Housing Price Prediction Model

STOR 664: Minji Kim, SooHyun Kim

December 1, 2021

1 Introduction

In 2020, according to the National Association of Realtors, it is estimated that about 5.64 million houses were sold just in the United States, which is the highest number of houses sold in a year since 2006 [6]. In addition to this high number of people that actually buy and sell houses, other parties including developers, investors, appraisers, and banks are also involved with this process [4]. With all these parties concerned with housing prices, it is important to be able to accurately predict housing prices. In this project, we aim to look in deeper into this issue with two research questions;

(a) How to accurately predict housing prices?

(b) Do transformation and standardization of variables increase the prediction performance? What do they imply for this house price prediction model?

To answer these two questions, we will aim to build the best model to predict housing prices using data from residential house sales in Ames, Iowa between the years 2006 and 2010. We will first conduct a simple exploratory data analysis (EDA) to look into the statistical method to implement in our research. Next, we will discuss the appropriate data transformation method to implement for both the response and exploratory variables. After, we will look at individual data points to identify outliers or influential data points. Finally, we will use the data shrinkage method to remedy the multicollinearity of variables and fit the best model for housing price prediction.

1.1 Data Description

In order to look into housing price prediction, we will be using data from De Cock [2] retrieved from Kaggle [7]. This data set contains 1,461 observations of residential house sales in Ames, Iowa between the year 2006 and 2010. This data set contains comprehensive information about each house sale. It contains 79 explanatory variables about the houses. This would be useful in predicting the as it is widely accepted that the physical condition, concept and the location of the house has the biggest impact on its price (Alfiyatin et al 2017). The variables included in this data include multiple variables that are able to measure each of these three categories. Moreover, this data set spans from 2006 to 2010 allowing analysis into the changes in the prediction variables of housing price prediction over time. For detailed explanation of each variable and data collection refer to De Cock, D. J. J. o. S. E. 2011. "Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project." (19:3).

1.2 Simple EDA

The response variable is the sale price of the property. In order to look deeper into housing price, we compare this distribution along with a few exploratory variables of interest. First, housing price is compared to the date the property was sold. In Figure 1, it seems like the higher

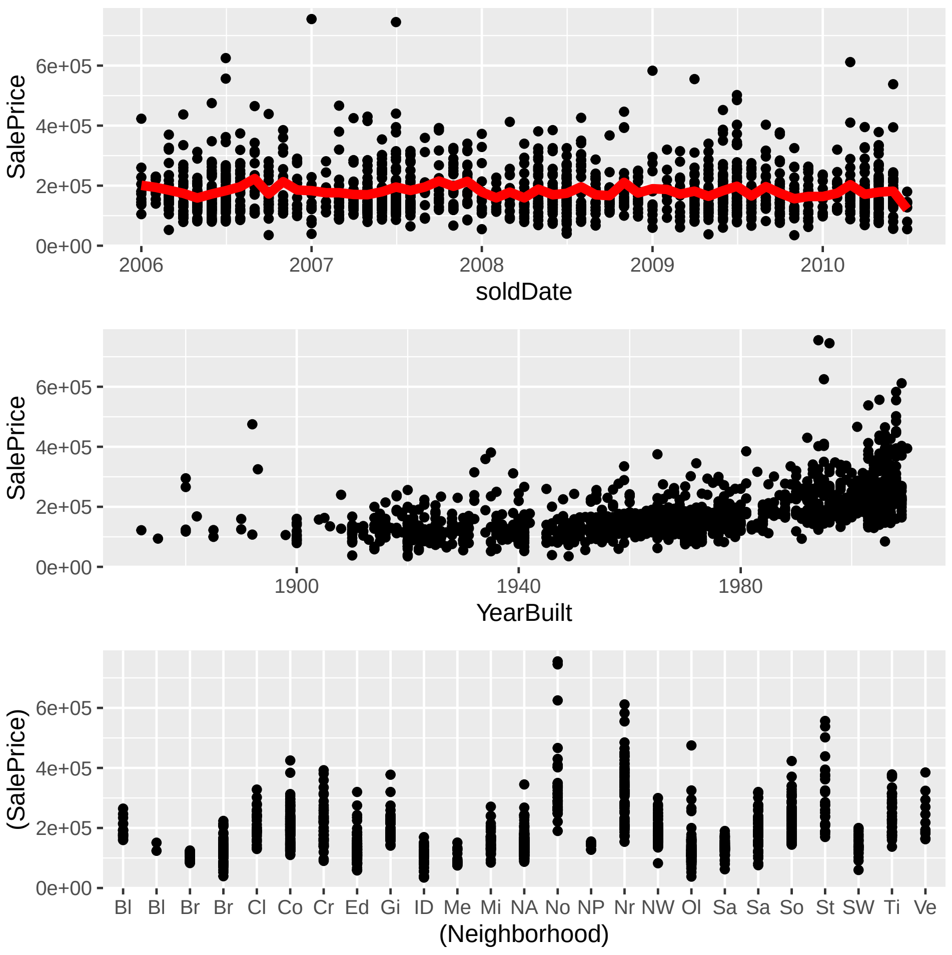

Figure 1: Observation: Scatter plots of `SalePrice` along with some predictor variables

the price of a house, the greater the increase. This hints to the possibility that the data may not be efficiently explained with a linear trend and that the error will not follow the normality assumption when liner models are applied. A transformation on the response variable before fitting a linear model is needed. This will be further discussed in the following section. Also, comparing housing price to the year the property was built, as seen in Figure 1, it seems to be an exponentially increasing. Finally, the housing price is plotted against the neighborhood in Figure 1. It appears that there is some trend and heteroskedasticity in housing price between neighborhoods. This follows the common understanding that location is an essential factor of housing price.

2 Data Transform

2.1 Transformation of Explanatory Variables

Here, we observe 79 predictors and deal with the following problems:

(a) Missing data imputation.

(b) Categorical Variables

- Ordered factors
- Initial discoveries of significant factors using FDR

(c) Numerical Variables

- Yeo-Johnson Transform
- Multicollinearity

To be specific, exploratory variables break down into two categories: numerical and categorical. After initial data pre-processing in step (a), we get 21 numerical variables and 49 categorical variables. In this section, we will look into each group of variables and perform data-adaptive transforms.

(a) Missing data imputation

One of the main issues with this data set is the missing values with the dependent variables. Figure 2 shows the proportions of missing values in percentage for each variable. Out of the 79 variables, there are 19 variables have at least one missing variable. Six variables with a missing value of more than 10 percent are dropped; PoolQC, MiscFeature, Alley, Fence, FireplaceQu, LotFrontage. For the numeric variable of GarageYrBlt, which is the year the garage was built, 0 is assigned to the missing values. Other variables with the missing values were replaced with 'None' as they were either concerning variables related to the garage or the basement. The data points that had a missing value were meaningful as they indicated houses without either a garage or a basement. In addition, MasVnrType, MasVnrArea, and Electric were dropped as it would not be possible to assign values to the missing data points of a categorical data.

