TK_18_dec

January 5, 2025

1 Data Mining and Machine Learning Lab

1.1 Experiment 1

1.1.1 Importing the necessary libraries

[15]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

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

[16]:
```
df1 = pd.read_csv(r"D:\study material\VIT_Data_Science\Winter_Sem\Data Mining␣
↪and Machine Learning Lab\Class_notes\18_Dec_ML_exp1\housepricedata.csv")
```

[17]:
```
df1
```

[17]:
```
      LotArea  OverallQual  OverallCond  TotalBsmtSF  FullBath  HalfBath  \
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

      BedroomAbvGr  TotRmsAbvGrd  Fireplaces  GarageArea  AboveMedianPrice
0                3             8           0         548                 1
1                3             6           1         460                 1
2                3             6           1         608                 1
3                3             7           1         642                 0
4                4             9           1         836                 1
…               …             …          …          …                …
1455             3             7           1         460                 1
```

1456	3	7	2	500	1
1457	4	9	2	252	1
1458	2	5	0	240	0
1459	3	6	0	276	0

[1460 rows x 11 columns]

1.1.3 First five observations from your dataset

[18]:
```
df1.head(5)
```

[18]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	\
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	

	BedroomAbvGr	TotRmsAbvGrd	Fireplaces	GarageArea	AboveMedianPrice
0	3	8	0	548	1
1	3	6	1	460	1
2	3	6	1	608	1
3	3	7	1	642	0
4	4	9	1	836	1

1.1.4 Last five observations from our dataset

[19]:
```
df1.tail(5)
```

[19]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	\
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	

	BedroomAbvGr	TotRmsAbvGrd	Fireplaces	GarageArea	AboveMedianPrice
1455	3	7	1	460	1
1456	3	7	2	500	1
1457	4	9	2	252	1
1458	2	5	0	240	0
1459	3	6	0	276	0

1.1.5 Shape of your dataset

[20]:
```
df1.shape
```

[20]: (1460, 11)

1.1.6 info of your dataset

[21]:
```
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        1460 non-null   int64
 10  AboveMedianPrice  1460 non-null   int64
dtypes: int64(11)
memory usage: 125.6 KB
```

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

[22]:
```
df2 = (df1.iloc[0:100])[['LotArea','BedroomAbvGr']]
```

[23]:
```
df2
```

[23]:

	LotArea	BedroomAbvGr
0	8450	3
1	9600	3
2	11250	3
3	9550	3
4	14260	4
..	…	…
95	9765	3
96	10264	3
97	10921	3
98	10625	2
99	9320	3

[100 rows x 2 columns]

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

[24]:
```
df2.to_csv('DF11.csv')
```

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

[25]:
```
print("Maximum of LotArea column is:",df1['LotArea'].max())
```

```
Maximum of LotArea column is: 215245
```

[26]:
```
print("Minimum of LotArea column is:",df1['LotArea'].min())
```

```
Minimum of LotArea column is: 1300
```

1.1.10 Find the observations from your dataset with LotArea > 10650 from your DF1 dataframe.

[27]:
```
df1[df1['LotArea']>10650]
```

[27]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	\
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	

	BedroomAbvGr	TotRmsAbvGrd	Fireplaces	GarageArea	AboveMedianPrice
2	3	6	1	608	1
4	4	9	1	836	1
5	1	5	0	480	0
10	3	5	0	384	0
11	4	11	2	736	1
…	…	…	…	…	…
1442	3	10	1	812	1
1446	3	6	0	312	0
1448	2	6	0	384	0
1453	3	6	0	0	0
1456	3	7	2	500	1

[502 rows x 11 columns]

1.1.11 Find the mean, median of your column TotalBsmtSF and find the unique entries (non-repeated ones)

[28]:
```
print("Mean of the column TotalBsmtSf is:",df1.TotalBsmtSF.mean())
```

```
Mean of the column TotalBsmtSf is: 1057.4294520547944
```

[29]:
```
print("Median of the column TotalBsmtSf is:",df1.TotalBsmtSF.median())
```

```
Median of the column TotalBsmtSf is: 991.5
```

