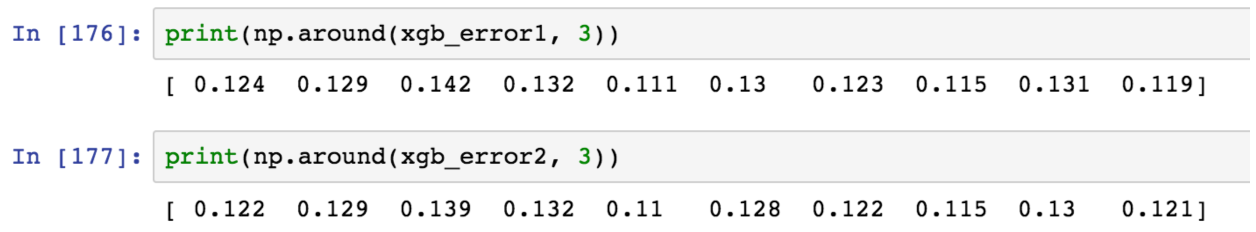
STAT542 Project1
Tianqi Wu
2018/10/17

Pre-Processing

First, following nominal variables are dropped because of dominant level: 'Street', 'Utilities', 'Land_Slope', 'Condition_2', 'Roof_Matl', 'Heating', 'Pool_QC', 'Misc_Feature'. Following numeric variables are dropped since they either contain most 0 entries or not correlated to response variable at all: 'Low_Qual_Fin_SF', 'Three_season_porch' , 'Pool_Area', 'Misc_Val'. 'Longitude' and 'Latitude' are also dropped according to the suggestion. There is only one variable 'Garage_Yr_Blt' containing missing value and imputation strategy of median is used first. However, 'Garage_Yr_Blt' is still dropped since there is no effect on the model performance after removing it. Bins are created for date variables like 'Year_Built' and cut-off points are decided by building decision tree model. However, it turns out treating them as continuous variables maximizes the model performance.
Then, the frequency of nominal variables are calculated. Infrequent levels are merged to 'other' and the optimal cut-off value is 0.01 of the total observations. Also, one hot encoding is done to those nominal variables. Winsorization is done to to the numeric variables in one direction. It takes values exceeding 95th quantile to be equal to that quantile. Then, some ordinal variables like 'Overall_Qual' are relabeled. For example, values of very excellent, excellent and very good are labeled as 4 and values of good and above_average are labeled as 3. A dictionary is created to map these ordinal variables.
Finally, correlations of the variables to 'Sale_Price' are calculated and variables with correlation lower than 0.1 are dropped. Other variable selection methods like AIC are tried but they do not give desired improvement.

Model

The two prediction models used in the project are from the same XGboost algorithm with two different sets of parameter. Package xgboost from python was used for modeling, which may need installation before running. The first one used most default parameters with not much tuning. For the second set of tuning parameter, grid search was used to find out the optimal ones. Max_depth and min_child weight were further tuned since they have the highest impact on the



```
In [176]: print(np.around(xgb_error1, 3))
          [ 0.124  0.129  0.142  0.132  0.111  0.13   0.123  0.115  0.131  0.119]

In [177]: print(np.around(xgb_error2, 3))
          [ 0.122  0.129  0.139  0.132  0.11   0.128  0.122  0.115  0.13   0.121]
```

model outcome. My performance on the 10 splits are the following:

The models produce RMSE under 0.132 except fold 3. The running time of whole script is 76 seconds and the computer system is Macbook Pro 3.1GHz, 8GB memory.