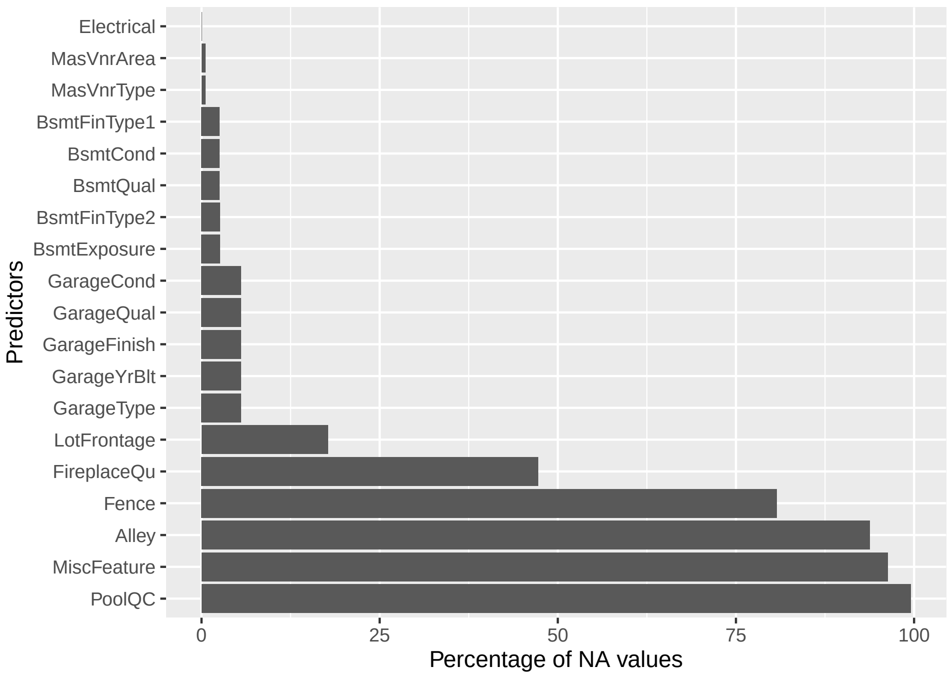

Figure 2: `NA` percentage of predictor variables

(b) Categorical Data Transformation

We consider transformation of categorical and ordinal variables. With categorical data, we use the `as.factor` function to encode the variables as categorical. For ordinal data, we assign a numeric value to each level. For example, with 'ExterQual', the values are given as excellent, good, average/typical, fair, and poor were replaced by numeric value from 5 to 1 respectively. The `as.ordered` function, which encodes the variables into a ordinal variable, is used. In contrast, the `as.factor` function assigns a hierarchical relationship between the values.

Due to the overwhelming number of categorical variables, we select and trim some negligible variables in advance. In this initial variable selection process, we deal with *multiple hypothesis testing*. We will refer to finding significant factors in explaining SalePrice as "*discovering*" them. Our goal here is to discover a set of significant categorical variables. We test each simple linear regression model between SalePrice and each factor to get the models' p-values. Note that since we will further apply variable selection or shrinkage models to fit multiple linear models, we prefer to adopt a liberal criterion for the discoveries. As a result, rather than focusing on the Type I error, which puts an emphasis on falsely detecting non-significant factor as significant, we would like to control false discovery rate(FDR). Indeed, controlling FDR is always less conservative than controlling FWER. This fact is proved in the following proposition.

Proposition 2.1. *A decision rule D has rejected R out of m hypotheses; V of these decisions were false discoveries, while S of them were true discoveries. Then FDR equals $E(V/R)$. Let FWER be the family-wise error rate, i.e.* $\mathrm{P}(\textit{any false discovery}) = \mathrm{P}(V \geq 1)$.

Then, FWER(D) $\geq$ FDR (D).

Proof. Note that $R = V + S$. Thus, both $\frac{V}{R} \leq 1_{\{V \geq 1\}}$ when $R \geq 1$ and $0 \leq 1_{\{V \geq 1\}}$ when $R = 0$ hold. Now we can prove the proposition as following.

$$\mathrm{FDR}(D) = \mathrm{E}(\ \frac{V}{R} \cdot 1_{\{R \geq 1\}} + 0 \cdot 1_{\{R=0\}}\ )\ \leq\ P(V \geq 1) = \mathrm{FWER}(D)$$

□

As a result, we would like to control FDR in our initial variable trimming. Categorical variables in house price data such as GarageQual and GarageCond, or BsmtFinType1 and BsmtFinType2 are highly related. This implies that our p-values are less likely to be independent. As a result, we control the FDR by the method of Benjamini and Yuketieli [1]. This method is more conservative than the usual Benjamini-Hochberg result. Thus, we set the significance level a bit higher as 0.2, to encourage discovering significant variables. The following is the corresponding R code and results. With further examination of the result, we omit 7 variables: Condition2, LandSlope, Street, MoSold, Utilities, BsmtHalfBath, and YrSold.

```
> fac.var = housing2 %>% dplyr::select(is.factor)
> pvals = sapply(fac.var, function(ff){
   lm.f = lm(SalePrice ~ ff, data= data.frame(ff, SalePrice = housing2$SalePrice))
   fstat = pf(summary(lm.f)$fstatistic
   return(1-fstat[[1]], df1 = fstat[[2]], df2 = fstat[[3]]))
 })
> pvals = sort(pvals)
> padj = p.adjust(pvals, method="BY")
> padj[which(padj>0.2)]

 [1]  Condition2  LandSlope   Street   MoSold   Utilities BsmtHalfBath  YrSold
 [1]      0.277      0.278     0.651    1.000      1.000        1.000    1.000
```

(c) Numerical Data Transformation

Figure 3 shows the original distributions of numeric predictors and their empirical densities. Although the normal assumptions is not a necessity for predictor variables, the high skewness in the density would damage overall linear trend. We clearly observe skewed densities in some

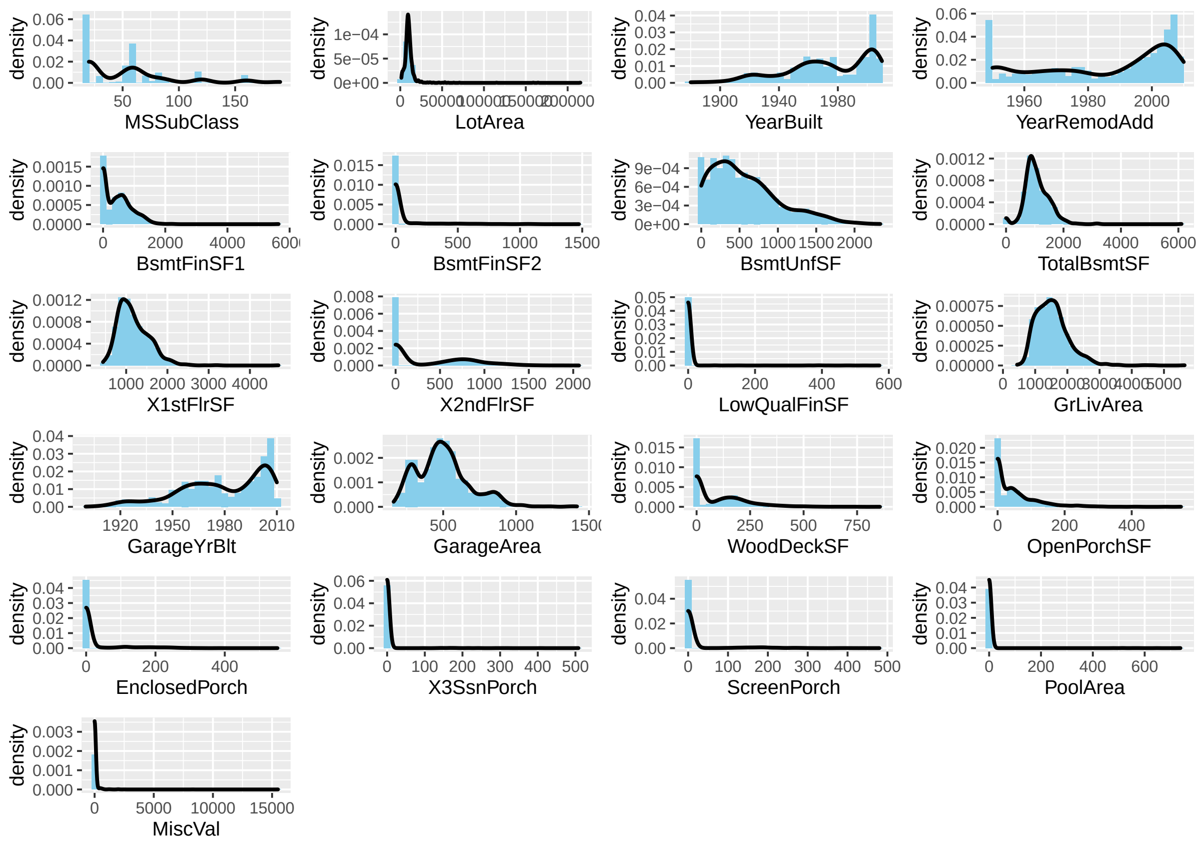

Figure 3: Distribution of numeric predictors without any transform

explanatory variables. Thus, we consider applying the following Yeo-Johnson Transform [10], which is an extension of the Box-Cox Transform for non-positive numerical variables.

$$y_i^{(\lambda)} = \begin{cases} ((y_i+1)^\lambda - 1)/\lambda & \text{if } \lambda \neq 0, y \geq 0 \\ \log(y_i+1) & \text{if } \lambda = 0, y \geq 0 \\ -((-y_i+1)^{2-\lambda} - 1)/(2-\lambda) & \text{if } \lambda \neq 2, y < 0 \\ -\log(-y_i+1) & \text{if } \lambda = 2, y < 0 \end{cases}$$

Here, the best λ value for each variable is obtained using the `boxCox` function in the `car` package.

