House Price Prediction

Group member: Junxi Chen, Xinmei Wang, Zhongwen Liang, Louise Li

Introduction

The goal of this project is to determine the estimated price and find the relationship between house features and how these variables are used to predict the house price. Many people might be considering what is a good price for purchasing a new house and they could be unaware of factors that could affect the price of the house. We recognize these problems and we want to do our analysis to help them. Then, Our data are a collection of (almost) every aspect of residential homes in Ames, Iowa. this dataset that We will be using specifically the train data from the House Prices - Advanced Regression Techniques data. This train data set includes 81 columns but we will only discover some of them, such as sale_price, LotArea, LotShape, and so on. And there is also a test data set which includes most of the columns, except selling price. In other words, we can use these two data sets to compare to verify and predict the value of some variables. Our research results would be helpful to people who are interested in living or moving to the residential areas in Ames, Iowa. Our findings could also help people who are considering purchasing a house to understand some factors that could affect the price of the house and help them to predict the price as well. Furthermore, we will also use the equation for linear regression, LU, QR, Cholesky, and so on. Which we list the important equation here:

$$Linear\ Regression\colon\ y = \beta_0 + \beta_1 X1 + \beta_2 X2 + \ldots$$

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - h_w(x^{(i)}))^2$$

Proposed Method

At the beginning of the project, we need to build a matrix or a model that can contain all of the necessary variables. In this matrix, the first column elements will be 1 so that it can be used to calculate the intercept in linear regression efficiently. The next few columns will be our chosen potential influential variables: lot area, total basement square feet, first and second-floor area, and the garage area. The

reasons why we choose them include because they are not binary variables nor categorical variables, the numeric variables are convenient as well as visualized, and these variables are the most fundamental indicators if we are considering buying a house. The last column of the matrix would be the sale price, which is the response variable we choose to determine.

As we download the dataset from Kaggle, it includes two different parts, which are the training dataset and the test dataset. We know the training dataset may be better to be 80% of the total amount, while the given dataset is only 50% of the total amount. Since the given test dataset does not include the sale price, we cannot use part of the test dataset in the training dataset. Therefore, we would be using the training dataset and our chosen variables to predict the test dataset.

The first method we may apply is the lm method in R, which is the most common method in getting the linear regression model. Although the method is easy and brief, it may take more time than other methods so we would be using other methods to compare with the results as well.

Then we decide to apply the methods we have learned in this course, including the QR, Cholesky, and the LU method. From the course material, we know our methods use linear algebra matrix transformation and they can solve the coefficients faster. Because our methods do not use large matrix inverse, they will save time. In the QR method, the matrix is divided into an orthogonal matrix and an upper triangular matrix; in the Cholesky method, the matrix can be divided into a lower triangular matrix and an upper triangular matrix(transpose of each other); in the LU method, the matrix is divided into a lower triangular matrix and an upper triangular matrix.

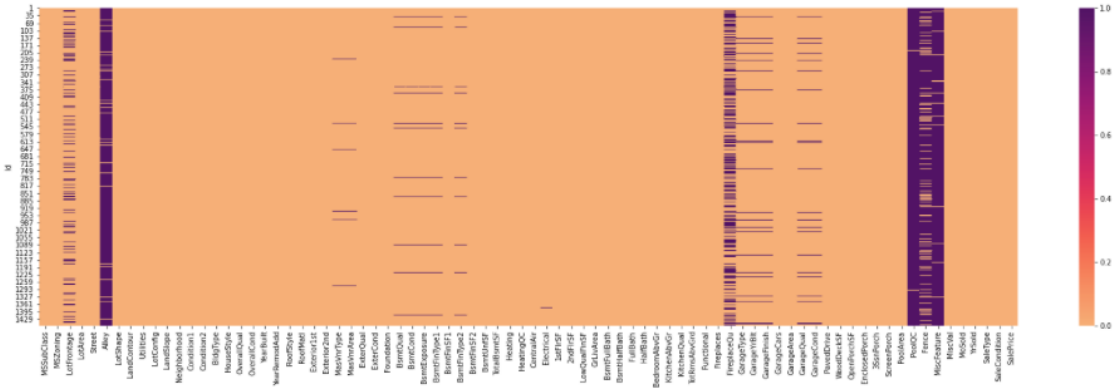
These methods are similar because all of them are matrix decompositions and we can use time measurement to compare their executing time to find out which is better.

However, there are some shortcomings in using these methods. For instance, LU is not stable and requires the use of backward and forward substitution. The drawbacks of QR decomposition are dependent on the process, but in most cases, they are also not very stable. As for Cholesky decomposition, it only serves for the positive definite matrix.

Besides, we know gradient descent is also appropriate for approximating linear regression. However, we notice that when we need to use the method, it requires the use of matrix inversion. Our X matrix contains 1460 rows which are really large so it will need a large amount of time. Therefore, in view of saving time and cost, we will avoid using that method in this condition. As for the power method, it will be appropriate if the number of coefficients can be the same as the number of formulas. In this case, the number of formulas exceeds the number of coefficients so we decide not to use it at last.

