Project-Housing_splitted

Report

Prepared by KISHAN SAPARIYA

Data Science Intern at Flip Robo Technologies

BATCH 34

FLIP ROBO

SME Name:

Khushboo Garg

Acknowledgement

- Here we discuss all the Acknowledgement which is help me to complete this project for that first I thank to fliprobo technologies for help complete this project when I stacked in the one of removing outlier then I discuss with the team of fliprobo technologies and they are help me a lot. Also one of my best friend is google when I forget any command or any name of function or library then I will use google by use of I complete the steps. Also I used sklearn site for finding hyper parameter of the model.

Problem Statement

- Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy.
- It is a very large market and there are various companies working in the domain.
- Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases.
- Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies.
- Our problem is related to one such housing company.
- A US-based housing company named **Surprise Housing** has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price.
- The company is looking at prospective properties to buy houses to enter the market. You are required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:
    - **Which variables are important to predict the price of variable?**
    - **How do these variables describe the price of the house?**

• Business Goal:

- We are required to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

Analytical Problem Framing

- For starting data analysis we should first import require basic Libraries for that we import pandas, numpy , matplotlib and seaborn for import data and visualization it. If it is not installed in our pc first we have to install it and then import it other wise we will get error such as module not found.

```
In [1]:
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

- Now our next step is to import data by use of pandas there are many functions in pandas for import libraries such as csv, xlsx and many more. But we observe that we have CSV file (Comma Separated value) so we use csv function. If our data is in other format we will use any other method to import it.

```
In [2]:
df=pd.read_csv('train.csv')
df
```

- Here we import our data which is only training data for use it to train our model. Now our data is imported. So now we take a look of it.

Out[2]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	Misc
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	

- As we observe there are many columns and row so now for check how many row and columns we use pandas shape function.

```
In [3]:
df.shape
Out[3]: (1168, 81)
```

- By applying pandas shape function we observe that there are total 1168 rows and 81 columns in 81 columns there are 80 input columns and 1 target column.
- Now we check unique values in each columns by use of pandas unique function to find unique values in column.
- As observe by unique values we observe that in Utilities column having only one single unique value so we drop it from our data frame.

```
In [19]:
df.drop(['Utilities'],axis=1,inplace=True)
```

- Now we observe for null values as we observe there are many columns such as 80 for check 80 columns null values one by one it is time consuming for that we use for loop to check all the null values in every columns by use of it's column name and use of for loop.

```
In [21]: df.columns
```

```
Out[21]: Index(['MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street', 'Alley',
       'LotShape', 'LandContour', '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')
```

- For finding the name of columns.

```
In [22]: Null=['MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street', 'Alley',
       'LotShape', 'LandContour', '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']
for i in Null:
    print(str(i),'=',df[i].isnull().sum())
```

- Now our code is available and run it and then we observe the null values.

```
MSSubClass = 0
MSZoning = 0
LotFrontage = 214
LotArea = 0
Street = 0
Alley = 1091
LotShape = 0
LandContour = 0
LotConfig = 0
LandSlope = 0
Neighborhood = 0
Condition1 = 0
Condition2 = 0
BldgType = 0
HouseStyle = 0
OverallQual = 0
OverallCond = 0
YearBuilt = 0
YearRemodAdd = 0
RoofStyle = 0
RoofMatl = 0
Exterior1st = 0
Exterior2nd = 0
MasVnrType = 7
MasVnrArea = 7
ExterQual = 0
ExterCond = 0
Foundation = 0
BsmtQual = 30
BsmtCond = 30
BsmtExposure = 31
BsmtFinType1 = 30
```

```
BsmtFinSF1 = 0
BsmtFinType2 = 31
BsmtFinSF2 = 0
BsmtUnfSF = 0
TotalBsmtSF = 0
Heating = 0
HeatingQC = 0
CentralAir = 0
Electrical = 0
1stFlrSF = 0
2ndFlrSF = 0
LowQualFinSF = 0
GrLivArea = 0
BsmtFullBath = 0
BsmtHalfBath = 0
FullBath = 0
HalfBath = 0
BedroomAbvGr = 0
KitchenAbvGr = 0
KitchenQual = 0
TotRmsAbvGrd = 0
Functional = 0
Fireplaces = 0
FireplaceQu = 551
GarageType = 64
GarageYrBlt = 64
GarageFinish = 64
GarageCars = 0
GarageArea = 0
GarageQual = 64
GarageCond = 64
PavedDrive = 0
WoodDeckSF = 0
OpenPorchSF = 0
EnclosedPorch = 0
3SsnPorch = 0
ScreenPorch = 0
PoolArea = 0
PoolQC = 1161
Fence = 931
MiscFeature = 1124
MiscVal = 0
MoSold = 0
YrSold = 0
SaleType = 0
SaleCondition = 0
SalePrice = 0
```

- By observe in it there are many columns having null values. We observe that above 500 null values we drop those columns.
- As we observe that these columns are contains above 500 Null values.

1. Alley = 1091
2. FireplaceQu = 551
3. PoolQC = 1161
4. Fence = 931
5. MiscFeature = 1124

- Now we drop that columns by use of drop functions.

```
In [24]: df.drop(['Alley','FireplaceQu','PoolQC','Fence','MiscFeature'],axis=1,inplace=True)

In [25]: df.head()
```

Out[25]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	...	EnclosedPorch	3SsnPorch	S
0	120	RL	NaN	4928	Pave	IR1	Lvl	Inside	Gtl	NPkVill	...	0	0	
1	20	RL	95.0	15865	Pave	IR1	Lvl	Inside	Mod	NAmes	...	0	0	
2	60	RL	92.0	9920	Pave	IR1	Lvl	CulDSac	Gtl	NoRidge	...	0	0	
3	20	RL	105.0	11751	Pave	IR1	Lvl	Inside	Gtl	NWAmes	...	0	0	
4	20	RL	NaN	16635	Pave	IR1	Lvl	FR2	Gtl	NWAmes	...	0	0	

5 rows × 74 columns

- Now we observe that those columns are dropped and now our data set size is 74 columns including our target columns.
- Now for remaining null values we fill it by check is data type and then we fill it by use of it's mean, median and mode according data type.

```
In [26]: df['LotFrontage'].dtype
Out[26]: dtype('float64')
```

as we observe this is in float so we fill it by mean.

```
In [27]: df['LotFrontage']=df['LotFrontage'].fillna(np.mean(df['LotFrontage']))
```

```
In [28]: df['MasVnrType'].dtype
Out[28]: dtype('O')
```

as we observe this is in object so we can fill it by it's max frequency.

```
In [29]: df['MasVnrType'].value_counts()
Out[29]: None       696
         BrkFace    354
         Stone       98
         BrkCmn      13
         Name: MasVnrType, dtype: int64
```

```
In [30]: df['MasVnrType']=df['MasVnrType'].fillna('None')
```

```
In [31]: df['MasVnrArea'].dtypes
Out[31]: dtype('float64')
```

As we observe this is in float type so we fill it by mean.

```
In [32]: df['MasVnrArea']=df['MasVnrArea'].fillna(np.mean(df['MasVnrArea']))
```

```
In [33]: df['BsmtQual'].dtypes
Out[33]: dtype('O')
```

```
In [34]: df['BsmtQual'].value_counts()
Out[34]: TA    517
         Gd    498
         Ex     94
         Fa     29
         Name: BsmtQual, dtype: int64
```

```
In [35]: df['BsmtQual']=df['BsmtQual'].fillna('TA')
```

```
In [36]: df['BsmtCond'].dtypes
Out[36]: dtype('O')
```

```
In [37]: df['BsmtCond'].value_counts()
Out[37]: TA    1041
         Gd      56
         Fa      39
         Po       2
         Name: BsmtCond, dtype: int64
```

```
In [38]: df['BsmtCond']=df['BsmtCond'].fillna('TA')
```

In [39]:
```
df['BsmtExposure'].dtypes
```

Out[39]:
```
dtype('O')
```

In [40]:
```
df['BsmtExposure'].value_counts()
```

Out[40]:
```
No    756
Av    180
Gd    108
Mn     93
Name: BsmtExposure, dtype: int64
```

In [41]:
```
df['BsmtExposure']=df['BsmtExposure'].fillna('No')
```

In [42]:
```
df['BsmtFinType1'].dtypes
```

Out[42]:
```
dtype('O')
```

In [43]:
```
df['BsmtFinType1'].value_counts()
```

Out[43]:
```
Unf    345
GLQ    330
ALQ    174
BLQ    121
Rec    109
LwQ     59
Name: BsmtFinType1, dtype: int64
```

In [44]:
```
df['BsmtFinType1']=df['BsmtFinType1'].fillna('Unf')
```

In [45]:
```
df['BsmtFinType2'].dtypes
```

Out[45]:
```
dtype('O')
```

In [46]:
```
df['BsmtFinType2'].value_counts()
```

Out[46]:
```
Unf    1002
Rec      43
LwQ      40
BLQ      24
ALQ      16
GLQ      12
Name: BsmtFinType2, dtype: int64
```

In [47]:
```
df['BsmtFinType2']=df['BsmtFinType2'].fillna('Unf')
```

In [48]:
```
df['GarageType'].dtypes
```

Out[48]:
```
dtype('O')
```

In [49]:
```
df['GarageType'].value_counts()
```

Out[49]:
```
Attchd     691
Detchd     314
BuiltIn     70
Basment     16
CarPort      8
2Types       5
Name: GarageType, dtype: int64
```

In [50]:
```
df['GarageType']=df['GarageType'].fillna('Attchd')
```

```
In [51]: df['GarageYrBlt'].dtypes
Out[51]: dtype('float64')

In [52]: df['GarageYrBlt']=df['GarageYrBlt'].fillna(np.median(df['GarageYrBlt']))

In [53]: df['GarageFinish'].dtypes
Out[53]: dtype('O')

In [54]: df['GarageFinish'].value_counts()
Out[54]: Unf    487
         RFn    339
         Fin    278
         Name: GarageFinish, dtype: int64

In [55]: df['GarageFinish']=df['GarageFinish'].fillna('Unf')

In [56]: df['GarageQual'].dtypes
Out[56]: dtype('O')

In [57]: df['GarageQual'].value_counts()
Out[57]: TA    1050
         Fa      39
         Gd      11
         Ex       2
         Po       2

In [58]: df['GarageQual']=df['GarageQual'].fillna('TA')

In [59]: df['GarageCond'].dtypes
Out[59]: dtype('O')

In [60]: df['GarageCond'].value_counts()
Out[60]: TA    1061
         Fa      28
         Gd       8
         Po       6
         Ex       1
         Name: GarageCond, dtype: int64

In [61]: df['GarageCond']=df['GarageCond'].fillna('TA')

In [62]: df['GarageYrBlt'].dtypes
Out[62]: dtype('float64')

In [63]: df['GarageYrBlt']=df['GarageYrBlt'].fillna(np.mean(df['GarageYrBlt']))
```

- Now we observe that we fill all the null values by check it's data type and fill it by mean, median and mode.
- Now check all once if there are any null values or not check.

```
In [65]: Nulll=['MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
                'LotShape', 'LandContour', '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', 'GarageType',
                'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
                'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
                'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal',
                'MoSold', 'YrSold', 'SaleType', 'SaleCondition', 'SalePrice']

