House Price Prediction with Large-Scale Attributes

Member：

R08942035　許軒瑋(組長)
R08944011　陳胤銓
R08942094　胡凱欣
R08921058　楊捷安

1. Motivation

Having a place to live is one of the basic needs of human beings. Despite that, we can rent a house and choose not to buy it. However, having houses that belong to us is one of our common dreams. Therefore, we are interested in the price of the house is reasonable or not. Given some attributes of the house, we like to predict how much the house worth. For instance, we would like to have two bedrooms, one kitchen and one bathroom in our dream house. The house needs to near MRT station and the house age can't surpass 10 years. Forwarding these attributes into our model, we would like to get the predicted house price and judge if the house price is a rip-off or a good deal. By using our predicted price, it can also become one of the judgment criteria in property investment.

2. Related Work

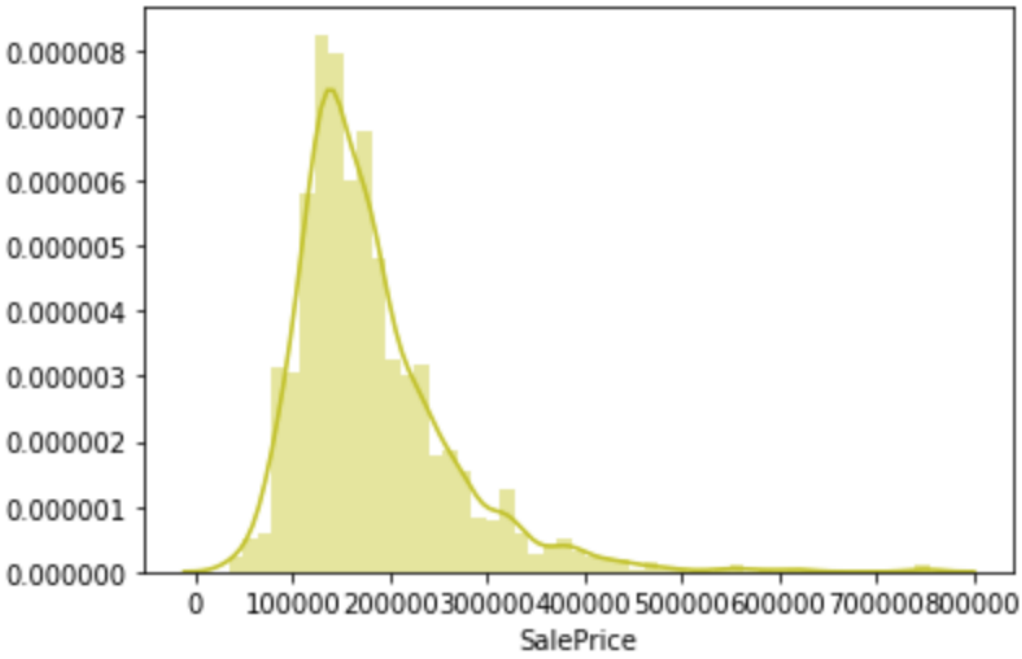
There are many similar prediction projects in Kaggle, like "Diamonds Price Prediction https://www.kaggle.com/shivam2503/diamonds", "Global Food Price Prediction https://www.kaggle.com/jboysen/global-food-prices", etc. All this kind of works want to find out which related factors will influence the price of the object, and which won't. Using the attributes correctly can help us to get a better algorithm to have better performance of prediction. Nevertheless, the attributes of the data usually wouldn't as many as this project. In the dataset of this project, the total number of attributes equals 81, of which 36 is quantitative, 43 categorical, Id and Sale Price. This means that we need to do lots of work in data analysis. On the other hand, there are only 1460 instances of training data. I think it is better to use some ML techniques like SVM

instead of using the Neural Network to accomplish our prediction algorithm.

3. Looking into data

The data was downloaded from Kaggle. It contains 1460 training data and 1459 testing data with 79 house attributes. The training data contains the sale price in each data row.

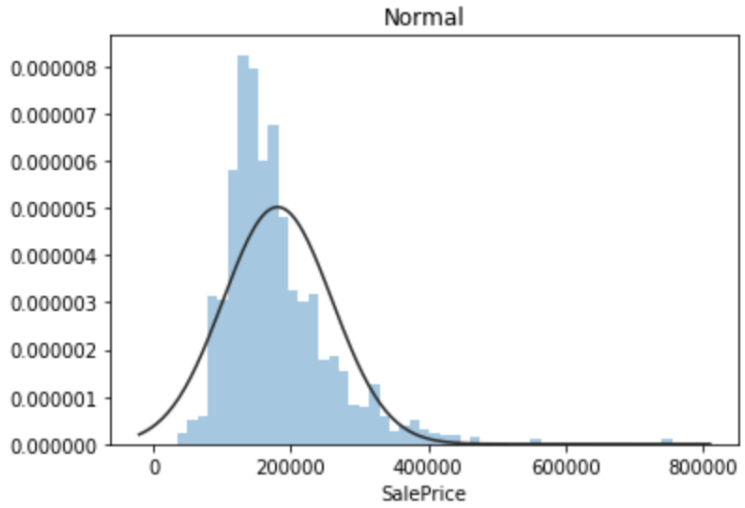
We first plot out the sale price's in order to take a glance at its distribution.



It has Skewness 1.88288 and Kurtosis 6.53268. It is obviously that it is not a normal distribution and has a positive skewness. The result is reasonable, because there should be seldom people that are rich enough to buy houses with high price.

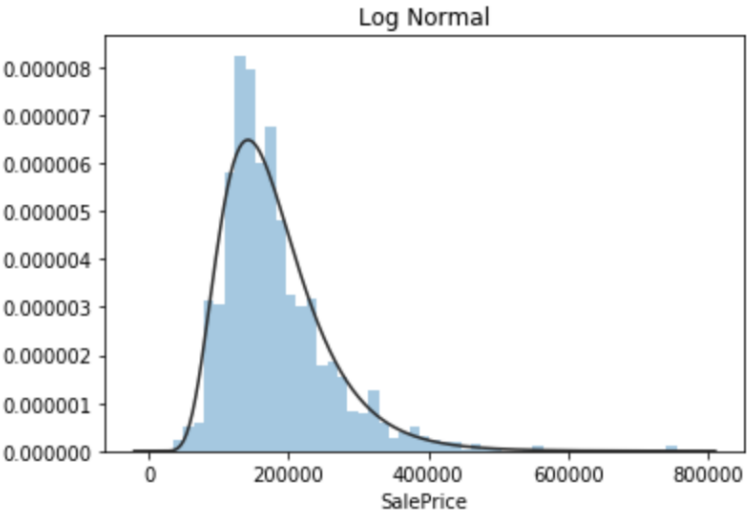
We also want to compare our data's distribution with other kind of distributions.

Normal distribution:



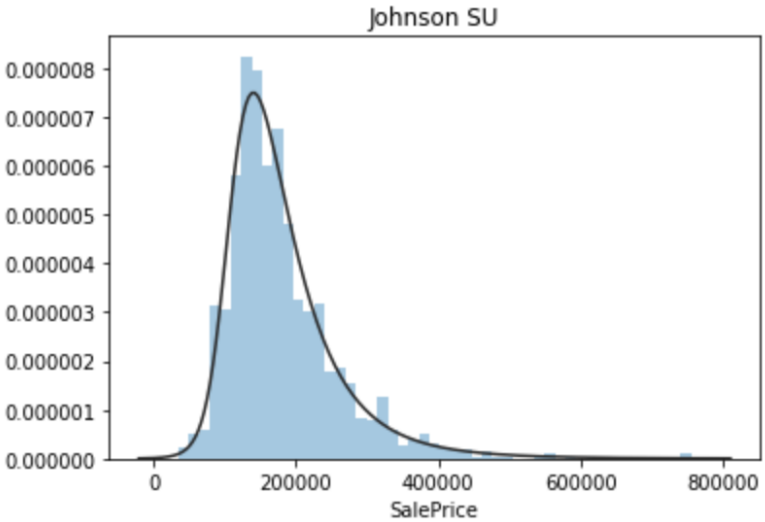
It shows that the distribution deviate from normal distribution.

Log-normal distribution:



The log-normal distribution fits our data better.

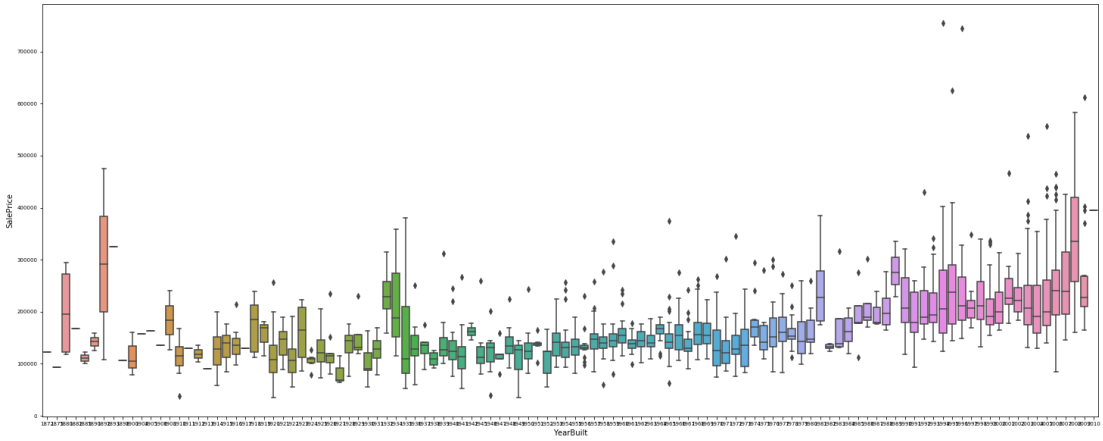
Johnson SU distribution:



We find an interesting result here. The house sale price fits Johnson SU distribution appropriately, which has been used successfully to model **asset returns for portfolio management.** Therefore, this might imply the sale price of houses may be a kind of investment.

The first intuition that comes to our mind is that the houses' sale price will have a strong positive correlation with the year they built.

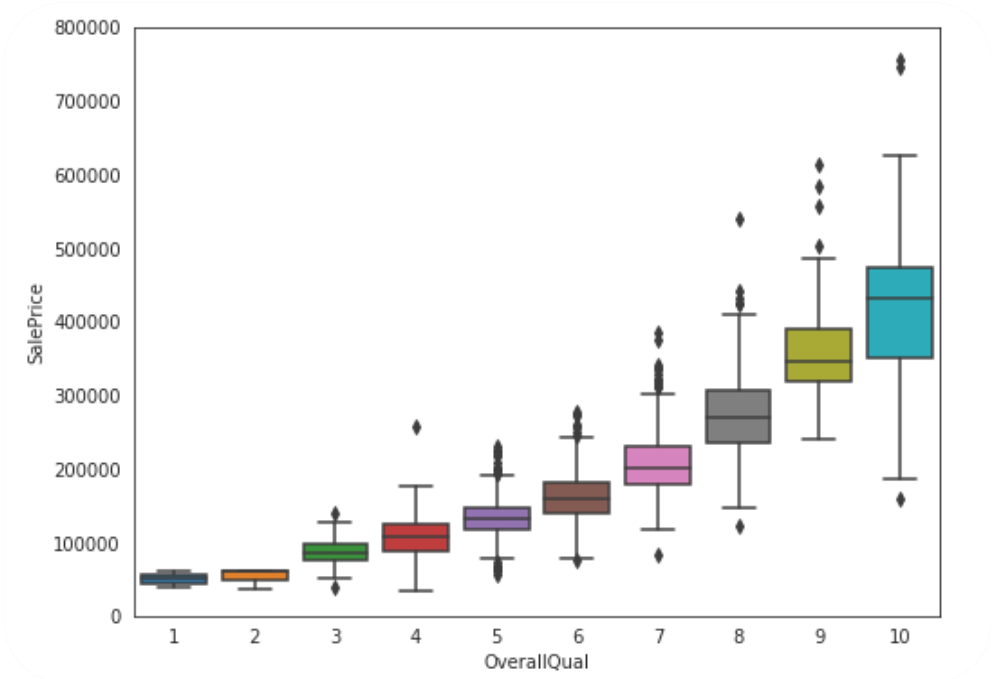
Box plot of SalePrice vs YearBuilt



This shows that the house price in nearly ten years have higher price, which prove our intuition.

