House_Price_Prediction

November 12, 2025

1 Import the necessary libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_recall_fscore_support, accuracy_score
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix


import warnings
warnings.filterwarnings("ignore")
```

[2]:
```
# Read the CSV File
df = pd.read_csv(r"/content/Bengaluru_House_Data.csv")
```

[3]:
```
df.head()
```

[3]:
```
              area_type   availability                  location       size  \
0  Super built-up  Area         19-Dec  Electronic City Phase II      2 BHK
1            Plot  Area  Ready To Move          Chikka Tirupathi  4 Bedroom
2        Built-up  Area  Ready To Move               Uttarahalli      3 BHK
3  Super built-up  Area  Ready To Move        Lingadheeranahalli      3 BHK
4  Super built-up  Area  Ready To Move                  Kothanur      2 BHK
```

```
   society total_sqft  bath  balcony   price
0  Coomee         1056   2.0      1.0   39.07
1  Theanmp        2600   5.0      3.0  120.00
2      NaN        1440   2.0      3.0   62.00
3  Soiewre        1521   3.0      1.0   95.00
4      NaN        1200   2.0      1.0   51.00
```

[4]:
```python
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

[5]:
```python
df['size'].value_counts()
```

[5]:
```
size
2 BHK         5199
3 BHK         4310
4 Bedroom      826
4 BHK          591
3 Bedroom      547
1 BHK          538
2 Bedroom      329
5 Bedroom      297
6 Bedroom      191
1 Bedroom      105
8 Bedroom       84
7 Bedroom       83
5 BHK           59
9 Bedroom       46
6 BHK           30
7 BHK           17
1 RK            13
10 Bedroom      12
9 BHK            8
8 BHK            5
11 BHK           2
10 BHK           2
11 Bedroom       2
27 BHK           1
19 BHK           1
43 Bedroom       1
16 BHK           1
14 BHK           1
12 Bedroom       1
13 BHK           1
18 Bedroom       1
Name: count, dtype: int64
```

[6]:
```python
df['bath'].value_counts()
```

[6]:
```
bath
2.0     6908
3.0     3286
4.0     1226
1.0      788
5.0      524
6.0      273
7.0      102
8.0       64
9.0       43
10.0      13
12.0       7
11.0       3
13.0       3
16.0       2
27.0       1
14.0       1
40.0       1
15.0       1
18.0       1
Name: count, dtype: int64
```

[7]:
```
df['balcony'].value_counts()
```

[7]:
```
balcony
2.0    5113
1.0    4897
3.0    1672
0.0    1029
Name: count, dtype: int64
```

[8]:
```
df['total_sqft'].value_counts()
```

[8]:
```
total_sqft
1200                     843
1100                     221
1500                     205
2400                     196
600                      180
1000                     172
1350                     133
1050                     123
1300                     117
1250                     114
900                      112
1400                     108
1800                     104
```

1150	101
1600	101
1140	91
2000	83
1450	70
1650	69
800	67
3000	66
1075	66
1020	63
2500	62
1160	60
1125	60
1550	60
950	59
1700	58
1180	58
1260	57
1255	56
1080	55
1220	55
1070	53
750	52
700	52
4000	48
1175	48
1225	48
1320	46
1240	46
2100	46
1230	45
1060	45
1210	44
850	43
1280	42
1185	41
1270	41
1410	40
1190	40
1170	40
1750	39
1025	38
1330	38
1290	37
1850	37
1310	37
1194	36

1065	36
1215	35
1090	35
500	34
1360	33
1115	33
2700	33
1464	32
1120	32
1900	32
3500	32
1205	31
2200	31
1340	31
1530	31
1430	31
1035	30
1560	30
1165	30
1130	29
1128	29
1145	29
3600	29
1275	28
1355	28
2800	28
1040	28
1105	27
1155	27
1420	27
1680	27
650	25
1245	25
1590	25
1216	25
1460	25
1760	25
1010	24
2600	24
1305	24
1475	24
1030	23
1575	23
1440	23
883	23
1110	23
1246	22

1495	22
985	22
1610	21
1370	21
1385	21
5000	21
1470	21
525	21
1325	21
1243	21
1027	21
1315	21
3200	21
1015	21
660	20
925	20
1480	20
550	20
1540	20
1390	20
1570	19
1265	19
1640	19
1365	19
1645	19
1520	19
920	19
1665	18
1485	18
975	18
1012	18
1820	18
940	18
1345	18
2072	18
1195	18
1525	18
1095	18
1196	18
4800	18
980	18
960	17
1375	17
400	17
1232	17
1295	17
1157	17

1720	16
1830	16
935	16
1425	16
3300	16
2300	16
1045	16
1135	16
1655	16
645	16
1235	16
1418	16
1490	16
1197	16
1660	15
1740	15
1153	15
1152	15
1285	15
1116	15
1380	15
4500	15
1950	15
1920	15
1580	15
630	15
1085	15
1630	14
905	14
1445	14
1595	14
720	14
675	14
450	14
1141	14
2250	13
1730	13
1005	13
1625	13
3900	13
970	13
1282	13
1314	13
1510	13
1535	13
840	13
1710	13

1082	13
1690	13
2215	13
1615	13
1875	13
3800	13
1151	12
1007	12
1056	12
1435	12
1404	12
3100	12
1890	12
1339	12
1691	12
1843	12
1296	12
1033	12
1455	12
1252	12
1565	12
2900	12
1405	11
1639	11
540	11
1162	11
1804	11
1174	11
1206	11
1346	11
2150	11
1685	11
1452	11
984	11
965	11
845	11
1515	11
1620	10
957	10
760	10
620	10
880	10
930	10
3400	10
995	10
4200	10
1555	10

1256	10
1161	10
3596	10
1308	10
1198	10
2350	10
918	10
1810	10
1465	10
1063	10
770	10
1415	10
1482	9
1156	9
1223	9
1047	9
1693	9
890	9
910	9
674	9
1092	9
3750	9
1186	9
705	9
1725	9
780	9
1605	9
6000	9
1108	9
2750	9
825	9
1715	9
3150	9
2790	9
945	9
3250	9
1756	9
1870	9
1224	9
1724	9
1419	9
2475	9
1222	9
1113	9
1790	9
2180	9
1745	9

1146	8
2145	8
1395	8
1476	8
812	8
2650	8
1532	8
1703	8
4400	8
1178	8
1163	8
990	8
870	8
3122	8
1192	8
1835	8
1089	8
2990	8
1670	8
1352	8
1299	8
1133	8
1026	8
5400	8
1187	8
656	8
1322	8
1221	8
1636	8
1088	8
1705	8
1447	8
875	8
1335	8
680	8
1044	7
1272	7
1254	7
615	7
1840	7
1041	7
1910	7
1427	7
1276	7
708	7
1093	7
1357	7

1718	7
1139	7
1545	7
1251	7
1508	7
1132	7
1262	7
1397	7
1374	7
1036	7
1277	7
1069	7
1453	7
1183	7
1112	7
982	7
1166	7
1199	7
7500	7
1342	7
1349	7
1847	7
1392	7
1101	7
1306	7
1564	7
1762	7
1096	7
2050	7
820	7
1917	7
1179	7
1143	7
1880	7
2010	6
1333	6
1016	6
1755	6
2425	6
1176	6
2280	6
1411	6
520	6
2275	6
1787	6
1785	6
1541	6

654	6
1181	6
610	6
1512	6
1602	6
1274	6
1313	6
936	6
1303	6
1268	6
3895	6
2774	6
1697	6
1257	6
1247	6
1127	6
1126	6
1484	6
860	6
1362	6
1451	6
2480	6
1770	6
1635	6
1719	6
1084	6
830	6
1571	6
1893	6
2225	6
1780	6
1102	6
1286	6
1019	6
1675	6
605	6
1107	6
1167	6
1213	6
440	6
1278	6
1583	6
1614	6
7000	6
810	6
1855	6
1083	6

1717	6
1304	6
2040	6
625	6
560	6
1991	6
1856	6
929	6
933	6
1846	6
1242	6
710	6
1735	6
1028	6
1936	6
1656	6
1053	6
530	6
2690	6
1173	6
924	6
1372	6
1062	6
1444	5
1930	5
1226	5
1293	5
1825	5
410	5
2254	5
1798	5
1297	5
1263	5
1309	5
1188	5
907	5
1168	5
1884	5
1344	5
2360	5
4395	5
1925	5
510	5
635	5
1738	5
1765	5
919	5

