HOUSING PRICE PREDICTION PROJECT

Submitted by:
Rahul Sharma

ACKNOWLEDGMENT

I would like to thank Flip Robo Technologies for providing me with the opportunity to work on this project “House Price Prediction”. I would like to thank to my SME Mr. Shubham Yadav who has constantly guided and helped me with his suggestions and instructions during this project.

Then I would like to thank my parents who have been helpful in the various phases of this project.

Some of the reference sources are as follows:

- Scikitlearn.org
- Towarddatascience.com
- Analytics Vidhya
- Sciencedirect.com
- Stackoverflow

TABLE OF CONTENTS

INTRODUCTION

BUSINESS PROBLEM FRAMING

This is a real estate problem where a US based housing company named Surprise Housing has decided to invest in Australian Market. Their agenda is to buy houses in Australia at prices below their actual value in the market and sell them at high prices to gain profit. To do this this company uses data analytics to decide in which property they must invest.

Company has collected the data of previously sold houses in Australia and with the help of this data they want to know to the value of prospective properties to decide whether it will suitable to invest in the properties or not.

To know the value of properties company has provided data to us to do data analysis and to extract the information of attributes which are important to predict the price of the houses. They want a machine learning model which can predict the price of houses and also the significance of each important attribute in house prediction i.e, how and to what intensity each variable impacts the price of the house.

CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM

In real estate the value of property usually increases with time as seen in many countries. One of the causes for this is due to rising population.

The value of property also depends on the proximity of the property, its size its neighbourhood and audience for which the property is subjected to be sold. For example if audience is mainly concerned of commercial purpose. Then the property which is located in densely populated area will be sold very fast and at high prices compared to the one located at remote place. Similarly if audience is concerned only on living place then property with less dense area having large area with all services will be sold at higher prices.

The company is looking at prospective properties to buy houses to enter the market. We 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.

MOTIVATION FOR THE PROBLEM UNDERTAKEN

To understand real world problems where Machine Learning and Data Analysis can be applied to help organizations in various domains to make better decisions with the help of which they can gain profit or can be escaped from any loss which otherwise could be possible without the study of data .One of such domain is Real Estate.

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.

ANALYTICAL PROBLEM FRAMING

MATHEMATICAL/ ANALYTICAL MODELING OF THE PROBLEM

In this project we have performed various mathematical and statistical analysis such as we checked description or statistical summary of the data using describe, checked correlation using corr and also visualized it using heatmap. Then we have used Z-Score to plot outliers and remove them.

In [18]:
```
# Let's check the statistical summary of our dataset

housing_train.describe()
```

Out[18]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2
count	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000
mean	724.136130	56.767979	70.807363	10484.749144	6.104452	5.595890	1970.930651	1984.758562	101.696918	444.726027	46.647260
std	416.159877	41.940650	22.440317	8957.442311	1.390153	1.124343	30.145255	20.785185	182.218483	462.664785	163.520016
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1875.000000	1950.000000	0.000000	0.000000	0.000000
25%	360.500000	20.000000	60.000000	7621.500000	5.000000	5.000000	1954.000000	1966.000000	0.000000	0.000000	0.000000
50%	714.500000	50.000000	70.000000	9522.500000	6.000000	5.000000	1972.000000	1993.000000	0.000000	385.500000	0.000000
75%	1079.500000	70.000000	79.250000	11515.500000	7.000000	6.000000	2000.000000	2004.000000	160.000000	714.500000	0.000000
max	1460.000000	190.000000	313.000000	164660.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	1474.000000

From this statistical analysis we make some of the interpretations that:

- In the columns Id, MSSubclass, LotArea, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfsF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtFullBath, HalfBath, TotRmsAbvGrd, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, Miscval, salePrice mean is considerably greater than median so the columns are positively skewed.
- In the columns FullBath, BedroomAbvGr, Fireplaces, Garagecars, GarageArea, YrSold Median is greater than mean so the columns are negatively skewed.
- In the columns Id, MSSubClass, LotFrontage, LotArea, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtHalfBath, BedroomAbvGr, ToRmsAbvGrd, GarageArea, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, MiscVal, SalePrice there is considerable difference between the 75 percentile and maximum so outliers are present.

DATA SOURCES AND THEIR FORMATS

The variable features of this problem statement are as :

MSSubClass: Identifies the type of dwelling involved in the sale

MSZoning: Identifies the general zoning classification of the sale

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access to property

Alley: Type of alley access to property

LotShape: General shape of property

LandContour: Flatness of the property

Utilities: Type of utilities available

LotConfig: Lot configuration

LandSlope: Slope of property

Neighborhood: Physical locations within Ames city limits

Condition1: Proximity to various conditions

Condition2: Proximity to various conditions (if more than one is present)

BldgType: Type of dwelling

HouseStyle: Style of dwelling

OverallQual: Rates the overall material and finish of the house

OverallCond: Rates the overall condition of the house

YearBuilt: Original construction date

YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)

RoofStyle: Type of roof

RoofMatl: Roof material

Exterior1st: Exterior covering on house

Exterior2nd: Exterior covering on house (if more than one material)

MasVnrType: Masonry veneer type

MasVnrArea: Masonry veneer area in square feet

ExterQual: Evaluates the quality of the material on the exterior

ExterCond: Evaluates the present condition of the material on the exterior

Foundation: Type of foundation

BsmtQual: Evaluates the height of the basement

BsmtCond: Evaluates the general condition of the basement

BsmtExposure: Refers to walkout or garden level walls

BsmtFinType1: Rating of basement finished area

BsmtFinSF1: Type 1 finished square feet

BsmtFinType2: Rating of basement finished area (if multiple types)

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

HeatingQC: Heating quality and condition

CentralAir: Central air conditioning

Electrical: Electrical system

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

Bedroom: Bedrooms above grade (does NOT include basement bedrooms)

Kitchen: Kitchens above grade

KitchenQual: Kitchen quality

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

Functional: Home functionality (Assume typical unless deductions are warranted)

Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

GarageType: Garage location

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

GarageCond: Garage condition

PavedDrive: Paved driveway

WoodDeckSF: Wood deck area in square feet

OpenPorchSF: Open porch area in square feet

EnclosedPorch: Enclosed porch area in square feet

3SsnPorch: Three season porch area in square feet

ScreenPorch: Screen porch area in square feet

PoolArea: Pool area in square feet

PoolQC: Pool quality

Fence: Fence quality

MiscFeature: Miscellaneous feature not covered in other categories

MiscVal: $Value of miscellaneous feature

MoSold: Month Sold (MM)

YrSold: Year Sold (YYYY)

SaleType: Type of sale

SaleCondition: Condition of sale

```
#checking the information oof the dataset
HP_train.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
```

```
#checking the datatypes
HP_train.dtypes
```

```
Id                 int64
MSSubClass         int64
MSZoning          object
LotFrontage      float64
LotArea            int64
                  ...
MoSold             int64
YrSold             int64
SaleType          object
SaleCondition     object
SalePrice          int64
Length: 81, dtype: object
```

DATA PREPROCESSING DONE

After loading all the required libraries we loaded the data into our jupyter notebook.

```
#importing the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from scipy import stats

from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression,Lasso,Ridge,Elastic
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV,cross_val_score
from sklearn.model_selection import GridSearchCV
```

```python
#loading the dataset
HP_train = pd.read_csv('train.csv')
HP_train
```

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

1168 rows × 81 columns

```python
#dimension of the dataset
HP_train.shape
```

(1168, 81)

```
There are : 1168 Rows and 81 Columns present in the dataset.
```

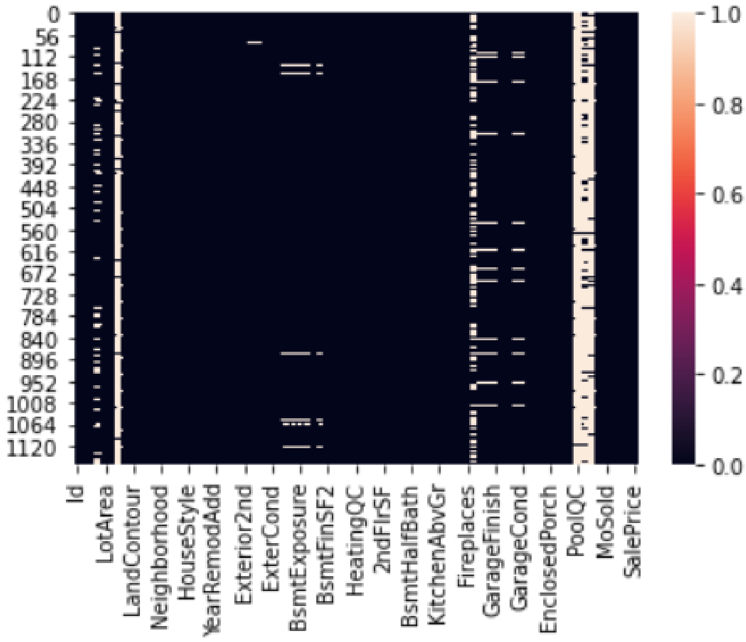
Feature Engineering has been used for cleaning of the data. Some unused columns have been deleted and even some columns have been bifurcated which was used in the prediction. We first done data cleaning. We first looked at missing values in columns of the dataset to see how much data was missing.

```
#checking for the null values in the dataset
HP_train[HP_train.columns[HP_train.isnull().any()]].isnull().sum()
```

```
LotFrontage      214
Alley           1091
MasVnrType         7
MasVnrArea         7
BsmtQual          30
BsmtCond          30
BsmtExposure      31
BsmtFinType1      30
BsmtFinType2      31
FireplaceQu      551
GarageType        64
GarageYrBlt       64
GarageFinish      64
GarageQual        64
GarageCond        64
PoolQC          1161
Fence            931
MiscFeature     1124
dtype: int64
```

```
sns.heatmap(HP_train.isnull())
```

```
]: <matplotlib.axes._subplots.AxesSubplot at 0x28259a03310>
```



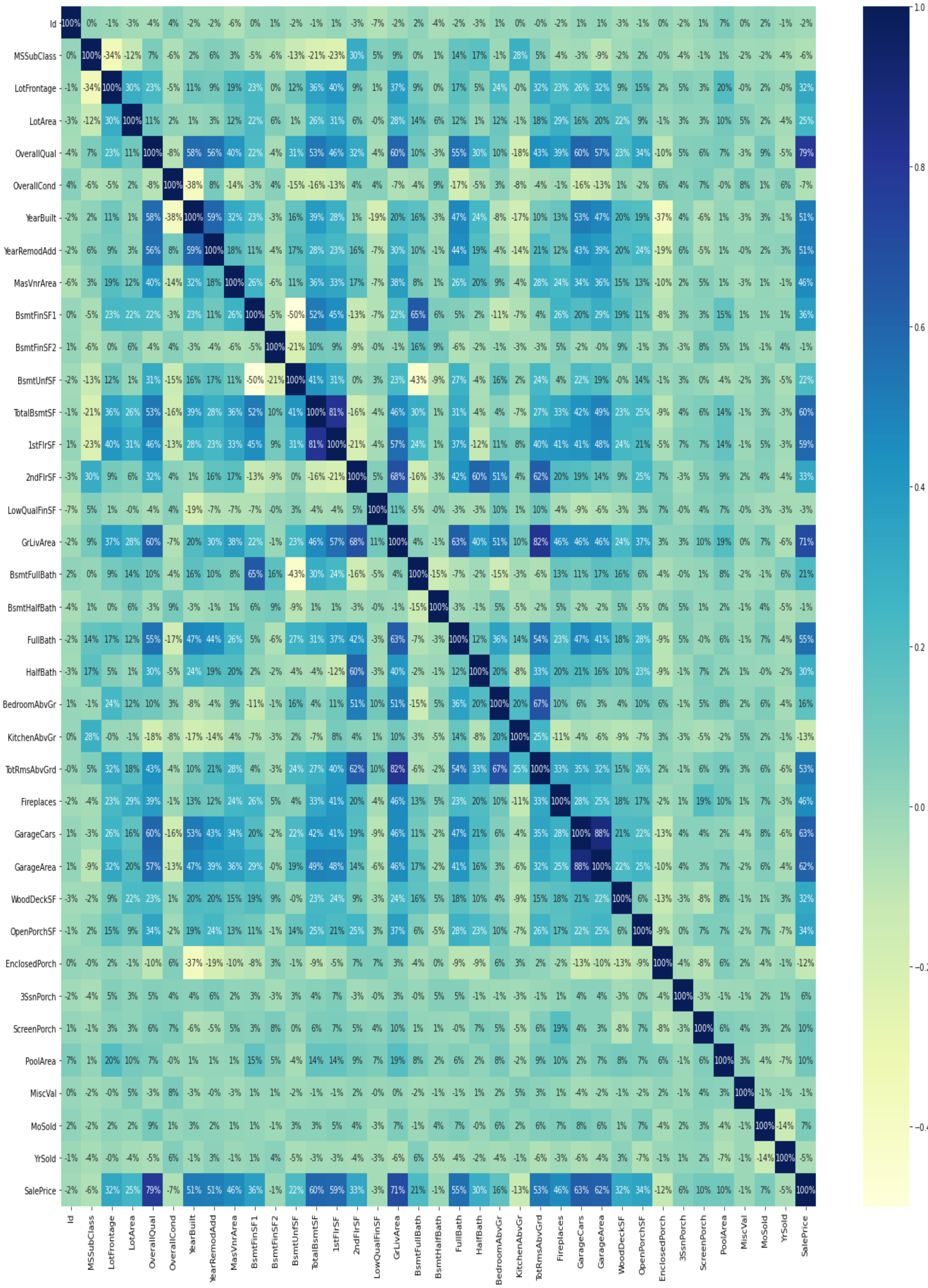
Now, we are filling the missing values of the categorical columns. It is shown below:

```python
# Let's fill the missing values in categorical columns as NA

columns = ["FireplaceQu", "GarageType", "GarageFinish", "GarageQual", "GarageCond", "BsmtExposure", "BsmtFinType2", "BsmtCond", '
HP_train[columns] = HP_train[columns].fillna('NA')

# Let's fill the missing values in MasVnrType with None

HP_train['MasVnrType'] = HP_train['MasVnrType'].fillna('None')

# Let's fill the missing values in GarageYrBlt with 0

HP_train['GarageYrBlt'] = HP_train['GarageYrBlt'].fillna('0')

# Let's Imputing the missing values and replace it with the median

HP_train['LotFrontage'].fillna(HP_train['LotFrontage'].median(),inplace=True)
HP_train['MasVnrArea'].fillna(HP_train['MasVnrArea'].median(),inplace=True)
```

Then we checked the correlation with the help of heatmap.

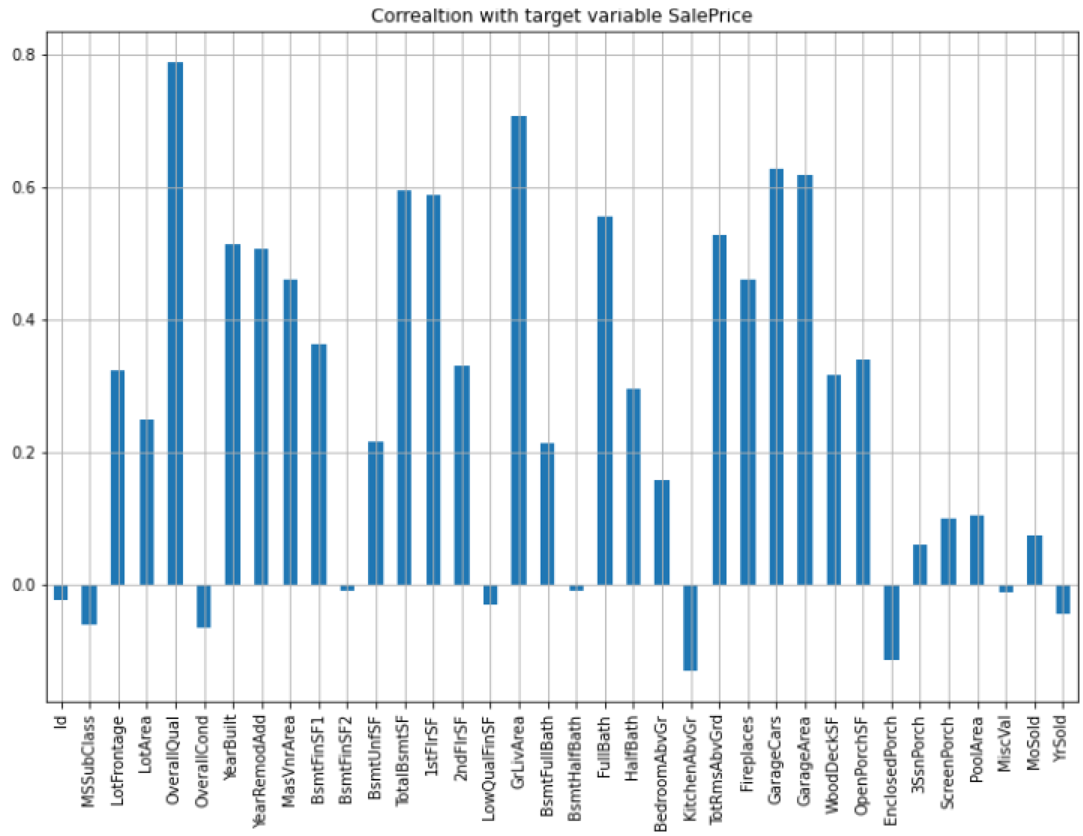
While checking the heatmap of correlation we observed that:

- SalePrice is highly positively correlated with the columns OverallQual, YearBuilt, YearRemodAdd, TotalBsmtSF, 1stFlrSF, GrLivArea, FullBath, TotRmsAbvGrd, GarageCars, GarageArea.
- SalePrice is negatively correlated with OverallCond, KitchenAbvGr, Encloseporch, YrSold.
- We observe multicollinearity in between columns so we will be using Principal Component Analysis(PCA).
- No correlation has been observed between the column Id and other columns so we will be dropping this column.


Id
MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
YearBuilt
YearRemodAdd
MasVnrArea
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
LowQualFinSF
GrLivArea
BsmtFullBath
BsmtHalfBath
FullBath
HalfBath
BedroomAbvGr
KitchenAbvGr
TotRmsAbvGrd
Fireplaces
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
EnclosedPorch
3SsnPorch
ScreenPorch
PoolArea
MiscVal
MoSold
YrSold
SalePrice
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

DATA INPUTS- LOGIC- OUTPUT RELATIONSHIPS

Here we check the correlation between all our feature variables with target variable label.



1. The column OverallQual is most positively correlated with SalePrice.

2. The column KitchenAbvGrd is most negatively correlated with SalePrice.

Set of assumptions related to the problem under consideration

By looking into the target variable label we assumed that it was a Regression type of problem.

We observed multicollinearity in between columns so we assumed that we will be using Principal Component Analysis (PCA).

We also observed that only one single unique value was present in Utilities column so we assumed that we will be dropping these columns.

HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS USED

HARDWARE:

Device specifications

IdeaPad 3 15IIL05

Device name	Rahul
Processor	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
Installed RAM	8.00 GB (7.75 GB usable)
Device ID	4CA2BA68-CE0C-4CD4-B61A-91B7C5CFDCAE
Product ID	00327-35884-66539-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Copy

Rename this PC

Windows specifications

Edition	Windows 10 Home Single Language
Version	20H2
Installed on	09-09-2020
OS build	19042.867
Serial number	PF2DKA0B
Experience	Windows Feature Experience Pack 120.2212.551.0

SOFTWARE:

Jupyter Notebook (Anaconda 3) – Python 3.7.6

Microsoft Excel 2019

LIBRARIES:

The tools, libraries and packages we used for accomplishing this project are pandas, numpy, matplotlib, seaborn, scipy stats, sklearn.decomposition pca, sklearn standardscaler, GridSearchCV, joblib.

From sklearn.preprocessing import StandardScaler

As these columns are different in scale, they are standardized to have common scale while building machine learning model. This is useful when you want to compare data that correspond to different units.

from sklearn.preprocessing import Label Encoder

Label Encoder and One Hot Encoder. These two encoders are parts of the SciKit Learn library in Python, and they are used to convert categorical data, or text data, into numbers, which our predictive models can better understand.

from sklearn.model_selection import train_test_split,cross_val_score

Train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets.

Through pandas library we loaded our csv file ‘Data file’ into dataframe and performed data manipulation and analysis.

With the help of numpy we worked with arrays.

With the help of matplotlib and seaborn we did plot various graphs and figures and done data visualization.

With scipy stats we treated outliers through winsorization technique.

With sklearn’s standardscaler package we scaled all the feature variables onto single scale.

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

HP_train = pd.get_dummies(HP_train, columns = categorical_cols, drop_first=True)
HP_train
```

	Id	MSSubClass	LotFrontage	LotArea	Alley	Utilities	OverallQual	OverallCond	YearBuilt	YearRemodAdd	...	SaleType_ConLI	SaleType_ConLw	Sale
0	127	120	70.0	4928	NaN	AllPub	6	5	1976	1976	...	0	0	
1	889	20	95.0	15865	NaN	AllPub	8	6	1970	1970	...	0	0	
2	793	60	92.0	9920	NaN	AllPub	7	5	1996	1997	...	0	0	
3	110	20	105.0	11751	NaN	AllPub	6	6	1977	1977	...	0	0	
4	422	20	70.0	16635	NaN	AllPub	6	7	1977	2000	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1163	289	20	70.0	9819	NaN	AllPub	5	5	1967	1967	...	0	0	
1164	554	20	67.0	8777	NaN	AllPub	4	5	1949	2003	...	0	0	
1165	196	160	24.0	2280	NaN	AllPub	6	6	1976	1976	...	0	0	
1166	31	70	50.0	8500	Pave	AllPub	4	4	1920	1950	...	0	0	
1167	617	60	70.0	7861	NaN	AllPub	6	5	2002	2003	...	0	0	

1168 rows × 250 columns

MODEL - TRAINING

In [79]:

```
HP_train_x = HP_train_copy.drop(columns=['SalePrice'],axis=1)
y = HP_train_copy['SalePrice']
```

In [80]:

```
#Scaling input variables
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(HP_train_x)
x=pd.DataFrame(x,columns=HP_train_x.columns)
```

In [81]:

```
#PCA
from sklearn.decomposition import PCA
pca=PCA(n_components=90)
xpca=pca.fit_transform(x)
x=xpca
```

In [82]:

```
pd.DataFrame(data=x)
```

0	0.024203	-1.896836	0.132835	0.813668	-2.206555	-1.805352	1.034935	1.148676	0.749111	1.905750	...	-0.325685	-1.675760	0.981776	-0.00743
1	-2.247517	-4.218635	2.433669	2.467969	5.428217	2.217162	4.356637	-0.558530	-2.470528	0.699643	...	0.624664	-0.433794	-0.081001	-0.12627
2	-3.177177	-0.067145	0.034210	-0.530481	1.283673	-2.883606	1.486921	0.123392	0.735488	-1.436449	...	-0.406263	-0.565125	0.478300	0.83073
3	-2.108239	-3.530542	1.215853	2.012713	1.144982	0.329248	-3.079172	-0.169040	1.561101	0.786966	...	-0.983040	0.428533	-0.006171	0.38298
4	-3.131142	-1.375665	0.344652	1.783524	0.114749	-0.337493	-0.858986	1.613669	-0.122118	-1.224684	...	-1.009613	0.969119	1.308214	-0.68463
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1163	3.795601	-2.918683	-1.471876	-0.272967	-2.503072	0.282076	-1.204942	-0.265730	0.679485	0.523214	...	-0.104531	-0.411484	-0.403443	-1.22085
1164	4.015035	2.373468	10.994015	-4.930745	-3.243131	0.556682	0.472173	-1.424523	-1.052938	-0.064363	...	0.194376	1.032288	1.056317	0.45919

from sklearn.linear_model import Linear Regressor

The library sklearn can be used to perform logistic regression in a few lines as shown using the Linear Regression class. It also supports multiple features. It requires the input values to be in a specific format hence they have been reshaped before training using the fit method.

from sklearn.tree import DecisionTree Regressor

Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data. The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy

from sklearn.ensemble import RandomForestRegressor

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

Through GridSearchCV we were able to find the right parameters for hyperparameter tuning. Through joblib we saved our model in csv format.

MODEL/S DEVELOPMENT AND EVALUATION

IDENTIFICATION OF POSSIBLE PROBLEM-SOLVING APPROACHES (METHODS)

We first converted all our categorical variables to numeric variables with the help of dummy variables to checkout and dropped the columns which we felt were unnecessary.

We observed skewness in data so we tried to remove the skewness through treating outliers with winsorization technique.

The data was improper scaled so we scaled the feature variables on a single scale using sklearn's StandardScaler package.

There were too many (256) feature variables in the data so we reduced it to 100 with the help of Principal Component Analysis(PCA) by plotting Eigenvalues and taking the number of nodes as our number of feature variables.

TESTING OF IDENTIFIED APPROACHES (ALGORITHMS)

The algorithms we used for the training and testing are as follows:-

- Linear Regression
- Lasso
- Ridge
- Elastic Net
- KNeighbors Regressor
- Decision Tree Regressor
- Random Forest Regressor
- Ada Boost Regressor
- Gradient Boosting Regressor

```
# Let's split the dataset into test and train

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=98)
```

```
model=[LinearRegression(),
       DecisionTreeRegressor(),
       KNeighborsRegressor(),
       Lasso(),
       Ridge(),
       ElasticNet(),
       RandomForestRegressor(),
       AdaBoostRegressor(),
       GradientBoostingRegressor()
      ]
for m in model:
    m.fit(x_train,y_train)
    print('score of',m,'is:',m.score(x_train,y_train))
    predm=m.predict(x_test)
    print('Error:')
    print('Mean absolute error:',mean_absolute_error(y_test,predm))
    print('Mean squared error:',mean_squared_error(y_test,predm))
    print('Root Mean Squared Error:',np.sqrt(mean_squared_error(y_test,predm)))
    print("r2_score:",r2_score(y_test,predm))
    print('*****************************************************************')
    print('\n')
```

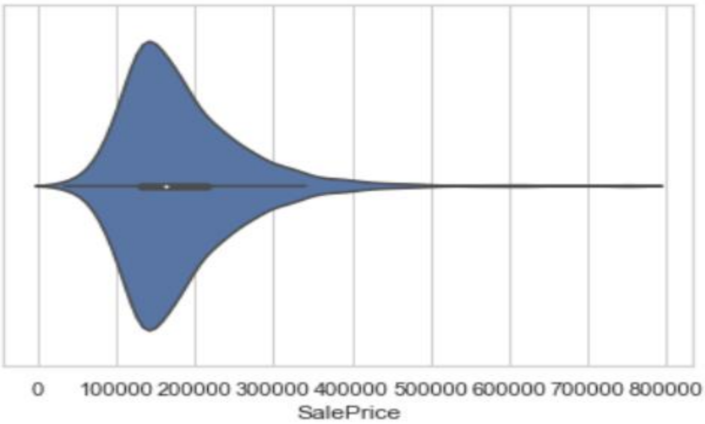
RUN AND EVALUATE SELECTED MODELS

```
score of LinearRegression() is: 0.8193724205930797
Error:
Mean absolute error: 21469.888017041507
Mean squared error: 946775046.8606462
Root Mean Squared Error: 30769.709892370553
r2_score: 0.8521998035857644
************************************************************


score of DecisionTreeRegressor() is: 1.0
Error:
Mean absolute error: 31141.619658119656
Mean squared error: 2355944553.0982904
Root Mean Squared Error: 48538.07323223997
r2_score: 0.632215626252951
************************************************************


score of KNeighborsRegressor() is: 0.7922005613367162
Error:
Mean absolute error: 26561.786324786324
Mean squared error: 1692406532.774017
Root Mean Squared Error: 41138.868880585636
r2_score: 0.7357999465806027
************************************************************


score of Lasso() is: 0.8193724100950655
Error:
Mean absolute error: 21467.05820809681
Mean squared error: 946624108.8469466
Root Mean Squared Error: 30767.257090077863
r2_score: 0.8522233663825927
************************************************************


score of ElasticNet() is: 0.8119839096893122
Error:
Mean absolute error: 20223.242005881573
Mean squared error: 919772937.6369351
Root Mean Squared Error: 30327.75853301617
r2_score: 0.8564150784391695
************************************************************


score of RandomForestRegressor() is: 0.9638030787444078
Error:
Mean absolute error: 19279.14829059829
Mean squared error: 799910274.4639188
Root Mean Squared Error: 28282.685064610094
r2_score: 0.8751267303975176
************************************************************


score of AdaBoostRegressor() is: 0.8366627944277063
Error:
Mean absolute error: 27251.98379996723
Mean squared error: 1273075603.2147005
Root Mean Squared Error: 35680.1850221478
r2_score: 0.8012613188008966
************************************************************


score of GradientBoostingRegressor() is: 0.972441511909555
Error:
Mean absolute error: 18536.634723025458
Mean squared error: 659473842.6497654
Root Mean Squared Error: 25680.222792058586
r2_score: 0.897050134773955
************************************************************
```

KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION

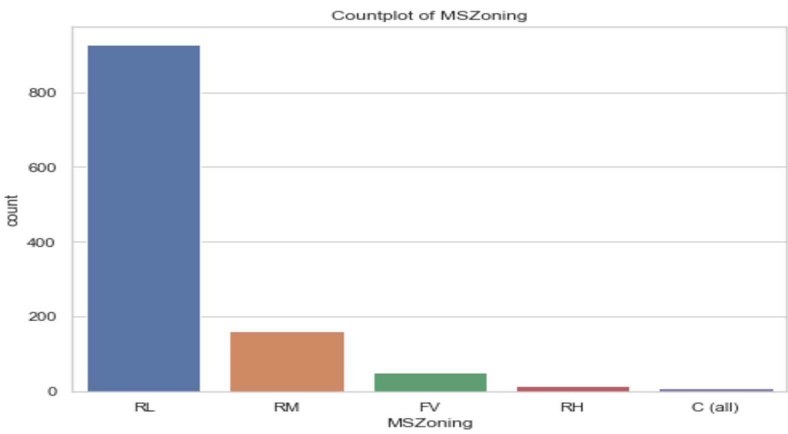
We used the metric Root Mean Squared Error by selecting the Gradient Boosting Regressor model which was giving us best(minimum) RMSE score.

DATA VISUALIZATIONS



Observation:

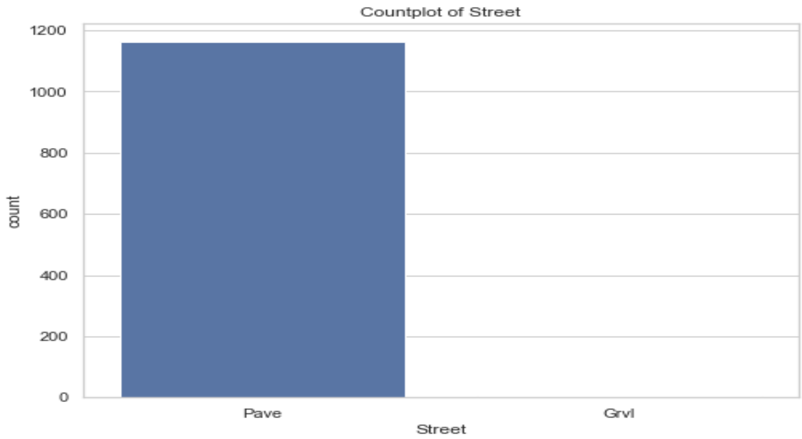
Maximum number of SalePrice lies between 140000 and 230000.




