Lab 3 - Finding Optimal House Rent

Submitted By
Name: Harsha KG
Register Number: 19112005
Class: **5 BSc Data Science**

Lab Overview

Objectives

- Train and Test through Different Regression Models for Better Pricdiction.
- .Help the client in finding a suitable residence in Lavasa

Problem Definition

THE BACKGROUND

You are working in the Lavasa Campus, helping our Public Relations Team to find houses for people who are in search for one. You currently have the dataset that shows the Building Type, Location, Size, Building Area, No of Bathrooms, No of Balconies and how many people stayed in the building in the academic year 2020-21. This dataset also shows you the rent that is demanded by the current building owners.

THE NEW PROFESSOR IN COLLEGE

Prof Naived George Eapen is coming to Lavasa on this upcoming Friday, and he has contacted you to get an idea about the rent of the accomodation facilities as available there. You, being an amazing analyst, is very confident that you will be able to help him with the requirements that he has.

Below are some suggestions that Prof Naived has in mind:

1. 1 BHK with 2 Baths in Portofino Street
2. A Fully-Furnished 2 BHK Room, in School Street
3. A Super-Furnished Single Room, anywhere in Lavasa
4. A Fully-Furnished 2 BHK Room with two balconies

He does not say what he has in mind with respect to other conditions, and he belives that you can provide case-wise results by populating other variable values.

THE IRRITATING JUNIORS

You started by considering 70% of Data for Training and 30% for testing. Your juniors, seeing you do the training and testing for predicting house values, asks you few doubts.

What happens if you use different Random States for splitting the dataset before the training process?

Is there any improvement in the Reduction of Training and Generalization Errors if you increase the percentage of Training to 75%, 80% and 90%?

What are the different Error Measures (Evaluation Metrics) in relation to Linear Regression? How much do you get in the above cases?

During LinearRegression() process, what is the impact of giving TRUE/FALSE as the value for Normalize Parameter?

Additional Exploration: Try to use Lasso Regression (L1) and Ridge Regression (L2) and ElasticNet (Hybrid Model) and note on the variation in results

Approach

Imported the Dataset using required libraries using python and did some preprocessing techniques before exploration to make the datset into standard format and then did some EDA. After that datset is splitted into training and testing into several ratio and compared the regression model with different ratio.At the end,help the client with insights for choosing best rental property.

Sections

1. Lab Overview
2. Dataset Overview
3. EDA
4. About Different Regression Models
5. Implementation and Evaluation of Different Regression Models
6. Conclusion

References

1. https://www.kaggle.com/c/house-prices-advanced-regression-techniques (https://www.kaggle.com/c/house-prices-advanced-regression-techniques)

About The Dataset

This dataset also shows you the rent that is demanded by the current building owners.

1. Building_Type- Is it a fully/semi/Un furnished Single Room, Flat, or Villa ?
2. Location- Correct location in which Building is located in Lavasa
3. Size-specify no of bedroom.
4. Areasqft- Tells us Whether they are supported with some loan, scholarship, sponsor, etc.
5. No of Bath- Specify no of bathrooms
6. No of Balcony- count of no of balconies
7. RentPerMonth-Rent paid per month

Import the Libraries

In [13]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import hvplot.pandas
import random
from sklearn.linear_model import Ridge


