house_prices

September 2, 2020

This kernel is going to solve House Prices: Advanced Regression Techniques, a popular competition on Kaggle. This competition's dataset can be downloaded from the following link

Competition Description:

Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this competition's dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence.

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.

1 Import Libraries

First, we import necessary libraries, such as:

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()
```

2 Import The Data

[2]:
```
test = pd.read_csv('/kaggle/input/house-prices-advanced-regression-techniques/
 ↪test.csv')
train = pd.read_csv('/kaggle/input/house-prices-advanced-regression-techniques/
 ↪train.csv')
```

3 Read The Data

[3]:
```
#set display.max_columns to display all columns
pd.set_option('display.max_columns', None)
```

[4]:
```
train.sample(5)
```

[4]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape
187	188	50	RL	60.0	10410	Pave	NaN	Reg
548	549	20	RM	49.0	8235	Pave	NaN	IR1
1301	1302	70	RL	NaN	7500	Pave	NaN	IR1
278	279	20	RL	107.0	14450	Pave	NaN	Reg
1368	1369	120	RM	NaN	4435	Pave	NaN	Reg

	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1
187	Lvl	AllPub	Inside	Gtl	OldTown	Norm
548	HLS	AllPub	Inside	Gtl	OldTown	Feedr
1301	Bnk	AllPub	Inside	Gtl	Crawfor	Norm
278	Lvl	AllPub	Inside	Gtl	NridgHt	Norm
1368	Lvl	AllPub	Inside	Gtl	CollgCr	Norm

	Condition2	BldgType	HouseStyle	OverallQual	OverallCond	YearBuilt
187	Norm	1Fam	1.5Fin	5	7	1916
548	RRNn	1Fam	1Story	5	7	1955
1301	Norm	1Fam	2Story	6	7	1942
278	Norm	1Fam	1Story	9	5	2006
1368	Norm	TwnhsE	1Story	6	5	2003

	YearRemodAdd	RoofStyle	RoofMatl	Exterior1st	Exterior2nd	MasVnrType
187	1987	Gable	CompShg	HdBoard	HdBoard	None
548	1995	Gable	CompShg	MetalSd	MetalSd	None
1301	1950	Gable	CompShg	Wd Sdng	Wd Sdng	None
278	2007	Gable	CompShg	CemntBd	CmentBd	BrkFace
1368	2004	Gable	CompShg	VinylSd	VinylSd	BrkFace

	MasVnrArea	ExterQual	ExterCond	Foundation	BsmtQual	BsmtCond
187	0.0	TA	TA	CBlock	Fa	TA
548	0.0	TA	Gd	CBlock	TA	TA
1301	0.0	TA	TA	CBlock	TA	TA
278	315.0	Ex	TA	PConc	Ex	TA
1368	170.0	Gd	TA	PConc	Gd	TA

	BsmtExposure	BsmtFinType1	BsmtFinSF1	BsmtFinType2	BsmtFinSF2
187	No	Unf	0	Unf	0
548	No	LwQ	180	Rec	645
1301	No	BLQ	547	Unf	0
278	Gd	Unf	0	Unf	0

```
1368          Av          GLQ         685          Unf           0

      BsmtUnfSF  TotalBsmtSF Heating HeatingQC CentralAir Electrical  \
187         660          660    GasA        Ex          Y      SBrkr
548           0          825    GasA        TA          Y      SBrkr
1301        224          771    GasA        Fa          Y      SBrkr
278        2121         2121    GasA        Ex          Y      SBrkr
1368        163          848    GasA        Ex          Y      SBrkr

      1stFlrSF  2ndFlrSF  LowQualFinSF  GrLivArea  BsmtFullBath  BsmtHalfBath  \
187        808       704           144       1656             0             0
548        825         0             0        825             1             0
1301       753       741             0       1494             0             0
278       2121         0             0       2121             0             0
1368       848         0             0        848             1             0

      FullBath  HalfBath  BedroomAbvGr  KitchenAbvGr KitchenQual  \
187          2         1             3             1          TA
548          1         0             2             1          TA
1301         1         0             3             1          Gd
278          2         1             3             1          Ex
1368         1         0             1             1          Gd

      TotRmsAbvGrd Functional  Fireplaces FireplaceQu GarageType  GarageYrBlt  \
187              8       Min2           0         NaN     Detchd       1916.0
548              4        Typ           0         NaN     Detchd       1963.0
1301             7        Typ           2          Gd     Attchd       1942.0
278              8        Typ           1          Ex     Attchd       2007.0
1368             4        Typ           0         NaN     Attchd       2003.0

     GarageFinish  GarageCars  GarageArea GarageQual GarageCond PavedDrive  \
187           Unf           1         180         Fa         Fa          N
548           RFn           2         720         TA         TA          Y
1301          Unf           1         213         TA         TA          P
278           Fin           3         732         TA         TA          Y
1368          Fin           2         420         TA         TA          Y

      WoodDeckSF  OpenPorchSF  EnclosedPorch  3SsnPorch  ScreenPorch  \
187            0            0              0        140            0
548          140           50              0          0            0
1301           0            0              0          0          224
278          124           98              0          0          142
1368         140            0              0          0            0

      PoolArea PoolQC  Fence MiscFeature  MiscVal  MoSold  YrSold SaleType  \
187          0    NaN  MnPrv         NaN        0       8    2009       WD
548          0    NaN  MnPrv         NaN        0       6    2008       WD
```

```
1301          0    NaN    NaN         NaN        0      11    2009       WD
278           0    NaN    NaN         NaN        0       5    2007      New
1368          0    NaN    NaN         NaN        0       6    2009       WD

     SaleCondition  SalePrice
187         Normal     135000
548         Normal     125000
1301        Normal     177500
278        Partial     415298
1368        Normal     144000
```

Now, let's take a quick look at the train and test datasets to gain some initial insight.

[12]:
```
combined_data = pd.concat([train, test], ignore_index=True)

combined_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2919 entries, 0 to 2918
Data columns (total 81 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Id              2919 non-null   int64
 1   MSSubClass      2919 non-null   int64
 2   MSZoning        2915 non-null   object
 3   LotFrontage     2433 non-null   float64
 4   LotArea         2919 non-null   int64
 5   Street          2919 non-null   object
 6   Alley           198 non-null    object
 7   LotShape        2919 non-null   object
 8   LandContour     2919 non-null   object
 9   Utilities       2917 non-null   object
 10  LotConfig       2919 non-null   object
 11  LandSlope       2919 non-null   object
 12  Neighborhood    2919 non-null   object
 13  Condition1      2919 non-null   object
 14  Condition2      2919 non-null   object
 15  BldgType        2919 non-null   object
 16  HouseStyle      2919 non-null   object
 17  OverallQual     2919 non-null   int64
 18  OverallCond     2919 non-null   int64
 19  YearBuilt       2919 non-null   int64
 20  YearRemodAdd    2919 non-null   int64
 21  RoofStyle       2919 non-null   object
 22  RoofMatl        2919 non-null   object
 23  Exterior1st     2918 non-null   object
 24  Exterior2nd     2918 non-null   object
 25  MasVnrType      2895 non-null   object
```

```
26  MasVnrArea     2896 non-null   float64
27  ExterQual      2919 non-null   object
28  ExterCond      2919 non-null   object
29  Foundation     2919 non-null   object
30  BsmtQual       2838 non-null   object
31  BsmtCond       2837 non-null   object
32  BsmtExposure   2837 non-null   object
33  BsmtFinType1   2840 non-null   object
34  BsmtFinSF1     2918 non-null   float64
35  BsmtFinType2   2839 non-null   object
36  BsmtFinSF2     2918 non-null   float64
37  BsmtUnfSF      2918 non-null   float64
38  TotalBsmtSF    2918 non-null   float64
39  Heating        2919 non-null   object
40  HeatingQC      2919 non-null   object
41  CentralAir     2919 non-null   object
42  Electrical     2918 non-null   object
43  1stFlrSF       2919 non-null   int64
44  2ndFlrSF       2919 non-null   int64
45  LowQualFinSF   2919 non-null   int64
46  GrLivArea      2919 non-null   int64
47  BsmtFullBath   2917 non-null   float64
48  BsmtHalfBath   2917 non-null   float64
49  FullBath       2919 non-null   int64
50  HalfBath       2919 non-null   int64
51  BedroomAbvGr   2919 non-null   int64
52  KitchenAbvGr   2919 non-null   int64
53  KitchenQual    2918 non-null   object
54  TotRmsAbvGrd   2919 non-null   int64
55  Functional     2917 non-null   object
56  Fireplaces     2919 non-null   int64
57  FireplaceQu    1499 non-null   object
58  GarageType     2762 non-null   object
59  GarageYrBlt    2760 non-null   float64
60  GarageFinish   2760 non-null   object
61  GarageCars     2918 non-null   float64
62  GarageArea     2918 non-null   float64
63  GarageQual     2760 non-null   object
64  GarageCond     2760 non-null   object
65  PavedDrive     2919 non-null   object
66  WoodDeckSF     2919 non-null   int64
67  OpenPorchSF    2919 non-null   int64
68  EnclosedPorch  2919 non-null   int64
69  3SsnPorch      2919 non-null   int64
70  ScreenPorch    2919 non-null   int64
71  PoolArea       2919 non-null   int64
72  PoolQC         10 non-null     object
73  Fence          571 non-null    object
```

