House Prices - Advanced Regression Techniques

Binhao Chen

Machine Learning

Final Project

This is a teamwork by Sheng-ya Mei and Binhao Chen.

Business Context:

When considering dream houses, home buyers may not give priority to the height of the basement ceiling or the proximity to an east-west railroad. Everyone has their own priority to the dream house, but the price is decided by the house markets.

In this competition, the business problem we are going through is how to predict the reasonable price for different types of houses. The dataset contains 79 explanatory variables that can specifically describe every detail of residential homes, also it brings insights into machine learning to apply LASSO or Random Forest even Gradient Boosting.

After getting the accurate model, the home buyers can check the money they need to buy their own dream houses. House dealers can also set a reasonable price for their customers, preventing the premium price on houses caused by fluctuation. House companies can use the model to set reputations around the field and then convince more customers to buy dream houses.

Approach

The dataset we get is 1460 records with 80 features, therefore, the first idea coming up is that dealing with features is the major problem and then we are going to deal it with LASSO rather than Linear Regression. Also, from the instruction of the competition, they suggest we adopt advanced regression techniques like random forest and gradient boosting. The following paragraphs are written by the steps of Random Forest.

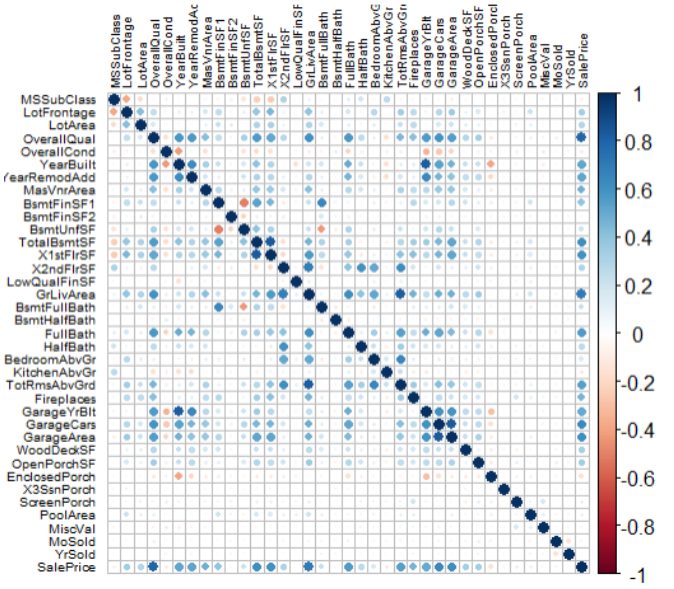
Explore Data Analysis

Among 80 features, there are 37 numeric features and 43 categorial features. Therefore, we will deal with it separately. Firstly, we are going through the numeric feature.

```
# Select only numeric columns from our train data. We will omit all null
values for now
train_dat_numeric <- select_if(train_dat, is.numeric)
corr <- cor(na.omit(train_dat_numeric))
corrplot(corr, tl.cex=0.5, tl.col='black')
data.frame(cor(na.omit(train_dat_numeric)))
```

From the correlation, we set 0.6 as the threshold for good correlation and we can find that *OverallQual*, *TotalBsmtSF*, *X1stFlrSF*, *GrLivArea*, *GarageCars*, *GarageArea* are having great correlation with *SalePrice*. Therefore, we will keep them for the model.

On the other hand, we are going to transform some features with low correlation with high skewness, to make sure that the distribution is similar to normal distribution.

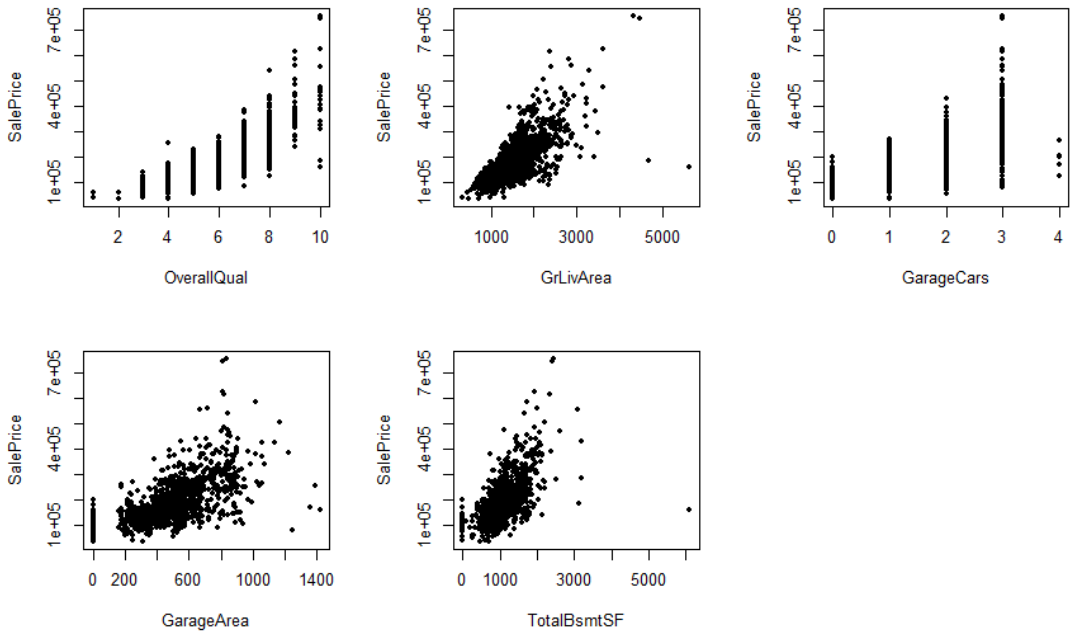


	skew <dbl>	na_cor <dbl>
OverallQual	0.2874151	0.7978807
TotalBsmtSF	1.7525671	0.6156122
X1stFlrSF	1.3619573	0.6079691
GrLivArea	1.5478862	0.7051536
GarageCars	0.2057410	0.6470336
GarageArea	0.7329115	0.6193296
SalePrice	1.9310267	1.0000000

Outliers

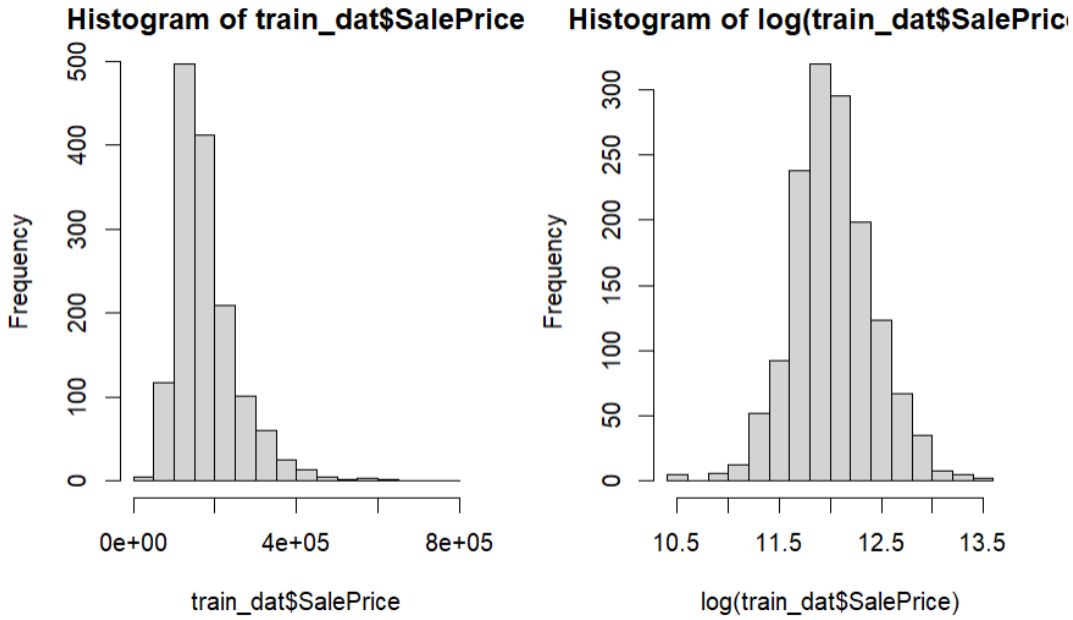
Focusing on these powerful features, we are intending to find the outliers, in order to prevent exaggerated impacts on training model. We drop the value that *TotalBsmtSF* over 5000 and *X1stFlrSF* over 4000, and *GrLivArea* over 4000 and *SalePrice* less 300000. These two points in the graphs are far from the groups.

```
train_dat[train_dat$TotalBsmtSF > 5000 & train_dat$X1stFlrSF > 4000, ]
train_dat[train_dat$GrLivArea > 4000 & train_dat$SalePrice < 300000, ]
```



Log Transform

Continuing from the skewness, we find that SalePrice is extremely skewed to the right and a majority part of the data stays on the left side. Therefore, we will adopt log transformation for SalePrice and then the distribution is almost normalized.



Cleaning Data

Cleaning data is the most complex process in the whole model. We need to distinguish them from fake null values, filling mean values, or filling mode values.

	parameter <chr>	na_count <int>
PoolQC	PoolQC	1452
MiscFeature	MiscFeature	1404
Alley	Alley	1367
Fence	Fence	1177
FireplaceQu	FireplaceQu	690
LotFrontage	LotFrontage	259
GarageType	GarageType	81
GarageYrBlt	GarageYrBlt	81
GarageFinish	GarageFinish	81
GarageQual	GarageQual	81

1-10 of 19 rows

Fake Null Values

It is really tricky because some categorical data contain a Null value itself. However, the machine detects the Null value as Null. Hence, we consider it as Fake Null Values and then we will fill all the Null as Null itself back.

Fill.na

1. Mode

   Other than the Null value, we applied all mode values to the category data and make sure that all missing value is well fixed.

2. Mean

   Other than categorical data, we adopt mean values for all numerical data. The process is that we will find the median of 'LotFrontage' after grouping by feature 'Neighborhood' then we will use this median to fill out missing 'LotFrontage' based on the 'Neighborhood' they are in.

Remove Unnecessary Feature

Other than the Null value and Fill.na Values, there is one feature named *Utilities* that all values from training and test are *AllPub*, so we can ignore the effect of Utilities.

```
train_dat <- train_dat %>%
  mutate(PoolQC = ifelse(is.na(PoolQC),'None',PoolQC),
         MiscFeature = ifelse(is.na(MiscFeature),'None',MiscFeature),
         Alley = ifelse(is.na(Alley),'None',Alley),
         Fence = ifelse(is.na(Fence),'None',Fence),
         FireplaceQu = ifelse(is.na(FireplaceQu),'None',FireplaceQu),
         GarageType = ifelse(is.na(GarageType),'None',GarageType),
         GarageYrBlt = ifelse(is.na(GarageYrBlt),0,GarageYrBlt),
         GarageFinish = ifelse(is.na(GarageFinish),'None',GarageFinish),
         GarageQual = ifelse(is.na(GarageQual),'None',GarageQual),
         GarageCond = ifelse(is.na(GarageCond),'None',GarageCond),
         BsmtExposure = ifelse(is.na(BsmtExposure),'None',BsmtExposure),
         BsmtFinType2 = ifelse(is.na(BsmtFinType2),'None',BsmtFinType2),
         BsmtQual = ifelse(is.na(BsmtQual),'None',BsmtQual),
         BsmtCond = ifelse(is.na(BsmtCond),'None',BsmtCond),
         BsmtFinType1 = ifelse(is.na(BsmtFinType1),'None',BsmtFinType1),
         MasVnrType = ifelse(is.na(MasVnrType),'None',MasVnrType),
         MasVnrArea = ifelse(is.na(MasVnrArea),0,MasVnrArea),
         Electrical = ifelse(is.na(Electrical),mode(train_dat$Electrical),Electrical)
         )
```

Finally, we will use the *Null indicator* to check all Null value are fixed.

Modeling

Label encoding

For the categorial data, we want to use encoding for their category. We adopt the one hot encoding and use the formular as.factor to transform these features.

```
# Transform to categorical features
full_dat$MSSubClass <- as.factor(full_dat$MSSubClass)
full_dat$OverallQual<- as.factor(full_dat$OverallQual)
full_dat$OverallCond<- as.factor(full_dat$OverallCond)
```

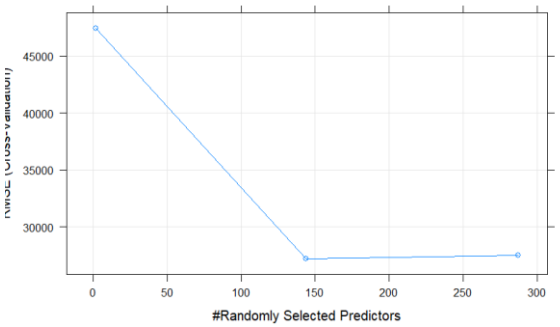
Model Training

1. Random Forest (Cross-validation)

   The best model is when we use is 144 mtry where the RMSE is 27235.28 and R square is 0.8915, under the condition of 5 cross-validations. If we plot our rf_model, we can see the point where the machine chose to be the best number

of predictors with the least RMSE. This shows our bias and variance trade-off. We want a model that is as simple as possible and as complex as necessary.

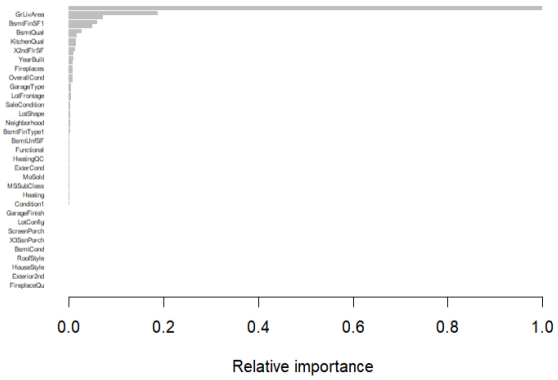
```
Random Forest

1458 samples
  79 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 1167, 1167, 1166, 1166, 1166
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
    2   47403.93  0.7921125  30176.90
  144   27235.38  0.8915446  17109.41
  287   27497.41  0.8872381  17391.55

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 144.
```



2. XGBoost

Applied XGBoost, the RMSE is much less than the Random Forest result as verbose increase to 7 and more. The XGBoost shows the importance rank. *GrLivArea* and *BsmtFinSF1is* the most important features related with the regression model.

```
[2]     train-rmse:101755.866546
[3]     train-rmse:73971.145460
[4]     train-rmse:54384.955619
[5]     train-rmse:40560.586077
[6]     train-rmse:31056.475242
[7]     train-rmse:24280.090232
[8]     train-rmse:19738.704612
[9]     train-rmse:16599.367110
[10]    train-rmse:14384.717832
```



Results:

The final result from Kaggle is 0.1552. We think we have done a really good job following all materials from 656 Machine Learning Course.