The Ames Iowa House Price Prediction

Group Member: Yahui Zhou, Jiaxin Yang, Jiangyue Zhu, Jike Zhong, Tingwei Guan

2022-11-24

Part I: Exploratory Data Analysis

Part I.A Check data - (1) Basic information about data

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

print("Basic information for training dataset")
```

```
## [1] "Basic information for training dataset"
```

```
print(is.data.frame(train))
```

```
## [1] TRUE
```

```
print(dim(train))
```

```
## [1] 1460   81
```

```
print("Basic information for training dataset")
```

```
## [1] "Basic information for training dataset"
```

```
print(is.data.frame(test))
```

```
## [1] TRUE
```

```
print(dim(test))
```

```
## [1] 1447   81
```

```
str(train)
```

```
## 'data.frame':    1460 obs. of  81 variables:
##  $ Id           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ MSSubClass   : int  60 20 60 70 60 50 20 60 50 190 ...
##  $ MSZoning     : chr  "RL" "RL" "RL" "RL" ...
##  $ LotFrontage  : int  65 80 68 60 84 85 75 NA 51 50 ...
##  $ LotArea      : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 ...
##  $ Street       : chr  "Pave" "Pave" "Pave" "Pave" ...
##  $ Alley        : chr  NA NA NA NA ...
##  $ LotShape     : chr  "Reg" "Reg" "IR1" "IR1" ...
##  $ LandContour  : chr  "Lvl" "Lvl" "Lvl" "Lvl" ...
##  $ Utilities    : chr  "AllPub" "AllPub" "AllPub" "AllPub" ...
##  $ LotConfig    : chr  "Inside" "FR2" "Inside" "Corner" ...
##  $ LandSlope    : chr  "Gtl" "Gtl" "Gtl" "Gtl" ...
##  $ Neighborhood : chr  "CollgCr" "Veenker" "CollgCr" "Crawfor" ...
##  $ Condition1   : chr  "Norm" "Feedr" "Norm" "Norm" ...
##  $ Condition2   : chr  "Norm" "Norm" "Norm" "Norm" ...
##  $ BldgType     : chr  "1Fam" "1Fam" "1Fam" "1Fam" ...
##  $ HouseStyle   : chr  "2Story" "1Story" "2Story" "2Story" ...
##  $ OverallQual  : int  7 6 7 7 8 5 8 7 7 5 ...
##  $ OverallCond  : int  5 8 5 5 5 5 5 6 5 6 ...
##  $ YearBuilt    : int  2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 ...
##  $ YearRemodAdd : int  2003 1976 2002 1970 2000 1995 2005 1973 1950 1950 ...
##  $ RoofStyle    : chr  "Gable" "Gable" "Gable" "Gable" ...
##  $ RoofMatl     : chr  "CompShg" "CompShg" "CompShg" "CompShg" ...
##  $ Exterior1st  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Sdng" ...
##  $ Exterior2nd  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Shng" ...
##  $ MasVnrType   : chr  "BrkFace" "None" "BrkFace" "None" ...
##  $ MasVnrArea   : int  196 0 162 0 350 0 186 240 0 0 ...
##  $ ExterQual    : chr  "Gd" "TA" "Gd" "TA" ...
##  $ ExterCond    : chr  "TA" "TA" "TA" "TA" ...
##  $ Foundation   : chr  "PConc" "CBlock" "PConc" "BrkTil" ...
##  $ BsmtQual     : chr  "Gd" "Gd" "Gd" "TA" ...
##  $ BsmtCond     : chr  "TA" "TA" "TA" "Gd" ...
##  $ BsmtExposure : chr  "No" "Gd" "Mn" "No" ...
##  $ BsmtFinType1 : chr  "GLQ" "ALQ" "GLQ" "ALQ" ...
##  $ BsmtFinSF1   : int  706 978 486 216 655 732 1369 859 0 851 ...
##  $ BsmtFinType2 : chr  "Unf" "Unf" "Unf" "Unf" ...
##  $ BsmtFinSF2   : int  0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF    : int  150 284 434 540 490 64 317 216 952 140 ...
##  $ TotalBsmtSF  : int  856 1262 920 756 1145 796 1686 1107 952 991 ...
##  $ Heating      : chr  "GasA" "GasA" "GasA" "GasA" ...
##  $ HeatingQC    : chr  "Ex" "Ex" "Ex" "Gd" ...
##  $ CentralAir   : chr  "Y" "Y" "Y" "Y" ...
##  $ Electrical   : chr  "SBrkr" "SBrkr" "SBrkr" "SBrkr" ...
##  $ X1stFlrSF    : int  856 1262 920 961 1145 796 1694 1107 1022 1077 ...
##  $ X2ndFlrSF    : int  854 0 866 756 1053 566 0 983 752 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  1710 1262 1786 1717 2198 1362 1694 2090 1774 1077 ...
##  $ BsmtFullBath : int  1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  2 2 2 1 2 1 2 2 2 1 ...
##  $ HalfBath     : int  1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr : int  3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 2 2 ...
##  $ KitchenQual  : chr  "Gd" "TA" "Gd" "Gd" ...
##  $ TotRmsAbvGrd : int  8 6 6 7 9 5 7 7 8 5 ...
##  $ Functional   : chr  "Typ" "Typ" "Typ" "Typ" ...
##  $ Fireplaces   : int  0 1 1 1 1 0 1 2 2 2 ...
##  $ FireplaceQu  : chr  NA "TA" "TA" "Gd" ...
##  $ GarageType   : chr  "Attchd" "Attchd" "Attchd" "Detchd" ...
##  $ GarageYrBlt  : int  2003 1976 2001 1998 2000 1993 2004 1973 1931 1939 ...
##  $ GarageFinish : chr  "RFn" "RFn" "RFn" "Unf" ...
##  $ GarageCars   : int  2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea   : int  548 460 608 642 836 480 636 484 468 205 ...
##  $ GarageQual   : chr  "TA" "TA" "TA" "TA" ...
##  $ GarageCond   : chr  "TA" "TA" "TA" "TA" ...
##  $ PavedDrive   : chr  "Y" "Y" "Y" "Y" ...
##  $ WoodDeckSF   : int  0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF  : int  61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch: int  0 0 0 272 0 0 0 228 205 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : chr  NA NA NA NA ...
##  $ Fence        : chr  NA NA NA NA ...
##  $ MiscFeature  : chr  NA NA NA NA ...
##  $ MiscVal      : int  0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold       : int  2 5 9 2 12 10 8 11 4 1 ...
##  $ YrSold       : int  2008 2007 2008 2006 2008 2009 2007 2009 2008 2008 ...
##  $ SaleType     : chr  "WD" "WD" "WD" "WD" ...
##  $ SaleCondition: chr  "Normal" "Normal" "Normal" "Abnorml" ...
##  $ SalePrice    : int  208500 181500 223500 140000 250000 143000 307000 200000 129900 118000 ...
```

Part I.A Check data - (2) Data type

After checking the data, we found that the data type in the dataset is not very accurate. Some categorical data is stored as numeric type. Thus, we manually listed out the categorical and numerical variables and correct the data type accordingly.

We have two types of variables: 52 Categorical variables, and 28 Numeric variables.

Categorical variables (52)
– Nominal (27)
– Ordinal (25)

Nominal (27): MSSubClass, MSZoning, Street, Alley, LotShape, LandContour, Utilities, LotConfig, Neighborhood, Condition1, Condition2, BldgType, HouseStyle, RoofStyle, CentralAir, RoofMatl, Exterior1st, Exterior2nd, MasVnrType, Foundation, Heating, Functional, GarageType, Fence, MiscFeature, SaleType, SaleCondition.

Ordinal (25): id, LandSlope, OverallQual, OverallCond, YearBuilt, YearRemodAdd, ExterQual, ExterCond, BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, HeatingQC, Electrical, KitchenQual, FireplaceQu, GarageYrBlt, GarageFinish, GarageQual, GarageCond, PoolQC, MoSold, YrSold, PavedDrive.

Numeric variables (28): LotFrontage, LotArea, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtFullBath, BsmtHalfBath, FullBath, HalfBath, Bedroom, Kitchen, TotRmsAbvGrd, Fireplaces, GarageCars, GarageArea, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, MiscVal.

```r
# check data type
# find categorical and numerical variables
category_col = c("Id", "LandSlope", "OverallQual", "OverallCond", "YearBuilt",
                 "YearRemodAdd", "ExterQual", "ExterCond", "BsmtQual", "BsmtCond",
                 "BsmtExposure", "BsmtFinType1", "BsmtFinType2", "HeatingQC",
                 "Electrical", "KitchenQual", "FireplaceQu", "GarageFinish",
                 "GarageQual", "GarageCond", "PoolQC", "MoSold", "YrSold", "PavedDrive",
                 "MSSubClass", "MSZoning", "Street", "Alley", "LotShape", "LandContour",
                 "Utilities", "LotConfig", "Neighborhood", "Condition1", "Condition2",
                 "BldgType", "HouseStyle", "RoofStyle", "CentralAir", "RoofMatl",
                 "Exterior1st", "Exterior2nd", "MasVnrType", "Foundation", "Heating",
                 "Functional", "GarageType", "Fence", "MiscFeature", "SaleType",
                 "SaleCondition")
numeric_col = c("LotFrontage", "LotArea", "MasVnrArea", "BsmtFinSF1", "BsmtFinSF2",
                "BsmtUnfSF", "TotalBsmtSF", "X1stFlrSF", "X2ndFlrSF", "LowQualFinSF",
                "GrLivArea", "BsmtFullBath", "BsmtHalfBath", "FullBath", "HalfBath",
                "BedroomAbvGr", "KitchenAbvGr", "TotRmsAbvGrd", "Fireplaces",
                "GarageCars", "GarageArea", "WoodDeckSF", "OpenPorchSF", "EnclosedPorch",
                "X3SsnPorch", "ScreenPorch", "PoolArea", "MiscVal", "GarageYrBlt")
# change the type of variable
train = train %>% mutate_at(category_col, as.character)
train = train %>% mutate_at(numeric_col, as.integer)

# check whether the type is changed successfully
str(train)
```

```
## 'data.frame':    1460 obs. of  81 variables:
##  $ Id           : chr  "1" "2" "3" "4" ...
##  $ MSSubClass   : chr  "60" "20" "60" "70" ...
##  $ MSZoning     : chr  "RL" "RL" "RL" "RL" ...
##  $ LotFrontage  : int  65 80 68 60 84 85 75 NA 51 50 ...
##  $ LotArea      : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 ...
##  $ Street       : chr  "Pave" "Pave" "Pave" "Pave" ...
##  $ Alley        : chr  NA NA NA NA ...
##  $ LotShape     : chr  "Reg" "Reg" "IR1" "IR1" ...
##  $ LandContour  : chr  "Lvl" "Lvl" "Lvl" "Lvl" ...
##  $ Utilities    : chr  "AllPub" "AllPub" "AllPub" "AllPub" ...
##  $ LotConfig    : chr  "Inside" "FR2" "Inside" "Corner" ...
##  $ LandSlope    : chr  "Gtl" "Gtl" "Gtl" "Gtl" ...
##  $ Neighborhood : chr  "CollgCr" "Veenker" "CollgCr" "Crawfor" ...
##  $ Condition1   : chr  "Norm" "Feedr" "Norm" "Norm" ...
##  $ Condition2   : chr  "Norm" "Norm" "Norm" "Norm" ...
##  $ BldgType     : chr  "1Fam" "1Fam" "1Fam" "1Fam" ...
##  $ HouseStyle   : chr  "2Story" "1Story" "2Story" "2Story" ...
##  $ OverallQual  : chr  "7" "6" "7" "7" ...
##  $ OverallCond  : chr  "5" "8" "5" "5" ...
##  $ YearBuilt    : chr  "2003" "1976" "2001" "1915" ...
##  $ YearRemodAdd : chr  "2003" "1976" "2002" "1970" ...
##  $ RoofStyle    : chr  "Gable" "Gable" "Gable" "Gable" ...
##  $ RoofMatl     : chr  "CompShg" "CompShg" "CompShg" "CompShg" ...
##  $ Exterior1st  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Sdng" ...
##  $ Exterior2nd  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Shng" ...
##  $ MasVnrType   : chr  "BrkFace" "None" "BrkFace" "None" ...
##  $ MasVnrArea   : int  196 0 162 0 350 0 186 240 0 0 ...
##  $ ExterQual    : chr  "Gd" "TA" "Gd" "TA" ...
##  $ ExterCond    : chr  "TA" "TA" "TA" "TA" ...
##  $ Foundation   : chr  "PConc" "CBlock" "PConc" "BrkTil" ...
##  $ BsmtQual     : chr  "Gd" "Gd" "Gd" "TA" ...
##  $ BsmtCond     : chr  "TA" "TA" "TA" "Gd" ...
##  $ BsmtExposure : chr  "No" "Gd" "Mn" "No" ...
##  $ BsmtFinType1 : chr  "GLQ" "ALQ" "GLQ" "ALQ" ...
##  $ BsmtFinSF1   : int  706 978 486 216 655 732 1369 859 0 851 ...
##  $ BsmtFinType2 : chr  "Unf" "Unf" "Unf" "Unf" ...
##  $ BsmtFinSF2   : int  0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF    : int  150 284 434 540 490 64 317 216 952 140 ...
##  $ TotalBsmtSF  : int  856 1262 920 756 1145 796 1686 1107 952 991 ...
##  $ Heating      : chr  "GasA" "GasA" "GasA" "GasA" ...
##  $ HeatingQC    : chr  "Ex" "Ex" "Ex" "Gd" ...
##  $ CentralAir   : chr  "Y" "Y" "Y" "Y" ...
##  $ Electrical   : chr  "SBrkr" "SBrkr" "SBrkr" "SBrkr" ...
##  $ X1stFlrSF    : int  856 1262 920 961 1145 796 1694 1107 1022 1077 ...
##  $ X2ndFlrSF    : int  854 0 866 756 1053 566 0 983 752 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  1710 1262 1786 1717 2198 1362 1694 2090 1774 1077 ...
##  $ BsmtFullBath : int  1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  2 2 2 1 2 1 2 2 2 1 ...
##  $ HalfBath     : int  1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr : int  3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 2 2 ...
##  $ KitchenQual  : chr  "Gd" "TA" "Gd" "Gd" ...
##  $ TotRmsAbvGrd : int  8 6 6 7 9 5 7 7 8 5 ...
##  $ Functional   : chr  "Typ" "Typ" "Typ" "Typ" ...
##  $ Fireplaces   : int  0 1 1 1 1 0 1 2 2 2 ...
##  $ FireplaceQu  : chr  NA "TA" "TA" "Gd" ...
##  $ GarageType   : chr  "Attchd" "Attchd" "Attchd" "Detchd" ...
##  $ GarageYrBlt  : int  2003 1976 2001 1998 2000 1993 2004 1973 1931 1939 ...
##  $ GarageFinish : chr  "RFn" "RFn" "RFn" "Unf" ...
##  $ GarageCars   : int  2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea   : int  548 460 608 642 836 480 636 484 468 205 ...
##  $ GarageQual   : chr  "TA" "TA" "TA" "TA" ...
##  $ GarageCond   : chr  "TA" "TA" "TA" "TA" ...
##  $ PavedDrive   : chr  "Y" "Y" "Y" "Y" ...
##  $ WoodDeckSF   : int  0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF  : int  61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch: int  0 0 0 272 0 0 0 228 205 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : chr  NA NA NA NA ...
##  $ Fence        : chr  NA NA NA NA ...
##  $ MiscFeature  : chr  NA NA NA NA ...
##  $ MiscVal      : int  0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold       : chr  "2" "5" "9" "2" ...
##  $ YrSold       : chr  "2008" "2007" "2008" "2006" ...
##  $ SaleType     : chr  "WD" "WD" "WD" "WD" ...
##  $ SaleCondition: chr  "Normal" "Normal" "Normal" "Abnorml" ...
## $ SalePrice     : int  208500 181500 223500 140000 250000 143000 307000 200000 129900 118000 ...
```

Here, we can see the data type are correct.

Part I.A Check data - (3) duplicate / null value

```
# check number of duplicated records
sum(duplicated(train))
```

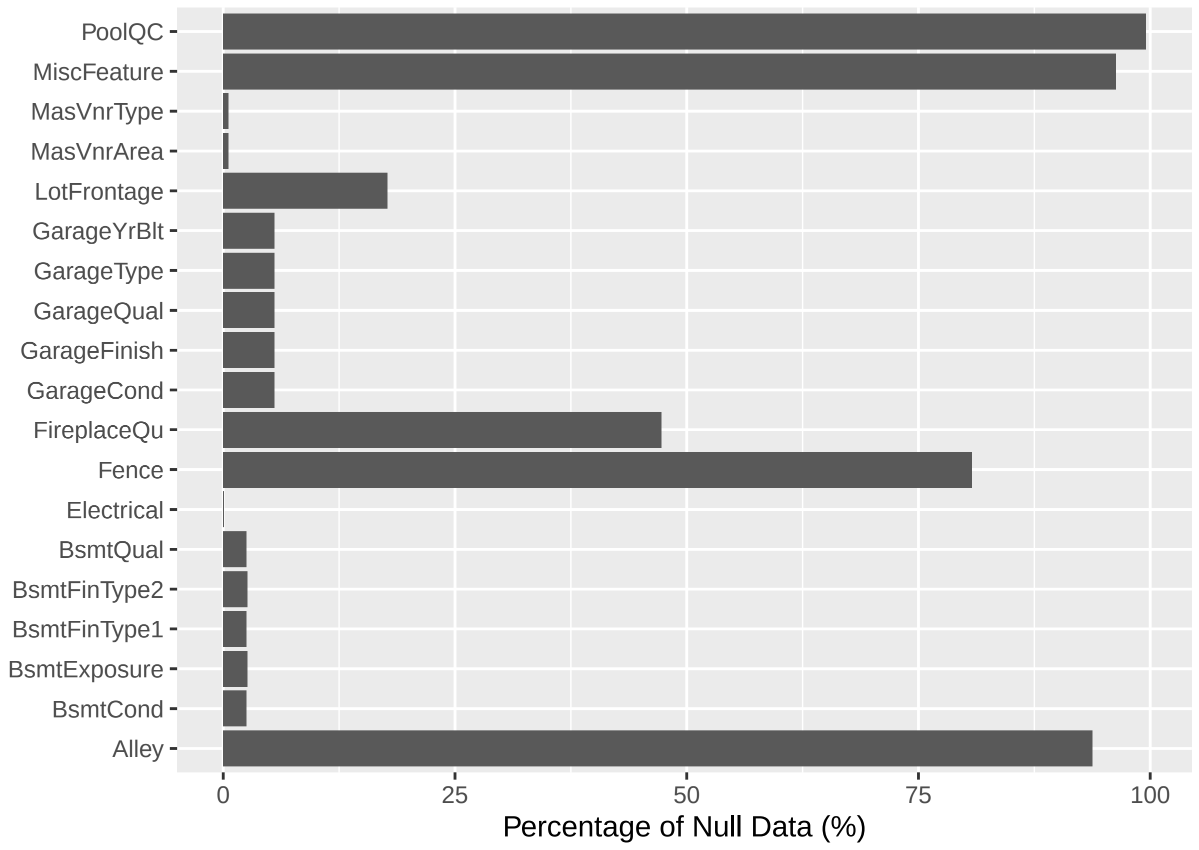
```
## [1] 0
```

```
# check number/percentage of NA data
na_per = c()
col_names = c()
for (i in 2: 80) {
  if (sum(is.na(train[,i]))/dim(train)[1]*100 > 0) {
    na_per = append(na_per, sum(is.na(train[,i]))/dim(train)[1]*100)
    col_names = append(col_names, colnames(train)[i])
  }
}

# draw the visualization to see percentage of NA
df = as.data.frame(col_names, na_per)
```

```
## Warning in as.data.frame.vector(x, ..., nm = nm): 'row.names' is not a character
## vector of length 19 -- omitting it. Will be an error!
```

```
pt = ggplot(data = df, aes(x = na_per, y = col_names)) +
      geom_bar(stat="identity") +
      labs(x = "Percentage of Null Data (%)", y = "")
pt
```



In the Data Processing section, we first checked the duplicated records and found there is no duplicated records. Then, we checked the number and percentage of null values for each variable. We built a visualization for the variables with null values. After checking the data, we found that NA does not only stands for the missing data. For the categorical data, we found that NA means "Not Accessible". Thus, we replaced the "NA" in the categorical variables with "No". For the numeric variables, NA can mean 0 or missing. Thus, we decide to use the medium value to replace "NA".

```
# replace the null value
train <- train %>% mutate_if(is.numeric, function(x) ifelse(is.na(x), median(x, na.rm=T),x))
train <- train %>% mutate_if(is.character, ~replace_na(., "No"))
test <- test %>% mutate_if(is.numeric, function(x) ifelse(is.na(x), median(x, na.rm=T),x))
test <- test %>% mutate_if(is.character, ~replace_na(., "No"))
print(dim(train))
```

```
## [1] 1460   81
```

```
print(dim(test))
```

```
## [1] 1447   81
```

```
head(train)
```

```
##   Id MSSubClass MSZoning LotFrontage LotArea Street Alley LotShape LandContour
## 1  1         60       RL          65    8450   Pave    No      Reg         Lvl
## 2  2         20       RL          80    9600   Pave    No      Reg          Lvl
## 3  3         60       RL          68   11250   Pave    No      IR1          Lvl
## 4  4         70       RL          60    9550   Pave    No      IR1          Lvl
## 5  5         60       RL          84   14260   Pave    No      IR1          Lvl
## 6  6         50       RL          85   14115   Pave    No      IR1          Lvl
##   Utilities LotConfig LandSlope Neighborhood Condition1 Condition2 BldgType
## 1    AllPub    Inside       Gtl      CollgCr       Norm       Norm     1Fam
## 2    AllPub       FR2       Gtl      Veenker      Feedr       Norm     1Fam
## 3    AllPub    Inside       Gtl      CollgCr       Norm       Norm     1Fam
## 4    AllPub    Corner       Gtl      Crawfor       Norm       Norm     1Fam
## 5    AllPub       FR2       Gtl      NoRidge       Norm       Norm     1Fam
## 6    AllPub    Inside       Gtl      Mitchel       Norm       Norm     1Fam
##   HouseStyle OverallQual OverallCond YearBuilt YearRemodAdd RoofStyle RoofMatl
## 1     2Story           7           5      2003         2003     Gable  CompShg
## 2     1Story           6           8      1976         1976     Gable  CompShg
## 3     2Story           7           5      2001         2002     Gable  CompShg
## 4     2Story           7           5      1915         1970     Gable  CompShg
## 5     2Story           8           5      2000         2000     Gable  CompShg
## 6     1.5Fin           5           5      1993         1995     Gable  CompShg
##   Exterior1st Exterior2nd MasVnrType MasVnrArea ExterQual ExterCond Foundation
## 1     VinylSd     VinylSd    BrkFace        196        Gd        TA      PConc
## 2     MetalSd     MetalSd       None          0        TA        TA     CBlock
## 3     VinylSd     VinylSd    BrkFace        162        Gd        TA      PConc
## 4     Wd Sdng     Wd Shng       None          0        TA        TA     BrkTil
## 5     VinylSd     VinylSd    BrkFace        350        Gd        TA      PConc
## 6     VinylSd     VinylSd       None          0        TA        TA       Wood
##   BsmtQual BsmtCond BsmtExposure BsmtFinType1 BsmtFinSF1 BsmtFinType2
## 1       Gd       TA           No          GLQ        706          Unf
## 2       Gd       TA           Gd          ALQ        978          Unf
## 3       Gd       TA           Mn          GLQ        486          Unf
## 4       TA       Gd           No          ALQ        216          Unf
## 5       Gd       TA           Av          GLQ        655          Unf
## 6       Gd       TA           No          GLQ        732          Unf
##   BsmtFinSF2 BsmtUnfSF TotalBsmtSF Heating HeatingQC CentralAir Electrical
## 1          0       150         856    GasA        Ex          Y      SBrkr
## 2          0       284        1262    GasA        Ex          Y      SBrkr
## 3          0       434         920    GasA        Ex          Y      SBrkr
## 4          0       540         756    GasA        Gd          Y      SBrkr
## 5          0       490        1145    GasA        Ex          Y      SBrkr
## 6          0        64         796    GasA        Ex          Y      SBrkr
##   X1stFlrSF X2ndFlrSF LowQualFinSF GrLivArea BsmtFullBath BsmtHalfBath FullBath
## 1       856       854            0      1710            1            0        2
## 2      1262         0            0      1262            0            1        2
## 3       920       866            0      1786            1            0        2
## 4       961       756            0      1717            1            0        1
## 5      1145      1053            0      2198            1            0        2
## 6       796       566            0      1362            1            0        1
##   HalfBath BedroomAbvGr KitchenAbvGr KitchenQual TotRmsAbvGrd Functional
## 1        1            3            1          Gd            8        Typ
## 2        0            3            1          TA            6        Typ
## 3        1            3            1          Gd            6        Typ
## 4        0            3            1          Gd            7        Typ
## 5        1            4            1          Gd            9        Typ
## 6        1            1            1          TA            5        Typ
##   Fireplaces FireplaceQu GarageType GarageYrBlt GarageFinish GarageCars
## 1          0          No     Attchd        2003          RFn          2
## 2          1          TA     Attchd        1976          RFn          2
## 3          1          TA     Attchd        2001          RFn          2
## 4          1          Gd     Detchd        1998          Unf          3
## 5          1          TA     Attchd        2000          RFn          3
## 6          0          No     Attchd        1993          Unf          2
##   GarageArea GarageQual GarageCond PavedDrive WoodDeckSF OpenPorchSF
## 1        548         TA         TA          Y          0          61
## 2        460         TA         TA          Y        298           0
## 3        608         TA         TA          Y          0          42
## 4        642         TA         TA          Y          0          35
## 5        836         TA         TA          Y        192          84
## 6        480         TA         TA          Y         40          30
##   EnclosedPorch X3SsnPorch ScreenPorch PoolArea PoolQC Fence MiscFeature
## 1             0          0           0        0     No    No          No
## 2             0          0           0        0     No    No          No
## 3             0          0           0        0     No    No          No
## 4           272          0           0        0     No    No          No
## 5             0          0           0        0     No    No          No
## 6             0        320           0        0     No MnPrv        Shed
##   MiscVal MoSold YrSold SaleType SaleCondition SalePrice
## 1       0      2   2008       WD        Normal    208500
## 2       0      5   2007       WD        Normal    181500
## 3       0      9   2008       WD        Normal    223500
## 4       0      2   2006       WD       Abnorml    140000
## 5       0     12   2008       WD        Normal    250000
## 6     700     10   2009       WD        Normal    143000
```

```
# check whether all the null values are solved
sum(is.na(train))
```

```
## [1] 0
```

```
sum(is.na(test))
```

```
## [1] 0
```

```
# We treated all the year as categorical value.
# Thus, here we change year back to categorical value after filling the null data
train$GarageYrBlt = as.character(train$GarageYrBlt)
test$GarageYrBlt = as.character(test$GarageYrBlt)
str(train)
```

```
## 'data.frame':    1460 obs. of  81 variables:
##  $ Id           : chr  "1" "2" "3" "4" ...
##  $ MSSubClass   : chr  "60" "20" "60" "70" ...
##  $ MSZoning     : chr  "RL" "RL" "RL" "RL" ...
##  $ LotFrontage  : int  65 80 68 60 84 85 75 69 51 50 ...
##  $ LotArea      : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 ...
##  $ Street       : chr  "Pave" "Pave" "Pave" "Pave" ...
##  $ Alley        : chr  "No" "No" "No" "No" ...
##  $ LotShape     : chr  "Reg" "Reg" "IR1" "IR1" ...
##  $ LandContour  : chr  "Lvl" "Lvl" "Lvl" "Lvl" ...
##  $ Utilities    : chr  "AllPub" "AllPub" "AllPub" "AllPub" ...
##  $ LotConfig    : chr  "Inside" "FR2" "Inside" "Corner" ...
##  $ LandSlope    : chr  "Gtl" "Gtl" "Gtl" "Gtl" ...
##  $ Neighborhood : chr  "CollgCr" "Veenker" "CollgCr" "Crawfor" ...
##  $ Condition1   : chr  "Norm" "Feedr" "Norm" "Norm" ...
##  $ Condition2   : chr  "Norm" "Norm" "Norm" "Norm" ...
##  $ BldgType     : chr  "1Fam" "1Fam" "1Fam" "1Fam" ...
##  $ HouseStyle   : chr  "2Story" "1Story" "2Story" "2Story" ...
##  $ OverallQual  : chr  "7" "6" "7" "7" ...
##  $ OverallCond  : chr  "5" "8" "5" "5" ...
##  $ YearBuilt    : chr  "2003" "1976" "2001" "1915" ...
##  $ YearRemodAdd : chr  "2003" "1976" "2002" "1970" ...
##  $ RoofStyle    : chr  "Gable" "Gable" "Gable" "Gable" ...
##  $ RoofMatl     : chr  "CompShg" "CompShg" "CompShg" "CompShg" ...
##  $ Exterior1st  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Sdng" ...
##  $ Exterior2nd  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Shng" ...
##  $ MasVnrType   : chr  "BrkFace" "None" "BrkFace" "None" ...
##  $ MasVnrArea   : num  196 0 162 0 350 0 186 240 0 0 ...
##  $ ExterQual    : chr  "Gd" "TA" "Gd" "TA" ...
##  $ ExterCond    : chr  "TA" "TA" "TA" "TA" ...
##  $ Foundation   : chr  "PConc" "CBlock" "PConc" "BrkTil" ...
##  $ BsmtQual     : chr  "Gd" "Gd" "Gd" "TA" ...
##  $ BsmtCond     : chr  "TA" "TA" "TA" "Gd" ...
##  $ BsmtExposure : chr  "No" "Gd" "Mn" "No" ...
##  $ BsmtFinType1 : chr  "GLQ" "ALQ" "GLQ" "ALQ" ...
##  $ BsmtFinSF1   : int  706 978 486 216 655 732 1369 859 0 851 ...
##  $ BsmtFinType2 : chr  "Unf" "Unf" "Unf" "Unf" ...
##  $ BsmtFinSF2   : int  0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF    : int  150 284 434 540 490 64 317 216 952 140 ...
##  $ TotalBsmtSF  : int  856 1262 920 756 1145 796 1686 1107 952 991 ...
##  $ Heating      : chr  "GasA" "GasA" "GasA" "GasA" ...
##  $ HeatingQC    : chr  "Ex" "Ex" "Ex" "Gd" ...
##  $ CentralAir   : chr  "Y" "Y" "Y" "Y" ...
##  $ Electrical   : chr  "SBrkr" "SBrkr" "SBrkr" "SBrkr" ...
##  $ X1stFlrSF    : int  856 1262 920 961 1145 796 1694 1107 1022 1077 ...
##  $ X2ndFlrSF    : int  854 0 866 756 1053 566 0 983 752 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  1710 1262 1786 1717 2198 1362 1694 2090 1774 1077 ...
##  $ BsmtFullBath : int  1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  2 2 2 1 2 1 2 2 2 1 ...
##  $ HalfBath     : int  1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr : int  3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 2 2 ...
##  $ KitchenQual  : chr  "Gd" "TA" "Gd" "Gd" ...
##  $ TotRmsAbvGrd : int  8 6 6 7 9 5 7 7 8 5 ...
##  $ Functional   : chr  "Typ" "Typ" "Typ" "Typ" ...
##  $ Fireplaces   : int  0 1 1 1 1 0 1 2 2 2 ...
##  $ FireplaceQu  : chr  "No" "TA" "TA" "Gd" ...
##  $ GarageType   : chr  "Attchd" "Attchd" "Attchd" "Detchd" ...
##  $ GarageYrBlt  : chr  "2003" "1976" "2001" "1998" ...
##  $ GarageFinish : chr  "RFn" "RFn" "RFn" "Unf" ...
##  $ GarageCars   : int  2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea   : int  548 460 608 642 836 480 636 484 468 205 ...
##  $ GarageQual   : chr  "TA" "TA" "TA" "TA" ...
##  $ GarageCond   : chr  "TA" "TA" "TA" "TA" ...
##  $ PavedDrive   : chr  "Y" "Y" "Y" "Y" ...
##  $ WoodDeckSF   : int  0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF  : int  61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch: int  0 0 0 272 0 0 0 228 205 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : chr  "No" "No" "No" "No" ...
##  $ Fence        : chr  "No" "No" "No" "No" ...
##  $ MiscFeature  : chr  "No" "No" "No" "No" ...
##  $ MiscVal      : int  0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold       : chr  "2" "5" "9" "2" ...
##  $ YrSold       : chr  "2008" "2007" "2008" "2006" ...
##  $ SaleType     : chr  "WD" "WD" "WD" "WD" ...
##  $ SaleCondition: chr  "Normal" "Normal" "Normal" "Abnorml" ...
##  $ SalePrice    : int  208500 181500 223500 140000 250000 143000 307000 200000 129900 118000 ...
```

Part I.B EDA

```
dim(train)
```

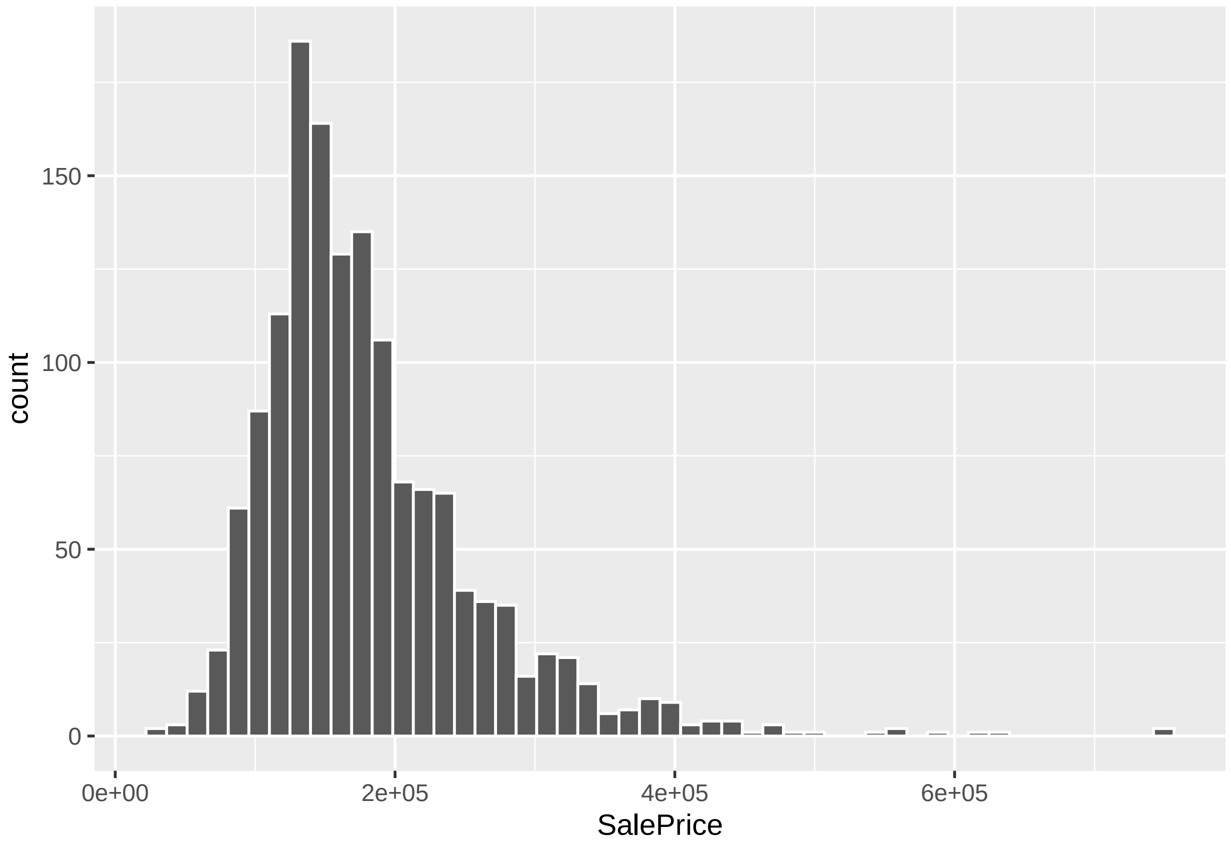
```
## [1] 1460   81
```

There are 1460 observations and 81 variables in the training data set. First, lets's start to explore the response - Sale Price (in dollars).

Part I.B EDA - (1) Response: SalePrice

```
ggplot(train, aes(x = SalePrice)) +
  geom_histogram(bins = 50, col= "white")
```

Part I.B EDA - (1.1) Histogram of SalePrice



The plot is right-skewed, which means that there is less expensive house than inexpensive ones.

```
summary(train$SalePrice)
```

Part I.B EDA - (1.2) Histogram of SalePrice

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   34900  129975  163000  180921  214000  755000
```

The median of Sale Price of the houses is $163000.
The mean of Sale Price of the houses is $180921.
The least expensive house is $34900.
The most expensive house is $755000.

Part I.B EDA - (2) Correlation

Part I.B EDA - (2.1) Correlation between numeric predictor variables Next, to investigate if there are early signs of variables are likely to be significant in predicting response. First, let's look at those numeric variables.

```
num_data <- subset(train, select = c(LotFrontage, LotArea, MasVnrArea, BsmtFinSF1,
                                     BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, X1stFlrSF,
                                     X2ndFlrSF, LowQualFinSF, GrLivArea, BsmtFullBath,
                                     BsmtHalfBath, FullBath, HalfBath, BedroomAbvGr,
                                     KitchenAbvGr, TotRmsAbvGrd, Fireplaces, GarageCars,
                                     GarageArea, WoodDeckSF, OpenPorchSF, EnclosedPorch,
                                     X3SsnPorch, ScreenPorch, PoolArea, MiscVal))

# Correlation between numerical data
cor_matrix = cor(num_data)

# We set 0.6 as the threshold for strong correlation
strong_cor = cor_matrix

# we fill all the absolute correlation value less than 0.6 as NA and the value on the diagonal as NA
strong_cor[abs(strong_cor) < 0.6] = NA
strong_cor[upper.tri(strong_cor, diag = TRUE)] = NA

# Then we find the variables with high correlation (>=0.6)
index <- which(strong_cor >= 0.6 | strong_cor <= -0.6, arr.ind = T)
strong_cor_var = cbind.data.frame(var1 = rownames(strong_cor)[index[,1]], # get the row name
                                  var2 = colnames(strong_cor)[index[,2]]) # get the column name
strong_cor_var
```

```
##            var1         var2
## 1 BsmtFullBath   BsmtFinSF1
## 2    X1stFlrSF  TotalBsmtSF
## 3    GrLivArea    X2ndFlrSF
## 4     HalfBath    X2ndFlrSF
## 5 TotRmsAbvGrd    X2ndFlrSF
## 6     FullBath    GrLivArea
## 7 TotRmsAbvGrd    GrLivArea
## 8 TotRmsAbvGrd BedroomAbvGr
## 9   GarageArea   GarageCars
```

From the correlation table above, we can clearly find that [BsmtFinSF1, BsmtFullBath], [TotalBsmtSF, X1stFlrSF], [GrLivArea, X2ndFlrSF], [HalfBath, X2ndFlrSF],[TotRmsAbvGrd, X2ndFlrSF],[GrLivArea, TotRmsAbvGrd], [GrLivArea, FullBath], [BedroomAbvGr, TotRmsAbvGrd], [GarageArea, GarageCars] have appear to be potentially problematic collinearity amongst the predictor variables.

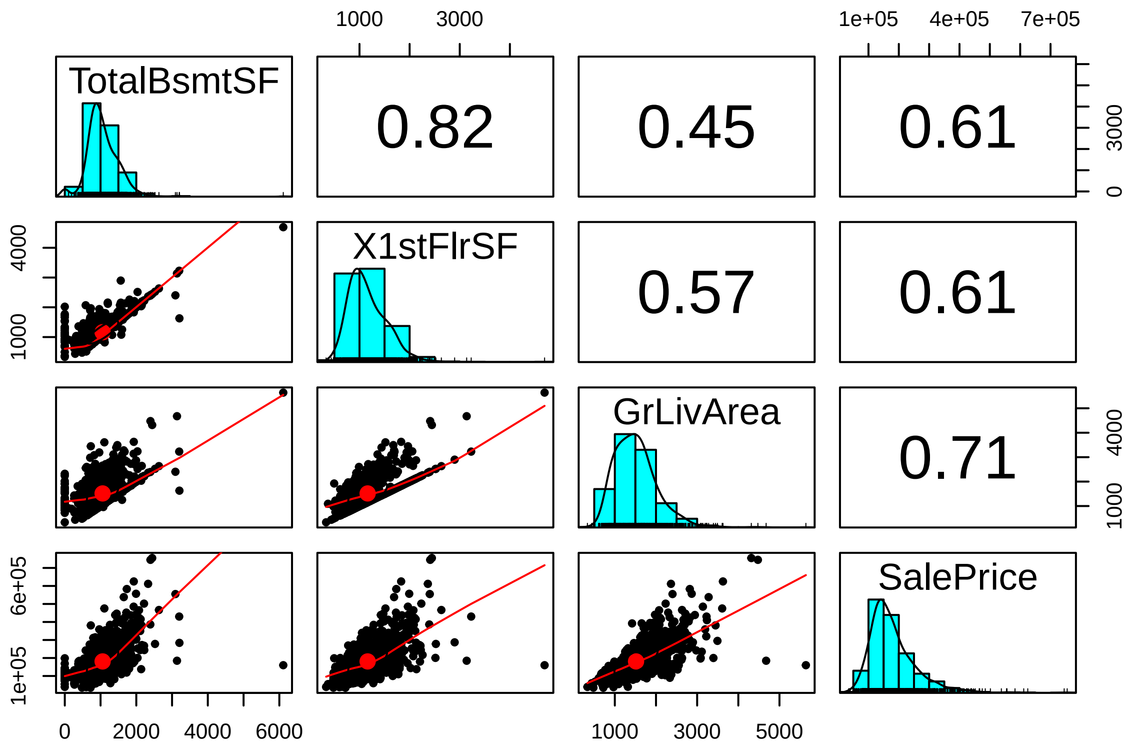
```
corr_xy = cor(train[,unlist(lapply(train, is.numeric))])
y_col = ncol(cor(train[,unlist(lapply(train, is.numeric))]))
corr_xy_df = cbind.data.frame(SalePrice=cor(train[,unlist(lapply(train, is.numeric))])[,y_col])
corr_xy_df
```

Part I.B EDA - (2.2) Correlation between numeric predictor variables and response variable

```
##                  SalePrice
## LotFrontage    0.33477085
## LotArea        0.26384335
## MasVnrArea     0.47261450
## BsmtFinSF1     0.38641981
## BsmtFinSF2    -0.01137812
## BsmtUnfSF      0.21447911
## TotalBsmtSF    0.61358055
## X1stFlrSF      0.60585218
## X2ndFlrSF      0.31933380
## LowQualFinSF  -0.02560613
## GrLivArea      0.70862448
## BsmtFullBath   0.22712223
## BsmtHalfBath  -0.01684415
## FullBath       0.56066376
## HalfBath       0.28410768
## BedroomAbvGr   0.16821315
## KitchenAbvGr  -0.13590737
## TotRmsAbvGrd   0.53372316
## Fireplaces     0.46692884
## GarageCars     0.64040920
## GarageArea     0.62343144
## WoodDeckSF     0.32441344
## OpenPorchSF    0.31585623
## EnclosedPorch -0.12857796
## X3SsnPorch     0.04458367
## ScreenPorch    0.11144657
## PoolArea       0.09240355
## MiscVal       -0.02118958
## SalePrice      1.00000000
```

From the last part, we find that the variables $TotalBsmtSF$, $X1stFlrSF$, $GrLivArea$, $GarageCars$, $GarageArea$ have strong correlation with the response $SalePrice$.

```
strong_collearity <- subset(train,
                            select = c(TotalBsmtSF, X1stFlrSF, GrLivArea, SalePrice))
pairs.panels(strong_collearity)
```



From the plot, we can verify that all of these three variables
(1) *TotalBsmtSF* (Total square feet of basement area)
(2) *X1stFlrSF* (First Floor square feet)
(3) *GrLivArea* (Above grade (ground) living area square feet)
have strong positive correlation with the response *SalePrice*.
In other words, as each of these three factors increases, *SalePrice* will get increase.

In addition, we would also explore those categorical variables that might be useful for predicting the response.

Part I.B EDA - (2.3) Correlation between categorical predictor variables and response variable

Part I.B EDA - (2.3) Categorical Predictor 1: OverallQual
OverallQual:

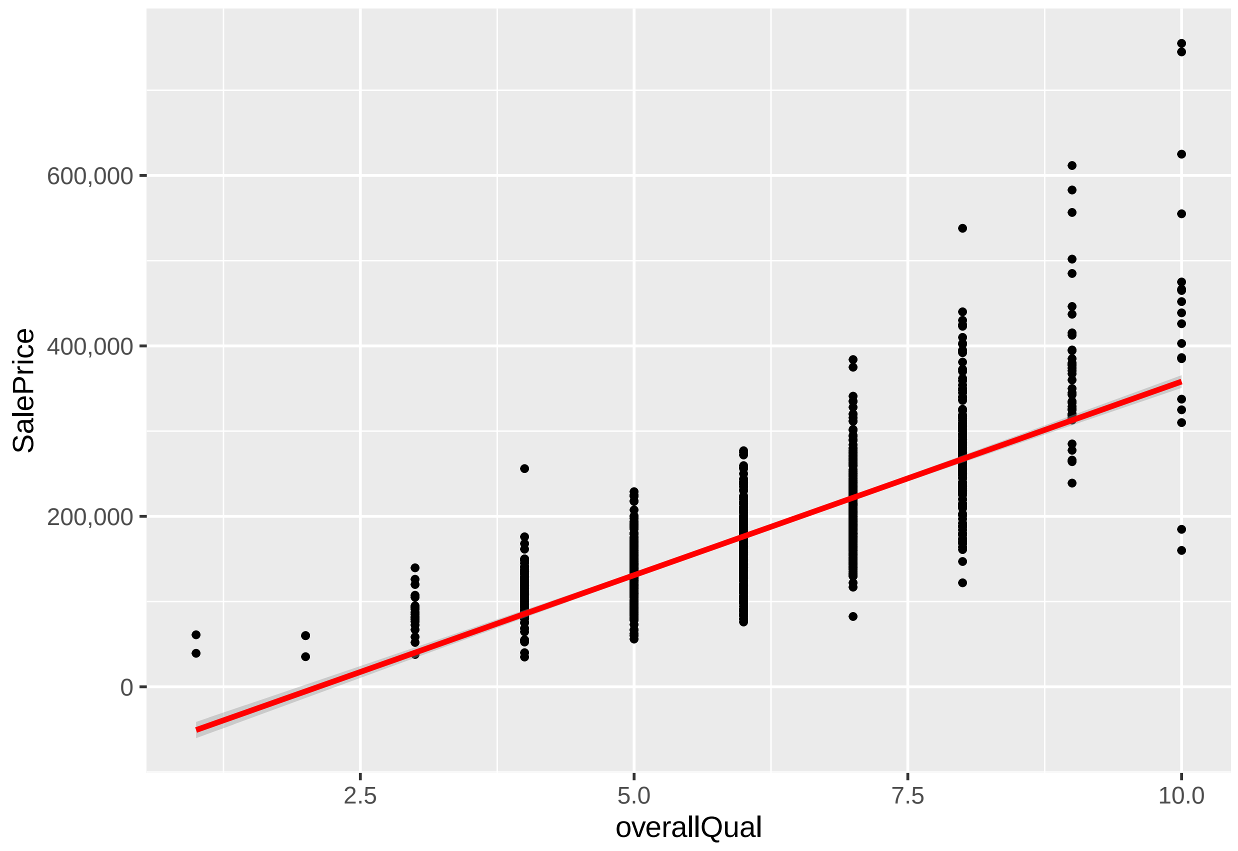
```
overallQual <- as.numeric(train$OverallQual)
cor(overallQual, train$SalePrice)
```

```
## [1] 0.7909816
```

Thus, *OverallQual* has a strong positive correlation with the *SalePrice*.

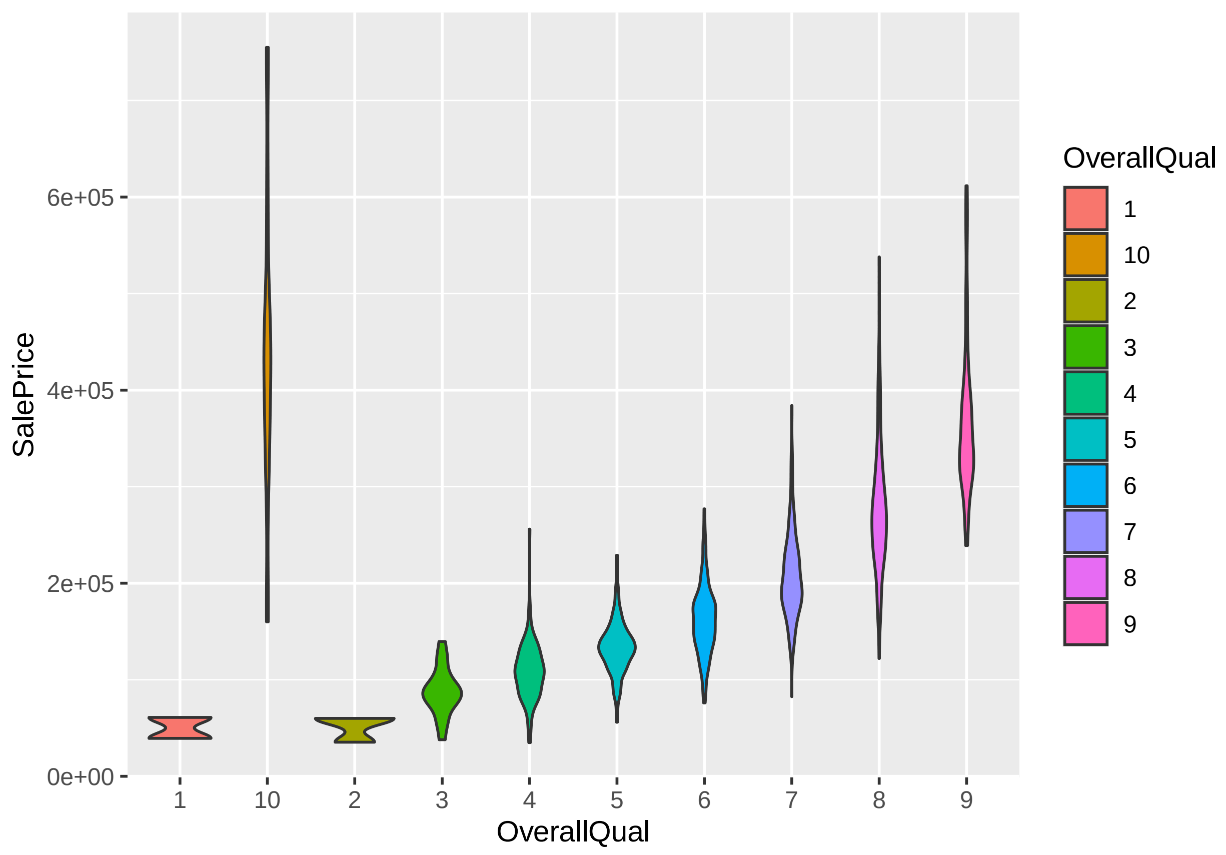
```
ggplot(train, aes(x=overallQual,y=SalePrice)) +
  geom_point(shape=20) +
  geom_smooth(method="lm", color = "red") +
  scale_y_continuous(labels=comma)
```

```
## 'geom_smooth()' using formula 'y ~ x'
```



This plot verifies that *OverallQual* has a strong positive correlation with the *SalePrice*.

```
train$OverallQual=as.factor(train$OverallQual)
train%>%ggplot(aes(x=OverallQual,y=SalePrice))+geom_violin(aes(fill=OverallQual))
```



OverallQual: Rates the overall material and finish of the house.

```
10    Very Excellent
9     Excellent
8     Very Good
7     Good
6     Above Average
5     Average
4     Below Average
3     Fair
2     Poor
1     Very Poor
```

From the plot above, we can find that as Rates the overall material and finish of the house increases, the sale price of the house gets increased. Besides, if the rates the overall material and finish of the house is below average, the sale price varies for the largest range other than that of other rates. In addition, there is no big difference of the mean sale price of house at rate = 6 and rate = 7.

