In [35]:
```
import pandas as pd
import numpy as nm
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.linear_model import LinearRegression
from sklearn import svm
from sklearn.svm import SVC
```

In [36]:
```
dataset=pd.read_csv('Houseprice.csv')
```

In [37]:
```
dataset
```

Out[37]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	c
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	
...	...	...	...	...	...	...	...	...	...	...
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	
4597	2014-07-09 00:00:00	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0	
4598	2014-07-10 00:00:00	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0	
4599	2014-07-10 00:00:00	2.206000e+05	3.0	2.50	1490	8102	2.0	0	0	

4600 rows × 18 columns

In [38]:
```
dataset.head()
```

Out[38]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	conditio
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	

In [39]:
```
dataset.tail()
```

Out[39]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view
4595	2014-07-09 00:00:00	308166.666667	3.0	1.75	1510	6360	1.0	0	0
4596	2014-07-09 00:00:00	534333.333333	3.0	2.50	1460	7573	2.0	0	0
4597	2014-07-09 00:00:00	416904.166667	3.0	2.50	3010	7014	2.0	0	0
4598	2014-07-10 00:00:00	203400.000000	4.0	2.00	2090	6630	1.0	0	0
4599	2014-07-10 00:00:00	220600.000000	3.0	2.50	1490	8102	2.0	0	0

In [40]:
```
dataset.sample(2)
```

Out[40]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condit
2676	2014-06-17 00:00:00	506000.0	3.0	1.75	2180	7700	1.0	0	0	
2212	2014-06-10 00:00:00	165000.0	3.0	1.00	970	6600	1.0	0	0	

