Regression Kernel and Ensemble

Vladimir Sokolov

10/25/2022

This notebook explores King County House Sales data from Kaggle.

Load the kc_housing_data.csv file and change waterfront into a factor.

```
df <- read.csv("kc_house_data.csv")
df$waterfront <- factor(df$waterfront)
str(df)
```

```
## 'data.frame':    21613 obs. of  21 variables:
##  $ id           : num  7.13e+09 6.41e+09 5.63e+09 2.49e+09 1.95e+09 ...
##  $ date         : chr  "20141013T000000" "20141209T000000" "20150225T000000" "20141209T000000" ...
##  $ price        : num  221900 538000 180000 604000 510000 ...
##  $ bedrooms     : int  3 3 2 4 3 4 3 3 3 3 ...
##  $ bathrooms    : num  1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
##  $ sqft_living  : int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890 ...
##  $ sqft_lot     : int  5650 7242 10000 5000 8080 101930 6819 9711 7470 6560 ...
##  $ floors       : num  1 2 1 1 1 1 2 1 1 2 ...
##  $ waterfront   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ view         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ condition    : int  3 3 3 5 3 3 3 3 3 3 ...
##  $ grade        : int  7 7 6 7 8 11 7 7 7 7 ...
##  $ sqft_above   : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890 ...
##  $ sqft_basement: int  0 400 0 910 0 1530 0 0 730 0 ...
##  $ yr_built     : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003 ...
##  $ yr_renovated : int  0 1991 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98178 98125 98028 98136 98074 98053 98003 98198 98146 98038 ...
##  $ lat          : num  47.5 47.7 47.7 47.5 47.6 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  1340 1690 2720 1360 1800 4760 2238 1650 1780 2390 ...
##  $ sqft_lot15   : int  5650 7639 8062 5000 7503 101930 6819 9711 8113 7570 ...
```

Simplify variables.

```
df<-df[-c(1,2,8,14:21)]
str(df)
```

```
## 'data.frame':    21613 obs. of  10 variables:
##  $ price      : num  221900 538000 180000 604000 510000 ...
##  $ bedrooms   : int  3 3 2 4 3 4 3 3 3 3 ...
##  $ bathrooms  : num  1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
##  $ sqft_living: int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890 ...
```

```
##  $ sqft_lot   : int  5650 7242 10000 5000 8080 101930 6819 9711 7470 6560 ...
##  $ waterfront : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ view       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ condition  : int  3 3 3 5 3 3 3 3 3 3 ...
##  $ grade      : int  7 7 6 7 8 11 7 7 7 7 ...
##  $ sqft_above : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890 ...
```

Check for null values.

```
sapply(df, function(x) sum(is.na(x)))
```

```
##       price    bedrooms   bathrooms sqft_living    sqft_lot  waterfront
##           0           0           0           0           0           0
##        view   condition       grade  sqft_above
##           0           0           0           0
```