Although we have already chosen these variables, we also need to make verification that these variables are suitable to use. Therefore, we need to apply the AIC method to decide whether they are appropriate or not. If the AIC method tells us we need to reduce any variable in order to get a more appropriate model, we will repeat our process again. After the model selection, categorical variables also should have effects on the sale price result. Therefore, we may select some categorical variables and decide which will make the House sell at a higher price. Finally, we will use the model from the training dataset to predict the test dataset. Moreover, besides the numeric response variables in the model, we will give some predictions based on the categorical variable.

Simulation Study



From the plot, we can see the original data set has a lot of missing values. In order to run all the methods in a proper time, we decided to choose some features that we are interested in.

```
Call:
lm(formula = y ~ X)

Residuals:
    Min      1Q  Median      3Q     Max
-660129  -19206    -174   18972  277342

Coefficients: (1 not defined because of singularities)
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.274e+04  4.227e+03  -5.379 8.72e-08 ***
X1                 NA         NA      NA       NA
X2           2.168e-01  1.273e-01   1.703   0.0887 .
X3           5.333e+01  4.869e+00  10.952  < 2e-16 ***
X4           6.214e+01  5.671e+00  10.957  < 2e-16 ***
X5           7.150e+01  2.956e+00  24.189  < 2e-16 ***
X6           1.013e+02  6.833e+00  14.830  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 45950 on 1454 degrees of freedom
Multiple R-squared:  0.6666,    Adjusted R-squared:  0.6654
F-statistic: 581.4 on 5 and 1454 DF,  p-value: < 2.2e-16
```

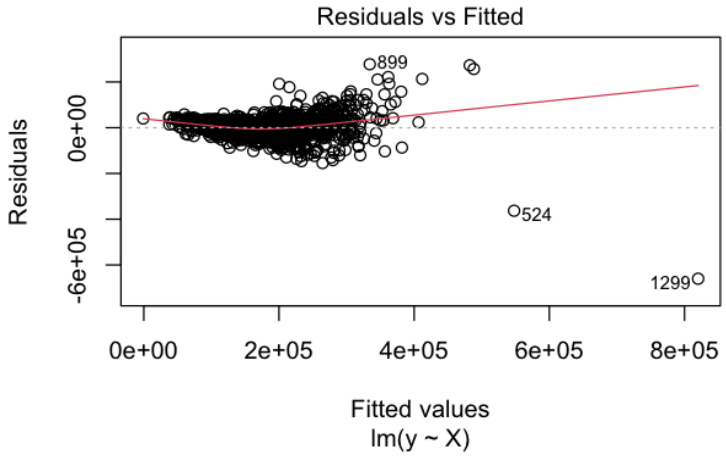
R-squared is 0.6666 tells us that approximately 66.66% of the variation in the sale price can be explained by the model:

$$\text{SalePrice} = \text{LotArea} + \text{TotalBsmtSF} + \text{1stFlrSF} + \text{2ndFlrSF} + \text{GarageArea}$$

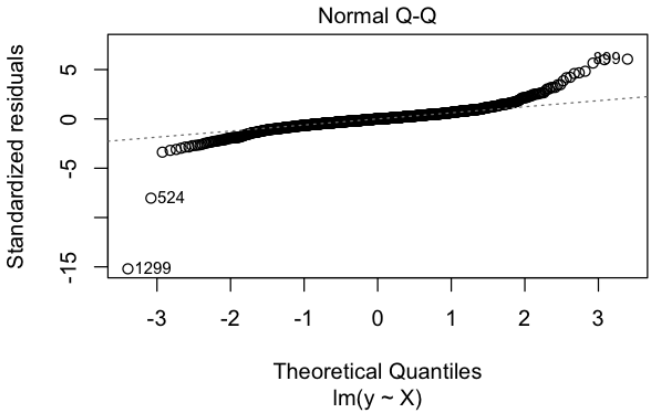
F-Statistics and p-value show the overall significance of the model.

The residual standard error gives an idea of how far observed Sale Prices are from the predicted or fitted sale price.

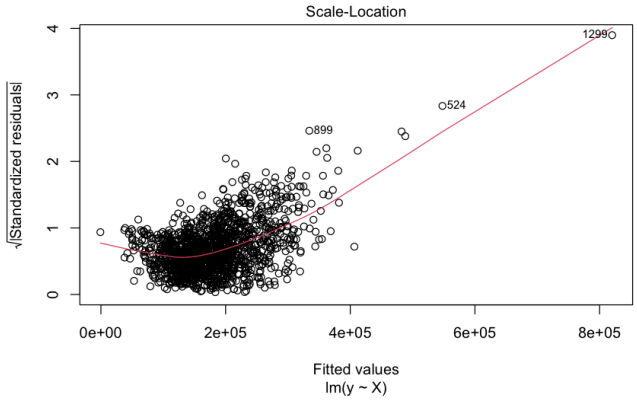
The slope for X5 which is "2ndFlrSF" (7.150e+01 is the effect of the second-floor square fee on the sale price adjusting or controlling for the other variables. If we associate an increase of 1 square foot in "2ndFlrSF" with an increase of $71.5 in sale price adjusting or controlling for the other variables.



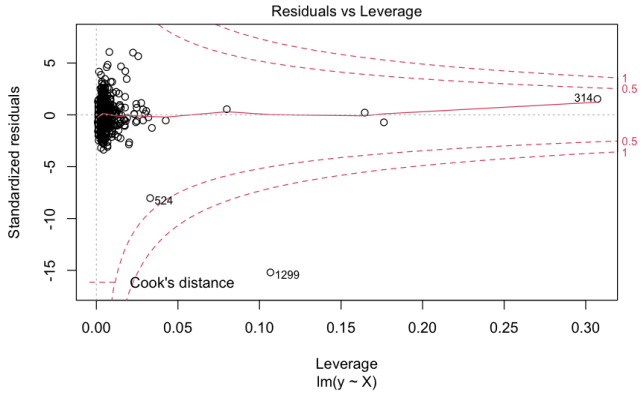
From the Residuals vs.Fitted plot, we can see that the relationship between predictor variables and an outcome is approximately linear. There are 3 extreme outliers.



From the Normal Q-Q plot, we can see that most of the quantile points lie on the theoretical normal line. Although two observations numbered as 524 and 1299 look a little off.



The distribution of residuals around the linear model in relation to the sale price. Most of the houses in the data are in the lower and median price range, the higher price, the fewer observations. It indicates that the sale price is right-skewed.



The Residuals vs. Leverage plot helps us to find the influential case if any. Not all outliers are influential in linear regression analysis. It looks like none of the outliers in our model are influential.

In addition, we also calculate the SSE/SSE to verify the R squared in the summary table, and the values are both 0.66.

AIC verification:

We will make use of MASS library and AICstep function here:

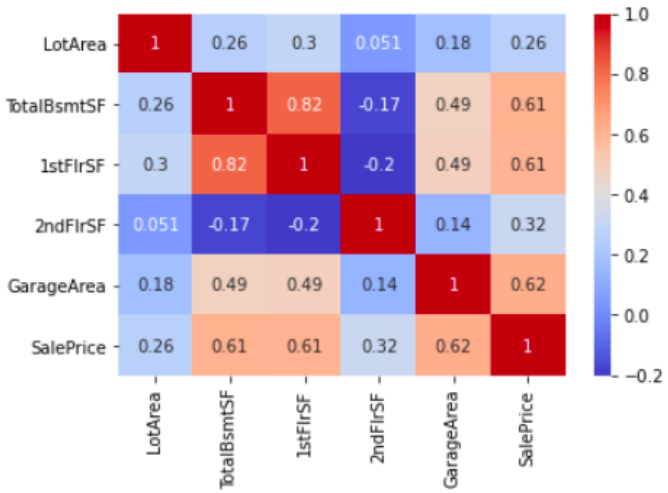
```
> stepAIC(model1,direction = 'both')
Start:  AIC=31353.16
data$SalePrice ~ data$LotArea + data$TotalBsmtSF + data$X1stFlrSF +
    data$X2ndFlrSF + data$GarageArea

                  Df  Sum of Sq        RSS   AIC
<none>                          3.0701e+12 31353
- data$LotArea      1 6.1251e+09 3.0763e+12 31354
- data$TotalBsmtSF  1 2.5326e+11 3.3234e+12 31467
- data$X1stFlrSF    1 2.5348e+11 3.3236e+12 31467
- data$GarageArea   1 4.6440e+11 3.5345e+12 31557
- data$X2ndFlrSF    1 1.2355e+12 4.3056e+12 31845

Call:
lm(formula = data$SalePrice ~ data$LotArea + data$TotalBsmtSF +
    data$X1stFlrSF + data$X2ndFlrSF + data$GarageArea)

Coefficients:
     (Intercept)      data$LotArea  data$TotalBsmtSF    data$X1stFlrSF
      -2.274e+04         2.168e-01         5.333e+01         6.214e+01
  data$X2ndFlrSF   data$GarageArea
       7.150e+01         1.013e+02
```

```
> AIC_value_estimated
[1] 35500.46
> AIC_value_verify
[1] 35498.46
```

In the first step, we can already make sure our chosen variables are appropriate so that the full model is fine to be used and we do not need to make a reduction of variables. Our estimated AIC value is also very close to the function value in R. Therefore, we assume our AIC part is correct.

Real Data Study

Visualization is generally easier to understand than reading the tabular data, so we choose using the Python Seaborn library to draw a correlation heatmap to visualize matrices. From the plot, we can see that “1stFlrSF” has a highly positive correlation to “TotalBsmtSF”, the variable “2ndFlrSF” has a negative correlation to “1stFlrSF”. However, the correlation value between the variable “2ndFlrSF” and “LotArea” is close to 0, which means they are not correlated. So should we keep them? We will use AIC to select the best model, where we can determine if keep these features.



For those methods that we used, we can conclude something. For the linear regression method, as we can see from the code since the p-value is small enough, which we got 2.2*10^-16 this can show that all of the coefficients reject the null hypothesis. And our null hypothesis that we assume is that beta = 0(the coefficients are all equal to 0). Hence, we can see that there is a relationship between each variable x and y. This just confirms our thought. Besides, the F-statistics is also large enough(which means the p-value is small enough) to reject the same null hypothesis.

```
> XX <- crossprod(X)
> XX.lu <- expand(lu(XX))
> L <- XX.lu$L
> U <- XX.lu$U
> P <- XX.lu$P
> bs <- forwardsolve(L, crossprod(P, crossprod(X, y)))
> LU_betahat <- backsolve(U, bs)
> LU_betahat
[1] -2.273524e+04  2.168051e-01  5.332838e+01  6.213760e+01  7.149911e+01  1.013422e+02
```

```
> cholR <- chol(yXXy)
> # regression coefficients
> Cho_betahat <- backsolve(cholR[1:6, 1:6], cholR[1:6, 7])
> Cho_betahat
[1] -2.273524e+04  2.168051e-01  5.332838e+01  6.213760e+01  7.149911e+01  1.013422e+02
```

```
> QR_betahat <- forwardsolve(x.qr$qr, yXXy[order(x.qr$pivot), 7],
+                            upper.tri = TRUE, transpose = TRUE)
> QR_betahat <- backsolve(x.qr$qr, QR_betahat)[x.qr$pivot]
> QR_betahat
[1] -2.273524e+04  2.168051e-01  5.332838e+01  6.213760e+01  7.149911e+01  1.013422e+02
```

```
> Xsvd <- svd(X)
> V = Xsvd$v
> D = Xsvd$d
> U = Xsvd$u
> SVD_betahat <- as.numeric(V %*% (crossprod(U, y) / D))
> SVD_betahat
[1] -2.273524e+04  2.168051e-01  5.332838e+01  6.213760e+01  7.149911e+01  1.013422e+02
```

We have implemented some functions to get the linear regression coefficients using the LU, Cholesky, QR, and SVD methods. Looking at the results, we see that our results are the same and this means that we did not make a mistake.

```
> #LU Time difference of 0.003777027 secs
> #LU Time difference of 0.003607988 secs
> #LU Time difference of 0.003432035 secs
> #LU Time difference of 0.004177094secs
> #LU Time difference of 0.003850222 secs
> (0.003777027+0.003607988+0.003432035+0.004177094+0.003850222)/5
[1] 0.003768873
```

```
> #Cho Time difference of 0.003640175 secs
> #Cho Time difference of 0.004565001 secs
> #Cho Time difference of 0.003637075 secs
> #Cho Time difference of 0.004323959 secs
> #Cho Time difference of 0.004182816 secs
> (0.003640175+0.004565001+0.003637075+0.004323959+0.004182816)/5
[1] 0.004069805
```

```
> #QR Time difference of 0.002748966 secs
> #QR Time difference of 0.004303932 secs
> #QR Time difference of 0.003558159 secs
> #QR Time difference of 0.003268957 secs
> #QR Time difference of 0.003854036 secs
> (0.002748966+0.004303932+0.003558159 +0.003268957+0.003854036)/5
[1] 0.00354681
```

```
> #SVD Time difference of 0.003331184 secs
> #SVD Time difference of 0.002810001 secs
> #SVD Time difference of 0.002781868 secs
> #SVD Time difference of 0.002848864 secs
> #SVD Time difference of 0.003234863 secs
> (0.003331184+0.002810001+0.002781868+0.002848864+0.003234863)/5
[1] 0.003001356
```

We compare the run time for these 4 methods each 5 times and get their average down as their runtime. We want to run multiple times for better accuracy. From the result, we see that the SVD method seems to be a little bit faster than the other three methods and the Cholesky method seems to be the slowest. This was a little bit unexpected because usually the Cholesky method is the fastest in computing compared to the other three and the SVD method is usually the slowest in computing time. We were expecting the computing time in this order: Cholesky, LU, QR, SVD, from fastest to slowest.

Cholesky decomposition has $\frac{n^3}{3} + O(n^2)$

LU decomposition has $2\sum_{i=1}^{n-1} i^2 = \frac{2n^3}{3} - n^2 + \frac{n}{3} \approx \frac{2n^3}{3}$

QR decomposition has $\frac{2n^3}{3} + n^2 + \frac{n}{3} - 2$

SVD decomposition is other than those three algorithms, it needs to find all eigenvectors for the matrix

