LAB 3 Prediction of Numeric Values

SUBMITTED BY:

Name : Kavitha.S
Reg no : 21122033
Class : 2MSDS

LAB OVERVIEW:

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?

PROBLEM DEFINITION:

To import the dataset given and perform various exploratory data analysis on it. Split the dataset into train and test. Create a linear regression model and finding the best split ratio. Observing the various evaluation metrics of the dataset for the various split ratios. Visualizing various observations.

APPROACH:

Use Pandas to Import the Dataset.
Performing necessary Exploratory Data Analysis. Visualizing the dataset using various plots from matplotlib and seaborn.
Use the train_test_split method available in SCIKIT to split the dataset into Train Dataset and Test Dataset.
Building a Regression model for the various train test split ratios of the dataset.
Based on the best R2 score, we will decide the best split ratio.

In [57]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn as sk
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

In [36]:
```
house = pd.read_csv("HousePrices - Lab3.csv")
```

In [3]:
```
house.head()
```

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

In [4]:
```
print('\nNumber of rows and columns in the data set: ',house.shape)
print('')
```

```
Number of rows and columns in the data set:  (1000, 8)
```

In [5]:
```
house.columns
```

Out[5]:
```
Index(['BuildingType', 'Location', 'Size', 'AreaSqFt', 'NoOfBath',
       'NoOfPeople', 'NoOfBalcony', 'RentPerMonth'],
      dtype='object')
```

The target variable here is the RentPerMonth and the rest 6 variables such as the BuildingType, Location , Size, AreaSqFt, NoOfBath , No Of People, No Of Balcony are the features (independent variable). There are multiple independent variable, so we need to fit Multiple linear regression.

EXPLORATORY DATA ANALYSIS

In [6]:
```
house.describe()
```

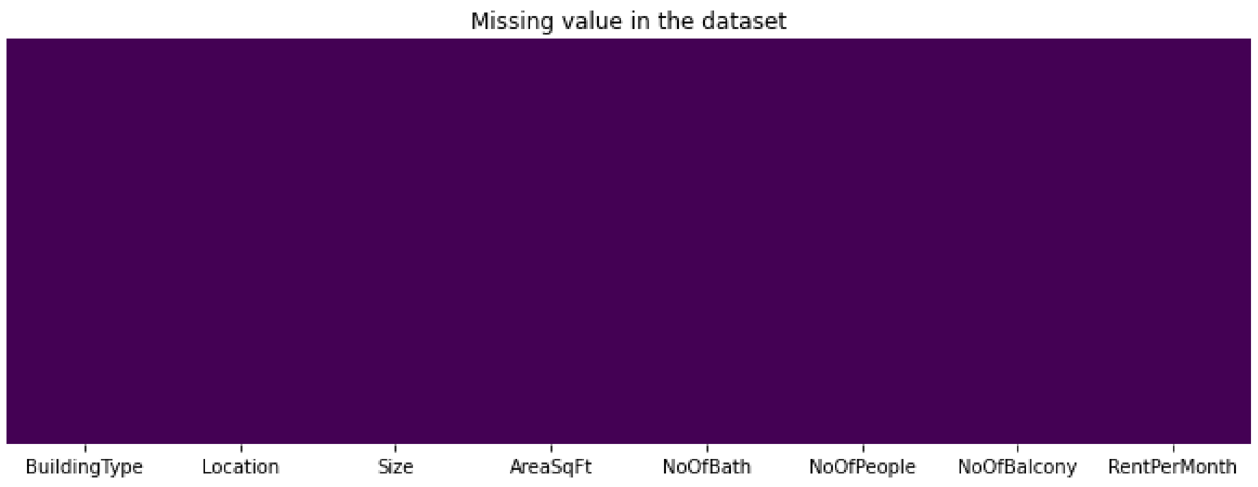
Out[6]:

	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 [7]:
```
house.isna().sum()
```

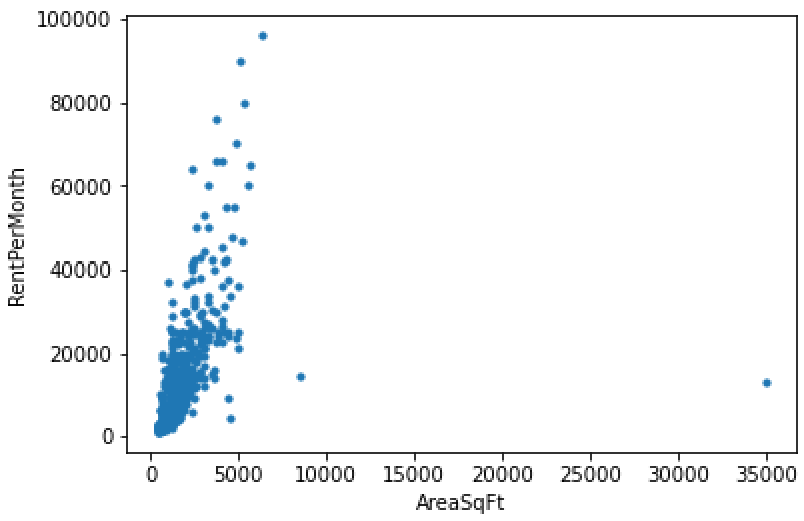
Out[7]:
```
BuildingType    0
Location        0
Size            0
AreaSqFt        0
NoOfBath        0
NoOfPeople      0
NoOfBalcony     0
RentPerMonth    0
dtype: int64
```

In [8]:
```
plt.figure(figsize=(12,4))
sns.heatmap(house.isnull(),cbar=False,cmap='viridis',yticklabels=False)
plt.title('Missing value in the dataset');
```



There is no missingvalues in the dataset

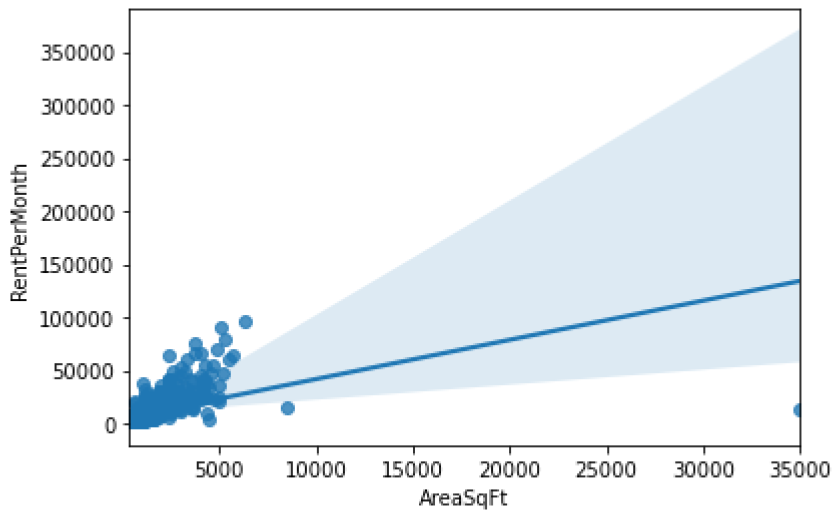
In [9]:
```
house.plot.scatter(x = 'AreaSqFt', y = 'RentPerMonth', s = 10);
```



In [10]:
```
sns.regplot(x = 'AreaSqFt', y = 'RentPerMonth',data=house)
```

```
<AxesSubplot:xlabel='AreaSqFt', ylabel='RentPerMonth'>
```

Out[10]:



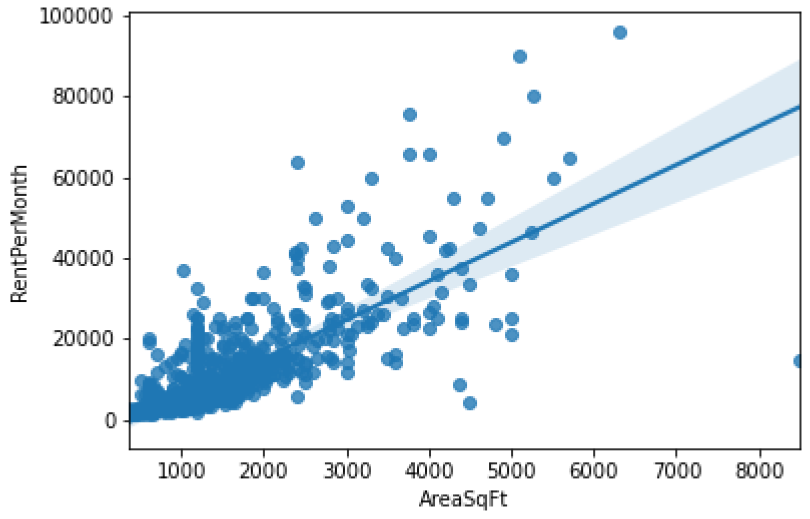
As we can see in the above graph, there is an outlier in the column area per square fit.There is one outlier far from the other points, though it only appears to slightly influence the line. Points that fall horizontally far from the line are points of high leverage; these points can strongly influence the slope of the least squares line. Outliers can have a dramatic impact on linear regression. It can change the model equation completely i.e. bad prediction or estimation.

In [11]:
```
house.drop(house.index[house['AreaSqFt'] == 35000], inplace = True)
```

Removing the outlier and plotting the regression line again.

In [12]:
```
sns.regplot(x = 'AreaSqFt', y = 'RentPerMonth',data=house)
```

Out[12]:
```
<AxesSubplot:xlabel='AreaSqFt', ylabel='RentPerMonth'>
```



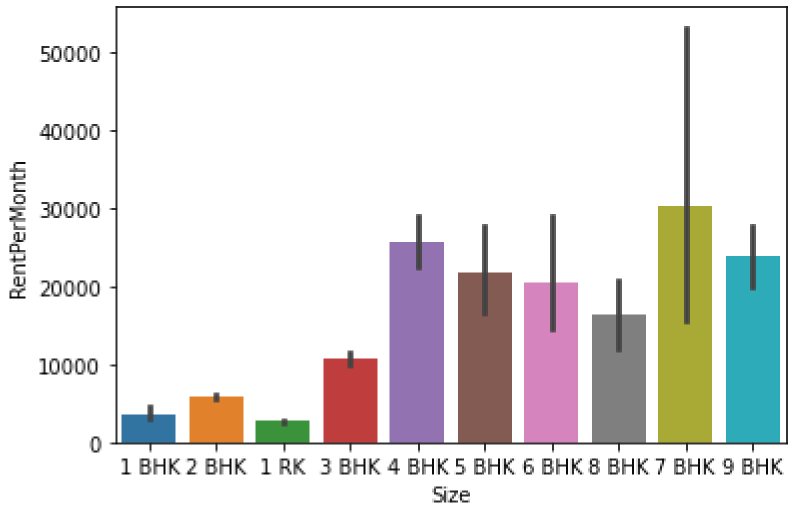
In [13]:
```
sns.barplot("Size","RentPerMonth",data=house)
```

```
C:\Users\SRIDHAR\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: P
ass the following variables as keyword args: x, y. From version 0.12, the only valid pos
itional argument will be `data`, and passing other arguments without an explicit keyword
```

```
will result in an error or misinterpretation.
  warnings.warn(
```

Out[13]:

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



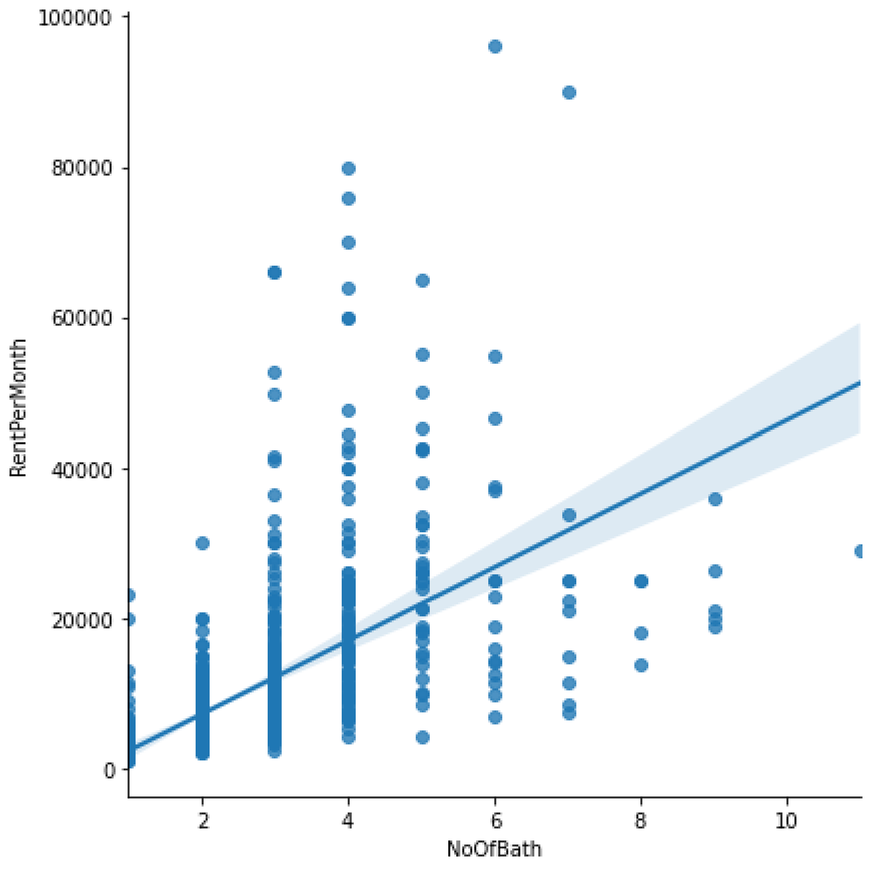
For our visualization purpose will fit regression line using seaborn library only for No of bath as independent variable and Rent per month as dependent variable.

