Housing Price Prediction(linear Rrgresssion)

Problem Statement:

Consider a real estate company that has a dataset containing the prices of properties in the Delhi region. It wishes to use the data to optimise the sale prices of the properties based on important factors such as area, bedrooms, parking, etc. Essentially, the company wants to identify the variables affecting house prices, e.g. area, number of rooms, bathrooms, etc.

To create a linear model that quantitatively relates house prices with variables such as number of rooms, area, number of bathrooms, etc.

To know the accuracy of the model, i.e. how well these variables can predict house prices.

In [1]:

```
import numpy as nd
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
#Traing and testing
import sklearn.linear_model
from sklearn.model_selection import train_test_split
#Development
from sklearn.linear_model import LinearRegression
linear_regression_model =LinearRegression()
#Evaluation
import sklearn.metrics
from sklearn.metrics import accuracy_score,r2_score
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

In [8]:

```
data =pd.read_csv("C:/Users/Oooba/Desktop/Analysis with pyhton/Housing predictive/Housing.csv")
data
```

Out[8]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	fu
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	
...	...	...	...	...	...	...	...	...	...	...	...	...	
540	1820000	3000	2	1	1	yes	no	yes	no	no	2	no	
541	1767150	2400	3	1	1	no	no	no	no	no	0	no	
542	1750000	3620	2	1	1	yes	no	no	no	no	0	no	
543	1750000	2910	3	1	1	no	no	no	no	no	0	no	
544	1750000	3850	3	1	2	yes	no	no	no	no	0	no	

545 rows × 13 columns

In [9]:

```
data.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 [10]:

```
data.isna().sum()
```

Out[10]:
```
price               0
area                0
bedrooms            0
bathrooms           0
stories             0
mainroad            0
guestroom           0
basement            0
hotwaterheating     0
airconditioning     0
parking             0
prefarea            0
furnishingstatus    0
dtype: int64
```

In [12]:
```
data.duplicated().sum()
```

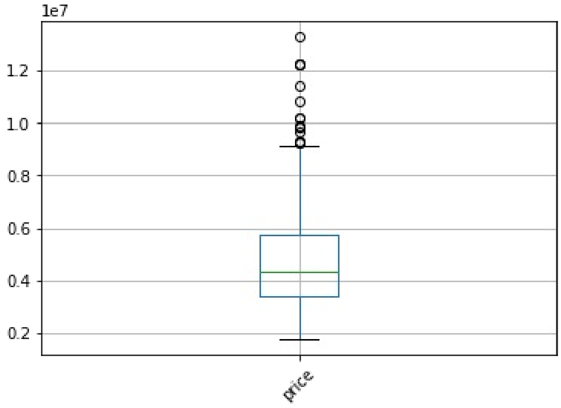
Out[12]:
```
0
```

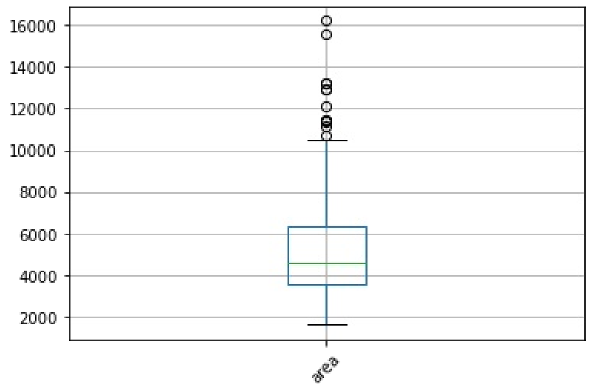
In [13]:
```
data.describe()
```

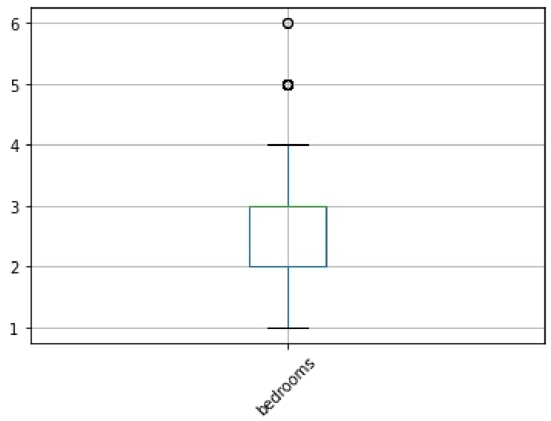
Out[13]:

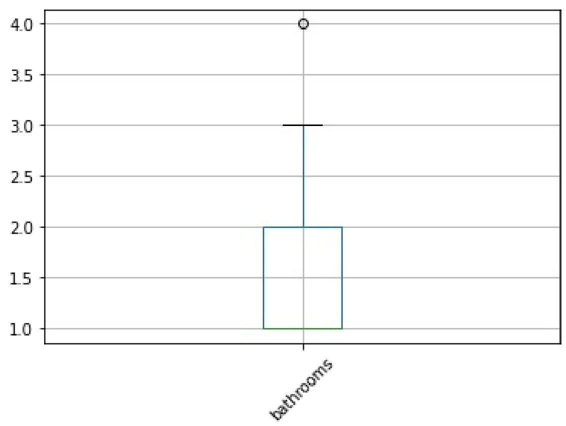
	price	area	bedrooms	bathrooms	stories	parking
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	545.000000
mean	4.766729e+06	5150.541284	2.965138	1.286239	1.805505	0.693578
std	1.870440e+06	2170.141023	0.738064	0.502470	0.867492	0.861586
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	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

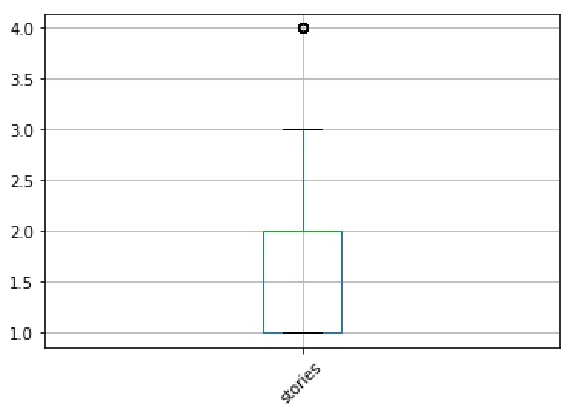
In [37]:
```
data.boxplot(column=['price'])
plt.xticks(rotation=45)
plt.show()

data.boxplot(column=['area'])
plt.xticks(rotation=45)
plt.show()

data.boxplot(column=['bedrooms'])
plt.xticks(rotation=45)
plt.show()

data.boxplot(column=['bathrooms'])
plt.xticks(rotation=45)
plt.show()

data.boxplot(column=['stories'])
plt.xticks(rotation=45)
plt.show()

data.boxplot(column=['parking'])
plt.xticks(rotation=45)
plt.show()
```

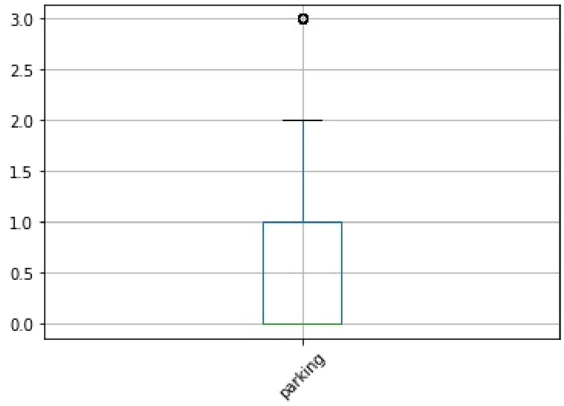



16000
14000
12000
10000
8000
6000
4000
2000
area


6
5
4
3
2
1
bedrooms


4.0
3.5
3.0
2.5
2.0
1.5
1.0
bathrooms


4.0
3.5
3.0
2.5
2.0
1.5
1.0
stories



In [44]:
```
data['mainroad'] = data['mainroad'].replace({'yes': 1, 'no': 0})
data['guestroom'] = data['guestroom'].replace({'yes': 1, 'no': 0})
data['basement'] = data['basement'].replace({'yes': 1, 'no': 0})
data['hotwaterheating'] = data['hotwaterheating'].replace({'yes': 1, 'no': 0})
data['airconditioning'] = data['airconditioning'].replace({'yes': 1, 'no': 0})
data['prefarea'] = data['prefarea'].replace({'yes': 1, 'no': 0})
data
```

Out[44]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	fu
0	13300000	7420	4	2	3	1	0	0	0	1	2	1	
1	12250000	8960	4	4	4	1	0	0	0	1	3	0	
2	12250000	9960	3	2	2	1	0	1	0	0	2	1	
3	12215000	7500	4	2	2	1	0	1	0	1	3	1	
4	11410000	7420	4	1	2	1	1	1	0	1	2	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
540	1820000	3000	2	1	1	1	0	1	0	0	2	0	
541	1767150	2400	3	1	1	0	0	0	0	0	0	0	
542	1750000	3620	2	1	1	1	0	0	0	0	0	0	
543	1750000	2910	3	1	1	0	0	0	0	0	0	0	
544	1750000	3850	3	1	2	1	0	0	0	0	0	0	

545 rows × 13 columns

In [45]:
```
data["furnishingstatus"].unique()
```

Out[45]:
```
array(['furnished', 'semi-furnished', 'unfurnished'], dtype=object)
```

In [51]:
```
data_encoded = pd.get_dummies(data, columns=['furnishingstatus'])
data_encoded
```

Out[51]:

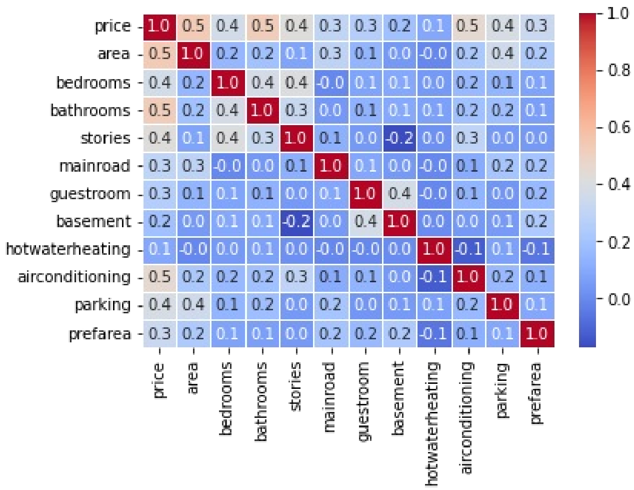
	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	fu
0	13300000	7420	4	2	3	1	0	0	0	1	2	1	
1	12250000	8960	4	4	4	1	0	0	0	1	3	0	
2	12250000	9960	3	2	2	1	0	1	0	0	2	1	
3	12215000	7500	4	2	2	1	0	1	0	1	3	1	
4	11410000	7420	4	1	2	1	1	1	0	1	2	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
540	1820000	3000	2	1	1	1	0	1	0	0	2	0	
541	1767150	2400	3	1	1	0	0	0	0	0	0	0	
542	1750000	3620	2	1	1	1	0	0	0	0	0	0	
543	1750000	2910	3	1	1	0	0	0	0	0	0	0	
544	1750000	3850	3	1	2	1	0	0	0	0	0	0	

545 rows × 15 columns

In [56]:

```
data_corr= data.corr()
color=sns.color_palette("coolwarm",as_cmap=True)
sns.heatmap(data_corr,cmap=color,annot=True,fmt="0.1f",linewidth=0.5)
```

Out[56]: <AxesSubplot:>



In [66]:

```
data_x =data_encoded.drop(columns=['price','area'])
x =data_x
y= data_encoded['price']
```

In [67]:

```
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.2,random_state=42)
x_train.shape #80%
```

Out[67]: (436, 13)

In [70]:

```
model=linear_regression_model.fit(x_train,y_train)
model.fit(x_train,y_train)
```

Out[70]: LinearRegression()

In [69]:

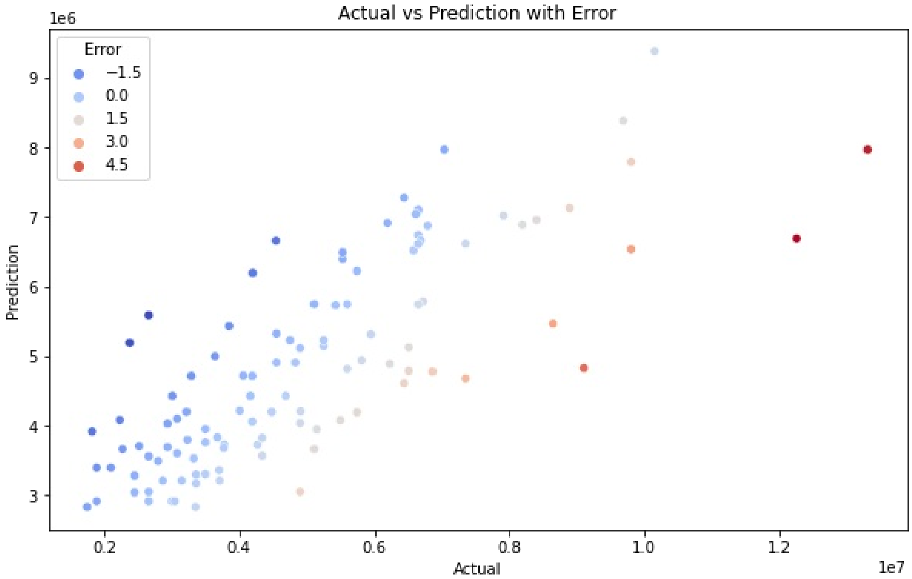
```
y_prede=model.predict(x_test)
y_error= y_test-y_prede
predection=pd.DataFrame({"Actual":y_test,"predicted":y_prede,"Error":y_error})
predection["abs_error"]=abs(predection["Error"])
mean_absolut_error=predection["abs_error"].mean()
predection.head(10)
```

Out[69]:

	Actual	predicted	Error	abs_error
316	4060000	4.718499e+06	-6.584993e+05	6.584993e+05
77	6650000	7.099163e+06	-4.491628e+05	4.491628e+05
360	3710000	3.211047e+06	4.989530e+05	4.989530e+05
90	6440000	4.608756e+06	1.831244e+06	1.831244e+06
493	2800000	3.492873e+06	-6.928734e+05	6.928734e+05
209	4900000	3.048447e+06	1.851553e+06	1.851553e+06
176	5250000	5.143756e+06	1.062444e+05	1.062444e+05
249	4543000	6.657920e+06	-2.114920e+06	2.114920e+06
516	2450000	3.039138e+06	-5.891377e+05	5.891377e+05
426	3353000	2.830353e+06	5.226470e+05	5.226470e+05

In [76]:

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Actual', y='predicted', data=predection, hue='Error', palette='coolwarm')
plt.xlabel('Actual')
plt.ylabel('Prediction')
plt.title('Actual vs Prediction with Error')
plt.legend(title='Error')
plt.show()
```



In [77]:

```
r2_score(y_test,y_prede)
print(f"Accuracy of the model={round(r2_score(y_test,y_prede)*100)}%")
```

```
Accuracy of the model=61%
```

In [78]:

```
print("Root Mean Squared Error (RMSE)=",mean_absolut_error**(0.5))
```

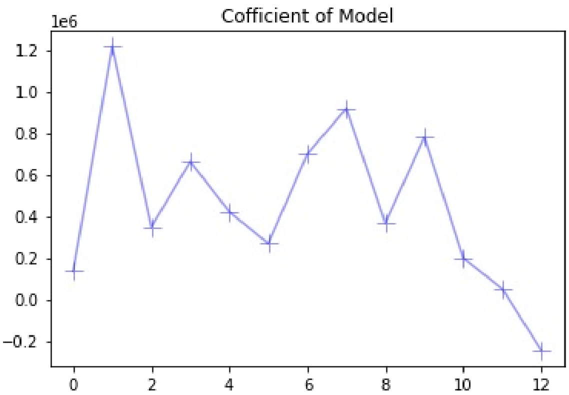
```
Root Mean Squared Error (RMSE)= 999.9353535607933
```

In [85]:

```
model_cof=model.coef_
plt.plot(model_cof,color="b",marker="+",markersize=12,alpha=0.4)
plt.title("Cofficient of Model")
```

Out[85]:

```
Text(0.5, 1.0, 'Cofficient of Model')
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js