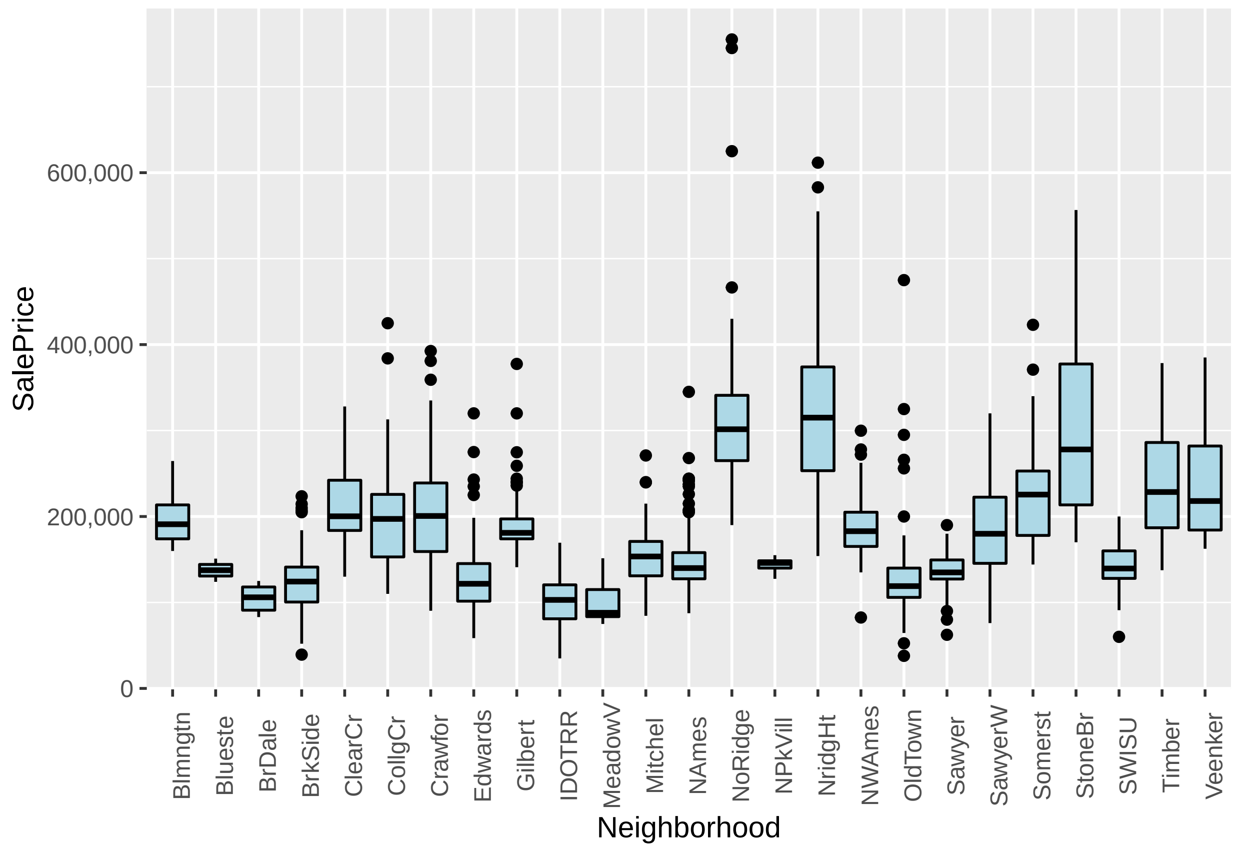
Part I.B EDA - (2.3) Categorical Predictor 2: Neighboorhood

Also, we can find that $Neighboorhood$ is also a good predictor for its positive strong correlation with $SalePrice$.

```
cor(train$TotalBsmtSF , train$SalePrice)
```

```
## [1] 0.6135806
```

```
ggplot(train, aes(x=Neighborhood,y=SalePrice)) +
  geom_boxplot(fill="light blue", color="black")+
  theme(axis.text.x=element_text(angle = 90)) +
  scale_y_continuous(labels=comma)
```



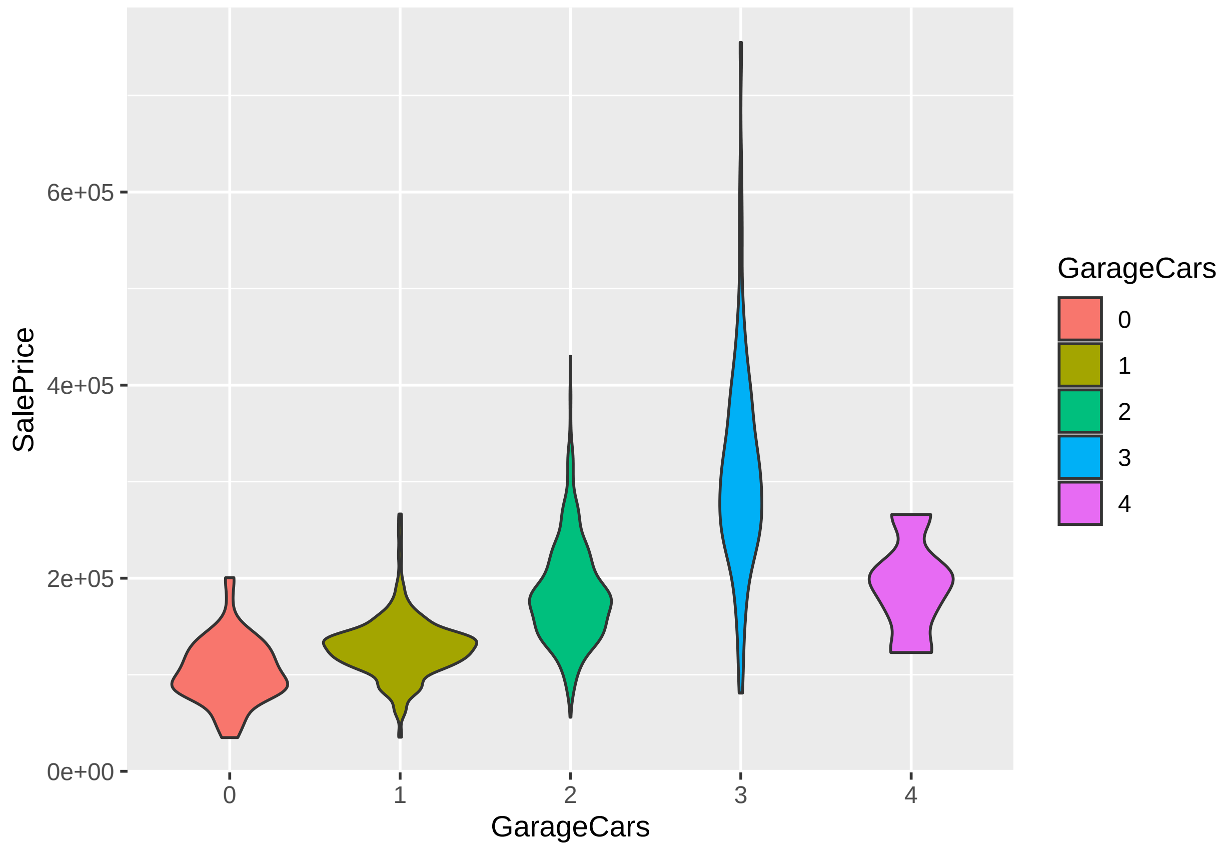
```
# change the scale from e-x type into real number
```

The neighborhoods of the house plays a significant role in the sale price. We can see that the houses around MeadowV were sold at the least expensive price, while those besides StroneBr were sold at the most expensive price. For houses' neighborhood is NoRidge, there are some of the most expensive price.

Part I.B EDA - (2.3) Categorical Predictor 2: GarageCars

Moreover, for the *GarageCars*, we would like to explore by plotting.

```
train$GarageCars=as.factor(train$GarageCars)
train%>%ggplot(aes(x=GarageCars,y=SalePrice))+geom_violin(aes(fill=GarageCars))
```



We can see that for the house of size of garage in car capacity as 3, the sale price is the highest, while the houses with size of garage in car capacity as 0 have the most inexpensive sale price.

At last, we infer that the age of house might be an important predictor.

Part I.B EDA - (2.3) Categorical Predictor 4: YearBuilt
House's age since being built (2022-YearBuilt):

```
year <- as.numeric(train$YearBuilt)
AgeHouse <- 2022 - year
```

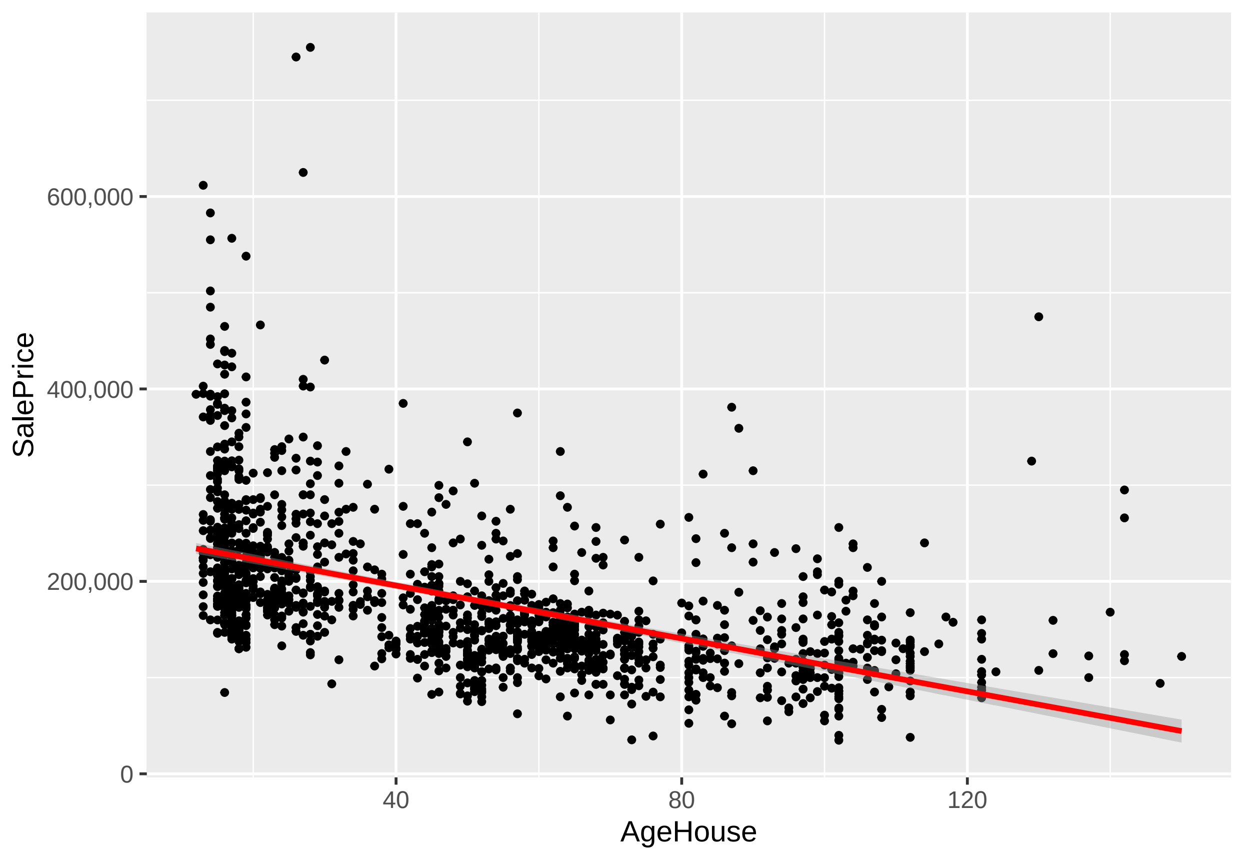
```
cor(AgeHouse, train$SalePrice)
```

```
## [1] -0.5228973
```

It shows that there is strong negative correlation between the age of house and the sale price.

```
ggplot(train, aes(x=AgeHouse,y=SalePrice)) + geom_point(shape=20) + geom_smooth(method="lm", color = "r
```

```
## `geom_smooth()` using formula 'y ~ x'
```



As the age of house increases, the sale price will get decreasing.

In brief, based on the EDA, there are lots of variables having weak correlation with the response. Thus, we would like to choose the LASSO and ridge regression for modeling.

Part II: Model Analysis

Part II.A - Motivation

Housing market is an important sector of the economy. Having an accurate prediction of housing price is of interest to the general public and the economic forecast. Conventional models for predictions include regression, decision trees, naive bayes, recurrent neural networks, etc. A good model should be able to generalize well to the test data, meaning that it should aim to capture the global minimum (in a strong convex optimization problem) without over-fitting or under-fitting, this means we need to take into account the bias-variance trade-off.

Motivated by these insights,
(1) we propose using Lasso (L1 regularization) for this particular task. Fitting a lasso-based regression model should ensure enough model capacity while minimizing the chance of over-fitting through regularization. It also has advantage over neural network given our limited amount of data samples. To find the best modeling strategy, we also test Ridge and Linear Regerssion along with Lasso.
(2) Though the sample size here is larger than the number of variables, which means there is no high-dimensional problem, we find there are collinearity problems among the predictor variables through EDA. Thus, we choose to use PLS and PCA based model - PCR to overcome this problems.

Part II.B - Math

(1) Linear Regression Model:

$$Y_i = \beta_o + \sum_{k=1}^{p} \beta_k X_k + e_i$$

$$e_i \overset{iid}{\sim} N(0, \sigma^2)$$

### (2) LASSO:

$$\min_{\beta_0, \beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - x_i^T \beta \right)^2 \right\} subject\ to \sum_{j=1}^{p} |\beta_j| \leq t$$

where against $s = t(\lambda)$

(3) Ridge:

$$\min_{\beta_0, \beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - x_i^T \beta \right)^2 \right\} subject\ to \sum_{j=1}^{p} \beta_j^2 \leq t$$

where against $s = t(\lambda)$

(4) PCR:

Let $Z_1, Z_2, ..., Z_m$ represent M < p linear combinations of our original p predictors. That is

$$Z_m = \sum_{j=1}^{p} \phi_{jm} x_j$$

$$z_{im} = u_{mp} x_{ip}$$

for some constants $\phi_{1m}, \phi_{2m}, ..., \phi_{pm}, m = 1, ..., M$. We can then fit the linear regression model.

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \epsilon_i$$

i = 1, . . . , n using least square. Note that in the upper equation, the regression coefficients are given by $\theta_0, \theta_1, ..., \theta_M$

(5) PLS:

Set

$$U_{mp} = \hat{\alpha}_p$$

from the regression model

$$y_i = \alpha_0 + \alpha_p X_{ip}^{(m)} + \epsilon_i$$

and calculate

$$z_{im} = \sum_{p=1}^{P} U_{mp} x_{ip}$$

for each p = 1, . . . ,P

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \epsilon_i$$

i = 1, . . . , n using least square.

Part II.C - Prepocess before applying model

We find that the predictions in train and test data set are not the same. Thus, we choose to remove the mismatch to ensure the same distribution between the train and test data set.

```
x_train <-  model.matrix(SalePrice~ . -1 , data = train)
y_train <-  train$SalePrice
x_test <-  model.matrix(SalePrice~ . -1 , data = test)
y_test <-  test$SalePrice
print(dim(x_train))
```

```
## [1] 1460 2028
```

```
print(dim(x_test))
```

```
## [1] 1447  341
```

```
#x_train-x_test
missing_cols = c()
for (var in colnames(x_train)){
  if (!(var %in% colnames(x_test))){
    missing_cols <- c(missing_cols, var)
  }
}

#x_test-x_train
missing_cols_2 = c()
for (var in colnames(x_test)){
  if (!(var %in% colnames(x_train))){
    missing_cols_2 <- c(missing_cols_2, var)
  }
}
#we simply remove the mismatch to ensure the same distribution between train and test dataset
x_train <- x_train[, !colnames(x_train) %in% missing_cols]
x_test <- x_test[, !colnames(x_test) %in% missing_cols_2]

train = train[, !colnames(train) %in% missing_cols]
train = train[, !colnames(train) %in% missing_cols_2]

test = test[, !colnames(test) %in% missing_cols]
test = train[, !colnames(train) %in% missing_cols]
```

Part II.D - Assumptions

Since Ridge Regression and Lasso Regression are special cases of the General Linear Model. They add penalty terms but otherwise all of the same conditions apply.
The normal linear regression model assumes:

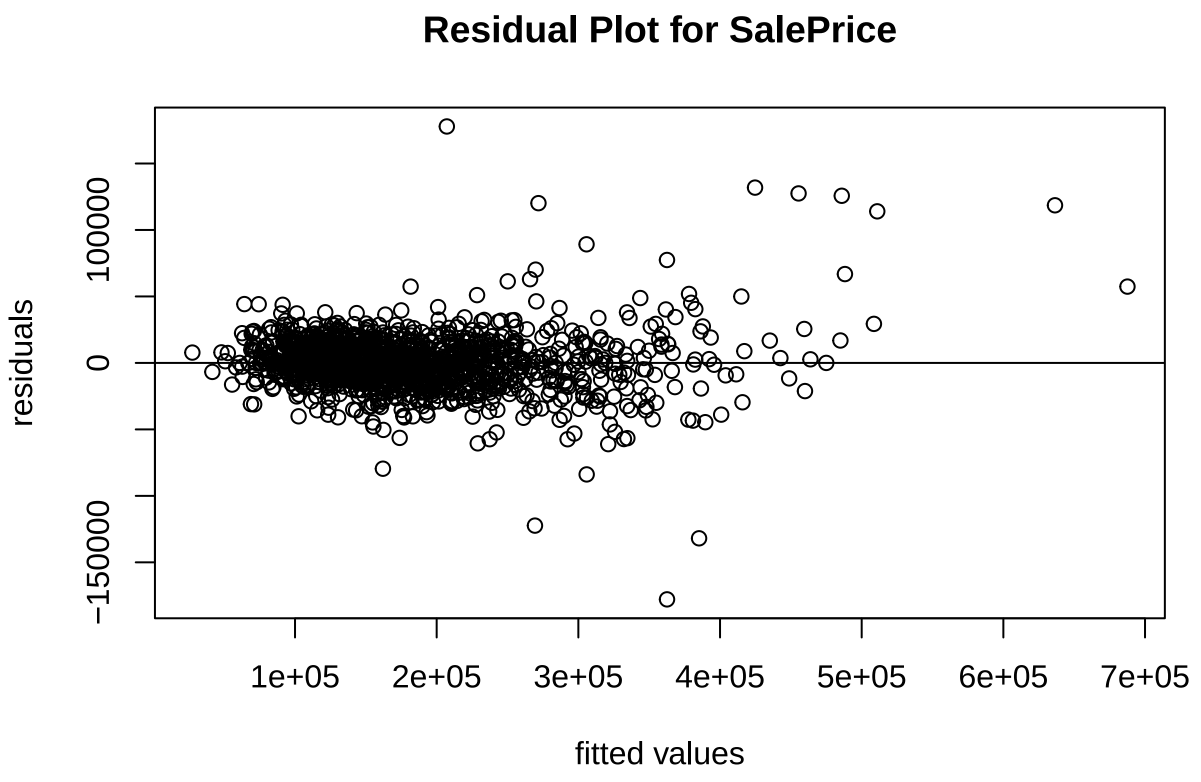
$$Y_i = \beta_o + \sum_{k=1}^{p} \beta_k X_k + e_i$$

$$e_i \overset{iid}{\sim} N(0, \sigma^2)$$

The final set of model assumptions for linear regression models are:
(1) Mean Function: $E(e_i|X) = 0$.
(2) Variance Function: $Var(e_i|X) = \sigma^2$.
(3) Normality of the errors
(4) Independence of the errors.
(5) Little/No Multicollinearity in data.

(4.1) Check the Mean Function: $E(e_i|X) = 0$.

```r
model.lm = lm(SalePrice ~., data = train)
plot(fitted(model.lm), resid(model.lm), xlab = "fitted values",
     ylab = "residuals", main = "Residual Plot for SalePrice")
abline(h = 0)
```

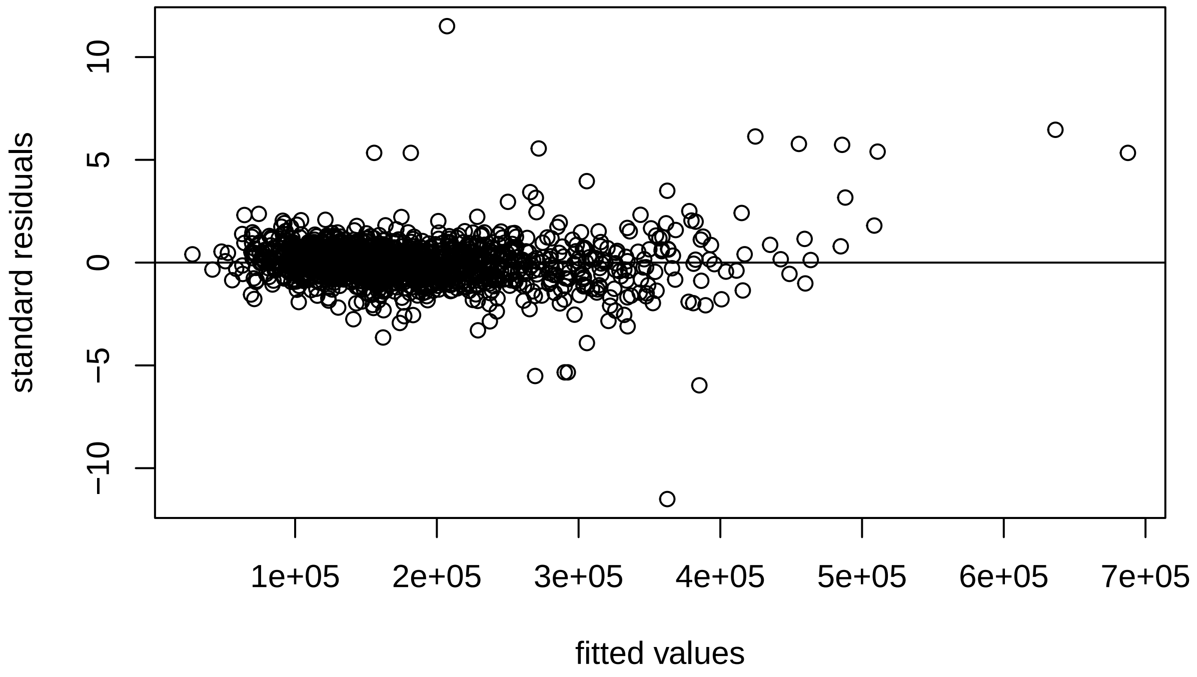


Based on the graph, it is noted that though there are few outliers, most dots are around 0, which means

that the mean of the error is approximately 0, which meet the assumption(1) of the model.
Thus, the fitted mean function is appropriate.

(4.2) Check Variance Function: $Var(e_i|X) = \sigma^2$.

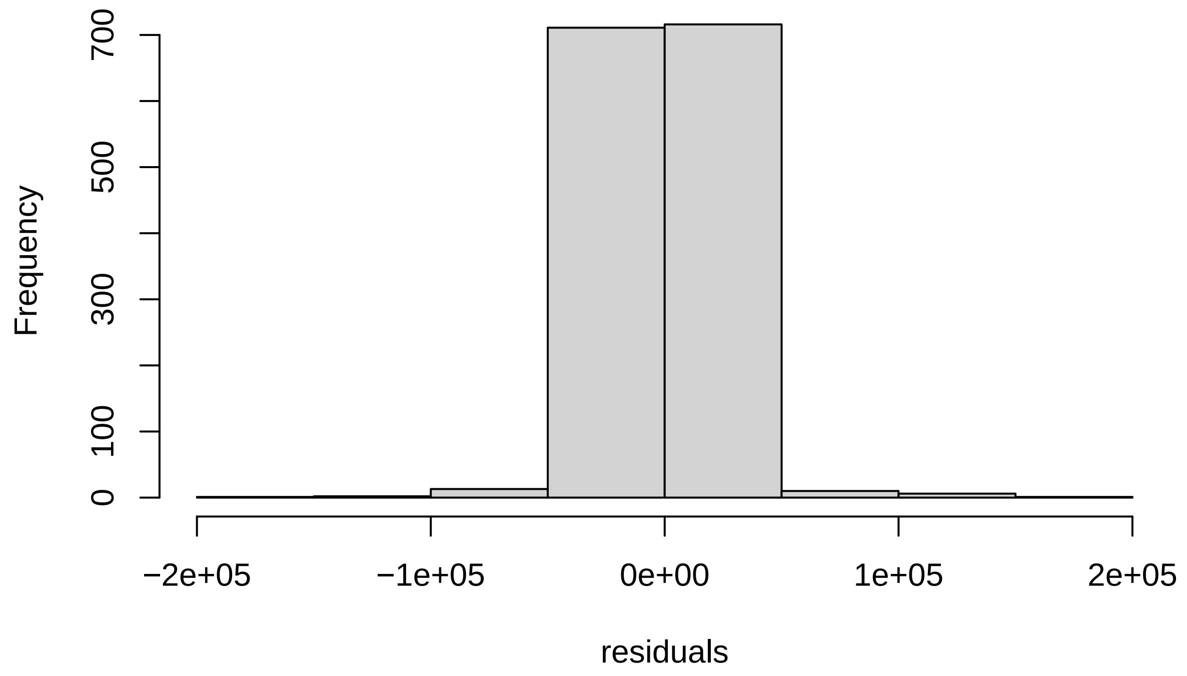
```
model.lm = lm(SalePrice ~., data = train)
plot(fitted(model.lm), rstandard(model.lm), xlab = "fitted values",
     ylab = "standard residuals", main = "")
abline(h = 0)
```



Based on the graph, it is noted that though there are few outliers, most dots are around 0 (a constant), which means that the variance of the error is approximately constant, which meet the assumption(2) of the model.
Thus, the fitted variance function is appropriate.

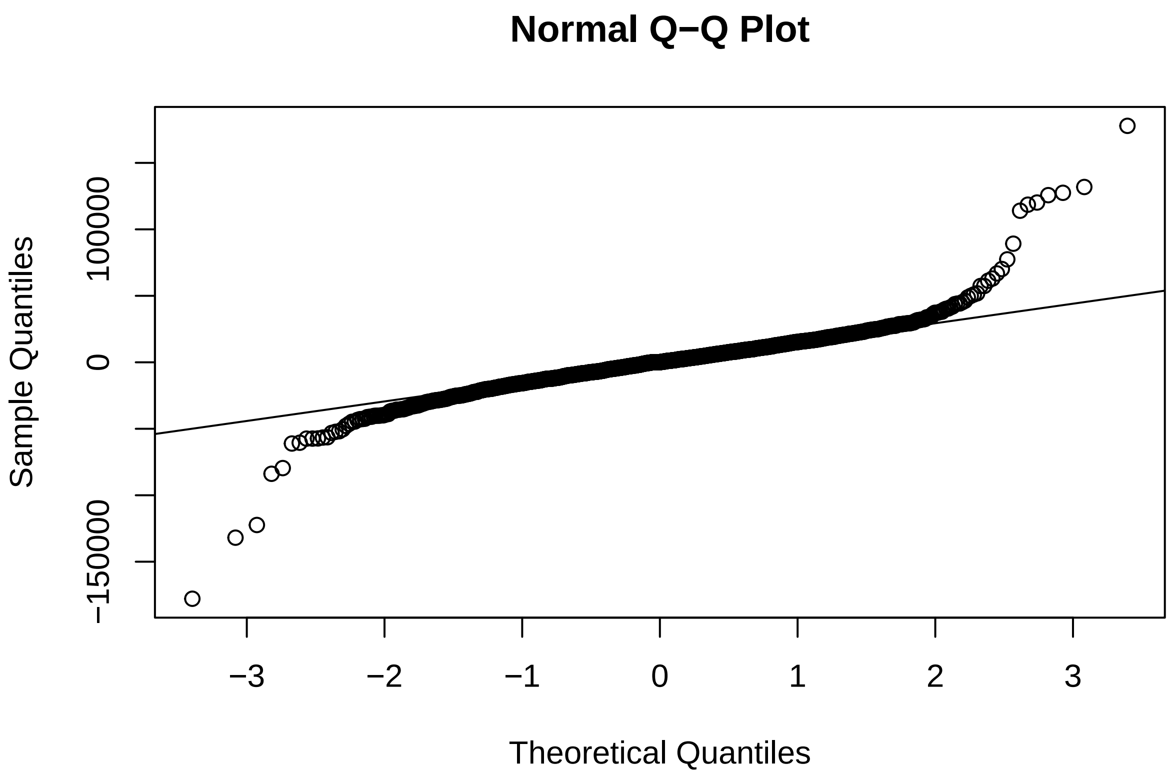
(4.3) Check Normality of the errors
Our two main graphical approaches will be: Histogram and Normal probability plot:

```
hist(resid(model.lm), xlab="residuals",main="")
```



Based on the histogram of residuals, it is noted that the graph is approximately symmetry and cut in half, each side is the mirror of the other.
Thus, the residuals are normally distributed, which means that the assumption(3) in the model is correct.

```
qqnorm(resid(model.lm)); qqline(resid(model.lm))
```



Based on the plot above, though we can see that points on the lower-end have lower measurement than the Normal model predicts and the points on the upper-end have higher measurement than the Normal model predicts, most points are approximately on the line. It might due to the outliers.
Thus, the residuals are normally distributed, which means that the assumption(3) in the model is correct.

(4.4) Check independence of the errors.
Common violation of independence in regression models are often related to structure in the mechanism that generated thye sample: error for data collected sequentially/spatially/clusters. The error for data collected spatially might cause the dependence of the error issue in this dataset, but it is hard to check with the available dataset.
Thus, we assume there is no such problem.

(4.5) Check the independence of the variables.
Based on the EDA in Part I, we can see that there are multicollineary among the variables.
Thus, it fails to meet the assumption(5).

Overall:
The final set of model assumptions are:
(1) Mean Function: $E(e_i|X) = 0$.
(2) Variance Function: $Var(e_i|X) = \sigma^2$.
(3) Normality of the errors
(4) Independence of the errors.
(5) Little/no Multicollinearity in data.
After checking, we see that the assumptions 1-3 are met. It is hard to check whether assumption 4 is met. Here, we assume the independence of the error.

Through the EDA in Part I, we see that there are collinearly issue in our dataset. In consideration of some useless variables exist in our data set, we still want to use Lasso to zero out them and see how many variables can help us to predict the price of the house.

Part II.E - Validation

Here we explain one important step to the data augmentation. For fairness purposes, we need to ensure that our train and test distributions are the same (e.g. predictors existing in test data must also exist in train data). For this reason, we first compute the confusion matrix of both the train and test data, we then drop the features that are in the disjoint set of train and test sets.

Part II.F - Models

Part II.F - Models - (1) Linear Regression Model

```r
set.seed(4620)
model.lm = lm(SalePrice~.,data = as.data.frame(train))
pred <- predict(model.lm, newdata = as.data.frame(test))
```

```
## Warning in predict.lm(model.lm, newdata = as.data.frame(test)): prediction from
## a rank-deficient fit may be misleading
```

```r
mse_lm = mean((pred-y_test)^2)
```

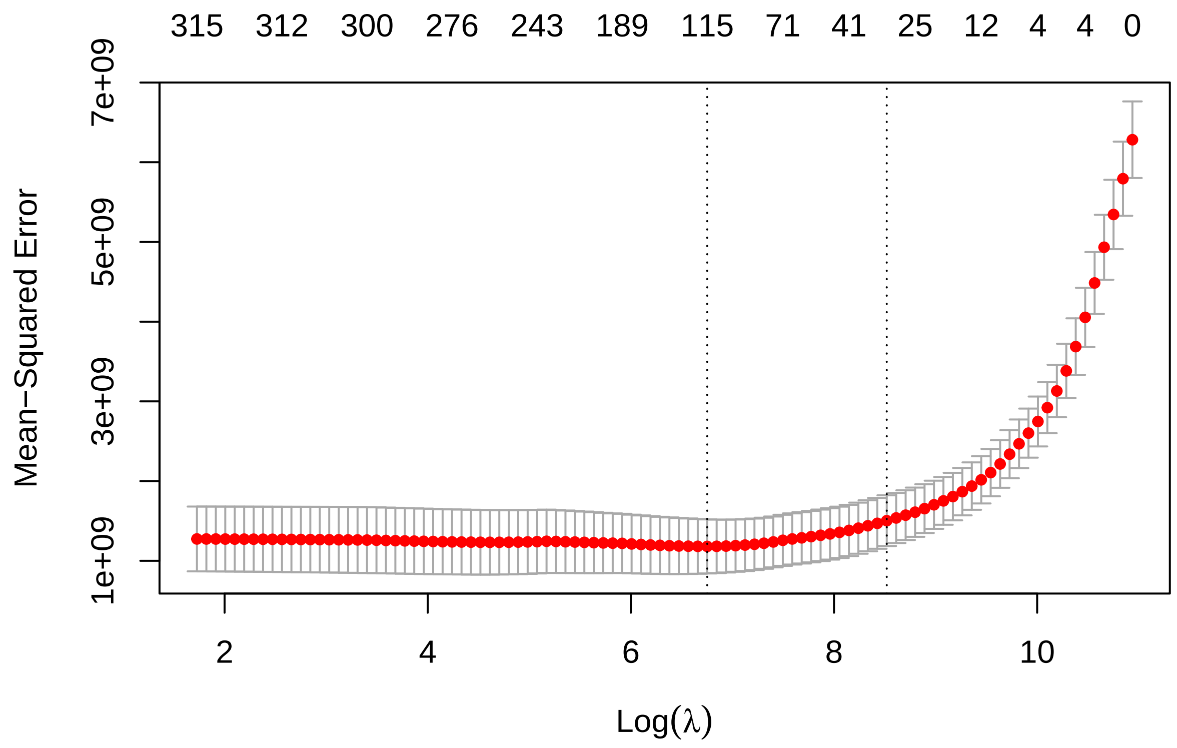
```
## Warning in pred - y_test: longer object length is not a multiple of shorter
## object length
```

```r
mse_lm
```

```
## [1] 12301384971
```

Part II.F - Models - (2) Lasso

```r
set.seed(4620)
cv.lasso <-  cv.glmnet(x_train, y_train, type.measure = "mse", alpha = 1)
plot(cv.lasso)
```

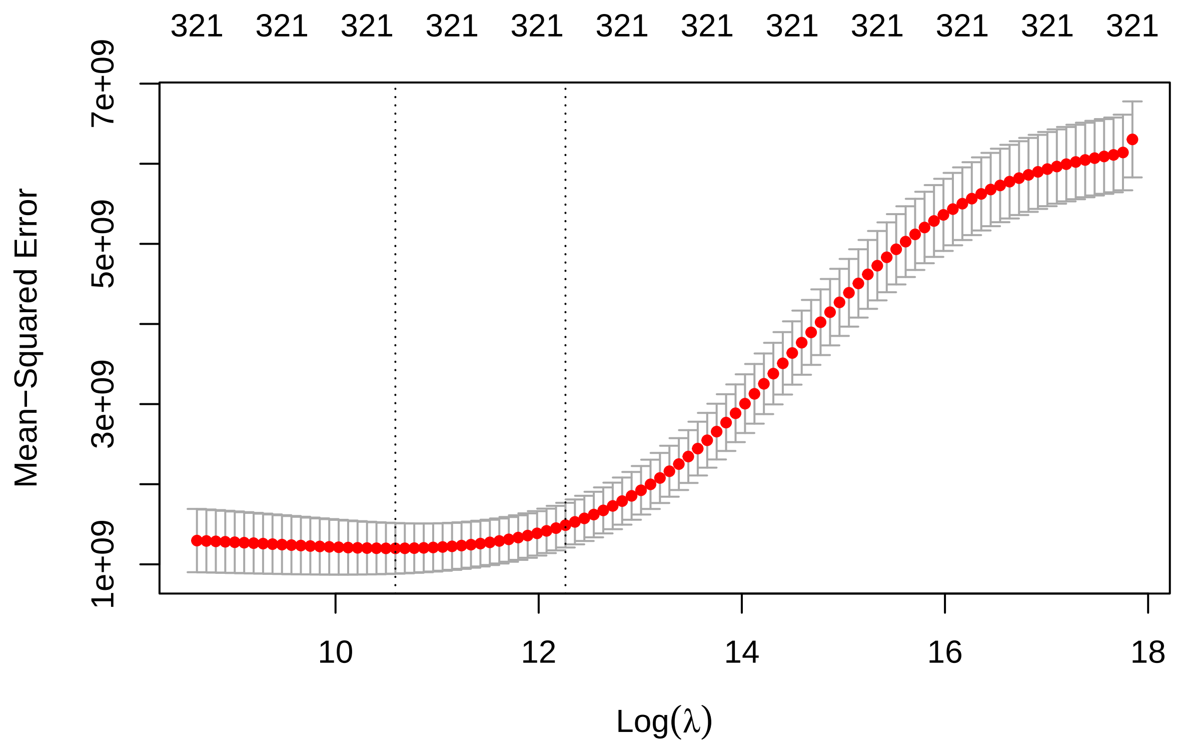


```
model.lasso =glmnet(x_train,y_train,lambda=cv.lasso$lambda.min, alpha=1)
pred <- predict(model.lasso,x_test)
mse_lasso = mean((pred-y_test)^2)
mse_lasso
```

```
## [1] 794699318
```

Part II.F - Models - (3) Ridge

```
set.seed(4620)
cv.ridge <-  cv.glmnet(x_train, y_train, type.measure = "mse", alpha = 0)
plot(cv.ridge)
```

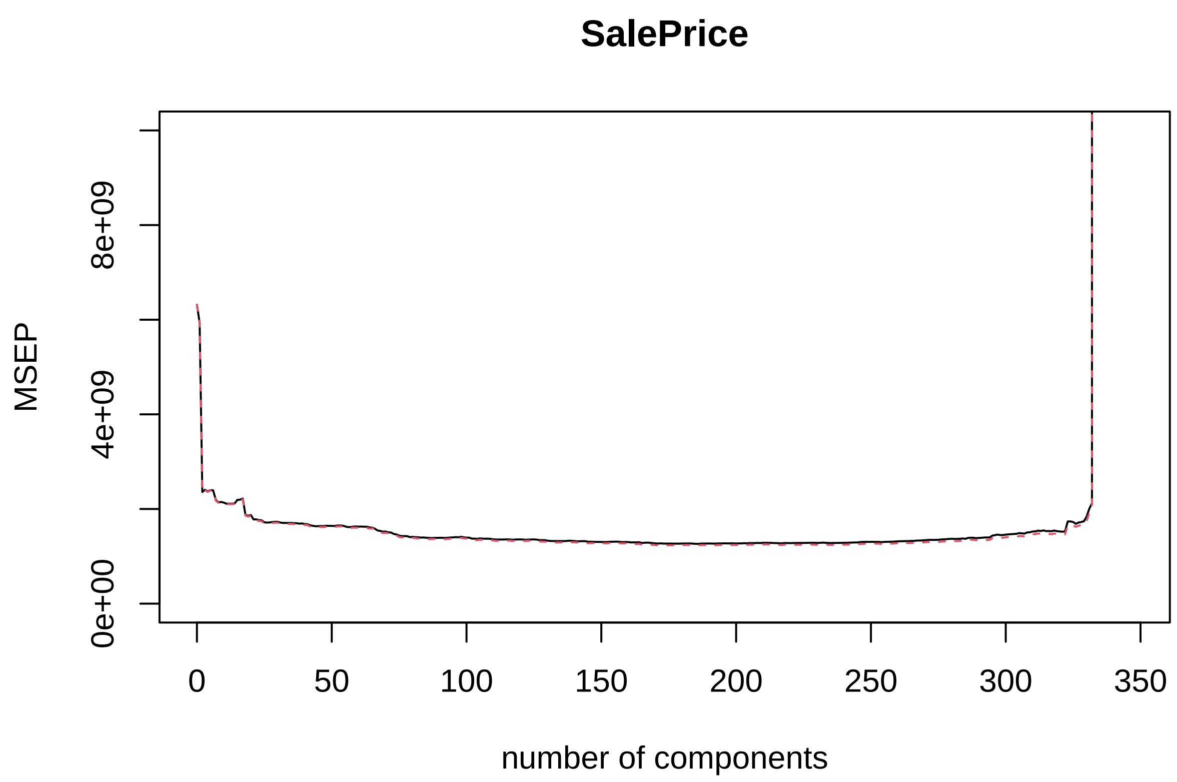


```
model.ridge = glmnet(x_train,y_train,lambda=cv.ridge$lambda.min, alpha=0)
pred <- predict(model.ridge,x_test)
mse_ridge = mean((pred-y_test)^2)
mse_ridge
```

```
## [1] 882528822
```

Part II.F - Models - (4) PCR

```
set.seed(4620)
model.pcr = pcr(SalePrice~.,data=train, validation="CV")
# summary(model.pcr)
validationplot(model.pcr,val.type="MSEP", ylim=c(0,9999999999))
```



```r
model.pcr2 = pcr(SalePrice~.,data=train,scale=TRUE,ncomp=170)
pcr.pred=predict(model.pcr2,ncomp=170)
mse_pcr = mean((as.vector(pcr.pred)-y_test)^2)
```

```
## Warning in as.vector(pcr.pred) - y_test: longer object length is not a multiple
## of shorter object length
```

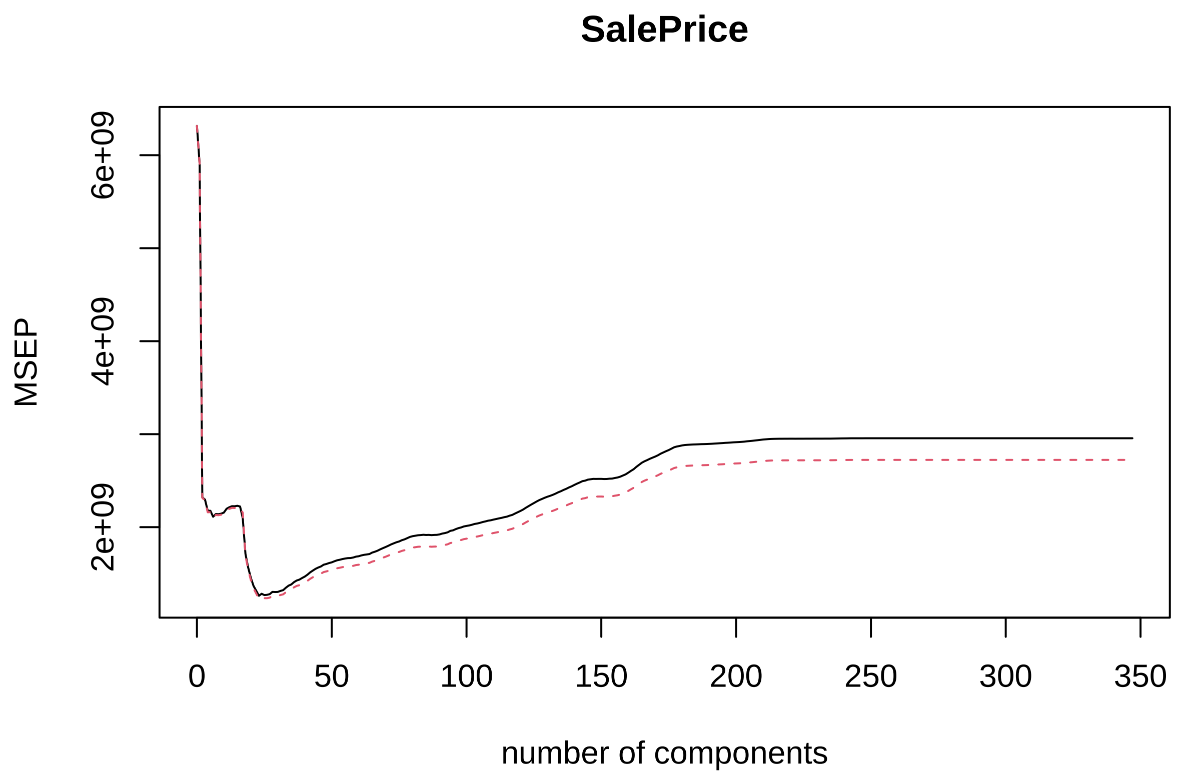
```r
mse_pcr
```

```
## [1] 11747770871
```

For the dataset, it looks like the smallest CV error occurs when we use 170 principal components in the regression for SalePrice This is fewer than the total number of predictors in the dataset (347), so it seems like the dimension-reduction in PCR gaining us much.

Part II.F - Models - (5) PLS

```r
set.seed(4620)
model.plsr = plsr(SalePrice~.,data=train,validation="CV")
# summary(model.plsr)
validationplot(model.plsr,val.type="MSEP")
```



```
model.plsr2 = plsr(SalePrice~.,data=train,scale=TRUE,ncomp=23)
pls.pred=predict(model.plsr2,ncomp=23)
mse_pls = mean((as.vector(pls.pred)-y_test)^2)
```

```
## Warning in as.vector(pls.pred) - y_test: longer object length is not a multiple
## of shorter object length
```

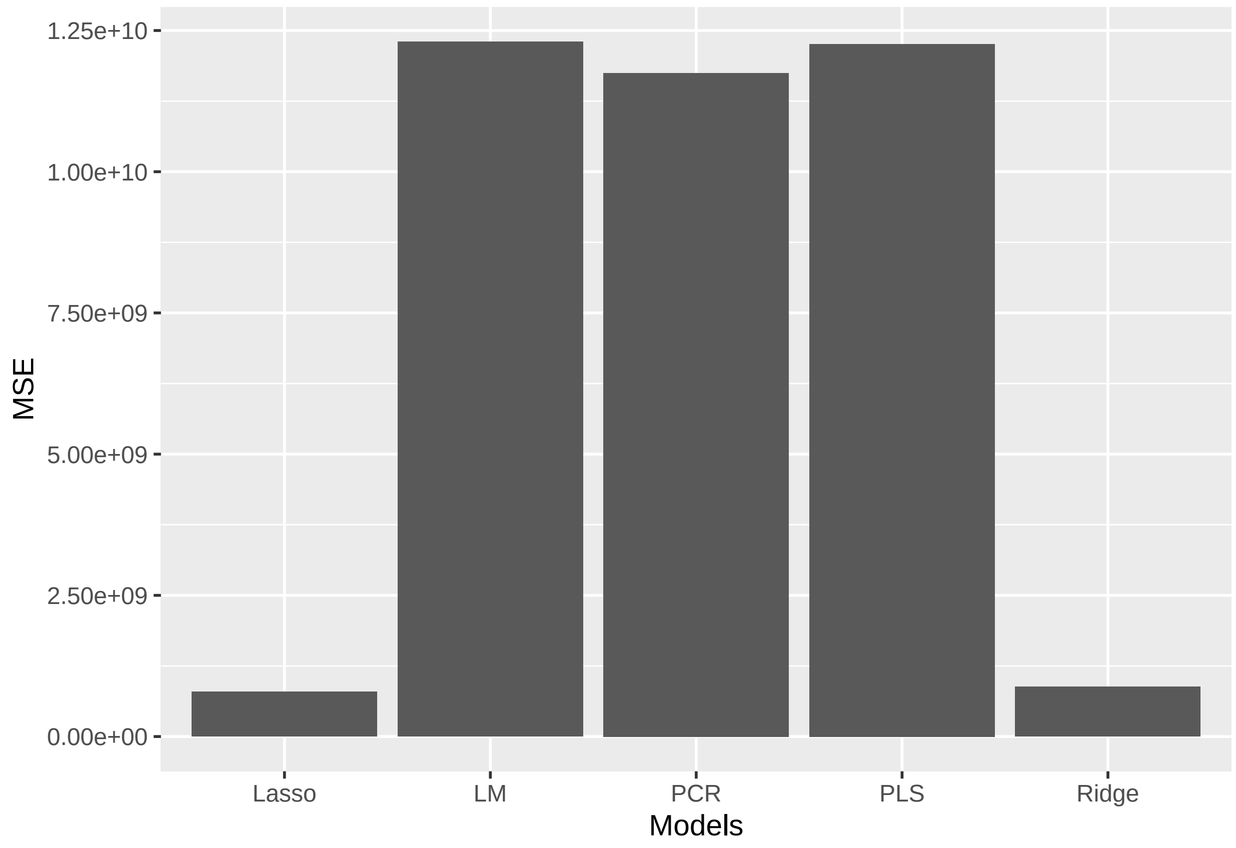
```
mse_pls
```

```
## [1] 12260383255
```

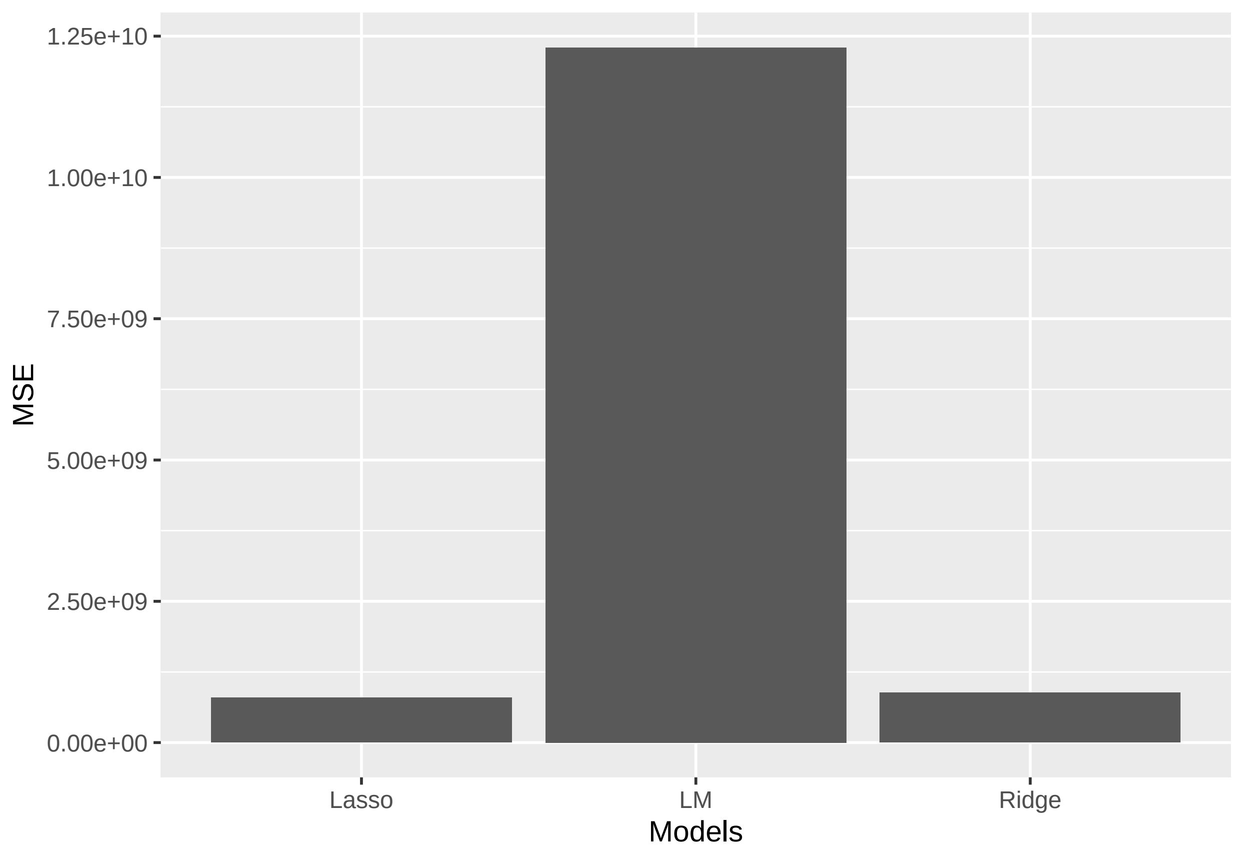
For the dataset, it looks like the smallest CV error occurs when we use 23 principal components in the regression for SalePrice This is fewer than the total number of predictors in the dataset (347), so it seems like the dimension-reduction in PLS gaining us much.

```
models = c("LM","Lasso","Ridge","PCR","PLS")
mse = c(mse_lm, mse_lasso, mse_ridge, mse_pcr, mse_pls)
compare = data.frame(mse, models)
ggplot(compare, aes(x=models,y=mse)) +
    geom_bar(stat="identity")+labs(x= "Models", y="MSE")
```

Part II.G - Comparison



```
models = c("LM","Lasso","Ridge")
mse = c(mse_lm, mse_lasso, mse_ridge)
compare2 = data.frame(mse, models)
ggplot(compare2, aes(x=models,y=mse))+geom_bar(stat="identity")+labs(x= "Models", y="MSE")
```



Part II.H - Results

Based on the MSE of 4 models, we find that
(1) Linear Regression Model has larger MSE than Lasso and Ridge and Lasso Regression.
(2) PLS and PCR have comparably similar MSE, while Lasso Regression and Ridge Regression have comparably similar MSE.
(3) Lasso and Ridge Regression's MSE much smaller than PLS and PCR's MSE.
(4) Lasso has the lowest MSE value.

To conclude, since Lasso and Ridge Regression's MSE are much smaller than Linear Regression, it means that there are many unnecessary variables in the dataset, which would have a negative influence on the prediction. Also, since Lasso and Ridge Regression's MSE much smaller than PLS and PCR's MSE, it means that the useless variables in the dataset have a larger negative influence on the models than the collinearity problems among the predictors. Thus, using Lasso regression model can help us to zero out the unnecessary factors and predict the SalePrice more accurately.