Project report on

MAKAAN PROPERTY HOUSE PRICE PREDICTION ANALYSIS

Submitted by

Ankita Burde

Under the guidance of

Mr. P.Bose

(Top Mentor)

1. Overview

People and real estate agencies buy or sell properties in order to either live in or as an investment or to run a business. There are multiple factors on which price of a property depends like city, location, size and sometimes the name of the builder can also be a deciding factor. Hence, building a model to predict the price of the property can help the customer as well as company. In this project, we are working on the dataset of the company Makaan.com for house price prediction. Makaan.com has quickly emerged as the preferred partner for consumers looking to rent, buy or sell a home. Makaan.com offers its online consumers maximum property options and has become one of the largest advertising platforms in online real estate in India.

2. Problem Statement

The company wants us to predict house prices depending on the various aspects with the help of Machine Learning model.

3. Data Description

Table -1 Makaan_Properties_Details(Details about the properties/different features)

a) Property Name
b) Property id
c) Property type
d) Property status
e) Price_per_unit_area
f) Posted On
g) Project URL
h) Builder id
i) Builder name
j) Property_building_status
k) City_id
l) City name
m) No_of_BHK
n) Locality_ID
o) Locality Name
p) Longitude
q) Latitude
r) Price
s) Size
t) Sub_urban_ID
u) Sub_urban_name
v) description

w) is furnished
x) listing_domain_score
y) is plot
z) is_RERA_registered
aa) is Apartment
bb) is_ready_to_move
cc) is_commercial_Listing
dd) is_PentaHouse
ee) is studio
ff) Listing Category

Table -2 Makaan_property_location_details

a) Property_id
b) City_id
c) City name
d) Locality_ID
e) Locality Name
f) Longitude
g) Latitude
h) Sub_urban_ID
i) Sub_urban_name

4. Loading the Dataset in Jupyter Notebook

```
# importing the packages--
import pandas as pd
import numpy as np
import math as m
import seaborn as sns
import sqlalchemy as sa
sns.set_style("whitegrid")
import plotly.express as px
import matplotlib.pyplot as plt
import pymysql
import mysql.connector
from sqlalchemy import create_engine
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import LinearSVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
import joblib
```

- Reading 1st Dataset

```
# reading the data file--
def read_data1():
    data1=pd.read_csv("C:/ANKITA/ASSIGNMENTS/Day 29 - 18th June/Capstone Project/Makaan_Properties_Details.csv",encoding='latin1
    return data1
```

```
# calling the function--
data1= read_data1()
print(data1.head(5))
```

```
              Property_Name  Property_id Property_type     Property_status  \
0              Arkiton Luxe     15446514     Apartment  Under Construction
1  Keshav Akshar Ocean Pearl     15367414     Apartment  Under Construction
2           Vishwa Opulence     14683118     Apartment        Ready to move
3             Satyam Sarjan      5476295     Apartment        Ready to move
4          Navkar Sunflower     15477040     Apartment  Under Construction

  Price_per_unit_area   Posted_On  \
0               4,285   1 day ago
1               7,000  2 days ago
2               5,752  2 days ago
3               2,486  5 days ago
4               5,324  8 days ago

                                        Project_URL   builder_id  \
0  https://www.makaan.com/ahmedabad/arkiton-life-...  100563465.0
1  https://www.makaan.com/ahmedabad/keshav-naraya...  100009433.0
2  https://www.makaan.com/ahmedabad/vishwa-develo...  100207731.0
3  https://www.makaan.com/ahmedabad/satyam-develo...     101303.0
4  https://www.makaan.com/ahmedabad/navkar-buildc...    1484289.0

                Builder_name Property_building_status  ... is_furnished  \
0          Arkiton life Space                  ACTIVE  ...  Unfurnished
1         Keshav Narayan Group                 ACTIVE  ...  Unfurnished
2  Vishwa Developers Ahmedabad                 ACTIVE  ...  Unfurnished
3           Satyam Developers                  ACTIVE  ...  Unfurnished
4    Navkar Buildcon Ahmedabad                 ACTIVE  ...  Unfurnished

  listing_domain_score is_plot is_RERA_registered is_Apartment  \
0                  4.0   False               True          True
1                  4.0   False               True          True
2                  4.0   False              False          True
3                  4.0   False              False          True
4                  4.0   False               True          True

   is_ready_to_move  is_commercial_Listing  is_PentaHouse  is_studio  \
0             False                 False          False      False
1             False                 False          False      False
2              True                 False          False      False
3              True                 False          False      False
4             False                 False          False      False

   Listing_Category
0              sell
1              sell
2              sell
3              sell
4              sell

[5 rows x 24 columns]
```

- Reading 2nd Dataset

```
# reading the data file--
def read_data2():
    data2=pd.read_csv("C:/ANKITA/ASSIGNMENTS/Day 29 - 18th June/Capstone Project/Makaan_property_location_details.csv")
    return data2
```

```
# calling the function--
data2= read_data2()
print(data2.head(2))
```

```
   Property_id  City_id  City_name  Locality_ID Locality_Name  Longitude  \
0     15579866        1  Ahmedabad        51749      Bodakdev  72.520195
1     15579809        1  Ahmedabad        51749      Bodakdev  72.502571

    Latitude  Sub_urban_ID Sub_urban_name
0  23.040195         10003     SG Highway
1  23.032154         10003     SG Highway
```

- Merging both the Datasets

```
# merging both the datasets--
data=data1.merge(data2,left_on='Property_id', right_on='Property_id',how = 'inner')
pd.set_option("display.max.columns",None)
data.head(2)
```

	Property_Name	Property_id	Property_type	Property_status	Price_per_unit_area	Posted_On	Project_URL	builder_id	Builder_
0	Arkiton Luxe	15446514	Apartment	Under Construction	4,285	1 day ago	https://www.makaan.com/ahmedabad/arkiton-life-...	100563465.0	Arkit
1	Arkiton Luxe	15446514	Apartment	Under Construction	4,285	1 day ago	https://www.makaan.com/ahmedabad/arkiton-life-...	100563465.0	Arkit

- Data Cleaning & Data Type Conversions

```
print(len(data[data['No_of_BHK']=='1 RK']))
print(len(data[data['No_of_BHK']=='2 RK']))
print(len(data[data['No_of_BHK']=='3 RK']))
```

```
7271
2
4
```

```
data.drop(data[(data['No_of_BHK']=='1 RK') |(data['No_of_BHK']=='2 RK') | (data['No_of_BHK']=='3 RK')].index,inplace=True)
data['No_of_BHK'].unique()
```

```
array(['3 BHK', '4 BHK', '2 BHK', '5 BHK', '1 BHK', '0 BHK', '12 BHK',
       '7 BHK', '6 BHK', '8 BHK', '10 BHK', '11 BHK', '9 BHK', '15 BHK',
       '14 BHK'], dtype=object)
```

```
# IN THE DATA '0 BHK' ARE THE'RESIDENTIAL PLOTS'
data[data['No_of_BHK']==0]

data[data['Property_type']=='Residential Plot'].head(2)
```

	Property_Name	Property_id	Property_type	Property_status	Price_per_unit_area	Posted_On	Project_URL	builder_id	Builder
1191	NaN	15528030	Residential Plot	Ready to move	13,650	9 days ago	https://www.makaan.com/ahmedabad/builder-proje...	NaN	
1192	NaN	15528240	Residential Plot	Ready to move	518	9 days ago	https://www.makaan.com/ahmedabad/builder-proje...	NaN	

```
data['Price_per_unit_area'] = data['Price_per_unit_area'].replace(',', '',regex=True)
data['Price_per_unit_area']=data['Price_per_unit_area'].astype(int)

data['Price'] = data['Price'].replace(',', '',regex=True)
data['Price']=data['Price'].astype(int)

data['Size']=data['Size'].replace("sq ft","",regex=True)
data['Size']=data['Size'].replace(",","",regex=True)
data['Size']=data['Size'].astype(int)

data['No_of_BHK']=data['No_of_BHK'].replace('BHK', '',regex=True)
data['No_of_BHK']=data['No_of_BHK'].astype(int)

data['is_RERA_registered']=data['is_RERA_registered'].astype('object')
data['is_PentaHouse']=data['is_PentaHouse'].astype('object')
```

- Dropping the columns

```
data.drop(columns=['Property_id','Posted_On', 'Project_URL','builder_id','Builder_name','description',
                   'listing_domain_score','Listing_Category', 'City_id','Locality_ID','Longitude', 'Latitude','Sub_urban_ID'],inpl
data.head(2)
```

	Property_Name	Property_type	Property_status	Price_per_unit_area	Property_building_status	No_of_BHK	Price	Size	Is_furnished	Is_plot	Is_RERA_reg
0	Arkiton Luxe	Apartment	Under Construction	4285	ACTIVE	3	7500000	1750	Unfurnished	False	
1	Arkiton Luxe	Apartment	Under Construction	4285	ACTIVE	3	7500000	1750	Unfurnished	False	

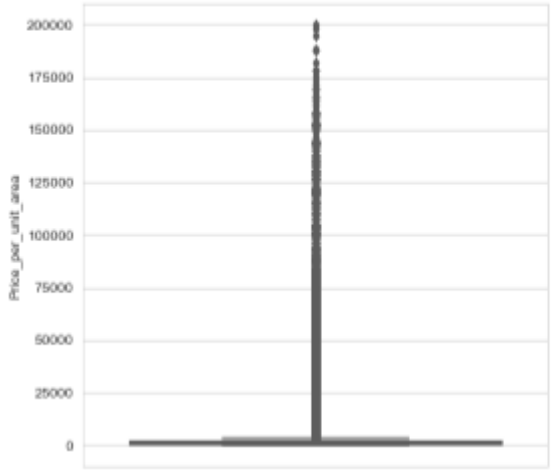
```
data.drop(['is_plot','is_Apartment','is_ready_to_move','is_commercial_Listing','is_studio'],axis=1,inplace=True)
data.head(2)
```

	Property_Name	Property_type	Property_status	Price_per_unit_area	Property_building_status	No_of_BHK	Price	Size	Is_furnished	Is_RERA_registered
0	Arkiton Luxe	Apartment	Under Construction	4285	ACTIVE	3	7500000	1750	Unfurnished	True
1	Arkiton Luxe	Apartment	Under Construction	4285	ACTIVE	3	7500000	1750	Unfurnished	True

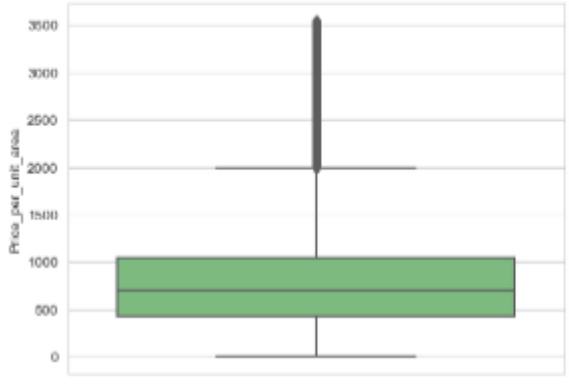
```
data.drop("Sub_urban_name",axis=1,inplace=True)
data.drop("Locality_Name",axis=1,inplace=True)
data.head(2)
```

- Outlier Treatment

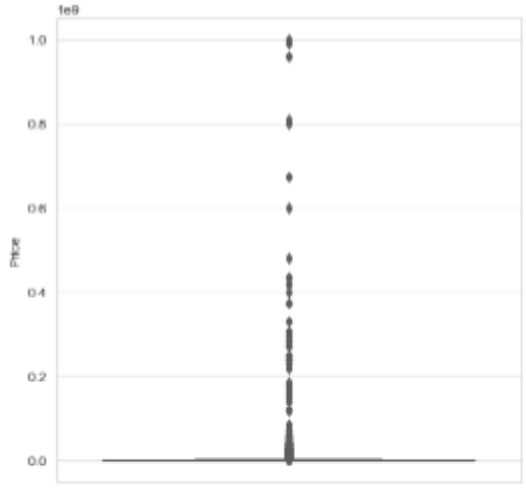
```
plt.figure(figsize=(6,6))
x = sns.boxplot(data=data,y="Price_per_unit_area",palette='Greens')
```



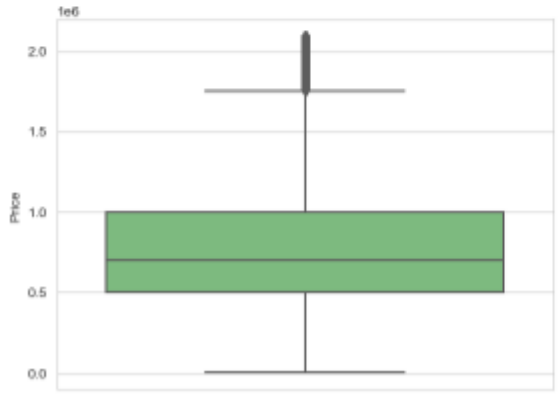

```
Q1 = data.Price_per_unit_area.quantile(0.25)
Q3 = data.Price_per_unit_area.quantile(0.75)
IQR = Q3 - Q1
data = data[(data.Price_per_unit_area >= Q1 - 1.5*IQR) & (data.Price_per_unit_area <= Q3 + 1.5*IQR)]

sns.boxplot(y=data.Price_per_unit_area,data=data,palette='Greens')
plt.show()
```



```
plt.figure(figsize=(6,6))
x = sns.boxplot(data=data,y="Price",palette='Greens')
```



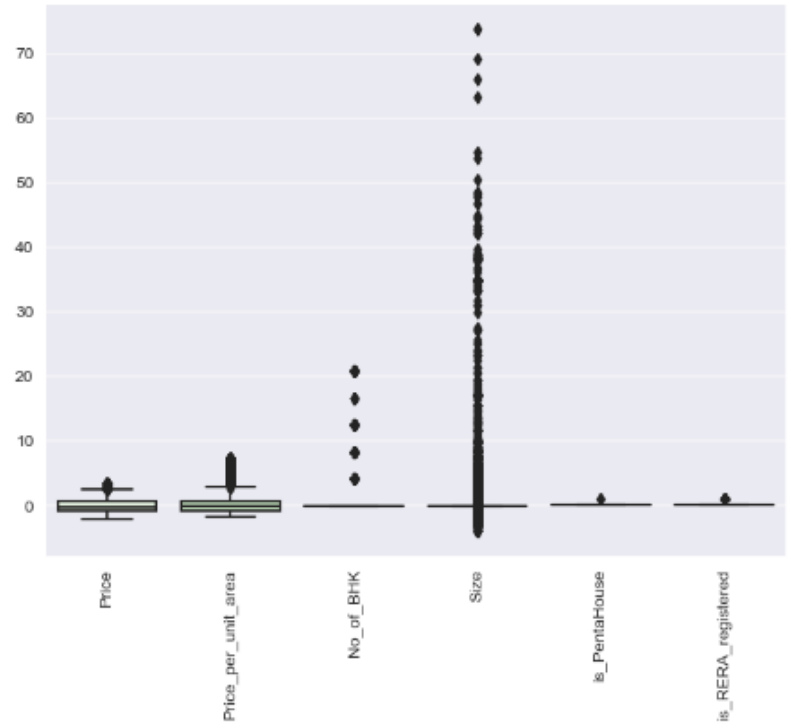

```
Q1 = data.Price.quantile(0.25)
Q3 = data.Price.quantile(0.75)
IQR = Q3 - Q1
data = data[(data.Price >= Q1 - 1.5*IQR) & (data.Price <= Q3 + 1.5*IQR)]

sns.boxplot(y=data.Price,data=data,palette='Greens')
plt.show()
```



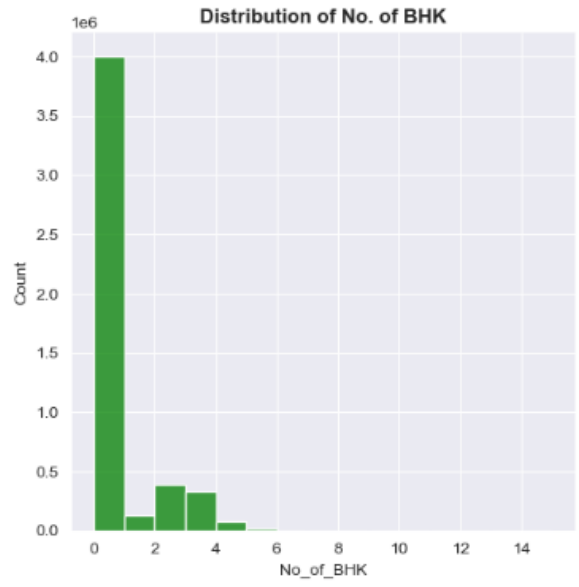
- Exploratory Data Analysis

```
# checking the outliers--
sns.set_style("darkgrid")
plt.figure(figsize=(8,6))
plt.xticks(rotation=90,fontsize="medium")
sns.boxplot(data=data.loc[:, ['Price','Price_per_unit_area', 'No_of_BHK', 'Size','is_PentaHouse','is_RERA_registered']],palette=
```

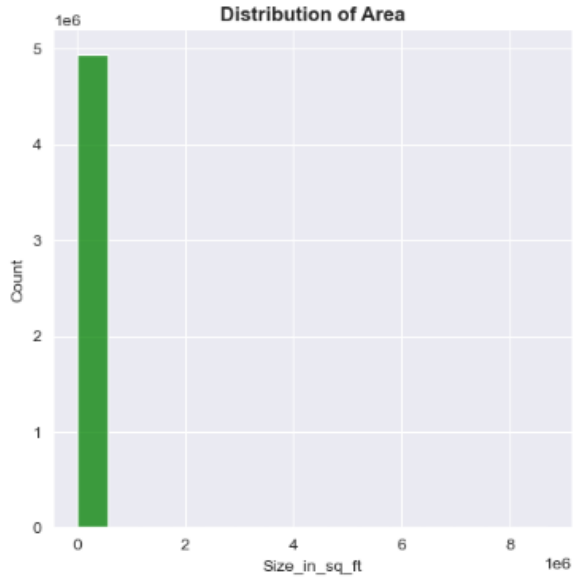
```
<Axes: >
```




```
sns.displot(x=data["No_of_BHK"],bins=15,color="green")
data["No_of_BHK"].mean()
plt.title("Distribution of No. of BHK",fontweight="bold")
plt.show()
```

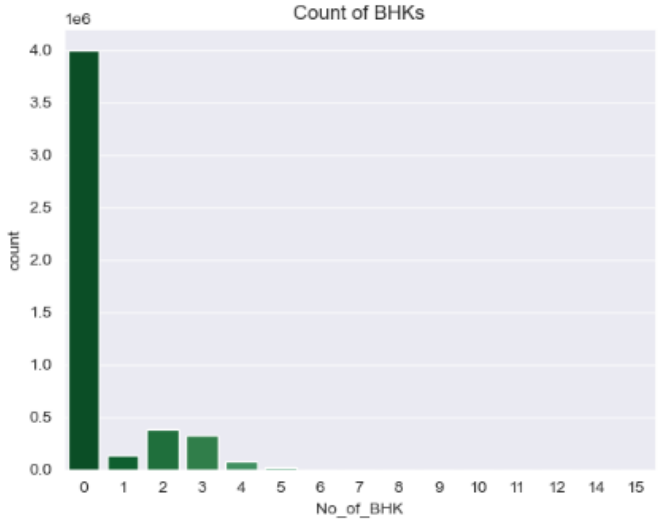


```
sns.displot(x=data["Size_in_sq_ft"],bins=15,color="green")
data["Size_in_sq_ft"].mean()
plt.title("Distribution of Area",fontweight="bold")
plt.show()
```



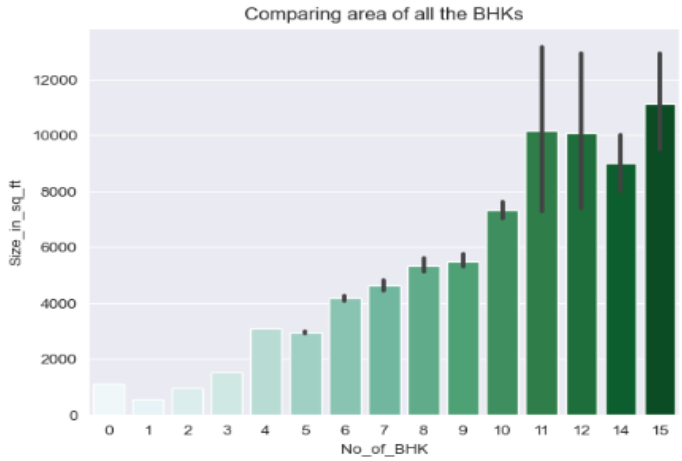

```
sns.countplot(x="No_of_BHK",data=data,palette='BuGn_r')
plt.title("Count of BHKs")
```

```
Text(0.5, 1.0, 'Count of BHKs')
```



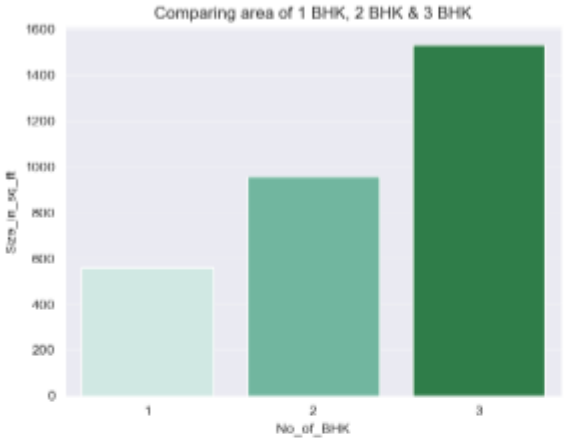
```
dfn=data[(data["No_of_BHK"]==0)|(data["No_of_BHK"]==1)|(data["No_of_BHK"]==2)|(data["No_of_BHK"]==3)|(data["No_of_BHK"]==4)
        |(data["No_of_BHK"]==5)|(data["No_of_BHK"]==6)|(data["No_of_BHK"]==7)|(data["No_of_BHK"]==8)|(data["No_of_BHK"]==9)
        |(data["No_of_BHK"]==10)|(data["No_of_BHK"]==11)|(data["No_of_BHK"]==12)|(data["No_of_BHK"]==14)|(data["No_of_BHK"]==15)
sns.barplot(x="No_of_BHK",y="Size_in_sq_ft",data=dfn,palette='BuGn')
plt.title("Comparing area of all the BHKs")
```

```
Text(0.5, 1.0, 'Comparing area of all the BHKs')
```



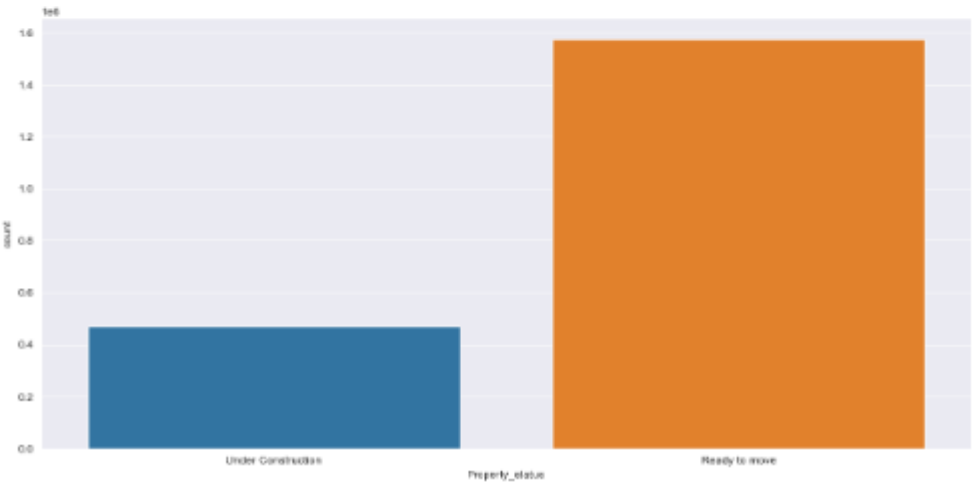

```
dfn=data[(data["No_of_BHK"]==1)|(data["No_of_BHK"]==2)|(data["No_of_BHK"]==3)]
sns.barplot(x="No_of_BHK",y="Size_in_sq_ft",data=dfn,palette='BuGn')
plt.title("Comparing area of 1 BHK, 2 BHK & 3 BHK")
```

```
Text(0.5, 1.0, 'Comparing area of 1 BHK, 2 BHK & 3 BHK')
```



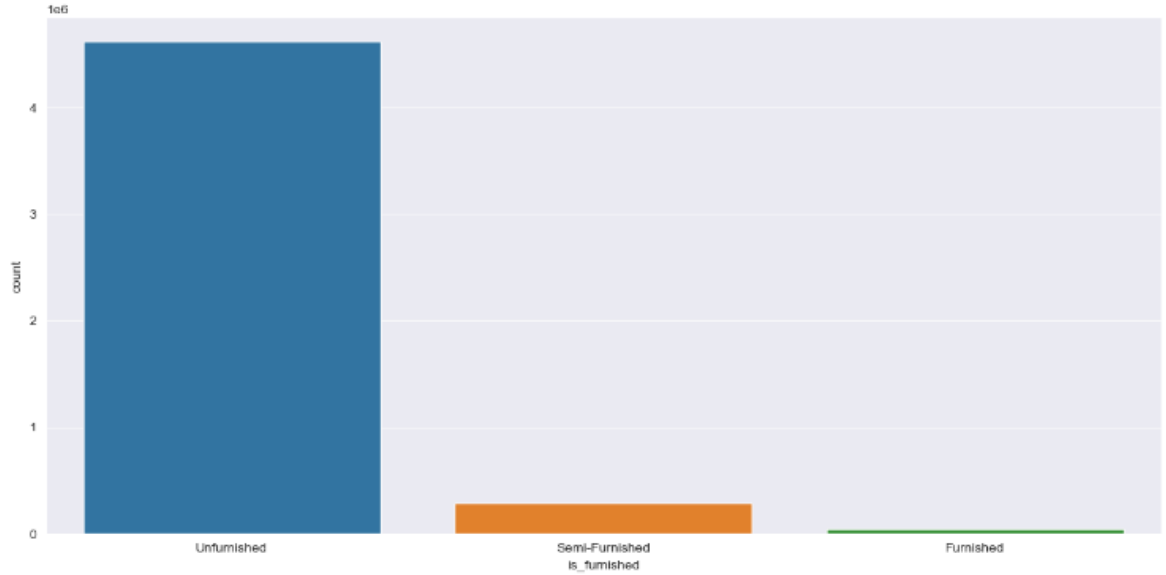

```
plt.figure(figsize = (15,7))
sns.countplot(x = data['Property_status'])
```

```
<Axes: xlabel='Property_status', ylabel='count'>
```



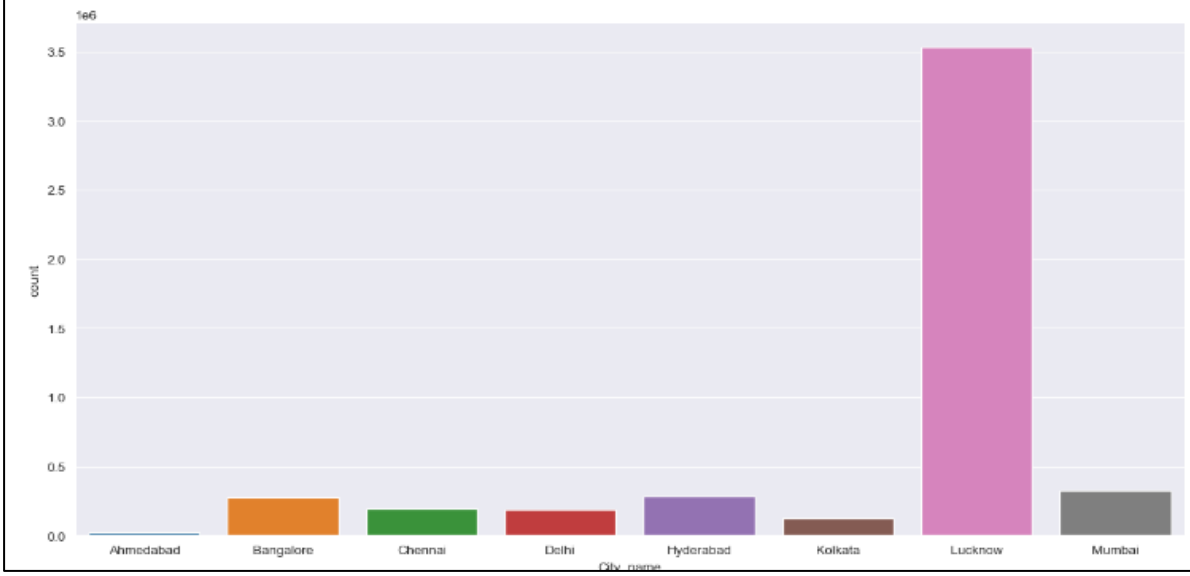
```
plt.figure(figsize = (15,7))
sns.countplot(x = data['is_furnished'])
```

```
<Axes: xlabel='is_furnished', ylabel='count'>
```



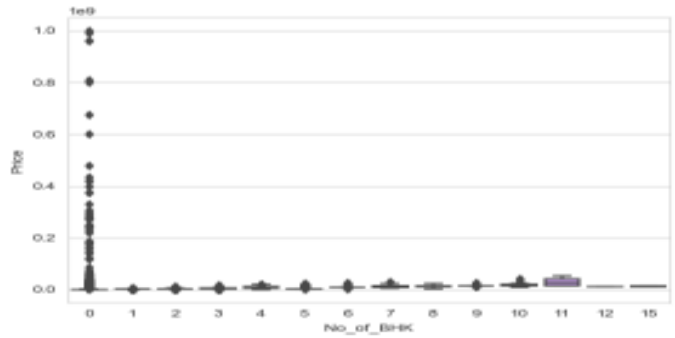

```
plt.figure(figsize = (15,7))
sns.countplot(x = data['City_name'])
```

```
<Axes: xlabel='City_name', ylabel='count'>
```



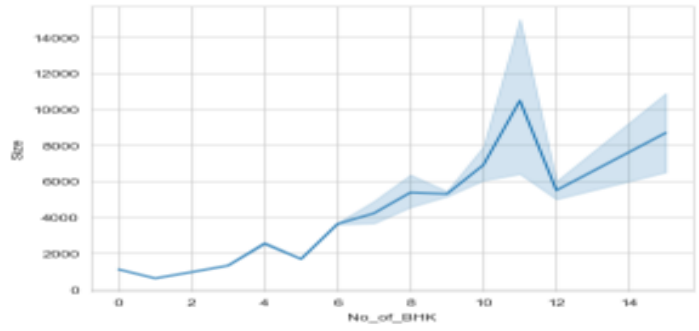
```
plt.figure(dpi = 100)
sns.boxplot(x = data.No_of_BHK, y = data.Price)
```

```
<Axes: xlabel='No_of_BHK', ylabel='Price'>
```



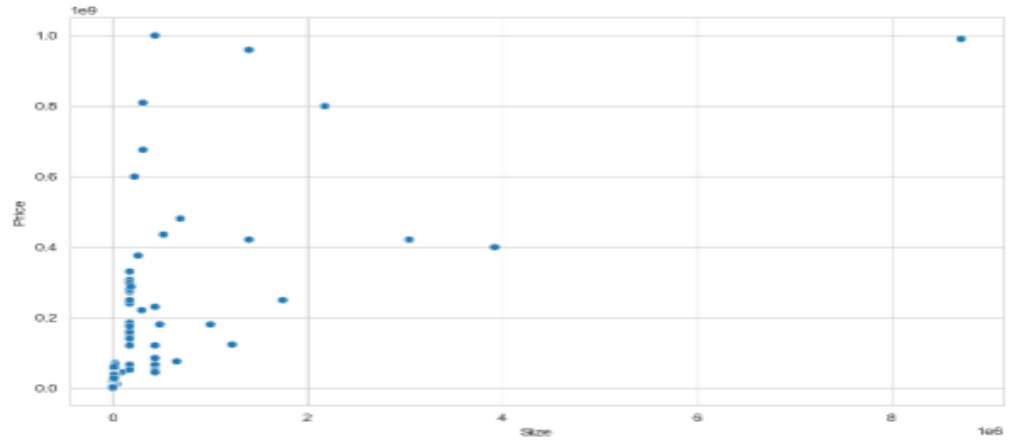

```
sns.lineplot(x=data["No_of_BHK"], y=data["Size"])
```

```
<Axes: xlabel='No_of_BHK', ylabel='Size'>
```



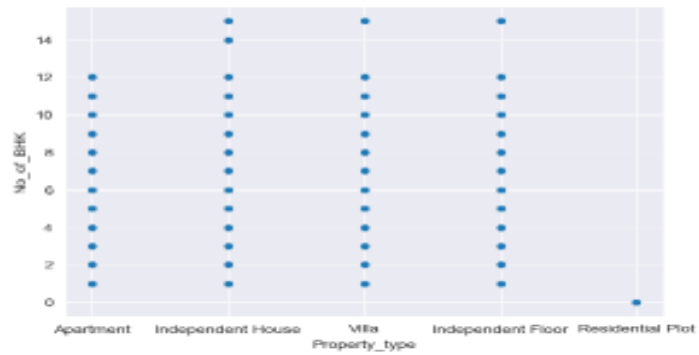

```
plt.figure(figsize = (10,6))
sns.scatterplot(x = data.Size, y = data.Price)
```

```
<Axes: xlabel='Size', ylabel='Price'>
```



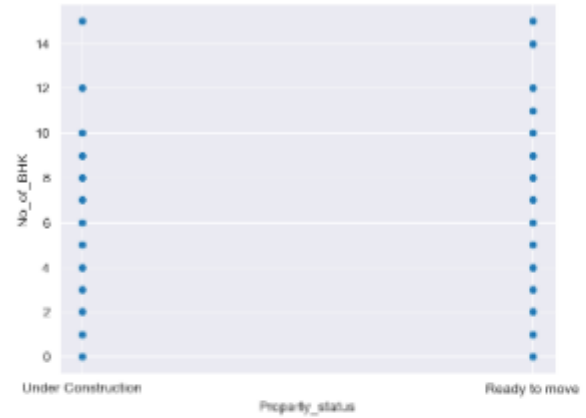

```
plt.figure(dpi = 100)
sns.scatterplot(x = data.Property_type, y = data.No_of_BHK)
```

```
<Axes: xlabel='Property_type', ylabel='No_of_BHK'>
```



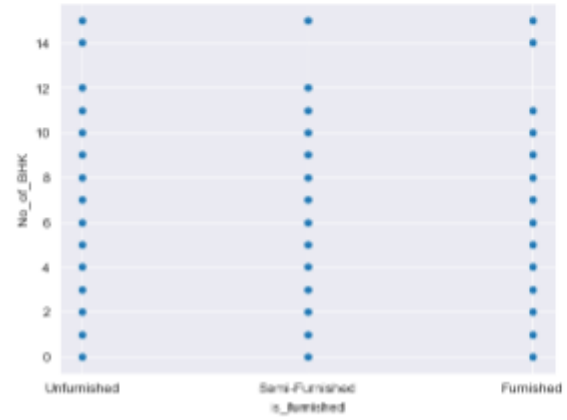
```
plt.figure(dpi = 100)
sns.scatterplot(x = data.Property_status, y = data.No_of_BHK)
```

```
<Axes: xlabel='Property_status', ylabel='No_of_BHK'>
```



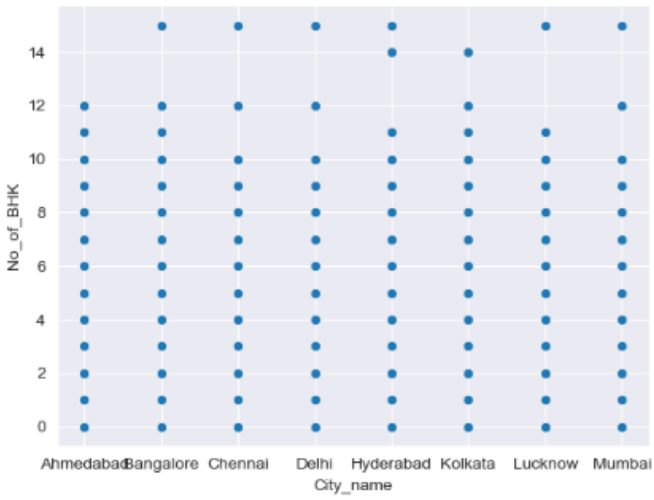

```
plt.figure(dpi = 100)
sns.scatterplot(x = data.is_furnished, y = data.No_of_BHK)
```

```
<Axes: xlabel='is_furnished', ylabel='No_of_BHK'>
```



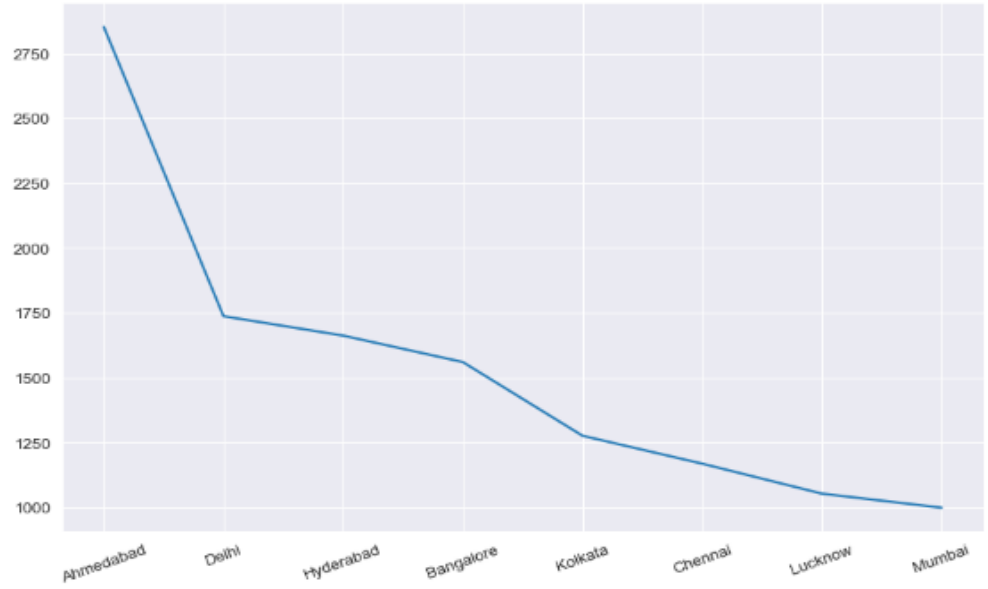

```
plt.figure(dpi = 100)
sns.scatterplot(x = data.City_name, y = data.No_of_BHK)
```

```
<Axes: xlabel='City_name', ylabel='No_of_BHK'>
```



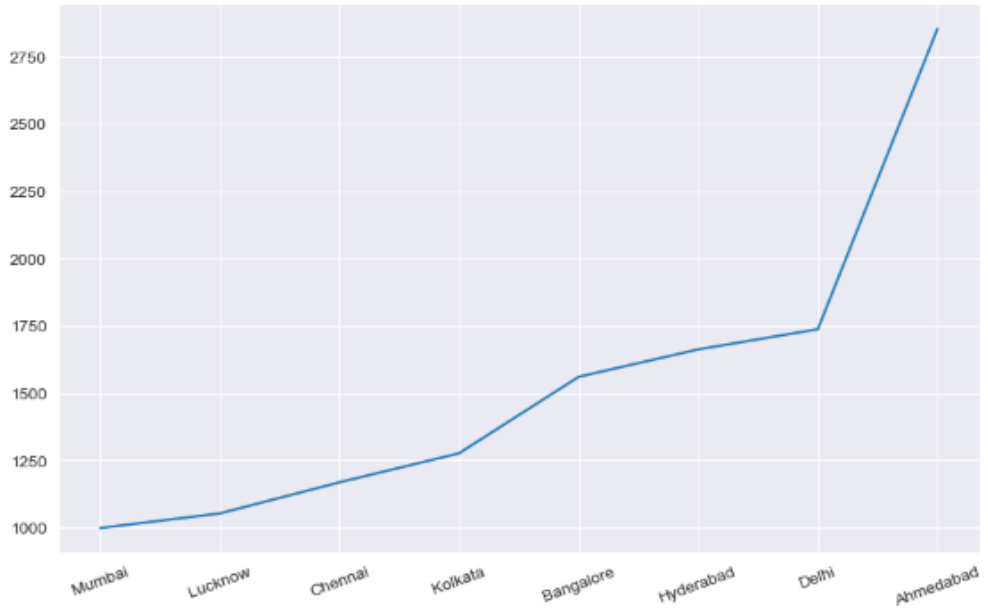
```
plt.figure(figsize = (10,6))
plt.plot(data.groupby('City_name')['Size_in_sq_ft'].mean().nlargest(10))
plt.xticks(rotation = 20)
```

```
([0, 1, 2, 3, 4, 5, 6, 7],
 [Text(0, 0, 'Ahmedabad'),
  Text(1, 0, 'Delhi'),
  Text(2, 0, 'Hyderabad'),
  Text(3, 0, 'Bangalore'),
  Text(4, 0, 'Kolkata'),
  Text(5, 0, 'Chennai'),
  Text(6, 0, 'Lucknow'),
  Text(7, 0, 'Mumbai')])
```



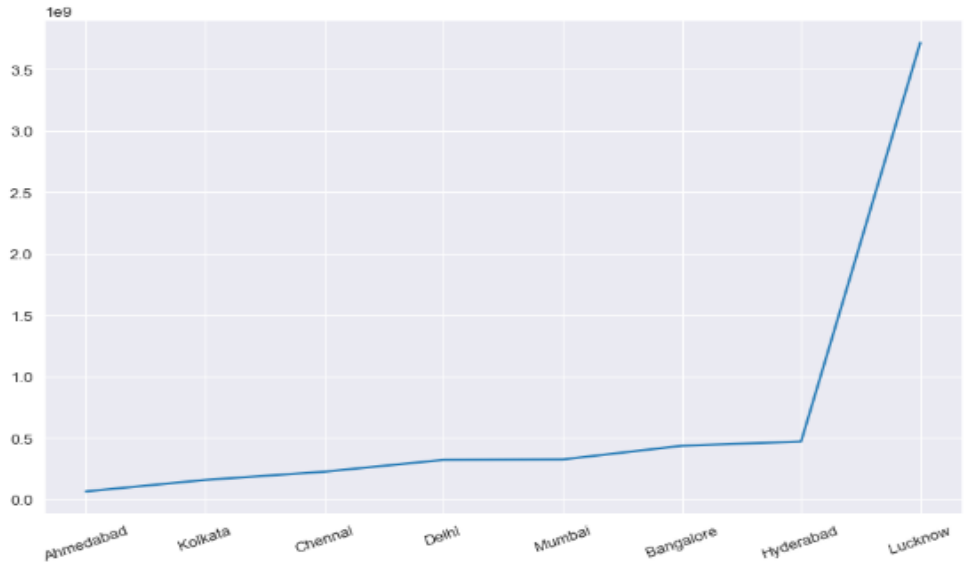

```
plt.figure(figsize = (10,6))
plt.plot(data.groupby('City_name')['Size_in_sq_ft'].mean().nsmallest(10))
plt.xticks(rotation = 20)
```

```
([0, 1, 2, 3, 4, 5, 6, 7],
 [Text(0, 0, 'Mumbai'),
  Text(1, 0, 'Lucknow'),
  Text(2, 0, 'Chennai'),
  Text(3, 0, 'Kolkata'),
  Text(4, 0, 'Bangalore'),
  Text(5, 0, 'Hyderabad'),
  Text(6, 0, 'Delhi'),
  Text(7, 0, 'Ahmedabad')])
```



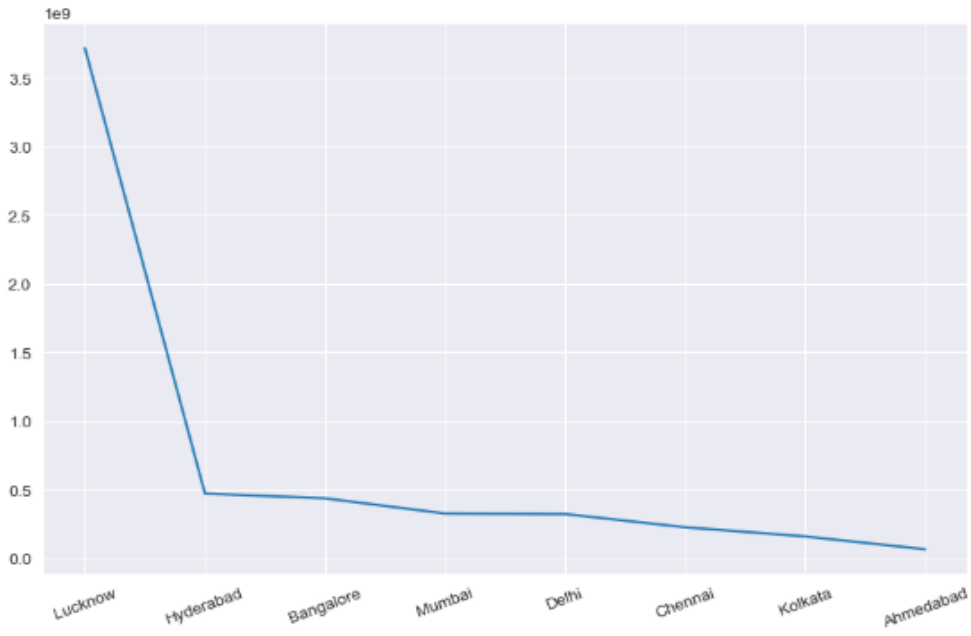
```
plt.figure(figsize = (10,6))
plt.plot(data.groupby('City_name')['Size_in_sq_ft'].sum().nsmallest(10))
plt.xticks(rotation = 20)
```

```
([0, 1, 2, 3, 4, 5, 6, 7],
 [Text(0, 0, 'Ahmedabad'),
  Text(1, 0, 'Kolkata'),
  Text(2, 0, 'Chennai'),
  Text(3, 0, 'Delhi'),
  Text(4, 0, 'Mumbai'),
  Text(5, 0, 'Bangalore'),
  Text(6, 0, 'Hyderabad'),
  Text(7, 0, 'Lucknow')])
```




```
plt.figure(figsize = (10,6))
plt.plot(data.groupby('City_name')['Size_in_sq_ft'].sum().nlargest(10))
plt.xticks(rotation = 20)
```

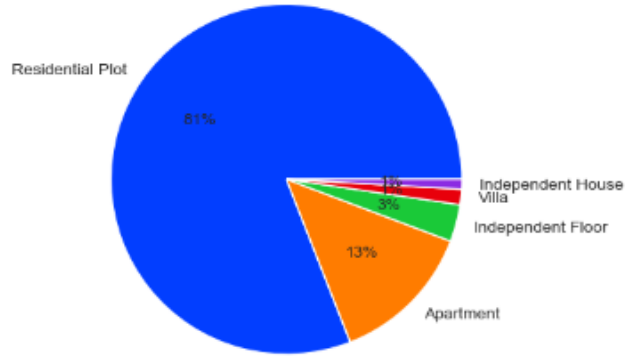
```
([0, 1, 2, 3, 4, 5, 6, 7],
 [Text(0, 0, 'Lucknow'),
  Text(1, 0, 'Hyderabad'),
  Text(2, 0, 'Bangalore'),
  Text(3, 0, 'Mumbai'),
  Text(4, 0, 'Delhi'),
  Text(5, 0, 'Chennai'),
  Text(6, 0, 'Kolkata'),
  Text(7, 0, 'Ahmedabad')])
```



```
a = data.Property_type.value_counts()
a
```

```
Residential Plot     4000379
Apartment             657676
Independent Floor     169424
Villa                  69972
Independent House      45253
Name: Property_type, dtype: int64
```

```
plt.figure()
colours = sns.color_palette('bright')[0:5]
labels = a.keys()
plt.pie(a, colors = colours, labels = labels, autopct='%.0f%%')
plt.show()
```

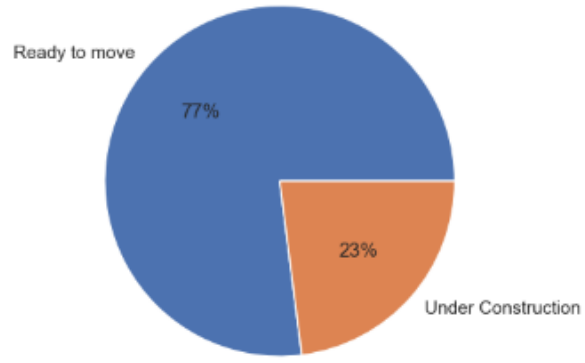


In the given dataset, 81% of the properties are Residential plots, 13% of the properties are Apartments, 3% of the properties are Independent Floor, 1% of the properties are Villa and 1% of the properties are Independent House.

```
b = data.Property_status.value_counts()
b
```

```
Ready to move         1576693
Under Construction     470570
Name: Property_status, dtype: int64
```

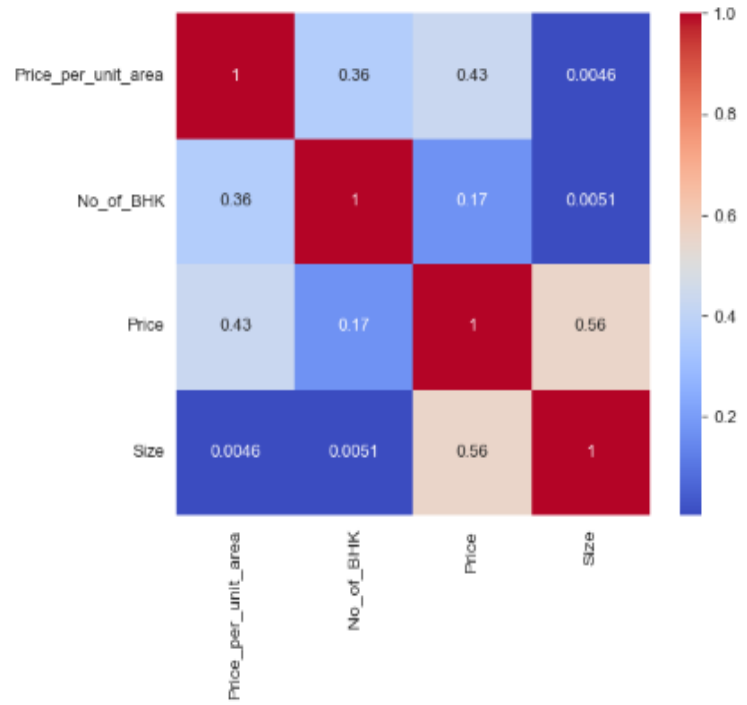
```
plt.pie(b, labels = b.keys(), autopct='%.0f%%')
plt.show()
```



In the given dataset, majority of the properties are in ready to move condition whereas minority are in under construction stage.

```
sns.heatmap(data [['Price_per_unit_area', 'No_of_BHK', 'Price', 'Size','is_RERA_registered','is_PentaHouse']].corr(),annot=True,
```

```
<Axes: >
```



- **Feature Engineering**

A. Encoding Labels:

```
# Label encoder--
lb= LabelEncoder()
data['is_RERA_registered']=lb.fit_transform(data['is_RERA_registered'])
data['is_PentaHouse']=lb.fit_transform(data['is_PentaHouse'])
data
```

	Property_Name	Price_per_unit_area	No_of_BHK	Price	Size	Is_RERA_registered	Is_PentaHouse	Property_type_Apartment	Property_type_Indepen F
269	Kaamnath Brahmdhara Residency	1828	2	1990892	1089	1	0	1	
272	Armaan Nandanvan Heights	2314	1	1750000	756	0	0	1	
281	Satyam Sarjan	2514	1	1607000	639	0	0	1	
1112	Indiabulls Centrum	3448	1	2000000	580	0	0	1	
1139	NaN	2949	2	2000000	678	0	0	0	
...	...	...	...	...	...	...	...	...	
4542672	NaN	624	0	1250000	2002	0	0	0	
4542677	NaN	2400	0	1200000	500	0	0	0	
4542682	Srusti Mount Valley	493	0	850000	1722	0	0	0	
4542684	Ranjanpada	229	0	250000	1089	0	0	0	
4542687	NaN	350	0	783650	2239	0	0	0	

3889142 rows × 28 columns

B. Dummy variables

```
data=pd.get_dummies(data,columns=['Property_type','Property_status','Property_building_status','is_furnished','City_name'])
data
```

	Property_Name	Price_per_unit_area	No_of_BHK	Price	Size	Is_RERA_registered	Is_PentaHouse	Property_type_Apartment	Property_type_Independ F
0	Arkiton Luxe	4285	3	7500000	1750	True	False	1	
1	Arkiton Luxe	4285	3	7500000	1750	True	False	1	
2	Arkiton Luxe	4285	3	7500000	1750	True	False	1	
3	Arkiton Luxe	4285	3	7500000	1750	True	False	1	
4	Arkiton Luxe	4285	3	7500000	1750	True	False	1	
...	...	...	...	...	...	...	...	...	
4542699	Rajlaxmi RaajLaxmi Towers	9626	1	3400000	346	True	False	1	
4542700	Rajlaxmi RaajLaxmi Towers	8568	1	3650000	426	True	False	1	
4542701	Rajlaxmi RaajLaxmi Towers	9861	1	3550000	360	True	False	1	
4542702	Rajlaxmi RaajLaxmi Towers	8813	2	5200000	590	True	False	1	
4542703	Rajlaxmi RaajLaxmi Towers	9659	1	3529577	358	True	False	1	

4935406 rows × 28 columns

C. Scaling of data

```
# scaling the data--
sc= StandardScaler()
sc1= StandardScaler()
scaled_data= data
scaled_data
#Standard scale No_of_BHK, Price_per_unit_area, Size
scaled_data['No_of_BHK']=sc.fit_transform(data[['No_of_BHK']])
scaled_data['Price_per_unit_area']=sc.fit_transform(data[['Price_per_unit_area']])
scaled_data['Size']=sc.fit_transform(data[['Size']])
data['Price']=sc1.fit_transform(data[['Price']])
df=scaled_data
```

- Model Building

```
# sampling of data--
train=data[data["Property_Name"]!='T']
test=data[data["Property_Name"]=='T']

X_train=pd.concat([train.iloc[:,2:6],train.iloc[:,9:29]],axis=1)
y_train=train.iloc[:,1]

X_test=pd.concat([test.iloc[:,2:6],test.iloc[:,9:29]],axis=1)
y_test=test.iloc[:,1]
```

```
# calling the train_test function--
print(X_train.shape),print(y_train.shape),print(X_test.shape),print(y_test.shape)
```

```
(2416372, 23)
(2416372,)
(1272770, 23)
(1272770,)

(None, None, None, None)
```

Linear Regression

```
# defining--
def modelling1(X_train,y_train,X_test):
    model1=LinearRegression()
    model1_train=model1.fit(X_train,y_train)
    print("Model 1 training completed.")
    return model1_train

print("Calling the modelling 1 function")
model1_train=modelling1(X_train,y_train,X_test)
```

```
Calling the modelling 1 function
Model 1 training completed.
```

```
def prediction():
    pred1=model1_train.predict(X_test)
    return pred1
print("Calling prediction 1 function")
pred1=prediction()
print(pred1)
```

```
Calling prediction 1 function
[ 3.59693837  4.63842511 -1.71876407 ... -0.10940766  1.68677974
 -1.57306695]
```

```
r2score_LR=(round(r2_score(y_test,pred1)*100,2))
print('r2score:',r2score_LR)
```

```
r2score: 92.97
```

```
rmse = m.sqrt(mean_squared_error(y_test,pred1))
print('RMSE:',rmse)
```

```
RMSE: 0.25403171621389586
```

```
mae=mean_absolute_error(y_test,pred1)
print('MAE:', mae)
```

```
MAE: 0.08127601722273224
```

Ridge Regression

```
# defining--
def modelling3(X_train,y_train,X_test):
    model3=Ridge()
    model3_train=model3.fit(X_train,y_train)
    print("Model 3 training completed.")
    return model3_train

print("Calling the modelling 3 function")
model3_train=modelling3(X_train,y_train,X_test)
```

```
Calling the modelling 3 function
Model 3 training completed.
```

```
def prediction():
    pred3=model3_train.predict(X_test)
    return pred3
print("Calling prediction 3 function")
pred3=prediction()
print(pred3)
```

```
Calling prediction 3 function
[ 3.61628292  4.64125083 -1.70818443 ... -0.10500743  1.69182318
 -1.5691979 ]
```

```
r2score_Ridge=(round(r2_score(y_test,pred3)*100,2))
print('r2score:',r2score_Ridge)
```

```
r2score: 92.97
```

```
rmse = m.sqrt(mean_squared_error(y_test,pred3))
print('RMSE:',rmse)
```

```
RMSE: 0.25402301973841984
```

```
mae=(mean_absolute_error(y_test,pred3))
print('MAE:' ,mae)
```

```
MAE: 0.08116258708885418
```

KNN Regression

```
# defining--
def modelling4(X_train,y_train,X_test):
    model4=KNeighborsRegressor()
    model4_train=model4.fit(X_train,y_train)
    print("Model 4 training completed.")
    return model4_train

print("Calling the modelling 4 function")
model4_train=modelling4(X_train,y_train,X_test)
```

```
Calling the modelling 4 function
Model 4 training completed.
```

```
def prediction():
    pred4=model4_train.predict(X_test)
    return pred4
print("Calling prediction 4 function")
pred4=prediction()
print(pred4)
```

```
Calling prediction 4 function
[ 4.28852914  3.6440438  -0.70111938 ... -0.71632692  4.21825978
 -0.79236466]
```

```
r2score_KNN=(round(r2_score(y_test,pred4)*100,2))
print('r2score:',r2score_KNN)
```

```
r2score: 99.47
```

```
rmse = m.sqrt(mean_squared_error(y_test,pred4))
print('RMSE:',rmse)
```

```
RMSE: 0.06949497741847285
```

```
mae=mean_absolute_error(y_test,pred4)
print('MAE:', mae)
```

```
MAE: 0.012297036170680356
```

Linear Support Vector Regressor

```
# defining--
def modelling5(X_train,y_train,X_test):
    model5=LinearSVR()
    model5_train=model5.fit(X_train,y_train)
    print("Model 5 training completed.")
    return model5_train

print("Calling the modelling 5 function")
model5_train=modelling5(X_train,y_train,X_test)
```

```
Calling the modelling 5 function
Model 5 training completed.
```

```
def prediction():
    pred5=model5_train.predict(X_test)
    return pred5
print("Calling prediction 5 function")
pred5=prediction()
print(pred5)
```

```
Calling prediction 5 function
[ 3.05528224  4.59318256 -2.07292235 ... -0.17719175  1.34546526
 -1.66087761]
```

```
r2score_LSV1=(round(r2_score(y_test,pred5)*100,2))
print('r2score:',r2score_LSV1)
```

```
r2score: 93.31
```

```
rmse = m.sqrt(mean_squared_error(y_test,pred5))
print('RMSE:',rmse)
```

```
RMSE: 0.24797819704061588
```

```
mae=mean_absolute_error(y_test,pred5)
print('MAE:', mae)
```

```
MAE: 0.05944733655460441
```

Random Forest Regressor

```
# defining--
def modelling5(X_train,y_train,X_test):
    model5=RandomForestRegressor()
    model5_train=model5.fit(X_train,y_train)
    print("Model 5 training completed.")
    return model5_train

print("Calling the modelling 5 function")
model5_train=modelling5(X_train,y_train,X_test)
```

```
Calling the modelling 5 function
Model 5 training completed.
```

```
def prediction():
    pred5=model5_train.predict(X_test)
    return pred5
print("Calling prediction 5 function")
pred5=prediction()
print(pred5)
```

```
Calling prediction 5 function
[ 5.64218433  2.44018294 -0.64702771 ... -0.49707605  4.22450012
 -1.11020238]
```

```
r2score_RF=(round(r2_score(y_test,pred5)*100,2))
print('r2score:',r2score_RF)
```

```
r2score: 99.97
```

```
rmse = m.sqrt(mean_squared_error(y_test,pred5))
print('RMSE:',rmse)
```

```
RMSE: 0.0179064013748251
```

```
mae=(mean_absolute_error(y_test,pred5))
print('MAE:' ,mae)
```

```
MAE: 0.0031316900296330845
```

Decision Tree Regressor

```
# defining--
def modelling6(X_train,y_train,X_test):
    model6=DecisionTreeRegressor()
    model6_train=model6.fit(X_train,y_train)
    print("Model 6 training completed.")
    return model6_train

print("Calling the modelling 6 function")
model6_train=modelling6(X_train,y_train,X_test)
```

```
Calling the modelling 6 function
Model 6 training completed.
```

```
def prediction():
    pred6=model6_train.predict(X_test)
    return pred6
print("Calling prediction 6 function")
pred6=prediction()
print(pred6)
```

```
Calling prediction 6 function
[ 5.52401121  2.31993844 -0.73730285 ... -0.4960797   4.24447969
 -1.11749152]
```

```
r2score_DT=(round(r2_score(y_test,pred6)*100,2))
print('r2score:',r2score_DT)
```

```
r2score: 99.94
```

```
rmse = m.sqrt(mean_squared_error(y_test,pred6))
print('RMSE:',rmse)
```

```
RMSE: 0.022936183686087425
```

```
mae=(mean_absolute_error(y_test,pred6))
print('MAE:' ,mae)
```

```
MAE: 0.0037417296476611782
```

Saving Linear Regression Model ¶

```
joblib.dump(model1_train,"Makaan_Linear_Model.pkl")
joblib.dump(model1_train,"Makaan_Linear_Model.joblib")
```

```
['Makaan_Linear_Model.joblib']
```

```
reg= joblib.load("Makaan_Linear_Model.joblib")
```

```
predictions = reg.predict(X_test)
predictions
```

```
array([ 3.60330963,  4.6674118 , -1.70295715, ..., -0.10929108,
        1.68750763, -1.57411957])
```

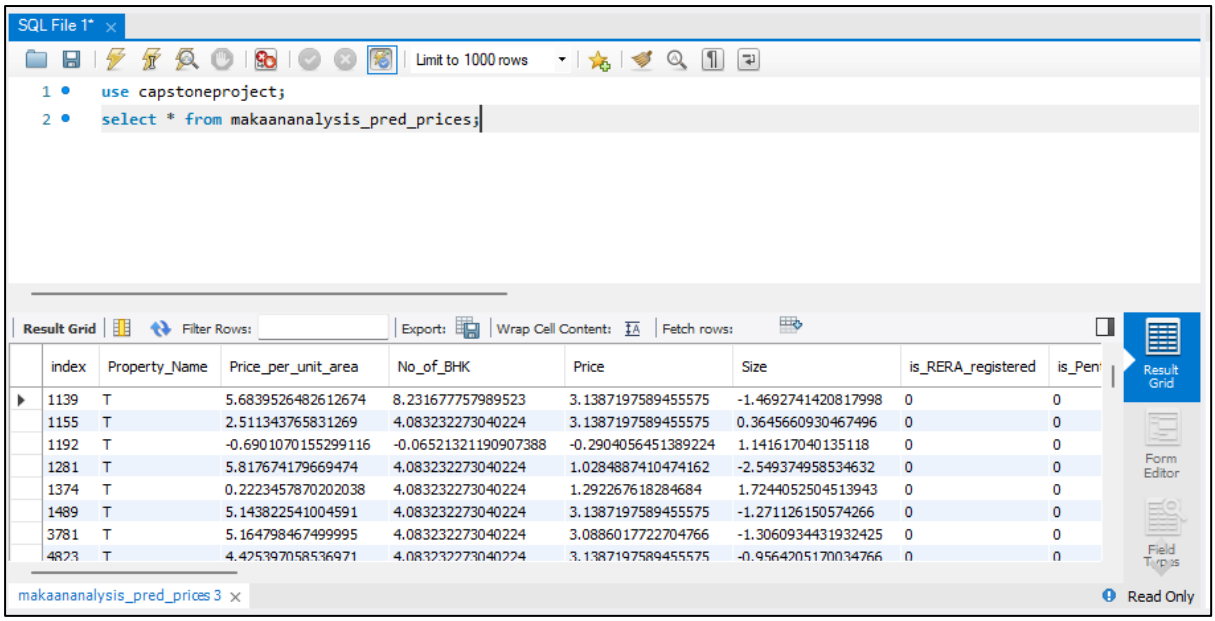
Importing pred prices to MySQL

```
data["Property_Name"].fillna('T',inplace=True)
test=data[data["Property_Name"]=='T']
test["Pred_Price"]=pred1

#accessing database--
engine = create_engine("mysql+pymysql://root:Topmen!orfeb23@localhost/Capstoneproject")
con=engine.connect()

#uploading dataframe to database--
test.to_sql(con=con,name="makaananalysis_pred_prices",if_exists="replace")
```

```
1272770
```

```
r2score=(round(r2_score(test["Price"],test["Pred_Price"])*100,2))
print('r2score:',r2score)
```

```
r2score: 74.78
```



5. Tableau Dashboard