```
 74  MiscFeature    105 non-null    object
 75  MiscVal        2919 non-null   int64
 76  MoSold         2919 non-null   int64
 77  YrSold         2919 non-null   int64
 78  SaleType       2918 non-null   object
 79  SaleCondition  2919 non-null   object
 80  SalePrice      1460 non-null   float64
dtypes: float64(12), int64(26), object(43)
memory usage: 1.8+ MB
```

[13]:
```
combined_data.describe(include="all")
```

[13]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	\
count	2919.000000	2919.000000	2915	2433.000000	2919.000000	2919	
unique	NaN	NaN	5	NaN	NaN	2	
top	NaN	NaN	RL	NaN	NaN	Pave	
freq	NaN	NaN	2265	NaN	NaN	2907	
mean	1460.000000	57.137718	NaN	69.305795	10168.114080	NaN	
std	842.787043	42.517628	NaN	23.344905	7886.996359	NaN	
min	1.000000	20.000000	NaN	21.000000	1300.000000	NaN	
25%	730.500000	20.000000	NaN	59.000000	7478.000000	NaN	
50%	1460.000000	50.000000	NaN	68.000000	9453.000000	NaN	
75%	2189.500000	70.000000	NaN	80.000000	11570.000000	NaN	
max	2919.000000	190.000000	NaN	313.000000	215245.000000	NaN	

	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	\
count	198	2919	2919	2917	2919	2919	2919	
unique	2	4	4	2	5	3	25	
top	Grvl	Reg	Lvl	AllPub	Inside	Gtl	NAmes	
freq	120	1859	2622	2916	2133	2778	443	
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

	Condition1	Condition2	BldgType	HouseStyle	OverallQual	OverallCond	\
count	2919	2919	2919	2919	2919.000000	2919.000000	
unique	9	8	5	8	NaN	NaN	
top	Norm	Norm	1Fam	1Story	NaN	NaN	
freq	2511	2889	2425	1471	NaN	NaN	
mean	NaN	NaN	NaN	NaN	6.089072	5.564577	
std	NaN	NaN	NaN	NaN	1.409947	1.113131	
min	NaN	NaN	NaN	NaN	1.000000	1.000000	
25%	NaN	NaN	NaN	NaN	5.000000	5.000000	

50%	NaN	NaN	NaN	NaN	6.000000	5.000000
75%	NaN	NaN	NaN	NaN	7.000000	6.000000
max	NaN	NaN	NaN	NaN	10.000000	9.000000

	YearBuilt	YearRemodAdd	RoofStyle	RoofMatl	Exterior1st	Exterior2nd	\
count	2919.000000	2919.000000	2919	2919	2918	2918	
unique	NaN	NaN	6	8	15	16	
top	NaN	NaN	Gable	CompShg	VinylSd	VinylSd	
freq	NaN	NaN	2310	2876	1025	1014	
mean	1971.312778	1984.264474	NaN	NaN	NaN	NaN	
std	30.291442	20.894344	NaN	NaN	NaN	NaN	
min	1872.000000	1950.000000	NaN	NaN	NaN	NaN	
25%	1953.500000	1965.000000	NaN	NaN	NaN	NaN	
50%	1973.000000	1993.000000	NaN	NaN	NaN	NaN	
75%	2001.000000	2004.000000	NaN	NaN	NaN	NaN	
max	2010.000000	2010.000000	NaN	NaN	NaN	NaN	

	MasVnrType	MasVnrArea	ExterQual	ExterCond	Foundation	BsmtQual	\
count	2895	2896.000000	2919	2919	2919	2838	
unique	4	NaN	4	5	6	4	
top	None	NaN	TA	TA	PConc	TA	
freq	1742	NaN	1798	2538	1308	1283	
mean	NaN	102.201312	NaN	NaN	NaN	NaN	
std	NaN	179.334253	NaN	NaN	NaN	NaN	
min	NaN	0.000000	NaN	NaN	NaN	NaN	
25%	NaN	0.000000	NaN	NaN	NaN	NaN	
50%	NaN	0.000000	NaN	NaN	NaN	NaN	
75%	NaN	164.000000	NaN	NaN	NaN	NaN	
max	NaN	1600.000000	NaN	NaN	NaN	NaN	

	BsmtCond	BsmtExposure	BsmtFinType1	BsmtFinSF1	BsmtFinType2	\
count	2837	2837	2840	2918.000000	2839	
unique	4	4	6	NaN	6	
top	TA	No	Unf	NaN	Unf	
freq	2606	1904	851	NaN	2493	
mean	NaN	NaN	NaN	441.423235	NaN	
std	NaN	NaN	NaN	455.610826	NaN	
min	NaN	NaN	NaN	0.000000	NaN	
25%	NaN	NaN	NaN	0.000000	NaN	
50%	NaN	NaN	NaN	368.500000	NaN	
75%	NaN	NaN	NaN	733.000000	NaN	
max	NaN	NaN	NaN	5644.000000	NaN	

	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	Heating	HeatingQC	CentralAir	\
count	2918.000000	2918.000000	2918.000000	2919	2919	2919	
unique	NaN	NaN	NaN	6	5	2	
top	NaN	NaN	NaN	GasA	Ex	Y	

freq	NaN	NaN	NaN	2874	1493	2723
mean	49.582248	560.772104	1051.777587	NaN	NaN	NaN
std	169.205611	439.543659	440.766258	NaN	NaN	NaN
min	0.000000	0.000000	0.000000	NaN	NaN	NaN
25%	0.000000	220.000000	793.000000	NaN	NaN	NaN
50%	0.000000	467.000000	989.500000	NaN	NaN	NaN
75%	0.000000	805.500000	1302.000000	NaN	NaN	NaN
max	1526.000000	2336.000000	6110.000000	NaN	NaN	NaN

	Electrical	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea	\
count	2918	2919.000000	2919.000000	2919.000000	2919.000000	
unique	5	NaN	NaN	NaN	NaN	
top	SBrkr	NaN	NaN	NaN	NaN	
freq	2671	NaN	NaN	NaN	NaN	
mean	NaN	1159.581706	336.483727	4.694416	1500.759849	
std	NaN	392.362079	428.701456	46.396825	506.051045	
min	NaN	334.000000	0.000000	0.000000	334.000000	
25%	NaN	876.000000	0.000000	0.000000	1126.000000	
50%	NaN	1082.000000	0.000000	0.000000	1444.000000	
75%	NaN	1387.500000	704.000000	0.000000	1743.500000	
max	NaN	5095.000000	2065.000000	1064.000000	5642.000000	

	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath	BedroomAbvGr	\
count	2917.000000	2917.000000	2919.000000	2919.000000	2919.000000	
unique	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	NaN	
mean	0.429894	0.061364	1.568003	0.380267	2.860226	
std	0.524736	0.245687	0.552969	0.502872	0.822693	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	1.000000	0.000000	2.000000	
50%	0.000000	0.000000	2.000000	0.000000	3.000000	
75%	1.000000	0.000000	2.000000	1.000000	3.000000	
max	3.000000	2.000000	4.000000	2.000000	8.000000	

	KitchenAbvGr	KitchenQual	TotRmsAbvGrd	Functional	Fireplaces	\
count	2919.000000	2918	2919.000000	2917	2919.000000	
unique	NaN	4	NaN	7	NaN	
top	NaN	TA	NaN	Typ	NaN	
freq	NaN	1492	NaN	2717	NaN	
mean	1.044536	NaN	6.451524	NaN	0.597122	
std	0.214462	NaN	1.569379	NaN	0.646129	
min	0.000000	NaN	2.000000	NaN	0.000000	
25%	1.000000	NaN	5.000000	NaN	0.000000	
50%	1.000000	NaN	6.000000	NaN	1.000000	
75%	1.000000	NaN	7.000000	NaN	1.000000	
max	3.000000	NaN	15.000000	NaN	4.000000	

	FireplaceQu	GarageType	GarageYrBlt	GarageFinish	GarageCars	\
count	1499	2762	2760.000000	2760	2918.000000	
unique	5	6	NaN	3	NaN	
top	Gd	Attchd	NaN	Unf	NaN	
freq	744	1723	NaN	1230	NaN	
mean	NaN	NaN	1978.113406	NaN	1.766621	
std	NaN	NaN	25.574285	NaN	0.761624	
min	NaN	NaN	1895.000000	NaN	0.000000	
25%	NaN	NaN	1960.000000	NaN	1.000000	
50%	NaN	NaN	1979.000000	NaN	2.000000	
75%	NaN	NaN	2002.000000	NaN	2.000000	
max	NaN	NaN	2207.000000	NaN	5.000000	

	GarageArea	GarageQual	GarageCond	PavedDrive	WoodDeckSF	\
count	2918.000000	2760	2760	2919	2919.000000	
unique	NaN	5	5	3	NaN	
top	NaN	TA	TA	Y	NaN	
freq	NaN	2604	2654	2641	NaN	
mean	472.874572	NaN	NaN	NaN	93.709832	
std	215.394815	NaN	NaN	NaN	126.526589	
min	0.000000	NaN	NaN	NaN	0.000000	
25%	320.000000	NaN	NaN	NaN	0.000000	
50%	480.000000	NaN	NaN	NaN	0.000000	
75%	576.000000	NaN	NaN	NaN	168.000000	
max	1488.000000	NaN	NaN	NaN	1424.000000	

	OpenPorchSF	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	\
count	2919.000000	2919.000000	2919.000000	2919.000000	2919.000000	
unique	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	NaN	
mean	47.486811	23.098321	2.602261	16.062350	2.251799	
std	67.575493	64.244246	25.188169	56.184365	35.663946	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	26.000000	0.000000	0.000000	0.000000	0.000000	
75%	70.000000	0.000000	0.000000	0.000000	0.000000	
max	742.000000	1012.000000	508.000000	576.000000	800.000000	

	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	\
count	10	571	105	2919.000000	2919.000000	2919.000000	
unique	3	4	4	NaN	NaN	NaN	
top	Ex	MnPrv	Shed	NaN	NaN	NaN	
freq	4	329	95	NaN	NaN	NaN	
mean	NaN	NaN	NaN	50.825968	6.213087	2007.792737	
std	NaN	NaN	NaN	567.402211	2.714762	1.314964	

```
min       NaN    NaN          NaN      0.000000     1.000000  2006.000000
25%       NaN    NaN          NaN      0.000000     4.000000  2007.000000
50%       NaN    NaN          NaN      0.000000     6.000000  2008.000000
75%       NaN    NaN          NaN      0.000000     8.000000  2009.000000
max       NaN    NaN          NaN  17000.000000    12.000000  2010.000000

       SaleType SaleCondition      SalePrice
count      2918          2919    1460.000000
unique        9             6            NaN
top          WD        Normal            NaN
freq       2525          2402            NaN
mean        NaN           NaN  180921.195890
std         NaN           NaN   79442.502883
min         NaN           NaN   34900.000000
25%         NaN           NaN  129975.000000
50%         NaN           NaN  163000.000000
75%         NaN           NaN  214000.000000
max         NaN           NaN  755000.000000
```

Check for missing values

[31]:
```
combined_data.isnull().sum().sort_values(ascending=False).head(36)
```

[31]:
```
PoolQC          2909
MiscFeature     2814
Alley           2721
Fence           2348
SalePrice       1459
FireplaceQu     1420
LotFrontage      486
GarageQual       159
GarageYrBlt      159
GarageFinish     159
GarageCond       159
GarageType       157
BsmtExposure      82
BsmtCond          82
BsmtQual          81
BsmtFinType2      80
BsmtFinType1      79
MasVnrType        24
MasVnrArea        23
MSZoning           4
Utilities          2
Functional         2
BsmtFullBath       2
BsmtHalfBath       2
```