In [41]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   date           4600 non-null   object 
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   float64
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4600 non-null   int64  
 5   sqft_lot       4600 non-null   int64  
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64  
 8   view           4600 non-null   int64  
 9   condition      4600 non-null   int64  
 10  sqft_above     4600 non-null   int64  
 11  sqft_basement  4600 non-null   int64  
 12  yr_built       4600 non-null   int64  
 13  yr_renovated   4600 non-null   int64  
 14  street         4600 non-null   object 
 15  city           4600 non-null   object 
 16  statezip       4600 non-null   object 
 17  country        4600 non-null   object 
dtypes: float64(4), int64(9), object(5)
memory usage: 647.0+ KB
```

In [42]:
```
dataset.describe()
```

Out[42]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
count	4.600000e+03	4600.000000	4600.000000	4600.000000	4.600000e+03	4600.000000	4600.000000
mean	5.519630e+05	3.400870	2.160815	2139.346957	1.485252e+04	1.512065	0.007174
std	5.638347e+05	0.908848	0.783781	963.206916	3.588444e+04	0.538288	0.084404
min	0.000000e+00	0.000000	0.000000	370.000000	6.380000e+02	1.000000	0.000000
25%	3.228750e+05	3.000000	1.750000	1460.000000	5.000750e+03	1.000000	0.000000
50%	4.609435e+05	3.000000	2.250000	1980.000000	7.683000e+03	1.500000	0.000000
75%	6.549625e+05	4.000000	2.500000	2620.000000	1.100125e+04	2.000000	0.000000
max	2.659000e+07	9.000000	8.000000	13540.000000	1.074218e+06	3.500000	1.000000

In [43]:
```
dataset.shape
```

Out[43]:
```
(4600, 18)
```

In [44]:
```
dataset.isnull()
```

Out[44]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	s
0	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	
4595	False	False	False	False	False	False	False	False	False	False	
4596	False	False	False	False	False	False	False	False	False	False	
4597	False	False	False	False	False	False	False	False	False	False	
4598	False	False	False	False	False	False	False	False	False	False	
4599	False	False	False	False	False	False	False	False	False	False	

4600 rows × 18 columns

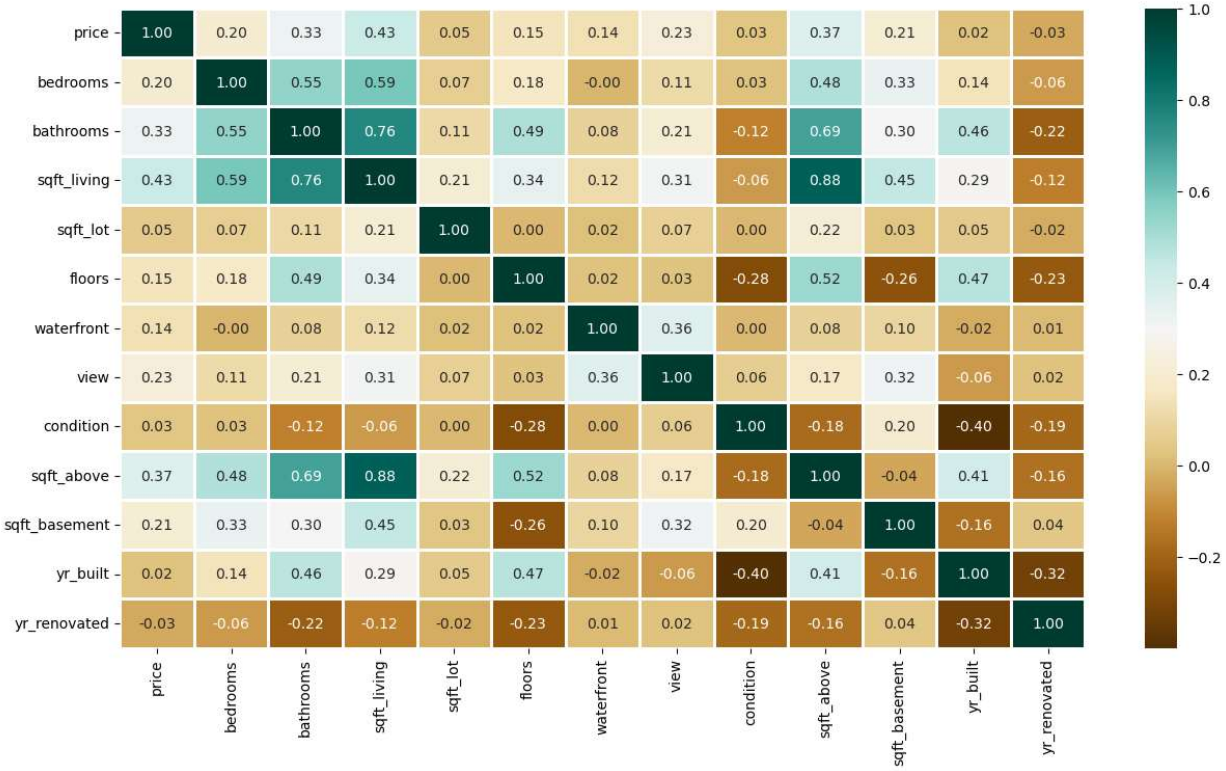
In [45]:
```
obj = (dataset.dtypes == 'object')
object_cols = list(obj[obj].index)
print("Categorical variables:",len(object_cols))

int_ = (dataset.dtypes == 'int')
num_cols = list(int_[int_].index)
print("Integer variables:",len(num_cols))
```

```
fl = (dataset.dtypes == 'float')
fl_cols = list(fl[fl].index)
print("Float variables:",len(fl_cols))
```

```
Categorical variables: 5
Integer variables: 0
Float variables: 4
```

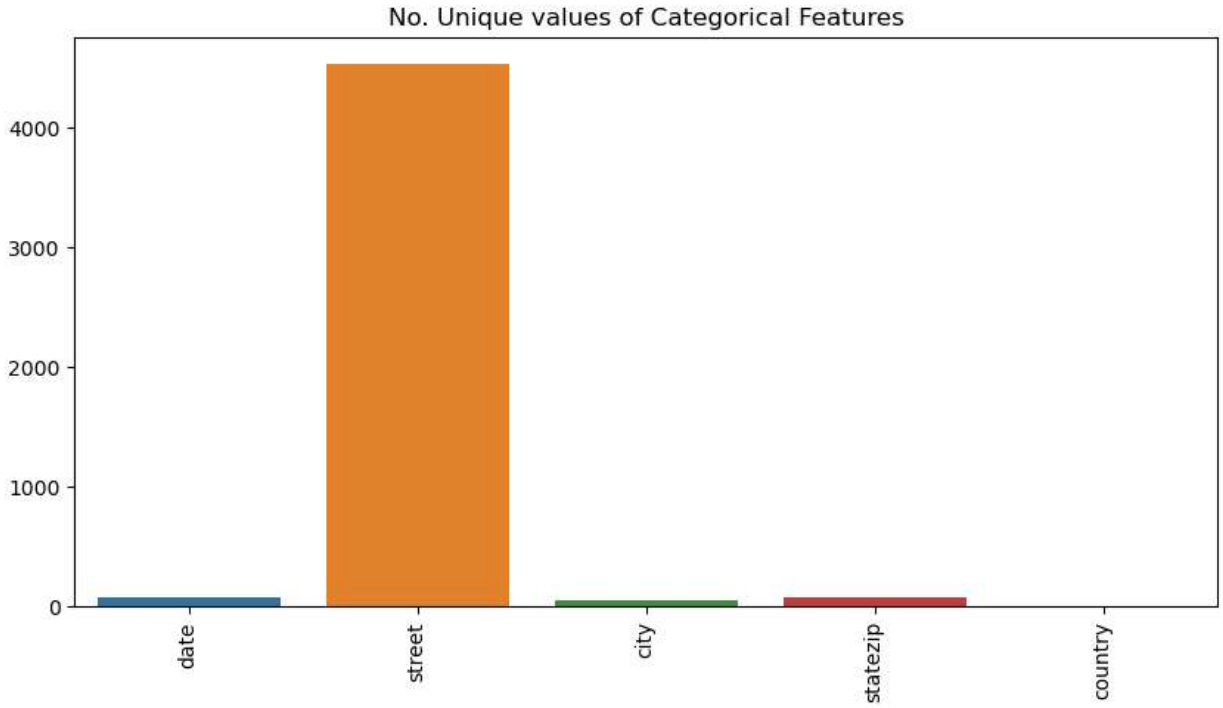
In [46]:
```
plt.figure(figsize=(15, 8
                    ))
sns.heatmap(dataset.corr(),
            cmap = 'BrBG',
            fmt = '.2f',
            linewidths = 2,
            annot = True)
```

Out[46]: <AxesSubplot:>



In [48]:
```
unique_values = []
for col in object_cols:
  unique_values.append(dataset[col].unique().size)
plt.figure(figsize=(10,5))
plt.title('No. Unique values of Categorical Features')
plt.xticks(rotation=90)
sns.barplot(x=object_cols,y=unique_values)
```

Out[48]: <AxesSubplot:title={'center':'No. Unique values of Categorical Features'}>



In [32]:
```
dataset.drop(['date'],
             axis=1,
             inplace=True)
```

In [49]:
```
dataset['price'] = dataset['price'].fillna(
  dataset['price'].mean())
```

In [50]:
```
new_dataset = dataset.dropna()
```

In [51]:
```
new_dataset
```

Out[51]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	c
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	
...	...	...	...	...	...	...	...	...	...	...
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	
4597	2014-07-09 00:00:00	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0	
4598	2014-07-10 00:00:00	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0	
4599	2014-07-10 00:00:00	2.206000e+05	3.0	2.50	1490	8102	2.0	0	0	

4600 rows × 18 columns

In [52]:
```
new_dataset.isnull().sum()
```

Out[52]:
```
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
dtype: int64
```

In [53]:
```
from sklearn.preprocessing import OneHotEncoder
s = (new_dataset.dtypes == 'object')
object_cols = list(s[s].index)
print("Categorical variables:")
print(object_cols)
print('No. of. categorical features: ',
      len(object_cols))
```

```
Categorical variables:
['date', 'street', 'city', 'statezip', 'country']
No. of. categorical features:  5
```

In [55]:
```
OH_encoder = OneHotEncoder(sparse=False)
OH_cols = pd.DataFrame(OH_encoder.fit_transform(new_dataset[object_cols]))
OH_cols.index = new_dataset.index
OH_cols.columns = OH_encoder.get_feature_names()
df_final = new_dataset.drop(object_cols, axis=1)
df_final = pd.concat([df_final, OH_cols], axis=1)
```

```
C:\Users\reddy\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWar
ning: Function get_feature_names is deprecated; get_feature_names is deprecated in 1.
0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

In [57]:
```
X = df_final.drop(['price'], axis=1)
Y = df_final['price']
```

In [58]:
```
X
```

Out[58]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sc
0	3.0	1.50	1340	7912	1.5	0	0	3	1340	
1	5.0	2.50	3650	9050	2.0	0	4	5	3370	
2	3.0	2.00	1930	11947	1.0	0	0	4	1930	
3	3.0	2.25	2000	8030	1.0	0	0	4	1000	
4	4.0	2.50	1940	10500	1.0	0	0	4	1140	
...	...	...	...	...	...	...	...	...	...	...
4595	3.0	1.75	1510	6360	1.0	0	0	4	1510	
4596	3.0	2.50	1460	7573	2.0	0	0	3	1460	
4597	3.0	2.50	3010	7014	2.0	0	0	3	3010	
4598	4.0	2.00	2090	6630	1.0	0	0	3	1070	
4599	3.0	2.50	1490	8102	2.0	0	0	4	1490	

4600 rows × 4729 columns

In [59]:
```
Y
```

Out[59]:
```
0       3.130000e+05
1       2.384000e+06
2       3.420000e+05
3       4.200000e+05
4       5.500000e+05
            ...
4595    3.081667e+05
4596    5.343333e+05
4597    4.169042e+05
4598    2.034000e+05
4599    2.206000e+05
Name: price, Length: 4600, dtype: float64
```

In [62]:
```
Features1 = ['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors']
target = 'price'
X1 = dataset[Features1]
y1 = dataset[target]
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.5, random_stat
```

In [63]:
```
X1
```

Out[63]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors
0	3.0	1.50	1340	7912	1.5
1	5.0	2.50	3650	9050	2.0
2	3.0	2.00	1930	11947	1.0
3	3.0	2.25	2000	8030	1.0
4	4.0	2.50	1940	10500	1.0
...	...	...	...	...	...
4595	3.0	1.75	1510	6360	1.0
4596	3.0	2.50	1460	7573	2.0
4597	3.0	2.50	3010	7014	2.0
4598	4.0	2.00	2090	6630	1.0
4599	3.0	2.50	1490	8102	2.0

4600 rows × 5 columns

In [64]:
```
y1
```

Out[64]:
```
0       3.130000e+05
1       2.384000e+06
2       3.420000e+05
3       4.200000e+05
4       5.500000e+05
            ...
4595    3.081667e+05
4596    5.343333e+05
4597    4.169042e+05
4598    2.034000e+05
4599    2.206000e+05
Name: price, Length: 4600, dtype: float64
```

In [65]:
```
lr = LinearRegression()
```

In [66]:
```
lr.fit(X_train, y_train)
```

Out[66]:
```
LinearRegression()
```

In [67]:
```
y_pred = lr.predict(X_test)
```

In [68]:
```
y_pred
```

Out[68]:
```
array([363715.81566704, 399446.44586553, 834230.23979669, ...,
       748123.84167885, 569569.52453238, 658083.41823349])
```

In [69]:
```
score = lr.score(X_test, y_test)
print("lr R^2 Score:", score)
```

```
lr R^2 Score: 0.10933671026238057
```

In [70]:
```
new_house = pd.DataFrame({'bedrooms': [2], 'bathrooms': [2.5], 'sqft_living': [600], '
predicted_price = lr.predict(new_house)
```

```
print("Predicted Price:", predicted_price[0])
```

```
Predicted Price: 146977.00302329706
```

In [ ]: