“House Price Prediction”

IDM TERM PROJECT- REPORT

Bilawal Ali 17278

Project Description.

Zameen.com is Pakistan's biggest Real Estate Company, an online property portal that lists real estate dealers, agencies and properties for sale and rent. It facilitates people to sell and rent their properties; these can be houses, flats, Farm Houses etc. It is functional in all major cities of Pakistan with 5 million visitors each month. It is the most authentic platform to rent and sell the property.

Problem Statement:

Zameen.com has the huge database, everyday there is a ton of data and they want this data to predict the prices of the houses/flats/farm-houses (given the other details as location, area etc) which is reasonable and accurate to the current market levels so that future agents/seller/buyers who wants to sell, rent or buy the properties will be satisfied with the overall prices according to the locations. So that these agents/buyers return to the platform again.

Objective:

So looking at the the problem we are responsible for building a Model for the given data to predict the prices of the houses by For Sale or For Rent.Also finding to show why same properties have different prices in different cities, how no# of bedrooms, bathrooms affect the overall price.

Data Description:

Overview of the data:

Dataset is easily available online and we found the data of the year 2019 (last updated by zameen.com). It consists of 16 columns and 168447 rows.

Mainly the data has the following 16 columns:

Ø property_id : Unique ID for every property

Ø location_id: Unique location id too.

Ø property_type: Flat, House, Penthouse, Farmhouse, Lower Portion,

Upper Portion and Room

Ø price : Price in Pakistani Rupees

Ø location: Location of the property

Ø city: City of the property

Ø province_name : Province of the property

Ø latitude: Latitude of the position of the property

Ø longitude: Longitude of the position of the property

Ø baths: Number of bathrooms

Ø area: Area of the property (In Maral & Kanal)

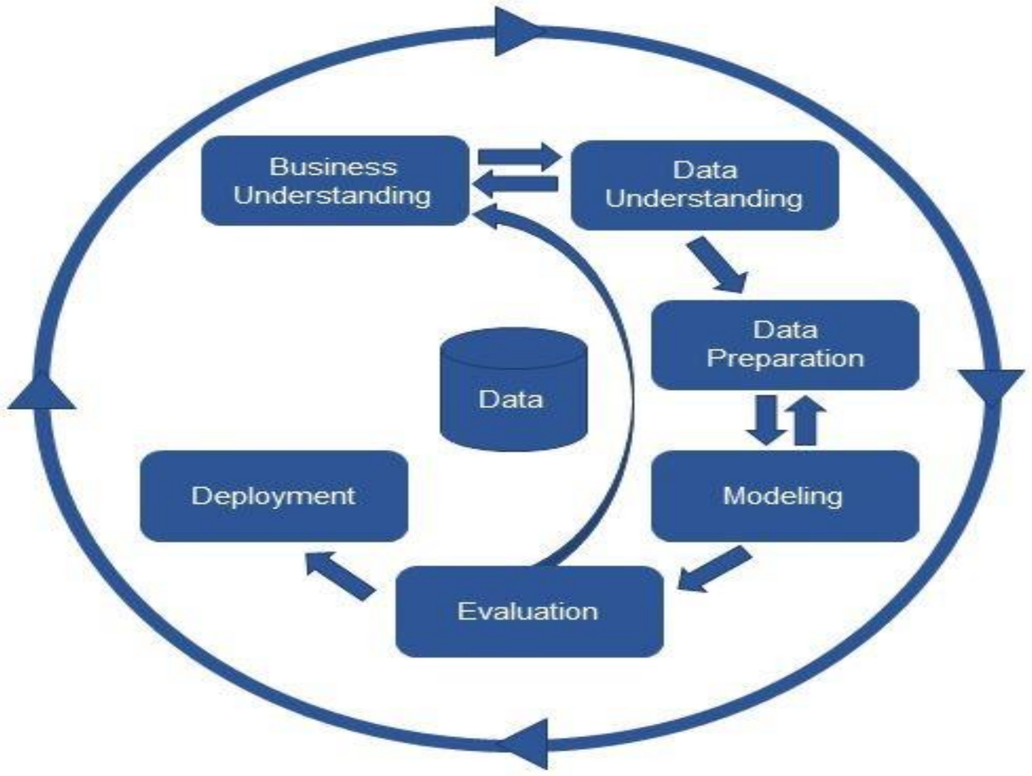
Ø purpose: For Sale or for Rent

Ø bedrooms: No# of Bedrooms

Ø agency: Agency Namce dealing the property

Ø agent: Agent name dealing the property

CRISP-DM MODEL



We will follow this CRISP-DM Model to understand the our data and get insights by going through data preparation and modeling part.

Business Understanding

We have already discussed the problem that we have to build a model that predicts the price of the house For Sale and For Rent.

Data Understanding

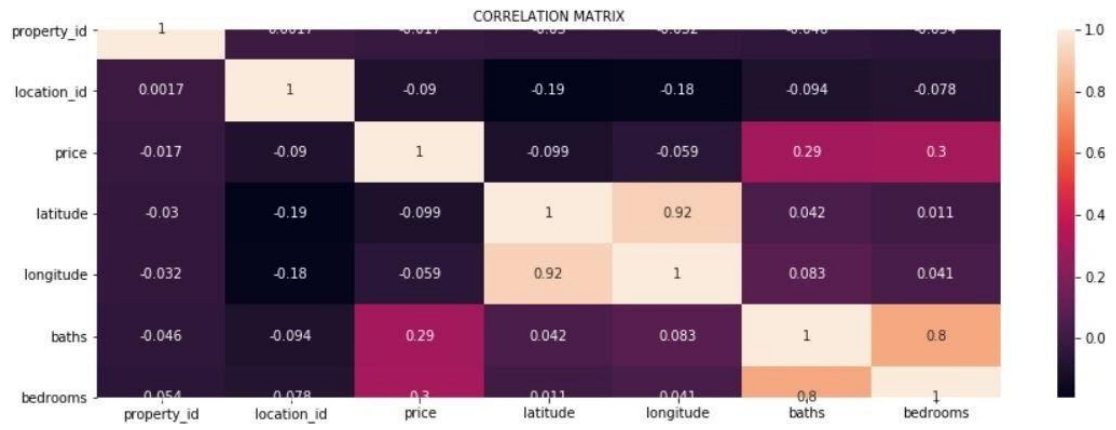
The given Data has only null/missing values in two columns (agency & agent) 70% rows were missing so we discarded these columns. Date added column was removed too as it does not have any effect on the price (as we know the data is of year 2019). We calculated the correlation of the price with other columns and found that property_id, location_id, latitude and longitude shows -ve corr so we dropped these columns too.

Property_type col has values : House 30501, Flat 11647, Upper Portion 3653

Lower Portion 2443, Room 316, FarmHouse 260, Penthouse 179

Purpose col has values : For Sale 35829, For Rent 13170

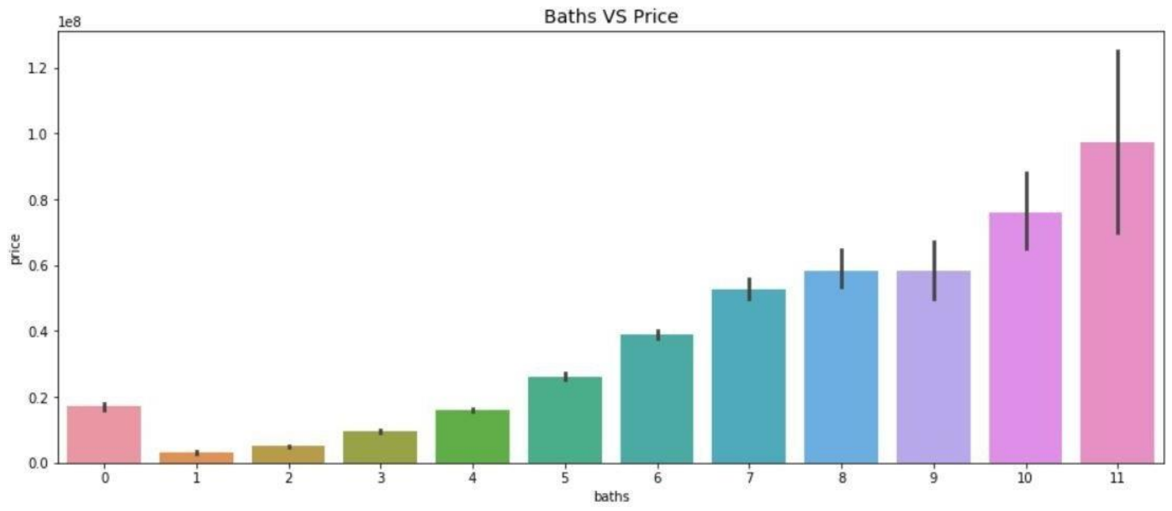
Correlation Matrix:



Area is given in the form of Kanal and Marla as (5 Marla, 2 Kanal) So we have to split it in two columns first area_type and area_size. Now we create another new column as Area_in_sq_ft It will convert Marla and Kanal into square foot and it gives the standard area for both Marla and Kanal. So drop the kanal, Maral and area size. Before finding area_sq_ft convert the string into float for multiplication purpose. Now we are remaining with nine columns.

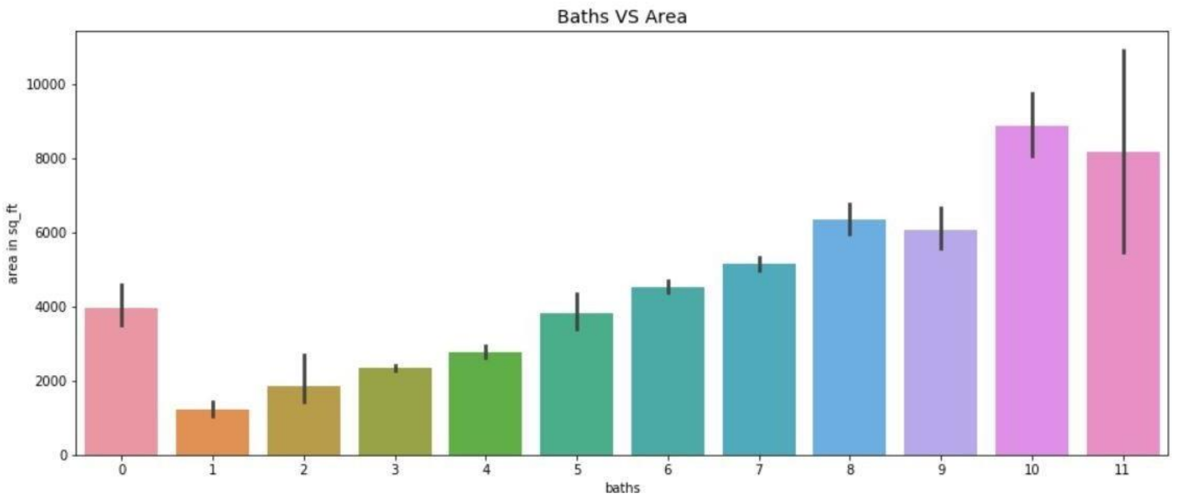
Find the relationship of those columns which has positive correlation with the price to understand more about these columns.

Bar plot between Baths and Price:



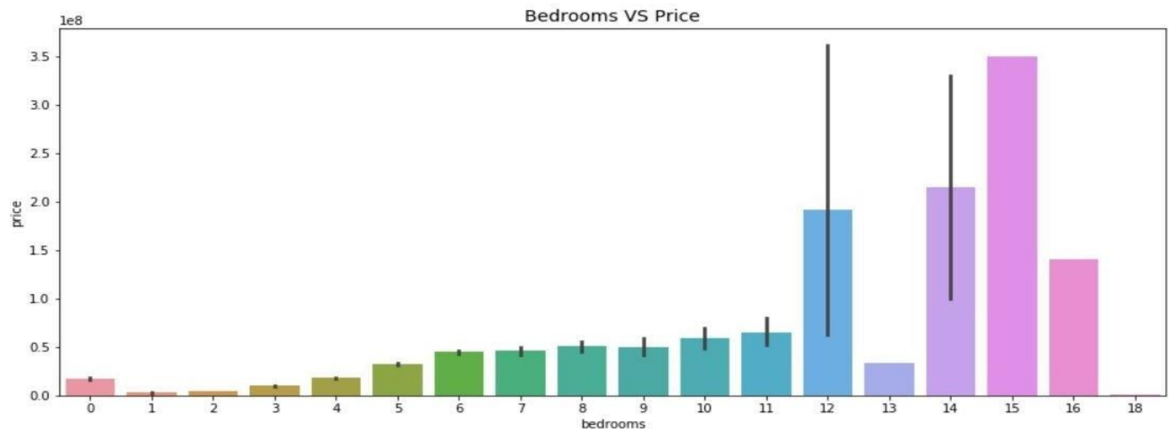
As number of baths increase the price increases too, here 0 baths price is almost same as house having 4 baths as this 0 baths shows a piece of land that does not have any baths. Let’s see the relation of baths with area.

Bar plot between Baths and Area:

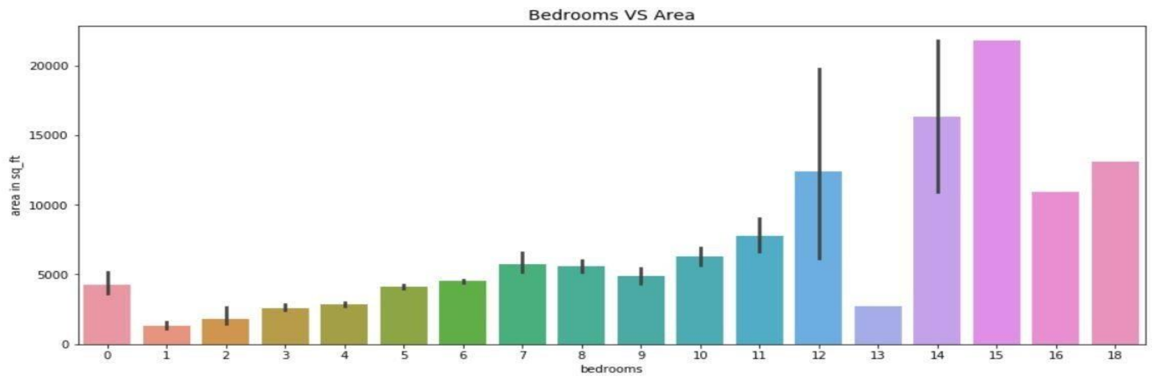


This reflect the baths vs price, as price increase so does the area.

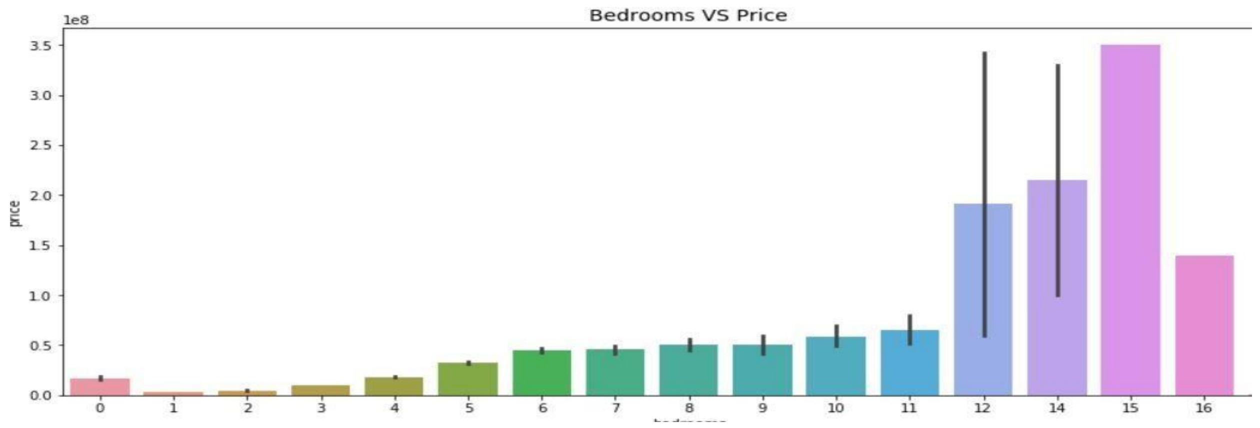
Bar plot between Bedrooms and Price:



So we see that if the bedrooms increase the price also increases, but with 13 bedrooms price drops drastically so we look at the relationship of bedrooms with area to further understand why price is very low for 13 bedrooms.

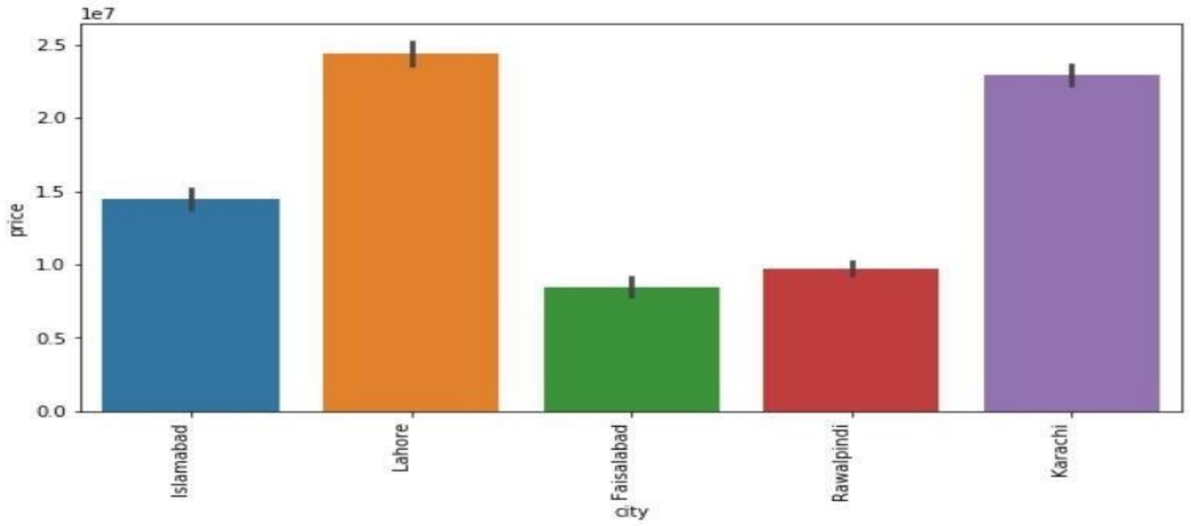


Here it shows the same relationship, so we remove/replace those rows where bedrooms are 13 with rows having bedrooms 3 as their price and area is almost the same.



After replacing the outliers.

Relationship between City and Price

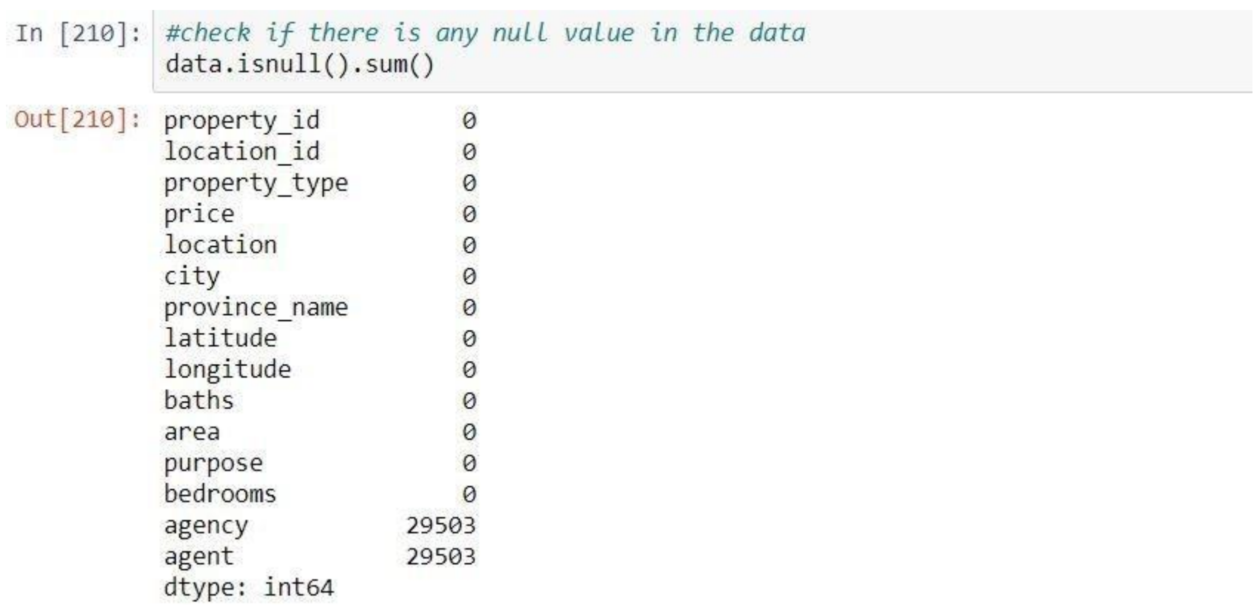


Prices in Lahore and Karachi are the highest, Islamabad is on 3rd, Rawalpindi on 4th and the lowest prices are in faisalabad. This shows that the same properties can have higher costs in some cities than others.

Data Preparation/pre-processing:

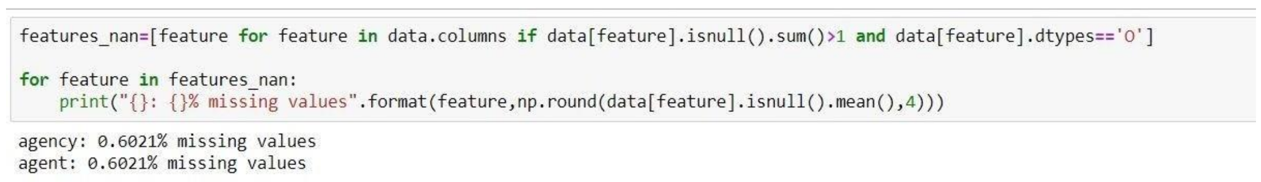
As we have understood our data and its multiple factors, now it will be easy to prepare the data so slets prepare the data while following the CRISP-DM model.

First thing is to find are there any null values in the data.



```
In [210]: #check if there is any null value in the data
          data.isnull().sum()

Out[210]: property_id          0
          location_id          0
          property_type        0
          price                0
          location             0
          city                 0
          province_name        0
          latitude             0
          longitude            0
          baths                0
          area                 0
          purpose              0
          bedrooms             0
          agency           29503
          agent            29503
          dtype: int64
```



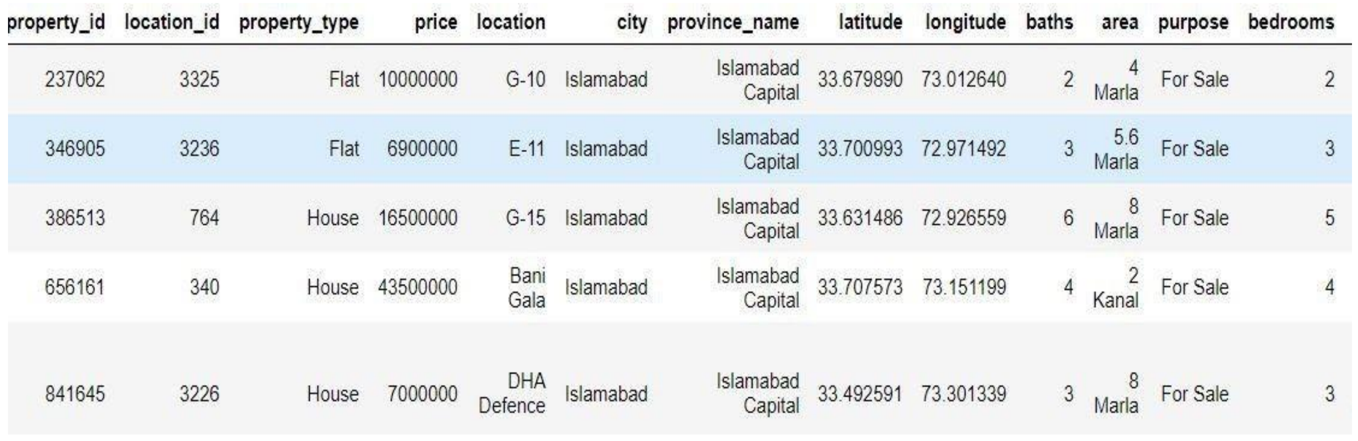
```
features_nan=[feature for feature in data.columns if data[feature].isnull().sum()>1 and data[feature].dtypes=='O']

for feature in features_nan:
    print("{}: {}% missing values".format(feature,np.round(data[feature].isnull().mean(),4)))

agency: 0.6021% missing values
agent: 0.6021% missing values
```

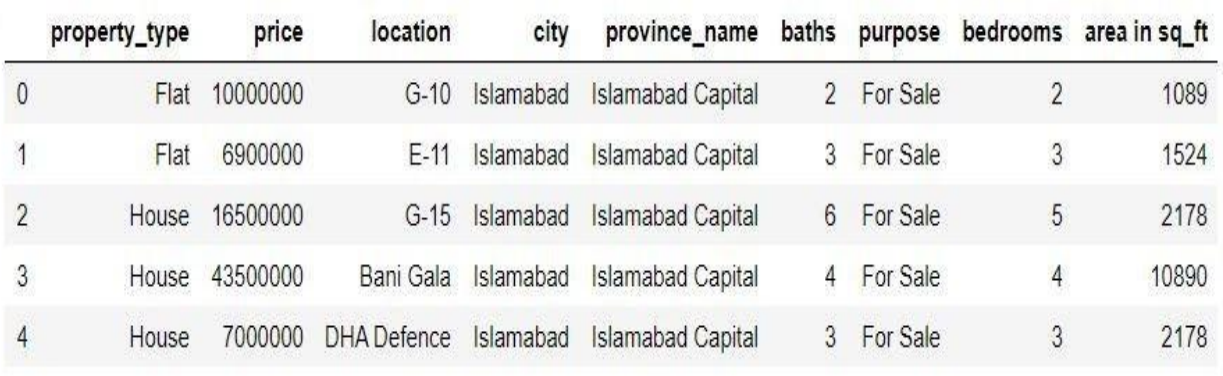
So two columns have more than 60% rows as null values so we drop these columns.

Lets the data now



property_id	location_id	property_type	price	location	city	province_name	latitude	longitude	baths	area	purpose	bedrooms
237062	3325	Flat	10000000	G-10	Islamabad	Islamabad Capital	33.679890	73.012640	2	4 Marla	For Sale	2
346905	3236	Flat	6900000	E-11	Islamabad	Islamabad Capital	33.700993	72.971492	3	5.6 Marla	For Sale	3
386513	764	House	16500000	G-15	Islamabad	Islamabad Capital	33.631486	72.926559	6	8 Marla	For Sale	5
656161	340	House	43500000	Bani Gala	Islamabad	Islamabad Capital	33.707573	73.151199	4	2 Kanal	For Sale	4
841645	3226	House	7000000	DHA Defence	Islamabad	Islamabad Capital	33.492591	73.301339	3	8 Marla	For Sale	3

We can clearly see that area is defined in two units (marla and Kanal) so lets first split it into different units and then create a column with the same unit(normalized value) for both Marla and Kanal.



	property_type	price	location	city	province_name	baths	purpose	bedrooms	area in sq_ft
0	Flat	10000000	G-10	Islamabad	Islamabad Capital	2	For Sale	2	1089
1	Flat	6900000	E-11	Islamabad	Islamabad Capital	3	For Sale	3	1524
2	House	16500000	G-15	Islamabad	Islamabad Capital	6	For Sale	5	2178
3	House	43500000	Bani Gala	Islamabad	Islamabad Capital	4	For Sale	4	10890
4	House	7000000	DHA Defence	Islamabad	Islamabad Capital	3	For Sale	3	2178

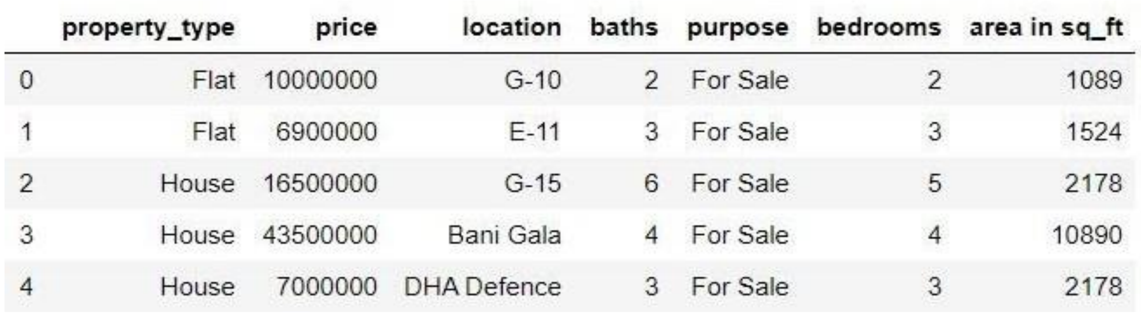
We have created the new column as area in sq_ft and dropping area column that has both the numeric and string value together.

Handling Categorical Variable

It was now time to manage categorical variables after dealing with all messy data successfully, as we can not directly feed these variables to our model.

To deal with categorical attributes, there are different techniques.we used get_dummies. By building a feature and using the get dummies pandas process, we encode the categorical variables. This returns a processed data frame that has all the categorical variables encoded.

After dropping the columns, the final shape of the data is,



	property_type	price	location	baths	purpose	bedrooms	area in sq_ft
0	Flat	10000000	G-10	2	For Sale	2	1089
1	Flat	6900000	E-11	3	For Sale	3	1524
2	House	16500000	G-15	6	For Sale	5	2178
3	House	43500000	Bani Gala	4	For Sale	4	10890
4	House	7000000	DHA Defence	3	For Sale	3	2178

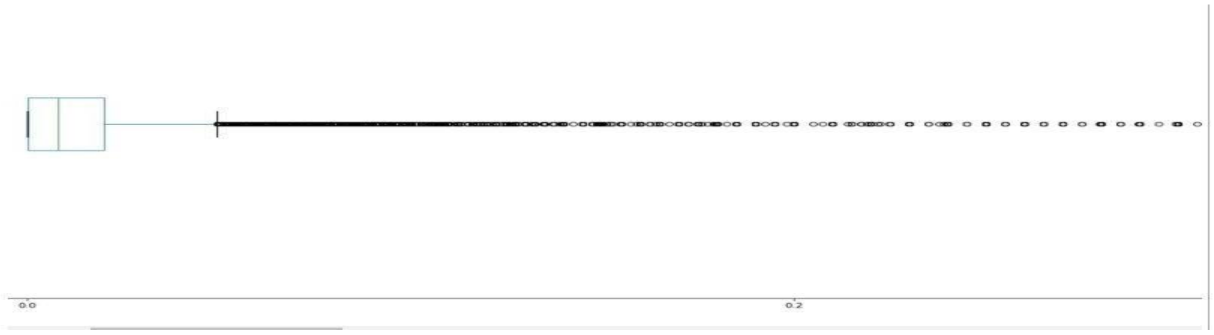
Now apply get_dummies function on property_type, location and purpose

We have now 1340 columns, wohoo.

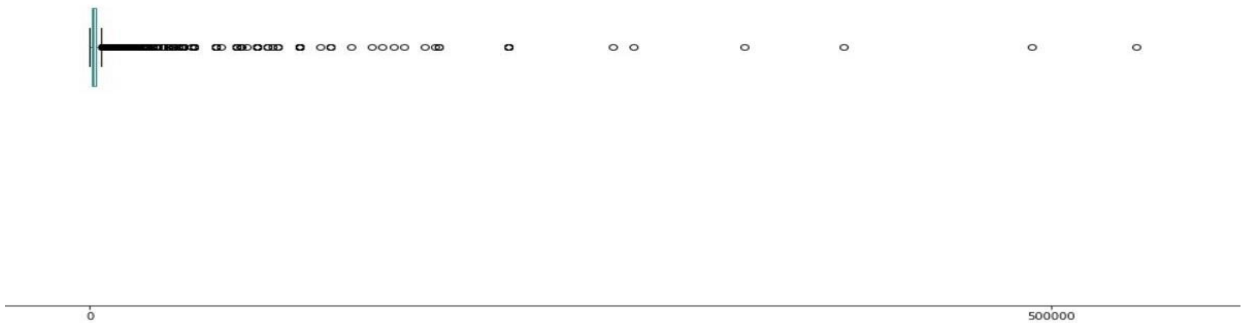
Data.shape = (48999, 1340)

Handling Outliers in the data:

Price: properties having price greater than 40 crore are outliers so we remove the rows all the having this price



Area: properties having area more than 50 thousand sq_ft are outliers



Finally we are done with the preparation of the data, now it is time to apply the models.

Our data is cleaned and ready for the models.

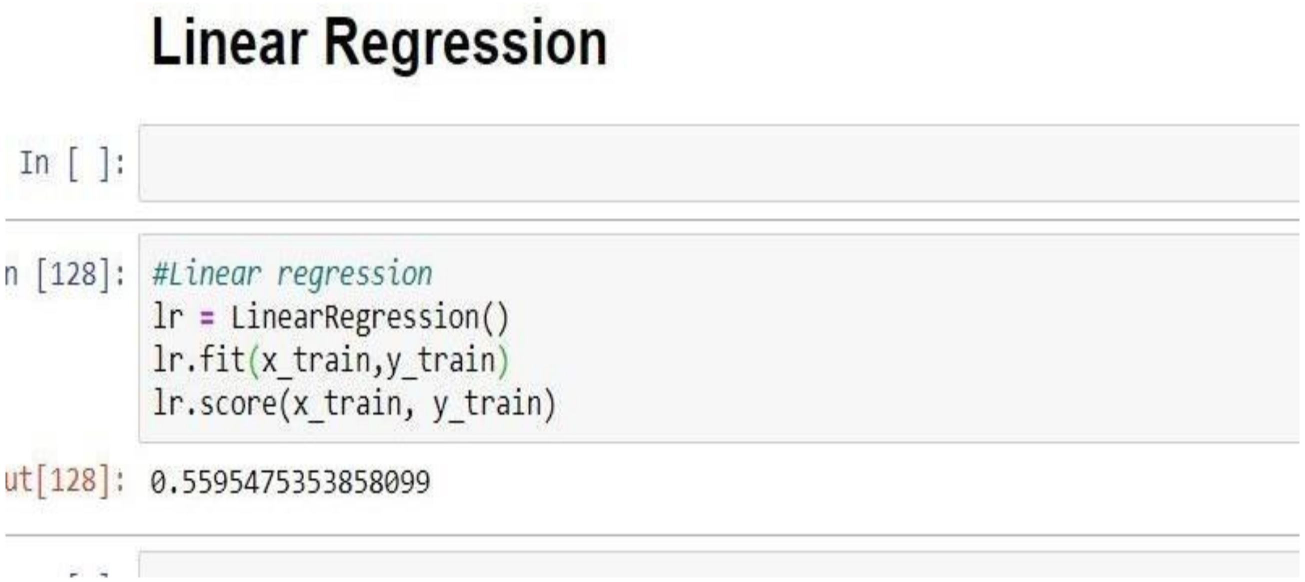
Modeling:

1. Linear Regression

We applied linear regression as an initial model to get started.

It gave an accuracy of 56.3 % without doing any significant work on the data, though it is not a good number.

After we train the data the accuracy drops to 55.9%.



Linear Regression

```
In [ ]:

n [128]: #Linear regression
         lr = LinearRegression()
         lr.fit(x_train,y_train)
         lr.score(x_train, y_train)

ut[128]: 0.5595475353858099
```

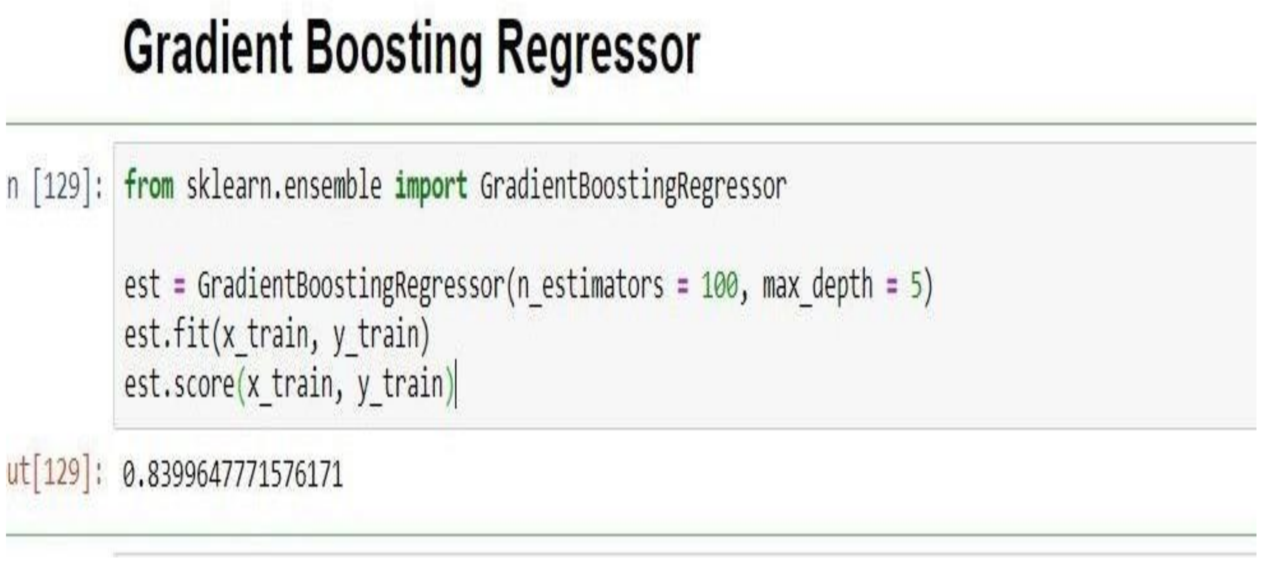
After removing the outliers and preparing the data does not have any impact on it so we decided to move on to other models.

2. Gradient Boosting Regressor

Grafing boosting regressor gave an accuracy of 70% when we set the depth at 2 and no# of models to 100.

But it was taking a lot of time it took 30 seconds at depth = 2 & no# of models = 100.

When we prepare the model and increases the depth from 2 to 5 with same 100 no of models it took 4 minutes to run and gave an accuracy of 83.99%.



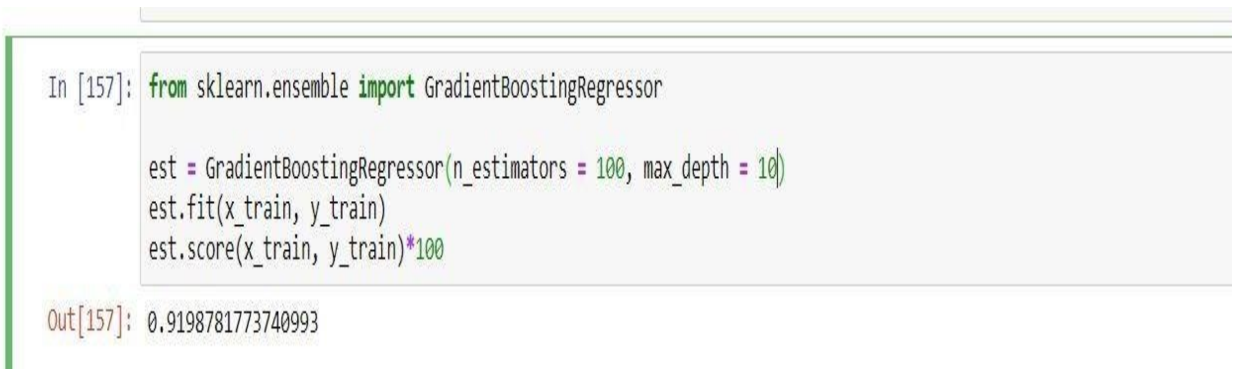
Gradient Boosting Regressor

```
n [129]: from sklearn.ensemble import GradientBoostingRegressor

         est = GradientBoostingRegressor(n_estimators = 100, max_depth = 5)
         est.fit(x_train, y_train)
         est.score(x_train, y_train)

ut[129]: 0.8399647771576171
```

When we increase the no# of models and depth of the model it takes more that 5 minute to run (it took 5 minutes when we just increases the depth from 5 to 10 and our PC got hanged if we increase the no# of models and depth at the same time).



```
In [157]: from sklearn.ensemble import GradientBoostingRegressor

          est = GradientBoostingRegressor(n_estimators = 100, max_depth = 10)
          est.fit(x_train, y_train)
          est.score(x_train, y_train)*100

Out[157]: 0.9198781773740993
```

So we thought of taking a small sample of our data but when we took the small portion of the data it drastically affects the accuracy of the overall predictions.

So we found a solution to it that we have a column in our data set named “Purpose” which shows what type of property is: “FOR SALE ” or “FOR RENT” and these both attributes have a drastic difference in their prices. For example if a house price is 1 crore then this same house can be rent for 20 thousands. So we split our data according to the column ‘purpose’ For sale and For Rent.

So let’s first split the data:

We have a Data set with (168446, 16) rows & columns.

FOR SALE:

data_sale = data[data["purpose"]=="For Sale"] = (130655, 16)

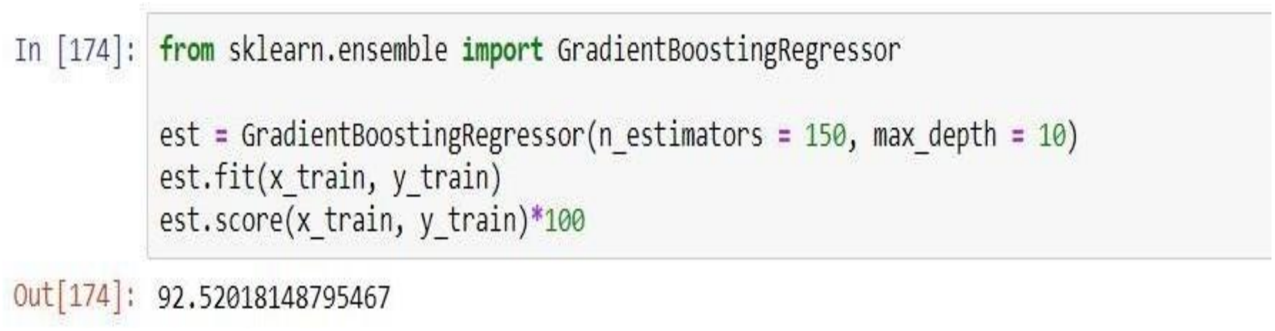
For RENT:

data_rent = data[data["purpose"]=="For Rent"] = (37791, 16)

But here we already have taken the sample of 49 thousands so further split it in FOR SALE and FOR RENT and comes out as: (35829, 15) For_sale and (13170, 15) For_rent so now we can easily work on it though we took (37791,16) sample for For_rent.

Gradient Boosting on the splitted data (“For Sale only”)

- Accuracy has improved from 83% to 92.5%, which is a very good improvement.
- It took less time (took 3 min to run as the now we dropped all the columns which has For rent in them)
-

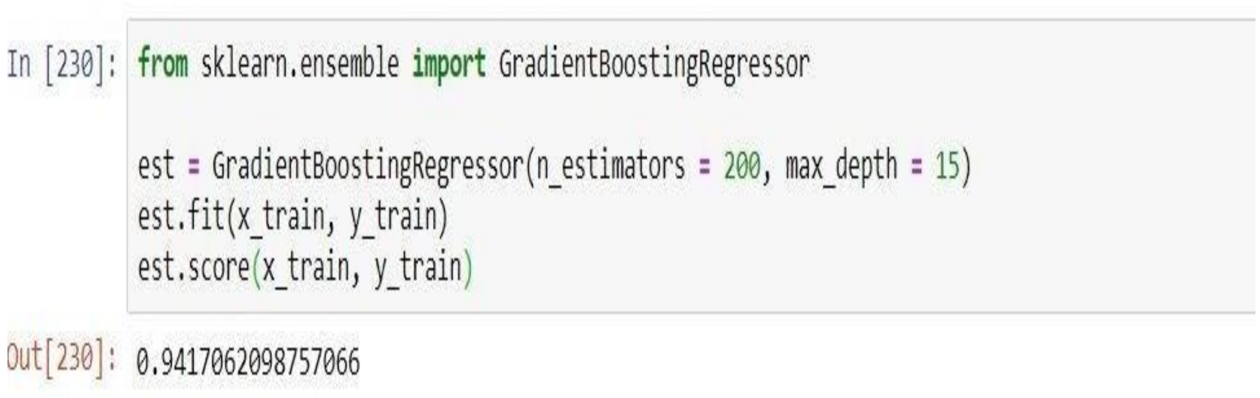


```
In [174]: from sklearn.ensemble import GradientBoostingRegressor

          est = GradientBoostingRegressor(n_estimators = 150, max_depth = 10)
          est.fit(x_train, y_train)
          est.score(x_train, y_train)*100

Out[174]: 92.52018148795467
```

-
- We increase the depth to 15 and no# of estimators to 200. Accyarcy improved to 94%



```
In [230]: from sklearn.ensemble import GradientBoostingRegressor

          est = GradientBoostingRegressor(n_estimators = 200, max_depth = 15)
          est.fit(x_train, y_train)
          est.score(x_train, y_train)

Out[230]: 0.9417062098757066
```

-

Libraries Used:

import pandas as pd: data management, as data cleaning and analyzing.

import matplotlib.pyplot as plt: for making the plots and graph

from sklearn.metrics import r2_score, mean_squared_error: Finding the R^square, error.

from sklearn.metrics import accuracy_score : To find the accuracy of the model

from sklearn.metrics import roc_curve : For ROC Curve fidining

import statistics

import scipy.stats as stats

import math

import re

Evaluation:

So the findings that we have so far Gradient Boosting Regressor was performing well among other models. R^2 and error values were very low when tested on different samples of the data.

- Though we observed the increase in accuracy by increasing the depth and no# of models in the model but it took a lot of time to run.
- So we split the data, and make 2 models , one for For_sale and One for For_rent
-
- Gradient Boosting with 200 no# of models and 15 depth gave the best accuracy of 94%.
- So wek took Gradient Boosting Regressor as our final model for this project.

Findings:

Limitations:

Though we preferred using all of the cleaned data for our models, It was taking a lot of time to run. After some observations, we found that prices of for rent and for sale properties are different, which caused inefficiency in our models. We decided to use separate models for properties on sale, which increased our efficiency.

Advice to the Zameen.com:

The one advice should be that add more description on area,bedrooms and bathroom of the house, what will be the area of the rooms, how big are the bedrooms. How much is the living area?

Secondly We found in our data that 25% properties had 0 bedrooms and 0 bathrooms, which cause a lot of misunderstanding in the data. We assumed that these properties with no bedrooms and bathrooms are lands not built yet(plots), but they have property types present, which caused ambiguity. Our advice is that Zameen should restrict the user to use 0 bedrooms and bathrooms on property types they have provided. They should use “Land” as property types to denote 0 bedrooms and baths.

Data pre_processing:

As the given data has many things that have very low or no contribution in the price estimation. So, our first job was to remove those attributes. They are property_id, location_id, location(as longitude and latitude were included so there is no need of location any more), city, province_name, agency (a great amount of values were null) and agent columns. After removing them, most relevant entries are left.

Now, the units of a column should be the same for all row-entries of that column. So, in the area field, all values were converted into square feet unit from kanal and marla.

Also, in “Purpose” column, one value was selected at a time because of the huge difference in the prices that might cause the model to be mislead.