Theoretically, Cholesky supposed to be the fast algorithm because it has $\frac{n^3}{3}$ which is less than LU $\frac{2n^3}{3}$. In our experiment, Cholesky has the shortest computing time, LU and QR have similar run time.

```
> predict(model,newdata = testX)
        1         2         3         4         5         6         7
194531.4065 171681.5691 209466.8047 198480.7548 272575.5881 161348.6356 239077.4071
        8         9        10        11        12        13        14
226669.7215 194060.3327 119419.2269 138693.0021 272197.2759 121053.7173 237207.7892
       15        16        17        18        19        20        21
159983.6015 134399.4348 144273.9659 112427.2036 167236.1144 146775.5646 287587.0602
       22        23        24        25        26        27        28
110074.1960 239795.1370 157475.6533 128808.8697 253476.8569 141118.2442 254743.4583
       29        30        31        32        33        34        35
191690.8781  63000.2723 127141.9803 148373.1054 171194.8848 205039.4486 215438.9158
       36        37        38        39        40        41        42
274487.6996 174387.1836 179342.3340 125960.0143  50156.7776 164023.8872 166655.2972
       43        44        45        46        47        48        49
130056.9759 118779.8548 142175.8158 239594.5606 263857.6222 254590.7497 114407.1708
```

```
       946         947         948         949         950         951         952
197340.7721 191796.0099 249701.5493 196472.9450 195527.6567 151554.7441 113477.1830
       953         954         955         956         957         958         959
107635.1397 204962.6963  90592.8990 211711.0670 136904.5210 159294.1010 187287.4877
       960         961         962         963         964         965         966
158715.6872  77897.0886 304483.2003 168385.6993 258830.7178 202677.3645 159842.4168
       967         968         969         970         971         972         973
128496.7558 131997.5700  74140.5929 105667.7307  96489.3437 175640.9173 111727.8976
       974         975         976         977         978         979         980
193317.8322 206198.5111 155898.6601  56005.9731 179148.3781 123077.0342 157257.7338
       981         982         983         984         985         986         987
151899.5367 288555.4143 163455.3400 271780.3894 145874.1233 138348.1312 137652.6935
       988         989         990         991         992         993         994
264351.3774 215615.4371 187712.6053 299782.3600 188890.1191 190603.3866 176563.4344
       995         996         997         998         999        1000
257030.8972 129730.1133 138108.8953 208632.4626 121292.1530 181142.0737
```

Here, we implemented the test data. As we mentioned before, the test data set does not contain slesprice and so we would use the coefficient that we found previously with the 4 methods to predict the sales price. The result is very long, so we would just show the first and last 50 of the predicted sales price.

As for the AIC part, the first step shows we do not need to reduce any variables but we can get interaction variables.

```
> step(model1,scope=~data$LotArea*data$TotalBsmtSF*data$X1stFlrSF*data$X2ndFlrSF*dat
a$GarageArea,direction = "both")
Start:  AIC=31353.16
data$SalePrice ~ data$LotArea + data$TotalBsmtSF + data$X1stFlrSF +
    data$X2ndFlrSF + data$GarageArea

                                    Df  Sum of Sq        RSS   AIC
+ data$LotArea:data$TotalBsmtSF     1 3.2472e+11 2.7454e+12 31192
+ data$LotArea:data$X1stFlrSF       1 3.0107e+11 2.7691e+12 31204
+ data$TotalBsmtSF:data$X1stFlrSF   1 2.6907e+11 2.8011e+12 31221
+ data$LotArea:data$GarageArea      1 1.4212e+11 2.9280e+12 31286
+ data$TotalBsmtSF:data$GarageArea  1 6.3791e+10 3.0063e+12 31325
+ data$X1stFlrSF:data$GarageArea    1 3.7718e+10 3.0324e+12 31337
+ data$TotalBsmtSF:data$X2ndFlrSF   1 2.1304e+10 3.0488e+12 31345
+ data$X1stFlrSF:data$X2ndFlrSF     1 1.8971e+10 3.0512e+12 31346
+ data$X2ndFlrSF:data$GarageArea    1 1.4457e+10 3.0557e+12 31348
<none>                                          3.0701e+12 31353
- data$LotArea                      1 6.1251e+09 3.0763e+12 31354
```

Repeating the steps in several times, we will get a new model with these interaction variables:

```
Step:  AIC=30946.25
data$SalePrice ~ data$LotArea + data$TotalBsmtSF + data$X1stFlrSF +
    data$X2ndFlrSF + data$GarageArea + data$LotArea:data$TotalBsmtSF +
    data$X1stFlrSF:data$GarageArea + data$TotalBsmtSF:data$X1stFlrSF +
    data$X2ndFlrSF:data$GarageArea + data$LotArea:data$GarageArea +
    data$X1stFlrSF:data$X2ndFlrSF + data$TotalBsmtSF:data$X2ndFlrSF +
    data$TotalBsmtSF:data$GarageArea + data$TotalBsmtSF:data$X1stFlrSF:data$X2ndFlrS
F +
    data$LotArea:data$TotalBsmtSF:data$GarageArea + data$TotalBsmtSF:data$X2ndFlrSF:
data$GarageArea

                                                Df  Sum of Sq        RSS
<none>                                                        2.2886e+12
- data$TotalBsmtSF:data$X2ndFlrSF:data$GarageArea  1 3.1694e+09 2.2918e+12
```

