

Housing Price Analysis Using Linear Regression

Part 1. EDA

I first noticed that only 'zipcode' was a non-numeric variable. Thus, I decided to rank the zip codes by their average housing price to get a sense of how they are related to each other.

```
ave.zip = tapply(price, zipcode, mean)
head(sort(ave.zip)) #bottom 5 zip code with the lowest average housing price

##      98002      98168      98032      98001      98148      98023
## 234284.0 240328.4 251296.2 280804.7 284908.6 286732.8

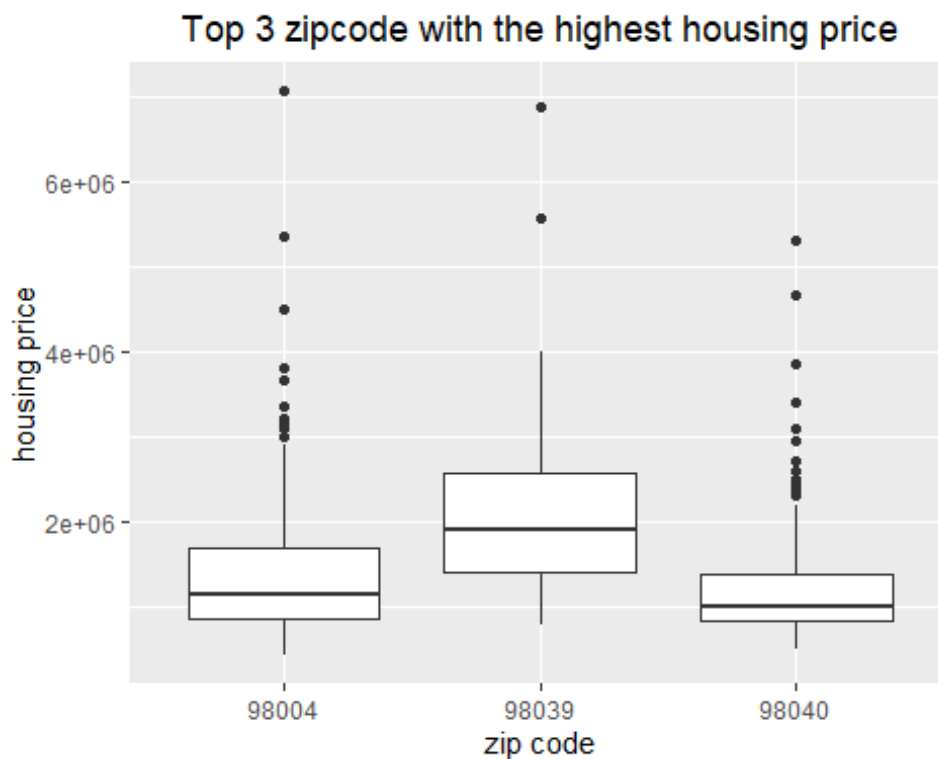
tail(sort(ave.zip)) #top 5 zip code with the highest average housing price

##      98109      98102      98112      98040      98004      98039
## 879623.6 901258.2 1095499.4 1194230.0 1355927.1 2160606.6

top.zip = housing[c(which(zipcode == "98039"),
                    which(zipcode == "98004"),
                    which(zipcode == "98040")),]
```

I sorted the zip codes and found the top 5 and bottom 5 zip codes whose housing prices were the most and least expensive, respectively. Below are boxplots for the top 3 zip codes.

```
#generate boxplot
library(ggplot2) # for ggplot
p = ggplot(top.zip, aes(x = as.factor(zipcode), y = price))+
  geom_boxplot()+
  xlab("zip code")+
  ylab("housing price")+
  labs(title = "Top 3 zipcode with the highest housing price")+
  theme(plot.title = element_text(hjust = 0.5))
p
```