In [14]:

```
sns.lmplot(x='NoOfBath',y='RentPerMonth',data=house,aspect=1,height=6)
```

Out[14]:

```
<seaborn.axisgrid.FacetGrid at 0x1dc6e25de50>
```

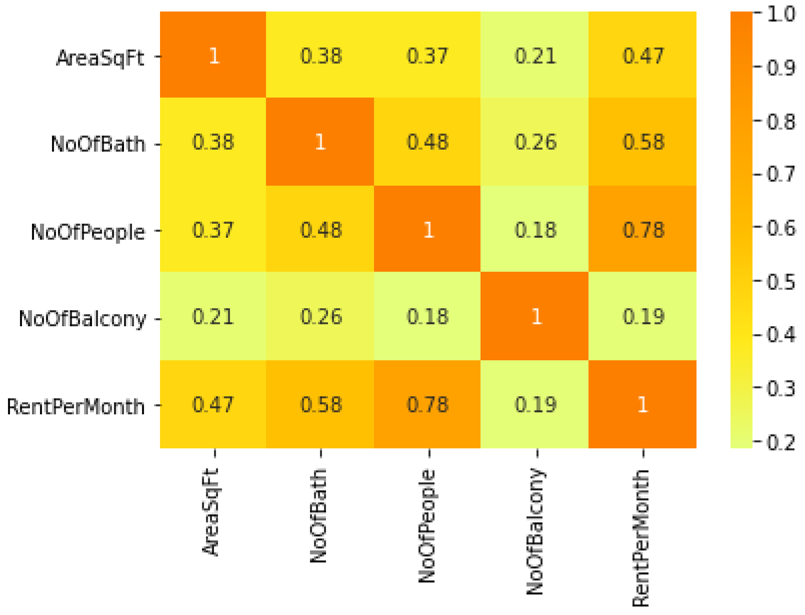


In above plot we fit regression line into the variables.

In [37]:

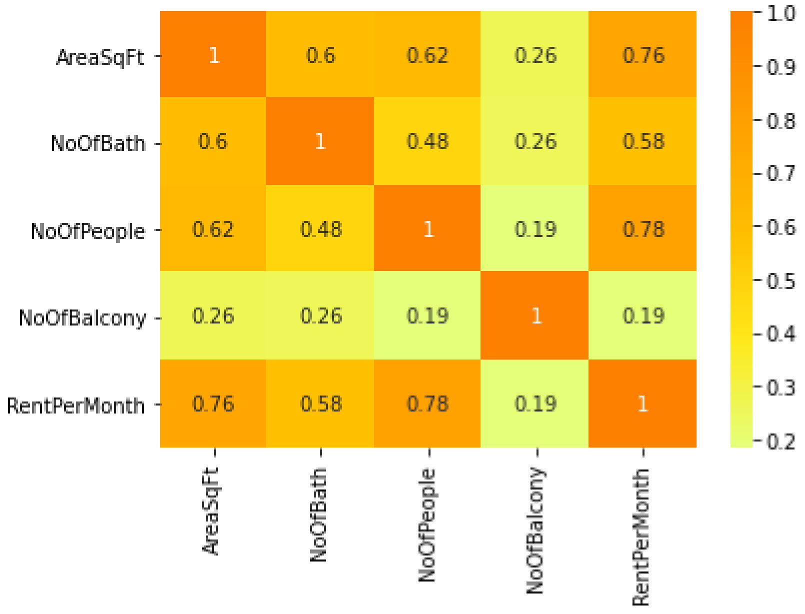
```
corr = house.corr()
```

```
sns.heatmap(corr, cmap = 'Wistia', annot= True);
```



The above graph was before removing the outlier, as you can see the correlation between the Area Sq ft and rent per month is not much stronger. But after removing the outlier, the correlation seems to have changed drastically. Making it strongly correlated to the Rent.

In [15]:
```
corr = house.corr()
sns.heatmap(corr, cmap = 'Wistia', annot= True);
```



The correlation between the number of people and the rent per month is higher. The correlation between the number of balcony and rent per month is weaker.

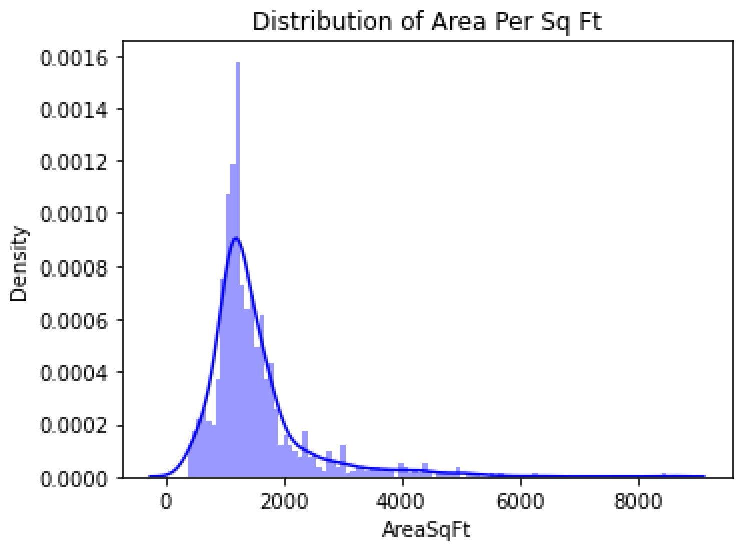
In [16]:
```
f= plt.figure(figsize=(12,4))

ax=f.add_subplot(121)
sns.distplot(house['AreaSqFt'],bins=100,color='b',ax=ax)
ax.set_title('Distribution of Area Per Sq Ft')
```

```
C:\Users\SRIDHAR\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibilit
y) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[16]:

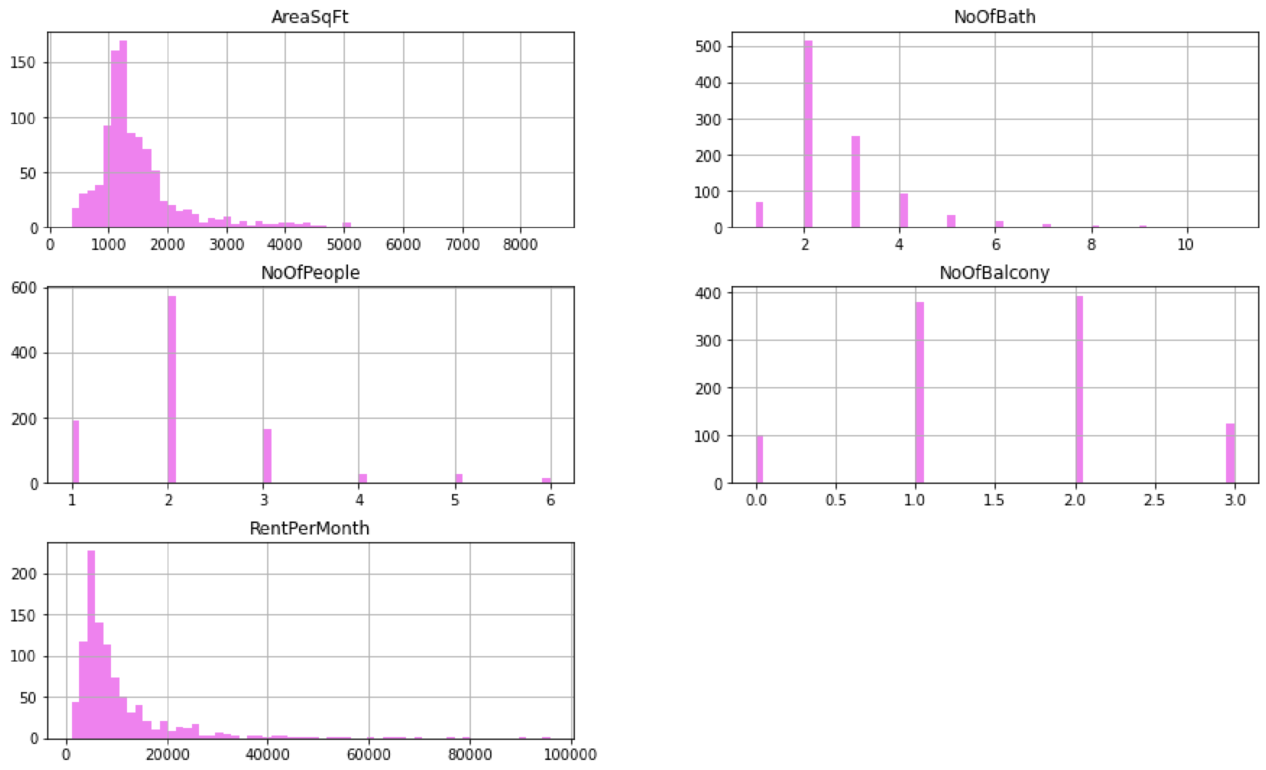
```
Text(0.5, 1.0, 'Distribution of Area Per Sq Ft')
```



In the plot the area per square feet varies from 350 to 9000, the plot is right skewed.

In [17]:

```python
house.hist(bins=60, figsize=(15,9),color='violet');
plt.show()
```

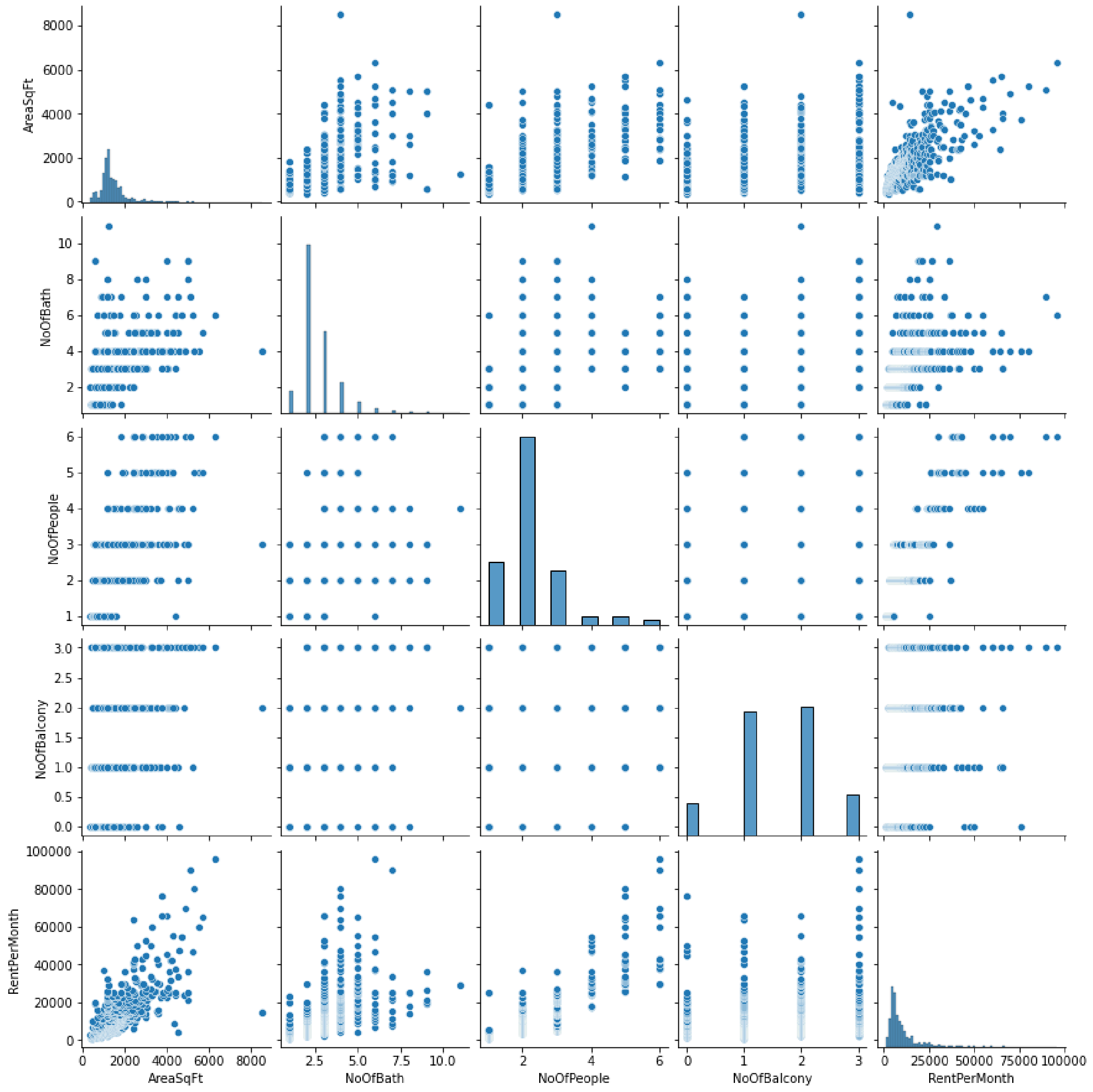


In [18]:

```python
sns.pairplot(house)
```

```
<seaborn.axisgrid.PairGrid at 0x1dc6e46a250>
```

Out[18]:



In [19]:
```
house['Location'].unique()
```

Out[19]:
```
array(['Portofino H', 'School Street', 'Portofino B', 'Portofino A',
       'Clubview Road', 'Portofino C', 'Portofino D', 'Starter Homes',
       'Portofino G', 'Portofino E', 'Portofino F'], dtype=object)
```

In [20]:
```
house['BuildingType'].unique()
```

Out[20]:
```
array(['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'], dtype=object)
```

In [21]:
```
house['Size'].unique()
```

Out[21]:
```
array(['1 BHK', '2 BHK', '1 RK', '3 BHK', '4 BHK', '5 BHK', '6 BHK',
       '8 BHK', '7 BHK', '9 BHK'], dtype=object)
```

Replacing all the Portofino subdivisions with portofino.

In [22]:
```
for i in house['Location']:
    if i!='School Street' and i!='Clubview Road' and i!='Starter Homes':
        house['Location'].replace({i:"Portofino"},inplace = True)
```

Encoding

Machine learning algorithms cannot work with categorical data directly, categorical data must be converted to number.
Label encoding refers to transforming the word labels into numerical form so that the algorithms can understand how to operate on them.

In [23]:
```
 # label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()
```

In [84]:
```
list=['BuildingType','Location','Size']
for i in list:
    print('Before Converting the data into machine-readable form, The values are')
    print('\n')
    print(house[i].unique())
    # Encode labels in column 'species'.
    house[i]= label_encoder.fit_transform(house[i])
    print('\n')
    print('After Converting the data into machine-readable form, The values are')
    print(house[i].unique())
    print('\n')
house.head()
```

```
Before Converting the data into machine-readable form, The values are


['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']


After Converting the data into machine-readable form, The values are
[3 5 4 1 8 6 0 7 2 9]


Before Converting the data into machine-readable form, The values are


['Portofino H' 'School Street' 'Portofino B' 'Portofino A' 'Clubview Road'
 'Portofino C' 'Portofino D' 'Starter Homes' 'Portofino G' 'Portofino E'
 'Portofino F']


After Converting the data into machine-readable form, The values are
[ 8  9  2  1  0  3  4 10  7  5  6]
```

```
Before Converting the data into machine-readable form, The values are


['1 BHK' '2 BHK' '1 RK' '3 BHK' '4 BHK' '5 BHK' '6 BHK' '8 BHK' '7 BHK'
 '9 BHK']


After Converting the data into machine-readable form, The values are
[0 2 1 3 4 5 6 8 7 9]
```

Out[84]:

	BuildingType	Location	Size	AreaSqFt	NoOfBath	NoOfPeople	NoOfBalcony	RentPerMonth
0	3	8	0	400.0	1	1	1	1100.0
1	3	8	0	450.0	1	1	1	1100.0
2	3	9	0	530.0	1	1	0	1166.0
3	3	2	0	400.0	1	1	0	1400.0
4	3	9	2	460.0	1	1	0	1500.0

QUESTION 2 & 3

Training a Linear Regression Model

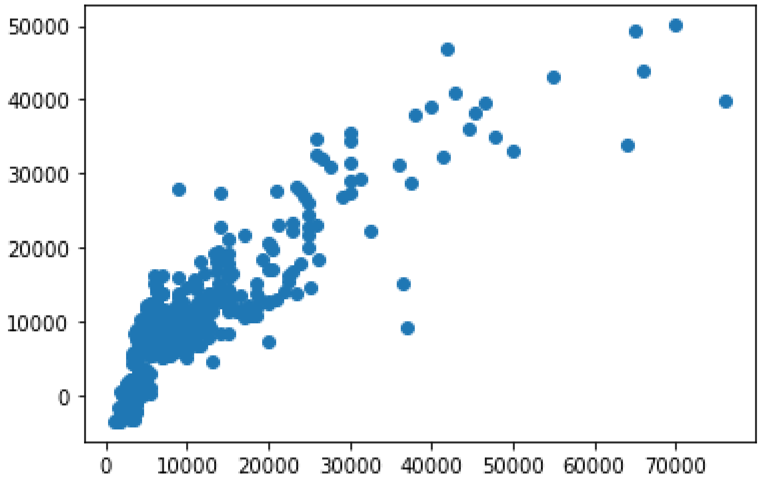
In [25]:

```
X = house[['BuildingType','Location','Size','AreaSqFt','NoOfBath','NoOfPeople','NoOfBal
y = house['RentPerMonth']
```

In [48]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=1
regressor = LinearRegression()
regressor.fit(X_train,y_train)
print('Intercept : ',regressor.intercept_)
predictions = regressor.predict(X_test)
plt.scatter(y_test,predictions)
score=r2_score(y_test,predictions)
print("score : ",score)
```

```
Intercept :  -12715.631070396275
score :  0.7638138013973114
```



In [69]:
```
coeff_df = pd.DataFrame(regressor.coef_,X.columns,columns=['Coefficient'])
coeff_df
```

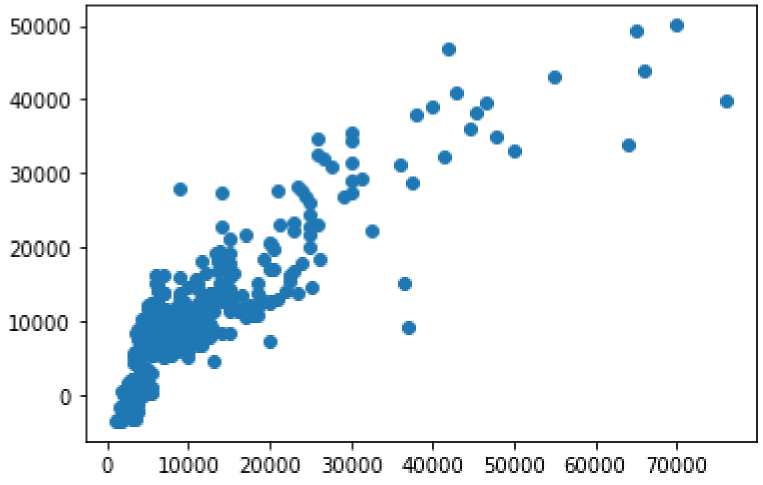
Out[69]:

	Coefficient
BuildingType	655.738643
Location	-442.219679
Size	-585.081902
AreaSqFt	4.959577
NoOfBath	1477.354856
NoOfPeople	5454.995061
NoOfBalcony	-263.146384

QUESTION 4:

In [60]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=1
regressor = LinearRegression(normalize=True)
regressor.fit(X_train,y_train)
print('Intercept : ',regressor.intercept_)
predictions = regressor.predict(X_test)
plt.scatter(y_test,predictions)
score=r2_score(y_test,predictions)
print("score : ",score)
```

```
Intercept :  -12715.631070396206
score :  0.7638138013973108
```

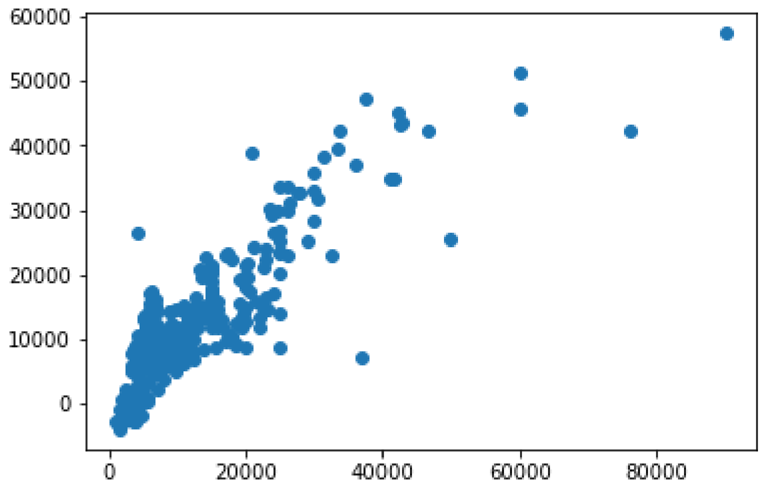


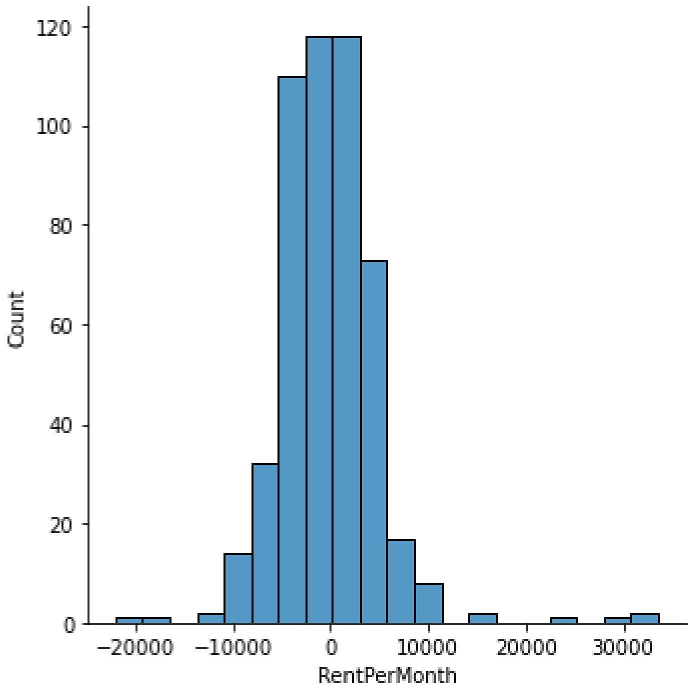
The parameter fit_intercept in LinearRegression() is set to True by default. It determines whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (i.e. data is expected to be centered). The normalize parameter is set to False by default. It is ignored when fit_intercept is set to False. If True, the regressors X will be normalized before regression by subtracting the mean and dividing by the l2-norm.

In [59]:
```
print('FOR TRAIN TEST RATIO : 50:50')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=2
regressor.fit(X_train,y_train)
```

```
print('Intercept : ',regressor.intercept_)
predictions =  regressor.predict(X_test)
plt.scatter(y_test,predictions)
sns.displot((y_test-predictions),bins=20)
score=r2_score(y_test,predictions)
print("R2_score : ",score)
reg = regressor.score(X_train,y_train)
print('Regression Score : ',reg)
print('Mean Absolute Error : ', metrics.mean_absolute_error(y_test, predictions))
print('Mean Squared Error : ', metrics.mean_squared_error(y_test, predictions))
print('Root Mean Squared Error : ', np.sqrt(metrics.mean_squared_error(y_test, predicti
```

```
FOR TRAIN TEST RATIO : 50:50
Intercept :  -13459.328710006246
R2_score :  0.7281375582554005
Regression Score :  0.7695277617912647
Mean Absolute Error :  3524.1729599714663
Mean Squared Error :  25213778.859468408
Root Mean Squared Error :  5021.332378907854
```





In [29]:

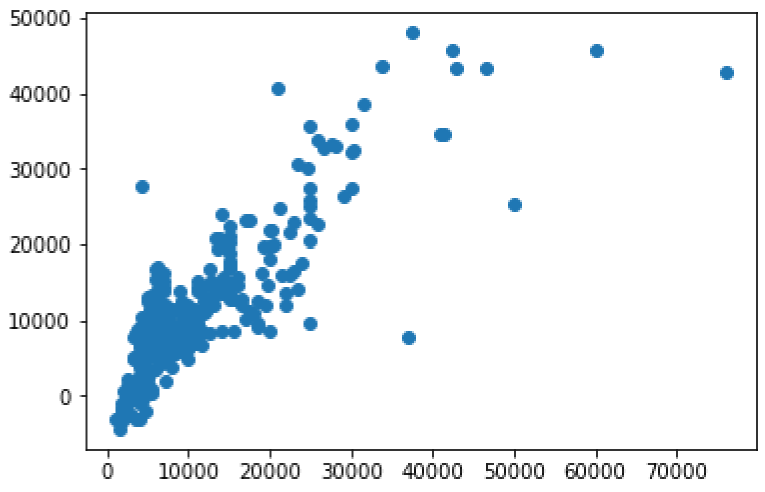
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

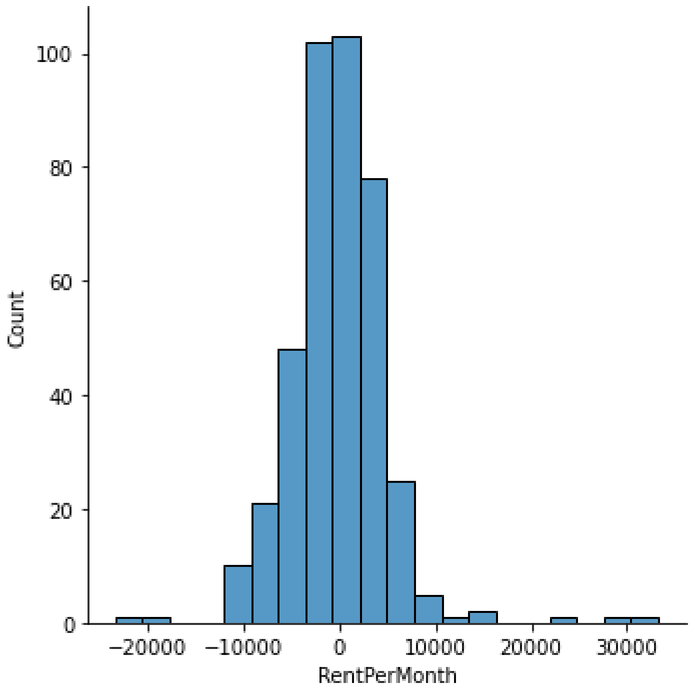
```
(499, 7)
(500, 7)
(499,)
(500,)
```

In [66]:

```
print('FOR TRAIN TEST RATIO : 60:40')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=2
regressor.fit(X_train,y_train)
reg = regressor.score(X_train,y_train)
print('Regression Score : ',reg)
print('Intercept : ',regressor.intercept_)
predictions =  regressor.predict(X_test)
plt.scatter(y_test,predictions)
sns.displot((y_test-predictions),bins=20)
score=r2_score(y_test,predictions)
print("R2 Score : ",score)
print('Mean Absolute Error : ', metrics.mean_absolute_error(y_test, predictions))
print('Mean Squared Error : ', metrics.mean_squared_error(y_test, predictions))
print('Root Mean Squared Error : ', np.sqrt(metrics.mean_squared_error(y_test, predicti
```

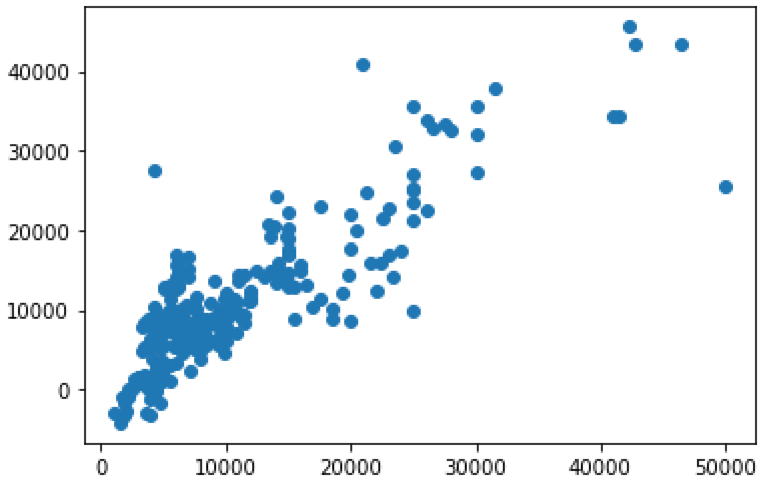
```
FOR TRAIN TEST RATIO : 60:40
Regression Score :  0.781870647487226
Intercept :  -13902.888222076368
R2 Score :  0.6745813645777065
Mean Absolute Error :  3556.7573626451417
Mean Squared Error :  25647671.794940338
Root Mean Squared Error :  5064.353048015149
```

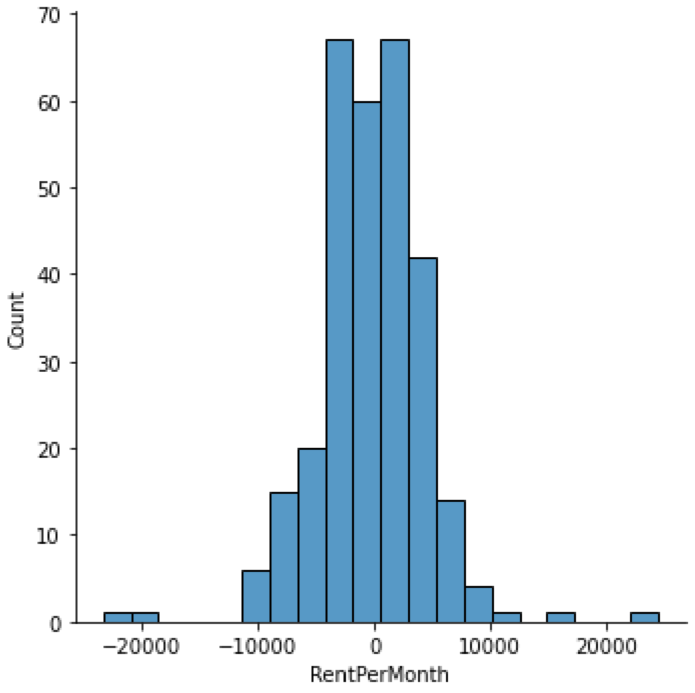




In [67]:
```
print('FOR TRAIN TEST RATIO : 70:30')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=2
regressor.fit(X_train,y_train)
reg = regressor.score(X_train,y_train)
print('Regression Score : ',reg)
print('Intercept : ',regressor.intercept_)
predictions =  regressor.predict(X_test)
plt.scatter(y_test,predictions)
sns.displot((y_test-predictions),bins=20)
score=r2_score(y_test,predictions)
print("R2 Score : ",score)
print('Mean Absolute Error : ', metrics.mean_absolute_error(y_test, predictions))
print('Mean Squared Error : ', metrics.mean_squared_error(y_test, predictions))
print('Root Mean Squared Error : ', np.sqrt(metrics.mean_squared_error(y_test, predicti
```

```
FOR TRAIN TEST RATIO : 70:30
Regression Score :  0.770056126382838
Intercept :  -13779.416378023063
R2 Score :  0.6672336533691589
Mean Absolute Error :  3422.39583019839
Mean Squared Error :  21427478.33641219
Root Mean Squared Error :  4628.9824299096435
```

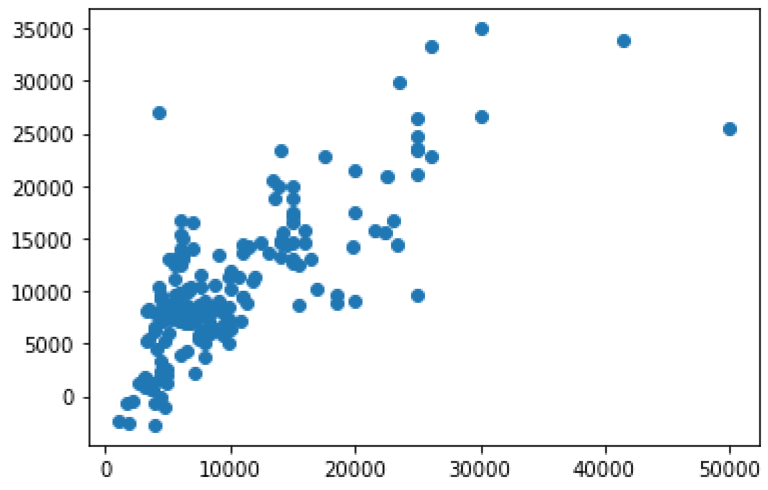


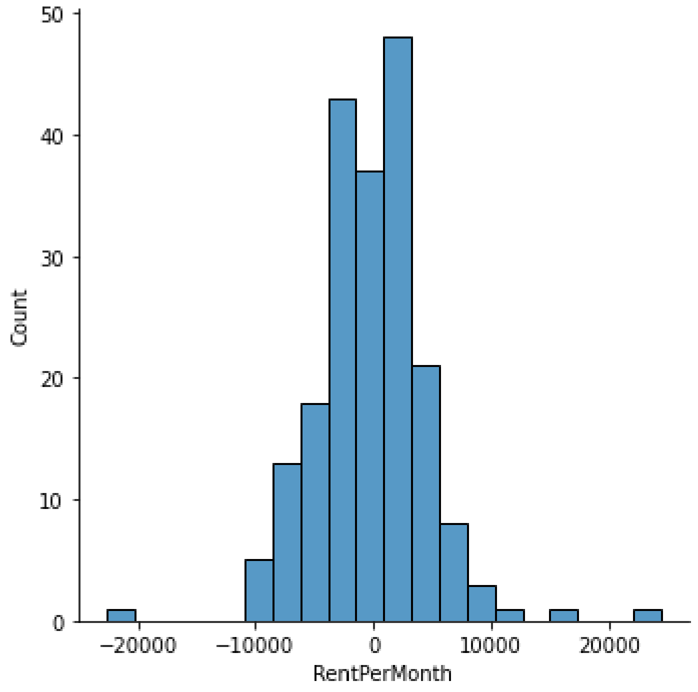


In [68]:

```
print('FOR TRAIN TEST RATIO : 80:20')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2
regressor.fit(X_train,y_train)
reg = regressor.score(X_train,y_train)
print('Regression Score : ',reg)
print('Intercept : ',regressor.intercept_)
predictions =  regressor.predict(X_test)
plt.scatter(y_test,predictions)
sns.displot((y_test-predictions),bins=20)
score=r2_score(y_test,predictions)
print("R2 Score : ",score)
print('Mean Absolute Error : ', metrics.mean_absolute_error(y_test, predictions))
print('Mean Squared Error : ', metrics.mean_squared_error(y_test, predictions))
print('Root Mean Squared Error : ', np.sqrt(metrics.mean_squared_error(y_test, predicti
```

```
FOR TRAIN TEST RATIO : 80:20
Regression Score :  0.7739185347819064
Intercept :  -12827.134812658895
R2 Score :  0.5584809277771455
Mean Absolute Error :  3516.1212157662517
Mean Squared Error :  22551378.806275953
Root Mean Squared Error :  4748.8292037381125
```





From the above test cases, it is observed that R2 score is high for the train test ratio 50:50.

QUESTION 1:

The dataset is about the rent of houses in Lavasa based on various features such as the location it is situated in, the size, the number of rooms. In the dataset the rent per month is the dependent variable(target) and all the other variables are independent making them the features of the dataset. There was an outlier in the column area per sq ft which was then removed because sometimes even a single outlier can make a bad prediction. As a result of that single outlier, the slope of the regression line changes greatly. There is a strong correlation between No of people and the rent per month. Before removing the outlier the correlation between area per sq ft and rent per month was weaker but after removing it became higher, showing how a single outlier can change the correlation between two variables. The Dataset was split in various train test ratios and performed Linear regression on it. The 50:50 split had higher R2 score, showing that maybe that split explains the relation between dependent and independent variable more than other splits.

USE CASE:

1. 1 BHK with 2 Baths in Portofino Street

In [83]:
```
df = pd.DataFrame()
df['BuildingType'] = [1]
df['Location'] = [8]
df['Size'] = [0]
df['AreaSqFt'] = [2]
df['NoOfBath'] = [2]
df['NoOfPeople'] = [5]
df['NoOfBalcony'] = [5]
y_pred =  regressor.predict(df)
print("Predicted rent is",y_pred)
```
```
Predicted rent is [13214.7186528]
```

1. Fully Furnished 2 BHK in School Street

In [75]:
```
df = pd.DataFrame()
df['BuildingType'] = [9]
df['Location'] = [2]
df['Size'] = [2]
df['AreaSqFt'] = [2]
df['NoOfBath'] = [3]
df['NoOfPeople'] = [5]
df['NoOfBalcony'] = [5]
y_pred =  regressor.predict(df)
print("Predicted rent is",y_pred)
```
```
Predicted rent is [21421.13692091]
```

1. Single Room anywhere in Lavasa

In [76]:
```
df = pd.DataFrame()
df['BuildingType'] = [3]
df['Location'] = [2]
df['Size'] = [3]
df['AreaSqFt'] = [2]
df['NoOfBath'] = [3]
df['NoOfPeople'] = [5]
df['NoOfBalcony'] = [5]
y_pred = regressor.predict(df1)
print("Predicted rent is",y_pred)
```
```
Predicted rent is [21421.13692091]
```

REFERENCES:

https://www.geeksforgeeks.org/pandas-scatter-plot-dataframe-plot-scatter/

https://www.kaggle.com/code/sudhirnl7/linear-regression-tutorial/notebook

https://study.com/learn/lesson/correlation-types-examples-statistics.html

https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/

https://www.geeksforgeeks.org/detect-and-remove-the-outliers-using-

python/#:~:text=Removing%20the%20outliers&text=Inplace%20%3DTrue%20is%20used%20to,it%20mu
https://stackabuse.com/linear-regression-in-python-with-scikit-learn/ https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html
https://www.google.com/search?q=displot+vs+distplot&rlz=1C1CHZN_enIN974IN974&oq=DISPLOT+VS+&aqs=chrome.0.0i512j69i57j0
8 https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/ https://stackoverflow.com/questions/23199796/detect-and-exclude-outliers-in-a-pandas-dataframe https://www.geeksforgeeks.org/python-coefficient-of-determination-r2-score/