FLIP ROBO

2022

HOUSE PRICE PREDICTION

Submitted by:

Bhushan Kumar Sharma,

Intern Data Scientist

ACKNOWLEDGMENT

*****

I would like to express my special gratitude to “Flip Robo” team, who has given me this opportunity to deal with this dataset and it has helped me to improve my model building skills.

A huge thanks to my academic team “Datatrained” who is the reason behind what I am today. Last but not least my parents who have been my backbone in every step of my life. And also thank to many other persons who has helped me directly or indirectly to complete the project.

Following are the external references which I used:

www.geeksforgeeks.org

www.stackoverflow.com

www.w3school.com

www.google.com

Datatrained Lectures

. INTRODUCTION

➢ Business Problem Framing

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. For the same purpose, the company has collected a data set from the sale of houses in Australia. The data is provided in the CSV file below.

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?

➢ Conceptual Background of the Domain Problem

You are required to model the price of houses with the available independent variables. This model will 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.

Determining the listed price of a house is a challenging task, due to the many factors that price on the market. The focus of this project is developing machine learning models that can accurately predict the price of a house based on its features, in order to make informed purchases.

We implement and evaluate various learning methods on a dataset consisting of the sale prices of different makes and models across features. Our results show that GradientBoostingRegressor is giving best results.

➢ Review of Literature

- Two Datasets are provided, one is train dataset and second is test dataset.
- Prize will get decided based on various feature column
- After Pre-processing and EDA, data is used to build Machine learning model, as we have to predict prize of house, which clearly indicating this is a regression problem.
- Various pre-processing techniques are used in this project to give best data to our machine learning model, and at final it's giving good accuracy.

➢ Motivation for the Problem Undertaken

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.

➢ Data Sources and their formats

This dataset is provided by fliprobo for building machine learning model to predict house price based on given feature column.

Dataset is in two part one is for building model and second is for which we have to predict price.

Train dataset:
Dataset is having 1168 rows and 81 columns including target. In this particular datasets I have object, float and integer types of columns.

Test dataset:
Dataset is having 292 rows and 80 columns and we have to predict price of house after creating ML model.

The information about features is as follows.

```
train_df.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Cond
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NPkVill	Norm	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	Inside	Mod	NAmes	Norm	
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	NoRidge	Norm	
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NWAmes	Norm	
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NWAmes	Norm	

```
train_df.head()
```

ıı	Condition2	BldgType	HouseStyle	OverallQual	OverallCond	YearBuilt	YearRemodAdd	RoofStyle	RoofMatl	Exterior1st	Exterior2nd	MasVnrType	MasVnrAr
m	Norm	TwnhsE	1Story	6	5	1976	1976	Gable	CompShg	Plywood	Plywood	None	0
m	Norm	1Fam	1Story	8	6	1970	1970	Flat	Tar&Grv	Wd Sdng	Wd Sdng	None	0
m	Norm	1Fam	2Story	7	5	1996	1997	Gable	CompShg	MetalSd	MetalSd	None	0
m	Norm	1Fam	1Story	6	6	1977	1977	Hip	CompShg	Plywood	Plywood	BrkFace	480
m	Norm	1Fam	1Story	6	7	1977	2000	Gable	CompShg	CemntBd	CmentBd	Stone	126

```
train_df.head()
```

MasVnrArea	ExterQual	ExterCond	Foundation	BsmtQual	BsmtCond	BsmtExposure	BsmtFinType1	BsmtFinSF1	BsmtFinType2	BsmtFinSF2	BsmtUnfSF	Tot
0.0	TA	TA	CBlock	Gd	TA	No	ALQ	120	Unf	0	958	
0.0	Gd	Gd	PConc	TA	Gd	Gd	ALQ	351	Rec	823	1043	
0.0	Gd	TA	PConc	Gd	TA	Av	GLQ	862	Unf	0	255	
480.0	TA	TA	CBlock	Gd	TA	No	BLQ	705	Unf	0	1139	
126.0	Gd	TA	CBlock	Gd	TA	No	ALQ	1246	Unf	0	356	

```
train_df.head()
```

TotalBsmtSF	Heating	HeatingQC	CentralAir	Electrical	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath	Bedroc
1078	GasA	TA	Y	SBrkr	958	0	0	958	0	0	2	0	
2217	GasA	Ex	Y	SBrkr	2217	0	0	2217	1	0	2	0	
1117	GasA	Ex	Y	SBrkr	1127	886	0	2013	1	0	2	1	
1844	GasA	Ex	Y	SBrkr	1844	0	0	1844	0	0	2	0	
1602	GasA	Gd	Y	SBrkr	1602	0	0	1602	0	1	2	0	

```
train_df.head()
```

BedroomAbvGr	KitchenAbvGr	KitchenQual	TotRmsAbvGrd	Functional	Fireplaces	FireplaceQu	GarageType	GarageYrBlt	GarageFinish	GarageCars	GarageAr
2	1	TA	5	Typ	1	TA	Attchd	1977.0	RFn	2	4
4	1	Gd	8	Typ	1	TA	Attchd	1970.0	Unf	2	6
3	1	TA	8	Typ	1	TA	Attchd	1997.0	Unf	2	4
3	1	TA	7	Typ	1	TA	Attchd	1977.0	RFn	2	5
3	1	Gd	8	Typ	1	TA	Attchd	1977.0	Fin	2	5

```
train_df.head()
```

GarageArea	GarageQual	GarageCond	PavedDrive	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	PoolQC	Fence	MiscFeatu
440	TA	TA	Y	0	205	0	0	0	0	NaN	NaN	Na
621	TA	TA	Y	81	207	0	0	224	0	NaN	NaN	Na
455	TA	TA	Y	180	130	0	0	0	0	NaN	NaN	Na
546	TA	TA	Y	0	122	0	0	0	0	NaN	MnPrv	Na
529	TA	TA	Y	240	0	0	0	0	0	NaN	NaN	Na

```
train_df.head()
```

OpenPorchSF	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice
205	0	0	0	0	NaN	NaN	NaN	0	2	2007	WD	Normal	128000
207	0	0	224	0	NaN	NaN	NaN	0	10	2007	WD	Normal	268000
130	0	0	0	0	NaN	NaN	NaN	0	6	2007	WD	Normal	269790
122	0	0	0	0	NaN	MnPrv	NaN	0	1	2010	COD	Normal	190000
0	0	0	0	0	NaN	NaN	NaN	0	6	2009	WD	Normal	215000

Data Information

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1168 entries, 0 to 1167
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1168 non-null   int64
 1   MSSubClass     1168 non-null   int64
 2   MSZoning       1168 non-null   object
 3   LotFrontage    954 non-null    float64
 4   LotArea        1168 non-null   int64
 5   Street         1168 non-null   object
 6   Alley          77 non-null     object
 7   LotShape       1168 non-null   object
 8   LandContour    1168 non-null   object
 9   Utilities      1168 non-null   object
 10  LotConfig      1168 non-null   object
 11  LandSlope      1168 non-null   object
 12  Neighborhood   1168 non-null   object
 13  Condition1     1168 non-null   object
 14  Condition2     1168 non-null   object
 15  BldgType       1168 non-null   object
 16  HouseStyle     1168 non-null   object
 17  OverallQual    1168 non-null   int64
 18  OverallCond    1168 non-null   int64
 19  YearBuilt      1168 non-null   int64
 20  YearRemodAdd   1168 non-null   int64
 21  RoofStyle      1168 non-null   object
 22  RoofMatl       1168 non-null   object
 23  Exterior1st    1168 non-null   object
 24  Exterior2nd    1168 non-null   object
 25  MasVnrType     1161 non-null   object
 26  MasVnrArea     1161 non-null   float64
 27  ExterQual      1168 non-null   object
 28  ExterCond      1168 non-null   object
 29  Foundation     1168 non-null   object
 30  BsmtQual       1138 non-null   object
 31  BsmtCond       1138 non-null   object
 32  BsmtExposure   1137 non-null   object
 33  BsmtFinType1   1138 non-null   object
 34  BsmtFinSF1     1168 non-null   int64
 35  BsmtFinType2   1137 non-null   object
 36  BsmtFinSF2     1168 non-null   int64
 37  BsmtUnfSF      1168 non-null   int64
 38  TotalBsmtSF    1168 non-null   int64
 39  Heating        1168 non-null   object
 40  HeatingQC      1168 non-null   object
 41  CentralAir     1168 non-null   object
```

```
42  Electrical     1168 non-null   object
43  1stFlrSF       1168 non-null   int64
44  2ndFlrSF       1168 non-null   int64
45  LowQualFinSF   1168 non-null   int64
46  GrLivArea      1168 non-null   int64
47  BsmtFullBath   1168 non-null   int64
48  BsmtHalfBath   1168 non-null   int64
49  FullBath       1168 non-null   int64
50  HalfBath       1168 non-null   int64
51  BedroomAbvGr   1168 non-null   int64
52  KitchenAbvGr   1168 non-null   int64
53  KitchenQual    1168 non-null   object
54  TotRmsAbvGrd   1168 non-null   int64
55  Functional     1168 non-null   object
56  Fireplaces     1168 non-null   int64
57  FireplaceQu    617 non-null    object
58  GarageType     1104 non-null   object
59  GarageYrBlt    1104 non-null   float64
60  GarageFinish   1104 non-null   object
61  GarageCars     1168 non-null   int64
62  GarageArea     1168 non-null   int64
63  GarageQual     1104 non-null   object
64  GarageCond     1104 non-null   object
65  PavedDrive     1168 non-null   object
66  WoodDeckSF     1168 non-null   int64
67  OpenPorchSF    1168 non-null   int64
68  EnclosedPorch  1168 non-null   int64
69  3SsnPorch      1168 non-null   int64
70  ScreenPorch    1168 non-null   int64
71  PoolArea       1168 non-null   int64
72  PoolQC         7 non-null      object
73  Fence          237 non-null    object
74  MiscFeature    44 non-null     object
75  MiscVal        1168 non-null   int64
76  MoSold         1168 non-null   int64
77  YrSold         1168 non-null   int64
78  SaleType       1168 non-null   object
79  SaleCondition  1168 non-null   object
80  SalePrice      1168 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 739.2+ KB
```

Keys: As we can see above all column are not having same not null values, mean dataset is having null value and we need to over them to remove it.

Target variable is SalePrice

Data Dtype:

Id	int64
MSSubClass	int64
MSZoning	object
LotFrontage	float64
LotArea	int64
Street	object
Alley	object
LotShape	object
LandContour	object
Utilities	object
LotConfig	object
LandSlope	object
Neighborhood	object
Condition1	object
Condition2	object
BldgType	object
HouseStyle	object
OverallQual	int64
OverallCond	int64
YearBuilt	int64
YearRemodAdd	int64
RoofStyle	object
RoofMatl	object
Exterior1st	object
Exterior2nd	object
MasVnrType	object
MasVnrArea	float64

ExterQual	object
ExterCond	object
Foundation	object
BsmtQual	object
BsmtCond	object
BsmtExposure	object
BsmtFinType1	object
BsmtFinSF1	int64
BsmtFinType2	object
BsmtFinSF2	int64
BsmtUnfSF	int64
TotalBsmtSF	int64
Heating	object
HeatingQC	object
CentralAir	object
Electrical	object
1stFlrSF	int64
2ndFlrSF	int64
LowQualFinSF	int64
GrLivArea	int64
BsmtFullBath	int64
BsmtHalfBath	int64
FullBath	int64
HalfBath	int64
BedroomAbvGr	int64
KitchenAbvGr	int64
KitchenQual	object
TotRmsAbvGrd	int64

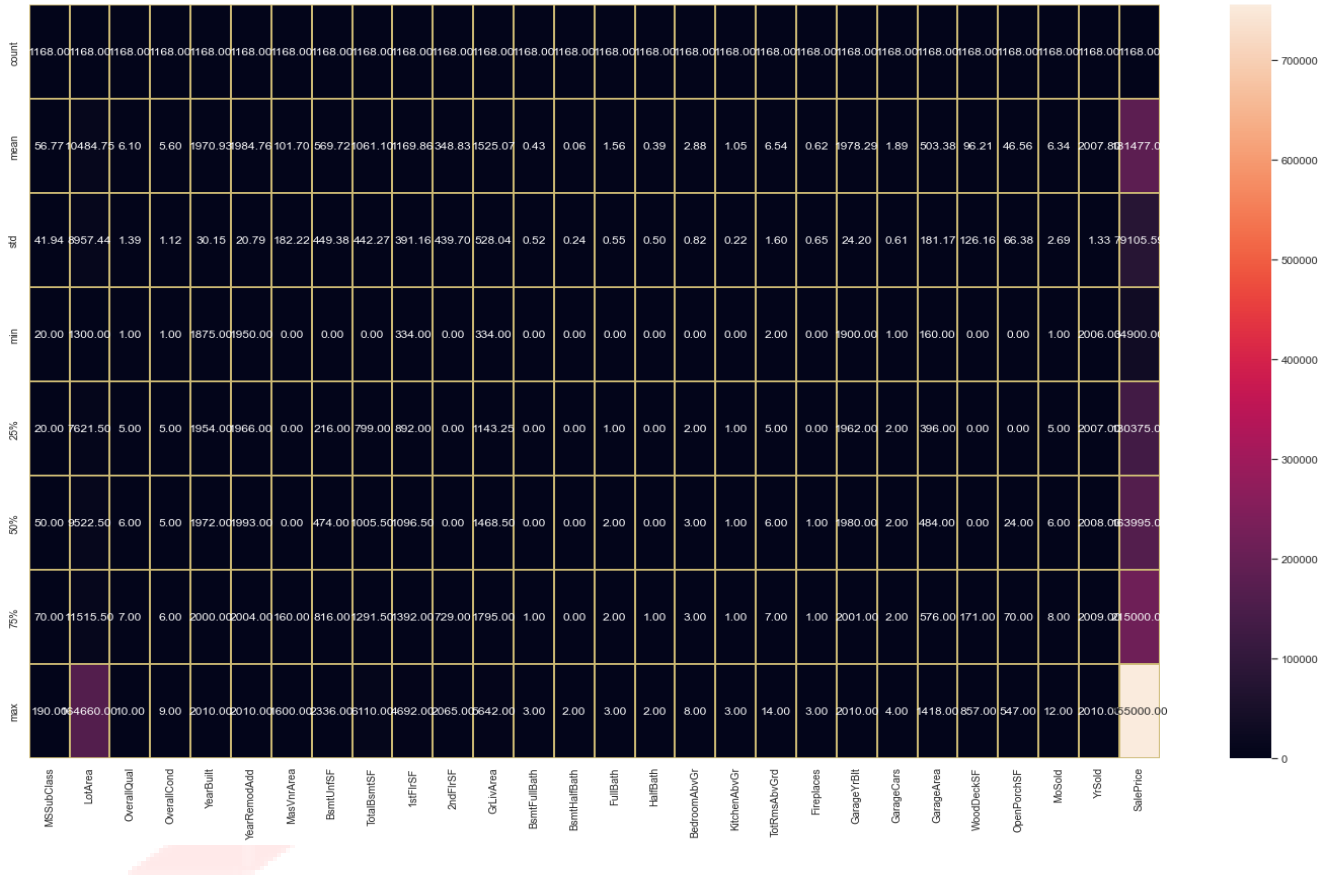
Functional	object
Fireplaces	int64
FireplaceQu	object
GarageType	object
GarageYrBlt	float64
GarageFinish	object
GarageCars	int64
GarageArea	int64
GarageQual	object
GarageCond	object
PavedDrive	object
WoodDeckSF	int64
OpenPorchSF	int64
EnclosedPorch	int64
3SsnPorch	int64
ScreenPorch	int64
PoolArea	int64
PoolQC	object
Fence	object
MiscFeature	object
MiscVal	int64
MoSold	int64
YrSold	int64
SaleType	object
SaleCondition	object
SalePrice	int64
dtype: object	

Total 81 columns we have received by data, in which 80 are feature columns and last one is target column (saleprize).

As we to have find the house price and target variable is in numeric form so it is any regression problem.

Analytical Problem Framing

- **Describe dataset:**



i) As every column is showing high difference between mean and 50% percentile, which indicating the skewness of the column

ii) Very high difference found between min and max value of kilometre and prize columns.

iii) Mean and 50 percentile value is having some value difference, mean these columns are having skewness

iv) Count for each column is same mean no null value present.

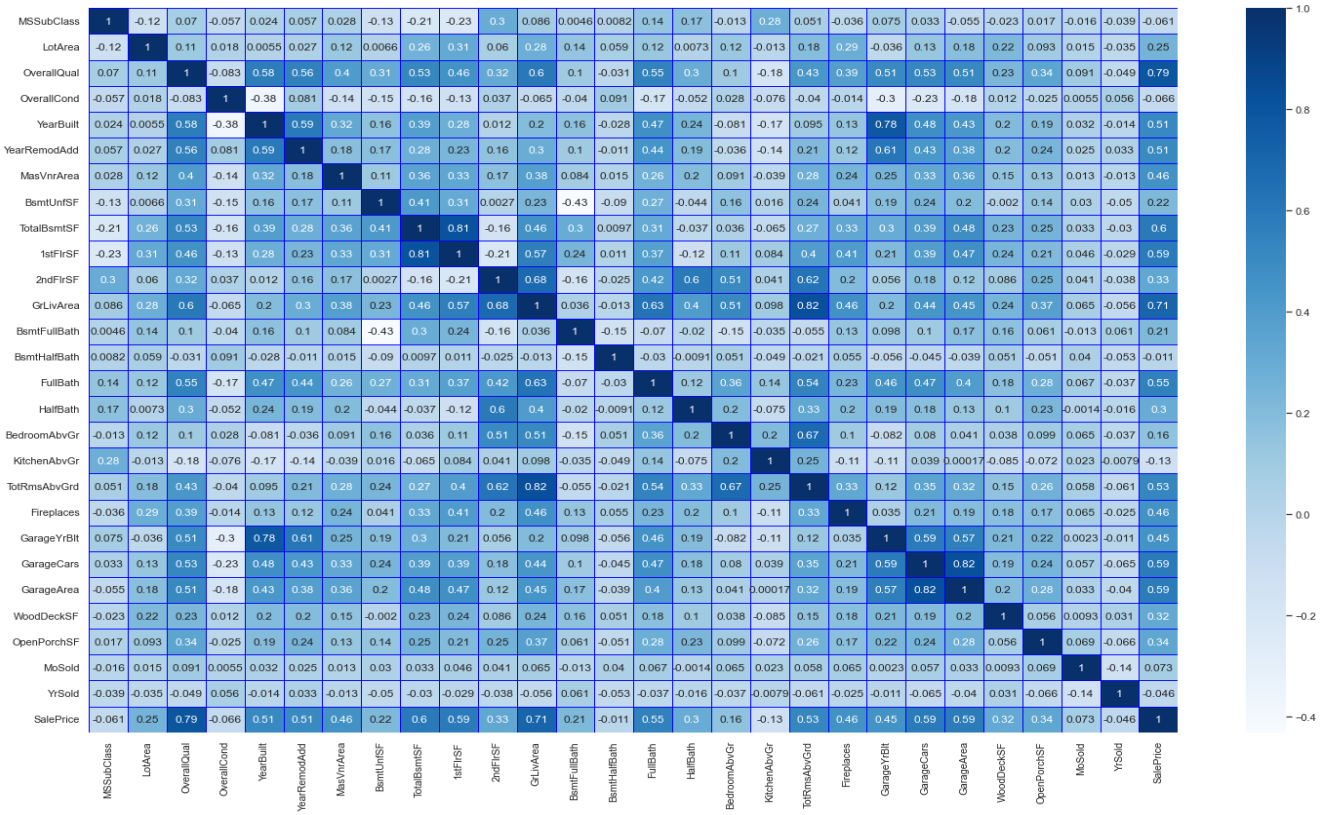
v) 25% and 75% is having difference, which also indicating , column spread.

The statistical figure (above mentioned) get to know by the df.describe() so many information, like min, max, standard deviation, 25 percentile, 50th percentile, 75 percentile.

Then by the help of correlation function I get to know the correlation of each columns with each other.

From the heat map one can visualized to see them clearly that they are positive correlated or the negative correlated, the dark side shows the negative correlation among each other and lighter side represents the positive correlation among the each other.

- **Correlation :**



i) Darker cell is having high value of correlation where light colour cell is having less value of correlation.

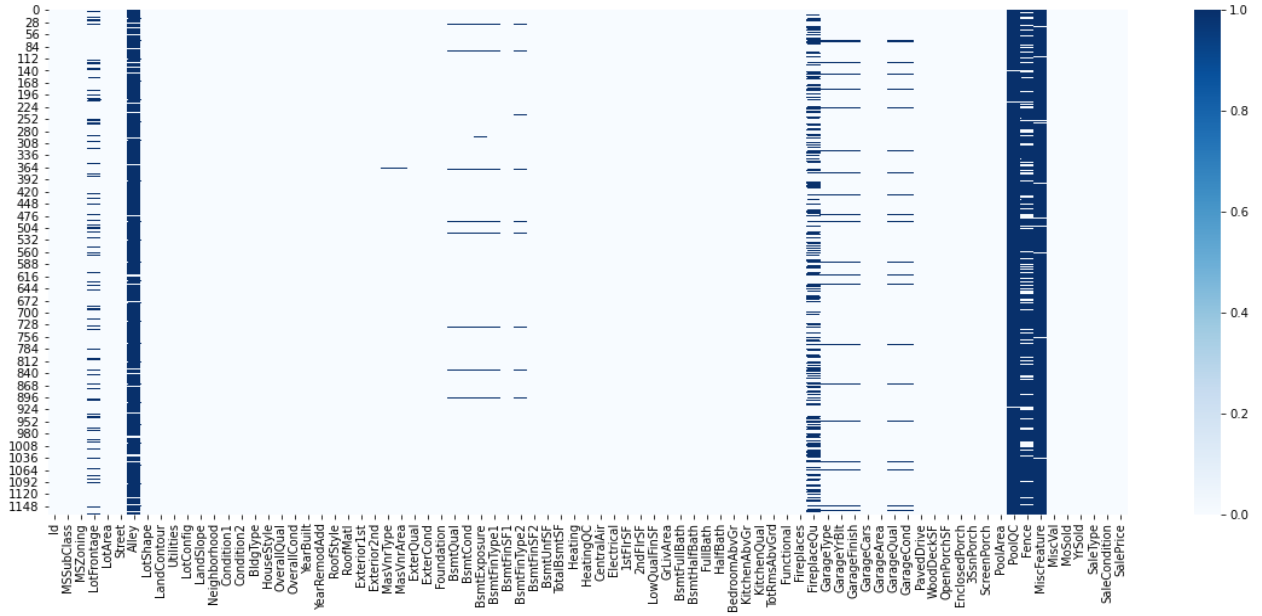
ii) Numeric columns are not much correlated with each other, as they are showing less value of correlation

iii) As few column are showing dark shade , which indicating high correlation, it would be helpful to build a good model of machine learning if we remove this relation by using variance inflation factor

iv) VIF is a technique used to remove multicollinearity

➢ Data Pre-processing:

i. Checked Null values:



```
train_null_value_per = train_df.isnull().sum() / train_df.shape[0] * 100
train_null_value_per[train_null_value_per > 15]
```

```
LotFrontage    18.321918
Alley          93.407534
FireplaceQu    47.174658
PoolQC         99.400685
Fence          79.708904
MiscFeature    96.232877
dtype: float64
```

```
# As these columns are having very high null values , therefore need to delete these columns from both
# daasets train and test
print(train_df.shape)
print(test_df.shape)
train_df.drop(columns = ['LotFrontage', 'Alley', 'FireplaceQu', 'PoolQC', 'Fence', 'MiscFeature'], inplace = True)
test_df.drop(columns = ['LotFrontage', 'Alley', 'FireplaceQu', 'PoolQC', 'Fence', 'MiscFeature'], inplace = True)
print(train_df.shape)
print(test_df.shape)
```

```
(1168, 81)
(292, 80)
(1168, 75)
(292, 74)
```

Keys: As these above mentioned columns are showing null value which is more than 15%, therefore I have decided to remove these column. As they are containing high null values

ii. Checked relation of columns then imputed values accordingly, maximum object type columns null values are replace with mode of the column.

```
train_df[train_df['MasVnrType'] == 'None']['MasVnrArea'].head(10)
```

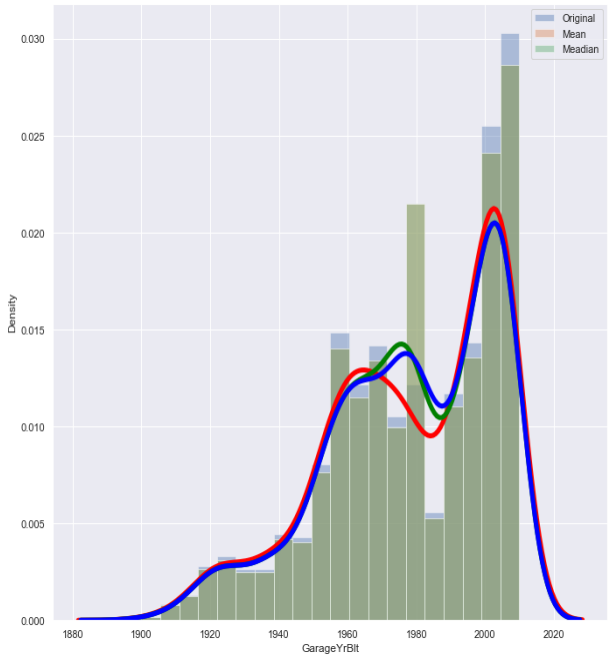
```
0     0.0
1     0.0
2     0.0
5     0.0
8     0.0
9     0.0
10    0.0
12    0.0
14    0.0
15    0.0
Name: MasVnrArea, dtype: float64
```

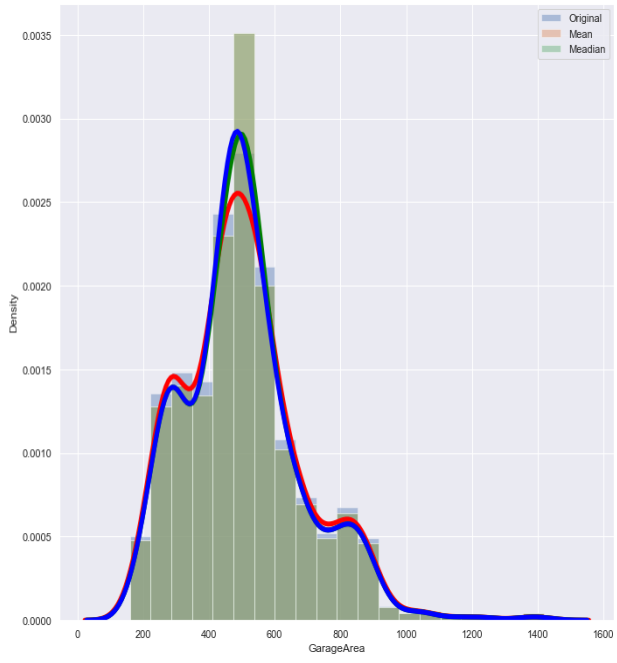
```
# As we can when MasVnrType column is None then MasVnrArea column has to be 0.0, thereforem we will impute
# values accordingly
train_df['MasVnrType'].mode()
# mode of MasVnrType is 'None', and we can impute mode value here as it is a categorical column
```

```
0    None
dtype: object
```

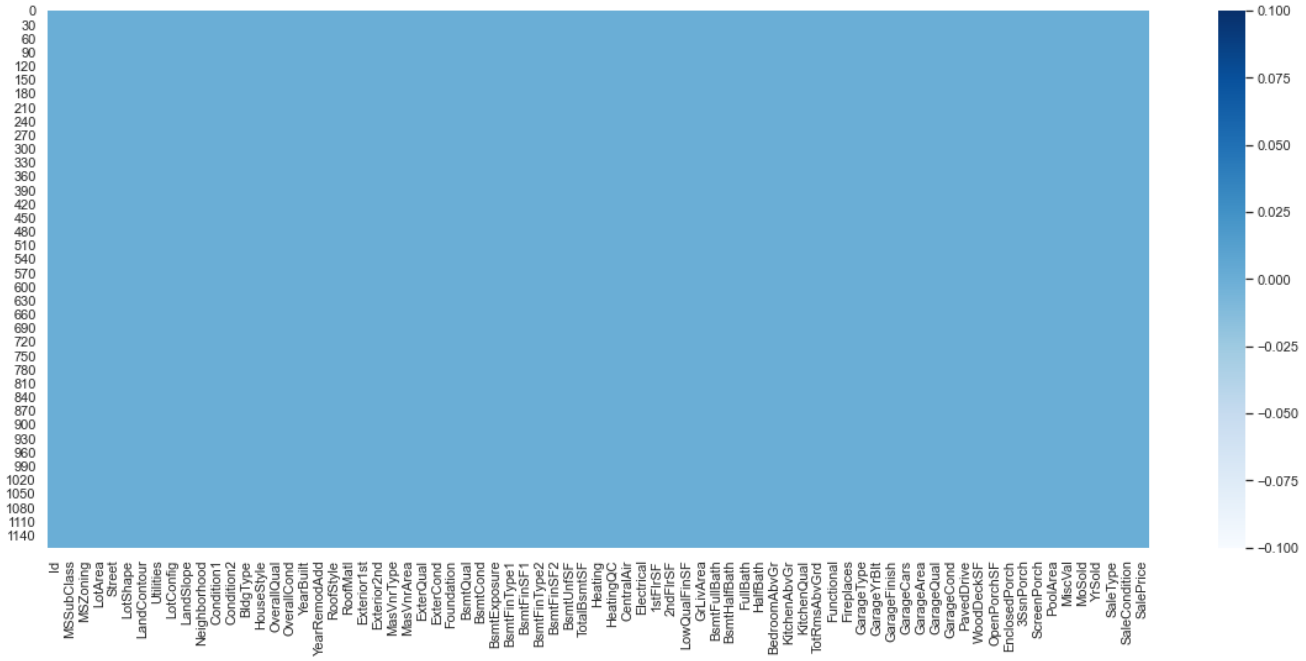
Note: This same process applied for each object type column, first checked relation, if found then imputed value accordingly else, mode value imputed

iii. Numerical type columns, here, I have checked impact of mean and median imputation then impute value.





iv. Again checked Null values



v. Few columns values has been divided into some range group and replace their value with range group.

Column names are given below:
BsmtFinSF1
BsmtFinSF2
LowQualFinSF
EnclosedPorch
3SsnPorch
ScreenPorch
PoolArea
MiscVal

Applying range opeartion for BsmtFinSF1

```
print( 'Min: ', train_df['BsmtFinSF1'].min())
print( 'Max: ', train_df['BsmtFinSF1'].max())
print('Lenght: ', len(train_df['BsmtFinSF1']))
```

```
Min:  0
Max:  5644
Lenght:  1168
```

```
BsmtFinSF1_class = []

for i in train_df['BsmtFinSF1']:
    if i in range (0, 101):
        BsmtFinSF1_class.append('0-100')

    elif i in range (101, 901):
        BsmtFinSF1_class.append('101-900')

    elif i in range(901,1801):
        BsmtFinSF1_class.append('901-1800')

    elif i in range(1801, 2701):
        BsmtFinSF1_class.append('1801-2700')

    elif i in range(2701, 3601):
        BsmtFinSF1_class.append('2701-3600')

    elif i in range(3601, 4501):
        BsmtFinSF1_class.append('3601-4500')

    elif i in range(4501, 5401):
        BsmtFinSF1_class.append('4501-5400')

    elif i in range(5401, 6000):
        BsmtFinSF1_class.append('5400+ ')
```

```
train_df['BsmtFinSF1_Class'] = BsmtFinSF1_class
```

Applying range opeartion for BsmtFinSF2

```
print( 'Min: ', train_df['BsmtFinSF2'].min())
print( 'Max: ', train_df['BsmtFinSF2'].max())
print('Lenght: ', len(train_df['BsmtFinSF2']))
```

```
Min:  0
Max:  1474
Lenght:  1168
```

```
BsmtFinSF2_class = []

for i in train_df['BsmtFinSF2']:
    if i in range (0, 101):
        BsmtFinSF2_class.append('0-100')
    elif i in range (101, 401):
        BsmtFinSF2_class.append('101-400')
    elif i in range(401,801):
        BsmtFinSF2_class.append('401-800')
    elif i in range(801, 1201):
        BsmtFinSF2_class.append('801-1200')
    elif i in range(1201, 2000):
        BsmtFinSF2_class.append('1200+')
len(BsmtFinSF2_class)
train_df['BsmtFinSF2_class'] = BsmtFinSF2_class
```

: 1168

Applying range opeartion for LowQualFinSF

```
print( 'Min: ', train_df['LowQualFinSF'].min())
print( 'Max: ', train_df['LowQualFinSF'].max())
print('Lenght: ', len(train_df['LowQualFinSF']))
```

```
Min:  0
Max:  572
Lenght:  1168
```

```
# for training dataset
LowQualFinSF_class = []

for i in train_df['LowQualFinSF']:
    if i in range (0, 1):
        LowQualFinSF_class.append('Zero')

    elif i in range (1, 151):
        LowQualFinSF_class.append('1-150')

    elif i in range(151,301):
        LowQualFinSF_class.append('151-300')

    elif i in range(301, 451):
        LowQualFinSF_class.append('301-450')

    elif i in range(451, 601):
        LowQualFinSF_class.append('451-600')
```

```
train_df['LowQualFinSF_class'] = LowQualFinSF_class
```

Applying range opeartion for EnclosedPorch

```
print( 'Min: ', train_df['EnclosedPorch'].min())
print( 'Max: ', train_df['EnclosedPorch'].max())
print('Lenght: ', len(train_df['EnclosedPorch']))
```

```
Min:  0
Max:  552
Lenght:  1168
```

```
# for training dataset
EnclosedPorch_class = []

for i in train_df['EnclosedPorch']:
    if i in range (0, 1):
        EnclosedPorch_class.append('Zero')

    elif i in range (1, 151):
        EnclosedPorch_class.append('1-150')

    elif i in range(151,301):
        EnclosedPorch_class.append('151-300')

    elif i in range(301, 451):
        EnclosedPorch_class.append('301-450')

    elif i in range(451, 601):
        EnclosedPorch_class.append('451-600')
```

```
train_df['EnclosedPorch_class'] = EnclosedPorch_class
```

Applying range opeartion for 3SsnPorch

```
print( 'Min: ', train_df['3SsnPorch'].min())
print( 'Max: ', train_df['3SsnPorch'].max())
print('Lenght: ', len(train_df['3SsnPorch']))
```

```
Min:  0
Max:  508
Lenght:  1168
```

```
# for training dataset
SsnPorch_class = []

for i in train_df['3SsnPorch']:
    if i in range (0, 1):
        SsnPorch_class.append('Zero')

    elif i in range (1, 151):
        SsnPorch_class.append('1-150')

    elif i in range(151,301):
        SsnPorch_class.append('151-300')

    elif i in range(301, 451):
        SsnPorch_class.append('301-450')

    elif i in range(451, 601):
        SsnPorch_class.append('451-600')
```

```
train_df['3SsnPorch_class'] = SsnPorch_class
```

Applying range opeartion for ScreenPorch

```
print( 'Min: ', train_df['ScreenPorch'].min())
print( 'Max: ', train_df['ScreenPorch'].max())
print('Lenght: ', len(train_df['ScreenPorch']))
```

```
Min:  0
Max:  480
Lenght:  1168
```

```
# for training dataset
ScreenPorch_class = []

for i in train_df['ScreenPorch']:
    if i in range (0, 1):
        ScreenPorch_class.append('Zero')

    elif i in range (1, 151):
        ScreenPorch_class.append('1-150')

    elif i in range(151,301):
        ScreenPorch_class.append('151-300')

    elif i in range(301, 451):
        ScreenPorch_class.append('301-450')

    elif i in range(451, 601):
        ScreenPorch_class.append('450+')
```

```
train_df['ScreenPorch_class'] = ScreenPorch_class
```

Applying range opeartion for PoolArea

```
print( 'Min: ', train_df['PoolArea'].min())
print( 'Max: ', train_df['PoolArea'].max())
print('Lenght: ', len(train_df['PoolArea']))
```

```
Min:  0
Max:  738
Lenght:  1168
```

```
# for training dataset
PoolArea_class = []

for i in train_df['PoolArea']:
    if i in range (0, 1):
        PoolArea_class.append('Zero')

    elif i in range (1, 151):
        PoolArea_class.append('1-150')

    elif i in range(151,301):
        PoolArea_class.append('151-300')

    elif i in range(301, 451):
        PoolArea_class.append('301-450')

    elif i in range(451, 601):
        PoolArea_class.append('450-600')

    elif i in range(601, 751):
        PoolArea_class.append('601-750')
```

```
train_df['PoolArea_class'] = PoolArea_class
```

Applying range opeartion for MiscVal

```
print( 'Min: ', train_df['MiscVal'].min())
print( 'Max: ', train_df['MiscVal'].max())
print('Lenght: ', len(train_df['MiscVal']))
```

```
Min:  0
Max:  15500
Lenght:  1168
```

```
# for training dataset
MiscVal_class = []

for i in train_df['MiscVal']:
    if i in range (0, 1):
        MiscVal_class.append('Zero')

    elif i in range (1, 1001):
        MiscVal_class.append('1-150')

    elif i in range(1001,6001):
        MiscVal_class.append('151-300')

    elif i in range(6001, 11001):
        MiscVal_class.append('301-450')

    elif i in range(11001, 16000):
        MiscVal_class.append('11001 -16000')

len(MiscVal_class)
```

```
1168
```

```
train_df['MiscVal_class'] = MiscVal_class
```

vi. One record of Exterior1st and Exterior2nd column was wrong updated, which was creating problem at the time of encoding

That wrong value replace with mode of that column

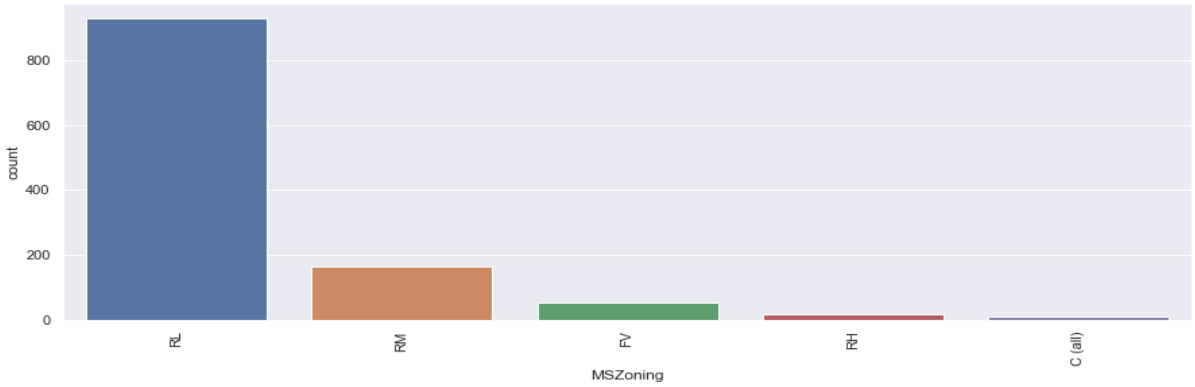
```
test_df['Exterior1st'] = test_df['Exterior1st'].replace ( 'CBlock', train_df['Exterior1st'].mode()[0])
test_df['Exterior2nd'] = test_df['Exterior2nd'].replace ( 'CBlock', train_df['Exterior2nd'].mode()[0])
```

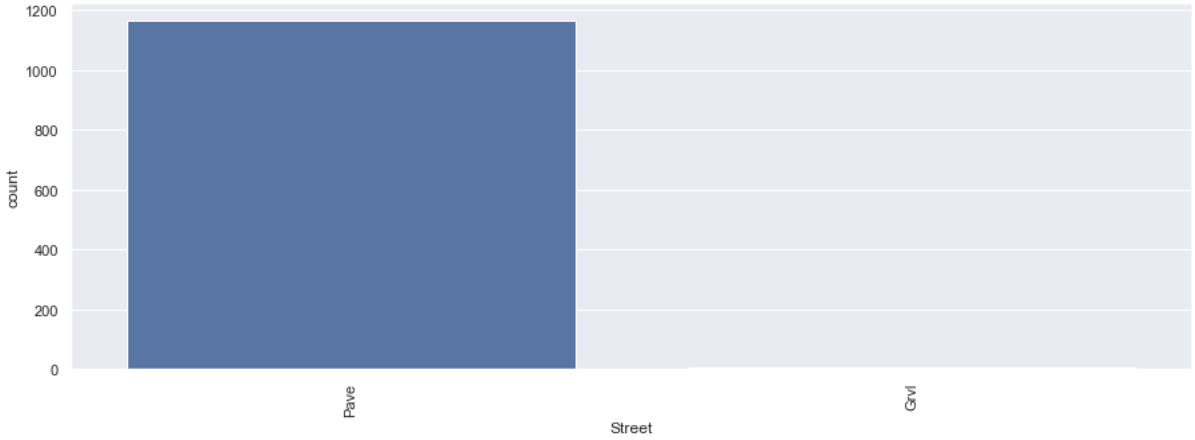
```
print(test_df['Exterior1st'].unique())
print('_____')
print(test_df['Exterior2nd'].unique())
```

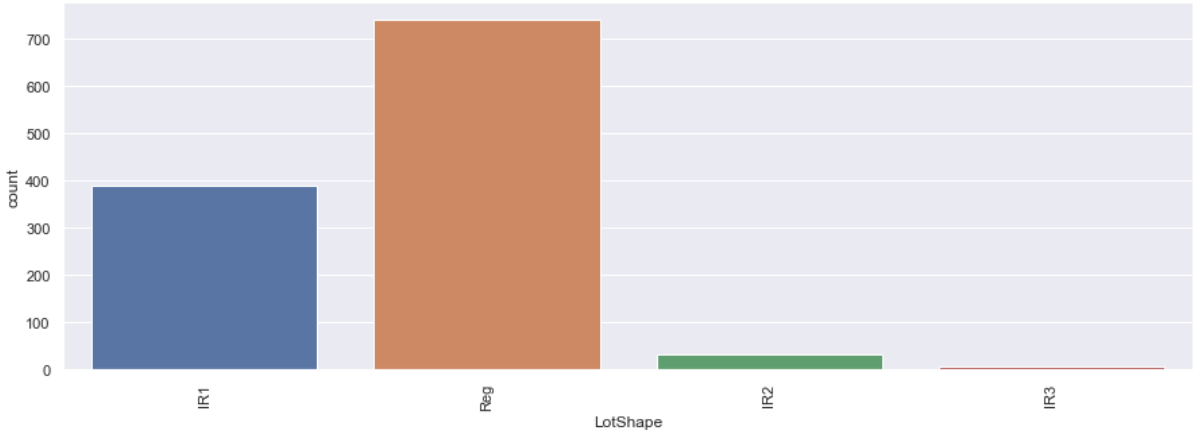
```
['VinylSd' 'HdBoard' 'MetalSd' 'CemntBd' 'Wd Sdng' 'Plywood' 'BrkFace'
 'AsbShng' 'WdShing' 'BrkComm' 'Stucco']
_____
['VinylSd' 'HdBoard' 'MetalSd' 'CmentBd' 'Wd Sdng' 'Plywood' 'Wd Shng'
 'Brk Cmn' 'ImStucc' 'AsbShng' 'BrkFace' 'Stone' 'Stucco']
```

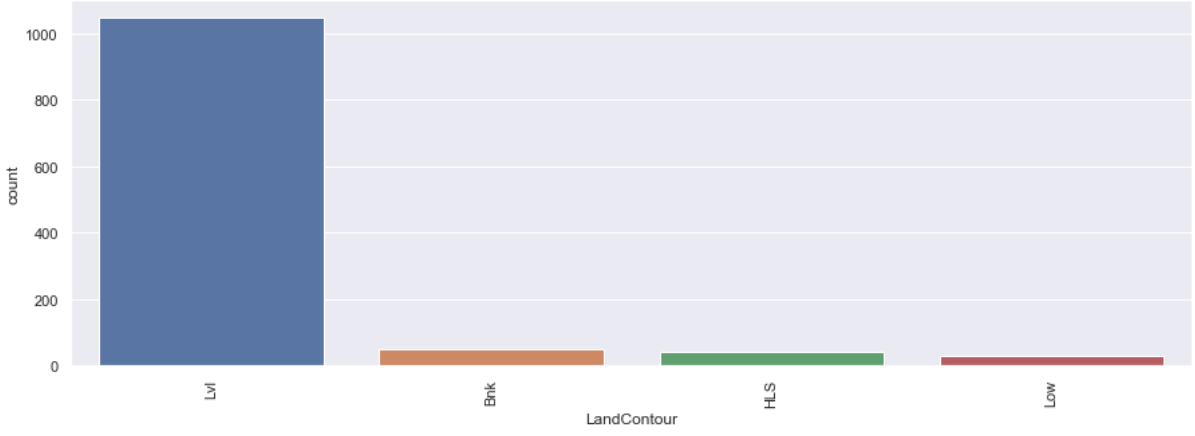
Visualization

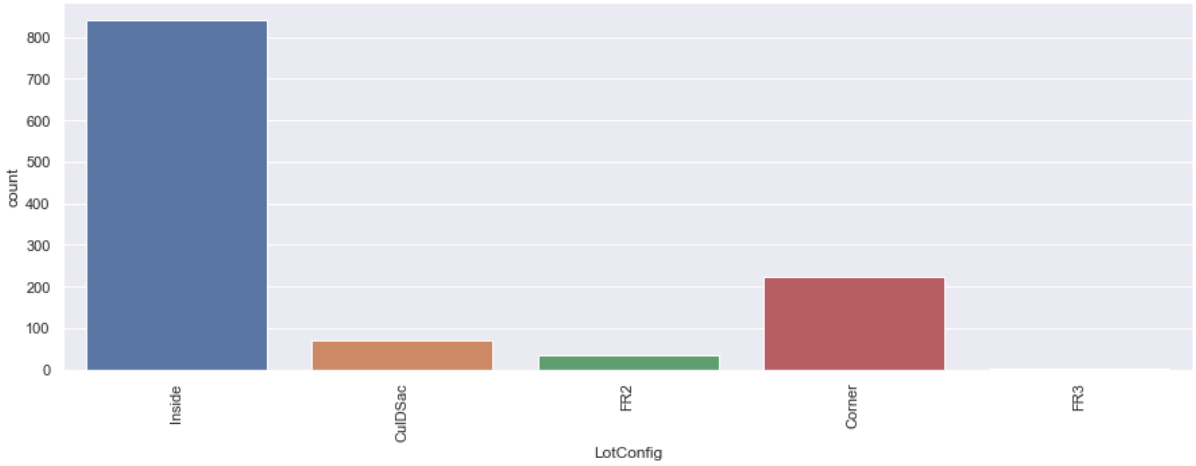
- **Bar Chart:**

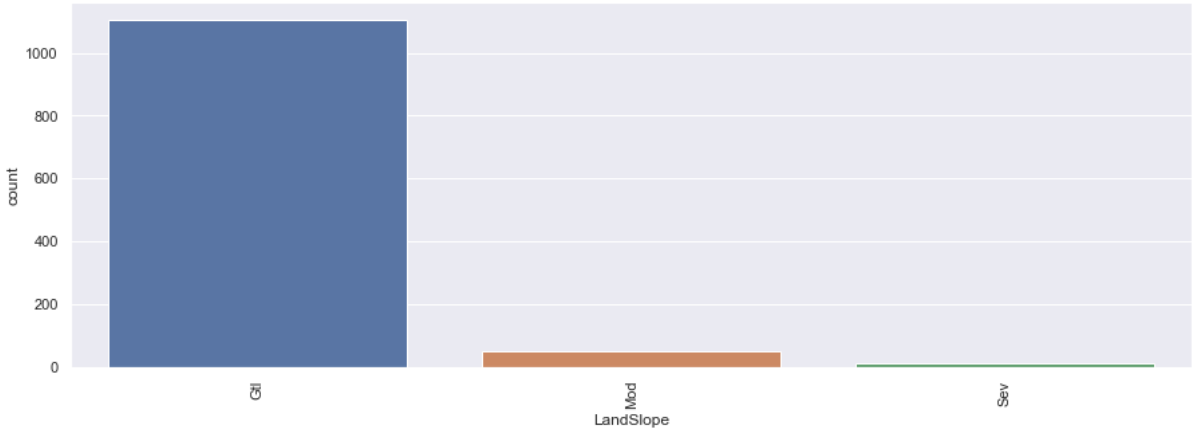

count
800
600
400
200
0
RL
RM
FV
RH
C (all)
MSZoning

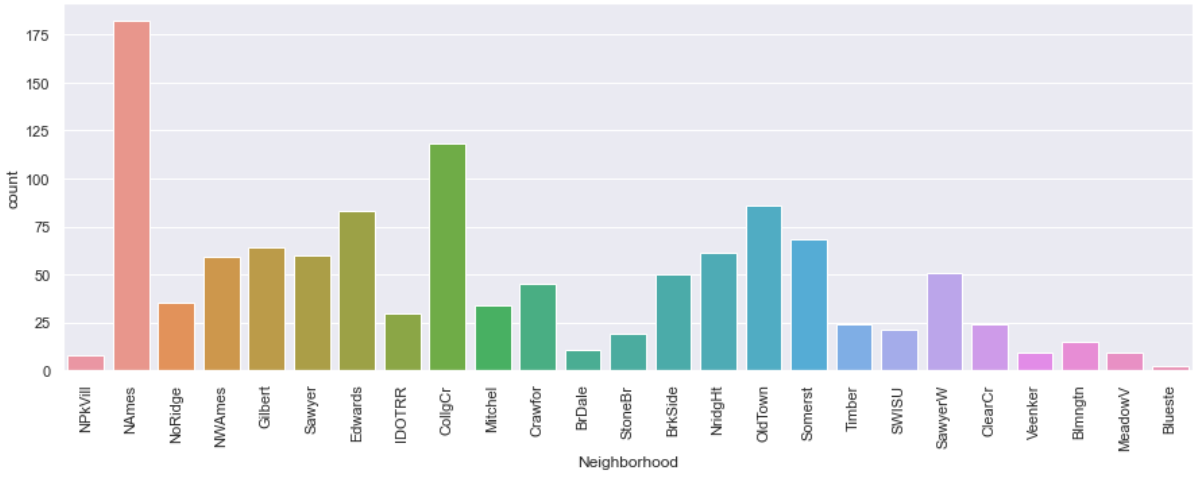

count
1200
1000
800
600
400
200
0
Pave
Grvl
Street

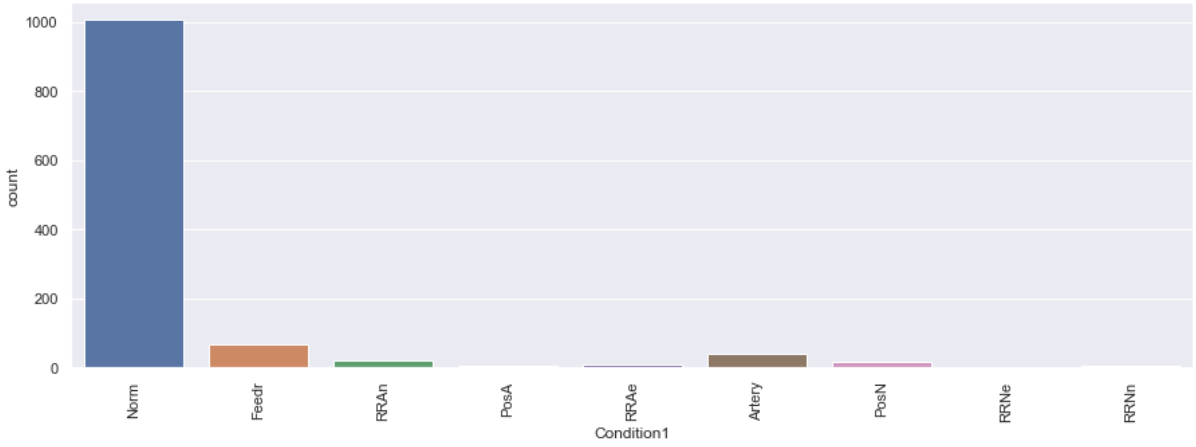

count
700
600
500
400
300
200
100
0
IR1
Reg
IR2
IR3
LotShape

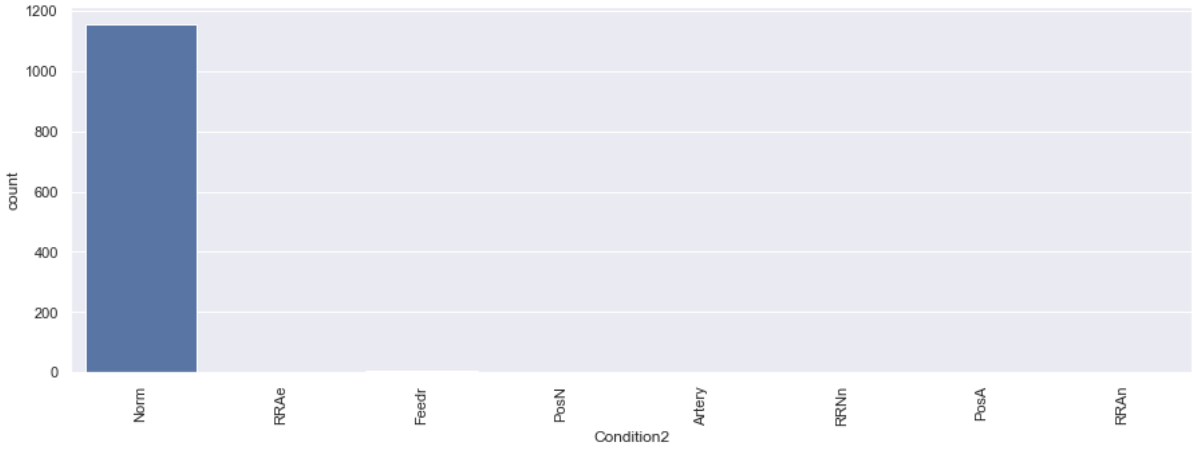

count
1000
800
600
400
200
0
Lvl
Bnk
HLS
Low
LandContour

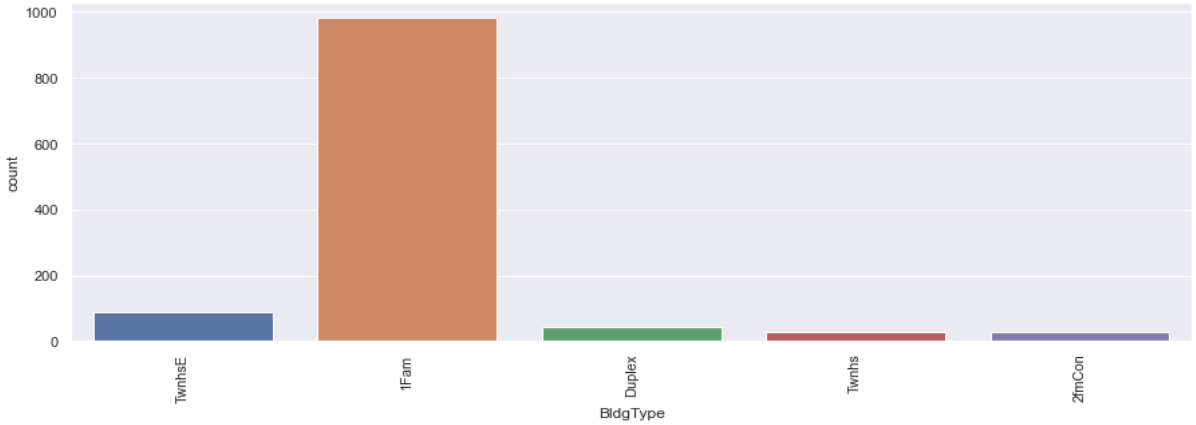

800
700
600
500
400
300
200
100
0
count
Inside
CulDSac
FR2
Corner
FR3
LotConfig

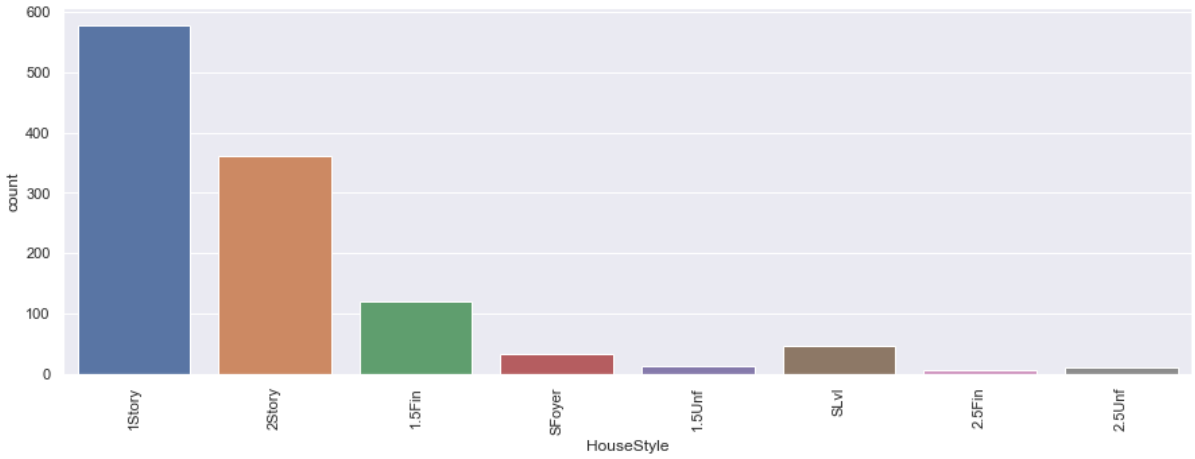

1000
800
600
400
200
0
count
Gtl
Mod
Sev
LandSlope

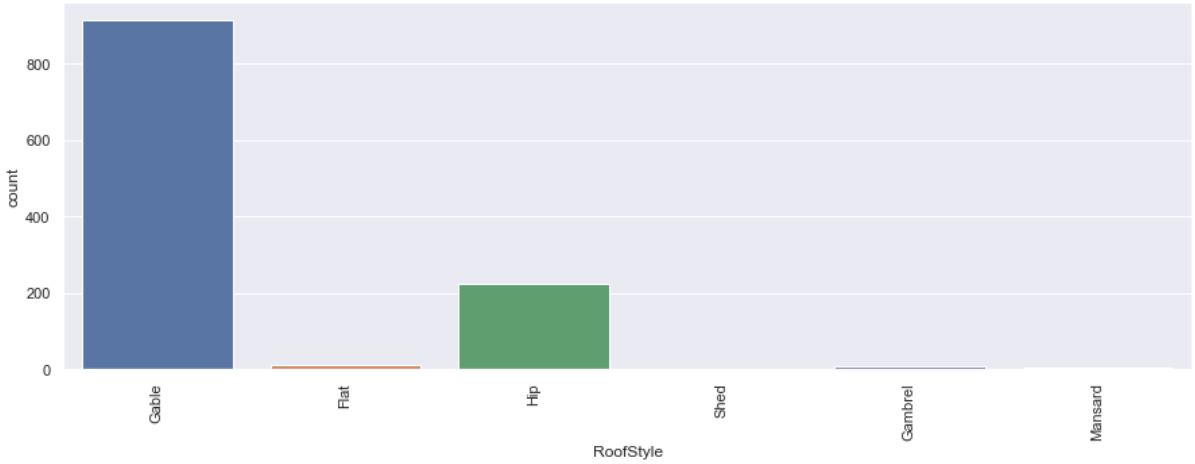

count
Neighborhood
NPkVill
NAmes
NoRidge
NWAmes
Gilbert
Sawyer
Edwards
IDOTRR
CollgCr
Mitchel
Crawfor
BrDale
StoneBr
BrkSide
NridgHt
OldTown
Somerst
Timber
SWISU
SawyerW
ClearCr
Veenker
Blmngtn
MeadowV
Blueste

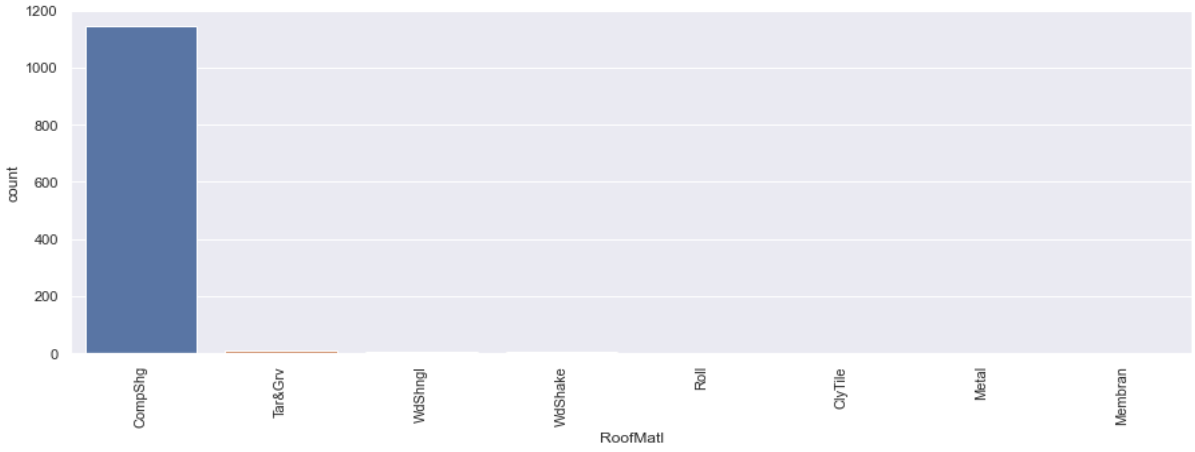

count
Condition1
Norm
Feedr
RRAn
PosA
RRAe
Artery
PosN
RRNe
RRNn

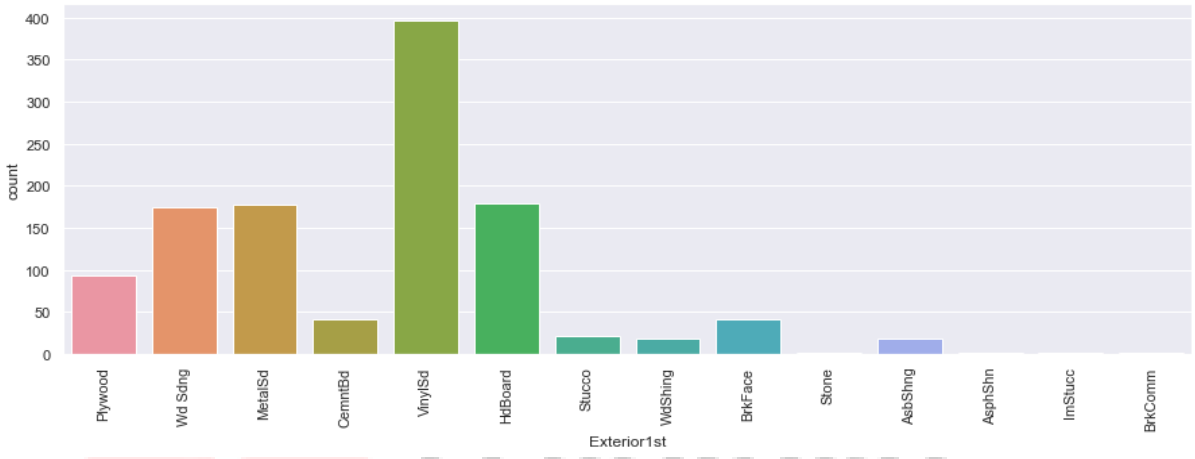

count
Condition2
Norm
RRAe
Feedr
PosN
Artery
RRNn
PosA
RRAn

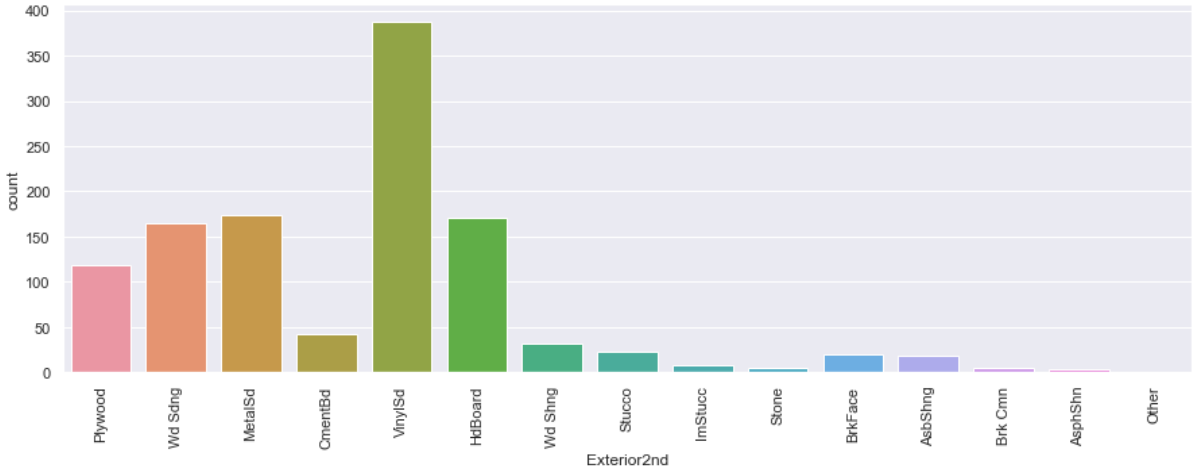

count
BldgType
TwnhsE
1Fam
Duplex
Twnhs
2fmCon

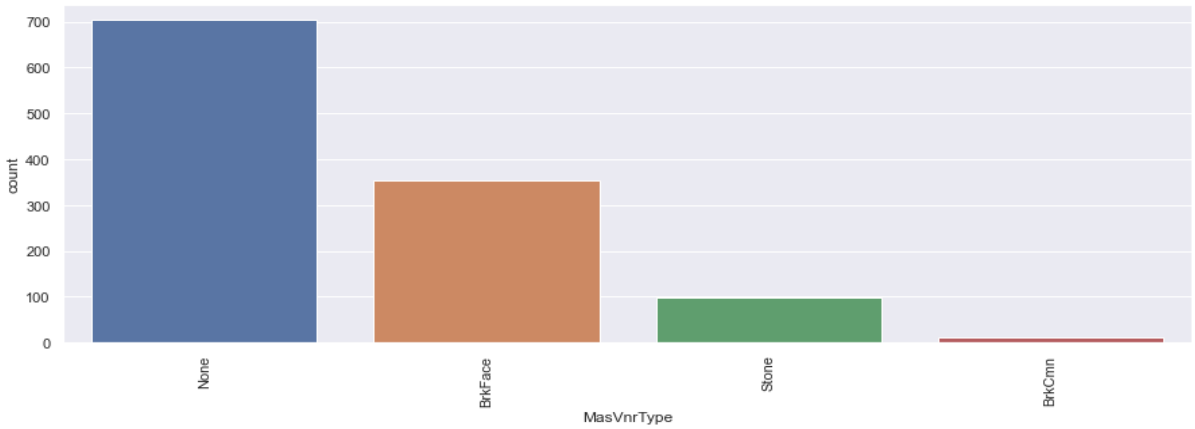

600
500
400
300
200
100
0
count
1Story
2Story
1.5Fin
SFoyer
1.5Unf
SLvl
2.5Fin
2.5Unf
HouseStyle

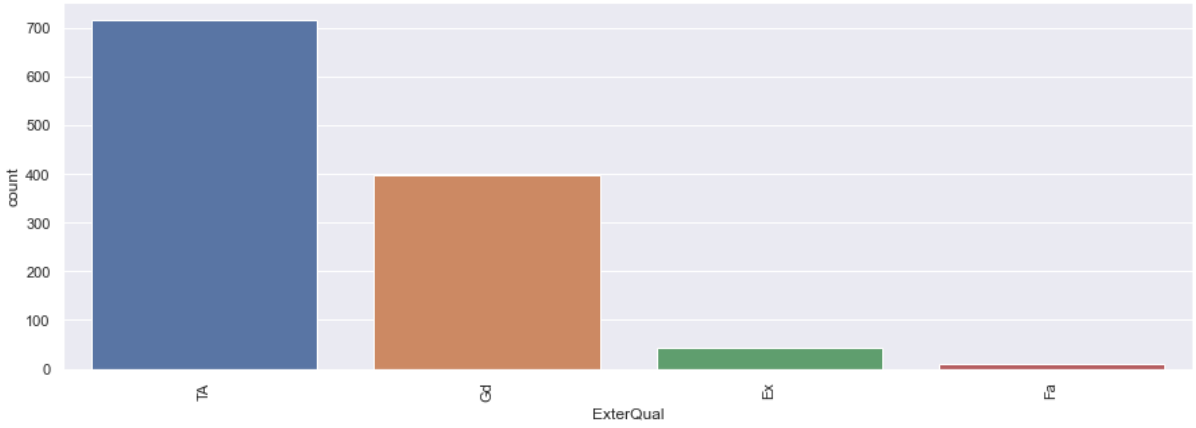

800
600
400
200
0
count
Gable
Flat
Hip
Shed
Gambrel
Mansard
RoofStyle

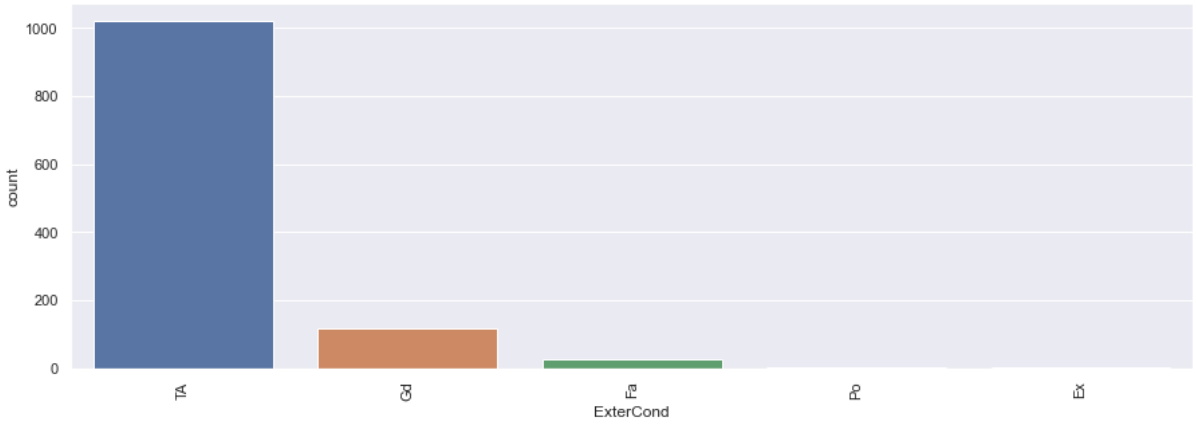

count
RoofMatl
CompShg
Tar&Grv
WdShngl
WdShake
Roll
ClyTile
Metal
Membran

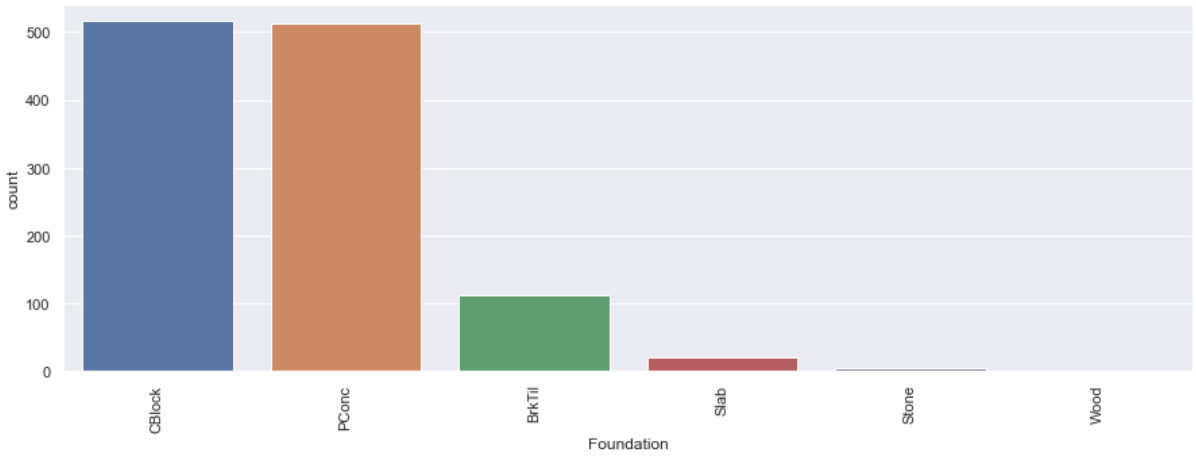

count
Exterior1st
Plywood
Wd Sdng
MetalSd
CemntBd
VinylSd
HdBoard
Stucco
WdShing
BrkFace
Stone
AsbShng
AsphShn
ImStucc
BrkComm

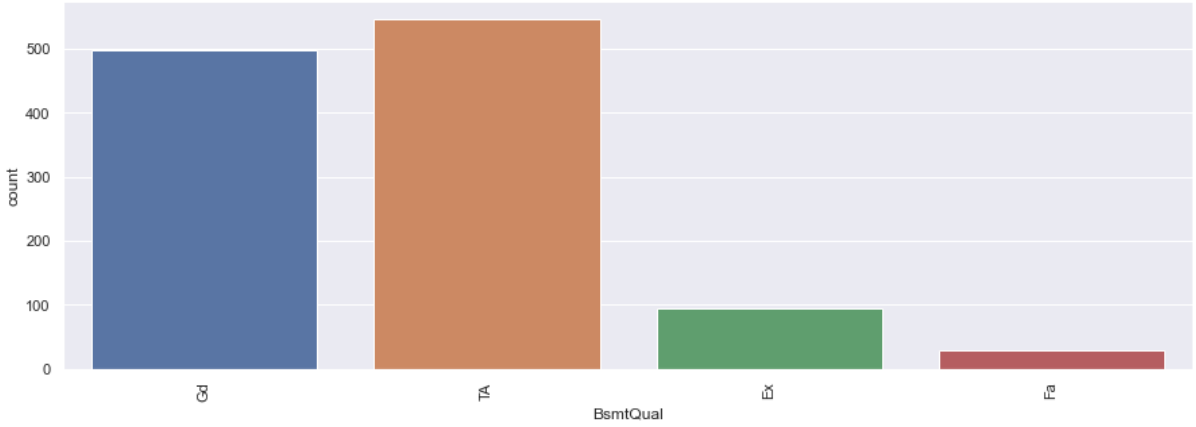

count
Exterior2nd
Plywood
Wd Sdng
MetalSd
CmentBd
VinylSd
HdBoard
Wd Shng
Stucco
ImStucc
Stone
BrkFace
AsbShng
Brk Cmn
AsphShn
Other

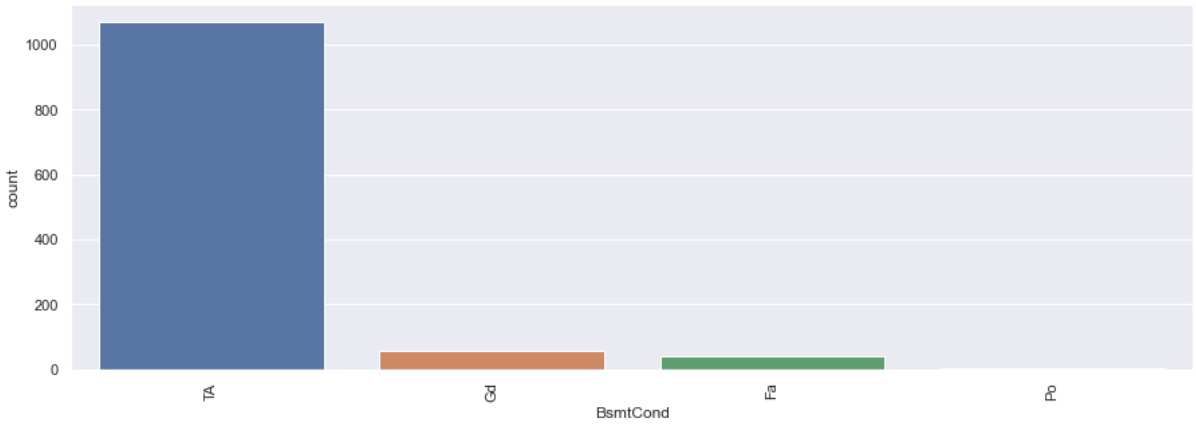

count
MasVnrType
None
BrkFace
Stone
BrkCmn

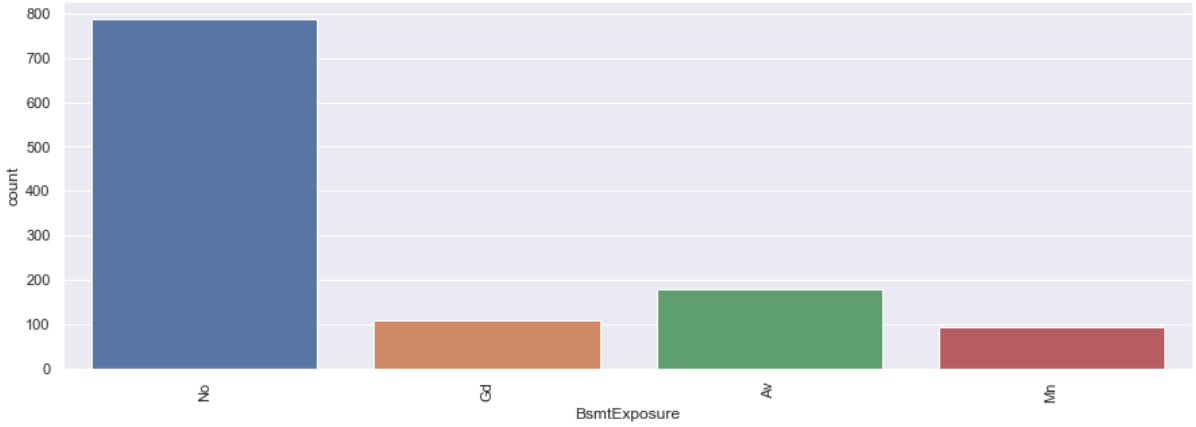

count
700
600
500
400
300
200
100
0
TA
Gd
Ex
Fa
ExterQual

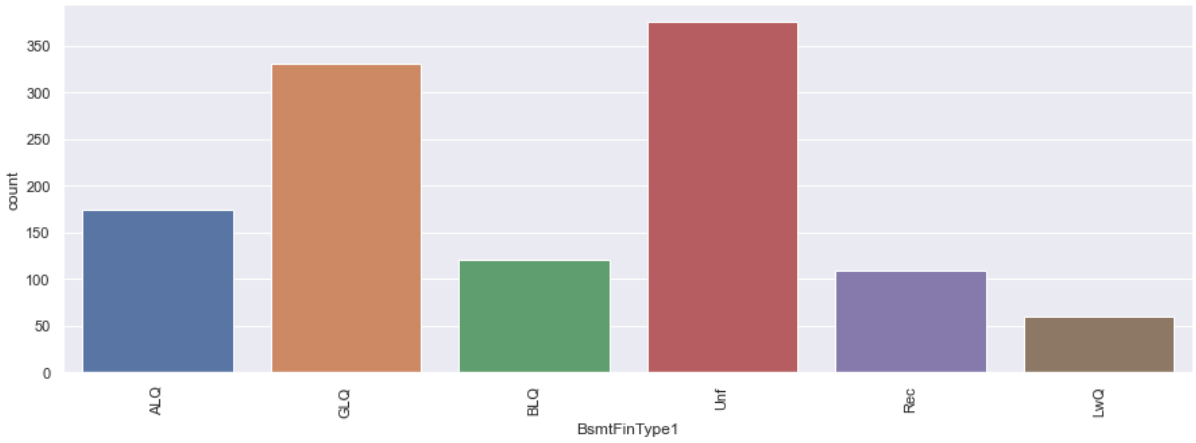

count
1000
800
600
400
200
0
TA
Gd
Fa
Po
Ex
ExterCond

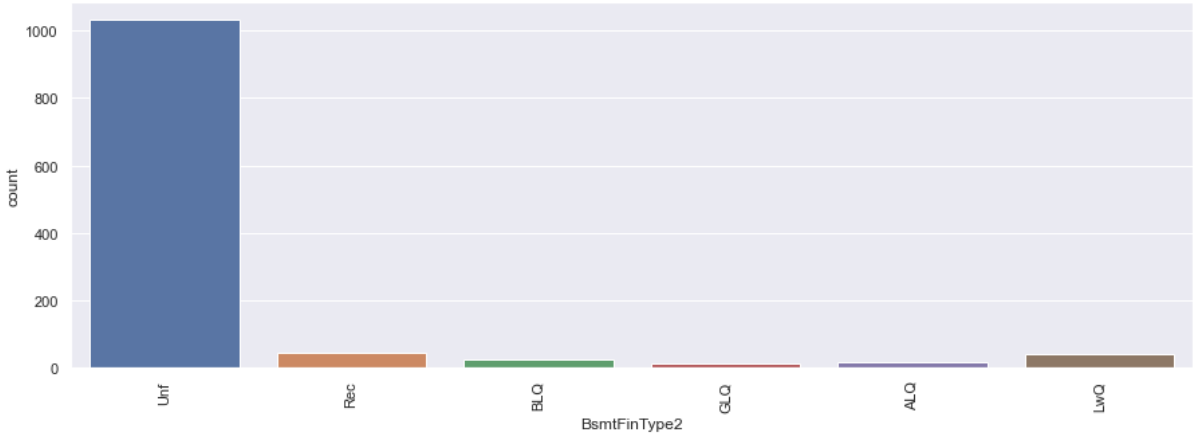

count
500
400
300
200
100
0
CBlock
PConc
BrkTil
Slab
Stone
Wood
Foundation

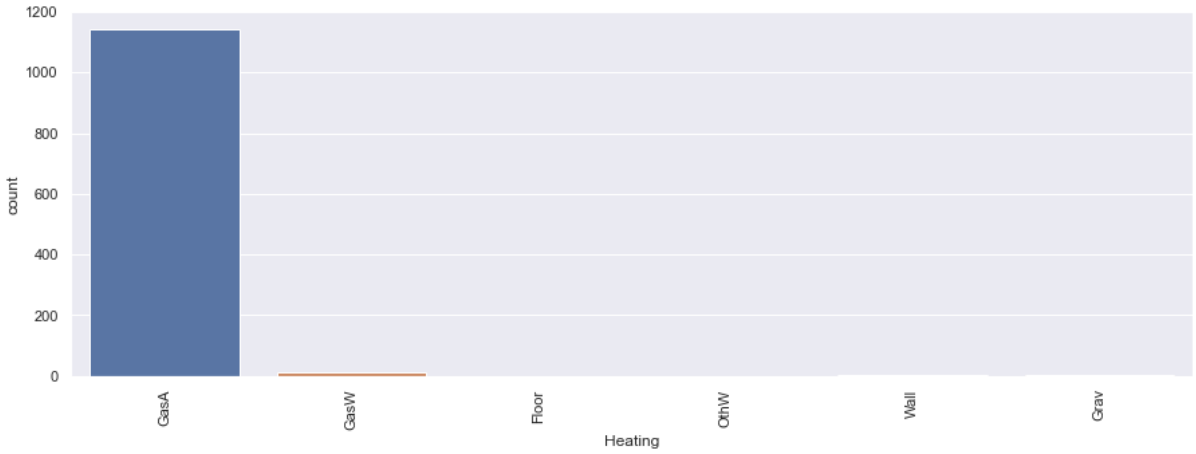

count
500
400
300
200
100
0
Gd
TA
Ex
Fa
BsmtQual

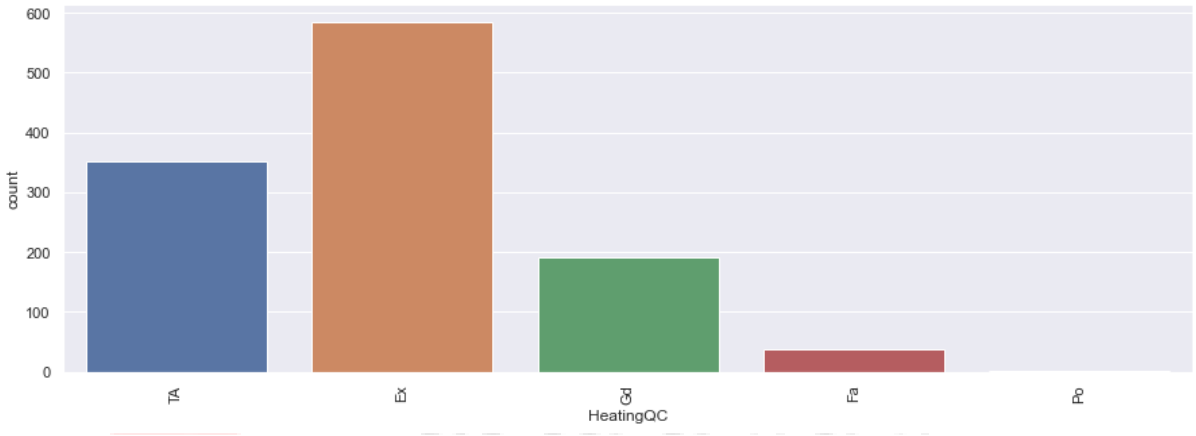

count
1000
800
600
400
200
0
TA
Gd
Fa
Po
BsmtCond

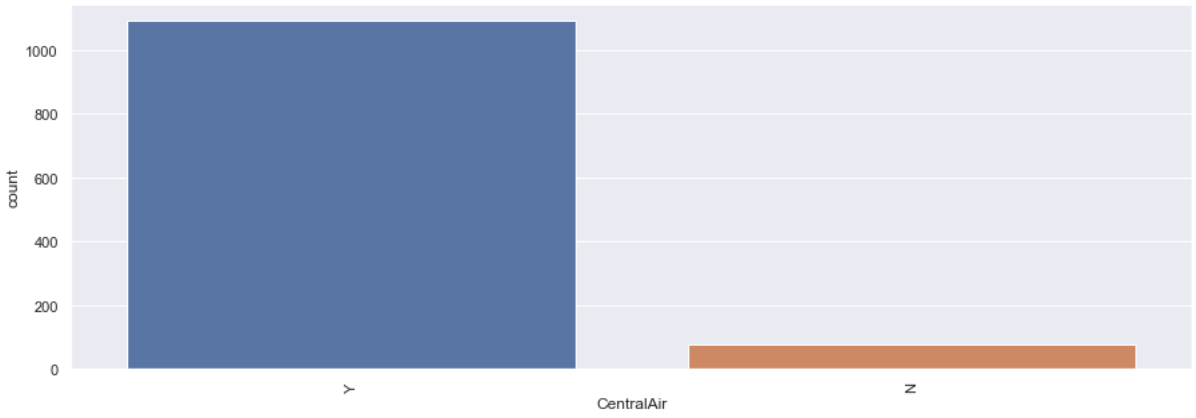

count
800
700
600
500
400
300
200
100
0
No
Gd
Av
Mn
BsmtExposure

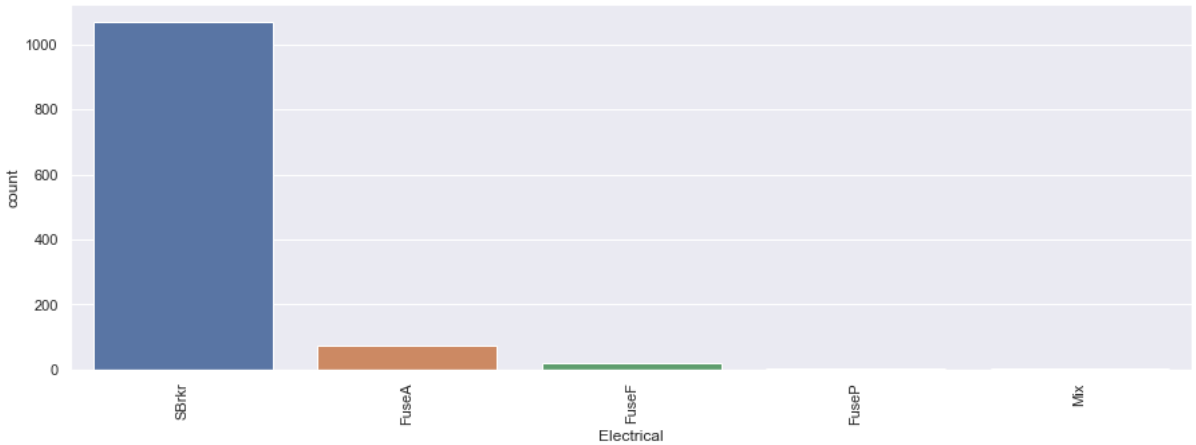

count
350
300
250
200
150
100
50
0
ALQ
GLQ
BLQ
Unf
Rec
LwQ
BsmtFinType1

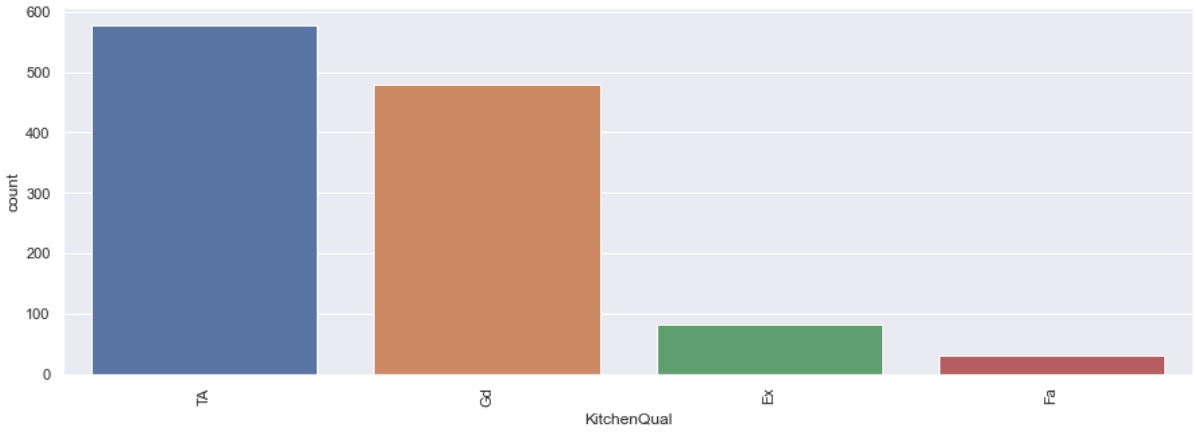

count
1000
800
600
400
200
0
Unf
Rec
BLQ
GLQ
ALQ
LwQ
BsmtFinType2

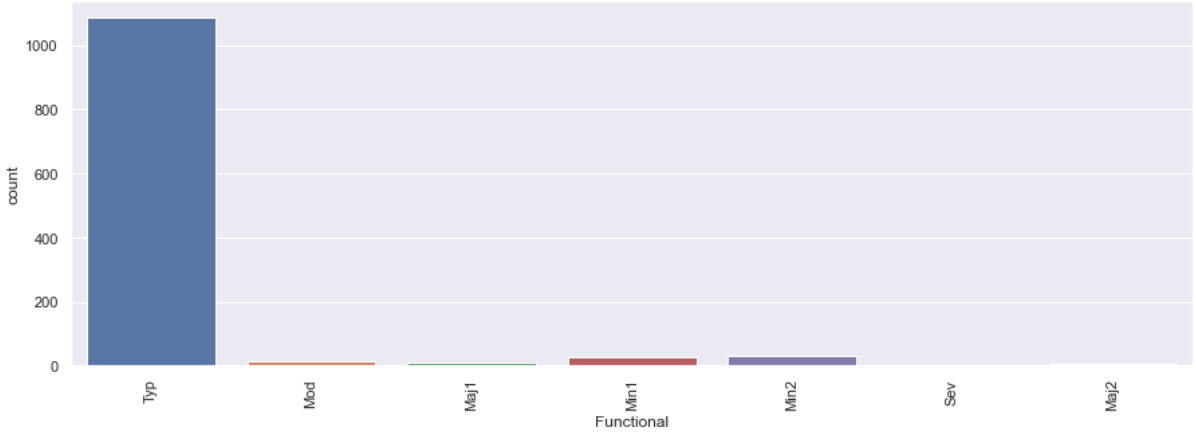

1200
1000
800
600
400
200
0
count
GasA
GasW
Floor
OthW
Wall
Grav
Heating

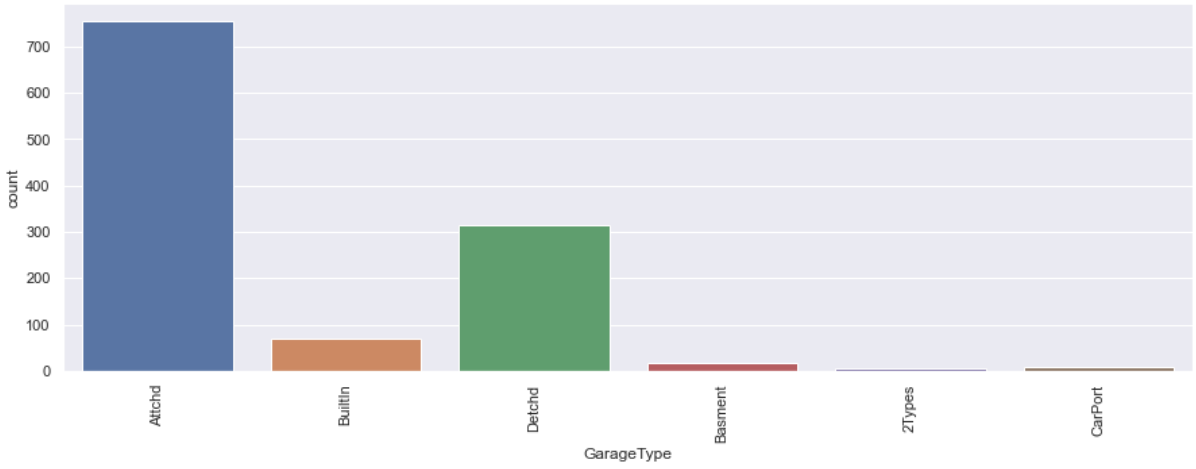

600
500
400
300
200
100
0
count
TA
Ex
Gd
Fa
Po
HeatingQC

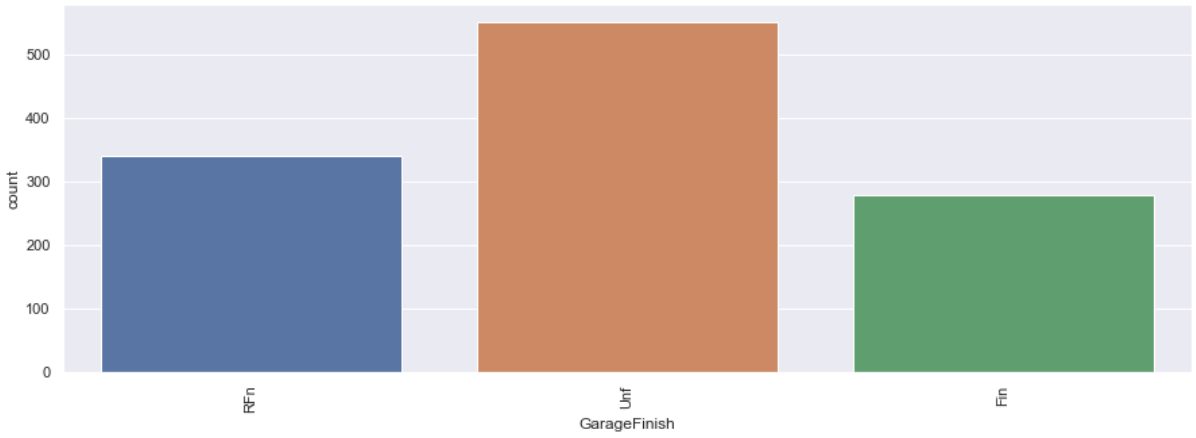

1000
800
600
400
200
0
count
Y
N
CentralAir

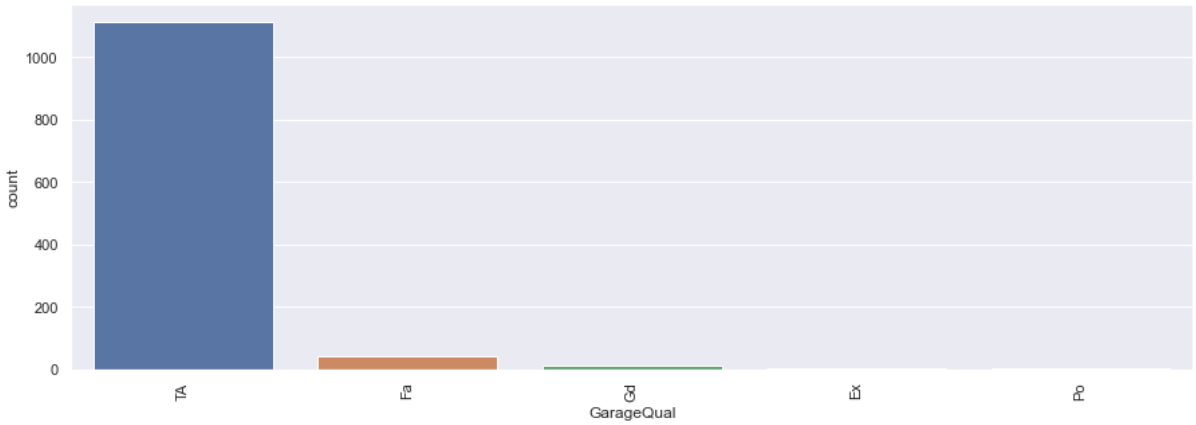

1000
800
600
400
200
0
count
SBrkr
FuseA
FuseF
FuseP
Mix
Electrical

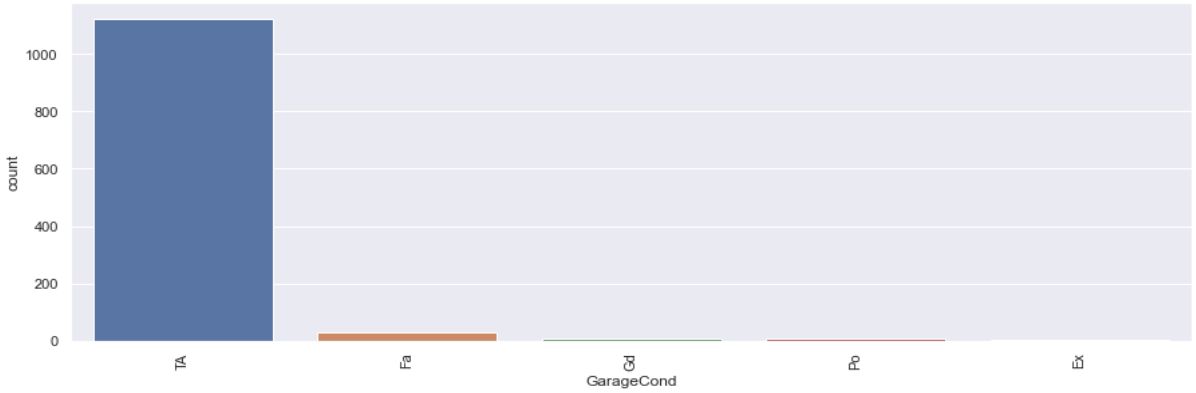

600
500
400
300
200
100
0
count
TA
Gd
Ex
Fa
KitchenQual

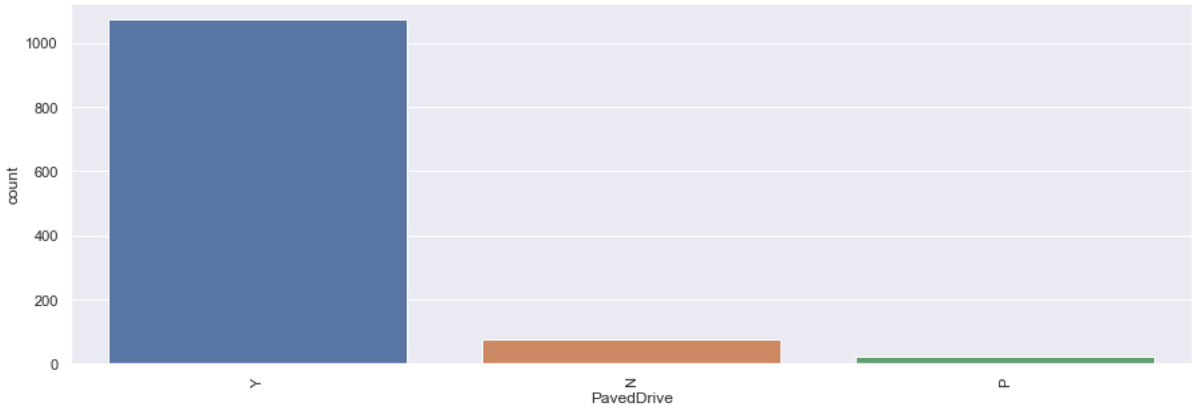

1000
800
600
400
200
0
count
Typ
Mod
Maj1
Min1
Min2
Sev
Maj2
Functional

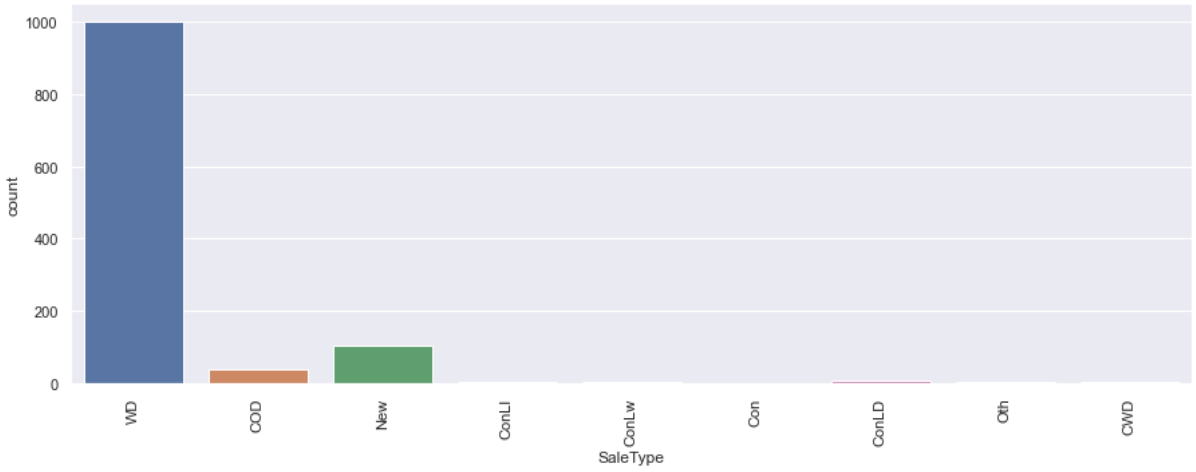

700
600
500
400
300
200
100
0
count
Attchd
BuiltIn
Detchd
Basment
2Types
CarPort
GarageType

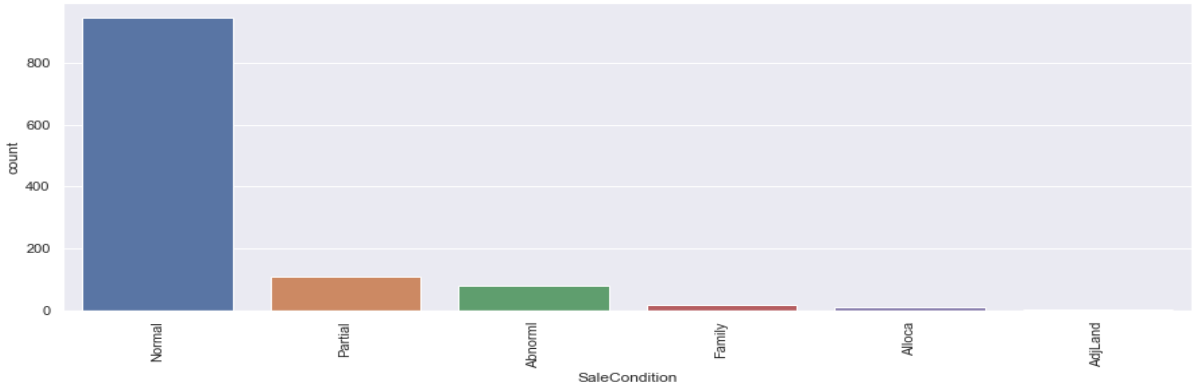

500
400
300
200
100
0
count
RFn
Unf
Fin
GarageFinish


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

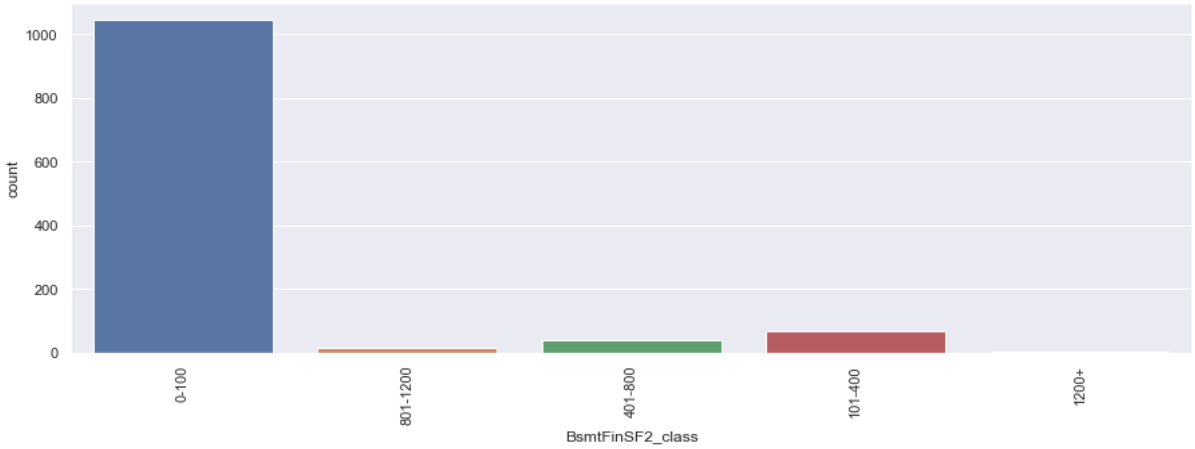

count
1000
800
600
400
200
0
TA
Fa
Gd
Po
Ex
GarageCond

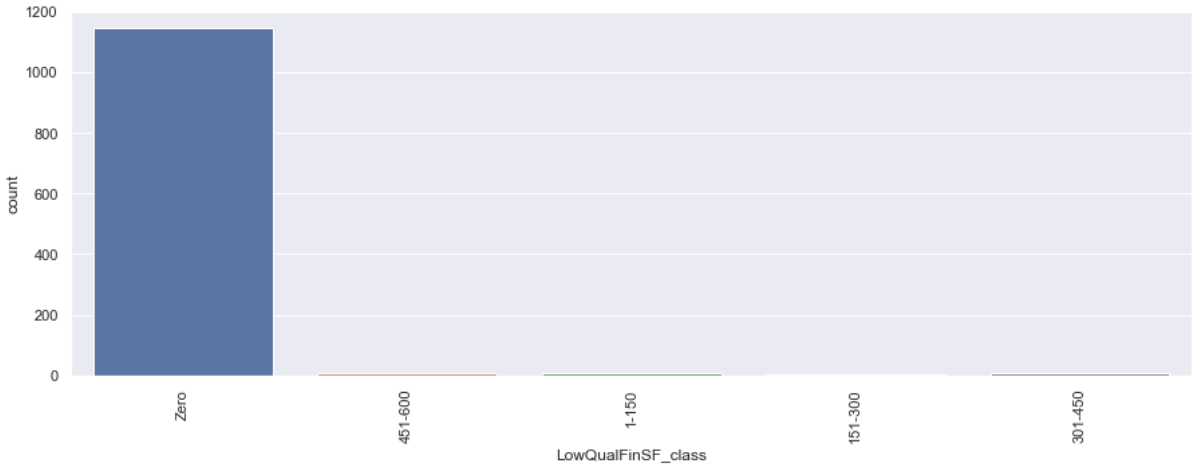

count
1000
800
600
400
200
0
Y
N
P
PavedDrive

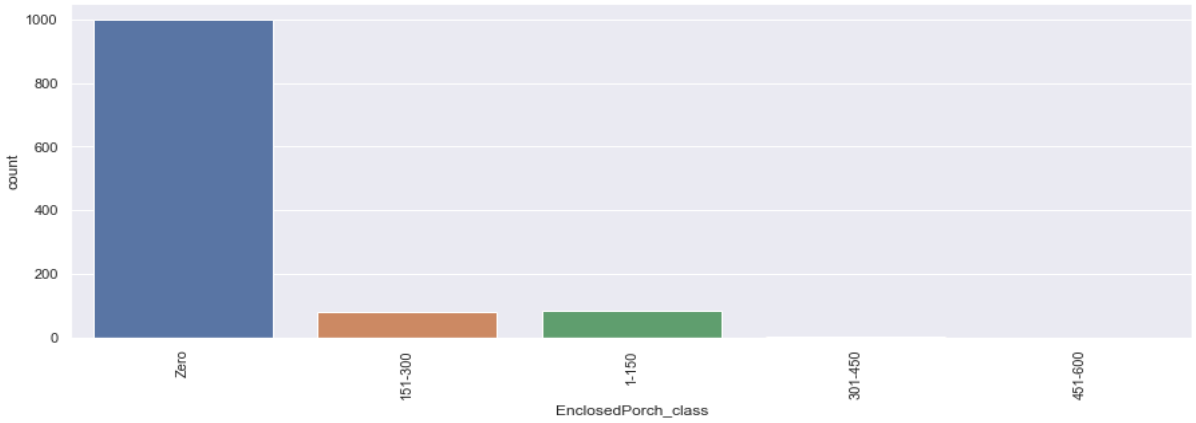

count
1000
800
600
400
200
0
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SaleType

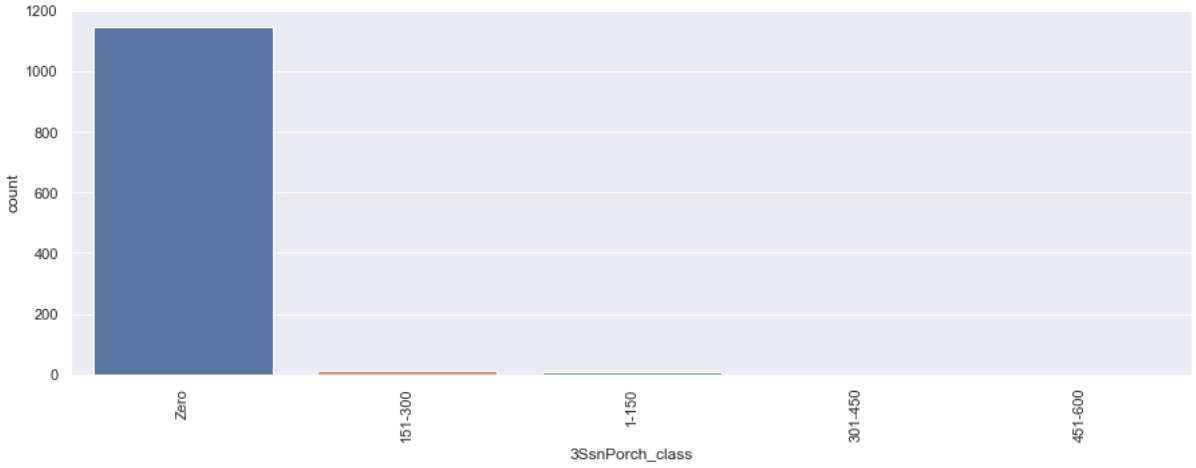

count
800
600
400
200
0
Normal
Partial
Abnorml
Family
Alloca
AdjLand
SaleCondition

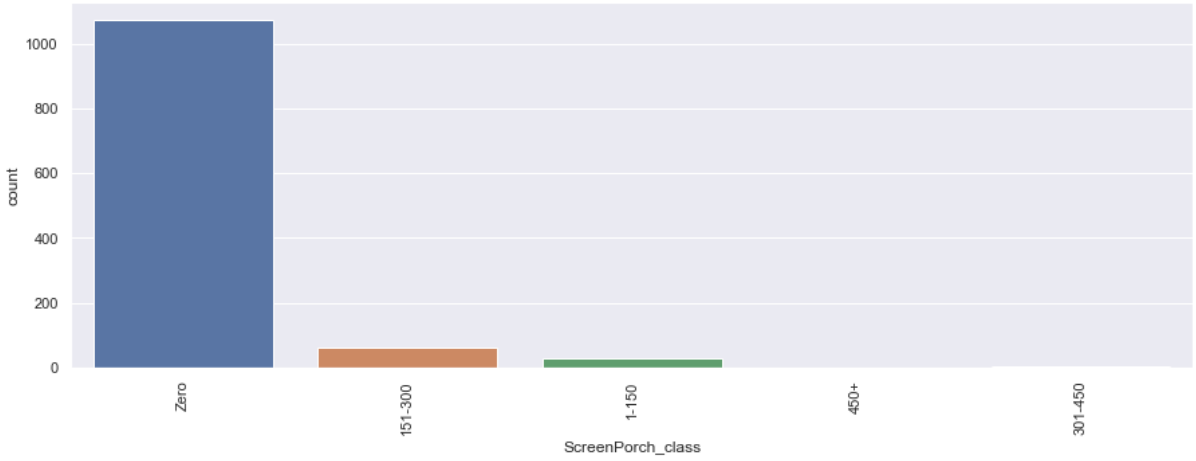

count
500
400
300
200
100
0
101-900
901-1800
0-100
1801-2700
5400+
BsmtFinSF1_Class

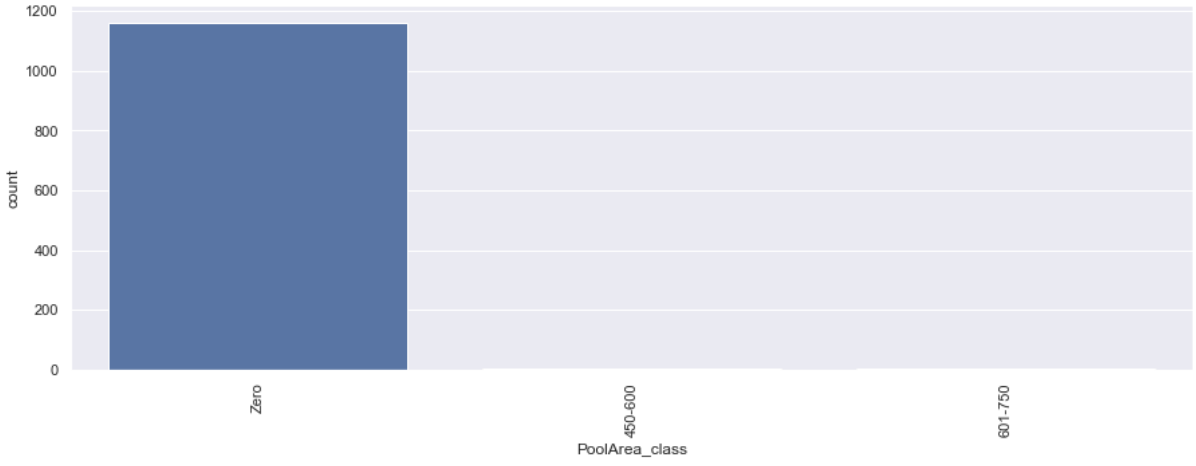

count
1000
800
600
400
200
0
0-100
801-1200
401-800
101-400
1200+
BsmtFinSF2_class

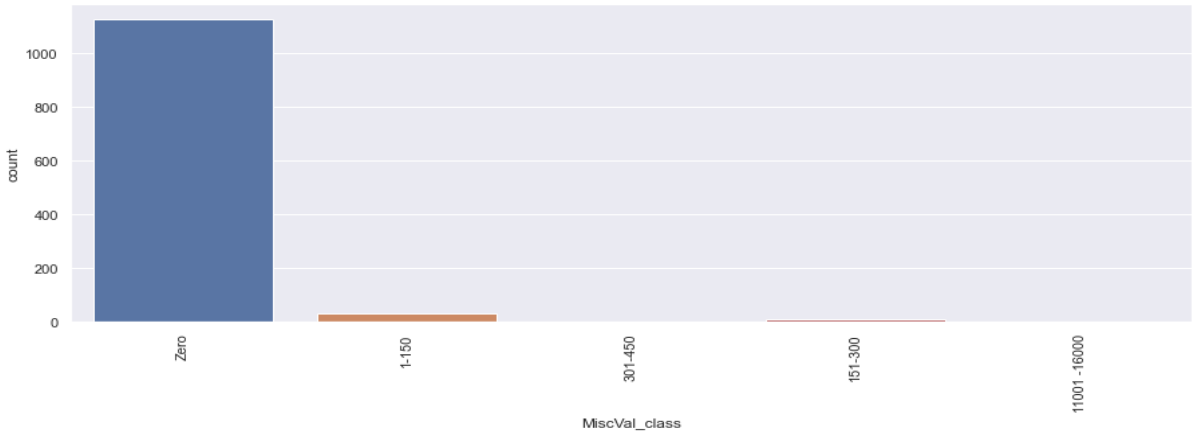

count
1200
1000
800
600
400
200
0
Zero
451-600
1-150
151-300
301-450
LowQualFinSF_class

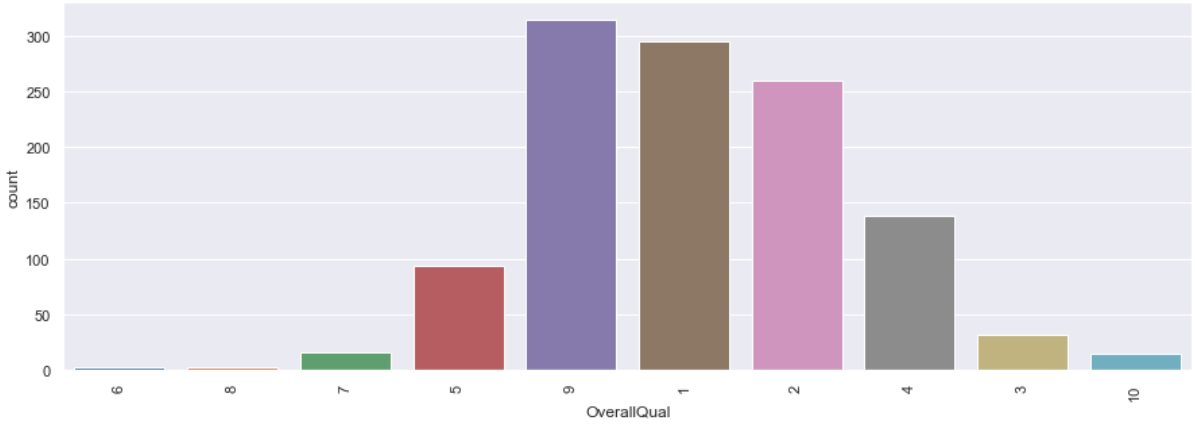

count
1000
800
600
400
200
0
Zero
151-300
1-150
301-450
451-600
EnclosedPorch_class

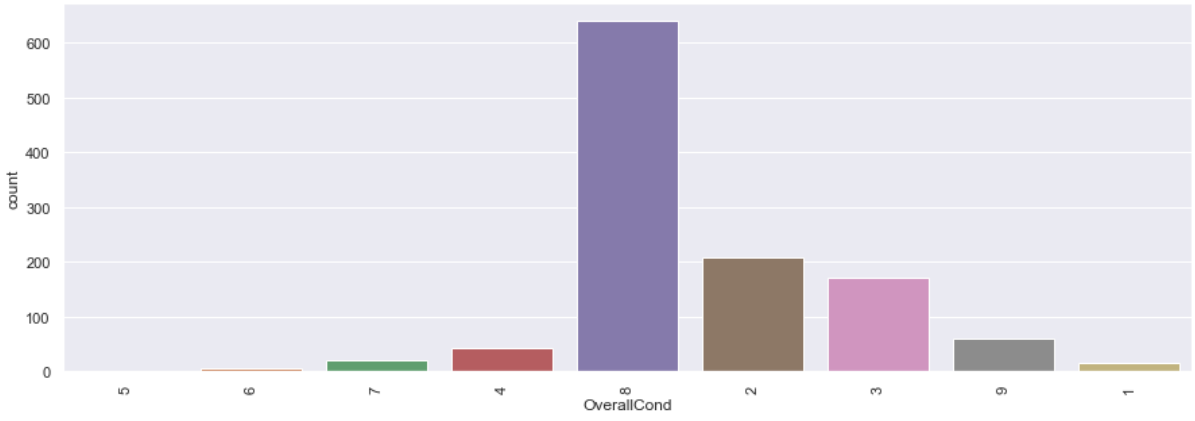

count
1200
1000
800
600
400
200
0
Zero
151-300
1-150
301-450
451-600
3SsnPorch_class

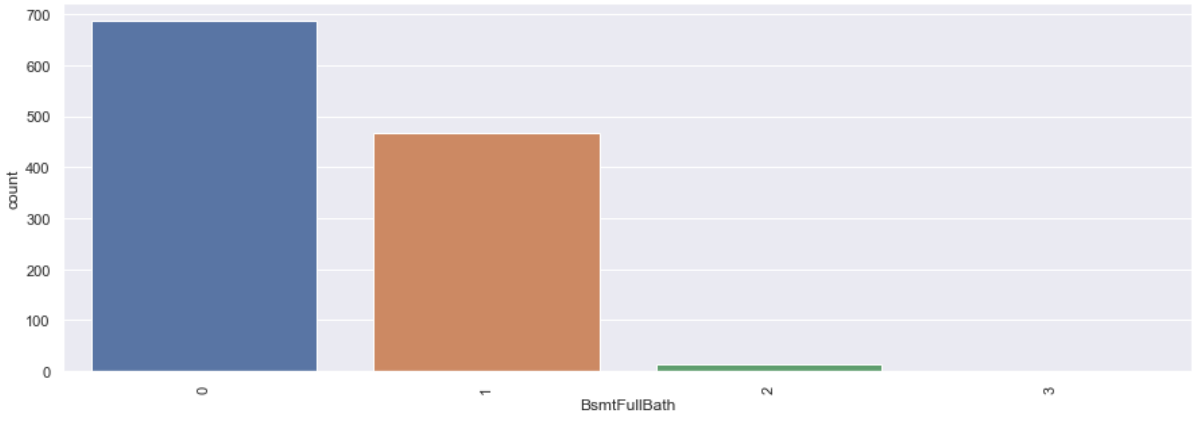

count
1000
800
600
400
200
0
Zero
151-300
1-150
450+
301-450
ScreenPorch_class

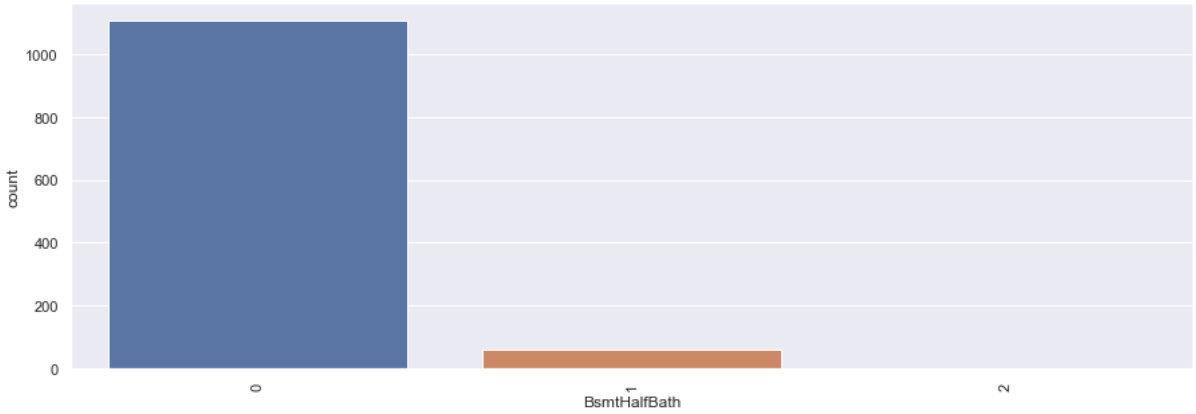

count
1200
1000
800
600
400
200
0
Zero
450-600
601-750
PoolArea_class

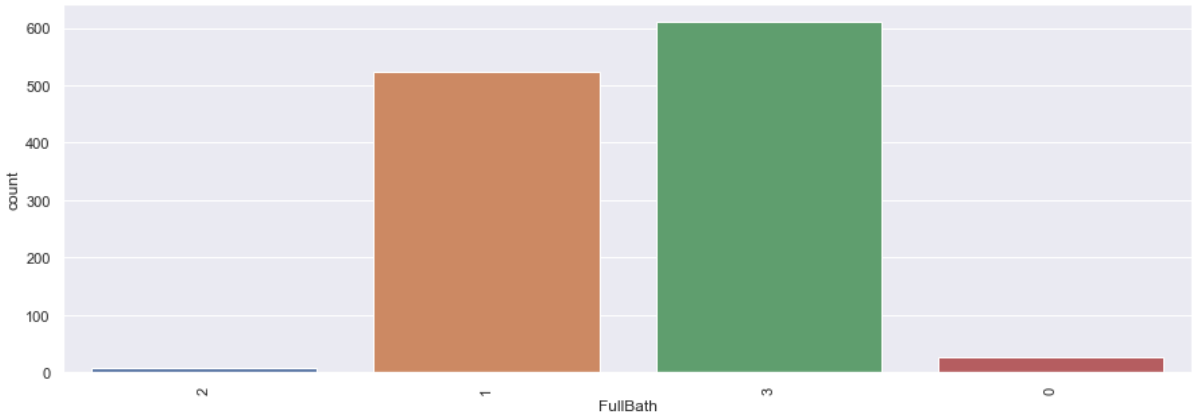

count
1000
800
600
400
200
0
Zero
1-150
301-450
151-300
11001 -16000
MiscVal_class

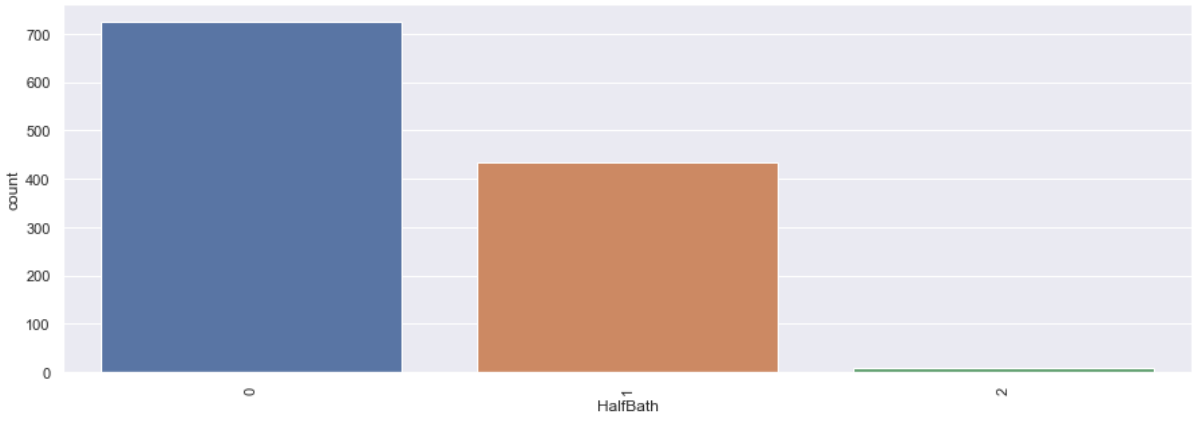

count
300
250
200
150
100
50
0
6
8
7
5
9
1
2
4
3
10
OverallQual

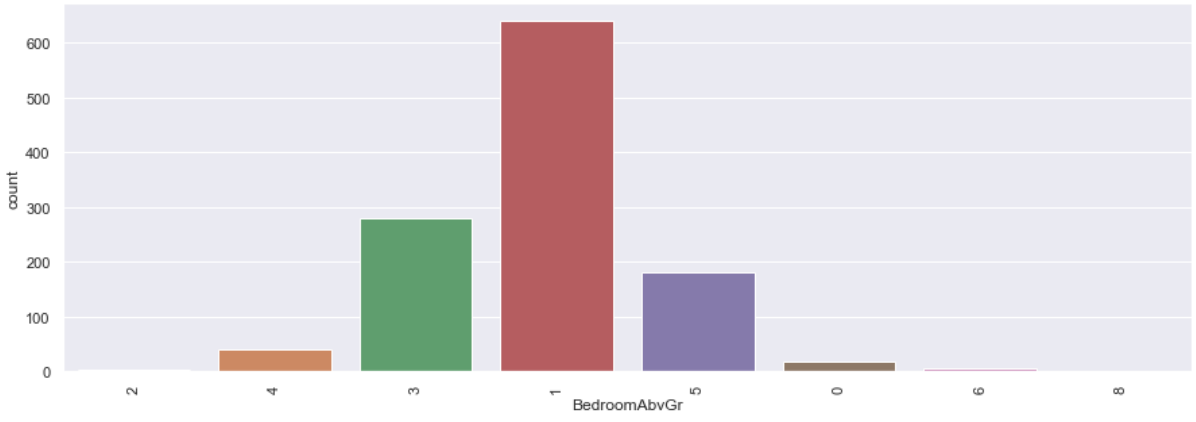

count
600
500
400
300
200
100
0
5
6
7
4
8
2
3
9
1
OverallCond

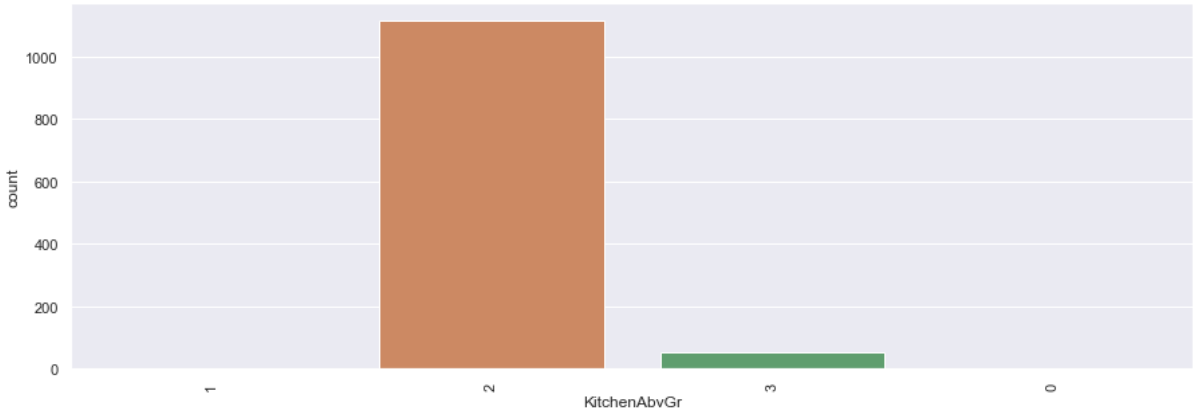

count
700
600
500
400
300
200
100
0
0
1
2
3
BsmtFullBath

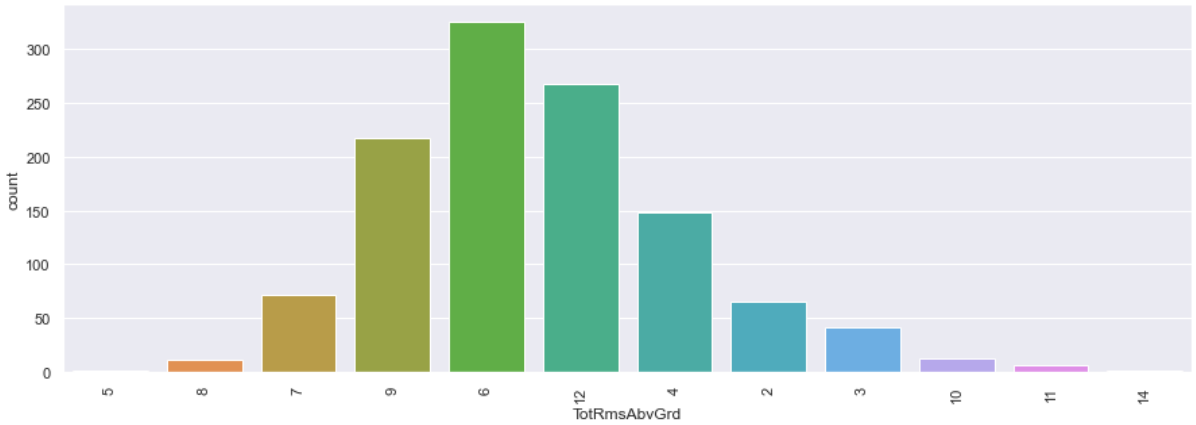

count
1000
800
600
400
200
0
0
1
2
BsmtHalfBath

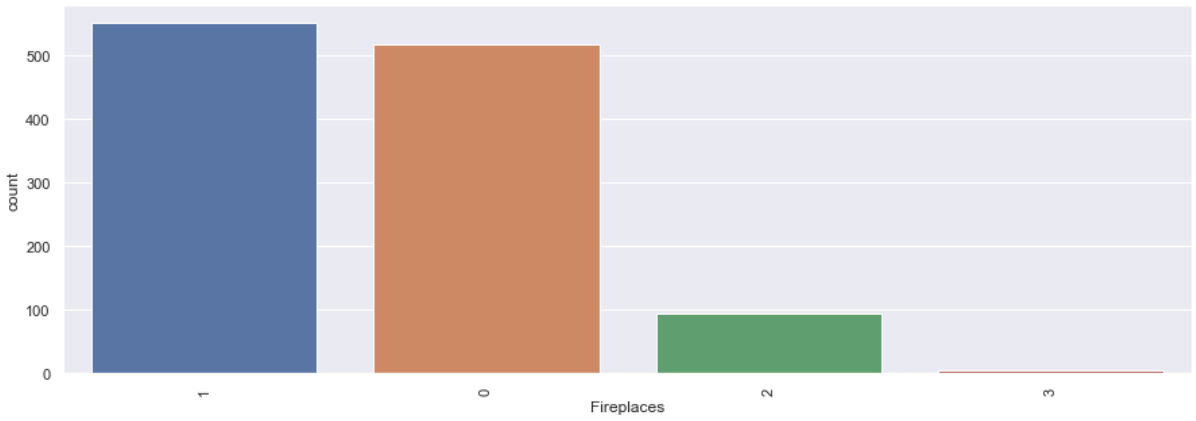

count
600
500
400
300
200
100
0
2
1
3
0
FullBath

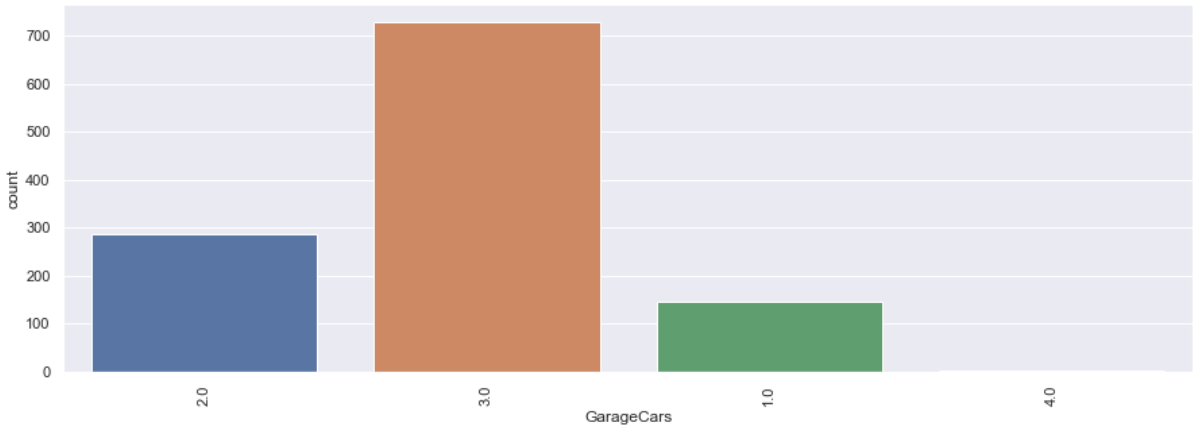

count
700
600
500
400
300
200
100
0
0
1
2
HalfBath


count
600
500
400
300
200
100
0
2
4
3
1
5
0
6
8
BedroomAbvGr

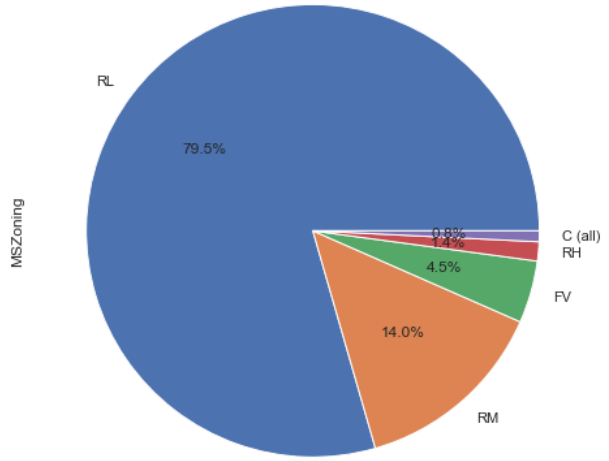

count
1000
800
600
400
200
0
1
2
3
0
KitchenAbvGr

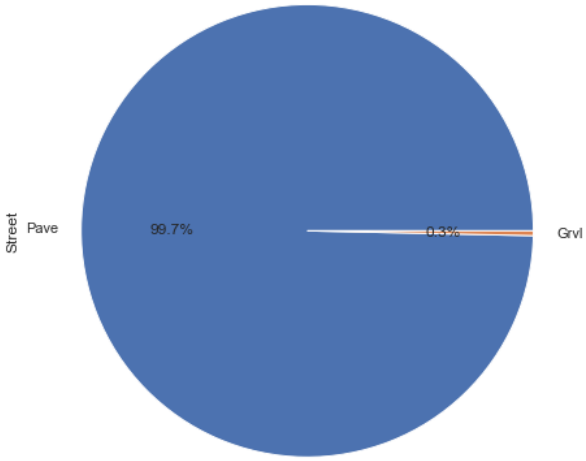

count
300
250
200
150
100
50
0
5
8
7
9
6
12
4
2
3
10
11
14
TotRmsAbvGrd

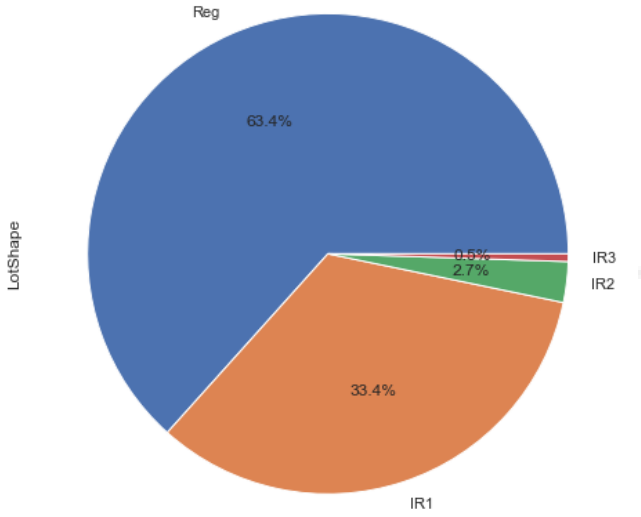

count
500
400
300
200
100
0
1
0
2
3
Fireplaces

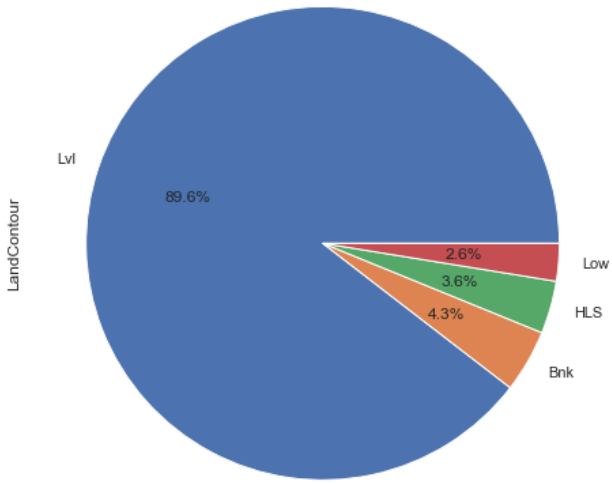

count
700
600
500
400
300
200
100
0
2.0
3.0
1.0
4.0
GarageCars

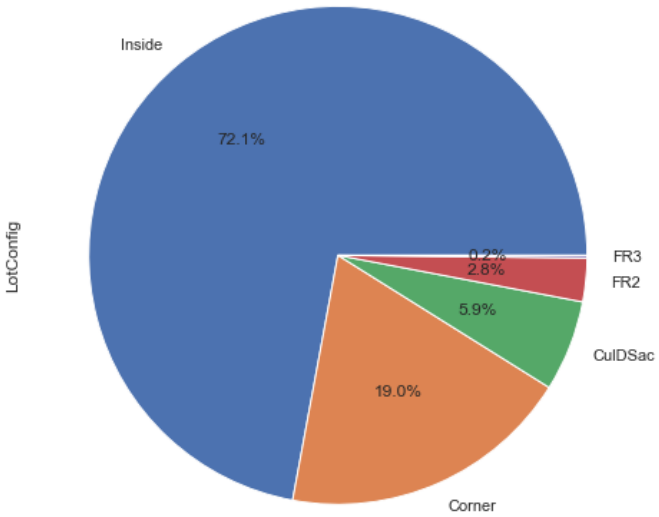
- **Pie Chart:**

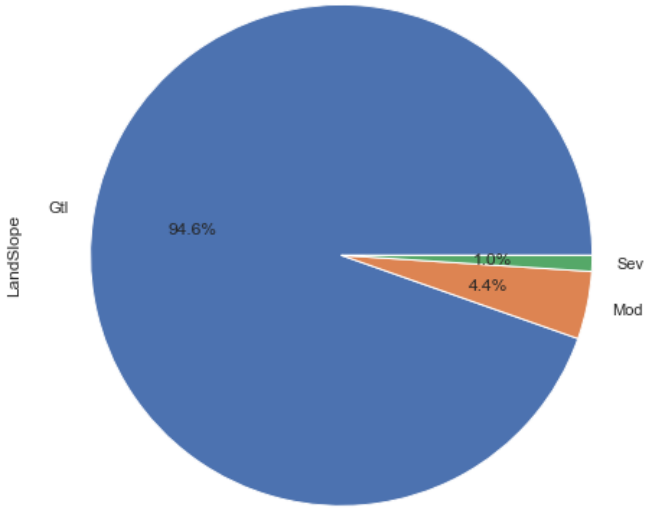

MSZoning
RL
79.5%
C (all)
0.8%
RH
1.4%
FV
4.5%
RM
14.0%

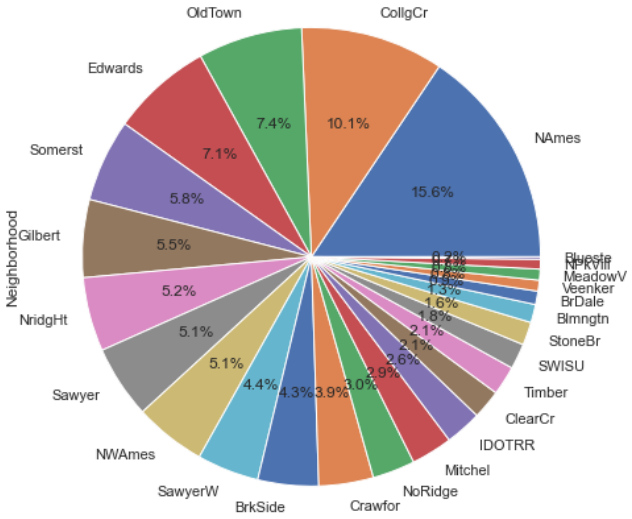

Street
Pave
99.7%
Grvl
0.3%

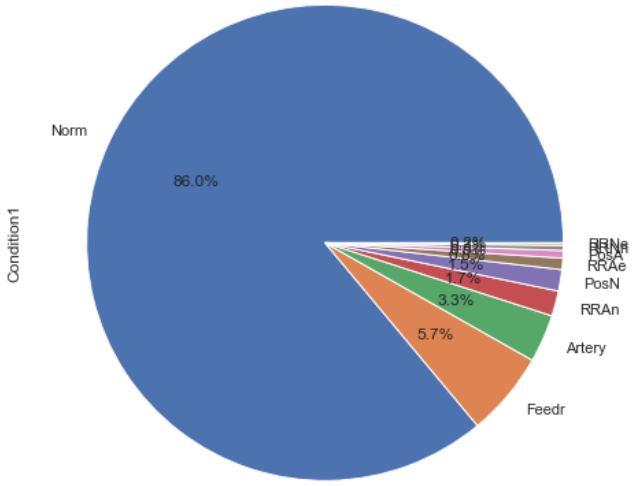

LotShape
Reg
63.4%
IR3
0.6%
IR2
2.7%
IR1
33.4%

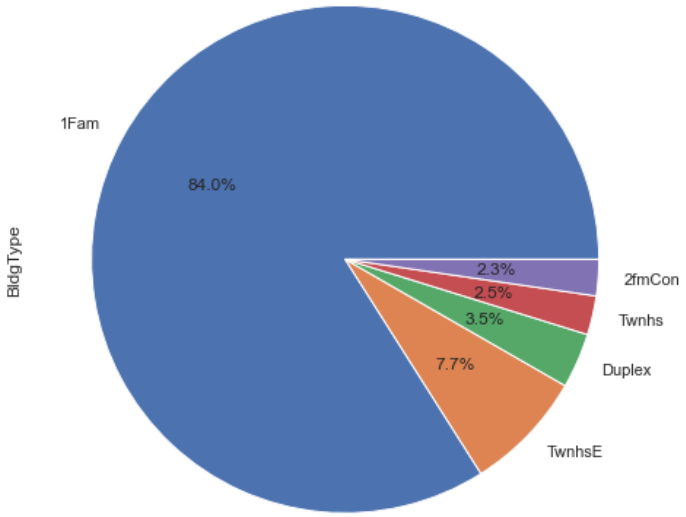

LandContour
Lvl
89.6%
Low
2.6%
HLS
3.6%
Bnk
4.3%

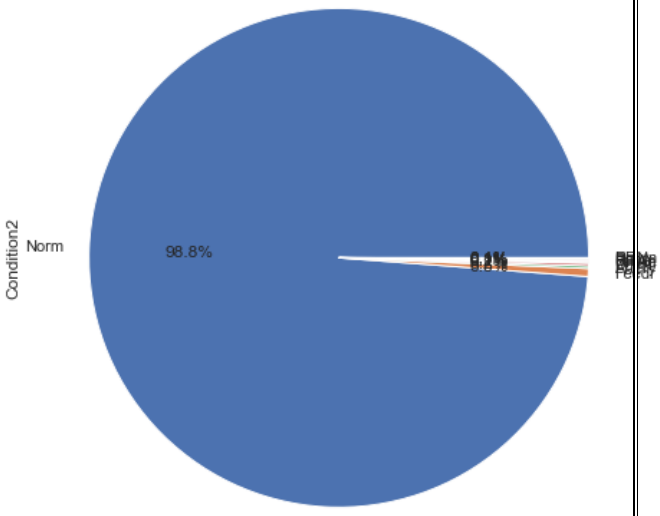

LotConfig
Inside
72.1%
FR3
0.2%
FR2
2.8%
CulDSac
5.9%
Corner
19.0%

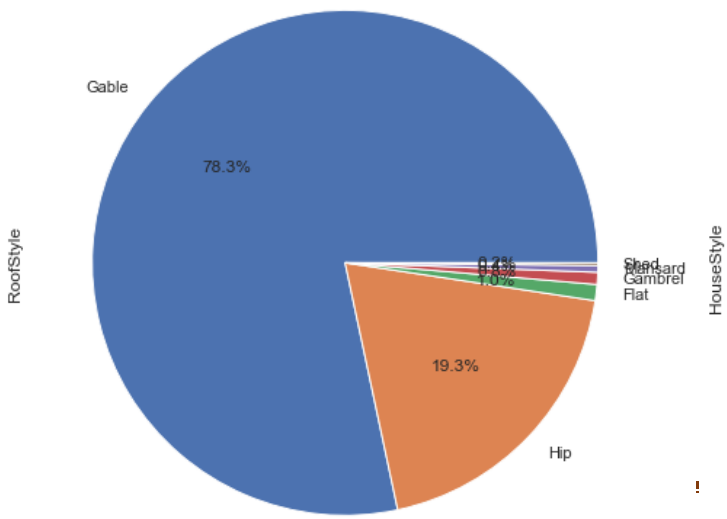

LandSlope
Gtl
94.6%
Sev
1.0%
Mod
4.4%

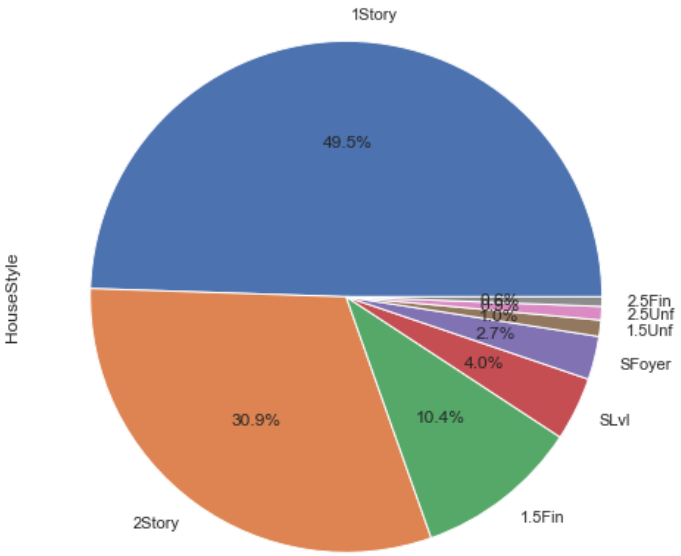

Neighborhood
OldTown
CollgCr
Edwards
Somerst
Gilbert
NridgHt
Sawyer
NWAmes
SawyerW
BrkSide
Crawfor
NoRidge
Mitchel
IDOTRR
ClearCr
Timber
SWISU
StoneBr
Blmngtn
BrDale
Veenker
MeadowV
NPkVill
Blueste
NAmes
15.6%
10.1%
7.4%
7.1%
5.8%
5.5%
5.2%
5.1%
5.1%
4.4%
4.3%
3.9%
3.0%
2.9%
2.6%
2.1%
2.1%
1.8%
1.6%

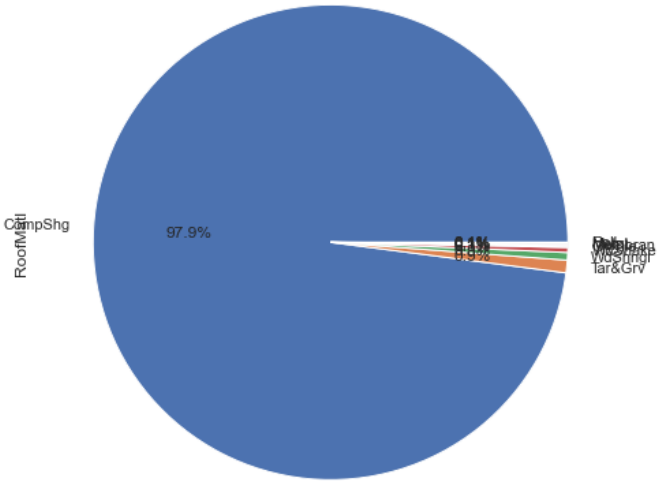

Condition1
Norm
86.0%
Feedr
5.7%
Artery
3.3%
RRAn
1.7%
PosN
1.5%
RRAe
PosA

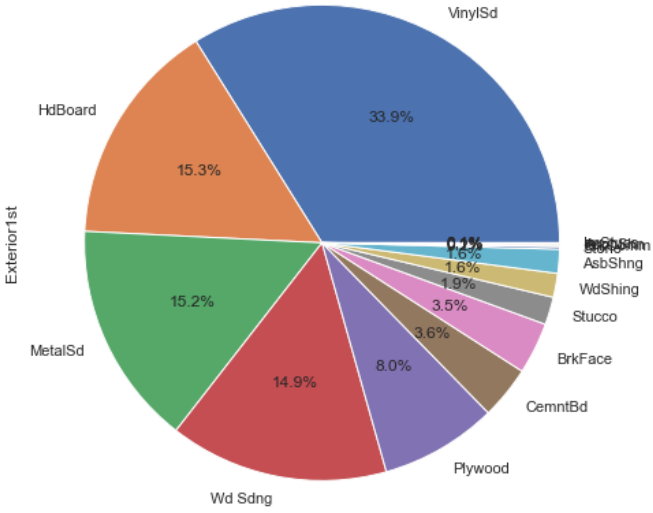

BldgType
1Fam
84.0%
2fmCon
2.3%
Twnhs
2.5%
Duplex
3.5%
TwnhsE
7.7%

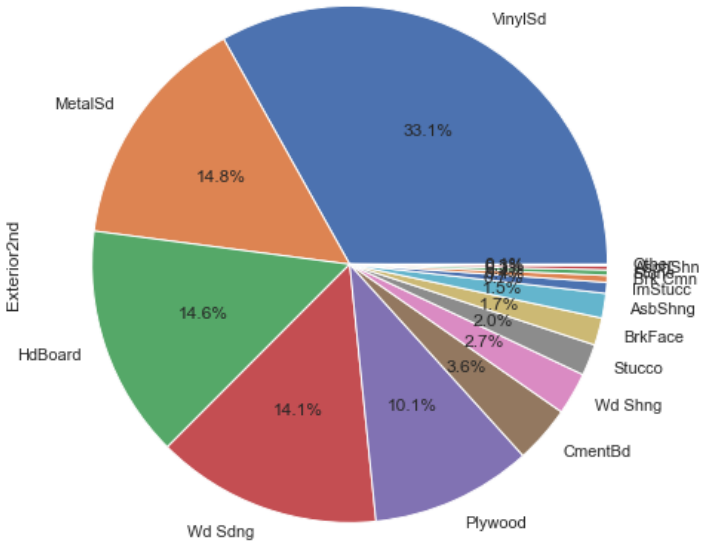

Condition2
Norm
98.8%

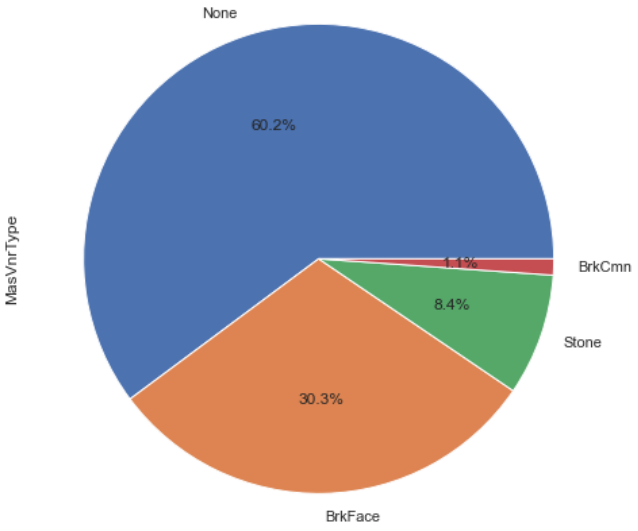

RoofStyle
Gable
78.3%
Hip
19.3%
Flat
1.0%
Gambrel
Mansard
Shed

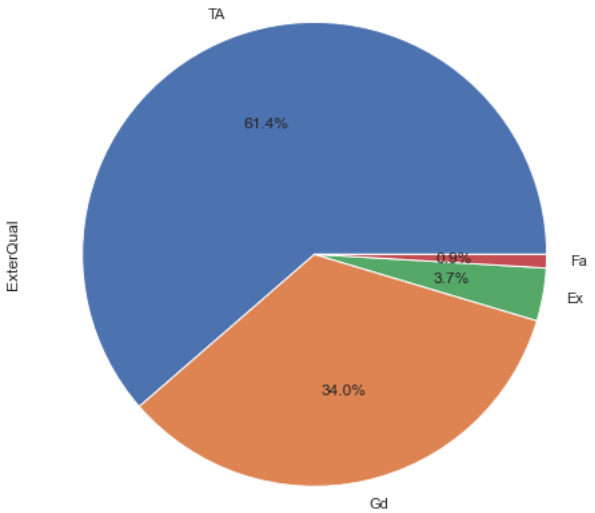

HouseStyle
1Story
49.5%
2Story
30.9%
1.5Fin
10.4%
SLvl
4.0%
SFoyer
2.7%
1.5Unf
2.5Unf
2.5Fin

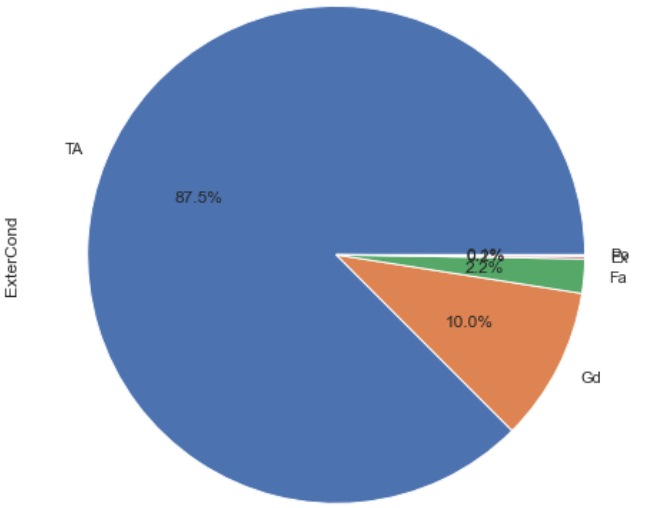

CompShg
97.9%

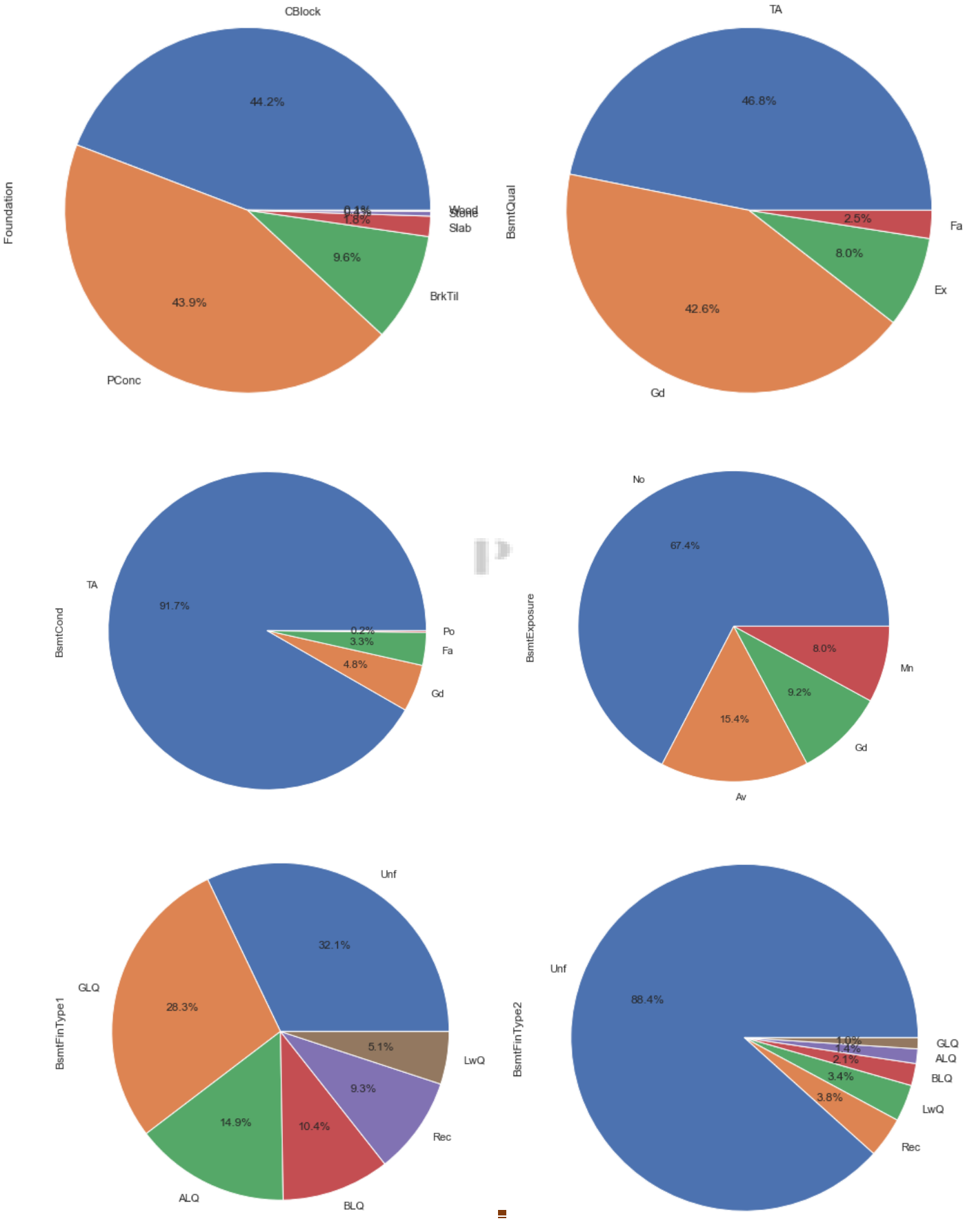

VinylSd
33.9%
HdBoard
15.3%
MetalSd
15.2%
Wd Sdng
14.9%
Plywood
8.0%
CemntBd
3.6%
BrkFace
3.5%
Stucco
1.9%
WdShing
1.6%
AsbShng
1.6%

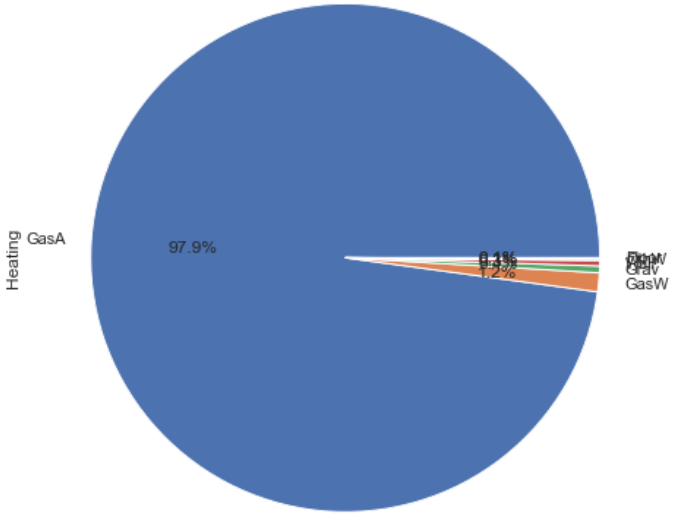

VinylSd
33.1%
MetalSd
14.8%
HdBoard
14.6%
Wd Sdng
14.1%
Plywood
10.1%
CmentBd
3.6%
Wd Shng
2.7%
Stucco
2.0%
BrkFace
1.7%
AsbShng
1.5%

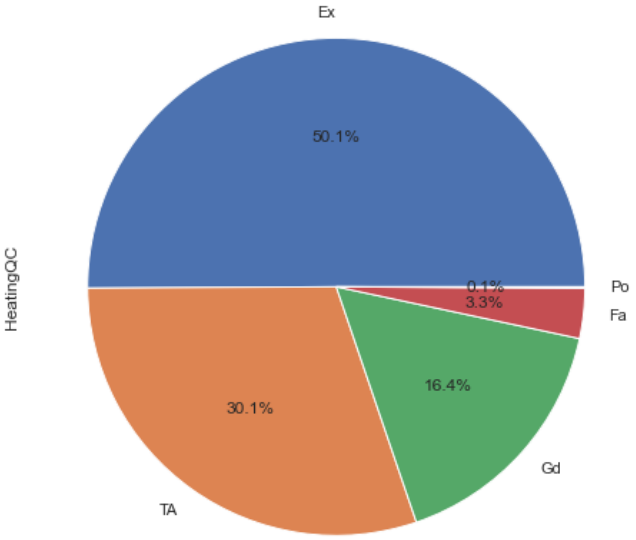

None
60.2%
BrkFace
30.3%
Stone
8.4%
BrkCmn
1.1%

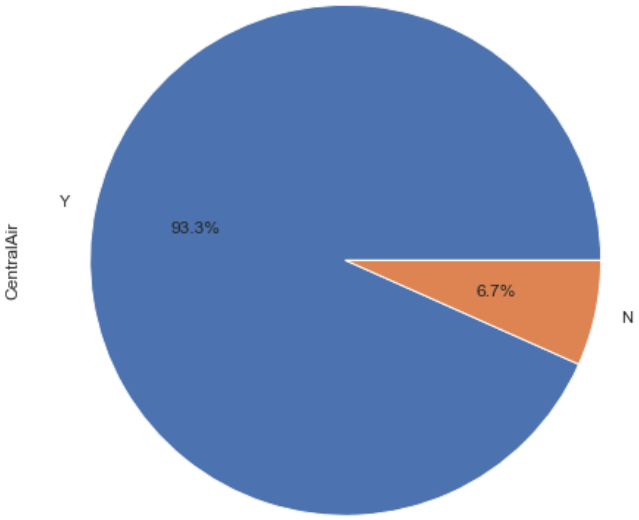

TA
61.4%
Gd
34.0%
Ex
3.7%
Fa
0.9%

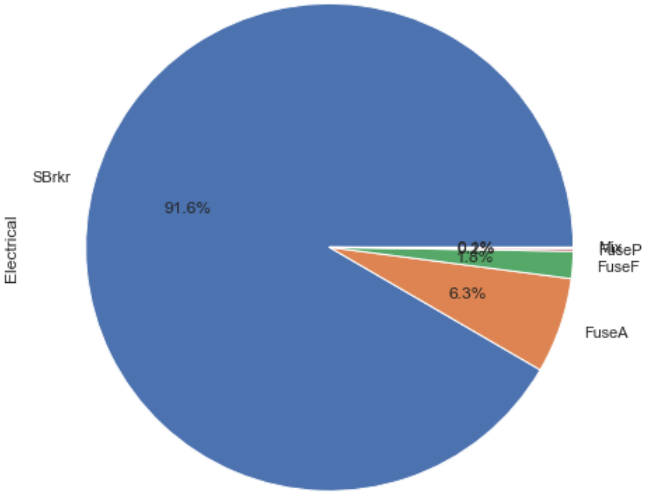

TA
87.5%
Gd
10.0%
Fa
2.2%

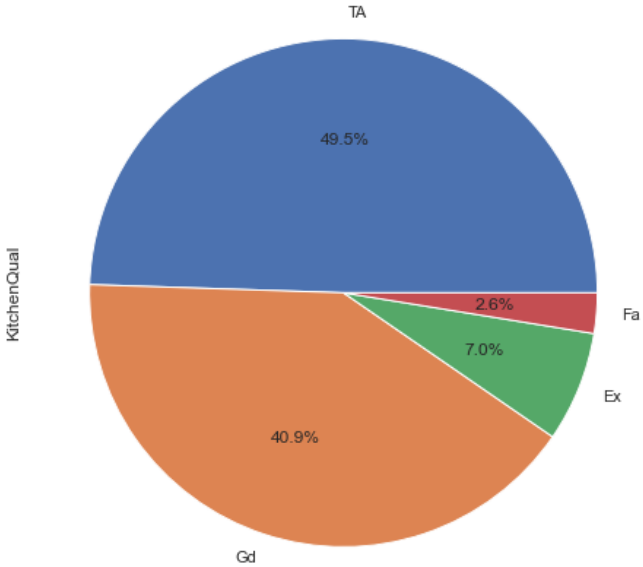

Foundation
CBlock
44.2%
PConc
43.9%
BrkTil
9.6%
Slab
1.8%
Stone
Wood
BsmtQual
TA
46.8%
Gd
42.6%
Ex
8.0%
Fa
2.5%
BsmtCond
TA
91.7%
Po
Fa
3.3%
Gd
4.8%
BsmtExposure
No
67.4%
Av
15.4%
Gd
9.2%
Mn
8.0%
BsmtFinType1
Unf
32.1%
GLQ
28.3%
ALQ
14.9%
BLQ
10.4%
Rec
9.3%
LwQ
5.1%
BsmtFinType2
Unf
88.4%
Rec
3.8%
LwQ
3.4%
BLQ
2.1%
ALQ
1.4%
GLQ
1.0%

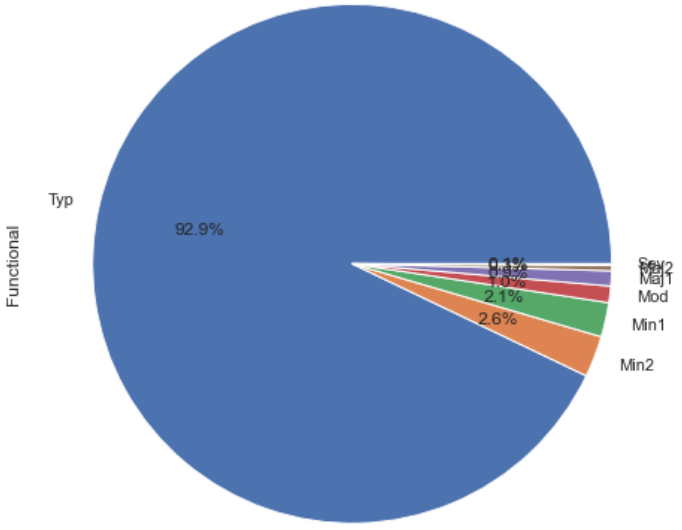

Heating
GasA
97.9%
GasW

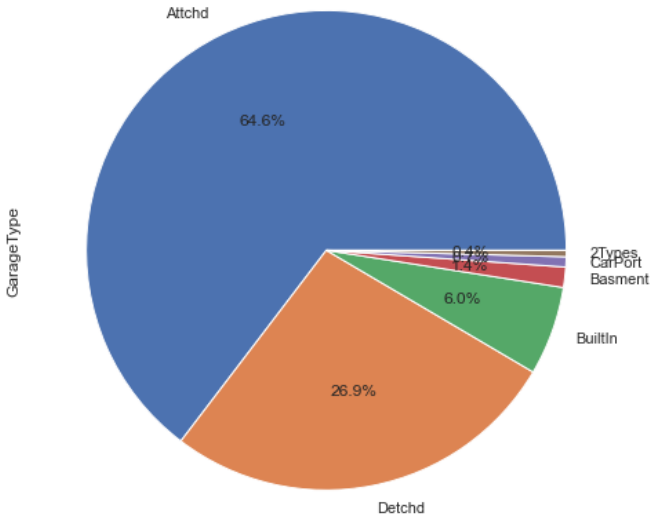

HeatingQC
Ex
50.1%
Po
0.1%
Fa
3.3%
16.4%
Gd
30.1%
TA

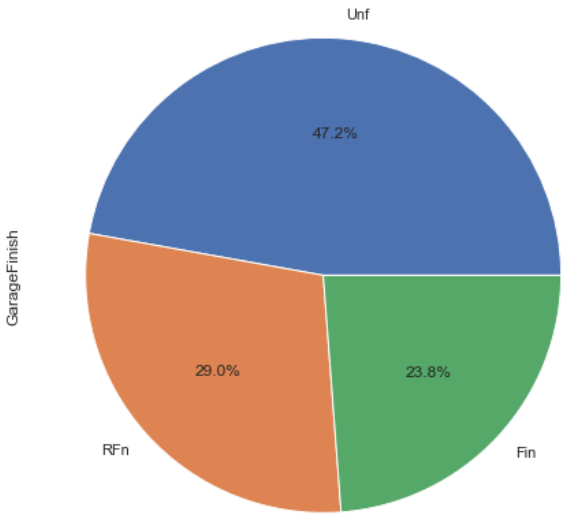

CentralAir
Y
93.3%
6.7%
N

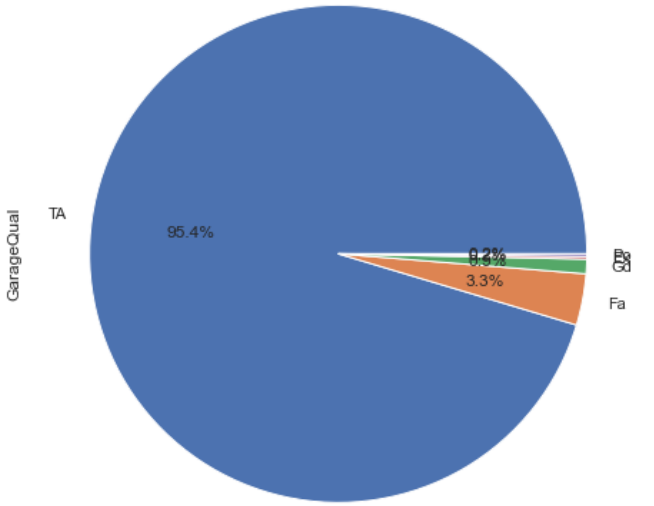

Electrical
SBrkr
91.6%
FuseP
FuseF
6.3%
FuseA

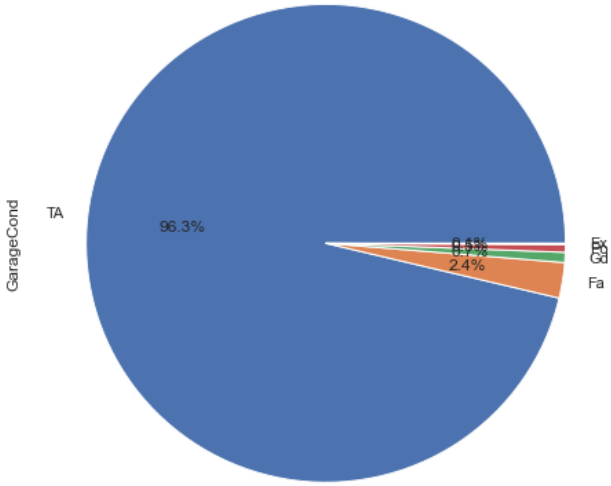

KitchenQual
TA
49.5%
2.6%
Fa
7.0%
Ex
40.9%
Gd

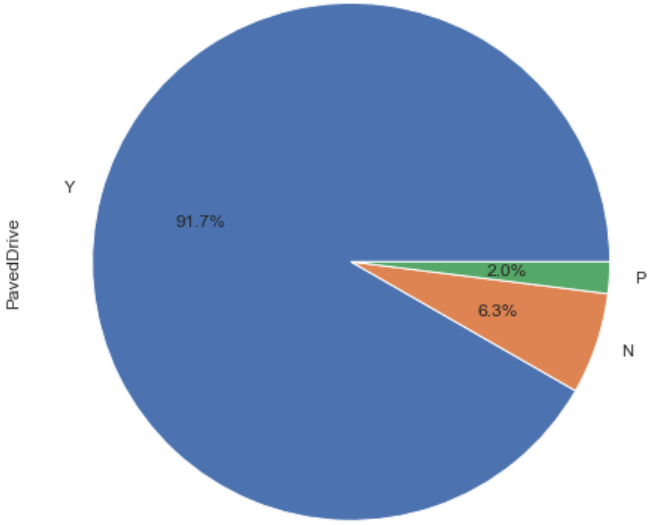

Functional
Typ
92.9%
Mod
2.1%
Min1
2.6%
Min2

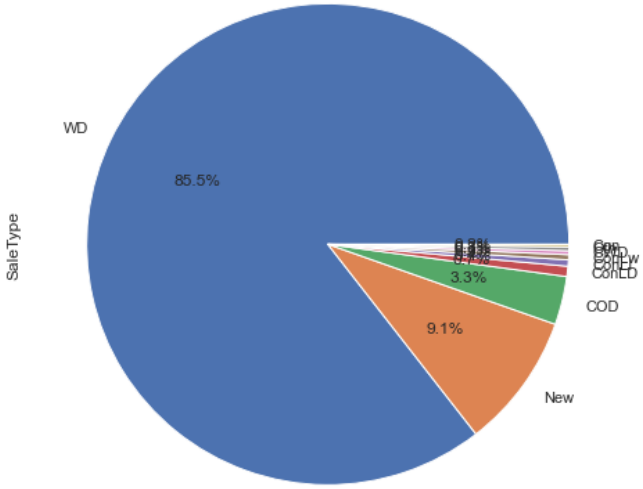

GarageType
Attchd
64.6%
2Types
CarPort
Basment
6.0%
BuiltIn
26.9%
Detchd

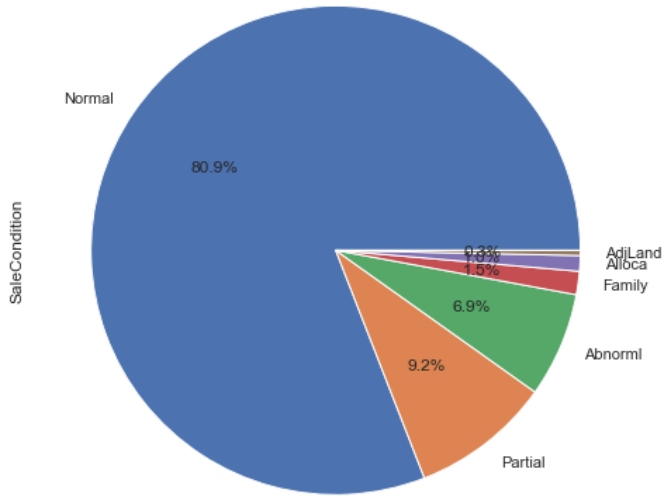

GarageFinish
Unf
47.2%
29.0%
RFn
23.8%
Fin

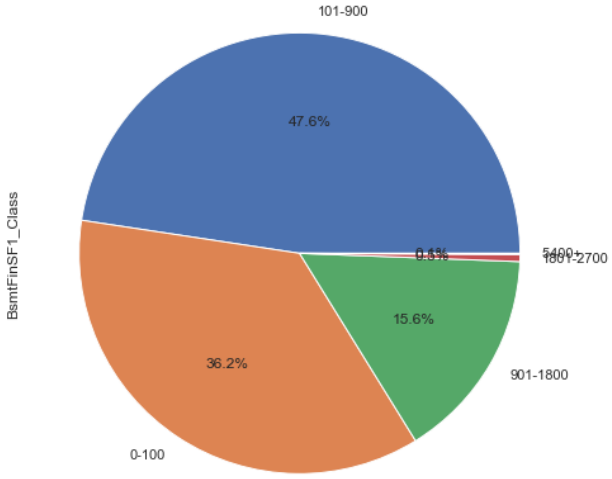

GarageQual
TA
95.4%
3.3%
Fa

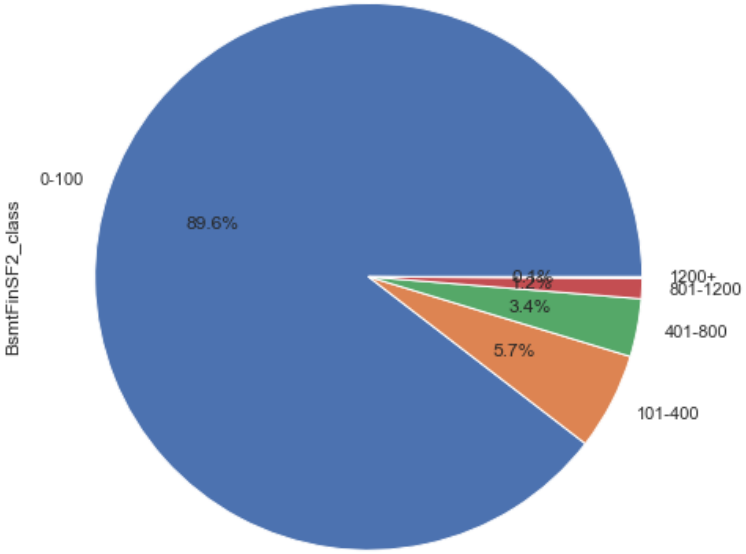

GarageCond
TA
96.3%
2.4%
Fa

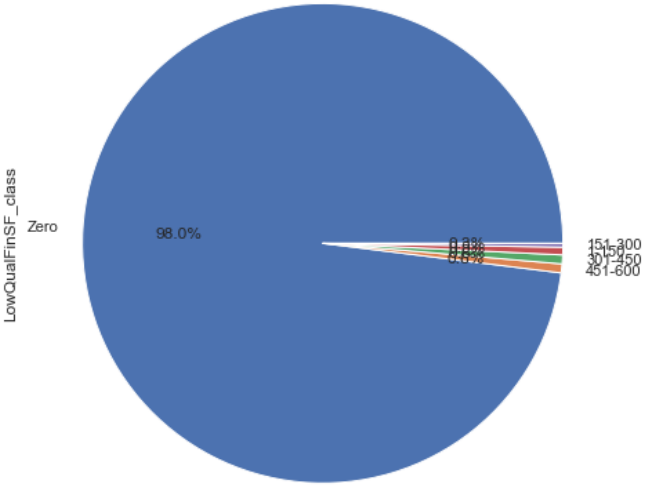

PavedDrive
Y
91.7%
2.0%
P
6.3%
N

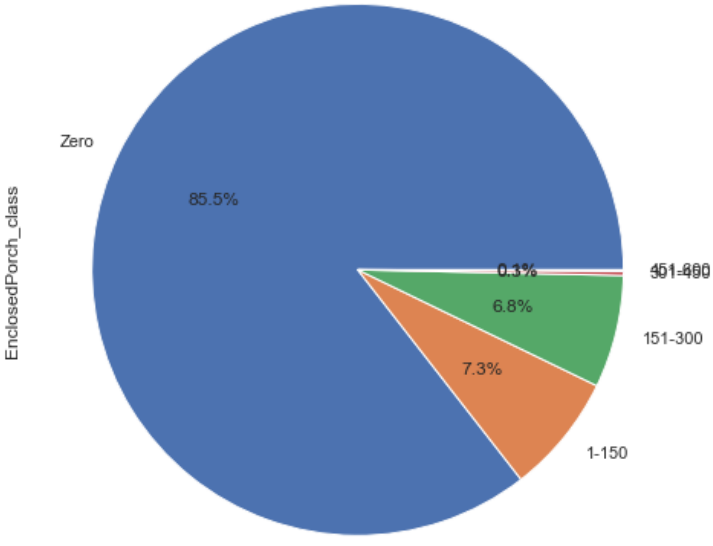

SaleType
WD
85.5%
3.3%
COD
9.1%
New

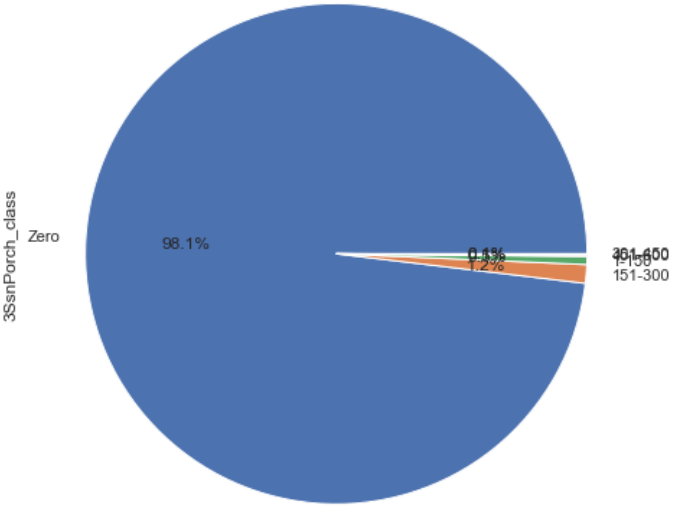

SaleCondition
Normal
80.9%
AdjLand
Alloca
Family
Abnorml
Partial
6.9%
9.2%

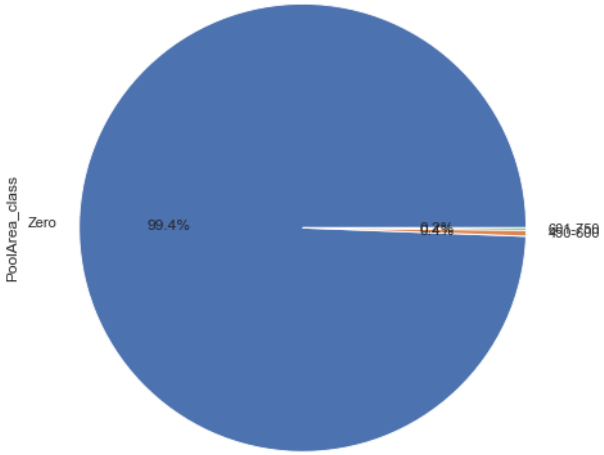

BsmtFinSF1_Class
101-900
47.6%
5400+
1801-2700
901-1800
15.6%
36.2%
0-100

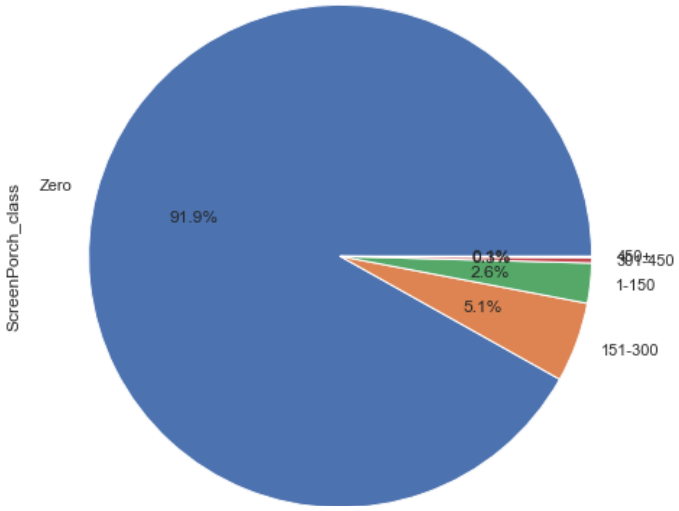

BsmtFinSF2_class
0-100
89.6%
1200+
801-1200
401-800
3.4%
5.7%
101-400

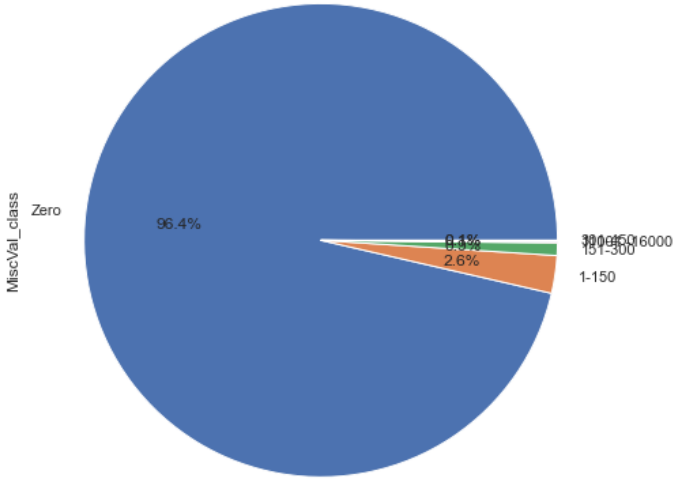

LowQualFinSF_class
Zero
98.0%
151-300
451-600

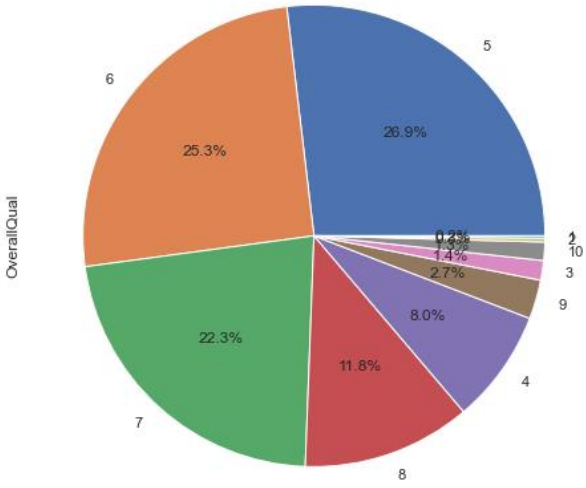

EnclosedPorch_class
Zero
85.5%
0.3%
151-300
6.8%
7.3%
1-150

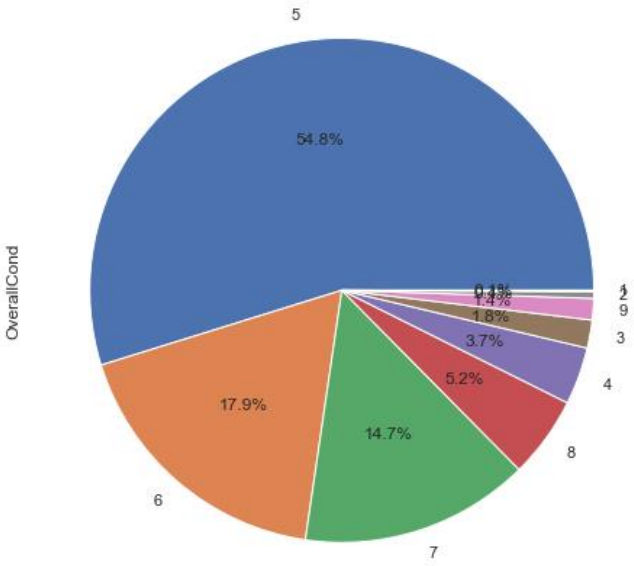

3SsnPorch_class
Zero
98.1%
151-300
1-150

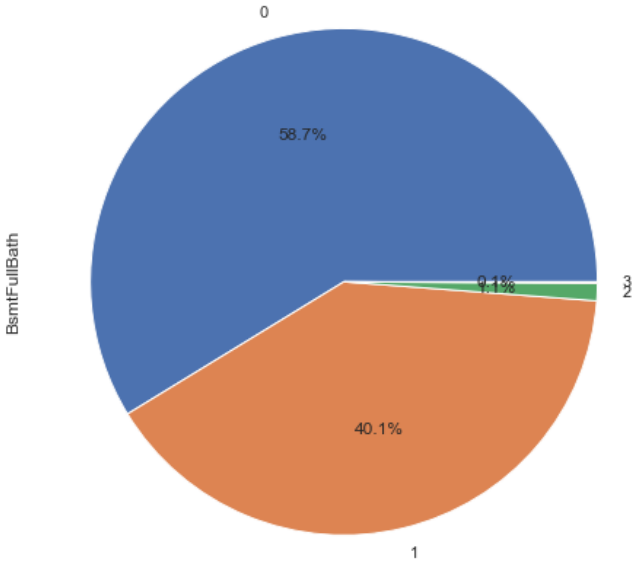

PoolArea_class
Zero
99.4%

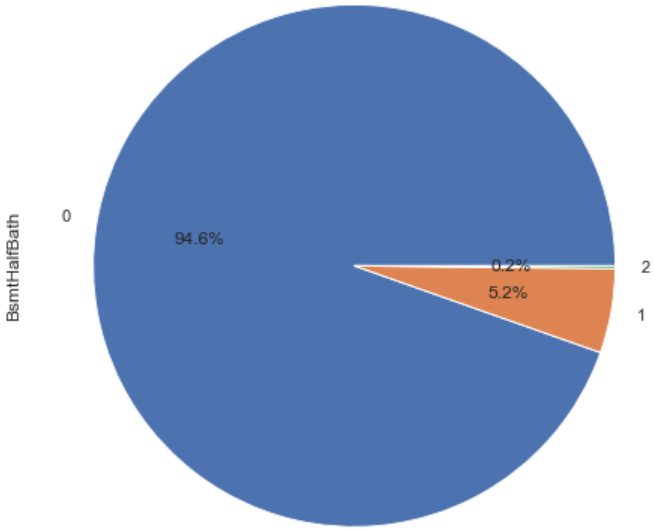

ScreenPorch_class
Zero
91.9%
2.6%
1-150
5.1%
151-300

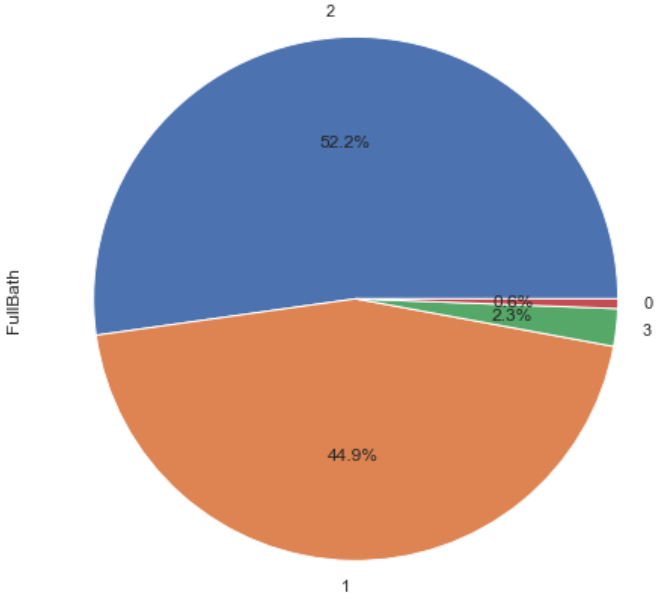

MiscVal_class
Zero
96.4%
2.6%
1-150

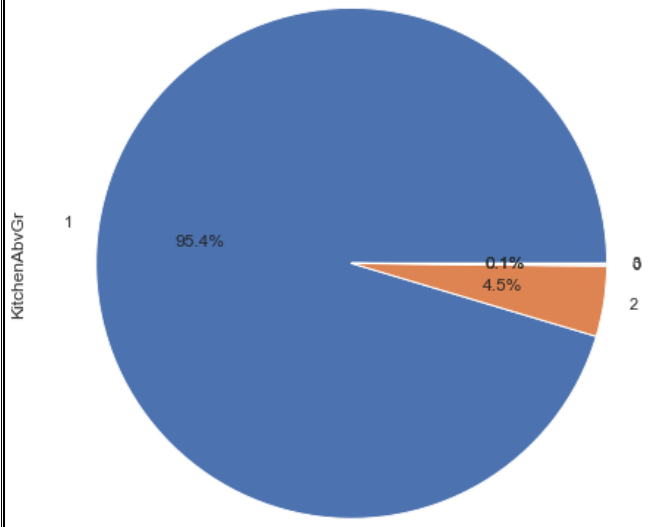

OverallQual
5
26.9%
6
25.3%
22.3%
11.8%
8.0%
2.7%
1.4%
7
8
4
3
9
10

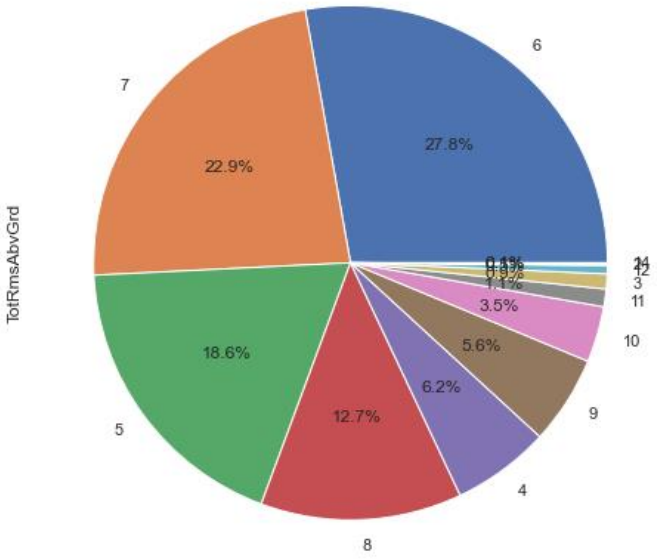

OverallCond
5
54.8%
17.9%
14.7%
6
7
5.2%
3.7%
1.8%
1.4%
8
4
3
9

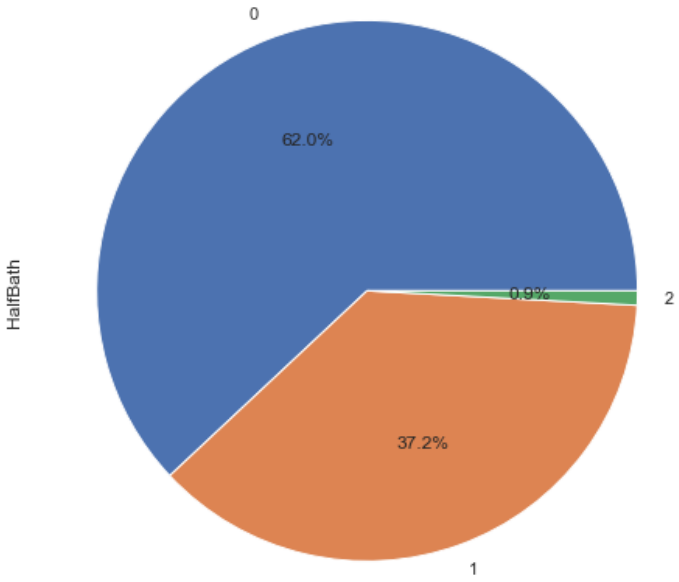

BsmtFullBath
0
58.7%
40.1%
1
1.1%
3
2

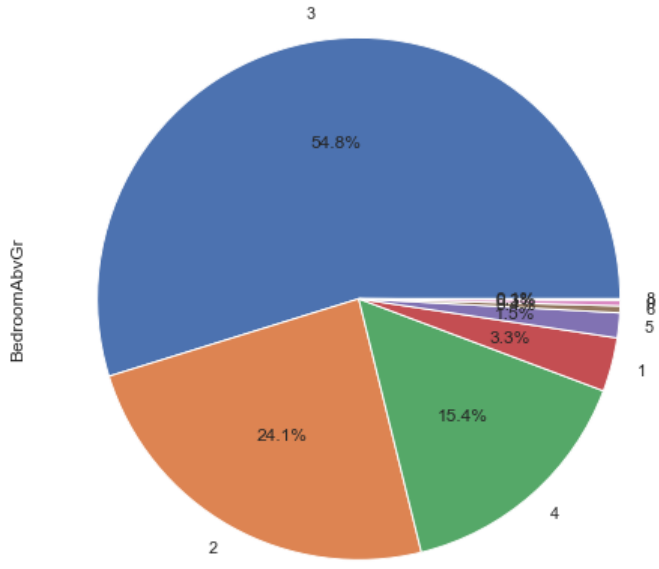

BsmtHalfBath
0
94.6%
0.2%
2
5.2%
1

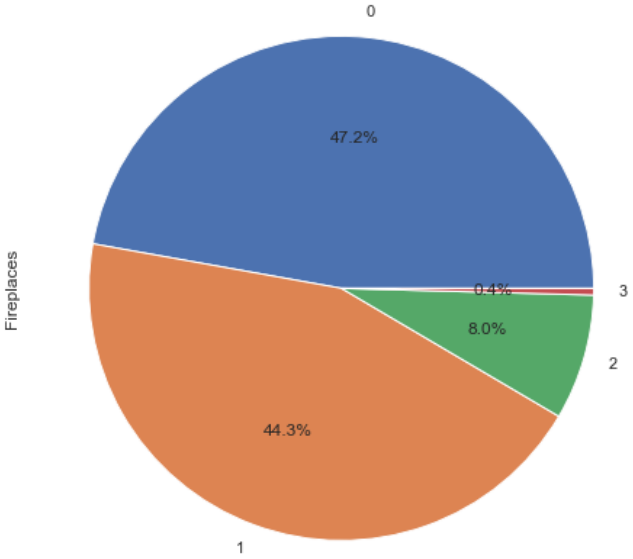

FullBath
2
52.2%
0.6%
0
2.3%
3
44.9%
1

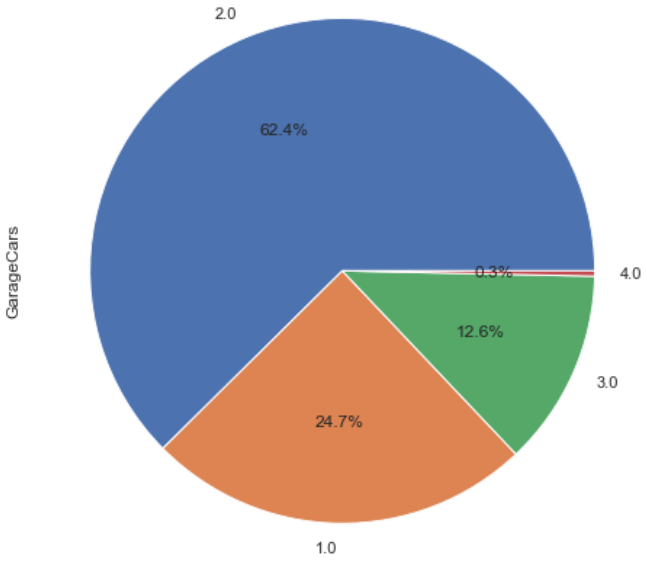

KitchenAbvGr
1
95.4%
0.1%
0
4.5%
2


TotRmsAbvGrd
6
7
27.8%
22.9%
3.5%
10
5.6%
18.6%
9
6.2%
5
12.7%
4
8

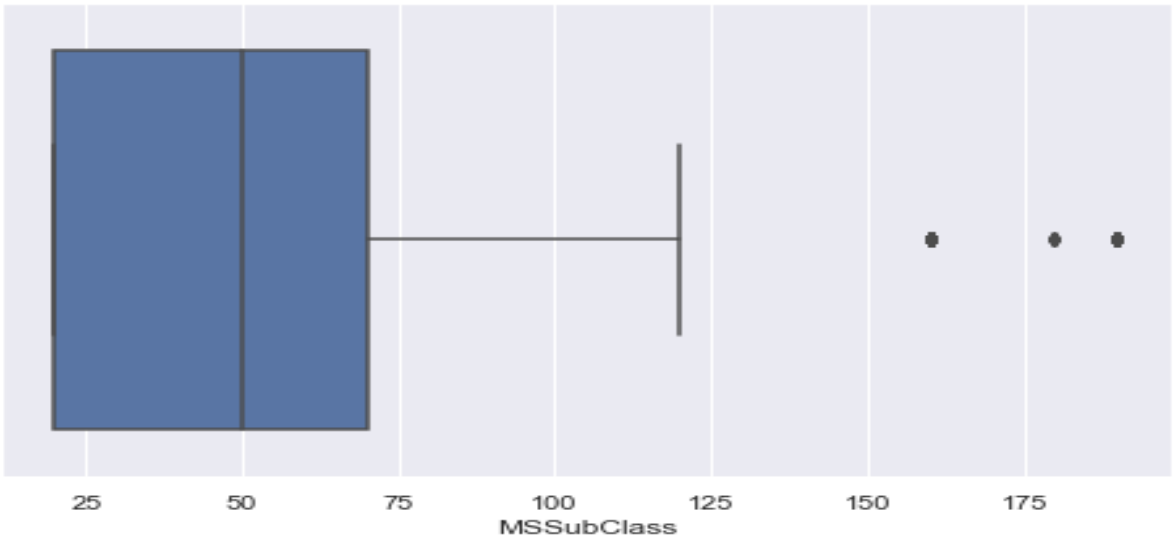

HalfBath
0
62.0%
0.9%
2
37.2%
1

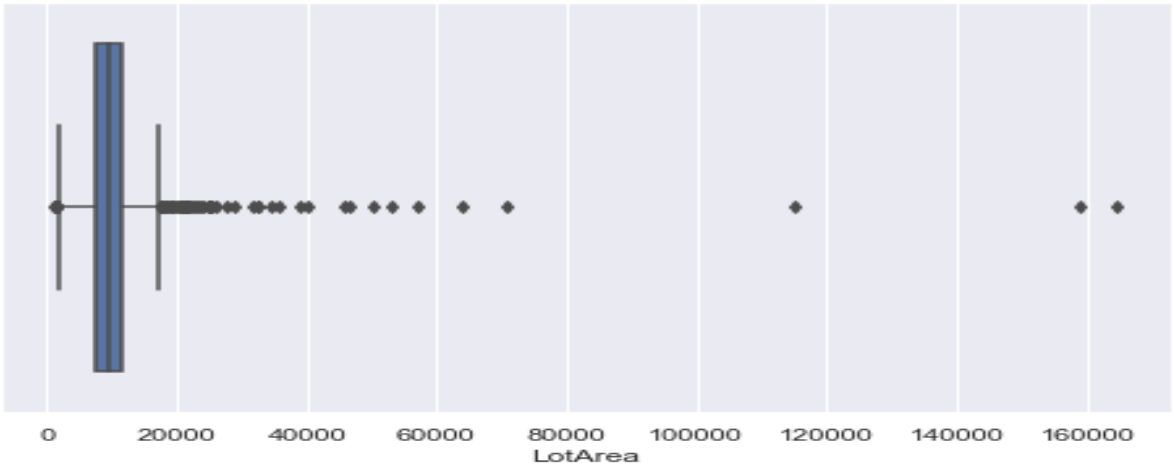

BedroomAbvGr
3
54.8%
1.5%
5
3.3%
1
15.4%
24.1%
4
2

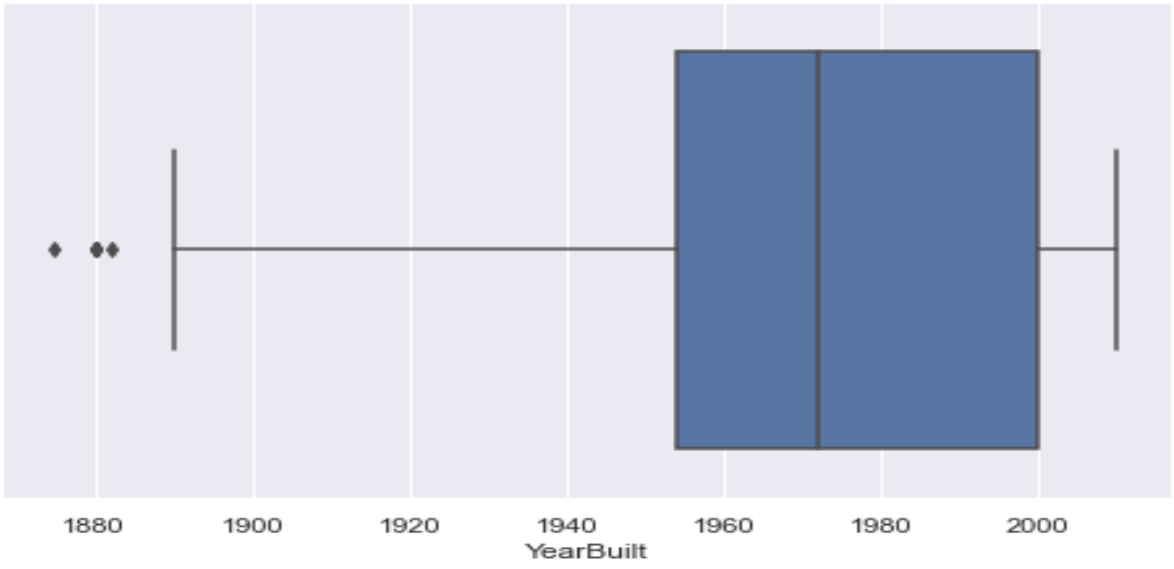

Fireplaces
0
47.2%
3
0.4%
8.0%
2
44.3%
1

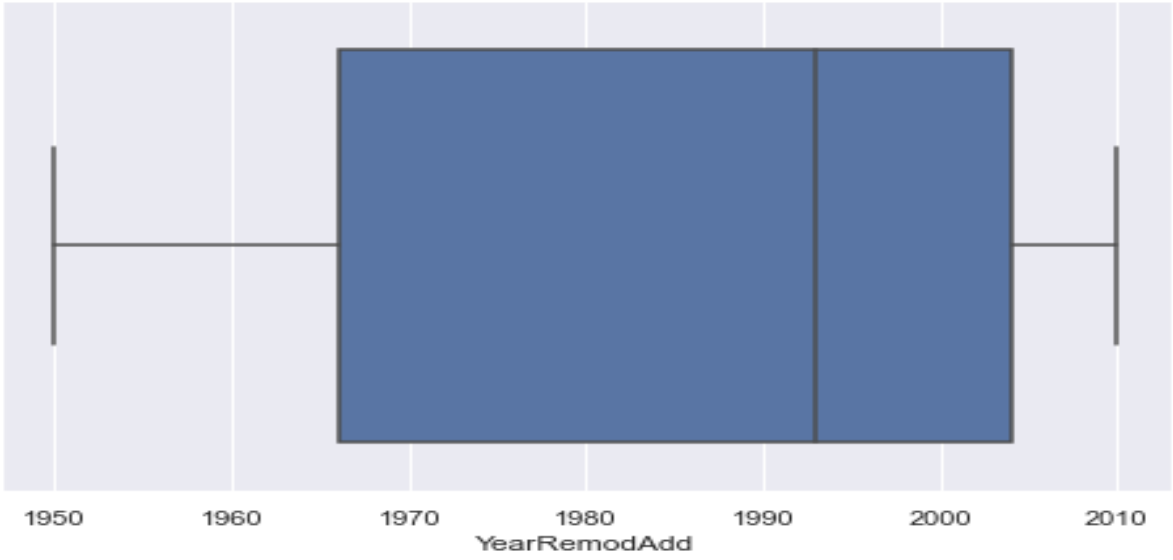

GarageCars
2.0
62.4%
4.0
0.3%
12.6%
3.0
24.7%
1.0

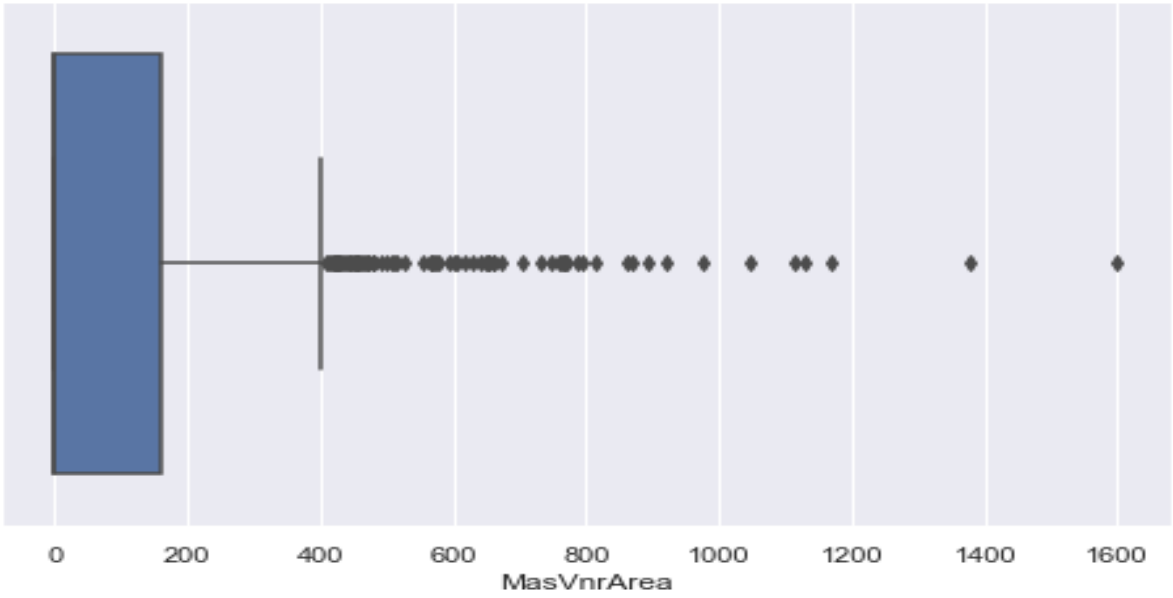
Outliers:

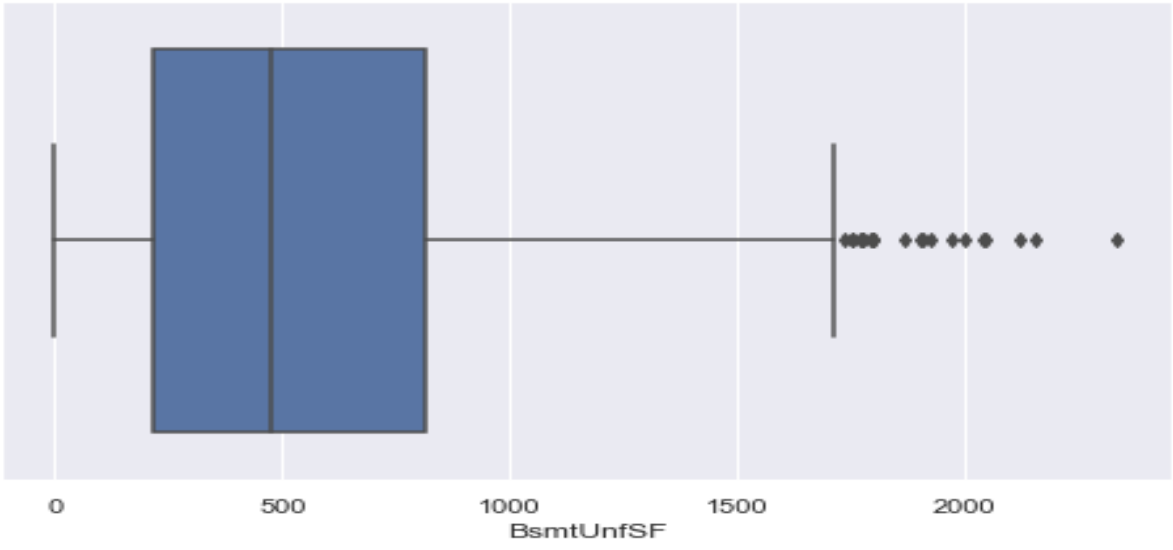

25
50
75
100
125
150
175
MSSubClass

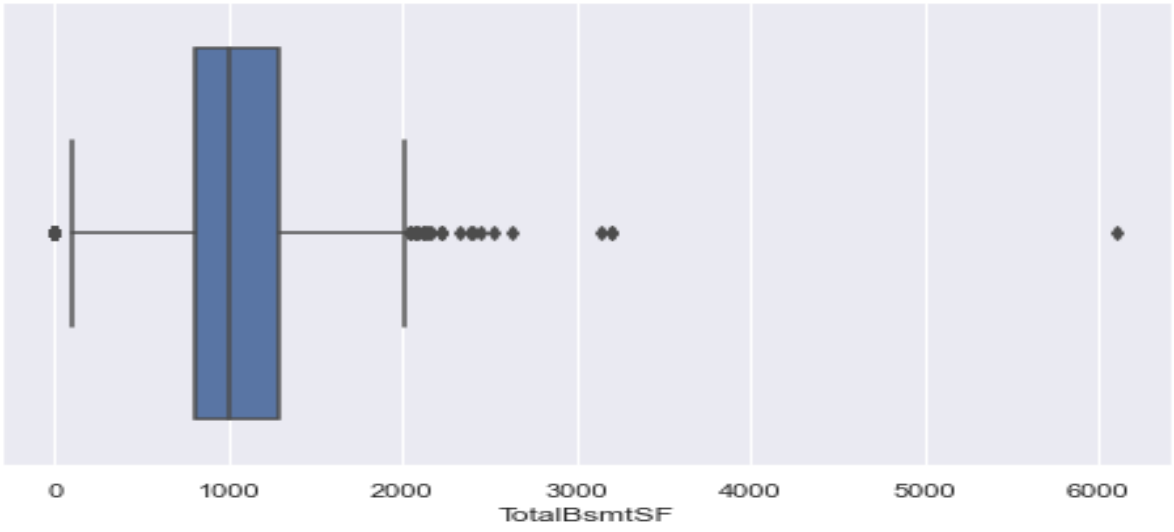

0
20000
40000
60000
80000
100000
120000
140000
160000
LotArea

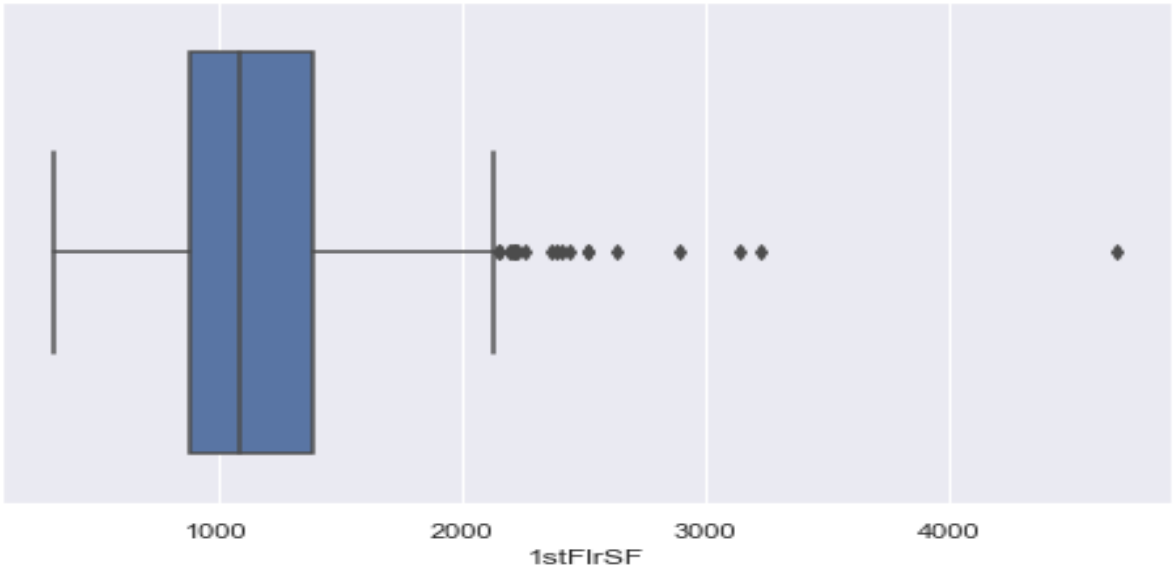

1880
1900
1920
1940
1960
1980
2000
YearBuilt

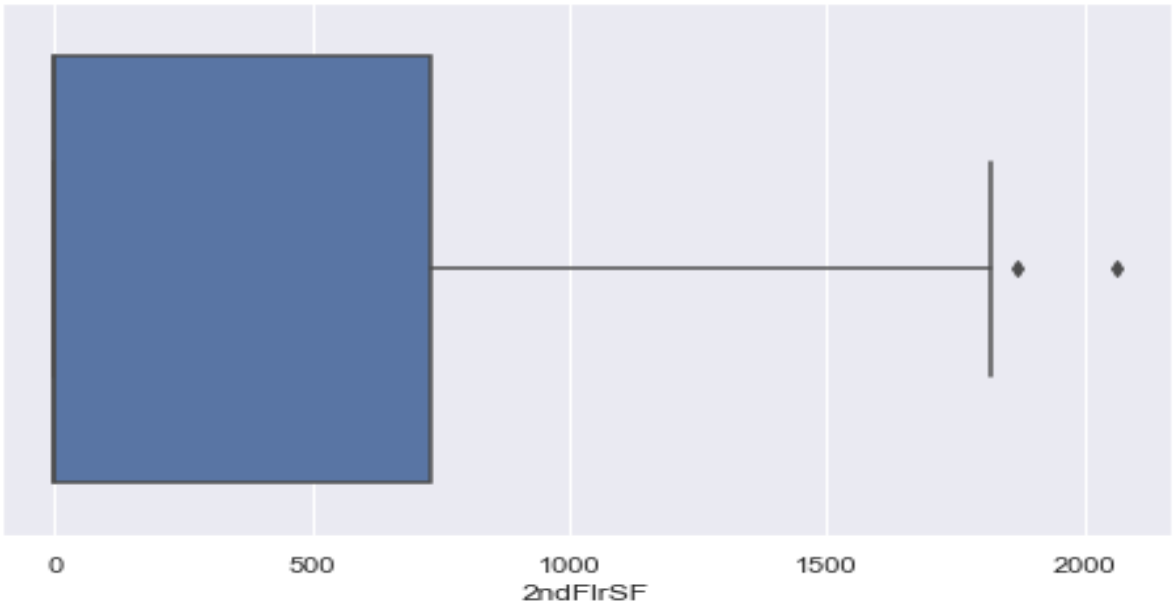

1950
1960
1970
1980
1990
2000
2010
YearRemodAdd

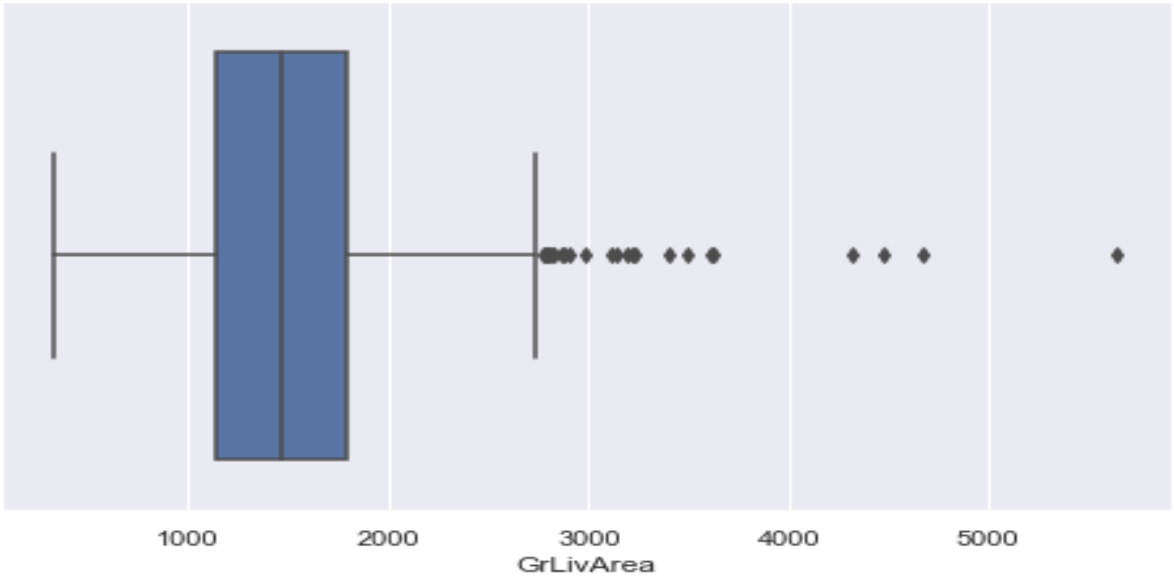

0
200
400
600
800
1000
1200
1400
1600
MasVnrArea

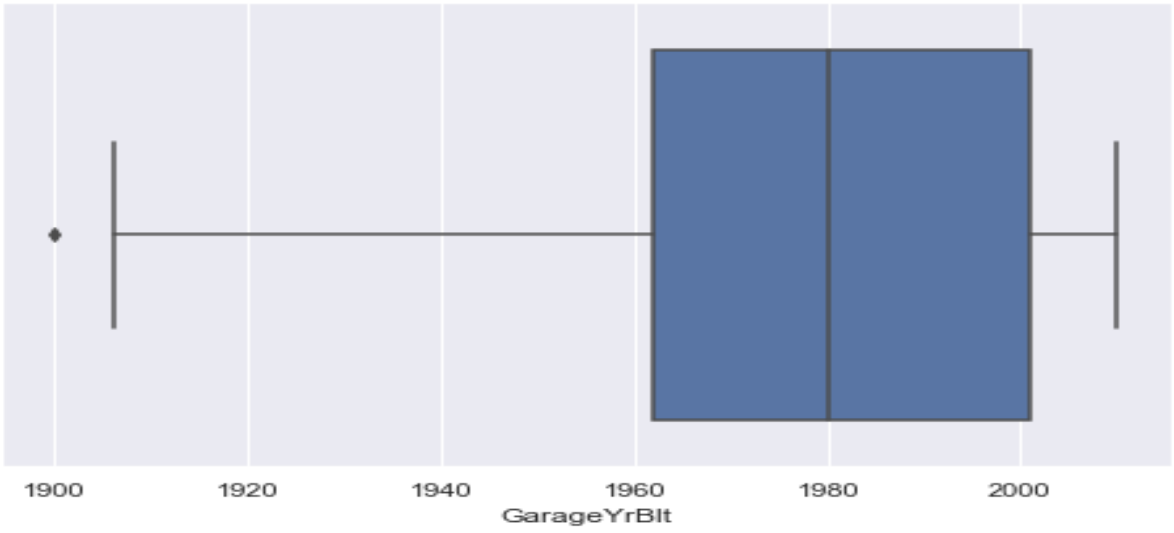

0
500
1000
1500
2000
BsmtUnfSF

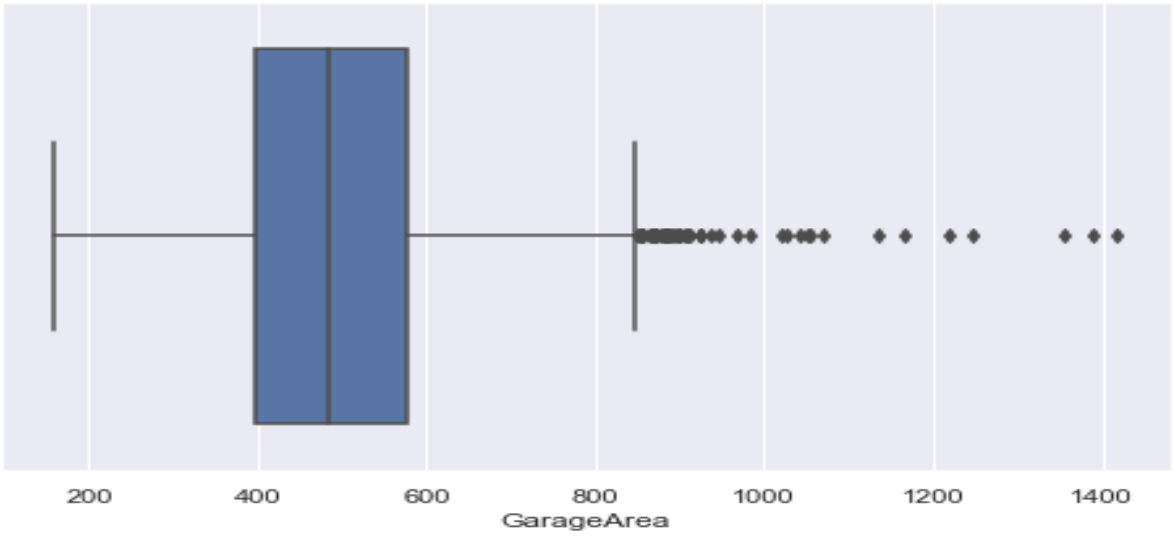

0
1000
2000
3000
4000
5000
6000
TotalBsmtSF

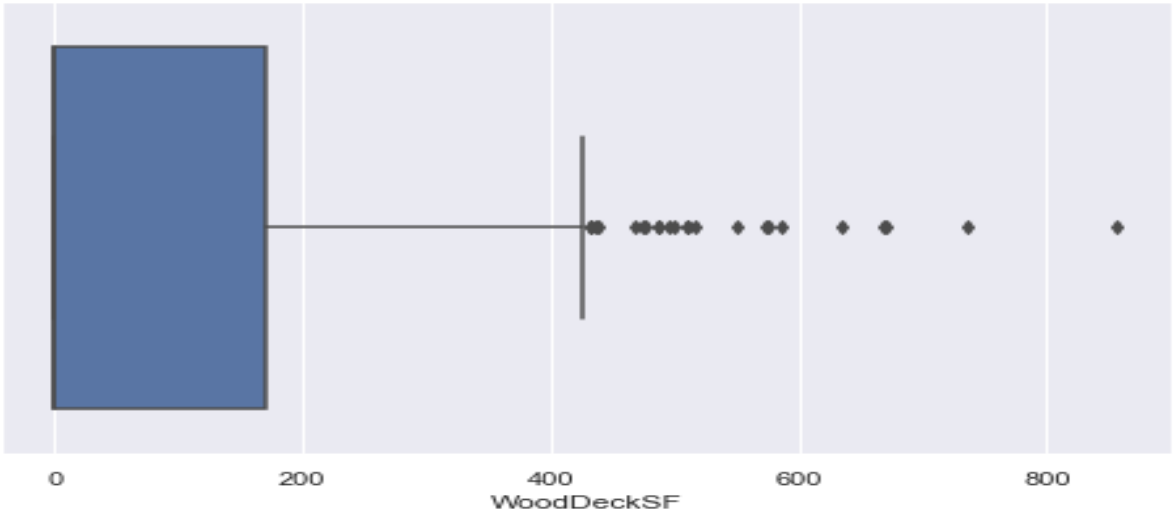

1000
2000
3000
4000
1stFlrSF

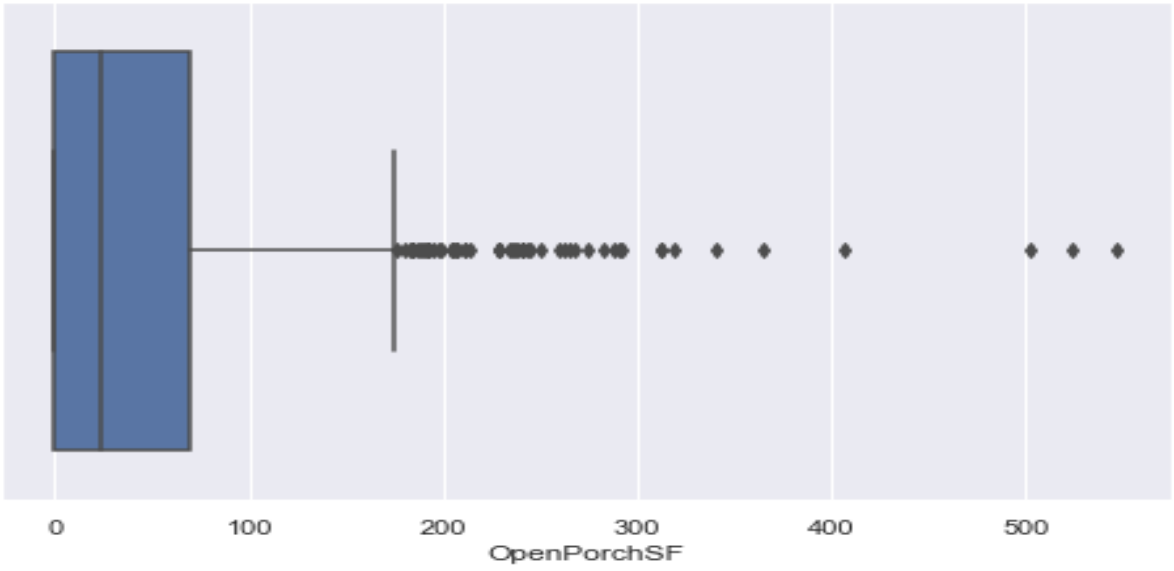

0
500
1000
1500
2000
2ndFlrSF

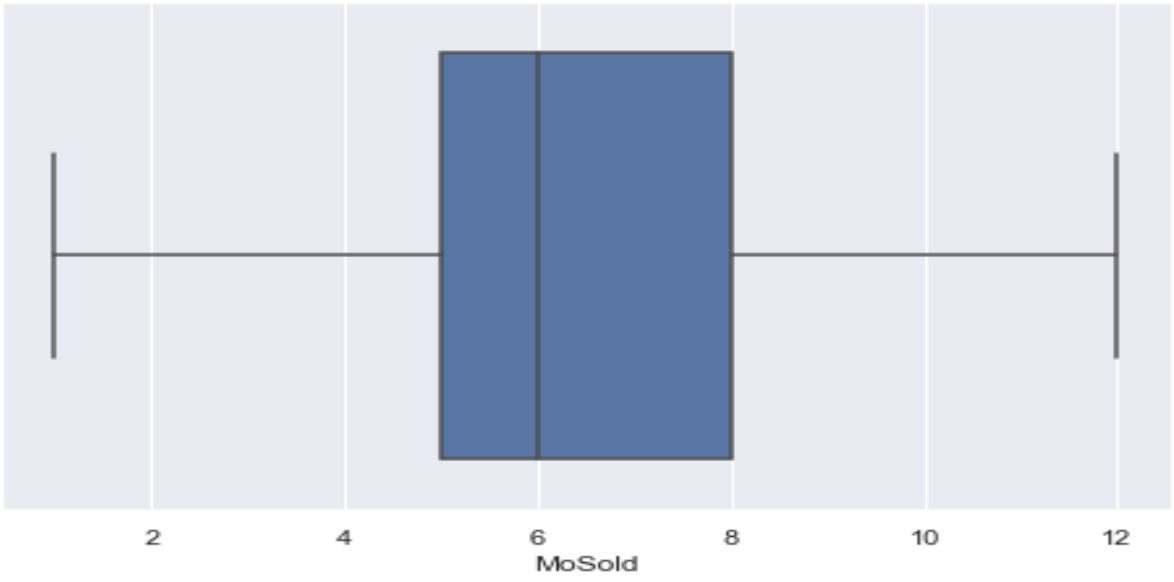

1000
2000
3000
4000
5000
GrLivArea

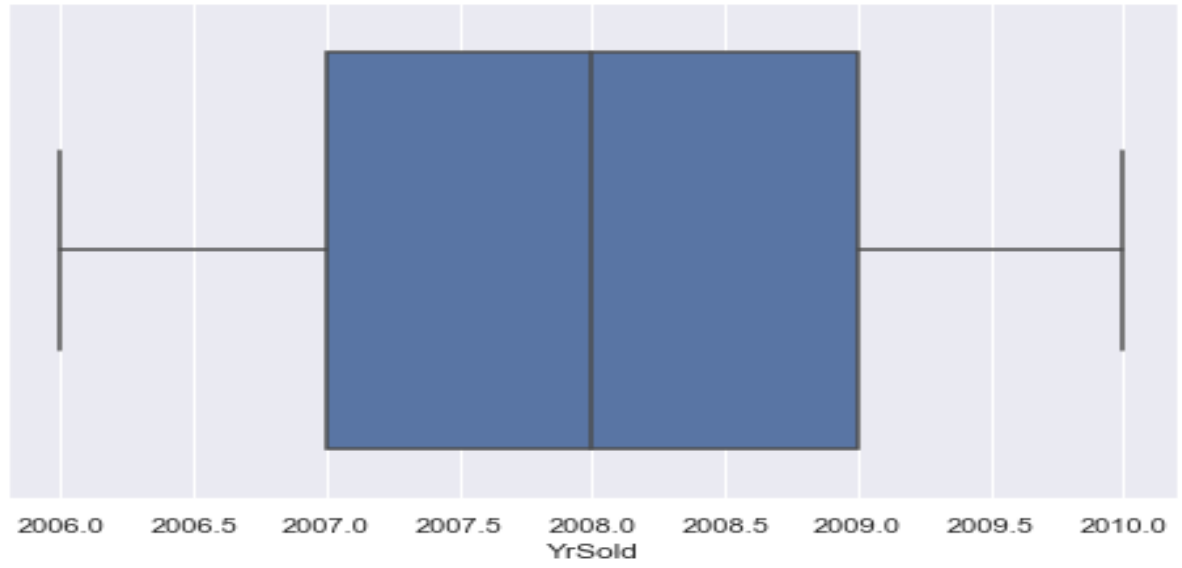

1900
1920
1940
1960
1980
2000
GarageYrBlt

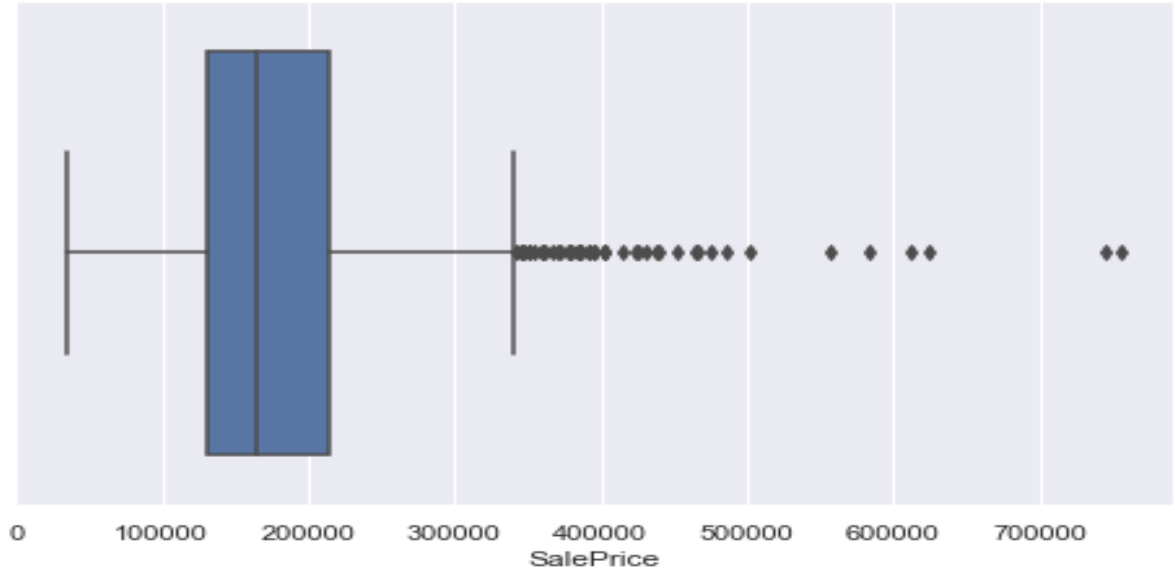

200
400
600
800
1000
1200
1400
GarageArea


0
200
400
600
800
WoodDeckSF


0
100
200
300
400
500
OpenPorchSF


2
4
6
8
10
12
MoSold

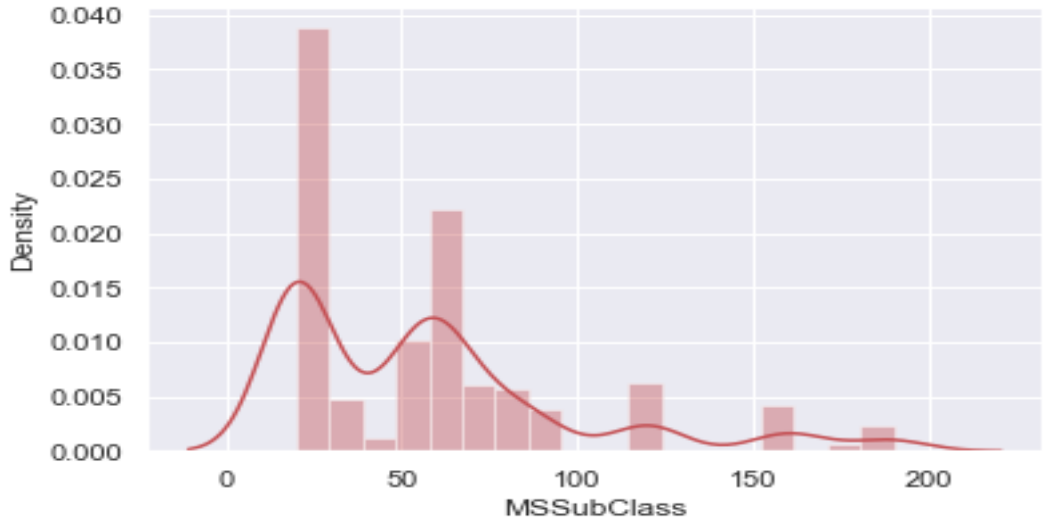


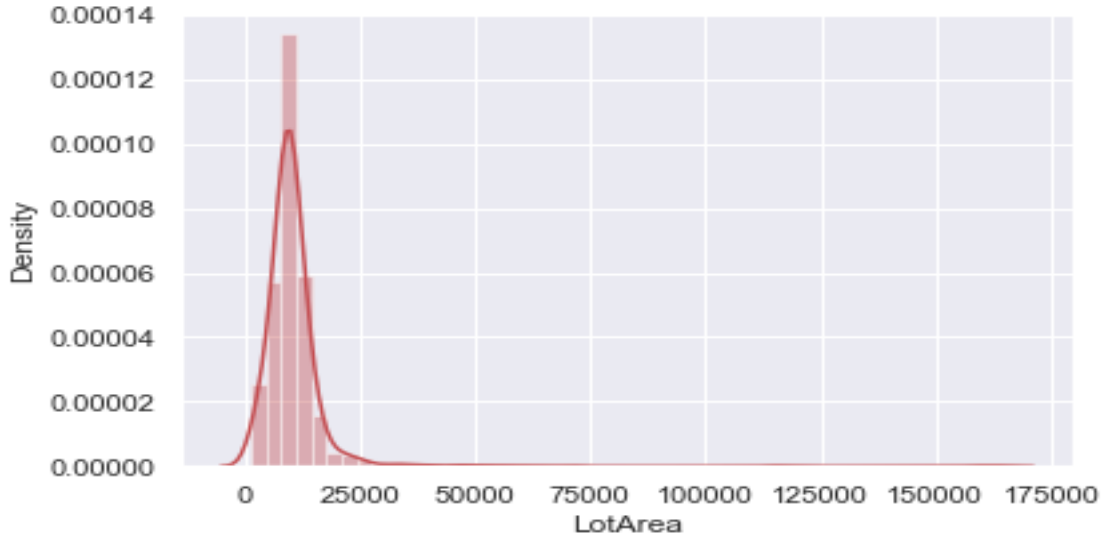


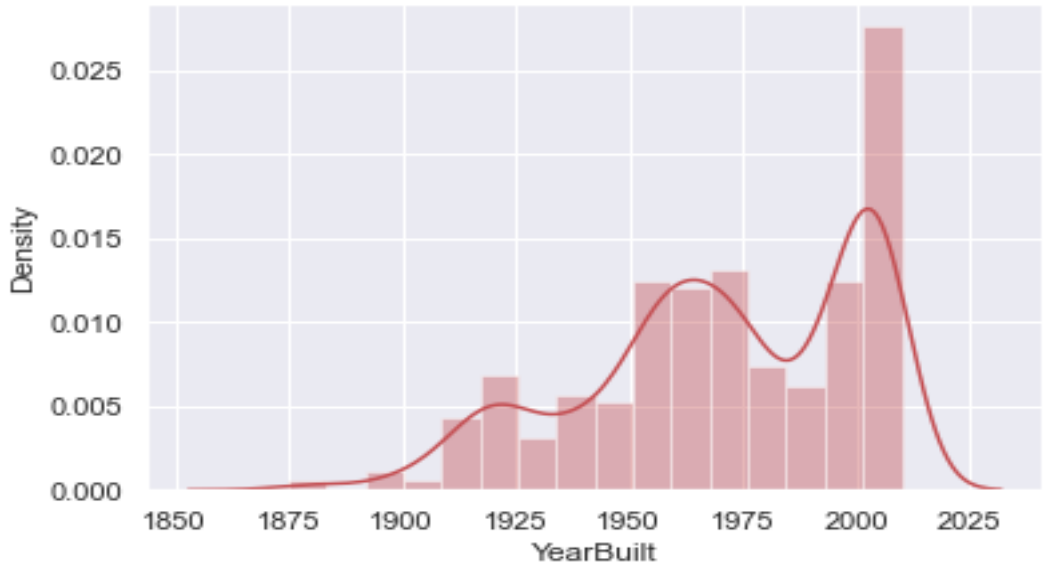
Keys: Columns are having outliers, but when I performed IQR and zscore method to remove outliers these both method was giving high loss of data.

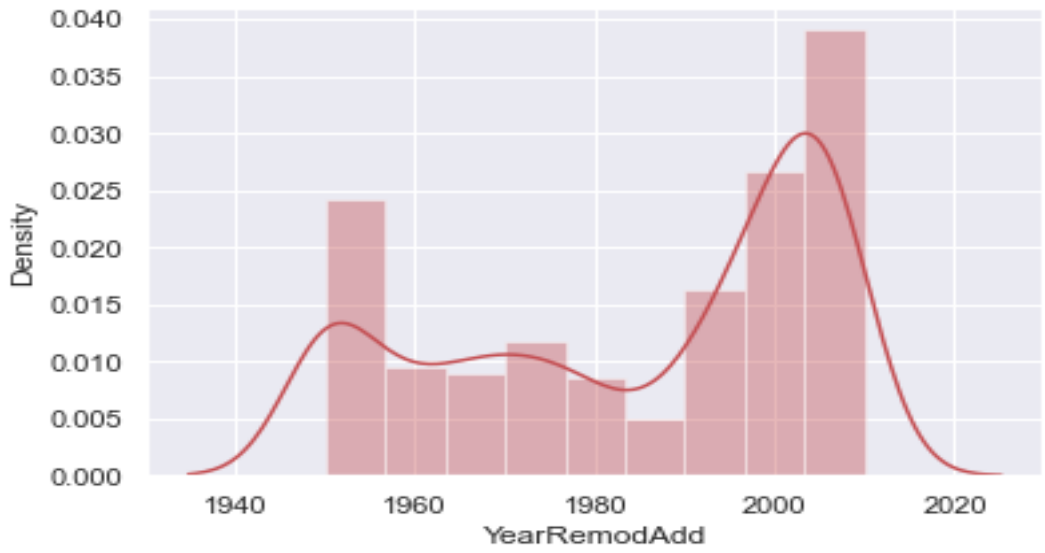
Therefore, outliers not removed and worked on existing dataset

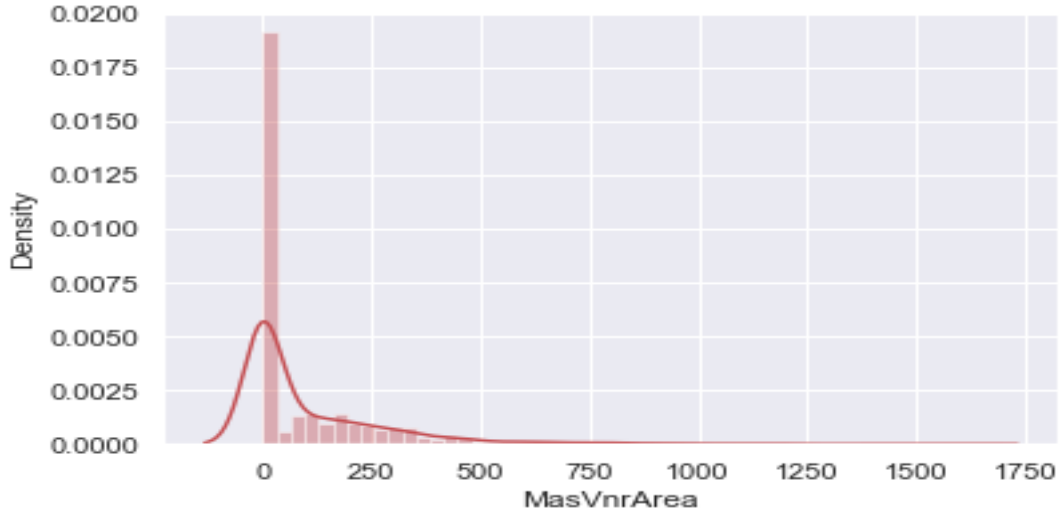
Distribution and Skewness:

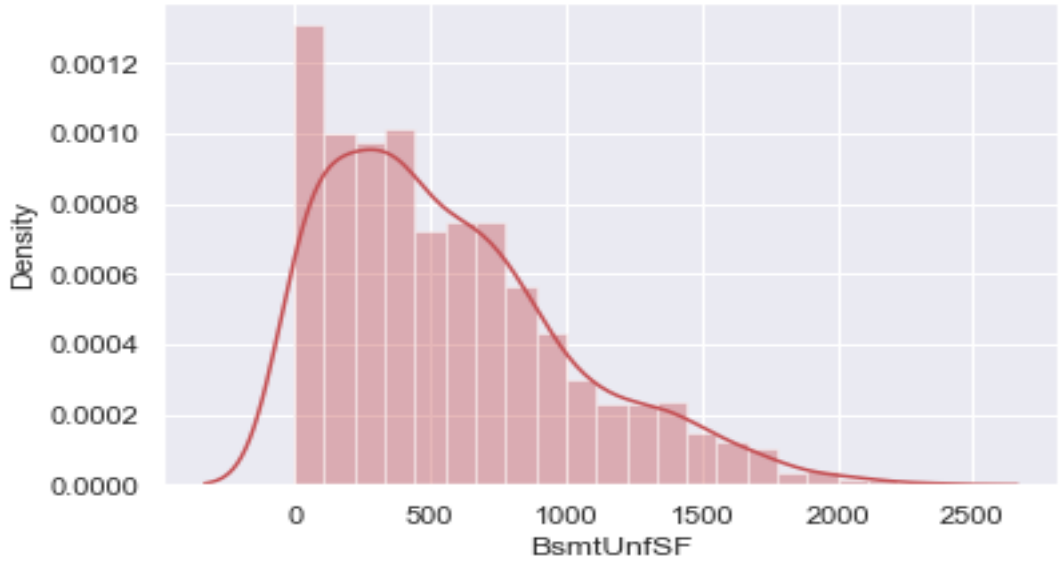


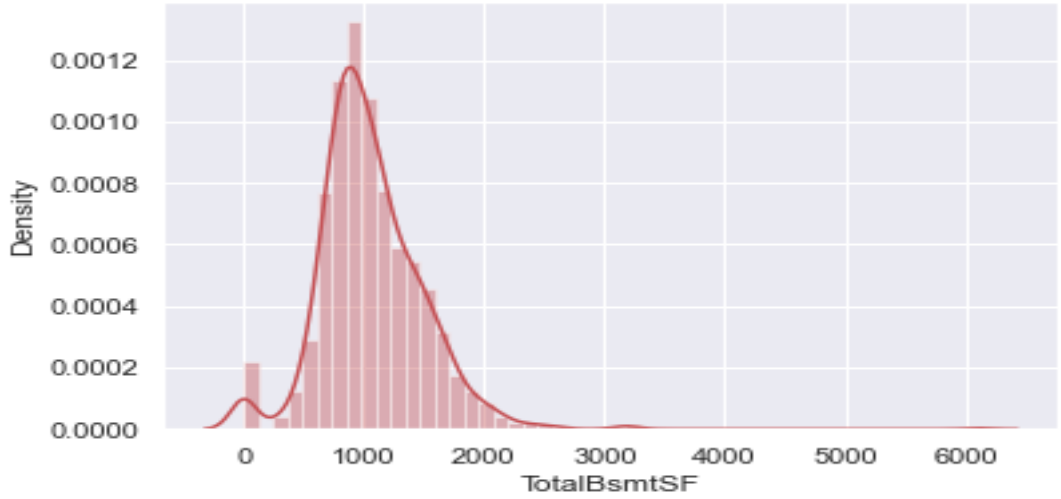


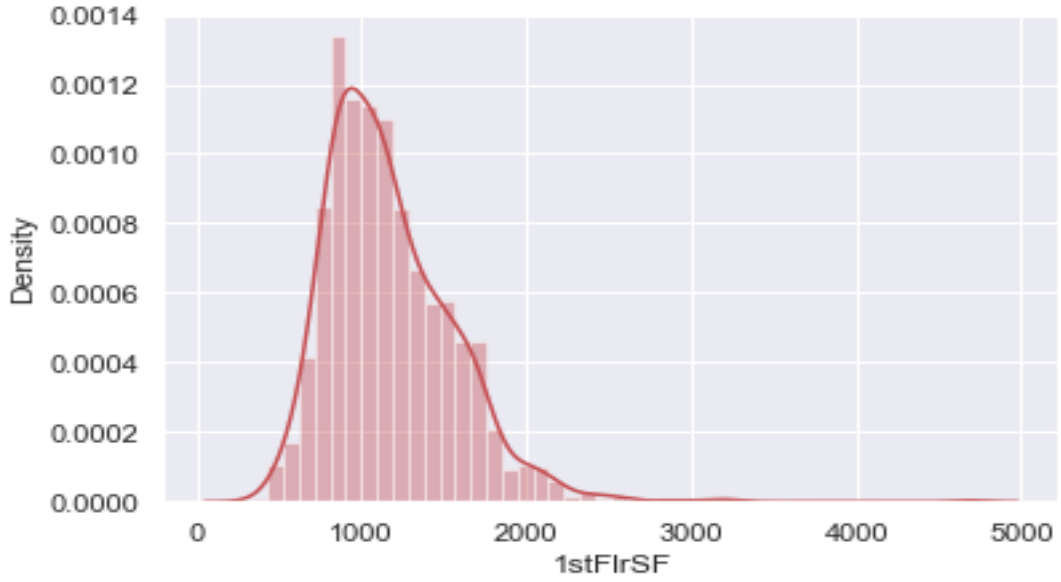


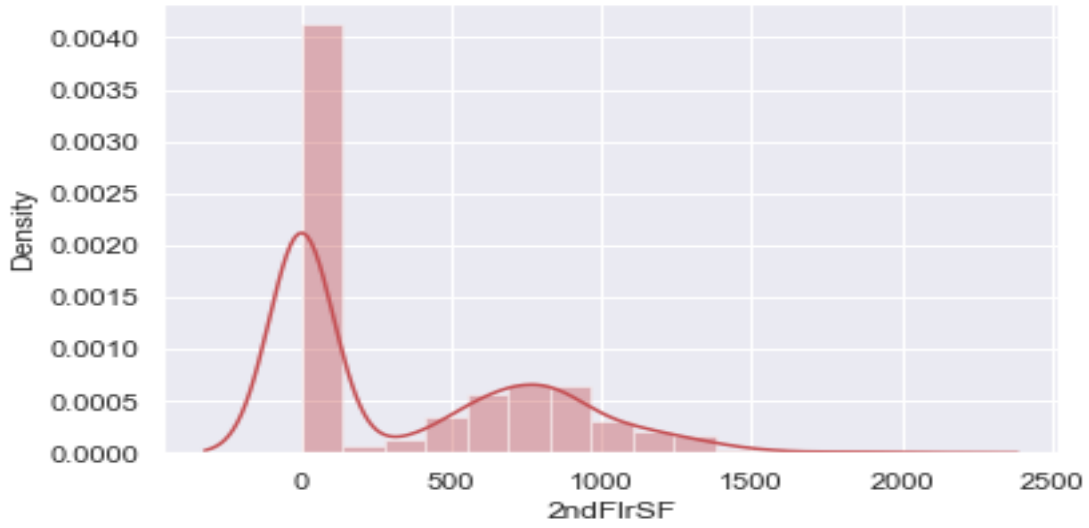

0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
Density
1940
1960
1980
2000
2020
YearRemodAdd

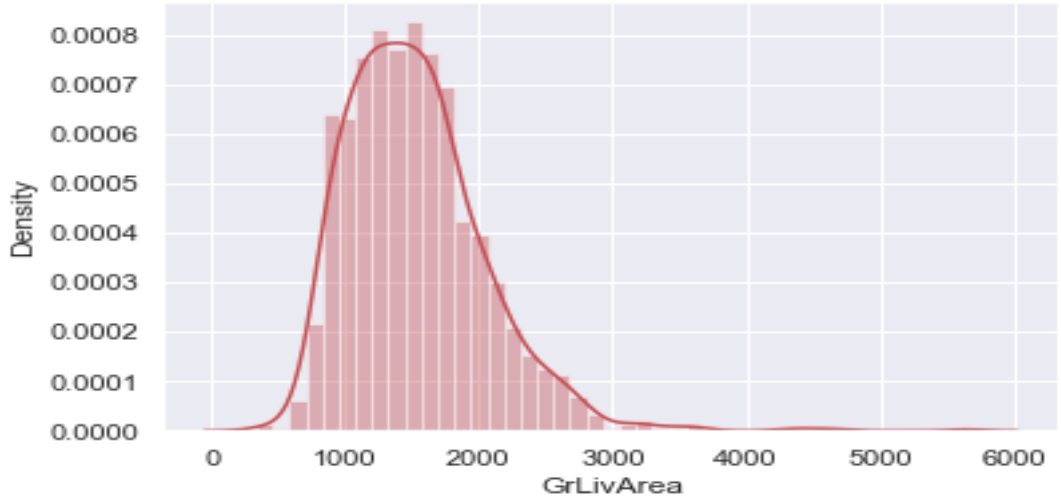

0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
Density
0
250
500
750
1000
1250
1500
1750
MasVnrArea

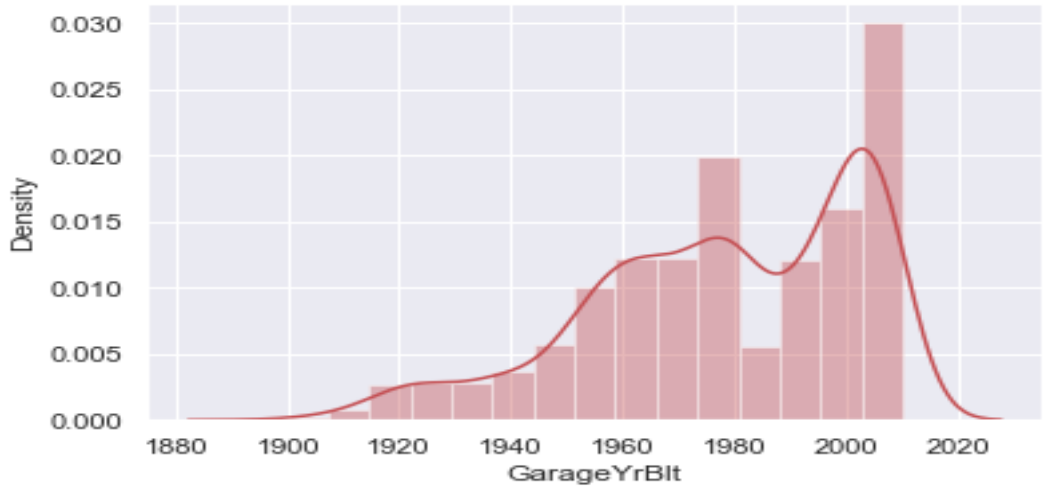

0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
500
1000
1500
2000
2500
BsmtUnfSF

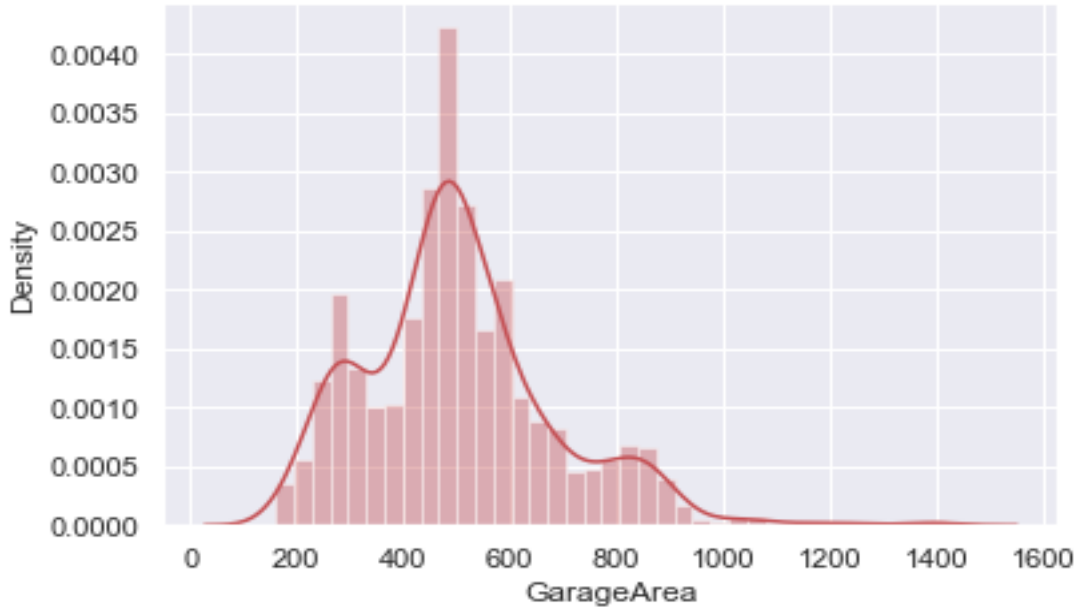

Density
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0
1000
2000
3000
4000
5000
6000
TotalBsmtSF

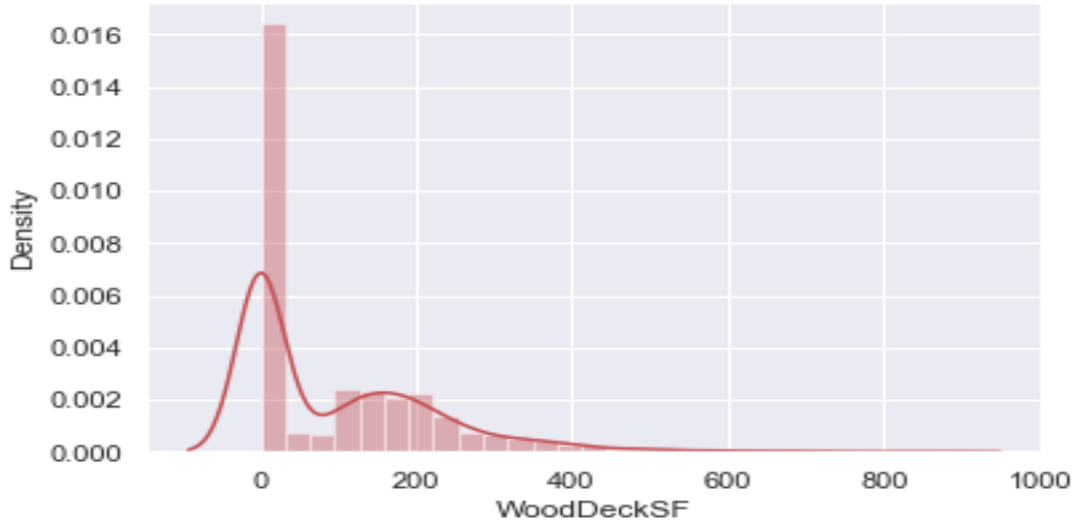

Density
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0
1000
2000
3000
4000
5000
1stFlrSF

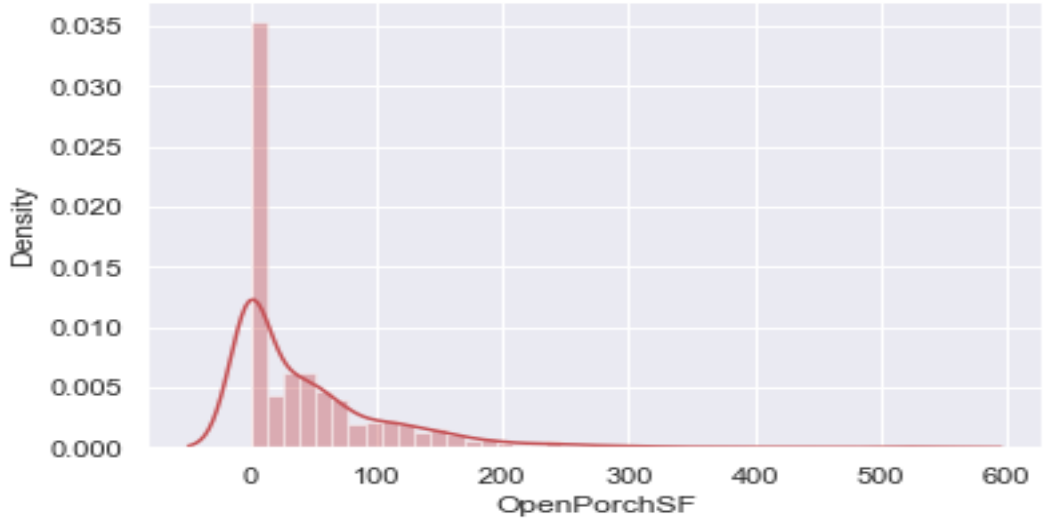

Density
0.0040
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
500
1000
1500
2000
2500
2ndFlrSF

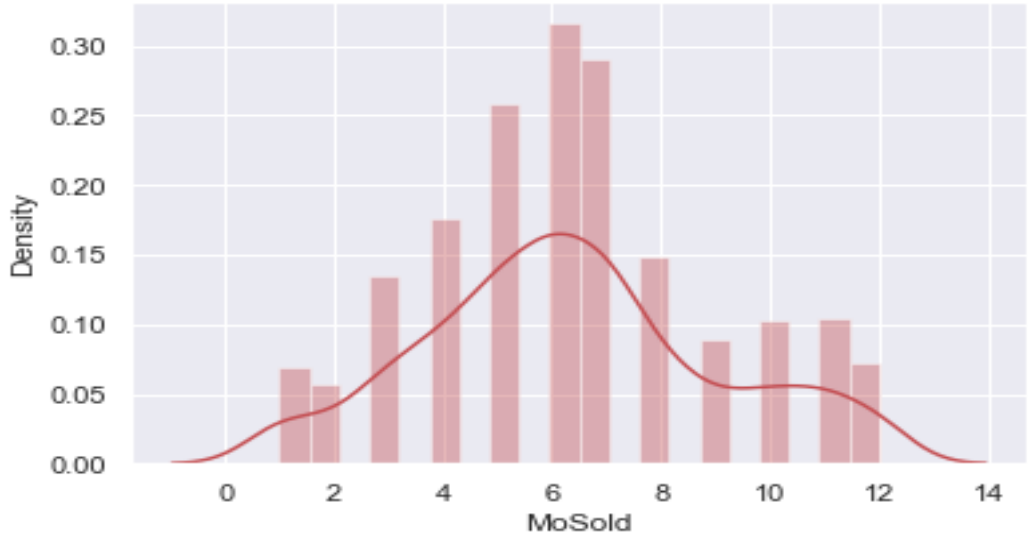

Density
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
0
1000
2000
3000
4000
5000
6000
GrLivArea

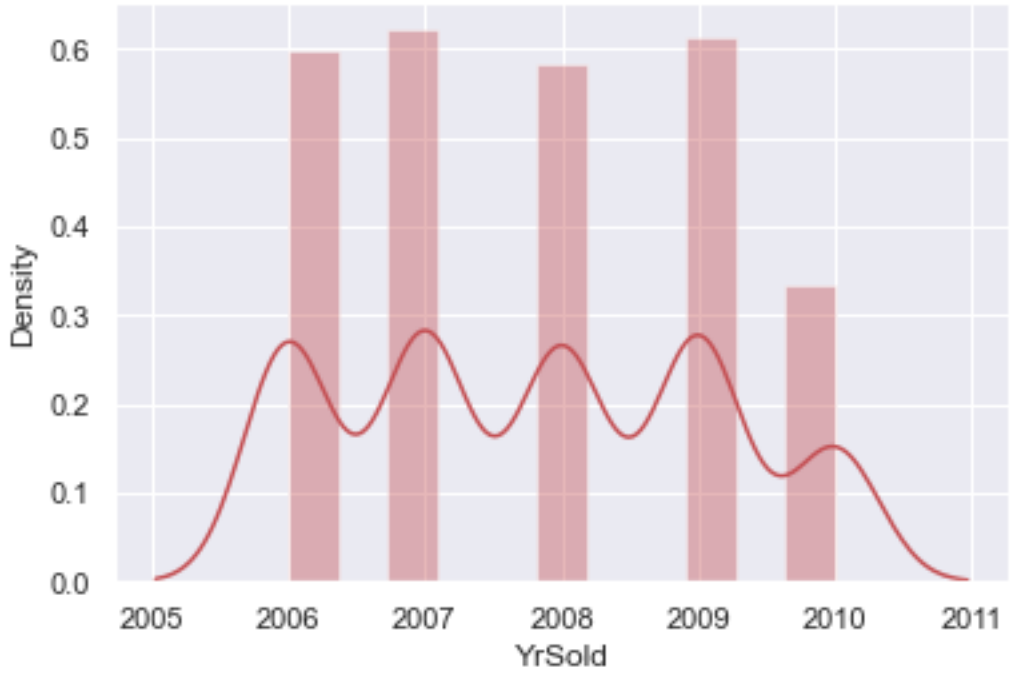

Density
0.030
0.025
0.020
0.015
0.010
0.005
0.000
1880
1900
1920
1940
1960
1980
2000
2020
GarageYrBlt

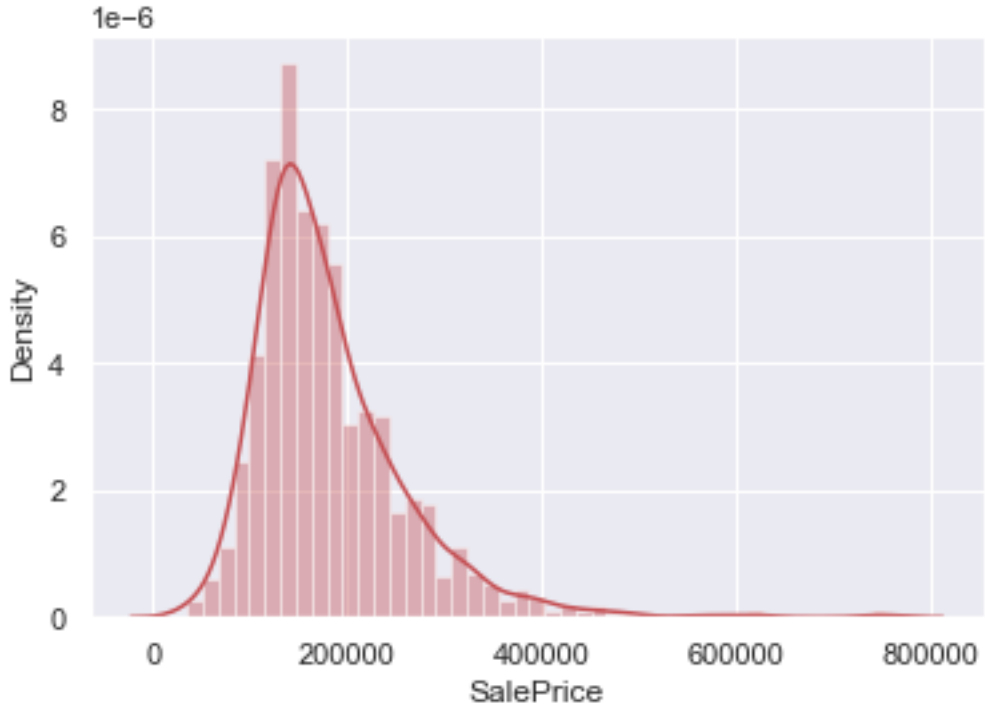

Density
0.0040
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
200
400
600
800
1000
1200
1400
1600
GarageArea


0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
Density
0
200
400
600
800
1000
WoodDeckSF

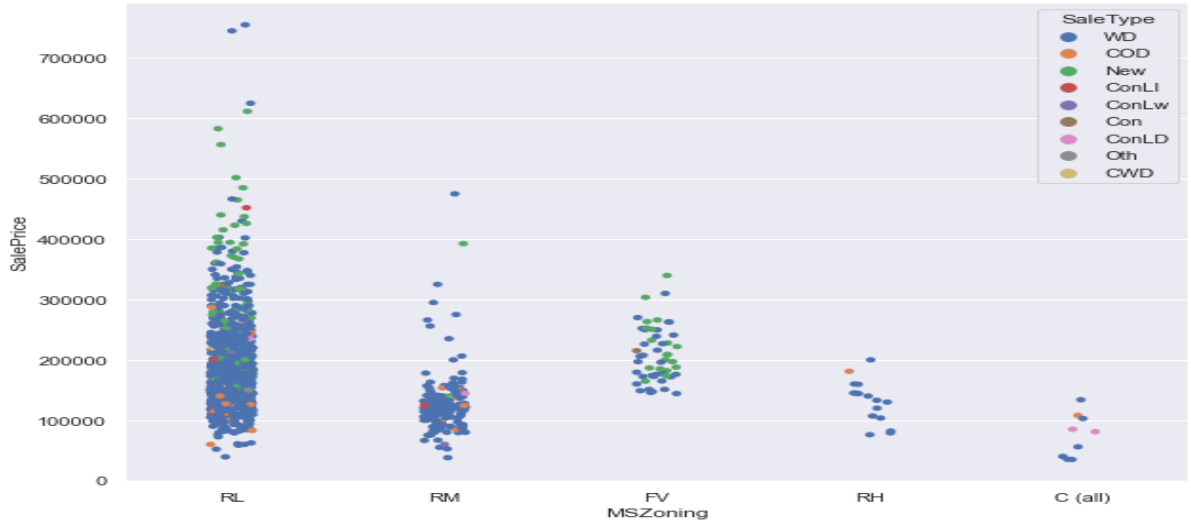

0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
Density
0
100
200
300
400
500
600
OpenPorchSF

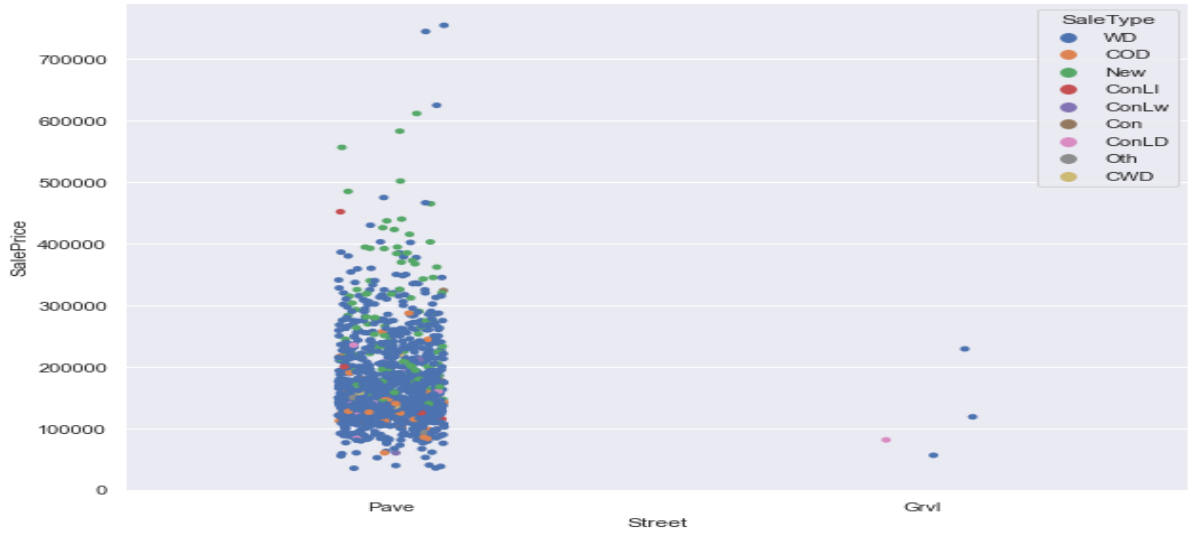

0.30
0.25
0.20
0.15
0.10
0.05
0.00
Density
0
2
4
6
8
10
12
14
MoSold

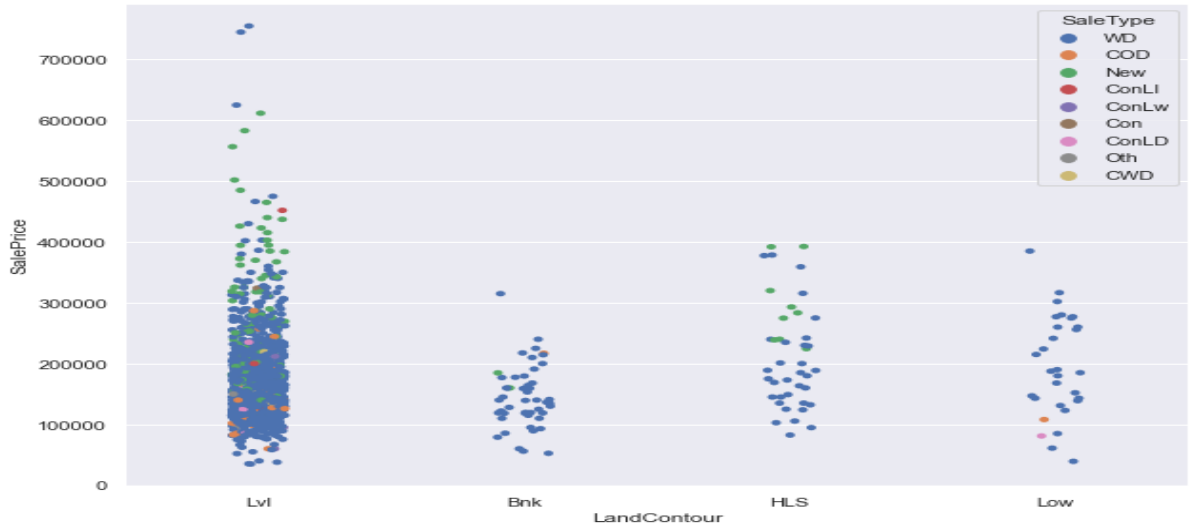

0.6
0.5
0.4
0.3
0.2
0.1
0.0
Density
2005
2006
2007
2008
2009
2010
2011
YrSold


1e−6
8
6
4
2
0
Density
0
200000
400000
600000
800000
SalePrice

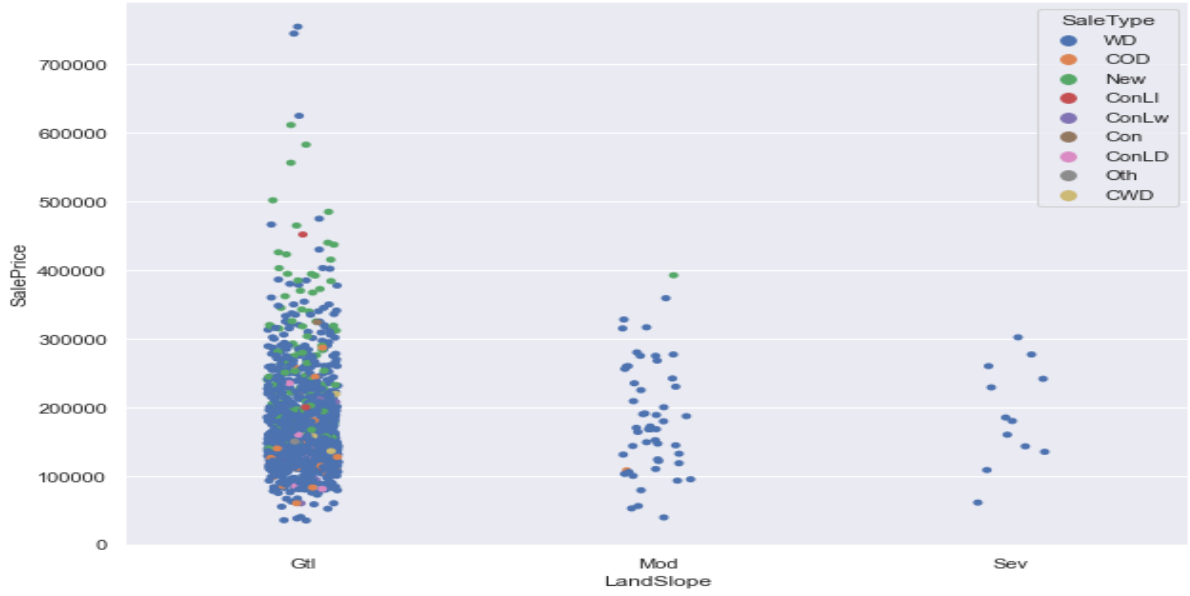
Bi-variate Analysis:

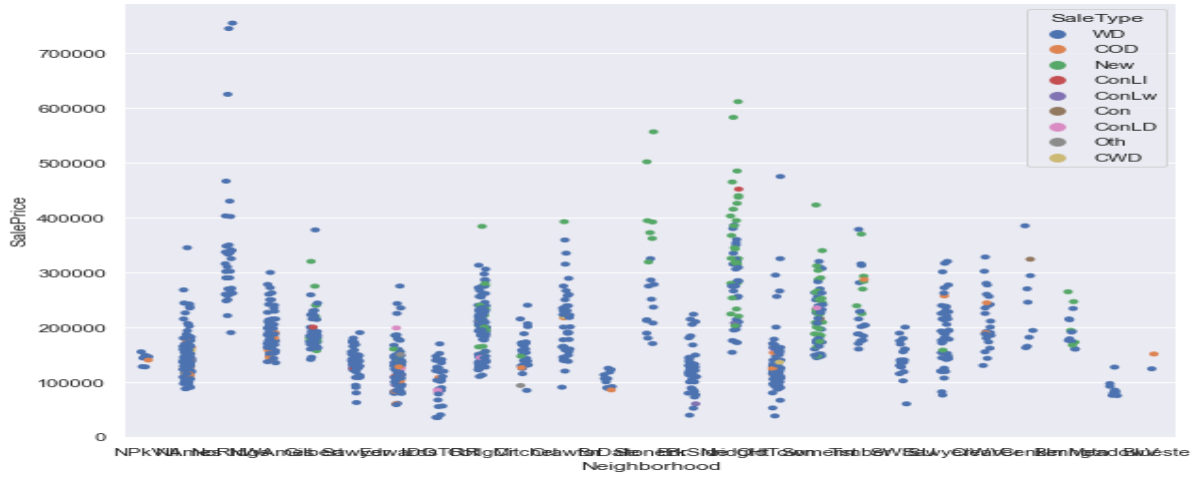


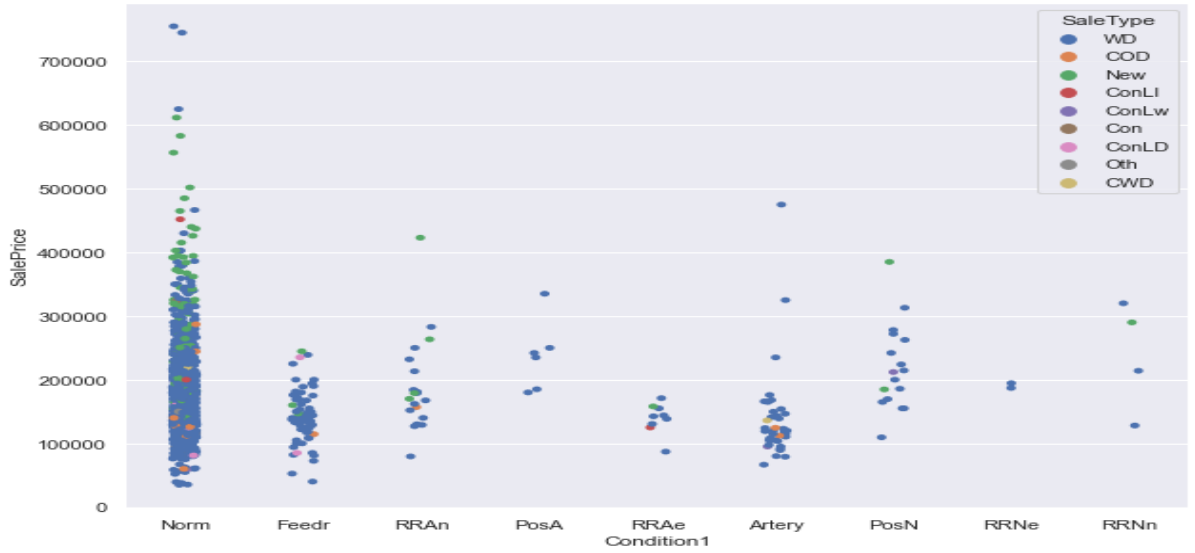


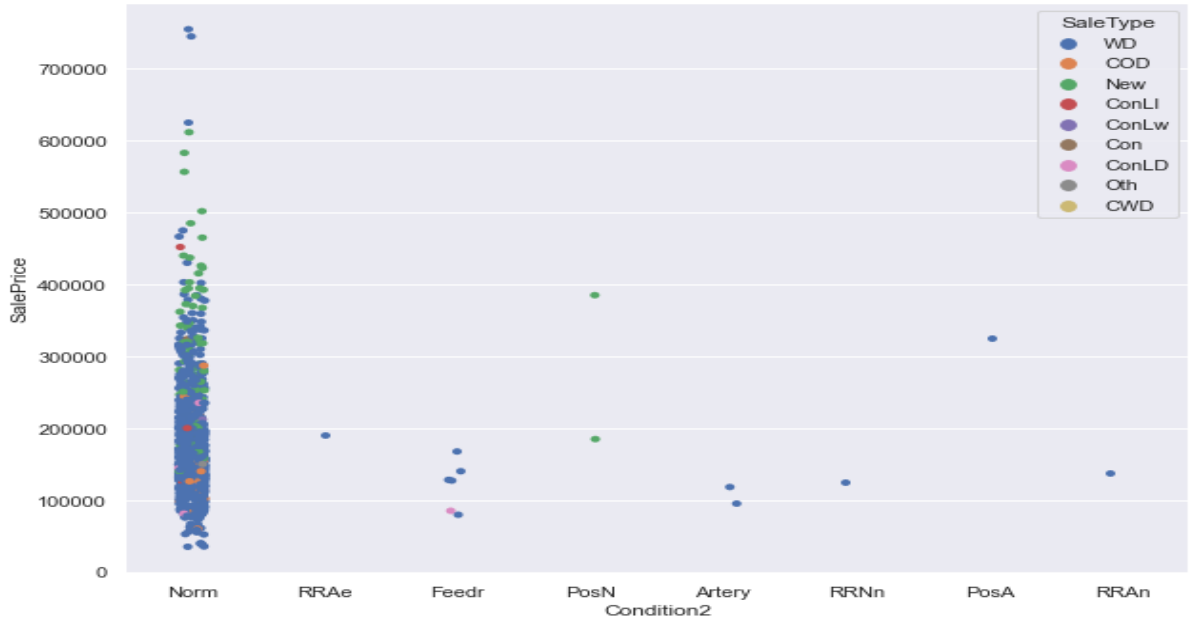

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Lvl
Bnk
HLS
Low
LandContour

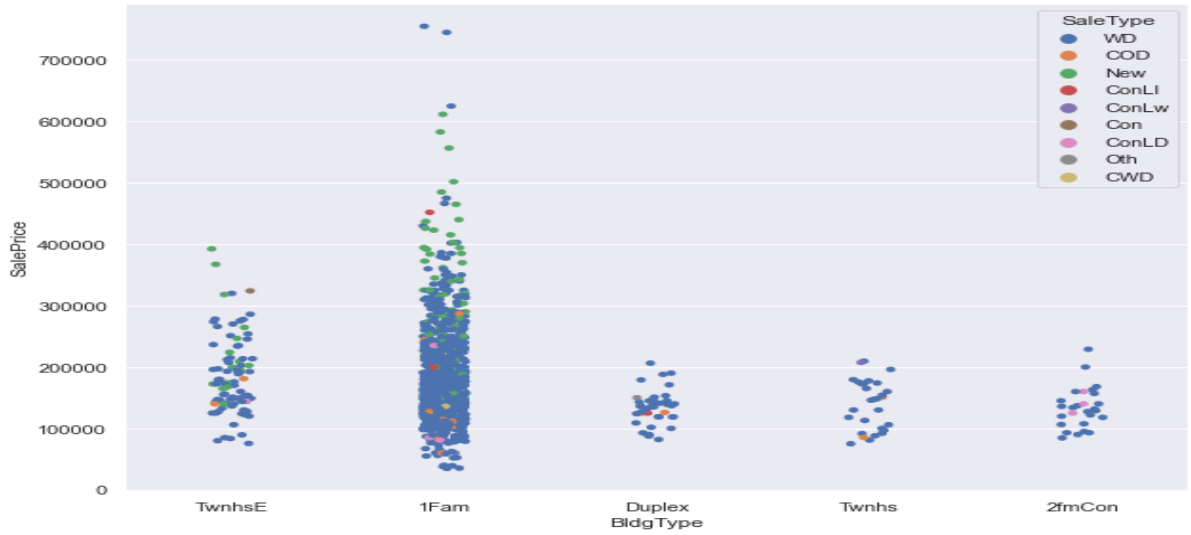

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Inside
CulDSac
FR2
Corner
FR3
LotConfig

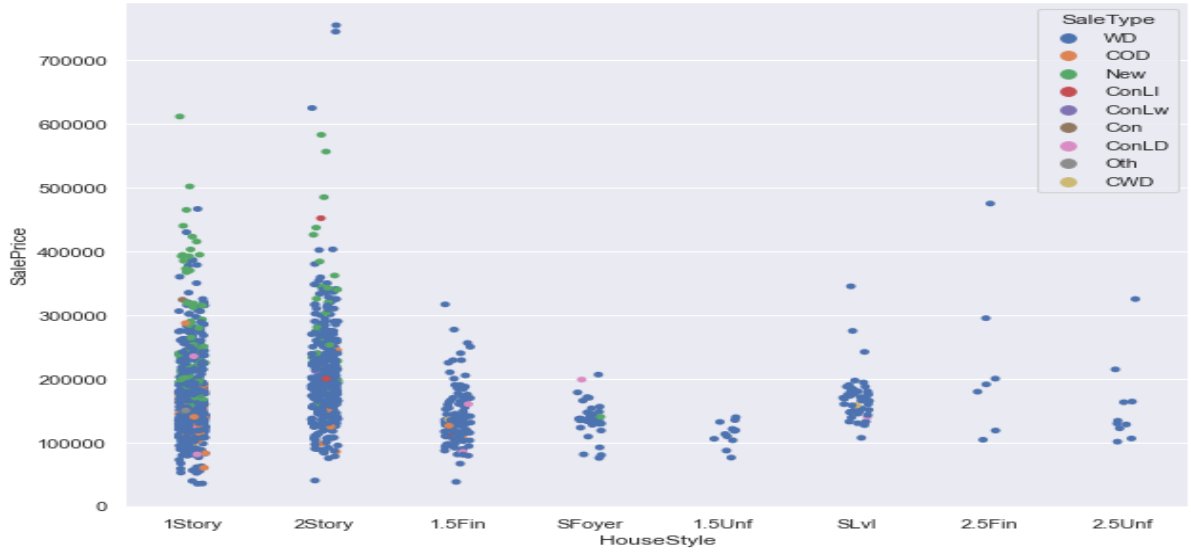

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Gtl
Mod
Sev
LandSlope

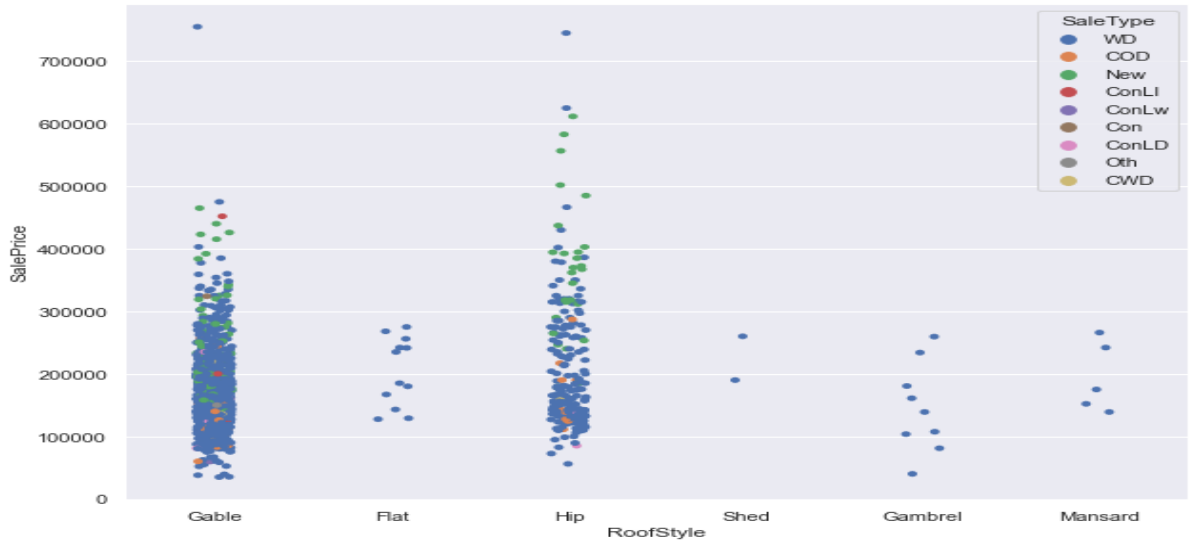

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Neighborhood

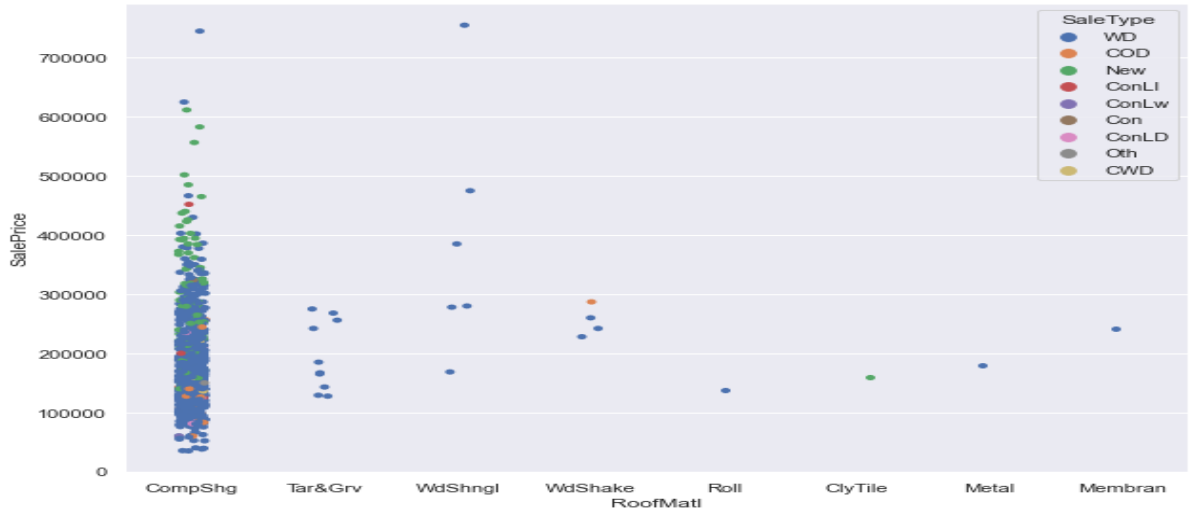

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Norm
Feedr
RRAn
PosA
RRAe
Artery
PosN
RRNe
RRNn
Condition1

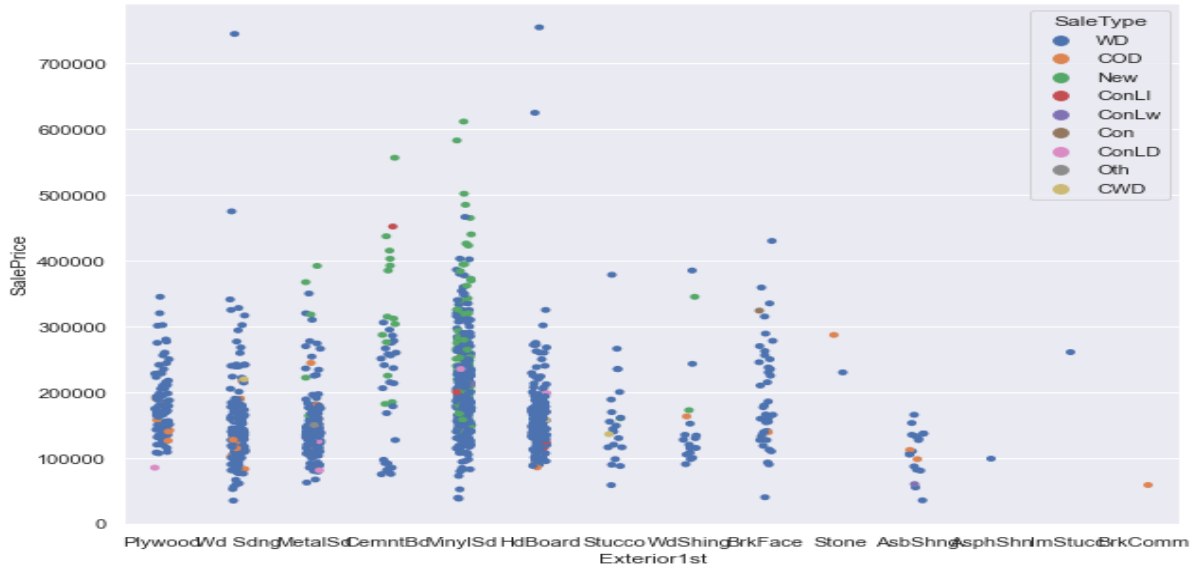

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Norm
RRAe
Feedr
PosN
Artery
RRNn
PosA
RRAn
Condition2

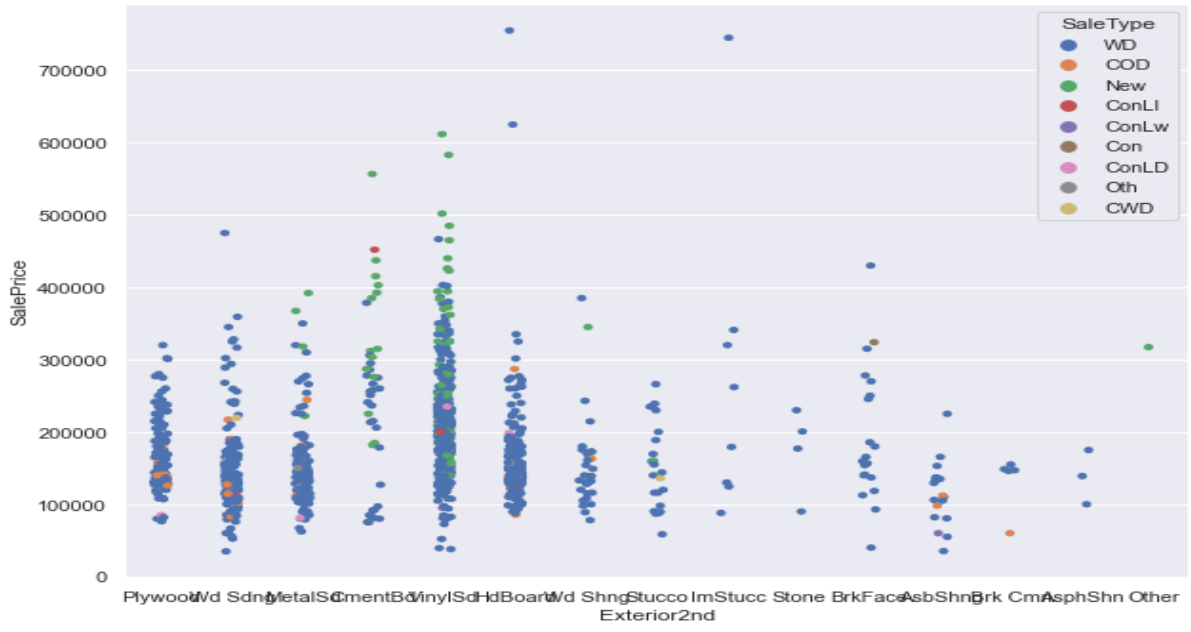

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TwnhsE
1Fam
Duplex
Twnhs
2fmCon
BldgType

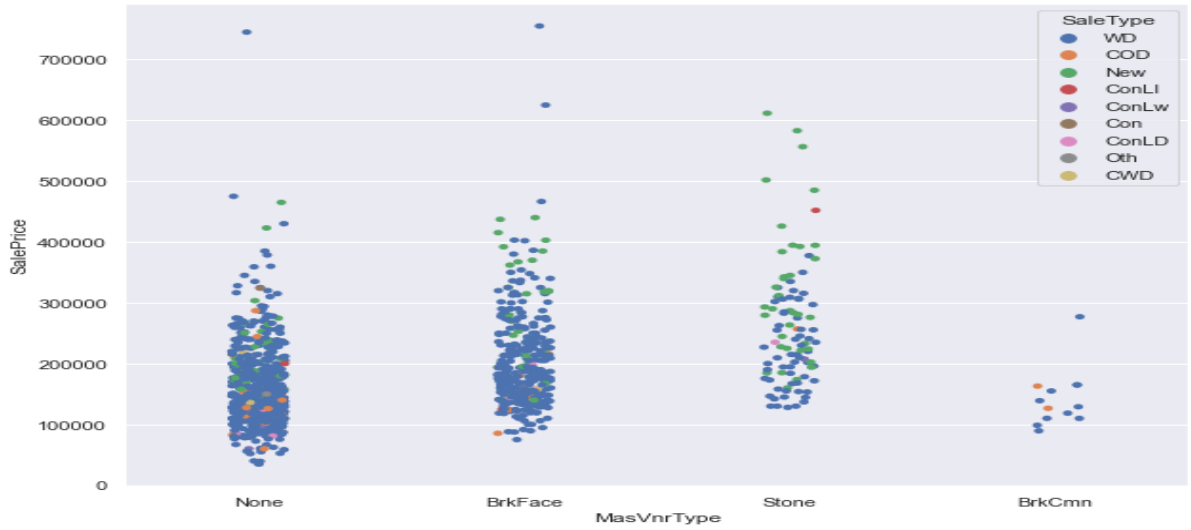

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
1Story
2Story
1.5Fin
SFoyer
1.5Unf
SLvl
2.5Fin
2.5Unf
HouseStyle

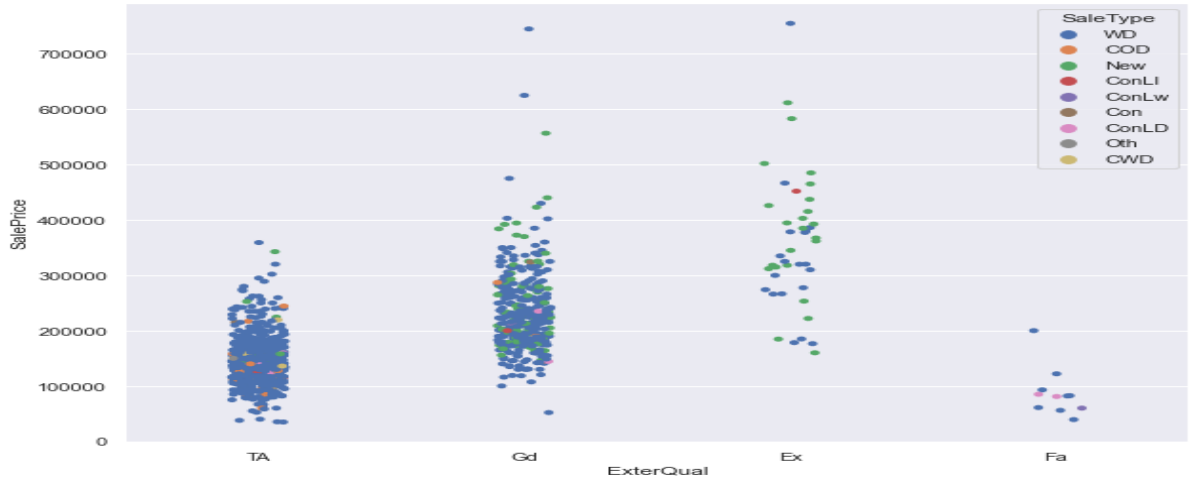

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Gable
Flat
Hip
Shed
Gambrel
Mansard
RoofStyle

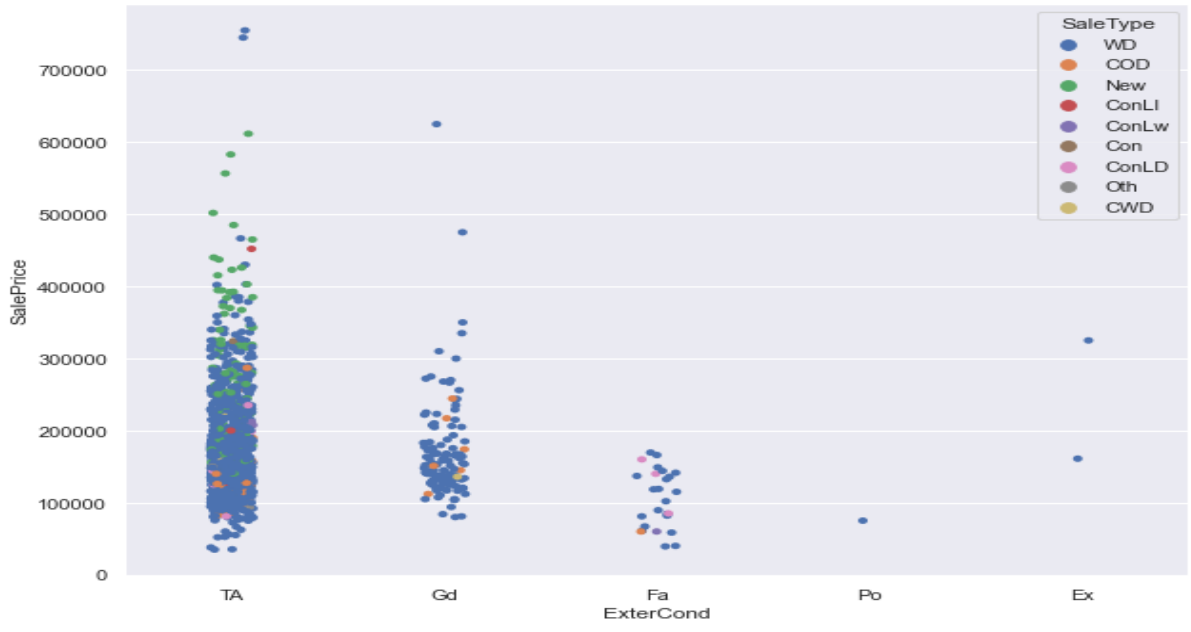

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
CompShg
Tar&Grv
WdShngl
WdShake
Roll
ClyTile
Metal
Membran
RoofMatl


SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Plywood
Wd Sdng
MetalSd
CemntBd
VinylSd
HdBoard
Stucco
WdShing
BrkFace
Stone
AsbShng
AsphShn
ImStucc
BrkComm
Exterior1st

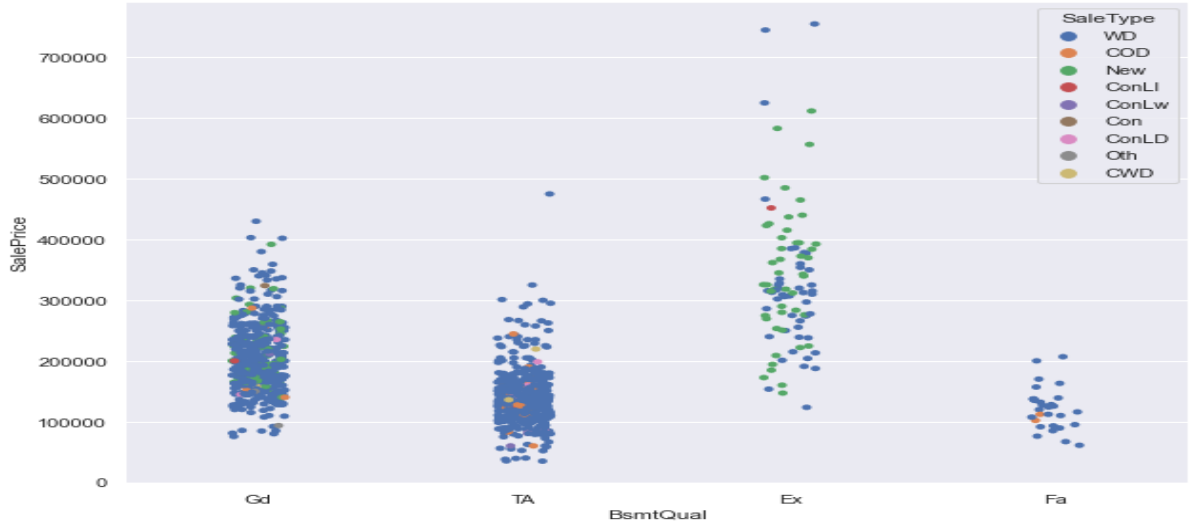

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Plywood
Wd Sdng
MetalSd
CmentBd
VinylSd
HdBoard
Wd Shng
Stucco
ImStucc
Stone
BrkFace
AsbShng
Brk Cmn
AsphShn
Other
Exterior2nd

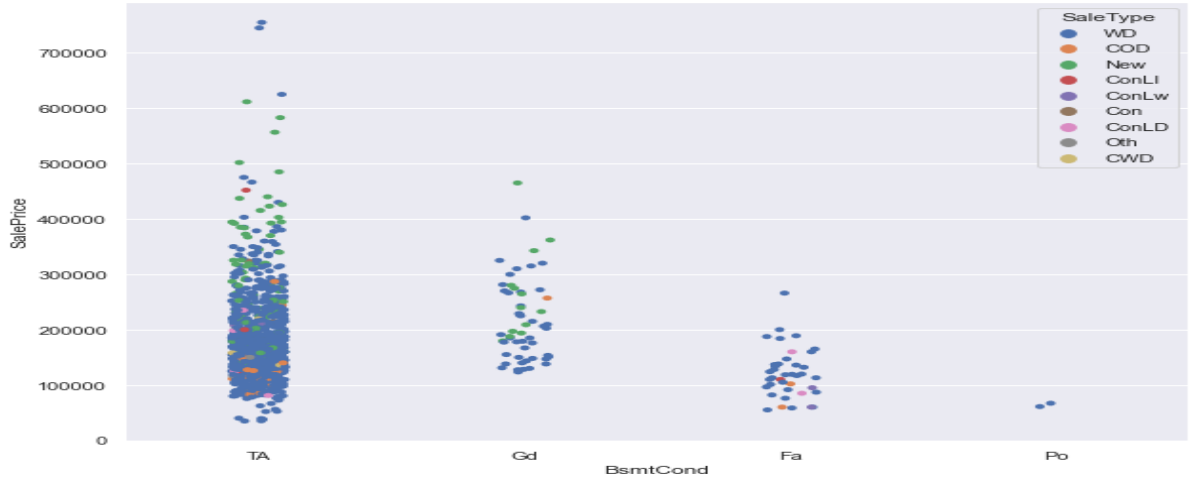

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
None
BrkFace
Stone
BrkCmn
MasVnrType

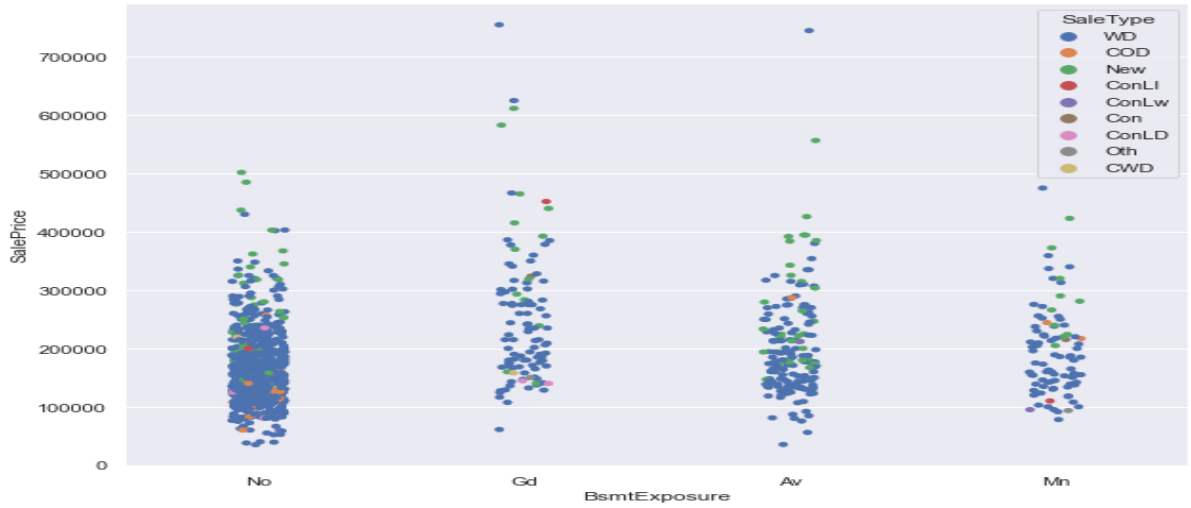

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Gd
Ex
Fa
ExterQual

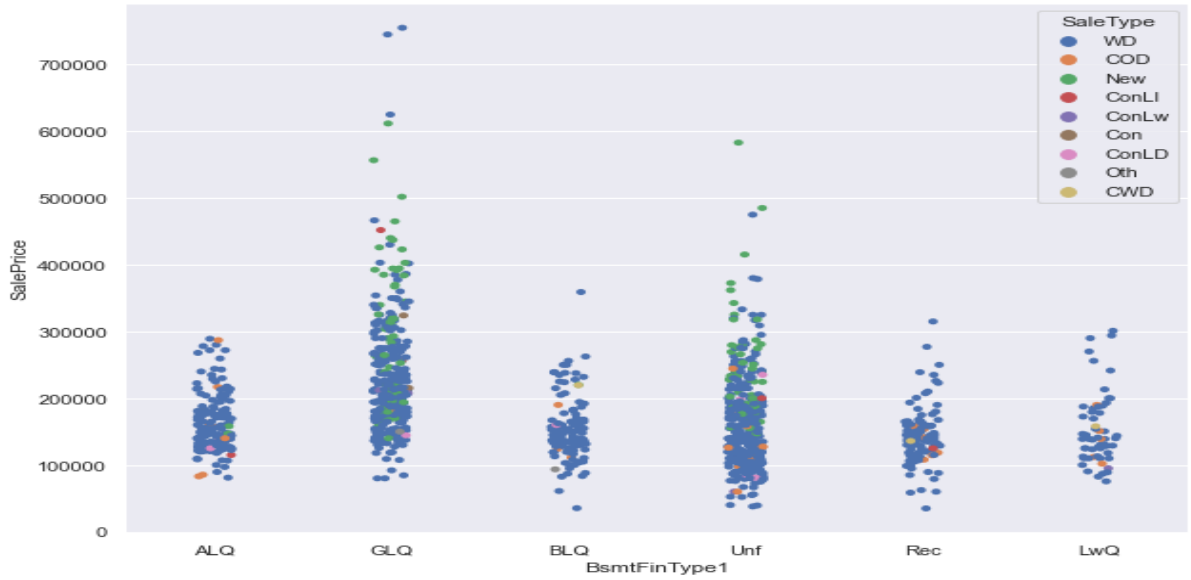

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Gd
Fa
Po
Ex
ExterCond

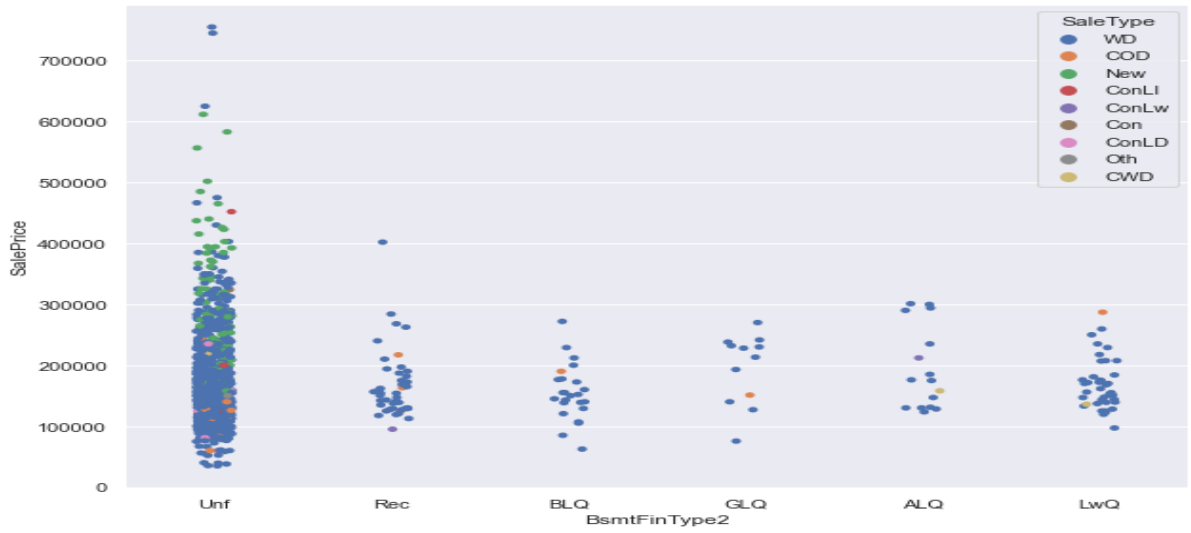

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
CBlock
PConc
BrkTil
Slab
Stone
Wood
Foundation

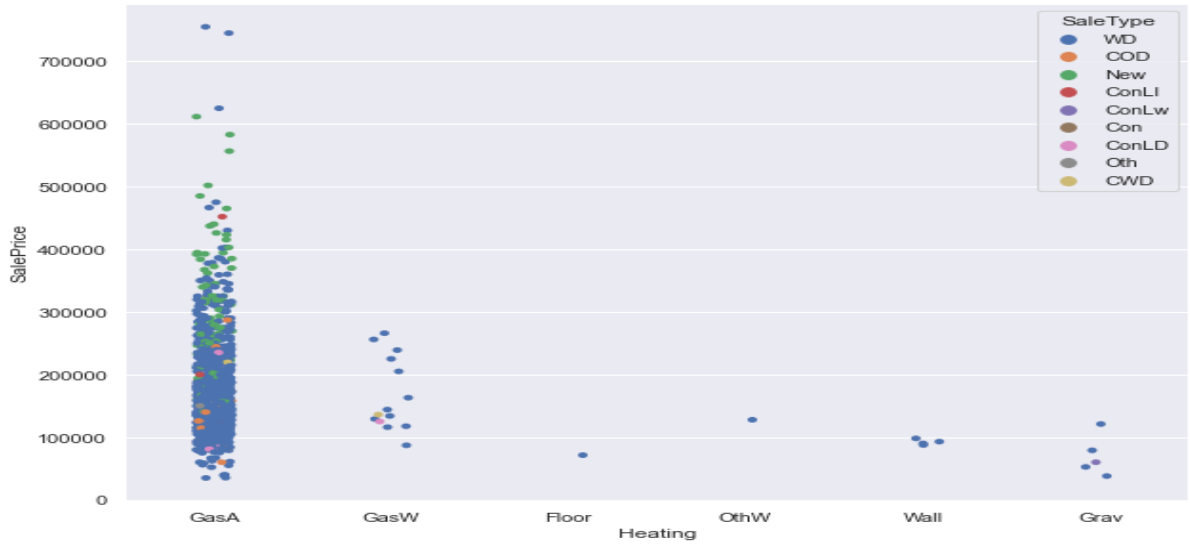

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Gd
TA
Ex
Fa
BsmtQual

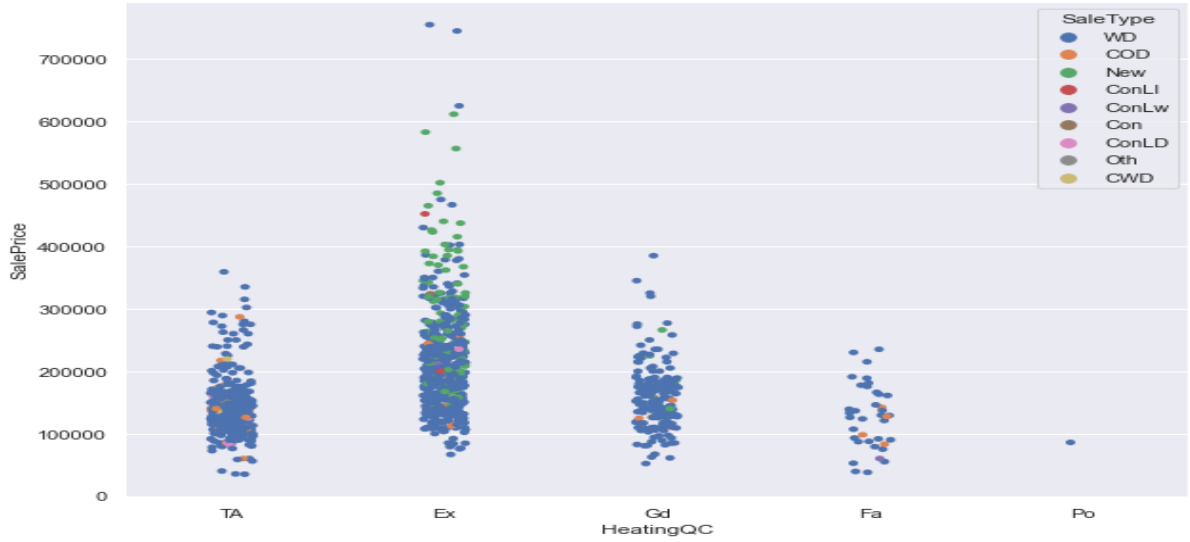

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Gd
Fa
Po
BsmtCond


SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
No
Gd
Av
Mn
BsmtExposure

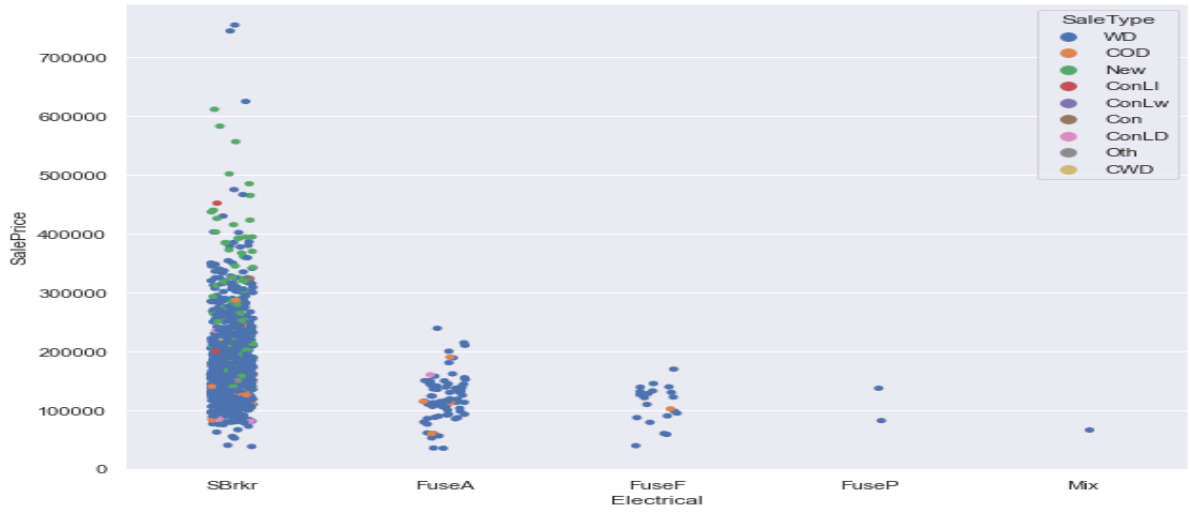

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
ALQ
GLQ
BLQ
Unf
Rec
LwQ
BsmtFinType1

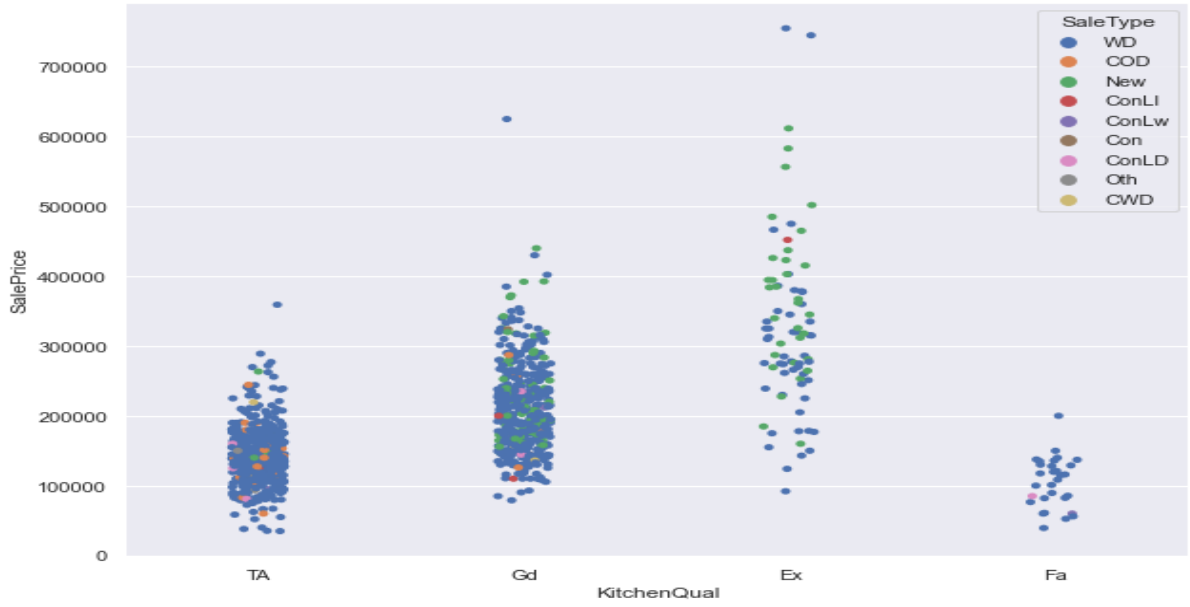

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Unf
Rec
BLQ
GLQ
ALQ
LwQ
BsmtFinType2

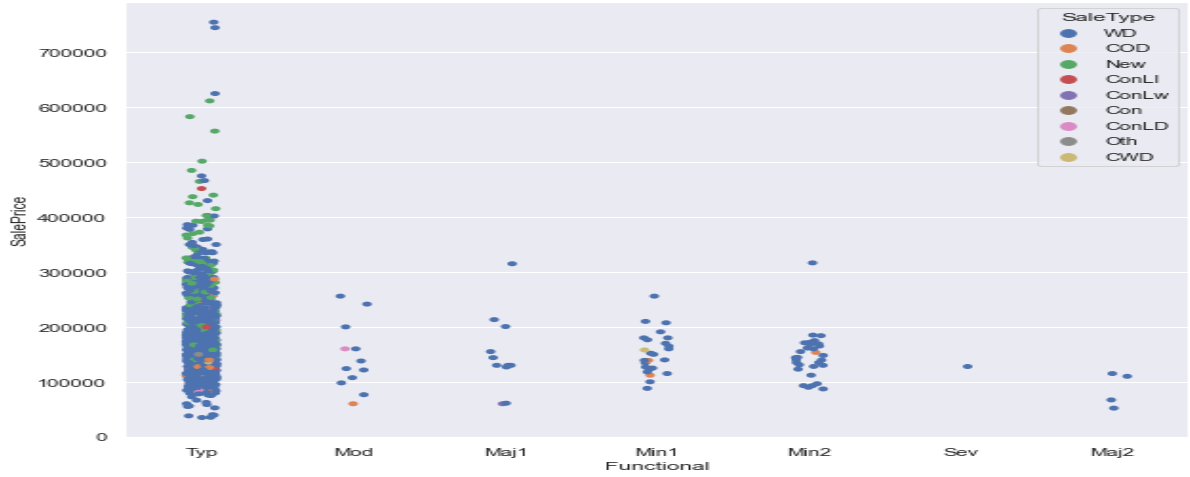

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
GasA
GasW
Floor
OthW
Wall
Grav
Heating

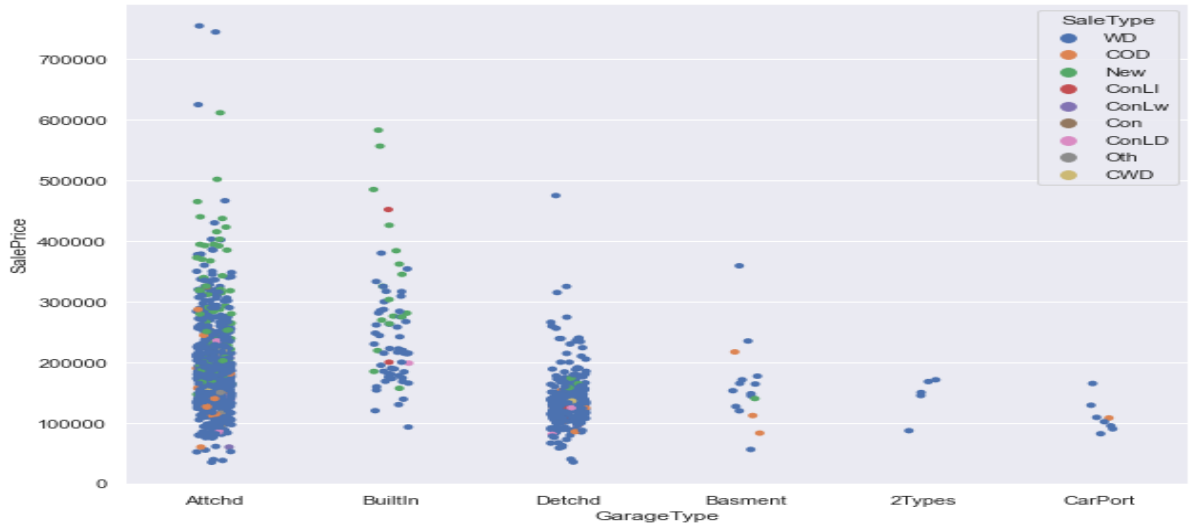

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Ex
Gd
Fa
Po
HeatingQC

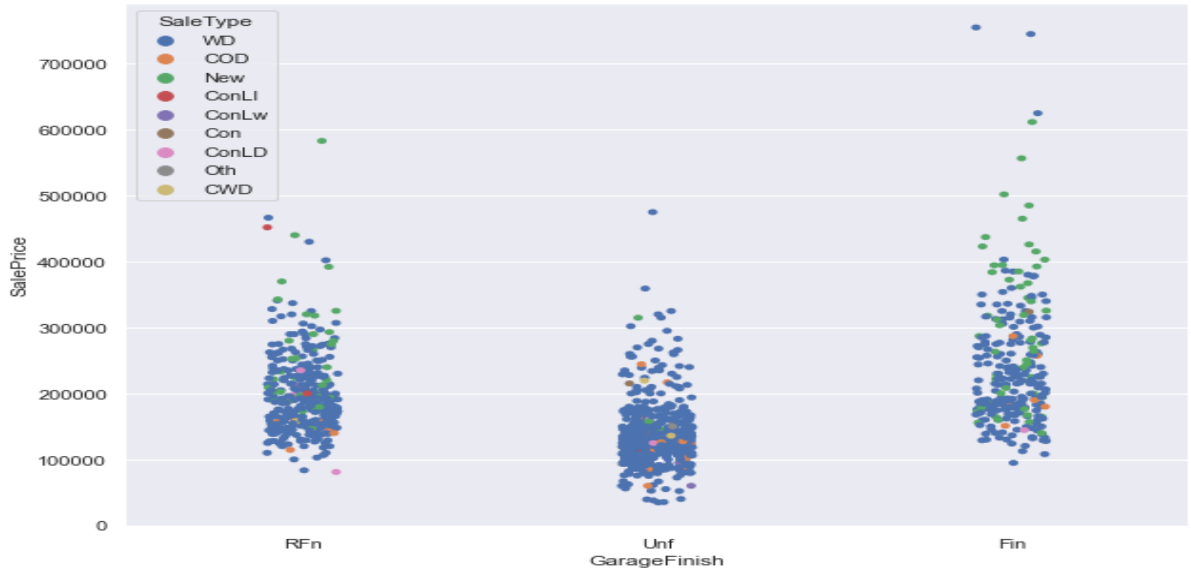

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Y
N
CentralAir

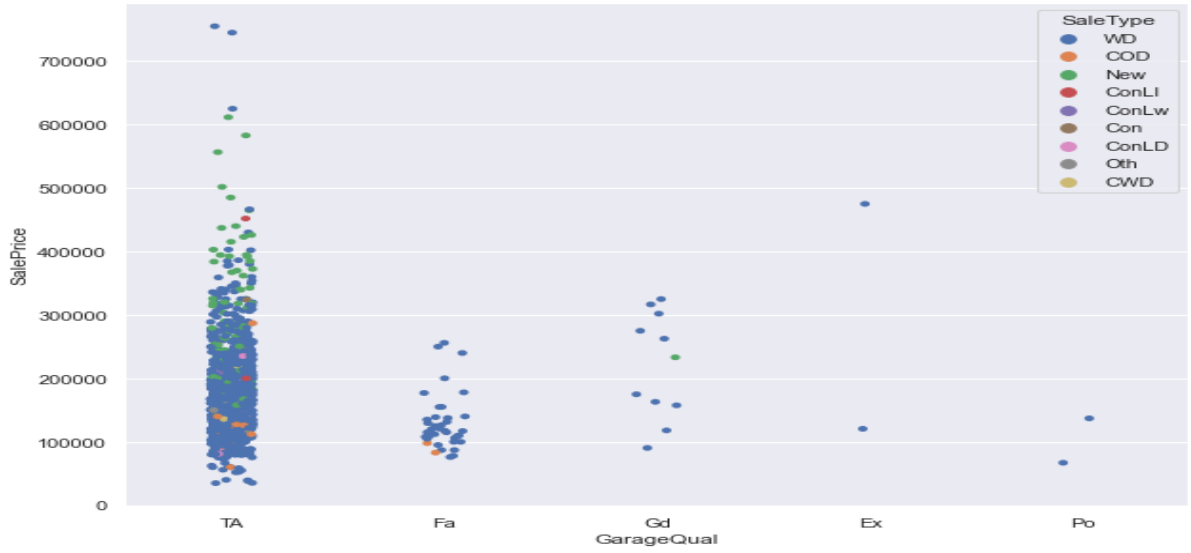

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
SBrkr
FuseA
FuseF
FuseP
Mix
Electrical

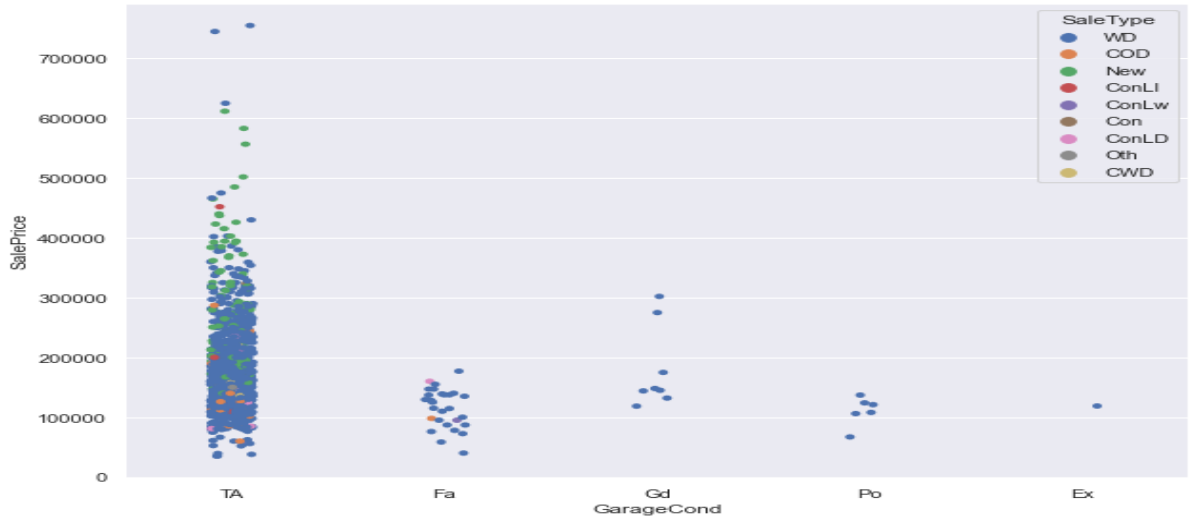

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Gd
Ex
Fa
KitchenQual

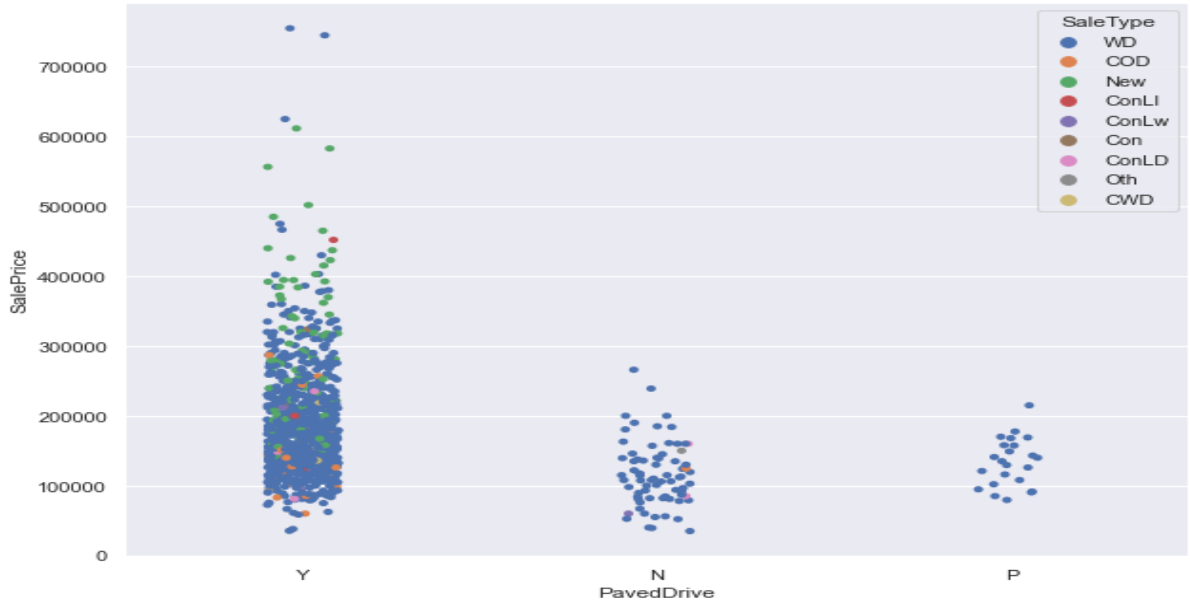

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Typ
Mod
Maj1
Min1
Min2
Sev
Maj2
Functional

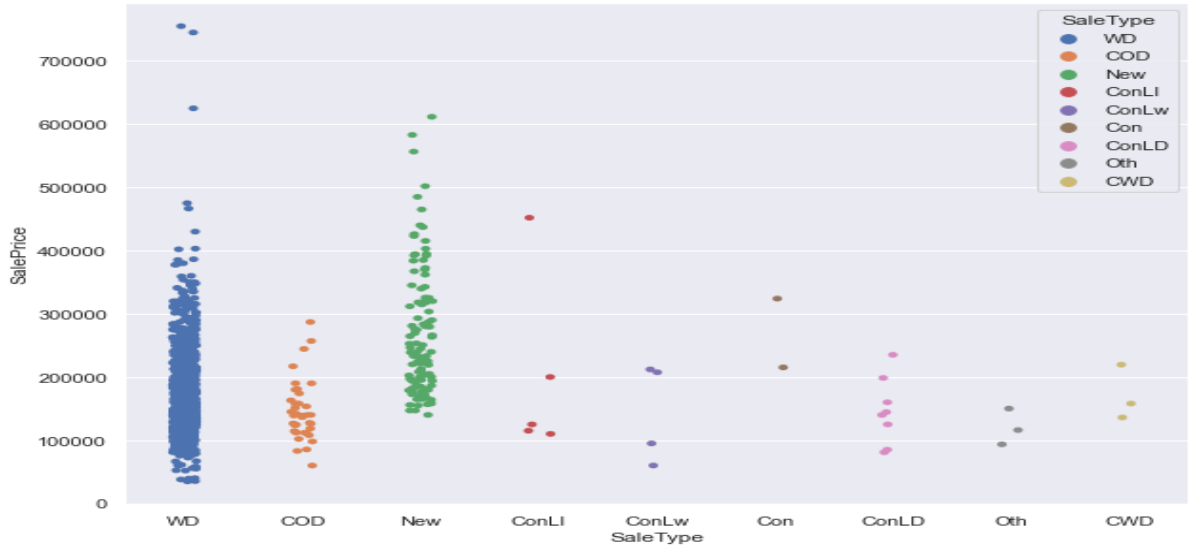

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Attchd
BuiltIn
Detchd
Basment
2Types
CarPort
GarageType

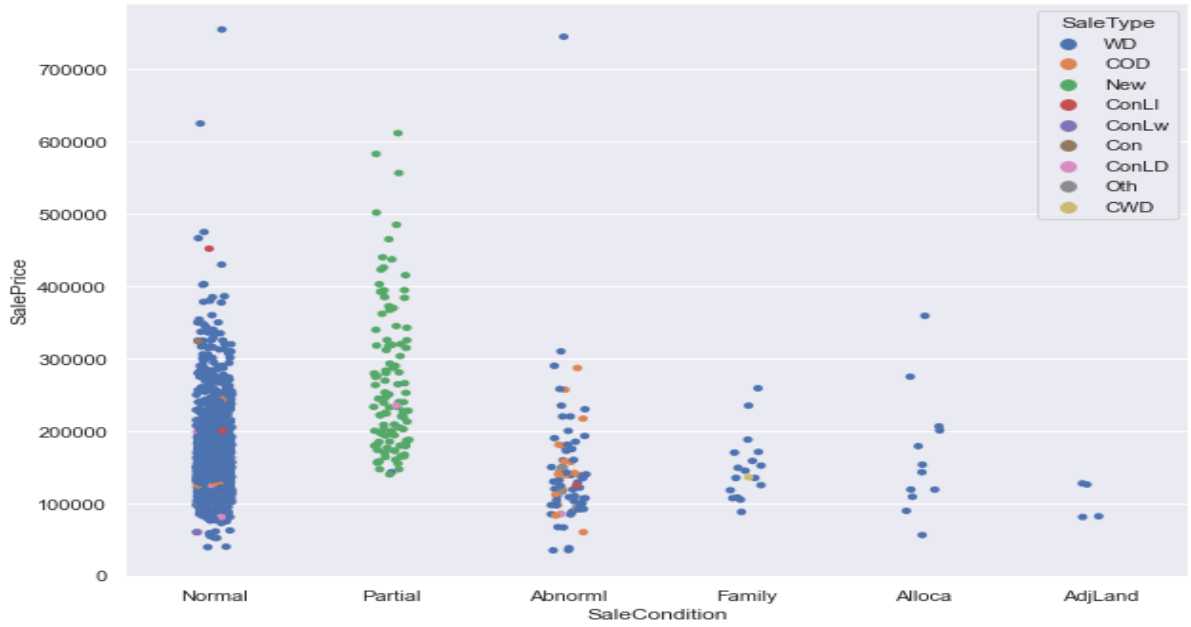

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
RFn
Unf
Fin
GarageFinish

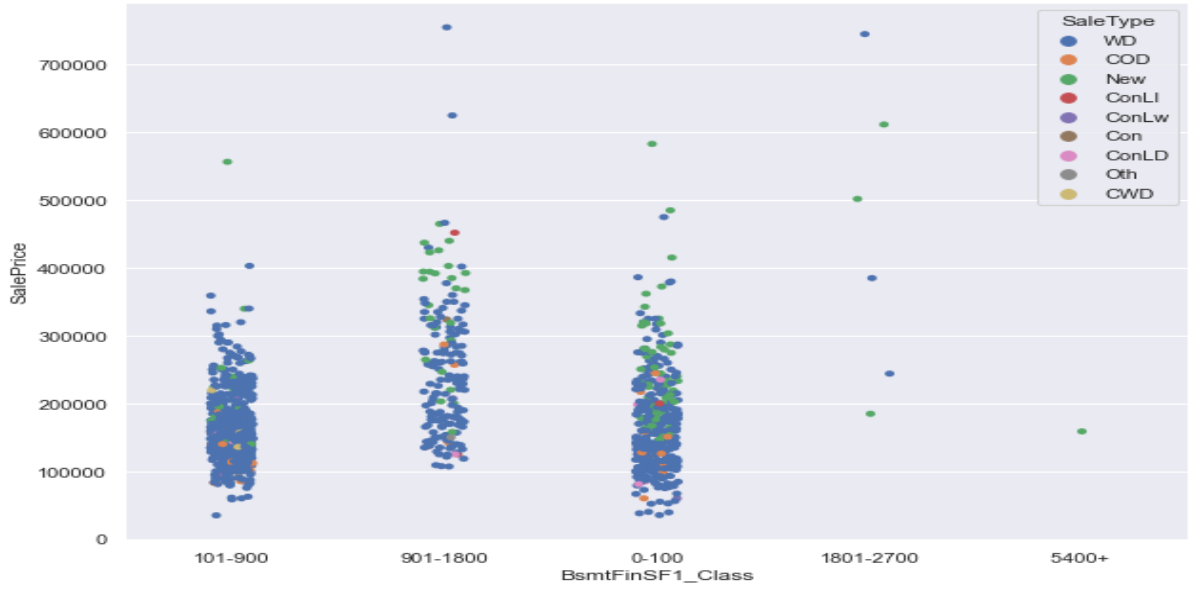

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
TA
Fa
Gd
Ex
Po
GarageQual

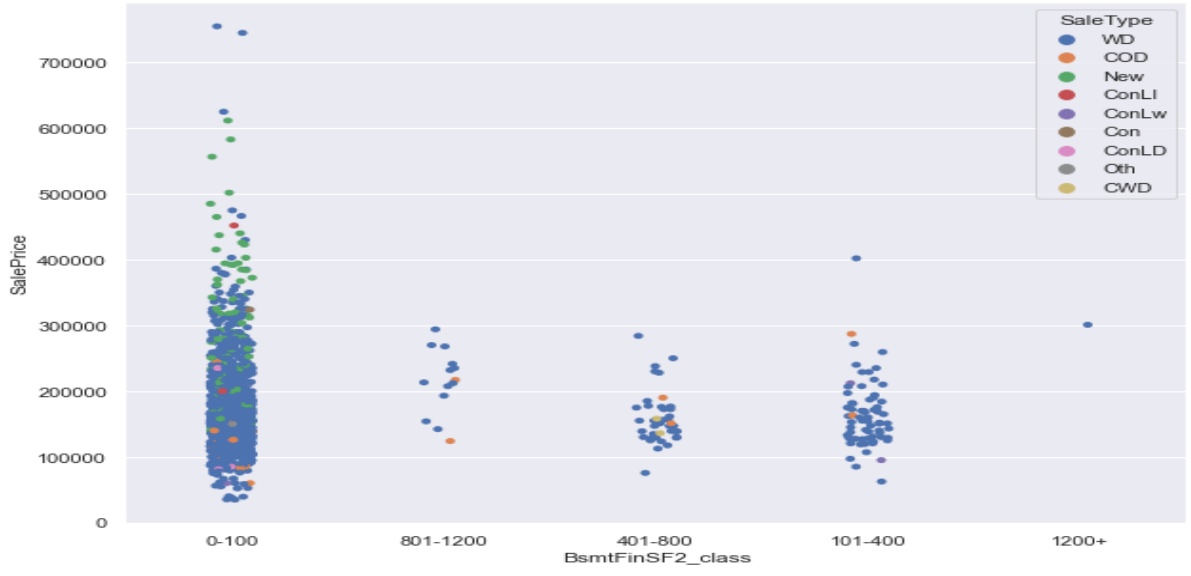

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
TA
Fa
Gd
Po
Ex
GarageCond

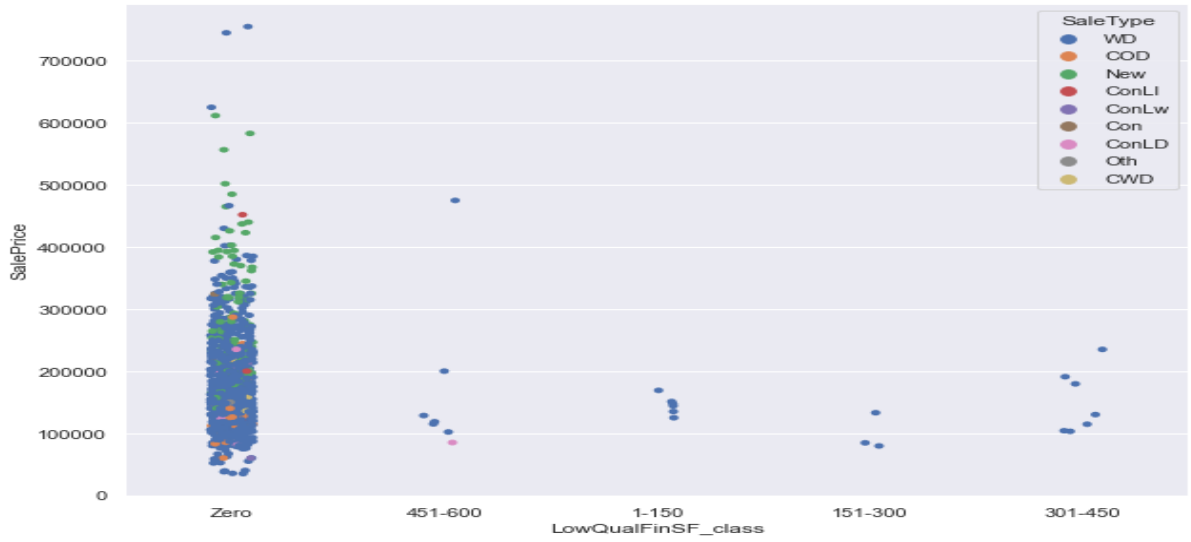

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Y
N
P
PavedDrive


SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SaleType

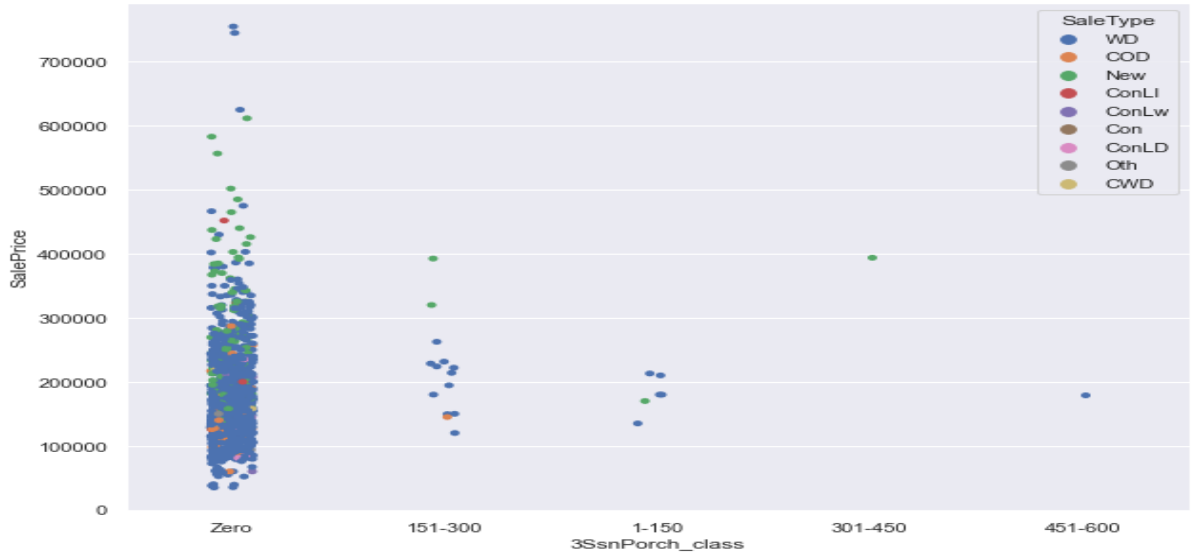

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Normal
Partial
Abnorml
Family
Alloca
AdjLand
SaleCondition

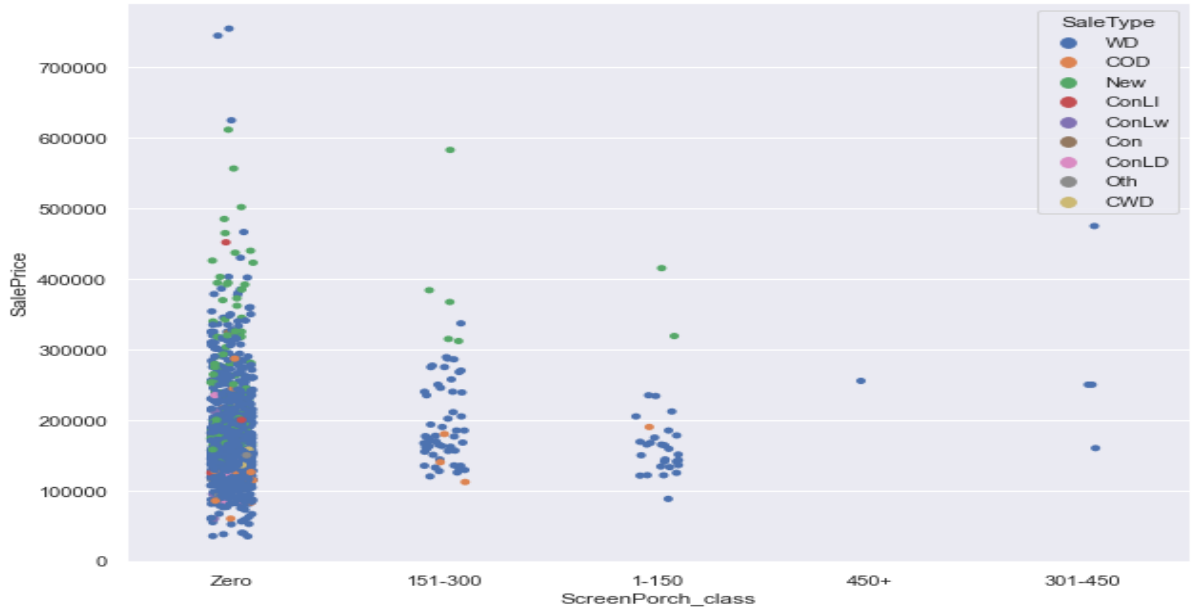

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
101-900
901-1800
0-100
1801-2700
5400+
BsmtFinSF1_Class

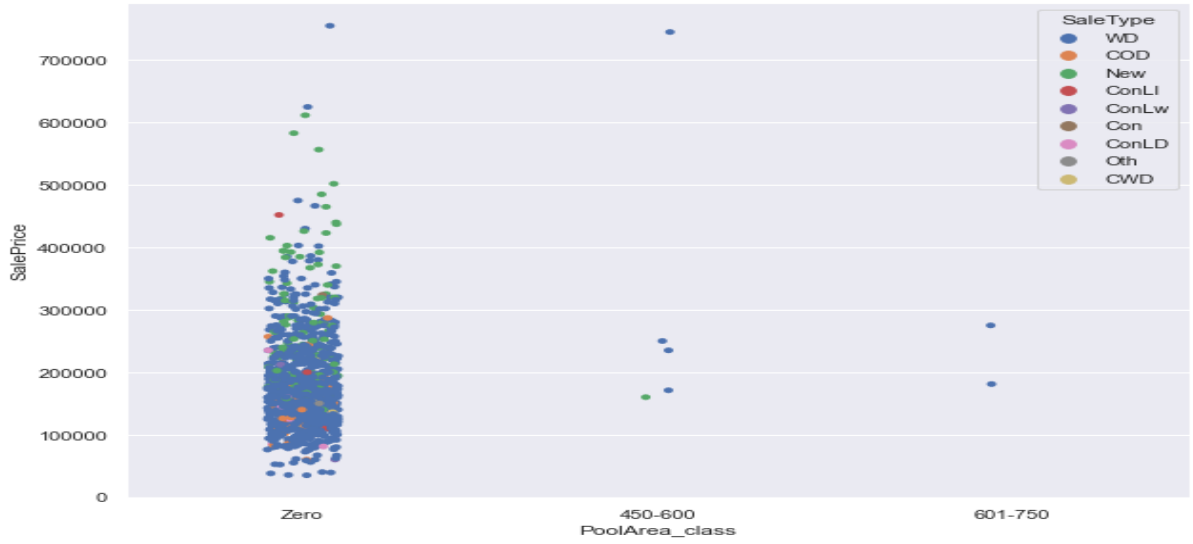

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0-100
801-1200
401-800
101-400
1200+
BsmtFinSF2_class

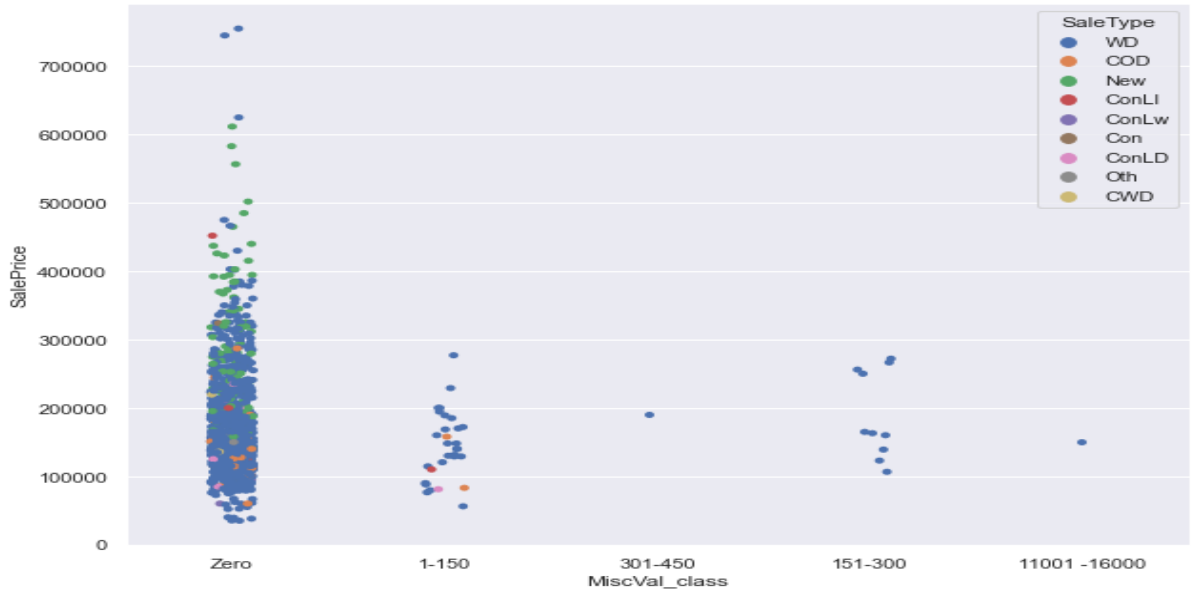

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Zero
451-600
1-150
151-300
301-450
LowQualFinSF_class


SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Zero
151-300
1-150
301-450
451-600
EnclosedPorch_class

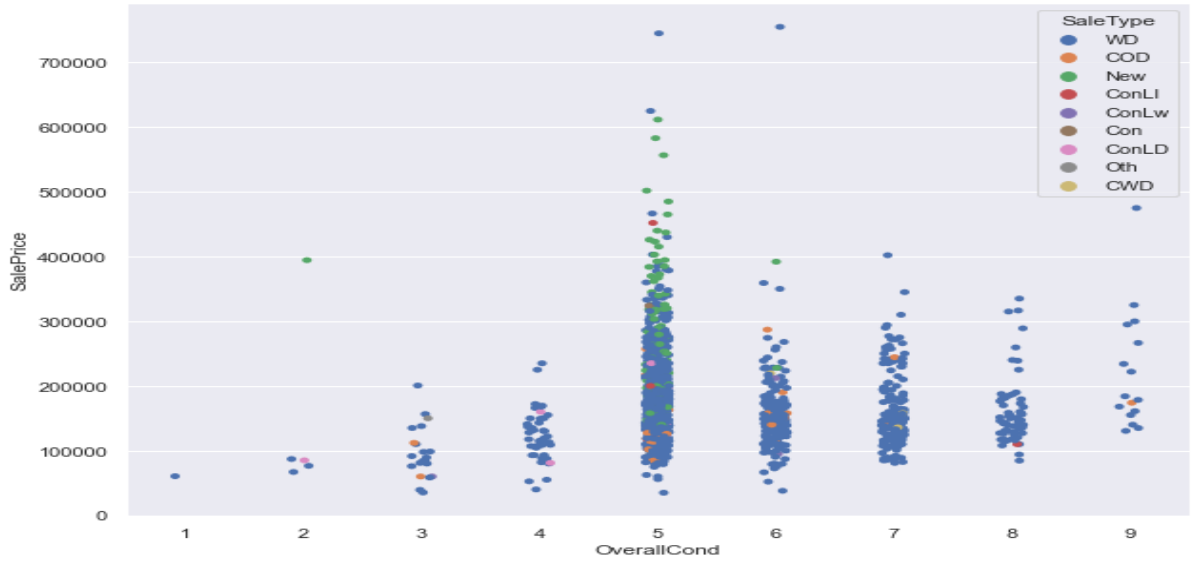

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Zero
151-300
1-150
301-450
451-600
3SsnPorch_class

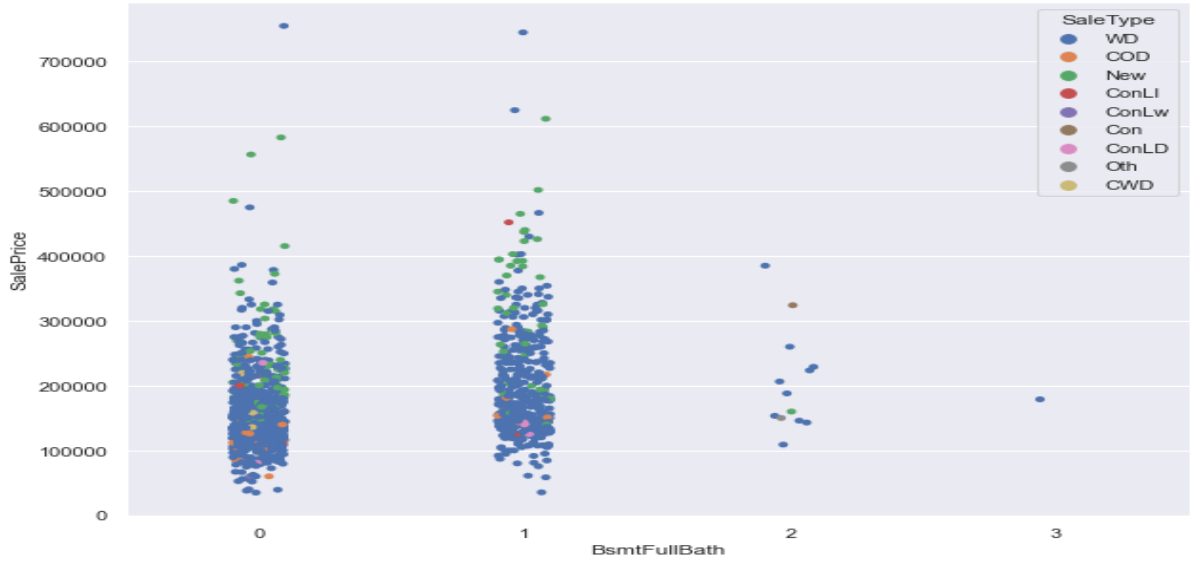

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Zero
151-300
1-150
450+
301-450
ScreenPorch_class

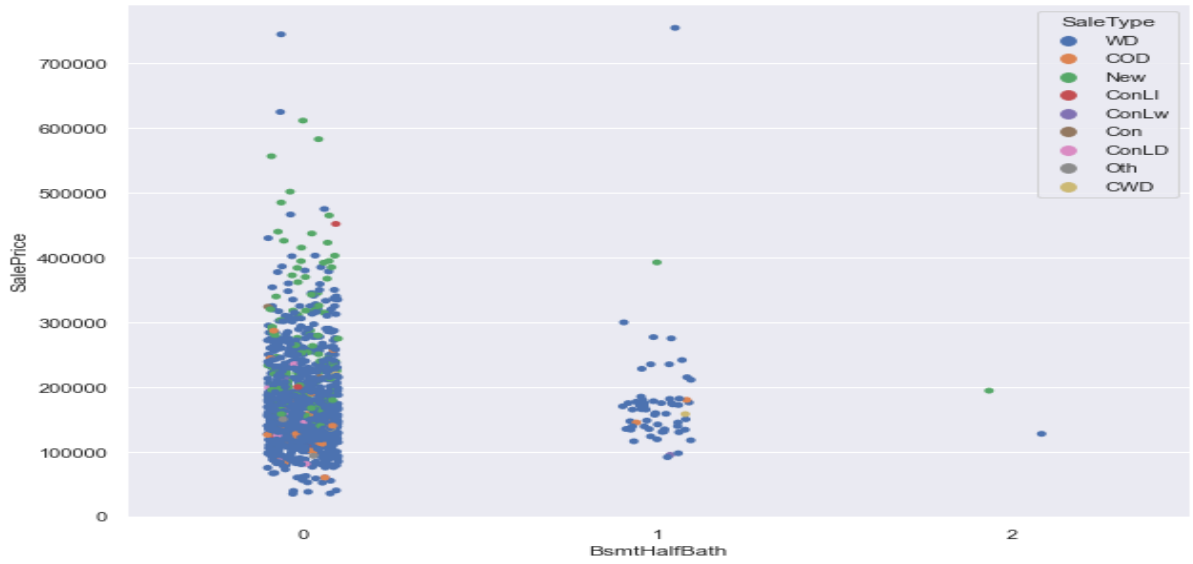

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Zero
450-600
601-750
PoolArea_class


SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Zero
1-150
301-450
151-300
11001 -16000
MiscVal_class

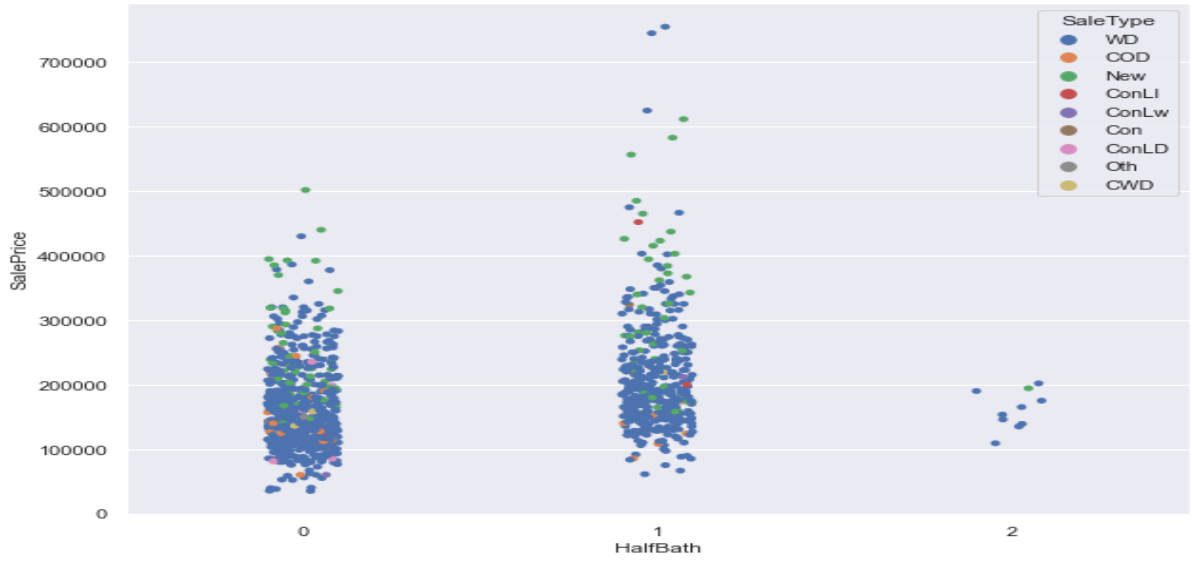

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
OverallQual

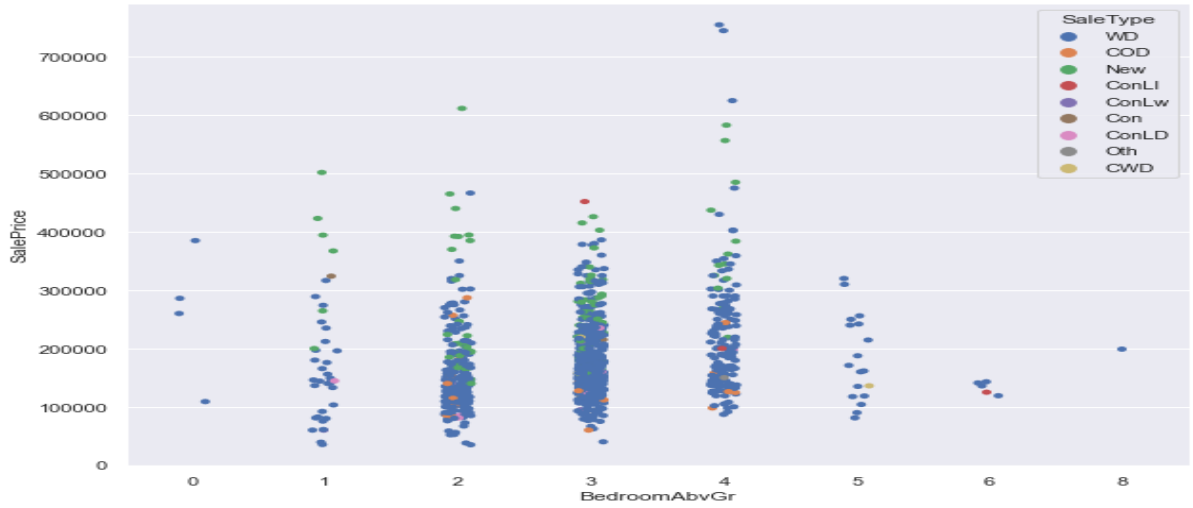

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
OverallCond


SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
BsmtFullBath

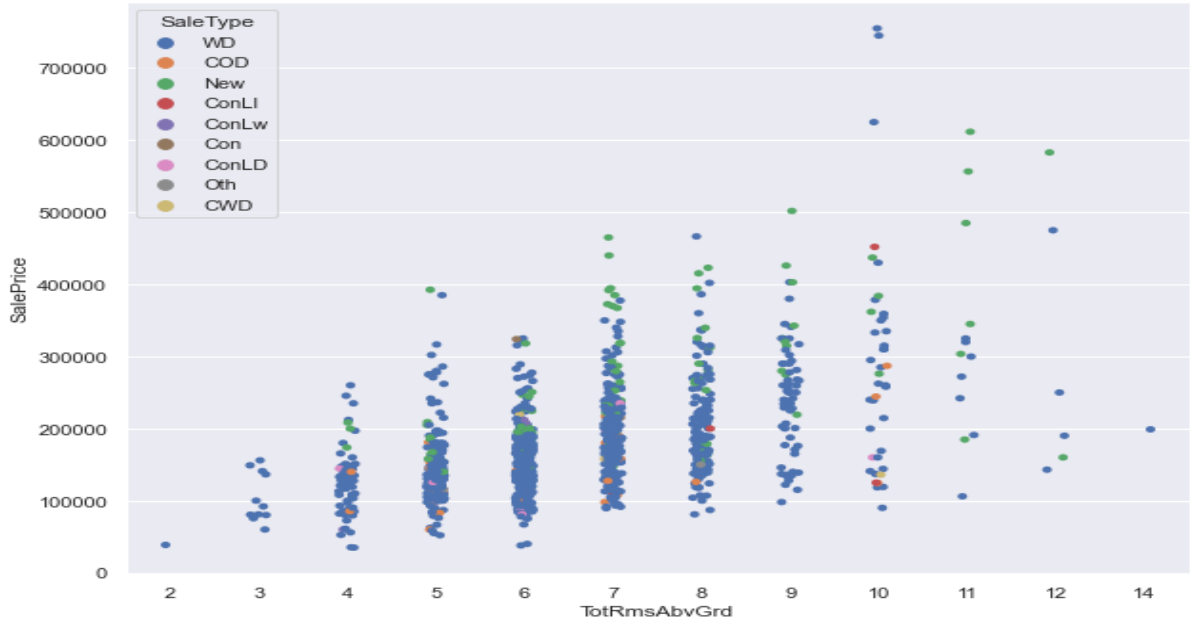

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
1
2
BsmtHalfBath

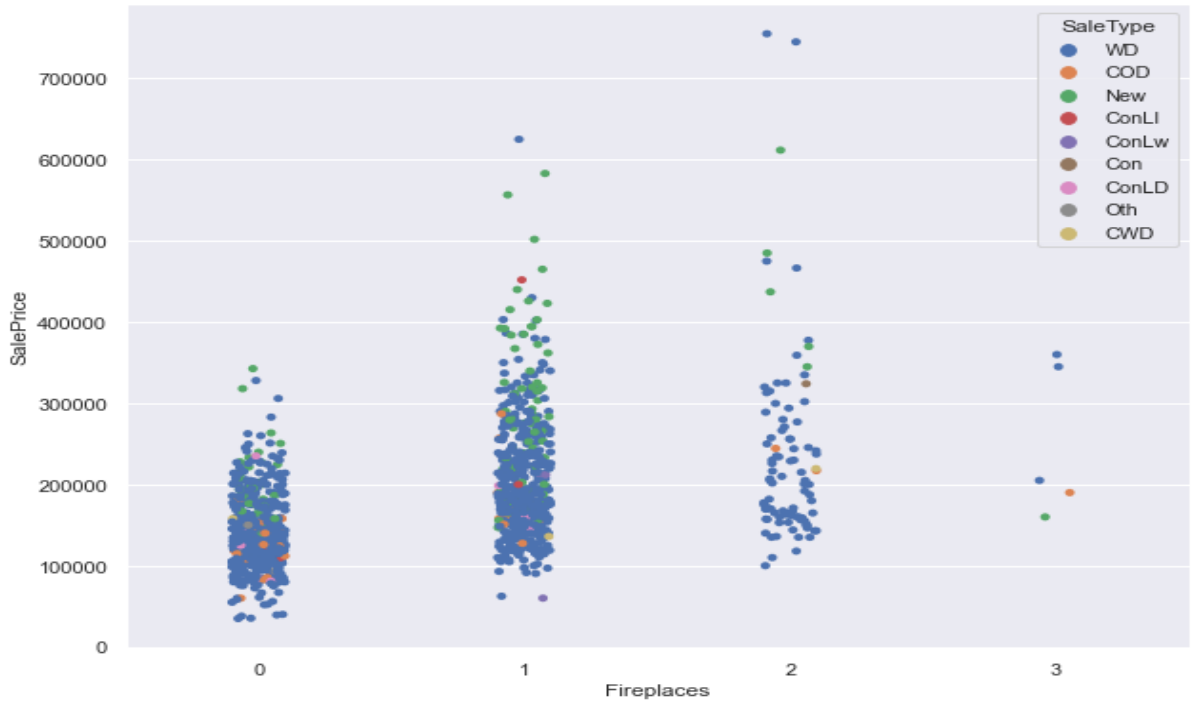

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
1
2
3
FullBath

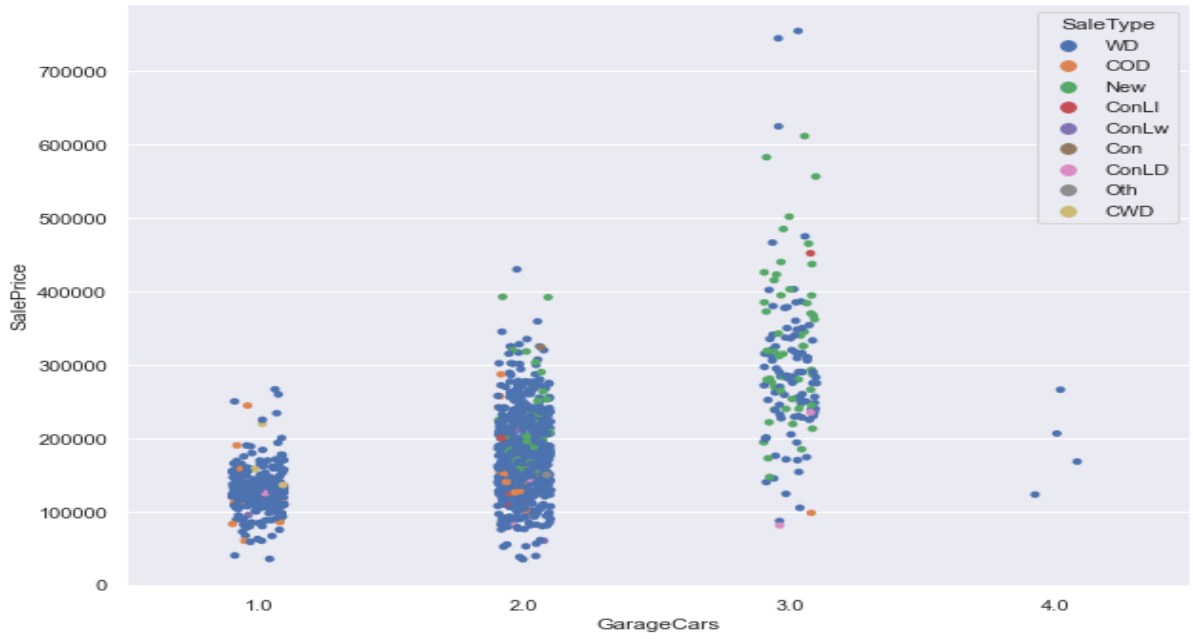

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
1
2
HalfBath


SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
1
2
3
4
5
6
8
BedroomAbvGr


SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
1
2
3
KitchenAbvGr

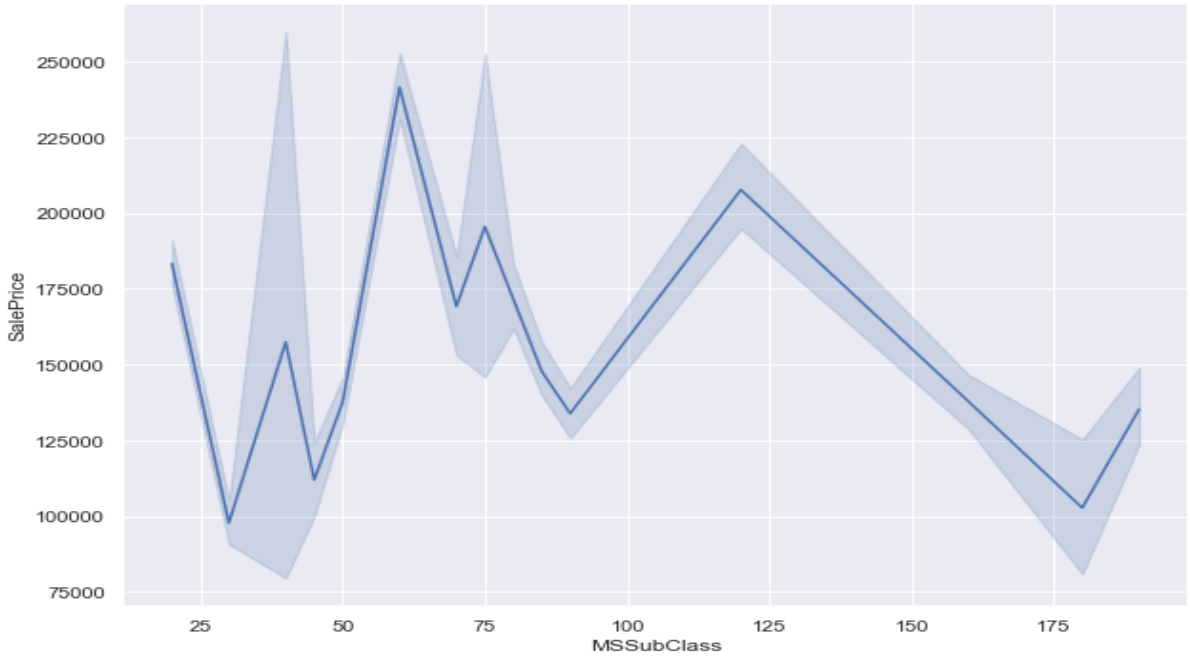

SaleType
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
2
3
4
5
6
7
8
9
10
11
12
14
TotRmsAbvGrd

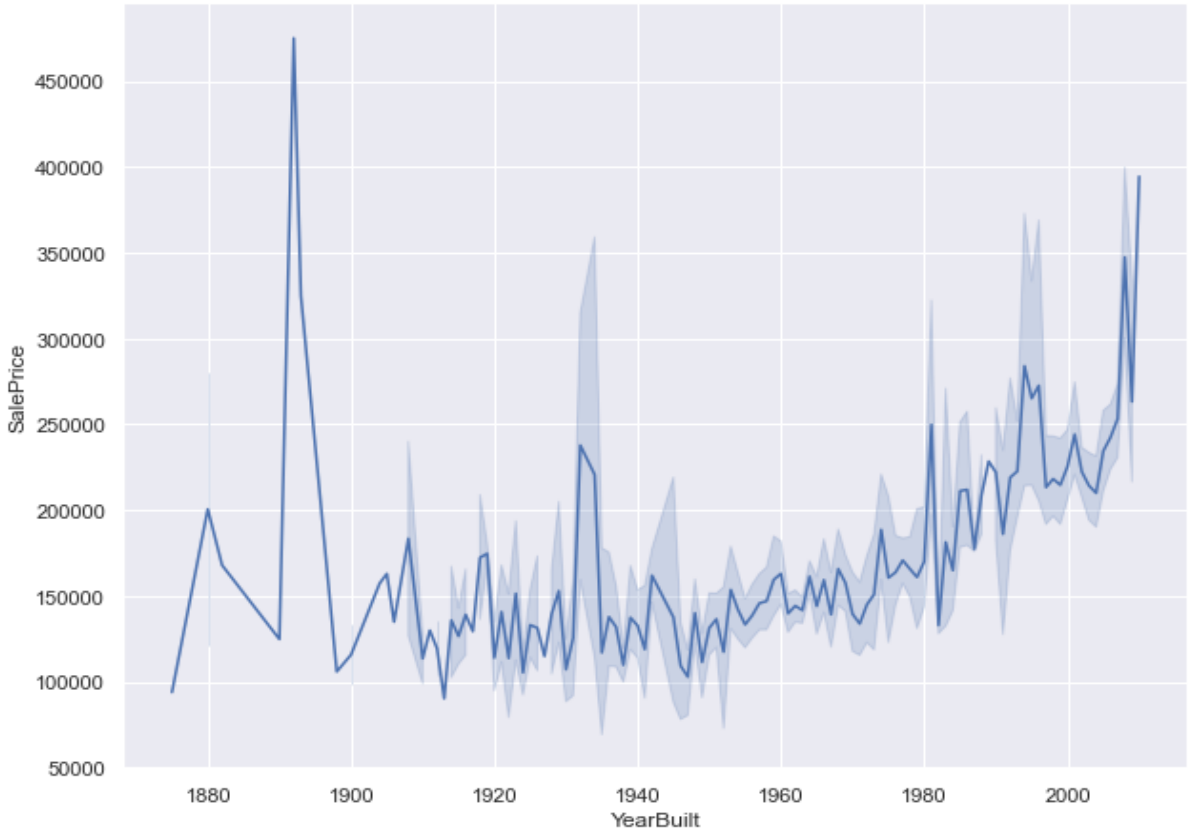


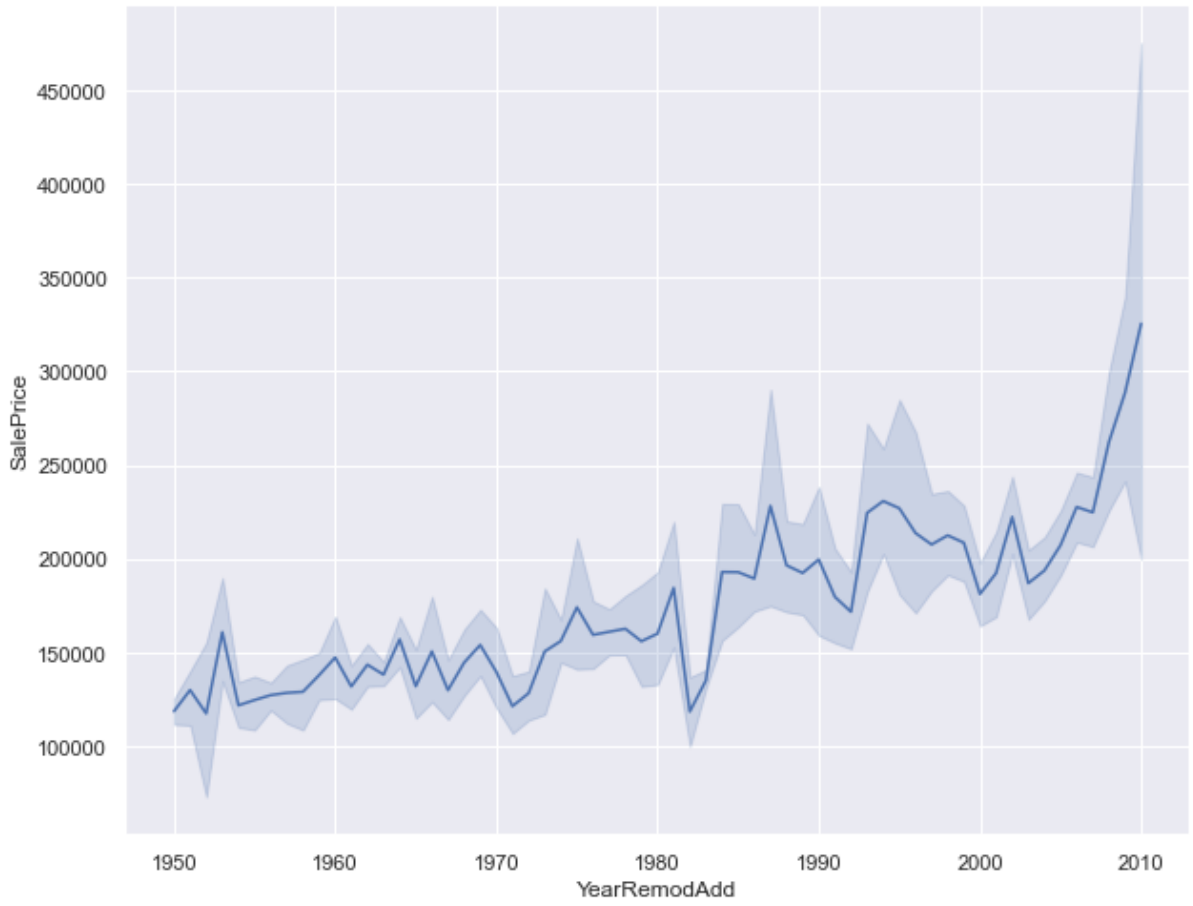


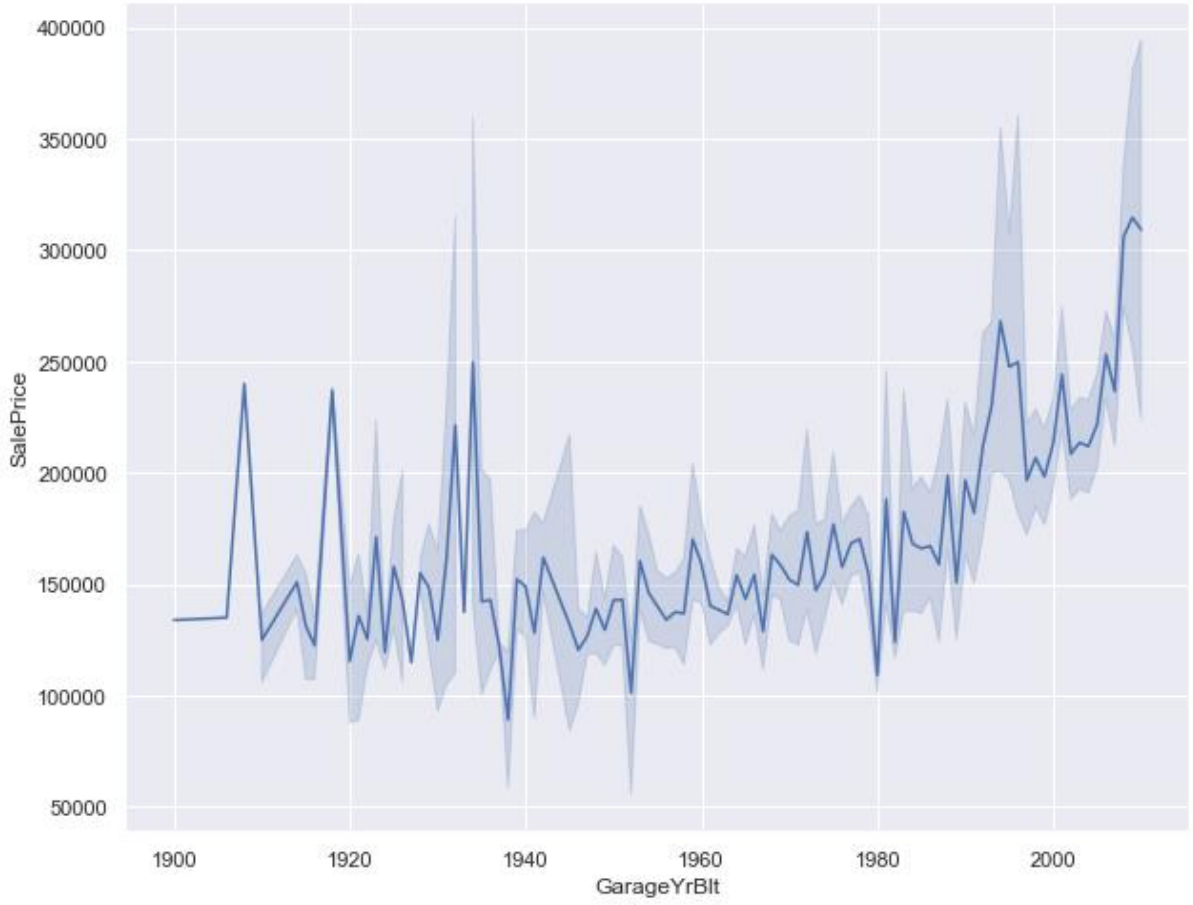
- Keys: Scatter plot is shown with sale price of property, by which can estimate about how column is impacting sale price

- **Line Plot:**

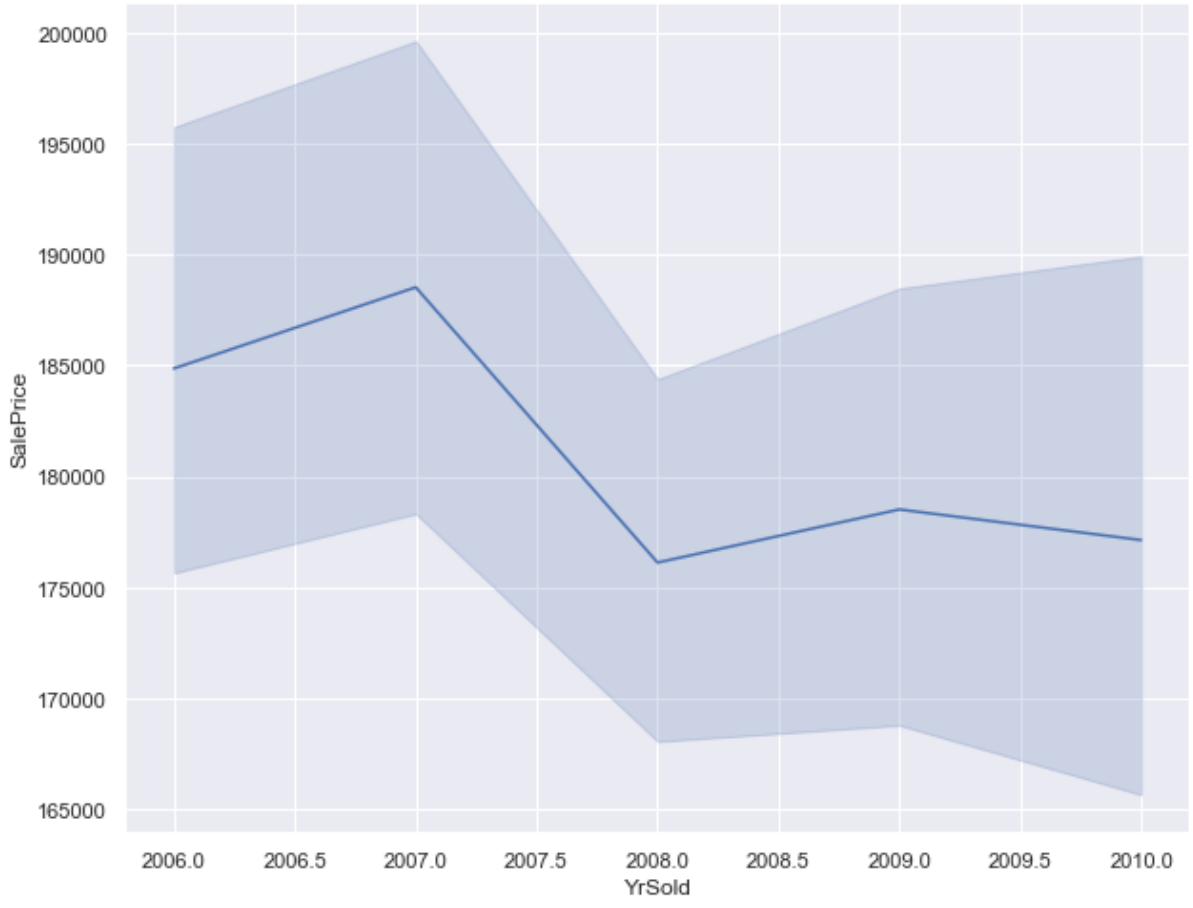

250000
225000
200000
175000
150000
125000
100000
75000
SalePrice
25
50
75
100
125
150
175
MSSubClass


450000
400000
350000
300000
250000
200000
150000
100000
50000
SalePrice
1880
1900
1920
1940
1960
1980
2000
YearBuilt


SalePrice
450000
400000
350000
300000
250000
200000
150000
100000
1950
1960
1970
1980
1990
2000
2010
YearRemodAdd


SalePrice
400000
350000
300000
250000
200000
150000
100000
50000
1900
1920
1940
1960
1980
2000
GarageYrBlt

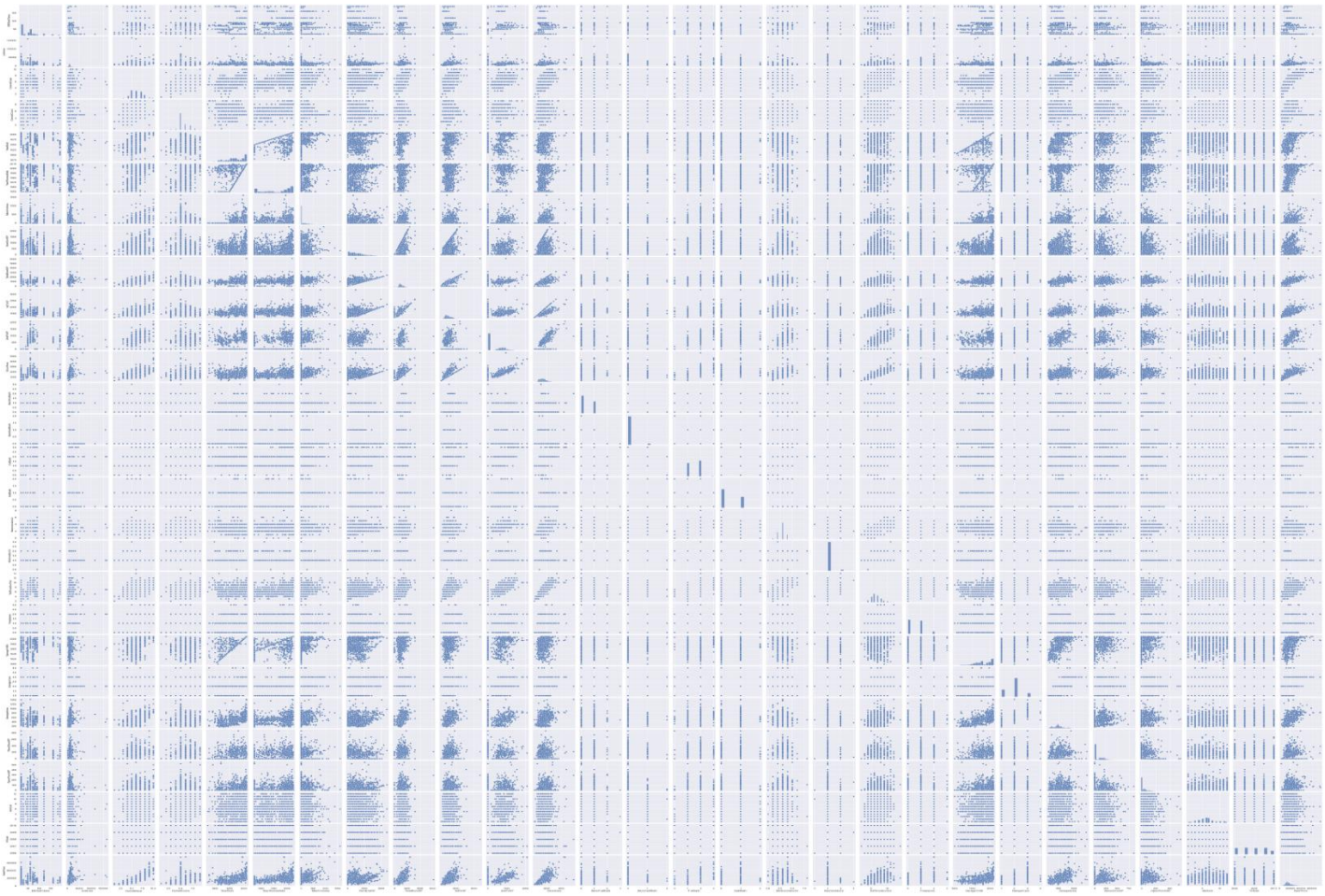

220000
200000
180000
160000
140000
SalePrice
2
4
6
8
10
12
MoSold


200000
195000
190000
185000
180000
175000
170000
165000
SalePrice
2006.0
2006.5
2007.0
2007.5
2008.0
2008.5
2009.0
2009.5
2010.0
YrSold

Key Points:

- Scatter plot is shown with sale price of property, by which can estimate about how column is impacting sale price
- Increase in yearRemodAdd and GarageYrbit, both are leading to increase sale price.
- Sale price is high as older "Yrsold " is.
- One can see the impact of these columns on saleprice

PAIR PLOT OF DATASET



- **Encoding of dataset:**

```
# Copy of train_df and test_df
encoded_train = train_df.copy()
encoded_test = test_df.copy()
```

```
from sklearn.preprocessing import OrdinalEncoder
oe = OrdinalEncoder()   # created instance of Ordinal Encoder
```

```
object_type = list(encoded_train.select_dtypes(include = 'object').keys())
len(object_type)
```

```
45
```

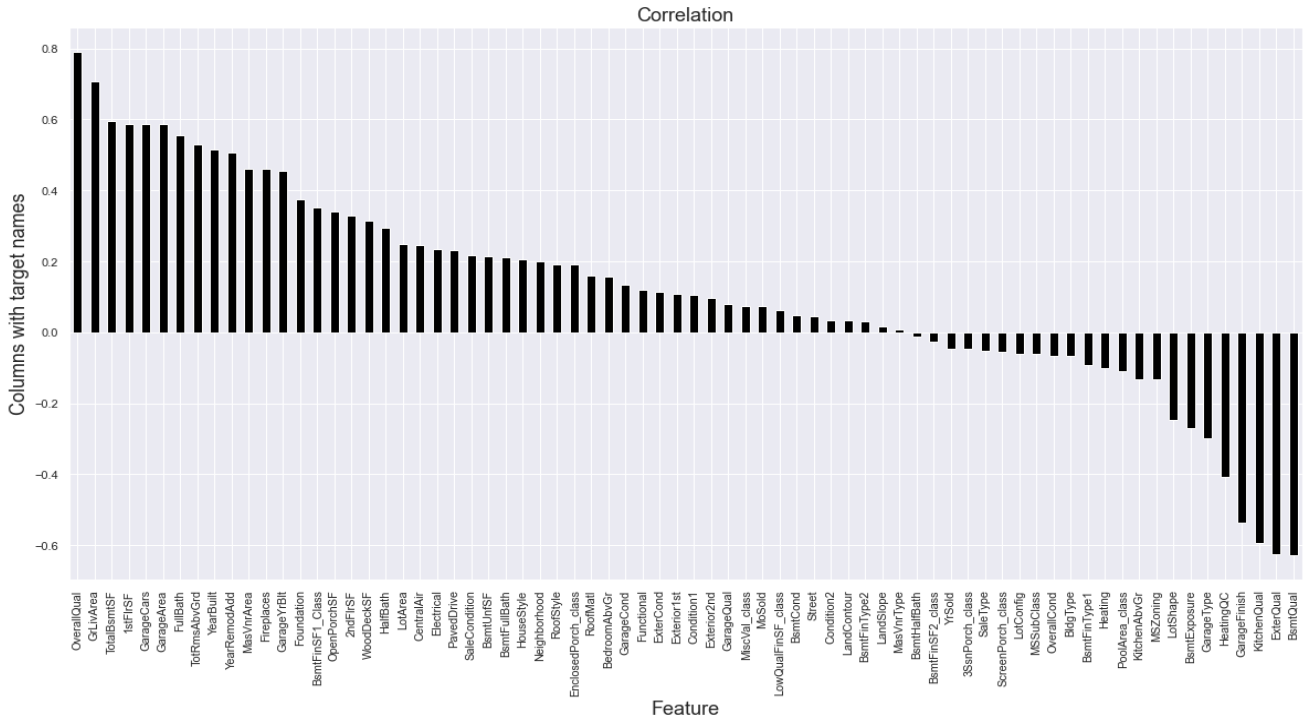
```
oe.fit(encoded_train[object_type])    # fit from train dataset
encoded_test[object_type] = oe.transform(encoded_test[object_type])
```

```
encoded_train[object_type] = oe.transform(encoded_train[object_type])
```

We have to encode feature column, therefore I have used ordinal encoding.

I have also checked model performance by applying OneHotEncoding but ordinal encoding is giving best result as compare to OneHotEncoding.

- **Data Inputs- Logic- Output Relationships**



"OverallQual" and "GLivArea" both are positively correlated with target variable.

"BsmtQual" and "ExterQual" both are negatively correlated with target variable

➢ State the set of assumptions

OUTLIERS:

Outliers are not removed from the dataset because it was giving high loss of data; therefore I have worked on existing dataset without removing outliers

1. Try zscore technique

```
from scipy.stats import zscore
```

```
z = np.abs(zscore(encoded_train) )
df_z = encoded_train[(z < 3).all(axis = 1)]
df_z.shape

# (465, 73)
```

```
(465, 73)
```

```
(encoded_train.shape[0] - df_z.shape[0] ) / encoded_train.shape[0]*100

# 60.18835616438356
```

```
60.18835616438356
```

2. IQR Technique

```
Q1 = encoded_train.quantile(0.25)
Q3 = encoded_train.quantile(0.75)
IQR = Q3 - Q1
```

```
df_IQR = encoded_train[~((encoded_train < (Q1 - 1.5*IQR) ) | (encoded_train > (Q3 + 1.5*IQR) )).any(axis = 1) ]
df_IQR.shape
# (84, 73)
```

```
(84, 73)
```

```
(encoded_train.shape[0] - df_IQR.shape[0] ) / encoded_train.shape[0]  * 100

# This method is giving 60% data loss, therefore we can not implement this method to the dataset
```

```
92.8082191780822
```

As both method of removing outliers, IQR and zscore is giving high loss data, therefore, I choose to work with existing data without removing outliers.

Seperating dataset into x1 and y1 form

```
x = encoded_train.drop(columns= ['SalePrice'])
y = encoded_train['SalePrice']

print('shape of x', x.shape)
print('shape of y', y.shape)
print('shape of test dataset: ', encoded_test.shape)
```

```
shape of x (1168, 72)
shape of y (1168,)
shape of test dataset:  (292, 72)
```

Separating Dataset into x and y form, where x is feature and y is target variable

Treatment of Skewness of columns:

```
from sklearn.preprocessing import PowerTransformer
pt = PowerTransformer(method='yeo-johnson')
from scipy import stats


pt.fit(x[num_col])
encoded_train[num_col] = pt.transform(encoded_train[num_col])
```

```
encoded_test[num_col] = pt.transform(encoded_test[num_col])
```

```
encoded_train[num_col].skew()
```

```
MSSubClass      0.064007
LotArea         0.032509
YearBuilt      -0.126641
YearRemodAdd   -0.225131
MasVnrArea      0.439526
BsmtUnfSF      -0.284390
TotalBsmtSF     0.286779
1stFlrSF       -0.002391
2ndFlrSF        0.280208
GrLivArea      -0.000054
GarageYrBlt    -0.136293
GarageArea      0.000291
WoodDeckSF      0.113026
OpenPorchSF    -0.002749
MoSold         -0.035838
YrSold          0.112893
dtype: float64
```

I have applied many skewness treatment on this dataset for removing skewness. After applying many skewness treatment this applied treatment is working best to remove skewness.

Multicollinearity

Multicollinearity is removed using variance inflation factor,

Here cal_vif is a my created function to calculate vif value of column, this function is created using variance inflation factor imported from statsmodels library

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

```
# function to calculate VIF
def cal_vif(data):
    vif = pd.DataFrame()
    vif['Columns Name'] = data.columns    # columns name
    vif['VIF'] = [variance_inflation_factor(data.values, i) for i in range(data.shape[1])]
    return (vif)
```

	Columns Name	VIF
0	MSSubClass	3.362308
1	LotArea	2.861014
2	YearBuilt	13909.329062
3	YearRemodAdd	16549.199285
4	MasVnrArea	1.765547
5	BsmtUnfSF	3.366975
6	TotalBsmtSF	25.011037
7	1stFlrSF	673.603256
8	2ndFlrSF	135.612987
9	GrLivArea	1093.190135
10	GarageYrBlt	25164.114865
11	GarageArea	17.948075
12	WoodDeckSF	1.844064
13	OpenPorchSF	1.834544
14	MoSold	6.651594
15	YrSold	14756.152411

	Columns Name	VIF
0	MSSubClass	3.334524
1	LotArea	2.835146
2	YearBuilt	8265.180342
3	YearRemodAdd	15535.781100
4	MasVnrArea	1.761914
5	BsmtUnfSF	3.323654
6	TotalBsmtSF	24.872871
7	1stFlrSF	651.901830
8	2ndFlrSF	131.568789
9	GrLivArea	1065.505588
10	GarageArea	14.339255
11	WoodDeckSF	1.822872
12	OpenPorchSF	1.829156
13	MoSold	6.642457
14	YrSold	12182.377295

	Columns Name	VIF
0	MSSubClass	3.329888
1	LotArea	2.832445
2	YearBuilt	6226.374354
3	MasVnrArea	1.743527
4	BsmtUnfSF	3.305136
5	TotalBsmtSF	24.854914
6	1stFlrSF	651.897880
7	2ndFlrSF	131.568237
8	GrLivArea	1065.169049
9	GarageArea	14.152404
10	WoodDeckSF	1.813654
11	OpenPorchSF	1.818040
12	MoSold	6.642341
13	YrSold	6014.344955

Step1: As we can see, GarageYrBlt column is showing high vif value, therefore I have remove it.

Step 2: YearRemodYr column is showing high vif, therefore this column is also removed

Step 3: Now, maximum column is in an acceptable range, and for those column which are showing high vif, we cannot delete much columns because it will lead to high loss of data.

Scaling:

Scaling of column, this step apply just before applying Machine learning of dataset, by followed this I have applied this Standard Scaling technique to the dataset

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()   # Instance of Standard Scaler
```

```
# Scalling Training dataset
x[x.columns] = ss.fit_transform(x[x.columns])
x.head()
```

	MSSubClass	MSZoning	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType	HouseStyle
0	1.508301	-0.021646	-0.620616	0.058621	-1.373107	0.318473	0.606420	-0.226126	0.142224	-0.037339	-0.023979	2.985495	-0.549930
1	-0.877042	-0.021646	0.600903	0.058621	-1.373107	0.318473	0.606420	3.295414	-0.024227	-0.037339	-0.023979	-0.403288	-0.549930
2	0.077095	-0.021646	-0.063075	0.058621	-1.373107	0.318473	-1.220661	-0.226126	0.475125	-0.037339	-0.023979	-0.403288	1.030838
3	-0.877042	-0.021646	0.141424	0.058621	-1.373107	0.318473	0.606420	-0.226126	0.308675	-0.037339	-0.023979	-0.403288	-0.549930
4	-0.877042	-0.021646	0.686902	0.058621	-1.373107	0.318473	-0.611634	-0.226126	0.308675	-0.037339	-0.023979	-0.403288	-0.549930

```
print(encoded_test.shape)
print(x.shape)
```

```
(292, 70)
(1168, 70)
```

```
y.shape
```

```
(1168,)
```

➢ Model/s Development and Evaluation

In Machine Learning, I have applied 3 base models of regression problem, after this I have also applied boosting techniques over this dataset and then selected one of them based on their performance.

After choosing best model, Hyper Parameter tuning is applied over it to increase the accuracy of model.

In final, finalize the final model for this dataset based on its performance.

In this machine learning process, I have created a function named "ML_Model" for smooth functioning of machine learning,

This function will give Training and testing data accuracy along with Mean Squared error and Mean Absolute Error and cv score on different-different cross fold values.

Libraries used in process of Machine Learning:

Machine learning

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import AdaBoostRegressor
import xgboost
from xgboost import XGBRegressor
```

creating instances for ML model and storing them into a list of models.

After storing instances into model list, apply ML_Model function with parameter model and dataset (x, y)

```
lr = LinearRegression()
dtr = DecisionTreeRegressor()
knn = KNeighborsRegressor()

models = [lr, dtr, knn]
ML_Model(models, x, y)
```

Here, ML_Model is my created function which will give performance of the model based on various performing factor accuracy, MSE, MAE and cv score.

 # Linear Regression Model Performance :

```
_____________________________________________________________________________
LinearRegression()  is giving best accurancy 0.8546463998110927 on random state of 37
Training accuracy is :  0.8326217719611156
Testing accuracy is : 0.8569051712981776
------------------------------------------------------------------------------
Mean Squared Error:  719212202.5057057
Mean Absolute Error:  20269.256900642973
------------------------------------------------------------------------------
Cross value score
cv score 0.692381605033251 at 2 cross fold
cv score 0.7412202227321302 at 3 cross fold
cv score 0.756604352733875 at 4 cross fold
cv score 0.7449757887406958 at 5 cross fold
cv score 0.7585327316079106 at 6 cross fold
cv score 0.7293223736940927 at 7 cross fold
------------------------------------------------------------------------------
```

- This algorithm is not giving over fitted model, but it is having high difference between cv and test data accuracy

 # DecisionTreeRegressor Model Performance:

```
_____________________________________________________________________________
DecisionTreeRegressor()  is giving best accurancy 0.8247455396270994 on random state of 12
Training accuracy is :  1.0
Testing accuracy is : 0.7861177776969679
------------------------------------------------------------------------------
Mean Squared Error:  1633946569.980057
Mean Absolute Error:  25167.05128205128
------------------------------------------------------------------------------
Cross value score
cv score 0.5002159106685531 at 2 cross fold
cv score 0.6858012782406403 at 3 cross fold
cv score 0.653282831632314 at 4 cross fold
cv score 0.7187454884073861 at 5 cross fold
cv score 0.7326260069099656 at 6 cross fold
cv score 0.6979664901626247 at 7 cross fold
------------------------------------------------------------------------------
```

- By applying this model, getting overfitted model, as its traing accuracy is higher than testing accuracy
- Training Accuracy > Testing Accuracy

 # KNeighborsRegressor Model Performance

```
_____________________________________________________________________________
KNeighborsRegressor()  is giving best accurancy 0.7213838331827729 on random state of 34
Training accuracy is :  0.8383771779034621
Testing accuracy is : 0.7855729216403198
------------------------------------------------------------------------------
Mean Squared Error:  1028584891.2813675
Mean Absolute Error:  21261.302564102563
------------------------------------------------------------------------------
Cross value score
cv score 0.7238722970882102 at 2 cross fold
cv score 0.7378697405780926 at 3 cross fold
cv score 0.7296852724791547 at 4 cross fold
cv score 0.7319152856571005 at 5 cross fold
cv score 0.7303304025735947 at 6 cross fold
cv score 0.727269195081118 at 7 cross fold
------------------------------------------------------------------------------
```

- It is having less accuracy and also having high difference between cv score value and accuracy value

 # Applied Boosting Techniques:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import AdaBoostRegressor
rfr = RandomForestRegressor()
gbr = GradientBoostingRegressor()
abr = AdaBoostRegressor()

models = [rfr, gbr, abr]
ML_Model(models, x, y)
```

- Bagging and Boosting techniques are applied on the dataset because normal algorithm was not giving enough result or good result

 # RandomForestRegressor Model Performance:

```
_____________________________________________________________________________
RandomForestRegressor()  is giving best accurancy 0.8602367294100308 on random state of 19
Training accuracy is :  0.9739880325466944
Testing accuracy is : 0.8829821893882084
------------------------------------------------------------------------------
Mean Squared Error:  698728252.8761048
Mean Absolute Error:  17877.57965811966
------------------------------------------------------------------------------
Cross value score
cv score 0.8086910312176738 at 2 cross fold
cv score 0.8588107218647809 at 3 cross fold
cv score 0.8390309744118734 at 4 cross fold
cv score 0.8414786265655934 at 5 cross fold
cv score 0.8517135755216628 at 6 cross fold
cv score 0.828347543734054 at 7 cross fold
------------------------------------------------------------------------------
```

- Having difference in accuracy of training and testing dataset, not much but having,

 # AdaBoostRegressor Model Performance:

```
_____________________________________________________________________________
AdaBoostRegressor()  is giving best accurancy 0.8078649544014966 on random state of 39
Training accuracy is :  0.8849071806084483
Testing accuracy is : 0.8230374575266938
------------------------------------------------------------------------------
Mean Squared Error:  968630008.4141606
Mean Absolute Error:  22007.300167520945
------------------------------------------------------------------------------
Cross value score
cv score 0.7634368146856891 at 2 cross fold
cv score 0.7935140877758146 at 3 cross fold
cv score 0.7845199169851335 at 4 cross fold
cv score 0.792675764114988 at 5 cross fold
cv score 0.7828340585920563 at 6 cross fold
cv score 0.7842546755765206 at 7 cross fold
------------------------------------------------------------------------------
```

- ML_Model function is created for smooth operation of machine learning
- Giving Good results

 # XGBoostRegressor Model Performance:

```
Training accuracy is :  0.9999741751479383
Testing accuracy is : 0.8839930186140819
--------------------------------------------------------------------------------
Mean Squared Error:  718522764.5732626
Mean Absolute Error:  17632.69255252849
--------------------------------------------------------------------------------
Cross value score
cv score 0.8353491059844554 at 2 cross fold
cv score 0.8511448028030243 at 3 cross fold
cv score 0.8206761353698495 at 4 cross fold
cv score 0.8388832744206691 at 5 cross fold
cv score 0.8594095268629013 at 6 cross fold
cv score 0.8313952730129088 at 7 cross fold
--------------------------------------------------------------------------------
```

- This is model is giving good resule but giving over fitted model

 # GradientBoostingRegressor Model Performance:

```
_____________________________________________________________________________
GradientBoostingRegressor()  is giving best accurancy 0.8917976009767302 on random state of 37
Training accuracy is :  0.9726752237962414
Testing accuracy is : 0.9049629969315385
--------------------------------------------------------------------------------
Mean Squared Error:  477667662.18253416
Mean Absolute Error:  16008.743091677125
--------------------------------------------------------------------------------
Cross value score
cv score 0.8388940013932447 at 2 cross fold
cv score 0.8750609850293148 at 3 cross fold
cv score 0.8513533945379466 at 4 cross fold
cv score 0.8611798083967706 at 5 cross fold
cv score 0.8711118848754861 at 6 cross fold
cv score 0.8503287678925674 at 7 cross fold
--------------------------------------------------------------------------------
```

- This model is giving good result as it not having much difference in accuracy of training and testing dataset
- And also not having high cv value and accuracy difference

Models	Training Accuracy	Testing Accuracy	CV Score	Difference

# RandomForestRegressor	0.973988	0.882982	0.85881	0.024
# GradientBoosting Regressor	0.972675	0.904962	0.87506	0.030
# AdaBoostRegressor	0.884907	0.823037	0.793514	0.030
# XGBRegressor	0.999974	0.883993	0.859409	0.025

- We can observed that least difference between testing accuracy and cv, found in xgboost but it is giving overfitted model and
- GradientBoost and AdaBoostRegressor is both are giving same cv and accuracy difference, and we select GradientBoosting because it is showing more accuracy as compare to adaboost

Ensemble Techniques:

Based on model performance, we can see, GradientBoosting is performing best as compare to other ML Models, Its cv value is also very close to accuracy of model.

Hence I had applied hyper parameter tuning for this Model to increase the accuracy of model

```
: from sklearn.model_selection import GridSearchCV
  x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.30, random_state = 37)

: parameter = {'alpha' : [0.9, 0.09, 0.1],
               'learning_rate' : [0.1, 0.01],
               'max_depth' : [ 3, 4, 5],
               'min_samples_leaf' : [1, 2, 3],
              'min_samples_split': [2,3,4],
              'n_estimators': [100, 50, 10]}

: gcv = GridSearchCV(estimator = GradientBoostingRegressor(), param_grid = parameter, cv = 7)
  gcv.fit(x_train, y_train)

: GridSearchCV(cv=7, estimator=GradientBoostingRegressor(),
               param_grid={'alpha': [0.9, 0.09, 0.1],
                           'learning_rate': [0.1, 0.01], 'max_depth': [3, 4, 5],
                           'min_samples_leaf': [1, 2, 3],
                           'min_samples_split': [2, 3, 4],
                           'n_estimators': [100, 50, 10]})
```

```
gcv.best_params_
```

```
{'alpha': 0.09,
 'learning_rate': 0.1,
 'max_depth': 3,
 'min_samples_leaf': 1,
 'min_samples_split': 3,
 'n_estimators': 100}
```

```
# 'parameter' these are the parameters which have to be applied on given model and then have to choose best
# one.
# GridSearchCV will help in that
```

```
gbr = GradientBoostingRegressor(alpha = 0.9, learning_rate = 0.1, max_depth = 3, min_samples_leaf=1,
                                min_samples_split=3 , n_estimators = 100)

models = [gbr]
ML_Model(models, x, y)
```

```
_____________________________________________________________________________
GradientBoostingRegressor(min_samples_split=3)  is giving best accurancy 0.8925363168251541 on random state o
Training accuracy is :  0.9709969915980627
Testing accuracy is : 0.9012624616581131
-----------------------------------------------------------------------------
Mean Squared Error:  558346098.2162528
Mean Absolute Error:  16042.284810874853
-----------------------------------------------------------------------------
Cross value score
cv score 0.8349190207390833 at 2 cross fold
cv score 0.8766740857081136 at 3 cross fold
cv score 0.8531874797993466 at 4 cross fold
cv score 0.8616931484565242 at 5 cross fold
cv score 0.8710084859163643 at 6 cross fold
cv score 0.8507762906963834 at 7 cross fold
-----------------------------------------------------------------------------
```


After applying hyper parameter tuning, we can see, cv and accuracy has little decreased, which is good for model

Final Model:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.30, random_state = 6)

final_model = GradientBoostingRegressor(alpha = 0.9, learning_rate = 0.1, max_depth = 3, min_samples_leaf=1,
                                        min_samples_split=3 , n_estimators = 100)
final_model.fit(x_train, y_train)
final_pred = final_model.predict(x_test)
final_pred_train = final_model.predict(x_train)

train_accuracy = r2_score(y_train, final_pred_train )
test_accuracy = r2_score(y_test, final_pred )
print('Training accuracy: ', train_accuracy)
print('Testing accuracy: ', test_accuracy)
print('_____________________________________________')
print('Mean squared error: ', mean_squared_error(y_test, final_pred ) )
print('Mean absolute error: ', mean_absolute_error(y_test, final_pred) )


cv_score = cross_val_score(final_model, x, y, cv = 3 ).mean()
print('cv score',  cv_score ,'at', i, 'cross fold')
```

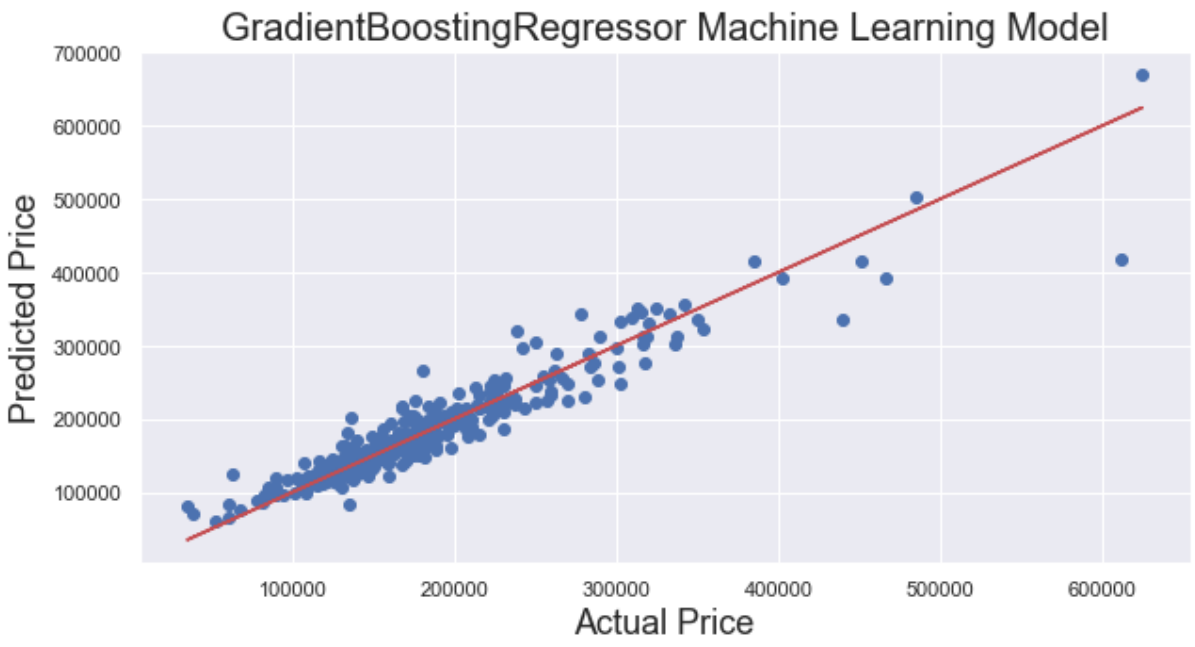
```
Training accuracy:  0.9709969915980627
Testing accuracy:  0.9006411168987434

--------------------------------------------------
Mean squared error:  561859710.4438533
Mean absolute error:  16020.68811082674
cv score 0.8773090209451722 at MiscVal_class cross fold
```

Graph for Model Performance:

```
plt.figure(figsize = (10, 5))
plt.scatter(x = y_test, y = final_pred, color = 'b')
plt.plot(y_test, y_test, color = 'r')
plt.xlabel('Actual Price', fontsize= 18 )
plt.ylabel('Predicted Price', fontsize = 18)
plt.title('GradientBoostingRegressor Machine Learning Model', fontsize = 20)
```

In below mentioned chart, we can see how model is performing,

As our model is having 90% accuracy based on this, its chart is visible to us



Deploying & Loading Model:

Deploy Model

```
import pickle
filename = 'house_price_predictor.pkl'
pickle.dump(final_model, open(filename, 'wb'))
```

Loading model

```
load_model =  pickle.load(open('house_price_predictor.pkl', 'rb'))
result = load_model.score(x_test, y_test)
print(result)
```

```
0.9006411168987434
```

Conclusion:

```
predicted = np.array(load_model.predict(x_test))
original = np.array(y_test)
# convert columns in to np.array
```

```
# print(predicted.shape)
# print(original.shape)
# print(x_test.shape)
# print(y_test.shape)
```

```
conclusion = pd.DataFrame({'Actual Price': original, 'Predicted Price': predicted},
                          index = range(len(original)))
```

```
conclusion['Predicted Price']= conclusion['Predicted Price'].apply(lambda x: round(x, 1))
```

```
conclusion.sample(10)
```

	Actual Price	Predicted Price
348	278000	343380.4
70	320000	329704.3
286	625000	670691.5
317	175900	225094.2
62	197900	209751.9
196	302000	332550.9
229	130000	137226.3
142	128950	126746.8
135	117000	128221.3
312	196000	179084.5

➢ Hardware and Software Requirements and Tools Used

All used libraries:

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import AdaBoostRegressor
import xgboost
from xgboost import XGBRegressor
```

Pandas: This library used for dataframe operations

Numpy: This library gives statistical computation for smooth functioning

Matplotlib: Used for visualization

Seaborn: This library is also used for visualization

Sklearn: This library having so many machine learning module and we can import them from this library

Pickle: This is used for deploying the model

Imblearn: This library is import to get SMOTE technique for balance the data

Scipy: It is import to perform outlier removing technique using zscore

Warning: To avoid unwanted warning shows in the output

I am giving this requirement and tool used, based on my laptop configuration.

Operating System: **Window 11**
RAM: **8 GB**

Processor: **i5 10th Generation**
Software: **Jupyter Notebook,**

➢ Observations from the whole problem.

i) Dataset was having null value.

ii) Impute values according to column relation with each other

iii) Multicollinearity was not present in the dataset

iv) Created group of range for few columns

v) Skewness is removing by using PowerTransformer

➢ Learning Outcomes of the Study in respect of Data Science

My learnings: - the power of visualization is helpful for the understanding the data and graphical representation, its help me to understand that what data is trying to say,

Data cleaning is one of the most important step to remove missing value or null value. Perform mean or median imputation based on its effect on the column

Various algorithms I have used in this dataset and to get best result and save that model. The best algorithm is GradientBoostingClassifier.

The challenge I faced while working on this project, one record was wrong updated in the Exterior1st and Exterior2nd columns which was created issue when I was encoding the data for further machine learning process.

➢ Limitations of this work and Scope for Future Work

Need to train this same model with data in Lacs or more than that, then only it would be become best model and it can perform in best way.