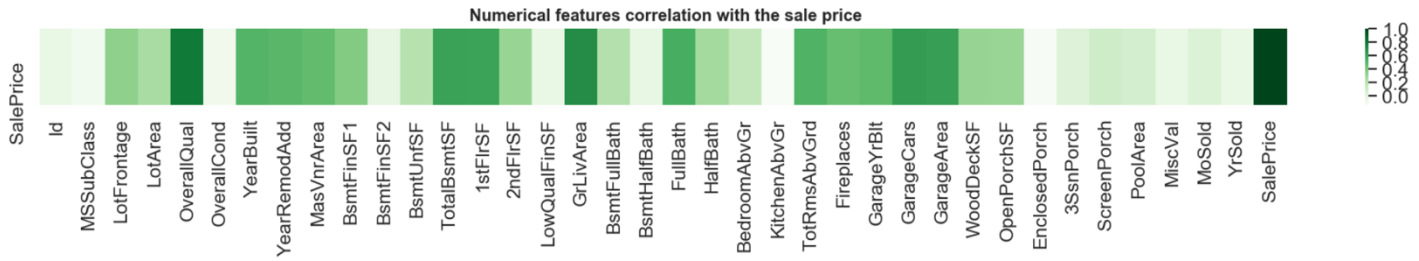
Moreover, the overall quality will also affect the price dramatically. The overall quality value in our data is rated as 1 to 10. The below table shows the quality that the number represent.

Box plot of SalePrice vs OverallQual



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

Correlation between Sale Price and attributes



	SalePrice
SalePrice	1
OverallQual	0.790982
GrLivArea	0.708624
GarageCars	0.640409
GarageArea	0.623431
TotalBsmtSF	0.613581
1stFlrSF	0.605852

FullBath	0.560664
TotRmsAbvGrd	0.533723
YearBuilt	0.522897
YearRemodAdd	0.507101
GarageYrBlt	0.486362
MasVnrArea	0.477493
Fireplaces	0.466929
BsmtFinSF1	0.38642

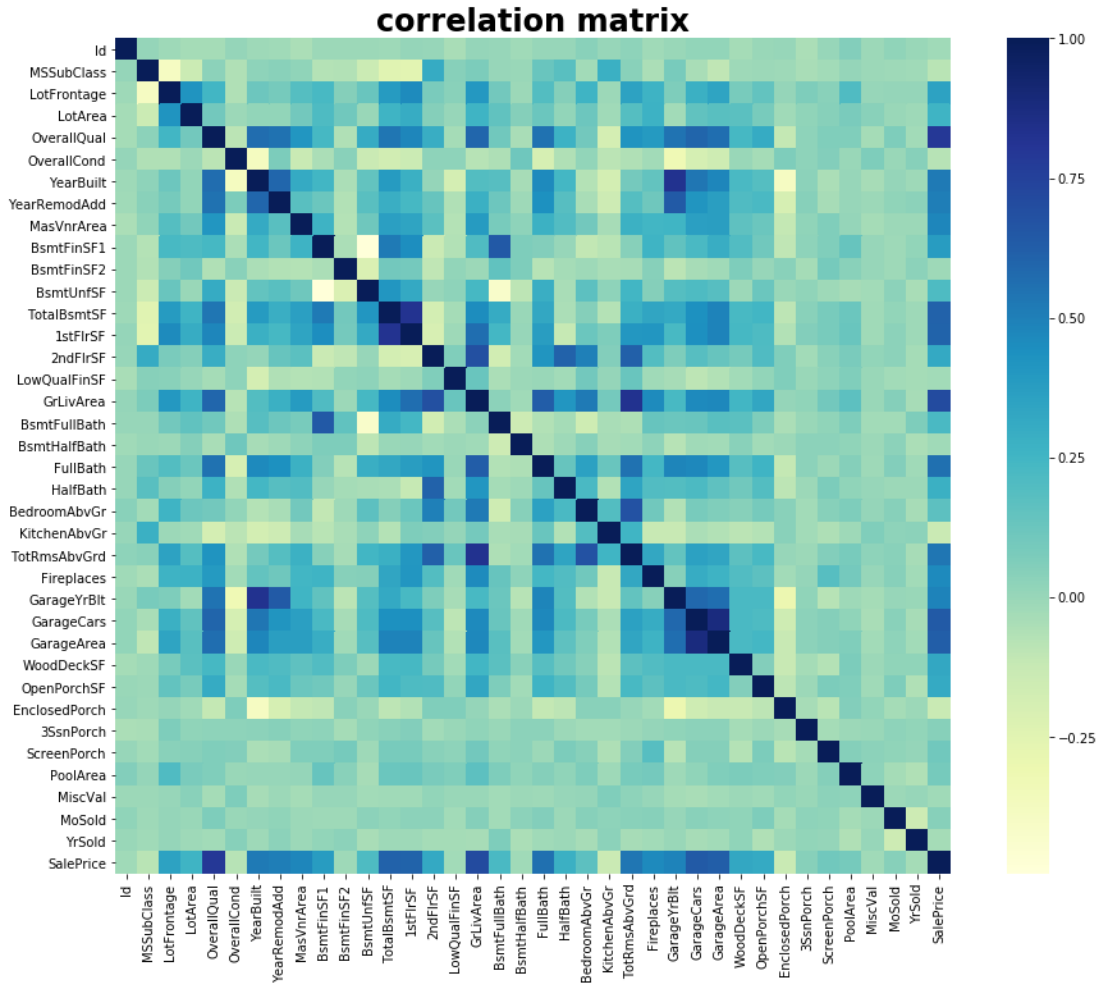
LotFrontage	0.351799
WoodDeckSF	0.324413
2ndFlrSF	0.319334
OpenPorchSF	0.315856
HalfBath	0.284108
LotArea	0.263843
BsmtFullBath	0.227122
BsmtUnfSF	0.214479

BedroomAbvGr	0.168213
KitchenAbvGr	0.135907
EnclosedPorch	0.128578
ScreenPorch	0.111447
PoolArea	0.0924035
MSSubClass	0.0842841
OverallCond	0.0778559
MoSold	0.0464322

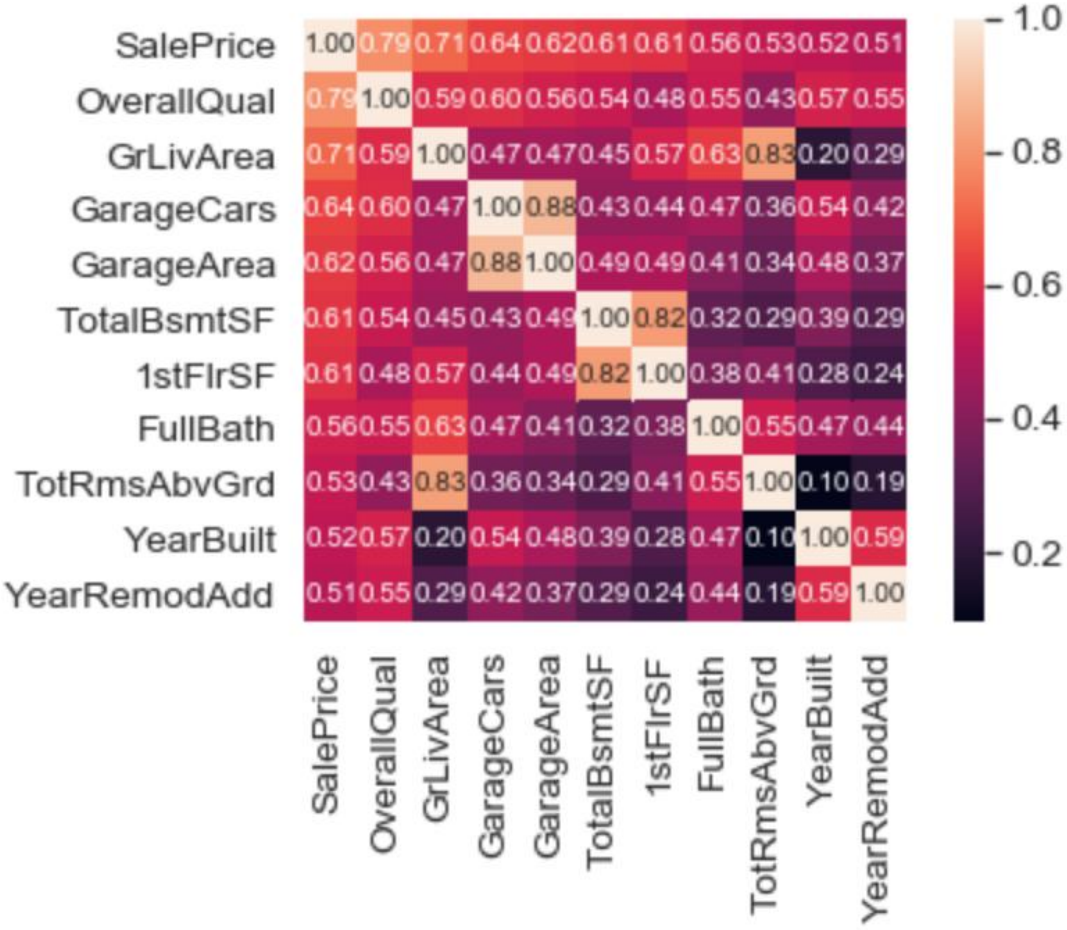
3SsnPorch	0.0445837
YrSold	0.0289226
LowQualFinSF	0.0256061
Id	0.0219167
MiscVal	0.0211896
BsmtHalfBath	0.0168442
BsmtFinSF2	0.0113781

Next, we want to understand the correlation between every attribute.

Correlation matrix:



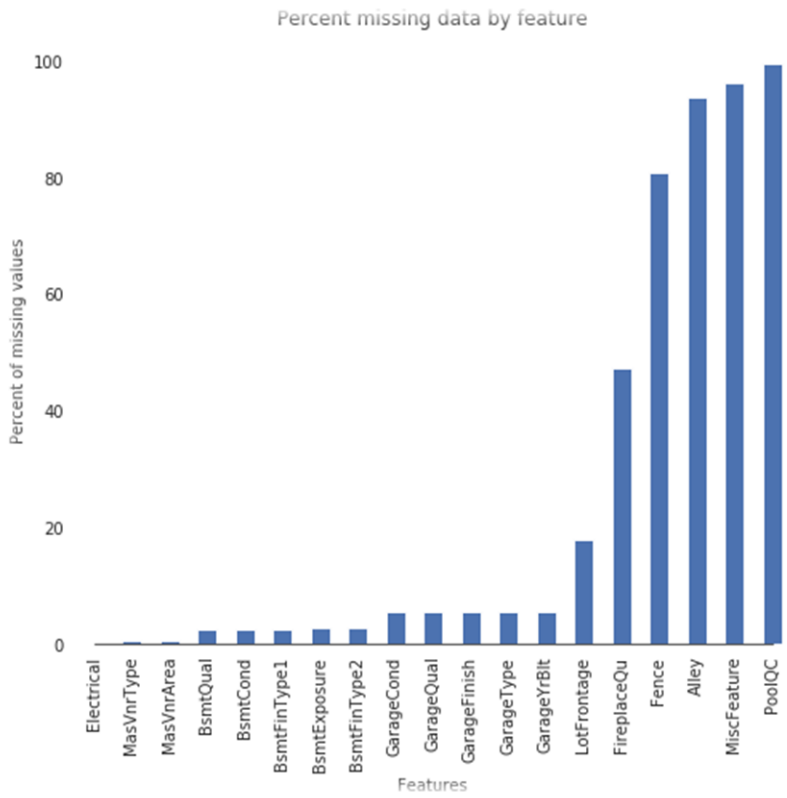
From the above correlation matrix, we pick the **top 10 highest** correlation and plot it out.



From the above matrix, garage area has 0.88 correlation with garage cars. Total rooms above grade (TotRmsAbvGrd) has 0.83 correlation with ground living area (GrLivArea). Those high correlation attributes look very reasonable. However, we will have to take care of these highly correlation attributes carefully in the method we applied in prediction.

4. Data Cleaning

There are lots of missing data in some particular attributes. We first look into which attributes have a lot of missing value.



	Count	Percent(%)
PoolQC	2907	99.691358
MiscFeature	2811	96.399177
Alley	2718	93.209877
Fence	2345	80.418381
FireplaceQu	1420	48.696845
LotFrontage	485	16.632373
GarageCond	159	5.452675
GarageQual	159	5.452675
GarageYrBlt	159	5.452675
GarageFinish	159	5.452675
GarageType	157	5.384088
BsmtCond	82	2.812071
BsmtExposure	82	2.812071
BsmtQual	81	2.777778
BsmtFinType2	80	2.743484
BsmtFinType1	79	2.709191
MasVnrType	24	0.823045
MasVnrArea	23	0.788752
MSZoning	4	0.137174
BsmtHalfBath	2	0.068587

We can see pool quality (PoolQC), MiscFeature (contains tennis court, elevator etc) and Alley has missing over 90 percent of value in our data. These missing attributes have a same common point. They are seldom appearing in our house. Swimming pool, alley, tennis court and elevator will only appear in luxury house, which make these attributes have missing value. Therefore, **we drop missing values that over 90%.**

Next, we fill the missing value and fill them with four types.

Type1: NA refers to none.

- Fence with missing value means no fence.
- Garage quality with missing value means no garage.

