STA 141C Final Report
Group 15

Lynn Waiyan Kyaw, Tracy Zhu, Yucheng Zhao

Table of Contents

I. Abstract 3

II. Introduction 3

2.2 Dataset Description 4

2.3 Data Cleaning and Preparation 4

III. Proposed methods 7

IV. Data Analysis Study 9

4.1 Model Building 9

4.2 House Price Predictions 12

4.3 Computation Speeds 13

V. Summary 14

VI. References 15

VII. Appendix 15

I. Abstract

In the United States, house prices are at a record high reflecting the increasingly competitive real estate market. Acknowledging the key factors that drive these unprecedented price levels is necessary for prospective homebuyers. House prices are not only determined by the amount of bedrooms or bathrooms, instead there are multiple factors that determine the price negotiations of a house. We will be analyzing the relationships between house prices and house features as well as predicting the prices of the residential homes in Ames, Iowa. In order to provide insight on the dynamics of house prices and their implications for future homeowners, we will be employing direct decomposition methods such as LU decomposition, QR decomposition, and Singular Value Decomposition (SVD) in order to perform linear regression on the dataset.
In addition, we will also utilize the LASSO (least absolute shrinkage and selection operator) regression analysis method in order to identify the best set of predictors and account for potential multicollinearity. In short, our regression analysis aims to analyze the most relevant factors influencing house prices. The dataset used was extracted from the Kaggle library: https://www.kaggle.com/c/house-price-advanced-regression- techniques [1]. After cleaning and preparing the dataset, the new dataset includes 7 independent numeric variables and the target variable that we are trying to predict being the property's sale price in dollars. After verifying our predictors with LASSO, it was concluded that the most efficient direct decomposition method, when computing house prices, was SVD decomposition when inputting it into the linear regression model. If our model can continue to closely predict house prices in Ames, Iowa, then we will be able to use this model to predict house prices in locations all around the United States of America.

II. Introduction

One of the most important steps of our lives can be the process of owning a home. The trend of the housing market has a profound effect on individuals, families, and even economies as a whole. Unlike before, the cost of homeownership is not as affordable as it was decades ago. Between 1996 to early 2007, there was a significant real-term appreciation of 33 percent in the price of new houses [2]. Looking ahead to 2022, the housing inventory is at a record low compared to the abundance of new houses available in 2007, resulting in an increased demand for new homes [3]. With the demand for new houses at an all-time high, whether you're a beginning real-estate investor or a first-time homebuyer, understanding the factors that determine house prices is key to knowing your way around the housing market. House prices are not simply determined by the amount of bedrooms, bathrooms, or square footage; rather, there are multiple factors that are not obvious to newer homebuyers. For example, factors such as the slope of the property or the height of a basement ceiling are not commonly taken into account by first-time homebuyers when considering home prices. In this statistical study, in order for homebuyers to understand the logistics that drive home prices, we will be analyzing the relationships between house prices and house features with the intention to predict a property's value.

2.1 Dataset Description

The house prices dataset that was utilized in this study was obtained from the Kaggle library: https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview. In short, this dataset describes almost every aspect and physical feature of the residential homes in Ames, Iowa. The house prices dataset has 79 explanatory variables with 36 numeric variables and 43 non-numeric variables. An example of the variables that are included within the dataset are:

- **SalePrice**: the property's sale price in dollars (the response variable that we are trying to predict)
- **LotFrontage**: the linear feet of the street connected to the property
- **LotArea**: the lot size in square feet
- **YearBuilt**: the original construction date
- **OverallQual**: the overall material and finish quality of the home

2.2 Data Cleaning and Preparation

Before working on the house price dataset, our first process is to prepare a comprehensive data cleaning process in order to ensure the accuracy and the reliability of our dataset. We started with conducting a correlation analysis by producing a correlation matrix for our variables. By calculating correlation coefficients with our response variable and our other predictors, we were able to identify the predictors that possess a strong linear relationship with our response variable, SalePrice.

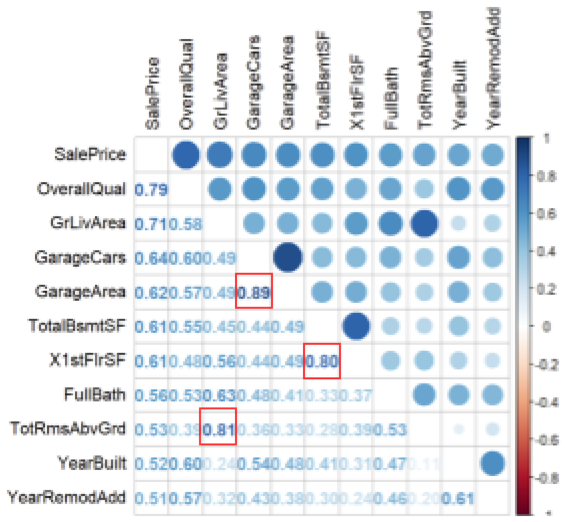


Figure 1: Correlation Plot

According to our correlation plot, a correlation coefficient greater than 0.5 indicates that there is a strong positive correlation between that predictor and the response variable, SalePrice. Furthermore, we searched for signs of multicollinearity which might cause issues for our statistical models. This is because high correlation coefficients can make it difficult for us to determine the unique contribution of each predictor variable from the house price dataset. With this in mind, we filtered and created a new dataset with 7 of the most highly-correlated predictor variables and updated the correlation plot. The updated correlation plot removed multicollinear variables, duplicated variables, and added only the high correlation coefficient variables (correlation coefficient > 0.5).

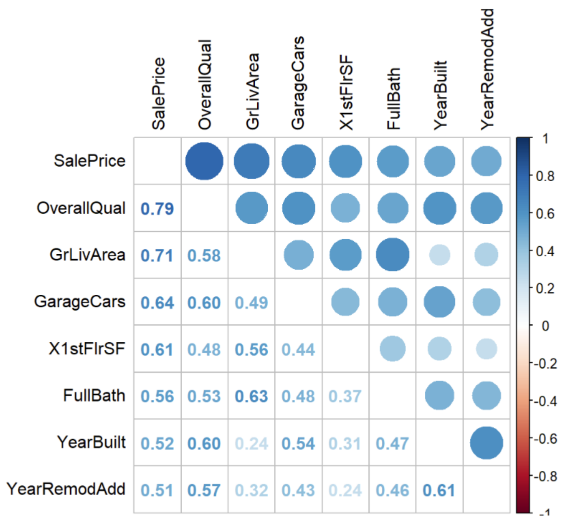


Figure 2: Updated Correlation Plot

Next, we constructed a box plot and scatter plot in order to assess the two most correlated predictors (correlation coefficient > 0.7 in the new dataset of 7 predictors) with SalePrice: Overall Quality and Above Grade Living Area.