```
Out[23]: RL          928
         RM          163
         FV           52
         RH           16
         C (all)       9
         Name: MSZoning, dtype: int64
```

Observation:

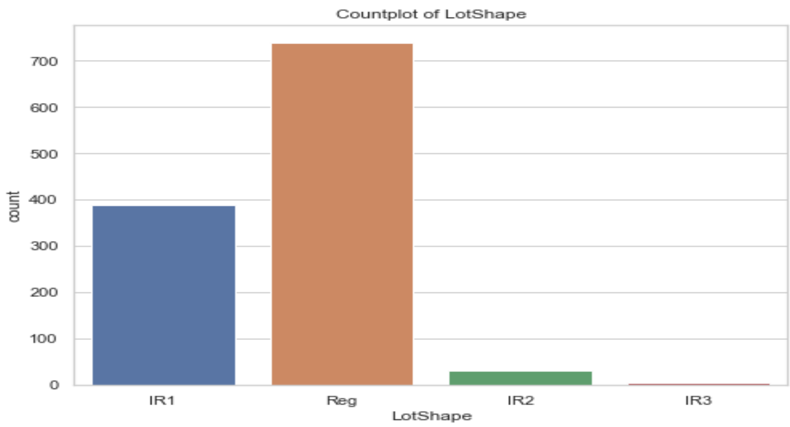
Maximum, 928 number of MSZoning are RL.




```
Out[24]: Pave     1164
         Grvl        4
         Name: Street, dtype: int64
```

Observation:

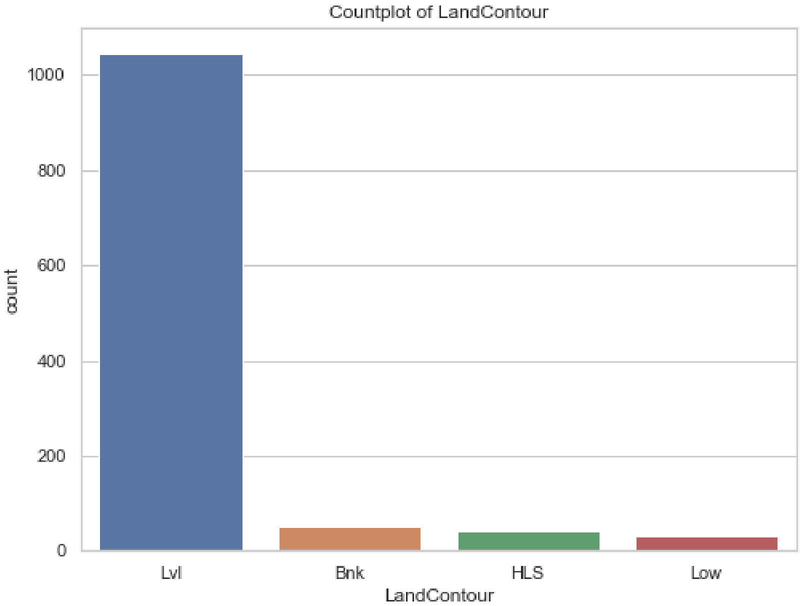
Maximum, 1164 number of Street are Pave where as only 4 are Grvl.




```
Out[25]: Reg     740
         IR1     390
         IR2      32
         IR3       6
         Name: LotShape, dtype: int64
```

Observation:

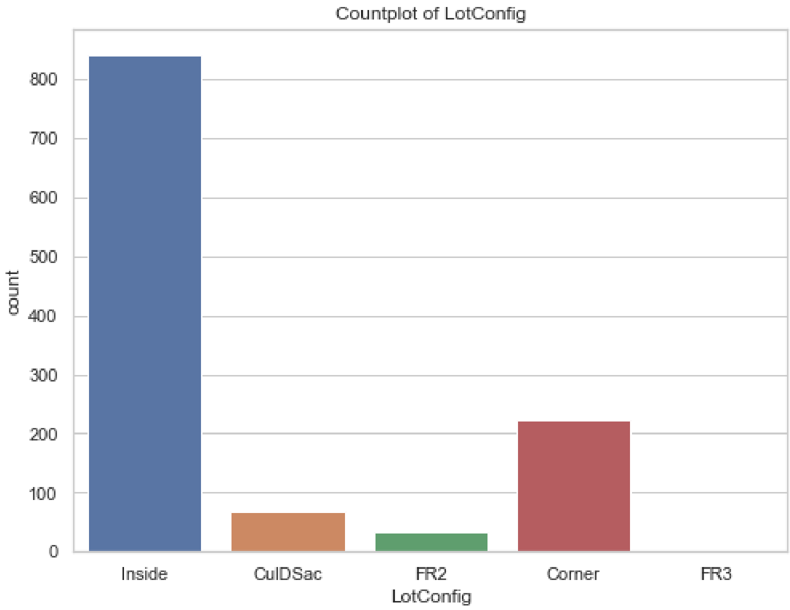
Maximum, 740 number of LotShape are Reg.




```
Out[26]: Lvl     1046
         Bnk       50
         HLS       42
         Low       30
         Name: LandContour, dtype: int64
```

Observation:

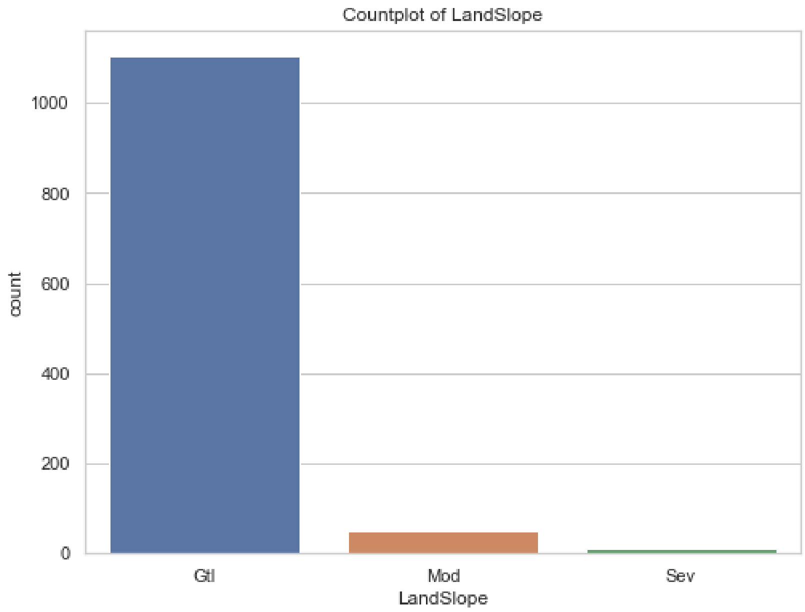
Maximum, 1046 number of LandContour are Lvl.




```
Out[27]: Inside      842
         Corner      222
         CulDSac      69
         FR2          33
         FR3           2
         Name: LotConfig, dtype: int64
```

Observation:

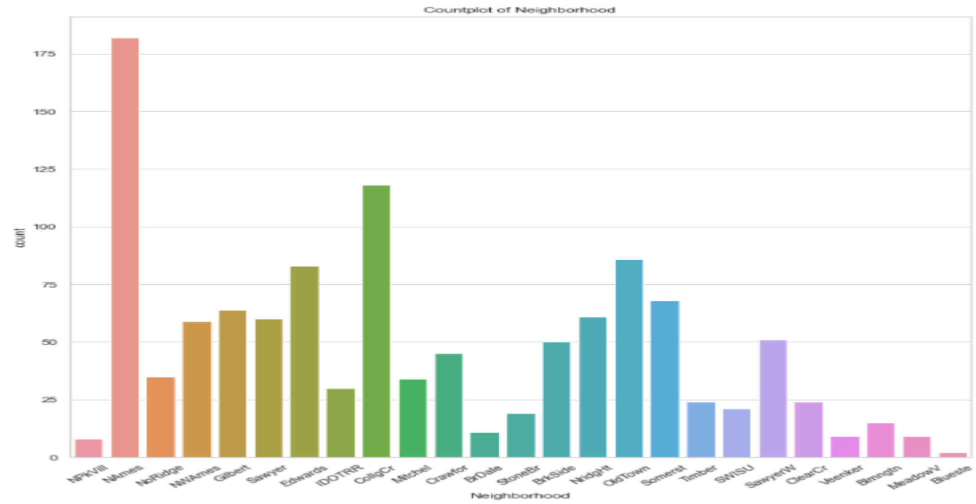
Maximum, 842 number of LotConfig are Inside.




```
Out[28]: Gtl    1105
         Mod      51
         Sev      12
         Name: LandSlope, dtype: int64
```

Observation:

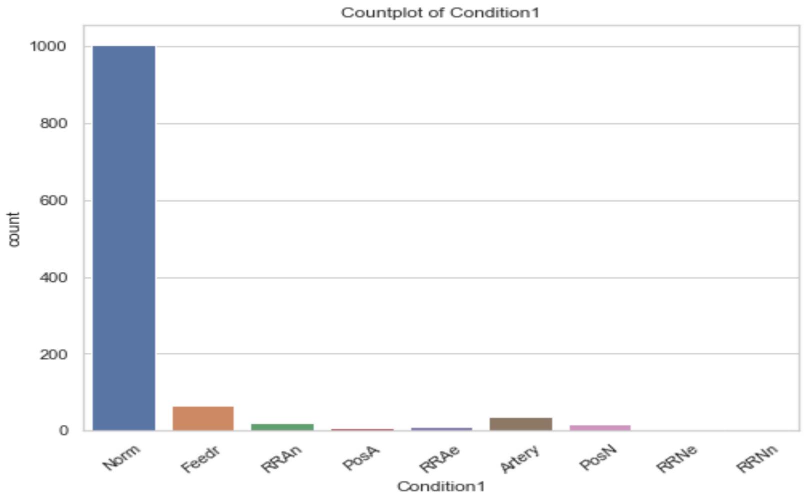
Maximum, 1105 number of LandSlope are Gtl.




```
29]:  NAmes       182
      CollgCr     118
      OldTown      86
      Edwards      83
      Somerst      68
      Gilbert      64
```

Observation:

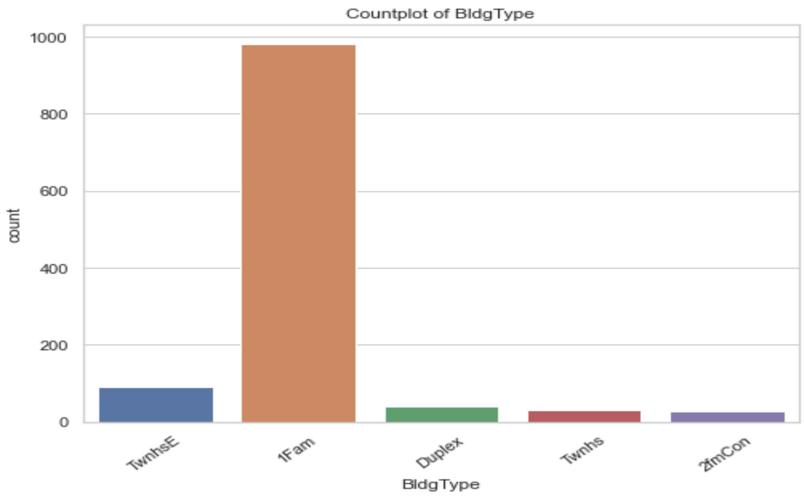
Maximum, 182 number of Neighborhood are Names.




```
Out[30]: Norm      1005
         Feedr       67
         Artery      38
         RRAn        20
         PosN        17
         RRAe         9
```

Observation:

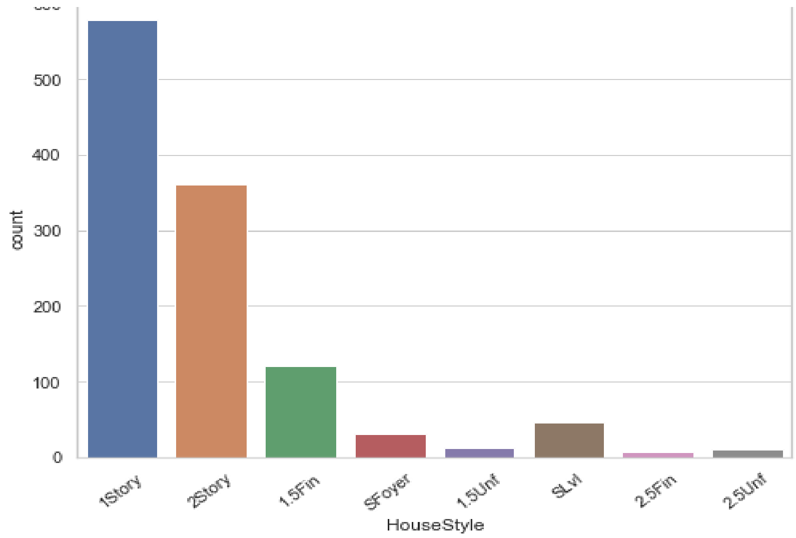
Maximum, 1005 number of Condition1 is Norm.




```
Out[31]: 1Fam       981
         TwnhsE      90
         Duplex      41
         Twnhs       29
         2fmCon      27
         Name: BldgType, dtype: int64
```

Observation:

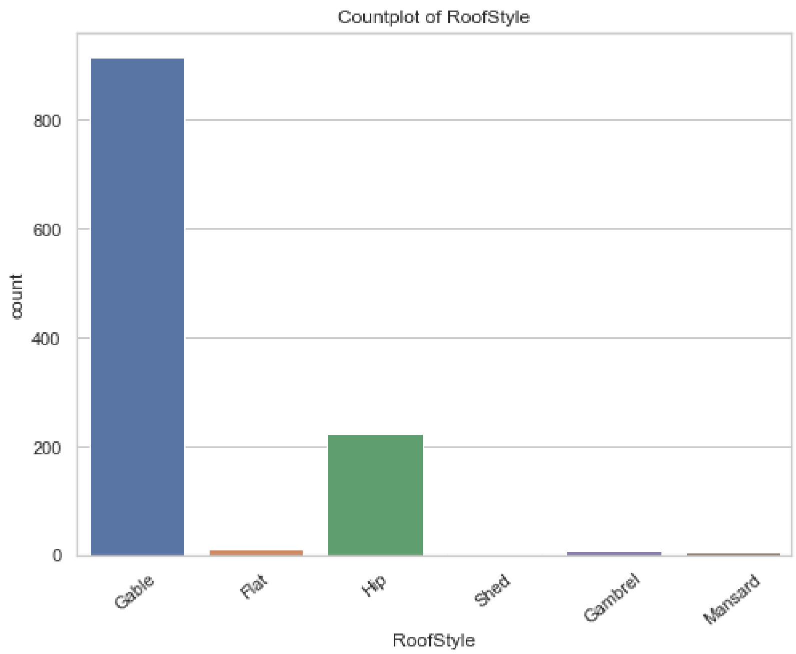
Maximum, 981 number of BldgType are 1Fam.




```
Out[32]: 1Story    578
         2Story    361
         1.5Fin    121
         SLvl       47
         SFoyer     32
         1.5Unf     12
         2.5Unf     10
         2.5Fin      7
```

Observation:

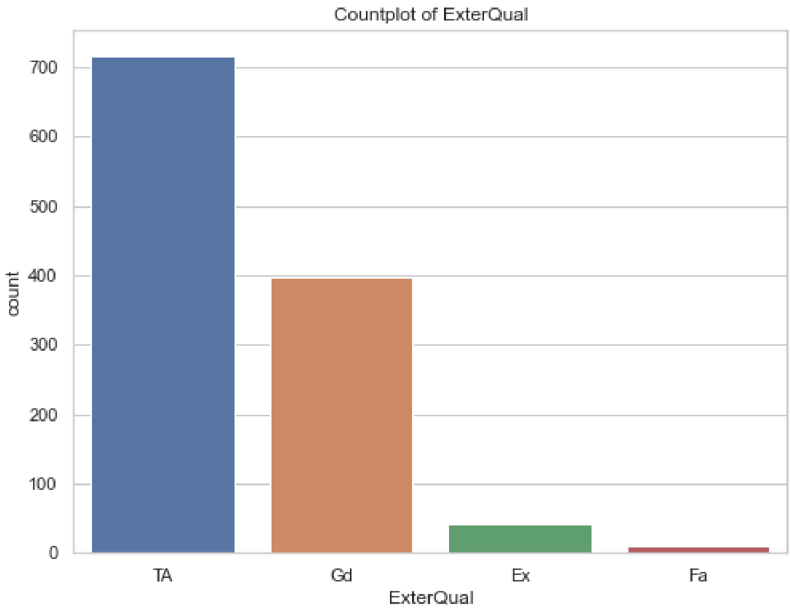
1 Story has highest number of count followed by 2Story, 1.5Fin, SlvL etc




```
Out[33]: Gable      915
         Hip        225
         Flat        12
         Gambrel      9
         Mansard      5
         Shed         2
```

Observation:

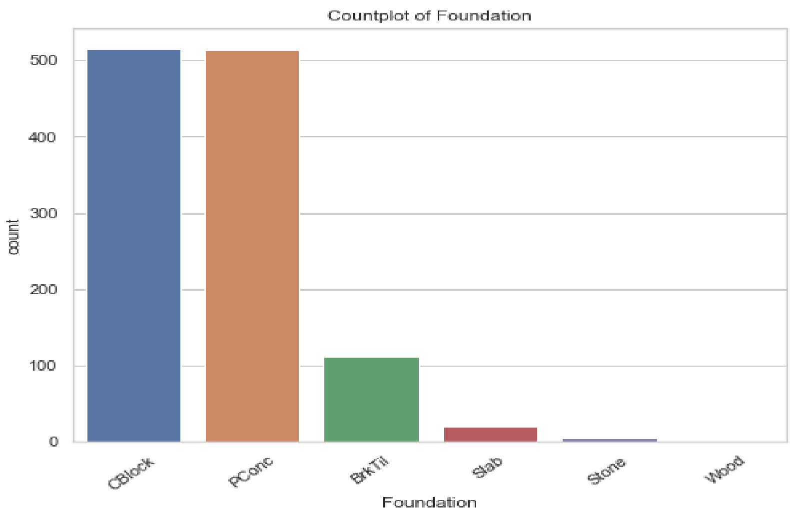
Maximum, 915 number of RoofStyle are Gable.




```
Out[34]: TA     717
         Gd     397
         Ex      43
         Fa      11
         Name: ExterQual, dtype: int64
```

Observation:

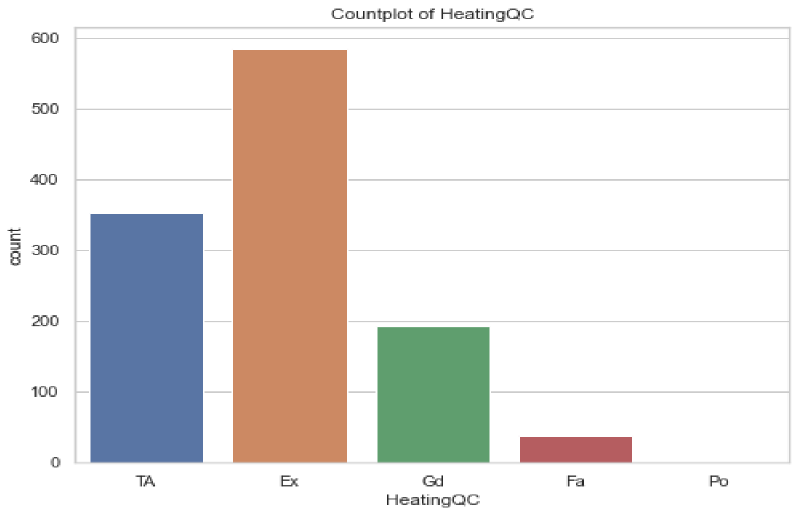
Maximum, 717 number of ExterQual is TA.




```
Out[35]: CBlock    516
         PConc     513
         BrkTil    112
         Slab       21
         Stone       5
```

Observation:

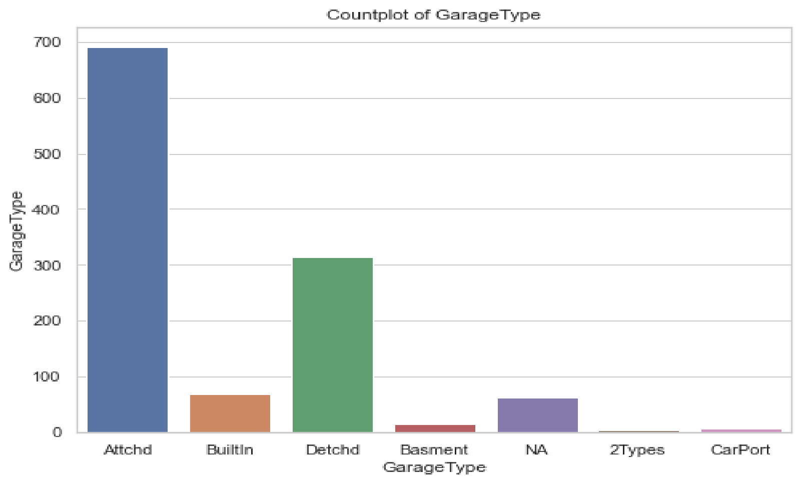
Maximum, 516 number of Foundation are CBlock.




```
Out[36]: Ex     585
         TA     352
         Gd     192
         Fa      38
         Po       1
         Name: HeatingQC, dtype: int64
```

Observation:

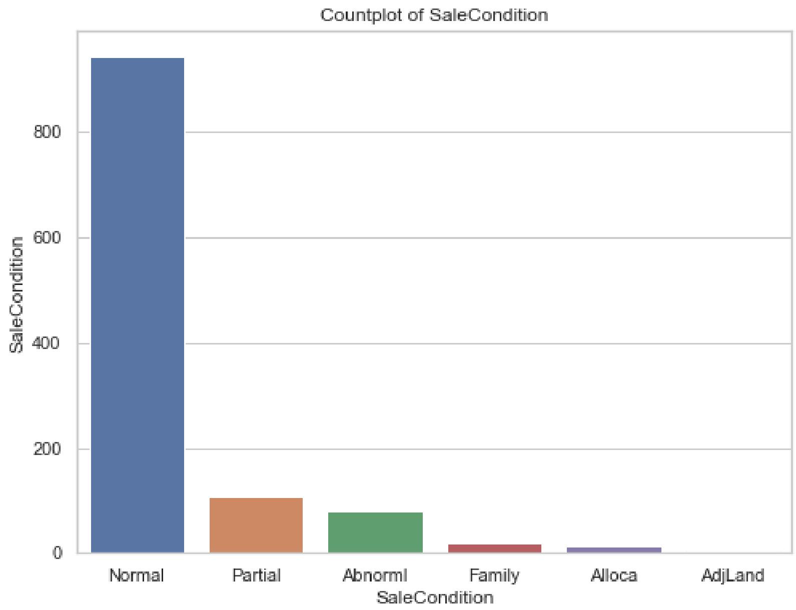
Maximum, 585 number of HeatingQC is Ex.




```
Out[37]: Attchd      691
         Detchd      314
         BuiltIn      70
         NA           64
         Basment      16
```

Observation:

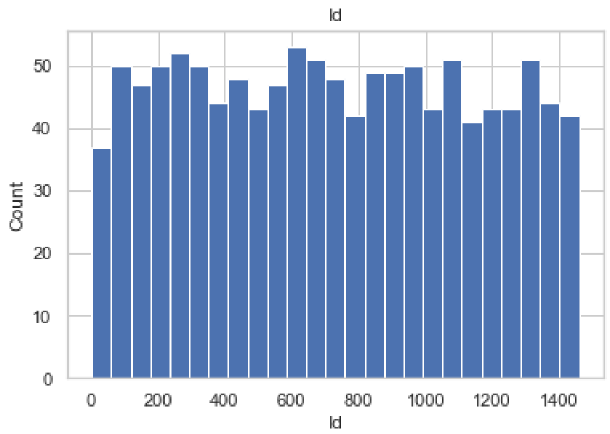
Maximum, 691 number of GarageType are Attchd.

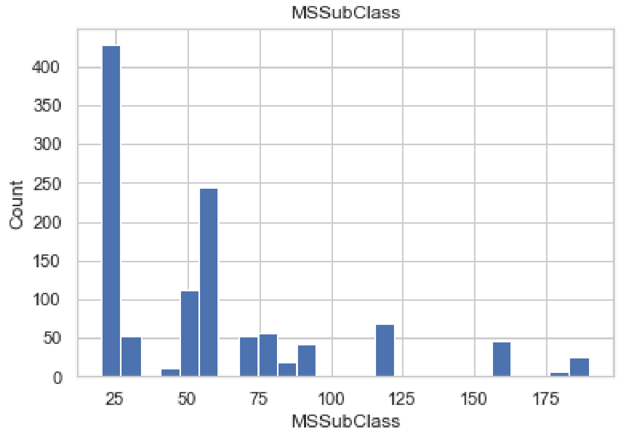


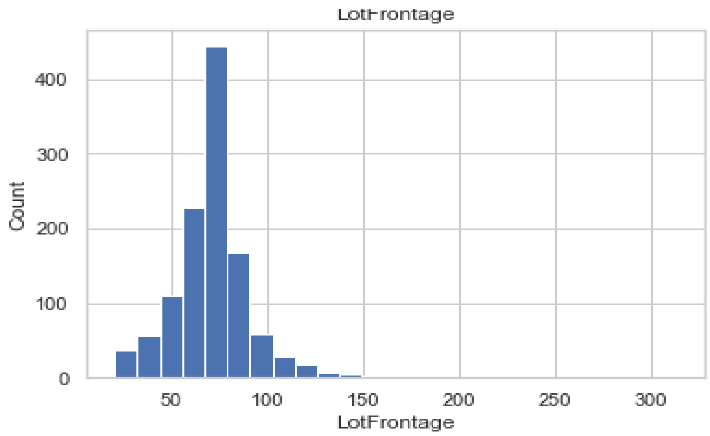

```
Out[38]: Normal     945
         Partial    108
         Abnorml     81
         Family      18
         Alloca      12
         AdjLand      4
         Name: SaleCondition, dtype: int64
```

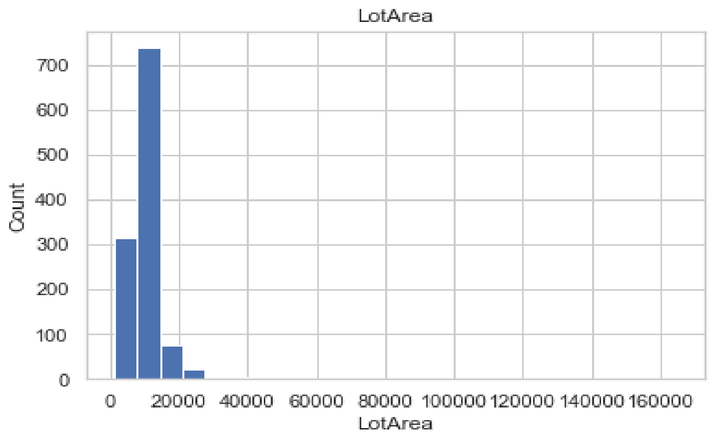
Observation:

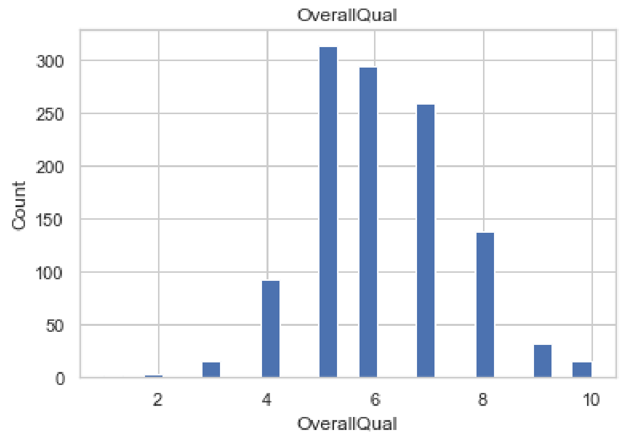
Maximum, 945 number of SaleCondition is normal.

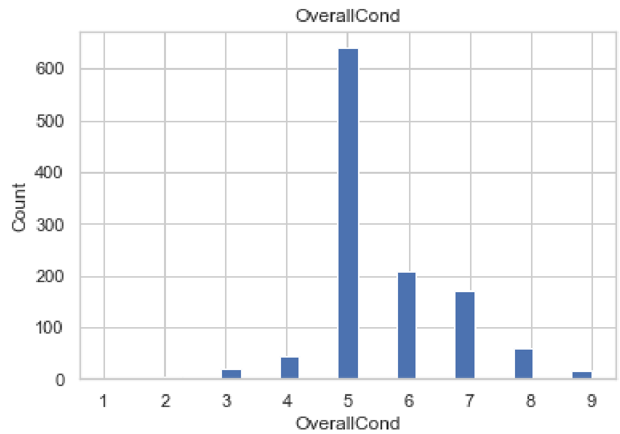

Id
Count
Id

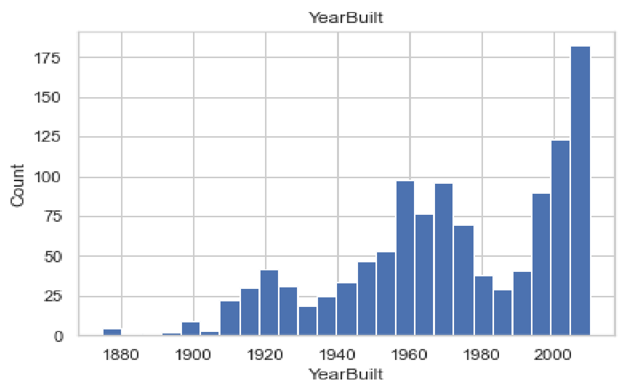

MSSubClass
Count
MSSubClass

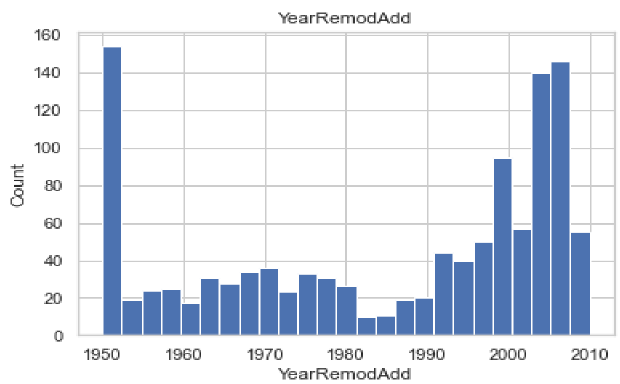

LotFrontage
Count
LotFrontage

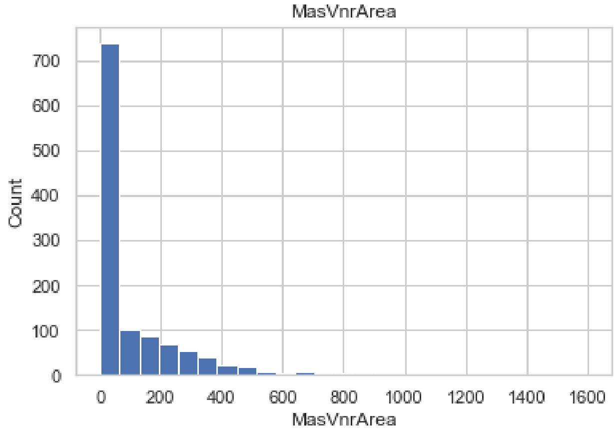

LotArea
Count
LotArea

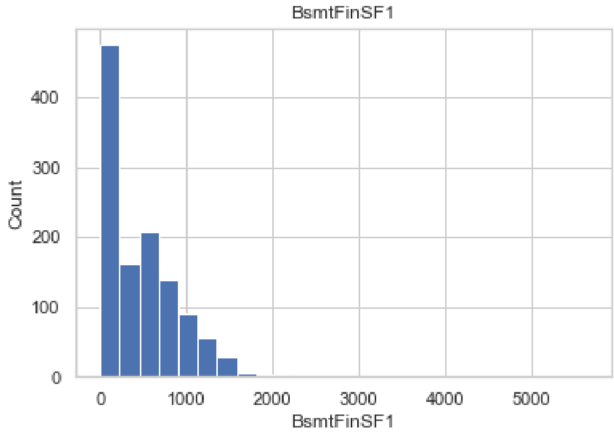

OverallQual
Count
0
50
100
150
200
250
300
2
4
6
8
10
OverallQual

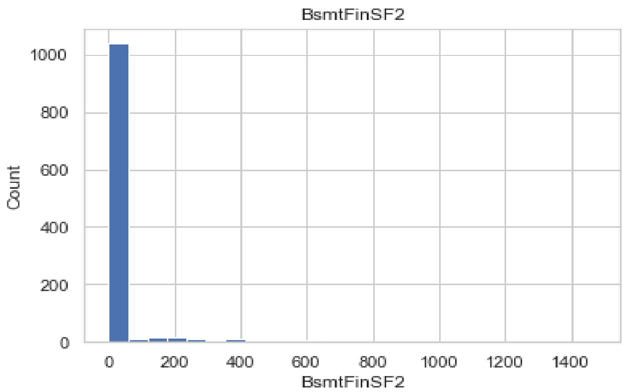

OverallCond
Count
0
100
200
300
400
500
600
1
2
3
4
5
6
7
8
9
OverallCond

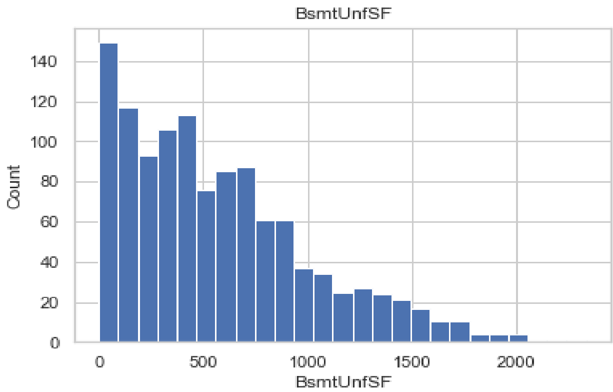

YearBuilt
Count
0
25
50
75
100
125
150
175
1880
1900
1920
1940
1960
1980
2000
YearBuilt

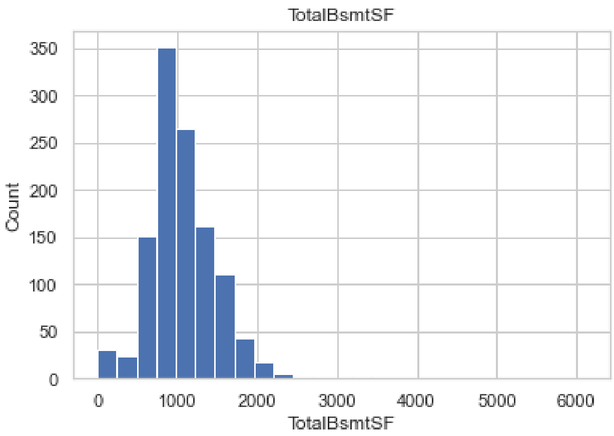

YearRemodAdd
Count
0
20
40
60
80
100
120
140
160
1950
1960
1970
1980
1990
2000
2010
YearRemodAdd

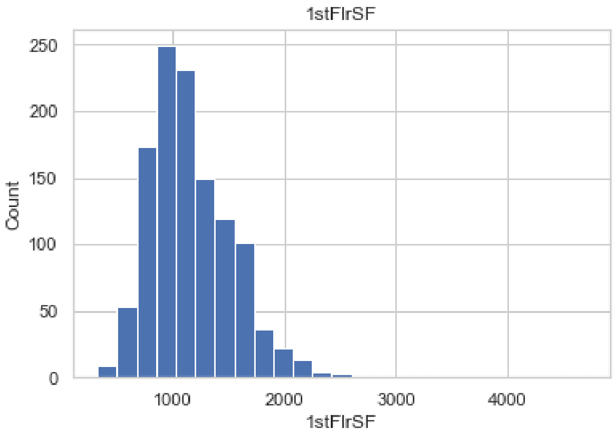

MasVnrArea
Count
700
600
500
400
300
200
100
0
0
200
400
600
800
1000
1200
1400
1600
MasVnrArea

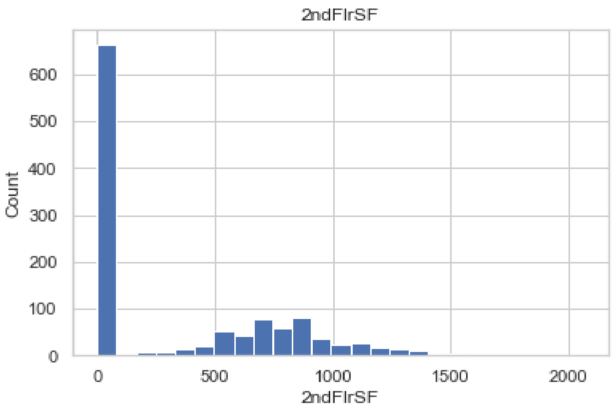

BsmtFinSF1
Count
400
300
200
100
0
0
1000
2000
3000
4000
5000
BsmtFinSF1

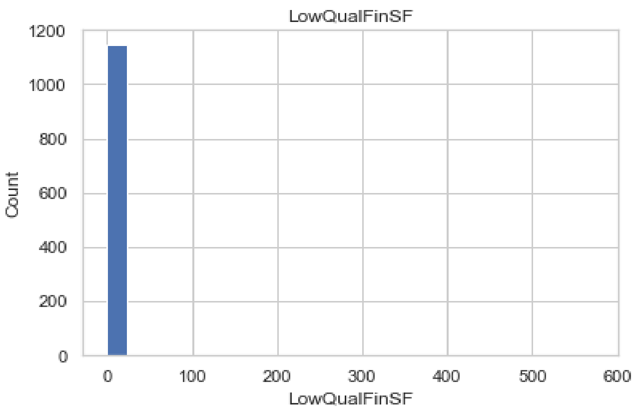

BsmtFinSF2
Count
1000
800
600
400
200
0
0
200
400
600
800
1000
1200
1400
BsmtFinSF2

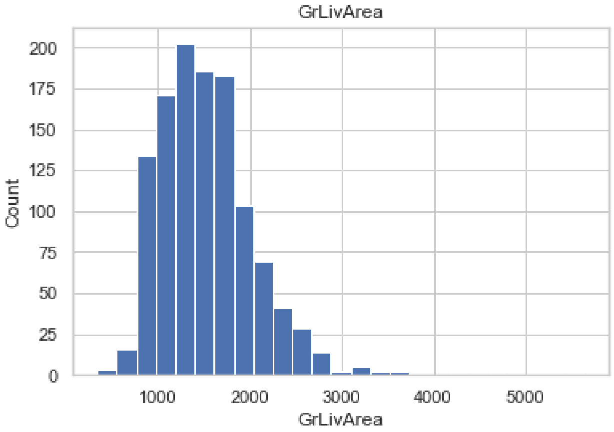

BsmtUnfSF
Count
140
120
100
80
60
40
20
0
0
500
1000
1500
2000
BsmtUnfSF

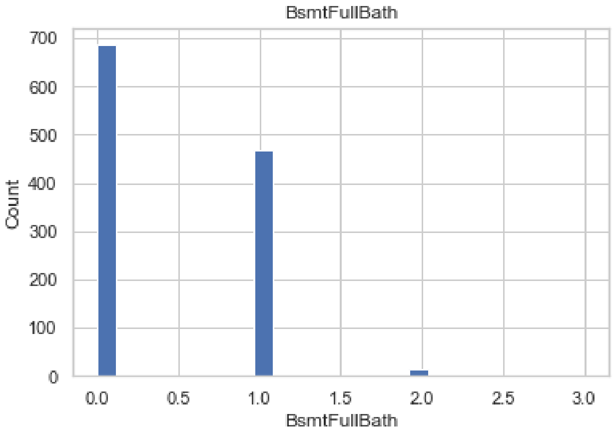

TotalBsmtSF
Count
350
300
250
200
150
100
50
0
0
1000
2000
3000
4000
5000
6000
TotalBsmtSF

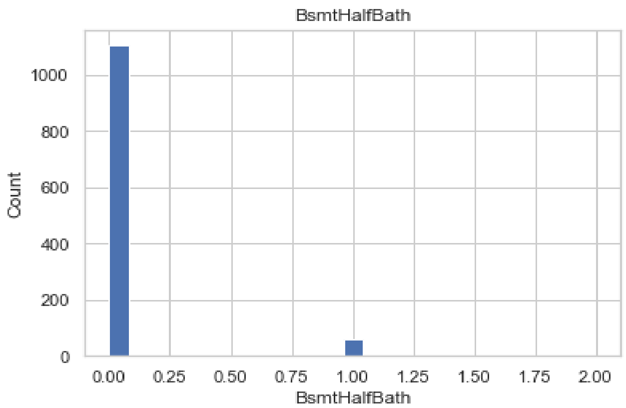

1stFlrSF
Count
250
200
150
100
50
0
1000
2000
3000
4000
1stFlrSF

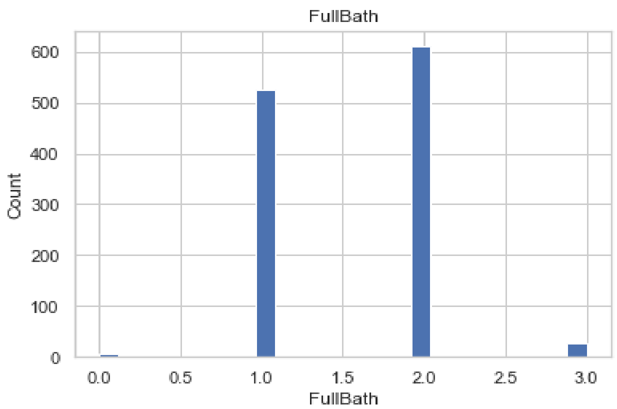

2ndFlrSF
Count
600
500
400
300
200
100
0
0
500
1000
1500
2000
2ndFlrSF

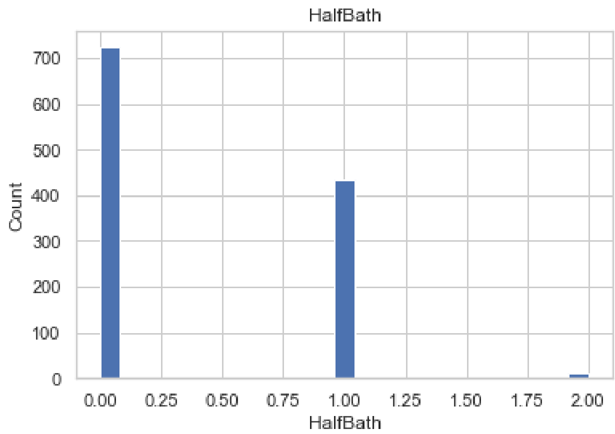

LowQualFinSF
Count
1200
1000
800
600
400
200
0
0
100
200
300
400
500
600
LowQualFinSF

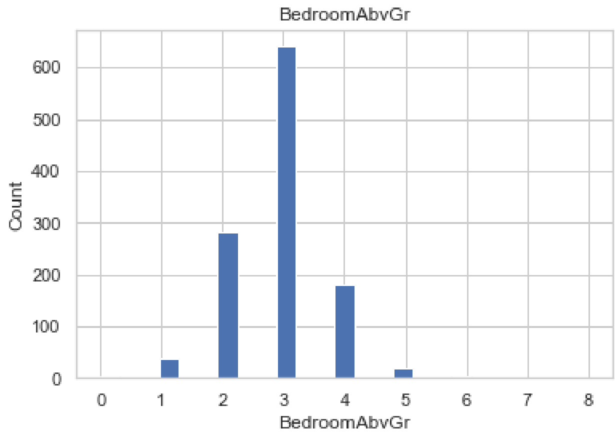

GrLivArea
Count
0
25
50
75
100
125
150
175
200
1000
2000
3000
4000
5000
GrLivArea

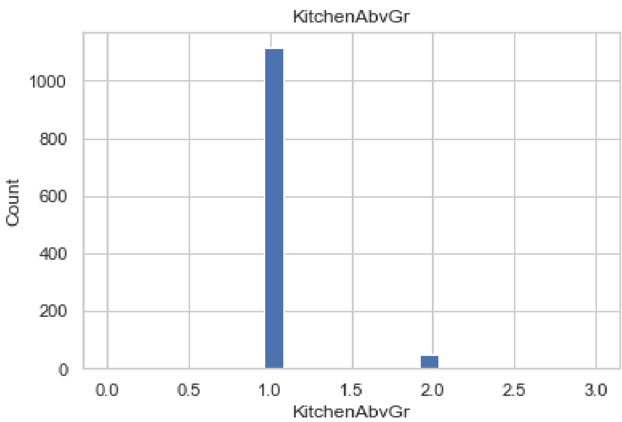

BsmtFullBath
Count
0
100
200
300
400
500
600
700
0.0
0.5
1.0
1.5
2.0
2.5
3.0
BsmtFullBath

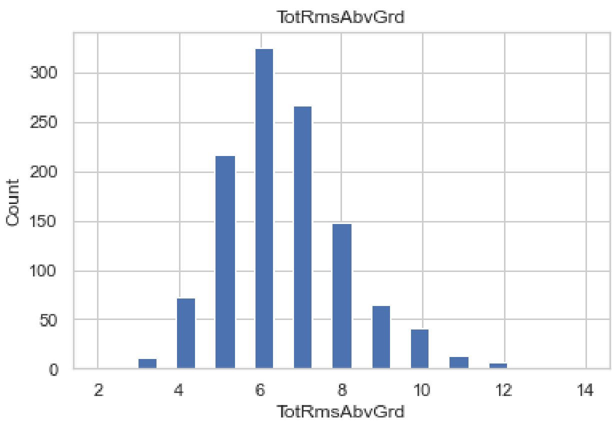

BsmtHalfBath
Count
0
200
400
600
800
1000
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
BsmtHalfBath

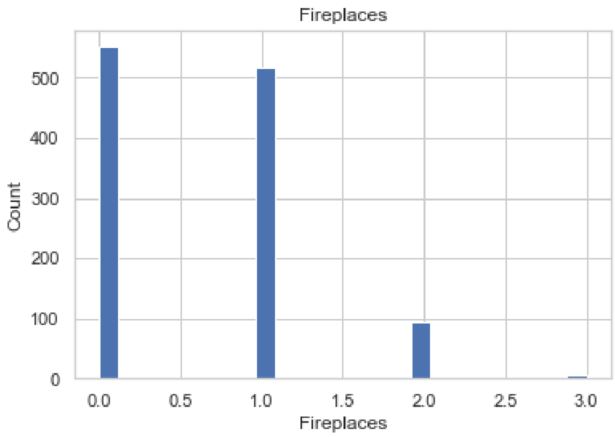

FullBath
Count
0
100
