

## Import and read the document

```
In [57]: #Import libraries
```

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
```

```
In [58]: print (os.getcwd()) #check the directory
```

```
C:\Users\Consultant\Documents\machine-learning
```

```
In [59]: os.chdir("C:/Users/Consultant/Documents/machine-learning") #change the directory
```

```
print (os.getcwd())
```

```
C:\Users\Consultant\Documents\machine-learning
```

```
In [60]: df=pd.read_csv('Anyinssan Nava Sánchez - raw_house_data.xlsx - Anyinssan Nava Sánchez - raw_house_data.csv.csv')  
df
```

Out[60]:

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitchen_features	fireplaces	floor_covering	HOA
0	21530491	5300000.0	85637	-110.378200	31.356362	2154.00	5272.00	1941	13	10	10500	0	None	None	None	None
1	21529082	4200000.0	85646	-111.045371	31.594213	1707.00	10422.36	1997	2	2	7300	0	None	None	None	None
2	3054672	4200000.0	85646	-111.040707	31.594844	1707.00	10482.00	1997	2	3	None	None	None	None	None	None
3	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1930	7	5	9019	4	None	None	None	None

```
In [61]: df.shape
```

Out[61]: (5000, 16)

```
In [62]: df.columns #check the names of columns
```

```
Out[62]: Index(['MLS', 'sold_price', 'zipcode', 'longitude', 'latitude', 'lot_acres',  
       'taxes', 'year_built', 'bedrooms', 'bathrooms', 'sqrt_ft', 'garage',  
       'kitchen_features', 'fireplaces', 'floor_covering', 'HOA'],  
      dtype='object')
```

```
In [63]: df.describe()
```

Out[63]:

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	firep
count	5.000000e+03	5.000000e+03	5000.000000	5000.000000	5000.000000	4990.000000	5.000000e+03	5000.000000	5000.000000	4975.00
mean	2.127070e+07	7.746262e+05	85723.025600	-110.912107	32.308512	4.661317	9.402828e+03	1992.32800	3.933800	1.88
std	2.398508e+06	3.185556e+05	38.061712	0.120629	0.178028	51.685230	1.729385e+05	65.48614	1.245362	1.13
min	3.042851e+06	1.690000e+05	85118.000000	-112.520168	31.356362	0.000000	0.000000e+00	0.00000	1.000000	0.00
25%	2.140718e+07	5.850000e+05	85718.000000	-110.979260	32.277484	0.580000	4.803605e+03	1987.00000	3.000000	1.00
50%	2.161469e+07	6.750000e+05	85737.000000	-110.923420	32.318517	0.990000	6.223760e+03	1999.00000	4.000000	2.00
75%	2.180480e+07	8.350000e+05	85749.000000	-110.859078	32.394334	1.757500	8.082830e+03	2006.00000	4.000000	3.00
max	2.192856e+07	5.300000e+06	86323.000000	-109.454637	34.927884	2154.000000	1.221508e+07	2019.00000	36.000000	9.00

```
In [64]: ddf=df.describe()  
ddf
```

Out[64]:

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	firep
count	5.000000e+03	5.000000e+03	5000.000000	5000.000000	5000.000000	4990.000000	5.000000e+03	5000.000000	5000.000000	4975.00
mean	2.127070e+07	7.746262e+05	85723.025600	-110.912107	32.308512	4.661317	9.402828e+03	1992.32800	3.933800	1.88
std	2.398508e+06	3.185556e+05	38.061712	0.120629	0.178028	51.685230	1.729385e+05	65.48614	1.245362	1.13
min	3.042851e+06	1.690000e+05	85118.000000	-112.520168	31.356362	0.000000	0.000000e+00	0.00000	1.000000	0.00
25%	2.140718e+07	5.850000e+05	85718.000000	-110.979260	32.277484	0.580000	4.803605e+03	1987.00000	3.000000	1.00
50%	2.161469e+07	6.750000e+05	85737.000000	-110.923420	32.318517	0.990000	6.223760e+03	1999.00000	4.000000	2.00
75%	2.180480e+07	8.350000e+05	85749.000000	-110.859078	32.394334	1.757500	8.082830e+03	2006.00000	4.000000	3.00
max	2.192856e+07	5.300000e+06	86323.000000	-109.454637	34.927884	2154.000000	1.221508e+07	2019.00000	36.000000	9.00

```
In [65]: a=df.columns.values.tolist()    ###check
b=ddf.columns.values.tolist()
for i in a:
    if i not in b:
        print(i)
```

```
bathrooms
sqrt_ft
garage
kitchen_features
floor_covering
HOA
```

```
In [66]: df['bathrooms'].unique()
```

```
Out[66]: array(['10', '2', '3', '5', '6', '4', '8', '7', '15', '4.5', '1', '9',
       '11', '18', '14', '3.5', 'None', '35', '2.5', '36'], dtype=object)
```

```
In [67]: df['sqrt_ft'].unique()
```

```
Out[67]: array(['10500', '7300', 'None', ..., '2106', '3601', '1772'], dtype=object)
```

```
In [68]: df['garage'].unique()
```

```
Out[68]: array(['0', 'None', '4', '3', '5', '2', '6', '15', '8', '7', '4.5', '3.5',
       '2.5', '1', '9', '22', '30', '12', '10', '11', '20', '13'],
      dtype=object)
```

```
In [70]: df['HOA'].unique()
```

```
Out[70]: array(['0', 'None', '55', '422', '220', '421', '141.67', '357', '148',  
'20,000', '142', '173', '167', '123', '300', '194', '1,717', '342',  
'240', '437', '112', '199', '178', '550', '258', '188', '169',  
'124', '320', '213', '153', '105', '159', '193', '69', '208', '50',  
'168', '191', '79', '157', '323', '1,100', '250', '158', '127',  
'149', '83', '118', '219', '88', '180', '259', '128', '117', '48',  
'130', '132', '135', '152', '33', '143', '5', '157.33', '129',  
'171', '273', '162', '131', '134', '211', '203', '190', '145',  
'83.33', '63', '177.34', '57', '115', '166', '184', '164', '125',  
'216', '195', '106', '214', '238', '212.88', '4', '30', '150',  
'19', '160', '138', '141', '146', '1,270', '15', '95', '19,480',  
'116', '311', '94', '21', '295', '57.33', '243', '23', '34', '215',  
'41', '18', '110', '233', '322', '43', '232', '100', '133', '4.16',  
'202', '700', '121', '40', '51', '179', '58', '102', '175', '97',  
'25', '139', '120', '137', '53', '2,000', '242', '49.43', '333',  
'270', '84', '8', '165', '506', '63.98', '35', '64', '258.08',  
'108', '187', '186', '1,600', '212', '12', '209', '107', '45',  
'52', '42', '145.83', '22', '96', '93', '122', '200', '241', '161',  
'37.5', '119.66', '155', '299', '154', '170', '221', '303', '140',  
'425', '172', '68', '75', '71', '44', '37', '6', '285', '67',  
'54.16', '9', '113', '17', '192', '1,010', '78', '609', '26',  
'213.88', '16', '54', '11', '39', '144', '56', '156', '263', '92',  
'119', '60', '76', '70', '516', '141.66', '343', '72', '126', '80',  
'247', '193.5', '20', '114', '43.75', '73', '317', '36.02',  
'1,290', '147', '32', '10', '332.66', '43.01', '198', '332.67',  
'69.16', '33.33', '16.67', '205', '58.33', '174', '249', '38',  
'66', '40.55', '2', '43.71', '103', '13', '185', '111', '98',  
'166.66', '8.33', '74', '49', '59', '89', '85', '62', '66.67',  
'151', '225.21', '65', '86', '109', '101', '81', '24', '77',  
'1,769', '50.42', '47', '73.72', '149.5', '188.33', '257', '73.33',  
'269', '4.17', '99', '225', '390', '40.77', '3', '194.51', '6.25',  
'55.58', '7', '87', '90', '53.34', '1,000', '275.08', '123.44',  
'267', '183', '136', '16.66', '97.66', '39.59', '58.36', '40.78',  
'62.5', '290', '330', '46', '36', '41.08', '5.5', '176', '1',  
'226', '15.41', '283', '82', '91', '252', '14', '275', '29', '324',  
'325', '87.66', '42.38', '5,900', '350', '233.33', '83.34',  
'8,333', '765', '68.66', '253', '15.45', '5.25', '88.33', '1,200',  
'60.5', '31', '189', '487.88', '20.83', '328', '168.92', '14.58',  
'368', '57.5', '61', '46.95', '750', '84.75', '34.17', '8.34',  
'5.41', '66.66', '45.9', '337', '202.75', '149.04', '134.5',  
'2.08', '35.83', '212.38', '10.83', '41.61', '925', '132.66',
```

```
'41.66', '500', '99.66', '18.75', '177', '234', '28', '104', '294',
'38.55', '63.33', '11.08', '210', '112.38', '5.4', '36.66',
'78.65'], dtype=object)
```

```
In [71]: df.isnull().sum() ##check null values
```

```
Out[71]: MLS          0
sold_price      0
zipcode         0
longitude        0
latitude         0
lot_acres       10
taxes           0
year_built      0
bedrooms        0
bathrooms       0
sqrt_ft         0
garage          0
kitchen_features 0
fireplaces      25
floor_covering   0
HOA             0
dtype: int64
```

In [72]: df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   MLS              5000 non-null    int64  
 1   sold_price       5000 non-null    float64 
 2   zipcode          5000 non-null    int64  
 3   longitude        5000 non-null    float64 
 4   latitude         5000 non-null    float64 
 5   lot_acres        4990 non-null    float64 
 6   taxes            5000 non-null    float64 
 7   year_built      5000 non-null    int64  
 8   bedrooms         5000 non-null    int64  
 9   bathrooms         5000 non-null    object  
 10  sqrt_ft          5000 non-null    object  
 11  garage           5000 non-null    object  
 12  kitchen_features 5000 non-null    object  
 13  fireplaces       4975 non-null    float64 
 14  floor_covering  5000 non-null    object  
 15  HOA              5000 non-null    object  
dtypes: float64(6), int64(4), object(6)
memory usage: 625.1+ KB
```

## modify object variables that should be numeric

```
In [17]: df['bathrooms'].value_counts() ###remainder of the number of nones for the bathroom feature:6
```

```
Out[17]: 3      1993  
4      1842  
5      654  
6      207  
2      189  
7      58  
8      19  
9      8  
3.5     7  
None    6  
1      3  
2.5     3  
35     3  
11     2  
10     1  
14     1  
18     1  
4.5     1  
15     1  
36     1  
Name: bathrooms, dtype: int64
```

```
In [18]: df['bathrooms']=np.where(df['bathrooms']=='None',0,df['bathrooms']) #change the nones to zeros to be able to graph  
df['bathrooms'].value_counts()
```

```
Out[18]: 3      1993  
4      1842  
5      654  
6      207  
2      189  
7      58  
8      19  
9      8  
3.5     7  
0      6  
1      3  
2.5     3  
35     3  
11     2  
10     1  
14     1  
18     1  
4.5     1  
15     1  
36     1  
Name: bathrooms, dtype: int64
```

```
In [19]: df.info() #bathrooms is an object so , the axis x appears in disorder
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   MLS              5000 non-null    int64  
 1   sold_price       5000 non-null    float64 
 2   zipcode          5000 non-null    int64  
 3   longitude        5000 non-null    float64 
 4   latitude         5000 non-null    float64 
 5   lot_acres        4990 non-null    float64 
 6   taxes             5000 non-null    float64 
 7   year_built       5000 non-null    int64  
 8   bedrooms          5000 non-null    int64  
 9   bathrooms         5000 non-null    object  
 10  sqrt_ft          5000 non-null    object  
 11  garage            5000 non-null    object  
 12  kitchen_features 5000 non-null    object  
 13  fireplaces        4975 non-null    float64 
 14  floor_covering   5000 non-null    object  
 15  HOA               5000 non-null    object  
dtypes: float64(6), int64(4), object(6)
memory usage: 625.1+ KB
```

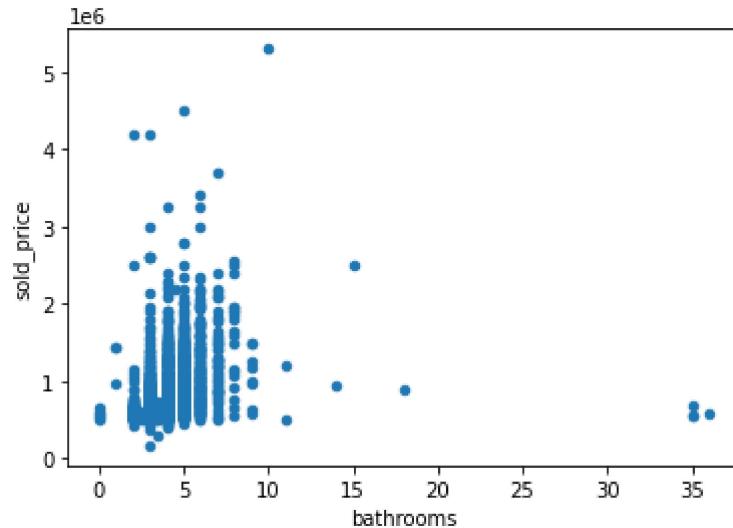
```
In [20]: df['bathrooms']=df['bathrooms'].astype(float)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   MLS              5000 non-null    int64  
 1   sold_price       5000 non-null    float64 
 2   zipcode          5000 non-null    int64  
 3   longitude        5000 non-null    float64 
 4   latitude         5000 non-null    float64 
 5   lot_acres        4990 non-null    float64 
 6   taxes            5000 non-null    float64 
 7   year_built       5000 non-null    int64  
 8   bedrooms         5000 non-null    int64  
 9   bathrooms         5000 non-null    float64 
 10  sqrt_ft          5000 non-null    object  
 11  garage           5000 non-null    object  
 12  kitchen_features 5000 non-null    object  
 13  fireplaces        4975 non-null    float64 
 14  floor_covering   5000 non-null    object  
 15  HOA              5000 non-null    object  
dtypes: float64(7), int64(4), object(5)
memory usage: 625.1+ KB
```

## scatter plot

```
In [21]: df.plot(kind='scatter',x='bathrooms',y='sold_price') ##check the relationship between the variables
```

```
Out[21]: <AxesSubplot:xlabel='bathrooms', ylabel='sold_price'>
```



```
In [22]: #So apparently there is no relationship between bathrooms and the variable 'y'  
#so it would be better if we delete that data  
#now let's do the same for the other variables  
#df['sqrt_ft'].value_counts()  
df = df.drop(df[df['bathrooms']==0].index)  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 4994 entries, 0 to 4999  
Data columns (total 16 columns):  
 #   Column           Non-Null Count  Dtype     
---  --  
 0   MLS              4994 non-null    int64    
 1   sold_price       4994 non-null    float64  
 2   zipcode          4994 non-null    int64    
 3   longitude        4994 non-null    float64  
 4   latitude         4994 non-null    float64  
 5   lot_acres        4984 non-null    float64  
 6   taxes            4994 non-null    float64  
 7   year_built       4994 non-null    int64    
 8   bedrooms         4994 non-null    int64    
 9   bathrooms         4994 non-null    float64  
 10  sqrt_ft          4994 non-null    object   
 11  garage           4994 non-null    object   
 12  kitchen_features 4994 non-null    object   
 13  fireplaces        4975 non-null    float64  
 14  floor_covering   4994 non-null    object   
 15  HOA               4994 non-null    object  
dtypes: float64(7), int64(4), object(5)  
memory usage: 663.3+ KB
```

```
In [23]: df['sqrt_ft'].value_counts() #do the same bathroom procedure for sqrt_ft
```

```
Out[23]: None      50
3541      50
3052      25
3420      18
3002      16
..
4362       1
5586       1
5117       1
3793       1
1772       1
Name: sqrt_ft, Length: 2362, dtype: int64
```

```
In [24]: df['sqrt_ft']=np.where(df['sqrt_ft']=='None',0,df['sqrt_ft'])
df['sqrt_ft'].value_counts()
```

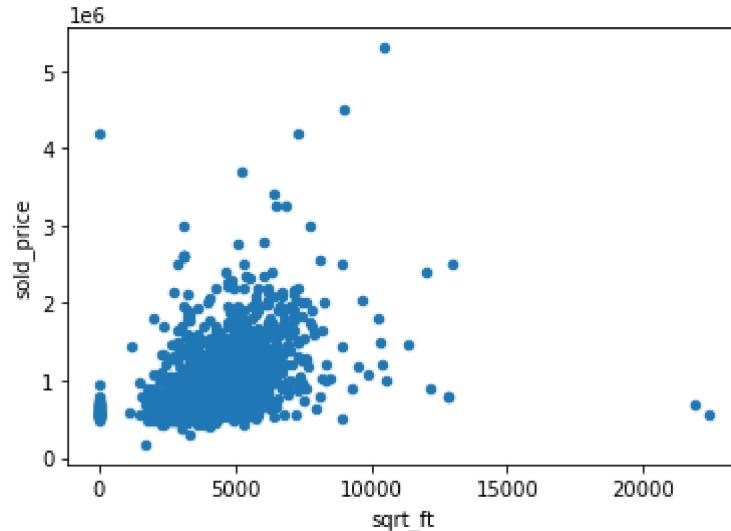
```
Out[24]: 0      50
3541      50
3052      25
3420      18
3002      16
..
4362       1
5586       1
5117       1
3793       1
1772       1
Name: sqrt_ft, Length: 2362, dtype: int64
```

```
In [25]: df['sqrt_ft']=df['sqrt_ft'].astype(float)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4994 entries, 0 to 4999
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   MLS              4994 non-null    int64  
 1   sold_price       4994 non-null    float64 
 2   zipcode          4994 non-null    int64  
 3   longitude         4994 non-null    float64 
 4   latitude          4994 non-null    float64 
 5   lot_acres        4984 non-null    float64 
 6   taxes             4994 non-null    float64 
 7   year_built       4994 non-null    int64  
 8   bedrooms          4994 non-null    int64  
 9   bathrooms         4994 non-null    float64 
 10  sqrt_ft          4994 non-null    float64 
 11  garage            4994 non-null    object  
 12  kitchen_features 4994 non-null    object  
 13  fireplaces        4975 non-null    float64 
 14  floor_covering   4994 non-null    object  
 15  HOA               4994 non-null    object  
dtypes: float64(8), int64(4), object(4)
memory usage: 663.3+ KB
```

```
In [26]: df.plot(kind='scatter',x='sqrt_ft',y='sold_price')
```

```
Out[26]: <AxesSubplot:xlabel='sqrt_ft', ylabel='sold_price'>
```



```
In [27]: #it seems that there a trend to increase, but there cannot be zero sqrt,  
#so we are going to impute the mode so as not to delete so many rows
```

```
In [28]: #df.groupby('sqrt_ft')['sold_price'].mean().sort_values(ascending = False).plot.bar()  
#plt.show()
```

```
In [29]: #df['sqrt_ft'].value_counts()
mode=df['sqrt_ft'].mode()[0]
df['sqrt_ft']=np.where(df['sqrt_ft']==0,df['sqrt_ft'].replace(mode),df['sqrt_ft']) #we replace the values of the
# we can have 0 sqrt_ft for a space
```

```
In [30]: #do the same bathroom procedure for garage
```

```
In [31]: df['garage'].value_counts()
```

```
Out[31]: 3      2792
2      1336
4      383
0      184
5      88
6      61
2.5     48
1      30
3.5     16
8      14
7      13
None     7
9      6
4.5     4
12     3
10     3
15     1
22     1
30     1
11     1
20     1
13     1
Name: garage, dtype: int64
```

In [32]: df.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4994 entries, 0 to 4999
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   MLS              4994 non-null    int64  
 1   sold_price       4994 non-null    float64 
 2   zipcode          4994 non-null    int64  
 3   longitude         4994 non-null    float64 
 4   latitude          4994 non-null    float64 
 5   lot_acres        4984 non-null    float64 
 6   taxes             4994 non-null    float64 
 7   year_built       4994 non-null    int64  
 8   bedrooms          4994 non-null    int64  
 9   bathrooms         4994 non-null    float64 
 10  sqrt_ft          4994 non-null    float64 
 11  garage            4994 non-null    object  
 12  kitchen_features 4994 non-null    object  
 13  fireplaces        4975 non-null    float64 
 14  floor_covering   4994 non-null    object  
 15  HOA               4994 non-null    object  
dtypes: float64(8), int64(4), object(4)
memory usage: 663.3+ KB
```

```
In [33]: df['garage']=np.where(df['garage']=='None',0,df['garage'])
df['garage'].value_counts()
```

```
Out[33]: 3      2792
2      1336
4      383
0      184
5      88
6      61
2.5     48
1      30
3.5     16
8      14
7      13
0       7
9       6
4.5     4
12      3
10      3
15      1
22      1
30      1
11      1
20      1
13      1
Name: garage, dtype: int64
```

```
In [34]: df['garage']=df['garage'].astype(float)
df.info()
```

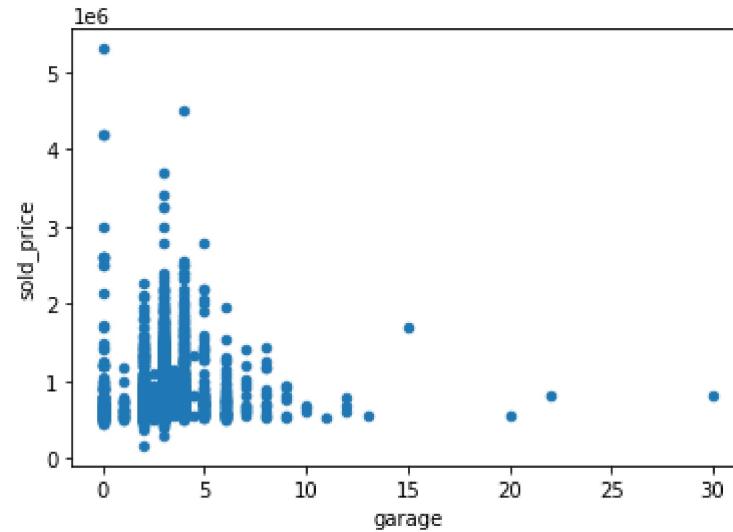
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4994 entries, 0 to 4999
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   MLS              4994 non-null    int64  
 1   sold_price       4994 non-null    float64 
 2   zipcode          4994 non-null    int64  
 3   longitude         4994 non-null    float64 
 4   latitude          4994 non-null    float64 
 5   lot_acres        4984 non-null    float64 
 6   taxes             4994 non-null    float64 
 7   year_built       4994 non-null    int64  
 8   bedrooms          4994 non-null    int64  
 9   bathrooms         4994 non-null    float64 
 10  sqrt_ft          4994 non-null    float64 
 11  garage            4994 non-null    float64 
 12  kitchen_features 4994 non-null    object  
 13  fireplaces        4975 non-null    float64 
 14  floor_covering   4994 non-null    object  
 15  HOA               4994 non-null    object  
dtypes: float64(9), int64(4), object(3)
memory usage: 663.3+ KB
```

```
In [35]: df['garage'].value_counts()
```

```
Out[35]: 3.0      2792  
2.0      1336  
4.0      383  
0.0      191  
5.0       88  
6.0       61  
2.5       48  
1.0       30  
3.5       16  
8.0       14  
7.0       13  
9.0        6  
4.5        4  
12.0       3  
10.0       3  
15.0       1  
22.0       1  
30.0       1  
11.0       1  
20.0       1  
13.0       1  
Name: garage, dtype: int64
```

```
In [36]: df.plot(kind='scatter',x='garage',y='sold_price')
```

```
Out[36]: <AxesSubplot:xlabel='garage', ylabel='sold_price'>
```



```
In [37]: #apparently there is a relationship, so then these values cannot be
#arbitrarily replaced, but at the same time there are few observations,
#so I think it is better to delete them
df = df.drop(df[df['garage']==0].index)
df.info()
```

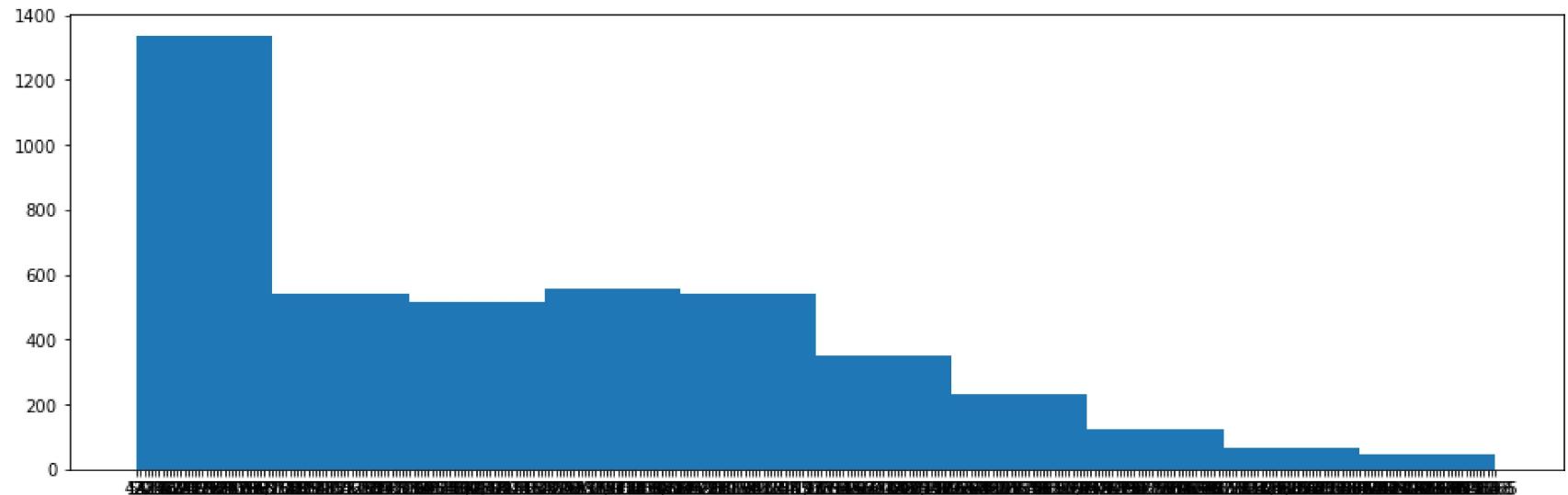
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4803 entries, 3 to 4998
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   MLS              4803 non-null    int64  
 1   sold_price       4803 non-null    float64 
 2   zipcode          4803 non-null    int64  
 3   longitude        4803 non-null    float64 
 4   latitude         4803 non-null    float64 
 5   lot_acres        4793 non-null    float64 
 6   taxes             4803 non-null    float64 
 7   year_built       4803 non-null    int64  
 8   bedrooms          4803 non-null    int64  
 9   bathrooms         4803 non-null    float64 
 10  sqrt_ft          4803 non-null    float64 
 11  garage            4803 non-null    float64 
 12  kitchen_features 4803 non-null    object  
 13  fireplaces        4784 non-null    float64 
 14  floor_covering   4803 non-null    object  
 15  HOA               4803 non-null    object  
dtypes: float64(9), int64(4), object(3)
memory usage: 637.9+ KB
```

```
In [38]: #procedure for HOA  
df['HOA'].value_counts()
```

```
Out[38]: 0           731  
None        498  
5            119  
100          107  
50            84  
...  
162            1  
166.66         1  
43.71          1  
203            1  
78.65          1  
Name: HOA, Length: 375, dtype: int64
```

```
In [39]: #Because the None data is many for the total data, a histogram can  
#be used to see the most repeated value and replace it with that data.  
plt.figure(figsize=(16,5))  
plt.hist(df[df['HOA']!='None']['HOA'])
```

```
Out[39]: (array([1336., 539., 516., 555., 543., 348., 230., 124., 68.,  
46.]),  
array([ 0. , 37.3, 74.6, 111.9, 149.2, 186.5, 223.8, 261.1, 298.4,  
335.7, 373. ]),  
<BarContainer object of 10 artists>)
```



```
In [40]: df['HOA']=np.where(df['HOA']=='None',0,df['HOA'])
```

```
In [41]: df['HOA'].value_counts()  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 4803 entries, 3 to 4998  
Data columns (total 16 columns):  
 #   Column           Non-Null Count  Dtype     
---  --    
 0   MLS              4803 non-null    int64    
 1   sold_price       4803 non-null    float64   
 2   zipcode          4803 non-null    int64    
 3   longitude        4803 non-null    float64   
 4   latitude         4803 non-null    float64   
 5   lot_acres        4793 non-null    float64   
 6   taxes             4803 non-null    float64   
 7   year_built       4803 non-null    int64    
 8   bedrooms          4803 non-null    int64    
 9   bathrooms         4803 non-null    float64   
 10  sqrt_ft          4803 non-null    float64   
 11  garage            4803 non-null    float64   
 12  kitchen_features 4803 non-null    object    
 13  fireplaces        4784 non-null    float64   
 14  floor_covering   4803 non-null    object    
 15  HOA               4803 non-null    object    
dtypes: float64(9), int64(4), object(3)  
memory usage: 637.9+ KB
```

```
In [42]: df['HOA']=df['HOA'].str.replace(',','.')  
#df['HOA']=np.where(df['HOA']=='None',0,df['HOA'])  
df['HOA']=df['HOA'].astype('float')
```

```
In [43]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4803 entries, 3 to 4998
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   MLS              4803 non-null    int64  
 1   sold_price       4803 non-null    float64 
 2   zipcode          4803 non-null    int64  
 3   longitude         4803 non-null    float64 
 4   latitude          4803 non-null    float64 
 5   lot_acres        4793 non-null    float64 
 6   taxes             4803 non-null    float64 
 7   year_built       4803 non-null    int64  
 8   bedrooms          4803 non-null    int64  
 9   bathrooms         4803 non-null    float64 
 10  sqrt_ft          4803 non-null    float64 
 11  garage            4803 non-null    float64 
 12  kitchen_features 4803 non-null    object  
 13  fireplaces        4784 non-null    float64 
 14  floor_covering   4803 non-null    object  
 15  HOA               4305 non-null    float64 
dtypes: float64(10), int64(4), object(2)
memory usage: 637.9+ KB
```

```
In [44]: df['HOA']=np.where(df['HOA'].isnull(),0,df['HOA'])
```

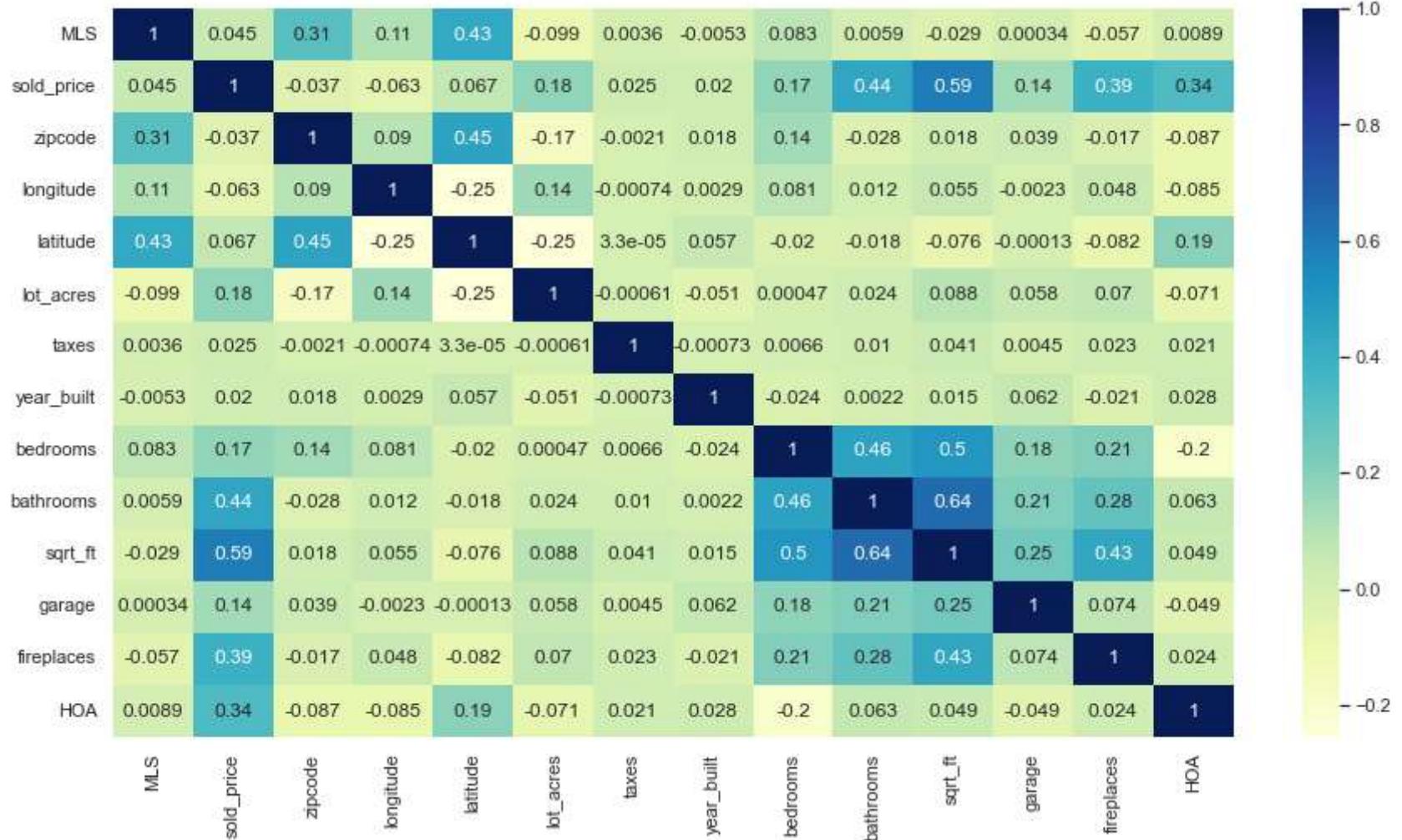
```
In [45]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4803 entries, 3 to 4998
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   MLS              4803 non-null    int64  
 1   sold_price       4803 non-null    float64 
 2   zipcode          4803 non-null    int64  
 3   longitude         4803 non-null    float64 
 4   latitude          4803 non-null    float64 
 5   lot_acres        4793 non-null    float64 
 6   taxes             4803 non-null    float64 
 7   year_built       4803 non-null    int64  
 8   bedrooms          4803 non-null    int64  
 9   bathrooms         4803 non-null    float64 
 10  sqrt_ft          4803 non-null    float64 
 11  garage            4803 non-null    float64 
 12  kitchen_features 4803 non-null    object  
 13  fireplaces        4784 non-null    float64 
 14  floor_covering   4803 non-null    object  
 15  HOA               4803 non-null    float64 
dtypes: float64(10), int64(4), object(2)
memory usage: 637.9+ KB
```

```
In [46]: #With this we finish the process for these variables
```

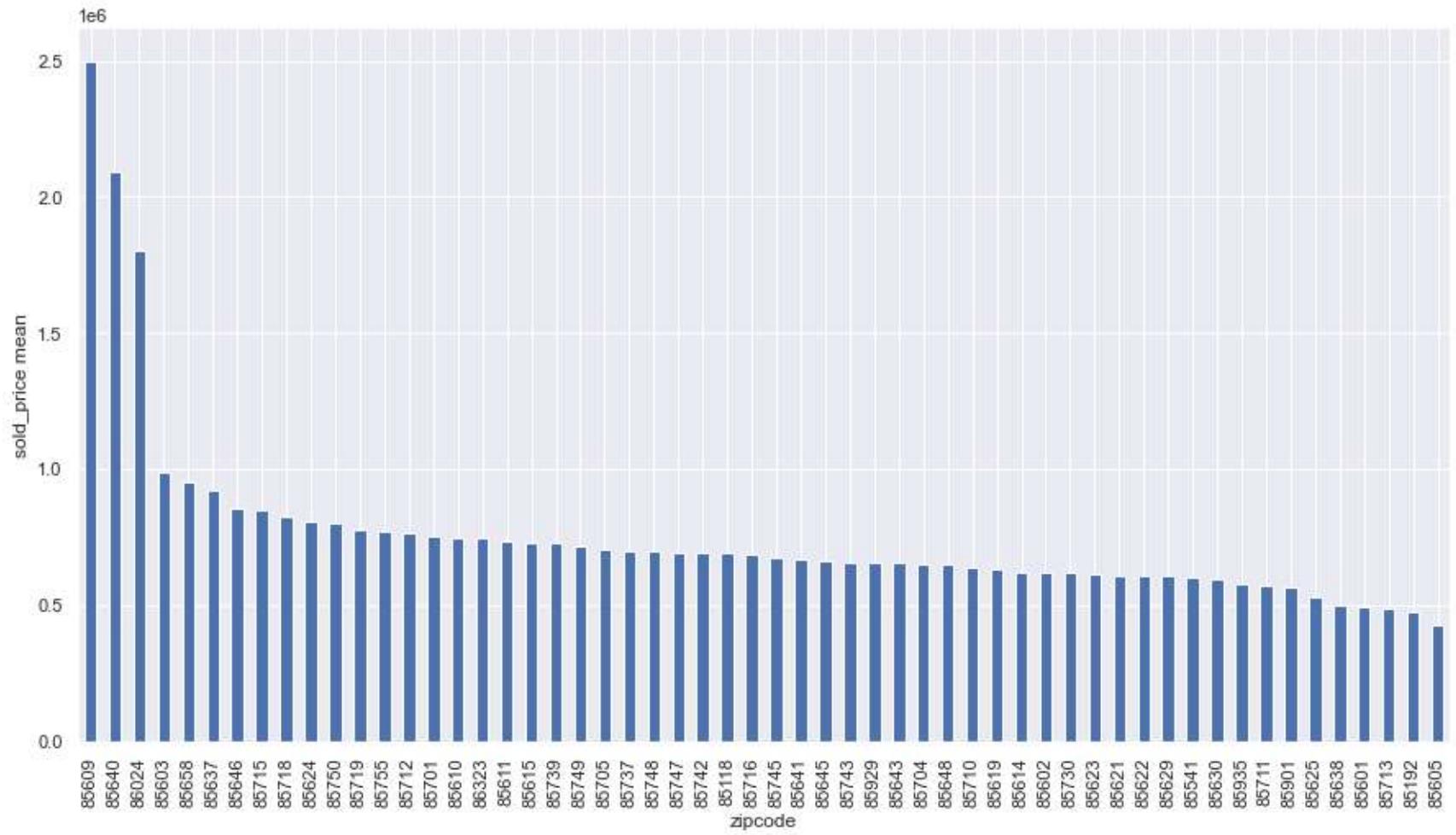
```
In [56]: sb.set(rc={"figure.figsize":(15, 8)})
dfplot1 = sb.heatmap(df.corr(), cmap="YlGnBu", annot=True)
dfplot1
```

Out[56]: <AxesSubplot:>



```
In [ ]: # we can see in the heat map that indeed the sqrt_ft are the ones with the highest correlation,  
#but with the other variables it does not have a strong correlation,  
#so we are just going to show a histogram of the zip codes.
```

```
In [76]: df.groupby('zipcode')['sold_price'].mean().sort_values(ascending = False).plot.bar()
plt.xlabel("zipcode")
plt.ylabel("sold_price mean")
plt.show()
```



## handling NaNs Values

```
In [47]: df.isnull().sum() #there are very few so I decided to eliminate them
```

```
Out[47]: MLS          0
sold_price      0
zipcode         0
longitude        0
latitude         0
lot_acres       10
taxes           0
year_built      0
bedrooms        0
bathrooms        0
sqrt_ft          0
garage           0
kitchen_features 0
fireplaces       19
floor_covering    0
HOA              0
dtype: int64
```

```
In [48]: df=df.dropna()
```

```
In [49]: df.isnull().sum()
```

```
Out[49]: MLS          0
sold_price      0
zipcode         0
longitude        0
latitude         0
lot_acres        0
taxes           0
year_built      0
bedrooms        0
bathrooms        0
sqrt_ft          0
garage           0
kitchen_features 0
fireplaces       0
floor_covering    0
HOA              0
dtype: int64
```

```
In [50]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4782 entries, 3 to 4998
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   MLS              4782 non-null    int64  
 1   sold_price       4782 non-null    float64 
 2   zipcode          4782 non-null    int64  
 3   longitude         4782 non-null    float64 
 4   latitude          4782 non-null    float64 
 5   lot_acres        4782 non-null    float64 
 6   taxes             4782 non-null    float64 
 7   year_built       4782 non-null    int64  
 8   bedrooms          4782 non-null    int64  
 9   bathrooms         4782 non-null    float64 
 10  sqrt_ft          4782 non-null    float64 
 11  garage            4782 non-null    float64 
 12  kitchen_features 4782 non-null    object  
 13  fireplaces        4782 non-null    float64 
 14  floor_covering   4782 non-null    object  
 15  HOA               4782 non-null    float64 
dtypes: float64(10), int64(4), object(2)
memory usage: 635.1+ KB
```

## handling categorical variables

```
In [51]: df.describe()
```

Out[51]:

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bath
count	4.782000e+03	4.782000e+03	4782.000000	4782.000000	4782.000000	4782.000000	4.782000e+03	4782.000000	4782.000000	4782.000000
mean	2.138991e+07	7.728091e+05	85724.498327	-110.913257	32.317169	2.737066	9.551398e+03	1993.933292	3.899624	3.8
std	1.898486e+06	2.987144e+05	36.319366	0.117324	0.166950	14.939447	1.768330e+05	59.852681	0.867482	1.1
min	3.047349e+06	1.690000e+05	85118.000000	-112.520168	31.361562	0.000000	0.000000e+00	0.000000	1.000000	1.0
25%	2.140806e+07	5.871250e+05	85718.000000	-110.980234	32.283876	0.590000	4.856648e+03	1989.000000	3.000000	3.0
50%	2.161509e+07	6.797500e+05	85737.000000	-110.922933	32.320090	0.990000	6.280395e+03	2000.000000	4.000000	4.0
75%	2.180485e+07	8.400000e+05	85750.000000	-110.859118	32.398420	1.670000	8.144557e+03	2006.000000	4.000000	4.0
max	2.192856e+07	4.500000e+06	86323.000000	-109.454637	34.927884	636.670000	1.221508e+07	2019.000000	10.000000	36.0



```
In [52]: # Join every single characteristic into one list
feature = df["kitchen_features"]
kitchen_chars = []
kitchen_vectors = []

for i in feature:
    description = i.lower().split(",")
    kitchen_chars += description

# Check the unique characteristics on the list
print(np.unique(kitchen_chars))

# Create a vector with all the unique values
values = []

for i in kitchen_chars:
    if ':' in i:
        aux = i.split(':')[0]
        values.append(aux) #Just keep the general feature
        #print(f"Saving {aux}")
    else:
        values.append(i)
        #print(f"Saving {i}")

values = np.unique(values) #These labels should be given to the programming team as additional data

for i in feature:
    vectorz = np.zeros(len(values))
    description = i.lower().split(",")

    for j in description:
        if ':' in j:
            j = j.split(':')[0]

        for k in range(len(values)):
            if values[k] == j:
                vectorz[k] = 1

    kitchen_vectors.append(vectorz)

# Create the column with the encoded vectors
df['kitchen_vectors'] = kitchen_vectors
```

```
#print(kitchen_vectors)
[' appliance color: almond' ' appliance color: black'
 ' appliance color: other' ' appliance color: stainless'
 ' appliance color: white' ' black' ' built in' ' butch block'
 ' convection' ' countertops: brazilian granite'
 ' countertops: brazillian slate' ' countertops: butcher block on isl'
 ' countertops: butcherblock/concret' ' countertops: caesar stone'
 ' countertops: caesarstone' ' countertops: carrera marble'
 ' countertops: ceasarstone' ' countertops: ceaserstone'
 ' countertops: ceramic' ' countertops: ceramic tile'
 ' countertops: chisled edge granite' ' countertops: colored concrete'
 ' countertops: combo' ' countertops: composite granite'
 ' countertops: concrete' ' countertops: concrete/wood'
 ' countertops: conrete/stone' ' countertops: corain'
 ' countertops: corian' ' countertops: corion' ' countertops: corrian'
 ' countertops: custom concrete' ' countertops: custom tile'
 ' countertops: custom tile &granite' ' countertops: dekton / granite'
 ' countertops: fascination quartzi' ' countertops: formica'
 ' countertops: ganite' ' countertops: garnite'
 ' countertops: gas / black' ' countertops: graniet'
 ' ...']
```

In [53]: values

Out[53]: array([' appliance color', ' black', ' built in', ' butch block',
 ' convection', ' countertops', ' desk', ' dishwasher',
 ' double sink', ' electric', ' electric oven', ' electric range',
 ' freezer', ' garbage disposal', ' gas cooktop', ' gas range',
 ' ge', ' indoor grill', ' island', ' lazy susan', ' microwave',
 ' missing', ' oven', ' pantry', ' prep sink', ' refrigerator',
 ' reverse osmosis', ' stainless', ' tile', ' warming drawer',
 ' water purifier', ' wet bar', ' wine cooler', '# of ovens',
 ' compactor', ' convection oven', 'desk', 'dishwasher',
 'double sink', 'electric range', 'freezer', 'garbage disposal',
 'gas range', 'island', 'microwave', 'none', 'oven', 'pantry',
 'quartzite', 'refrigerator', 'wet bar'], dtype='<U17')

In [54]: df

Out[54]:

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitc
3	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1930	7	5.0	9019.0	4.0	
4	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	6.0	6396.0	3.0	Re
5	21528016	3250000.0	85718	-110.910593	32.339090	1.67	27802.84	1999	3	4.0	6842.0	3.0	Re
6	21610478	2400000.0	85712	-110.883315	32.261069	2.10	19038.42	2001	9	8.0	12025.0	4.0	C
7	21211741	2500000.0	85750	-110.861002	32.331603	1.07	21646.00	2011	6	8.0	8921.0	4.0	Fre
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4993	21908358	565000.0	85750	-110.820216	32.307646	0.83	4568.71	1986	4	3.0	2813.0	2.0	E
4994	21909379	535000.0	85718	-110.922291	32.317496	0.18	4414.00	2002	3	2.0	2106.0	2.0	E
4995	21810382	495000.0	85641	-110.661829	31.907917	4.98	2017.00	2005	5	3.0	3601.0	3.0	D
4996	21908591	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1990	4	3.0	2318.0	3.0	E

MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitc
4998	21900515	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2009	4	4.0	3724.0	3.0

4782 rows × 17 columns



```
In [55]: # Join every single characteristic into one list
feature = df["floor_covering"]
floor_chars = []
floor_vectors = []

for i in feature:
    description = i.lower().split(",")
    floor_chars += description

# Check the unique characteristics on the list
print(np.unique(floor_chars))

# Create a vector with all the unique values
values = []

for i in floor_chars:
    if ':' in i:
        aux = i.split(':')[0]
        values.append(aux) #Just keep the general feature
        #print(f"Saving {aux}")
    else:
        values.append(i)
        #print(f"Saving {i}")

values = np.unique(values) #These labels should be given to the programming team as additional data

for i in feature:
    vectorz = np.zeros(len(values))
    description = i.lower().split(",")

    for j in description:
        if ':' in j:
            j = j.split(':')[0]

        for k in range(len(values)):
            if values[k] == j:
                vectorz[k] = 1

    floor_vectors.append(vectorz)

# Create the column with the encoded vectors
df['floor_vectors'] = floor_vectors
```

```
#print(kitchen_vectors)
```

```
['ceramic tile' 'concrete' 'granite' 'indoor/outdoor' 'laminate'  
'mexican tile' 'natural stone' 'other' 'other: 20 x 20 on diagonal'  
'other: acrylic overlay' 'other: bamboo' 'other: brazilian pergo'  
'other: brick' 'other: brick floor' 'other: brick in studio'  
'other: brick inlaid' 'other: brick pavers' 'other: canterra stone'  
'other: carpet bedrooms only' 'other: carpet- guest house'  
'other: concrete tile' 'other: cork' 'other: custom saltillo'  
'other: dyed concrete' 'other: egyptian sandstone' 'other: eng wood'  
'other: engineered wood' 'other: flagstone' 'other: gray saltillo'  
'other: hardwood' 'other: high end laminate'  
'other: itailian porclaine' 'other: italian tile' 'other: lime stone'  
'other: limestone' 'other: lux vinyl' 'other: marble'  
'other: marble-master bath' 'other: master bedroom/ tile'  
'other: mesquite wood floors' 'other: new plank tile'  
'other: new wood plank tile' 'other: organic wool carpet'  
'other: parquet' 'other: pergo' 'other: plank tile'  
'other: polished concrete' 'other: porcelain'  
'other: porcelain plank tile' 'other: porcelain tile'  
'other: porcelain tile 24x24' 'other: porcelain wood tile'  
'other: porcelain-wood' 'other: porcelain/engineered'  
'other: porclain tile' 'other: real polishd aggrgt' 'other: red brick'  
'other: refinished brick' 'other: rojo concrete overla'  
'other: saltillo' 'other: san marcos mex tile'  
'other: scored concrete' 'other: see remarks' 'other: slate'  
'other: slate tile' 'other: stained concrete' 'other: studio laminate'  
'other: talavera floors' 'other: terrazzo' 'other: throughout home'  
'other: tile bathrooms' 'other: tile/powder rm' 'other: travertine'  
'other: travertine & slate' 'other: travertine accents'  
'other: travertine entry' 'other: travertine tile'  
'other: travertine/flagstone' 'other: travertine/marble'  
'other: upg flooring' 'other: vinyl plank' 'other: vinyl planks'  
'other: wood laminate' 'other: wood laminate water' 'other: wood like'  
'other: wood plan laminate' 'vinyl' 'wood' 'carpet' 'ceramic tile'  
'concrete' 'laminate' 'mexican tile' 'natural stone' 'other'  
'other: 100% porcelain tile' 'other: brick' 'other: flagstone'  
'other: italian tile' 'other: luxury vinyl' 'other: none'  
'other: polish concrete' 'other: porcelain' 'other: porcelain tile'  
'other: porcelyn' 'other: quartzite' 'other: recycled porcelain'  
'other: saltillo tile' 'other: tbd' 'other: tile' 'other: tile-other'  
'other: travertine' 'wood']
```

```
In [56]: df.head()
```

```
Out[56]:
```

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built	bedrooms	bathrooms	sqrt_ft	garage	kitchen	
3	21919321	4500000.0	85646	-111.035925	31.645878	636.67	8418.58	1930	7	5.0	9019.0	4.0	D Dc Par	
4	21306357	3411450.0	85750	-110.813768	32.285162	3.21	15393.00	1995	4	6.0	6396.0	3.0	D Refrige	
5	21528016	3250000.0	85718	-110.910593	32.339090	1.67	27802.84	1999	3	4.0	6842.0	3.0	D Refrige	
6	21610478	2400000.0	85712	-110.883315	32.261069	2.10	19038.42	2001	9	8.0	12025.0	4.0	D Disp	
7	21211741	2500000.0	85750	-110.861002	32.331603	1.07	21646.00	2011	6	8.0	8921.0	4.0	C D Freeze	



```
In [57]: df.shape
```

```
Out[57]: (4782, 18)
```

```
In [ ]:
```