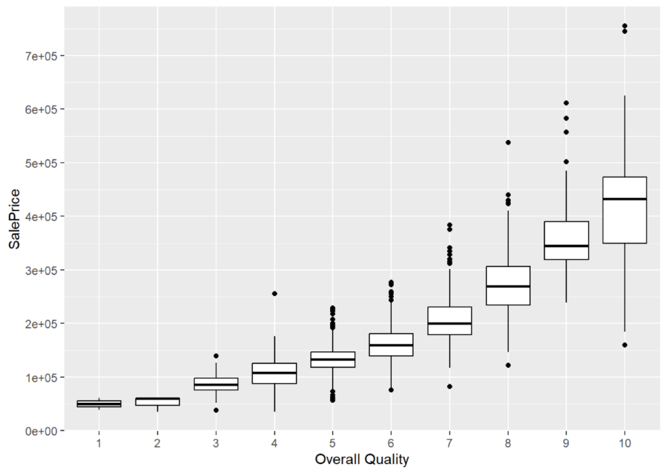


Figure 3:Boxplot of Overall Quality

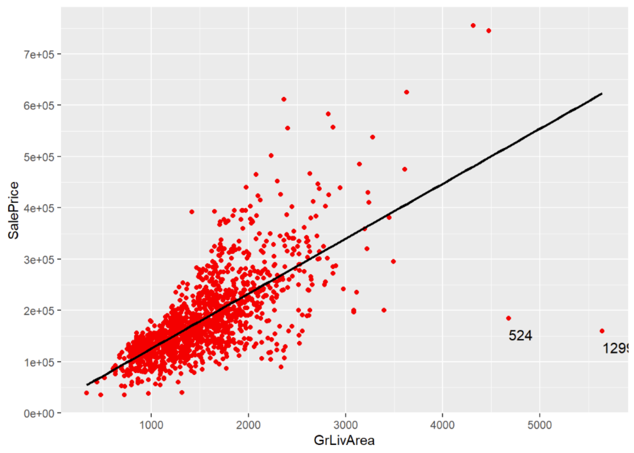


Figure 4: Scatter plot of Living Area

We noticed that for the scatter plot of the predictor variable, Above Grade Living Area, there are two influential points 524 and 1299. We decided to remove these unusual outliers because these two houses have the highest score on the predictor, Overall Quality, which is deviant.

```
 SalePrice OverallQual
    184750          10
    160000          10
```

After checking these two predictors, we will process the missing values in our new dataset by replacing them with interpolated values in order to ensure that our dataset is suitable for analysis. Interpolation is a technique that is used to estimate missing values based on the available data points. We then checked the correlation again using a heat map. According to the heat map, after replacing the missing values with interpolated values, we assessed that the correlations between the variables are still consistent when compared to the correlation matrix.

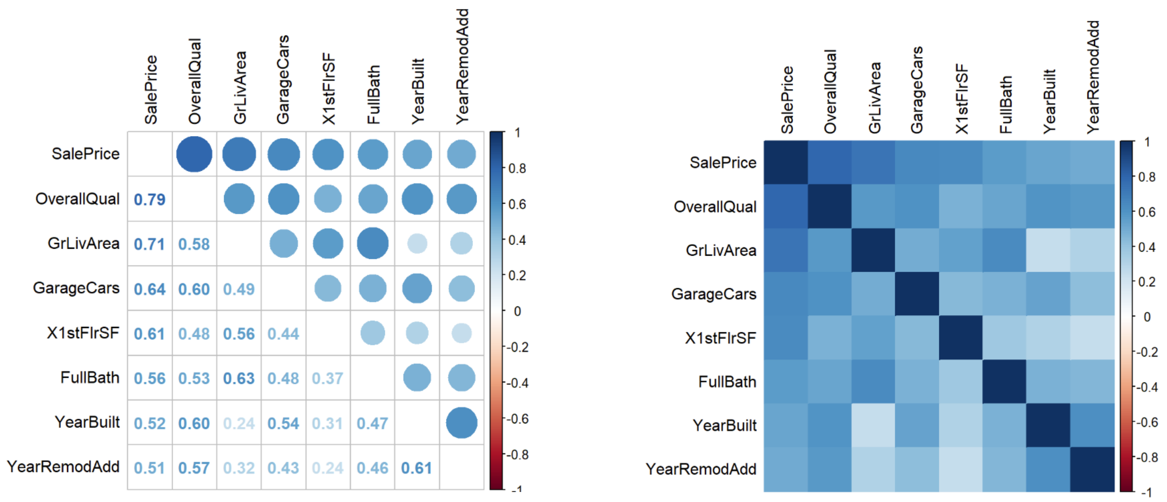


Figure 5: Correlation Matrix & Heatmap Correlation plot

For the last method of preparation for our new dataset, we will be normalizing the right-skewed distribution in order to address the skewness and achieve a more symmetrical and normally distributed variable. Since we will be performing linear regression, we need to make sure that the data follows a normal distribution. Normalizing the right-skewed distribution in this case ensures that these conditions are met for linear regression.

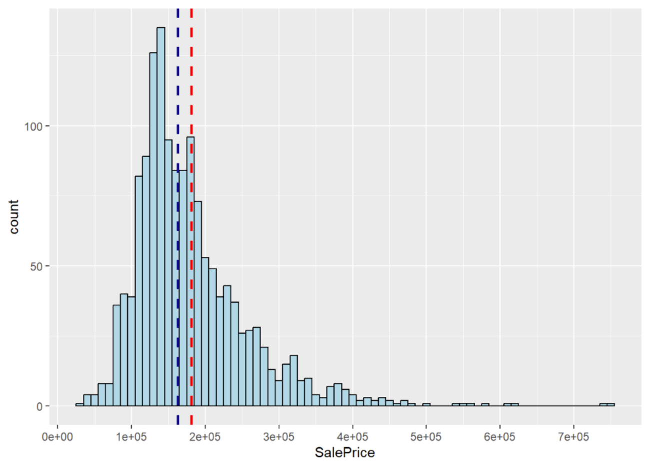


Figure 6: Before Normalizing the right-skewed Distribution

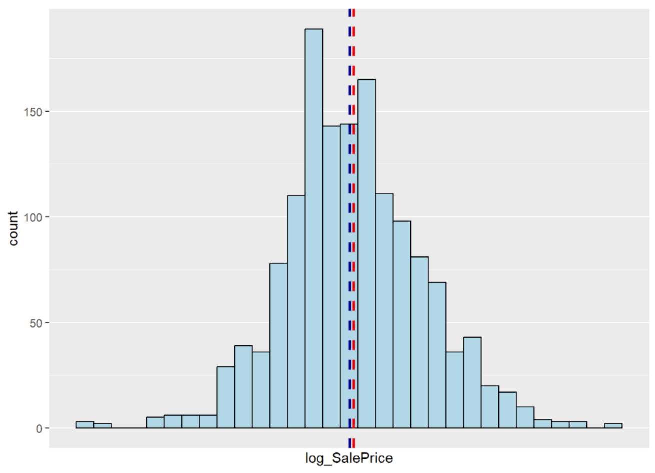
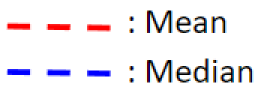


Figure 7: After



III. Proposed Methods

Linear Regression: We will be mainly utilizing linear regression in order to achieve our desired results. To predict house prices based on a house's features, this technique is highly recommended in order to establish a statistical relationship between the independent variables (house feature predictors) and the dependent variable (house prices). In the context of predicting house prices using linear regression, our approach involves estimating the regression coefficients using methods like QR, SVD, LU decomposition as well as the LASSO technique for our model.

LASSO (least absolute shrinkage and selection operator): this model selection technique, in the context of verifying our linear regression model, is going to be used to identify the most important predictors that should be included in our model for predicting house prices. LASSO can be a viable option to verify our data since it provides both predication and variable selection for our model as well as accounting for multicollinearity.

Algorithms: In order to estimate the regression coefficients, we implemented LU, QR, and Singular Value decomposition (SVD). We performed LU decomposition on X'X to obtain the permutation matrix, lower triangular matrix, an
d upper triangular matrix. Then we solved the regression coefficients by calculating beta = U^(-1)*L^(-1)*P^(-1)*X'Y

LU decomposition: factorizes a square matrix into the product of two matrices: a lower triangular matrix (L) and an upper triangular matrix (U). It is also known as LU factorization or LU factorization with pivoting. The decomposition can be represented as A = LU, where A is the original matrix. LU decomposition is useful for solving systems of linear equations, such as Ax = b, where A is a coefficient matrix, x is a vector of unknowns, and b is a vector of constants. Once A is decomposed into LU form, the system of equations can be solved more efficiently by solving two triangular systems, Ly = b and Ux = y.

QR decomposition: factorizes a matrix into the product of an orthogonal matrix (Q) and an upper triangular matrix (R). The decomposition can be represented as A = QR, where A is the original matrix. QR decomposition is commonly used in numerical algorithms, such as least squares regression and eigenvalue computations. The orthogonal matrix Q has the property that Q^T (transpose of Q) is equal to its inverse, making it useful for solving least squares problems. The upper triangular matrix R provides information about the structure and scaling of the original matrix.

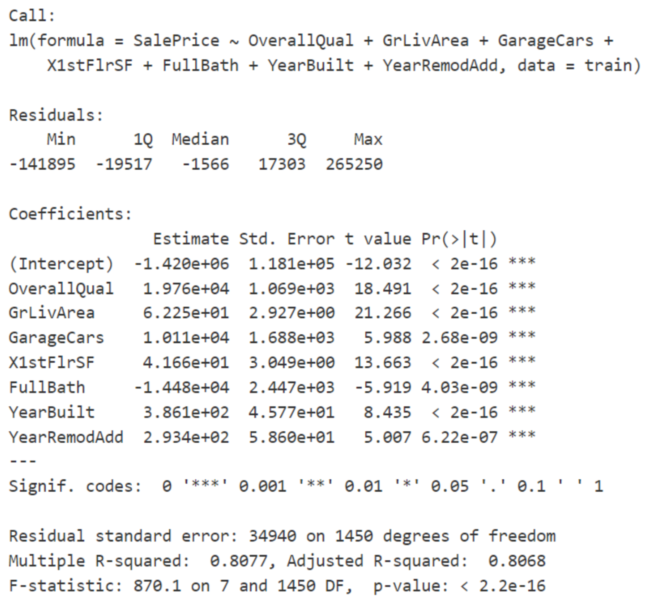
Singular Value Decomposition (SVD): decomposes a matrix into the product of three matrices: a left singular matrix (U), a diagonal matrix of singular values (Σ), and a right singular matrix (V^T, the transpose of V). The decomposition can be represented as A = UΣV^T, where A is the original matrix. SVD is a powerful decomposition technique that can be applied to any matrix, including rectangular or non-square matrices. It is widely used in applications such as image compression, data compression, dimensionality reduction, and collaborative filtering. In SVD, the singular values in Σ represent the magnitudes of the underlying components in the data, and the singular vectors in U and V capture the directions of these components. By selecting a subset of the singular values and corresponding singular vectors, it is possible to approximate the original matrix with lower rank, enabling data compression or noise reduction.

- **LU Decomposition** on X'X
    - Obtain P, L, U
    - Calculating beta = U^(-1)*L^(-1)*P^(-1)*X'Y
- **QR Decomposition** on X
    - Obtain Q and R
    - Solving for beta using qr.solve(R, t(Q) %*% Y)
- **Singular Value Decomposition (SVD)** on X'X
    - Obtain U, V, S^(-1), where S^(-1) = diag(1/D)

- Beta = V*S^(-1)*U'*X'*Y

IV. Data Analysis Study

4.1 Model Building

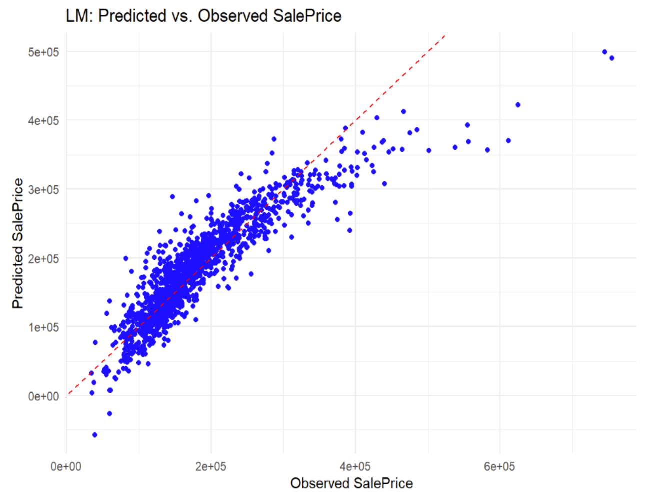


```
Call:
lm(formula = SalePrice ~ OverallQual + GrLivArea + GarageCars +
    X1stFlrSF + FullBath + YearBuilt + YearRemodAdd, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-141895  -19517   -1566   17303  265250

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.420e+06  1.181e+05 -12.032  < 2e-16 ***
OverallQual   1.976e+04  1.069e+03  18.491  < 2e-16 ***
GrLivArea     6.225e+01  2.927e+00  21.266  < 2e-16 ***
GarageCars    1.011e+04  1.688e+03   5.988 2.68e-09 ***
X1stFlrSF     4.166e+01  3.049e+00  13.663  < 2e-16 ***
FullBath     -1.448e+04  2.447e+03  -5.919 4.03e-09 ***
YearBuilt     3.861e+02  4.577e+01   8.435  < 2e-16 ***
YearRemodAdd  2.934e+02  5.860e+01   5.007 6.22e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 34940 on 1450 degrees of freedom
Multiple R-squared:  0.8077, Adjusted R-squared:  0.8068
F-statistic: 870.1 on 7 and 1450 DF,  p-value: < 2.2e-16
```



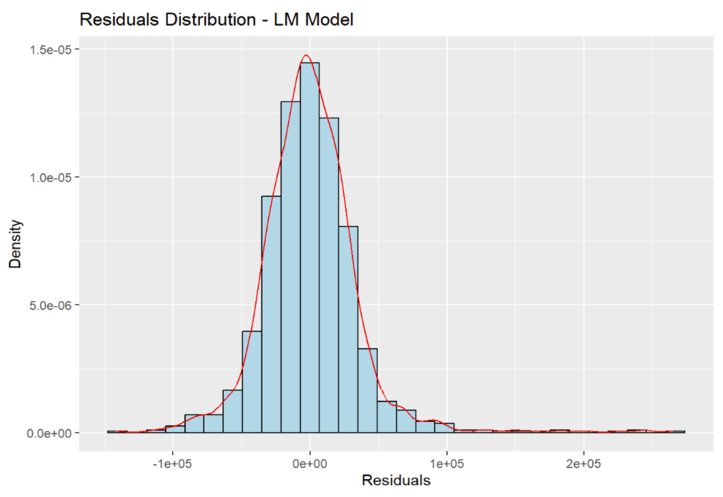
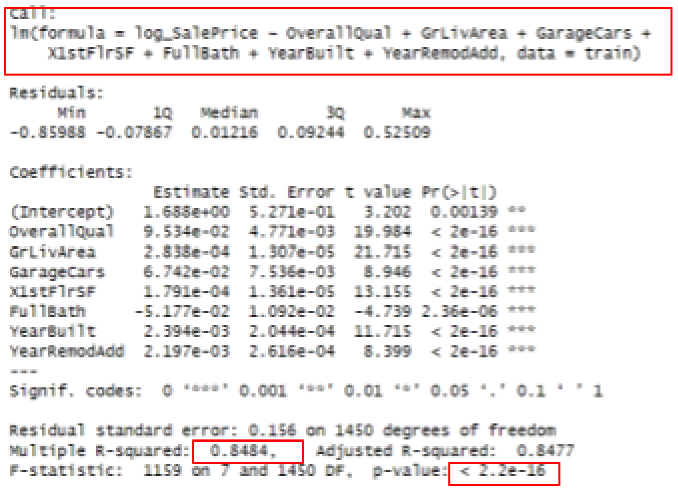


Figure 8: Linear regression results for the original data

For our model building segment, we decided to use the lm model. In the un-normalized data, R-squared is about 0.80, which means that approximately 80% of the variation in the sale price can be explained by the lm model. The F-Statistics and p-value proofs the significance of the lm model.

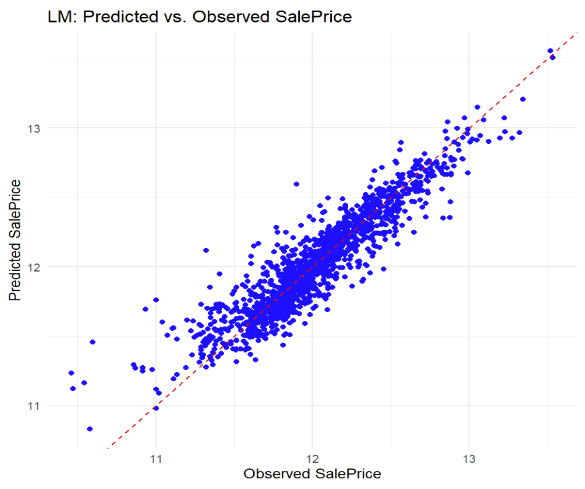


```
Call:
lm(formula = log_SalePrice ~ OverallQual + GrLivArea + GarageCars +
    X1stFlrSF + FullBath + YearBuilt + YearRemodAdd, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-0.85988 -0.07867  0.01216  0.09244  0.52509

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.688e+00  5.271e-01   3.202  0.00139 **
OverallQual   9.534e-02  4.771e-03  19.984  < 2e-16 ***
GrLivArea     2.838e-04  1.307e-05  21.715  < 2e-16 ***
GarageCars    6.742e-02  7.536e-03   8.946  < 2e-16 ***
X1stFlrSF     1.791e-04  1.361e-05  13.155  < 2e-16 ***
FullBath     -5.177e-02  1.092e-02  -4.739 2.36e-06 ***
YearBuilt     2.394e-03  2.044e-04  11.715  < 2e-16 ***
YearRemodAdd  2.197e-03  2.616e-04   8.399  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.156 on 1450 degrees of freedom
Multiple R-squared:  0.8484,	Adjusted R-squared:  0.8477
F-statistic:  1159 on 7 and 1450 DF,  p-value: < 2.2e-16
```



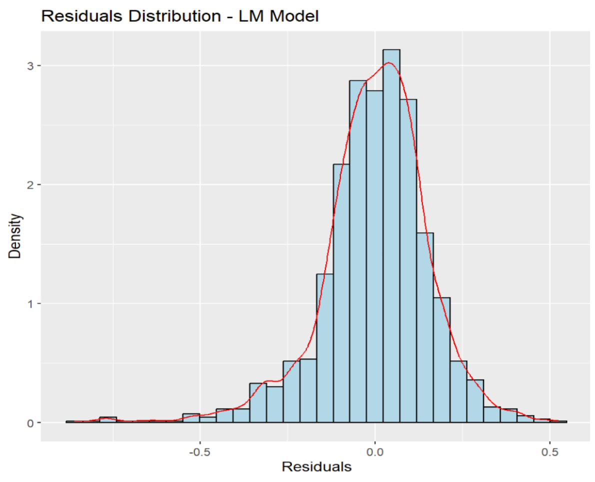


Figure 9: Linear regression results for the normalized data

On the other hand, normalized data has a better fit according to the scatter plot, and a higher R-squared value so we decided to use normalized data to fit the model. From the Residual Distribution Plot, we can see that the residuals exhibit a shape that have the qualities of a normal distribution, and that the residuals are centered around zero. This information that we gathered is significant in evaluating the validity of our linear regression model and the reliability of its predictions.

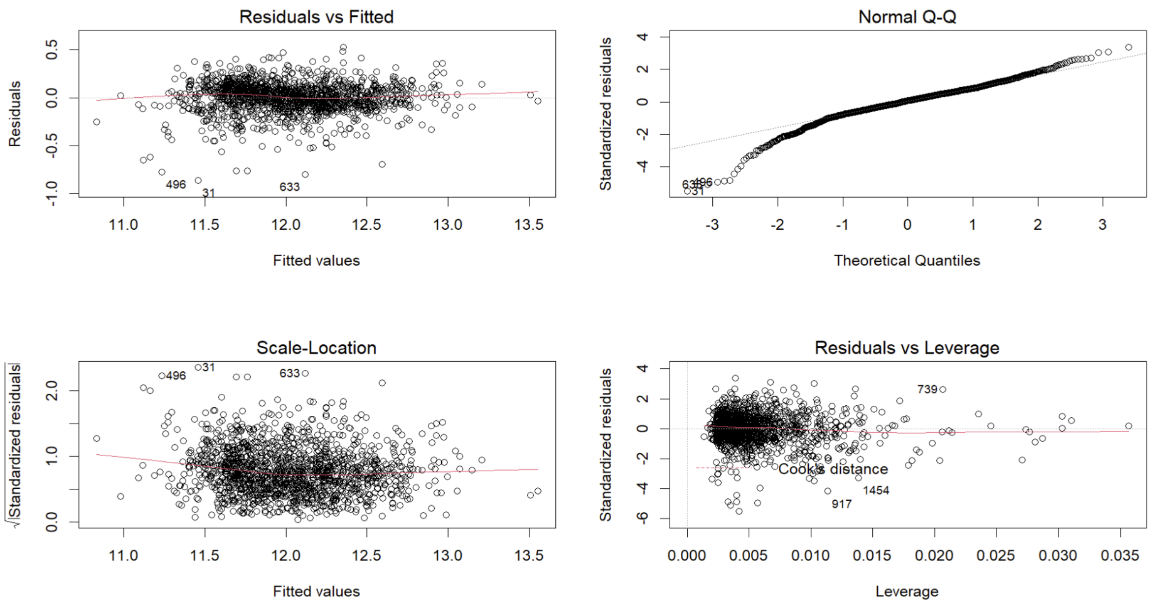
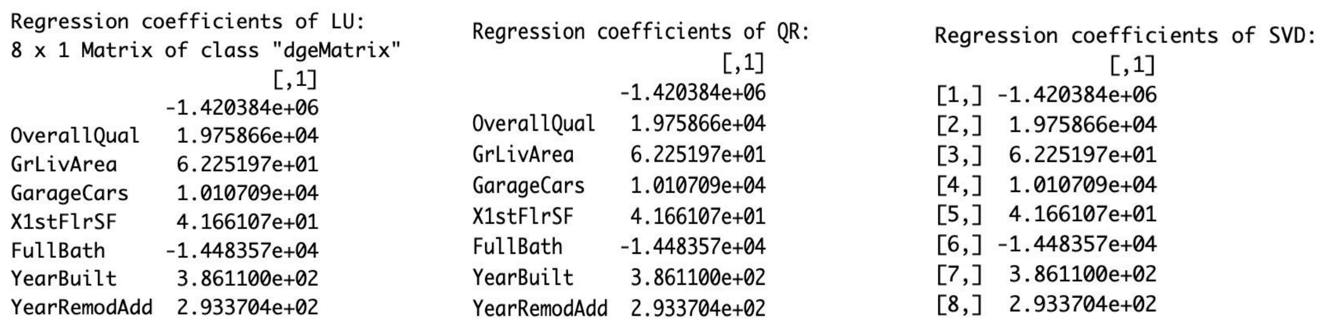


Figure 10: Diagnostic plots for the normalized data

In addition, from the Residuals vs. Fitted plot, we can see that the relationship between predictor variables and the output is approximately linear, although a few extreme outliers still exist. From the Normal Q-Q plot, we can see that most of the quantile points lie on the theoretical normal line, although the head and tail look a little off the line.



```
Regression coefficients of LU:
8 x 1 Matrix of class "dgeMatrix"
                      [,1]
             -1.420384e+06
OverallQual   1.975866e+04
GrLivArea     6.225197e+01
GarageCars    1.010709e+04
X1stFlrSF     4.166107e+01
FullBath     -1.448357e+04
YearBuilt     3.861100e+02
YearRemodAdd  2.933704e+02
```

```
Regression coefficients of QR:
                      [,1]
             -1.420384e+06
OverallQual   1.975866e+04
GrLivArea     6.225197e+01
GarageCars    1.010709e+04
X1stFlrSF     4.166107e+01
FullBath     -1.448357e+04
YearBuilt     3.861100e+02
YearRemodAdd  2.933704e+02
```

```
Regression coefficients of SVD:
              [,1]
[1,] -1.420384e+06
[2,]  1.975866e+04
[3,]  6.225197e+01
[4,]  1.010709e+04
[5,]  4.166107e+01
[6,] -1.448357e+04
[7,]  3.861100e+02
[8,]  2.933704e+02
```

Figure 11: LU, QR, and SVD decomposition results

The regression coefficients from the LU, QR, and SVD decomposition are identical, which means that the coefficients are likely correct. FullBath is the only negative coefficient, which implies that houses with fewer bathrooms are likely to have higher prices. Based on the absolute values of the regression coefficients obtained earlier, the variables that have the most influence on the house price are ranked as follows:

1. **OverallQual**: The Overall quality and finish of the house

2. **FullBath**: Full bathrooms above grade

3. **GarageCar**: Size of garage in car capacity

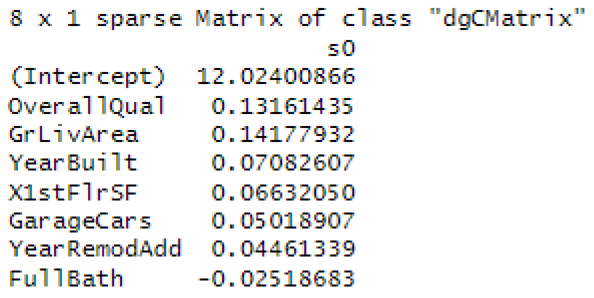
4. **YearBuilt**: Original construction date

5. **YearRemodAdd**: The remodel date (same as construction date if no remodeling or additions)

6. **GrLivArea**: The above grade (ground) living area in square feet

7. **1stFlrSF**: The first floor's square feet

With these rankings, we verified the significance and contribution of our predictors in the linear model. We will double check and verify our model again using the LASSO technique.



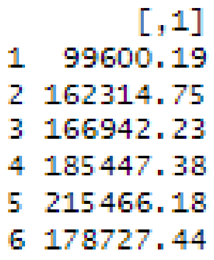
```
8 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  12.02400866
OverallQual   0.13161435
GrLivArea     0.14177932
YearBuilt     0.07082607
X1stFlrSF     0.06632050
GarageCars    0.05018907
YearRemodAdd  0.04461339
FullBath     -0.02518683
```

Figure 12: LASSO regression coefficients results

According to our LASSO regression coefficients results, the non-zero coefficients indicate the validity of our predictors and the viability of the model. The non-zero coefficients verify that these predictors have been selected as important features for predicting our target variable, SalePrice. Therefore, we are confident that the predictors that we chose are viable for our linear regression model.

4.2 House Price Predictions

After verifying our model's predictors, we will be now predicting the house prices using the LU, QR, SVD regression coefficients that we obtained from the decomposition methods. We first subsetted the new dataset X and added a column of 1's to account for the intercept term. Afterwards, we used the matrix multiplication operator in R in order to receive the house price predictions as seen below.

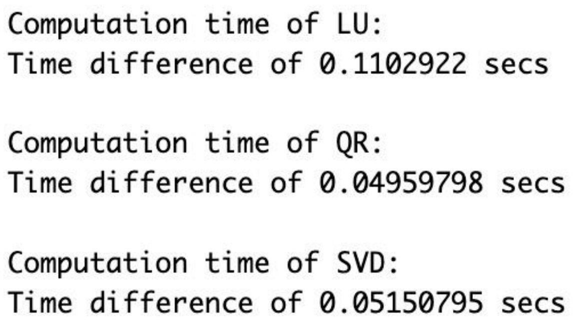


```
        [,1]
1   99600.19
2 162314.75
3 166942.23
4 185447.38
5 215466.18
6 178727.44
```

Figure 13: House predictions results from regression coefficients

4.3 Computation Speeds

In terms of computation speed, we observed that QR decomposition is faster than LU and SVD decomposition and LU decomposition is the slowest method. In theory, LU decomposition is faster than QR decomposition, due to its ability to solve linear equations faster, but for this particular study, QR decomposition is the most-time efficient method due to its ability to deal with ill-conditioned matrices.



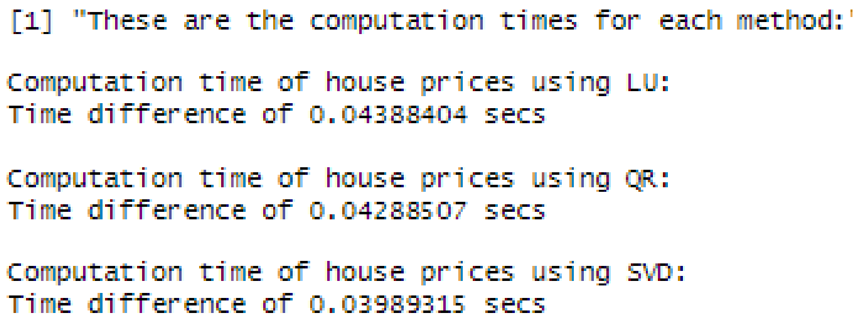
```
Computation time of LU:
Time difference of 0.1102922 secs

Computation time of QR:
Time difference of 0.04959798 secs

Computation time of SVD:
Time difference of 0.05150795 secs
```

Figure 13: Computation time of algorithms from RStudio

While QR decomposition may be faster in terms of the initial decomposition step, the actual computation of our predicted house prices varies since it depends on the implementation of the methods. In the output below, SVD method is the fastest when computing house prices because SVD decomposition can easily capture patterns in our data which can result in a more efficient and faster computation for our dataset.



```
[1] "These are the computation times for each method:'

Computation time of house prices using LU:
Time difference of 0.04388404 secs

Computation time of house prices using QR:
Time difference of 0.04288507 secs

Computation time of house prices using SVD:
Time difference of 0.03989315 secs
```

Figure 14: Computation time of decomp. Methods from RStudio

V. Summary

This statistical report compared three direct decomposition methods such as LU decomposition, QR decomposition, and Singular Value Decomposition (SVD). We chose to normalize our data in order to hold a better assumption for our linear regression assumption conditions.
From our results, we can conclude that our linear regression model is able to predict house prices using the regression coefficients, with SVD being the most efficient, from the direct decomposition methods as it displays the relationship between our predictors and our target variable. Each of these methods provides a different perspective on the coefficients and can offer additional information for our model's analysis and interpretation.

VI. References

[1] House Prices: Advanced Regression Techniques. (n.d.). Retrieved from Kaggle: https://www.kaggle.com/c/house-prices-advanced-regression-techniques

[2] Calem, P. S., Follain, J. R., Gousman, M., Laderman, E. S., & Nakamura, L. I. (2008). The Housing Crisis and State and Local Government Tax Revenue: Five Channels. Federal Reserve Bank of New York. Retrieved from https://www.newyorkfed.org/medialibrary/media/research/staff_reports/sr345.pdf

[3] FastExpert. (2023). Then and Now: Housing Market. FastExpert. Retrieved from https://www.fastexpert.com/blog/then-and-now-housing-market/

VII. Appendix

~~~
---
title: "Final presentation - Group 15"
author: "Tracy Zhu, Lynn Waiyan Kyaw, Yucheng Zhao"
date: '2023-06-09'
output: html_document
---


# Data cleaning
## Required packages
```{r}
library(tidyverse)
library(readr)
library(Metrics)
library(Matrix)
library(ggplot2)
library(corrplot)
library(dplyr)
library(MASS)
library(caret)
library(randomForest)
library(e1071)
library(glmnet)
library(zoo)
```

## Check the data
```{r}

train <- read.csv("train.csv", header=TRUE)

test <- read.csv("test.csv", header = TRUE)

cat('Dimension of train dataset:', dim(train), '\n')
~~~

~~~
cat('Column names of train dataset:', '\n')
colnames(train)
```

## Set SalePrice as the response variable and combine dataset
```{r}
# Take out test ID
test_Id <- test$Id
test$Id <- NULL
train$Id <- NULL
test$SalePrice <- NA

# Combine test and train dataset
dat1 <- rbind(train, test)
dim(dat1)
dat1 <- dat1 %>%
  mutate(across(where(is.integer), as.numeric))

# Check the combined dataset
num_var <- which(sapply(dat1, is.numeric))
cat('There are',length(num_var), 'numeric variables.','\n')
non_num_var <- which(!sapply(dat1, is.numeric))
cat('There are',length(non_num_var), 'non-numeric variables.','\n')
```

## Adjust the variables
### Find correlations with SalePrice and multicollinearity betweeen the predict
variables
```{r}
# Correlation
num_var <- which(sapply(dat1, is.numeric))
dat1_num_var <- dat1[, num_var]

# Use only the pairwise complete observations, ignor missing values
cor <- cor(dat1_num_var, use="pairwise.complete.obs")
sort_cor <- as.matrix(sort(cor[,'SalePrice'], decreasing = TRUE))

# Only keep high correlations (coefficient value > 0.5)
high_cor <- names(which(apply(sort_cor, 1, function(x) abs(x)>0.5)))
cor <- cor[high_cor, high_cor]
corrplot.mixed(cor, tl.col="black", tl.pos = "lt")
```

### Remove multicollinearity and see correlations again
```{r}
# Remove multidisciplinary and duplicated variables
rem <- c( 'GarageYrBlt', 'GarageArea', 'GarageCond', 'TotalBsmtSF', 'TotRmsAbvGrd')
dat1 <- dat1[,!(names(dat1) %in% rem)]

# Check again
num_var <- which(sapply(dat1, is.numeric))
cat('There are',length(num_var), 'numeric variables.','\n')
non_num_var <- which(!sapply(dat1, is.numeric))
cat('There are',length(non_num_var), 'non-numeric variables.','\n')
~~~

```
# See new correlations
dat1_num_var <- dat1[, num_var]
cor <- cor(dat1_num_var, use="pairwise.complete.obs")
sort_cor <- as.matrix(sort(cor[,'SalePrice'], decreasing = TRUE))

# Only keep high correlations (>0.5)
high_cor <- names(which(apply(sort_cor, 1, function(x) abs(x)>0.5)))
cor <- cor[high_cor, high_cor]
corrplot.mixed(cor, tl.col="black", tl.pos = "lt")
```

~~~
## Create a new dataset with 7 high-correlated predict variables
```{r}
pred <- c( 'SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars', 'X1stFlrSF',
'FullBath', 'YearBuilt', 'YearRemodAdd')
dat1 <- dat1[, (names(dat1) %in% pred)]
dat <- dat1[, match(pred, names(dat1))]
dat <- as.data.frame(lapply(dat, as.numeric))
~~~

## Check two most related predict variables (>0.7)
### Overall Quality
```{r}
ggplot(data=dat[!is.na(dat$SalePrice),], aes(x=factor(OverallQual), y=SalePrice))+
        geom_boxplot(col='black') + labs(x='Overall Quality') +
        scale_y_continuous(breaks= seq(0, 1000000, by=100000))
```

### Above Grade Living Area
```{r}
ggplot(data = dat[!is.na(dat$SalePrice),], aes(x = GrLivArea, y = SalePrice)) +
  geom_point(col = 'red') +
  geom_smooth(method = "lm", se = FALSE, color = "black") +
  scale_y_continuous(breaks = seq(0, 100000, by = 100000)) +
  geom_text(data = subset(dat, GrLivArea > 4500 & !is.na(SalePrice)),
            aes(x = GrLivArea, y = SalePrice, label = rownames(subset(dat, GrLivArea >
4500 & !is.na(SalePrice)))),
            hjust = 0, vjust = 2, size = 4)
```

## Check completeness
```{r}
# See if we need to remove two outliers?
dat[c(524, 1299), c('SalePrice', 'OverallQual')]
ncol(dat)
dat <- dat[-c(524, 1299),]
```

## Clean missing values
```{r}
NAcol <- which(colSums(is.na(dat)) > 0)
sort(colSums(sapply(dat[NAcol], is.na)), decreasing = TRUE)
cat('There are', length(NAcol), 'columns with missing values', '\n')
names(sort(colSums(sapply(dat[NAcol], is.na)), decreasing = TRUE))
```
```
```

~~~
### Replace missing values with interpolated values
```{r}
dat$GarageCars <- na.approx(dat$GarageCars, na.rm = FALSE)

# Check it again
NAcol <- which(colSums(is.na(dat)) > 0)
sort(colSums(sapply(dat[NAcol], is.na)), decreasing = TRUE)
cat('There are', length(NAcol), 'columns with missing values', '\n')
names(sort(colSums(sapply(dat[NAcol], is.na)), decreasing = TRUE))

# Heatmap
cor_matrix <- cor(dat, use = "pairwise.complete.obs")
corrplot(cor_matrix, method = "color", tl.col = "black")
~~~

## Normalize the right skewed distribution
```{r}
# Normalize the right-skewed distribution
ggplot(data = dat[!is.na(dat$SalePrice),], aes(x = SalePrice)) +
  geom_histogram(binwidth = 10000, color = "black", fill = "lightblue") +
  scale_x_continuous(breaks = seq(0, 1000000, by = 100000)) +
  geom_vline(aes(xintercept = mean(SalePrice)), color = "red", linetype = "dashed",
size = 1) +
  geom_vline(aes(xintercept = median(SalePrice)), color = "darkblue", linetype =
"dashed", size = 1)

dat$log_SalePrice <- log(dat$SalePrice)

ggplot(data = dat[!is.na(dat$SalePrice),], aes(x = log_SalePrice)) +
  geom_histogram(binwidth = 0.1, color = "black", fill = "lightblue") +
  scale_x_continuous(breaks = seq(0, 1000000, by = 100000)) +
  geom_vline(aes(xintercept = mean(log_SalePrice)), color = "red", linetype =
"dashed", size = 1) +
  geom_vline(aes(xintercept = median(log_SalePrice)), color = "darkblue", linetype =
"dashed", size = 1)

# Back-transformation of log_SalePrice to SalePrice
dat$SalePrice <- exp(dat$log_SalePrice)

# Check the distribution of back-transformed SalePrice
ggplot(data = dat[!is.na(dat$SalePrice),], aes(x = SalePrice)) +
  geom_histogram(binwidth = 10000, color = "black", fill = "lightblue") +
  scale_x_continuous(breaks = seq(0, 1000000, by = 100000)) +
  geom_vline(aes(xintercept = mean(SalePrice)), color = "red", linetype = "dashed",
size = 1) +
  geom_vline(aes(xintercept = median(SalePrice)), color = "darkblue", linetype =
"dashed", size = 1)
```

## Separate train and test dataset
```{r}
train <- dat[!is.na(dat$SalePrice) | !is.na(dat$log_SalePrice), c(-which(names(dat) ==
"log_SalePrice"))]
```

```
test <- dat[is.na(dat$SalePrice) | is.na(dat$SalePrice), c(-which(names(dat) ==
"log_SalePrice"))]

cat('Dimension of train dataset:', dim(train), '\n')
cat('Dimension of train dataset:', dim(test), '\n')

test <- as.data.frame(lapply(test, as.numeric))

str(train)
str(test)

# Check for NAs
cat(length(which(colSums(is.na(train)) > 0)), 'columns in train dataset have missing
values. ', '\n')
cat(length(which(colSums(is.na(test)) > 0)), 'columns in test dataset have missing
values:', '\n')
sort(colSums(sapply(test[NAcol], is.na)), decreasing = TRUE)
```

~~~
# Prepare the models
## Create dummy test dataset
```{r}
dummy_test <- train[, !(names(train) %in% "SalePrice")]
~~~

# Models
lm
```{r}
lm_model <- lm(SalePrice ~ OverallQual + GrLivArea + GarageCars + X1stFlrSF + FullBath
+ YearBuilt + YearRemodAdd, data = train)
summary(lm_model)

# Visualized predicted and observed values for LM model
lm_predicted <- predict(lm_model, newdata = dummy_test)
lm_comparison <- data.frame(Observed = train$SalePrice, Predicted = lm_predicted)
ggplot(lm_comparison, aes(x = Observed, y = Predicted)) +
  geom_point(color = "blue") +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  labs(x = "Observed SalePrice", y = "Predicted SalePrice") +
  ggtitle("LM: Predicted vs. Observed SalePrice") +
  theme_minimal()

# Residual plot for LM model
residuals_lm <- train$SalePrice - lm_predicted
ggplot() +
  geom_histogram(aes(x = residuals_lm, y = ..density..), color = "black", fill =
"lightblue") +
  geom_density(aes(x = residuals_lm), color = "red") +
  labs(x = "Residuals", y = "Density") +
  ggtitle("Residuals Distribution - LM Model")
```

## Accuracy
```{r}
# Calculate accuracy metrics for LM
```

~~~
lm_rmse <- sqrt(mean((train$SalePrice - lm_predicted)^2))
lm_mae <- mean(abs(train$SalePrice - lm_predicted))

# Print results for accuracy metrics
cat("Accuracy metrics for LM model:\n")
cat("RMSE:", lm_rmse, "\n")
cat("MAE:", lm_mae, "\n")
```

# Decomposition
```{r}
train <- train %>% mutate_if(is.integer, as.numeric)
X <- train[, sapply(train, is.numeric)]
X <- train[, -which(names(train) == "SalePrice")]
X <- as.matrix(X)
X <- cbind(1, X)
Y <- train$SalePrice
Y <- as.matrix(Y)
```

## LU decomposition
```{r}
start.time_lu = Sys.time()
XtX <- t(X)%*%X
XtY <- t(X)%*%Y
start.time <- Sys.time()
lu <- expand(lu(XtX))
L <- lu$L
U <- lu$U
P <- lu$P
reg_coef_lu <- solve(P%*%L%*%U, XtY)
end.time_lu = Sys.time()
cat("Computation time of LU:\n")
end.time_lu - start.time_lu
```

## QR decomposition
```{r}
start.time_qr = Sys.time()
qr <- qr(X)
Q <- qr.Q(qr)
R <- qr.R(qr)
reg_coef_qr <- qr.solve(R, t(Q) %*% Y)
end.time_qr = Sys.time()
cat("\nComputation time of QR:\n")
end.time_qr - start.time_qr
```

## SVD
```{r}
start.time_svd = Sys.time()
svd <- svd(XtX)
S_inv <- diag(1/(svd$d))
V <- svd$v
U <- svd$u
~~~

~~~
#reg_coef_svd <- solve(U%*%S%*%t(V), t(X)%*%Y)
#reg_coef_svd
reg_coef_svd <- V %*% S_inv %*% t(U) %*% t(X) %*% Y
end.time_svd = Sys.time()
cat("\nComputation time of SVD:\n")
end.time_svd - start.time_svd
```

## time summary
```{r}
# Regression coefficients from LU, QR, SVD
cat("\nRegression coefficients of LU:\n")
reg_coef_lu
cat("\nRegression coefficients of QR:\n")
reg_coef_qr
cat("\nRegression coefficients of SVD:\n")
reg_coef_svd

influence_rank<-colnames(X)[order(abs(reg_coef_lu),decreasing=T)[2:8]]
# 1. Overall quality 2. Full bathrooms above grade
# 3. Size of garage in car capacity 4. Original construction date
# 5. Remodel date 6. Above grade (ground) living area square feet
# 7. First Floor square feet
cat("\nRanking of variables that influence SalePrice the most:\n")
influence_rank

```

## Predict house prices:
```{r}
start_lu <- Sys.time()

# Using the LU decomposition regression coefficient

X_pred_LU <- subset(test, select = c(OverallQual, GrLivArea, GarageCars, X1stFlrSF,
FullBath, YearBuilt, YearRemodAdd))

X_pred_LU <- cbind(1, X_pred_LU) # Add a column of 1's to account for the intercept
term.

predicted_prices_LU <- as.matrix(X_pred_LU) %*% reg_coef_lu

print(head(predicted_prices_LU))

end_lu <- Sys.time()

computation_time_lu <- end_lu - start_lu
```

```{r}
start_qr <- Sys.time()

# Using the QR decomposition regression coefficient
~~~

~~~
X_pred_QR <- subset(test, select = c(OverallQual, GrLivArea, GarageCars, X1stFlrSF,
FullBath, YearBuilt, YearRemodAdd))

X_pred_QR <- cbind(1, X_pred_QR) # Add a column of 1's to account for the intercept
term.

predicted_prices_QR <- as.matrix(X_pred_QR) %*% reg_coef_qr

print(head(predicted_prices_QR))

end_qr <- Sys.time()

computation_time_qr <- end_qr - start_qr

```
```{r}
start_svd <- Sys.time()

# Using the SVD decomposition regression coefficient

X_pred_SVD <- subset(test, select = c(OverallQual, GrLivArea, GarageCars, X1stFlrSF,
FullBath, YearBuilt, YearRemodAdd))

X_pred_SVD <- cbind(1, X_pred_SVD) # Add a column of 1's to account for the intercept
term.

predicted_prices_SVD <- as.matrix(X_pred_SVD) %*% reg_coef_svd

print(head(predicted_prices_SVD))

end_svd <- Sys.time()

computation_time_svd <- end_svd - start_svd
```
Using the LU, QR, SVD decomposition regression coefficients we were able to predict
house prices.
```{r}
print("These are the computation times for each method:")
cat("\nComputation time of house prices using LU:\n")
print(computation_time_lu)
cat("\nComputation time of house prices using QR:\n")
print(computation_time_qr)
cat("\nComputation time of house prices using SVD:\n")
print(computation_time_svd)
```
~~~