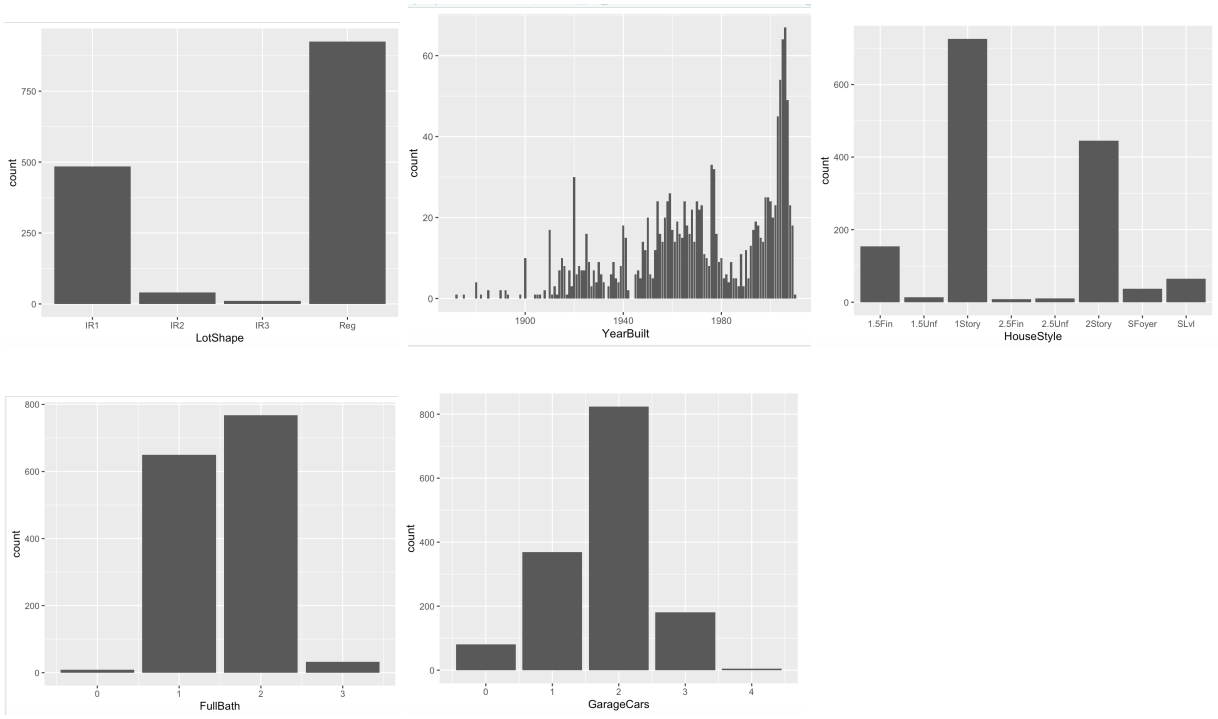
LotArea:TotalBsmtSF, X1stFlrSF:GarageArea, TotalBsmtSF:X1stFlrSF, X2ndFlrSF:GarageArea,LotArea:GarageArea ,X1stFlrSF:X2ndFlrSF ,TotalBsmtSF:X2ndFlrSF ,TotalBsmtSF:GarageArea,TotalBsmtSF:X1stFlrSF:X2ndFlrSF ,LotArea:TotalBsmtSF:GarageArea, TotalBsmtSF:X2ndFlrSF:GarageArea. The AIC value also drops from 31353.16 to 30946.25. With the new model, we find out the R squared increases from 0.6666 to 0.7515, which confirms that our interaction variables make sense. Therefore, we apply the predict model strategy again.

The first few predicted prices will be :

```
> predict(interactions_model,newdata = testX)
       1        2        3        4        5        6        7
187875.53 165628.06 206844.76 184632.00 294224.44 157604.27 245559.44
       8        9       10       11       12       13       14
227060.57 187081.02 121121.97 136063.86 291799.38 122811.78 241825.84
      15       16       17       18       19       20       21
155275.61 120445.14 138492.93  98750.21 160065.56 142035.16 316673.26
      22       23       24       25       26       27       28
114145.19 243575.45 145220.09 127778.96 262303.73 128446.39 267905.95
      29       30       31       32       33       34       35
187701.10  83618.14 129462.00 143212.51 165509.22 200429.07 214409.51
      36       37       38       39       40       41       42
292327.87 163252.76 172989.14 125558.10  97160.86 157329.04 164874.39
      43       44       45       46       47       48       49
122249.49 120086.95 138017.43 245107.31 276209.47 267776.59 123092.95
      50       51       52       53       54       55       56
124974.49 157887.36 115405.17 116812.66 270731.87 122783.30 199189.84
```

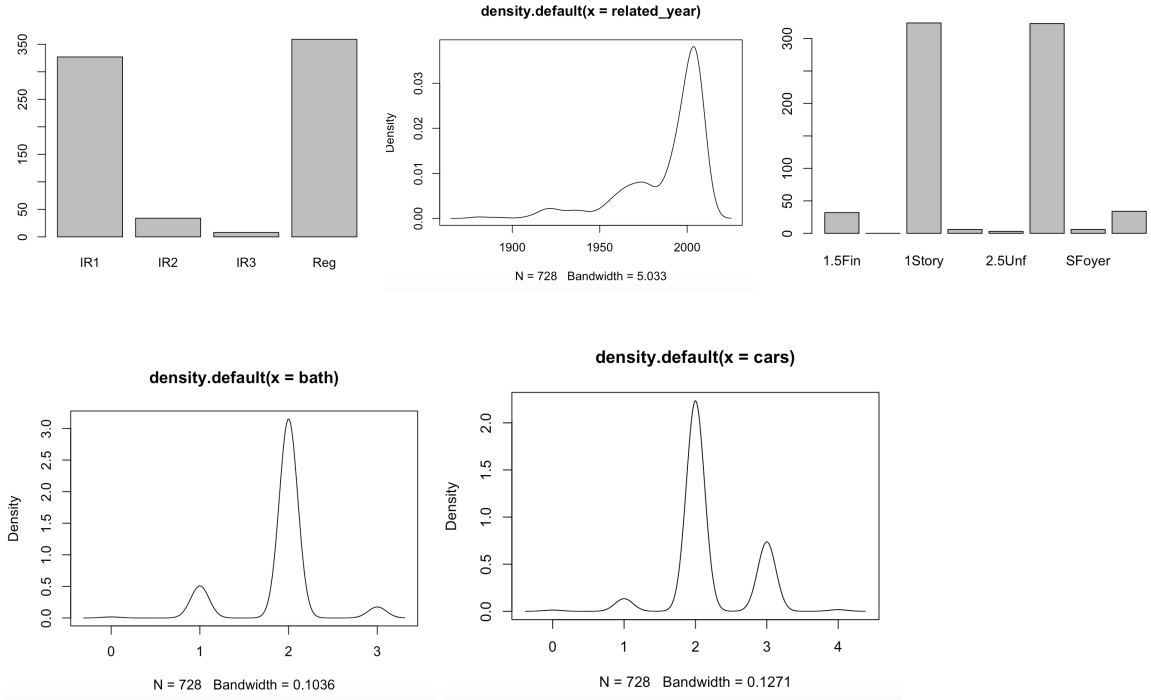
Besides the numeric model prediction, there are some categorical variables that can also show a trend for a high sale price. For instance, the lot shape, the built year, and the house style. There are some other numeric variables but the range of their values are small enough to be categorical variables such as the number of full baths and the number of garage cars.

Here are the density plots of mentioned categorical variables in our train data:



These charts show the constructors are more willing to build a house with either Reg or IR1 lot shape, house style with 1 to 2 stories, the number of full baths will be 1to 2, and the garages are more likely to be 2.

If we use the median values(the mean price may be affected by extreme values) of house price as the scale to decide it is a high price house or not, there are some interesting findings:



Although the number of Lotshape IR1 and Reg differs a lot in the whole train dataset, in the high sale price dataset, the number difference gets small. Meanwhile, the 1 story and 2 story HouseStyle differences also decrease. On the contrary, the differences of FullBath as well as GarageCar increase, which means people will be less likely to spend more money for buying a house with less full baths or garages. The house year built of high sale price corresponds to the distribution of the whole dataset, which means the year factor influences less.

Conclusion

All in all, depending on these things that we find, we can realize that the total garage area, first-floor area, and total basement area are all highly related to the selling price. Furthermore, we also use the lm function to confirm our thought that those variables are related. Besides, we learn from the class that QR, LU, Cholesky can achieve a faster speed to find the coefficient of the model, and here we discover that LU is faster compared with others. Then we use those methods to find that LU hat, QR hat, Chor hat, and SVD hat are all -2.273524e+04, 2.168051e-01, 5.332838e+01, 6.213760e+01, 7.149911e+01, 1.013422e+02. The interaction factors increase the accuracy of the fitted model, and for the test dataset to make some predictions, besides applying the numeric models, the category data shows that a house with 2 full baths

and 2 garages are likely to sell in a higher price, while the lot shape, years, and story depend on the

requirement of different customers.

Reference and acknowledgment

Data Source:
https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data?select=train.csv

https://math.stackexchange.com/questions/2840755/what-is-the-computation-time-of-lu-cholesky-and-qr-decomposition

Appendix

Using matrix transformation and multiplication to increase the speed of the calculations for computers. And during the project, we also find the time consumption of LU, QR, and Cholesky to find the better and quicker one. Besides, we also use the different symbols of the calculation sign to increase the speed.

```
# visualizing the null values in all columns via python
plt.figure(figsize=(30,8))
sns.heatmap(train_df.isnull(),cmap='flare')

library(MASS)
library(dplyr)
library(tidyr)
library(Matrix)
library(ggplot2)
data <- read.csv("~/Desktop/assignment/STA/STA141C/project/train.csv", header=TRUE)
data1 <- cbind(1, data$LotArea, data$TotalBsmtSF, data$X1stFlrSF,data$X2ndFlrSF, data$GarageArea, data$SalePrice)
testdata <- read.csv("~/Desktop/assignment/STA/STA141C/project/test.csv",sep = ",")
data2 <-cbind(1, testdata$LotArea, testdata$TotalBsmtSF, testdata$X1stFlrSF,testdata$X2ndFlrSF, testdata$GarageArea)
testX <-  data2[, 1:6] # X and y
testX <- data.frame(testX)
X <-  data1[, 1:6] # X and y
colnames(X) <-c('intercept','Lotarea','TotalBsmtSF','1stSF','2ndSF','Garagearea')
y <- as.matrix(data1[,7])

start_time <- Sys.time()
XX <- crossprod(X)
XX.lu <- expand(lu(XX))
L <- XX.lu$L
U <- XX.lu$U
P <- XX.lu$P
bs <- forwardsolve(L, crossprod(P, crossprod(X, y)))
betahat <- backsolve(U, bs)
sigma2hat <- crossprod(y - X %*% betahat) / (dim(X)[1] - 6)
```

```
betacov <- as.numeric(sigma2hat) * chol2inv(XX)
betase <- sqrt(diag(betacov))
end_time <- Sys.time()
end_time - start_time

#Cholesky
start_time <- Sys.time()
yXXy <- crossprod(data1)
# Cholesky decomposition of Grammian matrix
cholR <- chol(yXXy)
# regression coefficients
betahat <- backsolve(cholR[1:6, 1:6], cholR[1:6, 7])
# s.e. of regression coefficients
sigma2hat <- cholR[7, 7]^2 / (dim(data1)[1] - 6)
betacov <- sigma2hat * chol2inv(cholR[1:6, 1:6])
betase <- sqrt(diag(betacov))
end_time <- Sys.time()
end_time - start_time

#QR
start_time <- Sys.time()
x.qr <- qr(data1[, 1:6])
x.qr$rank
x.qr$pivot
# regression coefficients
betahat <- forwardsolve(x.qr$qr, yXXy[order(x.qr$pivot), 7],
                        upper.tri = TRUE, transpose = TRUE)
betahat <- backsolve(x.qr$qr, betahat)[x.qr$pivot]
# s.e. of regression coefficients
sigma2hat <- crossprod(Xy[, 7] - Xy[, 1:6] %*% betahat) / (dim(Xy)[1] - 6)
betacov <- as.numeric(sigma2hat) * chol2inv(x.qr$qr)
betase <- sqrt(diag(betacov))
betase <- betase[x.qr$pivot]
end_time <- Sys.time()
end_time - start_time

#SVD
start_time <- Sys.time()
Xsvd <- svd(X)
V <- Xsvd$v
D <- Xsvd$d
U <- Xsvd$u
SVD_betahat <- as.numeric(V %*% (crossprod(U, y) / D))
SVD_betahat
```

```
end_time <- Sys.time()
end_time - start_time

model <- lm(y~X)
#predicts the future values
predict(model,newdata = testX)

# residuals(MSE SSE RSS)
model1 <- lm(data$SalePrice~data$LotArea+ data$TotalBsmtSF+
data$X1stFlrSF+data$X2ndFlrSF+data$GarageArea) #ignore NA in model
predicted_train_y <-fitted(model1)
model_MSE <- mean(model1$residuals^2)
model_MSR <- mean((predicted_train_y - mean(y))^2)
model_SSE <- sum(model1$residuals^2)
model_SSR <- sum((predicted_train_y - mean(y))^2)
R_2 <- 1-model_SSE/(model_SSR+model_SSE)

# corr scatterplot
pairs(data$SalePrice~data$LotArea+ data$TotalBsmtSF+
data$X1stFlrSF+data$X2ndFlrSF+data$GarageArea,
main="Simple Scatterplot Matrix")

# AIC/BIC
AIC_value_estimated <-
nrow(data1)*(log(2*pi)+1+log((sum(model$residuals^2)/nrow(data1))))+((length(model$coefficients)+1)
*2)
model_AIC <-
step(model1,scope=~data$LotArea*data$TotalBsmtSF*data$X1stFlrSF*data$X2ndFlrSF*data$GarageA
rea,direction = "both")
interactions_model = lm(data$SalePrice~data$LotArea+ data$TotalBsmtSF+
data$X1stFlrSF+data$X2ndFlrSF+data$GarageArea+data$LotArea:data$TotalBsmtSF+data$X1stFlrSF:d
ata$GarageArea+data$TotalBsmtSF:data$X1stFlrSF+data$X2ndFlrSF:data$GarageArea+data$LotArea:d
ata$GarageArea+data$X1stFlrSF:data$X2ndFlrSF+data$TotalBsmtSF:data$X2ndFlrSF+data$TotalBsmt
SF:data$GarageArea+data$TotalBsmtSF:data$X1stFlrSF:data$X2ndFlrSF+data$LotArea:data$TotalBsm
tSF:data$GarageArea+data$TotalBsmtSF:data$X2ndFlrSF:data$GarageArea)
interactions_fitted = fitted(interactions_model)
summary(interactions_model)

#verification of AIC
AIC_value_verify <- AIC(model)
stepAIC(model1,direction = 'both')
# The chosen model is fine as all of the variables are related
summary(model1)
# The small enough P-values show that all of the coefficients reject the null
```

```
# hypothesis Beta = 0. Hence there is a relationship between each variable x and y.

#correlation table
cor(data1)

# PCA (verification)
X_cov <- cov(X[,2:6])
X_eigen <-eigen(X_cov)
train_PCA <-  prcomp(data1[,2:6], center = TRUE,scale. = TRUE)
summary(train_PCA)

#plot
ggplot(data, aes(x=LotShape)) + geom_bar() #lotshape
ggplot(data, aes(x=YearBuilt)) + geom_bar()
ggplot(data, aes(x=HouseStyle)) + geom_bar()
ggplot(data, aes(x=FullBath)) + geom_bar()
ggplot(data, aes(x=GarageCars)) + geom_bar()

#high y
high_y <- y[y>median(y)]
related_Lotshape <- data$LotShape[y>median(y)]
related_year <- data$YearBuilt[y>median(y)]
related_style <- data$HouseStyle[y>median(y)]
bath <- data$FullBath[y>median(y)]
cars <- data$GarageCars[y>median(y)]
plot(related_Lotshape)
plot(density(related_year))
plot(related_style)
plot(density(bath))
plot(density(cars))
```