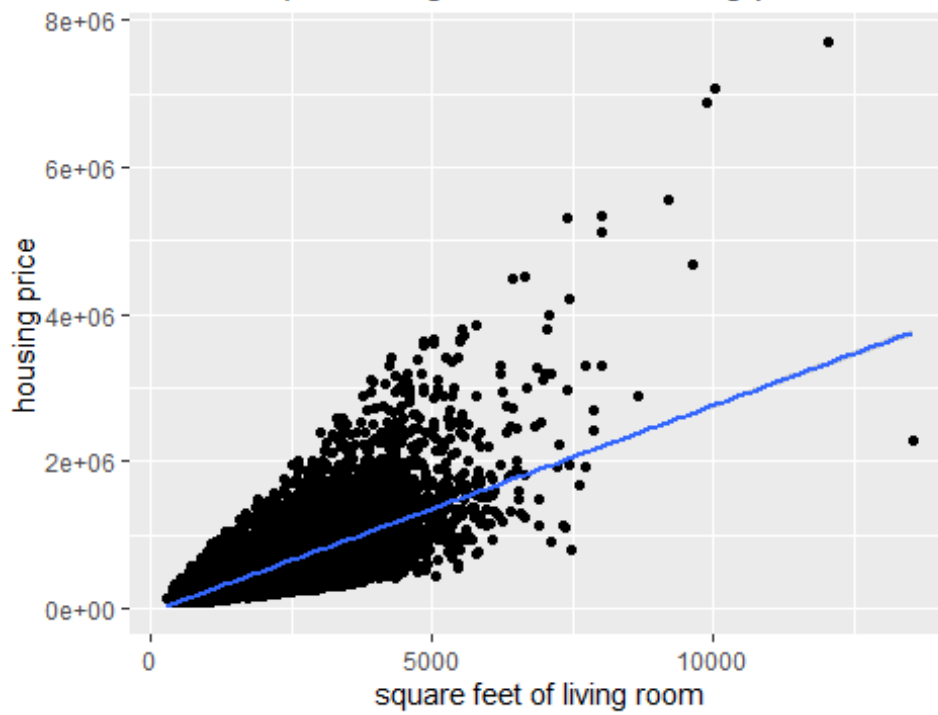
Part 2. EDA (continued)

Now I decided to look at how quantitative variables are related to housing price by plotting scatterplots. Below are the scatterplots for *'square feet of a living room'* and *'square feet of a parking lot'*, which indicate that these variables seem to have positive correlations with housing price. *'Square feet of a living room'* seems to be more highly correlated with housing price than *'square feet of a parking lot'* does.

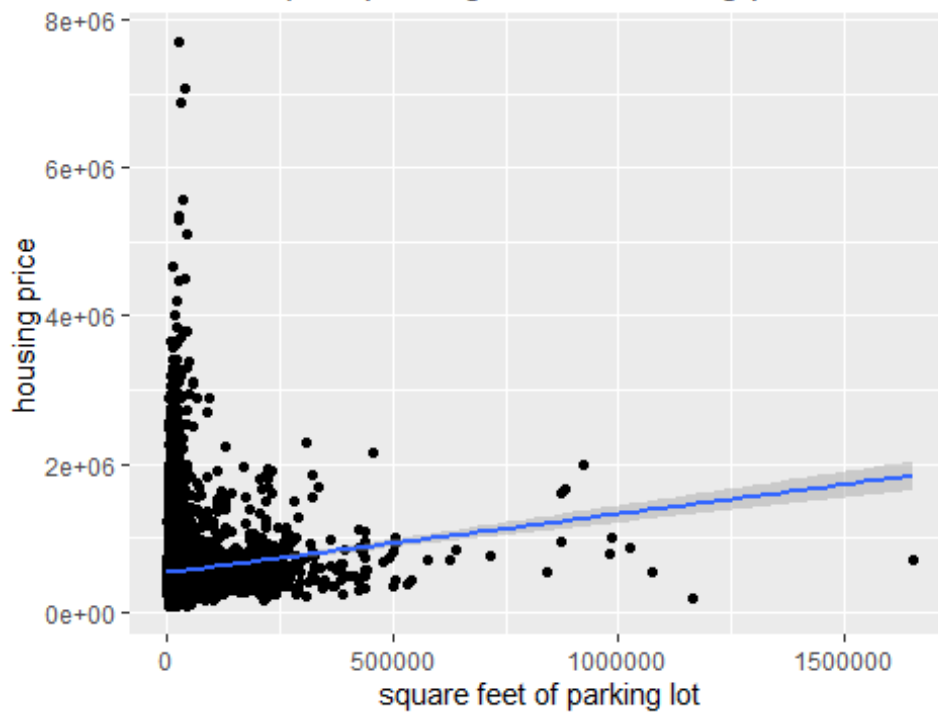
#square feet of living room and housing price

```
p1 = ggplot(housing, aes(x = sqft_living, y = price))+
  geom_point()+
  geom_smooth(method=lm)+
  xlab("square feet of living room")+
  ylab("housing price")+
  labs(title = "Sqft of living room and housing price")+
  theme(plot.title = element_text(hjust = 0.5))
p1
## `geom_smooth()` using formula 'y ~ x'
```