Figure 4 shows the correlation plot of 10 variables with the highest correlation values to the response variable. Selected variables are the following: SalePrice, GrLivArea, GarageArea, TotalBsmtSF, X1stFlrSF, YearBuilt, YearRemodAdd, GarageYrBlt, OpenPorchSF, LotArea. From the plots, there is a clear correlation between variables. Thus, the *multicollinearity* issue needs to be addressed. To diagnose the multicollinearity, we look at the condition number and VIF values. First, let X_n be the subset of the data matrix consisting of numerical variables. Then, we get the eigenvalues of $X_n^T X_n$, $\lambda_1 \geq \cdots \geq \lambda_{21}$. Note that we standardized the matrix before the eigenvalue decomposition. Then the condition number of our numerical dataset is derived by $\kappa_n = \sqrt{\frac{\lambda_1}{\lambda_{21}}} = 14.33$ [3], which is not negligible. We also see the variable-wise collinearity based on the variance inflation factors (VIFs). Table 1 shows VIFs of each variable. There are 3 variables whose VIF is bigger than 10, and 7 variables with VIF bigger than 4. Some collinearities are detected between variables, and thus we consider the shrinkage methods, PCA, Ridge and Lasso models, to manage the multicollinearity issue.

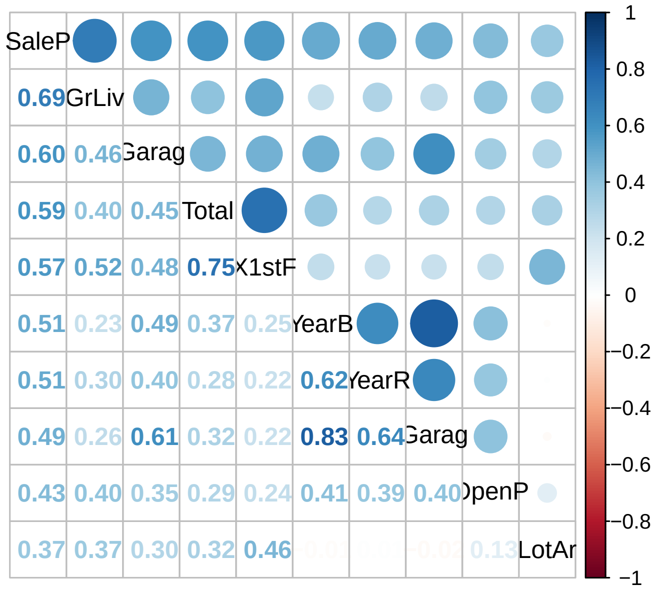

Figure 4: Correlation plot of variables having 10 largest correlation with SalePrice

GrLiveArea	X2ndFlrSF	X1stFlrSF	TotalBsmtSF	GarageYrBlt	BsmtUnfSF	YearBuilt
18.700	15.647	14.749	5.671	4.681	4.357	4.303
BsmtFinSF1	GarageArea	YearRemodAdd	MSSubClass	LotArea	OpenPorchSF	EnclosedPorch
3.523	2.292	1.919	1.897	1.707	1.412	1.302
BsmtFinSF2	WoodDeckSF	LowQualFinSF	ScreenPorch	PoolArea	MiscVal	X3SsnPorch
1.266	1.213	1.184	1.068	1.066	1.036	1.021

Table 1: Sorted VIF values

2.2 Stepwise variable selection and Data Transform

After conducting the EDA, it has highlighted the need for a transformation on the response variable, housing prices, to fit linear models. Here, we consider the following transformations:

$$\begin{aligned} y_t &= P_t \\ y_t &= C_1 \frac{P_t^\lambda - 1}{\lambda} \\ y_t &= C_2 \log P_t \end{aligned}$$

where P_t is the original housing prices. C_1 and C_2 are multiplied to the two transformations respectively due to scaling issues that may occur when comparing the models.

In this section we find the best linear regression model with variables selected using the the stepwise selection for each transformed y values. For each model, we applied `stepAIC` function to apply the variable selection process based on the AIC criteria for each response variable. Then, PRESS, AIC, and BIC criteria are computed to compare the three models. Based on the best model selected from this comparison, we will conduct a diagnosis for outliers and influential points. After omitting the outliers and influential points, we will construct a more sophisticated model and compare its efficiency.

The following Table 3 summarizes the variables selected by the AIC criteria for each transformed y values. For the Box-Cox transformation, we find the best λ at 0.1. We see that for each model the selected number of variables slightly differ. For Model 1, using the y without transformation, selected 36 variables. Model 2, with Box-Cox transformation selected 46 variables. The last model, Model 3, using the log transformation selected 44 variables. Among the

Models	Model1	Model2	Model3
Transform	Identity	Log	B-C($\lambda = 0.1$)
Number of Variables	36	46	44
Common Variables	(33) YearBuilt, YearRemodAdd, BsmtFinSF1, BsmtUnfSF, TotalBsmtSF, LowQualFinSF, GrLivArea, WoodDeckSF, OpenPorchSF, X3SsnPorch, ScreenPorch, PoolArea, MSZoning, LotConfig, Neighborhood, Condition1, BldgType, OverallQual, OverallCond, RoofMatl, Exterior1st, BsmtQual, BsmtExposure, FullBath, BedroomAbvGr, KitchenAbvGr, KitchenQual, TotRmsAbvGrd, Functional, Fireplaces, GarageCars, SaleCondition, LotArea		
s Distinct Variables	LotShape LandContour BsmtFinSF2	(10) LowQualFinSF, RoofMatl, Exterior1st, BsmtQual, BsmtExposure, HeatingQC, BsmtFullBath, Fireplaces, GarageCars, GarageQual	
		MSSubClass Heating HouseStyle	X1stFlrSF

Table 2: Variables selected using different response variable transformation.

three models, 37 variables were selected in common.

Based on these three models, we use AIC, BIC, and PRESS statistics to choose the optimal transformation. As we see in Table 3, based on the AIC, Model 2 with the log transformation has the lowest value (30917.12). For BIC statistics, Model 1 with the Box-Cox transformation is the optimal model (31777.42). PRESS statistics shows that Model 2 using a log transformation is the optimal model (4.815e+13). Therefore, based on the three statistics we conclude the log transformation is the optimal transformation. We will use the log transformation for further analysis in detecting outlier and influential points. However, it is important to note that as the optimal λ of 0.1 is close to 0, the transformation is similar to the log transformation. Therefore, when fitting the model, we will revisit the issue of response variable transformation.

Selected Var	Model 1			Model 2			Model 3		
Criteria	AIC	BIC	PRESS	AIC	BIC	PRESS	AIC	BIC	PRESS
Identity	31856.37	32635.51	1.022e+14	31907.16	32895.46	8.383e+13	31890.24	32815.79	8.892e+14
Log	31001.13	31780.28	6.682e+13	**30917.12**	31905.42	**4.815e+13**	30917.15	31842.70	5.179e+13
B-C($\lambda = 0.1$)	30998.29	**31777.42**	6.659e+13	30932.40	31920.70	4.877e+13	30929.54	31855.09	5.236e+13

Table 3: Results of AIC, BIC, and PRESS statistics for House price data.

