HOUSE PRICE PREDICTION

Capstone Project – Model Building & Tuning

MANISHA PARAKANDLA

6/12/22

PGP DSBA GREATLEARNING ONLINE

Business Problem:

A house value is simply more than location and square footage. Like the features that make up a person, an educated party would want to know all aspects that give a house its value. For example, you want to sell a house and you don't know the price which you may expect it can'tbe too low or too high. To find house price you usually try to find similar properties in your neighborhood and based on gathered data you will try to assess your house price.

Contents:

List of Figures:

List of Tables:

Introduction- Problem Understanding

Defining problem statement:

Whenever any individual/business wants to sell or buy a house, they generally face this kind of issue as they don't have clear understanding on the price which they should offer. Due to this there is a chance that they might offer too low or high price for the property. Hence, we can analyse the available data of the properties/houses in the area and can predict the price. We need to find out how these attributes/features influence the house prices. Right pricing is very important aspect for selling the house. It is very important to understand that what are the factors and how they are influencing the house price.

Need of the study/project:

we need to predict the right price of the house based on the study results from attributes/features and result of the models performed from the dataset. To find out the important aspects that reflect on the sale price of the houses.

Understanding business/social opportunity:

Many people don't know the features/aspects which accumulate property price, we can provide them House Buying & Selling guidance services in the area so they can buy or sell their property with most suitable price tag and they won't lose their money by offering low price or keep waiting for the buyers by putting high prices.

Objective

Our objective is to Build a model which will predict the house price and its relevant features that manipulate the sale price when selected features based on analysis are passed into to the model.

- we need to analyse the data and find out the significant/Important features from the given features dataset which affects the house price the most.
- **Build best feasible model to predict the house price with 95% confidence level.**

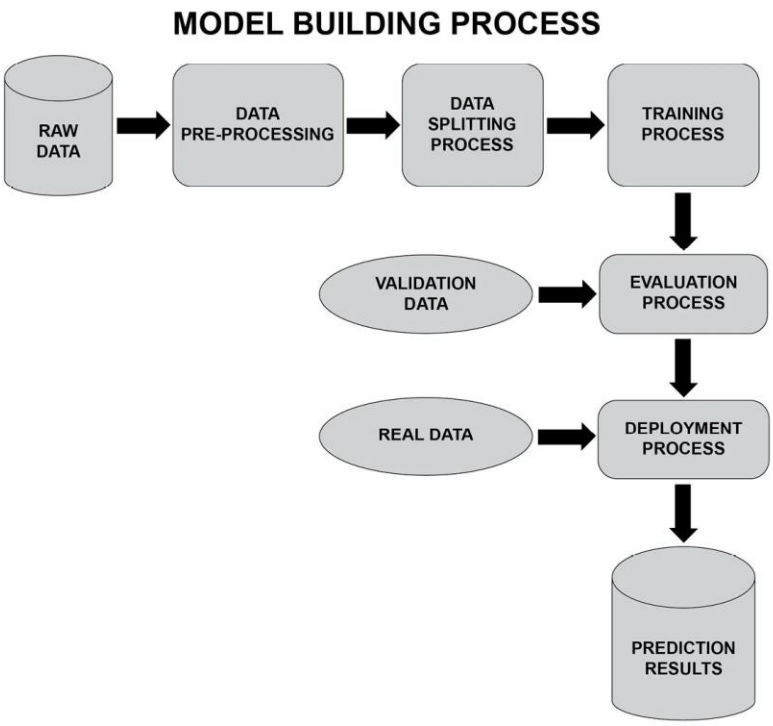


Figure 1 Model Building Process

Problem & Procedure:

1. Model building and interpretation.

a. Build various models (You can choose to build models for either or all of descriptive, predictive or prescriptive purposes)

b. Test your predictive model against the test set using various appropriate performance metrics

c. Interpretation of the model(s)

2. Model Tuning

a. Ensemble modelling, wherever applicable

b. Any other model tuning measures (if applicable)

c. Interpretation of the most optimum model and its implication on the business

We have already collected the data and pre-processed it to be ready for Model Building.

Splitting the data:

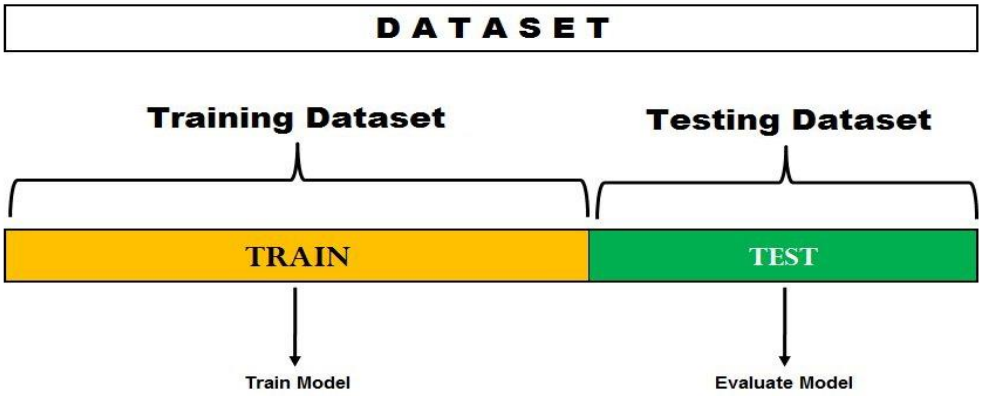


Figure 2 Train - Test Split

I choose to split the data t0 80:20 ratio.

Shape of X_train: (17290, 51)
Shape of X_test: (4323, 51)
Shape of y_train: (17290,)
Shape of y_test: (4323,)

Building Regression Models:

In regression analysis, model building is **the process of developing a probabilistic model that best describes the relationship between the dependent and independent variables**.

Regression analysis is done for one of two purposes:

1. To predict the value of the dependent variable for individuals for whom some information concerning the explanatory variables is available.
2. To estimate the effect of some explanatory variable on the dependent variable.

As a part of Model building, I have performed the models mentioned below:

1. Linear Regression Model
2. Ridge Model
3. Losso Model
4. Knn Model
5. Support Vector Regressor Model
6. Decision Tree Model
7. Ensemble Models (Bagging, Boosting)
8. Random Forest Model

The results of the models performed are stored in a data frame for comparison.

Method	Test Score	RMSE_test	MSE_test	MAE_test	Train Score	RMSE_train	MSE_train	MAE_train
Linear Regression Model	0.703668	135222.127869	1.828502e+10	105017.960441	0.682771	141050.005422	1.989510e+10	105017.960441
Ridge Model	0.703519	135256.091125	1.829421e+10	105033.966057	0.682736	141057.679352	1.989727e+10	108024.827496
Losso Model	0.682766	135238.694653	1.828950e+10	105030.625285	0.682766	141051.069227	1.989540e+10	108012.514007
knn Model	0.535146	169362.387969	2.868362e+10	128384.020547	0.998637	9244.836127	8.546700e+07	820.155986
Support Vector Regressor Model	-0.050712	254624.623389	6.483370e+10	191796.332853	-0.063240	258227.332216	6.668136e+10	193710.374217
Support Vector Regressor(Modified)	0.586507	159732.258253	2.551439e+10	124511.552092	0.571431	163944.525033	2.687781e+10	128075.700630
Dession Tree -1	0.483369	178545.397695	3.187846e+10	130580.327550	0.998637	9244.836127	8.546700e+07	820.155986
Decision Tree -2	0.679168	140700.982305	1.979677e+10	105796.746970	0.771146	119802.428860	1.435262e+10	90712.091345
Gradient Boosting Regressor	0.742221	126119.532265	1.590614e+10	97839.489718	0.749858	125250.678917	1.568773e+10	97839.489718
Bagging Regressor	0.745086	125416.558141	1.572931e+10	95504.808945	0.960147	49994.049363	2.499405e+09	36654.383026
Random Forest	0.743763	125741.711023	1.581098e+10	95482.119344	0.961959	48844.171797	2.385753e+09	36100.840827

Table 1 Model performance table for Train-Test data

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are. RMSE is a measure of how spread out these residuals are. In other words, it tells us how concentrated the data is around the line of best fit.

The Mean Squared Error (MSE) is a measure of how close a fitted line is to data points. For every data point, we take the distance vertically from the point to the corresponding y value on the curve fit (the error), and square the value. MSE is the average of the square of the errors. The larger the number the larger the error.

MAE is the Mean of Absolute value of Errors. Here, errors are the differences between the predicted values (the values predicted by our regression model) and the actual values of a variable.

From the above Train-Test performance table, we can observe that:

The **Linear regression model** performed with scores 0.68 & 0.70 in Train data set and Test data set.

The **Ridge model** also performed with scores 0.68 & 0.70 in Train data set and Test data set as linear regression model.

The **Losso model** performed with scores 0.68 & 0.68 in Train data set and Test data set

Though **KNN regressor** performed well in training set, the performance score in Test data set is very less. This shows that the model is overfitted in training set.

The above negative scores in **SVR model** are due to non-learning of the model in the training set which results in non-performance in Test data set. Later, I have modified the parameters and performed the model again. After performing the SVR model with modified parameters, it has not performed well with just ~0.57 scores in both training and test data sets.

Above performance of first **Decision Tree model** shows overfit in training set with 0.99 score and low performance in test set. Decision tree model with modified parameters has better performed on the training set and test set compared to initial decision tree model. But overall decision tree has not performed well compared to linear regression models.

From the model performance table, we can observe that, **KNN regressor model, SVR** and **decision tree models** have not performed well in comparison with **linear regression models**.

Let us see the Regression plots/Joint plot for few models:

A below plot shows the difference between the observed response and the fitted response values.

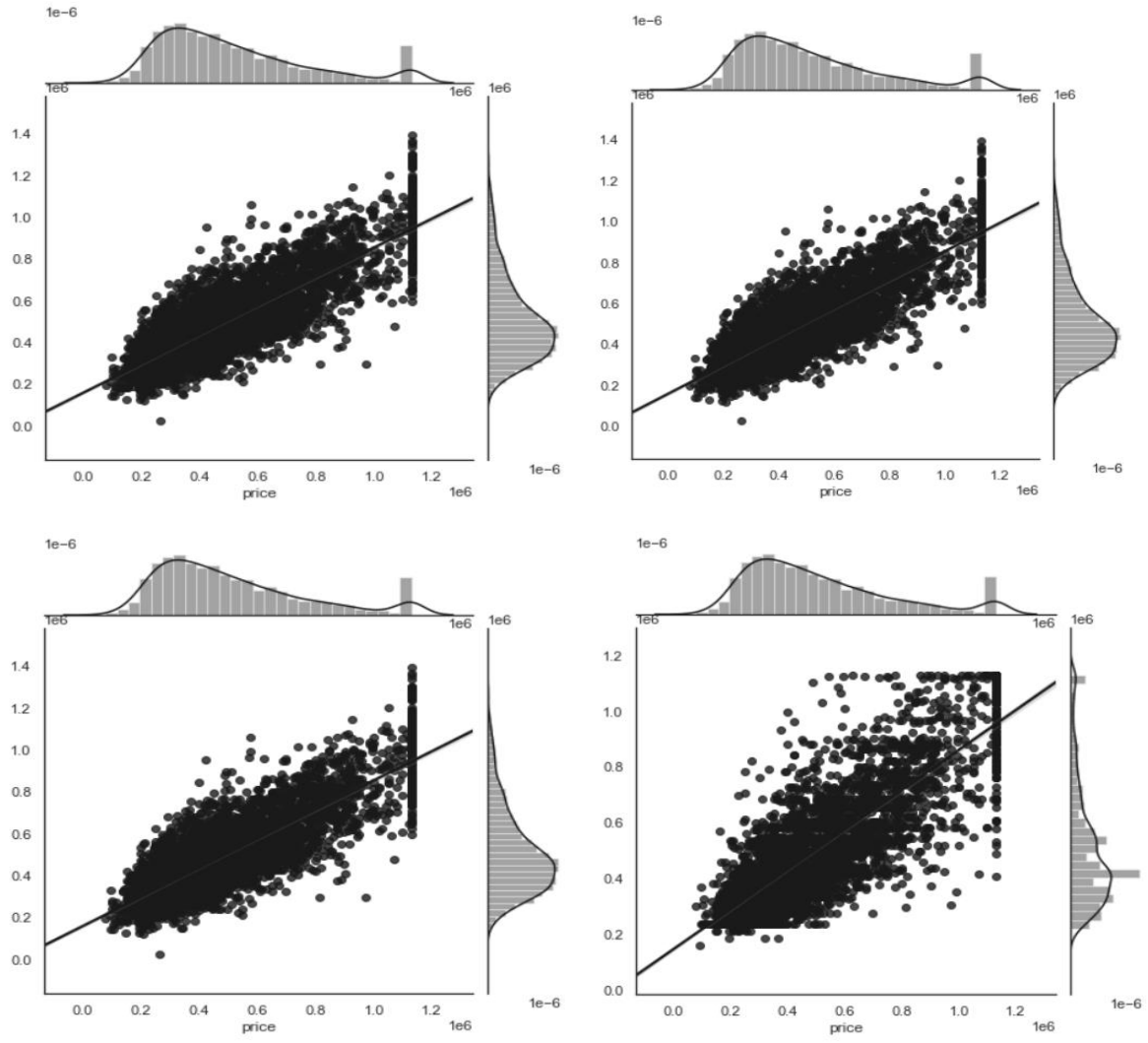


Figure 3 Regression Plot

Later, I have performed the **Ensemble Models** (Bagging, Boosting, Random Forest) models. The **Gradient boosting model** has provided good scores for both training and test sets. **Bagging model** also performed well in train and test sets, there seems to be overfitting in training set. We need to analyse further by hyper tuning the model. **Random Forest Model** performed well in both training and testing data. But there is overfitting for train data.

We can observe that, Ensemble models have performed well on train and test sets. These models can be selected for further analysis with hyper tuning and feature selection.

Let's check the feature importance using Random Forest Model:

```
First 20 feature importance:     Imp    96.121
dtype: float64
First 30 feature importance:     Imp    98.437
dtype: float64
```

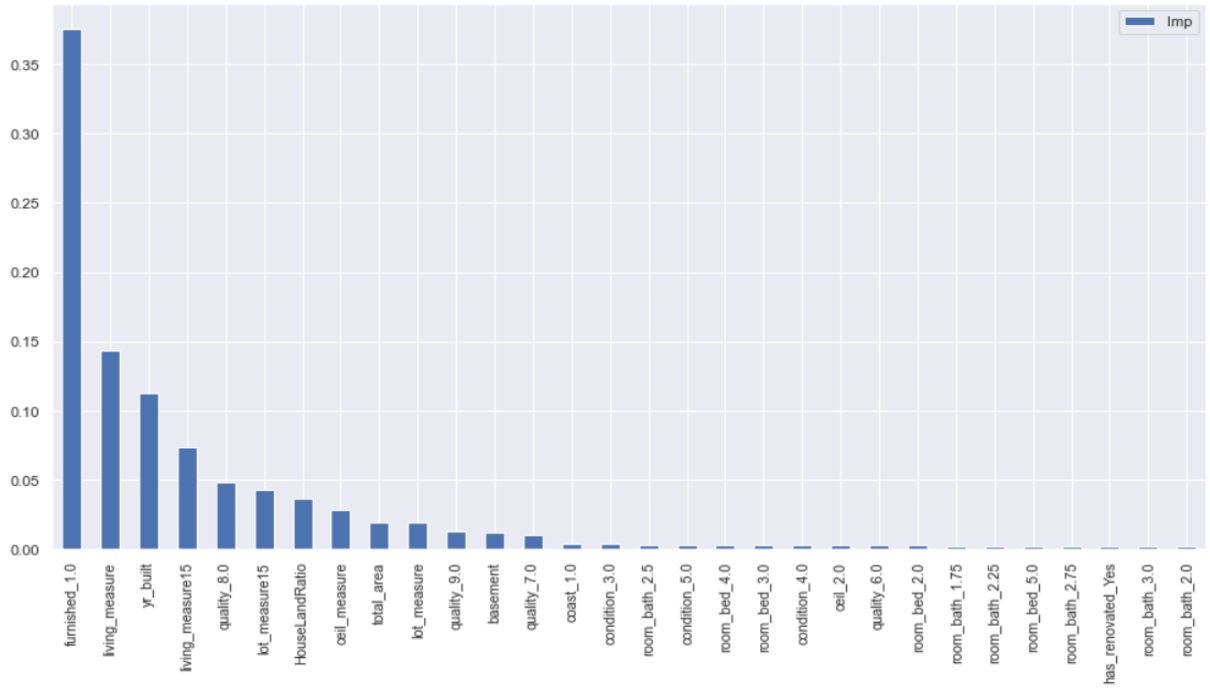


Figure 4 Feature Importance -1

From the Bar Plot,

- we can understand that, above are top 30 important features that account for 98% of variation in model.
- This need to be further analysed during hyper tuning of the models for better scores.

Model performance Summary:

- Ensemble methods are performing better than linear models.
- Among the ensemble models, Gradient Boosting regressor is giving better R2 score for both train and test data.
- we identified that the top 30 features are explaining the 98% variation in model (Random Forest).
- We can further hyper tune the model to improve the model performance.
- We can further explore and evaluate the features while hyper turning the ensemble models.

Let us now create a function/pipeline which will run the model and return the r2 score, RMSE, MSE of the model.

First, we need to import the pipeline library from sklearn.

I have created a function for all the models I need to perform.

After that, we need to create a new data frame to capture the results.

The above sequence of steps with pipeline function will run all the models and compile the scores in a data frame.

We can see that the above 2 steps are concise instead of running individual models and compiling the scores as earlier.

Method	Test score	RMSE_test	MSE_test	MAE_test	Train Score	RMSE_train	MSE_train	MAE_train
LR	0.703668	135222.127869	1.828502e+10	105017.960441	0.682771	141050.005422	1.989510e+10	108009.131351
KNNR	0.535146	169362.387969	2.868362e+10	128384.020547	0.998637	9244.836127	8.546700e+07	820.155986
DTR	0.487547	177822.059596	3.162068e+10	129895.505436	0.998637	9244.836127	8.546700e+07	820.155986
GBR	0.742221	126119.532265	1.590614e+10	97839.489718	0.749858	125250.678917	1.568773e+10	96281.918856
BGR	0.745086	125416.558141	1.572931e+10	95504.808945	0.960147	49994.049363	2.499405e+09	36654.383026
RFR	0.744323	125604.226061	1.577642e+10	95153.067131	0.961526	49121.655940	2.412937e+09	36274.778356

Table 2 Model Comparison table

We can clearly see that Gradient Boosting Regressor (GBR) is giving better result in comparison with other models.

Building function to return the feature importance for each the model

Will run above function with ensemble models: Gradient boosting, Random Forest, Bagging.

Feature Importance for Gradient Boosting Regressor Model:

```
First 25 feature importance:     Imp    99.633
dtype: float64
First 30 feature importance:     Imp    99.8
dtype: float64
```

['furnished_1.0','living_measure','yr_built','living_measure15','quality_8.0','lot_measure15','quality_9.0','HouseLandRatio','quality_7.0','coast_1.0','ceil_measure','basement','room_bath_2.5','condition_5.0','quality_6.0','condition_3.0','quality_10.0','ceil_3.0','total_area','lot_measure','quality_5.0','quality_11.0','room_bath_3.625','has_renovated_Yes','room_bed_2.0','condition_2.0','quality_12.0','room_bath_3.25','room_bath_1.0','room_bed_5.0']

Let's observe the feature importance using bar plot below:

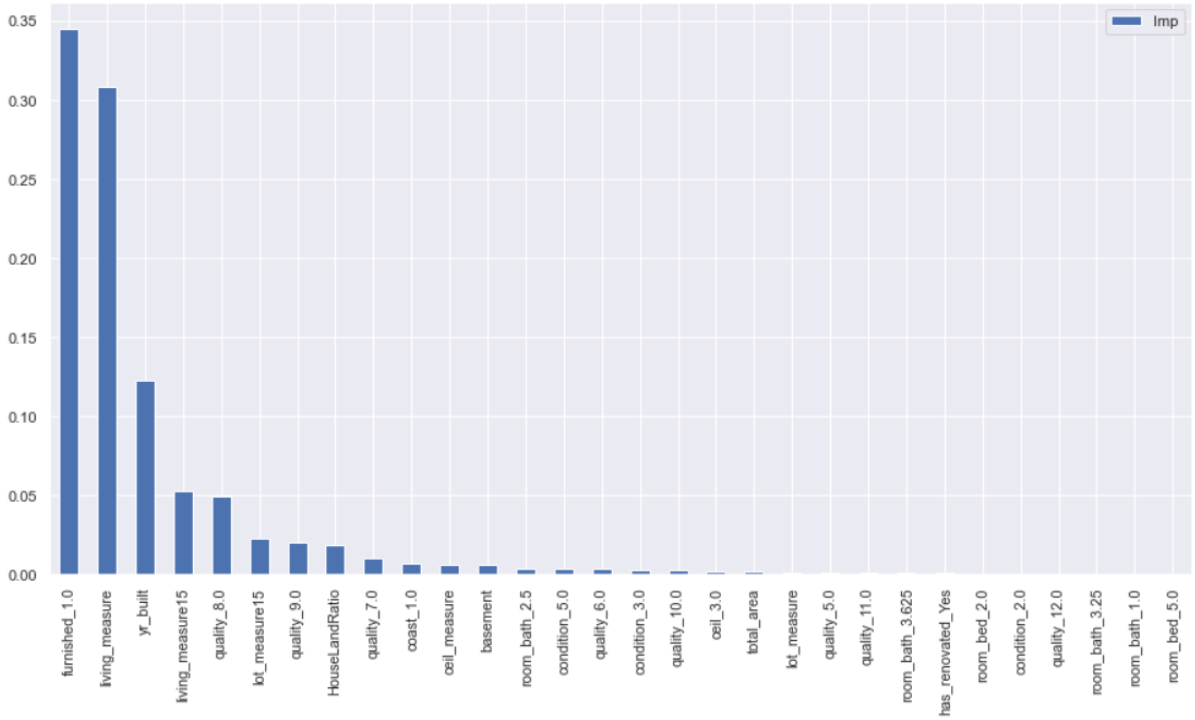


Figure 5 Feature Importance – 2

The top 30 features are covering about 99% in gradient boosting model.

This is very good coverage for just 30% of the variables.

Feature Importance for Random Forest Model:

```
First 25 feature importance:     Imp    97.431
dtype: float64
First 30 feature importance:     Imp    98.428
dtype: float64
```

['furnished_1.0','living_measure','yr_built','living_measure15','quality_8.0','lot_measure15','HouseLandRatio','ceil_measure','lot_measure','total_area','quality_9.0','basement','quality_7.0','coast_1.0','condition_3.0','room_bath_2.5','condition_5.0','room_bed_3.0','quality_6.0','room_bed_4.0','condition_4.0','ceil_2.0','room_bed_2.0','room_bath_1.75','room_bed_5.0','room_bath_2.25','has_renovated_Yes','room_bath_2.75','room_bath_3.0','room_bath_2.0']

Let's observe the feature importance using bar plot below:

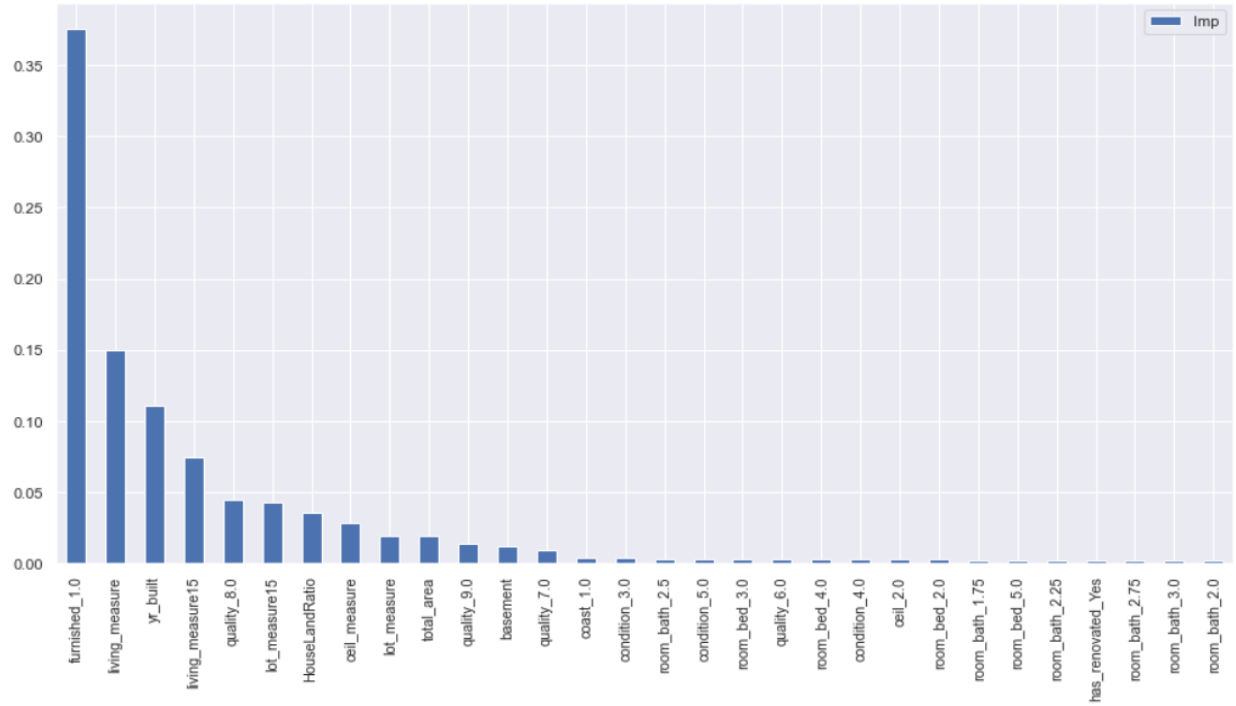


Figure 6 Feature Importance -3

From the above 2 feature list, we will consolidate all the features.

['room_bath_2.75', 'room_bath_2.25', 'room_bath_2.0', 'ceil_2.0', 'basement', 'has_renovated_Yes', 'room_bath_3.0', 'condition_4.0', 'condition_5.0', 'yr_built', 'quality_6.0', 'quality_8.0', 'room_bed_2.0', 'lot_measure15', 'coast_1.0', 'total_area', 'quality_12.0', 'room_bath_2.5', 'furnished_1.0', 'room_bed_4.0', 'lot_measure', 'room_bath_1.75', 'room_bed_3.0', 'living_measure', 'quality_5.0', 'room_bath_1.0', 'room_bath_3.25', 'quality_7.0', 'quality_10.0', 'living_measure15', 'condition_3.0', 'room_bath_3.625', 'room_bed_5.0', 'ceil_measure', 'condition_2.0', 'ceil_3.0', 'quality_9.0', 'HouseLandRatio', 'quality_11.0']

From two models, we have 39 importance features. We will freeze on the above 39 list and make another data frame (along with 'price').

Inference:

In random forest model top 30 features are explaining the 98% variance in the regression.

In gradient boosting model top 30 features are covering 99% variance.

Hence, we conclude that we will use features selection by considering the feature importance function in individual models.

Thus, we extracted 39 important features from both the models.

HYPERTUNING with Grid search CV

Since we have better performance in gradient boosting model, we will hyper tune the model for improving the score

Following are the parameters, we tune for the gradient boosting model.

'loss': ['ls','lad','huber'],

'bootstrap': ['True','False'],

'max_depth': range (5,11,1),

'max_features': ['auto','sqrt'],

'learning_rate': [0.05,0.1,0.2,0.25],

'min_samples_leaf': [4,10,20],

'min_samples_split': [5,10,1000],

'n_estimators': [10,50,100,150,200],

'subsample': [0.8,1]

First will tune each parameter separately

```
({'n_estimators': 400}, 0.7291264789732015)
```

n_estimators of 400 is best in range 50 to 400. Will test same until 1000

```
({'n_estimators': 1000}, 0.7333937820916592)
```

n_estimators of 1000 is giving best result in range 400 to 1000

```
({'learning_rate': 0.1,
  'min_samples_leaf': 10,
  'min_samples_split': 5,
  'n_estimators': 1000},
 0.73437544037117)
```

In combination of 4 parameters above values are giving best result. We can see n_estimators of 1000 is best again. Now, will change the ranges of other 3 parameters.

```
({'learning_rate': 0.1,
  'max_depth': 5,
  'min_samples_leaf': 5,
  'min_samples_split': 30,
  'n_estimators': 1000},
 0.7294779302792567)
```

Now the score has reduced compared to earlier run

Let’s try another time modifying the parameters

```
({'learning_rate': 0.1,
  'max_depth': 5,
  'min_samples_leaf': 10,
  'min_samples_split': 40,
  'n_estimators': 1000},
 0.7295953862235582)
```

The score is almost same as the previous run.

Let’s try for one more iteration,

```
({'learning_rate': 0.1,
  'max_depth': 5,
  'min_samples_leaf': 8,
  'min_samples_split': 50,
  'n_estimators': 1000},
 0.7299259520164757)
```

There is very marginal improvement in score. We are getting best score at min_samples_split of 40 among 30,40,50.

Now, we will tune the final set of parameters along with above finalized ones

'loss': ['ls','lad','huber'],

'max_features': ['auto','sqrt'],

'learning_rate': [0.1],

'max_depth': [5],

'min_samples_leaf': [8],

'min_samples_split': [40],

'n_estimators': [1000],

'subsample': [0.8,1]

```
({'learning_rate': 0.1,
  'loss': 'huber',
  'max_depth': 5,
  'max_features': 'sqrt',
  'min_samples_leaf': 8,
  'min_samples_split': 40,
  'n_estimators': 1000,
  'subsample': 1},
 0.736593660746616)
```

There is improvement in the score. will try one more iteration with changing other parameters.

```
({'learning_rate': 0.1,
  'loss': 'huber',
  'max_depth': 5,
  'max_features': 'sqrt',
  'min_samples_leaf': 5,
  'min_samples_split': 40,
  'n_estimators': 1000,
  'subsample': 1},
 0.7362144962305104)
```

The above iteration gives best result of **0.736.**

Final parameters that are giving best result on training set are:

({'learning_rate': 0.1,

'loss': 'huber',

'max_depth': 5,

'max_features': 'sqrt',

'min_samples_leaf': 8,

'min_samples_split': 40,

'n_estimators': 1000,

'subsample': 1},

0.736593660746616)

Hyper tuning Using graph:

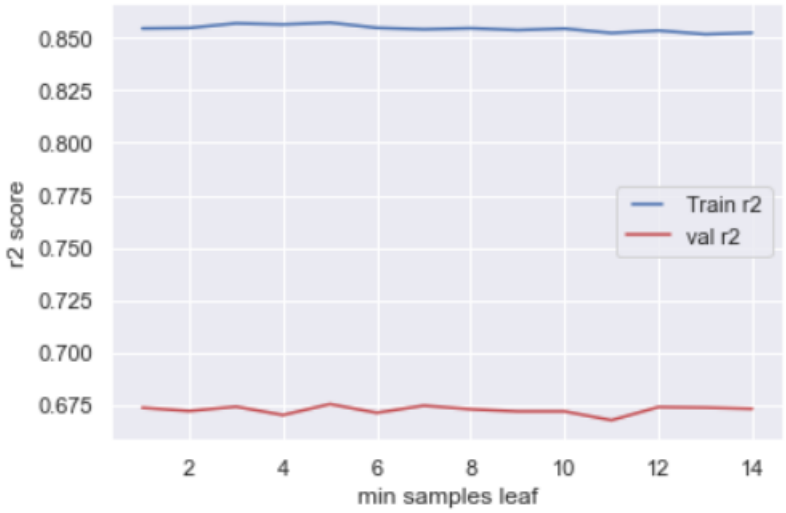


Figure 7 Hyper tuning plot - 1

From above, min_samples_leaf of 5 is giving best score.

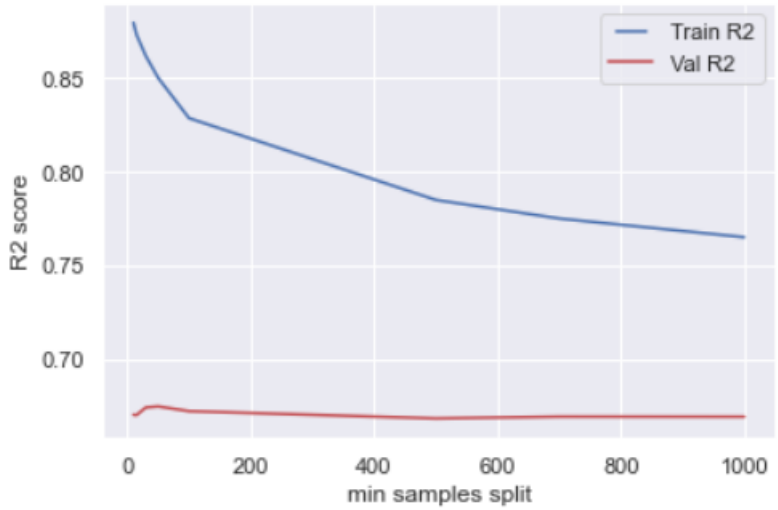


Figure 8 Hyper tuning plot – 2

From above plot, min_samples_splits of about 10 is giving best score. Will try expanding the range around 10.

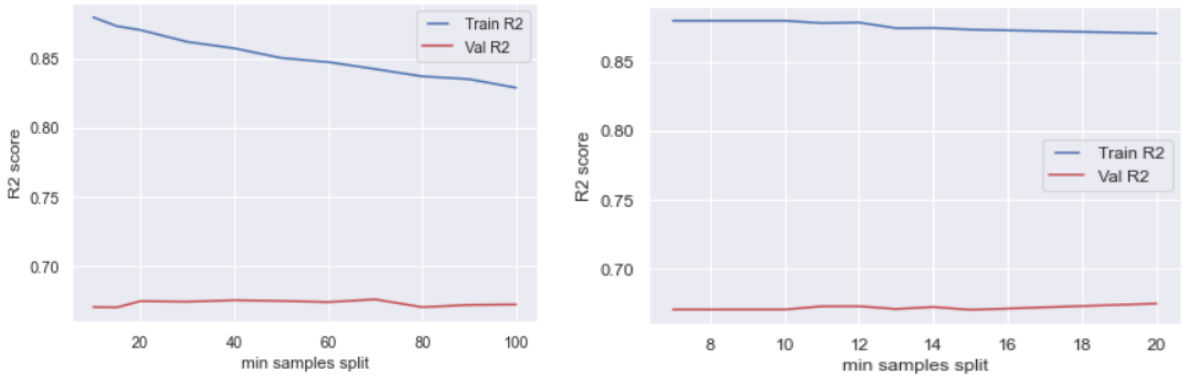


Figure 9 Hyper tuning plots - 3

From above plot, min_samples_splits of about 12 is giving best score

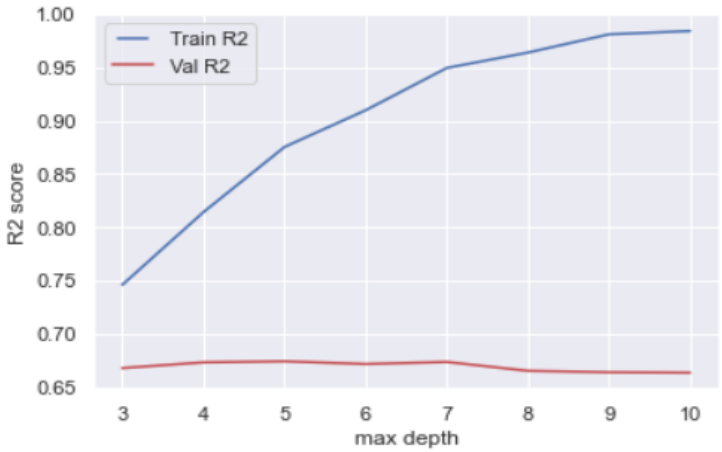


Figure 10 Hyper tuning plot - 4

From above plot, max_depth of about 6 is giving best score.

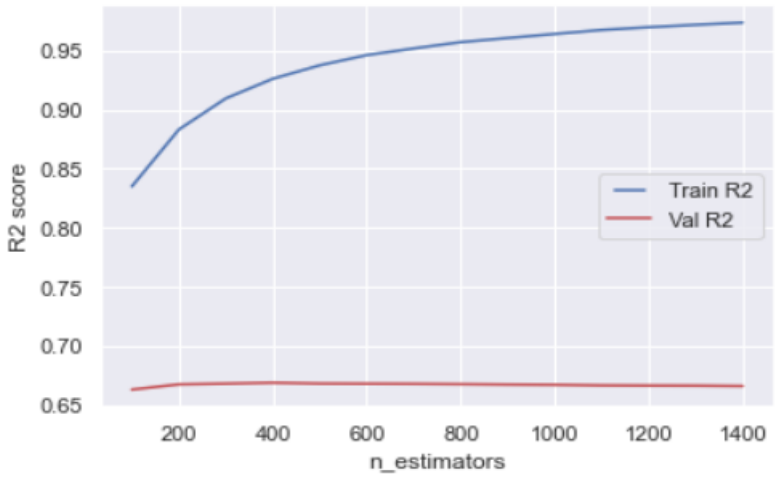


Figure 11 Hyper tuning plot – 5

From above plot, n_estimators of about 1000 is giving best score.

We can conclude from above that gridsearch CV is giving better results compared to that of tuning done by graphical method of individual parameters.

Final parameters that are giving best result on training set are:

(0.735649589588073,

{'learning_rate': 0.1, 'loss': 'huber', 'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 50, 'n_estimators': 1000, 'subsample': 1})

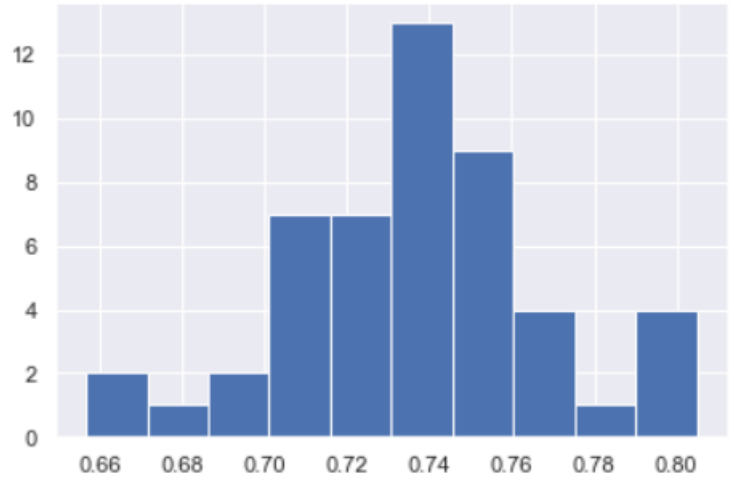
Final Model score:

Method	Train Score	RMSE_train	MSE_train	MAE_train	Test Score	RMSE_test	MSE_test	MAE_test
Gradient Boosting Final Model	0.878475	87300.945397	7.621455e+09	62867.773972	0.745155	125399.617617	1.572506e+10	95173.753529

After Hyper tuning, the performance scores of the model are improved for training & testing.

Confidence Interval:

```
[0.71002429 0.76877207 0.74862785 0.73091034 0.76150352 0.74377982
 0.72543974 0.68779144 0.70284918 0.80314136 0.75762733 0.72954179
 0.75896561 0.72249069 0.65677179 0.74353955 0.75614479 0.69355949
 0.7307417  0.73319464 0.71630881 0.71083506 0.74160629 0.73277915
 0.7260138  0.73538828 0.7568478  0.78811296 0.74094886 0.75157628
 0.70502659 0.73122989 0.76905258 0.71500968 0.77428114 0.72498316
 0.73688792 0.66318558 0.7563255  0.7584498  0.7326168  0.73924833
 0.79960362 0.74936487 0.71045305 0.71099983 0.80481855 0.73187747
 0.67852301 0.79360216]
Accuracy: 73.703% (3.232%)
```



95.0 confidence interval 66.7% and 80.2%

Final summary:

→ The ensemble models have performed well compared to that of linear,KNN,SVR models

→ The best performance is given by Gradient boosting model with training (score-0.74, RMSE:125250.678917), Validation (Testing) (score-0.74, RSME:126119.532265).

→After Hyper tuning the scores for training (score-0.87, RMSE: 87300.945397), Validation (Testing) (score-0.74, RSME: 125399.617617)

The 95% confidence interval scores range from 66.7 to 80%.

→The top key features that drive the price of the property are: 'furnished_1', 'yr_built', 'living_measure','quality_8', 'HouseLandRatio', 'lot_measure15', 'quality_9', 'ceil_measure', 'total_area'.

→The above data is also reinforced by the analysis done during bivariate analysis done with Exploratory data analysis

→For further improvisation in the model scores, the datasets can be made by treating outliers in different ways and further hyper tuning the ensemble models.