200
300
400
500
600
0.0
0.5
1.0
1.5
2.0
2.5
3.0
FullBath


HalfBath
Count
0
100
200
300
400
500
600
700
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
HalfBath

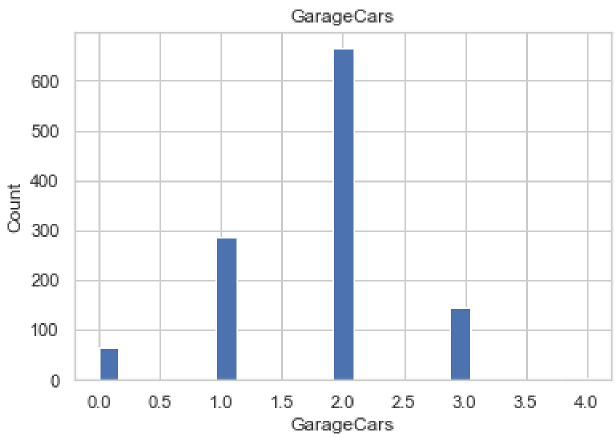

BedroomAbvGr
Count
0
100
200
300
400
500
600
0
1
2
3
4
5
6
7
8
BedroomAbvGr


KitchenAbvGr
Count
0
200
400
600
800
1000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
KitchenAbvGr

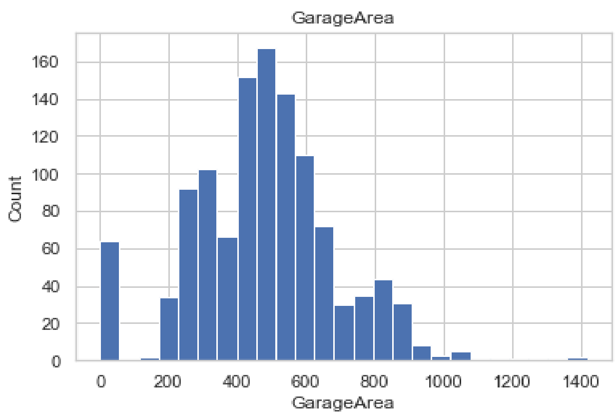

TotRmsAbvGrd
Count
0
50
100
150
200
250
300
2
4
6
8
10
12
14
TotRmsAbvGrd



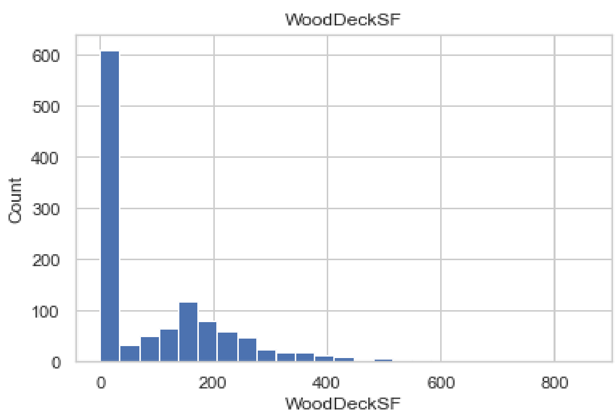
Fireplaces
Count
500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Fireplaces



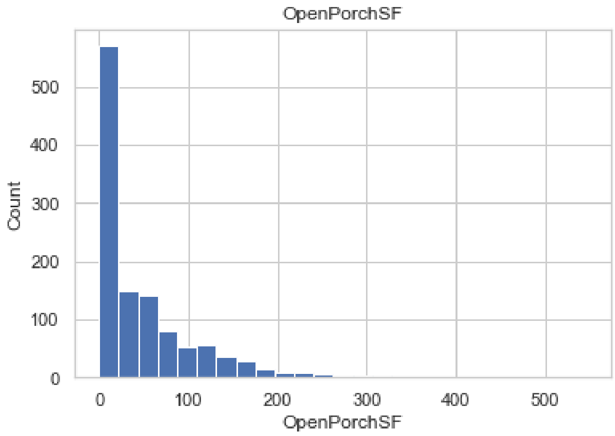
GarageCars
Count
600
500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
GarageCars

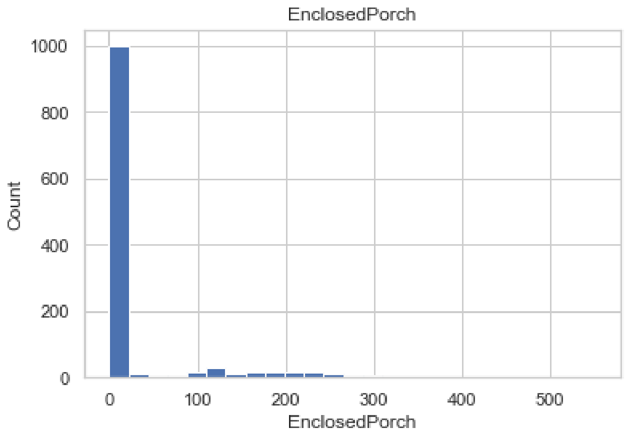


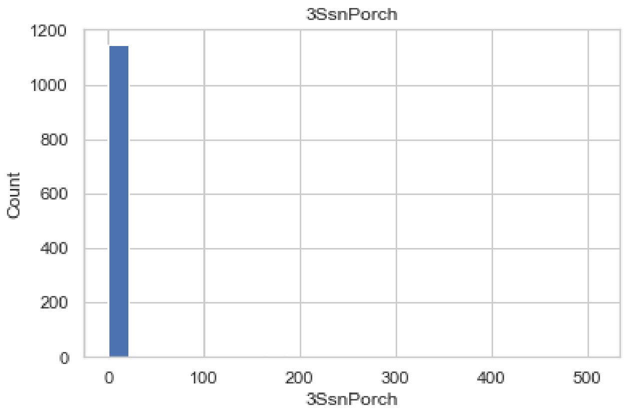
GarageArea
Count
160
140
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
GarageArea

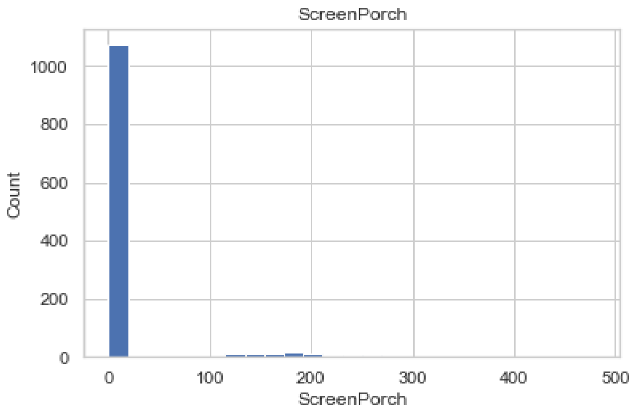


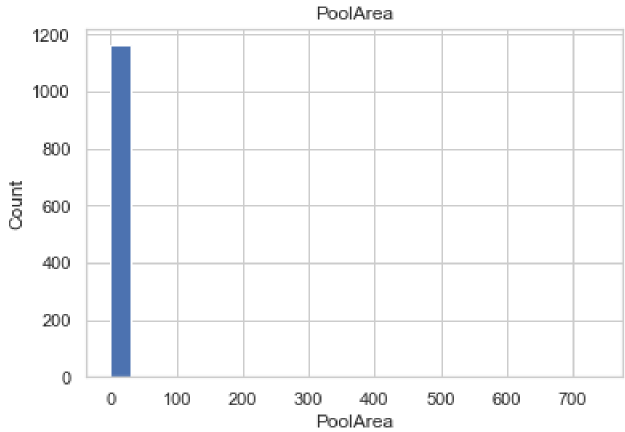
WoodDeckSF
Count
600
500
400
300
200
100
0
0
200
400
600
800
WoodDeckSF

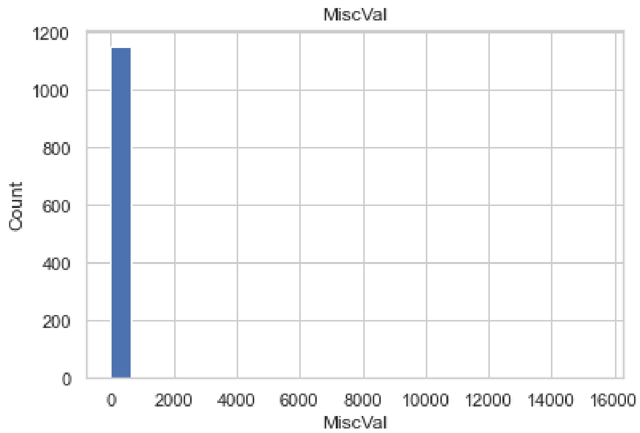

OpenPorchSF
Count
500
400
300
200
100
0
0
100
200
300
400
500
OpenPorchSF

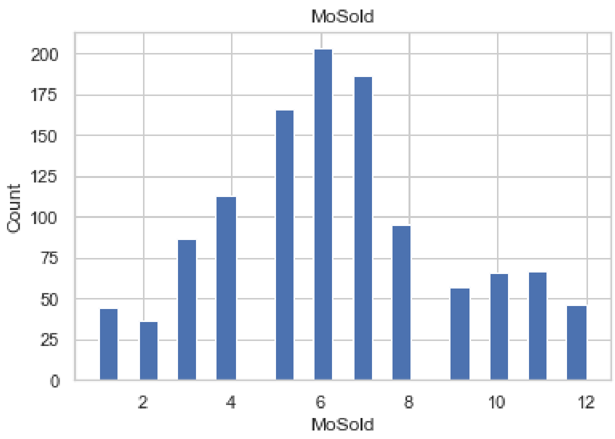

EnclosedPorch
Count
1000
800
600
400
200
0
0
100
200
300
400
500
EnclosedPorch


3SsnPorch
Count
1200
1000
800
600
400
200
0
0
100
200
300
400
500
3SsnPorch

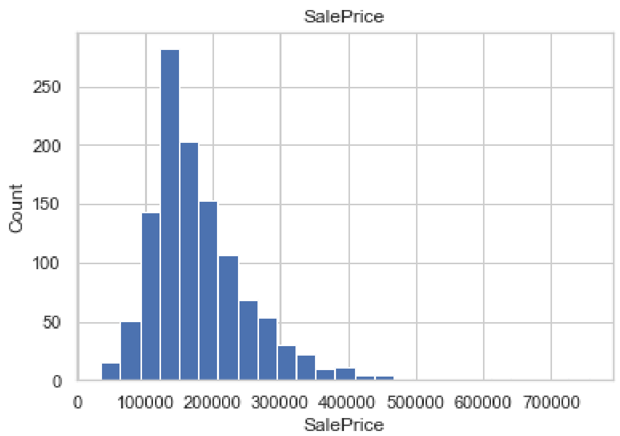

ScreenPorch
Count
1000
800
600
400
200
0
0
100
200
300
400
500
ScreenPorch

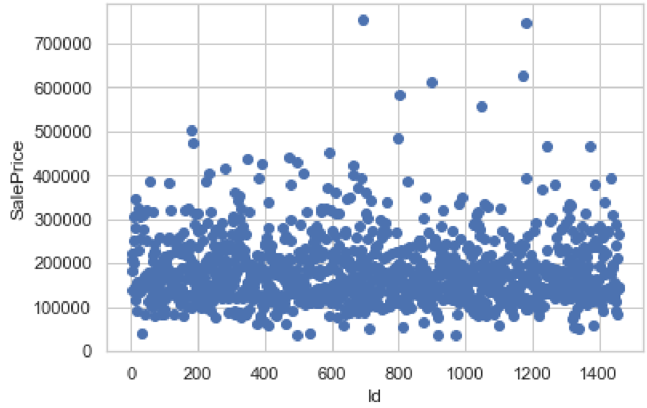

PoolArea
1200
1000
800
600
400
200
0
Count
0
100
200
300
400
500
600
700
PoolArea


MiscVal
1200
1000
800
600
400
200
0
Count
0
2000
4000
6000
8000
10000
12000
14000
16000
MiscVal

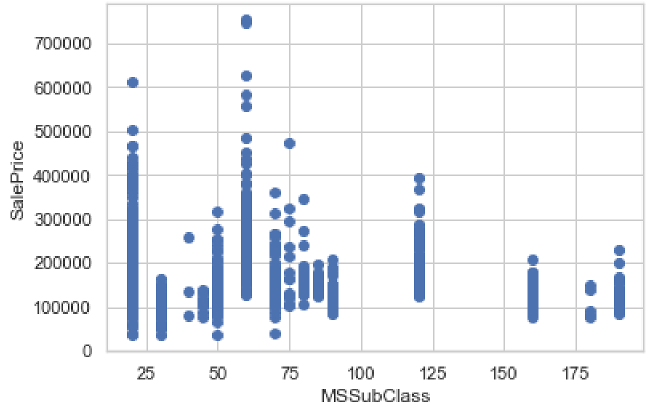

MoSold
Count
MoSold


YrSold
Count
YrSold

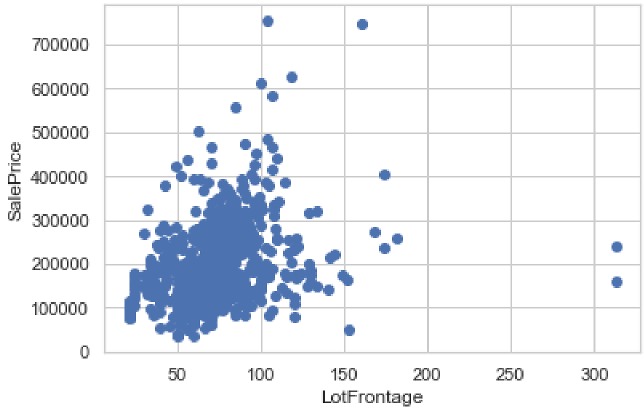

SalePrice
Count
SalePrice



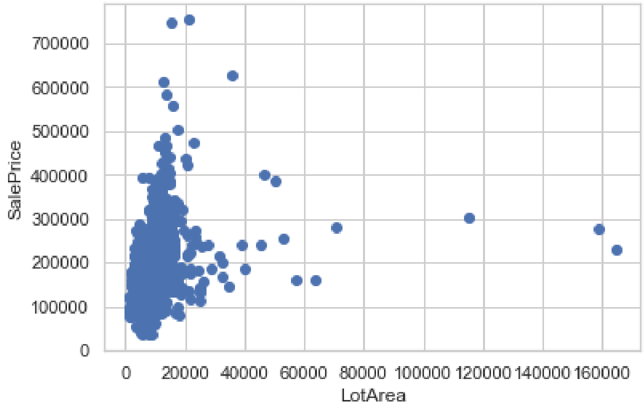
700000
600000
500000
400000
300000
200000
100000
0
SalePrice
0
200
400
600
800
1000
1200
1400
Id



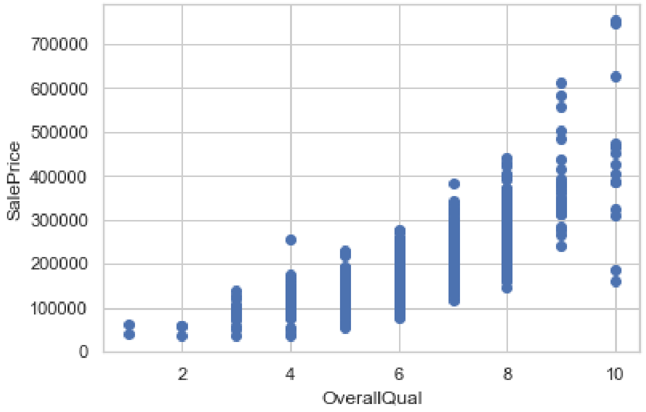
700000
600000
500000
400000
300000
200000
100000
0
SalePrice
25
50
75
100
125
150
175
MSSubClass

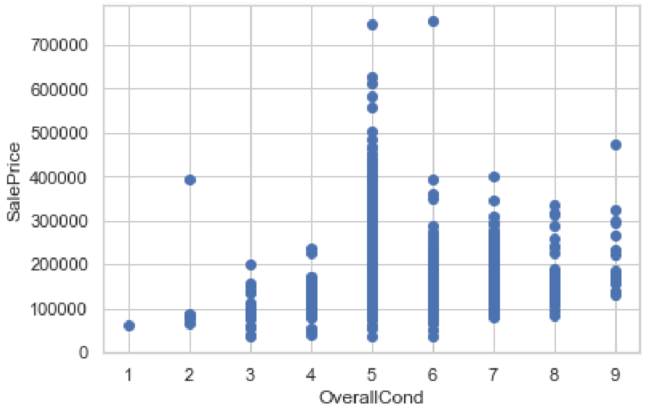


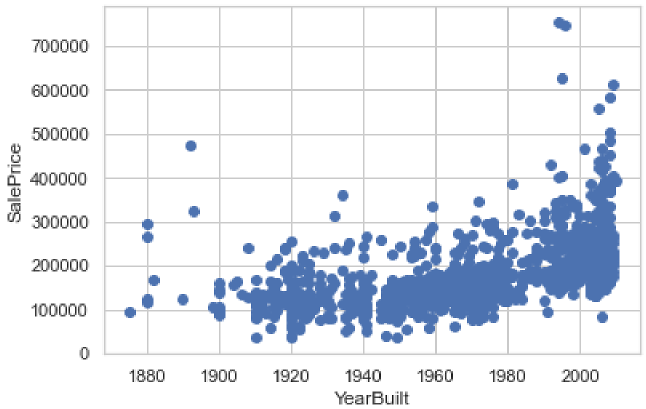
700000
600000
500000
400000
300000
200000
100000
0
SalePrice
50
100
150
200
250
300
LotFrontage

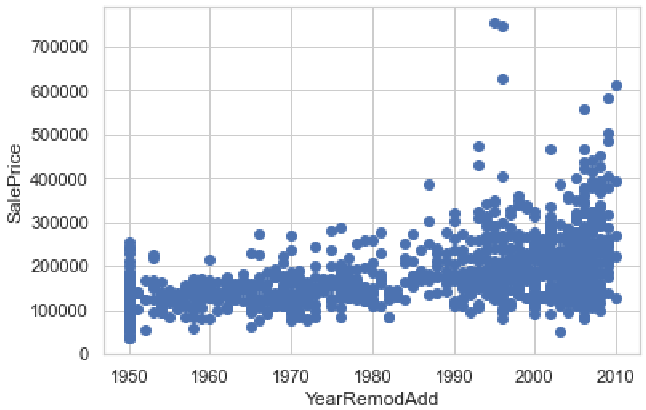


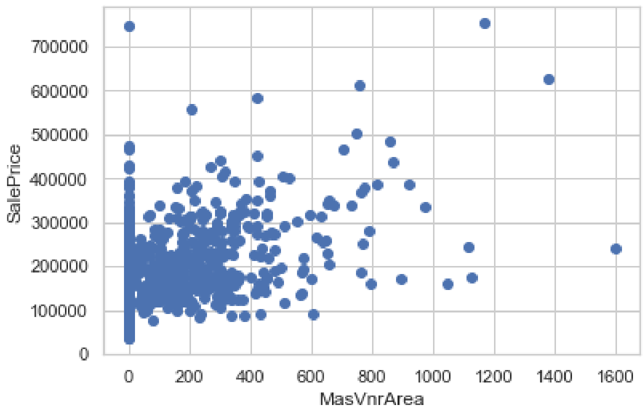
700000
600000
500000
400000
300000
200000
100000
0
SalePrice
0
20000
40000
60000
80000
100000
120000
140000
160000
LotArea

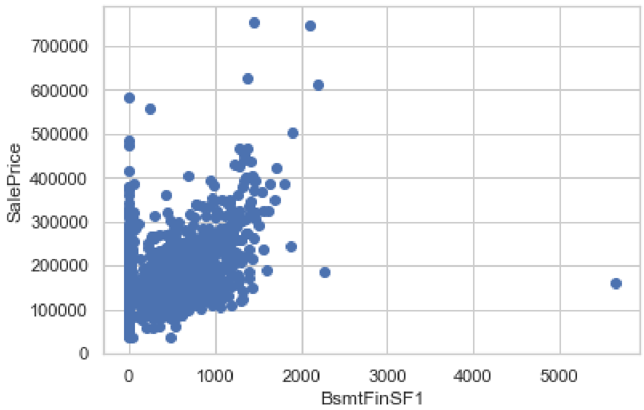

700000
600000
500000
400000
300000
200000
100000
0
SalePrice
2
4
6
8
10
OverallQual

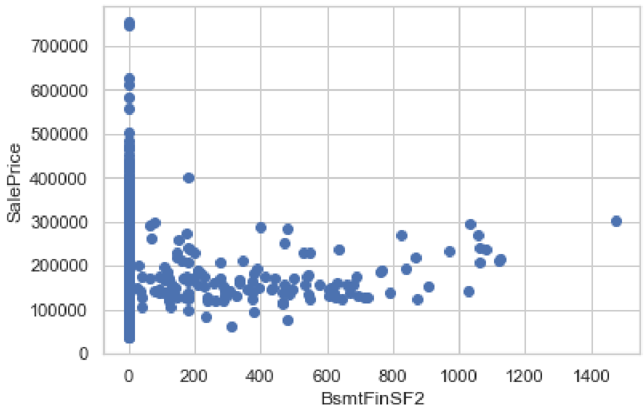

700000
600000
500000
400000
300000
200000
100000
0
SalePrice
1
2
3
4
5
6
7
8
9
OverallCond

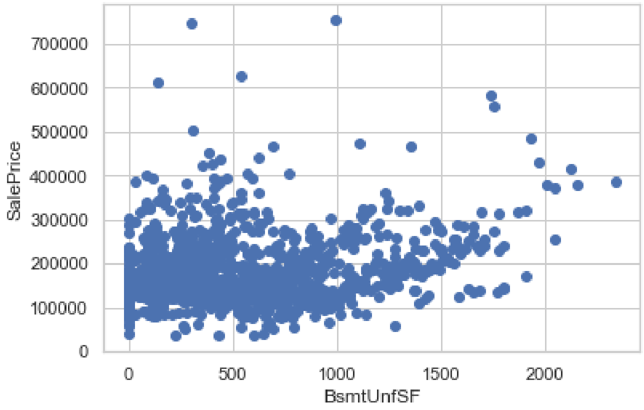

700000
600000
500000
400000
300000
200000
100000
0
SalePrice
1880
1900
1920
1940
1960
1980
2000
YearBuilt

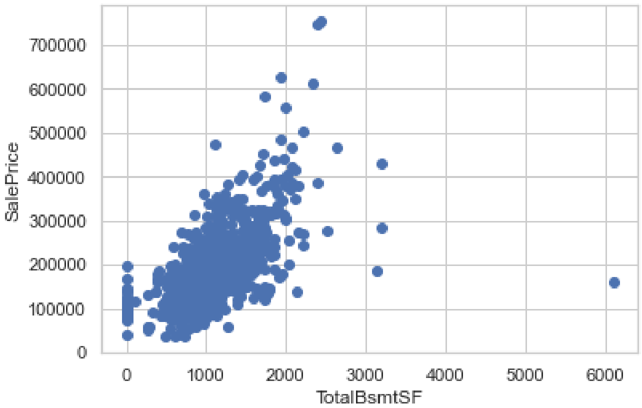

700000
600000
500000
400000
300000
200000
100000
0
SalePrice
1950
1960
1970
1980
1990
2000
2010
YearRemodAdd

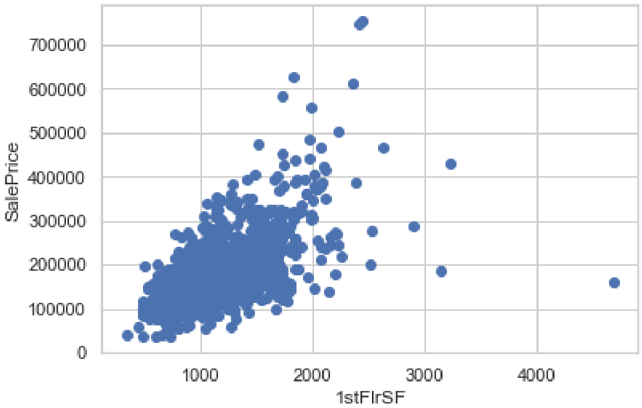

SalePrice
MasVnrArea

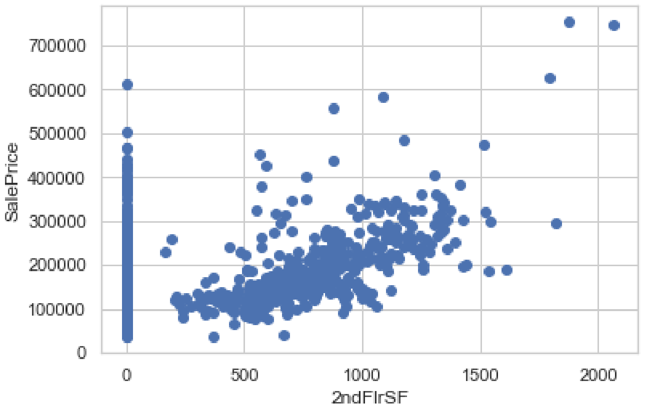

SalePrice
BsmtFinSF1

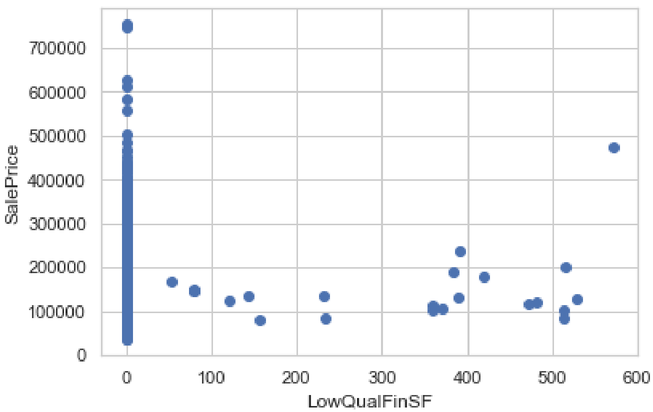

SalePrice
BsmtFinSF2

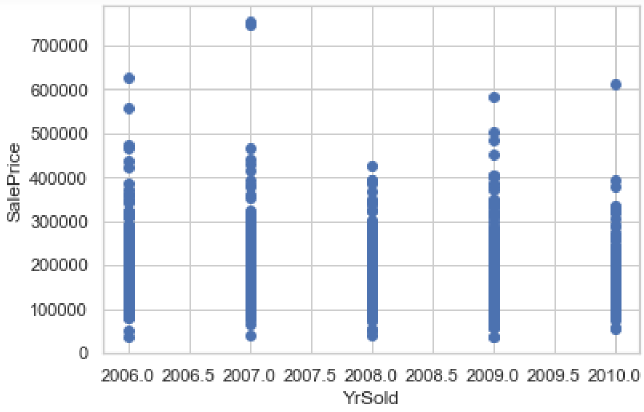

SalePrice
BsmtUnfSF

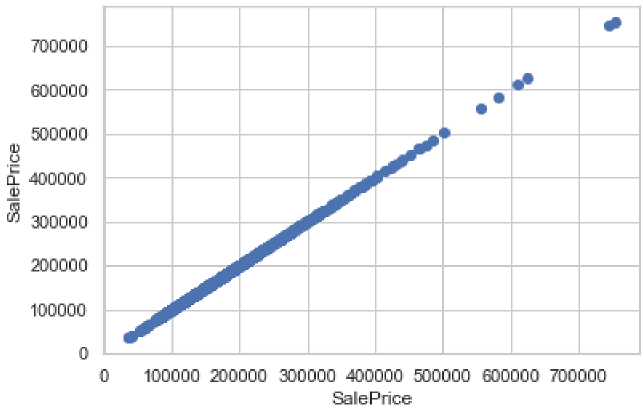

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
1000
2000
3000
4000
5000
6000
TotalBsmtSF

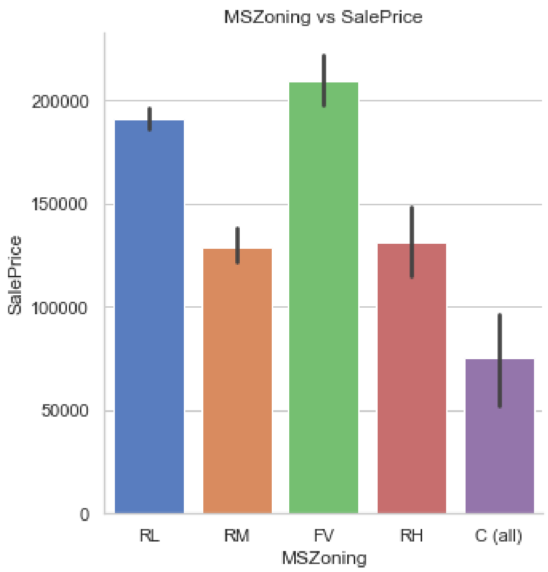

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
1000
2000
3000
4000
1stFlrSF


SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
500
1000
1500
2000
2ndFlrSF


SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
100
200
300
400
500
600
LowQualFinSF

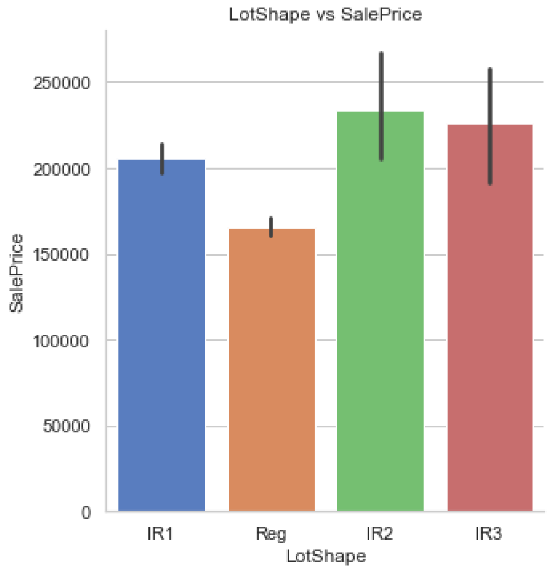






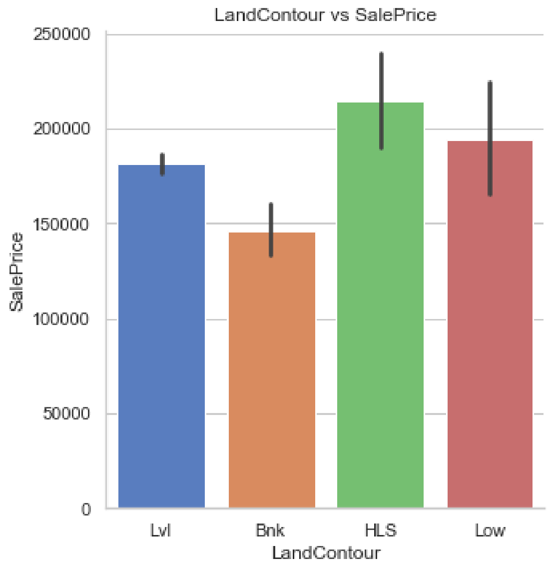
Observation:

SalePrice is maximum with FV MSZOning.



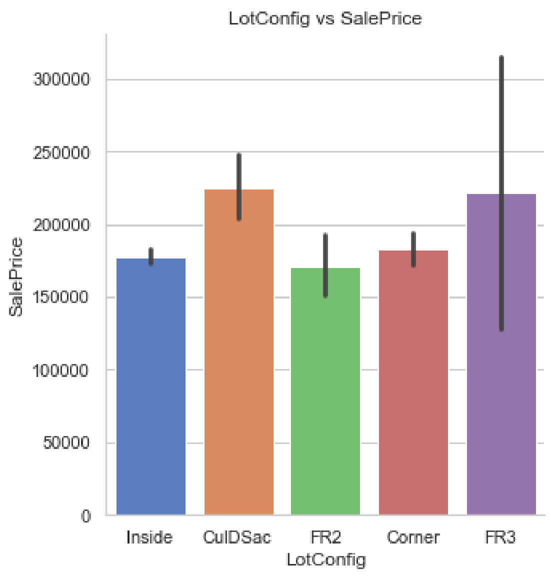
Observation:

SalePrice is maximum with IR2 LotShape.



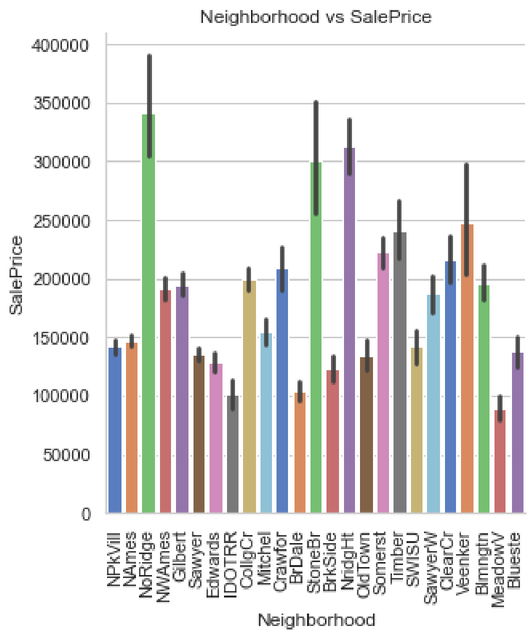
Observation:

SalePrice is maximum with HLS LandContour.



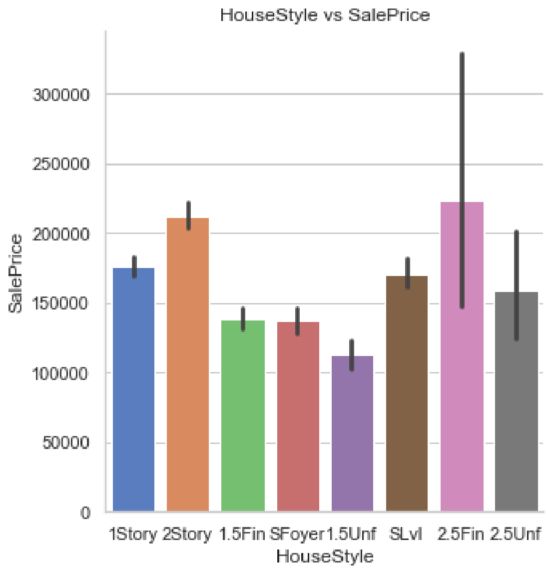
Observation:

SalePrice is maximum with CulDsac LotConfig.



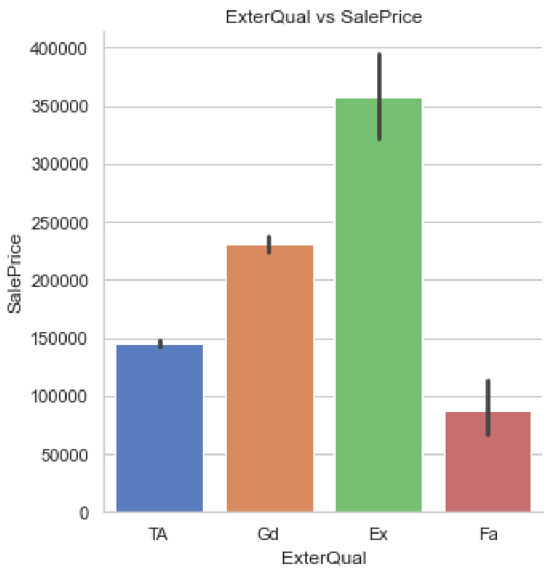
Observation:

SalePrice is maximum with NoRidge Neighborhood.



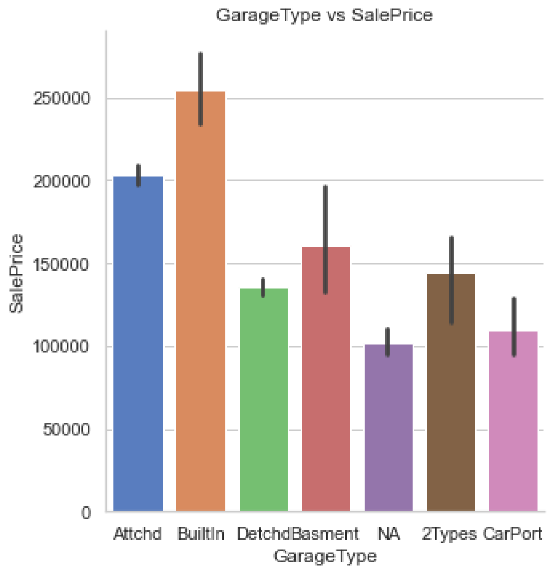
Observation:

SalePrice is maximum with 2.5Fin HouseStyle.



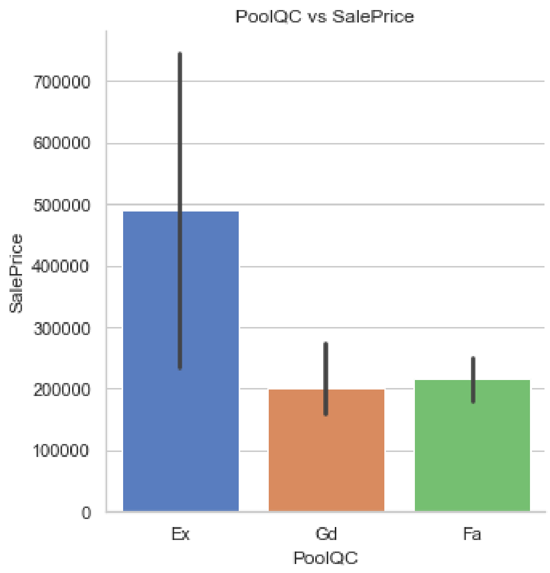
Observation:

SalePrice is maximum with Ex ExterQual.



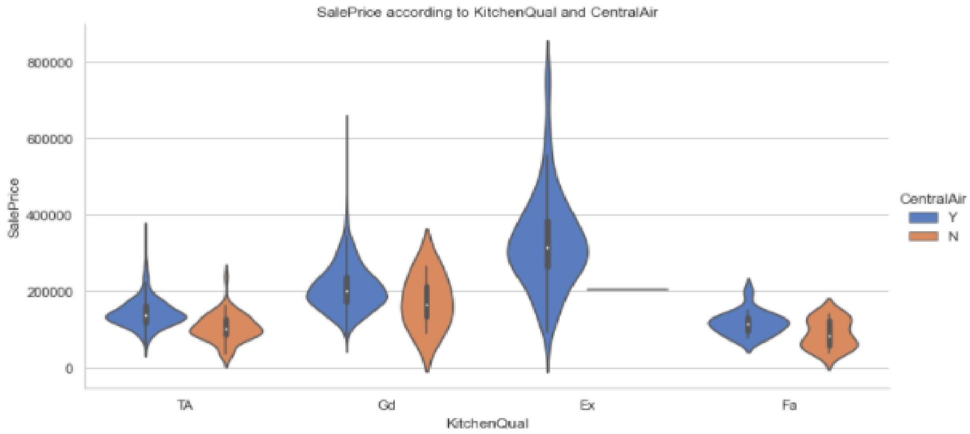
Observation:

SalePrice is maximum with Builtin GarageType.



Observation:

SalePrice is maximum with Ex PoolQC.



Observation:

SalePrice is maximum with Ex kitchenQual and CentralAir.

INTERPRETATION OF THE RESULTS

From the visualization we interpreted that the target variable SalePrice was highly positively correlated with the columns GrLivArea, YearBuilt, OverallQual, GarageCars, GarageArea.

From the preprocessing we interpreted that data was improper scaled.

```
#HYPERPARAMETER TUNING
# Let's Use the GridSearchCV to find the best paarameters in Gradientboosting Regressor

parameters={'learning_rate': [1.0,0.5,0.25,0.1,0.05,0.01],
           'n_estimators': [1,2,4,8,16,32,64,100]
           }

gb=GradientBoostingRegressor()

reg=GridSearchCV(gb,parameters)
reg.fit(x,y)
print(reg.best_params_)
```

```
{'learning_rate': 0.25, 'n_estimators': 64}
```

```
# Let's use the GradientBoosting Regressor with its best parameters

GB = GradientBoostingRegressor(learning_rate=0.25,n_estimators = 64)
GB.fit(x_train,y_train)
print('Score:',GB.score(x_train,y_train))
y_pred=GB.predict(x_test)
print('\n')
print('Mean absolute error:',mean_absolute_error(y_test,y_pred))
print('Mean squared error:',mean_squared_error(y_test,y_pred))
print('Root Mean Squared error:',np.sqrt(mean_squared_error(y_test,y_pred)))
print('\n')
print("r2_score:",r2_score(y_test,y_pred))
print('\n')
```

```
Score: 0.9851173738029483


Mean absolute error: 18578.499045764143
Mean squared error: 649334285.8556473
Root Mean Squared error: 25482.038494901608
```

From the modeling we interpreted that after hyperparameter tuning gradient Boosting regressor works best with respect to our model with minimum RMSE of 25482.0384949.

CONCLUSION

KEY FINDINGS AND CONCLUSIONS OF THE STUDY

In this project we have tried to show how the house prices vary and what are the factors related to the changing of house prices. The best(minimum) RMSE score was achieved using the best parameters of Gradient boosting Regressor through GridSearchCV though Lasso Regressor model performed well too.

LEARNING OUTCOMES OF THE STUDY IN RESPECT OF DATA SCIENCE

This project has demonstrated the importance of sampling effectively, modelling and predicting data.

Through different powerful tools of visualization we were able to analyse and interpret different hidden insights about the data.

Through data cleaning we were able to remove unnecessary columns and outliers from our dataset due to which our model would have suffered from overfitting or underfitting.

The few challenges while working on this project where:-

- Improper scaling
- Too many features
- Missing values
- Skewed data due to outliers

The data was improper scaled so we scaled it to a single scale using sklearns's package StandardScaler.

There were too many(256) features present in the data so we applied Principal Component Analysis(PCA) and we were able able to reduce our features upto 90 columns.

There were lot of missing values present in different columns which we imputed on the basis of our understanding.

The columns were skewed due to presence of outliers which we handled through winsorization technique.