House Price Prediction Project

Qi JIANG

Environment Setup

```
# Loading packages
library(dplyr)
library(ggplot2)
library(reshape2)
library(tidyverse)
library(readr)
library(naniar)
library(visdat)
library(rcompanion)
library(superml)
library(forecast)
library(randomForest)

# Loading data
# Data source: https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques
train <- read.csv('train.csv', stringsAsFactors = F)
test <- read.csv('test.csv', stringsAsFactors = F)
cat(paste(c("The train data dimension before dropping 'Id' column is "), sep=""), paste("(", sep=""), paste(dim(train), coll
apse = ","), paste(").", sep=""))
```

```
## The train data dimension before dropping 'Id' column is  ( 1460,81 ).
```

```
cat(paste(c("\nThe test data dimension before dropping 'Id' column is "), sep=""), paste("(", sep=""), paste(dim(test), coll
apse = ","), paste(").", sep=""))
```

```
##
## The test data dimension before dropping 'Id' column is  ( 1459,80 ).
```

```
# Dropping 'Id' column
train <- train[-1]
test <- test[-1]

# Checking again
cat(paste(c("\nThe train data dimension after dropping 'Id' column is "), sep=""), paste("(", sep=""), paste(dim(train), col
lapse = ","), paste(").", sep=""))
```

```
##
## The train data dimension after dropping 'Id' column is  ( 1460,80 ).
```

```
cat(paste(c("\nThe test data dimension after dropping 'Id' column is "), sep=""), paste("(", sep=""), paste(dim(test), colla
pse = ","), paste(").", sep=""))
```

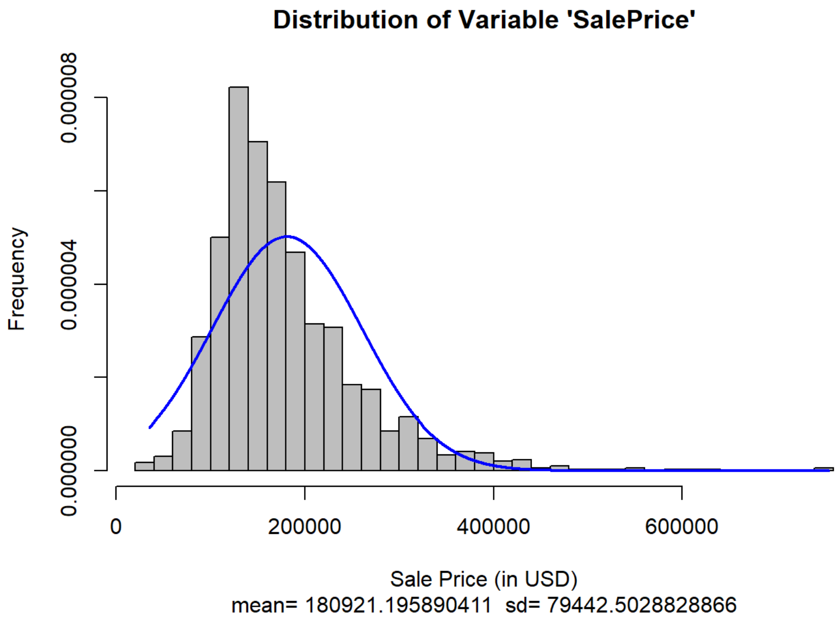
```
##
## The test data dimension after dropping 'Id' column is  ( 1459,79 ).
```

Data Processing

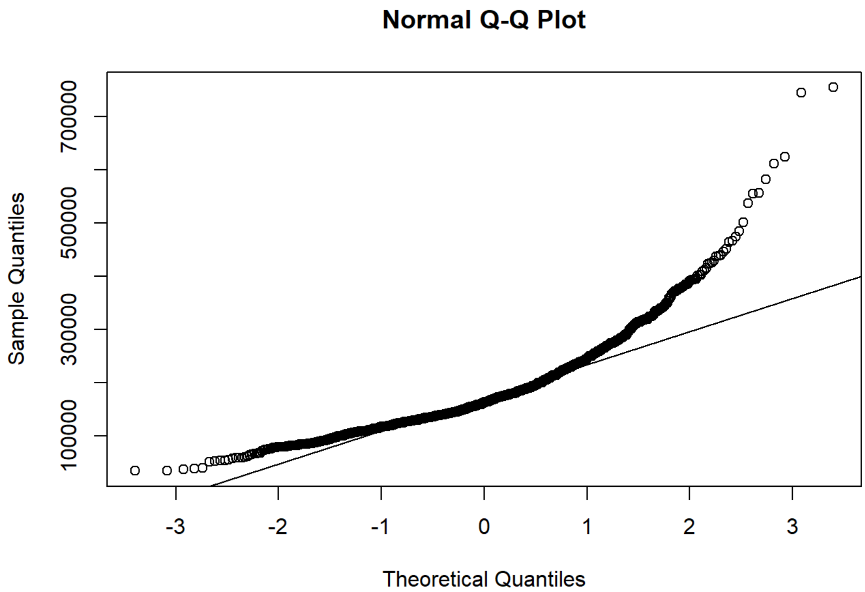
Target Variable: ‘SalePrice’

