Outliar

Dealing with Outliers

In statistics, an outlier is a data point that differs significantly from other observations.An outlier may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set. An outlier can cause serious problems in statistical analyses.

Remember that even if a data point is an outlier, its still a data point! Carefully consider your data, its sources, and your goals whenver deciding to remove an outlier. Each case is different!

Lecture Goals

- Understand different mathmatical definitions of outliers
- Use Python tools to recognize outliers and remove them

Useful Links

- Wikipedia Article
- NIST Outlier Links

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

Creating random age sample

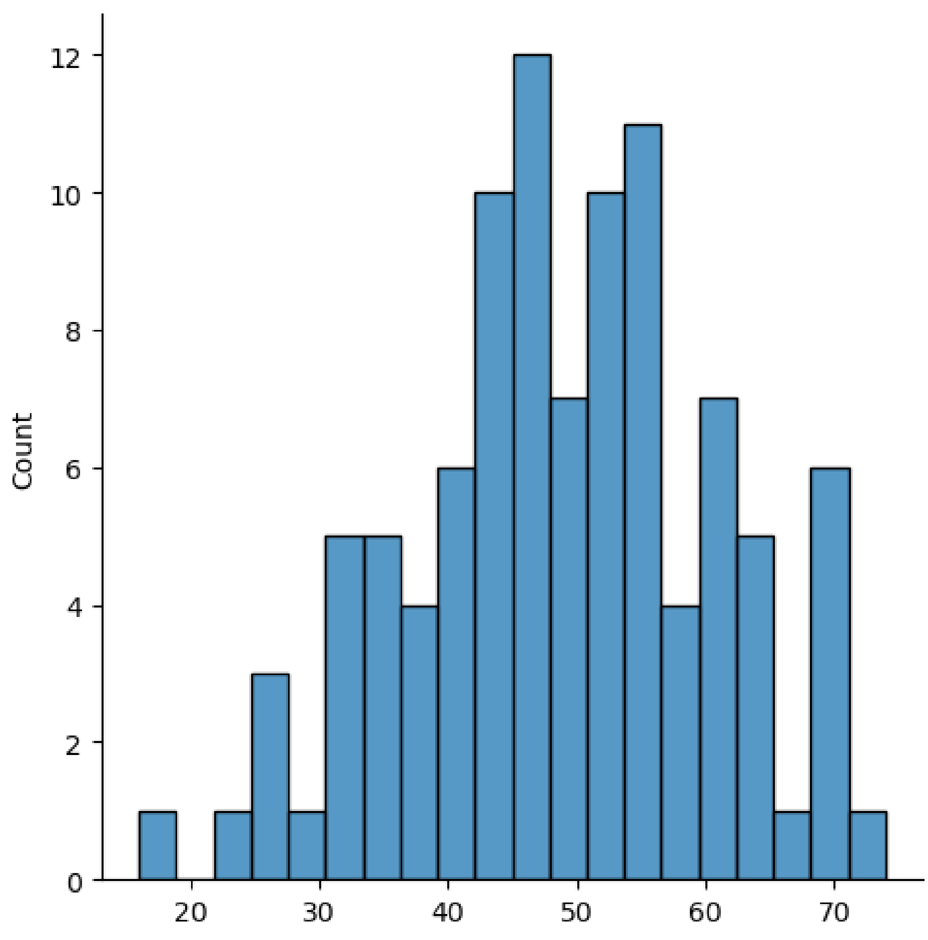
```
# Choose a mean,standard deviation, and number of samples

def create_ages(mu=50,sigma=13,num_samples=100,seed=42):

    # Set a random seed in the same cell as the random call to get the same values as us
    # We set seed to 42 (42 is an arbitrary choice from Hitchhiker's Guide to the Galaxy)
    np.random.seed(seed)

    sample_ages = np.random.normal(loc=mu,scale=sigma,size=num_samples)
    sample_ages = np.round(sample_ages,decimals=0)

    return sample_ages
```

```
sample = create_ages()
sample
```

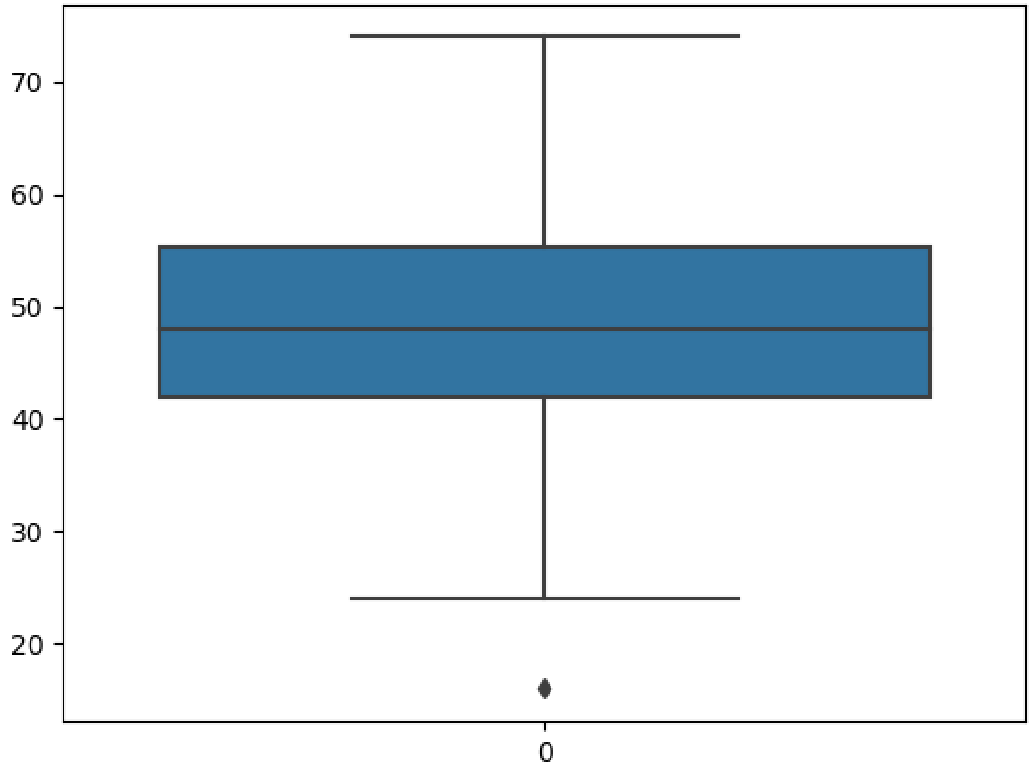
```
array([56., 48., 58., 70., 47., 47., 71., 60., 44., 57., 44., 44.,
53.,
       25., 28., 43., 37., 54., 38., 32., 69., 47., 51., 31., 43.,
51.,
       35., 55., 42., 46., 42., 74., 50., 36., 61., 34., 53., 25.,
33.,
       53., 60., 52., 48., 46., 31., 41., 44., 64., 54., 27., 54.,
45.,
       41., 58., 63., 62., 39., 46., 54., 63., 44., 48., 36., 34.,
61.,
       68., 49., 63., 55., 42., 55., 70., 50., 70., 16., 61., 51.,
46.,
       51., 24., 47., 55., 69., 43., 39., 43., 62., 54., 43., 57.,
51.,
       63., 41., 46., 45., 31., 54., 53., 50., 47.])
```

Visualize and Describe the Data

```
sns.displot(data=sample,bins=20);
plt.show()
```



```
sns.boxplot(data=sample)
plt.show();
```



Here we see the point that below 20 is outliar later we will see in quartile too

```
ser = pd.Series(sample)
```

```
ser
```

```
0     56.0
1     48.0
2     58.0
3     70.0
4     47.0
      ...
95    31.0
96    54.0
97    53.0
98    50.0
99    47.0
Length: 100, dtype: float64
```

```
ser.describe()
```

```
count    100.00000
mean      48.66000
std       11.82039
```

```
min       16.00000
25%       42.00000
50%       48.00000
75%       55.25000
max       74.00000
dtype: float64
```

```
# Inter quartile range = values at 75% - value at 25%
IQR = 55.25 - 42.0
lower_limit = 42.0 - 1.5*(IQR)
upper_limit = 55.25 + 1.5*(IQR)
```

```
lower_limit,upper_limit
```

```
(22.125, 75.125)
```

```
ser > lower_limit
```

```
0     True
1     True
2     True
3     True
4     True
      ...
95    True
96    True
97    True
98    True
99    True
Length: 100, dtype: bool
```

```
ser[ser > lower_limit]
```

```
0     56.0
1     48.0
2     58.0
3     70.0
4     47.0
      ...
95    31.0
96    54.0
97    53.0
98    50.0
99    47.0
Length: 99, dtype: float64
```

```
# Well for finding upper and lower limit we dont have to repeat that
we can use
# here we have to pass data and then percentiles
np.percentile(sample,[75,25])
```

```
array([55.25, 42.  ])
```

```
# we can store that
q75,q25 =np.percentile(sample,[75,25])
iqr = q75 - q25
```

```
q75 , q25, iqr
```

```
(55.25, 42.0, 13.25)
```

```
q25 - 1.5*iqr
```

```
22.125
```

There are many ways to identify and remove outliers:

- Trimming based off a provided value
- Capping based off IQR or STD
- https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba
- https://towardsdatascience.com/5-ways-to-detect-outliers-that-every-data-scientist-should-know-python-code-70a54335a623

Finding Outlier

About Dataset

Ames Data Set

Let's explore any extreme outliers in our Ames Housing Data Set

```
# Here with our data file we have a text file with full description about the columns and data in form of test file


# with open ("D:\\Study\\Programming\\python\\Python course from udemy\\Udemy - 2022 Python for Machine Learning & Data Science Masterclass\\01 - Introduction to Course\\1UNZIP-FOR-NOTEBOOKS-FINAL\\DATA\\Ames_Housing_Feature_Description.txt") as f:
#    print(f.read())
```

```
df = pd.read_csv("D:\\Study\\Programming\\python\\Python course from udemy\\Udemy - 2022 Python for Machine Learning & Data Science Masterclass\\01 - Introduction to Course\\1UNZIP-FOR-NOTEBOOKS-FINAL\\DATA\\Ames_Housing_Data.csv")
df.head()
```

```
         PID  MS SubClass MS Zoning  Lot Frontage  Lot Area Street
Alley  \
0  526301100           20        RL         141.0     31770   Pave
NaN
```

```
1  526350040           20        RH          80.0     11622   Pave
NaN
2  526351010           20        RL          81.0     14267   Pave
NaN
3  526353030           20        RL          93.0     11160   Pave
NaN
4  527105010           60        RL          74.0     13830   Pave
NaN

  Lot Shape Land Contour Utilities  ... Pool Area Pool QC  Fence Misc
Feature  \
0       IR1          Lvl    AllPub  ...         0     NaN    NaN
NaN
1       Reg          Lvl    AllPub  ...         0     NaN  MnPrv
NaN
2       IR1          Lvl    AllPub  ...         0     NaN    NaN
Gar2
3       Reg          Lvl    AllPub  ...         0     NaN    NaN
NaN
4       IR1          Lvl    AllPub  ...         0     NaN  MnPrv
NaN

  Misc Val Mo Sold Yr Sold  Sale Type  Sale Condition  SalePrice
0        0       5    2010        WD           Normal     215000
1        0       6    2010        WD           Normal     105000
2    12500       6    2010        WD           Normal     172000
3        0       4    2010        WD           Normal     244000
4        0       3    2010        WD           Normal     189900

[5 rows x 81 columns]
```

```
sns.heatmap(df.corr())
```

```
<AxesSubplot:>
```

```
df.corr()
```

```
                      PID  MS SubClass  Lot Frontage  Lot Area
Overall Qual  \
PID              1.000000    -0.001281     -0.096918  0.034868     -
0.263147
MS SubClass     -0.001281     1.000000     -0.420135 -0.204613
0.039419
Lot Frontage    -0.096918    -0.420135      1.000000  0.491313
0.212042
Lot Area         0.034868    -0.204613      0.491313  1.000000
0.097188
Overall Qual    -0.263147     0.039419      0.212042  0.097188
1.000000
Overall Cond     0.104451    -0.067349     -0.074448 -0.034759     -
0.094812
```

Year Built	-0.343388	0.036579	0.121562	0.023258	0.597027
Year Remod/Add	-0.157111	0.043397	0.091712	0.021682	0.569609
Mas Vnr Area	-0.229283	0.002730	0.222407	0.126830	0.429418
BsmtFin SF 1	-0.098375	-0.060075	0.215583	0.191555	0.284118
BsmtFin SF 2	-0.001145	-0.070946	0.045999	0.083150	-0.041287
Bsmt Unf SF	-0.087707	-0.130421	0.116743	0.023658	0.270058
Total Bsmt SF	-0.189642	-0.219445	0.353773	0.253589	0.547294
1st Flr SF	-0.141902	-0.247828	0.457391	0.332235	0.477837
2nd Flr SF	-0.003289	0.304237	0.029187	0.032996	0.241402
Low Qual Fin SF	0.056940	0.025765	0.005249	0.000812	-0.048680
Gr Liv Area	-0.107579	0.068061	0.383822	0.285599	0.570556
Bsmt Full Bath	-0.037759	0.013701	0.108915	0.125877	0.167858
Bsmt Half Bath	0.004328	-0.003329	-0.024724	0.026903	-0.041647
Full Bath	-0.171431	0.134631	0.184521	0.127433	0.522263
Half Bath	-0.166636	0.175879	0.041880	0.035497	0.268853
Bedroom AbvGr	0.006345	-0.019208	0.240442	0.136569	0.063291
Kitchen AbvGr	0.076470	0.257698	0.005407	-0.020301	-0.159744
TotRms AbvGrd	-0.068981	0.031898	0.353137	0.216597	0.380693
Fireplaces	-0.108056	-0.049955	0.257255	0.256989	0.393007
Garage Yr Blt	-0.256829	0.088754	0.076306	-0.008952	0.570569
Garage Cars	-0.237484	-0.045883	0.308706	0.179512	0.599545
Garage Area	-0.210606	-0.103239	0.358505	0.212822	0.563503
Wood Deck SF	-0.051135	-0.017310	0.120084	0.157212	0.255663
Open Porch SF	-0.071311	-0.014823	0.163040	0.103760	0.298412
Enclosed Porch	0.162519	-0.022866	0.012758	0.021868	-0.140332

```
3Ssn Porch      -0.024894    -0.037956      0.028564  0.016243
0.018240
Screen Porch    -0.025735    -0.050614      0.076666  0.055044
0.041615
Pool Area       -0.002845    -0.003434      0.173947  0.093775
0.030399
Misc Val        -0.008260    -0.029254      0.044476  0.069188
0.005179
Mo Sold         -0.050455     0.000350      0.011085  0.003859
0.031103
Yr Sold          0.009579    -0.017905     -0.007547 -0.023085     -
0.020719
SalePrice       -0.246521    -0.085092      0.357318  0.266549
0.799262

                 Overall Cond  Year Built  Year Remod/Add  Mas Vnr
Area  \
PID                  0.104451   -0.343388       -0.157111     -
0.229283
MS SubClass         -0.067349    0.036579        0.043397
0.002730
Lot Frontage        -0.074448    0.121562        0.091712
0.222407
Lot Area            -0.034759    0.023258        0.021682
0.126830
Overall Qual        -0.094812    0.597027        0.569609
0.429418
Overall Cond         1.000000   -0.368773        0.047680     -
0.135340
Year Built          -0.368773    1.000000        0.612095
0.313292
Year Remod/Add       0.047680    0.612095        1.000000
0.196928
Mas Vnr Area        -0.135340    0.313292        0.196928
1.000000
BsmtFin SF 1        -0.050935    0.279870        0.151790
0.301872
BsmtFin SF 2         0.041134   -0.027415       -0.062129     -
0.016019
Bsmt Unf SF         -0.136819    0.128998        0.164805
0.091668
Total Bsmt SF       -0.173344    0.407526        0.297481
0.397040
1st Flr SF          -0.157052    0.310463        0.242108
0.395736
2nd Flr SF           0.006218    0.016828        0.158939
0.121805
Low Qual Fin SF      0.009175   -0.144282       -0.060365     -
0.057701
Gr Liv Area         -0.115643    0.241726        0.316855
```

```
0.403611
Bsmt Full Bath      -0.042766    0.211849        0.134387
0.140113
Bsmt Half Bath       0.084455   -0.030626       -0.046292
0.015421
Full Bath           -0.214316    0.469406        0.457266
0.260153
Half Bath           -0.088127    0.269268        0.211771
0.192965
Bedroom AbvGr       -0.006137   -0.055093       -0.021536
0.080546
Kitchen AbvGr       -0.086386   -0.137852       -0.142404     -
0.050998
TotRms AbvGrd       -0.089816    0.111919        0.197528
0.279563
Fireplaces          -0.031702    0.170672        0.133322
0.272068
Garage Yr Blt       -0.326017    0.834849        0.652310
0.254784
Garage Cars         -0.181557    0.537443        0.425403
0.360159
Garage Area         -0.153754    0.480131        0.376438
0.373458
Wood Deck SF         0.020344    0.228964        0.217857
0.165467
Open Porch SF       -0.068934    0.198365        0.241748
0.143748
Enclosed Porch       0.071459   -0.374364       -0.220383     -
0.110787
3Ssn Porch           0.043852    0.015803        0.037412
0.013778
Screen Porch         0.044055   -0.041436       -0.046888
0.065643
Pool Area           -0.016787    0.002213       -0.011410
0.004617
Misc Val             0.034056   -0.011011       -0.003132
0.044934
Mo Sold             -0.007295    0.014577        0.018048     -
0.000276
Yr Sold              0.031207   -0.013197        0.032652     -
0.017715
SalePrice           -0.101697    0.558426        0.532974
0.508285

                 BsmtFin SF 1  ...  Wood Deck SF  Open Porch SF  \
PID                 -0.098375  ...     -0.051135      -0.071311
MS SubClass         -0.060075  ...     -0.017310      -0.014823
Lot Frontage         0.215583  ...      0.120084       0.163040
Lot Area             0.191555  ...      0.157212       0.103760
Overall Qual         0.284118  ...      0.255663       0.298412
```

```
Overall Cond       -0.050935  ...      0.020344      -0.068934
Year Built          0.279870  ...      0.228964       0.198365
Year Remod/Add      0.151790  ...      0.217857       0.241748
Mas Vnr Area        0.301872  ...      0.165467       0.143748
BsmtFin SF 1        1.000000  ...      0.224010       0.124947
BsmtFin SF 2       -0.054129  ...      0.098528      -0.005587
Bsmt Unf SF        -0.477875  ...     -0.039621       0.118880
Total Bsmt SF       0.536547  ...      0.229931       0.245627
1st Flr SF          0.457472  ...      0.227131       0.238041
2nd Flr SF         -0.164014  ...      0.089097       0.184538
Low Qual Fin SF    -0.066173  ...     -0.015646      -0.000761
Gr Liv Area         0.209633  ...      0.250153       0.340857
Bsmt Full Bath      0.640020  ...      0.186945       0.082268
Bsmt Half Bath      0.077548  ...      0.051430      -0.035069
Full Bath           0.077772  ...      0.179574       0.258675
Half Bath          -0.008457  ...      0.115212       0.180704
Bedroom AbvGr      -0.118959  ...      0.029711       0.083650
Kitchen AbvGr      -0.086738  ...     -0.087410      -0.068283
TotRms AbvGrd       0.047631  ...      0.154735       0.235684
Fireplaces          0.295882  ...      0.228064       0.159637
Garage Yr Blt       0.194238  ...      0.221991       0.231240
Garage Cars         0.255483  ...      0.241226       0.204182
Garage Area         0.309876  ...      0.238371       0.232912
Wood Deck SF        0.224010  ...      1.000000       0.039243
Open Porch SF       0.124947  ...      0.039243       1.000000
Enclosed Porch     -0.100455  ...     -0.119136      -0.059875
3Ssn Porch          0.050541  ...     -0.003967      -0.009458
Screen Porch        0.095874  ...     -0.052191       0.047548
Pool Area           0.084140  ...      0.094156       0.064135
Misc Val            0.092886  ...      0.056820       0.077254
Mo Sold            -0.001155  ...      0.016974       0.033651
Yr Sold             0.022397  ...      0.000882      -0.037467
SalePrice           0.432914  ...      0.327143       0.312951

                 Enclosed Porch  3Ssn Porch  Screen Porch  Pool
Area  \
PID                    0.162519   -0.024894     -0.025735  -0.002845

MS SubClass           -0.022866   -0.037956     -0.050614  -0.003434

Lot Frontage           0.012758    0.028564      0.076666   0.173947

Lot Area               0.021868    0.016243      0.055044   0.093775

Overall Qual          -0.140332    0.018240      0.041615   0.030399

Overall Cond           0.071459    0.043852      0.044055  -0.016787

Year Built            -0.374364    0.015803     -0.041436   0.002213
```

Year Remod/Add	-0.220383	0.037412	-0.046888	-0.011410
Mas Vnr Area	-0.110787	0.013778	0.065643	0.004617
BsmtFin SF 1	-0.100455	0.050541	0.095874	0.084140
BsmtFin SF 2	0.032380	-0.023325	0.062951	0.044398
Bsmt Unf SF	0.006229	-0.005446	-0.048083	-0.031999
Total Bsmt SF	-0.085225	0.037871	0.075341	0.072128
1st Flr SF	-0.065713	0.044061	0.098316	0.121821
2nd Flr SF	0.055429	-0.032172	0.011741	0.044602
Low Qual Fin SF	0.087326	-0.004505	0.006943	0.035200
Gr Liv Area	0.004030	0.006481	0.086804	0.135463
Bsmt Full Bath	-0.069235	0.027034	0.052208	0.043705
Bsmt Half Bath	-0.009334	0.026954	0.042326	0.066902
Full Bath	-0.117795	0.015435	-0.015130	0.028205
Half Bath	-0.081312	-0.023231	0.035990	0.001515
Bedroom AbvGr	0.052115	-0.047151	0.009250	0.036707
Kitchen AbvGr	0.027911	-0.021379	-0.056337	-0.013066
TotRms AbvGrd	0.017221	-0.025097	0.033731	0.072103
Fireplaces	-0.000250	0.018414	0.168004	0.098449
Garage Yr Blt	-0.300879	0.020617	-0.062515	-0.014513
Garage Cars	-0.132840	0.023345	0.043012	0.030393
Garage Area	-0.106272	0.029458	0.062436	0.053051
Wood Deck SF	-0.119136	-0.003967	-0.052191	0.094156
Open Porch SF	-0.059875	-0.009458	0.047548	0.064135
Enclosed Porch	1.000000	-0.032674	-0.063965	0.092596

3Ssn Porch	-0.032674	1.000000	-0.029430	-0.006501
Screen Porch	-0.063965	-0.029430	1.000000	0.026383
Pool Area	0.092596	-0.006501	0.026383	1.000000
Misc Val	0.008773	-0.000753	0.007162	0.011942
Mo Sold	-0.021324	0.027229	0.028169	-0.042223
Yr Sold	-0.000505	0.022668	-0.006116	-0.052541
SalePrice	-0.128787	0.032225	0.112151	0.068403

	Misc Val	Mo Sold	Yr Sold	SalePrice
PID	-0.008260	-0.050455	0.009579	-0.246521
MS SubClass	-0.029254	0.000350	-0.017905	-0.085092
Lot Frontage	0.044476	0.011085	-0.007547	0.357318
Lot Area	0.069188	0.003859	-0.023085	0.266549
Overall Qual	0.005179	0.031103	-0.020719	0.799262
Overall Cond	0.034056	-0.007295	0.031207	-0.101697
Year Built	-0.011011	0.014577	-0.013197	0.558426
Year Remod/Add	-0.003132	0.018048	0.032652	0.532974
Mas Vnr Area	0.044934	-0.000276	-0.017715	0.508285
BsmtFin SF 1	0.092886	-0.001155	0.022397	0.432914
BsmtFin SF 2	-0.005204	-0.009484	0.007105	0.005891
Bsmt Unf SF	-0.010166	0.021569	-0.036384	0.182855
Total Bsmt SF	0.083904	0.016678	-0.010405	0.632280
1st Flr SF	0.093003	0.040496	-0.013667	0.621676
2nd Flr SF	-0.005078	0.013247	-0.018530	0.269373
Low Qual Fin SF	-0.005939	0.011397	-0.002074	-0.037660
Gr Liv Area	0.067252	0.043665	-0.026489	0.706780
Bsmt Full Bath	-0.004868	-0.003471	0.044905	0.276050
Bsmt Half Bath	0.036982	0.022699	-0.019529	-0.035835
Full Bath	-0.009771	0.046032	-0.004754	0.545604
Half Bath	0.026648	-0.001311	0.001561	0.285056
Bedroom AbvGr	0.000887	0.053677	-0.018008	0.143913
Kitchen AbvGr	0.025145	0.035201	0.035421	-0.119814
TotRms AbvGrd	0.061134	0.043784	-0.030498	0.495474
Fireplaces	0.008192	0.032152	-0.007612	0.474558
Garage Yr Blt	-0.009265	0.024498	-0.005159	0.526965
Garage Cars	-0.016948	0.049847	-0.022488	0.647877
Garage Area	0.008466	0.039544	-0.013018	0.640401
Wood Deck SF	0.056820	0.016974	0.000882	0.327143
Open Porch SF	0.077254	0.033651	-0.037467	0.312951
Enclosed Porch	0.008773	-0.021324	-0.000505	-0.128787
3Ssn Porch	-0.000753	0.027229	0.022668	0.032225

```
Screen Porch     0.007162  0.028169 -0.006116   0.112151
Pool Area        0.011942 -0.042223 -0.052541   0.068403
Misc Val         1.000000  0.007333  0.008574  -0.015691
Mo Sold          0.007333  1.000000 -0.155554   0.035259
Yr Sold          0.008574 -0.155554  1.000000  -0.030569
SalePrice       -0.015691  0.035259 -0.030569   1.000000

[38 rows x 38 columns]
```

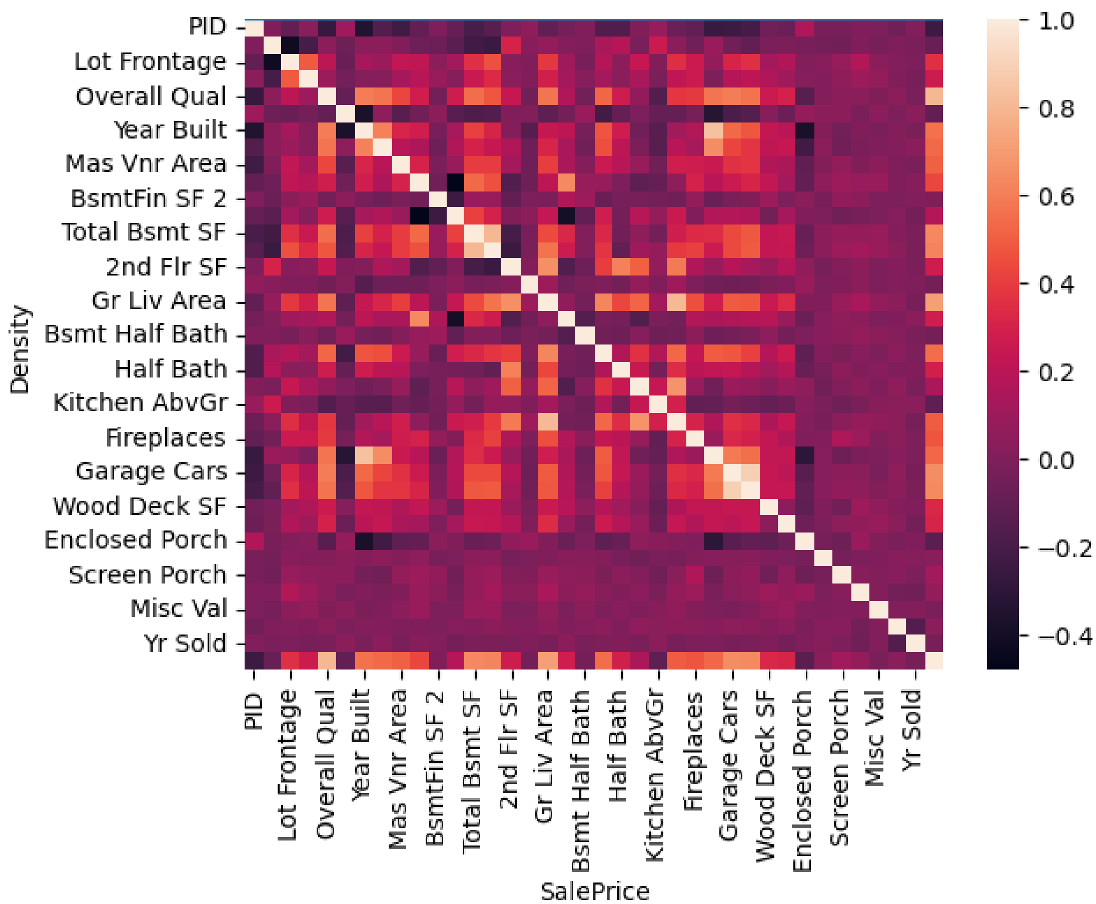
```
df.corr()["SalePrice"].sort_values()
```

```
PID               -0.246521
Enclosed Porch    -0.128787
Kitchen AbvGr     -0.119814
Overall Cond      -0.101697
MS SubClass       -0.085092
Low Qual Fin SF   -0.037660
Bsmt Half Bath    -0.035835
Yr Sold           -0.030569
Misc Val          -0.015691
BsmtFin SF 2       0.005891
3Ssn Porch         0.032225
Mo Sold            0.035259
Pool Area          0.068403
Screen Porch       0.112151
Bedroom AbvGr      0.143913
Bsmt Unf SF        0.182855
Lot Area           0.266549
2nd Flr SF         0.269373
Bsmt Full Bath     0.276050
Half Bath          0.285056
Open Porch SF      0.312951
Wood Deck SF       0.327143
Lot Frontage       0.357318
BsmtFin SF 1       0.432914
Fireplaces         0.474558
TotRms AbvGrd      0.495474
Mas Vnr Area       0.508285
Garage Yr Blt      0.526965
Year Remod/Add     0.532974
Full Bath          0.545604
Year Built         0.558426
1st Flr SF         0.621676
Total Bsmt SF      0.632280
Garage Area        0.640401
Garage Cars        0.647877
Gr Liv Area        0.706780
Overall Qual       0.799262
SalePrice          1.000000
Name: SalePrice, dtype: float64
```

```
sns.distplot(df["SalePrice"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\
distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

<AxesSubplot:xlabel='SalePrice', ylabel='Density'>
```

```
sns.scatterplot(x='Overall Qual',y='SalePrice',data = df)
plt.show();
```



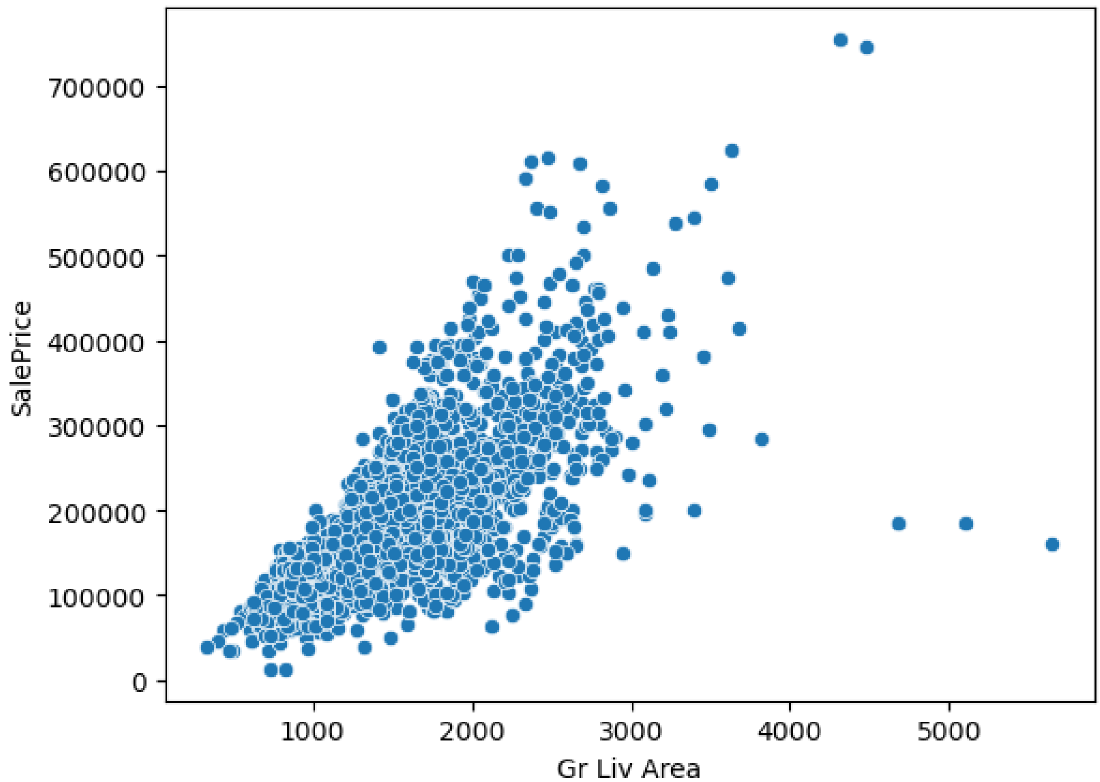
Here we can see that there are few houses which have overall quality 10 but sales as 5 overall quality price

```
df[(df['Overall Qual'] >8) & (df['SalePrice'] < 200000)]
```

```
            PID  MS SubClass MS Zoning  Lot Frontage  Lot Area Street
Alley  \
1182  533350090           60        RL           NaN     24572   Pave
NaN
```

```
1498  908154235           60        RL         313.0     63887   Pave
NaN
2180  908154195           20        RL         128.0     39290   Pave
NaN
2181  908154205           60        RL         130.0     40094   Pave
NaN

     Lot Shape Land Contour Utilities  ... Pool Area Pool QC Fence  \
1182       IR1          Lvl    AllPub  ...         0     NaN   NaN
1498       IR3          Bnk    AllPub  ...       480      Gd   NaN
2180       IR1          Bnk    AllPub  ...         0     NaN   NaN
2181       IR1          Bnk    AllPub  ...         0     NaN   NaN

     Misc Feature Misc Val Mo Sold Yr Sold  Sale Type  Sale Condition
\
1182          NaN        0       6    2008         WD          Family

1498          NaN        0       1    2008        New         Partial

2180         Elev    17000      10    2007        New         Partial

2181          NaN        0      10    2007        New         Partial


      SalePrice
1182     150000
1498     160000
2180     183850
2181     184750

[4 rows x 81 columns]
```

```
sns.scatterplot(x='Gr Liv Area',y='SalePrice',data = df)
plt.show();
```



Here we can see that have Greater living area and sales at low price it seems like outliear as have to find them

```
df[(df['Gr Liv Area'] > 4000) & (df['SalePrice'] < 300000)]
```

```
            PID  MS SubClass MS Zoning  Lot Frontage  Lot Area Street
Alley  \
1498  908154235           60        RL         313.0     63887   Pave
NaN
2180  908154195           20        RL         128.0     39290   Pave
NaN
2181  908154205           60        RL         130.0     40094   Pave
NaN

     Lot Shape Land Contour Utilities  ... Pool Area Pool QC Fence  \
1498       IR3          Bnk    AllPub  ...       480      Gd   NaN
2180       IR1          Bnk    AllPub  ...         0     NaN   NaN
2181       IR1          Bnk    AllPub  ...         0     NaN   NaN

     Misc Feature Misc Val Mo Sold Yr Sold  Sale Type  Sale Condition
\
1498          NaN        0       1    2008        New         Partial

2180         Elev    17000      10    2007        New         Partial

2181          NaN        0      10    2007        New         Partial
```

```
      SalePrice
1498     160000
2180     183850
2181     184750

[3 rows x 81 columns]
```

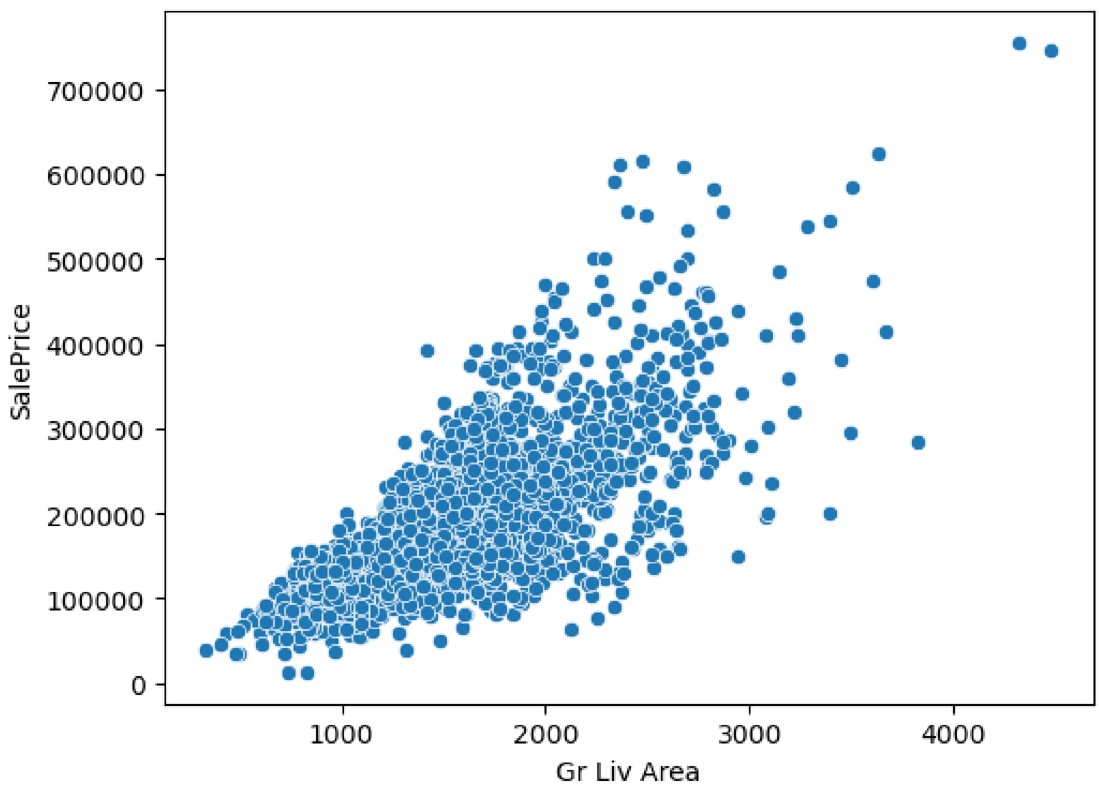
Deleting Outlier

```
# we store those in some data by put .index at last
drop_index = df[(df['Gr Liv Area'] > 4000) & (df['SalePrice'] <
300000)].index
drop_index
```

```
Int64Index([1498, 2180, 2181], dtype='int64')
```

```
df = df.drop(drop_index,axis=0)
```

```
sns.scatterplot(x='Gr Liv Area',y='SalePrice',data = df)
plt.show();
```



Save this csv for backup then we start dealing with missing values

```
df.to_csv('D:\\Study\\Ames_Housing_Data_No_outlier.csv',index=False)
```

Dealing with Missing Data

We already reviewed Pandas operations for missing data, now let's apply this to clean a real data file. Keep in mind, there is no 100% correct way of doing this, and this notebook just serves as an example of some reasonable approaches to take on this data.

Note: Throughout this section we will be slowly cleaning and adding features to the Ames Housing Dataset for use in the next section. Make sure to always be loading the same file name as in the notebook.

2nd Note: Some of the methods shown here may not lead to optimal performance, but instead are shown to display examples of various methods available.

```
# Here with our data file we have a text file with full description
about the columns and data in form of test file


# with open ("D:\\Study\\Programming\\python\\Python course from
udemy\\Udemy - 2022 Python for Machine Learning & Data Science
Masterclass\\01 - Introduction to Course\\1UNZIP-FOR-NOTEBOOKS-FINAL\\
DATA\\Ames_Housing_Feature_Description.txt") as f:
#    print(f.read())

df = pd.read_csv('D:\\Study\\Ames_Housing_Data_No_outlier.csv')
df.head()
```

```
         PID  MS SubClass MS Zoning  Lot Frontage  Lot Area Street
Alley  \
0  526301100           20        RL         141.0     31770   Pave
NaN
1  526350040           20        RH          80.0     11622   Pave
NaN
2  526351010           20        RL          81.0     14267   Pave
NaN
3  526353030           20        RL          93.0     11160   Pave
NaN
4  527105010           60        RL          74.0     13830   Pave
NaN

  Lot Shape Land Contour Utilities  ... Pool Area Pool QC  Fence Misc
Feature  \
0       IR1          Lvl    AllPub  ...         0     NaN    NaN
NaN
1       Reg          Lvl    AllPub  ...         0     NaN  MnPrv
NaN
2       IR1          Lvl    AllPub  ...         0     NaN    NaN
Gar2
3       Reg          Lvl    AllPub  ...         0     NaN    NaN
NaN
```

```
4       IR1          Lvl     AllPub  ...          0     NaN  MnPrv
NaN

  Misc Val Mo Sold Yr Sold  Sale Type  Sale Condition  SalePrice
0        0       5    2010        WD           Normal     215000
1        0       6    2010        WD           Normal     105000
2    12500       6    2010        WD           Normal     172000
3        0       4    2010        WD           Normal     244000
4        0       3    2010        WD           Normal     189900

[5 rows x 81 columns]
```

Removing the PID

We already have an index, so we don't need the PID unique identifier for the regression we will perform later on.

```
#Here PID is unique id of property we already have index number so we
can remove that
df=df.drop('PID',axis=1)
len(df.columns)
```

```
80
```

Observing NaN Features

```
df.isnull()
```

```
      MS SubClass  MS Zoning  Lot Frontage  Lot Area  Street  Alley  \
0           False      False         False     False   False   True
1           False      False         False     False   False   True
2           False      False         False     False   False   True
3           False      False         False     False   False   True
4           False      False         False     False   False   True
...           ...        ...           ...       ...     ...    ...
2922        False      False         False     False   False   True
2923        False      False          True     False   False   True
2924        False      False         False     False   False   True
2925        False      False         False     False   False   True
2926        False      False         False     False   False   True

      Lot Shape  Land Contour  Utilities  Lot Config  ...  Pool Area
Pool QC  \
0         False         False      False       False  ...      False
True
1         False         False      False       False  ...      False
True
2         False         False      False       False  ...      False
True
3         False         False      False       False  ...      False
True
4         False         False      False       False  ...      False
```

```
True
...          ...           ...        ...          ...  ...          ...
...
2922      False         False      False        False  ...      False
True
2923      False         False      False        False  ...      False
True
2924      False         False      False        False  ...      False
True
2925      False         False      False        False  ...      False
True
2926      False         False      False        False  ...      False
True

      Fence  Misc Feature  Misc Val  Mo Sold  Yr Sold  Sale Type  \
0      True          True     False    False    False      False
1     False          True     False    False    False      False
2      True         False     False    False    False      False
3      True          True     False    False    False      False
4     False          True     False    False    False      False
...     ...           ...       ...      ...      ...        ...
2922  False          True     False    False    False      False
2923  False          True     False    False    False      False
2924  False         False     False    False    False      False
2925   True          True     False    False    False      False
2926   True          True     False    False    False      False

      Sale Condition  SalePrice
0              False      False
1              False      False
2              False      False
3              False      False
4              False      False
...              ...        ...
2922           False      False
2923           False      False
2924           False      False
2925           False      False
2926           False      False

[2927 rows x 80 columns]
```

```python
# to see how many missing rows in each column
df.isnull().sum()
```

```
MS SubClass         0
MS Zoning           0
Lot Frontage      490
Lot Area            0
Street              0
```

```
                 ...
Mo Sold             0
Yr Sold             0
Sale Type           0
Sale Condition      0
SalePrice           0
Length: 80, dtype: int64
```

```
df.isnull().sum()>0
```

```
MS SubClass       False
MS Zoning         False
Lot Frontage       True
Lot Area          False
Street            False
                  ...
Mo Sold           False
Yr Sold           False
Sale Type         False
Sale Condition    False
SalePrice         False
Length: 80, dtype: bool
```

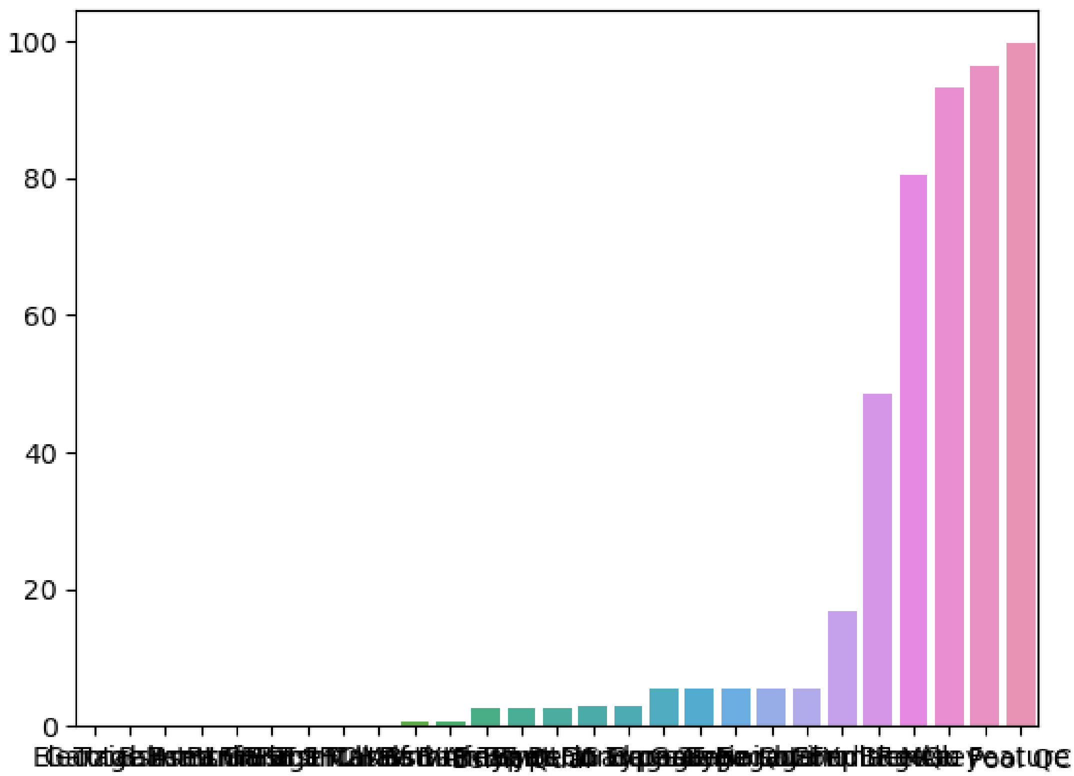
```
# we need to find out how many rows are missing so we will use
percentage
100*(df.isnull().sum())/len(df)
```

```
MS SubClass        0.00000
MS Zoning          0.00000
Lot Frontage      16.74069
Lot Area           0.00000
Street             0.00000
                     ...
Mo Sold            0.00000
Yr Sold            0.00000
Sale Type          0.00000
Sale Condition     0.00000
SalePrice          0.00000
Length: 80, dtype: float64
```

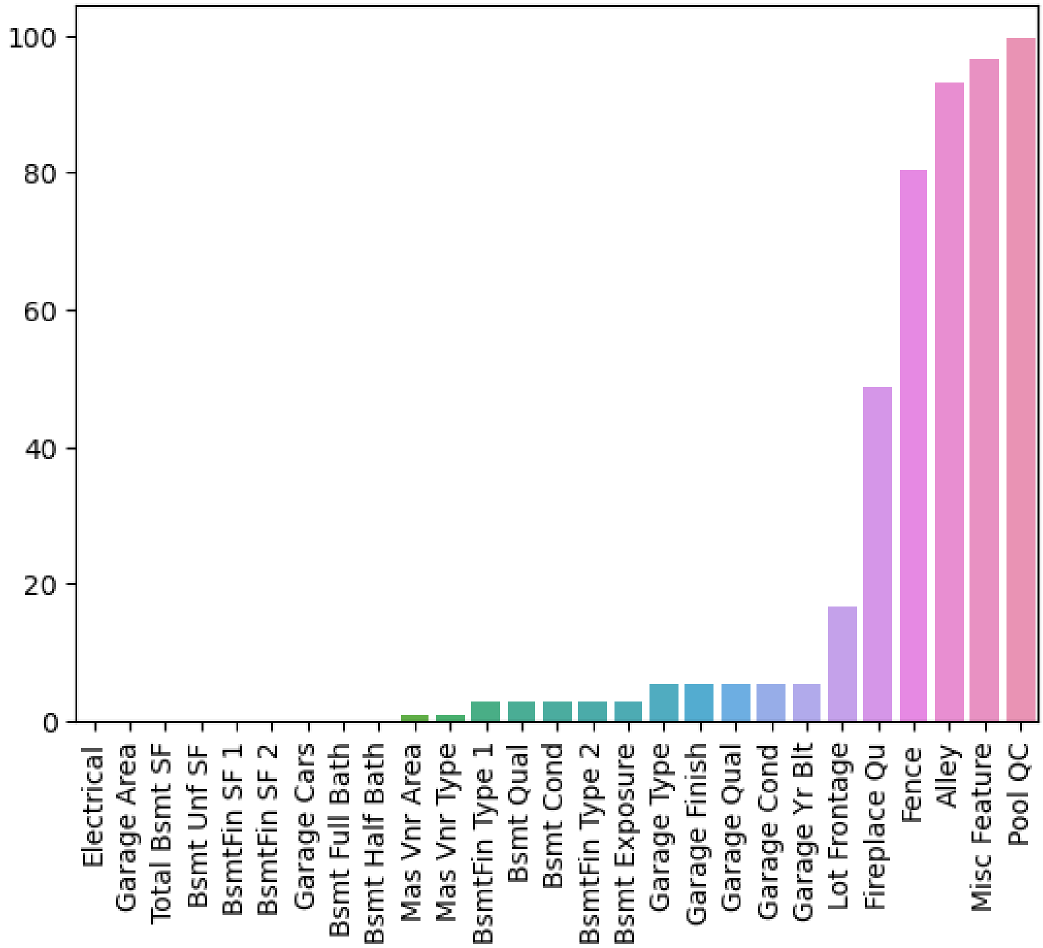
```
# let create function for converting into percentage and show values
which are more than 0
def percent_missing(df):
    percent_nan = 100*(df.isnull().sum())/len(df)
    percent_nan = percent_nan[percent_nan > 0].sort_values()

    return percent_nan

per_mis=percent_missing(df)
per_mis
```

```
Electrical          0.034165
Garage Area         0.034165
Total Bsmt SF       0.034165
Bsmt Unf SF         0.034165
BsmtFin SF 1        0.034165
BsmtFin SF 2        0.034165
Garage Cars         0.034165
Bsmt Full Bath      0.068329
Bsmt Half Bath      0.068329
Mas Vnr Area        0.785787
Mas Vnr Type        0.785787
BsmtFin Type 1      2.733174
Bsmt Qual           2.733174
Bsmt Cond           2.733174
BsmtFin Type 2      2.767339
Bsmt Exposure       2.835668
Garage Type         5.363854
Garage Finish       5.432183
Garage Qual         5.432183
Garage Cond         5.432183
Garage Yr Blt       5.432183
Lot Frontage       16.740690
Fireplace Qu       48.582166
Fence              80.457807
Alley              93.235395
Misc Feature       96.412709
Pool QC            99.590024
dtype: float64
```

```
# Here we are going to plot barplot to see missing data numbers
sns.barplot(x=per_mis.index,y=per_mis)
plt.show()
```




```
# Above we cant see names on x axis so we will use plt.xticks
sns.barplot(x=per_mis.index,y=per_mis)
plt.xticks(rotation=90)
plt.show();
```



Removing Features or Removing Rows

If only a few rows relative to the size of your dataset are missing some values, then it might just be a good idea to drop those rows. What does this cost you in terms of performace? It essentialy removes potential training/testing data, but if its only a few rows, its unlikely to change performance.

Sometimes it is a good idea to remove a feature entirely if it has too many null values. However, you should carefully consider why it has so many null values, in certain situations null could just be used as a separate category.

Take for example a feature column for the number of cars that can fit into a garage. Perhaps if there is no garage then there is a null value, instead of a zero. It probably makes more sense to quickly fill the null values in this case with a zero instead of a null. Only you can decide based off your domain expertise and knowledge of the data set!

Working based on Rows Missing Data

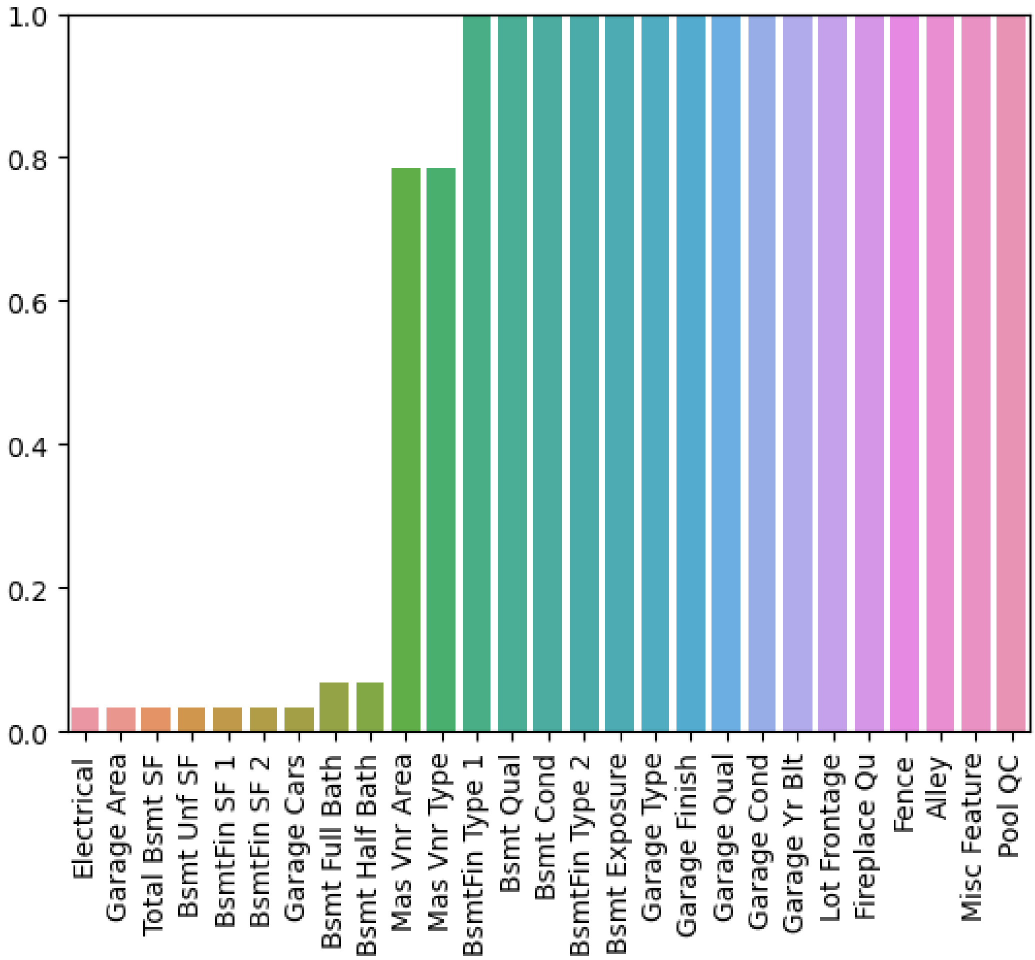
Filling in Data or Dropping Data?

Let's explore how to choose to remove or fill in missing data for rows that are missing some data. Let's choose some threshold where we decide it is ok to drop a row if its missing some data (instead of attempting to fill in that missing data point). We will choose 1% as our threshold. This means if less than 1% of the rows are missing this feature, we will consider just dropping that row, instead of dealing with the feature itself. There is no right answer here, just use common sense and your domain knowledge of the dataset, obviously you don't want to drop a very high threshold like 50% , you should also explore correlation to the dataset, maybe it makes sense to drop the feature instead.

Based on the text description of the features, you will see that most of this missing data is actually NaN on purpose as a placeholder for 0 or "none".

Example of Filling in Data : Basement Columns

```
# Here we are going to find those are between 0 and 1
sns.barplot(x=per_mis.index,y=per_mis)
plt.xticks(rotation=90)

# Set 1% Threshold
plt.ylim(0,1)
plt.show();
```



Let's drop or fill the rows based on this data. You could either manually fill in the data (especially the Basement data based on the description text file) OR you could simply drop the row and not consider it. Watch the video for a full explanation of this, in reality it probably makes more sense to fill in the Missing Basement data since its well described in the text description.

```
# Could also imply we should ex
per_mis[per_mis<1]
```

```
Electrical        0.034165
Garage Area       0.034165
Total Bsmt SF     0.034165
Bsmt Unf SF       0.034165
BsmtFin SF 1      0.034165
BsmtFin SF 2      0.034165
Garage Cars       0.034165
Bsmt Full Bath    0.068329
Bsmt Half Bath    0.068329
Mas Vnr Area      0.785787
Mas Vnr Type      0.785787
dtype: float64
```

Here we see that there is 0.034165 is missing in many columns we have to see what is it

```
100/len(df)
```

```
0.0341646737273659
```

That means there is one row missing in each of them where values are 0.034165 and 0.068329 is double of 0.034165

```
df[df['Electrical'].isnull()]
```

```
      MS SubClass MS Zoning  Lot Frontage  Lot Area Street Alley Lot 
Shape  \
1576           80        RL          73.0      9735   Pave   NaN      
Reg   

     Land Contour Utilities Lot Config  ... Pool Area Pool QC Fence  \
1576          Lvl    AllPub     Inside  ...         0     NaN   NaN   

     Misc Feature Misc Val Mo Sold  Yr Sold  Sale Type  Sale Condition 
\
1576          NaN        0       5     2008        WD           Normal   


      SalePrice  
1576     167500  

[1 rows x 80 columns]
```

```
# we want to find out garage area of this data
df[df['Electrical'].isnull()]['Garage Area']
```

```
1576    400.0
Name: Garage Area, dtype: float64
```

```
df[df['Bsmt Half Bath'].isnull()]
```

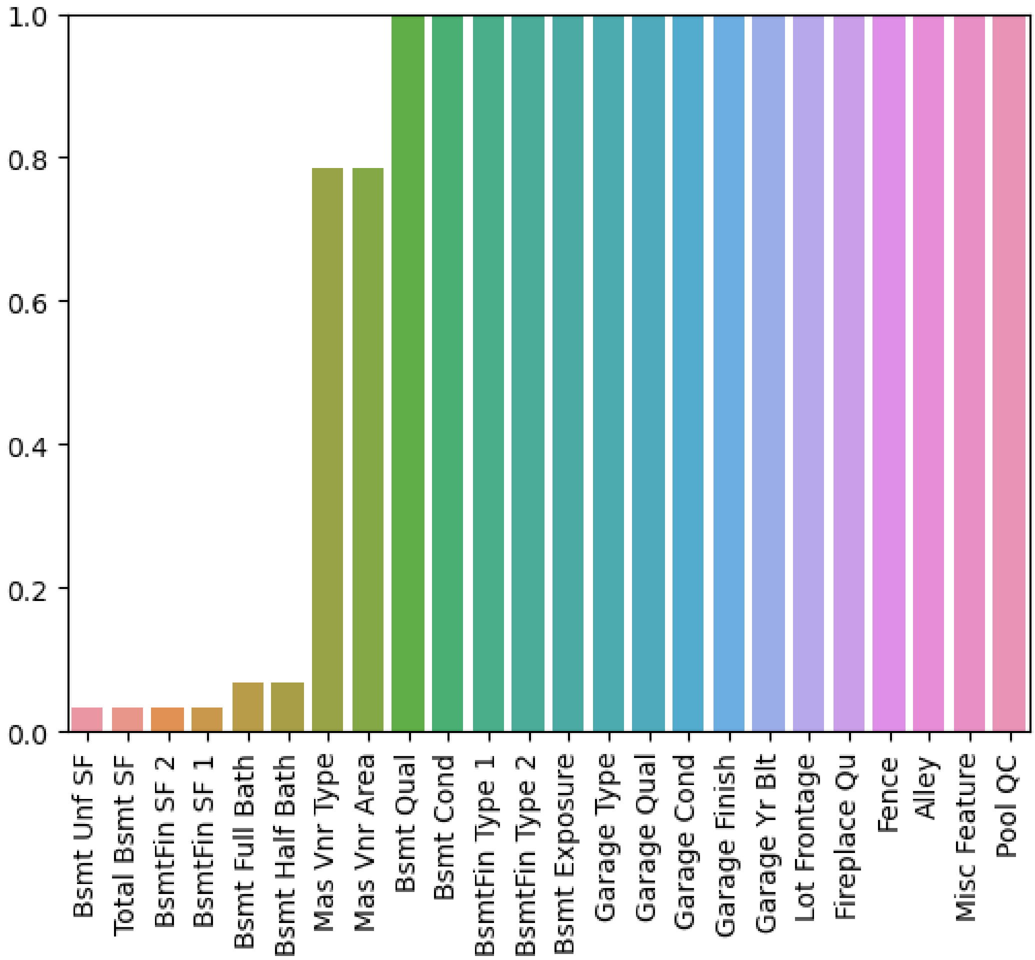
```
      MS SubClass MS Zoning  Lot Frontage  Lot Area Street Alley Lot 
Shape  \
1341           20        RM          99.0      5940   Pave   NaN      
IR1   
1497           20        RL         123.0     47007   Pave   NaN      
IR1   

     Land Contour Utilities Lot Config  ... Pool Area Pool QC 
Fence  \
1341          Lvl    AllPub        FR3  ...         0     NaN  MnPrv   

1497          Lvl    AllPub     Inside  ...         0     NaN    NaN   
```

```
     Misc Feature Misc Val Mo Sold  Yr Sold  Sale Type  Sale Condition
\
1341          NaN        0       4     2008      ConLD         Abnorml

1497          NaN        0       7     2008        WD           Normal


      SalePrice
1341      79000
1497     284700

[2 rows x 80 columns]
```

```
# Here we are going to rows drop where electrical and Garage cars
values are null
df = df.dropna(axis=0,subset=['Electrical','Garage Cars'])

per_mis = percent_missing(df)

per_mis[per_mis<1]
```

```
Bsmt Unf SF       0.034188
Total Bsmt SF     0.034188
BsmtFin SF 2      0.034188
BsmtFin SF 1      0.034188
Bsmt Full Bath    0.068376
Bsmt Half Bath    0.068376
Mas Vnr Type      0.786325
Mas Vnr Area      0.786325
dtype: float64
```

```
sns.barplot(x=per_mis.index,y=per_mis)
plt.xticks(rotation=90)
plt.ylim(0,1)
plt.show();
```




```
df[df['Bsmt Half Bath'].isnull()]
```

```
      MS SubClass MS Zoning  Lot Frontage  Lot Area Street Alley Lot
Shape  \
1341           20        RM          99.0      5940   Pave   NaN
IR1
1497           20        RL         123.0     47007   Pave   NaN
IR1

     Land Contour Utilities Lot Config  ... Pool Area Pool QC
Fence  \
1341          Lvl    AllPub        FR3  ...         0     NaN  MnPrv

1497          Lvl    AllPub     Inside  ...         0     NaN    NaN


     Misc Feature Misc Val Mo Sold  Yr Sold  Sale Type  Sale Condition
\
1341          NaN        0       4     2008      ConLD         Abnorml
```

```
1497           NaN         0       7     2008        WD           Normal


      SalePrice
1341      79000
1497     284700

[2 rows x 80 columns]
```

```
df[df['Bsmt Full Bath'].isnull()]
```

```
      MS SubClass MS Zoning  Lot Frontage  Lot Area Street Alley Lot
Shape  \
1341           20        RM          99.0      5940   Pave   NaN
IR1
1497           20        RL         123.0     47007   Pave   NaN
IR1

     Land Contour Utilities Lot Config  ... Pool Area Pool QC
Fence  \
1341          Lvl    AllPub        FR3  ...         0     NaN  MnPrv

1497          Lvl    AllPub     Inside  ...         0     NaN    NaN


     Misc Feature Misc Val Mo Sold  Yr Sold  Sale Type  Sale Condition
\
1341          NaN        0       4     2008      ConLD         Abnorml

1497          NaN        0       7     2008        WD           Normal


      SalePrice
1341      79000
1497     284700

[2 rows x 80 columns]
```

```
df[df['BsmtFin SF 2'].isnull()]
```

```
      MS SubClass MS Zoning  Lot Frontage  Lot Area Street Alley Lot
Shape  \
1341           20        RM          99.0      5940   Pave   NaN
IR1

     Land Contour Utilities Lot Config  ... Pool Area Pool QC
Fence  \
1341          Lvl    AllPub        FR3  ...         0     NaN  MnPrv
```

```
    Misc Feature Misc Val Mo Sold  Yr Sold  Sale Type  Sale Condition
\
1341          NaN        0       4     2008      ConLD         Abnorml


      SalePrice
1341     79000

[1 rows x 80 columns]
```

```
df[df['BsmtFin SF 1'].isnull()]
```

```
      MS SubClass MS Zoning  Lot Frontage  Lot Area Street Alley Lot
Shape  \
1341           20        RM          99.0      5940   Pave   NaN
IR1

     Land Contour Utilities Lot Config  ... Pool Area Pool QC
Fence  \
1341          Lvl    AllPub        FR3  ...         0     NaN  MnPrv


     Misc Feature Misc Val Mo Sold  Yr Sold  Sale Type  Sale Condition
\
1341          NaN        0       4     2008      ConLD         Abnorml


      SalePrice
1341     79000

[1 rows x 80 columns]
```

Here we see that there is 1341 common in all Bsmt

we are going to check the data description and we find that in these property basement is not available so we are not going to drop it we are going to put 0 rather then droping that

Filling in data based on column names. There are 2 types of basement features, numerical and string descriptives.

The numerical basement columns:

```
# These are selected by looking at description carefully

# BSMT NUMERIC ColUMNS --> fillna 0
bsmt_num_cols = ['Bsmt Unf SF', 'Total Bsmt SF', 'BsmtFin SF 2',
'BsmtFin SF 1',
       'Bsmt Full Bath', 'Bsmt Half Bath']

df[bsmt_num_cols] = df[bsmt_num_cols].fillna(0)
```

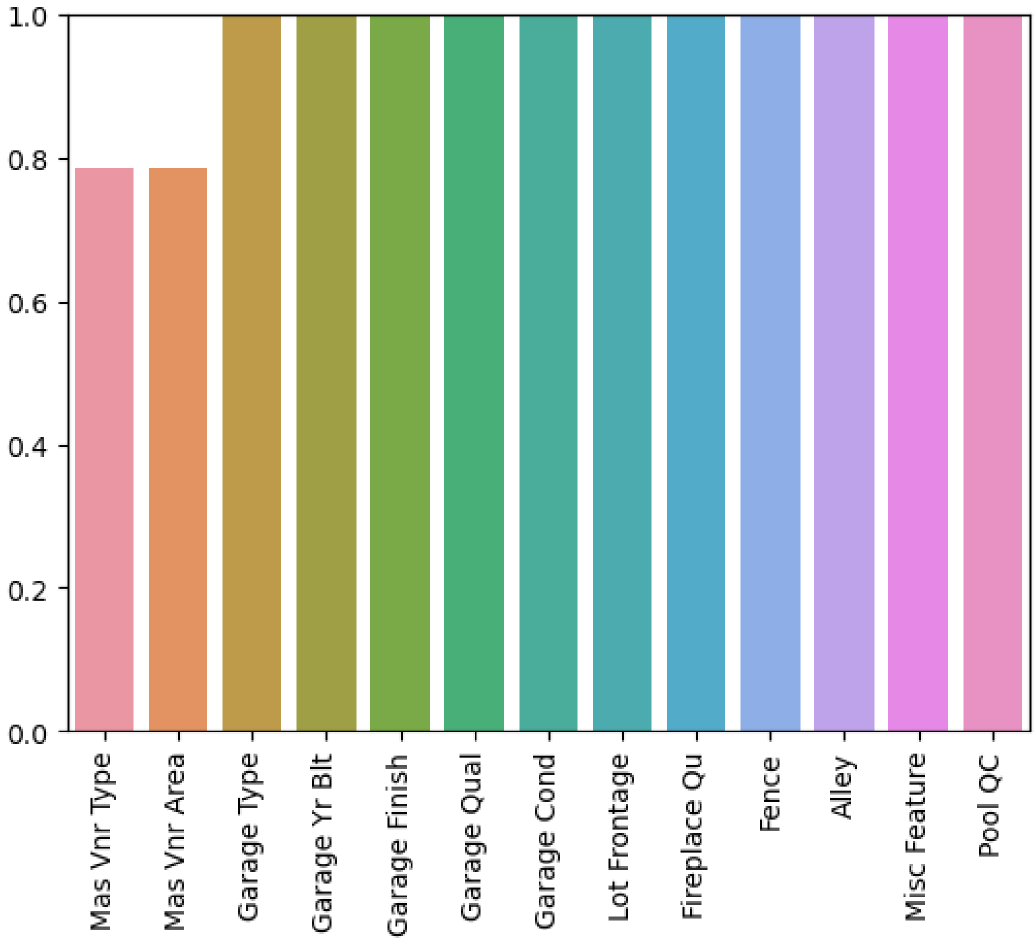
```
# BSMT STRING COLUMNS

bsmt_str_cols = ['Bsmt Qual','Bsmt Cond','Bsmt Exposure','BsmtFin Type 1','BsmtFin Type 2',]
df[bsmt_str_cols] = df[bsmt_str_cols].fillna('None')

df[df['Bsmt Full Bath'].isnull()]
```

```
Empty DataFrame
Columns: [MS SubClass, MS Zoning, Lot Frontage, Lot Area, Street, Alley, Lot Shape, Land Contour, Utilities, Lot Config, Land Slope, Neighborhood, Condition 1, Condition 2, Bldg Type, House Style, Overall Qual, Overall Cond, Year Built, Year Remod/Add, Roof Style, Roof Matl, Exterior 1st, Exterior 2nd, Mas Vnr Type, Mas Vnr Area, Exter Qual, Exter Cond, Foundation, Bsmt Qual, Bsmt Cond, Bsmt Exposure, BsmtFin Type 1, BsmtFin SF 1, BsmtFin Type 2, BsmtFin SF 2, Bsmt Unf SF, Total Bsmt SF, Heating, Heating QC, Central Air, Electrical, 1st Flr SF, 2nd Flr SF, Low Qual Fin SF, Gr Liv Area, Bsmt Full Bath, Bsmt Half Bath, Full Bath, Half Bath, Bedroom AbvGr, Kitchen AbvGr, Kitchen Qual, TotRms AbvGrd, Functional, Fireplaces, Fireplace Qu, Garage Type, Garage Yr Blt, Garage Finish, Garage Cars, Garage Area, Garage Qual, Garage Cond, Paved Drive, Wood Deck SF, Open Porch SF, Enclosed Porch, 3Ssn Porch, Screen Porch, Pool Area, Pool QC, Fence, Misc Feature, Misc Val, Mo Sold, Yr Sold, Sale Type, Sale Condition, SalePrice]
Index: []

[0 rows x 80 columns]
```

Dropping Rows

A few of these features appear that it is just one or two rows missing the data. Based on our description .txt file of the dataset, we could also fill in these data points easily, and that is the more correct approach, but here we show how to drop in case you find yourself in a situation where it makes more sense to drop a row, based on missing column features.

```
df.dropna() ---
    subset : array-like, optional
            Labels along other axis to consider, e.g. if you are dropping rows
            these would be a list of columns to include.

per_mis = percent_missing(df)

sns.barplot(x=per_mis.index,y=per_mis)
plt.xticks(rotation=90)
plt.ylim(0,1)
plt.show();
```

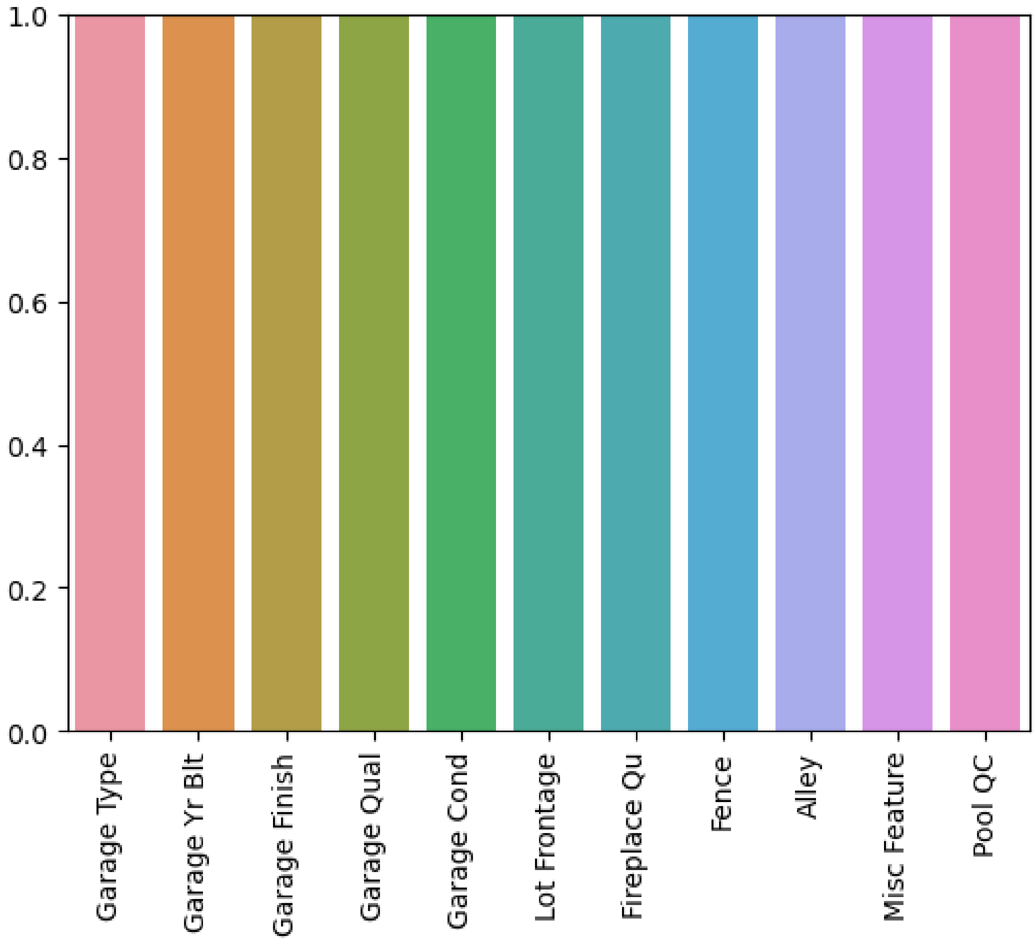


Now we are fix all the rows just 2 left (Mas Vnr Type and Mas Vnr Area) then we will focus on columns

Mas Vnr Feature

Based on the Description Text File, Mas Vnr Type and Mas Vnr Area being missing (NaN) is likely to mean the house simply just doesn't have a masonry veneer, in which case, we will fill in this data as we did before.

```
df['Mas Vnr Type'] = df['Mas Vnr Type'].fillna("None")

df['Mas Vnr Area'] = df['Mas Vnr Area'].fillna(0)

per_mis = percent_missing(df)

sns.barplot(x=per_mis.index,y=per_mis)
plt.xticks(rotation=90)
plt.ylim(0,1)
plt.show();
```



Filling In Missing Column Data

Our previous approaches were based more on rows missing data, now we will take an approach based on the column features themselves, since larger percentages of the data appears to be missing.

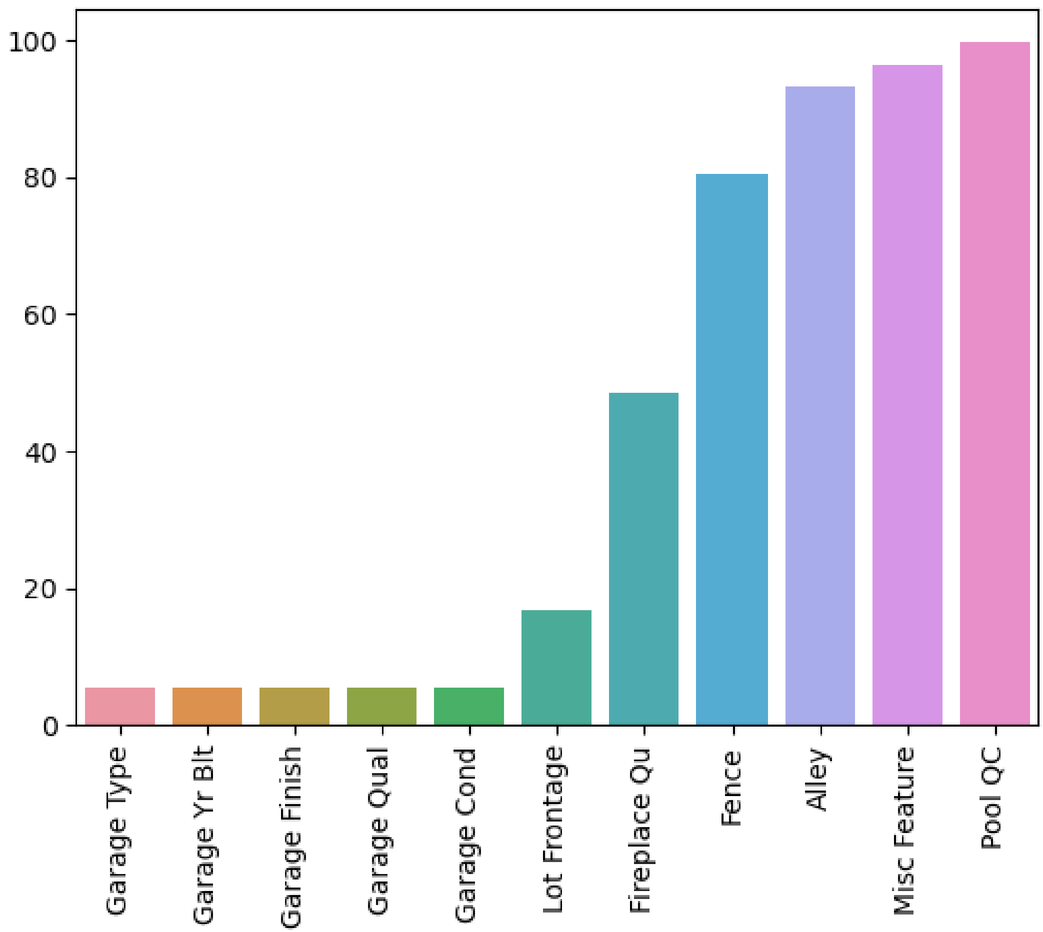
Garage Columns

Based on the data description, these NaN seem to indicate no garage, so we will substitute with "None" or 0.

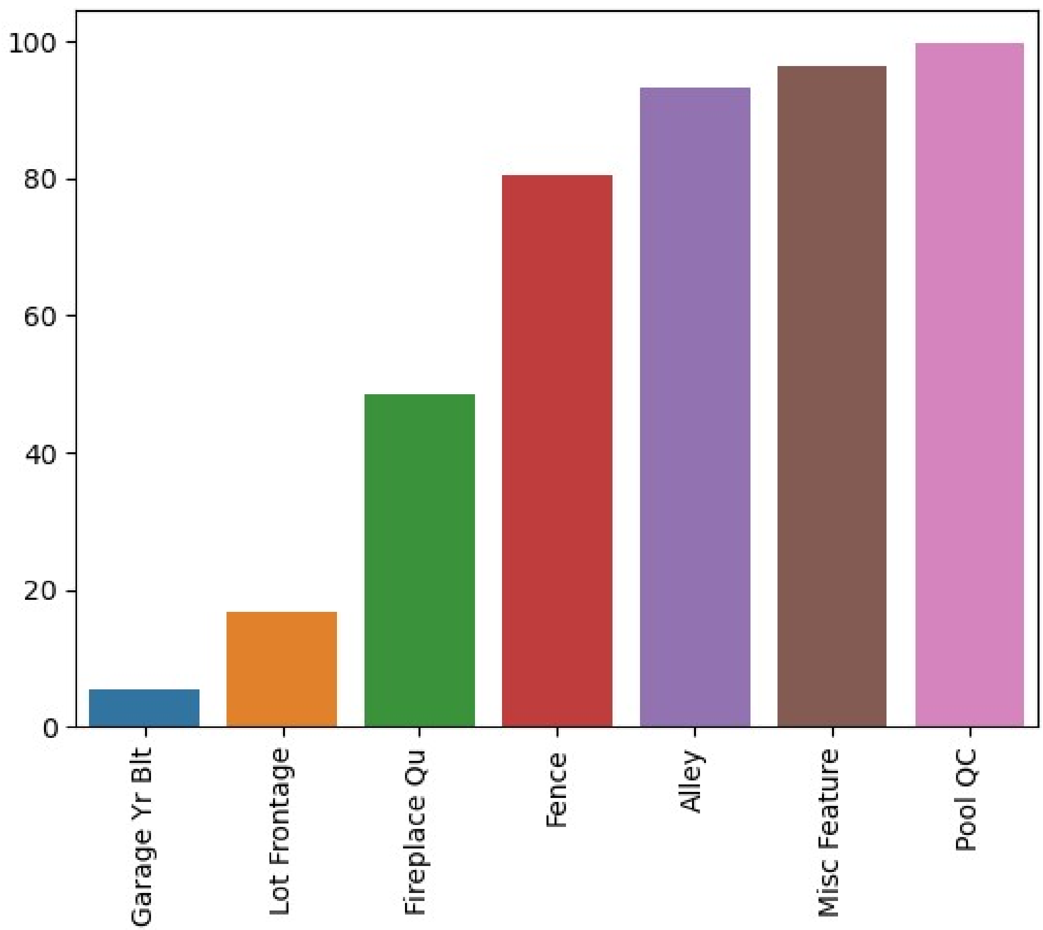
```
df.dropna() ---
    subset : array-like, optional
            Labels along other axis to consider, e.g. if you are
dropping rows
            these would be a list of columns to include.
```

```
per_mis = percent_missing(df)

sns.barplot(x=per_mis.index,y=per_mis)
```

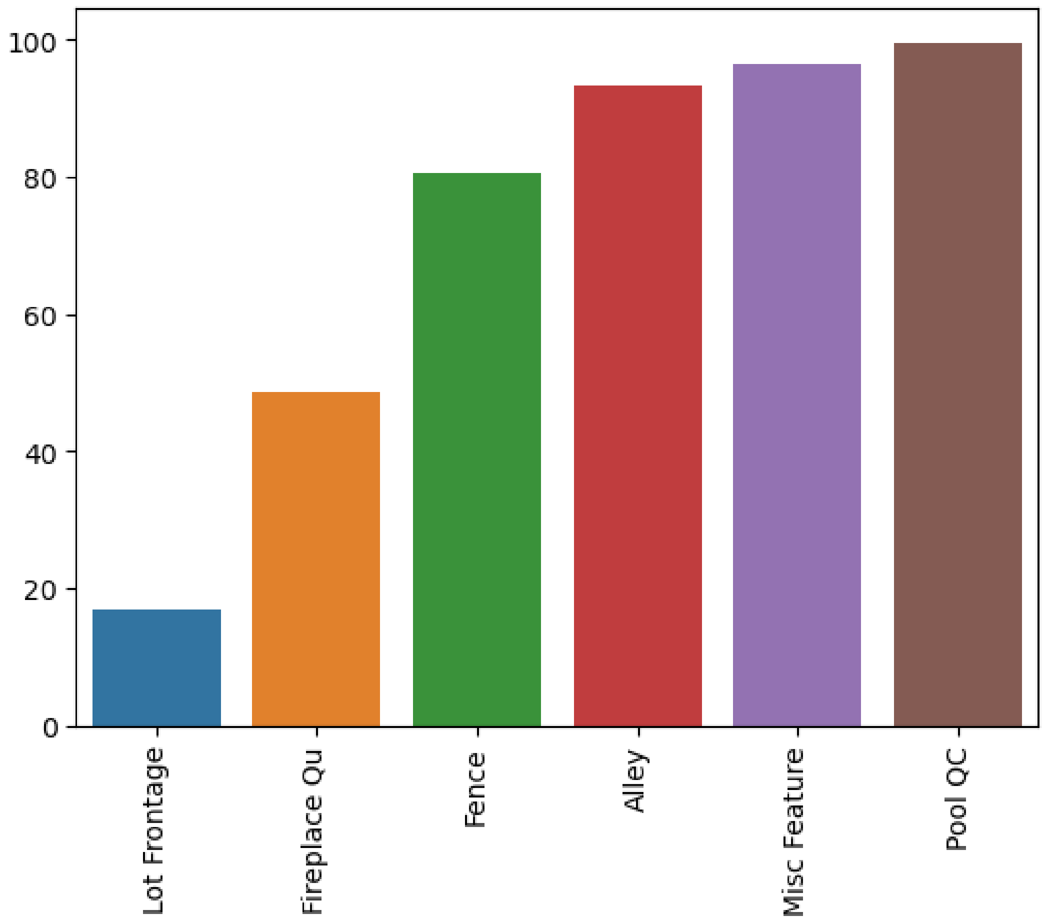
```
plt.xticks(rotation=90)
plt.show();
```




```
# BY reading the description we find out where data is missing in
garage type, Garage Finish ... that means dont have garage so we haev
to fill none there
gar_str_cols =['Garage Type','Garage Finish','Garage Qual','Garage
Cond']

df[gar_str_cols]=df[gar_str_cols].fillna("None")

per_mis = percent_missing(df)

sns.barplot(x=per_mis.index,y=per_mis)
plt.xticks(rotation=90)
plt.show();
```



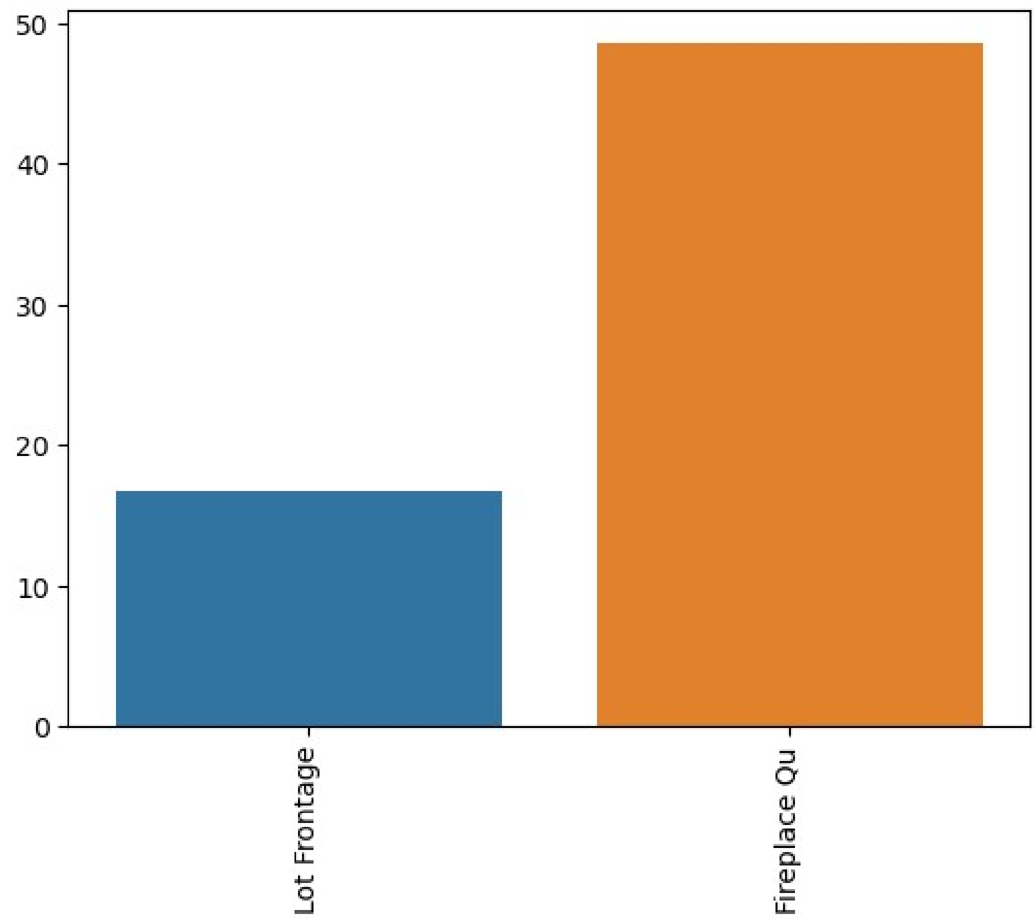

```
df['Garage Yr Blt'] = df['Garage Yr Blt'].fillna(0)

per_mis = percent_missing(df)

sns.barplot(x=per_mis.index,y=per_mis)
plt.xticks(rotation=90)
plt.show();
```



Dropping Feature Columns

Sometimes you may want to take the approach that above a certain missing percentage threshold, you will simply remove the feature from all the data. For example if 99% of rows are missing a feature, it will not be predictive, since almost all the data does not have any value for it. In our particular data set, many of these high percentage NaN features are actually plasceholders for "none" or 0. But for the sake of showing variations on dealing with missing data, we will remove these features, instead of filling them in with the appropriate value.

```
# Here we are droping all those where missing percentage are high
df= df.drop(['Fence', 'Alley', 'Misc Feature', 'Pool QC'],axis=1)

per_mis = percent_missing(df)

sns.barplot(x=per_mis.index,y=per_mis)
plt.xticks(rotation=90)
plt.show();
```




```
df['Fireplace Qu'].value_counts()
```

```
Gd    741
TA    600
Fa     75
Po     46
Ex     43
Name: Fireplace Qu, dtype: int64
```

Filling in Fireplace Quality based on Description Text

```
df['Fireplace Qu']= df['Fireplace Qu'].fillna("None")
```

```
df['Lot Frontage']
```

```
0       141.0
1        80.0
2        81.0
3        93.0
4        74.0
        ...
2922     37.0
2923      NaN
2924     62.0
```

```
2925     77.0
2926     74.0
Name: Lot Frontage, Length: 2925, dtype: float64
```

Neighborhood: Physical locations with Ames city limits

LotFrontage: Linear feet of street connected to property

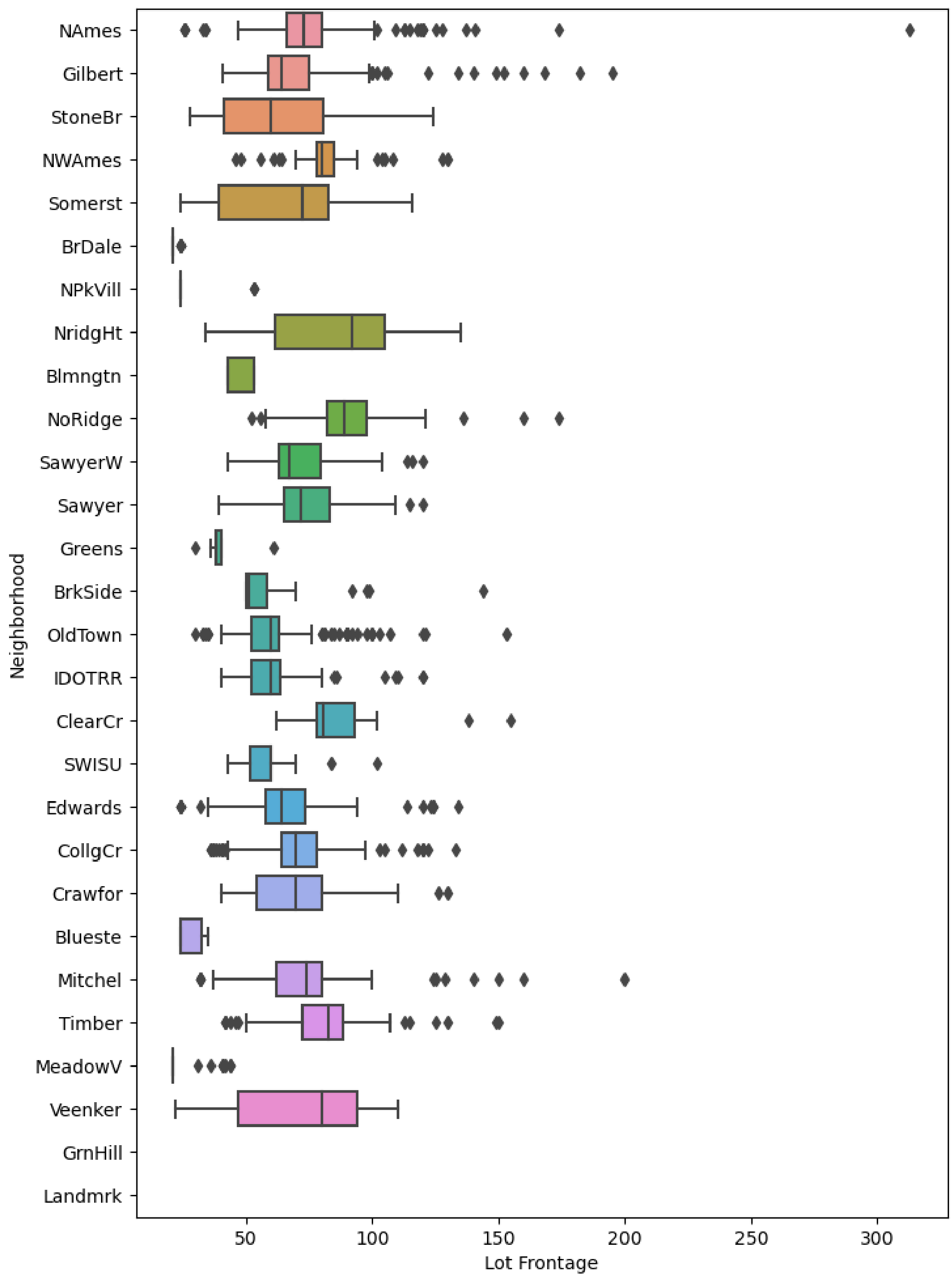
and these are link to each other

Impute Missing Data based on other Features

There are more complex methods, but usually the simpler the better, it avoids building models on top of other models.

More Info on Options: https://scikit-learn.org/stable/modules/impute.html

```
plt.figure(figsize=(8,12),dpi=100)
sns.boxplot(x='Lot Frontage',y='Neighborhood',data=df,orient= 'h')
plt.show()
```




```
df.groupby('Neighborhood')['Lot Frontage'].mean()
```

```
Neighborhood
Blmngtn    46.900000
Blueste    27.300000
BrDale     21.500000
```

```
BrkSide    55.789474
ClearCr    88.150000
CollgCr    71.336364
Crawfor    69.951807
Edwards    64.794286
Gilbert    74.207207
Greens     41.000000
GrnHill          NaN
IDOTRR     62.383721
Landmrk          NaN
MeadowV    25.606061
Mitchel    75.144444
NAmes      75.210667
NPkVill    28.142857
NWAmes     81.517647
NoRidge    91.629630
NridgHt    84.184049
OldTown    61.777293
SWISU      59.068182
Sawyer     74.551020
SawyerW    70.669811
Somerst    64.549383
StoneBr    62.173913
Timber     81.303571
Veenker    72.000000
Name: Lot Frontage, dtype: float64
```

```
# Here we are going to use apply function with groupby

df['Lot Frontage']=df.groupby('Neighborhood')['Lot
Frontage'].transform(lambda value: value.fillna(value.mean()))

df.isnull().sum()
```

```
MS SubClass       0
MS Zoning         0
Lot Frontage      3
Lot Area          0
Street            0
                 ..
Mo Sold           0
Yr Sold           0
Sale Type         0
Sale Condition    0
SalePrice         0
Length: 76, dtype: int64
```

still we can see there 3 missing values we have to fix them too

```
df['Lot Frontage'] = df['Lot Frontage'].fillna(0)
```

```python
df.isnull().sum()
```

```
MS SubClass       0
MS Zoning         0
Lot Frontage      0
Lot Area          0
Street            0
                 ..
Mo Sold           0
Yr Sold           0
Sale Type         0
Sale Condition    0
SalePrice         0
Length: 76, dtype: int64
```

No Data is missing now

Great! We no longer have any missing data in our entire data set! Keep in mind, we should eventually turn all these transformations into an easy to use function. For now, lets' save this dataset:

Dealing with Categorial Data

Many machine learning models can not deal with categorical data set as strings. For example linear regression can not apply a a Beta Coefficent to colors like "red" or "blue". Instead we need to convert these categories into "dummy" variables, otherwise known as "one-hot" encoding.

Numerical Column to Categorical

We need to be careful when it comes to encoding categories as numbers. We want to make sure that the numerical relationship makes sense for a model. For example, the encoding MSSubClass is essentially just a number code per class:

```
MSSubClass: Identifies the type of dwelling involved in the sale.

    20     1-STORY 1946 & NEWER ALL STYLES
    30     1-STORY 1945 & OLDER
    40     1-STORY W/FINISHED ATTIC ALL AGES
    45     1-1/2 STORY - UNFINISHED ALL AGES
    50     1-1/2 STORY FINISHED ALL AGES
    60     2-STORY 1946 & NEWER
    70     2-STORY 1945 & OLDER
    75     2-1/2 STORY ALL AGES
    80     SPLIT OR MULTI-LEVEL
    85     SPLIT FOYER
    90     DUPLEX - ALL STYLES AND AGES
   120     1-STORY PUD (Planned Unit Development) - 1946 & NEWER
   150     1-1/2 STORY PUD - ALL AGES
```

```
   160     2-STORY PUD - 1946 & NEWER
   180     PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
   190     2 FAMILY CONVERSION - ALL STYLES AND AGES
```

The number itself does not appear to have a relationship to the other numbers. While 30 > 20 is True, it doesn't really make sense that "1-STORY 1945 & OLDER" > "1-STORY 1946 & NEWER ALL STYLES". Keep in mind, this isn't always the case, for example 1st class seats versus 2nd class seats encoded as 1 and 2. Make sure you fully understand your data set to examine what needs to be converted/changed.

MSSubClass

```
# Convert to String
df['MS SubClass'] = df['MS SubClass'].apply(str)
```

Creating "Dummy" Variables

Avoiding MultiCollinearity and the Dummy Variable Trap

https://stats.stackexchange.com/questions/144372/dummy-variable-trap

```
# Example of creating Dummies
direction = pd.Series(['Up','Up','Down'])
```

```
direction
```

```
0      Up
1      Up
2    Down
dtype: object
```

```
pd.get_dummies(direction)
```

```
   Down  Up
0     0   1
1     0   1
2     1   0
```

```
# Here we can drop one column
pd.get_dummies(direction,drop_first=True)
```

```
   Up
0   1
1   1
2   0
```

Creating Dummy Variables from Object Columns

https://pandas.pydata.org/pandas-docs/stable/reference/api/
pandas.DataFrame.select_dtypes.html

```python
# Here it will return data type is object or string
df.select_dtypes(include='object')
```

```
     MS SubClass MS Zoning Street Lot Shape Land Contour Utilities Lot
Config  \
0             20        RL   Pave       IR1          Lvl    AllPub
Corner
1             20        RH   Pave       Reg          Lvl    AllPub
Inside
2             20        RL   Pave       IR1          Lvl    AllPub
Corner
3             20        RL   Pave       Reg          Lvl    AllPub
Corner
4             60        RL   Pave       IR1          Lvl    AllPub
Inside
...          ...       ...    ...       ...          ...       ...
...
2922          80        RL   Pave       IR1          Lvl    AllPub
CulDSac
2923          20        RL   Pave       IR1          Low    AllPub
Inside
2924          85        RL   Pave       Reg          Lvl    AllPub
Inside
2925          20        RL   Pave       Reg          Lvl    AllPub
Inside
2926          60        RL   Pave       Reg          Lvl    AllPub
Inside

     Land Slope Neighborhood Condition 1  ... Kitchen Qual Functional
\
0           Gtl        NAmes        Norm  ...           TA        Typ

1           Gtl        NAmes       Feedr  ...           TA        Typ

2           Gtl        NAmes        Norm  ...           Gd        Typ

3           Gtl        NAmes        Norm  ...           Ex        Typ

4           Gtl      Gilbert        Norm  ...           TA        Typ

...         ...          ...         ...  ...          ...        ...

2922        Gtl      Mitchel        Norm  ...           TA        Typ

2923        Mod      Mitchel        Norm  ...           TA        Typ

2924        Gtl      Mitchel        Norm  ...           TA        Typ

2925        Mod      Mitchel        Norm  ...           TA        Typ
```

```
2926         Mod      Mitchel        Norm  ...            TA         Typ


     Fireplace Qu Garage Type Garage Finish Garage Qual Garage Cond  \
0              Gd      Attchd           Fin          TA          TA
1            None      Attchd           Unf          TA          TA
2            None      Attchd           Unf          TA          TA
3              TA      Attchd           Fin          TA          TA
4              TA      Attchd           Fin          TA          TA
...           ...         ...           ...         ...         ...
2922         None      Detchd           Unf          TA          TA
2923         None      Attchd           Unf          TA          TA
2924         None        None          None        None        None
2925           TA      Attchd           RFn          TA          TA
2926           TA      Attchd           Fin          TA          TA

     Paved Drive Sale Type Sale Condition
0              P       WD          Normal
1              Y       WD          Normal
2              Y       WD          Normal
3              Y       WD          Normal
4              Y       WD          Normal
...          ...       ...            ...
2922           Y       WD          Normal
2923           Y       WD          Normal
2924           Y       WD          Normal
2925           Y       WD          Normal
2926           Y       WD          Normal

[2925 rows x 40 columns]
```

Here we are going to create two data one is for numericals and other is for stirngs we will apply dummies on object and then we will merge both

```
my_object_df=df.select_dtypes(include='object')

my_numeric_df=df.select_dtypes(exclude='object')
```

Converting

```
df_object_dummies=pd.get_dummies(my_object_df,drop_first=True)

df_object_dummies

      MS SubClass_150  MS SubClass_160  MS SubClass_180  MS
SubClass_190  \
0                   0                0                0
0
1                   0                0                0
0
2                   0                0                0
```

```
0
3                0                0                0
0
4                0                0                0
0
...            ...              ...              ...               .
..
2922             0                0                0
0
2923             0                0                0
0
2924             0                0                0
0
2925             0                0                0
0
2926             0                0                0
0

      MS SubClass_20  MS SubClass_30  MS SubClass_40  MS
SubClass_45  \
0                  1               0               0               0

1                  1               0               0               0

2                  1               0               0               0

3                  1               0               0               0

4                  0               0               0               0

...              ...             ...             ...             ...

2922               0               0               0               0

2923               1               0               0               0

2924               0               0               0               0

2925               1               0               0               0

2926               0               0               0               0


      MS SubClass_50  MS SubClass_60  ...  Sale Type_ConLw  Sale
Type_New  \
0                  0               0  ...                0
0
1                  0               0  ...                0
0
```

```
2                    0               0  ...                0
0
3                    0               0  ...                0
0
4                    0               1  ...                0
0
...                ...             ...  ...              ...
...
2922                 0               0  ...                0
0
2923                 0               0  ...                0
0
2924                 0               0  ...                0
0
2925                 0               0  ...                0
0
2926                 0               1  ...                0
0

      Sale Type_Oth  Sale Type_VWD  Sale Type_WD   Sale
Condition_AdjLand  \
0                 0              0              1
0
1                 0              0              1
0
2                 0              0              1
0
3                 0              0              1
0
4                 0              0              1
0
...             ...            ...            ...
...
2922              0              0              1
0
2923              0              0              1
0
2924              0              0              1
0
2925              0              0              1
0
2926              0              0              1
0

      Sale Condition_Alloca  Sale Condition_Family  Sale
Condition_Normal  \
0                         0                      0
1
1                         0                      0
1
```

```
2                          0                       0
1
3                          0                       0
1
4                          0                       0
1
...                      ...                     ...
...
2922                       0                       0
1
2923                       0                       0
1
2924                       0                       0
1
2925                       0                       0
1
2926                       0                       0
1

      Sale Condition_Partial
0                          0
1                          0
2                          0
3                          0
4                          0
...                      ...
2922                       0
2923                       0
2924                       0
2925                       0
2926                       0

[2925 rows x 238 columns]
```

```
final_df = pd.concat([my_numeric_df,df_object_dummies],axis=1)
final_df
```

```
      Lot Frontage  Lot Area  Overall Qual  Overall Cond  Year
Built  \
0       141.000000     31770             6             5        1960

1        80.000000     11622             5             6        1961

2        81.000000     14267             6             6        1958

3        93.000000     11160             7             5        1968

4        74.000000     13830             5             5        1997

...            ...       ...           ...           ...         ...
```

```
2922     37.000000      7937             6             6        1984

2923     75.144444      8885             5             5        1983

2924     62.000000     10441             5             5        1992

2925     77.000000     10010             5             5        1974

2926     74.000000      9627             7             5        1993


      Year Remod/Add  Mas Vnr Area  BsmtFin SF 1  BsmtFin SF 2  Bsmt
Unf SF  \
0               1960         112.0         639.0           0.0
441.0
1               1961           0.0         468.0         144.0
270.0
2               1958         108.0         923.0           0.0
406.0
3               1968           0.0        1065.0           0.0
1045.0
4               1998           0.0         791.0           0.0
137.0
...              ...           ...           ...           ...
...
2922            1984           0.0         819.0           0.0
184.0
2923            1983           0.0         301.0         324.0
239.0
2924            1992           0.0         337.0           0.0
575.0
2925            1975           0.0        1071.0         123.0
195.0
2926            1994          94.0         758.0           0.0
238.0

      ...  Sale Type_ConLw  Sale Type_New  Sale Type_Oth  Sale
Type_VWD  \
0     ...                0              0              0
0
1     ...                0              0              0
0
2     ...                0              0              0
0
3     ...                0              0              0
0
4     ...                0              0              0
0
```

```
...   ...              ...            ...            ...            ..
.
2922  ...                0              0              0
0
2923  ...                0              0              0
0
2924  ...                0              0              0
0
2925  ...                0              0              0
0
2926  ...                0              0              0
0

      Sale Type_WD   Sale Condition_AdjLand  Sale Condition_Alloca  \
0                1                        0                      0
1                1                        0                      0
2                1                        0                      0
3                1                        0                      0
4                1                        0                      0
...            ...                      ...                    ...
2922             1                        0                      0
2923             1                        0                      0
2924             1                        0                      0
2925             1                        0                      0
2926             1                        0                      0

      Sale Condition_Family  Sale Condition_Normal  Sale
Condition_Partial
0                         0                      1
0
1                         0                      1
0
2                         0                      1
0
3                         0                      1
0
4                         0                      1
0
...                     ...                    ...
...
2922                      0                      1
0
2923                      0                      1
0
2924                      0                      1
0
2925                      0                      1
0
2926                      0                      1
0
```

```
[2925 rows x 274 columns]
```

Final Thoughts

Keep in mind, we don't know if 274 columns is very useful. More columns doesn't necessarily lead to better results. In fact, we may want to further remove columns (or later on use a model with regularization to choose important columns for us). What we have done here has greatly expanded the ratio of rows to columns, which may actually lead to worse performance (however you don't know until you've actually compared multiple models/approaches).

```
final_df.corr()['SalePrice'].sort_values()
```

```
Exter Qual_TA       -0.591459
Kitchen Qual_TA     -0.527461
Fireplace Qu_None   -0.481740
Bsmt Qual_TA        -0.453022
Garage Finish_Unf   -0.422363
                       ...
Garage Cars          0.648488
Total Bsmt SF        0.660983
Gr Liv Area          0.727279
Overall Qual         0.802637
SalePrice            1.000000
Name: SalePrice, Length: 274, dtype: float64
```

```
OverallQual: Rates the overall material and finish of the house

       10   Very Excellent
       9    Excellent
       8    Very Good
       7    Good
       6    Above Average
       5    Average
       4    Below Average
       3    Fair
       2    Poor
       1    Very Poor
```

Most likely a human realtor rated this "Overall Qual" column, which means it highly likely takes into account many of the other features. It also means that any future house we intend to predict a price for will need this "Overall Qual" feature, which implies that every new house on the market that will be priced with our ML model will still require a human person!

Save Final DF

```
# df.to_csv('D:\\Study\\final_data.csv')
```