Project report on

Property Price Prediction

Submitted towards partial fulfilment of the criteria for award of PGPDSE by Great Lakes Institute of Management

Submitted By
Group: 2 [Batch: May 2022]

Group Members

1. **Apurv Thapa**
2. **Shubham Madhwal**
3. **Vaibhav Bharadwaj**
4. **Kartik Bijalwan**
5. **Pawan Kumar**
6. **Itsa**

Mentor
Mr. Pratik Sonar

GREAT LAKES
EXECUTIVE LEARNING

Great Lakes Institute of Management

ACKNOWLEDGEMENT

This is to certify that the work done by the team for the implementation and completion of this project is original and to the best of our knowledge. It is a team effort and each of the member has equally contributed in the project. We heartily thank our project guide, Mr. Pratik Sonar, for his guidance and suggestions during this project work.

Date:

Place:

CERTIFICATE OF COMPLETION

I hereby certify that the project titled "Property Price Prediction"for case resolution was undertaken and completed under the supervision of Mr. Pratik Sonar for Post Graduate Program in Data Science and Engineering (PGP – DSE).

Mentor : Mr. Pratik Sonar

TABLE OF CONTENTS

<u>Business Understanding</u>

Overview:

People and real estate agencies buy or sell houses, people buy to live in or as an investment and the agencies buy to run a business. Either way we believe that everyone should get exactly what they pay for over-valuation/under-valuation in housing market has always being a issue. There are multiple factors on which price of an house depends which includes city, location , size , and sometimes the name of the builder can also be a deciding factor. Taking those factors in account and studying the given in detail we can train and deploy ML model to predict the price of the house. Although there are other factors as well which determines the price of house like recession, GDP, bank loan rates,etc. which will not be the part of our project.
Predicting the prices will help the customer as well as company to select regions depending upon their budget. Also, using Eda we can classify the area with higher prices vs lower prices, and find other insights as well.
In this project we are working on the dataset of the company Makaan.com for Price prediction.

Business Problem:

i. Business Problem Understanding:

The company wants to predict prices of houses that will be listed in their site using Machine Learning Models. Based on the past data given to us, we need to predict the price.

ii. Business Objective:

The Business wants to improve the customer experience so that they can list their house without the fear of under/over valuation. Which would ultimately lead to new customers.

iii. Approach:

Studying past data provided by Makaan.com and understanding the dependency of these features onto prediction of price of a house by deploying linear regression machine learning algorithms striving for a higher accuracy in prediction.

iv. Conclusions:

By implementing prediction model (Linear Regression) company would be able achieve better predictions.

Data Description

Dataset consists of the dataset scraped from makaan.com. It has 32 features and 332096 observations.

Categorical Data Type:

1. Property_Name Name of the Property
2. Property_type Type of property(Apartment,Residential Plot ,Independent Floor,Independent House,Villa)
3. Property_status Status of property(Ready to move/Under construction)
4. Price_per_unit_area Price per sqr feet area
5. Posted_On Time since posted in min/hr/days/months/years
6. Project_URL The URL of the project
7. Builder_name Builder's name
8. Property_building_status Still for sale or not (active/inactive/unverified)
9. City_name City name
10. No_of_BHK Number of bedrooms
11. Locality_Name Unique Locality name
12. Price Price of the Property (Target Variable)
13. Size Total size of property in sq feet
14. Sub_urban_name Unique sub urban name
15. description Description given by the people who posted
16. is_furnished Is (furnished,semi-furnished,unfurished)
17. Listing_Category For selling or rent

Numerical Data Type:

1. Property_id Unique Property ID
2. builder_id Builder unique ID
3. City_id Unique ID of city
4. Locality_ID Unique Locality ID
5. Longitude Longitudinal Co-ordinates
6. Latitude Latitudinal Co-ordinates
7. Sub_urban_ID Unique sub urban id
8. listing_domain_score Score between 1 to 10 which tells how likely the ad will be shown first

Boolean Data Type:

1. is_plot — Whether a plot or not
2. is_RERA_registered — Whether RERA approved or not
3. is_Apartment — Whether apartment or not
4. is_ready_to_move — Whether ready to move or not
5. is_commercial_Listing — Whether a commercial or not
6. is_PentaHouse — Whether penthouse or not
7. is_studio — Whether studio or not