Assessing the normality assumption, Figure 5 shows the histogram and the QQ-plot of the residuals after fitting each model for three response variable. Looking at the histogram, all three types of response variables seem to generally follow the normal density. However, the QQ-plot shows that possible outlier may be present in the data set. This issue will be further discussed in the next section using the log transformed response variable.

2.3 Outlier Detection

Based on our previous analysis of the data, there seems to be outliers present. In order to identify these outliers, Figure 6 shows the residual plots of the Box-Cox transformed model, which was selected in the previous section. Based on this plot, 3 possible outlier points are identified: `494, 595, 1252`. To further investigate other possible outliers in the data, `outlierTest` function is used. Based on this function, 7 possible outliers were detected: `494, 595, 1252,`

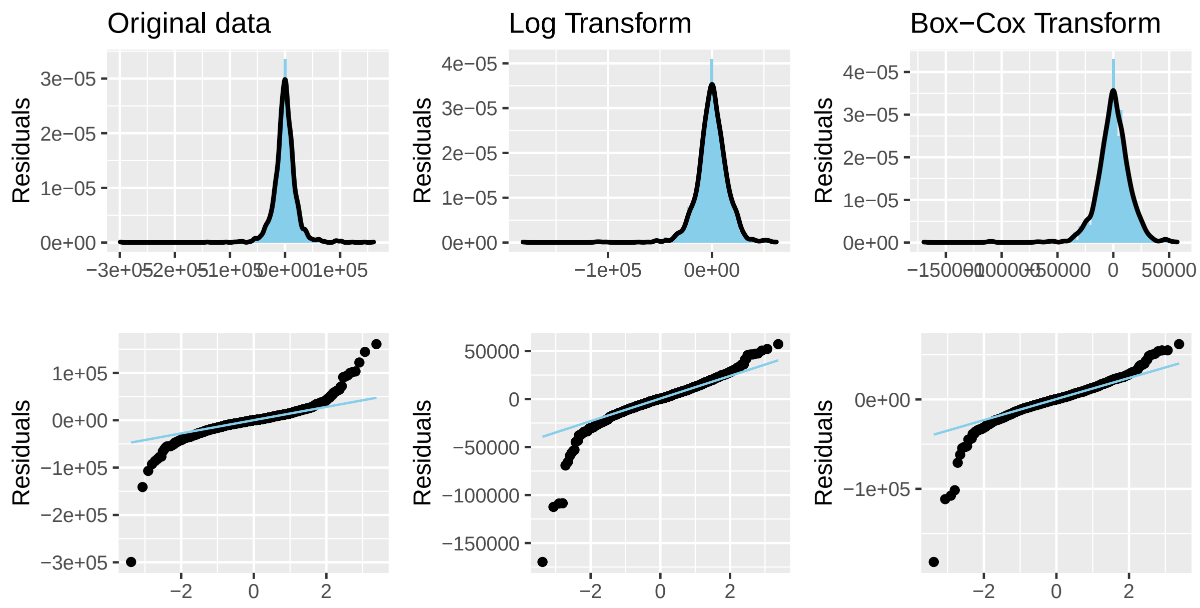

Figure 5: Check Normality: histogram and the QQ-plot of residuals for three types of response variable

`436, 554, 1355, 102`. The residual plots only three outliers with the highest Bonferroni p-values calculated based on the studentized residuals were given. Using the formula an additional 4 outliers were detected. Figure 7 shows plots of these outliers to provide a better understanding of where these outliers occurred. From Figure 7, the outliers in this data seems to be houses that were sold below the market price, as we can see from all three graphs that the outliers are located towards the bottom of the y-axis. In addition, the majority of the outliers are houses that were built more recently.

```
> outlierTest(model2)
```

```
       rstudent unadjusted p-value Bonferroni p
494  -12.410919         2.3913e-33   3.2618e-30
595   -7.223492         9.0232e-13   1.2308e-09
1252  -6.924734         7.1302e-12   9.7256e-09
436   -6.793410         1.7262e-11   2.3546e-08
554   -6.066409         1.7548e-09   2.3936e-06
1355  -4.267186         2.1367e-05   2.9145e-02
102    4.151995         3.5309e-05   4.8162e-02
```

After omitting theses outliers, we will fit the model again adding the Shrinkage model. Shrinkage models are used to address multicollinearity issue in our data set. As mentioned in“The LASSO and Other Regularization Methods in Regression” handout, When implementing a Shrinkage model, it is important to standardize the data as the Ridge and LASSO methods are more sensitive to scaling issues. Cross-Validation to evaluate the performance of each model and the response variable transformation will be done.

3 Shrinkage models

In this section, we will compare the following three methods :

Model 1: Ridge Regression **Model 2:** LASSO **Model 3:** PCR + LASSO

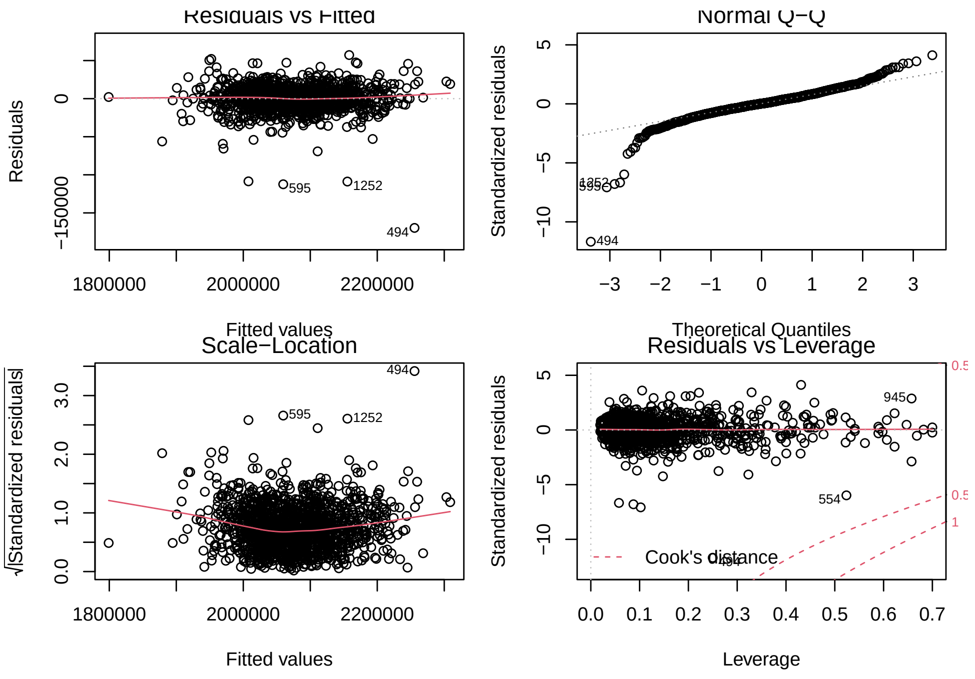

Figure 6: Residual plots of Log transformed model

in predicting house price. We consider three models based on shrinkage methods to treat the multicollinearity problem in the exploratory data. First of all, Ridge and LASSO models are applied through `glmnet` function in R. Model 3 considers applying principal component analysis to the numerical variables to select principal components that sufficiently explain their overall variance. Then, we combine the principal component analysis on the numerical variables with the LASSO.

Detailed steps conducted to compare these methods are as follows:

- 10% of the most recent data points are treated as the test set.
- With the remaining train data, we fitted models using three transformed response variables with the three shrinkage methods mentioned above.
- Using the 20 fold cross-validation for each of the models, we compare the MSE values to find the optimal transformation and shrinkage method. The results are shown in Table 4.
- Finally, we validate the accuracy of the optimal model by predicting the housing price of the test set.

