nm-assignment-houseprice-1

October 30, 2023

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
# Load Data
data = pd.read_csv("/content/House Price India.csv")
data.shape
```

[3]: (14620, 23)

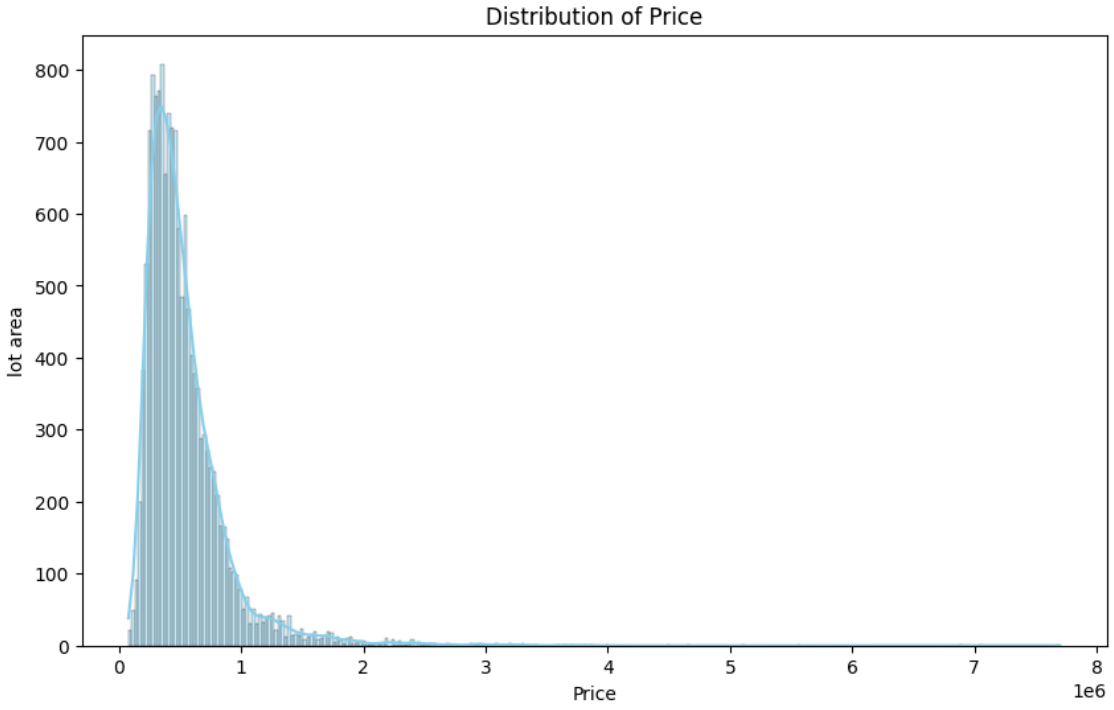
[5]:
```
data.head()
```

[5]:

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	…	Built Year	Renovation Year	Postal Code	Lattitude	Longitude	living_area_renov	lot_area_renov
0	6762810145	42491	5	2.50	3650	9050	2.0	0	4	5	…	1921	0	122003	52.8645	-114.557	2880	5400
1	6762810635	42491	4	2.50	2920	4000	1.5	0	0	5	…	1909	0	122004	52.8878	-114.470	2470	4000
2	6762810998	42491	5	2.75	2910	9480	1.5	0	0	3	…	1939	0	122004	52.8852	-114.468	2940	6600
3	6762812605	42491	4	2.50	3310	42998	2.0	0	0	3	…	2001	0	122005				
4	6762812919	42491	3	2.00	2710	4500	1.5	0	0	4	…	1929	0	122006				

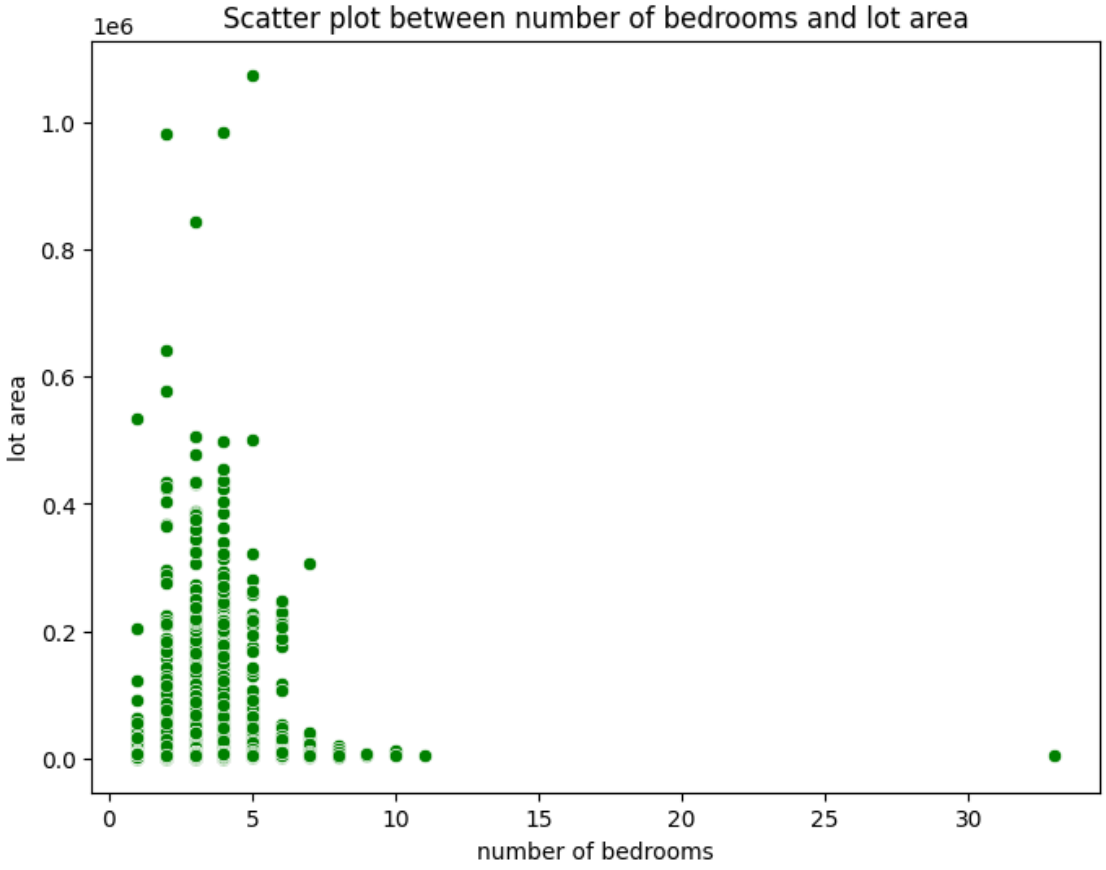
```
3    52.9532   -114.321               3350           42847
4    52.9047   -114.485               2060            4500

   Number of schools nearby  Distance from the airport    Price
0                         2                         58  2380000
1                         2                         51  1400000
2                         1                         53  1200000
3                         3                         76   838000
4                         1                         51   805000

[5 rows x 23 columns]
```

[6]:
```
# Task 3: Visualizations
# Univariate Analysis
# Example: Histogram for a single variable 'Price'
plt.figure(figsize=(10, 6))
sns.histplot(data['Price'], kde=True, color='skyblue')
plt.title('Distribution of Price')
plt.xlabel('Price')
plt.ylabel('lot area')
plt.show()
```

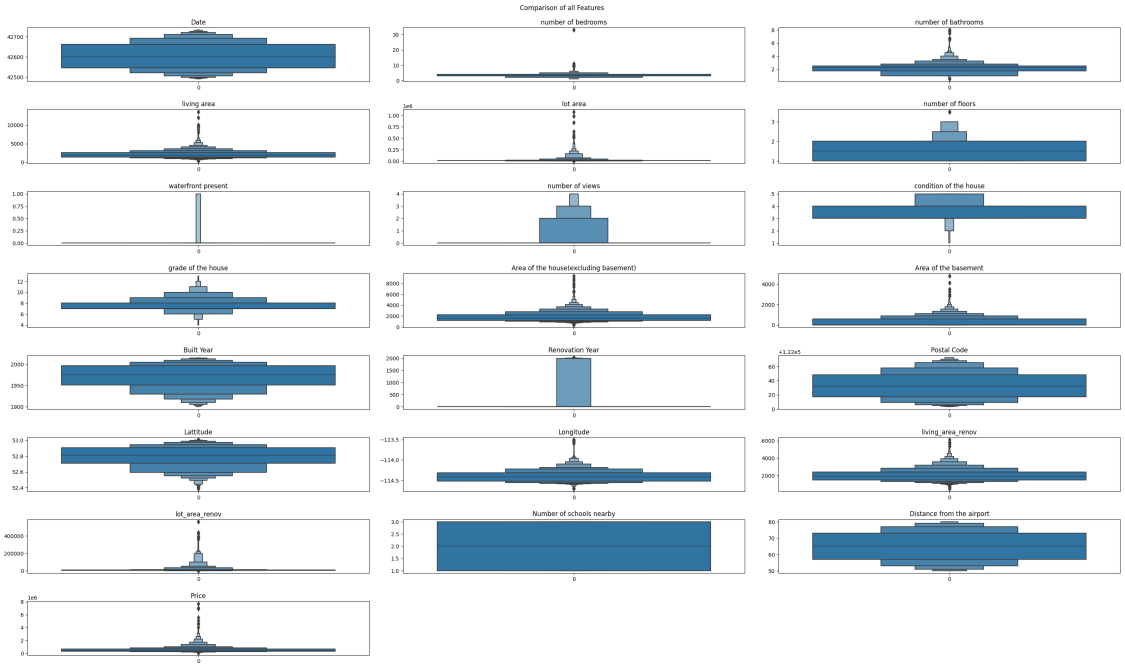


[7]:
```
# Bi-Variate Analysis
# Example: Scatter plot between 'GrLivArea' and 'SalePrice'
```

```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='number of bedrooms', y='lot area', data=data, color='green')
plt.title('Scatter plot between number of bedrooms and lot area')
plt.xlabel('number of bedrooms')
plt.ylabel('lot area')
plt.show()
```



[8]:
```
# Multivariate Analysis
 #Boxplot
fig = plt.figure(figsize=[30, 18])
for i in range(1, len(list(data.columns))):
    plt.subplot(8, 3, i)
    fig.tight_layout(pad=2)
    df_new = data.iloc[:, i]
    plt.suptitle('Comparison of all Features')
    plt.title(list(data.columns)[i])
    sns.boxenplot(df_new)

plt.show()
```



[ ]:
```
# Task 4: Descriptive statistics
descriptive_stats = data.describe()
print(descriptive_stats)
```

```
                 id          Date  number of bedrooms  number of bathrooms  \
count  1.462000e+04  14620.000000        14620.000000         14620.000000
mean   6.762821e+09  42604.538646            3.379343             2.129583
std    6.237575e+03     67.347991            0.938719             0.769934
min    6.762810e+09  42491.000000            1.000000             0.500000
25%    6.762815e+09  42546.000000            3.000000             1.750000
50%    6.762821e+09  42600.000000            3.000000             2.250000
75%    6.762826e+09  42662.000000            4.000000             2.500000
max    6.762832e+09  42734.000000           33.000000             8.000000

        living area      lot area  number of floors  waterfront present  \
count  14620.000000  1.462000e+04      14620.000000        14620.000000
mean    2098.262996  1.509328e+04          1.502360            0.007661
std      928.275721  3.791962e+04          0.540239            0.087193
min      370.000000  5.200000e+02          1.000000            0.000000
25%     1440.000000  5.010750e+03          1.000000            0.000000
50%     1930.000000  7.620000e+03          1.500000            0.000000
75%     2570.000000  1.080000e+04          2.000000            0.000000
max    13540.000000  1.074218e+06          3.500000            1.000000

       number of views  condition of the house  …    Built Year  \
count     14620.000000            14620.000000  …  14620.000000
```

```
mean          0.233105               3.430506  …   1970.926402
std           0.766259               0.664151  …     29.493625
min           0.000000               1.000000  …   1900.000000
25%           0.000000               3.000000  …   1951.000000
50%           0.000000               3.000000  …   1975.000000
75%           0.000000               4.000000  …   1997.000000
max           4.000000               5.000000  …   2015.000000

       Renovation Year    Postal Code     Lattitude     Longitude  \
count     14620.000000   14620.000000  14620.000000  14620.000000
mean         90.924008  122033.062244     52.792848   -114.404007
std         416.216661      19.082418      0.137522      0.141326
min           0.000000  122003.000000     52.385900   -114.709000
25%           0.000000  122017.000000     52.707600   -114.519000
50%           0.000000  122032.000000     52.806400   -114.421000
75%           0.000000  122048.000000     52.908900   -114.315000
max        2015.000000  122072.000000     53.007600   -113.505000

       living_area_renov  lot_area_renov  Number of schools nearby  \
count       14620.000000    14620.000000                14620.000000
mean         1996.702257    12753.500068                    2.012244
std           691.093366    26058.414467                    0.817284
min           460.000000      651.000000                    1.000000
25%          1490.000000     5097.750000                    1.000000
50%          1850.000000     7620.000000                    2.000000
75%          2380.000000    10125.000000                    3.000000
max          6110.000000   560617.000000                    3.000000

       Distance from the airport         Price
count                14620.000000  1.462000e+04
mean                    64.950958  5.389322e+05
std                      8.936008  3.675324e+05
min                     50.000000  7.800000e+04
25%                     57.000000  3.200000e+05
50%                     65.000000  4.500000e+05
75%                     73.000000  6.450000e+05
max                     80.000000  7.700000e+06

[8 rows x 23 columns]
```

[10]:
```
# Task 5: Handling missing values
# You can handle missing values by using functions like fillna(), dropna(), or␣
 ↪other appropriate methods based on the nature of missing data.
# Example: Filling missing values with the mean
data.fillna(data.mean(), inplace=True)
```

[11]:
```
# Dropping rows with missing values
data.dropna(inplace=True)
```

[12]:
```
# Filling missing values with mean
data.fillna(data.mean(), inplace=True)

# Filling missing values with median
data.fillna(data.median(), inplace=True)

# Filling missing values with mode
data.fillna(data.mode().iloc[0], inplace=True)
```

[13]:
```
# Using linear interpolation
data.interpolate(method='linear', inplace=True)
```

[14]:
```
from sklearn.impute import SimpleImputer

# Create an imputer object with a strategy (mean, median, most_frequent,␣
 ↪constant)
imputer = SimpleImputer(strategy='mean')

# Fit the imputer on the data
data_imputed = imputer.fit_transform(data)
```

[ ]:
```

```