Data Pre-Processing

1.Data Type Conversion:

As we go through the data we found some of the variables have incorrect datatype so we rectify those variables
with the correct datatypes.
To infer that we have used .apply(), lambda, .astype() function to change the datatypes in the data set.

```
In [9]: #Data_type conversions:

In [10]: #Price_per_unit_area(sq feet) is an object we convert it to float by removing comma mark
         df['Price_per_unit_area']=df['Price_per_unit_area'].apply(lambda x:float(x.replace(',','')))
         #from Price we remove comma and convert to int
         df['Price']=df['Price'].apply(lambda x:int(x.replace(',','')))
         #from size remove sq. ft.
         df['Size']=df['Size'].apply(lambda x: x.split(' ')[0])
         #from size we remove comma and convert to int
         df['Size']=df['Size'].apply(lambda x:int(x.replace(',','')))
         #change the No_of_BHK column to integer
         df['No_of_BHK']=df['No_of_BHK'].apply(lambda x:int(x.split(' ')[0]))

In [11]: # CHANGING ALL BOOLEAN DATATYPES TO OBJECT DATATYPES
         df.is_plot= df.is_plot.astype('object')
         df.is_RERA_registered= df.is_RERA_registered.astype('object')
         df.is_Apartment= df.is_Apartment.astype('object')
         df.is_ready_to_move=df.is_ready_to_move.astype('object')
         df.is_commercial_Listing= df.is_commercial_Listing.astype('object')
         df.is_PentaHouse= df.is_PentaHouse.astype('object')
         df.is_studio=df.is_studio.astype('object')
```

2.Feature Engineering:

Using the domain knowledge we will drop some columns and create new columns as well.

```
In [16]: #Removal unnecessary features.
```

```
In [17]: #Property Name has no effect on the price of the property,it also has 1lakh+ null values
#so we drop it all together
df.drop('Property_Name',axis=1,inplace=True)
#drop property_id
df.drop('Property_id',axis=1,inplace=True)
#Its better to drop Posted_On because it will change every minute and property prices do not change so frequently.
df.drop('Posted_On',axis=1,inplace=True)
#Every url is unique so we drop Project_URL
df.drop('Project_URL',axis=1,inplace=True)
#drop builder_id and builder_name
df.drop(['builder_id','Builder_name'],axis=1,inplace=True)
#We drop city_id, this way we can encode City_name.
df.drop('City_id',axis=1,inplace=True)
#drop Locality_Name and Locality_ID
df.drop(['Locality_Name','Locality_ID'],axis=1,inplace=True)
#we have City_name and Sub_urban_name to identify the area therefore, can drop Longitide and Latitude
df.drop(['Latitude','Longitude'],axis=1,inplace=True)
#drop Sub_urban_ID because its an int where as Sub_urban_name is object and can be encoded.
df.drop('Sub_urban_ID',axis=1,inplace=True)
#drop description as well
df.drop('description',axis=1,inplace=True)
#is_plot and is_Apartment information is given in Property_type, we drop is_plot and is_apartment
df.drop(['is_plot','is_Apartment'],axis=1,inplace=True)
#Listing Category has only one value i.e sell
df.drop('Listing_Category',axis=1,inplace=True)
#is_ready_to_move and Property_status tells the same thing drop is_ready_to_move
df.drop('is_ready_to_move',axis=1,inplace=True)
#none of the property are for commercial use
df.drop('is_commercial_Listing',axis=1,inplace=True)
#drop Price_per_unit_area as we create a new column later
df.drop('Price_per_unit_area',axis=1,inplace=True)
```

```
In [22]: #Since Price is the squared off value of price_per_unit_area*Size
#So we created a new feature price_per_sqft by dividing Price/Size
df['price_per_sqft']=df['Price']/df['Size']
```

After doing so we are left with 9 categorical variables and 4 numerical variables. Including 1 target variable.

3.Null Value Treatment:

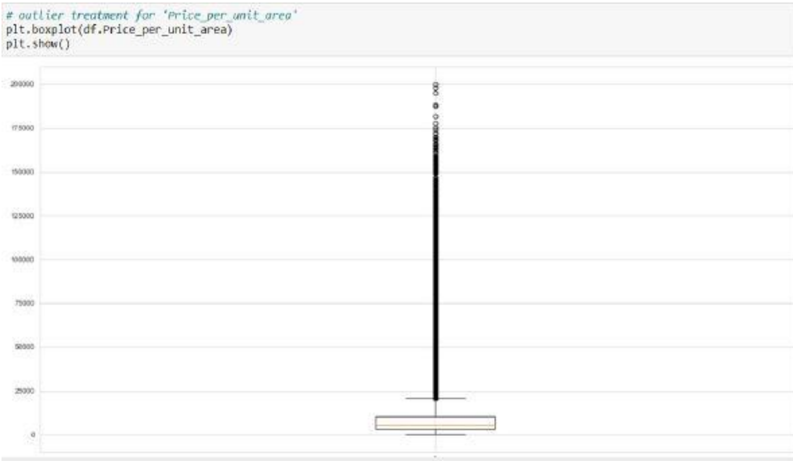
Null value treatment is essential for building most of the commonly used machine learning models. Such as linear regression, decision tree, random forest, etc.To infer that we have used .isnull() function to check the null values in the data set.

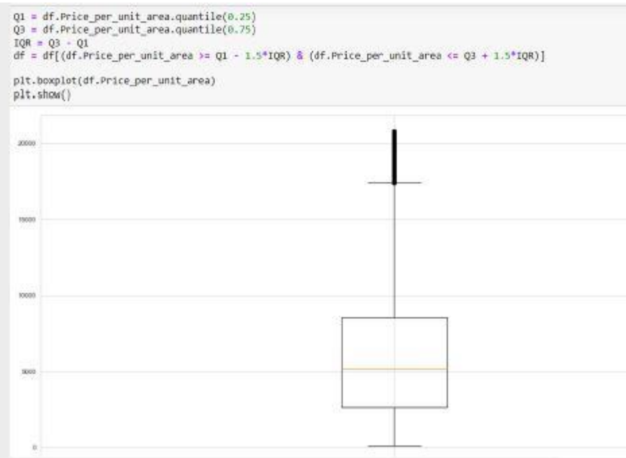
```
In [26]: df.isnull().sum()

Out[26]: Property_type                 0
         Property_status           60442
         Property_building_status      0
         City_name                     0
         No_of_BHK                     0
         Price                         0
         Size                          0
         Sub_urban_name                0
         is_furnished                  0
         listing_domain_score          0
         is_RERA_registered            0
         is_PentaHouse                 0
         is_studio                     0
         price_per_sqft                0
         dtype: int64
```

Here, only Property_status has null values. On futher study we realized that Property_type as Residential Plot had null values in Property_status column. And after studying the Residential Plot we came to know that all the Residential Plots had 'ready to move ' as Property_status. Thus, we filled the Property_status null values with 'ready to move'. Logically also this made sense as plot have no construction and they are always ready to move.

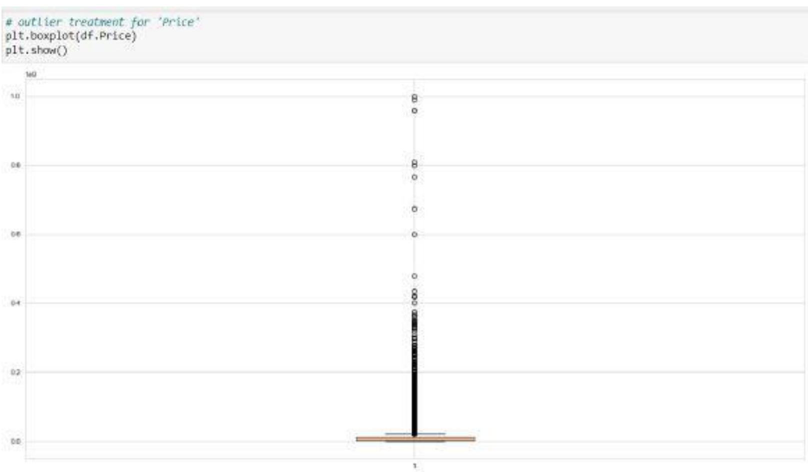
4.Outlier Treatment:

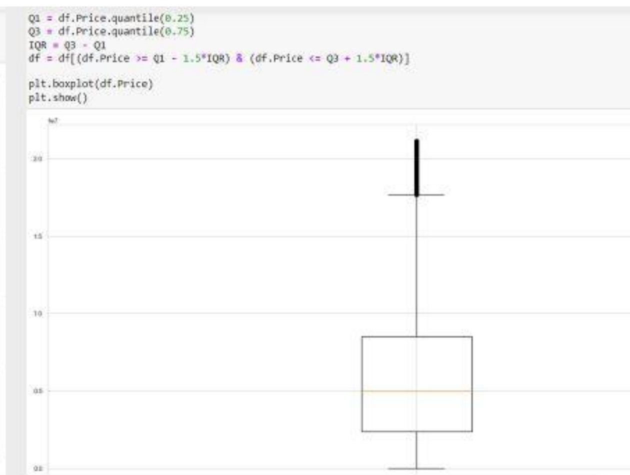
We remove the outliers using IQR method.



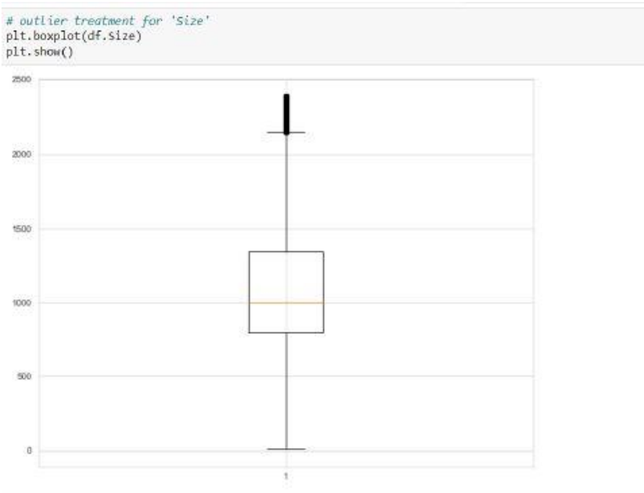


For Price per unit area.

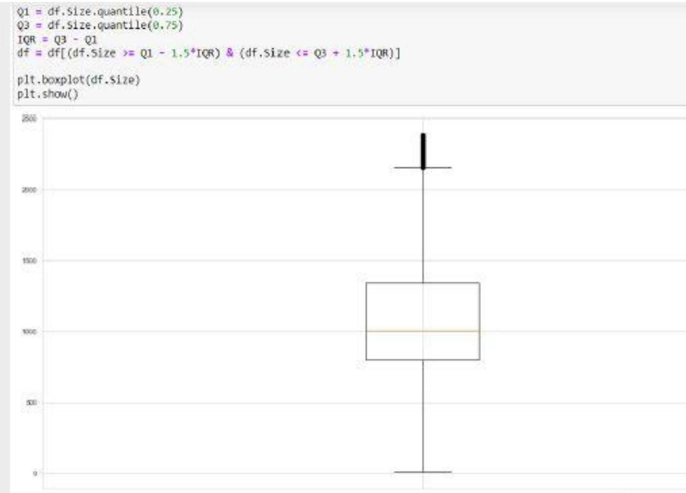




For Price.



# outlier treatment for 'Size'
plt.boxplot(df.Size)
plt.show()
2500
2000
1500
1000
500
0
1



Q1 = df.Size.quantile(0.25)
Q3 = df.Size.quantile(0.75)
IQR = Q3 - Q1
df = df[(df.Size >= Q1 - 1.5*IQR) & (df.Size <= Q3 + 1.5*IQR)]
plt.boxplot(df.Size)
plt.show()

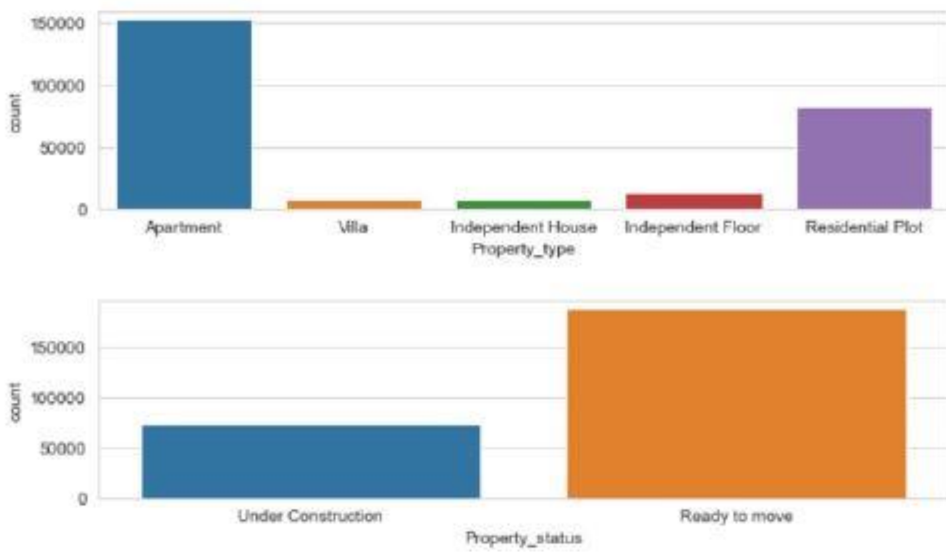
For Size

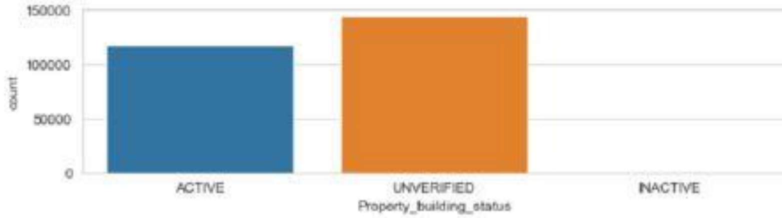
Exploratory Data Analysis

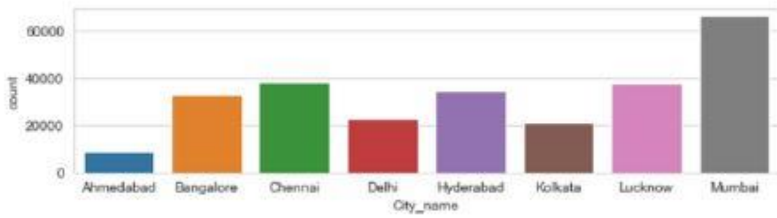
Univariate Analysis

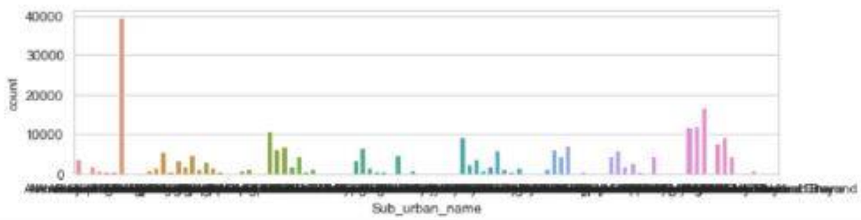
1.Categorical Columns:

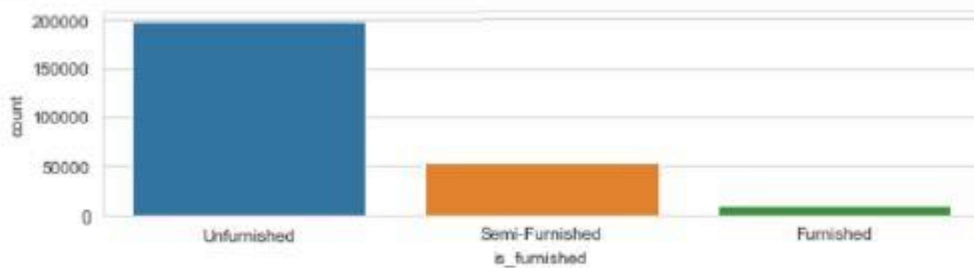
```
t=1
for i in df.select_dtypes('object').columns:
    plt.subplot(4,2,t)
    sns.countplot(df[i])
    t+=1
    plt.show()
```

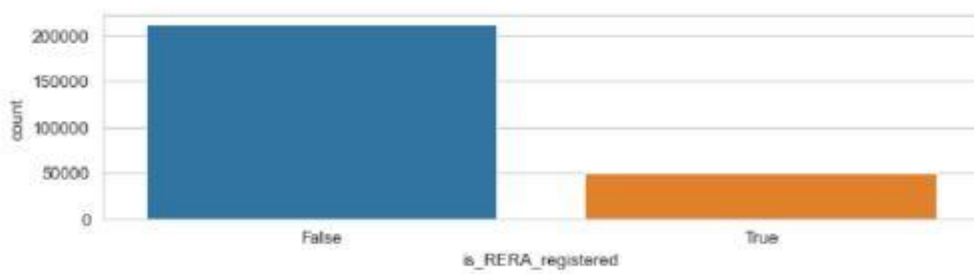


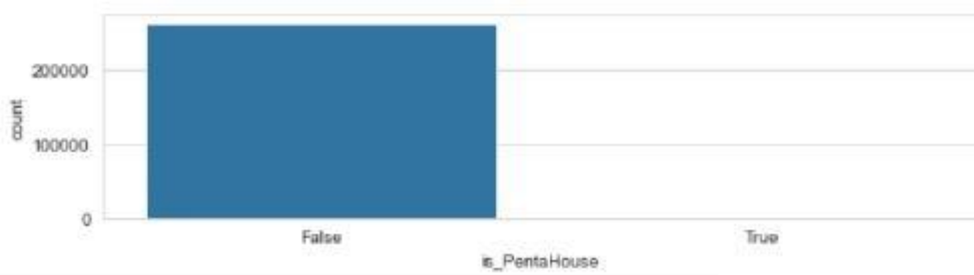

150000
100000
50000
0
count
ACTIVE
UNVERIFIED
INACTIVE
Property_building_status


60000
40000
20000
0
count
Ahmedabad
Bangalore
Chennai
Delhi
Hyderabad
Kolkata
Lucknow
Mumbai
City_name


40000
30000
20000
10000
0
count
Sub_urban_name

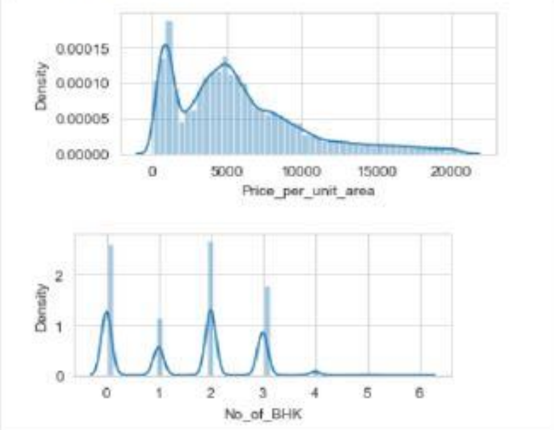

200000
150000
100000
50000
0
count
Unfurnished
Semi-Furnished
Furnished
is_furnished

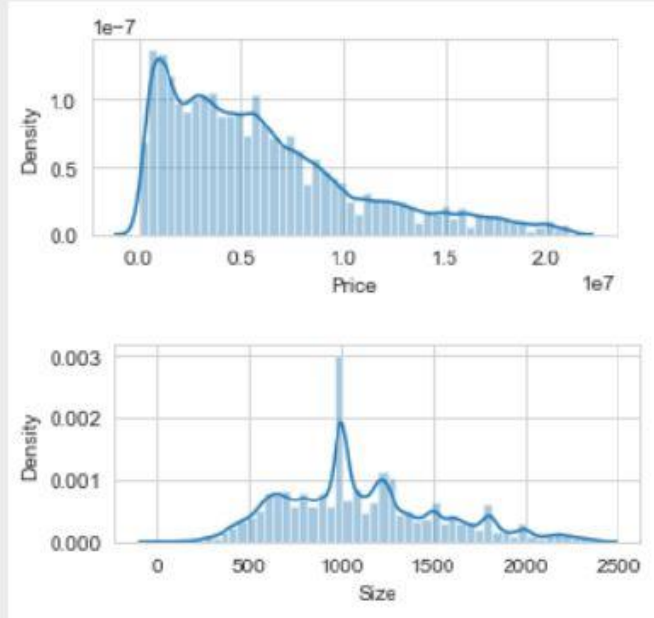

200000
150000
100000
50000
0
count
False
True
is_RERA_registered


200000
100000
0
count
False
True
is_PentaHouse

2.Numerical Columns:

```
t=1
for i in df.select_dtypes('number').columns:
    plt.subplot(4,2,t)
    sns.distplot(df[i])
    t+=1
    plt.show()
```





Business Insights:

After the univariate analysis, we inferred that there is mostly Apartments with some residential plots and least villas in property type.
Most of the property for sale is seen in Mumbai while least in Ahmedabad.
Properties we have seen in the data are mostly unfurnished with very less semifurnished and least in furnished state.
If we see the data most of the property has 2-3 no. of bhk and after 4bhk this no. has minimal values.
There is a huge spike in the size of the property at 1000sqft.
No. of properties are exponentially decreasing as the price is increasing.

Basic Model

We apply OLS Model as the base model:

OLS Regression Results

Dep. Variable:	Price	R-squared:	0.917
Model:	OLS	Adj. R-squared:	0.917
Method:	Least Squares	F-statistic:	2.422e+04
Date:	Thu, 06 Oct 2022	Prob (F-statistic):	0.00
Time:	20:49:30	Log-Likelihood:	-4.0698e+06
No. Observations:	262126	AIC:	8.140e+06
Df Residuals:	262006	BIC:	8.141e+06
Df Model:	119		
Covariance Type:	nonrobust		

Omnibus:	23475.577	Durbin-Watson:	1.238
Prob(Omnibus):	0.000	Jarque-Bera (JB):	121203.335
Skew:	-0.284	Prob(JB):	0.00
Kurtosis:	6.282	Cond. No.	3.80e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.8e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

As we can see their is high multicollinearity in our base model. So we will further remove it.

Feature Engineering

1.Treatment of Multicollinearity

V.I.F:

VIF (Variance Inflation Factor) measures how much the behavior (variance) of an independent variable is influenced by its interaction/correlation with other independent variables. VIF allows a quick measure of how much a variable is contributing to the standard error in regression. When significant multicollinearity exist the VIF will be very large. Hence, VIF can be used to eliminate columns that cause multicollinearity.

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
# the independent variables set
X = x

# VIF dataframe
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns

# calculating VIF for each feature
vif_data["VIF"] = [variance_inflation_factor(X.values, i)
                          for i in range(len(X.columns))]

print(vif_data)
```

```
                           feature         VIF
0              Price_per_unit_area    3.701914
1                        No_of_BHK    8.017027
2                             Size    2.637920
3    Property_type_Independent Floor    1.938276
4    Property_type_Independent House    1.185258
..                             ...         ...
115        is_furnished_Semi-Furnished    5.602790
116          is_furnished_Unfurnished    6.004452
117           is_RERA_registered_True    4.013846
118              is_PentaHouse_True    1.003099
119                           cons  606.083308

[120 rows x 2 columns]
```

Here we saw that two columns City_name and Sub_urban_name have very high VIF values. So, to reduce multicollinearity we had had to drop either one of them.

```python
df.drop('Sub_urban_name',axis=1,inplace=True)
# To reduce multicollinearity we either had to drop City_name or Sub_urban_name
# Sub_urban_name had less relevance with the data hence we dropped it.
```

After dropping such columns we check the VIF again.

```python
X = x

# VIF dataframe
vif_data1 = pd.DataFrame()
vif_data1["feature"] = X.columns

# calculating VIF for each feature
vif_data1["VIF"] = [variance_inflation_factor(X.values, i)
                          for i in range(len(X.columns))]

print(vif_data1)
```

```
                               feature        VIF
0                  Price_per_unit_area   2.325873
1                            No_of_BHK   7.817904
2                                 Size   2.438957
3      Property_type_Independent Floor   1.700802
4      Property_type_Independent House   1.146355
5       Property_type_Residential Plot   9.409755
6                  Property_type_Villa   1.173267
7   Property_status_Under Construction   3.536707
8     Property_building_status_INACTIVE  1.001031
9   Property_building_status_UNVERIFIED  2.139271
10                 City_name_Bangalore   4.508415
11                   City_name_Chennai   5.231284
12                     City_name_Delhi   4.397494
13                 City_name_Hyderabad   4.852101
14                   City_name_Kolkata   3.373291
15                   City_name_Lucknow   5.756078
16                    City_name_Mumbai   8.135273
17         is_furnished_Semi-Furnished   5.560586
18           is_furnished_Unfurnished   5.963044
19             is_RERA_registered_True   3.874158
20                 is_PentaHouse_True   1.001721
21                                cons  79.245379
```

As we can see that the VIF of every column is less than 10. Hence we can say that we have reduced the multicollinearity.

2.Tranformations

We apply transformation to make the columns normally distributed. As we seen earlier in univariate analysis that Price and Price_per_unit_area are right skewed so we apply square root transformations on it.

```
# apply transformations on Price,price per unit area are right skew- sqrt transformation
df1['Price_per_unit_area']=np.sqrt(df1['Price_per_unit_area'])
df1['Price']=np.sqrt(df1['Price'])
```

3.Scaling

We scale data when the range of the variables are not equal and we want to scale our variables in the same range. Here we apply Standard Scaler because it works better on normally distributed data. Standard Scaler is the type of scaling where the mean is 0 and the variance is 1.

```
#Scaling the data
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
```

```
scaled_df1 = df1
```

```
#Standard scale No_of_BHK, Price_per_unit_area, Size
scaled_df1['No_of_BHK']=sc.fit_transform(df1[['No_of_BHK']])
scaled_df1['Price_per_unit_area']=sc.fit_transform(df1[['Price_per_unit_area']])
scaled_df1['Size']=sc.fit_transform(df1[['Size']])
```

We didn’t scale the target variable.

Comparison and Selection of model

Final OLS model:

OLS Regression Results

Dep. Variable:	Price	R-squared:	0.969
Model:	OLS	Adj. R-squared:	0.969
Method:	Least Squares	F-statistic:	3.911e+05
Date:	Thu, 06 Oct 2022	Prob (F-statistic):	0.00
Time:	21:04:13	Log-Likelihood:	-1.7143e+06
No. Observations:	262126	AIC:	3.429e+06
Df Residuals:	262104	BIC:	3.429e+06
Df Model:	21		
Covariance Type:	nonrobust		

Omnibus:	71725.456	Durbin-Watson:	1.133
Prob(Omnibus):	0.000	Jarque-Bera (JB):	323636.325
Skew:	-1.274	Prob(JB):	0.00
Kurtosis:	7.811	Cond. No.	156.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

As we can see that we have removed the multicollinearity. And the R2 value is also 0.969 meaning that 96.9% of the variation in the target variable is due to the independent variables. Meaning our model is pretty great.
But R2 value being this high can lead to overfitting hence we check other models as well before finalizing our model.

Model Selection:

We created a function that creates Linear Regression, Lasso and Decision Tree model and compare the R2 value of each model and using the GridSearchCV we find the optimum Hyperparameters for our model.

```
from sklearn.model_selection import GridSearchCV

from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor

def find_best_model_using_gridsearchcv(x,y):
    algos = {
        'linear_regression': {
            'model':LinearRegression(),
            'params':{
                'normalize': [True,False]
            }
        },
         'lasso': {
             'model':Lasso(),
            'params':{
                'alpha':[1,2],
                'selection':['random','cyclic']
         }
         },
        'decision_tree': {
            'model':DecisionTreeRegressor(),
            'params':{
                'criterion': ['mse','friedman_mse'],
                'splitter': ['best','random']
            }
        }
    }
```

```python
    scores = []
    cv = ShuffleSplit(n_splits=5,test_size=0.2,random_state=0)
    for algo_name,config in algos.items():
        gs = GridSearchCV(config['model'],config['params'],cv=cv,return_train_score=False)
        gs.fit(x,y)
        scores.append({
            'model': algo_name,
            'best_score': gs.best_score_,
            'best_params': gs.best_params_
        })

    return pd.DataFrame(scores,columns=['model','best_score','best_params'])

find_best_model_using_gridsearchcv(x,y)
```

	model	best_score	best_params
0	linear_regression	0.968989	{'normalize': True}
1	lasso	0.968605	{'alpha': 1, 'selection': 'cyclic'}
2	decision_tree	0.999122	{'criterion': 'mse', 'splitter': 'best'}

Result and Conclusion

Looking at the different R2 value we came to the conclusion that for our problem the Random Forest is the best model. With the R2 value of 0.99912.

Bibliography

The dataset that we have used in this project is from Kaggle:
[Indian Cities Housing Property Price Dataset | Kaggle]