Table 4 contains the results of these steps. Based on the MSE we can see that Model 2 using the log transformation has the lowest MSE (0.061) and the lowest standard deviation (0.033). Therefore, we can conclude the LASSO method using the log transformation as the optimal model.

To illustrate in detail about the optimal model, the left plot in Figure 8 shows the result of the `plot fit` of the training set. From this plot, starting from the left with one non-zero β_j to the right end with all 66 parameters, the optimal λ is located relatively to the left side. This means that many of the parameters were dropped from the model. The right plot shows the MSE value against the $\log(\lambda)$. We can observe that the optimal $\log(\lambda)$ would be around -4.

With the optimal model, we predict the housing price with the test set, which contains the most recent data account for 10% of the whole data that was set aside. The results for this can be found in Figure fig:test. The scatter plot on the left shows how the predicted price compares

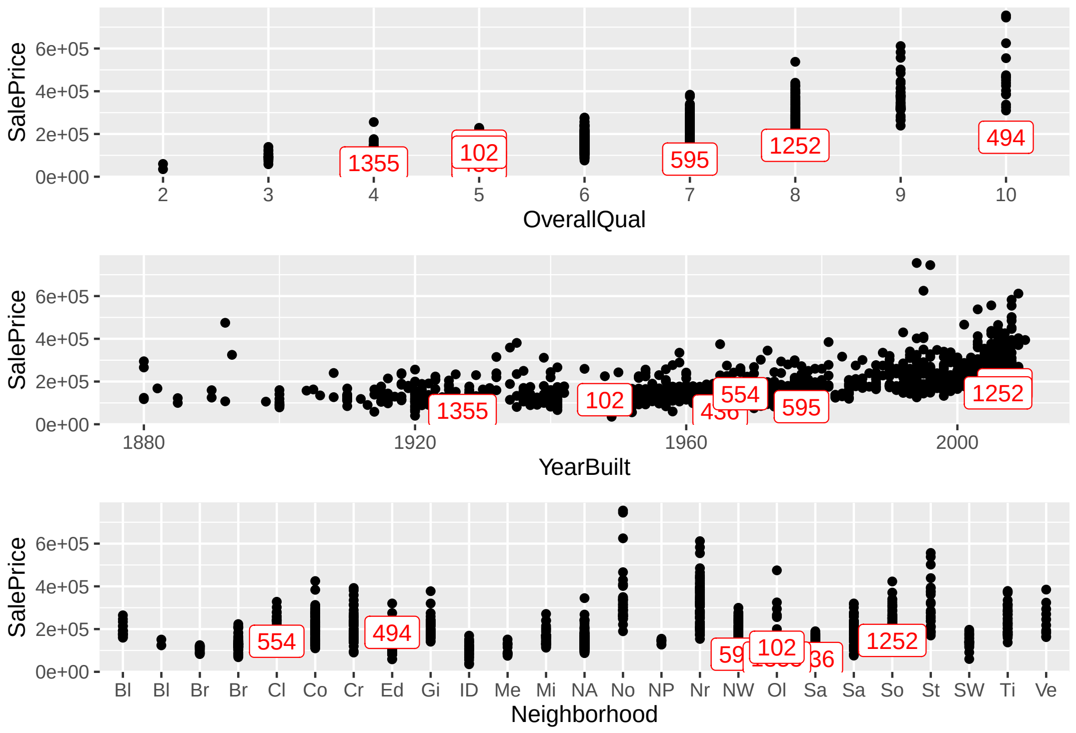

Figure 7: Outliers observation

to the real price. The real price is shown in the x-axis and the predicted price is shown on the y-axis. As the points are roughly around the straight line ($y = x$), we can conclude that our prediction was overall accurate. The plot on the right shows the histogram of the residuals. From this we can state that it follows a roughly normal distribution. Thus, we conclude that the linear model is adequate in housing price prediction.

What is even more interesting is when we did not take the step of using the FDR the MSE has increased. This result is also in Table 4, where the second row indicating "FDR" and "All" means the two different process of obtaining the MSE. The "All" indicates the whole linear model fitting process starts with all variables without initial variable trimming, and with the removal of missing data. "FDR" indicates the process proposed in this project, including the variable trimming using FDR control. Hence, it is important to be able to remove negligible variables at the first stage to reduce the model burden and to increase the performance as well.

Model	Identity	Log		B-C($\lambda = 0.1$)
	FDR	All	FDR	FDR
Model 1	0.112 (0.154)	0.165 (0.339)	0.066 (0.036)	0.067 (0.039)
Model 2	0.098 (0.097)	0.146 (0.323)	**0.061 (0.033)**	0.062 (0.034)
Model 3	0.103 (0.111)	0.157 (0.353)	0.066 (0.035)	0.066 (0.035)

Table 4: Validation MSE values

4 Discussion

Housing price prediction is a popular topic in various fields, but it is highly researched in the field of economics. This usually has to do with the impact of housing price on the economy.

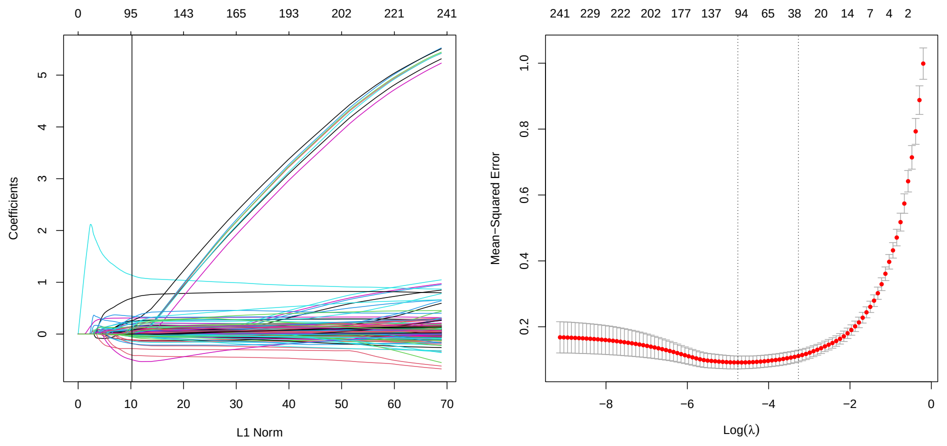

Figure 8: (Left) Result of `glmnet`, (Right) Result of `cv.glmnet`

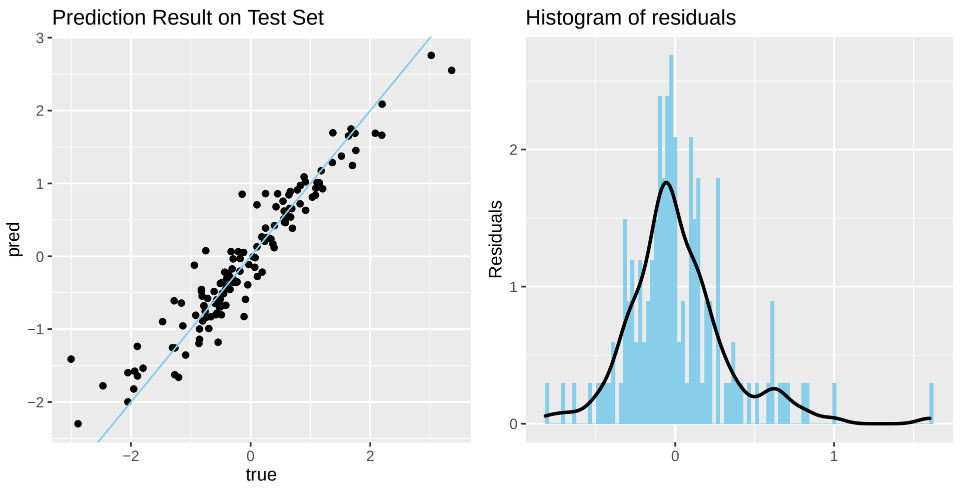

Figure 9: Diagnostic fitting result on the test set

From an individual level the fluctuation of housing prices may influence household consumption [8]. This is due to the changes in wealth portfolio of individuals becoming more reliant on housing than other forms of wealth (Greenspan and Kennedy, 2005). On a macroeconomic level, housing price fluctuation may have financial risk implications [9]. On both a microeconomic and macroeconomic level, it is important to be able to accurately predict housing prices. Our research, adds onto the growing literature on housing price forecast by implementing various statistical methods and considering direct house related variables. Despite the high interest in housing price prediction, building the optimal model for an accurate housing price forecast is challenging. This is due to the characteristics with housing data itself. As we have addressed, housing price itself is not normally distributed. There is much discussion as to the what distribution housing prices follow. One study found that without the presence of a bubble on the housing prices, they would roughly follow a lognormal distribution with a slightly fatter tail [5]. This study is quite consistent with our research where we found that the log transformation of the housing prices as the optimal transformation. Therefore, when building a housing price forecast model, transformation of the housing price is an essential step.