Sqft of living room and housing price



Sqft of parking lot and housing price



Part 3. Modeling

Before applying a linear regression model on the data, I used k-fold cross validation for model assessment. I used $K = 10$, since it was a commonly accepted value for the number of splits.

Next, I fit a linear regression model as follow:

$$\text{Housing price} \sim \text{bedrooms} + \text{bathrooms} + \text{sqft_living} + \text{sqft_lot}$$

Cross Validation (CV) error, which is an estimate of test MSE (Mean Square Error), was calculated by averaging ten CV errors.

Here we got \$66,351,069,200 for test MSE, and 0.502 for adjusted r-squared value.

Given the adjusted r-squared value, this model does not seem to have enough variables that can explain the relationship with the housing price. Test MSE and adjusted r-squared value will be used for feature selection afterwards.

```
#variables: bedroom, bathroom, sqft_living, sqft_lot
library(caret)

## Loading required package: lattice

##k-fold cross validation (approach 1)
k = 10 #number of folds
n = length(housing[,1])
set.seed(1)
folds = createFolds(seq(1:n),k) #split the data in k groups

cv.error.10 = rep(0,k)
for (i in 1:k){
  index = unlist(folds[i],use.names = FALSE)
  train = housing[-index,]
  test = housing[index,]
  model = lm(price ~ bedrooms + bathrooms + sqft_living + sqft_lot,
             data = train)
  cv.error.10[i] = mean((test$price - predict(model, test))^2)
}
mean(cv.error.10) ##CV ERROR WHEN K = 10

## [1] 66351069200
```

```

#cv MSE is $66,351,069,200

summary(model)$adj.r.squared

## [1] 0.5022257

#adj r squared is 0.502

```

I also tried glm function for k-fold cross validation. Not surprisingly we got a similar result for test MSE (\$66,351,830,466).

```

##k-fold cross validation (approach 2: using glm)
library(boot) #for glm

##
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':
##
##      melanoma

set.seed(1)
cv.error.10.2 = rep(0,10)
for(i in 1:10){
  glm.fit = glm(price ~ bedrooms + bathrooms + sqft_living + sqft_lot,
                data = housing)
  cv.error.10.2[i] = cv.glm(housing,glm.fit, K = 10)$delta[1]
}
mean(cv.error.10.2)

## [1] 66351830466

#cv MSE is $66,351,830,466

```

Part 4. Feature Engineering

I added 'zipcode' variable to improve the linear model previously built. Similarly, k-fold cross validation was used, and as a result, the **test MSE decreased dramatically** from \$66,351,069,200 to \$35,510,919,131. This result indicates that '**zipcode**' is a **significant factor for predicting housing price**.

```
#adding zipcode in the linear model

library(boot) #for glm
set.seed(1)
cv.error.zip = rep(0,10)
for(i in 1:10){
  glm.fit.zip = glm(price ~ bedrooms + bathrooms + sqft_living + sqft_lot + a
s.factor(zipcode),
                    data = housing)
  cv.error.zip[i] = cv.glm(housing, glm.fit.zip, K = 10)$delta[1]
}
mean(cv.error.zip)

## [1] 35510919131

#CV MSE is $35,510,919,131 (decreased from the previous model)
```

Part 5. Prediction

Using the improved linear model, I tried to predict a housing price for a fancy house. The predicted housing price is \$14,761,285, which I believe is reasonably expensive given the features of the house. The only concern here is the fact that the fancy house data are quite extreme compared to the values in the dataset used for modeling – the maximum housing price in the given dataset is \$7,700,000. Thus, the model we built might not be able to be extrapolated to this extreme case.

```
fancy$zipcode = as.factor(fancy$zipcode)
predict(glm.fit.zip, fancy)

##      1
## 14761285

#predicted housing price for the fancy house is $14,761,285
```

Part 6. Feature Engineering

Lastly, I tried out some different models to see if there is any room for improvement. I juggled them around by adding new variables, interaction terms, or quadratic terms. Based on the test MSE and adjusted r-squared value, I ultimately landed on my final model as below:

*Housing price ~ bedrooms + bathrooms + bedrooms*bathrooms + sqft_living + sqft_living^2 + sqft_lot + zipcode*

```
#improving the linear model based on the CV MSE
k = 10 #number of folds
n = length(housing[,1])
set.seed(1)
folds = createFolds(seq(1:n),k) #split the data in k groups

#adding terms
cv.error0 = rep(0,k)
cv.error1 = rep(0,k)
cv.error2 = rep(0,k)
cv.error3 = rep(0,k)
cv.error4 = rep(0,k)
cv.error5 = rep(0,k)

for (i in 1:k){
  index = unlist(folds[i],use.names = FALSE)
  train = housing[-index,]
  test = housing[index,]

  #adding zipcode
  m0 = lm(price ~ bedrooms + bathrooms + sqft_living + sqft_lot + as.factor(zipcode),
          data = train)
  m1 = lm(price ~ bedrooms + bathrooms + sqft_living + I(sqft_living^2) + sqft_lot + as.factor(zipcode),
          data = train)
  m2 = lm(price ~ bedrooms + bathrooms + sqft_living + I(sqft_lot^2) + as.factor(zipcode),
          data = train)
  m3 = lm(price ~ bedrooms + bathrooms + sqft_living + sqft_lot + yr_built + as.factor(zipcode),
```

```

    data = train)
  m4 = lm(price ~ bedrooms + bathrooms + sqft_living + sqft_lot + sqft_above
+ as.factor(zipcode))
  m5 = lm(price ~ bedrooms + bathrooms + bedrooms*bathrooms + sqft_living + I
(sqft_living^2) + sqft_lot + as.factor(zipcode),
    data = train)
  cv.error0[i] = mean((test$price - predict(m0, test))^2)
  cv.error1[i] = mean((test$price - predict(m1, test))^2)
  cv.error2[i] = mean((test$price - predict(m2, test))^2)
  cv.error3[i] = mean((test$price - predict(m3, test))^2)
  cv.error4[i] = mean((test$price - predict(m4, test))^2)
  cv.error5[i] = mean((test$price - predict(m5, test))^2)
}
mean(cv.error0)

## [1] 35494949727

mean(cv.error1)

## [1] 32870191570

mean(cv.error2)

## [1] 35552317005

mean(cv.error3)

## [1] 35157847543

mean(cv.error4)

## [1] 34834352240

mean(cv.error5) #Lowest CV ERROR

## [1] 32839272557

summary(m0)$adj.r.squared

## [1] 0.7379652

summary(m1)$adj.r.squared

## [1] 0.767428

summary(m2)$adj.r.squared

## [1] 0.7373796

summary(m3)$adj.r.squared

## [1] 0.7405543

```



```
summary(m4)$adj.r.squared
```

```
## [1] 0.7406675
```

```
summary(m5)$adj.r.squared #highest adjusted r squared
```

```
## [1] 0.7683188
```