MACHINE LEARNING

SUBMITTED BY,

NAME:JAGANNATH V V

REG.NO:21122024

CLASS:2MSCDS

Problem Definition

Common Instructions:

Use Pandas to Import the Dataset

Do the necessary Exploratory Data Analysis

Use the train_test_split method available in SCIKIT to split the dataset into Train Dataset and Test Dataset.

Show the Regression Score, Intercept and other parameters etc in the Output

Use visualizations and plots wherever possible Format the outputs neatly; Do Documentation, Data Set Description, Objectives, Observations, Conclusions etc as you have done in your previous lab

Objective:

- Understand the dataset and features.
- Analyse the dataset.
- Exploring the given insight.

Approach

- Importing all libraries which we needed.
- Perform data preprocessing technique to get balanced structured data.
- Perform statistical data analysis and derive valuable inference.
- perform exploratory data analysis and derive valuable inference.
- Visualizing things with some plot and derive valuable inference.
- Train and test through LinearRegression models for better prediction.

Questions

What are your observations on the Dataset?

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

Note down the errors/losses when the train-test ratio is 50:50, 60:40, 70:30, and 80:20

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

Cases Try to predict the rent of the below houses

1 BHK with 2 Baths in Portofino Street

Fully Furnished 2 BHK in School Street

Single Room anywhere in Lavasa

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [3]:
```
data=pd.read_csv(r"C:\Users\jagan\OneDrive\Desktop\DELL\2nd sem\machine learning\HouseP
data
```

Out[3]:

	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
...	...	...	...	...	...	...	...	...
995	Super Furnished Villa	Portofino D	4 BHK	4900.0	4	6	3	70000.0

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMonth
996	Super Furnished Villa	Portofino B	4 BHK	3750.0	4	5	0	76000.0
997	Super Furnished Villa	School Street	4 BHK	5270.0	4	5	3	80000.0
998	Super Furnished Villa	Portofino B	6 BHK	5100.0	7	6	3	90000.0
999	Super Furnished Villa	Portofino B	7 BHK	6300.0	6	6	3	96000.0

1000 rows × 8 columns

In [4]:
```
data.head(5)
```

Out[4]:

	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

In [5]:
```
for i in data['Location']:
    if i!="School Street" and i!="Clubview Road" and i!="Starter Homes":
        data['Location'].replace({i:'Portofino'},inplace=True)
```

In [6]:
```
data
```

Out[6]:

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMonth
0	Minimum Budget Rooms	Portofino	1 BHK	400.0	1	1	1	1100.0
1	Minimum Budget Rooms	Portofino	1 BHK	450.0	1	1	1	1100.0

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMonth
2	Minimum Budget Rooms	School Street	1 BHK	530.0	1	1	0	1166.0
3	Minimum Budget Rooms	Portofino	1 BHK	400.0	1	1	0	1400.0
4	Minimum Budget Rooms	School Street	2 BHK	460.0	1	1	0	1500.0
...	...	...	...	...	...	...	...	...
995	Super Furnished Villa	Portofino	4 BHK	4900.0	4	6	3	70000.0
996	Super Furnished Villa	Portofino	4 BHK	3750.0	4	5	0	76000.0
997	Super Furnished Villa	School Street	4 BHK	5270.0	4	5	3	80000.0
998	Super Furnished Villa	Portofino	6 BHK	5100.0	7	6	3	90000.0
999	Super Furnished Villa	Portofino	7 BHK	6300.0	6	6	3	96000.0

1000 rows × 8 columns

In [7]:
```
data.info()
```

```
<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 [8]:
```
data.describe()
```

Out[8]:

	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMonth

	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMonth
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	1548.270010	2.661000	2.168000	1.544000	10476.633500
std	1345.141175	1.247251	0.959529	0.838312	10509.508971
min	375.000000	1.000000	1.000000	0.000000	1100.000000
25%	1090.000000	2.000000	2.000000	1.000000	4890.500000
50%	1270.000000	2.000000	2.000000	2.000000	7000.000000
75%	1664.250000	3.000000	2.000000	2.000000	11925.000000
max	35000.000000	11.000000	6.000000	3.000000	96000.000000

In [9]:
```
data.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	max
AreaSqFt	1000.0	1548.27001	1345.141175	375.0	1090.0	1270.0	1664.25	35000.0
NoOfBath	1000.0	2.66100	1.247251	1.0	2.0	2.0	3.00	11.0
NoOfPeople	1000.0	2.16800	0.959529	1.0	2.0	2.0	2.00	6.0
NoOfBalcony	1000.0	1.54400	0.838312	0.0	1.0	2.0	2.00	3.0
RentPerMonth	1000.0	10476.63350	10509.508971	1100.0	4890.5	7000.0	11925.00	96000.0

In [10]:
```
data.tail()
```

Out[10]:

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMonth
995	Super Furnished Villa	Portofino	4 BHK	4900.0	4	6	3	70000.0
996	Super Furnished Villa	Portofino	4 BHK	3750.0	4	5	0	76000.0
997	Super Furnished Villa	School Street	4 BHK	5270.0	4	5	3	80000.0
998	Super Furnished Villa	Portofino	6 BHK	5100.0	7	6	3	90000.0
999	Super Furnished Villa	Portofino	7 BHK	6300.0	6	6	3	96000.0

In [11]:
```
data["RentPerMonth"].value_counts
```

Out[11]:
```
<bound method IndexOpsMixin.value_counts of 0       1100.0
1       1100.0
2       1166.0
3       1400.0
4       1500.0
        ...
995    70000.0
996    76000.0
997    80000.0
998    90000.0
999    96000.0
Name: RentPerMonth, Length: 1000, dtype: float64>
```

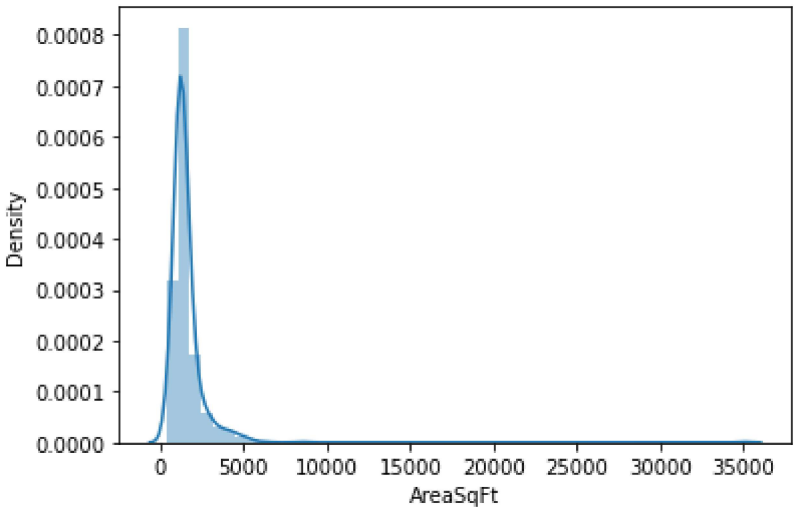
In [12]:
```
data.Location.value_counts
```

Out[12]:
```
<bound method IndexOpsMixin.value_counts of 0          Portofino
1          Portofino
2      School Street
3          Portofino
4      School Street
           ...
995        Portofino
996        Portofino
997    School Street
998        Portofino
999        Portofino
Name: Location, Length: 1000, dtype: object>
```

VISUALIZATION

In [13]:
```
ax=sns.distplot(data.AreaSqFt)
```

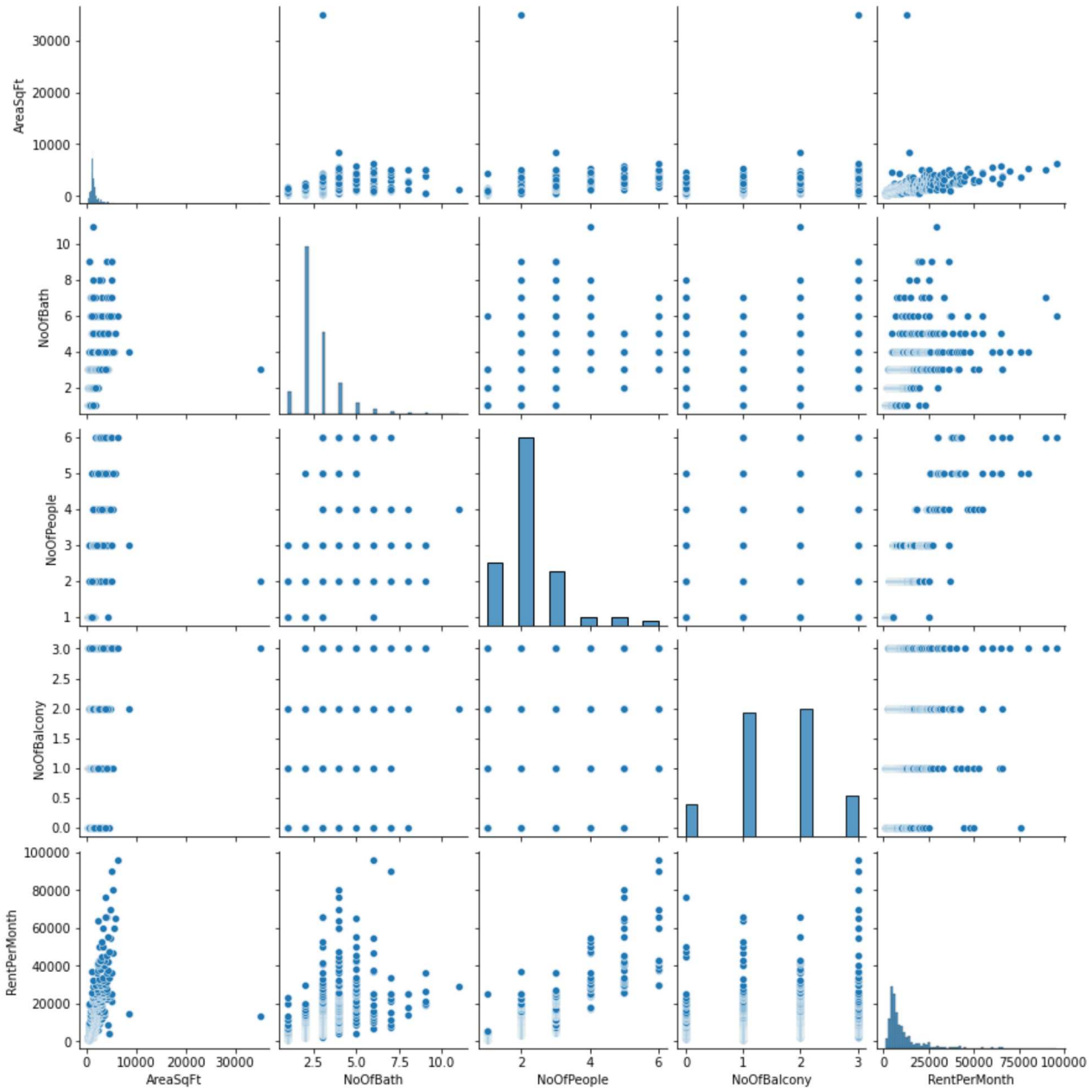
```
C:\Users\jagan\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [14]:
```
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x19b92e5b910>
```

Out[14]:

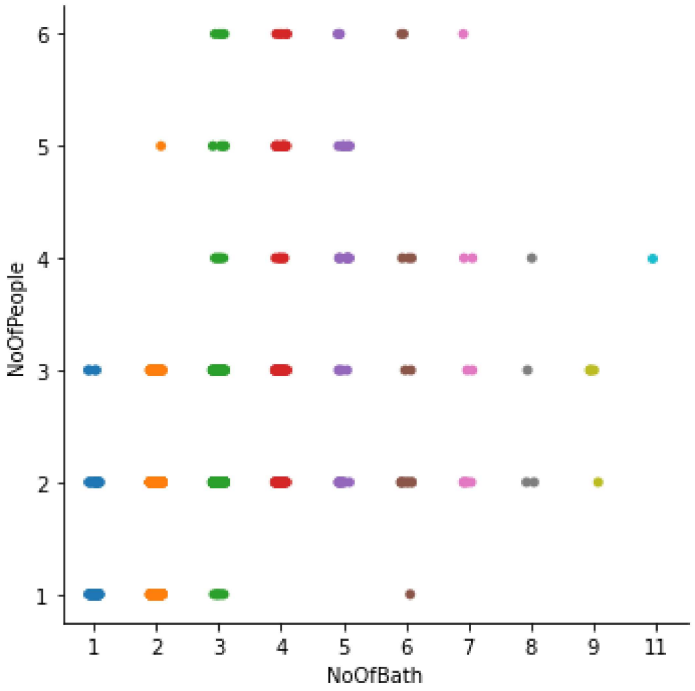


In [15]:

```
sns.catplot(x="NoOfBath",y="NoOfPeople",data=data)
```

Out[15]:

```
<seaborn.axisgrid.FacetGrid at 0x19b936e29a0>
```



In [16]:
```
from sklearn.linear_model import LinearRegression
```

In [17]:
```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
```

In [18]:
```
data['BuildingType']=le.fit_transform(data['BuildingType'])
data['Location']=le.fit_transform(data['Location'])
data['Size']=le.fit_transform(data['Size'])
```

3. Note down the errors/losses when the train-test ratio is 50:50, 60:40, 70:30, and 80:20

50:50

In [69]:
```
x_train,x_test,y_train,y_test= train_test_split(data.iloc[:,:7],data['RentPerMonth'],te
```

In [70]:
```
model=LinearRegression()
```

In [71]:
```
model.fit(x_train,y_train)
```

Out[71]:
```
LinearRegression()
```

In [72]:
```
y_pred=model.predict(x_test)
```

In [73]:
```
df1=pd.DataFrame({'Actual value':y_test,'predicted value':y_pred})
```

In [74]:
```
df1
```

Out[74]:

	Actual value	predicted value
890	20300.0	21825.550774
694	10000.0	7315.109902
798	13500.0	17334.446798
147	4100.0	6590.895394
858	16500.0	16620.259678
...	...	...
643	9100.0	11830.881537
686	9900.0	12081.384139
698	10000.0	12375.226702
897	21300.0	26967.333225
181	4400.0	6212.383001

500 rows × 2 columns

In [ ]:
```

```

In [75]:
```
data
```

Out[75]:

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMonth
0	3	1	0	400.0	1	1	1	1100.0
1	3	1	0	450.0	1	1	1	1100.0
2	3	2	0	530.0	1	1	0	1166.0
3	3	1	0	400.0	1	1	0	1400.0
4	3	2	2	460.0	1	1	0	1500.0
...	...	...	...	...	...	...	...	...
995	9	1	4	4900.0	4	6	3	70000.0
996	9	1	4	3750.0	4	5	0	76000.0
997	9	2	4	5270.0	4	5	3	80000.0
998	9	1	6	5100.0	7	6	3	90000.0
999	9	1	7	6300.0	6	6	3	96000.0

1000 rows × 8 columns

In [76]:
```
print(model.coef_)
```

```
[ 523.8990979  -860.09955689  -74.67963911    5.3074351  1287.01501514
 4829.14590797 -638.36835808]
```

In [77]:
```
print(model.intercept_)
```

```
-11414.385102805429
```

In [78]:
```
from sklearn import metrics
```

In [79]:
```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 3781.2471538792524
```

In [80]:
```
print('Mean Squared Error:', np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))
```

```
Mean Squared Error: 9.14819786523915
```

In [81]:
```
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Root Mean Squared Error: 9397.489566908966
```

In [82]:
```
x_train
```

Out[82]:

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony
827	4	1	6	1407.0	4	3	1
419	5	1	3	1500.0	3	3	1
221	5	2	2	1200.0	2	2	2
207	5	1	2	1145.0	2	1	2
531	1	1	2	1105.0	2	2	1
...	...	...	...	...	...	...	...
924	6	0	4	2470.0	5	4	2
223	3	2	3	1025.0	2	2	1
271	5	0	2	1080.0	2	3	2
474	4	1	2	1246.0	2	2	1
355	4	1	3	1464.0	3	2	2

500 rows × 7 columns

In [90]:
```
x4=metrics.mean_absolute_error(y_test, y_pred)
y4=metrics.mean_squared_error(y_test, y_pred)
z4=metrics.mean_squared_error(y_test, y_pred)
```

In [91]:
```
x_test
```

Out[91]:

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony
890	6	0	3	2376.0	3	3	1
694	1	1	3	1450.0	2	2	1
798	8	1	3	1735.0	3	3	3
147	3	0	2	940.0	2	2	1
858	4	1	3	1875.0	3	3	2
...	...	...	...	...	...	...	...
643	4	1	3	1640.0	4	2	2
686	1	1	6	1300.0	6	2	0
698	4	0	3	1823.0	3	2	2
897	6	1	5	3050.0	5	3	1
181	5	2	3	1250.0	2	2	3

500 rows × 7 columns

In [85]:
```
y_train
```

Out[85]:
```
827    15000.0
419     6195.0
221     4680.0
207     4579.0
531     7500.0
        ...
924    24700.0
223     4700.0
271     5000.0
474     6700.0
355     5600.0
Name: RentPerMonth, Length: 500, dtype: float64
```

In [86]:
```
y_test
```

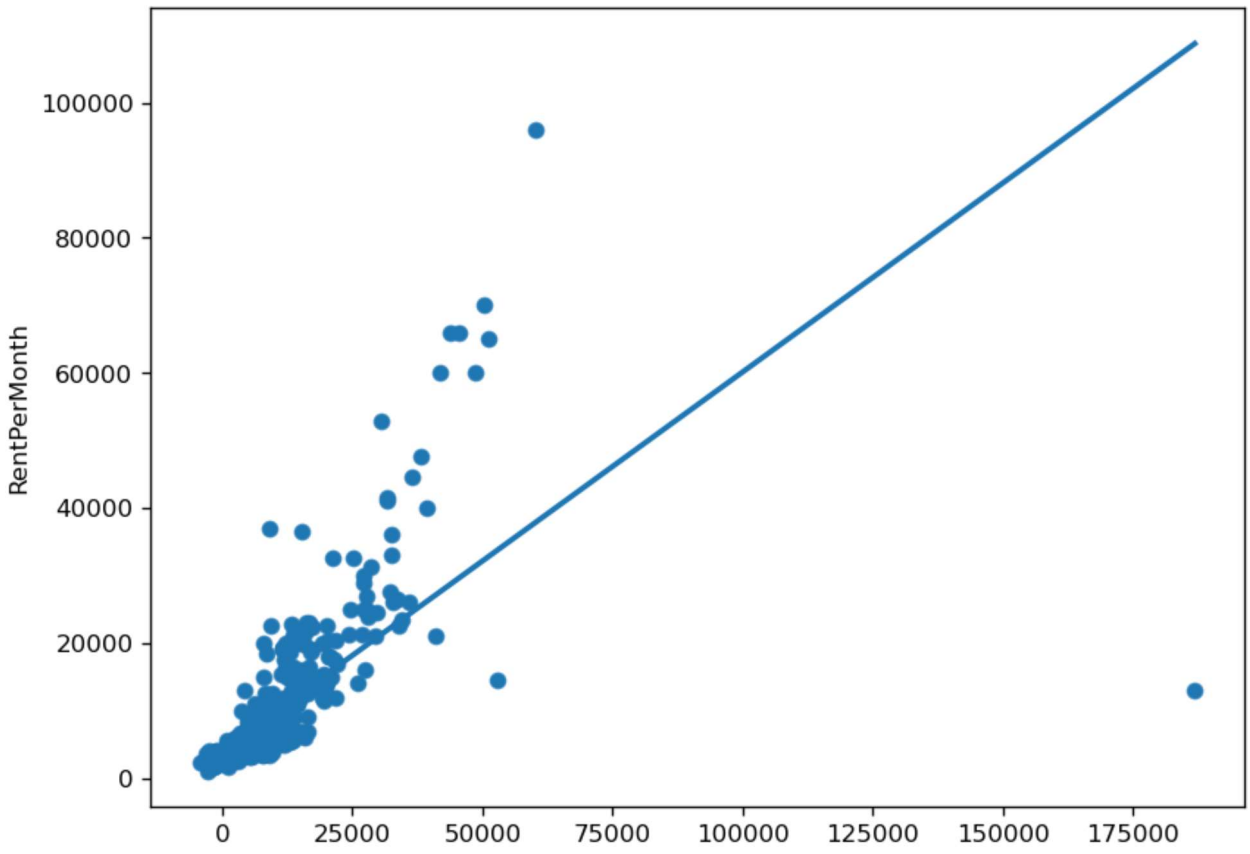
Out[86]:
```
890    20300.0
694    10000.0
798    13500.0
147     4100.0
858    16500.0
        ...
643     9100.0
686     9900.0
698    10000.0
897    21300.0
181     4400.0
Name: RentPerMonth, Length: 500, dtype: float64
```

In [87]:
```
from matplotlib.pyplot import figure
```

```
figure(figsize=(8,6), dpi=120)
sns.scatterplot(y_pred,y_test)
sns.regplot(y_pred,y_test,ci=None)
```

```
C:\Users\jagan\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variables as keyword args: x, y. From version 0.12, the only valid posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
C:\Users\jagan\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variables as keyword args: x, y. From version 0.12, the only valid posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```

Out[87]: <AxesSubplot:ylabel='RentPerMonth'>



60:40

In [47]:
```
x_train,x_test,y_train,y_test= train_test_split(data.iloc[:,:7],data['RentPerMonth'],te
```

In [48]:
```
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
df1=pd.DataFrame({'Actual value':y_test,'predicted value':y_pred})
df1
data
```

Out[48]:

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMonth

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMonth
0	3	1	0	400.0	1	1	1	1100.0
1	3	1	0	450.0	1	1	1	1100.0
2	3	2	0	530.0	1	1	0	1166.0
3	3	1	0	400.0	1	1	0	1400.0
4	3	2	2	460.0	1	1	0	1500.0
...	...	...	...	...	...	...	...	...
995	9	1	4	4900.0	4	6	3	70000.0
996	9	1	4	3750.0	4	5	0	76000.0
997	9	2	4	5270.0	4	5	3	80000.0
998	9	1	6	5100.0	7	6	3	90000.0
999	9	1	7	6300.0	6	6	3	96000.0

1000 rows × 8 columns

In [49]:
```
print(model.coef_)
print(model.intercept_)
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))
print('Mean Squared Error:', np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
x1=
```

```
  File "C:\Users\jagan\AppData\Local\Temp/ipykernel_27044/868482619.py", line 8
    x1=
       ^
SyntaxError: invalid syntax
```

In [50]:
```
x1=metrics.mean_absolute_error(y_test, y_pred)
y1=metrics.mean_squared_error(y_test, y_pred)
z1=metrics.mean_squared_error(y_test, y_pred)
```

In [51]:
```
x_test
```

Out[51]:

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony
923	6	1	4	2894.0	4	4	1
921	0	1	3	2150.0	4	3	2
516	4	1	2	1080.0	2	2	2
87	3	1	2	805.0	2	1	2
879	6	1	3	1630.0	3	2	2
...	...	...	...	...	...	...	...

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony
499	4	1	2	1195.0	2	2	1
436	5	0	2	1195.0	2	2	0
485	4	2	3	1282.0	2	2	2
22	5	1	2	656.0	2	1	1
715	4	0	3	1710.0	3	2	2

400 rows × 7 columns

70:30

In [52]:
```
x_train,x_test,y_train,y_test= train_test_split(data.iloc[:,:7],data['RentPerMonth'],te
```

In [53]:
```
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
df2=pd.DataFrame({'Actual value':y_test,'predicted value':y_pred})
df2
data
```

Out[53]:

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMonth
0	3	1	0	400.0	1	1	1	1100.0
1	3	1	0	450.0	1	1	1	1100.0
2	3	2	0	530.0	1	1	0	1166.0
3	3	1	0	400.0	1	1	0	1400.0
4	3	2	2	460.0	1	1	0	1500.0
...	...	...	...	...	...	...	...	...
995	9	1	4	4900.0	4	6	3	70000.0
996	9	1	4	3750.0	4	5	0	76000.0
997	9	2	4	5270.0	4	5	3	80000.0
998	9	1	6	5100.0	7	6	3	90000.0
999	9	1	7	6300.0	6	6	3	96000.0

1000 rows × 8 columns

In [54]:
```
print(model.coef_)
print(model.intercept_)
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.log(np.sqrt(metrics.mean_squared_error(y_test, y_p
```

```
[ 6.25217156e+02 -4.45483431e+02 -7.06469434e+02  8.49636904e-01
  2.56841905e+03  6.62318101e+03  4.97007159e+01]
-12424.854263855186
Mean Absolute Error: 4040.098721901082
Mean Squared Error: 8.75410947721206
Root Mean Squared Error: 8.75410947721206
```

In [55]:
```
x2=metrics.mean_absolute_error(y_test, y_pred)
y2=metrics.mean_squared_error(y_test, y_pred)
z2=metrics.mean_squared_error(y_test, y_pred)
```

In [ ]:
```
```

In [56]:
```
x_test
```

Out[56]:

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony
923	6	1	4	2894.0	4	4	1
921	0	1	3	2150.0	4	3	2
516	4	1	2	1080.0	2	2	2
87	3	1	2	805.0	2	1	2
879	6	1	3	1630.0	3	2	2
...	...	...	...	...	...	...	...
857	4	2	4	1200.0	3	3	1
782	8	1	2	1200.0	2	2	2
598	1	0	3	1515.0	3	2	1
93	3	0	2	770.0	1	2	1
554	4	0	2	1279.0	2	2	1

300 rows × 7 columns

80:20

In [57]:
```
x_train,x_test,y_train,y_test= train_test_split(data.iloc[:,:7],data['RentPerMonth'],te
```

In [58]:
```
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
df3=pd.DataFrame({'Actual value':y_test,'predicted value':y_pred})
df3
data
```

Out[58]:

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMonth
0	3	1	0	400.0	1	1	1	1100.0

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMonth
1	3	1	0	450.0	1	1	1	1100.0
2	3	2	0	530.0	1	1	0	1166.0
3	3	1	0	400.0	1	1	0	1400.0
4	3	2	2	460.0	1	1	0	1500.0
...	...	...	...	...	...	...	...	...
995	9	1	4	4900.0	4	6	3	70000.0
996	9	1	4	3750.0	4	5	0	76000.0
997	9	2	4	5270.0	4	5	3	80000.0
998	9	1	6	5100.0	7	6	3	90000.0
999	9	1	7	6300.0	6	6	3	96000.0

1000 rows × 8 columns

In [59]:
```
print(model.coef_)
print(model.intercept_)
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))
print('Mean Squared Error:', np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))
```

```
[ 6.06392938e+02 -5.63454546e+02 -5.34329164e+02  9.73094996e-01
  2.49212397e+03  6.65509106e+03  1.03807932e+02]
-12706.176734055973
Mean Absolute Error: 4114.144610014843
Mean Squared Error: 8.707710550107851
Mean Squared Error: 8.707710550107851
```

In [60]:
```
x3=metrics.mean_absolute_error(y_test, y_pred)
y3=metrics.mean_squared_error(y_test, y_pred)
z3=metrics.mean_squared_error(y_test, y_pred)
```

In [61]:
```
x_test
```

Out[61]:

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony
923	6	1	4	2894.0	4	4	1
921	0	1	3	2150.0	4	3	2
516	4	1	2	1080.0	2	2	2
87	3	1	2	805.0	2	1	2
879	6	1	3	1630.0	3	2	2
...	...	...	...	...	...	...	...
711	4	1	4	1863.0	3	2	2

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony
517	1	1	4	700.0	4	2	1
984	2	1	3	2610.0	3	4	0
886	0	1	4	1200.0	4	3	2
961	7	0	3	2480.0	3	5	1

200 rows × 7 columns

Questions

1. What are your observations on the Dataset?

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

3. Note down the errors/losses when the train-test ratio is 50:50, 60:40, 70:30, and 80:20

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

1. Observation on the dataset

- The dataset shows the
  1. Building Type - Is it a fully/semi/Un furnished Single Room, Flat, or Villa ?
  2. Location - Where is the property located?
  3. Size - Is it 1BHK, 2BHK, 3BHK ?
  4. AreaSqFt - How much big is the property ?
  5. No of Bath - How many bathrooms in the property?
  6. No of Balcony - How many balconies in the property?
  7. No of People - How many people stayed in the building in the academic year 2020-21.
  8. RentPerMonth - Rent to be paid per month which is demanded by the current building owners.

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

- Different Error Measures (Evaluation Metrics) in relation to Linear Regression are MAE, MSE, RMSE.

In [97]:
```
tdf=pd.DataFrame(columns=["SPLITRATIO","MAE","MSE","RMSE"])
F=[
    ["50:50",x1,y1,z1],["60:40",x2,y2,z2],["70:30",x3,y3,z3],["80:20",x4,y4,z4]

]
for i in F:
    t_df=pd.DataFrame([i],columns=["SPLITRATIO","MAE","MSE","RMSE"])
    tdf=tdf.append(t_df,ignore_index=True)
```

In [98]:
```
tdf
```

Out[98]:

	SPLITRATIO	MAE	MSE	RMSE
0	50:50	3914.141979	3.389727e+07	3.389727e+07
1	60:40	4040.098722	4.015345e+07	4.015345e+07
2	70:30	4114.144610	3.659496e+07	3.659496e+07
3	80:20	3781.247154	8.831281e+07	8.831281e+07

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

- During LinearRegression() process if we put Normalize Parameter equals to false, then the errors are less.

Conclusion:

- In this lab, i tried to predict the values from the given dataset, and we implemented various graphs using various libraries in order to get valuable insights.next we implemented and evaluated LinearRegression model to get high accuracy in term of predicting rental price

Reference

- https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/
- https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/
- https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model
- https://www.kaggle.com/c/house-prices-advanced-regression-techniques

In [ ]:
```

```