# Analysis Report for House-Price-Prediction

## 0. Introduction

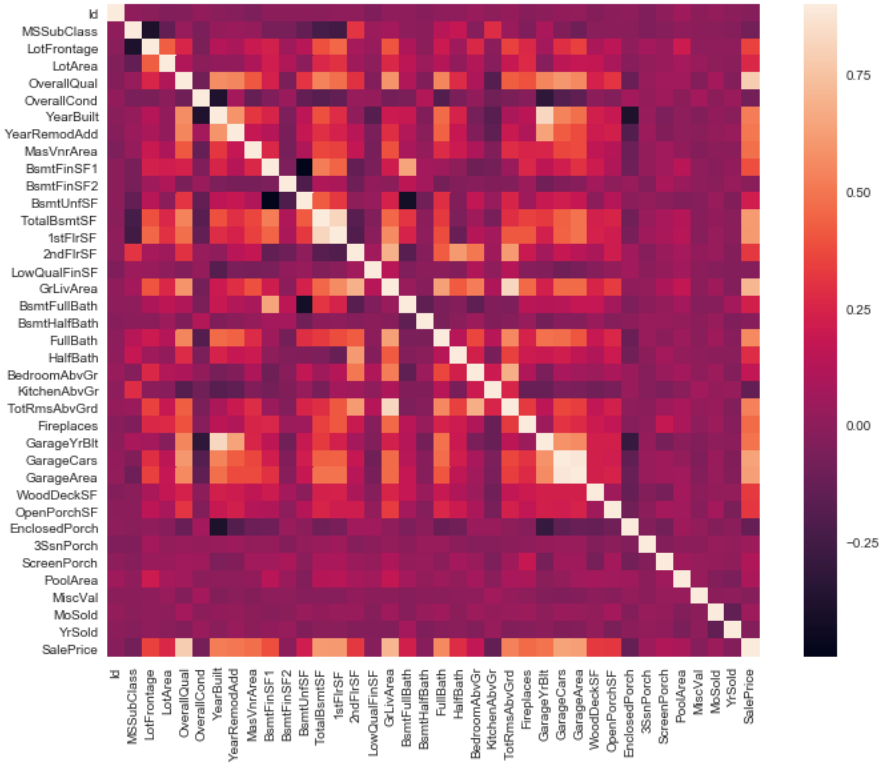
## 1. Data-Exploring

The given datasets (train.csv and test.csv) provide us 79 features describing almost every aspect of residential homes. Before we start predicting the sale price for the houses in test set. We first explore the train set which contains the final sale price, so that we can find the relationship between features and price, as well as the relationship between features. This part will help a lot in our data processing part.

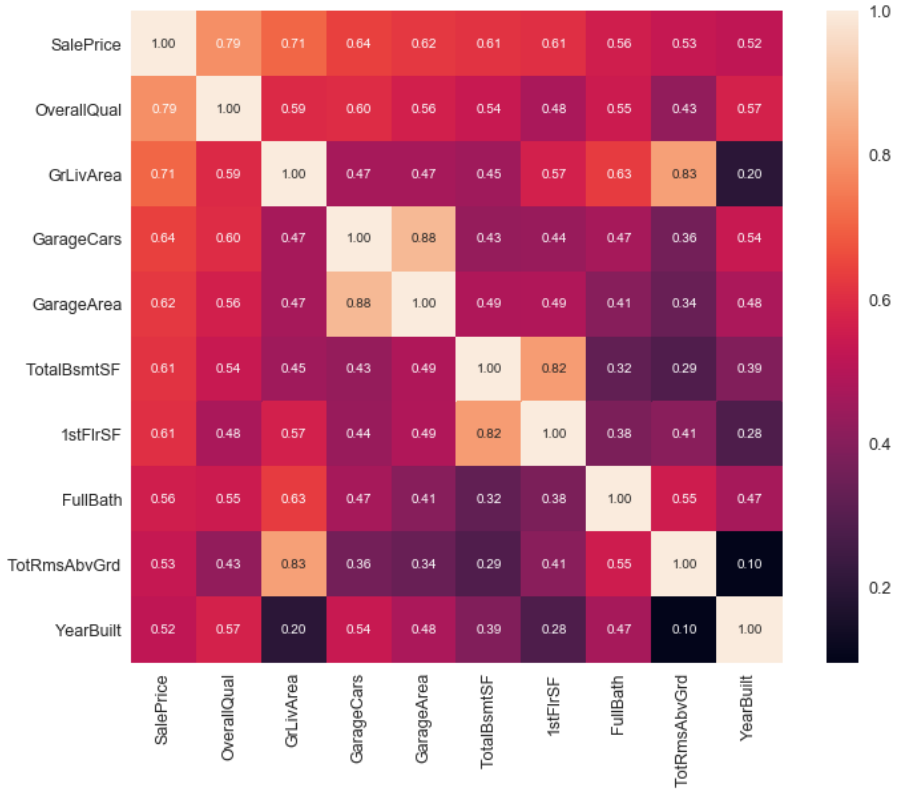
## 1.1 Data Features Analysis

We start by describing the features and wrote the summary statistics including data type, count, mean, most common, frequency and so on into explore.csv files. It is clear that the sources have 3 data types: int, float and object, the object features should be transformed into numerical type. Some features have many null values, which need to be filled out in the future.

However this cannot show the relationship between features clearly. So here goes the interesting part. We computed pairwise correlation between columns and shown in heatmap:

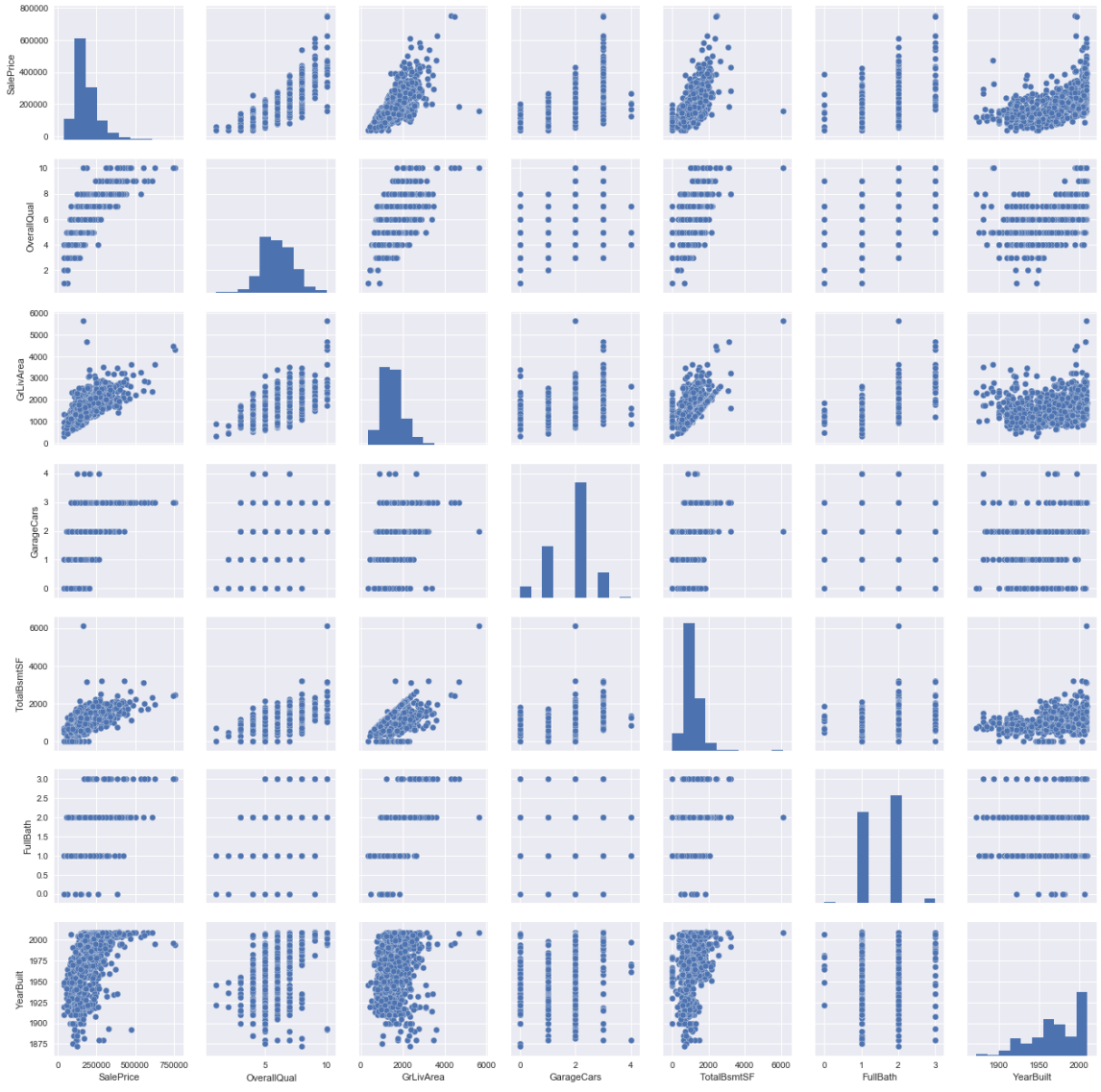



This demonstrate the correlation between features, lighter color means higher correlation while darker color means lower correlation. Then we got 10 features which have the highest correlation with sale price, and shown in heatmap:



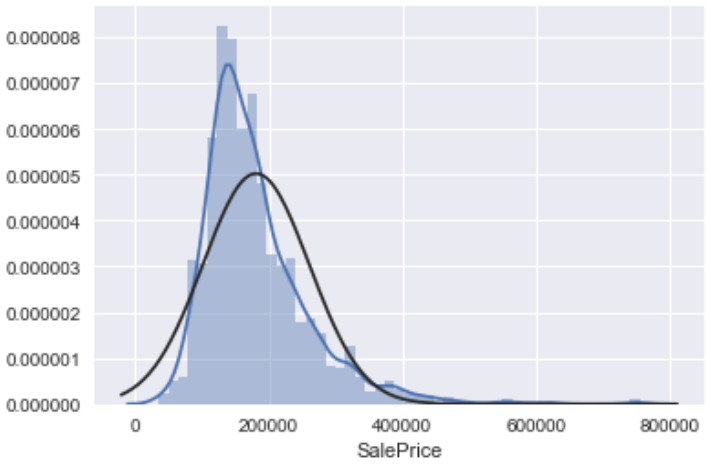
This is clear that the overall quality most influence the price, then the living area, the garage area, the basement area, number of bathrooms & other rooms, and the original construction date. This seems well according to our life experience.

Use the features we got above, it is easy to draw their scatter diagrams:



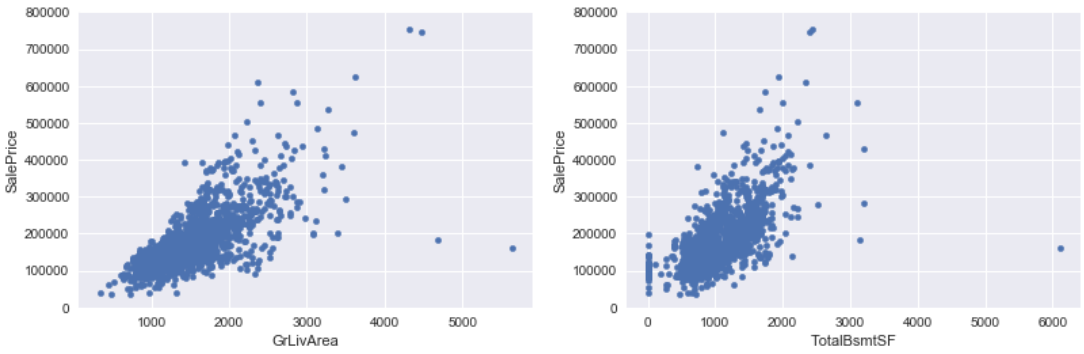

This diagram will help us a lot in feature choosing and transformation.

Draw histograms to see if the price conform to normal distribution:



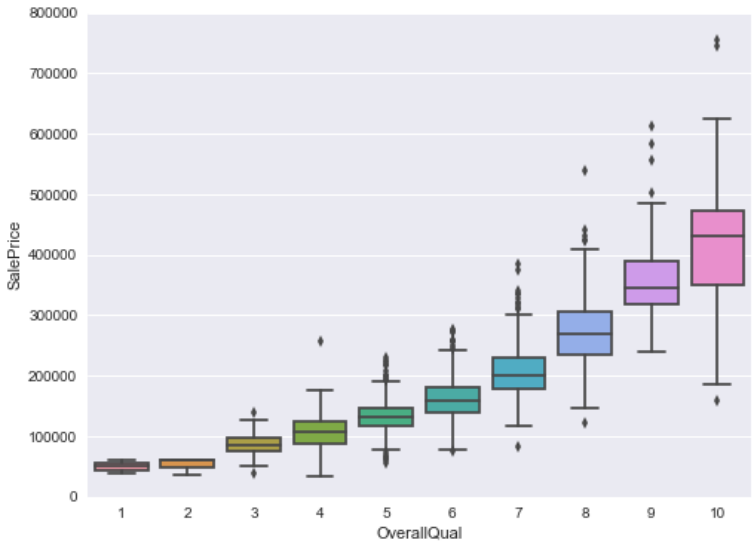

The price does not conform normal distribution.

Draw single diagrams for sale price and living area & basement area:



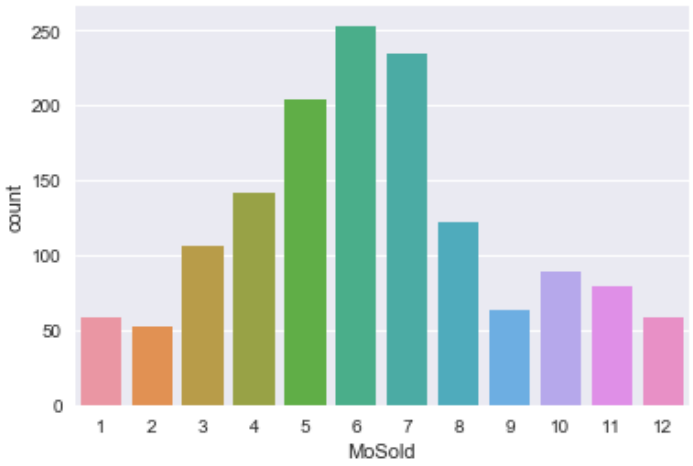

From the scatter diagrams, we can find outliers at the lower right of the images.

Draw box diagrams for sale price and overall quality:




From the box diagram, we can find outliers at some good quality houses.

Draw bar diagrams for the quantity of sale in different months:




From the bar diagram, we can find the peak is in May, June and July.

## 1.2 Data Quality Analysis

In this part we concatenate the features in training and testing set. From the overall analysis in last part, we can find many null values and outliers exist in the datasets. They might influence the quality of model training, so we searched and located the null values first to prepare for data cleaning.

For the null values related with pool quality (poolQC):

Print out the mean values of pool area in different quality, so that we can fill out null quality by giving the quality of mean area which has the most similar area size.

```
不同Poolqc的PoolArea的均值
 PoolQC
Ex    359.75
Fa    583.50
Gd    648.50
Name: PoolArea, dtype: float64
查看有PoolArea数据但是没有PoolQC的数据
      PoolQC  PoolArea
2420    NaN       368
2503    NaN       444
2599    NaN       561
查看无PoolArea数据但是有PoolQC的数据
 Empty DataFrame
Columns: [PoolQC, PoolArea]
Index: []
```

For the null values related with garage:

| index | missingNum | missingRatio | existNum | train_notna | test_notna |
|---|---|---|---|---|---|
| GarageType | 157 | 5.378554 | 2762 | 1379 | 1383 |
| GarageYrBlt | 159 | 5.447071 | 2760 | 1379 | 1381 |
| GarageFinish | 159 | 5.447071 | 2760 | 1379 | 1381 |
| GarageCars | 1 | 0.034258 | 2918 | 1460 | 1458 |
| GarageArea | 1 | 0.034258 | 2918 | 1460 | 1458 |
| GarageQual | 159 | 5.447071 | 2760 | 1379 | 1381 |
| GarageCond | 159 | 5.447071 | 2760 | 1379 | 1381 |

The null values of garage condition, garage type and so on might be wrote as the zero values on garage area & car capacity (means the houses don't have garages). Count them so that we can fill out the null values by "none" and "0".

For the null values related with basement:

| index | missingNum | missingRatio | existNum | train_notna | test_notna |
|---|---|---|---|---|---|
| BsmtQual | 81 | 2.774923 | 2838 | 1423 | 1415 |
| BsmtCond | 82 | 2.809181 | 2837 | 1423 | 1414 |
| BsmtExposure | 82 | 2.809181 | 2837 | 1422 | 1415 |
| BsmtFinType1 | 79 | 2.706406 | 2840 | 1423 | 1417 |
| BsmtFinSF1 | 1 | 0.034258 | 2918 | 1460 | 1458 |
| BsmtFinType2 | 80 | 2.740665 | 2839 | 1422 | 1417 |
| BsmtFinSF2 | 1 | 0.034258 | 2918 | 1460 | 1458 |
| BsmtUnfSF | 1 | 0.034258 | 2918 | 1460 | 1458 |
| TotalBsmtSF | 1 | 0.034258 | 2918 | 1460 | 1458 |
| BsmtFullBath | 2 | 0.068517 | 2917 | 1460 | 1457 |
| BsmtHalfBath | 2 | 0.068517 | 2917 | 1460 | 1457 |

Same as before, we could find that the basement quality, condition, exposure and quality of finish area have a lot null values.

First, there exists many data (1256) which has same values between quality and condition score. So we fill out basement quality and condition by each other if one is null and the other one is not null.

Second, we found 3 rows with the basement exposure is null and the other feature is not null, filled out them by mode.

Third, fill out the quality of finished area by mode.

Last, for other features with several null, simply filled out them by “none” and “0”.

For the null values related with masonry veneer (MasVnrType & MasVnrArea):

```
不同type的MasVnrArea的均值
 MasVnrType
BrkCmn     161.0
BrkFace    203.0
None         0.0
Stone      200.0
Name: MasVnrArea, dtype: float64
MasVnrtype为空而MasVnrArea不为空
      MasVnrType  MasVnrArea
2610         NaN       198.0
```

As what we did before, we first found 1 row with the type is null and the area is not null. Like pool quality and pool area, we can fill out null type by giving the type of mean area which has the most similar area size.

For the null values related with dwelling types (MSSubClass and MSZoning):

| | MSSubClass | MSZoning |
|---|---|---|
| 1915 | 30 | NaN |
| 2216 | 20 | NaN |
| 2250 | 70 | NaN |
| 2904 | 20 | NaN |

MSSubClass use number 20, 30, …, 120 to represent different dwelling types. MSZoning use characters to represent different zoning types. We use cross-

tabulation to show their relationship:

| MSZoning | C (all) | FV | RH | RL | RM |
|---|---|---|---|---|---|
| MSSubClass | | | | | |
| 20 | 3 | 34 | 4 | 1016 | 20 |
| 30 | 8 | 0 | 2 | 61 | 67 |
| 40 | 0 | 0 | 0 | 4 | 2 |
| 45 | 0 | 0 | 1 | 6 | 11 |
| 50 | 7 | 0 | 2 | 159 | 119 |
| 60 | 0 | 43 | 0 | 529 | 3 |
| 70 | 4 | 0 | 3 | 57 | 63 |
| 75 | 0 | 0 | 0 | 9 | 14 |
| 80 | 0 | 0 | 0 | 115 | 3 |
| 85 | 0 | 0 | 0 | 47 | 1 |
| 90 | 0 | 0 | 4 | 92 | 13 |
| 120 | 0 | 19 | 6 | 117 | 40 |
| 150 | 0 | 0 | 0 | 1 | 0 |
| 160 | 0 | 43 | 0 | 21 | 64 |
| 180 | 0 | 0 | 0 | 0 | 17 |
| 190 | 3 | 0 | 4 | 31 | 23 |

We can fill out type of zoning by finding the most common ones, like 20 to RL, 30 to RM and 70 to RM.

For the null values related with object features:

We can fill out them by their mode values.

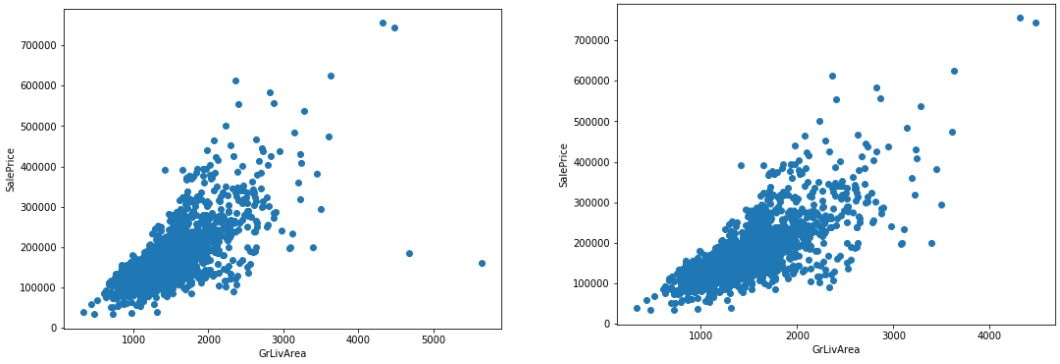
## 2. Data Preprocessing

After the deep exploring in first part, we now start preprocessing the data sets, cleaning the data and transforming the csv data into a 2-dimensional matrix (which just contains float and integer numbers), as well as adding new features to display the relationships in numerical. So that the new data set can be fitted in learning models well.

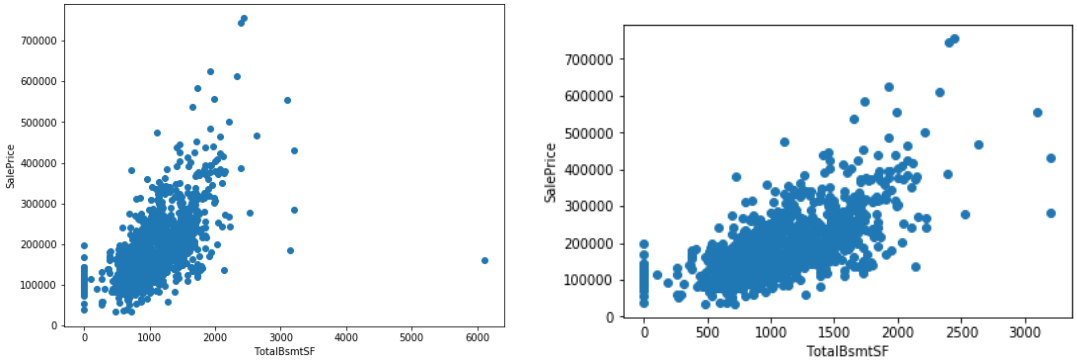
## 2.1 Data Cleaning

First we filled out the null values as what we said in data quality analysis. Btw you

must remember the outliners we found in data features analysis. These exist outliers between living areas & basement areas and the sale price, we dropped them conditionally. This can be shown by following graphs:




Scatter graphs of living areas and price (before & after)




Scatter graphs of basement areas and price (before & after)

## 2.2 Data Transformation

First, for object type features:

1. Some features can be transformed simply by 0, 1, 2, 3, … as the descriptions of these features are independent with each other like shown below:

```
ExterQual: Evaluates the quality of the material on the exterior

       Ex      Excellent
       Gd      Good
       TA      Average/Typical
       Fa      Fair
       Po      Poor
```

ExterQual: Ex to 5, Gd to 4, TA to 3, Fa to 2, Po to 1

BsmtFinType1: Rating of basement finished area

| | |
|---|---|
| GLQ | Good Living Quarters |
| ALQ | Average Living Quarters |
| BLQ | Below Average Living Quarters |
| Rec | Average Rec Room |
| LwQ | Low Quality |
| Unf | Unfinshed |
| NA | No Basement |

BsmtFinType: NA to 0, Unf to 1, LwQ to 2, Rec to 3, BLQ to 4, ALQ to 5, GLQ to 6

2. Some features can be transformed simply by 0, 1 as the descriptions of these features can be separate into two situations like shown below:

SaleCondition: Condition of sale

| | |
|---|---|
| Normal | Normal Sale |
| Abnorml | Abnormal Sale - trade, foreclosure, short sale |
| AdjLand | Adjoining Land Purchase |
| Alloca | Allocation - two linked properties with separate deeds, typically condo with a garage unit |
| Family | Sale between family members |
| Partial | Home was not completed when last assessed (associated with New Homes) |

SaleCondition: Normal, Partial to 0, others to 1 (might reduce the price)

MoSold: Month Sold (MM)

MoSold: 5, 6, 7 to 1, others to 0 (May, June and July are sale peak)

3. For quality and condition features, separate them into good quality, bad quality, good condition and bad condition. Concatenate them into quality list.
4. For features related with time, construct new features for several situations: original construction date are different with remodel date; remodel date later than sold date; construction date later than sold date; the house age (now minus construction date); sold time (now minus sold date). Then map the year in YearBuilt (original construction date) and YearRemodAdd (remodel date) into several year groups (20 years as a group). Concatenate them into year list.

Second, for numerical type features:

Note that we did transformed all the object features into numerical in last part.

Scaling all these data by their 3/4 quantile. Then we normalize the data by using log so that they can conform to normal distribution, which can get faster fitting speed in model construction. Next we converted the data into dummy variables (one-hot coding), so that all variables become 0 and 1. Finally we generated many new features by linking the area features with the quality list, year list as well as the other object features in last part.

## 3. Model Construction

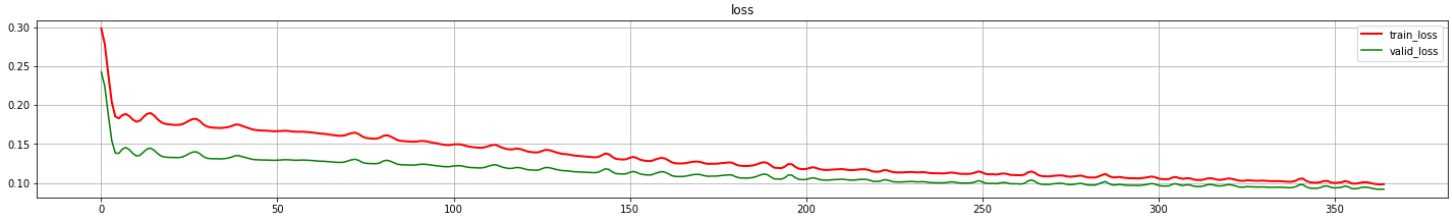
This part runs much faster than the data processing part. It seems that once we got good standard datasets, the only thing is fit them into the models package and adjust suitable parameters. However we still took a lot time on learning the meaning of models, so that we can trained the most appropriate models. In machine learning part, we chose lasso model, elastic net and xgboost for training and prediction. In deep learning part, we chose Convolutional Neural Network.

## 3.1 ML Models and Performance

| | Loss value | Std Deviation of loss |
|---|---|---|
| Lasso Model | 0.1101 | 0.0055 |
| Elastic Net | 0.1101 | 0.0057 |
| XGBoost | 0.1226 | 0.0043 |

## 3.2 CNN Models and Performance

Before we found keras, we first used tensorflow to write our CNN model. We added one line and reshape the trained data like a 20*20 pixels picture. Then we used two convolution layers and one fully connected layer. The patch in two layers is 2*2. In first layer, input 1 and output 32 feature maps (20*20*32). In second layer, input 32 and output 64 feature maps (20*20*64). This looks like a three-dimensional picture with height of 64. Before we put the picture into fully connected layer, we mapped the picture into an array with length of 512. Every neuron from the last layer (512) is connected to every neuron of the fully-connected layer. Here is the loss value with the number of iterations.




Then we fitted keras into our tensorflow code. Keras is much easier for writing. However some parameters in packaged keras might not that suitable this time. So the loss value seems higher than tensorflow.

# 4. Application of TF.js

We found that the standard keras model can be loaded into tensorflow.js directly. Just by running a converter in tf.js, the model will be saved as an model.json. Then front end http load the json file and fetch the required data to draw API. However the preparing job, data processing part and model training part took us so many time, so I just finish the load job in front end. The visualization will be finished in the future.