Another critical finding of this research is how the LASSO method had the best performance. In the dataset used for this study, we start off with 79 possible variables related to the houses. Due to the high number of variables there is the issue of missing data and multicollinearity. For instance, in the data set they had variables like 'GarageQual,' measuring garage quality, and

‘GarageCond,’ measuring garage condition. It is quite obvious that these two variables would have a high correlation. In addition, in this study we proposed a novel way of using multiple hypothesis testing to trim out negligible explanatory variables. To obtain this goal, we suggest controlling FDR to *discover* a set of significant predictors in less conservative way compared to Type-I error control methods. In our project, we applied this step to omit 6 variables at the beginning. We also need to understand a major difference between the Ridge and LASSO model. While for the Ridge model all of the variables are included with a shrinkage, in the LASSO model variables can be shirked to 0, meaning that they are not included in the model. From this we learn a valuable lesson in housing prices that people are not concerned about all aspects of the house. Rather it is important to clearly identify the variable that house buyers and sellers are concerned about. In conclusion, when building a housing prediction model, one should prioritize clearly identifying variables that would impact housing prices.

References

[1] Yoav Benjamini and Daniel Yekutieli. The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, 29(4):1165 – 1188, 2001.

[2] Dean De Cock. Ames, iowa: Alternative to the boston housing data as an end of semester regression project. *Journal of Statistics Education*, 19(3):null, 2011.

[3] J.J. Faraway. Linear models with r. *Chapman and Hall/CRC Press*, 2014.

[4] James Frew and G Jud. Estimating the value of apartment buildings. *Journal of Real Estate Research*, 25(1):77–86, 2003.

[5] Takaaki Ohnishi, Takayuki Mizuno, Chihiro Shimizu, and Tsutomu Watanabe. On the evolution of the house price distribution. 2011.

[6] Diana Olick. Existing home sales in 2020 hit highest point since 2006, but listings are at a record low. 2021.

[7] Kaggle Project. House prices-advanced regression techniques. `https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview`.

[8] J. K. Rapach, D. E. Strauss. Forecasting real housing price growth in the eighth district states. *Federal Reserve Bank of St. Louis. Regional Economic Development*, 3(2):33–42, 2007.

[9] Beibei Wang, Jingbin Xia and Huiling Qiao. Time-varying impact of housing price fluctuations on banking financial risk. *Managerial and Decision Economics*, 2021.

[10] In-Kwon Yeo and Richard A. Johnson. A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959, 2000.

APPENDIX

664_Final Code

Loading Library

```r
library(ggplot2)
library(dplyr)
library(zoo)
library(lubridate)
library(MASS)
library(grid)
library(gridExtra)
library(car)
library(tidyverse)
library(glmnet)
library(corrplot)
```

Defining function for Distribution Histogram

```r
dist.data = function(data, num=1, color = "skyblue", title=""){
  ggplot(data) +
  geom_histogram(aes(x=data[,num], y=..density..), fill = color, position="identity", bins = 100) +
  geom_density(aes(x=data[,num], y=..density..), size = 1)+ggtitle(title)+ylab('Residuals')+xlab('')
```

Defining function for QQ plot

```r
qq.data = function(data, color = "skyblue"){
  ggplot() +
  aes(sample = data) +
  geom_qq(distribution = qnorm) +
  geom_qq_line(col = color)+ylab('Residuals')+xlab('')
```

Loading and Cleaning the Data

```r
train <- as_tibble(read.csv("train.csv"))
train = train%>% dplyr::select(-"Id")
train$soldDate <- parse_date_time(paste(train$YrSold, sapply(train$MoSold,function(x){return(ifelse(x<10, paste("
0",x,sep=""),x))} ), sep="/"),"%Y/%m")
mean.month <- tapply(train$SalePrice,as.factor(train$soldDate), mean)
tr_means <- train %>%
  group_by(Neighborhood, YrSold) %>%
  summarise(mean = mean(log(SalePrice)))
```

Figure 1: Scatter plots of SalePrice along with some predictor variables

```r
p = list()
p[[1]]<-ggplot() + geom_point(aes(x=soldDate, y=SalePrice), data=train) +
  geom_line(aes(x = parse_date_time(names(mean.month),orders = "%Y-%m-%d"),
                y = mean.month),size=2,col="red") # plot_by_time
p[[2]]<-ggplot(train)+
  geom_point(aes(x = YearBuilt, y = SalePrice))
p[[3]]<-ggplot()+
  geom_point(aes(x = (Neighborhood), y = (SalePrice)), data= train)+
  scale_x_discrete(breaks= unique(train$Neighborhood) , labels= substring(unique(train$Neighborhood),1,2) )#plot_
by_neighbor

grid.arrange(grobs=p, ncol=1)
```

Figure 2: NA percentage of predictor variable

```r
X = train %>% dplyr::select(-"SalePrice")
Y = train$SalePrice

#missing values
missing = X %>%
  sapply(is.na) %>%
  colSums() %>%
  sort(decreasing = TRUE)
missing = missing/ nrow(X)*100
missing = stack(missing)
missing.with.NA=missing[missing$values>0,]
ggplot(data=missing.with.NA,aes(y=ind, x=values))+ ylab('Predictors') + xlab('Percentage of NA values') +
    geom_bar(stat="identity")
```

Replacing NA Values with Adequate Value

```
replace.with.none=function(var){
  train[var][is.na(train[var])]='None'
}
None.type=c('Garag
<<>>=
eType', 'GarageFinish', 'GarageQual', 'GarageCond', 'BsmtExposure', 'BsmtFinType2', 'BsmtQual', 'BsmtCond', 'Bsmt
FinType1')
replace.with.none(None.type)

#GarageYrBlt
train$GarageYrBlt[is.na(train$GarageType)] = 0

housing2=train %>%
  dplyr::select(-c('PoolQC', 'MiscFeature', 'Alley', 'Fence', 'FireplaceQu', 'LotFrontage','MasVnrType', 'MasVnrA
rea', 'Electrical'))
housing2= na.omit(housing2)
```