Type2: NA refers to 0

- Total square feet of basement area(TotalBsmSF) is 0.
- Other missing areas will also fill 0 in their missing value.

Type3: NA refers to typical values

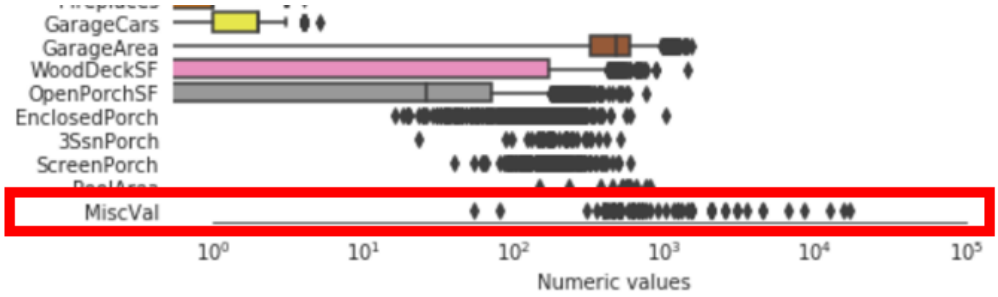
- **Ex. KitchenQual： Kitchen quality**
  - **Ex Excellent**
  - **Gd Good**
  - **TA Typical/Average**
  - **Fa Fair**
  - **Po Poor**

Type4: Using other features to fill in

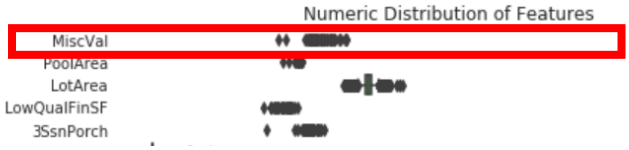
- Ex. **MSZoning**： MSSubClass
  - MSSubClass: Identifies the type of dwelling involved in the sale.
  - MSZoning: Identifies the general zoning classification of the sale.
- Ex. **LotFrontage**： Neighborhood + LotArea
  - LotFrontage: Linear feet of street connected to property
  - LotArea: Lot size in square feet
  - Neighborhood: Physical locations within Ames city limits

5. Feature Skewness

Before forwarding the data into our estimation model, we need to deal with feature skewness problem. Once the data contains skewness problem, the statistical model will be interfered to some degree. First, we plot out the skewness of the data attributes.



From the attached figure, we can see that the attribute “MiscVal” has unignorable positive skew. After normalize the skew attribute, we can see the distribution in the following figure.



By fixing the feature skewness, we can be more confident that the model can estimate more precisely.

6. Add Some interesting features

First, we add some feature with binary value “True” or “False”. For instance, “has2ndfloor” means if the house has second floor. In origin attributes set, there is only “2ndFlrSF” which indicates the square feet of the second floor. Adding this kind of attributes is like a hint to tell model

that maybe having second floor is a more important feature than the area size of second floor.

```
all_features['haspool'] = all_features['PoolArea'].apply(lambda x: 1 if x > 0 else 0)
all_features['has2ndfloor'] = all_features['2ndFlrSF'].apply(lambda x: 1 if x > 0 else 0)
all_features['hasgarage'] = all_features['GarageArea'].apply(lambda x: 1 if x > 0 else 0)
all_features['hasbsmt'] = all_features['TotalBsmtSF'].apply(lambda x: 1 if x > 0 else 0)
all_features['hasfireplace'] = all_features['Fireplaces'].apply(lambda x: 1 if x > 0 else 0)
```

Besides, we create some features by summing up related attributes. For instance, “Total_sqr_footage” is created by summing up the value of “BsmtFinSF1”, “BsmtFinSF2”, “1stFlrSF”, “2ndFlrSF”. By changing data granularity, it will have some effect on the result.

```
all_features['TotalSF'] = all_features['TotalBsmtSF'] + all_features['1stFlrSF'] + all_features['2ndFlrSF']
all_features['YrBltAndRemod'] = all_features['YearBuilt'] + all_features['YearRemodAdd']

all_features['Total_sqr_footage'] = (all_features['BsmtFinSF1'] + all_features['BsmtFinSF2'] +
                                     all_features['1stFlrSF'] + all_features['2ndFlrSF'])
all_features['Total_Bathrooms'] = (all_features['FullBath'] + (0.5 * all_features['HalfBath']) +
                                   all_features['BsmtFullBath'] + (0.5 * all_features['BsmtHalfBath']))
all_features['Total_porch_sf'] = (all_features['OpenPorchSF'] + all_features['3SsnPorch'] +
                                  all_features['EnclosedPorch'] + all_features['ScreenPorch'] +
                                  all_features['WoodDeckSF'])
```

7. Model

We use blending method for this regression problem, the estimation sale price of the house can be calculated from three different models. And that is …

Predicting sale price: 0.3 * xgboost model +
0.2 * random forest model +
0.5 * support vector regression model

Here is our configuration of our model:

(1) XGBoost Regressor

```
XGBRegressor(learning_rate=0.01,
             n_estimators=6000,
             max_depth=4,
             min_child_weight=0,
             gamma=0.6,
             subsample=0.7,
             colsample_bytree=0.7,
             objective='reg:squarederror',
             nthread=-1,
             scale_pos_weight=1,
             seed=27,
             reg_alpha=0.00006,
             random_state=42)
```

(2) Random Forest Regressor

```
RandomForestRegressor(n_estimators=1200,
                      max_depth=15,
                      min_samples_split=5,
                      min_samples_leaf=5,
                      max_features=None,
                      oob_score=True,
                      random_state=42)
```

(3) Support Vector Regressor

```
make_pipeline(RobustScaler(), SVR(C= 20, epsilon= 0.008, gamma=0.0003))
```

8. Results

We plot the score (RMSE) for those three models and our blended model.

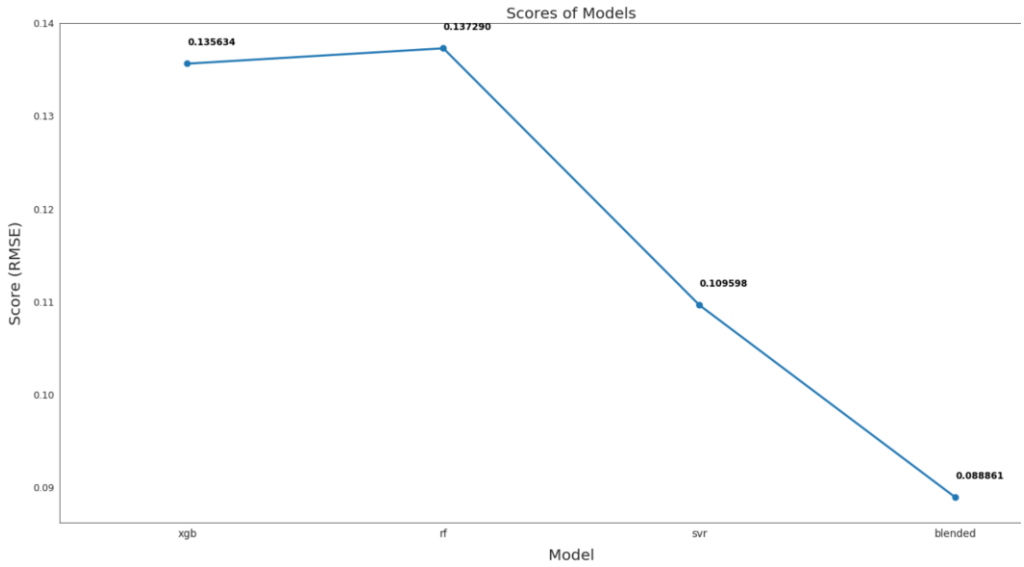
The result is:

XGBoost: 0.135634

Random Forest: 0.137290

SVR: 0.109598

Blended: 0.088861



The result shows that ensemble these three models can effectively reduce the root mean square error.

9. Reference

(1) https://www.kaggle.com/lavanyashukla01/how-i-made-top-0-3-on-a-kaggle-competition

(2) https://www.kaggle.com/marsggbo/kaggle#%E4%B8%80%E3%80%81%E6%98%8E%E7%A1%AE%E7%9B%AE%E7%9A%84

(3) https://www.kaggle.com/massquantity/all-you-need-is-pca-lb-0-11421-top-4

(4) https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

(5) http://studyai.site/2018/03/09/%E3%80%90%E7%BF%BB%E8%AF%91%E3%80%91kaggle%E7%AB%9E%E8%B5%9B%20%E6%88%BF%E4%BB%B7%E9%A2%84%E6%B5%8B/

Data Source：

https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview