Import Library

In [54]:

```
#====== Pandas ======
import pandas as pd
pd.set_option("display.max_columns",None)

#===== Numpy =====
import numpy as np

#===== Visualisation =====
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set("notebook")

#====== Stats and Transformation =====
from scipy.stats import (ttest_ind,
                        f_oneway,
                        chi2_contingency,
                        yeojohnson,
                        boxcox,
                        spearmanr)
from sklearn.preprocessing import PowerTransformer
from scipy.stats import chi2
#===== Model =====
from sklearn.model_selection import train_test_split,RandomizedSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRFRegressor

#==== Evaluation ====
from sklearn.metrics import (r2_score,
                      mean_absolute_error,
                      median_absolute_error)
from yellowbrick.regressor import PredictionError, ResidualsPlot
#==== Encoder Geo ====
from geopy.geocoders import Nominatim
geolocator = Nominatim(user_agent="my_user_agent")

#==== Function ====
def missing_check(df):
    missing = df.isnull().sum()
    percent = 100*(missing/len(df))
    number_unique = df.nunique()
    data_type = df.dtypes
    return pd.DataFrame({"Missing":missing,
                         "Percent_Missing":percent,
                         "Number_Unique":number_unique,
                         "Data_Types":data_type}).sort_values("Percent_Missing",ascending=


import warnings
warnings.filterwarnings("ignore")
```

Read Data Set

In [55]:

```
df = pd.read_csv("House_Rent_Dataset.csv",low_memory=False)
```

Data Investigation

In [56]:

```
#read first 5 of row
df.head()
```

Out[56]:

	Posted On	BHK	Rent	Size	Floor	Area Type	Area Locality	City	Furnishing Status	Tenar Preferre
0	2022-05-18	2	10000	1100	Ground out of 2	Super Area	Bandel	Kolkata	Unfurnished	Bachelors/Famil
1	2022-05-13	2	20000	800	1 out of 3	Super Area	Phool Bagan, Kankurgachi	Kolkata	Semi-Furnished	Bachelors/Famil
2	2022-05-16	2	17000	1000	1 out of 3	Super Area	Salt Lake City Sector 2	Kolkata	Semi-Furnished	Bachelors/Famil
3	2022-07-04	2	10000	800	1 out of 2	Super Area	Dumdum Park	Kolkata	Unfurnished	Bachelors/Famil
4	2022-05-09	2	7500	850	1 out of 2	Carpet Area	South Dum Dum	Kolkata	Unfurnished	Bachelor

In [57]:

```
#check missing values
missing_check(df)
```

Out[57]:

	Missing	Percent_Missing	Number_Unique	Data_Types
Posted On	0	0.0	81	object
BHK	0	0.0	6	int64
Rent	0	0.0	243	int64
Size	0	0.0	615	int64
Floor	0	0.0	480	object
Area Type	0	0.0	3	object
Area Locality	0	0.0	2235	object
City	0	0.0	6	object
Furnishing Status	0	0.0	3	object
Tenant Preferred	0	0.0	3	object
Bathroom	0	0.0	8	int64
Point of Contact	0	0.0	3	object

every columns do not have missing values, but we can see that **Posted On** have object as data types, we need to convert to timestamp of data type

In [58]:

```
df["Posted On"] = pd.to_datetime(df['Posted On'],infer_datetime_format=False)
```

In [59]:

```
#sorting data based on Posted On
df = df.sort_values("Posted On",ascending=False)
```

In [60]:

```
#check duplicated
df.duplicated().sum()
```

Out[60]:

```
0
```

the data not have duplicated

In [61]:

```
#check every unique values
for x in df.columns:
    print(f"========= {x} =========")
    print(f"{df[x].unique()}")
    print()
```

```
========= Posted On =========
['2022-07-11T00:00:00.000000000' '2022-07-10T00:00:00.000000000'
 '2022-07-09T00:00:00.000000000' '2022-07-08T00:00:00.000000000'
 '2022-07-07T00:00:00.000000000' '2022-07-06T00:00:00.000000000'
 '2022-07-05T00:00:00.000000000' '2022-07-04T00:00:00.000000000'
 '2022-07-03T00:00:00.000000000' '2022-07-02T00:00:00.000000000'
 '2022-07-01T00:00:00.000000000' '2022-06-30T00:00:00.000000000'
 '2022-06-29T00:00:00.000000000' '2022-06-28T00:00:00.000000000'
 '2022-06-27T00:00:00.000000000' '2022-06-26T00:00:00.000000000'
 '2022-06-25T00:00:00.000000000' '2022-06-24T00:00:00.000000000'
 '2022-06-23T00:00:00.000000000' '2022-06-22T00:00:00.000000000'
 '2022-06-21T00:00:00.000000000' '2022-06-20T00:00:00.000000000'
 '2022-06-19T00:00:00.000000000' '2022-06-18T00:00:00.000000000'
 '2022-06-17T00:00:00.000000000' '2022-06-16T00:00:00.000000000'
 '2022-06-15T00:00:00.000000000' '2022-06-14T00:00:00.000000000'
 '2022-06-13T00:00:00.000000000' '2022-06-12T00:00:00.000000000'
 '2022-06-11T00:00:00.000000000' '2022-06-10T00:00:00.000000000'
 '2022-06-09T00:00:00.000000000' '2022-06-08T00:00:00.000000000'
 '2022-06-07T00:00:00.000000000' '2022-06-06T00:00:00.000000000'
```

In [62]:

```
#feature engineering from column Floor
df["Floor"] = df["Floor"].apply(lambda x : x.replace("out ",""))
df["Floors"] = df["Floor"].apply(lambda x : x.split("of")[0])
df["Total_Number_of_Floors"] = df["Floor"].apply(lambda x : x.split("of")[-1])

#drop columns
df = df.drop("Floor",axis=1)

df.head()
```

Out[62]:

	Posted On	BHK	Rent	Size	Area Type	Area Locality	City	Furnishing Status	Tena Preferr
3552	2022-07-11	2	12000	550	Super Area	Choolaimedu	Chennai	Unfurnished	Bachelors/Fam
4341	2022-07-10	2	21000	1100	Carpet Area	Himayath Nagar, NH 7	Hyderabad	Semi-Furnished	Bachelors/Fam
3743	2022-07-10	3	15000	1200	Carpet Area	Madambakkam	Chennai	Unfurnished	Bachelors/Fam
3385	2022-07-10	3	38000	1300	Carpet Area	Chromepet, GST Road	Chennai	Unfurnished	Bachelors/Fam
3520	2022-07-10	3	65000	1444	Super Area	Nungambakkam	Chennai	Semi-Furnished	Bachelo

In [63]:

```
df["Area Locality"] = df["Area Locality"].apply(lambda x:x.strip().split(",")[0] if len(x)>

df["Address"] = df["City"] + "," + df["Area Locality"]
df.head()
```

Out[63]:

	Posted On	BHK	Rent	Size	Area Type	Area Locality	City	Furnishing Status	Tena Preferr
3552	2022-07-11	2	12000	550	Super Area	Choolaimedu	Chennai	Unfurnished	Bachelors/Fam
4341	2022-07-10	2	21000	1100	Carpet Area	Himayath Nagar	Hyderabad	Semi-Furnished	Bachelors/Fam
3743	2022-07-10	3	15000	1200	Carpet Area	Madambakkam	Chennai	Unfurnished	Bachelors/Fam
3385	2022-07-10	3	38000	1300	Carpet Area	Chromepet	Chennai	Unfurnished	Bachelors/Fam
3520	2022-07-10	3	65000	1444	Super Area	Nungambakkam	Chennai	Semi-Furnished	Bachelo

Extract Longitude and Latitude

In [64]:

```
# extract latitude and longitude
# df_geo = pd.DataFrame({'Address':[],
#                         'latitude':[],
#                         'longitude':[]})

# for x in df['Address'].unique():
#     try:
#         city =x
#         country ="India"
#         loc = geolocator.geocode(city+','+ country)
#         print(f"Succes addres for {x}")
#         df_geo = df_geo.append({'Address':x,
#                         'latitude':loc.latitude,
#                         'longitude':loc.longitude},ignore_index=True)
#     except:
#         print(f'Not success for address {x}')
#         print("Take City Only")
#         city =x.split(",")[0]
#         country ="India"
#         loc = geolocator.geocode(city+','+ country)
#         df_geo = df_geo.append({'Address':x,
#                         'latitude':loc.latitude,
#                         'longitude':loc.longitude},ignore_index=True)
```

In [65]:

```
df_geo = pd.read_csv("data_lat_lon.csv")
df_geo.head()
```

Out[65]:

	Unnamed: 0	Address	latitude	longitude
0	0	Chennai,Choolaimedu	13.062334	80.225401
1	1	Hyderabad,Himayath Nagar	17.399564	78.484392
2	2	Chennai,Madambakkam	13.083694	80.270186
3	3	Chennai,Chromepet	12.946277	80.137037
4	4	Chennai,Nungambakkam	13.052811	80.249847

In [66]:

```
df_geo = df_geo.drop('Unnamed: 0',axis=1)
df_geo.head()
```

Out[66]:

	Address	latitude	longitude
0	Chennai,Choolaimedu	13.062334	80.225401
1	Hyderabad,Himayath Nagar	17.399564	78.484392
2	Chennai,Madambakkam	13.083694	80.270186
3	Chennai,Chromepet	12.946277	80.137037
4	Chennai,Nungambakkam	13.052811	80.249847

In [67]:

```
df["Address"] = df["Address"].replace(['Mumbai,Midas Enclave CHS ',
                                       'Mumbai,Jumbo Darshan ',
                                       'Mumbai,holy complex ',
                                       'Mumbai,Lodha Azzuro ',
                                       'Kolkata,Baishnavghata patuli ',
                                       'Mumbai,Lodha NCP Commercial ',
                                       'Mumbai,Bindra Complex '],
                                      ['Mumbai,Midas Enclave CHS',
                                       'Mumbai,Jumbo Darshan',
                                       'Mumbai,holy complex',
                                       'Mumbai,Lodha Azzuro',
                                       'Kolkata,Baishnavghata patuli',
                                       'Mumbai,Lodha NCP Commercial',
                                       'Mumbai,Bindra Complex'])
```

In [68]:

```
df = pd.merge(df,df_geo[["latitude","longitude","Address"]],on="Address",how="left")
df.head()
```

Out[68]:

	Posted On	BHK	Rent	Size	Area Type	Area Locality	City	Furnishing Status	Tenant Preferred
0	2022-07-11	2	12000	550	Super Area	Choolaimedu	Chennai	Unfurnished	Bachelors/Family
1	2022-07-10	2	21000	1100	Carpet Area	Himayath Nagar	Hyderabad	Semi-Furnished	Bachelors/Family
2	2022-07-10	3	15000	1200	Carpet Area	Madambakkam	Chennai	Unfurnished	Bachelors/Family
3	2022-07-10	3	38000	1300	Carpet Area	Chromepet	Chennai	Unfurnished	Bachelors/Family
4	2022-07-10	3	65000	1444	Super Area	Nungambakkam	Chennai	Semi-Furnished	Bachelors

In [69]:

```
missing_check(df)
```

Out[69]:

	Missing	Percent_Missing	Number_Unique	Data_Types
Posted On	0	0.0	81	datetime64[ns]
BHK	0	0.0	6	int64
Rent	0	0.0	243	int64
Size	0	0.0	615	int64
Area Type	0	0.0	3	object
Area Locality	0	0.0	2174	object
City	0	0.0	6	object
Furnishing Status	0	0.0	3	object
Tenant Preferred	0	0.0	3	object
Bathroom	0	0.0	8	int64
Point of Contact	0	0.0	3	object
Floors	0	0.0	57	object
Total_Number_of_Floors	0	0.0	69	object
Address	0	0.0	2186	object
latitude	0	0.0	1071	float64
longitude	0	0.0	1071	float64

Exploratory Data Analysis

In [70]:

```
# splitting data
X = df.drop("Rent",axis=1).copy()
y = df["Rent"].copy()

X_train,X_test,y_train,y_test = train_test_split(X,y,
                                                 test_size=0.3,random_state=42)
```

Continuous

In [71]:

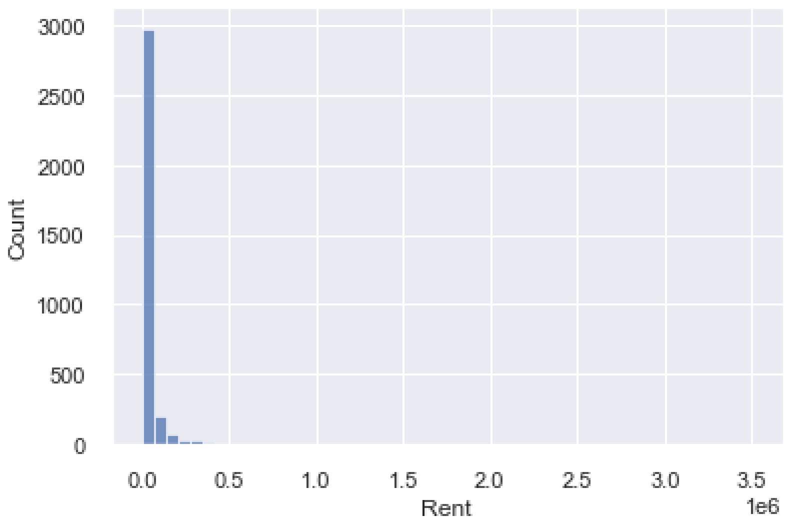
```
X_train["Rent"] = y_train
```

In [72]:

```
sns.histplot(X_train["Rent"],bins=50)
```

Out[72]:

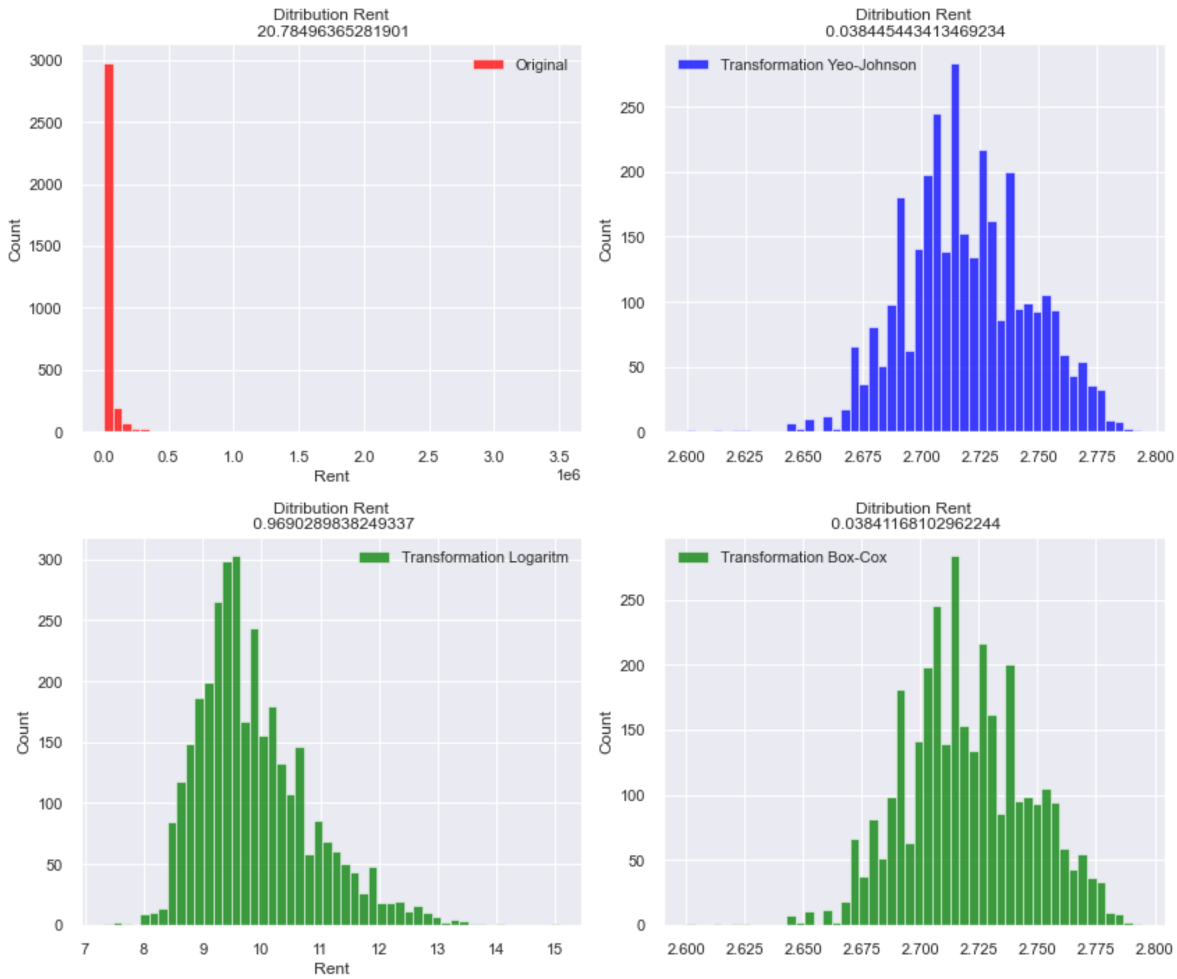
```
<AxesSubplot:xlabel='Rent', ylabel='Count'>
```



As we can see that **Rent** highly skewed and there is have outliers, in here i will filter data and do some transformation

In [73]:

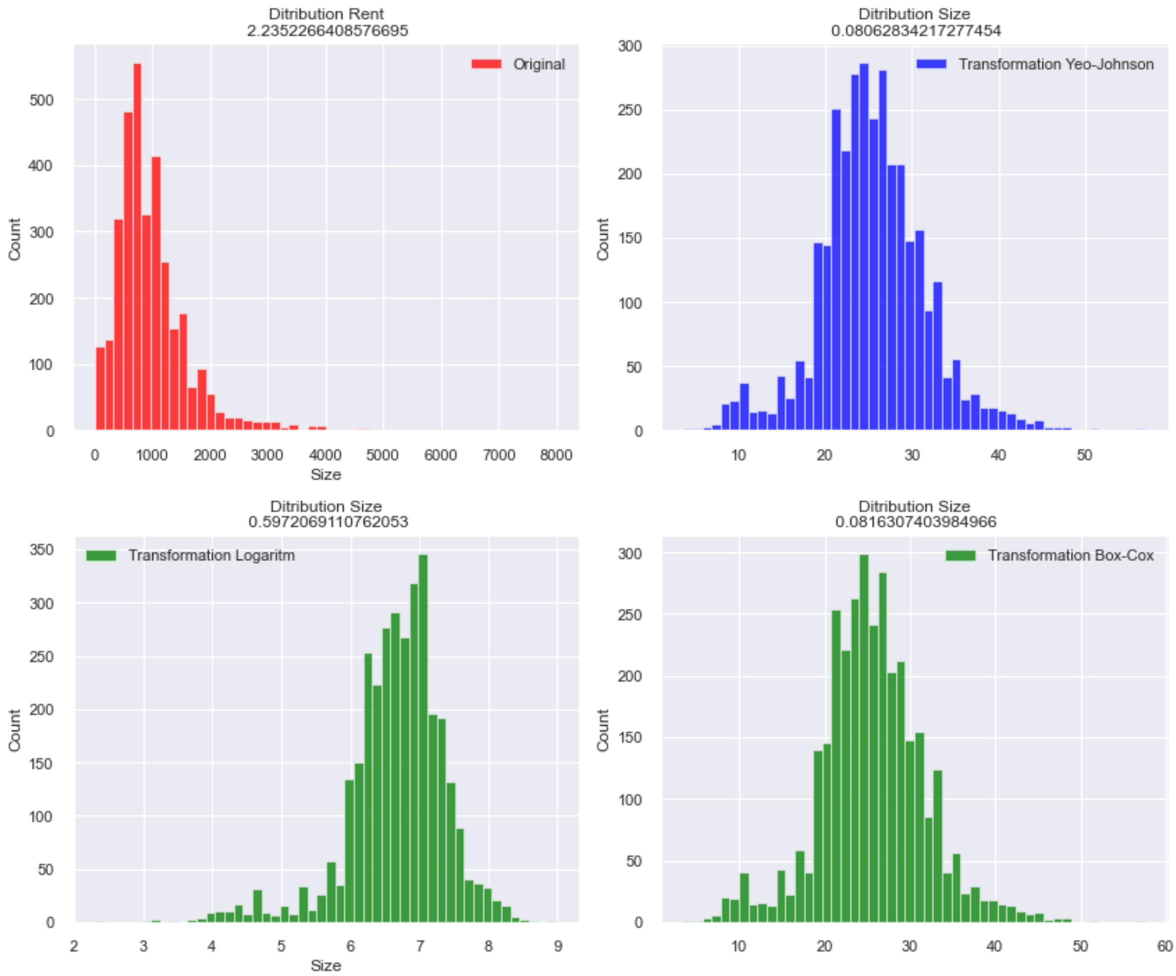
```
# transformation
plt.figure(figsize=(12,10))

plt.subplot(221)
sns.histplot(X_train['Rent'],bins=50,label='Original',color='red')
plt.title(f'Ditribution Rent\n {X_train["Rent"].skew()}')
plt.legend(loc='best')

plt.subplot(222)
sns.histplot(yeojohnson(X_train['Rent'])[0],bins=50,label='Transformation Yeo-Johnson',colo
plt.title(f'Ditribution Rent\n {pd.Series(yeojohnson(X_train["Rent"])[0]).skew()}')
plt.legend(loc='best')

plt.subplot(223)
sns.histplot(np.log(X_train['Rent']),bins=50,label='Transformation Logaritm',color='green')
plt.title(f'Ditribution Rent\n {np.log(X_train["Rent"]).skew()}')
plt.legend(loc='best')

plt.subplot(224)
sns.histplot(boxcox(X_train['Rent'])[0],bins=50,label='Transformation Box-Cox',color='green
plt.title(f'Ditribution Rent\n {pd.Series(boxcox(X_train["Rent"])[0]).skew()}')
plt.legend(loc='best')

plt.tight_layout()
plt.show()
```



as we can see that transformation yeo-johnson and boxcox can effectively make Rent's distribution to be normal distribution. i will choose transformation yeojohnson

In [74]:

```
# transformation
plt.figure(figsize=(12,10))

plt.subplot(221)
sns.histplot(X_train['Size'],bins=50,label='Original',color='red')
plt.title(f'Ditribution Rent\n {X_train["Size"].skew()}')
plt.legend(loc='best')

plt.subplot(222)
sns.histplot(yeojohnson(X_train['Size'])[0],bins=50,label='Transformation Yeo-Johnson',colo
plt.title(f'Ditribution Size\n {pd.Series(yeojohnson(X_train["Size"])[0]).skew()}')
plt.legend(loc='best')

plt.subplot(223)
sns.histplot(np.log(X_train['Size']),bins=50,label='Transformation Logaritm',color='green')
plt.title(f'Ditribution Size\n {np.sqrt(X_train["Size"]).skew()}')
plt.legend(loc='best')

plt.subplot(224)
sns.histplot(boxcox(X_train['Size'])[0],bins=50,label='Transformation Box-Cox',color='green
plt.title(f'Ditribution Size\n {pd.Series(boxcox(X_train["Size"])[0]).skew()}')
plt.legend(loc='best')

plt.tight_layout()
plt.show()
```



In [75]:

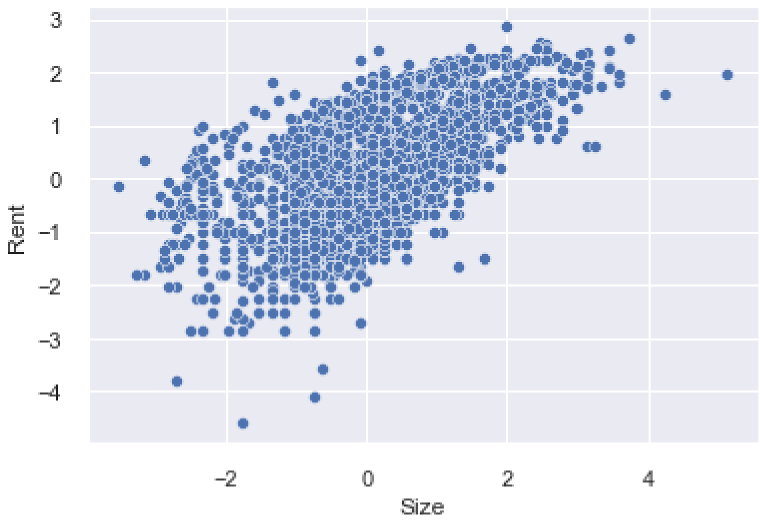
```
pt  = PowerTransformer()
X_train[["Rent","Size"]] = pt.fit_transform(X_train[["Rent",'Size']])
```

In [76]:

```
sns.scatterplot(data=X_train,x="Size",y="Rent")
```

Out[76]:

```
<AxesSubplot:xlabel='Size', ylabel='Rent'>
```



Size and rent have relation non-linear, this is indicate size can be predictor, next we need to check correlation between size and rent using spearmant correlation

Outliers Detection

In [77]:

```
from sklearn.cluster import DBSCAN

outlier_detection = DBSCAN()
clusters = outlier_detection.fit_predict(X_train[["Size","Rent"]])
list(clusters).count(-1)
```

Out[77]:

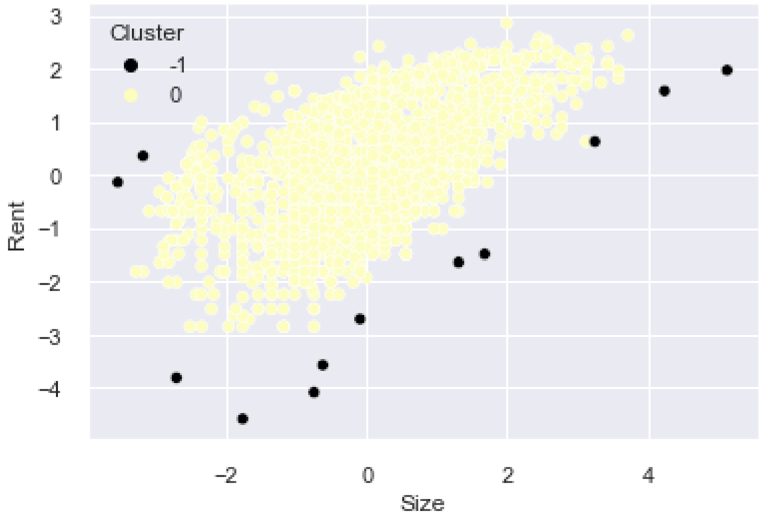
```
12
```

In [78]:

```
X_train["Cluster"] = clusters

sns.scatterplot(data=X_train,x="Size",y="Rent",hue="Cluster",palette="magma")
```

Out[78]:

```
<AxesSubplot:xlabel='Size', ylabel='Rent'>
```



In [79]:

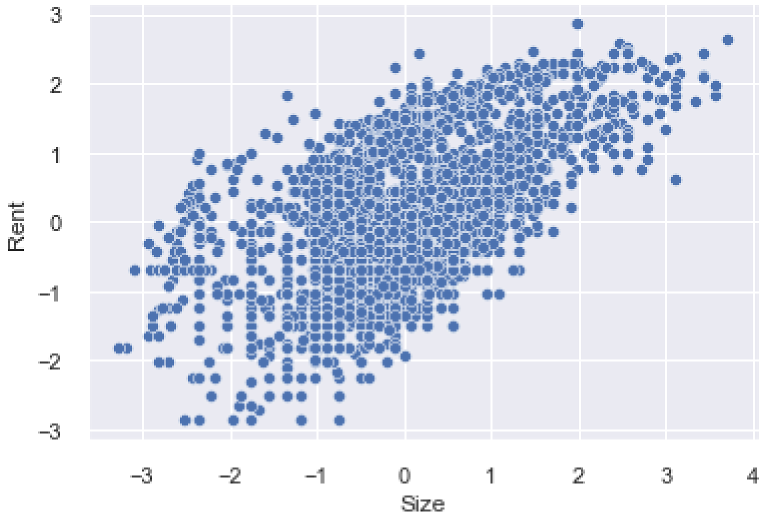
```
X_train = X_train[X_train["Cluster"]!=-1]

X_train = X_train.drop("Cluster",axis=1)
```

In [80]:

```
sns.scatterplot(data=X_train,x="Size",y="Rent")
```

Out[80]:

```
<AxesSubplot:xlabel='Size', ylabel='Rent'>
```



In [81]:

```
coef_s,p = spearmanr(X_train['Size'],X_train["Rent"])

print("Spearman Correlation Size and Rent ",coef_s)
print('P-value ',p)
```

```
Spearman Correlation Size and Rent  0.53774303974742 98
P-value  1.7148440187851328e-247
```

based on spearman correlation test, correlation between size and rent is 0.5 (high) and this is significant because p-value less than 0.05

Discrete

In [82]:

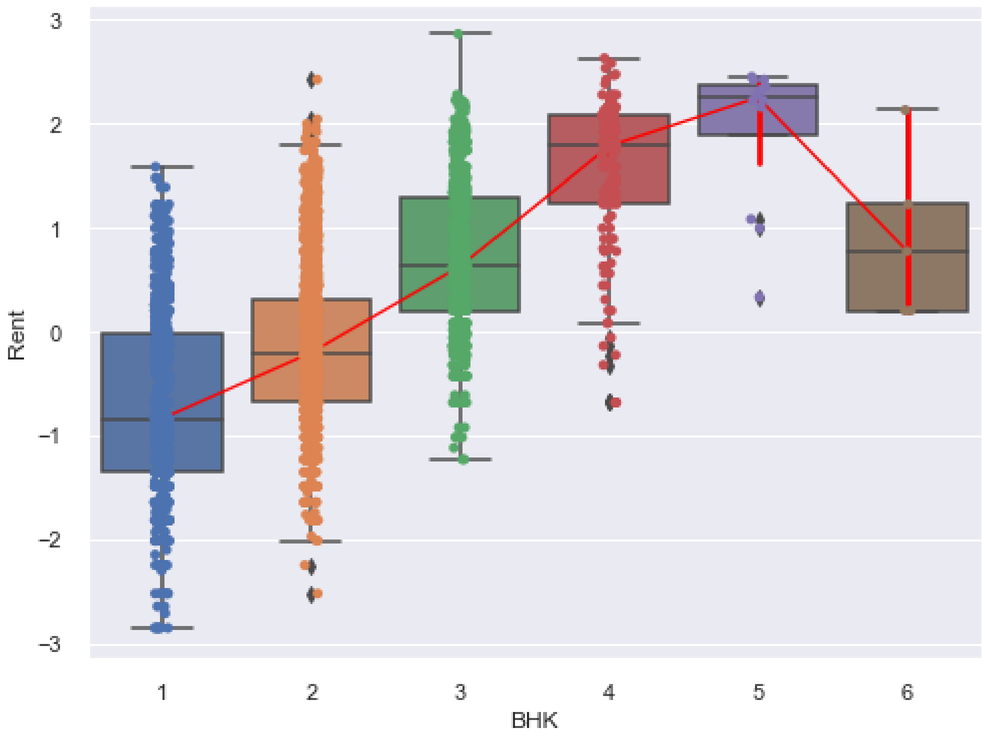
```
dis_var = [var for var in X_train.columns
           if X_train[var].nunique() < 25 and var !='Rent'
           and X_train[var].dtypes != 'O']

len(dis_var)
```

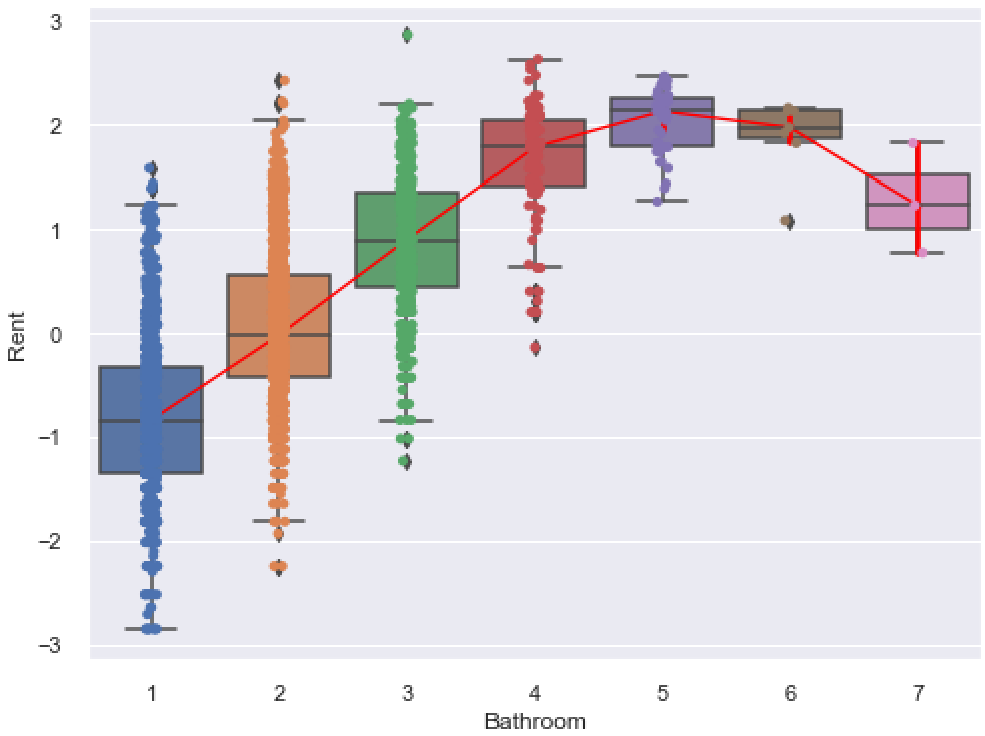
Out[82]:

2

In [83]:

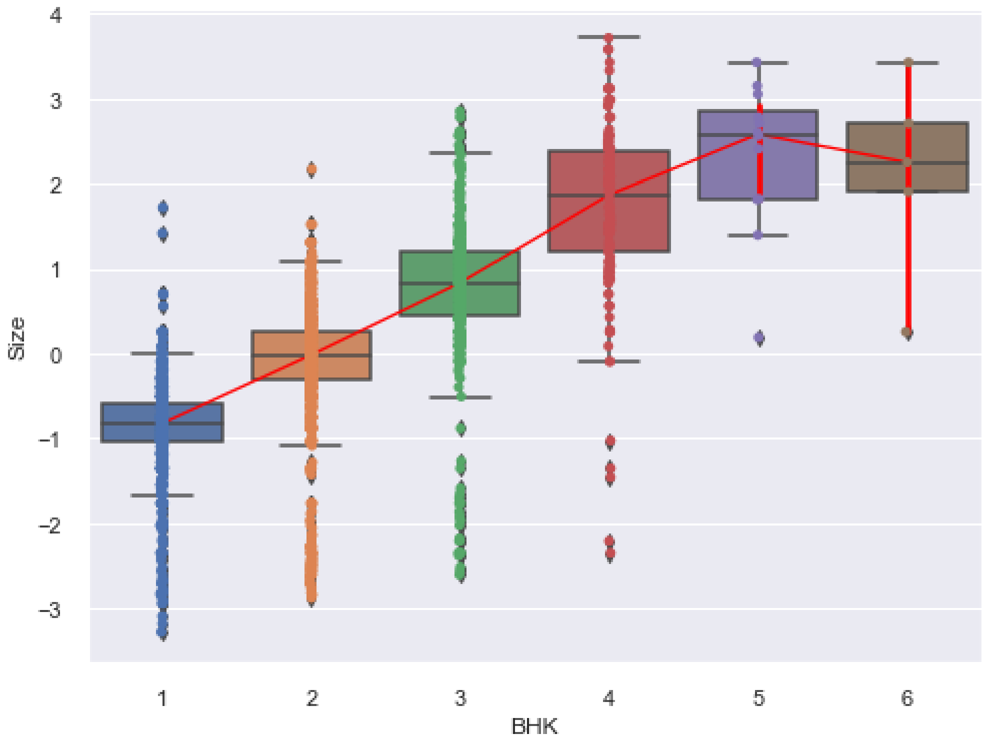
```
for x in dis_var:
    plt.figure(figsize=(8,6))
    sns.boxplot(data=X_train,x=x,y="Rent")
    sns.pointplot(data=X_train,x=x,y="Rent",color="red",estimator=np.median,
    errorbar=('ci',95),scale=0.5)
    sns.stripplot(data=X_train,x=x,y="Rent",jitter=0.05)
    plt.show()
```

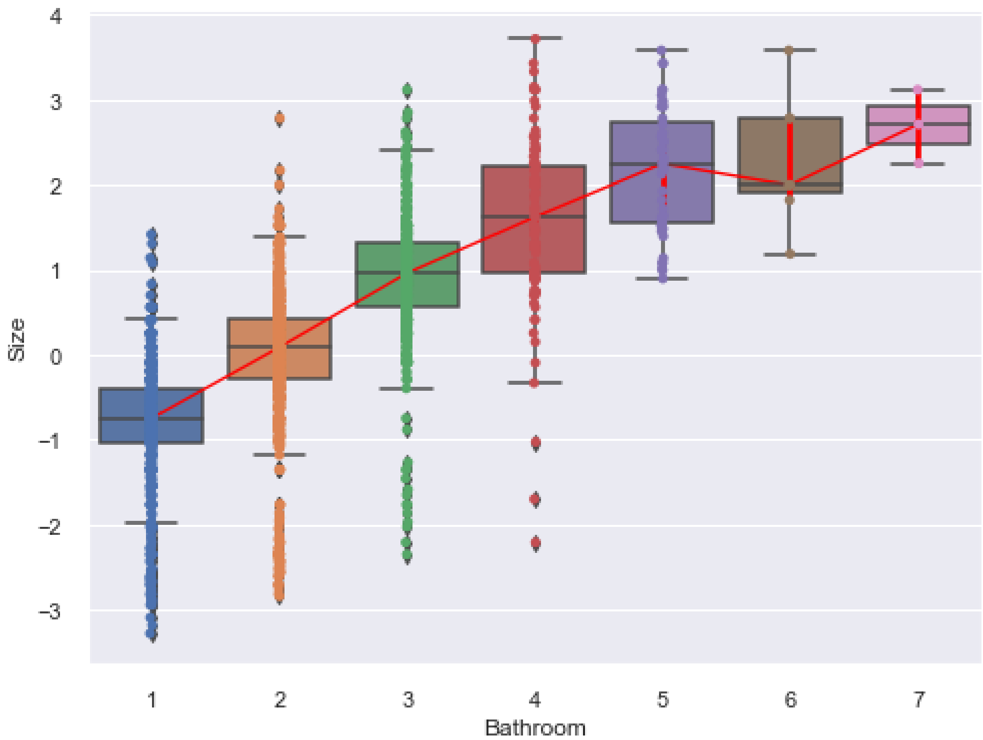




In [84]:

```
for x in dis_var:
    plt.figure(figsize=(8,6))
    sns.boxplot(data=X_train,x=x,y="Size")
    sns.pointplot(data=X_train,x=x,y="Size",color="red",estimator=np.median,
    errorbar=('ci',95),scale=0.5)
    sns.stripplot(data=X_train,x=x,y="Size",jitter=0.01)
    plt.show()
```





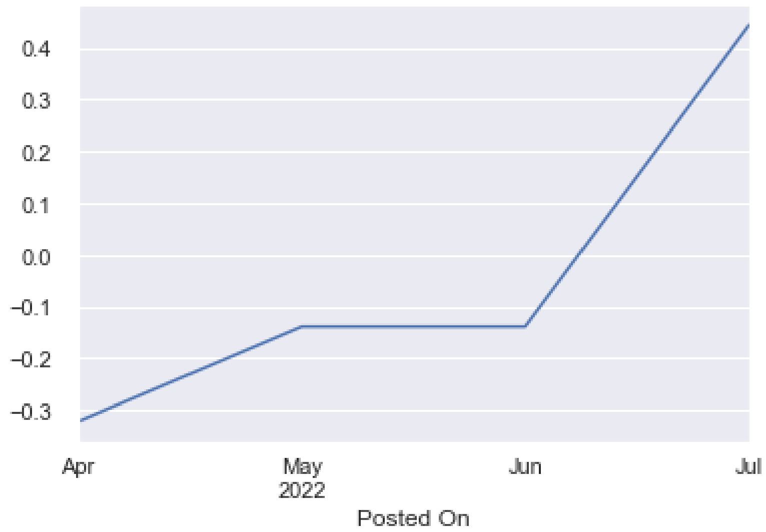
as we can see that Rent will increase with BHK and Bathroom increase, and this is make sense because when BHK and Bathroom increase the size of room will increase too.

In [85]:

```
X_train.groupby(pd.Grouper(key='Posted On', freq='M'))["Rent"].median().plot()
```

Out[85]:

```
<AxesSubplot:xlabel='Posted On'>
```



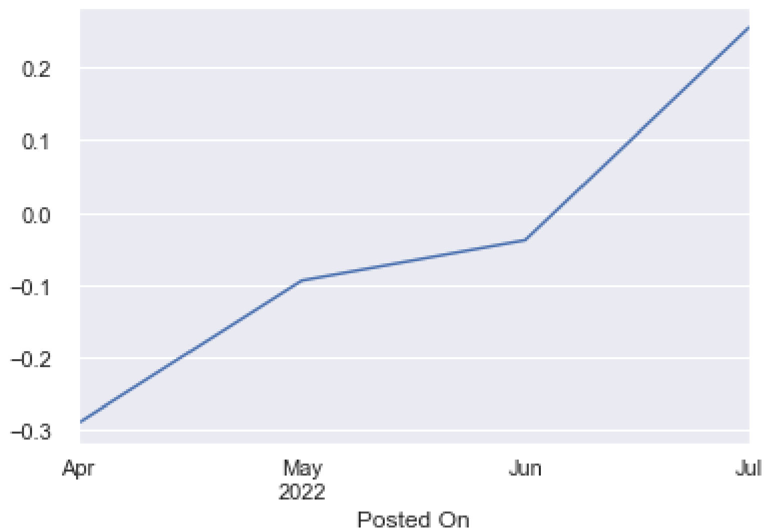
seen if the house that was posted recently has a higher rental price

In [86]:

```
X_train.groupby(pd.Grouper(key='Posted On', freq='M'))["Size"].median().plot()
```

Out[86]:

```
<AxesSubplot:xlabel='Posted On'>
```



The increase in house rent prices is increasing every month, supported by the size of the houses posted recently that are also getting bigger.

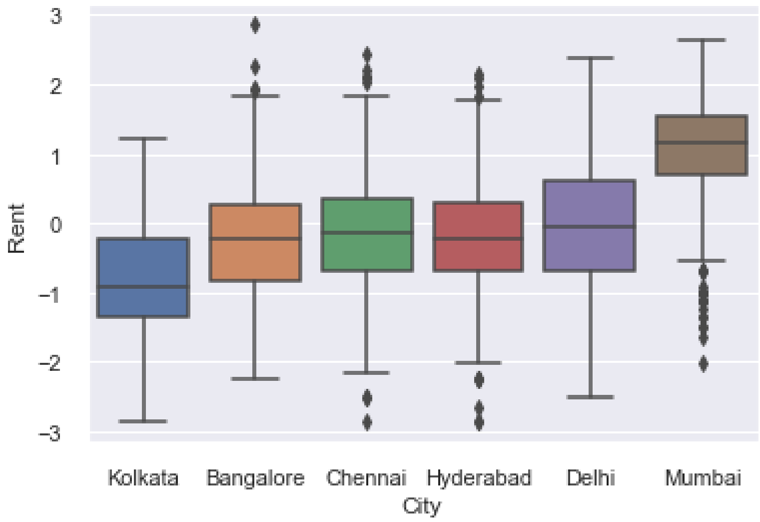
City

In [87]:

```
sns.boxplot(data=X_train,x="City",y="Rent",
            order=["Kolkata","Bangalore","Chennai","Hyderabad",
            "Delhi","Mumbai"])
```

Out[87]:

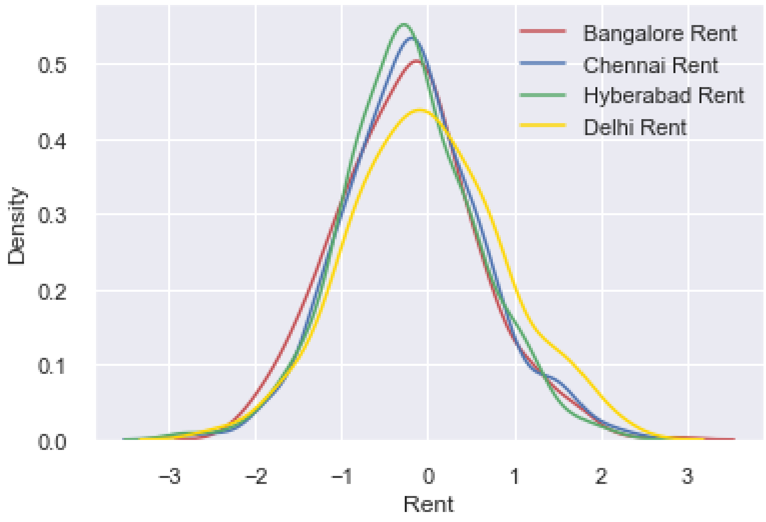
```
<AxesSubplot:xlabel='City', ylabel='Rent'>
```



As we can see, the rent house in Kolkata are the lowest and Mumbai is the most expensive city to rent, which is quite reasonable because Mumbai is a big city. Bangalore Chennai, Hyderabad and Delhi need to be tested to determine if the difference is significant

In [88]:

```
sns.distplot(X_train[X_train["City"]=="Bangalore"]["Rent"],color="r",label='Bangalore Rent'
sns.distplot(X_train[X_train["City"]=="Chennai"]["Rent"],color="b",label='Chennai Rent',his
sns.distplot(X_train[X_train["City"]=="Hyderabad"]["Rent"],color="g",label='Hyberabad Rent'
sns.distplot(X_train[X_train["City"]=="Delhi"]["Rent"],color="gold",label='Delhi Rent',hist

plt.legend(loc="best")
plt.show();
```



In [89]:

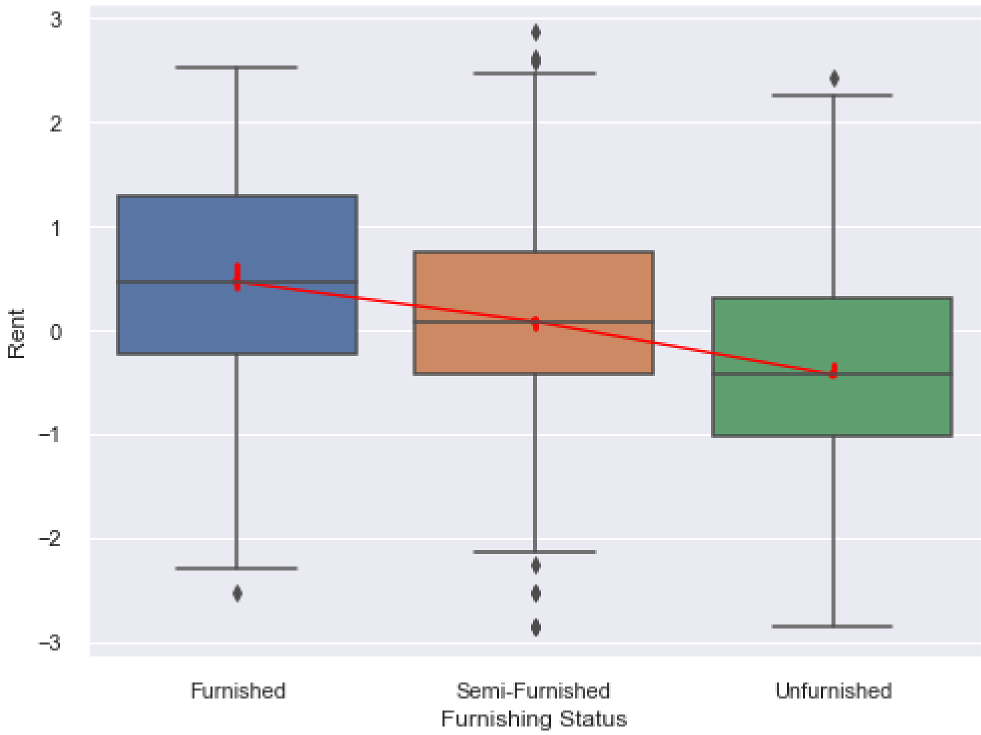
```
Bangalore = X_train[X_train["City"]=="Bangalore"]["Rent"]
Chennai = X_train[X_train["City"]=="Chennai"]["Rent"]
Hyderabad = X_train[X_train["City"]=="Hyderabad"]["Rent"]
Delhi = X_train[X_train["City"]=="Delhi"]["Rent"]

ftest,p = f_oneway(Bangalore,Chennai,Hyderabad,Delhi)

if p<0.05:
    print(f"Significant different, f-test {ftest}")
else:
    print('Not Significant Different')
```

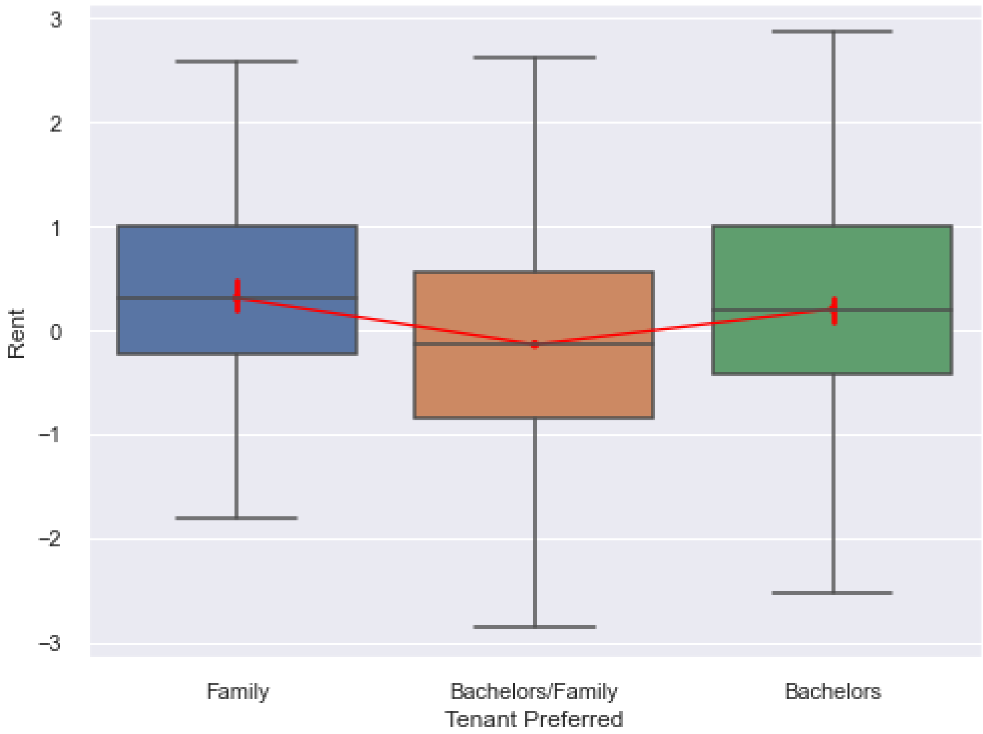
```
Significant different, f-test 7.336268909077812
```

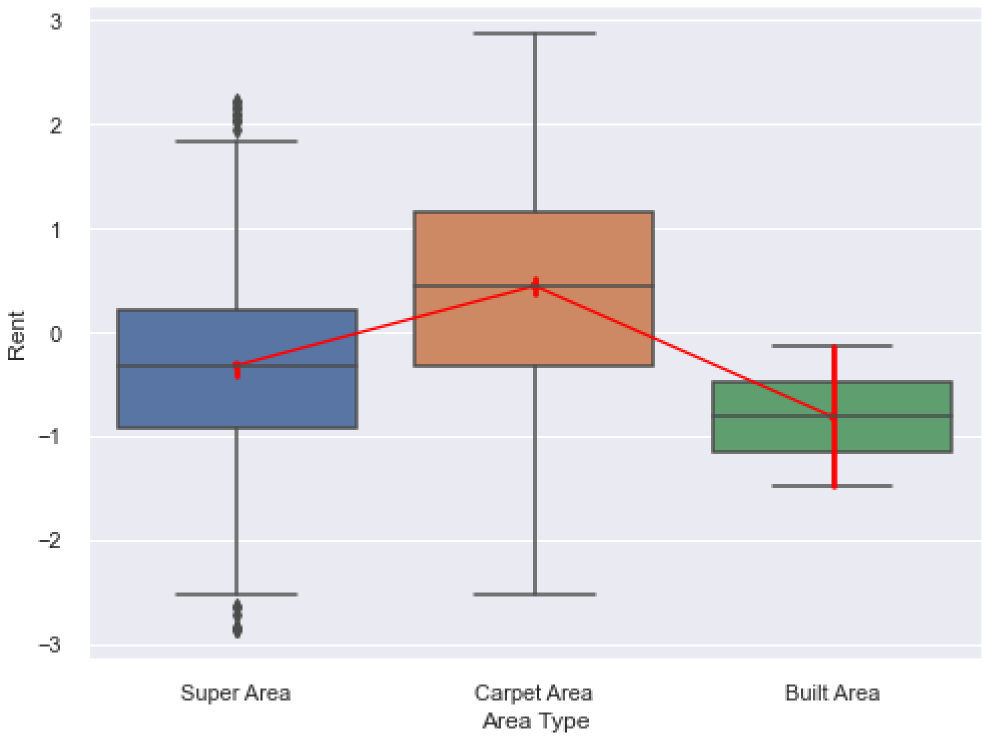
because Ho reject, we can confidence that rent house Bangolore,Chennai Hyderabad,delhi at least different

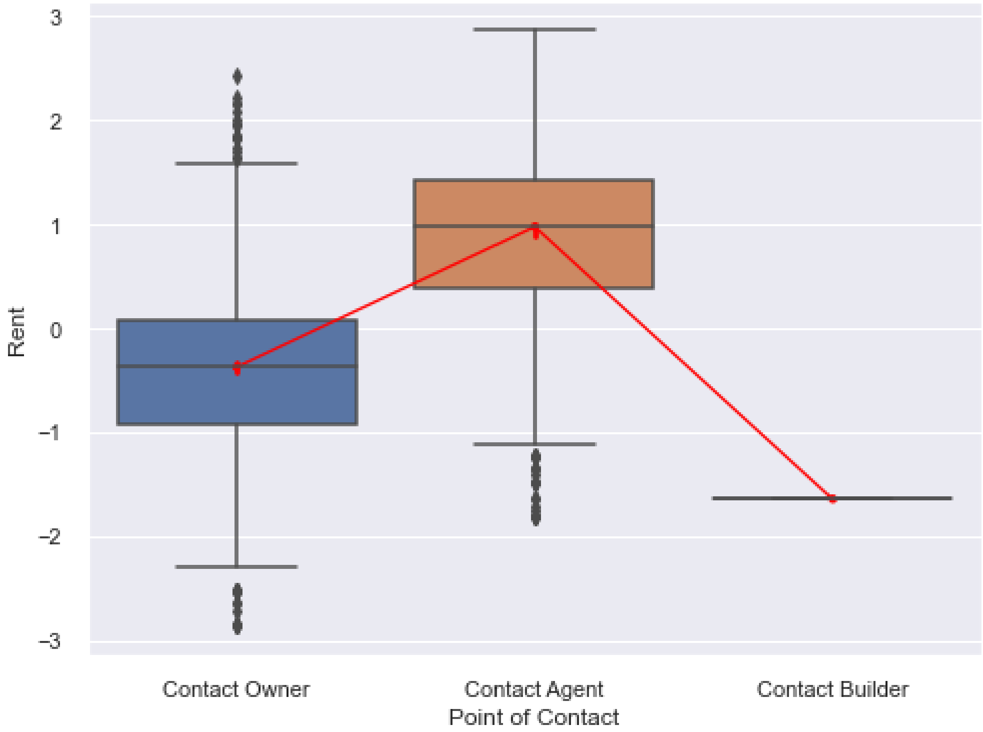
In [90]:

```
for x in ["Furnishing Status","Tenant Preferred","Area Type","Point of Contact"]:
    plt.figure(figsize=(8,6))
    sns.boxplot(data=X_train,x=x,y="Rent")
    sns.pointplot(data=X_train,x=x,y="Rent",color="red",estimator=np.median,
    errorbar=('ci',95),scale=0.5)
    #sns.stripplot(data=X_train,x=x,y="Size",jitter=0.01)
    plt.show()
```




3
2
1
0
−1
−2
−3
Rent
Family
Bachelors/Family
Bachelors
Tenant Preferred

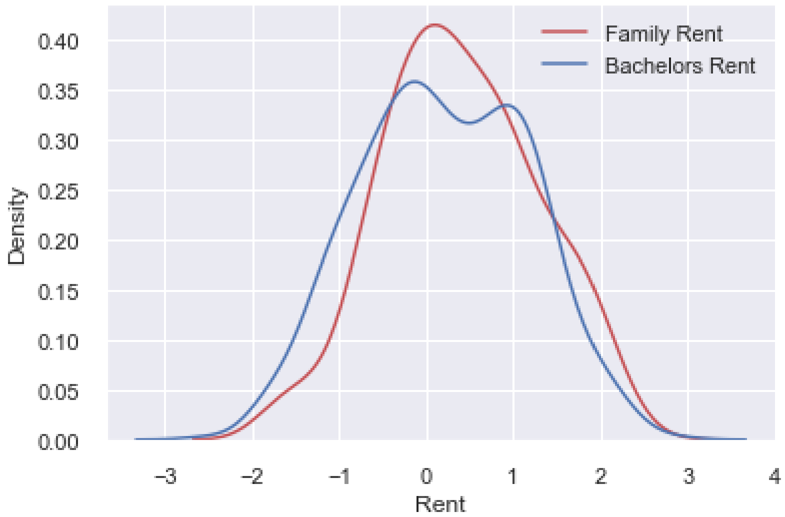

3
2
1
0
−1
−2
−3
Rent
Super Area
Carpet Area
Built Area
Area Type



as we can see that furnishing status, Area type, Point of Contact have different Rent.
for tenant preferred we need to check if family and bachelor have rent significant different

In [91]:

```
sns.distplot(X_train[X_train["Tenant Preferred"]=="Family"]["Rent"],color="r",label='Family
sns.distplot(X_train[X_train["Tenant Preferred"]=="Bachelors"]["Rent"],color="b",label='Bac

plt.legend(loc="best")
plt.show();
```



Check Varience between Tenant Preferred Family and Bachelors

In [92]:

```
# Ho = Variance is same
# H1 = Variance is not same
from scipy.stats import levene
levene(X_train[X_train["Tenant Preferred"]=="Family"]["Rent"],
X_train[X_train["Tenant Preferred"]=="Bachelors"]["Rent"])
```

Out[92]:

```
LeveneResult(statistic=3.645577203402605, pvalue=0.05653711360112609)
```

Because Ho Accept, so the variance Rent between Tenant Preferred Family and Bachelors is same, so we can move forward to mann whitney u test to see if Rent between Family and bachelors is significant difference. drop this columns

Is median differrent ?

In [93]:

```
# Ho = Median not significantly different
# H1 = Median significantly different
from scipy.stats import mannwhitneyu
mannwhitneyu(X_train[X_train["Tenant Preferred"]=="Family"]["Rent"],
X_train[X_train["Tenant Preferred"]=="Bachelors"]["Rent"])
```

Out[93]:

```
MannwhitneyuResult(statistic=102081.0, pvalue=0.008183720313892183)
```

because we accept Ho, so Rent between family and bachelors is not significant difference

Data Preparation

In [94]:

```
#drop posted on Area Locality,Floors,Total_Number_of_Floors,Address
X_train = X_train.drop(["Posted On","Area Locality","Floors","Total_Number_of_Floors","Tena
"Address"],axis=1)

X_train.head()
```

Out[94]:

	BHK	Size	Area Type	City	Furnishing Status	Bathroom	Point of Contact	latitude	longitude
2707	2	0.087204	Super Area	Bangalore	Furnished	1	Contact Owner	12.962267	77.530001
1188	1	-0.462179	Super Area	Mumbai	Semi-Furnished	2	Contact Owner	19.318390	72.899170
4637	2	0.631527	Super Area	Bangalore	Semi-Furnished	2	Contact Owner	12.976794	77.590082
1730	1	-1.026502	Super Area	Bangalore	Semi-Furnished	1	Contact Owner	13.040052	77.557389
3534	2	0.560595	Super Area	Chennai	Semi-Furnished	2	Contact Owner	13.050338	80.229938

In [95]:

```
mapping_poc = {"Contact Builder":1,
                "Contact Owner":2,
                "Contact Agent":3}

mapping_area = {"Built Area":1,
                "Super Area":2,
                "Carpet Area":3}


mapping_furnishing = {'Unfurnished':1,
                      'Semi-Furnished':2,
                      'Furnished':3}

mapping_city  = {"Kolkata":1,
                 "Bangalore":2,
                 "Hyderabad":2,
                 "Chennai":3,
                 "Delhi":4,
                 "Mumbai":5}
```

In [96]:

```
#area
X_train["Area Type"] = X_train["Area Type"].map(mapping_area)
X_test["Area Type"] = X_test["Area Type"].map(mapping_area)

#City
X_train["City"] = X_train["City"].map(mapping_city)
X_test["City"] = X_test["City"].map(mapping_city)

#Furnishing Status
X_train["Furnishing Status"] = X_train["Furnishing Status"].map(mapping_furnishing)
X_test["Furnishing Status"] = X_test["Furnishing Status"].map(mapping_furnishing)


#point of contact
X_train["Point of Contact"] = X_train["Point of Contact"].map(mapping_poc)
X_test["Point of Contact"] = X_test["Point of Contact"].map(mapping_poc)
```

In [97]:

```
y_train = X_train["Rent"]
X_train = X_train.drop('Rent',axis=1)

X_test = X_test[X_train.columns]
```

In [98]:

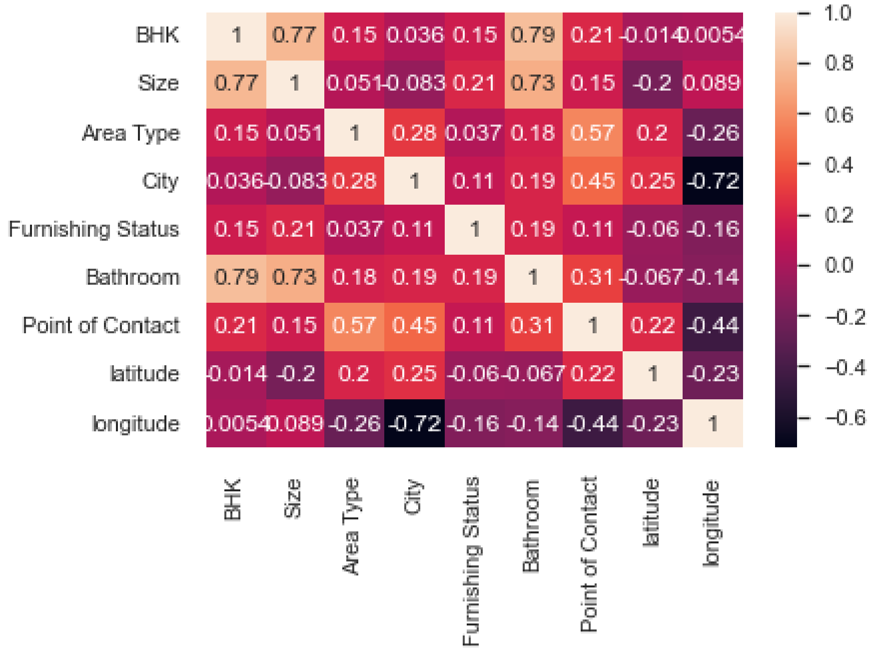
```
X_test['Rent'] = y_test

X_test[["Rent","Size"]] = pt.transform(np.array(X_test[["Rent",'Size']]))
y_test = X_test['Rent']

X_test = X_test[X_train.columns]
```

In [99]:

```
sns.heatmap(X_train.corr(method='spearman'),
            annot=True,
            fmt='.2g')
```

Out[99]:

```
<AxesSubplot:>
```



Modelling

In [100]:

```
def model_train(x_train,y_train,x_test,y_test,model):
    model.fit(x_train,y_train)
    predict = model.predict(x_train)
    predit_test = model.predict(x_test)
    print(f'R2 score training {r2_score(y_train,predict)}')
    print(f'MAE score training {mean_absolute_error(y_train,predict)}')
    print(f'RMSE score training {np.sqrt(mean_absolute_error(y_train,predict))}')
    print(f'Median absolute error score training {median_absolute_error(y_train,predict)}')
    print("--"*50)
    print(f'R2 score test {r2_score(y_test,predit_test)}')
    print(f'MAE score test {mean_absolute_error(y_test,predit_test)}')
    print(f'RMSE score test {np.sqrt(mean_absolute_error(y_test,predit_test))}')
    print(f'Median absolute error score test {median_absolute_error(y_test,predit_test)}')
```

Linear Regression

In [101]:

```
linreg = LinearRegression()
model_train(X_train,y_train,X_test,y_test,linreg)
```

```
R2 score training 0.7689857105087458
MAE score training 0.3651548131506384
RMSE score training 0.604280409371873
Median absolute error score training 0.28991156761416276
---------------------------------------------------------------------------
------------------------
R2 score test 0.7235067046038224
MAE score test 0.3745418640284296
RMSE score test 0.6119982549226995
Median absolute error score test 0.2880171622283637
```

Decision Tree

In [102]:

```
dt = DecisionTreeRegressor(random_state=42)
model_train(X_train,y_train,X_test,y_test,dt)
```

```
R2 score training 0.9926808553565561
MAE score training 0.019624412065435713
RMSE score training 0.1400871588170583
Median absolute error score training 0.0
---------------------------------------------------------------------------
------------------------
R2 score test 0.6109164170380214
MAE score test 0.42545723492501386
RMSE score test 0.6522708294297805
Median absolute error score test 0.3093289752611289
```

Random Forest

In [103]:

```
rf = RandomForestRegressor(random_state=42)
model_train(X_train,y_train,X_test,y_test,rf)
```

```
R2 score training 0.9687785499402934
MAE score training 0.12633624455577902
RMSE score training 0.35543810228474243
Median absolute error score training 0.09208062152276769
---------------------------------------------------------------------------
------------------------
R2 score test 0.7606666910159066
MAE score test 0.33105709174204584
RMSE score test 0.5753756092693241
Median absolute error score test 0.24184834808728045
```

XGBOOST

In [104]:

```
xgb  = XGBRFRegressor(random_state=42)
model_train(X_train,y_train,X_test,y_test,xgb)
```

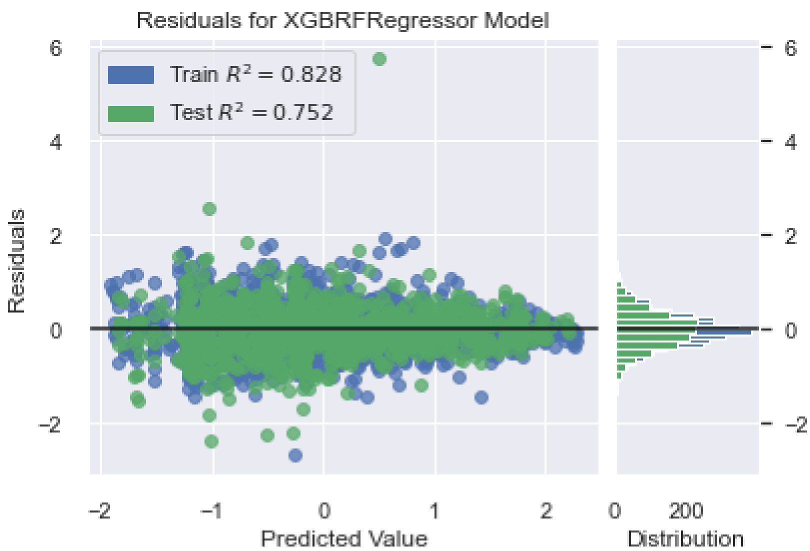
```
R2 score training 0.8284860529192394
MAE score training 0.3087166390954929
RMSE score training 0.5556227489002704
Median absolute error score training 0.24429503216181553
----------------------------------------------------------------------------
------------------------
R2 score test 0.7522237539579457
MAE score test 0.3497777109532276
RMSE score test 0.5914200799374567
Median absolute error score test 0.2681802019716182
```

best model is XGBoost Regressor

Assumption Test

In [105]:

```
visualizer = ResidualsPlot(xgb,hist=True)

visualizer.fit(X_train, y_train)  # Fit the training data to the visualizer
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
visualizer.show()
```



Out[105]:

```
<AxesSubplot:title={'center':'Residuals for XGBRFRegressor Model'}, xlabel
='Predicted Value', ylabel='Residuals'>
```

residual random and based on histogram residual distribution is normal distributed

In [106]:

```
visualizer = PredictionError(xgb)

visualizer.fit(X_train, y_train)  # Fit the training data to the visualizer
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
visualizer.show()
```

Out[106]:

```
<AxesSubplot:title={'center':'Prediction Error for XGBRFRegressor'}, xlabel
='$y$', ylabel='$\\hat{y}$'>
```

features with the XGBoost model can explain 75% of the data and the remaining 25% are other factors

Explainer

In [107]:

```
import shap
```

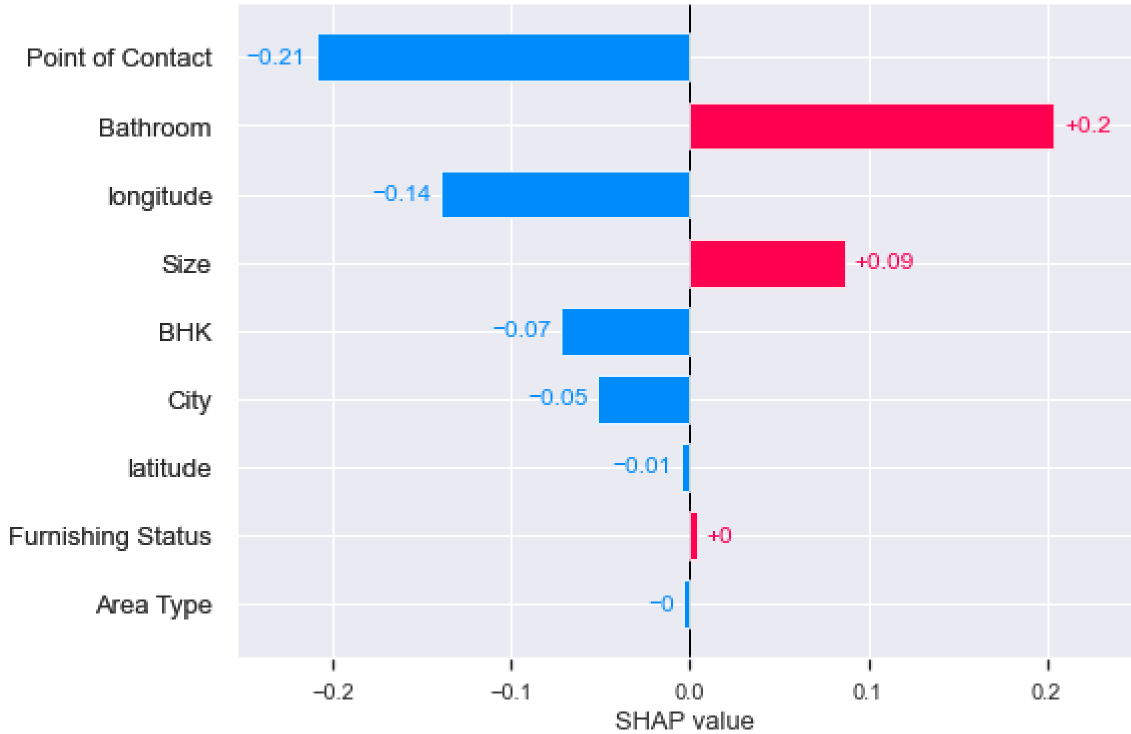
In [109]:

```
# Fits the explainer
explainer = shap.Explainer(xgb.predict, X_test)

# Calculates the SHAP values
shap_values = explainer(X_test)
```

```
Exact explainer: 1425it [01:32, 14.51it/s]
```

In [111]:

```
shap.plots.bar(shap_values[0])
```



Positive Impact to Rent :

1.Bathroom have positve impact to prediction Rent, contributing avegrage +0.2 usd (scalling level)

2.Size have positve impact to prediction Rent contributing avegrage +0.09 usd (scalling level)

Negative impact :

1.Point of Contract have negative impact to prediction Rent, contributing -0.21 usd (scalling level)