Data Transformation for Categorical Variables

```
# Ex/Gd/TA/Fa/Po
first.group = c('ExterQual', 'ExterCond', 'BsmtQual','BsmtCond','KitchenQual', 'HeatingQC','GarageQual','GarageCo
nd')
for (i in first.group){
  housing2[i] = c('Ex'=5,'Gd'=4,'TA'=3,'Fa'=2,'Po'=1,'None'=0)[housing2[i][[1]]]
  housing2[i] = as.ordered(housing2[i][[1]])
}

#GLQ/ALQ/BLQ/Rec/LWQ
second.group =c('BsmtFinType1', 'BsmtFinType2')
for (i in second.group){
  housing2[i] = c('GLQ'=6,'ALQ'=5,'BLQ'=4,'Rec'=3,'LwQ'=2,'Unf'=1,'None'=0)[housing2[i][[1]]]
  housing2[i] = as.ordered(housing2[i][[1]])
}

#Typ/Min1
housing2['Functional'] = c('Typ'=7,'Min1'=6,'Min2'=5,'Mod'=4,'Maj1'=3,'Maj2'=2,'Sev'=1,'Sal'=0)[housing2['Functio
nal'][[1]]]

housing2['LandSlope'] = c('Gtl'=3, 'Mod'=2, 'Sev'=1)[housing2['LandSlope'][[1]]]

housing2['GarageFinish'] = c('Fin'=3,'RFn'=2, 'Unf'=1, 'None'=0)[housing2['GarageFinish'][[1]]]

housing2['LotShape'] = c('Reg'=4, 'IR1'=3, 'IR2'=2,'IR3'=1)[housing2['LotShape'][[1]]]

housing2['PavedDrive'] = c('Y'=3, 'P'=2, 'N'=1)[housing2['PavedDrive'][[1]]]

housing2['BsmtExposure'] = c('Gd'=4,'Av'=3,'Mn'=2,'No'=1,'None'=0)[housing2['BsmtExposure'][[1]]]
third.group = c('Functional', 'LandSlope', 'GarageFinish', 'LotShape', 'PavedDrive', 'BsmtExposure')
for (i in third.group){
  housing2[i] = as.ordered(housing2[i][[1]])
}
for(i in colnames(housing2 %>% select_if(is.character))){
  housing2[i] = as.factor(housing2[i][[1]])
}

order.group = c('OverallQual','OverallCond',"BedroomAbvGr", "KitchenAbvGr", "TotRmsAbvGrd")
factor.group = c("YrSold","MoSold",'BsmtFullBath', "BsmtHalfBath", "FullBath", "HalfBath", "GarageCars","Fireplac
es")
for (i in order.group){
  housing2[i] = as.ordered(housing2[i][[1]])
}
for (i in factor.group){
  housing2[i] = as.factor(housing2[i][[1]])
}

housing2 = housing2%>%dplyr::select(-"soldDate")
```

FDR

```r
#FDR, Find discoveries
fac.var = housing2 %>% dplyr::select(is.factor)
pvals = sapply(fac.var, function(ff){
  lm.f = lm(SalePrice ~ ff, data= data.frame(ff, SalePrice = housing2$SalePrice))
  return(1-pf(summary(lm.f)$fstatistic[[1]],df1=summary(lm.f)$fstatistic[[2]],df2=summary(lm.f)$fstatistic[[3]]))
})
pvals = sort(pvals)
padj = p.adjust(pvals, method="BY")
padj[which(padj>0.2)]
```

Data Transformation for Numerical Variables & Figure 3: Distribution of numeric predictors without any transfrom

```r
num.var = housing2 %>% dplyr::select(is.numeric) %>% dplyr::select(-"SalePrice")
p = lapply(num.var,
           function(x){ggplot() +
  geom_histogram(aes(x=x, y=..density..), fill = "skyblue", position="identity",data =data.frame(x)) +
  geom_density(aes(x=x, y=..density..), size = 1,data =data.frame(x))})
for( i in 1:length(names(num.var))){
  p[[i]] = p[[i]] + xlab(names(num.var)[i])
}
lambda.vec =c()
for(ii in 1:length(num.var)){
  xx = num.var[[ii]]
  bc.mod <- car::boxCox(lm(xx~1,data=data.frame(xx)), family="yjPower")
  b.lambda<-bc.mod$x[which.max(bc.mod$y)]
  lambda.vec[ii]=b.lambda
  num.var[[ii]] = yjPower(xx, lambda=b.lambda, jacobian.adjusted=FALSE)
}

grid.arrange(grobs=p, ncol = 4)
#correlation
cormat = cor(data.frame(num.var,SalePrice=housing2$SalePrice)) # correlations of all numeric variables
cor.sorted = names(sort(abs(cormat[, 'SalePrice']), decreasing = TRUE))
cormat = cormat[cor.sorted[1:10],cor.sorted[1:10]]
rownames(cormat) = substring(rownames(cormat),1,5)
colnames(cormat) = substring(colnames(cormat),1,5)
corrplot.mixed(cormat, tl.col="black", tl.pos = "d")
#Multicollinearity
X = data.matrix(num.var)
X = scale(X)
XtX = t(X)%*%X
egdecomp <- eigen(XtX)
kappa = sqrt(egdecomp$values[1]/min(egdecomp$values))
vifs = (vif(lm(SalePrice~., data =data.frame(scale(num.var),SalePrice=scale(housing2$SalePrice)))))
order = sort(vifs, index.return =TRUE)
```

Encoding Categorical and Ordinal Variables

```r
drop = names(padj[which(padj>0.2)])
housing4 <- as_tibble(cbind(num.var, housing2%>%dplyr::select(is.factor)%>% dplyr::select(-drop),  SalePrice= hou
sing2$SalePrice ))
```

Variable Selection using AIC Stepwise Selection for Model with Response Variable Transformation.

```r
lmod <- lm(SalePrice ~ ., data = housing4)
model1 <- stepAIC(lmod, direction = "both", trace =0)
geo.mean <- exp(mean(log(housing4$SalePrice)))
C1 = geo.mean
logmod <- lm(C1*log(SalePrice) ~ ., data = housing4)
model2 <- stepAIC(logmod, direction = "both", trace =0)
bc.mod <- boxCox(model1,plotit = FALSE)
lambda<-bc.mod$x[which.max(bc.mod$y)]
print(lambda) # This is the best lambda by Box-Cox method selected in this step.
C2 = geo.mean^{1-lambda}
bcmod <- lm(C2*(SalePrice^lambda-1)/lambda ~ ., data = housing4)
model3 <- stepAIC( bcmod, direction = "both", trace = 0)
```

Table3: Calculating the AIC, BIC, PRESS Statistics for Each Model and Response Variable Transformation

```
col1 <-colnames(model1$model)
col2 <-colnames(model2$model)
col3 <-colnames(model3$model)
common = col1[col1 %in% col2]
common = common[common %in% col3]
dim(model1$model);dim(model2$model);dim(model3$model)
model = list()
model[["1orig"]] <- lm(SalePrice ~ ., data= model1$model)
model[["1log"]] <- lm(C1*log(SalePrice) ~ ., data= model1$model)
model[["1bc"]] <- lm(C2*(SalePrice^lambda-1)/lambda ~ ., data= model1$model)
data2 = (as_tibble(model2$model[,-1])) %>% mutate(SalePrice = housing4$SalePrice)
model[["2orig"]] <- lm(SalePrice ~ ., data= data2)
model[["2log"]] <- lm(C1*log(SalePrice) ~ ., data= data2)
model[["2bc"]] <- lm(C2*(SalePrice^lambda-1)/lambda ~ ., data= data2)
data2 = (as_tibble(model3$model[,-1])) %>% mutate(SalePrice = housing4$SalePrice)
model[["3orig"]] <- lm(SalePrice ~ ., data= data2)
model[["3log"]] <- lm(C1*log(SalePrice) ~ ., data= data2)
model[["3bc"]] <- lm(C2*(SalePrice^lambda-1)/lambda ~ ., data= data2)
sapply(model,AIC)
sapply(model,BIC)
sapply(model,function(x){sum((x$residuals/ hatvalues(x))^2)})
BIC(model1)
BIC(model2)
BIC(model3)
press1 <- sum((model1$residuals/ hatvalues(model1))^2)
press2 <- sum((model2$residuals/ hatvalues(model2))^2)
press3 <- sum((model3$residuals/ hatvalues(model3))^2)
```