from sklearn.model_selection import train_test_split
```

Loading and Pre-Processing

In [14]:
```
df=pd.read_csv("HousePricePrediction.csv")
```

In [15]:
```
df.head(34)
```

Out[15]:

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMonth
0	Minimum Budget Rooms	Portofino H	1 BHK	400.0	1	1	1	1100.0
1	Minimum Budget Rooms	Portofino H	1 BHK	450.0	1	1	1	1100.0
2	Minimum Budget Rooms	School Street	1 BHK	530.0	1	1	0	1166.0
3	Minimum Budget Rooms	Portofino B	1 BHK	400.0	1	1	0	1400.0
4	Minimum Budget Rooms	School Street	2 BHK	460.0	1	1	0	1500.0
5	Minimum Budget Rooms	Portofino A	1 BHK	600.0	1	1	1	1500.0
6	Semi Furnished Single Room	School Street	1 BHK	654.0	1	1	0	1513.5
7	Semi Furnished Single Room	School Street	1 BHK	645.0	1	1	1	1645.0
8	Semi Furnished Single Room	School Street	1 BHK	645.0	1	1	1	1645.0
9	Semi Furnished Single Room	Clubview Road	2 BHK	880.0	1	1	1	1650.0
10	Minimum Budget Rooms	School Street	2 BHK	650.0	1	1	1	1700.0
11	Minimum Budget Rooms	Portofino H	1 BHK	686.0	1	1	1	1700.0
12	Minimum Budget Rooms	Portofino C	1 BHK	666.0	1	1	0	1753.5
13	Minimum Budget Rooms	Portofino A	1 BHK	665.0	1	1	0	1841.0
14	Minimum Budget Rooms	Portofino A	1 BHK	386.0	1	1	0	1850.0
15	Minimum Budget Rooms	Portofino D	1 BHK	416.0	1	1	1	1850.0
16	Minimum Budget Rooms	School Street	1 BHK	1200.0	1	1	0	2000.0
17	Semi Furnished Single Room	Portofino H	1 BHK	530.0	1	1	1	2000.0
18	Minimum Budget Rooms	School Street	2 BHK	800.0	1	1	1	2000.0
19	Semi Furnished Single Room	Portofino B	1 BHK	425.0	1	1	0	2003.0
20	Semi Furnished Single Room	Portofino A	1 BHK	525.0	1	1	0	2153.0
21	Semi Furnished Single Room	Portofino A	2 BHK	460.0	1	1	0	2200.0
22	Semi Furnished Single Room	Portofino H	2 BHK	656.0	2	1	1	2200.0
23	Semi Furnished Single Room	School Street	2 BHK	805.0	2	1	1	2214.0
24	Semi Furnished Single Room	Portofino C	2 BHK	900.0	2	1	2	2250.0
25	Minimum Budget Rooms	Starter Homes	1 BHK	500.0	1	1	1	2300.0
26	Minimum Budget Rooms	Clubview Road	2 BHK	400.0	3	1	3	2300.0

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMonth
27	Minimum Budget Rooms	Clubview Road	2 BHK	660.0	1	1	1	2310.0
28	Minimum Budget Rooms	Portofino A	2 BHK	810.0	2	1	2	2450.0
29	Semi Furnished Single Room	Clubview Road	1 BHK	469.0	1	1	0	2500.0
30	Semi Furnished Single Room	Portofino H	2 BHK	656.0	2	1	1	2500.0
31	Semi Furnished Single Room	Clubview Road	2 BHK	891.0	2	1	1	2500.0
32	Semi Furnished Single Room	Clubview Road	2 BHK	1000.0	2	1	1	2500.0
33	Minimum Budget Rooms	Portofino A	2 BHK	1000.0	2	1	1	2500.0

In [16]:
```
df.shape
```

