Lab Assignment 1

Name: Soumyadeep Ganguly

Reg No.: 24MDT0082

In [40]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Load the file to your python program in to a data frame DF1

In [3]:
```
DF1 = pd.read_csv('housepricedata.csv')
```

Out[3]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	BedroomAbvGr
0	8450	7	5	856	2	1	3
1	9600	6	8	1262	2	0	3
2	11250	7	5	920	2	1	3
3	9550	7	5	756	1	0	3
4	14260	8	5	1145	2	1	4

Print the following details of the data frame DF1.

1. First five observations from your dataset
2. Last five observations from our dataset
3. Shape of your dataset
4. info of your dataset

In [4]:
```
# 1. First five observations from your dataset
DF1.head()
```

Out[4]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	BedroomAbvGr
0	8450	7	5	856	2	1	3
1	9600	6	8	1262	2	0	3
2	11250	7	5	920	2	1	3
3	9550	7	5	756	1	0	3
4	14260	8	5	1145	2	1	4

In [5]:
```
# 2. Last five observations from our dataset
DF1.tail()
```

Out[5]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	BedroomAb
1455	7917	6	5	953	2	1	
1456	13175	6	6	1542	2	0	
1457	9042	7	9	1152	2	0	
1458	9717	5	6	1078	1	0	
1459	9937	5	6	1256	1	1	

In [6]:
```
# 3. Shape of your dataset
DF1.shape
```

Out[6]: (1460, 11)

In [9]:
```
# 4. info of your dataset

DF1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   LotArea           1460 non-null   int64  
 1   OverallQual       1460 non-null   int64  
 2   OverallCond       1460 non-null   int64  
 3   TotalBsmtSF       1460 non-null   int64  
 4   FullBath          1460 non-null   int64  
 5   HalfBath          1460 non-null   int64  
 6   BedroomAbvGr      1460 non-null   int64  
 7   TotRmsAbvGrd      1460 non-null   int64  
 8   Fireplaces        1460 non-null   int64  
 9   GarageArea        1454 non-null   float64
 10  Abovemedianprice  1460 non-null   object 
dtypes: float64(1), int64(9), object(1)
memory usage: 125.6+ KB
```

Create a new dataframe DF2 with the first 100 observations of your dataset with only the columns 'LotArea' and 'BedroomAbvGr' [Use iloc operator]

In [15]:
```
DF2 = DF1.loc[:100, ['LotArea','BedroomAbvGr']]
DF2
```

Out[15]:

	LotArea	BedroomAbvGr
0	8450	3
1	9600	3
2	11250	3
3	9550	3
4	14260	4
...	...	...
96	10264	3
97	10921	3
98	10625	2
99	9320	3
100	10603	3

101 rows × 2 columns

Write or export your dataset DF2 to a csv file DF11.csv and save it.

In [17]:
```
DF2.to_csv("DF11.csv")
```

Find the maximum and minimum of the 'LotArea' column for your dataset DF1

In [18]:
```
DF1['LotArea'].max()
```

Out[18]: np.int64(215245)

In [19]:
```
DF1['LotArea'].min()
```

Out[19]: np.int64(1300)

Find the observations from your dataset with LotArea > 10650 from your DF1 data frame.

In [20]:
```
DF1[DF1['LotArea']>10650]
```

Out[20]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	BedroomAb
2	11250	7	5	920	2	1	
4	14260	8	5	1145	2	1	
5	14115	5	5	796	1	1	
10	11200	5	5	1040	1	0	
11	11924	9	5	1175	3	0	
...	...	...	...	...	...	...	
1442	11003	10	5	1017	2	1	
1446	26142	5	7	1188	1	0	
1448	11767	4	7	560	1	1	
1453	17217	5	5	1140	1	0	
1456	13175	6	6	1542	2	0	

502 rows × 11 columns

Find the mean, median of your column 'TotalBsmtSF' and find the unique entries

In [21]:
```
DF1['TotalBsmtSF'].mean()
```

Out[21]:
```
np.float64(1057.4294520547944)
```

In [22]:
```
DF1['TotalBsmtSF'].median()
```

Out[22]:
```
np.float64(991.5)
```

In [23]:
```
DF1['TotalBsmtSF'].unique()
```

Out[23]:
```
array([ 856, 1262,  920,  756, 1145,  796, 1686, 1107,  952,  991, 1040,
       1175,  912, 1494, 1253,  832, 1004,    0, 1114, 1029, 1158,  637,
       1777, 1060, 1566,  900, 1704, 1484,  520,  649, 1228, 1234, 1398,
       1561, 1117, 1097, 1297, 1057, 1088, 1350,  840,  938, 1150, 1752,
       1434, 1656,  736,  955,  794,  816, 1842,  384, 1425,  970,  860,
       1410,  780,  530, 1370,  576, 1143, 1947, 1453,  747, 1304, 2223,
        845, 1086,  462,  672, 1768,  440,  896, 1237, 1563, 1065, 1288,
        684,  612, 1013,  990, 1235,  876, 1214,  824,  680, 1588,  960,
        458,  950, 1610,  741, 1226, 1053,  641,  789,  793, 1844,  994,
       1264, 1809, 1028,  729, 1092, 1125, 1673,  728,  732, 1080, 1199,
       1362, 1078,  660, 1008,  924,  992, 1063, 1267, 1461, 1907,  928,
        864, 1734,  910, 1490, 1728,  715,  884,  969, 1710,  825, 1602,
       1200,  572,  774, 1392, 1232, 1572, 1541,  882, 1149,  644, 1617,
       1582,  720, 1064, 1606, 1202, 1151, 1052, 2216,  968,  504, 1188,
       1593,  853,  725, 1431,  855, 1726, 1360,  755, 1713, 1121, 1196,
        617,  848, 1424, 1140, 1100, 1157, 1212,  689, 1070, 1436,  686,
        798, 1248, 1498, 1010,  713, 2392,  630, 1203,  483, 1373, 1194,
       1462,  894, 1414,  996, 1694,  735,  540,  626,  948, 1845, 1020,
       1367, 1444, 1573, 1302, 1314,  975, 1604,  963, 1482,  506,  926,
       1422,  802,  740, 1095, 1385, 1152, 1240, 1560, 2121, 1160,  807,
       1468, 1575,  625,  858,  698, 1079,  768,  795, 1416, 1003,  702,
       1165, 1470, 2000,  700,  319,  861, 1896,  697,  972, 2136,  716,
       1347, 1372, 1249, 1136, 1502, 1162,  710, 1719, 1383,  844,  596,
       1056, 3206, 1358,  943, 1499, 1922, 1536, 1208, 1215,  967,  721,
       1684,  536,  958, 1478,  764, 1848, 1869,  616,  624,  940, 1142,
       1062,  888,  883, 1394, 1099, 1268,  953,  744,  608,  847,  683,
        870, 1580, 1856,  982, 1026, 1293,  939,  784, 1256,  658, 1041,
       1682,  804,  788, 1144,  961, 1260, 1310, 1141,  806, 1281, 1034,
       1276, 1340, 1344,  988,  651, 1518,  907,  901,  765,  799,  648,
       3094, 1440, 1258,  915, 1517,  930,  813, 1533,  872, 1242, 1364,
        588,  709,  560, 1375, 1277, 1626, 1488,  808,  547, 1976, 2153,
       1705, 1833, 1792, 1216,  999, 1113, 1073,  954,  264, 1269,  190,
       3200,  866, 1501,  777, 1218, 1368, 1084, 2006, 1244, 3138, 1379,
       1257, 1452,  528, 2035,  611,  707,  880, 1051, 1581, 1838, 1650,
        723,  654, 1204, 1069, 1709,  998,  993, 1374, 1389, 1163, 1122,
       1496,  846,  372, 1164, 1050, 2042, 1868, 1437,  742,  770, 1722,
       1814, 1430, 1058,  908,  600,  965, 1032, 1299, 1120,  936,  783,
       1822, 1522,  980, 1116,  978, 1156,  636, 1554, 1386,  811, 1520,
       1952, 1766,  981, 1094, 2109,  525,  776, 1486, 1629, 1138, 2077,
       1406, 1021, 1408,  738, 1477, 2046,  923, 1291, 1195, 1190,  874,
        551, 1419, 2444, 1210,  927, 1112, 1391, 1800,  360, 1473, 1643,
       1324,  270,  859,  718, 1176, 1311,  971, 1742,  941, 1698, 1584,
       1595,  868, 1153,  893, 1349, 1337, 1720, 1479, 1030, 1318, 1252,
        983, 1860,  836, 1935, 1614,  761, 1413,  956,  712,  650,  773,
       1926,  731, 1417, 1024,  849, 1442, 1649, 1568,  778, 1489, 2078,
       1454, 1516, 1067, 1559, 1127, 1390, 1273,  918, 1763, 1090, 1054,
       1039, 1148, 1002, 1638,  105,  676, 1184, 1109,  892, 2217, 1505,
       1059,  951, 2330, 1670, 1623, 1017, 1105, 1001,  546,  480, 1134,
       1104, 1272, 1316, 1126, 1181, 1753,  964, 1466,  925, 1905, 1500,
        585, 1632,  819, 1616, 1161,  828,  945,  979,  561,  696, 1330,
        817, 1098, 1428,  673, 1241,  944, 1225, 1266, 1128,  485, 1930,
       1396,  916,  822,  750, 1700, 1007, 1187,  691, 1574, 1680, 1346,
        985, 1657,  602, 1022, 1082,  810, 1504, 1220, 1132, 1565, 1338,
       1654, 1620, 1055,  800, 1306, 1475, 2524, 1992, 1193,  973,  854,
        662, 1103, 1154,  942, 1048,  727,  690, 1096, 1459, 1251, 1247,
       1074, 1271,  290,  655, 1463, 1836,  803,  833,  408,  533, 1012,
       1552, 1005, 1530,  974, 1567, 1006, 1042, 1298,  704,  932, 1219,
       1296, 1198,  959, 1261, 1598, 1683,  818, 1600, 2396, 1624,  831,
       1224,  663,  879,  815, 1630, 2158,  931, 1660,  559, 1300, 1702,
       1075, 1361, 1106, 1476, 1689, 2076,  792, 2110, 1405, 1192,  746,
```

```
1986,  841, 2002, 1332,  935, 1019,  661, 1309, 1328, 1085, 6110,
1246,  771,  976, 1652, 1278, 1902, 1274, 1393, 1622, 1352,  420,
1795,  544, 1510,  911,  693, 1284, 1732, 2033,  570, 1980,  814,
 873,  757, 1108, 2633, 1571,  984, 1205,  714, 1746, 1525,  482,
1356,  862,  839, 1286, 1485, 1594,  622,  791,  708, 1223,  913,
 656, 1319, 1932,  539, 1221, 1542])
```

Sort the dataset DF1 according to the 'TotalBsmtSF' column of your dataset DF1 in ascending and descending order

In [24]:
```
DF1.sort_values(by='TotalBsmtSF', ascending=True)
```

Out[24]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	BedroomAb
39	6040	4	5	0	2	0	
736	8544	3	4	0	2	0	
1179	8335	5	5	0	1	0	
371	17120	4	4	0	2	0	
392	8339	5	7	0	1	0	
...	...	...	...	...	...	...	
440	15431	10	5	3094	2	0	
523	40094	10	5	3138	3	1	
496	12692	8	5	3200	3	0	
332	10655	8	5	3206	2	0	
1298	63887	10	5	6110	2	1	

1460 rows × 11 columns

In [25]:
```
DF1.sort_values(by='TotalBsmtSF', ascending=False)
```

Out[25]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	BedroomAb
1298	63887	10	5	6110	2	1	
332	10655	8	5	3206	2	0	
496	12692	8	5	3200	3	0	
523	40094	10	5	3138	3	1	
440	15431	10	5	3094	2	0	
...	...	...	...	...	...	...	
868	14762	5	6	0	2	0	
362	7301	7	5	0	3	0	
897	7018	5	5	0	2	0	
371	17120	4	4	0	2	0	
39	6040	4	5	0	2	0	

1460 rows × 11 columns

Find the empty cells in 'GarageArea' column of your dataset DF1 and fill it with the average value of the column 'GarageArea'

In [30]:
```
DF1[DF1['GarageArea'].isna()]
```

Out[30]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	BedroomAbvG
2	11250	7	5	920	2	1	
3	9550	7	5	756	1	0	
12	12968	5	6	912	1	0	
13	10652	7	5	1494	2	0	
14	10920	6	5	1253	1	1	
15	6120	7	8	832	1	0	

In [33]:
```
DF1['GarageArea'].fillna(DF1['GarageArea'].mean(), inplace=True)
```

```
C:\Users\sambh\AppData\Local\Temp\ipykernel_2512\853437331.py:1: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assi
gnment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  DF1['GarageArea'].fillna(DF1['GarageArea'].mean(), inplace=True)
```

In [36]:
```
DF1['GarageArea'].isna().sum()
```

Out[36]:
```
np.int64(0)
```

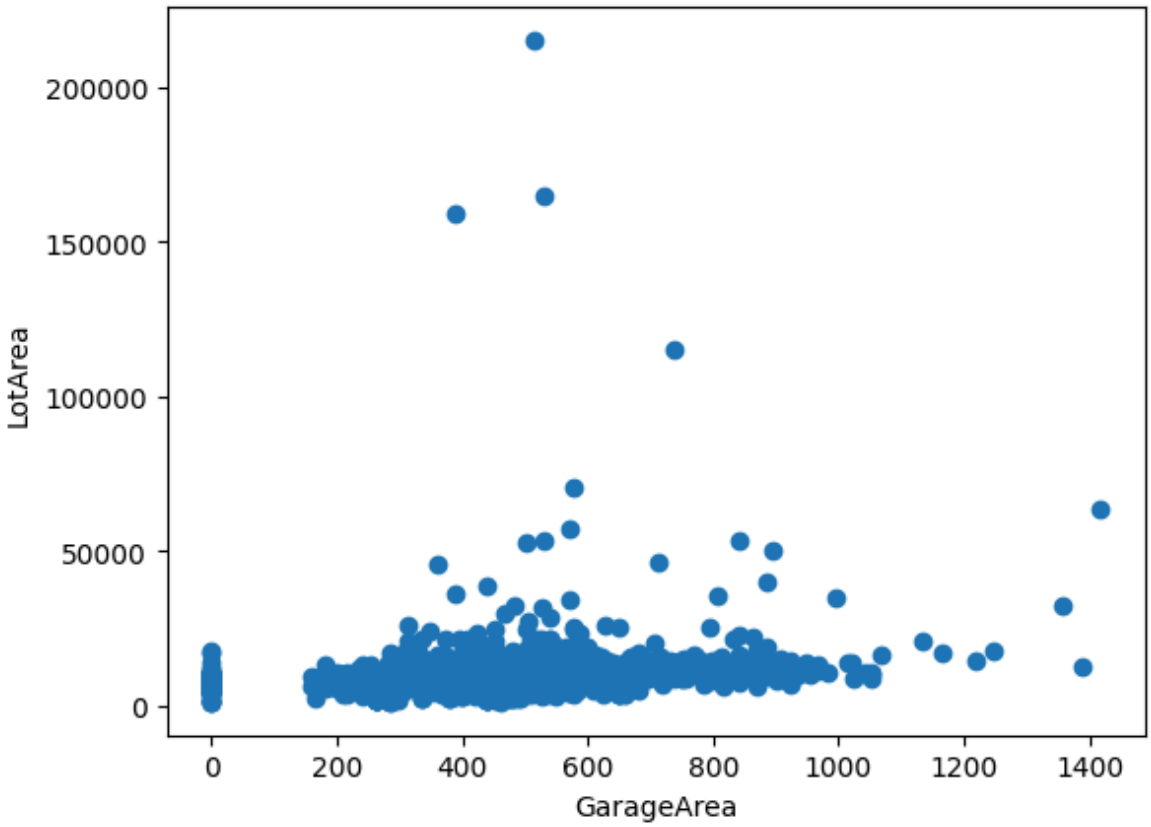
Replace the column named Above median price in your dataframe with 1's where ever you habe Yes and 0 where ever you have No.

In [37]:
```
DF1['Abovemedianprice'] = DF1['Abovemedianprice'].replace({'Yes':1,'no':0})
```

```
C:\Users\sambh\AppData\Local\Temp\ipykernel_2512\1000514148.py:1: FutureWarning:
Downcasting behavior in `replace` is deprecated and will be removed in a future v
ersion. To retain the old behavior, explicitly call `result.infer_objects(copy=Fa
lse)`. To opt-in to the future behavior, set `pd.set_option('future.no_silent_dow
ncasting', True)`
  DF1['Abovemedianprice'] = DF1['Abovemedianprice'].replace({'Yes':1,'no':0})
```

Draw a scatterplot with columns 'GarageArea' on x axis and 'LotArea' on y-axis

In [42]:
```
plt.scatter(DF1['GarageArea'], DF1['LotArea'])
plt.xlabel("GarageArea")
plt.ylabel("LotArea")
plt.show()
```



Drop the column 'GarageArea' from your dataset DF1

In [44]:
```
DF1.drop(['GarageArea'], axis=1)
```

Out[44]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	BedroomAb
0	8450	7	5	856	2	1	
1	9600	6	8	1262	2	0	
2	11250	7	5	920	2	1	
3	9550	7	5	756	1	0	
4	14260	8	5	1145	2	1	
...	...	...	...	...	...	...	
1455	7917	6	5	953	2	1	
1456	13175	6	6	1542	2	0	
1457	9042	7	9	1152	2	0	
1458	9717	5	6	1078	1	0	
1459	9937	5	6	1256	1	1	

1460 rows × 10 columns

Q1. Now, normalize the columns of the dataset1 using the above technique and save it to a new

csv file DF3.csv

In [46]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [49]:
```
scaler = MinMaxScaler()
x = DF1.loc[:, DF1.columns[:]]
scaled_data = scaler.fit_transform(x)
mydf = pd.DataFrame(scaled_data, columns=DF1.columns)
mydf
```

Out[49]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	BedroomAb
0	0.033420	0.666667	0.500	0.140098	0.666667	0.5	0
1	0.038795	0.555556	0.875	0.206547	0.666667	0.0	0
2	0.046507	0.666667	0.500	0.150573	0.666667	0.5	0
3	0.038561	0.666667	0.500	0.123732	0.333333	0.0	0
4	0.060576	0.777778	0.500	0.187398	0.666667	0.5	0
...	...	...	...	...	...	...	
1455	0.030929	0.555556	0.500	0.155974	0.666667	0.5	0
1456	0.055505	0.555556	0.625	0.252373	0.666667	0.0	0
1457	0.036187	0.666667	1.000	0.188543	0.666667	0.0	0
1458	0.039342	0.444444	0.625	0.176432	0.333333	0.0	0
1459	0.040370	0.444444	0.625	0.205565	0.333333	0.5	0

1460 rows × 11 columns

In [50]:
```
mydf.to_csv('DF3.csv')
```

Q2. Now normalize the whole data to the range (2,3) using Min-Max normalization

In [51]:
```
scaler2 = MinMaxScaler(feature_range=(2,3))
scaled_data2 = scaler.fit_transform(x)
mydf2 = pd.DataFrame(scaled_data2, columns=DF1.columns)
mydf2
```

Out[51]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	BedroomAb
0	0.033420	0.666667	0.500	0.140098	0.666667	0.5	0
1	0.038795	0.555556	0.875	0.206547	0.666667	0.0	0
2	0.046507	0.666667	0.500	0.150573	0.666667	0.5	0
3	0.038561	0.666667	0.500	0.123732	0.333333	0.0	0
4	0.060576	0.777778	0.500	0.187398	0.666667	0.5	0
...	...	...	...	...	...	...	
1455	0.030929	0.555556	0.500	0.155974	0.666667	0.5	0
1456	0.055505	0.555556	0.625	0.252373	0.666667	0.0	0
1457	0.036187	0.666667	1.000	0.188543	0.666667	0.0	0
1458	0.039342	0.444444	0.625	0.176432	0.333333	0.0	0
1459	0.040370	0.444444	0.625	0.205565	0.333333	0.5	0

1460 rows × 11 columns

DECIMAL SCALING

In [54]:
```
j = len(str(DF1['LotArea'].max()))
DF3 = DF1.copy()
DF3['LotArea'] = DF1['LotArea']/10**j
DF3
```

Out[54]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	BedroomAb
0	0.008450	7	5	856	2	1	
1	0.009600	6	8	1262	2	0	
2	0.011250	7	5	920	2	1	
3	0.009550	7	5	756	1	0	
4	0.014260	8	5	1145	2	1	
...	...	...	...	...	...	...	
1455	0.007917	6	5	953	2	1	
1456	0.013175	6	6	1542	2	0	
1457	0.009042	7	9	1152	2	0	
1458	0.009717	5	6	1078	1	0	
1459	0.009937	5	6	1256	1	1	

1460 rows × 11 columns

Q3. Now do decimal scaling for the original column data of the column LotArea of your initial dataframe and print the results.

ZScore normalization

In [57]:
```
from sklearn.preprocessing import StandardScaler
zscaler = StandardScaler()

scaled_data3 = zscaler.fit_transform(x)
mydf3 = pd.DataFrame(scaled_data3, columns=DF1.columns)
mydf3
```

Out[57]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	Bedroom
0	-0.207142	0.651479	-0.517200	-0.459303	0.789741	1.227585	0.1
1	-0.091886	-0.071836	2.179628	0.466465	0.789741	-0.761621	0.1
2	0.073480	0.651479	-0.517200	-0.313369	0.789741	1.227585	0.1
3	-0.096897	0.651479	-0.517200	-0.687324	-1.026041	-0.761621	0.1
4	0.375148	1.374795	-0.517200	0.199680	0.789741	1.227585	1.3
...	...	...	...	...	...	...	
1455	-0.260560	-0.071836	-0.517200	-0.238122	0.789741	1.227585	0.1
1456	0.266407	-0.071836	0.381743	1.104925	0.789741	-0.761621	0.1
1457	-0.147810	0.651479	3.078570	0.215641	0.789741	-0.761621	1.3
1458	-0.080160	-0.795151	0.381743	0.046905	-1.026041	-0.761621	-1.0
1459	-0.058112	-0.795151	0.381743	0.452784	-1.026041	1.227585	0.1

1460 rows × 11 columns

Q4. Now try to standardize the whole data in the dataframe and print the dataframe.

Train-Test Data Splitting

In [58]:
```
from sklearn.model_selection import train_test_split

x = DF1.drop('LotArea',axis=1)
y = DF1['LotArea']

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3,random_sta

x_train
```

Out[58]:

	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	BedroomAbvGr	TotR
135	7	6	1304	2	0	3	
1452	5	5	547	1	0	2	
762	7	5	756	2	1	3	
932	9	5	1905	2	0	3	
435	7	6	799	2	1	3	
...	...	...	...	...	...	...	
1095	6	5	1314	2	0	3	
1130	4	3	1122	2	0	4	
1294	5	7	864	1	0	2	
860	7	8	912	1	1	3	
1126	7	5	1373	2	0	2	

1022 rows × 10 columns

In [59]:
```
x_test
```

Out[59]:

	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	BedroomAbvGr	TotR
892	6	8	1059	1	0	3	
1105	8	5	1463	2	1	3	
413	5	6	1008	1	0	2	
522	6	7	1004	2	0	3	
1036	9	5	1620	2	0	2	
...	...	...	...	...	...	...	
331	5	6	1056	1	0	3	
323	3	8	1162	1	0	3	
650	7	6	813	2	1	3	
439	6	8	684	1	0	3	
798	9	5	1926	3	1	4	

438 rows × 10 columns

In [60]:
```
y_train
```

Out[60]:
```
135     10400
1452     3675
762      8640
932     11670
435     10667
        ...
1095     9317
1130     7804
1294     8172
860      7642
1126     3684
Name: LotArea, Length: 1022, dtype: int64
```

In [61]:
```
y_test
```

Out[61]:
```
892      8414
1105    12256
413      8960
522      5000
1036    12898
        ...
331      8176
323      5820
650      8125
439     12354
798     13518
Name: LotArea, Length: 438, dtype: int64
```

In [ ]:
```

```