Rsquared RMSESD RsquaredSD  
## 1 0.002571405 188565.7 0.7370914 8214.015 0.01718554  
## 2 0.002883034 189988.8 0.7332492 8720.588 0.01658799  
## 3 0.003025883 190042.0 0.7330008 8842.990 0.01635345  
## 4 0.003211805 190657.5 0.7311106 9404.038 0.01618891  
## 5 0.003391812 191255.6 0.7294279 8968.525 0.01532059  
## 6 0.003795596 192889.0 0.7245934 8920.259 0.01474279  
## 7 0.004125403 193708.2 0.7222608 9183.175 0.01405155  
## 8 0.004172721 193708.2 0.7222608 9183.175 0.01405155  
## 9 0.004194801 193775.6 0.7219990 9177.969 0.01458693  
## 10 0.004341524 194057.6 0.7210698 8913.915 0.01558800  
## 11 0.006180285 199178.1 0.7060982 10371.145 0.01621369  
## 12 0.006688211 202223.8 0.6969940 10690.697 0.01791686  
## 13 0.007173163 203939.3 0.6918447 10029.754 0.01657623  
## 14 0.008008066 205019.3 0.6884694 9375.142 0.01831596  
## 15 0.008691089 209515.0 0.6748434 10420.413 0.02045982  
## 16 0.009813059 211589.2 0.6686036 11386.174 0.01779485  
## 17 0.012878915 220585.0 0.6400012 12457.556 0.01904740  
## 18 0.020774840 230161.6 0.6072999 9816.072 0.01711536  
## 19 0.024744699 235857.8 0.5873241 8986.471 0.02112441  
## 20 0.028072436 239060.8 0.5753306 6756.346 0.03018842  
## 21 0.030034119 249386.8 0.5378707 7425.597 0.02924087  
## 22 0.040822368 260397.7 0.4974046 15732.262 0.01841911  
## 23 0.064259982 270459.4 0.4559730 9046.616 0.02099674  
## 24 0.114781587 295714.5 0.3506940 26213.730 0.06483775  
## 25 0.320270438 335793.9 0.3033326 27192.802 0.01111211

ridgeTune$results

## lambda RMSE Rsquared RMSESD RsquaredSD  
## 1 0.000 201348.9 0.6997163 11180.77 0.01532767  
## 2 0.005 201351.1 0.6997229 11169.00 0.01534875  
## 3 0.010 201361.3 0.6997145 11155.94 0.01537023  
## 4 0.015 201379.1 0.6996924 11141.66 0.01539202  
## 5 0.020 201404.3 0.6996576 11126.21 0.01541404  
## 6 0.025 201436.5 0.6996113 11109.66 0.01543623  
## 7 0.030 201475.6 0.6995544 11092.06 0.01545852  
## 8 0.035 201521.2 0.6994876 11073.46 0.01548086  
## 9 0.040 201573.3 0.6994117 11053.91 0.01550321  
## 10 0.045 201631.5 0.6993275 11033.46 0.01552553  
## 11 0.050 201695.9 0.6992356 11012.15 0.01554777  
## 12 0.055 201766.1 0.6991365 10990.03 0.01556992  
## 13 0.060 201842.0 0.6990307 10967.13 0.01559193  
## 14 0.065 201923.6 0.6989189 10943.49 0.01561380  
## 15 0.070 202010.6 0.6988013 10919.15 0.01563549  
## 16 0.075 202102.9 0.6986785 10894.14 0.01565699  
## 17 0.080 202200.5 0.6985509 10868.49 0.01567829  
## 18 0.085 202303.3 0.6984186 10842.24 0.01569937  
## 19 0.090 202411.0 0.6982822 10815.41 0.01572022  
## 20 0.095 202523.7 0.6981419 10788.03 0.01574083  
## 21 0.100 202641.2 0.6979979 10760.13 0.01576119

pcrTune$results

## ncomp RMSE Rsquared RMSESD RsquaredSD  
## 1 1 365218.8 0.009732164 17418.806 0.003674047  
## 2 2 365302.2 0.009334006 17350.841 0.003683634  
## 3 3 263937.6 0.482891756 11791.718 0.009994156  
## 4 4 257817.8 0.506568429 11230.831 0.013067321  
## 5 5 256875.0 0.510074147 10732.703 0.015343279  
## 6 6 256324.3 0.512241293 10845.536 0.015477857  
## 7 7 253416.4 0.523310295 11359.882 0.013130092  
## 8 8 247780.4 0.544312629 10903.540 0.013875455  
## 9 9 232713.2 0.598418058 9521.581 0.013348744  
## 10 10 226070.0 0.621046889 10557.081 0.014573112  
## 11 11 221132.1 0.637589001 10874.989 0.011577165  
## 12 12 221085.7 0.637744402 10936.536 0.011606628  
## 13 13 220740.5 0.638857879 10885.339 0.011786328  
## 14 14 220748.4 0.638835182 10847.874 0.011683072  
## 15 15 209092.7 0.676125432 11559.517 0.011496245  
## 16 16 206388.4 0.684575054 10830.676 0.009735948  
## 17 17 201348.9 0.699716295 11180.769 0.015327673  
## 18 18 201325.1 0.699741722 11130.371 0.015278786

plsTune$results

## ncomp RMSE Rsquared RMSESD RsquaredSD  
## 1 1 3.650951e+05 0.01052339 1.738239e+04 0.003871754  
## 2 2 3.449337e+05 0.13810801 2.033140e+04 0.075220390  
## 3 3 2.617864e+05 0.49128530 1.158768e+04 0.010429008  
## 4 4 2.564127e+05 0.51191290 1.094599e+04 0.014805819  
## 5 5 2.560819e+05 0.51315603 1.080886e+04 0.015755280  
## 6 6 2.535649e+05 0.52285462 1.162723e+04 0.013314605  
## 7 7 2.507277e+05 0.53341971 1.139962e+04 0.012815981  
## 8 8 2.477204e+05 0.54453383 1.089989e+04 0.013873070  
## 9 9 2.206967e+05 0.63898222 1.042195e+04 0.011767970  
## 10 10 2.186016e+05 0.64589168 1.133762e+04 0.012303106  
## 11 11 2.102745e+05 0.67239754 1.054156e+04 0.011793290  
## 12 12 2.046334e+05 0.68979381 1.158921e+04 0.014036170  
## 13 13 2.031801e+05 0.69426845 1.118777e+04 0.012457610  
## 14 14 2.028876e+05 0.69514967 1.132813e+04 0.012910717  
## 15 15 2.023715e+05 0.69666662 1.128415e+04 0.013494422  
## 16 16 2.014443e+05 0.69945798 1.103692e+04 0.014704557  
## 17 17 2.013489e+05 0.69971629 1.118077e+04 0.015327664  
## 18 18 1.317923e+17 0.01062210 1.516210e+17 0.004190730

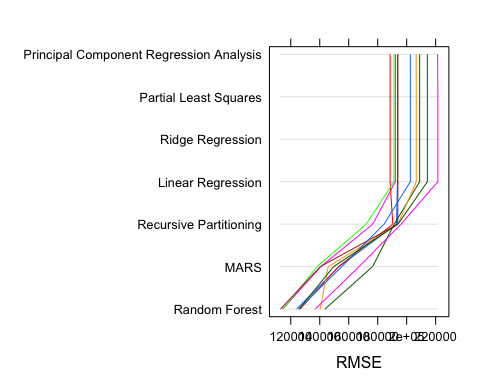
rfTune$results

## mtry RMSE Rsquared RMSESD RsquaredSD  
## 1 1 177234.3 0.8191911 14039.887 0.01365816  
## 2 2 142946.4 0.8604216 11742.144 0.01367836  
## 3 4 132227.8 0.8748278 10334.757 0.01163621  
## 4 6 128909.0 0.8797568 11027.941 0.01524431  
## 5 8 128991.7 0.8787783 10940.005 0.01613129  
## 6 10 127714.6 0.8809891 11117.201 0.01654135  
## 7 12 126924.1 0.8821206 9789.902 0.01384304  
## 8 14 126431.9 0.8830863 10887.586 0.01601876  
## 9 16 126841.3 0.8822263 10076.280 0.01348422  
## 10 18 127958.6 0.8798717 11206.197 0.01592613

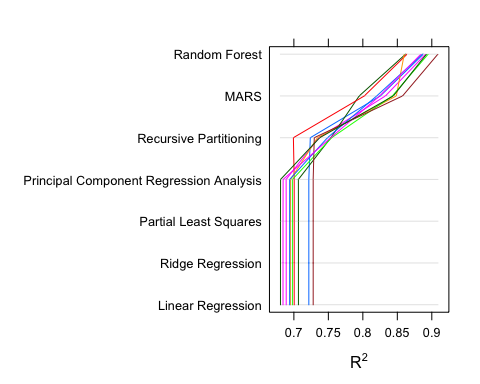
marsTuned$results

## degree nprune RMSE Rsquared RMSESD RsquaredSD  
## 1 1 2 274195.0 0.4426346 14578.930 0.04931547  
## 18 2 2 272726.1 0.4485349 14611.634 0.04896916  
## 2 1 3 249881.9 0.5363547 10265.309 0.02864829  
## 19 2 3 246571.4 0.5483444 11457.388 0.03371203  
## 3 1 4 225093.0 0.6240259 11944.847 0.02536216  
## 20 2 4 229468.0 0.6076553 12848.995 0.05286321  
## 4 1 5 206612.4 0.6832630 10399.086 0.01517336  
## 21 2 5 210596.1 0.6702919 11520.887 0.03407173  
## 5 1 6 194103.4 0.7205775 10935.384 0.01687189  
## 22 2 6 197131.8 0.7111430 9338.429 0.02084027  
## 6 1 7 186310.8 0.7427537 10368.361 0.01334423  
## 23 2 7 186680.6 0.7409882 8898.978 0.02117512  
## 7 1 8 187995.2 0.7420285 24520.299 0.04291790  
## 24 2 8 180184.6 0.7583728 9104.175 0.02459087  
## 8 1 9 186899.3 0.7453746 23830.581 0.04051292  
## 25 2 9 177884.2 0.7647039 9465.314 0.02691669  
## 9 1 10 185289.2 0.7497198 23845.421 0.04031964  
## 26 2 10 172943.9 0.7782279 9579.405 0.02523266  
## 10 1 11 183487.9 0.7543182 23185.218 0.03991297  
## 27 2 11 168330.4 0.7900343 9857.609 0.02361499  
## 11 1 12 181155.2 0.7601430 21382.704 0.03510837  
## 28 2 12 163841.7 0.8006576 10175.244 0.02495362  
## 12 1 13 180602.3 0.7618252 24073.596 0.04149042  
## 29 2 13 160314.0 0.8097485 10021.785 0.02089147  
## 13 1 14 178384.9 0.7672942 21770.283 0.03618683  
## 30 2 14 156791.8 0.8177308 8971.199 0.02110435  
## 14 1 15 177932.2 0.7691194 23024.880 0.03753506  
## 31 2 15 157201.4 0.8173221 11999.081 0.02230959  
## 15 1 16 177017.2 0.7714519 23261.758 0.03825057  
## 32 2 16 154940.7 0.8227315 11469.373 0.01959051  
## 16 1 17 175278.7 0.7756524 22995.265 0.03796012  
## 33 2 17 153434.0 0.8259294 12232.051 0.02008873  
## 17 1 18 173899.8 0.7788536 20327.277 0.03159255  
## 34 2 18 151983.4 0.8292709 12205.684 0.02012909

allsamples <- resamples(list("Linear Regression"=lmTune0, "Recursive Partitioning"=cartTune,"Ridge Regression"= ridgeTune,"Principal Component Regression Analysis"=pcrTune,"Partial Least Squares"=plsTune, "Random Forest"=rfTune,"MARS"=marsTuned))  
parallelplot(allsamples)



parallelplot(allsamples,metric = "Rsquared")



**## Plan of Work**

As you can see from the above plots, it is quite evident that Random forest is the best model amongst all the other regression methods as it has the least RMSE and the highest R^2 value.

In this project submission, we just explored different regression methods and compared all of them. Although, Random forest turned out to be the best model and the second best model being MARS, it is very important to note that these are very complex methods that take very large computational time. Since the RMSE value of Random forest model is significantly lower than linear regression models, it seems like a reasonable thing to do to go ahead with Random Forest method.

For the next project submission, there are a few things to consider after deciding on the final model. Predictor selection is one of those things. We can reduce the complexity of the model by reducing predictor variables that don’t have much effect on the response variable. Normalization and Accuracy-Simplicity trade-off are the other factors to be considered.