The Very Famous House Price Prediction Problem:

Problem Statement: With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this competition challenges you to predict the final price of each home.

Linning up my Strategy to Win this Challenge:

1. Understand the problem by knowing your data
2. Univariate data analysis.
3. Multivariate data analysis.
4. Check for missing data.
5. Finding the outliers in the data.
6. Checking Skewness in the data.
7. Convert Categorical data items into numerical data items.
8. Scaling the data
9. Training Model on this data.

Step 1: Understand the problem by knowing the data.

This Step involves having a look at the shape of the data, the content of the data, different type of datatypes present in the data. Also gathering some knowledge about problem by studies articles, blogs, researches on market trends about this problem.

In [117…

```python
#importing useful libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
pd.set_option('display.max_columns', None)
```

In [118…

```python
df_train = pd.read_csv('./train.csv')
df_test = pd.read_csv('./test.csv')
```

In [119…

```python
# Getting names of all the columns in the dataset
df_train.columns
```

Out[119…

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

In [120…

```python
# Printing the first 5 Rows of the data
df_train.head(5)
```

Out[120…

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	C
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	Veenker	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	

In [121…

```python
# Getting information about each Column in the data
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          91 non-null     object
 7   LotShape       1460 non-null   object
 8   LandContour    1460 non-null   object
 9   Utilities      1460 non-null   object
 10  LotConfig      1460 non-null   object
 11  LandSlope      1460 non-null   object
 12  Neighborhood   1460 non-null   object
 13  Condition1     1460 non-null   object
 14  Condition2     1460 non-null   object
 15  BldgType       1460 non-null   object
 16  HouseStyle     1460 non-null   object
 17  OverallQual    1460 non-null   int64
 18  OverallCond    1460 non-null   int64
 19  YearBuilt      1460 non-null   int64
 20  YearRemodAdd   1460 non-null   int64
 21  RoofStyle      1460 non-null   object
 22  RoofMatl       1460 non-null   object
 23  Exterior1st    1460 non-null   object
 24  Exterior2nd    1460 non-null   object
 25  MasVnrType     1452 non-null   object
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object
 28  ExterCond      1460 non-null   object
 29  Foundation     1460 non-null   object
 30  BsmtQual       1423 non-null   object
 31  BsmtCond       1423 non-null   object
 32  BsmtExposure   1422 non-null   object
 33  BsmtFinType1   1423 non-null   object
 34  BsmtFinSF1     1460 non-null   int64
 35  BsmtFinType2   1422 non-null   object
 36  BsmtFinSF2     1460 non-null   int64
 37  BsmtUnfSF      1460 non-null   int64
 38  TotalBsmtSF    1460 non-null   int64
 39  Heating        1460 non-null   object
 40  HeatingQC      1460 non-null   object
 41  CentralAir     1460 non-null   object
 42  Electrical     1459 non-null   object
 43  1stFlrSF       1460 non-null   int64
 44  2ndFlrSF       1460 non-null   int64
 45  LowQualFinSF   1460 non-null   int64
 46  GrLivArea      1460 non-null   int64
 47  BsmtFullBath   1460 non-null   int64
 48  BsmtHalfBath   1460 non-null   int64
 49  FullBath       1460 non-null   int64
 50  HalfBath       1460 non-null   int64
 51  BedroomAbvGr   1460 non-null   int64
 52  KitchenAbvGr   1460 non-null   int64
 53  KitchenQual    1460 non-null   object
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

In [122…

```python
# Getting a statistical picture of the data items in terms of count, mean, standard deviation, distribution of
df_train.describe()
```

Out[122…

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1452.000000	1460.000000
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.267808	1984.865753	103.685262	443.63972
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	30.202904	20.645407	181.066207	456.09809
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000	0.00000
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.000000	1967.000000	0.000000	0.00000
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.000000	1994.000000	0.000000	383.50000
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.000000	2004.000000	166.000000	712.25000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.00000

In [123…

```python
df_train.shape
```

Out[123…

```
(1460, 81)
```

In [124…

```python
# Numerical Features
num_feat=df_train.select_dtypes(exclude=['object']).columns
print(num_feat)
```

```
Index(['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual',
       'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1',
       'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd',
       'Fireplaces', 'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF',
       'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea',
       'MiscVal', 'MoSold', 'YrSold', 'SalePrice'],
      dtype='object')
```

In [125…

```python
# Categorical Features
cat_feat=df_train.select_dtypes(include=['object']).columns
print(cat_feat)
```

```
Index(['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities',
       'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
       'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st',
       'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation',
       'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
       'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual',
       'Functional', 'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageQual',
       'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature',
       'SaleType', 'SaleCondition'],
      dtype='object')
```

Step 2: Univariate data analysis.

In [126…

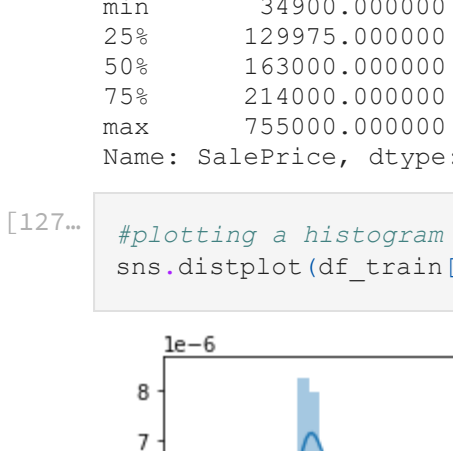
```python
# For the variable Saleprice, understanding its values using different statistical terms like
# mean, count, standard deviation
df_train['SalePrice'].describe()
```

Out[126…

```
count      1460.000000
mean     180921.195890
std       79442.502883
min       34900.000000
25%      129975.000000
50%      163000.000000
75%      214000.000000
max      755000.000000
Name: SalePrice, dtype: float64
```

In [127…

```python
#plotting a histogram to check how the different values of saleprice are distributed
sns.distplot(df_train['SalePrice']);
```



Observations drawn looking at the Histogram

- Deviate from the normal distribution.
- Have appreciable positive skewness.
- Show peakedness.

In [128…

```python
#skewness and kurtosis
print("Skewness: %f" % df_train['SalePrice'].skew())
print("Kurtosis: %f" % df_train['SalePrice'].kurt())
```

```
Skewness: 1.882876
Kurtosis: 6.536282
```

3. Bivariate Data Analysis

Analysing all Against Saleprice (target)

Part 1: For Numerical Features

In [129…

```python
# Filtering out the numerical features in a different data frame
df_train_num=df_train[num_feat]
df_train_num.head(5)
```

Out[129…

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	BsmtUn
0	1	60	65.0	8450	7	5	2003	2003	196.0	706	0	
1	2	20	80.0	9600	6	8	1976	1976	0.0	978	0	
2	3	60	68.0	11250	7	5	2001	2002	162.0	486	0	
3	4	70	60.0	9550	7	5	1915	1970	0.0	216	0	
4	5	60	84.0	14260	8	5	2000	2000	350.0	655	0	

Correlation: Finding How closely one variable relate with another i.e. How strongly the value of the variable increase or decrease with change in value of another variable

In [130…

```python
# Checking the variables whose absolute correlation with Saleprice is more than 0.5
df_corr=df_train_num.corr()['SalePrice'][:-1]
high_corr_feat=df_corr[abs(df_corr)>0.5].sort_values(ascending=False)
print("There is {} strongly correlated value with SalePrice:\n{}".format(len(high_corr_feat), high_corr_feat))
```

```
There is 10 strongly correlated value with SalePrice:
OverallQual     0.790982
GrLivArea       0.708624
GarageCars      0.640409
GarageArea      0.623431
TotalBsmtSF     0.613581
1stFlrSF        0.605852
FullBath        0.560664
TotRmsAbvGrd    0.533723
YearBuilt       0.522897
YearRemodAdd    0.507101
Name: SalePrice, dtype: float64
```

In [131…

```python
# Checking the variables whose absolute correlation with Saleprice is less than 0.05
less_corr_feat=df_corr[abs(df_corr)<0.05].sort_values(ascending=True)
print("There is {} loosely correlated value with SalePrice:\n{}".format(len(less_corr_feat), less_corr_feat))
```

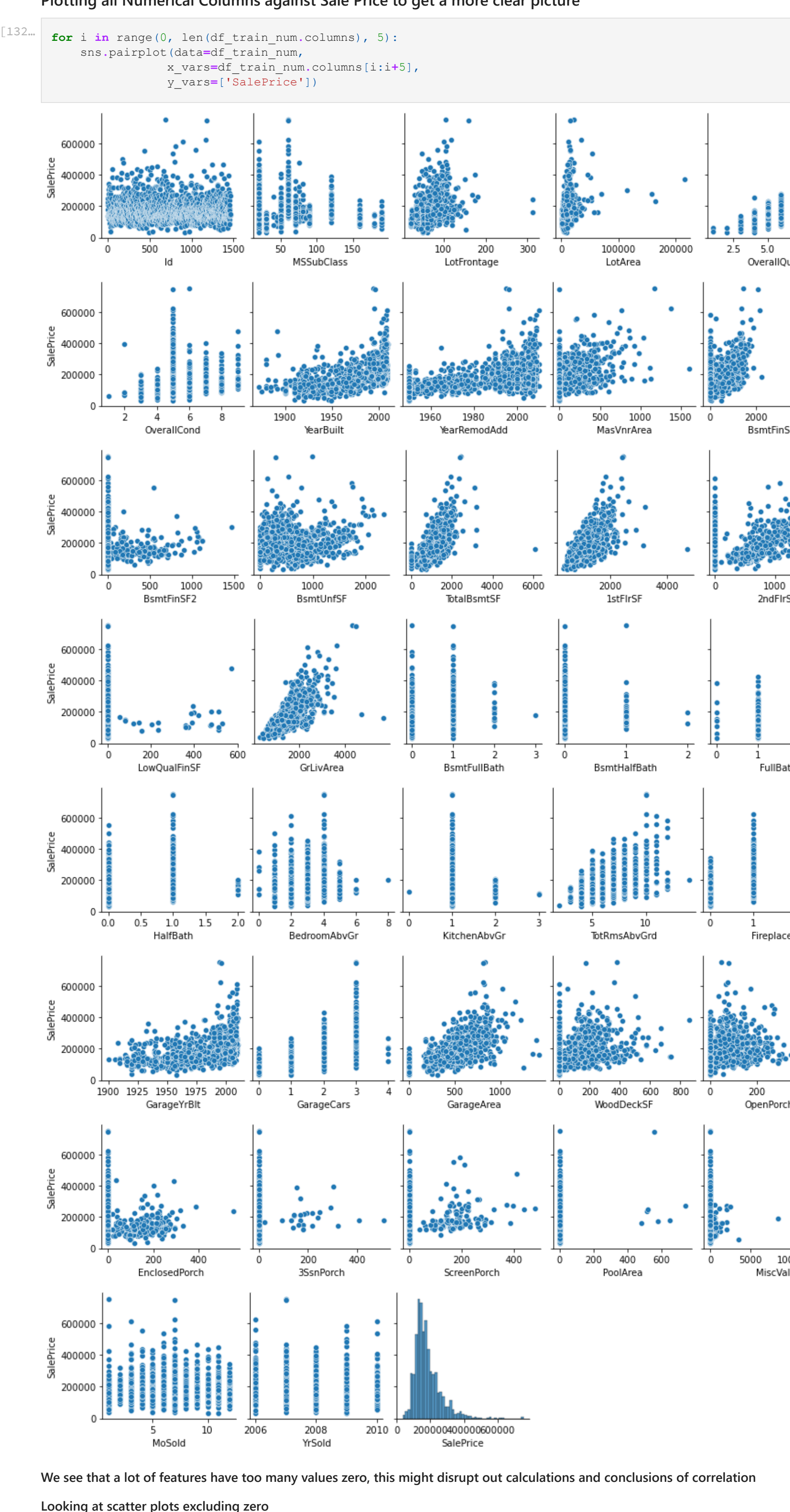
```
There is 8 loosely correlated value with SalePrice:
YrSold          -0.028923
LowQualFinSF    -0.025606
Id              -0.021917
MiscVal         -0.021190
BsmtHalfBath    -0.016844
BsmtFinSF2      -0.011378
3SsnPorch        0.044584
MoSold           0.046432
Name: SalePrice, dtype: float64
```

As Sale price value is almost independent of values of above variables, they might be excluded from analysis for prediction house prices, but let's look further before making any haste

Plotting all Numerical Columns against Sale Price to get a more clear picture

In [132…

```python
for i in range(0, len(df_train_num.columns), 5):
    sns.pairplot(data=df_train_num,
                 x_vars=df_train_num.columns[i:i+5],
                 y_vars=['SalePrice'])
```



We see that a lot of features have too many values zero, this might disrupt out calculations and conclusions of correlation

Looking at scatter plots excluding zero

I find 'OverallCond' is somehow linearly correlated to SalePrice

But 'LowQualFinSF','YrSold'','MiscVal','Id','BsmtHalfBath','BsmtFinSF2' still do not show any relation and are still eligible to be dropped but let's do further analysis

Part 2: For Categorical Features

In [133…

```python
# Getting list of different categories which each categorical feature belong too
for catg in list(cat_feat) :
    print(df_train[catg].value_counts())
    print('#'*50)
```

```
RL         1151
RM          218
FV           65
RH           16
C (all)      10
Name: MSZoning, dtype: int64
##################################################
Pave    1454
Grvl       6
Name: Street, dtype: int64
##################################################
Grvl    50
Pave    41
Name: Alley, dtype: int64
##################################################
Reg    925
IR1    484
IR2     41
IR3     10
Name: LotShape, dtype: int64
##################################################
Lvl    1311
Bnk      63
HLS      50
Low      36
Name: LandContour, dtype: int64
##################################################
AllPub    1459
NoSeWa       1
Name: Utilities, dtype: int64
##################################################
Inside     1052
Corner      263
CulDSac      94
FR2          47
FR3           4
Name: LotConfig, dtype: int64
##################################################
Gtl    1382
Mod      65
Sev      13
Name: LandSlope, dtype: int64
##################################################
NAmes      225
CollgCr    150
OldTown    113
Edwards    100
Somerst     86
Gilbert     79
NridgHt     77
Sawyer      74
NWAmes      73
SawyerW     59
BrkSide     58
Crawfor     51
Mitchel     49
NoRidge     41
Timber      38
IDOTRR      37
ClearCr     28
SWISU       25
StoneBr     25
Blmngtn     17
MeadowV     17
BrDale      16
Veenker     11
NPkVill      9
Blueste      2
Name: Neighborhood, dtype: int64
##################################################
Norm      1260
Feedr       81
Artery      48
RRAn        26
PosN        19
RRAe        11
PosA         8
RRNn         5
RRNe         2
Name: Condition1, dtype: int64
##################################################
Norm      1445
Feedr        6
PosN         2
Artery       2
RRNn         2
PosA         1
RRAn         1
RRAe         1
Name: Condition2, dtype: int64
##################################################
1Fam      1220
TwnhsE     114
Duplex      52
Twnhs       43
2fmCon      31
Name: BldgType, dtype: int64
##################################################
1Story    726
2Story    445
1.5Fin    154
SLvl       65
SFoyer     37
1.5Unf     14
2.5Unf     11
2.5Fin      8
Name: HouseStyle, dtype: int64
##################################################
Gable      1141
Hip         286
Flat         13
Gambrel      11
Mansard       7
Shed          2
Name: RoofStyle, dtype: int64
##################################################
CompShg    1434
Tar&Grv      11
WdShngl       6
WdShake       5
Roll          1
Membran       1
ClyTile       1
Metal         1
Name: RoofMatl, dtype: int64
##################################################
VinylSd    515
HdBoard    222
MetalSd    220
Wd Sdng    206
Plywood    108
CemntBd     61
BrkFace     50
WdShing     26
Stucco      25
AsbShng     20
BrkComm      2
Stone        2
CBlock       1
AsphShn      1
ImStucc      1
Name: Exterior1st, dtype: int64
##################################################
VinylSd    504
MetalSd    214
HdBoard    207
Wd Sdng    197
Plywood    142
CmentBd     60
Wd Shng     38
Stucco      26
BrkFace     25
AsbShng     20
ImStucc     10
Brk Cmn      7
Stone        5
AsphShn      3
CBlock       1
Other        1
Name: Exterior2nd, dtype: int64
##################################################
None       864
BrkFace    445
Stone      128
BrkCmn      15
Name: MasVnrType, dtype: int64
##################################################
TA    906
Gd    488
Ex     52
Fa     14
Name: ExterQual, dtype: int64
##################################################
TA    1282
Gd     146
Fa      28
Ex       3
Po       1
Name: ExterCond, dtype: int64
##################################################
PConc     647
CBlock    634
BrkTil    146
Slab       24
Stone       6
Wood        3
Name: Foundation, dtype: int64
##################################################
TA    649
Gd    618
Ex    121
Fa     35
Name: BsmtQual, dtype: int64
##################################################
TA    1311
Gd      65
Fa      45
Po       2
Name: BsmtCond, dtype: int64
##################################################
No    953
Av    221
Gd    134
Mn    114
Name: BsmtExposure, dtype: int64
##################################################
Unf    430
GLQ    418
ALQ    220
BLQ    148
Rec    133
LwQ     74
Name: BsmtFinType1, dtype: int64
##################################################
Unf    1256
Rec      54
LwQ      46
BLQ      33
ALQ      19
GLQ      14
Name: BsmtFinType2, dtype: int64
##################################################
GasA     1428
GasW       18
Grav        7
Wall        4
OthW        2
Floor       1
Name: Heating, dtype: int64
##################################################
Ex    741
TA    428
Gd    241
Fa     49
Po      1
Name: HeatingQC, dtype: int64
##################################################
Y    1365
N      95
Name: CentralAir, dtype: int64
##################################################
SBrkr    1334
FuseA      94
FuseF      27
FuseP       3
Mix         1
Name: Electrical, dtype: int64
##################################################
TA    735
Gd    586
Ex    100
Fa     39
Name: KitchenQual, dtype: int64
##################################################
Typ     1360
Min2      34
Min1      31
Mod       15
Maj1      14
Maj2       5
Sev        1
Name: Functional, dtype: int64
##################################################
Gd    380
TA    313
Fa     33
Ex     24
Po     20
Name: FireplaceQu, dtype: int64
##################################################
Attchd     870
Detchd     387
BuiltIn     88
Basment     19
CarPort      9
2Types       6
Name: GarageType, dtype: int64
##################################################
Unf    605
RFn    422
Fin    352
Name: GarageFinish, dtype: int64
##################################################
TA    1311
Fa      48
Gd      14
Po       3
Ex       3
Name: GarageQual, dtype: int64
##################################################
TA    1326
Fa      35
Gd       9
Po       7
Ex       2
Name: GarageCond, dtype: int64
##################################################
Y    1340
N      90
P      30
Name: PavedDrive, dtype: int64
##################################################
Gd    3
Fa    2
Ex    2
Name: PoolQC, dtype: int64
##################################################
MnPrv    157
GdPrv     59
GdWo      54
MnWw      11
Name: Fence, dtype: int64
##################################################
Shed    49
Othr     2
Gar2     2
TenC     1
Name: MiscFeature, dtype: int64
##################################################
WD       1267
New       122
COD        43
ConLD       9
ConLw       5
ConLI       5
CWD         4
Oth         3
Con         2
Name: SaleType, dtype: int64
##################################################
Normal     1198
Partial     125
Abnorml     101
Family       20
Alloca       12
AdjLand       4
Name: SaleCondition, dtype: int64
##################################################
```

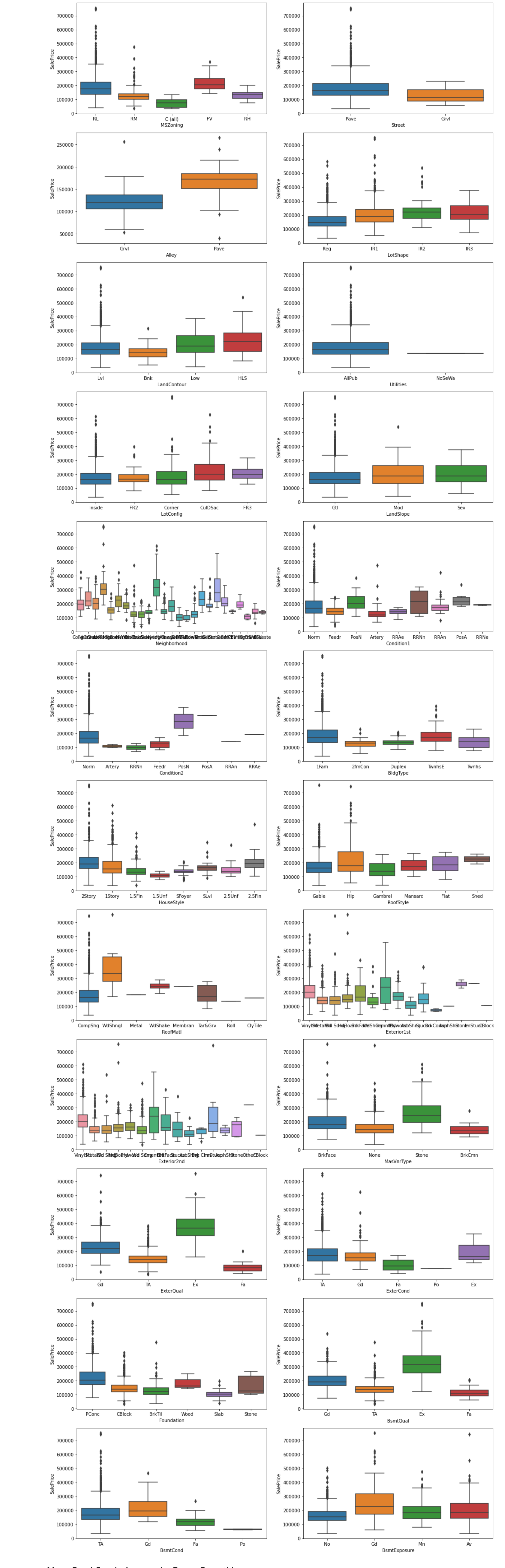
Plotting Box Plots which are Generally Good Visual representation to Observe how categorical values are distributed

In [134…

```python
li_cat_feats = list(cat_feat)
nr_rows = 12
nr_cols = 2

fig, axs = plt.subplots(nr_rows, nr_cols, figsize=(nr_cols*7,nr_rows*4))
for r in range(0,nr_rows):
    for c in range(0,nr_cols):
        i = r*nr_cols+c
        if i < len(li_cat_feats):
            sns.boxplot(x=li_cat_feats[i], y='SalePrice', data=df_train, ax = axs[r][c])

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



Many Good Conclusions can be Drawn From this :

House in Pave Street are Costilier than house in Grvl street.

House with Central air are Costilier than house without central air.

Many Categorical Featues like Electrical, heatingQC, BsmtFinType1 and many more are much helpful in classifying the house to a specific price segment.

Many features have outliers which will require treatment

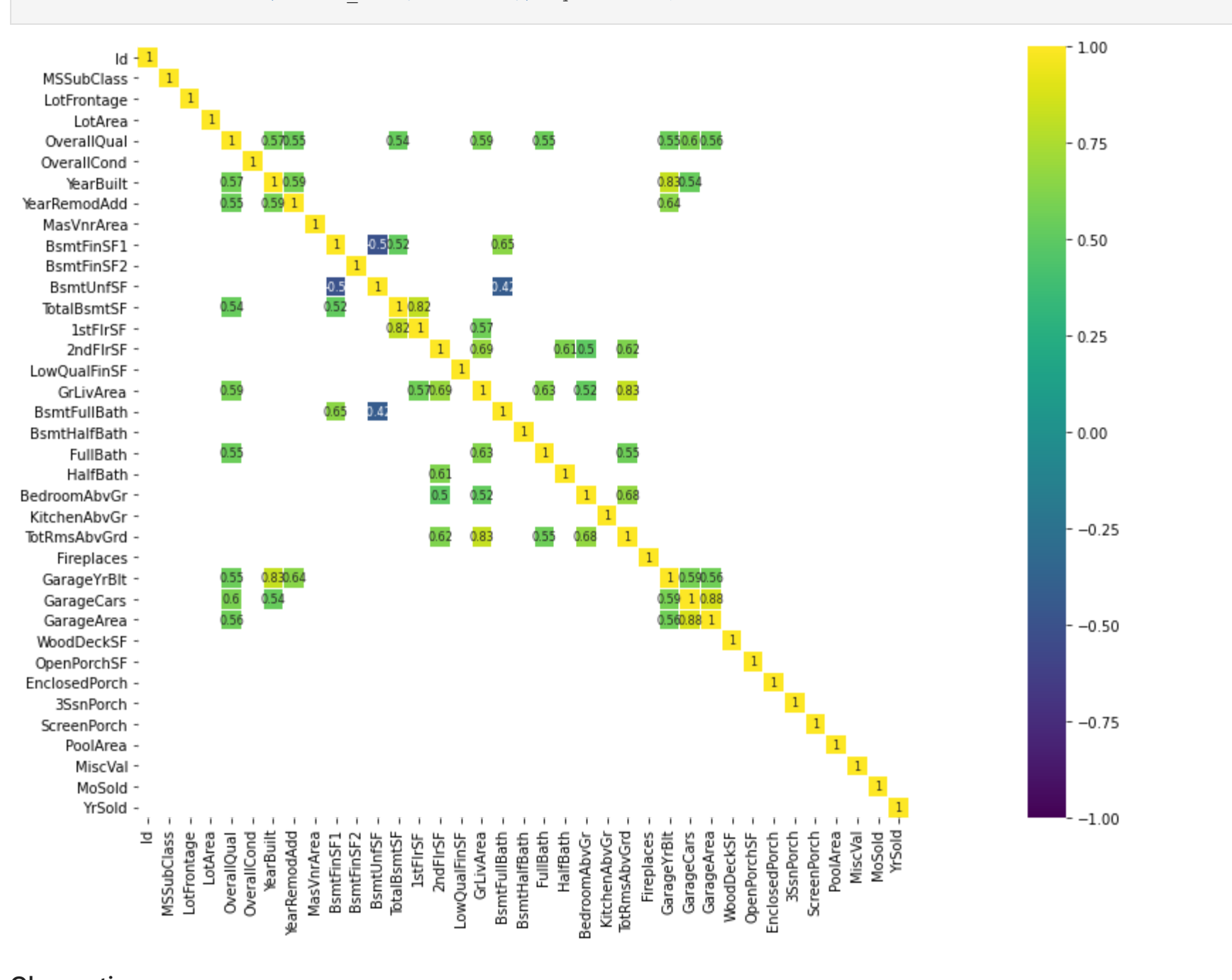
Step 4: Multivariate Data Analysis

This Step involves learning relationships between every pair of variable in dataset

Best place to start with is to draw a Correlation Matrix (heatmap) filtering pairs which have high absolute correlation

In [135…
```
corr = df_train_num.drop('SalePrice', axis=1).corr() # We already examined SalePrice correlations
plt.figure(figsize=(30, 10))

sns.heatmap(corr[(corr >= 0.5) | (corr <= -0.4)],
            cmap='viridis', vmax=1.0, vmin=-1.0, linewidths=0.1,
            annot=True, annot_kws={"size": 8}, square=True);
```



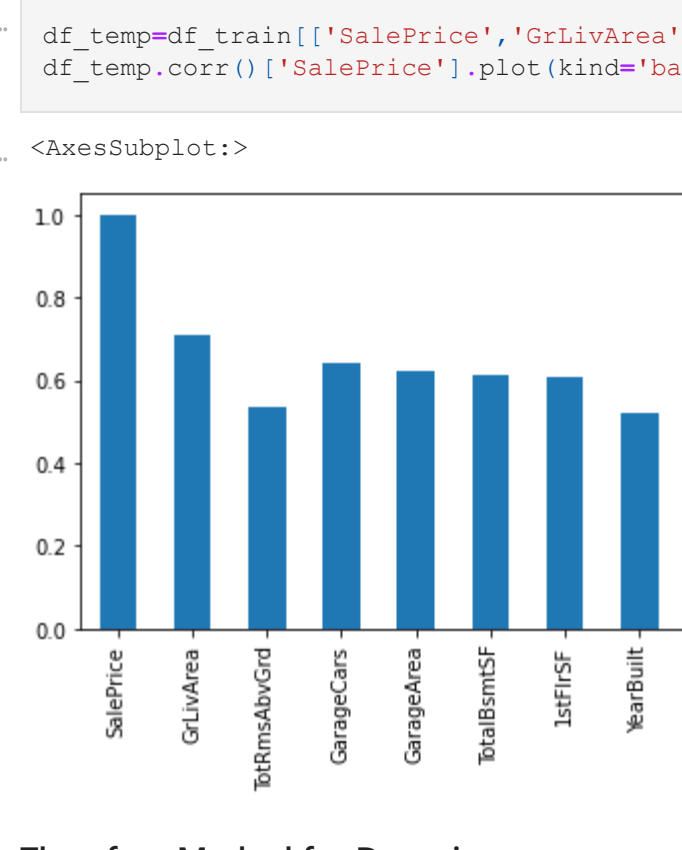
Observations:

Below features have Shown high correlation to one another:

- #### 'GrLivArea' and 'TotRmsAbvGrd' #### 'GarageCars' and 'GarageArea' #### 'TotalBsmtSF' and '1stFlrSF' #### 'YearBuilt' and 'GarageYrBlt' #### Of these similar features we drop the one that has smaller correlation coeffiecient to Target.

In [136…
```
df_temp=df_train[['SalePrice','GrLivArea','TotRmsAbvGrd','GarageCars','GarageArea','TotalBsmtSF','1stFlrSF','Ye
df_temp.corr()['SalePrice'].plot(kind='bar')
```

Out[136… <AxesSubplot:>



Therefore Marked for Dropping are:

GrLivArea, TotRmsAbvGrd, GarageArea, 1stFlrSF, GarageYrBlt

from Before

'LowQualFinSF','YrSold'','MiscVal','Id','BsmtHalfBath','BsmtFinSF2'

Step 5: Missing Data Analysis

In [137…
```
# Finding the Percentage of values missing
total = df_train.isnull().sum().sort_values(ascending=False)
percent = (df_train.isnull().sum()*100/df_train.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
```

Out[137…

	Total	Percent
PoolQC	1453	99.520548
MiscFeature	1406	96.301370
Alley	1369	93.767123
Fence	1179	80.753425
FireplaceQu	690	47.260274
LotFrontage	259	17.739726
GarageYrBlt	81	5.547945
GarageCond	81	5.547945
GarageType	81	5.547945
GarageFinish	81	5.547945
GarageQual	81	5.547945
BsmtFinType2	38	2.602740
BsmtExposure	38	2.602740
BsmtQual	37	2.534247
BsmtCond	37	2.534247
BsmtFinType1	37	2.534247
MasVnrArea	8	0.547945
MasVnrType	8	0.547945
Electrical	1	0.068493
Id	0	0.000000

Filling missing values

For a few columns there is lots of NaN entries.

However, reading the data description we find this is not missing data:

For PoolQC, NaN is not missing data but means no pool, likewise for Fence, FireplaceQu etc

In [138…
```
df_train['SalePrice'] = np.log1p(df_train['SalePrice'])
```

In [148…
```
df_train1=df_train.drop('SalePrice',axis=1)
df_all= pd.concat([df_train1,df_test]).reset_index(drop=True)

df_all.shape
```

Out[148… (2919, 80)

In [140…
```
df_train.shape
```

Out[140… (1460, 81)

In [77]:
```
y_train=df_train['SalePrice']
y_train.shape
```

Out[77]: (1460,)

MSZoning, Utilities, Exteriors, Electrical, Functional, Utilities and SaleType

There are not many missing values, so we will just go with the mode of the neighbourhood.

In [150…
```
feats = ['MSZoning', 'Utilities', 'Exterior1st', 'Exterior2nd', 'Electrical', 'Functional',
         'SaleType']
model = df_train.groupby('Neighborhood')[feats].apply(lambda x: x.mode().iloc[0])

for f in feats:
    df_all[f].fillna(df_all['Neighborhood'].map(model[f]), inplace=True)
```

Lot frontage

My best guess is that it depends somewhat strongly on the configuration of the lot (inside/corner/cul/2-frontage/3-frontage).

In [153…
```
df_all['LotFrontage'] = df_all['LotFrontage'].fillna(df_train['LotFrontage'].median())
```

KitchenQual

Again, very few missing values. We will substitute in this case with the 'OverallQual' value.

In [154…
```
df_all['KitchenQual'].fillna(df_all['OverallQual'], inplace=True)
```

Basement, garage, fireplaces and other features

We can interpret an NA in all these things as "the house does not have this feature".

In [155…
```
bsmt = ['BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1',
        'BsmtFinSF1', 'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'BsmtFullBath',
        'BsmtHalfBath',
        'TotalBsmtSF']
fire = ['Fireplaces', 'FireplaceQu']
garage = ['GarageQual', 'GarageCond', 'GarageType', 'GarageFinish', 'GarageCars',
          'GarageArea', 'GarageYrBlt']
masn = ['MasVnrType', 'MasVnrArea']
others = ['Alley', 'Fence', 'PoolQC', 'MiscFeature']

cats = df_all.columns[df_all.dtypes == 'object']
nums = list(set(df_all.columns) - set(cats))

# Be sure the category 'None' is also handled here
df_all['MasVnrType'].replace({'None': np.nan}, inplace=True)

df_all[cats] = df_all[cats].fillna('0')
df_all[nums] = df_all[nums].fillna(0)
```

In [156…
```
df_all.shape
```

Out[156… (2919, 80)

In [78]:
```
df_all['age']=df_all['YrSold']-df_all['YearBuilt']
# Some of the non-numeric predictors are stored as numbers; convert them into strings
#will convert those columns into dummy variables later.
df_all[['MSSubClass']] = df_all[['MSSubClass']].astype(str)
df_all['YrSold'] = df_all['YrSold'].astype(str) #year
df_all['MoSold'] = df_all['MoSold'].astype(str) #month
df_all.shape
```

Out[78]: (2919, 81)

In [79]:
```
df_all['GarageCars'] = df_all['GarageCars'].fillna(df_all.groupby('GarageArea')['GarageCars'].transform('mean')
df_all['GrLivArea'] = df_all['GrLivArea'].fillna(df_all.groupby('TotRmsAbvGrd')['GrLivArea'].transform('mean'))
df_all['TotalBsmtSF'] = df_all['TotalBsmtSF'].fillna(df_all.groupby('1stFlrSF')['TotalBsmtSF'].transform('mean'
df_all['GarageYrBlt'] = df_all['GarageYrBlt'].fillna(df_all.groupby('YearBuilt')['GarageYrBlt'].transform('mear
```

In [80]:
```
#Functional: Home functionality (Assume typical unless deductions are warranted)
df_all['Functional'] = df_all['Functional'].fillna('Typ')
df_all['Electrical'] = df_all['Electrical'].fillna('SBrkr') #Filling with modef
# data description states that NA refers to "No Pool"
df_all["PoolQC"] = df_all["PoolQC"].fillna("None")
# Replacing the missing values with 0, since no garage = no cars in garage inferred from data dictionary
df_all['GarageYrBlt'] = df_all['GarageYrBlt'].fillna(0)
df_all['KitchenQual'] = df_all['KitchenQual'].fillna("TA")
df_all['Exterior1st'] = df_all['Exterior1st'].fillna(df_all['Exterior1st'].mode()[0])
df_all['Exterior2nd'] = df_all['Exterior2nd'].fillna(df_all['Exterior2nd'].mode()[0])
df_all['SaleType'] = df_all['SaleType'].fillna(df_all['SaleType'].mode()[0])
# Replacing the missing values with None inferred from data dictionary
for col in ['GarageType', 'GarageFinish', 'GarageQual', 'GarageCond']:
    df_all[col] = df_all[col].fillna('None')
# Replacing the missing values with None
# NaN values for these categorical basement df_all, means there's no basement
for col in ('BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2'):
    df_all[col] = df_all[col].fillna('None')
#Replacing missing value it with median beacuse of outliers
df_all['LotFrontage'] = df_all.groupby('Neighborhood')['LotFrontage'].transform(lambda x: x.fillna(x.median()))
# Replacing the missing values with None
# We have no particular intuition around how to fill in the rest of the categorical df_all
# So we replace their missing values with None
objects = []
for i in df_all.columns:
    if df_all[i].dtype == object:
        objects.append(i)
df_all.update(df_all[objects].fillna('None'))

numeric_dtypes = [ 'int64','float64']
numerics = []
for i in df_all.columns:
    if df_all[i].dtype in numeric_dtypes:
        numerics.append(i)
df_all.update(df_all[numerics].fillna(0))

df_all['MSZoning'] = df_all.groupby('MSSubClass')['MSZoning'].transform(lambda x: x.fillna(x.mode()[0]))

df_all['LotFrontage'] = df_all['LotFrontage'].transform(lambda x: x.fillna(x.median()))
```

In [81]:
```
df_all.shape
```

Out[81]: (2919, 81)

In [157…
```
#missing data
total = df_all.isnull().sum().sort_values(ascending=False)
percent = (df_all.isnull().sum()*100/df_train.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
```

Out[157…

	Total	Percent
Id	0.0	0.0
MSSubClass	0.0	0.0
GarageType	0.0	0.0
FireplaceQu	0.0	0.0
Fireplaces	0.0	0.0
Functional	0.0	0.0
TotRmsAbvGrd	0.0	0.0
KitchenQual	0.0	0.0
KitchenAbvGr	0.0	0.0
BedroomAbvGr	0.0	0.0
HalfBath	0.0	0.0
FullBath	0.0	0.0
BsmtHalfBath	0.0	0.0
BsmtFullBath	0.0	0.0
GrLivArea	0.0	0.0
LowQualFinSF	0.0	0.0
2ndFlrSF	0.0	0.0
1stFlrSF	0.0	0.0
Electrical	0.0	0.0
GarageYrBlt	0.0	0.0

All Missing Data has been Dealth with

In [158…
```
#Now Dropping the Columns we have Marked for deletion
df_all=df_all.drop(['GrLivArea','TotRmsAbvGrd','GarageArea','1stFlrSF','GarageYrBlt','LowQualFinSF','YrSold','N
```

In [159…
```
df_all.shape
```

Out[159… (2919, 69)

In [160…
```
df_all.columns
```

Out[160…
```
Index(['MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street', 'Alley',
       'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope',
       'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle',
       'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'RoofStyle',
       'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'MasVnrArea',
       'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond',
       'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2',
       'BsmtUnfSF', 'TotalBsmtSF', 'Heating', 'HeatingQC', 'CentralAir',
       'Electrical', '2ndFlrSF', 'BsmtFullBath', 'FullBath', 'HalfBath',
       'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual', 'Functional',
       'Fireplaces', 'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageCars',
       'GarageQual', 'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MoSold', 'SaleType', 'SaleCondition'],
      dtype='object')
```

Adjusting the type of variable First, there are a few features that are not represented with the right type of variable:

'MSSubClass': represented as an integer, when it is just a category label (we will use 'object' for now) 'MoSold': represented as an integer, a month is just a category label out of 12 possibilities (we will use 'object' for now) 'BsmtFullBath', 'BsmtHalfBath': these two represent integers and not floats (or I at least I do not know what a third of half bathroom is) years: a year, in the context of this dataset, is an integer, and not a float 'GarageCars': represented as a float, it is an actual integer (nobody wants to have 0.5 car at home)

In [165…
```
years = ['YearBuilt', 'YearRemodAdd']
df_all['MSSubClass'] = df_all['MSSubClass'].astype('object', copy=False)
df_all['MoSold'] = df_all['MoSold'].astype('object', copy=False)
df_all['BsmtFullBath'] = df_all['BsmtFullBath'].astype('int64', copy=False)
df_all['GarageCars'] = df_all['GarageCars'].astype('int64', copy=False)
df_all[years] = df_all[years].astype('int64', copy=False)
```

Step 6: Finding the Outliers in the data

Outlier handling is done using the mean and the standard deviation. We have taken three features which have highest correlation to the sale price based on which outliers have been removed.

In [86]:
```
outlier_features = ["OverallQual","GarageCars","TotalBsmtSF"]

means,sds = np.mean(df_all[outlier_features]),np.std(df_all[outlier_features])
lower,upper = means - 2.5*sds , means + 2.5*sds
def compare(x):
    count=0
    for col in outlier_features:
        if x[col]>lower[col] and x[col]<upper[col]:
            count=count+1
    if count==len(outlier_features):
        return True
    else:
        return False

index = len(df_train)
y= y_train.array
sub_data = df_all.loc[:index-1]
test_1 = df_all.loc[index:,:]

data_train_new = sub_data.loc[sub_data.apply(lambda x: compare(x),axis=1)]
y= y[sub_data.apply(lambda x: compare(x),axis=1)]

index = len(data_train_new)
df_all = pd.concat([data_train_new,test_1]).reset_index(drop=True)
```

In [170…
```
df_all.shape
```

Out[170… (2919, 319)

Step 7: Finding Skewness in Data

The point here is to test 'SalePrice' in a very lean way. We'll do this paying attention to:

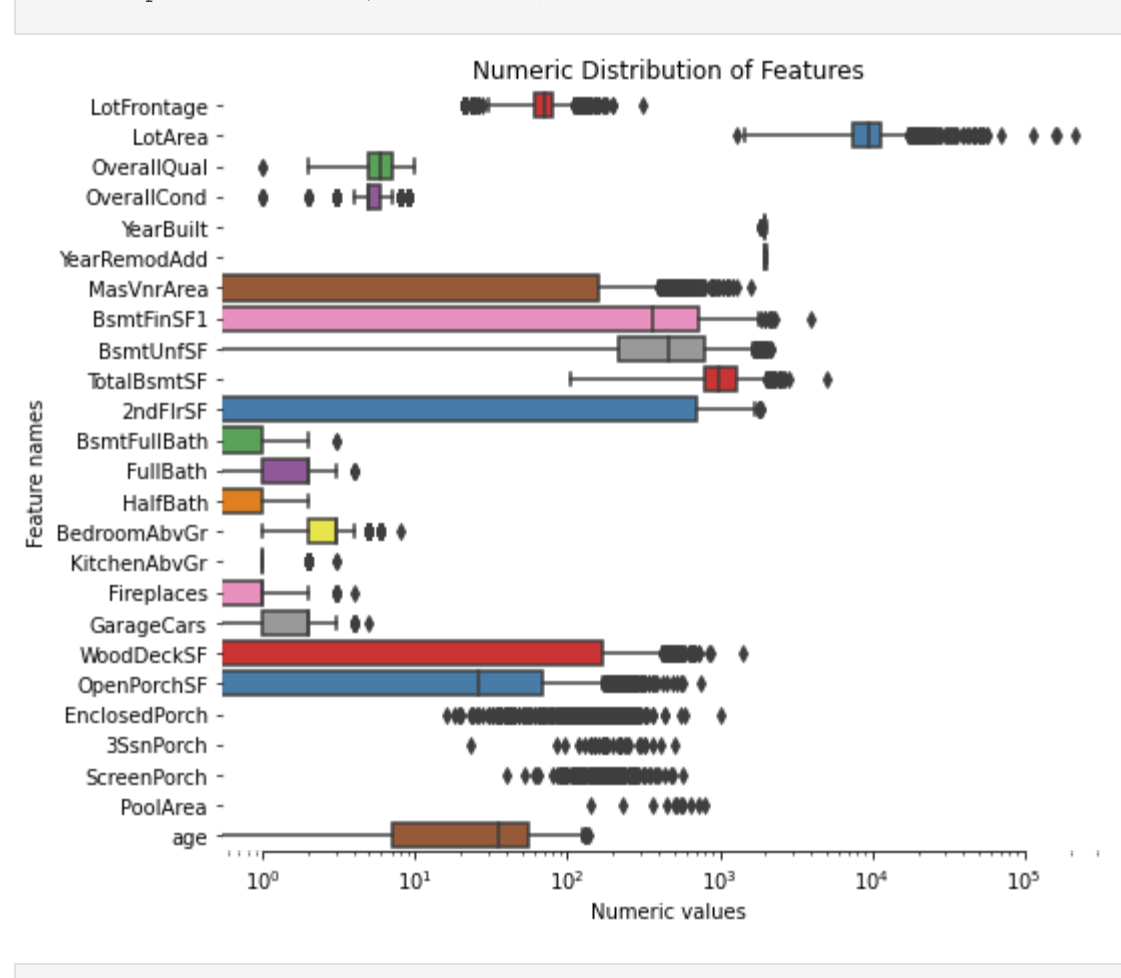
We will draw boxplot - Kurtosis and skewness. Normal probability plot - Data distribution should closely follow the diagonal that represents the normal distribution.

In [87]:
```
from scipy.stats import skew,boxcox_normmax
from scipy.special import boxcox1p
df_all_num = df_all.select_dtypes(include=['int64','float64'])
skew_features = df_all_num.apply(lambda x: skew(x)).sort_values(ascending=False)
high_skew = skew_features[skew_features > 0.5]
skew_index = high_skew.index
skewness = pd.DataFrame({'Skew' :high_skew})
skew_features
```

Out[87]:
```
PoolArea         18.598016
LotArea          13.227511
3SsnPorch        11.304124
KitchenAbvGr      4.308230
EnclosedPorch     4.001455
ScreenPorch       3.931652
MasVnrArea        2.537081
OpenPorchSF       2.508356
WoodDeckSF        1.868117
LotFrontage       1.112298
BsmtFinSF1        0.964203
BsmtUnfSF         0.911018
2ndFlrSF          0.826585
Fireplaces        0.736080
HalfBath          0.699694
BsmtFullBath      0.634100
OverallCond       0.587560
age               0.586691
TotalBsmtSF       0.550503
BedroomAbvGr      0.333155
OverallQual       0.173972
FullBath          0.169433
GarageCars       -0.251691
YearRemodAdd     -0.445471
YearBuilt        -0.588338
dtype: float64
```

Let's visualize skewed variable using boxplots and distplot

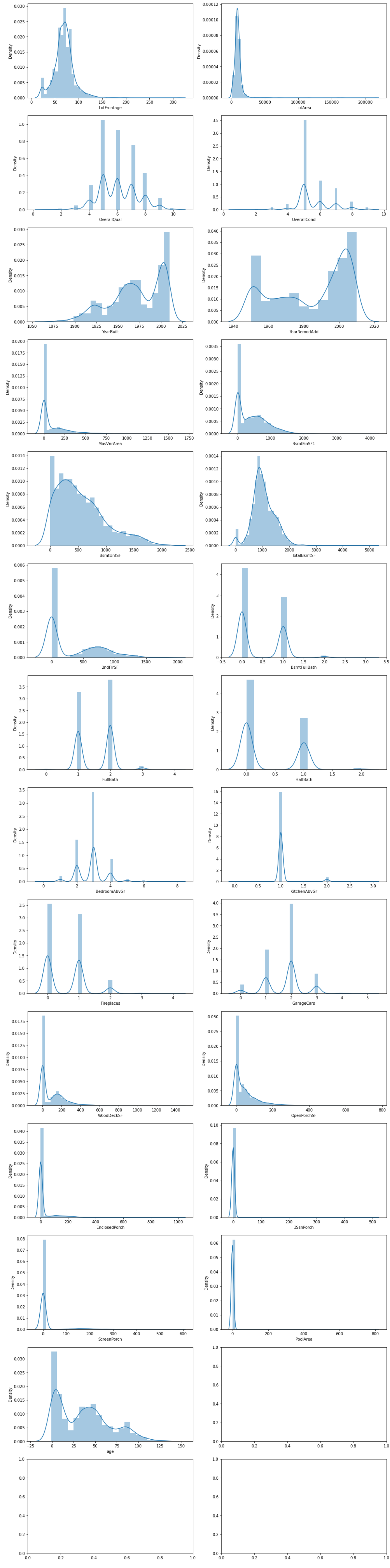
In [88]:
```
f, ax = plt.subplots(figsize=(8, 7))
ax.set_xscale("log")
ax = sns.boxplot(data=df_all_num , orient="h", palette="Set1")
ax.xaxis.grid(False)
ax.set(ylabel="Feature names")
ax.set(xlabel="Numeric values")
ax.set(title="Numeric Distribution of Features")
sns.despine(trim=True, left=True)
```



In [89]:
```
num_feat=df_all.select_dtypes(exclude=['object']).columns
li_num_feats = list(num_feat)
nr_rows = 15
nr_cols = 2

df_all_copy=df_all.copy()

fig, axs = plt.subplots(nr_rows, nr_cols, figsize=(nr_cols*7,nr_rows*4))
for r in range(0,nr_rows):
    for c in range(0,nr_cols):
        i = r*nr_cols+c
        if i < len(li_num_feats):
            sns.distplot(df_all[li_num_feats[i]],ax = axs[r][c]);
plt.tight_layout()
plt.show()
```



In [90]:
```
# Normalize skewed features using a Box-Cox power transformation, we can use other techniques but am using boxp
# as it works very well on this dataset
for i in skew_index:
    df_all[i] = boxcox1p(df_all[i], boxcox_normmax(df_all[i] + 1.002))
```

In [91]:
```
df_all_num = df_all.select_dtypes(include=['int64','float64'])
skew_features = df_all_num.apply(lambda x: skew(x)).sort_values(ascending=False)
high_skew = skew_features[skew_features > 0.5]
skew_index = high_skew.index
skewness = pd.DataFrame({'Skew' :high_skew})
skew_features
```

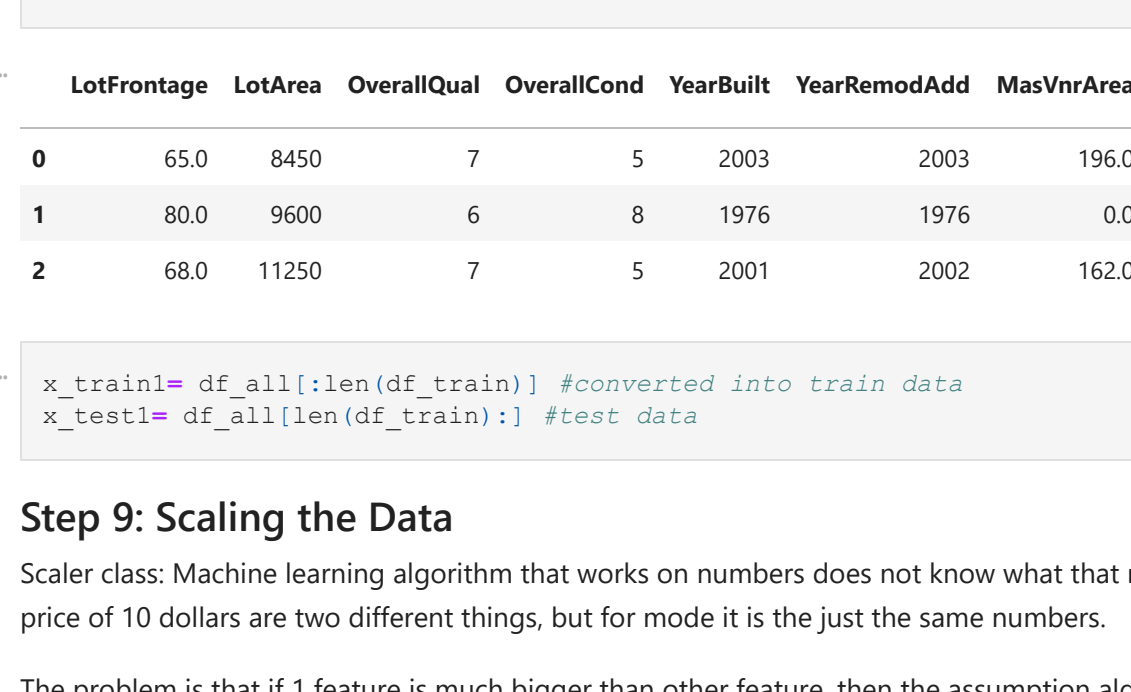
Out[91]:
```
PoolArea         16.199734
3SsnPorch         8.809104
KitchenAbvGr      3.778599
ScreenPorch       3.152066
EnclosedPorch     2.140485
MasVnrArea        0.981665
2ndFlrSF          0.887824
WoodDeckSF        0.789439
HalfBath          0.735962
BsmtFullBath      0.626280
OpenPorchSF       0.621761
Fireplaces        0.560109
OverallCond       0.381814
BsmtFinSF1        0.366564
BedroomAbvGr      0.333155
TotalBsmtSF       0.273947
OverallQual       0.173972
FullBath          0.169433
LotFrontage       0.056948
BsmtUnfSF         0.052920
age               0.008823
LotArea          -0.144092
GarageCars       -0.251691
YearRemodAdd     -0.445471
YearBuilt        -0.588338
dtype: float64
```

Rechecking if skewness is fixed

In [92]:
```
li_num_feats = list(num_feat)
nr_rows = 30
nr_cols = 2

fig, axs = plt.subplots(nr_rows, nr_cols, figsize=(nr_cols*7,nr_rows*4))
for r in range(0,nr_rows):
        i = r
        if i < len(li_num_feats):
            sns.distplot(df_all_copy[li_num_feats[i]],ax = axs[r][0]);
            sns.distplot(df_all[li_num_feats[i]],ax = axs[r][1]);
plt.tight_layout()
plt.show()
```

In [93]:
```
# Let's make sure we handled all the skewed values
f, ax = plt.subplots(figsize=(8, 7))
ax.set_xscale("log")
ax = sns.boxplot(data=df_all[skew_index] , orient="h", palette="Set1")
ax.xaxis.grid(False)
ax.set(ylabel="Feature names")
ax.set(xlabel="Numeric values")
ax.set(title="Numeric Distribution of Features")
sns.despine(trim=True, left=True)
```



Step 8: Converting Categorical Values into Numerical Values

In [169...
```
df_all_num= df_all.select_dtypes(include=['float64','int64']).columns  # Numerical columns
df_all_cat = df_all.select_dtypes(exclude=['float64','int64']) # selecting object and categorical features only
df_all_dummy= pd.get_dummies(df_all_cat)
df_all=pd.concat([df_all,df_all_dummy],axis=1) # joining converted dummy feature and original df_all dataset
df_all= df_all.drop(df_all_cat.columns,axis=1) #removing original categorical columns
df_all.shape
```

Out[169... (2919, 319)

In [171...
```
df_all.head(3)
```

Out[171...

	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtUnfSF	TotalBsmtSF	2ndFlrSF	B
0	65.0	8450	7	5	2003	2003	196.0	706.0	150.0	856.0	854	
1	80.0	9600	6	8	1976	1976	0.0	978.0	284.0	1262.0	0	
2	68.0	11250	7	5	2001	2002	162.0	486.0	434.0	920.0	866	

In [172...
```
x_train1= df_all[:len(df_train)] #converted into train data
x_test1= df_all[len(df_train):] #test data
```

Step 9: Scaling the Data

Scaler class: Machine learning algorithm that works on numbers does not know what that number represents. A weight of 10 grams and a price of 10 dollars are two different things, but for mode it is the just the same numbers.

The problem is that if 1 feature is much bigger than other feature, then the assumption algorithm makes that since first feature is bigger than second one, than it is more important.

We can solve this problem by getting all features in same not big range. Let's say −1≤xi≤1 , where xi is a feature

There are many scalers, but we will be using Min-Max scaler.

xnew=x−xminxmax−xmin

MinMaxScaler scales all the data features in the range [0,1]

In [175...
```
sc = StandardScaler()
x_train = sc.fit_transform(x_train1)
x_test = sc.transform(x_test1)
```

Step 10: Training Models on this data.

A.LinearRegression

In [223...
```
lr = LinearRegression().fit(x_train, y_train)
```

In [218...
```
r_sq = lr.score(x_train, y_train)
print('coefficient of determination - rsquare - on training data :', r_sq)
```

```
coefficient of determination - rsquare - on training data : 0.9558501439818528
```

In [219...
```
maxcoef = np.argsort(-np.abs(Rr.coef_))
coef = lr.coef_[maxcoef]
for i in range(0, 5):
    print("{:.<025} {:< 010.4e}".format(df_all.columns[maxcoef[i]], coef[i]))
```

```
TotalBsmtSF.............  1.5517e-01
RoofMatl_ClyTile........ -2.1815e+10
2ndFlrSF................  9.8110e-02
OverallQual.............  6.0376e-02
GarageCars..............  2.8345e-02
```

In [220...
```
from sklearn.linear_model import LinearRegression, RidgeCV, LassoCV, ElasticNetCV
from sklearn.model_selection import cross_val_score
```

B. Linear regression, L1 regularisation¶

In [221...
```
# Create linear regression object
Ls = LassoCV()

# Train the model using the training sets
Ls.fit(x_train, y_train)
```

Out[221... LassoCV()

In [224…
```
r_sq = Ls.score(x_train, y_train)
print('coefficient of determination - rsquare - on training data :', r_sq)
```

```
coefficient of determination - rsquare - on training data : 0.9339325943077308
```

In [225…
```
maxcoef = np.argsort(-np.abs(Ls.coef_))
coef = Ls.coef_[maxcoef]
for i in range(0, 5):
    print("{:.<025} {:< 010.4e}".format(df_all.columns[maxcoef[i]], coef[i]))
```

```
OverallQual.............  9.7745e-02
TotalBsmtSF.............  9.3288e-02
2ndFlrSF................  5.7333e-02
Condition2_PosN......... -4.8166e-02
GarageCars..............  4.2297e-02
```

C. Linear regression, L2 regularisation¶

In [226…
```
# Create linear regression object
Rr = RidgeCV()

# Train the model using the training sets
Rr.fit(x_train, y_train)
```

Out[226…
```
RidgeCV(alphas=array([ 0.1,  1. , 10. ]))
```

In [227…
```
r_sq = Rr.score(x_train, y_train)
print('coefficient of determination - rsquare - on training data :', r_sq)
```

```
coefficient of determination - rsquare - on training data : 0.9548127192732505
```

In [228…
```
maxcoef = np.argsort(-np.abs(Rr.coef_))
coef = Rr.coef_[maxcoef]
for i in range(0, 5):
    print("{:.<025} {:< 010.4e}".format(df_all.columns[maxcoef[i]], coef[i]))
```

```
TotalBsmtSF.............  9.5422e-02
2ndFlrSF................  6.7126e-02
OverallQual.............  5.9949e-02
Condition2_PosN......... -5.2500e-02
OverallCond.............  4.6214e-02
```