Let’s plot the distribution of *SalePrice*.

```
options(scipen = 999) # to avoid scientific notation on x-axis
plotNormalHistogram(train$SalePrice, prob=TRUE, breaks=30, main=c("Distribution of Variable 'SalePrice'"),sub=paste("mean=",
mean(train$SalePrice), "\t sd=", sd(train$SalePrice)), xlab=c("Sale Price (in USD)"), ylab=c("Frequency"))
```




```
qqnorm(train$SalePrice)
qqline(train$SalePrice)
```



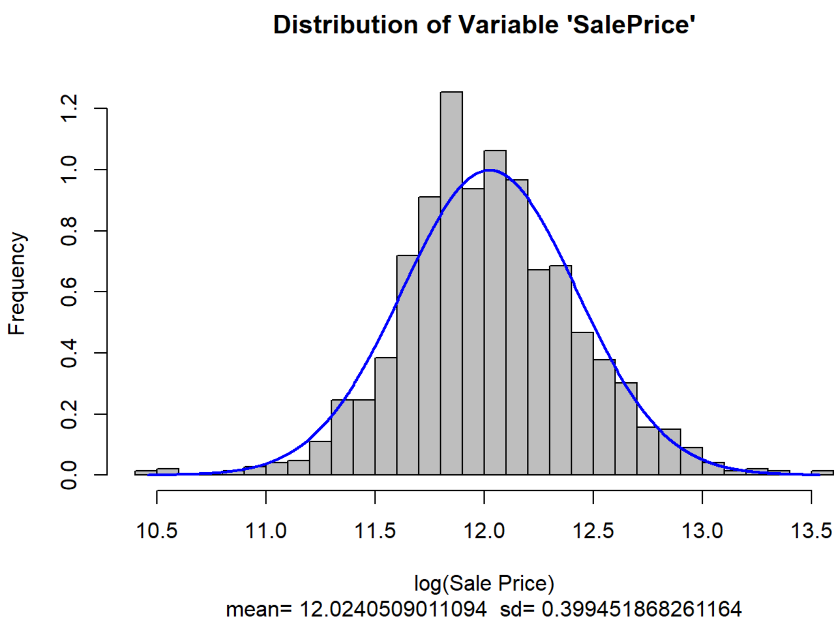
The target variable *SalePrice* is right skewed. In order to fit linear regression models, it is more appropriate to do a log-transformation on this variable.

Log-transformation on Target Variable

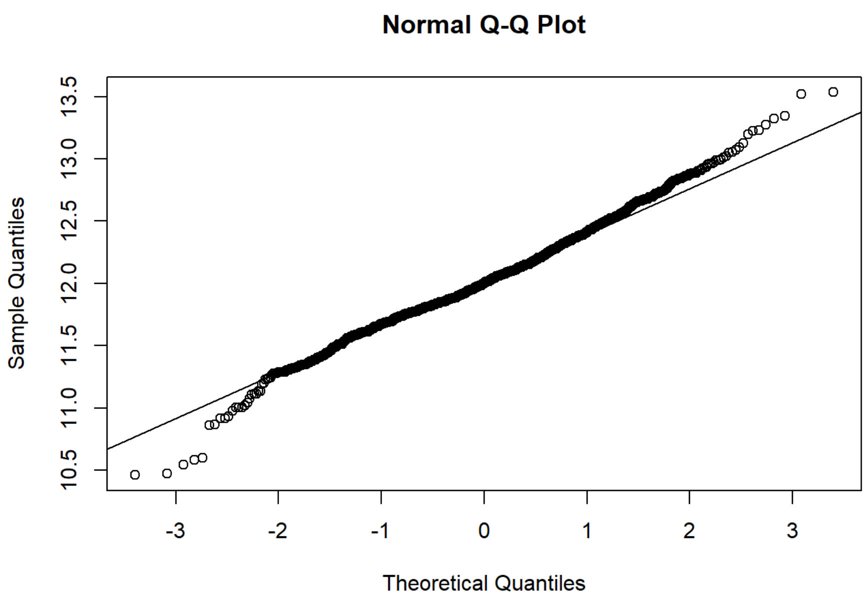
```
train <- mutate(train, logSalePrice = log(SalePrice))
```