#A Divide into Train and Test

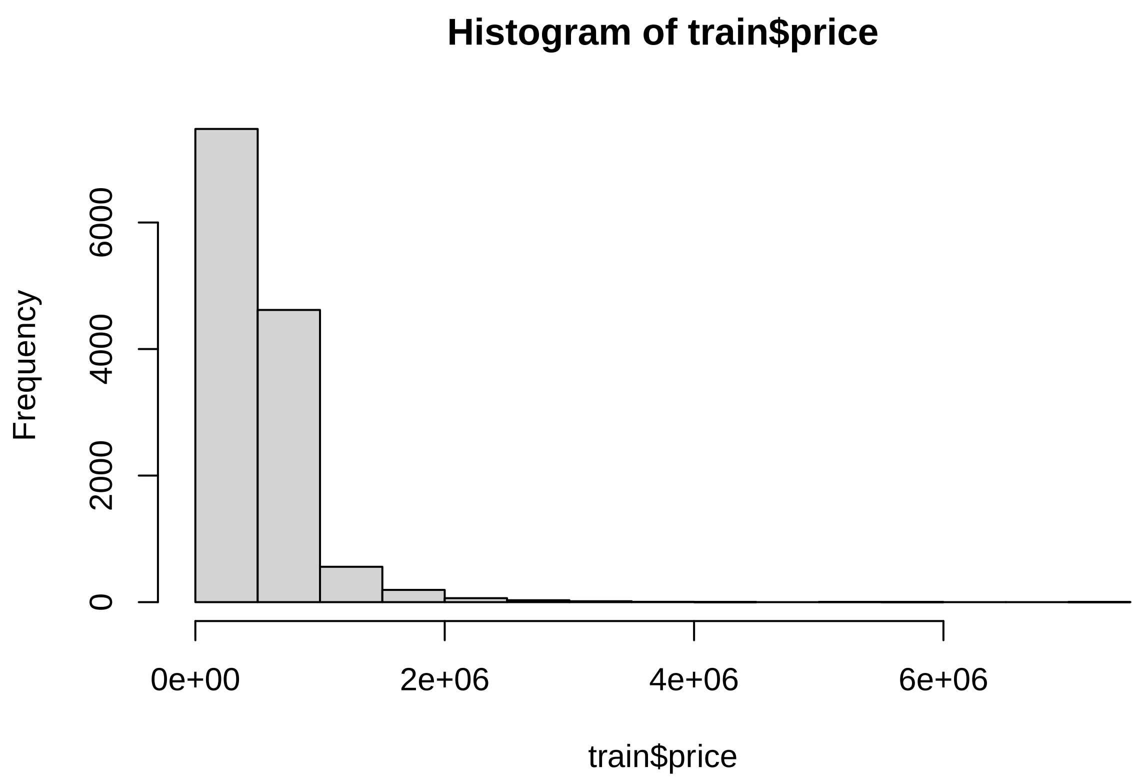
```
set.seed(12345)
spec <- c(train=.6, test=.2, validate=.2)
i <- sample(cut(1:nrow(df), nrow(df)*cumsum(c(0,spec)), labels=names(spec)))
train <- df[i=="train",]
test <- df[i=="test",]
vald <- df[i=="validate",]
```

#B Explore training data

```
summary(train$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   75000  324650  450000  541675  649000 7062500
```

```
hist(train$price)
```

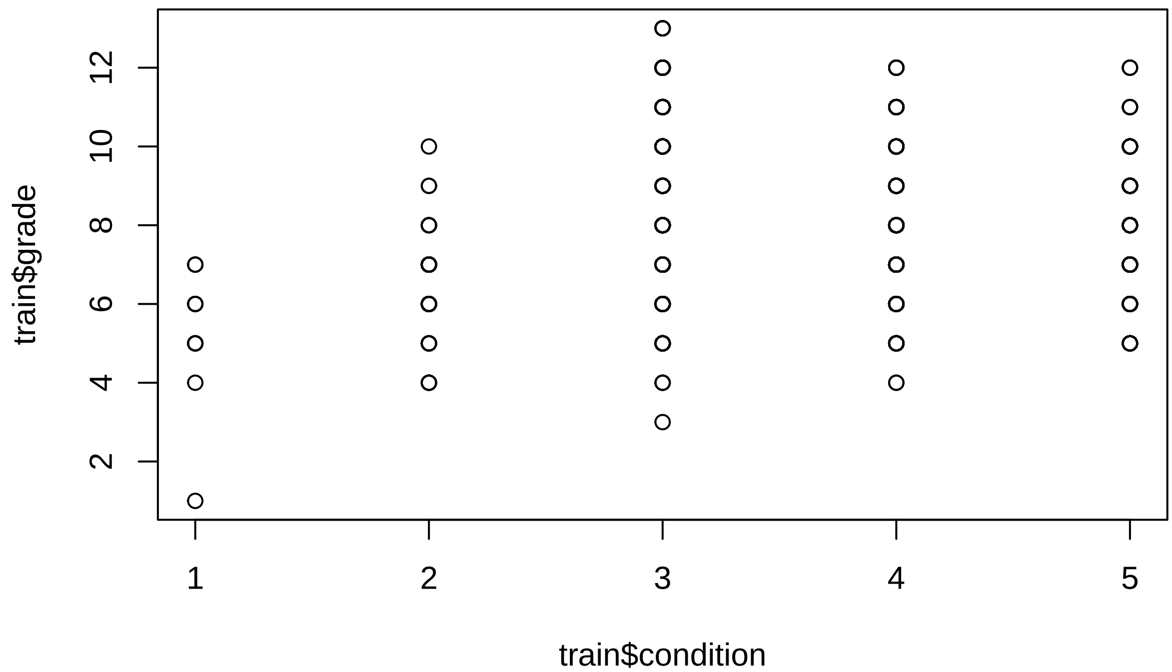



```
summary(train$sqft_living)
```

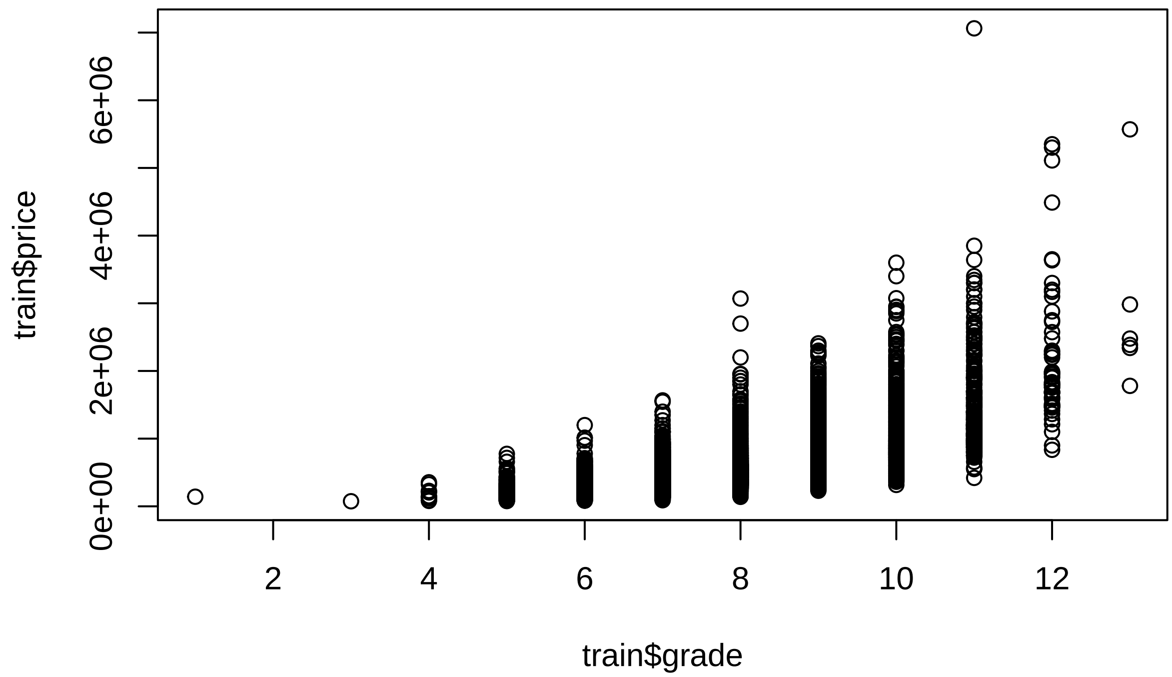
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     290    1420    1910    2084    2560   13540
```

```
hist(train$sqft_living)
```


```
plot(train$condition, train$grade)
```




```
plot(train$grade, train$price)
```




```
summary(train)
```

```
##      price            bedrooms        bathrooms       sqft_living   
##  Min.   :  75000   Min.   : 0.000   Min.   :0.000   Min.   :  290  
##  1st Qu.: 324650   1st Qu.: 3.000   1st Qu.:1.500   1st Qu.: 1420  
##  Median : 450000   Median : 3.000   Median :2.250   Median : 1910  
##  Mean   : 541675   Mean   : 3.369   Mean   :2.115   Mean   : 2084  
##  3rd Qu.: 649000   3rd Qu.: 4.000   3rd Qu.:2.500   3rd Qu.: 2560  
##  Max.   :7062500   Max.   :33.000   Max.   :8.000   Max.   :13540  
##     sqft_lot       waterfront      view          condition         grade       
##  Min.   :    520   0:12875    Min.   :0.0000   Min.   :1.000   Min.   : 1.000  
##  1st Qu.:   5060   1:   92    1st Qu.:0.0000   1st Qu.:3.000   1st Qu.: 7.000  
##  Median :   7590              Median :0.0000   Median :3.000   Median : 7.000  
##  Mean   :  14907              Mean   :0.2324   Mean   :3.403   Mean   : 7.667  
##  3rd Qu.:  10586              3rd Qu.:0.0000   3rd Qu.:4.000   3rd Qu.: 8.000  
##  Max.   :1164794              Max.   :4.0000   Max.   :5.000   Max.   :13.000  
##    sqft_above  
##  Min.   : 290  
##  1st Qu.:1190  
##  Median :1570  
##  Mean   :1793  
##  3rd Qu.:2230  
##  Max.   :9410  
```

#C SVM Classification

Linear Kernel with c of 10.

```r
library(e1071)
svm1 <- svm(price~., data=train, kernel="linear", cost=10, scale=TRUE)
summary(svm1)
```

```
##
## Call:
## svm(formula = price ~ ., data = train, kernel = "linear", cost = 10,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  10424
```

Predict and output correlation.

```r
pred1 <- predict(svm1, newdata=test)
cor(pred1, test$price)
```

```
## [1] 0.7743688
```

Polynomial kernel with c of 10.

```r
svm2 <-svm(price~., data=train, kernel="polynomial", cost=10, scale=TRUE)
summary(svm2)
```

```
##
## Call:
## svm(formula = price ~ ., data = train, kernel = "polynomial", cost = 10,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  3
##       gamma:  0.1
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  10358
```

Predict and output correlation.

```
pred2 <- predict(svm2, newdata=test)
cor(pred2, test$price)
```

```
## [1] 0.8327338
```

Radial kernel with cost of 10 and gamma of 1.

```
svm3 <- svm(price~., data=train, kernel="radial", cost=10, gamma=1, scale=TRUE)
summary(svm3)
```

```
##
## Call:
## svm(formula = price ~ ., data = train, kernel = "radial", cost = 10,
##     gamma = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  10434
```

Predict and output correlation.

```
pred3 <- predict(svm3, newdata=test)
cor(pred3, test$price)
```

```
## [1] 0.6686222
```

Try to tune linear

```
set.seed(12345)
tune.out <-tune(svm, price~., data=vald, kernel="linear", ranges=list(cost=c(0.1,1,10,100)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   100
##
## - best performance: 55164957350
```

```
##
## - Detailed performance results:
##    cost       error  dispersion
## 1   0.1 55422055127 12829376818
## 2   1.0 55169020363 12747922518
## 3  10.0 55179373966 12737825986
## 4 100.0 55164957350 12740135424
```

Next trying the tuned linear kernel with c of 100

```r
svm4 <- svm(price~., data=train, kernel="linear", cost=100, scale=TRUE)
summary(svm4)
```

```
##
## Call:
## svm(formula = price ~ ., data = train, kernel = "linear", cost = 100,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  100
##       gamma:  0.1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  10495
```

Predict and output correlation.

```r
pred4 <- predict(svm4, newdata=test)
cor(pred4, test$price)
```

```
## [1] 0.7747252
```

Try to tune polynomial

```r
set.seed(12345)
tune.out <-tune(svm, price~., data=vald, kernel="polynomial", ranges=list(cost=c(0.1,1,10,100)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
```

```
## - best performance: 65062145510
##
## - Detailed performance results:
##    cost        error   dispersion
## 1   0.1  65062145510  47918694986
## 2   1.0  68380435724  55219008830
## 3  10.0 156142255043 297681248260
## 4 100.0  72375934349  18527186748
```

Next trying the tuned polynomial kernel with c of .1

```r
svm5 <- svm(price~., data=train, kernel="polynomial", cost=.1, scale=TRUE)
summary(svm5)
```

```
##
## Call:
## svm(formula = price ~ ., data = train, kernel = "polynomial", cost = 0.1,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  0.1
##      degree:  3
##       gamma:  0.1
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  10529
```

Predict and output correlation.

```r
pred5 <- predict(svm5, newdata=test)
cor(pred5, test$price)
```

```
## [1] 0.8404673
```

Try to tune radial.

```r
tune.out <-tune(svm, price~., data=vald, kernel="radial", ranges=list(cost=c(0.1,1,10,100), gamma=c(0.5
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
```

```
##    10   0.5
##
## - best performance: 69478258706
##
## - Detailed performance results:
##     cost gamma        error  dispersion
## 1    0.1   0.5  87742520174 19533959334
## 2    1.0   0.5  72944093315 17376073511
## 3   10.0   0.5  69478258706 16384742889
## 4  100.0   0.5  90016146007 17554211903
## 5    0.1   1.0  99197708130 20815783141
## 6    1.0   1.0  84361752939 18801967108
## 7   10.0   1.0  82857836280 17891382392
## 8  100.0   1.0  99255338225 15833654764
## 9    0.1   2.0 109139288513 21331796704
## 10   1.0   2.0  94057654989 19972350309
## 11  10.0   2.0  92039562121 18680503326
## 12 100.0   2.0 107175980186 15805789973
## 13   0.1   3.0 113628828052 21613329185
## 14   1.0   3.0  98460693699 20391031335
## 15  10.0   3.0  96598616807 18896815002
## 16 100.0   3.0 111372655518 14253816266
```

Next trying the tuned radial kernel with c of 10 and gamma of .5

```
library(e1071)
svm6 <- svm(price~., data=train, kernel="radial", cost=10, gamma=0.5, scale=TRUE)
summary(svm6)
```

```
##
## Call:
## svm(formula = price ~ ., data = train, kernel = "radial", cost = 10,
##     gamma = 0.5, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.5
##     epsilon:  0.1
##
##
## Number of Support Vectors:  10398
```

Predict and output correlation.

```
pred6 <- predict(svm6, newdata=test)
cor(pred6, test$price)
```

```
## [1] 0.7217801
```

#D Analysis

```
cor(pred1, test$price)
```

```
## [1] 0.7743688
```

```
cor(pred2, test$price)
```

```
## [1] 0.8327338
```

```
cor(pred3, test$price)
```

```
## [1] 0.6686222
```

```
cor(pred4, test$price)
```

```
## [1] 0.7747252
```

```
cor(pred5, test$price)
```

```
## [1] 0.8404673
```

```
cor(pred6, test$price)
```

```
## [1] 0.7217801
```

The order is linear, polynomial, radial, tuned linear, tuned polynomial, tuned radial. I am not sure how accurate the results are because it said (WARNING: reaching max number of iterations) for a few so I am assuming that a few of the have room for improvement with better hardware. Linear had almost no improvement with tuning while polynomial improved a decent amount and radial improved significantly. After tuning the results were not too far apart despite errors. Linear is likely to be a good fit so it had the best initial results and tuning didn't help that much, though it might have helped more if more c values were explored. Polynomial and radial needed the tuning a lot more and benefited a lot, but were the ones most impacted by the warning errors. Overall linear seems to represent the data the best with radial close behind.