Figure 5: Check Normality: histogram and the QQ-plot of residuals for three types of response variables

```
resi <- data.frame(orig = model1$residuals, log = model3$residuals, boxcox = model2$residuals)
p<-list()
p[[1]]<-dist.data(resi,num = 1, title = "Original data")
p[[4]]<- qq.data(resi$orig)
p[[2]]<- dist.data(resi,num=2, title = "Log Transform")
p[[5]]<- qq.data(resi$boxcox)
p[[3]]<- dist.data(resi,num = 3, title = "Box-Cox Transform")
p[[6]]<- qq.data(resi$log)
grid.arrange(grobs=p,ncol=3)
```

Figure 6: Residual plots of Log transformed

```
par(mar=c(4,4,1,1))
par(mfrow=c(2,2))
plot(model2)
```

Outlier Test

```
outlierTest(model2)
```

Figure 7: Outliers observation

```
outlierTest(model2)
ot <- outlierTest(model2)
outliers = as.integer(names(ot[[1]]))
housing2[outliers,] %>% dplyr::select(c("SalePrice", "YrSold"))
housing2$soldDate <- parse_date_time(paste(housing2$YrSold,sapply(housing2$MoSold, function(x){return(ifelse(as.n
umeric(x)<10, paste("0",x,sep=""),x))} ), sep="/"),"%Y/%m")
p = list()
p[[1]] <- ggplot() + geom_point(aes(x=OverallQual, y=SalePrice), data=housing2[-outliers,]) + geom_point(aes(x=Ov
erallQual, y=SalePrice), data=housing4[outliers,],col = 'red',size=2)+ geom_label(aes(x=OverallQual, y=SalePrice,
label = outliers), data=housing2[outliers,],col = 'red')
p[[2]] <- ggplot() + geom_point(aes(x=YearBuilt, y=SalePrice), data=housing2[-outliers,]) + geom_point(aes(x=Year
Built, y=SalePrice), data=housing2[outliers,],col = 'red',size=2)+ geom_label(aes(x=YearBuilt, y=SalePrice, label
= outliers), data=housing2[outliers,],col = 'red')
p[[3]] <- ggplot() + geom_point(aes(x=Neighborhood, y=SalePrice), data=housing2[-outliers,]) + geom_point(aes(x=N
eighborhood, y=SalePrice), data=housing2[outliers,],col = 'red',size=2)+ geom_label(aes(x=Neighborhood, y=SalePri
ce, label = outliers), data=housing2[outliers,],col = 'red') + scale_x_discrete(breaks= unique(train$Neighborhood
) , labels= substring(unique(train$Neighborhood),1,2) )
grid.arrange(grobs=p, ncol=1)
```

Dividing Test and Train Data & Standardization

```r
library(glmnet)
library(pls)

housing5 = housing4[-outliers,]

housing5 = housing5 %>% mutate(y.id= SalePrice, y.log = log(SalePrice), y.bc = bcPower(SalePrice, lambda=lambda,
jacobian.adjusted=FALSE)) %>% dplyr::select(-"SalePrice")



#step3 : Test set - Most recent 10% data
recent.idx = order(housing2[-outliers,]$soldDate,decreasing=TRUE)[1:(nrow(housing5)/10)]
Test = housing5[recent.idx,]
Train = housing5[-recent.idx,]

#step4 : standardize
mean = apply(housing5 %>% dplyr::select(where(is.numeric)),2,mean)
sd = apply(housing5 %>% dplyr::select(where(is.numeric)),2,sd)
Train = Train %>% mutate_at(colnames(housing5 %>% dplyr::select(where(is.numeric))), ~(scale(.) %>% as.vector))
Test= as_tibble(data.frame(as.data.frame(t((t(Test %>% dplyr::select(where(is.numeric)))-mean)/sd)), Test%>% dply
r::select(where(is.factor))))
```

Cross-validation: Just Change response variables and apply this three times

```r
library(glmnet)
set.seed(2021)
smp = sample(1:100,nrow(Train),replace = T)
#Ridge and LASSO and PCR
mmx = model.matrix(y.log ~ . -y.id - y.bc, data=Train)
COM = NULL
for(i in 1:20){
  random_sample = which(smp!=i)
#Ridge Fitting
  cvfit.r = cv.glmnet(mmx[random_sample,-1],Train$y.log[random_sample],alpha=0, nfolds=50)
lambda.r = cvfit.r$lambda.min
fit.r=cvfit.r$glmnet.fit
y_predicted.r = predict(fit.r,s=lambda.r,newx = (mmx[-random_sample,-1]))
COM$ridge[i] = (mean((y_predicted.r-(Train$y.log[-random_sample]))^2))
#LASSO fitting
cvfit.l=cv.glmnet(mmx[random_sample,-1],Train$y.log[random_sample],alpha=1, nfolds=50)
lambda.l = cvfit.l$lambda.min
fit.l=cvfit.l$glmnet.fit
y_predicted.l = predict(fit.l,s=lambda.l,newx = (mmx[-random_sample,-1]))
COM$lasso[i]= (mean((y_predicted.l-(Train$y.log[-random_sample]))^2))
#PCR fitting
set.seed(2021)
pcrcv <- pcr(y.log ~ . -y.id - y.bc, data=Train[random_sample,], validation = "CV")
pcrmse = RMSEP(pcrcv)
r.min = which.min(pcrmse$val[1,1,])-1# choose
mmx.pcr = model.matrix(y.log ~ . -y.id - y.bc, data=Train[,])
pcr.orig = prcomp(mmx.pcr[,-1])

PCRtrain = Train %>% dplyr::select(-where(is.numeric)) %>% mutate(y.log = Train$y.log)
PCRtrain = as_tibble(cbind(as.data.frame(PCRtrain), pcr.orig$x[,1:r.min]) )
mmx.pcr = model.matrix(y.log ~ . , data=PCRtrain)
cvfit.pcr=cv.glmnet(mmx.pcr[random_sample,-1],Train$y.log[random_sample],alpha=1, nfolds=50)
lambda.pcr = cvfit.pcr$lambda.min
fit.pcr=cvfit.pcr$glmnet.fit
y_predicted.pcr = predict(fit.pcr,s=lambda.pcr,newx = (mmx.pcr[-random_sample,-1]))
COM$pcr[i] = (mean((y_predicted.pcr-(Train$y.log[-random_sample]))^2))
print(i)
}

#result
a = round(rbind(sapply(COM,mean),sapply(COM,mean),sapply(COM,mean)),3) #mean
b = round(rbind(sapply(COM,sd),sapply(COM,sd),sapply(COM,sd)),3)# sd
rownames(a) = c("ID","Log","BC")
a;b
```

Figure 8, 9

```r
cvfit.l=cv.glmnet(mmx[,-1],Train$y.log,alpha=1, nfolds=50)
lambda.l = cvfit.l$lambda.min
fit.l=cvfit.l$glmnet.fit
test.m =  model.matrix(y.log ~ . -y.id - y.bc, data=Test)
y_predicted.l = predict(fit.l,s=lambda.l,newx = test.m[,-1])
(mean((y_predicted.l-(Test$y.log))^2))
pred = y_predicted.l
true = (housing5$y.log[recent.idx] - mean( housing5$y.log[-recent.idx]) )/sd( housing5$y.log[-recent.idx])
#Figure 9
p = list()
p[[1]] = ggplot(data.frame( pred = y_predicted.l, true = true)) + geom_point(aes(x= true, y = pred)) +geom_abline
(slope=1, col='skyblue')+ggtitle('Prediction Result on Test Set')
p[[2]] = dist.data(data.frame(y_predicted.l-true), title = "Histogram of residuals")
grid.arrange(grobs=p, ncol=2)

save(cvfit.l, fit.l, y_predicted.l, true, "beforefdr.rda")
mse.before=mean((y_predicted.l-true)^2)

#figure 8
mse.fdr = mean((y_predicted.l-true)^2)
par(mfrow= c(1,2))
ll = length(coef(cvfit.l, s= lambda.l))
l11 = sum(abs(coef(cvfit.l, s = lambda.l)[2:ll]))
plot(fit.l)
lines(c(l11,l11),c(-10,10) )
plot(cvfit.l)
```