Checking Again

```
options(scipen = 999) # to avoid scientific notation on x-axis
plotNormalHistogram(train$logSalePrice, prob=TRUE, breaks=30, main=c("Distribution of Variable 'SalePrice'"),sub=paste("mean
=", mean(train$logSalePrice), "\t sd=", sd(train$logSalePrice)), xlab=c("log(Sale Price)"), ylab=c("Frequency"))
```




```
qqnorm(train$logSalePrice)
qqline(train$logSalePrice)
```



Now, the skewness is corrected.

Feature Engineering

Note that we need to apply the same transformation on both train and test data.

```
# Combining train and test data
train <- mutate(train, UsedToTrain=TRUE) ## creating an indicator
test <- mutate(test, UsedToTrain=FALSE, SalePrice=0, logSalePrice=0)
full <- rbind(train, test)
cat(paste("The dimension of full dataset is "), paste(dim(full), collapse = ","))
```

```
## The dimension of full dataset is  2919,82
```

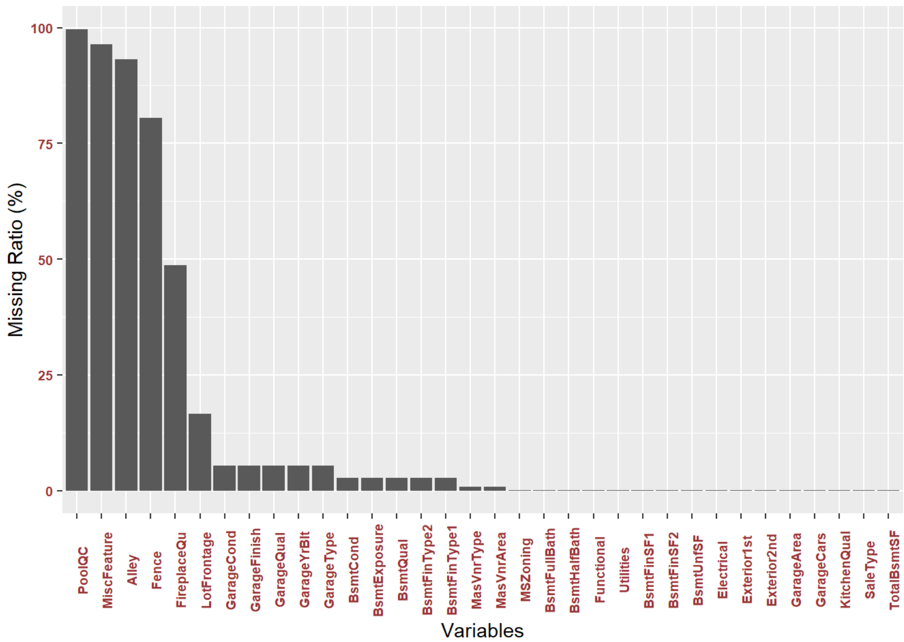
Missing Data

```
full.NAratio <- as.data.frame(sort(colMeans(is.na(full))*100, decreasing = TRUE))
colnames(full.NAratio) <- "MissingRatio"
full.NAratio <- filter(full.NAratio, MissingRatio>0)
full.NAratio
```

	MissingRatio <dbl>
PoolQC	99.65741692
MiscFeature	96.40287770
Alley	93.21685509
Fence	80.43850634
FireplaceQu	48.64679685
LotFrontage	16.64953751
GarageYrBlt	5.44707091
GarageFinish	5.44707091
GarageQual	5.44707091
GarageCond	5.44707091

1-10 of 34 rows Previous **1** 2 3 4 Next

```
ggplot(full.NAratio, aes(x=reorder(row.names(full.NAratio), -MissingRatio), y=MissingRatio)) + geom_bar(stat = "identity") + theme(axis.text.x = element_text(face="bold", color="#993333", size=7, angle=90),axis.text.y = element_text(face="bold", color="#993333", size=7, angle=0)) + ylab("Missing Ratio (%)") + xlab("Variables")
```



Imputation on Non-Random Missing Data

Many *NA* values above represents the absence of a facility, such as *PoolQC*. An *NA* value of *PoolQC* just means that there is no pool in this property. Same cases are variable *Alley,BsmtQual,BsmtCond,BsmtExposure,BsmtFinType1,FireplaceQu,GarageType,GarageFinish,GarageQual,GarageCond,PoolQC,Fence,MiscFeature,BsmtF*

So, we need to replace these “false” missing values with ‘None’.

```
for (col in c('Alley','BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','FireplaceQu','GarageType','GarageFinish','GarageQual','GarageCond','PoolQC','Fence','MiscFeature','BsmtFinType2', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'MasVnrType')){
  full[[col]] <- replace_na(full[[col]], 'None')
}
```

Imputation on ‘LotFrontage’

Since the lengths of lot frontage are very close within a neighborhood, we decide to replace ‘NA’ in *LotFrontage* of a data point by the median value in its neighborhood.

```
full <- full %>% group_by(Neighborhood) %>% mutate(LotFrontage=ifelse(is.na(LotFrontage), median(LotFrontage, na.rm=TRUE), LotFrontage))
```

Imputation on Numerical Variables

```
for (col in c('GarageYrBlt', 'GarageArea', 'GarageCars', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF','TotalBsmtSF', 'BsmtFullBath', 'BsmtHalfBath', 'MasVnrArea')){
  full[[col]] <- replace_na(full[[col]], 0)
}
```

Other Cases

- *Utilities*: the entire data set have value “AllPub”, except one in the train dataset.

```
table(select(train, Utilities))
```

```
##
## AllPub NoSeWa
##   1459      1
```

```
table(select(test, Utilities))
```

```
##
## AllPub
##   1457
```

So, this feature will not be effective in predictive models. We need to drop variable *Utilities*.

```
full <- full[,names(full)!='Utilities']
```

- *Functional*: based on data description, ‘NA’ means typical

```
full[['Functional']] <- replace_na(full[['Functional']], 'typical')
```

- *MSZoning*: there are 4 missing values

```
table(full$MSZoning)
```

```
##
## C (all)      FV      RH      RL      RM
##      25     139      26    2265     460
```

```
sum(is.na(full$MSZoning))
```

```
## [1] 4
```

We decide to replace them by the most common value ‘RL’.

```
full[['MSZoning']] <- replace_na(full[['MSZoning']], 'RL')
```

- *Electrical, KitchenQual, Exterior1st, Exterior2nd, SaleType*: each of them has only 1 missing value

```
apply(full[, c('Electrical', 'KitchenQual', 'Exterior1st', 'Exterior2nd', 'SaleType')], 2, function(x){sum(is.na(x))})
```

```
##  Electrical KitchenQual Exterior1st Exterior2nd    SaleType
##           1           1           1           1           1
```

So, we can just drop that data point.

```
full <- filter(full, !(is.na(Electrical)|is.na(KitchenQual)|is.na(Exterior1st)|is.na(Exterior2nd)|is.na(SaleType)))
```

Checking Again for Missing Values

```
full.NAratio <- as.data.frame(sort(colMeans(is.na(full))*100, decreasing = TRUE))
colnames(full.NAratio) <- "MissingRatio"
full.NAratio <- filter(full.NAratio, MissingRatio>0)
full.NAratio
```

```
0 rows
```

There is no missing value now!

Converting Some Numerical Variables That Are Actually Categorical

Variables *MSSubClass,OverallCond,YrSold,MoSold* were entered as numerical data. But they are actually categorical data.

```
for (col in c("MSSubClass", "OverallCond", "YrSold", "MoSold")){
  full[[col]] <- as.factor(full[[col]])
}
```

Encoding Some Categorical Variables That Are Ordinal

Variables *FireplaceQu, BsmtQual, BsmtCond, GarageQual, GarageCond, ExterQual, ExterCon,HeatingQC, PoolQC, KitchenQual, BsmtFinType1, BsmtFinType2, Functional, Fence, BsmtExposure, GarageFinish, LandSlope, LotShape, PavedDrive, Street, Alley, CentralAir, MSSubClass, OverallCond, YrSold, MoSold* need to be encoded into ordinal variables.

```
for (col in c('FireplaceQu', 'BsmtQual', 'BsmtCond', 'GarageQual', 'GarageCond',
        'ExterQual', 'ExterCond','HeatingQC', 'PoolQC', 'KitchenQual', 'BsmtFinType1',
        'BsmtFinType2', 'Functional', 'Fence', 'BsmtExposure', 'GarageFinish', 'LandSlope',
        'LotShape', 'PavedDrive', 'Street', 'Alley', 'CentralAir', 'MSSubClass', 'OverallCond',
        'YrSold', 'MoSold')){
  lbl = LabelEncoder$new()
  full[[col]] = lbl$fit_transform(full[[col]])
}
```

Chossing Variables

```
# Selecting these important variables
predictors <- c('UsedToTrain','MSZoning','Neighborhood','BldgType','HouseStyle','OverallQual','OverallCond','YearBuilt', 'Ex
terQual','ExterCond','BsmtQual','BsmtCond','TotalBsmtSF','HeatingQC', 'CentralAir','Electrical','GrLivArea','BedroomAbvGr',
'KitchenAbvGr','KitchenQual','TotRmsAbvGrd','Functional','Fireplaces','FireplaceQu','GarageArea','GarageQual','GarageCond',
'OpenPorchSF','PoolArea','Fence','MoSold','YrSold','SaleType','SaleCondition','SalePrice','logSalePrice')
```

Splitting Train and Test Dataset

```
train.processed <- full[,predictors] %>% filter(UsedToTrain==TRUE)
test.processed <- full[,predictors] %>% filter(UsedToTrain==FALSE)
```

Modelling

M1: Linear Regression Model

Dividing Training and Validation

```
set.seed(1)
train.index <- sample(c(1:dim(train.processed)[1]), dim(train.processed)[1]*0.9)
m1.train <- train.processed[train.index, ]
m1.valid <- train.processed[-train.index, ]
```

Building the Model

```
m1 <- lm(logSalePrice~.-(logSalePrice+SalePrice+UsedToTrain), data=m1.train)
```

Prediction Analysis

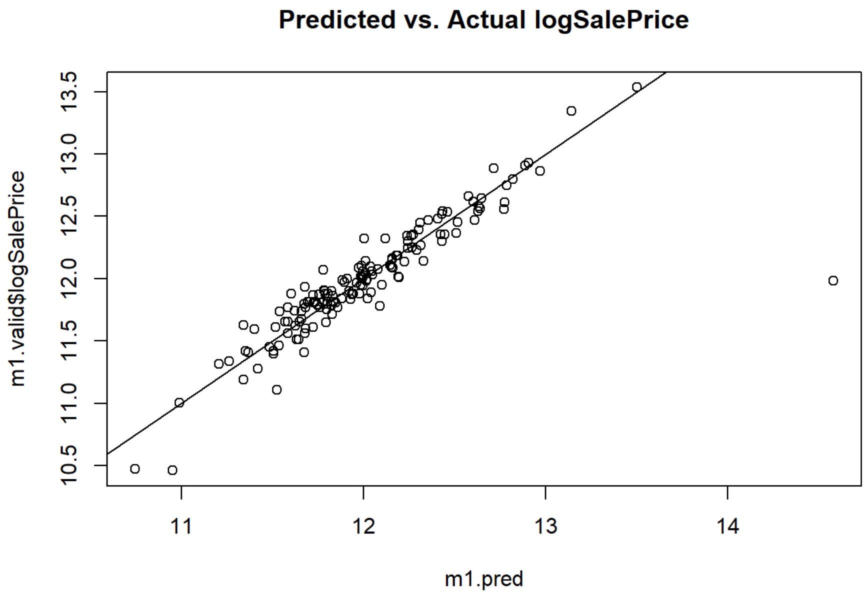
```
m1.pred <- predict(m1, newdata=m1.valid, type="response")
m1.res <- m1.valid$logSalePrice - m1.pred
head(cbind("Predicted" = m1.pred, "Actual" = m1.valid$logSalePrice, "Residual" = m1.res), n=10)
```

```
##    Predicted   Actual    Residual
## 1   11.76097 11.77144  0.01046558
## 2   12.78888 12.75130 -0.03758258
## 3   11.79592 11.64833 -0.14759466
## 4   11.51605 11.60824  0.09218576
## 5   11.68023 11.77144  0.09120512
## 6   11.35152 11.41861  0.06709038
## 7   11.58374 11.76718  0.18343684
## 8   11.58269 11.56172 -0.02097260
## 9   11.54091 11.73607  0.19516155
## 10  11.41955 11.27720 -0.14235157
```

```
accuracy(m1.pred, m1.valid$logSalePrice)
```

```
##                  ME      RMSE       MAE        MPE      MAPE
## Test set -0.0178656 0.2489712 0.1127157 -0.1604401 0.9486125
```

```
plot(m1.pred, m1.valid$logSalePrice, main = "Predicted vs. Actual logSalePrice")
abline(0,1)
```


Predicted vs. Actual logSalePrice
m1.valid$logSalePrice
10.5
11.0
11.5
12.0
12.5
13.0
13.5
11
12
13
14
m1.pred