Final Report

House Prices – Advanced Regression Techniques

Yiming Lai

Problem statement

House buying is a long and uncertain process especially for the first time buyers. How much should I place the bid so that I can get my dream house without overpaying is probably one of the top questions that comes to a home buyer's mind. In this project, we are given a dataset that describes almost every aspect of residential homes in Ames, Iowa, and our goal is to predict the house price based on those features and get a sense on what are the most critical features that determine the house price.

Data wrangling and cleaning

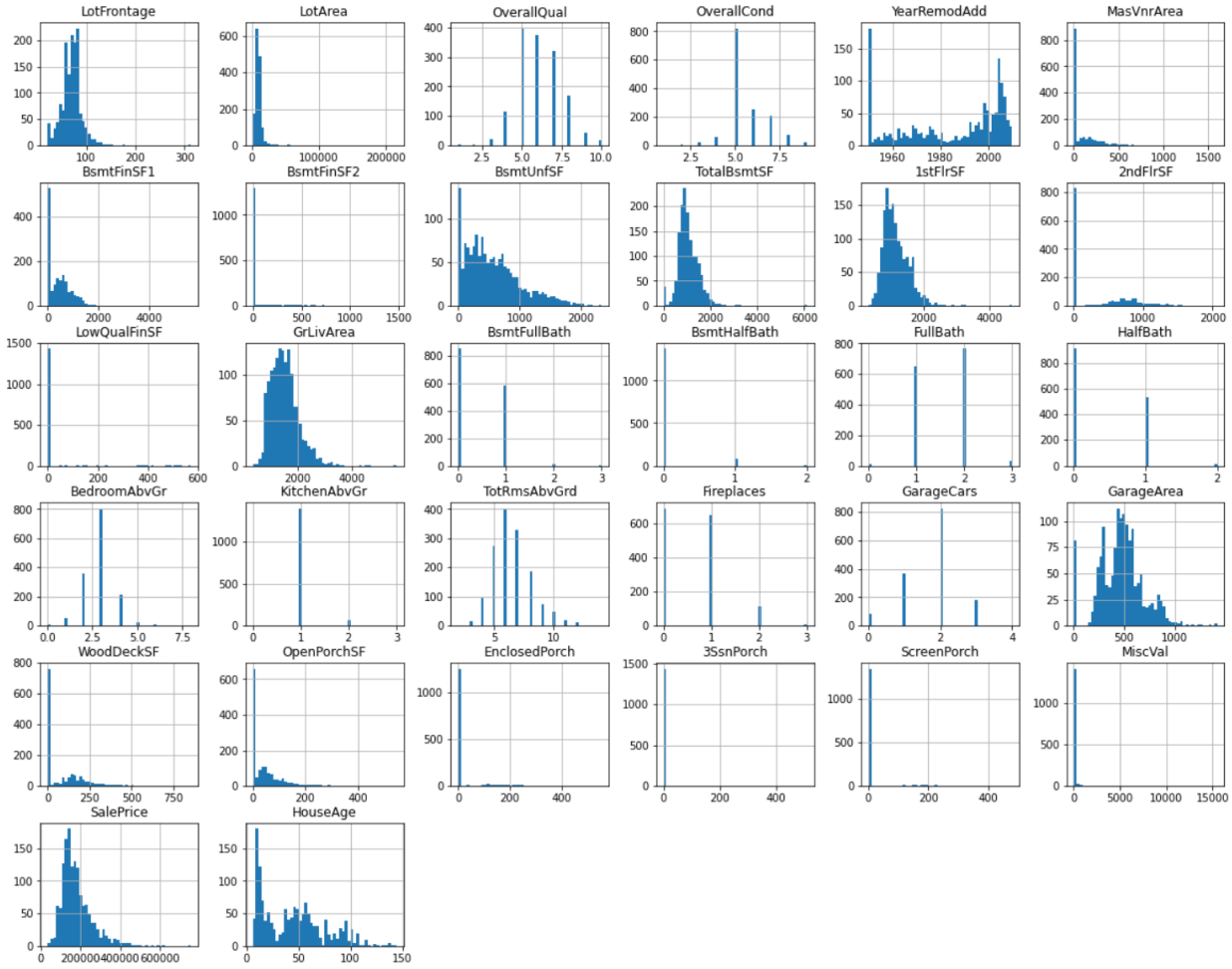
The dataset contains 79 features that include both numerical and categorical variables. There are 1460 data points in the training set and 1459 data points in the test set. We process the data as the following steps

- There is a clear fat finger error in the GarageYrBlt feature. 2207 should be replaced by 2007
- Remove features PoolQC, MiscFeature, PoolArea, Alley and Fence since more than 80% of data in those features are missing. We cannot get too much information from those features
- Some missing values actually mean that specific house doesn't have such categorical feature, such as FireplaceQu, GarageFinish etc. We replace those missing values with None
- Some missing values actually mean that specific house doesn't have such numerical feature, such as FireplaceQu, GarageFinish etc. We replace those missing values with 0
- Replace the missing values in LotFrontage by the mean value in the specific neighborhood the house belongs to. Note that here we want to use the mean from the training set only in order to prevent data leakage
- For the rest of the missing values, replace them with the most common value in the corresponding neighborhood
- YearBuilt doesn't make too much sense when building the model. Convert it to a new feature HouseAge

The final dataset has total 2905 data points and 1446 of them are the training data

EDA and feature engineering

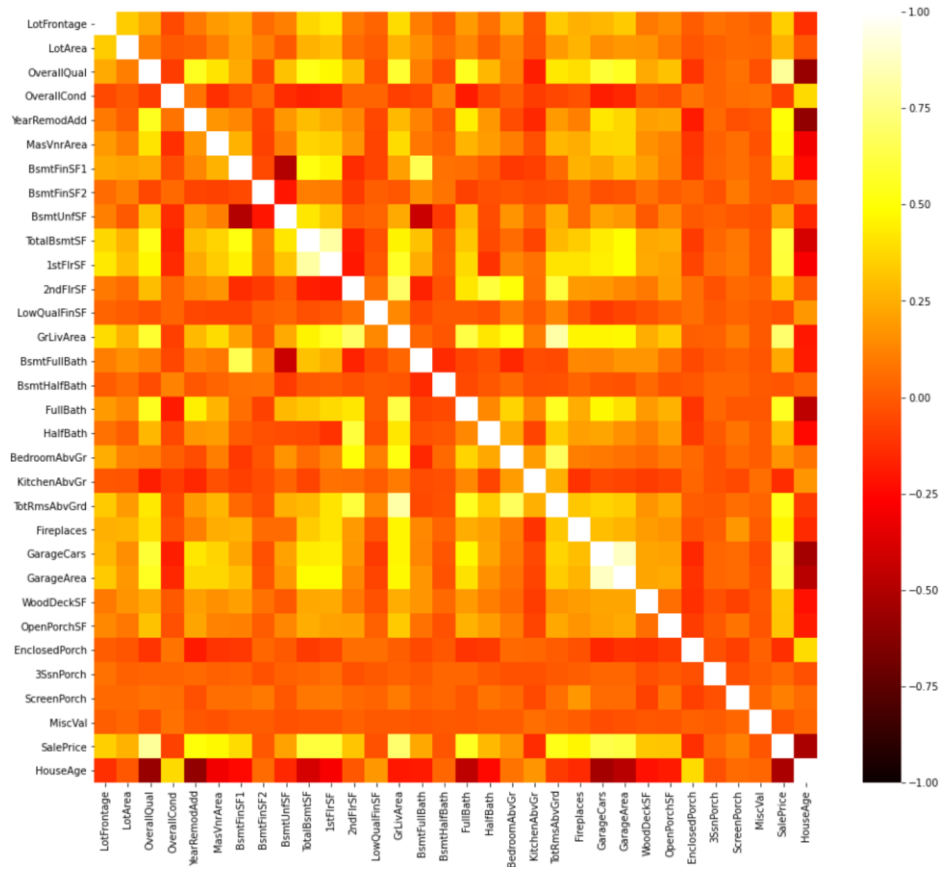
As the first step we can plot the histogram for all the features to check the data distribution as shown below



From the histogram, we can see that some features are skewed and we may need to do a log-transformation later.

Since there are many different features in this dataset, we want to perform feature selection to keep only the important features for the subsequent modeling stage in order to avoid the overfitting and the curse of dimensionality problem. Here we deal with the numerical variables and the categorical variables separately.

For numerical variables, we first plot the Pearson correlation map below. We can see that some features are correlated with each others (correlation > 0.8). In particular, TotalBsmtSF is highly correlated to 1stFlrSF, GarageCars is highly correlated the GarageArea. This makes sense since the area of the basement is usually similar to the area of the 1st floor, and larger the garage, more cars can be parked inside the garage. We drop 1stFlrSF and GarageArea since they have slightly less correlation to the SalePrice.

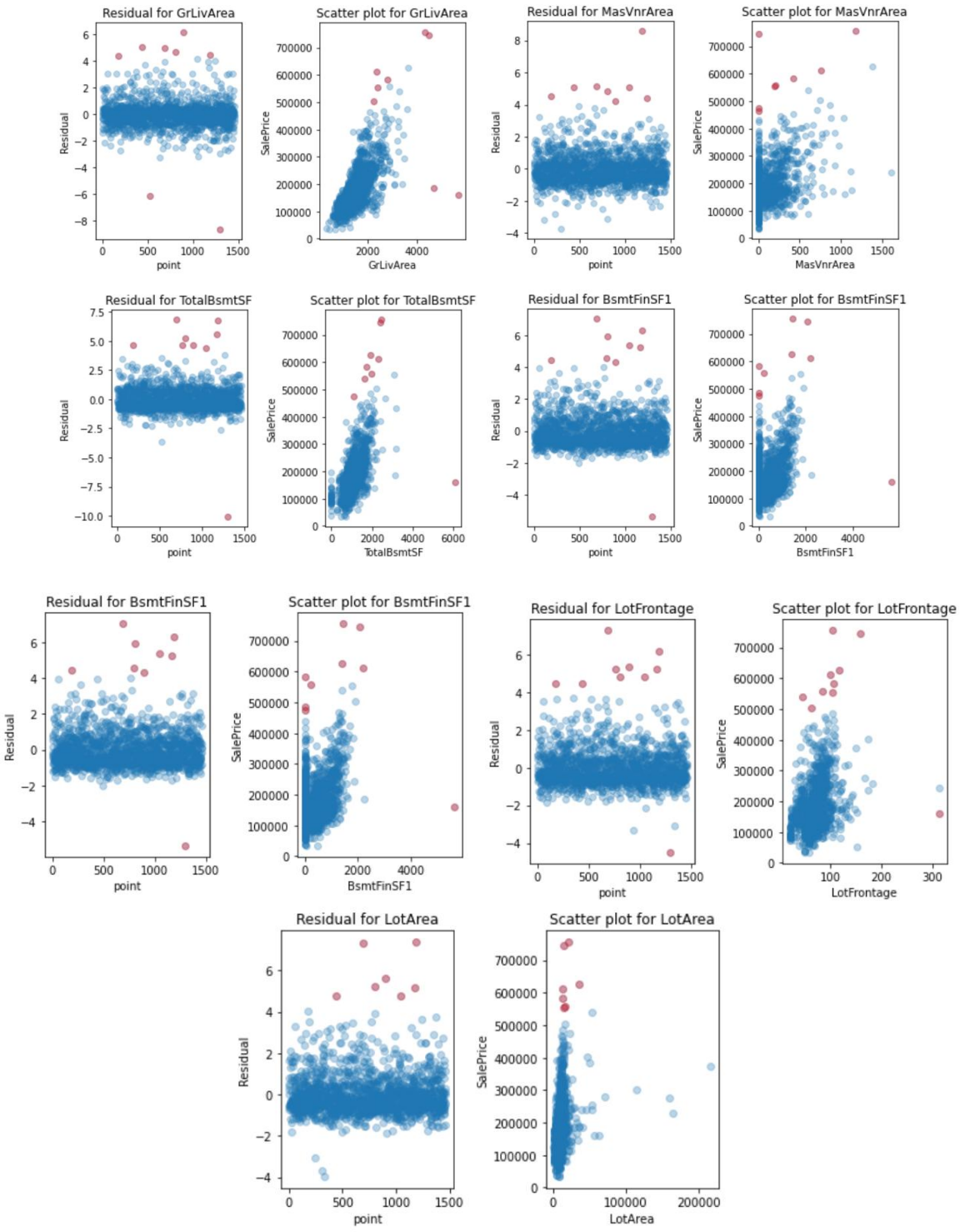


Furthermore, for the modeling later, we only keep the features that have > 0.1 correlation with the SalePrice.

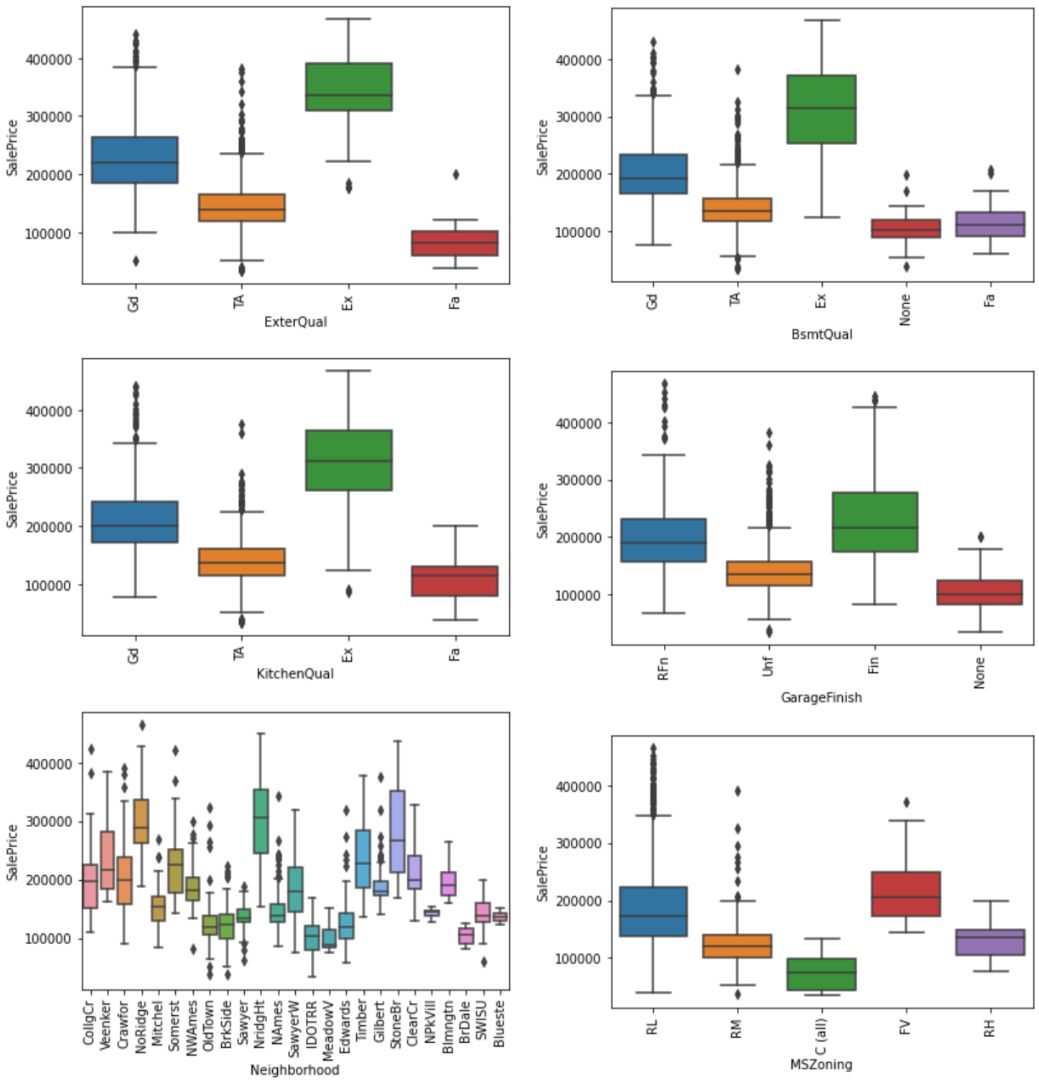
As for the categorical variables, we use Chi-Squared test to check the correlation between the features and our target value, and drop the features that have p-values > 0.05. In order to perform the Chi-Squared test, we divide the SalePrice into 5 different price ranges as the following: ‘very low’, ‘low’, ‘medium’, ‘high’, ‘very high’. Total of 5 categorical variables (Street, Utilities, LandSlope, MoSold, YrSold) were dropped from the Chi-Squared test.

Now we can take a closer look at the dataset. From the from scatter plot between features and the SalePrice, we notice that we have outliers in our data set. To remove the outliers, the approach we use here is to use a simple linear regression model and calculate the studentized residuals for the features and remove the data points that have the corrected p-values less than 0.05. The scatter plot and the corresponding residual plot are shown below. The red dots are the identified outliers that will be removed.

As we can see from the scatter plot that essentially the SalePrice is proportional to all the features that are related to the area. This agrees with our intuition. Bigger the house, higher the SalePrice!



Some of the categorical variables are also plotted below



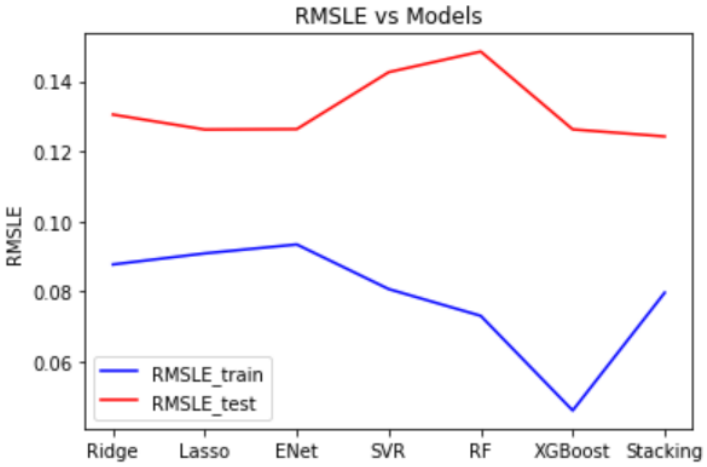
By looking at the ExterQual, KitchenQual and BsmtQual we can see that in general better the quality, higher the SalePrice (Ex > Gd > Ta > Fa), which agrees with our intuition. SalePrice also depends on the neighborhood (pysical locations within the city), this may be correlated to the many different factors such as the safety index, nearby schools, and walking scores, etc. For the GarageFinish, we can also see that the house with interior finished garage usually has higher SalePrice. For the MSZoning, it appears that the SalePrice is lower for the house located in the commercial region.

Modeling

As mentioned above, we perform the log-transformation to convert the SalePrice and the features. This helps us to de-emphasize the outliers and make the data close to normal distribution. The final step we take before start modeling our dataset is to one-hot-encode the categorical variables that we can use them in the machine learning models.

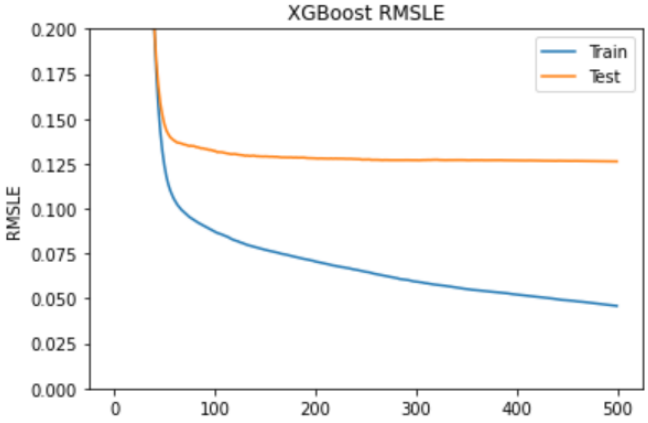
The models we choose are the Ridge regressor, Lasso regressor, ElasticNet, Support Vector regressor, Random Forest and XGBoost. We use either GridSearchCV or RandomizedSearchCV in order to find the best performing model based on the CV score and its corresponding hyperparatmers. The CV score is calculated by the RMSE (or more precisely the RMSLE since our SalePrice is log-transformed). We also build a stacking regressor based on all the models above to see if we can further improve the model performance. After training, we get the RMSLE scores for both training set and the test set for each model and the result is summarized below.

```
RMSLE(train) for Ridge: 0.08763077541934929
RMSLE(test) for Ridge: 0.1304877448414943

RMSLE(train) for Lasso: 0.09079906675875786
RMSLE(test) for Lasso: 0.12620327608023368

RMSLE(train) for ENet: 0.09334957829123673
RMSLE(test) for ENet: 0.1263293883413303

RMSLE(train) for SVR: 0.08058579443635039
RMSLE(test) for SVR: 0.1426133603346932

RMSLE(train) for RF: 0.07294982174601274
RMSLE(test) for RF: 0.14850367527465005

RMSLE(train) for XGBoost: 0.04587235763058956
RMSLE(test) for XGBoost: 0.12624786588997225

RMSLE(train) for Stacking: 0.07957217338030946
RMSLE(test) for Stacking: 0.12423559022005141
```

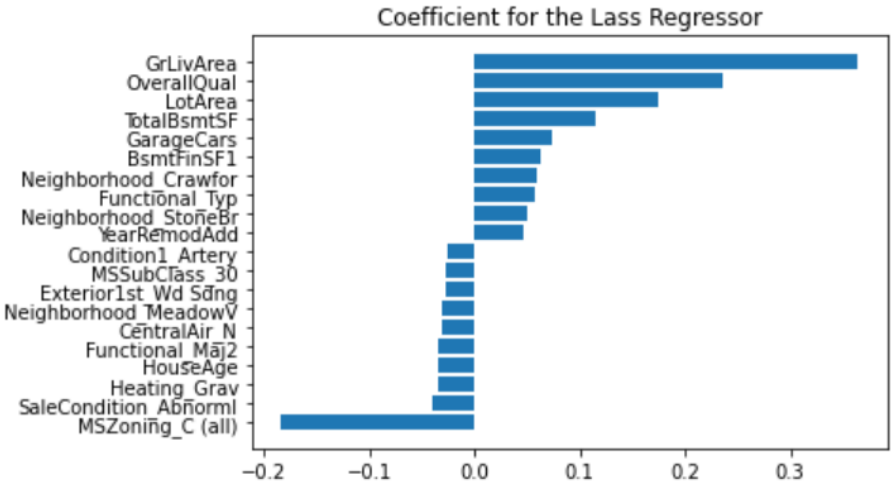


Summary

As we can see from the above figure, the stacking model gives us the best RMSLE score for the test data set and this is the model we choose for the final prediction. We also notice that we have the overfitting problem for the Random Forest and the XGBoost because there is a big gap between the RMSLE score for the training set and RMSLE score for the test set. Figure below shows the RMSLE vs number of iteration for the XGBoost model. We can see that score for the test already saturates after about 100 iterations while the score for the training set keeps going down. This is a typical indicator for the overfitting.



The figure below shows the top 10 important features we got from the Lasso regressor. As we expected before, larger the house, higher the SalePrice as 4 out of the top 5 positive contributors to the SalePrice are related to the size of the house. Besides the size, better the overall quality, higher the SalePrice, which can be expected as well. As for the negative contributors, the house age actually doesn't downgrade the value of the house too much. The most negative impact to the SalePrice is the location of the house. The SalePrice tends to be lower if the house is located in the commercial region.



Future work

To further improve the accuracy of our prediction, we may try the following approaches

- Outliers detection: here we use a simple Linear model for the outliers detection, we may try to use more complex models to filter out the outliers
- As discussed above, we may have overfitting problems with our models. For the tree-based model such as XGBoost, we may implement the early-stop hyperparameter to prevent overfitting. We can also try to further decrease the number of the features in our model to prevent overfitting, such as using PCA to keep only the most important PCA components, or we can simply discard the features that have 0 importance in the Lasso regressor.