Task: House Price Prediction using linear regression

Read and Understanding the Data

importing Libraries

In [74]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
```

Importing Data and Checking out

In [75]:

```
HouseDF=pd.read_csv('Housing.csv')
```

In [76]:

```
HouseDF.head()
```

Out[76]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwate
0	13300000	7420	4	2	3	yes	no	no	
1	12250000	8960	4	4	4	yes	no	no	
2	12250000	9960	3	2	2	yes	no	yes	
3	12215000	7500	4	2	2	yes	no	yes	
4	11410000	7420	4	1	2	yes	yes	yes	

In [77]:

```
HouseDF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   price             545 non-null    int64 
 1   area              545 non-null    int64 
 2   bedrooms          545 non-null    int64 
 3   bathrooms         545 non-null    int64 
 4   stories           545 non-null    int64 
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64 
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

In [78]:

```
HouseDF.describe()
```

Out[78]:

	price	area	bedrooms	bathrooms	stories	parking
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	545.000000
mean	4.766729e+06	5150.541284	2.965138	1.286239	1.805505	0.693578
std	1.870440e+06	2170.141023	0.738064	0.502470	0.867492	0.861586
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	0.000000
25%	3.430000e+06	3600.000000	2.000000	1.000000	1.000000	0.000000
50%	4.340000e+06	4600.000000	3.000000	1.000000	2.000000	0.000000
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	1.000000
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000

In [79]:

```
HouseDF.columns
```

Out[79]:

```
Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'mainroad',
       'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
       'parking', 'prefarea', 'furnishingstatus'],
      dtype='object')
```

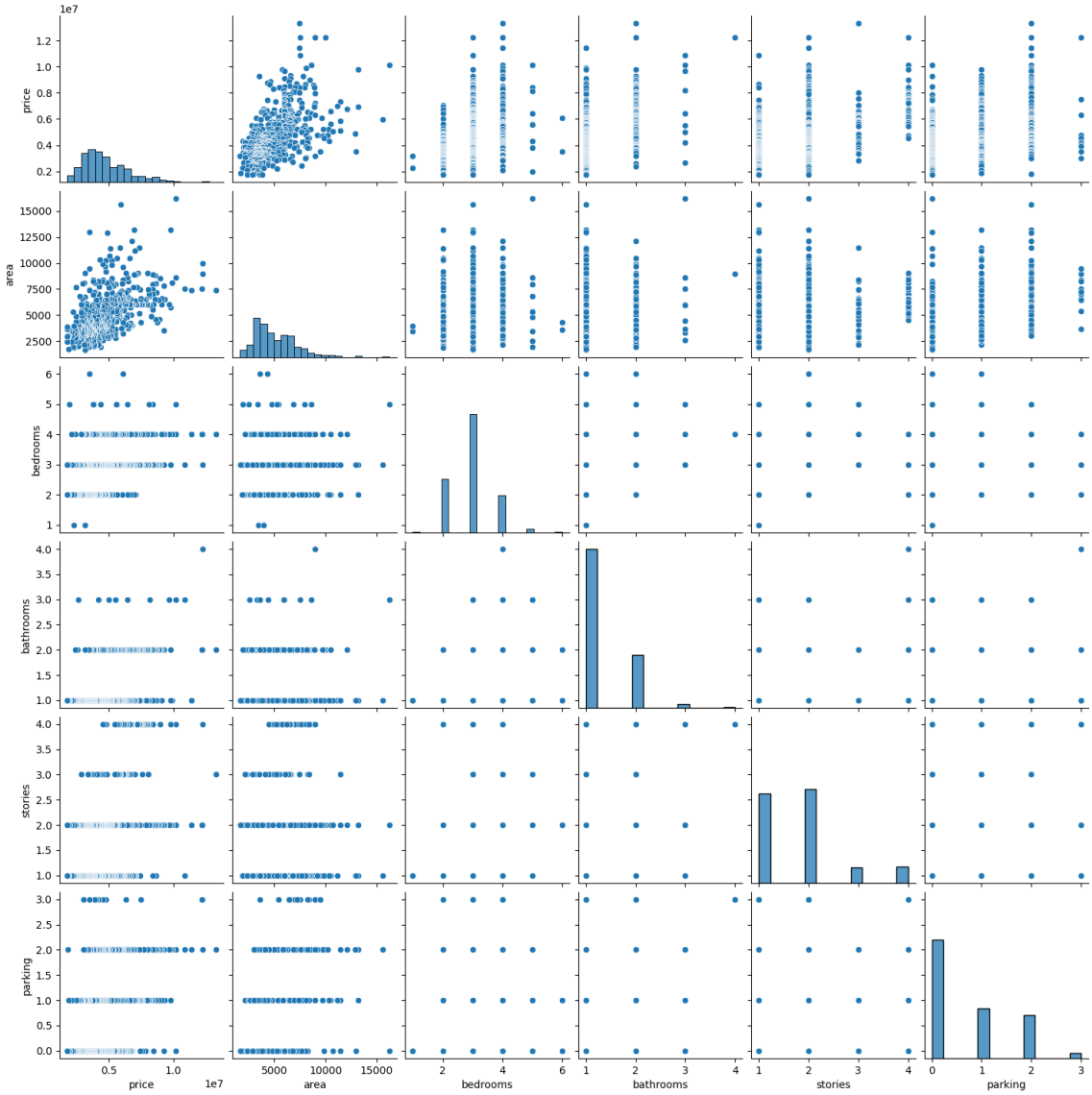
Exploratory Data Analysis

In [80]:

```
sns.pairplot(HouseDF)
```

Out[80]:

```
<seaborn.axisgrid.PairGrid at 0x120ae7aad90>
```

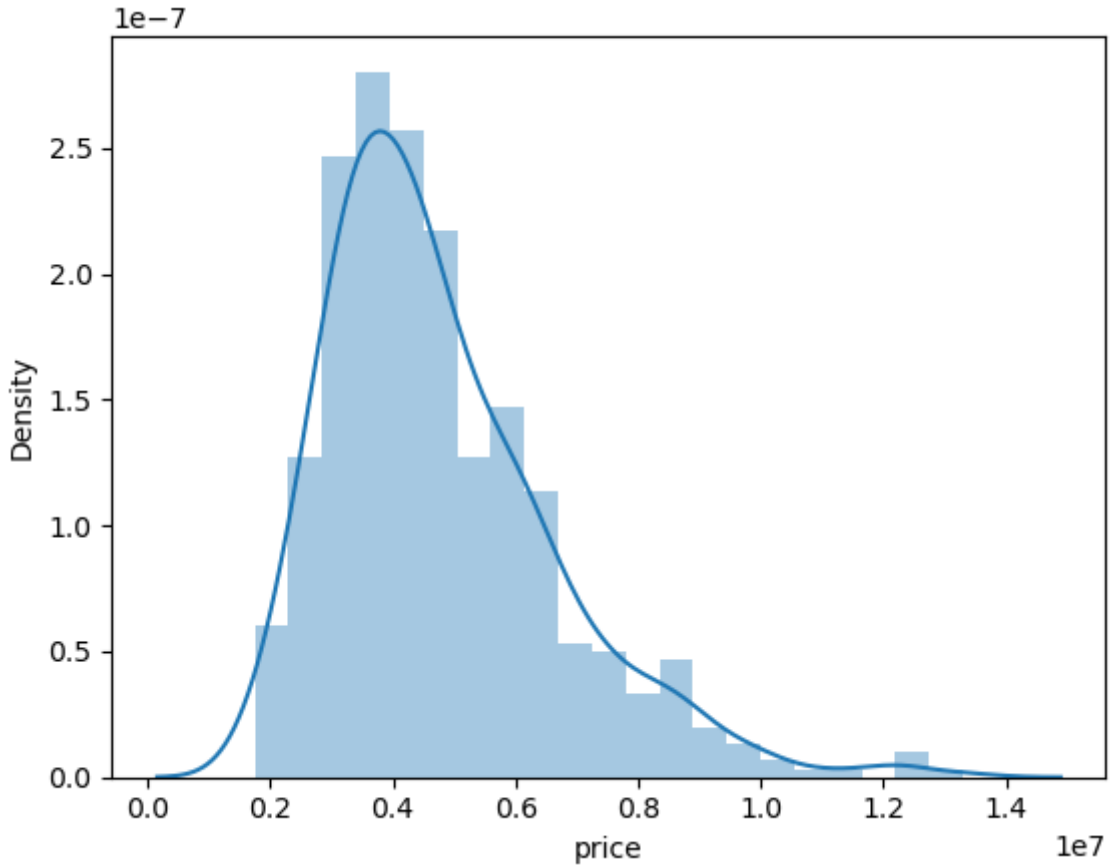


In [82]:

```
sns.distplot(HouseDF['price'])
```

Out[82]:

```
<Axes: xlabel='price', ylabel='Density'>
```

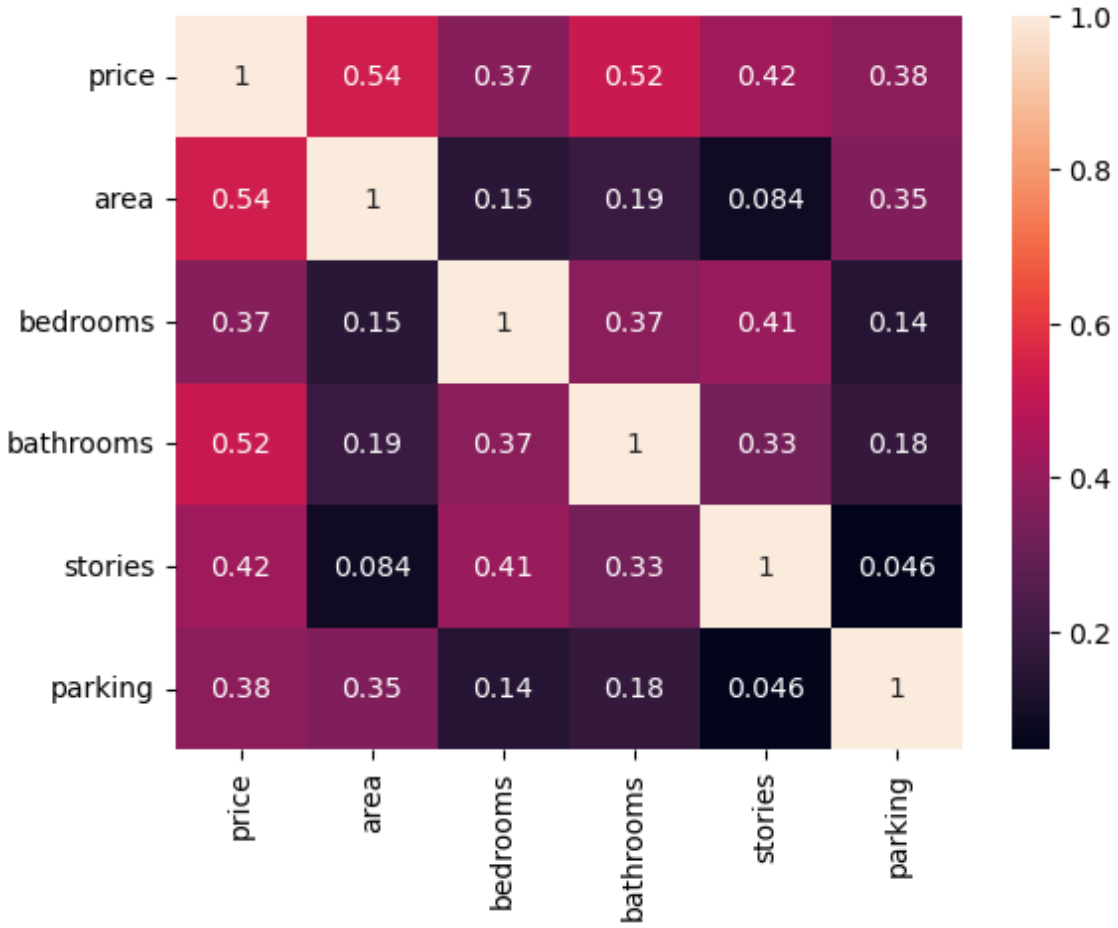


In [83]:

```
sns.heatmap(HouseDF.corr(), annot=True)
```

Out[83]:

<Axes: >



Training a Linear Regression Model

X and Y List

In [84]:

```
X=HouseDF[['area','bedrooms','bathrooms','stories','parking']]
y=HouseDF['price']
```

Split Data into Train,Test

In [85]:

```
from sklearn.model_selection import train_test_split
```

In [86]:

```
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.4, random_state=101)
```

Creating and Traing The LinearRegression Model

In [95]:

```
from sklearn.linear_model import LinearRegression
```

In [96]:

```
lm = LinearRegression()
```

In [97]:

```
lm.fit(X_train,y_train)
```

Out[97]:

LinearRegression

LinearRegression()

LinearRegression Model Evaluation

In [104]:

```
# Access the intercept
intercept = lm.intercept_
print(intercept)
```

```
-245989.43902394734
```

In [105]:

```
coeff_df = pd.DataFrame(lm.coef_,X.columns,columns=['Coefficient'])
coeff_df
```

Out[105]:

	Coefficient
area	3.492829e+02
bedrooms	1.283724e+05
bathrooms	1.232385e+06
stories	5.085921e+05
parking	4.068285e+05

Predictions from our Linear Regression Model

In [106]:

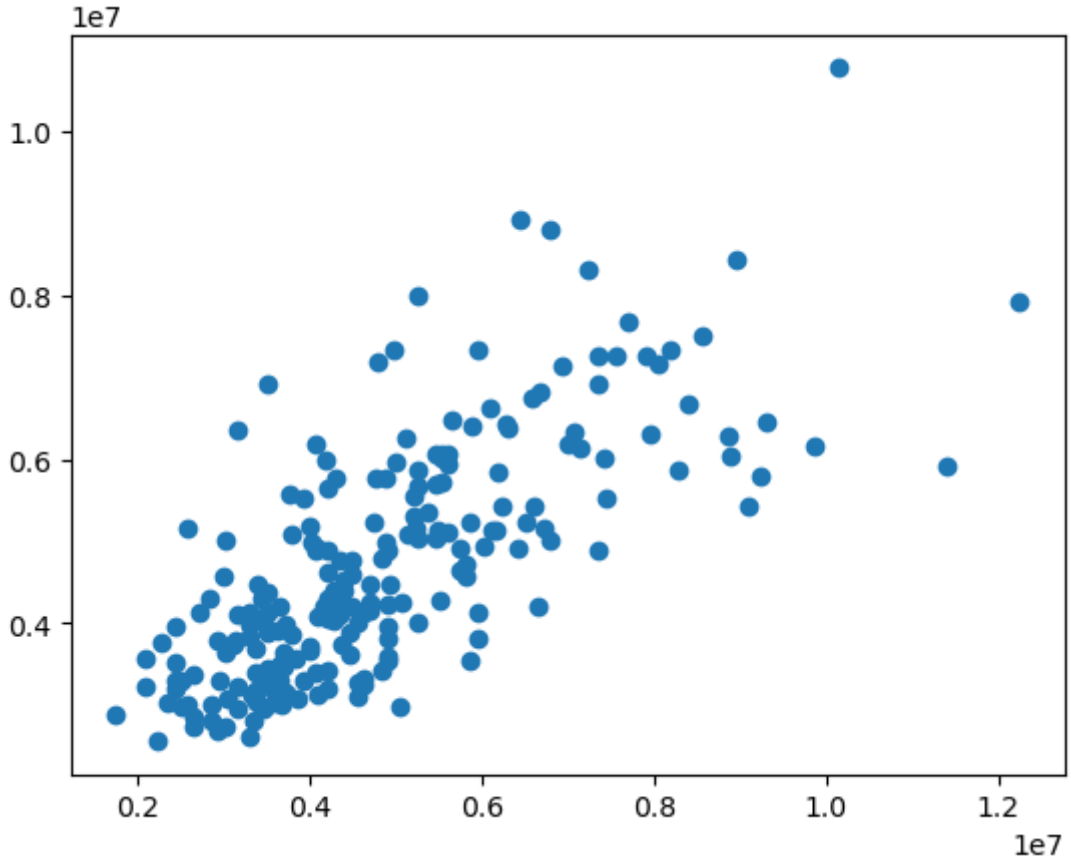
```
predictions=lm.predict(X_test)
```

In [107]:

```
plt.scatter(y_test, predictions)
```

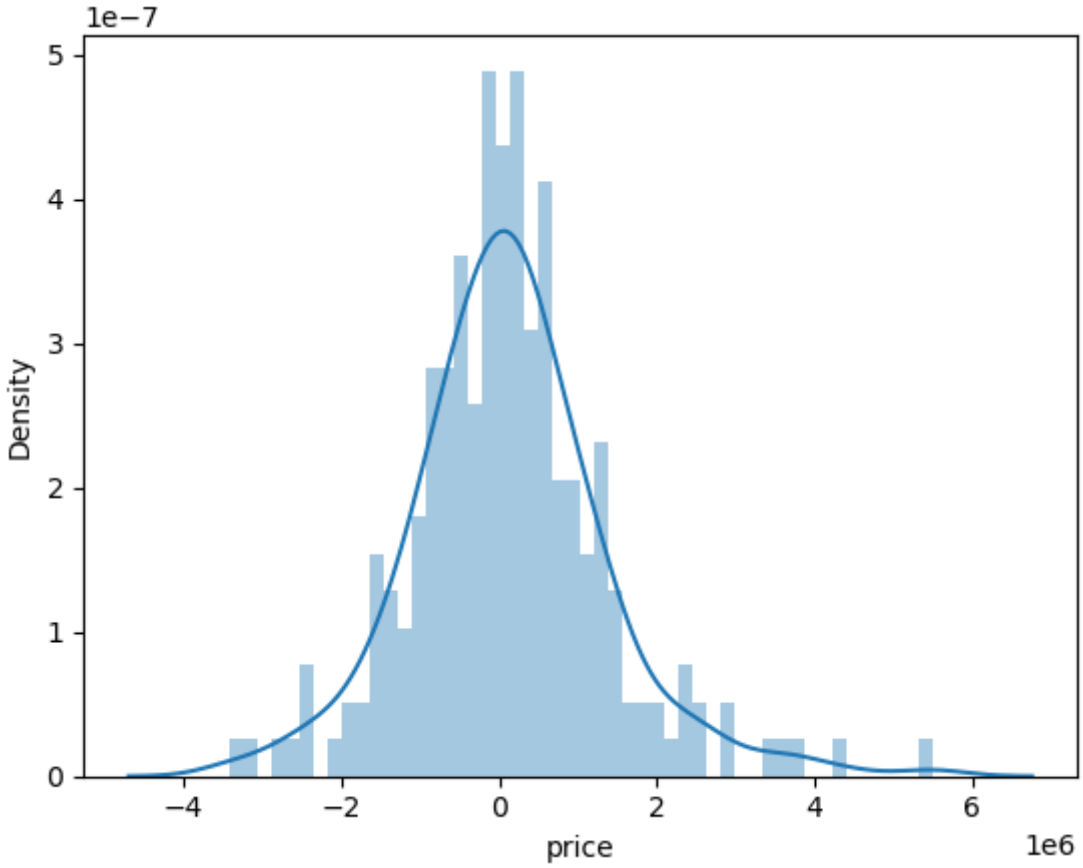
Out[107]:

```
<matplotlib.collections.PathCollection at 0x120a83c4910>
```



In [108]:

```
sns.distplot((y_test-predictions),bins=50);
```



Regression Evaluation Metrics

In [109]:

```
from sklearn import metrics
```

In [110]:

```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 900485.3566224393
MSE: 1554656659048.5005
RMSE: 1246858.7165547267
```