[30]:
```
print("Unique of the column TotalBsmtSf is:\n",df1.TotalBsmtSF.unique())
```

```
Unique of the column TotalBsmtSf is:
 [ 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]
```

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

[31]:
```python
##sorting in descending order
df1.sort_values(by=['TotalBsmtSF'],ascending=False)
```

[31]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	\
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	
…	…	…	…	…	…	…	
1412	7200	4	5	0	2	0	
1179	8335	5	5	0	1	0	
102	7018	5	5	0	2	0	
259	12702	5	5	0	1	0	
1048	21750	5	4	0	1	0	

	BedroomAbvGr	TotRmsAbvGrd	Fireplaces	GarageArea	AboveMedianPrice
1298	3	12	3	1418	0
332	3	7	1	880	1
496	4	10	1	546	1
523	3	11	1	884	1
440	2	10	2	672	1
…	…	…	…	…	…
1412	2	6	0	420	0
1179	3	5	1	0	0
102	4	8	0	410	0

```
259                2          4         0         308                0
1048               3          9         1         336                0

[1460 rows x 11 columns]
```

[32]:
```
#sorted in ascending order
df1.sort_values(by=['TotalBsmtSF'])
```

[32]:
```
      LotArea  OverallQual  OverallCond  TotalBsmtSF  FullBath  HalfBath  \
646      7200            5            5            0         1         0
1035    11500            4            3            0         1         0
392      8339            5            7            0         1         0
749      8405            4            3            0         2         0
1011     9825            5            5            0         2         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

      BedroomAbvGr  TotRmsAbvGrd  Fireplaces  GarageArea  AboveMedianPrice
646              3             7           0         420                 0
1035             3             5           0         290                 0
392              3             5           0         294                 0
749              4             9           0         240                 0
1011             4             8           0           0                 0
…             …             …           …          …                …
440              2            10           2         672                 1
523              3            11           1         884                 1
496              4            10           1         546                 1
332              3             7           1         880                 1
1298             3            12           3        1418                 0

[1460 rows x 11 columns]
```

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

[33]:
```
df1['GarageArea'].isnull().sum()
```

[33]: 0

[34]:
```
df1['GarageArea'].fillna(df1['GarageArea'].mean(),inplace=True)
```

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

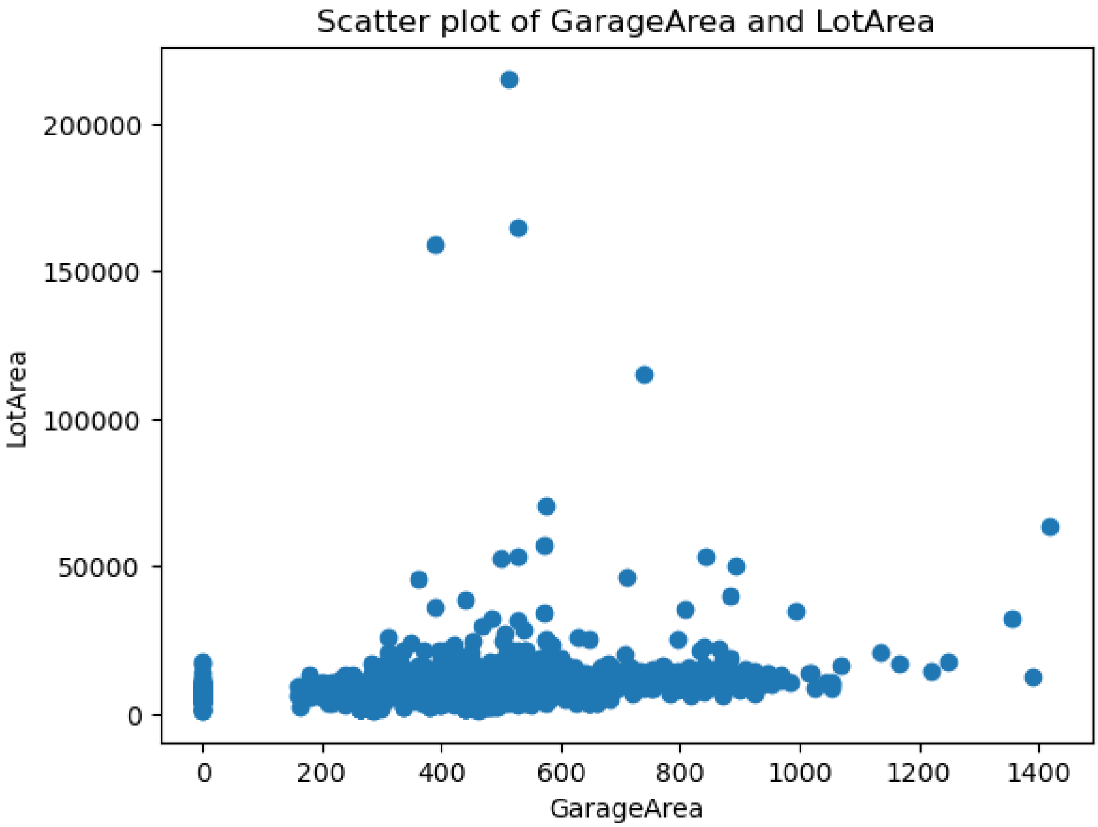
[35]:
```
df1['AboveMedianPrice'].replace({'Yes':1,'no':0},inplace=True)
```

[36]:
```
df1['AboveMedianPrice']
```

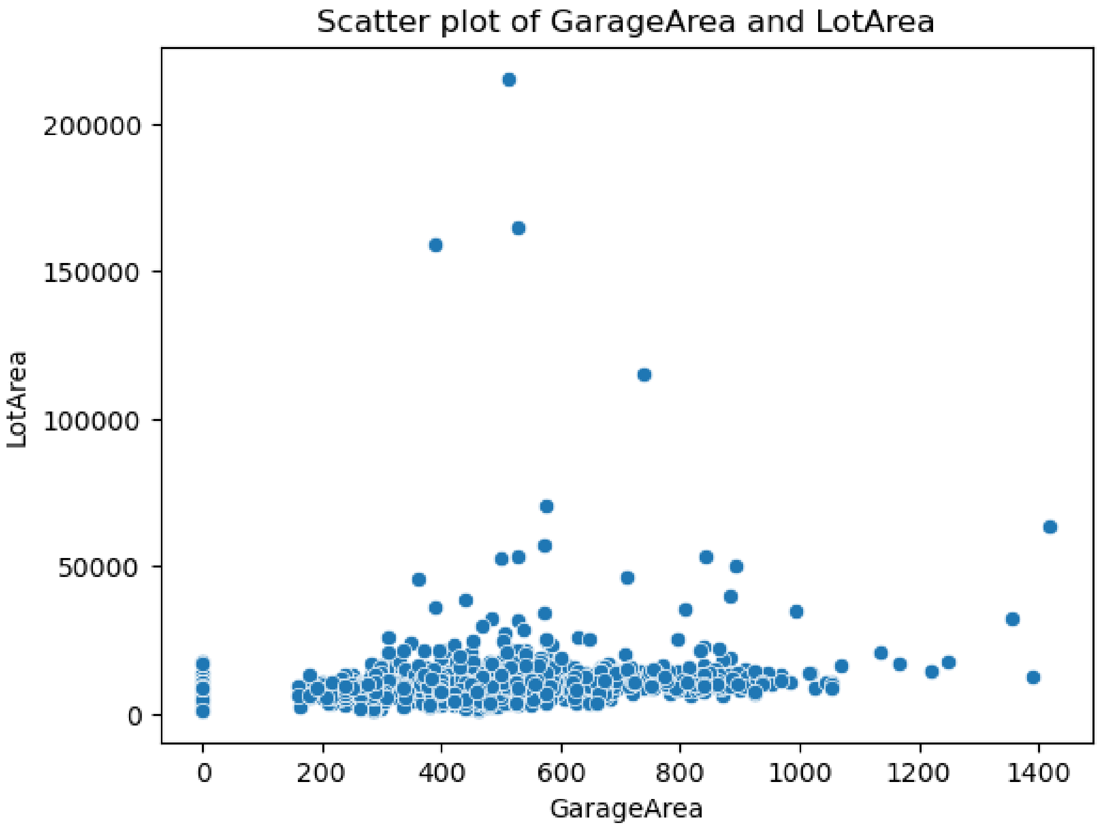
[36]:
```
0       1
1       1
2       1
3       0
4       1
       ..
1455    1
1456    1
1457    1
1458    0
1459    0
Name: AboveMedianPrice, Length: 1460, dtype: int64
```

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

[37]:
```
plt.scatter(df1['GarageArea'],df1['LotArea'])
plt.title("Scatter plot of GarageArea and LotArea")
plt.xlabel("GarageArea")
plt.ylabel("LotArea")
plt.show()
```



[38]:
```
import seaborn as sns
sns.scatterplot(x=df1['GarageArea'],y = df1['LotArea'])
plt.title("Scatter plot of GarageArea and LotArea")
plt.xlabel("GarageArea")
plt.ylabel("LotArea")
plt.show()
```



1.1.16 Drop the column 'GarageArea' from your dataset DF1

[39]:
```
df1.drop('GarageArea',axis=1,inplace=True)
```

[40]:
```
df1
```

[40]:
```
      LotArea  OverallQual  OverallCond  TotalBsmtSF  FullBath  HalfBath  \
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

      BedroomAbvGr  TotRmsAbvGrd  Fireplaces  AboveMedianPrice
```

```
0                 3             8           0                 1
1                 3             6           1                 1
2                 3             6           1                 1
3                 3             7           1                 0
4                 4             9           1                 1
…             …             …          …               …
1455              3             7           1                 1
1456              3             7           2                 1
1457              4             9           2                 1
1458              2             5           0                 0
1459              3             6           0                 0

[1460 rows x 10 columns]
```

1.2 Q1. Now, normalize the columns of the dataset1 using the above technique and save it to a new csv file DF3.csv

[41]:
```python
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler()
col_name = df1.columns[:]
x = df1.loc[:, col_name]
x = pd.DataFrame(data = min_max_scaler.fit_transform(x), columns = col_name)
print(x)
x.to_csv('df3.csv')
```

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

      BedroomAbvGr  TotRmsAbvGrd  Fireplaces  AboveMedianPrice
0            0.375      0.500000    0.000000               1.0
1            0.375      0.333333    0.333333               1.0
2            0.375      0.333333    0.333333               1.0
3            0.375      0.416667    0.333333               0.0
4            0.500      0.583333    0.333333               1.0
…             …          …          …                …
1455         0.375      0.416667    0.333333               1.0
1456         0.375      0.416667    0.666667               1.0
1457         0.500      0.583333    0.666667               1.0
```

```
1458          0.250     0.250000    0.000000                0.0
1459          0.375     0.333333    0.000000                0.0

[1460 rows x 10 columns]
```

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

[42]:
```python
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler(feature_range=(2,3))
col_name = df1.columns[:]
x = df1.loc[:, col_name]
x = pd.DataFrame(data = min_max_scaler.fit_transform(x), columns = col_name)
print(x)
```

```
       LotArea  OverallQual  OverallCond  TotalBsmtSF  FullBath  HalfBath  \
0     2.033420     2.666667        2.500     2.140098  2.666667       2.5
1     2.038795     2.555556        2.875     2.206547  2.666667       2.0
2     2.046507     2.666667        2.500     2.150573  2.666667       2.5
3     2.038561     2.666667        2.500     2.123732  2.333333       2.0
4     2.060576     2.777778        2.500     2.187398  2.666667       2.5
…          …           …           …           …         …
1455  2.030929     2.555556        2.500     2.155974  2.666667       2.5
1456  2.055505     2.555556        2.625     2.252373  2.666667       2.0
1457  2.036187     2.666667        3.000     2.188543  2.666667       2.0
1458  2.039342     2.444444        2.625     2.176432  2.333333       2.0
1459  2.040370     2.444444        2.625     2.205565  2.333333       2.5

      BedroomAbvGr  TotRmsAbvGrd  Fireplaces  AboveMedianPrice
0            2.375      2.500000    2.000000               3.0
1            2.375      2.333333    2.333333               3.0
2            2.375      2.333333    2.333333               3.0
3            2.375      2.416667    2.333333               2.0
4            2.500      2.583333    2.333333               3.0
…              …             …           …                 …
1455         2.375      2.416667    2.333333               3.0
1456         2.375      2.416667    2.666667               3.0
1457         2.500      2.583333    2.666667               3.0
1458         2.250      2.250000    2.000000               2.0
1459         2.375      2.333333    2.000000               2.0

[1460 rows x 10 columns]
```

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

[43]:
```
max_abs_values = abs(df1.max())
```

[44]:
```
max_abs_values
```

[44]:
```
LotArea             215245
OverallQual             10
OverallCond              9
TotalBsmtSF           6110
FullBath                 3
HalfBath                 2
BedroomAbvGr             8
TotRmsAbvGrd            14
Fireplaces               3
AboveMedianPrice         1
dtype: int64
```

[45]:
```
df88 = df1['LotArea'].apply(lambda x : x/
 ↪10**(len(str(max_abs_values['LotArea']))))
```

[46]:
```
df88
```

[46]:
```
0       0.008450
1       0.009600
2       0.011250
3       0.009550
4       0.014260
          …
1455    0.007917
1456    0.013175
1457    0.009042
1458    0.009717
1459    0.009937
Name: LotArea, Length: 1460, dtype: float64
```

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

[47]:
```
from sklearn import preprocessing
standard_scaler = preprocessing.StandardScaler()
col_name = df1.columns
x = df1.loc[:, col_name]
x = pd.DataFrame(data = standard_scaler.fit_transform(x), columns = col_name)
print(x)
```

```
      LotArea  OverallQual  OverallCond  TotalBsmtSF  FullBath  HalfBath  \
0   -0.207142     0.651479    -0.517200    -0.459303  0.789741  1.227585
```

```
1    -0.091886    -0.071836     2.179628     0.466465  0.789741 -0.761621
2     0.073480     0.651479    -0.517200    -0.313369  0.789741  1.227585
3    -0.096897     0.651479    -0.517200    -0.687324 -1.026041 -0.761621
4     0.375148     1.374795    -0.517200     0.199680  0.789741  1.227585
…          …            …           …          …         …        …
1455 -0.260560    -0.071836    -0.517200    -0.238122  0.789741  1.227585
1456  0.266407    -0.071836     0.381743     1.104925  0.789741 -0.761621
1457 -0.147810     0.651479     3.078570     0.215641  0.789741 -0.761621
1458 -0.080160    -0.795151     0.381743     0.046905 -1.026041 -0.761621
1459 -0.058112    -0.795151     0.381743     0.452784 -1.026041  1.227585

      BedroomAbvGr  TotRmsAbvGrd  Fireplaces  AboveMedianPrice
0         0.163779      0.912210   -0.951226          1.002743
1         0.163779     -0.318683    0.600495          1.002743
2         0.163779     -0.318683    0.600495          1.002743
3         0.163779      0.296763    0.600495         -0.997264
4         1.390023      1.527656    0.600495          1.002743
…              …             …          …                …
1455      0.163779      0.296763    0.600495          1.002743
1456      0.163779      0.296763    2.152216          1.002743
1457      1.390023      1.527656    2.152216          1.002743
1458     -1.062465     -0.934130   -0.951226         -0.997264
1459      0.163779     -0.318683   -0.951226         -0.997264

[1460 rows x 10 columns]
```

[48]:
```
x
```

[48]:
```
       LotArea  OverallQual  OverallCond  TotalBsmtSF  FullBath  HalfBath  \
0    -0.207142     0.651479    -0.517200    -0.459303  0.789741  1.227585
1    -0.091886    -0.071836     2.179628     0.466465  0.789741 -0.761621
2     0.073480     0.651479    -0.517200    -0.313369  0.789741  1.227585
3    -0.096897     0.651479    -0.517200    -0.687324 -1.026041 -0.761621
4     0.375148     1.374795    -0.517200     0.199680  0.789741  1.227585
…          …            …           …          …         …        …
1455 -0.260560    -0.071836    -0.517200    -0.238122  0.789741  1.227585
1456  0.266407    -0.071836     0.381743     1.104925  0.789741 -0.761621
1457 -0.147810     0.651479     3.078570     0.215641  0.789741 -0.761621
1458 -0.080160    -0.795151     0.381743     0.046905 -1.026041 -0.761621
1459 -0.058112    -0.795151     0.381743     0.452784 -1.026041  1.227585

      BedroomAbvGr  TotRmsAbvGrd  Fireplaces  AboveMedianPrice
0         0.163779      0.912210   -0.951226          1.002743
1         0.163779     -0.318683    0.600495          1.002743
2         0.163779     -0.318683    0.600495          1.002743
3         0.163779      0.296763    0.600495         -0.997264
4         1.390023      1.527656    0.600495          1.002743
```

```
...          ...          ...         ...            ...
1455     0.163779     0.296763    0.600495        1.002743
1456     0.163779     0.296763    2.152216        1.002743
1457     1.390023     1.527656    2.152216        1.002743
1458    -1.062465    -0.934130   -0.951226       -0.997264
1459     0.163779    -0.318683   -0.951226       -0.997264

[1460 rows x 10 columns]
```

1.4 Train Test splitting of data for model training.

1.4.1 Now perfrom 70:30 train test split for our dataframe data with the target variable or output variable as LotArea and print the training and testing data which may be we can use for fitting a model for this data. Say like trying to predict the LotArea for a house based on all the other features.

[49]:
```python
from sklearn.model_selection import train_test_split
x =df1.drop('LotArea', axis=1)
y = df1['LotArea']
#Split the data into training and testing sets (70% train, 30% test)
x_train,x_test,y_train,y_test = train_test_split(x, y, test_size=0.3,␣
 ↪random_state=42)
#Print the testing and training data
print("Training Features:\n", x_train)
print("Testing Features:\n", x_test)
print("Training Target:\n", y_train)
print("Testing Target:\n", y_test)
```

```
Training Features:
       OverallQual  OverallCond  TotalBsmtSF  FullBath  HalfBath  BedroomAbvGr
\
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

      TotRmsAbvGrd  Fireplaces  AboveMedianPrice
135              7           1                 1
1452             5           0                 0
762              7           0                 1
932              8           1                 1
```

```
435             6           1                 1
…             …          …               …
1095            6           1                 1
1130            7           2                 0
1294            5           0                 0
860             7           1                 1
1126            7           1                 1

[1022 rows x 9 columns]
Testing Features:
       OverallQual  OverallCond  TotalBsmtSF  FullBath  HalfBath  BedroomAbvGr
\
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

      TotRmsAbvGrd  Fireplaces  AboveMedianPrice
892              6           0                 0
1105             9           2                 1
413              5           1                 0
522              7           2                 0
1036             6           1                 1
…             …          …               …
331              6           0                 0
323              6           0                 0
650              7           0                 1
439              7           0                 0
798             11           2                 1

[438 rows x 9 columns]
Training Target:
 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
Testing Target:
 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
```

[50]:
```python
# Linear Regression model
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
model = LinearRegression()
model.fit(x_train, y_train)

# Predicting the LotArea for test data
y_pred = model.predict(x_test)

# Evaluating the model's performance
mse = mean_squared_error(y_test, y_pred)  # Mean Squared Error
r2 = r2_score(y_test, y_pred)            # R-squared score

print("Model Coefficients:", model.coef_)
print("Model Intercept:", model.intercept_)
print("Mean Squared Error:", mse)
print("R-squared Score:", r2)

results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print(results)
```

```
Model Coefficients: [-1784.57317854   535.39223068     6.34890747   105.64506875
  -350.6656404    499.86480782   514.75964975  3611.59648719
  2444.37464586]
Model Intercept: 3657.748961708652
Mean Squared Error: 31147397.821340438
R-squared Score: 0.03359678635359464
      Actual     Predicted
892     8414   8650.738138
1105   12256  17007.199705
413     8960  11637.704604
```

```
522      5000  15609.753689
1036    12898  10914.330395
…       …            …
331      8176   9345.480133
323      5820  14658.395143
650      8125   6947.662984
439     12354   6784.657486
798     13518  19797.199861

[438 rows x 2 columns]
```