Out[16]:
```
(1000, 8)
```

In [17]:
```
df.info()#to get info about filled values in the dataframe for every column
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   BuildingType  1000 non-null   object 
 1   Location      1000 non-null   object 
 2   Size          1000 non-null   object 
 3   AreaSqFt      1000 non-null   float64
 4   NoOfBath      1000 non-null   int64  
 5   NoOfPeople    1000 non-null   int64  
 6   NoOfBalcony   1000 non-null   int64  
 7   RentPerMonth  1000 non-null   float64
dtypes: float64(2), int64(3), object(3)
memory usage: 62.6+ KB
```

In [18]:
```
df.columns
```

Out[18]:
```
Index(['BuildingType', 'Location', 'Size', 'AreaSqFt', 'NoOfBath',
       'NoOfPeople', 'NoOfBalcony', 'RentPerMonth'],
      dtype='object')
```

In [19]:
```
df.isnull().sum()
```

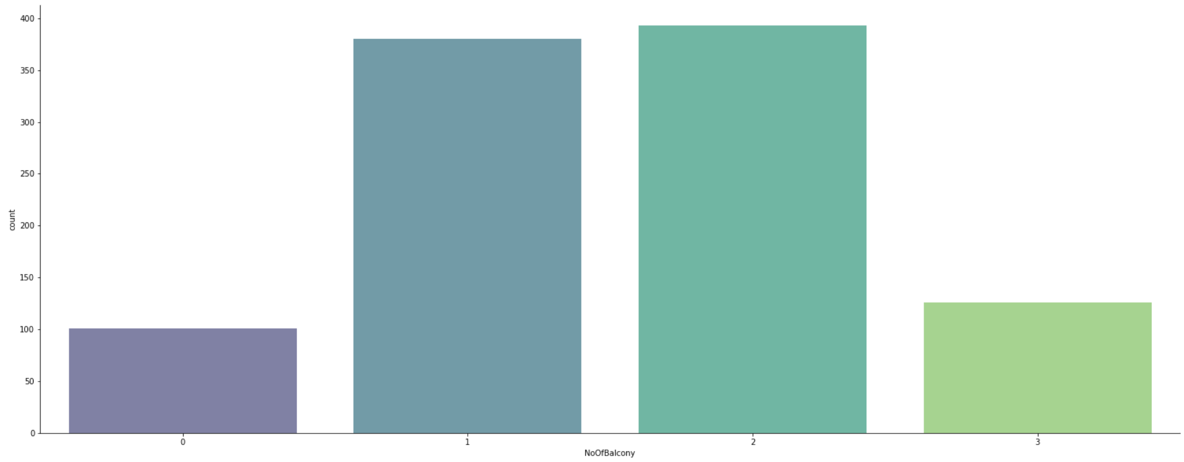
Out[19]:
```
BuildingType    0
Location        0
Size            0
AreaSqFt        0
NoOfBath        0
NoOfPeople      0
NoOfBalcony     0
RentPerMonth    0
dtype: int64
```

In [20]:
```
df.describe(include='all')
```

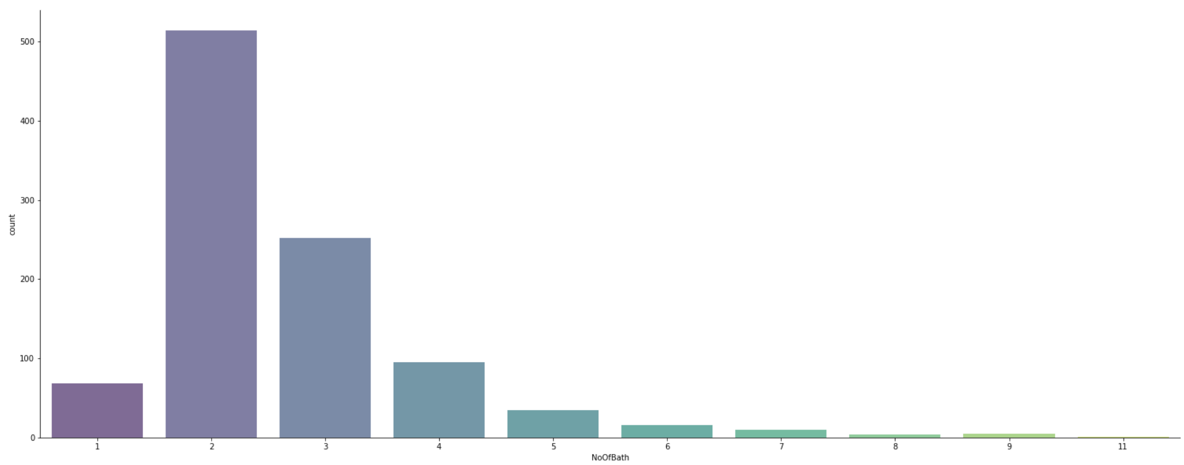
Out[20]:

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMon
count	1000	1000	1000	1000.000000	1000.000000	1000.000000	1000.000000	1000.0000
unique	10	11	10	NaN	NaN	NaN	NaN	Na
top	Semi Furnished Single Room	Clubview Road	2 BHK	NaN	NaN	NaN	NaN	Na
freq	274	213	429	NaN	NaN	NaN	NaN	Na
mean	NaN	NaN	NaN	1548.270010	2.661000	2.168000	1.544000	10476.6335
std	NaN	NaN	NaN	1345.141175	1.247251	0.959529	0.838312	10509.5089
min	NaN	NaN	NaN	375.000000	1.000000	1.000000	0.000000	1100.0000
25%	NaN	NaN	NaN	1090.000000	2.000000	2.000000	1.000000	4890.5000
50%	NaN	NaN	NaN	1270.000000	2.000000	2.000000	2.000000	7000.0000
75%	NaN	NaN	NaN	1664.250000	3.000000	2.000000	2.000000	11925.0000
max	NaN	NaN	NaN	35000.000000	11.000000	6.000000	3.000000	96000.0000

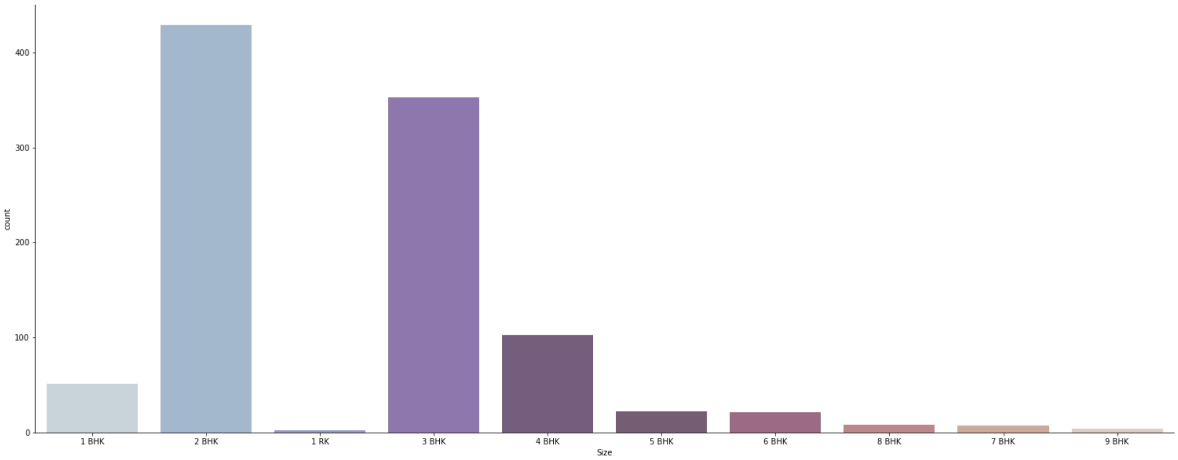
In [21]:
```
plt.figure(figsize=[26,10])
sns.countplot(x='NoOfBalcony',palette='viridis',alpha=0.7,data=df)
sns.despine()
```



In [22]:
```
plt.figure(figsize=[26,10])
sns.countplot(x='NoOfBath',palette='viridis',alpha=0.7,data=df)
sns.despine()
```

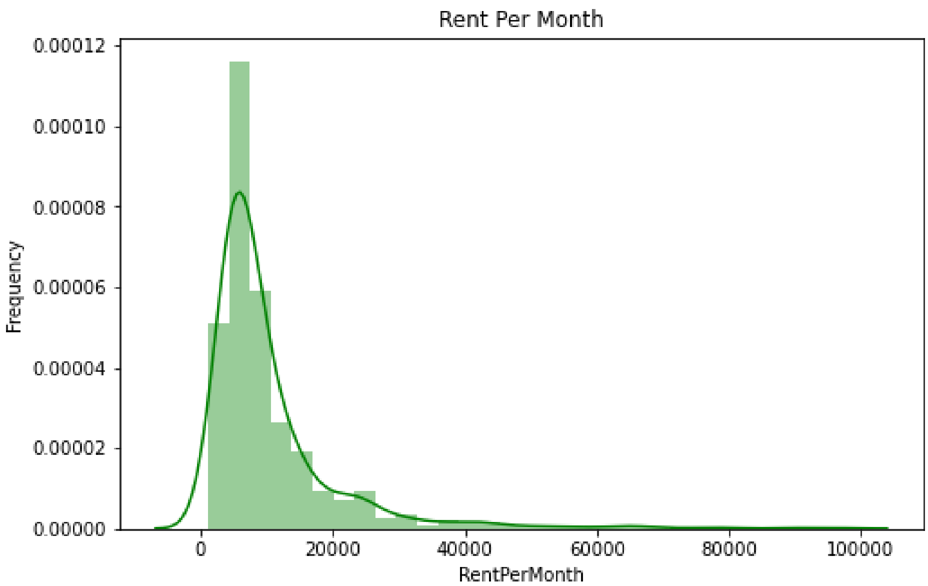


In [23]:
```
plt.figure(figsize=[26,10])
sns.countplot(x='Size',palette='twilight',alpha=0.7,data=df)
sns.despine()
```



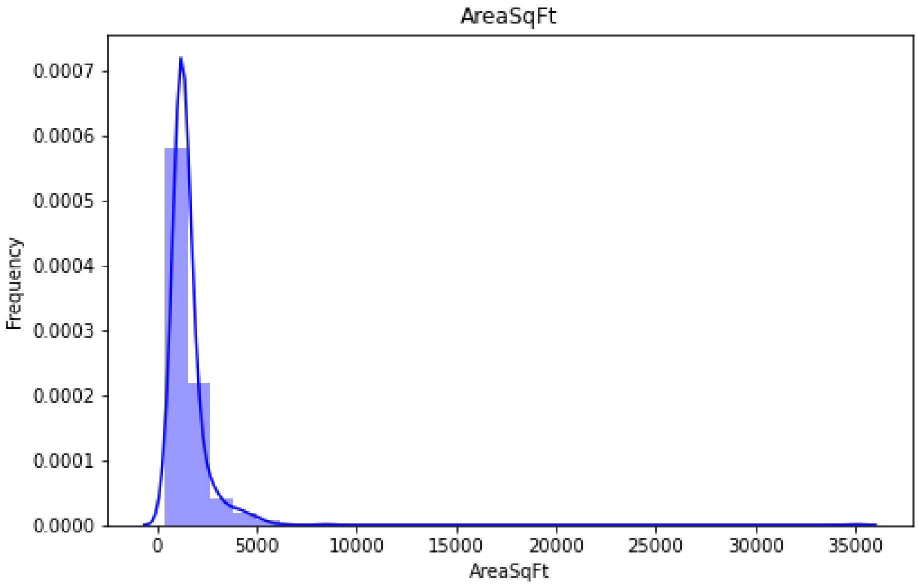
In [24]:
```
plt.figure(figsize=(8,5))
sns.distplot(df['RentPerMonth'], kde = True, color='g', bins = 30)
plt.ylabel('Frequency')
plt.title('Rent Per Month')
plt.show()
```

```
C:\Users\HP\anaconda3\anacondaorginal\lib\site-packages\seaborn\distributions.py:261
9: FutureWarning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

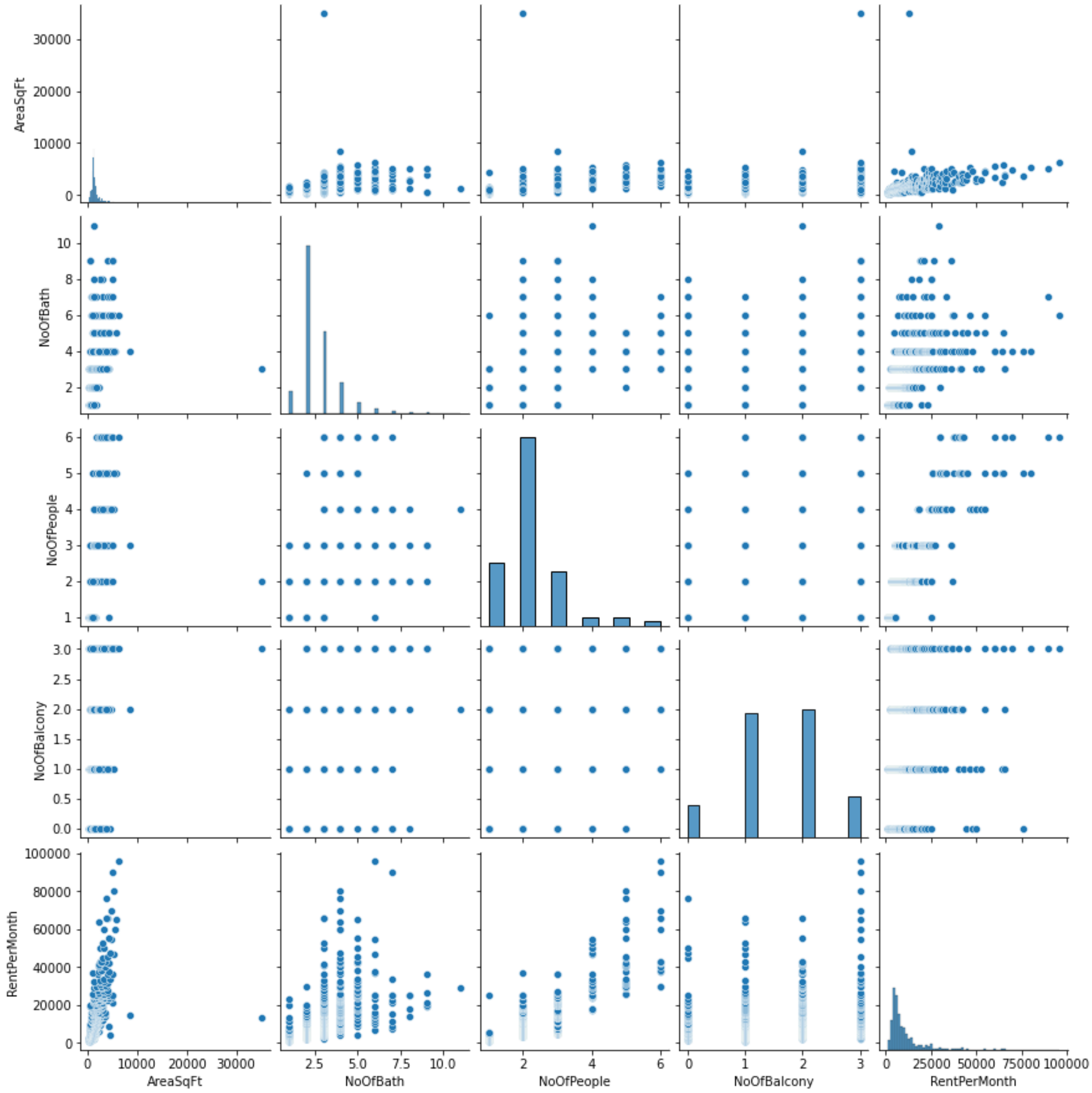


In [25]:
```
plt.figure(figsize=(8,5))
sns.distplot(df['AreaSqFt'], kde = True, color='b', bins = 30)
plt.ylabel('Frequency')
plt.title('AreaSqFt')
plt.show()
```

```
C:\Users\HP\anaconda3\anacondaorginal\lib\site-packages\seaborn\distributions.py:261
9: FutureWarning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

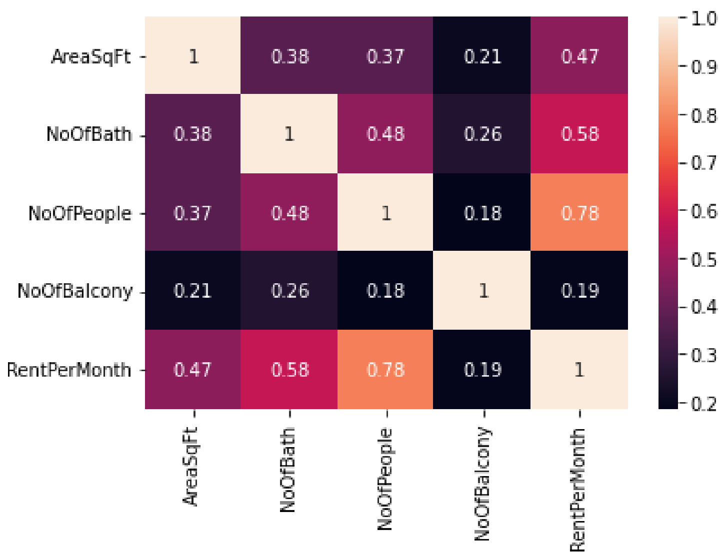


In [26]:
```
sns.pairplot(data=df,palette='Blues')
plt.show()
```



In [33]:
```
sns.heatmap(df.corr(), annot=True)
```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x12de897fe80>



Split it into Features and Target

In [34]:
```
X = df[['Size','AreaSqFt','NoOfBath','NoOfPeople','NoOfBalcony']]
y = df['RentPerMonth']
```

In [35]:
```
df['Size'] = df['Size'].str.replace('BHK','')
df['Size'] = df['Size'].str.replace('RK','')
```

Training and Testing

In [36]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .30, random_sta
te = 8)
```

In [37]:
```
from sklearn import metrics
from sklearn.model_selection import cross_val_score

def cross_val(model):
    pred = cross_val_score(model, X, y, cv=10)
    return pred.mean()

def print_evaluate(true, predicted):
    mae = metrics.mean_absolute_error(true, predicted)
    mse = metrics.mean_squared_error(true, predicted)
    rmse = np.sqrt(metrics.mean_squared_error(true, predicted))
    r2_square = metrics.r2_score(true, predicted)
    print('MAE:', mae)
    print('MSE:', mse)
    print('RMSE:', rmse)
    print('R2 Square', r2_square)
    print('__________________________________')

def evaluate(true, predicted):
    mae = metrics.mean_absolute_error(true, predicted)
    mse = metrics.mean_squared_error(true, predicted)
    rmse = np.sqrt(metrics.mean_squared_error(true, predicted))
    r2_square = metrics.r2_score(true, predicted)
    return mae, mse, rmse, r2_square
```

In [38]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

pipeline = Pipeline([
    ('std_scalar', StandardScaler())
])

X_train = pipeline.fit_transform(X_train)
X_test = pipeline.transform(X_test)
```

In [39]:
```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression(normalize=True)
lin_reg.fit(X_train,y_train)
```

Out[39]:
```
LinearRegression(normalize=True)
```

In [40]:
```
# print the intercept
print(lin_reg.intercept_)
```

```
10563.420714285716
```

In [41]:
```
coeff_df = pd.DataFrame(lin_reg.coef_, X.columns, columns=['Coefficient'])
coeff_df
```

Out[41]:

	Coefficient
Size	756.489445
AreaSqFt	1356.389028
NoOfBath	1935.611925
NoOfPeople	6802.630708
NoOfBalcony	-81.066590

In [42]:
```
pred = lin_reg.predict(X_test)
```

In [43]:
```
test_pred = lin_reg.predict(X_test)
train_pred = lin_reg.predict(X_train)

print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)
print('Train set evaluation:\n_____________________________________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:
_____________________________________
MAE: 4029.6462125390954
MSE: 33731914.40446294
RMSE: 5807.918250497586
R2 Square 0.6599134411967313
__________________________________
Train set evaluation:
_____________________________________
MAE: 3955.100627820552
MSE: 35559217.843001105
RMSE: 5963.1550242301355
R2 Square 0.6910422404289258
__________________________________
```

In [44]:
```
test_pred = lin_reg.predict(X_test)
train_pred = lin_reg.predict(X_train)

print('Test set evaluation:\n_____________')
print_evaluate(y_test, test_pred)
print('Train set evaluation:\n_____________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:
_____________
MAE: 4029.6462125390954
MSE: 33731914.40446294
RMSE: 5807.918250497586
R2 Square 0.6599134411967313
__________________________________
Train set evaluation:
_____________
MAE: 3955.100627820552
MSE: 35559217.843001105
RMSE: 5963.1550242301355
R2 Square 0.6910422404289258
__________________________________
```

In [45]:
```
pd.DataFrame({'True Values': y_test, 'Predicted Values': pred}).hvplot.scatter(x='True Values', y='Predicted Values')
```

Out[45]:

In [46]:
```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression(normalize=False)
lin_reg.fit(X_train,y_train)
```

Out[46]:
```
LinearRegression()
```

print the intercept

In [47]:
```
print(lin_reg.intercept_)
```

```
10563.420714285716
```

In [48]:
```
coeff_df = pd.DataFrame(lin_reg.coef_, X.columns, columns=['Coefficient'])
coeff_df
```

Out[48]:

	Coefficient
Size	756.489445
AreaSqFt	1356.389028
NoOfBath	1935.611925
NoOfPeople	6802.630708
NoOfBalcony	-81.066590

In [49]:
```
est_pred = lin_reg.predict(X_test)
train_pred = lin_reg.predict(X_train)

print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)
print('Train set evaluation:\n_____________________________________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:
_____________________________________
MAE: 4029.6462125390954
MSE: 33731914.40446294
RMSE: 5807.918250497586
R2 Square 0.6599134411967313
__________________________________
Train set evaluation:
_____________________________________
MAE: 3955.100627820551
MSE: 35559217.8430011
RMSE: 5963.155024230135
R2 Square 0.6910422404289258
__________________________________
```

In [50]:
```
results_df = pd.DataFrame(data=[["Linear Regression", *evaluate(y_test, test_pred) , 
cross_val(LinearRegression())]], 
                          columns=['Model', 'MAE', 'MSE', 'RMSE', 'R2 Square', "Cross 
Validation"])
results_df
```

Out[50]:

	Model	MAE	MSE	RMSE	R2 Square	Cross Validation
0	Linear Regression	4029.646213	3.373191e+07	5807.91825	0.659913	-189.153433

Ridge Regression

In [51]:
```
model = Ridge(alpha=100, solver='cholesky', tol=0.0001, random_state=42)
model.fit(X_train, y_train)
pred = model.predict(X_test)

test_pred = model.predict(X_test)
train_pred = model.predict(X_train)

print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)
print('====================================')
print('Train set evaluation:\n_____________________________________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:
_____________________________________
MAE: 3766.295286504833
MSE: 32688749.075766582
RMSE: 5717.407548510652
R2 Square 0.67043067726716
__________________________________
====================================
Train set evaluation:
_____________________________________
MAE: 3774.4747576607556
MSE: 36322363.657398194
RMSE: 6026.803767951815
R2 Square 0.6844116159286016
__________________________________
```

In [52]:
```
results_df_2 = pd.DataFrame(data=[["Ridge Regression", *evaluate(y_test, test_pred) , cross_val(Ridge())]],
                            columns=['Model', 'MAE', 'MSE', 'RMSE', 'R2 Square', "Cross Validation"])
results_df = results_df.append(results_df_2, ignore_index=True)
results_df
```

Out[52]:

	Model	MAE	MSE	RMSE	R2 Square	Cross Validation
0	Linear Regression	4029.646213	3.373191e+07	5807.918250	0.659913	-189.153433
1	Ridge Regression	3766.295287	3.268875e+07	5717.407549	0.670431	-188.913057

LASSO Regression

In [53]:
```
from sklearn.linear_model import Lasso

model = Lasso(alpha=0.1,
              precompute=True,
#               warm_start=True,
              positive=True,
              selection='random',
              random_state=42)
model.fit(X_train, y_train)

test_pred = model.predict(X_test)
train_pred = model.predict(X_train)

print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)
print('====================================')
print('Train set evaluation:\n_____________________________________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:
_____________________________________
MAE: 4037.454430671962
MSE: 33749449.18585603
RMSE: 5809.427612584223
R2 Square 0.6597366548041229
__________________________________
====================================
Train set evaluation:
_____________________________________
MAE: 3956.5764794264705
MSE: 35565052.081629254
RMSE: 5963.644194754517
R2 Square 0.6909915494012627
__________________________________
```

In [54]:
```
results_df_2 = pd.DataFrame(data=[["Lasso Regression", *evaluate(y_test, test_pred) , cross_val(Lasso())]],
                            columns=['Model', 'MAE', 'MSE', 'RMSE', 'R2 Square', "Cross Validation"])
results_df = results_df.append(results_df_2, ignore_index=True)
results_df
```

Out[54]:

	Model	MAE	MSE	RMSE	R2 Square	Cross Validation
0	Linear Regression	4029.646213	3.373191e+07	5807.918250	0.659913	-189.153433
1	Ridge Regression	3766.295287	3.268875e+07	5717.407549	0.670431	-188.913057
2	Lasso Regression	4037.454431	3.374945e+07	5809.427613	0.659737	-189.134286

Elastic Net

In [55]:
```
from sklearn.linear_model import ElasticNet

model = ElasticNet(alpha=0.1, l1_ratio=0.9, selection='random', random_state=42)
model.fit(X_train, y_train)

test_pred = model.predict(X_test)
train_pred = model.predict(X_train)

print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)
print('====================================')
print('Train set evaluation:\n_____________________________________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:
_____________________________________
MAE: 4007.27283286183
MSE: 33602616.59927238
RMSE: 5796.776397211848
R2 Square 0.6612170270264837
__________________________________
====================================
Train set evaluation:
_____________________________________
MAE: 3938.2746890185163
MSE: 35564427.825836554
RMSE: 5963.59185607437
R2 Square 0.6909969732739689
__________________________________
```

In [56]:
```
results_df_2 = pd.DataFrame(data=[["Elastic Net Regression", *evaluate(y_test, test_p
red) , cross_val(ElasticNet())]],
                            columns=['Model', 'MAE', 'MSE', 'RMSE', 'R2 Square', "Cro
ss Validation"])
results_df = results_df.append(results_df_2, ignore_index=True)
results_df
```

Out[56]:

	Model	MAE	MSE	RMSE	R2 Square	Cross Validation
0	Linear Regression	4029.646213	3.373191e+07	5807.918250	0.659913	-189.153433
1	Ridge Regression	3766.295287	3.268875e+07	5717.407549	0.670431	-188.913057
2	Lasso Regression	4037.454431	3.374945e+07	5809.427613	0.659737	-189.134286
3	Elastic Net Regression	4007.272833	3.360262e+07	5796.776397	0.661217	-152.868602

In [ ]:

Polynomial Regression

In [57]:
```
from sklearn.preprocessing import PolynomialFeatures

poly_reg = PolynomialFeatures(degree=2)

X_train_2_d = poly_reg.fit_transform(X_train)
X_test_2_d = poly_reg.transform(X_test)

lin_reg = LinearRegression(normalize=True)
lin_reg.fit(X_train_2_d,y_train)

test_pred = lin_reg.predict(X_test_2_d)
train_pred = lin_reg.predict(X_train_2_d)

print('Test set evaluation:\n_____________________________________')
print_evaluate(y_test, test_pred)
print('====================================')
print('Train set evaluation:\n_____________________________________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:
_____________________________________
MAE: 2455.3319656714907
MSE: 19140141.004973438
RMSE: 4374.944685933005
R2 Square 0.8070283052618701
__________________________________
====================================
Train set evaluation:
_____________________________________
MAE: 2671.5042294865675
MSE: 17881774.27004752
RMSE: 4228.684697402671
R2 Square 0.8446334522873383
__________________________________
```

In [58]:
```
results_df_2 = pd.DataFrame(data=[["Polynomail Regression", *evaluate(y_test, test_pred), 0]],
                            columns=['Model', 'MAE', 'MSE', 'RMSE', 'R2 Square', 'Cross Validation'])
results_df = results_df.append(results_df_2, ignore_index=True)
results_df
```

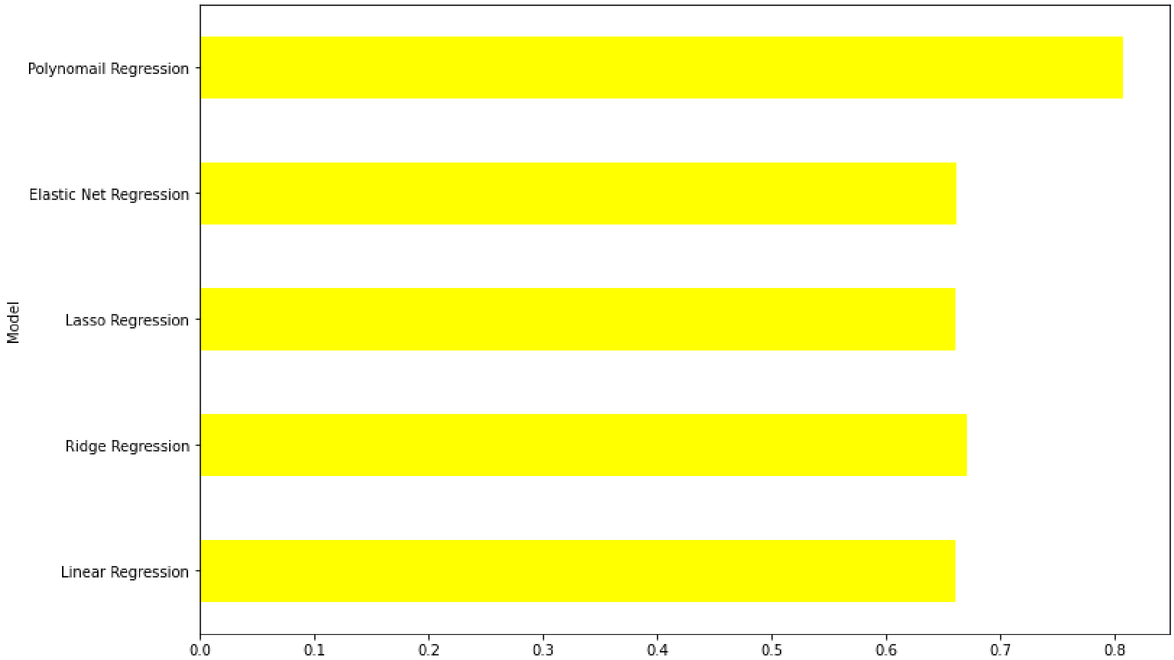
Out[58]:

	Model	MAE	MSE	RMSE	R2 Square	Cross Validation
0	Linear Regression	4029.646213	3.373191e+07	5807.918250	0.659913	-189.153433
1	Ridge Regression	3766.295287	3.268875e+07	5717.407549	0.670431	-188.913057
2	Lasso Regression	4037.454431	3.374945e+07	5809.427613	0.659737	-189.134286
3	Elastic Net Regression	4007.272833	3.360262e+07	5796.776397	0.661217	-152.868602
4	Polynomail Regression	2455.331966	1.914014e+07	4374.944686	0.807028	0.000000

Models Comparison

In [59]:
```
results_df.set_index('Model', inplace=True)
results_df['R2 Square'].plot(kind='barh',color='yellow',figsize=(12, 8))
```

Out[59]: <matplotlib.axes._subplots.AxesSubplot at 0x12de8d5c850>



Regression Model Implementation with different implemenation and Evaluation with different split (75:25,80:20,90:10)

In [60]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .25, random_sta
te = 8)
```

In [61]:
```
from sklearn import metrics
from sklearn.model_selection import cross_val_score

def cross_val(model):
    pred = cross_val_score(model, X, y, cv=10)
    return pred.mean()

def print_evaluate(true, predicted):
    mae = metrics.mean_absolute_error(true, predicted)
    mse = metrics.mean_squared_error(true, predicted)
    rmse = np.sqrt(metrics.mean_squared_error(true, predicted))
    r2_square = metrics.r2_score(true, predicted)
    print('MAE:', mae)
    print('MSE:', mse)
    print('RMSE:', rmse)
    print('R2 Square', r2_square)
    print('____________')

def evaluate(true, predicted):
    mae = metrics.mean_absolute_error(true, predicted)
    mse = metrics.mean_squared_error(true, predicted)
    rmse = np.sqrt(metrics.mean_squared_error(true, predicted))
    r2_square = metrics.r2_score(true, predicted)
    return mae, mse, rmse, r2_square
```

In [62]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

pipeline = Pipeline([
    ('std_scalar', StandardScaler())
])

X_train = pipeline.fit_transform(X_train)
X_test = pipeline.transform(X_test)
```

In [63]:
```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression(normalize=True)
lin_reg.fit(X_train,y_train)
```

Out[63]:
```
LinearRegression(normalize=True)
```

In [64]:
```
# print the intercept
print(lin_reg.intercept_)
```

```
10509.06733333333
```

In [65]:
```
coeff_df = pd.DataFrame(lin_reg.coef_, X.columns, columns=['Coefficient'])
coeff_df
```

Out[65]:

	Coefficient
Size	574.593285
AreaSqFt	1365.197525
NoOfBath	1971.485842
NoOfPeople	6602.312514
NoOfBalcony	31.825087

In [66]:
```
pred = lin_reg.predict(X_test)
```

In [67]:
```
test_pred = lin_reg.predict(X_test)
train_pred = lin_reg.predict(X_train)

print('Test set evaluation:\n_____________')
print_evaluate(y_test, test_pred)
print('Train set evaluation:\n_____________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:
_____________
MAE: 4059.8377815782783
MSE: 35271984.62141491
RMSE: 5939.0221940496995
R2 Square 0.6825148862196837
_____________
Train set evaluation:
_____________
MAE: 3880.5604165913487
MSE: 34656018.09829068
RMSE: 5886.936223392494
R2 Square 0.6851805639249118
____________
```

In [68]:
```
pd.DataFrame({'True Values': y_test, 'Predicted Values': pred}).hvplot.scatter(x='True Values', y='Predicted Values')
```

Out[68]:

In [69]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .10, random_sta
te = 8)
```

In [70]:
```
from sklearn import metrics
from sklearn.model_selection import cross_val_score

def cross_val(model):
    pred = cross_val_score(model, X, y, cv=10)
    return pred.mean()

def print_evaluate(true, predicted):
    mae = metrics.mean_absolute_error(true, predicted)
    mse = metrics.mean_squared_error(true, predicted)
    rmse = np.sqrt(metrics.mean_squared_error(true, predicted))
    r2_square = metrics.r2_score(true, predicted)
    print('MAE:', mae)
    print('MSE:', mse)
    print('RMSE:', rmse)
    print('R2 Square', r2_square)
    print('____________')

def evaluate(true, predicted):
    mae = metrics.mean_absolute_error(true, predicted)
    mse = metrics.mean_squared_error(true, predicted)
    rmse = np.sqrt(metrics.mean_squared_error(true, predicted))
    r2_square = metrics.r2_score(true, predicted)
    return mae, mse, rmse, r2_square
```

In [71]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

pipeline = Pipeline([
    ('std_scalar', StandardScaler())
])

X_train = pipeline.fit_transform(X_train)
X_test = pipeline.transform(X_test)
```

In [72]:
```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression(normalize=True)
lin_reg.fit(X_train,y_train)
```

Out[72]:
```
LinearRegression(normalize=True)
```

In [73]:
```
# print the intercept
print(lin_reg.intercept_)
```

```
10370.962222222222
```

In [74]:
```
coeff_df = pd.DataFrame(lin_reg.coef_, X.columns, columns=['Coefficient'])
coeff_df
```

Out[74]:

	Coefficient
Size	315.445802
AreaSqFt	1454.682579
NoOfBath	2181.731306
NoOfPeople	6386.514897
NoOfBalcony	-178.653465

In [75]:
```
pred = lin_reg.predict(X_test)
```

In [76]:
```
test_pred = lin_reg.predict(X_test)
train_pred = lin_reg.predict(X_train)

print('Test set evaluation:\n_____________')
print_evaluate(y_test, test_pred)
print('Train set evaluation:\n_____________')
print_evaluate(y_train, train_pred)
```

```
Test set evaluation:
_____________
MAE: 4454.41536035513
MSE: 50399587.55818312
RMSE: 7099.266691580414
R2 Square 0.6816159323173014
____________
Train set evaluation:
_____________
MAE: 3725.496231389967
MSE: 32852799.62832211
RMSE: 5731.736179232442
R2 Square 0.6868146842461976
____________
```

Helping Prof Naived to find accomodation

In [77]:
```
BuildingTypes = df.BuildingType.unique()
for building in BuildingTypes:
    print(building)
```

```
Minimum Budget Rooms
Semi Furnished Single Room
Semi Furnished Flat
Fully Furnished Single Room
Super Furnished Single Room
Semi Furnished Villa
Fully Furnished Flat
Super Furnished Flat
Fully Furnished Villa
Super Furnished Villa
```

In [78]:
```
import random
New_df = pd.DataFrame(columns=['BuildingType', 'Location', 'Size', 'AreaSqFt', 'NoOfB
ath', 'NoOfPeople', 'NoOfBalcony'])
New_df
```

Out[78]:

BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony

In [79]:
```
BuildingTypes = df.BuildingType.unique()
for Building in BuildingTypes:
    print(Building)
```

```
Minimum Budget Rooms
Semi Furnished Single Room
Semi Furnished Flat
Fully Furnished Single Room
Super Furnished Single Room
Semi Furnished Villa
Fully Furnished Flat
Super Furnished Flat
Fully Furnished Villa
Super Furnished Villa
```

In [80]:
```
NoOfBath = [1, 2]
NoOfPeople = [1, 2]
NoOfBalcony = [0, 1, 2]
Location = ["Portofino","School Street"]
Size = ["1 BHK", "2 BHK"]
```

In [81]:
```
for Building in BuildingTypes:
    for Locate in Location:
        for size in Size:
            for Bathroom in NoOfBath:
                for People in NoOfPeople:
                    for Balcony in NoOfBalcony:



                        Build = {}
                        Build['BuildingType'] = Building
                        Build['Location'] = Locate
                        Build['Size'] = size
                        Build['AreaSqFt'] = random.randint(350, 600)
                        Build['NoOfBath'] = Bathroom
                        Build['NoOfPeople'] = People
                        Build['NoOfBalcony'] = Balcony


                        New_df = New_df.append(Build, ignore_index = True)
```

In [82]:
```
New_df.head(15)
```

Out[82]:

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony
0	Minimum Budget Rooms	Portofino	1 BHK	537	1	1	0
1	Minimum Budget Rooms	Portofino	1 BHK	434	1	1	1
2	Minimum Budget Rooms	Portofino	1 BHK	517	1	1	2
3	Minimum Budget Rooms	Portofino	1 BHK	477	1	2	0
4	Minimum Budget Rooms	Portofino	1 BHK	370	1	2	1
5	Minimum Budget Rooms	Portofino	1 BHK	452	1	2	2
6	Minimum Budget Rooms	Portofino	1 BHK	396	2	1	0
7	Minimum Budget Rooms	Portofino	1 BHK	382	2	1	1
8	Minimum Budget Rooms	Portofino	1 BHK	475	2	1	2
9	Minimum Budget Rooms	Portofino	1 BHK	511	2	2	0
10	Minimum Budget Rooms	Portofino	1 BHK	376	2	2	1
11	Minimum Budget Rooms	Portofino	1 BHK	403	2	2	2
12	Minimum Budget Rooms	Portofino	2 BHK	445	1	1	0
13	Minimum Budget Rooms	Portofino	2 BHK	403	1	1	1
14	Minimum Budget Rooms	Portofino	2 BHK	366	1	1	2

In [83]:
```
New_df.to_csv("HousePrices .csv")
```

In [84]:
```
N_New_df=pd.read_csv("HousePrices .csv")
```

In [85]:
```
N_New_df['RentPerMonth']=(10308.4978+(1485.9016*N_New_df['AreaSqFt'])+(2811.5748*N_New_df['NoOfBath'])+(53934.4292*N_New_df['NoOfPeople'])+(-252.6471*N_New_df['NoOfBalcony']))
```

In [86]:
```
N_New_df
```

Out[86]:

	Unnamed: 0	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMc
0	0	Minimum Budget Rooms	Portofino	1 BHK	537	1	1	0	864983.6
1	1	Minimum Budget Rooms	Portofino	1 BHK	434	1	1	1	711683.1
2	2	Minimum Budget Rooms	Portofino	1 BHK	517	1	1	2	834760.3
3	3	Minimum Budget Rooms	Portofino	1 BHK	477	1	2	0	829763.9
4	4	Minimum Budget Rooms	Portofino	1 BHK	370	1	2	1	670519.8
...	...	...	...	...	...	...	...	...	
475	475	Super Furnished Villa	School Street	2 BHK	485	2	1	1	790275.7
476	476	Super Furnished Villa	School Street	2 BHK	463	2	1	2	757333.2
477	477	Super Furnished Villa	School Street	2 BHK	412	2	2	0	735991.9
478	478	Super Furnished Villa	School Street	2 BHK	472	2	2	1	824893.4
479	479	Super Furnished Villa	School Street	2 BHK	379	2	2	2	686451.9

480 rows × 9 columns

In [87]:
```
def main():
    while True:
        C = int(input("Enter your choice:"))
        if C == 0:
            print("Thank you")
            break
        if C == 1:
            print("Minimum Budget Rooms\n"
                   "Semi Furnished Single Room\n"
                    "Semi Furnished Flat\n"
                    "Fully Furnished Single Room\n"
                    "Super Furnished Single Room\n"
                    "Semi Furnished Villa\n"
                     "Fully Furnished Flat\n"
                     "Super Furnished Flat\n"
                     "Fully Furnished Villa\n"
                     "Super Furnished Villa\n")
            Struct= str(input('Search the type of building by Type:'))
            RPM = pd.DataFrame(N_New_df[N_New_df['BuildingType'].str.contains(Struct
)])
            return RPM


main()
```

```
Enter your choice:1
Minimum Budget Rooms
Semi Furnished Single Room
Semi Furnished Flat
Fully Furnished Single Room
Super Furnished Single Room
Semi Furnished Villa
Fully Furnished Flat
Super Furnished Flat
Fully Furnished Villa
Super Furnished Villa

Search the type of building by Type:Minimum Budget Rooms
```

Out[87]:

	Unnamed: 0	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMor
0	0	Minimum Budget Rooms	Portofino	1 BHK	537	1	1	0	864983.66
1	1	Minimum Budget Rooms	Portofino	1 BHK	434	1	1	1	711683.14
2	2	Minimum Budget Rooms	Portofino	1 BHK	517	1	1	2	834760.33
3	3	Minimum Budget Rooms	Portofino	1 BHK	477	1	2	0	829763.99
4	4	Minimum Budget Rooms	Portofino	1 BHK	370	1	2	1	670519.87
5	5	Minimum Budget Rooms	Portofino	1 BHK	452	1	2	2	792111.16
6	6	Minimum Budget Rooms	Portofino	1 BHK	396	2	1	0	658283.11
7	7	Minimum Budget Rooms	Portofino	1 BHK	382	2	1	1	637227.84
8	8	Minimum Budget Rooms	Portofino	1 BHK	475	2	1	2	775164.04
9	9	Minimum Budget Rooms	Portofino	1 BHK	511	2	2	0	883096.22
10	10	Minimum Budget Rooms	Portofino	1 BHK	376	2	2	1	682246.86
11	11	Minimum Budget Rooms	Portofino	1 BHK	403	2	2	2	722113.55
12	12	Minimum Budget Rooms	Portofino	2 BHK	445	1	1	0	728280.71
13	13	Minimum Budget Rooms	Portofino	2 BHK	403	1	1	1	665620.19
14	14	Minimum Budget Rooms	Portofino	2 BHK	366	1	1	2	610389.19
15	15	Minimum Budget Rooms	Portofino	2 BHK	421	1	2	0	746553.50
16	16	Minimum Budget Rooms	Portofino	2 BHK	518	1	2	1	890433.31
17	17	Minimum Budget Rooms	Portofino	2 BHK	535	1	2	2	915440.99
18	18	Minimum Budget Rooms	Portofino	2 BHK	430	2	1	0	708803.76
19	19	Minimum Budget Rooms	Portofino	2 BHK	476	2	1	1	776902.59

	Unnamed: 0	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMor
20	20	Minimum Budget Rooms	Portofino	2 BHK	473	2	1	2	772192.23
21	21	Minimum Budget Rooms	Portofino	2 BHK	390	2	2	0	703302.12
22	22	Minimum Budget Rooms	Portofino	2 BHK	461	2	2	1	808548.49
23	23	Minimum Budget Rooms	Portofino	2 BHK	504	2	2	2	872189.61
24	24	Minimum Budget Rooms	School Street	1 BHK	405	1	1	0	668844.64
25	25	Minimum Budget Rooms	School Street	1 BHK	473	1	1	1	769633.31
26	26	Minimum Budget Rooms	School Street	1 BHK	537	1	1	2	864478.36
27	27	Minimum Budget Rooms	School Street	1 BHK	407	1	2	0	725750.88
28	28	Minimum Budget Rooms	School Street	1 BHK	520	1	2	1	893405.11
29	29	Minimum Budget Rooms	School Street	1 BHK	531	1	2	2	909497.38
30	30	Minimum Budget Rooms	School Street	1 BHK	594	2	1	0	952491.62
31	31	Minimum Budget Rooms	School Street	1 BHK	502	2	1	1	815536.03
32	32	Minimum Budget Rooms	School Street	1 BHK	590	2	1	2	946042.72
33	33	Minimum Budget Rooms	School Street	1 BHK	529	2	2	0	909842.45
34	34	Minimum Budget Rooms	School Street	1 BHK	569	2	2	1	969025.86
35	35	Minimum Budget Rooms	School Street	1 BHK	522	2	2	2	898935.84
36	36	Minimum Budget Rooms	School Street	2 BHK	484	1	1	0	786230.87
37	37	Minimum Budget Rooms	School Street	2 BHK	505	1	1	1	817182.16
38	38	Minimum Budget Rooms	School Street	2 BHK	447	1	1	2	730747.22
39	39	Minimum Budget Rooms	School Street	2 BHK	521	1	2	0	895143.66

	Unnamed: 0	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMor
40	40	Minimum Budget Rooms	School Street	2 BHK	546	1	2	1	932038.55
41	41	Minimum Budget Rooms	School Street	2 BHK	531	1	2	2	909497.38
42	42	Minimum Budget Rooms	School Street	2 BHK	381	2	1	0	635994.58
43	43	Minimum Budget Rooms	School Street	2 BHK	545	2	1	1	879429.80
44	44	Minimum Budget Rooms	School Street	2 BHK	464	2	1	2	758819.12
45	45	Minimum Budget Rooms	School Street	2 BHK	471	2	2	0	823660.15
46	46	Minimum Budget Rooms	School Street	2 BHK	352	2	2	1	646585.22
47	47	Minimum Budget Rooms	School Street	2 BHK	417	2	2	2	742916.17

Conclusion

In this lab we have tried to find some Insightful info in house prediction using different kinds of regression model. Implemented various regression model and evaluated the found the one with highest score which will help the public relation team to find a perfect accomadation for the client

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: