RENT PRICE PREDICTION FOR HOUSE

Done By,

Aiswariya C Nair

Anjana Krishnan

1.INTRODUCTION

1.1 PROJECT OVERVIEW

The main aim of this project is to create a model based on historical data and estimating the leaving expenses on some given city. House rent changed to respect with city so in this project data is taken based on different-different city's so that the peoples can understand based on different-different city's what is the leaving expenses in that particular city. Real estate is the least transparent industry in our ecosystem .Housing prices keep changing day in and day out and sometimes are hyped rather than being based on valuation. Predicting housing prices with real factors is the main crux of our research project.

Here we aim to make our evaluations based on every basic parameter that is considered while determining the price. Our goal is to deliver a perfect software which will be benefitting our user in an interactive way. The focus is to create an "easy to use" website, which will allow a first time customer to complete their needs with ease. To make a customizable system including maximum option. More appealing system and main page. To increase the ease of productivity.

1.2 PURPOSE

We want to provide a system which is close to big companies like 99acres and many others. We want to overcome the problems of existing price prediction of house in the market. Providing better services than the previous ones. Removal of data storing through manual means. The main aim of this project is to create a model based on historical data and estimating the leaving expenses on some given city. House rent changed to respect with city so in this project data is taken based on different-different city's so that the peoples can understand based on different-different city's what is the leaving expenses in that particular city.

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

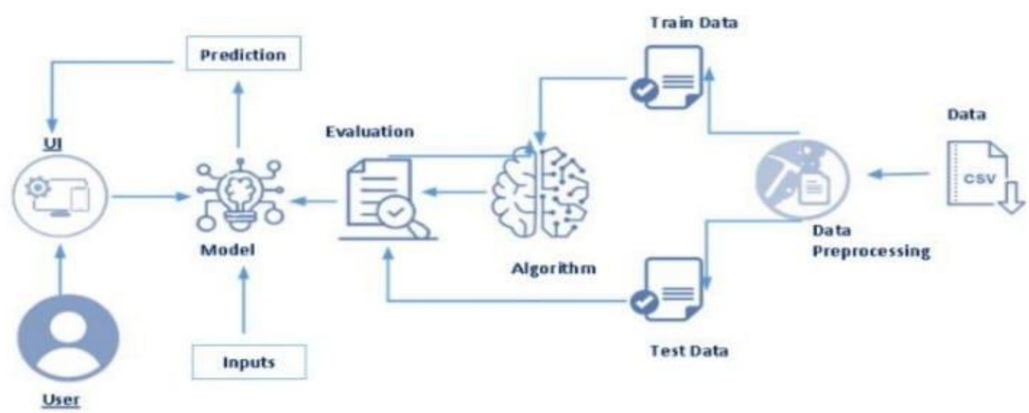
It does not offer customization. Existing system requires a great amount of manual work has to be done. The amount of manual work increases exponentially with increase in services. Needs a lot of working staff and extra attention on all the records. It requires a reliable internet connection. System may provide inaccurate results if data not entered correctly. Major problem was lack of security.

2.2 PROPOSED SYSTEM

Rent price prediction for house can help the developer determine the selling price of a house and can help the customer to arrange the right time to purchase a house. There are three factors that influence the price of a house which include physical conditions, concept and location. As earlier, House prices were determined by calculating the acquiring and selling price in a locality. Therefore, the Rent Price Prediction for House model is very essential in filling the information gap.

3.THEORITICAL ANALYSIS

- **3.1 BLOCK DIAGRAM**



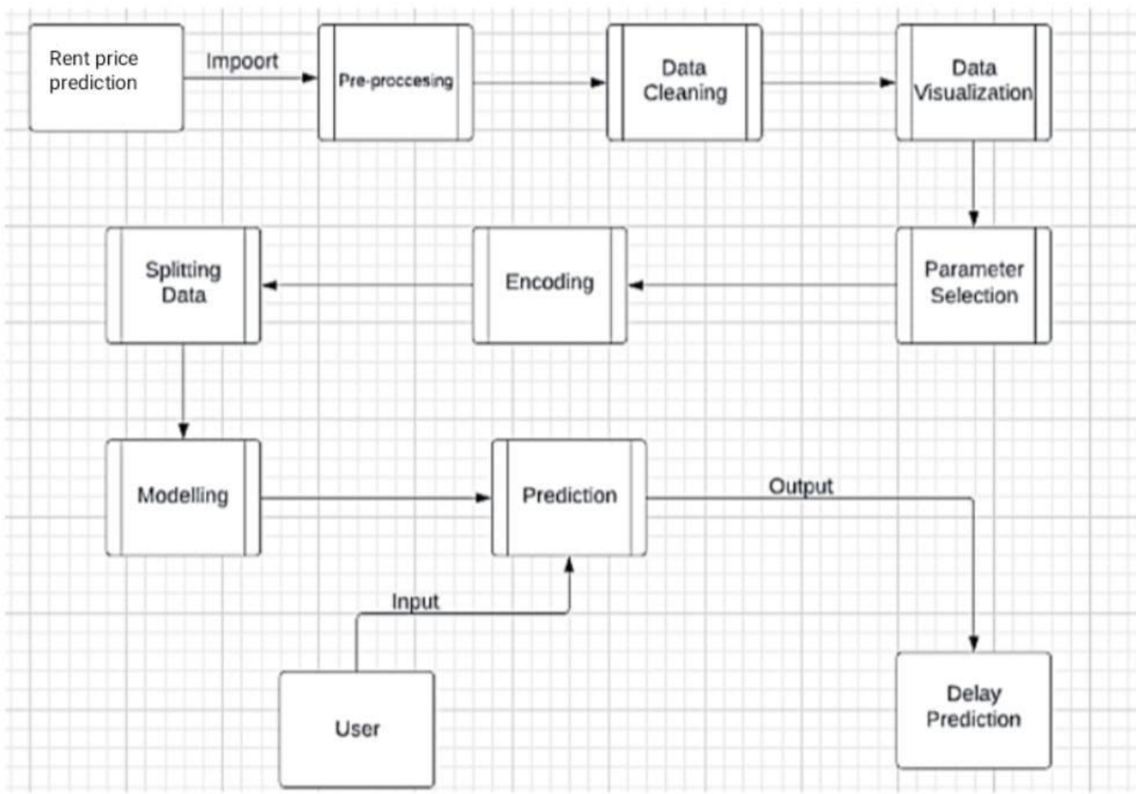
3.2 HARDWARE/SOFTWARE DESIGNING

- Laptop
- Anaconda Navigator
- Jupyter Notebook
- Spyder
- IBM Cloud

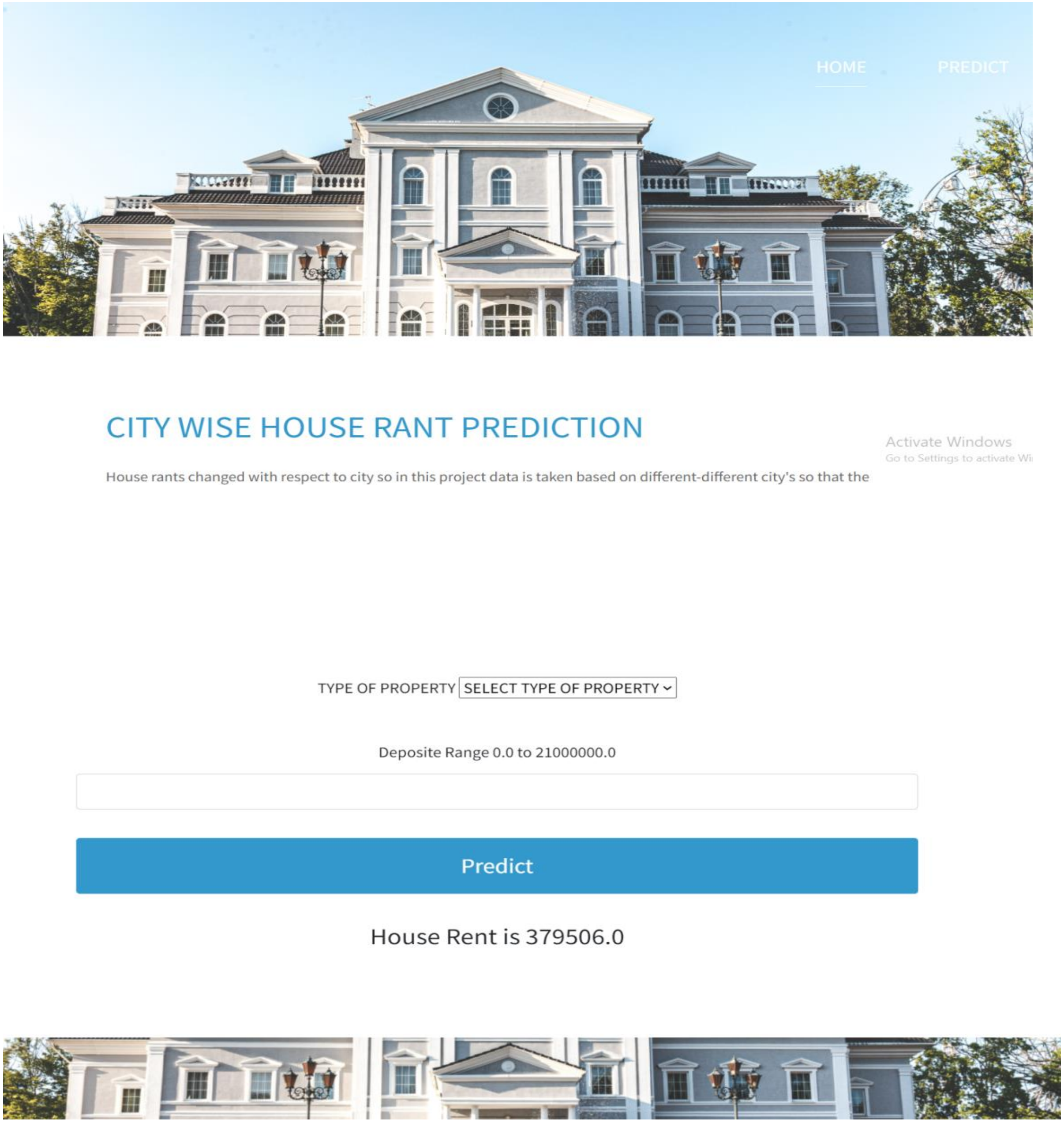
4.EXPERIMENTAL INVESTIGATIONS

While working with the model we get to find out the calculations of house rents that are being carried out. By calculating all the inputs given by the user ,the house rents are predicted according to their priority bases.

5.FLOWCHART



6.RESULT


HOME
PREDICT
CITY WISE HOUSE RANT PREDICTION
Activate Windows
Go to Settings to activate Wi
House rants changed with respect to city so in this project data is taken based on different-different city's so that the
TYPE OF PROPERTY
SELECT TYPE OF PROPERTY
Deposite Range 0.0 to 21000000.0
Predict
House Rent is 379506.0

7.ADVANTAGES

It overcomes all the problems of existing system. Buying and Selling of house can be done in more convenient way. Payment can be easily done using online method. It makes system very effective for buying and selling of house. Admin can view all reports regarding buying and selling of house which can be helpful for decision making. Easy add update delete process.

DISADVANTAGES

Rising property prices can also discourage productive lending, and lead to capital being misallocated. When housing markets boom, banks tend to engage in more mortgage lending. But because lenders face capital constraints, this is often accompanied by reduced lending to businesses

8.APPLICATIONS

Rent price prediction for house can help the developer determine the selling price of a house and can help the customer to arrange the right time to purchase a house. There are three factors that influence the price of a house which include physical conditions, concept and location. As earlier, House prices were determined by calculating the acquiring and selling price in a locality. Therefore, the Rent Price Prediction for House model is very essential in filling the information gap.

9.CONCLUSION

Prediction house prices are expected to help people who plan to buy a house so they can know the price range in the future, then they can plan their finance well. In addition, house price predictions are also beneficial for property investors to know the trend of housing prices in a certain location.

10.FUTURE SCOPE

Better customization-The system provides the user with three tabs: one for selecting house as per their need, one for buy and selling option, and one for meeting with the owner of house.Hence the system will decrease workload of the employees and benefit the admin due to the database/information system as the information will be stored in the system and can be viewed at any time.The system will be able to guide a user through the website and make them to complete their buy and selling of house through interactive menu.

11.BIBILOGRAPHY

SmartInternz student portal

YouTube

APPENDIX

Predict :- Monthly Rent

In [70]:
```python
#Importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
```

In [71]:
```python
# Loading DataSet
df = pd.read_csv('99acres_data.csv')
df
```

Out[71]:

	city	monthly_rant	BHKS	Baths	sqft_per_inch	build_up_area	Type_of_property	location_of_the_property	deposit
0	mumbai	17500.0	1.0	2 Baths	470	Carpet Area	Residential	Kolshet Road	75000.0
1	mumbai	75000.0	3.0	3 Baths	1800	Super built-up Area	Residential	Sector 21 Nerul	400000.0
2	mumbai	60000.0	2.0	2 Baths	950	Super built-up Area	Residential	Wadala	200000.0
3	mumbai	52000.0	3.0	3 Baths	1300	Carpet Area	Residential	Hiranandani Estate	300000.0
4	mumbai	30000.0	1.0	1 Bath	550	Built-up Area	Residential	Kanjurmarg (East)	150000.0
...	...	...	...	...	...	...	...	...	...
146523	Trivandraum	10000.0	2.0	3 Baths	1200	Built-up Area	Independent	Anayara	25000.0
146524	Trivandraum	21000.0	2.0	2 Baths	1155	Carpet Area	Residential	Kazhakkoottam	50000.0
146525	Trivandraum	10000.0	2.0	2 Baths	861	Built-up Area	Residential	Vattiyoorkkav	30000.0
146526	Trivandraum	33000.0	4.0	5 Baths	3200	Plot Area	Independent	Pattom	150000.0
146527	Trivandraum	8000.0	4.0	5 Baths	2178	Plot Area	Independent	Kallampally	24000.0

146528 rows × 9 columns

In [72]:
```python
df.describe()
```

Out[72]:

	monthly_rant	BHKS	sqft_per_inch	deposit
count	1.465280e+05	146528.000000	1.465280e+05	1.465280e+05
mean	3.414242e+04	2.159703	2.463806e+03	1.203667e+05
std	8.428243e+04	1.107673	1.816119e+05	2.937736e+05
min	5.000000e+02	1.000000	1.000000e+00	0.000000e+00
25%	1.300000e+04	1.000000	6.800000e+02	3.000000e+04
50%	2.000000e+04	2.000000	1.057000e+03	6.000000e+04
75%	3.290000e+04	3.000000	1.500000e+03	1.250000e+05
[illegible]	[illegible]	[illegible]	4.356000e+07	2.100000e+07

63243634c8fdbdf38de5f4719d2a56631fddd56b40d579

In [73]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146528 entries, 0 to 146527
Data columns (total 9 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   city                      146528 non-null  object
 1   monthly_rant              146528 non-null  float64
 2   BHKS                      146528 non-null  float64
 3   Baths                     146528 non-null  object
 4   sqft_per_inch             146528 non-null  int64
 5   build_up_area             146528 non-null  object
 6   Type_of_property          146528 non-null  object
 7   location_of_the_property  146528 non-null  object
 8   deposit                   146528 non-null  float64
dtypes: float64(3), int64(1), object(5)
memory usage: 10.1+ MB
```

In [74]:
```python
df['Baths'] = df['Baths'].str.split(" ", n = 2, expand = True)[0]
```

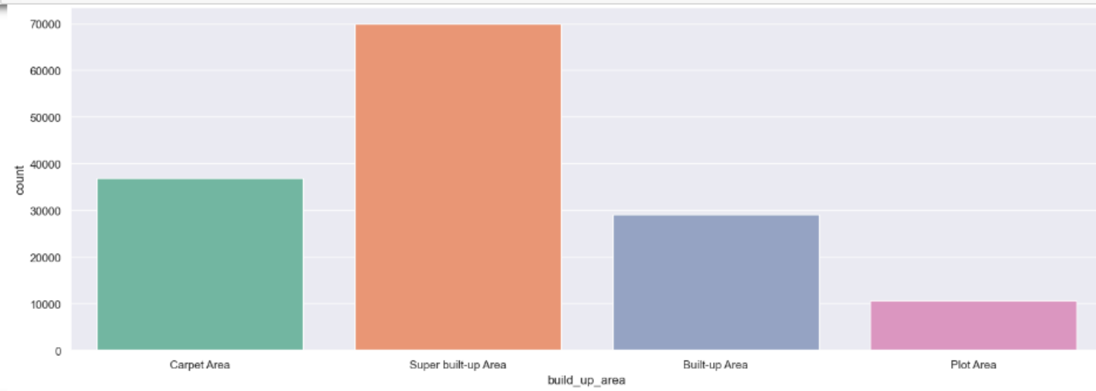
In [75]:
```python
df.Baths.unique()
```

Out[75]:
```
array(['2', '3', '1', '5', '4', '10', '6', 'RK\n1', 'Baths', 'Bath', '9',
       '15', '76', '8', '7', '12', 'BHK', '22', '16', '20', '17', '13',
       '11', '18', '30', '24', '14', '23', '59', '48', 'RK', '40', '98',
       '19', '26', '35', '28', '21', '37', '27', '50'], dtype=object)
```

ks/Training/House Rent Prediction GUided - LE.ipynb#

Data Visulization

In [76]:
```
print("Numbers Of Area Type :")
print()
print(df['build_up_area'].value_counts())
sns.set(rc = {'figure.figsize':(18,6)})
sns.countplot(x='build_up_area', data=df, palette = 'Set2')
```

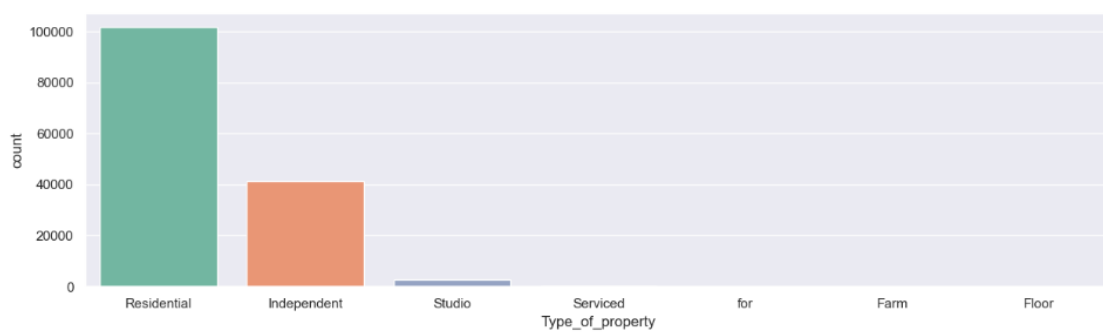


In [77]:
```
print("Types of Property :")
print()
print(df['Type_of_property'].value_counts())
sns.set(rc = {'figure.figsize':(15,4)})
sns.countplot(x='Type_of_property', data=df, palette = 'Set2')
```

```
Types of Property :

Residential    102024
Independent     41243
Studio           2606
Serviced          363
Farm              161
for               118
Floor              13
Name: Type_of_property, dtype: int64
```

Out[77]: <AxesSubplot:xlabel='Type_of_property', ylabel='count'>



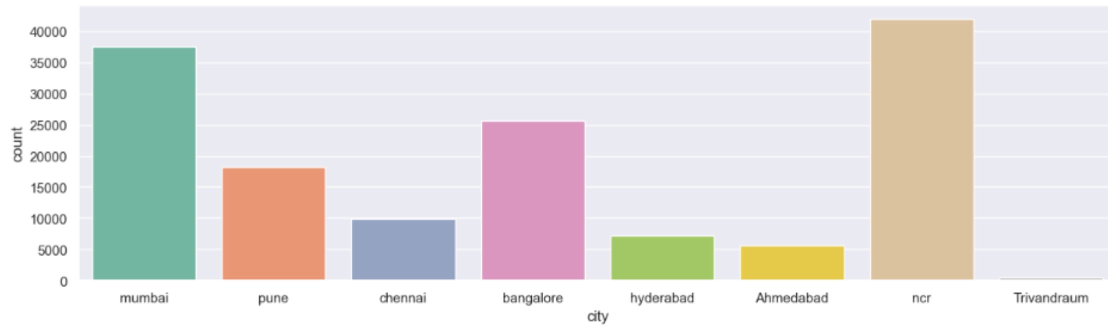
In [*]:
```
print("Property Located Based On City :- ")
print()
print(df['city'].value_counts())
sns.set(rc = {'figure.figsize':(15,4)})
sns.countplot(x='city', data=df, palette = 'Set2')
```

In [79]:
```
print("Property Located Based On City :- ")
print()
print(df.groupby('city')['location_of_the_property'].value_counts())
sns.set(rc = {'figure.figsize':(15,4)})
sns.countplot(x='city', data=df, palette = 'Set2')
```

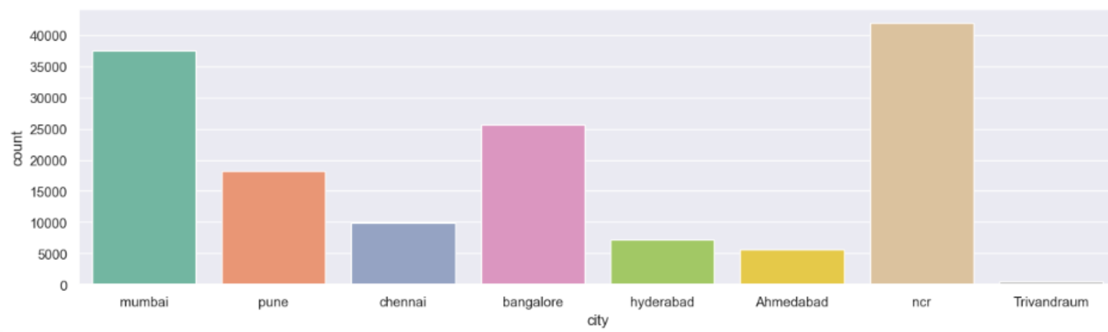
```
Property Located Based On City :-

city       location_of_the_property
Ahmedabad  South Bopal                 542
           Satellite                   372
           Vaishnodevi Circle          240
           Thaltej                     232
           Gota                        227
                                      ...
pune       wadheshwar nagar              1
           wakad                         1
           wakad ,pune                   1
           wakad bridge                  1
           yerwada                       1
Name: location_of_the_property, Length: 12859, dtype: int64
```

Out[79]: <AxesSubplot:xlabel='city', ylabel='count'>



Out[79]: <AxesSubplot:xlabel='city', ylabel='count'>



Checking Correlation

In [80]:
```
df.corr()
```

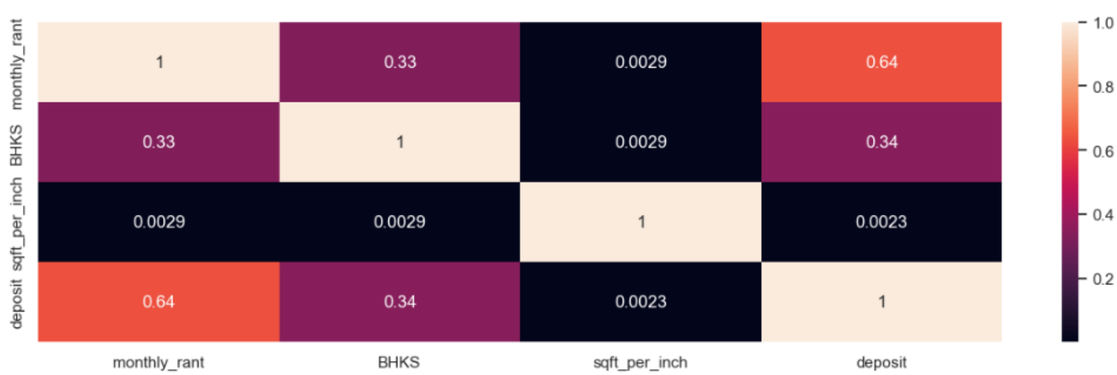
Out[80]:

	monthly_rant	BHKS	sqft_per_inch	deposit
monthly_rant	1.000000	0.328428	0.002948	0.635764
BHKS	0.328428	1.000000	0.002898	0.341362
sqft_per_inch	0.002948	0.002898	1.000000	0.002275
deposit	0.635764	0.341362	0.002275	1.000000

In [81]:
```
corr = df.corr()
```

In [82]:
```
sns.heatmap(corr,annot=True)
```

Out[82]: <AxesSubplot:>



Column Baths

In [83]:
```
# Droping a features
df.drop('Baths',axis=1,inplace=True)
```

Column Deposite

In [84]:
```
#Checking Null values
df.isnull().sum()
```

Out[84]:
```
city                        0
monthly_rant                0
BHKS                        0
sqft_per_inch               0
build_up_area               0
Type_of_property            0
location_of_the_property    0
deposit                     0
dtype: int64
```

In [85]:
```
# How many unique categories is there
df.build_up_area.unique()
```

In [86]:
```
df.head(2)
```

Out[86]:

	city	monthly_rant	BHKS	sqft_per_inch	build_up_area	Type_of_property	location_of_the_property	deposit
0	mumbai	17500.0	1.0	470	Carpet Area	Residential	Kolshet Road	75000.0
1	mumbai	75000.0	3.0	1800	Super built-up Area	Residential	Sector 21 Nerul	400000.0

Column Type Of Property

In [87]:
```
# Checking Unique Values
df.Type_of_property.unique()
```

Out[87]:
```
array(['Residential', 'Independent', 'Studio', 'Serviced', 'for', 'Farm',
       'Floor'], dtype=object)
```

In [88]:
```
#df['Type_of_property'] = df['Type_of_property'].str.replace("for",' ')
```

In [89]:
```
df.shape
```

Out[89]:
```
(146528, 8)
```

Filtering DataSet

In [90]:
```
# Removing for
df = df[df.Type_of_property!='for']
df.shape
```

Out[90]:
```
(146410, 8)
```

In [91]:
```
# Removing Serviced
df = df[df.Type_of_property!='Serviced']
df.shape
```

Out[91]:
```
(146047, 8)
```

In [92]:
```
# removing floor
df = df[df.Type_of_property!='Floor']
df.shape
```

Out[92]:
```
(146034, 8)
```

In [93]:
```
df.Type_of_property.unique()
```

Out[93]:
```
array(['Residential', 'Independent', 'Studio', 'Farm'], dtype=object)
```

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

Outlier Treatment

deposit

In [*]:
```
sns.boxplot(df['deposit'])
```

In [96]:
```
df['deposit'] = np.log(df['deposit']+1)
```

In [97]:
```
Q1 = df['deposit'].quantile(0.25)
Q3 = df['deposit'].quantile(0.75)
IQR = Q3 - Q1
```

In [98]:
```
((df['deposit'] < (Q1 - 1.5 * IQR)) | (df['deposit'] > (Q3 + 1.5 * IQR))).mean()

mask = (df['deposit'] < (Q1 - 1.5 * IQR)) | (df['deposit'] > (Q3 + 1.5 * IQR))
df[mask] = np.nan
```

In [*]:
```
sns.boxplot(df['deposit'])
```

- We clearly observed in boxplot data is right skewed

In [100]:
```
df.isnull().sum()
```

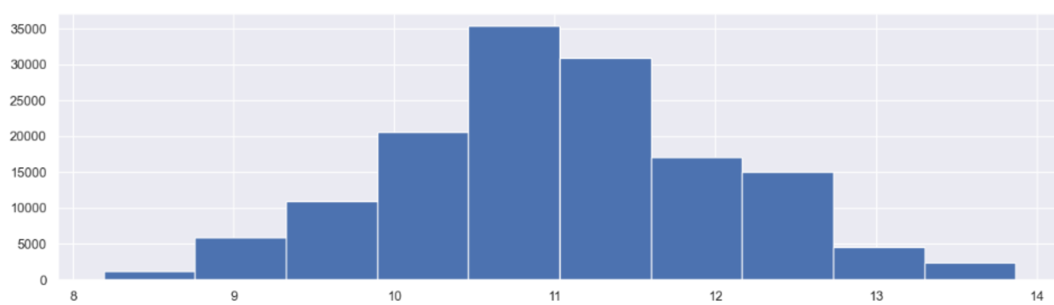
Out[100]:
```
city                        2263
monthly_rant                2263
BHKS                        2263
sqft_per_inch               2263
build_up_area               2263
Type_of_property            2263
location_of_the_property    2263
deposit                     2263
dtype: int64
```

In [101]:
```
df.dropna(inplace=True)
```

In [102]:
```
df['deposit'].hist()
```

In [102]: df['deposit'].hist()

Out[102]: <AxesSubplot:>



In [103]:
```
#df['deposit']=df['deposit']**(1/2)
#df['deposit'].hist()
```

In [104]:
```
df.head(2)
```
Out[104]:

	city	monthly_rant	BHKS	sqft_per_inch	build_up_area	Type_of_property	location_of_the_property	deposit
0	mumbai	17500.0	1.0	470.0	Carpet Area	Residential	Kolshet Road	11.225257
1	mumbai	75000.0	3.0	1800.0	Super built-up Area	Residential	Sector 21 Nerul	12.899222

sqft_per_inch

In [*]:
```
sns.boxplot(df.sqft_per_inch)
```
In [106]:
```
Q1 = df['sqft_per_inch'].quantile(0.25)
Q3 = df['sqft_per_inch'].quantile(0.75)
IQR = Q3 - Q1
```
In [107]:
```
((df['sqft_per_inch'] < (Q1 - 1.5 * IQR)) | (df['sqft_per_inch'] > (Q3 + 1.5 * IQR))).mean()

mask = (df['sqft_per_inch'] < (Q1 - 1.5 * IQR)) | (df['sqft_per_inch'] > (Q3 + 1.5 * IQR))
df[mask] = np.nan
```
In [*]:
```
sns.boxplot(df.sqft_per_inch)
```
In [109]:
```
df.isnull().sum()
```
Out[109]:
```
city                        6755
monthly_rant                6755
BHKS                        6755
sqft_per_inch               6755
build_up_area               6755
Type_of_property            6755
location_of_the_property    6755
deposit                     6755
dtype: int64
```
In [110]:
```
df.dropna(inplace=True)
```
In [111]:
```
df.head(2)
```
Out[111]:

	city	monthly_rant	BHKS	sqft_per_inch	build_up_area	Type_of_property	location_of_the_property	deposit
0	mumbai	17500.0	1.0	470.0	Carpet Area	Residential	Kolshet Road	11.225257
1	mumbai	75000.0	3.0	1800.0	Super built-up Area	Residential	Sector 21 Nerul	12.899222

In [112]:
```
df.drop('location_of_the_property',axis=1,inplace=True)
```
en=ab63243634c8fdbdf38de5f4719d2a56631fddd56b4(

In [113]:
```
df.shape
```
Out[113]: (137016, 7)

In [*]:
```
df.skew()
```
Data Transformation

In [115]:
```
df['monthly_rant'] = np.log(df['monthly_rant']+1)
```
In [*]:
```
df.skew()
```
In [117]:
```
df.BHKS.max(), df.BHKS.min()
```
Out[117]: (32.0, 1.0)

In [118]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 137016 entries, 0 to 146527
Data columns (total 7 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   city              137016 non-null  object
 1   monthly_rant      137016 non-null  float64
 2   BHKS              137016 non-null  float64
 3   sqft_per_inch     137016 non-null  float64
 4   build_up_area     137016 non-null  object
 5   Type_of_property  137016 non-null  object
 6   deposit           137016 non-null  float64
dtypes: float64(4), object(3)
memory usage: 8.4+ MB
```

Encoding

In [119]:
```
from sklearn.preprocessing import LabelEncoder

cty = LabelEncoder()
b_u_a = LabelEncoder()
T_o_p = LabelEncoder()
#l_o_t_p = LabelEncoder()


df['city'] = cty.fit_transform(df['city'])
df['build_up_area'] = b_u_a.fit_transform(df['build_up_area'])
df['Type_of_property'] = T_o_p.fit_transform(df['Type_of_property'])
#df['location_of_the_property'] = l_o_t_p.fit_transform(df['location_of_the_property'])
```

In [120]:
```
print("city",df['city'].unique())
print(cty.inverse_transform(list(df['city'].unique())))
print()
print("build_up_area:",df['build_up_area'].unique())
print(b_u_a.inverse_transform(list(df['build_up_area'].unique())))
print()
print("Type_of_property", df['Type_of_property'].unique())
print(T_o_p.inverse_transform(list(df['Type_of_property'].unique())))
print()
#print("location_of_the_property",df['location_of_the_property'].unique())
#print(l_o_t_p.inverse_transform(list(df['location_of_the_property'].unique())))
```

In [121]:
```
df.head()
```

Out[121]:

	city	monthly_rant	BHKS	sqft_per_inch	build_up_area	Type_of_property	deposit
0	5	9.770013	1.0	470.0	1	2	11.225257
1	5	11.225257	3.0	1800.0	3	2	12.899222
2	5	11.002117	2.0	950.0	3	2	12.206078
3	5	10.859018	3.0	1300.0	1	2	12.611541
4	5	10.308986	1.0	550.0	0	2	11.918397

In [122]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 137016 entries, 0 to 146527
Data columns (total 7 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   city              137016 non-null  int32
 1   monthly_rant      137016 non-null  float64
 2   BHKS              137016 non-null  float64
 3   sqft_per_inch     137016 non-null  float64
 4   build_up_area     137016 non-null  int32
 5   Type_of_property  137016 non-null  int32
 6   deposit           137016 non-null  float64
dtypes: float64(4), int32(3)
memory usage: 6.8 MB
```

---------------------------------Modeling---------------------------------------

In [123]:
```python
#Seperating the variable Independent matrix X and dependent Vector y

X = df.drop('monthly_rant',axis=1)
y = df.monthly_rant
```

In [124]:
```python
X.head(2)
```

Out[124]:

	city	BHKS	sqft_per_inch	build_up_area	Type_of_property	deposit
0	5	1.0	470.0	1	2	11.225257
1	5	3.0	1800.0	3	2	12.899222

In [125]:
```python
y.head(2)
```

Out[125]:
```
0     9.770013
1    11.225257
Name: monthly_rant, dtype: float64
```

In [126]:
```python
# Spliting the data into Training set & Test set
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, random_state=101)
```

In [127]:
```python
X_train.shape
```

Out[127]:
```
(109612, 6)
```

In [128]:
```python
# Random Forest
from sklearn.ensemble import RandomForestRegressor
forest = RandomForestRegressor()
forest.fit(X_train,y_train)
```

Out[128]:
```
RandomForestRegressor()
```

In [129]:
```python
y_predict=forest.predict(X_test)
y_predict_train=forest.predict(X_train)
```

In [130]:
```python
from sklearn.metrics import r2_score

print('Random Forest Train r2_score',r2_score(y_train,y_predict_train))
print('Random Forest Test r2_score',r2_score(y_test,y_predict))
```

```
Random Forest Train r2_score 0.9531628468121752
Random Forest Test r2_score 0.8778343005157727
```

In [131]:
```python
from sklearn.model_selection import RandomizedSearchCV
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 200, num = 2)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
print(random_grid)
```

```
{'n_estimators': [10, 200], 'max_features': ['auto', 'sqrt'], 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, No
ne], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'bootstrap': [True, False]}
```

In [*]:
```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestRegressor()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,
                               n_iter = 10, cv = 3, verbose=2, random_state=42, n_jobs = -1)
# Fit the random search model
rf_random.fit(X_train,y_train)
```

In [ ]:
```
y_pred=rf_random.predict(X_test)
y_pred_train=rf_random.predict(X_train)
```

Checking Performance of the Model

In [ ]:
```
from sklearn.metrics import r2_score

print('Random Forest Train r2_score',r2_score(y_train,y_pred_train))
print('Random Forest Test r2_score',r2_score(y_test,y_pred))
```

In [ ]:
```
rf_random.predict([[5,3.0,1800.0,3,2,12.899222]])
```

Save Model

In [ ]:
```
import pickle
```

In [ ]:
```
# Saving the model
#pickle.dump(rf_random, open('rf_rand_model.pkl','wb'))
```

In [ ]:
```
model = pickle.load(open('rf_rand_model.pkl','rb'))
print(model.predict([[5,1.0,470.0,1,2,11.225257]]))
```

In [ ]:
```
```