         for i in Nulll:
             print(str(i),'=',df[i].isnull().sum())
```

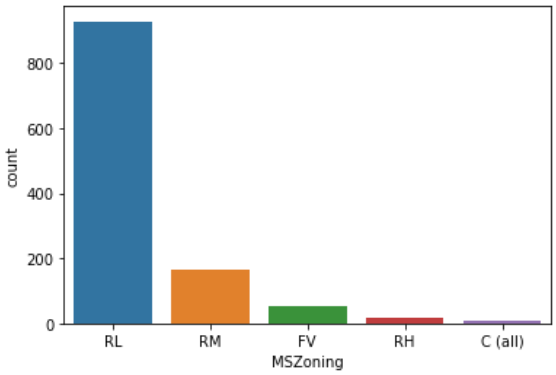
- By observing out put of this for loop we observe that there are no null values in this data set now we filled it.
- Now start EDA process of all the data. For EDA we observe that there are total two types of data in this dataset we observe that there are total two types of data numerical and object. By use of for loop we By forget data in two part ordinal and numerical.

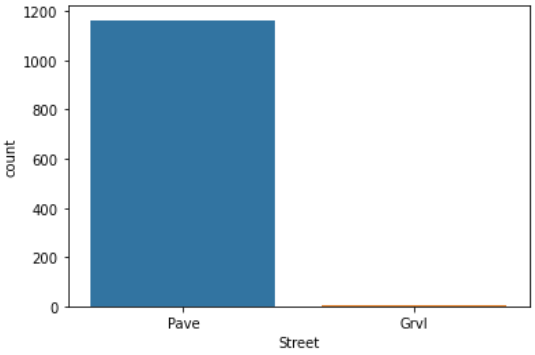
In [68]:
```
Byforget=['MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'LotShape', 'LandContour', 'LotConfig', 'LandSlope', 'Neighborhood',
       'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'OverallQual',
       'OverallCond', 'YearBuilt', 'YearRemodAdd', 'RoofStyle', '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', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal',
       'MoSold', 'YrSold', 'SaleType', 'SaleCondition', 'SalePrice']
objects=[]
numerical=[]
for i in Byforget:
    if df[i].dtypes=='object':
        objects.append(i)
    else:
        numerical.append(i)
```

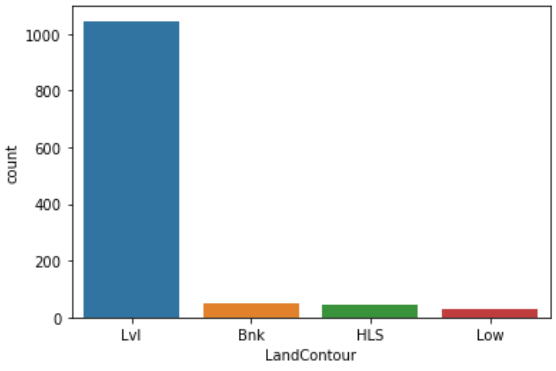
- By observing this for loop we observe that our data is now By forget in two part object and numerical.

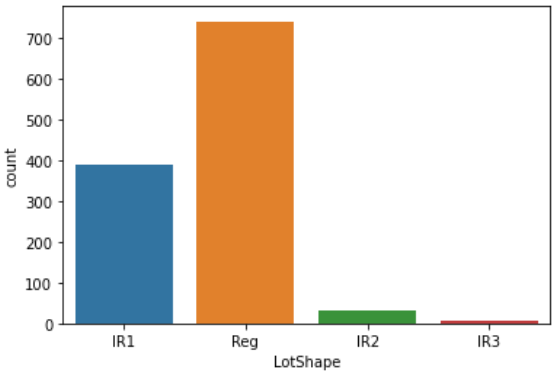
In [71]:
```
for i in objects:
    sns.countplot(df[i])
    plt.show()
```

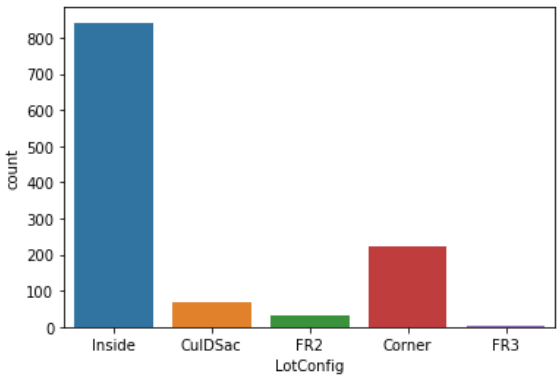
- By observing this for loop we observe count plot and then we conclude some part of it.

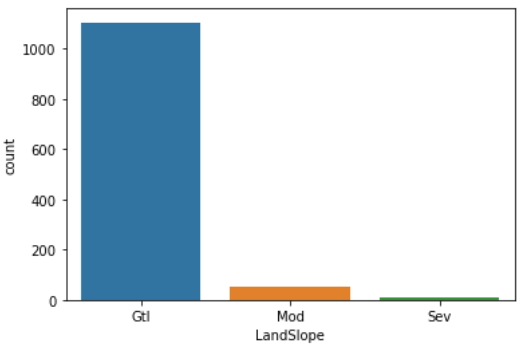


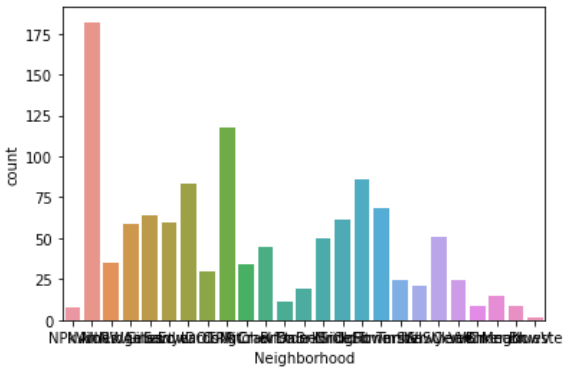


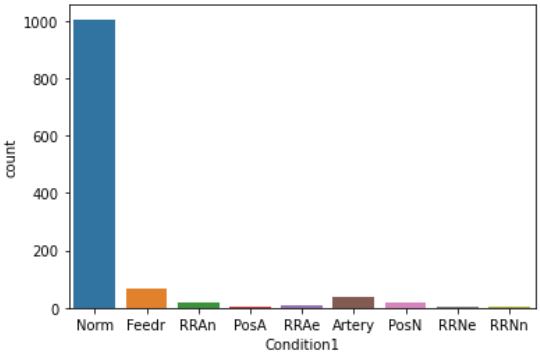


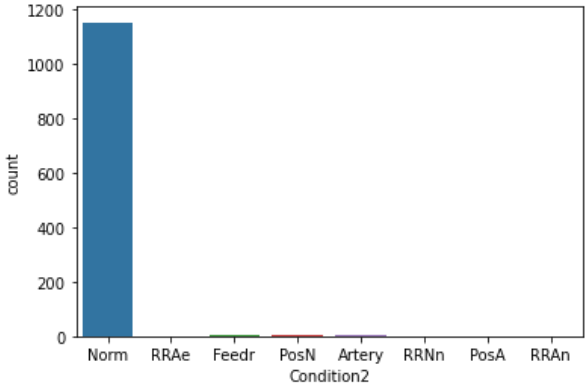


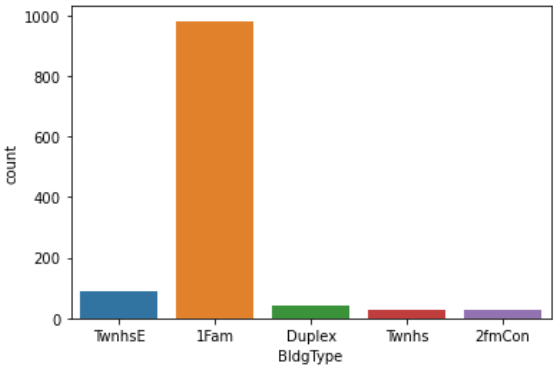


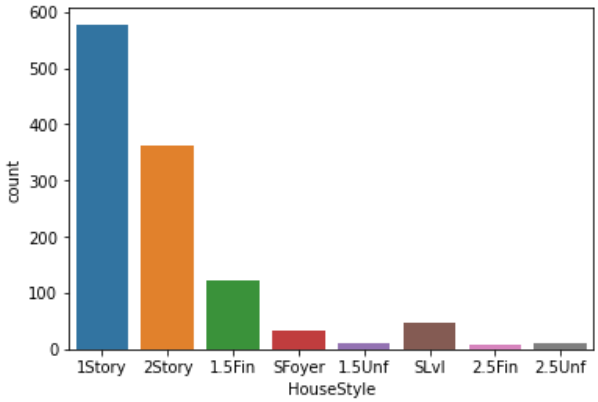


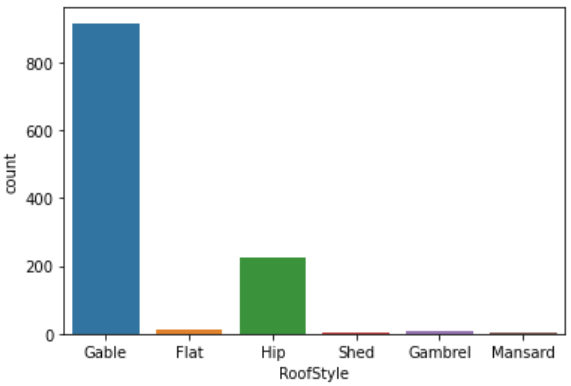

count
Neighborhood

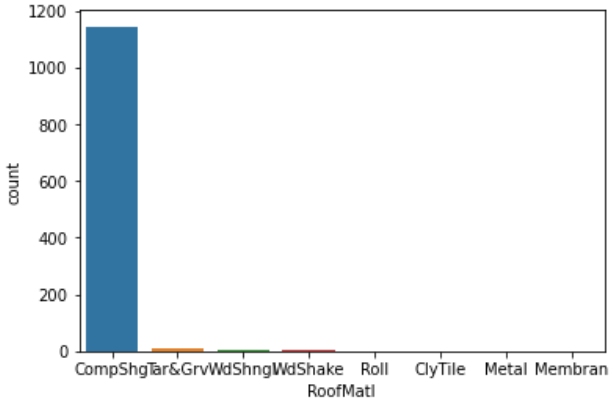

count
Norm
Feedr
RRAn
PosA
RRAe
Artery
PosN
RRNe
RRNn
Condition1

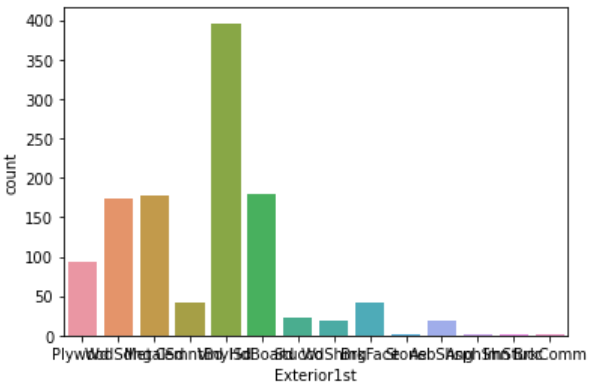

count
Norm
RRAe
Feedr
PosN
Artery
RRNn
PosA
RRAn
Condition2

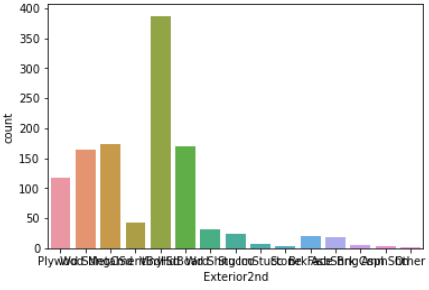

count
TwnhsE
1Fam
Duplex
Twnhs
2fmCon
BldgType

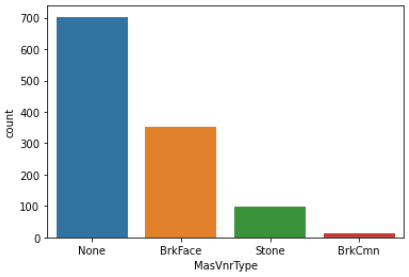

count
1Story
2Story
1.5Fin
SFoyer
1.5Unf
SLvl
2.5Fin
2.5Unf
HouseStyle

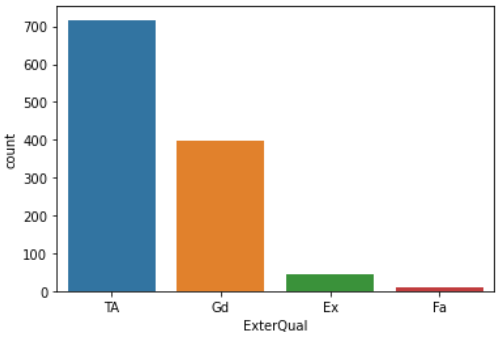

count
Gable
Flat
Hip
Shed
Gambrel
Mansard
RoofStyle

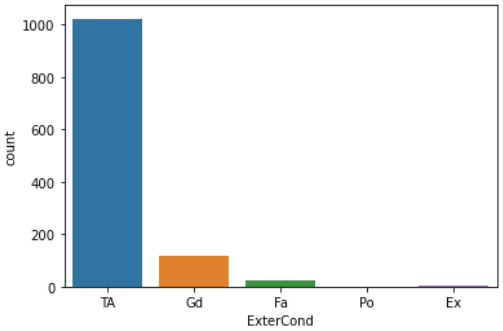

count
CompShg
Tar&Grv
WdShng
WdShake
Roll
ClyTile
Metal
Membran
RoofMatl

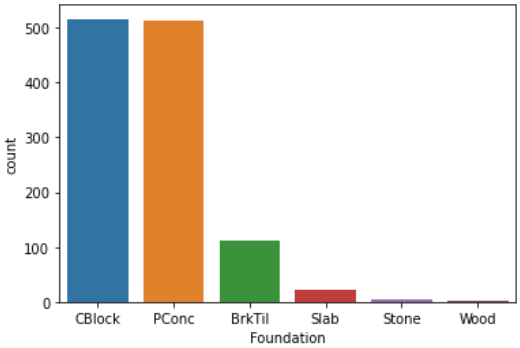

count
Exterior1st

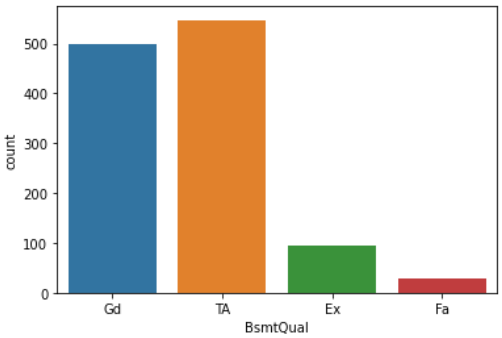

count
Exterior2nd

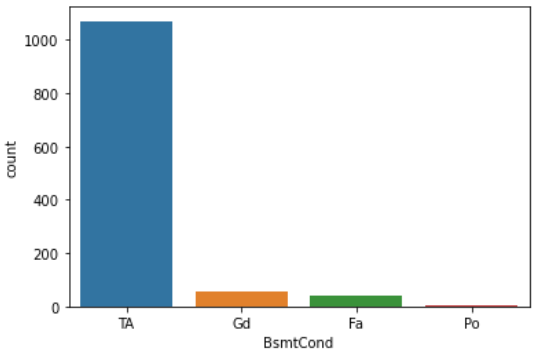

count
None
BrkFace
Stone
BrkCmn
MasVnrType

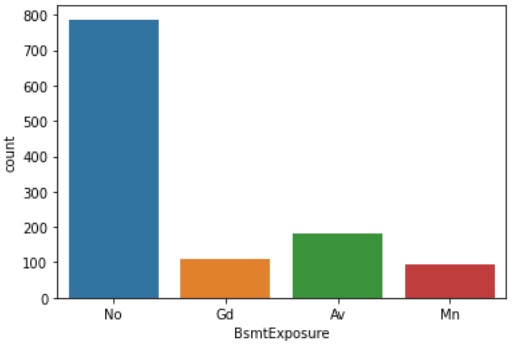

count
ExterQual
TA
Gd
Ex
Fa

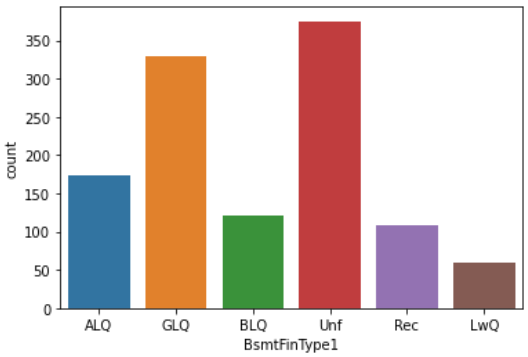

count
ExterCond
TA
Gd
Fa
Po
Ex

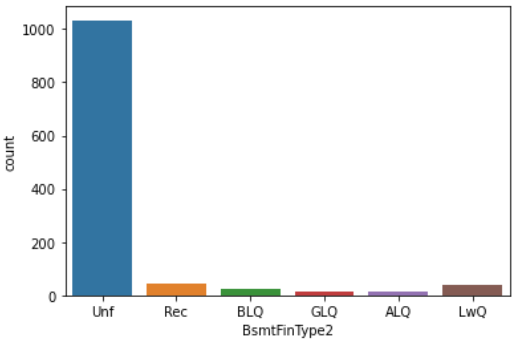

count
Foundation
CBlock
PConc
BrkTil
Slab
Stone
Wood

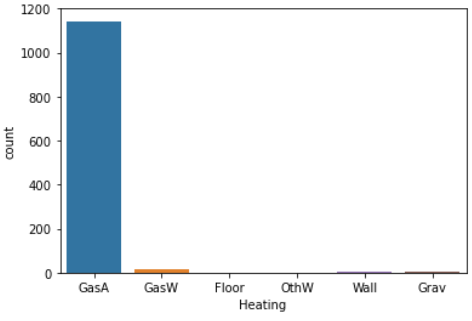

count
BsmtQual
Gd
TA
Ex
Fa

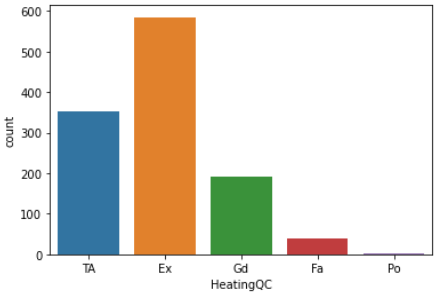

count
BsmtCond
TA
Gd
Fa
Po

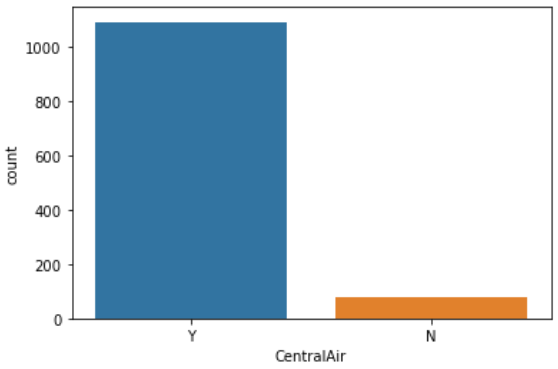

count
BsmtExposure
No
Gd
Av
Mn

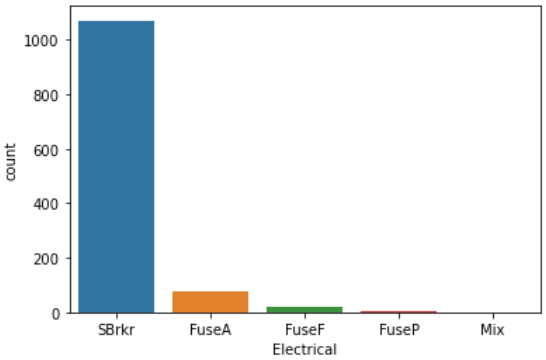

count
BsmtFinType1
ALQ
GLQ
BLQ
Unf
Rec
LwQ

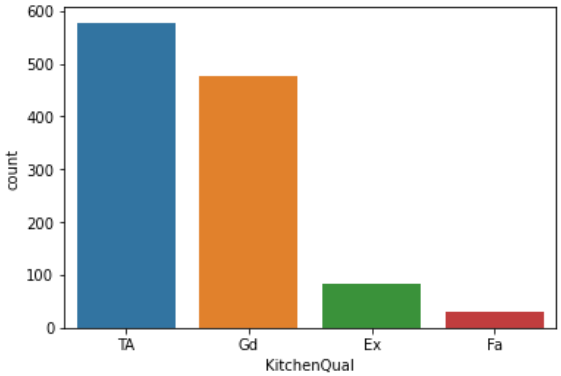

count
BsmtFinType2
Unf
Rec
BLQ
GLQ
ALQ
LwQ

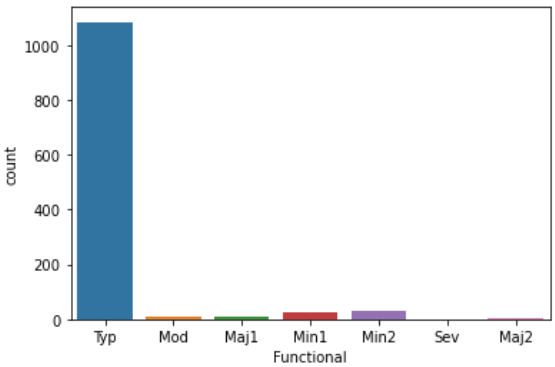

count
Heating
GasA
GasW
Floor
OthW
Wall
Grav

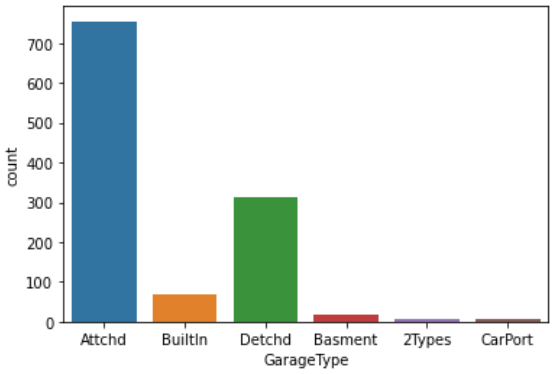

count
HeatingQC
TA
Ex
Gd
Fa
Po

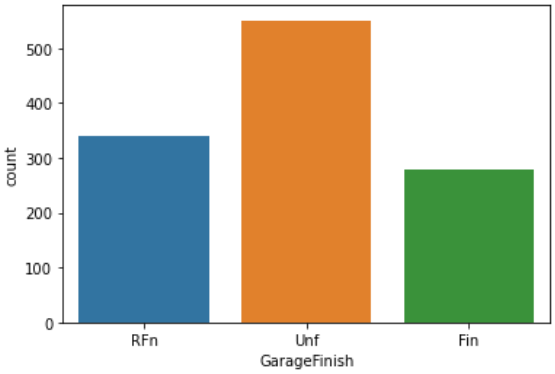

count
1000
800
600
400
200
0
Y
N
CentralAir

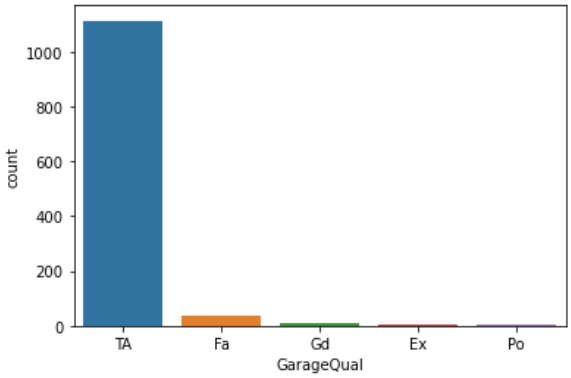

count
1000
800
600
400
200
0
SBrkr
FuseA
FuseF
FuseP
Mix
Electrical

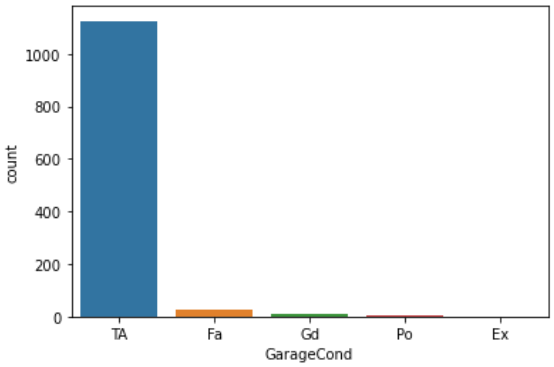

count
600
500
400
300
200
100
0
TA
Gd
Ex
Fa
KitchenQual

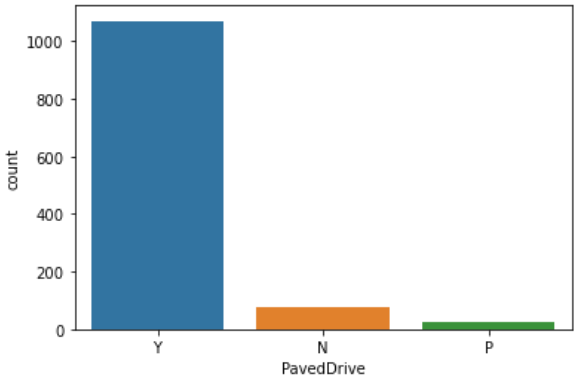

count
1000
800
600
400
200
0
Typ
Mod
Maj1
Min1
Min2
Sev
Maj2
Functional

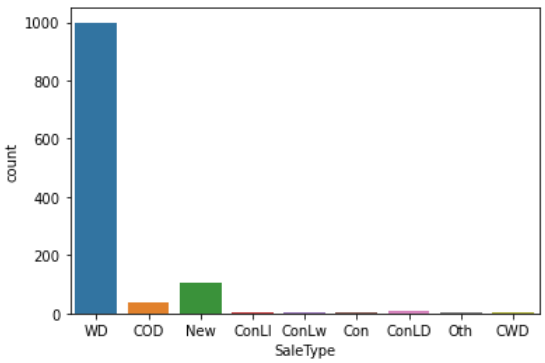

count
700
600
500
400
300
200
100
0
Attchd
BuiltIn
Detchd
Basment
2Types
CarPort
GarageType

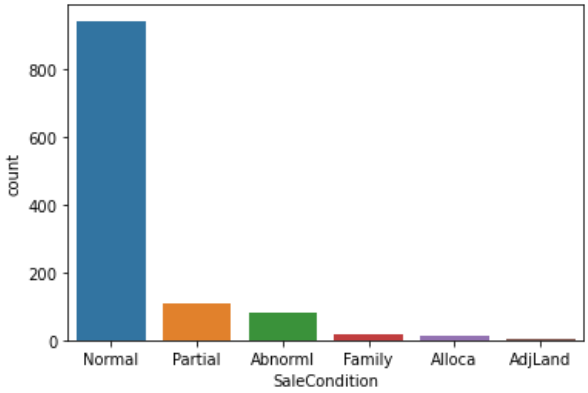

count
500
400
300
200
100
0
RFn
Unf
Fin
GarageFinish


count
1000
800
600
400
200
0
TA
Fa
Gd
Ex
Po
GarageQual


count
1000
800
600
400
200
0
TA
Fa
Gd
Po
Ex
GarageCond


count
1000
800
600
400
200
0
Y
N
P
PavedDrive


count
1000
800
600
400
200
0
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SaleType

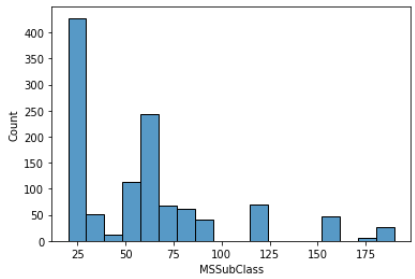


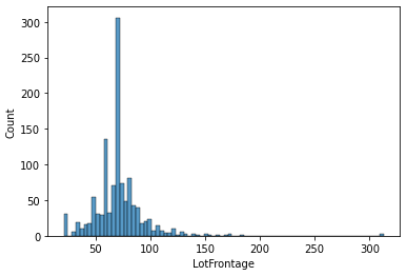
- Observation of above data.
  1. MSZoning in this column majority are Residential Low Density.
  2. street in this column majority means most of are in paved.
  3. Lotshape in this column majority are in Regular.
  4. LandContour in this column majority are in Near Flat/Level.
  5. LotConfig in this column majority are in Inside lot.
  6. LandSlope in this column majority are in Gentle slope.
  7. Condition 1 in this column majority are in Normal.
  8. Condition 2 in this column majority are in Normal.
  9. BldgType in this column majority are in Single-family Detached.
  10. HouseStyle in this column majority are in One story.
  11. Roofstyle in this column majority are in Gable.
  12. Roofmati in this column majority are in Standard (Composite) Shingle.
  13. Masvnrtype in this column majority are in None.
  14. ExterQual in this column majority are in Average/Typical.
  15. ExterCound in this column majority are in Average/Typical.
  16. Foundation in this column majority are in Cinder Block and Poured Contrete
  17. BsmtQual in this column majority are in Good (90-99 inches) and Typical (80-89 inches)
  18. Bsmtcond in this column majority are in Typical - slight dampness allowed.
  19. BsmtExposure in this column majority are in No Exposure.
  20. Bsmtfintype1 in this column majority are in Good Living Quarters and Unfinshed.
  21. Bsmtfintype2 in this column majority are in Unfinshed.
  22. Heating in this column majority are in Gas forced warm air furnace.
  23. HeatingQc in this column majority are in Excellent.
  24. Central air in this column majority are has central air.
  25. Electrical in this column majority are in Standard Circuit Breakers & Romex.
  26. KitchenQual in this column majority are in Typical/Average.
  27. Functional in this column majority are in Typical Functionality.
  28. Garagetype in this column majority are has Attached to home garagetype.
  29. Gragefinish in this column majority are in Unfinished.
  30. GrageQual in this column majority are in Typical/Average garagequal.
  31. GarageCond in this column majority are in Typical/Average.
  32. PavedDrive in this column majority are has Paved drive.
  33. Sale type in this column majority are in Warranty Deed - Conventional.
  34. salecondition in this column majority are in Normal Sale condition.

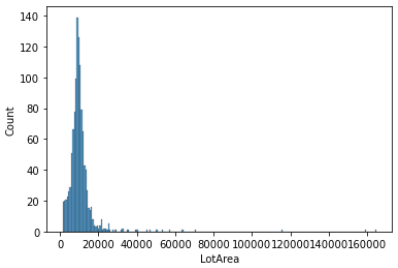
- Now we observe for numerical data for numerical type data we use hist plot by observing that graph we observe some observation.

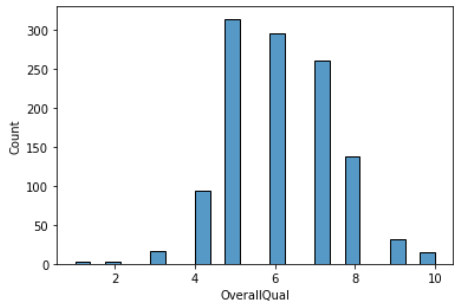
```
In [72]: for i in numerical:
             sns.histplot(df[i])
             plt.show()
```

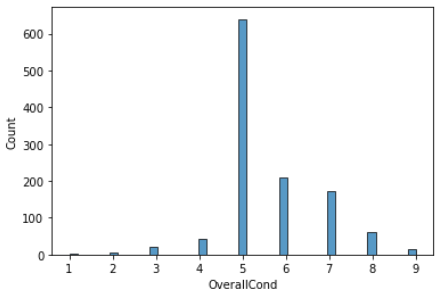
- By use of this for loop we plot histplot.

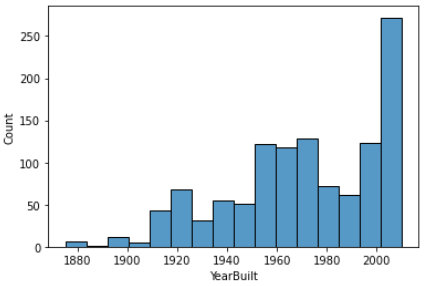


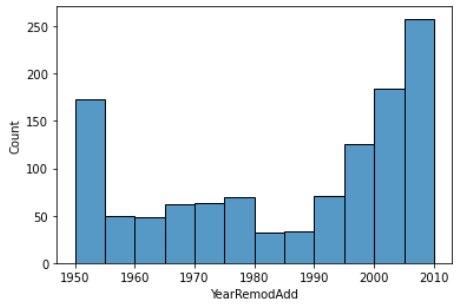


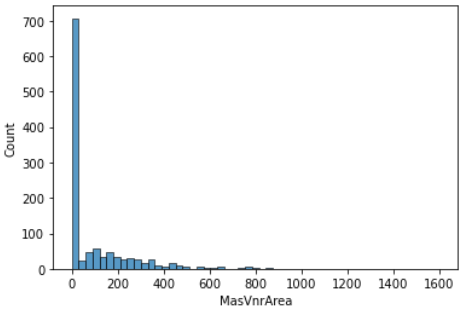


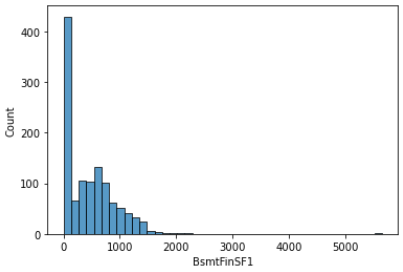


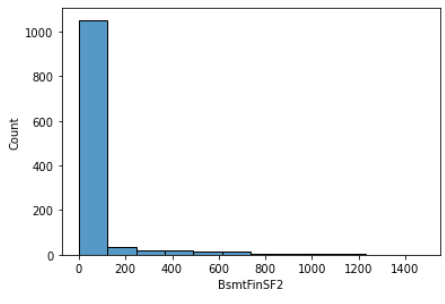


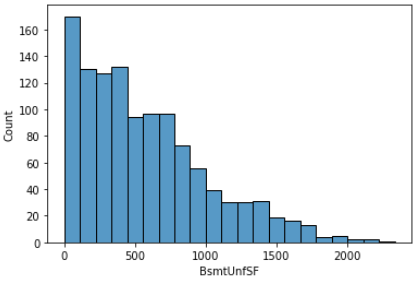


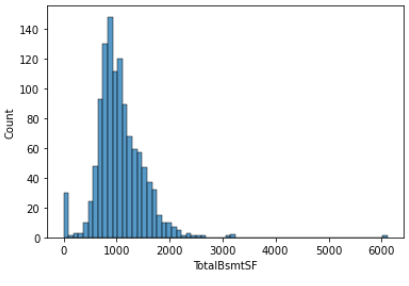


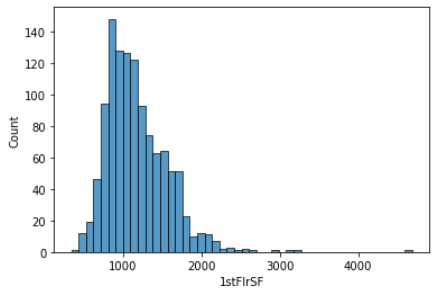


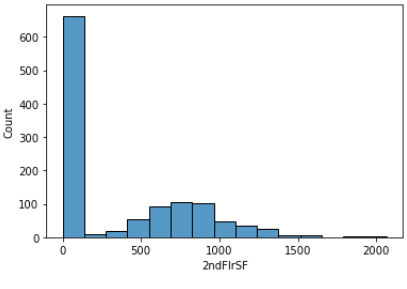


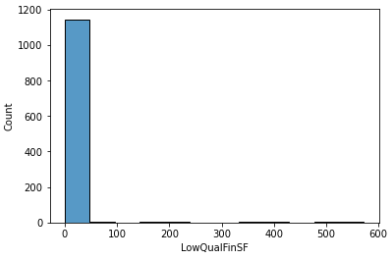


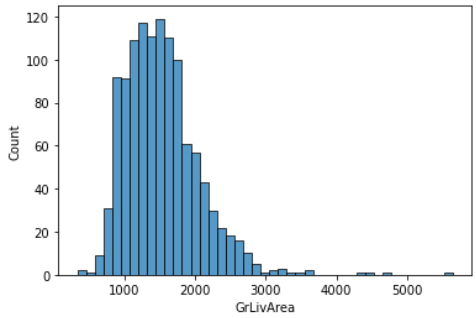


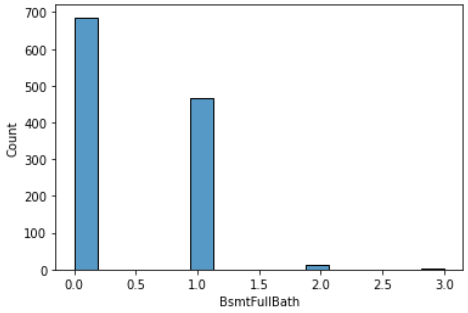


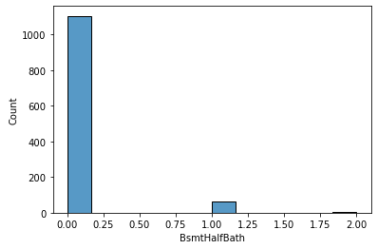


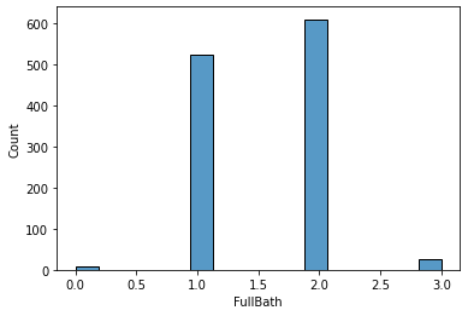


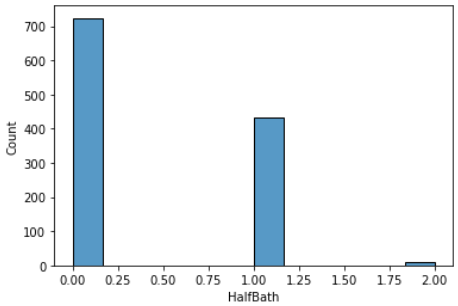


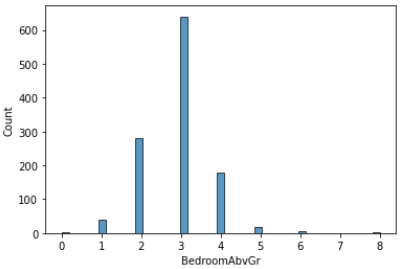

Count
GrLivArea

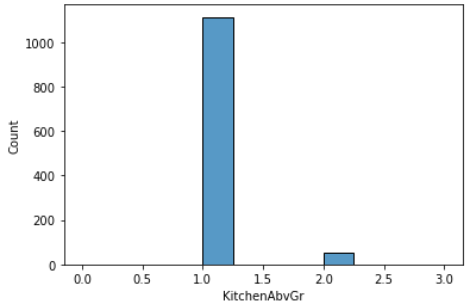

Count
BsmtFullBath

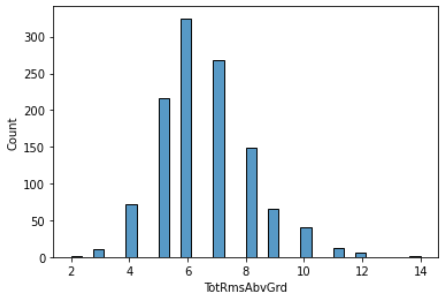

Count
BsmtHalfBath

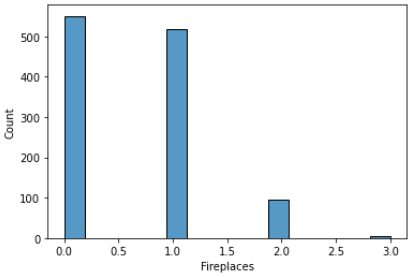

Count
FullBath

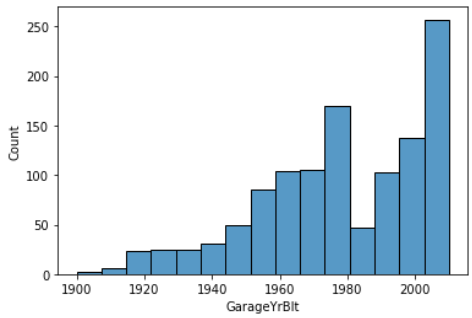

Count
HalfBath

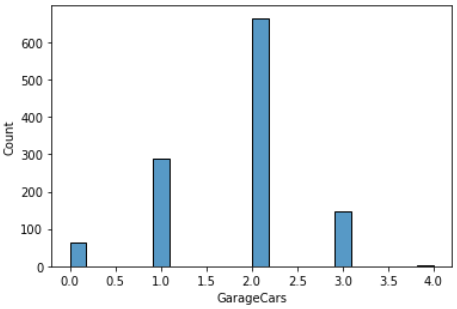

Count
BedroomAbvGr

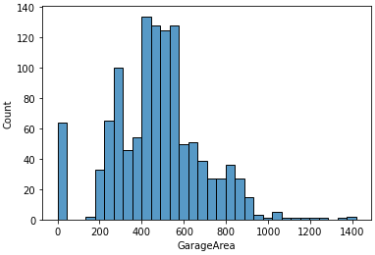

Count
KitchenAbvGr

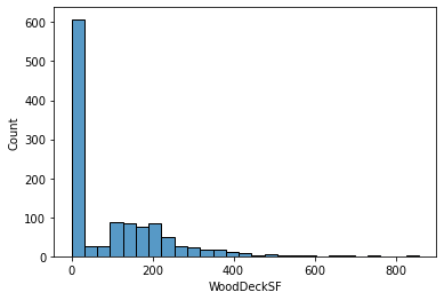

Count
TotRmsAbvGrd

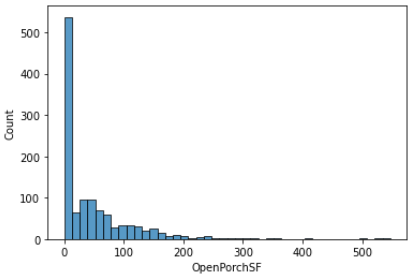

Count
Fireplaces

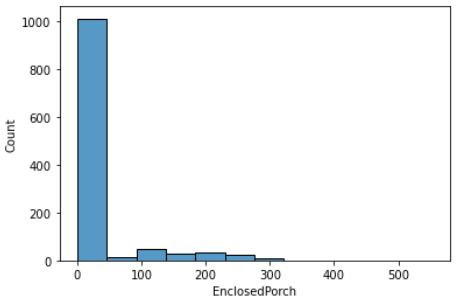

Count
GarageYrBlt

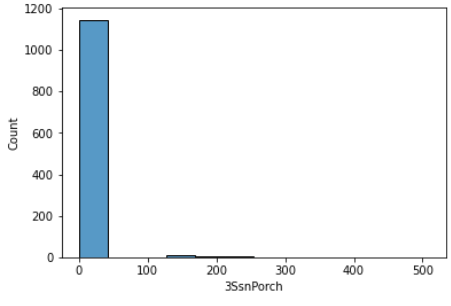

Count
GarageCars

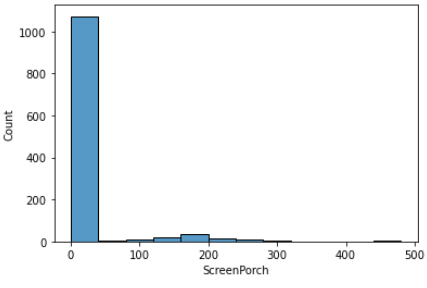

Count
GarageArea

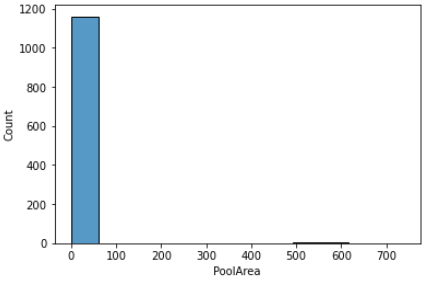

Count
WoodDeckSF

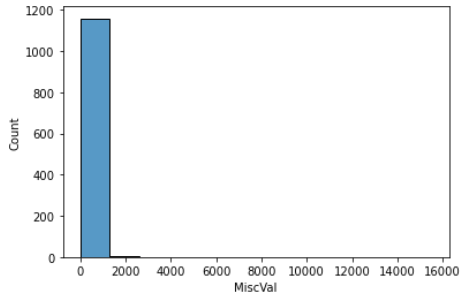

Count
OpenPorchSF

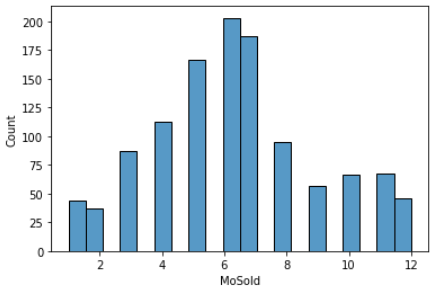

Count
EnclosedPorch

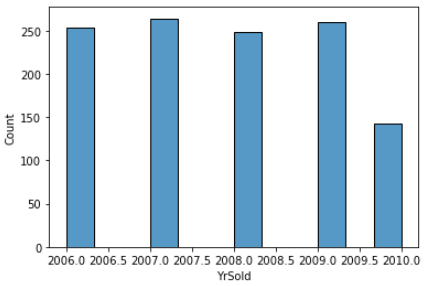

Count
3SsnPorch

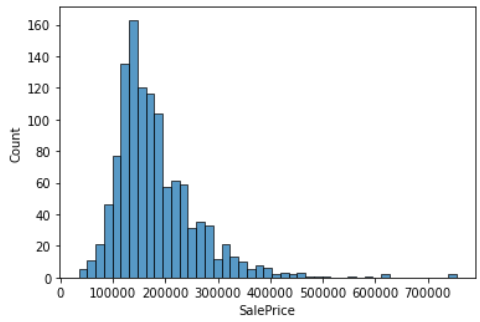

Count
ScreenPorch


Count
PoolArea









- Now we observe all the observation of data.

  1. MSSubClass as we observe that in this columns majority data is in 1-STORY 1946 & NEWER ALL STYLES,1-STORY 1945 & OLDER
  2. Lotfrontage in this column majority data is between 65 to 80.
  3. Lotarea majority are is less than 10000.
  4. OverallQual in this column majority are in Good,Above Average,Average Below, Average.
  5. Overallcond in this column majority are in 5 means Average.
  6. Yearbuilt in this column majority are in 1990 to 2000.
  7. Yearremondadd in this column majority are in 2000 to 2010.
  8. MasVnrarea in this column majortiy are less than 200.
  9. BsmtFinSF1 in this column majority are in less than 800.
  10. BsmtFinSF2 in this column majority are in less than 100.
  11. BsmtUnfSF in this column majority are in less than 500.
  12. TotalBsmtSF in this column majority are around 1000.
  13. 1stfirsf in this column majority are in 900 to 1200.
  14. 2ndfirsf in this column majority are less than 100.
  15. LowQualFinSF in this column majority are less than 50.
  16. GrLivearea in this column majority are 1000 to 2000.
  17. BsmtfullBath in this column majority are 0 and 1 around.
  18. BsmthalfBath in this column majority are 0 and 1 around.
  19. FullBath in this column majority are 1 and 2.
  20. HalfBath in this column majority are 0 and 1 around.
  21. BedroomAbvGr in this column majority are in 3.
  22. KitchenAbvGr in this column majority are in 1 and 2.
  23. TotRmsAbvGrd in this column majority are in 6 and 7.
  24. Fireplaces in this column majority are in 0 and 1.
  25. Grageyrbuilt in this column majority are in 2010 and 1970.
  26. GarageCars in this column majority are in 2.
  27. Garagearea in this column majority are in 400 to 600.
  28. WbodDeskSF in this column majority are in 0 to 100.
  29. OpenPorchSF in this column majority are in 0 to 50.
  30. Enclosedporch in this column majority are in 0 to 50.
  31. 3SsnPorch in this column majority are in less than 50.
  32. ScreenPorch in this column majority are in less than 50.
  32. Poolarea in this column majority are in less than 50.
  33. Miskval in this column majority are in less than 1800.

34. Mosold in this column majority are in 6 to 7.
35. Yrsold in this column majority are in 2006 to 2009.
36. Saleprice in this column majority are in between 100000 to 200000.

- Now as we above observed our data is in object and numerical type data for convert it into numerical type data we use label encoding technique to convert it.

In [73]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

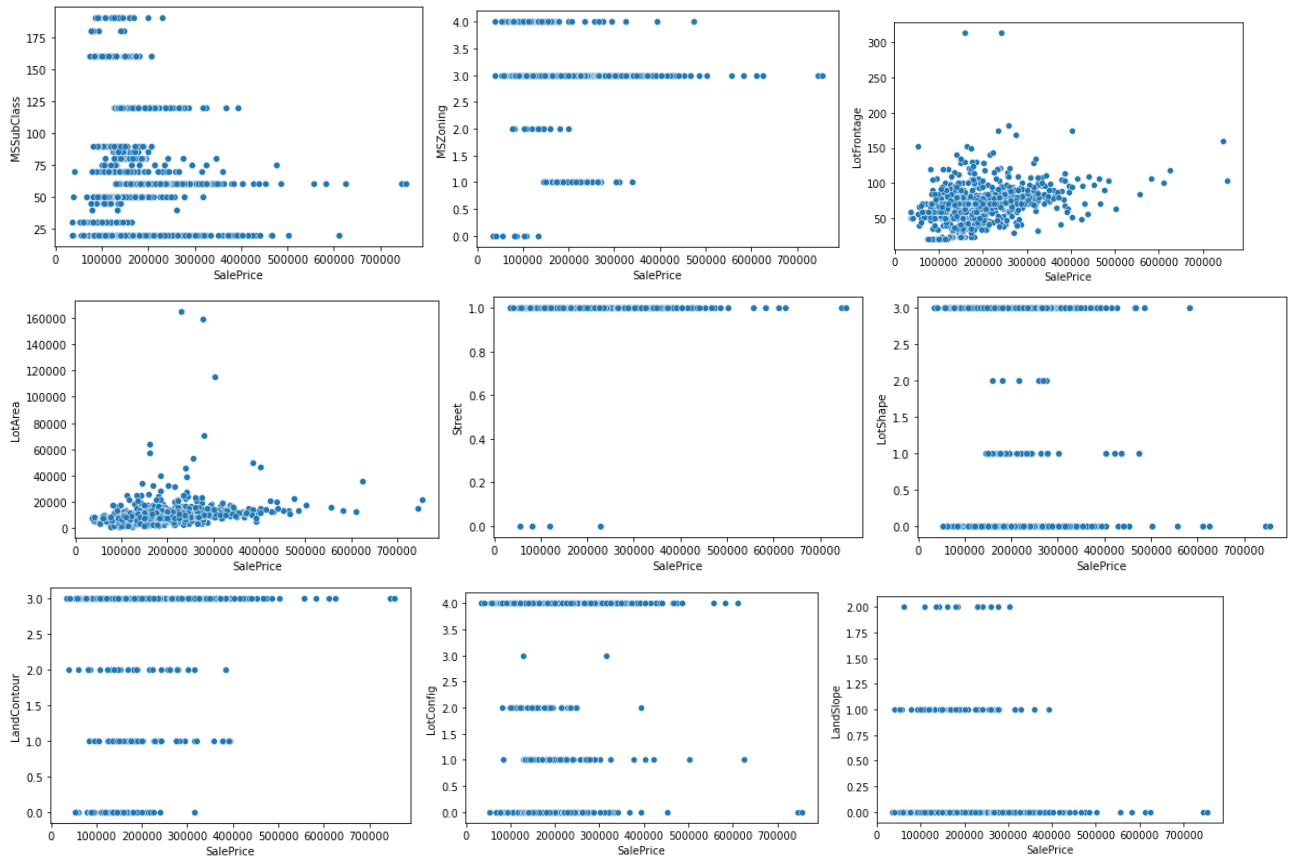
- We should convert only object type data for that we use for loop.

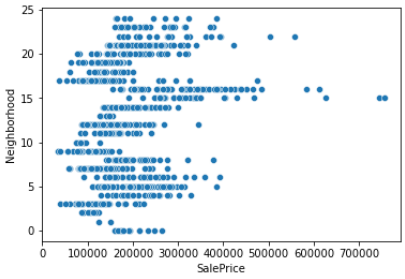
In [75]:
```
for i in objects:
    df[i]=le.fit_transform(df[i])
```

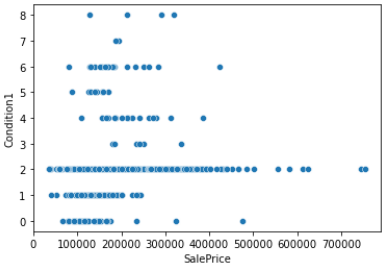
- Now we observe our data is converted or not for check it we use another for loop.

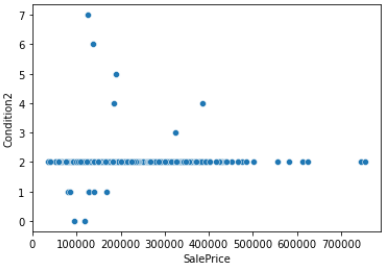
In [76]:
```
mz=['MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'LotShape', 'LandContour', 'LotConfig', 'LandSlope', 'Neighborhood',
       'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'OverallQual',
       'OverallCond', 'YearBuilt', 'YearRemodAdd', 'RoofStyle', 'RoofMatl',
       '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', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal',
       'MoSold', 'YrSold', 'SaleType', 'SaleCondition', 'SalePrice']
for i in mz:
    print(str(i),'=',df[i].dtypes)
```

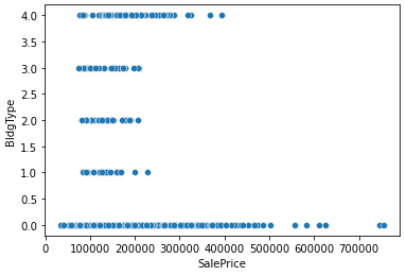
- By observing this for loop output we observe that our data is now converted into numerical type.
- Now we plot Bivariate analysis by each column for that we use scatter plot and bar plot. By observing those we conclude some results.

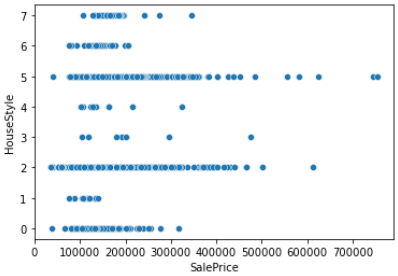


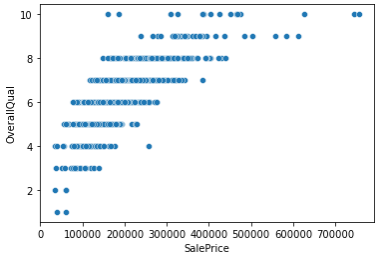

Neighborhood
SalePrice

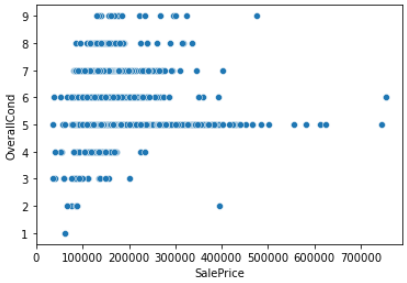

Condition1
SalePrice

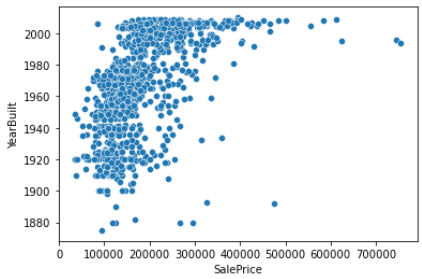

Condition2
SalePrice

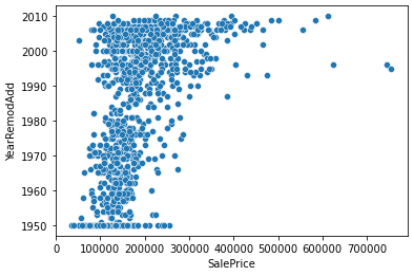

BldgType
SalePrice

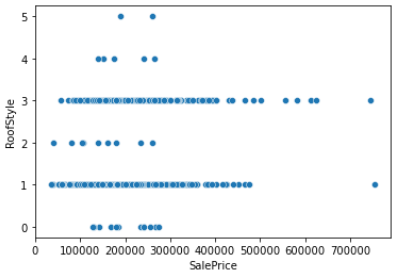

HouseStyle
SalePrice

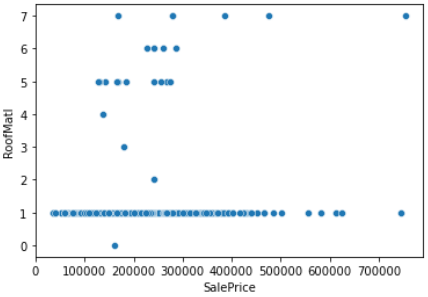

OverallQual
SalePrice

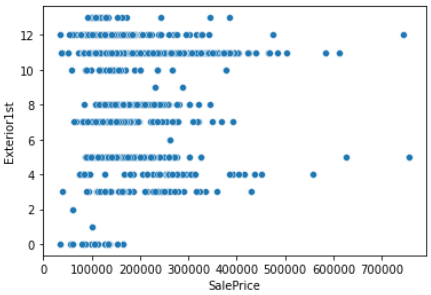

OverallCond
SalePrice

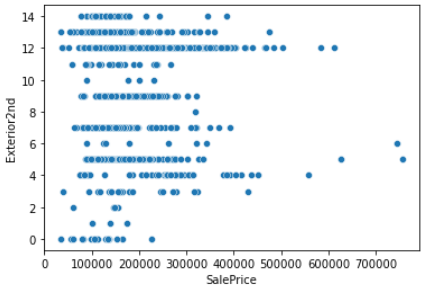

YearBuilt
SalePrice

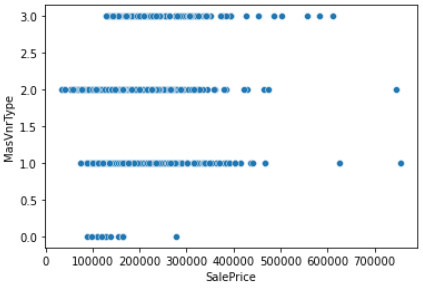

YearRemodAdd
SalePrice

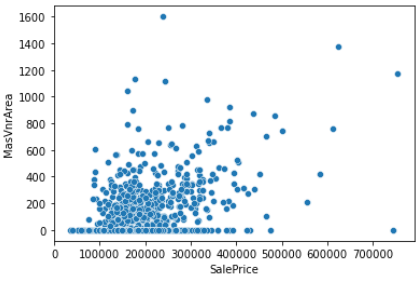

RoofStyle
SalePrice

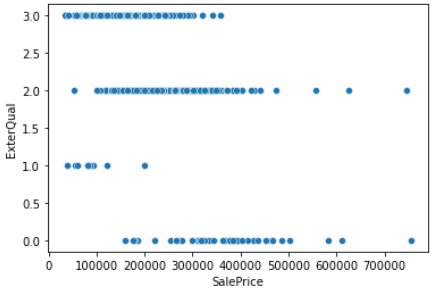

RoofMatl
SalePrice

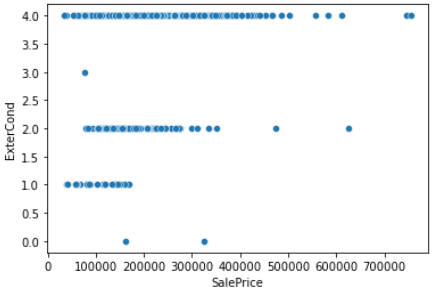

Exterior1st
SalePrice

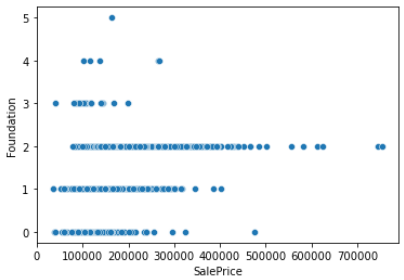

Exterior2nd
SalePrice

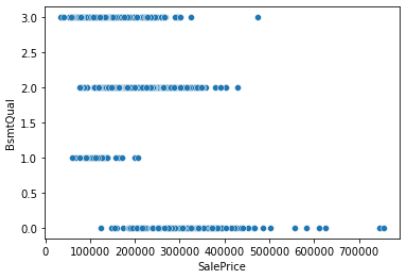

MasVnrType
SalePrice

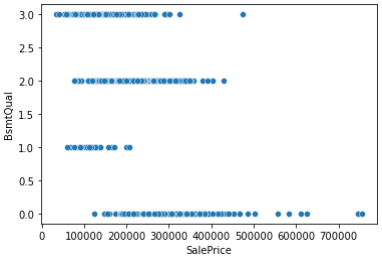

MasVnrArea
SalePrice

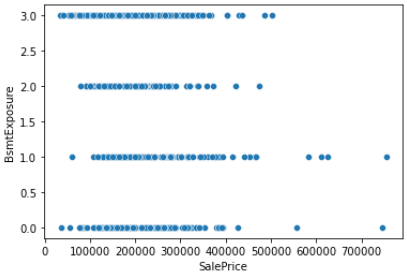

ExterQual
SalePrice

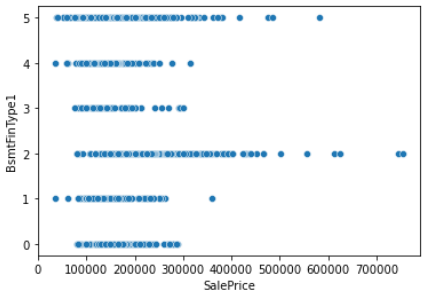

ExterCond
SalePrice

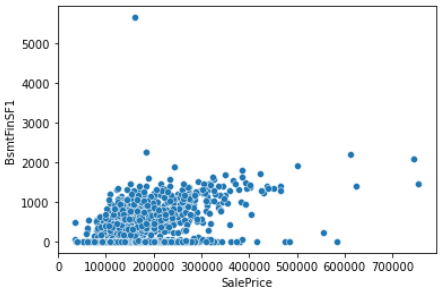

Foundation
SalePrice

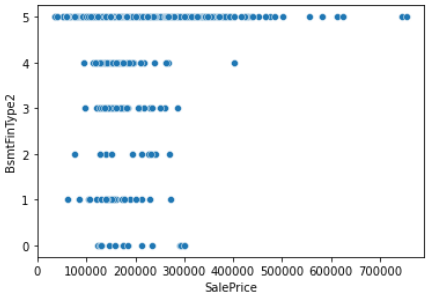

BsmtQual
SalePrice

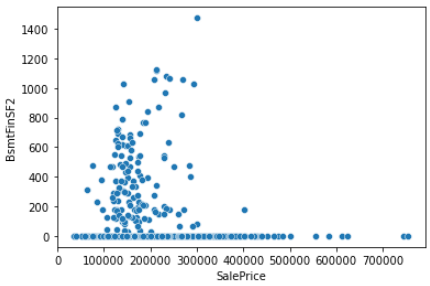

BsmtQual
SalePrice

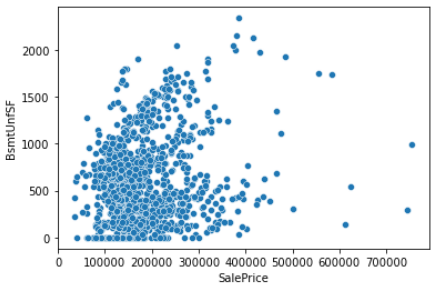

BsmtExposure
SalePrice

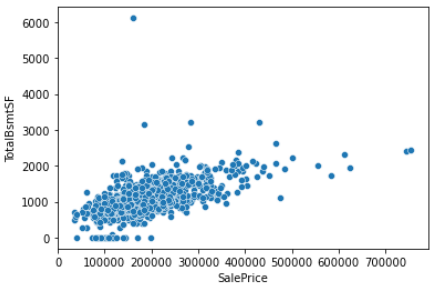

BsmtFinType1
SalePrice

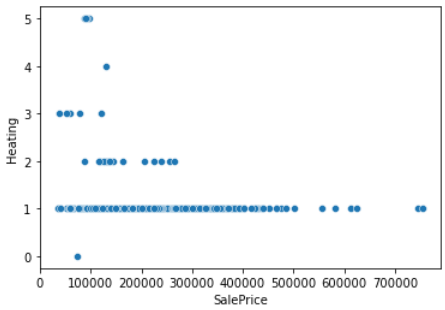

BsmtFinSF1
SalePrice

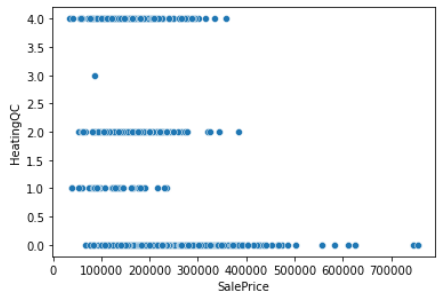

BsmtFinType2
SalePrice

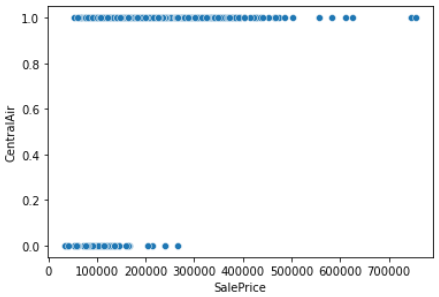

BsmtFinSF2
SalePrice

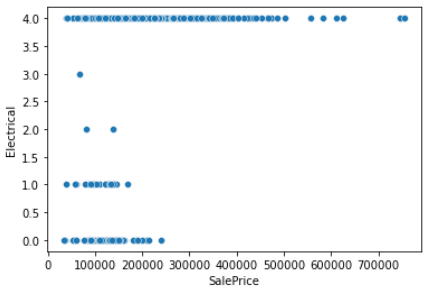

BsmtUnfSF
SalePrice

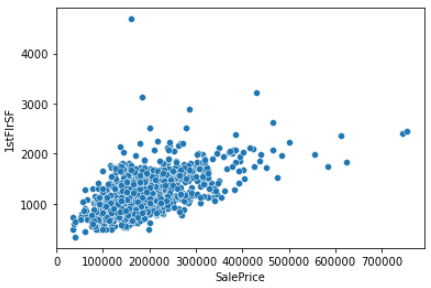

TotalBsmtSF
SalePrice

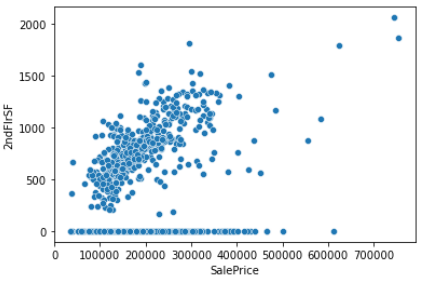

Heating
SalePrice

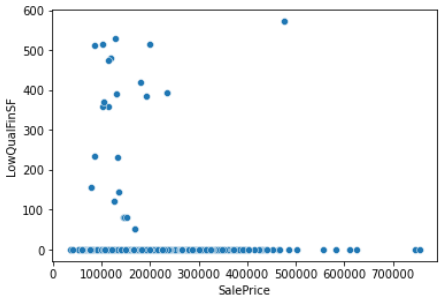

HeatingQC
SalePrice

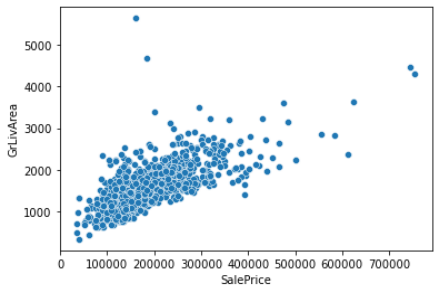

CentralAir
SalePrice

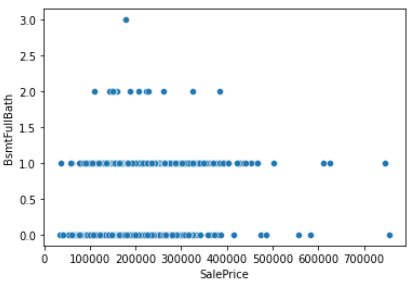

Electrical
SalePrice

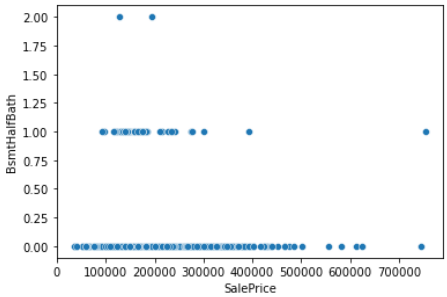

1stFlrSF
SalePrice

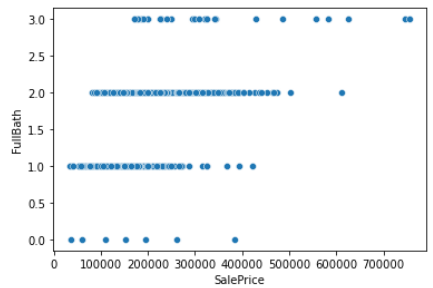

2ndFlrSF
SalePrice

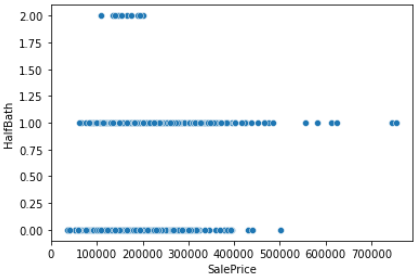

LowQualFinSF
SalePrice

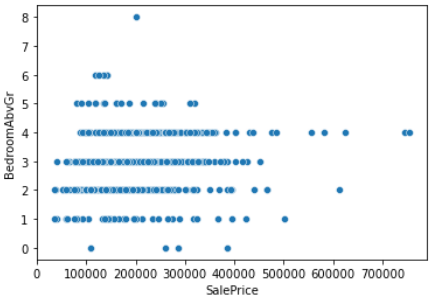

GrLivArea
SalePrice

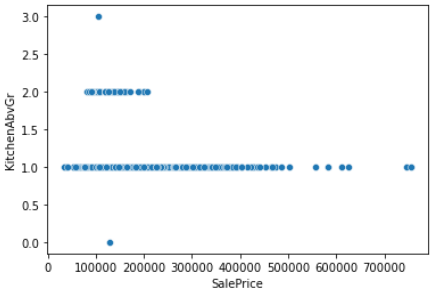

BsmtFullBath
SalePrice

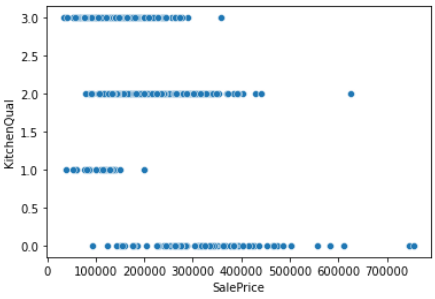

BsmtHalfBath
SalePrice

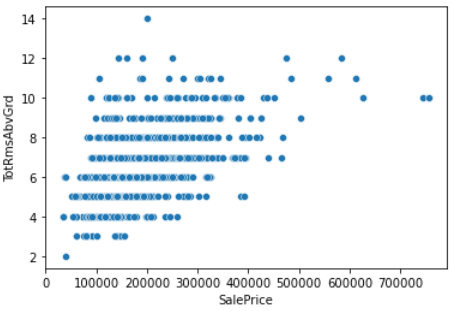

FullBath
SalePrice

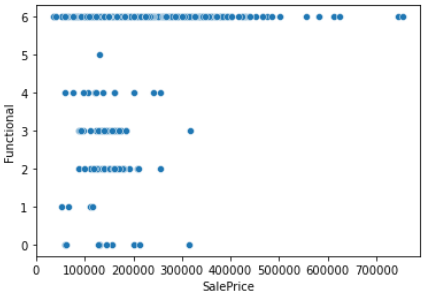

HalfBath
SalePrice

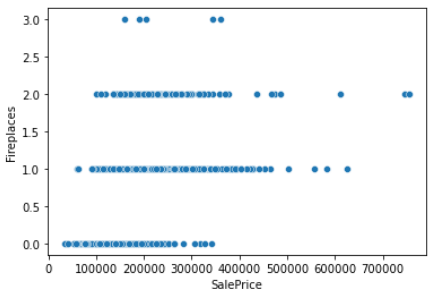

BedroomAbvGr
SalePrice

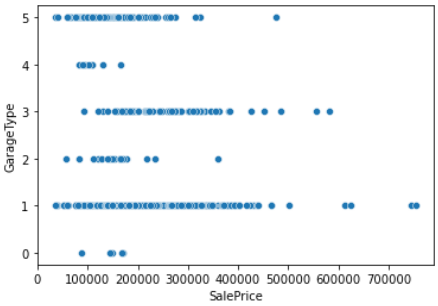

KitchenAbvGr
SalePrice

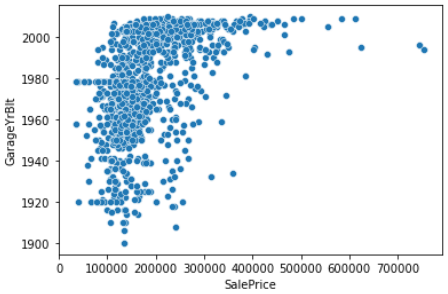

KitchenQual
SalePrice

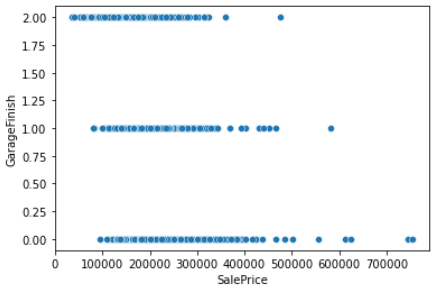

TotRmsAbvGrd
SalePrice

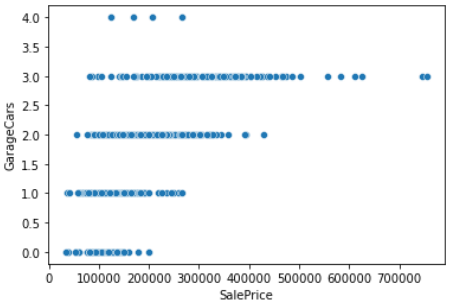

Functional
SalePrice

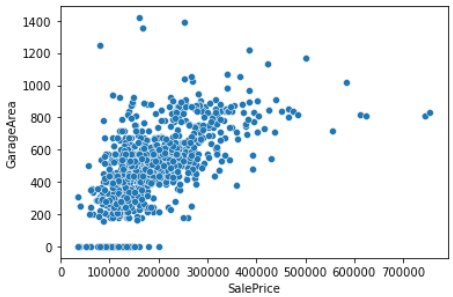

Fireplaces
SalePrice

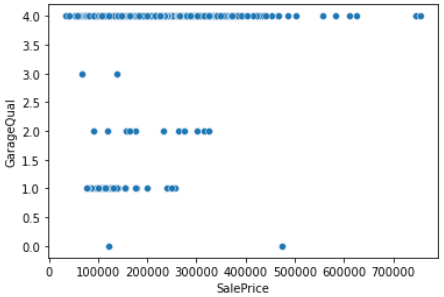

GarageType
SalePrice

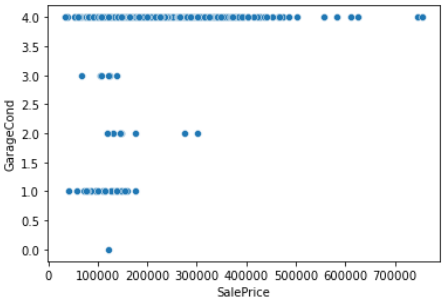

GarageYrBlt
SalePrice

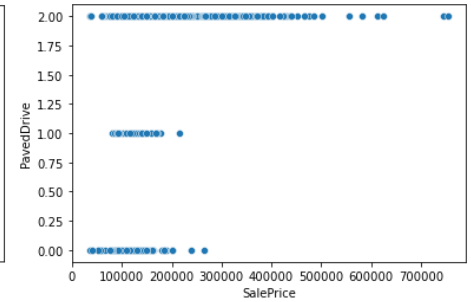

GarageFinish
SalePrice

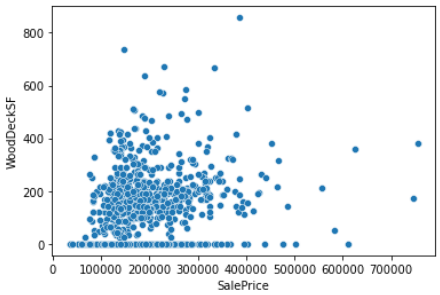

GarageCars
SalePrice

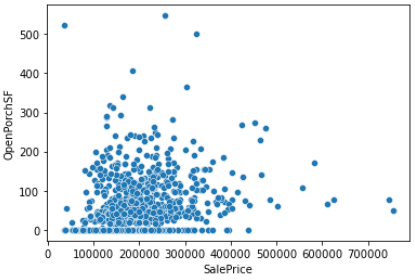

GarageArea
SalePrice

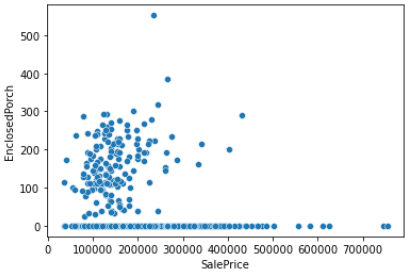

GarageQual
SalePrice

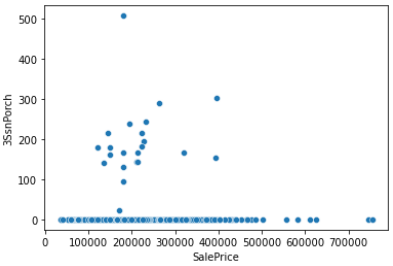

GarageCond
SalePrice

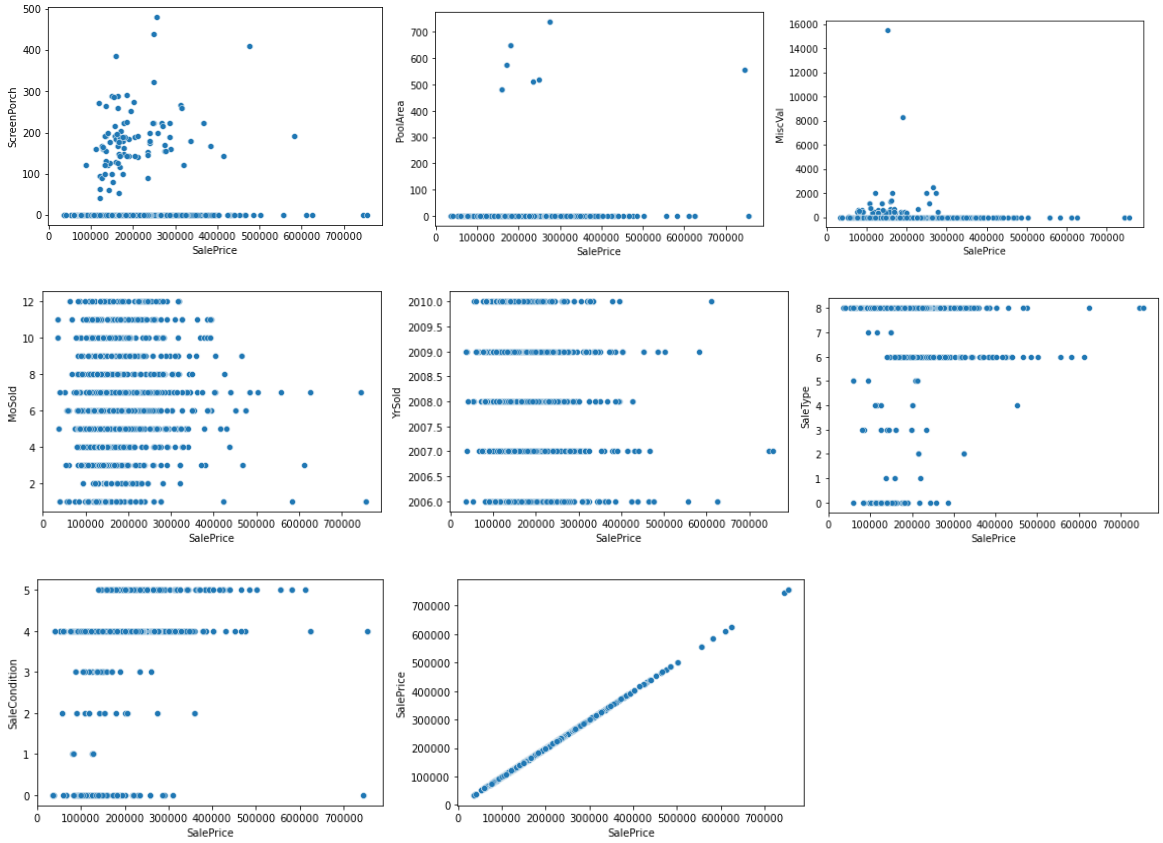

PavedDrive
SalePrice


WoodDeckSF
SalePrice

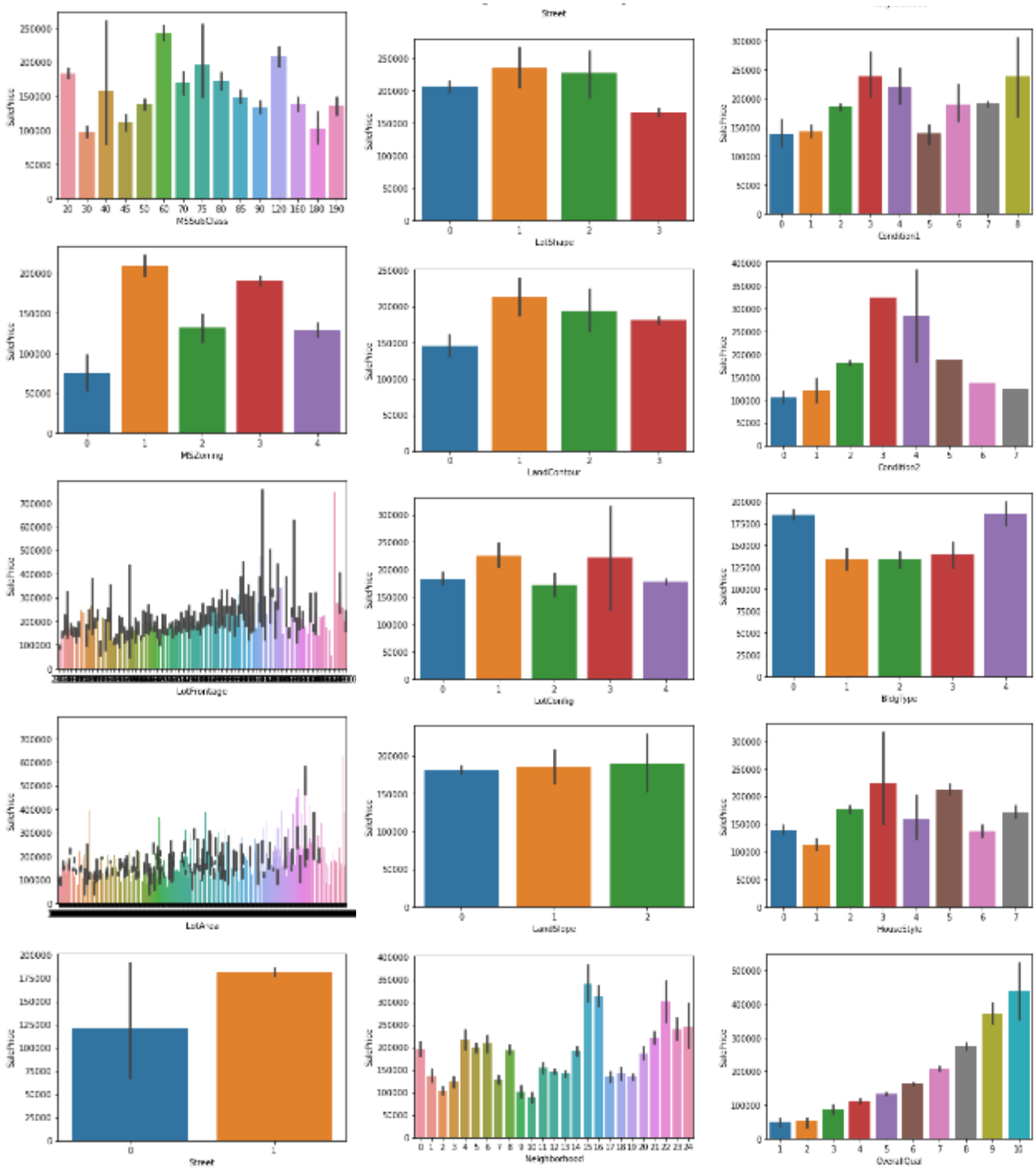

OpenPorchSF
SalePrice

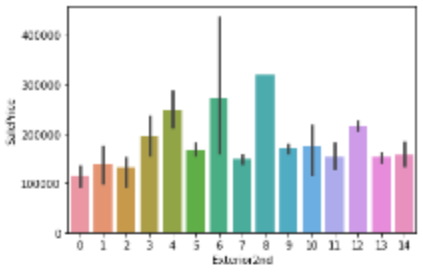

EnclosedPorch
SalePrice

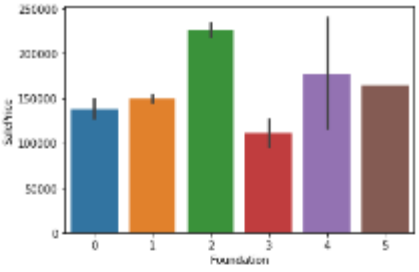

3SsnPorch
SalePrice

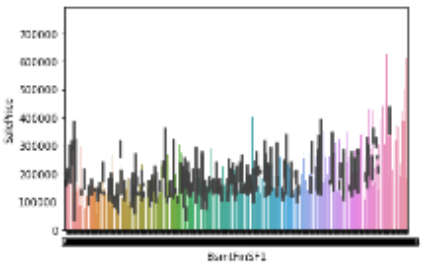


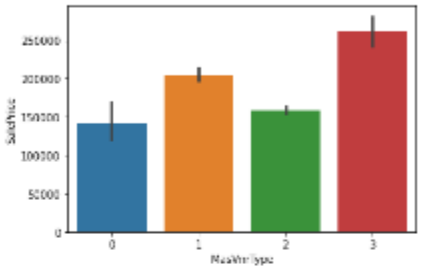
- Now we observe bar plot.

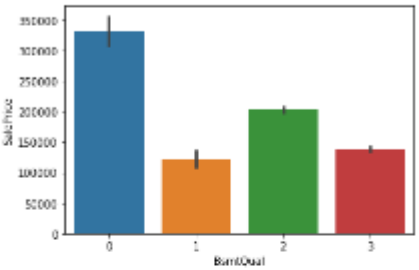

SalePrice
MSSubClass
Street
LotShape
Condition1
MSZoning
LandContour
Condition2
LotFrontage
LotConfig
BldgType
LotArea
LandSlope
HouseStyle
Street
Neighborhood
OverallQual
OverallCond
YearBuilt
RoofStyle
RoofMatl
ExterQual

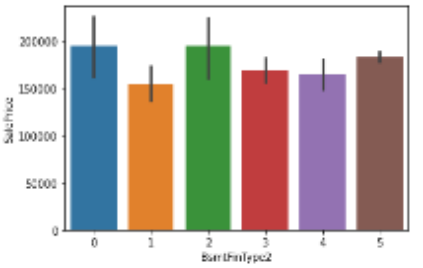

SalePrice
Exterior2nd

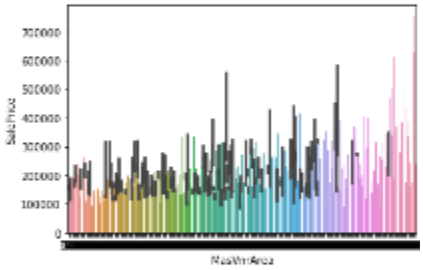

SalePrice
Foundation

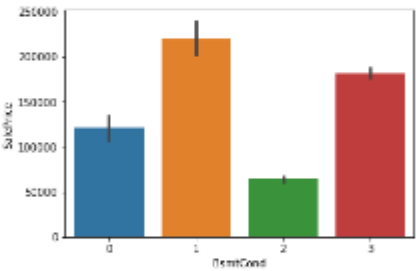

SalePrice
BsmtUnfSF1

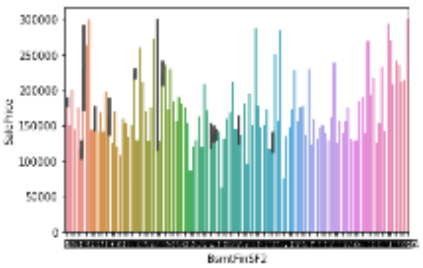

SalePrice
MasVnrType

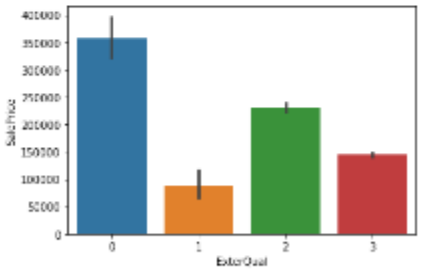

SalePrice
BsmtQual

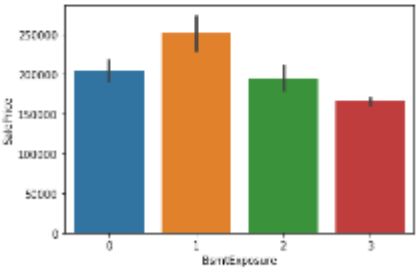

SalePrice
BsmtFinType2

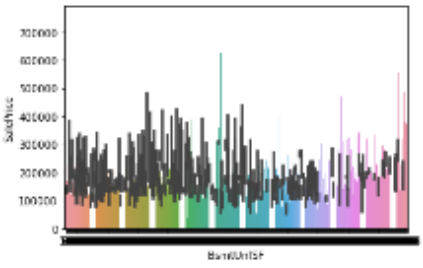

SalePrice
MasVnrArea

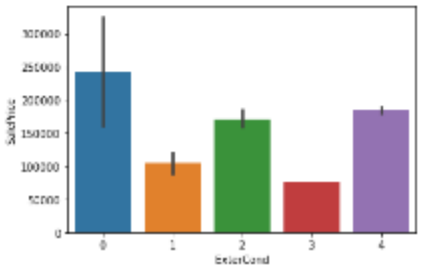

SalePrice
BsmtCond

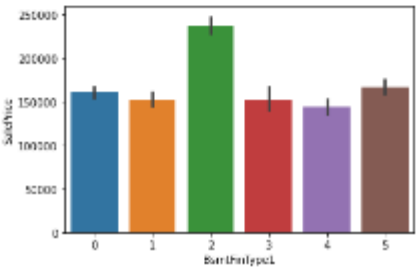

SalePrice
BsmtFinSF2

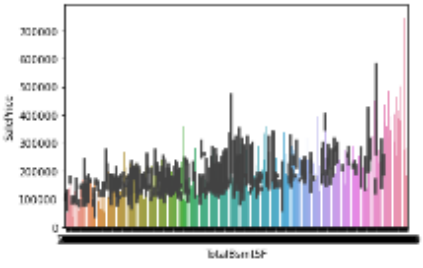

SalePrice
ExterQual

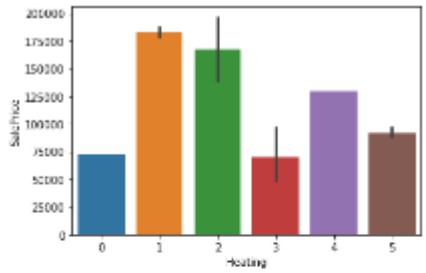

SalePrice
BsmtExposure

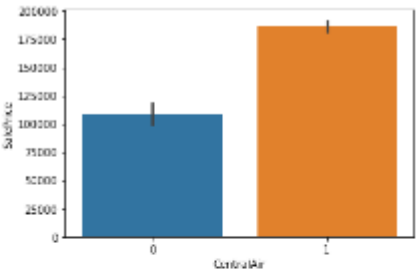

SalePrice
BsmtUnfSF

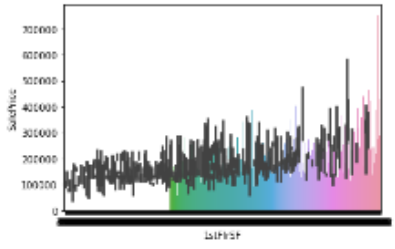

SalePrice
ExterCond

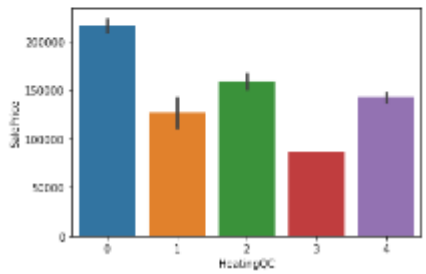

SalePrice
BsmtFinType1

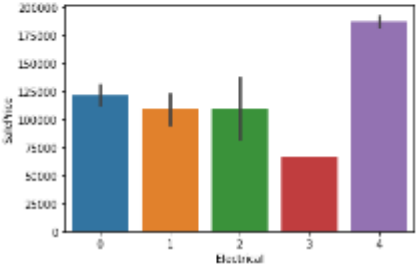

SalePrice
TotalBsmtSF

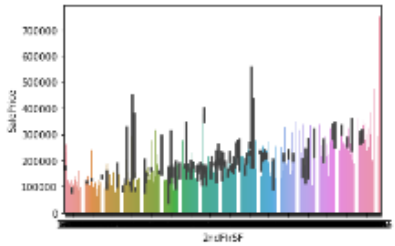

SalePrice
Heating

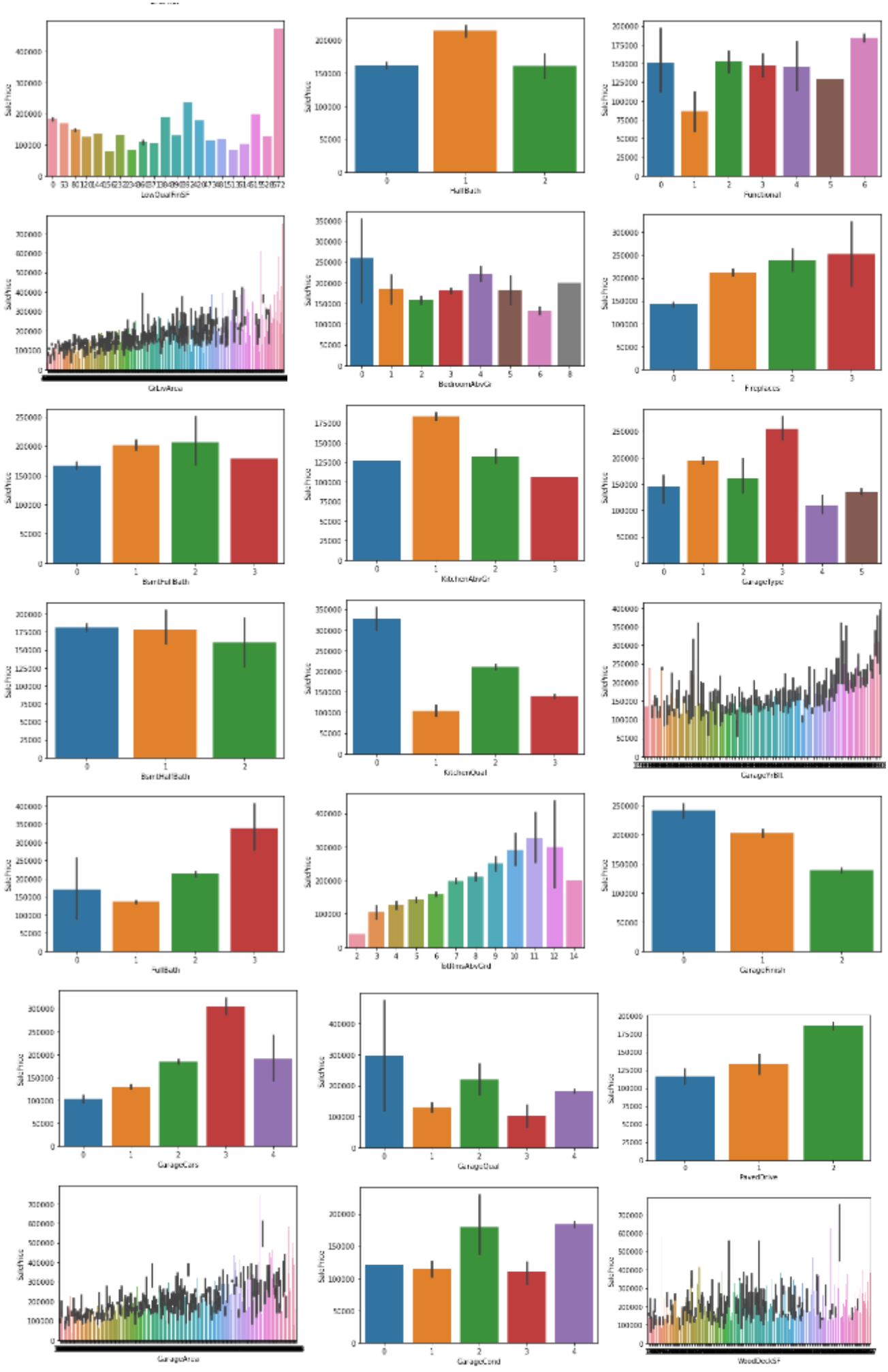

SalePrice
CentralAir

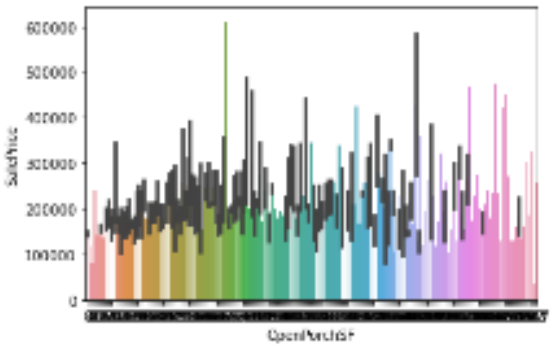

SalePrice
1stFlrSF

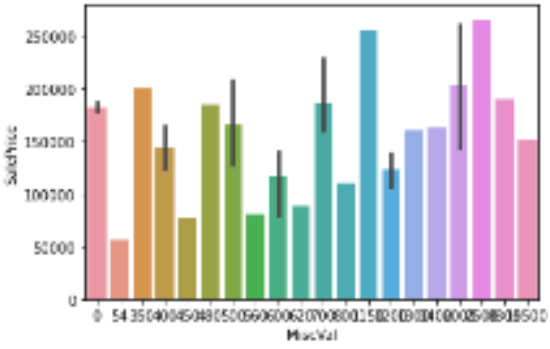

SalePrice
HeatingQC

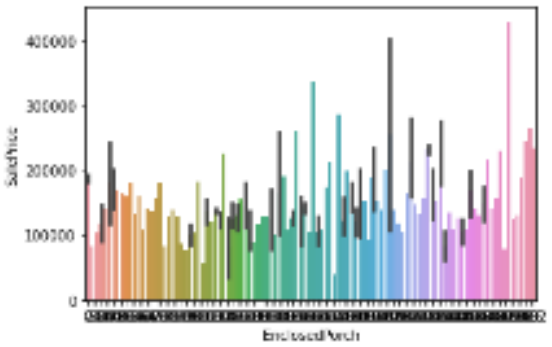

SalePrice
ElectricAl

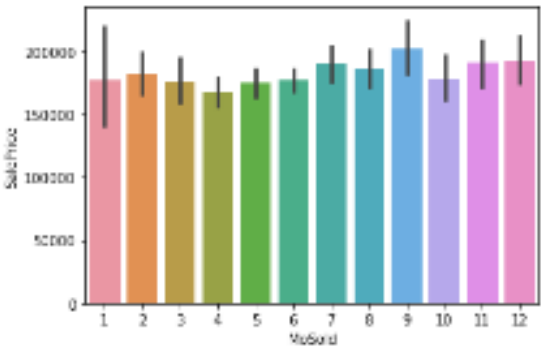

SalePrice
2ndFlrSF

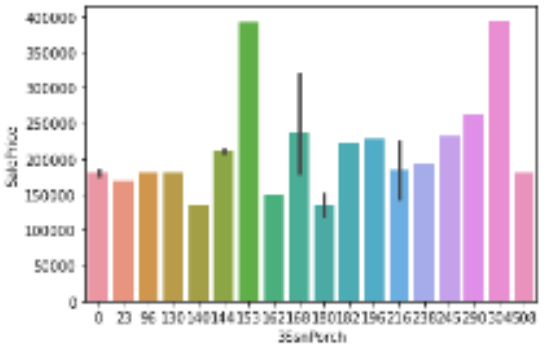

SalePrice
LowQualFinSF
FullBath
Functional
GrLivArea
BedroomAbvGr
Fireplaces
BsmtFullBath
KitchenAbvGr
GarageType
BsmtHalfBath
KitchenQual
GarageYrBlt
FullBath
TotRmsAbvGrd
GarageFinish
GarageCars
GarageQual
PavedDrive
GarageArea
GarageCond
WoodDeckSF

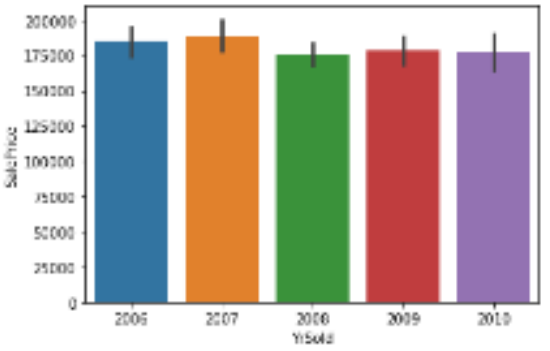

SalePrice
OpenPorchSF

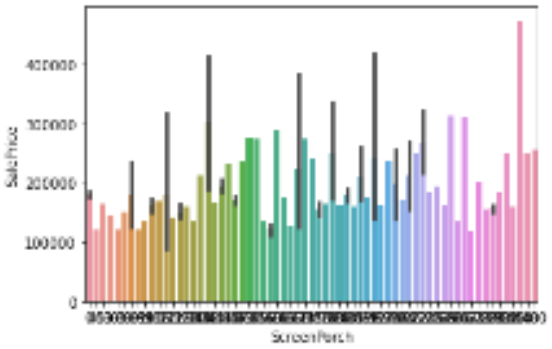

SalePrice
MiscVal

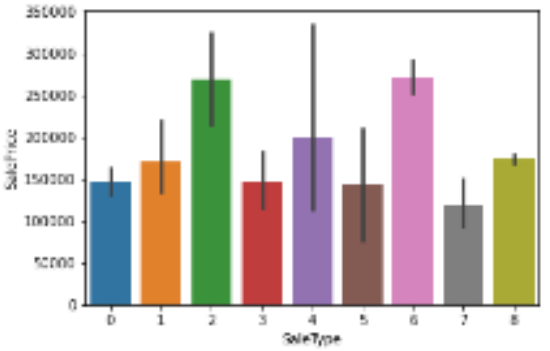

SalePrice
EnclosedPorch

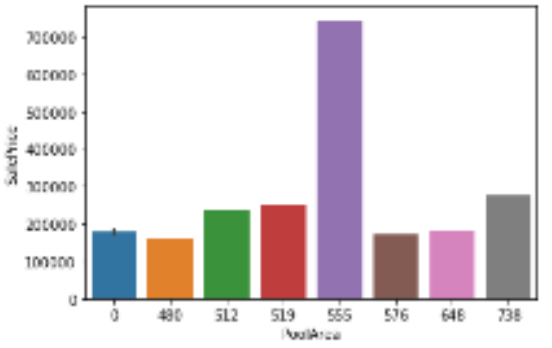

SalePrice
MoSold

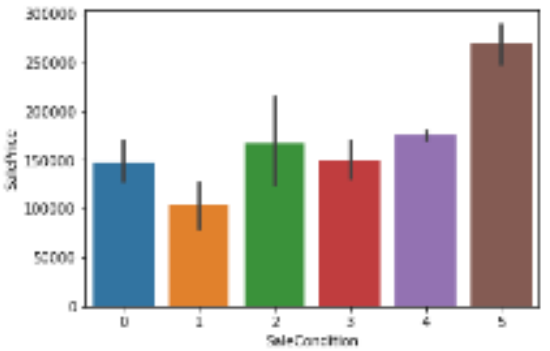

SalePrice
3SsnPorch


SalePrice
YrSold


SalePrice
ScreenPorch


SalePrice
SaleType


SalePrice
PoolArea


SalePrice
SaleCondition

- Now we observe all the graph and getting some output.

1. Sales price vs Yrsold in every year sale price is approx 175000.
2. Sales price vs MsSubclass in 60 and 120 class having higher sale price.
3. Sales price vs MsZoning in 1 and 3 zone has very high price.
4. Sales price vs Lotfrontage
5. Sales price vs LotArea
6. Sales price vs street in 1 means yes street having high sale price.
7. Sales price vs Lotspace in 1 and 2 Lotspace having high price.
8. Sales price vs LandContour in 1 and 2 having high price.
9. Sales price vs LotConfig in 1 and 3 having high price.
10. Sales price vs LandSlope this is not mendatory for sale price.
11. Sales price vs Neighbourhood 15,16,22 neighbourhood having high sale price.
12. Sales price vs Condition1 3 and 8 having high sale price.
13. Sales price vs Condition2 3 and 4 having high sale price.
14. Sales price vs BldgType in 0 and 4 having high sale price.
15. Sales price vs HouseStyle in 3 and 5 is having high sale price.
16. Sales price vs Overall by incerease it we observe sale price is increase.
17. Sales price vs OverallCond 9 and 5 is having high sale price.
18. Sales price vs Yearbuilt
19. Sales price vs YearRemodAdd
20. Sales price vs Roofstyle in 0,3 and 5 roofstyle having high sale price.
21. Sales price vs Roofmati in 7 having high sale price.
22. Sales price vs Exterior1st 6,9 and 4 having high sale price.
23. Sales price vs Exterior2nd 8 and 6 having high sale price.
24. Sales price vs Masvnrtypr in 3 has having high sale price.
25. Sales price vs Masvnrarea
26. Sales price vs ExterQual in this 0 is having high sale price.
27. Sales price vs Extercond in there are 0 is having high sale price.
28. Sales price vs Foundation there are 2 type of foundation is having high sale price.
29. Sales price vs BsmtQual there are 0 BsmtQual having high sale price.
30. Sales price vs BsmtCond there are 1 type of BsmtCond is having high sale price.
31. Sales price vs BsmtExposure there are 1 type of BsmtExposure having high sale price.
32. Sales price vs Bsmtfintype1 there are 2 is having high sale price.
33. Sales price vs Bsmitfinsf
34. Sales price vs Bsmtfintype1 there are 0 and 2 is having high sale price.
35. Sales price vs Bsmitfinsf2
36. Sales price vs BsmtUnfsf
37. Sales price vs TotalBsmtSf
38. Sales price vs Heating there are 1 and 2 is having high sale price.
39. Sales price vs HeatingQC there are 0 type is having high sale price.
40. Sales price vs CentalAir 1 means there are central air is having high sale price.
41. Sales price vs Electrical there are 4th type of electrical is having high sale price.
42. Sales price vs 1stFlrsf
43. Sales price vs 2ndFlrsf
44. Sales price vs LowQualsf is having high price if there are 72.
45. Sales price vs Grlivarea
46. Sales price vs BsmtFullBath is having high sale price if it is 1 and 2.
47. Sales price vs BsmtHalfBath is having high sale price if it is 0 and 1.
48. Sales price vs FullBath is having high price if it is 3.
49. Sales price vs HalfBath is having high price if it is 1.
50. Sales price vs BedroomAbvGr is having high price if it is 0,4 and 8.
51. Sales price vs KitchenabvGr has high price if it is 1.

52. Sales price vs KitchenQual having high price if it is 0.
53. Sales price vs TotRmsabvGrd is having high price if it is 10 to 12.
54. Sales price vs Functional 6,2 and 0 is having high sale price.
55. Sales price vs Fireplaces is having high sale price if it is 3.
56. Sales price vs GarageType is having high sale price if it is 3.
57. Sales price vs GarageyrBuilt
58. Sales price vs Garagefinish is having high sale price if it is 0.
59. Sales price vs GarageCars is having high sale price if it is 3 car parking.
60. Sales price vs Garagearea
61. Sales price vs GarageQual is having high price if it is 0 and 2.
62. Sales price vs Garagecond is having high price if it is 2 and 4.
63. Sales price vs PavedDrive is having high sale price if it is 2.
64. Sales price vs WoodDeskSf.
65. Sales price vs OpenporchSF
66. Sales price vs EnclosedPorch
67. Sales price vs 3SsnPorch is having high sale price if it is 30 and 15.
68. Sales price vs ScreenPorch
69. Sales price vs PoolArea is having high sale price if it is 555.
70. Sales price vs MiscVal
71. Sales price vs MoSold is having high sale price if it is 9 and 7.
72. Sales price vs YrSold is having high sale price if it is 2006,2007
73. Sales price vs SaleType is having high sale price if it is 2 and 6.
74. Sales price vs SaleCondition is having high sale price if it is 5.

- Now we start multivariate analysis by use of df.corr() function.

```
In [80]: df.corr()
```

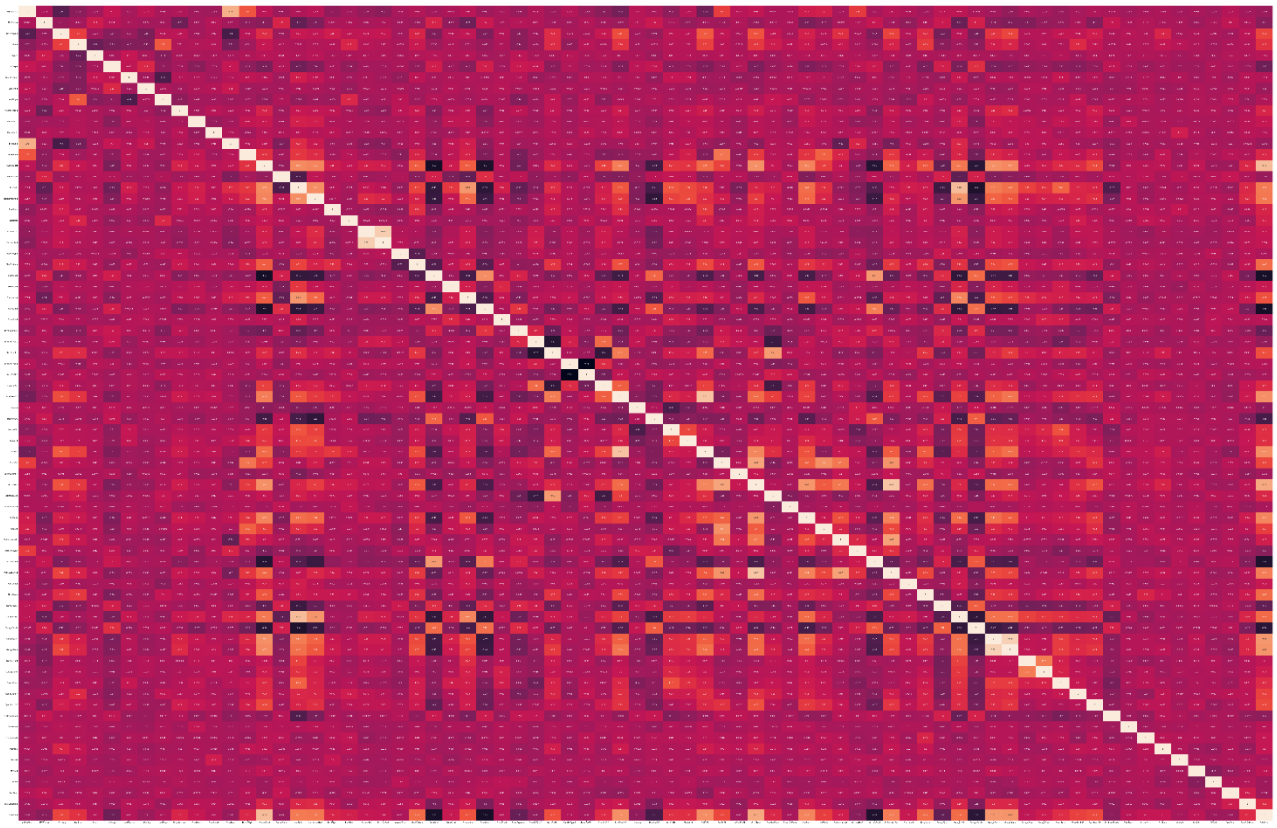
- Output of this function is like this.

Out[80]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	...	EnclosedPorc
MSSubClass	1.000000	0.007478	-0.336681	-0.124151	-0.035981	0.104485	-0.021387	0.076880	-0.014930	0.013918	...	-0.00425
MSZoning	0.007478	1.000000	-0.069661	-0.023328	0.140215	0.053655	0.001175	-0.027246	-0.023952	-0.251833	...	0.11122
LotFrontage	-0.336681	-0.069661	1.000000	0.299452	-0.035309	-0.144523	-0.073451	-0.192468	0.046051	0.065824	...	0.02090
LotArea	-0.124151	-0.023328	0.299452	1.000000	-0.263973	-0.189201	-0.159038	-0.152063	0.395410	0.010707	...	-0.00744
Street	-0.035981	0.140215	-0.035309	-0.263973	1.000000	-0.012941	0.105226	0.000153	-0.141572	0.001420	...	0.02136
...	...	...	...	...	...	...	...	...	...	...	...	.
MoSold	-0.016015	-0.051646	0.022517	0.015141	-0.008860	-0.050418	-0.023872	0.019084	0.030526	0.023378	...	-0.03652
YrSold	-0.038595	-0.004964	-0.003885	-0.035399	-0.019635	0.021421	0.009499	-0.009817	-0.005352	0.026181	...	-0.00576
SaleType	0.035050	0.079854	-0.035356	0.005421	0.025920	-0.015161	-0.041763	-0.002039	0.056004	-0.023081	...	-0.00823
SaleCondition	-0.028981	0.004501	0.065091	0.034236	0.014176	-0.054905	0.047715	0.043692	-0.061461	0.042340	...	-0.09156
SalePrice	-0.060775	-0.133221	0.323779	0.249499	0.044753	-0.248171	0.032836	-0.060452	0.015485	0.198942	...	-0.11500

74 rows × 74 columns

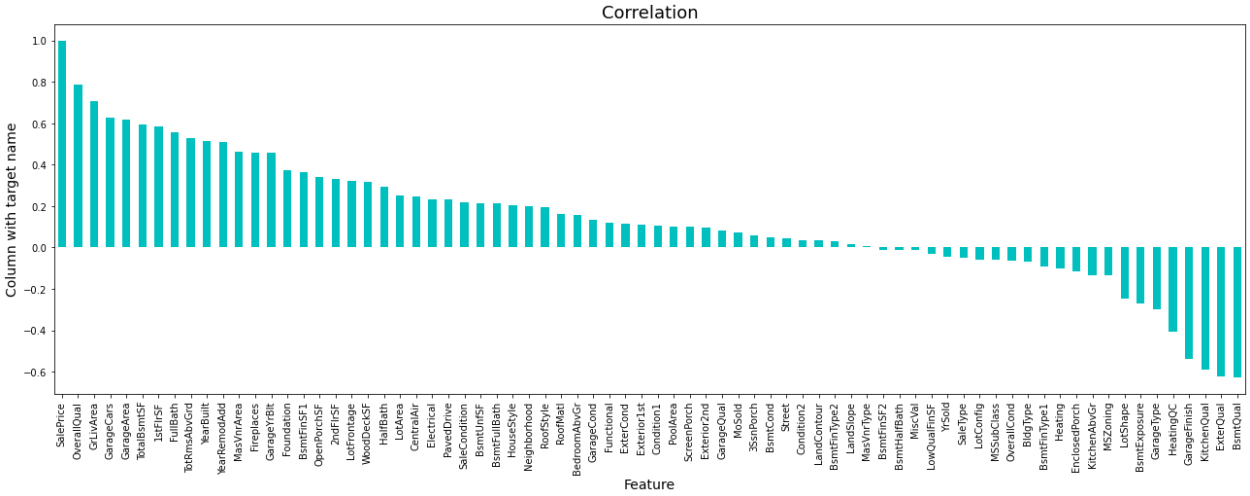
- We cannot clearly observe anything for that we plot heatmap.



- By observing this heatmap we find some positive and negative correlation ship with our target variable.

1. Sales price vs LotFrontage this is positively correlate.
2. Sales price vs OverallQual this is positively correlate.
3. Sales price vs Yearbuilt this is positively correlate.
4. Sales price vs YearRemonded this is positively correlate.
5. Sales price vs MasVararea this is positively correlate.
6. Sales price vs Extrer Qual this is negatively correlate.
7. Sales price vs Foundation this is positively correlate.
8. Sales price vs BsmtQual this is negatively correlate.
9. Sales price vs TotalBsmtSf this is positively correlate.
10. Sales price vs 1stFirstSf this is positively correlate.
11. Sales price vs BsmtFullBath this is positively correlate.
12. Sales price vs HalfBath this is positively correlate.
13. Sales price vs Kitchen this is positively correlate.
14. Sales price vs TotRms this is positively correlate.
15. Sales price vs Fireplaces this is positively correlate.
16. Sales price vs GarageCars this is positively correlate.
17. Sales price vs Garagearea this is positively correlate.

- Now we plot one graph and check its relationship with target variable only.



- Now for statistical analysis we use pandas describe function.

```
In [83]: df.describe()
```

- Output of this function should be like this.

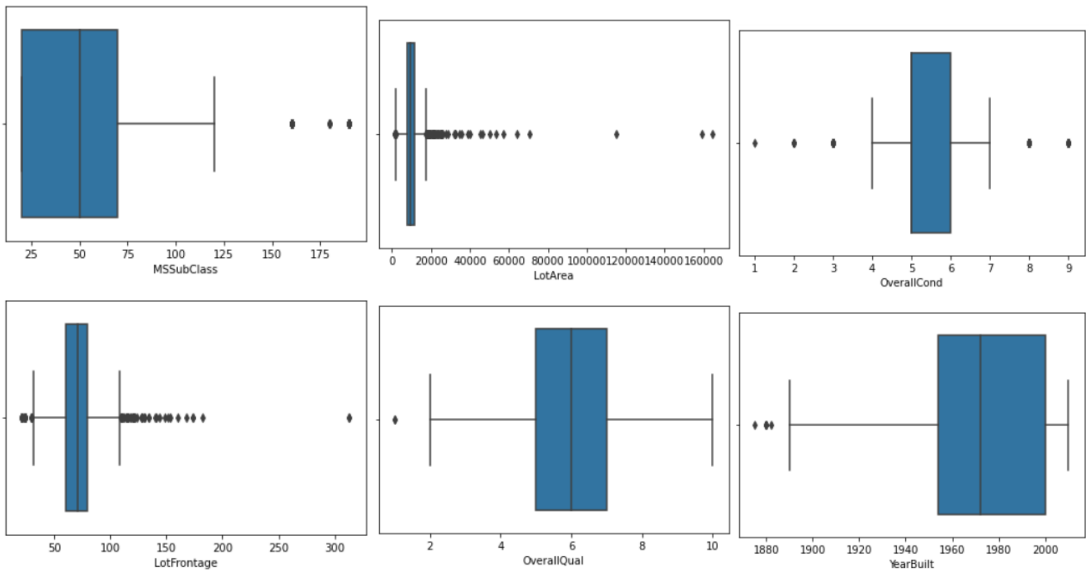
Out[83]:

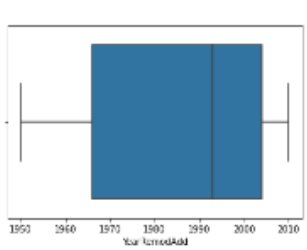
	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	...	Enclos
count	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	...	116
mean	56.767979	3.013699	70.988470	10484.749144	0.996575	1.938356	2.773973	3.004281	0.064212	12.145548	...	2
std	41.940650	0.633120	22.437056	8957.442311	0.058445	1.412262	0.710027	1.642667	0.284088	6.010364	...	6
min	20.000000	0.000000	21.000000	1300.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
25%	20.000000	3.000000	60.000000	7621.500000	1.000000	0.000000	3.000000	2.000000	0.000000	7.000000	...	
50%	50.000000	3.000000	70.988470	9522.500000	1.000000	3.000000	3.000000	4.000000	0.000000	12.000000	...	
75%	70.000000	3.000000	79.250000	11515.500000	1.000000	3.000000	3.000000	4.000000	0.000000	17.000000	...	
max	190.000000	4.000000	313.000000	164660.000000	1.000000	3.000000	3.000000	4.000000	2.000000	24.000000	...	55

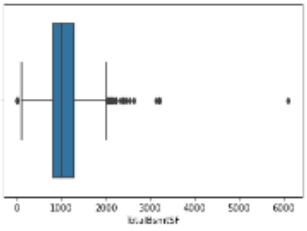
8 rows × 74 columns

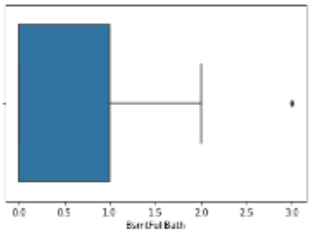
- Now we check for outliers in numerical type data only in input variables.
- For that we use box graph for plot it each column we use for loop.

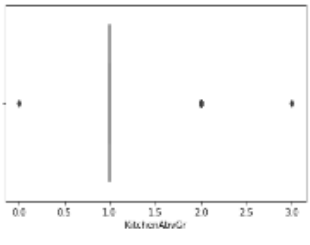
```
In [85]: for i in numerical[:-1]:
             plt.plot(figsize=(6,4))
             sns.boxplot(df[i])
             plt.show()
```

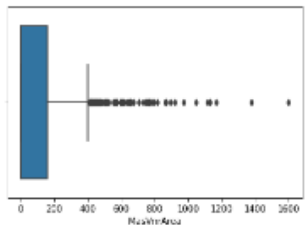


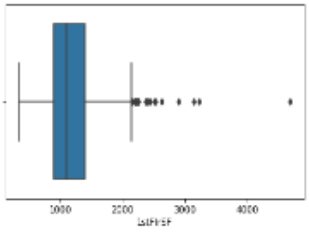

1950 1960 1970 1980 1990 2000 2010
YearRemodAdd

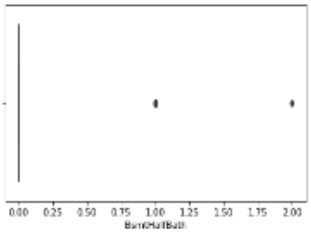

0 1000 2000 3000 4000 5000 6000
TotalBsmtSF

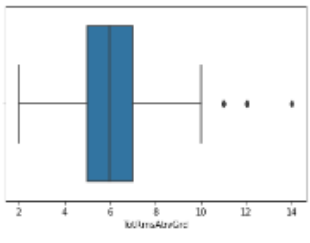

0.0 0.5 1.0 1.5 2.0 2.5 3.0
BsmtFullBath

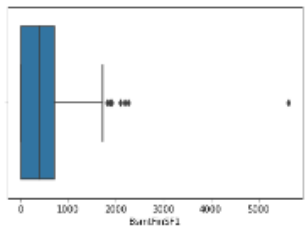

0.0 0.5 1.0 1.5 2.0 2.5 3.0
KitchenAbvGr

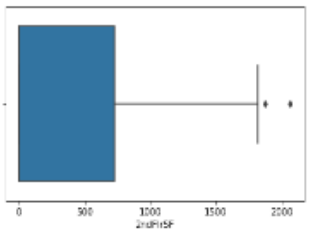

0 220 400 600 800 1000 1200 1400 1600
MasVnrArea

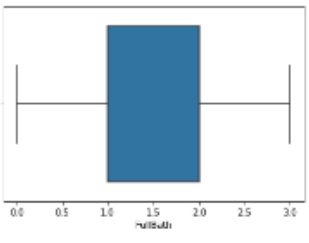

1000 2000 3000 4000
1stFlrSF

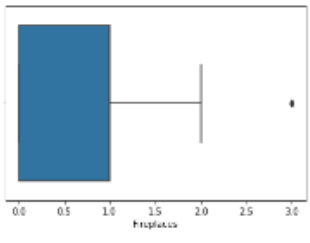

0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
BsmtHalfBath

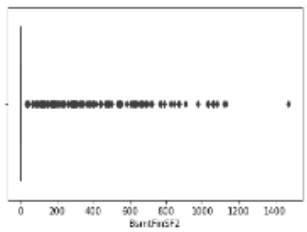

2 4 6 8 10 12 14
TotRmsAbvGrd

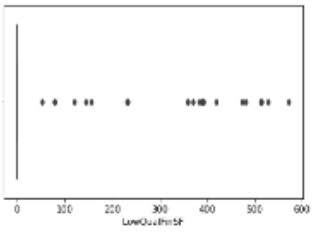

0 1000 2000 3000 4000 5000
BsmtFinSF1

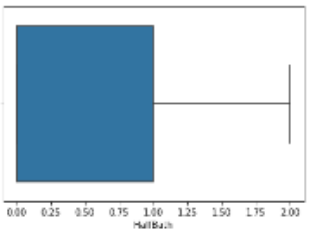

0 500 1000 1500 2000
2ndFlrSF

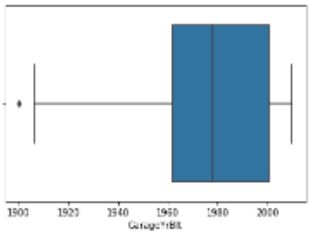

0.0 0.5 1.0 1.5 2.0 2.5 3.0
FullBath

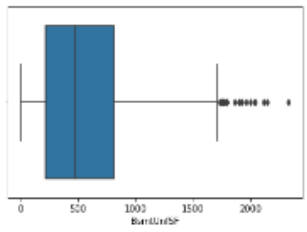

0.0 0.5 1.0 1.5 2.0 2.5 3.0
Fireplaces

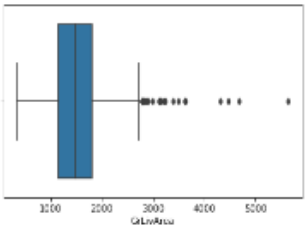

0 200 400 600 800 1000 1200 1400
BsmtFinSF2

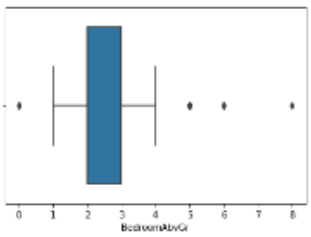

0 100 200 300 400 500 600
LowQualFinSF

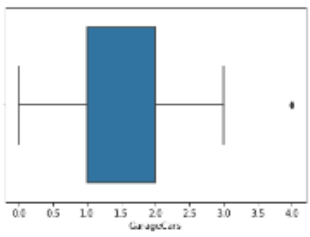

0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
HalfBath

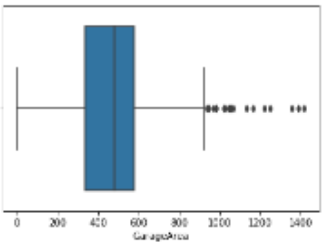

1900 1920 1940 1960 1980 2000
GarageYrBlt

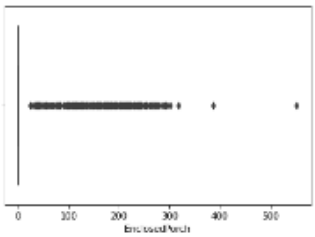

0 500 1000 1500 2000
BsmtUnfSF

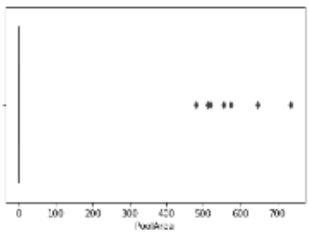

1000 2000 3000 4000 5000
GrLivArea

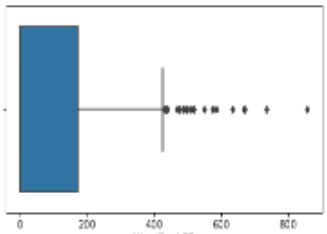

0 1 2 3 4 5 6 7 8
BedroomAbvGr

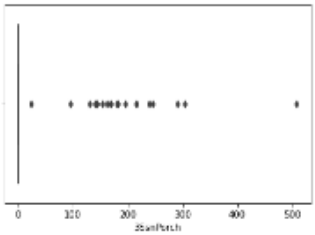

0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
GarageCars

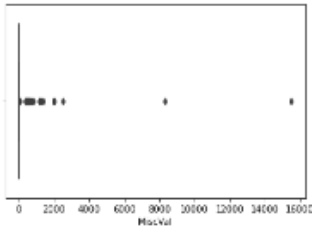

0 200 400 600 800 1000 1200 1400
GarageArea

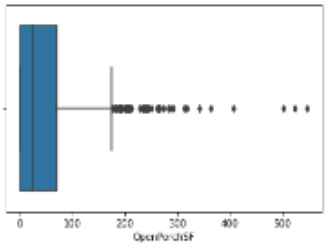

0 100 200 300 400 500
EnclosedPorch

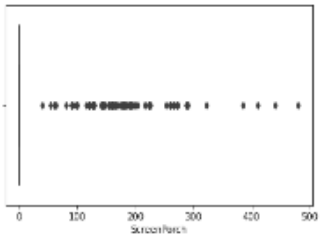

0 100 200 300 400 500 600 700
PoolArea

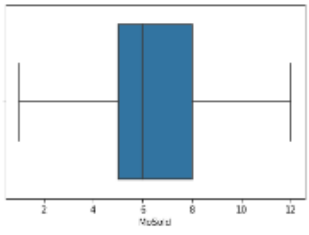

0 200 400 600 800
WoodDeckSF

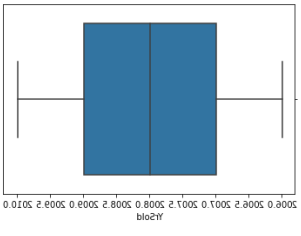

0 100 200 300 400 500
3SsnPorch


0 2000 4000 6000 8000 10000 12000 14000 16000
MiscVal


0 100 200 300 400 500
OpenPorchSF


0 100 200 300 400 500
ScreenPorch


2 4 6 8 10 12
MoSold


2006 2007 2008 2009 2010
YrSold

- By observing all the above graph we observe that many columns are contain outliers we dropped it.
  1. Lotarea
  2. MasVnrarea
  3. BsmtFinsF2
  4. LowQualFinSf
  5. WoodDickSf
  6. OpenPorchSF
  7. Enclosed Porch
  8. 3Snporch
  9. Screen porch
  10. pool area
  11. Misc val

  these columns are having highly outliers.
  By observe above columns we observe that
  BsmtFinsF2,EnclosedPorch,3snPorch and screen porch these are having very high outliers for that drop this columns by check its skewness.

- Now we drop those columns.

In [91]:
```
df.drop(['BsmtFinSF2','EnclosedPorch','3SsnPorch','ScreenPorch','LotArea'],axis=1,inplace=True)
df.head()
```

- Now we have to remove skewness in this for removing skewness we use boxcox, sqrt ,log and yeojohnson.
- By applying it in each columns they contains skewness then we check for output of all above 4 method and then we decide to which one is best here one example.

In [95]:
```
#Lets find the best method for skewness for negative and positive both values
def skee(a):
    model=[np.sqrt(a),np.log(a),yeojohnson(a)[0]]
    print('original skewness is:',a.skew())
    print('\n')
    for m in model:
        x=m
        print(skew(m))
        print('\n')
```

In [96]:
```
#Lets find the best method for skewness for only positive value
def skeep(a):
    model=[np.sqrt(a),np.log(a),boxcox(a)[0],yeojohnson(a)[0]]
    print('original skewness is:',a.skew())
    print('\n')
    for m in model:
        x=m
        print(skew(m))
        print('\n')
```

In [97]:
```
skee(df['MasVnrArea'])
```

```
original skewness is: 2.8346577812934406


1.074033008376098


nan


0.41583498274711855
```

In [98]:
```
df['MasVnrArea']=yeojohnson(df['MasVnrArea'])[0]
```

- By use of this algorithm, we remove skewness in all the data.
- When we remove outliers in the data by use of zscore then our data loss is around 50% and for IQR method it is 89% so we skip this steps for remove outliers in data.
- Now we split our variable in two-part input and output.

In [134]:
```
x=df.drop(['SalePrice'],axis=1)
y=df['SalePrice']
x.head()
```

Out[134]:

	MSSubClass	MSZoning	LotFrontage	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	Condition1	...	GarageCond	PavedDrive	W
0	120	3	70.98847	1	0	3	4	0	13	2	...	4	2	
1	20	3	95.00000	1	0	3	4	1	12	2	...	4	2	
2	60	3	92.00000	1	0	3	1	0	15	2	...	4	2	
3	20	3	105.00000	1	0	3	4	0	14	2	...	4	2	
4	20	3	70.98847	1	0	3	2	0	14	2	...	4	2	

5 rows × 67 columns

In [135]:
```
y.head()
```

Out[135]:
```
0    128000
1    268000
2    269790
3    190000
4    215000
Name: SalePrice, dtype: int64
```

- Since we observe that our target variable is in numerical type so we should use regression model.
- Now for regression algorithm we import libraries.

In [138]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
from sklearn.model_selection import train_test_split
```

- For finding best random state we use below code for it.

In [139]:
```
for i in range(0,1000):
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=i,test_size=0.25)
    lr.fit(x_train,y_train)
    pred_train=lr.predict(x_train)
    pred_test=lr.predict(x_test)
    if round(r2_score(y_train,pred_train)*100,1)==round(r2_score(y_test,pred_test)*100,1):
        print('At random state',i,'The model perform very well')
        print('Random State = ',i)
        print("Training r2_score is = ",r2_score(y_train,pred_train))
        print("Test r2_score is = ",r2_score(y_test,pred_test))
        print('\n')
```

- By observe output of above code we observe that at 343 random state train and test score are same.

```
At random state 343 The model perform very well
Random State =  343
Training r2_score is =  0.8212570043281072
Test r2_score is =  0.8210814836435751
```

- Now apply all the algorithms and check its error,r2_score and Cross-Val score.

In [140]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=343,test_size=0.25)
```

In [141]:
```
lr.fit(x_train,y_train)
pred_test_lr=lr.predict(x_test)
print("r2_score is = ",r2_score(y_test,pred_test_lr)*100)
print('mean_absolute_error = ',mean_absolute_error(y_test,pred_test_lr))
print('mean_squared_error = ',mean_squared_error(y_test,pred_test_lr))
print('root_mean_squared_error = ',np.sqrt(mean_squared_error(y_test,pred_test_lr)))
```

```
r2_score is =  82.10814836435752
mean_absolute_error =  19901.86801182776
mean_squared_error =  1106678018.0024643
root_mean_squared_error =  33266.770477496975
```

- Now observe here only output r2_score and all the errors of all the models.

In [146]:
```
# GradientBoostingRegressor
gb.fit(x_train,y_train)
pred_test_gb=gb.predict(x_test)
print("r2_score is = ",r2_score(y_test,pred_test_gb)*100)
print('mean_absolute_error = ',mean_absolute_error(y_test,pred_test_gb))
print('mean_squared_error = ',mean_squared_error(y_test,pred_test_gb))
print('root_mean_squared_error = ',np.sqrt(mean_squared_error(y_test,pred_test_gb)))
```

```
r2_score is =  89.19619045011066
mean_absolute_error =  15980.330771985744
mean_squared_error =  668256074.498702
root_mean_squared_error =  25850.64940187581
```

In [149]:
```
# AdaBoostRegressor
ada.fit(x_train,y_train)
pred_test_ada=ada.predict(x_test)
print("r2_score is = ",r2_score(y_test,pred_test_ada)*100)
print('mean_absolute_error = ',mean_absolute_error(y_test,pred_test_ada))
print('mean_squared_error = ',mean_squared_error(y_test,pred_test_ada))
print('root_mean_squared_error = ',np.sqrt(mean_squared_error(y_test,pred_test_ada)))
```

```
r2_score is =  82.16439693629188
mean_absolute_error =  23668.721658489478
mean_squared_error =  1103198833.210892
root_mean_squared_error =  33214.43712018754
```

In [155]:
```
# RandomForestRegressor
rfc.fit(x_train,y_train)
pred_test_rfc=rfc.predict(x_test)
print("r2_score is = ",r2_score(y_test,pred_test_rfc)*100)
print('mean_absolute_error = ',mean_absolute_error(y_test,pred_test_rfc))
print('mean_squared_error = ',mean_squared_error(y_test,pred_test_rfc))
print('root_mean_squared_error = ',np.sqrt(mean_squared_error(y_test,pred_test_rfc)))
```

```
r2_score is =  87.45614532064563
mean_absolute_error =  17178.358527397257
mean_squared_error =  775884381.1897249
root_mean_squared_error =  27854.701240360217
```

In [158]:
```
# SVR
svc.fit(x_train,y_train)
pred_test_svc=svc.predict(x_test)
print("r2_score is = ",r2_score(y_test,pred_test_svc)*100)
print('mean_absolute_error = ',mean_absolute_error(y_test,pred_test_svc))
print('mean_squared_error = ',mean_squared_error(y_test,pred_test_svc))
print('root_mean_squared_error = ',np.sqrt(mean_squared_error(y_test,pred_test_svc)))
```

```
r2_score is =  -2.8794593917470523
mean_absolute_error =  53941.35394170616
mean_squared_error =  6363479785.737403
root_mean_squared_error =  79771.42211179015
```

Very very less acuracy no need to go ahed for SVR

In [159]:
```
# DecisionTreeRegressor
dtc.fit(x_train,y_train)
pred_test_dtc=dtc.predict(x_test)
print("r2_score is = ",r2_score(y_test,pred_test_dtc)*100)
print('mean_absolute_error = ',mean_absolute_error(y_test,pred_test_dtc))
print('mean_squared_error = ',mean_squared_error(y_test,pred_test_dtc))
print('root_mean_squared_error = ',np.sqrt(mean_squared_error(y_test,pred_test_dtc)))
```

```
r2_score is =  75.94383398157143
mean_absolute_error =  26744.41095890411
mean_squared_error =  1487963944.2671232
root_mean_squared_error =  38574.1356904743
```

In [164]:
```
# KNeighborsRegressor
knn.fit(x_train,y_train)
pred_test_knn=knn.predict(x_test)
print("r2_score is = ",r2_score(y_test,pred_test_knn)*100)
print('mean_absolute_error = ',mean_absolute_error(y_test,pred_test_knn))
print('mean_squared_error = ',mean_squared_error(y_test,pred_test_knn))
print('root_mean_squared_error = ',np.sqrt(mean_squared_error(y_test,pred_test_knn)))
```

```
r2_score is =  69.31991047961824
mean_absolute_error =  28349.315753424657
mean_squared_error =  1897678415.5149314
root_mean_squared_error =  43562.350895181626
```

- Now we compare all the above methods by use of r2_score,erros,cross-val-score.

Model	R2_score	CV	Cross_val_score	MAE	MSE	RMSE
Linear	82.10	8	76.94	19901	1106678018	33266
GBR	89.12	3	87.84	15980	668256074	25850
ADA	82.16	3	80.13	23668	1103198833	33214
RFR	87.46	3	85.26	17178	775884381	27854
SVR	-	-	-	-	-	-
DTR	75.94	6	73.60	26744	147963944	38574
KNN	69.32	3	68.98	28349	1897678415	43562

- As we observe in above compare table, we observe that in GBR we get high accuracy and low error as compare to others. so, we apply hyper parameters on GBR method.
- Now we apply all they hyper parameter to this model.

In [170]:
```
grid_param={

    'loss':['squared_error', 'absolute_error','quantile'],
    'learning_rate' :[0.1,0,1,1.1],
    'n_estimators' : [100, 200, 400],
    'criterion' :['friedman_mse', 'squared_error'],
    'random_state':[None,100,10,200],
    'max_features':[None,'sqrt','auto']
}
```

In [171]:
```
from sklearn.model_selection import GridSearchCV
```

In [173]:
```
gd_sr=GridSearchCV(estimator=gb,
                   param_grid=grid_param,
                   scoring='accuracy',
                   cv=3)
```

In [174]:
```
gd_sr.fit(x,y)
```

Out[174]:
```
GridSearchCV(cv=3, estimator=GradientBoostingRegressor(),
             param_grid={'criterion': ['friedman_mse', 'squared_error'],
                         'learning_rate': [0.1, 0, 1, 1.1],
                         'loss': ['squared_error', 'absolute_error',
                                  'quantile'],
                         'max_features': [None, 'sqrt', 'auto'],
                         'n_estimators': [100, 200, 400],
                         'random_state': [None, 100, 10, 200]},
             scoring='accuracy')
```

In [175]:
```
best_parameters=gd_sr.best_params_
print(best_parameters)
```

```
{'criterion': 'friedman_mse', 'learning_rate': 1, 'loss': 'quantile', 'max_features': None, 'n_estimators': 100, 'random_sta
te': None}
```

- We find out all the hyper parameter now we apply it and check r2_score.

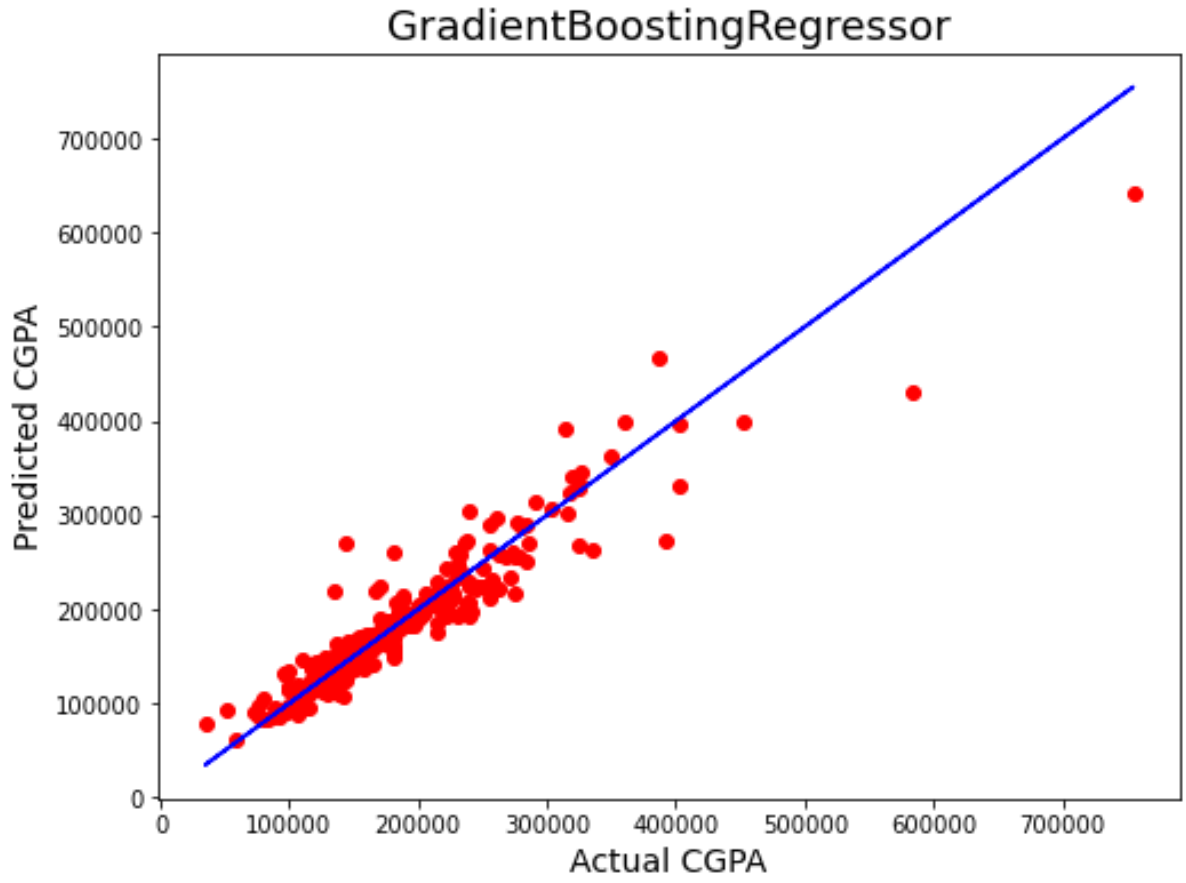
In [176]:
```
gbb=GradientBoostingRegressor(criterion='friedman_mse',learning_rate=1,loss='quantile',max_features=None,n_estimators=100,
```

In [177]:
```
# GradientBoostingRegressor
gbb.fit(x_train,y_train)
pred_test_gbb=gbb.predict(x_test)
print("r2_score is = ",r2_score(y_test,pred_test_gbb)*100)
print('mean_absolute_error = ',mean_absolute_error(y_test,pred_test_gbb))
print('mean_squared_error = ',mean_squared_error(y_test,pred_test_gbb))
print('root_mean_squared_error = ',np.sqrt(mean_squared_error(y_test,pred_test_gbb)))
```

```
r2_score is =  53.891810916694794
mean_absolute_error =  35609.45890410959
mean_squared_error =  2851964142.5342464
root_mean_squared_error =  53403.78397205807
```

- Now we observe it's default parameters are ok by applying hyper parameter we observe that our r2_score is decrease so we keep with default parameters.
- Now we plot graph for test and pred_test.

In [178]:
```
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=pred_test_gb,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Actual CGPA',fontsize=14)
plt.ylabel('Predicted CGPA',fontsize=14)
plt.title('GradientBoostingRegressor',fontsize=18)
plt.show()
```



- Now our model is ready now we saved our model in obj file by use of joblib library.

In [179]:
```
import joblib
```

In [180]:
```
joblib.dump(gb,'GradientBoostingRegressor_Project_Housign_Splitted.obj')
```

Out[180]: ['GradientBoostingRegressor_Project_Housign_Splitted.obj']

- Now our model is ready to test data and predict for that we load our test dataset.

In [183]:
```
dft=pd.read_csv('test.csv')
dft
```

Out[183]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	PoolQC	Fence	Mis
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	...	0	0	NaN	NaN	
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	...	0	0	NaN	NaN	
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
287	83	20	RL	78.0	10206	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	
288	1048	20	RL	57.0	9245	Pave	NaN	IR2	Lvl	AllPub	...	0	0	NaN	NaN	
289	17	20	RL	NaN	11241	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	
290	523	50	RM	50.0	5000	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	
291	1379	160	RM	21.0	1953	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	

292 rows × 80 columns

- First we should remove all column which we removed earlier for train model.

```
In [184]: dft.drop(['Id','Utilities','Alley','FireplaceQu','PoolQC','Fence','MiscFeature','BsmtFinSF2','EnclosedPorch','3SsnPorch','Scr
```

```
In [185]: dft.shape
Out[185]: (292, 67)
```

- Now we have to converted object data into numerical data for that we use Label Encoder.

```
In [188]: Byforgets=['MSSubClass', 'MSZoning', 'LotFrontage', 'Street', 'LotShape',
                     'LandContour', 'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1',
                     'Condition2', 'BldgType', 'HouseStyle', 'OverallQual', 'OverallCond',
                     'YearBuilt', 'YearRemodAdd', 'RoofStyle', 'RoofMatl', 'Exterior1st',
                     'Exterior2nd', 'MasVnrType', 'MasVnrArea', 'ExterQual', 'ExterCond',
                     'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1',
                     'BsmtFinType2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating', 'HeatingQC',
                     'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',
                     'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',
                     'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual', 'TotRmsAbvGrd',
                     'Functional', 'Fireplaces', 'GarageType', 'GarageYrBlt', 'GarageFinish',
                     'GarageCars', 'GarageArea', 'GarageQual', 'GarageCond', 'PavedDrive',
                     'WoodDeckSF', 'OpenPorchSF', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold',
                     'SaleType', 'SaleCondition']
objectss=[]
numericals=[]
for i in Byforgets:
    if dft[i].dtypes=='object':
        objectss.append(i)
    else:
        numericals.append(i)
```

```
In [190]: for i in objectss:
              dft[i]=le.fit_transform(dft[i])
```

```
In [192]: dft.head(1)
```

Out[192]:

	MSSubClass	MSZoning	LotFrontage	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	Condition1	...	GarageCond	PavedDrive	W
0	20	2	86.0	1	0	1	0	0	21	2	...	4	2	

1 rows × 67 columns

- Now our data is ready for test it.
- We load our model.

Load model

```
In [193]: import joblib
          load_house=joblib.load("GradientBoostingRegressor_Project_Housign_Splitted.obj")
          load_house
Out[193]: GradientBoostingRegressor()
```

- We loaded our model now we give test data to model and then we create data frame for it and join that data frame with our test data data frame.

```
In [206]: pred=load_house.predict(dft)
```

In [209]:
```
dfo=pd.DataFrame({
    'SalePriceTest':pred
})
dfo
```

Out[209]:

	SalePriceTest
0	344363.515966
1	211257.070541
2	276093.584139
3	198477.783601
4	344705.012849
...	...
225	205871.949795
226	358441.073233
227	201368.513514
228	238107.773333
229	187517.239688

230 rows × 1 columns

- Now we join this data frame with our test data frame.

In [210]:
```
result = pd.concat([dft, dfo], axis=1)
result.head()
```

Out[210]:

	MSSubClass	MSZoning	LotFrontage	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	Condition1	...	PavedDrive	WoodDeckSF
0	20.0	2.0	86.0	1.0	0.0	1.0	0.0	0.0	21.0	2.0	...	2.0	178.0
3	70.0	2.0	75.0	1.0	3.0	0.0	4.0	0.0	5.0	2.0	...	2.0	0.0
4	60.0	2.0	86.0	1.0	0.0	3.0	1.0	0.0	20.0	1.0	...	2.0	100.0
6	180.0	3.0	35.0	1.0	3.0	3.0	4.0	0.0	6.0	2.0	...	2.0	0.0
7	20.0	2.0	107.0	1.0	3.0	3.0	4.0	0.0	15.0	2.0	...	2.0	0.0

5 rows × 68 columns

In [210]:
```
result = pd.concat([dft, dfo], axis=1)
result.head()
```

Out[210]:

Slope	Neighborhood	Condition1	...	PavedDrive	WoodDeckSF	OpenPorchSF	PoolArea	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePriceTest
0.0	21.0	2.0	...	2.0	178.0	51.0	0.0	0.0	7.0	2007.0	5.0	2.0	344363.515966
0.0	5.0	2.0	...	2.0	0.0	0.0	0.0	0.0	7.0	2009.0	5.0	2.0	198477.783601
0.0	20.0	1.0	...	2.0	100.0	18.0	0.0	0.0	1.0	2008.0	5.0	2.0	344705.012849
0.0	6.0	2.0	...	2.0	0.0	28.0	0.0	0.0	5.0	2006.0	5.0	2.0	126513.035021
0.0	15.0	2.0	...	2.0	0.0	102.0	0.0	0.0	1.0	2008.0	4.0	3.0	130660.131437

- As we observe our data frame are concat with each others.

Tools to be used

- Jupyter notebook
- Anaconda
- Sklearn
- Pandas
- Matplotlib
- Seaborn
- SciPy
- Kaggle