```
GarageArea         1
BsmtFinSF2         1
Exterior1st        1
TotalBsmtSF        1
GarageCars         1
BsmtUnfSF          1
Electrical         1
BsmtFinSF1         1
KitchenQual        1
SaleType           1
Exterior2nd        1
Street             0
dtype: int64
```

3.0.1 Quick observations on the combined data

- Total houses: 2919
- Feature that can be dropped from training immediately:
  - **SalePrice:** Target array.
  - **Id**
- There is feature that given in numerical but contain categorical variables. Therefore, we have to converted to categorical variables. These features are:
  - **MSSubClass**
- From data_description.txt, there are None value that mean house did not have that kind of feature. So, we can fill missing values with string, e.g. 'None'. These features are:
  - **Alley**
  - **FireplaceQu**
  - **Fence**
  - **MiscFeature**
  - **MasVnrType:** Need to double check this missing values with the MasVnrArea feature. If NaN value in the MasVnrType feature but the MasVnrArea feature has non-zero value, then we must fill the missing value with type. Otherwise, NaN value means no type and we can fill missing values with string to represents no type.
  - **PoolQC:** Need to double check this missing values with the PoolArea feature. If NaN value in the PoolQC feature but the PoolArea feature has non-zero value, then we must fill the missing value with pool quality. Otherwise, NaN value means no pool and we can fill missing values with string to represents no pool.
- Features that have missing values:
  - **MSZoning:** There are 4 missing values.
  - **LotFrontage:** There are some missing data around 16.6%
  - **Utilities:** There are 2 missing values.
  - **Exterior1st:** There is 1 missing value.
  - **Exterior2nd:** There is 1 missing value. But, since the Exterior2nd feature is optional, we have to investigate further.
  - **MasVnrArea:** There are some missing data around 0.8%
  - **Electrical:** There is 1 missing value.

  - **KitchenQual:** There is 1 missing value.
  - **Functional:** There are 2 missing values. Note from data_description.txt: '(Assume typical unless deductions are warranted)'
  - **SaleType:** There is 1 missing value.
- Features that need more deep analysis:
  - **Bsmt++:** Features that start with Bsmt.
  - **Garage++:** Features that start with Garage.

3.0.2 Correlation of all the train features with target variable

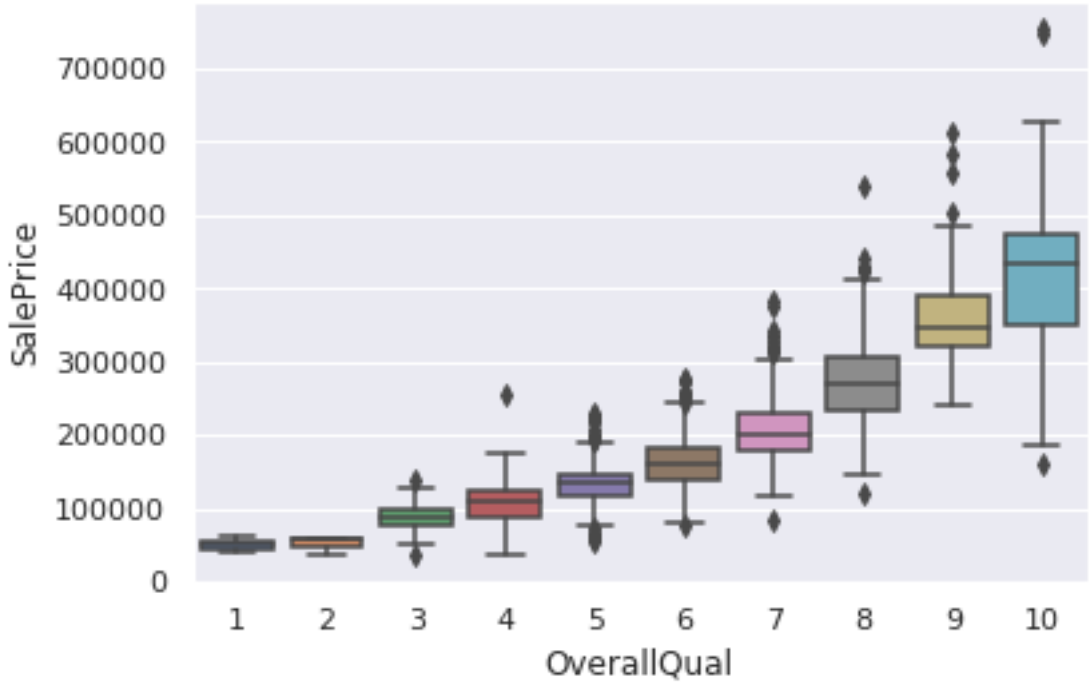
[32]:
```
(train.corr()**2)['SalePrice'].sort_values(ascending = False)[1:]
```

[32]:
```
OverallQual      0.625652
GrLivArea        0.502149
GarageCars       0.410124
GarageArea       0.388667
TotalBsmtSF      0.376481
1stFlrSF         0.367057
FullBath         0.314344
TotRmsAbvGrd     0.284860
YearBuilt        0.273422
YearRemodAdd     0.257151
GarageYrBlt      0.236548
MasVnrArea       0.228000
Fireplaces       0.218023
BsmtFinSF1       0.149320
LotFrontage      0.123763
WoodDeckSF       0.105244
2ndFlrSF         0.101974
OpenPorchSF      0.099765
HalfBath         0.080717
LotArea          0.069613
BsmtFullBath     0.051585
BsmtUnfSF        0.046001
BedroomAbvGr     0.028296
KitchenAbvGr     0.018471
EnclosedPorch    0.016532
ScreenPorch      0.012420
PoolArea         0.008538
MSSubClass       0.007104
OverallCond      0.006062
MoSold           0.002156
3SsnPorch        0.001988
YrSold           0.000837
LowQualFinSF     0.000656
Id               0.000480
```

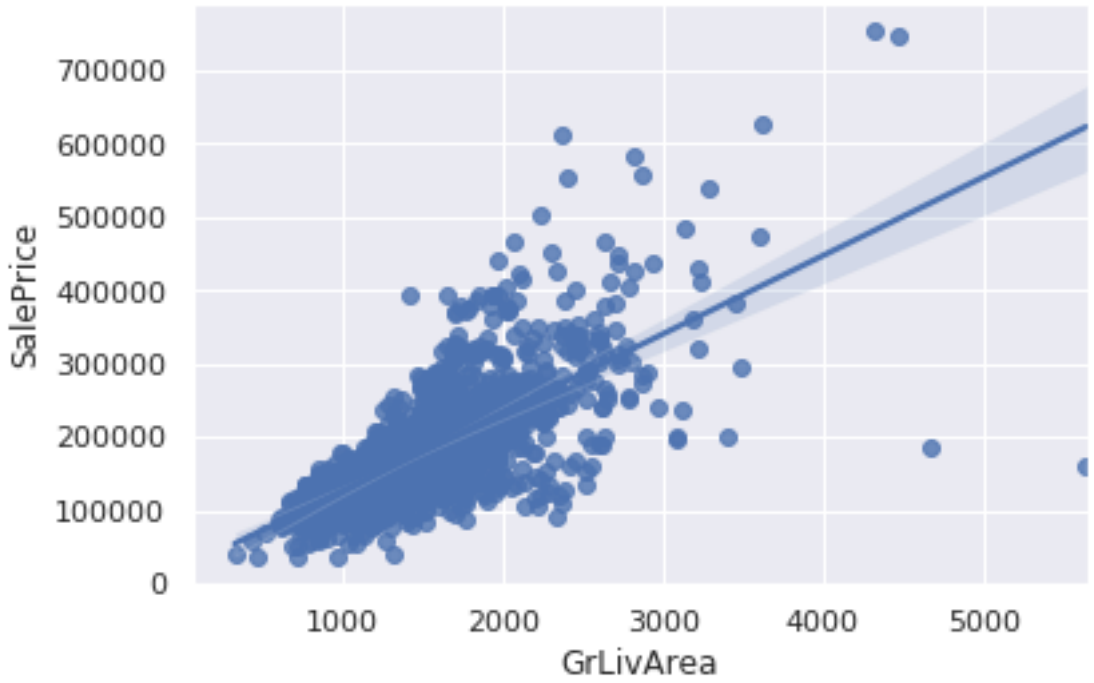
```
MiscVal          0.000449
BsmtHalfBath     0.000284
BsmtFinSF2       0.000129
Name: SalePrice, dtype: float64
```

Plot some top of the most correlated one.

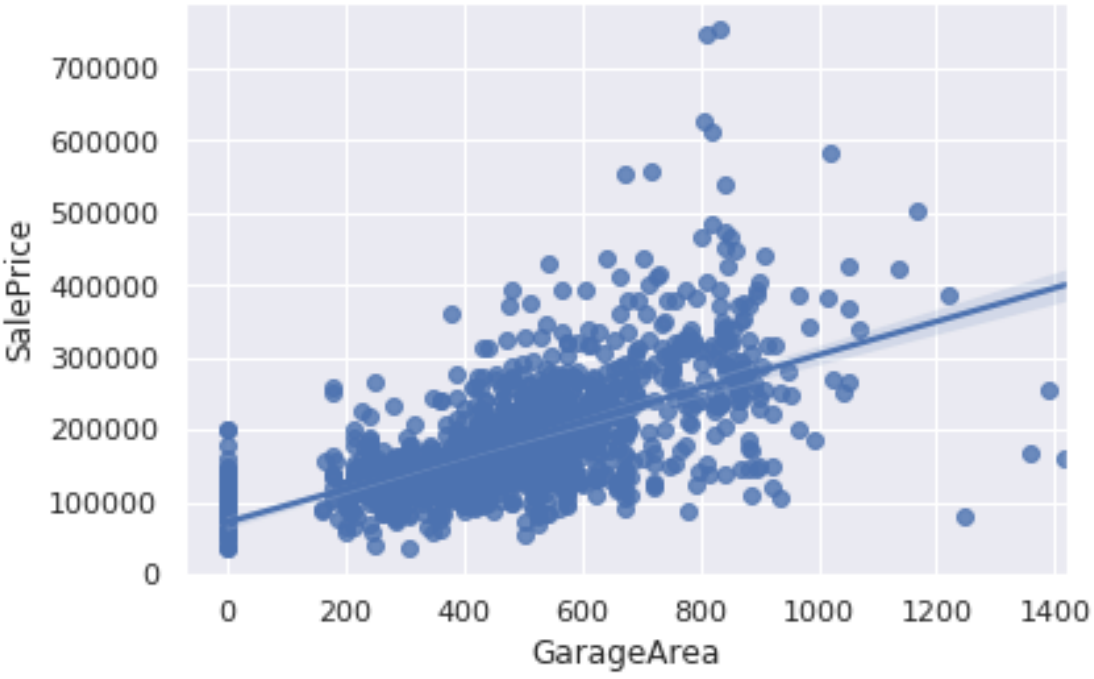
[33]:
```
sns.boxplot(train['OverallQual'], train['SalePrice']);
```



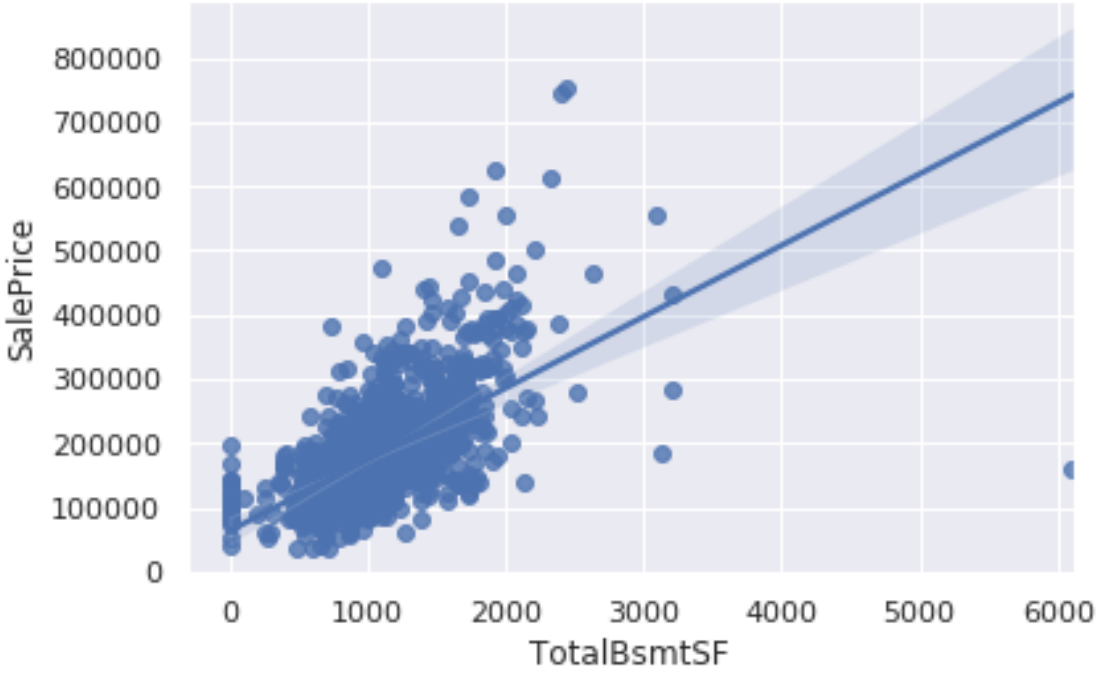
[34]:
```
sns.regplot(train['GrLivArea'], train['SalePrice']);
```



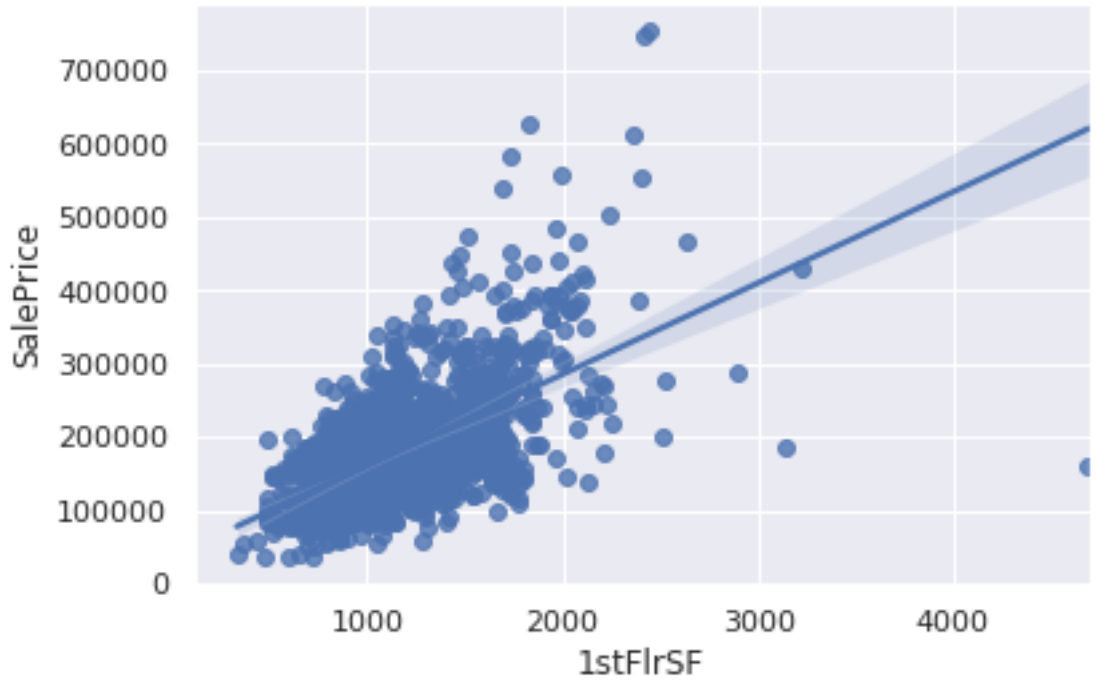
[35]:
```
sns.regplot(train['GarageArea'], train['SalePrice']);
```



[36]:
```
sns.regplot(train['TotalBsmtSF'], train['SalePrice']);
```



[37]:
```
sns.regplot(train['1stFlrSF'], train['SalePrice']);
```



Quick observations on the correlation: * From the plots above, I think the residuals are not pure random fluctuations around the true line, which is bad news. Let's plot residual to make sure.

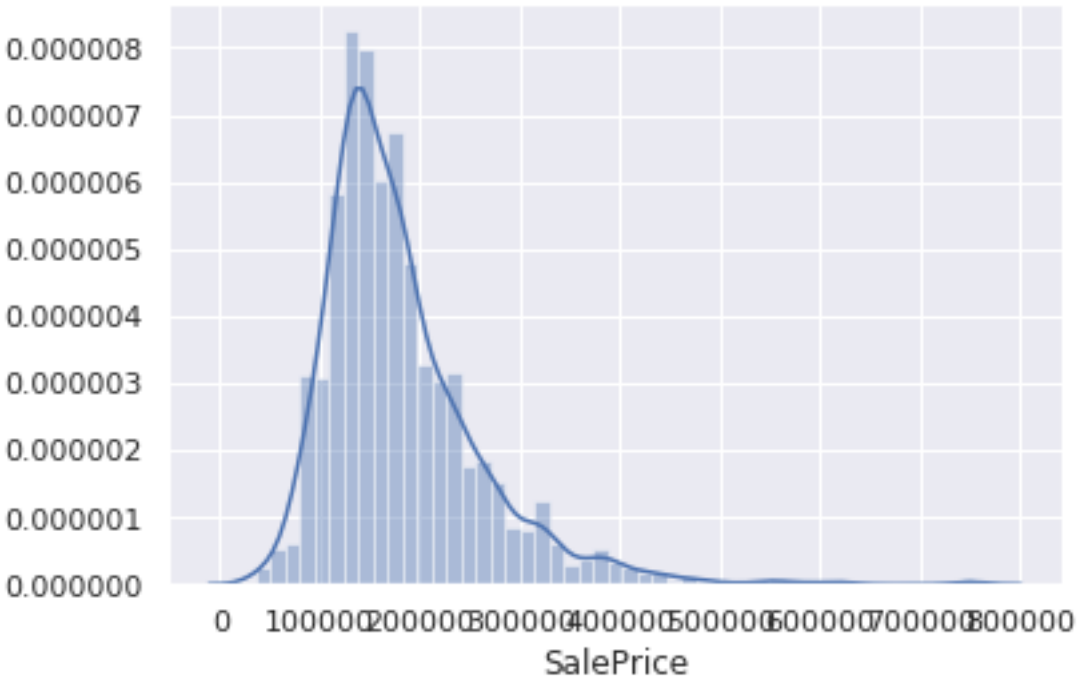
[38]:
```
sns.residplot(train['GrLivArea'], train['SalePrice']);
```

- The residuals plot shows that as GrLivArea value increases, the variance also increases, which is the characteristics known as Heteroscedasticity.
- Get rid of outliers?
- Need to check distirbution of target array and all training features.

3.0.3 Target Array

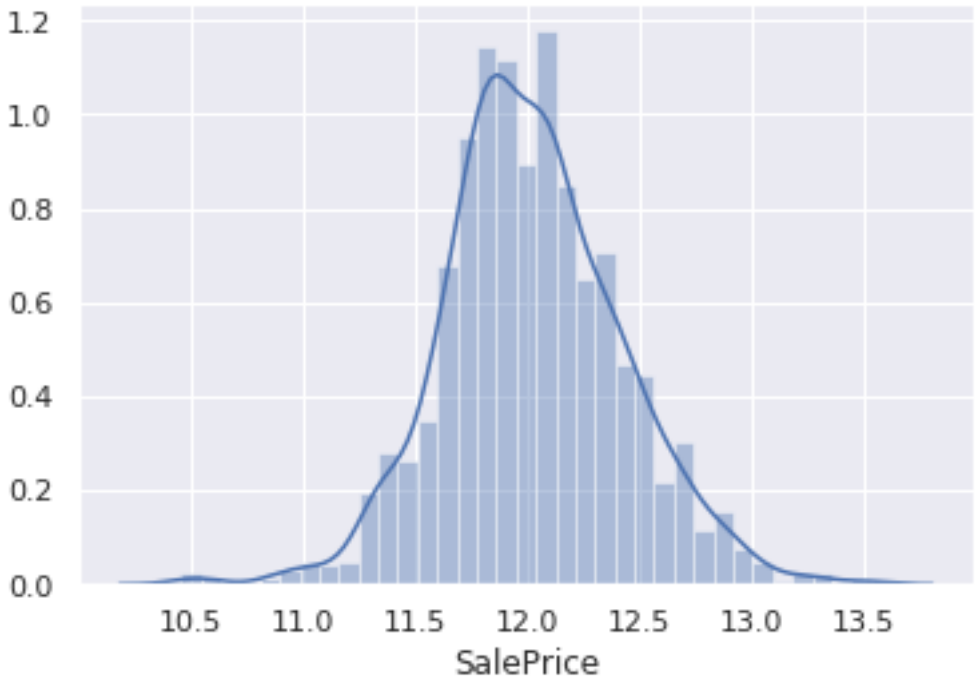
check distribution of target array

[39]:
```
sns.distplot(train['SalePrice']);
```



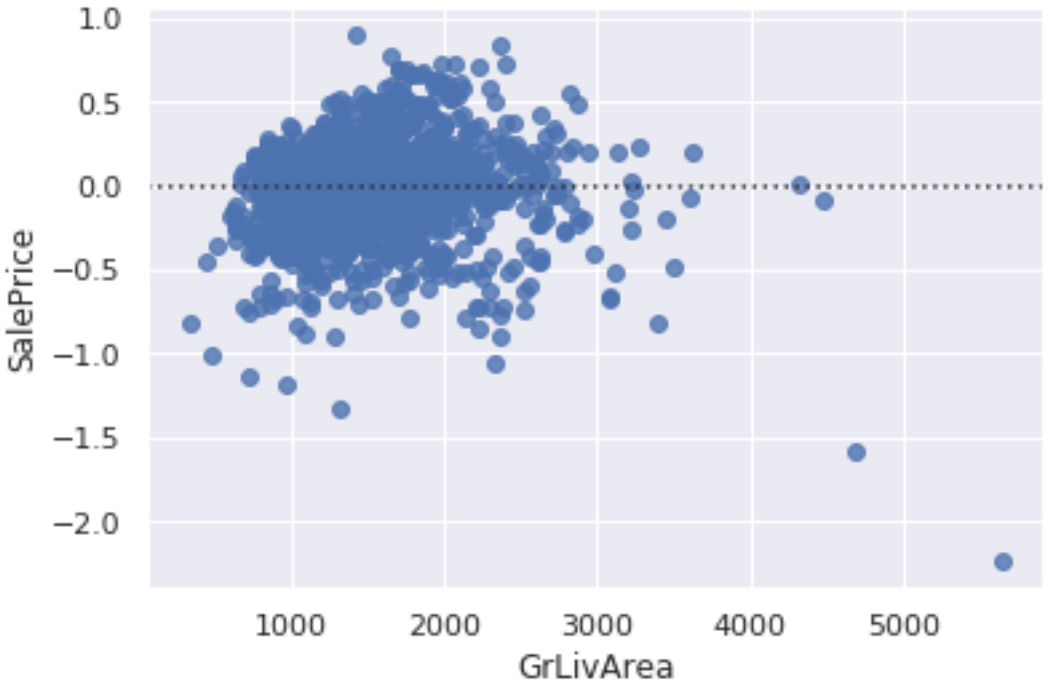
From the graph above we can see that distribution of target array is not following a normal distribution. We can transform it using numpy.log1p, so target array follows a normal distribution.

[40]:
```
#use numpy.log1p in order to target variable follows a normal distribution
train['SalePrice'] = np.log1p(train['SalePrice'])

sns.distplot(train['SalePrice']);
```



After transformed, residual plot become:

[41]:
```
#check residual plot after transformed
sns.residplot(train['GrLivArea'], train['SalePrice']);
```



4 Exploratory Data Analysis

[42]:
```
#check for outliers
#quantile_high = train['SalePrice'].quantile(0.99)
#quantile_low = train['SalePrice'].quantile(0.01)

#outliers_id = train[(train['SalePrice'] > quantile_high) | (train['SalePrice']␣
↪< quantile_low)].index
```

4.0.1 Drop Features

[43]:
```
#copy features that are needed later
target_array = train['SalePrice'].copy()
test_id = test['Id'].copy()

#drop features
train.drop(['Id','SalePrice'], axis=1, inplace=True)
test.drop(['Id'], axis=1, inplace=True)

print(train.shape)
print(test.shape)
```

```
(1460, 79)
(1459, 79)
```

4.0.2 Change to Categorical

[44]:
```
#change to categorical
train['MSSubClass'] = train['MSSubClass'].astype('str')
test['MSSubClass'] = test['MSSubClass'].astype('str')
```

4.0.3 (Some) 'None' Values According to data_description.txt

[45]:
```
features_with_none = ['Alley',
                      'FireplaceQu',
                      'Fence',
                      'MiscFeature']

for feature in features_with_none:
```

```
    train[feature].fillna('None', inplace=True)
    test[feature].fillna('None', inplace=True)
```

4.0.4 Fill Missing Data : With Mode

[46]:
```
#fill missing values with mode
features_fill_with_mode = ['Utilities',
                           'Exterior1st',
                           'Exterior2nd',
                           'Electrical',
                           'KitchenQual',
                           'Functional',
                           'SaleType']

for feature in features_fill_with_mode:
    train[feature].fillna(train[feature].mode()[0], inplace=True)
    test[feature].fillna(test[feature].mode()[0], inplace=True)
```

4.0.5 Fill Missing Data: Special Treatment

- MSZoning Feature

[47]:
```
#fill missing values with mode of each MSSubClass
train['MSZoning'] = train.groupby('MSSubClass')['MSZoning'].apply(lambda df: df.
 ↪fillna(df.mode()[0]))
test['MSZoning'] = test.groupby('MSSubClass')['MSZoning'].apply(lambda df: df.
 ↪fillna(df.mode()[0]))
```

- LotFrontage Feature

[48]:
```
#fill missing values with mean of each Neighborhood
train['LotFrontage'] = train.groupby('Neighborhood')['LotFrontage'].
 ↪apply(lambda df: df.fillna(df.mean()))
test['LotFrontage'] = test.groupby('Neighborhood')['LotFrontage'].apply(lambda␣
 ↪df: df.fillna(df.mean()))
```

- MasVnrType Feature

[49]:
```
#Train dataset
#check if there are missing values in the MasVnrType, but the MasVnrArea != 0
mas_vnr_features = ['MasVnrArea','MasVnrType']

mask = (train['MasVnrArea'] != 0) & \
       train['MasVnrArea'].notnull() & \
       train['MasVnrType'].isnull()
```

```
train[mask][mas_vnr_features]
```

[49]:
```
Empty DataFrame
Columns: [MasVnrArea, MasVnrType]
Index: []
```

[50]:
```
#Test dataset
#check if there are missing values in the MasVnrType, but the MasVnrArea != 0
mask = (test['MasVnrArea'] != 0) & \
       test['MasVnrArea'].notnull() & \
       test['MasVnrType'].isnull()

test[mask][mas_vnr_features]
```

[50]:
```
      MasVnrArea MasVnrType
1150       198.0        NaN
```

[51]:
```
#fill missing data
train['MasVnrType'].fillna('None', inplace=True)

test.loc[mask, 'MasVnrType'] = test['MasVnrType'].mode()[0]
test['MasVnrType'].fillna('None', inplace=True)
```

- MasVnrArea Feature

[52]:
```
#Train dataset
#check if there are missing values in the MasVnrArea, but the MasVnrType !=␣
 ↪'None'
mask = (train['MasVnrType'] != 'None') & \
        train['MasVnrArea'].isnull()

train[mask][mas_vnr_features]
```

[52]:
```
Empty DataFrame
Columns: [MasVnrArea, MasVnrType]
Index: []
```

[53]:
```
#Test dataset
#check if there are missing values in the MasVnrArea, but the MasVnrType !=␣
 ↪'None'
mask = (test['MasVnrType'] != 'None') & \
        test['MasVnrArea'].isnull()

test[mask][mas_vnr_features]
```

[53]:
```
Empty DataFrame
Columns: [MasVnrArea, MasVnrType]
Index: []
```

[54]:
```
#fill missing data with 0
train['MasVnrArea'].fillna(0, inplace=True)
test['MasVnrArea'].fillna(0, inplace=True)
```

- PoolQC Feature

[55]:
```
#Train dataset
#check if there are missing values in the PoolQC, but the PoolArea != 0
pool_features = ['PoolArea','PoolQC']

mask = (train['PoolArea'] != 0) & \
       train['PoolQC'].isnull()

train[mask][pool_features]
```

[55]:
```
Empty DataFrame
Columns: [PoolArea, PoolQC]
Index: []
```

[56]:
```
#Test dataset
#check if there are missing values in the PoolQC, but the PoolArea != 0
mask = (test['PoolArea'] != 0) & \
       test['PoolQC'].isnull()

test[mask][pool_features]
```

[56]:

	PoolArea	PoolQC
960	368	NaN
1043	444	NaN
1139	561	NaN

[57]:
```
#fill missing values
train['PoolQC'].fillna('None', inplace=True)

test.loc[mask, 'PoolQC'] = test['PoolQC'].mode()[0]
test['PoolQC'].fillna('None', inplace=True)
```

4.0.6 Basement Features

- Categorical features

[58]:
```
#Train dataset
#check if there are missing values in the categorical features, but the␣
 ↪TotalBsmtSF != 0
bsmt_features_1 =␣
 ↪['BsmtQual','BsmtFinType1','BsmtFinType2','BsmtCond','BsmtExposure']

mask = (train['BsmtQual'].isnull() | \
        train['BsmtFinType1'].isnull() | \
        train['BsmtFinType2'].isnull() | \
        train['BsmtCond'].isnull() | \
        train['BsmtExposure'].isnull() ) \
        & \
       ((train['TotalBsmtSF'] != 0) & \
        train['TotalBsmtSF'].notnull())

train[mask][np.concatenate([bsmt_features_1,['TotalBsmtSF']])]
```

[58]:

	BsmtQual	BsmtFinType1	BsmtFinType2	BsmtCond	BsmtExposure	TotalBsmtSF
332	Gd	GLQ	NaN	TA	No	3206
948	Gd	Unf	Unf	TA	NaN	936

[59]:
```
#Test dataset
#check if there are missing values in the categorical features, but the␣
 ↪TotalBsmtSF != 0
mask = (test['BsmtQual'].isnull() | \
        test['BsmtFinType1'].isnull() | \
        test['BsmtFinType2'].isnull() | \
        test['BsmtCond'].isnull() | \
        test['BsmtExposure'].isnull() ) \
        & \
       ((test['TotalBsmtSF'] != 0) & \
        test['TotalBsmtSF'].notnull())

test[mask][np.concatenate([bsmt_features_1,['TotalBsmtSF']])]
```

[59]:

	BsmtQual	BsmtFinType1	BsmtFinType2	BsmtCond	BsmtExposure	TotalBsmtSF
27	Gd	Unf	Unf	TA	NaN	1595.0
580	Gd	GLQ	Rec	NaN	Mn	1426.0
725	TA	BLQ	Unf	NaN	No	1127.0
757	NaN	Unf	Unf	Fa	No	173.0
758	NaN	Unf	Unf	TA	No	356.0
888	Gd	Unf	Unf	TA	NaN	725.0
1064	TA	ALQ	Unf	NaN	Av	995.0

[60]:
```
#fill missing data
for feature in bsmt_features_1:
```

```
    train.loc[mask & train[feature].isnull(), feature] = train[feature].
↪mode()[0]
    train[feature].fillna('None', inplace=True)

    test.loc[mask & test[feature].isnull(), feature] = test[feature].mode()[0]
    test[feature].fillna('None', inplace=True)
```

- Numerical feature

[62]:
```
#Train dataset
#check if there are missing values in the numerical features, but the BsmtQual !
↪= 'None'
bsmt_features_2 =␣
↪['BsmtFinSF1','BsmtFinSF2','BsmtUnfSF','TotalBsmtSF','BsmtFullBath','BsmtHalfBath']

mask = (train['BsmtFinSF1'].isnull() | \
        train['BsmtFinSF2'].isnull() | \
        train['BsmtUnfSF'].isnull() | \
        train['TotalBsmtSF'].isnull() | \
        train['BsmtFullBath'].isnull() | \
        train['BsmtHalfBath'].isnull() ) \
        & \
       (train['BsmtQual'] != 'None')

train[mask][np.concatenate([bsmt_features_2,['BsmtQual']])]
```

[62]:
```
Empty DataFrame
Columns: [BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, BsmtFullBath,
BsmtHalfBath, BsmtQual]
Index: []
```

[63]:
```
#Test dataset
#check if there are missing values in the numerical features, but the BsmtQual !
↪= 'None'
mask = (test['BsmtFinSF1'].isnull() | \
        test['BsmtFinSF2'].isnull() | \
        test['BsmtUnfSF'].isnull() | \
        test['TotalBsmtSF'].isnull() | \
        test['BsmtFullBath'].isnull() | \
        test['BsmtHalfBath'].isnull() ) \
        & \
       (test['BsmtQual'] != 'None')

test[mask][np.concatenate([bsmt_features_2,['BsmtQual']])]
```

[63]:
```
Empty DataFrame
Columns: [BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, BsmtFullBath,
```

```
BsmtHalfBath, BsmtQual]
Index: []
```

[64]:
```
#fill missing data with 0
for feature in bsmt_features_2:
    train[feature].fillna(0, inplace=True)
    test[feature].fillna(0, inplace=True)
```

4.0.7 Garage Features

- Categorical feature

[65]:
```
#Train dataset
#check if there are missing values in the categorical features, but the␣
 ↪GarageCars or GarageArea != 0
garage_features_1 = ['GarageType','GarageFinish','GarageQual','GarageCond']

mask = (train['GarageType'].isnull() | \
        train['GarageFinish'].isnull() | \
        train['GarageQual'].isnull() | \
        train['GarageCond'].isnull()) \
        & \
       ((train['GarageCars'] != 0) | \
        (train['GarageArea'] !=0))

train[mask][np.concatenate([garage_features_1,['GarageCars','GarageArea']])]
```

[65]:
```
Empty DataFrame
Columns: [GarageType, GarageFinish, GarageQual, GarageCond, GarageCars,
GarageArea]
Index: []
```

[66]:
```
#Test dataset
#check if there are missing values in the categorical features, but the␣
 ↪GarageCars or GarageArea != 0
garage_features_1 = ['GarageType','GarageFinish','GarageQual','GarageCond']

mask = (test['GarageType'].isnull() | \
        test['GarageFinish'].isnull() | \
        test['GarageQual'].isnull() | \
        test['GarageCond'].isnull()) \
        & \
       ((test['GarageCars'] != 0) | \
        (test['GarageArea'] !=0))

test[mask][np.concatenate([garage_features_1,['GarageCars','GarageArea']])]
```

[66]:
```
     GarageType GarageFinish GarageQual GarageCond  GarageCars  GarageArea
666      Detchd          NaN        NaN        NaN         1.0       360.0
1116     Detchd          NaN        NaN        NaN         NaN         NaN
```

[67]:
```
#fill missing data
for feature in garage_features_1:
    train[feature].fillna('None', inplace=True)

    test.loc[mask & test[feature].isnull(), feature] = test[feature].mode()[0]
    test[feature].fillna('None', inplace=True)
```

- Numerical feature

[68]:
```
#Train dataset
#check if there are missing values in the numerical features, but the␣
 ↪GarageType != 'None'
garage_features_2 = ['GarageYrBlt','GarageCars','GarageArea']

mask = (train['GarageYrBlt'].isnull() | \
        train['GarageCars'].isnull() | \
        train['GarageArea'].isnull() ) \
        & \
       (train['GarageType'] != 'None')

train[mask][np.concatenate([garage_features_2,['GarageType']])]
```

[68]:
```
Empty DataFrame
Columns: [GarageYrBlt, GarageCars, GarageArea, GarageType]
Index: []
```

[69]:
```
#Test dataset
#check if there are missing values in the numerical features, but the␣
 ↪GarageType != 'None'
garage_features_2 = ['GarageYrBlt','GarageCars','GarageArea']

mask = (test['GarageYrBlt'].isnull() | \
        test['GarageCars'].isnull() | \
        test['GarageArea'].isnull() ) \
        & \
       (test['GarageType'] != 'None')

test[mask][np.concatenate([garage_features_2,['GarageType']])]
```

[69]:
```
      GarageYrBlt  GarageCars  GarageArea GarageType
666           NaN         1.0       360.0     Detchd
1116          NaN         NaN         NaN     Detchd
```

– GarageYrBlt

[70]:
```
test.loc[mask, ['GarageYrBlt','YearBuilt']]
```

[70]:

	GarageYrBlt	YearBuilt
666	NaN	1910
1116	NaN	1923

[71]:
```
#fill above missing data as same as YearBuilt
test.loc[mask, 'GarageYrBlt'] = test.loc[mask, 'YearBuilt'].astype('float64')

#fill the rest with 0
test['GarageYrBlt'].fillna(0, inplace=True)
train['GarageYrBlt'].fillna(0, inplace=True)
```

- GarageCars and GarageArea

[72]:
```
#fill missing data with mode for Detchd GarageType
test['GarageCars'].fillna(test.groupby('GarageType')['GarageCars'].agg(pd.
↪Series.mode)['Detchd'], inplace=True)

#fill missing data with ean for Detchd GarageType
test['GarageArea'].fillna(test.groupby('GarageType')['GarageArea'].
↪mean()['Detchd'], inplace=True)
```

4.0.8 Exploratory Data Analysis: Epilogue

- Check for any missing data

[73]:
```
#check for any missing data
print('missing data in the train dataset : ', train.isnull().any().sum())
print('missing data in the test dataset : ', test.isnull().any().sum())
```

```
missing data in the train dataset :  0
missing data in the test dataset :  0
```

[74]:
```
#train_copy = train.copy()
#test_copy = test.copy()
```

- Creating new features

[75]:
```
datasets = [train, test]

for dataset in datasets:
    #dataset['house_age'] = ((dataset['YrSold'] - dataset['YearRemodAdd']) +␣
↪(dataset['MoSold'] / 12)) / 10
    dataset['TotalSF'] = (dataset['TotalBsmtSF'] \
                          + dataset['1stFlrSF'] \
                          + dataset['2ndFlrSF'])
```

```
    dataset['total_bathrooms'] = dataset['BsmtFullBath'] + dataset['FullBath']␣
↪+ \
                                 (0.5 * (dataset['BsmtHalfBath'] + ␣
↪dataset['HalfBath']))

    dataset['Total_porch_sf'] = (dataset['OpenPorchSF'] \
                               + dataset['3SsnPorch'] \
                               + dataset['EnclosedPorch'] \
                               + dataset['ScreenPorch'] \
                               + dataset['WoodDeckSF'])

    dataset['has_been_remod'] = (dataset['YearBuilt'] !=␣
↪dataset['YearRemodAdd'])
    dataset['has_garage'] = dataset['GarageType'] != 'None'
    dataset['has_basement'] = dataset['BsmtQual'] != 'None'
    dataset['has_2ndFloor'] = dataset['2ndFlrSF'] > 0
    dataset['has_fireplace'] = dataset['Fireplaces'] > 0
    dataset['has_pool'] = dataset['PoolArea'] > 0
    dataset['has_fence'] = dataset['Fence'] != 'None'
```

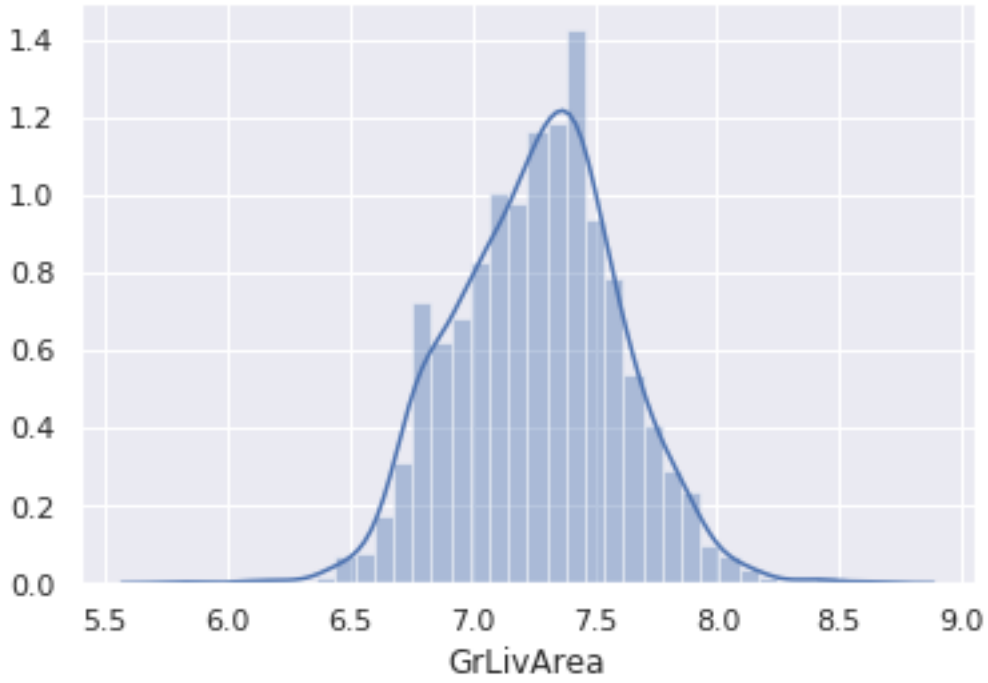
- Normality test

[76]:
```
#define a normality test function
def normalityTest(data, alpha=0.05):
    """data (array)   : The array containing the sample to be tested.
           alpha (float)  : Significance level.
           return True if data is normal distributed"""

    from scipy import stats

    statistic, p_value = stats.normaltest(data)

    #null hypothesis: array comes from a normal distribution
    if p_value < alpha:
        #The null hypothesis can be rejected
        is_normal_dist = False
    else:
        #The null hypothesis cannot be rejected
        is_normal_dist = True

    return is_normal_dist
```

[77]:
```
#check normality of all numericaal features and transform it if not normal␣
↪distributed
for feature in train.columns:
    if (train[feature].dtype != 'object'):
```

```
        if normalityTest(train[feature]) == False:
            train[feature] = np.log1p(train[feature])
            test[feature] = np.log1p(test[feature])
```

[78]:
```
#check distribution after transformation
sns.distplot(train['GrLivArea']);
```



- Creating dummies

[79]:
```
#combine the train and the test datasets
train['Source'] = 'train'
test['Source'] = 'test'

combined_data = pd.concat([train, test], ignore_index=True)

print(train.shape, test.shape, combined_data.shape)
```

```
(1460, 90) (1459, 90) (2919, 90)
```

[80]:
```
#create dummies
combined_data = pd.get_dummies(combined_data, drop_first=True)

combined_data.shape
```

[80]:
```
(2919, 284)
```

- Creating features matrix (X) and target array (y)

[81]:
```
X = combined_data[combined_data['Source_train'] == 1].copy()
X.drop(['Source_train'], axis=1, inplace=True)

y = target_array

X_predict = combined_data[combined_data['Source_train'] == 0].copy()
X_predict.drop(['Source_train'], axis=1, inplace=True)
```

- Check for overfitting

[82]:
```
for i in X.columns:
    counts = X[i].value_counts()
    print (counts)
```

```
4.110874    143
4.262680     70
4.394449     69
3.931826     57
4.330733     53
           ...
4.927254      1
3.688879      1
4.718499      1
5.010635      1
4.934474      1
Name: LotFrontage, Length: 133, dtype: int64
8.881975    25
9.169623    24
8.699681    17
9.036106    14
9.105091    14
            ..
8.981053     1
9.224539     1
9.395242     1
9.185945     1
9.503085     1
Name: LotArea, Length: 1073, dtype: int64
1.791759    397
1.945910    374
2.079442    319
2.197225    168
1.609438    116
2.302585     43
1.386294     20
2.397895     18
```

```
1.098612      3
0.693147      2
Name: OverallQual, dtype: int64
1.791759    821
1.945910    252
2.079442    205
2.197225     72
1.609438     57
1.386294     25
2.302585     22
1.098612      5
0.693147      1
Name: OverallCond, dtype: int64
7.604396    67
7.603898    64
7.603399    54
7.604894    49
7.602900    45
            ..
7.559038     1
7.556951     1
7.535297     1
7.536897     1
7.540622     1
Name: YearBuilt, Length: 112, dtype: int64
7.576097    178
7.604396     97
7.604894     76
7.603898     73
7.603399     62
           …
7.606387      6
7.577122      5
7.592870      5
7.594381      5
7.576610      4
Name: YearRemodAdd, Length: 61, dtype: int64
0.000000    869
4.691348      8
5.198497      8
4.290459      8
4.795791      7
           …
5.686975      1
3.737670      1
5.929589      1
3.891820      1
4.043051      1
```

```
Name: MasVnrArea, Length: 327, dtype: int64
0.000000    467
3.218876     12
2.833213      9
3.044522      5
6.842683      5
           …
6.447306      1
7.009409      1
5.955837      1
7.051856      1
4.897840      1
Name: BsmtFinSF1, Length: 637, dtype: int64
0.000000    1293
5.198497       5
5.926926       3
5.686975       2
5.673323       2
           …
6.214608       1
3.583519       1
5.783825       1
6.967909       1
5.010635       1
Name: BsmtFinSF2, Length: 144, dtype: int64
0.000000    118
6.591674      9
5.953243      8
5.707110      7
6.398595      7
           …
5.710427      1
7.144407      1
6.818924      1
6.553933      1
6.855409      1
Name: BsmtUnfSF, Length: 780, dtype: int64
0.000000    37
6.762730    35
6.511745    17
6.816736    15
6.947937    14
           ..
6.901737     1
6.965080     1
7.454141     1
7.305860     1
6.486161     1
```

```
Name: TotalBsmtSF, Length: 721, dtype: int64
6.762730    25
6.947937    16
6.816736    14
6.744059    12
6.796824    12
            ..
7.111512     1
7.436617     1
7.008505     1
6.763885     1
7.642524     1
Name: 1stFlrSF, Length: 753, dtype: int64
0.000000    829
6.591674     10
6.224558      9
6.304449      8
6.511745      8
           …
6.091310      1
7.239215      1
6.936343      1
6.161207      1
6.501290      1
Name: 2ndFlrSF, Length: 417, dtype: int64
0.000000    1434
4.394449       3
5.888878       2
4.795791       1
6.177944       1
5.451038       1
6.173786       1
5.459586       1
6.270988       1
6.244167       1
5.918894       1
6.350886       1
3.988984       1
6.042633       1
6.242223       1
5.973810       1
6.161207       1
5.056246       1
5.327876       1
5.953243       1
5.986452       1
4.976734       1
5.968708       1
```

```
6.246107       1
Name: LowQualFinSF, dtype: int64
6.762730    22
6.947937    14
6.796824    11
7.284135    10
6.744059    10
            ..
7.025538     1
7.972121     1
7.074117     1
7.006695     1
7.560080     1
Name: GrLivArea, Length: 861, dtype: int64
0.000000    856
0.693147    588
1.098612     15
1.386294      1
Name: BsmtFullBath, dtype: int64
0.000000    1378
0.693147      80
1.098612       2
Name: BsmtHalfBath, dtype: int64
1.098612    768
0.693147    650
1.386294     33
0.000000      9
Name: FullBath, dtype: int64
0.000000    913
0.693147    535
1.098612     12
Name: HalfBath, dtype: int64
1.386294    804
1.098612    358
1.609438    213
0.693147     50
1.791759     21
1.945910      7
0.000000      6
2.197225      1
Name: BedroomAbvGr, dtype: int64
0.693147    1392
1.098612      65
1.386294       2
0.000000       1
Name: KitchenAbvGr, dtype: int64
1.945910    402
2.079442    329
```

```
1.791759    275
2.197225    187
1.609438     97
2.302585     75
2.397895     47
2.484907     18
1.386294     17
2.564949     11
2.708050      1
1.098612      1
Name: TotRmsAbvGrd, dtype: int64
0.000000    690
0.693147    650
1.098612    115
1.386294      5
Name: Fireplaces, dtype: int64
0.000000    81
7.603898    65
7.604396    59
7.603399    53
7.602900    50
            ..
7.554335     1
7.567346     1
7.553287     1
7.564238     1
7.550135     1
Name: GarageYrBlt, Length: 98, dtype: int64
1.098612    824
0.693147    369
1.386294    181
0.000000     81
1.609438      5
Name: GarageCars, dtype: int64
0.000000    81
6.089045    49
6.357842    47
5.484797    38
6.184149    34
            ..
6.173786     1
6.665684     1
5.293305     1
6.582025     1
6.381816     1
Name: GarageArea, Length: 441, dtype: int64
0.000000    761
5.262690     38
```

```
4.615121     36
4.976734     33
4.795791     31
           …
3.806662      1
6.754604      1
5.036953      1
5.204007      1
5.288267      1
Name: WoodDeckSF, Length: 274, dtype: int64
0.000000    656
3.610918     29
3.891820     22
3.044522     21
3.828641     19
           …
5.241747      1
5.365976      1
4.454347      1
4.927254      1
5.676754      1
Name: OpenPorchSF, Length: 202, dtype: int64
0.000000    1252
4.727388      15
4.574711       6
5.262690       5
4.795791       5
           …
5.153292       1
4.477337       1
3.218876       1
5.659482       1
5.620401       1
Name: EnclosedPorch, Length: 120, dtype: int64
0.000000    1436
5.129899       3
5.379897       2
4.976734       2
5.198497       2
5.036953       1
4.875197       1
3.178054       1
5.283204       1
5.771441       1
5.476464       1
5.720312       1
6.011267       1
5.209486       1
```

```
6.232448       1
5.505332       1
4.948760       1
5.093750       1
4.574711       1
5.673323       1
Name: 3SsnPorch, dtype: int64
0.000000    1344
5.262690       6
5.416100       5
4.795791       5
5.198497       4
            …
4.859812       1
5.036953       1
6.089045       1
4.394449       1
5.624018       1
Name: ScreenPorch, Length: 76, dtype: int64
0.000000    1453
6.253829       1
6.475433       1
6.320768       1
6.357842       1
6.605298       1
6.240276       1
6.175867       1
Name: PoolArea, dtype: int64
0.000000    1408
5.993961      11
6.216606       8
6.552508       5
7.601402       4
6.398595       4
6.111467       4
7.090910       2
6.175867       2
6.431331       1
9.024131       1
5.860786       1
7.048386       1
4.007333       1
6.329721       1
8.160804       1
7.244942       1
6.685861       1
7.824446       1
9.648660       1
```

```
7.170888       1
Name: MiscVal, dtype: int64
1.945910    253
2.079442    234
1.791759    204
1.609438    141
2.197225    122
1.386294    106
2.397895     89
2.484907     79
2.302585     63
2.564949     59
0.693147     58
1.098612     52
Name: MoSold, dtype: int64
7.605890    338
7.604894    329
7.604396    314
7.605392    304
7.606387    175
Name: YrSold, dtype: int64
7.455298    22
7.489412    12
7.436617    11
7.398174    10
7.689371     9
            ..
7.815611     1
7.572503     1
8.172729     1
7.868254     1
8.066208     1
Name: TotalSF, Length: 963, dtype: int64
1.098612    456
1.252763    295
0.693147    228
1.386294    186
1.504077    144
0.916291    129
1.609438     13
1.704748      7
1.791759      1
1.945910      1
Name: total_bathrooms, dtype: int64
0.000000    254
4.795791     21
5.262690     18
5.129899     16
```

```
4.948760     15
           …
6.335054      1
4.564348      1
6.006353      1
6.011267      1
3.401197      1
Name: Total_porch_sf, Length: 427, dtype: int64
0.000000    764
0.693359    696
Name: has_been_remod, dtype: int64
0.693359    1379
0.000000      81
Name: has_garage, dtype: int64
0.693359    1423
0.000000      37
Name: has_basement, dtype: int64
0.000000    829
0.693359    631
Name: has_2ndFloor, dtype: int64
0.693359    770
0.000000    690
Name: has_fireplace, dtype: int64
0.000000    1453
0.693359       7
Name: has_pool, dtype: int64
0.000000    1179
0.693359     281
Name: has_fence, dtype: int64
0    1460
Name: MSSubClass_150, dtype: int64
0    1397
1      63
Name: MSSubClass_160, dtype: int64
0    1450
1      10
Name: MSSubClass_180, dtype: int64
0    1430
1      30
Name: MSSubClass_190, dtype: int64
0    924
1    536
Name: MSSubClass_20, dtype: int64
0    1391
1      69
Name: MSSubClass_30, dtype: int64
0    1456
1       4
```

```
Name: MSSubClass_40, dtype: int64
0    1448
1      12
Name: MSSubClass_45, dtype: int64
0    1316
1     144
Name: MSSubClass_50, dtype: int64
0    1161
1     299
Name: MSSubClass_60, dtype: int64
0    1400
1      60
Name: MSSubClass_70, dtype: int64
0    1444
1      16
Name: MSSubClass_75, dtype: int64
0    1402
1      58
Name: MSSubClass_80, dtype: int64
0    1440
1      20
Name: MSSubClass_85, dtype: int64
0    1408
1      52
Name: MSSubClass_90, dtype: int64
0    1395
1      65
Name: MSZoning_FV, dtype: int64
0    1444
1      16
Name: MSZoning_RH, dtype: int64
1    1151
0     309
Name: MSZoning_RL, dtype: int64
0    1242
1     218
Name: MSZoning_RM, dtype: int64
1    1454
0       6
Name: Street_Pave, dtype: int64
1    1369
0      91
Name: Alley_None, dtype: int64
0    1419
1      41
Name: Alley_Pave, dtype: int64
0    1419
1      41
```

```
Name: LotShape_IR2, dtype: int64
0    1450
1      10
Name: LotShape_IR3, dtype: int64
1    925
0    535
Name: LotShape_Reg, dtype: int64
0    1410
1      50
Name: LandContour_HLS, dtype: int64
0    1424
1      36
Name: LandContour_Low, dtype: int64
1    1311
0     149
Name: LandContour_Lvl, dtype: int64
0    1459
1       1
Name: Utilities_NoSeWa, dtype: int64
0    1366
1      94
Name: LotConfig_CulDSac, dtype: int64
0    1413
1      47
Name: LotConfig_FR2, dtype: int64
0    1456
1       4
Name: LotConfig_FR3, dtype: int64
1    1052
0     408
Name: LotConfig_Inside, dtype: int64
0    1395
1      65
Name: LandSlope_Mod, dtype: int64
0    1447
1      13
Name: LandSlope_Sev, dtype: int64
0    1458
1       2
Name: Neighborhood_Blueste, dtype: int64
0    1444
1      16
Name: Neighborhood_BrDale, dtype: int64
0    1402
1      58
Name: Neighborhood_BrkSide, dtype: int64
0    1432
1      28
```

```
Name: Neighborhood_ClearCr, dtype: int64
0    1310
1     150
Name: Neighborhood_CollgCr, dtype: int64
0    1409
1      51
Name: Neighborhood_Crawfor, dtype: int64
0    1360
1     100
Name: Neighborhood_Edwards, dtype: int64
0    1381
1      79
Name: Neighborhood_Gilbert, dtype: int64
0    1423
1      37
Name: Neighborhood_IDOTRR, dtype: int64
0    1443
1      17
Name: Neighborhood_MeadowV, dtype: int64
0    1411
1      49
Name: Neighborhood_Mitchel, dtype: int64
0    1235
1     225
Name: Neighborhood_NAmes, dtype: int64
0    1451
1       9
Name: Neighborhood_NPkVill, dtype: int64
0    1387
1      73
Name: Neighborhood_NWAmes, dtype: int64
0    1419
1      41
Name: Neighborhood_NoRidge, dtype: int64
0    1383
1      77
Name: Neighborhood_NridgHt, dtype: int64
0    1347
1     113
Name: Neighborhood_OldTown, dtype: int64
0    1435
1      25
Name: Neighborhood_SWISU, dtype: int64
0    1386
1      74
Name: Neighborhood_Sawyer, dtype: int64
0    1401
1      59
```

```
Name: Neighborhood_SawyerW, dtype: int64
0    1374
1      86
Name: Neighborhood_Somerst, dtype: int64
0    1435
1      25
Name: Neighborhood_StoneBr, dtype: int64
0    1422
1      38
Name: Neighborhood_Timber, dtype: int64
0    1449
1      11
Name: Neighborhood_Veenker, dtype: int64
0    1379
1      81
Name: Condition1_Feedr, dtype: int64
1    1260
0     200
Name: Condition1_Norm, dtype: int64
0    1452
1       8
Name: Condition1_PosA, dtype: int64
0    1441
1      19
Name: Condition1_PosN, dtype: int64
0    1449
1      11
Name: Condition1_RRAe, dtype: int64
0    1434
1      26
Name: Condition1_RRAn, dtype: int64
0    1458
1       2
Name: Condition1_RRNe, dtype: int64
0    1455
1       5
Name: Condition1_RRNn, dtype: int64
0    1454
1       6
Name: Condition2_Feedr, dtype: int64
1    1445
0      15
Name: Condition2_Norm, dtype: int64
0    1459
1       1
Name: Condition2_PosA, dtype: int64
0    1458
1       2
```

```
Name: Condition2_PosN, dtype: int64
0    1459
1       1
Name: Condition2_RRAe, dtype: int64
0    1459
1       1
Name: Condition2_RRAn, dtype: int64
0    1458
1       2
Name: Condition2_RRNn, dtype: int64
0    1429
1      31
Name: BldgType_2fmCon, dtype: int64
0    1408
1      52
Name: BldgType_Duplex, dtype: int64
0    1417
1      43
Name: BldgType_Twnhs, dtype: int64
0    1346
1     114
Name: BldgType_TwnhsE, dtype: int64
0    1446
1      14
Name: HouseStyle_1.5Unf, dtype: int64
0    734
1    726
Name: HouseStyle_1Story, dtype: int64
0    1452
1       8
Name: HouseStyle_2.5Fin, dtype: int64
0    1449
1      11
Name: HouseStyle_2.5Unf, dtype: int64
0    1015
1     445
Name: HouseStyle_2Story, dtype: int64
0    1423
1      37
Name: HouseStyle_SFoyer, dtype: int64
0    1395
1      65
Name: HouseStyle_SLvl, dtype: int64
1    1141
0     319
Name: RoofStyle_Gable, dtype: int64
0    1449
1      11
```

```
Name: RoofStyle_Gambrel, dtype: int64
0    1174
1     286
Name: RoofStyle_Hip, dtype: int64
0    1453
1       7
Name: RoofStyle_Mansard, dtype: int64
0    1458
1       2
Name: RoofStyle_Shed, dtype: int64
1    1434
0      26
Name: RoofMatl_CompShg, dtype: int64
0    1459
1       1
Name: RoofMatl_Membran, dtype: int64
0    1459
1       1
Name: RoofMatl_Metal, dtype: int64
0    1459
1       1
Name: RoofMatl_Roll, dtype: int64
0    1449
1      11
Name: RoofMatl_Tar&Grv, dtype: int64
0    1455
1       5
Name: RoofMatl_WdShake, dtype: int64
0    1454
1       6
Name: RoofMatl_WdShngl, dtype: int64
0    1459
1       1
Name: Exterior1st_AsphShn, dtype: int64
0    1458
1       2
Name: Exterior1st_BrkComm, dtype: int64
0    1410
1      50
Name: Exterior1st_BrkFace, dtype: int64
0    1459
1       1
Name: Exterior1st_CBlock, dtype: int64
0    1399
1      61
Name: Exterior1st_CemntBd, dtype: int64
0    1238
1     222
```

```
Name: Exterior1st_HdBoard, dtype: int64
0    1459
1       1
Name: Exterior1st_ImStucc, dtype: int64
0    1240
1     220
Name: Exterior1st_MetalSd, dtype: int64
0    1352
1     108
Name: Exterior1st_Plywood, dtype: int64
0    1458
1       2
Name: Exterior1st_Stone, dtype: int64
0    1435
1      25
Name: Exterior1st_Stucco, dtype: int64
0    945
1    515
Name: Exterior1st_VinylSd, dtype: int64
0    1254
1     206
Name: Exterior1st_Wd Sdng, dtype: int64
0    1434
1      26
Name: Exterior1st_WdShing, dtype: int64
0    1457
1       3
Name: Exterior2nd_AsphShn, dtype: int64
0    1453
1       7
Name: Exterior2nd_Brk Cmn, dtype: int64
0    1435
1      25
Name: Exterior2nd_BrkFace, dtype: int64
0    1459
1       1
Name: Exterior2nd_CBlock, dtype: int64
0    1400
1      60
Name: Exterior2nd_CmentBd, dtype: int64
0    1253
1     207
Name: Exterior2nd_HdBoard, dtype: int64
0    1450
1      10
Name: Exterior2nd_ImStucc, dtype: int64
0    1246
1     214
```

```
Name: Exterior2nd_MetalSd, dtype: int64
0    1459
1       1
Name: Exterior2nd_Other, dtype: int64
0    1318
1     142
Name: Exterior2nd_Plywood, dtype: int64
0    1455
1       5
Name: Exterior2nd_Stone, dtype: int64
0    1434
1      26
Name: Exterior2nd_Stucco, dtype: int64
0    956
1    504
Name: Exterior2nd_VinylSd, dtype: int64
0    1263
1     197
Name: Exterior2nd_Wd Sdng, dtype: int64
0    1422
1      38
Name: Exterior2nd_Wd Shng, dtype: int64
0    1015
1     445
Name: MasVnrType_BrkFace, dtype: int64
1    872
0    588
Name: MasVnrType_None, dtype: int64
0    1332
1     128
Name: MasVnrType_Stone, dtype: int64
0    1446
1      14
Name: ExterQual_Fa, dtype: int64
0    972
1    488
Name: ExterQual_Gd, dtype: int64
1    906
0    554
Name: ExterQual_TA, dtype: int64
0    1432
1      28
Name: ExterCond_Fa, dtype: int64
0    1314
1     146
Name: ExterCond_Gd, dtype: int64
0    1459
1       1
```

```
Name: ExterCond_Po, dtype: int64
1    1282
0     178
Name: ExterCond_TA, dtype: int64
0    826
1    634
Name: Foundation_CBlock, dtype: int64
0    813
1    647
Name: Foundation_PConc, dtype: int64
0    1436
1      24
Name: Foundation_Slab, dtype: int64
0    1454
1       6
Name: Foundation_Stone, dtype: int64
0    1457
1       3
Name: Foundation_Wood, dtype: int64
0    1425
1      35
Name: BsmtQual_Fa, dtype: int64
0    842
1    618
Name: BsmtQual_Gd, dtype: int64
0    1423
1      37
Name: BsmtQual_None, dtype: int64
0    811
1    649
Name: BsmtQual_TA, dtype: int64
0    1395
1      65
Name: BsmtCond_Gd, dtype: int64
0    1423
1      37
Name: BsmtCond_None, dtype: int64
0    1458
1       2
Name: BsmtCond_Po, dtype: int64
1    1311
0     149
Name: BsmtCond_TA, dtype: int64
0    1326
1     134
Name: BsmtExposure_Gd, dtype: int64
0    1346
1     114
```

```
Name: BsmtExposure_Mn, dtype: int64
1    953
0    507
Name: BsmtExposure_No, dtype: int64
0    1422
1      38
Name: BsmtExposure_None, dtype: int64
0    1312
1     148
Name: BsmtFinType1_BLQ, dtype: int64
0    1042
1     418
Name: BsmtFinType1_GLQ, dtype: int64
0    1386
1      74
Name: BsmtFinType1_LwQ, dtype: int64
0    1423
1      37
Name: BsmtFinType1_None, dtype: int64
0    1327
1     133
Name: BsmtFinType1_Rec, dtype: int64
0    1030
1     430
Name: BsmtFinType1_Unf, dtype: int64
0    1427
1      33
Name: BsmtFinType2_BLQ, dtype: int64
0    1446
1      14
Name: BsmtFinType2_GLQ, dtype: int64
0    1414
1      46
Name: BsmtFinType2_LwQ, dtype: int64
0    1422
1      38
Name: BsmtFinType2_None, dtype: int64
0    1406
1      54
Name: BsmtFinType2_Rec, dtype: int64
1    1256
0     204
Name: BsmtFinType2_Unf, dtype: int64
1    1428
0      32
Name: Heating_GasA, dtype: int64
0    1442
1      18
```

```
Name: Heating_GasW, dtype: int64
0    1453
1       7
Name: Heating_Grav, dtype: int64
0    1458
1       2
Name: Heating_OthW, dtype: int64
0    1456
1       4
Name: Heating_Wall, dtype: int64
0    1411
1      49
Name: HeatingQC_Fa, dtype: int64
0    1219
1     241
Name: HeatingQC_Gd, dtype: int64
0    1459
1       1
Name: HeatingQC_Po, dtype: int64
0    1032
1     428
Name: HeatingQC_TA, dtype: int64
1    1365
0      95
Name: CentralAir_Y, dtype: int64
0    1433
1      27
Name: Electrical_FuseF, dtype: int64
0    1457
1       3
Name: Electrical_FuseP, dtype: int64
0    1459
1       1
Name: Electrical_Mix, dtype: int64
1    1335
0     125
Name: Electrical_SBrkr, dtype: int64
0    1421
1      39
Name: KitchenQual_Fa, dtype: int64
0    874
1    586
Name: KitchenQual_Gd, dtype: int64
1    735
0    725
Name: KitchenQual_TA, dtype: int64
0    1455
1       5
```

```
Name: Functional_Maj2, dtype: int64
0    1429
1      31
Name: Functional_Min1, dtype: int64
0    1426
1      34
Name: Functional_Min2, dtype: int64
0    1445
1      15
Name: Functional_Mod, dtype: int64
0    1459
1       1
Name: Functional_Sev, dtype: int64
1    1360
0     100
Name: Functional_Typ, dtype: int64
0    1427
1      33
Name: FireplaceQu_Fa, dtype: int64
0    1080
1     380
Name: FireplaceQu_Gd, dtype: int64
0    770
1    690
Name: FireplaceQu_None, dtype: int64
0    1440
1      20
Name: FireplaceQu_Po, dtype: int64
0    1147
1     313
Name: FireplaceQu_TA, dtype: int64
1    870
0    590
Name: GarageType_Attchd, dtype: int64
0    1441
1      19
Name: GarageType_Basment, dtype: int64
0    1372
1      88
Name: GarageType_BuiltIn, dtype: int64
0    1451
1       9
Name: GarageType_CarPort, dtype: int64
0    1073
1     387
Name: GarageType_Detchd, dtype: int64
0    1379
1      81
```

```
Name: GarageType_None, dtype: int64
0    1379
1      81
Name: GarageFinish_None, dtype: int64
0    1038
1     422
Name: GarageFinish_RFn, dtype: int64
0    855
1    605
Name: GarageFinish_Unf, dtype: int64
0    1412
1      48
Name: GarageQual_Fa, dtype: int64
0    1446
1      14
Name: GarageQual_Gd, dtype: int64
0    1379
1      81
Name: GarageQual_None, dtype: int64
0    1457
1       3
Name: GarageQual_Po, dtype: int64
1    1311
0     149
Name: GarageQual_TA, dtype: int64
0    1425
1      35
Name: GarageCond_Fa, dtype: int64
0    1451
1       9
Name: GarageCond_Gd, dtype: int64
0    1379
1      81
Name: GarageCond_None, dtype: int64
0    1453
1       7
Name: GarageCond_Po, dtype: int64
1    1326
0     134
Name: GarageCond_TA, dtype: int64
0    1430
1      30
Name: PavedDrive_P, dtype: int64
1    1340
0     120
Name: PavedDrive_Y, dtype: int64
0    1458
1       2
```

```
Name: PoolQC_Fa, dtype: int64
0    1457
1       3
Name: PoolQC_Gd, dtype: int64
1    1453
0       7
Name: PoolQC_None, dtype: int64
0    1406
1      54
Name: Fence_GdWo, dtype: int64
0    1303
1     157
Name: Fence_MnPrv, dtype: int64
0    1449
1      11
Name: Fence_MnWw, dtype: int64
1    1179
0     281
Name: Fence_None, dtype: int64
1    1406
0      54
Name: MiscFeature_None, dtype: int64
0    1458
1       2
Name: MiscFeature_Othr, dtype: int64
0    1411
1      49
Name: MiscFeature_Shed, dtype: int64
0    1459
1       1
Name: MiscFeature_TenC, dtype: int64
0    1456
1       4
Name: SaleType_CWD, dtype: int64
0    1458
1       2
Name: SaleType_Con, dtype: int64
0    1451
1       9
Name: SaleType_ConLD, dtype: int64
0    1455
1       5
Name: SaleType_ConLI, dtype: int64
0    1455
1       5
Name: SaleType_ConLw, dtype: int64
0    1338
1     122
```

```
Name: SaleType_New, dtype: int64
0    1457
1       3
Name: SaleType_Oth, dtype: int64
1    1267
0     193
Name: SaleType_WD, dtype: int64
0    1456
1       4
Name: SaleCondition_AdjLand, dtype: int64
0    1448
1      12
Name: SaleCondition_Alloca, dtype: int64
0    1440
1      20
Name: SaleCondition_Family, dtype: int64
1    1198
0     262
Name: SaleCondition_Normal, dtype: int64
0    1335
1     125
Name: SaleCondition_Partial, dtype: int64
```

[83]:
```
def overfit_zeros(df, limit=99.95):
    """df (dataframe)  : data
       limit (float)   : limit to be called overfitted
       Returns a list of features that have redundant zeroes and caused␣
 ↪overfitting.
    """
    overfit = []

    for i in df.columns:
        counts = df[i].value_counts()
        zeros = counts.iloc[0]
        if zeros / len(df) * 100 > limit:
            overfit.append(i)

    overfit = list(overfit)

    return overfit
```

[84]:
```
#drop overfitted features
features_overfitted_train = overfit_zeros(X)

X.drop(features_overfitted_train, axis=1, inplace=True)
X_predict.drop(features_overfitted_train, axis=1, inplace=True)
```

5 Creating a Model

In this case, we will compare Linear Regression model and Gradient Boosting Regressor to get the smallest Mean Squared Error (MSE). We begin by splitting data into two subsets: for training data and for testing data.

[86]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .1,␣
↪random_state = 0)
```

[87]:
```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import GradientBoostingRegressor

model_linear_reg = LinearRegression()
model_GBReg = GradientBoostingRegressor()
```

[88]:
```
#search grid for optimal parameters
from sklearn.model_selection import GridSearchCV

linear_reg_param_grid = {'normalize': [True, False],
                         'n_jobs': [None, -1]}

GBReg_param_grid = {'n_estimators' : [3000],
                    'learning_rate' : [0.05],
                    'max_depth' : [4],
                    'max_features' : ['sqrt'],
                    'min_samples_leaf' : [15],
                    'min_samples_split' : [10],
                    'loss' : ['huber'],
                    'random_state' : [42]}

grid_linear_reg = GridSearchCV(model_linear_reg, linear_reg_param_grid, cv=5)
grid_GBReg = GridSearchCV(model_GBReg, GBReg_param_grid, cv=5)

grid_linear_reg.fit(X_train, y_train)
grid_GBReg.fit(X_train, y_train)

#print(grid.best_params_)
#print(grid.best_score_)
```

[88]:
```
GridSearchCV(cv=5, error_score=nan,
             estimator=GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0,
                                                 criterion='friedman_mse',
                                                 init=None, learning_rate=0.1,
                                                 loss='ls', max_depth=3,
                                                 max_features=None,
```

```
                                                max_leaf_nodes=None,
                                                min_impurity_decrease=0.0,
                                                min_impurity_split=None,
                                                min_samples_leaf=1,
                                                min_samples_split=2,
                                                min_weight_fraction_leaf=0.0,
                                                n_estimators=100,
                                                n_iter_n…
                                                subsample=1.0, tol=0.0001,
                                                validation_fraction=0.1,
                                                verbose=0, warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid={'learning_rate': [0.05], 'loss': ['huber'],
                         'max_depth': [4], 'max_features': ['sqrt'],
                         'min_samples_leaf': [15], 'min_samples_split': [10],
                         'n_estimators': [3000], 'random_state': [42]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

[89]:
```
#calculate Mean Squared Error
from sklearn.metrics import mean_squared_error

print('MSE linear regression: ', mean_squared_error(y_test, grid_linear_reg.
 ↪best_estimator_.predict(X_test)))
print('MSE Gradient Boosting Regressor : ', mean_squared_error(y_test,␣
 ↪grid_GBReg.best_estimator_.predict(X_test)))
```

```
MSE linear regression:  0.0199487065549333
MSE Gradient Boosting Regressor :  0.011674093709881473
```

[90]:
```
#use the best model
model = grid_GBReg.best_estimator_
y_predict = model.predict(X_predict)

#transform the values back
y_predict = np.expm1(y_predict)
```

[91]:
```
#sava results to a file
results = pd.DataFrame({'Id': test_id, 'SalePrice': y_predict})
results.to_csv('my_submission.csv', index=False)
```

[ ]:
```

```