House Prediction Model

Abstract: This mini project is based on the Regression problem which predicts sthe price of house on the basis of certain factors such as location,size,etc.So price is dependent variable which depends on the independent variables (location,size,etc). We first import "Bengaluru_House_Data" dataset then fill the missing values and then perform feature engineering to remove anomalies and then perform the different AI/MI models to test the accuracy of this model.

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv('C:\\Documents\\Bengaluru_House_Data.csv')
df=df.drop(['society'],axis=1)
df.head()
```

Out[2]:

	area_type	availability	location	size	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	1440	2.0	3.0	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	1521	3.0	1.0	95.00
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	1200	2.0	1.0	51.00

In [3]:
```
df.shape
```

Out[3]:
```
(13320, 8)
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   area_type     13320 non-null  object 
 1   availability  13320 non-null  object 
 2   location      13319 non-null  object 
 3   size          13304 non-null  object 
 4   total_sqft    13320 non-null  object 
 5   bath          13247 non-null  float64
 6   balcony       12711 non-null  float64
 7   price         13320 non-null  float64
dtypes: float64(3), object(5)
memory usage: 832.6+ KB
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
area_type         0
availability      0
location          1
size             16
total_sqft        0
bath             73
balcony         609
price             0
dtype: int64
```

In [6]:
```
df['balcony'].fillna(df['balcony'].median(),inplace=True)
```

In [7]:
```
df.dropna(inplace=True,axis=0)
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
area_type       0
availability    0
location        0
size            0
total_sqft      0
bath            0
balcony         0
price           0
dtype: int64
```

In [9]:
```
df.nunique()
df.head()
```

Out[9]:

	area_type	availability	location	size	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	1440	2.0	3.0	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	1521	3.0	1.0	95.00
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	1200	2.0	1.0	51.00

In [10]:
```
df['area_type'].unique()
```

Out[10]:
```
array(['Super built-up  Area', 'Plot  Area', 'Built-up  Area',
       'Carpet  Area'], dtype=object)
```

In [11]:
```
df['bath'].unique()
df=df.drop(['availability'],axis=1)
```

In [12]:
```
df['size'].unique()
```

Out[12]:
```
array(['2 BHK', '4 Bedroom', '3 BHK', '4 BHK', '6 Bedroom', '3 Bedroom',
       '1 BHK', '1 RK', '1 Bedroom', '8 Bedroom', '2 Bedroom',
       '7 Bedroom', '5 BHK', '7 BHK', '6 BHK', '5 Bedroom', '11 BHK',
       '9 BHK', '9 Bedroom', '27 BHK', '10 Bedroom', '11 Bedroom',
       '10 BHK', '19 BHK', '16 BHK', '43 Bedroom', '14 BHK', '8 BHK',
       '12 Bedroom', '13 BHK', '18 Bedroom'], dtype=object)
```

In [13]:
```
df['location'].unique()
```

Out[13]:
```
array(['Electronic City Phase II', 'Chikka Tirupathi', 'Uttarahalli', ...,
       '12th cross srinivas nagar banshankari 3rd stage',
       'Havanur extension', 'Abshot Layout'], dtype=object)
```

In [14]:
```
df['total_sqft'].unique()
```

Out[14]:

	area_type	location	size	total_sqft	bath	balcony	price
0	Super built-up Area	Electronic City Phase II	2 BHK	1056	2.0	1.0	39.07
1	Plot Area	Chikka Tirupathi	4 BHK	2600	5.0	3.0	120.00
2	Built-up Area	Uttarahalli	3 BHK	1440	2.0	3.0	62.00
3	Super built-up Area	Lingadheeranahalli	3 BHK	1521	3.0	1.0	95.00
4	Super built-up Area	Kothanur	2 BHK	1200	2.0	1.0	51.00

Feature Engineering:

In [15]:
```
df['size']=df['size'].str.replace("Bedroom","BHK")
df.head()
```

Out[15]:
```
array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
      dtype=object)
```

In [16]:
```
def is_float(x):
    try:
        float(x)
    except:
        return False
    return True
```

In [17]:
```
df[~df['total_sqft'].apply(is_float)]
```

Out[17]:

	area_type	location	size	total_sqft	bath	balcony	price
30	Super built-up Area	Yelahanka	4 BHK	2100 - 2850	4.0	0.0	186.000
122	Super built-up Area	Hebbal	4 BHK	3067 - 8156	4.0	0.0	477.000
137	Super built-up Area	8th Phase JP Nagar	2 BHK	1042 - 1105	2.0	0.0	54.005
165	Super built-up Area	Sarjapur	2 BHK	1145 - 1340	2.0	0.0	43.490
188	Super built-up Area	KR Puram	2 BHK	1015 - 1540	2.0	0.0	56.800
...	...	...	...	...	...	...	...
12975	Super built-up Area	Whitefield	2 BHK	850 - 1060	2.0	0.0	38.190
12990	Super built-up Area	Talaghattapura	3 BHK	1804 - 2273	3.0	0.0	122.000
13059	Super built-up Area	Harlur	2 BHK	1200 - 1470	2.0	0.0	72.760
13265	Super built-up Area	Hoodi	2 BHK	1133 - 1384	2.0	0.0	59.135
13299	Super built-up Area	Whitefield	4 BHK	2830 - 2882	5.0	0.0	154.500

190 rows × 7 columns

In [18]:
```
df['total_sqft'].describe()
```

Out[18]:
```
count     13246
unique     2067
top        1200
freq        843
Name: total_sqft, dtype: object
```

In [19]:
```
def convert_sqft_to_num(x):
    tokens = x.split('-')
    if len(tokens) == 2:
        return (float(tokens[0])+float(tokens[1]))/2
    try:
        return float(x)
    except:
        return None
#df4 = df3.copy()
df.total_sqft = df.total_sqft.apply(convert_sqft_to_num)
df = df[df.total_sqft.notnull()]
df.head(2)
```

Out[19]:

	area_type	location	size	total_sqft	bath	balcony	price
0	Super built-up Area	Electronic City Phase II	2 BHK	1056.0	2.0	1.0	39.07
1	Plot Area	Chikka Tirupathi	4 BHK	2600.0	5.0	3.0	120.00

In [20]:
```
df['Price_per_sqr_feet']=df['price']*100000/df['total_sqft']
```

In [21]:
```
df.head()
```

Out[21]:

	area_type	location	size	total_sqft	bath	balcony	price	Price_per_sqr_feet
0	Super built-up Area	Electronic City Phase II	2 BHK	1056.0	2.0	1.0	39.07	3699.810606
1	Plot Area	Chikka Tirupathi	4 BHK	2600.0	5.0	3.0	120.00	4615.384615
2	Built-up Area	Uttarahalli	3 BHK	1440.0	2.0	3.0	62.00	4305.555556
3	Super built-up Area	Lingadheeranahalli	3 BHK	1521.0	3.0	1.0	95.00	6245.890861
4	Super built-up Area	Kothanur	2 BHK	1200.0	2.0	1.0	51.00	4250.000000

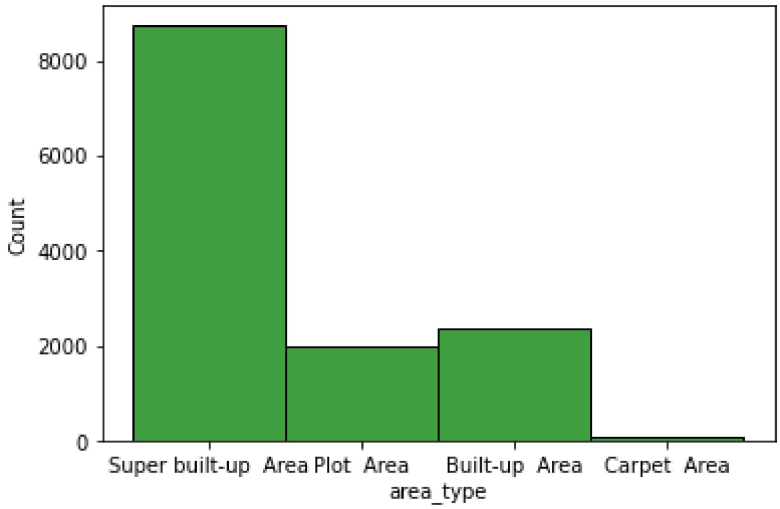
In [22]:
```
df = df.reset_index(drop=True)
```

In [23]:
```
df['area_type'].unique()
```

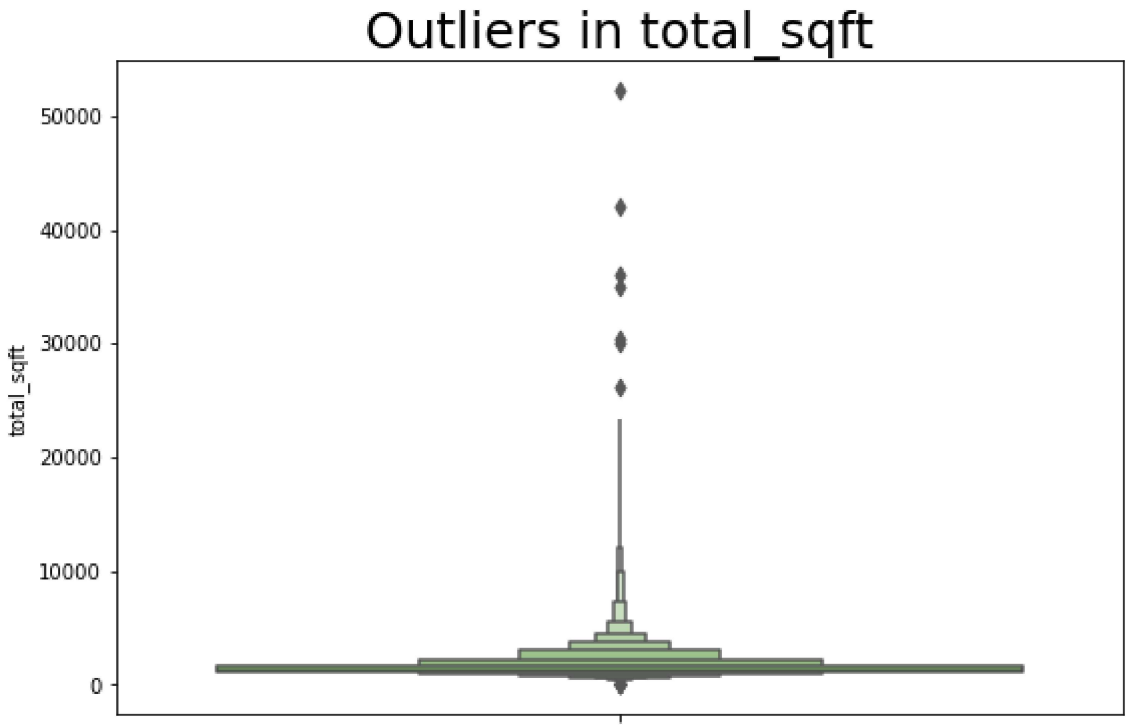
Out[23]:
```
array(['Super built-up  Area', 'Plot  Area', 'Built-up  Area',
       'Carpet  Area'], dtype=object)
```

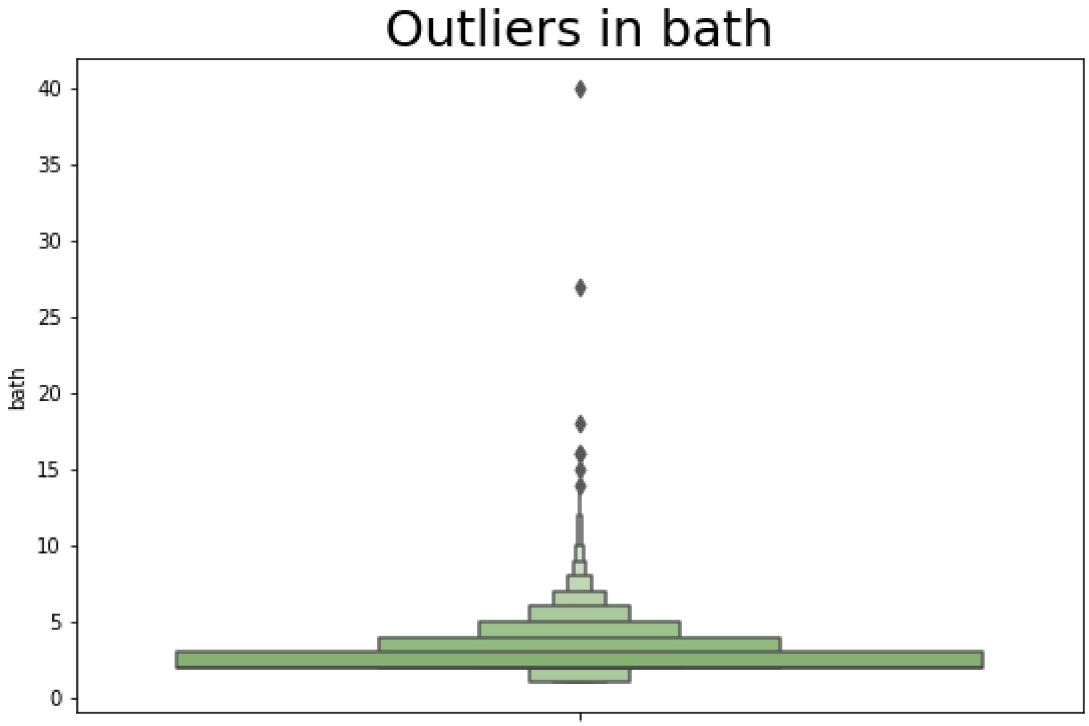
In [24]:
```
sns.histplot(df.area_type,color='green')
```

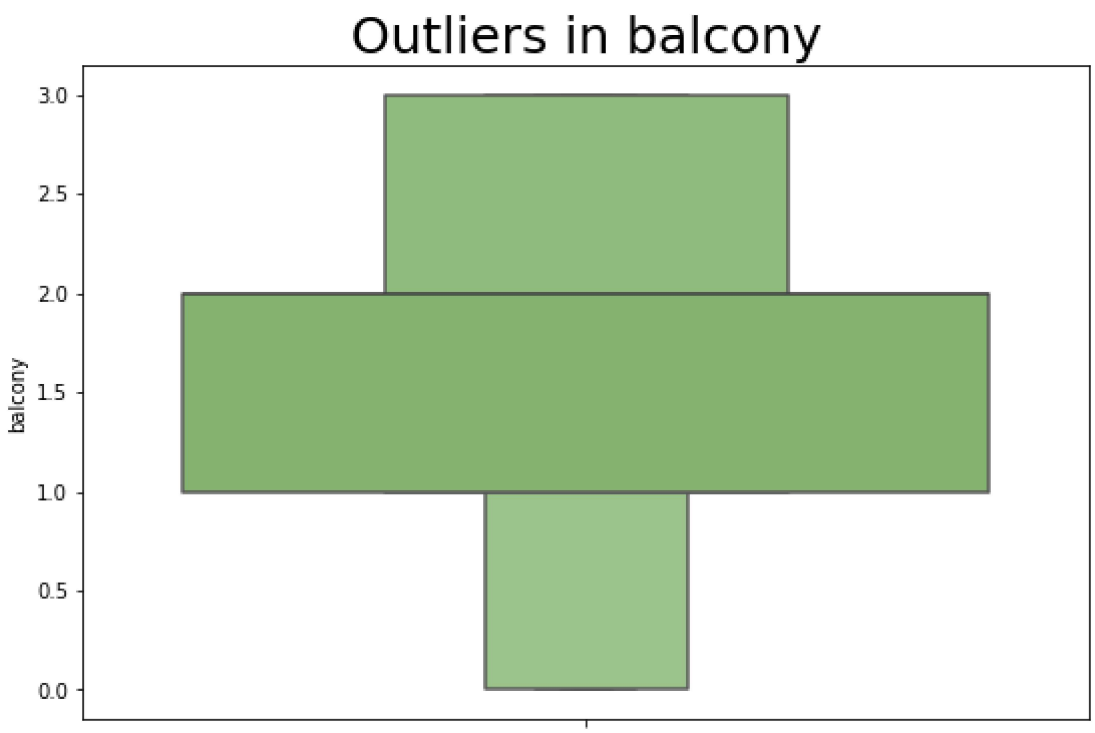
Out[24]:
```
<AxesSubplot:xlabel='area_type', ylabel='Count'>
```

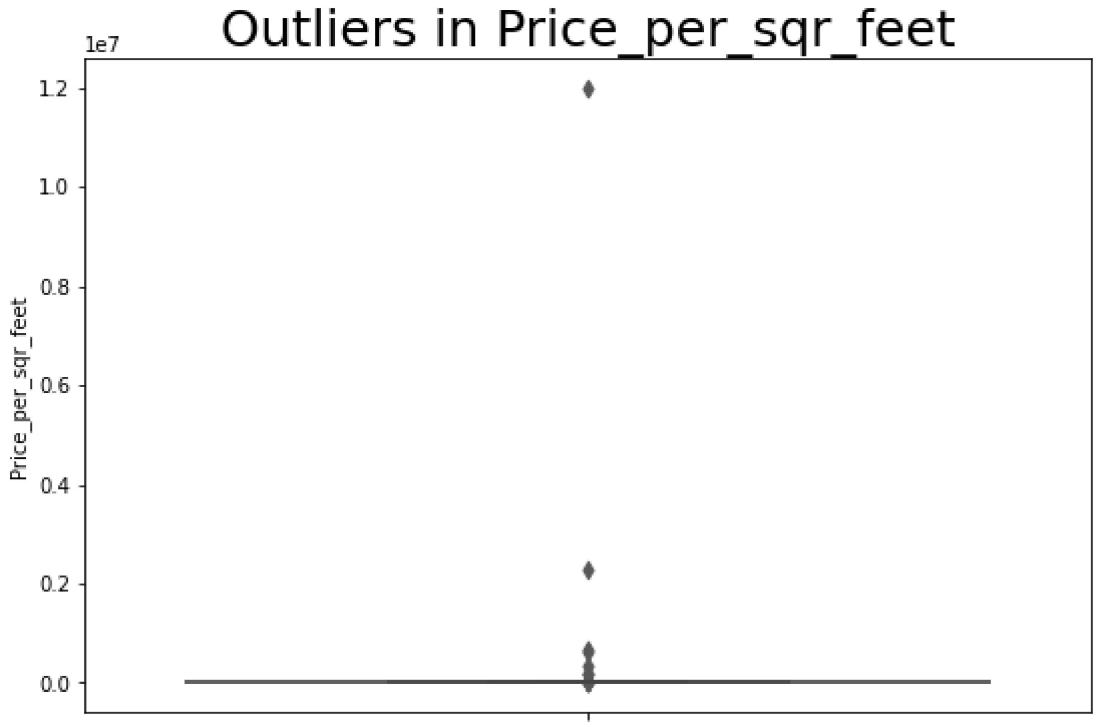


In [25]:
```
for col in ["total_sqft", "bath", "balcony", "Price_per_sqr_feet"]:
    plt.figure(figsize=(9,6));
    sns.boxenplot(y=col,data=df,palette='summer');
    plt.title(f'Outliers in {col}',fontsize=25,fontweight=5);
    plt.show()
```









Removl of Anomalies:

In [26]:
```
df = df[(df["total_sqft"] >= 100) | (df["total_sqft"] <= 5000)]
df=df[(df['bath']<=10)]
df.head()
```

Out[26]:

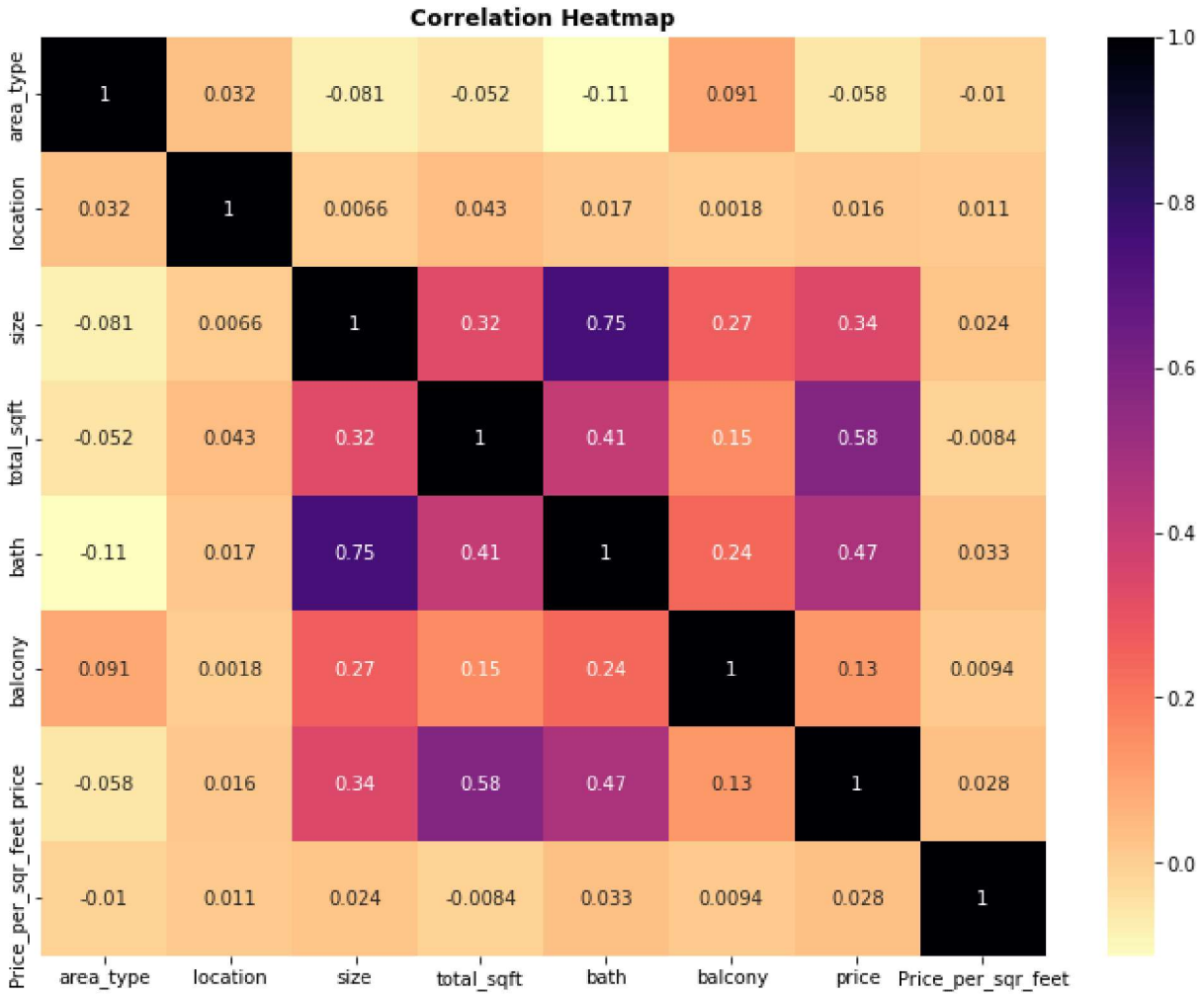
	area_type	location	size	total_sqft	bath	balcony	price	Price_per_sqr_feet
0	Super built-up Area	Electronic City Phase II	2 BHK	1056.0	2.0	1.0	39.07	3699.810606
1	Plot Area	Chikka Tirupathi	4 BHK	2600.0	5.0	3.0	120.00	4615.384615
2	Built-up Area	Uttarahalli	3 BHK	1440.0	2.0	3.0	62.00	4305.555556
3	Super built-up Area	Lingadheeranahalli	3 BHK	1521.0	3.0	1.0	95.00	6245.890861
4	Super built-up Area	Kothanur	2 BHK	1200.0	2.0	1.0	51.00	4250.000000

In [27]:
```
print(np.where((df["Price_per_sqr_feet"]<1000) | (df['Price_per_sqr_feet']>50000)
```

```
(array([  345,   665,   798,  1004,  1104,  1863,  4035,  4913,  5332,
        5900,  6345,  7000,  7154,  7563,  7787,  7850,  8293,  9129,
        9421, 11429, 11616, 12309, 12336], dtype=int64),)
```

In [28]:
```
df = df[(df["Price_per_sqr_feet"]>=1000) | (df['Price_per_sqr_feet']<=50000)]
```

In [29]:
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df['area_type'] = encoder.fit_transform(df['area_type'])
df['location'] = encoder.fit_transform(df['location'])
df['size'] = encoder.fit_transform(df['size'])
plt.figure(figsize = (12,9))
sns.heatmap(df.corr(), annot = True, cmap = "magma_r")
plt.title("Correlation Heatmap",fontdict = {"fontweight":"bold"})
plt.show()
```



In [31]:
```
x = df.drop(["price", "area_type"], axis = 1)
y = df["price"]
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2)
```

Applying Different Models and Check Their Accuracy:

In [32]:
```
#Applying Linear Model Regression
from sklearn import linear_model
model = linear_model.LinearRegression()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)
from sklearn.metrics import r2_score
r2_score = model.score(x_test,y_test)
print(r2_score*100,'%')
```

```
48.20449597472085 %
```

In [34]:
```
df.iloc[5695]
```

Out[34]:
```
area_type              3.000000
location             185.000000
size                   5.000000
total_sqft          1290.000000
bath                   2.000000
balcony                2.000000
price                 80.000000
Price_per_sqr_feet  6201.550388
Name: 5706, dtype: float64
```

In [35]:
```
from sklearn.ensemble import RandomForestRegressor
RFG=RandomForestRegressor(n_estimators=11)
RFG.fit(x_train, y_train)
y_pred = RFG.predict(x_test)
r2_score = RFG.score(x_test,y_test)
print(r2_score*100,'%')


#predict_price=model.predict([[5,1233.0,2.0,3.0,8110.6201.550388]])
#predict_price=RFG.predict([[5,1290.000000,2.0,2.0,6201.550388]])
#print(predict_price)
```

```
98.8245480824521 %
```

In [37]:
```
#Applying decision Tree Regressor
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
r2_score = model.score(x_test,y_test)
print(r2_score*100,'%')
```

```
94.45696947172763 %
```

In [38]:
```
#Applying KNeighors Regressor
from sklearn.neighbors import KNeighborsRegressor
model = KNeighborsRegressor(n_neighbors=10)
model.fit(x_train, y_train)
KNeighborsRegressor(n_neighbors=10)
y_pred = model.predict(x_test)
r2_score = model.score(x_test,y_test)
print(r2_score*100,'%')
```

```
95.51008782900668 %
```

Exporting The Model To Pickle File

In [ ]:
```
# Export the tested model to a pickle file
import pickle
with open('house_prices_predict_model.pickle','wb') as f:
     pickle.dump(RFG,f)
```