HOUSE PRICE PREDICTION- CAPSTONE PROJECT - BUSINESS REPORT

This Business Report shall provide detailed explanation of how we approached each problem given in the capstone project. It shall also provide relative resolution and explanation with regards to the problems.

PRABHJOT KAUR CHAHAL (PGP-DSBA -Online Mar_C 2021)

2021

Contents

PROBLEM UNDERSTANDING 5

DATA REPORT 7

Exploratory Data Analysis: 12

UNIVARIATE ANALYSIS 12

b) Analyzing Feature: dayhours 13

d) Analyzing Feature: room_bed 13

e) Analyzing Feature: room_bath 14

f) Analyzing Feature: Living measure 14

g) Analyzing Feature: lot_measure 15

h) Analyzing Feature: ceil 15

i) Analyzing Feature: coast, sight, condition 15

j) Analyzing Feature: quality 16

m) Analyzing Feature: yr_built 18

n) Analyzing Feature: yr_renovated 18

o) Analyzing Feature: furnished 19

BIVARIATE ANALYSIS 20

a) PairPlot – Pease visit jupyter notebook for better visualization 20

b) Analyzing Bivariate for Feature: month_year 22

c) Analyzing Bivariate for Feature: room_bed 22

d) Analyzing Bivariate for Feature: room_bath 23

e) Analyzing Bivariate for Feature: living_measure 23

f) Analyzing Bivariate for Feature: lot_measure 24

g) Analyzing Bivariate for Feature: ceil 25

h) Analyzing Bivariate for Feature: coast 25

i) Analyzing Bivariate for Feature: sight 25

j) Analyzing Bivariate for Feature: condition 26

k) Analyzing Bivariate for Feature: quality 27

l) Analyzing Bivariate for Feature: ceil_measure 27

m) Analyzing Bivariate for Feature: basement 28

n) Analyzing Bivariate for Feature: yr_renovated 30

p) Analyzing Bivariate for Feature: furnished 31

MULTIVARIATE ANALYSIS 32

MISSING VALUES TREATMENT 33

OUTLIER TREATMENT 34

INSIGHTS FROM EDA 35

MODEL BUILDING AND INTERPRETATION. 36

- Build various models 36

➢ Test your predictive model against the test set using various appropriate performance metrics 36
REGRESSION MODELS 37
KNN Regressor: 39
Support vector regressor: 39
Decision Tree Regressor: 39
Model Tuning 40
Ensemble techniques: 40
Boosting and Bagging: 40
BUILDING FUNCTION/PIPELINE FOR MODELS 41
FEATURE SELECTION (PCA) 42
HYPERTUNING with Gridsearch CV: 46
INSIGHTS 49
RECOMMENDATIONS 49

TABLE OF FIGURES

FIGURE 1: OUTLIERS PRESENT IN DIFFERENT COLUMNS 12
FIGURE 2: UNIVARIATE ANALYSIS FOR PRICE 13
FIGURE 3: UNIVARIATE ANALYSIS FOR BEDROOM 13
FIGURE 4: UNIVARIATE ANALYSIS FOR BATHROOM 14
FIGURE 5: UNIVARIATE ANALYSIS FOR LIVING MEASURE 14
FIGURE 7: UNIVARIATE ANALYSIS FOR CEIL 15
FIGURE 8: UNIVARIATE ANALYSIS FOR QUALITY 16
FIGURE 9: UNIVARIATE ANALYSIS FOR CEIL MEASURE 16
FIGURE 10: UNIVARIATE ANALYSIS FOR BASEMENT 17
FIGURE 11: DISTRIBUTION OF HOUSES HAVING BASEMENT 18
FIGURE 12: UNIVARIATE ANALYSIS FOR YR_BUILT 18
FIGURE 13: UNIVARIATE ANALYSIS FOR YR_RENOVATED 18
FIGURE 14: UNIVARIATE ANALYSIS FOR HOUSES FURNISHED OR NOT 19
FIGURE 15: PAIRPLOT ANALYSIS 20
FIGURE 16: BIVARIATE ANALYSIS FOR MONTH YEAR 22
FIGURE 16: BIVARIATE ANALYSIS FOR BEDROOM 22
FIGURE 17: BIVARIATE ANALYSIS FOR BATHROOM 23
FIGURE 18: BIVARIATE ANALYSIS FOR LIVING MEASURE 23
FIGURE 20: BIVARIATE ANALYSIS FOR CEIL 25
FIGURE 21: BIVARIATE ANALYSIS FOR COAST 25
FIGURE 22: BIVARIATE ANALYSIS FOR SIGHT 25
FIGURE 23: BIVARIATE ANALYSIS FOR CONDITION 26
FIGURE 24: BIVARIATE ANALYSIS FOR QUALITY 27
FIGURE 25: BIVARIATE ANALYSIS FOR CEIL_MEASURE 27
FIGURE 26: BIVARIATE ANALYSIS FOR BASEMENT 28
FIGURE 27: BIVARIATE ANALYSIS FOR YR_RENOVATED 30

PROBLEM UNDERSTANDING

INTRODUCTION

The section aims at introducing the project and the understanding of the objectives of this analysis.
In simpler terms, it targets to understand the real estate market of the geographical location given. Any house is not merely square foot of space that it occupies but, different other factors like, number of bedrooms, bathrooms, floors, parking space, garden space, waterfront/ beachfront, age of the house, age of renovation of the house, etc., are few of the important points that play a major role in determining its cost. So through this project we try to derive below patterns

- Take advantage of all of the feature variables available, use it to analyze and predict house prices.
- Among the available attributes, we try to identify the most valuable attributes highly affecting the price fluctuations
- What is the extent of impact of these factors on the price of the house?
- How to derive a best deal for a house based on these factors?

PROBLEM STATEMENT

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't be too low or too high. To find house price you usually try to find similar properties in your neighborhood and ***based on gathered data we will try to assess your house price***.

1. cid: a notation for a house
2. dayhours: Date house was sold
3. price: Price is prediction target
4. room_bed: Number of Bedrooms/House
5. room_bath: Number of bathrooms/bedrooms
6. living_measure: square footage of the home
7. lot_measure: quare footage of the lot
8. ceil: Total floors (levels) in house
9. coast: House which has a view to a waterfront
10. sight: Has been viewed
11. condition: How good the condition is (Overall)
12. quality: grade given to the housing unit, based on grading system
13. ceil_measure: square footage of house apart from basement
14. basement_measure: square footage of the basement
15. yr_built: Built Year
16. yr_renovated: Year when house was renovated
17. zipcode: zip
18. lat: Latitude coordinate
19. long: Longitude coordinate
20. living_measure15: Living room area in 2015(implies-- some renovations) This might or might not have affected the lotsize area
21. lot_measure15: lotSize area in 2015(implies-- some renovations)
22. furnished: Based on the quality of room
23. total_area: Measure of both living and lot

NEED/SCOPE OF THE PROJECT

The prediction outcomes can help various real estate stakeholders to make more informed decisions.
As business environment of residential house property of Washington DC, especially in and around Seattle City as this dataset belongs to that part during 2014-15 and compared with the price trend of the existing dataset. With the improvement of people's living standards, the demand for houses increases. In the United States, house sales have grown by 34% in the last decade and reached a record high of 5.51 million (units) last year.House price prediction, has therefore attracted widespread attentions because the prediction outcomes can help various real estate stakeholders to make more informed decisions.

How can we get the profitable pricing for the houses and buildings so that neither the seller nor the buyer is at a loss?

- That is where the factors affecting the price of the house come into picture.
- If a fair evaluation of all the factors, how they contribute, why they contribute is made, then a profitable figure can be derived which leads to a win-win situation for both the parties.

- Business & social opportunity is immense, the model can be used by Housing websites algorithms for predicting appropriate prices for selling or buying houses without any bias.
- It would help the companies to use data for various commercial uses.
- Various Brokers & Investors can take more informed & right decisions.

BUSINESS OR SOCIAL OPPORTUNITY

Real estate is a sector contributing a lot to the revenue of the country and also an always active field, this also provides a lot employment opportunities to blue color workers who put their blood, sweat and tears to make a living, also as mentioned above this is the investment option chosen by majority of the public hence the model must aim to do the due diligence to all contributors of the real industry, that is the model should be capable of determining a price that is profitable to the builders, contractors and the buyers.

DATA REPORT

COLLECTION OF DATA:

This section aims at giving a gist on how the data is collected. It can be primary or secondary, first hand or survey based, etc. Since this is the Capstone Project driven by the Great Learning, hence the data of "House Price Prediction" is provided to us from the learning platform. In real life, the data can be from first hand records of brokers and owners, or maybe a record out from the survey of various localities. It can also be a personal interview records done by the society owners. With the data come the need and the reason to clean it, so, if the data is first hand, we can expect certain impurities and misses in the data which can be tackled by using various data science tools. Current data too seems to have certain impurities and misses which we will deal and clean using the tools of data science. All of this is elaborated in the sections following:

DATA DICTIONARY:

This section aims at understanding the meaning of various columns given in the data from the learning platform. This dictionary can also be referred for drawing interpretations of the graphs and the tables given the following sections. This section is taken directly from the PDF shared on the learning platform w.r.t the project of House Price Prediction. No changes had been done to the excel of data. All the changes and cleaning of data has been done only in Python Notebook.

1. cid: a notation for a house
2. dayhours: Date house was sold
3. price: Price is prediction target
4. room_bed: Number of Bedrooms/House
5. room_bath: Number of bathrooms/bedrooms
6. living_measure: square footage of the home
7. lot_measure: quare footage of the lot
8. ceil: Total floors (levels) in house
9. coast: House which has a view to a waterfront
10. sight: Has been viewed
11. condition: How good the condition is (Overall)
12. quality: grade given to the housing unit, based on grading system
13. ceil_measure: square footage of house apart from basement
14. basement_measure: square footage of the basement
15. yr_built: Built Year
16. yr_renovated: Year when house was renovated
17. zipcode: zip
18. lat: Latitude coordinate
19. long: Longitude coordinate
20. living_measure15: Living room area in 2015(implies-- some renovations) This might or might not have affected the lotsize area
21. lot_measure15: lotSize area in 2015(implies-- some renovations)
22. furnished: Based on the quality of room
23. total_area: Measure of both living and lot

As we can see, there are 23 variables given to us so far to help predict the price of the house, it is up to us to take advantage of these variables and build a model that can predict a pricing qualified enough to meet the expectations stated above.

INITIAL EXPLORATORY DATA ANALYSIS:

This section aims at giving a starter look of the data. We will understand how many rows and columns are there in the data. Also, we will explore other visual aspects as to what type of data every column contains, how many missing values/ null values/ NaN values exist in each column, Further, we will take a descriptive look of the data, where we find the mean, median and mode of the data along with other descriptive measures like standard deviation and count. We will also check that if there are any duplicates in the data and if there are any "0" in the data. Further, we will go for understanding the attributes mentioned in different columns, what are their importance and what do they mean. If anyone of them is leading to confusion due to their names, then we will also attempt a renaming of the columns.
All of this will be attempted in the Python Notebook and in the below sections, we will only see the snippets and the interpretations of the codes.

Why all of this study is essential? Basic detailing of any dataset i.e. knowing the number of rows and columns is important because then only we will understand the hugeness of the data and the end to end understanding of all the rows and columns. Similarly, knowing the type of data that each column entails is important since it will help us to know the data format and if that needs any change for a successful data analysis. Multiple times alphabets and signs creep into a column where only numbers are expected. This leads to conversion / interpretation that the column consists of "object"/ "alphabets" and not "numbers"/ "integers" / "float". This can lead to confusion and erroneous outcome.
At times it so happens with the data that the data has been collected but at a lot of instances, it's missing. Now, if there is missing data, then it can impact our analysis. For example, if out of 10 people, we don't know what payment modes 8 people use, then, doing an analysis of the rest 2 and pasting that analysis for all the 10 would be unfair. Hence, as data scientists, it's essential that we understand that where and how much of the data is missing. This helps us to flag it to the company to tell them that what our analysis will cover what we can't help and what they can expect from us. This keeps the analysis to be fair and just. Hence, knowing and treating the missing values is very essential. To treat the missing values in the given dataset, we will first know that how much of the data is missing. If the missing data is less than 30 percentages of total data, then we will simply drop the data. Since dropping it won't lead to a bigger loss, but keeping it may lead to erroneous outcome. If the missing data is more than 30 percentages, then, we impute the values using measures of central tendency as per the data structure.

Descriptive analysis of the data is important to understand the measures of central tendency of the data. This helps us to know whether the data is equally distributed or skewed in nature. This also helps us to know that whether the data has any outliers or not. If the outliers exist, then we can take a call to eradicate them and get a cleaner version. The same amount of negative impact happens when we have duplicates in the data. Duplicates in the data leads to destroying the variance of the data and it's more of a duplication of the same analysis. Hence, if we encounter any duplicates, we shall treat them as well.

Multiple times the data can have "0". This 0 can mean multiple things. It can be part of a grading/ ranking system. It can also have a categorical scaling system. It can even mean an absent value. So, knowing the number of "0" in the data helps us to see that why do they exist, what should we do with them, let them stay or treat them to something better outcome.

We can see the initial look of the data. This tells us that the data has 23 columns.

	12235	14791	1742	17829	14810
cid	1425059178	7942601475	5652600185	3529200190	5631500992
dayhours	2014-05-07 00:00:00	2014-05-20 00:00:00	2014-05-02 00:00:00	2014-05-14 00:00:00	2014-05-15 00:00:00
price	460000	345600	750000	325000	390000
room_bed	3.0	5.0	3.0	3.0	3.0
room_bath	2.0	3.5	1.75	2.5	2.5
living_measure	1760.0	2800.0	2240.0	2220.0	2240.0
lot_measure	9055.0	5120.0	10578.0	6049.0	10800.0
ceil	2	2.5	2	2	2
coast	0	0	0	0	0
sight	0.0	0.0	0.0	0.0	0.0
condition	4	3	5	4	3
quality	7.0	9.0	8.0	8.0	8.0
ceil_measure	1760.0	2800.0	1550.0	2220.0	2240.0
basement	0.0	0.0	690.0	0.0	0.0
yr_built	1985	1903	1923	1990	1996
yr_renovated	0	2005	0	0	0
zipcode	98052	98122	98115	98031	98028
lat	47.6534	47.6059	47.6954	47.3972	47.7433
long	-122.128	-122.31	-122.292	-122.182	-122.229
living_measure15	2010.0	1780.0	1570.0	1980.0	1900.0
lot_measure15	9383.0	5120.0	10578.0	7226.0	9900.0
furnished	0.0	1.0	0.0	0.0	0.0
total_area	10815	7920	12818	8269	13040
month_year	May-2014	May-2014	May-2014	May-2014	May-2014
has_basement	No	No	Yes	No	No
has_renovated	No	Yes	No	No	No

These columns are the different factors that impact the price of the house. Factors like number of bedrooms, number of bathrooms, number of floors, quality of house, condition of house, etc., to name a few. Each column has a different name and a different meaning.

cid is the identity number of the house and hence it's a random combination of digits. We see that number of bedrooms and bathrooms are in decimal numbers or a float type. This means that there can be a store room or an adjacent small bathroom in such cases. If one notices the dayhours column, then there is "T" in between which means that it is not in proper date format and we will need to convert it. We see that longitude column has negative values which can mean West direction of longitude. Similarly, can be a possibility of occurrence in the latitude column too which means north of the equator. Coast and furnished column have 0 and 1 as entries. This means that 0 for coast means the house has no coast view. 1 means it has the coast view. Similarly, 0 for furnished means it's not furnished home and 1 means it's a furnished home.

Output of the shape command tells us the number of ***rows and columns*** in the dataset.

```
(21613, 23)
```

From this, we see that there are 21613 rows and 23 columns. This information tallies from the image too where we got 23 columns in the data. Also, there are total 21613 rows which mean there are 21613 entries of different instances. These rows can be consisting of missing data or duplicates. They can also have unwanted inputs like an object variable in the float/integer column.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 23 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   cid              21613 non-null  int64
 1   dayhours         21613 non-null  object
 2   price            21613 non-null  int64
 3   room_bed         21505 non-null  float64
 4   room_bath        21505 non-null  float64
 5   living_measure   21596 non-null  float64
 6   lot_measure      21571 non-null  float64
 7   ceil             21571 non-null  object
 8   coast            21612 non-null  object
 9   sight            21556 non-null  float64
 10  condition        21556 non-null  object
 11  quality          21612 non-null  float64
 12  ceil_measure     21612 non-null  float64
 13  basement         21612 non-null  float64
 14  yr_built         21612 non-null  object
 15  yr_renovated     21613 non-null  int64
 16  zipcode          21613 non-null  int64
 17  lat              21613 non-null  float64
 18  long             21613 non-null  object
 19  living_measure15 21447 non-null  float64
 20  lot_measure15    21584 non-null  float64
 21  furnished        21584 non-null  float64
 22  total_area       21584 non-null  object
dtypes: float64(12), int64(4), object(7)
memory usage: 3.8+ MB
```

When we dissect the data given, we see that number of bedrooms and number of bathrooms have ***null values***. We get to know this because the above image says that bedrooms and bathrooms have only 21505 non null values. This means that the rest of the entries i.e. (21613 – 21505) number of entries are actually null or NaN. We see a similar happening in the living_measure, in lot_measure, in ceil, coast, sight, condition, quality, ceil_measure,basement, yr_built, living_measure15, lot_measure15, furnished and total_area. All these *columns have null values*. It's important to know and then treat the null values. As mentioned before, if the null values are not treated, then they can lead to erroneous outcome.

Another prominent observation from above image is ***the type of data*** in each column. We see that the data type is either *object, or float64 or int64*. Object datatype happens when alphabets or signs creep in the dataset.

Float64 happens when there are decimals and int64 meaning integer64 happens when there are integer values. An astonishing thing to note is that dayhours is object. It's due to presence of "T" in between. In conclusion of this above image, we see that 12 columns are in float64 nature, 4 columns are of int64 nature and 7 columns are of object nature. If an alphabet or sign exists in the majorly numerical column, then it means that it needs some data cleaning to get to a proper dataset.

As mentioned earlier, it's very essential to know the descriptive statistics of the data. The **descriptive statistics** help us to know the skewness or normal distribution of the data.

Descriptive Statistics Of The Data:

	count	mean	std	min	25%	50%	75%	max
cid	21613.0	4.580302e+09	2.876566e+09	1.000102e+06	2.123049e+09	3.904930e+09	7.308900e+09	9.900000e+09
price	21613.0	5.401822e+05	3.673622e+05	7.500000e+04	3.219500e+05	4.500000e+05	6.450000e+05	7.700000e+06
room_bed	21505.0	3.371355e+00	9.302886e-01	0.000000e+00	3.000000e+00	3.000000e+00	4.000000e+00	3.300000e+01
room_bath	21505.0	2.115171e+00	7.702481e-01	0.000000e+00	1.750000e+00	2.250000e+00	2.500000e+00	8.000000e+00
living_measure	21596.0	2.079861e+03	9.184961e+02	2.900000e+02	1.429250e+03	1.910000e+03	2.550000e+03	1.354000e+04
lot_measure	21571.0	1.510458e+04	4.142362e+04	5.200000e+02	5.040000e+03	7.618000e+03	1.068450e+04	1.651359e+06
sight	21556.0	2.343663e-01	7.664376e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.000000e+00
quality	21612.0	7.656857e+00	1.175484e+00	1.000000e+00	7.000000e+00	7.000000e+00	8.000000e+00	1.300000e+01
ceil_measure	21612.0	1.788367e+03	8.281025e+02	2.900000e+02	1.190000e+03	1.560000e+03	2.210000e+03	9.410000e+03
basement	21612.0	2.915225e+02	4.425808e+02	0.000000e+00	0.000000e+00	0.000000e+00	5.600000e+02	4.820000e+03
yr_renovated	21613.0	8.440226e+01	4.016792e+02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.015000e+03
zipcode	21613.0	9.807794e+04	5.350503e+01	9.800100e+04	9.803300e+04	9.806500e+04	9.811800e+04	9.819900e+04
lat	21613.0	4.756005e+01	1.385637e-01	4.715590e+01	4.747100e+01	4.757180e+01	4.767800e+01	4.777760e+01
living_measure15	21447.0	1.987066e+03	6.855196e+02	3.990000e+02	1.490000e+03	1.840000e+03	2.360000e+03	6.210000e+03
lot_measure15	21584.0	1.276654e+04	2.728699e+04	6.510000e+02	5.100000e+03	7.620000e+03	1.008700e+04	8.712000e+05
furnished	21584.0	1.967198e-01	3.975279e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00

From above analysis we got to know,

- There are few variables, i.e. top 5 and ceil are seem to be normally distributed. Price is the target variable.
- Most columns distribution is Right-Skewed and only few features are Left-Skewed (like room_bath, yr_built, lat).
- We have columns which are Categorical in nature are -> coast, yr_renovated, furnished

Through running the code of ***duplicates***, we were able to understand that there are *no duplicates* in the data. This means that there was no error or duplication the recording of the data.

Further, as mentioned earlier, we are doing an analysis for how many and in which all columns there are ***null values.***

```
cid                    0
dayhours               0
price                  0
room_bed             108
room_bath            108
living_measure        17
lot_measure           42
ceil                  42
coast                  1
sight                 57
condition             57
quality                1
ceil_measure           1
basement               1
yr_built               1
yr_renovated           0
zipcode                0
lat                    0
long                   0
living_measure15     166
lot_measure15         29
furnished             29
total_area            29
dtype: int64
```

- Above analysis, tells us that in the total entries of 21613, there are max missing null values of 166 count in the living_measure15.
- Next, we observe that the columns that have high number of missing data are the number of bedrooms and bathrooms.
- Rest of the columns have substantially less numbers of missing data, like lot_measure15, furnished, total_area have only 29 null values.
- Also, sight and condition too have very lesser i.e. just 57 of the missing values.
- An interesting analysis here is that 166 is the highest number of null value spaces and it is very less than 30 percentages of the total data of 21613. 166 is approximately 7 to 8 percentage of the total data. This implies that maximum only 7 to 8 percentage of the data is missing or null in nature and hence, all these can be dropped from the dataset. Dropping it won't lead to any harm in our analysis. We shall be dropping it all using the SimpleImputer From sklearn. Post dropping the null values, we give a recheck the data in EDA section.

Exploratory Data Analysis:

This section aims at a deeper level of data cleaning for the dataset. It targets to give the univariate analysis, bivariate analysis, remove the unwanted variables, remove the missing values (already done in previous section) outlier treatment, variable transformation and addition of any new variables.

Why is this essential? Since we cannot work on an unclean data, hence the Exploratory Data Analysis aims at cleaning the data to make it ready for processing. Unclean data, filled with missing values, outliers, unwanted variables can make the analysis erroneous and outcome to be misguiding.

UNIVARIATE ANALYSIS

We can see, there are lots of features which *have* ***outliers***. So we might need to treat those before building model

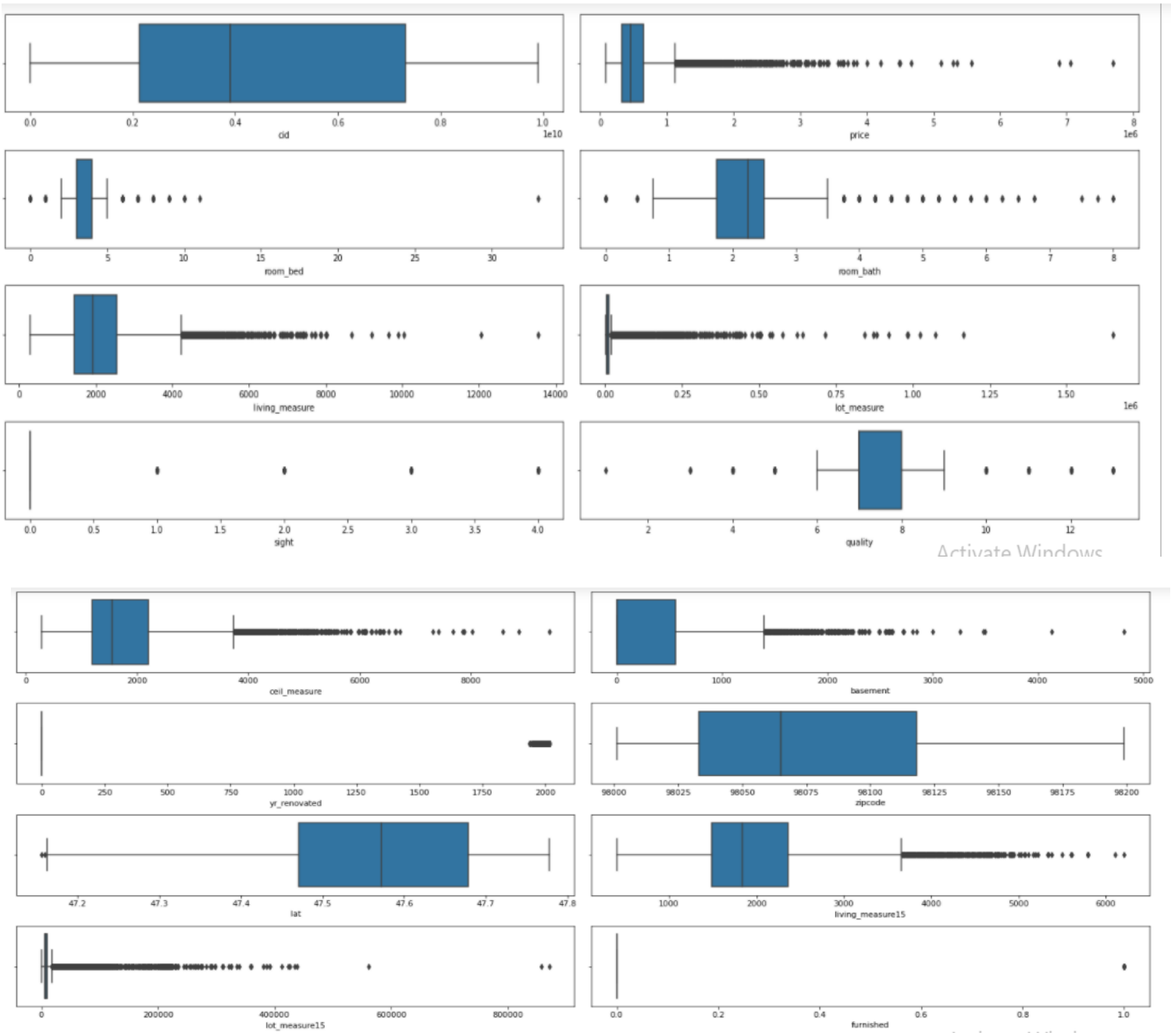


Figure 1: Outliers present in different columns

a) Analyzing Feature: cid:

- We have 176 properties that were sold more than once in the given data

b) Analyzing Feature: dayhours

- We will create new data frame that can be used for modeling
- We will convert the dayhours to 'month_year' as sale month-year is relevant for analysis.
- We successfully converted dayhours feature to month_year for better analysis.
- So the time line of the sale data of the properties is from May-2014 to May-2015 and April month have the highest mean price.

```
0       April-2015
1       March-2015
2      August-2014
3     October-2014
4    February-2015
Name: month_year, dtype: object
```

```
April-2015      2231
July-2014       2211
June-2014       2180
August-2014     1940
October-2014    1878
March-2015      1875
September-2014  1774
May-2014        1768
December-2014   1471
November-2014   1411
February-2015   1250
January-2015     978
May-2015         646
Name: month_year, dtype: int64
```

```
month_year
April-2015      561933.463021
August-2014     536527.039691
December-2014   524602.893270
February-2015   507919.603200
January-2015    525963.251534
July-2014       544892.161013
June-2014       558123.736239
March-2015      544057.683200
May-2014        548166.600113
May-2015        558193.095975
November-2014   522058.861800
October-2014    539127.477636
September-2014  529315.868095
Name: price, dtype: float64
```

c) Analyzing Feature: Price (our Target)

- The Price is ranging from 75,000 to 77, 00,000 and distribution is right-skewed.

```
count    2.161300e+04
mean     5.401822e+05
std      3.673622e+05
min      7.500000e+04
25%      3.219500e+05
50%      4.500000e+05
75%      6.450000e+05
max      7.700000e+06
Name: price, dtype: float64
```

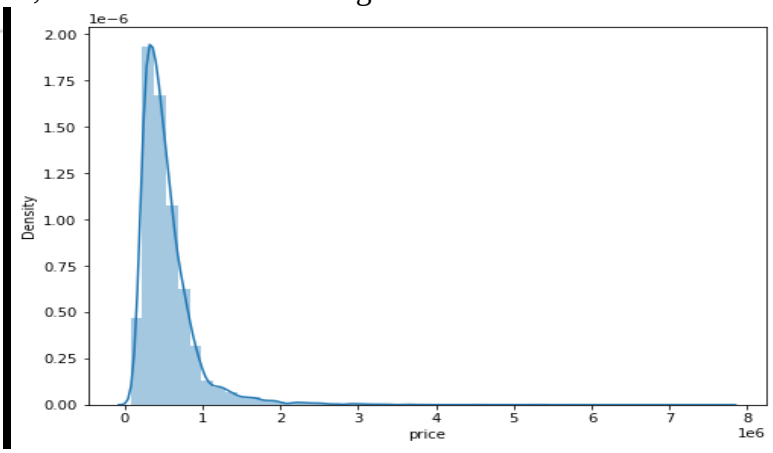


Figure 2: Univariate analysis for price

d) Analyzing Feature: room_bed

- The value of 33 seems to be outlier we need to check the data point before imputing the same
- Most of the houses/properties have 3 or 4 bedrooms

```
3.0     9767
4.0     6854
2.0     2747
5.0     1595
6.0      270
1.0      197
7.0       38
0.0       13
8.0       13
9.0        6
10.0       3
33.0       1
11.0       1
Name: room_bed, dtype: int64
```

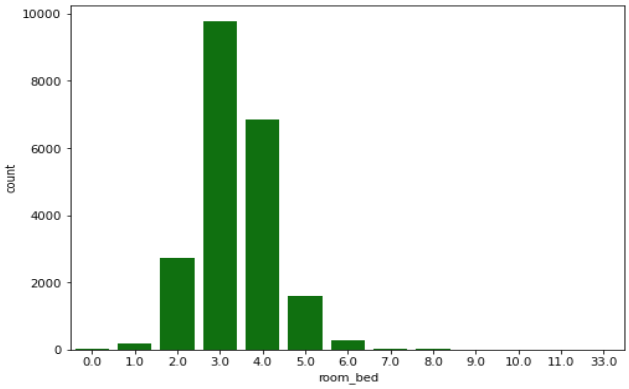


Figure 3: Univariate analysis for bedroom

e) Analyzing Feature: room_bath

- Majority of the properties have bathroom in the range of 1.0 to 2.5

```
0.00       10
0.50        4
0.75       72
1.00     3829
1.25        9
1.50     1439
1.75     3031
2.00     1917
2.25     2039
2.50     5358
2.75     1178
3.00      750
3.25      588
3.50      726
3.75      155
4.00      135
4.25       78
4.50      100
4.75       23
5.00       21
5.25       13
5.50       10
5.75        4
6.00        6
6.25        2
6.50        2
6.75        2
7.50        1
7.75        1
8.00        2
Name: room_bath, dtype: int64
```

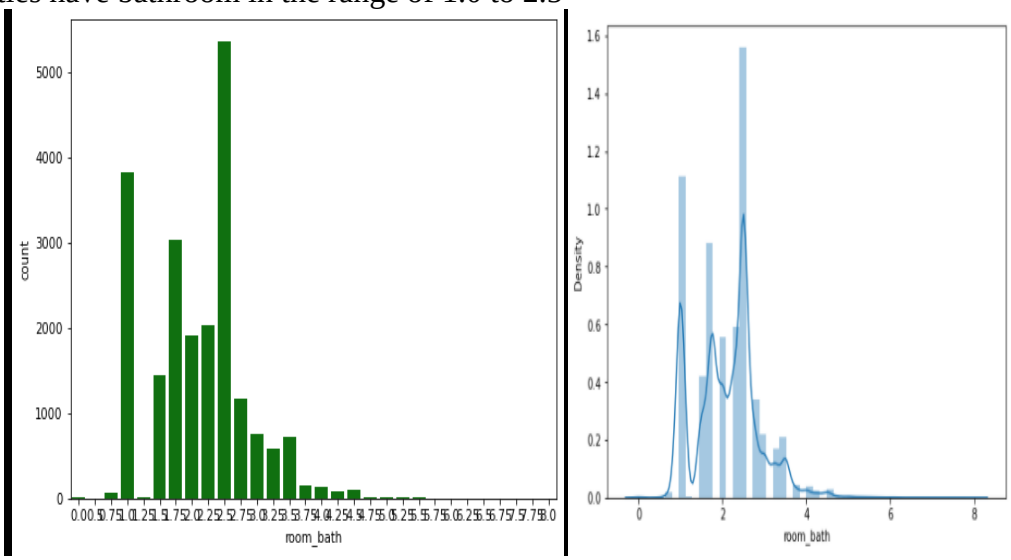


Figure 4: Univariate analysis for bathroom

f) Analyzing Feature: Living measure

- Data distribution tells us, living_measure is right-skewed.
- There are many outliers in living measure. Need to review further to treat the same.

```
count    21596.000000
mean      2079.860761
std        918.496121
min        290.000000
25%       1429.250000
50%       1910.000000
75%       2550.000000
max      13540.000000
Name: living_measure, dtype: float64
```

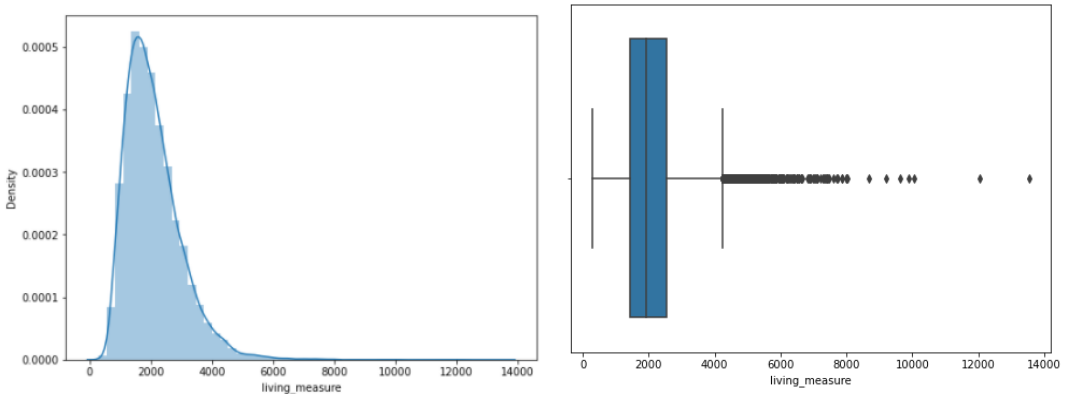


Figure 5: Univariate analysis for living measure

	cid	dayhours	price	room_bed	room_bath	living_measure	lot_measure	ceil	coast	sight	...	yr_built	yr_renovated	zipcode	lat	
1068	6762700020	2014-10-13	7700000	6.0	8.00	12050.0	27600.0	2.5	0	3.0	...	1910	1987	98102	47.6298	-12
1245	1924059029	2014-06-17	4670000	5.0	6.75	9640.0	13068.0	1	1	4.0	...	1983	2009	98040	47.5570	-1
7928	1225069038	2014-05-05	2280000	7.0	8.00	13540.0	307752.0	3	0	4.0	...	1999	0	98053	47.6675	-12
10639	9208900037	2014-09-19	6890000	6.0	7.75	9890.0	31374.0	2	0	4.0	...	2001	0	98039	47.6305	
10718	9808700762	2014-06-11	7060000	5.0	4.50	10040.0	37325.0	2	1	2.0	...	1940	2001	98004	47.6500	-12
12794	2470100110	2014-08-04	5570000	5.0	5.75	9200.0	35069.0	2	0	0.0	...	2001	0	98039	47.6289	-12
20038	1247600105	2014-10-20	5110000	5.0	5.25	8010.0	45517.0	2	1	4.0	...	1999	0	98033	47.6767	-12
20193	2303900035	2014-06-11	2890000	5.0	6.25	8670.0	64033.0	2	0	4.0	...	1965	2003	98177	47.7295	-12
20746	6072800246	2014-07-02	3300000	5.0	6.25	8020.0	21738.0	2	0	0.0	...	2001	0	98006	47.5675	-12

- We have only 9 properties/house which have more than 8k living_measure. So will treat these outliers.

g) Analyzing Feature: lot_measure

```
count    2.157100e+04
mean     1.510458e+04
std      4.142362e+04
min      5.200000e+02
25%      5.040000e+03
50%      7.618000e+03
75%      1.068450e+04
max      1.651359e+06
Name: lot_measure, dtype: float64
```

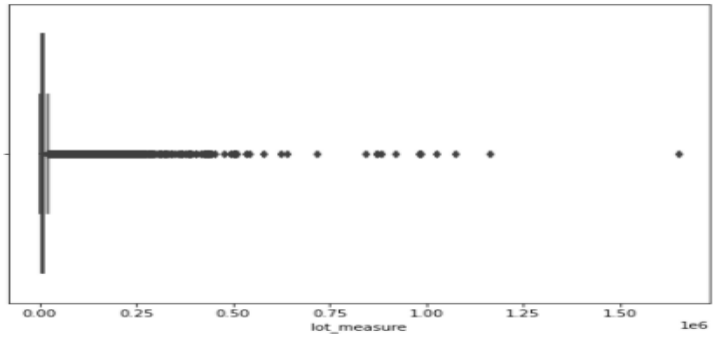


Figure 6: Univariate analysis for lot measure

	cid	dayhours	price	room_bed	room_bath	living_measure	lot_measure	ceil	coast	sight	...	yr_built	yr_renovated	zipcode	lat	l
11674	1020069017	2015-03-27	700000	4.0	1.0	1300.0	1651359.0	1	0	3.0	...	1920	0	98022	47.2313	-122

1 rows × 24 columns

- We have only 1 property with more than 12,50,000 lot_measure. So we need to treat this.

h) Analyzing Feature: ceil

- We can see, most houses have 1 floor
- Above grapth confirming the same, that most properties have 1 and 2 floors

```
1.0    10719
2.0     8210
1.5     1905
3.0      610
2.5      161
3.5        8
Name: ceil, dtype: int64
```

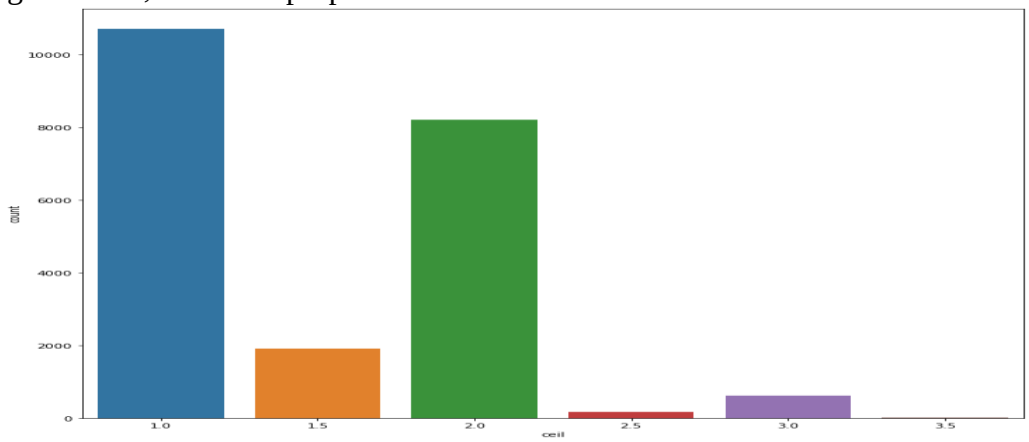


Figure 7: Univariate analysis for ceil

i) Analyzing Feature: coast, sight, condition

```
0.0    21452
1.0      161
Name: coast, dtype: int64
```

```
0.0    19437
2.0      959
3.0      510
1.0      332
4.0      318
Name: sight, dtype: int64
```

```
3.0    14063
4.0     5655
5.0     1694
2.0      171
1.0       30
Name: condition, dtype: int64
```

- coast - most houses donot have waterfront view, very few are waterfront
- sight - most sights have not been viewed
- condition - Overall most houses are rated as 3 and above for its condition overall

j) Analyzing Feature: quality

- Quality - most properties have quality rating between 6 to 10

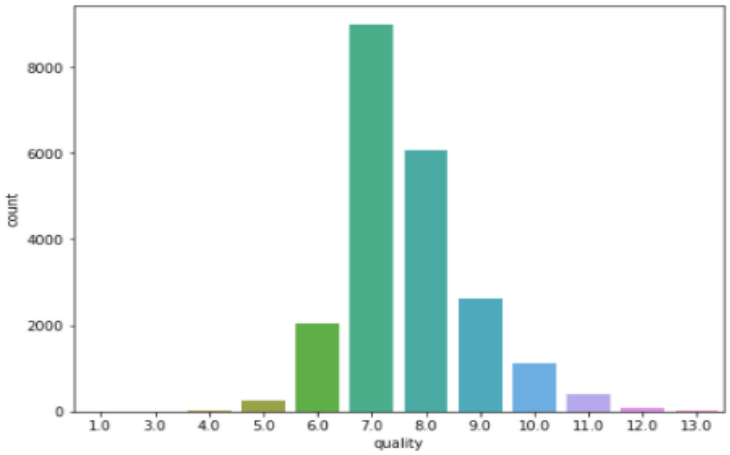


Figure 8: Univariate analysis for quality

Checking the no. of data points with quality rating as 13

	cid	dayhours	price	room_bed	room_bath	living_measure	lot_measure	ceil	coast	sight	...	yr_built	yr_renovated	zipcode	lat	
388	853200010	2014-07-01	3800000	5.0	5.50	7050.0	42840.0	1	0	2.0	...	1978	0	98004	47.6229	-1
1068	6762700020	2014-10-13	7700000	6.0	8.00	12050.0	27600.0	2.5	0	3.0	...	1910	1987	98102	47.6298	-12
3271	7237501190	2014-10-10	1780000	4.0	3.25	4890.0	13402.0	2	0	0.0	...	2004	0	98059	47.5303	-12
3649	2426039123	2015-01-30	2420000	5.0	4.75	7880.0	24250.0	2	0	2.0	...	1996	0	98177	47.7334	-12
4371	1725059316	2014-11-20	2390000	4.0	4.00	6330.0	13296.0	2	0	2.0	...	2000	0	98033	47.6488	-12
8420	9831200500	2015-03-04	2480000	5.0	3.75	6810.0	7500.0	2.5	0	0.0	...	1922	0	98102	47.6285	-12
10639	9208900037	2014-09-19	6890000	6.0	7.75	9890.0	31374.0	2	0	4.0	...	2001	0	98039	47.6305	
10832	4139900180	2015-04-20	2340000	4.0	2.50	4500.0	35200.0	1	0	0.0	...	1988	0	98006	47.5477	-12
11459	1068000375	2014-09-23	3200000	6.0	5.00	7100.0	18200.0	2.5	0	0.0	...	1933	2002	98199	47.6427	-12
12794	2470100110	2014-08-04	5570000	5.0	5.75	9200.0	35069.0	2	0	0.0	...	2001	0	98039	47.6289	-12
16985	2303900100	2014-09-11	3800000	3.0	4.25	5510.0	35000.0	2	0	4.0	...	1997	0	98177	47.7296	-1
20193	2303900035	2014-06-11	2890000	5.0	6.25	8670.0	64033.0	2	0	4.0	...	1965	2003	98177	47.7295	-12
20547	3303850390	2014-12-12	2980000	5.0	5.50	7400.0	18898.0	2	0	3.0	...	2001	0	98006	47.5431	-12

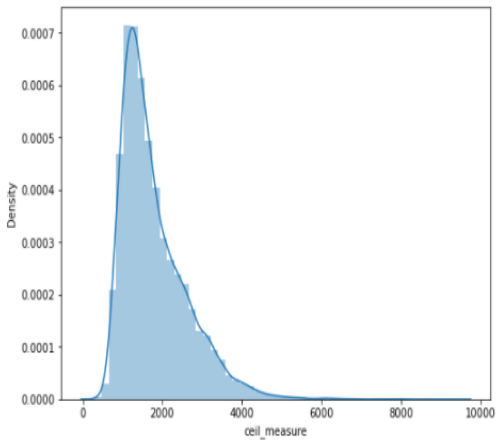
13 rows × 24 columns

- There are only 13 propeties which have the highest quality rating

k) Analyzing Feature: ceil_measure

- ceil_measure - its highly skewed
- There is no pattern in Ceil Vs Ceil_measure
- The vertival lines at each point represent the inter quartile range of values at that point

```
Skewness is : 1.4467467285674296

count    21612.000000
mean      1788.366556
std        828.102535
min        290.000000
25%       1190.000000
50%       1560.000000
75%       2210.000000
max       9410.000000
Name: ceil_measure, dtype: float64
```



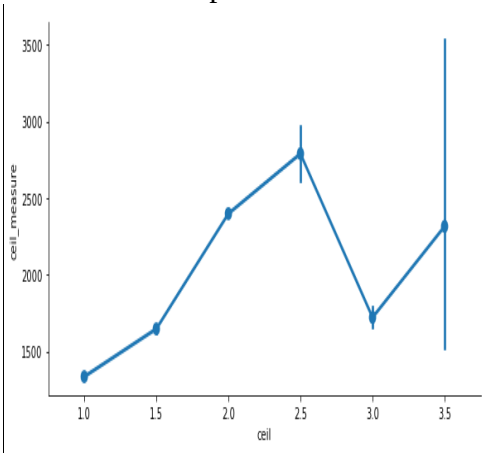
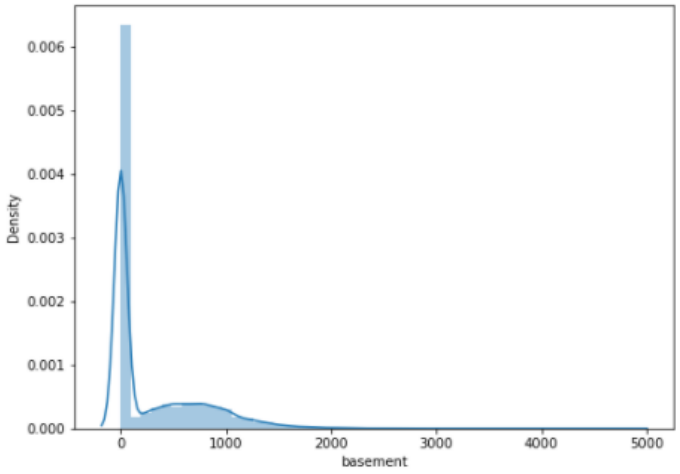


Figure 9: Univariate analysis for ceil measure

l) Analyzing Feature: basement

- We can see 2 gaussians, which tells us there are propeties which don't have basements and some have the basements
- We have almost 60% of the properties without basement
- Houses have zero measure of basement i.e. they donot have basements



Let's plot boxplot for properties which have basements only

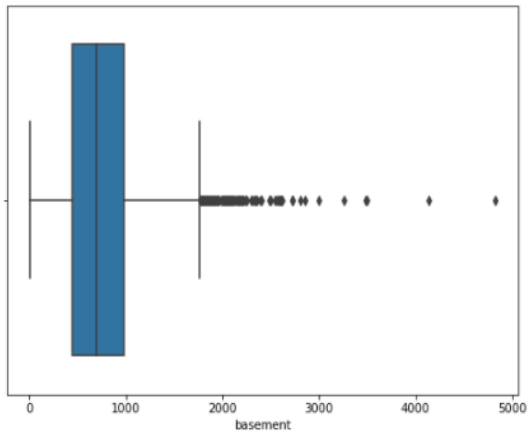


Figure 10: Univariate analysis for basement

- We can clearly see, there are outliers. We need to treat this before our model.

Checking the no. of data points with 'basement' greater than 4000

	cid	dayhours	price	room_bed	room_bath	living_measure	lot_measure	ceil	coast	sight	...	yr_built	yr_renovated	zipcode	lat	
1245	1924059029	2014-06-17	4670000	5.0	6.75	9640.0	13068.0	1	1	4.0	...	1983	2009	98040	47.5570	-12
7928	1225069038	2014-05-05	2280000	7.0	8.00	13540.0	307752.0	3	0	4.0	...	1999	0	98053	47.6675	-121

2 rows × 24 columns

- We have only 2 properties with more than 4,000 measure basement

Distribution of houses having basement:

- Distribution having basement is right-skewed

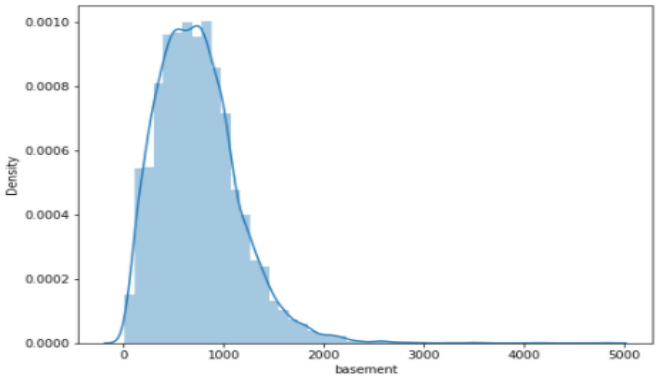


Figure 11: Distribution of houses having basement

m) Analyzing Feature: yr_built

- The built year of the properties range from 1900 to 2014 and we can see upward trend with time

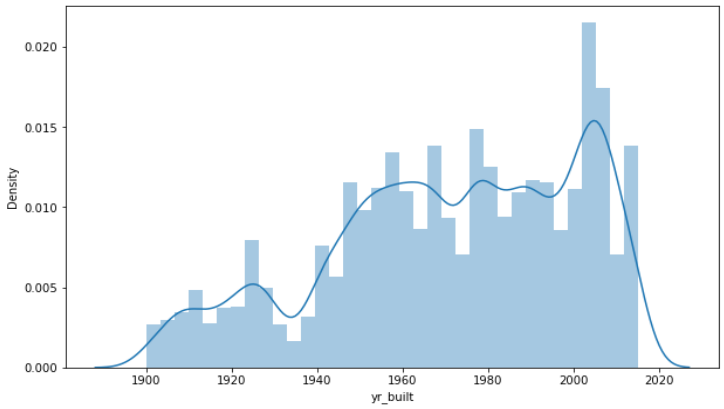


Figure 12: Univariate analysis for yr_built

n) Analyzing Feature: yr_renovated

- Only 914 houses were renovated out of 21613 houses

- yr_renovated - plot of houses which are renovated

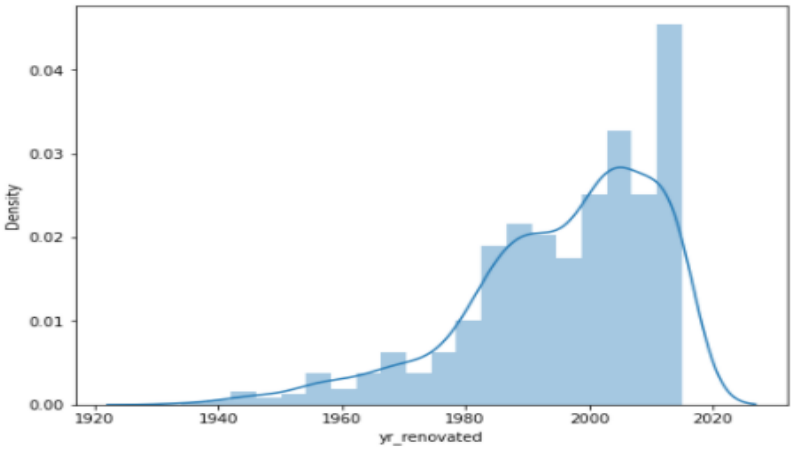


Figure 13: Univariate analysis for yr_renovated

- Now will create age column from columns : yr_built & yr_renovated

o) Analyzing Feature: furnished

- Most properties are not furnished. Furnish column need to be converted into categorical column

```
0.0    17338
1.0     4246
Name: furnished, dtype: int64
```

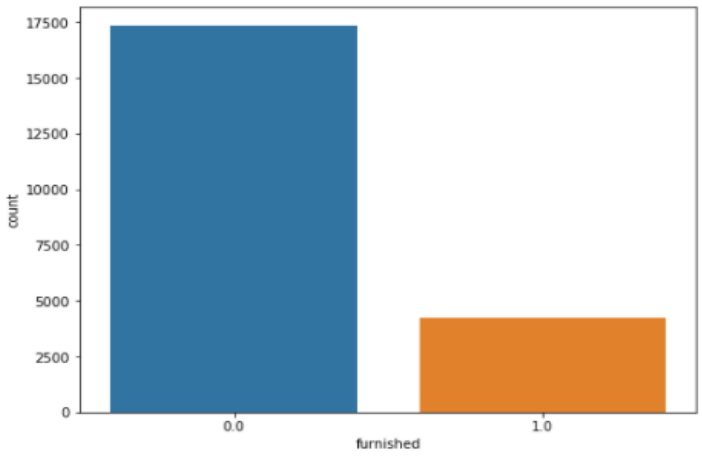


Figure 14: Univariate analysis for houses furnished or not

BIVARIATE ANALYSIS

a) PairPlot – Pease visit jupyter notebook for better visualization

let's plot all the variables and confirm our above deduction with more confidence

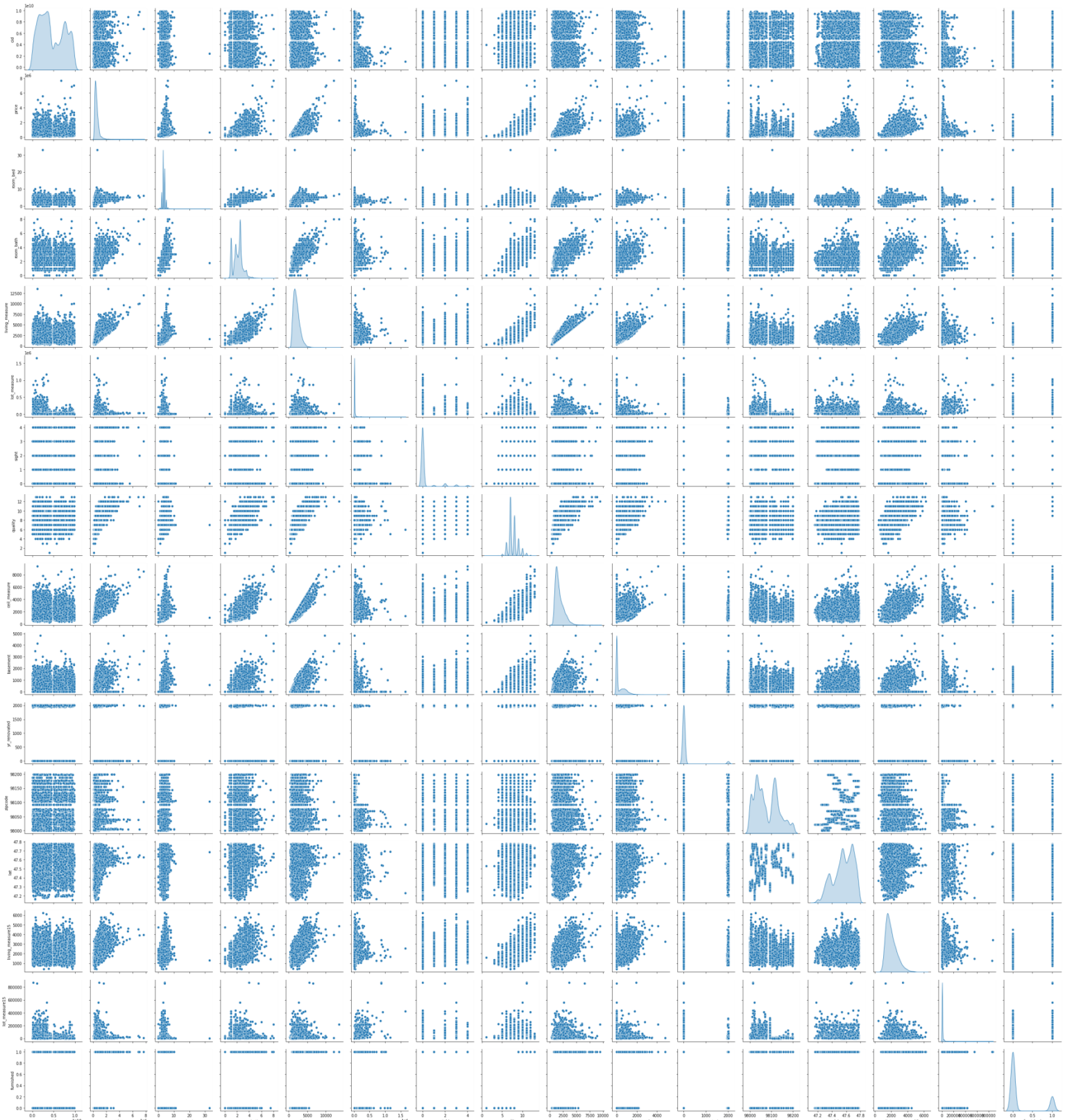


Figure 15: Pairplot analysis

From above pair plot, we observed/deduced below

1. **price:** price distribution is Right-Skewed as we deduced earlier from our 5-factor analysis
2. **room_bed:** our target variable (price) and room_bed plot is not linear. It's distribution have lot of gaussians
3. **room_bath:** It's plot with price has **somewhat linear relationship**. Distribution has number of gaussians.
4. **living_measure:** Plot against price has **strong linear relationship**. It also have linear relationship with room_bath variable. So **might remove one of these 2**. Distribution is Right-Skewed.
5. **lot_measure: No clear relationship** with price.
6. **ceil: No clear relationship** with price. We can see, it's **have 6 unique values** only. Therefore, we can **convert this column into categorical column** for values.
7. **coast: No clear relationship** with price. Clearly it's **categorical variable with 2 unique values**.
8. **sight: No clear relationship** with price. This has **5 unique values**. Can be **converted to Categorical variable**.
9. **condition: No clear relationship** with price. This has **5 unique values**. Can be **converted to Categorical variable**.
10. **quality: Somewhat linear relationship with price**. Has **discrete values from 1 - 13. Can be converted to Categorical variable**.
11. **ceil_measure: Strong linear relationship with price**. Also with room_bath and living_measure features. Distribution is **Right-Skewed**.
12. **basement: No clear relationship** with price.
13. **yr_built: No clear relationship** with price.
14. **yr_renovated: No clear relationship** with price. Have **2 unique values. Can be converted to Categorical Variable** which tells whether house is renovated or not.
15. **zipcode, lat, long: No clear relationship** with price or any other feature.
16. **living_measure15: Somewhat linear relationship with target feature**. It's same as living_measure. Therefore we can drop this variable.
17. **lot_measure15: No clear relationship** with price or any other feature.
18. **furnished: No clear relationship** with price or any other feature. **2 unique values so can be converted to Categorical Variable**
19. **total_area: No clear relationship with price**. But it has **Very Strong linear relationship with lot_measure**. So one of it can be dropped.

b) Analyzing Bivariate for Feature: month_year

month_year	mean	median	size
April-2015	561933.463021	476500	2231
August-2014	536527.039691	442100	1940
December-2014	524602.893270	432500	1471
February-2015	507919.603200	425545	1250
January-2015	525963.251534	438500	978
July-2014	544892.161013	465000	2211
June-2014	558123.736239	465000	2180
March-2015	544057.683200	450000	1875
May-2014	548166.600113	465000	1768
May-2015	558193.095975	455000	646
November-2014	522058.861800	435000	1411
October-2014	539127.477636	446900	1878
September-2014	529315.868095	450000	1774

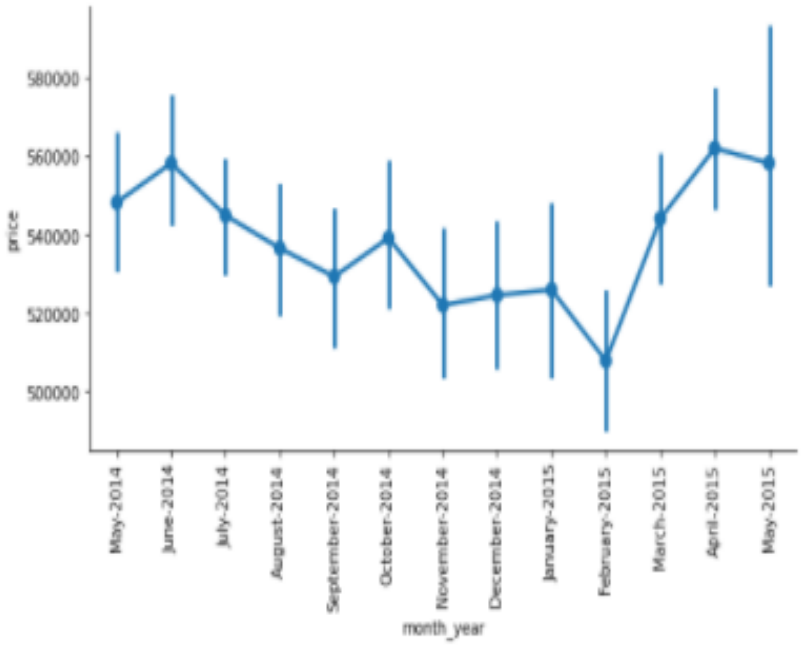


Figure 16: Bivariate analysis for month year

- The mean price of the houses tend to be high during March,April, May as compared to that of September, October, November,December period.

c) Analyzing Bivariate for Feature: room_bed

room_bed	mean	median	size
0.0	4.102231e+05	288000	13
1.0	3.189286e+05	299000	197
2.0	4.013572e+05	373500	2747
3.0	4.662970e+05	413000	9767
4.0	6.357284e+05	549950	6854
5.0	7.867329e+05	619000	1595
6.0	8.274895e+05	652500	270
7.0	9.514478e+05	728580	38
8.0	1.105077e+06	700000	13
9.0	8.939998e+05	817000	6
10.0	8.200000e+05	660000	3
11.0	5.200000e+05	520000	1
33.0	6.400000e+05	640000	1

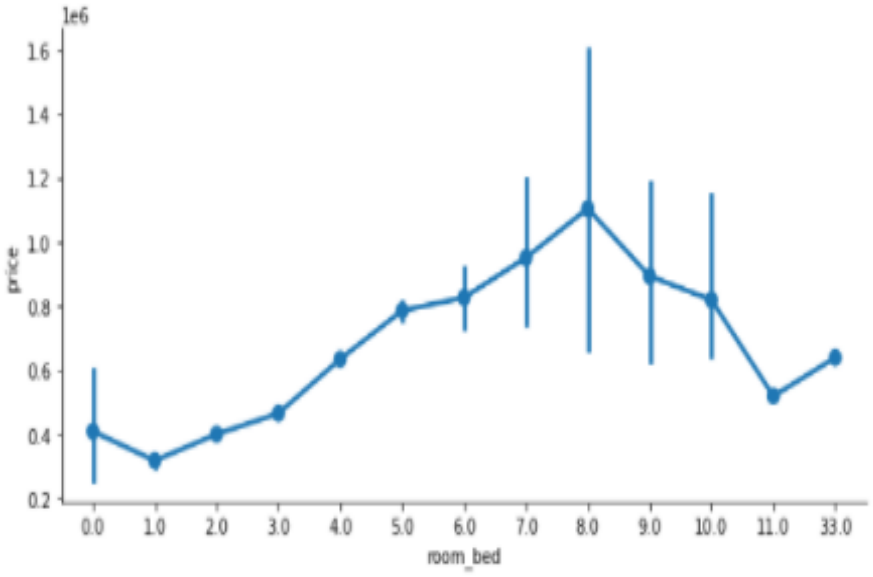


Figure 16: Bivariate analysis for bedroom

- Room_bed - outliers can be seen easily. Mean and median of price increases with number bedrooms/house uptill a point
- There is clear increasing trend in price with room_bed

d) Analyzing Bivariate for Feature: room_bath

room_bath	mean	median	size
0.00	4.490950e+05	317500	10
0.50	2.373750e+05	264000	4
0.75	2.945209e+05	273500	72
1.00	3.470689e+05	320000	3829
1.25	6.217722e+05	516500	9
1.50	4.091834e+05	370000	1439
1.75	4.549409e+05	422800	3031
2.00	4.575184e+05	422500	1917
2.25	5.337925e+05	472500	2039
2.50	5.536595e+05	499950	5358
2.75	6.605538e+05	605000	1178
3.00	7.067325e+05	600000	750
3.25	9.710299e+05	837352	588
3.50	9.329520e+05	823250	726
3.75	1.198179e+06	1070000	155
4.00	1.271616e+06	1060000	135
4.25	1.535072e+06	1390000	78
4.50	1.334211e+06	1060000	100
4.75	2.022300e+06	2300000	23
5.00	1.674167e+06	1430000	21
5.25	1.817962e+06	1420000	13
5.50	2.522500e+06	2340000	10
5.75	2.492500e+06	1930000	4
6.00	2.948333e+06	2895000	6
6.25	3.095000e+06	3095000	2
6.50	1.710000e+06	1710000	2
6.75	2.735000e+06	2735000	2
7.50	4.500000e+05	450000	1
7.75	6.890000e+06	6890000	1
8.00	4.990000e+06	4990000	2

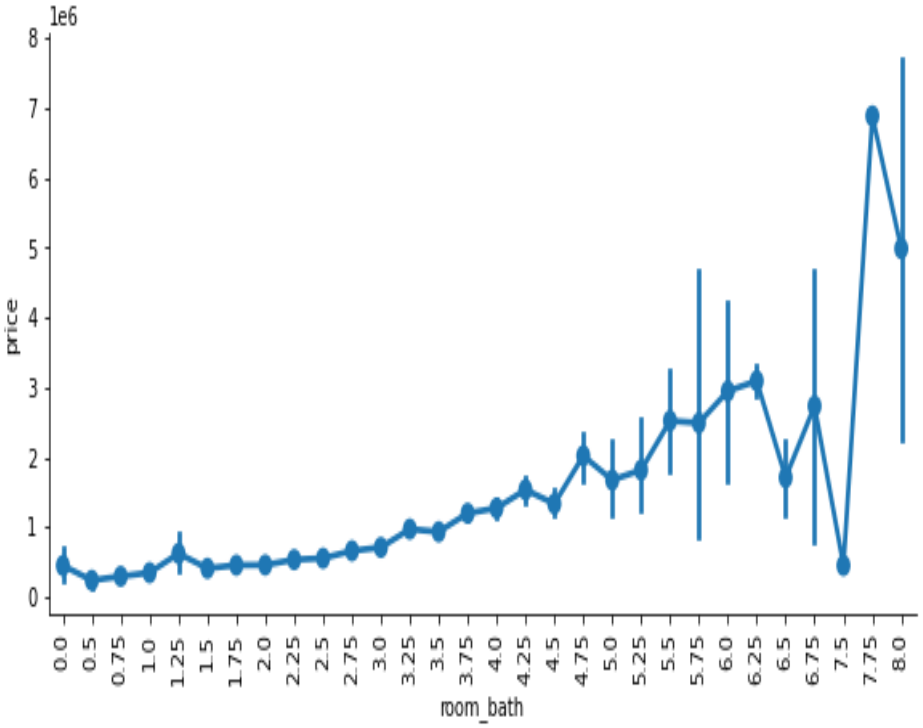


Figure 17: Bivariate analysis for bathroom

- room_bath - outliers can be seen easily. Overall mean and median price increares with increasing room_bath
- There is upward trend in price with increase in room_bath

e) Analyzing Bivariate for Feature: living_measure

```
count    21596.000000
mean      2079.860761
std        918.496121
min        290.000000
25%       1429.250000
50%       1910.000000
75%       2550.000000
max      13540.000000
Name: living_measure, dtype: float64
```

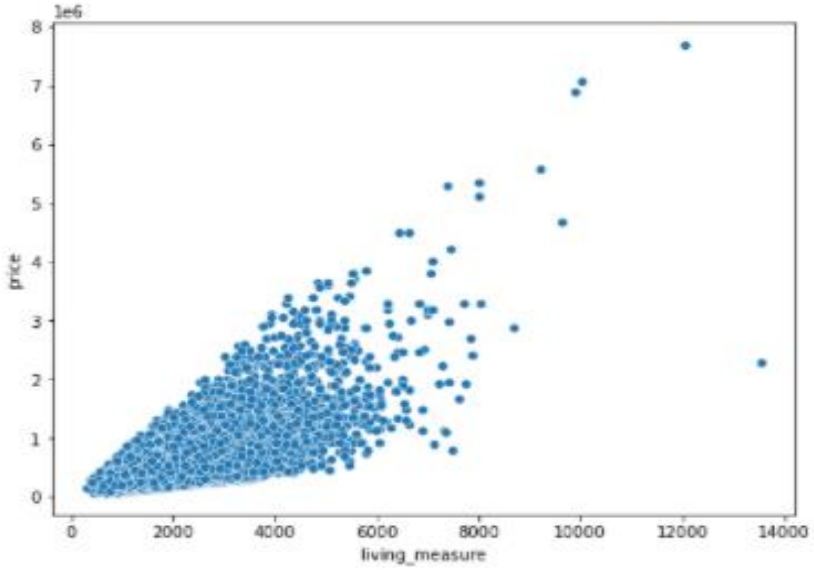


Figure 18: Bivariate analysis for living measure

- living_measure - price increases with increase in living measure
- There is clear increment in price of the property with increment in the living measure But there seems to be one outlier to this trend. Need to evaluate the same

f) Analyzing Bivariate for Feature: lot_measure

```
count    2.157100e+04
mean     1.510458e+04
std      4.142362e+04
min      5.200000e+02
25%      5.040000e+03
50%      7.618000e+03
75%      1.068450e+04
max      1.651359e+06
Name: lot_measure, dtype: float64
```

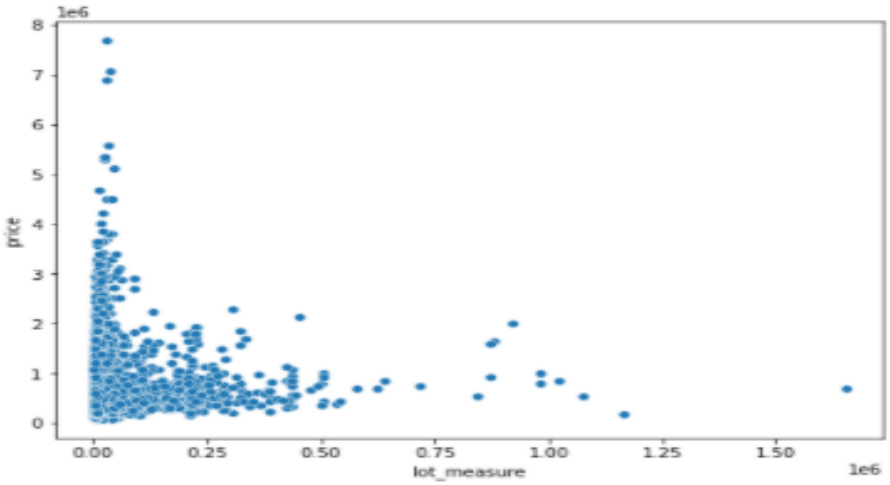
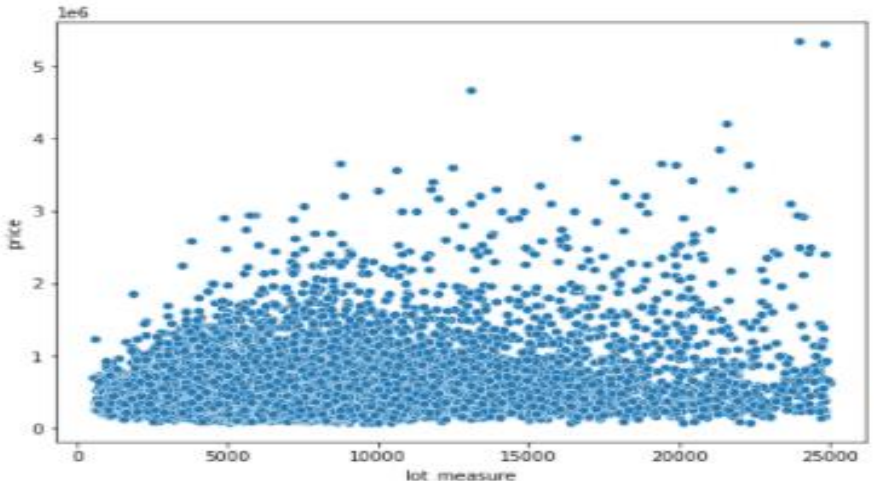


Figure 19: Bivariate analysis for lot measure

- There doesnt seem to be no relation between lot_measure and price trend
- lot_measure - there seems to be no relation between lot_measure and price
- lot_measure - data value range is very large so breaking it get better view.

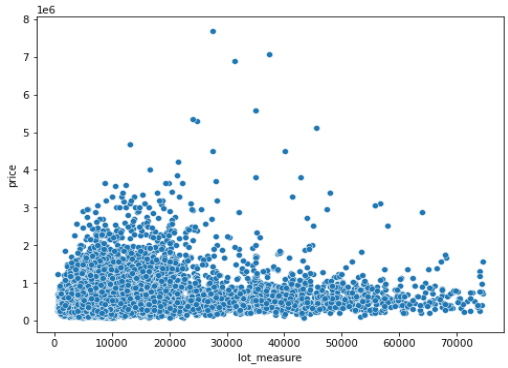
lot_measure <25000

```
count    19675.000000
mean      7760.899212
std       4250.035512
min        520.000000
25%       4995.000000
50%       7254.000000
75%       9620.000000
max      24969.000000
Name: lot_measure, dtype: float64
```



- Almost 95% of the houses have <25000 lot_measure. But there is no clear trend between lot_measure and price

lot_measure >100000 - price increases with increase in living measure



g) Analyzing Bivariate for Feature: ceil

ceil	mean	median	size
1.0	4.427711e+05	390000	10719
1.5	5.593744e+05	525000	1905
2.0	6.491210e+05	543250	8210
2.5	1.061021e+06	799200	161
3.0	5.831248e+05	490500	610
3.5	9.339375e+05	534500	8

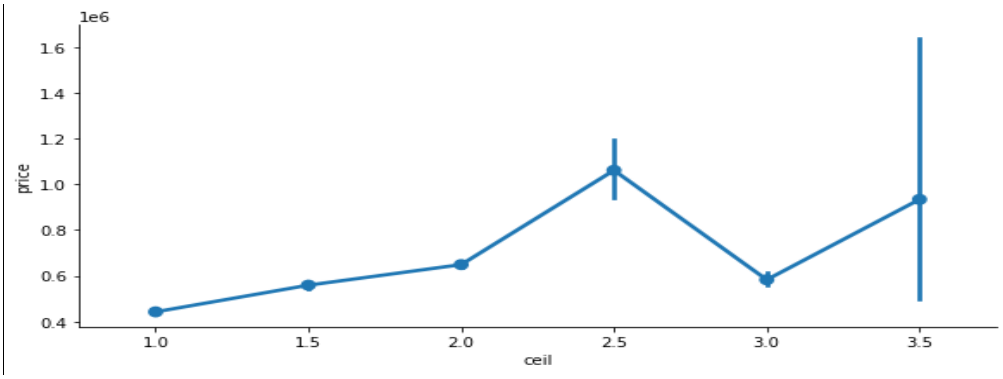


Figure 20: Bivariate analysis for ceil

- ceil - median price increases initially and then falls
- There is some slight upward trend in price with the ceil

h) Analyzing Bivariate for Feature: coast

	living_measure		price	
	median	mean	median	mean
coast				
0.0	1910.0	2071.699184	450000	5.317155e+05
1.0	2830.0	3166.465839	1410000	1.668301e+06

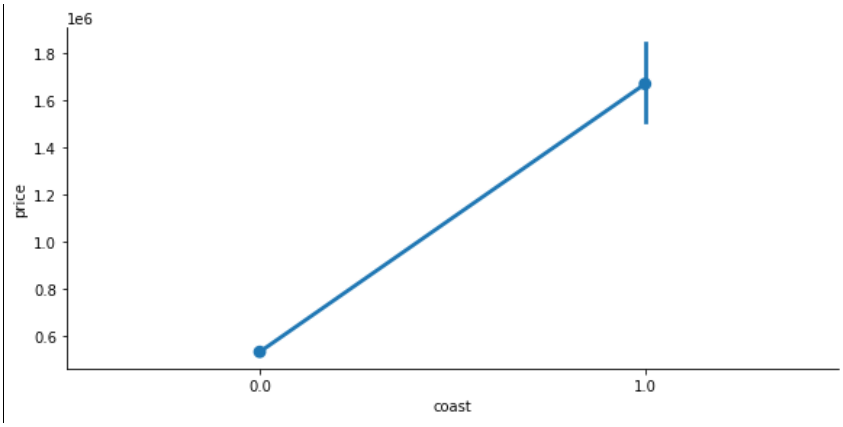


Figure 21: Bivariate analysis for coast

- coast - mean and median of waterfront view is high however such houses are very small in compare to non-waterfront
- Also, living_measure mean and median is greater for waterfront house.
- The house properties with water_front tend to have higher price compared to that of non-water_front properties

i) Analyzing Bivariate for Feature: sight

	price			living_measure		
	mean	median	size	mean	median	size
sight						
0.0	4.965541e+05	432500	19437	1997.560344	1850.0	19437
1.0	8.125186e+05	690944	332	2568.960843	2420.0	332
2.0	7.918609e+05	675000	959	2654.433647	2460.0	959
3.0	9.724684e+05	802500	510	3018.564706	2840.0	510
4.0	1.466554e+06	1190000	318	3354.433962	3070.0	318

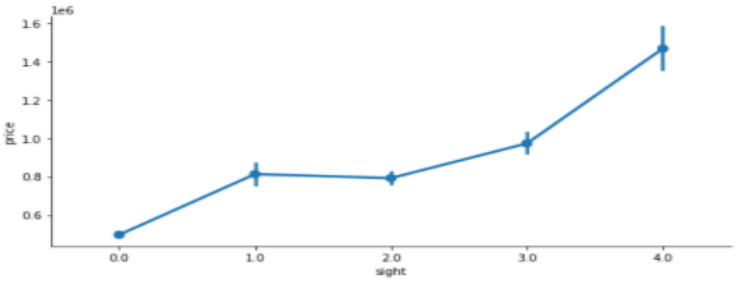
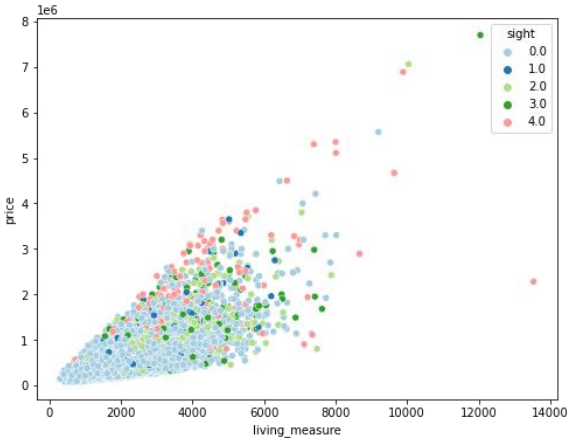


Figure 22: Bivariate analysis for sight

- sight - have outliers. The house sighted more have high price (mean and median) and have large living area as well.
- Properties with higher price have more no.of sights compared to that of houses with lower price



- Sight - Viewed in relation with price and living_measure
- Costlier houses with large living area are sighted more.
- The above graph also justify that: Properties with higher price have more no.of sights compared to that of houses with lower price

j) Analyzing Bivariate for Feature: condition

	price			living_measure		
	mean	median	size	mean	median	size
condition						
1.0	334431.666667	262500	30	1216.000000	1000.0	30
2.0	326423.327485	279000	171	1414.152047	1330.0	171
3.0	542364.148048	451000	14063	2148.725559	1970.0	14063
4.0	520643.219275	440000	5655	1950.395114	1820.0	5655
5.0	612515.489965	525444	1694	2022.696217	1880.0	1694

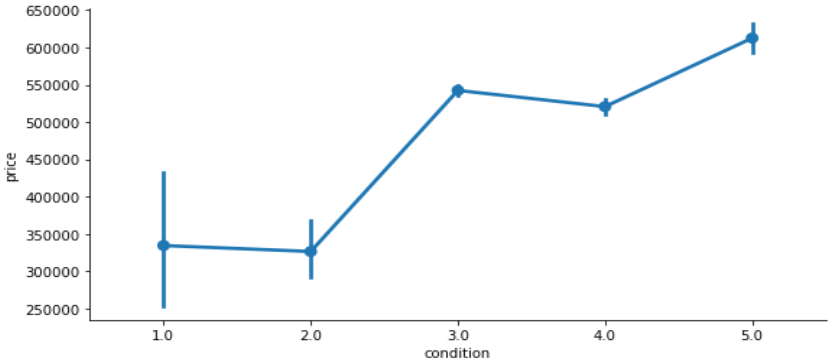
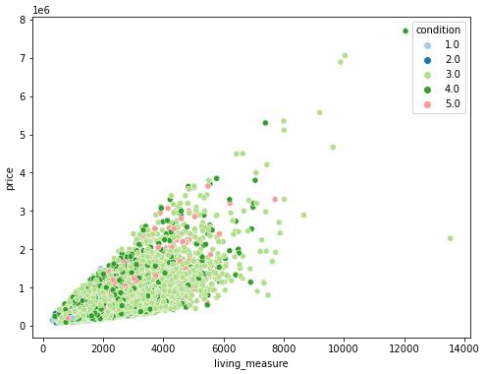


Figure 23: Bivariate analysis for condition

- condition - as the condition rating increases its price and living measure mean and median also increases.
- The price of the house increases with condition rating of the house



- condition - Viewed in relation with price and living_measure. Most houses are rated as 3 or more.
- We can see some outliers as well
- So we found out that smaller houses are in better condition and better condition houses are having higher prices

k) Analyzing Bivariate for Feature: quality

	price			living_measure		
	mean	median	size	mean	median	size
quality						
1.0	1.420000e+05	142000.0	1	290.000000	290.0	1
3.0	2.056667e+05	262000.0	3	596.666667	600.0	3
4.0	2.143810e+05	205000.0	29	660.482759	660.0	29
5.0	2.485240e+05	228700.0	242	983.326446	905.0	242
6.0	3.019166e+05	275276.5	2038	1191.743741	1120.0	2038
7.0	4.025933e+05	375000.0	8981	1689.603944	1630.0	8981
8.0	5.429310e+05	510000.0	6067	2184.397624	2150.0	6067
9.0	7.737382e+05	720000.0	2615	2868.404598	2820.0	2615
10.0	1.072347e+06	914327.0	1134	3520.299824	3450.0	1134
11.0	1.497792e+06	1280000.0	399	4395.448622	4260.0	399
12.0	2.192500e+06	1820000.0	90	5471.588889	4965.0	90
13.0	3.710769e+06	2980000.0	13	7483.076923	7100.0	13

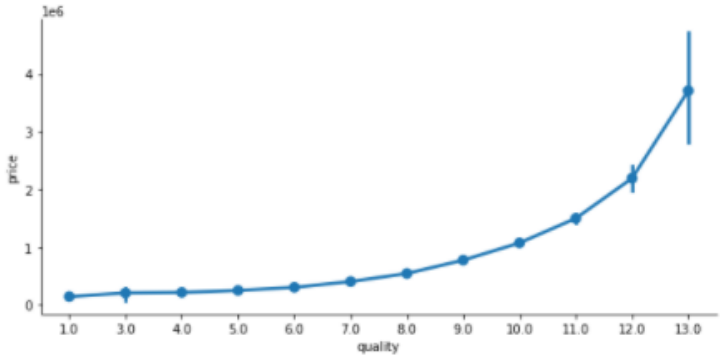
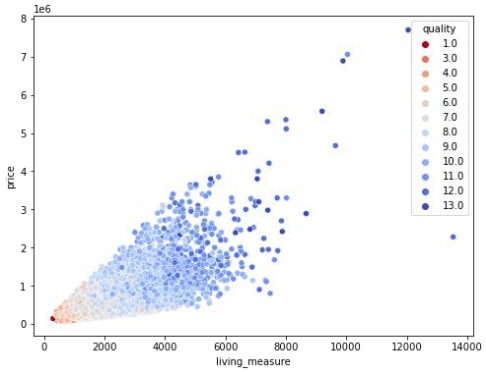


Figure 24: Bivariate analysis for quality

- quality - with grade increase price and living_measure increase (mean and median)
- There is clear increase in price of the house with higher rating on quality



- quality - Viewed in relation with price and living_measure. Most houses are graded as 6 or more.
- We can see some outliers as well

l) Analyzing Bivariate for Feature: ceil_measure

```
count    21612.000000
mean      1788.366556
std        828.102535
min        290.000000
25%       1190.000000
50%       1560.000000
75%       2210.000000
max       9410.000000
Name: ceil_measure, dtype: float64
```

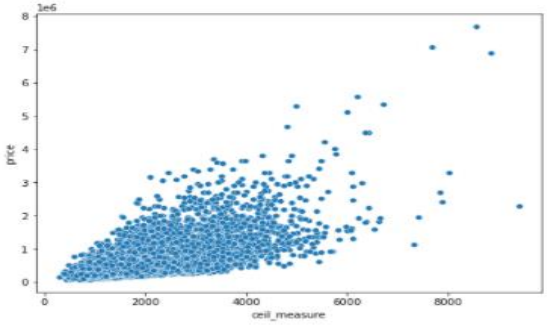


Figure 25: Bivariate analysis for ceil_measure

- ceil_measure - price increases with increase in ceil measure
- There is upward trend in price with ceil_measure

m) Analyzing Bivariate for Feature: basement

```
count    21612.000000
mean       291.522534
std        442.580840
min          0.000000
25%          0.000000
50%          0.000000
75%        560.000000
max       4820.000000
Name: basement, dtype: float64
```

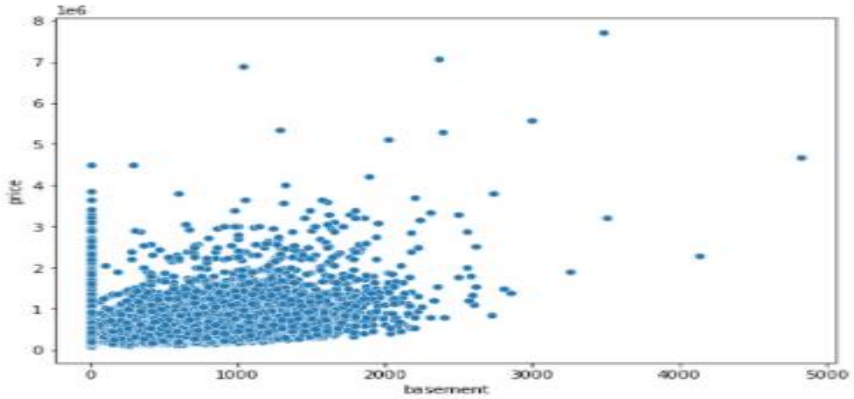
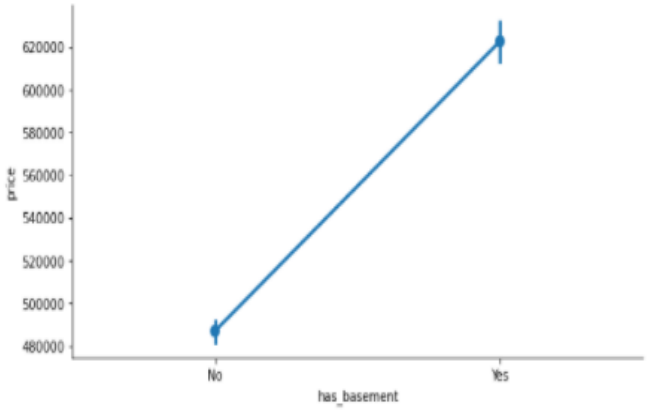


Figure 26: Bivariate analysis for basement

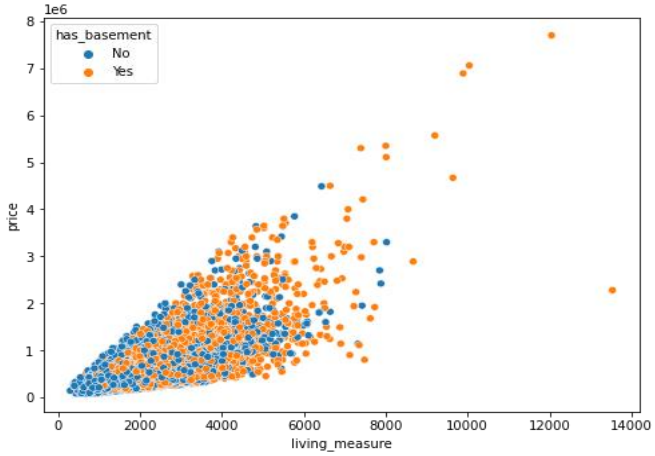
- basement - price increases with increase in ceil measure
- We will create the categorical variable for basement 'has_basement' for houses with basement and no basement.This categorical variable will be used for further analysis.

Binning Basement to analyse data

	price			living_measure		
	mean	median	size	mean	median	size
has_basement						
No	486957.543010	411500	13125	1928.880043	1740.0	13125
Yes	622518.174384	515000	8487	2313.246758	2100.0	8487



- basement - after binning, data shows with basement houses are costlier and have higher
- The houses with basement has better price compared to that of houses without basement.

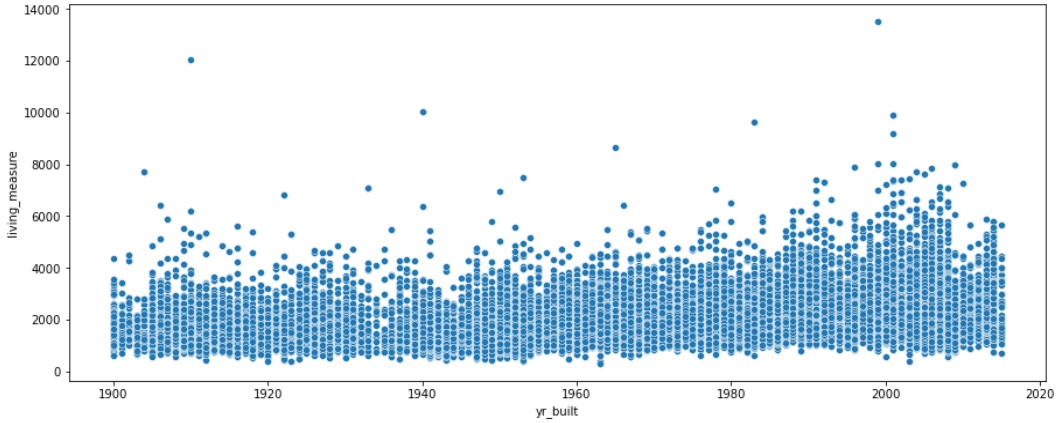


- basement - have higher price & living measure

yr_built - outliers can be seen easily

yr_built	mean	median	size
1900.0	581536.632184	549000.0	87
1901.0	557108.344828	550000.0	29
1902.0	673192.592593	624000.0	27
1903.0	480958.195652	461000.0	46
1904.0	583867.755556	478000.0	45
...	...	...	...
2011.0	544648.384615	440000.0	130
2012.0	527436.982353	448475.0	170
2013.0	678599.582090	565000.0	201
2014.0	677006.203833	590962.5	574
2015.0	762970.162162	628000.0	37

116 rows × 3 columns



We will create new variable: Houselandratio - This is proportion of living area in the total area of the house. We will explore the trend of price against this houselandratio.

- HouseLandRatio - Computing new variable as ratio of living_measure/total_area
- Significes - Land used for construction of house

```
12235    16.0
14791    35.0
1742     17.0
17829    27.0
14810    17.0
Name: HouseLandRatio, dtype: float64
```

n) Analyzing Bivariate for Feature: yr_renovated

yr_renovated	mean	median	size
1934	459950.000000	459950.0	1
1940	378400.000000	378400.0	2
1944	521000.000000	521000.0	1
1945	398666.666667	375000.0	3
1946	351137.500000	351137.5	2
...	...	...	...
2011	607498.153846	577000.0	13
2012	625181.818182	515000.0	11
2013	664960.810811	560000.0	37
2014	655030.098901	575000.0	91
2015	659156.250000	651000.0	16

69 rows × 3 columns

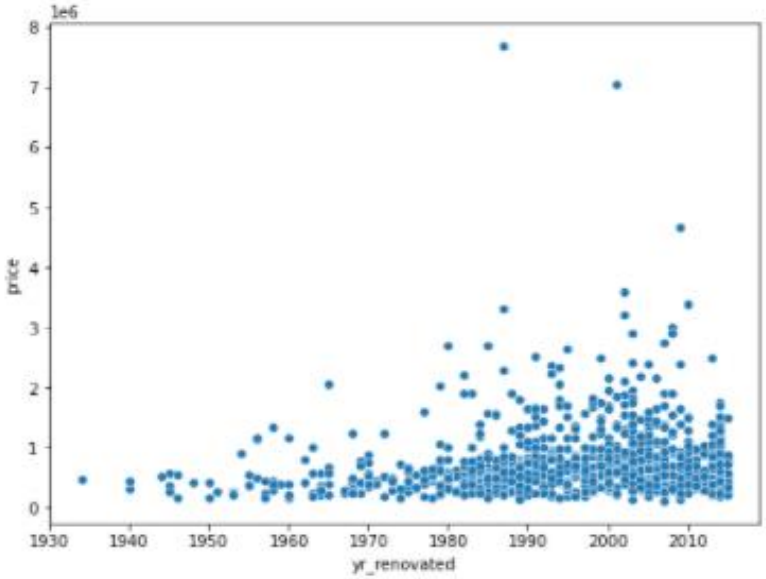


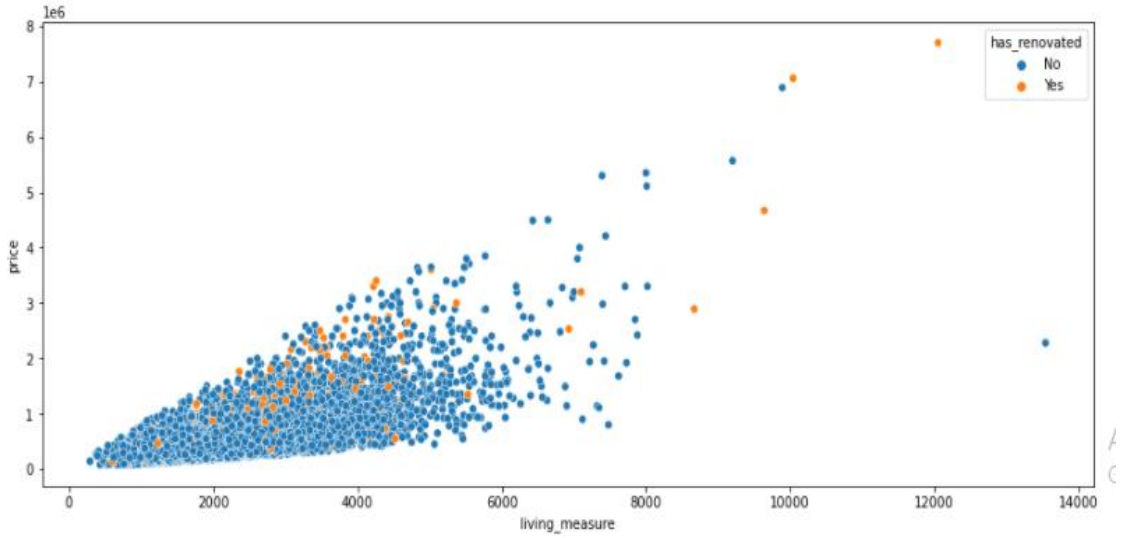
Figure 27: Bivariate analysis for yr_renovated

- So most houses are renovated after 1980's.
- We will create new categorical variable 'has_renovated' to categorize the property as renovated and non-renovated. For further analysis we will use this categorical variable.

Let's try to group yr_renovated. Binning to analyze data

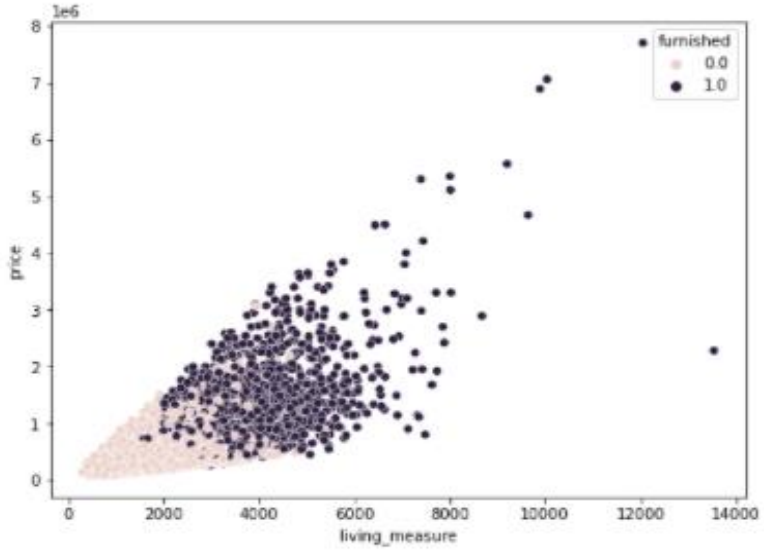
- has_renovated - renovated have higher mean and median, however it does not confirm if the prices of house renovated
- HouseLandRatio - Renovated house utilized more land area for construction of house
- Renovated properties have higher price than others with same living measure space.

	price			HouseLandRatio		
	mean	median	size	mean	median	size
has_renovated						
No	530447.958597	448000	20699	22.110526	20.0	20699
Yes	760628.777899	600000	914	22.320920	21.0	914



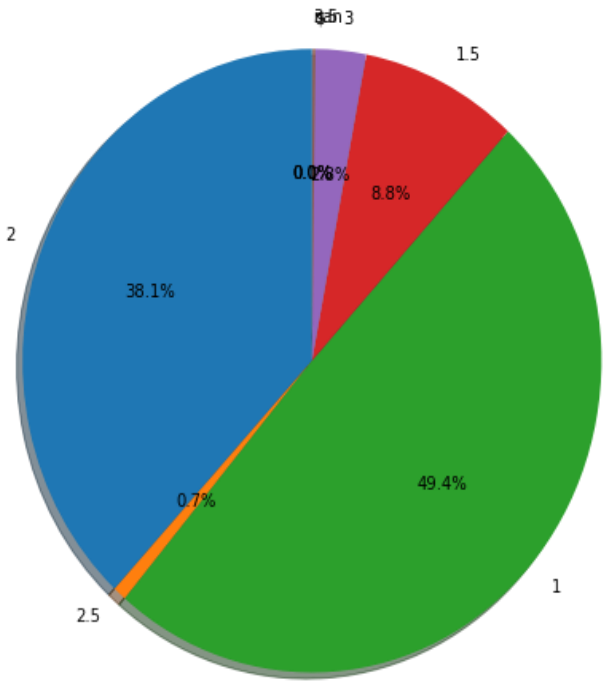
p) Analyzing Bivariate for Feature: furnished

	price			living_measure		
	mean	median	size	mean	median	size
furnished						
0.0	437277.805110	401000	17338	1792.083574	1720.0	17338
1.0	960565.753179	810000	4246	3255.845802	3110.0	4246

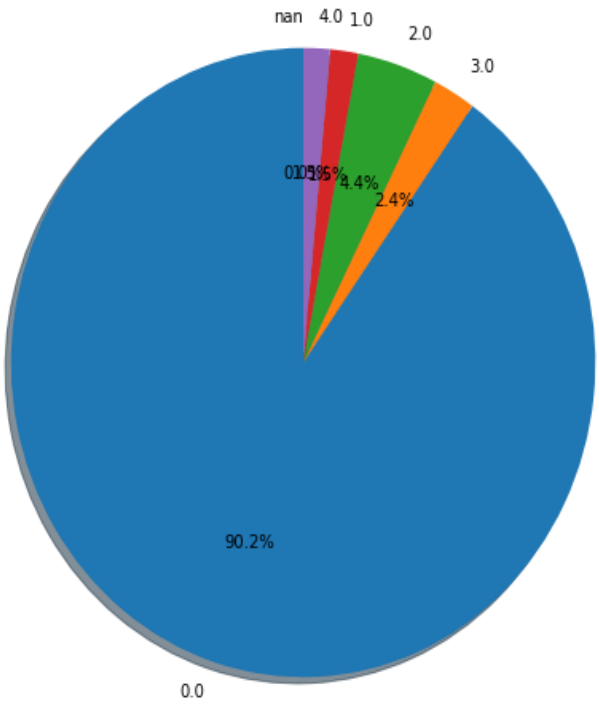


- furnished - Furnished has higher price value and has greater living_measure
- Furnished houses have higher price than that of the Non-furnished houses

In terms of Number of floors, most of the houses are of **single (1) & double (2)** floor
In terms of Sight Viewed, almost 90% of the time sight is not viewed. And maximum number of sight viewed is 4 times.



Number of Floors



Sight Viewed

MULTIVARIATE ANALYSIS

Let’s see corelatoin between the different features:

	cid	price	room_bed	room_bath	living_measure	lot_measure	ceil	coast	sight	condition	quality	ceil_measure
cid	1.000000	-0.016797	0.001774	0.005660	-0.012130	-0.132043	0.019379	-0.004184	0.012056	-0.024273	0.008140	-0.010821
price	-0.016797	1.000000	0.308040	0.525179	0.702149	0.089733	0.256385	0.266042	0.397522	0.035556	0.667477	0.605593
room_bed	0.001774	0.308040	1.000000	0.515778	0.576447	0.032051	0.174255	-0.006256	0.078367	0.027469	0.356740	0.477708
room_bath	0.005660	0.525179	0.515778	1.000000	0.754617	0.088055	0.497986	0.062128	0.186939	-0.126276	0.665037	0.685556
living_measure	-0.012130	0.702149	0.576447	0.754617	1.000000	0.172958	0.352780	0.102531	0.284927	-0.058891	0.762723	0.876653
lot_measure	-0.132043	0.089733	0.032051	0.088055	0.172958	1.000000	-0.005735	0.021301	0.075144	-0.008594	0.113869	0.183656
ceil	0.019379	0.256385	0.174255	0.497986	0.352780	-0.005735	1.000000	0.024136	0.028885	-0.262111	0.457335	0.522214
coast	-0.004184	0.266042	-0.006256	0.062128	0.102531	0.021301	0.024136	1.000000	0.400203	0.017683	0.082513	0.071329
sight	0.012056	0.397522	0.078367	0.186939	0.284927	0.075144	0.028885	0.400203	1.000000	0.046042	0.251095	0.167572
condition	-0.024273	0.035556	0.027469	-0.126276	-0.058891	-0.008594	-0.262111	0.017683	0.046042	1.000000	-0.144768	-0.158338
quality	0.008140	0.667477	0.356740	0.665037	0.762723	0.113869	0.457335	0.082513	0.251095	-0.144768	1.000000	0.755923
ceil_measure	-0.010821	0.605593	0.477708	0.685556	0.876653	0.183656	0.522214	0.071329	0.167572	-0.158338	0.755923	1.000000
basement	-0.005174	0.323825	0.302569	0.283389	0.435035	0.015400	-0.244899	0.079237	0.277769	0.173720	0.168403	-0.051925
yr_built	0.021586	0.053541	0.154317	0.506326	0.317193	0.052821	0.486468	-0.025917	-0.052942	-0.360446	0.446255	0.423001
yr_renovated	-0.016907	0.126442	0.018843	0.050194	0.055253	0.007782	0.005351	0.088358	0.103078	-0.060494	0.014417	0.023291
zipcode	-0.008224	-0.053168	-0.152805	-0.204149	-0.199646	-0.129423	-0.058419	0.030289	0.084666	0.003040	-0.184854	-0.261171
lat	-0.001891	0.306919	-0.009125	0.024505	0.052764	-0.086142	0.050372	-0.014139	0.006160	-0.015471	0.114103	-0.000784
living_measure15	-0.002860	0.585186	0.390823	0.568001	0.756188	0.145653	0.278147	0.086096	0.280976	-0.094068	0.713114	0.731558
lot_measure15	-0.138750	0.082603	0.029484	0.087096	0.183548	0.717727	-0.011423	0.030706	0.073020	-0.003027	0.119360	0.194246
furnished	-0.010013	0.566040	0.259316	0.485162	0.633157	0.119326	0.347411	0.069526	0.219995	-0.122695	0.788717	0.652618

basement	yr_built	yr_renovated	zipcode	lat	living_measure15	lot_measure15	furnished
-0.005174	0.021586	-0.016907	-0.008224	-0.001891	-0.002860	-0.138750	-0.010013
0.323825	0.053541	0.126442	-0.053168	0.306919	0.585186	0.082603	0.566040
0.302569	0.154317	0.018843	-0.152805	-0.009125	0.390823	0.029484	0.259316
0.283389	0.506326	0.050194	-0.204149	0.024505	0.568001	0.087096	0.485162
0.435035	0.317193	0.055253	-0.199646	0.052764	0.756188	0.183548	0.633157
0.015400	0.052821	0.007782	-0.129423	-0.086142	0.145653	0.717727	0.119326
-0.244899	0.486468	0.005351	-0.058419	0.050372	0.278147	-0.011423	0.347411
0.079237	-0.025917	0.088358	0.030289	-0.014139	0.086096	0.030706	0.069526
0.277769	-0.052942	0.103078	0.084666	0.006160	0.280976	0.073020	0.219995
0.173720	-0.360446	-0.060494	0.003040	-0.015471	-0.094068	-0.003027	-0.122695
0.168403	0.446255	0.014417	-0.184854	0.114103	0.713114	0.119360	0.788717
-0.051925	0.423001	0.023291	-0.261171	-0.000784	0.731558	0.194246	0.652616
1.000000	-0.133172	0.071317	0.074820	0.110508	0.200068	0.017539	0.092624
-0.133172	1.000000	-0.223782	-0.346118	-0.148182	0.325979	0.070654	0.304756
0.071317	-0.223782	1.000000	0.064357	0.029398	-0.002053	0.007950	0.017330
0.074820	-0.346118	0.064357	1.000000	0.267048	-0.279263	-0.147479	-0.138747
0.110508	-0.148182	0.029398	0.267048	1.000000	0.048822	-0.085936	0.080947
0.200068	0.325979	-0.002053	-0.279263	0.048822	1.000000	0.184830	0.620313
0.017539	0.070654	0.007950	-0.147479	-0.085936	0.184830	1.000000	0.129689
0.092624	0.304756	0.017330	-0.138747	0.080947	0.620313	0.129689	1.000000

HEATMAP

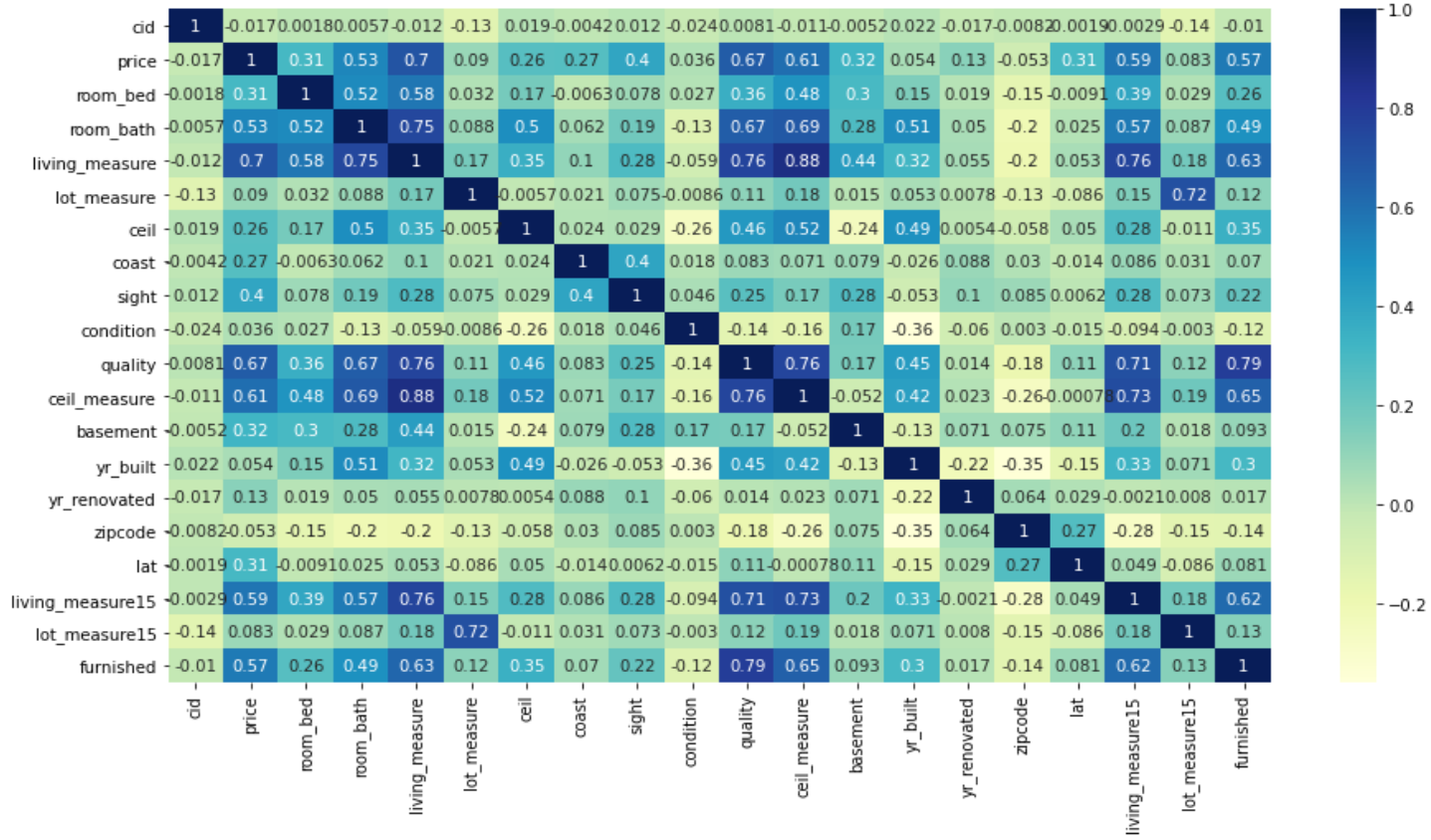


Figure 28: Heatmap

We have linear relationships in below featues as we got to know from above matrix

1. **price**: room_bath, living_measure, quality, living_measure15, furnished
2. **living_measure**: price, room_bath. So we can consider dropping 'room_bath' variable.
3. **quality**: price, room_bath, living_measure
4. **ceil_measure**: price, room_bath, living_measure, quality
5. **living_measure15**: price, living_measure, quality. So we can consider dropping living_measure15 as well. As it's giving same info as living_measure.
6. **lot_measure15**: lot_measure. Therefore, we can consider dropping lot_measure15, as it's giving same info.
7. **furnished**: quality
8. **total_area**: lot_measure, lot_measure15. Therefore, we can consider dropping total_area feature as well. As it's giving same info as lot_measure.

IS THE DATA UNBALANCED? IF SO, WHAT CAN BE DONE? PLEASE EXPLAIN IN THE CONTEXT OF THE BUSINESS.

- Checking data balance is very critical for Classification problems.
- In case of Regression, if a variable is significant then there should be adequate rows for all values of that variable.

ANY BUSINESS INSIGHTS USING CLUSTERING

- Clustering is generally done on Unsupervised data.
- We will use features selection by considering the feature importance fucntion in individual models. Thus to extract important features using PCA.

MISSING VALUES TREATMENT

Important questions when thinking about missing data:

- How prevalent is the missing data?
- Is missing data random or does it have a pattern?

The answer to these questions is important for practical reasons because missing data can imply a reduction of the sample size. This can prevent us from proceeding with the analysis. Moreover, from a substantive perspective, we need to ensure that the missing data process is not biased and hidding an inconvenient truth.

Sometimes, in a dataset we will have missing values such as NaN or empty string in a cell. We need to take care of these missing values so that our machine learning model doesn’t break. To handle missing values, there are three approaches followed.

Replace the missing value with a large negative number (e.g. -999).
Replace the missing value with mean of the column.
Replace the missing value with median of the column.

To find if a column in our dataset has missing values, we have used pd.isnull(df).any() which returns a boolean for each column in the dataset that tells if the column contains any missing value. In this dataset, there are missing values which have been treated with the help of SimpleImputer that helped to replace the missing values with mean of the column.
Ignore has_basement as it been made only to do the binning of basement column and to analyze data.

```
cid                   0
dayhours              0
price                 0
room_bed              0
room_bath             0
living_measure        0
lot_measure           0
ceil                  0
coast                 0
sight                 0
condition             0
quality               0
ceil_measure          0
basement              0
yr_built              0
yr_renovated          0
zipcode               0
lat                   0
long                  0
living_measure15      0
lot_measure15         0
furnished             0
total_area            0
month_year            0
has_basement          1
HouseLandRatio        0
has_renovated         0
dtype: int64
```

OUTLIER_TREATMENT

We have seen **outliers** for columns room_bath(33 bed), living_measure, lot_measure, ceil_measure and Basement and going to treat outliers and drop it for these columns only.

In summary, after treating outliers, we have lost about 15% of the data. We will analyse the impact of this data loss during the model evaluation.

Let's see the feature/columns and drop the unneccessary features

```
Index(['cid', 'dayhours', 'price', 'room_bed', 'room_bath', 'living_measure',
       'lot_measure', 'ceil', 'coast', 'sight', 'condition', 'quality',
       'ceil_measure', 'basement', 'yr_built', 'yr_renovated', 'zipcode',
       'lat', 'long', 'living_measure15', 'lot_measure15', 'furnished',
       'total_area', 'month_year', 'has_basement', 'HouseLandRatio',
       'has_renovated'],
      dtype='object')
```

As we already have this information in other features. We will drop the unwanted columns from new copied dataframe instance : cid,dayhours,yr_renovated,zipcode,lat,long using drop function.

Final columns after dropping unwanted columns from new copied dataframe:

```
Index(['price', 'room_bed', 'room_bath', 'living_measure', 'lot_measure',
       'ceil', 'coast', 'sight', 'condition', 'quality', 'ceil_measure',
       'basement', 'yr_built', 'living_measure15', 'lot_measure15',
       'furnished', 'total_area', 'month_year', 'has_basement',
       'HouseLandRatio', 'has_renovated'],
      dtype='object')
```

Creating dummies for categorical variables: 'room_bed', 'room_bath', 'ceil', 'coast', 'sight', 'condition', 'quality', 'furnished','City', 'has_basement', 'has_renovated'

```
Index(['price', 'living_measure', 'lot_measure', 'ceil_measure', 'basement',
       'yr_built', 'living_measure15', 'lot_measure15', 'total_area',
       'month_year', 'HouseLandRatio', 'room_bed_1.0', 'room_bed_2.0',
       'room_bed_3.0', 'room_bed_4.0', 'room_bed_5.0', 'room_bed_6.0',
       'room_bed_7.0', 'room_bed_8.0', 'room_bed_9.0', 'room_bed_10.0',
       'room_bed_11.0', 'room_bath_0.5', 'room_bath_0.75', 'room_bath_1.0',
       'room_bath_1.25', 'room_bath_1.5', 'room_bath_1.75', 'room_bath_2.0',
       'room_bath_2.25', 'room_bath_2.5', 'room_bath_2.75', 'room_bath_3.0',
       'room_bath_3.25', 'room_bath_3.5', 'room_bath_3.75', 'room_bath_4.0',
       'room_bath_4.25', 'room_bath_4.5', 'room_bath_4.75', 'room_bath_5.0',
       'room_bath_5.25', 'room_bath_5.75', 'ceil_1.5', 'ceil_2.0', 'ceil_2.5',
       'ceil_3.0', 'ceil_3.5', 'coast_1.0', 'sight_1.0', 'sight_2.0',
       'sight_3.0', 'sight_4.0', 'condition_2.0', 'condition_3.0',
       'condition_4.0', 'condition_5.0', 'quality_4.0', 'quality_5.0',
       'quality_6.0', 'quality_7.0', 'quality_8.0', 'quality_9.0',
       'quality_10.0', 'quality_11.0', 'quality_12.0', 'furnished_1.0',
       'has_basement_Yes', 'has_renovated_Yes'],
      dtype='object')
```

INSIGHTS FROM EDA

This section aims at giving certain business insights from the analysis done above.
This takes into account all the pointers that have been coded/ discussed/ elaborated above.

1. There is missing data in the dataset. Suggesting to avoid such instances in future for a proper analysis without dropping out any entry.
2. 2. There are outliers in the data. Suggesting to give a check for the reasons for the same. It's unusual to accept that a house with just 1.5 bedrooms can be priced at 1,20,00,000.
3. Living measure is the most essential variable that impacts the price of the house. Hence, while buying or selling or even while valuating a house, it's essential to see that how much square foot of area is covered under living measure of the house.
4. Lot measure is not a useful variable to determine the price of the house. It can be minutely noted during the valuation of any property.
5. Focus more on houses which are centrally located in the area. That's a lucrative place to sell the house. Not much of profit can be expected from houses on the coastlines.
6. A house with 3 bedrooms and 3 bathrooms will be the most lucrative offer to any prospective buyer.

MODEL BUILDING AND INTERPRETATION.

- **Build various models**
- **Test your predictive model against the test set using various appropriate performance metrics**

The following notebook presents a thought process of predicting a continuous variable through Machine Learning methods. More specifically, we want to predict house prices based on multiple features using regression analysis.

Before building the model we need to perform and understand few aspects:

1) Build a new dataframe for building models with engineered features.
2) We will split our dataset into a training set and testing set using sklearn train_test_split() in the ratio of 80:20.
   Train set- Use for learning and to fit the parameters of classifier.
   Testing Set - Use to assess the performance of fully specified classifier of finally tuned model.
   Validation Set- Use to time the parameters of classifier and helps in feature selection.
   - Helps to qualify performance
   - Holding test set to avoid peeking, locking it away until evaluation and learning is done and simply wishing to obtain an independent evaluation of final hypothesis or evaluation. Gives unbiased evaluation.
3) Price prediction is a supervised (labeled inputs and outputs) - regression problem (predicts continuous quality) which is going to use different machine learning algorithms.
4) We will train numerous regression models on the train data (e.g., multiple linear regression, lasso, ridge, KNN, Support vector regressor-SVR, Decision tree regressor-DT) and evaluate their performance using R-square, Root Mean Squared Error (RMSE), MSE, MAE on the test data

'dff' is the data frame which is ready for modeling.

	price	living_measure	lot_measure	ceil_measure	basement	yr_built	living_measure15	lot_measure15	total_area	month_year	...	quality_6.0	quality_7.0	quality_8.0	quality_9.0	quality_10.0	quality_11.0	quality_12.0	furnished_1.0	has_basement_Yes	has_renovated_Yes
12235	460000	1760.0	9055.0	1760.0	0.0	1985.0	2010.0	9383.0	10815.0	May-2014	...	0	1	0	0	0	0	0	0	0	0
14791	345600	2800.0	5120.0	2800.0	0.0	1903.0	1780.0	5120.0	7920.0	May-2014	...	0	0	0	1	0	0	0	1	0	1
1742	750000	2240.0	10578.0	1550.0	690.0	1923.0	1570.0	10578.0	12818.0	May-2014	...	0	0	1	0	0	0	0	0	1	0
17829	325000	2220.0	6049.0	2220.0	0.0	1990.0	1980.0	7226.0	8269.0	May-2014	...	0	0	1	0	0	0	0	0	0	0
14810	390000	2240.0	10800.0	2240.0	0.0	1996.0	1900.0	9900.0	13040.0	May-2014	...	0	0	1	0	0	0	0	0	0	0

5 rows × 69 columns

Price prediction is a linear regression problem(establishing a relationship between a dependent variable and one or more independent variables) , since we have to do continuous price prediction instead of logical operator type solution, so below is list of models that can be implemented.

REGRESSION MODELS

In this section, we will train numerous regression models on the train data (e.g., multiple linear regression, lasso, ridge, KNN, Support vector regressor-SVR, Decision tree regressor-DT) and **evaluate their performance using 3 error metric-Root Mean Squared Error (RMSE), MSE, MAE** on the test data as follow:

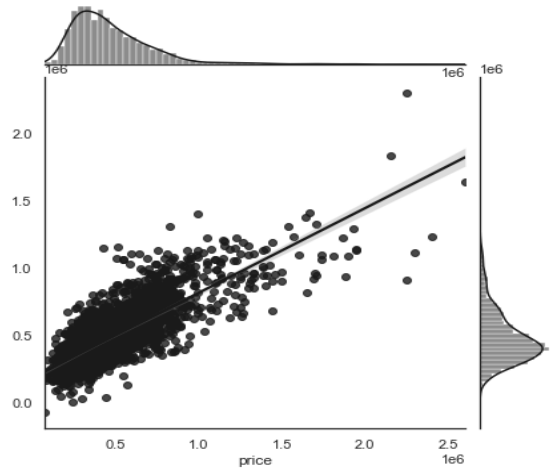
Model score and Deduction for each Model in a DataFrame

	Method	Val Score	RMSE_vl	MSE_vl	MAE_vl	train Score	RMSE_tr	MSE_tr	MAE_tr
0	Linear Reg Model1	0.622318	159467.070404	2.542975e+10	113309.839017	0.623767	155370.510001	2.414000e+10	111870.452000
0	Linear-Reg Lasso1	0.624855	158930.629167	2.525894e+10	113247.375142	0.623681	155388.292306	2.414552e+10	111894.555654
0	Linear-Reg Ridge1	0.625127	158873.013444	2.524063e+10	113297.137425	0.623263	155474.675112	2.417237e+10	111992.775261
0	knn1	0.393344	202106.119760	4.084688e+10	138782.844505	0.998802	8768.399424	7.688483e+07	782.211363
0	SVR1	-0.041009	264749.597492	7.009235e+10	178245.041225	-0.050705	259645.350463	6.741571e+10	179598.539492
0	SVR2	0.451004	192261.623225	3.696453e+10	131552.868372	0.447034	188360.333364	3.547962e+10	131617.700073
0	DT1	0.364733	206816.954719	4.277325e+10	138860.583874	0.998802	8768.399424	7.688483e+07	782.211363
0	DT2	0.556408	172822.416420	2.986759e+10	117148.399865	0.749671	126734.766320	1.606170e+10	93987.746486

➢ Interpretation of the models

Multiple Linear Regression

- Let's first predict house prices using simple (one input) linear regression.
- Linear Regression – is one of the most common models for regression problems.
- The linear regression model performed with scores 0.62 & .62 in training data set and validation data set respectively
- Shows homogeneity of variance.



Ridge ,lasso regression –

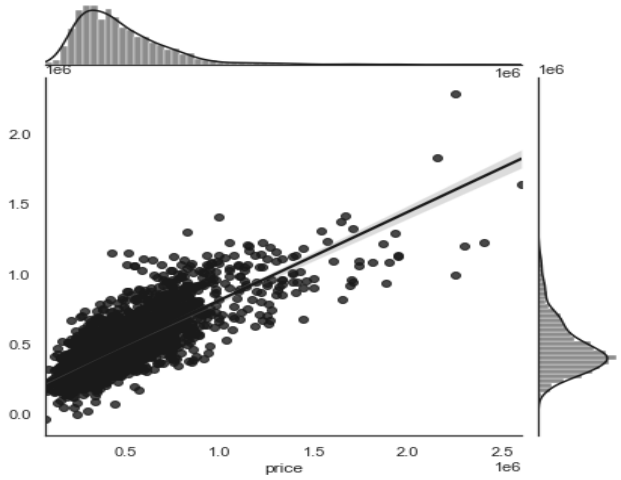
- There are two popular regularization techniques, each of them aiming at decreasing the size of the coefficients by penalizing coefficients.

LASSO REGRESSION:

- Lasso regression jointly shrinks coefficients to avoid over fitting, and implicitly performs feature selection by setting some coefficients exactly to 0 for sufficiently large penalty

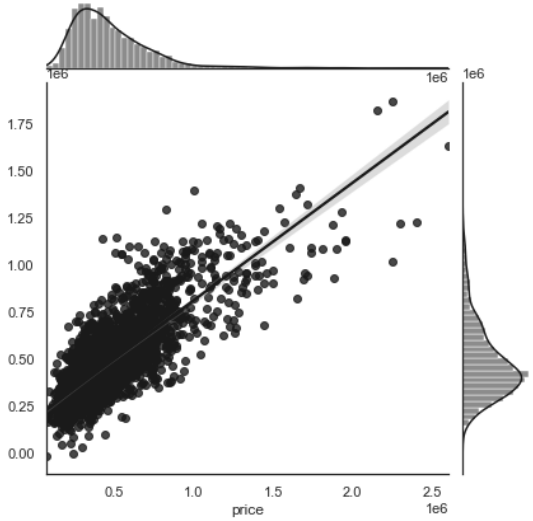
strength alpha (here called "L1_penalty"). In particular, lasso takes the RSS term of standard least squares and adds a 1-norm cost of the coefficients.

- The lasso linear regression model performed with scores 0.62 & 0.63 in training data set and validation data set respectively. The coefficeints of 1 variable in lasso model is almost '0', signifying that the variable with '0' coefficient can be dropped.
- Shows homogeneity of variance.



RIDGE REGRESSION:

- Ridge regression aims to avoid overfitting by adding a cost to the Residual Sum of Squares (RSS) term of standard least squares that depends on the 2-norm of the coefficients. The result is penalizing fits with large coefficients. The strength of this penalty, and thus the fit vs. model complexity, is controlled by a parameter alpha (here called "L2_penalty").
- The Ridge linear regression model performed with scores 0.62 & 0.63 in training data set and validation data set respectively. The coefficeints of variables in ridge model are all non-zero, indicating that none of the variables can be dropped.
- Shows homogeneity of variance.



In summary, Linear models have performed almost with similar results in both regularized model and non-regularized models

KNN Regressor:

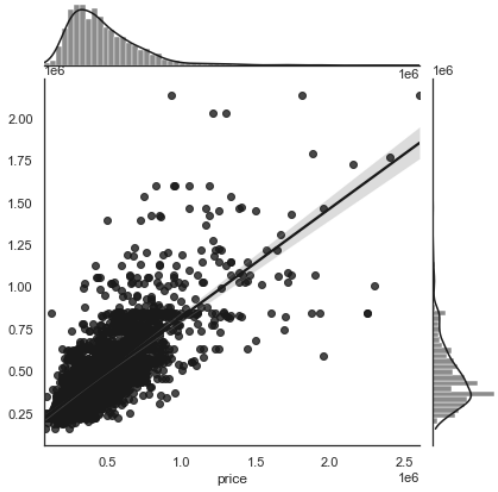
- The k-NN algorithm is used for estimating continuous variables. One such algorithm uses a weighted average of the k nearest neighbors, weighted by the inverse of their distance. It is the nonparamtric equivalent of ordinary least square regression.
- Though KNN regressor performed well in training set, the performance score in validation set is very less. This shows that the model is overfitted in training set.

Support vector regressor:

- The above negative scores in SVR1 model is due to non-learning of the model in the training set which results in non-performance in validation set.
- The SVR2 model with modified parameters has not performed well with just ~0.45 in both training and validation data sets

Decision Tree Regressor:

- Is one of the most commonly used, practical approaches for supervised learning. It can be used to solve both Regression and Classification
- Above performance of initial Decision tree- DT1 model shows over fit in training set with 0.99 score and low performance in validation set
- Above decision tree model with modified parameter DT2 has better performed on the training set and validation set compared to initial decision tree model. But overall decision tree has not performed well than linear regression models.
- Shows heteroscedasticity ("different scatter").



To evaluate the performance we used **3 error metric-Root Mean Squared Error (RMSE), MSE, MAE and R- Square**

- MAE – Easiest to understand. It is average error of absolute difference between the actual and predicted values in the dataset.
- MSE – More popular than MAE, punishes larger error, tends to be useful in real world. It is an average of square between original and predicted value.
- RMSE- It measures std of residuals or distance between predicted and actual value. Interprets in y-units.

  Lower value of MAE, MSE, RMSE, higher the accuracy.
- R-square – It is a proportion of variance. Better fit the model when the value is closer to 1 and away from 0.

Model Tuning

- **Ensemble modelling, wherever applicable**
- **Any other model tuning measures(if applicable)**
- **Interpretation of the most optimum and its implication on business**

Ensemble techniques:

- For more precise modeling we use these techniques.
- It uses more than one model

The techniques we use to ensemble are- gradient boosting regressor GB1, Bagging model BGG1 and random forest regressor RF1

	Method	Val Score	RMSE_vl	MSE_vl	MAE_vl	train Score	RMSE_tr	MSE_tr	MAE_tr
0	Linear Reg Model1	0.622318	159467.070404	2.542975e+10	113309.839017	0.623767	155370.510001	2.414000e+10	111870.452000
0	Linear-Reg Lasso1	0.624855	158930.629167	2.525894e+10	113247.375142	0.623681	155388.292306	2.414552e+10	111894.555654
0	Linear-Reg Ridge1	0.625127	158873.013444	2.524063e+10	113297.137425	0.623263	155474.675112	2.417237e+10	111992.775261
0	knn1	0.393344	202106.119760	4.084688e+10	138782.844505	0.998802	8768.399424	7.688483e+07	782.211363
0	SVR1	-0.041009	264749.597492	7.009235e+10	178245.041225	-0.050705	259645.350463	6.741571e+10	179598.539492
0	SVR2	0.451004	192261.623225	3.696453e+10	131552.868372	0.447034	188360.333364	3.547962e+10	131617.700073
0	DT1	0.364733	206816.954719	4.277325e+10	138860.583874	0.998802	8768.399424	7.688483e+07	782.211363
0	DT2	0.556408	172822.416420	2.986759e+10	117148.399865	0.749671	126734.766320	1.606170e+10	93987.746486
0	GB1	0.686828	145210.926188	2.108621e+10	105167.620490	0.732043	131121.047469	1.719273e+10	98049.360527
0	BGG1	0.666885	149763.125718	2.242899e+10	102657.068038	0.951959	55519.380196	3.082402e+09	38604.116644
0	RF1	0.672570	148479.712079	2.204622e+10	101631.598060	0.953282	54749.718709	2.997532e+09	38479.630604

Boosting and Bagging:

- It's another type machine learning model which works on intuition that best possible next model, when combined with previous models, minimizes the overall prediction error.
- Gradient boosting model has provided good scores in both training and validation sets
- Bagging model also performed well in training and validation sets.There seems to be overfitting in training set. We need to analyse further by hypertuning

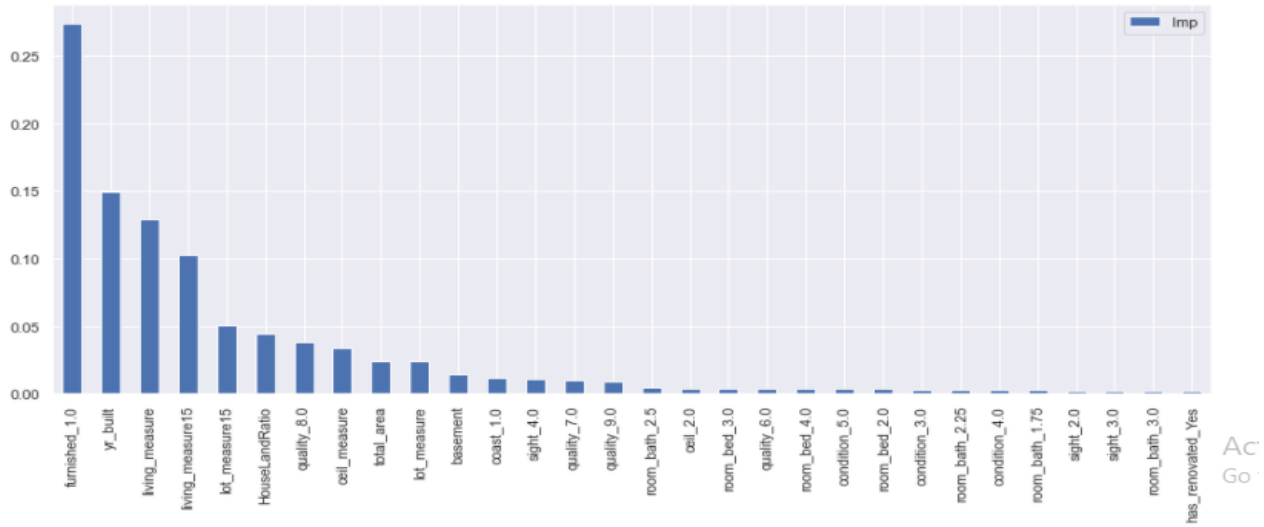
Random forest:

- supervised learning technique used for regression problem, it's another form of regression
- Random forest model has performed well in training and validation set. There is scope of further analysis on this model

Ensemble models: in summary ensemble models have performed well on training and validation sets. These models will be selected for further analysis with hypertuning and feature selection

FEATURE IMPORTANCE

```
First 20 feature importance:     Imp    94.805
dtype: float64
First 30 feature importance:     Imp    97.614
dtype: float64
```



Above are top 30 important features that account for 97.6% of variation in model. This need to be further analysed during hypertuning of the models for better scores

Model performance Summary:

Ensemble methods are performing better than linear models. Of all the ensemble models, Gradient boosting regressor is giving better R2 score. We identified top 30 features that are explaining the 95% variation in model (Random Forest). Will further hypertune the model to improve the model performance. Will further explore and evaluate the features while hyperturning the ensemble models

BUILDING FUNCTION/PIPELINE FOR MODELS

Use to automate machine learning workflow. It helps to avoid data leakage.

	Method	val score	RMSE_val	MSE_val	MAE_vl	train Score	RMSE_tr	MSE_tr	MAE_tr
0	LR	0.622318	159467.070404	2.542975e+10	113309.839017	0.623767	155370.510001	2.414000e+10	111870.452000
0	KNNR	0.393344	202106.119760	4.084688e+10	138782.844505	0.998802	8768.399424	7.688483e+07	782.211363
0	DTR	0.378366	204585.824427	4.185536e+10	139010.716604	0.998802	8768.399424	7.688483e+07	782.211363
0	GBR	0.686828	145210.926188	2.108621e+10	105167.620490	0.732043	131121.047469	1.719273e+10	98049.360527
0	BGR	0.666885	149763.125718	2.242899e+10	102657.068038	0.951959	55519.380196	3.082402e+09	38604.116644
0	RFR	0.670795	148881.633768	2.216574e+10	101849.940206	0.953375	54695.433317	2.991590e+09	38409.160816

Sequence of steps with pipeline function will run all the models and compile the scores in result_dff dataframe. We can see that the above 2 steps are concise instead of running individual models and compiling the scores as earlier.

We can clearly see gradient boosting is giving better result in comparison with other ensemble methods. Also the score of 0.73 on training set indicates no overfitting of the model.

FEATURE SELECTION (PCA)

- Now, we will explore the possibility of features reduction using PCA
- will drop the price column as it is the target variable
- Let's first transform the entire X (independent variable data) to zscores.
- We will create the PCA dimensions on this distribution.
- As PCA for Independent columns of Numerical types, let's pass numerical_cols

```
Covariance Matrix
%s [[ 1.00005467  0.20024458  0.84567294 ...  0.54463658  0.20077865
   0.05302665]
 [ 0.20024458  1.00005467  0.16635178 ...  0.08859049 -0.02984819
  -0.00571129]
 [ 0.84567294  0.16635178  1.00005467 ...  0.5752248  -0.27710506
   0.01804377]
 ...
 [ 0.54463658  0.08859049  0.5752248  ...  1.00005467 -0.04532651
   0.01975327]
 [ 0.20077865 -0.02984819 -0.27710506 ... -0.04532651  1.00005467
   0.04560111]
 [ 0.05302665 -0.00571129  0.01804377 ...  0.01975327  0.04560111
   1.00005467]]
```

- ***As we can see, near the value to 1, more the features related.***

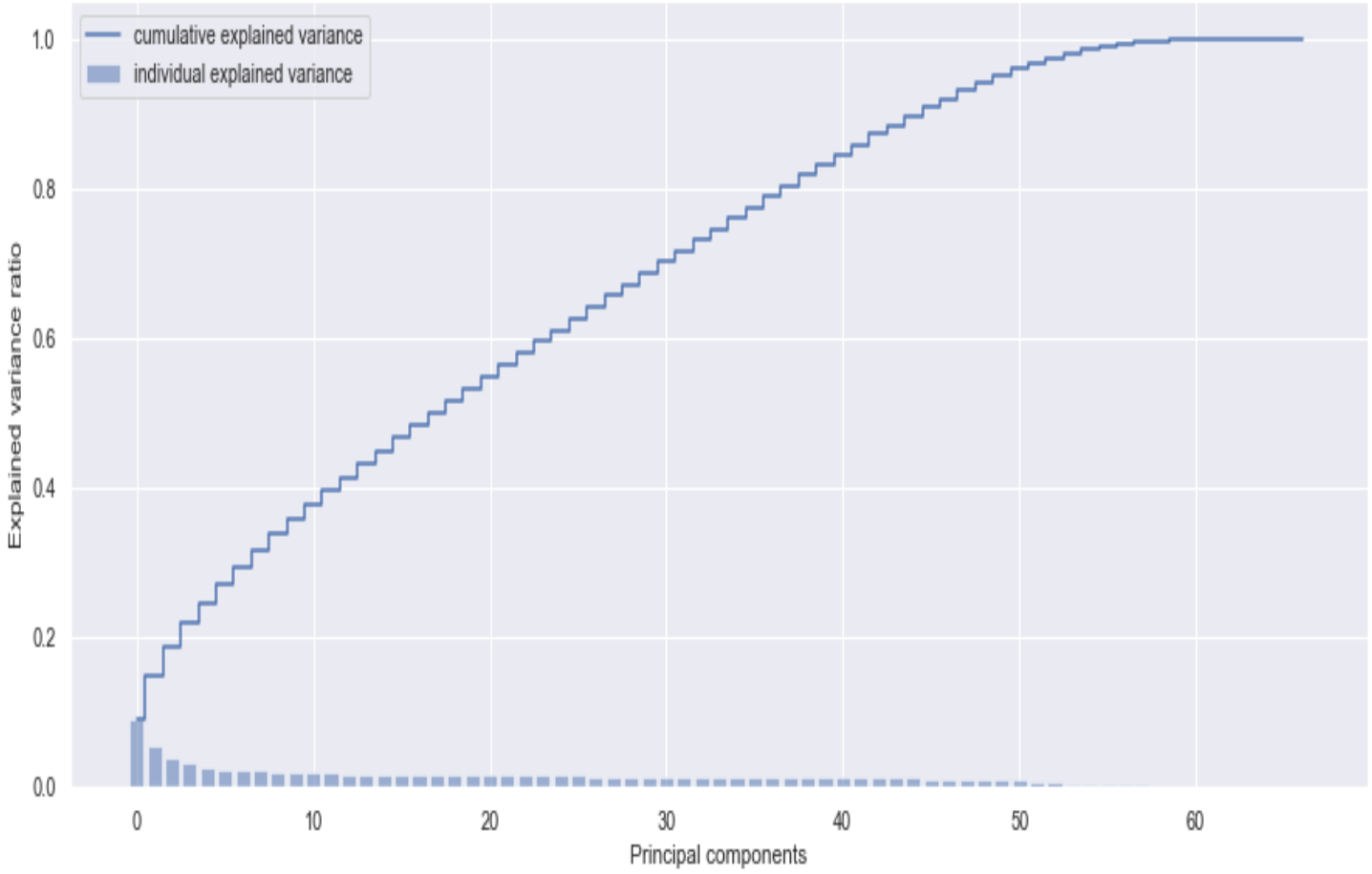
```
Eigen Vectors
%s [[-3.50987349e-01  1.30548696e-01 -1.64255703e-01 ... -2.81054251e-03
  -1.46079142e-02  5.53815512e-03]
 [-3.33915644e-02  4.36858894e-01  1.72644546e-01 ... -1.19860258e-02
  -5.73372582e-03  5.60014397e-05]
 [-3.56644158e-01  4.69324827e-02  1.05528323e-01 ... -2.95000846e-02
  -1.96573573e-02  2.39280946e-02]
 ...
 [-2.76544767e-01  5.37542794e-02  6.97850702e-02 ...  2.78705025e-02
   3.04105619e-02  2.05487408e-02]
 [ 1.01497118e-02  9.25984830e-02 -4.82904082e-01 ...  4.13743032e-02
   1.76136146e-02 -3.34194000e-02]
 [-3.94624088e-03  1.76739286e-02 -7.50377119e-02 ...  8.96965980e-02
   7.59174011e-02  9.98643699e-04]]

 Eigen Values
%s [6.05437729e+00 3.82121982e+00 2.73004536e+00 2.19636771e+00
 1.68619241e+00 1.66030525e+00 7.13849881e-02 1.16196651e-01
 1.24350819e-01 2.25561599e-03 4.52328491e-04 9.60490917e-04
 9.96176864e-04 7.69089850e-05 1.28531305e-04 2.02201936e-01
 2.31729628e-01 2.89314049e-01 3.57185341e-01 3.68890646e-01
 1.52904039e+00 4.18748357e-01 4.74714629e-01 1.47240243e+00
 1.45032043e+00 1.37191399e+00 6.02292265e-01 1.30109497e+00
 6.75168031e-01 1.26925371e+00 7.00736317e-01 7.39674712e-01
 1.20676361e+00 1.19163252e+00 7.84941529e-01 7.99977998e-01
 8.09658836e-01 8.26913124e-01 1.15912766e+00 1.14497505e+00
 8.87555466e-01 1.12396287e+00 1.11285281e+00 9.06248781e-01
 9.21274919e-01 9.24811710e-01 1.08379318e+00 1.07738666e+00
 1.07440786e+00 9.36893142e-01 1.06193249e+00 9.51981415e-01
 1.05183404e+00 1.04575883e+00 1.04013932e+00 9.64044502e-01
 9.70874881e-01 1.02933499e+00 1.02433312e+00 9.81893531e-01
 9.84095874e-01 1.01018535e+00 1.00724124e+00 9.90813048e-01
 1.00239067e+00 9.97855699e-01 9.95784279e-01]
```

- Let's Sort eigenvalues in descending order
- Make a set of (eigenvalue, eigenvector) pairs
- Sort the (eigenvalue, eigenvector) pairs from highest to lowest with respect to eigenvalue
- Extract the descending ordered eigenvalues and eigenvectors
- Let's confirm our sorting worked, print out eigenvalues

Eigenvalues in descending order:

[6.054377288502749, 3.8212198213610784, 2.730045359316942, 2.1963677135232094, 1.6861924114544478, 1.6603052456605476, 1.5290403928902079, 1.4724024338158581, 1.4503204349232623, 1.371913987749931, 1.3010949739444932, 1.2692537084741373, 1.2067636076554211, 1.1916325213085277, 1.1591276566822963, 1.1449750504487 02, 1.1239628730714506, 1.1128528142228786, 1.0837931756399743, 1.07738666106949, 1.0744078621547317, 1.0619324943554702, 1.051834037017902, 1.0457588342535022, 1.0401393194470583, 1.0293349890401253, 1.0243331196533327, 1.0101853540743237, 1.0072412374031956, 1.0023906698249423, 0.99785569861937, 0.9957842790870207, 0.9908130480718813, 0.9840958742731802, 0.9818935310626306, 0.9708748810224105, 0.9640445021915954, 0.951981415149707, 0.9368931415637474, 0.9248117104186362, 0.9212749188067385, 0.9062487806769346, 0.8875554655290976, 0.8269131240213816, 0.8096588362874728, 0.7999779983830945, 0.7849415287125 26, 0.7396747118963148, 0.7007363169350038, 0.6751680312392517, 0.6022922649008055, 0.4747146286898541, 0.4187483569108324, 0.368890645645291, 0.35718534135867, 0.2893140488034741 7, 0.23172962798940375, 0.20220193646979068, 0.12435081850153018, 0.11619665116503083, 0.07138498806666438, 0.0022556159857851214, 0.0009961768640302892, 0.0009604909165112093, 0.0004523284908327585, 0.0001285313048720 1523, 7.690898504305116e-05]

- An array of variance explained by each eigen vector... there will be 67 entries as there are 67 eigen vectors)
- There will be 90 entries with 67th entry cumulative reaching almost 100%



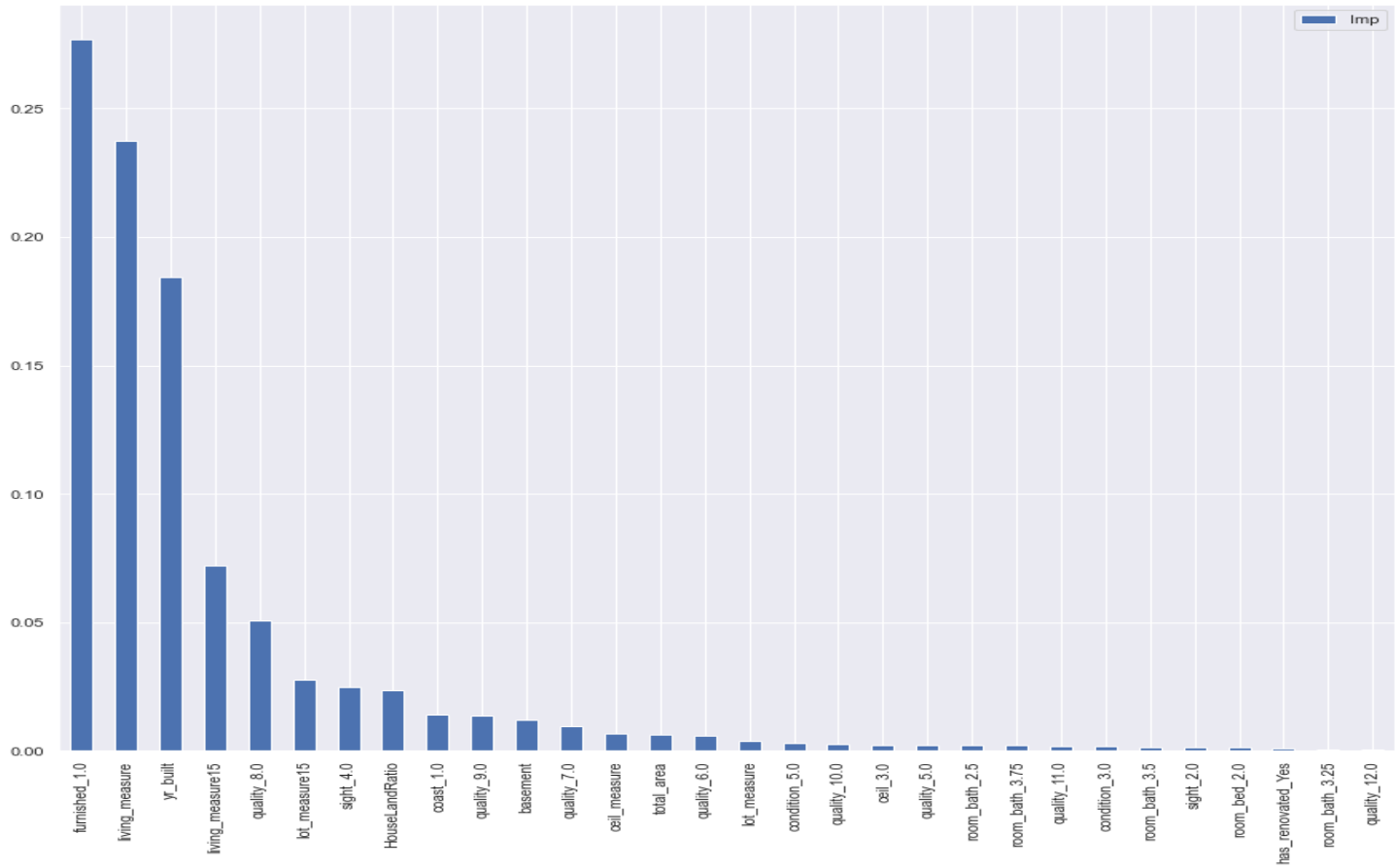
- 52 dimensions covering 97% variance in the data. So we can reduce to 52 dimension space

Now will recall the ensemble models from our initial run to check the feature selection using feature*imp* from individual models

- Will run function with ensemble models: Gradient boosting, Random forest, Bagging

A) Gradient boost model:

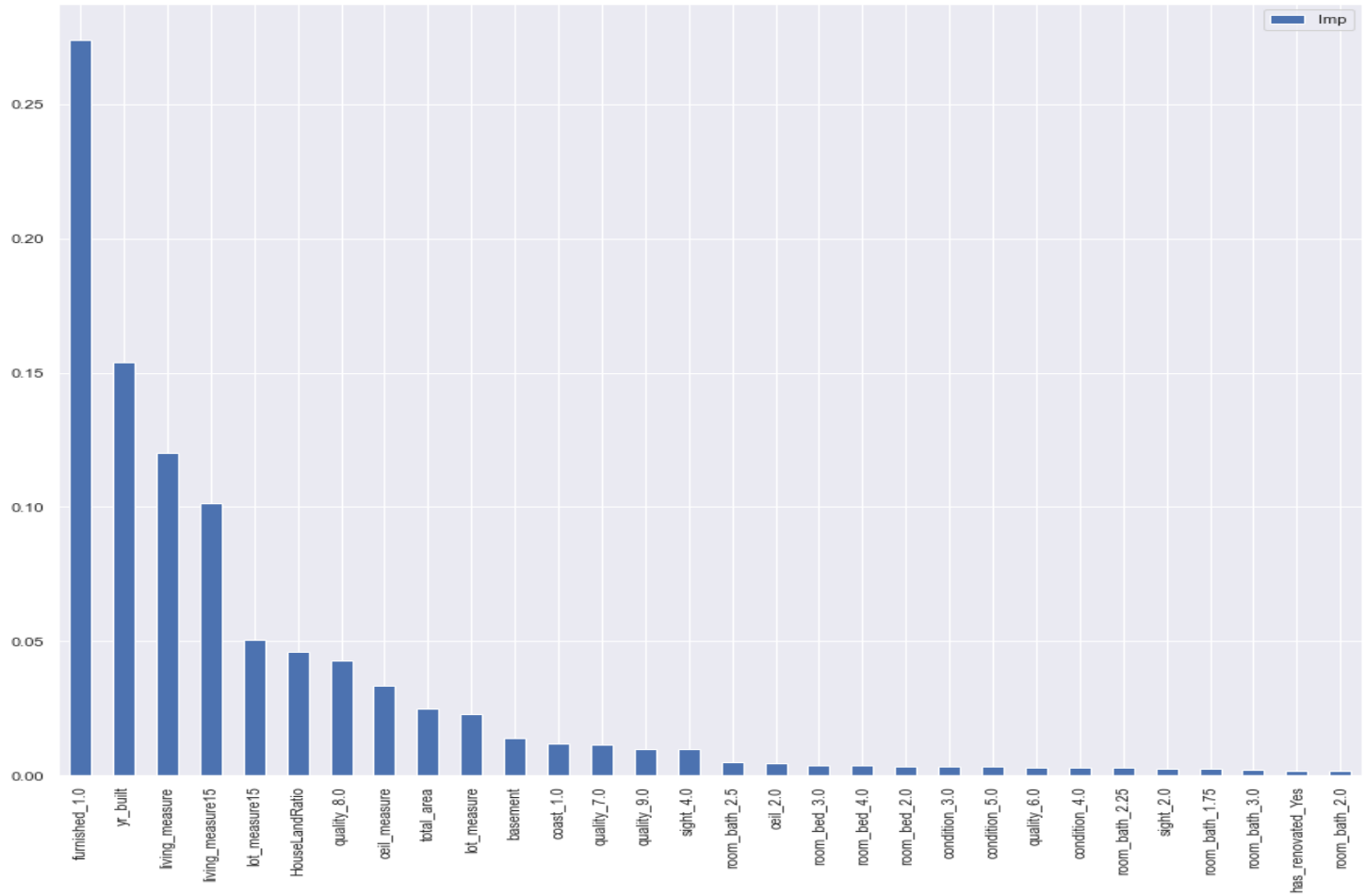
```
First 25 feature importance:     Imp    98.917
dtype: float64
First 30 feature importance:     Imp    99.453
dtype: float64

['furnished_1.0',
 'living_measure',
 'yr_built',
 'living_measure15',
 'quality_8.0',
 'lot_measure15',
 'sight_4.0',
 'HouseLandRatio',
 'coast_1.0',
 'quality_9.0',
 'basement',
 'quality_7.0',
 'ceil_measure',
 'total_area',
 'quality_6.0',
 'lot_measure',
 'condition_5.0',
 'quality_10.0',
 'ceil_3.0',
 'quality_5.0',
 'room_bath_2.5',
 'room_bath_3.75',
 'quality_11.0',
 'condition_3.0',
 'room_bath_3.5',
 'sight_2.0',
 'room_bed_2.0',
 'has_renovated_Yes',
 'room_bath_3.25',
 'quality_12.0']
```



The top 30 features are covering about 99% in gradient boosting model. This is very good coverage for just 30% of the variables

B) Random Forest model

```
First 25 feature importance:      Imp     96.441
dtype: float64
First 30 feature importance:      Imp     97.56
dtype: float64

['furnished_1.0',
 'yr_built',
 'living_measure',
 'living_measure15',
 'lot_measure15',
 'HouseLandRatio',
 'quality_8.0',
 'ceil_measure',
 'total_area',
 'lot_measure',
 'basement',
 'coast_1.0',
 'quality_7.0',
 'quality_9.0',
 'sight_4.0',
 'room_bath_2.5',
 'ceil_2.0',
 'room_bed_3.0',
 'room_bed_4.0',
 'room_bed_2.0',
 'condition_3.0',
 'condition_5.0',
 'quality_6.0',
 'condition_4.0',
 'room_bath_2.25',
 'sight_2.0',
 'room_bath_1.75',
 'room_bath_3.0',
 'has_renovated_Yes',
 'room_bath_2.0']
```



The top 30 features are covering about 97.5% in random forest model

Now will extract the top 30 features from the above model

```
['furnished_1.0', 'living_measure', 'yr_built', 'living_measure15', 'quality_8.0', 'lot_measure15', 'sight_4.0', 'HouseLandRati
o', 'coast_1.0', 'quality_9.0', 'basement', 'quality_7.0', 'ceil_measure', 'total_area', 'quality_6.0', 'lot_measure', 'conditi
on_5.0', 'quality_10.0', 'ceil_3.0', 'quality_5.0', 'room_bath_2.5', 'room_bath_3.75', 'quality_11.0', 'condition_3.0', 'room_b
ath_3.5', 'sight_2.0', 'room_bed_2.0', 'has_renovated_Yes', 'room_bath_3.25', 'quality_12.0']
['furnished_1.0', 'yr_built', 'living_measure', 'living_measure15', 'lot_measure15', 'HouseLandRatio', 'quality_8.0', 'ceil_mea
sure', 'total_area', 'lot_measure', 'basement', 'coast_1.0', 'quality_7.0', 'quality_9.0', 'sight_4.0', 'room_bath_2.5', 'ceil_
2.0', 'room_bed_3.0', 'room_bed_2.0', 'condition_5.0', 'room_bed_4.0', 'condition_3.0', 'quality_6.0', 'condition_4.0', 'room_b
ath_2.25', 'sight_2.0', 'room_bath_1.75', 'room_bath_3.0', 'has_renovated_Yes', 'room_bed_5.0']
```

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

```
38
['coast_1.0', 'condition_4.0', 'condition_3.0', 'ceil_3.0', 'condition_5.0', 'room_bath_2.5', 'yr_built', 'room_bath_2.25', 'Ho
useLandRatio', 'room_bath_1.75', 'room_bed_2.0', 'quality_10.0', 'living_measure15', 'quality_12.0', 'room_bed_4.0', 'basemen
t', 'sight_4.0', 'ceil_measure', 'living_measure', 'ceil_2.0', 'quality_9.0', 'room_bed_5.0', 'room_bath_3.25', 'quality_7.0',
'room_bed_3.0', 'room_bath_3.0', 'total_area', 'room_bath_3.75', 'quality_8.0', 'lot_measure15', 'furnished_1.0', 'has_renovate
d_Yes', 'quality_11.0', 'lot_measure', 'room_bath_3.5', 'quality_6.0', 'quality_5.0', 'sight_2.0']
```

From two models we have 38 importance features.

We will freeze on the above 38 list and make another dataframe (along with 'price')

	price	ceil_measure	coast_1.0	quality_8.0	condition_4.0	quality_5.0	has_renovated_Yes	living_measure15	room_bath_3.0	furnished_1.0	...	room_be
12235	460000	1760.0	0	0	1	0	0	2010.0	0	0	...	
14791	345600	2800.0	0	0	0	0	1	1780.0	0	1	...	
1742	750000	1550.0	0	1	0	0	0	1570.0	0	0	...	
17829	325000	2220.0	0	1	1	0	0	1980.0	0	0	...	
14810	390000	2240.0	0	1	0	0	0	1900.0	0	0	...	

5 rows × 39 columns

Eventhough PCA is helping us to reduce dimensions upto about numerous dimensions, we can see that in our random forest model top 30 features are explaining the 97.5% variance in the regression and 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 38 important features

HYPERTUNING with Gridsearch CV:

Here we define the combination and do the training.
Since we have better performance in gradient boosting model, we will hypertune 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 of 350 is best in range 50 to 400. Will test same until 1000

n_estimators of 1000 is giving best result in range 300 to 1000

In combination of 4 parameters above values are giving best result. We can see n_estimators of 500 is best.
Now the score has reduced compared to earlier run

The perforemed iteration gives best result of 0.6759.

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

```
'learning_rate': 0.1,
  'min_samples_leaf': 20,
  'min_samples_split': 5,
  'n_estimators': 500},
 0.6759087726904568)
```

We can conclude from above that gridsearch CV is giving better results compared to that of tuning done by any other method to avoid confusion.

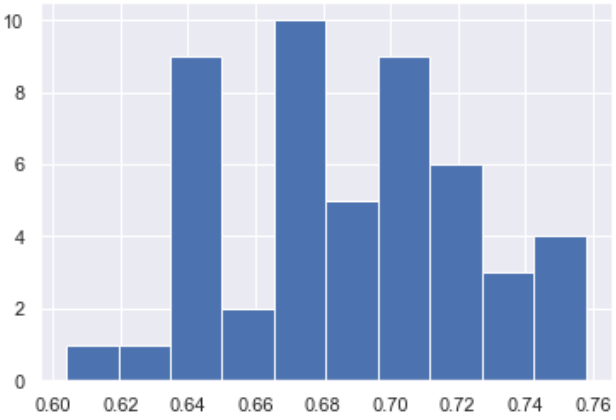
After hypertunning with GridSearchCV and finding the best parameters and making model out of it:

The best performance is given by Gradient boosting model with training (score-0.82,RMSE-105884), Validation (score-0.68.1,RSME-146570), Testing(score-0.677,RMSE-148559).

- Lowest RMSE
- Highest R-square
- Best computation speed
- Realistic Feature Importance

Confidence interval:

```
[0.67229071 0.64820077 0.75782791 0.66764233 0.6666573  0.71343338
 0.64181537 0.67256249 0.64441217 0.70065714 0.70219897 0.72562325
 0.75417056 0.67025383 0.6423067  0.68557476 0.73403047 0.60394036
 0.64548059 0.70746472 0.72058583 0.69485993 0.64112871 0.70246623
 0.71273139 0.67174256 0.70342233 0.68476627 0.68027571 0.74423032
 0.61959008 0.66815478 0.64405262 0.64992398 0.71133613 0.75720168
 0.68860687 0.7037949  0.71815504 0.68822918 0.73988198 0.71855115
 0.70911734 0.64399831 0.70416557 0.73337264 0.65554605 0.66098852
 0.67327301 0.67429641]
Accuracy: 68.750% (3.626%)
```



The 95% confidence interval scores range from 0.62 to 0.75.
Gives accuracy (Cross validation score) of 68.75%

it represents the performance of the system on the 80% of the data instead of just the 20% of the training set

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

INSIGHTS

- We are given a dataset of Houses of a city with their prices and various attributes.
- We are supposed to build a model to accurately predict the sale prices of houses through various given attributes & find their relation with prices.
- Our objective is to find best model with highest accuracy and low errors which gives most accurate price prediction compared to actual prices and can be applied in real world for commercial use.
- 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-0.82,RMSE-105884), Validation (score-0.68.1,RSME-146570), Testing(score-0.677,RMSE-148559).
- 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.

RECOMMENDATIONS

- The company can create a mobile application for the salesmen who are on site. The salesman can enter input for 38 variables and the output for the same will be the price of the house.
- We recommend that the company deploys **Gradient boosting model with training** (score-0.82,RMSE-105884), Validation (score-0.68.1,RSME-146570), Testing(score-0.677,RMSE-148559
- This model can also be used by housing finance companies to evaluate the house prices.
- It is strongly insisted that the user enters all the input most accurately to his/her knowledge to get the most accurate results. Any input entered incorrectly or left blank would result in incorrect prediction by the model.
- Factors like GDP, average income and the population can have impact on house prize and lead to better findings.
- It is suggested that the salesmen quote a price 10% higher than the model prediction so that there is room for negotiation

THE END