1 Data Processing

I first deal with the Missing Values. For numerical missing values, i use median for LotFrontage and 0 for MasVnrArea which are the most frequent element. And i use 1900 for GarageYrBlt as 1900 is the minimal value of the Years, and i make the plots which indicates the linearship between garageYrBlt and salesprice.For categorical columns, Electrical is the only columns with one missing values, i will use the mode 'SBrkr' as substitution.

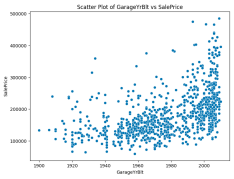

Figure 1: GarageYrBlt and SalePrice

Then deal with numerical columns. As theres numerical columns like years and month, which might not have strong linear relationship with the saleprice, so i plot them against saleprice and leave some of them numerical but encode the other.

For categorical columns, i will do one-hot encoding on most columns except for those duplicate ones, for instance conditon1 and condition2. Also, for those quality columns, the response will be transformed into integers by ordinal encoding as linear relationship exists in them columns. For example, 'Ex' = 5, 'Gd' = 4.

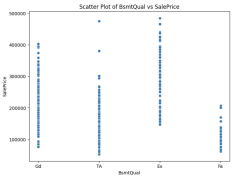

Figure 2: BsmtQual and SalePrice

For those special cases, i will create separate columns for all responses and fill the corresponding columns with 1 and 0 otherwise. for example, if condition1 = 'A', condition2 = 'B',and the data space = A,B,C, then the fixed data will become condition_A = 1, condition_B = 1, condition_C = 0.But for BsmtFintype, i use the value of BsmtFinSF instead of a 1.

Finally, I finished data transformation, converting all features to numerical, resulting in 270 columns in total.

2 Modeling and Model Tuning

To find a better model, the core part is parameter selection. To choose the best combination of parameters, I used standardization to transform all parameters into a Gaussian distribution for comparison. Then i split the training data

to train_data and test_data, where i first train data on train_data and test on test_data, then, with the choosing parameters, do the modelling again on the whole training data to get the final model.

The first model I used was linear regression, but the predicted y had a large gap from the true y with score = 4651957683133456.

Forward selection was better in computing time compared to best subset selection. I ran the function with AIC penalty to pick out the necessary parameters, fitted the model with the whole training data, and then tested it. The prediction scored 24415.

$$\underset{\beta}{\text{minimize}} \left(\sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} X_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right) \tag{1}$$

Lasso regression, evaluated with a loss function consisting of test error and penalty controlled by tuning parameter lambda. The loss function is now a trade-off between bias and variance. I used `lasso.cv` from `sklearn.linear_model` for tuning parameter selection. The final score was 22924 with best lambda selected is 1239.

3 Overfitting and Underfitting

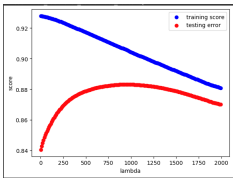

Figure 3: Lambda and Training Error and Testing Error

But i also tried to code the best lambda myself, and i loop through lambda from 1 to 2000.And i found the best lambda located at 980.With the graph shown, we can see that when lambda is small, the penalty is less impactive, the variance tends to be higher, the test error is high but training error is low, which is overfitting. But when lambda is approaching high values, the penalty term is given more weight then RSS, which reduce the variability but increase the bias. Underfitting is caused due to the lack of variability.

4 Final Model

I finally choose the lambda 980, and run the model again on the whole training data for the final model. This model gave a score of 22698, a bit better than the `lassoCV` score.