1358	5
1052	5
2850	5
1231	5
915	5
3450	5
1204	5
1852	5
640	5
1749	5
1533	5
1364	5
1329	5
420	5
1354	5
1098	5
1935	5
1801	5
545	5
1754	5
1123	5
1328	5
1664	5
1403	5
1662	5
1109	5
1031	5
1022	5
1862	5
1142	5
1009	5
1868	5
1241	5
1457	5
1184	5
865	5
1244	5
1826	5
1436	5
1326	5
1585	5
1505	5
1351	5
715	5
1253	5
1076	5
1559	5

1424	5
1865	5
1408	5
1586	5
1458	5
740	5
1208	5
2340	5
1077	5
1567	5
1267	5
4050	5
1148	5
2710	5
1654	5
1269	5
1057	5
2119	5
1021	5
1683	5
1573	5
2830 - 2882	5
1055	5
966	4
1459	4
2640	4
2830	4
1138	4
1608	4
2273	4
1334	4
1976	4
884	4
1881	4
755	4
1367	4
1318	4
1203	4
902	4
1904	4
1933	4
3700	4
1171	4
1059	4
1698	4
2560	4
967	4

416	4
2450	4
2805	4
1353	4
12000	4
1517	4
497	4
1980	4
1158	4
360	4
1343	4
1072	4
993	4
1494	4
10000	4
2240	4
662	4
1081	4
1169	4
1752.12	4
2135	4
972	4
955	4
1945	4
1008	4
1238	4
1449	4
1711	4
418	4
3520	4
814	4
1067	4
1795	4
1046	4
992	4
795	4
1111	4
1837	4
1708	4
1929	4
1061	4
996	4
4100	4
665	4
1776	4
602	4
1149	4

1332	4
1504	4
1073	4
914	4
1466	4
3436	4
1592	4
927	4
1885	4
1806	4
1596	4
1079	4
1864	4
1891	4
1074	4
1432	4
2439	4
1091	4
1695	4
1563	4
1234	4
1611	4
877	4
4750	4
946	4
1702	4
595	4
775	4
1975	4
1634	4
1259	4
1118	4
1051	4
1279	4
2760	4
1202	4
2160	4
1732	4
1058	4
1236	4
1489	4
8000	4
1219	4
2070	4
861	4
1207	4
1134	4

1037	4
1842	4
1394	4
580	4
1201	4
1258	4
1291	4
921	4
3850	4
1537	4
460	4
1616	4
1569	4
1521	4
1034	4
1104	4
1767	4
1938	4
1407	4
1386	4
1965	4
1682	3
2940	3
1591	3
1603	3
1599	3
2325	3
1006	3
375	3
1384	3
1164	3
1594	3
1632	3
3155	3
2106	3
1172	3
2357	3
1018	3
1566	3
1011	3
690	3
1191	3
672	3
1416	3
664	3
1536	3
1819	3

3040	3
3630	3
1758	3
1448	3
1704	3
2144	3
2440	3
1307	3
891	3
1129	3
1227	3
896	3
1356	3
1039	3
1985	3
1477	3
3385	3
1853	3
5100	3
1496	3
1860	3
1312	3
1584	3
2017	3
1788	3
1144	3
1071	3
435	3
2882	3
658	3
829	3
2470	3
661	3
994	3
958	3
2159	3
1209	3
1672	3
1576	3
1032	3
648	3
3526	3
1498	3
1942	3
1845	3
2390	3
1311	3

1692	3
702	3
2880	3
1737	3
1347	3
1858	3
3252	3
2060	3
2483	3
2264	3
4111	3
425	3
1017	3
1553	3
923	3
3205	3
1200 - 2400	3
909	3
3335	3
1117	3
575	3
1523	3
3067	3
2289	3
9600	3
1398	3
2330	3
782	3
2502	3
1097	3
937	3
1653	3
1212	3
1783	3
912	3
1301	3
1817	3
1539	3
2367	3
1832	3
1694	3
670	3
480	3
1154	3
1518	3
1706	3
1577	3

1513	3
1644	3
2080	3
1531	3
1086	3
1652	3
1426	3
1491	3
2559	3
1777.26	3
1336	3
1896	3
2061	3
1626	3
3630 - 3800	3
3050	3
735	3
1519	3
1294	3
1147	3
823	3
2020	3
1897	3
1713	3
1779	3
745	3
2065	3
3730	3
1382	3
1527	3
1839	3
1768	3
1805	3
1689	3
2610	3
2503	3
2292	3
2249.81 - 4112.19	3
1984	3
1960	3
663	3
2230	3
2087.01	3
1621	3
2770	3
565	3
1024	3

1233	3
1919	3
2550	3
4600	3
999	3
1469	3
1068	3
6500	3
703	3
1302	3
1428	3
1043	3
2257	3
1182	3
2062	3
1903	3
1066	3
1478	3
1124	3
2099	3
7200	3
1488	3
1751	3
2095	3
1678	3
6200	3
2422	3
1726	3
1094	3
1341	3
1121	3
1588	3
2732	3
2392	2
1549	2
2849	2
734	2
799	2
730	2
2540	2
1463	2
4260	2
1486	2
2493	2
3754	2
2658	2
4144	2

2378	2
3508	2
3626	2
1214	2
711	2
1709	2
1249	2
834	2
1578	2
2265	2
794	2
8321	2
2430	2
3960	2
1441	2
2197	2
1283	2
1099	2
11000	2
470	2
1239	2
1542	2
3356	2
712	2
1677	2
913	2
1431	2
1492	2
4003	2
1107.83	2
3170	2
3198	2
551	2
1804 - 2273	2
1808	2
1863	2
1417	2
2520	2
1409	2
3758	2
2569	2
1348	2
986	2
1369.1	2
2388	2
524 - 894	2
3295	2

2462	2
3522	2
1819.18	2
3126	2
2170	2
1137	2
646	2
4025	2
350	2
3785	2
805	2
2420	2
2456	2
1391	2
1948	2
1402	2
2320	2
3360	2
2675	2
3035	2
997	2
4190	2
3210	2
1607	2
3418	2
2140	2
1298	2
1438	2
1623	2
1844	2
991	2
1319	2
3009	2
2409	2
1430 - 1630	2
613 - 648	2
570	2
2950	2
633	2
1423	2
2030	2
1211	2
1359	2
589	2
1371	2
1106	2
815	2

835	2
1789	2
3657	2
1048	2
1746	2
2024	2
1999	2
1388	2
1866	2
2002	2
1584.01	2
596	2
655	2
1996	2
2045	2
585	2
1546	2
3951	2
682	2
5800	2
3675	2
726	2
3012	2
1743	2
2214	2
1739	2
2689	2
1281	2
856	2
1078	2
1651	2
2006	2
1331	2
1667	2
2172	2
1237	2
4346	2
2268	2
2259	2
1949	2
1507	2
876	2
1365 - 1700	2
1261	2
827	2
765	2
2025	2

2460	2
1666	2
3024	2
1421	2
2665	2
1970	2
3408	2
381	2
1002	2
1763	2
620 - 934	2
2260	2
6136	2
485	2
1581	2
2089	2
1499	2
6652	2
1814	2
1684	2
1524	2
1950.2	2
1876	2
1327	2
2075	2
1381	2
1676	2
1363	2
1552	2
1744	2
2093	2
3090	2
1390 - 1600	2
2047	2
4250	2
1013	2
1323	2
1859	2
644	2
3425	2
1757	2
882	2
2780	2
142.61Sq. Meter	2
1699	2
1454	2
971	2

1752	2
2051	2
3004	2
381 - 535	2
3262	2
1159	2
1023	2
3216	2
1612	2
509	2
3453	2
2210	2
943	2
1827	2
2321	2
527	2
1136	2
1679	2
1643	2
2720	2
1229	2
4000 - 5249	2
1556	2
10961	2
1548	2
1688	2
693	2
1434	2
1914	2
725	2
1255 - 1350	2
2777.29	2
3940	2
2403	2
922	2
645 - 936	2
1823	2
4235	2
1366	2
1493	2
3875	2
1248	2
1419.59	2
2266	2
1721	2
1103	2
938	2

1538	2
1728	2
1376	2
849	2
606	2
4300	2
1468	2
535	2
1368	2
1786	2
1707	2
3420	2
4104	2
2111	2
3761	2
2557	2
793	2
2238	2
686	2
1554	2
1001	2
3206	2
3260	2
2556	2
432	2
1995	2
1937	2
697	2
3095	2
1618 - 1929	2
7800	2
1613	2
1284	2
2666	2
1894	2
1674	2
3633	2
1003	2
5500	2
1516	2
973	2
2375	2
1633	2
1618	2
2376	2
2524	2
2580 - 2591	2

1442	2
1414	2
1115 - 1130	2
1622	2
3930	2
1934	2
1799	2
1396	2
901	2
3565	2
2041	2
1601	2
952	2
1836	2
4700	2
4689	1
2785	1
2100 - 2850	1
1989	1
754	1
2444	1
3290	1
1932.47	1
2086	1
2204 - 2362	1
911	1
2206	1
1659	1
2485	1
1437 - 1629	1
850 - 1060	1
30400	1
1701	1
2528 - 3188	1
1330.74	1
869	1
3010 - 3410	1
5700	1
2031	1
2153	1
475	1
1783 - 1878	1
120Sq. Yards	1
3729	1
953	1
5515	1
24Sq. Meter	1

1321	1
1317	1
2176	1
3554	1
3671	1
1681	1
2236	1
2955	1
712 - 938	1
2171.66	1
1747	1
1273	1
2615	1
772	1
2999.97	1
1289	1
1377	1
1892	1
4428	1
2912	1
2059	1
888	1
532	1
3227	1
1915	1
963	1
1053.4	1
540 - 670	1
315Sq. Yards	1
1650 - 2538	1
3445	1
2283	1
3815	1
4827	1
1574	1
1125 - 1500	1
813	1
3884	1
1874	1
2885	1
1462	1
607	1
3293 - 5314	1
2386	1
1210 - 1477	1
892	1
4818	1

4355	1
908	1
3369 - 3464	1
3530	1
3950	1
1205.47	1
3595	1
1482 - 1846	1
944	1
832	1
3259	1
706	1
4356	1
2138	1
910.2	1
673	1
704 - 730	1
2408	1
685	1
704	1
2725 - 3250	1
1732.46	1
2592	1
4470	1
624	1
722	1
2775	1
769	1
1712	1
833	1
669	1
3811	1
3860	1
3027	1
1734	1
3537	1
956	1
1218	1
2363	1
885	1
1451.5	1
3016	1
461.82	1
1974 - 2171	1
3640	1
500Sq. Yards	1
2043	1

2779	1
1020.07	1
5965	1
2806 - 3019	1
2416	1
3905	1
2107	1
2431	1
4640	1
818	1
9000	1
1288	1
858	1
1689.28	1
2805 - 3565	1
4960	1
3179	1
2928	1
1932	1
2182	1
2461	1
1437	1
2155	1
904	1
977	1
2401	1
934 - 1437	1
980 - 1030	1
888 - 1290	1
1360 - 1890	1
2370	1
5	1
1004	1
5985	1
2466 - 2856	1
3580	1
2220	1
1070 - 1315	1
1888	1
3040Sq. Meter	1
300	1
1778	1
2006.8	1
1922	1
989	1
1818	1
2820	1

4772	1
5600	1
4051	1
1777	1
2721	1
590	1
3161	1
871	1
3515	1
3366	1
1562	1
2105	1
1500 - 2400	1
2162	1
1733	1
1974	1
385 - 440	1
2631	1
2792	1
1255 - 1863	1
1300 - 1405	1
2383	1
3080	1
787	1
660 - 700	1
6613	1
502	1
3339	1
1483	1
1564 - 1850	1
1452.19	1
3010	1
1446 - 1506	1
5530	1
792	1
555	1
4830	1
1939	1
714	1
351	1
2526	1
3436 - 3643	1
596 - 804	1
1776.42	1
2319	1
621	1
981	1

1548.3	1
2302	1
1525.84	1
188.89Sq. Yards	1
797	1
2396	1
1551	1
3563	1
462	1
3Cents	1
1310 - 1615	1
36000	1
1113.12	1
1782 - 2000	1
2167	1
117Sq. Yards	1
2100 - 5405	1
2448	1
770 - 841	1
2171	1
42000	1
1793	1
4850	1
668	1
2710 - 3360	1
395	1
1452.55	1
296	1
2570	1
881	1
3160	1
1769	1
133.3Sq. Yards	1
1554.3	1
4920	1
627	1
3405.1	1
6040	1
4510	1
204Sq. Meter	1
1766	1
870 - 1080	1
45Sq. Yards	1
948	1
1014	1
695	1
11338	1

30000	1
3190	1
4460	1
3297	1
3401	1
1469 - 1766	1
2246	1
3245	1
515	1
3025	1
1004 - 1204	1
361.33Sq. Yards	1
987	1
2121	1
1646	1
78.03Sq. Meter	1
1208.51	1
3300 - 3335	1
983	1
2039	1
1316	1
1005.03 - 1252.49	1
1901	1
2005	1
605 - 624	1
4260 - 4408	1
2582	1
2663	1
2127	1
1349 - 3324	1
45	1
951	1
1652.5	1
3532	1
2601	1
906	1
2384	1
941	1
2563 - 2733	1
2172.65	1
3555	1
6150	1
671	1
581.91	1
3450 - 3472	1
2337	1
5080	1

4075	1
1987	1
2245	1
1597	1
2423	1
3131	1
396	1
4273	1
1775	1
1687	1
2435	1
2328	1
1180 - 1630	1
806	1
1628	1
1660.4	1
1250 - 1305	1
824	1
2519	1
1673	1
670 - 980	1
3584	1
2168	1
3125	1
1831	1
1270 - 1275	1
840 - 1010	1
1193	1
1500Sq. Meter	1
620 - 933	1
2028	1
2611	1
2324	1
2968	1
1100 - 1225	1
1565 - 1595	1
2232	1
2932	1
3820	1
24Guntha	1
934	1
84.53Sq. Meter	1
651	1
2.09Acres	1
1482 - 1684	1
583	1
981 - 1249	1

2108	1
52272	1
2479.13	1
3042	1
2204	1
916	1
688	1
3500 - 3600	1
122Sq. Yards	1
1266.67	1
942 - 1117	1
2679	1
2572	1
3425 - 3435	1
1269.72	1
1266	1
817	1
800 - 2660	1
1741	1
3463	1
2120	1
505	1
1408 - 1455	1
2901	1
4050 - 4075	1
697Sq. Meter	1
1716	1
638	1
4856	1
3664	1
539	1
2173	1
655 - 742	1
2695 - 2940	1
727	1
1722	1
2000 - 5634	1
1429	1
1574Sq. Yards	1
3606	1
947	1
2424	1
3516	1
3770	1
1378	1
2090	1
3329	1

785	1
716Sq. Meter	1
1412	1
1439	1
628	1
804.1	1
580 - 650	1
1373	1
3680	1
2863	1
766	1
1593	1
598 - 958	1
1631	1
1500Cents	1
1829	1
5720	1
2132	1
1815	1
660 - 670	1
3044	1
1750 - 2640	1
790	1
2476	1
1877	1
1649	1
2079	1
3589	1
1606	1
4634	1
3467.86	1
2195	1
14000	1
1617	1
1010 - 1300	1
2122	1
2Acres	1
3103 - 3890	1
1450 - 1950	1
2023	1
2274.24	1
4900	1
864	1
2293	1
1567.2	1
26136	1
959	1

132Sq. Yards	1
855	1
1691.2	1
3073	1
764	1
340	1
2465	1
547.34 - 827.31	1
5924	1
552	1
2365	1
1561	1
1637	1
24	1
1445 - 1455	1
884 - 1116	1
850 - 1093	1
1087	1
2462 - 2467	1
763 - 805	1
3307 - 3464	1
1.26Acres	1
5422	1
1542.14	1
3746	1
2144.6	1
1824	1
15Acres	1
3990	1
2501	1
3301.8	1
1450 - 1595	1
3204	1
3489	1
4166	1
1440 - 1884	1
2842	1
1558.67	1
2134	1
2515	1
2130	1
1100Sq. Meter	1
947.55	1
15	1
1467	1
3968	1
2169	1

3235	1
30Acres	1
1000 - 1285	1
4239	1
3978	1
3621	1
2008	1
694	1
633 - 666	1
315	1
2112.95	1
5.31Acres	1
4408	1
1962	1
1568	1
4900 - 4940	1
3060	1
873	1
1610 - 1880	1
755 - 770	1
540 - 740	1
10200	1
5200	1
20000	1
3508 - 4201	1
6830	1
3692	1
1503	1
2317	1
666	1
3670	1
1114	1
2489	1
1100Sq. Yards	1
520 - 645	1
3293	1
700 - 900	1
3602	1
596 - 861	1
2251	1
1998	1
3056	1
808	1
826	1
3870	1
4097	1
664 - 722	1

2380	1
3569	1
151.11Sq. Yards	1
3435	1
3144	1
2825	1
3606 - 5091	1
650 - 665	1
3033	1
1181.7	1
1338	1
2625	1
1502	1
667	1
4360	1
1160 - 1195	1
1000Sq. Meter	1
445	1
2181	1
2118	1
4400 - 6800	1
2496	1
8400	1
1558	1
1054	1
2646	1
540 - 565	1
2736	1
616	1
1471	1
3855	1
7400	1
4170	1
1506	1
1872	1
5108	1
1393	1
3197	1
1217	1
3230	1
1925 - 2680	1
4110	1
2533	1
615 - 985	1
2297	1
4125Perch	1
1641	1

1120 - 1145	1
4400 - 6640	1
3090 - 5002	1
35000	1
1230 - 1290	1
2048	1
1609	1
1861	1
3362	1
1410 - 1710	1
1079 - 1183	1
2800 - 2870	1
2015	1
469	1
1660 - 1805	1
2704	1
3760	1
3071	1
2429	1
2925	1
1719.3	1
1668	1
1337	1
5270	1
1547	1
1324	1
2003	1
2504	1
4290	1
750 - 800	1
2845	1
1195 - 1440	1
1544	1
961	1
2791	1
1753	1
1511	1
8500	1
2110	1
4560	1
1Grounds	1
1160 - 1315	1
706 - 716	1
3560	1
2940Sq. Yards	1
2996	1
1902	1

753	1
1763.25	1
527 - 639	1
456	1
4320	1
534	1
2249.81	1
1119	1
943 - 1220	1
1797	1
1816	1
2041 - 2090	1
588	1
1234.6	1
492	1
1905	1
5384	1
1723	1
1629	1
1113.27	1
34.46Sq. Meter	1
11890	1
3309	1
998	1
2733	1
1389	1
2970	1
1907	1
4446	1
2470 - 2790	1
783 - 943	1
717	1
3117	1
1791 - 4000	1
2648	1
2223	1
45.06Sq. Meter	1
799 - 803	1
2185	1
5230	1
1867	1
3496	1
612	1
1792	1
1587	1
2125	1
1529	1

2801.25	1
1688.12	1
939	1
11	1
2023.71	1
451	1
3045	1
2026	1
3005	1
4201	1
4382	1
1921	1
1669	1
2285	1
1642	1
1361	1
3270	1
857	1
2247	1
2980	1
2087	1
4278	1
4500 - 5540	1
10030	1
5150	1
2956	1
1522	1
5425	1
978	1
4041	1
3628	1
2437	1
524	1
1230 - 1490	1
1909	1
1736	1
1520 - 1740	1
2405	1
1145 - 1340	1
1015 - 1540	1
86.72Sq. Meter	1
2735	1
1287	1
284	1
2077	1
499	1
2826	1

4550	1
1049	1
2404	1
854 - 960	1
2597	1
1981	1
2650 - 2990	1
7514	1
1.25Acres	1
3350	1
1255 - 1375	1
1733.5	1
10624	1
691	1
3092	1
610 - 615	1
3913	1
2098	1
1761	1
2497	1
276	1
3067 - 8156	1
1042 - 1105	1
1563.05	1
1828	1
1940	1
1189	1
4482	1
1150 - 1194	1
2507	1
684 - 810	1
866.28	1
3410	1
2088	1
581	1
516	1
1383	1
2316	1
1626.6	1
4303	1
1990	1
2495	1
3075	1
2082	1
896.9	1
2064	1
488	1

660 - 780	1
2916	1
2312	1
1520 - 1759	1
2511	1
2957 - 3450	1
2894	1
1413	1
4450	1
784	1
1235 - 1410	1
3484 - 3550	1
1139.7	1
38Guntha	1
3280	1
929 - 1078	1
2150 - 2225	1
3381	1
1370.07	1
1550 - 1590	1
777.4	1
1886	1
623	1
2546	1
1879	1
1401	1
2795	1
3535	1
1200 - 1800	1
3019	1
1510 - 1670	1
1748	1
1604	1
1248.52	1
2035	1
8840	1
5666 - 5669	1
614	1
2162.03	1
6729	1
893	1
2019	1
1473	1
4040	1
2406	1
2400 - 2600	1
2137	1

2787	1
3175	1
1052 - 1322	1
842	1
6Acres	1
1140 - 1250	1
4304	1
4209	1
2415	1
1731	1
1589	1
976	1
4694	1
302	1
629 - 1026	1
2282	1
2695	1
60	1
1215 - 1495	1
2875	1
2295	1
1686	1
1020 - 1130	1
2758	1
1133 - 1384	1
774	1
1264	1
1857	1
1916	1
4000 - 4450	1
142.84Sq. Meter	1
3734	1
964	1
4007	1
300Sq. Yards	1
2505	1
567	1
1400 - 1421	1
4350	1
1443	1
886	1
16335	1
747	1
1623.29	1
650 - 760	1
5480	1
5656	1

```
866               1
3504              1
4723              1
2453              1
2651              1
2270              1
2342              1
6600              1
1627.86           1
2215 - 2475       1
1918              1
2113              1
3876              1
897               1
2776              1
3480              1
1                 1
2372              1
167Sq. Meter      1
1076 - 1199       1
2872              1
1648              1
1379              1
3124              1
9200              1
613               1
250               1
2395              1
1557              1
1200 - 1470       1
7150              1
1369              1
5665.84           1
2920              1
6688              1
1331.95           1
Name: count, dtype: int64
```

[9]:

```python
def extract_int_size(text):
    if isinstance(text, str):  # Check if text is a string
        # Using regular expression to find first sequence of digits
        match = re.search(r'\d+', text)
        if match:
            return int(match.group())  # Convert matched digits to integer
    return None  # Return None for non-string or when no digits found

# Apply function to extract integer size
```

```
df['size'] = df['size'].apply(extract_int_size)
```

[10]:
```
df.head()
```

[10]:
```
              area_type   availability                  location  size  \
0  Super built-up  Area         19-Dec  Electronic City Phase II   2.0
1            Plot  Area  Ready To Move          Chikka Tirupathi   4.0
2        Built-up  Area  Ready To Move               Uttarahalli   3.0
3  Super built-up  Area  Ready To Move        Lingadheeranahalli   3.0
4  Super built-up  Area  Ready To Move                  Kothanur   2.0

   society total_sqft  bath  balcony   price
0  Coomee        1056   2.0      1.0   39.07
1  Theanmp       2600   5.0      3.0  120.00
2      NaN       1440   2.0      3.0   62.00
3  Soiewre       1521   3.0      1.0   95.00
4      NaN       1200   2.0      1.0   51.00
```

[11]:
```
def extract_integer(value):

    match = re.search(r'\d+', str(value))
    if match:
        return int(match.group())
    else:
        return None

df['total_sqft'] = df['total_sqft'].apply(extract_integer)
```

[12]:
```
df.shape
```

[12]:
```
(13320, 9)
```

[13]:
```
df.head(3)
```

[13]:
```
              area_type   availability                  location  size  \
0  Super built-up  Area         19-Dec  Electronic City Phase II   2.0
1            Plot  Area  Ready To Move          Chikka Tirupathi   4.0
2        Built-up  Area  Ready To Move               Uttarahalli   3.0

   society  total_sqft  bath  balcony   price
0  Coomee         1056   2.0      1.0   39.07
1  Theanmp        2600   5.0      3.0  120.00
2      NaN        1440   2.0      3.0   62.00
```

[14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

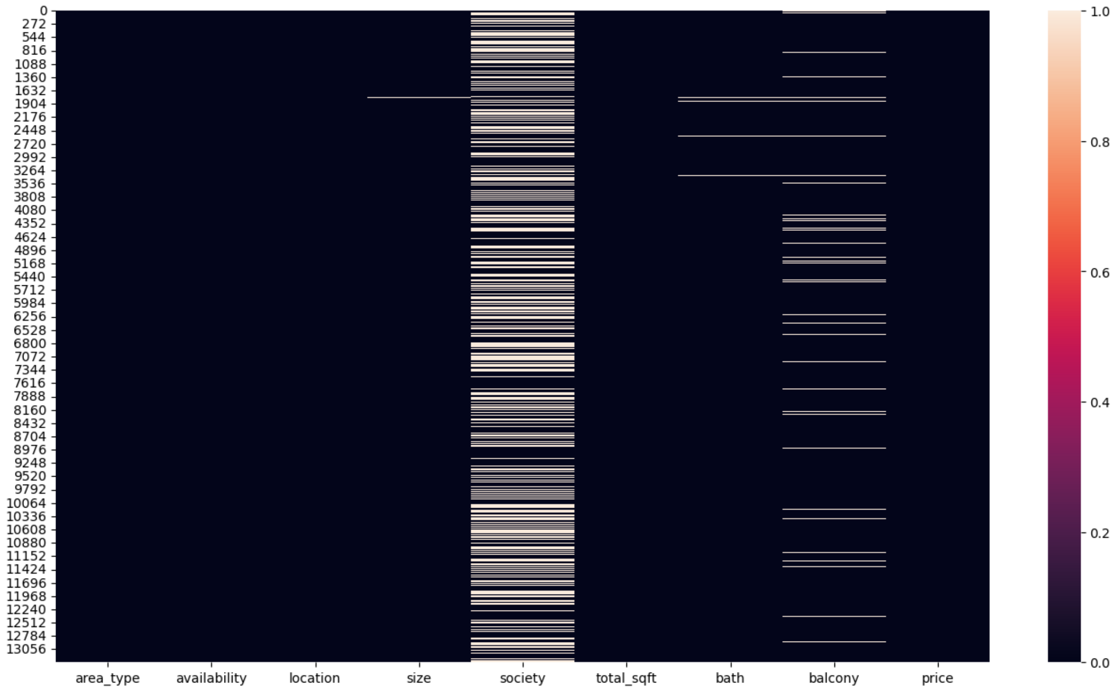
```
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   area_type     13320 non-null  object
 1   availability  13320 non-null  object
 2   location      13319 non-null  object
 3   size          13304 non-null  float64
 4   society       7818 non-null   object
 5   total_sqft    13320 non-null  int64
 6   bath          13247 non-null  float64
 7   balcony       12711 non-null  float64
 8   price         13320 non-null  float64
dtypes: float64(4), int64(1), object(4)
memory usage: 936.7+ KB
```

[15]:
```
df.describe()
```

[15]:

	size	total_sqft	bath	balcony	price
count	13304.000000	13320.000000	13247.000000	12711.000000	13320.000000
mean	2.803743	1552.947072	2.692610	1.584376	112.565627
std	1.294974	1236.591541	1.341458	0.817263	148.971674
min	1.000000	1.000000	1.000000	0.000000	8.000000
25%	2.000000	1100.000000	2.000000	1.000000	50.000000
50%	3.000000	1274.000000	2.000000	2.000000	72.000000
75%	3.000000	1675.000000	3.000000	2.000000	120.000000
max	43.000000	52272.000000	40.000000	3.000000	3600.000000

[16]:
```
plt.figure(figsize=(16,9))
sns.heatmap(df.isnull())
plt.show()
```

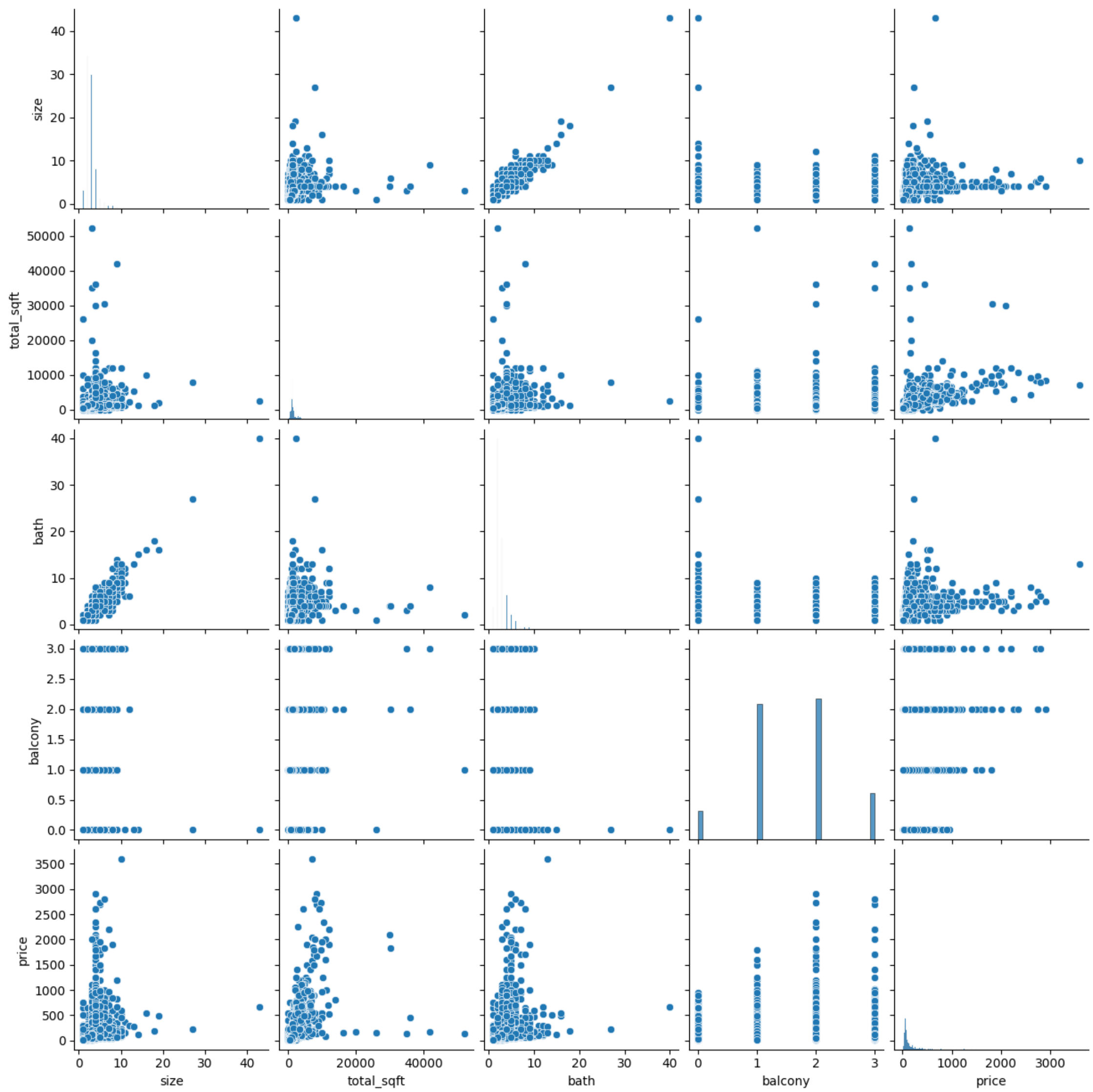


[17]:
```
df.isnull().sum()/ len(df) * 100
```

[17]:
```
area_type        0.000000
availability     0.000000
location         0.007508
size             0.120120
society         41.306306
total_sqft       0.000000
bath             0.548048
balcony          4.572072
price            0.000000
dtype: float64
```

[18]:
```
sns.pairplot(df)
```

[18]:
```
<seaborn.axisgrid.PairGrid at 0x798b54d44470>
```




```
[19]: # Drop columns safely (ignore if they don't exist)
      df.drop(columns=['society', 'area_type', 'availability', 'location'],␣
       ↪errors='ignore', inplace=True)

      # Fill missing values in numeric columns
      df['bath'].fillna(df['bath'].median(), inplace=True)
      df['balcony'].fillna(df['balcony'].median(), inplace=True)
      df['size'].fillna(df['size'].median(), inplace=True)
```

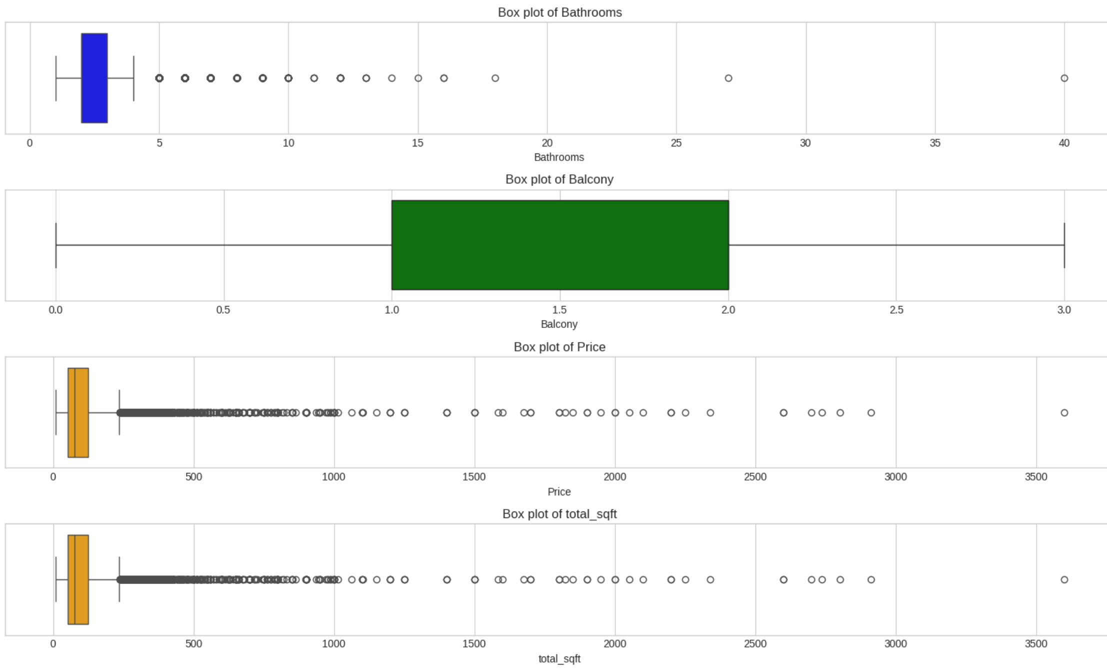
```
[20]: df.duplicated().sum()
```

```
[20]: np.int64(1637)
```

[21]:
```python
# Drop Duplicate Rows
df.drop_duplicates(inplace=True)
```

[22]:
```python
import matplotlib.pyplot as plt

plt.style.use('seaborn-v0_8-whitegrid')  # works fine with latest Matplotlib
```

[23]:
```python
# Create subplots for each variable
plt.figure(figsize=(15, 9))

# Box plot for 'bath'
plt.subplot(4, 1, 1)
sns.boxplot(x='bath', data=df, color='blue', orient='v')
plt.title('Box plot of Bathrooms')
plt.xlabel('Bathrooms')

# Box plot for 'balcony'
plt.subplot(4, 1, 2)
sns.boxplot(x='balcony', data=df, color='green', orient='v')
plt.title('Box plot of Balcony')
plt.xlabel('Balcony')

# Box plot for 'price'
plt.subplot(4, 1, 3)
sns.boxplot(x='price', data=df, color='orange', orient='v')
plt.title('Box plot of Price')
plt.xlabel('Price')

# Box plot for 'total_sqft'
plt.subplot(4, 1, 4)
sns.boxplot(x='price', data=df, color='orange', orient='v')
plt.title('Box plot of total_sqft')
plt.xlabel('total_sqft')

plt.tight_layout()
plt.show()
```



[24]:
```python
# Calculate quartiles and IQR for each column
Q1_bath = df['bath'].quantile(0.25)
Q3_bath = df['bath'].quantile(0.75)
IQR_bath = Q3_bath - Q1_bath

Q1_balcony = df['balcony'].quantile(0.25)
Q3_balcony = df['balcony'].quantile(0.75)
IQR_balcony = Q3_balcony - Q1_balcony

Q1_price = df['price'].quantile(0.25)
Q3_price = df['price'].quantile(0.75)
IQR_price = Q3_price - Q1_price

Q1_total_sqft = df['total_sqft'].quantile(0.25)  # Calculate Q1 for total_sqft
Q3_total_sqft = df['total_sqft'].quantile(0.75)  # Calculate Q3 for total_sqft
IQR_total_sqft = Q3_total_sqft - Q1_total_sqft   # Calculate IQR for total_sqft

# Define upper and lower bounds to filter outliers
lower_bound_bath = Q1_bath - 1.5 * IQR_bath
upper_bound_bath = Q3_bath + 1.5 * IQR_bath

lower_bound_balcony = Q1_balcony - 1.5 * IQR_balcony
upper_bound_balcony = Q3_balcony + 1.5 * IQR_balcony

lower_bound_price = Q1_price - 1.5 * IQR_price
```

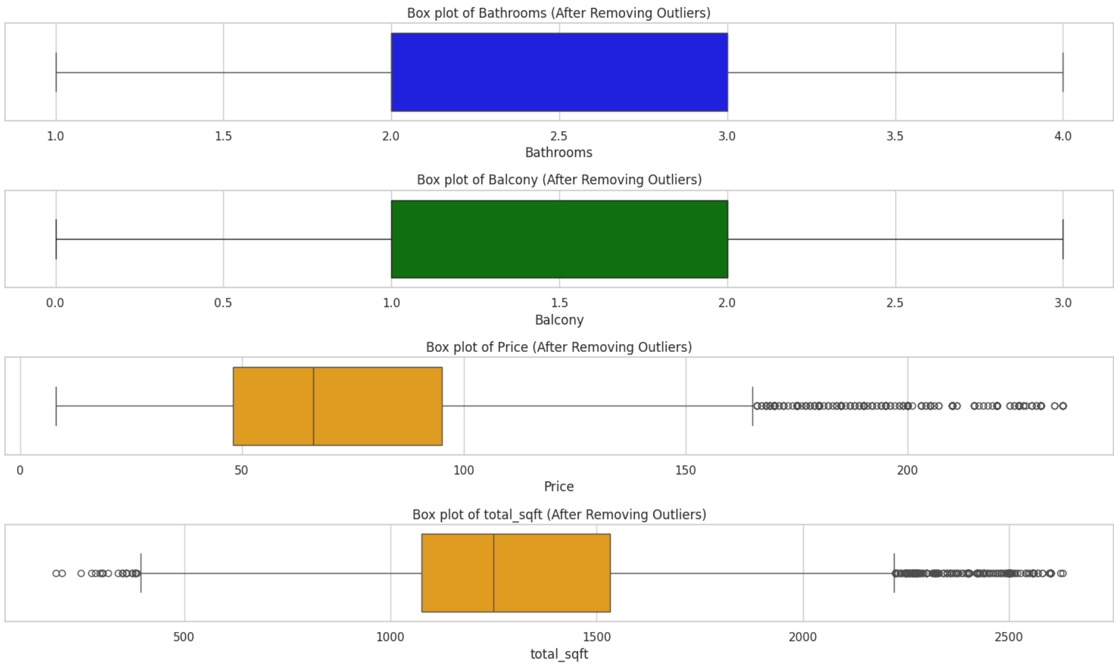
```
upper_bound_price = Q3_price + 1.5 * IQR_price

lower_bound_total_sqft = Q1_total_sqft - 1.5 * IQR_total_sqft  # Define lower␣
↪bound for total_sqft
upper_bound_total_sqft = Q3_total_sqft + 1.5 * IQR_total_sqft  # Define upper␣
↪bound for total_sqft

# Filter rows where values are within the defined bounds
df_filtered = df[
    (df['bath'] >= lower_bound_bath) & (df['bath'] <= upper_bound_bath) &
    (df['balcony'] >= lower_bound_balcony) & (df['balcony'] <=␣
↪upper_bound_balcony) &
    (df['price'] >= lower_bound_price) & (df['price'] <= upper_bound_price) &
    (df['total_sqft'] >= lower_bound_total_sqft) & (df['total_sqft'] <=␣
↪upper_bound_total_sqft)
]

# Display the shape of the filtered DataFrame
print("Shape of filtered DataFrame:", df_filtered.shape)
```

```
Shape of filtered DataFrame: (9703, 5)
```

[25]:
```
import seaborn as sns
sns.set_theme(style="whitegrid")  # replaces old seaborn-whitegrid style
```

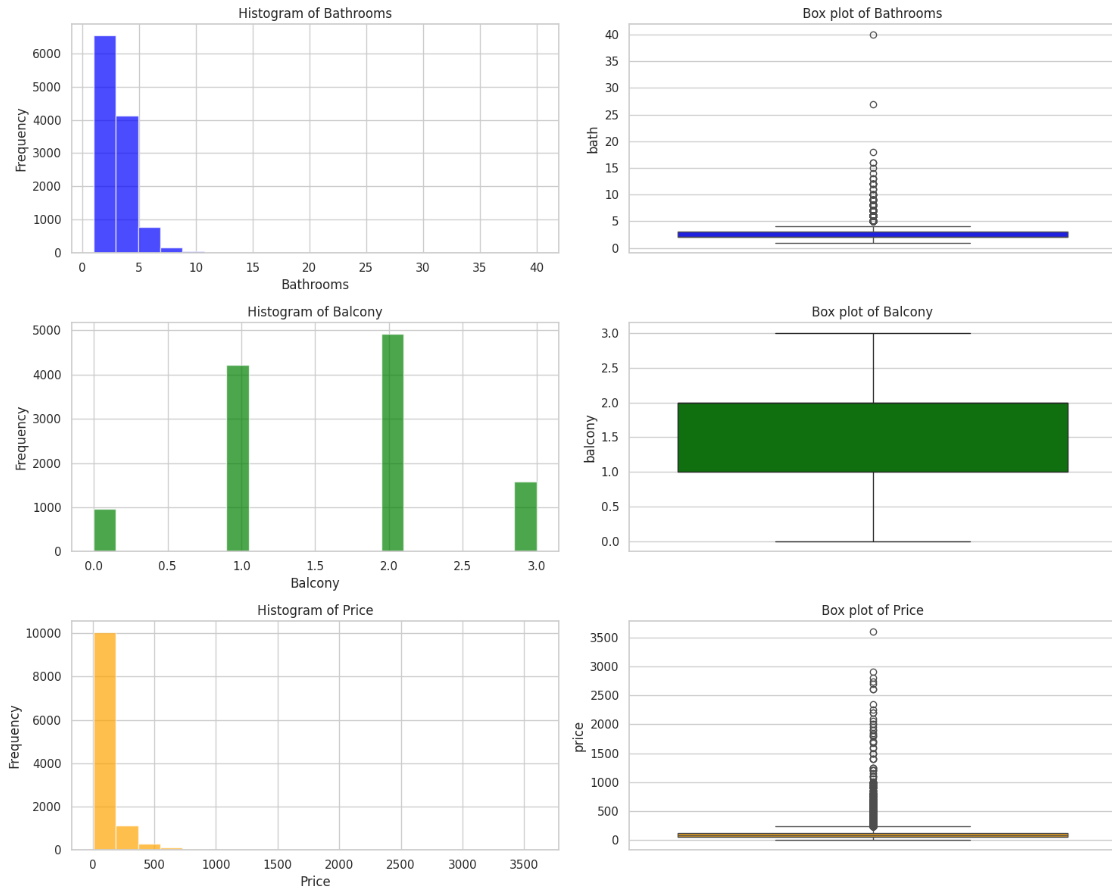
[26]:
```
# Create subplots for each variable
plt.figure(figsize=(15, 9))

# Box plot for 'bath' after removing outliers
plt.subplot(4, 1, 1)
sns.boxplot(x='bath', data=df_filtered, color='blue', orient='v')
plt.title('Box plot of Bathrooms (After Removing Outliers)')
plt.xlabel('Bathrooms')

# Box plot for 'balcony' after removing outliers
plt.subplot(4, 1, 2)
sns.boxplot(x='balcony', data=df_filtered, color='green', orient='v')
plt.title('Box plot of Balcony (After Removing Outliers)')
plt.xlabel('Balcony')

# Box plot for 'price' after removing outliers
plt.subplot(4, 1, 3)
sns.boxplot(x='price', data=df_filtered, color='orange', orient='v')
plt.title('Box plot of Price (After Removing Outliers)')
plt.xlabel('Price')

# Box plot for 'total_sqft' after removing outliers
```

```python
plt.subplot(4, 1, 4)
sns.boxplot(x='total_sqft', data=df_filtered, color='orange', orient='v')
plt.title('Box plot of total_sqft (After Removing Outliers)')
plt.xlabel('total_sqft')

plt.tight_layout()
plt.show()
```

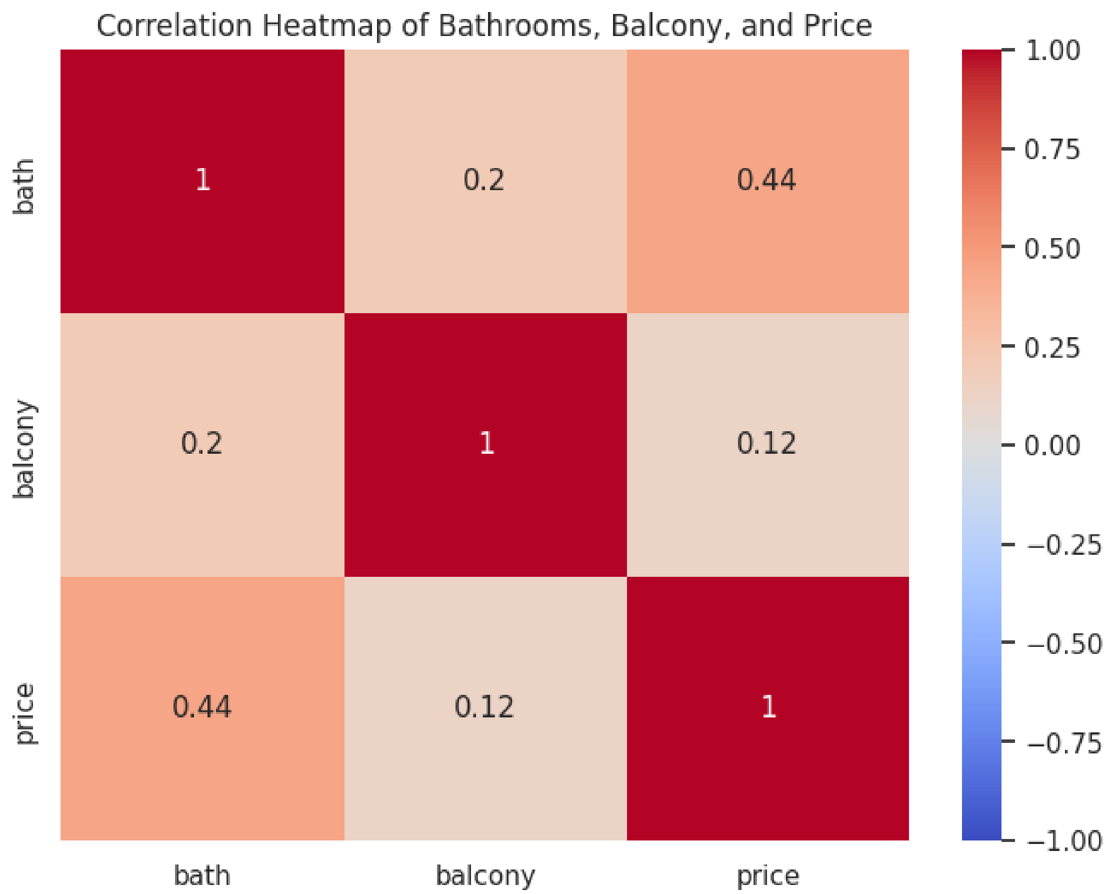


[27]:
```python
import matplotlib.pyplot as plt

# Create subplots for histograms
plt.figure(figsize=(15, 12))

# Histogram for 'bath'
plt.subplot(3, 2, 1)
plt.hist(df['bath'], bins=20, color='blue', alpha=0.7)
plt.title('Histogram of Bathrooms')
plt.xlabel('Bathrooms')
plt.ylabel('Frequency')

# Histogram for 'balcony'
plt.subplot(3, 2, 3)
plt.hist(df['balcony'], bins=20, color='green', alpha=0.7)
plt.title('Histogram of Balcony')
plt.xlabel('Balcony')
```

```
plt.ylabel('Frequency')

# Histogram for 'price'
plt.subplot(3, 2, 5)
plt.hist(df['price'], bins=20, color='orange', alpha=0.7)
plt.title('Histogram of Price')
plt.xlabel('Price')
plt.ylabel('Frequency')

# Box plot for 'bath'
plt.subplot(3, 2, 2)
sns.boxplot(y='bath', data=df, color='blue')
plt.title('Box plot of Bathrooms')

# Box plot for 'balcony'
plt.subplot(3, 2, 4)
sns.boxplot(y='balcony', data=df, color='green')
plt.title('Box plot of Balcony')

# Box plot for 'price'
plt.subplot(3, 2, 6)
sns.boxplot(y='price', data=df, color='orange')
plt.title('Box plot of Price')

plt.tight_layout()
plt.show()
```



[28]:
```python
# Compute the correlation matrix
correlation_matrix = df[['bath', 'balcony', 'price']].corr()

# Plotting the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Heatmap of Bathrooms, Balcony, and Price')
plt.show()
```



[29]:
```python
from sklearn.preprocessing import MinMaxScaler

# Select columns to scale (excluding 'price')
columns_to_scale = ['size', 'total_sqft', 'bath', 'balcony','price']

# Initialize the MinMaxScaler
scaler = MinMaxScaler()

# Fit and transform the selected columns
df[columns_to_scale] = scaler.fit_transform(df[columns_to_scale])
```

[30]:
```python
# Display the scaled DataFrame
df.head()
```

[30]:

	size	total_sqft	bath	balcony	price
0	0.023810	0.020183	0.025641	0.333333	0.008650
1	0.071429	0.049722	0.102564	1.000000	0.031180
2	0.047619	0.027530	0.025641	1.000000	0.015033
3	0.047619	0.029079	0.051282	0.333333	0.024220

```
4  0.023810    0.022938  0.025641  0.333333  0.011971
```

[31]:
```
from sklearn.model_selection import train_test_split

X = df.drop('price', axis=1)
y = df['price']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[32]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
from sklearn.linear_model import LinearRegression, Lasso, Ridge, BayesianRidge,␣
 ↪ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor,␣
 ↪GradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
```

[33]:
```
# Initialize models

# Random Forest Regressor
rf_regressor = RandomForestRegressor(random_state=42)

# Decision Tree Regressor
dt_regressor = DecisionTreeRegressor(random_state=42)

# Lasso Regression
lasso_reg = Lasso(alpha=0.1)

# Ridge Regression
ridge_reg = Ridge(alpha=1.0)

# AdaBoost Regressor
adaboost_reg = AdaBoostRegressor(random_state=42)

# Gradient Boosting Regressor
gb_regressor = GradientBoostingRegressor(random_state=42)

# Bayesian Ridge Regression
bayesian_reg = BayesianRidge()

# K-Nearest Neighbors Regressor
knn_reg = KNeighborsRegressor()
```

```python
# ElasticNet Regression
elastic_net = ElasticNet(alpha=0.1, l1_ratio=0.5)

# Support Vector Regression (SVR)
svr = SVR()
```

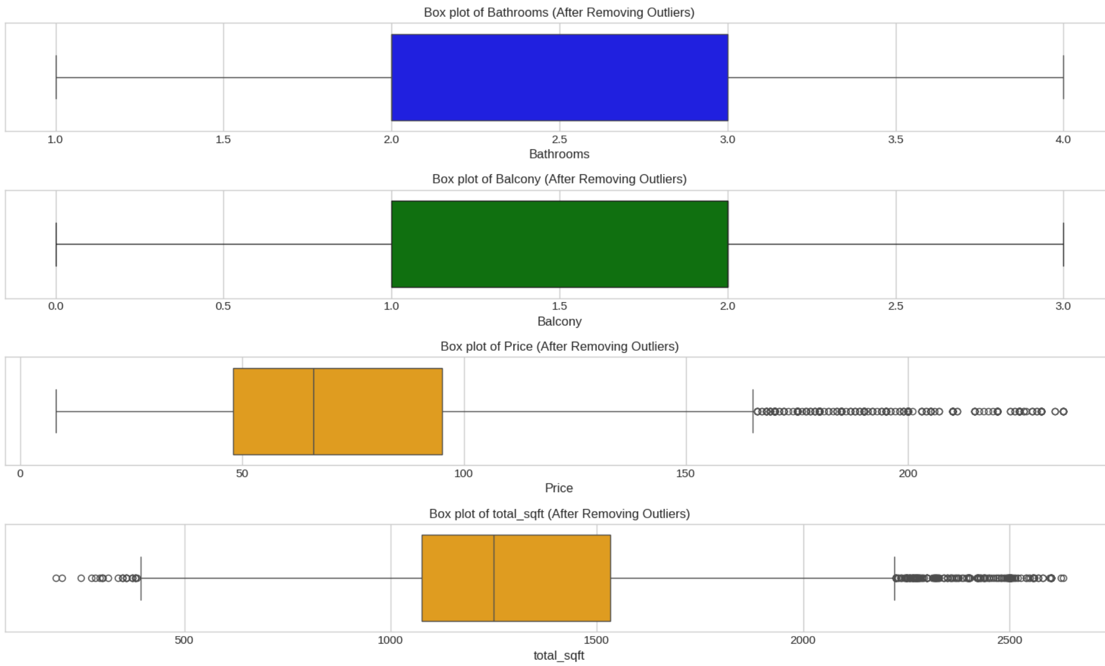
[34]:
```python
# List of models
models = {
    'Random Forest': rf_regressor,
    'Decision Tree': dt_regressor,
    'Lasso Regression': lasso_reg,
    'Ridge Regression': ridge_reg,
    'AdaBoost Regression': adaboost_reg,
    'Gradient Boosting Regression': gb_regressor,
    'Bayesian Ridge Regression': bayesian_reg,
    'K-Nearest Neighbors Regression': knn_reg,
    'ElasticNet Regression': elastic_net,
    'Support Vector Regression': svr
}

# Dictionary to store results
results = {'Model': [], 'MSE': [], 'R2': []}

# Train, predict, and evaluate each model
for name, model in models.items():
    print(f"Training {name}...")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    results['Model'].append(name)
    results['MSE'].append(mse)
    results['R2'].append(r2)
    print(f"{name} Metrics:")
    print(f"Mean Squared Error (MSE): {mse}")
    print(f"R-squared (R2): {r2}")
    print("--------------------")

# Convert results to DataFrame
results_df = pd.DataFrame(results)
```

```
Training Random Forest…
Random Forest Metrics:
Mean Squared Error (MSE): 0.0011908803472652232
R-squared (R2): 0.5242995243768505
--------------------
```

```
Training Decision Tree…
Decision Tree Metrics:
Mean Squared Error (MSE): 0.0020284248264154153
R-squared (R2): 0.18974004659032473
--------------------
Training Lasso Regression…
Lasso Regression Metrics:
Mean Squared Error (MSE): 0.0025044833699512034
R-squared (R2): -0.00042286616954356676
--------------------
Training Ridge Regression…
Ridge Regression Metrics:
Mean Squared Error (MSE): 0.0015192417268644353
R-squared (R2): 0.39313465562213
--------------------
Training AdaBoost Regression…
AdaBoost Regression Metrics:
Mean Squared Error (MSE): 0.001285730806904951
R-squared (R2): 0.48641124377224654
--------------------
Training Gradient Boosting Regression…
Gradient Boosting Regression Metrics:
Mean Squared Error (MSE): 0.0010572596181557968
R-squared (R2): 0.5776746972365212
--------------------
Training Bayesian Ridge Regression…
Bayesian Ridge Regression Metrics:
Mean Squared Error (MSE): 0.0014429051028808293
R-squared (R2): 0.4236275329459167
--------------------
Training K-Nearest Neighbors Regression…
K-Nearest Neighbors Regression Metrics:
Mean Squared Error (MSE): 0.0009755030358801712
R-squared (R2): 0.6103325920142378
--------------------
Training ElasticNet Regression…
ElasticNet Regression Metrics:
Mean Squared Error (MSE): 0.0025044833699512034
R-squared (R2): -0.00042286616954356676
--------------------
Training Support Vector Regression…
Support Vector Regression Metrics:
Mean Squared Error (MSE): 0.006281436546748873
R-squared (R2): -1.5091373451136412
--------------------
```

[35]:

```
import matplotlib.pyplot as plt
import seaborn as sns

#  Use the updated Seaborn style syntax (works in Colab)
plt.style.use('seaborn-v0_8-whitegrid')

# Create subplots for each variable
plt.figure(figsize=(15, 9))

# Box plot for 'bath' after removing outliers
plt.subplot(4, 1, 1)
sns.boxplot(x='bath', data=df_filtered, color='blue', orient='v')
plt.title('Box plot of Bathrooms (After Removing Outliers)')
plt.xlabel('Bathrooms')

# Box plot for 'balcony' after removing outliers
plt.subplot(4, 1, 2)
sns.boxplot(x='balcony', data=df_filtered, color='green', orient='v')
plt.title('Box plot of Balcony (After Removing Outliers)')
plt.xlabel('Balcony')

# Box plot for 'price' after removing outliers
plt.subplot(4, 1, 3)
sns.boxplot(x='price', data=df_filtered, color='orange', orient='v')
plt.title('Box plot of Price (After Removing Outliers)')
plt.xlabel('Price')

# Box plot for 'total_sqft' after removing outliers
plt.subplot(4, 1, 4)
sns.boxplot(x='total_sqft', data=df_filtered, color='orange', orient='v')
plt.title('Box plot of total_sqft (After Removing Outliers)')
plt.xlabel('total_sqft')

plt.tight_layout()
plt.show()
```



[36]:
```python
# Find the best model based on MSE and R2
best_model_mse = results_df.loc[results_df['MSE'].idxmin()]
best_model_r2 = results_df.loc[results_df['R2'].idxmax()]

print("\nBest Model based on MSE:")
print(best_model_mse)

print("\nBest Model based on R2:")
print(best_model_r2)
```

```
Best Model based on MSE:
Model    K-Nearest Neighbors Regression
MSE                            0.000976
R2                             0.610333
Name: 7, dtype: object

Best Model based on R2:
Model    K-Nearest Neighbors Regression
MSE                            0.000976
R2                             0.610333
Name: 7, dtype: object
```

[36]:

[36]: