House Price Prediction
Kaggle Competition

Chelsea Li
George Washington University
2022.01

Table of Contents

1.Background

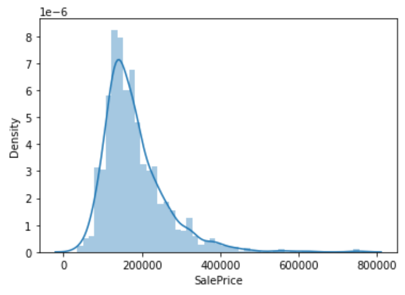
1.1 Data description

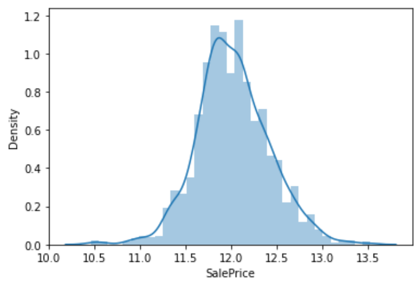
This Data set is from Kaggle competition(https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview/evaluation) about house price prediction. It contains three files. Train.csv, test.csv and data description.txt.

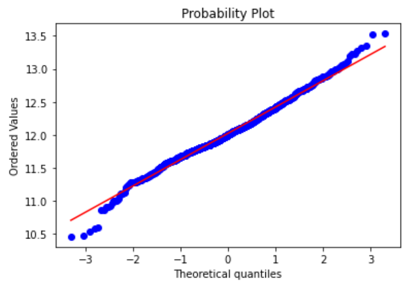
In the train data set, 1460 instances and 80 independent features with one target, house price. In the test data set, 1459 instances and 80 independent features. In the data description file, it explains the meaning of 80 independent features in detail.

1.2 Target variable

The target variable is house price. The mean of the SalePrice is 180921, min is 34900 and max is 755000. We can also see from the histogram that the SalePrice is positive skewed. Log transformation can change it to normal distributed.







1.3 Independent variables

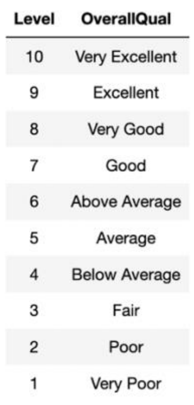
80 features about the hourse condisions were collected. 43 categorical variables and 37 numerical variables. Based on personal experience, four variables were chose which may play important roles when people plan to buy a house. They are OverallQual, YearBuilt,TotalBsmtSF, and GrLivArea.

As for YearBuilt, it means the original construction date.

As for TotalBsmtSF, it means the total square feet of basement area.

As for GrLivArea, it means the above grade (ground) living area square feet.

As for OverallQual, it rates the overall material and finish of the house and has ten level of it.



Level	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

2. Data Preprocessing

2.1 General preprocessing (Missing data)

Let’s dive deep into missing values and fill them correctly. Missing ratio was checked first. The principle of making up missing value is to find the basic logic of data. Dose the Null value mean missing or zero? What value should be used to fill? First, when the percent of missing value is over 80% it is hard to use other 20% to present these 80%. Sometimes, there features don't have a lot of contribution to our models. We can choose to delete them. As in our dataset, we can find four features which missing value percent over 80% (PoolQc,MiscFeature,Alley,Fence). However, the null meaning of them is zero. So, we fill 'None' in them instead of deleting them directly.

Missing percent that is around 50% needs our attention most because they can have a very big influence of our dataset. In our train data set, FireplaceQu NA has 48% of missing percent. means no fireplace in this house so we also use 'None' to fill.

LotFrontage is the space connected with neighborhood, so we use the median number of their neighborhood to fill missing value.

As for other missing values whose percent are lower even than 5%. To be honest, we can just drop this instance when there are a lot of data or we can use mean, mode, ‘None’ to fill them because they will not really influence our models.

In the process, train dataset and test dataset were filled separately with their median or mode number in order to reduce the influence of each other.

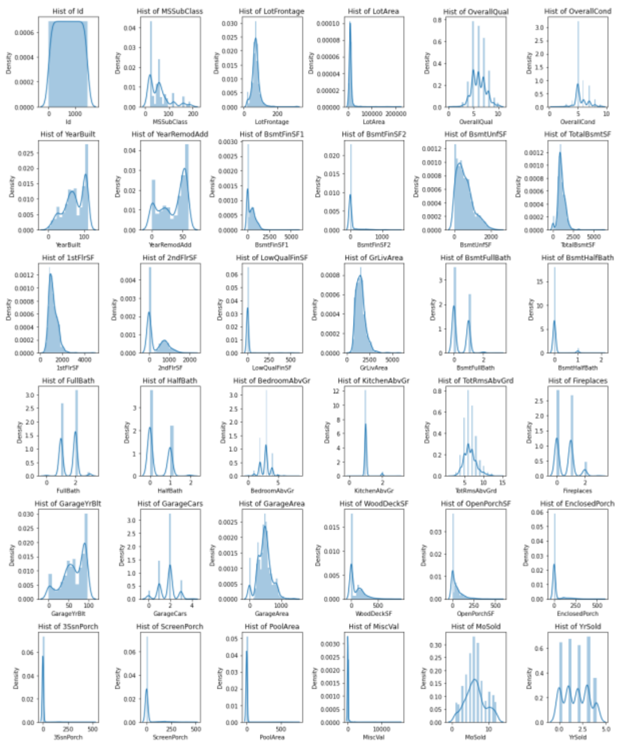
A column named Utilities was drop.

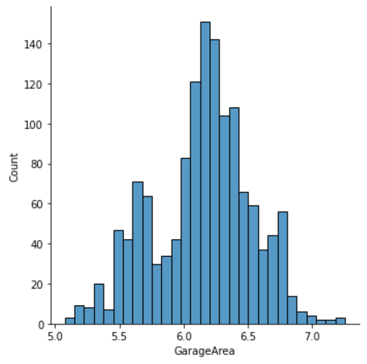
2.2 Categorical Features

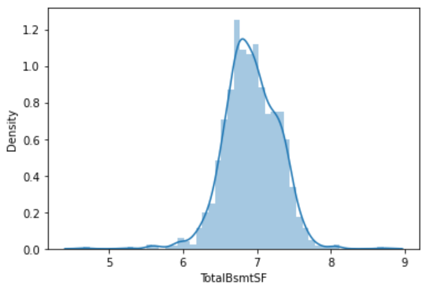
43 categorical variables and four numerical variables about year. They are YearBuilt, YearRemodAdd, GarageYrBlt, YrSold. We also need to encode these years together with categorical variables. LabelEncoder was used to encode categorical variables on train data set and test data set separately.

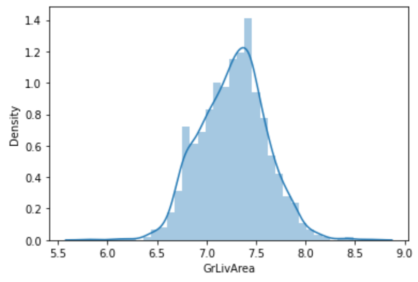
2.3 Numerical Features

From the histogram, BsmtUnfSF, TotalBsmtSF, GrLivArea, GarageArea are not normal distributed. Log transformation will work.



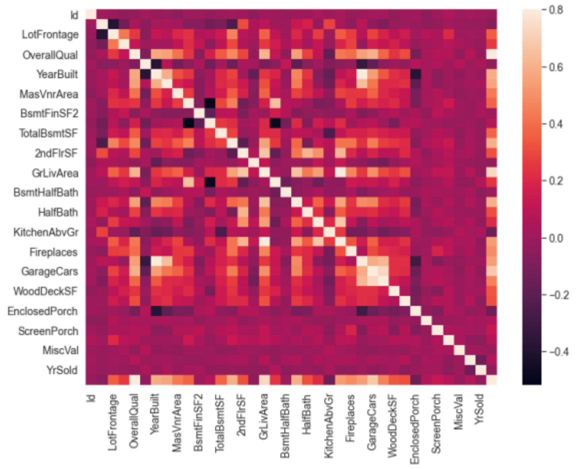


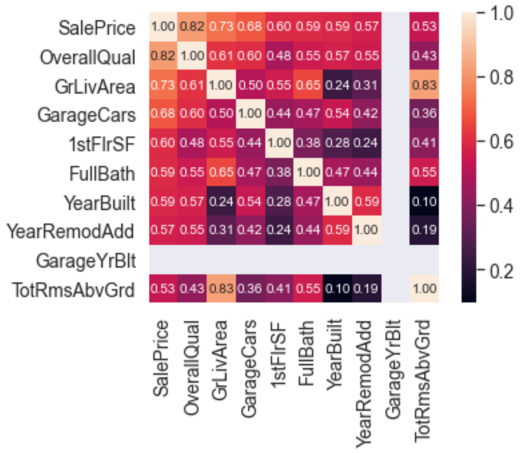




3. Exploratory Data Analysis

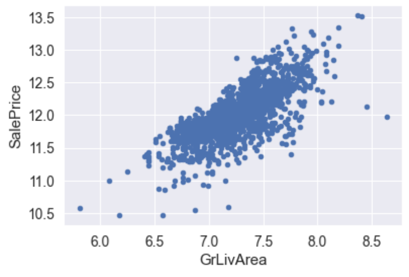
Heatmap was plotted to explore the relationship between each features. We can find some features are significant shown by white block. Such as GarageCars and WoodDeckSF, Fireplaces and YearBuilt. The collinearity will influence linear model much. Regarding random forest model will be used, we think this may not have much influence.

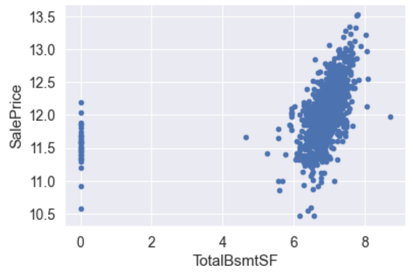


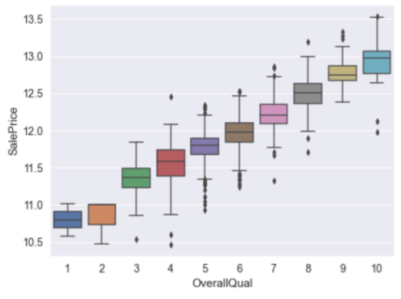


Look closer, we can see OverallQual, GrLivArea are strongly related with saleprice. What's more, GarageArea and GarageCars are highly related which makes sense. TotalBsmtSF and 1stFLrSF are highly ralated. These variables are collinearity.

Then, we are going to explore the relationship on SalePrice with GrlivArea,GrlivArea and OverallQual.





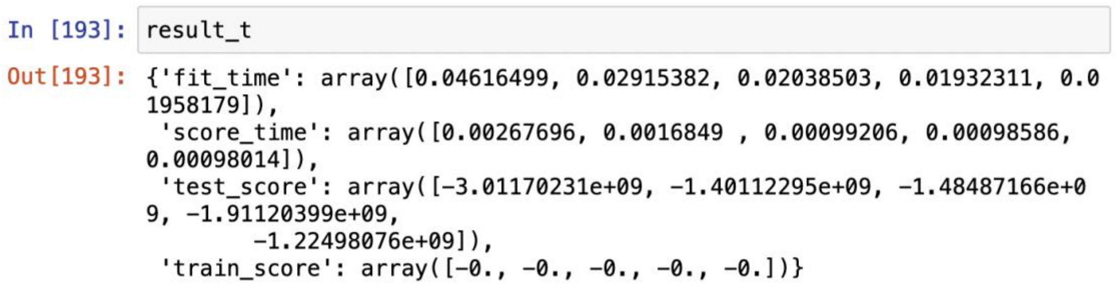


The plots show a positive relationship. The x-axis increases a sale price becomes higher.

4. Modeling

4.1 Decision Tree

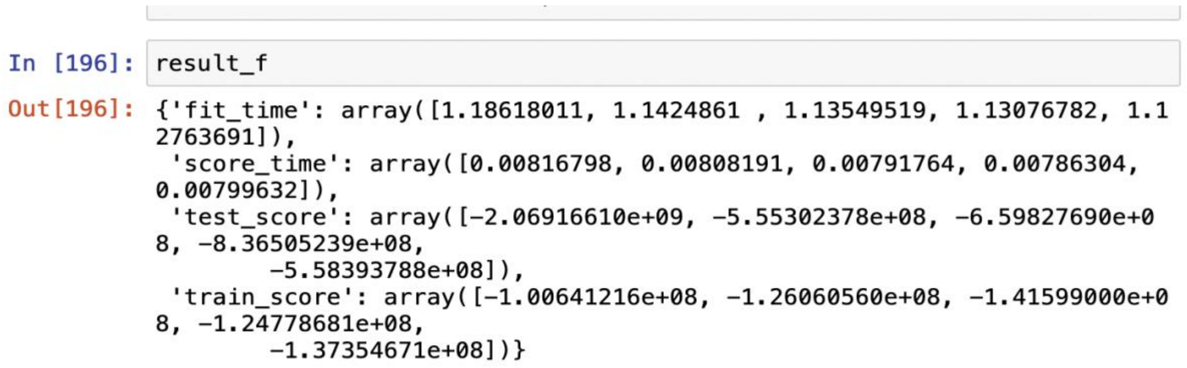
High score on train dataset while not good score on test dataset so decision tree model is overfitted based on default parameters.



```
In [193]: result_t
Out[193]: {'fit_time': array([0.04616499, 0.02915382, 0.02038503, 0.01932311, 0.0
          1958179]),
           'score_time': array([0.00267696, 0.0016849 , 0.00099206, 0.00098586,
          0.00098014]),
           'test_score': array([-3.01170231e+09, -1.40112295e+09, -1.48487166e+0
          9, -1.91120399e+09,
                  -1.22498076e+09]),
           'train_score': array([-0., -0., -0., -0., -0.])}
```

4.2 Random Forest
Step 1: Build Benchmark

Random forest performs better on test dataset and is not overfitted as decision tree. Default parameters were used to build model.

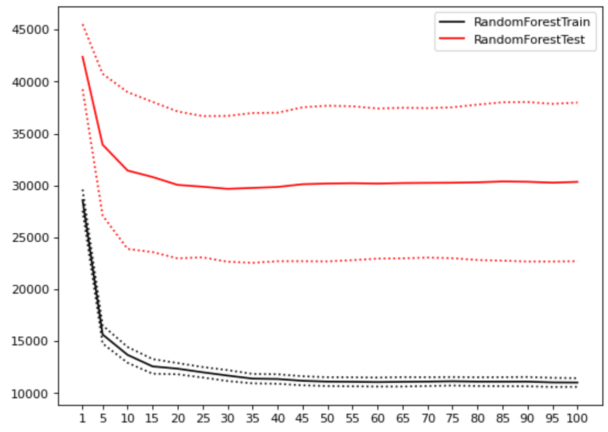


```
In [196]: result_f
Out[196]: {'fit_time': array([1.18618011, 1.1424861 , 1.13549519, 1.13076782, 1.1
          2763691]),
           'score_time': array([0.00816798, 0.00808191, 0.00791764, 0.00786304,
          0.00799632]),
           'test_score': array([-2.06916610e+09, -5.55302378e+08, -6.59827690e+0
          8, -8.36505239e+08,
                  -5.58393788e+08]),
           'train_score': array([-1.00641216e+08, -1.26060560e+08, -1.41599000e+0
          8, -1.24778681e+08,
                  -1.37354671e+08])}
```

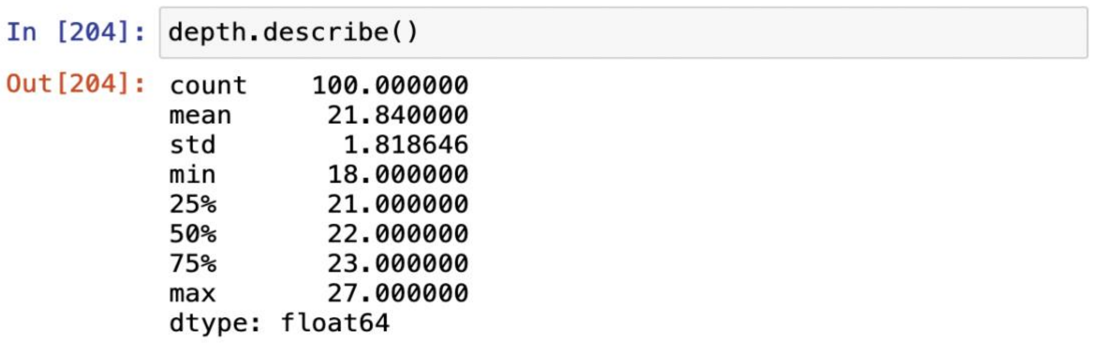
Step 2: Find parameter space

Study curves and .tree_ attribute were used to find parameter space.

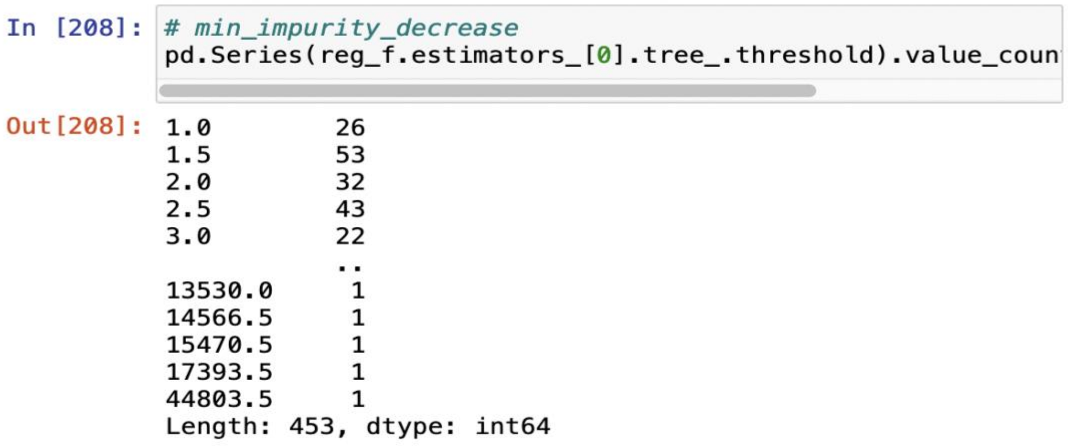
A study curve on n_estimators , we set n_estimators to range(20,100,5), more number of trees in order to offer more trees when building simpler trees with other parameters.



We set max_depth around 10-25 and min_impurity_decrease to [2,10] to reduce to divide more leaves.



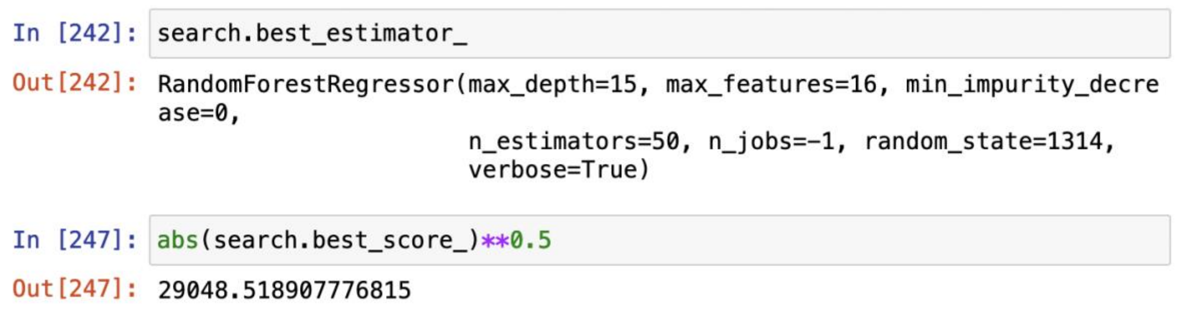
```
In [204]: depth.describe()

Out[204]: count    100.000000
          mean      21.840000
          std        1.818646
          min       18.000000
          25%       21.000000
          50%       22.000000
          75%       23.000000
          max       27.000000
          dtype: float64
```



```
In [208]: # min_impurity_decrease
          pd.Series(reg_f.estimators_[0].tree_.threshold).value_coun

Out[208]: 1.0        26
          1.5        53
          2.0        32
          2.5        43
          3.0        22
                     ..
          13530.0     1
          14566.5     1
          15470.5     1
          17393.5     1
          44803.5     1
          Length: 453, dtype: int64
```

4.3 Hyperparameter Tunning I

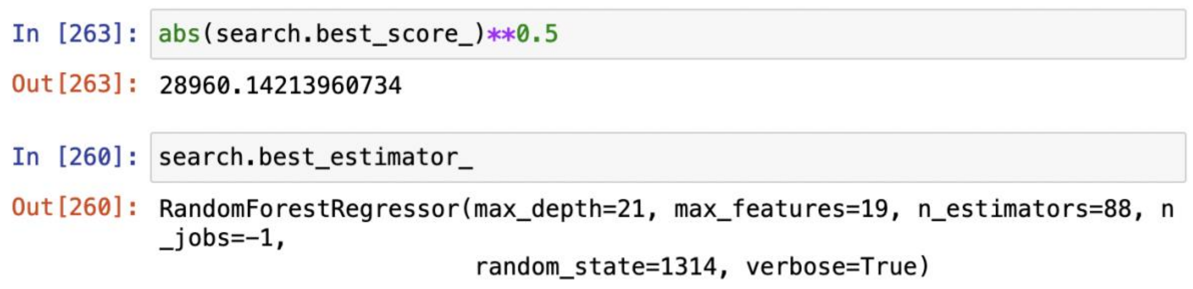
First randomized search was used based on the basic parameter space. The RMSE is around 29048.519 Then based on the best parameter result, I reset the parameter space to smaller range but dense area which is better for randomized search performance.



```
In [242]: search.best_estimator_

Out[242]: RandomForestRegressor(max_depth=15, max_features=16, min_impurity_decre
          ase=0,
                                n_estimators=50, n_jobs=-1, random_state=1314,
                                verbose=True)

In [247]: abs(search.best_score_)**0.5

Out[247]: 29048.518907776815
```

4.4 Hyperparameter Tunning II

I set n_estimators from 50 to 100 step 2. Max_depth from 10 to 25 step 1. Max_features from 10 to 20 step 1. It helps me to find a better RMSE.



```
In [263]: abs(search.best_score_)**0.5

Out[263]: 28960.14213960734

In [260]: search.best_estimator_

Out[260]: RandomForestRegressor(max_depth=21, max_features=19, n_estimators=88, n
          _jobs=-1,
                                random_state=1314, verbose=True)
```

4.5 Retrain and Test

Finally, this list of parameters was used to build a model with train dataset. Five-fold cross validate showed that RMSE on train dataset is 28332.089.

5. House Price Prediction

Test dataset was used to predict house price on the model built above and the result was uploaded to Kaggle competition.